INTEGRASI METODE *ARTIFICIAL NEURAL NETWORK* (ANN) DENGAN *BIDIRECTIONAL PARTICLE SWARM OPTIMIZATION* (BPSO) UNTUK OPTIMASI DOSIS PUPUK PADA TANAMAN PALAWIJA

SKRIPSI

KONSENTRASI KOMPUTASI DAN SISTEM CERDAS

Untuk memenuhi sebagian persyaratan meraihgelar Sarjana Komputer

Disusun Oleh :

EUNIKE ENDARIAHNA SURBAKTI

NIM. 115060801111081

KEMENTRIAN RISET TEKNOLOGI PENDIDIKAN TINGGI

UNIVERSITAS BRAWIJAYA

PROGRAM STUDI INFORMATIKA/ ILMU KOMPUTER

PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER

MALANG

2015

INTEGRASI METODE *ARTIFICIAL NEURAL NETWORK* (ANN) DENGAN *BIDIRECTIONAL PARTICLE SWARM OPTIMIZATION* (BPSO) UNTUK OPTIMASI DOSIS PUPUK PADA TANAMAN PALAWIJA

SKRIPSI

KONSENTRASI KOMPUTASI DAN SISTEM CERDAS

Untuk memenuhi sebagian persyaratan meraihgelar Sarjana Komputer

Disusun Oleh :

EUNIKE ENDARIAHNA SURBAKTI

NIM. 115060801111081

KEMENTRIAN RISET TEKNOLOGI PENDIDIKAN TINGGI

UNIVERSITAS BRAWIJAYA

PROGRAM STUDI INFORMATIKA/ ILMU KOMPUTER

PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER

MALANG

2015

LEMBAR PERSETUJUAN

INTEGRASI METODE *ARTIFICIAL NEURAL NETWORK* (ANN) DENGAN *BIDIRECTIONAL PARTICLE SWARM OPTIMIZATION* (BPSO) UNTUK OPTIMASI DOSIS PUPUK PADA TANAMAN PALAWIJA

SKRIPSI

KONSENTRASI KOMPUTASI DAN SISTEM CERDAS

Untuk memenuhi sebagian persyaratan meraihgelar Sarjana Komputer

Disusun Oleh :

EUNIKE ENDARIAHNA SURBAKTI

NIM. 115060801111081

Telah diperiksa dan disetujui oleh :

Dosen Pembimbing I,	Dosen Pembimbing II,
Imam Cholissodin, S.Si.,M.Kom	**Candra Dewi, S.Kom.,M.Sc**
NIK. 85071916110422	**NIP. 197711142003122 001**

PERNYATAAN

ORISINALITAS SKRIPSI

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah SKRIPSI ini, tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila ternyata di dalam naskah SKRIPSI ini dapat dibuktikan terdapat unsur-unsur PLAGIASI, saya bersedia SKRIPSI ini digugurkan dan gelar akademik yang saya peroleh (SARJANA) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 11 November 2015

Mahasiswa,

Eunike Endariahna Surbakti
NIM. 115060801111081

KATA PENGANTAR

Terimakasih kepada Tuhan Yesus yang telah memberikan anugerah dan kasihNya, sehingga penulis dapat menyelesaikan skripsi dengan judul "**Integrasi Metode *Artificial Neural Network* (ANN) dengan *Bidirectional Particle Swarm Optimization* (BPSO) untuk Optimasi Dosis Pupuk pada Tanaman Palawija*"***. Skripsi ini disusun dan diajukan sebagai syarat untuk memperoleh gelar sarjana pada Program Studi Informatika/ Ilmu Komputer, Program teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya.

Penulis menyadari bahwa skripsi ini tidak akan dapat diselesaikan dengan baik tanpa keterlibatan dari berbagai pihak. Untuk itu, penulis menyampaikan ucapan terimakasih yang sebesar-besarnya kepada :

1. Imam Cholissodin, S.Si,.M.Kom., selaku dosen pembimbing utama yang telah memberikan bimbingan dan masukan dalam penulisan skripsi ini.
2. Candra Dewi, S.Kom,.M.Sc., selaku dosen pembimbing kedua yang telah memberikan bimbingan dan masukan dalam penulisan skripsi ini.
3. Ir.Sutrisno, M.T, Ir. Heru Nurwasito, M. Kom, Himawat Aryadita, ST., Msc, dan Edy Santoso, S.Si., M.Kom selaku Ketua, Wakil Ketua I, Wakil Ketua II, dan Wakil Ketua III Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya.
4. Drs. Mardji, MT dan Issa Arwani, S.Kom., M.Sc selaku Ketua dan Sekertaris Program Studi Informatika dan Ilmu Komputer Universitas Brawijaya.
5. Aswin Suharsono, S.T,.M.T., selaku Dosen Penasehat Akademik.
6. Segenap Bapak dan Ibu dosen atas kesediaan membagi ilmu kepada penulis selama menempuh pendidikan di Program Studi Informatika dan Ilmu Komputer Universitas Brawijaya.
7. Segenap staf dan karyawan yang membantu penulis selama proses penyelesaian skripsi ini.
8. Kedua orang tua Jantianus Surbakti dan Ingan Malem Tarigan serta adik Esekiel Mako Haganta Surbakti dan Hosea Sinalsal Surbakti yang telah memberikan doa, nasehat dan semangat kepada penulis.

9. Kepada Bpk/Ibu Gembala dan jemaat GPdI Kristus Jawaban Medan dan Bpk/Ibu Gembala dan jemaat GPdI Gloria Malang yang sudah menjadi keluarga kedua bagi penulis.
10. Teman-teman sepelayanan saya di Bright Generation Om. Harry, Cindy. Kezia, Meiske, Tinus, Rere dan lainnya yang telah menolong, menegur dan memberikan semangat kepada penulis agar tidak menyerah terus maju.
11. Teman-teman TIF 2011, terutama teman-teman TIF kelas I 2011 mbak Ayu, Ifa, Jasicka, Ami, Dyah, Tika, dan lainnya yang telah memberikan dorongan dan bantuan dalam berbagai hal selaman menempuh pendidikan di Universitas Brawijaya.
12. Teman-teman asrama Griya Brawijaya B.301-B.305 Dina, Fatma, Atina, Tyas, Ella, Yaya, Rara, Seli, Yuni dan lainnya yang telah berbagi dalam setiap hal kepada penulis.

Tidak ada manusia yang sempurna,maka dari itupenulis sangat mengharapkan saran dan kritik yang membangun dari pembaca sehingga di kesempatan berikutnya penulis dapat meyusun laporan yang lebih baik lagi. Penulis berharap agar skripsi ini memberikan manfaat bagi *civitas academica* Teknik Informatika khususnya, Universitas Brawijaya dan semua pihak yang berkepentingan.

Malang, 11 November 2015

Eunike Endariahna Surbakti
NIM. 115060801111081

ABSTRAK

Eunike Endariahna Surbakti, 2015. Integrasi Metode *Artificial Neural Network* (ANN) dengan *Bidirectional Particle Swarm Optimization* (BPSO) untuk Optimasi Dosis Pupuk pada Tanaman Palawija. Skripsi Program Studi Informatika/ Ilmu Komputer, Fakultas Ilmu Komputer Universitas Brawijaya. Pembimbing : Imam Cholissodin, S.Si.,M.Kom dan Candra Dewi, S.Kom., M.Sc.

Kemajuan Ilmu Pengetahuan dan Teknologi (IPTEK) semakin pesat terutama dalam bidang pertanian. Salah satu aspek dalam pertanian yang sangat diperhatikan adalah pemupukan. Dalam pemberian pupuk melibatkan banyak jenis pupuk dan kombinasi dosis yang berbeda. Palawija merupakan tanaman untuk rotasi tanam, saat tidak musim padi. Palawija juga ditanaman di dataran tinggi dimana padi tidak tumbuh. Pemberian pupuk memberikan dampak berbeda bagi Palawija. Metode ANN dapat digunakan untuk mengetahui pengaruh pada tanaman yang ditimbulkan dari pemberian pupuk. Selanjutnya, dipilih dua pengaruh pada tanaman untuk dioptimasi dosis pupuknya menggunakan metode BPSO. Metode ANN terbukti sangat baik dalam memprediksi nilai karena proses pembelajaran yg dilakukan dan BPSO mampu mengoptimasi dengan vektor lebih dari satu sehingga mempercepat proses sistem. Dari hasil pengujian yang dilakukan parameterANN*backpropagation*menggunakan data latih 90%, data uji 10%, iterasi 100 kali, jumlah *hiddenlayer*10, *learning rate* 0.6 dan momentum 0.6 dengan didapatkan nilai MSE terkecil 8.6023E-03.Untuk Parameter BPSO menggunakan nilai standard pada PSO (*particle Swarm Optimization*) pada penelitian Navalertporn (2011). Sehingga untuk mendapatkan bobot kering tanaman 4.4964 ton/ha dan hasil produksi 6.99985 ton/ha maka dibutuhkan pupuk Urea 0.191 ton/ha atau 191 kg/ha, SP_{36} 0.201 ton/ha atau 201 kg/ha, KCL 0.288 ton/ha atau 288 kg/ha dan Bioncah 48.3 ton/ha.

Kata Kunci : *Artificial Neural Network, Bidirectional Particle Swarm Optimization, Optimasi Dosis Pupuk, Pengaruh pada tanaman Palawija.*

ABSTRACT

Advancement of Science and Technology growing so fast, especially in the field of agriculture. Fertilization is one of aspect that very considerables. In the fertilizer application involves many types of fertilizer and combination of different doses. Palawija are plants for crop rotation, when not the rice season. Palawija also crops in the highlands where rice isn't grow. Fertilizer application can give influence of different impacts for Palawija. ANN method can be used to determine the effect on the plants arising from fertilizer application. Next, user input two of effect on crops selected for optimization doses of fertilizer using BPSO. ANN method proved to be very good at predicting the value from learning process andBPSO is able to optimize the vectormore than one so as to quickly the process of the system. From the test results, parameters ANN using training data of 90%, 10% test data, iterating 100 times, the number of hiddenlayer 10, learning rate of 0.6 and momentum 0.6 with the smallest MSE value obtained 8.6023E-03. The parameter values of BPSO use standard on PSO (Particle Swarm Optimization) from research had been done by Navalertporn(2011). So as to get the plant dry weight 4.4964 ton / ha and yield 6.99985 ton / ha is needed Urea 0191 ton / ha or 191 kg / ha, SP_{36} 0201 ton / ha or 201 kg / ha, KCL 0288 ton / ha or 288 kg / ha and Bioncah 48.3 ton / ha.

Keywords: *Artificial Neural Network, Bidirectional Particle Swarm Optimization, Optimization Dose Fertilizer, Effect on Palawija plants.*

DAFTAR ISI

KATA PENGANTAR iv
ABSTRAK vi
DAFTAR ISI viii
DAFTAR GAMBAR xii
DAFTAR TABEL xiv
DAFTAR PERSAMAAN xvii
DAFTAR LAMPIRAN xix
BAB I 1
1.1Latar Belakang 1
1.2 Rumusan Masalah 3
1.3 Batasan Masalah 3
1.4Tujuan 3
1.5Manfaat 3
1.6 Sistematika Penulisan 4
BAB II 6
2.1Kajian Pustaka 6
2.2 Tanaman Palawija 10
2.3 Jagung 11
2.3.1 Ciri Morfologis Tanaman Jagung 12
2.3.2 Deskripsi Jagung Hibrida 14
2.4Lahan Kering 15
2.5Pupuk dan Jenisnya 16
2.5.1 Urea $CO(NH_2)_2$ 17
2.5.2 SP_{36} 18
2.5.3KCL 19
2.5.4 Biochar 21
2.6Jaringan Syaraf Tiruan 23
2.6.1Kelebihan JST 23
2.6.2Model Matematika 24
2.6.3Fungsi Aktivasi 25
2.6.4 Backpropagation 28

2.6.5Normalisasi Data 31
2.6.6Denormalisasi 32
2.6.7Jumlah *HiddenLayer* 32
2.6.8*Mean Square Error*(MSE) 33
2.7 *Particle Swarm Optimization* (PSO) 353
2.8*Bidirectional Particle Swarm Optimization* (BPSO) 35
BAB III 40
3.1 Tahapan Penelitian 40
3.1.1 Studi Literatur 41
3.1.2 Pengumpulan Data 41
3.1.3 Analisis Kebutuhan Sistem 41
3.1.4 Perancangan Sistem 43
3.1.5 Implementasi Sistem 44
3.1.6 Pengujian 45
3.1.7 Kesimpulan dan Saran 45
3.2 Alir Perancangan Sistem 46
3.2.1 Perhitungan Pengaruh Pemberian Dosis Menggunakan ANN 47
3.2.2 Perhitungan Optimasi Pengaruh pada Tanaman dengan BPSO 56
3.3 Perhitungan Manual ANN *Backpropagation* 68
3.3.1 Analisis Permasalahan 68
3.3.2 Proses Normalisasi Data latih dan Data uji 70
3.3.3 Perhitungan Pelatihan Data Latih ANN *Backpropagation* 72
3.3.4 Perhitungan Pengujian Data Uji ANN *Backpropagation* 91
3.3.5 Proses Denormalisasi 95
3.4 Perhitungan Manual BPSO (Bidirectional Particle Swarm Optimization) . 95
3.4.1 Menentukan Parameter BPSO 96
3.4.2 Inisialisasi Populasi Partikel dan GBEST 96
3.4.3 Menghasilkan Keturunan $Child_1$ dan $Child_2$ 98
3.4.4 GBEST dan PBEST untuk $child_1$ 98
3.4.5 GBEST dan PBEST untuk $child_2$ 99
3.4.6 Menghitung Kecepatan $child_1$ dan $child_2$ 100
3.4.7 Menghitung Posisi baru $child_1$ dan $child_2$ 101

3.4.8 Memperbaharui GBEST 102
3.4.9 Perbaharui Partikel 102
3.5 Perancangan *User Interface* 104
3.5.1 Proses Pelatihan ANN 104
3.5.2Proses Pengujian ANN 106
3.5.3 Proses Optimasi BPSO 107
3.5.4 Proses Detail Optimasi BPSO 108
3.6 Perancangan Uji Coba dan Evaluasi 110
3.6.1 Pengujian terhadap Variasi Data Latih dan Data Uji 110
3.6.2 Pengujian Jumlah Iterasi 110
3.6.3 Pengujian Variasi Jumlah *HiddenLayer* 111
3.6.4 Pengujian Variasi Nilai *Learning Rate* dan Momentum 112
BAB IV 113
4.1 Lingkungan Implementasi 113
4.1.1Lingkungan Perangkat Keras 113
4.1.2Lingkungan Perangkat Lunak 113
4.2Batasan Implementasi 114
4.3 Implementasi Program 114
4.3.1 Proses Normalisasi 114
4.3.2 Proses Pelatihan ANN *Backpropagation* 115
4.3.3 Proses Pengujian ANN *Backpropagation* 120
4.3.4 Proses Denormalisasi 121
4.3.5 Optimasi BPSO (*Bidirectional Particle Swarm Optimization*) 122
4.4 Implementasi Antarmuka 131
4.4.1 Antarmuka Hasil Pelatihan *Backpropagation* 132
4.4.2 Antarmuka Hasil Pengujian *Backpropagation* 132
4.4.3 Antarmuka Hasil Optimasi BPSO 133
4.4.4 Antarmuka Hasil Detail Optimasi BPSO 134
BAB V 136
5.1Pengujian 136
5.2 Hasil Pengujian 136
5.2.1 Hasil Pengujian Terhadap Variasi Data Latih dan Data Uji 136

5.2.2 Hasil Pengujian Variasi Jumlah Itrasi 137
5.2.3 Hasil Pengujian Variasi Nilai *HiddenLayer* 138
5.2.4 Hasil Pengujian Variasi Nilai *Learning Rate* dan Momentum 139
5.3 Analisi Hasil Pengujian 140
5.3.1 Analisis Hasil Pengujian Terhadap Perbandingan Jumlah Data Latih140
5.3.2 Analisis Hasil Pengujian Terhadap Perbandingan Jumlah Iterasi..... 141
5.3.3 Analisis Hasil Pengujian Terhadap Variasi *HiddenLayer* 143
5.3.4 Analisis Hasil Pengujian Nilai *Learning Rate* dan Momentum..... 144
BAB VI 145
6.1 Kesimpulan 145
6.2 Saran 146
DAFTAR PUSTAKA 147
LAMPIRAN 14750

DAFTAR GAMBAR

Gambar 2.1 Hasil Palawija 10
Gambar 2.2 Lahan Pertanian Jagung Hibrida 11
Gambar 2.3 Tanah Lahan Kering 15
Gambar 2.4 Jenis-jenis pupuk anorganik 20
Gambar 2.5 Biochar 22
Gambar 2.6 Proses Pembuatan Biochar dengan Pirolisis 22
Gambar 2.7 Model Matematis dari JST 24
Gambar 2.8 Grafik Fungsi *Hard Limit* 25
Gambar 2.9 Grafik Fungsi *Threshold* 25
Gambar 2.10 Grafik Fungsi Bipolar *Threshold* 26
Gambar 2.11 Grafik Fungsi Sigmoid Niner 26
Gambar 2.12 Grafik Fungsi Identitas 27
Gambar 2.13 Arsitektur *Backpropagation* 28
Gambar 3.1 Metode Penelitian 40
Gambar 3.2 Diagram Blok Sistem 44
Gambar 3.3 Diagram Alir Perancangan Sistem 46
Gambar 3.4 Diagram Alir Proses ANN *Backpropagation* 47
Gambar 3.5 Diagram Alir Proses Normalisasi 49
Gambar 3.6 Diagram Alir Proses Pelatihan *Backpropagation* 50
Gambar 3.7 Diagram Alir *Feedforward* 52
Gambar 3.8 Diagram Alir *Backpropagation* 53
Gambar 3.9 Diagram Alir Proses Pengujian *Backpropagation* 54
Gambar 3.10 Diagram Alir Proses Denormalisasi 56
Gambar 3.11 Diagram Alir Perhitungan Optimasi BPSO 58
Gambar 3.12 Diagram Alir Perhitungan Vektor 600
Gambar 3.13 Diagram Alir Menghitung PBEST dan GBEST untuk $child_1$ 62
Gambar 3.14 Diagram Alir Menghitung PBEST dan GBEST untuk $child_2$ 64
Gambar 3.15 Diagram Alir *Update* GBEST 65
Gambar 3.16 Diagram Alir *Update* Posisi Partikel 67
Gambar 3.17 Arsitektur *Backpropagation* Sistem yang Dibangun 69
Gambar 3.18 Rancang *Interface* Proses Pelatihan *Backpropagation* ANN 105

Gambar 3.19 Rancang *Interface* Proses Pengujian ANN 106
Gambar 3.20 Rancangan *Interface* Porses Optimasi BPSO 108
Gambar 4.1*source code* Proses Normalisasi 115
Gambar 4.2*source code* Proses *Feedforward* 118
Gambar 4.3*source code* Proses *Backpropagation* 120
Gambar 4.4*source code* Proses Pengujian ANN *Backpropagation* 121
Gambar 4.5 *source code* Proses Denormalisasi 121
Gambar 4.6*source code* Proses Denormalisasi 122
Gambar 4.7*source code* ProsesMembangkitkan Partikel dan GBEST 123
Gambar 4.8*source code* Proses Memilih GBEST dan PBEST $child_1$ 125
Gambar 4.9*source code* Proses Memilih GBEST dan PBEST $child_2$ 126
Gambar 4.10*source code*Proses Menghitung Kecepatan dan Posisi Baru 127
Gambar 4.11*source code* Proses Perbaharui Partikel dan GBEST 131
Gambar 4.12 Antarmuka Hasil Pelatihan *Backpropagation* 132
Gambar 4.13 Antarmuka Hasil Pengujian *Backpropagation* 133
Gambar 4.14 Antarmuka Optimasi BPSO 134
Gambar 4.15 Antarmuka Detail Optimasi BPSO 135
Gambar 5.1 Grafik Hasil Pengujian Variasi Data Latih dan Data Uji 141
Gambar 5.2 Grafik Hasil Pengujian Variasi Jumlah Iterasi 142
Gambar 5.3 Grafik Hasil Pengujian Variasi *HiddenLayer* 143
Gambar 5.4 Grafik Hasil Pengujian Variasi *Learnig Rate* dan Momentum 144

DAFTAR TABEL

Tabel 2.1 Perbandingan Obyek dan Metode 7
Tabel 2.2 Perbandingan *Input*, Proses dan *Output* 8
Tabel 2.3 Deskripsi Jagung Hibrida 14
Tabel 2.4 Rekomendasi Pemupukan N 18
Tabel 2.5 Rekomendasi Pemupukan P 19
Tabel 2.6 Rekomendasi Pemupukan K 20
Tabel 2.7 Hasil Analisis Bahan Arang (Biochar) dari Limbah Pertanian 21
Tabel 3.1 Analisis Kebutuhan Data Penelitian 43
Tabel 3.2 Data Latih sebelum Normalisasi 70
Tabel 3.3 Data Uji sebelum Normalisasi 70
Tabel 3.4 Data Latih Normalisasi 71
Tabel 3.5 Data Uji Normalisasi 711
Tabel 3.6 Data K3 Normalisasi 72
Tabel 3.7 Bobot *Input* ke *Hiddenlayer* 72
Tabel 3.8 Jumlah Z_in_j bobot sinyal *input* 73
Tabel 3.9 Hasil Perhitungan sinyal *Input*dengan Fungsi *Sigmoid Biner* 74
Tabel 3.10 Bobot *Hiddenlayer* ke *Outputlayer* 74
Tabel 3.11 Jumlah Y_in_k bobot sinyal *hidden* ke *output* 75
Tabel 3.12 Hasil Perhitungan sinyal *output*dengan Fungsi *Sigmoid Biner* 76
Tabel 3.13 Target data ke-1 setelah normalisasi 77
Tabel 3.14*Errot Output* ke Target 77
Tabel 3.15 Perubahan Bobot 78
Tabel 3.16 Jumlah delta unit tersembunyi ke *output* 78
Tabel 3.17*Error* pada *HiddenLayer* 79
Tabel 3.18 Perubahan Bobot *HiddenLayer* 80
Tabel 3.19*Error* pada iterasi pertama 80
Tabel 3.20 Bobot baru pada unit *output* 81
Tabel 3.21 Bobot baru pada unit *hidden* 81
Tabel 3.22 Bobot*Input* ke *Hiddenlayer*iterasi kedua 82
Tabel 3.23 Jumlah Z_in_j bobot sinyal *input*iterasi kedua 83
Tabel 3.24 Hasil Perhitungan sinyal *Input*iterasi kedua 84

Tabel 3.25 Bobot *Outputlayer* iterasi kedua 84
Tabel 3.26 Jumlah Y_in_k iterasi kedua 85
Tabel 3.27 Hasil Perhitungan sinyal *output*iterasi kedua 86
Tabel 3.28 *Errot Output* ke Target iterasi kedua 87
Tabel 3.29 Perubahan Bobot *Output*iterasi kedua 87
Tabel 3.30 Jumlah delta unit tersembunyi ke *output*iterasi kedua 88
Tabel 3.31*Error* pada *HiddenLayer*iterasi kedua 89
Tabel 3.32 Perubahan Bobot *HiddenLayer* iterasi kedua 90
Tabel 3.33*Error* pada iterasi kedua 900
Tabel 3.34 Bobot baru pada unit *output*iterasi kedua 91
Tabel 3.35 Bobot baru pada unit *hidden* iterasi kedua 91
Tabel 3.36 Bobot*Input* ke *Hiddenlayer*Data Uji 92
Tabel 3.37 Jumlah Z_in_j data uji 92
Tabel 3.38 Hasil Perhitungan sinyal *Input*Data Uji 93
Tabel 3.39 Bobot *Outputlayer* Data uji 93
Tabel 3.40 Jumlah Y_in_k Data Uji 94
Tabel 3.41 Hasil Perhitungan sinyal *output*Data Uji 95
Tabel 3.42 Hasil Denormalisasi Data Uji 95
Tabel 3.43 Populasi Partikel Awal 97
Tabel 3.44 Populasi untuk GBEST 98
Tabel 3.45 NC Populasi GBEST 98
Tabel 3.46 GBEST dan PBEST untuk $child_1$ 99
Tabel 3.47*euclidean distance* untuk $child_2$ 99
Tabel 3.48 GBEST dan PBEST untuk $child_2$ 100
Tabel 3.49 r_1,r_2 untuk $child_1$ dan $child_2$ 100
Tabel 3.50 Perubahan Kecepatan $child_1$ dan $child_2$ 100
Tabel 3.51 Posisi Baru $child_1$ 101
Tabel 3.52 Posisi Baru $child_2$ 101
Tabel 3.53 Memperbaharui GBEST untuk a_1 102
Tabel 3.54 Perbaharui Partikel 102
Tabel 3.55 Perbaharui GBEST untuk a_5 103
Tabel 3.56 Rancang Pengujian Variasi Data Latih dan Data Uji 110

Tabel 3.57 Rancang Pengujian Jumlah Iterasi 111
Tabel 3.58 Pengujian Variasi Jumlah *HiddenLayer* 111
Tabel 3.59 Pengujian Variasi Nilai *Learning Rate* dan Momentum 112
Tabel 5.1 Hasil Pengujian Variasi Data Latih dan Data Uji 137
Tabel 5.2 Hasil Pengujian Variasi Jumlah Iterasi 138
Tabel 5.3 Hasil Pengujian Variasi Nilai *HiddenLayer* 139
Tabel 5.4 Hasil Pengujian Nilai *HiddenLayer* dan momentum 140

DAFTAR PERSAMAAN

Persamaan(2-1) Fungsi *Hard Limit* 25
Persamaan (2-2) Fungsi *Threshold* 25
Persamaan (2-3) Fungsi Bipolar *Threshold* 26
Persamaan (2-4) Fungsi Sigmoid Biner 26
Persamaan (2-5) Menghitung Turunan Fungsi Sigmoid Biner 26
Persamaan (2-6) Fungsi Sigmoid Bipolar 27
Persamaan (2-7) Menghitung Turunan Fungsi Sigmoid Bipolar 27
Persamaan (2-8) Fungsi Identitas (*Linear*) 27
Persamaan (2-9) Menjumlahkan Bobot Sinyal*Input* 29
Persamaan(2-10)Menerapkan Fungsi Aktivasi pada *HiddenLayer* 29
Persamaan(2-11) Menjumlahkan Bobot Sinyal*Output* 30
Persamaan (2-12) Menerapkan Fungsi Aktivasi pada *OutputLayer* 30
Persamaan (2-13) Menghitung *Error* pada Target 30
Persamaan (2-14) Menghitung Perubahan Bobot*Output* dengan Momentum 30
Persamaan (2-15) Menjumlahkan Delta *Input* 30
Persamaan (2-16) Menghitung Informasi *Error* 31
Persamaan (2-17)Menghitung Perubahan Bobot *Input* dengan Momentum 31
Persamaan (2-18) Menghitung Bobot *Output*Baru 31
Persamaan (2-19)Menghitung Bobot *Input* Baru 31
Persamaan (2-20) Menghitung Normalisasi 32
Persamaan (2-21) Menghitung Proses Hasil Denormalisasi 32
Persamaan(2-22) Menghitung Jumlah *HiddenLayer* 32
Persamaan (2-23) Menghitung *Mean Square Error* 33
Persamaan (2-24) Menghitung Kecepatan Partikel 36
persamaan (2-25) Menghitung Vektor *Output* Pertama. 36
Persamaan (2-26)Menghitung Vektor *Output*Kedua 36
Persamaan (2-27) Menghitung Posisi awal $Child_1$dan *Child2* 36
Persamaan (2-28)Menghitung Kecepatan awal $Child_1$ dan $Child_2$ 36
Persamaan (2-29) Menghitung *Euclidean Distance* 37
Persamaan (2-30) Fungsi *Share* 37
Persamaan (2-31) Menghitung *Niche Radius* 37

Persamaan (2-32) Menghitung Jarak 37
Persamaan (2-33) Menghitung *Niche Count* 38
Persamaan (2-34) Menghitung Perubahan Kecepatan $Child_1$ dan $Child_2$ 38
Persamaan (2-35)Menghitung Perubahan Posisi $Child_1$dan $Child_2$ ..... **Error! Bookmark not defined.**
Persamaan (2-36) Memperbaharui Partikel 39
Persamaan (2-37)Memperbaharui PBEST 39
Persamaan(2-38) Mengubah nilai Vektor *Output* Pertama 39
Persamaan (2-39)Mengubah nilai Vektor *Output*Kedua 39

DAFTAR LAMPIRAN

LAMPIRAN 1 150
Hasil Wawancara 150
LAMPIRAN 2 151
Data Utama Kombinasi Dosis Pupuk dan Pengaruh pada Tanaman 151
LAMPIRAN 3 157
Iterasi Pertama 157
Bobot*inputlayer* ke *hiddenlayer* 157
Bobot *hiddenlayer* ke *outputlayer* 157
Perubahan Bobot*hiddenlayer* ke *outputlayer* 157
Perubahan Bobot *hiddenlayer* ke *inputlayer* 157
Iterasi Kedua 157
Bobot*inputlayer* ke *hiddenlayer* 157
Bobot *hiddenlayer* ke *outputlayer* 157
Perubahan Bobot*hiddenlayer* ke *outputlayer* 157
Perubahan Bobot *hiddenlayer* ke *inputlayer* 157

BAB I
PENDAHULUAN

1.1 Latar Belakang

Kemajuan teknologi yang pesat dalam sektor pertanian diikuti dengan pertumbuhan jumlah penduduk, berdampak pada kebutuhan akan pangan dan produk-produk pertanian lain terus meningkat [WIN-08]. Negara-negara sedang berkembang menghadapi masalah kuantitas pangan, sedangkan permasalahan yang dihadapi negara maju adalah kualitas bahan pangan (*food quality*), pencemaran lingkungan hidup (*pollution*) dan efisiensi biaya produksi agar daya saing produk tinggi. Inilah tantangan yang harus dihadapin dewasa ini dengan Ilmu Pengetahuan dan Teknologi (IPTEK) melalui riset-riset disegala aspek pertanian [WIJ-08].Salah satu aspek dalam peningkatan pertanian adalah pemupukan.

Pupuk dibutuhkan karena tanah sudah tidak mampu memenuhi unsur hara tanaman secara alami [WIN-08].Pemupukan juga dilakukan sebagai upaya untuk mencukupi kebutuhan produksi.Namun, apabila pemupukan dilakukan secara berlebihan akan menimbulkan dampak negatif bagi lingkungan hidup dan meningkatkan biaya produksi yang pada akhirnya akan menurunkan daya saing produk, maka pupuk harus digunakan secara bijaksana[WIJ-08]. Penggunaan pupuk secara bijaksana akan terwujud apabila tersedia teknologi dan pengetahuan yang memadai antara tanaman, unsur hara dan lingkungan hidup.Oleh sebab itu, pada pemupukan sangat penting memperhatikan pemberian dosis yang optimal.

Pemberian dosis yang optimal melibatkan banyak kombinasi dari jenis pupuk. Sangat penting untuk memperhatikan jenis pupuk yang aman bagi lingkungan. Kombinasi dari jenis pupuk menghasilkan dampak yang berbeda pada tanaman. Untuk mendapatkan kombinasi pupuk dengan hasil yang diharapkan dapat diselesaikan dengan algoritma *Particle Swarm Optimization* (PSO). Pada penelitian yang dilakukan Ria (2009), algoritma ini diterapkan pada

penjadwalansumber dayaproyek. Penjadwalan sumber daya (*resources scheduling*) adalah

proses menentukan jenis dan jumlah sumber daya yang dibutuhkansesuai dengan jadwal. Kebutuhan sumber daya pada setiap waktu berbeda-beda, sehingga terjadi fluktuasi kebutuhan sumber daya.Fluktuasi adalah keadaan naik-turun harga karena pengaruh permintaan dan penawaran.PSO digunakan untuk mengoptimasi penjadwalan sumber daya pada setiap aktivitas proyek agar diperoleh nilai fluktuasi terkecil.Hasil yang didapatkan PSO mampu menghasilkan solusi yang optimal karena menyeimbangkan antara eksploitasi dan eksplorasi.Eksploitasi adalah pencarian solusi terbaru dari solusi terbaik yang telah ada. Eksplorasi adalah mencari solusi pada parameter lainnya yang belum digunakan untuk prediksi ke depan[RIA-09].

Selain itu, pendekatan terbaruoleh Navalertporn(2011) dalam masalah optimasi adalah dengan integrasi Metode *Artificial Neural Network* (ANN) dengan *Bidirectional Particle Swarm Optimization* (BPSO).Algoritma ini digunakan dalam meningkatkan optimasi pembuatan atap dari semen. Dalam menentukan kualitas semen, dilakukan proses menekan dengan baja, masalahnya sulit menentukan kualitas atap semen yang dihasilkan.Karena masalah tersebut, kemudian produsen melakukan pendekatan secara tradisional yaitu berdasarkan keahlian dan penilaian manusia.Tetapi pendekatan ini menimbulkan masalah penundaan pekerjaan, limbah dan kesalahan jaminan bahwa keputusan yang diambil untuk menentukan tekanan baja adalah optimal.Sehingga digunakan ANN untuk menentukan kualitas semen dariparameter *input* yang diberikan, kemudian BPSO mengoptimalkan nilai kualitas atap semen yang dipilih sehingga menghasilkan *output*yang optimal. Dari hasil pengujian yang dilakukan, didapatkan peningkatan hasil produksi dari 60% menjadi 97% [NAV-11].

Berdasarkan penguraian diatas, makapada penelitian inidilakukan proses perancangan dan implementasi metode ANN dan BPSO. Metode ini digunakan dalam menentukan dosis yang tepat pada macam-macampupuk. ANN digunakan agar mengetahuipengaruh pada tanaman yang diberikan oleh kombinasi dosis pupuk. Kemudian BPSO mengoptimalkan dosispupuk, sesuai dengan pengaruh yang ingin dimaksimalkan oleh petani. Diharapkan sistem ini, dapat membantu setiap petani dalam menentukan pemberian dosis dengan kombinasi yang optimal dan aman bagi lingkungan juga manusia.

1.2 Rumusan Masalah

Berdasarkan latar belakang masalah yang telah dijelaskan, maka rumusan masalah sebagai berikut :

1. Bagaimana mengimplementasikan metode ANN dengan BPSO untuk optimasi dosis pupuk pada tanaman palawija?
2. Bagaimana tingkat akurasi dari implmentasi metode ANN dengan BPSO untuk optimasi dosis pupuk pada tanaman palawija?

1.3 Batasan Masalah

Batasan masalah yang dijadikan sebagai pedoman dalam pelaksanaan penelitian ini adalah sebagai berikut :

1. Jenis tanaman palawijayang digunakan sebagaimedia tanam adalah jagung pada lahan kering.
2. Jenis pupuk untuk optimasi dosis pada sistem ini merupakan kombinasi pupuk UREA, SP_{36}, KCL dan biochar.
3. Pengaruh pada tanaman yang diamati adalah bobot kering tanaman, bobot 1000 butir, panjang tongkol, diameter tongkol dan hasil produksi.

1.4 Tujuan

Tujuan yang ingin dicapai dari pembuatan sistem ini adalah sebagai berikut :

1. Melakukan implementasi metode ANN dengan BPSO dalam pemberian pupuk optimal sesuai kebutuhan tanaman palawija.
2. Menghitung tingkat akurasi dari implementasi metode ANN dengan BPSO untuk optimasi penggunaan pupuk sesuai kebutuhan tanaman palawija.

1.5 Manfaat

Manfaat yang ingin dirasakan dari pembuatan sistem ini adalah sebagai berikut:

1. Membantu petani dalam menentukan dosis pupuk sesuai dengan hasil yang diharapkan.

2. Menjadi acuan dalam memberikan dosis pada pupuk yang baik bagi lingkungan.
3. Menjadi perbandingan terhadap pemberian berbagai jenis pupuk dengan pengaruh pada tanaman yang dihasilkan.
4. Meningkatkan daya saing dalam menghasilkan pupuk dengan kualitas yang baik untuk produksi pertanianagar maksimal.

1.6 Sistematika Penulisan

Sistematika penulisan penelitian ditunjukkan untuk memberikan gambaran dan uraian dari penyusun tugas akhir secara garis besaryang meliputi beberapa babsebagai berikut :

BAB I : Pendahuluan

Menguraikan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan penelitian,manfaat penelitian dan sistematika pembahasan.

BAB II : Tinjauan Pustaka

Menguraikan tentang kajian pustaka,dasar teori dan referensi yang mendasari integrasi metode ANN dengan BPSO untuk optimasi penggunaan pupuk pada tanaman palawija.

BAB III : Metode Penelitian dan Perancangan

Menguraikan tentang metode dan langkah kerja yang dilakukan dalam penulisan tugas akhir yang terdiri dari studi literatur, metode pengambilan data, analisis kebutuhan, perancangan sistem, implementasi pengujian dan analisis serta pengambilan kesimpulan.

BAB IV : Implementasi

Membahas implementasi dari sistem optimasi dosis pupuk pada tanaman palawija yang sesuai dengan perancangan sistem yang telah dibuat.

BAB V : Pengujian dan Analisis

Memuat hasil pengujian dan analisis terhadap sistem optimasi penggunaan dosis pupuk pada tanaman palawija yang telah direalisasikan.

BAB VI : Penutup

Memuat kesimpulan yang diperoleh dari pembuatan dan pengujian perangkat lunakyang dikembangkan dalam tugas akhir ini serta saran-saran untuk pengembangan lebih lanjut.

BAB II
TINJAUAN PUSTAKA DAN DASAR TEORI

Pada bab ini akan dijelaskan mengenai tinjauan pustaka dan dasar teori yang mendukung penelitian. Diantaranya, teori mengenai tanaman palawija, tanaman jagung, lahan kering, pemupukan, jenis pupuk yang diberikan, dosis pemberian pupuk,jaringan syaraf tiruan, *backpropagation*, algoritma *bidirectional particle swarm optimization* dan kajian pustaka.

2.1 Kajian Pustaka

Kajian Pustaka yang dijelaskan pada bab ini yang pertama adalah, kasus penelitian kombinasi pupuk dan pestisida untuk pertanian rendah karbon, menggunakan *fuzzy mathematics*[WAN-12].Penelitian ini bertujuan untuk membandingkan hasil konsep dan teori dari penelitian sebelumnya dengan teknologi komputasi menggunakan *fuzzy*.Penelitian dilakukan dengan 3 kondisi kombinasi dosis pupuk dan pestisida untuk menemukan mana yang optimal. Metode pada komputasi ini diawali mengevaluasi kumpulanfaktordan tingkatnya.Tingkatnya dibagi kedalam 4 tingkat pada dua efek yaitu negatif dan positif.Kemudian kumpulan faktor dibentuk dalam matriks dengan nilai *fuzzy* sesuai dengan keanggotaan kelas. Tentukan matriks bobot dari setiap faktor hingga didapatkan indeksnya. Nilai indeks akan dinormalisasi untuk dicari kembali bobotnya. Bobot ini akan dievaluasi efek positif dan negatif sesuai dengan tingkatnya atau *grade.* Dalam penelitian ini diketahui, bahwa algoritma komputasi yang dihailkan lebih baik dibandingkan dengan konsep dan teori sebelumnya yang sudah sering digunakan petani.

Selanjutnya adalah, algoritma *Particle Swarm Optimization* (PSO) merupakan algoritma yang banyak diterapkan dalam penelitian untuk sistem pengambilan keputusan dan optimasi.PSO merupakan salah satu cabang ilmu dari perkembangan algoritma evolusi. Pada penelitian yang berjudul "Analisis dan Penerapan Algoritma *Particle Swarm Optimization* (PSO) pada Optimasi Penjadwalan Sumber Daya Proyek", PSO digunakan untuk menemukan rencana jaringan, yang parameternya adalah waktu mulai (*start time*). Penggunaan sumber

daya mencapai keadaan yang seimbang, ketika nilai fluktuasi kecil. Untuk itu, posisi partikel sebagai waktu mulai dari setiap aktivitas, dan ruang solusi adalah kumpulan waktu mulai dari semua aktivitas. Implementasi PSO dengan inisialisasi populasi dari M partikel secara random dan kemudian mencari posisi terbaik (solusi atau optimum) dengan memperbaharui (*update*) generasi sampai mencapai posisi yang relatif atau mencapai batas iterasi maksimum (T). Di setiap iterasi atau generasi, *local best* dan *global best* diputuskan melalui evaluasi nilai fluktuasi pada populasi partikel. Sehingga didapatkan PSO mampu menghasilkan nilai fluktuasi yang kecil [RIA-09].

Kemudian kajian pustaka ketiga adalah, Integrasi ANN dan BPSO yang merupakan kombinasi algoritma baru dalam penyelesaian masalah optimasi. Penelitian yang telah dilakukan Navalertporn (2011) metode ini digunakan dalam menentukan nilai karakteristik kualitas genteng semen. BPSO membangkitkan dua *child* dengan sitem pencarian GBEST yang berbeda untuk $child_1$dan $child_2$. Sama dengan PSO, hanya BPSO menggunakan nilai minimalisasi dalam optimasi.Sehingga dapat diketahui kualitas karakteristik sebuah semen, baik, buruk atau rata-rata berdasarkan nilai yang dihasilkan dari optimasi BPSO [NAV-11].

Pada penelitian ini, menerapkan integrasi ANN dan BPSO sebagai solusi untuk pemberian dosis pupuk pada tanaman palawija. Berdasarkan tinjauan pustaka yang telah diuraikan, implementasi algoritma ANN akan menentukan jenis pupuk yang akan digunakan berdasarkan dengan dosis yang diberikan, ANN akan memberikan *output*dari pemberian kombinasi dosis tersebut.*Output*dari pemberian dosis yang dianggap kurang maksimal akan dioptimasi pada BPSO sehingga algoritma ini memberikan kombinasi dosis sesuai dengan*output*yang diharapkan. Perbandingan objek danmetode penelitian yang akan dilakukan dengan penelitian sebelumnya ditunjukan pada tabel berikut :

Tabel 2.1 Perbandingan Obyek dan Metode

Pembanding	Objek	Metode
Sebelum	Karakteristik kualitas dan parameter proses keramik	Algoritma ANN dan BPSO
Usulan	Jenis dan kombinasi dosis pupuk	Algoritma ANN dan BPSO

Pada Tabel 2.2 Perbandingan *Input, Output* dan Proses dari integrasi algoritma ANN dan BPSO pada penelitian yang dilakukan dengan penelitian sebelumnya ditunjukan sebagai berikut:

Tabel 2.2 Perbandingan *Input*, Proses dan *Output*

No	Judul	Objek dan Proses *input*	*Output* Hasil Penelitian
1	*A Methodological Approach to Assess the Combined reduction of Chemical Pesticides and Chemical Fertilizerx for Low-Carbon Agriculture*	Tiga kondisis *treatment* kombinasi dosis : 1. Tidak diberikan pupuk boron. 2. Diberi 12% pupuk Granular Boron. 3. Diberikan 25% Foliar Boron. Proses*input* : 1. Pembentukan faktor evaluasi kelas. 2. Pembentukan faktor untuk penilaian matriks. 3. Menentukan berat setiap faktor, membuat matriks bobot. 4. Melakukan normalisasi. 5. Menghitung hasil evaluasi. 6. Menentukan nilai indeks evaluasi. 7. Evaluasi faktor set dan evaluasi kelas. 8. Menentukan fungsi keanggotaan dan penilaian. 9. Hitung hasil faktor dan evaluasi.	Y1=max(yi)=y2 =0.3676, Y2=max(yi)=y2 =0.3795, Y3=max(yi)=y2 =0.3769, Dari hasil penelitian menggunakan *fuzzy* didapatkan treatment 2 lebih baik .
2	Analisis dan Penerapan Algoritma *Particle Swarm Optimization* (PSO) pada Optimasi Penjadwalan Sumber Daya Proyek	Objek : Nama Aktivitas, Durasi, *Predecessor*, dan Jumlah sumber daya proyek. Parameter PSO (jumlah partikel, jumlah iterasi, c_1, c_2 dan *inertia weight*) Proses*input* : 1. Inisialisasi posisi dan kecepatan 2. Posisi = kumpulan *start time* kecepatan = nilai perubahan jarak dari aktivitas. 3. Evaluasi nilai fluktuasi. 4. Membandingkan dan meng*update* nilai *LocalBest* dan *GlobalBest.* 5. Nilai fluktuasi < localbest(P_{id}) $P_{id} = F_i$ Nilai fluktuasi < globalbest (P_{gd}) $P_{gd}=f_i$ 6. *Update* posisi dan kecepatan.	*Start time* yang menghasilkan nilai fluktuasi yang optimal.
3	*Optimization of Tile Manufacturing Process Using Particle Swarm Optimization*	*Input* untuk proses ANN : - *Surface Quality* (SQ), *Edge Sharpness* (ES), *Weight* (W), *Average Thicknes* (AT) dan *Breaking Strength* (BS) - *Moisture of mortar, Weight of mortar, press time, pressure, Wet mix, Dry mix.* *Input* untuk proses BPSO : W, C_1 dan C_2, Posisi X, Jumlah populasi dan Jumlah iterasi	Nilai GBEST yang optimal untuk proses parameter dan MSE.

		Proses ANN : 1. Menentukan Data *training* dan data *testing* jumlah iterasi dan*hiden node, learning rate momentum* 2. Lakukan proses pelatihan yaitu *feedforward, backpropagation* dan perbaikan bobot. 3. Diperoleh bobot dengan iterasi dan nilai MSE yang mendekati target. 4. Lakukan proses pengujian hanya sampai tahap *feedforward*. 5. Menghasilkann nilai karakteristik kualitas Proses BPSO : 1. Inisialisasi Parameter 2. Membangkitkan populasi partikel dan GBEST 3. Menghitung *gbest* dan *pbest* untuk *child*1 4. Menghitung *gbest* dan *pbest* untuk *child*2 5. *Update*kecepatan, posisi*child*1 dan *child*2 6. *Updategbest* 7. *Update* populasi partikel	
4	**Integrasi Metode *Artificial Neural Network* (ANN) dengan *Bidirectional Particle Swarm Optimization* (BPSO) untuk Optimasi Dosis Pupuk pada Tanaman Palawija**	*Input* untuk proses ANN : Learning rate, momentum, MSE, jumlah iterasi *Input* untuk proses BPSO : w, C_1 dan C_2, Posisi X, Jumlah populasi dan Jumlah iterasi Proses ANN : 1. Menentukan Data *training* dan data *testing* jumlah iterasi dan *hiden node, learning rate,momentum* 2. Lakukan proses pelatihan yaitu *feedforward, backpropagation* dan perbaikan bobot. 3. Diperoleh bobot dengan iterasi dan nilai MSE yang mendekati target. 4. Lakukan proses pengujian hanya sampai tahap *feedforward*. 5. Menghasilkan *output*pengaruh pada tanaman palawija Proses BPSO : 1. Inisialisasi Parameter, dan penagruh pada tanaman yang akan dioptimasi 2. Membangkitkan populasi partikel dan GBEST 3. Menghitung *gbest* dan *pbest* untuk *child*1 4. Menghitung *gbest* dan *pbest* untuk *child*2 5. *Update* kecepatan, posisi *child*1 dan *child*2 6. *Updategbest* 7. *Update* populasi partikel	Nilai dosis yang optimal untuk setiap jenis pupuk dan MSE.

Sumber :[WAN-12] [RIA-09] [NAV-11]

2.2 Tanaman Palawija

Palawija adalah semua jenis tanaman semusim yang ditanam pada lahan kering. Menurut Kamus Besar Bahasa Indonesia (KBBI), palawija adalah tanaman yang ditanaman di sawah atau ladang selain padi. Biasanya palawija ditanam di sawah kering setelah petani menanam padinya. Tanaman palawija harus diambil hasilnya sebelum musim tanam padi berikutnya. Palawija secara harfiah dapat diartikan sebagai tanaman kedua. Maksud daritanaman kedua yaitu palawija merupakan tanaman-tanaman hasil pertanian yangkedua setelah tanaman pokok kita yaitu padi. Dalam pengertian sekarang, palawija berarti semua tanaman pertanian semusimyang ditanam pada lahan kering. Yang termasuk tanaman palawija yaitu kacangtanah, jagung, ketela pohon, kedelai, dan umbi jalar. Dapat dikatakan bahwatanaman palawija ini merupakan hasil produksi sekunder dari petani yang manahasil produksi primer mereka yaitu padi. Tanaman palawija ini juga bisa digunakanuntuk menggantikan padi sebagai makanan pokok. Pada saat ini para petani biasanya memanfaatkan lahan pertanian mereka untuk menanam tanaman palawijauntuk mendapatkan hasil tambahan. Sehingga kini banyak kita jumpai ladang yang ditanami padi sekaligus juga ditanmi jagung danketela pohon [KOU-08].

Dalam sistem yang menekankan pertanian berkelanjutan, palawija merupakan salah satu komponen untuk melakukan rotasi tanaman. Palawija mampu menghemat air di musim kering sehingga tidak memberikan beban bagi irigasi, terutama ketika irigasi tidak mampu memberikan cukup air bagi sawah. Palawija juga mampu menjadi sumber penghidupan di dataran tinggi di mana padi tidak dapat tumbuh [KOU-08]. Gambar 2.1 Hasil Palawija yang secara umum, dikembangkan di pertanian :

Gambar 2. 1 Hasil Palawija
Sumber: Ketahanan Pangan Nasional

2.3 Jagung

Jagung (Zea mays L.) sampai saat ini masih merupakan komoditi strategis karena di beberapa daerah jagung masih merupakan bahan makanan pokok kedua setelah beras. Jagung juga mempunyai arti penting dalam pengembangan industri di Indonesia karena merupakan bahan baku untuk industri pangan maupun industri pakan ternak. Dengan semakin berkembangnya industri pengolahan pangan dan pakan ternak di Indonesia maka kebutuhan akan jagung terusmeningkat, tetapi jika tidak diimbangi dengan peningkatan produksi yangmemadai akan menyebabkan Indonesia harus mengimpor jagung dalam jumlah besar. Peningkatan produktivitas tanaman jagung dapat dilakukanmelalui kombinasi penerapan teknologi, khususnya penggunaan varietas unggul dan pemupukan dosis yang tinggi agar dapat menghasilkan panen yang maksimal [MOE-13].

Jagung merupakan tanaman daerah panas.Dapat tumbuh di dataran rendah maupun dataran tinggi, yaitu sekitar 1-2000 dari permukaan laut. Jagung dapat ditanam pada tanah ringan maupun tanah berat, yang penting tanah tersebut harus gembur dan banyak mengandung humus. Curah hujan ideal sekitar 85-200 mm/bulan dan harus merata.Sebaiknya ditanam awal musim hujan atau menjelang musim kemarau dan membutuhkan sinar matahari.Suhu optimum antara 23^{o} C - 30^{o} C. Varietas unggul dari jagung adalah jagung hibrida. Cirinya antara lain tanggap terhadap pemupukan, umur pendek, berdaya hasil tinggi, tahan terhadap penyakit dan hama, beradaptasi baik pada berbagai lingkungan serta tegap dan tahan terhadap rebah [KUR-14]. Gambar 2.2 Lahan Pertanian Jagung Hibrida yang menjadi salat satu aspek palawija yang sangat berpengaruh :

Gambar 2. 2 Lahan Pertanian Jagung Hibrida
Sumber: Dinas Pertanian Provinsi Jawa Barat, 2012

2.3.1 Ciri Morfologis Tanaman Jagung

Pemahaman morfologi atau bentuk luar tanaman dan fase pertumbuhan jagung sangat membantu dalam mengidentifikasi pertumbuhan tanaman. Satu siklus hidupnya diselesaikan dalam 80-150 hari. Tinggi tanaman jagung sangat bervariasi, meskipun tanaman jagung rata-rata ketinggiannya antara 1 m – 3 m, ada varietas yang dapat mencapai tinggi 6 m [PUR-08].

2.3.1.1 Akar

Akar jagung tergolong akar serabut yang dapat mencapai kedalaman 8 m, meskipun sebagian besar berada pada kisaran 2 m. Tanaman jagung mempunyai akar serabut. Tanaman jagung mempunyai akar serabut dengan tiga jenis akar, yaitu akar seminal, akar adventif, dan akar kait atau penyangga. Fungsi dari akar penyangga adalah menjaga tanaman agar tetap tegak dan mengatasi rebah batang. Akar ini juga membantu penyerapan hara dan air. Perkembangan akar jagung (kedalam dan penyebarannya) bergantung pada varietas, pengolahan tanah, fisik dan kimia tanah, keadaan air tanah, dan pemupukan. Pemupukan nitrogen dengan takaran berbeda menyebabkan perbedaan perkembangan (*plasticity*) sistem perakaran dan jagung.

2.3.1.2 Batang

Batang jagung tegak dan mudah terlihat. Batang jagung berwarna hijau sampai keunguan, berbentuk bulat dengan penampang melintang selebar 125 – 250 cm. Batang berbuku – buku yang dibatasi oleh ruas – ruas. Daun terdiri atas pelepah dan helaian daun. Helaian daun memanjang dengan ujung daun meruncing. Antara pelepah daun dan helaian daun dibatasi oleh spikulasi yang berguna untuk menghalangi masuknya air hujan atau embun ke dalam pelepah daun. Jumlah daun berkisar 10 – 20 helai pertanaman. Daun berada pada setiap ruas batang dengan kedudukan yang saling berlawanan.

2.3.1.3 Daun

Daun jagung adalah daun sempurna. Bentuknya memanjang antara pelepah dan helai daun terdapat ligula. Ligula ini berbulu dan berlemak, fungsi ligula

adalah mencegah air masuk kedalam kelopak daun dan batang, tulang daun sejajar dengan ibu tulang daun. Permukaan daun ada yang licin dan ada yang berambut.

2.3.1.4 Bunga

Bunga betina jagung berupa tongkol yang terbungkus oleh semacam pelepah dengan rambut. Rambut jagung sebenarnya adalah tangkai putik. Tanaman jagung memiliki bunga jantan dan bunga betina yang terpisah dalam satu tanaman. Bunga betina berwarna putih panjang dan biasa disebut rambut jagung. Bunga betina dapat menerima tepung sari disepanjang rambutnya. Bunga jantan tumbuh dibagian pucuk tanaman berupa karangan bunga (*inflorescence*), serbuk sari berwarna kuning dan beraroma khas. Bunga betina tersusun dalam tongkol. Tongkol tumbuh dari buku, diantara batang dan 8 pelepah daun (ketiak daun). Bunga jantan cenderung siap untuk penyerbukan 2 – 5 hari lebih cepat dari bunga betinanya (*protandri*). Penyerbukan pada jagung terjadi bila serbuk sari dari bunga jantan jatuh dan menempel pada rambut tongkol (bunga betina). Pada jagung umumnya terjadi penyerbukan silang (*cross pollinated crop*). Penyerbukan terjadi dari serbuk sari tanaman lain. Sangat jarang penyerbukan yang serbuk sarinya dari tanaman sendiri.

