DETEKSI PENYAKIT TANAMAN DAUN BAYAM MENGGUNAKAN METODE GLCM DAN *ARTIFICIAL NEURAL NETWORK* (ANN)

Oleh

FITRIATY ISHANAN

T3116085

SKRIPSI

Untuk memenuhi salah satu syarat ujian
guna memperoleh gelar Sarjana

PROGRAM SARJANA
TEKNIK INFORMATIKA
UNIVERSITAS ICHSAN GORONTALO
GORONTALO
2020

DETEKSI PENYAKIT TANAMAN DAUN BAYAM MENGGUNAKAN METODE GLCM DAN *ARTIFICIAL NEURAL NETWORK* (ANN)

Oleh

FITRIATY ISHANAN

T3116085

SKRIPSI

Untuk memenuhi salah satu syarat ujian
guna memperoleh gelar Sarjana

PROGRAM SARJANA
TEKNIK INFORMATIKA
UNIVERSITAS ICHSAN GORONTALO
GORONTALO
2020

PENGESAHAN SKRIPSI

DETEKSI PENYAKIT TANAMAN DAUN BAYAM MENGGUNAKAN METODE GLCM DAN *ARTIFICIAL NEURAL NETWORK* (ANN)

Oleh

FITRIATY ISHANAN

T3116085

SKRIPSI

Untuk memenuhi salah satu syarat ujian guna memperoleh gelar Sarjana Program Studi Teknik Informatika, ini telah disetujui oleh Tim Pembimbing

Gorontalo, 01 Juli 2020

Pembimbing I	Pembimbing II
Hastuti Dalai, M.Kom	**Yusrianto Malago, S.Kom, M.Kom**
NIDN. 0918038803	**NIDN. 0909108901**

PERSETUJUAN SKRIPSI

DETEKSI PENYAKIT TANAMAN DAUN BAYAM MENGGUNAKAN METODE GLCM DAN *ARTIFICIAL NEURAL NETWORK* (ANN)

Oleh

FITRIATY ISHANAN

T3116085

Di Periksa oleh Panitia Ujian Strata Satu (S1)
Universitas Ichsan Gorontalo
Gorontalo, 01 Juli 2020

1. Ketua Penguji
   Zohrahayaty, M.Kom ……
2. Anggota
   Sudirman S. Panna, M.Kom ……
3. Anggota
   Muis Nanja, M.Kom ……
4. Anggota
   Hastuti Dalai, M.Kom ……
5. Anggota
   Yusrianto Malago, S.Kom, M.Kom ……

Mengetahui :

Dekan Fakultas Ilmu Komputer	**Ketua Program Studi**
Zohrahayaty, M.Kom	**Irvan Abraham Salihi, M.Kom**
NIDN. 0912117702	**NIDN. 0928028101**

PERNYATAAN SKRIPSI

Dengan ini saya menyatakan bahwa :

1. Karya tulis (Skripsi) saya ini adalah asli dan belum pernah diajukan untuk mendapatkan gelar akademik (Sarjana) baik Universitas Ichsan Gorontalo maupun diperguruan tinggi lainnya.
2. Karya tulis (Skripsi) saya ini adalah murni gagasan, rumusan, dan penelitian saya sendiri, tanpa bantuan pihak lain, kecuali arahan dari Tim Pembimbing.
3. Dalam karya tulis (Skripsi) saya ini tidak terdapat karya atau pendapat yang telah dipublikasikan orang lain, kecuali secara tertulis dicantumkan sebagai acuan/sitasi dalam naskah dan dicantumkan pula dalam daftar pustaka.
4. Pernyataan ini saya buat dengan sesungguhnya dan apabila dikemudian hari terdapat penyimpangan dan ketidakbenaran dalam peryataan ini, maka saya bersedia menerima sanksi akademik berupa pencabutan gelar yang telah diperoleh karena karya tulis ini, serta sanksi lainnya sesuai dengan norma-norma yang berlaku di Universitas Ichsan Gorontalo.

Gorontalo, 01 Juli 2020

Yang Membuat Peryataan,

Fitriaty Ishanan

Abstract

Disease detection in this study was divided into three groups of leaf diseases namely White Rust, Curly Virus, Manganese Deficiency. This study aims to classify diseases based on leaf images using the feature extraction method that is the gray level co-occurrence matrix. This research method consists of: conversion of RGB data to grayscale, image normalization, detection of leaf diseases, feature extraction and classification. Data collection taken from real data amounted to 101 images used in this study consisted of 81 training data and 20 testing data which are RGB image data with JPG format. Each data consists of three disease categories, for training data consists of 16 white rust leaf images, 13 manganese deficiency leaf images and 14 curly virus leaf image. Whereas the testing data consisted of 6 white rust leaf images, 7 manganese deficiency leaf images and 7 curly virus leaf disease images. The image data is processed into grayscale image which is then detected by leaf disease. After the leaf disease is obtained, segmentation is performed on the location of the leaf found. The next step is to calculate the characteristics using the gray level co-occurrence matrix. The algorithm used for the classification process is the Artificial Neural Network algorithm. The final test results show that the proposed method has been able to detect spinach leaf disease with the accuracy calculated using a confusion matrix of 90%. Thus the application of the gray level co-occurrence matrix method and Artificial Neural Network on the problem of detection of spinach leaf disease needs to be developed again accurate results.

Keywords : *Detection of Leaf Disease, Gray Level Co-Occurrence Matrix, Artificial Neural Network*

Abstrak

Deteksi penyakit dalam penelitian ini dibagi menjadi tiga kelompok penyakit daun yaitu Karat Putih, Virus Keriting, Kekurangan Mangan. Penelitian ini bertujuan untuk mengklasifikasi penyakit berdasarkan citra daun dengan menggunakan metode fitur ekstraksi yaitu *gray level co-occurrence matrix*. Metode penelitian ini terdiri dari: konversi data *rgb* ke *grayscale*, normalisasi citra, deteksi penyakit daun, ekstraksi fitur dan klasifikasi. Pengumpulan data yang diambil dari data real berjumlah 101 *image* yang dipakai dalam penelitian ini terdiri 81 data training dan 20 data testing yang merupakan data *image* RGB dengan format JPG. Masing-masing data terdiri dari tiga kategori penyakit, untuk data training terdiri dari 16 *image* daun penyakit karat putih, 13 *image* daun penyakit kekurangan mangan dan 14 *image* daun penyakit virus keriting. Sedangkan untuk data testing terdiri dari 6 *image* daun penyakit karat putih, 7 *image* daun penyakit kekurangan mangan dan 7 *image* daun penyakit virus keriting. Data citra tersebut diolah menjadi citra grayscale yang kemudian dilakukan deteksi penyakit daun. Setelah didapat penyakit daun kemudian dilakukan *segmentasi* pada bagian lokasi daun yang ditemukan. Yang selanjutnya dilakukan perhitungan ciri menggunakan *gray level co-occurrence matrix*. Algoritma yang digunakan untuk proses klasifikasi adalah algoritma *Artificial Neural Network*. Hasil akhir pengujian menunjukkan bahwa metode yang diusulkan telah mampu mendeteksi penyakit daun bayam dengan hasil akurasi yang dihitung menggunakan *confusion matrix* sebesar 90%. Dengan demikian penerapan metode *gray level co-occurrence matrix* dan *Artificial Neural Network* pada masalah deteksi penyakit daun bayam perlu dikembangkan lagi hasil akurat.

Kata kunci : *Deteksi Penyakit Daun Bayam, Gray Level Co-Occurrence Matrix, Artificial Neural Network*

KATA PENGANTAR

Alhamdulillah, penulis dapat menyelesaikan Skripsi ini dengan judul: "Deteksi Penyakir Tanaman Daun Bayam Menggunakan Metode GLCM dan *Artificial Neural Network* (ANN)" sebagai salah satu syarat Ujian Akhir guna memperoleh gelar Sarjana Komputer pada Program Studi Teknik Informatika Fakultas Ilmu Komputer Universitas Ichsan Gorontalo.

Penulis menyadari sepenuhyna bahwa skripsi ini tidak mungkin terwujud tanpa bantuan dan dorongan dari berbagai pihak, baik bantuan moril maupun material. Untuk itu, dengan segala keikhlasan dan kerendahan hati, penulis mengucapkan banyak terima kasih dan segala penghargaan yang setinggi-tingginya kepada:

1. Bapak Muhammad Ichsan Gaffar, SE.M.Ak, selaku Ketua Yayasan Pengembangan Ilmu Pengetahuan dan Teknologi (YPIPT) Ichsan Gorontalo;
2. Bapak Dr. Abdul Gaffar La Tjokke,M.si, selaku Rektor Universitas Ichsan Gorontalo ;
3. Ibu Zohrahayati,M.Kom, selaku Dekan Fakultas Ilmu Komputer Universitas Ichsan Gorontalo;
4. Bapak Sudirman S. Panna,M.Kom, selaku Pembantu Dekan I Bidang Akademik Fakultas Ilmu Komputer Universitas Ichsan Gorontalo;
5. Ibu Irma Surya Kumala Idris,M.Kom, selaku Pembantu Dekan II Bidang Administrasi Umum dan Keuangan Fakultas Ilmu Komputer Universitas Ichsan Gorontalo;
6. Bapak Sudirman Melangi,M.Kom, selaku Pembantu Dekan III Bidang Kemahasiswaan Fakultas Ilmu Komputer Universitas Ichsan Gorontalo;
7. Bapak Irvan Abraham Salihi,M.Kom, selaku Ketua Jurusan Teknik Informatika Fakultas Ilmu Komputer Universitas Ichsan Gorontalo;
8. Ibu Hastuti Dalai,M.Kom, selaku Pembimbing I;
9. Bapak Yusrianto Malago,M.Kom, selaku Pembimbing II;

10. Bapak dan Ibu Dosen Universitas Ichsan Gorontalo yang telah mendidik dan mengajarkan berbagai disiplin ilmu kepada penulis;
11. Kedua Orang Tua aya yang tecinta, atas segala kasih sayang, jerih payah dan doa restunya dalam membesarkan dan mendidik penulis;
12. Rekan-rekan seperjuangan yang telah banyak memberikan bantuan dan dukungan moril yang sangat besar kepada penulis;
13. Kepada semua pihak yang ikut membantu dalam penyelesaian proposal ini yang tak sempat penulis sebutkan satu-persatu;

Semoga Allah SWT, melimpahkan balasan atas jasa-jasa mereka kepada kami.

Penulis menyadari sepenuhnya bahwa apa yang telah dicapai ini masih jauh dari kesempurnaan dan masih banyak terdapat kekurangan. Oleh karena itu, penulis sangat mengharapkan adanya kritik dan saran yang konstruktif. Akhirnya penulis berharap semoga hasil yang telah dicapai ini dapat bermanfaat bagi kita semua, Amin.

Gorontalo, Juni 2020

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

BAB I

PENDAHULUAN

1.1 Latar Belakang

Di pelosok manapun penyakit tanaman sangat merugikan dunia pertanian dan perkebunan *(Guan et al 2009)*, diperlukan proses identifikasi melalui deteksi penyakit tanaman agar tidak terjadi penyebaran virus ke tanaman yang lain. Pendeteksian penyakit tanaman yang dimaksud bisa dilakukan dengan secara langsung pada tanaman. Menurut para ahli deteksi penyakit tanaman harus benar-benar teliti di dalam melakukan pendeteksian tersebut karena minimnya waktu jika dilakukan di perkebunan yang luas oleh karena itu dibutukan waktu yang banyak apalagi memakai metode manual. Deteksi penyakit tanaman secara otomatis merupakan topik yang menjadi fokus utama para pemilik perkebunan ataupun yang tertarik dengan bidang agrikultur. Berdasarkan pengamatan bahwa perkembangan teknologi visual dan produk digital ditunjang karena menggunakan pemrosesan gambar jika dilakukan dengan metode deteksi otomatis. Berdasarkan dari bentuk daun yang terkena penyakit, tekstur dan warna dapat dikategorikan masuk dalam pendeteksian. Bagaimanapun juga, deteksi otomatis ini masih banyak kekurangan dikarenakan sangat kompleksnya tanda dari penyakit tanaman itu sendiri [1].

Tanaman daun bayam (blitum album) merupakan jenis tanaman penting bagi kesehatan karena mempunyai banyak kandungan zat gizi alami seperti vitamin, mineral, dan antioksidan. Sangat membantu penjagaan kesehatan jika kita selalu mengkonsumsi sayuran bayam tersebut karena kandungan gizi dan serat alaminnya pun diakui para masyarakat bahwa dapat melindungi dan menjaga daya tahan tubuh serta melancarkan pencernaan. Hal ini memudahkan sisa-sisa metabolisme yang tidak berguna keluar dari tubuh sehingga tidak mengendap dan menimbulkan penyakit, begitu banyak hal positif yang kita dapatkan jika kita mengkonsumsi tanaman bayam ini setiap hari, Akan tetapi meskipun tanaman ini merupakan tanaman yang kaya akan vitamin dan nutrisinya, bayam juga biasa terserang penyakit berikut ini

beberapa penyakit daun bayam yaitu karat putih, virus keriting (spinach blight), kekurangan mangan(Mn) [2].

Pupuk organik dapat memperbaiki struktur tanah dan efek negatif yang ditimbulkan oleh pupuk ini tidak sebesar pupuk anorganik. Hal ini sesuai dengan pernyataan Aryal dan Xu (1999) bahwa pupuk sintetis atau pupuk anorganik menyebabkan pencemaran lingkungan, gangguan kesehatan, dan kerusakan tanah. Pupuk organic dapat berupa pupuk kandang, pupuk kompos, dan pupuk organik cair. Pupuk kompos yang merupakan pupuk organik padat terbuat dari sisa tumbuhan yang mati yang telah terdegradasi. Pupuk ini banyak dijual dipasaran dan bahkan dapat diproduksi sendiri dari limbah rumah tangga dan limbah organic lainnya [3]

Di dalam daun bayam (blitum album) terdapat cukup banyak kandungan protein, begitu banyaknya khasiat dari tanaman bayam ini jadi selain dimakan bisa juga dibuat obat tradisional dan yang pasti sangat manjur dan dapat diramu sendiri. Oleh karena itu diharapkan dapat mempermudah pekerjaan petani tanaman bayam dalam mengidentifikasi penyakit pada tanaman daun bayam menggunakan cara organik agar dapat segera di tanggulangi sehingga memperkecil resiko penyakit yang mengganggu produktivitas tanaman bayam [4].

Dalam proses identifikasi penyakit tanaman bayam tersebut dapat menggunakan teknologi komputasi dari pengolahan citra digital (*Digital Image Processing*). Karena diera saat ini teknologi yang berkembang semakin pesat dengan adanya aplikasi dari teknologi 4 berbagai bidang. Beberapa bidang inilah yang direkomendasi masuk dalam teknologi pengolahan citra yaitu seperti pertanian, industri, kedokteran, geologi, kelautan dan lain sebagainnya [1]. Meluasnnya penerapan dari teknologi ini membangun semangat penelitian yang bergerak pada bidang pertanian untuk menunjang pertanian itu sendiri. Berbagai macam penerapan yaitu terkait deteksi penyakit tanaman seperti deteksi kematangan pada buah sangat cocok jika dikembangkan dalam bidang pertanian[1].

Metode GLCM (Gray-Level Coooccurrance Matrix), yang digunakan untuk menganalisa Citra digital. Teknik ini sangat kental dengan matematika, seiring dengan pesatnya komputasi dengan bantuan komputer menjadikan

penemuan ini lebih bermanfaat dan banyak digunakan. Sehingga memudahkan para peneliti untuk mendapatkan nilai pada fitur ekstraksi dan menghasilkan tingkat akurasi yang lebih tinggi [5]. Berdasarkan penelitian di atas membuktikan bahwa dari hasil penelitian mereka metode GLCM adalah metode yang paling efektif digunakan dan fitur ekstraksi yang paling baik dari fitur ekstraksi lain.

Peneliti lain pernah meneliti dengan judul penelitian Identifikasi Tumbuhan Obat Herbal Berdasarkan Citra Daun Menggunakan Algoritma Gray Level Co-occurance Matrix dan K-Nearest Neighbor berdasarkan jarak terdekat antara citra uji dan citra latih, Algoritma tersebut dapat digunakan untuk mengidentifikasi tumbuhan obat herbal berdasarkan citra daun dengan akurasi rata-rata sebesar 83,33% [6].

Adapun beberapa metode komputasi yang dapat digunakan untuk menyelesaikan masalah deteksi, antara lain : Artivicial Neural Network (ANN), Support Vector Machine (SVM), Learning Vector Quantization (LVQ), Deep Learning, K-Nearest Neighbor (K-NN), ID-3, C4.5, CART, Random Forest, Naive Bayes (NB), Linear Discriminant Analysis (LDA), dan K-Support Vector Nearest Neighbor [7].

Diantara metode-metode tersebut algoritma Artivicial Neural Network (ANN) yang paling sering digunakan untuk deteksi penyakit pada daun tembakau dengan akurasi(73%)[8] dan itu merupakan akurasi yang lebih baik dibandingkan metode-metode lainnya yang pernah diterapkan untuk deteksi penyakit daun. Seperti algoritma SVM dengan akurasi 71% [9], LDA Fuzzy K-Nearest Neighbor Lp-Norm dengan akurasi 72% [10], itulah beberapa contoh yang penulis paparkan.

Berdasarkan latar belakang di atas, maka dengan ini peneliti melakukan penelitian dengan judul: **“Deteksi Penyakit Tanaman Daun Bayam Menggunakan Metode GLCM dan *Artificial Neural Network* (ANN)”.** Semoga ini dapat memberikan kontribusi.

1.2 Identifikasi Masalah

Berdasarkan latar belakang yang telah di susun di atas maka dapat di tarik beberapa permasalahan yang timbul pada pendeteksian penyakit tanaman bayam itu sendiri antara lain:

1. Minimnya perkembangan teknologi visual dan produk digital ditunjang dalam hal pendeteksian penyakit tanaman bayam (blitum album).
2. Masih menggunakan cara manual untuk mendeteksi penyakit tanaman daun bayam (blitum album).

1.3 Rumusan Masalah

Adapun masalah yang diteliti dalam penelitian ini adalah:

1. Bagaimana melakukan deteksi penyakit pada tanaman daun bayam (blitum album) menggunakan metode GLCM dan *Aritificial Neural Network* (ANN)?
2. Bagaimana implementasi pendeteksian penyakit tanaman daun bayam (blitum album) dengan menggunakan metode GLCM dan *Artificial Neural Netwok* (ANN)?

1.4 Tujuan Penelitian

1. Untuk mengetahui deteksi penyakit pada tanaman daun bayam (blitum album) menggunakan metode GLCM dan Aritificial Neural Network (ANN).
2. Untuk mengetahui implementasi pendeteksian penyakit tanaman daun bayam (blitum album) dengan menggunakan metode GLCM dan *Artificial Neural Netwok* (ANN).

1.5 Manfaat Penelitian

1.5.1 Manfaat Teoritis

Penelitian ini di harapakan dapat memberikan ilmu pengetahuan di bidang teknologi komputer khususnya dalam ilmu pengetahuan *GLCM* dan *Artificial Neural Netwok* (ANN).

1.5.2 Manfaat Praktis

Hasil penelitian ini dapat di gunakan sebagai salah satu bahan dasar dalam penerapan metode *GLCM* dan *Artificial Neural Network* dalam melakukan pendeteksian penyakit tanaman.

BAB II

TINJAUAN PUSTAKA

2.1 Tinjauan Studi

Ada beberapa penelitian yang terkait dengan metode ANN seperti dibawah ini :

Table 2.1. Penelitian Terkait

NO	PENELITI	JUDUL	METODE	HASIL
1	Diana Laily Fithri (2013) [8]	Deteksi penyakit pada daun tembakau dengan menerapkan algoritma artificial neural network	Metode Artificial Neural Network	Dari hasil pengujian terhadap sampel sebanyak 20 daun untuk tahap training dan 10 sampel daun untuk tahap testing, dengan perbandingan penyakit bercak daun dan cacar daun adalah 50 : 50, learning rate sebesar 0,7, lapisan masukan sebanyak 8 buah, dan 1 buah lapisan luaran, didapat bahwa metode ANN Perceptron memiliki persentase keberhasilan pengenalan penyakit sebesar 61% - 73% untuk data non-learning, dan 100% untuk data learning pada kedua jenis daun tersebut.
2	Agnesia, Gina (2015) [11]	Implementasi Algoritma Backpro-pagation Jaringan Syaraf Tiruan	Metode Backpro-pagation Jaringan Syaraf Tiruan	Pelatihan dalam penelitian ini menghasilkan bobot ideal dengan menggunakan 1 lapisan input, 2 lapisan tersembunyi, dan 1 lapisan output dengan 104 neuron pada lapisan tersembunyi pertama, 5

NO	PENELITI	JUDUL	METODE	HASIL
		(Mustika Mentari, 2016) (Laily, 2013)uan untuk Mendeteksi Penyakit Tanaman Karet (Hevea brasiliensis		neuron pada lapisan tersembunyi kedua,fungsi aktivasi lapisan tersembunyi pertama dan kedua adalah sigmoid bipolar, fungsi aktivasi lapisan output adalah linear, dan learning rate 0,01, pada 148.013 iterasi dengan error 1x10-5.
3	Fernandya Riski Hartantri, Ardi Pujiyanta (2014) [12]	Deteksi Penyakit dan Serangan Hama Tanaman Buah Salak Menggunakan Jaringan Syaraf Tiruan (Jst) dengan Metode Perceptron	Metode Perceptorn	Berdasarkan pelatihan yang dilakukan dari 30 data latih dengan variasi nilai alpha 0.1, 0.3, 0.6, 0.9, dan 1, didapat nilai learning rate 0.6 dengan 19 iterasi untuk waktu pembelajaran terbaik yaitu ± 00.49 detik. Dari penelitian yang dilakukan menghasilkan Aplikasi Deteksi Penyakit dan Serangan Hama Menggunakan Jaringan Syaraf Tiruan Dengan Metode Perceptron telah mampu digunakan untuk pengenalan pola penyakit tanaman buah salak, serta memberikan solusi yang cukup akurat dari hasil diagnosa sesuai dengan hasil pelatihan didapat hasil akurasi pengujian yaitu 90% dengan jumlah data yang diuji 60 terdapat 54 data yang akurat.

