PENERAPAN METODE *BACKPROPAGATION* UNTUK IDENTIFIKASI PENYAKIT PADA CITRA BUAH TANAMAN JERUK

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
mencapai gelar Sarjana Komputer

Disusun oleh:
Fajar Faruq Maulana
NIM: 145150209111009

PROGRAM STUDI INFORMATIKA
PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2016

PENGESAHAN

PENERAPAN METODE *BACKPROPAGATION* UNTUK IDENTIFIKASI PENYAKIT PADA CITRA BUAH TANAMAN JERUK

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Fajar Faruq Maulana
NIM: 145150209111009

Skripsi ini telah diuji dan dinyatakan lulus pada
27 Januari 2016
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Imam Cholissodin, S.Si, M.Kom	Drs. Marji, M.T
NIK: 201201 850719 1 001	NIK: 19670801 199203 1 001

Mengetahui
Ketua Program Studi Informatika

Drs. Marji, M.T
NIK: 19670801 199203 1 001

PERNYATAAN ORISINALITAS

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 disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah 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, 27 Januari 2016

Fajar Faruq Maulana

NIM: 145150209111009

KATA PENGANTAR

Assalamu'alaykum Warohmatullohi Wabarokatuh.

Bismillahirrohmanirrohim, segala puja dan puji syukur atas kehadirat Allah SWT yang senantiasa melimpahkan rahmat dan hidayah-Nya serta telah memberikan kemudahan sehingga penulis mampu menyelesaikan penelitian dengan judul "PENERAPAN METODE *BACKPROPAGATION* UNTUK IDENTIFIKASI PENYAKIT PADA CITRA BUAH TANAMAN JERUK".

Berbagai pihak telah ikut berperan membantu penulis dalam menyelesaikan tugas akhir ini dengan memberikan arahan dan bimbingan serta motivasi.Untuk itu pada kesempatan ini penulis mengucapkan terima kasih yang sebesar – besarnya kepada :

1. Bapak Drs. Marji, M.T, selaku Ketua Program Studi Informatika Universitas Brawijaya Malang dan dosen pembimbing dua yang telah memberikan semangat, dorongan dan motivasi hingga terselesaikannya skripsi ini.
2. Bapak Imam Cholissodin, S.Si, M.Kom selaku dosen pembimbing satu yang telah memberikan bimbingan, dukungan dan arahan dalam penyelesaian skripsi ini.
3. Bapak Purwoto selaku Ketua Kelompok Tani Desa Selorejo Kecamatan Dau Kota Batu yang telah memberikan bimbingan dan pengarahan di lapangan.
4. Kedua orang tua saya tercinta, Ibunda Chorul Waqingatin dan Ayahanda Mardjono yang selalu memberikan do'a yang tiada henti.
5. Teman-teman seperjuangan Prodi Informatika Universitas Brawijaya Malang Jalur SAP angkatan 2014, khususnya Mohammad Saherman, Bangkit Solehudin dan Danu Narendro, terimakasih atas bantuan dan dukungannya selama ini.
6. Seluruh pihak yang telah membantu, baik yang secara langsung atau tidak, sekali lagi saya ucapkan banyak terima kasih.

Demikian laporan ini disusun, dengan harapan dapat memberikan manfaat serta pengetahuan bagi pembaca.

Wassalamu'alaykum Warohmatullohi Wabarokatuh.

Malang, 27 Januari 2016

Penulis

fajarfaruqmaulana@gmail.com

ABSTRAK

Buah tanaman jeruk merupakan salah satu buah yang rentan terkena penyakit. Sehingga pengenalan serangan hama dan penyakit pada tanaman jeruk dibutuhkan pakar atau SDM petani yang telah berpengalaman dan tidak bisa dilakukan pengenalan secara langsung oleh orang awam yang ingin membudidayakan jeruk, biaya penelitian laboratorium untuk melakukan identifikasi penyakit tanaman buah jeruk masih cukup dibilang mahal untuk kalangan petani. Sejalan dengan berkembangnya teknologi informasi saat ini dalam segala bidang, penulis akan menerapkan teknologi informasi dalam bidang pertanian khususnya dalam pengendalian penyakit buah tanaman jeruk. Sehingga dalam kasus ini dibutuhkan sistem bantu sistem cerdas yang dapat mengidentifikasi penyakit melalui citra digital buah tanaman jeruk yang terkena penyakit. Sistem cerdas yang dibuat pada penelitian ini menggunakan algoritma Jaringan Syaraf Tiruan *Backpropagation* dengan data latih serta data uji yang telah dilakukan *preprocessing* citra dengan cara pemotongan pada bagian citra buah jeruk yang berpenyakit selanjutnya dilakukan segmentasi menggunakan metode *otsu*. Skenario pengujian akurasi menghasilkan tingkat akurasi tertinggi sebesar 100% dengan jenis ekstraksi fitur *RGBYUV*, *threshold* otsu +20, dengan data latih 90% dan data uji 10%, serta parameter pada Algoritma *Backpropagation* menggunakan maksimum iterasi 100, dan *learning rate* 0.1.

Kata Kunci : *Backpropagation*, *Otsu,* JST , Pengolahan Citra Digital

ABSTRACT

Fruit citrus is one of the fruits are prone to diseases. So the introduction of pests and diseases of citrus plants needed specialist or HR farmers who are experienced and can not be done introduction directly by ordinary people who want to cultivate oranges, the cost of laboratory research to identify plant disease citrus fruit is still quite arguably expensive for farmers. In line with the development of information technology in all fields, the authors will apply information technology in agriculture, especially in the control of plant disease citrus fruit. So in this case the system needs an intelligent system that can help identify the disease through digital imagery citrus fruit crops affected by the disease. Intelligent system that is made in this study using Backpropagation Neural Network algorithm with training data and test data that has been done by cutting the image preprocessing on the image of the diseased citrus fruit is then performed segmentation using Otsu method. Accuracy test scenarios produce the highest accuracy rate of 100% with this type of feature extraction RGBYUV, +20 Otsu threshold, the training data and test data of 90% to 10%, as well as the parameters of the Backpropagation algorithm using iterative maximum of 100, and the learning rate 0.1.

Keyword : Backpropagation, Otsu, JST , Digital Image Processing

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

1.1 Latar belakang

Jeruk merupakan salah satu komoditas utama hortikultura yang telah lama dikenal dan dikembangkan di Indonesia. Jeruk kaya akan vitamin dan kandungan nutrisi yang sangat diperlukan oleh tubuh sehingga jeruk sampai sekarang masih banyak diminati di Indonesia untuk digunakan dalam sektor pangan maupun usaha. Namun pada saat ini produktivitas jeruk Indonesia mengalami penurunan yang cukup drastis. Menurut data dari Badan Pusat Statistik, Indonesia mengalami penurunan produksi jeruk dari tahun 2010 sampai tahun 2013. Pada tahun 2010, produksi jeruk mencapai 1.937.773 Ton tetapi pada tahun 2013 menurun menjadi 1.548.401 Ton (BPS, 2015).

Menurut Balai Penelitian Tanaman Jeruk dan Buah Subtropika, masalah utama yang menyebabkan rendahnya produktivitas jeruk di Indonesia adalah terjadinya wabah penyakit CVPD (*Citrus Vein Phloem Degeneration*) yang disebabkan oleh bakteri *Liberibacter asiaticus* dan tidak kunjung tuntas pengendaliannya. Penyakit ini ditularkan oleh serangga vector *Diaphorina citri*. Selain itu juga penyakit yang merugikan dalam buah tanaman jeruk adalah embun jelaga yang mengakibatkan buah sulit masak, thrips mengakibatkan penurunan pada kualitas buah serta kutu batok yang mengakibatkan datangnya semut hitam serta mengakibatkan munculnya penyakit lain. Disamping itu sistem usahatani jeruk bervariasi antar petani maupun antar daerah mengakibatkan beragamnya serangan hama dan penyakit lain yang tentunya perlu mendapat perhatian khusus. Untuk mencegah penurunan hasil produksi jeruk diakibatkan hama penyakit diperlukan usaha dalam pengendaliannya (BALIJETSTRO, 2011).

Pengenalan pada serangan hama dan penyakit pada tanaman jeruk dibutuhkan pakar atau SDM petani yang telah berpengalaman dan tidak bisa dilakukan pengenalan secara langsung oleh orang awam yang ingin membudidayakan jeruk, biaya penelitian laboratorium untuk identifikasi penyakit jeruk sendiri sekarang masing dibilang cukup mahal untuk kalangan petani jeruk. Sejalan dengan berkembangnya teknologi informasi saat ini dalam segala bidang, maka penulis akan menerapkan teknologi informasi dalam bidang pertanian khususnya dalam pengendalian penyakit tanaman jeruk. Sehingga dalam kasus ini dibutuhkan alat bantu sistem cerdas yang dapat mengidentifikasi penyakit melalui citra digital buah tanaman jeruk yang rentan terkena penyakit.

Beberapa penelitian sebelumnya yang berkaitan dengan pengolahan citra digital adalah penelitian yang dilakukan oleh Tri Deviasari dengan judul Deteksi Kanker Paru-Paru Dari Citra Foto *Rontgen* Menggunakan Jaringan Syaraf Tiruan *Backpropagation*. Pada penelitian ini dibangun suatu program aplikasi yang dapat mengelompokkan citra foto *rontgen* paru-paru ke dalam kategori normal, kanker paru-paru atau

penyakit paru lain. Proses ini diawali dengan pengolahan citra *cropping*, *resizing*, *median filter, BW labelling* dan ekstraksi fitur menggunakan transformasi *wavelet haar*. Ekstraksi fitur citra foto rontgen menggunakan fitur energi dan koefisien setiap subband yang kemudian dijadikan masukan jaringan saraf tiruan *backpropagation*. Parameter yang digunakan untuk proses pelatihan dan pengujian menggunakan jaringan saraf tiruan *backpropagation* adalah *Hidden layer* sebanyak 10, learning rate 0,1 dan target eror 0,001. Hasil pengujian jaringan saraf tiruan *backpropagation* dengan menggunakan data baru diperoleh tingkat akurasi sebesar 86,67 % dalam mendeteksi keabnormalan dari citra foto rontgen paru (Wulan, et al., 2013).

Penelitian selanjutnya dilakukan oleh Hagiuang Wang,et al melakukan penelitian tentang Pengenalan Citra Digital Pada Penyakit Tanaman Menggunakan Metode *Backpropagation.* Pada penelitian yang dilakukan Hagiuang Wang,et al melakukan diagnosa penyakit tanaman pada anggur serta gandum berdasarkan pengolahan Gambar dan pengenalan pola. Setelah Gambar dilakukan pemrosesan dan dikompresi selanjutnya dilakukan clustering oleh algoritma Kmeans untuk segmentasi citra penyakit. *Backpropagation* (BP) digunakan sebagai pengklasifikasi untuk mengidentifikasi penyakit anggur dan penyakit gandum. Hasil penelitian menunjukkan bahwa identifikasi penyakit dicapai dengan menggunakan jaringan BP secara efektif. Sedangkan dimensi data fitur tidak dikurangi dengan menggunakan *Principal Component Analysis* (PCA), hasil yang optimal didapatkan dari identifikasi penyakit anggur diperoleh akurasi 100% dan prediksi akurasi 97,14%, sedangkan untuk penyakit gandum diperoleh akurasi dan prediksi akurasi 100% (Wang, et al., 2012).

Kemudian penelitian terkait juga dilakukan oleh Kesari Verma,et al melakukan Klasifikasi Citra Digital menggunakan Algoritma *Backpropagation*. Dalam penelitian tersebut telah mengimplementasikan jaringan syaraf tiruan *backpropagation* dalam mengolah dataset citra digital yang kompleks. Algoritma *Backpropagation* merupakan salah satu teknik yang kuat serta efisien untuk pengolahan dataset *numeric* yang besar. Dalam penelitian ini juga membandingkan kinerja algoritma *Backpropagation* dengan algoritma *Machine Learning* seperti *naive bayes classifier*, *decision tree*, *lazy classifier* dan lainnya. Hasilnya algoritma *Backpropagation* adalah salah satu solusi terbaik dalam masalah klasifikasi citra digital dengan menggunakan teknik akurasi *k crossfold* didapatkan tingkat akurasi sebesar 97,02% (Verma, et al., 2014).

Berdasarkan paparan yang telah dijelaskan, penulis mengusulkan penelitian identifikasi peyakit pada citra buah jeruk menggunakan metode yang sesuai dengan penelitian sebelumnya yaitu menggunakan metode *Backpropagation* sehingga penulis mengajukan penelitian yang berjudul "Penerapan Metode *Backpropagation* Untuk Identifikasi Penyakit Pada Citra Buah Jeruk".

2.1 Rumusan Masalah

Rumusan masalah yang dilakukan dalam penelitian adalah :

1. Bagaimana cara mengimplementasikan metode *backpropagation* untuk identifikasi penyakit pada citra buah tanaman jeruk.
2. Bagaimana tingkat akurasi metode *backpropagation* untuk identifikasi penyakit pada citra buah tanaman jeruk.

3.1 Tujuan

Sesuai dengan rumusan masalah yang telah disebutkan tujuan dari penelitian ini adalah :

1. Menerapkan metode *backpropagation* untuk identifikasi penyakit pada citra buah tanaman jeruk.
2. Mengetahui tingkat akurasi metode *backpropagation* untuk identifikasi penyakit pada citra buah tanaman jeruk.

4.1 Manfaat Penelitian

Manfaat yang diharapkan dalam penelitian ini adalah :

1. Bagi petani tanaman jeruk, hasil dari penelitian ini dapat membantu para petani mengetahui penyakit pada tanaman jeruk melalui citra buah.
2. Bagi Peneliti, dapat lebih memahami dan menerapkan Metode *Backpropagation* untuk identifikasi penyakit pada citra buah tanaman jeruk.
3. Membantu mengatasi persoalan keterbatasan pakar penyakit buah jeruk dalam hal sosialisasi pengendalian penyakit buah jeruk.
4. Membantu meningkatkan produktivitas buah jeruk bagi para petani dengan adanya penelitian ini diharapkan dapat mencegah menyebar luasnya penyakit pada tanaman buah jeruk.

5.1 Batasan Masalah

Batasan masalah dalam penelitian ini adalah :

1. Penelitian ini dibuat hanya untuk identifikasi penyakit pada buah tanaman jeruk melalui citra buah.
2. Identifikasi Penyakit hanya dilakukan satu obyek buah saja sesuai hasil *cropping* dan mendeteksi 1 penyakit.

3. Citra buah jeruk yang akan dijadikan data pelatihan dan data uji adalah jeruk terserang penyakit Embun Jelaga dengan buah jeruk hijau,hama Thrips dengan buah jeruk hijau, dan Hama Kutu Batok dengan buah jeruk hijau serta kuning.
4. Jenis buah jeruk yang dijadikan data pelatihan dan data uji adalah jenis buah jeruk manis atau jeruk peras (*Citrus Sinensis Osbeck*).
5. Kamera yang digunakan dalam pengambilan data citra buah jeruk menggunakan Kamera DSLR Nikon D90 lensa AF-S 18 – 105 mm.
6. Pengambilan data citra buah jeruk dilakukan dengan jarak pixel yang bervariasi.
7. Bahasa pemrograman yang digunakan dalam aplikasi ini adalah bahasa pemrograman JAVA desktop.
8. Hasil yang diperoleh tidak dibandingkan dengan metode lain.

6.1 Sistematika Pembahasan

Untuk sitematika penulisan yang diterapkan oleh penulis antara lain :

BAB 1 PENDAHULUAN

Bab ini berisi tentang Latar Belakang, Rumusan Masalah, Batasan Masalah, Mafaat Penelitian, dan Sistematika Penulisan.

BAB 2 LANDASAN KEPUSTAKAAN

Bab ini berisi Pengantar, Materi Kajian Pustaka, dan Dasar-Dasar Teori yang mendukung dalam Pengembangan dan Perancangan Aplikasi "Implementasi Metode *Backpropagation* Untuk Identifikasi Penyakit Pada Citra Buah Tanaman Jeruk".

BAB 3 METODOLOGI

Pada bab ini akan dijelaskan mengenai metodologi yang digunakan dalam "Implementasi Metode *Backpropagation* Untuk Identifikasi Penyakit Pada Citra Buah Tanaman Jeruk".

BAB 4 PERANCANGAN

Dalam bab ini membahas tentang tahapan-tahapan perancangan yang dilakukan dalam penelitian untuk proses "Implementasi Metode *Backpropagation* Untuk Identifikasi Penyakit Pada Citra Buah Tanaman Jeruk".

BAB 5 IMPLEMENTASI

Dalam bab ini akan dijelaskan mengenai "Implementasi Metode *Backpropagation* Untuk Identifikasi Penyakit Pada Citra Buah Tanaman Jeruk".

BAB 6 PENGUJIAN DAN ANALISA

Dalam bab ini akan dijelaskan mengenai pengujian program dan analisa hasil penelitian.

BAB 6 PENUTUP

Memuat kesimpulan yang diperoleh dari pembuatan dan pengujian perangkat lunak yang dikembangkan dalam penelitian ini serta saran untuk pengembangan lebih lanjut.

BAB 2 LANDASAN KEPUSTAKAAN

Pada bab ini membahas dan melakukan landasan pustaka untuk mencapai tujuan dalam penelitian ini yang telah disebutkan pada bab sebelumnya. Kajian pustaka yang dilakukan meliputi buku teks, jurnal serta hasil penelitian yang telah ada. Bab kajian pustaka ini menampilkan penjelasan mengenai tahapan untuk melakukan "Implementasi Metode *Backpropagation* Untuk Identifikasi Penyakit Pada Citra Buah Tanaman Jeruk ".

2.1. Kajian Pustaka

Kajian pustaka pada penelitian ini adalah menerapkan metode pada penelitian sebelumnya yang memiliki kesamaan dalam membahas Pengolahan Citra Digital diantaranya dilakukan oleh Tri Deviasari dengan judul Deteksi Kanker Paru-Paru Dari Citra Foto *Rontgen* Menggunakan Jaringan Syaraf Tiruan *Backpropagation*. Pada penelitian ini dibangun suatu program aplikasi yang dapat mengelompokkan citra foto *rontgen* paru-paru ke dalam kategori normal, kanker paru-paru atau penyakit paru lain. Proses ini diawali dengan pengolahan citra cropping, resizing, median filter, BW labelling dan ekstraksi fitur menggunakan transformasi wavelet haar. Ekstraksi fitur citra foto rontgen menggunakan fitur energi dan koefisien setiap subband yang kemudian dijadikan masukan jaringan saraf tiruan *backpropagation*. Parameter yang digunakan untuk proses pelatihan dan pengujian menggunakan jaringan saraf tiruan *backpropagation* adalah *Hidden layer* sebanyak 10, learning rate 0,1 dan target eror 0,001. Hasil pengujian Jaringan Syaraf Tiruan *Backpropagation* dengan menggunakan data baru diperoleh tingkat akurasi sebesar 86,67 % dalam mendeteksi keabnormalan dari citra foto rontgen paru (Wulan, et al., 2013).

Penelitian selanjutnya dilakukan oleh Hagiuang Wang,et al melakukan penelitian tentang Pengenalan Citra Digital Pada Penyakit Tanaman Menggunakan Metode *Backpropagation.* Pada penelitian yang dilakukan Hagiuang Wang,et al melakukan diagnosa penyakit tanaman pada anggur serta gandum berdasarkan pengolahan Gambar dan pengenalan pola. Setelah Gambar dilakukan pemrosesan dan dikompresi selanjutnya dilakukan clustering oleh algoritma Kmeans untuk segmentasi citra penyakit. *Backpropagation* (BP) digunakan sebagai pengklasifikasi untuk mengidentifikasi penyakit anggur dan penyakit gandum. Hasil penelitian menunjukkan bahwa identifikasi penyakit dicapai dengan menggunakan jaringan BP secara efektif. Sedangkan dimensi data fitur tidak dikurangi dengan menggunakan *Principal Component Analysis* (PCA), hasil yang optimal didapatkan dari identifikasi penyakit anggur diperoleh akurasi 100% dan prediksi akurasi 97,14%, sedangkan untuk penyakit gandum diperoleh akurasi dan prediksi akurasi 100% (Wang, et al., 2012).

Kemudian penelitian terkait juga dilakukan oleh Kesari Verma,et al melakukan Klasifikasi Citra Digital menggunakan Algoritma *Backpropagation*. Algoritma *Backpropagation* merupakan salah satu teknik yang kuat serta efisien untuk pengolahan dataset *numeric* yang besar. Dalam penelitian ini juga membandingkan kinerja Algoritma *Backpropagation* dengan Algoritma *Machine Learning* seperti *Naive Bayes Classifier*, *Decision Tree*, *Lazy Classifier* dengan menggunakan parameter yang sama antara lain jumlah data latih, jumlah data uji, jumlah kelas, jumlah ekstraksi fitur, jumlah input layer, jumlah hidden layer serta *learning parameter*. Hasilnya algoritma *Backpropagation* adalah salah satu solusi terbaik dalam masalah klasifikasi citra digital karena bersifat adaptif dalam melakukan pembagian kategori pada tiap kelas citra digital dengan tingkat akurasi sebesar 97,02% (Verma, et al., 2014).

Yang terkahir penelitian dilakukan oleh Darma Putra dengan judul Binerisasi Citra Tangan menggunakan metode Otsu. Dengan metode otsu mampu memberikan hasil citra tangan biner yang sangat memuaskan dengan membandingkan hasil binerisasi segmentasi menggunakan kalkulasi metode otsu dengan hasil segmentasi menggunakan aplikasi pengolah gambar (Putra, 2004). Untuk lebih detail tentang penelitian sebelumnya dapat dilihat pada Tabel 2.1.

Tabel 2.1 Kajian Pustaka

No.	Judul dan Peneliti	Metode dan Objek	Hasil
1.	**Judul :** Deteksi Kanker Paru-Paru Dari Citra Foto *Rontgen* Menggunakan Jaringan Saraf Tiruan *Backpropagation* **Peneliti :** Tri Deviasari (Universitas Airlangga)	**Metode :** - Backpropagation **Objek :** - Deteksi kanker paru-paru dari citra foto Rontgen	- Mengetahui tingkat akurasi deteksi kanker paru-paru dari citra *Rontgen* didapatkan akurasi 86,67%
2.	**Judul :** Image Recognition of Plant Diseases Based on Backpropagation Networks **Peneliti :** Hagiuang Wang,et al (China Agricultural University)	**Metode :** - Backpropagation **Objek :** - Deteksi penyakit pada tanaman anggur dan gandum	- Mengetahui tingkat kefektifan deteksi penyakit pada tanaman anggur dan gandum didapatkan akurasi 97,47% untuk Anggur dan 100% untuk Gandum
3.	**Judul :** Image Classification using Backpropagation Algorithm **Peneliti :** Kesari Verma,et al (NIT Raipur, India)	**Metode :** - Backpropagation **Objek :**	- Dengan menggunakan teknik perhitungan akurasi *k crossfold* didapatkan metode Backpropagation memiliki

		- Membandingkan Algoritma Backpropagation dengan *Algoritma Machine Learning* dalam pengolahan Citra	tingkat akurasi sebesar 97,2%
4	**Judul :** Binerisasi Citra Tangan Dengan Metode Otsu **Peneliti :** Darma Putra (Jurusan Teknik Elektro, Fakultas Teknik, Universitas Udayana)	**Metode :** - Otsu **Objek :** - Mencari ciri-ciri geometri citra tangan seperti panjang dan lebar setiap jari-jari tangan dan luas area telapak tangan.	- Berdasarkan hasil uji coba, metode Otsu mampu memberikan hasil citra tangan biner yang sangat memuaskan dengan membandingkan hasil binerisasi dengan menggunakan aplikasi pengolah gambar
5	**Judul :** Penerapan Algoritma *Backpropagation* untuk identifikasi penyakit buah tanaman Jeruk **Peneliti :** Fajar Faruq Maulana	**Metode :** - Backpropagation **Objek :** - Deteksi penyakit pada buah tanaman jeruk	- Diharapkan mendapatkan akurasi >=80% pada hasil pengujian

Pada penelitian sebelumnya dapat disimpulkan bahwa melakukan identifikasi citra untuk dikategorikan pada masing-masing kelas memerlukan *preprocessing* pada citra tersebut dan mengolahnya kedalam metode pembelajaran. Metode *backpropagation* sendiri merupakan metode pembelajaran yang bersifat adaptif atau dapat menyesuaikan obyek yang akan dikenali sehingga dengan banyaknya data pembelajaran menjadikan metode *backpropagation* lebih konsisten untuk bekerja lebih baik dalam proses pengenalan suatu obyek. Sehingga dalam penelitian ini diambil metode *backpropagation* untuk mendeteksi penyakit pada citra buah tanaman jeruk. Pada penelitian ini proses tahapan yang harus dilakukan untuk dapat diimplementasikan yaitu tahap perancangan desain sistem ,penulisan kode program, pengujian serta penerapan program.

2.2. Perancangan

2.2.1. Pengertian Perancangan

Perancangan adalah proses untuk mengaplikasikan berbagai macam teknik dan prinsip untuk tujuan pendefenisian secara rinci perangkat lunak, proses atau system agar dapat direalisasikan dalam bentuk fisik. Hasil dari suatu perancangan atau model-model hasil perancangan akan didokumentasikan dalam dokumen seperti SDD (Software Design Dokument), dimana dokumen ini akan mengGambarkan rancanga solusi yang telah dibuat oleh pengembang perangkat lunak (Presman, 2013).

2.2.2. Analisis Kebutuhan

Analisis kebutuhan pada penelitian ini disesuaikan dengan tujuan yang dibahas dalam bab sebelumnya yakni menciptakan "Implementasi Metode *Backpropagation* Untuk Identifikasi Penyakit Pada Citra Buah Tanaman Jeruk"**.**

Dengan analisis kebutuhan memberikan perancang perangkat lunak . Informasi-informasi yang dapat diterjemahkan menjadi arsitektur sistem, antarmuka-antarmuka sistem dan perancangan berperingkat komponen. Terakhir model- model spesifikasi kebutuhan dan spesifikasi-spesifikasi kebutuhan perangkat lunak memungkinkan para pengembang dapat mengoptimalkan fungsionalitas perangkat lunak yang diharapkan (Presman, 2013).

2.3. Penyakit Tanaman Buah Jeruk

2.3.1. Embun Jelaga

Patogen : *Capnodium citri* Berk. & Desm.

Gejala daun, ranting dan buah terserang dilapisi oleh lapisan berwarna hitam. Pada musim kering lapisan ini dapat dikelupas dengan menggunakan tangan dan mudah tersebar oleh angina. Buah yang tertutup lapisan hitam ini biasanya ukurannya lebih kecil dan terlambat matang (masak). Adanya kutu daun jenis aphis *Leurodicus sp*., *Pseudococcus sp.,* Coccos viridis yang mengeluarkan sekresi embun madu merupakan medium yang baik untuk pertumbuhan jamur ini (BALIJETSTRO, 2011).

Gambar 2.1 Serangan Embun Jelaga Pada Buah Jeruk

2.3.2. Kanker

Patogen : *Xanthomonas axonopodis* pv.citri.

Gejala awal berupa bercak putih pada sisi bawah daun yang selanjutnya warna hijau gelap,kadang-kadang berwarna kuning di sepanjang tepinya. Bagian tengah terbentuk gabus warna coklat.Luka terjadi pada bagian atas dan bawah daun.Pada buah ditandai dengan gejala serupa dengan di daun tetapi bagian tepi tidak berwarna kuning.

Gambar 2.2 Serangan Kanker Pada Tanaman Jeruk

(BALIJETSTRO, 2011)

2.4. Hama Tanaman Buah Jeruk

2.4.1 Thrips (*Scirtothrips citri*)

Ordo : Thysanoptera

Famili : Thripidae

Gejala thrips menyerang bagian tangkai dan daun muda mengakibatkan helai daun menebal, kedua sisi daun agak menggulung ke atas dan pertumbuhannya tidak normal. Serangan pada buah terjadi mulai pada fase bunga dan ketika buah manis sangat muda, dengan meninggalkan bekas luka berwarna coklat keabu-abuan yang disertai garis nekrotis di sekeliling luka, tampak di permukaan kulit buah di sekeliling tangkai atau melingkar pada sekeliling kulit buah. Kerusakan serangan hama ini dapat menurunkan kualitas sebesar 30-60%.

Gambar 2.3 Bekas Serangan Trips pada buah

(BALIJETSTRO, 2011)

2.4.2 Kutu Batok Hijau (*Coccus Viridis*)

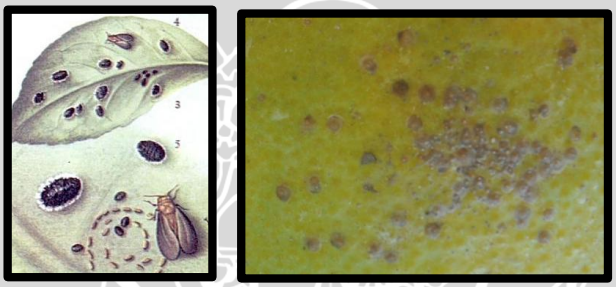
Kebanyakan kutu-kutu ini berada kulit buah, tulang daun dan ranting. Kutu dewasa berwarna hijau, yang tertutup dengan perisai (batok) yang tidak begitu keras. Kutu ini mengundang datangnya semut hitam, sehingga mengakibatkan adanya penyakit lain.

Bagian tanaman yang diserang :

Kutu ini banyak menempel pada kulit buah, tulang-tulang daun, ranting dan pucuk-pucuk tunas baru.

Gejala-gejala dan akibat serangannya :

Aktivitas kutu ini sangat merugikan karena mengeluarkan cairan manis, sehingga mengundang datangnya semut-semut hitam. Sedangkan semut-semut hitam itu sendiri mengeluarkan jelaga hitam yang dapat mengakibatkan lapuk hitam.



Gambar 2.4 Kutu batok pada daun dan kulit buah

(Kanisius, 1994)

2.5. Pengolahan Citra

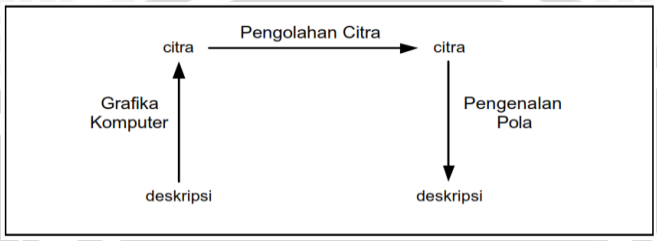
Citra (image) istilah lain untuk Gambar– sebagai salah satu komponen multimedia memegang peranan sangat penting sebagai bentuk informasi visual. Citra mempunyai karakteristik yang tidak dimiliki oleh data teks, yaitu citra kaya dengan informasi. Ada sebuah peribahasa yang berbunyi “sebuah Gambar bermakna lebih dari seribu kata” (*a picture is more than a thousand words*). Maksudnya tentu sebuah Gambar dapat memberikan informasi yang lebih banyak daripada informasi tersebut disajikan dalam bentuk kata-kata (tekstual).

Agar citra yang mengalami gangguan mudah diinterpretasi (baik oleh manusia maupun mesin), maka citra tersebut perlu dimanipulasi menjadi citra lain yang kualitasnya lebih baik. Bidang studi yang menyangkut hal ini adalah pengolahan citra (*image processing*).

Di dalam bidang komputer, sebenarnya ada tiga bidang studi yang berkaitan dengan data citra, namun tujuan ketiganya berbeda, yaitu:

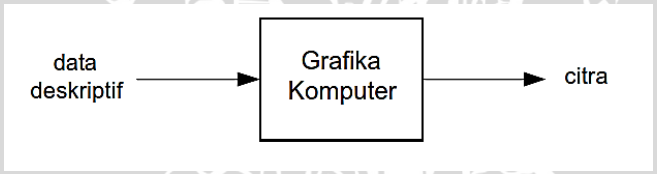
1. Grafika Komputer (*computer graphics*).
2. Pengolahan Citra (*image processing*).
3. Pengenalan Pola (*pattern recognition/image interpretation*).

Hubungan antara ketiga bidang (grafika komputer, pengolahan citra, pengenalan pola) ditunjukkan pada Gambar



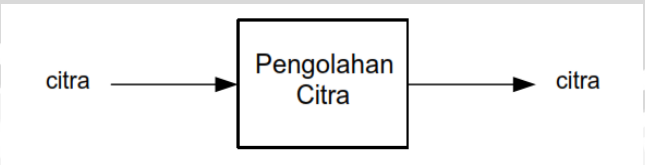
Gambar 2.5 Tiga Bidang Studi Yang Berkaitan Dengan Citra

Grafika Komputer bertujuan menghasilkan citra (lebih tepat disebut grafik atau *picture*) dengan primitif-primitif geometri seperti garis, lingkaran, dan sebagainya. Primitif-primitif geometri tersebut memerlukan data deskriptif untuk melukis elemen-elemen Gambar. Contoh data deskriptif adalah koordinat titik, panjang garis, jari-jari lingkaran, tebal garis, warna, dan sebagainya. Grafika komputer memainkan peranan penting dalam visualisasi dan virtual reality.



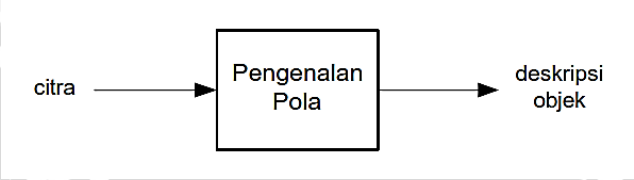
Gambar 2.6 Grafika Komputer Dalam Menghasilkan Citra

Pengolahan Citra bertujuan memperbaiki kualitas citra agar mudah diinterpretasi oleh manusia atau mesin (dalam hal ini komputer). Teknik-teknik pengolahan citra mentransformasikan citra menjadi citra lain. Jadi, masukannya adalah citra dan keluarannya juga citra, namun citra keluaran mempunyai kualitas lebih baik daripada citra masukan. Termasuk ke dalam bidang ini juga adalah pemampatan citra (*image compression*).



Gambar 2. 7 Sistem Kerja Pengolahan Citra

Pengenalan Pola mengelompokkan data numerik dan simbolik (termasuk citra) secara otomatis oleh mesin (dalam hal ini komputer). Tujuan pengelompokan adalah untuk mengenali suatu objek di dalam citra. Manusia bisa mengenali objek yang dilihatnya karena otak manusia telah belajar mengklasifikasi objek-objek di alam sehingga mampu membedakan suatu objek dengan objek lainnya. Kemampuan sistem visual manusia inilah yang dicoba ditiru oleh mesin. Komputer menerima masukan berupa citra objek yang akan diidentifikasi, memproses citra tersebut, dan memberikan keluaran berupa deskripsi objek di dalam citra.



Gambar 2.8 Pengenalan Pola Mengolah Citra

(Ahmad, 2005)

2.5.1. Digitalisasi Citra

Agar dapat diolah dengan dengan komputer digital, maka suatu citra harus direpresentasikan secara numerik dengan nilai-nilai diskrit. Masing-masing elemen pada citra digital (berarti elemen matriks) disebut image element, picture element atau *pixel* atau pel. Jadi, citra yang berukuran N × M mempunyai NM buah *pixel*. Sebagai contoh, misalkan sebuah berukuran 256 × 256 *pixel* dan direpresentasikan secara numerik dengan matriks yang terdiri dari 256 buah baris (di-indeks dari 0 sampai 255) dan 256 buah kolom (di-indeks dari 0 sampai 255) seperti contoh berikut:

$$\begin{bmatrix} 0 & 134 & 145 & \dots & \dots & 231 \\ 0 & 167 & 201 & \dots & \dots & 197 \\ 220 & 187 & 189 & \dots & \dots & 120 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 221 & 219 & 210 & \dots & \dots & 156 \end{bmatrix}$$

Gambar 2.9 Matrix Citra Digital

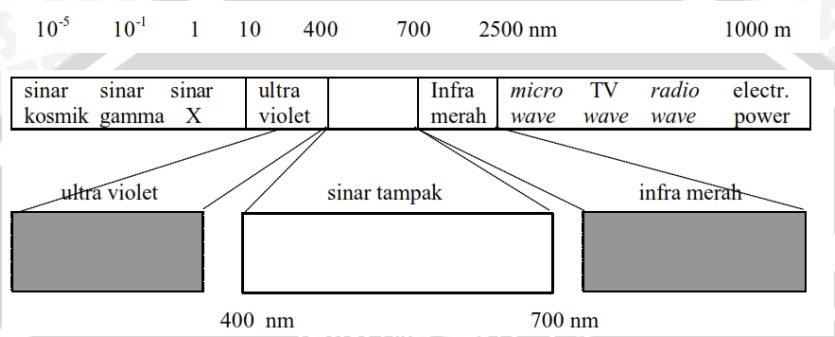
Pixel pertama pada koordinat (0, 0) mempunyai nilai intensitas 0 yang berarti warna *pixel* tersebut hitam, *pixel* kedua pada koordinat (0, 1) mempunyai intensitas 134 yang berarti warnanya antara hitam dan putih, dan seterusnya. Proses digitalisasi citra ada dua macam:

1. Digitalisasi spasial (x, y), sering disebut sebagai penerokan (sampling).
2. Digitalisasi intensitas f(x, y), sering disebut sebagai kuantisasi.

(Ahmad, 2005)

2.5.2. Dasar – Dasar Warna

Warna yang diterima oleh mata dari sebuah objek ditentukan oleh warna sinar yang dipantulkan oleh objek tersebut. Sebagai contoh, suatu objek berwarna hijau karena objek tersebut memantulkan sinar biru dengan panjang gelombang 450 sampai 490 nanometer (nm). Warna sinar yang direspon oleh mata adalah sinar tampak (*visible spectrum*) dengan panjang gelombang berkisar dari 400 (biru) sampai 700 nm (merah).



Gambar 2.10 Spektrum Cahaya

Warna-warna yang diterima oleh mata manusia merupakan hasil kombinasi cahaya dengan panjang gelombang berbeda. Penelitian memperlihatkan bahwa kombinasi warna yang memberikan rentang warna yang paling lebar adalah *red* (R), *green* (G), dan *blue* (B). Ketiga warna tersebut dinamakan warna pokok (primaries), dan sering disingkat sebagai warna dasar *RGB*. Warna-warna lain dapat diperoleh dengan mencampurkan ketiga warna pokok tersebut dengan perbandingan tertentu (meskipun tidak sepenuhnya benar, karena tidak semua kemungkinan warna dapat dihasilkan dengan kombinasi *RGB* saja), sesuai dengan teori Young (1802) yang menyatakan bahwa sembarang warna dapat dihasilkan dari percampuran warna-warna pokok C1, C2, dan C3 dengan persentase tertentu :

$$C = a\,C1 + b\,C2 + c\,C3 \qquad (2.1)$$

Bila citra warna didigitasi, maka tiga buah filter digunakan untuk mengekstraksi intensitas warna merah, hijau, dan biru, dan bila ketiganya dikombinasikan kita memperoleh persepsi warna (Ahmad, 2005).

2.5.3. Konversi Citra Berwarna Ke Citra Berskala Keabuan

Pada pengolahan citra, citra berwarna seringkali diubah menjadi citra berskala keabuan untuk suatu kepentingan tertentu. Konversi dapat dilakukan dengan menggunakan dua cara. Cara pertama adalah dengan menggunakan rumus berikut:

$$R' = G' = B' = \frac{R+G+B}{3} \qquad (2.2)$$

Dalam hal ini, komponen R, G, dan B yang baru (R',G',B') diisi dengan nilai yang sama, yang diperoleh melalui penjumlahan R, G, dan B asli dan kemudian dibagi dengan 3. Cara kedua dilakukan dengan menggunakan rumus berikut:

$$R' = G' = B' = 0.2989R + 0.5870G + 0.1141B \quad (2.3)$$

(Kadir, 2013)

2.5.4. Model Warna *CMY* dan *CMYK*

Warna *cyan* (C) , *magenta* (M), dan *yellow* (Y) adalah warna komplementer terhadap *red*, *green*, dan *blue*. Dua buah warna disebut komplementer jika dicampur dengan perbandingan yang tepat menghasilkan warna putih. Misalnya, magenta jika dicampur dengan perbandingan yang tepat dengan green menghasilkan putih, karena itu magenta adalah komplemen dari dari green.

Model CMY dapat diperoleh dari model *RGB* dengan perhitungan berikut:

$$C = 1 - R \quad (2.4)$$

$$M = 1 - G \quad (2.5)$$

$$Y = 1 - B \quad (2.6)$$

Model CMY dapat digunakan untuk mencetak citra berwarna, tetapi karena ketidaksempurnaan tinta, model CMY tidak dapat menghasilkan warna hitam dengan baik. Karena itu, model CMY disempurnakan menjadi model *CMYK*, yang dalam hal ini K menyatakan warna keempat, dengan perhitungan sebagai berikut :

$$K = \min(C, M, Y) \quad (2.7)$$

$$C = C - K \quad (2.8)$$

$$M = M - K \quad (2.9)$$

$$Y = Y - K \quad (2.10)$$

(Ahmad, 2005)

Konversi *RGB* ke *CMYK* juga dapat dihitung dengan rumus berikut :

$$R' = R/255 \quad (2.11)$$

$$G' = G/255 \quad (2.12)$$

$$B' = B/255 \quad (2.13)$$

$$K = 1 - \max(R', G', B') \quad (2.14)$$

$$C = (1 - R' - K)/(1 - K) \quad (2.15)$$

$$M = (1 - G' - K)/(1 - K) \quad (2.16)$$

$$Y = (1 - B' - K)/(1 - K) \quad (2.17)$$

(Basuki, n.d.)

2.5.5. *YUV Color Model*

YUV, YIQ, YCbCr, YCC, YES merupakan model warna digunakan dalam siaran TV berwarna. YIQ dan *YUV* model analog untuk NTSC dan PAL sistem sementara Model YCbCr adalah standar digital. Sejak peradaban teknologi citra dikenal hanya dapat mengenali dua bentuk Gambar *RGB* dan Gambar hitam dan putih, warna *YUV* Model dikembangkan untuk menyediakan kompatibilitas antara kedua bentuk untuk sistem pertelevisian. Jenis model warna tersebut sangat cocok untuk diterapkan dalam pengolahan citra. Rumus yang digunakan untuk konversi dari *RGB* ke *YUV* adalah sebagai berikut :

$$Y = 0.2999\,R' + 0.587\,G' + 0.114\,B' \quad (2.18)$$

$$U = -0.147\,R' - 0.289\,G' + 0.436\,B' = 0.492\,(B' - Y) \quad (2.19)$$

$$V = 0.615\,R' - 0.515\,G' - 0.100\,B' = 0.877\,(R' - Y) \quad (2.20)$$

Rumus yang digunakan untuk konversi dari *YUV* ke *RGB* adalah sebagai berikut :

$$R' = Y + 1.140\,V \quad (2.21)$$

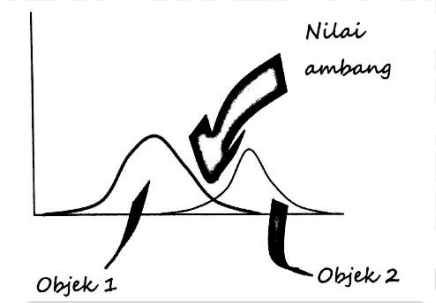
$$G' = Y - 0.395U - 0.581\,V \quad (2.22)$$

$$B' = Y + 2.032\,U \quad (2.23)$$

(Ibrahiem, et al., 2012)

2.5.6. Metode *OTSU*

Metode otsu dipublikasikan oleh Nobuyuki Otsu pada tahun 1979. Metode ini menentukan nilai ambang dengan cara membedakan dua kelompok, yaitu objek dan latarbelakang, yang memiliki bagian yang saling bertumpukan, berdasarkan histogram.



Gambar 2.11 Penentuan nilai ambang untuk memperoleh hasil yang optimal

Prinsip metode otsu adalah sebagai berikut. Pertama-tama, probabilitas nilai intensitas *i* dalam histogram dihitung melalui :

$$p(i) = \frac{n_i}{N}, p(i) \geq 0, \sum_1^{256} p(i) = 1 \quad (2.24)$$

Selanjutnya mencari rerata total $(m_T = \sum_{i=1}^{N} i.p(i))$ (2.25)

dengan n_i menyatakan jumlah *pixel* beritensitas *i* dan *N* menyatakan jumlah semua *pixel* dalam citra. Jika histogram dibagi menjadi dua kelas (objek dan latarbelakang), pembobotan pada kedua kelas dinyatakan sebagai berikut :

$$w_1\,(t) = \sum_{i=1}^{t} p(i) \quad (2.26)$$

$$w_2\,(t) = \sum_{i=t+1}^{L} p(i) = 1 - w_1\,(t) \quad (2.27)$$

Dalam hal ini, L menyatakan jumlah aras keabuan. Rerata kedua kelas dihitung melalui:

$$m_1(t) = \sum_{i=1}^{t} i.p(i)/w_1(t) \quad (2.28)$$

$$m_2(t) = \sum_{i=t+1}^{L} i.p(i)/w_2(t) \quad (2.29)$$

Menghitung BCV (*Between Class Variance)* dinyatakan dengan

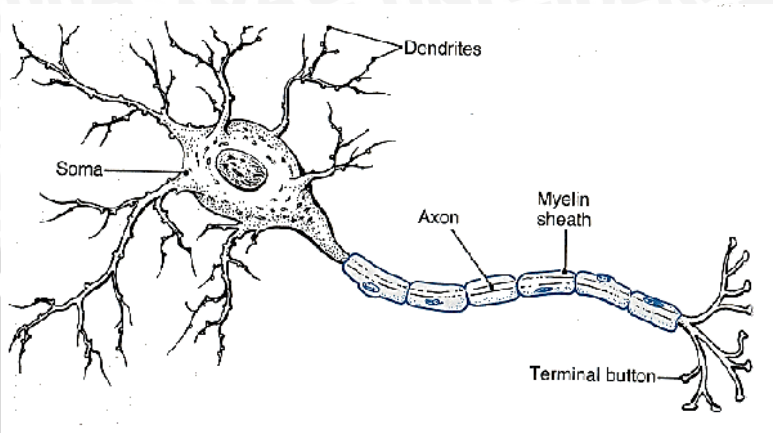
$$\sigma_B{}^2(t) = W_1.[m_1(t) - m_T]^2 + W_2.[m_2(t) - m_T]^2 \quad (2.30)$$

Cara yang terakhir adalah dengan mencari nilai maksimum dari BCV untuk menentukan nilai threshold yang dipakai (Kadir, 2013).