2.3.1.5 Biji

Panen jagung mulai dapat dilakukan jika biji sudah masak secara fisiologi yaitu pada waktu kandungan bahan kimia dalam biji telah mencapai jumlah optimal. Kadar air biji merupakan kriteria untuk saat panen yang tepat dimana biji jagung yang telah masak secara fisiologis jika kandungan air dalam biji sekitar 25-30 %. Selain dari kadar air juga dapat dilihat dari tanda-tanda luar tanaman yaitu menguningnya daun dan kelobot, biji berwarna kuning emas, mengkilat dan keras (untuk jagung kuning). Umur tanaman kurang baik digunakan sebagai pedoman untuk menentukan umur panen, karena dipengaruhi oleh banyak faktor, diantaranya adalah curah hujan, suhu udara dan kesuburan tanah. Sekalipun demikian, umumnya saat panen dicapai pada usia 7-8 minggu setelah tanaman jagung berbunga. Tanaman jagung mempunyai satu atau dua tongkol, tergantung varietas. Varietas Hibrida rata – rata memiliki 2 tongkol. Tongkol jagung

diselimuti oleh daun kelebot. Tongkol jagung memiliki rata-rata diameter 3cm – 5cm dengan panjang tongkol 13cm – 20cm. Setiap tongkol terdiri atas 10 – 16 baris biji jagung yang terdiri dari 200 – 400 butir biji jagung.

2.3.2 Deskripsi Jagung Hibrida

Jagung Hibrida merupakan tanaman semusim (*annual*). Satu siklus hidupnya diselesaikan dengan 120 hari setelah tanam (HST). Deskripsi Jagung Hibrida ditunjukkan pada Tabel 2.3 Deskripsi Jagung Hibrida sebagai berikut:

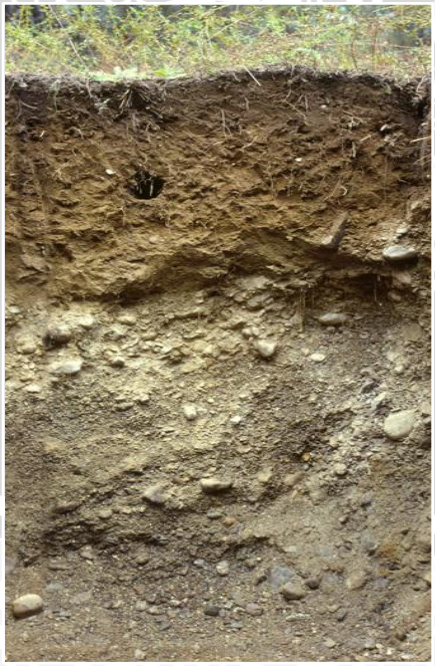
Tabel 2.3 Deskripsi Jagung Hibrida

Tanggal dilepas	995
Asal	F1 dari silang tunggal antara FS 4 dengan FS 9. FS 4 dan FS 9 merupakan tropical inbred yang dikembangkan oleh Charoen Seed Co., Ltd. Thailand dan Dekalb Plant Genetic, USA
Umur	50% keluar rambut: ± 56 hari
Panen	± 103 hari
Batang	Tinggi dan tegap
Warna batang	Hijau
Tinggi tanaman	± 232 cm
Daun	Panjang, lebar dan terkulai
Warna daun	Hijau cerah
Keragaman tanaman	Seragam
Perakaran	Baik
Kerebahan	Tahan
Tongkol	Sedang, silindris dan seragam
Kedudukan tongkol	Ditengah-tengah batang
Kelobot	Menutup tongkol dengan baik
Tipe biji	Setengah mutiara (semi flint)
Warna biji	Kuning oranye
Jumlah baris/ tongkol	12 – 14 baris
Bobot 1000 biji	± 265 gram
Rata-rata hasil	8.9 ton/ ha pipilan kering
Potensi hasil	13 ton/ha pipilan kering
Ketahanan	Toleran terhadap penyakit bulai dan karat daun
Keterangan	Baik ditanam didataran rendah sampai ketinggian 1000 m dpl.

Pada Tabel 2.3 Deskripsi Jagung Hibrida untuk tinggi tanaman bisa diukur dari permukaan tanah hingga teratas sebelum bunga jantan. Pada umumnya tanaman Jagung Hibrida tidak memiliki kemampuan untuk membentuk anakan. Batang beruas-ruas, ruas terbungkus pelepah daun dan batang cukup kokoh.

2.4 Lahan Kering

Lahan kering adalah lahan yang secara fisik tidak diari atau tidak mendapatkan irigasi, hanya mengandalkan curah hujan. Indonesia mempunyai lahan kering dengan luas sekitar 49 juta Ha. Lahan kering sangat berpotensial untuk dikembangkan tetapi produktivitasnya cepat menurun jika tidak diusahakan dengan baik. Selain itu, hambatan lainnya adalah tingkat kesuburan kimianya relative rendah, kandungan bahan organik rendah, hara nitrogen, fosfor dan kalium relatif rendah dan sifat tanah kurang baik. Sebagian besar tanah dari lahan kering tersebut sebelumnya berupa hutan, padang alang- alang dansemak belukar [NUS-07]. Oleh karena itu dalam pengolahannya sangat penting memperhatikan kandungan hara yang akan diberikan kedalam lahankering.Berikut adalah Gambar 2.3 Tanah Lahan Kering yang banyak di jumpai di lahan pertanian :



Gambar 2.3 Tanah Lahan Kering
Sumber: Bahan Kajian Dasar Ilmu Tanah Universitas Brawijaya, 2013

2.5 Pupuk dan Jenisnya

Tanah sangat penting artinya bagi usaha pertanian karena kehidupan dan perkembangan tumbuh-tumbuhan dan segala mahluk hidup di dunia sangat memerlukan tanah. Tetapi arti penting ini terkadang diabaikan manusia, sehingga tanah tidak lagi berfungsi sebagai mana mestinya. Dalam usaha pertanian, para petani harus sadar bahwa melakukan pertanaman secara terus menerus tanpa memperhatikan pemeliharaan atas tanahnya agar seimbang tentunya akan menimbulkan resiko. Dengan demikian akan mengakibatkan merosotnya hasil dan bahkan pada akhirnya tanah tidak mampu lagi menunjukan produktivitasnya [MUL-08].

Pemberian atau penambahan zat-zat didalam tanah melalui pupuk tidak semudah yang diperkirakan oleh kebanyakan orang.Pupuk adalah material yang ditambahkan pada media tanam atau tanaman untuk mencukupi kebutuhan hara yang diperlukan tanaman sehingga mampu berproduksi dengan baik.Material pupuk dapat berupa bahan organik ataupun non-organik (mineral).Pemupukan bertujuan menyelidiki tentang zat-zat apakah yang perlu diberikan kepada tanah sehubung dengan kekurangan zat-zat yang diperlukan untuk pertumbuhan, perkembangan dan hasil produksi yang tinggi.Sebelumnya harus ada perlakuan yang diteliti terlebih dahulu sebelum zat- zat itu diberikan kedalam tanah, seperti pengaruh yang langsung dan tidak langsung pada sifat tanah serta tanaman yang dibudidayakan. Secara umum Pupuk dapat dibedakan berdasarkan bahan asal, senyawa, jumlah dan macam hara yang dikandungnya.

Berdasarkan asalnya dibedakan:

1. Pupuk alam ialah pupuk yang terdapat di alam atau dibuat dengan bahan alam tanpa proses yang berarti. Misalnya: pupuk kompos, pupuk kandang, guano, pupuk hijau dan pupuk batuan P.
2. Pupuk buatan ialah pupuk yang dibuat oleh pabrik. Misalnya: TSP, urea, rustika dan nitrophoska. Pupuk ini dibuat oleh pabrik dengan mengubah sumber daya alam melalui proses fisika dan/atau kimia.

Berdasarkan senyawanya dibedakan:

1. Pupuk organik ialah pupuk yang berupa senyawa organik. Kebanyakan pupuk alam tergolong pupuk organik: pupuk kandang, kompos, guano. Pupuk alam yang tidak termasuk pupuk organik misalnya *rock phosphat*, umumnya berasal dari batuan sejenis apatit [$Ca_3(PO_4)_2$].
2. Pupuk anorganik atau mineral merupakan pupuk dari senyawa anorganik. Hampir semua pupuk buatan tergolong pupuk anorganik.

Berdasarkan jumlah hara yang dikandungnya dibedakan:

1. Pupuk yang hanya mengandung satu hara tanaman saja. Misalnya: urea hanya mengandung hara N, TSP hanya dipentingkan P saja (sebetulnya juga mengandung Ca).
2. Pupuk majemuk ialah pupuk yang mengandung dua atau lebih dua hara tanaman. Contoh: NPK, amophoska, nitrophoska dan rustika.

Berdasarkan macam hara tanaman dibedakan:

1. Pupuk makro ialah pupuk yang mengandung hanya hara makro saja: NPK, nitrophoska, gandasil.
2. Pupuk mikro ialah pupuk yang hanya mengandung hara mikro saja misalnya: mikrovet, mikroplek, metalik.
3. Campuran makro dan mikro misalnya pupuk gandasil, bayfolan, rustika. Sering juga ke dalam pupuk campur makro dan mikro ditambahkan juga zat pengatur tumbuh (hormon tumbuh).

Pada penelitian ini dijelaskan kombinasi jenis pupuk yang akan digunakan [SOE-13]:

2.5.1 Urea $CO(NH_2)_2$

Pupuk urea adalah pupuk buatan senyawa kimia organic dari $CO(NH_2)_2$, pupuk padat berbentuk butiran bulat kecil . Urea larut sempurna di dalam air. Pupuk ini mempunyai kadar N 45% tidak mengasamkan tanah. Nitrogen berperan dalam proses pertumbuhan tanaman dan meningkatkan kandungan klorofil tanaman [SYE-12]. Rekomendasi pemupukan N pada tanaman jagung dengan

takaran pupuk N berdasarkan target hasildan kenaikan hasil jika diberi N dibanding tanpa pemberian N dapat dilihat pada Tabel 2.3

Tabel 2.4 Rekomendasi Pemupukan N

Kenaikan Hasil dibandingkan tanpa N (t/ha)	Target Hasil		
	Rendah	Sedang	Tinggi
	…… Takaran N (kg/ha)……		
1			
2	80		
3	120	105	
4	160	140	120
5		175	150
6			180
7			210
8			240

Sumber : [GOZ-11]

Pada tabel 2.3 rekomendasi pemupukan N, berdasarkan target hasil dibagi atas tiga yaitu rendah sedang dan tinggi. Takaran pemberian rendah diberi rentang bawah yaitu 80 N kg/ha, batas tengah 120 N kg/ha dan batas atas 160 N kg/ha. Sama halnya dengan target hasil sedang dan tinggi diberi rentang pemberian pupuk N, sesuai dengan kenaikan hasil tanaman jagung. Perhitungan untuk pupuk Urea yang dibutuhkan jika takaran minimal N (kg/ha) adalah 80 kg/ha dan takaran maksimal 240 kg/ha adalah :

Pupuk Urea yang dibutuhkan $= 80 \text{ kg/ha} * \frac{100}{45} = 177{,}78 \text{ kg/ha}$

Pupuk Urea yang dibutuhkan $= 240 \text{kg/ha} * \frac{100}{45} = 533{,}33 \text{ kg/ha}$

Sehingga didapatkan rata-rata (*range*) dosis untuk pemberian pupuk Urea pada tanaman jagung hibrida adalah 177.78 kg/ha-533,33 kg/ha.

2.5.2 SP_{36}

Mengandung 36% fosfor dalam bentuk P2O5. Pupuk ini terbuat dari fosfat alam dan sulfat. Berbentuk butiranan dan berwarna au-abu. Sifatnya agak sulit larut dalam air dan bereaksi lambat sehingga selalu digunakan sebagai pupuk dasar. Reaksi kimianya tergolong netral, tidak higroskopis, dan tidak bersifat membakar. Fosfor berperan dalam meningkatkan kualitas biji/buah dan meningkatkan bobot biji. Rekomendasi pemupukan P pada Tanaman jagung

dengan takaran pupuk P berdasarkan target hasil dan kenaikan hasil jika diberi P dibanding tanpa pemberian P dapat dilihat pada Tabel 2.4

Tabel 2.5 Rekomendasi Pemupukan P

Kenaikan Hasil dibanding tanpa P (ton/ha)	Target Hasil	
	5 – 8 ton/ha	9 – 12 ton/ha
	..... takaran P_2O_5 (kg/ha).....	
0	5 -10	10 -15
0,5	25 -30	30 -35
1,0	45 – 50	50 -55
1,5	65 – 70	70 -75
2,0	85 – 90	90- 95
2,5	105 – 110	110 – 115

Sumber : [GOZ-11]

Pada tabel 2.4 rekomendasi pemupukan P untuk mendapatkan target hasil sekitar 5-8 ton/ha dengan tidak ada kenaikan hasil dapat diberikan pupuk P dengan dosis 5-10 P_2O_5 kg/ha. Perhitungan untuk pupuk SP_{36} yang dibutuhkan jika takaran minimal P (kg/ha) adalah 5 kg/ha dan takaran maksimal 115 kg/ha adalah :

Pupuk SP_{36}yang dibutuhkan = 5 kg/ha*$\frac{100}{36}$ = 13.8 kg/ha

Pupuk SP_{36} yang dibutuhkan = 115 kg/ha*$\frac{100}{36}$ = 319,4 kg/ha

Sehingga didapatkan rata-rata (*range*) dosis untuk pemberian pupuk SP_{36} pada tanaman jagung hibrida adalah 13.8 kg/ha- 319,4kg/ha

2.5.3 KCL

Pupuk kalium klorida atau potassium klorida (KCL). Ada 2 macam pupuk KCl yang beredar di pasaran, yaitu KCL 80 (mengandung 50% K_2O) dan KCL 90 (mengandung 53% K_2O). Fungsi kalium bagi tanaman adalah mempengaruhi susunan dan mengedarkan karbohidrat di dalam tanaman, mempercepat metabolisme unsur nitrogen dan mencegah bunga dan buah agar tidak mudah gugur. Kalium juga berperan dalam memperkokoh batang, akar dan daun sehingga tidak mudah roboh atau terserang penyakit. Rekomendasi pemupukan K pada tanaman jagung dengan takaran pupuk K berdasarkan target hasil dan kenaikan hasil jika diberi K dibanding tanpa pemberian K dapat dilihat pada Tabel 2.5

Tabel 2.6 Rekomendasi Pemupukan K

Kenaikan Hasil dibandingkan tanpa K (t/ha)	Target Hasil		
	4 -7 ton/ha	7 – 10 ton/ha	10 -12 ton/ha
	…… Takaran N (kg/ha)……		
0	20-30	30 -40	40 -50
0,5	40 – 50	50 -60	60 -70
1,0	60 -70	70 -80	80 -90
1,5	80 -90	90 -100	100 -110
2,0	100 -110	110 -120	120 -130
2,5	120 – 130	130 -140	140 -150

Sumber : [GOZ-11]

Perhitungan untuk pupuk KCL(mengandung 50% K_2O) yang dibutuhkan jika takaran minimal K (kg/ha) adalah 20 kg/ha dan takaran maksimal 150 kg/ha adalah :

Pupuk KCL yang dibutuhkan = 20kg/ha*$\frac{100}{50}$ = 40 kg/ha

Pupuk KCL yang dibutuhkan = 150kg/ha*$\frac{100}{50}$ = 300 kg/ha

Sehingga didapatkan rata-rata (*range*) dosis untuk pemberian pupuk KCL pada tanaman jagung hibrida adalah 40 kg/ha - 300 kg/ha. Berikut adalah contoh gambar dari pupuk yang digunakan pada penelitian ini pada Gambar 2.4 Jenis-jenis Pupuk Anorganik :

Gambar 2.4 Jenis-jenis Pupuk Anorganik
Sumber : Petrokimia Gresik

2.5.4 Biochar

Biochar atau yang lebih dikenal sebagai arang, merupakan materi padat yang terbentuk dari karbonisasi biomassa (berat basah). Biochar merupakan bentuk lain dari biomassa yang telah mengalami proses pembakaran secara pirolisis, yaitu pembakaran dengan sistem kedap udara [UTO-12]. Biochar ditambahkan ke tanah dengan tujuan meningkatkan fungsi tanah dan mengurangi emisi dari biomassa yang secara alami terurai menjadi gas rumah kaca. Menurut Bambang (2012), bahan baku pembuatan biochar umunya adalah residu biomassa pertanian atau kehutanan, termasuk potong kayu, tempurung kelapa, tandan kelapa sawit, tongkol jagung, sekam padi, atau kulit buah kacang-kacangan, kuli-kulit kayu, sisa-sisa usaha perkayuan, serta bahan organik yang berasal dari sampah kertas dan kotoran hewan. Limbah tersebut mengalami pembakaran dalam keadaan oksigen yang rendah atau tanpa oksigen akan dihasilkan 3 substansi, yaitu metana dan hidrogen yang dapat dijadikan bahan bakar, bio-oil yang dapat diperbaharui dan arang hayati (biochar) yang mempunyai sifat stabil dan kaya karbon (>50%).

Sebagai bahan organik, biochar bersifat agak basa, dan mengandung beberapa unsur hara essensial, terutama K, dan P, sehingga diyakini pemanfaatan biochar untuk pertanian dapat memperbaiki kesuburan dan prosuktivitas tanah pertanian [UTO-12]. Hasil penelitian Nurida (2008) menunjukkan bahwa biochar limbah pertanian dapat meningkatkan kandungan C-organik dan kandungan unsur hara makro seperti N,P dan K. Hasil penelitian analisis biochar limbah pertanian disajikan pada Tabel 2.7 :

Tabel 2.7 Hasil Analisis Bahan Arang (Biochar) dari Limbah Pertanian

Variabel	Tempurung kelapa	Kulit buah kakao	Tempurung kelapa sawit	Sekam padi
C-organik total (%)	24.33	37.5	37.53	35.98
Asam humat (%)	0.56	0.91	2.1	0.79
Asam Sulfat (%)	0.71	3.31	2.36	1.57
Kadar Abu	2.09	13.65	10.04	27.05
Kadar N	0.2	1.91	1.09	0.73
C/N rasio	122	20	34	49
Kadar P (%)	0.02	0.4	0.09	0.14
Kadar K (%)	0.01	0.47	0.01	0.03

Sumber :[NUR-08]

Pada penelitian Widowati (2010), hasil penelitian menunjukkan biochar pupuk kandang dengan dan tanpa N,P,K berpengaruh baik terhadap ketersediaan N,P,K tanah. Penurunan bahan organik tanah dari kombinasi biochar dan pupuk organik lebih kecil dibandingkan dengan biochar kemudian dengan pupuk organik. Biochar tanpa N,P,K menghasilkan biomassa mikrobia lebih besar dibandingkan dengan pupuk organik tanpa N,P,K. Produksi biomassa tanaman tidak berbeda antara biochar dan pupuk organik dengan dan tanpa N,P,K.

Gambar 2.5 Biochar
Sumber :www.biochar-international.org, diakses pada 19 Maret 2015

Gambar 2.6 Proses Pembuatan Biochar dengan Pirolisis
Sumber :www.biochar-international.org, diakses pada 19 Maret 2015

2.6 Jaringan Syaraf Tiruan

Jaringan syaraf tiruan (JST) adalah paradigma pengolahan informasi yang terinspirasi oleh sistem syaraf secara biologis, seperti proses informasi pada otak manusia. Elemen kunci dari paradigma ini adalah struktur dari sistem pengolahan informasi yang terdiri dari sejumlah besar elemen pemrosesan yang saling berhubungan (neuron), bekerja serentak untuk menyelesaikan masalah tertentu. Cara kerja JST seperti cara kerja manusia, yaitu belajar melalui contoh [SUT-11] .

2.6.1 Kelebihan JST

JST mempunyai kemampuan yang luar biasa untuk mendapatkan informasi dari data yang rumit atau tidak tepat, mampu menyelesaikan permasalahn yang tidak terstruktur dan sulit didefinisikan, dapat belajar dari pengalaman, mampu mengakusisi pengethauan walaupun tidak ada kepastian, mampu melakukan generalisasi dan ekstraksi dari suatu pola data tertentu, dapat menciptakan suatu pola pengetahuan melalui pengaturan diri atau kemampuan belajar (*self organizing*), mampu memilih suatu *input* data kedalam kategori tertentu yang sudah ditetapkan (klasifikasi), mampu menggambarkan suatu objek secara keseluruhan walaupun hanya diberikan sebagian data dari objek tersebut (asosiasi), mempunyai kemampuan mengolah data-data *input* tanpa harus mempunyai target (*self organizing*), dan mampu menemukan jawaban terbaik sehingga mampu meminimalkan fungsi biaya (optimasi) [SUT-11].Kelebihan-kelebihan yang diberikan oleh JST antara lain :

1. Belajar Adaptive : kemampuan untuk memepelajari bagaimana melakukan pekerjaan berdasarkan data yang diberikan untuk pelatihan atau pengalaman awal
2. *Self-Organisation* : sebuah JST dapat membuat organisasis sendiri atau representasi dari informasi yang diterimanya selama waktu belajar.
3. *Real Time Operation* : perhitungan JST dapat dilakukan secara pararel sehingga perangkat keras yang dirancang dan diproduksi secara khusus dapat mengambil keuntungan dari kemampuan ini

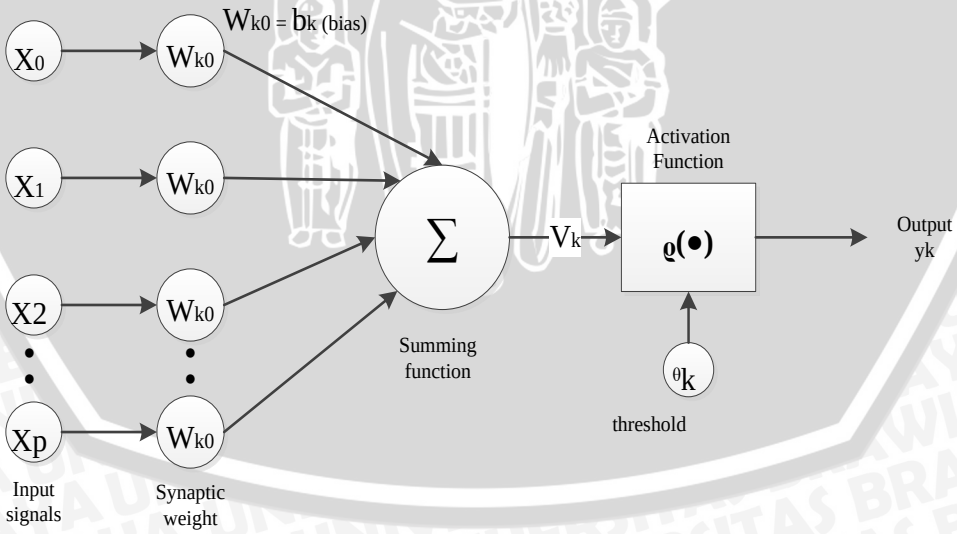
2.6.2 Model Matematika

JST merupakan generalisasi model matematis dengan beberapa asumsi berikut [SUT-11] :

1. Pemrosesan informasi terjadi pada neuron.
2. Sinyal dikirimkan di antara neuron-neuron melalui penghubung dendrit dan akson.
3. Penghubung antarelemen memiliki bobot yang akan menambah atau mengurangi sinyal.
4. Untuk menentukan *output*, setiap neuron memiliki fungsi aktivitas yang dikenakan pada jumlah semua *input*-nya. Besar *output* akan dibandingkan dengan nilai *threshold* tertentu.

Berdasarkan model matematis tersebut, baik tidaknya suatu model JST ditentukan oleh hal-hal berikut :

1. Arsitektur jaringan, yaitu sebuah arsitektur yang menentukan pola anta neuron.
2. Metode pembelajaran (*learning method*), yaitu metode yang digunakan untuk menentukan dan mengubah bobot.
3. Fungsi aktivasi.Secara matematis, proses ini dijelaskan pada gambar 2.7



Gambar 2.7 Model Matematis dari JST
Sumber : [SUT-11]

2.6.3 Fungsi Aktivasi

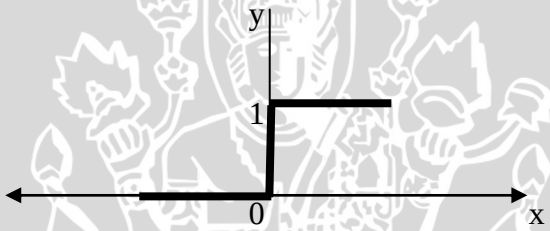
Fungsi aktivasi merupakan penentuan keluaran suatu neuron yaitu mendefinisikan nilai *output* dari sebuah neuron. Fungsi aktivasi merupakan fungsi matematis yang berfungsi untuk membatasi dan menentukan jangkauan keluaran suatu neuron. Beberapa jenis jfungsi aktivasi yang terdapat pada jaringan syaraf tiruan antara lain :

1. Fungsi *Hard Limit* (Tangga Biner)

Fungsi ini sering digunakan pada single *layer* net, *output*dari fungsi ini merupakan bilangan biner dari (0 atau 1) dengan persamaan (2-1) :

$$y = \begin{cases} 0 & \text{jika } x \leq 0 \\ 1 & \text{jika } x > 0 \end{cases} \quad \textbf{(}2\text{-}1\textbf{)}$$

Gambar dari fungsi aktivasi *hard limit* adalah sebagai berikut :



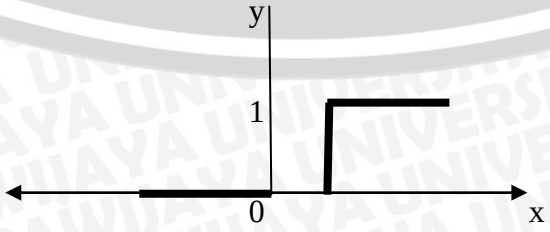
Gambar 2.8 Grafik Fungsi *Hard Limit*
Sumber : [SIA-05]

2. Fungsi *Threshold*

Fungsi ini merupakan perkembangan dari fungsi aktivasi *hard limit*, namun dengan menambahkan nilai *threshold*atau nilai ambang(θ). Rumus fungsi aktivasi ini adalah :

$$y = \begin{cases} 1 & \text{jika } x \geq \theta \\ 0 & \text{jika } x < \theta \end{cases} \quad (2\text{-}2)$$

Gambar dari fungsi aktivasi *treshold* adalah sebagai berikut :

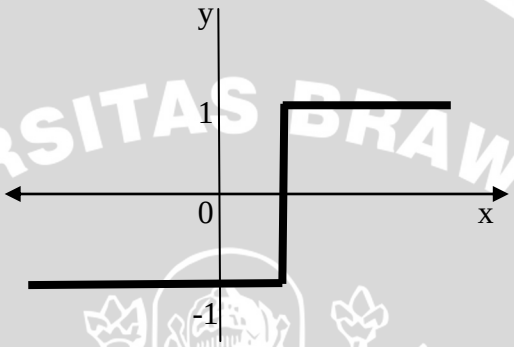


Gambar 2.9 Grafik Fungsi *Threshold*
Sumber : [SIA-05]

3. Fungsi Bipolar *Threshold*

Fungsi ini merupakan pengembangan dari fungsi aktivasi *threshold*(dengan nilai ambang θ). Fungsi *threshold* yang dibuat tidak bernilai 0 atau 1, tapi bernilai -1 atau 1.Berikut Rumus fungsi aktivasi dan gambar :

$$y = \begin{cases} 1 & \text{jika } x \geq \theta \\ -1 & \text{jika } x < \theta \end{cases} \quad (2\text{-}3)$$



Gambar 2.10 Grafik Fungsi Bipolar *Threshold*
Sumber : [SIA-05]

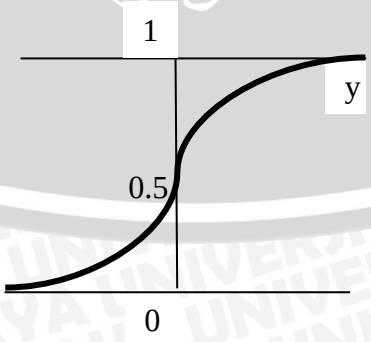
4. Fungsi Sigmoid Biner

Fungsi sigmoid sering dipakai karena memenuhi beberapa syarat yaitukontinu, terdeferensial dengan mudah dan merupakan fungsi yang tidak naik turun. Fungsi ini berada dalam range 0 dan 1. Rumus fungsi aktivasi ini adalah:

$$y = \frac{1}{1+e^{-x}} \quad (2\text{-}4)$$

$$y' = f(x)(1 - f(x)) \quad (2\text{-}5)$$

Gambar dari fungsi aktivasi sigmoid biner adalah sebagai berikut :



Gambar 2.11 Grafik Fungsi Sigmoid Biner
Sumber : [SIA-05]

Keterangan :

e : Bilangan eural yaitu 2.71828

x :Hasil penjumlahan dari sinyal-sinyal *input*

y : Fungsi untuk mengaktivasi nilai *x*

y' : Turunan dari *f(x)*

5. Fungsi Sigmoid Bipolar (*Symetric Hard Limit*)

Fungsi sigmoid bipolar ini sama dengan fungsi sigmoid biner, tetapi fungsi ini berada dalam range -1 sampai 1. Berikut fungsi aktivasi sigmoid bipolar :

$$y = \frac{2}{1+e^{-x}} - 1 \qquad (2\text{-}6)$$

$$y' = \frac{(1+f(x))(1-f(x))}{2} \qquad (2\text{-}7)$$

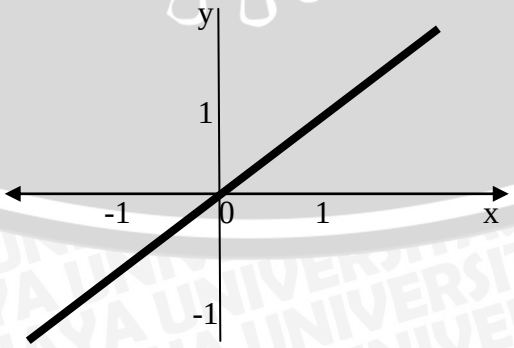
Fungsi aktivasi yang sering digunakan dalam jaringan syaraf tiruan *backpropagation* adalah fungsi sigmoid biner dengan range [0,1] dan fungsi sigmoid bipolar dengan range [-1,1].

6. Fungsi Identitas (*Linear*)

Fungsi identitas sering digunakan jika kita menginginkan keluaran jaringan berupa sembarang bilangan riil, bukan hanya pada range (0,10 atau (-1,1)

$$y = x \qquad (2\text{-}8)$$

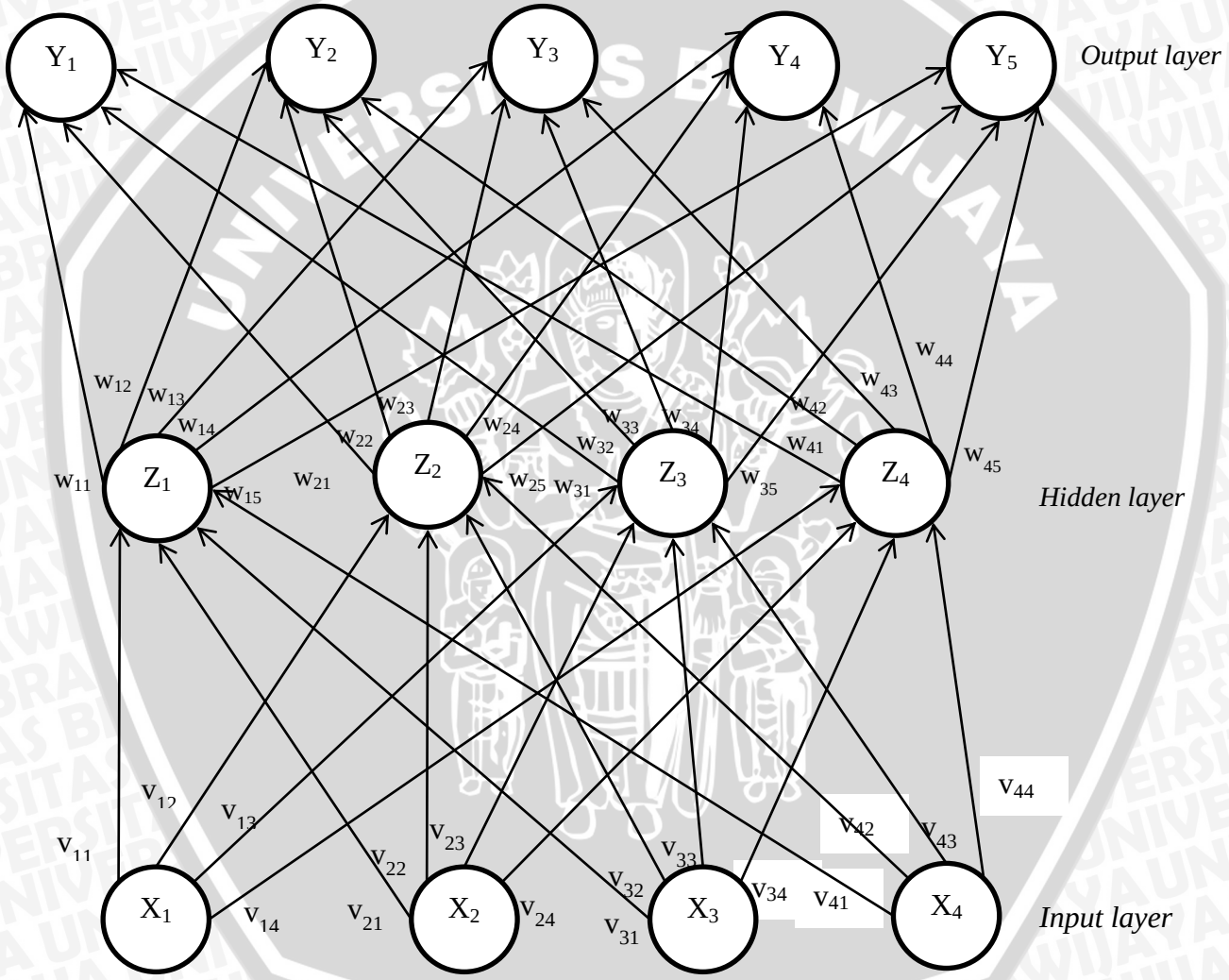
Gambar dari fungsi aktivasi identitas adalah sebagai berikut :



Gambar 2.12 Grafik Fungsi Identitas
Sumber : [SIA-05]

2.6.4 Backpropagation

Backpropagation adalah metode penurunan gradien untuk meminimalkan kuadrat eror luaran. Ada tiga tahap yang harus dilakukan dalam pelatihan, yaitu tahap perambatan maju(*forward propagation*), tahap perambatan balik, dan tahap perubahan bobot dan bias (Sutujo, 2011). Arsitektur jaringan ini terdiri dari *inputlayer*, *hiddenlayer*, dan *outputlayer*seperti pada Gambar 2.13 :



Gambar 2.13 Arsitektur*Backpropagation*
Sumber :[SUT-11]

Keterangan :

X_1, X_2, X_3, X_4 = lapisan *input*

$v_{11}, v_{12}, \ldots\ldots, v_{44}$ = matriks bobot antara *input*dan *hiddenlayer*

Z_1, Z_2, Z_3, Z_4 = *hiddenlayer*

$w_{11}, w_{12}, \ldots\ldots, w_{45}$ = matriks bobot antara *hiddenlayer*dan *output*

Y_1, Y_2, Y_3, Y_4, Y_5 = lapisan *output*

Algoritma *backpropagation*akan melakukan inisialisasi bobot (ambil nilai random yang cukup kecil), arsitektur yang dibangun tidak menggunakan bias pada *inputlayer* dan *hiddenlayer*.Kemudian selama kondisi berhenti bernilai salah, kerjakan :

Tahap Perambatan maju (*feedforward)*

1. Setiap unit *input* (X_i, i=1,2,3,…,n) menerima sinyal X_i dan meneruskan sinyal tersebut ke semua unit pada lapisan tersembunyi.

   Keterangan :

   x_i = unit *input* i

   i = indeks untuk unit *input*

   n = jumlah unit masukan

2. Setiap unit tersembunyi (Z_j, j=1,2,3,…,p) menjumlahkan bobot sinyal *input* dengan persamaan (2-1)berikut,

   $$z\_in_j = \sum_{i=1}^{n} x_i v_{ij} \quad (2\text{-}9)$$

   dan menerapkan fungsi aktivasi untuk menghitung sinyal *output*nya pada persamaan (2-2):

   $$z_j = f(z\_in_j) \quad (2\text{-}10)$$

   Biasanya fungsi aktivasi yang digunakan adalah fungsi sigmoid biner dapat dilihat pada persamaan (2-4) dan (2-5), kemudian mengirimkan sinyal tersebut kesemua unit *output*.

   Keterangan :

   z_j = unit *hidden* j

   j = indeks untuk unit *hidden*

   p = jumlah unit tersembunyi

   v_{ji} = bobot dari unit *input* i menuju unit *hidden* j

   z_in_j = *input* jaringan ke z_j

3. Setiap unit *output*(Y_k, k=1,2,3,…,m) menjumlahkan bobot sinyal *output*pada persamaan (2-11):

   $y\_in_k = \sum_{j=1}^{p} Z_j w_{jk}$ (2-11)

   Kemudian menerapkan fungsi aktivasisigmoid biner pada persamaan (2-12) dan (2-12) untuk menghitung sinyal *output*nyadengan persamaan:

   $y_k = f(y_{-in_k})$ (2-12)

   Keterangan :

   k = indeks untuk unit *output*

   m = jumlah unit *output*

   y_in_k = *input* jaringan ke y_k

   w_{jk} = bobot dari unit *hidden*menuju ke unit *output*k

   y_k = unit *output* k

Tahap Perambatan balik (*backpropagation)*

1. Setiap unit *output*(Y_k, k=1,2,3,…,m) menerima pola target yang sesuai dengan pola *input* pelatihan, kemudian hitung *error* dengan menggunakan persamaan (2-13) berikut.

   $\delta_k = (t_k - y_k) y_k (1 - y_k)$ (2-13)

   Kemudian untuk menghitung suku perubahan bobot menggunakan *momentum*

   $\Delta w_{jk} = (\alpha \delta_k z_j) + (\eta\, \alpha \delta_k z_j)$ (2-14)

   Sekaligus mengirimkan δ_k ke unit-unit yang ada di lapisan paling kanan.

   Keterangan :

   t_k = target

   δ_k = informasi *error*pada unit *output*y_k ke unit *hidden*

   Δw_{jk} = perubahan bobot dari unit *hiddenlayer*menuju unit *outputlayer*

   α = *learning rate*

   η = *momentum*

2. Setiap unit tersembunyi (Z_j, j=1,2,3,…,p) menjumlahkan delta *input*-nya (dari unit-unit yang berada pada lapisan di kanannya) pada persamaan (2-15):

   $\boldsymbol{\delta\_in_j = \sum_{k=1}^{m} \delta_k w_{jk}}$ (2-15)

Untuk menghitung informasi *error*, kalikan nilai ini dengan turunan dari fungsi aktivasinya pada persamaan (2-16).

$$\delta_j = \delta\_in_j f(\text{z\_in}_j) \quad (2\text{-}16)$$

kemudian hitung koreksi bobot untuk menggunakan *momentum* dengan persamaan berikut.

$$\Delta v_{ij} = (\alpha \delta_j x_i) + (\eta\, \alpha \delta_j x_i) \quad (2\text{-}17)$$

Keterangan :

δ_j = informasi *error* dari *outputlayer* ke unit *hiddenz*$_j$

δ_in_j = informasi *error* pada *hiddenlayer*

Δv_{ij} = perubahan bobot dari unit *inputlayer*menuju unit *hiddenlayer*

Tahap Perubahan Bobot

1. Setiap unit *output*(Y_k, k=1,2,3,…,m) dilakukan perubahan bobot (j=0,1,2,…,p) dengan persamaan (2-12) berikut.

   $$w_{jk}\ (baru) = w_{jk}\ (lama) + \Delta w_{jk} \quad (2\text{-}18)$$

   Setiap unit tersembunyi (Z_j, j=1,2,3,…,p) dilakukan perubahan bobot (i=0,1,2,…,n) dengan persamaan (2-13) berikut.

   $$v_{ij}\ (baru) = v_{ij}\ (lama) + \Delta v_{ij} \quad (2\text{-}19)$$

   Keterangan :

   $w_{kj}(baru)$ = bobot baru dari unit *hiddenlayer*menuju unit *outputlayer*

   $w_{kj}(lama)$ = bobot lama dari unit *hiddenlayer*menuju unit *outputlayer*

   $v_{ij}(baru)$ = bobot baru dari unit *inputlayer*menuju unit *hiddenlayer*

   $v_{ij}(lama)$ = bobot lama dari unit *inputlayer*menuju unit *hiddenlayer*

2. Tes kondisi berhenti

2.6.5 Normalisasi Data

Normalisasi data merupakan suatu proses untuk melakukan skala data sehingga suatu data berada dalam suatu rentang nilai tertentu. Proses normalisasi dilakukan untuk menghasilkan, data yang baik, karena kualitas data yang digunakan mempengaruhi luaran yang dihasilkan. Metode normalisasi data yang digunakan dalam penelitian ini adalah metode *Min-Max*. Normalisasi *Min Max*

merupakan metode yang melakukan transformer linear terhadap data asli, persamaan matematika dari metode ini adalah sebagai berikut [SIA-05] :

$$x' = \left(0.8 * \frac{x - \min value}{\max value - \min value} \right) + 0.1 \quad (2\text{-}20)$$

Keterangan :

x = data

x' = hasil mormalisasi

min value = nilai minimum data

max value = nilai maksimum data

2.6.6 Denormalisasi

Denormalisasi adalah proses merubah bilangan asli normalisasi kedalam luaran nilai data asli atau bukan dalam rentang nilai tertentu. Persamaan matematika dari denormalisasi adalah sebagai berikut [SIA-05] :

$$x'' = \frac{(\max value - \min value) * (x' - 0.1)}{0.8} + \min value \quad (2\text{-}21)$$

Keterangan :

x" = Hasil Denormalisasi

2.6.7 Jumlah *HiddenLayer*

Penelitian yang dilakukan Shibata dan Ikeda (2009) berjudul "*Effect of Number of Hidden Neurons on Learning in Large-scale Layered Neural Networks* " meneliti efek dari stabilitas dari neuron tersembunyi atau *hiddenlayer*di proses pembelajaran jaringan syaraf tiruan. Hal ini diterapkan di masalah pemetaan nomor acak [KSB-09]. Rumus untuk menentukan jumlah node tersembunyi atau *hiddenlayer*adalah sebagai berikut :

$$p = \sqrt{n * m} \quad (2\text{-}22)$$

Keterangan :

p =jumlah *hiddenlayer*

n = jumlah *inputlayer*

m = jumlah *outputlayer*

Contoh :

Jika pada sebuah arsitektur *backpropagation multilayer* yang ingin dibagung ditentukan jumlah jaringan *input* adalah 4 dan jumlah jaringan *output* adalah 5 tentukan jumlah *hiddenlayer* ?

Jawab :

n = 4

m = 5

p = $\sqrt{4*5}$

= $\sqrt{20}$

= 4.4721≈ 4

Jadi jumlah *hiddenlayer* yang disarankan berdasarkan Shibata dan Ikkeda adalah 4 buah*hiddenlayer*.

2.6.8 *Mean Square Error*(MSE)

Mean Square Error(MSE) merupakan salah satu metode yang dapat digunakan untuk melakukan evaluasi kesalahan. Pada jaringan syaraf tiruan MSE dapat digunakan untuk meningkatkan optimasi dari proses yang dilakukan. Persamaan matematika dari MSE adalah sebagai berikut [EDO-14]:

$$\boldsymbol{MSE = \frac{1}{k}\sum_{k=1}^{n}(y_k - t_k)^2} \quad (2\text{-}23)$$

Keterangan :

n = jumlah data

y_k = nilai *output* data

t_k = nilai target data

2.7 *Particle Swarm Optimization* (PSO)

Particle Swarm Optimization adalah salah satu metode optimasi yang terinspirasi dari perilaku gerakan kawanan hewan seperti ikan (*school of fish*), hewan herbivora (*herd*), dan burung (*flock*) yang kemudian tiap objek hewan disederhanakan menjadi sebuah partikel. Suatu partikel dalam ruang memiliki posisi yang dikodekan sebagai vektor koordinat. Vektor posisi ini dianggap

sebagai keadaan yang sedang ditempati oleh suatu partikel di ruang pencarian [KEN-95]. Setiap posisi dalam ruang pencarian merupakan alternatif solusi yang dapat dievaluasi menggunakan fungsi objektif. Setiap partikel bergerak dengan kecepatan. *Particle Swarm Optimization* menerapkan sifat masing-masing individu dalam satu kelompok besar. Kemudian menggabungkan sifat-sifat tersebut untuk menyelesaikan permasalahan.

Particle Swarm Optimization pertama kali dimunculkan pada tahun 1995, sejak saat itulah para peneliti banyak menurunkan dan mengembangkan metode PSO. Particle Swarm Optimization memiliki kesamaan sifat dengan teknik komputasi seperti Algoritma Genetika (*Genetic Algorithm*). Meskipun GA bukanlah sebuah pendekatan yang terbaik. Akan tetapi telah menunjukkan bahwa kinerja algoritma PSO lebih cepat daripada GA dalam menyelesaikan permasalahan komputasi telah menunjukkan bahwa algoritma PSO dapat memperoleh jadwal yang lebih baik daripada GA [SUN-07].

Sistem PSO diinisialisasi oleh sebuah populasi solusi secara acak dan selanjutnya mencari titik optimum dengan cara update tiap hasil pembangkitan. Dalam konteks optimasi multivariabel, kawanan diasumsikan mempunyai ukuran tertentu atau tetap dengan setiap partikel posisi awalnya terletak di suatu lokasi yang acak dalam ruang multidimensi. Setiap partikel diasumsikan memiliki dua karakteristik yaitu posisi dan kecepatan. Setiap partikel bergerak dalam ruang atau space tertentu dan mengingat posisi terbaik yang pernah dilalui atau ditemukan terhadap sumber makanan atau nilai fungsi objektif. Setiap partikel menyampaikan informasi atau posisi bagusnya kepada partikel yang lain dan menyesuaikan posisi dan kecepatan masing-masing berdasarkan informasi yang diterima mengenai posisi yang bagus tersebut. Dengan demikian perilaku partikel pada PSO didasarkan pada kombinasi dari 3 faktor simpel berikut [SUY-10]:

1. Kohesi - terbang bersama
2. Separasi - jangan terlalu dekat
3. Penyesuaian(alignment) - mengikuti arah bersama

2.8*Bidirectional Particle Swarm Optimization* (BPSO)

Algoritma BPSO pertamakali dikembangkan oleh Zhang dan Huang.Metode ini digunakan karena mampu memberikan solusi dengan distribusi yang baik dalam memecahkan berbagai masalah pengujian standard [YZH-04].Secara umum, prinsip dasar BPSO adalah untuk melatih setiap partikelmencari secara bersamaan di tetangga dan di suatu daerah di mana partikel jarang didistribusikan dengan menggabungkanstrategi pencarian terisolasi dan strategi pencarian tetangga. Sebagai tambahanpopulasi standar, POP, sebuah penyimpanan eksternal (disebut GBEST) digunakan untuk menyimpan partikel terbaik secara *global*.Jika partikel yang ditemukan lebih baik dari partikel yang telah disimpan maka, partikel yang baru akan menggantikan partikel yang disimpan sebelumya. Dengan seperti ini mendorong untuk menghasilkan solusi yang lebih optimal secara global [NAV-11].Langkah-langkah dari algoritma BPSO adalah sebagai berikut :

1. Tentukan parameter BPSO

Pada penelitian ini, dalam menentukan parameter BPSO sesuai standard*particle swarm optimization* (PSO) [DBR-07] yaitu :

- w = bobot
- $c_1 = c_2$ = *Learning vector*
- χ = *Constriction factor*
- *Population size* (jumlah populasi)
- Jumlah iterasi

Learning vector disebut juga sebagai konstanta positif atau koefisien akselerasi. *Constriction factor* disebut sebagi faktor pembatas atau penyempit untuk mengatur besarnya perubahan kecepatan.

2. Inisialisasi

Inisialisasi partikel untuk kawanan ataupopulasi dan kumpulan GBEST yang disimpan dengan posisi dan kecepatan vektor secara acak. Posisi dari setiap partikel dibangkitkan secar random dengan rentang nilai yang telah diberikan oleh

pakar sesuai pada sub bab 2.5.1 - 2.5.5. Sedangkan untuk kecepatan dari setiap parameter *input* dibangkitkan secara random dari rentang nilai [$-V_{max}$, V_{max}] yang didapatkan dari persamaan [DBR-07]:

$$V_{\max} = 60\% * X_{k,\max} \quad (2\text{-}24)$$

Keterangan :

$V_{\max}$ = kecepatan maksimal partikel

$x_{k,\max}$ = nilai maksimal dari parameter *input*

Evaluasi tujuan vektor $F = (f_1, f_2)$ dimana f_1, f_2 didapatkan dengan persamaan minimalisasi :

$$f_1 = \frac{1}{y_k(x)} \quad (2\text{-}25)$$

$$f_2 = \frac{1}{y_k(x)} \quad (2\text{-}26)$$

Keterangan :

$y_k(x)$= Luaran pengaruh pada tanaman

f_1= posisi vektor ke-1

f_2= posisi vektor ke-2

3. Menghasilkan keturunan partikel

Dua keturunan $child_1$ dan $child_2$akan dibangkitkan untuk setiap partikel pada kumpulannya, posisi dan kecepatan keturunan diwarisi sepenuhnya dari *parent*nya. Diasumsikan bahawa $child_1$ dan $child_2$berasal dari partikel ke-*a*, posisi dan kecepatan mereka dilambangkan dengan X_l, Xr,V_l,V_r dimana :

$$Xi = X_l = X_r \quad (2\text{-}27)$$

$$V_i = V_l = Vr \quad (2\text{-}28)$$

Keterangan :

X_i =Posisi partikel ke-*i*

X_l=Posisi partikel child 1

X_r=Posisi partikel child 2

V_i= Kecepatan partikel ke-*i*

V_l= Kecepatan partikel child 1

V_r= Kecepatan partikel child 2

4. Mencari GBEST dan PBEST untuk *child*$_1$

Untuk setiap anggota dalam GBEST, dihitung *niche count* (NC)pada strategi pencarian terisolasi (*isolated searching strategy*)dengan persamaan [YZH-04] :

$$d_{ij} = \sqrt{\sum_{t=1}^{n} (f_t(x_i)) - f_t(x_j))2} \quad (2\text{-}29)$$

Keterangan :

d_{ij}= *euclidean distance*

n= jumlah populasi

t= 1,2,..n,

i = 1,2,..p,

j = 1,2,..$pi \neq j$

kemudian nilai fungsi *share* dihitung dengan persamaan :

$$s(d_{ij}) = \begin{cases} 1 - \frac{d_{ij}}{\sigma_{share}} & d_{ij} < \sigma_{share} \\ 0 & \boldsymbol{d_{ij} > \sigma_{share}} \end{cases} \quad (2\text{-}30)$$

Untuk menghitung niche radius digunakan persamaan [DEB-89] :

$$\sigma_{share} = \frac{r}{\sqrt[p]{q}} \quad (2\text{-}31)$$

$$r = \frac{1}{2}\sqrt{\sum_{k=1}^{p} \left(x_{k,\max} - x_{k,\min}\right)^2} \quad (2\text{-}32)$$

Keterangan :

r = jarak atau *radius*

p= jumlah parameter *input* GBEST

k=indeks *input*GBEST

$x_{k,\max}$ = nilai maksimal dari parameter *input* GBEST

$x_{k,\max}$ = nilai minimal dari parameter *input* GBEST

q= jumlah anggota GBEST

σ_{share} adalah niche radius yang merupakan sebuah nilai pinalti atau *fitness* untuk mengurangi nilai keragaman pada populasi untuk mengurangi kemunculan keturunan atau generasi selanjutnya [WID-13]. Selanjutnya, *niche count*untuk dengan persamaan pada anggota GBEST $i = 1,2,\ldots\ldots p$:

$$\boldsymbol{niche(i) = \sum_{j=1}^{p} s(d_{ij})} \quad (2\text{-}33)$$

Anggota dengan nilai NC terkecil akan dipilih sebagai GBEST untuk $child_1$. Sedangkan nilai PBEST untuk $child_1$akan tetap sama dengan PBEST *parent*.

5. Mencari GBEST dan PBEST untuk $child_2$

Setiap anggota dari GBEST, dihitung Euclidean distance diukur jarak setiap anggota terhadap $child_2$ dengan persamaan 2.21.Anggota dengan nilai NC terkecil akan dipilih sebagai GBEST untuk $child_2$. Sedangkan nilai PBEST untuk $child_2$akan sama dengan PBEST *parent*.

6. Perbaharui Kecepatan dan Posisi$child_1$ dan $child_2$

Menghitung perubahan kecepatan dari setiap anak dengan persamaan :

$$\boldsymbol{v_{id,new} = w\, v_{id} + c_1 r_1 (p_{id} - x_{id}) + c_2 r_2 (p_{id} - x_{id})} \quad (2\text{-}34)$$

$$\boldsymbol{x_{id,new} = x_{id} + \chi V_{id}} \quad (2\text{-}35)$$

Keterangan :

$v_{id,new}$ = *update* kecepatan baru

v_{id} = kecepatan sebelumnya

r_1,r_2=bilangan random antara (0,1)

P_{id}= posisi PBEST

X_{id}= posisi dari partikel ke-*i*

$x_{id,new}$ = *update*posisi partikel ke-*i*

7. Perbaharui GBEST

Jika nilai X'_l dan$X'r$layak dan tidak terdominasi dengan anggota di GBEST. Proses seleksi ini terjadi jika kedua nilai vektor (f1,f2) dari partikel lebih kecil dari anggota yang ada pada GBEST. GBEST diperbaharui dengan menambahkan X'_l dan $X'r$ dan mengeliminasi anggota lainnya yang nilai vektornya lebih kecil terhadap X'_l dan $X'r$. Jika nilai X'_l dan$X'r$ tidak layak maka tidak ada perubahan pada anggota GBEST.

8. Perbaharui partikel

Jika X'_l layak kemudian partikel diperbaharui dengan membandingkan nilai terbaik antara $child_1$ dan $child_2$

$X_i = X'_l$ dan$V_i = V'_l$ (2-36)

Selanjutnya, X_i baru akan dibandingkan dengan nilai P_i dan yang terbaik akan dipilih salah satu sebagi P_i terbaru yang menjadi *parent.*

$P_i = X_i$ (2-37)

Jika X'_l tidak layak tidak ada pembaharuan pada partikel. Kondisi ini diulang kembali dari langkah ke 4 sampai langkah ke 9, untuk semua partikel yang dibangkitkan dan akhir iterasi.

9. Kondisi berhenti

Pada iterasi terakhir, setiap anggota pada GBEST akan dianggap optimal dalam memberikan solusi pada permasalahan. Namun untuk memahami agar mudah dan praktis setiap solusi di ubah ke nilai sebenarnya dari $T(x), H(x)$ atau *output* dari ANN lainnya. Misalnya :

$$y_k(x) = \frac{1}{f_1} \quad (2\text{-}38)$$

$$y_k(x) = \frac{1}{f_2} \quad (2\text{-}39)$$

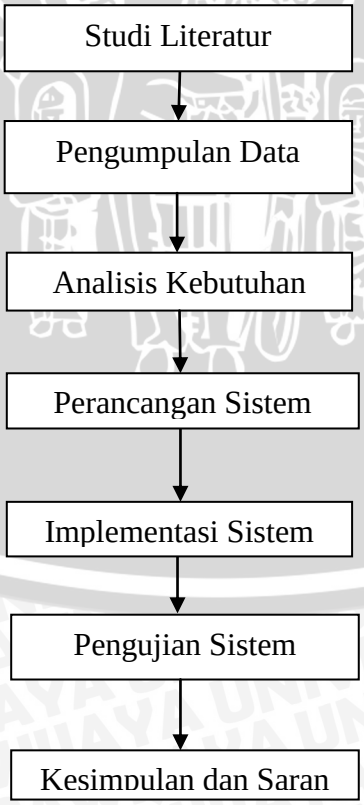
Dipilih salah satu dari anggota GBEST yang terbaik sebagai solusi akhir yang paling optimal.