4	Fittria Shofrotun Ni'mah, T. Sutojo, De Rosal Ignatius Moses Setiadi(2018) [6]	Identifikasi Tumbuhan Obat Herbal Berdasarkan Citra Daun Menggunakan Algoritma Gray Level Co-occurence Matrix dan K-Nearest Neighbor	Algotitma Gray Level Co-occurrence Matrix dan K-Nearst Neighbor	Analisis tekstur yang digunakan adalah GLCM dengan mengekstrak nilai kontras, korelasi, energi dan homogenitas. Klasifikasi dilakukan dengan KNN. Hasil percobaan menunjukkan akurasi identifikasi menggunakan metode 9-fold cross validation mencapai 83.33% dengan menggunakan 9 subset.

2.2 Tinjauan Pustaka

2.2.1 Deteksi

Deteksi adalah suatu proses untuk memeriksa atau melakukan pemeriksaan terhadap sesuatu dengan menggunakan cara dan teknik tertentu. Deteksi dapat digunakan untuk berbagai persoalan, misalnya dalam sistem pendeteksi suatu penyakit, dimana sistem mengidentifikasi masalah-masalah yang berhubungan dengan penyakit yang biasa disebut gejala. Tujuan dari deteksi adalah memecahkan suatu persoalan atau maslah dengan berbagai cara tergantung metode yang diterapkan sehingga menghasilkan sebuah solusi [13].

2.2.2 Penyakit Tanaman

Penyakit tanaman dapat menyebabkan pertumbuhan serta perkembangan tanaman tidak semaksimal mungkin. Penyebap suatu tanaman yang tidak kokoh lagi dan berkembang dengan maksimal disebabpakan oleh penyakit yang menyerang serta hama pengganggu.

ada berbagai macam hama dan penyakit tanaman yang menjadi fokus bagi para petani atau peladang. Akibatnya, mereka mengalami kerugian besar karena maslah hama dan penyakit tanaman yang menyerang perkebunan atau pertanian mereka.

Hama dan penyakit tanaman perlu diketahui agar penanggulangan teratasi. Apalagi dalam budidaya berbagai tanaman, terkadang hama dan penyakit menjadi persoalan yang biasa terjadi, namun, kita dapat mengendalikanya agar tetap mendapatkan hasil yang maksimal dan terbaik [14].

2.2.3 Daun Bayam

Penyakit pada tanaman daun bayam biasanya kurang merugikan,terutama bila kondisi lingkungan sekitar tanaman terpelihara dengan baik. Penyakit penting yang sering dijumpai pada tanaman bayam yaitu sebagai beriku [15].

1. Penyakit Karat Putih

Gambar 2.1 : Penyakit Karat Putih

Terbentuknya bercak-bercak putih yang agak melepuh pada daun, terutama pada sisi bawah tanaman bayam. Pada populasi karat meningkat dapat terjadi kerusakan daun sehingga mengakibatkan hasil dan mutu daun menurun. Penyakit ini disebabkan oleh cendawan Albogo candida. Pengendalian penyakit cukup dilakukan dengan membongkar tanaman dan melakukan peremajaan kembali bila serangan sudah parah.

2. Penyakit Virus Keriting (*Spinach Blight*)

Gambar 2.2 : Penyakit Virus Keriting (*Spinach Blight*)

Penyebab penyakit adalah sejenis virus dan jenis cucumbor mosaic virus (CMC). Serangan virus menyebabkan daun menyempit,mengkerut, menggulung dan mengecil. Pada daun timbul bercak-bercak. Permukaan daun berwarna kuning dan belang-belang(mosaic). Penyakit ini paling banyak menyerang daun mudah.

Pengendalian dan pencegahan dilakukan dengan cara :

Mencabut tanaman yang sakit dan membakar tanaman yang sudah terinfeksi agar tidak menyebar, kemudian melakukan peremajaan tanaman kembali melakukan penggiliran tanaman (*rotas*) dengan tanaman yang tidak sefamili dengan bayam sebagai tindakan pencegahan. Selain itu, lahan harus dibersihkan dari tanaman-tanaman yang mungkin sebagai inang bagi penyakit ini.

3. Penyakit Kekurangan Mangan (Mn)

Gambar 2.3 : Penyakit Kekurangan Mangan (Mn)

Gejala penyakit ini ditandai dengan timbulnya bitnik-bintik kuning pada daun dan tepi daun menjadi keriting. Pertumbuhan daun menjadi lambat. Gejala serangan ini banyak dijumpai bila keadaan cuaca sangat panas. Sebagai penyebab dari gejala tersebut adalah kekurangan mangan (Mn).

Pengendalian dan pencegahan dilakukan sebagai berikut :

Tanah dapat diberi multonik yang mengandung Mn saat serangan pertama muncul pencegahannya dilakukan dengan memberikan kapur saat pengolahan tanah, terutama pada tanah yang diduga kekurangan mangan Mn.

2.2.4 Pengolahan Citra Digital

Pengolahan citra digital adalah pemrosesan citra, khususnya dengan menggunakan komputer sehingga citra itu kualitasnya menjadi lebih baik dan menghasilkan informasi nilai untuk tiap-tiap warnanya. Pengolahan citra dapat didefinisikan dengan pengertian computer vision yang dikaitkan dengan perolehan citra, pemrosesan, klasifikasi, penganan, dan pencakupan keseluruhan, pengambilan keputusan yang diikuti pengidentifikasian citra [16]. Pengolahan citra digital adalah proses yang bertujuan untuk memanipulasi dan menganalisis citra menggunakan bantuan komputer. Pengolahan citra digital dapat dikelompokkan dalam dua aspek kegiatan :

1. Memperbaiki kualitas gambar, agar dapat lebih mudah diinterpretasi oleh mata manusia.
2. Mengolah informasi yang terdapat pada suatu gambar untuk keperluan pengenalan objek secara otomatis.

Aspek kedua sangat erat hubungannya dengan ilmu pengenalan pola (*pattern recognition*) yang umumnya bertujuan mengenali suatu objek dengan cara mengekstrak informasi penting yang terdapat pada suatu citra[17].

2.2.5 Jenis Citra Digital

Pada aplikasi pengolahan citra digital umumnya, citra digital dibagi menjadi 3, RGB *image*, *Grayscale image* dan *Binary image* [18]:

1. RGB

   Citra berwarna atau citra RGB adalah citra yang menyediakan beberapa warna dalam bentuk komponen R (merah), G (hijau), dan B (biru). Masing-masing komponen menggunakan delapan bit yang nilainya berkisar antara antara 0 sampai dengan 255 [19].

2. *Grayscale*

   Grayscale adalah suatu citra citra yang hanya memiliki warna tingkat keabuan. Gambar *grayscale* 8 bit memiliki 256 tingkat warna abu-abu mulai

dari putih hingga hitam [20]. Tingkat *Grayscale* dapat di lihat pada gambar 2.4 :

Gambar 2.4 : Tingkat Grayscale [21]

Untuk mendapatkan nilai *grayscale* dapat dilakukan dengan mengambil rata-rata dari nilai R, G dan B [21]. seperti di bwah ini :

$$s = \frac{r + g + b}{3} \qquad (1)$$

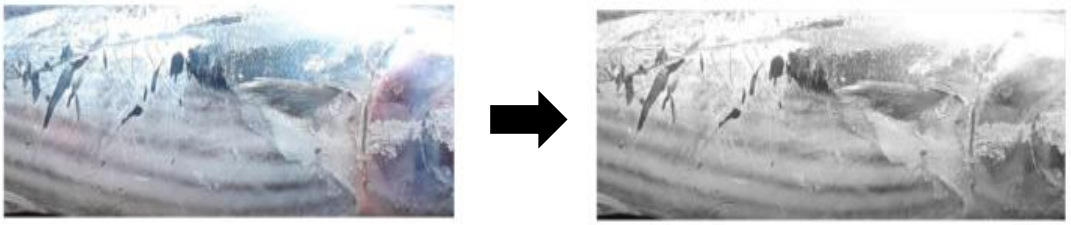
3. Binary Image

Binary image terdiri dari warna hitam atau putih, karena hanya ada dua warna untuk per piksel, maka hanya perlu 1 bit setiap piksel (0 dan 1) atau apabila d lam 8 bit (0 dan 255), sehingga sangat efisien dalam hal penyimpanan. atau gambar arsitektur. Binary image merupakan hasil pengolahan dari *black and white image* [18].

2.2.6 Proses Grayscale

Proses grayscale adalah proses dimana citra input yang terdiri dari tiga layer warna RGB dirubah hanya menjadi satu layer, dimana layer tersebut menyatakan tingkat keabuan disetiap titik pada citra. Proses ini diperlukan karena pada proses ekstraksi ciri GLCM memerlukan citra grayscale sebagai input-nya. Proses perubahan ini dilakukan dengan melakukan penjumlahan komponen RGB yang telah diberikan bobot masing-masing [22]

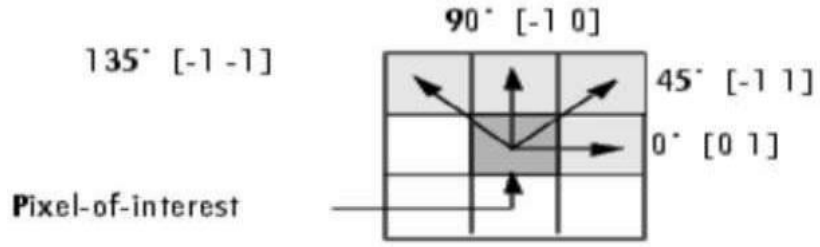
Gambar Asli Gambar Grayscale



Gambar 2.5 : Grayscale[22]

2.2.7 Gray Level Co-occurrence Matrix

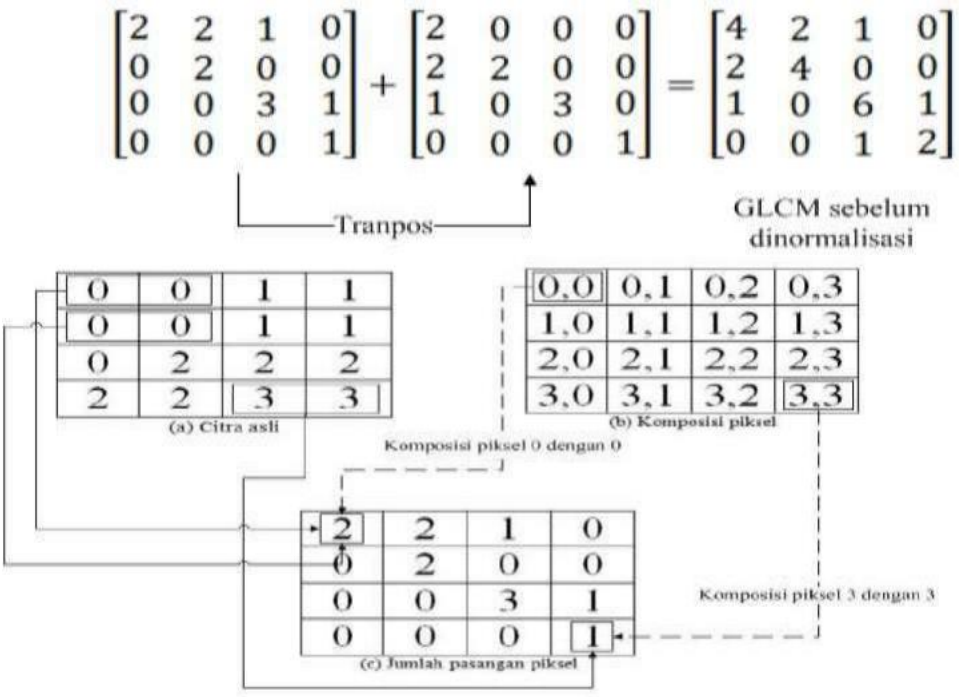
Metode Gray Level Co-occurrence Matrix (GLCM) adalah salah satu ekstraksi order kedua pada fitur statistic tekstur. Ekstraksi order kedua menunjukan hubungan statistic antara 2 piksel. GLCM adalah sebuah matriks dengan jumlah baris dan kolom sebanding dengan jumlah *Gray Level* (G) dalam suatu citra. Metode GLCM menggunakan citra berkala keabuan(Grayscale). GLCM merupakan suatu metode untuk melakukan ekstraksi ciri berbasis statistikal, perolehan ciri diperoleh dari nilai piksel matrix yang mempunyai nilai tertentu dan membentuk suatu sudut pola . GLCM merepresentasikan hubungan dua piksel yang bertetangga dimana dua piksel yang berhubugan tersebut memiliki intensitas keabuan tertentu serta memiliki jarak dan arah tertentu diantara keduanya. Jarak dinyatakan piksel dan arah dinyatakan sudut. Jarak dapat bernilai 1, 2, 3 dan seterusnya sedangkan arah dapat bernilai 0°, 45°, 90°, 135° dan seterusnya [23].



Gambar 2.6 : Piksel dengan berbagai sudut [23]**.**

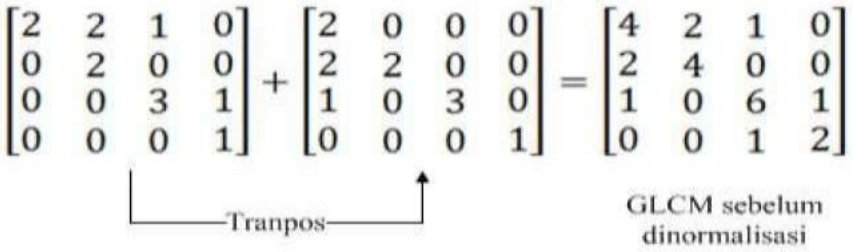
Sebagai ilustrasi, ketetanggan piksel yang dapat dipilih ke arah timur (kanan). Salah satu untuk merepresentasikan hubungan ini yaitu berupa (1,0), yang menyatakan

hubungan dua piksel bernilai 1 diikuti dengan piksel bernilai 0, sehingga jumlah kelompok piksel memenuhi hubungan tersebut dihitung[23].



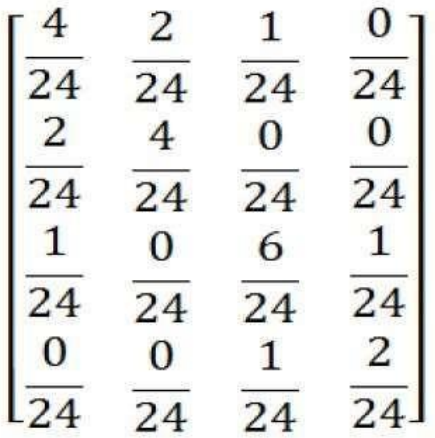
Gambar 2.7 : Contoh penentuan awal matriks GLCM[23].

Matriks pada Gambar 2.5 dinamakan matrix framework. Matriks ini kemudian diolah menjadi matriks dimetris dengan cara menambahkan hasil transposnya [23]. Seperti yang terlihat pada Gambar 2.6.



Gambar 2.8 : Matriks framework menjadi matriks simetris [23].

Untuk menghilangkan ketergantungan pada ukuran citra, nilai-nilai elemen GLCM perlu dinormalisasikan sehingga jumlahnya bernilai 1. Dengan demikian, hasil normalisasi dari matriks GLCM pada Gambar 2.7.



Gambar 2.9. : Normalisasi matriks GLCM [23].

Untuk mendapatkan fitur tekstur GLCM, hanya 14 besaran yang diusulkan untuk dipakai. Beberapa fitur yang akan dipakai nantinya adalah *contrast*, *homogenity*, *entropy*, *energy* sebagai berikut :

1. Contrast digunakan untuk mengukur frekuensi spasial dari citra dan perbedaan momen GLCM. Perbedaan yang dimaksudkan adalah perbedaan tinggi dan rendah pixel, Contrast merupakan ukuran keberadaan variasi atas keabuan pixel citra dihitung.

$$Contrast = \sum_{i1}\sum_{i2}(i1 - i2)^2\, p(i_1, i_2) \qquad (2)$$

2. Homogeneity digunakan untuk mengukur homogenitas yaitu ukuran kedekatan distribusi masing-masing elemen pada matriks GLCM ke matriks GLCM diagonal. Homogenity dihitung dengan cara berikut :

$$Homogeneity = \sum_{i1}\sum_{i2}\frac{p(i_1, i_2)}{1 + |i_1 - i_2|} \qquad (3)$$

3. Entropy adalah mengukur heterogenitas dan berkorelasi dengan standar deviasi. Penghitungan Entrophy digunakan untuk mengukur kompleksitas (keacakan) citra, entropy akan bernilai tinggi ketika citra tidak seragam. Entropy dihitung dengan cara berikut :

$$Entropy = -\sum_{i1}\sum_{i2} p(i_1, i_2)\log p(i_1, i_2) \quad (4)$$

4. Energy adalah pengukuran intensitas keseragaman piksel. Sebuah keadaan homogen mengandung hanya sedikit gray level tetapi memiliki nilai piksel Pij yang tinggi, oleh karena itu jumlah dari pangkat Pij akan tinggi.

$$Energy = \sum_{i1}\sum_{i2} p^2\,(i_1, i_2) \quad (5)$$

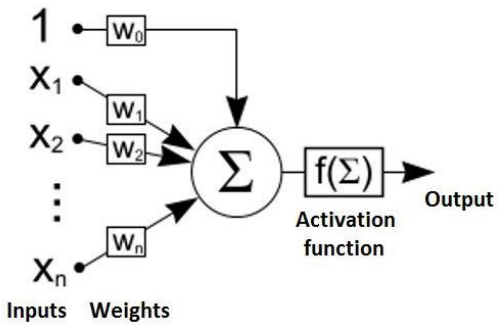
2.2.8 Artificial Neural Network

Artificial Neural Network (ANN) adalah salah satu algortima komputasi cerdas pada bidang machine learning. Algoritma ini terdistribusi paralel, terbuat dari unit-unit yang sederhana, dan memiliki kemampuan untuk menyimpan pengetahuan yang diperoleh secara eksperimental dan siap pakai untuk berbagai tujuan. ANN meniru otak manusia dari sudut: (1) Pengetahuan diperoleh oleh *network* dari lingkungan, melalui suatu proses pembelajaran; (2) Kekuatan koneksi antar unit yang disebut *synaptic weights*, berfungsi untuk menyimpan pengetahuan yang telah diperoleh oleh *network* tersebut. Pada tahun 1943, Mc. Culloch dan Pitts memperkenalkan model matematika yang merupakan penyederhanaan dari struktur sel saraf yang sebenarnya.

$$y = f(\sum_{i=1}^{n} X_i W_i)$$

Korelasi antara ketiga komponen pada persamaan di atas yaitu: Signal x berupa vektor berdimensi n (x1 , x2 ,…,xn)T akan mengalami penguatan oleh *synapse* w (w1, w2 ,…,wn)T. Selanjutnya akumulasi dari penguatan tersebut akan mengalami transformasi oleh fungsi aktifasi f. Fungsi f ini akan memonitor, bila akumulasi penguatan signal itu telah melebihi batas tertentu, maka sel *neuron* yang

semula berada dalam kondisi “0”, akan mengeluarkan signal “1”. Berdasarkan nilai *output* tersebut (y), sebuah *neuron* dapat berada dalam dua status: “0” atau “1”. *Neuron* disebut dalam kondisi *firing* bila menghasilkan output bernilai “1”.



Gambar 2.10 : Neural Network Model

Gambar 1 memperlihatkan bahwa sebuah *neuron* memiliki tiga komponen:

1. Synapse (w1 , w2 ,…,wn) T;
2. Alat penambah (adder);
3. fungsi aktifasi (f).

Sebuah *neural network* dapat dianalisa dari dua sisi [24]:

1. Bagaimana *neuron-neuron* tersebut dirangkaikan dalam suatu jaringan (arsitektur);
2. Bagaimana jaringan tersebut dilatih agar memberikan *output* sesuai dengan yang dikehendaki (algoritma pembelajaran).

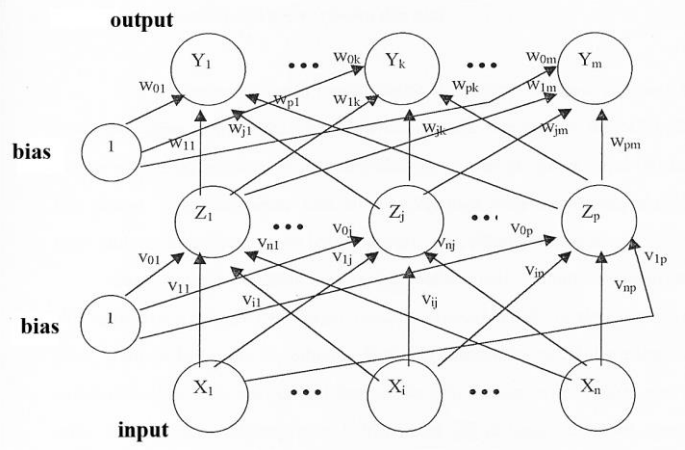
Algoritma pembelajaran ini menentukan cara bagaimana nilai penguatan yang optimal diperoleh secara otomatis. Berdasarkan arsitekturnya, *neural network* dapat dikategorikan, antara lain, *singlelayer neural network*, *multilayer neural network*, dan *recurrent neural network*. Berbagai algoritma pembelajaran antara lain *Hebb’s law*, *Delta rule*, *Backpropagation algorithm*, dan *Self Organizing Feature Map*. Berawal dari diperkenalkannya model matematika *neuron* oleh Mc. Culloch dan Pitts, penelitian di bidang *neural network* berkembang cukup pesat, dan mencapai puncak keemasan pertama pada era tahun 60, dan puncak kedua pada pertengahan tahun 80-an [24].

Dewasa ini, *neural network* telah diaplikasikan di berbagai bidang. Hal ini dikarenakan *neural network* memiliki kelebihan-kelebihan sebagai berikut [24]:

1. Dapat memecahkan problema *non-linear* yang umum dijumpai di aplikasi;
2. Kemampuan memberikan jawaban terhadap *pattern* yang belum pernah dipelajari *(generalization);*
3. Dapat secara otomatis mempelajari data numerik yang diajarkan pada jaringan tersebut.