2.6. Jaringan Syaraf Biologi

Jaringan Syaraf biologi merupakan kumpulan dari sel-sel syaraf (*neuron*) yang memiliki tugas untuk mengolah informasi (Puspitaningrum, 2006). Komponen – komponen utama dari sebuah *neuron* dapat dikelompokkan menjadi 3 bagian, yaitu :

1. *Dendrit* yang bertugas untuk menerima informasi.
2. *Badan sel (soma)* yang berfungsi sebagai tempat pengolahan informasi.
3. *Akson (neurit)* yang bertugas mengirimkan impuls-impuls ke sel saraf lainnya.



Gambar 2.12 Sel Syaraf Biologi

Gambar 2.11 menunjukkan sebuah *neuron* menerima impuls-impuls sinyal dari *neuron* yang lain melalui dendrit dan mengirimkan sinyal yang dibangkitkan oleh badan sel melalui akson. Akson dari sel saraf biologi bercabang-cabang dan berhubungan dengan dendrit dari sel saraf lainnya dengan cara mengirimkan impuls melalui sinapsis, yaitu penghubung antara dua buah sel saraf. Sinapsis memiliki kekuatan yang dapat meningkat dan menurun tergantung seberapa besar tingkat propagasi yang diterimanya (Yunita, 2012).

2.6.1. Jaringan Syaraf Tiruan (JST)

Jaringan Syaraf tiruan diperkenalkan pertama kali oleh McCulloch dan Pitts di tahun 1943. McCulloch dan Pitts menyimpulkan bahwa kombinasi beberapa *neuron* sederhana menjadi sebuah sistem *neural* akan meningkatkan kemampuan komputasinya. Bobot dalam jaringan yang diusulkan oleh McCulloch dan Pitts diatur untuk melakukan fungsi logika sederhana. Fungsi aktivasi yang dipakai adalah fungsi *threshold*. Tahun 1949, Hebb mencoba mengkaji proses belajar yang dilakukan oleh *neuron*. Teori ini dikenal sebagai Hebbian Law.Tahun 1958, Rosenblatt memperkenalkan dan mulai mengembangkan model jaringan yang disebut perceptron. Metode pelatihan diperkenalkan untuk mengoptimalkan hasil iterasinya (Siang, 2009).

Tahun 1982, Hopfield telah memperluas aplikasi JST untuk memecahkan masalah-masalah optimasi. Hopfield telah berhasil memperhitungkan fungsi energi ke dalam jaringan syaraf, yaitu agar jaringan memiliki kemampuan mengingat atau memperhitungkan suatu obyek dengan obyek yang pernah dikenal atau diingat sebelumnya (*associative memory*). Konfigurasi jaringan yang demikian dikenal

sebagai *recurrent network*. Salah satu aplikasinya adalah *Travelling Salesman Problem* (TSP) (Puspitaningrum, 2006).

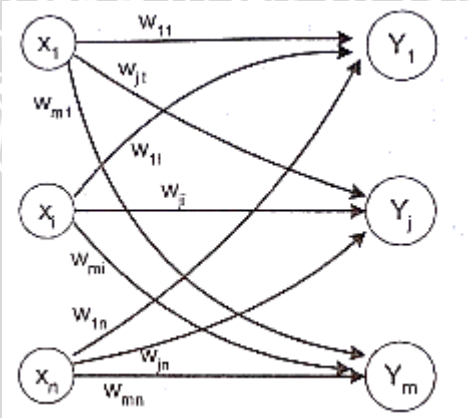
Pada tahun 1986, Rumelhart menciptakan suatu algoritma belajar yang dikenal sebagai propagasi balik *(backpropagation)*. Bila algoritma ini diterapkan pada perceptron yang memiliki lapisan banyak *(multi layer perceptron)*, maka dapat dibuktikan bahwa pemilahan pola-pola yang tidak linier dapat diselesaikan. Perkembangan JST yang ramai dibicarakan sejak tahun 1990-an adalah aplikasi model-model jaringan syaraf tiruan untuk menyelesaikan berbagai masalah di dunia nyata (Siang, 2009).

2.6.2. Arsitektur Jaringan Syaraf Tiruan

Beberapa arsitektur jaringan yang sering dipakai dalam jaringan syaraf tiruan antara lain:

a. Jaringan Layar Tunggal (*single layer network*)

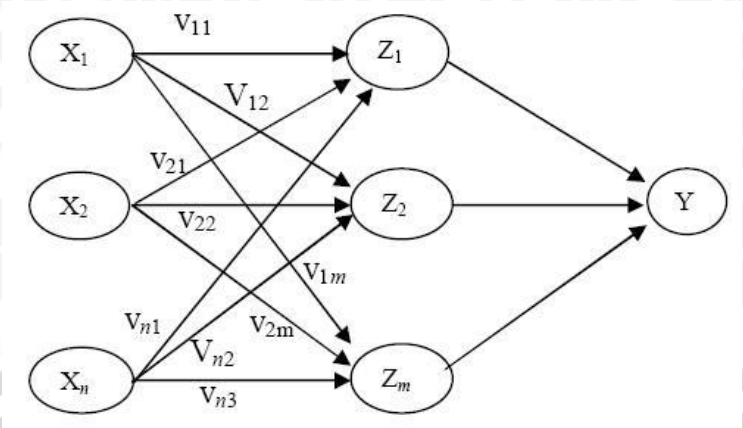
Dalam jaringan ini, sekumpulan *Input neuron* dihubungkan langsung dengan sekumpulan *output*nya. Dalam beberapa model (misal *perceptron*), hanya ada sebuah unit *neuron output* .



Gambar 2.13 Jaringan Layar Tunggal

b. Jaringan Layar Jamak (*multi layer network*)

Jaringan layar jamak merupakan perluasan dari layar tunggal. Dalam jaringan ini, selain unit *Input output*, ada unit-unit lain (sering disebut layar tersembunyi). Dimungkinkan pula ada beberapa layar tersembunyi. Sama seperti pada unit *Input* dan *output*, unit-unit dalam satu layar tidak saling berhubungan.



Gambar 2.14 Jaringan Layar Jamak

c. Jaringan *Reccurent*

Model jaringan *reccurent* mirip dengan jaringan layar tunggal ataupun ganda. Hanya saja, ada *neuron* yang memberikan sinyal pada unit *Input* (sering disebut feedback loop)

(Siang, 2009)

2.6.3. Fungsi Aktivasi

Dalam jaringan syaraf tiruan, fungsi aktivasi dipakai untuk menentukan keluaran suatu *neuron*. Argumen fungsi aktivasi adalah net masukan (kombinasi linier masukan dan bobotnya). Jika net = $\sum x_i w_i$, maka fungsi aktivasinya adalah f(net) = f($\sum x_i w_i$).

Beberapa fungsi aktivasi yang sering dipakai adalah sebagai berikut :

a. Fungsi *threshold* (batas ambang)

$$f(x) = \begin{cases} 1 & jika\ x \geq a \\ 0 & jika\ x < a \end{cases} \qquad (2.31)$$

Untuk beberapa kasus, fungsi *threshold* yang dibuat tidak berharga 0 atau 1, tapi berharga -1 atau 1 (sering disebut *threshold* bipolar).

Jadi

$$f(x) = \begin{cases} 1 & jika\ x \geq a \\ -1 & jika\ x < a \end{cases} \qquad (2.32)$$

b. Fungsi sigmoid

$$f(x) = \frac{1}{1+e^{-x}} \quad (2.33)$$

Fungsi sigmoid sering dipakai karena nilai fungsinya yang terletak antara 0 dan 1 dan dapat diturunkan dengan mudah.

$$f'(x) = f(x)(1 - f(x)) \quad (2.34)$$

c. Fungsi identitas

$$f(x) = x \quad (2.35)$$

Fungsi identitas sering dipakai apabila kita menginginkan keluaran jaringan berupa sembarang bilangan riil (bukan hanya pada range [0,1] atau [-1,1] (Siang, 2009)

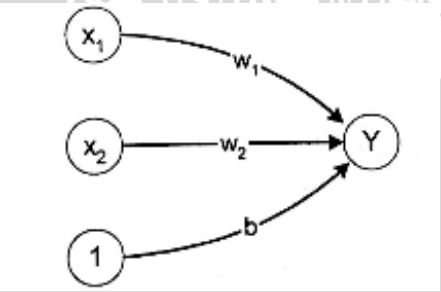
2.6.4. Bias

Dalam suatu jaringan syaraf tiruan hendaknya ditambahkan sebuah unit masukan yang nilainya selalu = 1 untuk mempercepat proses pelatihan . Unit yang sedemikian disebut Bias. Bias dapat dipandang sebagai sebuah *Input* yang nilainya =1. Bias berfungsi untuk mengubah nilai *threshold* menjadi = 0 (bukan = a). Jika melibatkan bias, maka keluaran unit penjumlah adalah :

$$net = b + \sum_i x_i w_i \quad (2.36)$$

Fungsi aktivasi *threshold* menjadi :

$$f(net) = \begin{cases} 1 & jika\ net \geq 0 \\ -1 & jika\ net < 0 \end{cases} \quad (2.37)$$



Gambar 2.15 Bias

(Yunita, 2012)

2.6.5. Laju Pembelajaran/*learning rate* (η)

Learning Rate memiliki pengaruh penting terhadap waktu yang dibutuhkan untuk tercapainya target yang diinginkan. Secara perlahan akan mengoptimalkan nilai perubahan bobot dan menghasilkan *Error* yang lebih kecil (Fajri, 2011).

Jika nilai learning rate yang digunakan terlalu kecil maka terlalu banyak *epoch* yang dibutuhkan untuk mencapai nilai target yang diinginkan, sehingga menyebabkan proses training membutuhkan waktu yang lama. Semakin besar nilai learning rate yang digunakan maka proses pelatihan jaringan akan semakin cepat, namun jika terlalu besar justru akan mengakibatkan jaringan menjadi tidak stabil dan menyebabkan nilai *Error* berulang bolak-balik diantara nilai tertentu, sehingga mencegah *Error* mencapai target yang diharapkan. Oleh karena itu pemilihan nilai variable learning rate harus seoptimal mungkin agar didapatkan proses training yang cepat (Hermawan, 2006).

2.6.6. Standar *Backpropagation*

Kelemahan JST yang terdiri dari layar tunggal membuat perkembangan JST menjadi terhenti pada sekitar tahun 1970 an. Penemuan *backpropagation* yang terdiri dari beberapa layar membuka kembali cakrawala. Terlebih setelah berhasil ditemukannya berbagai aplikasi yang dapat diselesaikan dengan *Backpropagation*, membuat JST semakin diminati orang.

JST dengan layar tunggal memiliki keterbatasan dalam penggenalan pola. Kelemahan ini bisa ditanggulangi dengan menambahkan satu/beberapa layar tersembunyi diantara layar masukan dan keluaran. Meskipun penggunaan lebih dari satu layar tersembunyi memiliki kelebihan manfaat untuk beberapa kasus, tapi pelatihannya memerlukan waktu yang lama. Maka umumnya orang mulai mencoba dengan sebuah layar tersembunyi lebih dahulu.

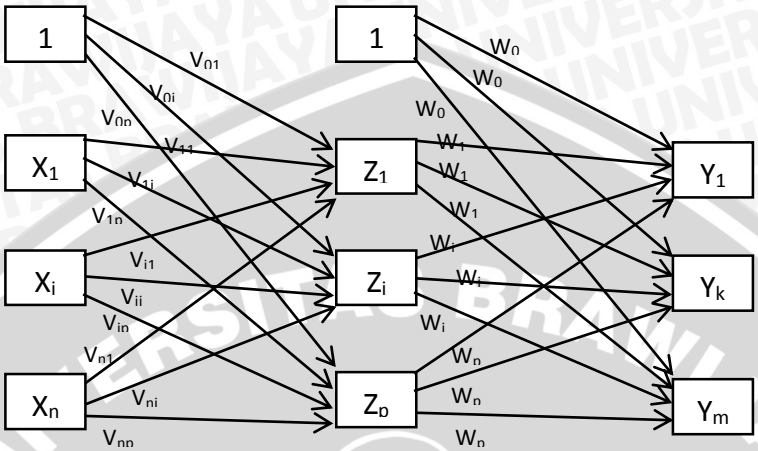
Seperti model JST lain, *Backpropagation* melatih jaringan untuk mendapatkan keseimbangan antara kemampuan jaringan untuk mengenali pola yang digunakan selama pelatihan serta kemampuan jaringan untuk memberikan respon yang benar terhadap pola masukan yang serupa (tapi tidak sama) dengan pola yang dipakai selama pelatihan (Siang, 2009).

2.6.7. Arsitektur *Backpropagation*

Backpropagation memiliki beberapa unit yang ada dalam satu atau lebih layar tersembunyi. Gambar di bawah ini adalah arsitektur *Backpropagation* dengan n buah masukan (ditambah sebuah bias), serta m buah unit keluaran.

V_{ij} merupakan bobot garis dari unit masukan x_i ke unit layar tersembunyi z_j (v_{0j} merupakan bobot garis yang menghubungkan bias di unit masukan ke unit layar

tersembunyi z_i). w_{jk} merupakan bobot dari unit layar tersembunyi z_j ke unit keluaran y_k (w_{0k} merupakan bobot dari bias di layar tersembunyi ke unit keluaran z_k)



Gambar 2.16 Jaringan Syaraf Backpropagation Dengan Satu Lapisan Tersembunyi

(Siang, 2009)

Arsitektur jaringan ini terdiri dari *Input layer, Hidden layer,* dan *output layer* seperti Gambar 2.16 :

1. Lapisan Masukan (*Input Layer*)

   Pada *Input Layer* pada *Backpropagation* terdapat 1 lapisan *layer* yang terdiri dari beberapa *neuron* yang menghubungkan lingkungan luar dengan jaringan syaraf. *Input Layer* merupakan saluran yang menyajikan lingkungan external ke dalam pola jaringan syaraf (Heaton, 2008).

2. Lapisan Tersembunyi (*Hidden Layer*)

   Hidden Layer merupakan bagian yang sangat penting dalam melakukan pemrosesan jaringan syaraf dari *Input layer* menuju *output layer*. Menurut (Yunita, 2012) *Hidden layer* berjumlah minimal 1 lapis yang terdiri dari *neuron-neuron* tersembunyi mulai dari *neuron* tersembunyi pertama sampai *neuron* tersembunyi ke-p.

   Ada beberapa teori yang menjelaskan jumlah *neuron* yang dibutuhkan dalam *Hidden layer*. Menurut (Panchal, et al., 2011) menentukan *Hidden Neuron* dapat menggunakan teori *Hopfiled Neural Network* dimana jumlah *Hidden Neuron* sama dengan Jumlah *Input Neuron*. Selanjutnya menurut (Heaton, 2008) ada beberapa aturan metode untuk mendefinisikan jumlah *neuron* yang digunakan *Hidden Layer* diantaranya :

   a. Jumlah *Hidden Neuron* harus diantara ukuran *Input Layer* dan ukuran *Output Layer.*

b. Jumlah *Hidden Neuron* harus 2/3 dari ukuran *Input Layer* ditambah ukuran *Output Layer* .
c. Jumlah *Hidden Neuron* harus kurang dari dua kali ukuran *Input Layer.*

Aturan – aturan tersebut merupakan pertimbangan dalam menentukan jumlah *Hidden Neuron* dalam Jaringan Syaraf Tiruan. Perlu diketahui bahwa penentuan jumlah *Hidden Neuron* dipengaruhi oleh banyak faktor yaitu :

a. Jumlah *Input* dan *output*
b. Jumlah data training
c. Banyaknya Target
d. Kompleksitas fungsi klasifikasi yang akan dilakukan pembelajaran
e. Algoritma pembelajaran yang digunakan

Sehingga perlu dilakukan *trial and Error* untuk menentukan jumlah *Hidden* Neuron demi mendapatkan hasil yang maksimal (Panchal, et al., 2011).

3. Lapisan Keluaran (*Output Layer*)

Output Layer terdapat 1 lapisan *layer* yang terdiri dari beberapa *neuron* merupakan keluaran yang harus dilakukan dari pemrosesan *Input Layer* dan *Hidden Layer. Output Layer* adalah pola yang sebenarnya diberikan ke dalam lingkungan luar. Pola yang diberikan oleh *Output Layer* dapat langsung ditelusuri kembali ke *Input Layer*. Jumlah *neuron output* harus langsung berhubungan dengan jenis pekerjaan yang jaringan saraf adalah untuk melakukan (Heaton, 2008).

2.6.8. Pelatihan Standar *Backpropagation*

Pelatihan *Backpropagation* meliputi tiga fase. Fase pertama adalah fase maju. Pola masukan dihitung maju mulai dari layar masukan hingga layar keluaran menggunakan fungsi aktivasi yang ditentukan. Fase kedua adalah fase mundur. Selisih antara keluaran jaringan dengan target yang diinginkan merupakan kesalahan yang terjadi. Kesalahan tersebut dipropagasikan mundur, dimulai dari garis yang berhubungan langsung dengan unit-unit di layar keluaran. Fase ketiga adalah modifikasi bobot untuk menurunkan kesalahan yang terjadi.

Algoritma pelatihan untuk jaringan dengan satu layar tersembunyi (dengan fungsi aktivasi *sigmoid biner*) adalah sebagai berikut:

***Langkah* 0** : Inisialisasi semua bobot dengan bilangan acak kecil

***Langkah* 1** : Jika kondisi penghentian belum terpenuhi, lakukan langkah 2-9

***Langkah* 2** : Untuk setiap pasang data pelatihan, lakukan langkah 3-8

Fase 1 :Tahap Perambatan Maju (*Forward Propagation*)

***Langkah* 3** : Tiap unit masukan menerima sinyal dan meneruskannya ke unit tersembunyi di atasnya

***Langkah* 4** : Hitung semua keluaran di unit tersembunyi (Z_i, j=1,2,3,…,p)

$$z\_net_j = v_{j0} + \sum_{i=1}^{n} x_i v_{ji} \quad (2.38)$$

$$z_j = f\left(z\_net_j\right) = \frac{1}{1+ e^{-z\_net_j}} \quad (2.39)$$

Langkah 5 : Hitung semua keluaran jaringan di unit (y_k, k=1,2,3,…,m)

$$y\_net_k = w_{k0} + \sum_{j=1}^{p} z_j w_{kj} \quad (2.40)$$

$$y_k = f\left(y\_net_k\right) = \frac{1}{1+ e^{-y\_net_k}} \quad (2.41)$$

Fase 2 : Tahap Perambatan Balik (*Backpropagation*)

Langkah 6 : Hitung faktor δ unit keluaran berdasarkan kesalahan setiap unit keluaran(y_k, k=1,2,3,…,m) .

$$\delta_k = (t_k - y_k) f'(y\_net_k) = (t_k - y_k) y_k (1 - y_k) \quad (2.42)$$

δ_k merupakan unit kesalahan yang akan dipakai dalam perubahan bobot layar di bawahnya (langkah 7)

Hitung suku perubahan bobot w_{jk} (yang akan dipakai nanti untuk merubah bobot w_{jk}) dengan laju percepatan α

$$\Delta w_{kj} = \alpha\, \delta_k\, z_j \quad ; \quad k = 1, 2, \dots, m \quad ; \quad j = 0, 1, \dots, p \quad (2.43)$$

Langkah 7 : Hitung faktor δ unit tersembunyi berdasarkan kesalahan di setiap unit tersembunyi (z_j, j=1,2,3,…,p)

$$\delta\_net_j = \sum_{k=1}^{m} \delta_k w_{kj} \quad (2.44)$$

Faktor δ unit tersembunyi :

$$\delta_i = \delta\_net_j\, f'\left(z_{net_j}\right) = \delta\_net_j\, z_j\, (1 - z_j) \quad (2.45)$$

Hitung suku perubahan bobot v_{ji} (yang akan dipakai nanti untuk merubah bobot v_{ji})

$$\Delta v_{ji} = \alpha\, \delta_i\, x_i \quad ; \quad j = 1, 2, \dots, p \quad ; \quad i=0,1, \dots, n \quad (2.46)$$

Fase III : Perubahan Bobot

Langkah 8 : Hitung semua perubahan bobot

Perubahan bobot garis yang menuju ke unit keluaran :

$$w_{kj}\ (baru) = w_{kj}(lama) + \Delta w_{kj} \quad (k = 1, 2, \ldots, m ; j=0, 1, \ldots , p) \qquad (2.47)$$

Perubahan bobot garis yang menuju ke unit tersembunyi :

$$v_{ji}\ (baru) = v_{ji}(lama) + \Delta v_{ji} \quad (j = 1, 2, \ldots, m ; i=0, 1, \ldots , p) \qquad (2.48)$$

Langkah 9 **:** Memeriksa *Stop Condition*

Jika *Stop Condition* telah terpenuhi, maka pelatihan jaringan dapat dihentikan, Untuk *Stop Condition*, terdapat dua cara yang dapat dipakai yaitu sebagai berikut :

1. Dengan membatasi iterasi (*epoch*) yang ingin dilakukan. Satu iterasi (*epoch*) adalah perulangan langkah ke-3 sampai dengan ke-8 untuk semua data pelatihan yang ada.
2. Dengan membatasi *Error*. Pada metode *Backpropagation* digunakan *Mean Squared Error* untuk menghitung rata-rata *Error* antara *output* yang dikehendaki pada data pelatihan dan *output* yang dihasilkan oleh jaringan.

Setelah pelatihan selesai dilakukan, didapatkan bobot dan bias yang dapat dipakai untuk perhitungan data pengujian. Dalam hal ini, hanya propses *FeedForward* saja yang digunakan untuk perhitungan terhadap penentuan identifikasi penyakit terhadap data uji.

Apabila fungsi aktivasi yang dipakai bukan sigmoid biner, maka langkah 4 dan 5 harus disesuaikan. Demikian juga turunannya pada langkah 6 dan 7 (Siang, 2009) .

2.6.9. Normalisasi dan Denormalisasi Data

Pada algoritma *backpropagation*, fungsi aktivasi yang dipakai harus memenuhi beberapa syarat yaitu: kontinyu, terdeferensial dengan mudah, dan merupakan fungsi yang tidak turun. Pada penelitian ini digunakan fungsi sigmoid biner, dimana rentang nilai akan berkisar antara 0 dan 1. Data akan dinormalisasi pada interval [0..1], tapi akan lebih baik jika dilakukan normalisasi pada interval yang lebih kecil [0.1..0,9] mengingat fungsi sigmoid merupakan fungsi asimtotik yang nilainya tidak pernah mencapai 0 ataupun 1. Data yang digunakan perlu untuk dinormalisasi sebelum diolah dan akan kembali dinormalisasi setelah didapat hasil dari sistem untuk memperoleh nilai yang sebenarnya. Persamaan normalisasi dan denormalisasi yang akan digunakan didefinisikan berturut-turut pada persamaan 2.39 dan persamaan 2.24 :

$$y = \frac{0.8(x-min)}{\max - min} + 0.1 \qquad (2.49)$$

$$y' = \frac{x'-(0,1)}{0,8}(max - min) + min \quad (2.50)$$

Keterangan :

y = data setelah dinormalisasi

y' = data setelah didenormalisasi

x = data sebelum dinormalisasi

x' = data sebelum didenormalisasi

max = nilai data terbesar

min = nilai data terkecil

(Siang, 2009)

2.6.10. Inisialisasi Bobot dan Bias Awal Secara Random

Pemilihan bobot awal sangat mempengaruhi jaringan syaraf dalam mencapai minimum global (atau mungkin hanya local saja) terhadap nilai *Error*, serta cepat tidaknya proses pelatihan menuju kekonvergenan. Apabila nilai bobot terlalu besar, maka *Input* ke setiap lapisan tersembunyi atau lapisan *output* akan jatuh pada daerah dimana turunan fungsi sigmoidnya akan sangat kecil. Sebaiknya, apabila nilai bobot awal terlalu kecil, maka *Input* ke setiap lapisan tersembunyi atau lapisan *output* akan sangat kecil, yang akan menyebabkan proses pelatihan akan berjalan sangat lambat. Biasanya bobot awal diinisialisasi secara random dengan nilai antara -0.5 sampai 0.5 (atau -1 sampai 1, atau interval yang lainnya) (Kusumadewi, 2004).

2.6.11. Inisialisasi Bobot dan Bias Awal Dengan Metode *Nguyen-widrow*

Metode *Nguyen-widrow* akan menginisialisasi bobot-bobot lapisan dengan nilai antara -0.5 sampai 0.5. Sedangkan bobot-bobot dari lapisan *Input* ke lapisan tersembunyi dirancang sedemikian rupa sehingga dapat meningkatkan kemampuan lapisan tersembunyi dalam melakukan proses pembelajaran.Metode *Nguyen-widrow* secara sederhana dapat diimplementasikan dengan prosedur sebagai berikut :

0. Tetapkan: *n* = jumlah *neuron* (unit) pada lapisan *Input*
   p = jumlah *neuron* (unit) pada lapisan tersembunyi
   β = faktor penskalaan ($\beta = 0.7 * p^{1/n}$) (2. 51)
1. Kerjakan untuk setiap unit pada lapisan tersembunyi (j=1,2,….,p)
   a. Inisialisasi semua bobot dari lapisan *Input* ke lapisan tersembunyi:
      v_{ji} = bilangan random antara -0.5 sampai 0.5 (atau antara –ϒ sampai ϒ).
   b. Hitung : $||v_j||$
   c. Inisialisasi ulang bobot-bobot:

$$v_{ji} = \frac{\beta v_{ji}}{\left||v_j|\right|} \quad (2.52)$$

d. Set Bias :

$b1_j$ = bilangan random antara –β sampai β.

Analisis *Nguyen-widrow* didasarkan atas fungsi aktivasi tangen hiperbolik (Kusumadewi, 2004).

2.6.12. *Mean Square Error* (MSE)

MSE merupakan *Error* rata-rata kuadrat dari selisih antara *output* jaringan dengan *output* target. Tujuan utama adalah memperoleh nilai *Error* sekecil-kecilnya dengan secara iterative mengganti nilai bobot yang terhubung pada semua *neuron* pada jaringan. Untuk mengetahui seberapa banyak bobot yang diganti, setiap iterasi memerlukan perhitungan *Error* yang berasosiasi dengan setiap *neuron* pada *output* dan *Hidden layer*. Rumus perhitungan MSE adalah sebagai berikut (Bayata, et al., 2011) :

$$MSE = \frac{1}{N} \sum_{i=1}^{N} (t_k - y_k)^2 \quad (2.53)$$

Keterangan :

t_k = nilai *output* target

y_k = nilai *output* jaringan

N = jumlah *output* dari *neuron*

2.7. EVALUASI

Pada tahap evaluasi bertujuan untuk mengetahui tingkat akurasi dari hasil penggunaan Metode *Backpropagation* dengan cara menghitung jumlah data uji yang kelasnya diprediksi secara benar. Adapun cara mengukur kinerja dari sistem akan dilakukan evaluasi dengan menggunakan perhitungan presentase pembagian sampel pengujian sebagai berikut :

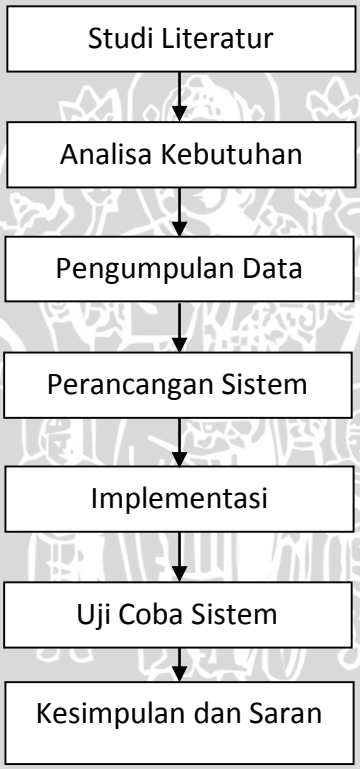
$$Tingkat\ Akurasi\ (\%) = \frac{Jumlah\ Sampel\ Pengujian\ Benar}{Jumlah\ Sampel}\ x\ 100\% \quad (2.54)$$

BAB 3 METODOLOGI

Pembahasan pada bab ini meliputi metodologi yang dilakukan dalam penelitian untuk proses "Implementasi Metode *Backpropagation* Untuk Identifikasi Penyakit Pada Citra Buah Tanaman Jeruk".

3.1. Metode Penelitian

Metode penelitian yang dilakukan terdiri dari Studi Literatur, Analisa Kebutuhan, Pengumpulan Data, Perancangan Sistem, Implementasi, Uji Coba Sistem Serta Kesimpulan dan Saran. Berikut diagram alir dari metode penelitian ditunjukkan pada Gambar 3.1.



Gambar 3.1 Diagram Alir Metodologi Penelitian

3.2. Studi Literatur

Studi literatur merupakan metode yang digunakan untuk mendapatkan dasar teori sebagai sumber acuan untuk penulisan penelitian dan pengembangan aplikasi. Teori dan pustaka yang berkaitan dengan penelitian ini meliputi :

1. Perancangan
2. Penyakit tanaman buah jeruk
3. Hama tanaman buah jeruk
4. Pengolahan Citra
5. Jaringan Syaraf Tiruan
6. *Backpropagation*
7. Evaluasi

3.3. Analisa Kebutuhan

Kegiatan pada analisa kebutuhan adalah dengan menentukan apa saja kebutuhan yang diperlukan dalam "Implementasi Metode *Backpropagation* untuk Deteksi Penyakit Pada Buah Tanaman Jeruk". Analisa kebutuhan pada tahap ini terdiri dari analisa kebutuhan peragkat dan analisa kebutuhan data yang dijelaskan pada Tabel 3.1.

Tabel 3. 1 Analisa Kebutuhan

Kebutuhan Perangkat	**Kebutuhan Data**
1. Kebutuhan *Hardware* untuk pengambilan data citra buah jeruk, meliputi: a. Kamera DSLR Nikon D90 lensa AF-S 18 – 105 mm 2. Kebutuhan *Hardware* untuk keperluan komputasi, meliputi: a. Komputer/laptop Spesifikasi - Sistem Operasi Windows 7 - RAM 3 GB - Processor Pentium D 2.00 GHz 3. Kebutuhan *Software*, meliputi: a. IDE Netbeans 8.0	1. Data masukan, meliputi : Data latih dan data uji yang berupa citra buah jeruk manis yang berpenyakit. Jenis hama dan penyakit antara lain thrips, embun jelaga dank utu batok. Data latih dan data uji terdiri dari 3 macam citra buah yang terkena hama thrips, penyakit embun jelaga dan hama kutu batok 2. Data keluaran, meliputi : Kelas hasil klasifikasi citra buah

3.4. Pengumpulan Data

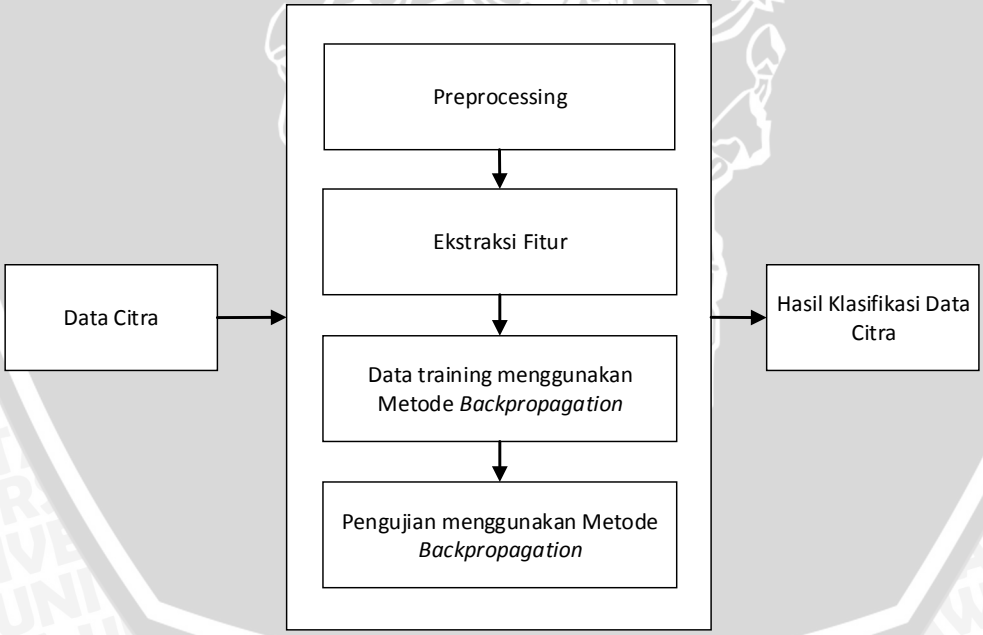
Untuk mendukung keperluan analisa dan perancangan dalam pengembangan penelitian "Implementasi Metode *Backpropagation* untuk Deteksi Penyakit Pada Buah Tanaman Jeruk" diperlukan sejumlah data pendukung yang berasal dari Petani Jeruk di Desa Selorejo Kecamatan Dau Kota Batu. Pengumpulan data tersebut dilakukan dengan cara :

1. Untuk data Primer, dikumpulkan dengan cara sebagai berikut .
   a. Wawancara, yaitu mengadakan Tanya jawab secara langsung dengan personel yang mengetahui tentang obyek yang diteliti.
   b. Observasi (pengamatan), yaitu dengan cara melakukan pencatatan secara cermat dan sistematis, langsung di lokasi obyek penelitian yang berkaitan dengan kegiatan yang dilakukan.
2. Untuk data sekunder, dilakukan dengan membaca atau mempelajari buku teks, literatur, jurnal, artikel, dan lain-lain.

3.5. Perancangan Sistem

Perancangan arsitektur sistem adalah tahap dimana penulis sudah memulai tahap merancang suatu sistem serta menerapkan teori-teori dari pustaka dan data dari sampel digabungkan dengan ilmu yang didapat diimplementasikan untuk merancang serta mengembangkan suatu pendeteksi penyakit buah tanaman jeruk melalui citra digital.

Untuk lebih jelas mengenai alur kerja sistem dibuat diagram blok pada Gambar 3.2 yang menjelaskan hubungan antara *Input*, proses serta *output*.



Gambar 3.2 Diagram Blok Identifikasi Penyakit Pada Buah Tanaman Jeruk

Penjelasan lebih detal pada Gambar 3.2 adalah sebagai berikut:

a. *Input*

   Input berupa data citra buah jeruk yang teridentifikasi penyakit serta hama.

b. Proses

   Proses komputasi pada sistem ini menggunakan *Backpropagation* untuk melakukan identifikasi jenis penyakit pada citra buah jeruk sebelum dilakukan komputasi untuk identifikasi penyakit pada citra buah jeruk dilakukan *preprocessing* serta ekstraksi fitur pada citra buah jeruk.

c. *Output*

   Menampilkan hasil identifikasi penyakit citra buah jeruk dari pengujian metode *Backpropagation.*

3.6. Implementasi

Pada tahap implementasi sistem, akan dilakukan pembuatan sistem yang merujuk pada perancangan siistem yang dilakukan sebelumnya. Implementasi dilakukan dengan bahasa pemrograman Java, basis data menggunakan MySQL , serta *tools* pendukung lainnya.

3.7. Uji Coba Sistem

Uji coba pada penelitian identifikasi penyakit pada citra buah jeruk dilakukan untuk mengetahui parameter apa saja yang mempengaruhi tingkat akurasi pengujian pada jaringan syaraf tiruan. Skenario pengujian yang akan dilakukan antara lain Pengujian Itterasi Maksimum Terhadap Tingkat Akurasi, Pengujian Learning Rate Terhadap Tingkat Akurasi, Pengujian Data Latih Terhadap Tingkat Akurasi, Pengujian Data Uji Terhadap Tingkat Akurasi dan Pengujian *Threshold Otsu* Terhadap Tingkat Akurasi.

3.8. Kesimpulan dan Saran

Kesimpulan dan Saran dilakukan setelah tahap uji coba sistem selesai. Kesimpulan dapat diperoleh berdasarkan hasil data uji yang telah dilakukan dari keseluruhan penelitian. Kemudian dapat disimpulkan saran juga untuk memperbaiki kekurangan yang ada serta pengembangan selanjutnya.

BAB 4 PERANCANGAN

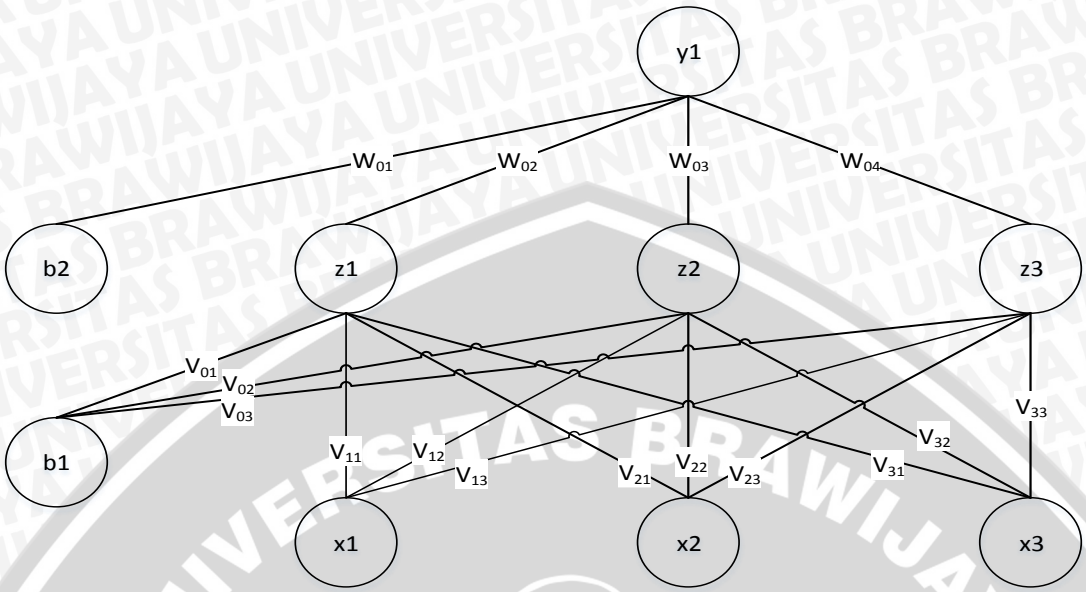
Bab ini akan membahas tentang perancangan sistem yang akan dibuat merupakan identifikasi penyakit serta hama buah jeruk dengan pelatihan menggunakan algoritma *backpropagation*. Terdapat 2 menu utama dalam sistem ini diantaranya pelatihan dan pengujian. Langkah awal yang harus dilakukan adalah proses pelatihan terhadap data sampel yang selanjutnya dijadikan data pengujian. Pada proses pelatihan dan pengujian digunakan data citra buah jeruk yang telah dilakukan preprocessing dan ekstraksi fitur. Data latih dikumpulkan dalam database untuk memudahkan proses pelatihan.

Pada proses awal dilakukan *preprocessing* pada citra buah jeruk yaitu dengan melakukan *cropping* diambil bagian citra yang terkena penyakit, selanjutnya dilakukan segmentasi dengan menggunakan metode *otsu* untuk membagi citra yang terkena penyakit dengan yang tidak dan yang terakhir dilakukan pengambilan ciri dari citra yaitu *red*, *green,* dan *blue*. *Preprocessing* pada citra juga digunakan supaya ciri dari citra diambil tanpa melibatkan latar citra. Kemudian ciri dari citra disimpan dalam Tabel untuk selanjutnya diproses dalam pelatihan serta pengujian.

Setelah diperoleh ekstraksi fitur dari citra, dilakukan pemrosesan untuk mendapatkan kelas dari citra yang terkena penyakit serta hama ataupun yang normal dengan menggunakan algoritma *backpropagation* dengan melakukan pelatihan di awal serta pengujian untuk mendapatkan akurasi setelah memperoleh bobot dengan *Error* terkecil pada proses pelatihan.

4.1. Penentuan Arsitektur Jaringan Syaraf Tiruan

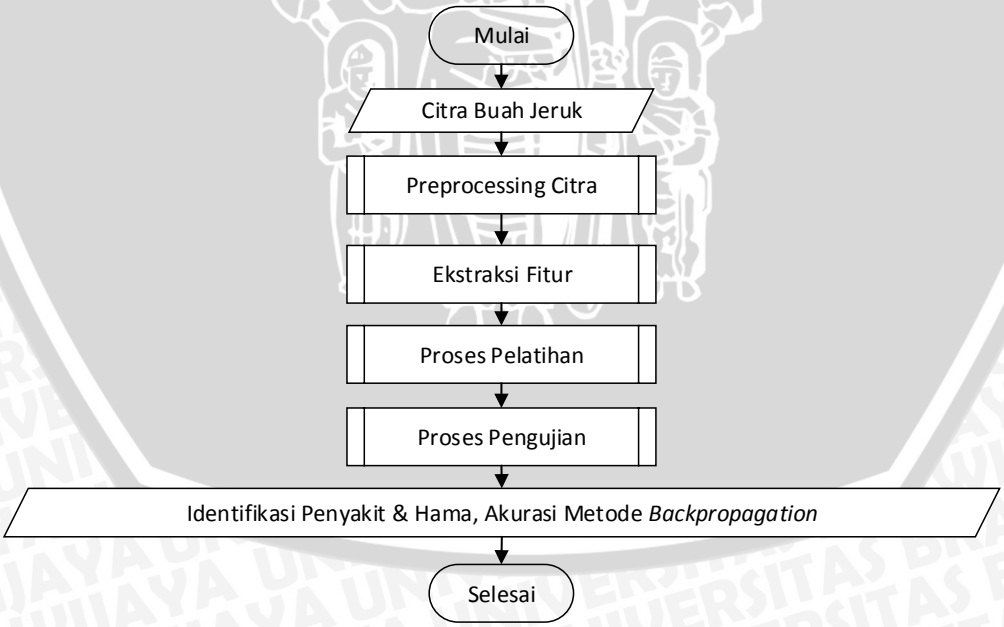
Pada penelitian ini parameter *Input* berupa citra warna *RGB* buah jeruk yang teridentifikasi terkena penyakit serta hama dengan hasil *output* berupa klasifikasi jenis penyakit serta hama dan jeruk yang normal. Nilai X merupakan *Input* warna kombinasi warna *RGB* dari hasil *preprocessing citra* buah jeruk yang memiliki ukuran *pixel* 250 x 250 yang teridentifikasi penyakit serta hama dan jeruk normal sehigga terdapat 3 jenis *Input* antara lain X_1 merupakan warna R(*RED*), X_2 merupakan warna G(*GREEN*), dan X_3 merupakan B(*Blue*). Nilai Y merupakan *output* dari hasil klasifikasi kelas terhadap citra buah jeruk dengan 1 keluaran diasumsikan sebagai Y1. Untuk banyak *neuron* dan lapisan *Hidden layer* digunakan aturan Heaton, jumlah *neuron* pada *Hidden layer* ini 2/3 dari jumlah *Input* ditambah *output* sehingga 3 *neuron* pada *Hidden layer*. Arsitektur jaringan yang digunakan pada penelitian ini ditunjukkan pada Gambar 4.1.



Gambar 4.1 Arsitektur Jaringan Syaraf Tiruan Yang Digunakan

4.2. Diagram Alir Sistem

Perancangan diagram alir sistem menggambarkan alur proses bagaimana sistem bekerja. Dalam sistem pengenalan penyakit serta hama buah jeruk alur yang digunakan ditunjukkan pada Gambar 4.2.



Gambar 4.2 Diagram Alir Sistem

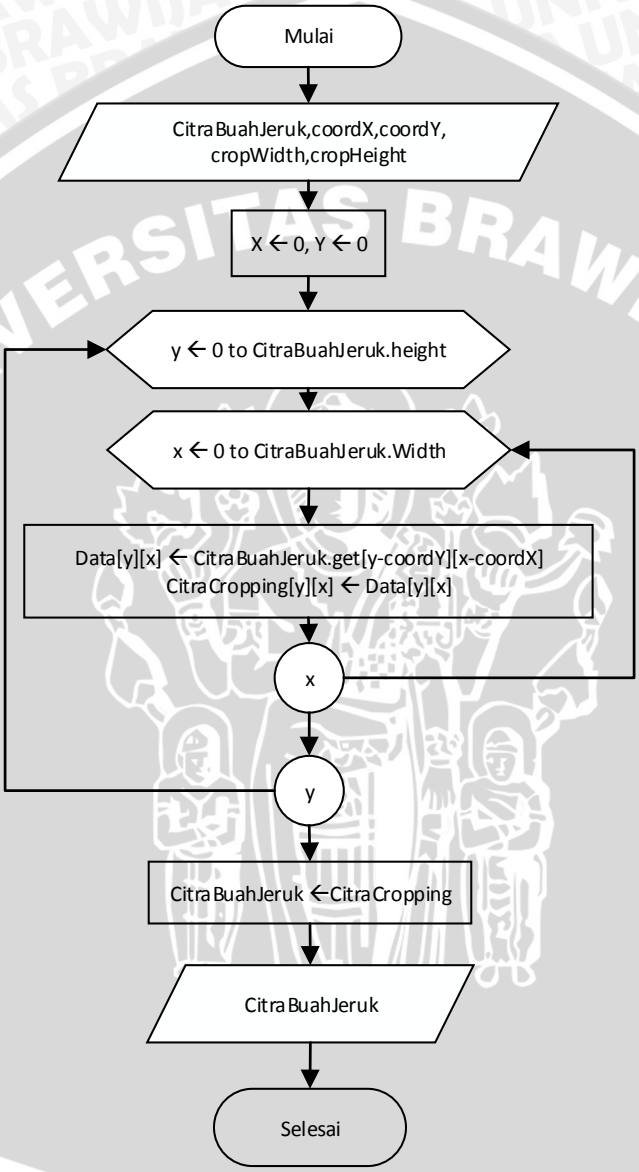
4.2.1. *Preprosessing* Citra dan Ekstraksi Fitur

Pada penelitian ini, tahap *preprocessing* citra menggunakan teknik *cropping* serta segmentasi dengan metode *otsu* untuk mendapatkan bagian citra buah jeruk yang terkena penyakit. Selanjutnya didapat nilai *RGB* masing-masing data citra yang telah dilakukan segmentasi dihitung rata-rata untuk mengambil ekstraksi fitur nilai R, G dan B dan dimasukkan pada variabel *Input* untuk diproses menggunakan metode *backpropagation*, untuk alur prosesnya dapat dilihat pada Gambar 4.3.

