IMPLEMENTASI *CONTENT BASED IMAGE RETRIEVAL* MENGGUNAKAN JARINGAN SYARAF TIRUAN *OPTICAL BACKPROPAGATION* UNTUK SISTEM ANOTASI CITRA OTOMATIS

SKRIPSI

oleh:
ANDRI SANTOSO
0810960032-96

PROGRAM STUDI ILMU KOMPUTER
JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS BRAWIJAYA
MALANG
2012

UNIVERSITAS BRAWIJAYA

IMPLEMENTASI *CONTENTBASED IMAGE RETRIEVAL* MENGGUNAKAN JARINGAN SYARAF TIRUAN *OPTICAL BACKPROPAGATION* UNTUK SISTEM ANOTASI CITRA OTOMATIS

SKRIPSI
Sebagai salah satu syarat untuk memperoleh gelar Sarjana Ilmu Komputer

oleh:
ANDRI SANTOSO
0810960032-96

PROGRAM STUDI ILMU KOMPUTER
JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS BRAWIJAYA
MALANG
2012


UNIVERSITAS BRAWIJAYA

LEMBARPENGESAHAN SKRIPSI

IMPLEMENTASI *CONTENT BASED IMAGE RETRIEVAL* MENGGUNAKAN JARINGAN SYARAF TIRUAN *OPTICAL BACKPROPAGATION* UNTUK SISTEM ANOTASI CITRA OTOMATIS

oleh:
ANDRI SANTOSO
0810960032-96

Setelah dipertahankan di depan Majelis Penguji
pada tanggal 16Juli 2012
dan dinyatakan memenuhi syarat untuk memperoleh gelar
Sarjana Komputer dalam bidang Ilmu Komputer

Pembimbing I,	Pembimbing II,
Drs. Muh Arif Rahman, M.Kom.	**Yusi Tyroni M, S.Kom., M.S.**
NIP 196604231991111001	**NIP 198002282006041001**

Mengetahui,
Ketua Jurusan Matematika
Fakultas MIPA Universitas Brawijaya

Dr. Abdul Rouf Alghofari, M.Sc.
NIP 196709071992031001

LEMBAR PERNYATAAN

Saya yang bertanda tangan di bawah ini:

Nama	**: Andri Santoso**
NIM	**: 0810960032-96**
Jurusan	**: Matematika**
Program Studi	**: Ilmu Komputer**
Penulis tugas akhir berjudul	**: Implementasi *Content Based Image Retrieval* Menggunakan Jaringan Syaraf Tiruan *Optical Backpropagation* Untuk Sistem Anotasi Citra Otomatis.**

Dengan ini menyatakan bahwa:

1. **isi dari tugas akhir yang saya buat adalah benar-benar karya sendiri dan tidak menjiplak karya orang lain, selain nama-nama yang termaktub di isi dan tertulis di daftar pustaka dalam tugas akhir ini,**
2. **apabila dikemudian hari ternyata tugas akhir yang saya tulis terbukti hasil jiplakan, maka saya akan bersedia menanggung segala resiko yang akan saya terima.**

Demikian pernyataan ini dibuat dengan segala kesadaran.

Malang, 16 Juli 2012
Yang menyatakan,

Andri Santoso
NIM 0810960032

IMPLEMENTASI *CONTENTBASED IMAGE RETRIEVAL* MENGGUNAKAN JARINGAN SYARAF TIRUAN *OPTICAL BACKPROPAGATION* UNTUK SISTEM ANOTASI CITRA OTOMATIS

ABSTRAK

Citra digital baru diciptakan dan dibagikan dengan laju yang sangat cepat. Hal ini membuat kebutuhan akan metode *information retrieval* (IR) yang efisien juga semakin meningkat. Pelabelan citra adalah salah satu pendekatan yang baik untuk mendapatkan IR dengan tingkat akurasi yang tinggi,tetapi proses pelabelan citra secara manual merupakan pekerjaan yang melelahkan dan membutuhkan sumber daya yang besar. Sistem anotasi citra otomatis mampu menangani masalah ini.

Penelitian ini menjelaskan tentang pemanfaatan *ContentBased Image Retrieval*(CBIR) untuk mengembangkan sistem anotasi citra otomatis. Pendekatan ini diharapkan memiliki hasil yang baik karena sistem anotasi citra otomatis memiliki kemampuan untuk melakukan pengindeksan citra secara otomatis. CBIR mampu mengekstraksi fitur semantik dari citra secara otomatis berdasarkan tampilan visual atau konten yang terkandung pada sebuah citra. CBIR digunakan untuk menyusun *classifier* untuk citra obyek nyata dan *Haar Wavelet* digunakan untuk membangkitkan nilai vektor fitur dari citra. *Classifier*dibentuk dengan menggunakan jaringan syaraf tiruan yang dilatih menggunakan metode pelatihan *optical backpropagation*.

Hasil pengujian pada citra obyek nyata menunjukkan bahwa sistem yang telah dikembangkan memiliki tingkat efektifitas dan efisiensi yang tinggi. Sistem mampu mencapai tingkat akurasi 100% dengan menggunakan data uji dari kelas apel dan pistol. Sistem menghasilkan nilai rata-rata *recall* sebesar 0.979, rata-rata *precission* sebesar 0.631, dan rata-rata *F-Measure* sebesar 0.752. Artinya adalah sistem mampu melakukan pelabelan citra dengan baik menggunakan beberapa data latih yang berbeda-beda.

.

Kata Kunci: *haar wavelet*, CBIR, jaringan syaraf tiruan, *optical backpropagation*, sistem anotasi citra otomatis

IMPLEMENTATION OF CONTENTBASED IMAGE RETRIEVAL USING OPTICAL BACKPROPAGATION NEURAL NETWORK FOR AUTOMATED IMAGE ANNOTATION

ABSTRACT

New digital images are created and shared at incredible rate and thus efficient information retrieval from image database has increased tremendously. Labeling image is great approach to get accurate information retrieval. However, manual image labeling is wearyand expensive task. Automatic image annotation allows to handle this problem. Its ability to give label for image automatically,on the ways to retrieve images based on content, makes this approach can overcome the shortcoming of manual image labeling.

This research describethe method for automatic imageslabeling using ContentBased Image Retrieval (CBIR). This is promising approach because of its ability to automatically index and retrieve images based on their semantic features and visual appearance.CBIR allows to extract image feature vector from an image automatically based on its visual content. CBIR was utilized to create an automatic image classifier for real object images and Haar Wavelet was used to generate feature vector from image. The classifier was created using neural network for learning and trained using optical backpropagation algorithm. This classifier is used to produce labelled images.

Experimental result on real object images show the high effectiveness and efficiency of the proposed system. Image annotation achieved an accuracy rate of 100% on the test collections from apple and gun classes.The classifier get average recall of 0.979, average precission of 0.631, and average F-Measure of 0.752. It means that the proposed system is able to maintain good results using various training data.

Keywords: haar wavelet, CBIR, neural network, optical backpropagation, automatic image annotation

KATA PENGANTAR

Alhamdulillah, dengan memanjatkan puji syukur kehadirat Allah SWT yang telah memberikan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan skripsi yang berjudul "Implementasi *Content Based Image Retrieval* Menggunakan Jaringan Syaraf Tiruan *Optical Backpropagation* Untuk Sistem Anotasi Citra Otomatis" dengan baik.

Penulis menyadari bahwa selama penyusunan skripsi ini tidak terlepas dari bantuan, dukungan, bimbingan, serta motivasi dari banyak pihak. Untuk itu, dengan ketulusan serta kerendahan hati penulis menyampaikan ucapan terima kasih yang sebesar-besarnya kepada:

1. Drs. Muhammad Arif Rahman, M.Kom., selaku pembimbing I atas segala bimbingan, nasehat, motivasi serta kesabaran yang telah diberikan selama penulisan skripsi ini,
2. Yusi Tyroni M., S.Kom., M.S., selaku pembimbing II yang telah meluangkan waktu dan membimbing dengan sabar serta memberikan pengarahan kepada penulis,
3. Dany Primanita K, S.T., selaku dosen pembimbing akademik yang telah banyak memberikan perhatian dan bimbingan selama penulis menempuh pendidikan di Program Studi Ilmu Komputer , Jurusan Matematika, Fakultas MIPA, Universitas Brawijaya,
4. Dr. Abdul Rouf Alghofari, M.Sc., selaku Ketua Jurusan Matematika, Fakultas Matematika dan Ilmu Pengetahuan Alam,Universitas Brawijaya,
5. Drs. Marji, M.T, selaku Ketua Program Studi Ilmu Komputer, Jurusan Matematika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Brawijaya,
6. segenap bapak dan ibu dosen yang telah mendidik dan mengajarkan ilmunya kepada penulis selama menempuh pendidikan di Program Studi Ilmu Komputer, Jurusan Matematika, Fakultas MIPA, Universitas Brawijaya,
7. segenap staf dan karyawan di Jurusan Matematika, Fakultas MIPA, Universitas Brawijaya yang telah banyak membantu penulis dalam pelaksanaan penyusunan skripsi ini,

8. kedua orang tua serta adik dan keluarga yang tercinta, terima kasih atas semua doa, kasih sayang dan perhatian yang tulus serta dukungan yang telah diberikan,
9. sahabat-sahabatku, teman seperjuangan ilkom2k8 yang telah banyak memberikan dukungan demi kelancaran pelaksanaan penyusunan skripsi ini,
10. semua pihak yang telah terlibat baik secara langsung maupun tidak langsung yang tidak dapat penulis sebutkan satu per satu.

Penulis menyadari bahwa dalam penulisan skripsi ini masih terdapat kekurangan. Untuk itu penulis mengharapkan kritik dan saran. Akhir kata, semoga skripsi ini dapat memberikan manfaat bagi semua pihak.

Malang, 16 Juli 2012

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL


UNIVERSITAS BRAWIJAYA

DAFTAR LAMPIRAN


UNIVERSITAS BRAWIJAYA

BAB I
PENDAHULUAN

1.1. Latar Belakang

Pencarian informasi digital menjadi bagian yang penting dalam kehidupan kita sehari-hari. Beberapa faktor yang mempengaruhinya adalah pertumbuhan informasi digital yang pesat dan kemudahan pencarian informasimelalui berbagai media elektronik seperti internetmaupun media penyimpanan lokal.Seiring dengan pertumbuhaninformasi tersebut, jumlah citra digital juga bertambah dengan cepat sehingga dibutuhkansuatumetode *image retrieval* yang efektif dan efisien yang mampu melakukan penelusuran dan pencarian citra padakoleksi citra yang besar.

Salah satumetode*image retrieval* yang telah dikembangkan untuk melakukan pencarian pada koleksi citra adalahmetode *Text-based Image Retrieval*. Pada metode ini, pencariancitra dilakukan berdasarkan pada informasi tekstual yang melekat pada sebuah citra. Pada beberapa format citra, informasi tekstual dapat disimpan pada *metadata* yang terkandung secara langsung pada sebuah citra. Salah satu contoh format citra yang dapat menyimpan *metadata* tersebut adalah format JPEG.

Metode *Text-based Image Retrieval*memiliki kelebihan yaitu bersifat *high-speed* dan *low-cost*, sehingga metode ini dapatmemperoleh informasi yang terkandung pada citra dengan nilai *recall* yang tinggi dan dengan komputasi yang ringandan waktu yang relatif cepat (Luo dkk., 2003). Sayangnyainformasi tekstual yang dapat diandalkan pada sebuah citra jarang tersedia, sehingga pencarian berbasis teks tidak dapat dilakukan pada citra-citra tersebut. Beberapa cara yang dapat dilakukan untuk mengatasi hal tersebut adalahdengan memberikan label untuk citra yang ada pada sebuah koleksi citra secara manual, tetapi proses tersebut menjadi tidak efisienkarena dilakukan secara manual sehingga diperlukan metode lain yangdapat memperbaiki metode ini (Yavlinsky, 2007).

Metode lain yang telah dikembangkan untuk *image retrieval* adalah *Content-based Image Retrieval (CBIR)*. CBIR merupakansalah satu metode dalam bidang *image retrieval* yang mendapatkan perhatian cukup besardari kalangan peneliti.

CBIR mengekstraksi fitur yang terkandung di dalam sebuah citra (seperti fitur warna, tekstur, bentuk dan struktur) untuk mendapatkan karakteristik daricitra tersebut. Salah satu penerapan dari CBIR adalah *Query by Examples (QBE)*yaitu pencarian citra menggunakan sebuah citra contoh. Citra-citra yang memiliki kesamaan didapatkan dari perbandingan tingkat kesamaan fitur (Yang, 2004).

Metode CBIR mampu "melihat" citra secara visual, sehingga metode ini dapat digunakan untuk mencari citra pada koleksi citra yang tidak memiliki informasi tekstual. Pada perkembangannya,metode CBIR menjadi tidak *user-friendly* karena setiap kali akan dilakukan pencarian, metode ini membutuhkan sebuah citra contoh yang digunakan sebagai *query*. Hal ini membuat pengguna menjadi repot dan tidak sesuai dengan konsep *high-level* yang digunakan manusia untuk melakukan pencarian (Chen dkk., 2010).

Dari kedua metode *image retrieval* yangada, pada dasarnya pencarian berbasis teks menyediakan hasil berupa kesamaan semantik sedangkan pencarian berbasis konten menyediakan hasil berupa kesamaan visual.Kedua metode tersebut memiliki sifat yang *independence* sehingga penggabungan dari kedua metode tersebut dapat meningkatkan performa dari sistem *image retrieval*dengan mengambil keuntungan yang dimiliki oleh masing-masing metode tersebut (Barrios dkk., 2009).

Untuk menggabungkan metode pencarian citra berbasis teks dan metode pencarian citra berbasis konten, maka salah satu penerapannya adalah dengan membuat sistem anotasi citra otomatis. Tujuan dari sistem ini adalah untuk memberikan label atau informasi secara otomatis pada sebuah citra berdasarkan konten dari citra tersebut, sehingga pencarian citra berbasis teks dapat dilakukan pada citra-citra yang tidak memiliki informasi tekstual(Tsai dkk., 2011).

Sistem anotasi citra telah berkembang sebagai sebuah alternatif dari sistem pencarian citra menggunakan citra contoh. Sistem ini menawarkan keuntungan bahwa konten dari sebuah citra lebih cocok dispesifikasikan menggunakan bahasa alami atau sebuah kata kunci yang merepresentasikan citra tersebut. Kemampuan seperti itu bisa memfasilitasi penggunauntuk melakukan pencarian

citra pada koleksi citra tidak berlabel dengan menggunakan bahasa alami(Yavlinsky dkk., 2005).

Untuk dapat memberikan label pada citra secara otomatis, maka komputer harus diberikan pengetahuan terlebih dahulu tentang pola-pola citra dari beberapa data contoh.

Proses pemberian pengetahuan pada komputer dapat dilakukan dengan menggunakan metode Jaringan Syaraf Tiruan (JST). Seiring dengan perkembangan teknik kecerdasan komputasional pada dunia komputer, algoritma JST telah diterapkan di berbagai bidang, salah satunya adalah bidang *image retrieval*. Beberapa penelitian pada bidang *image retrieval* juga telah menggunakan algoritma JST untuk melatih komputer agar mampu melakukan tugas *image retrieval*.

Salah satu keunggulan dari jaringan syaraf buatan adalah kemampuannya untuk belajar dari contoh sehingga penerapannya pada *image retrieval* diharapkan mampu untuk mempelajari pola dari citra-citra masukan.

Pada metode JST, komputer dilatih untuk mempelajari pola dengan menggunakan data latih yang sudah ditentukan. Hasil akhir dari proses pelatihan JST adalah sebuah pengetahuan yang dimiliki oleh komputer yang dapat digunakan untuk mengenali citra-citra baru.Dari kemampuan mengenali citra tersebut, komputer akan mampu mengklasifikasikan citra kedalam kelompok-kelompok yang sudah ditentukan. Kemudian setiap citra diberi labelsesuai dengan hasil pengklasifikasian tersebut.

Berdasarkan latar belakang yang telah dipaparkan, maka judul yang digunakan untuk penelitian ini adalah "**Implementasi *Content Based Image Retrieval* Menggunakan Jaringan Syaraf Tiruan *Optical Backpropagation* untuk Sistem Anotasi Citra Otomatis**"

1.2. Rumusan Masalah

Permasalahan yang akan dibahas dalam penelitian ini adalah sebagai berikut:

1. Bagaimana menerapkan*Content Based Image Retrieval* menggunakan jaringan syaraf tiruan *Optical Backpropagation*untuk mengembangkan sebuahsistem anotasi citra otomatis.
2. Bagaimana tingkat akurasi untuk beberapa data sampeldari sistem yang menerapkan *Content Based Image*

*Retrieval*menggunakan jaringan syaraf tiruan *Optical Backpropagation*untuksistem anotasi citra otomatis.

1.3. Batasan Masalah

Dari permasalahan yang telah dirumuskan, batasan permasalahan yang digunakan untuk merancang dan membuat sistem ini adalah sebagai berikut:

1. Citra yang digunakan pada penelitian ini adalah citra berformat JPG.
2. Citra yang digunakan pada penelitian ini adalah citra yang diambil dengan pencahayaan baik.
3. Citra yang digunakan pada penelitian ini adalah citra sebuah obyek benda, bukan citra pemandangan.
4. Transformasi *wavelet* yang digunakan pada proses ekstraksi fitur adalah *transformasi haar wavelet*.
5. Proses ektraksi fitur menggunakan citra berdimensi 128x128 dan menggunakan komponen *diagonal moment*. Jika citra masukan tidak berdimensi 128 x 128, maka akan dilakukan proses normalisasi terlebih dahulu terhadap citra tersebut.

1.4. Tujuan

Berdasarkan rumusan masalah yang telah dipaparkan, tujuan dari penelitian ini adalah sebagai berikut:

1. Menerapkan*Content Based Image Retrieval* menggunakan jaringan syaraf tiruan *Optical Backpropagation*untuk mengembangkan sebuah sistem anotasi citra otomatis.
2. Mengevaluasi tingkat akurasi kinerja sistem anotasi citra yang dibangun dengan menerapkan *Content Based Image Retrieval*menggunakan jaringan syaraf tiruan *Optical Backpropagation*.

1.5. Manfaat

Manfaat dari penelitian tentang pengimplementasian *Content Based Image Retrieval* menggunakan jaringan syaraf tiruan*Optical Backpropagation*adalah untuk mengembangkan sebuah sistem anotasi citra otomatis yang dapat digunakan untuk memberikan label secara otomatis pada citra berdasarkan konten yang terkandung pada citra tersebut,sehingga hal ini dapat membantu proses pencarian citra

ketika sebuah atau beberapa citra tidak memiliki informasi tekstual yang dapat menggambarkan citra tersebut.

1.6. Sistematika Penulisan

Pembahasan dalam penelitian ini terdiri dari beberapa bab, antara lain:

1. **BAB I PENDAHULUAN**
   Memuat mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat dan sistematika penulisan penelitian ini.
2. **BAB II DASAR TEORI**
   Berisi teori-teori yang digunakan sebagai dasar penelitian pada penulisan penelitian ini.
3. **BAB III METODOLOGI DAN PERANCANGAN**
   Menjabarkan metode-metode yang digunakan dalam penelitian dan berisi perancangan model sistem yang akan diimplementasikan.
4. **BAB IV IMPLEMENTASI DAN PEMBAHASAN**
   Menjelaskan implementasi perangkat lunak dan pembahasan hasil analisa penelitian yang telah dilakukan.
5. **BAB V PENUTUP**
   Berisi kesimpulan dari penelitian yang telah dilakukan dan saran-saran mengenai hasil penelitian yang perlu dikembangkan.