2.2.9 Penerapan Metode Artificial Neural Network

Salah satu metode pelatihan terawasi pada *neural network* adalah metode *backpropagation*, di mana ciri dari metode ini adalah meminimalkan *error* pada output yang dihasilkan oleh jaringan. Pada gambar di bawah ini, *unit input* dilambangkan X, *hidden unit* dilambangkan Z, dan *unit output* dilambangkan Y. Bobot antara X dan Z dilambangkan dengan v sedangkan bobot antara Z dan Y dilambangkan dengan w [24].



Gambar 2.11 : Arsitektur Jaringan Backpropogation

Penerapan *backpropagation network* terdiri dari 2 tahap [24]:

1. Tahap pelatihan, di mana pada tahap ini diberikan sejumlah data pelatihan dan target;
2. Tahap pengujian atau evaluasi, dilakukan setelah selesai tahap pelatihan.

Pada intinya, pelatihan dengan metode *backpropagation* terdiri dari tiga langkah, yaitu [24]:

1. Data dimasukkan ke *input* jaringan (*feedforward*);
2. Perhitungan dan propagasi balik dari *error* yang bersangkutan;

Pembaharuan (*adjustment*) bobot dan bias.

Saat umpan maju (*feedforward*), setiap unit *input* (Xi) akan menerima sinyal *input* dan akan menyebarkan sinyal tersebut pada tiap *hidden unit* (Zj), setiap *hidden unit* kemudian akan menghitung aktivasinya dan mengirim sinyal (zj) ke tiap unit *output*. Kemudian setiap unit *output* (Yk) juga akan menghitung aktivasinya (yk) untuk menghasilkan respons terhadap *input* yang diberikan jaringan. Saat proses pelatihan (*training*), setiap unit *output* membandingkan aktivasinya (yk) dengan nilai target (tk) untuk menentukan besarnya *error*. Berdasarkan *error* ini dihitung faktor □k, di mana faktor ini digunakan untuk mendistribusikan *error* dari *output* ke *layer* sebelumnya. Dengan cara yang sama, faktor □j juga dihitung pada *hidden* unit Zj, di mana faktor ini digunakan untuk memperbaharui bobot antara *hidden layer* dan *input layer*. Setelah semua faktor ditentukan, bobot untuk semua *layer* diperbaharui. Notasi yang digunakan dalam algoritma pelatihan [24]

x = Data *training input* x = (x1,…,xi,…,xn)

t = Data *training* untuk *target output* t = (t1,…,tk,…,tm)

α = *Learning rate* yaitu parameter untuk mengontrol perubahan bobot \
selama pelatihan.

Xi = Unit *input* ke-i

Zj = *Hidden* unit ke-j

Yk = Unit *output* ke-k

v0j = Bias untuk *hidden* unit ke-j

vij = Bobot antara unit *input* ke-i dengan *hidden* unit ke-j

w0k = Bias untuk unit *output* ke-k

Wjk = Bobot antara *hidden* unit ke-j dengan unit *output* ke-k

δk = Faktor koreksi *error* untuk bobot wjk

δj = Faktor koreksi *error* untuk bobot vij

m = Momentum

Tahap-tahap pelatihan [24]

1. Tahap 0: Inisialisasi bobot dan bias. Baik bobot maupun bias dapat diset dengan sembarang angka (acak) dan biasanya angka di sekitar 0 dan 1 atau -1 (bias positif atau negatif).
2. Tahap 1: Jika *stopping condition* masih belum terpenuhi, jalankan tahap 2-9
3. Tahap 2: Untuk setiap data *training*, lakukan tahap 3-8
4. Tahap 3: Setiap unit *input* (Xi,i=1,…,n) menerima sinyal *input* xi dan menyebarkan sinyal tersebut pada seluruh unit pada *hidden layer*. Perlu diketahui bahwa *input* xi yang dipakai di sini adalah *input* training data yang sudah diskalakan.
5. Tahap 4: Setiap *hidden unit* (Zj,j=1,…,p) akan menjumlahkan sinyal-sinyal *input* yang sudah berbobot, termasuk biasnya,

$$z_{in_j} = V_{0j} + \sum_{i=1}^{n} x_i v_{ij} \quad \text{..... Rumus 2.1}$$

dan memakai fungsi aktivasi yang telah ditentukan untuk menghitung sinyal *output* dari *hidden unit* yang bersangkutan,

$$z_j = f\left(z_{in_j}\right) \text{ … … … … … …} \quad \text{Rumus 2.2}$$

lalu mengirim sinyal *output* ini ke seluruh *unit* pada *unit output*

6. Tahap 5: Setiap *unit output* (Yk,k=1,…,m) akan menjumlahkan sinyal-sinyal *input* yang sudah berbobot termasuk biasnya,

$$y_{in_k} = w_{0k} + \sum_{j=1}^{p} z_j w_{jk} \text{ … … … … … …} \quad \text{Rumus 2.3}$$

7. dan memakai fungsi aktivasi yang telah ditentukan untuk menghitung sinyal *output* dari *unit output* yang bersangkutan,

$$z_j = f\left(z_{in_j}\right) \dots\dots\dots\dots\dots\dots \text{Rumus 2.4}$$

8. Tahap 6: Propagasi balik *error* (*backpropagation of error*). Setiap *unit output* (Yk,k=1,…,m) menerima suatu *target* (*output* yang diharapkan) yang akan dibandingkan dengan *output* yang dihasilkan.

$$\delta_k = (t_k - y_k)\, f'(y_{in_k}) \dots\dots\dots\dots\dots\dots\dots \text{Rumus 2.5}$$

Faktor □k ini digunakan untuk menghitung koreksi *error* (□wjk) yang nantinya akan dipakai untuk memperbaharui wjk, di mana:

$$\Delta w_{jk} = \alpha \delta_k z_j \dots\dots\dots\dots\dots\dots \text{Rumus 2.6}$$

Selain itu juga dihitung koreksi bias □w0k yang nantinya akan dipakai untuk memperbaharui w0k, di mana:

$$\Delta w_{0k} = \alpha \delta_k \dots\dots\dots\dots\dots\dots\dots\dots \text{Rumus 2.7}$$

Faktor δ_k kemudian dikirimkan ke *layer* di depannya

9. Tahap 7: Setiap *hidden unit* (Zj,j=1,…,p) menjumlah *input delta* (yang dikirim dari *layer* pada tahap 6) yang sudah berbobot.

$$\delta\_in_j = \sum_{k=1}^{m} \delta_k w_{jk} \dots\dots\dots\dots\dots\dots \text{Rumus 2.8}$$

Kemudian hasilnya dikalikan dengan turunan dari fungsi aktivasi yang digunakan jaringan untuk menghasilkan faktor koreksi *error* □j, di mana:

$$\delta_j = \delta_{in_j} f'\left(z_{in_j}\right) \dots\dots\dots\dots\dots\dots\dots\dots \text{Rumus 2.9}$$

Faktor □j ini digunakan untuk menghitung koreksi *error* (□vij) yang nantinya akan dipakai untuk memperbaharui vij, di mana:

$$\Delta V_{ij} = \alpha \delta_j x_i \dots\dots\dots\dots\dots\dots\dots \text{Rumus 2.10}$$

Selain itu juga dihitung koreksi bias □v0j yang nantinya akan dipakai untuk memperbaharui v0j, di mana:

$$\Delta V_{0j} = \alpha \delta_j \dots\dots\dots\dots\dots\dots \text{Rumus 2.11}$$

10. Tahap 8: Pembaharuan bobot dan bias: Setiap *unit output* (Yk,k=1,…,m) akan memperbaharui bias dan bobotnya dengan setiap *hidden unit*.

$$w_{jk}(baru) = w_{jk}(lama) + \Delta w_{jk} \ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{Rumus 2.12}$$

11. Demikian pula untuk setiap *hidden unit* akan memperbaharui bias dan bobotnya dengan setiap *unit input.*

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

12. Tahap 9: Memeriksa *stopping condition Jika stop condition* telah terpenuhi, maka pelatihan jaringan dapat dihentikan. Untuk menentukan stopping condition terdapat dua cara yang biasa dipakai, yaitu: (1) Membatasi iterasi yang ingin dilakukan. Misalnya jaringan akan dilatih sampai iterasi yang ke-500. Yang dimaksud dengan satu iterasi adalah perulangan tahap 3 sampai tahap 8 untuk semua *training data* yang ada; (2) Membatasi *error*.

Misalnya menentukan besar antara output yang dikehendaki dan output yang dihasilkan oleh jaringan. Jika terdapat sebanyak m *training* data, maka untuk menghitung *Mean Square Error* digunakan persamaan berikut:

$$MSE = 0.5\,\{(tk1 - yk1)2 + (tk2 - yk2)2 + \cdots + (tkm - ykm)2\}$$

Setelah pelatihan selesai, *backpropagation network* (BPN) dianggap telah pintar sehingga apabila jaringan diberi input tertentu, jaringan akan menghasilkan output seperti yang diharapkan. Cara mendapatkan output tersebut adalah dengan mengimplementasikan metode *backpropagation* yang sama seperti proses pelatihan, tetapi hanya pada bagian umpan majunya saja. Notasi yang digunakan dalam algoritma pengujian:

Xi = Unit input ke-i

Zj = Hidden unit ke-j

Yk = Unit output ke-k

v0j = Bias untuk hidden unit ke-j

vij = Bobot antara unit input ke-i dengan hidden unit ke-j

w0k = Bias untuk unit output ke-k

Wjk = Bobot antara hidden unit ke-j dengan unit output ke-k

Tahap-tahap pengujian pada model ANN, sbb[24]:

1. Tahap 0: Inisialisasi bobot sesuai dengan bobot yang telah dihasilkan pada proses pelatihan di atas;
2. Tahap 1: Untuk setiap *input*, lakukan tahap 2-4;
3. Tahap 2: Untuk setiap *input* i=1,…,n skalakan bilangan dalam *range* fungsi aktivasi seperti yang dilakukan pada proses pelatihan di atas
4. Tahap 3: untuk j=1,…,p:

$$z_{in_j} = v_{0j} + \sum_{i=1}^{n} x_i v_{ij} \quad \text{Rumus 2.13}$$

5. Tahap 4: Untuk k=1,…,m:

$$y\_in_k = w_{0k} + \sum_{j=1}^{p} z_j w_{jk} \quad \text{Rumus 2.14}$$

$$y_k = f(y\_in_k)$$

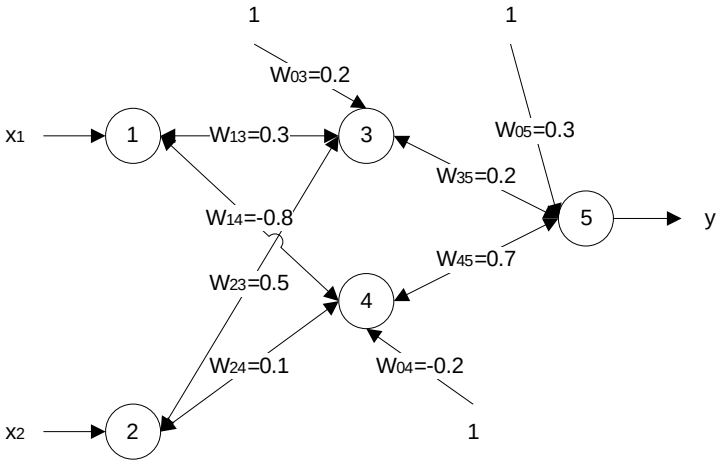
Variabel y_k adalah *output* yang masih dalam skala menurut *range* fungsi aktivasi. Untuk mendapatkan nilai *output* yang sesungguhnya, y_k harus dikembalikan seperti semula.

Contoh :

jaringan terdiri dari 2 unit input (X1 dan X7) dan 1 output yang diambil dari dataset untuk data ke 2, 16, 41, dan 62.

Table 2.2. Contoh manual ANN [24]

No.	X1	X7	y
2.	2	3	0
16.	2	6	1
41.	1	4	1
62.	1	2	0



Gambar 2.12 : Arsitektur Jaringan Backpropogation

Kondisi bobot saat inisialisasi seperti ditunjukkan pada gambar di atas dengan jumlah neuron pada layer tersembunyi = 2.

Laju pembelajaran ($\boldsymbol{\eta}$) = 0.1;

Fungsi aktivasi yang digunakan adalah Sigmoid Biner;

Target *error* = 0.0001 dengan kriteria SSE;

Momentum yang digunakan = 0.95;

Maksimum jumlah iterasi pelatihan = 1,000 kali.

Maka:

1. Nilai pada neuron di hidden layer 3 dan 4:

$$\boldsymbol{v_j(p) = \sum_{i=1}^{n} x_i(p) w_{ij}(p)} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{ Rumus 2.18}$$

$$\boldsymbol{y_j(p) = \frac{1}{1 + e^{-v_j(p)}}}$$

(P) adalah iterasi ke-i xcvx .

$v_3(1) = x1w_{13} + x2w_{23} + 1w_{03} = 2 * 0.3 + 3 * 0.5 + 1 * 0.2 = 2.3$

$y_3(1) = \frac{1}{1 + e^{-2.3}} = \frac{1}{1 + 2.71828^{-2.3}} = 0.908877$

$v_4(1) = x1w_{14} + x2w_{24} + 1w_{04} = 2 * (-0.8) + 3 * 0.1 + 1 * (-0.2) = 1.7$

$$y_4(1) = \frac{1}{1 + e^{-1.7}} = 0.845535$$

2. Nilai pada neuron di output layer:

$$\boldsymbol{v_k(p)} = \sum_{\boldsymbol{j=1}}^{\boldsymbol{m}} \boldsymbol{x_j(p)w_{jk}(p)} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots. \text{ Rumus 2.19}$$

$$\boldsymbol{y_k(p)} = \frac{\boldsymbol{1}}{\boldsymbol{1 + e^{-v_k(p)}}}$$

$\mathrm{v_5(1) = Y3(1)w_{35} + Y4(1)w_{45} + 1w_{05} =}\ 0.908877 * 0.2 + 0.845535 * 0.7 + 1 * 0.3 = 2.054411496$

$$\mathrm{y_5(1)} = \frac{1}{1 + \mathrm{e}^{-2.054411496}} = 0.886392478$$

3. Hitung gradient error untuk neuron pada output laye r:

$$\boldsymbol{e_k(p) = y_{dk}(p) - y_k(p)}$$
$$\boldsymbol{\delta_k(p) = y_k(p) * [1 - y_k(p)] * e_k(P)}$$

Untuk data pertama, target/class nilai yang diharapkan adalah $\boldsymbol{y_d = 0}$ sedangkan keluaran yang didaptkan $\boldsymbol{y_5(1) = 0.886392478}$ **.** Maka hitung *error* pada iterasi pertama untuk data pertama pada neuron 5 di output layer:

$e_5^1(1) = y_d - y_5(1) = 0 - 0.886392478 = \boldsymbol{-0.886392478}$

$$\delta_5(1) = y_5(1) * \left(1 - y_5(1)\right) * e_5^1(1)$$
$$= 0.886392478 * (1 - 0.886392478) * (-0.886392478)$$
$$= -0.089260478$$

4. Hitung koreki bobot untuk output layer, $\Delta w_{35}, \Delta w_{45}, dan\ \Delta w_{05}$:

$$\boldsymbol{\Delta w_{jk}(p) = \eta . y_j(p) * \delta_k(p)}$$

$\Delta w_{35} = \eta * y_3(1) * \delta_5(1) = 0.1 * 0.91 * (-0.089) = -0.008112679$

$\Delta w_{45} = \eta * y_4(1) * \delta_5(1) = 0.1 * 0.85 * (-0.089) = -0.007547282$

$\Delta w_{05} = \eta * 1 * \delta_5(1) = 0.1 * 1 * (-0.089) = -0.0089266048$

5. Hitung *gradient error* pada hidden layer $\delta_3(1)\ dan\ \delta_4(1)$:

$$\boldsymbol{\delta_j(p) = y_j(p) * [\left(1 - y_j(p)\right] + \sum_{k=1}^{1} \delta_k(p) * w_{jk}(p)}$$

$$\delta_3(1) = y_3(1) * \left(1 - y_3(1)\right) * \sum_{k=1}^{1} \delta_k(1).w_{3k}(1)$$
$$= y_3(1) * \left(1 - y_3(1)\right) * \delta_5(1).w_{35}(1)$$
$$= 0.91 * (1 - 0.91) * (-0.089) * 0.2 = -0.001478505$$

$$\delta_4(1) = y_4(1) * \left(1 - y_4(1)\right) * \sum_{k=1}^{1} \delta_k(1).w_{4k}(1)$$
$$= y_4(1) * \left(1 - y_4(1)\right) * \delta_5(1).w_{45}(1)$$
$$= 0.85 * (1 - 0.85) * (-0.089) * 0.7 = -0.008160558$$

6. Hitung koreksi bobot untuk hidden layer $\Delta w_{13}, \Delta w_{23}, \Delta w_{03}, \Delta w_{14}, \Delta w_{24}, dan\ \Delta w_{04}$

$$\boldsymbol{\Delta w_{ij}(p) = \eta * x_i(p) * \delta_j(p)}$$

$\Delta w_{13} = \eta * x1 * \delta_3(1) = 0.1 * 2 * (-0.0015) = -0.0003$

$\Delta w_{23} = \eta * x2 * \delta_3(1) = 0.1 * 3 * (-0.0015) = -0.00044$

$\Delta w_{03} = \eta * 1 * \delta_3(1) = 0.1 * 1 * (-0.0015) = -0.00015$

$\Delta w_{14} = \eta * x1 * \delta_4(1) = 0.1 * 2 * (-0.0082) = -0.00163$

$\Delta w_{24} = \eta * x2 * \delta_4(1) = 0.1 * 3 * (-0.0082) = -0.00245$

$\Delta w_{04} = \eta * 1 * \delta_4(1) = 0.1 * 1 * (-0.0082) = -0.00082$

7. Perbaharui bobot untuk neuron pada hidden layer w_{35}, w_{45}, w_{05}:

$$\boldsymbol{w_{ij}(p + 1) = w_{ij}(p) + \Delta w_{ij}(p)}$$

$w_{35}(2) = w_{35}(1) + \Delta w_{35} = 0.2 + (-0.008112679) = 0.191887321$

$w_{45}(2) = w_{45}(1) + \Delta w_{45} = 0.7 + (-0.007547282) = 0.692452718$

$w_{05}(2) = w_{05}(1) + \Delta w_{05} = 0.3 + (-0.0089266048) = 0.291073952$

8. Perbaharui bobot pada layer tersembunyi, $w_{13}, w_{23}, w_{03}, w_{14}, w_{24}, w_{04}$:

$w_{13}(2) = w_{13}(1) + \Delta w_{13} = 0.3 + (-0.0003) = 0.299704$

$w_{23}(2) = w_{23}(1) + \Delta w_{23} = 0.5 + (-0.00044) = 0.499556$

$w_{03}(2) = w_{03}(1) + \Delta w_{03} = 0.2 + (-0.00015) = 0.199852$

$w_{14}(2) = w_{14}(1) + \Delta w_{14} = 0.8 + (-0.00163) = -0.798368$

$w_{24}(2) = w_{24}(1) + \Delta w_{24} = 0.1 + (-0.00245) = 0.097552$

$w_{04}(2) = w_{04}(1) + \Delta w_{04} = (-0.2) + (-0.00082) = -0.20082$

9. Naikkan 1 langkah iterasi (***p***), kembali ke langkah 2 dan ulangi proses tersebut sampai ***kriteria error*** tercapai.
10. Dengan demikian, didapatkan bobot akhir pada iterasi pertama untuk data pertama[2, 3]:

    $w_{13} = 0.299704$;

    $w_{23} = 0.499556$;

    $w_{03} = 0.199852$;

    $w_{14} = -0.798368$;

    $w_{24} = 0.097552$;

    $w_{04} = -0.20082$;

    $w_{35} = 0.191887321$;

    $w_{45} = 0.692452718$;

    $w_{05} = 0.291073952$

Error yang didapatkan adalah = $= -\mathbf{0.886392478}$. Selanjutnya proses di atas dulangi untuk data ke-dua [2, 6], data ke-tiga [1, 4], dan data ke-empat [1, 2] sehingga iterasi pertama selesai dilakukan. Pada setiap data yang diproses dalam satu iterasi, error keluaran disimpan untuk dihitung sebagai kriteria error, seperti SSE, MSE, dsb. Apabila hasil MSE < error yang didapatkan maka iterasi berhenti, sebaliknya dilakukan perambatan terus hingga batas perulangan/epoch.

2.2.10 Artificial Intelligence (kecerdasan buatan)

AI atau Artificial Intelligence adalah sebuah bidang ilmu yang digunakan untuk membuat hidup manusia lebih baik dari masa ke masa. Pada halnya upaya ini dilakukan dengan memberikan kecerdasan buatan pada mesin agar dapat berpikir seolah-olah seperti manusia.

Komputer tersebut dapat dibuat menjadi sebuah entitas yang cerdas dengan upaya pemberian data-data dalam sebuah database. Selain diberi data, komputer akan diberikan kemampuan untuk mempelajari data. Data terkait dipelajari dan di training. Training dan pembelajaran data ini akan membuat sistem mampu menentukan keputusan dan melakukan tugas untuk memudahkan manusia di masa depan nanti.

Fungsi dari kecerdasan buatan yaitu memiliki cukup banyak fungsi. Artificial Intelligence diharapkan dapat melakukan sebuah hal yang akan mempermudah manusia mulai dari pemrosesan bahasa alami, mengenai persepsi, penalaran, menggerakkan dan manipulasi objek terkait, mengenai pengetahuan, dan juga melakukan pembelajaran.

Jika di ambil kesimpulan secara menyeluruh, kecerdasan buatan ini berusaha untuk membuat aplikasi atau lebih tepatnya lagi robot yang memiliki kecerdasan yang mirip atau bahkan lebih dari kecerdasan yang dimiliki oleh manusia.

Dengan menggunakan kecerdasan buatan, disini manusia seperti bos yang hanya memerintah saja sedangkan aplikasi atw robot cerdas bersangkutan yang mengeksekusi pekerjaan kasar tersebut. Robot dengan kecerdasan buatan akan memiliki pengetahuan yang lebih tinggi dari pada manusia karena untuk memberikan kecerdasan buatan manusia hanya perlu memasukkan data dan sistem yang akan dipelajari [25].