Gambar 4.3 *Preprocessing* Citra dan Ekstraksi Fitur

1. *Cropping* Citra Buah Jeruk

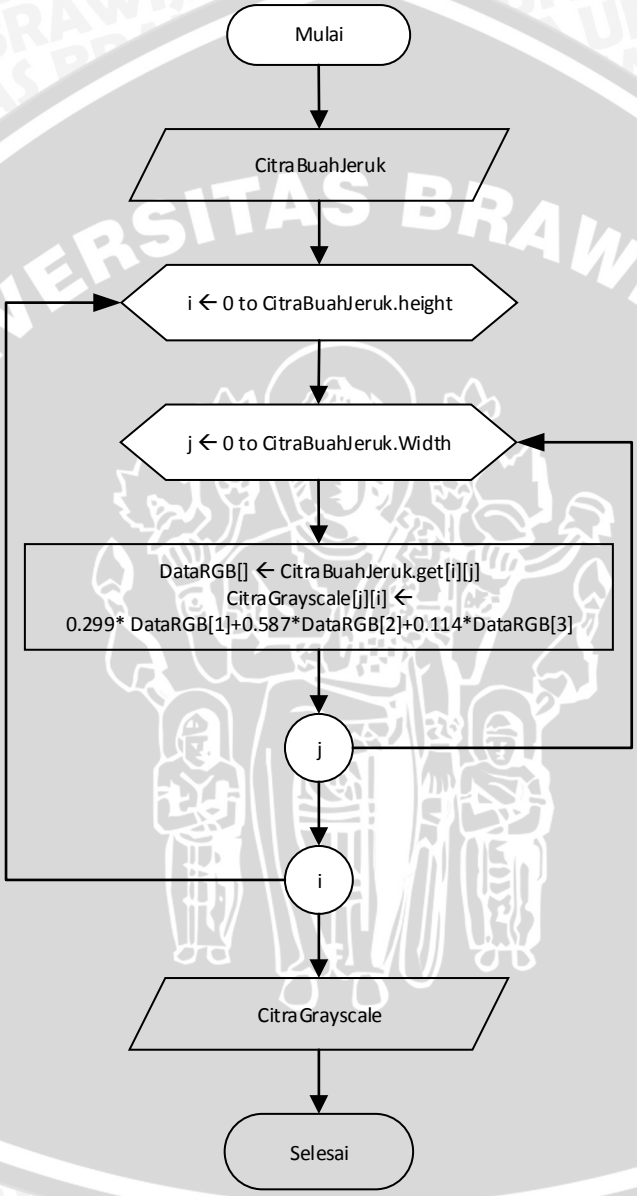
Proses *Cropping* citra buah jeruk merupakan proses pemotongan gambar asli yang diambil bagian citra yang berpenyakit. Pada penelitian ini *cropping* yang disediakan berukuran 32x32 *pixel*, 64x64 *pixel*, 128x128 *pixel* dan 256 x 256 *pixel*. Alur algoritma *cropping* ditunjukkan pada Gambar 4.4.



Gambar 4.4 *Cropping* Citra Buah Jeruk

2. Konversi Data Citra ke *Grayscale*

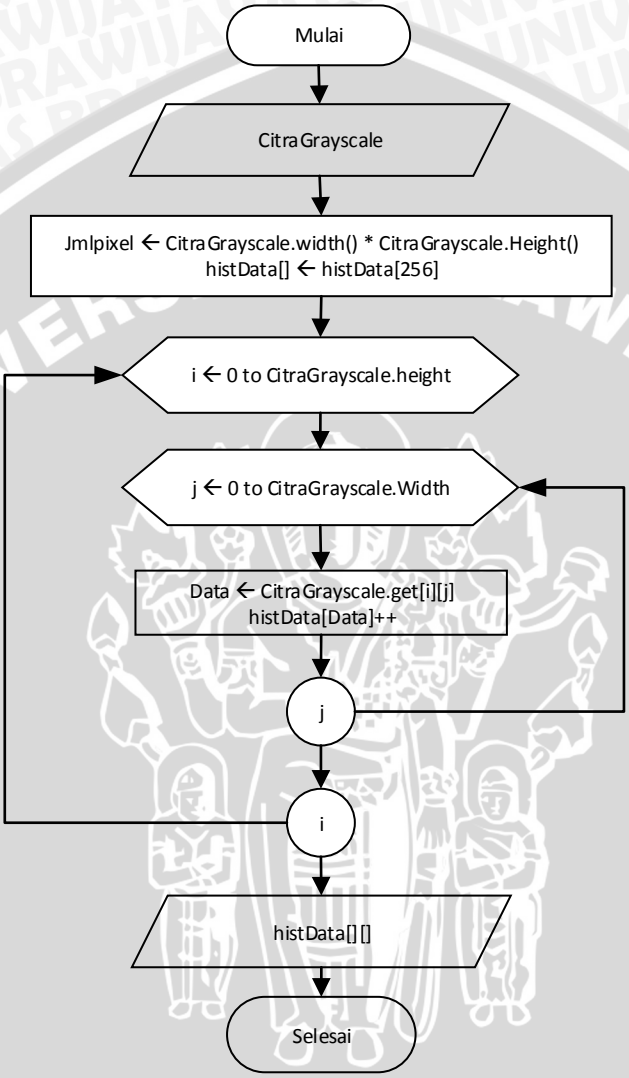
Proses konversi data citra ke *Grayscale* merupakan langkah yang digunakan untuk mengubah citra asli ke dalam level keabuan menggunakan rumus 2.3. Alur konversi data citra ke *Grayscale* ditunjukkan pada Gambar 4.5.



Gambar 4.5 Konversi Data Citra ke *Grayscale*

3. Perhitungan nilai histogram *Grayscale*

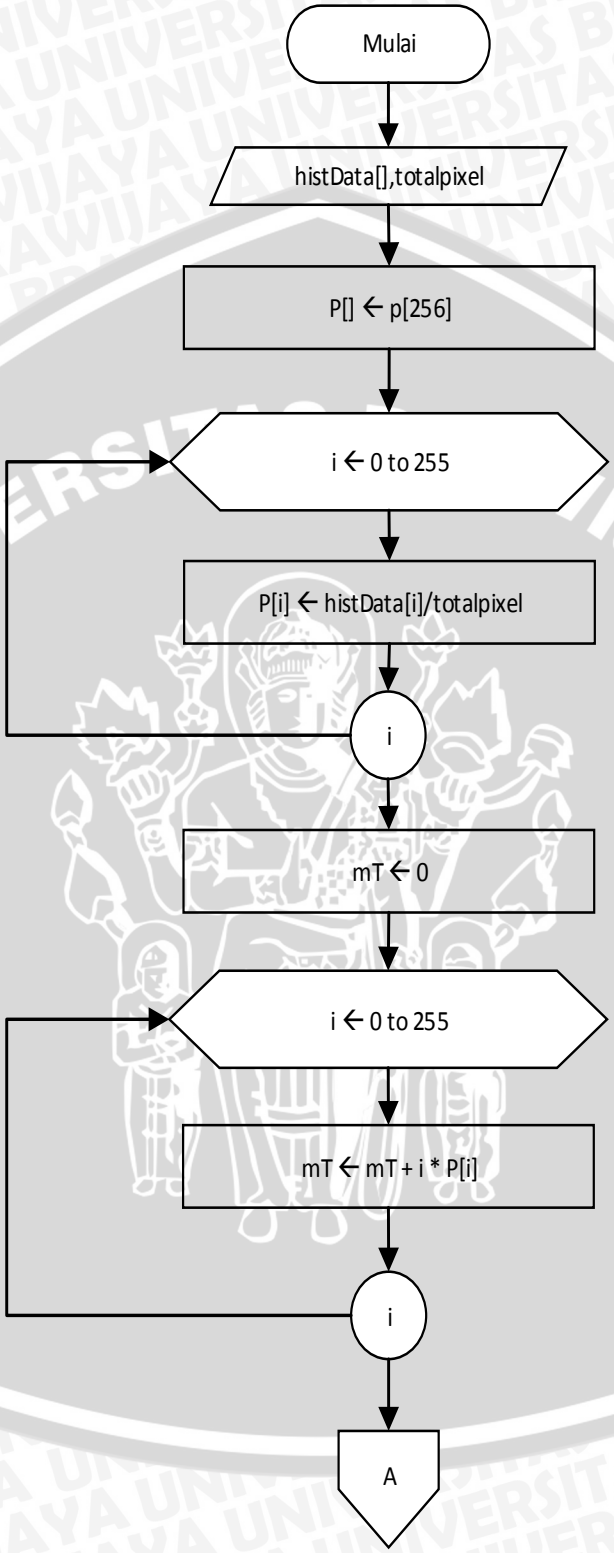
Proses perhitungan nilai histogram *Grayscale* merupakan proses menghitung jumlah intensitas masing-masing *pixel* pada *Grayscale* sesuai range derajat keabuan. Proses perhitungan nilai histogram *Grayscale* ditunjukkan pada Gambar 4.6.

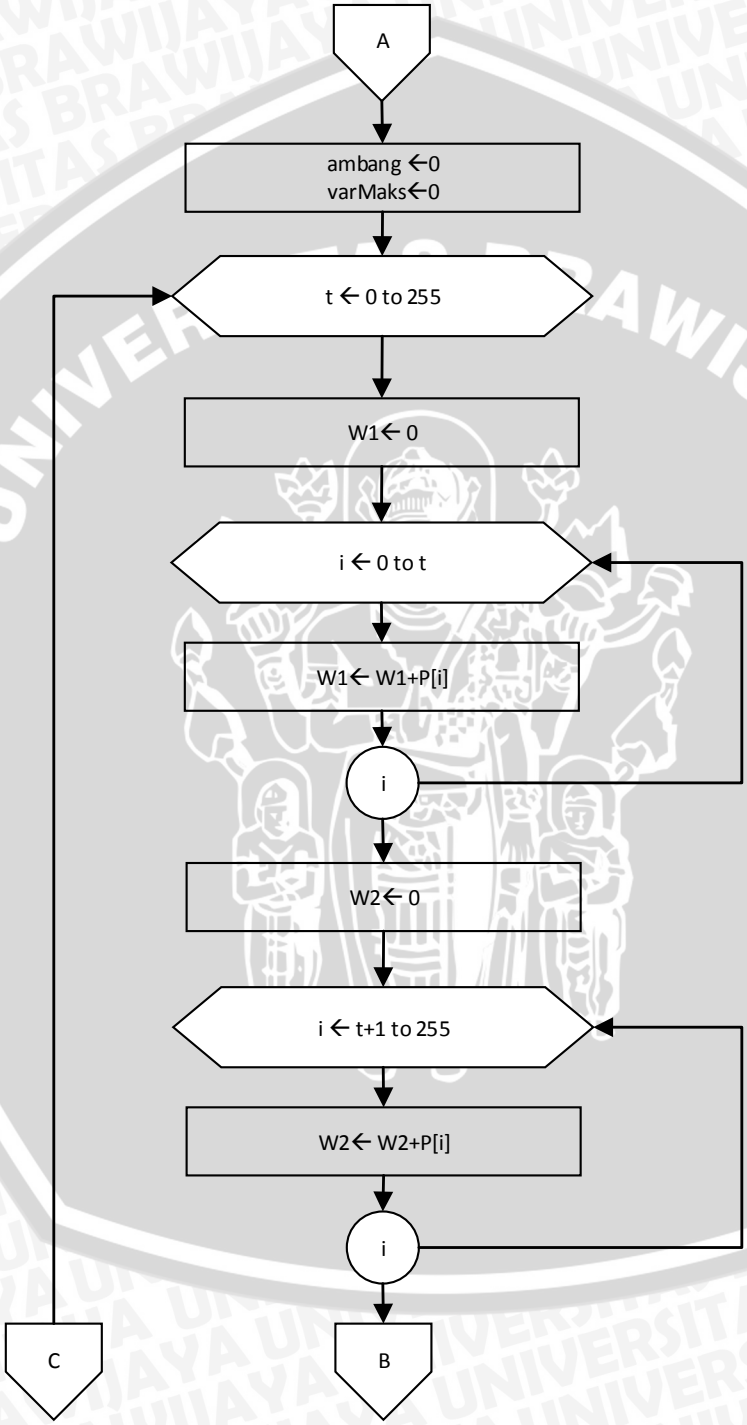


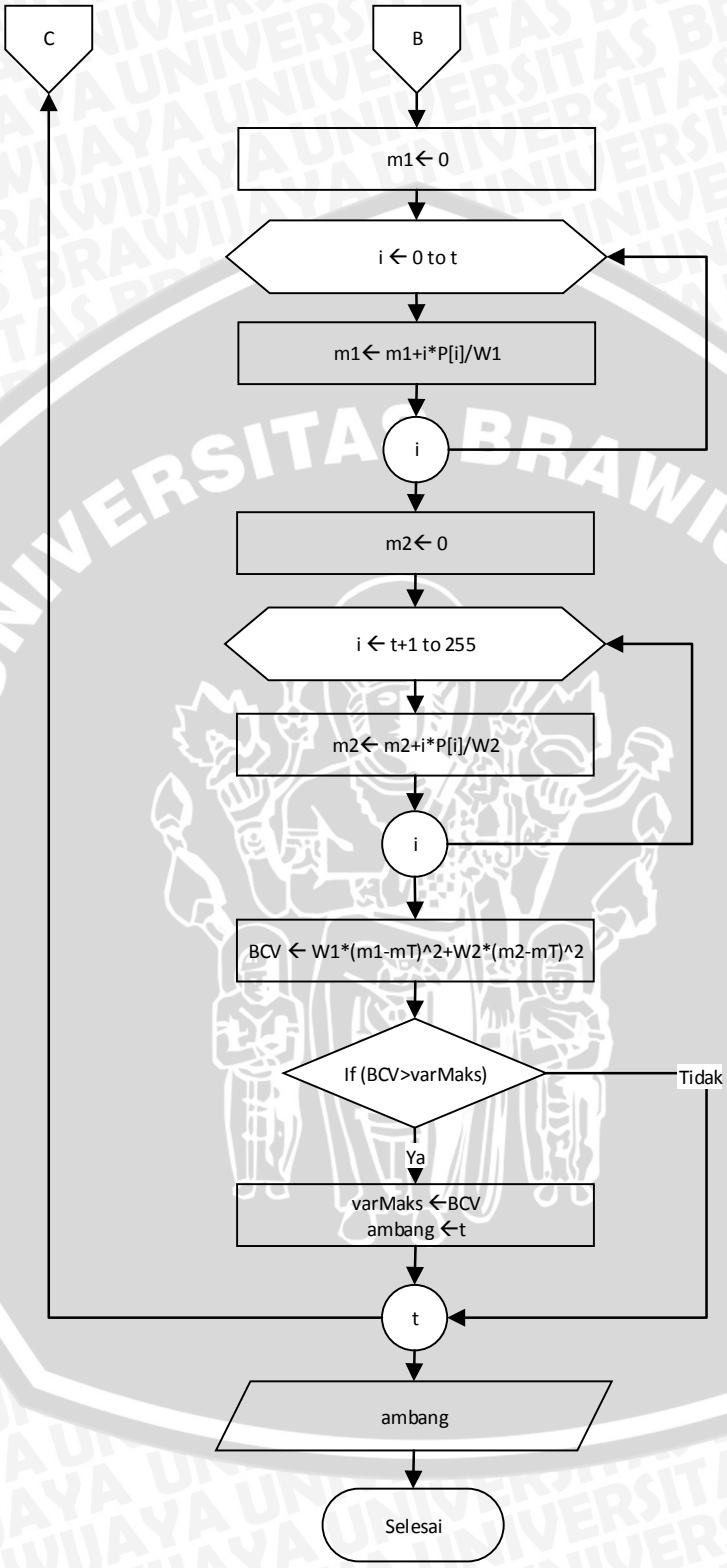
Gambar 4.6 Perhitungan Nilai Histogram *Grayscale*

4. Perhitungan metode *Otsu*

Proses perhitungan metode *Otsu* merupakan algoritma yang mempunyai keluaran nilai *threshold* untuk selanjutnya digunakan untuk menentukan area data citra buah jeruk berpenyakit. Metode *Otsu* telah dijelaskan pada subbab 2.5.6, alur perhitungan metode *otsu* ditunjukkan pada Gambar 4.7.


Mulai
histData[],totalpixel
P[] ← p[256]
i ← 0 to 255
P[i] ← histData[i]/totalpixel
i
mT ← 0
i ← 0 to 255
mT ← mT + i * P[i]
i
A

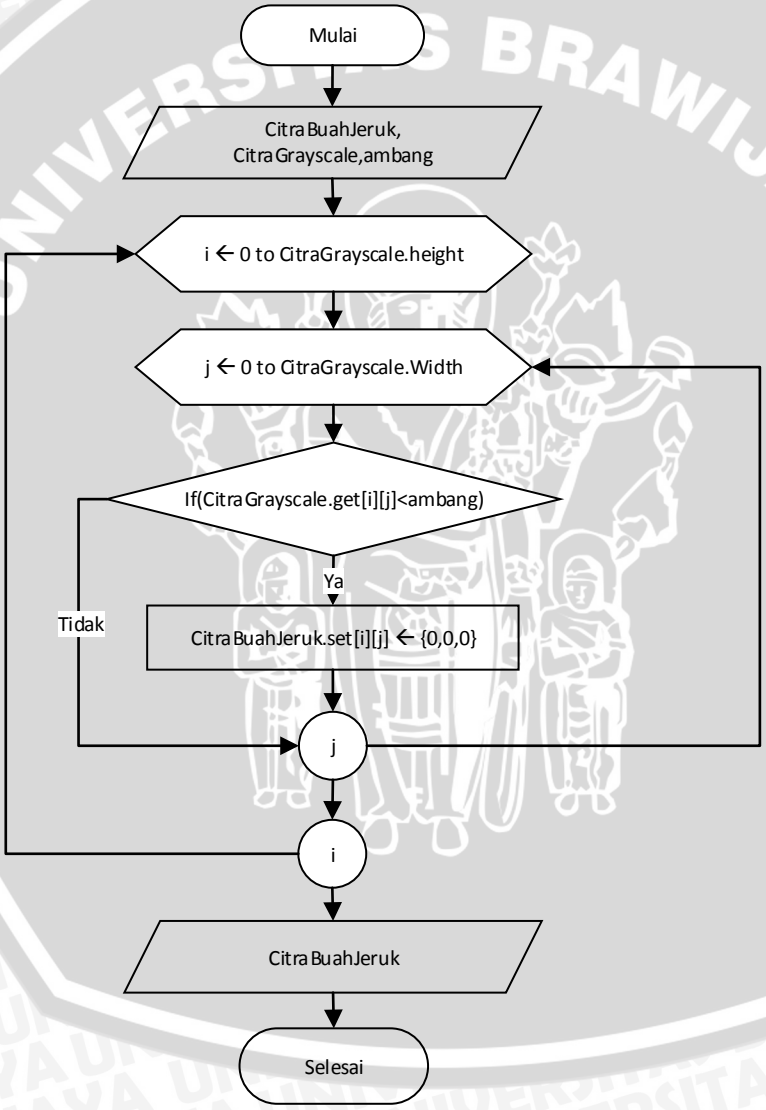

A
ambang ←0
varMaks←0
t ← 0 to 255
W1← 0
i ← 0 to t
W1← W1+P[i]
i
W2← 0
i ← t+1 to 255
W2← W2+P[i]
i
C
B



Gambar 4.7 Perhitungan Metode Otsu

5. Pengambilan Citra Buah Berpenyakit

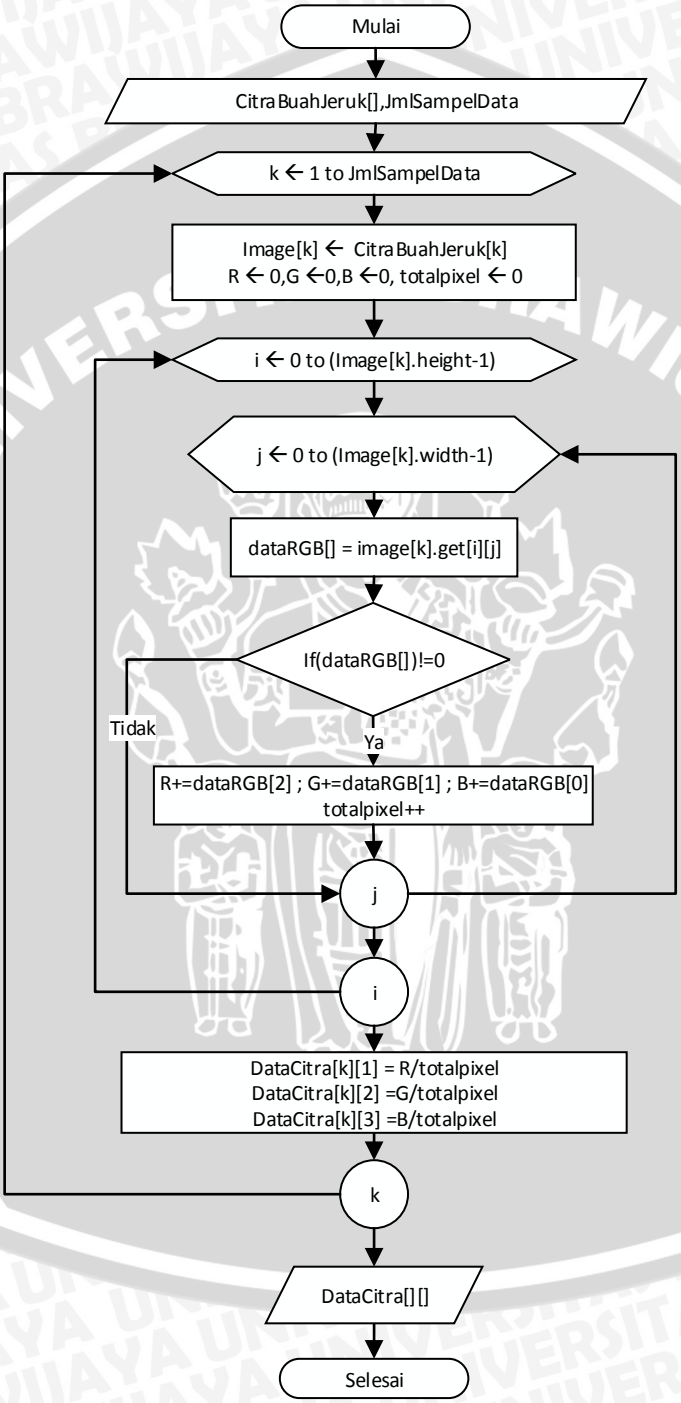
Proses pengambilan citra buah berpenyakit merupakan lanjutan dari proses metode *otsu* dimana nilai histogram *Grayscale* yang melebihi nilai threshold diambil dan digunakan untuk proses selanjutnya. Posisi *pixel* histogram *Grayscale* yang memenuhi syarat dapat menentukan posisi *pixel* data citra asli yang selanjutnya diambil untuk proses selanjutnya sedangkan posisi *pixel* yang tidak memenuhi syarat nilai *RGB* dari data citra asli akan diubah menjadi 0. Berikut alur dari proses pengambilan citra buah berpenyakit ditunjukkan pada Gambar 4.8.



Gambar 4.8 Pengambilan Citra Buah Berpenyakit

6. Perhitungan Rata-Rata *RGB*

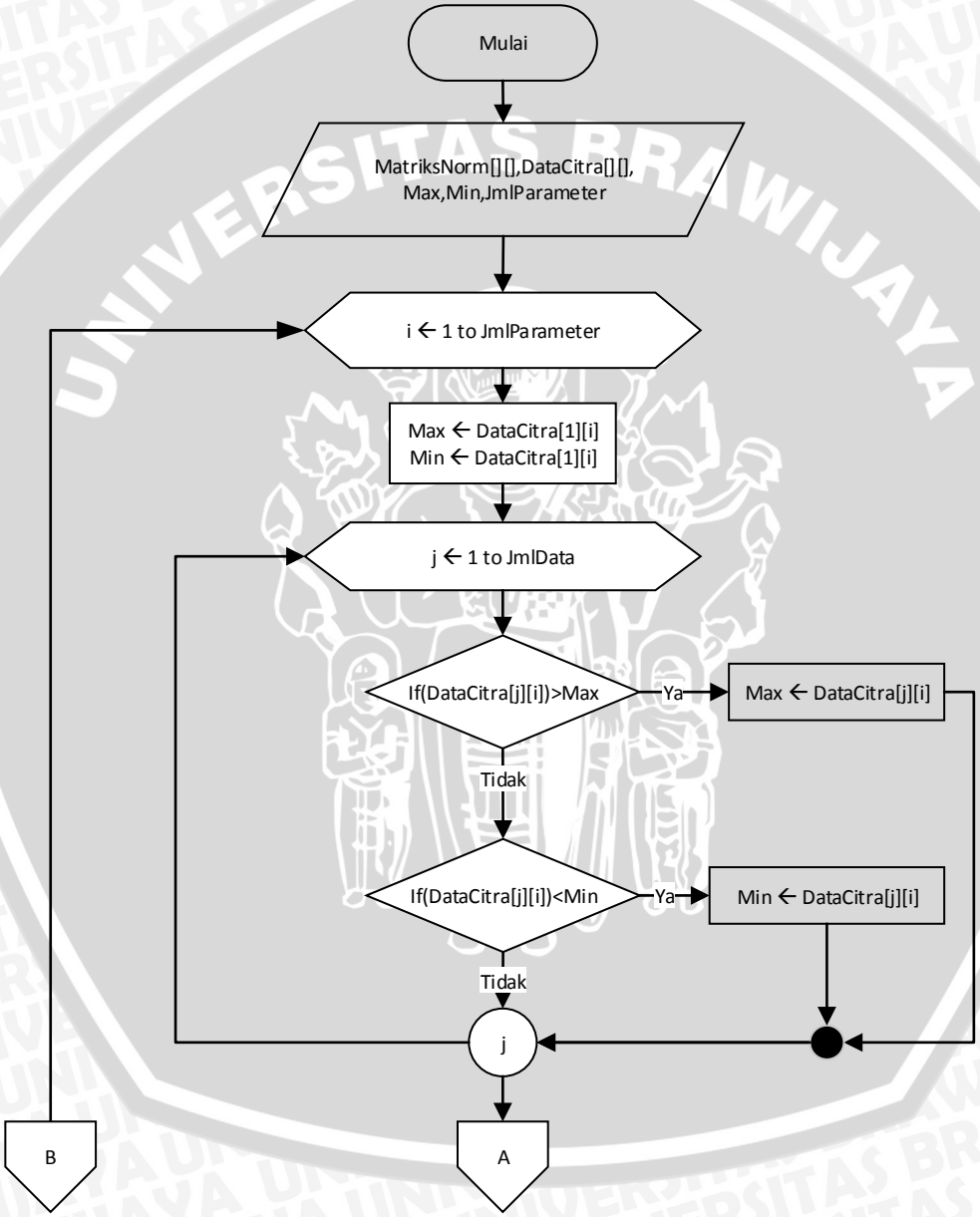
Proses perhitungan rata-rata *RGB* merupakan setelah didapatkan citra yang berpenyakit selanjutnya dilakukan perhitungan rata-rata nilai *RGB* dengan alur yang ditunjukkan pada Gambar 4.9.

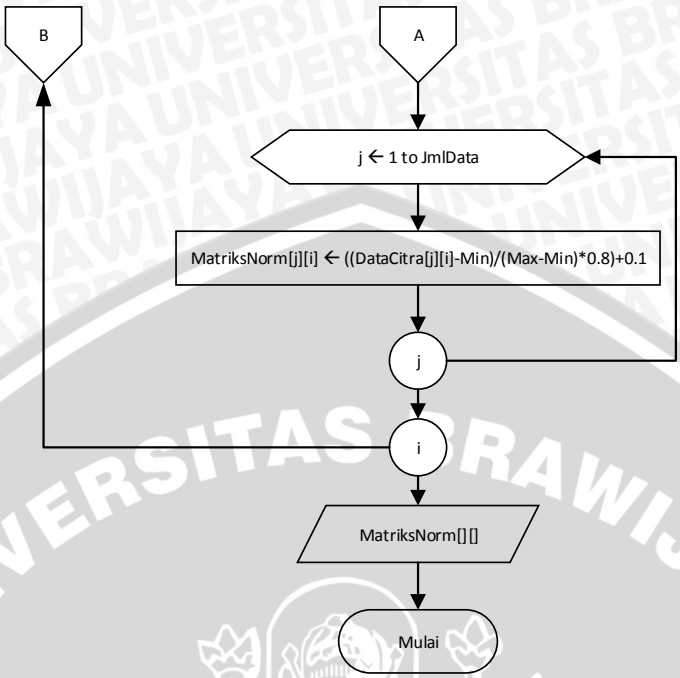


Gambar 4.9 Perhitungan rata-rata *RGB*

4.2.2. Normalisasi Data

Data yang digunakan data *Input* dan data *Output* citra buah jeruk yang terlebih dahulu dilakukan normalisasi. Hal ini bertujuan agar data berada pada range [0,1] karena pada arsitektur JST digunakan fungsi aktivasi sigmoid. Persamaan 2.49 digunakan pada proses normalisasi citra buah jeruk. Proses normalisasi data disajikan dalam *flowchart* pada Gambar 4.10.

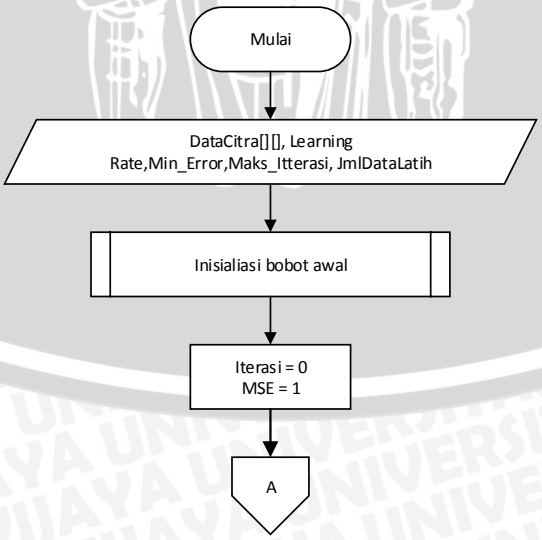


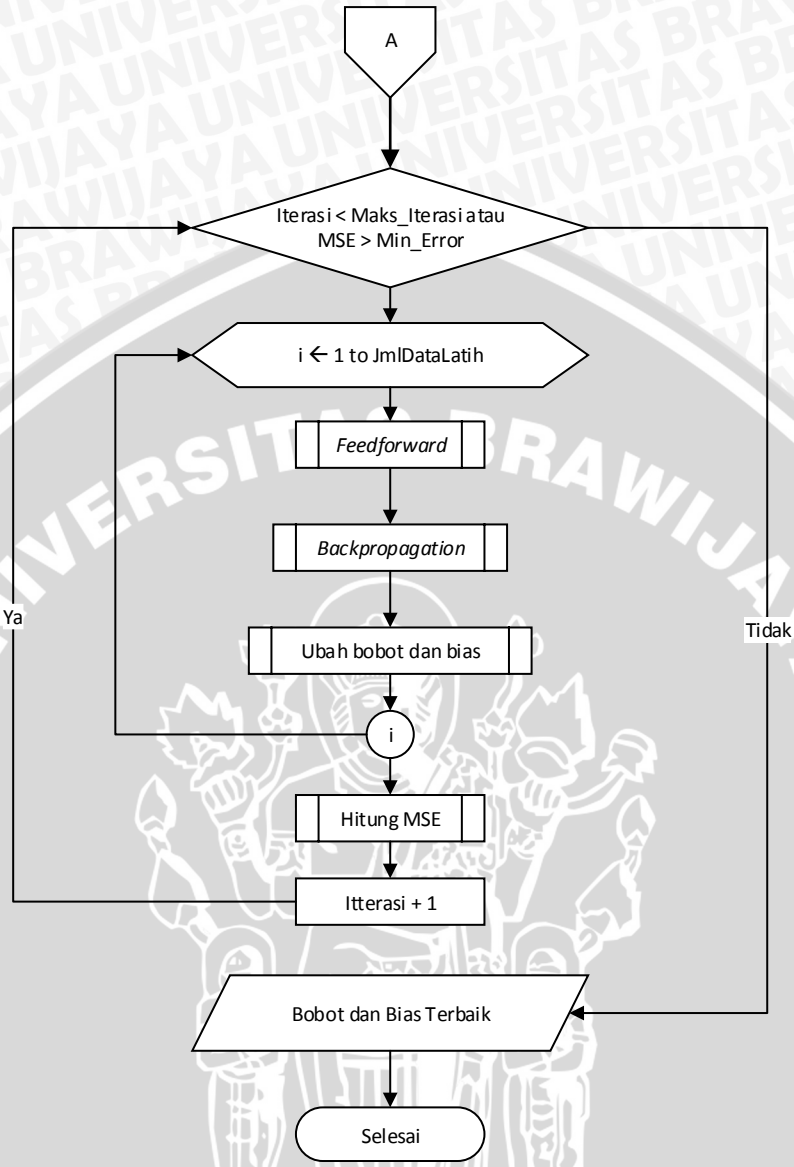


Gambar 4.10 Normalisasi Data

4.2.3. Pelatihan Jaringan Syaraf Tiruan *Backpropagation*

Algoritma pelatihan *backpropagaton* ini diharapkan mampu melakukan identifikasi penyakit serta hama dengan baik. langkah-langkah algoritma *backpropagation* untuk pelatihan JST yang disajikan pada Gambar 4.11 adalah sebagai berikut :

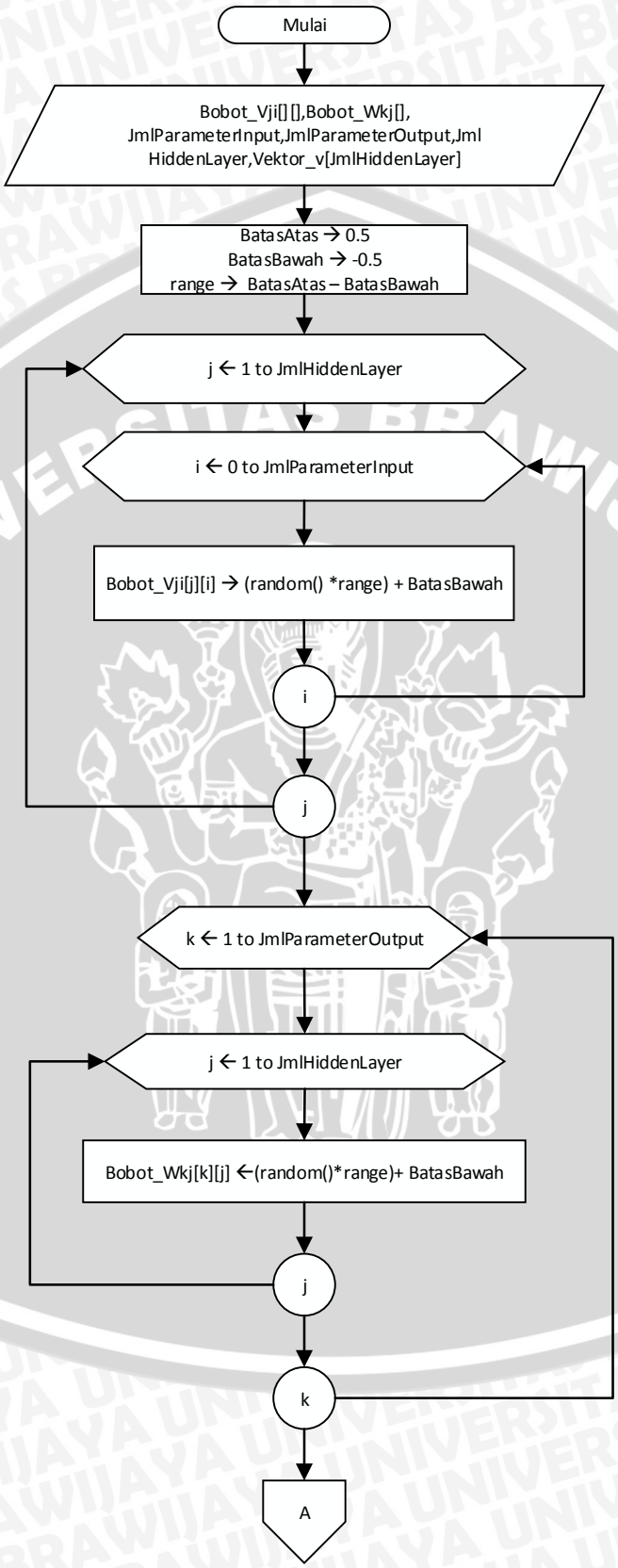


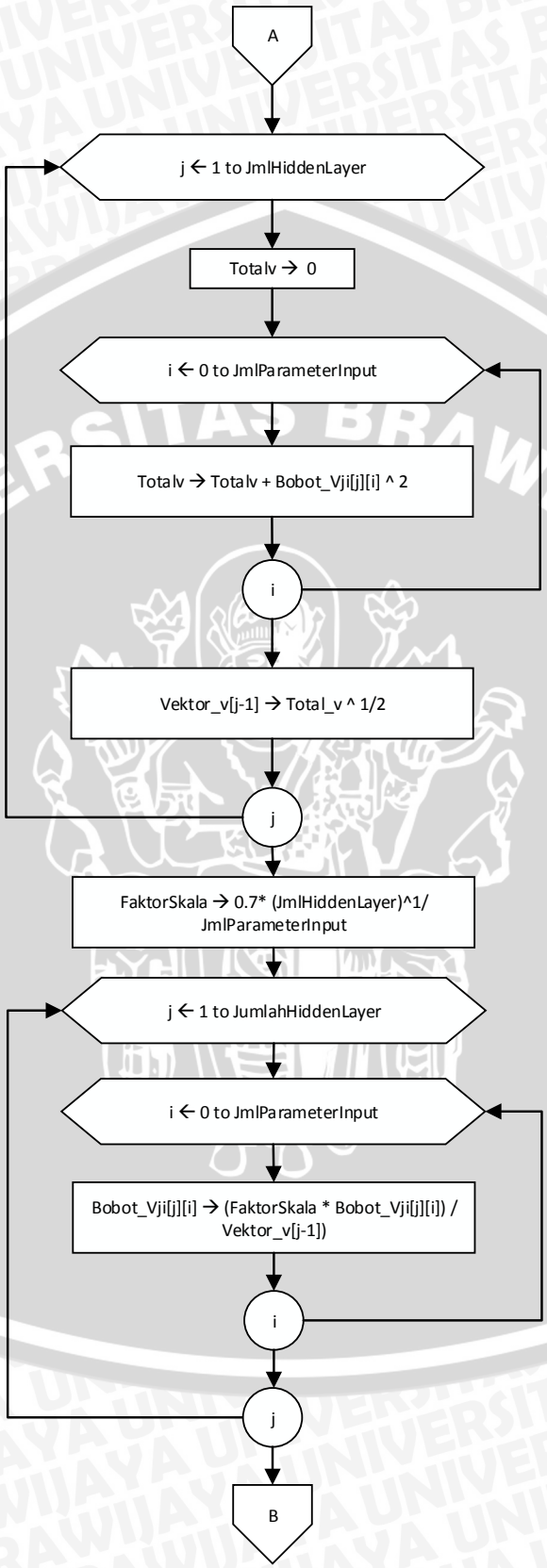


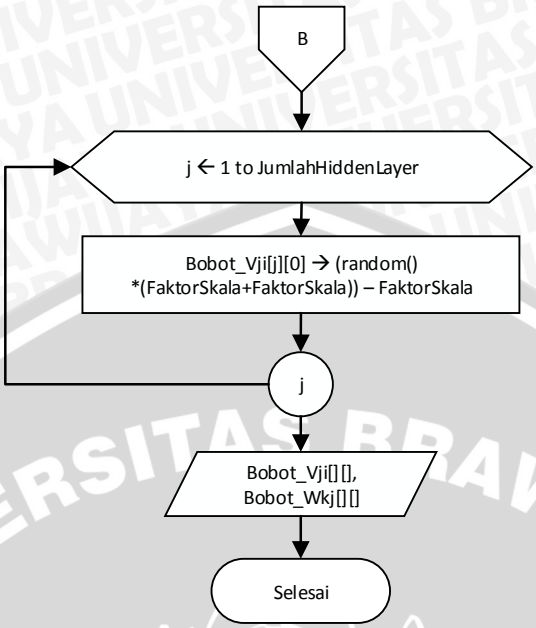
Gambar 4.11 Pelatihan Jaringan Syaraf tiruan Backpropagation

1. Inisialisasi Bobot Awal

Bobot merupakan suatu nilai berupa angka yang menghubungkan antar lapisan *neuron*. Inisialisasi bobot awal pelatihan ini dilakukan menggunakan algoritma *Nguyen-widrow*. Inisialisasi bobot awal disajikan pada Gambar 4.12.


Mulai
Bobot_Vji[][],Bobot_Wkj[],
JmlParameterInput,JmlParameterOutput,Jml
HiddenLayer,Vektor_v[JmlHiddenLayer]
BatasAtas → 0.5
BatasBawah → -0.5
range → BatasAtas – BatasBawah
j ← 1 to JmlHiddenLayer
i ← 0 to JmlParameterInput
Bobot_Vji[j][i] → (random() *range) + BatasBawah
i
j
k ← 1 to JmlParameterOutput
j ← 1 to JmlHiddenLayer
Bobot_Wkj[k][j] ←(random()*range)+ BatasBawah
j
k
A

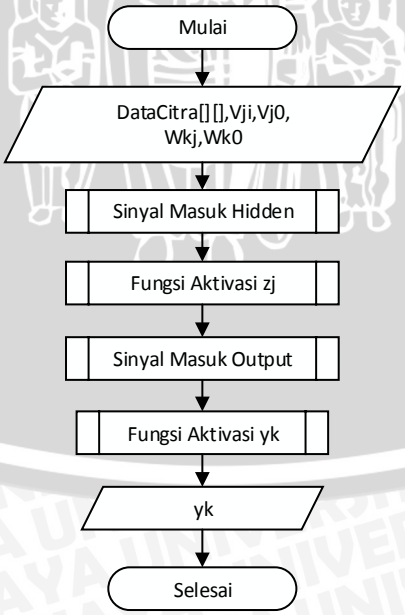

A
j ← 1 to JmlHiddenLayer
Totalv → 0
i ← 0 to JmlParameterInput
Totalv → Totalv + Bobot_Vji[j][i] ^ 2
i
Vektor_v[j-1] → Total_v ^ 1/2
j
FaktorSkala → 0.7* (JmlHiddenLayer)^1/ JmlParameterInput
j ← 1 to JumlahHiddenLayer
i ← 0 to JmlParameterInput
Bobot_Vji[j][i] → (FaktorSkala * Bobot_Vji[j][i]) / Vektor_v[j-1])
i
j
B



Gambar 4.12 Inisialisasi bobot awal *Nguyen-widrow*

2. Proses *FeedForward*

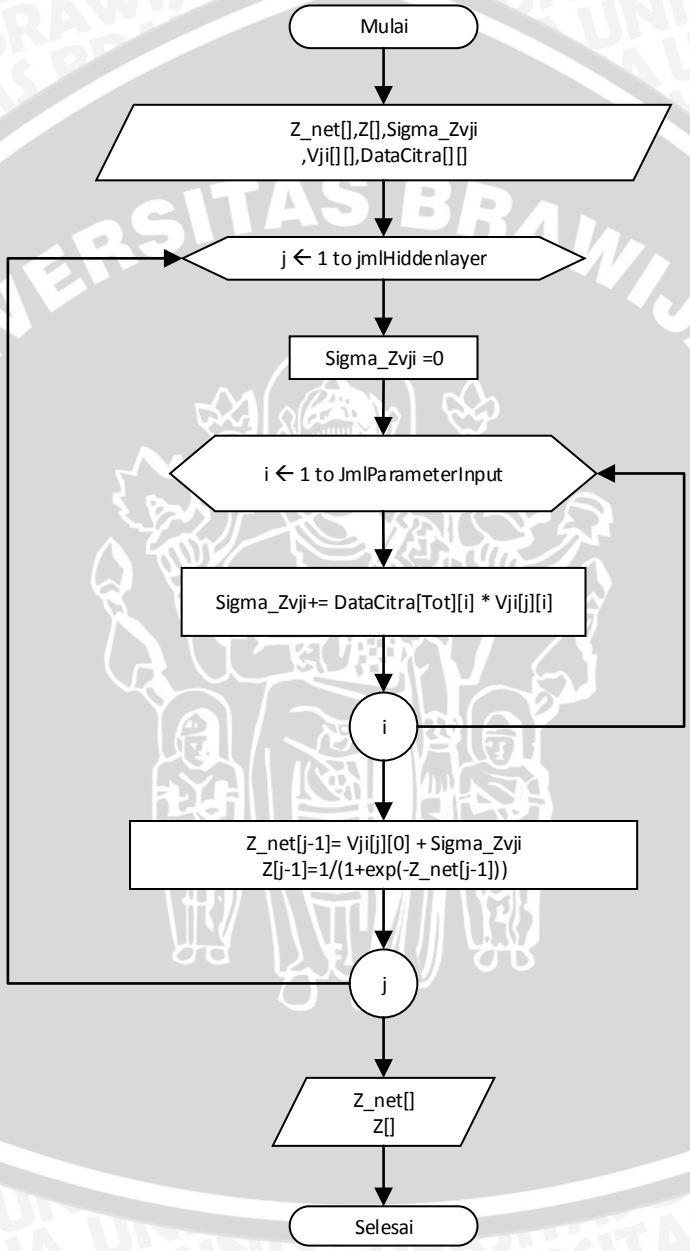
Proses *FeedForward* merupakan proses pada JST yang diawali dengan menjumlahkan perkalian antara *Input* yang telah dinormalisasi dengan bobot yang ada dan kemudian menghitung nilai aktivasinya yang kemudian dari hasil perhitungan tersebut dijadikan masukan oleh lapisan yang berada di atasnya. Proses *FeedForward* dapat dilihat pada Gambar 4.13.