BAB III
METODOLOGI PENELITIAN DAN PERANCANGAN

Pada bab metodologi penelitian dan perancangan ini dijelaskan tahapan dalam pembuatan sistem optimasi dosisi pupuk pada tanaman palawija menggunakan ANN dan BPSO. Tahapan penelitian meliputi analisis kebutuhan sistem, deskripsi umum sistem, data yang digunakan, perancangan sistem, proses ANN dan BPSO, perancangan *user interface*, dan perancangan uji coba dan evaluasi.

3.1 Tahapan Penelitian

Tahapan penelitian ini menjelaskan langkah-langkah yang akan dilakukan pada penelitian integrase metode ANN dan BPSO dalam optimasi dosis pada tanaman palawija. Tahapan penelitian ini ditunjukkan pada Gambar 3.1 seperti berikut ini :



Gambar 3.1 Metode Penelitian

Pada Gambar 3.1 Metode Penelitian tahapan penelitian dimulai dari pengumpulan data, analisis dan perancangan sistem, implementasi sistem, pengujian sistem dan evaluasi, serta pengambilan kesimpulan dan saran.

3.1.1 Studi Literatur

Studi literatur dilakukan untuk mempelajari tentang dasar teori yang digunakan untuk menunjang penulisan dan penegerjaan penelitian.Sumber literatur berupa*paper*, jurnal, buku, *e-book* sumber lisan dan penelitian sebelumnya.Referensi utama yang berkaitan dengan algoritma ANN, BPSO, bahasa pemrograman C# dan pengujian sistem.

3.1.2 Pengumpulan Data

Pengumpulan data dilakukan dengan wawancara kepada pakar Dr. Widowati Dekan, Fakultas Pertanian Universitas Tribhuwan Tunggadewi, mencari dan mengumpulkan hasil penelitian skripsi Fakultas Pertanian Universitas Brawijaya.

3.1.3 Analisis Kebutuhan Sistem

Analisis kebutuhan sistem berisi langkah-langkah untuk menganalisis semua kebutuhan agar sistem dapat berjalan.Analisis ini terdiri dari deskripsi dan analisis kebutuhan data yang diperlukan.

3.1.3.1 Deskripsi Umum Sistem

Sistem yang dibangun pada penelitian ini adalah system yang mengintegrasi metode A*rtificial Neural Network* (ANN) menggunakan b*ackpropagation* dan *Bidirectional Particle Swarm Optimization* (BPSO) untuk optimasi pemberian dosis pada berbagai jenis pupuk tanaman palawija. Algoritma ANN digunakan sistem untuk menentukan pengaruh pada tanaman.Jenis pupuk yang digunakan adalah pupuk UREA, SP_{36}, KCL, dan Biochar Sampah Kota. Setiap jenis pupuk diberikan dalam takaran ton/ha. *User*akan memasukkan nilai dosis yang diberikan pada setiap jenis pupuk pada tanaman.Data masukan akan

diproses, kemudian sistem akan memberikan hasil pengaruh terhadap tanaman dari pemberian dosis pupuk tersebut.Setiap jenis pupuk diberikan 6 perilaku kombinasi dosis pupuk yang berbeda berdasarkan pakar. 6 perilaku kombinasi dosis ini memberikan pengaruh yang berbeda beda pada setiap pengaruh tanaman yang diamati.Pengaruh pada tanaman berdasarkan pakar yang perlu diamati adalah bobot kering tanaman dalam satuan (ton/ha), bobot kering ini untuk mengamati pengaruh pertumbuhan pada tanaman. Selanjutnya, bobot 1000 butir dalam satuan (gram), diameter tongkol, panjang tongkol dalam satuan (cm) dan hasil produksi jagung dalam satuan (ton/ ha).Dari 6 perilaku kombinasi dosis ini, masing-masing perilaku kombinasi dosis diulangi sebanyak 30 kali, sehingga total data berjumlah 180 perilaku kombinasi dosis masing-masing jenis pupuk dan pengaruh pada tanaman.

Hasil dari data luaran ANN, akan dijadikan data masukan oleh algoritma BPSO. Dari data luaran ANN, akan dipilih dua pengaruh pada tanaman yang akan dimaksimalkan nilainya, BPSO akan membangkitkan secara random setiap partikel dosis pada jenis pupuk kemudian, menghitung nilai vektor dari dua *output*pengaruh pada tanaman yang dipilih jika lebih baik akan disimpan dalam GBEST.Hal ini akan dilakukan sampai batas iterasi maksimum yang ditentukan, sehingga didapatkan dosis pada setiap jenis pupuk dengan nilai hasil vektor terkecil yang tersimpan didalam GBEST sebagai partikel yang terpilih dari dua *output*pengaruh pada tanaman yang diharapkan. Sehingga *user*dapat memberikan jumlah takaran dosis dengan berbagai jenis pupuk untuk tanaman sesuai dengan hasil yang *user* inginkan.

3.1.3.2 Data yang Digunakan

Data yang digunakan pada penelitian ini adalah :

1. Data jenis pupuk yang sudah digunakan dan diteliti pengaruhnya terhadap tanaman.
2. Data takaran range dosis pemberian pada masing-masing jenis pupuk.
3. Data hasil fisik tanaman jagung dari berbagai prilaku kombinasi pemberian dosis.

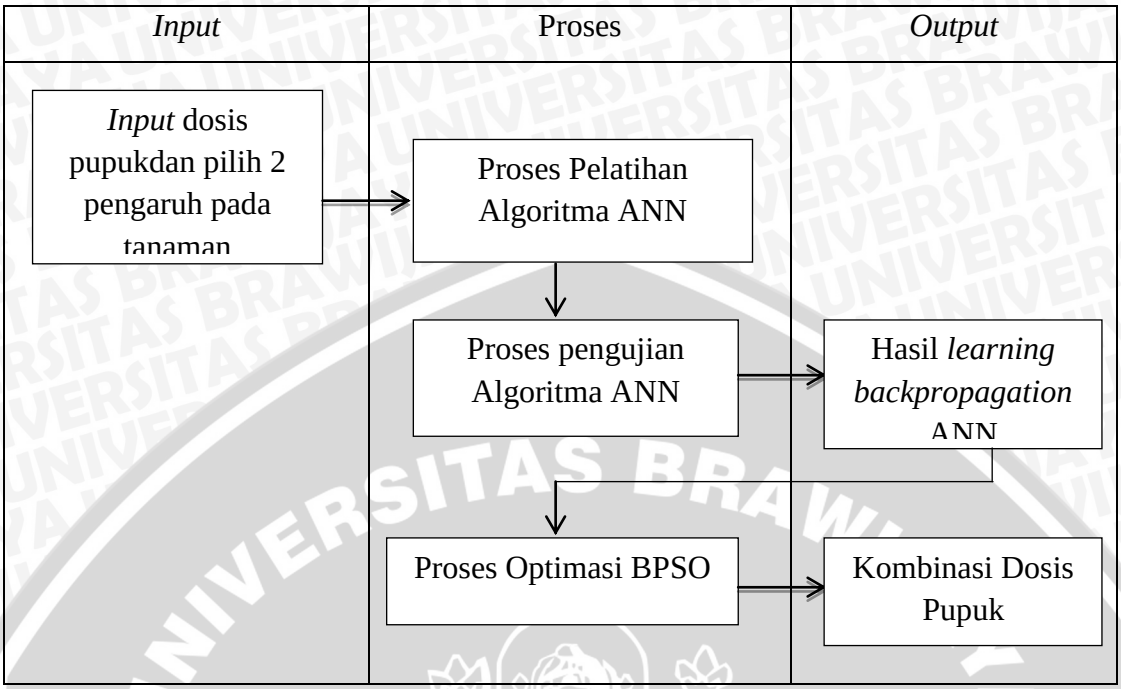
Analisis kebutuhan data beserta sumber, metode, dan kegunaannya dijelaskan pada Tabel 3.1 Analisi Kebutuhan Data Penelitianberikut ini :

Tabel 3.1 Analisis Kebutuhan Data Penelitian

No	Kebutuhan Data	Sumber Data	Metode	Kegunaan Data
1.	Data hasil analisis Jenis-jenis pupuk yang digunakan pada tanaman jagung	Literatur buku, jurnal, paper, Pakar Dr. Ir Widowati Dekan fakultas Pertanian Universitas Tribhuwana Tunggadewi	Literatur, wawancara	Sebagai *input*an jenis-jenis pupuk yang akan digunakan untuk proses *backpropagation*
2.	Data *range* dosis pupuk berdasarkan jenis – jenis pupuk	Pakar Dr. Ir Widowati, Dekan fakultas Pertanian Universitas Tribhuwana Tunggadewi	Literatur, wawancara	Digunakan untuk memberikan dosis pada jenis pupuk sesuai dengan takaran dosis umumnya
3.	Data pengaruh pemberian terhadap tanaman jagung	Literatur buku, jurnal, paper, Pakar Dr. Ir Widowati, Dekan fakultas Pertanian Universitas Tribhuwana Tunggadewi	Wawancara	Digunakan pada proses pelatihan, pengujian *Backpropagaion* sebagai target luaran
4.	Data *range* pengaruh pada tanaman jagung	Literatur buku, jurnal, paper, Pakar Dr. Ir Widowati, Dekan fakultas Pertanian Universitas Tribhuwana Tunggadewi	wawancara	Digunakan untuk perhitungan dalam proses BPSO untuk optimasi

3.1.4 Perancangan Sistem

Setelah mengetahui analisis kebutuhan yang diperlukan pada sistem maka proses dilanjutkan pada langkah perancangan sistem. Berdasarkan analisis kebutuhan sistem maka arsitektur rancang sistem yang dibagun untuk optimasi dosis pupuk pada tanaman palawija menggunakan integrasi metode ANN dan BPSO dijelaskan dengan skenario pengujian pada Gambar 3.2 Diagram Blok Sistemberikut :



Gambar 3.2 Diagram Blok Sistem

Pada Gambar 3.2 di atas yang akan dilakukan oleh pengguna adalah memasukkan dosis yang ingin diberikan pada masing-masing pupuk. Pemberian dosis ini berdasarkan rentang dosis yang lazim atau wajar untuk diberikan kepada tanaman palawija. Dosis ini akan ditentukan pengaruh pemberiannya pada tanaman palawija menggunakan ANN. Sebelumnya Algoritma ANN akan menentukan bobot, target dan pelatihan pada data latih pemberian macam-macam dosis jenis pupuk yang didapat dari sumber penelitian sebelumnya. Sehingga, ANN dapat menentukan prediksi pengaruh pada tanaman dari dosis yang diberikan oleh pengguna. Dari pengaruh pada tanaman palawija yang dihasilkan, pengguna akan memilih dua pengaruh yang akan dibangkitkan sebagi vektor pada algoritma BPSO, berdasarkan pengaruh yang ingin optimalkan nilainya. Hasil akhir dari sistem ini adalah dosisi pupuk yang terbaik untuk tanaman palawija.

3.1.5 Implementasi Sistem

Implementasi yang akan dilakukan pada penelitian ini berdasarkan perancangan sistem yang ingin dibangun. Implementasi dari penelitian ini akan dibangun menggunakan Bahasa pemrograman C# yang mendukung integrasi algoritma ANN dan BPSO. Pada sistem ini yang menjadi *input*adalah jumlah dosis

pada masing-masing jenis pupuk, parameter-parameter pada ANN berupa bobot dan nilai target. Sedangkan *output*yang dihasilkan oleh BPSO diambil dari keanggotaan GBEST dengan nilai vektor yang terbesar, untuk dosis pupuk yang terbaik. Langkah- langkah dalam implmentasi sistem ini adalah sebagai berikut :

1. Membuat *Interface* (anatarmuka).
2. Perhitungan algoritma ANN dengan data latih dan data uji untuk mendapatkan pengaruh pada tanaman palawija.
3. Perhitungan BPSO untuk mengoptimalkan nilai vektor yang dipilih dari ANN.
4. Luaran dosis terbaik masing-masing jenis pupuk anggota GBEST berdasarkan nilai optimasi vektor terbesar.

3.1.6 Pengujian

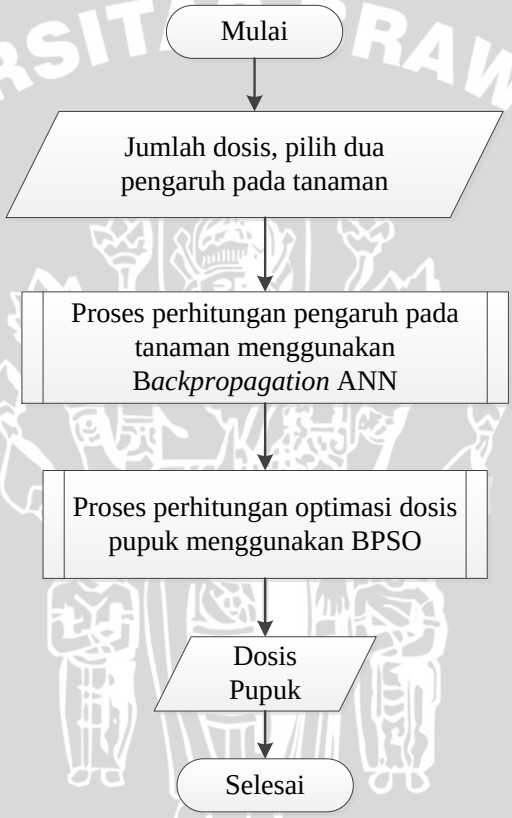
Pengujian dilakukan agar sistem dapat menghasilakn luaran yang diharapkan dan bekerja dengan baik sesuai kebutuhan dari sistem. Pengujian dilakukan berdasarkan varian jumlah data latih data uji, varianjumlah maksimal iterasi, besarnya nilai MSE (*minimum square error*), pengujian terhadap jumlah varian *hiddenlayer*, varian nilai *learning rate* dan nilai *momentum* pada ANN.

3.1.7 Kesimpulan dan Saran

Pengambilan kesimpulan dilakukan setelah semua tahap perancangan, implementasi dan pengujian sistem telah selesai. Hasil akhir dari penelitian ini akan dibandingkan ketika sistem bisa mengahasilkan luaran yang sesuai dengan penelitian yang telah ada sebelumnya, lebih baik, atau lebih buruk. Kesimpulan diambil berdasarkan hasil pengujian sistem dan analisa dari penggunaan integrasi metode ANN dan BPSO sesuai dengan rumusan masalah yang diuraikan sebelumnya.Penelitian ini juga membutuhkan saran sebagai masukan untuk memperbaiki kesalahan yang ada, menyempurnakan penulisan dan memberikan ide pertimbangan untuk perkembangan penelitian selanjutnya.

3.2 Alir Perancangan Sistem

Pada bagian aliran perancangan sistem menjelaskan tentang rancangan proses atau alur kerja sistem dari penelitian yang akan dilakukan secara bertahap. Diagram alur ini akan mempermudah dalam proses perancangan sistem. Aliran perancangan sistem untuk optimasi dosis pupuk pada tanaman palawija menggunakan metode jaringan syaraf tiruan *backpropagation* dan metode *bidirectional particle swarm optimization* pada program yang akan dibangun ditunjukkan pada Gambar 3.3 Diagram Alir Perancangan Sistemberikut :



Gambar 3.3 Diagram Alir Perancangan Sistem

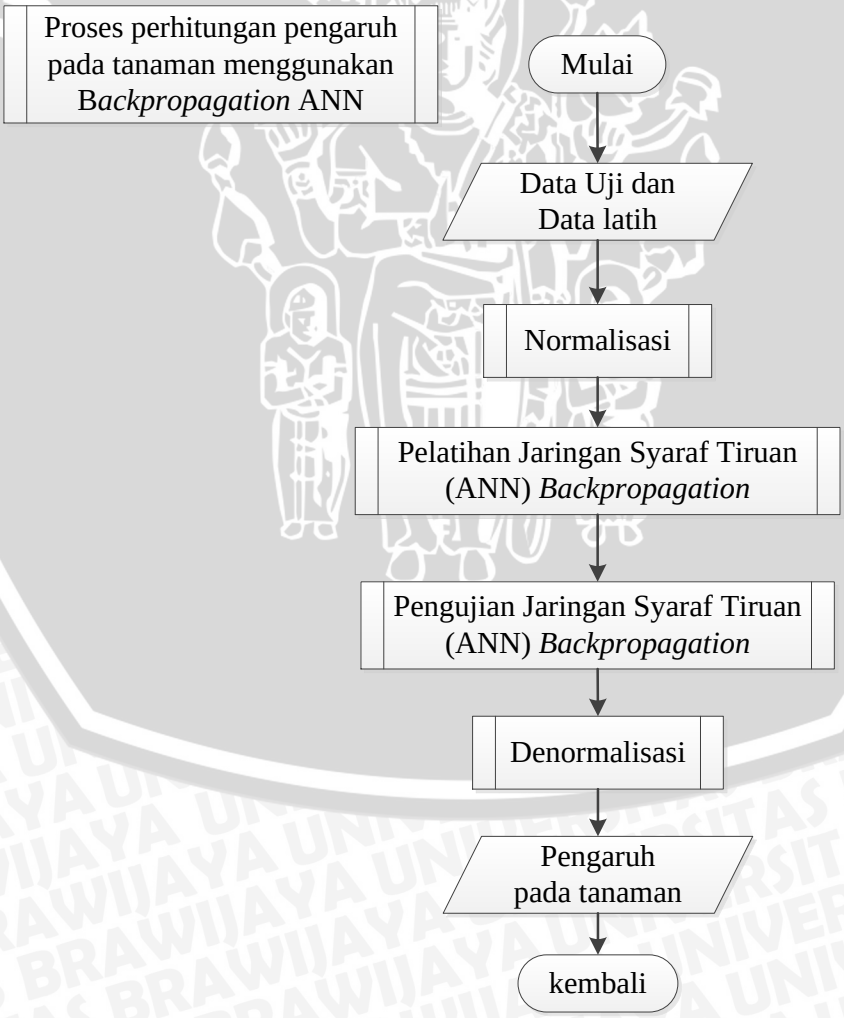
Pada Gambar 3.3 Diagram Alir Perancangan Sistemmenjelaskan mengenai proses yang dilakukan sistem dalam penelitian optimasi dosis untuk palawija menggunakan metode ANN dan BPSO sebagai berikut :

1. Proses diawali dengan memberikan jumlah takaran dosis pada masing-masing jenis pupuk yang ada dan memilih dua hasil *output*pada tanaman yang ingin dioptimalkan.

2. Kemudian, sistem akan melakukan proses penentuan target, bobot dan pengujian dari pelatihan yang akan dilakukan dengan *backpropagation sehingga* ANN dapat menentukan hasil pengaruh pada tanaman berdasarkan dosis yang diberikan oleh pengguna.
3. Setelah itu, BPSO akan mengoptimalkan 2 (dua) pengaruh pada tanaman yang dipilih berdasarkan kombinasi pemberian dosis dari masing-masing jenis pupuk.
4. Hasil akhir akan ditampilkan kombinasi dosis terbaik.

3.2.1 Perhitungan Pengaruh Pemberian Dosis Menggunakan ANN

Untuk menghitung pengaruh kombinasi dosis masing-masing jenis pupuk pada pengaruh hasil tanaman, maka pada penelitian digunakan *backpropagation neural network* yang ditunjukkan pada Gambar 3.4 dibawah ini :



Gambar 3.4 Diagram Alir ProsesANN *Backpropagation*

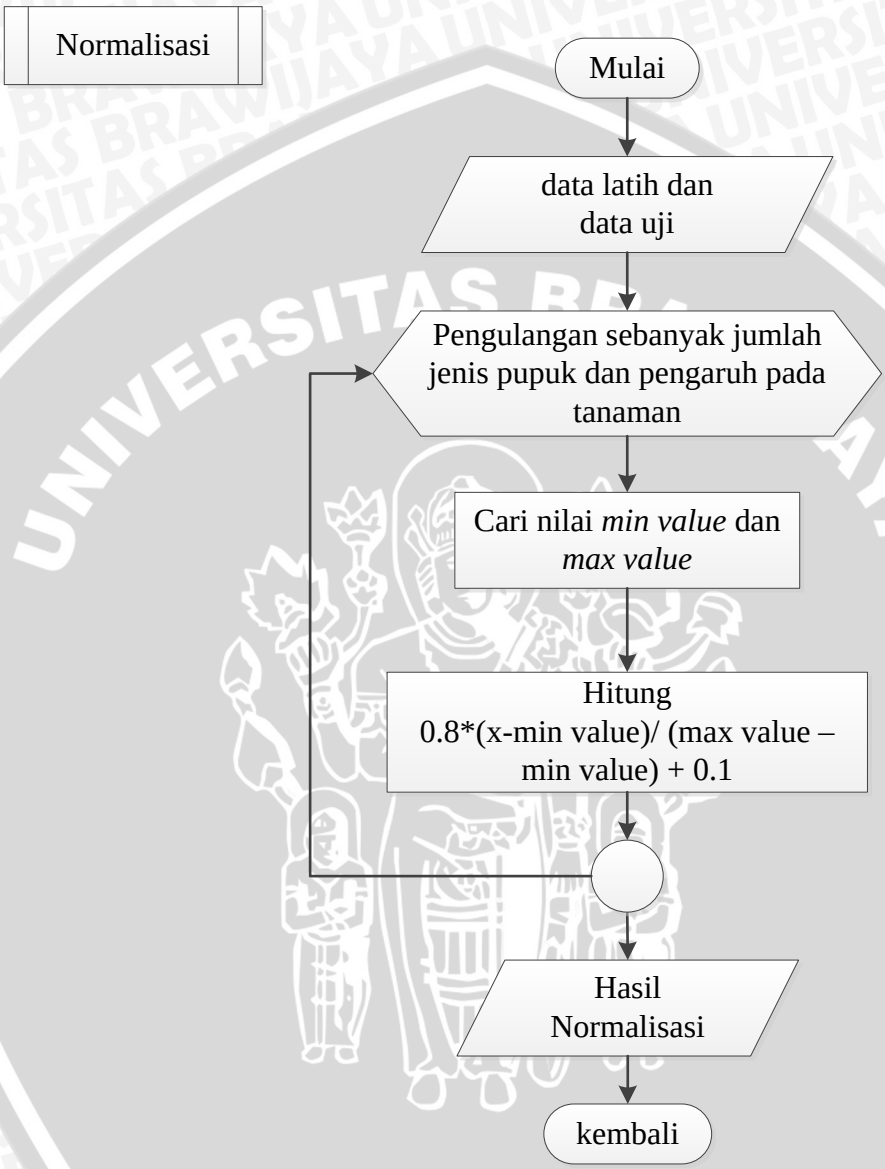
Pada Gambar 3.4 Diagram Alir Proses ANN *backpropagation* adalah sebagai berikut :

1. Proses pertama adalah melakukan normalisasi, untukdata membuat kedalam skala rentang nilai dari (0.1-0.9). Normalisasi dilakukan untuk data yang nilainya lebih dari satu. Untuk proses normalisasi dihitung dengan Persamaan (2-20).
2. Selanjutnya akan dilakukan pelatihan data latih agar ditemukan bobot terbaik untuk menentukan target atau *output* yang diharapkan dari pengaruh pada tanaman
3. Kemudian, dilakukan pengujian pada data uji, untuk membuktikan bahwa bobot yang dihasilkan dari pelatihan sudah mengahasilkan luaran yang terbaik atau mendekati nilai target.
4. Selanjutnya luaran pada tanaman dan *inputan* kombinasi dosis akan di denormalisasi untuk mengembalikan nilai ke rentang semula atau nilai *real* sehingga pemberian pupuk dan pengaruh pada tanaman dapat diamati sesuai dengan nilai aslinya. Proses denormalisasi dihitung sesuai dengan persamaan (2-21).

3.2.1.1 Proses Normalisasi Data

Proses normalisasi digunakan karena jaringan syaraf tiruan dilatih dengan keluaran diskrit. Diskrit artinya adalah keluaran nilai yang tidak saling berhubungan, maka setiap variable harus dinormalisasi terlebih dahulu.Secara umum jika dilakukan normalisasi maka dibutuhkan juga proses denormalisasi pada tahap terakhir setalah data berhasil diolah. Pada perancangan program ini, normalisasi dilakukan untuk nilai *input*kombinasi dosis dan *output*pengaruh pada tanaman yang nilainya diatas satu. Data yang dinormalisasi yaitu Biochar sampah kota, bobot kering tanaman, bobot 1000 butir, dimater tongkol, panjang tongkol dan hasil produksi. Normalisasi dilakukan dengan mencari nilai batas maksimal dan minimal dari data yang digunakan. Kemudian data yang dinormalisasi dikurangi dengan nilai minimum dan dibagi dengan nilai *range* data. Supaya data berada pada interval 0.1 sampai 0.9, maka data dikalikan dengan 0.8 dan ditambah

0.1. Berikut adalah diagram alir proses normalisasi yang dilakukan sebelum data akan digunakan pada metode jaringan syaraf tiruan *backpropagation* ANN dijelaskan pada Gambar 3.5 Diagram Alir Proses Normalisasi dibawah ini:



Gambar 3.5 Diagram Alir Proses Normalisasi

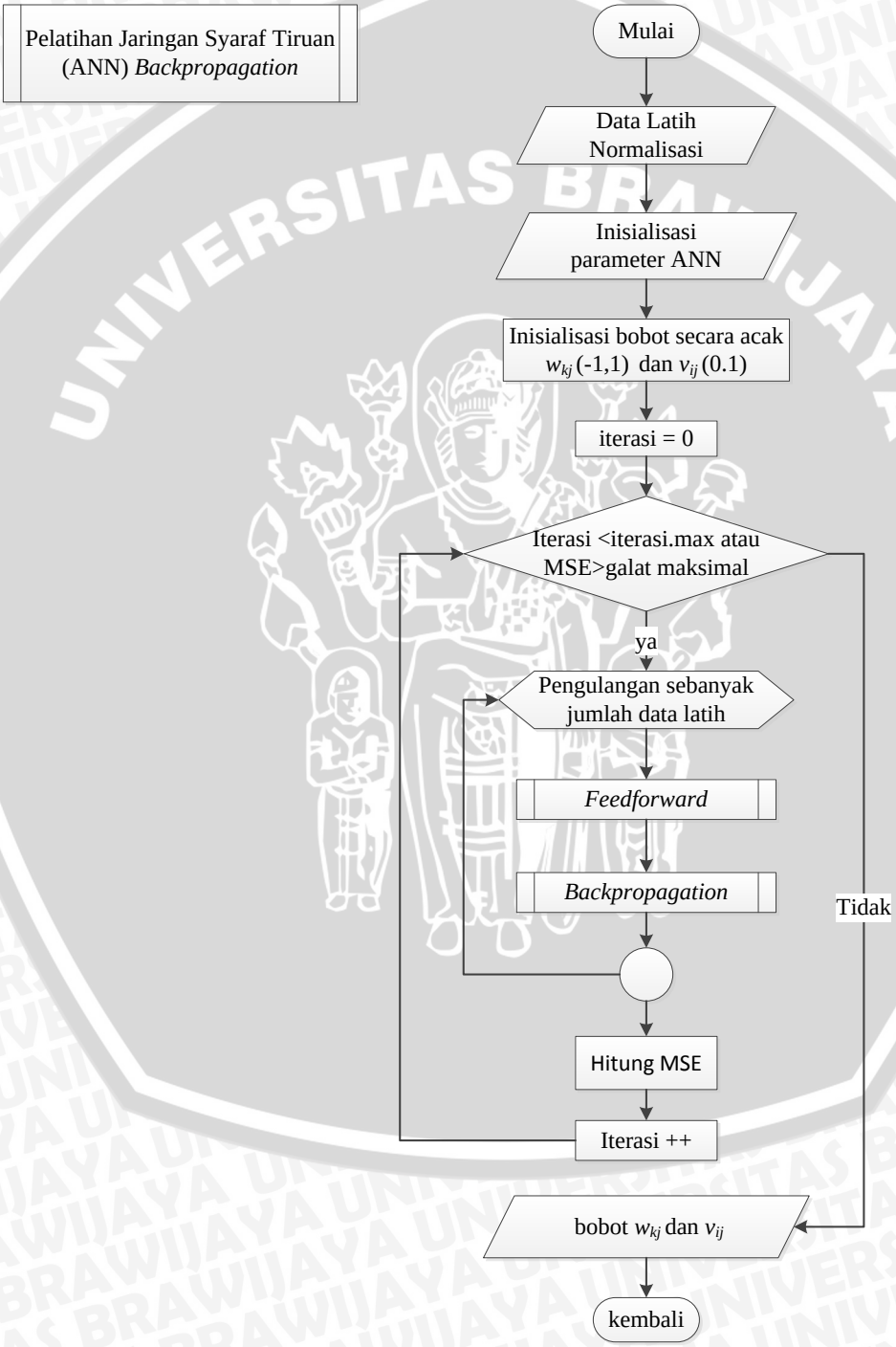
Pada Gambar 3.5 Diagram Alir Proses Normalisasi dilakukan dengan langkah-langkah sebagai berikut :

1. Masukan nilai dari data latih dan data uji.
2. Pengulangan sebanyak jumlah jumlah jenis pupuk dan pengaruh pada tanaman untuk dinormalisasi.
3. Cari batas nilai min *value* dan mak *value* dari masing masing data.

4. Hitung menggunakan persamaan (2-20).
5. Didapatkan data baruhasil luaran normalisasi.

3.2.1.2 Proses Pelatihan ANN*Backpropagatiom*

Tahapan pelatihan *backpropagation* dijelaskan pada Gambar 3.6 berikut :



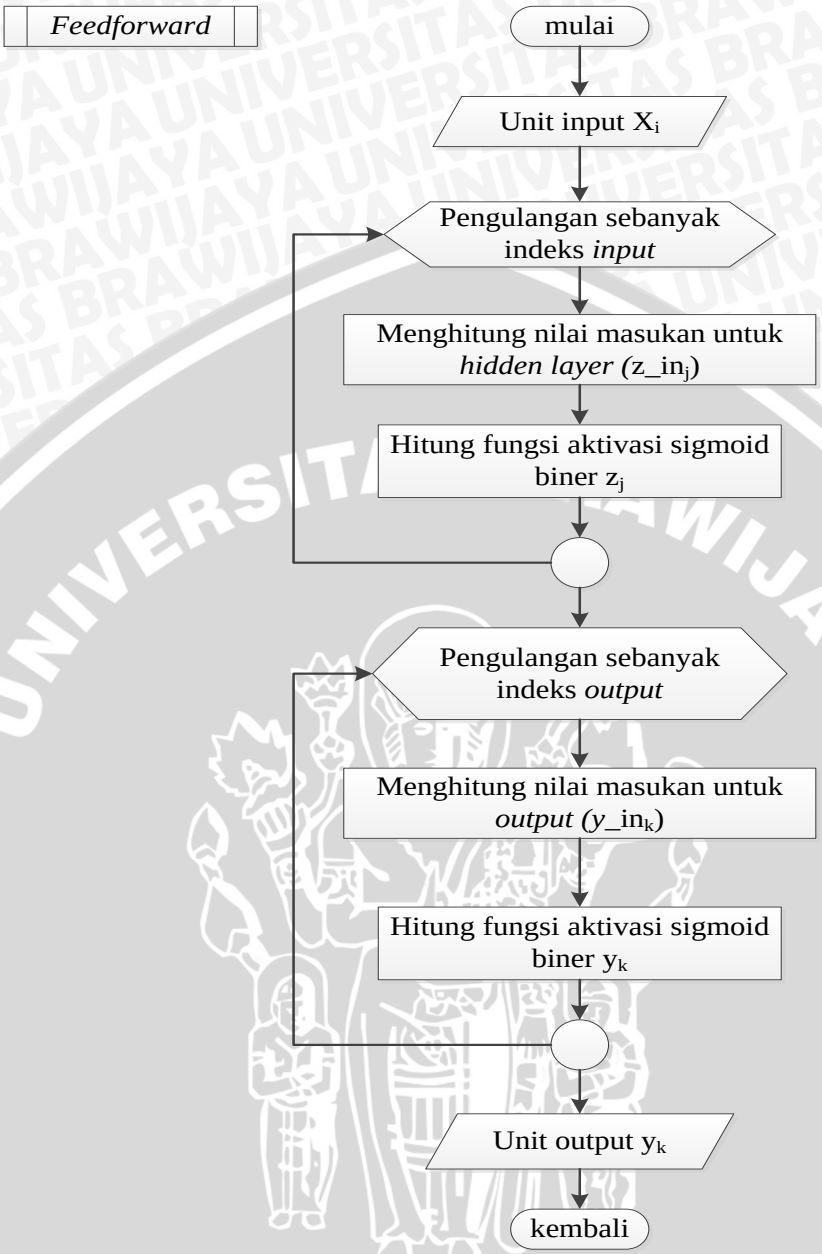
Gambar 3.6 Diagram Alir Proses Pelatihan *Backpropagation*

Tahapan-tahapan pada Gambar 3.6 Diagram Alir Proses pelatihan *Backpropagation*adalah sebagai berikut :

1. Proses awal *input* data latih yamg telah dinormalisasi sebelumnya.
2. Masukan parameter inisialisasi pada ANN *backpropagation*, yaitu jumlah maksimal iterasi, besarnya nilai galat maksimal, jumlah varian *hiddenlayer*, nilai *learning rate* dan nilai *momentum*.
3. Membangkitkan secara acak bobot untuk neuron. Untuk bobot neuron *input*dibangkitkan antara range -1 sampai 1. Sedangkan, untuk bobot neuron *output* dibangkitkan dengan range antara 0 sampai 1.
4. Dimulai dari iterasi ke 0
5. Cek kondisi jika iterasi masih kurang dari iterasi.max atau MSE masih lebih besar dari galat maksimal maka lakukan langkah selanjutnya, jika telah tercapai lakukan langkah ke 11.
6. Lakukan pengulangan ke semua jumlah data latih.
7. Lakukan langkah *feedforward*.
8. Lakukan langkah *backpropagation.*
9. Hitung nilai MSE yang dihasilkan dengan persamaan (2-23).
10. Jika seluruh anggota dalam data latih telah selesai diproses, maka lanjut ke langkah ke 11. Jika belum lakukan kembali langkah 6, dengan menaikan iterasi.
11. Didapatkan perubahan bobot dari v_{ij} dengan persamaan (2-18) dan dan w_{kj}dengan persamaan(2-19).
12. Proses selesai.

3.2.1.3 Proses *feedforward* pada ANN *Backpropagation*

Pada Proses *feedforward* atau perambatan maju, tiap unit *input* menerima sebuah masukan sinyal ke tiap *hiddenlayer*. Kemudian tiap *hiddenlayer* menghitung aktivasinya dan mengirimkan sinyal ke setiap unit *output*. Tiap unit *output* kemudian menghitung aktivasinya untuk membentuk respon pada jaringan dan memberikan pola masukan. Berikut diagram alir dari proses *feedforward* pada Gambar 3.7 Diagram Alir *Feedforward*:



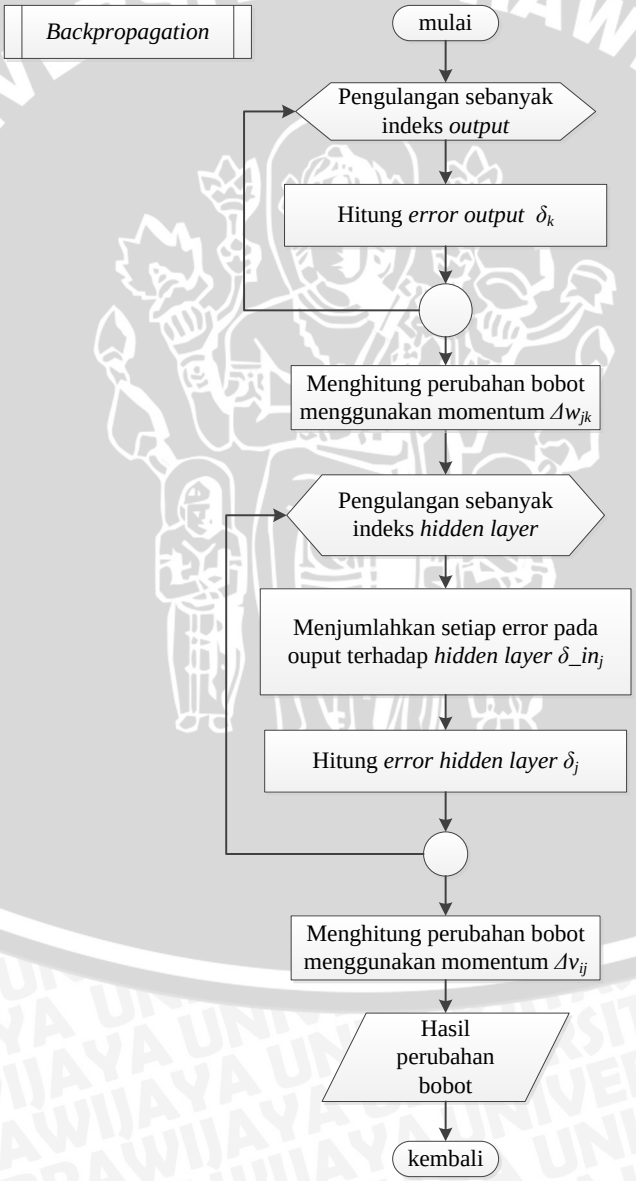
Gambar 3.7 Diagram Alir *Feedforward*

Pada Gambar 3.7 Diagram Alir *Feedforward* dijelaskan tahap-tahap *feedforward* sebagai berikut :

1. Lakukan pengulangan sebanyak indeks *input.*
2. Menghitung indeks *input* terhadap nilai masukan untuk *hiddenlayer*dengan persaman (2-9).
3. Kemudian hitung nilai untuk *hiddenlayer*menggunakan fungsi aktivasi sigmoid biner sesuai persamaan (2-10).

4. Setelah pengulangan selesai untuk semua indeks *input* lakukan pengulangan sebanyak indeks *output*.
5. Lakukan perhitungan nilai masukan untuk*output* sesuai dengan persamaan pada (2-11).
6. Kemudian hitung nilai *output* menggunkan fungsi aktivasi sigmoid biner sesuai persamaan (2-12).
7. Setelah pengulangan untuk semua indeks *output*, proses selesai.

3.2.1.4 Proses Backpropagation ANN



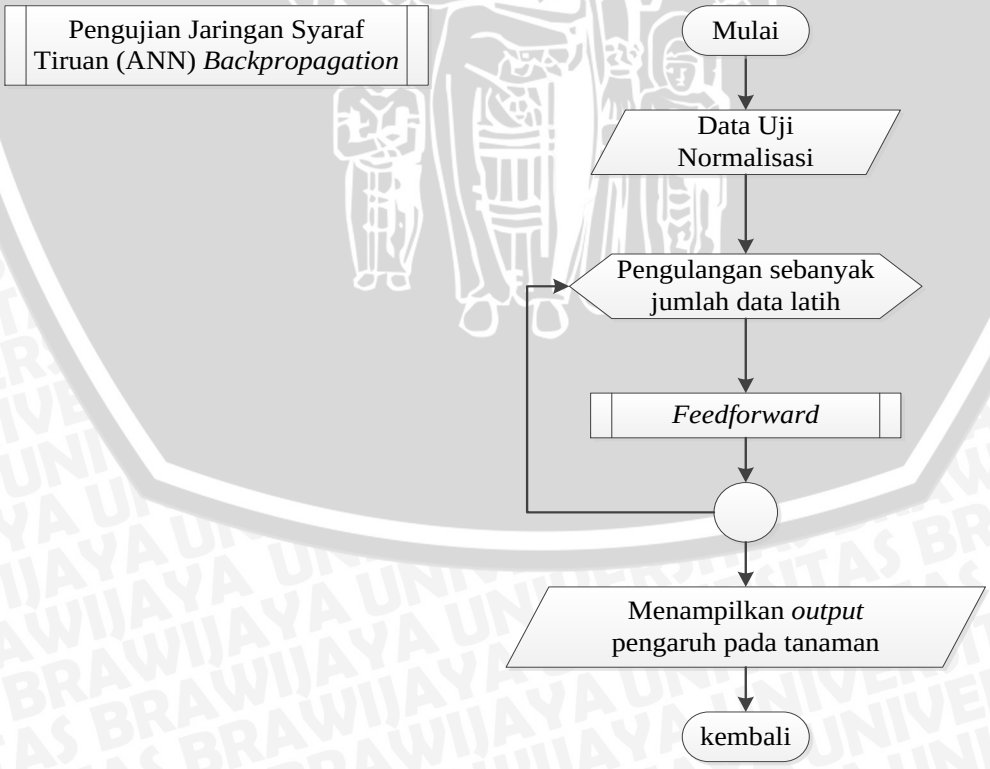
Gambar 3.8 Diagram Alir *Backpropagation*

Pada Gambar 3.8 Diagram Alir *Backpropagation* dijelaskan tahap-tahap proses *backpropagation* adalah sebagai berikut :

1. Lakukan pengulangan sebanyak jumlah indeks *output.*
2. Hitung *error* pada masing-masing indeks *output* dengan persamaan (2-13).
3. Setelah pengulangan selesai, kemudian hitung perubahan bobot *hiddenlayer* ke *output*δw_{jk} dengan persamaan (2-14).
4. Kemudian lakukan pengulangan sebanyak jumlah indeks *hiddenlayer*.
5. Jumlahkan setiap*error*pada indeks*output*terhadap*hiddenlayer*δ_in_j dengan persamaan (2-15).
6. Kemudian hitung *error* pada *hiddenlayer* dengan persamaan (2-16).
7. Setelah pengulangan, hitung perubahan bobot *inputlayer* ke *hiddenlayer*δv_{ij}dengan persamaan (2-17).
8. Proses selesai.

3.2.1.5 Proses Pengujian ANN*Backpropagation*

Setelah proses pelatihan, dilakukan proses pengujian diagram alir proses pengujian ANN *backpropagation* dijelaskan pada Gambar berikut :



Gambar 3.9 Diagram Alir Proses Pengujian *Backpropagation*

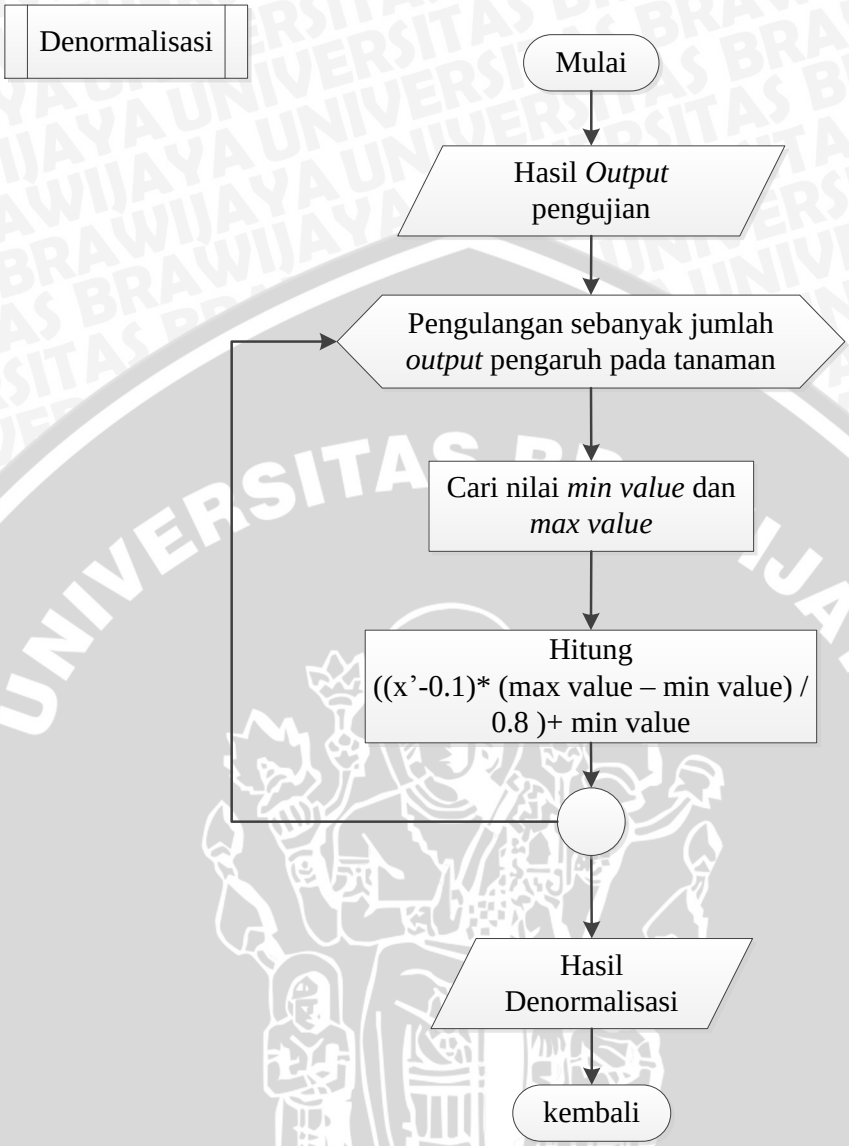
Proses pengujian hanya melakukan tahap perambatan maju saja. Seluruh bobot *input* diambil dari nilai bobot terakhir yang diperoleh dari proses pelatihan. Pada tahap pengujian ini, jaringan diharapkan dapat mengenali pola berdasarkan data baru yang diberikan (generalisasi). Pada Gambar 3.9 Diagram Alir Proses Pengujian *Backpropagation* dijelaskan tahap-tahap proses pengujian sebagai berikut :

1. *Input* data uji yang telah dinormalisasi.
2. Kemudian lakukan pengulangan sebanyak jumlah data uji.
3. Setiap data uji melakukan proses *feedforward* yang telah dijelaskan pada sub bab 3.2.1.3.
4. Kemudian setelah pengulangan selesai, ditampilkan hasil luaran pengaruh tanaman pada setiap data uji.Menampilkan pengaruh pada tanaman dari pemberian dosis.

3.2.1.6 Proses Denormalisasi

Proses Denormalisasi dilakukan setelah pengujian data uji jaringan syaraf tiruan *backpropagation* telah selesai dilakukan. Denormalisasi dilakukan untuk mengembalikan nilai pengaruh pada tanaman ke dalam *range real* (asli). Nilai yang diubah kedalam *rangereal* akan mempermudah memberikan takaran dosis pada implementasinya dan melihat perubahan secara nyata pada tanaman jagung. Proses denormalisasi tidak bisa dilakukan tanpa proses normalisasi sebelumnya. Langkah-langkah proses denomalisasi juga sama dengan proses normalisasi hanya yang membedakan posisi pembagi dan pembilang dari persamaan masing-masing. Pada Gambar 3.10 proses denormalisasi dilakukan dengan langkah-langkah sebagai berikut :

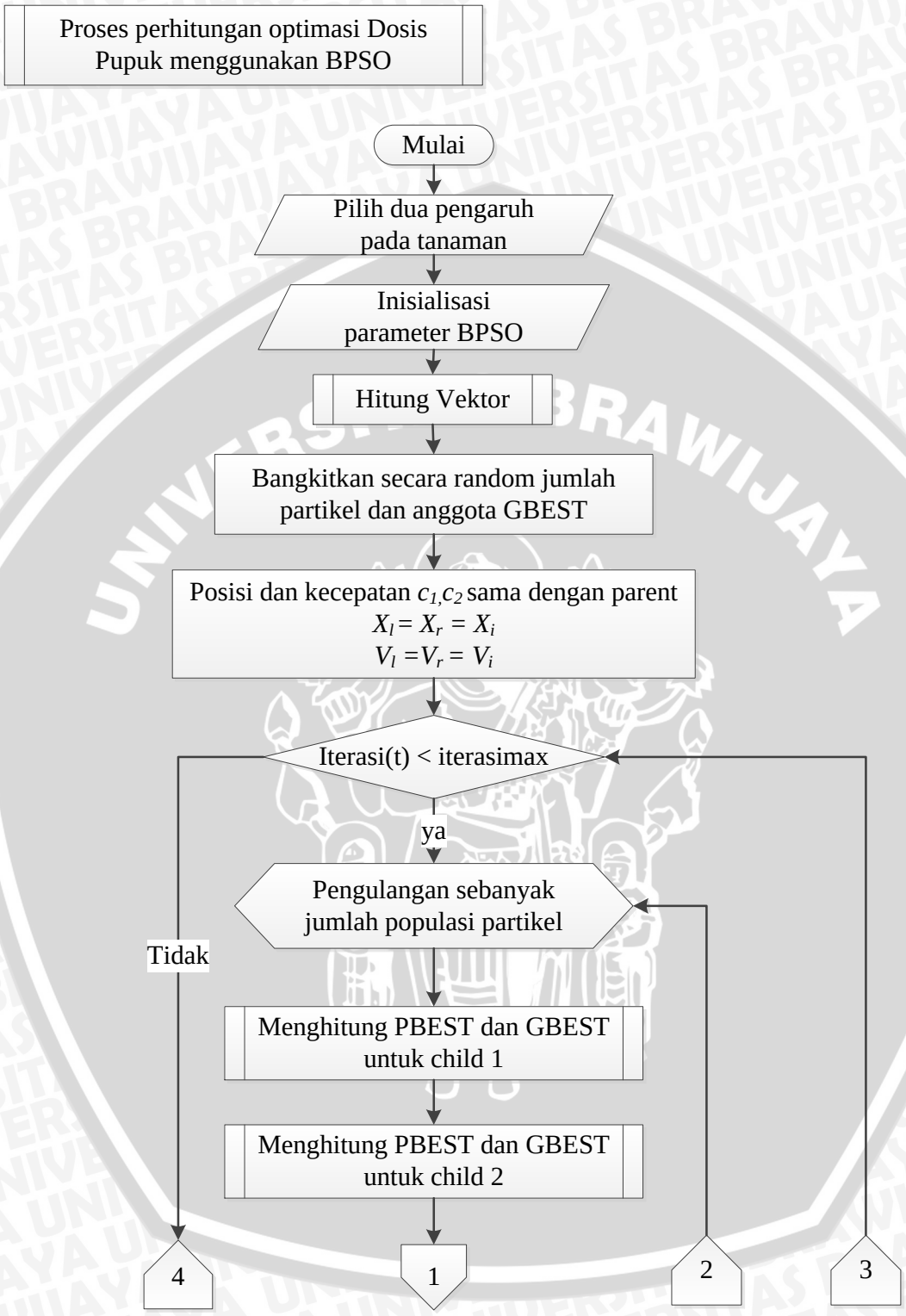
1. Masukan hasil dari nilai *output*pengujian
2. Pengulangan sebanyak jumlah *output*pengaruh pada tanaman.
3. Cari batas nilai min *value* dan mak *value* dari masing masing data.
4. Hitung menggunakan persamaan (2-21).
5. Lakukan proses pengulangan nomor 3-4.
6. Menampilkan data baru hasil luaran denormalisasi.