UNIVERSITAS BRAWIJAYA

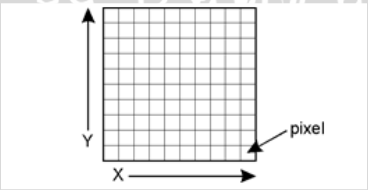
BAB II
TINJAUAN PUSTAKA

2.1. Konsep Dasar Citra Digital

Citra merupakan hasil representasi dari bentuk nyata obyek tiga dimensi menjadi gambar pada bidang dua dimensi yang didapatkan melalui beberapa proses pengolahan. Citra didapatkan dari kombinasi antara sumber energi (iluminasi) dan pantulan atau penyerapan energi dari sebuah obyek nyata yang sedang diubah menjadi citra(Gonzales & Woods, 2002).

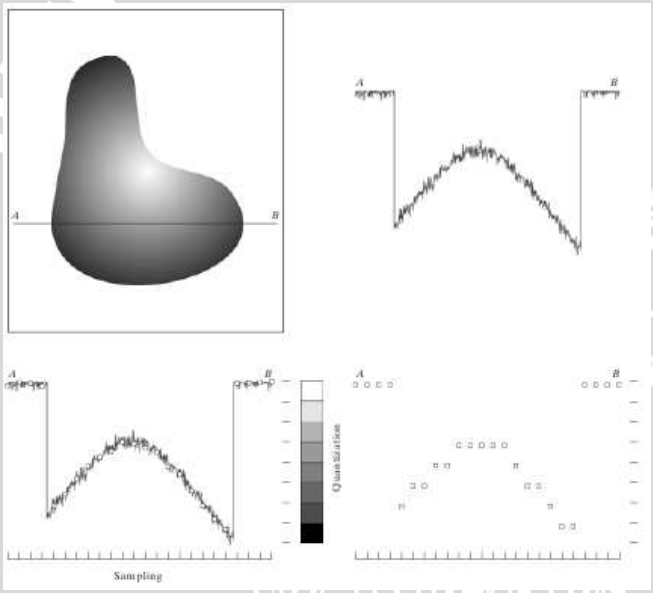
Proses pertama yang dilakukan pada pengolahan citra digital adalah akuisisi citra. Terdapat 2 elemen penting dari proses akuisisi citra ini. Elemen yang pertama adalah perangkat *sensor* yang sensitif dengan spektrum energi elektromagnetik yang menghasilkan keluaran berupa sinyal elektris sebanding dengan tingkat energi yang ada. Elemenyang kedua adalah *digitizer* yang berfungsi untuk mengubah keluaran sinyal elektris dari perangkat *sensor* fisik menjadi bentuk digital. Terdapat beberapa perangkat *sensor* yang dapat digunakan untuk proses akuisisi citra, antara lain: *Single Sensor, Sensor Strips, dan Sensor Arrays*(Gonzales & Woods, 2002).

Sebuah citra dapat didefinisikan sebagai fungsi dua dimensi *f(x,y)* dimana *x,y* adalah koordinat pada domain spasial,dan nilai dari *f(x,y)* adalah intensitas atau derajat keabuan dari komponen-komponen yang terbatas (*finite components*). Nilai *f(x,y)* ini juga merepresentasikan nilai *pixel* pada citra. *Pixel* adalah sebuah istilah yang paling umum digunakan untuk merujuk pada satuan terkecil pada citra. Seperti yang terlihat padaGambar 2.1, *pixel* merepresentasikan setiap titik yang menyusun sebuah citra. Setiap *pixel* mengandung 1 warna,dan kombinasi dari masing-masing *pixel* pada sebuah citra akan menyusun sebuah citra secara utuh (Gonzales & Woods, 2002).



Gambar 2.1*Pixel*sebagai Komponen Penyusun Citra

Ketika citra didapatkan dari proses fisik dengan menggunakan perangkat sensor yang ada, keluaran dari perangkat sensor tersebut adalah gelombang voltase kontinu yang amplitudo dan perilaku spasialnya dipengaruhi oleh keadaan lingkungan sekitar obyek. Untuk mendapatkan citra dalam bentuk digital,maka keluaran yang dihasilkan dari perangkat sensor tersebut harus diubah ke dalam bentuk digital dengan melakukan *sampling* dan kuantisasi. Proses *sampling* adalah proses digitalisasi dari nilai koordinat sebuah citra, sedangkan proses kuantisasi adalah proses digitalisasi dari nilai amplitudo sebuah citra. Proses *sampling* dan kuantisasi dapat dilihat padaGambar 2.2.



Gambar 2.2 Proses *Sampling* dan Kuantisasi

Konsep dasar dari proses *sampling* dan kuantisasi adalah mengubah citra yang pada awalnya masih dalam bentuk fungsi kontinu menjadi bentuk digital. Hal ini dapat dicapai dengan melakukan *sampling* dari fungsi kontinu untuk nilai koordinat dan amplitudo. Hasil dari proses sampling dan kuantisasi adalah matriks angka yang merepresentasikan nilai pixel dari sebuah citra digital seperti terlihat pada Gambar 2.3(Gonzales & Woods, 2002).

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \ldots. & f(0,M-1) \\ f(1,0) & \ldots. & \ldots. & f(1,M-1) \\ \ldots. & \ldots. & \ldots. & \ldots. \\ f(N-1,0) & f(N-1,1) & \ldots. & f(N-1,M-1) \end{bmatrix}$$

Gambar 2.3 Representasi Matriks untuk Citra Digital

2.2. Format Citra Digital

2.2.1. *Joint Photographic Experts Group* (JPEG)

Joint Photographic Experts Group (JPEG) adalah format citra yang paling sering digunakan untuk aplikasi *web*, *email*, fotografi, dan lain-lain. Format citra JPEG tidak mendukung transparansi ataupun animasi sehingga ukurannya menjadi relatif kecil dan bersifat *portable*. Format ini sangat popular karena memiliki tingkat kompresi tinggi dengan kualitas gambar yang baik.Semakin sering sebuah citra dilakukan kompresi, maka informasi yang ada pada citra tersebut juga akan semakin berkurang dan kualitasnya akan semakin menurun.

Format JPEG dapat melakukan kompresi gambar dengan efektif karena standard JPEG memanfaatkan ketidakmampuan sistem mata dan otak manusia untuk melihat sebuah detail yang sangat halus. Kompresi JPEG menghilangkan informasi warna yang tidak bisa dibedakan oleh mata manusia secara visual, dan kemudian melakukan kompresi dari *data stream* hasil penghilangan informasi tersebut untuk mendapatkan citra dalam ukuran yang lebih kecil, dengan seakan-akan tidak ada perubahan pada citra tersebut(Terras, 2008).

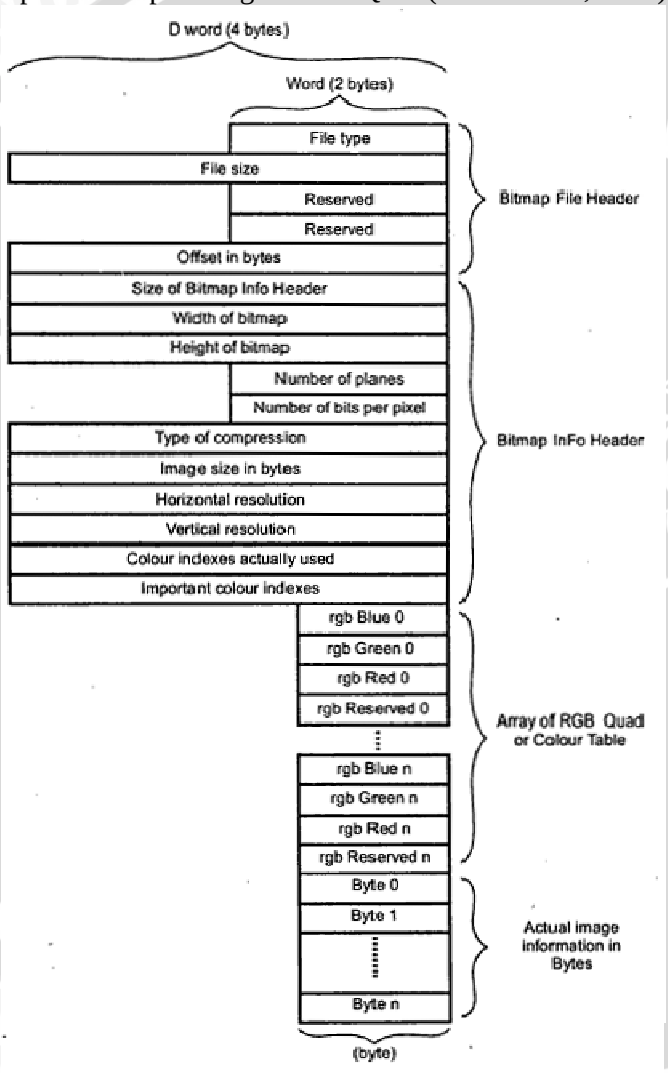
Format JPEG juga sering disebut dengan format citra "*lossy*". Maksudnya adalah informasi-informasi penting dari sebuah citra dapat hilang ketika berkas citra dikompres menjadi format JPEG.

2.2.2. *Microsoft Windows Bitmap* (BMP)

Format citra *Microsoft Windows Bitmap* atau lebih sering disebut format *bitmap* adalah format citra yang paling umum digunakan di lingkungan kerja*Windows*. Format citra ini berukuran relatif lebih besar dari format citra lainnya.

Citra berformat bitmap disimpan dalam format *Device-Independent Bitmap (DIB)* yang memungkinkan *Windows* untuk

menampilkan citra pada semua perangkat *display*. Ekstensi standar dari format *bitmap* adalah .BMP. Setiap berkas *bitmap* tersusun dari *Bitmap File Header, Bitmap Info* (terdiri dari *Bitmap Info Header* dan *RGB Quad*), dan *Byte*. Struktur file BMP dapatdilihat pada Gambar 2.4.Untuk proses penyimpanan warna, format citra *bitmap* menyimpan warna pada bagian *RGB Quad*(D.A. & A.P., 2007).



Gambar 2.4Struktur Berkas Citra BMP

2.3. *Image Metadata*

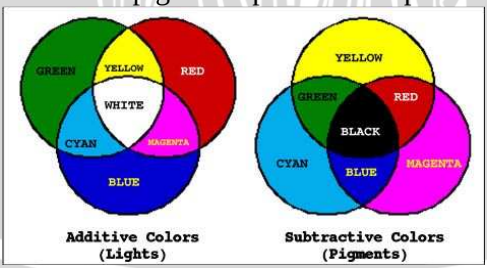
Selain data asli seperti warna, ukuran, dan data citra, beberapa format citra mampu menyimpan informasi tambahan seperti informasi tentang *copyrights*, *captions*, kata kunci, dan lain-lain. Salah satu format citra yang dapat menyimpan informasi tersebut adalah format citra JPEG. Informasi ini tersimpan pada *metadata* yang melekat secara langsung pada sebuah citra. *Metadata* dapat digunakan untuk menggambarkan citra atau mengklasifikasikan citra(Tratz, 2001).

Salah satu standar yang mendefinisikan informasi tersebut adalah standar IPTC. IPTC merupakan standar yang diterima secara luas oleh para *photographer* sehingga standar ini menjadi standar yang paling sering digunakan untuk memberikan informasi tambahan pada citra JPEG.Gambar 2.5menunjukkan informasi pada *metadata* yang terkandung pada sebuah citra.

Gambar 2.5 Informasipada *Metadata* Sebuah Citra

2.4. Model Warna pada Citra

Tujuan dari pemodelan warna adalah untuk membuat sebuah standar yang dapat diterima secara umum untuk merepresentasikan warna. Pada dasarnya, terdapat dua jenis model warna yaitu model warna cahaya dan warna pigmen seperti terlihat pada Gambar 2.6.



Gambar 2.6Jenis Warna

Sebagian besar model warna yang digunakan pada saat ini berorientasi pada perangkat keras dan aplikasi. Model warna yang berorientasi pada perangkat keras antara lain *RGB (red, green, blue)*yang digunakan untuk warna monitor, *CMY (cyan, magenta, yellow)* dan *CMYK (cyan, magenta, yellow, black)*yang digunakan untuk mencetak warna pada *printer*, dan *HSI (hue, saturation, intensity)* yang mendekati cara manusia menggambarkan dan menginterpretasikan sebuah warna. Model warna yang berorientasi pada aplikasi digunakan untuk membuat manipulasi warna, seperti yang digunakan untuk membuat warna pada gambar animasi(Gonzales & Woods, 2002).

2.4.1. Model Warna RGB

Format RGB adalah format warna yang menggabungkan warna *red*(R), *green*(G), dan *blue*(B) menjadi sebuah warna.

Teknologi RGB dipengaruhi oleh penelitian empiris yang menyebutkan bahwa berbagai macam warna dapat diperoleh dengan cara menggabungkan cahaya warna merah, hijau, dan biru dengan porsi yang berbeda-beda. Dengan teknik ini, dapat diperoleh sebuah warna dengan cara membentuk pengukuran dari tingkat kecerahan sebuah obyek pada setiap *pixel* menggunakan komponen merah, hijau, dan biru dari cahaya yang masuk. Tidak semua warna dapat diwakili oleh teknik RGB, tetapi teknik ini adalah salah satu teknik representasi warna yang sangat baik(Efford, 2000).

Pada model warna RGB, setiap *pixelf(x,y)* adalah sebuah vektor yang memiliki 3 komponen, yaitu nilai R,G, dan B. Setiap komponen pada RGB direpresentasikan dengan 8 bit, sehingga model RGB juga disebut dengan model warna 24-bit. Dan setiap komponen direpresentasikan dengan nilai antara 0-255, sehingga model warna RGB ini mampu menampilkan warna sebanyak 256x256x256=16.777.216 warna(Gonzales & Woods, 2002).

2.4.2. Model Warna HSI

Model warna HSI adalah model warna dimanametoderepresentasi warnanya paling mendekati pada cara manusia melihat dan merepresentasikan sebuah warna. Ketika manusia melihat sebuah obyek, sistem penglihatan manusia menggambarkan obyek dengan *hue*, *saturation*, dan *brightness*. *Hue*

adalah atribut warna yang merepresentasikan warna asli (kuning, oranye, dan merah), *saturation* memberikan ukuran cahaya ketika warna asli dipengaruhi oleh cahaya putih, dan *brightness* merepresentasikan intensitas pencahayaan dari sebuah obyek.

Kita dapat memperoleh nilai HSI dari nilai RGB pada sebuah citra dengan cara melakukan konversi RGB ke HSI. Untuk melakukan konversi tersebut, dapat digunakan persamaan-persamaan sebagai berikut (dengan asumsi bahwa nilai warna RGB sudah dinormalisasikan pada rentang 0-1):

a. Untuk mendapatkan nilai komponen *hue*, dapat digunakan persamaan 2.1.

$$H = \begin{cases} 0 & jika B \leq G \\ 360 - \theta & jika B > G \end{cases} \qquad (2.1)$$

dimana:

$$\theta = \cos^{-1}\left\{\frac{\frac{1}{2}[(R-G)+(R-B)]}{\sqrt{(R-G)^2+(R-B)(G-B)}}\right\}$$

H = nilai komponen *hue*pada model warna HSI
R = nilai warna merah pada citra
G = nilai warna hijau pada citra
B = nilai warna biru pada citra

Setelah didapatkan nilai komponen *hue*, maka nilai ini harus dinormalisasikan pada rentang 0-1 dengan cara membagi hasil perhitungan H dengan 360.

b. Untuk mendapatkan nilai komponen *saturation*, dapat digunakan persamaan 2.2.

$$S = 1 - \frac{3}{(R+G+B)} \qquad (2.2)$$

dimana:
S = nilai komponen *saturation* pada model warna HSI

c. Untuk mendapatkan nilai komponen *intensity*, dapat digunakan persamaan 2.3.

$$I = \frac{1}{3}(R + G + B) \qquad (2.3)$$

dimana:
I = nilai komponen *intensity* pada model warna HSI

(Gonzales & Woods, 2002)

2.5. *Text-based Image Retrieval*

Text-based Image Retrieval adalah salah satu metode untuk melakukan pencarian citra. Metode ini melakukan pencarian berdasarkan pada informasi tekstual yang melekat pada sebuah citra.Metode ini mampu mendekati konsep yang digunakan manusia untuk melakukan pencarian. Hal ini dikarenakan metode ini menggunakan kata kunci yang dimengerti manusia untuk melakukan pencarian.Jika dibandingkan dengan pencarian citra menggunakan citra, maka teknik ini lebih mendekati konsep *high-level* yang digunakan manusia untuk melakukan pencarian(Chen dkk., 2010).

Salah satu implementasi dari metode pencarian berbasis teks adalah mesin pencari di internet. Alih-alih memberikan informasi secara manual pada citra yang tersimpan pada *image database*, mesin pencari secara otomatis mengekstraksi informasi sebuah citra dari *metadata*(seperti deskripsi citra atau informasi-informasi lain) yang terdapat pada halaman *web* tempat citra tersebut berada, kemudian menyimpan informasi tersebut pada *database* yang terstruktur. Tetapi teknik ini masih memiliki kelemahan yaitu dapat dengan mudah melakukan kesalahan ketika mengasosiasikan citra dengan informasi yang ada pada halaman webtersebut (Yavlinsky, 2007).

2.6. *Content Based Image Retrieval*

Content Based Image Retrieval(CBIR) adalah sebuah teknik yang digunakan untuk mendapatkan informasi dari sebuah citra secara otomatis (tanpa proses manual seperti pemberian kata kunci pada citra) pada sebuah *database* citra. Dengan menggunakan metode CBIR, informasi dari citra didapatkan dengan cara mengekstrasi fitur sehingga metode CBIR akan terlihat berjalan

secara alami. Tujuan dari CBIR adalah untuk mendapatkan hasil terbaik untuk *image retrieval* sehingga CBIR diharapkan menyerupai konsep manusia dalam mengenali sebuah citra.

CBIR berbeda dengan penerimaan informasi klasik dikarenakan *database* citra bersifat tidak terstruktur. Citra digital hanya terdiri dari sekumpulan intensitas *pixel-pixel* yang membentuk sebuah *array* tanpa memiliki sebuah arti(Eakins & Graham, 1999).

D*atabase* citra sangat berbeda dengan database teks dimana materi penyusun *database* teks (kalimat yang tersimpan dalam format ASCII) sudah disusun secara logikal oleh penulis, sedangkan pada citra masih berupa data mentah yang tidak ada artinya. Salah satu hal terpenting pada pemrosesan citra adalah ekstraksi fitur dari data mentah citra, sehingga hasil ekstraksi tersebut dapat diproses untuk mendapatkan informasi dari citra tersebut. Beberapa fitur yang sering digunakan untuk CBIR antara lain warna, bentuk, dan tekstur(Santini & Jain, 1997).

Pada CBIR, citra akan diekstraksi secara otomatis berdasarkan konten visual dari citra tersebut (seperti warna, tekstur, atau bentuk). Untuk melakukan ekstraksi tersebut maka digunakan transformasi *wavelet*. Transformasi *wavelet* ini terbukti efektif untuk mengekstraksi konten visual dari gambar dan merepresentasikannya dalam bentuk fitur (Latha dkk., 2007).