Gambar 4.13 Proses *FeedForward*

a. Proses Hitung Sinyal Masuk *Hidden* z_net dan Fungsi aktivasi zj

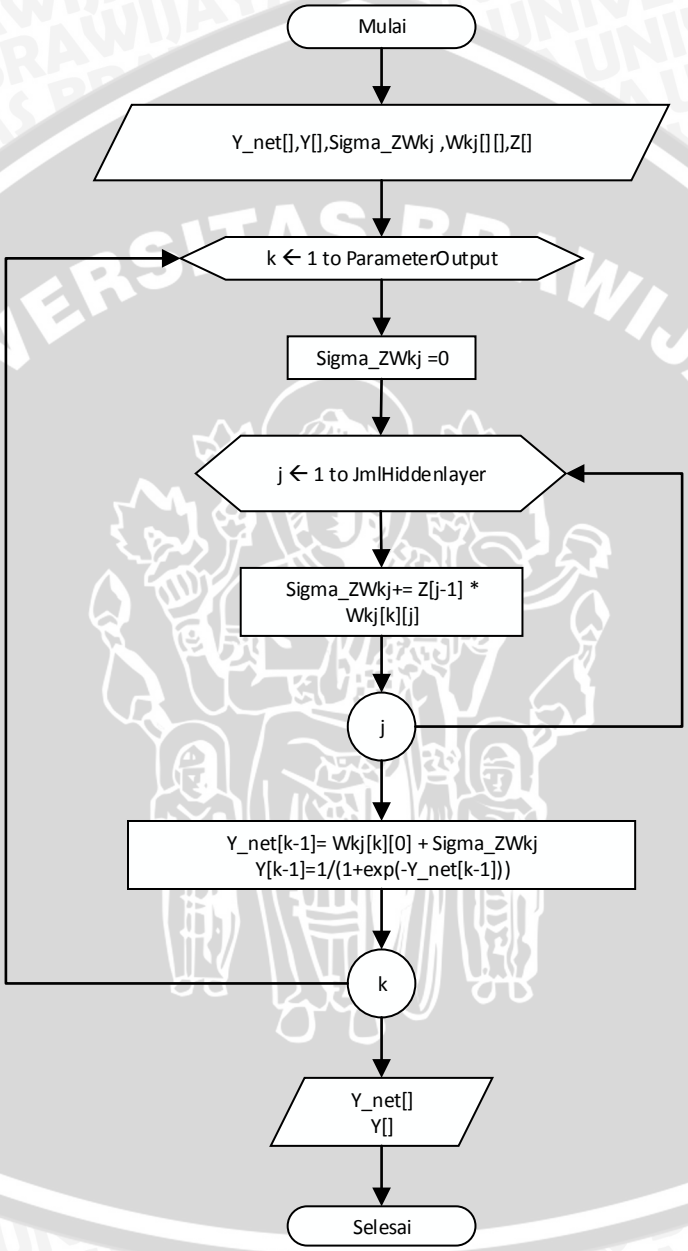
Proses menghitung sinyal masukan pada *Hidden layer* dan fungsi aktivasi ini menggunakan persamaan 2.38 dan persamaan 2.39. Proses Hitung sinyal masuk *Hidden* z_net dan fungsi aktivasi zj ditunjukkan pada Gambar 4.14.



Gambar 4.14 Proses Hitung Sinyal Masuk Hidden z_net dan Fungsi aktivasi zj

b. Proses Hitung SInyal Masuk *Output* y_net dan fungsi aktivasi yk

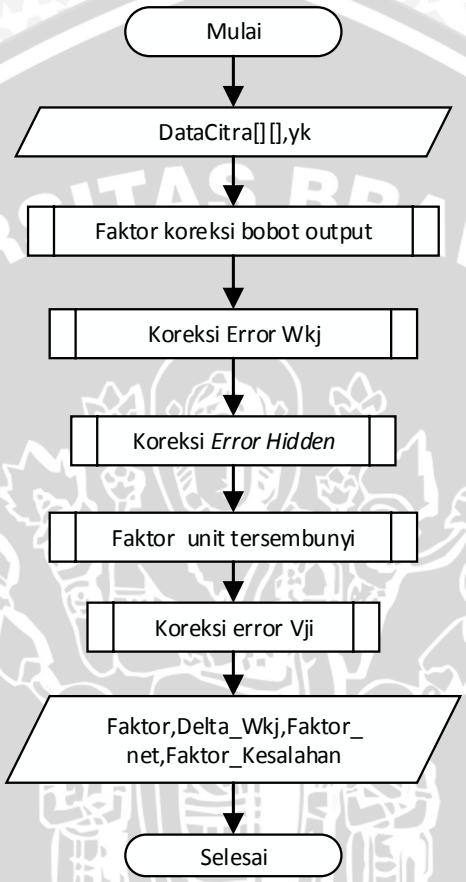
Proses menghitung sinyal masukan pada *output layer* y_net dan fungsi aktivasi yk ini menggunakan persamaan 2.40 dan persamaan 2.41. Proses hitung sinyal masuk *output* y_net dan fungsi aktivasi yk ditunjukkan pada Gambar 4.15.



Gambar 4.15 Proses Hitung SInyal Masuk *Output* Y_net dan fungsi aktivasi yk

3. Proses *backpropagation*

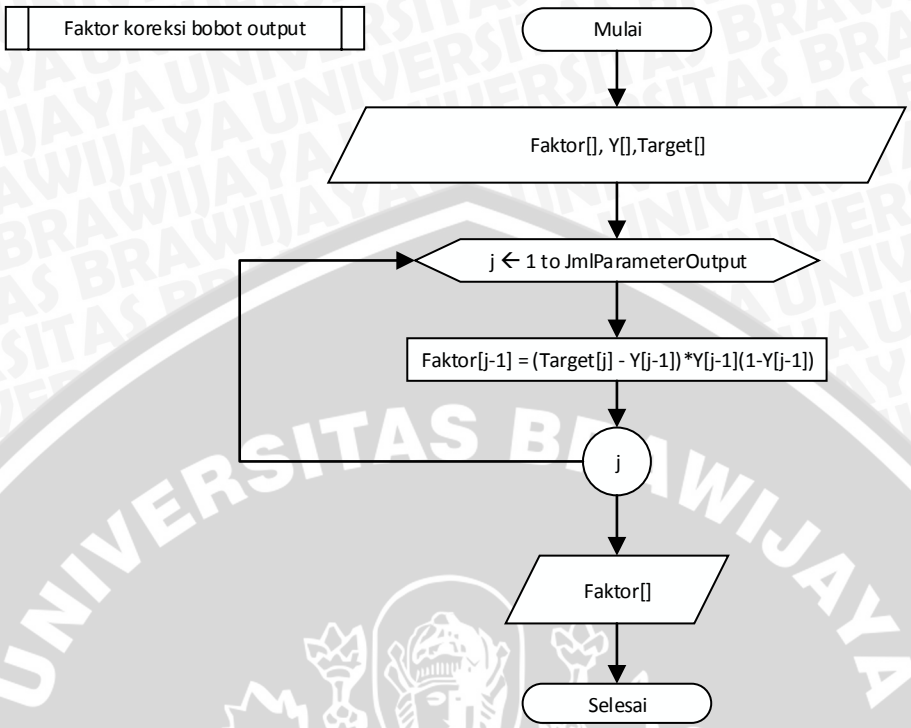
Proses *backpropagation* merupakan langkah untuk menghitung informasi kesalahan pada tiap *neuron* pada masing-masing *layer* yang dimulai dari kesalahan pada *output layer* hingga kesalahan pada *Hidden layer*. Proses *backpropagation* diperlihatkan pada Gambar 4.16.



Gambar 4.16 Proses backpropagation

a. Proses Hitung Faktor Koreksi Bobot *Output* δ_k

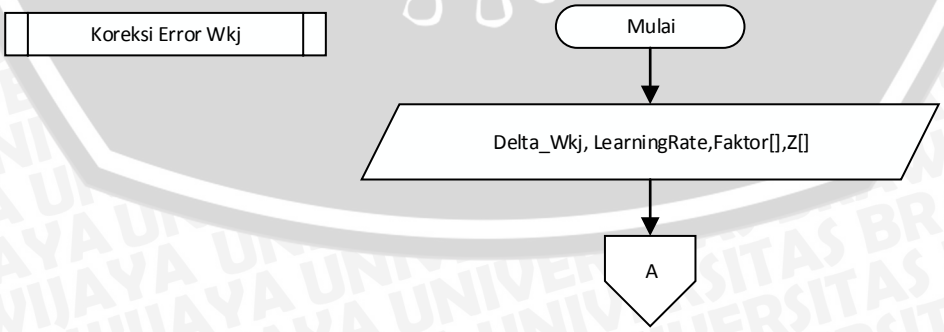
Proses pertama dilakukan pada *backpropagation* adalah menghitung faktor koreksi bobot *output* pada *output layer*. Perhitungannya seperti pada persamaan 2.42. Proses hitung faktor koreksi bobot *output* dapat dilihat pada Gambar 4.17.

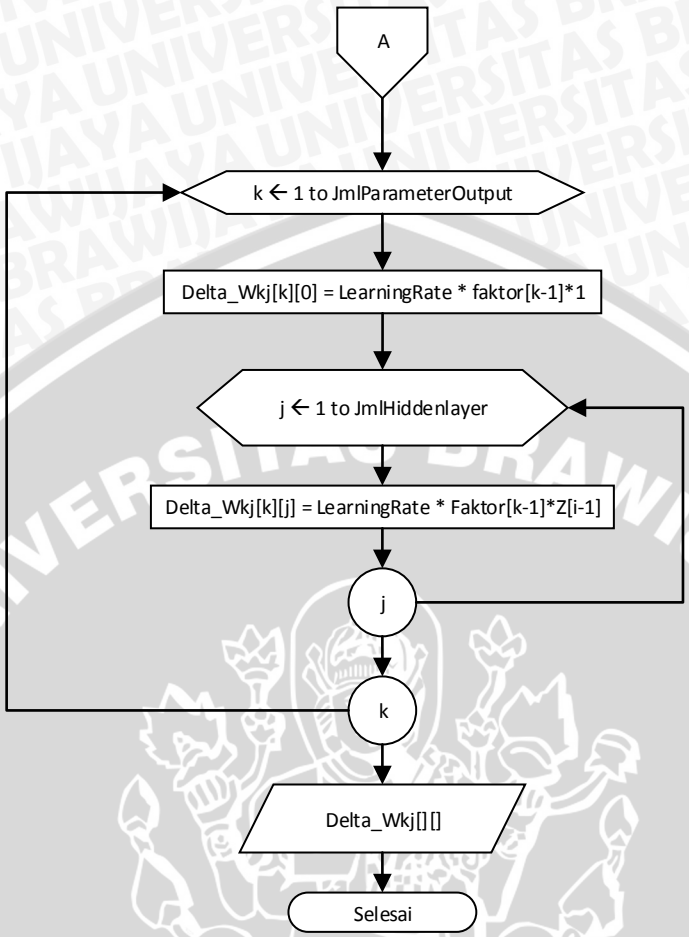


Gambar 4.17 Proses Hitung Faktor Koreksi Bobot *Output* δ_k

b. Proses Hitung Koreksi *Error* Δw_{kj}

Proses yang dilakukan setelah proses menghitung faktor koreksi bobot *output* pada *output layer* adalah proses hitung koreksi *Error* Δw_{kj}. Nilai koreksi *Error* Δw_{kj} ini nantinya akan digunakan untuk merubah bobot Δw_{kj}. Perhitungannya menggunakan persamaan 2.43. Proses hitung koreksi *Error* Δw_{kj} dapat dilihat pada Gambar 4.18.

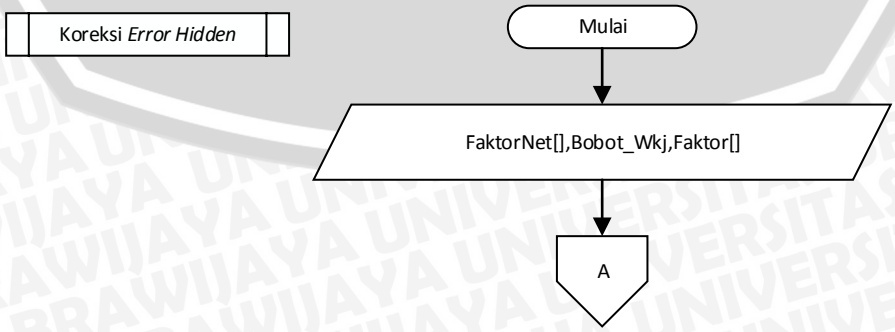


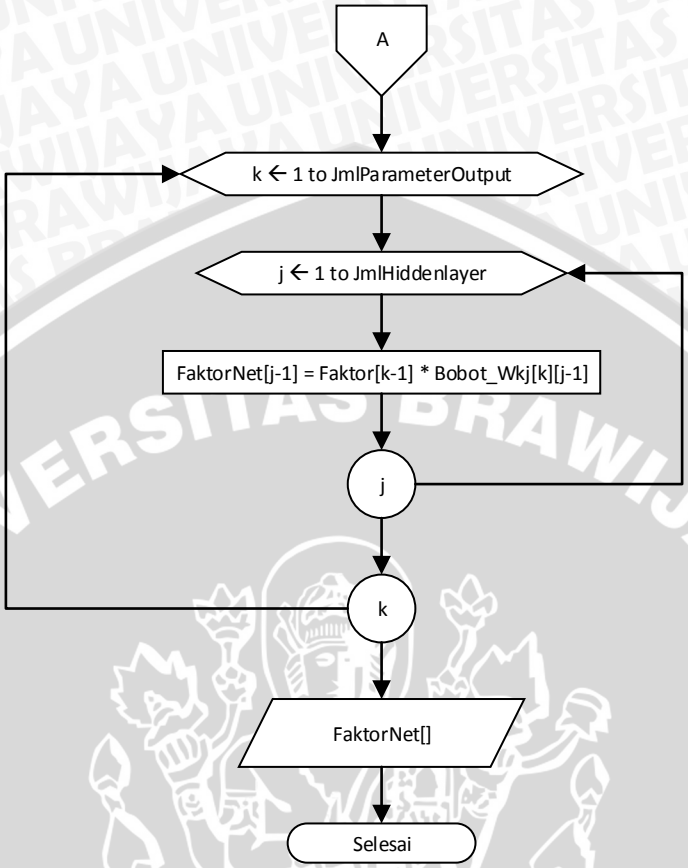


Gambar 4.18 Proses Hitung Koreksi *Error* ΔW_{kj}

c. Proses Hitung Faktor Koreksi *Error Hidden* δ_net_j

Selanjutnya dilakukan proses hitung faktor koreksi *Error Hidden*. Perhitungannya dapat dilihat pada persamaan 2.44. Proses hitung faktor koreksi *Error Hidden* dapat dilihat pada Gambar 4.19.

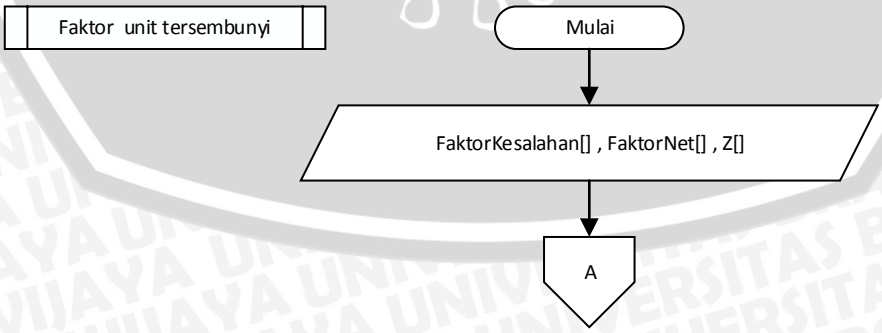


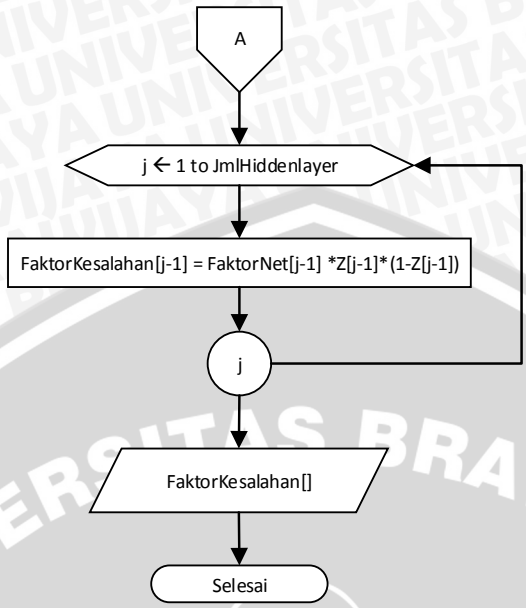


Gambar 4.19 Proses Hitung Faktor Koreksi *Error* Hidden δ_net_j

d. Proses Hitung Faktor δ_i unit tersembunyi

Selanjutnya dilakukan proses hitung faktor δ_i unit tersembunyi. Perhitungannya dapat dilihat pada persamaan 2.45. Proses hitung faktor faktor δ_i unit tersembunyi dapat dilihat pada Gambar 4.20.

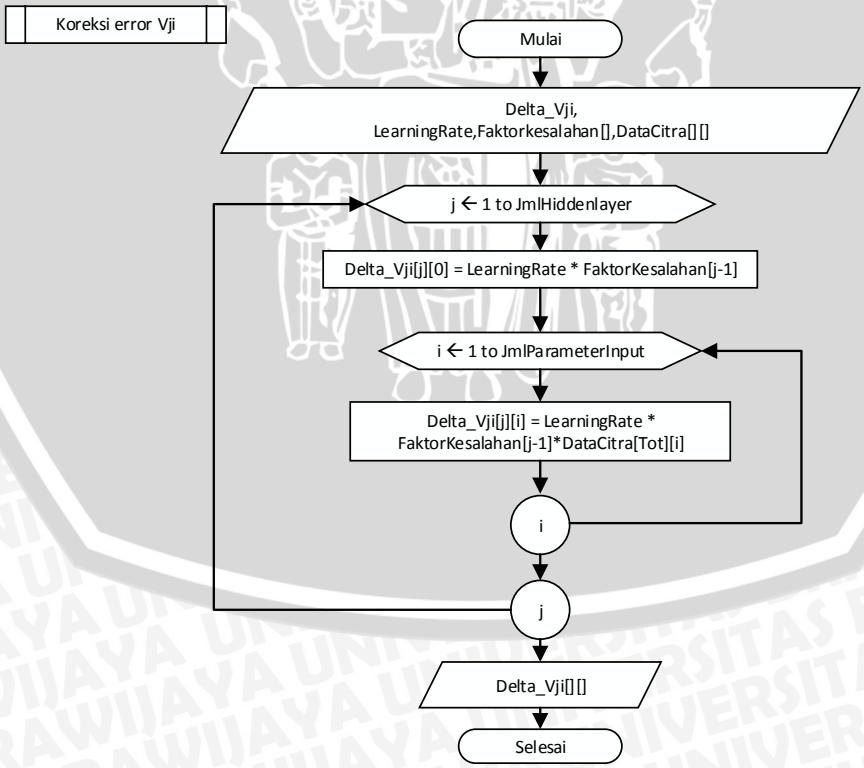




Gambar 4.20 Proses Hitung Faktor δ_i unit tersembunyi

e. Proses Hitung Koreksi *Error* Δv_{ji}

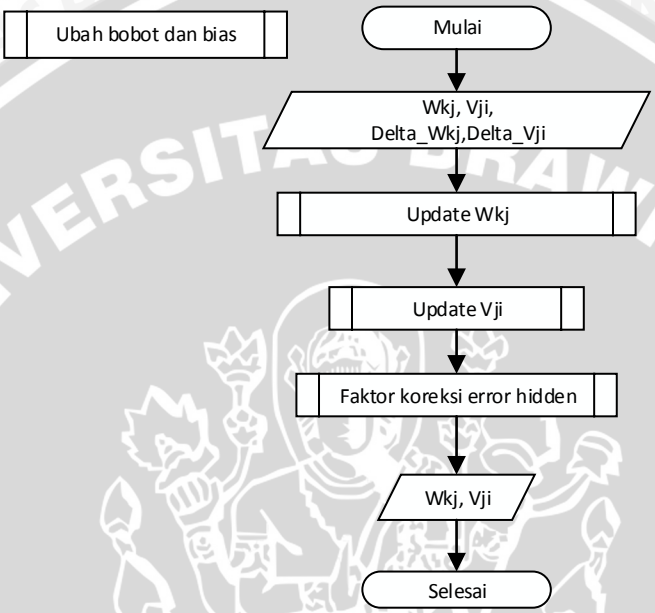
Kemudian dilakukan proses hitung koreksi *Error* Δv_{ji} perhitungannya dapat dilihat pada persamaan 2.46 prosesnya dapat dilihat pada Gambar 4.21.



Gambar 4.21 Proses Hitung Koreksi *Error* Δv_{ji}

4. Proses Ubah Bobot dan Bias

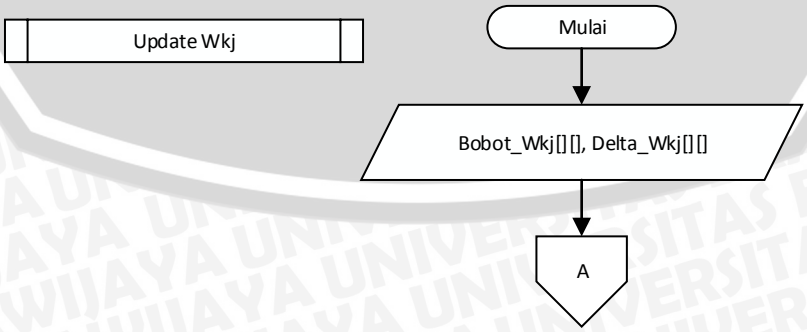
Hasil dari proses *backpropagation* adalah nilai koreksi *Error* bobot dan bias yang digunakan untuk melakukan perubahan nilai bobot dan bias. Proses perbahan bobot dan bias dilakukan untuk mendapatkan nilai bobot dan nilai bias yang baru. Proses perbahan bobot dan bias dapat dilihat pada Gambar 4.22.

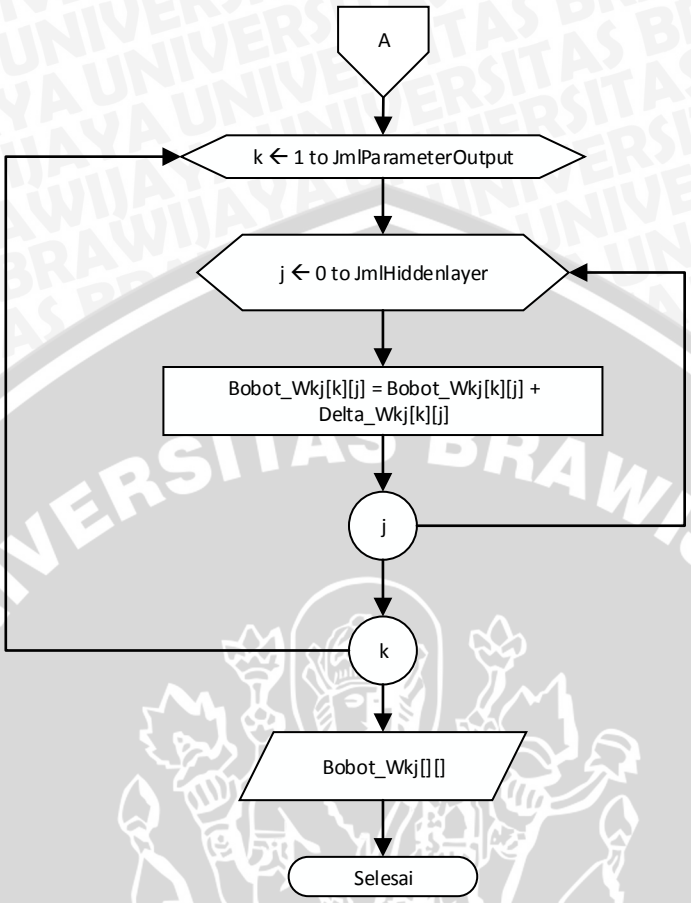


Gambar 4.22 Proses Ubah Bobot dan Bias

a. Proses Update w_{kj}

Proses yang pertama kali dilakukan pada ubah bobot dan bias adalah proses *update* w_{kj}. Proses ini dilakukan untuk mendapatkan nilai bobot w_{kj} yang baru. Perhitungan pada proses ini dapat dilihat pada persamaan 2.47. Proses update w_{kj} dapat dilihat pada Gambar 4.23.

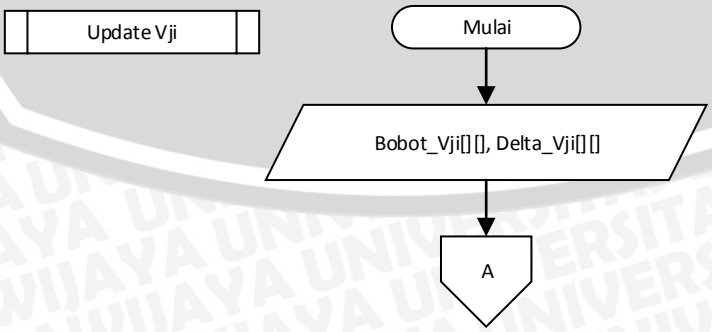


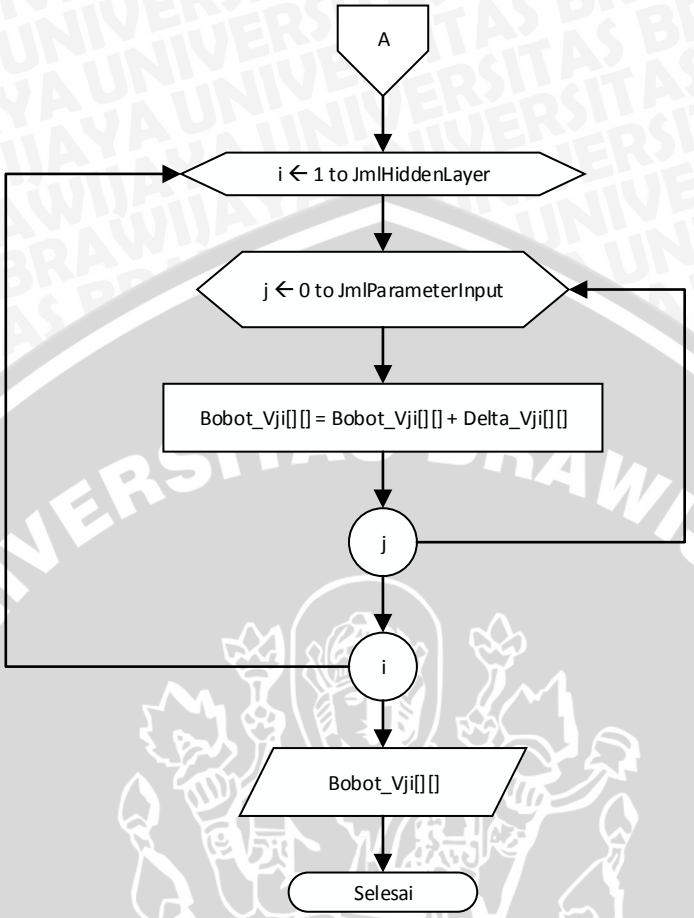


Gambar 4.23 Proses Update w_{kj}

b. Proses Update v_{ji}

Setelah mendapatkan bobot wjk yang baru, proses selanjutnya adalah *update* v_{ji} yang digunakan untuk mendapatkan nilai bobot v_{ji} yang baru. Perhitungan pada proses ini dapat dilihat pada persamaan 2.48. Proses *update* v_{ji} dapat dilihat pada Gambar 4.24.

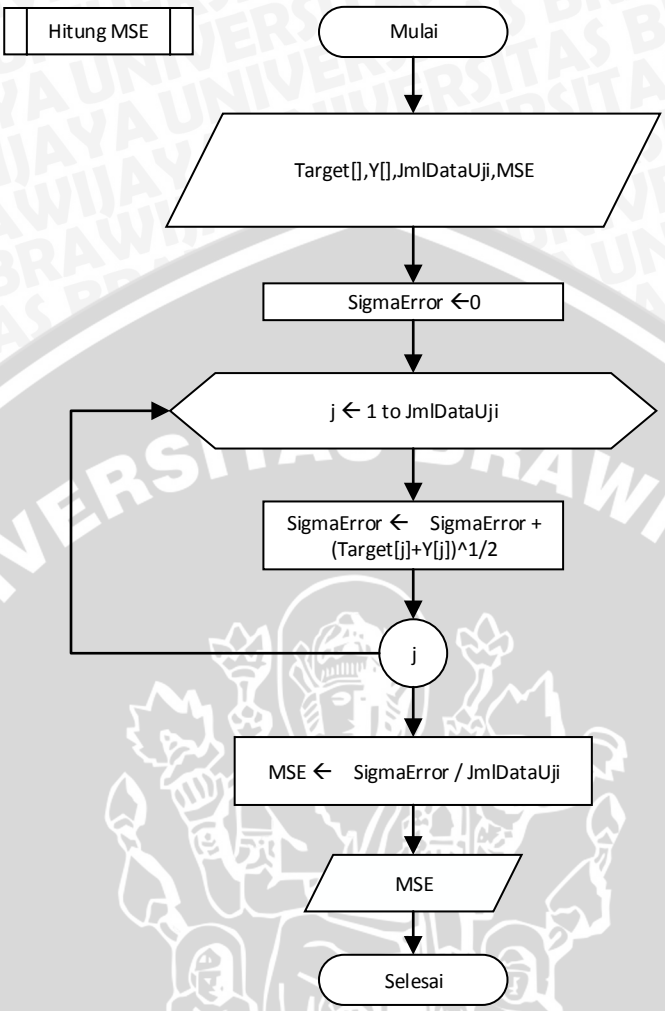




Gambar 4.24 Proses Update v_{ji}

4. Memeriksa *Stop Condition* MSE

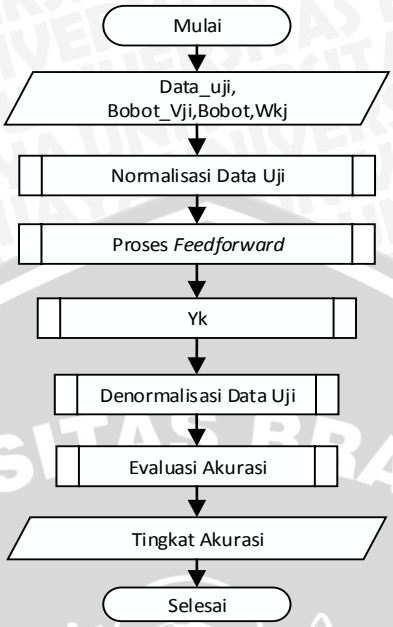
Langkah terkahir dari pelatihan JST dengan *backpropagation* adalah memeriksa *Stop Condition* pada penelitian ini digunakan 2 jenis parameter *Stop Condition* yaitu maksimum iterasi serta minimum *Error/* minimum MSE. Maksimum iterasi membatasi setiap perulangan dari pelatihan JST *backpropagation* untuk memperoleh bobot serta bias baru sedangkan minimum *Error* membatasi pecapaian nilai minimum MSE pada setiap akhir proses perulangan mendapatkan bobot dan bias baru pada pelatihan JST *backpropagation* ditunjukkan pada Gambar 4.25.



Gambar 4.25 Memeriksa Stop Condition MSE

4.2.4. Pengujian

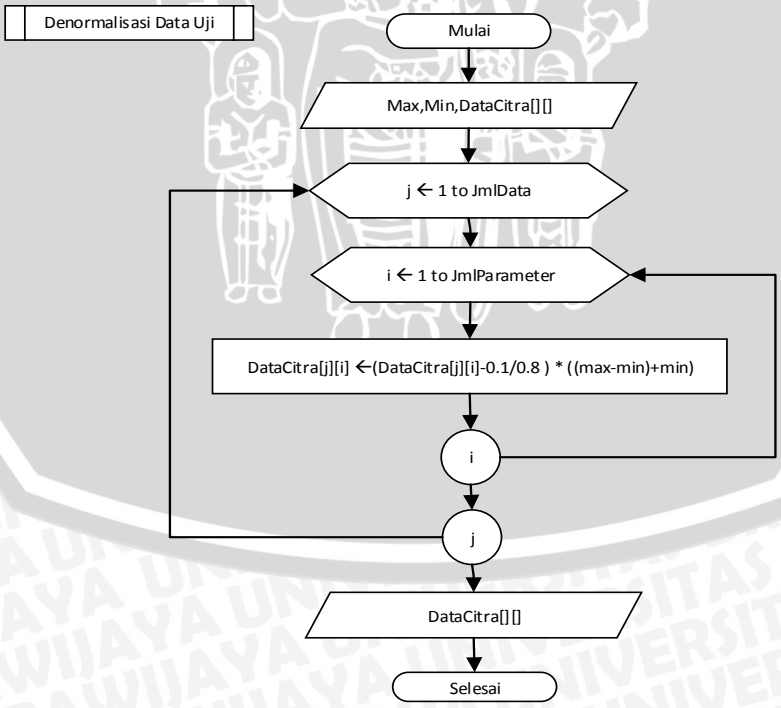
Pada proses pengujian, langkah-langkah yang digunakan pada metode *backpropagation* adalah normalisasi data uji, proses *FeedForward* serta denormalisasi nilai target dan *ouput* jaringan tersebut untuk mengetahui hasil dari klasifikasi. Proses pengujian dapat dilihat pada Gambar 4.26.



Gambar 4.26 Diagram Alir Pengujian

1. Denormalisasi Data Uji

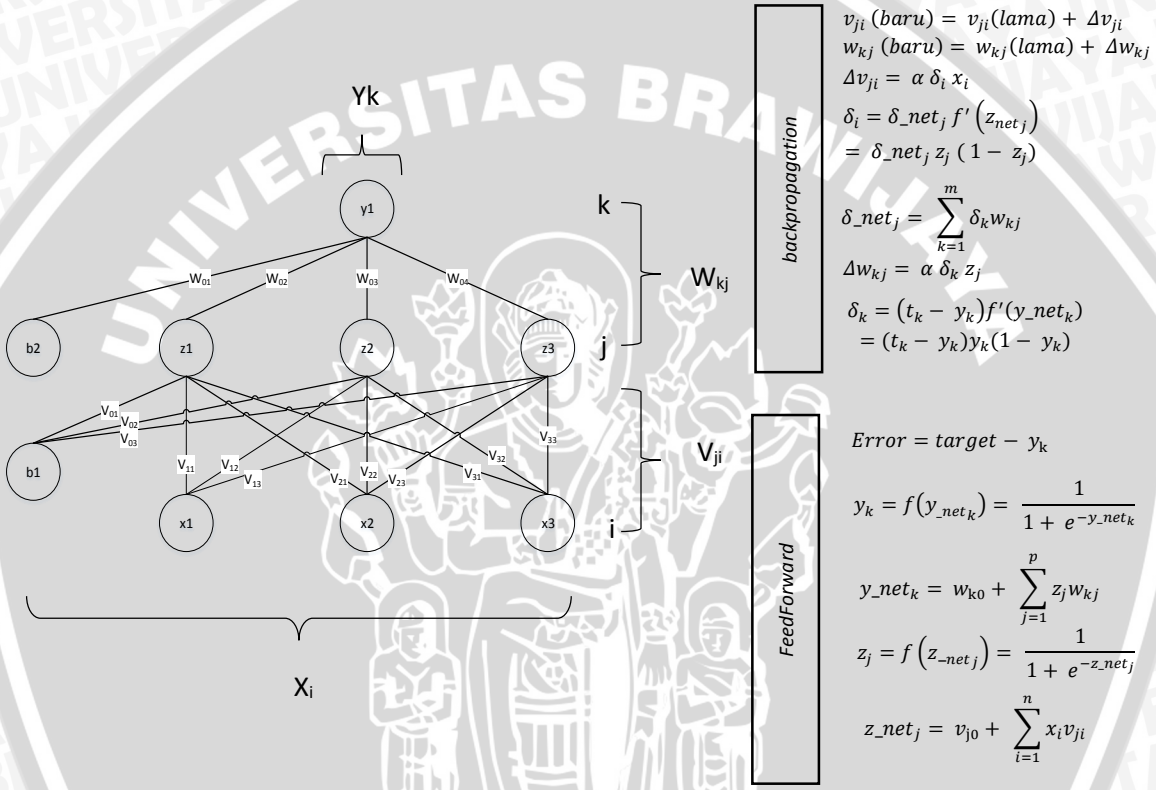
Denormalisasi merupakan proses pengembalian nilai normalisasi ke dalam nilai asli pada jaringan *backpropagation* untuk menentukan target yang sesuai. Proses denormalisasi data uji sesuai rumus 2.50. ditunjukkan pada Gambar 4.27.



Gambar 4. 27 Denormalisasi Data Uji

4.3. Perhitungan Manual

Pada perhitungan manual ini akan digunakan 3 data citra buah jeruk yang terkena penyakit embun jelaga, 3 data citra buah jeruk yang terkena hama trips serta 4 citra buah jeruk yang normal. Pada proses pelatihan data, digunakan beberapa parameter yaitu *learning rate*= 0.2, maksimum iterasi = 1, dan minimum *Error* = 0.01 yang akan diterapkan pada perhitungan metode *Backpropagation* seperti ditunjukkan pada Gambar 4.28.



Gambar 4.28 Algoritma Backpropagation

4.3.1. Preprocessing Citra dan Ekstraksi Fitur

Dalam *preprocessing* citra, dimisalkan terdapat citra yang berukuran 5 x 5 *pixel*s diambil dari bagian *cropping* citra yang teridentifikasi terkena hama atau penyakit pada buah jeruk dengan nilai *red*, *green* dan *blue* (*RGB*) seperti ditunjukkan pada Tabel 4.1.

Tabel 4.1 Nilai *RGB* sampel preprocessing citra

Pixel		x														
		0			1			2			3			4		
		R	G	B	R	G	B	R	G	B	R	G	B	R	G	B
y	0	252	192	94	232	185	115	157	133	97	135	126	117	115	157	133
	1	227	167	69	215	168	98	149	125	89	134	125	116	98	149	125
	2	216	156	58	213	166	96	161	137	101	156	147	138	96	161	137
	3	225	165	67	229	182	112	187	163	127	188	179	170	112	187	163
	4	98	149	125	149	125	89	134	125	116	149	125	89	134	125	116

Setelah mendapatkan nilai *RGB* dari citra yang teridentifikasi penyakit maka langkah selanjutnya adalah melakukan pendetailan bagian berpenyakit menggunakan segmentasi dengan metode *otsu.*

1. Proses *manual Cropping*

Untuk melakukan pemilihan citra yang berpenyakit pada buah jeruk dilakukan proses pemotongan secara *manual* (*Manual Cropping*). Berikut langkah-langkah dalam melakukan *manual cropping* :

Langkah 1 : Inisialisasi posisi *cropping* serta ukuran hasil *cropping*

Misal :

Posisi awal *cropping* adalah (0,0) sehingga x = 0 , y = 0.

Ukuran hasil *cropping* adalah 4 x 4 pixel

Langkah 2 : Menentukan titik akhir *cropping*

x (hasil) = x(awal) + ukuran pixel (x)

x (hasil) = 0 + 4 = 4

y (hasil) = y(awal) + ukuran pixe (y)

y (hasil) = 0 + 4 = 4

Karena posisi awal dinilai sebagai pixel 1 x 1 maka dari hasil perhitungan diatas masing-masing hasilnya dikurangi 1 .

x (hasil) = 4 - 1 = 3

y (hasil) = 4 -1 = 3

Jadi titik akhir *cropping* berada pada koordinat (3,3)

Tabel 4.2 Hasil proses *Cropping* dari titik (0,1) sampai (1,2)

Pixel		x														
		0			1			2			3			4		
		R	G	B	R	G	B	R	G	B	R	G	B	R	G	B
y	0	252	192	94	232	185	115	157	133	97	135	126	117	115	157	133
	1	227	167	69	215	168	98	149	125	89	134	125	116	98	149	125
	2	216	156	58	213	166	96	161	137	101	156	147	138	96	161	137
	3	225	165	67	229	182	112	187	163	127	188	179	170	112	187	163
	4	98	149	125	149	125	89	134	125	116	149	125	89	134	125	116

Keterangan :

Hasil Proses *Cropping*

2. Segmentasi menggunakan metode *Otsu*

Untuk melakukan proses dalam metode otsu ada beberapa tahap yang harus dilakukan berikut langkah – langkahnya :

a. Konversi Citra Kedalam *Grayscale*

Proses konversi citra kedalam aras keabuan (*Grayscale*) dilakukan menggunakan rumus 2.3 berikut perhitungannya :

$$Pixel(x,y) = 0.2989R + 0.5870G + 0.1141B$$

$$Pixel(0,0) = 0.2989 * 252 + 0.5870 * 192 + 0.1141 * 94 = 199$$

Ulangi perhitungan konversi citra *Grayscale* sampai *pixel* (3,3) hasilnya dapat dilihat pada Tabel 4.3.

Tabel 4.3 Hasil konversi citra kedalam *Grayscale*

Pixel		x			
		0	1	2	3
y	0	199	191	136	128
	1	174	174	128	127
	2	163	172	140	149
	3	172	188	166	181

b. Perhitungan *Histogram Grayscale*

Setelah melakukan konversi data citra ke dalam *Grayscale* selanjutnya melakukan perhitungan nilai histogram. Nilai histogram diperoleh dari kemunculan derajat

keabuan pada citra perhitungan nilai histogram dari citra *Grayscale* sebelumnya. Berikut hasil perhitungan histogram ditunjukkan pada Tabel 4.4.

Tabel 4.4 Nilai Histogram

i	*ni*
127	1
128	2
136	1
140	1
149	1
163	1
166	1
172	2
174	2
181	1
188	1
191	1
199	1

i = Derajat Keabuan

ni = jumlah *pixel* yang memiliki derajat keabuan i

c. Perhitungan Metode *Otsu*

Prinsip metode *otsu* adalah sebagai berikut. Pertama-tama, probabilitas nilai intensitas *i* dalam histogram dihitung melalui rumus 2.24 sebagai berikut :

$p(i) = \frac{n_i}{N}, p(i) \geq 0, \sum_1^{256} p(i) = 1$

N = Jumlah *pixel* pada citra

$p(127) = \frac{1}{16} = 0.0625$

Berturut-turut nilai p(i) dihitung sesuai *histogram* yang hasilnya dapat dilihat pada Tabel 4.5.

Tabel 4. 5 Nilai p(i)

i	*ni*	*pi = ni/N*
127	1	0.0625
128	2	0.125
136	1	0.0625
140	1	0.0625
149	1	0.0625
163	1	0.0625
166	1	0.0625
172	2	0.125

174	2	0.125
181	1	0.0625
188	1	0.0625
191	1	0.0625
199	1	0.0625
N	**16**	

Langkah selanjutnya adalah menentukan rerata total dengan cara menjumlahkan semua nilai $i.p(i)$ sesuai dengan rumus 2.25.

$$m_T = \sum_{i=1}^{N} i.p(i)$$

Hasil dari perhitungan m_T dapat dilihat pada Tabel 4.6.

Tabel 4.6 Perhitungan m_T

i	*ni*	*pi = ni/N*	*i.p(i)*
127	1	0.0625	7.9375
128	2	0.125	23.9375
136	1	0.0625	32.4375
140	1	0.0625	41.1875
149	1	0.0625	50.5
163	1	0.0625	60.6875
166	1	0.0625	71.0625
172	2	0.125	92.5625
174	2	0.125	114.3125
181	1	0.0625	125.625
188	1	0.0625	137.375
191	1	0.0625	149.3125
199	1	0.0625	**161.75**
N	**16**	**mT**	**161.75**

Dengan n_i menyatakan jumlah *pixel* beritensitas *i* dan *N* menyatakan jumlah semua *pixel* dalam citra. Jika histogram dibagi menjadi dua kelas (objek dan latarbelakang), pembobotan pada kedua kelas ditunjukkan pada rumus 2.26 dan 2.27 , perhitungannya dapat dilihat pada Tabel 4.7 dan 4.8 sebagai berikut :

$$w_1\,(t) = \sum_{i=1}^{t} p(i)$$

$$w_1\,(127) = w_1\,(126) + p(127) = 0 + 0.0625 = 0.0625$$

Perhitungan tersebut dilakukan berturut-turut sampai i = 199 hasilnya dapat dilihat pada Tabel 4.6.