2.2.11 Confusion Matrix

Confusion matrix merupakan suatu metode yang biasanya digunakan untuk melakukan perhitungan akurasi. Confusion matrix memberikan keputusan yang diperoleh penilaian *Performance* deteksi berdasarkan objek dengan benar atau salah (gurunescu). Confusion matrix berisi informasi actual dan prediksi pada sistem deteksi.

Table 2.3. Tabel confusion matrix [26]

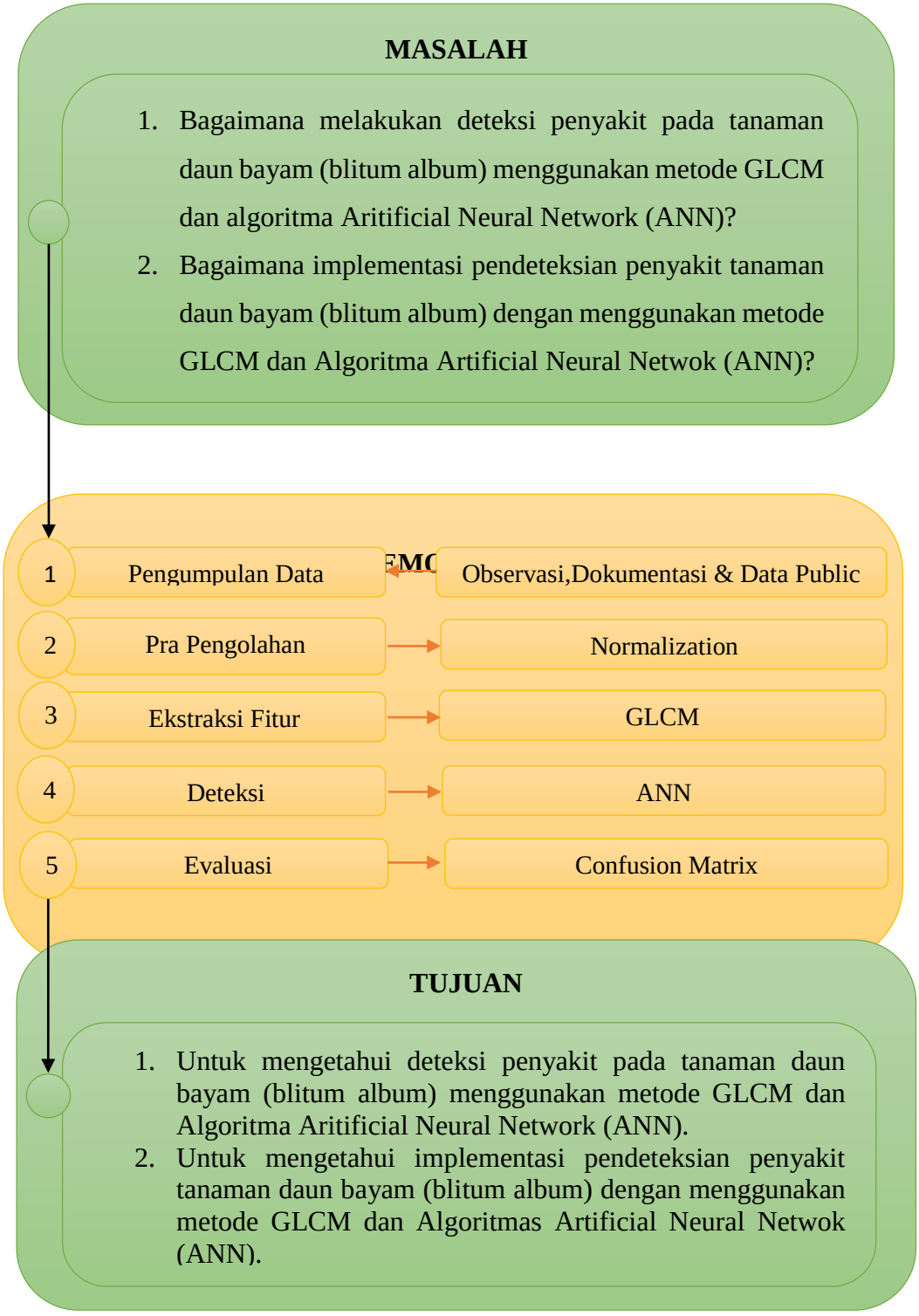
	Actual	Actual
	Positive	Negative
Predicted positive	*TP*	*FP*
Predicted Negative	*FN*	*TN*

Recall	=	TP/(TP+FN)
Precision	=	TP/(TP+FP)
True Positive rate	=	TP/(TP+FN)
False Negative rate	=	FP/(FP+TN)

Keterangan :

- True positive (TP) = proporsi positif dalam dataset yang dideteksi positif.
- True negative (TN) = proporsi negative dalam dataset yang dideteksi negativ.
- False positive (FP) = proporsi negativ dalam dataset yang dideteksi positif.
- False negative (FN) = proporsi negativ dalam dataset yang dideteksi negativ.

2.1 Kerangka Pikir



Gambar 2.13 : Kerangka Pikir

BAB III

METODE PENELITIAN

3.1 Jenis Metode,Subjek,Objek,Waktu dan Lokasi Penelitian

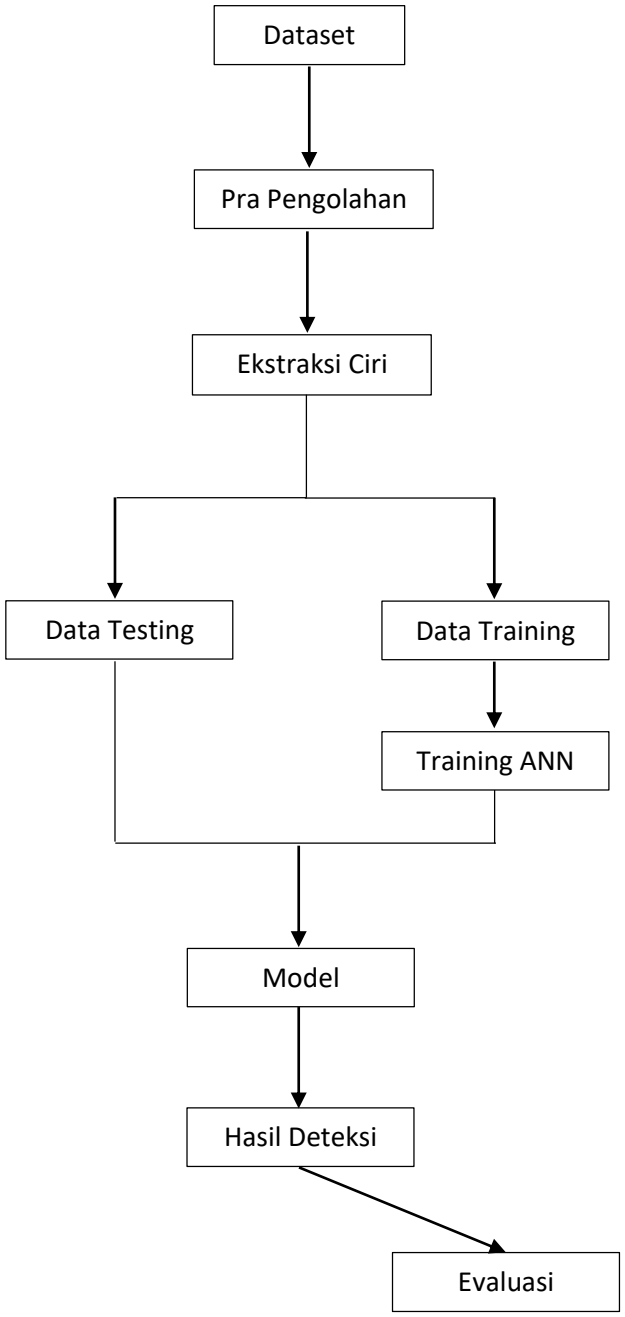
Dipandang dari tingkat penerapanya, maka penelitian ini merupakan penelitian terapan. Dipandang dari jenis informasi yang di olah, maka penelitian ini merupakan penelitian kuantitatif. Dipandang dari perlakuan terhadap data, maka penelitian ini merupakan merupakan konfimatori.

Penelitian ini merupakan metode penelitian *eksperimen* dan *studi kasus*.dengan demikian jenis penelitian ini adalah penelitian *eksperimental* dan *studi kasus.* Subjek penelitian ini adalah deteksi pada penyakit tanaman daun bayam. Penelitian ini mulai dari september 2019 s/d April 2020.

3.2 Pengumpulan Data

Data primer penelitian ini adalah penyakit daun bayam yang merupakan data observasi dan dokumentasi. Citra yang dikumpulkan dibagi menjadi dua dataset yaitu: data latih dan data uji. Data latih yang digunakan sebagai pembanding terhadap data uji pada proses pendeteksian penyakit daun bayam. Sedangkan data uji adalah data digunakan untuk mengetahui hasil deteksi jenis penyakit pada daun bayam yang didapatkan melalui proses deteksi.

3.3 Pemodelan



Gambar 3.1 : **Pemodelan**

3.3.1 Pra Pengolahan Data

Pada tahap ini, data yang telah ada dilakukan praproses terlebih dahulu. Praproses dilakukan secara manual dengan mengubah warna latar pada citra daun menjadi putih. Proses untuk mengubah warna dilakukan dengan memisahkan citra daun dari latarnya dan memindahkan citra tersebut ke sebuah latar putih berukuran 318*318 pixel. Selanjutnya, dilakukan normalisasi histogram pada setiap gambar untuk membuat kesamaan nilai thresholding masing-masing gambar. Setelah setiap gambar telah dinormalisasi,kemudian dilakukan teknik threshold untuk mendeteksi penyakit yang ada. Pada projek ini dilakukan threshold bawah, yaitu mengambil warna yang mendekati nilai 0 pada channel green untuk mendeteksi penyakit blackspot. Selain itu dilakukan threshold atas yaitu mengambil warna yang mendekati nilai 255 pada channel green dan yellow untuk mendeteksi penyakit frogeye.

3.3.2 Ekstraksi Ciri

Pada proses ekstraksi ciri data input akan dibaca dan kemudian diolah dalam dua proses yang berlainan. Proses ekstraksi fitur tekstur menggunakan metode GLCM (*Gray Level Co-Occurrence Matrix*). Masukan dari proses ini adalah citra penyakit daun bayam. Semua citra diambil pada kondisi pencahayaan yang terang. Output dari proses ini adalah matriks Nx1 dimana N adalah jumlah vektor ciri.

3.3.3 Data Training

Data training ini merupakan kumpulan data yang telah terdeteksi penyakitnya yang selanjutnya akan dilatih menggunakan algoritma Artificial Neural Network *backpropagation*. Jumlah data training yang digunakan berupa 10% dari sampel daun bayam dengan deteksi penyakit yang berbeda. Data training ini berupa hasil dari ekstraksi GLCM (*Gray Level Co-Occurrence Matrix*) terhadap penyakit daun bayam.

3.3.4 Training ANN

Proses *Training* menggunakan ANN menjadikan data training yang berupa angka atau parameter hasil dari citra penyakit tanaman daun bayam yang telah diolah melalui fitur ekstraksi GLCM sebagai data inputan untuk algoritma ANN. Arsitektur dari algoritma ANN akan dilakukan secara eksperimen dan studi kasus untuk menenetukan komposisi jumlah layer dan neuron yang tepat untuk mendapatkan hasil kinerja terbaik. Kemudian mengambil mayoritas hasil ketentuan yang telah didapatkan untuk dijadikan pendeteksian dari test.

3.3.5 Model Deteksi

Model deteksi merupakan tahapan yang terdiri dari *input,hiden layer* dan *output* dengan kombinasi bobot terbaik yang dihasilkan dari proses pelatihan data training menggunakan metode ANN. Model ini akan digunakan untuk pendeteksian dan *testing* berdasarkan *output* yang dihasilkan tersebut.

3.3.6 Data Testing

Data testing merupakan data yang telah terdeteksi penyakitnya,untuk menguji data yang telah dilatih,jumlah data training yang digunakan berupa 90% dari sampel daun bayam dengan deteksi penyakit yang berbeda. data testing merupakan ciri dari hasil ekstraksi GLCM (*Gray Level Co-Occurrence Matrix*) terhadap penyakit tanaman daun bayam.

3.3.7 Hasil Deteksi

Hasil deteksi merupakan output, pada data testing yang didapatkan dari proses deteksi yang menggunakan algoritma ANN berdasarkan model yang diperoleh dari data training.

3.3.8 Evaluasi

Proses evaluasi bertujuan untuk mengetahui kinerja dari metode GLCM (*Gray Level Co-Occurrence Matrix*) terhadap penyakit tanaman daun bayam yang digunakan. Proses evaluasi dilakukan pada seluruh data *testing* kemudian hasil *output* yang akan dipetakan kedalam *Confusion Matrix* untuk dihitung nilai akurasinya.

BAB IV

HASIL PENELITIAN

4.1 Hasil Pengumpulan Data

Seperti yang telah diuraikan pada tahapan metode penelitian bahwa dataset gambar yang digunakan pada penelitian ini adalah berupa data real. Kami telah melakukan proses pengumpulan data tersebut dan berhasil di dapat. Data gambar terdiri 101 citra dan terbagi menjadi 81 data training dan 20 data testing dengan format file gambar adalah jpg. Berikut ini beberapa sampel data yang berhasil diambil :

Table 4.1. Data Gambar Penyakit Daun Bayam

Penyakit Karat Putih	Penyakit Virus Keriting	Penyakit Kekurangan Mangan
		
		
		

Untuk melakukan proses analisis pada penelitian ini adalah dengan melakukan pemilihan data gambar yang dijadikan sebagai data training dan data testing, sehingganya dari pekerjaan yang kami lakukan pada penelitian ini kami menentukan dari dataset yang telah berhasil kami dapatkan ditentukan jumlah

data training adalah sebanyak 81 Data gambar dan data testing adalah sebanyak 20 Data gambar.

4.2 Hasil Pemodelan

4.2.1 Pra Pengolahan

Pra Pengolahan Citra (image pre-processing), yaitu proses paling awal dalam pengolahan citra sebelum proses utama dilakukan. Pada tahap ini citra daun akan dikonversi untuk mendapatkan citra daun yang sesuai kebutuhan dimana citra RGB di konversi ke citra Grayscale selanjutnya citra grayscale dikonversi ke histogram equalization untuk mendapatkan hasil. Berikut 2 tahapan yang dilakukan dalam pra-pengolahan:

Proses konversi RGB ke Graycsale dan Grayscale ke Histogram Equalization

Berikut ini merupakan rumus konversi citra RGB ke Grayscale dan rumus konversi citra grayscale ke citra histogram equalization beserta perintah yang digunakan dalam bahasa python dan hasil outputnya:

Rumus Konversi Citra RGB ke Grayscale:

$$S = \frac{R+G+B}{3}$$

Dimana:

K_o = Nilai keabuan hasil histogram

C_i = Distribusi komulatif dari nilai skala keabuan ke-*i* dari citra asli

Round = Fungsi pembulatan ke bilangan terdekat

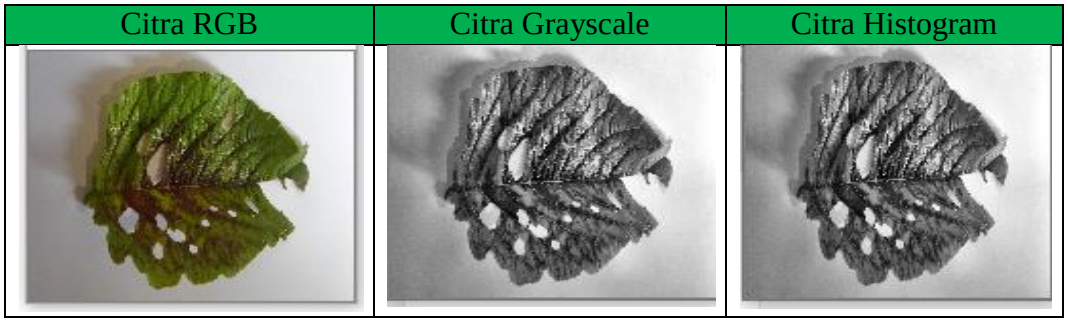
W = Lebar Citra

h = Tinggi Citra

1. **Pengubahan citra warna ke *grayscale* dan normalisasi citra dengan menggunakan *Histogram Equalization***

Pra-pengolahan pertama yang akan dilakukan adalah merubah citra *training* atau citra *testing* yang awalnya berbentuk citra dari RGB (red, green, blue) menjadi citra bentuk grayscale, perubahan ini dilakukan karena citra grayscale memiliki persamaan yang sederhana dan mampu mengurangi kebutuhan memory

dimana nilai warna putih diwakili dengan angka 255 dan nilai warna hitam diwakili dengan angka 0. Setelah citra asli di konversi ke citra abu-abu maka pra-pengolahan selanjutnya adalah normalilasi citra dengan histogram ekualisasi. Histogram ekualisasi adalah sebuah proses yang mengubah distribusi nilai derajat keabuan pada sebuah citra sehingga menjadi seragam. Tujuan dari histogram equalization adalah untuk memperoleh penyebaran histogram yang merata sehingga setiap derajat keabuan memiliki jumlah piksel yang relatif sama. Berikut gambar proses perubahan citra warna ke grayscale dan normalisasi citra abu-abu dengan histogram ekualisasi.



Gambar 4.1. Hasil output dari proses konversi RGB ke Grayscale dan Grayscale ke *Histogram Equalization.*

4.2.2 Citra Segmentasi

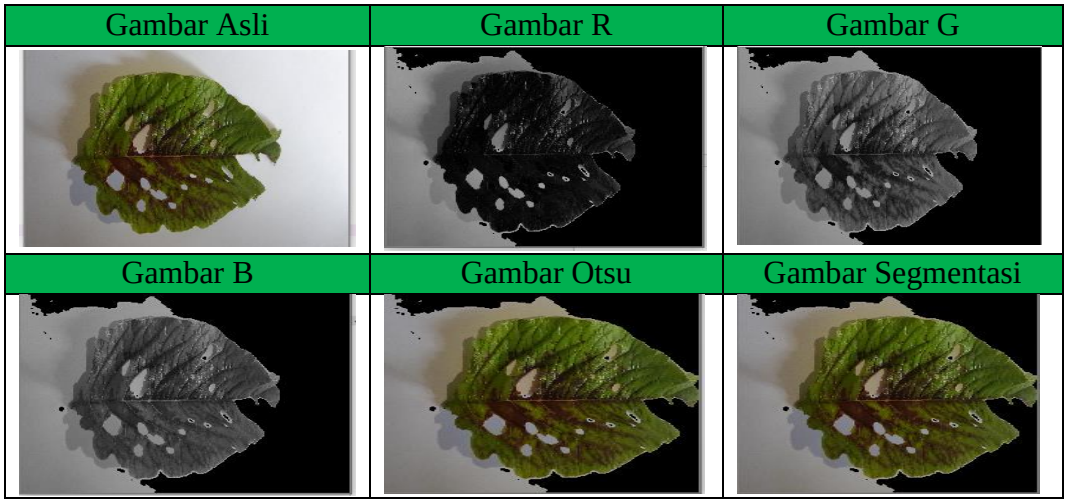
Segmentasi citra merupakan bagian dari proses pengolahan citra. Segmentasi citra (*image segmentation*) mempunyai arti membagi suatu citra menjadi wilayah-wilayah yang homogeny berdasarkan kriteria keserupaan tertentu antara tingkat keabuan suatu piksel dengan tingkat keabuan piksel-piksel tetangganya, kemudian hasil dari proses segmentasi ini akan digunakan untuk proses lebih lanjut.

4.2.2.1. Metode Otsu *Thresholding*

Metode otsu merupakan salah satu metode untuk segmentasi citra digital dengan menggnakan nilai ambang secara otomatis, yakni mengubah citra digital warna abu-abu menjadi hitam putih berdasarkan perbandingan nilai ambang dengan nilai warna piksel citra digital. Metode Otsu *Thresholding* diperkenalkan pertama

kali oleh Nobuyuki Otsu, dalam jurnal ilmiahnya yang berjudul "*A Thresholding* Selection Method from *Grayscale Histogram*" pada tahun 1079.

Berikut ini proses citra RGB yang di ubah ke Segmentasi beserta perintah yang digunakan dalam bahasa python dan hasil outputnya:



Gambar Gambar 4.2. Citra Segmentasi

4.2.3 Ekstraksi Fitur

Tahap ini dilakukan untuk proses mengubah gambar segmentasi menggunakanakan ekstraksi fitur GLCM. GLCM merepresentasikan hubungan dua piksel yang bertetangga dimana dua piksel yang berhubugan tersebut memiliki intensitas keabuan tertentu serta memiliki jarak dan arah tertentu diantara keduanya. Jarak dinyatakan piksel dan arah dinyatakan sudut. Jarak dapat bernilai 1, 2, 3 dan seterusnya sedangkan arah dapat bernilai 0°, 45°, 90°, 135° dan seterusnya. Berikut contoh perhitungan matrix dengan ukuran 10x10 pixel:

Table 4.3. Hasil dari citra

xy	0	1	2	3	4	5	6	7	8	9	10
0	149	146	146	149	147	153	147	148	148	149	147
1	148	146	148	148	145	149	149	151	149	150	151
2	146	145	146	145	147	148	148	147	147	151	149
3	148	148	149	147	146	148	145	147	150	149	152
4	144	149	149	146	147	148	142	149	147	154	152
5	145	146	147	149	149	147	149	152	151	148	150
6	144	146	144	147	146	147	146	146	146	150	148
7	149	147	144	146	149	142	146	148	150	149	148
8	148	147	148	145	146	148	152	146	151	150	148
9	149	143	148	145	145	148	144	147	149	146	147
10	141	145	138	145	149	147	147	146	150	142	150

untuk proses seslanjutnya adalah menghitung nilai dari matriks dengan mengisikan jumlah hubungan spasial. Matriks kookurensi yang akan dibuat adalah hubungan spasial untuk d=1 dan sudut 0^0. Matriks dibuat dengan mengisikan jumlah hubungan sasial yang ada matriks grayscale. Maka didapatkanlah matriks kookurensi seperti pada tabel berikut :

Table 4.4. Hasil Matrix GLCM

xy	138	141	142	143	144	145	146	147	148	149
138	0	0	0	0	0	2	0	0	0	0
141	0	0	0	0	0	1	0	0	0	0
142	0	0	0	0	0	0	1	0	1	2
143	0	0	0	0	0	0	0	0	1	1
144	0	0	0	0	0	0	2	2	0	2
145	1	1	0	0	0	0	4	1	5	2
146	0	0	1	0	3	3	2	6	6	5
147	0	0	0	0	2	2	6	2	5	8
148	0	0	1	1	1	5	4	5	4	3
149	0	0	2	1	1	2	5	8	4	3

Untuk mendapatkan nilai dari normalisasi yaitu dengan cara menjumlahkan semua matriks simetris, kemudian dijadikan pembagi untuk semua piksel yang ada pada matriks simetris.untuk mencari probabilitas maka digunakan perhitungan P(i,j) = (i,j) / total jumlah pasangan.