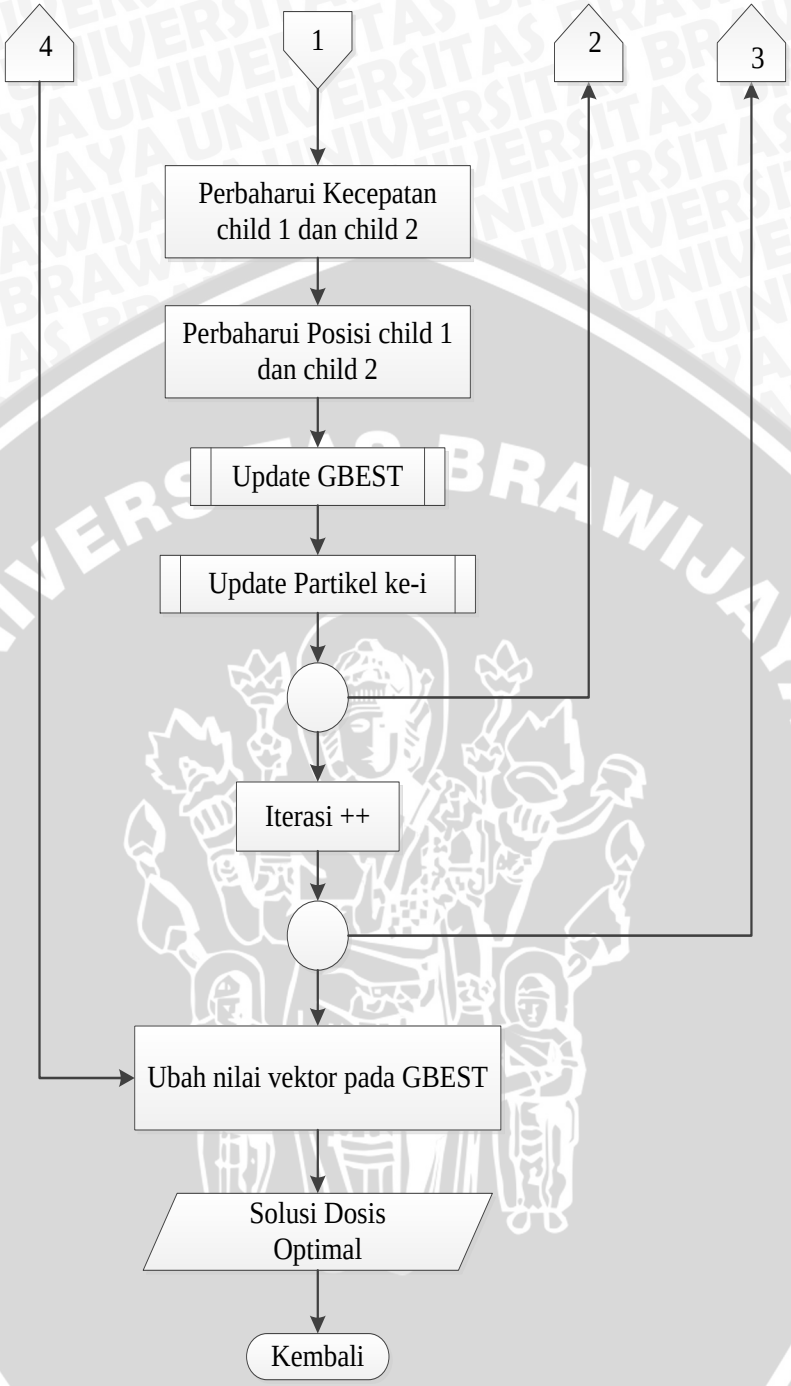


Gambar 3.10 Diagram Alir Proses Denormalisasi

3.2.2 Perhitungan Optimasi Pengaruh pada Tanaman dengan BPSO

Luaran yang dihasilkan ANN berupa nilai pengaruh pada tanaman, algoritma BPSO akan mengoptimalkan nilai pengaruh yang ingin dioptimasi oleh pengguna. Sehingga didapatkan luaran berupa kombinasi dosis pupuk yang terbaik sesuai dengan nilai pengaruh pada tanaman yang diharapkan. Pada Gambar 3.11 akan dijelaskan diagram alir proses optimasi pada BPSO:


Proses perhitungan optimasi Dosis Pupuk menggunakan BPSO
Mulai
Pilih dua pengaruh pada tanaman
Inisialisasi parameter BPSO
Hitung Vektor
Bangkitkan secara random jumlah partikel dan anggota GBEST
Posisi dan kecepatan c_1, c_2 sama dengan parent
$X_l = X_r = X_i$
$V_l = V_r = V_i$
Iterasi(t) < iterasimax
ya
Tidak
Pengulangan sebanyak jumlah populasi partikel
Menghitung PBEST dan GBEST untuk child 1
Menghitung PBEST dan GBEST untuk child 2
1
2
3
4



Gambar 3.11 Diagram Alir Perhitungan Optimasi BPSO

Pada Gambar 3.11 Diagram Alir Perhitungan Optimasi BPSO tahapan-tahapan optimasi yang dilakukan oleh BPSO adalah sebagai berikut :

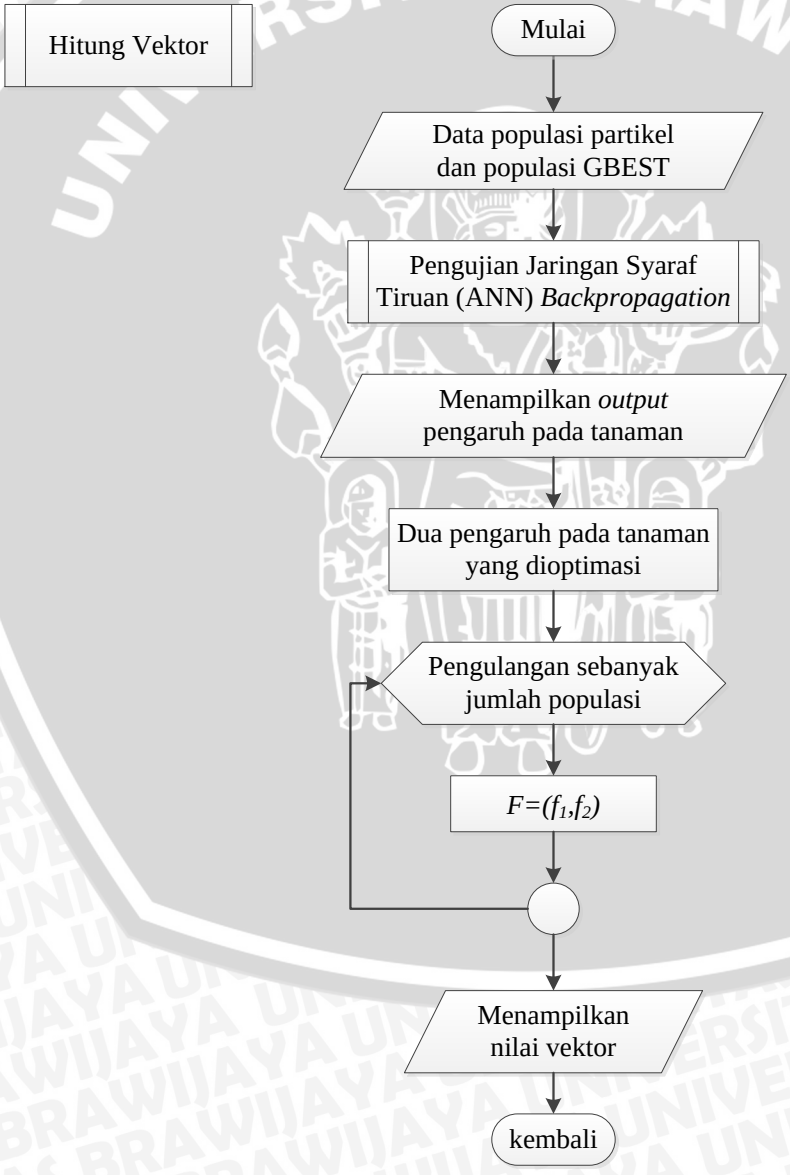
1. *Input* dua paremeter luaran pengaruh pada tanaman yang ingin dioptimasi
2. *Input* nilai untuk parameter BPSO yaitu bobot (w), *learning vector* (c_1, c_2), *Constriction factor*(χ), *population size* (jumlah populasi) dan jumlah iterasi.Sesuai dengan persamaan nilai parameter BPSO pada sub bab 2.7
3. Kemudian bangkitkan secara random untuk 30 populasi partikel dan 8 untuk populasi GBEST, posisi partikel dan kecepatannya dibangkitkan secara random sesuai proses inisialisasi pada persamaan (2-24).
4. Hitung vektor dari setiap populasi partikel yang dibangkitkan dan populasi GBEST.
5. Inisialisasi posisi dan kecepatan dari $child_1$ dan $child_2$ sama dengan *parent.*
6. Cek kodisi keadaan jika iterasi < iterasi max
7. Untuk iterasi dimulai t=0 dilakukan pengulangan langkah-langkah sebanyak jumlah populasi partikel berikut :
   a. Mencari GBEST dan PBEST pada $child_1$dengan strategi pencarian terisolasi (*isolated searching strategy).*
   b. Mencari GBEST dan PBEST pada $child_2$dengan strategi pencarian tetangga (*neighborhood searching strategy).*
   c. Kemudian dilakukan perubahan kecepatan dari setiap $child_1$ dan $child_2$dengan persamaan (2-34).
   d. Menghitung perubahan posisi setiap $child_1$ dan $child_2$dengan persamaan (2-35).
   e. Kemudian lakukan pembaharuan (*update)* pada kumpulan populasi GBEST.
   f. Menentukan posisi partikel ke-i dengan membandingkan hasil vektordari $child_1$ dan $child_2$.
   g. Perulangan dilakukan sampai iterasi maksimal.
8. Jika iterasi telah maksimal, maka pada kumpulan populasi GBEST akan dipilih satu anggota dengan nilai vektor terkecil dari populasi yang ada. Nilai

fitness vektor ini akan diubah ke persamaan (2-25) dan (2-26) untuk melihat hasil nilai akhir terbesar.

9. Ditampilkan member yang terpilih dengan kombinasi dosisyang optimal pada masing-masing jenis pupuk.

3.2.2.1 Pehitungan vektor

Proses menentukan nilai vektor pada populasi partikel dan populasi GBEST melibatkan proses pengujian atau *training backpropagation* ANN. Berikut adalah diagram alir dari proses perhitungan vektor BPSO :



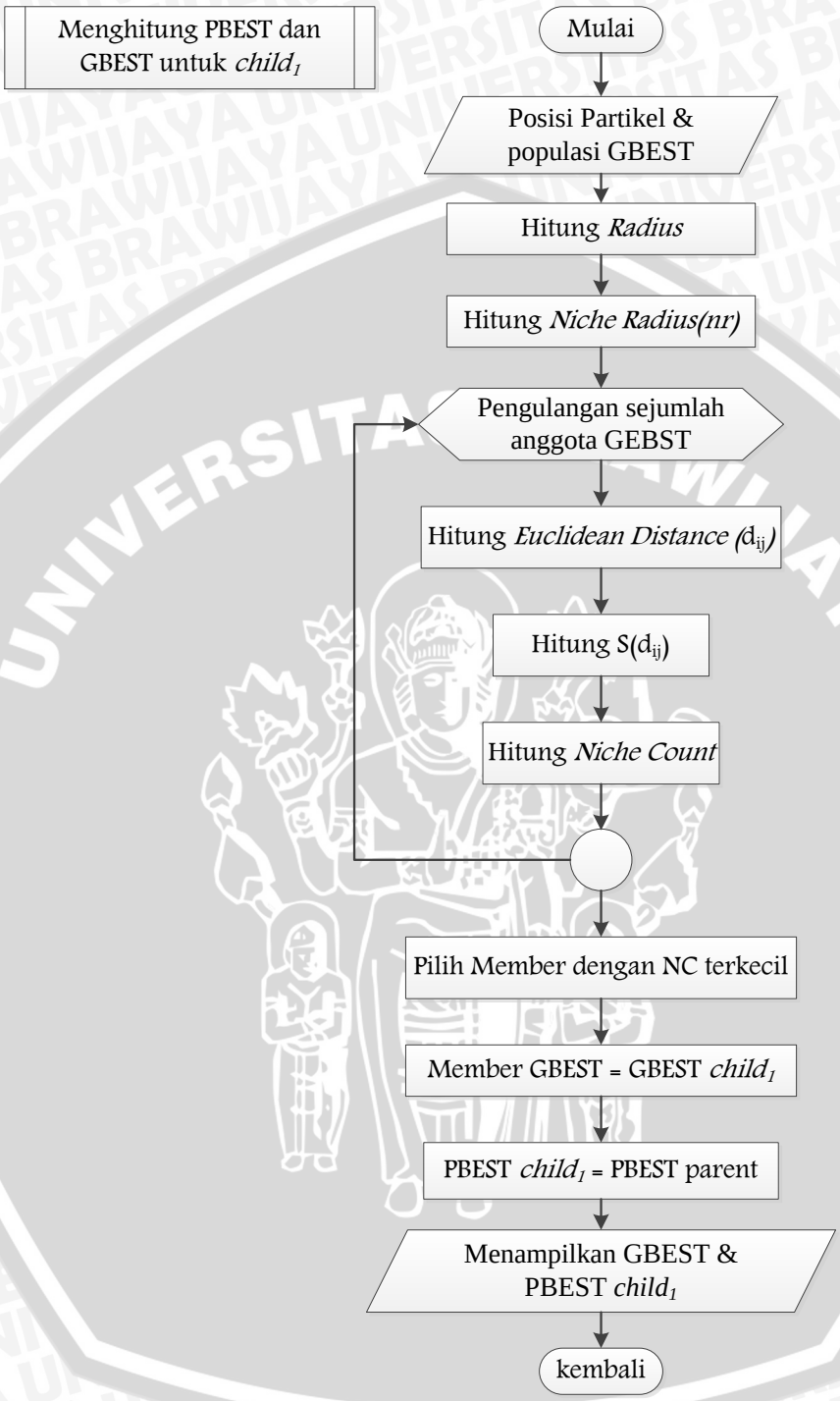
Gambar 3.12 Diagram Alir Perhitungan Vektor

Gambar 3.12 Diagram Alir Perhitungan Vektor, proses menentukan nilai vektor pada populasi partikel dan populasi GBEST untuk menjadi nilai pembanding dalam memperbaharui anggota dengan posisi yang terbaik untuk GBEST dan posisi terbaru pada populasi partikel di iterasi selanjutnya. Berikut adalah tahap-tahap perhitungan vektor :

1. *input*data populasi partikel dan populasi GBEST
2. Data ini diproses pada pengujian *backpropagtion* yang dijelaskan pada sub bab 3.2.1.5 dengan bobot terbaik dan nilai *error* terkecil.
3. Kemudian didapatkan sebanyak lima luaran pengaruh pada tanaman berdasarkan kombinasi dosis pada masing-masing data populasi partikel dan GBEST.
4. Dua pengaruh pada tanaman yang ingin dioptimasi, akan dihitung nilai vektornya.
5. Lakukan pengulangan sebanyak jumlah data populasi partikel dan populasi anggota GBEST.
6. Lakukan perhitungan vektor pada setiap data dengan persamaan (2-25) dan persamaan (2-26).
7. Setelah proses pengulangang sebanyak jumlah data telah selesai kemudian menampilkan hasil perhitungan vektor.

3.2.2.2Menghitung GBEST dan PBEST untuk $child_1$

Metode BPSO , dikatakan *bidirectional* karena proses pembaharuan untuk PBEST dan GBEST dari $child_1$dan*child2* berbeda. Proses untuk menemukan PBEST dan GBEST dari $child_1$ dengan cara strategi pencarian terisolai atau *isolated searching strategy*. Strategi pencarian terisolasi adalah pencarian pada kumpulan GBEST dengan nilai minimal NC (*niche count). Niche count* didapatkan dari proses perhitungan *niche radius. Niche radius* didapatkan dari penjumlahan *Euclidean distance* di antara setiap anggota GBEST.Nilai NC yang terkecil dipilih sebagai GBEST untuk $child_1$. Berikut diagram alir untuk menemukan GBEST dan PBEST pada $child_1$akan dijelaskan pada Gambar 3.13Diagram Alir Menghitung PBEST dan GBEST untuk $child_1$:



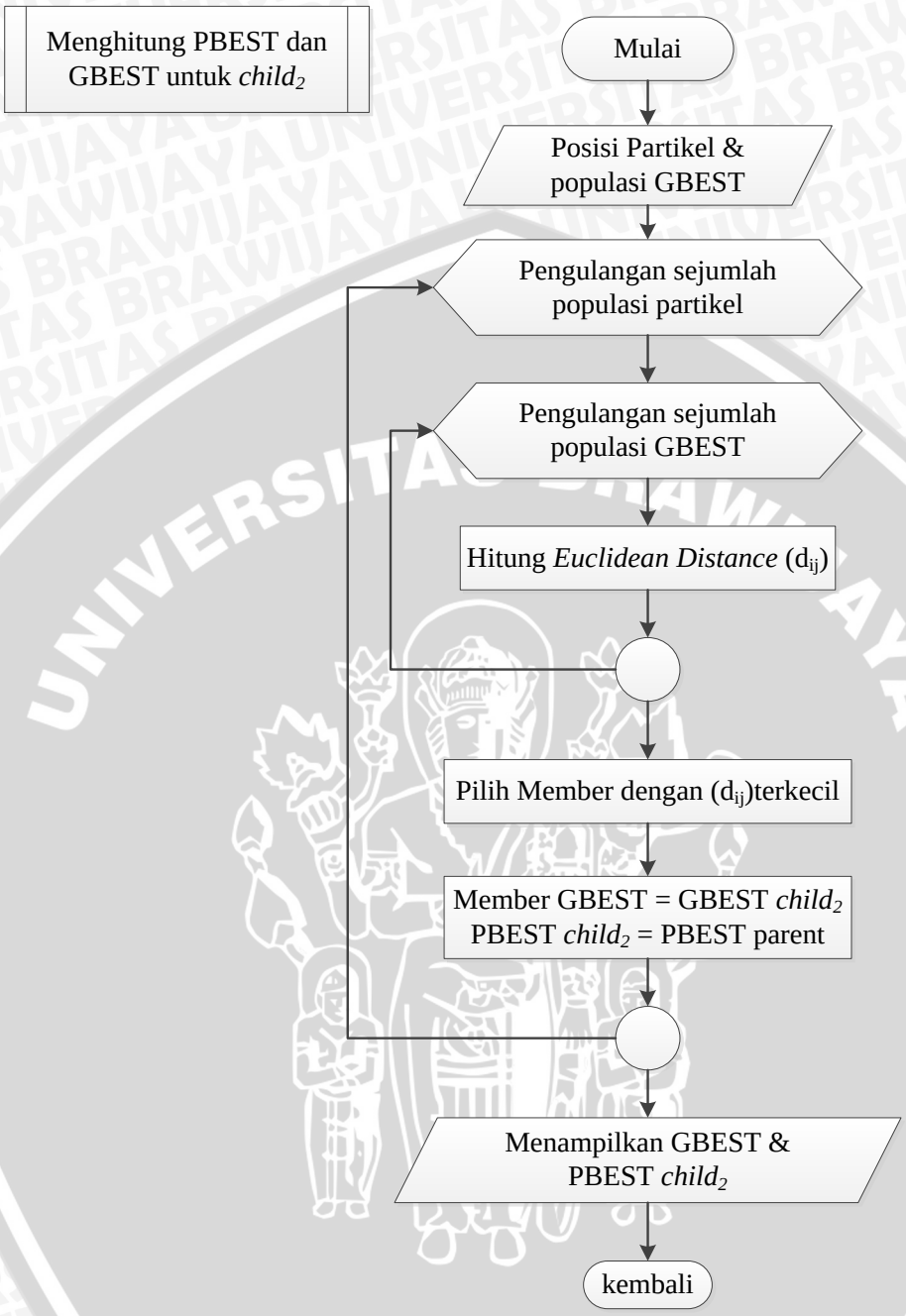
Gambar 3.13 Diagram Alir Menghitung PBEST dan GBEST untuk $child_1$

Pada Gambar 3.13 Diagram Alir Menghitung PBEST dan GBEST untuk $child_1$dijelaskantahapan-tahapan dalam menghitung PBEST dan GBEST untuk $child_1$ adalah sebagai berikut :

1. Data yang diperlukan untuk diproses pada penentuan PBEST dan GBEST untuk *child*$_1$ adalah kumpulan populasi dari GBEST.
2. Menghitung *radius* dari kumpulan populasi anggota GBEST menggunakan persamaan (2-32).
3. Menghitung *niche radius* dari populasi GBEST dengan persamaan (2-31). *Niche radius* disebut juga sigma share, nilai ini didapatkan dengan membagi nilai *radius* dengan jumlah anggota GBEST akar jumlah parameter luaran pengaruh pada tanaman yaitu sebanyak lima.
4. Kemudian dilakukan pengulangan sebanyak jumlah anggota populasi dari GBEST.
5. Hitung *Euclidean distance* antara setiap anggota GBEST dengan anggota GBEST lainnya (2-29).
6. *Euclidean distance* yang telah dihitung kemudian dimasukan kedalam fungsi *share*, jika nilai *Euclidean distance* lebih kecil dari nilai *niche radius* akan dihitung dengan persamaan (2-30) jika lebih besar maka fungsi *share* yang dihasilkan bernilai nol.
7. Kemudian hitung *niche count* pada setiap anggota GBEST dengan menjumlahkan fungsi *share*, dengan persamaan (2-33).
8. Setelah pengulangan sebanyak jumlah populasi GBEST telah selesai, kemudian nilai niche count yang paling kecil akan dipilih sebagai GBEST *child*$_1$.
9. Untuk PBEST *child*$_1$ sama dengan PBEST pada *parent* nya. Proses menghitung PBEST dan GBEST untuk *child*$_1$ dengan *isolated searching strategy* atau startegi pencarian terisolasi telah selesai.

3.2.2.3 Menghitung GBEST dan PBEST untuk *child*$_2$

Proses untuk menemukan PBEST dan GBEST dari *child*$_2$ dengan cara strategi pencarian tetangga atau *neighborhood searching strategy*. Strategi pencarian tetangga adalah pencarian pada populasi GBEST dengan *Euclidean Distance*terkecil antara anggota populasi partikel terhadap semua populasi pada GBEST. Berikut diagram alir untuk menghitung GBEST dan PBEST pada *child*$_2$.



Gambar 3.14 Diagram Alir Menghitung PBEST dan GBEST untuk *child*$_2$

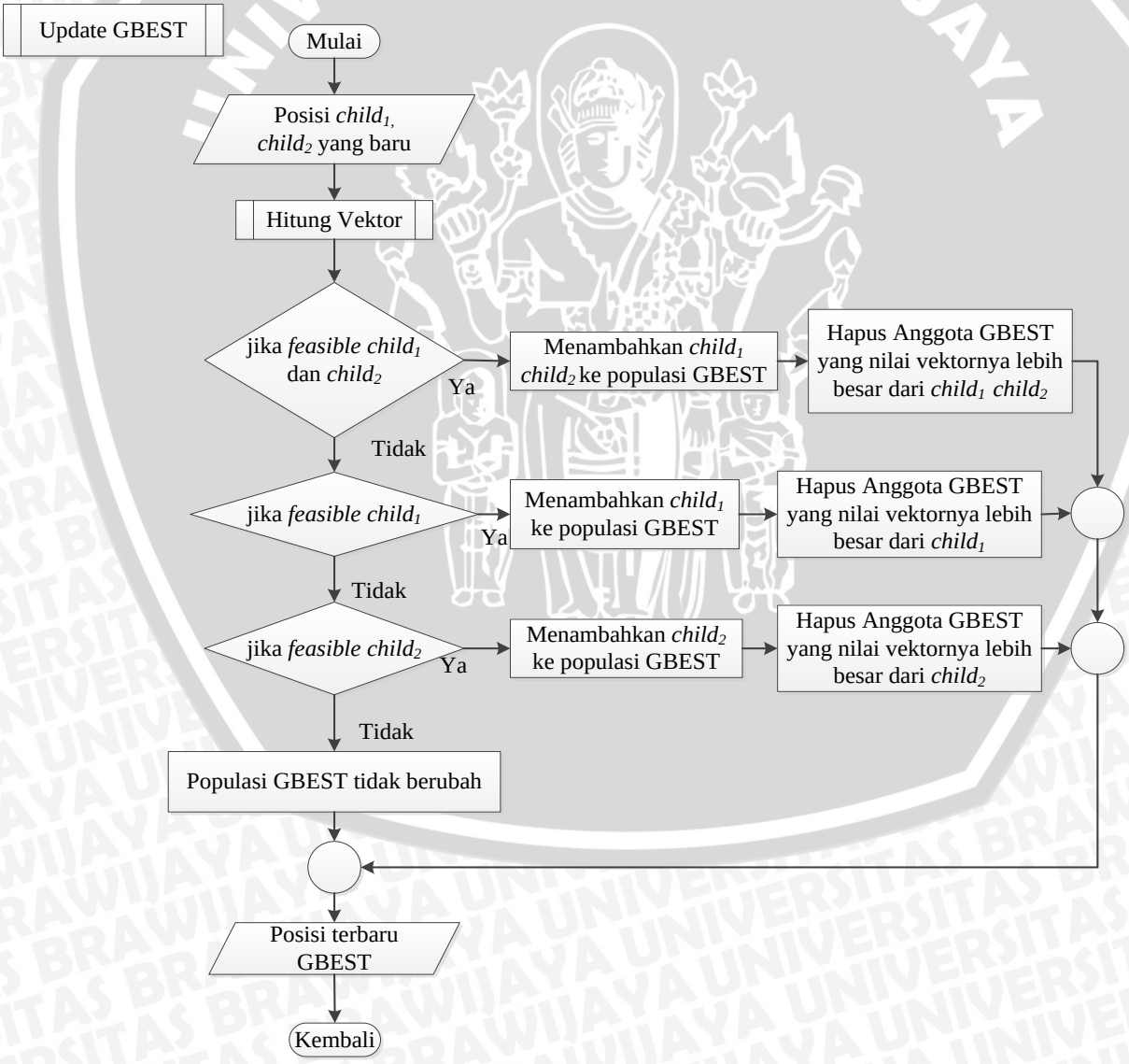
Berikut adalah tahap-tahap untuk menentukan GBEST dan PBEST pada *child*$_2$:

1. Lakukan proses pengulangan sebanyak jumlah anggota populasi partikel
2. Untuk populasi ke-i lakukan proses pengulangan sebanyak jumlah populasi GBEST
3. Hitung jarak *Euclidean distance* antara partikel ke setiap anggota populasi GBEST dengan persamaan (2-29).

4. Setelah menghitung Euclidean distance, pilih anggota GBEST dengan jarak nilai terkecil. Anggota GBEST dengan jarak terpendek terhadapa partikel ke-i akan dipilih sebagai GBEST *child*$_2$.
5. Untuk PBEST *child*$_2$ sama dengan PBEST pada *parent* nya. Proses menghitung PBEST dan GBEST untuk *child*$_2$ dengan *neighborhood searching strategy* atau startegi pencarian terisolasi telah selesai.

3.2.2.4*Update* populasi GBEST

Untuk memperbaharui setiap populasi pada GBEST akan dijelaskan pada Gambar 3.15 diagram alir dibawah ini:



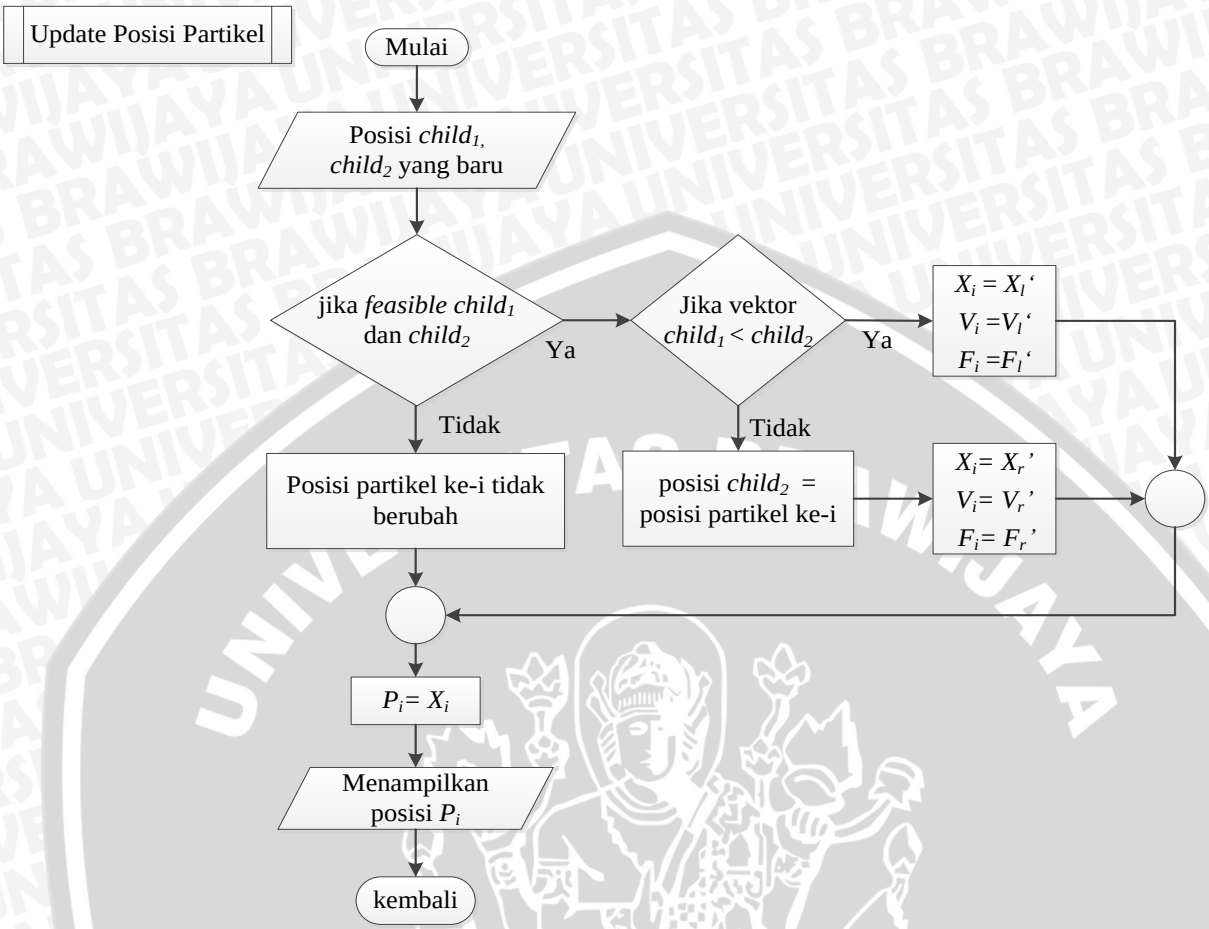
Gambar 3.15 Diagram Alir *Update* GBEST

Memperbaharui nilai GBEST diperlukan agar kumpulan solusi nilai terbaik yang diberikan sudah optimal. Oleh sebab itu setiap partikel pada iterasi pertama sampai iterasi maksimal akan ada proses *update* untuk populasi dari GBEST. Berikut tahap-tahap update GBEST pada Gambar 3.15 Diagram Alir *Update* GBEST :

1. Data yang dibutuhkan adalah posisi terbaru dari $child_1$ dan $child_2$.
2. Kemudian menghitung vektor dari $child_1$ dan $child_2$
3. Untuk melakukan *update* GBEST ada beberapa kondisi keadaan yang mungkin bisa terjadi yaitu :
   a. Jika nilai yang *feasible*(layak) adalah $child_1$ dan $child_2$maka $child_1$ dan $child_2$ditambahkan ke populasi GBEST dan menghapus anggota GBEST yang vektornya lebih besar dari $child_1$ dan $child_2$
   b. Jika nilai yang *feasible* adalah $child_1$ maka $child_1$ ditambahkan ke populasi GBEST dan menghapus anggota GBEST yang nilai vektornya lebih besar dari $child_1$.
   c. Jika nilai yang *feasible* adalah $child_2$maka $child_2$ditambahkan ke populasi GBEST dan menghapus anggota GBEST yang nilai vektornya lebih besar dari $child_2$.
   d. Jika tidak ada yang *feasible* antara $child_1$ dan $child_2$maka tidak ada penambahan anggota pada populasi GBEST atau tidak ada perubahan pada GBEST.
4. Proses*update* populasi GBEST telah selesai dan kembali ke langkah selanjutnya yaitu *update* populasi partikel.

3.2.2.5 *Update* Posisi Partikel ke-i

Proses *Update* Posisi Partikel dilakukan untuk menentukan posisi terbaru dari partikel, untuk mendapatkan posisi terbaru dengan nilai kecepatan dan vektor yang terbaik. Pada proses update partikel ini juga, membandingkan hasil posisi baru yang dihasilkan oleh $child_1$ dan $child_2$.Proses membandingkan $child_1$ dan $child_2$dicari posisi yang *feasible* untuk menggantikan posisi partikel ke-i untuk iterasi selanjutnya. Berikut adalah diagram alir dari proses *update* partikel pada Gambar 3.16 Diagram Alir *Update*Posisi Partikel:



Gambar 3. 16 Diagram Alir *Update* Posisi Partikel

Berikut tahap-tahap update posisi partikel pada Gambar 3.16 Diagram Alir *Update*Posisi Partikel :

1. Cek kondisi keadaan posisi terbaru dari $child_1$ dan $child_2$mana yang feasible atau nilainya layak.
2. Jika kedua posis terbaru dari $child_1$ dan $child_2$ layak maka, cek kondisi nilai vektor yang dihasilkan oleh $child_1$ dan $child_2$.
3. Jika vektor $child_1 < child_2$ maka posisi$child_1$ akan menjadi posisi baru partikel ke-i pada iterasi selanjutnya. Kemudian update kecepatan dan vektor posisi partikel baru sama dengan kecepatan dan vektor$child_1$
4. Jika tidak maka posisi $child_2$ akan menjadi posisi baru partikel ke-i pada iterasi selanjutnya.Kemudian update kecepatan dan vektor posisi partikel baru sama dengan kecepatan dan vektor $child_2$

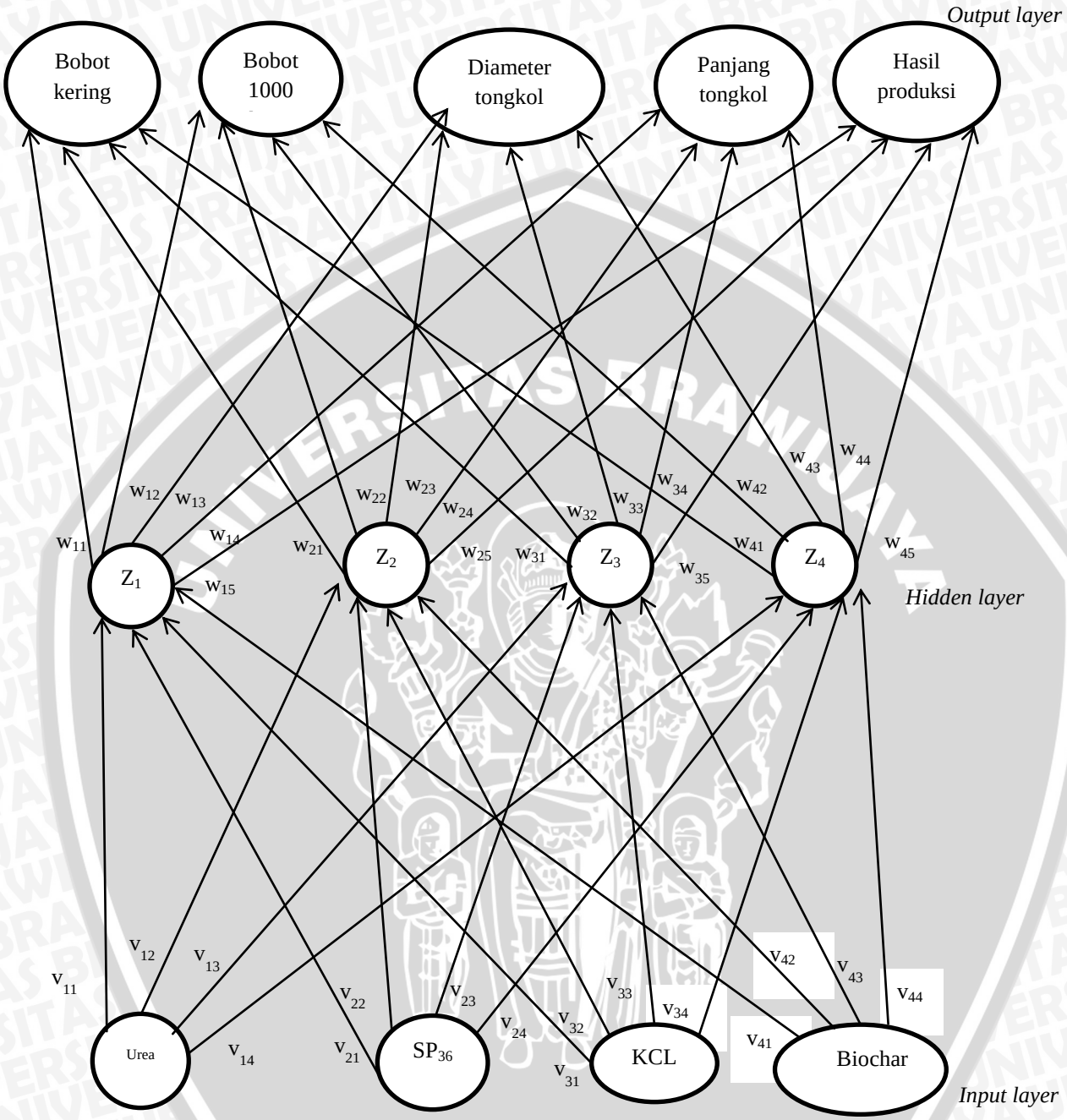
5. Jika tidak ada nilai yang layak dari $child_1$ dan $child_2$ maka posisi partikel ke-i tidak berubah. Kecepatan dan vektor juga tidak berubah sama dengan posisi partikel sebelumnya.
6. Kemudian langkah terakhir update posisi partikel sama denagn posisi PBEST.

3.3 Perhitungan Manual ANN *Backpropagation*

Pada bagian ini dijelaskan perhitungan manual proses ANN *backpropagation* yang meliputi proses perhitungan manual normalisasi data latih dan data uji, proses inisialisasi parameter ANN, membangkitkan bobot secara random, menghitung *error*, pengulangan iterasi dan pembaruan bobot untuk proses pelatihan. Setelah itu bobot yang terbaik pada proses pelatihan akan digunakan untuk proses pengujian hasil pengaruh pada tanaman sesuai dengan data uji yang digunakan berikut adalah perhitungan manual untuk proses ANN *backpropagation.*

3.3.1 Analisis Permasalahan

Dalam ANN *Backpropagation* kasus yang diselesaikan adalah menentukan nilai bobot terbaik agar menghasilkan target terbaik atau mendekati pengaruh pada tanaman dengan nilai *error*yang kecil. Pada kasus ini ada 4 kombinasi jenis pupuk yang diberikan yaitu pupuk UREA, SP_{36}, KCL dan Biochar Sampah Kota. Setiap jenis pupuk diberikan 6 perilaku kombinasi dosis pupuk yang berbeda berdasarkan pakar. 6 perilaku kombinasi dosis ini memberikan pengaruh yang berbeda beda pada setiap pengaruh tanaman yang diamati. Ada 5 pengaruh pada tanaman yang diamati yaitu, bobot kering tanaman, bobot 1000 butir, panjang tongkol, diameter tongkol dan hasil produksi tanaman. Pada proses perhitungan manualisasi ini untuk proses pelatihan *backpropagation* diberikan jumlah data latih sebanyak 12, kemudian dilakukan perhitungan smapai ietrasi ke dua, bobot terakhir dari iterasi ke-2 akan digunakn dalam pengujian data uji sebanyak 6 data uji. Kemudian bobot ini juga digunakan untuk penentuan nilai vektor pada proses BPSO. Pad BPSO jumlah data yang menjadi populasi partikel adalah sebanyak 5 data, sedangkan populasi awal GBEST sebanyak 3 data. kasus ini :



Gambar 3.17 Arsitektur *Backpropagation* Sistem yang Dibangun

Pada Gambar 3.14 adalah arsitekturr backpropagation yang akan diselesaikan untuk perhitungan manualisasi. Pada Jaringan Arsitektur ini terdapat 4 neuron*input*, 5 neuron *output* dan jumlah *hiddenlayer* sebanyak 4 buah. Untuk menghitung jumlah *hiddenlayer* digunakan persamaan (2-22). Pada proses manualisasi ini data latih adalah 12 buah data, dan data uji sebanyak 6.

3.3.2 Proses Normalisasi Data latih dan Data uji

Pada perhitungan ini akan dilakukan normalisasi untuk data latih dan data uji. Data latih yang digunakan adalah 12 buah dan data uji yang digunakan adalah 6 buah. Berikut pada Tabel 3.2 dan Tabel 3.3 adalah data asli dari data uji dan data latih sebelum dilakukan normalisasi.

Tabel 3.2 Data Latih sebelum Normalisasi

No	Nama Perlakuan	Jenis Pupuk (Ton/ha)				Pengaruh Pertumbuhan dan Hasil pada Jagung				
		Urea (U)	SP_{36} (S)	KCL (K)	Biochar Sampah Organik Kota (B)	Bobot kering tanaman (ton/ha) (BK)	Bobot 1000 butir (gram) (BB)	Diameter Tongkol (cm) (DT)	Panjang Tongkol (cm) (PT)	Hasil Produksi (ton/ha) (HP)
1	K1	0.2	0.277	0.2	0	4.61	251.7	4.99	19.79	5.75
2	K2	0.2	0.277	0.2	0	4.89	225.5	3.99	13.97	5.2
3	BK0	0.2	0.277	0	30	5.58	251.8	5	20.40	6.69
4	BK0	0.2	0.277	0	30	5.77	245	4.87	20.32	6.17
5	BK1/4	0.2	0.277	0.05	30	6	257.1	4.76	19.50	8.44
6	BK1/4	0.2	0.277	0.05	30	6.17	242.2	4.83	20.47	6.28
7	BK1/2	0.2	0.277	0.1	30	5.41	256.4	4.96	20.22	5.84
8	BK1/2	0.2	0.277	0.1	30	5.39	246.8	4.86	20.40	5.59
9	BK3/4	0.2	0.277	0.15	30	4.92	255.3	4.9	20.27	7.24
10	BK3/4	0.2	0.277	0.15	30	4.87	250	4.7	20.19	5.7
11	BK1	0.2	0.277	0.2	30	4.55	250.5	4.95	19.37	6.25
12	BK1	0.2	0.277	0.2	30	4.47	250	4.77	20.30	6.7

Tabel 3.3 Data Uji sebelum Normalisasi

No	Nama Perlakuan	Jenis Pupuk (Ton/ha)			
		Urea	SP_{36}	KCL	Biochar Sampah Organik Kota
1	K3	0.2	0.277	0.2	0
2	BK0	0.2	0.277	0	30
3	BK1/4	0.2	0.277	0.05	30
4	BK1/2	0.2	0.277	0.1	30
5	BK3/4	0.2	0.277	0.15	30
6	BK1	0.2	0.277	0.2	30

Proses Normalisasi pada Tabel 3.2 Data Latih sebelum Normalisasi dan Tabel 3.3 Data Uji sebelum Normalisasi dilakukan pada range nilai diatas satu sehingga untuk pupuk Urea, SP36 dan KCL tidak perlu dilakukan normalisasi. Normalisasi dilakukan pada Biochar sampah kota, dan *output* pengaruh pertumbuhan dan hasil jagung. Dilakukan penyederhanaan nama untuk *input*dan *output* sesuai dengan persamaan *backpropagation.* Berikut adalah hasil normalisasi untuk data latih dan data uji pada Tabel 3.4 dan Tabel 3.5 :

Tabel 3.4 Data Latih Normalisasi

No	Nama Perlakuan	Jenis Pupuk (Ton/ha)				Pengaruh pertumbuhan dan hasil pada Jagung				
		U	S	K	B	BK	BB	DT	PT	HP
1	K3	0.2	0.277	0.2	0.1	0.165882	0.763291	0.892079	0.816308	0.235802
2	K3	0.2	0.277	0.2	0.1	0.297647	0.1	0.1	0.1	0.1
3	BK0	0.2	0.277	0	0.9	0.622353	0.765823	0.9	0.891385	0.467901
4	BK0	0.2	0.277	0	0.9	0.711765	0.593671	0.79703	0.882031	0.339506
5	BK1/4	0.2	0.277	0.05	0.9	0.82	0.9	0.709901	0.780615	0.9
6	BK1/4	0.2	0.277	0.05	0.9	0.9	0.522785	0.765347	0.9	0.366667
7	BK1/2	0.2	0.277	0.1	0.9	0.542353	0.882278	0.868317	0.868738	0.258025
8	BK1/2	0.2	0.277	0.1	0.9	0.532941	0.639241	0.789109	0.891877	0.196296
9	BK3/4	0.2	0.277	0.15	0.9	0.311765	0.85443	0.820792	0.875385	0.603704
10	BK3/4	0.2	0.277	0.15	0.9	0.288235	0.720253	0.662376	0.865538	0.223457
11	BK1	0.2	0.277	0.2	0.9	0.137647	0.732911	0.860396	0.764615	0.359259
12	BK1	0.2	0.277	0.2	0.9	0.1	0.720253	0.717822	0.879077	0.47037

Pada Tabel 3.4 Data Latih Normalisasi penyerderhanaan nama U untuk Urea, S untuk SP_{36}, K untuk KCL dan B untuk Biochar. Sedangkan, penyerderhanaan nama untuk BK bobot kering, BB bobot 1000 butir, DT diameter tongkol, PT panjang tongkol dan HP hasil prosuksi. Min*value* untuk dosis B adalah 0 ton/ha sedangkan nilai maks*value* untuk dosis B adalah 30 ton/ha, sehingga contoh perhitungan normalisasi untuk B didapatkan 0.1 untuk perlakuan pertama yaitu K3 dengan data asli pemberian dosis adalah 0 ton/ha adalah :

Normalisasi dosis B $= (0.8 * (0 - 0/30 - 0)) + 0.1 = 0.1$

Tabel 3.5 Data Uji Normalisasi

No	Nama Perlakuan	Jenis Pupuk (Ton/ha)			
		U	S	K	B
1	K3	0.2	0.277	0.2	0.1
2	BK0	0.2	0.277	0	0.9
3	BK1/4	0.2	0.277	0.05	0.9
4	BK1/2	0.2	0.277	0.1	0.9
5	BK3/4	0.2	0.277	0.15	0.9
6	BK1	0.2	0.277	0.2	0.9

Pada Tabel 3.3 nilai min *value* untuk pengaruh pada tanaman jagung BK adalah 4.37 ton/ha sedangkan nilai maks *value* untuk dosis B adalah 5.93 ton/ha, data asli untuk contoh perhitungan normalisasi untuk B sehingga didapatkan 0.1 pada Tabel 3.5 untuk perlakuan pertama yaitu K3 dengan data asli Bobot kering tanaman adalah 4.37 ton/ha adalah :

Normalisasi Pengaruh pada BK $= (0.8 * (4.37 - 4.37/5.93 - 5.93)) + 0.1 = 0.1$

3.3.3 Perhitungan Pelatihan Data Latih ANN *Backpropagation*

Proses pelatihan ini dilakukan untuk melatih ANN dengan cara melakukan perubahan bobot, pada kasus permasalahan ini proses pelatihan dilakukan pada 12 data latih dengan iterasi sebanyak dua kali. Berikut adalah proses manualisasi untuk iterasi pertama data latih :

3.3.3.1 Iterasi Pertama

Iterasi dimulai dari t=1, pada iterasi ini adalah langkah awal dari membangkitkan kecepatan secara random(V_{ij}) antara bobot dari unit *input* (i) menuju untu *hidden* (j).

1. Perhitungan *Feedforward*

1. Proses *Inputlayer* ke *hiddenlayer*

Proses *inputlayer* ke *hiddenlayer* Vij bobot dibangkitkan secara acak untuk data pertama dari (-1,1). Jumlah *hiddenlayer* pada manualiasai ada 4 buah. Berikut adalahdata pertama dari K3 no 1 pada tabel 3.6 yang telah dinormalisasi dan bobot nya ditunjukkan pada tabel 3.7 :

Tabel 3.6 Data K3 Normalisasi

No	Nama Perlakuan	Jenis Pupuk (Ton/ha)			
		U	S	K	B
1	K3	0.2	0.277	0.2	0.1

Tabel 3.7 Bobot *Input* ke *Hiddenlayer*

$V_{i,j}$	Z_1	Z_2	Z_3	Z_4
X_1	0.129857	0.173633	0.171860	-0.395562
X_2	-0.911927	0.455795	0.281215	-0.385098
X_3	-0.895038	-0.182430	-0.856605	0.788745
X_4	0.105887	-0.661232	-0.597939	-0.993903

2. Menjumlahkan bobot sinyal *input*

Kemudian menjumlahkan bobot sinyal*input*:

Z_in_1= (0.2*0.129857)+(0.277*-0.911927)+(0.2*-0.895038)+(0.1*0.105887)

=-0.3951

Z_in_2 = (0.2*0.173633)+(0.277*0.455795)+(0.2*-0.182430)+(0.1*-0.661232)

=0.05837

Z_in_3 = (0.2*0.171860)+(0.277*0.281215)+(0.2*-0.856605)+(0.1*-0.597939)

=-0.1188

Z_in_4 = (0.2*-0.395562)+(0.277*-0.385098)+(0.2*0.788745)+(0.1*-0.993903)

=-0.1274

Berikut adalah hasil menjumlahkan bobot sinyal yang didapatkan dari 12 data latih yang telah dinormalisasi :

Tabel 3.8 Jumlah Z_in_j bobot sinyal *input*

Z_in_1	Z_in_2	Z_in_3	Z_in_4
-0.3951	0.05837	-0.1188	-0.1274
-0.3898	0.06704	-0.1126	-0.1206
-0.1262	-0.4252	-0.4205	-1.0744
-0.0904	-0.3681	-0.3774	-1.036
-0.1029	-0.3226	-0.38	-0.9613
-0.06	-0.26	-0.3277	-0.9088
-0.0691	-0.2089	-0.3247	-0.8278
-0.0266	-0.1464	-0.2789	-0.7818
-0.0346	-0.0988	-0.2793	-0.7022
0.0082	-0.04	-0.2283	-0.6475
-0.0002	0.00317	-0.2307	-0.5667
0.03671	0.05281	-0.1883	-0.5228

Bobot untuk data ke-2 sampai ke-12 didapatkan pada proses perubahan bobot pada tahap terakhir, hanya data ke-1 pada iterasi pertama yang bobotnya dibangkitkan secara acak.

3. Menerapkan Fungsi Aktivasi menghitung sinyal *input*

Fungsi aktivasi yang digunakan adalah fungsi aktivasi *sigmoid biner*, berikut merupaka contoh proses perhitungan sinyal *input*:

e = Bilangan *eural* yaitu 2.71828

$$Z_1 = \frac{1}{(1+2.71)^{-0.39505}} = 0.59721$$

$$Z_2 = \frac{1}{(1+2.71)^{-0.05837}} = 0.48546$$

$$Z_3 = \frac{1}{(1+2.71)^{-0.11885}} = 0.52959$$

$$Z_4 = \frac{1}{(1+2.71)^{-0.12743}} = 0.53172$$

Berikut hasil sinyal *input* dengan menerapkan fungsi aktivasi sigmoid biner :

Tabel 3.9 Hasil Perhitungan sinyal *Input*dengan Fungsi *Sigmoid Biner*

Z_1	Z_2	Z_3	Z_4
0.59721	0.48546	0.52959	0.53172
0.59595	0.4833	0.52804	0.53017
0.5314	0.60442	0.60329	0.7449
0.52251	0.59072	0.59296	0.73837
0.52562	0.57971	0.59359	0.72433
0.51497	0.56445	0.58096	0.71564
0.51724	0.55192	0.58026	0.69961
0.50666	0.53649	0.56911	0.69128
0.50869	0.5247	0.56923	0.67481
0.49805	0.51011	0.55676	0.66478
0.50018	0.49942	0.55739	0.64774
0.49104	0.48712	0.54699	0.63881

4. Bobot dari *hiddenlayer* ke *inputlayer*

Untuk data ke-1 bobot yang dibangkitkan secara acak dengan range (0.1), sedangkan data ke-2 sampai ke-12 bobot ditentukan berdasarkan proses perubahan bobot. Berikut bobot dari *hiddenlayer* ke *outputlayer* untuk data ke-1 :

Tabel 3.10 Bobot *Hiddenlayer* ke *Outputlayer*

$W_{j,k}$	Y_1	Y_2	Y_3	Y_4	Y_5
Z_1	0.28272	0.81647	0.52666	0.18164	0.1126
Z_2	0.7472	0.8269	0.8483	0.6993	0.0434
Z_3	0.3444	0.3067	0.8545	0.5617	0.4649
Z_4	0.9195	0.3373	0.6859	0.3582	0.8915

5. Menjumlahkan setiap bobot sinyal *output*

Y_in_1=(0.59721*0.2872)+(0.4854*0.7472)+(0.52959*0.3444)+(0.5317*0.9195)

=1.2029

Y_in_2=(0.59721*0.81647)+(0.4854*0.8269)+(0.52959*0.3067)+(0.5317*0.3373)

=1.2308

Y_in_3=(0.59721*0.52666)+(0.4854*0.8483)+(0.52959*0.8545)+(0.5317*0.6859)

=1.5436

Y_in_4=(0.59721*0.18164)+(0.4854*0.6993)+(0.52959*0.5617)+(0.5317*0.3582)

=0.9359

Y_in_5=(0.59721*0.1126)+(0.4854*0.0434)+(0.52959*0.4649)+(0.5317*0.8915)

=0.8085

Berikut adalah hasil penjumlahan sinyal *output* dengan bobot yang diberikan :

Tabel 3.11 Jumlah Y_in_k bobot sinyal *hidden* ke *output*

Y_in_1	Y_in_2	Y_in_3	Y_in_4	Y_in_5
1.2029	1.2308	1.5436	0.9359	0.8085
1.1873	1.3243	1.6465	1.0446	0.7905
1.4946	1.4601	1.9331	1.2169	0.9584
1.5628	1.5568	2.0198	1.3576	0.9996
1.6453	1.6248	2.0955	1.4919	1.0053
1.7287	1.7221	2.1283	1.5770	1.1491
1.8206	1.7623	2.1803	1.6860	1.1593
1.8449	1.8380	2.2226	1.7666	1.1447
1.8703	1.8874	2.2677	1.8537	1.1167
1.8537	1.9399	2.2889	1.9118	1.1708
1.8434	1.9905	2.3141	1.9789	1.1493
1.8094	2.0216	2.3383	2.0148	1.1533

6. Menerapkan Fungsi Aktivasi Menghitung Sinyal *Output*

Fungsi aktivasi yang digunakan adalah fungsi aktivasi sigmoid biner, berikut merupakan contoh proses perhitungan sinyal *output* :

$$Y_1 = \frac{1}{(1+2.71)^{1.2029}} = 0.23161$$

$$Y_2 = \frac{1}{(1+2.71)^{1.2308}} = 0.2267$$

$$Y_3 = \frac{1}{(1+2.71)^{1.5436}} = 0.1767$$

$$Y_4 = \frac{1}{(1+2.71)^{0.9359}} = 0.28231$$

$$Y_5 = \frac{1}{(1+2.71)^{0.8085}} = 0.30873$$

Berikut hasil perhitungan sinyal *output*dengan menerapkan fungsi aktivasi sigmoid biner :

Tabel 3.12 Hasil Perhitungan sinyal *output*dengan Fungsi *Sigmoid Biner*

Y_1	Y_2	Y_3	Y_4	Y_5
0.23161	0.2267	0.1767	0.28231	0.30873
0.23439	0.21078	0.16227	0.26088	0.31257
0.18391	0.18914	0.12707	0.22914	0.27778
0.17393	0.1748	0.11778	0.2053	0.26962
0.16243	0.16522	0.11016	0.18432	0.26851
0.15143	0.15227	0.107	0.17191	0.24129
0.14003	0.14717	0.10214	0.15698	0.23944
0.13714	0.13795	0.09834	0.14663	0.24211
0.13417	0.1322	0.09442	0.1361	0.24726
0.1361	0.12631	0.09263	0.12944	0.23736
0.13731	0.12085	0.09054	0.12208	0.24126
0.14138	0.11759	0.08858	0.1183	0.24053

2. Perhitungan *Backpropagation*

Kemudian proses selanjutnya, *backpropagation.* Pada proses perambatan balik tahap-tahapnya adalah sebagai berikut :

1. Hitung *erroroutput*

Pada proses ini unit *output*berusaha mencapai nilai target yang sesuai dengan pola *input* pelatihan kemudian menghitung *error*yang dihasilkan. Berikut adalah target dari data yang telah dinormalisasikan data ke-1 pada Tabel 3.13Target data ke-1 setelah normalisasi :

Tabel 3.13 Target data ke-1 setelah normalisasi

No	Nama Perlakuan	Pengaruh pertumbuhan dan hasil pada Jagung				
		BK	BB	DT	PT	HP
1	K3	0.165882	0.763291	0.892079	0.816308	0.235802

δ_1 = (0.165882*0.23161) *(1-0.23161) * 0.23161 = -0.0117

δ_2 = (0.763291*0.2267) *(1-0.2267) * 0.2267 = 0.0941

δ_3 = (0.892079*0.1767) *(1-0.1767) * 0.1767= 0.1041

δ_4 = (0.816308*0.28231) *(1-0.28231) * 0.28231= 0.1082

δ_5 = (0.235802*0.30873) *(1-0.30873) * 0.30873= -0.0156

Berikut adalah hasil *error* antara *output*dengan target yang diberikan sebagai data latih pada Tabel 3.14*ErrorOutput* ke Target :

Tabel 3.14*Error Output* ke Target

δ_1	δ_2	δ_3	δ_4	δ_5
-0.0117	0.0941	0.1041	0.1082	-0.0156
0.0114	-0.0184	-0.0085	-0.0310	-0.0457
0.0658	0.0884	0.0857	0.1170	0.0381
0.0773	0.0604	0.0706	0.1104	0.0138
0.0895	0.1013	0.0588	0.0897	0.1240
0.0962	0.0478	0.0629	0.1036	0.0230
0.0484	0.0923	0.0703	0.0942	0.0034
0.0468	0.0596	0.0612	0.0933	-0.0084
0.0206	0.0829	0.0621	0.0869	0.0663
0.0179	0.0655	0.0479	0.0829	-0.0025
0.0000	0.0650	0.0634	0.0689	0.0216
-0.0050	0.0625	0.0508	0.0794	0.0420

2. Menghitung perubahan bobot

Selanjutnya proses menghitung perubahan bobot, proses ini menggunakan *momentum*dan *learning rate*. Nilai *momentum* dan *learning rate* ditentukan sesuai penelitian Navalertpon, yaitu 0.4 dan 0.64 [NAV-11]. Proses ini juga sekaligus mengirimkan informasi *error*yang didapatkan ke lapisan paling kanan yaitu*output*. Berikut adalah contoh manualisasi perhitungan perubahan bobot terhadap *outputlayer* :

ΔW_{11}= (0.64*-0.0117*0.59721)+(0.4*0.64*-0.0117*0.59721) = -0.00626

ΔW_{12} = (0.64*0.0941*0.59721)+(0.4*0.64*0.0941*0.59721) = -0.050336

ΔW_{13} = (0.64*0.1041*0.59721)+(0.4*0.64*0.1041*0.59721) = 0.055688

ΔW_{14} = (0.64*0.1082*0.59721)+(0.4*0.64*0.1082*0.59721) = 0.057894

ΔW_{15} = (0.64*-0.0156*0.59721)+(0.4*0.64*-0.0156*0.59721) = -0.00833

Berikut adalah perubahan bobot yang didapatkan pada data latih ke-1 pada iterasi yang pertama pada Tabel 3.15 Perubahan Bobot :

Tabel 3.15 Perubahan Bobot

$\Delta W_{j,k}$	Y_1	Y_2	Y_3	Y_4	Y_5
Z_1	-0.00626	0.050336	0.055688	0.057894	-0.00833
Z_2	-0.00509	0.040917	0.045268	0.047061	-0.00677
Z_3	-0.00555	0.044636	0.049383	0.051339	-0.00739
Z_4	-0.00557	0.044816	0.049581	0.051546	-0.00298

.