2.6.1. *Color Retrieval*

Penerimaan informasi berdasarkan warna pada sebuah citra dilakukan dengan cara menghitung *histogram* yang menunjukkan proporsi warna dari setiap *pixel* pada citra. Kemudian informasi *histogram* tersebut disimpan pada sebuah *database*. Ketika dilakukan pencarian citra, *histogram* yang sudah tersimpan di dalam *database* digunakan untuk dicocokan dengan *histogram* citra yang digunakan sebagai *query*. Citra pada *database* yang memiliki *histogram* paling mendekati dengan *histogram* citra *query* akan ditampilkan sebagai hasil dari pencarian citra. Metode pencocokan yang dapat digunakan adalah metode *histogram intersection*(Eakins & Graham, 1999).

Metode *histogram* membutuhkan perhitungan yang sederhana dan tidak dipengaruhi oleh variasi kecil pada sebuah citra. Tetapi, metode *histogram* tidak memperhatikan distribusi warna spasial

sehingga metode ini tidak cocok untuk citra yang memiliki banyak variasi warna(Park dkk., 2004).

2.6.2. *Shape Retrieval*

Terdapat dua tipe utama dari fitur bentuk yang sering digunakan untuk ekstraksi fitur. Yang pertama adalah fitur global seperti *aspect ratio*, *circularity*, dan *moment invariants* dan yang kedua adalah fitur lokal seperti himpunan dari *consecutive boundary segments*(Eakins & Graham, 1999).

Fitur *shape* adalah fitur yang berbasis pada informasi kontur dari sebuah citra, dan umumnya diekstraksi menggunakan pendeteksian tepi dan *correlogram*. Pendeteksian tepi menghasilkan keluaran yang lebih efektif ketika digunakan untuk citra dengan informasi kontur yang jelas(Park dkk., 2004).

2.6.3. *Texture Retrieval*

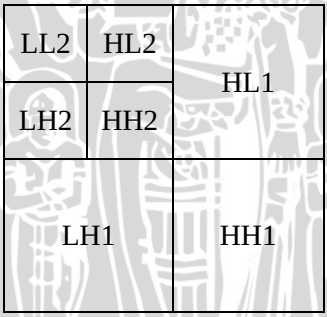
Tekstur adalah fitur yang baik dan berguna untuk diterapkan pada beberapa bidang. Beberapa metode yang dapat digunakan untuk mengekstrasi fitur dari tekstur sebuah obyek antara lain *Filter Gabor*, Metode Statistika, dan Transformasi *wavelet*. Metode-metode tersebut dapat digunakan untuk mengekstrasi informasi dari pola-pola yang berulang dari sebuah citra yang tidak dapat ditentukan dengan analisis fitur warna ataupun fitur bentuk(Park dkk., 2004)

2.7. Transformasi *Haar Wavelet*

Pada dasarnya, terdapat 2 macam transformasi yang dapat dilakukan pada citra, yaitu transformasi *piksel* dan transformasi ruang/domain. Pada transformasi *pixel*, proses transformasi dilakukan pada domain yang sama, artinya adalah citra asli dan citra hasil transformasi berada pada domain yang sama yaitu pada domain spasial. Salah satu contoh transformasi *pixel* ini adalah proses rotasi, translasi, dan *scaling* pada citra. Adapun pada transformasi ruang, proses transformasi dilakukan dengan mengubah citra dari suatu domain ke domain yang lainnya, sehingga citra asli dan citra hasil transformasi berada pada domain yang berbeda. Salah satu contoh dari transformasi ruang adalah proses transformasi *wavelet*dimana proses transformasi ini mengubah citra yang berada pada domain spasial menjadi citra pada domain frekuensi. (Suhendra, 2004)

Transformasi *wavelet* adalah salah satu bagian dari *multiresolution processing*yang terdapat pada bidang pengolahan citra digital. Sesuai dengan namanya, *multiresolution processing* berkonsentrasi pada analisis sinyal (atau citra) pada lebih dari saturesolusi. Pendekatan ini berusaha melakukan analisis pada banyak resolusi dengan alasan bahwa fitur dari citra yang tidak terdeteksi pada suatu resolusi diharapkan dapat dideteksi pada resolusi yang lainnya. Hasil dari proses transformasi *wavelet* ini memiliki dimensi yang sama dengan citra asli. (Gonzales & Woods, 2002)

Transformasi wavelet telah digunakan sebagai alat bantu untuk melakukan ekstraksi ciri dari sebuah citra. Pada transformasi *wavelet*, citra asli dibagi menjadi 4 *subband*. Pada keempat *subband* ini terdapat informasi yang sensitif yang secara garis besar dibagi menjadi frekuensi tinggi dan frekuensi rendah. Dari keempat *subband* tersebut dapat dilakukan analisis lebih lanjut untuk mendapatkan informasi ciri yang terkandung pada citra. Adapun pembagian keempat *subband* ini dapat dilihat padaGambar 2.7 yang menggambarkan hasil pembagian *subband* pada transformasi *wavelet* level 2 (Park dkk., 2004).



Gambar 2.7 Pembagian *Subband*pada Transformasi *Wavelet*

Terdapat beberapa jenis transformasi *wavelet*yang sering digunakan pada pengolahan sinyal antara lain *haar wavelet, symlet wavelet, deubechies wavelet, coifflet wavelet*, dan lain-lain (Ramadijanti, 2006).

Transformasi *haar wavelet* adalah salah satu jenis transformasi *wavelet* yang paling lama dikenal jika dibandingkan dengan beberapa

jenis transformasi *wavelet* yang ada. Fungsi *haar* ini sendiri sudah digunakan sejak 1910 oleh matematikawan Hungaria bernama Alfred Haar. Metode transformasi *wavelet* ini melakukan proses transformasi vektor dengan cara melakukan *averaging*(untuk selanjutnya disebut dengan proses perataan) dan *differencing*(untuk selanjutnya disebut dengan proses penskalaan) pada vektor tersebut, dimana hasil dari proses perataan disebut dengan *approximation coefficients* dan hasil dari proses penskalaan disebut dengan *detail coefficients*. Misalkan terdapat sebuah vektoryang akan ditransformasikan. Vektor ini terdiri dari nilai $a_i$dimana *a* adalah komponen dari sebuah vector dan *i = 1 ...n*.Jika dilakukan transformasi maka setengah dari jumlah data pada vektor tersebut (*n/2*) adalah bagian *approximation coefficient*dan setengah bagian yang lainnya merupakan bagian *detail coefficient*. Nilai *approximation coefficient* ini disimpan pada bagian *averages* dan nilai *detail coefficient* disimpan pada bagian *wavelets coefficient* seperti yang terlihat padaGambar 2.8(Latha dkk., 2007).

Gambar 2.8 Transformasi *HaarWavelet*

Untuk lebih memahami tentang cara kerja dari transformasi *wavelet*, maka akan dijabarkan contoh proses transformasi *wavelet* 1 dimensi. Misalkan diberikan sebuah vektor dengan nilai sebagai berikut:

[9 7 3 5]

Untuk melakukan transformasi *wavelet*, langkah pertama adalah melakukan perataanuntuk *pixel* secara bersama-sama sehingga didapatkan vektor hasil perataan dengan resolusi yang lebih rendah. Adapun vektor hasil perataan adalah sebagai berikut:

[8 4]

Dari vektor hasil perataan tersebut terlihat bahwa beberapa informasi dari vektor asli telah hilang pada saat proses perataan dilakukan. Untuk mengembalikan vektor tersebut menjadi vektor awal, maka dibutuhkan sebuah *detail coefficient* yang berfungsi

untuk menyimpan informasi yang hilang. Pada contoh perhitungan ini, nilai dari *detail coefficient* yang pertama adalah 1, karena nilai hasil perataan yang didapatkan adalah bernilai 1 lebih kecil dari 9, dan 1 lebih besar dari 7. Hal yang sama juga dilakukan untuk nilai *detail coefficient* yang kedua, nilai *detail coefficient* yang kedua adalah -1, karena 4 + (-1) = 3 dan 4 - (-1) = 5. Proses ini disebut dengan proses dekomposisi. Jika proses dekomposisi ini dilakukan secara berulang dan *detail coefficient*disimpan untuk setiap proses dekomposisi, maka didapatkan hasil seperti yang terlihat pada Tabel 2.1.

Tabel 2.1Nilai Dekomposisi Transformasi *Haar Wavelet* 1D

Resolusi	**Perataan**	***Detail coefficient***
4	$[9 \quad 7 \quad 3 \quad 5]$	-
2	$[8 \quad 4]$	$[1 \quad -1]$
1	$[6]$	$[2]$

Jika nilai akhir dari proses perataan dan semua *detail coefficient* yang tersimpan dituliskan menjadi suatu nilai tunggal yang merepresentasikan proses transformasi *wavelet* secara keseluruhan, maka hasil dari proses transformasi *wavelet* untuk vektor awal pada contoh perhitungan ini dapat dinyatakan sebagai berikut:

$$[6 \quad 2 \quad 1 \quad -1]$$

Proses perhitungan transformasi *wavelet*yang dilakukan dengan cara melakukan proses perataan dan penskalaan secara berulang ini disebut dengan*filter bank*(Stollnitz dkk., 1995).

Secara umum, rumus untuk menghitung nilai perataan dan penskalaan dari transformasi *wavelet* pada sebuah citra *P* dengan ukuran *mxn*dapat dinyatakan pada persamaan 2.4, persamaan 2.5, persamaan 2.6, dan persamaan 2.7.

$$H_0: f_{m,n} = \frac{x_{2m,2n} + x_{2m+1,2n} + x_{2m,2n+1} + x_{2m+1,2n+1}}{4} \quad (2.4)$$

$$H_1: f_{m,n} = (x_{2m,2n} + x_{2m+1,2n}) - (x_{2m,2n+1} + x_{2m+1,2n+1}) \quad (2.5)$$

$$H_2: f_{m,n} = (x_{2m,2n} + x_{2m,2n+1}) - (x_{2m+1,2n} + x_{2m+1,2n+1}) \quad (2.6)$$

$$H_3: f_{m,n} = (x_{2m+1,2n} + x_{2m,2n+1}) - (x_{2m,2n} + x_{2m+1,2n+1}) \quad (2.7)$$

dimana :
H_0 = nilai hasil perataan
H_1 = nilai hasil penskalaan secara vertikal
H_2 = nilai hasil penskalaan secara horizontal
H_3 = nilai hasil penskalaan secara diagonal

Hasil perhitunganH_0adalah hasil untuk komponen LL, H_1adalah hasil untuk komponen LH, H_2adalah hasil untuk komponen HL, dan H_3adalah hasil untuk komponen HH sehingga transformasi ini akan menghasilkan empat *subband* LL, LH, HL dan HH. Sebelum nilai hasil perhitungan H_1, H_2, dan H_3tersebut dimasukkan ke *detail coefficient*, nilai-nilai tersebut dijadikan nilai absolut terlebih dahulu. Hal ini dikarenakan nilai komponen warna pada citra tidak memiliki nilai negatif (Soto & Derado, 2004).

2.8. *Sliding Window-based Feature Extraction*

Feature extraction adalah sebuah metode yang digunakan untuk mendapatkan informasi dari sebuah citra. Salah satu komponen yang dapat digunakan untuk ektraski fitur adalah tekstur. Elemen yang digunakan pada ekstraksi fitur berdasarkan tekstur terdiri dari tingkat kekasaran citra, tingkat kontras citra, dan *directionality*. Komponen-komponen yang dapat digunakan untuk mendapatkan nilai fitur dari tekstur citra antara lain: *contrast, diagonal moment, energy, entropy, homeogeneity, second diagonal moment, dan uniformity*. Nilai-nilai tersebut didapatkan dengan menggunakan persamaan 2.8(Park dkk., 2004).

$$Contrast = \sum_{i=0}^{N-1} \sum_{j=1}^{N-1} (i-j)^2 P(i,j)$$

$$DiagonalMoment = \sum_{i=0}^{N-1} \sum_{j=1}^{N-1} \sqrt{\frac{|i-j|P(i,j)}{2}}$$

$$Energy = \sum_{i=0}^{N-1}\sum_{j=1}^{N-1} P(i,j)^2$$

$$Entropy = -\sum_{i=0}^{N-1}\sum_{j=1}^{N-1} P(i,j)\log(P(i,j))$$

$$Homegeneity = \sum_{i=0}^{N-1}\sum_{j=1}^{N-1} \frac{P(i,j)}{1+(i-j)^2}$$

$$SecondDiagonalMoment = \sum_{i=0}^{N-1}\sum_{j=1}^{N-1} \frac{|i-j|P(i,j)}{2}$$

$$Uniformity = \sum_{i=0}^{N-1}\sum_{j=1}^{N-1} \frac{P(i,j)}{1+|i-j|} \quad (2.8)$$

dimana:

P(i,j) = nilai *pixel* pada baris ke-*i* dan kolom ke-*j (i = 0,1, ... , m* dan *j = 0,1, ... , n*)

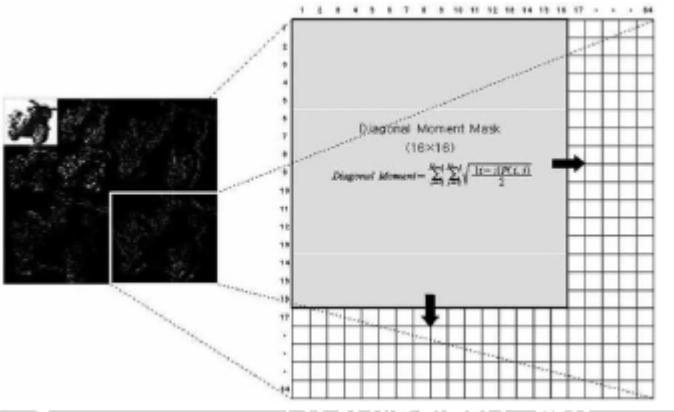
m = jumlah baris pada citra

n = jumlah kolom pada citra

Dari beberapa komponen yang dapat digunakan untuk mendapatkan nilai fitur tekstur pada citra, komponen yang paling efektif adalah *diagonal moment*(Park dkk., 2004).

Untuk melakukan ekstraksi fitur, salah satu metode yang digunakan pada *feature extraction* adalah metode *slidingwindow-based feature extraction*. Metode ini mendefinisikan sebuah *window*pada sebuah citra dan kemudian menggesernya dari kiri ke kanan, dan dari atas ke bawah. Setiap perpindahan *window*ini akan dihitung nilai fitur dari citra tersebut menggunakan persamaan 2.8(sesuai dengan komponen yang digunakan untuk menghitung nilai fitur). Ilustrasi dari proses ekstraksi fitur menggunakan

metodes*lidingwindow-based feature extraction*dapat dilihat pada Gambar 2.9(Park dkk., 2004).



Gambar 2.9 Ilustrasi *Sliding Window-based Feature Extraction*

2.9. Konsep dasar pengenalan pola

Pengenalan pola menjadi bidang yang semakin berkembang karena kebutuhan manusia untuk membuat sistem cerdas yang dapat membantu otomatisasi pekerjaan manusia. Pengenalan pola merupakan komponen penting yang digunakan untuk mengolah data ataupun mengambil keputusan.

Pola merupakan entitas yang terdefinisi dan dapat didefinisikan melalui ciri-cirinya (dalam dunia pengolahan citra digital, ciri suatu citra didapatkan dari fitur yang terkandung dalam citra tersebut) sehingga dengan ciri tersebut suatu pola dapat dibedakan dengan pola yang lain. Untuk mendapatkan hasil pengenalan pola yang maksimal, maka dibutuhkan ciri yang baik yang dapat membedakan pola yang satu dengan pola yang lain. Ciri yang baik adalah ciri yang memiliki tingkat pembeda yang tinggi sehingga hasil pengelompokan pola memiliki tingkat akurasi yang tinggi(Munir, 2004).

2.10. Jaringan Syaraf Tiruan

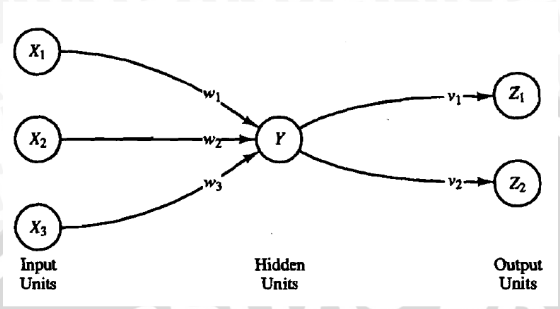
Seiring dengan perkembangan komputer yang sangat pesat, para ahli komputer berusaha menjadikan komputer agar dapat melakukan aktifitas seperti manusia, walaupun dari sudut pandang manusia hal tersebut adalah pekerjaan yang sederhana. Salah satu

aktifitas yang menginspirasi perkembangan ilmu jaringan syaraf tiruan adalah kemampuan manusia untuk belajar dari contoh. Dengan sebuah atau beberapa contoh dan disertai dengan *feedback* dari lingkungan sekitar (seperti guru, orang tua, atau teman), manusia dapat dengan mudah mengenali huruf A atau membedakan antara kucing dengan burung. Kemampuan ini semakin baik dan meningkat seiring bertambahnya pengalaman yang dimiliki oleh manusia. Aktifitas manusia lain yang menginspirasi jaringan syaraf tiruan adalah kemampuan manusia untuk memaksimalkan sumber daya dengan tetap memperhatikan batasan-batasan yang sudah ditentukan. Beberapa bidang yang menerapkan jaringan syaraf tiruan antara lain pemrosesan sinyal, pengenalan pola, ilmu pengobatan, pembuatan suara, pengenalan suara, dan bisnis(Fausset, 1994).

Jaringan syaraf tiruan (JST) juga disebut sebagaisebuah konsep otak buatan, yaitu model matematika dari otak manusia dalam bidang teknik atau bidang lainnya (Lippmann, 1987).

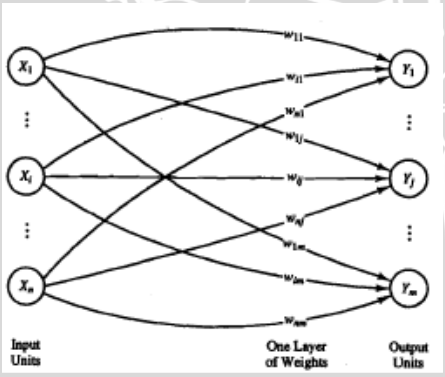
Jaringan syaraf tiruan juga disebut sebagai sistem pemroses informasi yang memiliki karakteristik hampir sama dengan jaringan syaraf manusia. Secara sederhana, arsitektur jaringan syaraf tiruan dapat dilihat pada Gambar 2.10. Pemodelan jaringan syaraf manusia kedalam model matematis menjadi jaringan syaraf tiruan didasari oleh beberapa hal, antara lain:

1. Pemrosesan informasi terjadi di bagian jaringan syaraf bernama *neuron.*
2. Setiap sinyal yang melewati *neuron*akan melalui sebuah penghubung.
3. Setiap penghubung memiliki bobot masing-masing
4. Setiap *neuron* akan menjumlahkan sinyal input yang terboboti dan kemudian menerapkan fungsi aktivasi untuk menentukan sinyal keluaran dari sistem.