Tabel 4.7 Perhitungan $w_1\ (t)$

i	*ni*	*pi = ni/N*	*i.p(i)*	*t*	*w1*
127	1	0.0625	7.9375	127	0.0625
128	2	0.125	23.9375	128	0.1875
136	1	0.0625	32.4375	136	0.25
140	1	0.0625	41.1875	140	0.3125
149	1	0.0625	50.5	149	0.375
163	1	0.0625	60.6875	163	0.4375
166	1	0.0625	71.0625	166	0.5
172	2	0.125	92.5625	172	0.625
174	2	0.125	114.3125	174	0.75
181	1	0.0625	125.625	181	0.8125
188	1	0.0625	137.375	188	0.875
191	1	0.0625	149.3125	191	0.9375
199	1	0.0625	**161.75**	199	1
N	**16**	**mT**	**161.75**		

$$w_2\ (t) = \sum_{i=t+1}^{L} p(i) = 1 - w_1\ (t)$$

$$w_2\ (127) = 1 - w_1\ (127) = 1 - 0.0625 = 0.9375$$

Perhitungan tersebut dilakukan berturut-turut sampai i = 199 hasilnya dapat dilihat pada Tabel 4.8.

Tabel 4.8 Perhitungan $w_2\ (t)$

t	*w1*	*w2*
127	0.0625	0.9375
128	0.1875	0.8125
136	0.25	0.75
140	0.3125	0.6875
149	0.375	0.625
163	0.4375	0.5625
166	0.5	0.5
172	0.625	0.375
174	0.75	0.25
181	0.8125	0.1875
188	0.875	0.125

191	0.9375	0.0625
199	1	0

Dalam hal ini, L menyatakan jumlah aras keabuan. Rerata kedua kelas dihitung pada rumus 2.28 dan 2.29 sebagai berikut:

$$m_1(t) = \sum_{i=1}^{t} i.p(i)/w_1(t)$$

$$m_1(127) = m_1(126) + \left(\frac{127.p(127)}{w_1(127)}\right) = 0 + \left(\frac{127.0.0625}{0.0625}\right) = 127$$

Perhitungan tersebut dilakukan berturut-turut sampai t = 199 hasilnya dapat dilihat pada Tabel 4.9.

Tabel 4.9 Perhitungan $m_1\ (t)$

i	*ni*	*pi = ni/N*	*i.p(i)*	*t*	*w1*	*m1*
127	1	0.0625	7.9375	127	0.0625	127
128	2	0.125	23.9375	128	0.1875	127.6667
136	1	0.0625	32.4375	136	0.25	129.75
140	1	0.0625	41.1875	140	0.3125	131.8
149	1	0.0625	50.5	149	0.375	134.6667
163	1	0.0625	60.6875	163	0.4375	138.7143
166	1	0.0625	71.0625	166	0.5	142.125
172	2	0.125	92.5625	172	0.625	148.1
174	2	0.125	114.3125	174	0.75	152.4167
181	1	0.0625	125.625	181	0.8125	154.6154
188	1	0.0625	137.375	188	0.875	157
191	1	0.0625	149.3125	191	0.9375	159.2667
199	1	0.0625	**161.75**	199	1	161.75
N	**16**	***mT***	**161.75**			

$$m_2(t) = \sum_{i=t+1}^{L} i.p(i)/w_2(t)$$

$$m_2(127) = \sum_{i=t+1}^{L} i.p(i)/w_2(127) = \sum_{i=t+1}^{L} i.p(i)/0.9375$$

Selanjutnya dihitung penjumlahan nilai *L* dari 0 – 255 yang hasilnya ditunjukkan pada Tabel 4.10.

Tabel 4.10 Perhitungan $\sum_{i=t+1}^{L} i.p(i)/w_2(127)$

t	*i = t+1*	*pi = ni/N*	$\sum_{i=t+1}^{L} i.p(i)/0.9375$
127	128	0.0625	17.06667
135	136	0.125	9.066667
139	140	0.0625	9.333333
148	149	0.0625	9.933333
162	163	0.0625	10.86667
165	166	0.0625	11.06667
171	172	0.0625	22.93333
173	174	0.125	23.2
180	181	0.125	12.06667
187	188	0.0625	12.53333
190	191	0.0625	12.73333
198	199	0.0625	13.26667
		$m_2(127)$	164.0667

Berturut-turut dilakukan perhitungan sampai i = 199 berikut hasilnya ditampilkan pada Tabel 4.11.

Tabel 4.11 Perhitungan $m_2\ (t)$

i	ni	pi = ni/N	i.p(i)	t	w2	m2
127	1	0.0625	7.9375	127	0.9375	164.0667
128	2	0.125	23.9375	128	0.8125	169.6154
136	1	0.0625	32.4375	136	0.75	172.4167
140	1	0.0625	41.1875	140	0.6875	175.3636
149	1	0.0625	50.5	149	0.625	178
163	1	0.0625	60.6875	163	0.5625	179.6667
166	1	0.0625	71.0625	166	0.5	181.375
172	2	0.125	92.5625	172	0.375	184.5
174	2	0.125	114.3125	174	0.25	189.75
181	1	0.0625	125.625	181	0.1875	192.6667
188	1	0.0625	137.375	188	0.125	195
191	1	0.0625	149.3125	191	0.0625	199

199	1	0.0625	161.75	199	0	0
N	16	mT	161.75			

Rumus di atas menunjukkan bahwa BCV adalah jumlah varians kelas secara individual yang telah diboboti dengan probabilitas kelas masing-masing. Adapun BCV dinyatakan dengan rumus 2.30 sebagai berikut :

$$\sigma_B{}^2(t) = W_1.[m_1(t) - m_T]^2 + W_2.[m_2(t) - m_T]^2$$

$$\sigma_B{}^2(127) = 0.0625.[m_1(127) - 161.75]^2 + 0.9375.[m_2(127) - 161.75]^2$$

$$\sigma_B{}^2(127) = 0.0625.[127 - 161.75]^2 + 0.9375.[164.0667 - 161.75]^2$$

$$\sigma_B{}^2(127) = 80.50417$$

setelah mendapatkan nilai BCV dicari nilai maksimal BCV untuk mendapatkan *threshold* yang digunakan ditunjukkan pada Tabel 4.12.

Tabel 4.12 Perhitungan BCV serta nilai maksimal BCV

t	w1	w2	m1	m2	BCV
127	0.0625	0.9375	127	164.0667	80.50417
128	0.1875	0.8125	127.6667	169.6154	268.0785
136	0.25	0.75	129.75	172.4167	341.3333
140	0.3125	0.6875	131.8	175.3636	407.7284
149	0.375	0.625	134.6667	178	440.1042
163	0.4375	0.5625	138.7143	179.6667	412.7232
166	0.5	0.5	142.125	181.375	385.1406
172	0.625	0.375	148.1	184.5	310.5375
174	0.75	0.25	152.4167	189.75	261.3333
181	0.8125	0.1875	154.6154	192.6667	220.5785
188	0.875	0.125	157	195	157.9375
191	0.9375	0.0625	159.2667	199	92.50417
199	1	0	161.75	0	0
				Max(BCV)	**440.1042**

Dari tabel 4.12. disimpulkan bahwa proses perhitungan otsu memperoleh *threshold* pada nilai 149.

Setelah diperoleh *threshold* selanjutnya dilakukan ekstraksi pada citra *Grayscale* untuk dilakukan pemilihan penyakit berikut citra *Grayscale* asli sebelum dilakukan binerisasi pada Tabel 4.13.

Tabel 4.13 Citra *Grayscale* asli sebelum dilakukan binerisasi

Pixel		x			
		0	1	2	3
y	0	199	191	136	128
	1	174	174	128	127
	2	163	172	140	149
	3	172	188	166	181

Selanjutnya dilakukan binerisasi dengan cara *if pixel(x,y) > threshold = 0 else 1* , berikut hasilnya ditunjukkan pada Tabel 4.14.

Tabel 4.14 Binerisasi citra *Grayscale*

Pixel		x			
		0	1	2	3
y	0	0	0	1	1
	1	0	0	1	1
	2	0	0	1	1
	3	0	0	0	0

Terakhir adalah memasukkan nilai biner 1 dengan nilai *RGB* asli berikut hasilnya ditunjukkan pada Tabel 4.15.

Tabel 4.15 Mengisi nilai 1 Binerisasi dengan nilai *RGB*

Pixel		x											
		0			1			2			3		
		R	G	B	R	G	B	R	G	B	R	G	B
y	0	0	0	0	0	0	0	157	133	97	135	126	117
	1	0	0	0	0	0	0	149	125	89	134	125	116
	2	0	0	0	0	0	0	161	137	101	156	147	138
	3	0	0	0	0	0	0	0	0	0	0	0	0

3. Perhitungan Ekstraksi Fitur

Perhitungan ekstraksi fitur dilakukan setelah hasil *preprocessing* telah selesai. Perhitungan ekstraksi fitur bertujuan untuk mendapatkan rata-rata nilai *RGB* secara terpisah. Nilai rata-rata *RGB* merupakan hasil penjumlahan masing-masing *RGB* dibagi

jumlah *pixels*. Perhitungan rata-rata nilai *RGB* secara individu berikut hasil perhitungan rata-rata nilai *RGB* dari Tabel 4.16.

Tabel 4.16 Penjumlahan nilai *RGB* masing-masing *pixel*

Pixel		*x*											
		0			1			2			3		
		R	G	B	R	G	B	R	G	B	R	G	B
y	0	0	0	0	0	0	0	157	133	97	135	126	117
	1	0	0	0	0	0	0	149	125	89	134	125	116
	2	0	0	0	0	0	0	161	137	101	156	147	138
	3	0	0	0	0	0	0	0	0	0	0	0	0
	Total	**0**	**0**	**0**	**0**	**0**	**0**	**467**	**395**	**287**	**425**	**398**	**371**

$$Rata-rata\ R = \frac{Total\ nilai\ R\ pada\ citra}{Jumlah\ pixels\ Penyakit} = \frac{0+0+467+425}{6} = \frac{892}{6} = 148.6666667$$

$$Rata-rata\ G = \frac{Total\ nilai\ G\ pada\ citra}{Jumlah\ pixels\ Penyakit} = \frac{0+0+395+398}{6} = \frac{793}{6} = 132.1666667$$

$$Rata-rata\ B = \frac{Total\ nilai\ B\ pada\ citra}{Jumlah\ pixels\ Penyakit} = \frac{0+0+287+371}{6} = \frac{658}{6} = 109.6666667$$

Dari perhitungan tersebut, didapatkan nilai rata-rata R = 148.6666667, nilai rata-rata G = 132.16666667, dan nilai rata-rata B= 109.6666667.

4.3.2. Normalisasi Data

Pada subbab ini akan dijelaskan mengenai contoh perhitungan manual pada proses identifikasi penyakit serta hama pada citra buah jeruk menggunakan jaringan syaraf tiruan *backpropagation*. Data yang digunakan adalah data rata-rata nilai R, G, dan B, masing-masing sampel citra yang telah dilakukan *preprocessing* serta ekstraksi fitur. Berikut data *RGB* citra yang telah diolah ditunjukkan pada Tabel 4.17.

Tabel 4.17 Data Rata-Rata *RGB* citra buah jeruk

Data ke-	Penyakit/Hama	Rata-Rata			T
		R	G	B	
1	Embun Jelaga	86.151	99.956	92.484	1
2	Embun Jelaga	42.750	48.537	44.700	1
3	Embun Jelaga	45.055	49.884	44.054	1
4	Embun Jelaga	56.753	62.955	58.235	1
5	Thrips	170.313	167.095	121.247	2

6	Thrips	151.963	169.467	158.177	2
7	Thrips	145.641	158.079	145.286	2
8	Kutu Batok	116.200	110.400	83.385	3
9	Kutu Batok	150.465	145.155	116.868	3
10	Kutu Batok	181.322	164.197	143.504	3

Keterangan :

Target 1 = Penyakit Embun Jelaga

Target 2 = Hama Trips

Target 3 = Hama Kutu Batok

Data latih berupa 10 data citra buah jeruk dengan masing-masing memiliki target sesuai penyakit yang teridentifikasi, berikut data latih sebelum di normalisasi pada Tabel 4.18.

Tabel 4.18 Data latih sebelum dinormalisasi

Data ke-	Penyakit/Hama	Rata-Rata			(Y) T
		X1 (R)	*X2* (G)	*X3* (B)	
1	Embun Jelaga	86.151	99.956	92.484	1
2	Embun Jelaga	42.750	48.537	44.700	1
3	Embun Jelaga	45.055	49.884	44.054	1
4	Embun Jelaga	56.753	62.955	58.235	1
5	Thrips	170.313	167.095	121.247	2
6	Thrips	151.963	169.467	158.177	2
7	Thrips	145.641	158.079	145.286	2
8	Kutu Batok	116.200	110.400	83.385	3
9	Kutu Batok	150.465	145.155	116.868	3
10	Kutu Batok	181.322	164.197	143.504	3

Sebelum data digunakan, terlebih dahulu data dilakukan normalisasi dengan menggunakan persamaan 2.49 dengan langkah sebagai berikut :

$$y = \frac{0.8(x - min)}{\max - min} + 0.1$$

Nilai maksimum data (*max*) =

$x_1 = 181.322$

$x_2 = 169.467$

$x_3 = 158.177$

$Y = 3.0$

Nilai minimum data (*min*) =

$x_1 = 42.750$

$x_2 = 48.537$

$x_3 = 44.054$

$Y = 1.0$

Untuk data pertama data ke -1 sebagai berikut

$$x_1 = \frac{0.8(86.151 - 42.750)}{181.322 - 42.750} + 0.1 = 0.351$$

$$x_2 = \frac{0.8(99.956 - 48.537)}{169.467 - 48.537} + 0.1 = 0.440$$

$$x_3 = \frac{0.8(92.484 - 44.054)}{158.177 - 44.054} + 0.1 = 0.439$$

$$y_{(target)} = \frac{0.8(1 - 1)}{3 - 1} + 0.1 = 0.1$$

Berturut-turut langkah di atas dilakukan untuk semua pasang data yang akan diolah. Data yang belum dinormalisasi pada Tabel dihasilkan data yang telah dilakukan normalisasi ditampilkan pada Tabel 4.19.

Tabel 4. 19 Data latih setelah dinormalisasi

Data ke-	Penyakit/Hama	Rata-Rata			T
		R	G	B	
1	Embun Jelaga	0.351	0.440	0.439	0.1
2	Embun Jelaga	0.100	0.100	0.105	0.1
3	Embun Jelaga	0.113	0.109	0.100	0.1
4	Embun Jelaga	0.181	0.195	0.199	0.1
5	Thrips	0.836	0.884	0.641	0.5
6	Thrips	0.731	0.900	0.900	0.5
7	Thrips	0.694	0.825	0.810	0.5

8	Kutu Batok	0.524	0.509	0.376	0.9
9	Kutu Batok	0.722	0.739	0.610	0.9
10	Kutu Batok	0.900	0.865	0.797	0.9

4.3.3. Penentuan Bobot Awal

Penentuan bobot awal pada pelatihan jaringan ini menggunakan algoritma *Nguyen-widrow*, Pada penelitian ini, arsitektur jaringan yang digunakan adalah 3-3-1, yaitu 3 *neuron Input layer*, 3 *neuron Hidden layer*, dan 1 *neuron output layer*, Pertama diberi nilai random antara -0,5 dan 0,5 yang mempresentasikan bobot dari jaringan syaraf tiruan seperti Gambar 4.28. Pemberian *random* bobot awal v_{ji} ditunjukkan pada Tabel 4.20.

Tabel 4.20 Random bobot awal v_{ji} antara -0.5 dan 0.5

v_{ji}	*j*			
i		**1**	**2**	**3**
	1	-0.222	-0.15	-0.424
	2	-0.492	0.024	-0.197
	3	-0.305	-0.123	0.021

Kemudian menghitung β (faktor skala) untuk menentukan bobot bias awal yang dipakai menggunakan persamaan 2.51 sebagai berikut :

$$\beta = 0.7\ \sqrt[n]{p}$$

n = jumlah *neuron* pada *Input layer*

p = jumlah *neuron* pada *Hidden layer*

$$\beta = 0.7\ \sqrt[3]{3} = 1{,}00957$$

Kemudian menghitung nilai $||v_j||$ yang merupakan panjang vektor pada setiap *Hidden layer* menggunakan persamaan 2.52 sebagai berikut :

$$\left|\left|v_j\right|\right| = \sqrt{v_{j1}^2 + v_{j2}^2 + \cdots + v_{jn}^2}$$

$$||v_1|| = \sqrt{v_{11}^2 + v_{12}^2 + v_{13}^2}$$

$$\left|\left|v_1\right|\right| = \sqrt{(-0.222)^2 + (-0.492)^2 + (-0.305)^2} = 0.6199782$$

$$\left|\left|v_2\right|\right| = \sqrt{(-0.15)^2 + (-0.024)^2 + (-0.123)^2} = 0.195461$$

$$||v_3|| = \sqrt{(0.424)^2 + (-0.197)^2 + (-0.021)^2} = 0.4680021$$

Selanjutnya menghitung bobot yang dipakai sebagai inisialisasi dari *Input layer* ke *Hidden layer* menggunakan persamaan 2.52 sebagai berikut :

$$v_{ji} = \frac{\beta\, v_{ji}}{\left|\left|v_j\right|\right|}$$

Untuk data pertama v_{ji} sebagai berikut :

$$v_{11} = \frac{\beta * v_{11}}{||v_1||} = \frac{1{,}00957 * (-0{,}222)}{0.6199782} = -0.362$$

Berturut-turut persamaan di atas digunakan untuk menghitung bobot baru yang akan digunakan sebagai inisialisasi dari *Input layer* ke *Hidden layer* yang ditampilkan pada Tabel 4.21.

Tabel 4.21 Nilai bobot baru v_{ji}

v_{ji}		*j*		
		1	**2**	**3**
i	**1**	-0.362	-0.775	0.915
	2	-0.801	-0.124	-0.425
	3	-0.497	-0.635	-0.045

Bias awal dari *Input layer* ke *Hidden layer* yang dipakai adalah bilangan acak antara –β dan β. Jadi nilainya adalah antara bilangan -1,00957 dan 1,00957. Nilai v_{j0} dapat dilihat pada Tabel 4.22.

Tabel 4.22 Nilai bias awal v_{j0}

v_{j0}	*j*		
0	**1**	**2**	**3**
	-0.771	0.124	-0.916

Inisialisasi bobot dari *Hidden layer* ke *output layer* (w_{kj}) diperoleh dari proses *random* antara -0,5 dan 0,5. Pemberian *random* bobot ditunjukkan pada Tabel 4.23 di bawah ini.

Tabel 4.23 Bobot dari Hidden layer ke *output* layer (w_{kj})

w_{kj}		***k*=1**
j	**1**	-0.168
	2	-0.427
	3	-0.239

Sedangkan inisialisasi bias dari *Hidden layer* ke *output layer* (w_{k0}) juga diperoleh dari *random* antara -0,5 dan 0,5. Pemberian *random* bobot ditunjukkan pada Tabel 4.24 di bawah ini

Tabel 4.24 Bias dari Hidden layer ke *output* layer (w_{k0})

w_{k0}	***k*=1**
0	0.451

4.3.4. Proses *FeedForward*

Setelah melakukan proses normalisasi data dan penginisialisasian bobot, tahap selanjutnya adalah proses *FeedForward*. Pada metode *Backpropagation* proses ini dilakukan pada langkah ke 3 sampai ke 5 perhitungannya sebagai berikut :

***Langkah* 3** : Tiap unit masukan menerima sinyal dan meneruskannya ke unit tersembunyi di atasnya

***Langkah* 4** : Hitung semua keluaran di unit tersembunyi (Z_i, j=1,2,3,…,p) dengan persamaan 2.38

$$z\_net_j = v_{j0} + \sum_{i=1}^{n} x_i v_{ji}$$

Keterangan :

z_net_j = sinyal masukan pada *Hidden layer* ke – *j*

v_{j0} = bias ke *Hidden layer* ke - *j*

v_{ji} = bobot antara unit *Input* ke – *i* dan *Hidden layer* ke – *j*

x_i = unit *Input* ke – *i*

$$z\_net_1 = v_{10} + x_1 v_{11} + x_2 v_{12} + x_2 v_{13}$$

$$z\_net_1 = -0.771 + (0.351 * (-0.362)) + (0.440 * (-0.801)) + (0.439 * (-0.497)) = -1.468$$

$$z_{net_2} = 0.124 + (0.351 * (-0.775)) + (0.440 * (-0.124)) + (0.439 * (-0.635)) = -0.482$$

$z\_net_3 = -0.916 + (0.351 * (0.915)) + (0.440 * (-0.425)) + (0.439 * (-0.045))$
$= -0.802$

Langkah selanjutnya menghitung aktivasi *Hidden layer* (z_j) dengan menggunakan persamaan 2.39

$$z_j = f\left(z\_net_j\right) = \frac{1}{1+ e^{-z\_net_j}}$$

Keterangan :

$z_{\_net_j}$ = sinyal masukan pada *Hidden layer* ke – *j*

z_j = aktivasi *Hidden layer* ke - *j*

$$z_1 = \frac{1}{1+ e^{-(-1.468)}} = 0.187$$

$$z_2 = \frac{1}{1+ e^{-(-0.482)}} = 0.382$$

$$z_3 = \frac{1}{1+ e^{-(-0.802)}} = 0.310$$

Langkah 5 : Hitung semua keluaran jaringan di unit (y_k, k=1,2,3,…,m) menggunakan persamaan 2.40

$$y\_net_k = w_{k0} + \sum_{j=1}^{p} z_j w_{kj}$$

Keterangan :

y_net_k = sinyal masukan *output* ke – *k*

w_{k0} = bias ke *Hidden layer* ke – *k*

w_{kj} = *Output* ke – *k* dan *Hidden layer* ke – *j*

z_j = aktivasi *Hidden layer* ke - *j*

Nilai didapat dari *Hidden layer* yang berada di lapisan sebelumnya, berikut dijelaskan perhitungan secara manual

$$y\_net_1 = w_{10} + z_1 w_{11} + z_2 w_{12} + z_3 w_{13}$$

$y\_net_1 = 0.451 + (0.187 * (-0.168) + (0.382 * (-0.427)) + \big(0.310 *$
$(-0.239)\big) = 0.183$

Kemudian hitung nilai aktivasi *output* jaringan (y_k) dengan menggunakan persamaan 2.41

$$y_k = f\left(y\_net_k\right) = \frac{1}{1+ e^{-y\_net_k}}$$

Keterangan :

y_k = aktivasi *output* ke – *k*

y_net_k = sinyal masukan *output* ke – *k*

$y_1 = f(y\_net_1) = \frac{1}{1+ e^{-(0.183)}} = 0.546$

4.3.5. Proses Backpropagation

Setelah melakukan proses *FeedForward*, kemudian langkah selanjutnya adalah proses *backpropagation*. Pada metode *Backpropagation* proses ini dilakukan pada langkah ke 6 sampai ke 7 perhitungannya sebagai berikut :

Langkah 6 : Hitung faktor δ unit keluaran berdasarkan kesalahan setiap unit keluaran(y_k, k=1,2,3,…,m) dengan menggunakan persamaan 2.42

$\delta_k = (t_k - y_k) f'(y\_net_k) = (t_k - y_k) y_k (1 - y_k)$

δ_k merupakan unit kesalahan yang akan dipakai dalam perubahan bobot layar di bawahnya (langkah 7)

Keterangan :

δ_k = faktor koreksi *Error* bobot w_{jk}

t_k = target *output* (*desired output*) ke – *k*

y_k = aktivasi *output* ke – *k*

$\delta_k = (t_1 - y_1) y_1 (1 - y_1)$

$\delta_1 = (0.1 - 0.546) * 0.546(1 - 0.546) = -0.110$

Hitung suku perubahan bobot w_{kj} (yang akan dipakai nanti untuk merubah bobot w_{kj}) dengan laju percepatan α = 0,2 menggunakan persamaan 2.43

$\Delta w_{kj} = \alpha \delta_k z_j \quad ; \quad k = 1, 2, \dots, m \quad ; \quad j = 0, 1, \dots, p$

Keterangan :

δ_k = faktor koreksi *Error* bobot w_{kj}

α = laju percepatan (*learning rate*)

Δw_{kj} = nilai koreksi *Error* bobot w_{kj}

z_j = aktivasi *Hidden layer* ke – *j*

$\Delta w_{10} = \alpha \delta_1 z_0 = 0.2 * -0.110 * 1 = -0.022$

$\Delta w_{11} = \alpha \delta_1 z_1 = 0.2 * -0.110 * 0.187 = -0.005$

$\Delta w_{12} = \alpha \delta_1 z_2 = 0.2 * -0.110 * 0.382 = -0.008$

$\Delta w_{13} = \alpha\, \delta_1\, z_3 = 0.2 * -0.110 * 0.310 = -0.007$

Langkah 7 : Hitung faktor δ unit tersembunyi berdasarkan kesalahan di setiap unit tersembunyi (z_j, j=1,2,3,…,p) menggunakan persamaan 2.44

$\delta\_net_j = \sum_{k=1}^{m} \delta_k w_{kj}$

Keterangan :

δ_net_j = jumlah delta bobot *Hidden layer* ke – *j*

δ_k = faktor koreksi *Error* bobot w_{kj}

w_{kj} = bobot antara *output* ke – *k* dan *Hidden layer* ke – *j*

$\delta\_net_1 = \delta_1 w_{11} = -0.110 * -0.168 = 0.019$

$\delta\_net_2 = \delta_1 w_{12} = -0.110 * -0.427 = 0.047$

$\delta\_net_3 = \delta_1 w_{13} = -0.110 * -0.239 = 0.026$

Hitung faktor koreksi *Error* (δ_i) pada *Hidden layer* menggunakan persamaan 2.45

Faktor δ unit tersembunyi :

$\delta_i = \delta\_net_j\, f'\left(z_{net_j}\right) = \delta\_net_j\, z_j\, (1 - z_j)$

Keterangan :

δ_i = faktor koreksi *Error* bobot v_{ij}

δ_net_j = jumlah delta bobot *Hidden layer* ke – *j*

z_j = aktivasi *Hidden layer* ke – *j*

$\delta_1 = \delta\_net_1\, z_1\, (1 - z_1) = 0.019 * 0.187 * (1 - 0.187) = 0.003$

$\delta_2 = \delta\_net_2\, z_2\, (1 - z_2) = 0.047 * 0.382 * (1 - 0.382) = 0.011$

$\delta_3 = \delta\_net_3\, z_3\, (1 - z_3) = 0.026 * 0.310 * (1 - 0.310) = 0.006$

Hitung suku perubahan bobot v_{ji} (yang akan dipakai nanti untuk merubah bobot v_{ji}) dengan menggunakan persamaan 2.46

$\Delta v_{ji} = \alpha\, \delta_i\, x_i$; j = 1, 2, … , p ; i=0,1, … ,n

Keterangan :

Δv_{ji} = nilai koreksi *Error* bobot v_{ji}

α = laju percepatan (*learning rate*)

δ_i = faktor koreksi *Error* bobot v_{ji}

x_i = unit *Input* ke – *i*

Untuk nilai koreksi *Error* bobot Δv_{11}, dilakukan perhitungan seperti berikut :

$\Delta v_{11} = \alpha\, \delta_1\, x_1 = 0.2 * -0.003 * 0.351 = 0.00020$

Berturut-turut persamaan di atas digunakan untuk menghitung nilai koreksi *Error* bobot Δv_{ji} yang ditampilkan pada Tabel 4.25.

Tabel 4.25 Nilai koreksi *Error* bobot Δv_{ji}

Δv_{ji}		***j***		
		1	**2**	**3**
i	**1**	0.00020	0.00078	0.00040
	2	0.00025	0.00098	0.00050
	3	0.00025	0.00098	0.00050

Selain itu, juga dihitung koreksi bias v_{j0} (Δv_{j0}) yang nantinya untuk memperbarui v_{j0}.

$\Delta v_{10} = \alpha\, \delta_1 1 = 0.2 * 0.003 * 1 = 0.00056$

$\Delta v_{20} = \alpha\, \delta_2 1 = 0.2 * 0.011 * 1 = 0.00223$

$\Delta v_{30} = \alpha\, \delta_3 1 = 0.2 * 0.006 * 1 = 0.00113$

4.3.6. Ubah Bobot Dan Bias

Setelah melakukan proses *backpropagation*, tahap selanjutnya adalah proses ubah bobot dan bias yang merupakan proses terakhit dari pelatihan data.

Langkah – langkah dalam proses perubahan bobot dan bias dengan persamaan 2.47 adalah sebagai berikut:

Perubahan bobot garis yang menuju ke unit keluaran :

$w_{kj}\ (baru) = w_{kj}(lama) + \Delta w_{kj}$ (k = 1, 2, …,m ; j=0, 1, … , p)

Keterangan :

$w_{kj}\ (baru)$ = bobot antara *Hidden layer* ke – *j* dan *output* ke – *k* baru

$w_{kj}(lama)$ = bobot antara *Hidden layer* ke – *j* dan *output* ke – *k* lama

Δw_{kj} = nilai koreksi *Error* bobot w_{jk}

$w_{11}\ (baru) = w_{11}(lama) + \Delta w_{11} = -0.168 + -0.004 = -0.17213$

$w_{12}\ (baru) = w_{12}(lama) + \Delta w_{12} = -0.427 + -0.008 = -0.43543$

$w_{13}\ (baru) = w_{13}(lama) + \Delta w_{13} = -0.239 + -0.007 = -0.24584$

Bias menuju *output* ke – *k* baru dihitung dengan rumus :

$w_{10}\ (baru) = w_{10}(lama) + \Delta w_{10} = 0.451 + -0.022 = 0.42891$

Perubahan bobot garis yang menuju ke unit tersembunyi :

$v_{ji}\ (baru) = v_{ji}(lama) + \Delta v_{ji}$ (k = 1, 2, …,m ; j=0, 1, … , p)

Keterangan :

$v_{ji}\ (baru)$ = bobot antara unit *Hidden layer* ke – *j* dan *Input* ke – *i* baru

$v_{ji}(lama)$ = bobot antara unit *Hidden layer* ke – *j* dan *Input* ke – *i* lama

Δv_{ji} = nilai koreksi *Error* bobot v_{ji}

Untuk bobot v_{11} (baru) dilakukan perhitungan seperti berikut .

$v_{11}\ (baru) = v_{11}(lama) + \Delta v_{11} = -0.362 + 0.00020 = -0.361$

Berturut- turut persamaan diatas digunakan untuk menghitung bobot v_{ji} (baru) yang ditampilkan pada Tabel 4.26.

Tabel 4.26 Nilai bobot v_{ji} baru

v_{ji}		*j*		
		1	**2**	**3**
i	**1**	-0.361	-0.774	0.915
	2	-0.801	-0.123	-0.424
	3	-0.496	-0.634	-0.045

Bias menuju *Hidden layer* ke – i baru dihitung dengan rumus :

$v_{j0}\ (baru) = v_{j0}(lama) + \Delta v_{j0}$

$v_{10}\ (baru) = v_{10}(lama) + \Delta v_{10} = -0.771 + 0.001 = -0.770$

$v_{20}\ (baru) = v_{20}(lama) + \Delta v_{20} = 0.124 + 0.002 = 0.126$

$v_{30}\ (baru) = v_{30}(lama) + \Delta v_{30} = -0.916 + 0.001 = -0.915$

Berturut-turut akan ditampilkan bobot w_{kj} baru, w_{k0} baru, v_{ji} baru dan v_{j0} baru yang diperoleh dari data latih ke 1 yang ditampilkan pada Tabel 4.27, 4.28 , 4.29 dan 4.30.

Tabel 4.27 Nilai bobot w_{kj} baru data latih ke- 1

w_{kj}		**k = 1**
j	**1**	-0.172

	2	-0.435
	3	-0.246

Tabel 4.28 Nilai bias w_{k0} baru data latih ke- 1

w_{k0}	$k = 1$
0	0.429

Tabel 4.29 Nilai bobot v_{ji} baru data latih ke- 1

v_{ji}	j		
	1	**2**	**3**
i **1**	-0.361	-0.774	0.915
2	-0.801	-0.123	-0.424
3	-0.496	-0.634	-0.045

Tabel 4.30 Nilai bias v_{j0} baru data latih ke- 1

v_{j0}	j		
0	**1**	**2**	**3**
	-0.770	0.126	-0.915

Bobot w_{kj} baru, w_{k0} baru, v_{ji} baru dan v_{j0} baru diatas merupakan bobot yang diperoleh pada data latih ke -1. Hitung kembali data latih ke -2 sampai data latih ke –10 sesuai dengan contoh perhitungan data latih ke-1 hingga mendapatkan 1 iterasi pelatihan hingga didapat bobot optimal yang akan digunakan dalam proses identifikasi penyakit citra buah jeruk. Bobot w_{kj} baru, w_{k0} baru, v_{ji} baru dan v_{j0} baru yang dihasilkan setelah dilakukan perhitungan 1 iterasi berturut-turut ditampilkan pada Tabel 4.31 , 4.32 , 4.33 , dan 4.34.

Tabel 4.31 Nilai bobot w_{kj}

w_{kj}	k=1	
j	**1**	-0.182
	2	-0.451
	3	-0.246

Tabel 4.32 Nilai bias w_{k0}

w_{k0}	***k*=1**
0	0.416

Tabel 4.33 Nilai bobot v_{ji}

v_{ji}		***j***		
		1	**2**	**3**
i	**1**	-0.362	-0.776	0.913
	2	-0.801	-0.125	-0.426
	3	-0.497	-0.636	-0.046

Tabel 4.34 Nilai bias v_{j0}

v_{j0}	***j***		
0	**1**	**2**	**3**
	-0.769	0.128	-0.914

Dengan menggunakan bobot tersebut dapat dilakukan iterasi kedua dengan langkah perhitungan yang sama pada iterasi pertama namun menggunakan bobot yang diperoleh pada iterasi pertama. Berikut hasil bobot iterasi kedua berturut-turut ditampilkan pada Tabel 4.35, 4.36, 4.37, 4.38.

Tabel 4.35 Nilai bobot w_{kj}

w_{kj}	***k*=1**	
j	**1**	-0.1953
	2	-0.4721
	3	-0.2522

Tabel 4.36 Nilai bias w_{k0}

w_{k0}	***k*=1**
0	0.3875

Tabel 4.37 Nilai bobot v_{ji}

v_{ji}		*j*		
		1	**2**	**3**
i	**1**	-0.3621	-0.8016	-0.4968
	2	-0.8016	-0.4968	0.1320
	3	-0.4968	0.1320	-0.7783

Tabel 4.38 Nilai bias v_{j0}

v_{j0}	*j*		
0	**1**	**2**	**3**
	-0.7675	-0.3621	-0.8016

4.3.7. Memeriksa Stop Condition

Pada penelitian ini *Stop Condition* yang digunakan adalah jumlah maksimum iterasi serta minimum *Error* dengan menggunakan metode *Mean Squared Error*(MSE). Jika *Stop Condition* telah terpenuhi, maka pelatihan jaringan dapat dihentikan. Setelah pelatihan selesai dilakukan, didapatkan bobot dan bias yang dapat dipakai untuk perhitungan data pengujian. Dalam hal ini, hanya proses *FeedForward* saja yang digunakan untuk perhitungan terhadap penentuan identifikasi penyakit terhadap data uji.

Karena pada perhitungan manual ini hanya dilakukan 1 kali iterasi maka proses pada penelitian ini berhenti pada iterasi pertama, setelah mencapai 1 kali iterasi dihitung rata-rata *Error* menggunakan *Mean Squared Error* (MSE). Pada Tabel 4.39. ditampilkan perbandingan nilai target dan nilai *output* hasil keluaran jaringan.

Tabel 4.39 Nilai target dan nilai ouput jaringan (Y)

Data ke -	Target	*Y*
1	0,1	0.546
2	0,1	0.523
3	0,1	0.516
4	0,5	0.513
5	0,5	0.532

6	0,5	0.535
7	0,9	0.532
8	0,9	0.517
9	0,9	0.533
10	0,9	0.544

Perhitungan rata-rata *Error* menggunakan Mean Squared *Error* (MSE) dengan cara persamaan 2.43

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(t_k - y_k)^2$$

Keterangan :

t_k = nilai *output* target

y_k = nilai *output* jaringan

N = jumlah *output* dari *neuron*

$$MSE = \frac{\begin{matrix}(0.1 - 0.546)^2 + (0.1 - 0.523)^2 + (0.1 - 0.516)^2 + (0.1 - 0.513)^2 + (0.5 - 0.532)^2 \\ +(0.5 - 0.535)^2 + (0.5 - 0.532)^2 + (0.9 - 0.517)^2 + (0.9 - 0.533)^2 + (0.9 - 0.544)^2\end{matrix}}{10} = 0.113$$

4.3.8. Proses Pengujian

Pada proses pengujian, langkah-langkah yang digunakan pada metode *backpropagation* adalah normalisasi data uji ,proses *FeedForward* serta denormalisasi nilai target dan *ouput* jaringan tersebut untuk mengetahui hasil dari klasifikasi. Berikut data uji yang digunakan ditampilkan pada Tabel 4.40 dan data uji ternormalisasi pada Tabel 4.41. Sedangkan bobot yang digunakan pada Tabel 4.35 , 4.36 , 4.37 , dan 4.38.

Tabel 4.40 Data uji yang digunakan

Data ke-	Rata-Rata			Target Seharusnya
	R	G	B	
1	170.313	167.095	121.247	2
2	151.963	169.467	158.177	2
3	86.151	99.956	92.484	1
4	116.200	110.400	83.385	3
5	145.641	158.079	145.286	2

Tabel 4.41 Data uji setelah dilakukan normalisasi

Data ke-	Rata-Rata			Target(*Y*)
	X1 (R)	*X2* (G)	*X3* (B)	
1	0.836	0.884	0.641	0.5
2	0.731	0.900	0.900	0.5
3	0.351	0.440	0.439	0.1
4	0.524	0.509	0.376	0.9
5	0.694	0.825	0.810	0.5

Tabel 4.42 Hasil pengujian menggunakan proses *Forward* z_net

Data ke -	*z_net1*	*z_net2*	*z_net3*
1	-2.098	-1.040	-0.559
2	-2.201	-1.124	-0.674
3	-1.466	-0.477	-0.802
4	-1.552	-0.580	-0.671
5	-2.082	-1.028	-0.671

Tabel 4.43 Hasil pengujian menggunakan proses *Forward* (*zj*)

Data ke -	*z1*	*z2*	*z3*
1	0.109	0.261	0.364
2	0.100	0.245	0.338
3	0.188	0.383	0.310
4	0.175	0.359	0.338
5	0.111	0.263	0.338

Tabel 4.44 Hasil pengujian menggunakan proses *Forward* y_net_k dan y_k

Data ke -	*y_net1*	*y1*
1	0.151	0.538
2	0.167	0.542

3	0.092	0.523
4	0.099	0.525
5	0.156	0.539

Kemudian melakukan denormalisasi pada nilai *output* jaringan (Y) sesuai dengan target data latih untuk memperoleh hasil klasifikasi dengan persamaan 2.40 berikut ini

$$y' = \frac{x'-(0,1)}{0,8}(max - min) + min$$

Keterangan :

y = data setelah dinormalisasi

y' = data setelah didenormalisasi

x = data sebelum dinormalisasi

x' = data sebelum didenormalisasi

max = nilai data terbesar

min = nilai data terkecil

$$y^1 = \frac{x^1-(0,1)}{0,8}(max - min) + \min = \frac{0.538-(0,1)}{0,8}(3-1) + 1 = 2.094 = 2\ (dibulatkan)$$

Berturut-turut persamaan di atas dikerjakan untuk memperoleh denormalisasi *output* data uji ke 1 sampai ke 5 berikut hasil dari denormalisasi pada Tabel 4.45 data uji ke 1 sampai ke 5 :

Tabel 4.45 Hasil denormalisasi

Data ke-	*Y*	(*Y*) Dibulatkan
1	2.094	2
2	2.104	2
3	2.057	2
4	2.062	2
5	2.097	2

Dari hasil denormalisasi diperoleh hasil klasifikasi penyakit pada pembulatan Y *output*, berikut hasil dari klasifikasi dari Tabel 4.46.

Tabel 4.46 Hasil klasifikasi sistem

Data ke-	*Y*	(*Y*) Dibulatkan	Penyakit/Hama
1	2.094	2	Trips
2	2.104	2	Trips
3	2.057	2	Trips
4	2.062	2	Trips
5	2.097	2	Trips

Untuk mengetahui hasil akurasi dengan data sebenarnya menggunakan persamaan 2.44

$$Tingkat\ Akurasi\ (\%) = \frac{Jumlah\ Sampel\ Pengujian\ Benar}{Jumlah\ Sampel}\ x\ 100\%$$

Tabel 4.47 Tabel kebenaran antara sistem dengan hasil sebenarnya

Data ke-	Hasil Seharusnya	Hasil Sistem
1	2	2
2	2	2
3	1	2
4	3	2
5	2	2

Berdasarkan dari Tabel 4.47 dapat dihitung tingkat akurasi sistem sebagai berikut :

$$Tingkat\ Akurasi\ (\%) = \frac{3}{5}\ x\ 100\% = 60\%$$

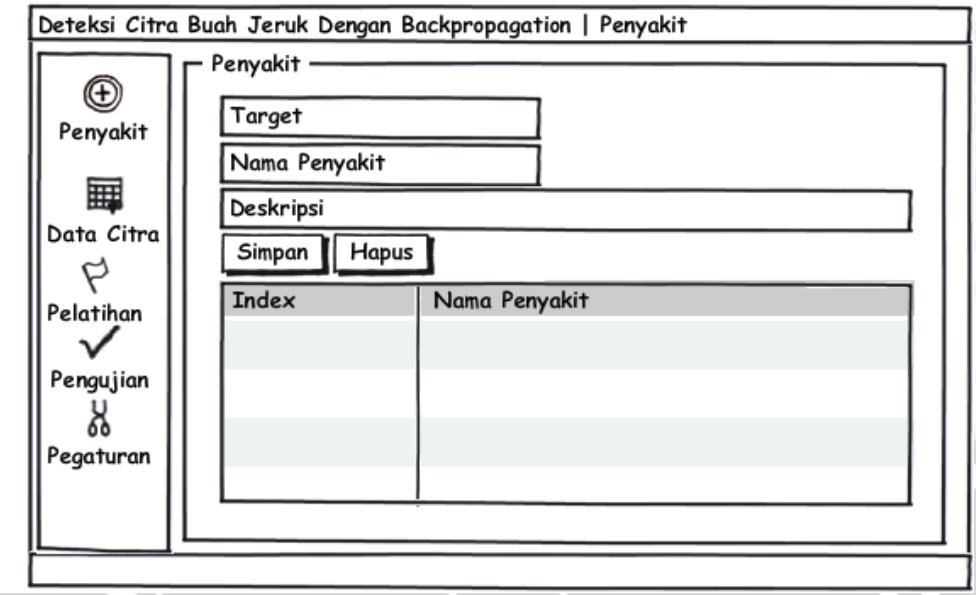
Atau bisa dikatakan tingkat akurasi sistem adalah 60%.

4.4. Perancangan Antarmuka

Untuk mempermudah interaksi antara user dengan sistem dibutuhkan antarmuka untuk menjembatani komunikasi antara user dengan sistem. Pada aplikasi deteksi penyakit pada citra buah tanaman jeruk terdapat 5 menu yaitu : *Input* data penyakit, *Input* data latih, pelatihan, pengujian, serta pengaturan. Berikut dijelaskan 5 menu yang telah disebutkan sebelumnya.

4.4.1. *Input* Data Penyakit

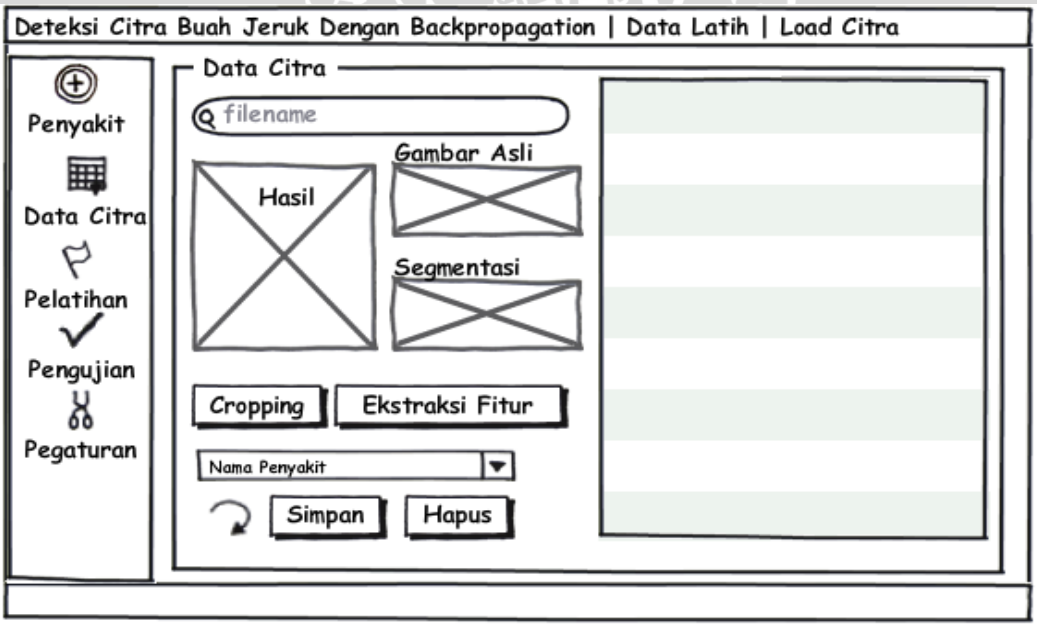
Antarmuka *Input* data penyakit digunakan untuk melakukan penambahan penyakit ,desain perancangan ditunjukkan pada Gambar 4.29.