3. Menjumlahkan Delta *input*

Kemudian setiap unit tersembunyi menjumlahkan setiap delta *input* nya pada unit-unit yang berada pada lapisan paling kanan atau *output.*

δ_in_1=(0.0117*0.2872)+(0.0941*0.81647)+(0.1041*0.52666)+(0.1082*0.18164) +(-0.0156*0.1126) = 0.1462

δ_in_2=(0.0117*0.7472)+(0.0941*0.8269)+(0.1041*0.8483)+(0.1082*0.6993) +(-0.0156*0.0434) = 0.2323

δ_in_3=(0.0117*0.3444)+(0.0941*0.3067)+(0.1041*0.8545)+(0.1082*0.5617) +(-0.0156*0.4649) = 0.1673

δ_in_4=(0.0117*0.9195)+(0.0941*0.3373)+(0.1041*0.6859)+(0.1082*0.3582) +(-0.0156*0.8915) = 0.1848

Berikut adalah hasil menjumlahkan setiap unit tersembunyi kepada *output* , semuan data latih pada iterasi pertama :

Tabel 3.16 Jumlah delta unit tersembunyi ke *output*

δ_in_1	δ_in_2	δ_in_3	δ_in_4
0.1462	0.2323	0.1673	0.1848
-0.0300	-0.0400	-0.0502	-0.0564
0.1731	0.2879	0.2170	0.2429
0.1545	0.2699	0.1994	0.2179

0.2063	0.3085	0.2603	0.3145
0.1694	0.2902	0.2233	0.2404
0.2030	0.3012	0.2242	0.2590
0.1707	0.2653	0.2014	0.2165
0.2014	0.2773	0.2446	0.2897
0.1641	0.2353	0.1844	0.2068
0.1702	0.2292	0.1985	0.2190
0.1710	0.2279	0.2082	0.2392

4. Menghitung informasi *error*

Berikut adalah proses manualisasi untuk menghitung informasi *error* yang didapatkan dari *hiddenlayer*:

δ_1 = (0.59721*0.1462) *(1-0.1462) = 0.0351

δ_2 = (0.48546*0.2323) *(1-0.2323) = 0.0580

δ_3 = (0.52959*0.1673) *(1-0.1673) = 0.0417

δ_4 = (0.53172*0.1848) *(1-0.1848) = 0.0460

Hasil dari informasi *error*pada *hiddenlayer*untuk data latih pada iterasi pertama adalah :

Tabel 3. 17*Error* pada *HiddenLayer*

δ1	δ2	δ3	δ4
0.0352	0.0580	0.0417	0.0460
-0.0072	-0.0100	-0.0125	-0.0140
0.0431	0.0688	0.0519	0.0462
0.0385	0.0653	0.0481	0.0422
0.0514	0.0752	0.0628	0.0630
0.0423	0.0714	0.0544	0.0493
0.0507	0.0745	0.0546	0.0549
0.0427	0.0660	0.0494	0.0467
0.0503	0.0692	0.0600	0.0642
0.0410	0.0588	0.0455	0.0467
0.0426	0.0573	0.0490	0.0506
0.0427	0.0569	0.0516	0.0559

5. Perubahan Bobot pada *HiddenLayer*

Kemudian tahap terakhir pada perhitungan *backpropagation* adalah proses menghitung koreksi perubahan bobot pada *hiddenlayer*.

ΔV_{11}= (0.64*0.2*0.0352)+(0.4*0.64*0.2*0.0352) = 0.0063025

ΔV_{21} = (0.64*0.277*0.0580)+(0.4*0.64*0.277*0.0580) = 0.0087289

ΔV_{31} = (0.64*0.2*0.0417)+(0.4*0.64*0.2*0.0417) = 0.0063025

ΔV_{41} = (0.64*0.1*0.0460)+(0.4*0.64*0.1*0.0460) = 0.0031512

Berikut adalah perubahan bobot pada *hiddenlayer*yang didapatkan pada data latih ke-1 pada iterasi yang pertama pada tabel 3.18 :

Tabel 3.18 Perubahan Bobot*HiddenLayer*

.

$\Delta V_{i,j}$	Z_1	Z_2	Z_3	Z_4
X_1	0.0063025	0.0103989	0.00747	0.00523
X_2	0.0087289	0.0144024	0.01034	0.00725
X_3	0.0063025	0.0103989	0.00747	0.00523
X_4	0.0031512	0.0051994	0.00373	0.00262

6. Hitung MSE

Nilai MSE yang didapatkan untuk data ke-1 dan rata-rata *error* yang dihasilkan oleh data latih pada iterasi pertama adalah sebagai berikut :

error = 1/5 *((0.16588-0.2361) + (0.76329-0.2267) + (0.89208-0.1767) + (0.81631-0.28231) + (0.2358-0.30873)= 0.32946238

Tabel 3.19*Error* pada iterasi pertama

	Error
	0.32946238
	-0.0966201
	0.5288875
	0.47714443
	0.64502192
	0.52585739
	0.52693329
	0.45782289
	0.54517148
	0.40748315
	0.42845488
	0.43572537
MSE	0.22015

Nilai MSE yang didapatkan untuk 12 data latih pada iterasi pertama adalah MSE = 1/12 * $\sum$ $(error)^2$ = 0.22015

3. Perhitungan Perbaikan Bobot

Pada perhitungan ini akan dicari bobot terbaru pada setiap unit *output* maupun unit tersembunyi, berikut adalah langka-langkah proses manualisasinya :

1. Perubahan Bobot pada unit *output*

Proses perhitungan Perubahan Bobot pada unit *output* sebagi berikut :

W_{11}=0.28272 + (-0.00626) = 0.27467

W_{12}=0.81647 + (0.050336) =0.8668

W_{13}=0.52666 + (0.055688) =0.52666

W_{14}=0.18164 + (0.057894) =0.18164

W_{15}=0.1126 + (-0.00833) =0.1126

Berikut adalah hasil dari perubahan bobot untuk data ke-2 :

Tabel 3.20 Bobot baru pada unit *output*

$W_{j,k}$	Y_1	Y_2	Y_3	Y_4	Y_5
Z_1	0.27646	0.8668	0.58235	0.23953	0.10427
Z_2	0.74211	0.86782	0.89356	0.74638	0.03662
Z_3	0.3389	0.35132	0.90391	0.613	0.45749
Z_4	0.9140	0.38212	0.73549	0.40978	0.88853

2. Perubahan Bobot pada unit *hidden*

Proses perhitungan Perubahan Bobot pada unit *hidden*sebagi berikut :

V_{11}=0.129857 + (0.0063025) = 0.136160

V_{21}=(-0.911927) + (0.0087289) = -0.903198

V_{31}=(-0.895038) + (0.0063025) =0.888736

V_{41}=0.105887 + (0.0031512) =0.109039

Berikut adalah hasil dari perubahan bobot *hidden*untuk data ke-2 masih pada iterasi pertama :

Tabel 3.21 Bobot baru pada unit *hidden*

$V_{i,j}$	Z_1	Z_2	Z_3	Z_4
X_1	0.136160	0.184032	0.179329	-0.387316
X_2	-0.903198	0.470197	0.291558	-0.373678
X_3	-0.888736	-0.172031	-0.849137	0.796991
X_4	0.109039	-0.656032	-0.594205	-0.989780

3.3.3.2Iterasi Kedua

Selanjutnya, untuk proses iterasi ke-dua bobot yang digunakan oleh*outputlayer*dan *hiddenlayer* adalah bobot data terakhir yang di dapatkan pada iterasi pertama. Iterasi dilanjutkan kembali dimulai dari dari data ke-1 data latih, datanya terletak pada tabel 3.4 Data latih Normalisasi. Prosesnya adalah sebagai berikut :

1. Perhitungan *Feedforward*

1. Proses *Inputlayer* ke *hiddenlayer*

Bobot yang digunakan untuk proses *inputlayer*ke *hiddenlayer* adalah bobot (V_{ij}) yang didapatkan dari data terakhir, pada iterasi pertama. Data latih yang ke-1 akan dihitung proses manualisasinya sesuai dengan tabel 3.6 Data K3 Normalisasi. Ini adalah bobot baru untuk data latih ke-1 pada iterasi kedua.

Tabel 3.22 Bobot*Input* ke *Hiddenlayer*iterasi kedua

$V_{i,j}$	Z_1	Z_2	Z_3	Z_4
X_1	0.207017	0.290887	0.262302	-0.311004
X_2	-0.805060	0.618192	0.406477	-0.267986
X_3	-0.857560	-0.127209	-0.813863	0.82770
X_4	0.433068	-0.168032	-0.211857	-0.624325

2. Menjumlahkan bobot sinyal *input*

Kemudian menjumlahkan bobot sinyal*input* :

Z_in_1= (0.2*0.207029)+(0.277*-0.805044)+(0.2*-0.857555)+(0.1*0.433123)
=-0.3098

Z_in_2 = (0.2*0.290899)+(0.277*0.618208)+(0.2*-0.127204)+(0.1*-0.167972)
=0.18719

Z_in_3 = (0.2*0.262327)+(0.277*0.406511)+(0.2*-0.813856)+(0.1*-0.211751)
=-0.0189

Z_in_4 = (0.2*-0.306741)+(0.277*-0.262081)+(0.2*0.832575)+(0.1*-0.617121)
=-0.0333

Berikut adalah hasil menjumlahkan bobot sinyal yang didapatkan dari 12 data latih yang telah dinormalisasi :

Tabel 3.23 Jumlah Z_in_j bobot sinyal *input*iterasi kedua

Z_in_1	Z_in_2	Z_in_3	Z_in_4
-0.3098	0.18717	-0.0189	-0.0333
-0.3029	0.19675	-0.0109	-0.0258
0.21666	0.09041	-0.0163	-0.6897
0.26333	0.15635	0.04013	-0.6309
0.26267	0.21213	0.0512	-0.5342
0.31669	0.28438	0.11888	-0.4501
0.31973	0.34564	0.13598	-0.3406
0.37139	0.4146	0.19395	-0.2771
0.37413	0.47106	0.20676	-0.1773
0.42734	0.5372	0.27137	-0.102
0.42969	0.58883	0.28214	-0.0054
0.47604	0.64462	0.33653	0.05279

Bobot untuk data ke-2 sampai ke-12 didapatkan pada proses perubahan bobot pada tahap terakhir, hanya data ke-1 pada iterasi kedua yang bobotnya didapatkan dari bobot terakhir pada iterasi sebelumnya.

3. Menerapkan Fungsi Aktivasi menghitung sinyal *input*

Fungsi aktivasi yang digunakan adalah fungsi aktivasi *sigmoid biner*, berikut merupaka contoh proses perhitungan sinyal *input*:

e = Bilangan *eural* yaitu 2.71828

$$Z_1 = \frac{1}{(1+2.71)^{-0.3098}} = 0.5766$$

$$Z_2 = \frac{1}{(1+2.71)^{0.1872}} = 0.4353$$

$$Z_3 = \frac{1}{(1+2.71)^{-0.0189}} = 0.5047$$

$$Z_4 = \frac{1}{(1+2.71)^{-0.0333}} = 0.50831$$

Berikut hasil sinyal *input* dengan menerapkan fungsi aktivasi sigmoid biner :

Tabel 3.24 Hasil Perhitungan sinyal *Input*iterasi kedua

Z_1	Z_2	Z_3	Z_4
0.57661	0.45348	0.50471	0.50831
0.57492	0.45112	0.50272	0.50643
0.44621	0.47748	0.50406	0.66544
0.43474	0.46111	0.49	0.65226
0.4349	0.44733	0.48724	0.63009
0.42172	0.42959	0.47041	0.61034
0.42098	0.4147	0.46616	0.58409
0.40848	0.39811	0.45181	0.56863
0.40782	0.3847	0.44865	0.54407
0.39507	0.36922	0.43277	0.52541
0.39451	0.35732	0.43014	0.50135
0.38353	0.34465	0.4169	0.48685

4. Bobot dari *hiddenlayer* ke *inputlayer*

Untuk data ke-1 bobot diperoleh dari bobot data terakhir, pada iterasi sebelumnya, sedangkan data ke-2 sampai ke-12 bobot ditentukan berdasarkan proses perubahan bobot. Berikut bobot dari *hiddenlayer* ke *outputlayer* untuk data ke-1 proses iterasi kedua :

Tabel 3.25 Bobot *Outputlayer* iterasi kedua

$W_{j,k}$	Y_1	Y_2	Y_3	Y_4	Y_5
Z_1	0.49766	1.16311	0.84685	0.61386	0.21003
Z_2	0.98304	1.18881	1.17965	1.15468	0.15814
Z_3	0.58705	0.6809	1.20273	1.03846	0.58256
Z_4	1.21555	0.78794	1.09657	0.92647	1.03955

5. Menjumlahkan setiap bobot sinyal *output*

Y_in_1=(0.5766*0.4976)+(0.4353*0.9830)+(0.5047*0.5870)+(0.50831*1.21555)

=1.6469

Y_in_2=(0.5766*1.1631)+(0.4353*1.1888)+(0.5047*0.6809)+(0.50831*0.78794)

=1.9571

Y_in_3=(0.5766*0.8468)+(0.4353*1.1796)+(0.5047*1.2027)+(0.50831*1.0965)

=2.1877

Y_in_4=(0.5766*0.6138)+(0.4353*1.15468)+(0.5047*1.0384)+(0.50831*0.9264)

=1.8726

Y_in_5=(0.5766*0.21003)+(0.4353*0.1581)+(0.5047*0.58256)+(0.50831*1.0395)

=1.0153

Berikut adalah hasil penjumlahan sinyal *output* dengan bobot yang diberikan :

Tabel 3.26 Jumlah Y_in_k iterasi kedua

Y_in_1	Y_in_2	Y_in_3	Y_in_4	Y_in_5
1.6469	1.9571	2.1877	1.8726	1.0153
1.6408	2.0148	2.2467	1.9393	1.0058
1.8142	2.0220	2.3463	2.0381	1.1174
1.8248	2.0334	2.3480	2.0614	1.1318
1.8456	2.0413	2.3549	2.0917	1.1213
1.8547	2.0431	2.3188	2.0784	1.1923
1.8749	2.0296	2.3074	2.0905	1.1765
1.8541	2.0273	2.2857	2.0832	1.1455
1.8361	2.0215	2.2749	2.0896	1.1059
1.7855	2.0039	2.2378	2.0681	1.1141
1.7461	2.0005	2.2174	2.0674	1.0781
1.6917	1.9796	2.1917	2.0444	1.0583

5. Menerapkan Fungsi Aktivasi menghitung sinyal *output*

Fungsi aktivasi yang digunakan adalah fungsi aktivasi sigmoid biner, berikut merupaka contoh proses perhitungan sinyal *output* :

$$Y_1 = \frac{1}{(1+2.71)^{1.6469}} = 0.16221$$

$$Y_2 = \frac{1}{(1+2.71)^{1.9571}} = 0.12443$$

$$Y_3 = \frac{1}{(1+2.71)^{2.1877}} = 0.10147$$

$$Y_4 = \frac{1}{(1+2.71)^{1.87266}} = 0.1339$$

$$Y_5 = \frac{1}{(1+2.71)^{1.0153}} = 0.26656$$

Berikut hasil perhitungan sinyal *output*dengan menerapkan fungsi aktivasi sigmoid biner :

Tabel 3.27 Hasil Perhitungan sinyal *output*iterasi kedua

Y_1	Y_2	Y_3	Y_4	Y_5
0.16221	0.12443	0.10147	0.1339	0.26656
0.16304	0.1183	0.09623	0.12637	0.26841
0.1408	0.11755	0.08793	0.1159	0.24713
0.13952	0.11637	0.0878	0.11354	0.24447
0.13705	0.11557	0.08725	0.11053	0.24641
0.13598	0.11539	0.09016	0.11184	0.23351
0.13363	0.11677	0.0911	0.11064	0.23634
0.13606	0.117	0.0929	0.11137	0.24196
0.13818	0.1176	0.09381	0.11073	0.24926
0.1443	0.11943	0.09701	0.11287	0.24775
0.14921	0.1198	0.0988	0.11294	0.25449
0.15623	0.12201	0.1011	0.11525	0.25825

2. Perhitungan *Backpropagation*

Kemudian proses selanjutnya, *backpropagation.* Pada proses perambatan balik tahap-tahapnya adalah sebagai berikut :

1. Hitung *erroroutput*

Pada proses ini unit *output*berusaha mencapai nilai target yang sesuai dengan pola *input* pelatihan kemudian menghitung *error*yang dihasilkan. Target dari data yang telah dinormalisasikan data ke-1 terletak pada tabel 3.13.

δ_1 = (0.165882*0.16239) *(1-0.16239) * 0.16239= -0.005

δ_2 = (0.763291*0.12453) *(1-0.12453) * 0.12453= 0.0696

δ_3 = (0.892079*0.10157) *(1-0.10157) * 0.10157= 0.0721

δ_4 = (0.816308*0.13401) *(1-0.13401) * 0.13401= -0.0791

δ_5 = (0.235802*0.26769) *(1-0.26769) * 0.26769= -0.0060

Berikut adalah hasil *error* antara *output*dengan target yang diberikan sebagai data latih pada tabel 3.28 :

Tabel 3.28*Errot Output* ke Target iterasi kedua

δ_1	δ_2	δ_3	δ_4	δ_5
0.0005	0.0696	0.0721	0.0791	-0.0060
0.0184	-0.0019	0.0003	-0.0029	-0.0331
0.0583	0.0672	0.0651	0.0795	0.0411
0.0687	0.0491	0.0568	0.0773	0.0176
0.0808	0.0802	0.0496	0.0659	0.1214
0.0898	0.0416	0.0554	0.0783	0.0238
0.0473	0.0789	0.0644	0.0746	0.0039
0.0467	0.0540	0.0587	0.0772	-0.0084
0.0207	0.0765	0.0618	0.0753	0.0663
0.0178	0.0632	0.0495	0.0754	-0.0045
-0.0015	0.0647	0.0678	0.0653	0.0199
-0.0074	0.0641	0.0560	0.0779	0.0406

2. Menghitung perubahan bobot

Nilai *momentum* dan *learning rate*yaitu 0.4 dan 0.64. Proses ini juga sekaligus mengirimkan informasi *error*yang didapatkan ke lapisan paling kanan yaitu*output*. Berikut adalah contoh manualisasi perhitungan perubahan bobot terhadap *outputlayer* :

ΔW_{11}= (0.64*0.0005*0.5766)+(0.4*0.64*0.0005*0.5766) = 0.000258

ΔW_{12} = (0.64*0.0696*0.5766)+(0.4*0.64*0.0696*0.5766) = 0.03596

ΔW_{13} = (0.64*0.0721*0.5766)+(0.4*0.64*0.0721*0.5766) = 0.037241

ΔW_{14} = (0.64*0.0791*0.5766)+(0.4*0.64*0.0792*0.5766) = 0.040886

ΔW_{15} = (0.64*-0.0060*0.5766)+(0.4*0.64*-0.0063*0.5766) = -0.00311

Berikut adalah perubahan bobot yang didapatkan pada data latih ke-1 pada iterasi yang pertama pada tabel 3.29 :

Tabel 3.29 Perubahan Bobot*Output*iterasi kedua

$\Delta W_{j,k}$	Y_1	Y_2	Y_3	Y_4	Y_5
Z_1	0.000258	0.03596	0.037241	0.040886	-0.00311
Z_2	0.000203	0.028282	0.029289	0.032156	-0.00244
Z_3	0.000226	0.031476	0.032597	0.035788	-0.00272
Z_4	0.000227	0.0317	0.032829	0.036043	-0.00274

3. Menjumlahkan Delta *input*

Kemudian setiap unit tersembunyi menjumlahkan setiap delta *input* nya pada unit-unit yang berada pada lapisan paling kanan atau *output.*

δ_in_1=(0.0005*0.4977)+(0.0696*1.1631)+(0.0721*0.8469)+(0.0792*0.6139) +(-0063*0.1585) = 0.1896

δ_in_2=(0.0005*0.983)+(0.0696*1.1888)+(0.0721*1.1797)+(0.0792*1.1547) +(-0063*0.2104) = 0.2588

δ_in_3=(0.0005*0.587)+(0.0696*0.6871)+(0.0721*1.2028)+(0.0792*1.0385) +(-0063*0.5829) = 0.2135

δ_in_4=(0.0005*1.2155)+(0.0696*0.7879)+(0.0721*1.0965)+(0.0792*0.9264) +(-0063*1.0292) = 0.2015

Berikut adalah hasil menjumlahkan setiap unit tersembunyi kepada *output* , semuan data latih pada iterasi pertama :

Tabel 3. 30 Jumlah delta unit tersembunyi ke *output*iterasi kedua

δ_in_1	δ_in_2	δ_in_3	δ_in_4
0.1896	0.2587	0.2135	0.2016
-0.0016	0.0075	-0.0125	-0.0160
0.2274	0.3183	0.2719	0.3183
0.2049	0.2983	0.2473	0.2914
0.2640	0.3510	0.3254	0.4333
0.2228	0.3267	0.2749	0.3398
0.2525	0.3385	0.2775	0.3115
0.2191	0.3042	0.2507	0.2769
0.2591	0.3285	0.3071	0.3565
0.2129	0.2795	0.2356	0.2488
0.2240	0.2805	0.2561	0.2688
0.2288	0.2838	0.2703	0.2899

4. Menghitung informasi *error*

Berikut adalah proses manualisasi untuk menghitung informasi *error* yang didapatkan dari *hiddenlayer*:

δ_1 = (0.1896*0.5766) *(1-0.5766) = 0.0463

δ_2 = (0.2588*0.4535) *(1-0.4535) = 0.0641

δ_3 = (0.2135*0.5047) *(1-0.5047) = 0.0534

δ_4 = (0.2015*0.5073) *(1-0.5073) = 0.0504

Hasil dari informasi *error*pada *hiddenlayer*untuk data latih pada iterasi pertama adalah :

Tabel 3. 31*Error* pada *HiddenLayer*iterasi kedua

δ1	δ2	δ3	δ4
0.0463	0.0641	0.0534	0.0504
-0.0004	0.0019	-0.0031	-0.0040
0.0562	0.0794	0.0680	0.0709
0.0504	0.0741	0.0618	0.0661
0.0649	0.0868	0.0813	0.1010
0.0543	0.0801	0.0685	0.0808
0.0616	0.0822	0.0691	0.0757
0.0529	0.0729	0.0621	0.0679
0.0626	0.0778	0.0760	0.0884
0.0509	0.0651	0.0578	0.0620
0.0535	0.0644	0.0628	0.0672
0.0541	0.0641	0.0657	0.0724

5. Perubahan Bobot pada *HiddenLayer*

Kemudian tahap terakhir pada perhitungan *backpropagation* adalah proses menghitung koreksi perubahan bobot pada *hiddenlayer*.

ΔV_{11}= (0.64*0.2*0.0463)+(0.4*0.64*0.2*0.0463) = 0.008296

ΔV_{21} = (0.64*0.277*0.0641)+(0.4*0.64*0.277*0.0641) = 0.011489

ΔV_{31} = (0.64*0.2*0.0534)+(0.4*0.64*0.2*0.0534) = 0.008296

ΔV_{41} = (0.64*0.1*0.0504)+(0.4*0.64*0.1*0.0504) = 0.004148

Berikut adalah perubahan bobot pada *hiddenlayer*yang didapatkan pada data latih ke-1 pada iterasi yang kedua pada tabel 3.32 :

Tabel 3. 32 Perubahan Bobot *HiddenLayer* iterasi kedua

.

$\Delta V_{i,j}$	Z_1	Z_2	Z_3	Z_4
X_1	0.0082956	0.0114894	0.0082956	0.0041478
X_2	0.0114894	0.0082956	0.0041478	0.0114937
X_3	0.0082956	0.0041478	0.0114937	0.0159188
X_4	0.0041478	0.0114937	0.0159188	0.0114937

6. Hitung MSE

Nilai MSE yang didapatkan untuk data ke- 1 dan rata-rata *error* yang dihasilkan oleh data latih pada iterasi pertama adalah sebagai berikut :

error = 1/5 *((0.16588-0.162) + (0.76329-0.125) + (0.89208-0.102) + (0.81631-0.134) + (0.2358-0.268))= 1.4994619

Berikut adalah hasil *error* yang dihasilkan oleh setiap data latih pada iterasi kedua, Tabel 3.33 *Error*pada iterasi kedua :

Tabel 3. 33*Error* pada iterasi kedua

	Error
	0.4169589
	-0.01494247
	0.5876297
	0.5244597
	0.68273956
	0.55358323
	0.5462473
	0.47003709
	0.55129794
	0.40770127
	0.42391753
	0.42693558
MSE	0.24321

Sehingga nilai MSE yang didapatkan untuk 12 data latih pada iterasi kedua adalah :

MSE = 1/12 * $\sum$ $(error)^2$ = 0.24321

3. Perhitungan Perbaikan Bobot

Pada perhitungan ini akan dicari bobot terbaru pada setiap unit *output* maupun unit tersembunyi, berikut adalah langka-langkah proses manualisasinya :

1. Perubahan Bobot pada unit *output*

Proses perhitungan Perubahan Bobot pada unit *output* sebagi berikut :

W_{11}=0.49766+ (0.000258) = 0.49792

W_{12}=1.16311+ (0.03596) = 1.119907

W_{13}=0.84685+ (0.037241) =0.88409

W_{14}=0.61386+ (0.040886) =0.65475

W_{15}=0.210+ (-0.00311) =0.20692

Berikut adalah hasil dari perubahan bobot untuk data ke-2 masih pada iterasi pertama :

Tabel 3.34 Bobot baru pada unit *output*iterasi kedua

$W_{j,k}$	Y_1	Y_2	Y_3	Y_4	Y_5
Z_1	0.49792	1.19907	0.88409	0.65475	0.20692
Z_2	0.98324	1.21709	1.20894	1.18684	0.1557
Z_3	0.58728	0.71857	1.23533	1.07425	0.57984
Z_4	1.2158	0.81964	1.1294	0.96251	1.03681

2. Perubahan Bobot pada unit *hidden*

Proses perhitungan Perubahan Bobot pada unit *hidden*sebagi berikut :

V_{11}=0.207017+ 0.00829= 0.215310

V_{21}=(-0.805060) + 0.01149= -0.793574

V_{31}=(-0.857560) + 0.00829=-0.849267

V_{41}=0.433068+ 0.00415=0.437214

Proses perubahan bonbot pada unit *hidden* ini menjadi langkah terakhir pada proses pelatihan data latih, *backpropagtion.* Bobot terakhir didapatkan, agar digunakan pada proses pengujian data uji.Berikut adalah hasil dari perubahan bobot *hidden*untuk data ke-2 untuk iterasi kedua :

Tabel 3.35 Bobot baru pada unit *hidden* iterasi kedua

$V_{i,j}$	Z_1	Z_2	Z_3	Z_4
X_1	0.215310	0.302377	0.271866	-0.301977
X_2	-0.793574	0.634104	0.419722	-0.255483
X_3	-0.849267	-0.115720	-0.804299	0.836738
X_4	0.437214	-0.162287	-0.207075	-0.619811

3.3.4 Perhitungan Pengujian Data Uji ANN *Backpropagation*

Pada manualisasi ini, iterasi hanya dilakukan sebanyak dua kali. Bobot terakhir data ke-12 pada iterasi kedua, dijadikan bobot terbaik untuk proses pengujian data uji. Proses pengujian hanya sampai pada tahapan *feedforward.* Data uji yang diproses sudah dinormalisasi terlebih dahulu sesuai pada tabel 3.5, berikut adalah proses pengujian data uji :

1. Proses*inputlayer* ke *hiddenlayer*

Bobot yang digunakan untuk proses *inputlayer*ke *hiddenlayer* adalah bobot (V_{ij}) yang didapatkan pada data terakhir, pada iterasi kedua. Ini adalah bobot baru untuk data uji dari iterasi kedua, yang akan digunakan pada data uji :

Tabel 3.36 Bobot*Input* ke *Hiddenlayer*Data Uji

$V_{i,j}$	Z_1	Z_2	Z_3	Z_4
X_1	0.306132	0.425048	0.380134	-0.180830
X_2	-0.66787	0.804004	0.569674	-0.87695
X_3	-0.808902	-0.063277	-0.757161	0.889290
X_4	0.846191	0.388398	0.2828368	-0.071792

2. Menjumlahkan bobot sinyal *input*

Kemudian menjumlahkan bobot sinyal*input* :

Z_in_1= (0.2*0.306132)+(0.277*-0.66787)+(0.2*-0.808902)+(0.1*0.846191)
=-0.2009

Z_in_2 = (0.2*0.425048)+(0.277*0.804004)+(0.2*-0.063277)+(0.1*0.388398)
=0.3339

Z_in_3 = (0.2*0.380134)+(0.277*0.569674)+(0.2*-0.757161)+(0.1*0.2828368)
=0.1106

Z_in_4 = (0.2*-0.180830)+(0.277*-0.87695)+(0.2*0.889290)+(0.1*-0.071792)
=0.1102

Berikut adalah hasil menjumlahkan bobot sinyal yang didapatkan dari 6 data uji yang telah dinormalisasi :

Tabel 3.37 Jumlah Z_in_j data uji

Z_in_1	Z_in_2	Z_in_3	Z_in_4
-0.2009	0.3339	0.11063	0.11022
0.63782	0.65728	0.48796	-0.1251
0.59738	0.65411	0.4501	-0.0806
0.55693	0.65095	0.41224	-0.0361
0.51649	0.64779	0.37438	0.00832
0.47604	0.64462	0.33653	0.05279

3. Menerapkan Fungsi Aktivasi menghitung sinyal *input*

Fungsi aktivasi yang digunakan adalah fungsi aktivasi *sigmoid biner*, berikut merupaka contoh proses perhitungan sinyal *input*:

$$Z_1 = \frac{1}{(1+2.71)^{-0.2009}} = 0.5499$$

$$Z_2 = \frac{1}{(1+2.71)^{0.3339}} = 0.4175$$

$$Z_3 = \frac{1}{(1+2.71)^{-0.1106}} = 0.4724$$

$$Z_4 = \frac{1}{(1+2.71)^{-0.1102}} = 0.4726$$

Berikut hasil sinyal *input* dengan menerapkan fungsi aktivasi sigmoid biner :

Tabel 3.38 Hasil Perhitungan sinyal *Input*Data Uji

Z_1	Z_2	Z_3	Z_4
0.54991	0.41754	0.47245	0.47256
0.34618	0.3418	0.38073	0.53113
0.35536	0.34251	0.38966	0.52008
0.36465	0.34322	0.39868	0.50901
0.37404	0.34393	0.40776	0.49793
0.38353	0.34465	0.4169	0.48685

4. Bobot dari *hiddenlayer* ke *inputlayer*

Berikut bobot dari *hiddenlayer* ke *outputlayer* untuk proses pengujian :

Tabel 3. 39 Bobot *Outputlayer*Data uji

$W_{j,k}$	Y_1	Y_2	Y_3	Y_4	Y_5
Z_1	0.67108	1.41374	1.0825	0.90408	0.29752
Z_2	1.15766	1.43005	1.40583	1.43521	0.25117
Z_3	0.77841	0.95713	1.45564	1.35163	0.68505
Z_4	1.46001	1.12042	1.40733	1.31369	1.17502

5. Menjumlahkan setiap bobot sinyal *output*

Y_in_1=(0.5499*0.67108)+(0.4175*1.15766)+(0.4724*0.7784)+(0.4715*1.46001)

=1.9101

Y_in_2=(0.5499*1.41374)+(0.4175*1.43005)+(0.4724*0.9571)+(0.4715*1.12042)

=2.3562

Y_in_3=(0.5499*1.0825+(0.4175*1.40583)+(0.4724*1.4556)+(0.4715*1.40733)

= 2.5350

Y_in_4=(0.5499*0.90408)+(0.4175*1.43521)+(0.4724*1.3516)+(0.4715*1.31369)

=2.3558

Y_in_5=(0.5499*0.29752)+(0.4175*0.25117)+(0.4724*0.68505)+(0.4715*1.7502)

=1.1474

Hasil penjumlahan sinyal *output* dengan bobot yang diberikan :

Tabel 3. 40 Jumlah Y_in_k Data Uji

Y_in_1	Y_in_2	Y_in_3	Y_in_4	Y_in_5
1.9101	2.3562	2.5350	2.3558	1.1474
1.6998	1.9377	2.1569	2.0159	1.0738
1.6976	1.9479	2.1653	2.0228	1.0698
1.6955	1.9582	2.1739	2.0298	1.0659
1.6935	1.9688	2.1827	2.0370	1.0621
1.6917	1.9796	2.1917	2.0444	1.0583

6. Menerapkan Fungsi Aktivasi menghitung sinyal *output*

Fungsi aktivasi yang digunakan adalah fungsi aktivasi sigmoid biner, berikut merupaka contoh proses perhitungan sinyal *output* :

$$Y_1 = \frac{1}{(1+2.71)^{1.9101}} = 0.12963$$

$$Y_2 = \frac{1}{(1+2.71)^{2.3562}} = 0.08715$$

$$Y_3 = \frac{1}{(1+2.71)^{2.5350}} = 0.07397$$

$$Y_4 = \frac{1}{(1+2.71)^{2.3558}} = 0.08718$$

$$Y_5 = \frac{1}{(1+2.71)^{1.1474}} = 0.24161$$

Berikut hasil perhitungan sinyal *output*dengan menerapkan fungsi aktivasi sigmoid biner :

Tabel 3.41 Hasil Perhitungan sinyal *output*Data Uji

Y_1	Y_2	Y_3	Y_4	Y_5
0.12963	0.08715	0.07397	0.08718	0.24161
0.15517	0.12655	0.1043	0.11819	0.25531
0.15545	0.12544	0.10352	0.11747	0.25606
0.15573	0.12431	0.10273	0.11675	0.2568
0.15599	0.12316	0.10192	0.116	0.25753
0.15623	0.12201	0.1011	0.11525	0.25825

3.3.5 Proses Denormalisasi

Proses denormalisasi dilakukan untuk mengubah kembali nilai pengaruh pada tanaman ke dalam nilai *real* atau asli. Y_1 adalah *output* dari bobot kering tanaman. Range nilai minimal dan maksimal untuk proses denormalisasi diambil berdasarkan Tabel 3.3 data uji sebelum normalisasi. Berikut adalah proses denormalisasi Tabel 3.41 Hasil Perhitungan sinyal *output*Data Uji pengaruh pada tanaman yang diperoleh pada proses pengujian:

$$\text{Denormalisasi } Y_1 = \frac{(5.93-4.37)*(0.12963-0.1)}{0.8} + 4.37 = 4.42777$$

Tabel 3.42 Hasil Denormalisasi Data Uji

Y_1"	Y_2"	Y_3"	Y_4"	Y_5"
4.42777	231.632	4.73056	19.1647	5.65365
4.47758	232.76	4.74156	19.25	5.67627
4.47814	232.728	4.74128	19.2481	5.6775
4.47867	232.696	4.74099	19.2461	5.67872
4.47918	232.663	4.7407	19.244	5.67993
4.47966	232.63	4.7404	19.2419	5.68111

3.4 Perhitungan Manual BPSO (Bidirectional Particle Swarm Optimization)

Proses optimasi menggunakan algoritma BPSO, yang dioptimasi pada perhitungan manual ini adalah *output* dari Y_1 dan Y_5 setelah denormalisasi.

Y_1adalah bobot kering dari tanaman sedangkan Y_5 adalah hasil produksi tanaman. Berikut adalah perhitungan manual untuk optimasi :

3.4.1 Menentukan Parameter BPSO

Parameter BPSO, dalam menentukan nilainya pada perancangan ini menggunakan nilai standard *particle swarm optimization* (PSO) yaitu :

- w = 1
- $c_1 = c_2$ = 2.05
- χ = 0.729
- *Population size* (jumlah populasi) = 5 partikel
- Jumlah iterasi = 1 kali

Kemudian menetapkan range pemberian dosis dan batas pertumbuhan pada tanaman agar optimasi masih dalam jangkau nilai yang wajar untuk diterapkan dan pertumbuhan yang baik :

- Urea = $0.177 \leq U \leq 0.533$ ton/ha
- SP_{36} = $0.0138 \leq S \leq 0.319$ ton/ha
- KCL = $0.04 \leq K \leq 0.3$ ton/ha
- Biochar = $10 \leq B \leq 50$ ton/ha
- Bobot Kering Tanaman = $4 \leq BK \leq 5$ ton/ha
- Hasil Produksi = $6 \leq HP \leq 8$ ton/ha

3.4.2 Inisialisasi Populasi Partikel dan GBEST

Pada proses ini, untuk iterasi awal dibangkitkan secara random, posisi populasi partikel sebanyak 5 (lima) partikel dan anggota member GBEST sebanyak 5 (lima). Setiap populasi yang dibangkitkan dihitung vektor dengan proses pengujian ANN. Untuk populasi partikel dicari kecepatan atau *velocity* dari setiap posisi. Berikut adalah proses membangkitkan posisi populasi partikel dan anggota GBEST :

Tabel 3.43 Populasi Partikel Awal

Posisi Partikel					Kecepatan Partikel				Posisi Vektor	
Data	X'Ua	X'Sa	X'Ka	X'Ba	V'Ua	V'Sa	V'Ka	V'Ba	f_1	f_2
a_1	0.178	0.09	0.116	11	-0.2431	-0.0431	-0.15	-27	0.24802	0.157584865
a_2	0.5	0.307	0.241	12	0.1432	-0.1615	0.12	-15	0.24708	0.157070697
a_3	0.432	0.215	0.113	14	0.0156	0.1023	0.13	10	0.24727	0.157237933
a_4	0.312	0.118	0.271	25	-0.1781	0.0817	-0.09	23	0.24705	0.156991151
a_5	0.417	0.178	0.186	39	0.2341	0.181	-0.05	27	0.24608	0.15658147

Posisi partikel dibangkitkan secara acak dengan nilai *range* yang telah ditentukan pada inisialisasi parameter BPSO. Kecepatan partikel dibangkitkan juga secara acak dengan range yang dihitung dengan persamaan (2-24). Kecepatan yang dihasilkan untuk setiap dosis masing-masing pupuk adalah :

$Vmaks_{urea} = 60\% * 0.533 = 0.3198$

$V_{urea} = (-Vmaks_{urea}, Vmaks_{urea}) = (-0.3198, 0.3198)$

$Vmaks_{sp36} = 60\% * 0.319 = 0.1914$

$V_{sp36} = (-Vmaks_{sp36}, Vmaks_{sp36}) = (-0.1914, 0.1914)$

$Vmaks_{kcl} = 60\% * 0.3 = 0.18$

$V_{kcl} = (-Vmaks_{kcl}, Vmaks_{kcl}) = (-0.18, 0.18)$

$Vmaks_{bionchar} = 60\% * 50 = 30$

$V_{biochar} = (-Vmaks_{biochar}, Vmaks_{biochar}) = (-30, 30)$

Sedangkan untuk vektor yang dihasilkan pada setiap posisi digunakan persamaan (2-25) dan(2-26). Jika didapatkan pada pengujian ANN, Y_1 dan Y_5 dari a_1 adalah 4.03921 dan 6.4579 maka contoh perhitungan vektor untuk $a_1$adalah :

$$f_1 = \frac{1}{4.03192} = 0.24802$$

$$f_2 = \frac{1}{6.34579} = 0.15758$$

Selanjutnya membangkitkan posisi untuk GBEST sebanyak 5 partikel, posisi GBEST dibangkitkan dengan range secara random, untuk menentukan posisi vektor dari f_1 dan f_2 dilakukan pengujian pada ANN *backpropagation* kemudian dihitung denga persamaan sebagi berikut :

Tabel 3.44 Populasi untuk GBEST

Member	U	S	K	B	f_1	f_2
M1	0.278	0.217	0.172	33	0.246547172	0.15680663
M2	0.52	0.105	0.198	27	0.246650018	0.15686532
M3	0.441	0.268	0.299	41	0.245740884	0.15634935
M4	0.24882	0.22645	0.17924	19.6	0.247250862	0.15715716
M5	0.43812	0.07776	0.11151	11.7	0.247604643	0.15740046

3.4.3 Menghasilkan Keturunan *Child$_1$* dan *Child$_2$*

Untuk posisi pertama dari *child$_1$* dan *child$_2$*sama dengan posisi dan kecepatan dari *parent* nya.

$Xi = X_l = X_r$

$V_i = V_l = Vr$

3.4.4 GBEST dan PBEST untuk *child$_1$*

GBEST untuk *child$_1$*dengan mencari NC terkecil dari populasi GBEST, rumus untuk mencari *euclidiean distance* menggunakan persamaan (2-29). Selanjutnya, mencari nilai *radius* dan *sigma share* dengan persamaan (2-31) dan (2-32) berikut adalah hasil NC dari populasi partikel untuk a_1:

Tabel 3.45NC Populasi GBEST

Member	U	S	K	B	NC	f_1	f_2	d1	d2	d3	d4	d5	r	niche radius
M1	0.278	0.217	0.172	33	0.5664	0.2465	0.1568	0	6.005	8.002	13.38	21.2	14.61	9.772
M2	0.52	0.105	0.198	27	0.6285	0.2466	0.15686	6.005	0	14.001	7.39	15.2		
M3	0.441	0.268	0.299	41	**0.1810**	0.2457	0.15634	8.002	14.001	0	21.39	29.2		
M4	0.2488	0.226	0.17924	19.6	0.44120	0.2472	0.15715	13.389	7.39	21.39	0	7.83		
M5	0.4381	0.0777	0.11151	11.7	0.1980	0.2476	0.15740	21.223	15.2	29.22	7.83	0		

$$D_2,M_1=\sqrt{(0.278-0.52)^2+(0.217-0.105)^2+(0.172-0.198)^2+(33-27)^2}=6.00$$

$$r=\frac{1}{2}*\sqrt{(0.52-0.278)^2+(0.268-0.105)^2+(0.299-.0172)^2+(41-27)^2}$$

$$=14.61$$

$$\sigma_{share}=\frac{14.61}{\sqrt[4]{5}}=9.772$$

$$NC,M_1=(1-(6.005/9.772))+(1-(8.002/9.772))=0.5664$$

Sehingga dipilih M_3, sebagai GBEST untuk *child*$_1$, sedangkan PBEST *child*$_{1=}$ GBEST *child*$_1$. Berikut pada Tabel 3.46 GBEST dan PBEST untuk*child*$_1$ yang diperoleh untuk posisi terpilih pada *child*$_1$:

Tabel 3.46GBEST dan PBEST untuk *child*$_1$

GBEST untuk*child*$_1$NC terkecil				PBEST *child*$_1$ = GBEST*child*$_1$			
X'Ub	X'Sb	X'Kb	X'Bb	P'Ub	P'Sb	P'Kb	P'Bb
0.441	0.268	0.299	41	0.441	0.268	0.299	41
0.441	0.268	0.299	41	0.441	0.268	0.299	41
0.441	0.268	0.299	41	0.441	0.268	0.299	41
0.441	0.268	0.299	41	0.441	0.268	0.299	41
0.441	0.268	0.299	41	0.441	0.268	0.299	41

3.4.5 GBEST dan PBEST untuk *child*$_2$

GBEST untuk *child*$_2$dengan mencari *euclidean distance* terkecil dari populasi GBEST, berikut adalah hasil *euclidean distance* dari populasi partikel :

Tabel 3.47*euclidean distance* untuk *child*$_2$

Data	Jarak antara d(g,h)					
	da,m_1	da,m_2	da,m_3	da,m_4	da,m_5	da,m_6
a_1	**0.0016663**	0.091538388	0.094821	15.16183099	10.8434	
a_2	0.0005368	0.000433926	0.001344	**0.00016697**	0.0005	0.000395
a_3	0.00072779	0.0006249	0.001535	**2.3924E-05**	0.000309	0.000204
a_4	0.00049991	0.000397043	0.001307	0.00066162	**0.00031**	
a_5	0.0004691	0.00057198	0.000337	**0.00030741**	0.00066	

$Da_1,m_{1=} \sqrt{(0.248-0.2465)^2+(0.1575-0.1568)^2} = 0.0016663$

$Da_1,m_{2=} \sqrt{(0.248-0.2466)^2+(0.1575-0.15686)^2} = 0.091538388$

$Da_1,m_{3=} \sqrt{(0.248-0.2457)^2+(0.1575-0.1563)^2} = 0.094821$

$Da_1,m_{4=} \sqrt{(0.248-0.2472)^2+(0.1575-0.1571)^2} = 15.161830$

$Da_1,m_{5=} \sqrt{(0.248-0.2476)^2+(0.1572-0.1574)^2} = 10.8434$

Populasi GBEST untuk a_1,a_4, dan $a_5$setelah di *update*populasi GBEST hanya terdiri dari 5 populasi sedangkan a_2 dan a_3 memiliki 6 anggota. Kemudian pada a_1 dipilih M_1sebagai PBEST dan GBEST karena jarak *Euclidean*

*distance*terpendek untuk *child*$_2$, sehingga didapatkan GBEST dan PBEST terbaru untuk *child*$_2$adalah sebagai berikut pada Tabel 3.48 GBEST dan PBEST untuk *child*$_2$:

Tabel 3.48GBEST dan PBEST untuk *child*$_2$

GBEST *child*$_2$				PBEST *child*$_2$			
X'Uc	X'Sc	X'Kc	X'Bc	P'Uc	P'Sc	P'Kc	P'Bc
0.278	0.217	0.172	33	0.278	0.217	0.172	33
0.248817	0.226453	0.179239	19.6102	0.248817	0.226453	0.179239	19.61019918
0.248817	0.226453	0.179239	19.6102	0.248817	0.226453	0.179239	19.61019918
0.399977	0.29229	0.223462	22.61903	0.399977	0.29229	0.223462	22.61903362
0.445484	0.302013	0.251413	27.6253	0.445484	0.302013	0.251413	27.62529611

3.4.6 Menghitung Kecepatan *child*$_1$ dan *child*$_2$

Dalam menghitung kecepatan untuk *child*$_1$ dan *child*$_2$dibutuhkan r_1,$r_2$sebagai bilangan random antara (0,1). Berikut adalah kecepatan baru yang dihasilkan untuk *child*$_1$ dan *child*$_2$:

Tabel 3.49r_1,r_2 untuk *child*$_1$ dan *child*$_2$

random (0,1) for child 1		random (0,1) for child 2	
r_1	r_2	r_1	r_2
0.362972763	0.531072729	0.792814893	0.855687963
0.962224362	0.901874655	0.028445645	0.16167015
0.562275555	0.157007994	0.954066999	0.158517166
0.617778431	0.780137547	0.105299242	0.966532007
0.245677968	0.01967891	0.261686485	0.376878111

Data	*Velocitychild*$_1$				*Velocitychild*$_2$			
	V'Ub	V'Sb	V'Kb	V'Bb	V'Uc	V'Sc	V'Kc	V'Bc
a_1	0.043228	0.150688	0.049232	5.660973	-0.06768	0.179678	-0.05177	11.59153
a_2	0.034118	-0.2336	0.227233	38.61645	0.059952	-0.1882	0.099531	-12.4778
a_3	0.018497	0.119359	0.189867	18.69039	-0.04393	0.106022	0.151525	11.82309
a_4	0.028207	0.321592	-0.04522	48.58851	-0.00378	0.427036	-0.18419	18.28238
a_5	0.235068	0.184631	-0.04544	27.08068	0.256107	0.276812	0.000538	18.2119

Tabel 3.50 Perubahan Kecepatan *child*$_1$ dan *child*$_2$

Berikut adalah proses menghitung kecepatan yang dihasilkan untuk *child*$_1$*adalah sebagai berikut :*

V'Ub,a_1=(1*-0.2431)+(2.05*0.3629*(0.178-0.178))+(2.05*0.531*(0.441-0.178))

= 0.043228

V'Sb,a_1=(1*-0.0431)+(2.05*0.3629*(0.09-0.09))+(2.05*0.531*(0.268-0.09))
=0.150688

V'Kb,a_1= (1*-0.15)+(2.05*0.3629*(0.116-0.116))+(2.05*0.268*(0.299-0.116))
=0.049232

V'Bb,a_1= (1*-27)+(2.05*0.3629*(11-11))+(2.05*0.268*(41-11))
=5.660973

3.4.7 Menghitung Posisi baru *child$_1$* dan *child$_2$*

Pada proses ini setiap posisi baru akan dilakukan perhitungan vektor menggunakan proses pengujian ANN. Selain itu, posis baru yang dihasilkan juga dilakukan pengecekan apakah posisi yang dihasilkan masih dalam rentang nilai yang diijinkan, pada parameter BPSO yang telah diinisialisasi di awal. Dijelaskan pada Tabel 3.51 Posisi Baru *child$_1$*:

Tabel 3.51 Posisi Baru *child$_1$*

Update posisi Child$_1$ X'b				Vektor posisi baru child$_1$	
X'Ub	X'Sb	X'Kb	X'Bb	f1	f2
0.209513	0.199852	0.15189	15.12685	0.247583	0.157341
0.524872	0.136702	0.406653	40.15139	0.24584	0.15633
0.445484	0.302013	0.251413	27.6253	0.246822	0.15694
0.332563	0.352441	0.238035	60.42102	0.244774	0.155923
0.588365	0.312596	0.152873	58.74182	0.24456	0.155925

X'Ub,a_1=(0.178 + (0.04328 * 0.729)) = 0.209513

X'Sb,a_1=(0.09 + (0.15068* 0.729)) = 0.199852

X'Kb,a_1=(0.116 + (0.049232* 0.729))= 0.15189

X'Bb,a_1=(11 + 5.660972*0.729)) = 15.12685

Sedangkan posisi baru yang dihasilkan dari *child$_2$*dijelaskan pada Tabel 3.52 Posisi Baru *child$_2$*sebagai berikut:

Tabel 3. 52 Posisi Baru *child$_2$*

Update posisi child$_2$ X'c				Vektor posisi baru child$_2$	
X'Uc	X'Sc	X'Kc	X'Bc	f1	f2
0.128658	0.220986	0.078262	19.45022	0.247478736	0.157325761
0.543705	0.169806	0.313558	2.903685	0.247674437	0.15732789
0.399977	0.29229	0.223462	22.61903	0.24673803	0.156890731
0.374974	0.443353	0.177098	45.55907	0.245355329	0.156251809
0.601176	0.360639	0.213194	59.80945	0.244378473	0.155797358

3.4.8 Memperbaharui GBEST

Proses perbaharui GBEST sesuai dengan keadaan kondisi pada sub bab 3.2.2.4. Pada $a_{1,}$ posisi *child*$_1$*feasible* atau layak, sedangkan *child*$_2$ yang dihasilkantidak *feasible*berikut adalah hasil update GBEST pada $a_1$pada Tabel 3.53 Memperbaharui GBEST untuk $a_{1:}$

Tabel 3.53 Memperbaharui GBEST untuk a_1

Member	U	S	K	B	NC	f1	f2
M1	0.278	0.217	0.172	33	0.566426668	0.246547172	0.15680663
M2	0.52	0.105	0.198	27	0.854724886	0.246650018	0.15686532
M3	0.441	0.268	0.299	41	**0.181040724**	0.245740884	0.15634935
M4	0.248817	0.226453	0.179239	19.61019918	1.761387636	0.246385509	0.15670164
M5	0.209513	0.199852	0.15189	15.12684918	1.098594307	0.24673803	0.15689073

3.4.9 Perbaharui Partikel

Kemudian dilakukan perubahan partikel sesuai dengan keadaan kondisi yang dihasilkan *child*$_1$ dan *child*$_2$sesuai sub bab 3.2.2.5, partikel yang diperbaharui nantinya akan dibandingkan hasil dengan vektor terkecil antara *child*$_1$ dan *child*$_2$sebagai berikut :

Tabel 3.54 Perbaharui Partikel

Posisi Partikel terbaru					Kecepatan Partikel Vi terbaru				Posisi Vektor	
posisi	X'Ua	X'Sa	X'Ka	X'Ba	V'Ua	V'Sa	V'Ka	V'Ba	f1	f2
a1	0.209	0.199	0.151	15.12	0.043228	0.15	0.049	5.66	0.2475	0.1573
a2	0.5	0.307	0.241	12	0.1432	-0.16	0.12	-15	0.24708	0.15707
a3	0.445484	0.301	0.251	27.62	0.0184	0.1193	0.1898	18.690	0.24638	0.1567
a4	0.312	0.118	0.271	25	-0.178	0.081	-0.09	23	0.24704	0.15699
a5	0.417	0.178	0.186	39	0.2341	0.181	-0.05	27	0.2465	0.1566

Pada Tabel 3.54 Perbaharui Partikel, posisi baru ini akan menjadi posisi partikel pada iterasi selanjutnya, sampai batas iterasi yang ditentukan. Pada sistem yang akan dibuat batas iterasi untuk proses BPSO adalah 100 iterasi. Jadi dilakukan pengulangan sampai 100 kali. Didapatkan hasil dari anggota GBEST adalah solusi optimasi terbaik, pada anggota a_5, pada Tabel 3.55 Perbaharui GBEST untuk $a_5$sebagai berikut

Tabel 3.55 Perbaharui GBEST untuk a_5

Member	U	S	K	B	NC	f_1	f_2
M1	0.278	0.217	0.172	33	0.761475546	0.246547172	0.15680663
M2	0.52	0.105	0.198	27	1.797366321	0.246650018	0.15686532
M3	0.441	0.268	0.299	41	**0**	0.245740884	0.15634935
M4	0.445484	0.302013	0.251413	27.62529611	1.79818609	0.246385509	0.15670164
M5	0.399977	0.29229	0.223462	22.61903362	0.978109937	0.24673803	0.15689073

Nilai pada anggota GBEST terkecil akan dipilih sebagi solusi terbaik setelah iterasi terakhir, pada semua data yang menjadi populasi GBEST. Sehingga pada proses manualisasi metode BPSO ini hanya dilakukan sebanyak jumlah anggota pada populasi partikel 5 buah dan iterasi satu kali saja. Anggota GBEST yang terpilih dengan vektor terkecil terdapat pada member GBEST M_3. GBEST dengan member M_3 menjadi solusi terbaik dan optimal dalam pemberian dosis pada pengaruh tanaman terhadap bobot kering dan hasil produksi. Vektor dari GBEST M_3 ini dikembalikan kedalam range aslinya yaitu :

$$\text{Bobot kering tanaman} = \frac{1}{0.2457408}$$

$$= 4.06932 \text{ton/ha}$$

$$\text{Hasil Produksi} = \frac{1}{0.156349}$$

$$= 6.39593 \text{ ton/ha}$$

Sehingga kesimpulan yang diperoleh pada proses manualisasi ANN dan BPSO, untuk mendapatkan bobot kering tanaman dengan hasil 4.06932 ton/ha dan hasil produksi6.39539ton/ha yang optimal maka disarankan pemberian terhadap masing-masing jenis dosis pupuk untuk pupuk Urea adalah 441 kg/ha, pupuk SP_{36}268 kg/ha, pupuk KCL sebanyak 299 kg/ha dan biochar41 ton/ha.