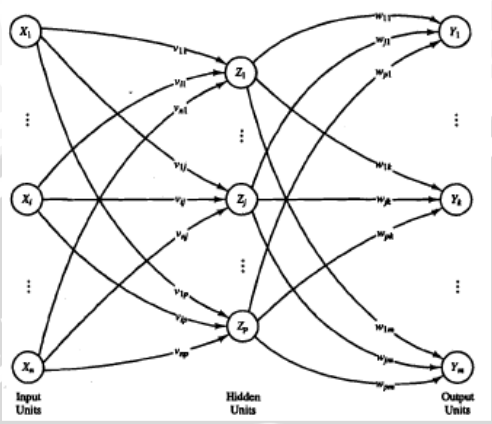


Gambar 2.10Jaringan Syaraf Tiruan Sederhana

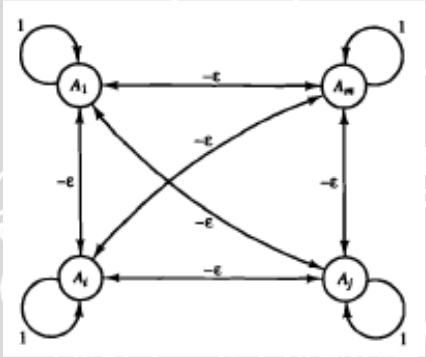
Berdasarkan jumlah *layer* yang membentuk sebuah jaringan, JST dapat dikategorikan menjadi *single layer*, *multilayer*, dan *competitive layer*. Berdasarkan arsitektur yang membentuknya, jaringan syaraf tiruan dibagi menjadi dua kategori yaitu struktur *feedforward* (yaitu arsitektur jaringan dimana sinyal mengalir dari lapisan masukan ke lapisan keluaran dengan arah maju) dan struktur *recurrent* (yaitu terdapat jalur sinyal yang berputar balik dari unit ke unit itu sendiri artinya lapisan keluaran akan memberikan lagi sinyal bagi lapisan masukan). Gambar 2.11dan Gambar 2.12menunjukkan jaringan syaraf tiruan *feedforward*, sedangkan Gambar 2.13menunjukkan jaringan syaraf tiruan *recurrent*(Fausset, 1994).



Gambar 2.11Jaringan Syaraf Buatan *Single Layer*



Gambar 2.12Jaringan Syaraf Buatan *Multilayer*


Gambar 2.13Jaringan Syaraf Buatan *Competitive Layer*

2.10.1. Inisialisasi Bobot dan Bias

Metode yang paling umum digunakan untuk menginisialisasi bobot adalah dengan cara membangkitkan sebuah bilangan kecil acak seperti membangkitkan bilangan acak antara rentang 0 – 1(Fausset, 1994).

Selain metode nilai acak, metode yang digunakan untuk menginisialisasi bobot adalah dengan cara memberikan nilai yang sama untuk semua bobot. Bobot awal akan mempengaruhi apakah jaringan mencapai titik minimum lokal atau global, dan seberapa cepat konvergensinya.

Selain dengan cara memberikan nilai acak atau nilai yang sama untuk semua bobot, terdapat cara lain untuk mendapatkan nilai

bobot awal untuk sebuah jaringan yaitu menggunakan metode *Nguyen-Widrow*(Nguyen & Widrow, 1990).

Metode *Nguyen-Widrow*diperkenalkan oleh dua peneliti dari Universitas Stanford yaitu Derrick Nguyen and Bernard Widrow. Nguyen dan Widrow memperkenalkan sebuah metode untuk menginisialisasi nilai bobot awal dan bias ke unit tersembunyi sehingga dapat memaksimalkan bobot awal jaringan. Harapan dari metode ini adalah kecepatan proses pembelajaran jaringan syaraf tiruan dapat ditingkatkan.

Untuk mengimplementasaikan metode Nguyen-Widrow pada proses inisialisasi bobot, berikut adalah langkah-langkah yang dilakukan:

1. Inisialisasi semua bobot (V_{ij}) dengan bilangan acak pada interval yang ditentukan. Dari penelitian yang telah dilakukan oleh Nguyen dan Widrow, interval nilai [-0.5, 0.5] menghasilkan *Mean Square Error (MSE)* dan waktu pelatihan jaringan yang singkat.

2. Hitung nilai beta dengan menggunakan persamaan 2.9

$$\beta = 0.7 \sqrt[N]{h} \qquad (2.9)$$

dimana:
β= faktor skala Nguyen-Widrow
h= jumlah unit tersembunyi (hidden layer)
N= jumlah unit masukan (input layer)

3. Menghitung nilai‖V‖ dengan menggunakan persamaan 2.10

$$\|V_j\| = \sqrt{\sum_{i=0}^{n} {V_{ij}}^2} \qquad (2.10)$$

dimana:
V_{ij} = nilai bobot lama

4. Menghitung nilai bobot V baru dengan menggunakan persamaan 2.11

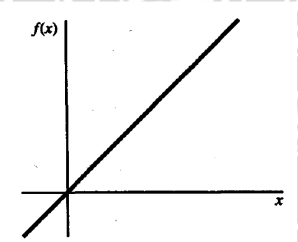
$$V_{ij(baru)} = \frac{\beta V_{ij(lama)}}{\|V_j\|} \qquad (2.11)$$

5. Menginisialisasi nilai bias Vdengan membangkitkan bilangan acak antara $-\beta$ dan β

2.10.2. Fungsi Aktivasi Umum

Fungsi aktivasi adalah sebuah fungsi yang digunakan untuk menentukan hasil keluaran dari sebuah *neuron* pada setiap lapisan jaringan syaraf tiruan. Fungsi aktivasi ini diterapkan pada setiap lapisan pada sebuah jaringan syaraf tiruan.

Terdapat beberapa macam fungsi aktivasi antara lain: fungsi aktivasi identitas, fungsi aktivasi biner, fungsi aktivasi biner *sigmoid*, fungsi aktivasi bipolar *sigmoid*. Untuk lapisan input, fungsi aktivasi yang digunakan adalah fungsi aktivasi identitas. Fungsi aktivasi identitas didefinisikan sebagai $f(x) = x$untuk semua nilai x. Jika digambarkan sebagai sebuah grafik, maka fungsi aktivasi identitas ini ditunjukkan oleh Gambar 2.14.

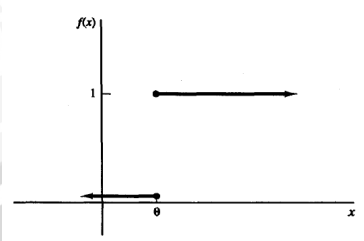


Gambar 2.14 Fungsi Aktivasi Identitas

Fungsi aktivasi yang selanjutnya adalah fungsi aktivasi biner. Fungsi aktivasi biner ini digunakan oleh jaringan *single-layer* untuk mengubah sinyal masukan yang berupa nilai kontinu menjadi nilai biner (0 dan 1) atau bipolar (-1 dan 1). Fungsi aktivasi ini dinyatakanseperti pada persamaan 2.12.

$$f(x) = \begin{cases} 1 & jikax \geq \theta \\ 0 & jikax < \theta \end{cases} \qquad (2.12)$$

Untuk kasus bipolar, maka angka 1 dan 0 diganti dengan -1 dan 1. Secara grafik, fungsi aktifasi biner dapat dilihat pada Gambar 2.15



Gambar 2.15 Fungsi Aktivasi Biner

Fungsi aktivasi yang selanjutnya adalah fungsi aktivasi *sigmoid*. Fungsi *sigmoid* memiliki kurva yang berbentuk S (seperti terlihat pada Gambar 2.16). Fungsi aktivasi sigmoid ini sering digunakan untuk jaringan *backpropagation*. Hal ini dikarenakan fungsi ini sangat mudah untuk dideferensiasikan.

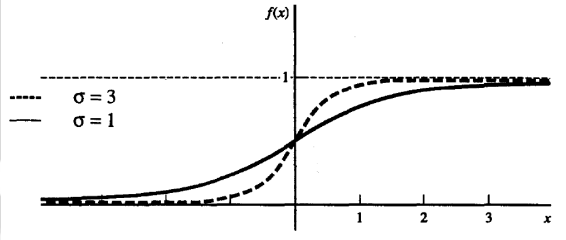
Fungsi sigmoid juga sering disebut dengan fungsi logistik. Fungsi ini dapat diskalakan pada rentang tertentu. Fungsi *sigmoid* yang sering digunakan untuk jaringan syaraf tiruan yang nilai keluarannya berupa biner atau nilai antara 0 dan 1 adalah fungsi *sigmoid* dengan skala 0 sampai 1,atau lebih sering disebut dengan *binary sigmoid*. Fungsi aktivasi *binary sigmoid*dinyatakan seperti pada persamaan 2.13.

$$f(x) = \frac{1}{1 + e^{-\sigma x}} \quad (2.13)$$

dan turunannya dinyatakan seperti pada persamaan 2.14.

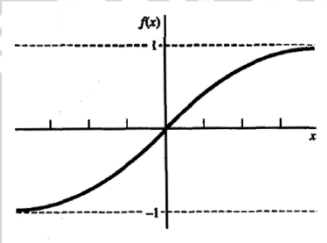
$$f'(x) = f(x)[1 - f(x)] \quad (2.14)$$

Jika digambarkan sebagai sebuah grafik, maka fungsi ini dapat dilihat pada Gambar 2.16dengan dua nilai steepness(σ).



Gambar 2.16 Fungsi Aktivasi Biner*Sigmoid*

Karena fungsi *sigmoid* ini dapat diskalakan sesuai dengan kasus yang ada, maka terdapat beberapa variasi dari fungsi sigmoid ini. Skala yang sering digunakan pada fungsi sigmoid adalah -1 sampai 1, atau sering disebut dengan bipolar *sigmoid*. Secara grafik, fungsi bipolar sigmoid ini dapat dilihat pada Gambar 2.17dengan nilai *steepness*(σ) = 1(Fausset, 1994).



Gambar 2.17 Fungsi Aktivasi Bipolar *Sigmoid*

2.10.3. Pengukuran Galat Jaringan

Metode pengukuran galat yang digunakan pada jaringan syaraf tiruan adalah metode *Sum Square Error (SSE)* dan *Root Mean Square Error (RMSE)*.SSE merupakan hasil penjumlahan nilai kuadrat *error*dari nilai prediksidan nilai *target*pada lapisan keluaran untukmasing-masing data, dimana hasil penjumlahan keseluruhan nilai SSE akan digunakan untuk menghitung nilai RMSE pada tiap iterasi. Untuk menghitung nilai SSE dapat digunakan persamaan 2.15.

$$SSE = \sum_{i-1}^{N} (t_{ij} - y_{ij}) \qquad (2.15)$$

dimana:
N = banyaknya nilai target pada lapisan keluaran
t_{ij} = nilai prediksi hasil keluaran dari jaringan
y_{ij} = nilai target dari data latih.

Dari nilai SSE, dihitung nilai *Mean Square Error (MSE)* yang digunakan untuk mengukur galat jaringan. MSE adalah sebuah cara untuk mengukur perbedaan antara nilai prediksi (nilai yang diberikan oleh *estimator*) dan nilai asli dari data yang sedang diprediksi.

Untuk menghitung nilai *Root Mean Square Error (RMSE)*dapat digunakan persamaan 2.16.

$$RMSE = \sqrt{\frac{SSE}{N}} \quad (2.16)$$

dimana:
N = jumlah data latih

2.11. Metode *Backpropagation*

Metode pelatihan*backpropagation* adalah sebuah metode sistematik untuk pelatihanjaringan syaraf tiruan*multilayer*. Metode ini memiliki dasar matematis yang kuat dan objektif. Algoritma ini mendapatkan bentuk persamaan dan nilai koefisien dalam formula dengan meminimalkan jumlah kuadrat *error* melalui model yang dikembangkan (*training set*).

Pelatihan suatu jaringan syaraf tiruan dengan menggunakan algoritma *backpropagation* meliputi dua tahapan yaitu perambatan maju dan perambatan mundur. Pada perambatan mundur dilakukan pengubahan bobot pada masing-masing lapisan pada JST.

Adapun algoritma dari pelatihan jaringan *Backpropagation* adalah sebagai berikut(Fausset, 1994):

Langkah 0: Inisialisasi Bobot
Langkah 1: Selama *stopping condition* masih belum terpenuhi, lakukan langkah 2-9
Langkah 2: Untuk setiap data training yang ada, lakukan langkah 3-8

Perambatan Maju

Langkah 3: Setiap unit masukan (*Xi, i=1...n*) menerima sinyal input, kemudian sinyal tersebut diteruskan ke semua unit diatasnya (unit tersembunyi)
Langkah 4: Setiap unit tersembunyi $(Z_j, j = 1 \dots p)$ menjumlahkan semua sinyal input yang sudah diboboti menggunakan persamaan 2.17.

$$z\_in_j = v_{0j} + \sum_{i=1}^{n} x_i v_{ij} \qquad (2.17)$$

Kemudian dihitung nilai dari fungsi aktivasinya menggunakan persamaan 2.18.

$$z_j = f(z\_in_j) \qquad (2.18)$$

dan mengirimkan sinyal ini ke semua unit diatasnya (unit keluaran)

Langkah 5: Setiap unit keluaran $(Y_k, k = 1 \dots m)$ menjumlahkan semua sinyal yang sudah terboboti dari lapisan tersembunyi menggunakan persamaan 2.19.

$$y\_in_k = w_{0k} + \sum_{j=1}^{p} z_j w_{jk} \qquad (2.19)$$

dan menghitung fungsi aktivasinya untuk mendapatkan nilai sinyal output menggunakan persamaan 2.20.

$$y_k = f(y\_in_k) \qquad (2.20)$$

Perambatan Mundur dari *Error*

Langkah 6: Setiap unit keluaran $(Y_k, j = 1 \dots m)$ mendapatkan nilai target yang sesuai dengan pola *training* masukan dan menghitung *error* menggunakan persamaan 2.21.

$$\delta_k = (t_k - y_k) f'(y\_in_k) \qquad (2.21)$$

menghitung nilai koreksi bobot untuk memperbarui nilai w_{jk}menggunakan persamaan 2.22.

$$\Delta w_{jk} = \alpha \delta_k z_j \qquad (2.22)$$

menghitung nilai koreksi bias untuk memperbarui nilai

w_{0k}menggunakanpersamaan 2.23

$$\Delta w_{0k} = \alpha \delta_k \qquad (2.23)$$

dan mengirimkan δ_k ke setiap unit di lapisan dibawahnya (lapisan tersembunyi)

Langkah 7: Setiap unit tersembunyi $(Z_j, j = 1 \dots p)$ menjumlahkan nilai *delta* dari unit di lapisan keluaran menggunakan persamaan 2.24.

$$\delta\_in_j = \sum_{k=1}^{m} \delta_k w_{jk} \qquad (2.24)$$

menghitung informasi *error* menggunakan persamaan 2.25.

$$\delta_j = \delta\_in_j \, f'(z\_in_j) \qquad (2.25)$$

menghitung nilai koreksi bobot untuk memperbarui nilai v_{ij}menggunakan persamaan 2.26.

$$\Delta v_{ij} = \alpha \delta_j x_i (2.26)$$

menghitung nilai koreksi bias untuk memperbarui nilai v_{0j} menggunakan persamaan 2.27.

$$\Delta v_{0j} = \alpha \delta_j (2.27)$$

Memperbarui Bobot dan Bias

Langkah 8: Setiap unit keluaran $(Y_k, k = 1 \dots m)$ memperbarui nilai bias dan bobot $(j = 0, \dots, p)$ menggunakan persamaan 2.28

$$w_{jk}(new) = w_{jk}(old) + \Delta w_{jk} \qquad (2.28)$$

Setiap unit tersembunyi $(Z_j, j = 1 \dots p)$ memperbarui nilai bias dan bobot $(i = 0, \dots, n)$ menggunakan persamaan 2.29

$$v_{ij}(new) = v_{ij}(old) + \Delta v_{ij} \qquad (2.29)$$

Langkah 9: Uji *stopping condition*

Berikut ini adalah beberapa istilah yang digunakan pada algoritma *backpropagation*:

X = vektor latih masukan

t = vektor target keluaran

δ_k = nilai *error* dari jaringan

Fungsi Aktivasi

Fungsi aktivasi yang sering digunakan pada metode pembelajaran*backpropagation* adalah fungsi aktivasi *binary sigmoid*. Fungsi aktivasi ini dinyatakan oleh persamaan 2.13 seperti yang sudah dipaparkan pada sub bab 2.10. Pada metode pembelajaran *backpropagation*, fungsi aktivasi yang digunakan harus memiliki beberapa karakteristik antara lain(Fausset, 1994):

1. Harus bersifat kontinu
2. Harus bisa diturunkan (diferensiasi). Dan untuk efisiensi perhitungan, maka turunan dari fungsi aktivasi ini diharapkan dapat dengan mudah dihitung. Pada beberapa fungsi aktivasi, nilai turunan sebuah fungsi dapat dihitung dari nilai fungsi itu sendiri sehingga hal ini akan mempermudah perhitungan turunan dari fungsi tersebut.

2.12. Metode *Optical Backpropagation*

Metode *Optical Backpropagation* (OBP) didesain untuk menyelesaikan masalah pada standar *Backpropagation* (BP) yang menggunakan fungsi nonlinear. Metode JST dengan menggunakan metode *Backpropagation*memang menunjukkan hasil yang cukup baik, tetapi metode *Backpropagation*memiliki beberapa kekurangan antara lain proses pelatihan membutuhkan waktu yang lama untuk mencapai konvergensi pada beberapa kasus dan metode ini juga bisa terperangkap pada lokal minima. Salah satu cara yang dapat dilakukan agar jaringan yang dilatih menggunakan metode*Backpropagation*tidak terperangkap pada lokal minima adalah dengan memberikan nilai *learning rate* yang sangat kecil, namun hal ini justru akan memperlambat kinerja jaringan untuk mencapai konvergensi. Sehingga diusulkan sebuah metode yang

mampu memperbaiki kinerja dari metode *Backpropagation*yaitu metode *Optical Backpropagation*(Otair & Salameh, 2005)

Aspek penting dalam metode ini yaitu kemampuan untuk keluar dari lokal minima dengan kecepatan konvergensi yang tinggi pada saat pelatihan. Prinsip dasar dari OBP ini terletak pada pengaturan *error* yang ditransmisikan ke belakang dari lapisan *output* pada tiap unit pada lapisan tersembunyi. Jika dalam BP, kesalahan pada *output* tunggal dinyatakan seperti pada persamaan 2.30.

$$\delta_k^0 = (t_k - y_k) y_k (1 - y_k) \qquad (2.30)$$

dimana
k = keluaran ke-k.
t_k= nilai output yang diinginkan
y_k =*output* actual dari unit ke-k

maka pada OBP, *error single output* diatur menggunakan persamaan 2.31dan 2.32(Otair & Salameh, 2005):

Jika $(t_k - y_k) \geq 0$ maka

$$New\delta_k^0 = \left(1 + e^{(t_k - y_k)^2}\right) y_k (1 - y_k) \qquad (2.31)$$

Jika $(t_k - y_k) < 0$ maka

$$New\delta_k^0 = -\left(1 + e^{(t_k - y_k)^2}\right) y_k (1 - y_k) \qquad (2.32)$$

Langkah-langkah pelatihan JST yang dilakukan pada metode pelatihan*optical backpropagation* sama seperti pada metode pelatihan *backpropagation*. Perbedaanya terletak pada nilai informasi error yang digunakan (δ_k^0). Pada metode OBP, nilai δ_k^0 diganti dengan $New\delta_k^0$sesuai dengan syarat yang terpenuhi. Nilai$New\delta_k^0$ ini akan meminimalkan tingkat error dari unit keluaran dengan lebih cepat jika dibandingkan dengan δ˚pk. Pada OBP, bobot pada unit tertentu akan mengalami perubahan yang cukup besar dari nilai awalnya(Otair & Salameh, 2005).

2.13. Sistem Anotasi Citra

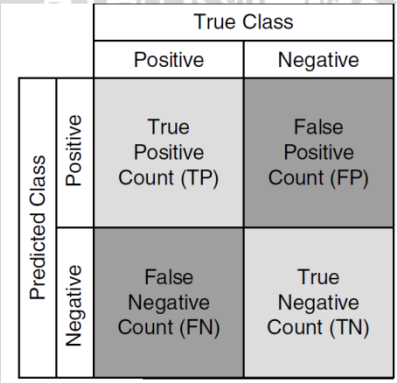
Sistem anotasi citra dapat dianggap sebagai sebuah proses untuk menerjemahkan daerah-daerah tertentu pada citra (disebut sebagai *blobs*) ke kata kunci yang berarti(Chen dkk., 2010).