Gambar 4. 29 Antarmuka *Input* Data Penyakit

4.4.2. *Input* Data Latih

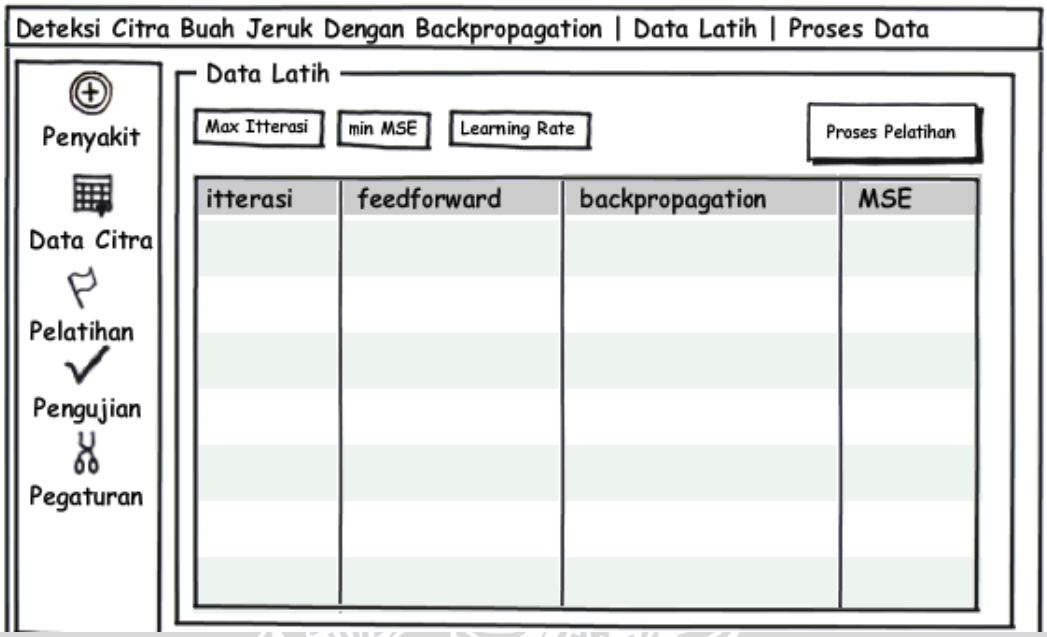
Input data latih digunakan untuk menambahkan citra yang akan digunakan dalam proses pelatihan. Pada antarmuka ini terdapat fasilitas untuk melakukan *preprocessing* citra serta ekstraksi fitur sebelum dilakukan penyimpanan. Desain *Input* Data Latih ditunjukkan pada Gambar 4.30



Gambar 4. 30 Antarmuka *Input* Data Latih

4.4.3. Pelatihan

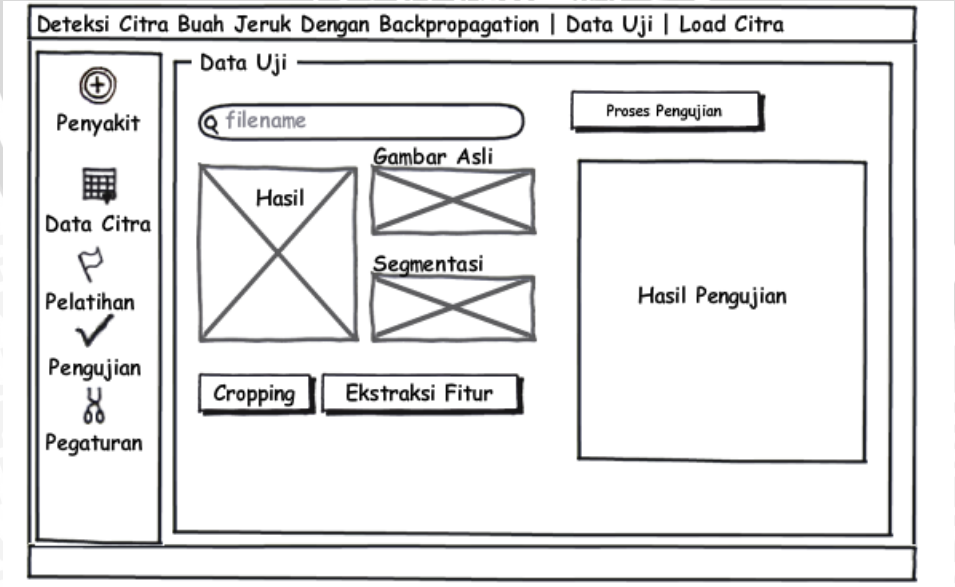
Antarmuka pelatihan digunakan untuk proses pelatihan dengan JST *Backpropagation*, desain antarmukanya ditunjukkan pada Gambar 4.31.



Gambar 4.31 Antarmuka Pelatihan

4.4.4. Pengujian

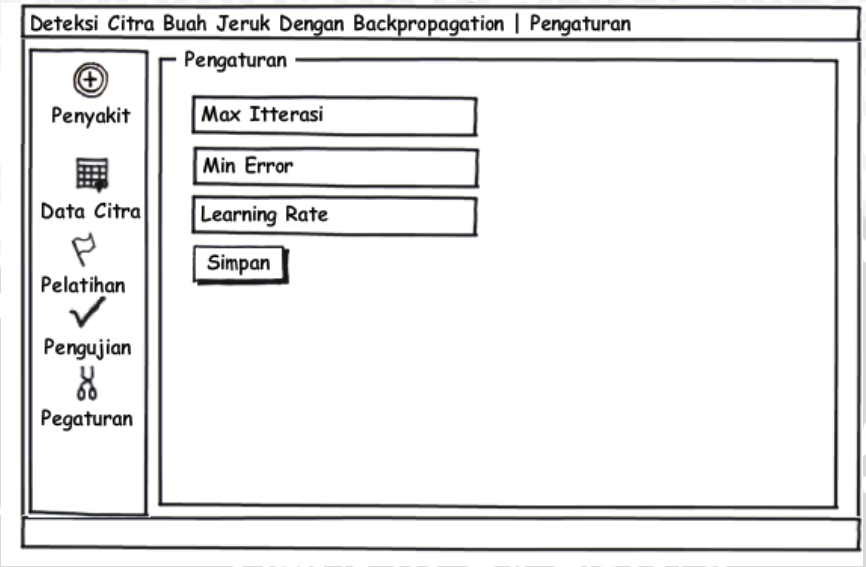
Antarmuka pengujian digunakan untuk proses pengujian dengan JST *Backpropagation*, desain antarmukanya ditunjukkan pada Gambar 4.32.



Gambar 4.32 Antarmuka Pengujian

4.4.5. Pengaturan

Antarmuka pengaturan digunakan untuk *Input* pengaturan JST *Backpropagation* berupa *Max* Itterasi, *MinError* dan *Learning Rate*, desain antarmukanya ditunjukkan pada Gambar 4.33.



Gambar 4.33 Antarmuka Pengaturan

4.5. Skenario Pengujian

Skenario pengujian dilakukan untuk mengetahui parameter apa saja yang mempengaruhi tingkat akurasi dalam deteksi penyakit pada citra buah tanaman jeruk. Berikut beberapa macam skenario pengujian yang akan dilakukan pada penelitian ini.

4.5.1. Pengujian Jenis Ekstraksi Fitur Terhadap Tingkat Akurasi

Pengujian jenis ekstraksi warna terhadap tingkat akurasi .Tiap Iterasi akan dilakukan percobaan sebanyak 5 kali. Nilai *learning rate* yang akan digunakan adalah 0,2 serta jumlah iterasi sebanyak 500 dengan bobot random yang berbeda menggunakan metode *Nguyen-widrow* pada tiap percobaan. Tabel pengujian ditampilkan pada Tabel 4.48.

Tabel 4.48 Pengujian Jenis Ekstraksi Fitur Terhadap Tingkat Akurasi

No	Ekstraksi Fitur	Percobaan ke-*i*					Rata-Rata Akurasi
		1	2	3	4	5	
1	*RGB*						
2	*CMYK*						
3	*YUV*						

4	*RGBCMYK*						
5	*RGBYUV*						

4.5.2. Pengujian *Threshold Otsu* Terhadap Tingkat Akurasi

Pengujian *threshold otsu* terhadap tingkat akurasi akan dilakukan dengan menambahkan atau mengurangi nilai *threshold otsu* mulai -50 sampai +50, sedangkan tiap Iterasi akan dilakukan percobaan sebanyak 5 kali. Nilai *learning rate* yang akan digunakan adalah 0,2 serta jumlah iterasi sebanyak 500 dengan bobot random yang berbeda menggunakan metode *Nguyen-widrow* pada tiap percobaan. Tabel pengujian ditampilkan pada Tabel 4.49.

Tabel 4.49 Pengujian Threshold *Otsu* Terhadap Tingkat Akurasi

No	*Threshold*	Percobaan ke-*i*					Rata-Rata Akurasi
		1	2	3	4	5	
1	-50						
2	-40						
3	-30						
4	-20						
5	-10						
6	0						
7	+10						
8	+20						
9	+30						
10	+40						
11	+50						

4.5.3. Pengujian Jumlah Persentase Data Latih dan Data Uji Terhadap Tingkat Akurasi

Pengujian jumlah persentase data latih dan data uji terhadap tingkat akurasi akan dilakukan dengan jumlah perbandingan persentase data latih dan data uji, nilai *learning rate* dan jumlah iterasi mengacu pada hasil terbaik pada skenario pengujian sebelumnya tiap Iterasi akan dilakukan percobaan sebanyak 5 kali. Nilai *learning rate* yang akan digunakan adalah 0,2 dengan bobot random yang berbeda menggunakan

metode *Nguyen-widrow* pada tiap percobaan. Tabel pengujian ditampilkan pada Tabel 4.50.

Tabel 4.50 Pengujian Jumlah Data Latih Terhadap Tingkat Akurasi

No	% Data Latih dan Data Uji	Percobaan ke-*i*					Rata-Rata Akurasi
		1	2	3	4	5	
1	90 % dan 10 %						
2	80 % dan 20 %						
3	70 % dan 30 %						
4	60 % dan 40 %						
5	50 % dan 50 %						
6	40 % dan 60 %						
7	30 % dan 70 %						
8	20 % dan 80 %						
9	10 % dan 90%						

4.5.4. Pengujian Iterasi Maksimum Terhadap Tingkat Akurasi

Pengujian Iterasi maksimum terhadap tingkat akurasi akan dilakukan dengan nilai Iterasi mulai 100 sampai 1000, sedangkan tiap Iterasi akan dilakukan percobaan sebanyak 5 kali. Nilai *learning rate* yang akan digunakan adalah 0,2 dengan bobot random yang berbeda menggunakan metode *Nguyen-widrow* pada tiap percobaan. Tabel pengujian ditampilkan pada Tabel 4.51.

Tabel 4.51 Pengujian Iterasi Maksimum Terhadap Tingkat Akurasi

No	Maksimum Iterasi	Percobaan ke-*i*					Rata-Rata
		1	2	3	4	5	
1	100						
2	200						
3	300						
4	400						
5	500						
6	600						

7	700						
8	800						
9	900						
10	1000						

4.5.5. Pengujian *Learning Rate* Terhadap Tingkat Akurasi

Pengujian *Learning Rate* terhadap tingkat akurasi akan dilakukan dengan nilai *learning rate* mulai 0,01 sampai 0,7 , dengan iterasi mengacu pada hasil terbaik pada skenario pengujian iterasi sebelumnya tiap Iterasi akan dilakukan percobaan sebanyak 5 kali dengan bobot random yang berbeda menggunakan metode *Nguyen-widrow* pada tiap percobaan. Tabel pengujian ditampilkan pada Tabel 4.52.

Tabel 4.52 Pengujian Learning Rate Terhadap Tingkat Akurasi

No	*Learning Rate*	Percobaan ke-*i*					Rata-Rata Akurasi
		1	2	3	4	5	
1	0.01						
2	0.03						
3	0.05						
4	0.07						
5	0.09						
6	0.1						
7	0.3						
8	0.5						
9	0.7						
10	0.9						

BAB V IMPLEMENTASI

Pada bab implementasi akan dibahas mengenai penerapan sistem yang telah dilakukan dari proses perancangan yang telah dijelaskan pada bab sebelumnya. Bab ini terdiri dari, implementasi program dan implementasi antarmuka.

5.1 Implementasi Program

Pada Sub bab ini akan dijelaskan implementasi program sesuai perancangan pada bab sebelumnya.

5.1.1 Proses *Load Single File*

Proses *Load Single File* ini berfungsi untuk melakukan pemanggilan awal pada data citra per satu file yang digunakan untuk *preprocessing* serta ekstraksi fitur untuk proses selanjutnya. Proses *Load Single File* ditunjukkan pada *Source Code* 5.1.

```
1   public void uploadgambar(JLabel targetPreview, JTextField
2   targetfilepath, JPanel targetjpanel, int maxfile) {
3   //Proses Inisialisasi Komponen Dialog upload gambar
4           ResetVariabel();
5           JFileChooser chooser = new JFileChooser();
6           ImagePreviewPanel preview = new ImagePreviewPanel();
7           chooser.setAccessory(preview);
8           chooser.addPropertyChangeListener(preview);
9           chooser.removeChoosableFileFilter(chooser.getFileFilter());
10          chooser.addChoosableFileFilter(new ImageFileFilter());
11          int returnVal = chooser.showOpenDialog(null);
12          Mat uploadgambar = new Mat();
13  //Proses mengecek keberadaan file
14          if (returnVal == JFileChooser.APPROVE_OPTION) {
15              file = chooser.getSelectedFile();
16  //Proses mengecek ukuran file
17              if (file.length() < maxfile) {
18                  setPath(file.getPath());
19                  uploadgambar = Imgcodecs.imread(getPath());
20  //proses inisialisasi file yang telah diupload ke variable citra
21                  setImage(uploadgambar);
22                  targetPreview.setIcon(new ImageIcon(getPath()));
23                  targetfilepath.setText(getPath());
24              } else {
25                  JOptionPane.showMessageDialog(targetjpanel, "File
26  terlalu besar , harus di bawah " + maxfile / 1000000 + " MB ...!!");
27                  targetfilepath.setText("");
28                  targetPreview.setIcon(null);
29              }
            }
        }
```

Source Code 5.1 Proses *Load Single File*

Penjelasan dari *Source Code* 5.1 adalah sebagai berikut :

1. Baris 3 - 12 merupakan proses inisialisasi komponen dialog upload gambar.
2. Baris 14 - 15 merupakan proses mengecek keberadaan file.

3. Baris 17 - 19 merupakan proses mengecek ukuran file.
4. Baris 21 - 23 merupakan proses inisialisasi file yang telah di upload ke variable citra.

5.1.2 Proses *Load Multiple File*

Proses *Load Multiple File* ini berfungsi untuk melakukan pemanggilan awal pada data citra banyak file yang digunakan untuk *preprocessing* serta ekstraksi fitur untuk proses selanjutnya. Proses *Load Multiple File* ditunjukkan pada *Source Code* 5.2.

```
1   public void uploadgambarmultiple(ZebraJTable ZJT, int control) {
2   //Proses inisialisasi komponen dialog upload
3           JenisPenyakit jp = new JenisPenyakit();
4           int otsuside = 0;
5           TableModelMultipleFile tmmf;
6           MultipleFile mf = new MultipleFile();
7           List<MultipleFile> ltmmf = new ArrayList<MultipleFile>();
8           JFileChooser chooser = new JFileChooser();
9           ImagePreviewPanel preview = new ImagePreviewPanel();
10          chooser.setMultiSelectionEnabled(true);
11          chooser.setAccessory(preview);
12          chooser.addPropertyChangeListener(preview);
13          chooser.removeChoosableFileFilter(chooser.getFileFilter());
14          chooser.addChoosableFileFilter(new ImageFileFilter());
15          int returnVal = chooser.showOpenDialog(null);
16  //Proses mengecek keberadaan file
17          if (returnVal == JFileChooser.APPROVE_OPTION) {
18              files = chooser.getSelectedFiles();
19  //Proses definisi multi file ke variable citra
20              for (int i = 0; i < files.length; i++) {
21                  ResetVariabel();
22                  String[] parts = files[i].getName().split("-");
23                  if        (parts.length        ==        1        ||
24  jp.Caritarget(Integer.valueOf(parts[0].toString()))==0) {
25                      continue;
26                  }else{
27                      otsuside                                          =
28  jp.CariOtsuSide(Integer.valueOf(parts[0].toString()));
29                  }
30                  Mat uploadgambar = new Mat();
31                  Mat hasilekstraksi = new Mat();
32                  uploadgambar = Imgcodecs.imread(files[i].toString());
33
34                  if (otsuside == 1) {
35                      EkstraksiFitur(uploadgambar, control*(-1));
36                      hasilekstraksi                                    =
37  HasilSegmentasiTerang(uploadgambar, getImageekstraksi());
38                  } else if (otsuside == 0) {
39                      EkstraksiFitur(uploadgambar, control);
40                      hasilekstraksi = HasilSegmentasiGelap(uploadgambar,
41  getImageekstraksi());
42                  }
43                  MultipleFile  temp  =  mf.setnilai(files[i].toString(),
44  createAwtImage(uploadgambar),     createAwtImage(getImageekstraksi()),
45  createAwtImage(hasilekstraksi),Integer.parseInt(parts[0].toString()),j
46  p.CariNamaPenyakit(Integer.parseInt(parts[0].toString())));
47                  ltmmf.add(temp);
48              }
49
50          }
```

```
51          tmmf = new TableModelMultipleFile(ltmmf);
52          ZJT.setDefaultRenderer(Image.class,                    new
53  ImageTableCellRenderer1());
54          ZJT.setDefaultRenderer(JComboBox.class,                new
55  ComboBoxTableCellRenderer(jp.getItemJenisPenyakit()));
56          ZJT.setEnabled(true);
57          ZJT.setModel(tmmf);
58          ZJT.setGridColor(Color.getHSBColor(0, 0, 40));
59      }
```

Source Code 5.2 Proses *Load Multiple File*

Penjelasan dari *Source Code* 5.2 adalah sebagai berikut :

1. Baris 3 - 15 merupakan proses inisialisasi komponen dialog upload gambar.
2. Baris 17 merupakan proses mengecek keberadaan file.
3. Baris 20 - 58 merupakan definisi *multi* file ke variabel citra .

5.1.3 *Cropping* Citra Buah Jeruk

Cropping Citra Buah Jeruk berfungsi untuk melakukan *preprocessing* pengambilan citra berpenyakit dengan memotong bagian citra yang terdeteksi. *Cropping* Citra Buah Jeruk ditunjukkan pada *Source Code* 5.3.

```
1       public Mat CropProcess(int coordX, int coordY, Mat matsrc, int
2   cropWidth, int cropHeight) {
3   //Proses inisialisasi ukuran cropping serta posisi cropping
4           Mat   CropProcess   =   new   Mat(cropHeight,   cropWidth,
5   matsrc.type(), new Scalar(0));
6           int x = 0;
7           int y = 0;
8   //Proses pemotongan gambar sesuai ukuran
9           for (y = 0; y < cropHeight; y++) {
10              for (x = 0; x < cropWidth; x++) {
11                  double[] dat = matsrc.get(y + coordY, x + coordX);
12                  CropProcess.put(y, x, dat);
13              }
14          }
15  //Proses penyimpanan hasil cropping ke variable citra
16          setImage(CropProcess);
17          return CropProcess;
18      }
```

Source Code 5.3 *Cropping* Citra Buah Jeruk

Penjelasan dari *Source Code* 5.3 adalah sebagai berikut :

1. Baris 4 - 7 merupakan proses inisialisasi ukuran *cropping* serta posisi cropping.
2. Baris 9 - 14 merupakan proses pemotongan gambar sesuai ukuran.
3. Bari 16 - 18 merupakan proses penyimpanan hasil *cropping* ke variabel citra.

5.1.4 Konversi Data Citra ke *Grayscale*

Konversi data citra ke *Grayscale* berfungsi untuk melakukan pengubahan citra yang telah dilakukan *preprocessing* ke dalam citra keabuan. Konversi Data Citra ke *Grayscale* pada *Source Code* 5.4.

```
1   public Mat ConvertGrayscale(Mat ImgSrc) {
2   //Proses inisialisasi citra grayscale
3           Mat grayscale = new Mat(ImgSrc.height(), ImgSrc.width(),
4   ImgSrc.type(), new Scalar(0));
5   //Proses konversi tiap pixel citra asli ke dalam bentuk grayscale
6           for (int y = 0; y < ImgSrc.height(); y++) {
7               for (int x = 0; x < ImgSrc.width(); x++) {
8                   double[] data = ImgSrc.get(y, x);
9                   double Y = (0.299 * data[2]) + (0.587 * data[1]) +
10  (0.114 * data[0]);
11                  Y = Math.round(Y);
12                  data[0] = Y;
13                  data[1] = Y;
14                  data[2] = Y;
15                  grayscale.put(y, x, data);
16              }
17          }
18          return grayscale;
19      }
```

Source Code 5.4 Konversi Data Citra ke *Grayscale*

Penjelasan dari *Source Code* 5.4 adalah sebagai berikut :

1. Baris 3 merupakan proses inisialisasi citra *grayscale*.
2. Baris 6 - 18 merupakan proses konversi tiap *pixel* pada citra asli ke dalam bentuk citra *grayscale.*

5.1.5 Perhitungan nilai histogram *Grayscale*

Perhitungan nilai histogram *Grayscale* merupakan proses untuk menghitung nilai kemunculan aras keabuan. Perhitungan nilai histogram *Grayscale* ditunjukkan pada *Source Code* 5.5.

```
1   public double[] Histogram(Mat ImgSrc, double jml_pixel) {
2   //Proses inisialisasi ukuran array histogram
3           double[] histData = new double[256];
4   //Proses pengambilan frekuensi citra ke dalam variable histogram
5           for (int i = 0; i < ImgSrc.height(); i++) {
6               for (int j = 0; j < ImgSrc.width(); j++) {
7                   double[] data = ImgSrc.get(i, j);
8                   histData[(int) data[1]]++;
9               }
10          }
            return histData;
    }
```

Source Code 5.5 Perhitungan nilai Histogram *Grayscale*

Penjelasan dari *Source Code* 5.5 adalah sebagai berikut :

1. Baris 3 merupakan proses inisialisasi ukuran array histogram.
2. Baris 17 merupakan proses pengambilan kemunculan aras keabuan dan menyimpannya ke variabel histogram.

5.1.6 Perhitungan metode Otsu

Perhitungan metode otsu merupakan proses segmentasi untuk mengambil penyakit pada tanaman buah jeruk. Perhitungan metode *Otsu* ditunjukkan pada *Source Code* 5.6.

```
1   public static double otsu(double[] ihist, double total) {
2           double[] p = new double[256];
3   //Proses Mencari nilai probabilitas intensitas dalam histogram
4           for (int i = 0; i <= 255; i++) {
5               p[i] = ihist[i] / total;
6           }
7   //Proses mencari rerata total
8           double mT = 0;
9           for (int i = 0; i <= 255; i++) {
10              mT = mT + i * p[i];
11          }
12
13  //Proses pembobotan kedua kelas
14          double ambang = 0;
15          double varMaks = 0;
16          for (int t = 0; t <= 255; t++) {
17              double w1 = 0.0;
18              for (int i = 0; i <= t; i++) {
19                  w1 = w1 + p[i];
20              }
21              double w2 = 0.0;
22              for (int i = t + 1; i <= 255; i++) {
23                  w2 = w2 + p[i];
24              }
25  //Proses menghitung rerata kedua kelas
26              double m1 = 0.0;
27              for (int i = 0; i <= t; i++) {
28                  if (w1 > 0) {
29                      m1 = m1 + i * p[i] / w1;
30                  }
31              }
32
33              double m2 = 0.0;
34              for (int i = t + 1; i <= 255; i++) {
35                  if (w2 > 0) {
36                      m2 = m2 + i * p[i] / w2;
37                  }
38              }
39
40  //Proses menghitung BCV
41              double bcv = w1 * Math.pow((m1 - mT), 2) + w2 *
42  Math.pow((m2 - mT), 2);
43  //Proses mencari nilai maksimal BCV untuk menentukan threshold
44              if (bcv > varMaks) {
45                  varMaks = bcv;
46                  ambang = t;
47              }
48          }
49          return ambang;
        }
```

Source Code 5.6 Perhitungan metode Otsu

Penjelasan dari *Source Code* 5.6 adalah sebagai berikut :

1. Baris 4 - 6 merupakan proses mencari nilai probabilitas intensitas dalam histogram.
2. Baris 8 - 11 merupakan proses mencari rerata total.
3. Baris 14 - 24 merupakan proses pembobotan kedua kelas.
4. Baris 26 - 38 merupakan proses menghitung rerata kedua kelas.
5. Baris 40 merupakan proses menghitung BCV .
6. Baris 43 - 48 merupakan proses perhitungan mencari nilai maksimal dari BCV untuk menentukan nilai *threshold* otsu.

5.1.7 Pengambilan Citra Buah Berpenyakit

Pengambilan Citra Buah berpenyakit dilakukan dengan melakukan binerisasi citra dan selanjutnya mengubah nilai biner yang dipilih dengan *RGB*. Pengambilan Citra Buah Berpenyakit ditunjukkan pada *Source Code* 5.7.

```
1   public void EkstraksiFitur(Mat ImgSrc, int Control) {
2   //Proses inisialisasi variable ektraksi fitur
3           Mat ekstraksifitur = new Mat(ImgSrc.height(), ImgSrc.width(),
4   ImgSrc.type(), new Scalar(0));
5           Mat grayscale = new Mat();
6           ConvertGrayscale(ImgSrc).copyTo(grayscale);
7           double jml_pixel = ImgSrc.height() * ImgSrc.width();
8           double ambang = otsu(Histogram(grayscale, jml_pixel), jml_pixel)
9   + Control;
10  //Proses binerisasi citra grayscale terhadap threshold otsu
11          for (int y = 0; y < ImgSrc.height(); y++) {
12              for (int x = 0; x < ImgSrc.width(); x++) {
13                  double[] data = grayscale.get(y, x);
14                  if (data[0] > ambang) {
15                      data[0] = 0.0;
16                      data[1] = 0.0;
17                      data[2] = 0.0;
18                  } else {
19                      data[0] = 255.0;
20                      data[1] = 255.0;
21                      data[2] = 255.0;
22                  }
23                  ekstraksifitur.put(y, x, data);
24
25              }
26          }
27          setImageekstraksi(ekstraksifitur);
28      }
29
30  //Method untuk mengambil citra penyakit jika nilai otsu = 1 atau 255
31  public Mat HasilSegmentasiTerang(Mat ImgSrc, Mat ImgEkstraksi) {
32          Mat segmentasiterang = new Mat(ImgSrc.height(), ImgSrc.width(),
33  ImgSrc.type(), new Scalar(0));
34          double total_pixel = 0;
35          for (int y = 0; y < ImgSrc.height(); y++) {
36              for (int x = 0; x < ImgSrc.width(); x++) {
37                  double[] dataImgSrc = ImgSrc.get(y, x);
38                  double[] dataEkstraksi = ImgEkstraksi.get(y, x);
39                  if (dataEkstraksi[0] == 255) {
40                      segmentasiterang.put(y, x, dataImgSrc);
```

```
41                      totalpixel++;
42                  }
43              }
44          }
45          return segmentasiterang;
46      }
47  //Method untuk mengambil citra penyakit jika nilai otsu = 0
48      public Mat HasilSegmentasiGelap(Mat ImgSrc, Mat ImgEkstraksi) {
49          Mat segmentasigelap = new Mat(ImgSrc.height(), ImgSrc.width(),
50  ImgSrc.type(), new Scalar(0));
51          double totalpixel = 0;
52          for (int y = 0; y < ImgSrc.height(); y++) {
53              for (int x = 0; x < ImgSrc.width(); x++) {
54                  double[] dataImgSrc = ImgSrc.get(y, x);
55                  double[] dataEkstraksi = ImgEkstraksi.get(y, x);
56                  if (dataEkstraksi[0] == 0) {
57                      segmentasigelap.put(y, x, dataImgSrc);
58                      totalpixel++;
59                  }
60              }
61          }
62          return segmentasigelap;
63      }
```

Source Code 5.7 Pengambilan Citra Buah Berpenyakit

Penjelasan dari *Source Code* 5.7 adalah sebagai berikut :

1. Baris 3 - 9 merupakan proses inisialisasi variabel ekstraksi fitur.
2. Baris 11 - 29 merupakan proses binerisasi citra *grayscale* terhadap *threshold otsu.*
3. Baris 31 - 46 merupakan method untuk mengambil citra penyakit jika nilai otsu = 1 atau 255.
4. Baris 48 - 63 merupakan method untuk mengambil citra penyakit jika nilai otsu = 0.

5.1.8 Perhitungan Rata-Rata *RGB*

Perhitungan Rata-Rata *RGB* dilakukan untuk mengambil ekstraksi fitur pada citra berpenyakit. Perhitungan Rata – Rata *RGB* ditunjukkan pada *Source Code* 5.8.

```
1   public void getRataRataRGB(Mat ImgSrc) {
2   //Proses inisialisasi variable Rata-rata RGB
3           double SumR = 0.0;
4           double SumG = 0.0;
5           double SumB = 0.0;
6           double totalpixel = 0.0;
7   //Proses perhitungan nilai rata-rata RGB
8           if (ImgSrc.width() > 0) {
9               setImgWidth(ImgSrc.width());
10              setImgHeight(ImgSrc.height());
11              for (int i = 0; i < ImgSrc.height(); i++) {
12                  for (int j = 0; j < ImgSrc.width(); j++) {
13                      double[] data = ImgSrc.get(i, j);
14                      if (data[0] > 0 || data[1] > 0 || data[2] > 0) {
15                          totalpixel++;
16                          SumR += data[2];
17                          SumG += data[1];
18                          SumB += data[0];
19                      }
```

```
20                  }
21              }
22              setAvgR(SumR / totalpixel);
23              setAvgG(SumG / totalpixel);
24              setAvgB(SumB / totalpixel);
25          }
26      }
```

Source Code 5.8 Perhitungan Rata-Rata *RGB*

Penjelasan dari *Source Code* 5.8 adalah sebagai berikut :

1. Baris 3 - 6 merupakan proses inisialisasi variabel rata - rata *RGB*.
2. Baris 8 - 26 merupakan proses perhitungan nilai rata - rata *RGB*.

5.1.9 Normalisasi Data

Normalisasi Data dilakukan untuk mengubah nilai rata-rata *RGB* ke dalam range 0 sampai 1. Normalisasi Data ditunjukkan pada *Source Code* 5.9.

```
1   public HashMap<String, Double> NormalisasiData() {
2   //Inisialisasi variable normalisasi data
3           int JmlData = this.JmlData;
4           int JmlParameterInput = this.JmlParameterInput;
5           int JmlParameterOutput = this.JmlParameterOutput;
6           HashMap<String, Double> MatriksNorm = new HashMap<>();
7           double Max;
8           double Min;
9   //Proses normalisasi untuk parameter input
10          for (int i = 1; i <= JmlParameterInput; i++) {
11              Max = this.DataCitra.get("IN_1" + i);
12              Min = this.DataCitra.get("IN_1" + i);
13              for (int j = 1; j <= JmlData; j++) {
14                  if (this.DataCitra.get("IN_" + j + i) > Max) {
15                      Max = this.DataCitra.get("IN_" + j + i);
16                  }
17                  if (this.DataCitra.get("IN_" + j + i) < Min) {
18                      Min = this.DataCitra.get("IN_" + j + i);
19                  }
20              }
21              for (int j = 1; j <= JmlData; j++) {
22                  MatriksNorm.put("IN_"      +      j      +      i,
23  ((this.DataCitra.get("IN_" + j + i) - Min) / (Max - Min) * 0.8) +
24  0.1);
25              }
26          }
27  //Proses normalisasi untuk parameter  output
28          for (int i = 1; i <= JmlParameterOutput; i++) {
29              Max = this.DataCitra.get("OUT_1" + i);
30              Min = this.DataCitra.get("OUT_1" + i);
31              for (int j = 1; j <= JmlData; j++) {
32                  if (this.DataCitra.get("OUT_" + j + i) > Max) {
33                      Max = this.DataCitra.get("OUT_" + j + i);
34                  }
35                  if (this.DataCitra.get("OUT_" + j + i) < Min) {
36                      Min = this.DataCitra.get("OUT_" + j + i);
37                  }
38              }
39              for (int j = 1; j <= JmlData; j++) {
40
```

```
41                  MatriksNorm.put("OUT_"      +      j      +      i,
42  ((this.DataCitra.get("OUT_" + j + i) - Min) / (Max - Min) * 0.8) +
43  0.1);
44              }
45          }
46          return MatriksNorm;
47      }
```

Source Code 5.9 Normalisasi Data

Penjelasan dari *Source Code* 5.9 adalah sebagai berikut :

1. Baris 3 - 8 merupakan proses inisialisasi variabel normalisasi data.
2. Baris 10 - 26 merupakan proses normalisasi untuk parameter *input.*
3. Baris 28 - 47 merupakan proses normalisasi untuk parameter *output.*

5.1.10 Pelatihan Jaringan Syaraf Tiruan *Backpropagation*

Pelatihan jaringan syaraf tiruan *Backpropagation* merupakan proses melatih sistem terhadap data latih yang diberikan dan selanjutnya memperoleh bobot selanjutnya digunakan untuk pengujian. Pelatihan jaringan Syaraf Tiruan *Backpropagation* ditunjukkan pada *Source Code* 5.10.

```
1   public void CalculateBackpro(HashMap<String, Double> DataCitra) {
2   //Proses inisialisasi variable backpropagation
3           this.Z_net = new double[this.JmlHiddenLayer + 1];
4           this.Z = new double[this.JmlHiddenLayer + 1];
5           this.Y_net = new double[this.JmlParameterOutput + 1];
6           this.Y = new double[this.JmlParameterOutput + 1];
7           this.Faktor = new double[this.JmlParameterOutput + 1];
8           this.FaktorNet = new double[this.JmlHiddenLayer + 1];
9           this.FaktorKesalahan = new double[this.JmlHiddenLayer + 1];
10          this.Delta_Vji    =    new    double[this.JmlHiddenLayer    +
11  1][this.JmlParameterInput + 1];
12          this.Delta_Wkj   =   new   double[this.JmlParameterOutput   +
13  1][this.JmlHiddenLayer + 1];
14          int Itterasi = 0;
15          double MSE = 1;
16  //Proses Langkah 1 : Jika Kondisi belum terpenuhi , lakukan langkah
17  2 - 9
18          while (Itterasi < this.Maks_Itterasi) {
19              if (MSE < this.Min_error) {
20                  break;
21              }
22  //Proses Langkah 2 : Untuk setiap data pelatihan lakukan langkah 3 -
23  8
24              double Sigmaerror = 0.0;
25              for (int Tot = 1; Tot <= this.JmlData; Tot++) {
26
27  //Proses pemanggilan method FEEDFORWAD
28                  feedforward(Tot, DataCitra);
29  //Proses pemanggilan method Backpropagation
30                  backpropagation(Tot, DataCitra);
31  //Proses pemanggilan method Ubah bobot dan bias
32                  ubahbobotbias();
33  //Proses menghitung MSE
34                  for (int k = 1; k <= this.JmlParameterOutput; k++) {
35
```

```
36                      Sigmaerror += Math.pow(DataCitra.get("OUT_" +
37  Tot + k) - this.Y[k], 2.0);
38                  }
39              }
40              MSE = Sigmaerror / this.JmlData;
41              Itterasi++;
42          }
43      }
44
```

Source Code 5.10 Pelatihan Jaringan Syaraf Tiruan *Backpropagation*

Penjelasan dari *Source Code* 5.10 adalah sebagai berikut :

1. Baris 3 - 15 merupakan proses inisialisasi variabel *backpropagation.*
2. Baris 18 - 21 merupakan Proses Jika Kondisi belum terpenuhi , lakukan langkah 2 – 9.
3. Baris 28 merupakan proses pemanggilan method *feedforward* .
4. Baris 30 merupakan proses pemanggilan method *Backpropagation.*
5. Baris 32 merupakan proses pemanggilan method ubah bobot dan bias.
6. Baris 34 - 43 merupakan proses menghitung MSE.

5.1.11 Inisialisasi Bobot Awal

Inisialisasi bobot awal menggunakan algoritma *Nguyen-widrow* merupakan proses mendeklarasikan bobot awal yang akan digunakan untuk pelatihan jaringan syaraf backpropagation. Inisialisasi Bobot Awal ditunjukkan pada *Source Code* 5.11.

```
1   public void BobotNguyenWidrow() {
2   //Proses inisialisasi variabel bobot awal
3           int JmlParameterInput = this.JmlParameterInput;
4           int JmlParameterOutput = this.JmlParameterOutput;
5           int JmlHiddenLayer = this.JmlHiddenLayer;
6           double[][]   Bobot_Vji   =   new   double[JmlHiddenLayer   +
7   1][JmlParameterInput + 1];
8           double[][]  Bobot_Wkj  =  new  double[JmlParameterOutput  +
9   1][JmlHiddenLayer + 1];
10          double[] Vektor_v = new double[JmlHiddenLayer];
11  //Proses random nilai vji dan wkj dengan batas atas 0.5 dan batas
12  bawah -0.5
13          double BatasBawah = -0.5;
14          double BatasAtas = 0.5;
15          double range = BatasAtas - BatasBawah;
16          for (int j = 1; j <= JmlHiddenLayer; j++) {
17              for (int i = 0; i <= JmlParameterInput; i++) {
18                  double randomValue = (range * rand.nextDouble()) +
19  BatasBawah;
20                  Bobot_Vji[j][i] = randomValue;
21              }
22          }
23
24          for (int k = 1; k <= JmlParameterOutput; k++) {
25              for (int j = 0; j <= JmlHiddenLayer; j++) {
26                  double randomValue = (range * rand.nextDouble()) +
27  BatasBawah;
28                  Bobot_Wkj[k][j] = randomValue;
29              }
30          }
```

```
31  //Proses menentukan nilai beta
32          for (int j = 1; j <= JmlHiddenLayer; j++) {
33              double Totalv = 0.0;
34              for (int i = 1; i <= JmlParameterInput; i++) {
35                  Totalv += Math.pow(Bobot_Vji[j][i], 2);
36              }
37              Vektor_v[j - 1] = Math.sqrt(Totalv);
38          }
39          double Faktorskala = 0.7 * Math.pow(JmlHiddenLayer, 1.0 /
40  JmlParameterInput);
41  //Proses menentukan nilai vektor Vji
42          for (int j = 1; j <= JmlHiddenLayer; j++) {
43              for (int i = 1; i <= JmlParameterInput; i++) {
44                  Bobot_Vji[j][i] = (Faktorskala * Bobot_Vji[j][i]) /
45  Vektor_v[j - 1];
46              }
47          }
48  //Proses menentukan bobot secara random sesuai nilai beta yang
49  diperoleh
50          for (int j = 1; j <= JmlHiddenLayer; j++) {
51              double randomValue = ((Faktorskala + Faktorskala) *
52  rand.nextDouble()) - Faktorskala;
53              Bobot_Vji[j][0] = randomValue;
54          }
55
56          this.Bobot_Vji = Bobot_Vji;
57          this.Bobot_Wkj = Bobot_Wkj;
58      }
```

Source Code 5.11 Inisialisasi Bobot Awal

Penjelasan dari *Source Code* 5.11 adalah sebagai berikut :

1. Baris 3 - 10 merupakan proses inisialisasi variabel bobot awal.
2. Baris 13 - 30 merupakan proses random nilai vji dan wkj dengan batas atas 0.5 dan batas bawah -0.5.
3. Baris 32 - 40 merupakan proses menentukan nilai beta.
4. Baris 42 - 47 merupakan proses menentukan nilai vektor Vji.
5. Baris 50 - 58 merupakan proses menentukan bobot secara random sesuai nilai beta yang diperoleh.

5.1.12 Proses *FeedForward*

Proses *FeedForward* merupakan bagian dari proses JST Backpropagation yang berfungsi melakukan kalkulasi antara *Input* layer dan hidden layer. Proses *FeedForward* ditunjukkan pada *Source Code* 5.12.

```
1   public void feedforward(int Tot, HashMap<String, Double> DataCitra)
2   {
3   //Proses Tahap Perambatan Maju
4   //Proses menghitung semua keluaran di unit tersembunyi Znet dan Z
5           //System.out.println("** Langkah 4");
6           double Sigma_ZVji;
7           for (int j = 1; j <= this.JmlHiddenLayer; j++) {
8               Sigma_ZVji = 0.0;
9               String dat = "";
```

```
10              for (int i = 1; i <= this.JmlParameterInput; i++) {
11                  Sigma_ZVji  +=  (DataCitra.get("IN_"  +  Tot  +  i)  *
12  this.Bobot_Vji[j][i]);
13              }
14              this.Z_net[j] = this.Bobot_Vji[j][0] + Sigma_ZVji;
15              //System.out.println("Znet[" + j + "] = " + this.Z_net[j]
16  + "        ");
17
18              //Fungsi Aktivasi
19              this.Z[j] = sigmoid(this.Z_net[j]);
20              //System.out.println("Z[" + j + "]    = " + Z[j]);
21          }
22          //System.out.println();
23
24          ////////////////////////////////////////////////////
25  Akhir Langkah 4
26  //Proses Langkah 5 menentukan nilai Ynet dan Y
27          //System.out.println("** Langkah 5");
28          double Sigma_ZWkj;
29          for (int k = 1; k <= this.JmlParameterOutput; k++) {
30              Sigma_ZWkj = 0.0;
31              for (int j = 1; j <= this.JmlHiddenLayer; j++) {
32                  Sigma_ZWkj += (Z[j] * this.Bobot_Wkj[k][j]);
33              }
34              this.Y_net[k] = this.Bobot_Wkj[k][0] + Sigma_ZWkj;
35              //System.out.println("Ynet[" + k + "] = " + this.Y_net[k]
36  + "        ");
37
38              //Fungsi aktivasi
39              this.Y[k] = sigmoid(this.Y_net[k]);
40              //System.out.println("Y[" + k + "]    = " + this.Y[k] +
41  "        ");
42          }
43  ////////////////////Akhir Langkah 5
44      }
```

Source Code 5.12 Proses *FeedForward*

Penjelasan dari *Source Code* 5.12 adalah sebagai berikut :

1. Baris 6 - 21 merupakan proses menghitung semua keluaran di unit tersembunyi Znet dan Z.
2. Baris 27 - 44 merupakan proses Langkah 5 menentukan nilai Ynet dan Y.

5.1.13 Proses *backpropagation*

Proses *backpropagation* merupakan bagian dari proses JST Backpropagation yang berfungsi melakukan kalkulasi antara hidden layer dan *output* layer. Proses *Backpropagation* ditunjukkan pada *Source Code* 5.13.

```
1   public  void  backpropagation(int  Tot,  HashMap<String,  Double>
2   DataCitra) {
3   //Proses Tahap Perambatan Balik
4   //Proses menentukan nilai faktor k dan delta WKj
5   //System.out.println("** Langkah 6");
6           for (int k = 1; k <= this.JmlParameterOutput; k++) {
7               this.Faktor[k]  =  (DataCitra.get("OUT_"  +  Tot  +  k)  -
8   this.Y[k]) * (this.Y[k] * (1 - this.Y[k]));
```

```
9           }
10          for (int k = 1; k <= this.JmlParameterOutput; k++) {
11              this.Delta_Wkj[k][0] = this.LearningRate * Faktor[k] * 1;
12              //System.out.println("Delta_W["  +  k  +  "0]  =  "  +
13  this.Delta_Wkj[k][0]);
14              for (int j = 1; j <= this.JmlHiddenLayer; j++) {
15                  this.Delta_Wkj[k][j]    =    this.LearningRate    *
16  this.Faktor[k] * this.Z[j];
17                  //System.out.println("Delta_W[" + k + j + "] = " +
18  this.Delta_Wkj[k][j]);
19              }
20          }
21          //System.out.println();
22          ////////////////////////////////////Akhir Langkah 6
23
24  //Proses menentukan nilai faktor net, faktor kesalahan dan faktor Vji
25          //System.out.println("** Langkah 7");
26          for (int k = 1; k <= this.JmlParameterOutput; k++) {
27              for (int j = 1; j <= this.JmlHiddenLayer; j++) {
28                  this.FaktorNet[j]      =      this.Faktor[k]      *
29  this.Bobot_Wkj[k][j];
30                  //System.out.println("Faktor Net [" + j + "] = " +
31  this.FaktorNet[j]);
32              }
33          }
34          for (int j = 1; j <= this.JmlHiddenLayer; j++) {
35              this.FaktorKesalahan[j] = this.FaktorNet[j] * this.Z[j]
36  * (1 - this.Z[j]);
37              //System.out.println("Faktor Kesalahan[" + j + "] = " +
38  this.FaktorKesalahan[j]);
39          }
40          for (int j = 1; j <= this.JmlHiddenLayer; j++) {
41              this.Delta_Vji[j][0]     =     this.LearningRate      *
42  this.FaktorKesalahan[j];
43              //System.out.println("Delta  VJ0"  +  j  +  0  +  "  =  "  +
44  this.Delta_Vji[j][0]);
45              for (int i = 1; i <= this.JmlParameterInput; i++) {
46                  this.Delta_Vji[j][i]    =    this.LearningRate    *
47  this.FaktorKesalahan[j] * DataCitra.get("IN_" + Tot + i);
48                  //////System.out.println("Delta V0J" + 0 + j + " = "
49  + this.LearningRate + "*" + this.FaktorKesalahan.get(j - 1) + "*"+
50  DataCitra.get("IN_" + Tot + i) + " = " + this.Delta_Vji.get("V_" + 0
51  + j));
52                  //System.out.println("Delta VJI" + j + i + " = " +
53  this.Delta_Vji[j][i]);
54              }
55          }
56          ////System.out.println();
57          ////////////////////////////////////////////////////////
58  Akhir Langkah 7
59      }
```

Source Code 5. 13 Proses *backpropagation*

Penjelasan dari *Source Code* 5.13 adalah sebagai berikut :

1. Baris 6 - 23 merupakan proses menentukan nilai faktor k dan delta WKj.
2. Baris 25 - 60 merupakan proses menentukan nilai faktor net, faktor kesalahan dan faktor Vji.