Ket :

i dan j adalah sifat keabuan dari resolusi 2 piksel yang berdekatan p (i,j) adalah Probabilitas kolom (i,j).

Dimana total jumlah pasangan = 130
Contoh perhitungan manual metode Artificial Neural Network untuk deteksi penyakit tanaman daun bayam

P(145,138) = 2/130 = 0,01538

P(145,141) = 1/130 = 0.00769

P(146,142) = 1/130 = 0.00769

P(148,142) = 1/130 = 0.00769

P(149,142) = 2/130 = 0,01538

P(148,143) = 1/130 = 0.00769

P(149,143) = 1/130 = 0.00769

P(146,144) = 2/130 = 0,01538

P(147,144) = 2/130 = 0,01538

P(149,144) = 2/130 = 0,01538

P(146,144) = 2/130 = 0,01538

P(138,145) = 2/130 = 0,01538

P(141,145) = 2/130 = 0,01538

P(146,145) = 4/130 = 0.03076

P(147,145) = 1/130 = 0.00769

P(148,145) = 5/130 = 0.03846

P(149,145) = 2/130 = 0,01538

P(142,146) = 1/130 = 0.00769

P(144,146) = 3/130 = 0.02307

P(145,146) = 3/130 = 0.02307

P(146,146) = 2/130 = 0,01538

P(147,146) = 6/130 = 0.04615

P(148,146) = 6/130 = 0.04615

P(149,146) = 6/130 = 0.03846

P(144,147) = 2/130 = 0,01538

P(145,147) = 2/130 = 0,01538

P(146,147) = 6/130 = 0.04615

P(147,147) = 2/130 = 0.01538

P(148,147) = 6/130 = 0.03846

P(149,147) = 8/130 = 0.06153

P(142,148) = 1/130 = 0.00769

P(143,148) = 1/130 = 0.00769

P(144,148) = 1/130 = 0.00769

P(145,148) = 6/130 = 0.03846

P(146,148) = 4/130 = 0.03076

P(147,148) = 6/130 = 0.03846

P(148,148) = 4/130 = 0.03076

P(149,148) = 3/130 = 0.02307

P(142,149) = 2/130 = 0,01538

P(143,149) = 1/130 = 0.00769

P(144,149) = 1/130 = 0.00769

P(145,149) = 2/130 = 0,01538

P(146,149) = 6/130 = 0.03846

P(147,149) = 8/130 = 0.06153

P(148,149) = 4/130 = 0.03076

P(149,149) = 3/130 = 0.02307

Berikut tabel matriks hasil perhitungan normalisasi:

Table 4.5. Hasil Normalisasi Matrix

xy	138	141	142	143	144	145	146	147	148	149
138	0	0	0	0	0	0.01538	0	0	0	0
141	0	0	0	0	0	0.00769	0	0	0	0
142	0	0	0	0	0	0	0.00769	0	0.00769	0.01538
143	0	0	0	0	0	0	0	0	0.00769	0.00769
144	0	0	0	0	0	0	0.01538	0.01538	0	0.01538
145	0.00769	0.00769	0	0	0	0	0.03076	0.00769	0.03846	0.01538
146	0	0	0.00769	0	0.02307	0.02307	0.01538	0.04615	0.04615	0.03846
147	0	0	0	0	0.01538	0.01538	0.04615	0.01538	0.03846	0.06153
148	0	0	0.00769	0.00769	0.00769	0.03846	0.03076	0.03846	0.03076	0.02307
149	0	0	0.01538	0.00769	0.00769	0.01538	0.03846	0.06153	0.03076	0.02307

Setelah didapatkan nilai co-occurrence matriks maka proses selanjutnya adalah menghitung nilai ciri statistiknya. Ciri statistik yang akan di hitung adalah *contras*, *homogenitas*, *energy*, *entropy* dan *dissimilarity*. Berikut adalah perhitungan fitur tekstur :

1. Perhitungan contras

$$Contrast = \sum_{i1}\sum_{i2}(i1 - i2)^2\, p(i_1, i_2)$$

i dan j adalah sifat keabuan dari resolusi 2 piksel yang berdekatan p (i,j) adalah Probabilitas kolom (i,j)

Syarat : Ketika nilai i dan j sama, sel berada pada diagonal dan (i-j) = 0. Nilai-nilai ini merepresentasikan pixel yang keseluruhannya mirip dengan tetangga mereka, sehingga mereka diberi bobot 0

Contoh perhitungan contras:

Contras = $(145\text{-}138)^2(0.015)+(145\text{-}141)^2(0.007)+(146\text{-}142)^2(0.007)+(148\text{-}142)^2$
$(0.007)+(149\text{-}142)^2(0.015)+(148\text{-}143)^2(0.007)+(149\text{-}143)^2(0.007)+(146\text{-}144)^2$
$(0.015)+(147\text{-}144)^2(0.007)+(149\text{-}144)^2(0.007)+(138\text{-}145)^2(0.007)+(141\text{-}145)^2$
$(0.007)+(146\text{-}145)^2(0.030)+(147\text{-}145)^2(0.007)+(148\text{-}145)^2(0.038)+(149\text{-}145)^2$
$(0.015)+(142\text{-}146)^2(0.007)+(144\text{-}146)^2(0.023)+(145\text{-}146)^2(0.023)+(146\text{-}146)^2$
$(0.015)+(147\text{-}146)^2(0.046)+(148\text{-}146)^2(0.046)+(149\text{-}146)^2(0.038)+(144\text{-}147)^2$
$(0.015)+(145\text{-}147)^2(0.015)+(146\text{-}147)^2(0.046)+(147\text{-}147)^2(0.015)+(148\text{-}147)^2$
$(0.038)+(149\text{-}147)^2(0.061)+(142\text{-}148)^2(0.007)+(143\text{-}148)^2(0.007)+(144\text{-}148)^2$
$(0.007)+(145\text{-}148)^2(0.038)+(146\text{-}148)^2(0.030)+(147\text{-}148)^2(0.038)+(148\text{-}148)^2$
$(0.030)+(149\text{-}148)^2(0.023)+(142\text{-}149)^2(0.015)+(143\text{-}149)^2(0.007)+(144\text{-}149)^2$
$(0.007)+(145\text{-}149)^2(0.015)+(146\text{-}149)^2(0.038)+(147\text{-}149)^2(0.061)+(148\text{-}149)^2$
$(0.030)+(149\text{-}149)^2(0.023)$

Contras = 0.735 + 0.112 + 0.112 + 0.252 + 0.735 + 0.175 + 0.252 + 0.06 + 0.063 + 0.175 + 0.343 + 0.112 + 0.030 + 0.028 + 0.342 + 0.24 + 0.112 + 0.92 + 0.023 + 0 + 0.046 + 0.184 + 0.342 + 0.135 + 0.06 + 0.046 + 0 + 0.038 + 0.244 + 0.252 + 0.175 + 0.112 + 0.342 + 0.12 + 0.038 + 0 + 0.023 + 0.735 + 0.252 + 0.175 + 0.24 + 0.342 + 0.244 + 0.030 + 0

Contras = **8.973**

2. Menghitung nilai *Energy*

Energy = $((0.015)^2) + ((0.007)^2) + ((0.007)^2) + ((0.007)^2) + ((0.015)^2) + ((0.007)^2)$ + $((0.007)^2)$ + $((0.015)^2)$ + $((0.015)^2)$ + $((0.015)^2)$ + $((0.007)^2)$ + $((0.007)^2)$ + $((0.030)^2)$ + $((0.007)^2)$ + $((0.038)^2)$ + $((0.015)^2)$ + $((0.007)^2)$ + $((0.023)^2)$ + $((0.023)^2)$ + $((0.015)^2)$ + $((0.046)^2)$ + $((0.046)^2)$ + $((0.038)^2)$ + $((0.015)^2)$ + $((0.015)^2)$ +

$((0.046)^2) + ((0.015)^2) + ((0.038)^2) + ((0.061)^2) + ((0.007)^2) + ((0.007)^2) + ((0.007)^2) + ((0.038)^2) + ((0.030)^2) + ((0.038)^2) + ((0.030)^2)$ $((0.023)^2) + ((0.015)^2) + ((0.007)^2) + ((0.007)^2) + ((0.015)^2) + ((0.038)^2) + ((0.061)^2) + ((0.030)^2) + ((0.023)^2)$

Energy = 0.000225 + 0.000049 + 0.000049 + 0.000049 + 0.000225 + 0.000049 + 0.000049 + 0.000225 + 0.000225 + 0.000225 + 0.000049 + 0.000049 + 0.0009 + 0.000049 + 0.001444 + 0.000225 + 0.000049 + 0.000529 + 0.000529 + 0.000225 + 0.002116 + 0.002116 + 0.001444 + 0.000225 + 0.000225 + 0.002116 + 0.000225 + 0.001444 + 0.003721 + 0.000049 + 0.000049 + 0.000049 + 0.001444 + 0.0009 + 0.001444 + 0.0009 + 0.000529 + 0.000225 + 0.000049 + 0.000049 + 0.000225 + 0.001444 + 0.003721 + 0.0009 + 0.000529

Energy = **0.031**

3. Menghitung nilai *Homogenity*

Homogenity=((0.015)/1+(145-138))+((0.007)/1+(145-141))+((0.007)/1+(146-142))+((0.007)/1+(148-142))+((0.015)/1+(149-142))+((0.007)/1+(148-143))+

((0.007)/1+(149-143))+((0.015)/1+(146-144))+((0.007)/1+(147-144))+((0.007)/1+(149-144))+((0.007)/1+(138-145))+((0.007)/1+(141-145))+((0.030)/1+(146-145))+((0.007)/1+(147-145))+((0.038)/1+(148-145))+((0.015)/1+(149-145))+((0.007)/1

+(142-146))+((0.023)/1+(144-146))+((0.023)/1+(145-146))+((0.015)/1+(146-146))+((0.046)/1+(147-146))+((0.046)/1+(148-146))+((0.038)/1+(149-146))+((0.015)/1+(144-147))+((0.015)/1+(145-147))+((0.046)/1+(146-147))+((0.015)/1+(147-147))+((0.038)/1+(148-147))+((0.061)/1+(149-147))+((0.007)/1+(142-148))+((0.007)/1+(143-148))+((0.007)/1+(144-148))+((0.038)/1+(145-148))+((0.030)/1+(146-148))+((0.038)/1+(147-148))+((0.030)/1+(148-148))+((0.023)/1+(149-148))+((0.015)/1+(142-149))+((0.007)/1+(143-149))+((0.007)/1+(144-149))+((0.015)/1+(145-149))+((0.038)/1+(146-149))+((0.061)/1+(147-149))+((0.030)/1+(148-149)) +(0.023)/1+(149-149)

Homogenity = 7.015 + 4.007 + 4.007 + 6.007 + 7.015 + 5.007 + 6.007 + 2.015 + 3.007 + 5.007 + -6.993 + -3.993 + 1.03 + 2.007 + 3.038 + 4.015 +-3.993 + -1.977 + -0.977 + 0.015 + 1.046 + 2.046 + 3.038 + -2.985 + -1.985 + -0.954 + 0.015 +

1.038 + 2.061 + -5.993 + -4.993 + -3.993 + -2.962 +-1.97 + -0.962 + 0.030 + 1.023 + -6.985 + -5.993 + -4.993 + -3.985 + -2.962 + -1.939 + -0.97 + 0.023

Homogenity = **-3.984**

4. Menghitung nilai *Entropy*

Entropy = (0,015.log 0,015) + (0,007.log 0.007) + (0,007.log 0.007) + (0,007.log 0.007) + (0,015.log 0,015) + (0,007.log 0.007) + (0,007.log 0.007) + (0,015.log 0,015) + (0,015.log 0,015) + (0,015.log 0,015) + (0,007.log 0.007) + (0,007.log 0.007) + (0,030.log 0.030) + (0,007.log 0.007) + (0,038.log 0.038) + (0,015.log 0.015) + (0,007.log 0.007) + (0,023.log 0.023) + (0,023.log 0.023) + (0,015.log 0.015) + (0,046.log 0.046) + (0,046.log 0.046) + (0,038.log 0.038) + (0,015.log 0.015) + (0,015.log 0.015) + (0,046.log 0.046) + (0,015.log 0.015) + (0,038.log 0.038) + (0,061.log 0.061) + (0,007.log 0.007) + (0,007.log 0.007) + (0,007.log 0.007) + (0,038.log 0.038) + (0,030.log 0.030) + (0,038.log 0.038) + (0,030.log 0.030) + (0,023.log 0.023) + (0,015.log 0.015) + (0,007.log 0.007) + (0,007.log 0.007) + (0,015.log 0.015) + (0,038.log 0.038) + (0,061.log 0.061) + (0,030.log 0.030) + (0,023.log 0.023)

Entropy = -0.0273586311 + -0.0150843137 + -0.0150843137 + -0.0150843137 + -0.0273586311 + -0.0150843137 + -0.0150843137 + -0.0273586311 + -0.0273586311 + -0.0273586311 + -0.0150843137 + -0.0150843137 + -0.0456863624 + -0.0150843137 + -0.0539682233 + -0.0273586311 + -0.0150843137 + -0.0376802598 + -0.0376802598 + -0.0273586311 + -0.0615131397 + -0.0615131397 + -0.0539682233 + -0.0273586311 + -

0.0273586311 + -0.0615131397 + -0.0273586311 + -0.0539682233 + -0.0740948801 + -0.0150843137 + -0.0150843137 + -0.0150843137 + -0.0539682233 + -0.0456863624 + -0.0539682233 + -0.0539682233 + -0.0376802598 + -0.0273586311 + -0.0150843137 + -0.0150843137 + -0.0273586311 + -0.0539682233 + -0.0539682233 + -0.0456863624 + -0.0376802598

Entropy = **-1.517644177**

Setelah semua proses perhitungan fitur dilakukan, langkah selanjutnya adalah menjumlahkan tiap matriks dari fitur tersebut sehingga didapatkan hasil sebagai berikut :

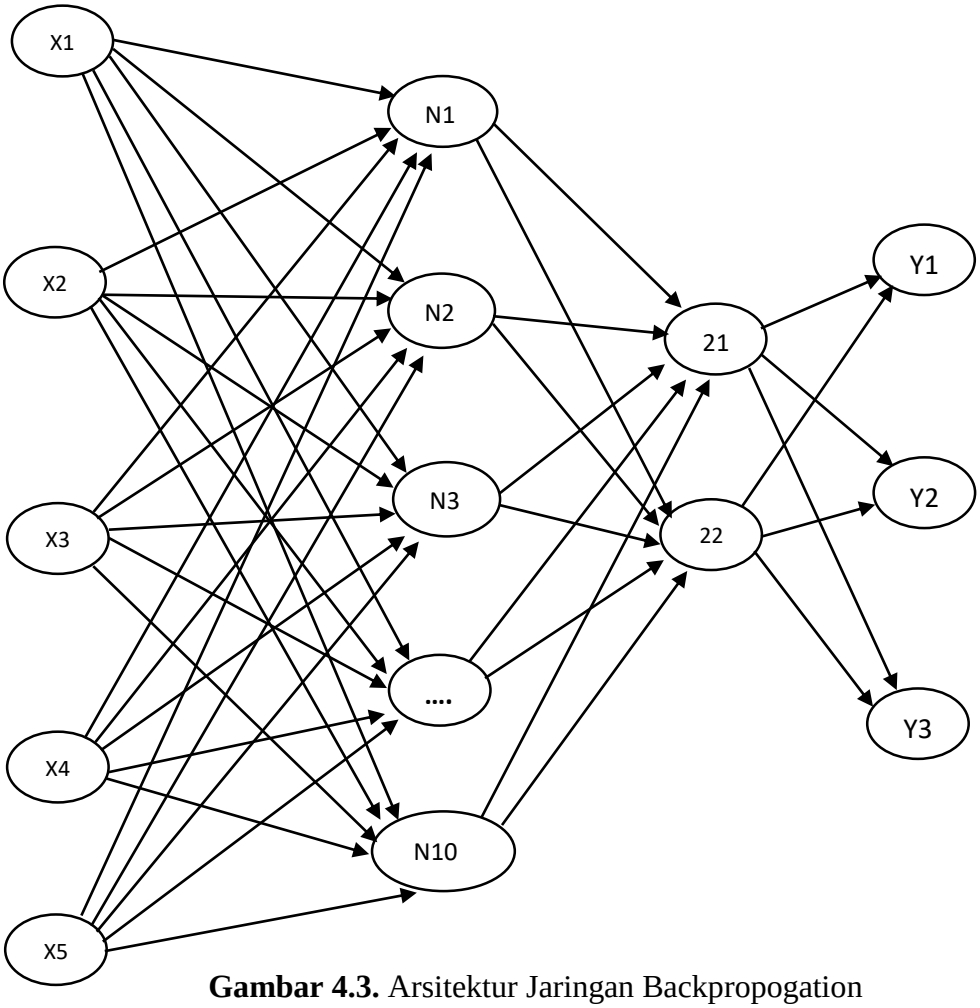
Table 4.6 : Nilai Hasil Fitur Ekstraksi

NO	Kontras	Homogenitas	Entropy	Energy
1	8.973	-3.984	-1.517644177	0.031

Langkah selanjutnya adalah merata-ratakan nilai-nilai dari semua sudut agar didapatkan satu nilai tunggal dari setiap fitur yang telah didapatkan sehingga memudahkan dalam penamaan klasifikasi.

empat ciri yang di dapatkan dari matrix *Co-Occurrence* inilah yang akan digunakan untuk tahap klasifikasi :

4.2.4. Klasifikasi



Gambar 4.3. Arsitektur Jaringan Backpropogation

Ket :

X1 – N1 = w11 0,3	**X2 – N1 =w12 2,3**	**X3 – N1 =w13 0,4**	**X4 – N1 =w14 3,5**	**X5 – N1 =w15 -0,5**
X1 – N2 = w21 0,5	**X2 – N2 = w22 0,8**	**X3 – N2 =w23 3,1**	**X4 – N2 =w24 0,6**	**X5 – N2 =w25 -,0,9**
X1 – N3 = w31 0,2	**X2 – N3 = w32 3,5**	**X3 – N3 = w33 2,6**	**X4 – N3 =w34 1,4**	**X5 – N3 =w35 -0,2**
X1 – … = w41 0,7	**X2 – … = w42 1,6**	**X3 – …= w43 1,8**	**X4 – … = w44 0,9**	**X5 – … = w45 -1,0**
X1–N10=w100 1,2	**X2–N10=w101 1.3**	**X3–10=w102 1,9**	**X4–N10=w103 0,3**	**X5–N10=w104 -1,1**

N1 – Z1 = W201 4,4	**N2 – Z1 = W202 3,0**	**N3– Z1 =W203 1,8**	**N… – Z1 =W204 2,4**	**N200 – Z1 =W205 -0,4**
N1 – Z2 = W301 1,5	**N2 – Z2 = W302 2,2**	**N3 – Z2 = W303 1,0**	**N… – Z2 =W304 3,2**	**N200 – Z2 =W305 -0,7**

Z1 – Y1 =w401 4,3
Z1 – Y2 =w501 1,9
Z1 – Y3 =w601 2,1

Z2 – Y1 =w402 2,5
Z2 – Y2 =w502 1,3
Z2 – Y3 =w602 3,3

Table 4.7. Tahap Training ANN

Image Training	X1	X2	X3	X4	X5	Y
	0.207776	1	0.33199	0.238059	0.708337	Karat Putih
	0.301565	0.713728	0.43954	0.455294	0.563357	Karat Putih
	0.271692	0.888239	0.406856	0.31055	0.632715	Karat Putih
	0.482692	0.032086	0.615273	0.699397	0.294774	Kekurangan Mangan

Tahap training menggunakan algoritma ANN:

Laju pembelajaran ($\boldsymbol{\eta}$) = 0.1;

Fungsi aktivasi yang digunakan adalah Sigmoid Biner;

Target *error* = 0.0001 dengan kriteria SSE

Maksimum jumlah iterasi pelatihan = 1,000 kali.