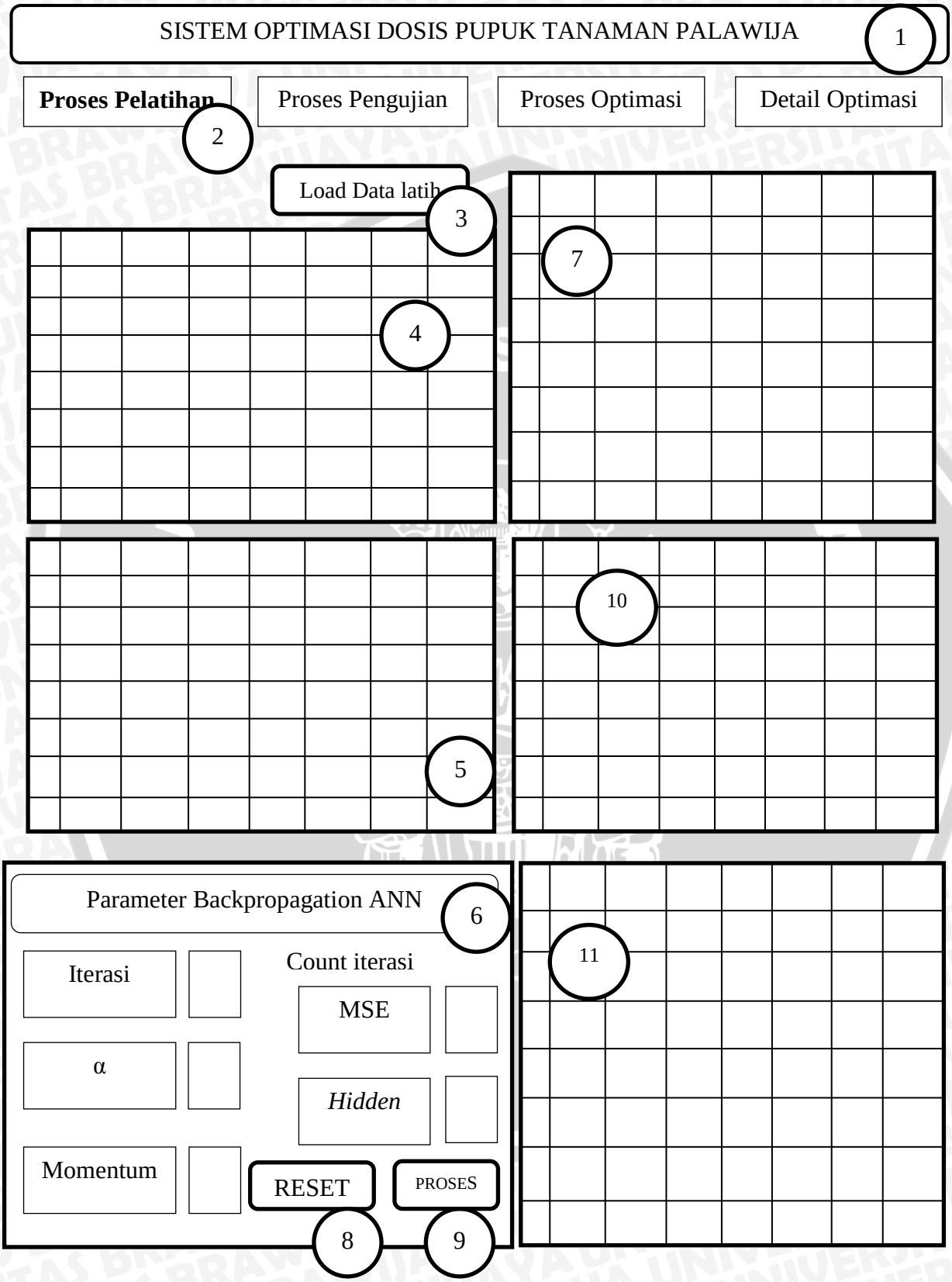
3.5 Perancangan *User Interface*

Pada perancangan *user interface* integrasi metode ANN dan BPSO untuk optimasi dosis pupuk pada tanaman palawija terdiri dari 3 halaman. Halaman tersebut terdiri dari proses pelatihan ANN, proses pengujian ANN dan optimasi metode BPSO. Proses rancang *user interface* ini akan dijelaskan lebih detail pada sub bab 3.5.1- sub bab 3.5.3

3.5.1 Proses Pelatihan ANN

Proses Pelatihan dilakukan agar setiap data latih berhasil mencapai target dan bobot dengan nilai MSE yang kecil. Berikut tampilan rancangan *interface* proses pelatihan untuk menjadi halaman *input*saat *user* ingin memasukan nilai parameter pelatihan terhadap data uji pada Gambar 3.18 Rancang *Interface* Proses Pelatihan Pengujian *backpropagation* ANN dengan keterangan :

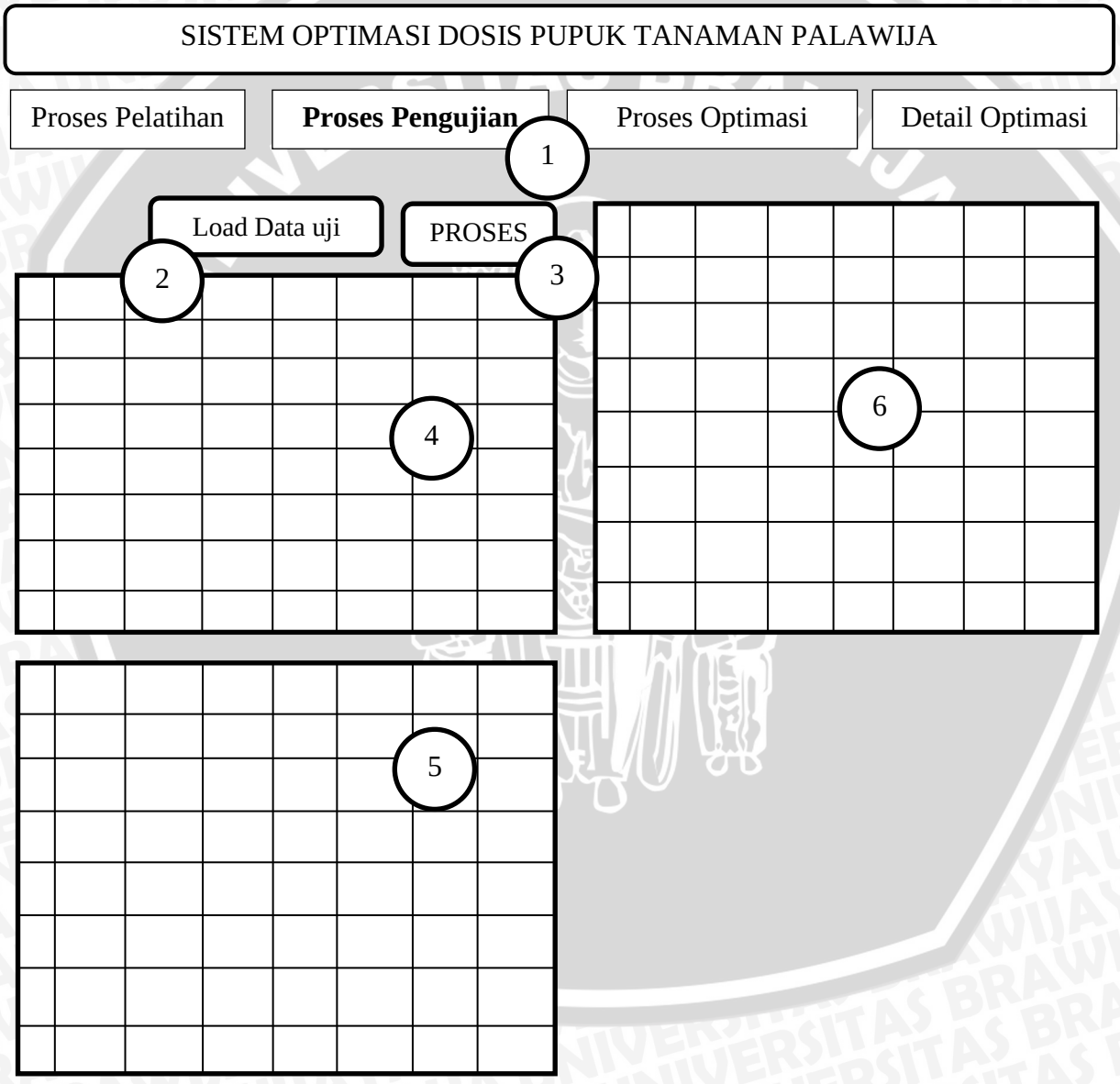
1. Judul Program
2. *Tab control* untuk melakukan proses pelatihan ANN *backpropagation*
3. *Button Load Data* untuk meload atau mengambil data latih yang diproses
4. *Datagridview* untuk menampilkan data latih
5. *Datagridview* untuk menmpilkan nilai minimal dan maksimal dari data latih
6. *Text box* untuk memasukkan nilai parameter proses pelatihan ANN. Nilai Parameter yang diisi pada proses pelatihan adalah :
   - Jumlah iterasi
   - Nilai *learning rate* (α)
   - Momentum
   - MSE, dan
   - Jumlah *hiddenlayer*
   - *Label* yang berisi *count iterasi*
7. *Datagridview* untuk menampilkan hasil normalisasi Data Latih
8. *Button Reset* untuk mengulang nilai yang dimasukan pada proses pelatihan
9. *Button Proses* untuk melakukan perhitungan proses pelatihan ANN
10. *Datagridview*untuk menampilkan hasil proses*feedforward* dari data latih
11. *Grid View* untuk menampilkan hasil MSE dari Proses *backpropagation*



Gambar 3.18 Rancang*Interface* Proses Pelatihan *Backpropagation* ANN

3.5.2 Proses Pengujian ANN

Setelah dilakukan proses pelatihan kemudian melakukan pengujian terhadap data uji. Hal ini dilakukan agar menghitung tingkat akurasi yang dihasilkan oleh proses pelatihan ANN. Halaman ini sebagai *input*bagi *user* ketika memberikan jumlah dosis pada masing-masing jenis pupuk, dan memberikan tampilan *output*dari pengaruh pada tanaman yang dihasilkan oleh pemberian dosis tersebut. Berikut adalah halaman *interface* untuk proses pengujian ANN :



Gambar 3.19 Rancang *Interface* Proses Pengujian ANN

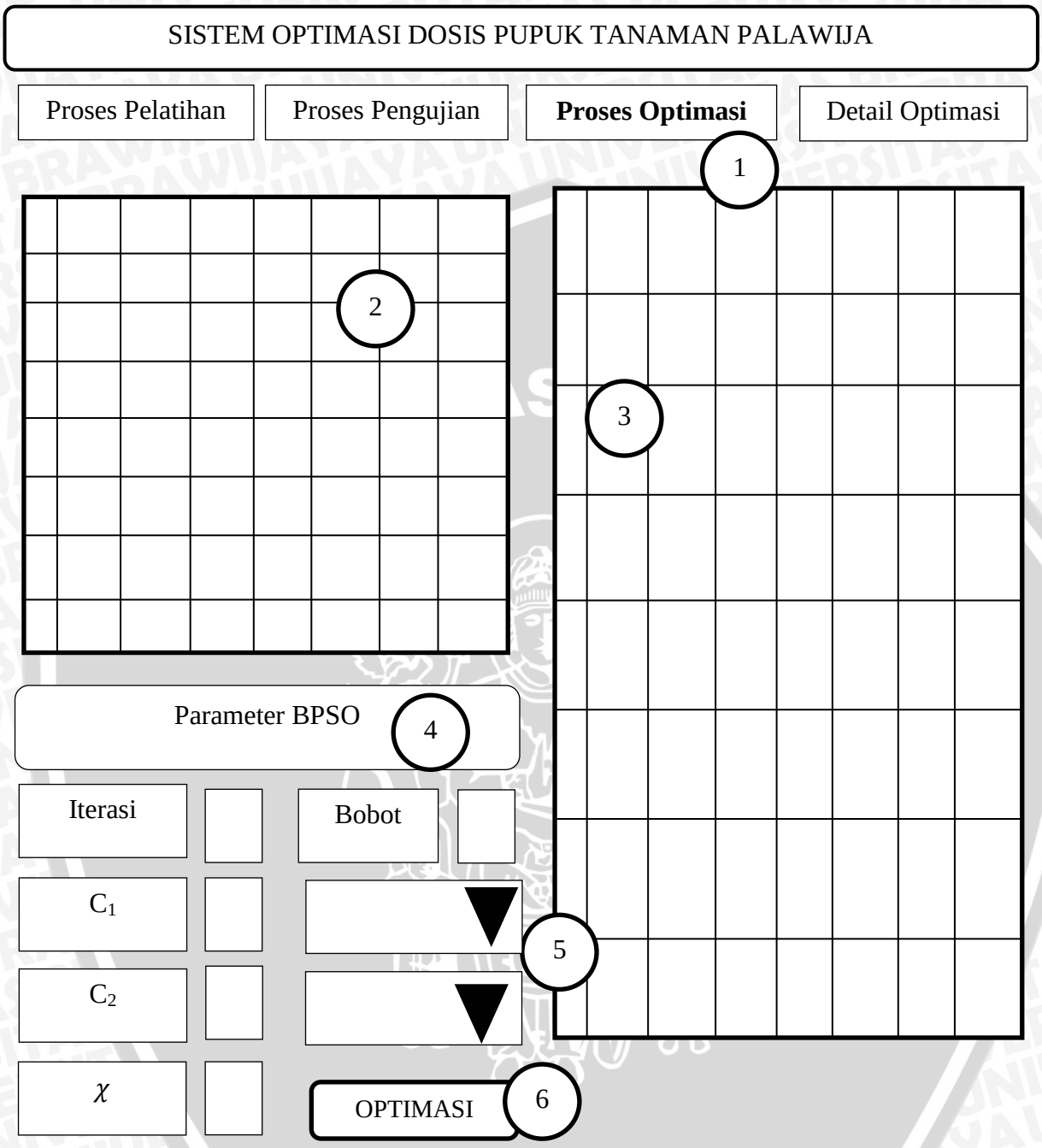
Berikut keterangan *interface* proses pengujian ANN *backpropagation* pada Gambar 3.19Rancang *Interface* Proses Pengujian ANN :

1. *Tab control* untuk melakukan proses pengujian ANN *backpropagation*
2. *Button Load Data* untuk meload atau mengambil data uji yang diproses
3. *Button Proses* untuk melakukan perhitungan proses pengujian ANN
4. *Datagridview* untuk menampilkan data uji
5. *Datagridview* untuk menampilkan hasil normalisasi data uji
6. *Datagridview* untuk menampilkan hasil proses feedfprward dari data uji

3.5.3 Proses Optimasi BPSO

Integrasi Metode ANN dan BPSO, optimasi dilakukan oleh metode BPSO. Pada halaman ini setelah mengetahui hasil *output*dari pemberian dosis pada jenis pupuk terhadap tanaman maka *user* akan memilih dua parameter pengaruh pada tanaman agar dioptimasi untuk menghasilkan nilai yang terbaik. Halaman ini akan menampilkan kombinasi dosis yang tepat untuk mendapatkan nilai optimal pada dua pengaruh pada tanaman yang dipilih. Berikut keterangan rancangan *interface* dari proses optimasi metode BPSO pada Gambar 3.20Rancangan *Interface* Porses Optimasi BPSO:

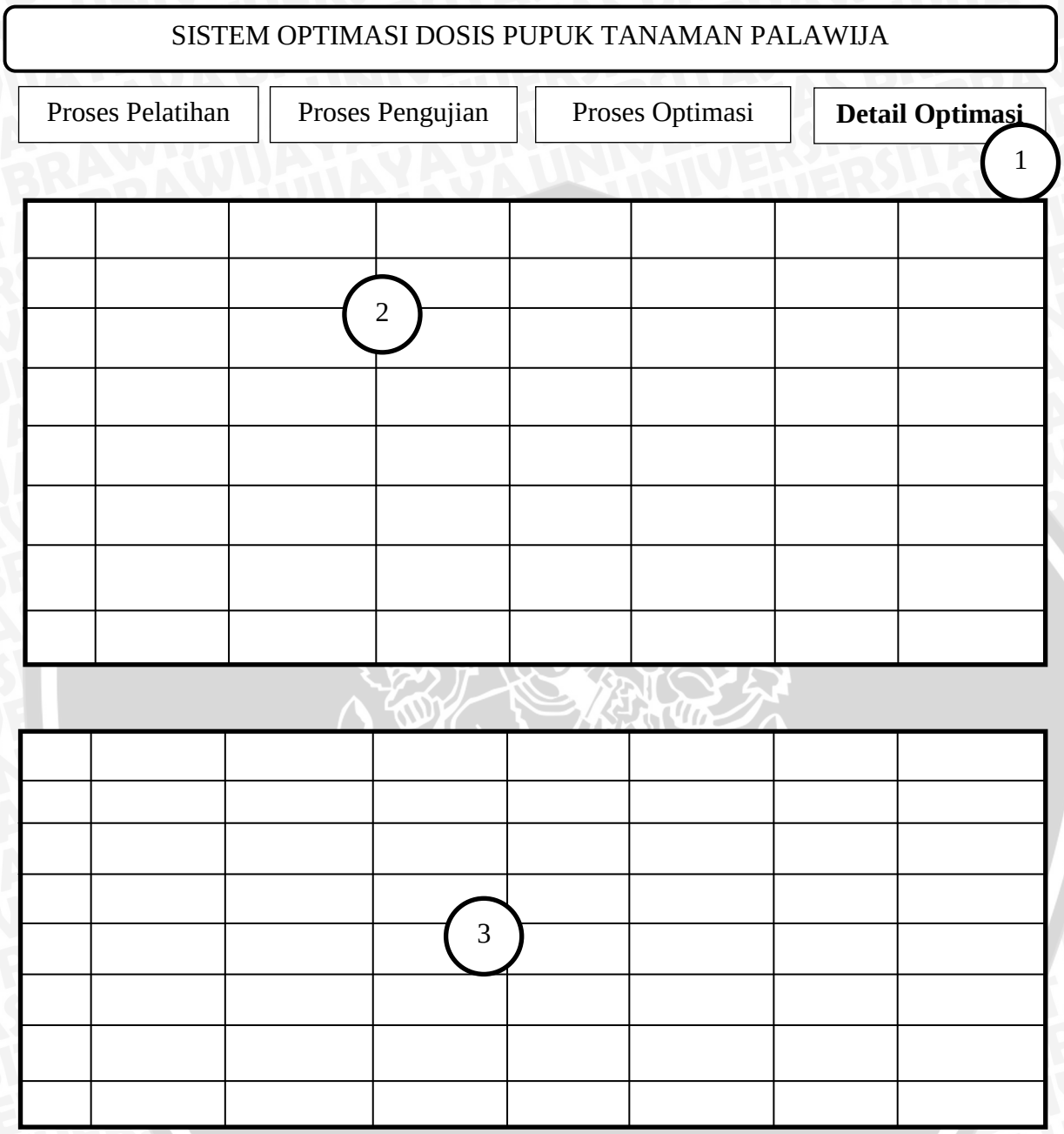
1. Judul Program
2. *Tab control* untuk melakukan proses Optimasi BPSO
3. *Datagridview* untuk menampilkan Populasi GBEST
4. *Datagridview* untuk menampilkan hasil Populasi Partikel
5. *Text box* untuk memasukkan nilai parameter Proses Optimasi BPSO. Nilai parameter BPSO yang di *input*adalah :
   - Jumlah Iterasi
   - Nilai C1
   - Nilai c2
   - Nilai bobot dan
   - *Constriction factor (χ)*
6. *Combo box* untuk memilih parameter pengaruh pada tanaman
7. *Button Optimasi*untuk melakukan perhitungan proses Optimasi BPSO



Gambar 3.20 Rancangan *Interface* Porses Optimasi BPSO

3.5.4 Proses Detail Optimasi BPSO

Pada Implementasi ini akan menampilkan detail dari hasil akhir BPSO. Hasil yang akan ditampilkan adalah posisi akhir populasi partikel dan populasi akhir dari populasi GBEST setelah iterasi terakhir. Pada halaman ini kita bisa melihat perubahan posisi akhir yang dihasilkan untuk pemberian dosis terbaik.



Gambar 3.21 Rancangan *Interface* Detail BPSO

Berikut keterangan rancangan *interface* dari proses Detail optimasi BPSO pada Gambar 3.21Rancangan *Interface* Detail BPSO:

1. *Tab control* untuk melihat detail proses Optimasi BPSO
2. *Datagridview* untuk menampilkan Populasi Akhir Partikel
3. *Datagridview* untuk menampilkan hasil Populasi akhir GBEST

3.6 Perancangan Uji Coba dan Evaluasi

Pada sub bab ini hal yang akan dijelaskan adalah membuat rancangan uji coba dan evaluasi dari sistem yang dibuat. Pengujian dilakukan untuk mengetahui prediksi yang dihasilkan proses pengujian ANN dan hasil optimasi yang diberikan oleh BPSO.

3.6.1 Pengujian terhadap Variasi Data Latih dan Data Uji

Pada pengujian ini akan dihitung nilai MSE yang dihasilkan dari perbedaan banyaknya jumlah data latih dan data uji. Setiap data akan memiliki target masing-masing untuk dicapai pada sistem. Perbedaan jumlah data latih akan memberikan hasil yang berbeda untuk bobot yang akan digunakan oleh data uji. Jumlah keseluruhan data yang akan dijadikan data uji dan data latih adalah 180 data. Pada pengujian, jika data latih 90% dan data uji 10% maka masing-masing jumlah 162 data latih dan 18 data uji. Berikut rancangan pengujian data latih dan data uji pada Tabel 3.56 Rancang Pengujian Variasi Data Latih dan Data Uji:

Tabel 3. 56 Rancang Pengujian Variasi Data Latih dan Data Uji

Perbandingan		Nilai MSE percobaan ke-*i*					Rata-rata MSE
Data Latih	Data Uji	1	2	3	4	5	
90%	10%						
80%	20%						
70%	30%						
60%	40%						
50%	50%						
40%	60%						
30%	70%						
20%	80%						
10%	90%						

3.6.2 Pengujian Jumlah Iterasi

Pengujian jumlah iterasi dilakukan dengan memberikan jumlah iterrasi yang berbeda, setelah didapatkan variasi data uji dan data latih yang terbaik. Dengan jumlah variasi data yang sama tetapi perbedaan iterasi akan diuji pada jumlah iterasi ke berapa, sistem menghasilkan bobot terbaik. Berikut adalah proses

pengujian terhadap variasi banyakanya jumlah iterasi, sesuai skenario pengujian pada Tabel 3.57 Rancangan Pengujian Jumlah Iterasi :

Tabel 3.57 Rancang Pengujian Jumlah Iterasi

Jumlah Iterasi	Nilai MSE percobaan ke-*i*					Rata-rata MSE
	1	2	3	4	5	
100						
200						
300						
400						
500						
600						
700						
800						
900						
1000						

3.6.3 Pengujian Variasi Jumlah *HiddenLayer*

Pengujian variasi jumlah *hiddenlayer* untuk mengenali target dan masukan. Semakin banyak jumlah *hiddenlayer* semakin akurat sistem dalam menentukan target pada setiap data latih, hanya waktu yang dibutuhkan semakin lama dalam mengenali setiap target. Berikut adalah proses pengujian pada Tabel 3.59 PengujianVariasi Jumlah *HiddenLayer* :

Tabel 3.58 Pengujian Variasi Jumlah *HiddenLayer*

Jumlah *Hiddenlayer*	Nilai MSE percobaan ke-*i*					Rata-rata MSE
	1	2	3	4	5	
1						
2						
3						
4						
5						
6						
7						
8						
9						
10						

3.6.4Pengujian Variasi Nilai *Learning Rate* dan Momentum

Proses pengujian variasi nilai *learning rate* dan momentum, dilakukan untuk mengetahui pengaruh pada MSE yang dihasilkan. Nilai *learning rate* berkisar antar 0.1 samapai 0.9 sedangkan untuk nilai momentum berkisar antara 0.5 sampai 0.9. Pada pengujian ini akan dipilih nilai *learning rate* dan momentum yang menghasilkan nilai MSE terendah. Proses ini dilakukan untuk mendapatkan nilai yang mendekati target sehingga hasil pengujian akan lebih baik. Berikut adalah proses rancangan pengujian yang akan dilakukan terhadap nilai *learning rate* dan momentum. Proses skenario perancangan ini dijelaskan pada Tabel 3.60 Pengujian Variasi Nilai *Learning Rate* dan momentum :

Tabel 3.59 Pengujian Variasi Nilai *Learning Rate* dan Momentum

Kombinasi		Nilai MSE percobaan ke-i					Rata-rata MSE
Learning rate	momentum	1	2	3	4	5	
0.9	0,5						
0.8	0,5						
0.7	0,6						
0.6	0,6						
0.5	0,7						
0.4	0,7						
0.3	0,8						
0.2	0,8						
0.1	0.9						
0.05	0.9						

BAB IV
IMPLEMENTASI PERANGKAT LUNAK

Pada proses implementasi ini membahas penerapan sistem dari proses perancangan yang telah dijelaskan pada bab sebelumnya. Implementasi diperlukan untuk mengetahui apakah sistem yang dibangun sudah sesuai dan mencapai tujuan yang diharapkan. Bab implementasi ini terdiri dari lingkungan implementasi, implementasi program, dan implementasi antar muka.

4.1 Lingkungan Implementasi

Untuk melakukan implementasi perangkat lunak, terlebih dahulu perlu disiapkan lingkungan implementasi untuk memenuhi kebutuhan sistem. Lingkungan implementasi ini menerapkan algoritma ANN *backpropagation* dan BPSO. Dalam membangun sistem ini, kebutuhan perangkat keras dan perangkat lunak yang digunakan adalah :

4.1.1 Lingkungan Perangkat Keras

Perangkat keras yang digunakan dalam dalam penelitian ini memiliki spesifikasi sebagai berikut :

1. *Processor Intel(R) Core(TM)5 Five CPU 2430M2,40GHz.*
2. *Memory* RAM 4 GB.
3. *Hardisk*500 GB.
4. Monitor 14 inch.
5. *Keyboard*

4.1.2 Lingkungan Perangkat Lunak

Perangkat lunak yang digunakan dalam pengembangan perangkat lunak dalam penelitian ini adalah :

1. Sistem Operasi Windows 7 Home Premium64-bit.
2. Microsoft Visual Studio Ultimate C# 2010 digunakan untuk membangun aplikasi optimasi dosis pupuk pada tanaman palawija.

3. Microsoft Office Excel 2010 digunakan untuk melakukan proses manualisasi perhitungan pada metode ANN *backpropagation* dan BPSO.
4. Microsoft Office Word 2010 digunakan untuk menyusun laporan tugas akhir.
5. Microsoft Visual Studio 2010 digunakan untuk membuat diagram alir proses metode ANN *backpropagation* dan BPSO.

4.2 Batasan Implementasi

Implementasi metode *Artificial Neural Networl*(ANN) *backpropagatioan* dan Algoritma *Bidirectional Particle Swarm Optimization* (BPSO) untuk optimasi dosis pupuk tanaman palawija ini memiliki batasan-batasan sebagai berikut :

1. Tanaman Palawija yang diamati adalah tanaman Jagung.
2. Data yang digunakan sebagai data latih dan data uji berdasarakan hasil penelitian sebelumnya oleh pakar Dr. Ir Widowati.
3. Jenis pupuk yang digunakan adalah pupuk Urea, KCL, SP_{36}, dan Biochar.
4. Hasil yang akan dioptimasi terhadap pengaruh pada tanaman adalah Bobot Kering Tanaman, Bobot 1000 Butir, Diameter Tongkol, Panjang Tongkol dan Hasil Produksi.

4.3Implementasi Program

Berdasarkan perancangan sistem yang telah dibuat pada bab 3, pada sub bab ini akan dibahas mengenai implementasi program. Sistem diimplementasikan pada kelas utama yang menampung seluruh proses. Tiap-tiap proses dilakukan oleh masing-masing fungsi seperti pada diagram alir pada sub bab 3.2.

4.3.1 Proses Normalisasi

Fungsi ini untuk menormalisasikan data uji dan latih agar diperoleh interval antara 0.1 sampai 0.9. Langkah yang dilakukan adalah mencari nilai minimal dan maksimal dinisialkan pada fungsi `min` dan `min_max`. Hasil normalisasi akan disimpan dalam variabel `tabel_normal`, sesuai Gambar 4.1*Source Code*Normalisasi.

```
1   //fungsi mencari nilai min dan max
2   publicvoid min(double[,] tbl, int row, int col)
3     {
4       for (int a = 0; a < col; a++)
5         {
6           max_min[a, 0] = tbl[0, a];
7           max_min[a, 1] = tbl[0, a];
8
9   for (int b = 0; b < row; b++)
10              {
11  If (tbl[b, a] <= max_min[a, 0]) max_min[a, 0] = tbl[b, a];
12  elseif(tbl[b, a]>= max_min[a,1]) max_min[a, 1]=tbl[b, a];
13  }
14        }
15  }
16
17  //fungsi normalisasi
18  publicvoid normalisasi(double[,] tabelnya, int barisnya,int kolomnya)
19  {
20  min(tabelnya, barisnya, kolomnya);
21  for (int a = 0; a < barisnya; a++)
22  {
23  for (int b = 0; b < kolomnya; b++)
24  {
25  if (max_min[b, 1] < 1) tabel_normal[a, b] = tabel[a, b];
26  else
27  {
28  //proses hitung hasil normalisasi setiap baris
29  tabel_normal[a, b] = ((0.8 *(tabel[a,b]-max_min[b, 0])
30  /(max_min[b, 1] - max_min[b, 0])) + 0.1);
31  }
32  }
33    }
34
35    }
```

Gambar 4.1*Source Code* Proses Normalisasi

Penjelasan Gambar 4.1 sebagai berikut:

1. Baris 11-12 adalah proses untuk mencari nilai minimal dan maksimal dari data yang di *input*.
2. Baris 29-30 adalah proses untuk melakukan normalisasi data dan pengulangan sebanyak jumlah baris dan kolom data.

4.3.2 Proses Pelatihan ANN *Backpropagation*

Pada sub bab ini akan membahas implementasi proses pelatihan menggunakan *artificial neural network* (ANN) *backpropagation*, untuk mencapai target pengaruh pada tanaman. Proses yang dilakukan dalam implementasi proses pelatihan adalah proses *feedforward*, proses *backpropagation* dan proses perbaikan bobot.

4.3.2.1 Proses *Feedforward*

Pada implementasi ini meneruskan sinyal masukan ke *hidden*layerdengan menggunakan fungsi aktivasi *sigmoid biner*. Langkah pertama adalah membangkitkan bobot dari *input*ke *hidden*dan *hidden*ke *output*. Fungsi membangkitkan bobot ini dijelaskan pada fungsi `bobot`. Kemudian meneruskan bobot ini untuk menjumlahkan sinyal masukan pada *input*dan *output*. Proses ini dijelaskan pada fungsi `outputIH` dan `outputIH2`. Selanjutnya, menghitung fungsi aktivasi yang diterapkan pada masing-masing sinyal *input*dan *output* menggunakan fungsi `fungsiSigmoid`. Proses terakhir kemudian menampilkan hasil *feedforward* sesuai dengan jumlah *hidden*yang ditentukan oleh *user*, pada fungsi `feedforward`.

```
1   // fungsi membangkitkan bobot untuk input ke hidden dan hidden ke output
2   publicdouble[,] bobot(int inialIterasi, intinput0, intinput_output0,
3   int   datanya,   int   bts_min,   int   bts_max,   double[,]   theBobot,
4   double[,]theUbah)
5   {
6   bobotnya = newdouble[input0, input_output0];
7   //iterasi ke-0 data ke-0 bobot input ke hidden dibangkitkan secara acak
8   if (inialIterasi == 0)
9   {
10  if (datanya == 0)
11  {
12  for (int a = 0; a <input0; a++)
13  {
14  for (int b = 0; b <input_output0; b++)
15  {
16  bobotnya[a,b]=randm.NextDouble()*(bts_max-(bts_min))
17  +(bts_min);
18  }
19  }
20          }
21  //untuk iterasi dan data tidak ke-0 bobot input ke hidden adalah bobot
22  data sebelumnya ditambah perubahan bobot
23  else{
24  for (int a = 0; a <input0; a++)
25  {
26  for (int b = 0; b <input_output0; b++)
27  {
28  bobotnya[a, b] = theBobot[a, b] + theUbah[a, b];
29  }
30  }
31  }
32  }
33  else
34  {
35  // untuk iterasi ke-0 data ke-0 bobot hidden ke output dibangkitkan
36  secara acak
37  if (datanya == 0)
38  {
39  for (int a = 0; a <input0; a++)
40  {
41  for (int b = 0; b <input_output0; b++)
```

```
{
bobotnya[a, b] = randm.NextDouble()*
(bts_max - (bts_min)) + (bts_min);
}
}
// untuk iterasi tidak ke-0 dan data ke-0 bobot input ke hidden dan
hidden ke output adalah bobot yang sama pada hasil akhir bobot pada
iterasi sebelumnya
bobotnya = theBobot;
}
else
//untuk iterasi dan data tidak ke-0 bobot hidden ke output adalah bobot
data sebelumnya ditambah perubahan bobot
{
for (int a = 0; a <input0; a++)
{
for (int b = 0; b <input_output0; b++)
{
bobotnya[a, b] = theBobot[a, b] + theUbah[a, b];
}
}
}
}
return bobotnya;
}
//fungsi menjumlahkan bobot sinyal input
publicdouble[] outputIH(double [,] bobot_Input_Hidden, intinput1,
intinput_output1, int datanya, double [,] tabelnya)
{
output_InHidden = newdouble[input_output1];
for (int a = 0; a <input1; a++)
{
double hasil = 0.0;
for (int b = 0; b <input_output1; b++)
        hasil = (bobot_Input_Hidden[a, b]*tabelnya[datanya,b]) + hasil;
output_InHidden[a] = hasil;
}
returnoutput_InHidden;
}
//fungsi menjumlahkan bobot sinyal output
publicdouble[]    outputIH2(double[,]    bobot_Input_Hidden,    intinput2,
intinput_output2, int datanya, double[] tabelnya)
{
output_InHidden = newdouble[input_output2];
for (int a = 0; a <input_output2; a++)
{
double hasil = 0.0;
for (int b = 0; b <input2; b++)
        hasil = (bobot_Input_Hidden[b, a] * tabelnya[b]) + hasil;
output_InHidden[a] = hasil;
}
returnoutput_InHidden;
}
// menghitung fungsi aktivasi sigmoid biner untuk sinyal input dan output
publicdouble[] fungsiSigmoid(double[] outputnya, int jumlah)
{
fungsi_sigmoid = newdouble[jumlah];
for (int a = 0; a < jumlah; a++)
{
fungsi_sigmoid[a] = 1 / (1 + Math.Pow(2.71,outputnya[a]));
 }
return fungsi_sigmoid;
}
//fungsi memanggil hasil yang dibutuhkan pada feedforward
```

```
106 publicvoid feedforward(int it, int row)
107 {
108 output_ih = newdouble[hidden]; output_ho = newdouble[output];
109     sigmoid = newdouble[hidden]; sigmoid_ho = newdouble[output];
110 {
111  bobot_ih=bobot(it,input,hidden,row,-1,1,bobot_ih,ubah_bobotHI);
112 output_ih = outputIH(bobot_ih, input, hidden, row, tabel_normal);
113 sigmoid = fungsiSigmoid(output_ih, hidden);
114 }
115
116 {
117 bobot_ho=bobot(it,hidden,output, row, 0,1, bobot_ho, ubah_bobotOH);
118 output_ho = outputIH2(bobot_ho, hidden, output, row, sigmoid); //
119 sigmoid_ho = fungsiSigmoid(output_ho, output);
120 }
121 }
```

Gambar 4.2*Source Code* Proses *Feedforward*

Penjelasan Gambar 4.2sebagai berikut:

1. Baris 16-17 adalah proses untuk iterasi pertama dan data ke-0 untuk bobot *input* ke *hidden* dibangkitkan secara acak.
2. Baris 28 adalah proses iterasi selanjutnya dan data tidak ke-0 bobot dari *input* ke *hidden* adalah bobot sebelumnya ditambah dengan perubahan bobot.
3. Baris 43-44 proses untuk iterasi pertama dan data ke-0 untuk bobot *hidden* ke *output* dibangkitkan secara acak.
4. Baris 50 adalah proses untuk iterasi selanjutnya dan data ke-0 bobot untuk *input* ke *hidden* dan *hidden* ke *output* sama dengan bobot sebelumnya.
5. Baris 60 adalah proses iterasi selanjutnya dan data tidak ke-0 bobot dari *hidden* ke *output* adalah bobot sebelumnya ditambah dengan perubahan bobot.
6. Baris 76-77 adalah proses untuk menjumlahkan bobot sinyal *input*.
7. Baris 90-91 adalah proses untuk menjumlahkan bobot sinyal *output*.
8. Baris 101 adalah proses untuk menghitung fungsi aktivasi
9. Baris 106-121 adalah proses memanggil nilai-nilai yang dibutuhkan untuk *feedforward.*

4.3.2.2 Proses *Backpropagation*

Pada proses ini akan dihitung hasil dari *outputlayer* yang kemudian akan memeriksa semua kesalahan ke *hiddenlayer*. Pada proses ini akan diperiksa kesalahan yang dihasilkan oleh *output* terhadap target *output* sebenarnya. Langkah pertama adalah menghitung informasi *error*dengan fungsi `fungsi_faktor`. Selanjutnya memasukkan nilai yang dibutuhkan dan ditampilkan pada proses

backpropagation dengan fungsi `backpropagation`. Kemudian menjumlahkan delta *inputlayer* dengan fungsi `sum_error`, menghitung perubahan bobot pada *hiddenlayer* ke *input*dan *output* dengan fungsi `ubahBobot`. Proses terakhir yang penting adalah menghitung *error* dan MSE dengan fungsi `hitung_mse`.

```
1   //fungsi menghitung informasi errorhidden
2   publicdouble[] fungsi_faktor(double[] tabelnya, double[] sigmo_akhir,
3   int pjg)
4   {
5   faktor_Onya = newdouble[pjg];
6   for (int a = 0; a < pjg; a++);
7   {
8   if(pjg!=hidden)
9   //menghitung informasi errorhidden ke outputlayer
10  faktor_Onya[a]=((tabelnya[a]-sigmo_akhir[a])*(1-sigmo_akhir[a])
11          *sigmo_akhir[a]);
12  else
13  //menghitung informasi errorinput ke hiddenlayer
14              faktor_Onya[a]=(tabelnya[a]*sigmo_akhir[a]*(1-sigmo_akhir[a]));
15  }
16  return faktor_Onya;
17  }
18  //fungsi memanggil nilai yang dibutuhkan dalam proses backpropagation
19  publicvoid backpropagation(int row2, double[,] tbl_normal)
20    {
21  ubah_bobotOH = newdouble[hidden, output];
22  sumErrorH = newdouble[hidden];
23  ubah_bobotHI = newdouble[input, hidden];
24  faktorErrorO = fungsi_faktor(tbl_normal_output, sigmoid_ho, output);
25  ubah_bobotOH = ubahBobot(hidden, output, faktorErrorO, sigmoid);
26  sumErrorH = sum_error(hidden, output, faktorErrorO, bobot_ho);
27  faktorErrorH = fungsi_faktor(sumErrorH, sigmoid, hidden);
28  ubah_bobotHI=ubahBobot(input,hidden, tbl_normal_input,faktorErrorH);
29  }
30  //fungsi menjumlahkan delta inputlayer
31  publicdouble[] sum_error(int masukn, int keluarn, double[] faktor_e,
32  double[,] bobot_outnya)
33
34  {
35  sumErrornya = newdouble[masukn];
36  for (int a = 0; a < masukn; a++)
37  {
38  double hasil=0.0;
39  for (int b = 0; b < keluarn; b++)
40  {
41  hasil = (faktor_e[b] * bobot_outnya[a, b])+hasil;
42  }
43  sumErrornya[a] = hasil;
44  }
45  return sumErrornya;
46  }
47  // fungsi menghitung perubahan bobot pada hiddenlayer ke input dan output
48  publicdouble[,] ubahBobot(int masukan, int keluaran, double[] faktornya,
49  double[] sigmoid_masukan)
50  {
51  ubah_bobotnya = newdouble[masukan, keluaran];
52  for (int a = 0; a < masukan; a++)
53  {
54  for (int b = 0; b < keluaran; b++)
55  {
```

```
56  ubah_bobotnya[a, b] = (learning * faktornya[b] *
57  sigmoid_masukan[a]) + (moment * learning * faktornya[b] *
58  sigmoid_masukan[a]);
59  }
60  }
61  return ubah_bobotnya;
62  }
63  // fungsi menghitung error dan MSE
64  publicdouble hitung_mse(double[] tblNormal_O, double[]sigmoid1)
65  {
66  double hasilE = 0;
67  for (int a = 0; a <output; a++)
68  {
69  hasilE = (tblNormal_O[a] - sigmoid1[a])+hasilE;
70  }
71  sementara = (hasilE/output);
72  return sementara;
73  }
74
```

Gambar 4.3*Source Code* Proses *Backpropagation*

Penjelasan Gambar 4.3 sebagai berikut:

1. Baris 10-11 adalah proses menghitung informasi *error* dari *hidden* ke *output layer*
2. Baris 14 adalah proses menghitung informasi *error* dari *input* ke *hidden layer*
3. Baris 19-29proses memanggil nilai-nilai yang dibutuhkan dalam *backpropagation.*
4. Baris 41 adalah proses menjumlahkan delta *input layer*.
5. Baris 57-59 adalah proses adalah proses menghitung perubahan bobot pada *hidden layer* ke *input* dan *output*.
6. Baris 65-74 adalah proses untuk menghitung *error* dana MSE.

4.3.3 Proses Pengujian ANN *Backpropagation*

Pada proses pengujian ANN *Backpropagation* hanya terdapat satu proses yaitu *feedforward*, langkah-langkah *feedforward* ini sama dengan proses pelatihan ANN. Nilai bobot yang diberikan pada pengujian adalah bobot terakhir yang didapat pada proses pelatihan. Sehingga, proses pengujian hanya memanggil nilai dari proses *feedforward* pada pelatihan. Fungsi untuk memanggil nilai ini dijelaskan pada `pengujian_ann`.

```
1   //fungsi memanggil nilai yang dibutuhkan untuk pengujian backpropagation
2   publicvoid pengujian_ann(int baris_input, int kolom_input, double[,]
3   tabel_input, double[,] bobot_ihnya, double[,] bobot_honya, int inp, int
4   outp, int hidd, int trigger)
5   {
6   tabel = tabel_input; baris  = baris_input; kolom = kolom_input;
7   bobot_ih = bobot_ihnya; bobot_ho = bobot_honya; input = inp;
8   output = outp; hidden = hidd;
9   tbl_input = newdouble[baris, kolom - output];
10  tbl_output = newdouble[baris, kolom - input];
11  tbl_denorm = newdouble[baris, kolom - input];
12  bobot_hoTampil = newdouble[baris, hidden, output];
13  bobot_ihTampil = newdouble[baris, input, hidden];
14  output_hoTampil = newdouble[baris, output];
15  output_ihTampil = newdouble[baris, hidden];
16  sigmoid_hoTampil = newdouble[baris, output];
17  sigmoid_ihTampil = newdouble[baris, hidden];
18  for (int b = 0; b < baris; b++)//baris=12  (baris)
19  {
20  for (int c = 0; c < kolom; c++)
21  {
22  if (c < 4) tbl_input[b,c] = tabel[b, c];
23  else  tbl_output[b,c - 4] = tabel[b, c];
24  }
25  }
26  normalisasi(tbl_input, baris, kolom-output,0);
27  for (int b = 0; b < baris; b++)
28  {
29  feedforward(0, b, 1);
30  denormalisasi(tbl_output, baris, kolom-4, b, trigger);
31  save_tampil(b, kolom-input);
32        }
33    }
```

Gambar 4.4*Source Code* Proses Pengujian ANN *Backpropagation*

Penjelasan Gambar 4.4 sebagai berikut:

1. Baris 6-18 adalah proses untuk deklarasi dan memanggil nilai-nilai yang dibutuhkan untuk pengujian ANN *backpropagation*
2. Baris 26 adalah prose normalisasi dilakukan pengulangan sebanyak jumlah data latih.
3. Baris 29 adalah proses memanggil fungsi *feedforward* karena pengujian ANN backpropagation hanya smapai tahap *feedforward*.
4. Baris 30 adalah proses memanggil fungsi denormalisasi.

4.3.4 Proses Denormalisasi

Pada proses denormalisasi ini, hasil *output*dari data uji akan diubah ke range nilai *real* atau sebenarnya. Proses denormalisasi juga digunakan pada BPSO. Pada BPSO niali yang telah didenormalisasi di perlukan untuk menghitung nilai vektor pada populasi partikel dan GBEST. Proses menghitung nilai vektor ini akan

dijelaskan pada sub bab 4.3.5.1Proses Denormalisasi ini dijelaskan pada fungsi `denormalisasi` sesuai dengan Gambar 4.5:

```
1   //fungsi proses denormalisasi
2   publicvoid denormalisasi(double[,]tblnya, int barisnya, int kolomnya,
3   int data, int trigger)
4     {
5       denorm = newdouble[kolomnya];
6   if (trigger == 0)
7         {
8           min(tblnya, barisnya, kolomnya);
9   //denormalisasi untuk data uji backpropagation
10  for (int b = 0; b < kolomnya; b++) denorm[b] =(((max_min[b, 1]
11          - max_min[b,0])*(sigmoid_ho[b] - 0.1)) / 0.8) + max_min[b, 0];
12        }
13  //denormalisasi untuk populasi partikel dan GBEST pada BPSO
14  elseif (trigger==1)
15              {
16  Double[,] tbl_tumbuh = newdouble[5, 2];
17                tbl_tumbuh[0, 0] = 4;   tbl_tumbuh[0, 1] = 5;
18                tbl_tumbuh[1, 0] = 230; tbl_tumbuh[1, 1] = 265;
19                tbl_tumbuh[2, 0] = 3;   tbl_tumbuh[2, 1] = 5;
20                tbl_tumbuh[3, 0] = 13;  tbl_tumbuh[3, 1] = 20;
21                tbl_tumbuh[4, 0] = 6;   tbl_tumbuh[4, 1] = 8;
22  for (int b = 0; b < kolomnya; b++)
23                  denorm[b] = (((tbl_tumbuh[b, 1] - tbl_tumbuh[b, 0]) *
24                  (sigmoid_ho[b] - 0.1)) / 0.8) + tbl_tumbuh[b, 0]; ;
25              }
26    }
```

Gambar 4.5*Source Code* Proses Denormalisasi

Penjelasan Gambar 4.5*Source Code* Proses Denormalisasi sebagai berikut:

1. Baris 17-21 adalah deklarasi nilai minimal dan maksimal untuk *output* pengaruh pada tanaman.
2. Baris 23-34 adalah proses untuk menghitung denormalisasi dengan pengulangan sebanyak jumlah *output*.

4.3.5 Proses Optimasi BPSO (*Bidirectional Particle Swarm Optimization*)

Proses Optimasi digunakan algoritma BPSO, algoritma ini akan melakukan optimasi sesuai dengan dua pengaruh pada tanaman yang dipilih sebagai vektor oleh *user*. Pada proses ini , kumpulan GBEST adalah solusi terbaik dalam pemberian dosis pupuk untuk mengoptimasi dua pengaruh tanaman. Berikut adalah langkah-langkah implementasi pada proses optimasi BPSO :

4.3.5.1 Proses Membangkitkan Populasi Partikel dan GBEST

Langkah pertama dalam proses optimasi BPSO adalah membangkitkan populasi partikel dan GBEST. Sesuai dengan standard PSO (*Particle Swarm*

Optimization) jumlah populasi partikel yang dibangkitkan adalah 30 partikel dan populasi GBEST 10 partikel. Partikel yang dibangkitkan memiliki rentang nilai dan kecepatan pada fungsi `range`. Kemudian proses membangkitkan partikel dijelaskan pada fungsi `random_partikel`, partikel dibangkitkan secara acak sesuai dengan rentang nilai yang telah dideklarasi pada `range`. Fungsi `hitung_aktivasi`adalah proses untuk menghitung vektor pada masing-masing populasi partikel dan GBEST.

```
1   //fungsi deklarasi batas range dosis pupuk dan kecepatan
 2  publicvoid range()
 3  {
 4  range_dosis = newdouble[kandungan, 2];
 5      range_dosis[0, 0] = 0.177;  range_dosis[0, 1] = 0.533;//urea
 6      range_dosis[1, 0] = 0.0138; range_dosis[1, 1] = 0.319;//sp
 7      range_dosis[2, 0] = 0.04;   range_dosis[2, 1] = 0.3;  //kcl
 8      range_dosis[3, 0] = 10;     range_dosis[3, 1] = 50;   //ba
 9      range_v = newdouble[kandungan, 2];
10  for (int a = 0; a < kandungan; a++)
11  {
12  double v = 0;
13            v = 0.6 * range_dosis[a, 1];
14            range_v[a, 0] = -v;
15            range_v[a, 1] = v;
16  }
17  }
18  //fungsi membangkitkan populasi untuk partikel dan GBEST
19  publicvoid random_partikel()
20  {
21  //populasi partikel 30 dan GBEST 10
22  partikel = 30;
23      mem_Gbest= 10;
24  //untuk populasi awal partikel dan GBEST
25  po_partikel=newdouble[partikel,kandungan];
26      kcptn_partikel = newdouble[partikel, kandungan];
27      po_gbest = newdouble[mem_Gbest, kandungan];
28  //untuk populasi selanjutnya sesuai proses perubahan partikel dan GBEST
29      po_partikel = randomnya(partikel, kandungan, range_dosis);
30      kcptn_partikel = randomnya(partikel, kandungan, range_v);
31      po_gbest = randomnya(mem_Gbest, kandungan, range_dosis);
32    }
33  //fungsi menghitung vektor partikel dan GBEST
34  publicdouble[,] hitung_aktivasi(double[,]tbl)
35    {
36  for (int a = 0; a < tbl.GetLength(0); a++)
37        {
38          tbl[a, 0] = 1/tbl[a, 0];
39          tbl[a, 1] = 1/tbl[a, 1];
40        }
41  return tbl;
42    }
```

Gambar 4.6*Source Code*Membangkitkan Partikel dan GBEST

Penjelasan Gambar 4.6sebagai berikut:

1. Baris 5-8 adalah proses deklarasi nilai minimal dan maksimal untuk pemberian dosis pupuk pada palawija.
2. Baris 12-15 adalah proses untuk menghitung rentang kecepatan pada partikel.
3. Baris 22-23 adalah proses membangkitkan jumlah populasi partikel dan GBEST untuk awal iterasi.
4. Baris 38-39 adalah proses untuk menghitung vektor pada populasi partikel dan GBEST.

4.3.5.2 Proses Memilih GBEST dan PBEST *child*$_1$

Dalam mencari GBEST dan PBEST untuk child$_1$ langkah pertama adalah menghitung *euclidean distance* pada masing-masing anggota GBEST. Proses ini dijelaskan pada fungsi `jarak`, setelah hitung mencari radius dari setiap populasi GBEST dengan fungsi `radius`. Nilai radius yang dihasilkan digunakan pada pencarian *niche count*seperti pada proses `hitung_NC`. Nilai NC terkecil yang akan dipilih sebagai GBEST dan PBEST bagi child$_1$. Proses ini dijelaskan dalam fungsi `hitung_PG_Child1`.

```
1  //fungsi menghitung NC pada GBEST
2  publicdouble[] hitung_nc(int panjang, double nilai_ncRadius, double[,]
3  jaraknya)
4    {
5  double[] hasil = newdouble[panjang];
6  for (int a = 0; a < panjang; a++)
7        {
8  double sementara = 0;
9  for (int b = 0; b < panjang; b++)
10           {
11 if (jaraknya[a, b] <= nilai_ncRadius)
12               {
13                 sementara=(1-(jaraknya[a, b]/nilai_ncRadius)+ sementara;
14               }
15           }
16         hasil[a]=sementara-1;
17       }
18 return hasil;  }
19 //fungsi menghitung radius pada GBEST
20 publicdouble radius(double[,] tabl)
21   {
22 double hasilnya=0;
23 for (int a = 0; a < tabl.GetLength(0); a++)
24       {
25         hasilnya = Math.Pow(max_min[a, 1] - max_min[a, 0],2)+hasilnya;
26       }
27 double hasile = (0.5) * (Math.Sqrt(hasilnya));
28 return hasile;; }
29 //fungsi menghitung Euclidean distance child1 dan child2 pada GBEST
```

```
30  publicdouble[,] jarak(double[,] tbl1, double[,] tbl2)
31    {
32  double[,] result = newdouble[tbl1.GetLength(0), tbl1.GetLength(0)];
33  for (int a = 0; a < tbl1.GetLength(0); a++)
34        {
35  double hasil = 0;
36  for (int b = 0; b < tbl2.GetLength(0); b++)
37            {
38              hasil = 0;
39  for (int c = 0; c < tbl2.GetLength(1); c++)
40                {
41                  hasil = Math.Pow(tbl1[a,c]-tbl2[b,c],2)+hasil;
42                }
43              result[a, b] = Math.Sqrt(hasil);
44            }
45        }
46  return result;
47    }
48  //fungsi mengambil nilai vektor anggota GBEST yang terpilih
49  publicdouble[,] Take_Vektor_pilih(double[,] get_tbl_denorm_bpso)
50    {
51      tbl_denormBpso = newdouble[get_tbl_denorm_bpso.GetLength(0), 2];
52  for (int a = 0; a < get_tbl_denorm_bpso.GetLength(0); a++)
53        {
54  for (int b = 0; b < get_tbl_denorm_bpso.GetLength(1); b++)
55            {
56  if (pilih1_bpso == b) tbl_denormBpso[a, 0] =
57                get_tbl_denorm_bpso[a, b];
58  elseif (pilih2_bpso == b) tbl_denormBpso[a, 1] =
59                  get_tbl_denorm_bpso[a, b];
60              }
61        }
62  return tbl_denormBpso;
63    }
64  //proses memilih child1 dengan nilai NC terkecil
65  publicvoid hitung_PG_Child1(double[,]po_partikel, double[,] po_gbest,
66  double[,] Vektor1, int indx)
67    {
68  //proses memanggil nilai yang dibutuhkan menghitung NC
69  Ann ann_alg = newAnn();
70      jarak1 = newdouble[Vektor1.GetLength(0), Vektor1.GetLength(0)];
71      nc = newdouble[jarak1.GetLength(1)];
72      g_child1 = newdouble[po_partikel.GetLength(0),
73      po_partikel.GetLength(1)];
74      ann_alg.min(po_gbest, po_gbest.GetLength(0), po_gbest.GetLength(1));
75      max_min = ann_alg.GetMaxMin();
76      nilai_r = radius(max_min);
77  //rumus menghitung NC
78      nc_radius = nilai_r / (Math.Pow(5, 0.25));
79      jarak1 = jarak(po_gbest, po_gbest);
80      nc = hitung_nc(jarak1.GetLength(1), nc_radius, jarak1);
81  double min = nc[0]; int detect = 0;
82  for (int c = 0; c < nc.GetLength(0); c++)
83          {
84  if (min > nc[c])
85              {
86                min = nc[c]; detect = c;
87              }
88          }
89  for (int c = 0; c < po_gbest.GetLength(1); c++)
90        g_child1[indx,c]=po_gbest[detect,c];
91        pbest_child1 = g_child1;
92    }
```

Gambar 4.7*Source Code* Proses Memilih GBEST dan PBEST *child*$_1$

Penjelasan Gambar 4.7sebagai berikut:

1. Baris 13-16 adalah proses menghitung nilai NC (*niche count).*
2. Baris 26-28 adalah proses untuk menghitung nilai *radius*.
3. Baris 43 adalah proses menghitung *Euclidean distance* untuk $child_1$.
4. Baris 51-64 adalah proses mengambil nilai vektor dari anggota GBEST yang terpilih.
5. Baris 88 adalah proses memilih nilai NC terkecil untuk dipilih sebagai anggota $child_1$.