5.1.14 Proses Ubah Bobot dan Bias

Proses ubah bobot dan bias merupakan proses untuk mendapatkan bobot baru yag akan digunakan pada iterasi selanjutnya. Proses Ubah Bobot dan Bias ditunjukkan pada *Source Code* 5.14.

```
1   public void ubahbobotbias() {
2   //Proses perhitungan bobot dan bias baru
3   //System.out.println("** Langkah 8");
4   //Proses perhitungan bobot wkj baru
5           for (int k = 1; k <= this.JmlParameterOutput; k++) {
6               for (int j = 0; j <= this.JmlHiddenLayer; j++) {
7                   this.Bobot_Wkj[k][j]    =    this.Bobot_Wkj[k][j]    +
8   this.Delta_Wkj[k][j];
9                   //System.out.println("WKJ" + k + j + " = " +
10  this.Bobot_Wkj[k][j]);
11              }
12          }
13  //Proses perhitungan bobot Vji baru
14          for (int j = 1; j <= this.JmlHiddenLayer; j++) {
15              for (int i = 0; i <= this.JmlParameterInput; i++) {
16                  this.Bobot_Vji[j][i]    =    this.Bobot_Vji[j][i]    +
17  this.Delta_Vji[j][i];
18                  //System.out.println("VJI" + j + i + " = " +
19  this.Bobot_Vji[j][i]);
20              }
21          }
22          //System.out.println();
23      }
```

Source Code 5.14 Proses Ubah Bobot dan Bias

Penjelasan dari *Source Code* 5.14 adalah sebagai berikut :

1. Baris 5 - 12 merupakan proses perhitungan bobot Wkj baru.
2. Baris 14 - 22 merupakan proses perhitungan bobot Vji baru.

5.1.15 Pengujian

Proses Pengujian merupakan proses untuk menguji data uji setelah mendapatkan bobot baru dari pelatihan JST *Backpropagation* dan akan mendapatkan nilai akurasi pengujian. Pengujian ditunjukkan pada *Source Code* 5.15.

```
1   //Proses inisialisasi normalisasi parameter data uji
2   HashMap<String,            Double>            DataCitra1              =
3   this.NormalisasiDataUji(DataUji, jmldatalatih);
4   //Proses inisialisasi variabel pengujian
5           this.Z_net = new double[this.JmlHiddenLayer + 1];
6           this.Z = new double[this.JmlHiddenLayer + 1];
7           this.Y_net = new double[this.JmlParameterOutput + 1];
8           this.Y = new double[this.JmlParameterOutput + 1];
9           this.Faktor = new double[this.JmlParameterOutput + 1];
10          this.FaktorNet = new double[this.JmlHiddenLayer + 1];
11          this.FaktorKesalahan = new double[this.JmlHiddenLayer + 1];
12          this.Delta_Vji    =    new    double[this.JmlHiddenLayer    +
13  1][this.JmlParameterInput + 1];
14          this.Delta_Wkj   =   new   double[this.JmlParameterOutput   +
15  1][this.JmlHiddenLayer + 1];
```

```
16
17  //Method perhitungan pengujian menggunakan langkah feedforward
18  backpropagation
19      public void CalculatePengujian() {
20  //Langkah 2 : Untuk setiap data pengujian lakukan langkah 3 - 8
21          int numAkurasi = 0;
22          for (int Tot = 1; Tot <= this.JmlData; Tot++) {
23
24              System.out.println("----------------------------------
25  ------------------------------Data ke - " + Tot);
26
27              //FEEDFORWAD
28              feedforward(Tot, DataCitra1);
29              if (round(DenormalisasiOutput(this.Y[1], jmldatalatih),
30  0) == DataUji.get("OUT_" + Tot + 1)) {
31                  numAkurasi++;
32              }
33              System.out.println("Denormalisasi   Target   =   "   +
34  DenormalisasiOutput(this.Y[1], jmldatalatih));
35          }
36          System.out.println("");
37          Double totalakurasi = ((double) numAkurasi / this.JmlData) *
38  100;
39  //Proses menghitung tingkat akurasi yang dihasilkan dari pengujian
40          System.out.println("AKURASI DATA UJI = " + numAkurasi + "/"
41  + this.JmlData + " * 100% = " + totalakurasi.toString() + "%");
      }
```

Source Code 5.15 Pengujian

Penjelasan dari *Source Code* 5.15 adalah sebagai berikut :

1. Baris 2 merupakan proses inisialisasi normalisasi parameter data uji.
2. Baris 5 - 15 merupakan proses inisialisasi variabel pengujian.
3. Baris 19 - 36 merupakan method perhitungan pengujian menggunakan langkah feedforward backpropagation.
4. Baris 37 - 41 merupakan proses menghitung tingkat akurasi yang dihasilkan dari pengujian.

5.1.16 Denormalisasi Data Uji

Denormalisasi data uji merupakan proses untuk mengembalikan nilai yang telah dinormalisasi ke dalam data asli.Denormalisasi Data Uji ditunjukkan pada *Source Code* 5.16.

```
1   public double DenormalisasiOutput(double Y, int jmldatalatih) {
2   //Proses inisialisasi variabel denormalisasi data uji
3           int JmlData = jmldatalatih;
4           int JmlParameterInput = this.JmlParameterInput;
5           int JmlParameterOutput = this.JmlParameterOutput;
6           HashMap<String, Double> MatriksNorm = new HashMap<>();
7           double Max;
8           double Min;
9           double output = 0;
10          /////PARAMETER OUTPUT
11
```

```
12  //Proses perhitungan denormalisasi output yang dihasilkan dari
13  pengujian
14          for (int i = 1; i <= JmlParameterOutput; i++) {
15              Max = this.DataCitra.get("OUT_1" + i);
16              Min = this.DataCitra.get("OUT_1" + i);
17              for (int j = 1; j <= JmlData; j++) {
18                  if (this.DataCitra.get("OUT_" + j + i) > Max) {
19                      Max = this.DataCitra.get("OUT_" + j + i);
20                  }
21                  if (this.DataCitra.get("OUT_" + j + i) < Min) {
22                      Min = this.DataCitra.get("OUT_" + j + i);
23                  }
24              }
25              //for (int j = 1; j <= JmlData; j++) {
26              output = (((Y - 0.1) / 0.8) * (Max - Min)) + Min;
27              //}
28          }
29          return output;
        }
```

Source Code 5.16 Denormalisasi Data Uji

Penjelasan dari *Source Code* 5.16 adalah sebagai berikut :

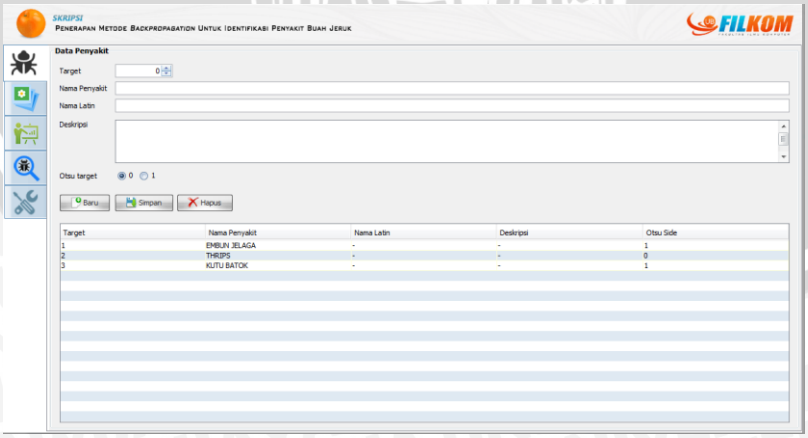
1. Baris 3 - 9 merupakan proses inisialisasi variabel denormalisasi data uji.
2. Baris 13 - 29 merupakan proses perhitungan denormalisasi *output* yang dihasilkan dari pengujian.

5.2 Implementasi Antarmuka

Pada sub bab implementasi antarmuka akan membahas penerapan desain dari antarmuka sesuai perancangan pada bab sebelumnya.

5.2.1 *Input* Data Penyakit

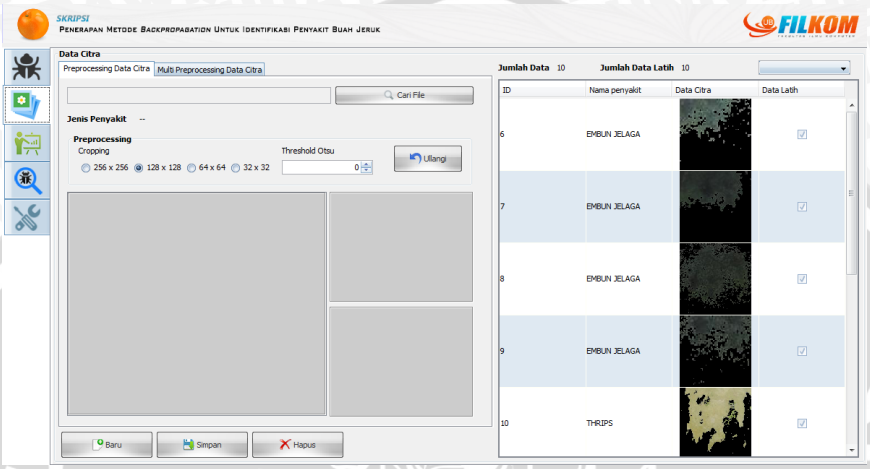
Antarmuka *Input* data penyakit digunakan untuk melakukan penambahan penyakit. Implementasi Antarmuka *Input* Data Penyakit ditunjukkan pada Gambar 5.1.



Gambar 5.1 Antarmuka *Input* Data Penyakit

5.2.2 *Input* Data Latih

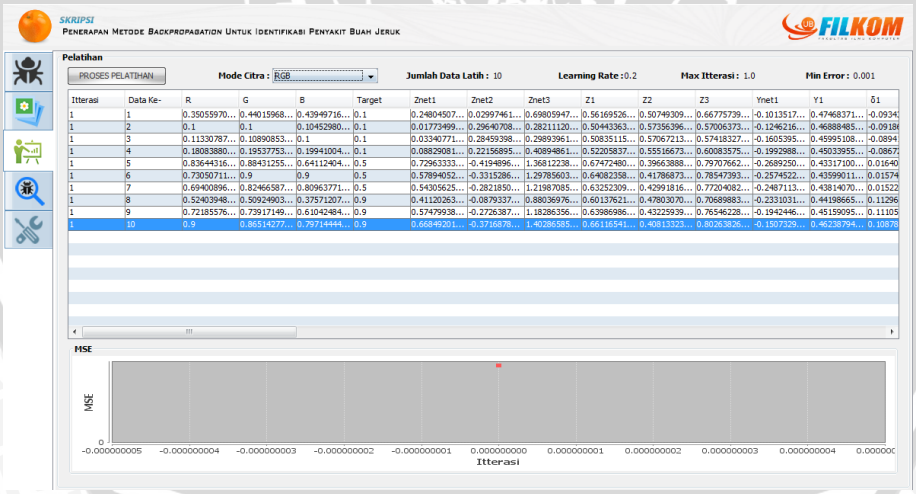
Input data latih digunakan untuk menambahkan citra yang akan digunakan dalam proses pelatihan. Pada antarmuka ini terdapat fasilitas untuk melakukan *preprocessing* citra serta ekstraksi fitur sebelum dilakukan penyimpanan. Implementasi *Input* Data Latih ditunjukkan pada Gambar 5.2.



Gambar 5.2 Antarmuka *Input* Data Latih

5.2.3 Pelatihan

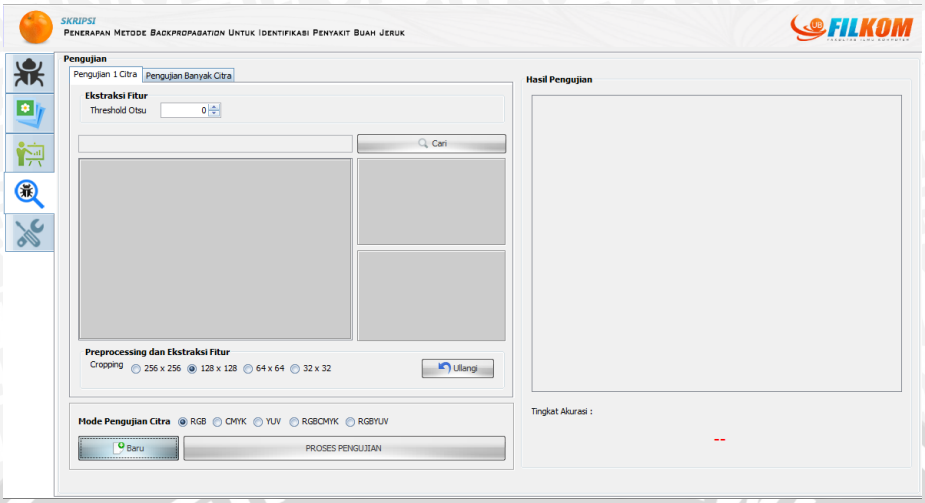
Antarmuka pelatihan digunakan untuk proses pelatihan dengan JST *Backpropagation*. Implementasi antarmukanya ditunjukkan pada Gambar 5.3.



Gambar 5.3 Antarmuka Pelatihan

5.2.4 Pengujian

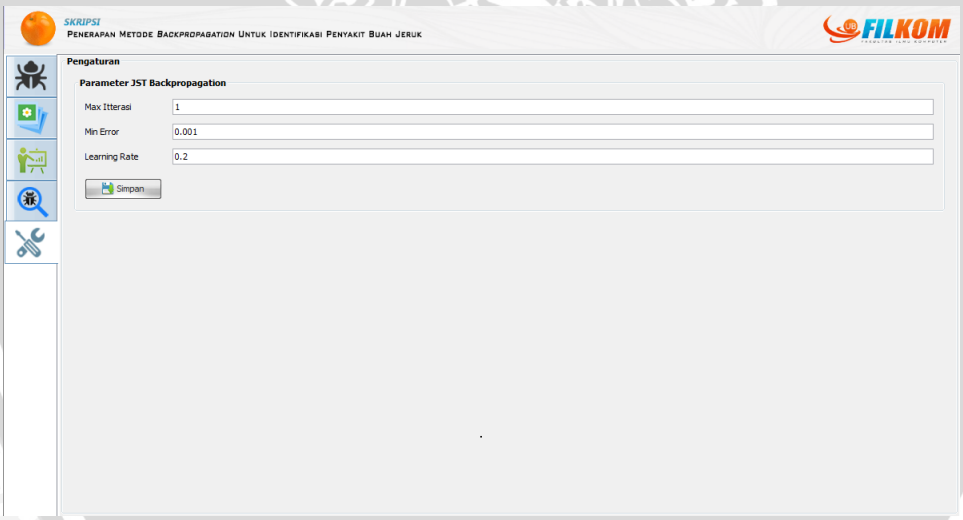
Antarmuka pengujian digunakan untuk proses pengujian dengan JST *Backpropagation*. Implementasi antarmukanya ditunjukkan pada Gambar 5.4.



Gambar 5.4 Antarmuka Pengujian

5.2.5 Pengaturan

Antarmuka pengaturan digunakan untuk *Input* pengaturan JST *Backpropagation* berupa *Max* Itterasi, *MinError* dan *Learning Rate*. Implementasi antarmukanya ditunjukkan pada Gambar 5.5.



Gambar 5.5 Antarmuka Pengaturan

BAB VI PENGUJIAN DAN ANALISIS

6.1 Pengujian

Pada bab ini akan dibahas tentang implementasi dari skenario pengujian yang telah dibahas pada perancangan bab 4. Terdapat 5 pengujian yang akan dilakukan yaitu Pengujian *Threshold Otsu*, Pengujian Jenis Ekstraksi Fitur, Pengujian Itterasi Maksimum, Pengujian *Learning Rate* dan pengujian persentase jumlah data uji dengan jumlah data latih. Pada setiap pengujian dilakukan 5 kali percobaan dengan bobot yang berbeda-beda. Berikut skenario pengujian yang dilakukan.

6.1.1. Pengujian Jenis Ekstraksi Fitur Terhadap Tingkat Akurasi

Pengujian jenis ekstraksi warna terhadap tingkat akurasi akan dilakukan dengan tiap Iterasi akan dilakukan percobaan sebanyak 5 kali. Nilai *learning rate* yang akan digunakan adalah 0,2, jumlah data latih masing-masing penyakit 30 dan data uji masing-masing penyakit 10, serta jumlah iterasi sebanyak 500 dengan bobot random yang berbeda menggunakan metode *Nguyen-widrow* pada tiap percobaan. Tabel 6.1.menampilkan hasil pengujian ektraksi fitur terhadap tingkat akurasi

Tabel 6.1 Pengujian Jenis Ekstraksi Fitur Terhadap Tingkat Akurasi

No	Ekstraksi Fitur	Percobaan ke-*i*					Rata-Rata Akurasi
		1	2	3	4	5	
1	*RGB*	96.67%	93.33%	93.33%	93.33%	93.33%	94%
2	*CMYK*	86.66%	86.66%	86.66%	86.66%	86.66%	86.66%
3	*YUV*	96.67%	86.66%	100%	100%	100%	96.67%
4	*RGBCMYK*	100%	100%	100%	100%	100%	100%
5	*RGBYUV*	100%	100%	100%	100%	100%	100%

Pada tabel 6.1 terlihat bahwa nilai rata-rata akurasi terkecil berada pada Ekstraksi fitur *CMYK* dengan nilai 86.66% sedangkan nIlai rata-rata akurasi terbesar berada ekstraksi fitur *RGBCMYK*, *RGBYUV* dengan nilai akurasi 100%.

6.1.2. Pengujian Threshold Otsu Terhadap Tingkat Akurasi

Pengujian *threshold otsu* terhadap tingkat akurasi akan dilakukan dengan menambahkan atau mengurangi nilai *threshold otsu* mulai -50 sampai +50, sedangkan tiap Iterasi akan dilakukan percobaan sebanyak 5 kali. Nilai *learning rate* yang akan digunakan adalah 0,2, jenis ekstraksi fitur menggunakan *RGBYUV*, jumlah data latih masing-masing penyakit 30 dan data uji masing-masing penyakit 10 serta jumlah iterasi sebanyak 500 dengan bobot random yang berbeda menggunakan metode

Nguyen-widrow pada tiap percobaan. Tabel 6.1. menunjukkan tingkat akurasi pengujian *threshold otsu* terhadap tingkat akurasi.

Tabel 6.2 Pengujian Threshold *Otsu* Terhadap Tingkat Akurasi

No	*Threshold*	Percobaan ke-*i*					Rata-Rata Akurasi
		1	2	3	4	5	
1	-50	83.33%	83.33%	83.33%	75%	83.33%	79.16%
2	-40	75%	75%	75%	75%	75%	75%
3	-30	91.66%	91.66%	91.66%	91.66%	91.66%	91.66%
4	-20	91.66%	91.66%	91.66%	83.33%	83.33%	88.32%
5	-10	100%	91.66%	100%	100%	83.33%	94.99%
6	0	91.66%	91.66%	91.66%	91.66%	91.66%	91.66%
7	+10	100%	100%	100%	100%	100%	100%
8	+20	100%	100%	100%	100%	100%	100%
9	+30	0%	0%	0%	0%	0%	0%
10	+40	0%	0%	0%	0%	0%	0%
11	+50	0%	0%	0%	0%	0%	0%

Pada tabel 6.1 terlihat bahwa nilai rata-rata akurasi terkecil berada pada *threshold* +30 sampai +50 dengan nilai 0% sedangkan nIlai rata-rata akurasi terbesar berada pada *threshold* +10 dan +20 dengan nilai akurasi 100%.

6.1.3. Pengujian Jumlah Persentase Data Latih dan Data Uji Terhadap Tingkat Akurasi

Pengujian jumlah persentase data latih dan data uji terhadap tingkat akurasi akan dilakukan dengan jumlah perbandingan persentase data latih dan data uji, jenis ekstraksi fitur yang digunakan *RGBYUV*, *threshold otsu* +20, nilai *learning rate* 0.2 dan jumlah iterasi 500 tiap Iterasi akan dilakukan percobaan sebanyak 5 kali. Bobot random yang digunakan berbeda menggunakan metode *Nguyen-widrow* pada tiap percobaan. Tabel pengujian ditampilkan pada Tabel 6.3.

Tabel 6.3 Pengujian Jumlah Data Latih Terhadap Tingkat Akurasi

No	% Data Latih dan Data Uji	Percobaan ke-*i*					Rata-Rata Akurasi
		1	2	3	4	5	
1	90 % dan 10 %	100%	100%	100%	100%	100%	100%
2	80 % dan 20 %	100%	100%	100%	100%	100%	100%
3	70 % dan 30 %	100%	96.29%	100%	100%	96.29%	98.52%
4	60 % dan 40 %	97.22%	100%	94.44%	97.22%	97.22%	97.22%
5	50 % dan 50 %	100%	100%	95.55%	100%	97.77%	98.67%
6	40 % dan 60 %	100%	98.14%	96.29%	100%	100%	98.89%
7	30 % dan 70 %	100%	98.41%	92.06%	93.65%	96.82%	96.19%
8	20 % dan 80 %	91.66%	93.05%	95.83%	86.11%	95.83%	92.50%
9	10 % dan 90%	75.30%	87.65%	83.95%	85.18%	65.43%	79.51%

Pada tabel 6.3 terlihat bahwa nilai rata-rata akurasi terkecil berada pada persentase data latih dan data uji 10% sampai 90% dengan nilai 79.51% sedangkan nIlai rata-rata akurasi terbesar berada pada persentase data latih dan data uji 90% dan 10% sampai 80% dan 20% dengan nilai akurasi 100%.

6.1.4. Pengujian Iterasi Maksimum Terhadap Tingkat Akurasi

Pengujian Iterasi maksimum terhadap tingkat akurasi akan dilakukan dengan mengacu pada skenario pengujian sebelumnya yaitu *threshold* +20, Jenis Ekstraksi Fitur *RGBCMYK*, nilai Iterasi mulai 100 sampai 1000, sedangkan tiap Iterasi akan dilakukan percobaan sebanyak 5 kali. Nilai *learning rate* yang akan digunakan adalah 0,2, jumlah data latih masing-masing penyakit 40 dan data uji masing-masing penyakit 4, dengan bobot random yang berbeda menggunakan metode *Nguyen-widrow* pada tiap percobaan. Tabel 6.4. menampilkan hasil pengujian iterasi terhadap tingkat akurasi.

Tabel 6.4 Pengujian Iterasi Maksimum Terhadap Tingkat Akurasi

No	Iterasi Maksimum	Percobaan ke-*i*					Rata-Rata
		1	2	3	4	5	
1	100	88.88%	88.88%	88.88%	88.88%	88.88%	88.88%
2	200	100%	88.88%	100%	100%	100%	97.78%
3	300	100%	100%	100%	100%	100%	100%

4	400	100%	100%	100%	100%	100%	100%
5	500	100%	100%	100%	100%	100%	100%
6	600	100%	100%	100%	100%	100%	100%
7	700	100%	100%	100%	100%	100%	100%
8	800	100%	100%	100%	100%	100%	100%
9	900	100%	100%	100%	100%	100%	100%
10	1000	100%	100%	100%	100%	100%	100%

Pada tabel 6.4 terlihat bahwa nilai rata-rata akurasi terkecil berada pada iterasi maksimum 100 dengan nilai akurasi sebesar 88.88%, kemudian naik pada iterasi maksimum 200 dengan nilai akurasi sebesar 97.78%, dan selanjutnya iterasi ke 300 sampai 1000 mengalami kenaikan akurasi yang konstan sebesar 100%.

6.1.5. Pengujian *Learning Rate* Terhadap Tingkat Akurasi

Pengujian *Learning Rate* terhadap tingkat akurasi akan dilakukan dengan mengacu pada skenario pengujian sebelumnya yaitu *threshold* +10, Jenis Ekstraksi Fitur *RGBYUV*, jumlah Iterasi 1000 , *learning rate* mulai 0,1 sampai 0,9 dengan jarak 0,1, jumlah data latih masing-masing penyakit 40 dan data uji masing-masing penyakit 4 ,tiap Iterasi akan dilakukan percobaan sebanyak 5 kali dengan bobot random yang berbeda menggunakan metode *Nguyen-widrow* pada tiap percobaan. Tabel 6.5. hasil pengujian *learning rate* terhadap tingkat akurasi.

Tabel 6.5 Pengujian Learning Rate Terhadap Tingkat Akurasi

No	*Learning Rate*	Percobaan ke-*i*					Rata-Rata Akurasi
		1	2	3	4	5	
1	0.01	66.66%	77.77%	66.66%	66.66%	66.66%	68.89%
2	0.03	88.88%	88.88%	88.88%	88.88%	66.66%	84.44%
3	0.05	100%	100%	100%	100%	100%	100%
4	0.07	100%	100%	100%	100%	100%	100%
5	0.09	100%	100%	100%	100%	100%	100%
6	0.1	100%	100%	100%	100%	100%	100%
7	0.3	100%	100%	100%	100%	100%	100%
8	0.5	100%	100%	100%	100%	100%	100%
9	0.7	100%	100%	100%	100%	100%	100%

10	0.9	100%	100%	100%	100%	100%	100%

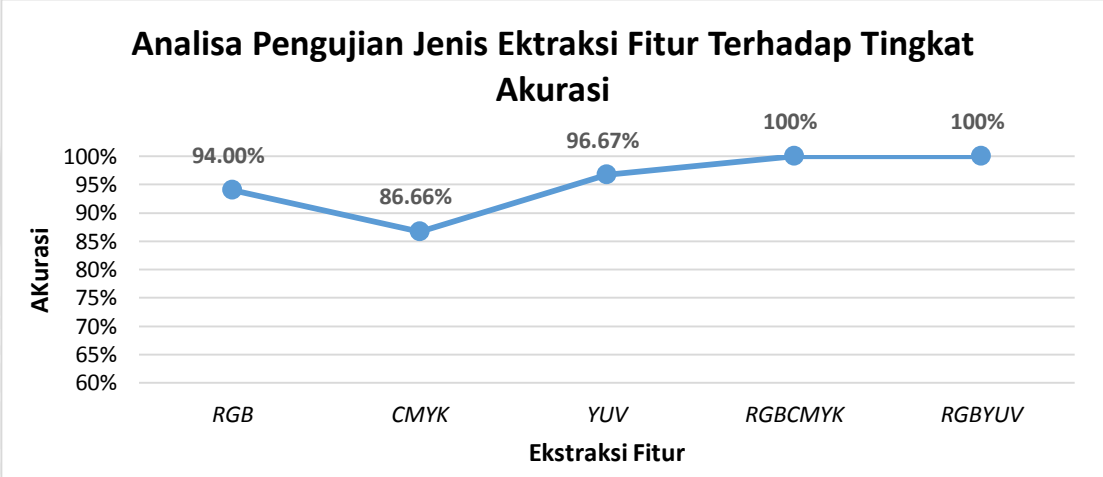
Pada Tabel 6.5. terlihat bahwa bahwa nilai rata-rata akurasi terkecil berada pada *learning rate* 0.01 dengan nilai akurasi sebesar 68.89% kemudian naik pada *learning rate* 0.03 dengan nilai akurasi 84.44% dan pada *learning rate* 0.05 sampai 0.7 mengalami kenaikan akurasi yang konstan sebesar 100%.

6.2 Analisa Hasil

Pada subbab ini akan dibahas tentang analisa hasil dari pengujian yang telah dilakukan pada penerapan metode *Backpropagation* untuk identifikasi penyakit pada citra buah tanaman jeruk.

6.2.1. Analisa Pengujian Jenis Ekstraksi Fitur Terhadap Tingkat Akurasi

Pada hasil pengujian pengaruh jenis ekstraksi fitur terhadap tingkat akurasi pada Tabel 6.1 terlihat bahwa nilai rata-rata akurasi terkecil berada pada ekstraksi fitur *CMYK* dengan nilai 58.33% sedangkan nIlai rata-rata akurasi terbesar berada ekstraksi fitur *RGBCMYK* dengan nilai akurasi 100%. Berdasarkan dari tabel 6.2 dapat dibuat grafik hubungan antara pengaruh jenis ekstraksi fitur terhadap tingkat akurasi yang terlihat pada Gambar 6.1.



Gambar 6.1 Analisa Pengujian Jenis Ekstraksi Fitur Terhadap Tingkat Akurasi

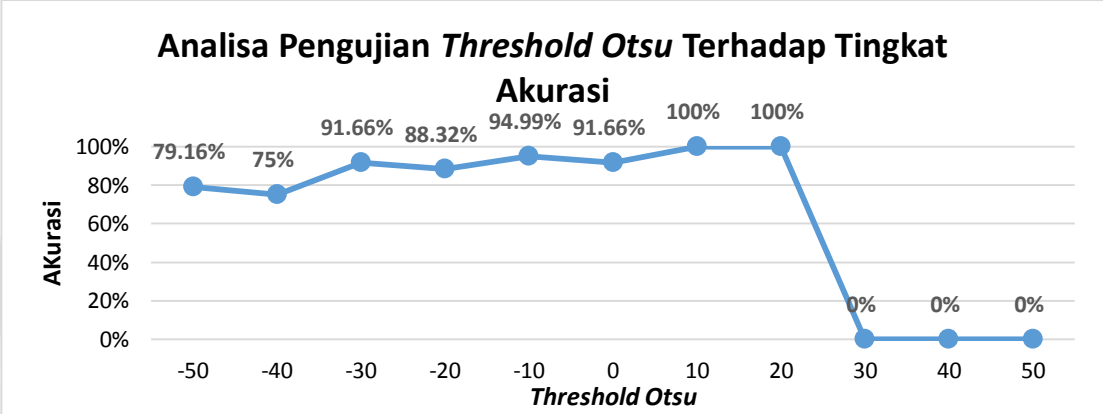
Pada Gambar 6.1 terlihat bahwa pada pengujian menggunakan ekstraksi fitur *RGB* nilai akurasinya sebesar 94%, kemudian menggunakan ekstraksi fitur *CMYK* turun menjadi 86.66%, selanjutnya naik akurasinya pada ekstraksi fitur *YUV* sebesar 96.67% dan naik lagi pada ekstraksi fitur *RGBCMYK* dan *RGBYUV* sebesar 100%. Dari Gambar 6.1 diketahui bahwa semakin banyak parameter yang digunakan dalam ekstraksi fitur semakin besar akurasinya seperti *RGBCMYK* dan *RGBYUV*.

Pada parameter ekstraksi fitur *YUV* juga memiliki akurasi yang tinggi tepat dibawah ekstraksi fitur gabungan seperti *RGBCMYK* dan *RGBYUV*, dilihat dari

karakteristiknya *YUV* dapat membedakan citra hitam putih dan citra berwarna sesuai yang telah dijelaskan pada subbab 2.5.5, begitu pula dengan data citra penyakit embun jelaga yang mempunyai dominan warna hitam sedangkan thrips mempunyai dominan warna putih dan kutu batok mempunyai dominan warna coklat sehingga ekstraksi fitur *YUV* cocok dalam studi kasus ini, untuk menunjang kinerja dari ektraksi fitur *YUV* perlu ditambahkan parameter ekstraksi fitur lagi, sehingga dipilih ekstraksi fitur *RGB* karena sesuai dengan hasil pengujian *RGB* memiliki akurasi yang cukup tinggi. Akhirnya pada penelitian kali ini dipilih gabungan ekstraksi fitur *RGBYUV* untuk pengujian selanjutnya.

6.2.2. Analisa Pengujian *Threshold Otsu* Terhadap Tingkat Akurasi

Pada hasil pengujian pengaruh *Threshold Otsu* terhadap tingkat akurasi seperti yang ditampilkan pada Tabel 6.2 terlihat bahwa nilai rata-rata akurasi terkecil berada pada *threshold* +30 sampai +50 dengan nilai 0% ,nIlai rata-rata akurasi terbesar berada pada *threshold* +10 dengan nilai akurasi 71.67%. Berdasarkan dari tabel 6.2 dapat dibuat grafik hubungan antara pengaruh *Threshold Otsu* terhadap tingkat akurasi yang terlihat pada Gambar 6.2.



Gambar 6.2 Analisa Pengujian *Threshold* Otsu Terhadap Tingkat Akurasi

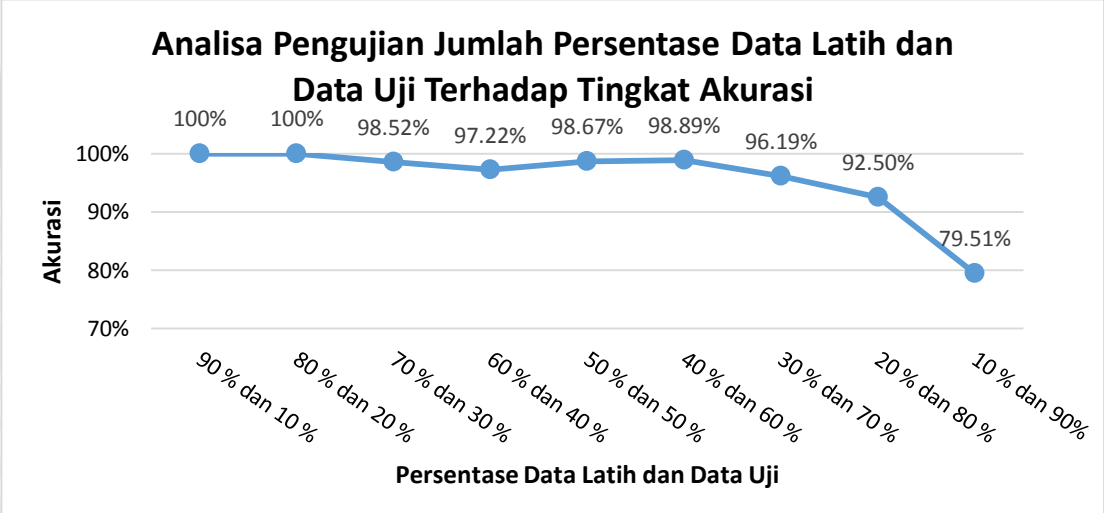
Pada Gambar 6.2 terlihat bahwa penambahan atau pengurangan *threshold otsu* memberikan pengaruh yang signifikan terhadap besaran nilai akurasi. Hal ini dapat dilihat pada *threshold otsu* -50 dengan akurasi 79.16%, kemudian turun pada *threshold otsu* -40 dengan akurasi 75% naik pada *threshold otsu* -30 dengan akurasi 91.66% turun pada *threshold otsu* -20 dengan akurasi 88.32%, naik lagi pada *threshold otsu* -10 sebesar 94.99% , turun pada *threshold otsu* 0 dengan akurasi 91.66%, naik lagi pada *threshold otsu* +10 dan *threshold otsu* +20 sebesar 100% dan pada *threshold otsu* +30 sampai +30 akurasi turun menjadi 0%. Sehingga terjadi penurunan yang signifikan pada penambahan *threshold otsu* +30 ke atas.

Penambahan maupun pengurangan jumlah *threshold otsu* sangat mempengaruhi tingkat akurasi seperti yang terlihat pada Gambar 6.2. Terlihat bahwa jika semakin menambah *threshold otsu* yang terjadi adalah hilangnya informasi citra

yang akan digunakan untuk ektraksi dan mempengaruhi proses algoritma *Backpropagation* dalam perhitungan bobot yang akan digunakan pada iterasi selanjutnya jika data citra memiliki nilai rata-rata ektraksi fitur sama dengan 0 maka secara otomatis bobot dalam algoritma *Backpropagation* juga menjadi 0 dan berakibat pada tingkat akurasi yang menjadi 0%. Faktor yang mempengaruhi tingkat akurasi pada penambahan maupun pengurangan *threshold otsu* adalah nilai dari *threshold otsu* dari citra itu sendiri berbeda-beda sehingga tingkat akurasi yang dihasilkan dari penambahan maupun pengurangan *threshold otsu* tidak mungkin terjadi secara konstan. Sehingga dipilih *threshold otsu* yang mempunyai penambahan nilai maksimum sebelum hilangnya informasi citra, sesuai dengan Gambar 6.2 untuk pengujian selanjutnya dipilih *threshold otsu* +20.

6.2.3. Analisa Pengujian Jumlah Persentase Data Latih dan Data Uji Terhadap Tingkat Akurasi

Pada hasil pengujian pengaruh jumlah persentase data latih dan data uji terhadap tingkat akurasi pada Tabel 6.3 terlihat bahwa nilai rata-rata akurasi terkecil berada pada persentase data latih dan data uji 10% sampai 90% dengan nilai 79.51% sedangkan nIlai rata-rata akurasi terbesar berada pada persentase data latih dan data uji 90% dan 10% sampai 80% dan 20% dengan nilai akurasi 100%. Berdasarkan dari tabel 6.5 dapat dibuat grafik hubungan antara pengaruh jumlah persentase data latih dan data uji terhadap tingkat akurasi yang terlihat pada Gambar 6.5.



Gambar 6.3 Analisa Pengujian Jumlah Persentase Data Latih dan Data Uji Terhadap Tingkat Akurasi

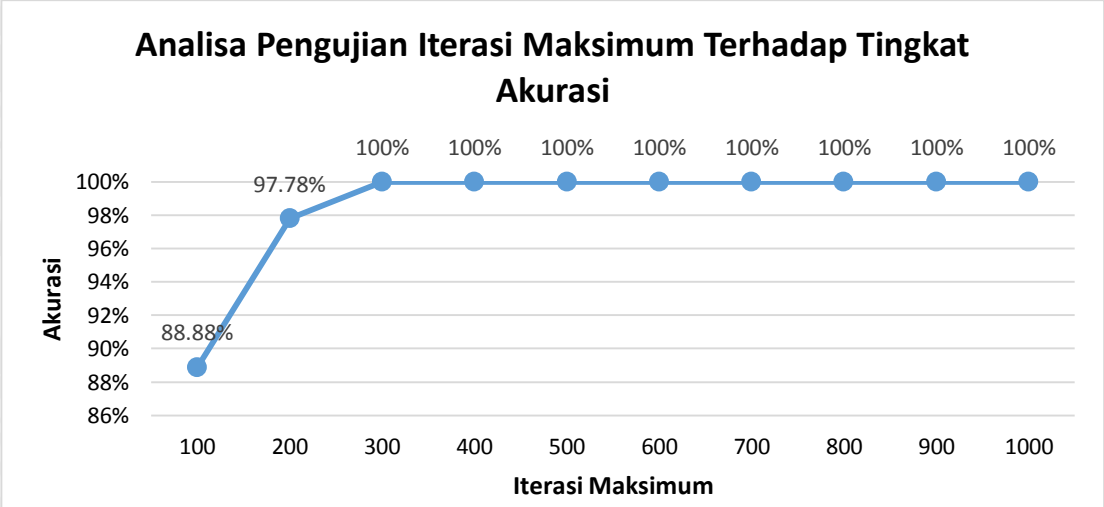
Pada Gambar 6.3 terlihat bahwa pemberian perbandingan data Latih dan data uji memberikan pengaruh pada nilai akurasi. Pada pemberian data latih 90% dan data uji 10 % sampai pemberian data latih 80% dan data uji 20% memberikan nilai akurasi 100%, nilai akurasi turun pada pemberian data latih 70% dan data uji 30% serta

pemberian data latih 60% dan data uji 40% sebesar 98.52% dan 97.22%, nilai akurasi naik pada pemberian data latih 50% dan data uji 50% serta data latih 40% dan data uji 60% sebesar 98.67% dan 98.89% selanjutnya turun secara konstan pada pemberian data latih 30% dan data uji 70% sampai data latih 10% dan data uji 90% dengan nilai akurasi terendah 79.51%.

Ada beberapa faktor yang mempengaruhi terhadap pengaruh jumlah persentase data latih dan data uji terhadap tingkat akurasi antara lain kualitas data latih dan data uji yang dipakai, semakin bagus kualitas data latih dan data uji yang dipakai maka dapat menunjang tingkat akurasi yang dihasilkan. Namun secara umum jumlah persetase data latih yang tinggi dan persentase data uji yang rendah maka akurasi juga semakin tinggi, sebaliknya jika jumlah persentase data latih yang rendah dan jumlah persentase data uji yang tinggi maka akurasi juga semakin rendah. Namun bisa saja terjadi persentase data latih yang rendah dan persentase data uji yang tinggi menunjukkan akurasi yang tinggi juga karena disebabkan kualitas data latih yang baik serta data uji yang baik. Sehingga untuk menjaga kestabilan proses pembelajaran serta proses pengujian dipilih persentase data latih tertinggi dan data uji terendah untuk pengujian selanjutnya menggunakan jumlah masing-masing data latih sebanyak 30 data yaitu 90% atau sejumlah 27 data untuk data latih dan 10% atau sejumlah 3 data untuk data uji.

6.2.4. Analisa Pengujian Iterasi Maksimum Terhadap Tingkat Akurasi

Pada hasil pengujian pengaruh iterasi maksimum terhadap tingkat akurasi pada Tabel 6.4 terlihat bahwa nilai rata-rata akurasi terkecil berada pada iterasi maksimum 100 dengan nilai akurasi sebesar 88.88%, kemudian naik pada iterasi maksimum 200 dengan nilai akurasi sebesar 97.78%, dan selanjutnya iterasi ke 300 sampai 1000 mengalami kenaikan akurasi yang konstan sebesar 100%. Berdasarkan dari tabel 6.3 dapat dibuat grafik hubungan antara pengaruh iterasi maksimum terhadap tingkat akurasi yang terlihat pada Gambar 6.3.

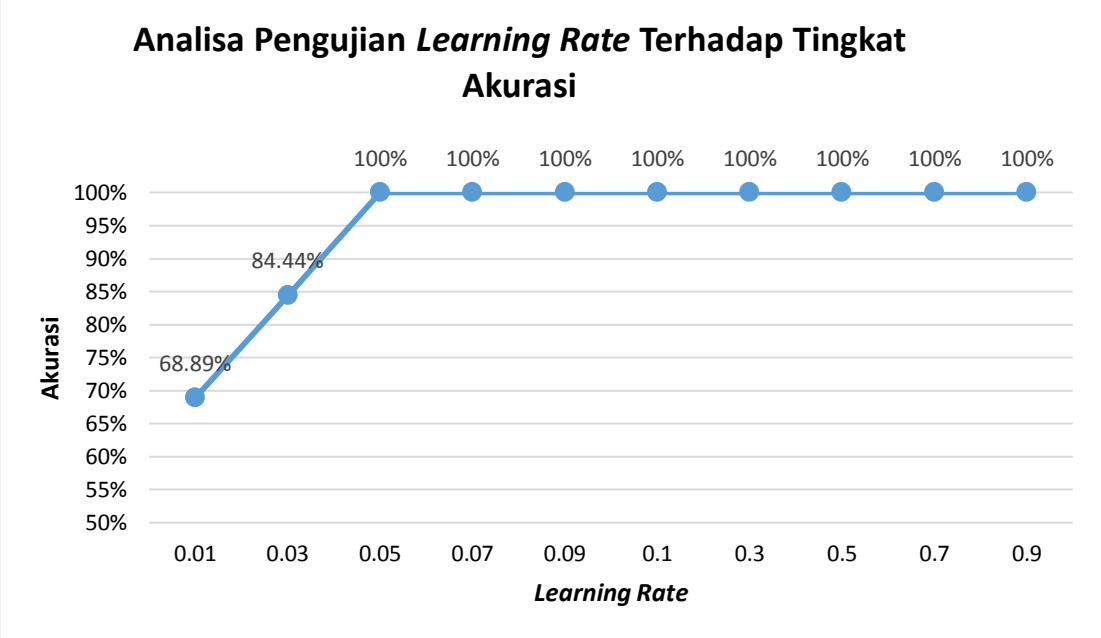


Gambar 6.4 Analisa Pengujian Iterasi Maksimum Terhadap Tingkat Akurasi

Pada Gambar 6.4 terlihat bahwa pemberian iterasi maksimum meberikan pengaruh pada tingkat akurasi. Pada pemberian iterasi maksimum 100 tingkat akurasi 88.88%, selanjutnya naik pada pemberian iterasi maksimum 200 dengan tingkat akurasi 97.78%, kemudian pada pemberian iterasi maksimum 300 sampai 1000 secara konstan memberikan tingkat akurasi yang signifikan yaitu 100%. Secara umum pemberian iterasi maksimum yang semakin besar nilai akurasi semakin besar pula namun proses pelatihan juga semakin lama, hal ini dikarenakan proses pelatihan jaringan *backpropagation* akan dilakukan berulang-ulang akan semakin baik karena memperbaiki kesalahan yang terjadi pada iterasi sebelumnya. Sehingga iterasi maksimum tertinggi dipilih untuk pengujian selanjutnya yaitu 1000.

6.2.5. Analisa Pengujian *Learning Rate* Terhadap Tingkat Akurasi

Pada hasil pengujian pengaruh *Learning Rate* terhadap tingkat akurasi pada Tabel 6.5 terlihat bahwa bahwa nilai rata-rata akurasi terkecil berada pada *learning rate* 0.01 dengan nilai akurasi sebesar 68.89% kemudian naik pada *learning rate* 0.03 dengan nilai akurasi 84.44% dan pada *learning rate* 0.05 sampai 0.7 mengalami kenaikan akurasi yang konstan sebesar 100%. Berdasarkan dari Tabel 6.5 dapat dibuat grafik hubungan antara pengaruh *Learning Rate* terhadap tingkat akurasi yang terlihat pada Gambar 6.5.