1. Nilai pada neuron hidden layer N1,N2, N3, ….. , dan N10 :

(P) adalah iterasi ke-i.

$$v_1(1) = x1w_{11} + x2w_{12} + x3w_{33} + x4w_{14} + x5w_{15}$$
$$= 0.207776 * 0.3 + 1 * 2.3 + 0.33199 * 0.4 + 0.238059 * 3.5 + 0.708337 * -0{,}5 = 2.9741668$$

$$y_1(1) = \frac{1}{1 + e^{-2.9741668}} = \frac{1}{1 + 0.0510899845^{-2.9741668}} = 0.9513933295$$

$$v_2(1) = x1w_{21} + x2w_{22} + x3w_{23} + x4w_{24} + x5w_{25}$$
$$= 0.207776 * 0.5 + 1 * 0.8 + 0.33199 * 3.1 + 0.238059 * 0.6 + 0.708337 * -0.9 = 1.4383891$$

$$y_2(1) = \frac{1}{1 + e^{-1.4383891}} = \frac{1}{1 + 0.2373097332^{-1.4383891}} = 0.8082050704$$

$$v_3(1) = x1w_{31} + x2w_{32} + x3w_{33} + x4w_{34} + x5w_{35}$$
$$= 0.207776 * 0.2 + 1 * 3.5 + 0.33199 * 2.6 + 0.238059 * 1.4 + 0.708337 * -0.2 = 4.5963444$$

$$y_3(1) = \frac{1}{1+e^{-4.5963444}} = \frac{1}{1+0.0100886485^{-4.5963444}} = 0.9900121158$$

$$v_{...}(1) = x1w_{41} + x2w_{42} + x3w_{43} + x4w_{44} + x5w_{45}$$
$$= 0.207776 * 0.7 + 1 * 1.6 + 0.33199 * 1.8 + 0.238059 * 0.9$$
$$+ 0.708337 * -1.0 = 1.8489413$$

$$y_{...}(1) = \frac{1}{1+e^{-1.8489413}} = \frac{1}{1+0.1574037215^{-1.8489413}} = 0.8640027515$$

$$v_{10}(1) = x1w_{100} + x2w_{101} + x3w_{102} + x4w_{103} + x5w_{104}$$
$$= 0.207776 * 1.2 + 1 * 1.3 + 0.33199 * 1.9 + 0.238059 * 0.3$$
$$+ 0.708337 * -1.1 = 1.4723592$$

$$y_{10}(1) = \frac{1}{1+e^{-1.4723592}} = \frac{1}{1+0.2293836843^{-1.4723592}} = 0.813415708$$

2. Nilai pada neuron di hiden layer Z1 dan Z2:

$$v_1(1) = n1w_{201} + n2w_{202} + n3w_{203} + \cdots w_{204} + n200w_{205}$$
$$= 0.207776 * 4.4 + 1 * 3.0 + 0.33199 * 1.8 + 0.238059 * 2.4$$
$$+ 0.708337 * -0.4 = 4.7998032$$

$$y_1(1) = \frac{1}{1+e^{-4.7998032}} = \frac{1}{1+0.0082313668^{-4.7998032}} = 0.9918358354$$

$$v_2(1) = n1w_{301} + n2w_{302} + n3w_{303} + \cdots w_{304} + n200w_{305}$$
$$= 0.207776 * 1.5 + 1 * 2.2 + 0.33199 * 1.0 + 0.238059 * 3.2$$
$$+ 0.708337 * -0.7 = 3.1096069$$

$$y_2(1) = \frac{1}{1+e^{-3.1096069}} = \frac{1}{1+0.0446184914^{-3.1096069}} = 0.9572872855$$

3. Nilai pada neuron di output layer:

$$v_5(1) = Z1(1)w_{401} + Z1(1)w_{501} + Z1(1)w_{601} + Z2(1)w_{402} + Z2(1)w_{502} + Z2(1)w_{602}$$
$$= 0.9513933295 * 4.3 + 0.8082050704 * 1.9 + 0.9900121158 * 2.1$$
$$+ 0.8640027515 * 2.5 + 0.8640027515 * 1.3 + 0.9918358354 * 3.3$$
$$= 14.2618751063$$

$$y_5(1) = \frac{1}{1+e^{-14.2618751063}} = 0.99999936$$

4. Hitung gradient error untuk neuron pada output layer :

Untuk data pertama, target/class nilai yang diharapkan adalah $\boldsymbol{y_d = 0}$ sedangkan keluaran yang didaptkan $\boldsymbol{y_5(1) = 0.99999936.}$ Maka hitung *error* pada iterasi pertama untuk data pertama pada neuron Y1,Y2 dan Y3 di output layer:

$$e_1^1(1) = y_d - y_1(1) = 0 - 0.99999936 = \mathbf{-0.99999936}$$

$$\delta_1(1) = y_1(1) * \big(1 - y_1(1)\big) * e_1^1(1)$$
$$= 0.99999936 * (1 - 0.99999936) * (-0.99999936)$$
$$= -0.00000064$$

$$e_2^1(1) = y_d - y_2(1) = 0 - 0.9999997949 = -0.9999997949$$

$$\delta_2(1) = y_2(1) * \big(1 - y_2(1)\big) * e_2^1(1)$$
$$= 0.99999936 * (1 - 0.99999936) * (-0.99999936)$$
$$= -0.00000064$$

$$e_3^1(1) = y_d - y_3(1) = 0 - 0.9999997949 = -0.9999997949$$

$$\delta_3(1) = y_3(1) * \big(1 - y_3(1)\big) * e_3^1(1)$$
$$= 0.99999936 * (1 - 0.99999936) * (-0.99999936)$$
$$= -0.00000064$$

5. Hitung koreki bobot untuk output layer,
   $\Delta w_{401}, \Delta w_{501}, \Delta w_{601}, \Delta w_{402}, \Delta w_{502}\ dan\ \Delta w_{602}$:

$$\Delta \boldsymbol{w_{jk}(p)} = \boldsymbol{\eta . y_j(p) * \delta_k(p)}$$

$$\Delta w_{401} = \eta * Z_1(1) * \delta_5(1) = 0.1 * 0.96 * (-0.00000064) = -0.0000000614$$

$$\Delta w_{501} = \eta * Z_1(1) * \delta_5(1) = 0.1 * 0.81 * (-0.00000064) = -0.0000000518$$

$$\Delta w_{601} = \eta * Z_1(1) * \delta_5(1) = 0.1 * 0.991 * (-0.00000064) = -0.0000000634$$

$$\Delta w_{402} = \eta * Z_2(1) * \delta_5(1) = 0.1 * 0.87 * (-0.00000064) = -0.0000000557$$

$$\Delta w_{502} = \eta * Z_2(1) * \delta_5(1) = 0.1 * 0.82 * (-0.00000064) = -0.0000000525$$

$$\Delta w_{602} = \eta * Z_2(1) * \delta_5(1) = 0.1 * 0.992 * (-0.00000064) = -0.0000000635$$

6. Hitung *gradient error* pada hidden layer $\delta_1(1)$ dan $\delta_2(1)$:

$$\delta_1(1) = y_1(1) * \big(1 - y_1(1)\big) * \sum_{k=1}^{1} \delta_k(1).w_{1k}(1)$$
$$= y_1(1) * \big(1 - y_1(1)\big) * \delta_1(1).w_{401}(1)$$
$$= 0.96 * (1 - 0.96) * (-0.00000064) * 4.3 = -0.0000001057$$

$$\delta_2(1) = y_2(1) * \big(1 - y_2(1)\big) * \sum_{k=1}^{1} \delta_k(1).w_{2k}(1)$$
$$= y_2(1) * \big(1 - y_2(1)\big) * \delta_2(1).w_{402}(1)$$
$$= 0.81 * (1 - 0.81) * (-0.00000064) * 2.5 = -0.0000002462$$

$$\delta_1(1) = y_1(1) * \big(1 - y_1(1)\big) * \sum_{k=1}^{1} \delta_k(1).w_{1k}(1)$$
$$= y_1(1) * \big(1 - y_1(1)\big) * \delta_1(1).w_{501}(1)$$
$$= 0.991 * (1 - 0.991) * (-0.00000064) * 1.9 = -0.0000000108$$

$$\delta_2(1) = y_2(1) * \left(1 - y_2(1)\right) * \sum_{k=1}^{1} \delta_k(1).w_{2k}(1)$$
$$= y_2(1) * \left(1 - y_2(1)\right) * \delta_2(1).w_{502}(1)$$
$$= 0.87 * (1 - 0.87) * (-0.00000064) * 1.3 = -0.0000000941$$

$$\delta_1(1) = y_1(1) * \left(1 - y_1(1)\right) * \sum_{k=1}^{1} \delta_k(1).w_{1k}(1)$$
$$= y_1(1) * \left(1 - y_1(1)\right) * \delta_1(1).w_{601}(1)$$
$$= 0.82 * (1 - 0.82) * (-0.00000064) * 2.1 = -0.0000001984$$

$$\delta_2(1) = y_2(1) * \left(1 - y_2(1)\right) * \sum_{k=1}^{1} \delta_k(1).w_{2k}(1)$$
$$= y_2(1) * \left(1 - y_2(1)\right) * \delta_2(1).w_{602}(1)$$
$$= 0.992 * (1 - 0.992) * (-0.00000064) * 3.3 = -0.0000000168$$

7. Hitung koreksi bobot untuk hidden layer

$\Delta w_{11}, \Delta w_{12}, \Delta w_{13}, \Delta w_{14}, \Delta w_{15}, \Delta w_{21}, \Delta w_{22}, \Delta w_{23}, \Delta w_{24}, \Delta w_{25}, \Delta w_{31}, \Delta w_{32}, \Delta w_{33}, \Delta w_{34},$

$\Delta w_{35}, \Delta w_{41}, \Delta w_{42}, \Delta w_{43}, \Delta w_{44}, \Delta w_{45}, \Delta w_{100}, \Delta w_{101}, \Delta w_{102}, \Delta w_{103}, \Delta w_{104}$

$\Delta w_{201}, \Delta w_{202}, \Delta w_{203}, \Delta w_{204}, \Delta w_{205}, \Delta w_{301}, \Delta w_{302}, \Delta w_{303}, \Delta w_{304}, \Delta w_{305}$:

$$\Delta \boldsymbol{w_{ij}}(\boldsymbol{p}) = \boldsymbol{\eta} * \boldsymbol{x_i}(\boldsymbol{p}) * \boldsymbol{\delta_j}(\boldsymbol{p})$$

$$\Delta w_{11} = \eta * x1 * \delta_1(1) = 0.1 * 0.207776 * (-0.00000011) = -0.0000000023$$

$$\Delta w_{12} = \eta * x2 * \delta_2(1) = 0.1 * 1 * (-0.00000011) = -0.000000011$$

$$\Delta w_{13} = \eta * x3 * \delta_3(1) = 0.1 * 0.33199 * (-0.00000011) = -0.0000000037$$

$$\Delta w_{14} = \eta * x4 * \delta_{\dots}(1) = 0.1 * 0.238059 * (-0.00000011) = -0.0000000026$$

$$\Delta w_{15} = \eta * x5 * \delta_{10}(1) = 0.1 * 0.708337 * (-0.00000011) = -0.0000000078$$

$$\Delta w_{21} = \eta * x1 * \delta_1(1) = 0.1 * 0.207776 * (-0.00000025) = -0.0000000052$$

$$\Delta w_{22} = \eta * x2 * \delta_2(1) = 0.1 * 1 * (-0.00000025) = -0.000000025$$

$$\Delta w_{23} = \eta * x3 * \delta_3(1) = 0.1 * 0.33199 * (-0.00000025) = -0.0000000083$$

$$\Delta w_{24} = \eta * x4 * \delta_{\dots}(1) = 0.1 * 0.238059 * (-0.00000025) = -0.000000006$$

$$\Delta w_{25} = \eta * x5 * \delta_{10}(1) = 0.1 * 0.708337 * (-0.00000025) = -0.0000000177$$

$$\Delta w_{31} = \eta * x1 * \delta_{1}(1) = 0.1 * 0.207776 * (-0.000000011) = -0.0000000002$$

$$\Delta w_{32} = \eta * x2 * \delta_{2}(1) = 0.1 * 1 * (-0.000000011) = -0.0000000011$$

$$\Delta w_{33} = \eta * x3 * \delta_{3}(1) = 0.1 * 0.33199 * (-0.000000011) = -0.0000000004$$

$$\Delta w_{34} = \eta * x4 * \delta_{\dots}(1) = 0.1 * 0.238059 * (-0.000000011) = -0.0000000003$$

$$\Delta w_{35} = \eta * x5 * \delta_{10}(1) = 0.1 * 0.708337 * (-0.000000011) = -0.0000000008$$

$$\Delta w_{41} = \eta * x1 * \delta_{1}(1) = 0.1 * 0.207776 * (-0.000000095) = -0.000000002$$

$$\Delta w_{42} = \eta * x2 * \delta_{2}(1) = 0.1 * 1 * (-0.000000095) = -0.0000000095$$

$$\Delta w_{43} = \eta * x3 * \delta_{3}(1) = 0.1 * 0.33199 * (-0.000000095) = -0.0000000032$$

$$\Delta w_{44} = \eta * x4 * \delta_{\dots}(1) = 0.1 * 0.238059 * (-0.000000095) = -0.0000000023$$

$$\Delta w_{45} = \eta * x5 * \delta_{10}(1) = 0.1 * 0.708337 * (-0.000000095) = -0.0000000067$$

$$\Delta w_{100} = \eta * x1 * \delta_{1}(1) = 0.1 * 0.207776 * (-0.000000199) = -0.0000000041$$

$$\Delta w_{101} = \eta * x2 * \delta_{2}(1) = 0.1 * 1 * (-0.000000199) = -0.0000000199$$

$$\Delta w_{102} = \eta * x3 * \delta_{3}(1) = 0.1 * 0.33199 * (-0.000000199) = -0.0000000066$$

$$\Delta w_{103} = \eta * x4 * \delta_{\dots}(1) = 0.1 * 0.238059 * (-0.000000199) = -0.0000000047$$

$$\Delta w_{104} = \eta * x5 * \delta_{10}(1) = 0.1 * 0.708337 * (-0.000000199) = -0.0000000141$$

$$\Delta w_{201} = \eta * n1 * \delta_{1}(1) = 0.1 * 0.207776 * (-0.000000017) = -0.0000000004$$

$$\Delta w_{202} = \eta * n2 * \delta_{2}(1) = 0.1 * 1 * (-0.000000017) = -0.0000000017$$

$$\Delta w_{203} = \eta * n3 * \delta_{3}(1) = 0.1 * 0.33199 * (-0.000000017) = -0.0000000006$$

$$\Delta w_{204} = \eta * n \dots * \delta_{\dots}(1) = 0.1 * 0.23805 * (-0.000000017) = -0.0000000004$$

$$\Delta w_{205} = \eta * n101 * \delta_{10}(1) = 0.1 * 0.708337 * (-0.000000017) = -0.0000000012$$

$$\Delta w_{301} = \eta * n1 * \delta_{1}(1) = 0.1 * 0.207776 * (-0.000000017) = -0.0000000004$$

$$\Delta w_{302} = \eta * n2 * \delta_{2}(1) = 0.1 * 1 * (-0.000000017) = -0.0000000017$$

$$\Delta w_{303} = \eta * n3 * \delta_{3}(1) = 0.1 * 0.33199 * (-0.000000017) = -0.0000000006$$

$$\Delta w_{304} = \eta * n \ldots * \delta_{\ldots}(1) = 0.1 * 0.23805 * (-0.000000017) = -0.0000000004$$

$$\Delta w_{305} = \eta * n101 * \delta_{10}(1) = 0.1 * 0.708337 * (-0.000000017) = -0.0000000012$$

8. Perbaharui bobot untuk neuron pada hidden layer
$W_{401,}W_{402,}W_{501,}W_{502,}W_{601,}$dan W_{602} :

$$W_{401}\ (2) = W_{401}\ (1) + \Delta W_{401} = 4{,}3 + (-0.0000000614) = 4.2999999386$$

$$W_{402}\ (2) = W_{402}\ (1) + \Delta W_{402} = 2{,}5 + (-0.0000000518) = 2.4999999482$$

$$W_{501}\ (2) = W_{501}\ (1) + \Delta W_{501} = 1{,}9 + (-0.0000000634) = 1.8999999366$$

$$W_{502}\ (2) = W_{502}\ (1) + \Delta W_{502} = 1{,}3 + (-0.0000000557) = 1.2999999443$$

$$W_{601}\ (2) = W_{601}\ (1) + \Delta W_{601} = 2{,}1 + (-0.0000000525) = 2.0999999475$$

$$W_{602}\ (2) = W_{602}\ (1) + \Delta W_{602} = 3{,}3 + (-0.0000000635) = 3.2999999365$$

9. Perbaharui bobot pada layer tersembunyi,

$\Delta w_{11}, \Delta w_{12}, \Delta w_{13}, \Delta w_{14}, \Delta w_{15}, \Delta w_{21}, \Delta w_{22}, \Delta w_{23}, \Delta w_{24}, \Delta w_{25}, \Delta w_{31}, \Delta w_{32}, \Delta w_{33}, \Delta w_{34},$

$\Delta w_{35}, \Delta w_{41}, \Delta w_{42}, \Delta w_{43}, \Delta w_{44}, \Delta w_{45}, \Delta w_{100}, \Delta w_{101}, \Delta w_{102}, \Delta w_{103}, \Delta w_{104}$

$\Delta w_{201}, \Delta w_{202}, \Delta w_{203}, \Delta w_{204}, \Delta w_{205}, \Delta w_{301}, \Delta w_{302}, \Delta w_{303}, \Delta w_{304}, \Delta w_{305}$:

$$W_{11}\ (2) = W_{11}\ (1) + \Delta W_{11} = 0{,}3 + (-0.0000000023) = 0.2999999977$$

$$W_{12}\ (2) = W_{12}\ (1) + \Delta W_{12} = 2{,}3 + (-0.000000011) = 2.299999989$$

$$W_{13}\ (2) = W_{13}\ (1) + \Delta W_{13} = 0{,}4 + (-0.0000000037) = 0.3999999963$$

$$W_{14}\ (2) = W_{14}\ (1) + \Delta W_{14} = 3{,}5 + (-0.0000000026) = 3.4999999974$$

$$W_{15}\ (2) = W_{15}\ (1) + \Delta W_{15} = -0{,}5 + (-0.0000000078) = -0.5000000078$$

$$W_{21}\ (2) = W_{21}\ (1) + \Delta W_{21} = 0{,}5 + (-0.0000000052) = 0.4999999948$$

$$W_{22}\ (2) = W_{22}\ (1) + \Delta W_{22} = 0{,}8 + (-0.000000025\ ) = 0.799999975$$

$$W_{23}\ (2) = W_{23}\ (1) + \Delta W_{23} = 3{,}1 + (-0.0000000083) = 3.0999999917$$

$$W_{24}\ (2) = W_{24}\ (1) + \Delta W_{24} = 0{,}6 + (-0.000000006) = 0.599999994$$

$$W_{25}\ (2) = W_{25}\ (1) + \Delta W_{25} = -0{,}9 + (-0.0000000177) = -0.9000000177$$

$$W_{31}\ (2) = W_{31}\ (1) + \Delta W_{31} = 0{,}2 + (-0.0000000002) = 0.1999999998$$

$$W_{32}\ (2) = W_{32}\ (1) + \Delta W_{32} = 3{,}5 + (-0.0000000011) = 3.4999999989$$

$W_{33}(2) = W_{35}(1) + \Delta W_{33} = 2{,}6 + (-0.0000000004) = 2.5999999996$

$W_{34}(2) = W_{34}(1) + \Delta W_{34} = 1{,}4 + (-0.0000000003) = 1.3999999997$

$W_{35}(2) = W_{35}(1) + \Delta W_{35} = -0.2 + (-0.0000000008) = -0.2000000008$

$W_{41}(2) = W_{41}(1) + \Delta W_{41} = 0{,}7 + (-0.000000002) = 0.699999998$

$W_{42}(2) = W_{42}(1) + \Delta W_{42} = 1{,}6 + (-0.0000000095) = 1.5999999905$

$W_{43}(2) = W_{43}(1) + \Delta W_{43} = 1{,}8 + (-0.0000000032) = 1.7999999968$

$W_{44}(2) = W_{44}(1) + \Delta W_{44} = 0{,}9 + (-0.0000000023) = 0.8999999977$

$W_{45}(2) = W_{45}(1) + \Delta W_{45} = -1{,}0 + (-0.0000000067) = -1.0000000067$

$W_{100}(2) = W_{100}(1) + \Delta W_{100} = 1{,}2 + (-0.0000000041) = 1.1999999959$

$W_{101}(2) = W_{101}(1) + \Delta W_{101} = 1{,}3 + (-0.0000000199) = 1.2999999801$

$W_{102}(2) = W_{102}(1) + \Delta W_{102} = 1{,}9 + (-0.0000000066) = 1.8999999934$

$W_{103}(2) = W_{103}(1) + \Delta W_{103} = 0{,}3 + (-0.0000000047) = 0.2999999953$

$W_{104}(2) = W_{104}(1) + \Delta W_{104} = -1{,}1 + (-0.0000000141) = -1.1000000141$

$W_{201}(2) = W_{201}(1) + \Delta W_{201} = 4{,}4 + (-0.0000000004) = 4.3999999996$

$W_{202}(2) = W_{202}(1) + \Delta W_{202} = 3{,}0 + (-0.0000000017) = 2.9999999983$

$W_{203}(2) = W_{203}(1) + \Delta W_{203} = 1{,}8 + (-0.0000000006) = 1.7999999994$

$W_{204}(2) = W_{204}(1) + \Delta W_{204} = 2{,}4 + (-0.0000000004) = 2.3999999996$

$W_{205}(2) = W_{205}(1) + \Delta W_{205} = -0{,}4 + (-0.0000000012) = -0.4000000012$

$W_{301}(2) = W_{301}(1) + \Delta W_{301} = 1{,}5 + (-0.0000000004) = 1.4999999996$

$W_{302}(2) = W_{302}(1) + \Delta W_{302} = 2{,}2 + (-0.0000000017) = 2.1999999983$

$W_{303}(2) = W_{303}(1) + \Delta W_{303} = 1{,}0 + (-0.0000000006) = 0.9999999994$

$W_{304}(2) = W_{304}(1) + \Delta W_{304} = 3{,}2 + (-0.0000000004) = 3.1999999996$

$W_{305}(2) = W_{305}(1) + \Delta W_{305} = -0{,}7 + (-0.0000000012) = -0.7000000004$

10. Naikkan 1 langkah iterasi (***p***), kembali ke langkah 2 dan ulangi proses tersebut sampai ***kriteria error*** tercapai.