4.3.5.3 Proses Memilih GBEST dan PBEST $child_2$

Proses memilih GBEST dan PBEST untuk $child_2$adalah menghitung *euclidean distance* antara populasi partikel dengan populasi GBEST. Kemudian nilai *euclidean distance* yang terkecil akan dipilih sebagai GBEST dan PBEST $child_2$. Proses ini dijelaskan pada fungsi`hitung_PG_Child`$_2$.

```
1  //proses memilih GBEST dan PBEST untuk child2
2  publicvoid hitung_PG_Child2(double[,] po_partikel, double[,] po_gbest,
3  double[,] v_awal, double[,] v_akhir, int indx)
4    {
5      hasil_jarak = newdouble [v_akhir.GetLength(0)];
6  g_child2=newdouble[po_partikel.GetLength(0),
7  po_partikel.GetLength(1)];
8  //proses menghitung Euclidean distance populasi GBEST dengan populasi
9  partikel
10 for (int a = 0; a < v_awal.GetLength(0); a++)
11       {
12         hasil_jarak[a]=Math.Sqrt((Math.Pow(v_awal[indx,0]-v_akhir
13 [a, 0],2))+ (Math.Pow(v_awal[indx, 1] - v_akhir[a, 1], 2)));
14       }
15 //nilai Euclidean terkecil dipilih sebagai child2
16 double min = hasil_jarak[0]; int detect = 0;
17 for (int c = 0; c < hasil_jarak.GetLength(0); c++)
18       {
19 if (min > hasil_jarak[c])
20           {
21             min = hasil_jarak[c]; detect = c;
22           }
23       }
24 for (int c = 0; c < po_partikel.GetLength(1); c++)
25     g_child2[indx, c] = po_gbest[detect, c];
26     pbest_child2 = g_child2;
27   }
```

Gambar 4.8*Source Code* Proses Memilih GBEST dan PBEST $child_2$

Penjelasan Gambar 4.8sebagai berikut:

1. Baris 12-13 adalah proses untuk menghitung *euclidean distance* populasi GBEST terhadap populasi partikel.
2. Baris 21 adalah memilih anggota GBEST dengan nilai *euclidean distance* terkecil sebagai partikel terpilih untuk *child*$_2$.

4.3.5.4 Proses Menghitung Kecepatan dan Posisi Baru *child*$_1$ dan *child*$_2$

Setelah didapatkan anggota untuk *child*$_1$ dan *child*$_2$, kemudian menghitung kecepatan baru dan posisi yang dihasilkan oleh *child*$_1$ dan *child*$_2$. Fungsi `perubahan_kcptn`akan mencari kecepatan baru yang dihasilkan untuk *child*$_1$ dan *child*$_2$. Kemudian menghitung posisi yang didapatkan dari kecepatan yang baru, proses ini terdapat pada fungsi `perubahan_posisi`.

```
1   //fungsi menghitung perubahan kecepatan
2   publicvoid perubahan_kcptn(double[,]tbl_ukuran_randm, int w, double c1,
3   double c2, double[,]partikel, double[,] pbest_partikel, double[,] kcptn,
4   int indx)
5   {
6       rndm_c1 = randome(0, 1, tbl_ukuran_randm,1);
7       rndm_c2 = randome(0, 1, tbl_ukuran_randm,2);
8   for (int a = 0; a < partikel.GetLength(1); a++)
9   {
10  //persamaan menghitung perubahan kecepatan child1
11  kcptn_c1[indx, a] = (w *kcptn[indx, a])+(c1 * rndm_c1[indx, 0] *
12  (pbest_partikel[indx, a] - partikel[indx, a])) + (c2 *
13  rndm_c1[indx, 1] * (pbest_child1[indx, a] - partikel[indx, a]));
14  //persamaan menghitung perubahan kecepatan child2
15  kcptn_c2[indx, a] =(w * kcptn[indx, a])+(c1 * rndm_c2[indx, 0] *
16  (pbest_partikel[indx, a] - partikel[indx, a])) + (c2 *
17  rndm_c2[indx, 1] * (pbest_child2[indx, a] - partikel[indx, a]));
18  }
19  }
20  //fungsi menghitung perubahan Posisi
21  publicvoid perubahan_posisi(double x, double[,] partikel,
22  double[,]gbest, double[,] vektor_gbest, int indx)
23  {
24  int[,] nilai = newint[4, 2]; int nilai2 = 0; int nilai1 = 0;
25  for (int a = 0; a < partikel.GetLength(1); a++)
26  {
27  //persamaan menghitung perubahan posisi child1
28  update_poC1[indx, a]=(partikel[indx,a] + (x *kcptn_c1[indx,a]));
29  //persamaan menghitung perubahan posisi child2
30          update_poC2[indx, a]=(partikel[indx,a]+(x * kcptn_c2[indx, a]));
31      }
```

Gambar 4.9*Source Code* Menghitung Kecepatan dan Posisi Baru

Penjelasan Gambar 4.9sebagai berikut:

1. Baris 6-7 adalah proses membangkitkan nilai secar acak untuk c_1 dan c_2.
2. Baris 11-13 adalah proses menghitung perubahan kecepatan baru untuk *child*$_1$.
3. Baris 15-17 adalah proses menghitung perubahan kecepatan baru untuk *child*$_2$.
4. Baris 28 adalah proses menghitung perubahan posisi untuk *child*$_1$.
5. Baris 30 adalah proses menghitung perubahan posisi unutuk *child*$_2$.

4.3.5.5 Proses Memperbaharui Posisi GBEST dan Partikel

Pada proses implementasi dari BPSO akan memperbaharui posisi yang terpilih sebagai GBEST dan Partikel. Implementasi memperbaharui posisi partikel dijelaskan pada fungsi `update_partikel`. Proses update partikel terdiri pada kondisi sesuai dengan sub bab aliran perancangan sistem 3.2.2.5, sedangkan update GBEST proses kondisi sesuai dengan sub bab aliran perancangan sistem 3.2.2.4. Implementasi perbaharuan GBEST dijelaskan pada fungsi `update_gbest`. Proses seleksi ini dilakukan agar didaptkan nilai vektor terkecil baik pada partikel dan GBEST. Kemudian nilai vektor ini akan diubah kedalam range *real* agar didaptkan nilai sebenarnya atau disebut juga proses denormalisasi. Dengan adanya proses seleksi ini, makan partikel dan GBEST yang diperbaharui akan terus mendapatkan hasil yang dosis terbaik dengan pengaruh pada tanaman yang optimal sesuai dengan *input*dari *user* atau petani.

```
1  //fungsi memperbaharui posisi populasi partikel
2  publicvoid update_partikel(double[] update, double[] vUpdate, double[]
3  update2, double[] vUpdate2, int indx, string status)
4  {
5  //jika vektor yang dihasilkan lebih besar dari nilai vektor sebelumnya
6  if (status == "single")
7  {
8  //posisi partikel tidak berubah
9  for (int a = 0; a < po_partikel.GetLength(1); a++)
10 po_partikel[indx, a] = update[a];
11 for(inta=0;a<Vektor1.GetLength(1);a++)
12 Vektor1[indx,a]= vUpdate[a];
13       }
14 //jika nilai vektor yang dihasilkan lebih kecil
15 else
16 {
17 if (vUpdate[0] < vUpdate2[0])
18 {
19 //cek nilai vektor ke-1 kemudian ganti posisi dan vektor ke-1
```

```
for (int a = 0; a < po_partikel.GetLength(1); a++)
po_partikel[indx, a] = update[a];
for (int a = 0; a < Vektor1.GetLength(1); a++)
Vektor1[indx, a] = vUpdate[a];
}
else
{
//cek nilai vektor ke-2 kemudian ganti posisi dan vektor ke-2
for (int a = 0; a < po_partikel.GetLength(1); a++)
po_partikel[indx, a] = update2[a];
for (int a = 0; a < Vektor1.GetLength(1); a++)
Vektor1[indx, a] = vUpdate2[a];
}
}
}
//fungsi memperbaharui posisi populasi GBEST
publicvoid update_gbest(double[] update, double[] vUpdate, double[]
update2, double[] vUpdate2, double[,] partikel, double[,] gbest,
double[,] vektor_gbest, int indx, string apa)
{
int tanda = 0;
    list_gbest.Clear();
    list_vektor.Clear();
//cek nilai pada anggota GBEST
for (int a = 0; a < gbest.GetLength(0); a++)
{
ArrayList inner_list = newArrayList();
for (int b = 0; b < gbest.GetLength(1); b++)
{
inner_list.Add(gbest[a, b]);
}
list_gbest.Add(inner_list);
}
for (int a = 0; a < gbest.GetLength(0); a++)
{
ArrayList inner_list2 = newArrayList();
for (int b = 0; b < vektor_gbest.GetLength(1); b++)
        inner_list2.Add(vektor_gbest[a, b]);
        list_vektor.Add(inner_list2);
      }
//proses seleksi jika hanya salah satu child yang feasible
if (apa == "single")
{
//deklarasi awal untuk lebih kecil dan melakukan pengecekan
if (tanda == 0)
{
for (int a = list_gbest.Count-1; a >= 0; a--)
//jika nilai lebih kecil makan mengganti nilai vektor sebelumnya
{
if (vUpdate[0] < vektor_gbest[a, 0])
{
list_gbest.RemoveAt(a);
                    list_vektor.RemoveAt(a);
                    tanda = 1;
}
}
        }
//proses menambah partikel kepada anggota GBEST
ArrayList list = newArrayList();
ArrayList list2 = newArrayList();
for (int a = 0; a < update.Length; a++) list.Add(update[a]);
for (int a = 0; a < vUpdate.Length; a++) list2.Add(vUpdate[a]);
        list_gbest.Add(list);
        list_vektor.Add(list2);
```

```
84          update_partikel(update, vUpdate, update, vUpdate, indx, apa);
85  po_gbest = newdouble[list_gbest.Count,
86  ((ArrayList)list_gbest[0]).Count];
87          Vektor2 = newdouble[list_vektor.Count,
88  ((ArrayList)list_vektor[0]).Count];
89  //proses mengubah nilai array ke array list
90  for (int a = 0; a < po_gbest.GetLength(0); a++)
91  {
92  ArrayList inner_list = (ArrayList)list_gbest[a];
93  ArrayList inner_list2 = (ArrayList)list_vektor[a];
94  for (int b = 0; b < po_gbest.GetLength(1); b++)
95  po_gbest[a, b] = (double)inner_list[b];
96  for (int b = 0; b < Vektor2.GetLength(1); b++)
97  Vektor2[a, b] = (double)inner_list2[b];
98  }
99  //proses jika child1 dan child2 feasible
100 else
101   {
102 for (int a = 0; a < mix_v.GetLength(0); a++)
103 {
104 for (int b = 0; b < gbest.GetLength(1); b++)
105 {
106 mix_gbest[0, b] = update[b];
107       mix_gbest[1, b] = update2[b];
108 }
109 mix_v[0, a] = vUpdate[a];
100             mix_v[1, a] = vUpdate2[a];
101 }
102 if (mix_v[0, 0] > mix_v[1, 0] ) max = 0;
103 else max = 1;
104 if (tanda == 0)
105 //deklarasi awal untuk lebih kecil dan melakukan pengecekan GBEST
106     {
107 for (int a = list_gbest.Count-1; a >= 0; a--)
108         {
109 //proses mengganti GBEST berdasarkan vektor2
110 if (mix_v[max, 0] < vektor_gbest[a, 0])
111             {
112               list_gbest.RemoveAt(a);
113               list_vektor.RemoveAt(a);
114               tanda = 1;
115             }
116           }
117         }
118 //menambah anggota baru kedalam anggota GBEST
119 for (int a = 0; a < 2; a++)
120 {
121 ArrayList list = newArrayList();
122 for (int b = 0; b < update.GetLength(0); b++)
123 {
124                   list.Add(mix_gbest[a,b]);
125 }
126 list_gbest.Add(list);
127 }
128 for (int a = 0; a < 2; a++)
129 {
130 ArrayList list2 = newArrayList();
131 for (int b = 0; b < vUpdate.GetLength(0); b++)
132 {
133 list2.Add(mix_v[a,b]);
134 }
135 list_vektor.Add(list2);
136 }
137 update_partikel(update,vUpdate,update2,
```

```
138                 vUpdate2,indx, apa);
139 po_gbest = newdouble[list_gbest.Count,
140 ((ArrayList)list_gbest[0]).Count];
141                 Vektor2 = newdouble[list_vektor.Count,
142 ((ArrayList)list_vektor[0]).Count];
143 for (int a = 0; a < po_gbest.GetLength(0); a++)
144 {
145 ArrayList inner_list = (ArrayList)list_gbest[a];
146 ArrayList inner_list2 = (ArrayList)list_vektor[a];
147 for (int b = 0; b < po_gbest.GetLength(1); b++)
148 po_gbest[a, b] = (double)inner_list[b];
149 for (int b = 0; b < Vektor2.GetLength(1); b++)
150 Vektor2[a, b] = (double)inner_list2[b];
151 }
    }
```

Gambar 4.10 *Source Code* Proses Perbaharui Partikel dan GBEST

Penjelasan Gambar 4.10sebagai berikut:

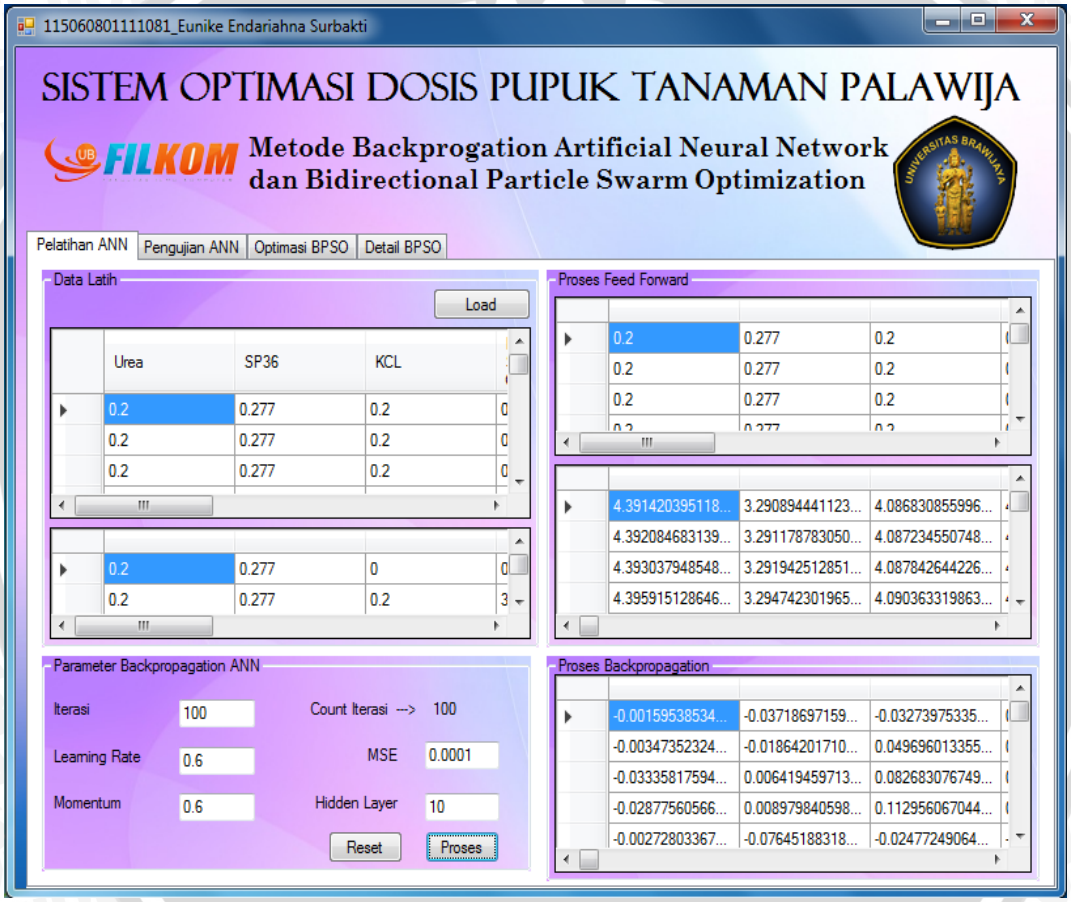
1. Baris 12 adalah proses posisi partikel dan vektor tidak berubah jika nilai vektor yang dihasilkan lebih besar dari vektor sebelumnya.
2. Baris 23,31 adalah proses posisi partikel dan vektor sesuai dengan posisi dan vektor baru jika nilai vektor yang dihasilkan lebih kecil.
3. Baris 71-72 adalah proses menggantikan anggota GBEST jika nilai vektor dari $child_1$dan $child_2$ yang terpilih lebih kecil dari anggota GBEST.
4. Baris 82-83 adalah menambahkan posisi $child_1$dan $child_2$ke dalam anggota GBEST jika nilai vektor yang dihasilkan layak.
5. Baris 108-109 adalah proses menambahkan vektor $child_1$dan $child_2$ke dalam anggota GBEST jika nilai vektor yang dihasilkan layak.
6. Baris 112-113 adalah proses menghapus vekotr anggota GBEST yang vektor nya lebih besar dari vektor vektor $child_1$dan $child_2$.

4.4 Implementasi Antarmuka

Implementasi antarmuka ini terdiri dari beberapa bagian yaitu, proses pelatihan *backpropagation*, proses pengujian *backpropagation*, proses optimasi BPSO dan detail BPSO yang terdiri dari posisi akhir populasi partikel dan GBEST. Berikut hasil implementasi program yang dibuat :

4.4.1 Antarmuka Hasil Pelatihan *Backpropagation*

Pada antarmuka ini akan ditampilkan hasil dari proses pelatihan *backpropagtion*. *Buttonload* digunakan untuk mengupload data latih, kemudian data latih ini akan ditampilkan pada *Datagridview*. Kemudian mengisi nilai-nilai yang digunakan sebagai parameter ANN. Jika ditekan tombol proses akan ditampilkan *Datagridview*yang berisi nilai maksimal dan minimum dari data latih, proses feedforward dan hasil akhir proses *backpropagation* berupa nilai MSE. Berikut Gambar4.11:

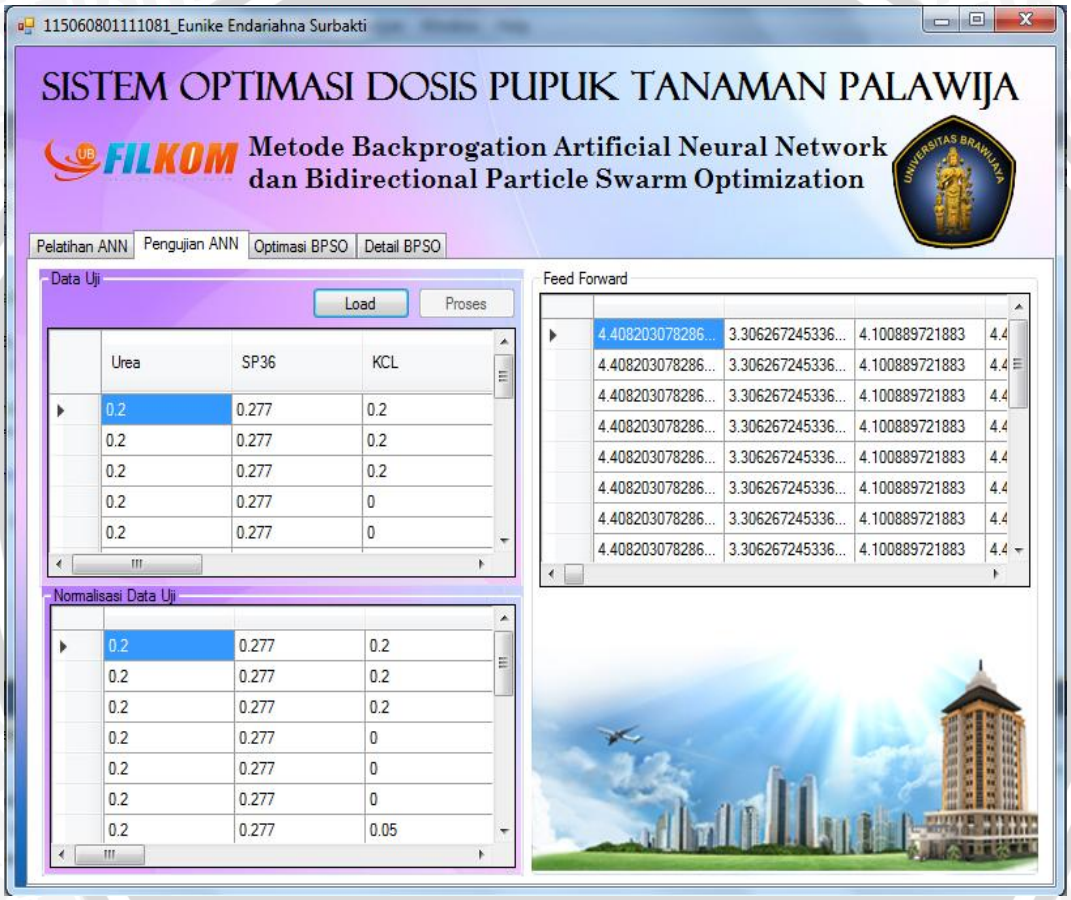


Gambar 4.11 Antarmuka Hasil Pelatihan *Backpropagation*

4.4.2 Antarmuka Hasil Pengujian *Backpropagation*

Pada implementasi ini, akan menampilkan hasil pengujian dari *Backpropagation*. Antarmuka ini menggunakan bobot terakhir pada proses pelatihan. Pada pengujian, prosesnya hanya sampai pada tahap *feedforward*. Langkah awal yang dilakukan adalah, tekan *button Loada untuk* mengupload data

uji. Kemudian menekan *button* proses, untuk melakukan proses pengujian *backpropagation*. Maka akan ditampilkan hasil dalam *datagridview*. Tampilan menggunakan *datagridview* digunakan agar dapat melihat hasil dan proses secara lebih detail dan lengkap. Hasil yang ditampilkan dalam *datagridview* adalah data uji, hasil normalisasi data uji dan proses *feedforward* data uji. Dalam Normalisasi Data uji juga ditampilkan *output* terakhir pengaruh pada tanaman yang telah dinormalisasi, ke dalam *range* nilai aslinya. Berikut Gambar 4.12 :

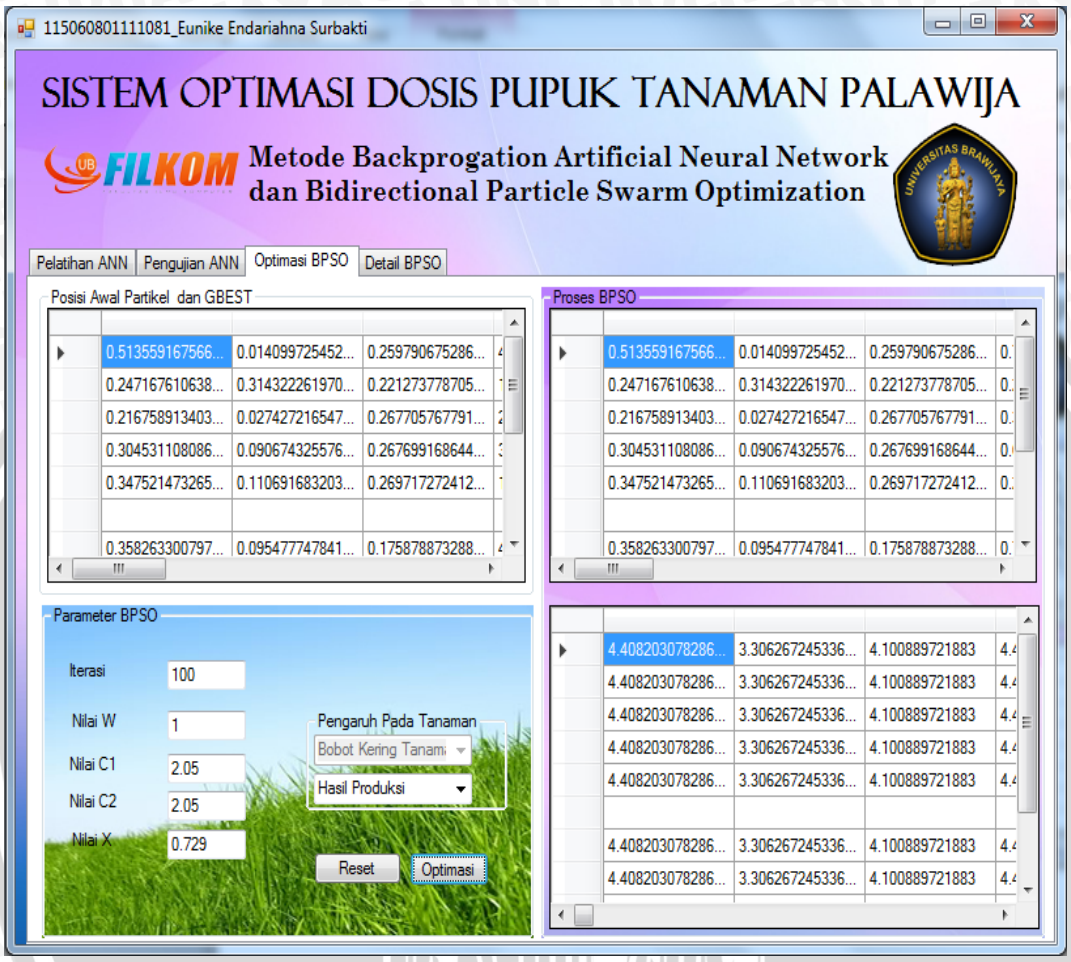


Gambar 4.12 Antarmuka Hasil Pengujian *Backpropagation*

4.4.3 Antarmuka Hasil Optimasi BPSO

Halaman antarmuka hasil optimasi akan menampilkan proses optimasi dengan metode BPSO. *User* terlebih dahulu memasukan nilai parameter yang dibutuhkan dalam optimasi menggunakan BPSO. Dalam proses optimasi ini, bobot yang digunakan adalah bobot didapatkan dari proses pelatihan dan pengujian ANN. Kemudian menekan *button* optimasi untuk melakukan proses

optimasi. Posisi awal partikel dan GBEST dan proses BPSO ditampilkan dalam bentuk tampilan *datagridview*. Berikut adalah tampilan antarmuka hasil optimasi pada Gambar 4.13:



Gambar 4.13 Antarmuka Optimasi BPSO

4.4.4 Antarmuka Hasil Detail Optimasi BPSO

Antarmuka hasil detail Optimasi BPSO,menampilkan posisi akhir dari GBEST dan partikel dalam *datagridview*. Antarmuka ini merupakan implementasi akhir dari sistem optimasi dosis pupuk. Pada antarmuka ditampilkan kombinasi dosis yang disarankan sesuai dengan pengaruh pada tanamanyang dioptimalkan. Berikutantarmuka hasil detail BPSO pada Gambar 4.14:

Gambar 4.14 Antarmuka Detail Optimasi BPSO

BAB V
PENGUJIAN DAN ANALISIS

Pada bab pengujian dan analisis membahas tentang pengujian dan analisis terhadap aplikasi yang telah dibuat. Kemudian prosespengujian yang dilakukan berdasarkan skenario yang telah dijelaskan pada Bab III kemudian menganalisis hasil pengujian yang didapatkan.

5.1 Pengujian

Pada sub bab ini dijelaskan mengenai pengujian aplikasi. Pengujian dilakukan hanya pada parameter *backpropagation.* Pengujian yang dilakukan berdasarkan skenario sebagai berikut :

1. Pengujian terhadap variasi jumlah data latih dan data uji.
2. Pengujian jumlah iterasi pada *backpropagation*.
3. Pengujian variasi jumlah *hiddenlayer*.
4. Pengujian variasi range nilai *learning rate* danmomentum.

5.2 Hasil Pengujian

Berdasarkan skenario pengujian yang telah dibuat, maka hasil dari proses pengujian tersebut adalah :

5.2.1 Hasil Pengujian Terhadap Variasi Data Latih dan Data Uji

Pengujian Terhadap variasi data uji dan data latih untuk mendapatkan, perbandingan jumlah data dengan nilai MSE yang terkecil. Pengujian dilakukan sebanyak 5 kali untuk masing-masing variasi perbandingan jumlah data latih dan data uji.

Pada pengujian ini terdapat 9 variasi jumlah data latih dan data uji. Pengujian awal dilakukan pada variasi data latih dan uji, sehingga untuk nilai parameter ANN mengacu hasil dari penelitianNavalertporn (2011). Jumlah iterasi yang digunakan adalah sebanyak100 iterasi, nilai *learning rate* adalah 0.64, nilai momentum 0.4 dan untuk jumlah *hiddenlayer*sebanyak 4. Berikut hasil pengujian pada Tabel 5.1:

Tabel 5.1 Hasil Pengujian Variasi Data Latih dan Data Uji

Perbandingan		Nilai MSE percobaan ke-*i*					Rata-rata MSE
Data Latih	Data Uji	1	2	3	4	5	
90%	10%	9.3668E-03	9.3103E-03	9.2373E-03	9.4618E-03	1.2592E-02	9.9936E-03
80%	20%	1.2965E-02	1.3222E-02	1.3253E-02	1.3238E-02	1.3126E-02	1.3161E-02
70%	30%	2.0570E-02	2.0593E-02	2.0606E-02	2.0569E-02	2.0590E-02	2.0586E-02
60%	40%	2.0163E-02	2.0373E-02	2.0392E-02	2.0207E-02	2.0441E-02	2.0315E-02
50%	50%	1.9542E-02	1.9542E-02	1.9305E-02	1.9461E-02	1.9562E-02	1.9482E-02
40%	60%	1.2538E-02	1.2784E-02	1.2775E-02	1.2439E-02	1.2638E-02	1.2635E-02
30%	70%	1.1589E-02	1.1659E-02	1.1661E-02	1.1581E-02	1.1535E-02	1.1605E-02
20%	80%	1.9980E-02	2.0092E-02	1.9905E-02	1.9455E-02	1.9922E-02	1.9871E-02
10%	90%	2.3911E-02	2.4108E-02	2.4125E-02	2.3654E-02	2.4383E-02	2.4036E-02

Pada Tabel 5.1 didapatkan perbandingan variasi jumlah data yang terbaik adalah data latih 90% dan data uji sebesar 10% dengan nilai MSE terkecil. Nilai MSE yang dihasilkan adalah 9.9936E-03.Semakin banyaknya data latih maka sistem pembelajaran akan semakin baik. Variasi data latih dan data uji yang menghasilkan nilai MSE terkecil dari hasil pengujian akan digunakan untuk proses pengujian selanjutnya.

5.2.2 Hasil Pengujian Variasi Jumlah Iterasi

Setelah didapatkan variasi data latih dan data uji, kemudian dilanjutkan proses pengujian variasi jumlah iterasi. Pengujian ini juga masih untuk mendapatkan nilai MSE terkecil. Pengujian Jumlah Iterasi adalah banyaknya pengulangan yang dilakukan untuk mendapatkan bobot terbaik sampai dengan jumlah iterasi maksimal. Variasi jumlah iterasi yang dilakukan adalah dari range 100-1000 dengan kelipatan seratus sehingga didapatkan 10 kali pengujian untuk jumlah iterasi. Data yang digunakan adalah data latih 90% berdasarkan hasil pengujian, sedangkan parameter ANN lainnya masih berdasarkan hasil penelitian Navalertporn (2011) nilai *learning rate* 0.64, nilai momentum adalah 0.4 dan

untuk jumlah *hiddenlayer*adalah 4. Hasil pengujian yang didapatkan pada jumlah iterasi ditunjukkan pada Tabel 5.2 :

Tabel 5.2 Hasil Pengujian Variasi Jumlah Iterasi

Jumlah Iterasi	Nilai MSE percobaan ke-*i*					Rata-rata MSE
	1	2	3	4	5	
100	9.2588E-03	9.4139E-03	9.3670E-03	9.3307E-03	9.3586E-03	9.3458E-03
200	9.4591E-03	9.4653E-03	9.4660E-03	9.4477E-03	9.4554E-03	9.4587E-03
300	9.4384E-03	9.4562E-03	9.5019E-03	9.4587E-03	9.4791E-03	9.4669E-03
400	9.4952E-03	9.4997E-03	9.5095E-03	9.5077E-03	9.4959E-03	9.5016E-03
500	9.4953E-03	9.4924E-03	9.4965E-03	9.4919E-03	9.4935E-03	9.4939E-03
600	9.4906E-03	9.5023E-03	9.4960E-03	9.4911E-03	9.5084E-03	9.4977E-03
700	9.4967E-03	9.5025E-03	9.5044E-03	9.5082E-03	9.5098E-03	9.5043E-03
800	9.5120E-03	9.5012E-03	9.5083E-03	9.5043E-03	9.5120E-03	9.5076E-03
900	9.5078E-03	9.5075E-03	9.5137E-03	9.5070E-03	9.5136E-03	9.5099E-03
1000	9.4947E-03	9.5093E-03	9.5006E-03	9.5061E-03	9.5047E-03	9.5031E-03

Pada Tabel 5.2 nilai MSE terkecil didapatkan dari iterasi yang ke-100. Nilai MSE yang diperoleh adalah 9.3458E-03. Dari pengujian didapatkan bahwa iterasi 100 menghasilkan nilai MSE yang lebih kecil. Pada rata-rata MSE yang didapatkan dari hasil pengujian perubahan nilai MSE tidak terlalu besar dan mendakati nilai stabil. Ini disebabkan dalam menghasilkan bobot pada pelatihan dibangkitkan secara random, tetapi semakin banyaknya iterasi maka sistem akan belajar dan menemukan pola yang semakin stabil.

5.2.3 Hasil Pengujian Variasi Nilai *HiddenLayer*

Proses pengujian Variasi Nilai *Hiddenlayer* ini dilakukan untuk mendapatkan banyaknya nilai bobot, sehingga sistem semakin teliti. Tetapi, pengujian ini melibatkan waktu lebih lama, karena semakin banyaknya jumlah *hiddenlayer* yang di *input*. Pada pengujian ini range *hiddenlayer*1-10, data latih 90%, data uji 10 % kemudian iterassi 100 sesuai hasil pengujian sedangkan nilai *learning rate* 0.64 , nilai momentum 0.4 masih menggunakan nilai parameter yang didapatkan dari pengujian Navalertporn (2011). Hasil pengujian ditampilkan pada Tabel 5.3 sebagai berikut :

Tabel 5.3 Hasil Pengujian Variasi Nilai *HiddenLayer*

Jumlah *Hiddenlayer*	Nilai MSE percobaan ke-*i*					Rata-rata MSE
	1	2	3	4	5	
1	0.00949585	0.009485334	0.009504295	0.009519997	0.009513514	9.5038E-03
2	0.00942244	0.009439214	0.009468283	0.009460159	0.009466371	9.4513E-03
3	0.00943802	0.009433775	0.009408727	0.009308037	0.009431745	9.4041E-03
4	0.00933172	0.009264569	0.00925281	0.009227626	0.009308762	9.2771E-03
5	0.00923412	0.009233434	0.009239055	0.009238781	0.009234523	9.2360E-03
6	0.00916919	0.009151205	0.009084285	0.008896773	0.009232952	9.1069E-03
7	0.00893747	0.008899469	0.008983357	0.009049867	0.009140308	9.0021E-03
8	0.0087233	0.008661485	0.008774236	0.008911225	0.008668561	8.7478E-03
9	0.00865829	0.008758841	0.008637658	0.008630312	0.008613004	8.6596E-03
10	0.00860622	0.008605419	0.008607905	0.008656086	0.008607006	8.6165E-03

Berdasarkan Tabel 5.3didapatkan nilai MSE terkecil pada jumlah *hiddenlayer* ke-10. Semakin banyak jumlah *hiddenlayer*maka bobot yang dibangkitkan untuk *input* ke *hidden* dan *hidden* ke*output*akan semakin banyak. Ini membuat sistem komputasi akan bekerja lebih lama tetapi semakin besar nilai *hiddenlayer* maka semakin teliti kerja sistem dalam mepredikasi nilai asli atau mendekati target. Hasil MSE yang dihasilkan dengan jumlah *hiddenlayer*10 adalah 8.6165E-03.

5.2.4 Hasil Pengujian Variasi Nilai *Learning Rate* dan Momentum

Pengujian yang terakhir dilakukan adalah variasi nilai *learning rate* dan momentum. Belum ada aturan pasti dalam pemberian nilai *learning rate* dan momentum dan momentum, sehingga diperlukan proses pengujian pada nilai paramet ini. Berdasarkan buku Sutojo dalam kecerdasan buatan Variasi *learning rate* antara nilai 0.9-0.05 sedangkan momentum antara 0.5-0.9. kemudian disarankan jika nilai *learning rate* besar maka nilai momentum yang digunakan kecil begitu sebaliknya. Parameter yang digunakan berdasarkan hasil pengujian

yang telah dilakukan sebelumnya yaitu data latih 90%, data uji 10%,iterasi 100 dan *hiddenlayer* 10. Hasil pengujian pada Tabel 5.4 didapatkan sebagai berikut: :

Tabel 5.4 Hasil Pengujian Nilai *Learning Rate* dan momentum

Kombinasi		Nilai MSE percobaan ke-*i*					Rata-rata MSE
Learning rate	momentum	1	2	3	4	5	
0.9	0,5	0.008719465	0.008758014	0.008725716	0.008740236	0.008677156	8.7241E-03
0.8	0,5	0.008694502	0.00866709	0.008678964	0.00864036	0.008651042	8.6664E-03
0.7	0,6	0.008673427	0.008629509	0.008615691	0.008654544	0.00859606	8.6338E-03
0.6	0,6	0.008602488	0.00860593	0.008601288	0.008600487	0.008601491	8.6023E-03
0.5	0,7	0.008605643	0.00861783	0.008630593	0.008651399	0.008632379	8.6276E-03
0.4	0,7	0.008810813	0.008848607	0.008770042	0.008906085	0.00884024	8.8352E-03
0.3	0,8	0.009039837	0.009844126	0.00992623	0.009430791	0.009768539	9.6019E-03
0.2	0,8	0.011949745	0.011568738	0.011501755	0.012079635	0.011091109	1.1638E-02
0.1	0.9	0.012732741	0.013949973	0.013616822	0.013355232	0.013437772	1.3419E-02
0.05	0.9	0.015516272	0.015281584	0.015107062	0.015141029	0.030440792	1.8297E-02

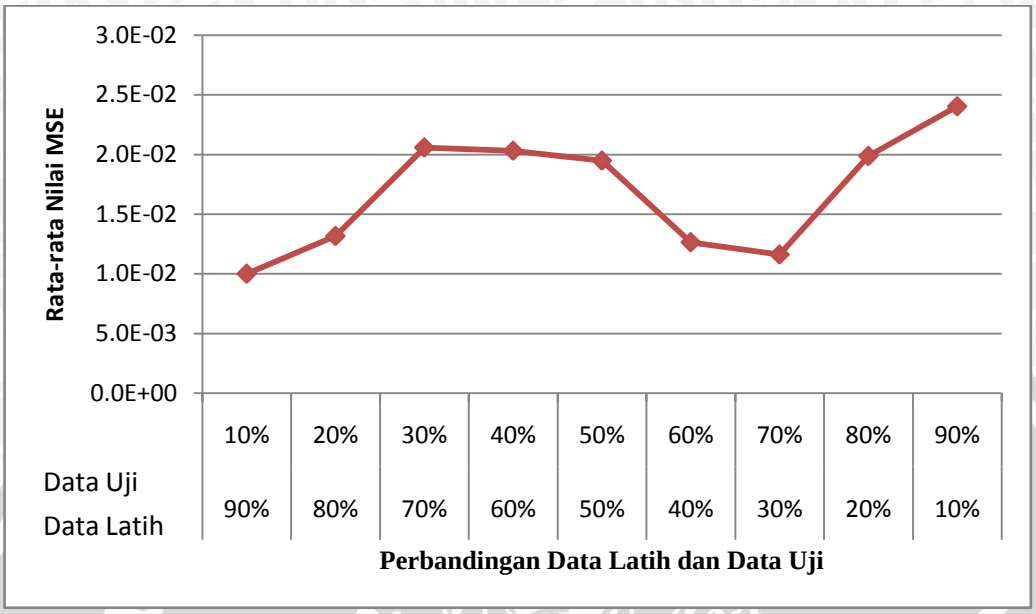
Pada Tabel 5.4didapatkan kombinasi variasi Nilai *Learning rate* dan momentum adalah 0.6 dan 0.6 dengan nilai MSE terkecil. Setiap variasi nilai *Learning rate* dan momentum menghasilkan nilai MSE yang berbeda sehingga *Learning rate* dan momentum berpengaruh dalam melatih sistem untuk menjadi teliti. Variasi nilai *learning rate* dan momentum tidak mempengaruhi proses atau waktunya komputasi. Dari pengujian yang dilakukan didapatkan hasil nilai MSE terkecil adalah 8.6023E-03.

5.3 Analisi Hasil Pengujian

Pada bagian sub bab analisis hasil pengujian ini dilakukan analisis terhadap empathasil pengujian yang telah dilakukan sebelumnya. Sehingga hasil analisis pengujian akan dijelaskan pada sub bab 5.3.1-5.3.4

5.3.1 Analisis Hasil Pengujian Terhadap Perbandingan Jumlah Data Latih dan Data Uji

Berdasarkan Tabel 5.1 maka dibuat grafik hasil pengujian seperti yang ditunjukkan pada Gambar 5.1 adalah sebagai berikut:

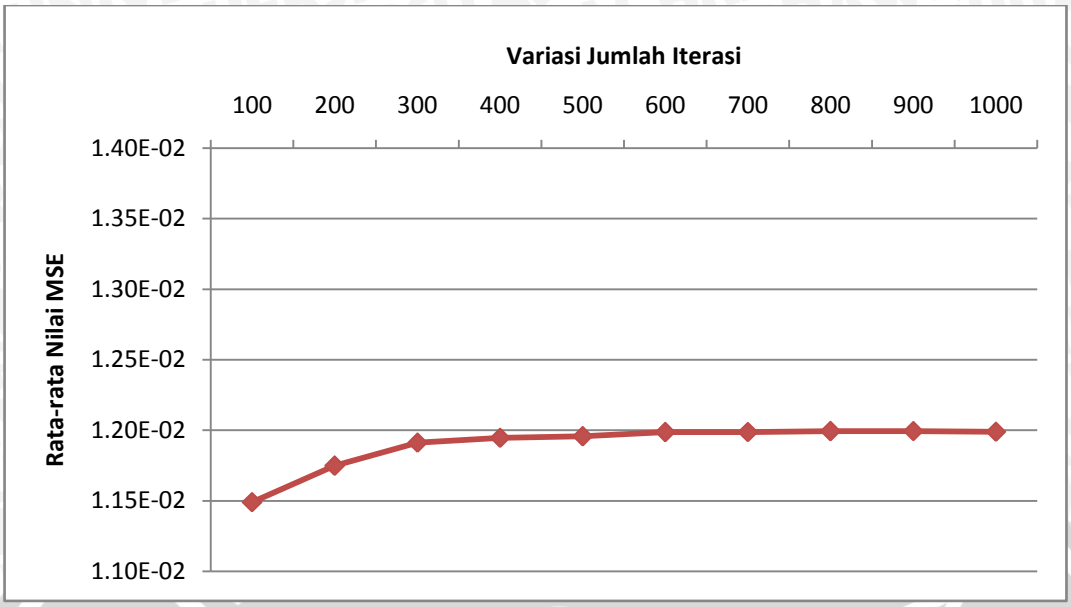


Gambar 5.1 Grafik Hasil Pengujian Variasi Data Latih dan Data Uji

Berdasarkan Gambar 5.1 memberikan pengaruh dalam menghasilkan nilai MSE. Organisasi data yang baik dan jumlah data mempengaruhi dalam memberikan tingkat ketelitian dari sistem yang dibangun. Selain itu kualitas data yang baik akan menghasilkan proses pembelajaran atau *learning* backpropagation yang baik juga. Sehingga data yang digunakan harus dari pakar atau telat terbukti keakuratannya dalam penelitian sebelumnya. Denganmelakukan proses pengujian pada data latih dan data uji, nilai MSE yang yang dihasilkan juga mendekatikonsisten dengan range nilai yang sangat kecil. Ini menunjukan bahwa bobot yang dihasilkan dari sistem juga konsisten dan optimal. Semakin banyak jumlah data latih akan berpengaruh terhadapa waktu kompuatasi yang semakin lama. Pada Gambar 5.1 didapatkan variasi data uji 10% dan data latih 90% memberikan nilai MSE terkecil dengan nilai 9.9936E-03.

5.3.2 Analisis Hasil Pengujian Terhadap Perbandingan Jumlah Iterasi

Berdasarkan Tabel 5.2maka dibuat grafik hasil pengujian seperti yang ditunjukkan pada Gambar 5.2 sebagai berikut :

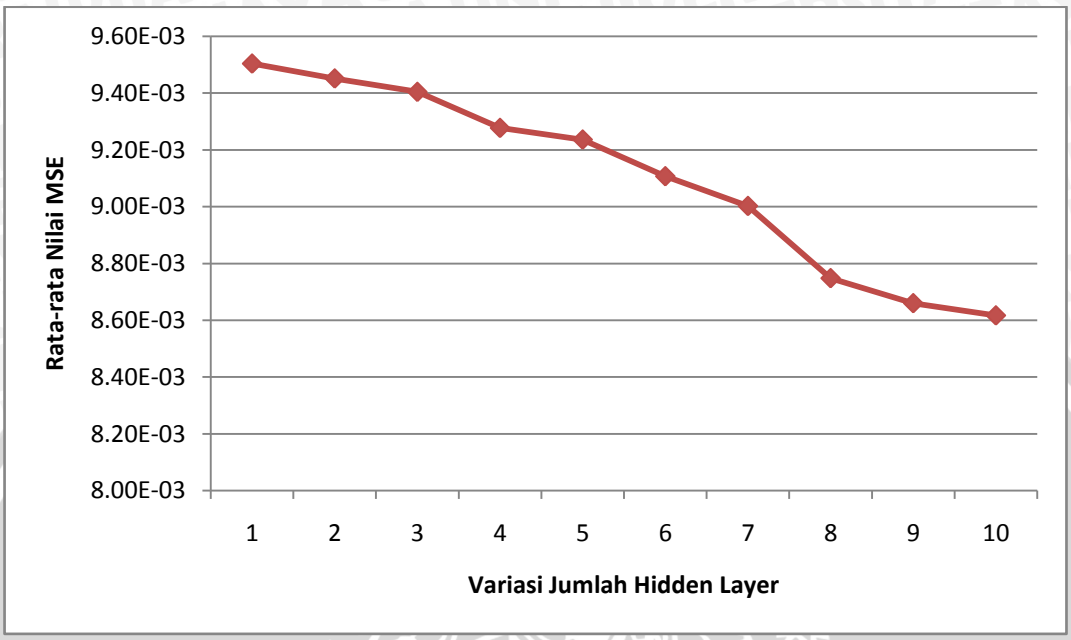


Gambar 5.2 Grafik Hasil Pengujian Variasi Jumlah Iterasi

Pada Gambar 5.2 Grafik Hasil Pengujian Variasi Jumlah Iterasi, didapatkan hasil rata-rata nilai MSE dari jumlah nilai iterasi. Dalam *backpropagation* banyak parameter yang mempengaruhi tingkat prediksi dari algoritma ini. Salah satunya, adalah dengan variasi jumlah iterasi. Variasi jumlah iterasi yang dilakukan dengan kelipatan 100, didapatkan nilai MSE mengalami perubahan. Dalam beberapa kasus tidak selalu, jumlah iterasi semakin banyak akan menghasilkan nilai MSE yang semakin kecil. Dalam Grafik ini didapatkan pada iterasi ke 200 sampai 1000 atau semakin banyak iterasi yang dilakukan maka MSE semakin besar dan mengalami peningkatan. Walaupun demikian nilai MSE yang dihasilkan juga lebih stabil walaupun mengalami penaikan atau menjadi stabil. Sehingga iterasi 100 sudah cukup baik sebagai hasil pengujian jumlah iterasi memberikan pengaruh melatih sistem. Berdasarkan Gambar 5.2 jumlah iterasi 100 menghasilkan nilai MSE terkecil dengan rata-rata nilai MSE 9.3458E-03. Terjadi penurunan nilai MSE yang dihasilkan dari pengujian data latih dan data uji. Ini membuktikan bahawa variasi jumlah iterasi mampu melatih sistem agar lebih optimal.

5.3.3 Analisis Hasil Pengujian Terhadap Variasi *HiddenLayer*

Selanjutnya, Berdasarkan Tabel 5.3 maka grafik hasil pengujian ditunjukkan pada Gambar 5.3sebagai berikut:

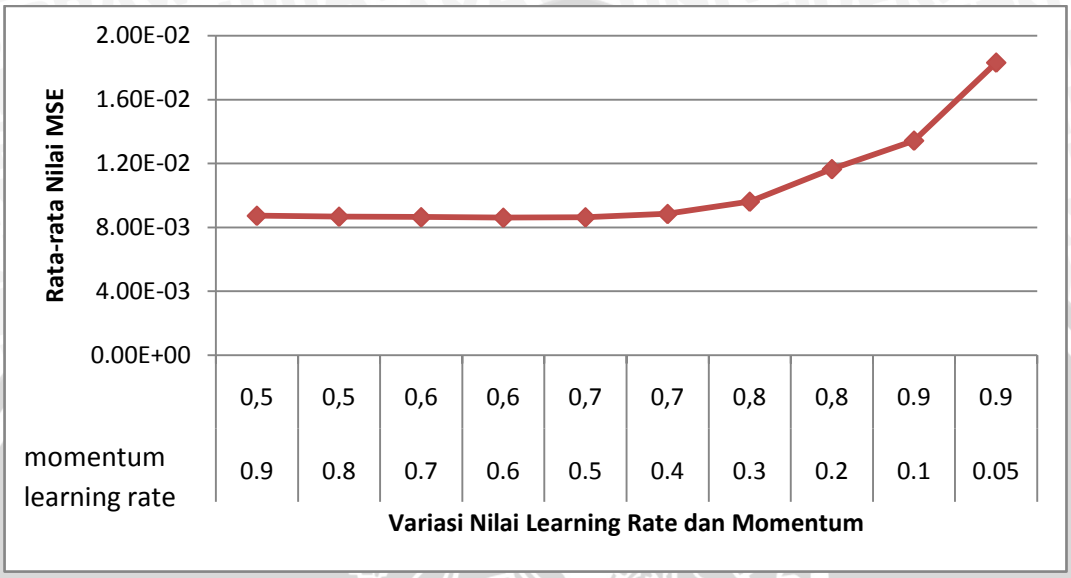


Gambar 5.3 Grafik Hasil Pengujian Variasi *HiddenLayer*

Dari hasil Pengujian yang dilakukan terhadap variasi *hiddenlayer*, diperoleh nilai MSE terkecil didapatkan dari jumlah *hiddenlayer*10. Banyaknya jumlah *hiddenlayer*, mempengaruhi sistem dalam menghasilkan perubahan selisih bobot yang semakin kecil dan teliti sehingga didapatkan nilai MSE yang baik. Jumlah *hidden layer* semakin besar juga menyebabkan bobot random yang dihasilkan dari *input* ke *hidden* dan *hidden* ke *output* juga semakin banyak. Sehingga proses komputasi yang dilakukan untuk proses pembelajaran *learning rate* akan membutuhkan waktu yang semakin lama agar optimal. Dalam pengujian ini, dilakukan variasi *hiddenlayer* antara 1-10 dengan kelipatan 1. Pada Gambar 5.3, nilai MSE yang dihasilkan oleh jumlah *hiddenlayer* 10 adalah 8.6165E-03. Pada pengujian ini nilai MSE dihailkan juga semakin kecil dan jumlah *hidden layer* yang digunakan juga besar sehingga sistem yang dihasilkan semakin baik. hasil dai pengujian jumlah *HiddenLayer* yang diperoleh ini digunakan pada pengujian selanjutnya yaitu pada pengujian *learning rate dan* momentum

5.3.4 Analisis Hasil Pengujian Variasi Nilai *Learning Rate* danMomentum

Pada pengujian terakhir terhadap Tabel 5.4nilai *learning rate* antara 0.05-0.9 sedangkan momentum 0.5-0.9 didapatkan grafik hasil pengujian pada Gambar 5.4 adalah sebagai berikut :



Gambar 5.4 Grafik Hasil Pengujian Variasi *Learning Rate* dan Momentum

Learning Rate dan Momentum digunakan untuk mepercepat proses pelatihan dalam menghasilkan bobot yang terbaik. Pada umumnya rentang nilai *learning rate*adalah 0.1-0.9 sedangkan momentum 0.5-0.9. Tidak ada aturan yang mutlak untuk nilai *learning rate* dan momentum agar menghasilkan nilai MSE yang kecil oleh sebab itu perlunya pengujian untuk *learning rate* dan momentum.Berdasarkan Gambar 5.4didapatkan nilai MSE terkecil dengan*learning rate*0.6 dan momentum 0.6. Nilai MSE yang dihasilkan adalah 8.6023E-03. Pengujian ini menunjukkan adanya perubahan hasil MSE dengan variasi dari nilai *learning rate* dan momentum. Parameter *learning rate* dan Momentumsangat mempengaruhi proses pelatihan. Dari hasil nilai MSE yang didapatkan adanya penurunan nilai MSE yang dihasilkan sehingga hasil akhir MSE pengujian ini akan menjadi niali parameter terbaik untuk sistem *learning backpropagtion.*

BAB VI
PENUTUP

Pada bab terakhir ini diberikan kesimpulan dan saran dari hasil penelitian yang telah dilakukan. Kesimpulan akan dijelaskan dalam sub bab 6.1 kemudian saran akan diuraikan dalam sub bab 6.2.