Tujuan dari sistem anotasi citra adalah untuk memberikan label atau kata kunci pada sebuah citra. Anotasi dapat didefinisikan dengan berbagai cara, antara lain mengasosiasikan kata kunci dengan bagian tertentu (*region*) pada citra atau memberikan sebuah "deskripsi" pada citra yang diharapkan dapat membantu proses pencarian citra (Tsai dkk., 2011).

2.14. Pengujian Tingkat Akurasi Sistem

Setelah sebuah sistem *predictor* berhasil dikembangkan, maka tingkat akurasi dari sistem*predictor* tersebut harus diuji. Terdapat beberapa standar pengukuran yang digunakan untuk melakukan pengujian ini, diantaranya adalah *recall*, *precision* dan *F-measure*. Metode ini adalah metode yang paling umum digunakan untuk menguji tingkat akurasi dari *image retrieval* karena kemampuannya untuk merepresentasikan relevansi dari hasil yang didapatkan pada *image retrieval*.

Pada kasus klasifikasi, sumber utama untuk melakukan pengujian tingkat akurasi adalah matriks koinsiden. Matriks ini ditunjukkan pada Gambar 2.18. Matriks ini digunakan untuk menghitung nilai *recall*, *precision*, dan *F-measure* pada kasus klasifikasi(Olson & Delen, 2008).



Gambar 2.18Matriks Koinsiden untuk Pengujian Sistem

Matriks koinsiden menunjukkan perbandingan antara hasil prediksi klasifikasi oleh sistem dan klasifikasi sebenarnya dari kelompok data.

Pada matriks koinsiden, *True Positive* (TP) menunjukkan bahwa data yang diklasifikasikan oleh sistem memang anggota sebenarnya dari klasifikasi tersebut. *False Positive* (FP) menunjukkan bahwa data yang yang diklasifikasikan oleh sistem ternyata bukan anggota sebenarnya dari klasifikasi tersebut,*False Negative* (FN) menunjukkan bahwa data yang tidak termasuk pada hasil klasifikasi seharusnya merupakan anggota dari kelompok tersebut, dan *True Negative* (TN) menunjukkan bahwa data yang tidak termasuk pada hasil klasifikasi ternyata memang bukan anggota dari klasifikasi tersebut. Pada kasus klasifikasi, nilai *recall* dan *precision* didapatkan dari matriks koinsiden ini.

Recall adalah nilai rasio diantara jumlah dokumen relevan yang berhasil didapatkan dan total dokumen relevan pada sebuah sistem. *Recall* digunakan untuk mengukur tingkat kemampuan *retrieval* dari sebuah sistem *image retrieval (IR)*. Semakin tinggi nilai *recall* menunjukkan bahwa sistem yang dibangun menghasilkan hasil pencarian yang semakin relevan(Chu, 2003). Pada kasus klasifikasi data, nilai *recall* didapatkan dari matriks koinsiden dengan menerapkan persamaan 2.33(Olson & Delen, 2008).

$$Recall = \frac{TP}{TP + FN} \tag{2.33}$$

dimana:
TP = nilai *True Positive Count* pada matriks koinsiden.
FN = nilai *False Negative Count* pada matriks koinsiden.

Sedangkan *precision* adalah nilai rasio diantara jumlah dokumen yang relevan dan jumlah dokumen yang berhasil didapatkan dari sebuah proses *retrieval*. Dengan kata lain, *precision* menyatakan persentase kebenaran dari seluruh dokumen hasil klasfikasi. Semakin tinggi nilai *precision* artinya sebuah sistem *image retrieval* menghasilkan hasil relevan yang lebih banyak daripada hasil yang tidak relevan. Pada dasarnya, nilai *precision* dapat digunakan untuk menilai kemampuan dari sistem *Image*

Retrieval (IR) untuk memisahkan hasil yang tidak relevan dari hasil yang relevan(Chu, 2003). Pada kasus klasifikasi data, nilai *precision* didapatkan dari matriks koinsiden dengan menerapkan persamaan 2.34 (Olson & Delen, 2008).

$$Precision = \frac{TP}{TP + FP} \quad (2.34)$$

dimana:
TP = nilai *True Positive Count* pada matriks koinsiden.
FP = nilai *False Negative Count* pada matriks koinsiden.

Recall dan *precision* adalah dua penilaian yang berbeda, sehingga diperlukan sebuah nilai yang dapat menggabungkan kedua nilai tersebut menjadi sebuah nilai tunggal untuk melakukan penilaian tingkat akurasi. Untuk menggabungkan *recall* dan *precision* menjadi satu penilaian tunggal dapat digunakan nilai *F-measure*. *F-measure* merupakan gabungan antara *recall* dan *precision*(Chu, 2003).

Pada kasus klasifikasi data, nilai *F-measure* didapatkan dari hasil perhitungan *recall* dan *precision* dengan menerapkan persamaan 2.35 (Olson & Delen, 2008).

$$F = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} \quad (2.35)$$

BAB III
METODOLOGI DAN PERANCANGAN

Pada bab ini akan dibahas mengenai metodologi dan perancangan sistem anotasi citra otomatis menggunakan metode jaringan syaraf tiruan *Optical Backpropagation* dan metode *Content Based Image Retrieval*menggunakan transformasi *Haar Wavelet*. Berikut ini adalah tahapan yang dilakukandalam penelitian ini:

1. Mempelajari metode transformasi *Wavelet* dan *Sliding-window Based Feature Extraction* yang akan digunakan untuk mengekstrasi fitur dari citra.
2. Mempelajari metode pelatihan jaringan syaraf tiruan *Optical Backpropagation*yang akan digunakan untuk proses klasifikasi citra menggunakan informasi fitur yang sudah terekstraksi dari citra.
3. Mengumpulkan data citra yang akan digunakan sebagai data pembelajaran jaringan syaraf tiruan dan sebagai data uji.
4. Menganalisa dan merancang perangkat lunak sistem anotasi citra otomatis.
5. Mengimplentasikan perangkat lunak berdasarkan analisa dan perancangan yang telah dilaksanakan.
6. Melakukan uji coba perangkat lunak yang telah dibuat dengan menggunakan beberapa citra uji.
7. Melakukan evaluasi perangkat lunak yang telah dibuat terhadap beberapa data uji menggunakan metode SSE, RMSE, dan*F-measure*.

3.1. Analisis Data

Data yang digunakan untuk penelitian ini adalah data citra dengan format JPG. Data citra didapatkan dari *WANG database*. *Database* ini merupakan kumpulan dari 1.000 citra yang diambil dari *Corel Stock Photo Database*yang diseleksi secara manual dandibentuk menjadi 10 kelompok kategori citra dimana pada masing-masing kelompoktersebut terdapat 100 citra. 10 kelompok citra yang terdapat pada *WANG database* ini antara lain:

1. Kelompok citra suku asmat
2. Kelompok citra pemandangan pantai

3. Kelompok citra pemandangan bangunan kuno
4. Kelompok citra bus bertingkat
5. Kelompok citra dinosaurus
6. Kelompok citra pemandangan gajah di alam liar
7. Kelompok citra bunga mawar
8. Kelompok citra pemandangan kuda di alam
9. Kelompok citra pemandangan gunung
10. Kelompok citra berbagai jenis makanan

WANG database yang terdiri dari 1.000 citra ini diunduh darihttp://wang.ist.psu.edu/docs/related/. Contoh citra yang diambil dari *WANG database* ini dapat dilihat pada Gambar 3.1.

Pada penelitian ini, citra yang digunakan adalah citra berdimensi 128x128 sehingga semua citra pada *WANG database* akan diubah menjadi citra berdimensi 128x128 terlebih dahulu.

Gambar 3.1 Contoh Citra dari *WANG Database*

Karena pada *WANG database* ini terdapat citra pemandangan, maka tidak semua citra akan digunakan pada penelitian ini melainkan hanya citra obyek saja yang akan digunakan.

Agar data yang diolah menjadi lebih beragam, maka perlu dilakukan penambahan data citra. Pada penelitian ini data tambahan yang digunakan adalah citra yang didapatkan dari internet. Citra yang diambil adalah citra dari sebuah obyek yang memiliki latar belakang putih. Beberapa citra dari internet yang akan digunakan pada penelitian ini antara lain citra apel, gajah, pistol, dan lain-lain. Contoh citra yang didapatkan dari internet dapat dilihat padaGambar 3.2.

Gambar 3.2 Contoh Data Citra dari *Internet*

3.2. Analisis Sistem

Dalam sub bab ini akan dibahas mengenai beberapa kegiatan analisis yang akan dilakukan pada penelitian tentang perancangansistem anotasi citra otomatis menggunakan metode jaringan syaraf tiruan *Optical Backpropagation*dan metode*Content Based Image Retrieval*menggunakan transformasi *Haar Wavelet*.

Pada analisis ini akan dibahas mengenai beberapa hal yaitu deskripsi sistem, kebutuhan sistem dan rancangan hasil dari sistem.

3.2.1. Deskripsi Sistem

Sistem yang akan dikembangkan pada penelitian ini berupa sebuah perangkat lunak/aplikasiyang dapat digunakan untuk memberikan label/deskripsi pada citra secara otomatis berdasarkan konten yang terkandung pada citra.

Aplikasi ini menerapkan metode *Content Based Image Retrieval*dengan menggunakan transformasi *Haar Wavelet* untuk mendapatkan fitur yang terkandung pada citra dan kemudian menggunakannya sebagai data input untuk melatih jaringan syaraf tiruan yang akan digunakan untuk mengenali citra, sehingga aplikasi ini mampu memberikan label pada citra secara otomatis.

Pemberian label pada citra dilakukan berdasarkan pada fitur yang telah diekstraksi dari citra yang direpresentasikan dalam bentuk vektor fitur.

Pada sistem ini, terdapat dua proses utama untuk melakukan anotasi citra yaitu proses pelatihan komputer dan proses pemberian label pada citra. Berikut ini merupakan alur proses yang akan dilaksanakan untuk melatih komputer agar komputer memiliki pengetahuan yang nantinya dapat digunakan untuk mengenali citra:

1. Pengguna memilih direktori (*folder*) yang berisi citra yang akan digunakan sebagai data latih.

2. Pengguna memberikan informasi tentang kelas-kelas yang sesuai dengan citra yang akan digunakan sebagai data latih dan komputer akan menyimpan informasi tersebut.
3. Pada saat pengguna memasukkan data citra masukan, komputer akan mengecek dimensi citra tersebut. Jika citra yang dimasukkan tidak berdimensi 128 x 128(dalam satuan *pixel*), maka sistem akan melakukan normalisasi sehingga dimensi citra menjadi 128 x 128. Proses ini dilakukan secara *background* sehingga perubahan dimensi pada citra asli tidak tampak secara langsung dari sisi pengguna.
4. Kemudian sistem akan melakukan ekstraksi fitur dari citra yang terdiri dari proses:
    a. Pengubahan model warna dari RGB ke HSI
       Proses pertama yang dilakukan untuk mengekstraksi fitur dari citra adalah mengubah citra dari model warna RGB agar didapatkan nilai intensitas dari citra. Nilai intensitas ini akan digunakan sebagai nilai inputan untuk proses transformasi *wavelet*.
    b. Transformasi *Wavelet*
       Merubah citra pada domain spasial menjadi domain frekuensi dan membagi-baginya kedalam beberapa *subband* untuk proses analisis lebih lanjut.
    c. Ekstraksi fitur menggunakan metode *Sliding-window*.
       Setelah proses transformasi *wavelet* berhasil dilakukan, maka langkah selanjutnya adalah menghitung nilai fitur dari citra. Penghitungan ini menggunakan metode *sliding-window* dengan komponen *diagonal moment*.
5. Dari hasil ekstraksi fitur, kemudiansistem akan membangun sebuah jaringan syaraf tiruan menggunakan nilai fitur dari citra yang sudah diketahui. Jaringan syaraf tiruan ini yang akan digunakan untuk melatih komputer.
6. Dari proses pelatihan tersebut, akan diperoleh sebuah arsitektur jaringan syaraf tiruan yang diharapkan mampu mengenali pola dari citra masukan baru.

Dan berikut ini adalah alur proses yang akan dilaksanakan pada saat pemberian label pada citra:

1. Pengguna memilih direktori (*folder*) dari citra yang akan diberi label.
2. Sistem akan memberikan label pada citra tersebut secara otomatis berdasarkan pada pengetahuan yang telah dimiliki oleh komputer.

3.2.2. Kebutuhan Sistem

Kebutuhan dari sistem ini meliputi kebutuhan perangkat lunak dan perangkat keras. Sistem yang dirancang pada penelitian ini akan dikembangkan pada*platform Microsoft Windows® 32 bit* sehingga sistem ini hanya dapat berjalan pada komputer yang menggunakan sistem operasi *Microsoft Windows*. Sistem ini akan dikembangkan menggunakan bahasa pemrograman *C#* dengan menggunakan perangkat lunak *Microsoft Visual Studio 2010*. Sedangkan untuk kebutuhan perangkat keras, pada saat perancangan dan pengujian, sistem ini akan dijalankan pada komputer dengan *processor* AMD Turion™ X2 Dual-Core Mobile RM-74 2.20 GHz dengan memori 2 Gigabyte.

3.2.3. Rancangan Hasil

Seperti yang sudah dipaparkan sebelumnya bahwa tujuan dari penelitian ini adalah untuk merancang dan mengembangkan sebuah sistem anotasi yang mampu memberikan deskripsi atau label pada sebuah citra, maka hasil yang diharapkan dari sistem anotasi citra ini adalah citra yang sudah memiliki label berdasarkan konten yang terkandung pada citra tersebut.

Dari hasil proses pelabelan citra tersebut,pengguna dapat melakukan pencarian citra. Pencarian citra ini dilakukan menggunakan kata kunci berdasarkan pada label yang telah tersimpan untuk masing-masing citra.

Metode pencarian citra menggunakan label ini dipilih karena metode ini diharapkan mampu memberikan hasil dengan cepat dan mampu mendekati konsep pencarian informasi yang dilakukan oleh manusia yaitu pencarian menggunakan kata kunci, bukan menggunakan sebuah citra contoh setiap akan dilakukan pencarian citra.

Adapun contoh dari hasil pencarian citra ini dapat dilihat padaGambar 3.3 yang menggambarkan hasil pencarian citra pada koleksi citra dengan menggunakan kata kunci *Dinosaurus*.

Kata kunci pencarian: *Dinosaurus*

Hasil Pencarian:


401.jpg
Label: Dinosaurus


402.jpg
Label: Dinosaurus


403.jpg
Label: Dinosaurus


404.jpg
Label: Dinosaurus


405.jpg
Label: Dinosaurus

Gambar 3.3Rancangan Hasil Pencarian Citra

3.3. Perancangan Sistem

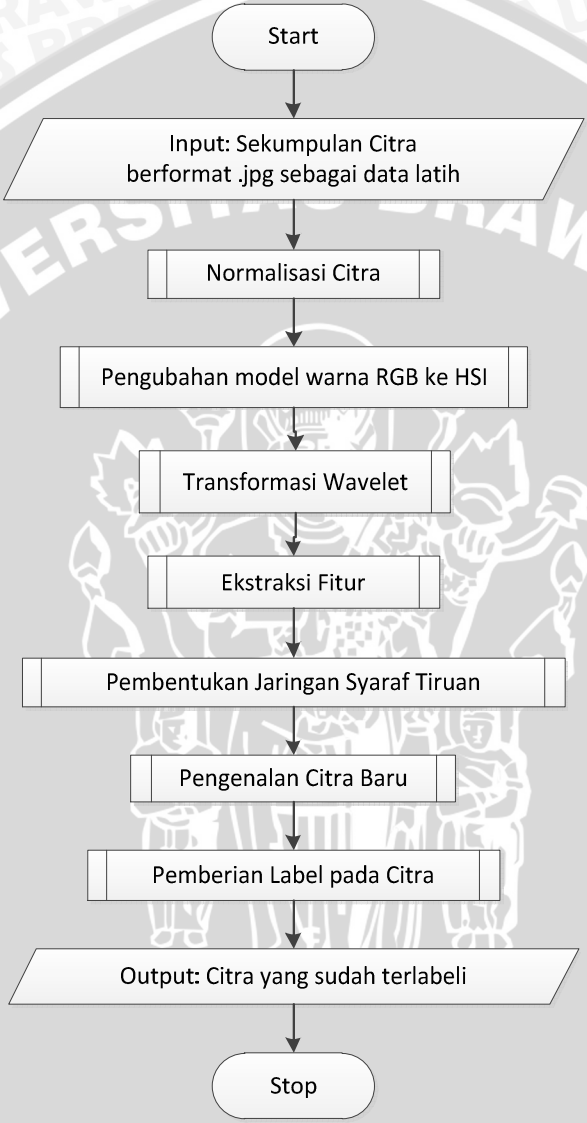
Dalam sub bab ini akan dibahas mengenai beberapa pemodelan sistem dan metode-metode yang digunakan dalam penelitian ini. Dalam perancangan sistem pada penelitian initerdapat beberapaprosesyang dilakukan antara lain perancangan proses, perancangan kelas, perancangan penyimpanan data dan perancangan antar muka.

3.3.1. Perancangan Proses

Salah satu bagian dari perancangan sistem adalah perancangan proses. Perancangan ini membahas mengenai proses-proses yang digunakan untuk membangun sebuah sistem anotasi citra otomatis yang menerapkan metode *ContentBased Image Retrieval*menggunakan jaringan syaraf tiruan*Optical Backpropagation.*

Adapun proses-proses tersebut antara lain normalisasi citra, transformasi *Wavelet*, pengubahan model warna RGB ke HSI, ekstraksi fitur, pembentukan jaringan syaraf tiruan, pengujian jaringan syaraf tiruan untuk pengenalan citra, pengenalan citra baru dan pemberian label pada citra.Semua proses tersebut dijalankan secara berurutan sehingga output pada suatu proses akan digunakan

sebagai input pada proses selanjutnya. Jika digambarkan pada sebuah bagan, maka perancangan proses sistem anotasi ini dapat dilihat pada *flowchart*seperti yang terlihat padaGambar 3.4.



Gambar 3.4*Flowchart* Proses Sistem Anotasi

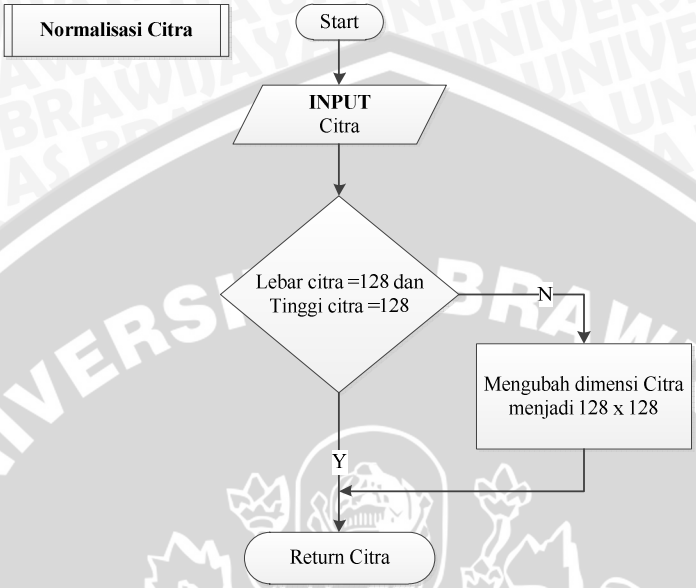
3.3.1.1. Normalisasi Citra

Proses pertama yang dilakukan pada sistem anotasi citra adalah menormalisasikan citra. Proses ini mengecek apakah citra yang dijadikan data masukan berdimensi 128x128 atau tidak, jika citra yang dimasukkan tidak berdimensi 128x128 maka dimensi citra akan diubah. Proses ini menjadi proses yang penting karena pada proses selanjutnya dibutuhkan matriks citra yang berukuran 128x128.