11. Dengan demikian, didapatkan bobot akhir pada iterasi pertama untuk data pertama[2,3,4,1]:

$W_{11} = -1.47872$

$W_{12} = 3.81562$

$W_{13} = -1.76724$

$W_{14} = -0.960296$

$W_{15} = -0.56721$

$W_{21} = -5.39219$

$W_{22} = 0.0670655$

$W_{23} = -5.33011$

$W_{24} = 4.04024$

$W_{25} = -5.42608$

$W_{31} = 2.99867$

$W_{32} = 0.361013$

$W_{33} = 2.65423$

$W_{34} - 4.6354$

$W_{35} = 4.99886$

$W_{41} = 0.895106$

$W_{42} = -1.84864$

$W_{43} = 1.12665$

$W_{44} = 0.673782$

$W_{45} = 0.151246$

$W_{100} = -2.75146$

$W_{101} = 5.47324$

$W_{102} = -3.25793$

$W_{103} = -2.16648$

$W_{104} = -0.221218$

$W_{301} = 3.87644$

$W_{202} = 9.88265$

$W_{203} = -4.72784$

$W_{204} = -0.931554$

$W_{205} = 6.97019$

$W_{301} = -0.786651$

$W_{302} = -2.82728$

$W_{303} = -0.324694$

$W_{304} = -1.34442$

$W_{305} = -1.15601$

$W_{401} = 32.3501$

$W_{402} = 2.36732$

$W_{501} = 1.4719$

$W_{502} = -0.073842$

$W_{601} = -34.0749$

$W_{602} = -1.97423$

Error yang didapatkan adalah = = $-\mathbf{0.99999936}$. Selanjutnya proses di atas diulangi untuk data ke-dua [0.301565,0.713728,0.43954,0.445294,0.563357], data ke-tiga [0.271692,0.888239,0.406856,0.31055,0.632715], dan data ke-dua puluh

[0.482692,0.032086,0.615273,0.699397,0.294774] sehingga iterasi pertama selesai dilakukan. Pada setiap data yang diproses dalam satu iterasi, error keluaran disimpan untuk dihitung sebagai kriteria error, seperti SSE, MSE, dsb. Apabila hasil MSE < error yang didapatkan maka iterasi berhenti, sebaliknya dilakukan perambatan terus hingga batas perulangan/epoch.

Table 4.8. Tahap Testing ANN

Image Testing	X1	X2	X3	X4	X5	Y
	0.042592968	96.05512768	0.206380639	0.47567945	6.537250492	Karat Putih

Tahap testing menggunakan algoritma ANN:

Laju pembelajaran ($\boldsymbol{\eta}$) = 0.1;

Fungsi aktivasi yang digunakan adalah Sigmoid Biner;

Target *error* = 0.0001 dengan kriteria SSE;

Maksimum jumlah iterasi pelatihan = 1,000 kali.

1. Nilai pada neuron hidden layer N1,N2, N3, ….. , dan N10 :

(P) adalah iterasi ke-i.

$$v_1(1) = x1w_{11} + x2w_{12} + x3w_{13} + x4w_{14} + x5w_{15}$$
$$= 0.042592968 * -1.47872 + 96.05512768 * 3.81562$$
$$+ 0.206380639 * -1.76724 + 0.47567945 * -0.960296$$
$$+ 6.537250492 * -0.56721 = 361.9173721596$$

$$y_1(1) = \frac{1}{1 + e^{-361.9173721596}} = \frac{1}{1 + 6.62647183E - 158^{-361.9173721596}} = 1$$

$$v_2(1) = x1w_{21} + x2w_{22} + x3w_{23} + x4w_{24} + x5w_{25}$$
$$= 0.042592968 * -5.39219 + 96.05512768 * 0.0670655$$
$$+ 0.206380639 * -5.33011 + 0.47567945 * 4.04024$$
$$+ 6.537250492 * -5.42608 = -28.437500727$$

$$y_2(1) = \frac{1}{1 + e^{-28.437500727}} = \frac{1}{1 + 4.46426899E - 13^{-28.437500727}} = 1$$

$$v_3(1) = x1w_{31} + x2w_{32} + x3w_{33} + x4w_{34} + x5w_{35}$$
$$= 0.042592968 * 2.99867 + 96.05512768 * 0.361013$$
$$+ 0.206380639 * 2.65423 + 0.47567945 * -4.6354$$
$$+ 6.537250492 * 4.99886 = 65.8264892199$$

$$y_3(1) = \frac{1}{1 + e^{-65.8264892199}} = \frac{1}{1 + 2.58177843E - 29^{-65.8264892199}} = 1$$

$$v_4(1) = x1w_{41} + x2w_{42} + x3w_{43} + x4w_{44} + x5w_{45}$$
$$= 0.042592968 * 0.895106 + 96.05512768 * -1.84864$$
$$+ 0.206380639 * 1.12665 + 0.47567945 * 0.673782$$
$$+ 6.537250492 * 0.151246 = -175.9914700271$$

$$y_4(1) = \frac{1}{1 + e^{-175.9914700271}} = \frac{1}{1 + 3.69722350E - 77^{-175.9914700271}} = 1$$

$$v_5(1) = x1w_{51} + x2w_{52} + x3w_{53} + x4w_{54} + x5w_{55}$$
$$= 0.042592968 * -0.177384 + 96.05512768 * 6.12018$$
$$+ 0.206380639 * -0.00510256 + 0.47567945 * -0.0174629$$
$$+ 6.537250492 * 4.75193 = 618.9223129317$$

$$y_5(1) = \frac{1}{1 + e^{-618.9223129317}} = \frac{1}{1 + 1.60492509E - 269^{-618.9223129317}} = 1$$

$$v_6(1) = x1w_{61} + x2w_{62} + x3w_{63} + x4w_{64} + x5w_{65}$$
$$= 0.042592968 * 0.246418 + 96.05512768 * 0.986554$$
$$+ 0.206380639 * 0.166221 + 0.47567945 * 1.56605 + 6.537250492$$
$$* -1.49287 = 85.7940435641$$

$$y_6(1) = \frac{1}{1 + e^{-85.7940435641}} = \frac{1}{1 + 5.49693118E - 38^{-85.7940435641}} = 1$$

$$v_7(1) = x1w_{71} + x2w_{72} + x3w_{73} + x4w_{74} + x5w_{75}$$
$$= 0.042592968 * 0.904243 + 96.05512768 * -15.0979$$
$$+ 0.206380639 * 1.33557 + 0.47567945 * -2.67716$$
$$+ 6.537250492 * 4.76781 = -1.420.0216637448$$

$$y_7(1) = \frac{1}{1 + e^{-1.420.0216637448}} = \frac{1}{1 + 0^{-1.420.0216637448}} = 1$$

$$v_8(1) = x1w_{81} + x2w_{82} + x3w_{83} + x4w_{84} + x5w_{85}$$
$$= 0.042592968 * 1.7659 + 96.05512768 * -3.09374$$
$$+ 0.206380639 * 2.21602 + 0.47567945 * 2.07407 + 6.537250492$$
$$* -0.365538 = -298.0400531564$$

$$y_8(1) = \frac{1}{1 + e^{-298.0400531564}} = \frac{1}{1 + 3.65468144E - 130^{-298.0400531564}} = 1$$

$$v_9(1) = x1w_{91} + x2w_{92} + x3w_{93} + x4w_{94} + x5w_{95}$$
$$= 0.042592968 * 1.73531 + 96.05512768 * -5.10383$$
$$+ 0.206380639 * 3.69881 + 0.47567945 * 4.37074 + 6.537250492$$
$$* 2.46101 = -471.2444574998$$

$$y_9(1) = \frac{1}{1 + e^{-471.2444574998}} = \frac{1}{1 + 2.19347395E - 205^{-471.2444574998}} = 1$$

$$v_{100}(1) = x1w_{100} + x2w_{101} + x3w_{102} + x4w_{103} + x5w_{104}$$
$$= 0.042592968 * -2.75146 + 96.05512768 * 5.47324$$
$$+ 0.206380639 * -3.25793 + 0.47567945 * -2.16648$$
$$+ 6.537250492 * -0.221218 = 522.4664930062$$

$$y_{100}(1) = \frac{1}{1 + e^{-522.4664930062}} = \frac{1}{1 + 1.24647941E - 227^{-522.4664930062}} = 1$$

2. Nilai pada neuron di hiden layer Z1 dan Z2:

$$v_1(1) = n1w_{301} + n2w_{302} + n3w_{303} + \cdots w_{304} + n200w_{305}$$
$$= 1 * 3.87644 + 1 * 9.88265 + 1 * -4.72784 + 1 * -0.93155 + 1$$
$$* 4.02937 + 1 * 1.26219 + 1 * -10.3396 + 1 * -0.88886 + 1$$
$$* -0.87776 + 1 * 6.97019 = -8.25523$$

$$y_1(1) = \frac{1}{1 + e^{-13.659671}} = \frac{1}{1 + 0.0002598957^{-13.659671}} = 0.9997401718$$

$$v_2(1) = n1w_{401} + n2w_{402} + n3w_{403} + \cdots w_{404} + n200w_{405}$$
$$= 1 * -0.78665 + 1 * -2.82728 + 1 * -0.32469 + 1 * -1.34442 + 1$$
$$* -1.14877 + 1 * -1.24184 + 1 * -1.58189 + 1 * -2.23414 + 1$$
$$* 0.36386 + 1 * -1.15601 = -12.28183$$

$$y_2(1) = \frac{1}{1 + e^{-12.28183}} = \frac{1}{1 + 0.0000046352^{-12.28183}} = 0.9999953648$$

3. Nilai pada neuron di output layer:

$$v_1(1) = Z1(1)w_{501} + Z2(1)w_{502}$$
$$= 0.9997401718 * 32.3501 + 0.9999953648 * 2.36732$$
$$= 34.7090035587$$

$$y_1(1) = \frac{1}{1 + e^{-34.7090035587}} = 1$$

$$v_2(1) = Z1(1)w_{601} + Z2(1)w_{602}$$
$$= 0.9997401718 * 1.4719 + 0.9999953648 * -0.073842$$
$$= 1.4048740304$$

$$y_2(1) = \frac{1}{1 + e^{-1.4048740304}} = 0.8029561846$$

$$v_3(1) = Z1(1)w_{701} + Z2(1)w_{702}$$
$$= 0.9997401718 * -34.0749 + 0.9999953648 * -1.97423$$
$$= -36.0402672291$$

$$y_3(1) = \frac{1}{1 + e^{36.0402672291}} = 2.22797758E - 16$$

4. Nilai rata-rata output Y1,Y2 dan Y3:

$$\text{softmax}(z)_i = \frac{\exp(z_i)}{\sum_{l=1}^{k} \exp(z_l)}$$

$$v_1 + v_2 + v_3 = 1 + 0.8029561846 + 2.22797758E-16$$

$$= 1.8029561846$$

$$y_1 = \frac{1}{1.8029561846} = 0.5546446489$$

$$y_2 = \frac{0.8029561846}{1.8029561846} = 0.4453553511$$

$$y_2 = \frac{2.22797758E-16}{1.8029561846} = 1.23573584E-16$$

Di dapat nilai yang terbesar yaitu **0.5546446489** yang ada pada tahapan pertama yaitu Penyakit Karat Putih.

4.3 Evaluasi

4.3.1 Confusion Matrix

Pada penelitian ini menggunakan confusion Matrix sebagai metode. Dalam perhitungan akurasi pada penerapan deteksi penyakit tanaman daun bayam pengguna e-learning. Evaluasi kinerja deteksi penyakit di dasarkan dari jumlah pengujian objek salah dan benar yang di deteksi dapat dilihat dari table berikut :

Table 4.9. Hasil *Confusion Matrix*

		Prediksi		
		Karat Putih	Virus Keriting	Kekurangan Mangan
Aktual	Karat Putih	6	0	0
	Virus Keriting	0	6	1
	Kekurangan Mangan	0	1	6

Akurasi = $\frac{18}{20} * 100\% = 90\%$

Table 4.10. Pengukuran Kinerja Metode

Daun	TP	TN	FP	FN	Hasil (%)		
					Accuration	Precision	Recall
Karat Putih	6	14	0	0	100%	100%	100%
Virus Keriting	6	12	1	1	90%	85%	85%
Kekurangan Mangan	6	12	1	1	90%	85%	85%
Rata-rata					93.3%	90%	90%

Perhitungan Accuration Penyakit Karat putih, Virus keriting, Kekurangan mangan:

$\frac{TP+TN}{TP+TN+FP+FN} = \frac{6+14}{6+14+0+0} = \frac{20}{20} = 1$ = 1 x 100% = 100%

$\frac{TP+TN}{TP+TN+FP+FN} = \frac{6+12}{6+12+1+1} = \frac{18}{20} = 0.9$ = 0.9 x 100% = 90%

$\frac{TP+TN}{TP+TN+FP+FN} = \frac{6+12}{6+12+1+1} = \frac{18}{20} = 0.9$ = 0.9 x 100% = 90%

Perhitungan Precision Penyakit karat putih, Virus keriting, Kekurangan mangan:

$\frac{TP}{TP+FP} = \frac{6}{6+0} = \frac{6}{6} = 1$ = 1 x 100% = 100%

$\frac{TP}{TP+FP} = \frac{6}{6+1} = \frac{6}{7} = 0.85$ = 0.85 x 100% = 85%

$\frac{TP}{TP+FP} = \frac{6}{6+1} = \frac{6}{7} = 0.85$ = 0.85 x 100% = 85%

Perhitungan Recall Penyakit karat putih, Virus keriting, Kekurangan mangan:

$\frac{TP}{TP+FN} = \frac{6}{6+0} = \frac{6}{6} = 1$ = 1 x 100% = 100%

$\frac{TP}{TP+FN} = \frac{6}{6+1} = \frac{6}{7} = 0.85$ = 0.85 x 100% = 85%

$\frac{TP}{TP+FN} = \frac{6}{6+1} = \frac{6}{7} = 0.85$ = 0.85 x 100% = 85%

Keterangan :

- True positive (TP) = 18
- True negative (TN) = 38
- False positive (FP) = 2
- False negative (FN) = 2

BAB V

PEMBAHASAN

5.1 Pembahasan Pengumpulan Data

Pengumpulan data yang diambil dari data real dengan jumlah data sebanyak 205 *image,* beberapa image tidak bisa diolah karena factor *blur,* objek daun yang miring dan tidak bisa di deteksi daun, sehingga peneliti mengambil data sebanyak 101 dari 205 *image* yang dipilih berdasarkan hasil pengolahan sistem. 101 *image* yang dipakai dalam penelitian ini terdiri 81 data training dan 20 data testing yang merupakan data *image* RGB dengan format JPG. Masing-masing data terdiri dari tiga kategori penyakit, untuk data training terdiri dari 16 *image* daun penyakit karat putih, 13 *image* daun penyakit kekurangan mangan dan 14 *image* daun penyakit virus keriting. Sedangkan untuk data testing terdiri dari 6 *image* daun penyakit karat putih, 7 *image* daun penyakit kekurangan mangan dan 7 *image* daun penyakit virus keriting.

5.2 Pembahasan Model

Sebelum pengolahan data dilakukan terlebih dahulu peneliti melakukan pra pengolahan data yaitu mengkonversi data *training* dan data *testing* dari *image* RGB ke *Grayscale.* Hal ini dilakukan karena inputan dari metode *Gray Level Co-Occurrence Matrix* merupakan inputan *image grayscale.* Jadi, seluruh data *image* RGB harus dikonversi ke *grayscale.* Setelah didapatkan hasil konversi citra, langkah berikut yang dilakukan yaitu menormalisasi citra menggunakan *Histogram Equalization.*

Proses selanjutnya yaitu pengolahan data, seluruh dataset(*training* dan *testing)* akan diekstraksi ciri menggunakan metode *Gray Level Co-Occurrence Matrix.* Ciri yang akan dihasilkan dari ekstraksi ciri yaitu *contrast, homogenitas, entropy, dan energy.* Hasil ekstraksi ciri inilah yang akan menjadi data inputan pada proses *training* menggunakan metode *Artificial Neural Network,* inputanya yaitu x1 = *contras,* x2 = *Homogenitas,* x3 = *entropy,* x4 = *energy.*

Beberapa pelatihan data dilakukan untuk mengatur tingkat akurasi tertinggi. Pelatihan data yang pertama : menggunakan orientasi arah 0^0,jarak piksel = 1, fungsi aktivasi = *logistic*, hidden layer 1 = 32 *neuron*, hidden layer 2 = 24 *neuron* dan iterasi maksimum = 100x, mampu menghasilkan tingkat akurasi yang dihitung menggunakan *confusion matrix* adalah 90%.

Pada pelatihan data yang kedua menggunakan orientasi arah sudut 45^0, jarak piksel = 1, fungsi aktivasi = *logistic*, hidden layer 1 = 32 *neuron*, hidden layer 2 = 24 *neuron* dan iterasi maksimum 100x, mampu menghasilkan tingkat akurasi sebesar 70%. Pada pelatihan data yang ketiga : menggunakan orientasi arah sudut 0^0, jarak piksel = 2, fungsi aktivasi = *logistic*, hidden layer 1 = 32 *neuron*, hidden layer 2 = 24 *neuron* dan iterasi maksimum 100x, mampu menghasilkan tingkat akurasi yang dihitung menggunakan *confusion matrix* adalah sebesar 50%. Pada pelatihan data yang keempat : menggunakan orientasi arah sudut 45^0, jarak piksel = 2, fungsi aktivasi = *logistic*, hidden layer 1 = 32 *neuron*, hidden layer 2 = 24 *neuron* dan iterasi maksimum 100x, mampu menghasilkan tingkat akurasi sebesar 56,6%.

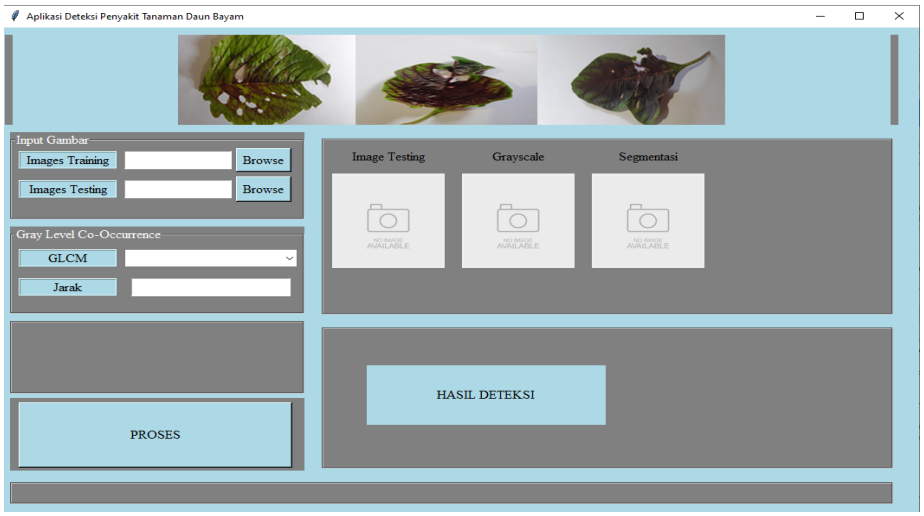
Sedangkan pada pelatihan selanjutnya, dengan menggunakan berbagai kombinasi nilai *Gray Level Co-Occurrence Matrix* dan *Artificial Neural Network*, menghasilkan akurasi 60%.

Untuk lebih jelasnya dapat dilihat pada table berikut :

Table 5.1**.** Pengujian Data

GLCM		**ANN**				**Akurasi**
Sudut	**Jarak**	**Aktivasi**	**Hidden Layer 1**	**Hidden Layer 2**	**Iterasi**	
0	1	*Logistic*	32	24	100x	90%
45	1	*Logistic*	32	24	100x	70%
90	2	*Logistic*	32	24	100x	50%
135	2	*Logistic*	32	24	100x	56,6%

5.3 Pembahasan Sistem



Gambar 5.1 : Tampilan Sistem

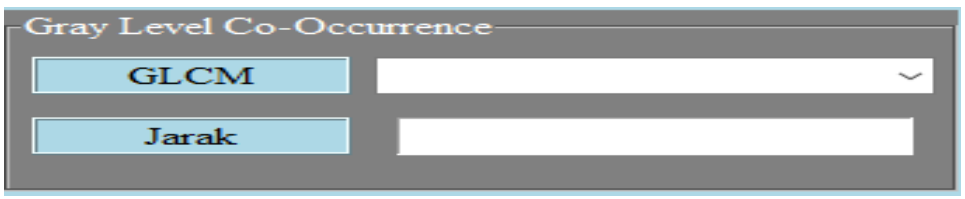
Sistem klasifikasi daun menggunakan satu *form* yang terdiri dari beberapa bagian :

1. Input Gambar

   Input gambar terdiri dari *image training* dan *image testing*.

Gambar 5.2 : Input Gambar

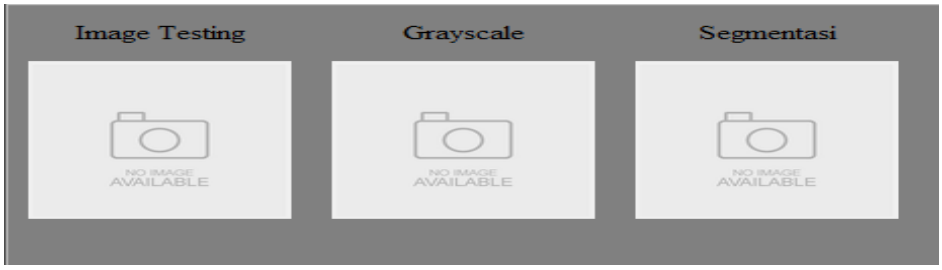
2. Input nilai *Gray Level Co-Occurrence Matrix* (GLCM), terdiri dari :
   - Orientasi sudut arah: 0^0, 45^0, 90^0, 135^0.
   - Jarak keterangan: 1, 2, 3 dan seterusnya.



Gambar 5.3 : Input Nilai

3. Prapengolahan

   Pada prapengolahan akan menampilkan output yang berupa hasil konversi *image* dari *RGB* ke *Grayscale,* hasil normalisasi citra menggunakan *Histogram Equalization,* deteksi daun menggunakan *Grayscale* dan *Segmentasi.*



Gambar 5.4 : Prapengolahan

4. Hasil Klasifikasi

   Hasil klasifikasi menampilkan kategori penyakit daun bayam (karat putih, kekurangan mangan, virus keriting) dan akurasi klasifikasi.

Gambar 5.5 : Hasil Klasifikasi

BAB V

PENUTUP

6.1 Kesimpulan

Berdasarkan hasil penelitian dan pembahasan yang telah diuraikan diatas maka dapat ditarik suatu kesimpulan sebagai berikut:

1. Hasil penggunaan metode *Gray Level Co-Occurrence Matrix* (GLCM) dalam mengekstraksi ciri daun menggunakan metode *Artificial Neural Network* mampu menghasilkan tingkat akurasi yang dihitung menggunakan *counfusion matrix* yaitu sebesar 90%, dan tingkat akurasi dari masing-masing penyakit yaitu karat putih dengan tingkat akurasi 100%, virus keriting hasil akurasi 90%, kekurangan mangan dengan akurasi 90%.
2. Hasil kinerja metode *Artificial Neural Network* dalam melakukan klasifikasi penyakit pada citra daun sudah cukup baik, Tahap testing menggunakan Algoritma ANN di dapatkan nilai sebesar **0.5546446489** dengan keterangan gambar yang ditesting yaitu penyakit karat putih, Apabila hasil MSE < error yang didapatkan maka iterasi berhenti, sebaliknya dilakukan perambatan terus hingga batas perulangan/epoch. hal ini dilihat dari proses klasifikasi menggunakan inputan hasil parameter dari fitur ekstraksi yang mampu mengklasifikasi penyakit sesuai kelas yang telah ditentukan.