6.1 Kesimpulan

Berdasarkan Penelitian dan pengujian yang telah dilakukan terhadapa optimasi dosis pupuk pada tanaman palawija menggunakan metode ANN dan BPSO adalah sebagai berikut :

1. Dalam Implementasi Algoritma *Artificial Neural Network* dan *Bidirectional Particle Swarm Optimization* langkah pertama yang dilakukan adalah proses pelatihan data latih dan pengujian data uji. Data yang digunakan adalah hasil penelitian terdahulu yang dilakukan oleh pakar mengenai Ketersediaan Hara NPK dengan Biochar pada pertumbuhan Vegetatif Tanaman Jagung. Objek yang diamati adalah Jagung, pupuk UREA, SP_{36}, KCL, dan Biochar. Kemudian *output*pengaruh pada tanaman yang diamati adalah bobot kering tanaman, panjang tongkol, diameter tongkol, bobot 1000 butir dan Hasil Produksi. Parameter pada *backpropagation* akan melatih data untuk mencari bobot terbaik dan meprediksi pengaruh pada tanaman dengan menggunak MSE (*Minimum Sequnce Error*). Semakin kecil nilai MSE, maka sistem yang dihasilkan semakin baik. Algoritma BPSO, akan meproses dua *input*pengaruh pada tanaman yang akan dioptimalkan. Parameter BPSO akan melakukan seleksi pada *partikel* yang terdiri dari komposisi pemberian masing-masing pupuk. Sehingga *output*yang dihasilkan, berupa kombinasi jumlah dosis dari masing-masing pupuk untuk mendapatkan hasil maksimal dari pengaruh pada tanaman.
2. Hasil Pengujian dari penelitian ini adalah nilai MSE pada *backpropagation* dan nilai vektor pada BPSO. Pada penelitian ini pengujian dilakukan hanya pada parameter *backpropagation.* Nilai MSE terkecil yang didapatkan

adalah8.6023E-03.Nilai ini didapatkan dengan pengujian menggunakan data latih

90%, data uji 10%, iterasi 100 kali, jumlah *hiddenlayer*10, *learning rate* 0.6 dan momentum 0.6. Sedangkan untuk parameter BPSO sesuai dengan *standard* PSO. Pengaruh pada tanaman yang dioptimalkan pada BPSO adalah Bobot Kering Tanaman dan Hasil Produksi. Nilai vektor akhir dari GBEST yang didapatkan dari pengaruh pada tanaman tersebut adalah 0.222239 dan 0.14286. Kemudian, nilai vektor ini akan diubah ke range nilai sebenarnya sehingga untuk mendapatkan bobot kering tanaman tanaman 4.4964 ton/ha dan hasil produksi 6.99985 ton/ha makadibutuhkan pupuk Urea 0.191 ton/ha atau 191 kg/ha, SP_{36}0.201 ton/ha atau 201 kg/ha, KCL 0.288 ton/ha atau 288 kg/ha dan Bioncah 48.3 ton/ha. Dari hasil yang didapatkan, sistem optimasi dosis pupuk palawija pada tanaman jagung ini mampu memberikan solusi hasil prediksi dan optimasi pengaruh pada tanaman dibandingkan dengan penelitian yang dilakukan pada tanah secara langsung. Sehingga sistem yang dihasilkan lebih efisien tidak memakan waktu, tetapi tetap dibutuhkannya data penelitian terlebih dahulu untuk melatih sistem.

6.2 Saran

1. Hasil penelitian dari implementasi Algoritma ANN dan BPSOuntuk optimasi dosis pupuk pada tanaman Palawija ini dapat dikembangkan dan dijadikan bahan penelitian lebih lanjut agar dapat memilih tanaman Palawija lainnya, lebih banyak jenis pupuk, jenis lahanserta pengaruh pada tanaman.
2. Pengujian pada sistem optimasi dosis pupuk ini juga dapat dilanjutkan pada parameter-parameter pada algoritma BPSO. Algoritma BPSO ini juga dapat dikembangkan jika yang ingin dioptimalkan pengaruh pada tanaman tidak hanya dua, tetapi lebih.
3. Pada *backpropagation* untuk menghasilkan bobotdapat mencoba algoritma *nguyen-widrow*sebagai perbandingan pada sistem dalam *running time* dan bobot akhir yang didapatkan.

DAFTAR PUSTAKA

[BAM-12] Bambang, Sampto A. 2012. Si Hitam Biochar yang Multiguna. PT. Perkebunan Nusantara X (Persero), Surabaya.

[DEB-89] Deb, Kalyanmoy and David E. Goldberg. 1989. *An Investigation of Niche and Species Formation in Genetic Function Optimization. Department of Engineering Mechanics, The Univesrity of Alabama, Tuscalososa, Albania.*

[EDO-14] Edo, Ryandhimas Zezario. 2014. Implementasi *Backpropagation Neural Network* dalam Pembangkitan Otomatis Fungsi Keanggotaan *Fuzzy* pad Penderita Penyakit Hepatitis, Skripsi Program Teknologi Informasi dan Ilmu Komputer. Universitas Brawijaya, Malang.

[GOZ-11] Gozali, Karnadi dan Yakup. 2011. Pengelolaan Hara dan Pemupukan pada Budidaya Tanaman Jagung (*Zea Mays L.*) di Lahan Kering. Program Studi Agroekoteknologi, Fakultas Pertanian, Universitas Sriwijaya, Palembang.

[DBR-07] D. Bratton, J. Kennedy, *Defining a standard for particle swarm optimization, in: Proceeding of the IEEE SwarmIntelligence Symposium, SIS'2007, 2007Apr.1–5, IEEE Press, Honolulu, Hawaii, New Jersey, 2007, pp. 120–127.*

[KEN-95] Kennedy, James and Russell Eberhart, *Particle Swarm Optimization, Purdue School of Engineering and Technology Indianapolis. Washington, DC.1995*

[KSH-09] K. Shibata and Y. Ikeda, *Effect of number of hidden neurons on learning in large-scale layered neural networks, in Proceedings of the ICROS-SICE International Joint Conference 2009 (ICCAS-SICE '09), pp. 5008–5013, August 2009.*

[KOU-08] Koul, Opender., dkk. 2008. *Areawide Pest Management: Theory and Implementation CABI. ISBN 1845933737. National Academy og Agricultural Sciences and The Indian Academy of Entomology. India.*

[KUR-14] Kurli, Anis. 2014. Pengaruh Pemberian Zeolit, Kompos Jerami dan Kombinasinya terhadap Perbaikan Sifat Fisik Inceptisol serta Produksi Tanaman Jagung, Skripsi Fakultas Pertanian. Universitas Brawijaya, Malang.

[MOE-13] Moelyohadi, Yopie., dkk. 2013. Pengaruh Kombinasi Pupuk Organik dan Hayati terhadap Pertumbuhan dan Produksi Galur

Jagung (*Zea mays.* L) Hasil Seleksi Efisiensi Hara pada Lahan Kering Marginal. Jurnal Lahan Suboptimal (2):100-110. Universitas Sriwijaya. Palembang.

[MUL-08] Mulyani, Mul Sutedjo. 2008. Pupuk dan Cara Pemupukan. Rineka Cipta. Jakarta.

[NAV-11] Navalertporn, Thitipong dan Afzulpurkar, Nitin. 2011. *Optimization Of Tile Manufacturing Process Using Particle Swarm Optimization. Industrial System Engineering, School of Engineering and Technology, Asian Institute of Technology. Thailand.*

[NUR-08] Nurida, N.L. 2008. Kualitas Limbah Pertanian sebagai Bahan Baku Pembenah berupa Biochar untuk Rehabilitasi Lahan. Prosiding Seminar Nasional dan Dialog Sumberdaya Lahan Pertanian (pp. 209-215). Bogor : Balai Besar Penelitian dan Pengembangan Sumberdaya Lahan Pertanian.

[NUS-07] Nusroh, Zaidatun. 2007. Studi Diversitas Makrofauna Tanah di Bawah Beberapa Tanaman Palawija yang Berbeda di Lahan Kering pada Saat Musim Penghujan, Skripsi Fakultas Pertanian. Universitas Sebelas Maret, Surakarta.

[PUR-08] Purwono dan Rudi Hartono. 2008. Bertanam Jagung Unggul. Penebar Swadaya. Jakarta

[RIA-09] Ria, Evi Zerda. 2009. Analisi dan Penerapan Algoritma *Particle Swarm Optimization* (PSO) pada Optimasi Penjadwalan Sumber Daya Proyek . Institut Teknologi Telkom. Bandung.

[SIT-10] Sitorus, S.R.P dan H.Soewandita. 2010. Rehabilitasi Lahan Terdegradasi melalui Penambahan Kompos Jerami dan Gambut untuk Keperluan Pertanian. Jurnal Tanah dan Iklim (31):27-37.

[SOE-13] Soemarno. 2013. Pupuk dan Pemupukan Ramah Lingkungan, Bahan Kajian Mata Kuliah Manajemen Kesuburan Tanah. Fakultas Pertanian Universitas Brawijaya, Malang.

[SUN-07] Sun, S , Abraham, A, Zhang, G , and Liu, H. *2007.A Particle Swarm Optimization Algorithm for Neighbor Selection in Peer-to-Peer Networks, 6th InterNational Conference on Computer Information Systems and Industrial Management Applications (CISIM'07),China.*

[SUY-10] Suyanto. 2010. Algotitma Optimasi. Graha Ilmu. Delat Buku. Yogyakarta.

[SUT-11] Sutojo., dkk. 2011. Kecerdasan Buatan. ANDI. Yogyakarta.

[SYE-12] Syekhfani. 2012. Rekomendasi Pemupukan Berimbang untuk Budidaya Tanaman Jagung. Fakultas Pertanian Universitas Brawijaya, Malang.

[UTO-12] Utomo, W. Hadi. 2012. Teknologi Biochar untuk Mendukung Pembangunan Berwawasan Lingkungan. Seminar Nasional HITI 2012.

[WAN-12] Wan, Nainfeng., *et a*l. 2012. *A Methodological Approach To Assess The Combinedd Reduction Of Chemical Pesticides And Chemical Fertilizers For Low-carbon Agriculture. Eco-environment Protection Research Institute, Shanghai Academy of Agricultural Science, Shangai Key Laboratory of Protected Horticultural Technology. China.*

[WID-08] Widowati, A.S., dkk. Ketersediaan Hara NPK dengan Biochar pada pertumbuhan Vegetatif Tanaman Jagung. Jurnal Ilmu-Ilmu hayati (*Life Sciences*)/ Lembaga Penelitian dan Pengabdian kepada Masyarakat Universitas Brawijaya. Malang.

[WID-13] Widhiyasa, Arief. 2013. Kajian *Genetic Algorithm*. Insitut Teknologi Bandung. Bandung.

[WIJ-08] Wijaya, K A. 2008. Nutrisi Tanaman sebagai Penentu Kualitas Hasil dan Resisitensi Alami Tanaman. Prestasi Pustaka Publisher. Jakarta.

[WIN-08] Winarso, Sugeng. Kesuburan Tanah Dasar Kesehatan dan Kualitas Tanah. Gaya Media. Yogyakarta.

[YZH-04] Y, Zhang., S, Huang. 2004. *A Novel Multiobjective Particle Swarm Optimization for Buoy-arrangement Design. Shenyang Institute of Automation the Graduate School of Chinese Academy of Science. Senyang, China.*

LAMPIRAN 1

Hasil Wawancara

14 April 2015

1. Dari pemberian pupuk dan bionchar, Apa saja 5 pengaruh pada tanaman yang ingin dioptimalkan?
   Pengaruh pada tanaman yang ingin diamati dan dioptimasi adalah bobot kering tanaman, bobot 1000 butir, diameter tongkol, panjang tongkol dan hasil produksi jagung
2. Kemudian dari 5 pengaruh pada tanaman apa saja 2 yang paling dominan ingin diamati petani dan pakar dan alasannya?
   Yang pertama adalah bobot kering tanaman, karena bobot kering tanaman kita menghitung berapa unsur hara dari pupuk dan tanah yang diserap oleh tanaman. Kemudian, hasil produksi jagung karena produksi jagung yang diharapkan oleh petani sebagai hasil untuk bisa dijual.
3. Mengapa pada bobot buah digunakan 1000 butir?
   Angka 1000 butir hanya dipakai sebagai acuan biasanya 100 butir atau 1000 butir.
4. Berapa rentang minimal dan maksimal dari 5 pengaruh pada tanaman yang dioptimalkan?
   Bobot kering tanaman sekitar 4-5 ton/ha, bobot 1000 butir 230-265 gram, diameter tongkol 3-5 cm, panjang tongkol 13-20 cm dan hasil produksi 6-8 ton/ha.
5. Bagaimana mengukur hasil pengaruh tanaman misalnya hasil produksi jagung 5 ton/ha?
   Setiap $1m^2$ diberikan satu perlakuan. Jika pada perlakuan pertama didapatkan $1m^2$ = 0.5 kg maka untuk 1 ha = 10000 m^2 sehingga didapatkan 10000 m^2 = 5000 kg = 5 ton jadi didapatkan hasil produksi jagung 5 ton/ha.
6. Berapa rentang dosis minimal dan maksimal dari pemberian pupuk dan biochar?
   Pupuk urea 0.177-0.533 ton/ha, SP_{36} 0.0138-0.319 ton/ha, KCL 0.04 ton/ha-0.3 ton/ha dan Biochar sekitar 10-50 ton/ha.

1 November 2015

1. Bagaimana menurut pakar hasil optimasi dosis pupuk dan pengaruh pada tanaman yang dihasilkan oleh sistem? Apakah sistem sudah cukup baik, akurat atau tidak dalam memberikan kombinasi dosis pupuk berdasarkan pengaruh pada tanaman yang dipilih?
   Sistem ini cukup baik dalam meprediksi pengaruh tanaman berdasarkan pupuk yang diberikan. Dalam pemberian pupuk, dosisnya juga masih sedikit boros atau berlebihan. Harapannya sistem ini bisa dikembangkan lebih baik dengan data yang aktual, jenis pupuk yang lain, dan lebih kompleks agar bisa digunakan secara luas sebagai panduan, terimakasih.

LAMPIRAN 2

Data Utama Kombinasi Dosis Pupuk dan pengaruh pada Tanaman

Data utama dari penelitian ini terdiri dari 6 kali perlakuan. Untuk setiap perlakuan dilakukan pada 30 media tanaman sehingga total data yang didapatkan 180 data.Warna merah diberikan pada setiap perlakuan dengan pengaruh pada tanaman yang tertinggi. Berikut adalah keterangan dari masing-masing perlakuan dan data yang didapatkan :

1. K1 perlakuan dengan pemberian urea 0.2 ton/ha, SP_{36} 0.277 ton/ha, KCL 0.2 ton/ha dan biochar 0 ton/ha (tidak diberikan).
2. BK0perlakuan dengan pemberian urea 0.2 ton/ha, SP_{36} 0.277 ton/ha, KCL 0 ton/ha (tidak diberikan) dan biochar 30 ton/ha.
3. BK1/4perlakuan dengan pemberian urea 0.2 ton/ha, SP_{36} 0.277 ton/ha, KCL 0.05 ton/ha dan biochar 30 ton/ha.
4. BK 1/2 perlakuan dengan pemberian urea 0.2 ton/ha, SP_{36} 0.277 ton/ha, KCL 0.1 ton/ha dan biochar 30 ton/ha.
5. BK3/4 perlakuan dengan pemberian urea 0.2 ton/ha, SP_{36} 0.277 ton/ha, KCL 0.15 ton/ha dan biochar 30 ton/ha.
6. BK1 perlakuan dengan pemberian urea 0.2 ton/ha, SP_{36} 0.277 ton/ha, KCL 0.2 ton/ha dan biochar 30 ton/ha.

No	Nama Perlakuan	Jenis Pupuk (Ton/ha)				Pengaruh pertumbuhan dan hasil pada Jagung				
		Urea	SP36	KCL	Biochar Sampah Organik Kota	Bobot kering tanaman (ton/ha)	Bobot 1000 butir (gram)	Diameter Tongkol (cm)	Panjang Tongkol (cm)	Hasil Produksi (ton/ha)
1	K1	0.2	0.277	0.2	0	4.61	251.7	4.99	19.79	5.75
2	K1	0.2	0.277	0.2	0	4.89	225.5	3.99	13.97	5.2
3	K1	0.2	0.277	0.2	0	4.37	237.7	4.74	19.83	5.42
4	K1	0.2	0.277	0.2	0	4.33	226.2	4.78	19.57	5.62
5	K1	0.2	0.277	0.2	0	4.21	229.3	4.72	19.39	5.58
6	K1	0.2	0.277	0.2	0	4.78	236.9	4.7	19.65	5.77
7	K1	0.2	0.277	0.2	0	4.9	237.5	3.93	18.72	5.61
8	K1	0.2	0.277	0.2	0	4.88	243.2	4.53	15.55	5.52
9	K1	0.2	0.277	0.2	0	4.56	250.9	4.77	19.37	5.74

10	K1	0.2	0.277	0.2	0	4.3	250.9	4.79	14.07	5.25
11	K1	0.2	0.277	0.2	0	4.89	240.1	3.9	19.49	5.31
12	K1	0.2	0.277	0.2	0	4.9	229.3	4.75	18.99	5.23
13	K1	0.2	0.277	0.2	0	4.76	224.2	4.57	19.56	5.5
14	K1	0.2	0.277	0.2	0	4.77	228.5	3.98	14.9	5.48
15	K1	0.2	0.277	0.2	0	4.33	239.7	3.85	15.11	5.58
16	K1	0.2	0.277	0.2	0	4.23	240.6	4.87	19.52	5.3
17	K1	0.2	0.277	0.2	0	4.89	241.7	4.82	19.63	5.55
18	K1	0.2	0.277	0.2	0	4.9	237.8	4.84	19.23	5.66
19	K1	0.2	0.277	0.2	0	4.7	229.2	4.59	19.46	5.21
20	K1	0.2	0.277	0.2	0	4.6	219.9	4.77	17.88	5.68
21	K1	0.2	0.277	0.2	0	4.55	225.3	3.93	19.49	5.21
22	K1	0.2	0.277	0.2	0	4.56	224.8	4.98	15.18	5.48
23	K1	0.2	0.277	0.2	0	4.78	249.5	4.71	19.66	5.44
24	K1	0.2	0.277	0.2	0	4.69	248.1	4.92	18.42	5.22
25	K1	0.2	0.277	0.2	0	4.83	240.2	4.83	17.92	5.47
26	K1	0.2	0.277	0.2	0	4.88	239.9	4.96	19.5	5.23
27	K1	0.2	0.277	0.2	0	4.66	250.4	4.78	13.69	5.54
28	K1	0.2	0.277	0.2	0	4.55	251.1	3.99	13.29	5.25
29	K1	0.2	0.277	0.2	0	4.3	252.5	4.69	15.87	5.57
30	K1	0.2	0.277	0.2	0	4.2	248.6	4.56	19.28	5.43
31	BK0	0.2	0.277	0	30	5.58	251.8	5	20.40	6.69
32	BK0	0.2	0.277	0	30	5.77	245	4.87	20.32	6.17
33	BK0	0.2	0.277	0	30	5.34	232	5.03	21.40	5.86
34	BK0	0.2	0.277	0	30	5.62	230.9	4.93	20.24	6.61
35	BK0	0.2	0.277	0	30	5.43	250.3	4.85	20.53	6.07
36	BK0	0.2	0.277	0	30	5.89	233.3	5.04	20.6	5.94
37	BK0	0.2	0.277	0	30	5.75	231	4.95	21.32	6.02
38	BK0	0.2	0.277	0	30	5.22	232	5.07	20.2	5.96
39	BK0	0.2	0.277	0	30	5.15	249.4	4.9	21.28	6.01
40	BK0	0.2	0.277	0	30	5.79	246.5	4.85	20.69	6.08
41	BK0	0.2	0.277	0	30	5.89	239.7	5.07	21.14	5.9
42	BK0	0.2	0.277	0	30	5.66	253.1	5.02	20.31	6.71
43	BK0	0.2	0.277	0	30	5.65	241	4.92	20.74	6.29
44	BK0	0.2	0.277	0	30	5.77	244.7	4.91	20.44	6.53
45	BK0	0.2	0.277	0	30	5.2	233.1	5.06	21.5	6.56
46	BK0	0.2	0.277	0	30	5.3	239.6	4.99	21.1	6.19
47	BK0	0.2	0.277	0	30	5.8	248.6	4.95	20.42	5.94
48	BK0	0.2	0.277	0	30	5.69	247.3	5.09	20.25	6.1
49	BK0	0.2	0.277	0	30	5.65	250.9	4.96	20.5	6.21

50	BK0	0.2	0.277	0	30	5.87	251.7	5.02	21.28	6.24
51	BK0	0.2	0.277	0	30	5.89	248.3	5.04	20.47	6.24
52	BK0	0.2	0.277	0	30	5.39	237.9	4.81	20.52	6.23
53	BK0	0.2	0.277	0	30	5.44	233	5.01	21.02	6.06
54	BK0	0.2	0.277	0	30	5.42	241.8	4.98	20.37	6.57
55	BK0	0.2	0.277	0	30	5.78	250.1	5.05	20.34	6.33
56	BK0	0.2	0.277	0	30	5.57	239.8	4.97	21.18	6.54
57	BK0	0.2	0.277	0	30	5.43	247.8	4.96	20.3	6.25
58	BK0	0.2	0.277	0	30	5.44	233.8	4.89	21.43	6.18
59	BK0	0.2	0.277	0	30	5.21	252	4.93	20.52	6.14
60	BK0	0.2	0.277	0	30	5.3	251.8	5	20.65	6.51
61	BK1/4	0.2	0.277	0.05	30	6	257.1	4.76	19.50	7.44
62	BK1/4	0.2	0.277	0.05	30	6.17	242.2	4.83	20.47	6.28
63	BK1/4	0.2	0.277	0.05	30	5.93	253	4.95	19.28	6.34
64	BK1/4	0.2	0.277	0.05	30	5.98	245.7	4.91	19.43	7.05
65	BK1/4	0.2	0.277	0.05	30	6.11	247.1	4.82	19.3	7.17
66	BK1/4	0.2	0.277	0.05	30	6.23	250.6	4.79	19.52	6.96
67	BK1/4	0.2	0.277	0.05	30	6.09	253.4	4.87	19.32	7.08
68	BK1/4	0.2	0.277	0.05	30	6.01	249.7	4.83	20.2	6.38
69	BK1/4	0.2	0.277	0.05	30	6.2	285.1	4.87	20.3	6.67
70	BK1/4	0.2	0.277	0.05	30	6.03	244.4	4.98	19.94	7.11
71	BK1/4	0.2	0.277	0.05	30	5.98	255	4.8	19.76	7.17
72	BK1/4	0.2	0.277	0.05	30	5.9	259.7	4.9	19.56	7.07
73	BK1/4	0.2	0.277	0.05	30	5.88	249.6	4.86	20.09	6.65
74	BK1/4	0.2	0.277	0.05	30	5.89	251.9	4.82	19.62	6.44
75	BK1/4	0.2	0.277	0.05	30	6.2	253.7	4.86	20.21	6.78
76	BK1/4	0.2	0.277	0.05	30	6.19	257.7	4.98	19.76	6.45
77	BK1/4	0.2	0.277	0.05	30	6.18	248.7	4.71	20.15	6.37
78	BK1/4	0.2	0.277	0.05	30	6.09	249.8	4.87	19.69	6.3
79	BK1/4	0.2	0.277	0.05	30	6.07	247.8	4.82	20.16	6.31
80	BK1/4	0.2	0.277	0.05	30	6	241.2	4.79	19.45	6.45
81	BK1/4	0.2	0.277	0.05	30	5.99	255.1	4.81	19.39	6.86
82	BK1/4	0.2	0.277	0.05	30	5.93	253.4	4.89	19.8	7.22
83	BK1/4	0.2	0.277	0.05	30	5.97	258.7	4.96	19.25	6.56
84	BK1/4	0.2	0.277	0.05	30	5.97	256.4	4.84	19.37	6.97
85	BK1/4	0.2	0.277	0.05	30	5.9	241.5	4.79	20.45	7.59
86	BK1/4	0.2	0.277	0.05	30	6.09	243.2	4.76	19.34	6.64
87	BK1/4	0.2	0.277	0.05	30	6.15	240.9	4.85	19.77	7.02
88	BK1/4	0.2	0.277	0.05	30	5.91	245.2	4.9	19.94	7.43
89	BK1/4	0.2	0.277	0.05	30	5.99	244.1	4.96	19.74	7.14

90	BK1/4	0.2	0.277	0.05	30	6.09	241	4.81	19.7	7.13
91	BK1/2	0.2	0.277	0.1	30	5.41	256.4	4.96	20.22	5.84
92	BK1/2	0.2	0.277	0.1	30	5.39	246.8	4.86	20.40	5.59
93	BK1/2	0.2	0.277	0.1	30	5.39	241.2	4.89	20.75	5.93
94	BK1/2	0.2	0.277	0.1	30	5.41	255.1	4.89	20.4	5.6
95	BK1/2	0.2	0.277	0.1	30	5.37	246.3	4.99	20.44	5.93
96	BK1/2	0.2	0.277	0.1	30	5.4	244.7	4.79	20.65	5.82
97	BK1/2	0.2	0.277	0.1	30	5.42	249.2	4.89	20.35	5.93
98	BK1/2	0.2	0.277	0.1	30	5.37	255.4	4.97	20.25	5.89
99	BK1/2	0.2	0.277	0.1	30	5.39	256.5	4.98	20.31	5.72
100	BK1/2	0.2	0.277	0.1	30	5.35	243.2	4.9	20.38	5.83
101	BK1/2	0.2	0.277	0.1	30	5.37	247.2	4.85	20.43	5.76
102	BK1/2	0.2	0.277	0.1	30	5.38	247.9	4.9	20.61	5.76
103	BK1/2	0.2	0.277	0.1	30	5.32	245.9	4.85	20.2	5.56
104	BK1/2	0.2	0.277	0.1	30	5.4	246.3	4.99	20.67	5.99
105	BK1/2	0.2	0.277	0.1	30	5.42	251.5	4.85	20.35	5.57
106	BK1/2	0.2	0.277	0.1	30	5.38	245	4.92	20.67	5.62
107	BK1/2	0.2	0.277	0.1	30	5.4	250.1	4.91	20.54	5.75
108	BK1/2	0.2	0.277	0.1	30	5.4	247.6	4.94	20.46	5.61
109	BK1/2	0.2	0.277	0.1	30	5.38	248.3	4.95	20.44	5.9
110	BK1/2	0.2	0.277	0.1	30	5.42	252.7	4.83	20.54	5.78
111	BK1/2	0.2	0.277	0.1	30	5.42	240.9	4.81	20.5	5.8
112	BK1/2	0.2	0.277	0.1	30	5.41	243.7	4.85	20.2	5.84
113	BK1/2	0.2	0.277	0.1	30	5.29	243.6	4.78	20.68	5.74
114	BK1/2	0.2	0.277	0.1	30	5.39	243.2	4.86	20.57	5.67
115	BK1/2	0.2	0.277	0.1	30	5.43	254.1	4.85	20.39	5.97
116	BK1/2	0.2	0.277	0.1	30	5.4	257.4	4.94	20.32	5.94
117	BK1/2	0.2	0.277	0.1	30	5.42	244.2	4.96	20.51	5.75
118	BK1/2	0.2	0.277	0.1	30	5.41	245.7	4.92	20.55	5.63
119	BK1/2	0.2	0.277	0.1	30	5.39	241.9	4.89	20.48	5.82
120	BK1/2	0.2	0.277	0.1	30	5.41	251.9	4.89	20.5	5.73
121	BK3/4	0.2	0.277	0.15	30	4.92	255.3	4.9	20.27	7.24
122	BK3/4	0.2	0.277	0.15	30	4.87	250	4.7	20.19	5.7
123	BK3/4	0.2	0.277	0.15	30	5.33	254.9	4.74	19.20	5.83
124	BK3/4	0.2	0.277	0.15	30	4.82	249	4.74	19.31	6.42
125	BK3/4	0.2	0.277	0.15	30	4.89	252.1	4.73	19.75	7.07
126	BK3/4	0.2	0.277	0.15	30	4.9	252.9	4.85	20.27	6.2
127	BK3/4	0.2	0.277	0.15	30	5.23	251.8	4.79	20.08	7.1
128	BK3/4	0.2	0.277	0.15	30	5.3	255	4.83	20.22	5.79
129	BK3/4	0.2	0.277	0.15	30	5.39	257.5	4.92	19.82	5.74

130	BK3/4	0.2	0.277	0.15	30	5.24	253.7	4.79	19.74	5.76
131	BK3/4	0.2	0.277	0.15	30	5.32	252.6	4.73	20.22	7.03
132	BK3/4	0.2	0.277	0.15	30	5.31	258.2	4.69	19.88	5.98
133	BK3/4	0.2	0.277	0.15	30	5.18	257.3	4.9	19.5	6.52
134	BK3/4	0.2	0.277	0.15	30	4.89	255.4	4.91	19.65	5.82
135	BK3/4	0.2	0.277	0.15	30	4.97	250.5	4.69	20.18	5.77
136	BK3/4	0.2	0.277	0.15	30	4.92	257.3	4.81	19.45	5.81
137	BK3/4	0.2	0.277	0.15	30	4.95	256.2	4.79	20.22	6.81
138	BK3/4	0.2	0.277	0.15	30	5.32	256.9	4.85	19.35	6.06
139	BK3/4	0.2	0.277	0.15	30	5.19	255.6	4.7	19.77	6.74
140	BK3/4	0.2	0.277	0.15	30	5.17	252.9	4.69	20.16	5.94
141	BK3/4	0.2	0.277	0.15	30	5.23	251.3	4.88	20.24	6.56
142	BK3/4	0.2	0.277	0.15	30	4.8	250.5	4.7	20.24	5.96
143	BK3/4	0.2	0.277	0.15	30	4.93	246.8	4.71	19.73	5.82
144	BK3/4	0.2	0.277	0.15	30	4.82	250.3	4.71	19.82	7.2
145	BK3/4	0.2	0.277	0.15	30	4.83	251.2	4.8	20.22	6.35
146	BK3/4	0.2	0.277	0.15	30	5.2	251.9	4.85	20.25	6.09
147	BK3/4	0.2	0.277	0.15	30	4.9	250.7	4.78	19.77	5.61
148	BK3/4	0.2	0.277	0.15	30	4.91	257.3	4.85	19.95	7.04
149	BK3/4	0.2	0.277	0.15	30	4.8	252.6	4.74	19.55	5.56
150	BK3/4	0.2	0.277	0.15	30	4.81	254.5	4.7	19.73	6.28
151	BK1	0.2	0.277	0.2	30	4.55	250.5	4.95	19.37	6.25
152	BK1	0.2	0.277	0.2	30	4.47	250	4.77	20.30	6.7
153	BK1	0.2	0.277	0.2	30	4.6	254.9	4.8	20.05	6.74
154	BK1	0.2	0.277	0.2	30	4.43	250.1	4.91	19.81	6.61
155	BK1	0.2	0.277	0.2	30	4.5	249.2	4.88	20.3	6.53
156	BK1	0.2	0.277	0.2	30	4.57	250.9	4.73	20.18	6.77
157	BK1	0.2	0.277	0.2	30	4.42	254.3	4.87	19.57	6.68
158	BK1	0.2	0.277	0.2	30	4.5	253.6	4.78	19.59	6.56
159	BK1	0.2	0.277	0.2	30	4.39	252.3	4.81	19.56	6.51
160	BK1	0.2	0.277	0.2	30	4.49	254.9	4.81	19.61	6.57
161	BK1	0.2	0.277	0.2	30	4.51	249.8	4.72	20.03	6.48
162	BK1	0.2	0.277	0.2	30	4.59	250.7	4.91	20.37	6.53
163	BK1	0.2	0.277	0.2	30	4.58	253.8	4.83	19.56	6.66
164	BK1	0.2	0.277	0.2	30	4.61	251.5	4.8	20.28	6.79
165	BK1	0.2	0.277	0.2	30	4.62	252.9	4.92	20.27	6.57
166	BK1	0.2	0.277	0.2	30	4.39	250.7	4.83	19.96	6.75
167	BK1	0.2	0.277	0.2	30	4.51	250.1	4.96	20.15	6.54
168	BK1	0.2	0.277	0.2	30	4.52	249.9	4.73	19.88	6.76
169	BK1	0.2	0.277	0.2	30	4.6	250.8	4.75	19.55	6.38

170	BK1	0.2	0.277	0.2	30	4.6	253.5	4.75	20.04	6.22
171	BK1	0.2	0.277	0.2	30	4.59	250.9	4.76	19.73	6.49
172	BK1	0.2	0.277	0.2	30	4.58	251.5	4.9	19.6	6.61
173	BK1	0.2	0.277	0.2	30	4.61	252.4	4.83	20.12	6.73
174	BK1	0.2	0.277	0.2	30	4.6	250.7	4.96	19.79	6.67
175	BK1	0.2	0.277	0.2	30	4.55	254.2	4.76	20.28	6.52
176	BK1	0.2	0.277	0.2	30	4.52	251.9	4.93	20.11	6.63
177	BK1	0.2	0.277	0.2	30	4.51	250.6	4.95	19.69	6.64
178	BK1	0.2	0.277	0.2	30	4.59	254	4.82	19.85	6.28
179	BK1	0.2	0.277	0.2	30	4.62	254.6	4.95	19.89	6.34
180	BK1	0.2	0.277	0.2	30	4.59	249.8	4.81	19.81	6.59

LAMPIRAN 3

Iterasi pertama

Bobot *inputlayer*ke *hiddenlayer*

V11	v21	v31	v41	v12	v22	v32	v42	V13	v23	v33	v43	V14	v24	v34	v44
0.129857	-0.911927	-0.895038	0.105887	0.173633	0.455795	-0.182430	-0.661232	0.171860	0.281215	-0.856605	-0.597939	-0.395562	-0.385098	0.788745	-0.993903
0.136160	-0.903198	-0.888736	0.109039	0.184032	0.470197	-0.172031	-0.656032	0.179329	0.291558	-0.849137	-0.594205	-0.390331	-0.377853	0.793976	-0.991287
0.134867	-0.904988	-0.890029	0.108392	0.182244	0.467721	-0.173819	-0.656926	0.177087	0.288454	-0.851378	-0.595326	-0.392829	-0.381314	0.791478	-0.992537
0.142589	-0.894293	-0.890029	0.143142	0.194578	0.484804	-0.173819	-0.601422	0.186382	0.301327	-0.851378	-0.553500	-0.384826	-0.370229	0.791478	-0.956521
0.149490	-0.884735	-0.890029	0.174197	0.206270	0.500996	-0.173819	-0.548811	0.194991	0.313250	-0.851378	-0.514759	-0.377185	-0.359646	0.791478	-0.922137
0.158703	-0.871975	-0.887725	0.215654	0.219732	0.519641	-0.170454	-0.488232	0.206248	0.328841	-0.848564	-0.464103	-0.364714	-0.342375	0.794595	-0.866019
0.166288	-0.861471	-0.885829	0.249785	0.232519	0.537351	-0.167257	-0.430690	0.215996	0.342342	-0.846127	-0.420237	-0.354831	-0.328686	0.797066	-0.821544
0.175376	-0.848883	-0.881285	0.290685	0.245873	0.555847	-0.160579	-0.370595	0.225783	0.355897	-0.841234	-0.376195	-0.345771	-0.316138	0.801596	-0.780775
0.183021	-0.838294	-0.877462	0.325087	0.257696	0.572222	-0.154668	-0.317393	0.234627	0.368146	-0.836811	-0.336396	-0.337648	-0.304887	0.805658	-0.744219
0.192043	-0.825799	-0.870696	0.365685	0.270090	0.589388	-0.145373	-0.261619	0.245376	0.383034	-0.828749	-0.288023	-0.326707	-0.289735	0.813863	-0.694987
0.199396	-0.815616	-0.865182	0.398771	0.280626	0.603980	-0.137470	-0.214207	0.253534	0.394332	-0.822631	-0.251316	-0.319246	-0.279400	0.819459	-0.661410
0.207017	-0.805060	-0.857560	0.433068	0.290887	0.618192	-0.127209	-0.168032	0.262302	0.406477	-0.813863	-0.211857	-0.311004	-0.267986	0.827700	-0.624325

Bobot *hiddenlayer* ke *outputlayer*

W11	W12	W13	W14	W15	W21	W22	W23	W24	W25	W31	W32	W33	W34	W35	W41	W42	W43	W44	W45
0.28272	0.81647	0.52666	0.18164	0.1126	0.7472	0.8269	0.8483	0.6993	0.0434	0.3444	0.3067	0.8545	0.5617	0.4649	0.9195	0.3373	0.6859	0.3582	0.8915
0.27646	0.8668	0.58235	0.23953	0.10427	0.74211	0.86782	0.89356	0.74638	0.03662	0.3389	0.35132	0.90391	0.613	0.45749	0.9140	0.38212	0.73549	0.4098	0.8841
0.28252	0.85696	0.57783	0.22297	0.07988	0.74703	0.85984	0.88989	0.73294	0.01684	0.34424	0.3426	0.8999	0.59832	0.43588	0.9194	0.37336	0.73146	0.39503	0.86238
0.31386	0.89907	0.61865	0.27866	0.09804	0.78266	0.90774	0.93632	0.79629	0.0375	0.37981	0.39041	0.94625	0.66155	0.4565	0.9633	0.43242	0.78872	0.47315	0.88785
0.35003	0.92736	0.65169	0.33035	0.10448	0.82356	0.93972	0.97368	0.85473	0.04478	0.42086	0.42251	0.98374	0.72022	0.46381	1.0144	0.47239	0.83541	0.54619	0.89696
0.39217	0.97509	0.67938	0.37258	0.1629	0.87003	0.99236	1.00422	0.9013	0.10921	0.46844	0.47641	1.01501	0.7679	0.52978	1.0724	0.5381	0.87353	0.60432	0.97737
0.43655	0.99715	0.70841	0.4204	0.17349	0.91868	1.01655	1.03603	0.95372	0.12081	0.51852	0.50131	1.04776	0.82185	0.54173	1.1338	0.56861	0.91366	0.67043	0.99202
0.459	1.03991	0.74097	0.46405	0.17506	0.94264	1.06217	1.07078	1.0003	0.12249	0.54371	0.54928	1.08429	0.87083	0.54349	1.1639	0.62599	0.95735	0.72901	0.99412
0.48026	1.06697	0.76877	0.50638	0.17124	0.96515	1.09083	1.10021	1.04512	0.11845	0.56759	0.57968	1.11552	0.91838	0.5392	1.1927	0.66257	0.99494	0.78625	0.98896
0.48967	1.10473	0.79708	0.546	0.20148	0.97485	1.12977	1.12941	1.08598	0.14963	0.57811	0.62193	1.1472	0.96271	0.57303	1.2050	0.71216	1.03211	0.83826	1.02866
0.49765	1.13398	0.81844	0.58301	0.20035	0.98302	1.15972	1.15129	1.12388	0.14848	0.58703	0.65462	1.17108	1.00408	0.57178	1.2155	0.75071	1.06028	0.88704	1.02718
0.49766	1.16311	0.84685	0.61386	0.21003	0.98304	1.18881	1.17965	1.15468	0.15814	0.58705	0.68709	1.20273	1.03846	0.58256	1.2156	0.78794	1.09657	0.92647	1.03955

Perubahan bobot dari *hiddenlayer* ke *Outputlayer*

Suku perubahan bobot Wjk dengan alpha = 0.64 dan momentum antara (0-1)= 0.4																			
ΔW11	ΔW12	ΔW13	ΔW14	ΔW15	ΔW21	ΔW22	ΔW23	ΔW24	ΔW25	ΔW31	ΔW32	ΔW33	ΔW34	ΔW35	ΔW41	ΔW42	ΔW43	ΔW44	ΔW45
-0.00626	0.050336	0.055688	0.057894	-0.00833	-0.00509	0.040917	0.045268	0.047061	-0.00677	-0.00555	0.044636	0.049383	0.051339	-0.00739	-0.00557	0.044816	0.049581	0.051546	-0.00741
0.006062	-0.00984	-0.00452	-0.01656	-0.02439	0.004916	-0.00798	-0.00367	-0.01343	-0.01978	0.005371	-0.00872	-0.004	-0.01468	-0.02161	0.005397	-0.00876	-0.00402	-0.01475	-0.02172
0.031332	0.042111	0.040821	0.055696	0.01816	0.035638	0.047897	0.04643	0.063349	0.020656	0.035571	0.047808	0.046343	0.06323	0.020617	0.043949	0.059069	0.057259	0.078124	0.025473
0.036178	0.028287	0.033043	0.051691	0.006443	0.040901	0.031979	0.037357	0.058439	0.007284	0.041057	0.032101	0.037499	0.058661	0.007312	0.05112	0.039969	0.046691	0.07304	0.009104
0.042131	0.047728	0.027688	0.042222	0.058414	0.046467	0.05264	0.030537	0.046567	0.064426	0.047581	0.053902	0.031269	0.047683	0.06597	0.058001	0.065706	0.038117	0.058126	0.080418
0.044384	0.022068	0.029025	0.047825	0.010591	0.048649	0.024189	0.031814	0.05242	0.011608	0.050073	0.024897	0.032745	0.053955	0.011948	0.061358	0.030508	0.040125	0.066115	0.014641
0.022453	0.042759	0.032564	0.043652	0.001568	0.023958	0.045626	0.034747	0.046579	0.001673	0.025188	0.047969	0.036531	0.048971	0.001759	0.030129	0.057378	0.043697	0.058577	0.002104
0.02126	0.027061	0.027803	0.042331	-0.00382	0.022511	0.028653	0.029439	0.044822	-0.00404	0.023881	0.030397	0.03123	0.04755	-0.00429	0.028746	0.036589	0.037592	0.057236	-0.00516
0.009402	0.03776	0.028306	0.039614	0.030235	0.009698	0.038946	0.029195	0.040858	0.031184	0.010522	0.042255	0.031676	0.04433	0.033834	0.012347	0.049585	0.037171	0.05202	0.039703
0.007981	0.029245	0.021366	0.037008	-0.00112	0.008173	0.02995	0.021881	0.0379	-0.00115	0.008922	0.032693	0.023885	0.041371	-0.00126	0.010519	0.038546	0.028162	0.048779	-0.00148
1.76E-05	0.029136	0.028403	0.030854	0.009678	1.76E-05	0.029087	0.028356	0.030803	0.009662	1.97E-05	0.03247	0.031653	0.034385	0.010785	2.25E-05	0.03723	0.036294	0.039426	0.012366
-0.00221	0.027503	0.022343	0.034901	0.018466	-0.00219	0.027279	0.02216	0.034615	0.018315	-0.00246	0.030638	0.024889	0.038879	0.020571	-0.00283	0.035285	0.028665	0.044775	0.023691

Perubahan bobot dari *hiddenlayer* ke *InputLayer*

ΔV11	Δv21	Δv31	Δv41	Δv12	Δv22	Δv32	Δv42	ΔV13	Δv23	Δv33	Δv43	Δv14	Δv24	Δv34	Δv44
0.0063	0.00873	0.0063	0.00315	0.0104	0.0144	0.0104	0.0052	0.00747	0.01034	0.00747	0.00373	0.00523	0.00725	0.00523	0.00262
-0.0013	-0.0018	-0.0013	-0.0006	-0.0018	-0.0025	-0.0018	-0.0009	-0.0022	-0.0031	-0.0022	-0.0011	-0.0025	-0.0035	-0.0025	-0.0012
0.00772	0.0107	0	0.03475	0.01233	0.01708	0	0.0555	0.00929	0.01287	0	0.04183	0.008	0.01108	0	0.03602
0.0069	0.00956	0	0.03106	0.01169	0.01619	0	0.05261	0.00861	0.01192	0	0.03874	0.00764	0.01058	0	0.03438
0.00921	0.01276	0.0023	0.04146	0.01346	0.01865	0.00337	0.06058	0.01126	0.01559	0.00281	0.05066	0.01247	0.01727	0.00312	0.05612
0.00758	0.0105	0.0019	0.03413	0.01279	0.01771	0.0032	0.05754	0.00975	0.0135	0.00244	0.04387	0.00988	0.01369	0.00247	0.04447
0.00909	0.01259	0.00454	0.0409	0.01335	0.0185	0.00668	0.0601	0.00979	0.01356	0.00489	0.04404	0.00906	0.01255	0.00453	0.04077
0.00764	0.01059	0.00382	0.0344	0.01182	0.01637	0.00591	0.0532	0.00884	0.01225	0.00442	0.0398	0.00812	0.01125	0.00406	0.03656
0.00902	0.0125	0.00677	0.0406	0.01239	0.01717	0.0093	0.05577	0.01075	0.01489	0.00806	0.04837	0.01094	0.01515	0.00821	0.04923
0.00735	0.01018	0.00551	0.03309	0.01054	0.01459	0.0079	0.04741	0.00816	0.0113	0.00612	0.03671	0.00746	0.01033	0.0056	0.03358
0.00762	0.01056	0.00762	0.0343	0.01026	0.01421	0.01026	0.04617	0.00877	0.01214	0.00877	0.03946	0.00824	0.01141	0.00824	0.03709
0.00765	0.01059	0.00765	0.03442	0.01019	0.01411	0.01019	0.04585	0.00924	0.01279	0.00924	0.04156	0.00903	0.0125	0.00903	0.04062

Iterasi Kedua

Bobot *InputLayer* ke *HiddenLayer*

V11	v21	v31	v41	v12	v22	v32	v42	V13	v23	v33	v43	V14	v24	v34	v44
0.207017	-0.805060	-0.857560	0.433068	0.290887	0.618192	-0.127209	-0.168032	0.262302	0.406477	-0.813863	-0.211857	-0.311004	-0.267986	0.827700	-0.624325
0.215310	-0.793574	-0.849267	0.437214	0.302377	0.634104	-0.115720	-0.162287	0.271866	0.419722	-0.804299	-0.207075	-0.301977	-0.255483	0.836728	-0.619811
0.215240	-0.793671	-0.849337	0.437179	0.302711	0.634567	-0.115386	-0.162120	0.271307	0.418948	-0.804858	-0.207355	-0.302691	-0.256473	0.836013	-0.620168
0.225311	-0.779723	-0.849337	0.482497	0.316942	0.654278	-0.115386	-0.098078	0.283487	0.435817	-0.804858	-0.152546	-0.289993	-0.238886	0.836013	-0.563026
0.234334	-0.767227	-0.849337	0.523100	0.330225	0.672674	-0.115386	-0.038307	0.294561	0.451155	-0.804858	-0.102711	-0.278150	-0.222483	0.836013	-0.509730
0.245960	-0.751124	-0.846431	0.575419	0.345776	0.694212	-0.111498	0.031673	0.309128	0.471330	-0.801217	-0.037159	-0.260051	-0.197416	0.840538	-0.428286
0.255698	-0.737638	-0.843996	0.619239	0.360121	0.714081	-0.107912	0.096227	0.321399	0.488325	-0.798149	0.018060	-0.245569	-0.177358	0.844159	-0.363115
0.266728	-0.722361	-0.838481	0.668875	0.374844	0.734472	-0.100550	0.162481	0.333776	0.505467	-0.791961	0.073755	-0.232008	-0.158577	0.850939	-0.302094
0.276213	-0.709224	-0.833739	0.711556	0.387904	0.752560	-0.094020	0.221252	0.344904	0.520880	-0.786396	0.123833	-0.219837	-0.141720	0.857024	-0.247324
0.287424	-0.693696	-0.825330	0.762008	0.401839	0.771860	-0.083569	0.283958	0.358519	0.539737	-0.776185	0.185100	-0.203988	-0.119769	0.868911	-0.176004
0.296543	-0.681067	-0.818491	0.803041	0.413506	0.788018	-0.074819	0.336458	0.368884	0.554092	-0.768412	0.231741	-0.192873	-0.104374	0.877248	-0.125984
0.306132	-0.667787	-0.808902	0.846191	0.425048	0.804004	-0.063277	0.388398	0.380134	0.569674	-0.757161	0.282368	-0.180830	-0.087695	0.889290	-0.071792

Bobot *hiddenlayer* ke *outputlayer*

W11	W12	W13	W14	W15	W21	W22	W23	W24	W25	W31	W32	W33	W34	W35	W41	W42	W43	W44	W45
0.49766	1.16311	0.84685	0.61386	0.21003	0.98304	1.18881	1.17965	1.15468	0.15814	0.58705	0.68709	1.20273	1.03846	0.58256	1.21555	0.78794	1.09657	0.92647	1.03955
0.49792	1.19907	0.88409	0.65475	0.20692	0.98324	1.21709	1.20894	1.18684	0.1557	0.58728	0.71857	1.23533	1.07425	0.57984	1.2158	0.81964	1.1294	0.96251	1.03681
0.50738	1.19809	0.88426	0.65325	0.18989	0.99067	1.21632	1.20907	1.18566	0.14233	0.59555	0.71771	1.23548	1.07294	0.56495	1.2241	0.81877	1.12955	0.96119	1.02181
0.53067	1.22498	0.9103	0.68502	0.20631	1.01559	1.24509	1.23693	1.21966	0.15991	0.62186	0.74808	1.26489	1.10883	0.5835	1.2589	0.85886	1.16838	1.00857	1.0463
0.55744	1.2441	0.93242	0.71515	0.21315	1.04397	1.26537	1.2604	1.25161	0.16716	0.65202	0.76963	1.28983	1.14278	0.59121	1.2990	0.88755	1.20158	1.05377	1.05656
0.58891	1.27534	0.95174	0.74082	0.26044	1.07635	1.2975	1.28027	1.27802	0.2158	0.68729	0.80463	1.31148	1.17155	0.64419	1.3446	0.93281	1.22957	1.09097	1.12507
0.62283	1.29105	0.97267	0.7704	0.26945	1.1109	1.31351	1.30159	1.30815	0.22498	0.72512	0.82216	1.33483	1.20454	0.65424	1.3937	0.95556	1.25986	1.13378	1.13811
0.64068	1.32083	0.99695	0.79854	0.27093	1.12848	1.34284	1.3255	1.33587	0.22643	0.74488	0.85514	1.3617	1.2357	0.65587	1.4185	0.99687	1.29354	1.17282	1.14016
0.65775	1.34058	1.01842	0.82681	0.26786	1.14512	1.36209	1.34643	1.36342	0.22344	0.76377	0.87698	1.38545	1.26697	0.65248	1.4422	1.02436	1.32343	1.21218	1.13589
0.66531	1.36852	1.041	0.85432	0.2921	1.15225	1.38845	1.36773	1.38938	0.24631	0.77208	0.90771	1.4103	1.29724	0.67914	1.4523	1.06164	1.35356	1.24888	1.16822
0.6716	1.39089	1.05853	0.881	0.29049	1.15813	1.40935	1.38412	1.41431	0.24481	0.77897	0.93222	1.4295	1.32646	0.67739	1.4607	1.09138	1.37687	1.28436	1.16609
0.67108	1.41374	1.0825	0.90408	0.29752	1.15766	1.43005	1.40583	1.43521	0.25117	0.77841	0.95713	1.45564	1.35163	0.68505	1.4600	1.12042	1.40733	1.31369	1.17502

Perubahan bobot dari *hiddenlayer* ke*outputlayer*

Suku perubahan bobot Wjk dengan alpha = 0.64 dan momentum antara (0-1)= 0.4																			
ΔW11	ΔW12	ΔW13	ΔW14	ΔW15	ΔW21	ΔW22	ΔW23	ΔW24	ΔW25	ΔW31	ΔW32	ΔW33	ΔW34	ΔW35	ΔW41	ΔW42	ΔW43	ΔW44	ΔW45
0.000258	0.03596	0.037241	0.040886	-0.00311	0.000203	0.028282	0.029289	0.032156	-0.00244	0.000226	0.031476	0.032597	0.035788	-0.00272	0.000227	0.0317	0.032829	0.036043	-0.00274
0.009462	-0.00098	0.000169	-0.0015	-0.01704	0.007424	-0.00077	0.000132	-0.00118	-0.01337	0.008274	-0.00086	0.000148	-0.00131	-0.0149	0.008335	-0.00087	0.000149	-0.00132	-0.01501
0.023291	0.026886	0.026039	0.031769	0.016422	0.024923	0.02877	0.027864	0.033996	0.017573	0.026311	0.030372	0.029415	0.035888	0.018552	0.034734	0.040095	0.038832	0.047378	0.024491
0.026761	0.019118	0.022126	0.030129	0.006838	0.028384	0.020278	0.023468	0.031956	0.007253	0.030163	0.021548	0.024938	0.033959	0.007707	0.040151	0.028684	0.033196	0.045204	0.010259
0.031475	0.031243	0.019322	0.025672	0.047294	0.032374	0.032136	0.019874	0.026405	0.048644	0.035263	0.035003	0.021648	0.028761	0.052985	0.045601	0.045265	0.027994	0.037193	0.068519
0.033919	0.015713	0.020928	0.029583	0.009006	0.034552	0.016007	0.021319	0.030135	0.009174	0.037835	0.017527	0.023345	0.032998	0.010045	0.04909	0.022741	0.030289	0.042814	0.013034
0.017849	0.029779	0.024274	0.028138	0.001477	0.017582	0.029335	0.023911	0.027718	0.001455	0.019764	0.032975	0.026879	0.031158	0.001635	0.024764	0.041318	0.033679	0.039041	0.002049
0.017074	0.019747	0.021473	0.02827	-0.00307	0.016641	0.019246	0.020928	0.027553	-0.00299	0.018886	0.021841	0.02375	0.031269	-0.00339	0.023769	0.027489	0.029891	0.039354	-0.00427
0.007554	0.02794	0.022582	0.027514	0.024236	0.007125	0.026356	0.021302	0.025955	0.022863	0.00831	0.030737	0.024843	0.030269	0.026663	0.010077	0.037274	0.030127	0.036706	0.032333
0.006291	0.022367	0.017531	0.026678	-0.0016	0.00588	0.020904	0.016384	0.024932	-0.0015	0.006892	0.024502	0.019204	0.029223	-0.00176	0.008367	0.029746	0.023314	0.035479	-0.00213
-0.00052	0.022853	0.023971	0.023078	0.007026	-0.00047	0.020698	0.021711	0.020902	0.006364	-0.00057	0.024917	0.026136	0.025162	0.007661	-0.00066	0.029042	0.030463	0.029327	0.008929
-0.00255	0.022023	0.019261	0.026765	0.013963	-0.00229	0.01979	0.017308	0.024051	0.012548	-0.00277	0.023939	0.020937	0.029094	0.015178	-0.00323	0.027955	0.024449	0.033975	0.017725

Perubahan bobot dari *hiddenlayer* ke *inputlayer*

ΔV11	Δv21	Δv31	Δv41	Δv12	Δv22	Δv32	Δv42	ΔV13	Δv23	Δv33	Δv43	Δv14	Δv24	Δv34	Δv44
0.00829	0.01149	0.00829	0.00415	0.01149	0.01591	0.01149	0.00574	0.00956	0.01325	0.00956	0.00478	0.00903	0.0125	0.00903	0.00451
-7E-05	-1E-04	-7E-05	-4E-05	0.00033	0.00046	0.00033	0.00017	-0.0006	-0.0008	-0.0006	-0.0003	-0.0007	-0.001	-0.0007	-0.0004
0.01007	0.01395	0	0.04532	0.01423	0.01971	0	0.06404	0.01218	0.01687	0	0.05481	0.0127	0.01759	0	0.05714
0.00902	0.0125	0	0.0406	0.01328	0.0184	0	0.05977	0.01107	0.01534	0	0.04983	0.01184	0.0164	0	0.0533
0.01163	0.0161	0.00291	0.05232	0.01555	0.02154	0.00389	0.06998	0.01457	0.02018	0.00364	0.06555	0.0181	0.02507	0.00452	0.08144
0.00974	0.01349	0.00243	0.04382	0.01435	0.01987	0.00359	0.06455	0.01227	0.017	0.00307	0.05522	0.01448	0.02006	0.00362	0.06517
0.01103	0.01528	0.00552	0.04964	0.01472	0.02039	0.00736	0.06625	0.01238	0.01714	0.00619	0.0557	0.01356	0.01878	0.00678	0.06102
0.00948	0.01314	0.00474	0.04268	0.01306	0.01809	0.00653	0.05877	0.01113	0.01541	0.00556	0.05008	0.01217	0.01686	0.00609	0.05477
0.01121	0.01553	0.00841	0.05045	0.01393	0.0193	0.01045	0.06271	0.01361	0.01886	0.01021	0.06127	0.01585	0.02195	0.01189	0.07132
0.00912	0.01263	0.00684	0.04103	0.01167	0.01616	0.00875	0.0525	0.01036	0.01435	0.00777	0.04664	0.01112	0.0154	0.00834	0.05002
0.00959	0.01328	0.00959	0.04315	0.01154	0.01599	0.01154	0.05194	0.01125	0.01558	0.01125	0.05063	0.01204	0.01668	0.01204	0.05419
0.00969	0.01343	0.00969	0.04362	0.01149	0.01591	0.01149	0.0517	0.01177	0.01631	0.01177	0.05298	0.01298	0.01798	0.01298	0.05841