Beberapa langkah yang dilakukan pada proses normalisasi citra ini antara lain:

1. Membaca citra masukan sehingga didapatkan informasi tentang tinggi dan lebar dari sebuah citra.
2. Melakukan pengecekan apakah ukuran tinggi dan lebar dari sebuah citra adalah 128.
3. Jika ukuran tinggi dan lebar dari sebuah citra adalah 128 maka tidak dilakukan proses apapun, tetapi jika tinggi dan lebar sebuah citra tidak 128 maka akan dilakukan proses pengubahan dimensi citra menjadi 128 x 128 *pixel*.

Adapun hasil dari proses ini adalah citra yang berdimensi 128x128.*Flowchart* dari proses normalisasi citra ini dapat dilihat pada Gambar 3.5.



Gambar 3.5*Flowchart* Proses Normalisasi Citra

3.3.1.2. Pengubahan Model Warna RGB ke HSI

Setelah dilakukan proses normalisasi pada citra, maka proses selanjutnya adalah mengubah model warna citra dari model RGB ke model HSI.

Proses ini bertujuan untuk mendapatkan nilai *intensity* yang terkandung pada citra. Nilai *intensity* ini sangat berguna untuk menganalisa pola dari suatu benda karena nilai ini terisolasi dari informasi warna, sehingga sebuah obyek yang sama tetapi memiliki warna yang berbeda tetap dianggap sebagai satu obyek saja.

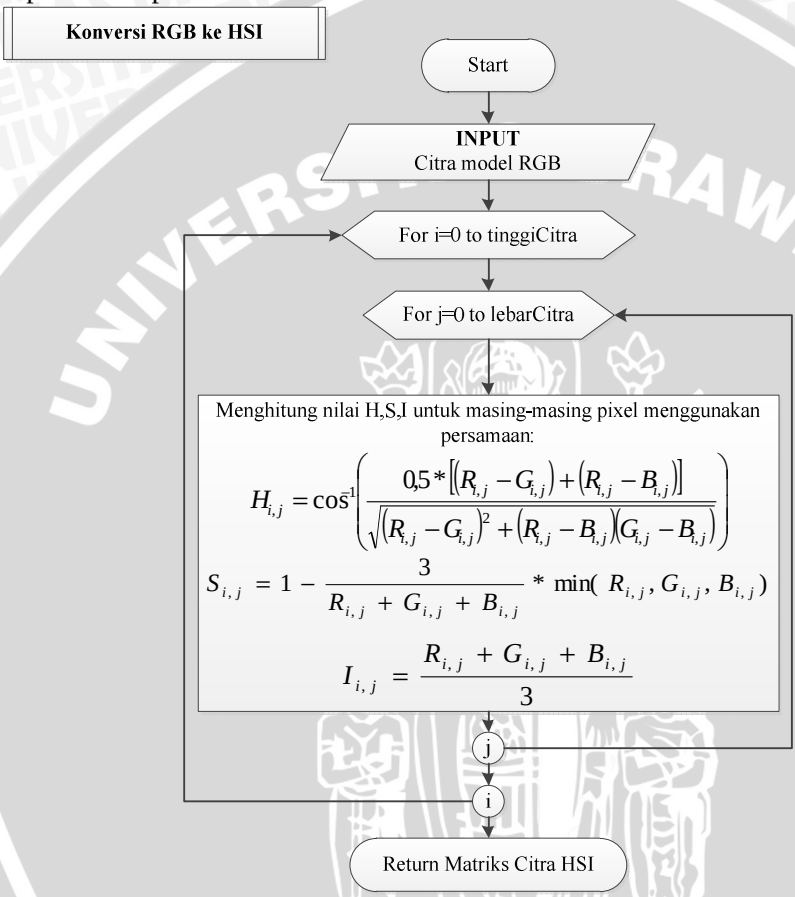
Langkah-langkah yang dilakukan untuk mendapatkan nilai HSI pada citra adalah sebagai berikut:

1. Membaca matriks citra inputan dalam model warna RGB.
2. Melakukan penelusuran untuk setiap *pixel* yang ada pada matriks citra tersebut dan menghitung nilai untuk komponen HSI dari komponen *red, green,* dan *blue* yang dimiliki oleh setiap *pixel* pada citra.

Hasil dari proses ini adalah citra yang sudah dalam model warna HSI. Pada penelitian ini, nilai pada model warna HSI yang

akan digunakan untuk proses ekstraksi fitur adalah nilai *intensity*. Nilai *intensity* ini adalah nilai tingkat keabuan dari sebuah citra.

Adapun flowchart dari proses pengubahan model warna ini dapat dilihat pada Gambar 3.6.



Gambar 3.6*Flowchart*Konversi Model Warna

3.3.1.3. Transformasi Wavelet

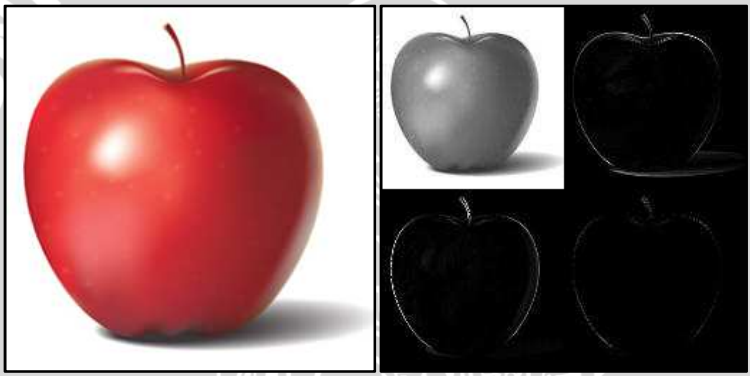
Setelah didapatkan citra yang berdimensi 128 x 128, maka proses selanjutnya adalah mengekstraksi fitur. Untuk dapat mengekstraksi nilai fitur dari citra, langkah pertama adalah melakukan transformasi *wavelet*. Proses ini akan mengklasifikasikan citra dalam beberapa *sub-band* (frekuensi rendah dan tinggi).

Dengan melakukan analisis terhadap *sub-band* yang didapatkan dari proses transformasi *Wavelet* ini, maka akan diperoleh informasi yang terkandung di dalam sebuah citra.

Adapun langkah-langkah yang dilakukan pada proses transformasi *Wavelet*antara lain:

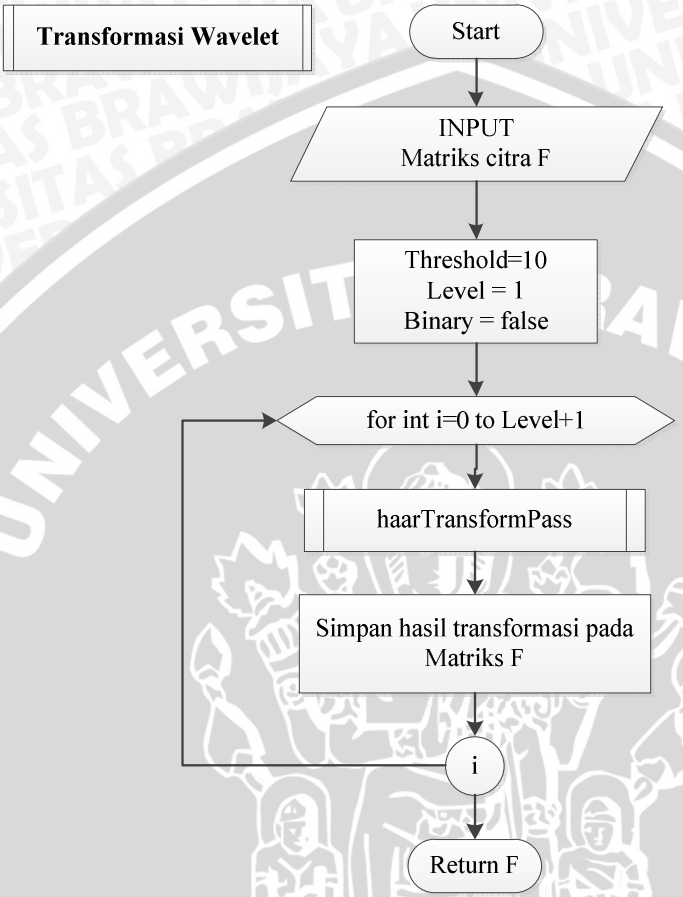
1. Membaca matriks intensitas citra
2. Melakukan proses *averaging* dan *differencing* sebanyak level transformasi sehingga didapatkan citra pada 4 *subband* yaitu LL, LH, HL, dan HH untuk setiap level.

Hasil dari transformasi *Wavelet*ini adalah citra yang sudah berada pada domain frekuensi dan sudah terbagimenjadi beberapa *sub-band* seperti yang terlihat pada Gambar 3.7.

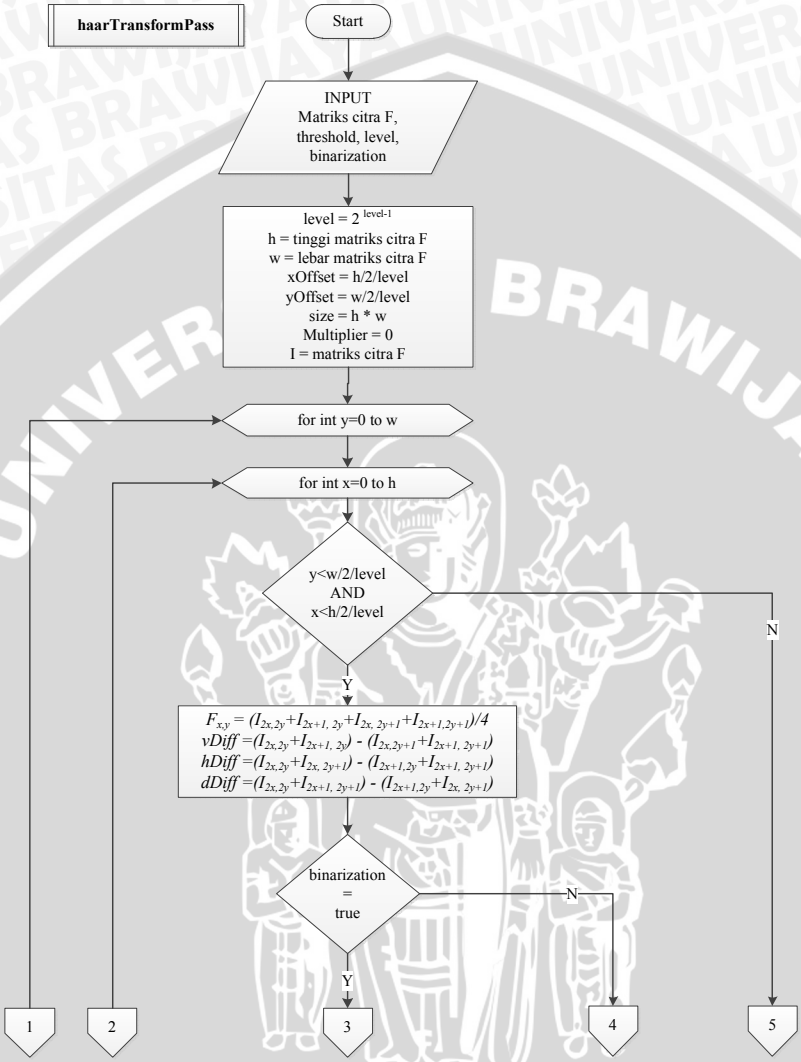


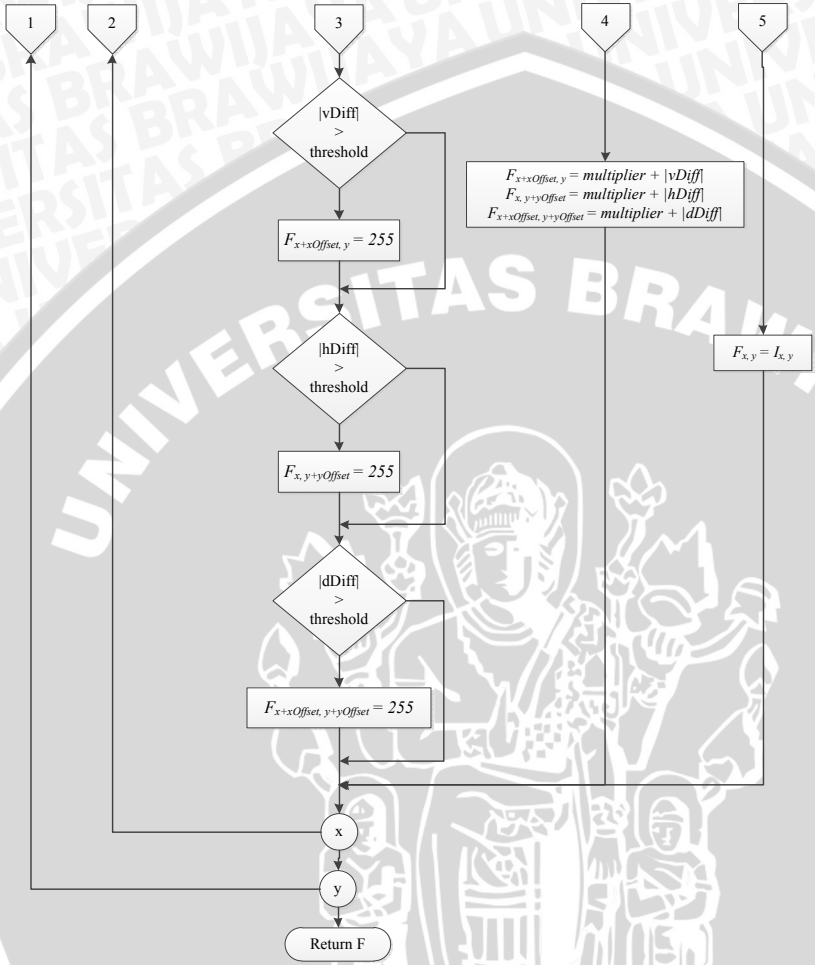
Gambar 3.7 Citra Asli dan Hasil Transformasi *Wavelet*

Adapun langkah-langkah yang dilakukan pada proses transformasi *Wavelet* dapat digambarkan menjadi *flowchart*seperti terlihat pada Gambar 3.8danGambar 3.9.



Gambar 3.8*Flowchart* Transformasi *Wavelet*


haarTransformPass
Start
INPUT
Matriks citra F,
threshold, level,
binarization
level = 2 level-1
h = tinggi matriks citra F
w = lebar matriks citra F
xOffset = h/2/level
yOffset = w/2/level
size = h * w
Multiplier = 0
I = matriks citra F
for int y=0 to w
for int x=0 to h
y<w/2/level
AND
x<h/2/level
N
Y
F x,y = (I2x,2y+I2x+1, 2y+I2x, 2y+1+I2x+1,2y+1)/4
vDiff =(I2x,2y+I2x+1, 2y) - (I2x,2y+1+I2x+1, 2y+1)
hDiff =(I2x,2y+I2x, 2y+1) - (I2x+1,2y+I2x+1, 2y+1)
dDiff =(I2x,2y+I2x+1, 2y+1) - (I2x+1,2y+I2x, 2y+1)
binarization
=
true
N
Y
1
2
3
4
5

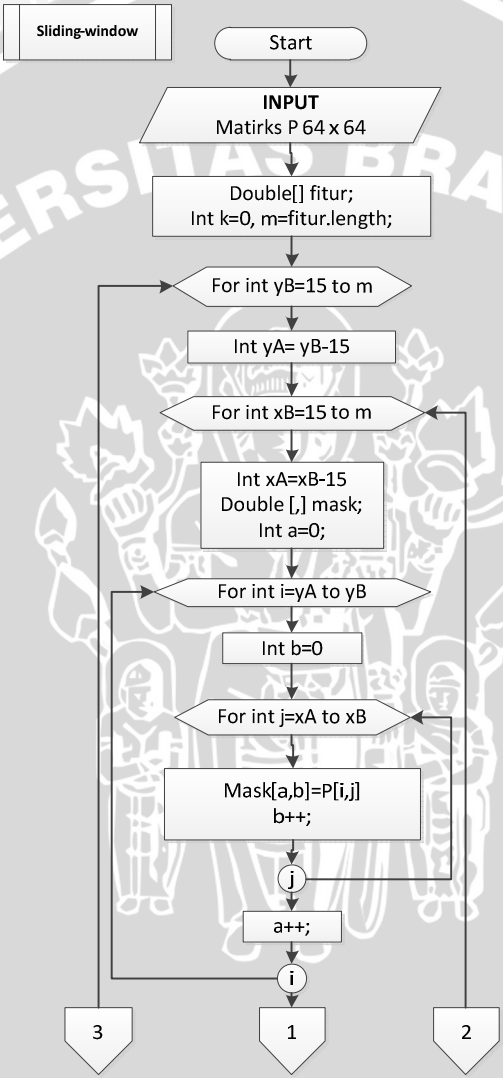


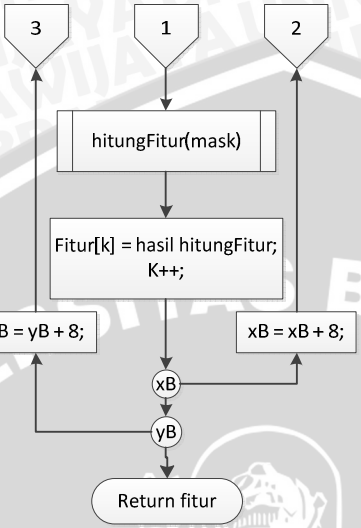
Gambar 3.9*Flowchart Haar Transform Pass*

3.3.1.4. Ekstraksi Fitur

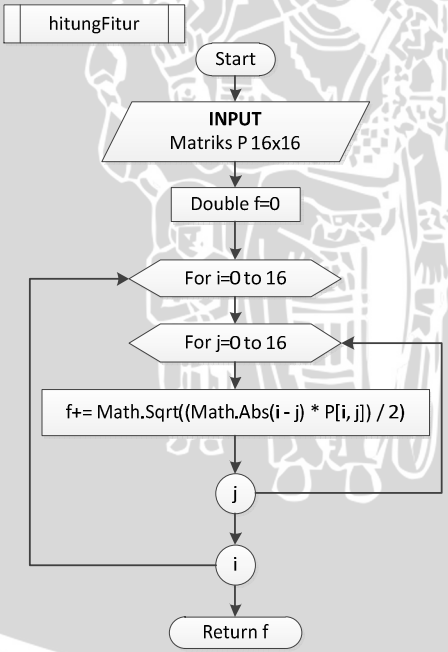
Proses ini bertujuan untuk mengekstraksi fitur yang masih terkandung pada sebuah citra. Pada penelitian ini, metode ekstraksi fitur yang digunakan adalah *Sliding Window Based Feature Extraction*dengan berdasarkan pada komponen *diagonal moment*. Hasil dari proses ini adalah fitur yang sudah terekstraksi dari citra dan tersimpan dalam bentuk vektor.

Adapun *flowchart* dari proses ekstraksi fitur menggunakan metode *sliding-window* dapat dilihat pada Gambar 3.10dan *flowchart* untuk menghitung nilai fitur dapat dilihat padaGambar 3.11.