6.2 Saran

Setelah melakukan penelitian, ada beberapa saran yang perlu diperhatikan untuk mencapai tujuan yang diharapkan, yaitu sebagai berikut :

1. Perlu diadakan penelitian lebih lanjut, dengan penggunaan metode lainnya terutama pada metode ekstraksi ciri sebagai perbandingan untuk mendapatkan hasil terbaik.
2. Untuk penelitian selanjutnya bisa menggunakan Algoritma lain yang tersedia sesuai dengan yang dimaksud.

DAFTAR PUSTAKA

[1] TRY SONY(G64130020), GISHELLA ERDYANING (G64130040), AMALIYA SUKMA RAGIL PRISTIYANTO (G64130044), MUHAMMAD RIZQI (G64130091), and DITA UTAMI PUTRI (G64130092), "Deteksi Penyakit Daun Menggunakan Artificial Neural Network (ANN)," 2010.

[2] ANISA RESTI, "PENENTUAN KADAR LOGAM TIMBAL (Pb) PADA DAUN BAYAM(Amaranthus spp.) MENGGUNAKAN DESTRUKSI BASAH SECARA SPEKTROSKOPI SERAPAN ATOM (SSA)," no. June, 2016.

[3] J. Biologi, V. No, I. Anastasia, M. Izatti, S. Widodo, and A. Suedy, "Pengaruh Pemberian Kombinasi Pupuk Organik Padat dan Organik Cair Terhadap Porositas Tanah dan Pertumbuhan Tanaman Bayam (Amarantus tricolor L.)," *J. Akad. Biol.*, vol. 3, no. 2, pp. 1–10, 2014.

[4] J. P. Informatika *et al.*, "Sistem pakar mendiagnosa hama dan penyakit pada tanaman bayam dengan metode naïve bayes," vol. 17, pp. 473–479, 2018.

[5] R. K. Dewi and R. V. H. Ginardi, "Identifikasi Penyakit pada Daun Tebu dengan Gray Level Co-Occurrence Matrix dan Color Moments," *J. Teknol. Inf. dan Ilmu Komput.*, vol. 1, no. 2, p. 70, 2014.

[6] F. S. Ni'mah, T. Sutojo, and D. R. I. M. Setiadi, "Identifikasi Tumbuhan Obat Herbal Berdasarkan Citra Daun Menggunakan Algoritma Gray Level Co-occurence Matrix dan K-Nearest Neighbor," *J. Teknol. dan Sist. Komput.*, vol. 6, no. 2, p. 51, 2018.

[7] M. K. Asmaul Husna Nasrullah, M. K. Amiruddin, M. K. Azminudin Aziz, M. E. Budy Santoso, S.Kom., and M. K. Apriyanto Alhamad, *PEDOMAN PENELITIAN ILMU KOMPUTER*. Gorontalo, 2018.

[8] Diana Laily Fithri, "DETEKSI PENYAKIT PADA DAUN TEMBAKAU DENGAN MENERAPKAN ALGORITMA ARTIFICIAL NEURAL NETWORK," vol. 3, no. 1, pp. 51–58, 2015.

[9] H. A. T. Zaki Imaduddin, "DETEKSI DAN KLASIFIKASI DAUN MENGGUNAKAN METODE ADABOOST DAN SVM," *J. content*, 2015.

[10] Mustika Mentari, Y. A. Sari, and R. K. Dewi, "Deteksi Kanker Kulit Melanoma dengan Linear Discriminant Analysis-fazzy k-Nearest Neigbhour Lp-Norm," 2016.

[11] Agnesia and Gina, "Implementasi Algoritma Backpropagation Jaringan Syaraf Tiruan untuk Mendeteksi Penyakit Tanaman Karet (Hevea brasiliensis)," 2015.

[12] Buah Salak Menggunakan Jaringan Syaraf Tiruan (Jst) dengan Metode F. R. Hartantri and A. Pujiyanta, "Deteksi Penyakit dan Serangan Hama Tanaman

Perceptron," 2014.

[13] "Landasan Teori."

[14] Husnul abdi, "Hama dan Penyakit Tanaman serta Cara Mengatasinya yang Perlu Diketahui," 2019.

[15] Zenzen Zainudhin, "Mengenal penyakit tanaman bayam dan pengendaliannya," 2016. [Online]. Available: https://www.agrotani.com/penyakit-tanaman-bayam/.

[16] R. P. NIM.10110448, "Pembangunan aplikasi deteksi dan tracking warna virtual drawing menggunakan algoritma color filtering," 2015.

[17] E. Miyamoto and T. M. Jr., "FAST CALCULATION OF HARALICK TEXTURE FEATURES Human Computer Interaction Institute Department of Electrical and Computer Engineering Carnegie Mellon University," *Hum. Comput. Interact. Inst. Dep. Electr. Comput. Eng. Carnegie Mellon Univ. Pittsburgh PA*, vol. 15213, pp. 1–6, 2011.

[18] S. in E. Science, "Technogenic Activity of Man and Local Sources of Environmental Pollution," 1981.

[19] Y. Fernando, "Klasifikasi Jenis Daging Berdasarkan Analisis Citra Tekstur Gray Level Co-Occurrence Matrices (Glcm) Dan Warna," *Semin. Nas. Sains dan Teknol. 2017*, no. Fakultas Teknik Universitas Muhammadiyah Jakarta, pp. 1–7, 2017.

[20] E. D. Handoyo, "Perancangan Mini Image Editor Versi 1.0 sebagai Aplikasi Penunjang Mata Kuliah Digital Image Processing," *J. Inform.*, vol. 2, no. 2, pp. 145–154, 2006.

[21] R. Santi, "Mengubah Citra Berwarna Menjadi Gray-Scale dan Citra Biner," *None*, vol. 16, no. 1, p. 243035, 2011.

[22] A. W. SANJAYA, "DETEKSI PENYAKIT KULIT MENGGUNAKAN ANALISIS FITUR WARNA DAN TEKSTUR DENGAN METODE COLOR MOMENT, GRAY LEVEL COOCCURENCE MATRIX, DAN JARINGAN SARAF TIRUAN BACKPROPAGATON," 2011.

[23] E. Alvansga, "Texture Recognition Using Glcm Method," 2019.

[24] M. K. Azminuddin Azis, *Manual Artificial Neural Network.* .

[25] Salamadian, "ARTIFICIAL INTELLIGENCE : Pengertian, Fungsi & Contoh Kecerdasan Buatan," 2019.

[26] Mahasiswa, "Pengujian Dengan Confusion Matrix," 2018. [Online]. Available: http://www.kuliahkomputer.com/2018/07/pengujian-dengan-confusion-matrix.html.

KODE PROGRAM

1. Pengubahan citra warna ke *grayscale* dan normalisasi citra dengan menggunakan *Histogram Equalization*

```
import cv2
import numpy as np
from skimage import exposure
class prosesGambar :
    def equalisasi_histogram(self, img):
        im = cv2.resize(img, (160, 160))
        gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
        hasil_equalisasi = exposure.equalize_adapthist(gray)
        return hasil_equalisasi, gray
def main():
    a = prosesGambar()
    img = cv2.imread('20200310_152402.jpg')
    hasil, gray = a.equalisasi_histogram(img)
    cv2.imshow('gambar asli', cv2.resize(img,(160, 160)))
    cv2.imshow('gambar grayscale', hasil)
    cv2.imshow('hasil equalisasi', hasil)
    cv2.waitKey(0)
    cv2.destroyA11Windows()
if __name__=='__main__':
    main()
```

2. Citra segmentasi

```
import cv2
import numpy as np
from skimage import feature
from scipy.stats import entropy
from imutils import paths
import os

class analisis_citra:
    def segmentasi(self, img) :
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        blur = cv2.GaussianBlur(gray, (5,5),0)
        ret, alpha = cv2.threshold(np.uint8(blur), 0, 255,
cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
        alpha [alpha==255] = 1
        rect = cv2.boundingRect(alpha)
        r = img[:,:,0] * alpha
        g = img[:,:,1] * alpha
        b = img[:,:,2] * alpha
        hasil = cv2.merge((r,g,b))
        x,y,w,h = rect
        result = hasil[y:y+h, x:x+w]
        return result
    def fitur_glcm(self, img):
        img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        matrix_glcm = feature.greycomatrix(img, [1], [0], symmetric=False,
normed=True)
        asm = feature.greycoprops(matrix_glcm, 'ASM') [0,0]
        contrast = feature.greycoprops(matrix_glcm, 'contrast')[0,0]
```

```
    energy = feature.greycoprops(matrix_glcm, 'energy')[0,0]
    homogeneity = feature.greycoprops(matrix_glcm, 'homogeneity')[0,0]
    ent = entropy(matrix_glcm.flatten())
    hasil_fitur = [asm, contrast, energy, homogeneity, ent]
    return hasil_fitur

def get_Img_Path(self, folderPath):
    imPath = []
    for path in paths.list_images(folderPath):
        imPath.append(path)
    return imPath

def get_label(self, folderPath):
    label = []
    imPath = self.get_Img_Path(folderPath)
    for item in imPath:
        label.append(os.path.split(os.path.dirname(item))[-1])
    return label

def training_data(self, folderPath):
    pathList = self.get_Img_Path(folderPath)
    images = []
    for imPath in pathList:
        img = cv2.imread(imPath)
        images.append(img)
    data_train = []
    for img in images:
        hasil_segmen = self.segmentasi(img)
```

```
            ekstraksi = self.fitur_glcm(hasil_segmen)
            data_train.append(ekstraksi)
        label_train = self.get_label(folderPath)
        return data_train, label_train

def main():
    a = analisis_citra()
    data_train, label = a.training_data('D:\data_bayam')
    print(data_train)
    print(label)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
if __name__ =='__main__':
    main()
```

3. Pembuatan Form Sistem

```
import tkinter as tk
import tkinter.filedialog as fdialog
import tkinter.ttk as ttk
from PIL import Image, ImageTk
import ekstraksiFitur_ANN as improses
import cv2

#from sklearn.preprocessing import OrdinalEncoder

class aplikasiku(tk.Frame):
    def __init__(self, parent):
        tk.Frame.__init__(self)
```

```
        self.parent = parent
        self.frames()
        self.form()

    def frames(self):
        #frame judul
        self.FrameAtas = tk.Frame(self.parent, width=1010, height=150,
background="gray", relief='raise')
        self.FrameAtas.grid(row=0, column=0, columnspan=3, padx=4, pady=10,
sticky="NSEW")
        self.FrameAtas.grid_propagate(0)

        #frame input gambar
        self.MainFrame1 = tk.LabelFrame(self.parent, height=120,
background="gray", text="Input Gambar", font="Times", fg="white")
        self.MainFrame1.grid(row=1, column=0, padx=10, sticky="NSEW")
        self.MainFrame1.grid_propagate(0)

        #glcm
        self.MainFrame2 = tk.LabelFrame(self.parent, height=120,
background="gray", text="Gray Level Co-Occurrence", font="Times",
fg="white")
        self.MainFrame2.grid(row=2, column=0, padx=10, pady=10,
sticky="NSEW")
        self.MainFrame2.grid_propagate(0)

        #Ann
        self.MainFrame3 = tk.LabelFrame(self.parent, height=100,
background="gray", font="Times", fg="white")
        self.MainFrame3.grid(row=3, column=0, padx=10, sticky="NSEW")
        self.MainFrame3.grid_propagate(0)
```

```
        #proses
        self.MainFrame4 = tk.Frame(self.parent, height=100, background="gray")
        self.MainFrame4.grid(row=5, column=0, padx=10, pady=6,
sticky="NSEW")
        self.MainFrame4.grid_propagate(0)

        #hasil
        self.MainFrame5 = tk.LabelFrame(self.parent, background="gray",
font="Times", fg="black")
        self.MainFrame5.grid(row=1, column=1, rowspan=2, columnspan=2,
padx=10, pady=9, sticky="NSEW")
        self.MainFrame5.grid_propagate(0)

        #hasil
        self.MainFrame6 = tk.LabelFrame(self.parent, background="gray",
font="Times", fg="gray")
        self.MainFrame6.grid(row=3, column=1, rowspan=3, columnspan=2,
padx=10, pady=9, sticky="NSEW")
        self.MainFrame6.grid_propagate(0)

        #kaki
        self.MainFrame7 = tk.LabelFrame(self.parent, height=30, width=1020,
background="gray")
        self.MainFrame7.grid(row=6, column=0, columnspan=3, padx=10, pady=10,
sticky="NSEW")
        self.MainFrame7.grid_propagate(0)

    def form(self):

        #frame
        #logo
```

```
        self.PathTestingImg ='C:/Users/ASUS/Documents/contoh
python/sampul/sampul.jpg'

        self.imT = Image.open(self.PathTestingImg)

        self.imT = ImageTk.PhotoImage(self.imT)

        self.labelx = tk.Label(self.FrameAtas, width=1010, height=120,
bg="lightblue", image=self.imT)

        self.labelx.pack(side="top")


        #frame 1

        #label data training

        self.label1 = tk.Label(self.MainFrame1, text="Images Training",
font="Times", relief="ridge", width=12, background="lightblue")

        self.label1.grid(row=3, column=0, sticky="NSEW", padx=8, pady=5)


        #textbox data training

        self.nilai = tk.StringVar(self.MainFrame1, value="")

        self.txtrain = tk.Entry(self.MainFrame1, textvariable = self.nilai, width=20)

        self.txtrain.grid(row=3, column=1, sticky="NSEW", pady=5)


        #button browse training

        self.path2 = tk.Button(self.MainFrame1, text="Browse", font="Times",
width=6, background="lightblue",
command=lambda:self.nilai.set(fdialog.askdirectory()))

        self.path2.grid(row=3, column=5, padx=5)


        #label data testing

        self.label2 = tk.Label(self.MainFrame1, text="Images Testing",
font="Times", relief="ridge", background="lightblue")

        self.label2.grid(row=4, column=0, sticky="NSEW", padx=8, pady=10)
```

```
        #textbox data testing
        self.nilai1 = tk.StringVar(self.MainFrame1, value="")
        self.txtest = tk.Entry(self.MainFrame1, textvariable = self.nilai1)
        self.txtest.grid(row=4, column=1, sticky="NSEW", pady=10)

        #button browse testing
        self.path1 = tk.Button(self.MainFrame1, text="Browse",
font="Times",background="lightblue",
command=lambda:self.nilai1.set(fdialog.askopenfilename()))
        self.path1.grid(row=4, column=5, sticky="NSEW", padx=5, pady=5)

        #frame 2
        #COMBOBOX GLCM
        arah = tk.IntVar(self.MainFrame2)
        self.labela = tk.Label(self.MainFrame2, text="GLCM", width=12,
font="Times", relief="ridge", background="lightblue")
        self.labela.grid(row=5, column=0, padx=8 , pady=10)

        #arah GLCM
        arah = tk.StringVar(self.MainFrame2, value="")
        self.cbarahglcm = ttk.Combobox(self.MainFrame2, textvariable = arah,
width=10)
        self.cbarahglcm['values']=(0, 45, 90, 135)
        self.cbarahglcm.current(0)
        self.cbarahglcm.grid(row=5, column=1, sticky="NSEW", pady=10)

        #jarak GLCM
        self.labelb = tk.Label(self.MainFrame2, text="Jarak", font="Times",
relief="ridge", width=12, background="lightblue")
        self.labelb.grid(row=6, column=0, sticky="NSEW", padx=8, pady=5)
```

```
        jarak = tk.StringVar(self.MainFrame2, value="")
        self.tbjarak = tk.Entry(self. MainFrame2, textvariable = jarak, width=30)
        self.tbjarak.grid(row=6, column=1,  sticky="NSEW", padx=8, pady=5 )

        #frame 4
        #button proses
        self.path2 = tk.Button(self.MainFrame4, width=34, height=4,
text="PROSES",
                               font="Times",background="lightblue",
command=self.PROSES)
        self.path2.grid(row=5, column=0, padx=10, pady=6, sticky="NSEW")

        #frame 5
        #output
        #label hasil image testing

#       images noimages
        self.label = tk.Label(self.MainFrame5, text="Image Testing", font="Times",
background="gray")
        self.label.grid(row=0, column=0, padx=30, pady=10)

        self.noimage1 ='C:/Users/ASUS/Documents/contoh python/noimages.jpg'
        self.noimageTest1 = Image.open(self.noimage1)
        self.noimageTest1 = self.noimageTest1.resize((126, 126),
Image.ANTIALIAS)
        self.noimageTest1 = ImageTk.PhotoImage(self.noimageTest1)
        self.labelTest1 = tk.Label(self.MainFrame5, image=self.noimageTest1)
        self.labelTest1.image_names = self.noimageTest1
        self.labelTest1.grid(row=1, column=0, padx=10, pady=1, sticky="NSEW")
```

```
        self.label = tk.Label(self.MainFrame5, text="Grayscale", font="Times",
background="gray")
        self.label.grid(row=0, column=1, padx=30, pady=10)


        self.noimage2 ='C:/Users/ASUS/Documents/contoh python/noimages.jpg'
        self.noimageTest2 = Image.open(self.noimage2)
        self.noimageTest2 = self.noimageTest2.resize((126, 126),
Image.ANTIALIAS)
        self.noimageTest2 = ImageTk.PhotoImage(self.noimageTest2)
        self.labelTest2 = tk.Label(self.MainFrame5, image=self.noimageTest2)
        self.labelTest2.image_names = self.noimageTest2
        self.labelTest2.grid(row=1, column=1, padx=10, pady=1, sticky="NSEW")


        self.label = tk.Label(self.MainFrame5, text="Segmentasi", font="Times",
background="gray")
        self.label.grid(row=0, column=2, padx=30, pady=10)


        self.noimage3 ='C:/Users/ASUS/Documents/contoh python/noimages.jpg'
        self.noimageTest3 = Image.open(self.noimage3)
        self.noimageTest3 = self.noimageTest3.resize((126, 126),
Image.ANTIALIAS)
        self.noimageTest3 = ImageTk.PhotoImage(self.noimageTest3)
        self.labelTest3 = tk.Label(self.MainFrame5, image=self.noimageTest3)
        self.labelTest3.image_names = self.noimageTest3
        self.labelTest3.grid(row=1, column=2, padx=10, pady=1, sticky="NSEW")


        #frame 6
        #label keterangan hasil output
        self.label12 = tk.Label(self.MainFrame6, text="HASIL DETEKSI",
width=60, height=4, font="Times", background="lightblue")
```

```
        self.label12.grid(row=2, column=0, sticky="NSEW", columnspan=5,
padx=50, pady=50)


        #frame 7

        #label kaki

        self.labele = tk.Label(self.MainFrame7, text="BY FITRIATY ISHANAN",
font="Times", fg="gray")

        self.labele.grid(row=2, column=2, sticky="NSEW", padx=350, pady=1)

        self.labele.config(background="gray")

        self.MainFrame7.grid_propagate(0)


    def PROSES(self):


        #get image training

        txttrain = self.txtrain.get()

        trainSet = self.txtest.get()

        GLCM = self.cbarahglcm.get()

        Jarak = self.tbjarak.get()

#         nilaiK = self.textboxAnn.get()


        a = improses.analisis_citra()

        hasil, gray, img_segmen = a.proses_rekognisi(txttrain, trainSet, GLCM,
Jarak)

        #print hasil

        self.label12.configure(text = hasil)


        imgTest = Image.open(self.txtest.get())

        imgTest = imgTest.resize((126, 126), Image.ANTIALIAS)

        imgTest = ImageTk.PhotoImage(imgTest)
```

```
        self.labelTest1.configure(image=imgTest)
        self.labelTest1.image_names = imgTest
#
#
        imgGray = Image.fromarray(gray)
        imgGray = imgGray.resize((126, 126), Image.ANTIALIAS)
        imgGray = ImageTk.PhotoImage(imgGray)
        self.labelTest2.configure(image=imgGray)
        self.labelTest2.image_names = imgGray
#
        imgsegmen = cv2.cvtColor(img_segmen, cv2.COLOR_BGR2RGB)
        imgsegmen = Image.fromarray(imgsegmen)
        imgsegmen = imgsegmen.resize((126,126), Image.ANTIALIAS)
        imgsegmen = ImageTk.PhotoImage(imgsegmen)
        self.labelTest3.configure(image=imgsegmen)
        self.labelTest3.image_names = imgsegmen

def main():
    root = tk.Tk()
    root.geometry('1060x670+130+10')
    root.configure(background='lightblue')
    app = aplikasiku(root)
    app.parent.title("Aplikasi Deteksi Penyakit Tanaman Daun Bayam")
    root.mainloop()
if __name__ =='__main__':
    main()
```

BIODATA MAHASISWA
UNIVERSITAS ICHSAN GORONTALO
TAHUN AJARAN 2019 / 2020

Nama	: Fitriaty Ishanan
NIM	: T3116085
Jenis Kelamin	: Perempuan
Tempat, Tgl Lahir	: Motongkad, 21 Maret 1993
Agama	: Islam
Alamat	: JL. Lumba-lumba, Kel. Ipilo
Golongan Darah	: O
Angkatan	: 2016
Suku Bangsa	: Indonesia
Program Studi	: Teknik Informatika
Fakultas	: Ilmu Komputer
Jenjang Pendidikan	: Strata 1 (S1)
IPK	: 3.79
Ukuran Toga	: M
No. Hp	: 0823 – 4684 – 0089
Pekerjaan	: Mahasiswa
Email	: rhaty2103@gmail.com
Judul Skripsi	: DETEKSI PENYAKIT TANAMAN DAUN BAYAM MENGGUNAKAN METODE GLCM DAN *ARTIFICIAL NEURAL NETWORK (ANN)*