Gambar 6.5 Analisa Pengujian *Learning Rate* Terhadap Tingkat Akurasi

Pada Gambar 6.5 terlihat bahwa pada pemberian nilai *learning rate* mempengaruhi tingkat akurasi. Pada pemberian *learning rate* 0.01 memberikan nilai

akurasi sebesar 68.89 selanjutnya naik pada *learning rate* 0.03 memberikan nilai akurasi sebesar 84.44% kemudian pada *learning rate* 0.05 sampai 0.9 memberikan nilai akurasi yang konstan yaitu 100%. Semakin besar *learning rate* juga akan mempercepat proses pelatihan namun tingkat ketelitian yang dihasilkan dengan *learning rate* rendah yang berakibat pada ketidakstabilan perhitungan jaringan *backpropagation.* Akan tetapi pada hasil pengujian menunjukkan semakin tinggi nilai *Learning Rate* maka tingkat akurasi semakin tinggi hal tersebut disebabkan karena laju *Learning Rate* dimungkinkan berada pada langkah yang tepat saat proses pembelajaran sehingga akurasi yang dihasilkan menjadi tinggi. Selanjutnya untuk menjaga kestabilan dalam proses pembelajaran untuk *learning rate* dipilih nilai tengah dari skenario pengujian antara 0.05 sampai 0.9 untuk menjaga kestabilan perhitungan jaringan backpropagation yang mempunyai nilai akurasi 100% yaitu 0.1.

BAB VII PENUTUP

7.1. Kesimpulan

1. Untuk melakukan implementasi metode *Backpropagation* dalam indentifikasi citra pada buah tanaman jeruk diperlukan data latih yang akan digunakan sebagai sumber pelatihan pada metode *Backpropagation* yang selanjutnya diproses pada tahap pengujian dimana telah diberikan data uji citra penyakit pada buah tanaman jeruk. Tingkat akurasi yang dihasilkan dalam proses pengujian identifikasi penyakit pada citra buah tanaman jeruk menggunakan rumus perbandingan antara hasil dari proses pengujian menggunakan metode *Backpropagation* dengan hasil yang seharusnya diperoleh dari masing-masing data uji yang selanjutnya dijadikan persentase untuk memperoleh hasil akurasi.
2. Berdasarkan hasil pengujian didapatkan hasil akurasi tertinggi sebesar 100% dengan menggunakan parameter ekstraksi fitur *RGBYUV*, *threshold* otsu +20, jumlah perbandingan persentase data latih 90% dengan data uji 10%.
3. Sesuai dengan hasil pengujian jaringan syaraf *Backpropagation* menggunakan 6 *neuron input layer* , 5 *neuron hidden layer* dan 1 *neuron output layer*. Sedangkan parameter yang digunakan adalah iterasi maksimum 1000 dan *learning rate* 0.1

7.2. Saran

1. Untuk penelitian selanjutnya dapat dikembangkan jika diketahui hasil segmentasi otsu menunjukkan jumlah *pixel* dari *foreground* sangat sedikit atau hampir tidak ada secara otomatis harus menyesuaikan kondisi *pixel* yang ideal menggunakan *adaptive thresholding*.
2. Untuk penelitian selanjutnya perlu ditambahkan identifikasi tidak hanya mendeteksi satu penyakit saja namun lebih dari satu dalam satu citra buah tanaman jeruk.
3. Perlu ditambahkan luasan persentase penyakit pada parameter pelatihan yang digunakan untuk mendeteksi seberapa parah penyakit tersebut menyebar luas pada buah tanaman jeruk.
4. Untuk penelitian selanjutnya perlu diterapkan pada aplikasi *mobile*.
5. Perlu ditambahkan identifikasi penyakit pada buah jeruk yang lain untuk penelitian selanjutnya.
6. Untuk penelitian selanjutnya perlu diterapkan *auto cropping* pada bagian citra yang terkena penyakit.

DAFTAR PUSTAKA

Ahmad, U., 2005. *Pengolahan Citra Digital dan Teknik Pemrogramannya.* Yogyakarta: Graha Ilmu.

BALIJETSTRO, 2011. *Pengenalan Dan Pengendalian Hama Dan Penyakit Tanaman Jeruk.* III ed. Batu: Balai Penelitian Tanaman Jeruk dan Buah Subtropika, Pusat Penelitian dan Pengembangan Hortikultura,Badan Penelitian dan Pengembangan Pertanian,Kementrian Pertanian.

Basuki, A., n.d. *COLOR SPACE.* [Online] Available at: http://nana.lecturer.pens.ac.id/index_files/materi/Teori_Citra/TeoriCitra2014/Image%20Color%20Spaces.pptx [Accessed 20 10 2015].

Bayata, H. F., Hattatoglu, F. & Karsli, N., 2011. Modeling of monthly traffic accidents with the artificial neural network method. *International Journal of the Physical Sciences,* Volume 6, pp. 244 -254.

BPS, 2015. *Produksi Tanaman Buah-Buahan Dan Sayuran (Ton).* [Online] Available at: http://www.bps.go.id/ [Accessed 29 September 2015].

Fajri, N. I., 2011. *Prediksi Suhu dengan Menggunakan Algoritma-Algortima yang Terdapat pada Artificial Neural Network*. Bandung, Institut Teknologi Bandung.

Hadihardaja, I. K. & Sutikno, S., 2005. Pemodelan Curah Hujan-Limpasan Menggunakan Artifical Neural Network (ANN) dengan metode Backpropagation. *Jurnal Teknik Sipil,* Volume 12, pp. 249-258.

Heaton, J., 2008. *Introduction to Neural Networks for Java.* 2 ed. Chesterfield: Heaton Research, Inc.

Hermawan, A., 2006. *Jaringan Saraf Tiruan Teori dan Aplikasi.* Yogyakarta: Andi.

Ibrahiem, N. A., Hasan, M. M., Z Kan, R. & Misra, P. K., 2012. Understanding Color Models: A Review. *ARPN Journal of Science and Technology ,* Volume 2, pp. 265 -275 .

K., 1994. *Budidaya Tanaman Jeruk.* Yogyakarta: KANISIUS.

Kadir, A., 2013. *Dasar Pengolahan Citra Dengan Delphi.* Yogyakarta: ANDI.

Kusumadewi, S., 2004. *Membangun Jaringan Syaraf Tiruan Menggunakan MATLAB dan EXCEL LINK.* Yogyakarta: GRAHA ILMU.

Panchal, G., Ganatra, A., Kosta, Y. & Panchal, D., 2011. Behaviour Analysis of Multilayer Perceptrons with Multiple Hidden Neurons and Hidden Layers. *International Journal of Computer Theory and Engineering,* Volume 3, pp. 332 - 337.

Presman, R. S., 2013. *Rekayasa Perangkat Lunak Edisi 7.* Jogjakarta: Andi Publisher.

Puspitaningrum, D., 2006. *Pengantar Jaringan Syaraf Tiruan.* Yogyakarta: Andi.

Putra, D., 2004. Binerisasi Citra Tangan Menggunakan Metode Otsu. *Journal of Electrical Technology,* Volume 3, pp. 11-13.

Siang, J. J., 2009. *Jaringan Syaraf Tiruan dan Pemrogramannya Menggunakan MATLAB.* Yogyakarta: Andi.

Sutotjo, T. & Mulyanto, E., 2011. *KECERDASAN BUATAN.* Yogyakarta: Andi.

Verma, K., Verma, L. K. & Tripathi, P., 2014. Image Classification using Backpropagation Algorithm. *JOURNAL OF COMPUTER SCIENCE AND SOFTWARE APPLICATION,* Volume 1, pp. 7-15.

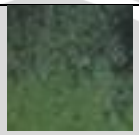
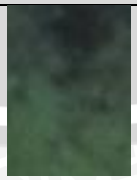
Wang, H., Li, G., Ma, Z. & Li, X., 2012. Image Recognition of Plant Diseases Based on Backpropagation Networks. *International Congress on Image and Signal Processing (CISP 2012) ,* pp. 894-900.

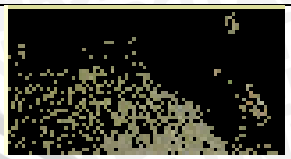
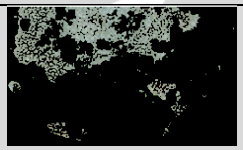
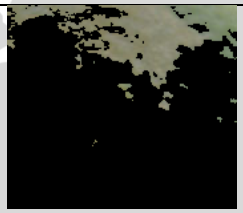
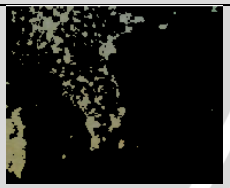
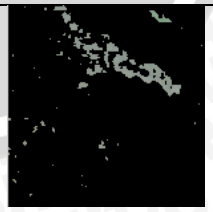
Wulan, T. D., Purwanti, E. & Yasin, M., 2013. Deteksi Kanker Paru-Paru Dari Citra Foto Rontgen Menggunakan Jaringan Saraf Tiruan Backpropagation. *Jurnal Universitas Airlangga,* Volume 1.

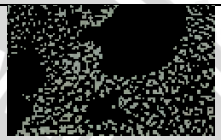
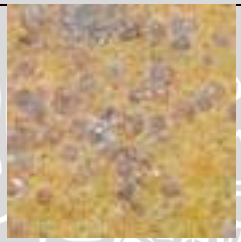
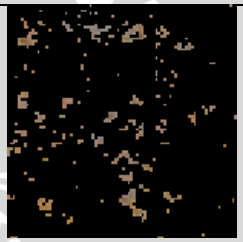
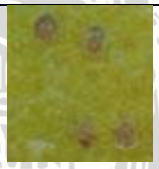
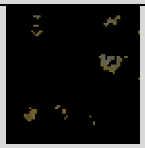
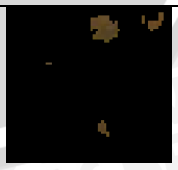
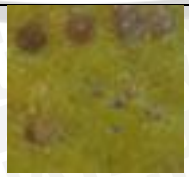
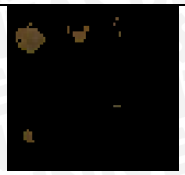
Yunita, T., 2012. Jaringan Saraf Tiruan Resilient Backpropagation Untuk Memprediksi Faktor Dominan Injury Severity Pada Kecelakaan Lalu Lintas.

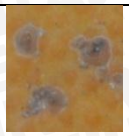
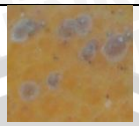
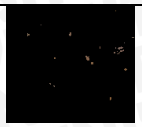
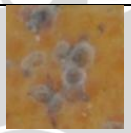
LAMPIRAN A DATA LATIH DAN DATA UJI

1. Data Latih

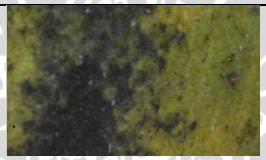
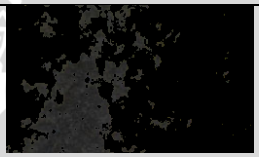
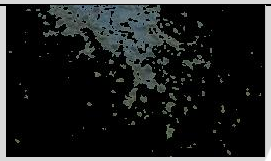
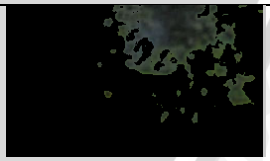
No	Nama Penyakit	Nama File	Citra Asli	Citra *Threshold Otsu* +20
1	Embun Jelaga	1-1.jpg		
2	Embun Jelaga	1-2.jpg		
3	Embun Jelaga	1-3.jpg		
4	Embun Jelaga	1-4.jpg		
5	Embun Jelaga	1-5.jpg		
6	Embun Jelaga	1-6.jpg		
7	Embun Jelaga	1-7.jpg		
8	Embun Jelaga	1-8.jpg		
9	Embun Jelaga	1-9.jpg		

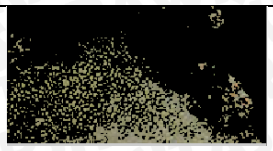
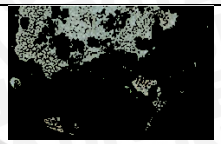
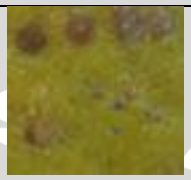
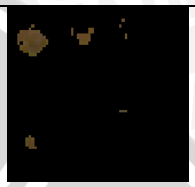
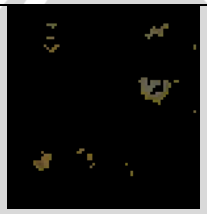
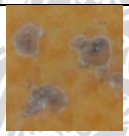
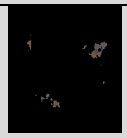
10	Embun Jelaga	1-10.jpg		
11	Thrips	2-1.jpg		
12	Thrips	2-2.jpg		
13	Thrips	2-3.jpg		
14	Thrips	2-4.jpg		
15	Thrips	2-5.jpg		
16	Thrips	2-6.jpg		
17	Thrips	2-7.jpg		

18	Thrips	2-8.jpg		
19	Thrips	2-9.jpg		
20	Thrips	2-10.jpg		
21	Kutu Batok	3-1.jpg		
22	Kutu Batok	3-2.jpg		
23	Kutu Batok	3-3.jpg		
24	Kutu Batok	3-4.jpg		
25	Kutu Batok	3-5.jpg		
26	Kutu Batok	3-6.jpg		

27	Kutu Batok	3-7.jpg		
28	Kutu Batok	3-8.jpg		
29	Kutu Batok	3-9.jpg		
30	Kutu Batok	3-10.jpg		

2. Data Uji

No	Nama Penyakit	Nama File	Citra Asli	Citra *Threshold Otsu* +20
1	Embun Jelaga	1-1.jpg		
2	Embun Jelaga	1-2.jpg		
3	Embun Jelaga	1-3.jpg		
4	Thrips	2-1.jpg		

5	Thrips	2-2.jpg		
6	Thrips	2-3.jpg		
7	Kutu Batok	3-1.jpg		
8	Kutu Batok	3-2.jpg		
9	Kutu Batok	3-3.jpg		

LAMPIRAN B LOG SKENARIO PENGUJIAN

1. Pengujian Ekstraksi Fitur

Iterasi = 500
Learning Rate = 0.2
Ekstraksi Fitur = *RGBYUV*

90 Data Latih 30 Embun Jelaga (Target 1), 30 Thrips (Target 2) , dan 30 Kutu Batok (Target 3)

Data ke-	*R*	*G*	*B*	*Y*	*U*	*V*	Target
1	45.69	54.92	49.45	51.58	-1.03	-5.13	1
2	40.13	45.99	45.04	44.17	0.45	-3.51	1
3	68.50	72.03	61.15	69.80	-4.23	-1.09	1
4	70.04	77.81	66.46	74.26	-3.80	-3.64	1
5	73.38	85.21	61.86	79.08	-8.44	-4.94	1
6	74.65	84.48	64.70	79.35	-7.18	-4.07	1
7	70.45	79.82	63.07	75.17	-5.92	-4.09	1
8	81.96	84.39	78.21	83.03	-2.34	-0.87	1
9	25.29	28.66	20.01	26.69	-3.27	-1.21	1
10	51.52	56.30	36.26	52.64	-8.04	-0.94	1
11	77.46	81.47	68.94	78.91	-4.87	-1.21	1
12	42.63	50.73	46.90	47.91	-0.48	-4.60	1
13	80.19	87.04	77.79	84.01	-3.03	-3.29	1
14	43.88	53.15	29.12	47.68	-9.11	-3.30	1
15	69.83	77.13	59.17	72.96	-6.76	-2.70	1
16	57.23	63.96	43.87	59.71	-7.77	-2.13	1
17	62.77	66.34	55.37	64.08	-4.26	-1.10	1
18	66.55	68.80	62.03	67.41	-2.62	-0.71	1
19	81.04	96.48	92.22	91.45	0.41	-9.07	1
20	41.45	39.55	25.13	38.51	-6.57	2.61	1
21	57.20	56.09	50.58	55.85	-2.56	1.23	1
22	29.42	32.67	29.52	31.37	-0.90	-1.68	1
23	49.68	60.87	51.56	56.51	-2.41	-5.95	1
24	34.25	36.31	33.76	35.43	-0.81	-1.01	1
25	44.91	54.47	50.92	51.25	-0.14	-5.52	1
26	26.30	28.60	25.30	27.56	-1.10	-1.08	1
27	44.57	54.30	38.16	49.59	-5.61	-4.37	1
28	36.72	42.62	32.85	39.78	-3.39	-2.65	1
29	34.73	38.98	33.76	37.15	-1.65	-2.09	1
30	37.00	42.07	43.25	40.72	1.26	-3.24	1
31	143.50	142.54	104.23	138.59	-16.85	4.42	2
32	143.61	154.80	141.49	150.07	-4.16	-5.55	2
33	148.00	153.75	123.06	148.66	-12.54	-0.47	2
34	141.01	152.24	141.81	147.82	-2.90	-5.86	2
35	139.34	146.60	131.97	142.89	-5.31	-3.00	2
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
88	106.82	91.18	82.42	94.95	-6.12	10.49	3
89	133.13	102.58	80.64	109.33	-14.06	20.98	3
90	115.90	91.31	76.65	97.09	-10.01	16.59	3

30 Data Latih 10 Embun Jelaga (Target 1), 10 Thrips (Target 2) , dan 10 Kutu Batok (Target 3)

Data Ke-	*R*	*G*	*B*	*Y*	*U*	*V*	Target
1	40.13	45.99	45.04	44.17	0.45	-3.51	1
2	44.91	54.47	50.92	51.25	-0.14	-5.52	1
3	26.30	28.60	25.30	27.56	-1.10	-1.08	1
4	36.72	42.62	32.85	39.78	-3.39	-2.65	1
5	34.73	38.98	33.76	37.15	-1.65	-2.09	1
6	37.00	42.07	43.25	40.72	1.26	-3.24	1

7	47.89	47.68	44.22	47.39	-1.54	0.47	1
8	82.74	98.17	94.27	93.19	0.57	-9.10	1
9	52.55	62.96	49.97	58.42	-4.13	-5.10	1
10	45.30	50.53	44.84	48.36	-1.71	-2.65	1
11	143.50	142.54	104.23	138.59	-16.85	4.42	2
12	154.10	164.78	147.80	159.79	-5.83	-4.87	2
13	149.24	163.26	148.20	157.49	-4.51	-7.12	2
14	133.99	136.94	110.65	133.18	-11.03	0.81	2
15	139.50	140.87	110.13	137.08	-13.20	2.23	2
16	140.48	151.42	139.96	146.97	-3.39	-5.58	2
17	142.50	147.17	114.07	142.13	-13.75	0.44	2
18	147.14	151.00	116.14	146.00	-14.63	1.11	2
19	162.60	160.60	118.23	156.52	-18.77	5.47	2
20	169.26	164.04	116.11	160.29	-21.66	8.01	2
21	108.46	91.04	80.23	95.12	-7.27	11.79	3
22	105.72	95.87	56.18	94.38	-18.75	10.02	3
23	95.18	73.29	37.82	75.88	-18.68	17.01	3
24	106.82	91.18	82.42	94.95	-6.12	10.49	3
25	133.13	102.58	80.64	109.33	-14.06	20.98	3
26	115.90	91.31	76.65	97.09	-10.01	16.59	3
27	102.15	90.83	62.97	91.13	-13.81	9.75	3
28	105.71	77.58	60.01	84.08	-11.80	19.05	3
29	102.17	90.36	50.27	89.41	-19.22	11.27	3
30	91.10	70.19	37.42	72.79	-17.36	16.14	3

Bobot *WKJ*

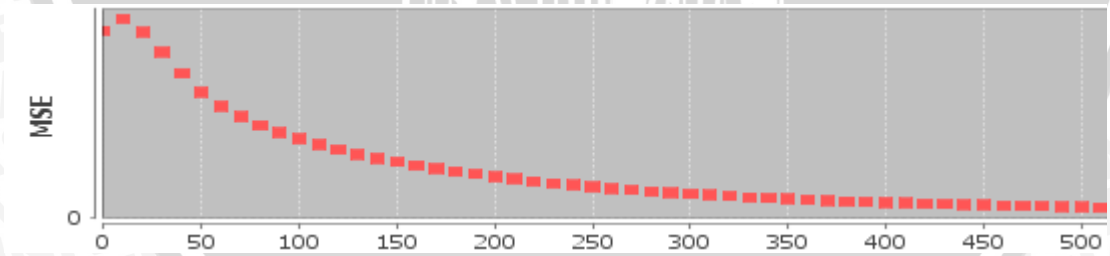
WKJ		K = 1
J	0	-1.48
	1	-1.47
	2	3.107
	3	-2.41
	4	5.94
	5	-4.12

Bobot *Vji*

Vji		j				
		1	2	3	4	5
i	0	-0.27	-1.16	1.29	-1.78	0.80
	1	-1.03	0.67	-0.61	4.35	0.23
	2	0.03	-1.44	0.68	1.07	1.85
	3	0.15	-0.73	-0.66	1.80	-0.03
	4	-0.38	-1.08	-0.09	2.36	1.63
	5	-0.57	0.40	-1.61	-3.92	-1.27
	6	-0.83	3.19	-0.97	3.46	-3.40

AKURASI DATA UJI = 30/30 * 100% = 100.0%

GRAFIK MSE



2. Pengujian *Threshold Otsu*

Iterasi = 500
Learning Rate = 0.2
Ekstraksi Fitur = *RGBYUV*
Threshold Otsu = +20

90 Data Latih 30 Embun Jelaga (Target 1), 30 Thrips (Target 2) , dan 30 Kutu Batok (Target 3)

Data ke-	R	G	B	Y	U	V	Target
1	45.69	54.92	49.45	51.58	-1.03	-5.13	1
2	40.13	45.99	45.04	44.17	0.45	-3.51	1
3	68.50	72.03	61.15	69.80	-4.23	-1.09	1
4	70.04	77.81	66.46	74.26	-3.80	-3.64	1
5	73.38	85.21	61.86	79.08	-8.44	-4.94	1
6	74.65	84.48	64.70	79.35	-7.18	-4.07	1
7	70.45	79.82	63.07	75.17	-5.92	-4.09	1
8	81.96	84.39	78.21	83.03	-2.34	-0.87	1
9	25.29	28.66	20.01	26.69	-3.27	-1.21	1
10	51.52	56.30	36.26	52.64	-8.04	-0.94	1
11	77.46	81.47	68.94	78.91	-4.87	-1.21	1
12	42.63	50.73	46.90	47.91	-0.48	-4.60	1
13	80.19	87.04	77.79	84.01	-3.03	-3.29	1
14	43.88	53.15	29.12	47.68	-9.11	-3.30	1
15	69.83	77.13	59.17	72.96	-6.76	-2.70	1
16	57.23	63.96	43.87	59.71	-7.77	-2.13	1
17	62.77	66.34	55.37	64.08	-4.26	-1.10	1
18	66.55	68.80	62.03	67.41	-2.62	-0.71	1
19	81.04	96.48	92.22	91.45	0.41	-9.07	1
20	41.45	39.55	25.13	38.51	-6.57	2.61	1
21	57.20	56.09	50.58	55.85	-2.56	1.23	1
22	29.42	32.67	29.52	31.37	-0.90	-1.68	1
23	49.68	60.87	51.56	56.51	-2.41	-5.95	1
24	34.25	36.31	33.76	35.43	-0.81	-1.01	1
25	44.91	54.47	50.92	51.25	-0.14	-5.52	1
26	26.30	28.60	25.30	27.56	-1.10	-1.08	1
27	44.57	54.30	38.16	49.59	-5.61	-4.37	1
28	36.72	42.62	32.85	39.78	-3.39	-2.65	1
29	34.73	38.98	33.76	37.15	-1.65	-2.09	1
30	37.00	42.07	43.25	40.72	1.26	-3.24	1
31	143.50	142.54	104.23	138.59	-16.85	4.42	2
32	143.61	154.80	141.49	150.07	-4.16	-5.55	2
33	148.00	153.75	123.06	148.66	-12.54	-0.47	2
34	141.01	152.24	141.81	147.82	-2.90	-5.86	2
35	139.34	146.60	131.97	142.89	-5.31	-3.00	2
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
88	106.82	91.18	82.42	94.95	-6.12	10.49	3
89	133.13	102.58	80.64	109.33	-14.06	20.98	3
90	115.90	91.31	76.65	97.09	-10.01	16.59	3

30 Data Uji 10 Embun Jelaga (Target 1), 10 Thrips (Target 2) , dan 10 Kutu Batok (Target 3)

Data Ke-	R	G	B	Y	U	V	Target
1	40.13	45.99	45.04	44.17	0.45	-3.51	1
2	44.91	54.47	50.92	51.25	-0.14	-5.52	1
3	26.30	28.60	25.30	27.56	-1.10	-1.08	1
4	36.72	42.62	32.85	39.78	-3.39	-2.65	1
5	34.73	38.98	33.76	37.15	-1.65	-2.09	1
6	37.00	42.07	43.25	40.72	1.26	-3.24	1
7	47.89	47.68	44.22	47.39	-1.54	0.47	1
8	82.74	98.17	94.27	93.19	0.57	-9.10	1
9	52.55	62.96	49.97	58.42	-4.13	-5.10	1
10	45.30	50.53	44.84	48.36	-1.71	-2.65	1
11	143.50	142.54	104.23	138.59	-16.85	4.42	2
12	154.10	164.78	147.80	159.79	-5.83	-4.87	2
13	149.24	163.26	148.20	157.49	-4.51	-7.12	2
14	133.99	136.94	110.65	133.18	-11.03	0.81	2
15	139.50	140.87	110.13	137.08	-13.20	2.23	2
16	140.48	151.42	139.96	146.97	-3.39	-5.58	2
17	142.50	147.17	114.07	142.13	-13.75	0.44	2
18	147.14	151.00	116.14	146.00	-14.63	1.11	2

19	162.60	160.60	118.23	156.52	-18.77	5.47	2
20	169.26	164.04	116.11	160.29	-21.66	8.01	2
21	108.46	91.04	80.23	95.12	-7.27	11.79	3
22	105.72	95.87	56.18	94.38	-18.75	10.02	3
23	95.18	73.29	37.82	75.88	-18.68	17.01	3
24	106.82	91.18	82.42	94.95	-6.12	10.49	3
25	133.13	102.58	80.64	109.33	-14.06	20.98	3
26	115.90	91.31	76.65	97.09	-10.01	16.59	3
27	102.15	90.83	62.97	91.13	-13.81	9.75	3
28	105.71	77.58	60.01	84.08	-11.80	19.05	3
29	102.17	90.36	50.27	89.41	-19.22	11.27	3
30	91.10	70.19	37.42	72.79	-17.36	16.14	3

Bobot *WKJ*

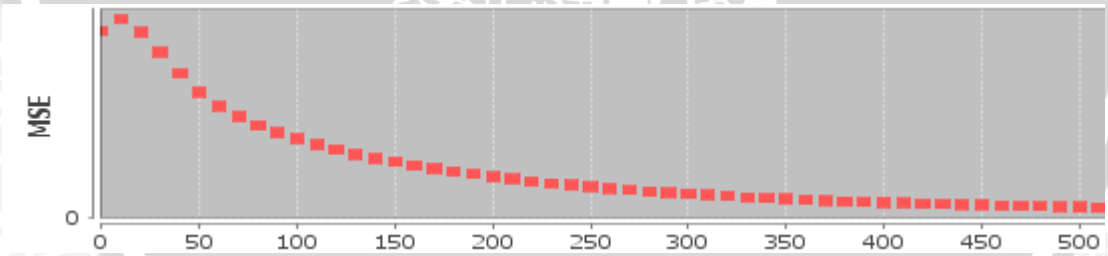
WKJ		K = 1
J	0	0.20
	1	-2.15
	2	-3.62
	3	-3.05
	4	3.58
	5	2.70

Bobot *Vji*

Vji		j				
		1	2	3	4	5
i	0	0.45	2.22	1.13	-0.66	-1.39
	1	-1.82	-2.47	-2.007	0.51	-0.095
	2	-1.13	-1.72	-1.175	-1.15	-1.04
	3	-1.08	-1.66	-1.161	-2.18	-1.23
	4	-1.56	0.88	-1.89	-0.74	-0.46
	5	0.80	0.88	0.67	-1.49	-0.63
	6	-0.88	-1.68	-1.48	4.28	3.086

AKURASI DATA UJI = 30/30 * 100% = 100.0%

GRAFIK MSE



3. Pengujian Perbandingan Data Latih dan Data Uji

Mode Ekstraksi = 4

Maks Iterasi = 500

Learning Rate = 0.2

Persentase Data Latih = 90.0%

Data Latih Target 1 = 30

90.0% Data Latih Target 1 = 27

10.0% Data Uji Target 1 = 3

Data Latih Target 2 = 30

90.0% Data Latih Target 2 = 27

10.0% Data Uji Target 2 = 3

Data Latih Target 3 = 30

90.0% Data Latih Target 3 = 27

10.0% Data Uji Target 3 = 3

Data ke-	R	G	B	Y	U	V	Target
1	45.69	54.92	49.45	51.58	-1.03	-5.13	1
2	40.13	45.99	45.04	44.17	0.45	-3.51	1
3	68.50	72.03	61.15	69.80	-4.23	-1.09	1
4	70.04	77.81	66.46	74.26	-3.80	-3.64	1
5	73.38	85.21	61.86	79.08	-8.44	-4.94	1
6	74.65	84.48	64.70	79.35	-7.18	-4.07	1
7	70.45	79.82	63.07	75.17	-5.92	-4.09	1
8	81.96	84.39	78.21	83.03	-2.34	-0.87	1
9	25.29	28.66	20.01	26.69	-3.27	-1.21	1
10	51.52	56.30	36.26	52.64	-8.04	-0.94	1
11	77.46	81.47	68.94	78.91	-4.87	-1.21	1
12	42.63	50.73	46.90	47.91	-0.48	-4.60	1
13	80.19	87.04	77.79	84.01	-3.03	-3.29	1
14	43.88	53.15	29.12	47.68	-9.11	-3.30	1
15	69.83	77.13	59.17	72.96	-6.76	-2.70	1
16	57.23	63.96	43.87	59.71	-7.77	-2.13	1
17	62.77	66.34	55.37	64.08	-4.26	-1.10	1
18	66.55	68.80	62.03	67.41	-2.62	-0.71	1
19	81.04	96.48	92.22	91.45	0.41	-9.07	1
20	41.45	39.55	25.13	38.51	-6.57	2.61	1
21	57.20	56.09	50.58	55.85	-2.56	1.23	1
22	29.42	32.67	29.52	31.37	-0.90	-1.68	1
23	49.68	60.87	51.56	56.51	-2.41	-5.95	1
24	34.25	36.31	33.76	35.43	-0.81	-1.01	1
25	44.91	54.47	50.92	51.25	-0.14	-5.52	1
26	26.30	28.60	25.30	27.56	-1.10	-1.08	1
27	44.57	54.30	38.16	49.59	-5.61	-4.37	1
28	143.50	142.54	104.23	138.59	-16.85	4.42	2
29	143.61	154.80	141.49	150.07	-4.16	-5.55	2
30	148.00	153.75	123.06	148.66	-12.54	-0.47	2
31	141.01	152.24	141.81	147.82	-2.90	-5.86	2
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
79	105.72	95.87	56.18	94.38	-18.75	10.02	3
80	95.63	73.37	37.46	76.02	-18.93	17.28	3
81	95.18	73.29	37.82	75.88	-18.68	17.01	3

9 Data Latih 3 Embun Jelaga (Target 1), 3 Thrips (Target 2) , dan 3 Kutu Batok (Target 3)

Data ke-	R	G	B	Y	U	V	Target
1	36.72	42.62	32.85	39.78	-3.39	-2.65	1
2	34.73	38.98	33.76	37.15	-1.65	-2.09	1
3	37.00	42.07	43.25	40.72	1.26	-3.24	1
4	140.48	151.42	139.96	146.97	-3.39	-5.58	2
5	144.53	154.55	137.95	149.79	-5.76	-4.50	2
6	136.92	148.47	135.02	143.61	-4.17	-5.76	2
7	106.82	91.18	82.42	94.95	-6.12	10.49	3
8	133.13	102.58	80.64	109.33	-14.06	20.98	3
9	115.90	91.31	76.65	97.09	-10.01	16.59	3

Bobot *WKJ*

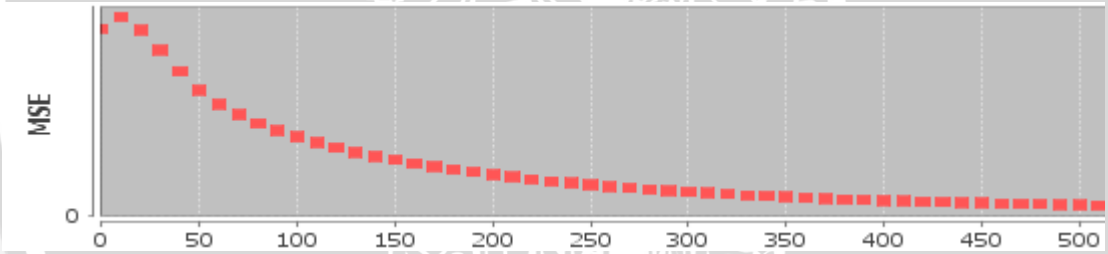
WKJ		K = 1
J	0	-1.32
	1	1.46
	2	-4.75
	3	5.04
	4	-4.001
	5	0.28

Bobot *Vji*

Vji		j				
		1	2	3	4	5
i	0	-0.46	1.40	-0.88	1.33	-0.72
	1	0.47	-3.21	0.45	-1.96	-0.13
	2	-0.46	-0.03	-2.92	-0.878	-0.15
	3	0.71	-1.77	-1.24	-1.546	0.34
	4	0.405	-1.71	-1.13	-1.53	-0.58
	5	0.55	1.94	1.17	1.32	-0.36
	6	1.52	-3.08	4.74	-3.11	0.83

AKURASI DATA UJI = 9/9 * 100% = 100.0%

GRAFIK MSE



4. Pengujian Iterasi Maksimum

Maks Iterasi = 1000

Learning Rate = 0.2

Persentase Data Latih = 90.0%

Data Latih Target 1 = 30

90.0% Data Latih Target 1 = 27

10.0% Data Uji Target 1 = 3

Data Latih Target 2 = 30

90.0% Data Latih Target 2 = 27

10.0% Data Uji Target 2 = 3

Data Latih Target 3 = 30

90.0% Data Latih Target 3 = 27

10.0% Data Uji Target 3 = 3

Data ke-	R	G	B	Y	U	V	Target
1	45.69	54.92	49.45	51.58	-1.03	-5.13	1

2	40.13	45.99	45.04	44.17	0.45	-3.51	1
3	68.50	72.03	61.15	69.80	-4.23	-1.09	1
4	70.04	77.81	66.46	74.26	-3.80	-3.64	1
5	73.38	85.21	61.86	79.08	-8.44	-4.94	1
6	74.65	84.48	64.70	79.35	-7.18	-4.07	1
7	70.45	79.82	63.07	75.17	-5.92	-4.09	1
8	81.96	84.39	78.21	83.03	-2.34	-0.87	1
9	25.29	28.66	20.01	26.69	-3.27	-1.21	1
10	51.52	56.30	36.26	52.64	-8.04	-0.94	1
11	77.46	81.47	68.94	78.91	-4.87	-1.21	1
12	42.63	50.73	46.90	47.91	-0.48	-4.60	1
13	80.19	87.04	77.79	84.01	-3.03	-3.29	1
14	43.88	53.15	29.12	47.68	-9.11	-3.30	1
15	69.83	77.13	59.17	72.96	-6.76	-2.70	1
16	57.23	63.96	43.87	59.71	-7.77	-2.13	1
17	62.77	66.34	55.37	64.08	-4.26	-1.10	1
18	66.55	68.80	62.03	67.41	-2.62	-0.71	1
19	81.04	96.48	92.22	91.45	0.41	-9.07	1
20	41.45	39.55	25.13	38.51	-6.57	2.61	1
21	57.20	56.09	50.58	55.85	-2.56	1.23	1
22	29.42	32.67	29.52	31.37	-0.90	-1.68	1
23	49.68	60.87	51.56	56.51	-2.41	-5.95	1
24	34.25	36.31	33.76	35.43	-0.81	-1.01	1
25	44.91	54.47	50.92	51.25	-0.14	-5.52	1
26	26.30	28.60	25.30	27.56	-1.10	-1.08	1
27	44.57	54.30	38.16	49.59	-5.61	-4.37	1
28	143.50	142.54	104.23	138.59	-16.85	4.42	2
29	143.61	154.80	141.49	150.07	-4.16	-5.55	2
30	148.00	153.75	123.06	148.66	-12.54	-0.47	2
31	141.01	152.24	141.81	147.82	-2.90	-5.86	2
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
80	95.63	73.37	37.46	76.02	-18.93	17.28	3
81	95.18	73.29	37.82	75.88	-18.68	17.01	3

9 Data Latih 3 Embun Jelaga (Target 1), 3 Thrips (Target 2), dan 3 Kutu Batok (Target 3)

Data ke-	R	G	B	Y	U	V	Target
1	36.72	42.62	32.85	39.78	-3.39	-2.65	1
2	34.73	38.98	33.76	37.15	-1.65	-2.09	1
3	37.00	42.07	43.25	40.72	1.26	-3.24	1
4	140.48	151.42	139.96	146.97	-3.39	-5.58	2
5	144.53	154.55	137.95	149.79	-5.76	-4.50	2
6	136.92	148.47	135.02	143.61	-4.17	-5.76	2
7	106.82	91.18	82.42	94.95	-6.12	10.49	3
8	133.13	102.58	80.64	109.33	-14.06	20.98	3
9	115.90	91.31	76.65	97.09	-10.01	16.59	3

Bobot *WKJ*

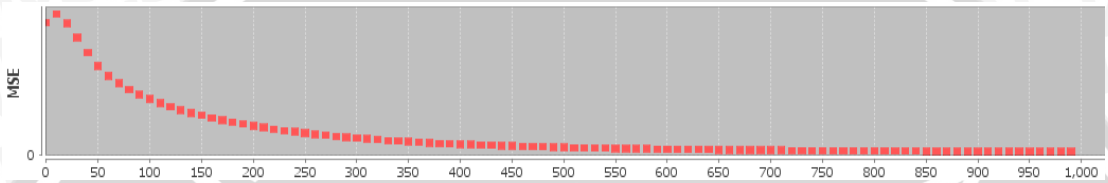
WKJ		K = 1
J	0	0.63
	1	-1.18
	2	-0.28
	3	-5.10
	4	-5.65
	5	4.91

Bobot *Vji*

Vji		j				
		1	2	3	4	5
i	0	-0.26	-0.92	2.54	0.71	-1.83
	1	-0.97	-0.59	-3.04	0.13	4.00
	2	-0.80	-0.43	-0.54	2.91	1.08
	3	-0.98	0.28	-1.63	1.05	1.68
	4	-0.31	-0.25	-1.87	1.86	1.54
	5	0.004	-0.26	0.85	-2.16	-2.74
	6	-1.51	-0.55	-4.26	-4.57	3.27

AKURASI DATA UJI = 9/9 * 100% = 100.0%

GRAFIK MSE



5. Pengujian *Learning Rate*

Mode Ekstraksi = 4

Maks Iterasi = 1000

Learning Rate = 0.1

Persentase Data Latih = 90.0%

Data Latih Target 1 = 30

90.0% Data Latih Target 1 = 27

10.0% Data Uji Target 1 = 3

Data Latih Target 2 = 30

90.0% Data Latih Target 2 = 27

10.0% Data Uji Target 2 = 3

Data Latih Target 3 = 30

90.0% Data Latih Target 3 = 27

10.0% Data Uji Target 3 = 3

Data ke-	R	G	B	Y	U	V	Target
1	45.69	54.92	49.45	51.58	-1.03	-5.13	1
2	40.13	45.99	45.04	44.17	0.45	-3.51	1
3	68.50	72.03	61.15	69.80	-4.23	-1.09	1
4	70.04	77.81	66.46	74.26	-3.80	-3.64	1
5	73.38	85.21	61.86	79.08	-8.44	-4.94	1
6	74.65	84.48	64.70	79.35	-7.18	-4.07	1
7	70.45	79.82	63.07	75.17	-5.92	-4.09	1
8	81.96	84.39	78.21	83.03	-2.34	-0.87	1
9	25.29	28.66	20.01	26.69	-3.27	-1.21	1
10	51.52	56.30	36.26	52.64	-8.04	-0.94	1
11	77.46	81.47	68.94	78.91	-4.87	-1.21	1
12	42.63	50.73	46.90	47.91	-0.48	-4.60	1
13	80.19	87.04	77.79	84.01	-3.03	-3.29	1
14	43.88	53.15	29.12	47.68	-9.11	-3.30	1
15	69.83	77.13	59.17	72.96	-6.76	-2.70	1
16	57.23	63.96	43.87	59.71	-7.77	-2.13	1

17	62.77	66.34	55.37	64.08	-4.26	-1.10	1
18	66.55	68.80	62.03	67.41	-2.62	-0.71	1
19	81.04	96.48	92.22	91.45	0.41	-9.07	1
20	41.45	39.55	25.13	38.51	-6.57	2.61	1
21	57.20	56.09	50.58	55.85	-2.56	1.23	1
22	29.42	32.67	29.52	31.37	-0.90	-1.68	1
23	49.68	60.87	51.56	56.51	-2.41	-5.95	1
24	34.25	36.31	33.76	35.43	-0.81	-1.01	1
25	44.91	54.47	50.92	51.25	-0.14	-5.52	1
26	26.30	28.60	25.30	27.56	-1.10	-1.08	1
27	44.57	54.30	38.16	49.59	-5.61	-4.37	1
28	143.50	142.54	104.23	138.59	-16.85	4.42	2
29	143.61	154.80	141.49	150.07	-4.16	-5.55	2
30	148.00	153.75	123.06	148.66	-12.54	-0.47	2
31	141.01	152.24	141.81	147.82	-2.90	-5.86	2
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
80	95.63	73.37	37.46	76.02	-18.93	17.28	3
81	95.18	73.29	37.82	75.88	-18.68	17.01	3

9 Data Latih 3 Embun Jelaga (Target 1), 3 Thrips (Target 2) , dan 3 Kutu Batok (Target 3)

Data ke-	R	G	B	Y	U	V	Target
1	36.72	42.62	32.85	39.78	-3.39	-2.65	1
2	34.73	38.98	33.76	37.15	-1.65	-2.09	1
3	37.00	42.07	43.25	40.72	1.26	-3.24	1
4	140.48	151.42	139.96	146.97	-3.39	-5.58	2
5	144.53	154.55	137.95	149.79	-5.76	-4.50	2
6	136.92	148.47	135.02	143.61	-4.17	-5.76	2
7	106.82	91.18	82.42	94.95	-6.12	10.49	3
8	133.13	102.58	80.64	109.33	-14.06	20.98	3
9	115.90	91.31	76.65	97.09	-10.01	16.59	3

Bobot *WKJ*

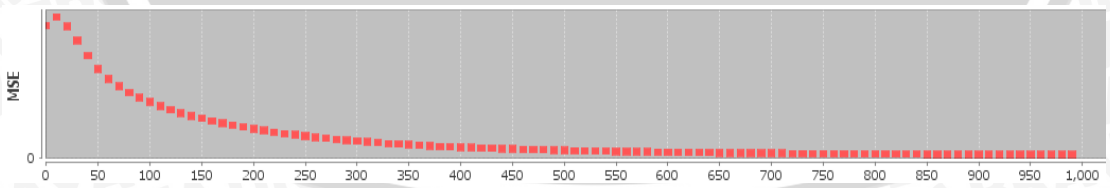
WKJ		K = 1
J	0	-1.46
	1	-4.29
	2	-1.59
	3	2.73
	4	5.64
	5	-3.42

Bobot *Vji*

Vji		j				
		1	2	3	4	5
i	0	0.46	0.71	-0.13	-2.38	0.98
	1	-0.55	0.086	2.148	3.28	-0.22
	2	1.72	0.168	-0.28	1.01	1.16
	3	1.54	-0.809	1.200	1.96	0.19
	4	1.72	-0.04	0.835	2.09	0.72
	5	-0.36	-0.831	-1.115	-2.67	-2.25
	6	-3.82	-0.996	2.271	4.19	-1.99

AKURASI DATA UJI = 9/9 * 100% = 100.0%

GRAFIK MSE



LAMPIRAN C BIAYA IDENTIFIKASI PENYAKIT JERUK DI LABORATORIUM

Jenis Uji	Biaya	Keterangan
Indeksing 5 virus jeruk 1. CPVD 2. CTV 3. CVEV 4. CPsV 5. CEV	 Rp 150.000,- Rp 100.000,- Rp 50.000,- Rp 50.000,- Rp 50.000,-	• Jumlah tanaman yang di indeksing minimal 25 sampel • Biaya tidak termasuk pajak dan transport petugas Balitjestro ke lokasi monitoring • Biaya dapat berubah berdasarkan fluktuasi harga bahan kimia dan immunoreagent
Pembersihan pohon induk jeruk	Rp 17.000.000,-	• Jumlah tanaman induk (BF) pervarietas sebanyak 15 tanaman dan telah bebas 5 penyakit utama (CVPD, CTV, CVEV, CPsV dan CEV) • Ukuran tinggi tanaman 25-30 cm • Biaya belum termasuk pajak • Biaya dapat berubah sewaktu-waktu berdasarkan fluktuasi harga bahan kimia
Analisa DNA Keragaman Genetik	Rp 300.000,-	• Harga per sampel 3 primer
Identifikasi, Pemotretan Mikroskopis Hama/Penyakit Jeruk dan Buah Subtropika	Minimal Rp 150.000,-	• Harga per jenis patogen baru • Rp 50.000,- untuk hama/penyakit yang sudah teridentifikasi sebelumnya

Sumber : http://balitjestro.litbang.pertanian.go.id/layanan/pengujian-di-laboratorium/