Gambar 3.10 Proses *Sliding-Window* untuk Ekstraksi Fitur



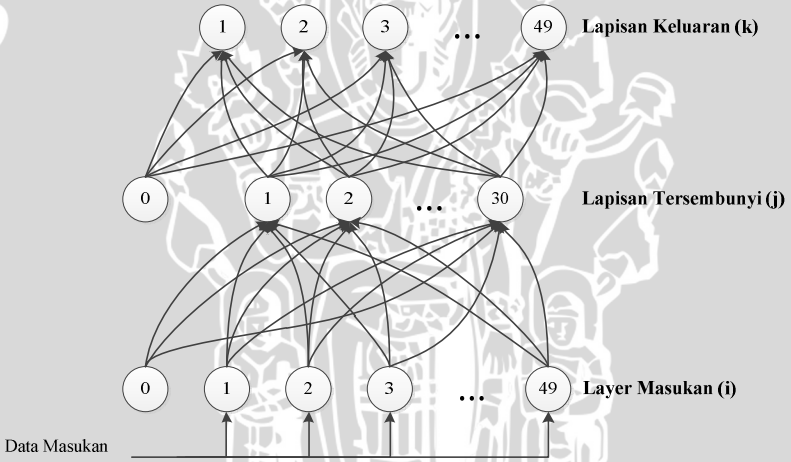
Gambar 3.11*Flowchart*Menghitung Fitur

3.3.1.5. Pelatihan Jaringan Syaraf Tiruan

Jaringan syaraf buatan digunakan untuk melatih komputer sehingga diharapkan komputer memiliki pengetahuan yang dapat digunakan untuk mengenali kedekatan pola antar citra.

Pada penelitian ini, metode pelatihan jaringan syaraf buatan yang digunakan adalah metode pelatihan*optical backpropagation*. Dan arsitektur jaringan yang digunakan adalah *multilayer perceptron* yang terdiri dari lapisan masukan, lapisan tersembunyi, dan lapisan keluaran. Setiap lapisan pada jaringan syaraf tiruan ini disusun oleh *neuron-neuron*. Lapisan masukan terdiri dari 49 *neuron*, lapisan tersembunyi terdiri dari 49 *neuron*, dan lapisan keluaran terdiri dari 30 *neuron*.

Adapaun arsitektur dari jaringan syaraf tiruan yang digunakan pada penelitian ini dapat dilihat pada Gambar 3.12.



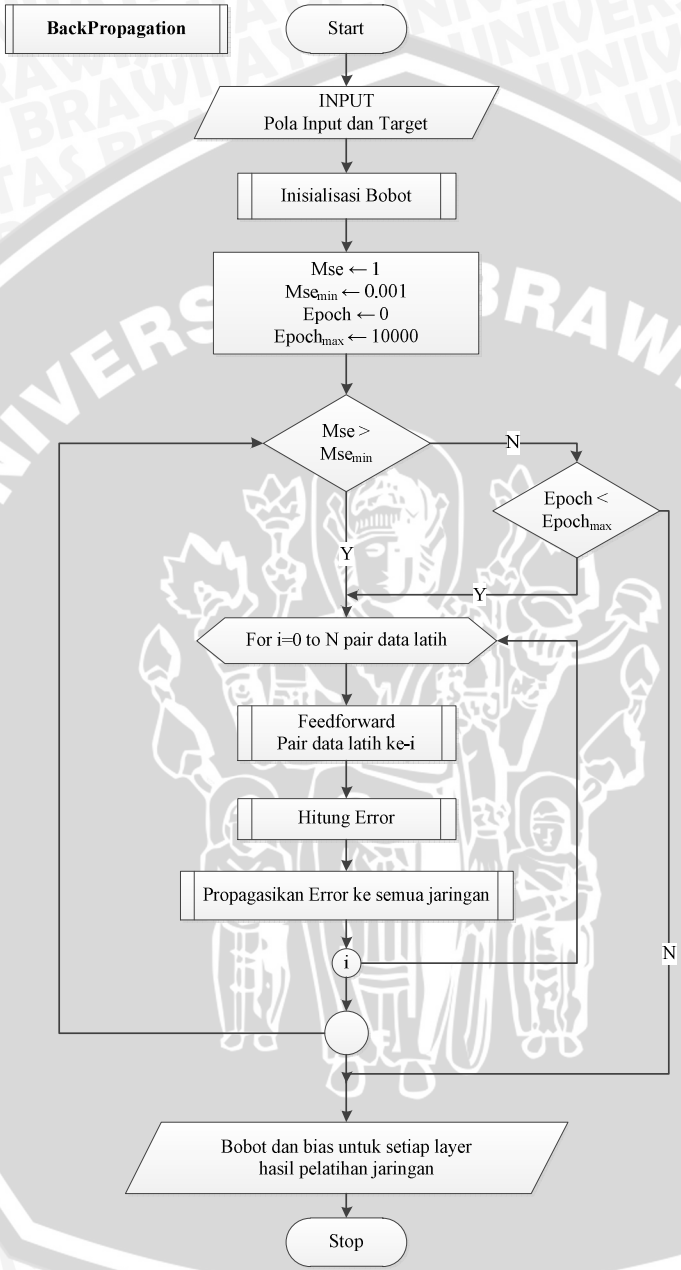
Gambar 3.12 Arsitektur Jaringan Syaraf Tiruan

Jaringan syaraf tiruan ini akan dilatih sampai mencapai konvergensi dengan minimum error sebesar 0.1, maksimum epoch sebesar 10000 dan *learning rate* (α)sebesar 0.01.

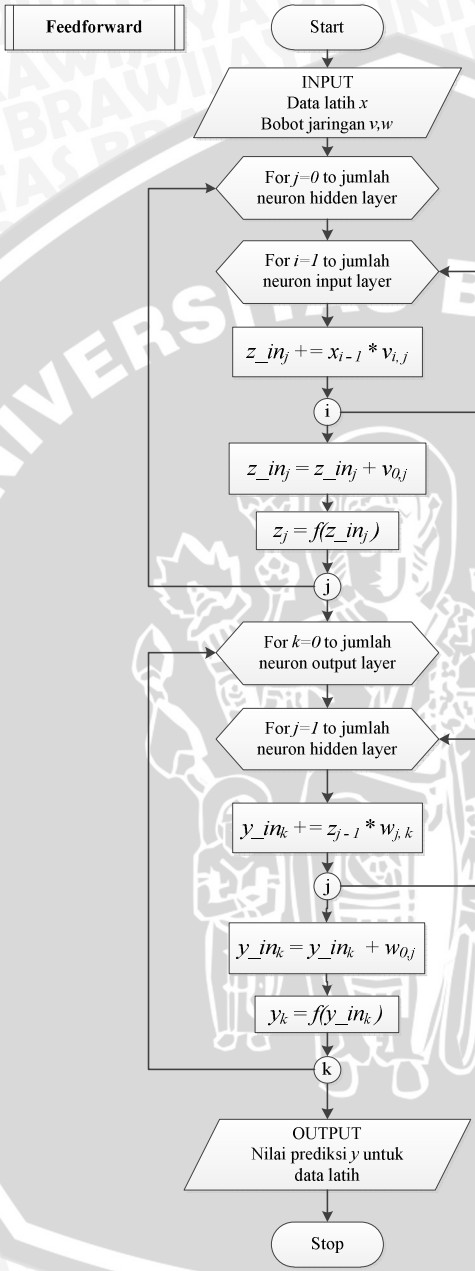
Langkah-langkah yang dilakukan pada proses pelatihan jaringan syaraf tiruan antara lain:

1. Menginisialisasi bobot awal untuk jaringan syaraf tiruan.
2. Menginisialisasi beberapa parameter yang digunakan untuk pelatihan JST, seperti bobot minimal yang diharapkan dan jumlah maksimal *epoch* yang diperbolehkan.
3. Melakukan perambatan maju (*feedforward*)
4. Menghitung galat yang dihasilkan dari hasil keluaran jaringan.
5. Memperbarui semua bobot untuk setiap lapisan pada jaringan syaraf tiruan berdasarkan galat yang diperoleh pada setiap *epoch*.
6. Mengulangi langkah ke-2 sampai ke-5 ketika kondisi untuk memberhentikan proses pelatihan belum terpenuhi.

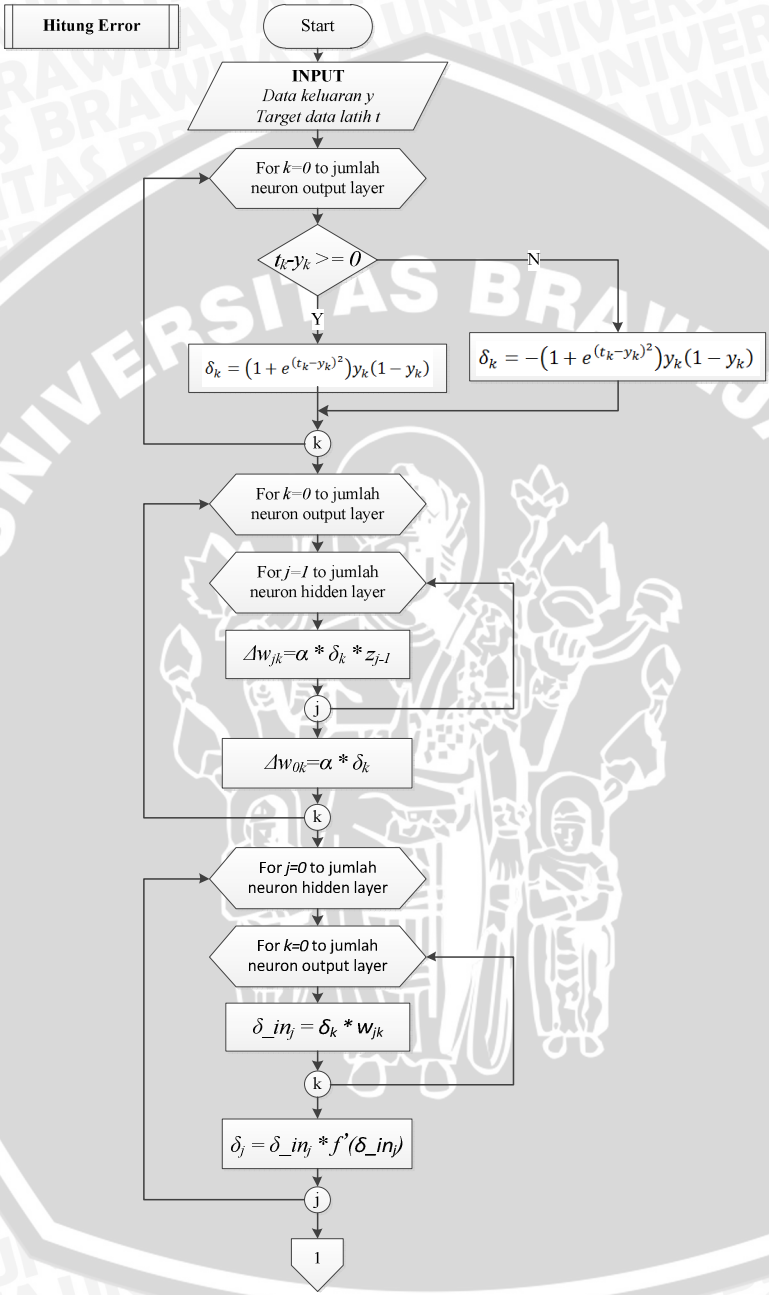
Langkah-langkah yang dilakukan pada proses pelatihan jaringan syaraf tiruan menggunakan metode *optical backpropagation* dapat digambarkan sebagai *flowchart* seperti yang terlihat pada Gambar 3.13 sampai Gambar 3.16.

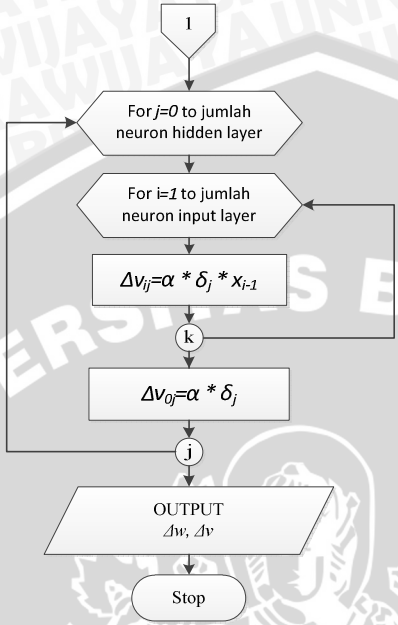


Gambar 3.13*Flowchart* Pelatihan Jaringan Syaraf Tiruan

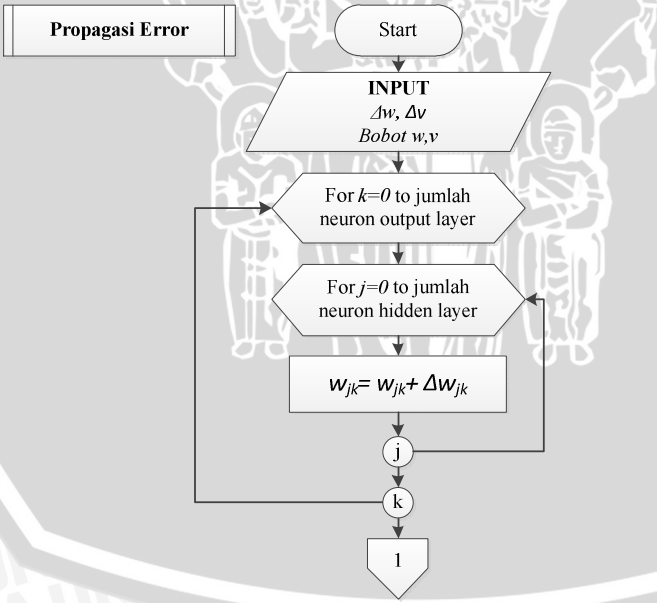


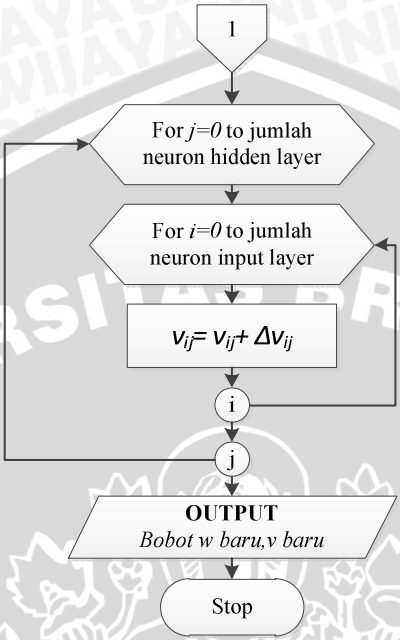
Gambar 3.14*Flowchart* Proses Perambatan Maju pada JST


Hitung Error
Start
INPUT
Data keluaran y
Target data latih t
For k=0 to jumlah neuron output layer
tk-yk >= 0
N
Y
$\delta_k = \left(1 + e^{(t_k - y_k)^2}\right) y_k(1 - y_k)$
$\delta_k = -\left(1 + e^{(t_k - y_k)^2}\right) y_k(1 - y_k)$
k
For k=0 to jumlah neuron output layer
For j=1 to jumlah neuron hidden layer
$\Delta w_{jk} = \alpha * \delta_k * z_{j-1}$
j
$\Delta w_{0k} = \alpha * \delta_k$
k
For j=0 to jumlah neuron hidden layer
For k=0 to jumlah neuron output layer
$\delta\_in_j = \delta_k * w_{jk}$
k
$\delta_j = \delta\_in_j * f'(\delta\_in_j)$
j
1



Gambar 3.15*Flowchart* Perhitungan Galat pada JST



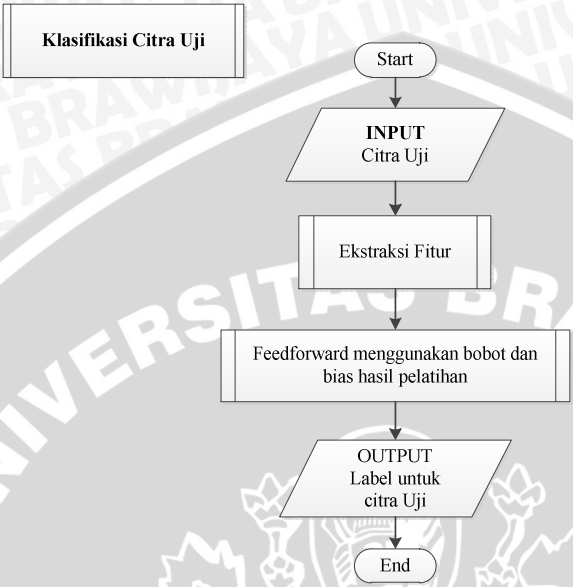


Gambar 3.16*Flowchart* Propagasi Galat pada JST

3.3.1.6. Klasifikasi Citra

Setelah komputer memiliki pengetahuan tentang pola-pola citra dan kelas-kelas yang sesuai dengan citra tersebut, maka proses selanjutnya adalah melakukan klasifikasi citra untuk citra-citra baru yang dimasukkan. Proses klasifikasi ini menentukan kelas yang sesuai terhadap sebuah citra berdasarkan konten yang dimiliki oleh citra tersebut.

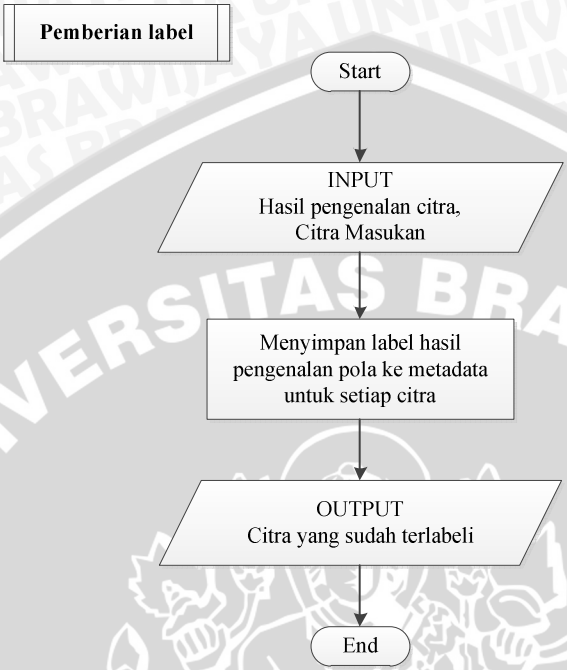
Adapun *flowchart* dari proses pengklasifikasian citra dapat dilihat pada Gambar 3.17.



Gambar 3.17*Flowchart* Klasifikasi Citra

3.3.1.7. Pemberian Label Citra

Setelah citra berhasil dikenali dan diklasifikasikan, maka proses selanjutnya adalah memberikan label pada citra sesuai dengan hasil pengklasifikasian tersebut. Label ini disimpan secara langsung pada *metadata* yang ada pada citra jpg. Proses pemberian label dapat dilihat padaGambar 3.18.



Gambar 3.18*Flowchart* Pemberian Label pada Citra

3.3.2. Perancangan Kelas

Dalam penelitian ini, untuk memecahkan permasalahan-permasalahan yang ada , digunakan sistem perancangan aplikasi berorientasi obyek. Beberapa komponen yang digunakan untuk membangun aplikasi ini antara lain 8kelas, 1 *interface*, dan 3*utility*.

Adapun kelas-kelas yang dirancang untuk membangun sistem ini antara lain:

1. **Kelas *HaarWavelet***
   Kelas ini digunakan untuk melakukan proses transformasi *Wavelet* untuk proses ekstraksi fitur pada citra.
2. **Kelas *ImageFeature***
   Kelas ini digunakan untuk merepresentasikan fitur dari sebuah citra. Kelas ini juga berfungsi untuk mengekstraksi fitur yang dimiliki oleh setiap citra.

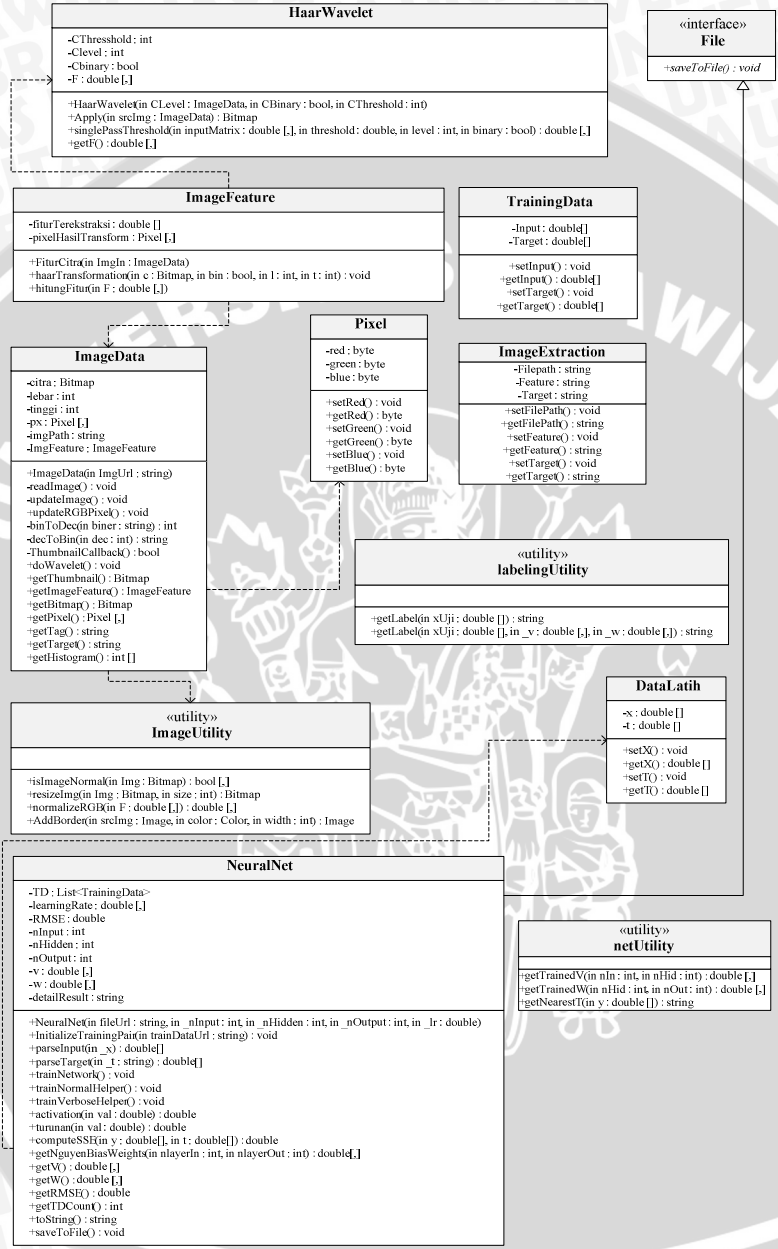
3. **Kelas *ImageData***
   Kelas ini digunakan untuk menyimpan semua informasi mengenai citra. Pada kelas ini, sebuah citra dibaca *pixel* per *pixel* sehingga citra tersebut dapat diolah pada proses pengenalan citra
4. **Kelas *Pixel***
   Kelas ini digunakan untuk merepresentasikan *pixel* yang dimiliki oleh citra. Kelas ini menyimpan komponen *red(R), green(G)* dan *Blue(B).*
5. **Kelas *ImageExtraction***
   Kelas ini digunakan untuk menyimpan informasi citra pada saat proses ekstraksi fitur citra dijalankan.
6. **Kelas *TrainingData***
   Kelas ini digunakan untuk merepresentasikan hasil ekstraksi fitur yang akan digunakan untuk melatih jaringan syaraf tiruan.
7. **Kelas *NeuralNet***
   Kelas ini digunakan untuk merepresentasikan arsitektur jaringan syaraf tiruan yang digunakan.
8. **Kelas DataLatih**
   Kelas ini digunakan untuk merepresentasikan data latih untuk masukan pada proses pelatihan jaringan syaraf tiruan.

Interface yang digunakan pada perancangan sistem ini adalah *interface**iFile***. *Interface* ini digunakan untuk melakukan operasi-operasi berkas seperti penyimpanan dan pembacaan berkas.

Pada perancangan kelas untuk sistem anotasi yang dikembangkan terdapat 3 u*tility* yang antara lain*utility* ***netUtility, ImageUtility,*** dan***labelingUtility*.** *Utility*ini digunakan untuk melakukan proses-proses statisyang digunakan untuk mengolah citra.

Diagram yang digunakan pada perancangan kelas ini adalah *UML (Unified Modeling Language) Class Diagram*. Adapun diagram UML yang digunakan pada perancangan kelas sistem anotasi ini dapat dilihat padaGambar 3.19.

.



Gambar 3.19 Diagram UML Perancangan Kelas Sistem Anotasi

3.3.3. Perancangan Penyimpanan Data

Dalam penelitian ini, semua data yang digunakan dan beberapa informasi lain disimpan dalam bentuk berkas dengan format *.ia* dan *.xml*. Untuk berkas format *.ia*, isi dari berkas ini didefinisikan sendiri untuk menyimpan beberapa data yang digunakan pada aplikasi ini, sedangkan untuk berkas format *.xml*, isi dari berkas ini mengikuti aturan penyimpanan data dalam bentuk *xml*.

Adapun data yang disimpan dalam bentuk *xml* ini adalah informasi vektor fitur untuk setiap citra yang akan diekstraksi.

Adapun data-data yang disimpan pada berkas.*ia*meliputi data hasil pelatihan pada jaringan syaraf tiruan, data tentang alamat direktori tempat citra disimpan, data labelpada citra dan beberapa data konfigurasi lainnya.

3.3.4. Perancangan Antarmuka

Perancangan antar muka dari perangkat lunak untuk anotasi citra otomatis ini dapat dilihat pada Gambar 3.20.

Gambar 3.20Perancangan Antar Muka

Adapun penjelasan dari bagian-bagian pada rancangan antarmuka untuk aplikasi ini adalah sebagai berikut:

1. *Toolbar*, untuk menempatkan beberapa perintah yang dapat dijalankan pada aplikasi.
2. *Groupbox* untuk menempatkan tombol-tombol untuk mengakses beberapa perintah dari aplikasi ini.
3. *Imagebox* yang digunakan untuk menampilkan citra berdasarkan *tag* yang dimiliki oleh citra.
4. *Groupbox* yang digunakan untuk menampilkan informasi tag yang dimiliki citra.
5. *Groupbox* jaringan syaraf tiruan untuk memasukkan nilai-nilai parameter yang dibutuhkan dan juga untuk menampilkan informasi dari proses pelatihan jaringan.
6. *Searchbox* yang digunakan untuk melakukan pencarian informasi yang terkandung pada citra.
7. *Informationbox* yang digunakan untuk menampilkan informasi *metadata* yang dimiliki oleh citra.
8. *Information box* yang digunakan untuk menampilkan informasi histogram dari setiap citra.

3.4. Contoh Perhitungan Manual

Langkah pertama yang dilakukan adalah ekstraksi fitur dari citra. Ekstraksi fitur dari citra menggunakan metode tranformasi *Wavelet* dan metode *Sliding-windows Based Feature Extraction.*

Pada contoh perhitungan ini tidak menggunakan data matriks citra asli melainkan matriks dengan ukuran yang lebih sederhana dengan ukuran 16 x 16. Hal ini untuk mempermudah dalam memahami konsep yang akan digunakan pada kasus nyata. Beberapa contoh perhitungan yang akan dibahas pada sub bab ini antara lain:

1. Perhitungan Ekstraksi Fitur Citra

Misalkan terdapat citra yang memiliki dimensi 16 x 16. Pada citra tersebut dilakukan pembacaan *pixel per pixel* sehingga didapatkan nilai matriks yang merepresentasikan nilai komponen warna (*red, green,* dan *blue*) pada setiap *pixel* pada citra tersebut.

Pengubahan Model Warna

Setelah dilakukan pembacaan citra dan didapatkan nilai komponen *red, green,* dan *blue* untuk setiap *pixel* dari citra tersebut. Maka langkah selanjutnya adalah menghitung nilai intensitas warna dengan menggunakan persamaan 2.3.

Misalkan pada sebuah *pixel* didapatkan nilai *Red(R)* = 143 *Green(B)* = 121, dan *Blue(B)* = 221, maka untuk mendapatkan nilai *intensity* dari pixel tersebut adalah sebagai berikut:

$$\frac{143 + 121 + 221}{3} = 162$$

Adapun hasil dari perhitungan intensitas warna untuk setiap *pixel* yang terdapat pada sebuah citra dapat dilihat pada Gambar 3.21.

Citra berukuran 16 x 16: 

Representasi Matriks F untuk intensitas citra

F_{ij}	0	1	2	3	4	5	6	7	...	15
0	230	246	240	230	228	227	234	226	...	249
1	232	253	239	251	254	247	250	253	...	254
2	221	254	235	222	232	237	216	234	...	240
3	228	246	237	239	226	226	241	220	...	254
4	239	253	139	143	239	231	238	226	...	216
5	224	243	124	60	188	236	188	217	...	254
6	237	242	243	153	78	123	84	83	...	246
7	229	249	230	215	86	73	71	73	...	251
8	230	248	221	186	108	45	39	52	...	249
9	236	254	199	158	155	161	66	60	...	253
10	215	248	198	188	226	241	127	109	...	242
11	235	254	228	223	234	226	190	154	...	253
12	220	241	224	222	221	231	213	156	...	253
13	226	234	216	224	211	222	142	121	...	231
14	230	233	209	218	210	217	162	185	...	235
15	226	251	237	234	230	231	237	233	...	249

Gambar 3.21Representasi Matriks dari Intensitas Citra Masukan

Transformasi *Wavelet*

Setelah didapatkan matriks yang merepresentasikan nilai intensitas, maka langkah selanjutnya adalah melakukan proses transformasi *Haar Wavelet* dari matriks tersebut sehingga didapatkan matriks T_{ij}sebagai hasil transformasi. Untuk melakukan transformasi ini digunakan proses *averaging* dan *differencing* pada sebuah daerah *pixel* dari citra asli seperti yang telah dijabarkan pada sub bab 2.7.

Berikut adalah contoh perhitungan untuk mendapatkan hasil dari proses *averaging* dan *differencing* pada daerah *pixel* yang pertama. Jika sebuah *pixel* pada sebuah citra dilambangkan dengan *px*, maka untuk menghitung nilai daerah *pixel* yang pertama yaitu $px_{0,0}$, $px_{0,1}$, $px_{1,0}$, dan $px_{1,1}$ adalah sebagai berikut:

$$
\begin{aligned}
averages &= \frac{px_{o,o} + px_{0,1} + px_{1,0} + px_{1,1}}{4} \\
&= \frac{230 + 246 + 232 + 253}{4} \\
&= 240.25 \\
&\approx 240 \\
hdiff &= (px_{o,o} + px_{0,1}) - (px_{1,0} + px_{1,1}) \\
&= (230 + 246) - (232 + 253) \\
&= -9 \\
vdiff &= (px_{o,o} + px_{1,0}) - (px_{0,1} + px_{1,1}) \\
&= (230 + 232) - (246 + 253) \\
&= -37 \\
ddiff &= (px_{o,1} + px_{1,0}) - (px_{0,0} + px_{1,1}) \\
&= (246 + 232) - (230 + 253) \\
&= -5
\end{aligned}
$$

Keterangan:
averages = hasil*averaging* untuk setiap *pixel* pada citra asli
hdiff = hasil *differencing* secara horizontal

vdiff = hasil *differencing* secara vertikal
ddiff = hasil *differencing* secara diagonal

Dari hasil perhitungan nilai *averaging*, *hdiff*, *vdiff*, dan *ddiff*, setiap nilai tersebut disusun padadaerah/kuadran tertentu yang terdapat pada matriks hasil transformasi *Wavelet*, yaitu daerah *averaging* (*Low-Low*) dan *detail coefficient* (*High-Low*, *Low-High*, dan *High-High*). Tetapi sebelum disusun pada daerah-daerah tersebut, nilai-nilai hasil perhitungan *detail coefficient* dijadikan nilai absolut. Hal ini dikarenakan sebuah citra digital tidak memiliki komponen warna negatif sehingga dilakukan pengubahan nilai pada *detail coefficient* menjadi nilai absolut.

Dari contoh perhitungan yang telah dijabarkan sebelumnya, nilai *averages* dimasukkan pada indeks ke [0; 0] pada matriks baru, nilai *hdiff* dimasukkan pada indeks ke [0; 8] pada matriks baru, nilai *vdiff* dimasukkan pada indeks ke [8; 0] pada matriks baru, dan nilai *ddiff* dimasukkan pada indeks ke [8; 8] pada matriks baru. Adapun hasil semua perhitungan dari proses transformasi citra ini adalah matriks hasil transformasi T_{ij} yang dapat dilihat padaGambar 3.22.

Bagian yang diberi warna biru adalah matriks hasil transformasi yang akan digunakan untuk proses ekstraksi fitur menggunakan metode *Sliding-window Based Feature Extraction*.

T_{ij}	0	...	8	9	10	11	12	13	14	15
0	240	...	9	20	46	43	31	38	14	7
1	237	...	1	19	17	11	1	3	44	15
2	240	...	25	98	46	59	118	65	165	106
3	239	...	1	49	42	23	74	36	2	3
4	242	...	12	50	163	35	44	21	27	2
5	238	...	26	65	7	108	1	5	3	3
6	230	...	1	6	19	106	39	6	33	18
7	235	...	14	44	34	123	54	27	41	24
8	37	...	5	22	6	11	5	6	14	3
9	51	...	15	15	5	39	5	1	26	43
10	33	...	5	68	56	41	106	121	1	30

11	25	...	15	75	58	3	94	76	16	13
12	36	...	0	6	69	19	26	25	11	6
13	52	...	14	5	23	18	67	15	21	19
14	29	...	13	10	1	36	29	32	25	26
15	28	...	22	12	6	27	24	23	7	4

Gambar 3.22 Matriks T Hasil Transformasi *Wavelet*

Ekstraksi Fitur Menggunakan Sliding-Window

Dari matriks hasil transformasi yang berukuran 16 x 16, diambil sub matriks berukuran 8 x 8. Sub matriks ini terletak pada bagian pojok matriks hasil transformasi atau pada proses transformasi *Wavelet* disebut juga dengan istilah *sub-band* HH1. Matriks ini dapat dilihat pada Gambar 3.23.

5	22	6	11	5	6	14	3
15	15	5	39	5	1	26	43
5	68	56	41	106	121	1	30
15	75	58	3	94	76	16	13
0	6	69	19	26	25	11	6
14	5	23	18	67	15	21	19
13	10	1	36	29	32	25	26
22	12	6	27	24	23	7	4

Gambar 3.23*Subband* HH1 untuk Ekstraksi Fitur

Kemudian dihitung nilai fitur untuk citra tersebut menggunakan persamaan 2.8. Sebagai contoh akan dihitung nilai pertama dan kedua untuk fitur dari citra tersebut.

$$Fitur_1 = (\sqrt{0 * 5/2} + \sqrt{1 * 22/2} + \sqrt{1 * 15/2} + \sqrt{0 * 15/2})$$
$$= 6.055237578$$

$$Fitur_2 = (\sqrt{0 * 22/2} + \sqrt{1 * 6/2} + \sqrt{1 * 15/2} + \sqrt{0 * 5/2})$$
$$= 4.470663595$$

Adapun hasil perhitungan untuk semua fitur pada citra tersebut dapat dilihat padaTabel 3.1.

Tabel 3.1Hasil Perhitungan Ekstraksi Fitur pada Citra.

Fitur ke-	Nilai fitur
1	6.055238
2	4.470664
3	3.926347
4	5.997019
5	3.313190
6	3.352858
7	4.830296
8	4.319752
9	7.412091
10	9.707383
11	6.108831
12	7.987217
13	11.383726
14	5.343916
15	8.569565
16	11.415227
17	9.912857
18	8.504855
19	14.633829
20	6.871521
21	6.701410
22	6.123724
23	7.117216
24	7.098415
25	9.937862
26	9.769965
27	6.363961
28	4.894718
29	4.377802
30	7.454809
31	6.473372
32	6.605551

33	9.323452
34	5.083821
35	4.972421
36	4.130649
37	5.627233
38	3.707107
39	10.030559
40	6.546499
41	7.240370
42	6.617741
43	5.552693
44	3.156597
45	5.974691
46	7.482121
47	7.464102
48	6.926699
49	5.476380

2. Perhitungan pada Pelatihan Jaringan Syaraf Tiruan

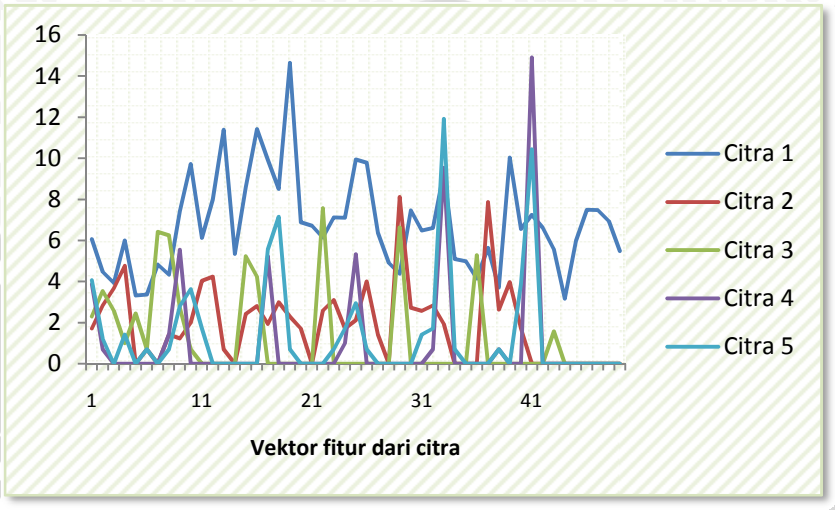
Dari hasil ekstraksi fitur yang sudah dilakukan, maka akan didapatkan fitur dari citra berupa vektor-vektor. Nilai vektor tersebut akan dijadikan data latih pada JST.

Misalkan terdapat 5 citra yang sudah diekstraksi fiturnya, maka 5 vektor tersebut akan digunakan sebagai data latih pada proses pelatihan jaringan syaraf tiruan.Semua citra yang akan dijadikan data latih dimasukkan ke dalam kelas-kelas tertentu yang sudah ditentukan. Kelas-kelas inilah yang akan dijadikan target pelatihan pada pelatihan jaringan syaraf tiruan.

Contoh data hasil ekstraksi fitur yang akan digunakan sebagai data latih dapat dilihat padaTabel 3.2. Setiap nilai fitur tersebut merepresentasikan informasi yang terkandung pada citra. Target pelatihan untuk masing-masing data latih dapat dilihat padaTabel 3.3,sedangkan grafik yang menggambarkan vektor hasil proses ekstraksi fitur tersebut dapat dilihat pada Gambar 3.24.

Tabel 3.2Data Masukan untuk Pelatihan JST

Id Citra	Sekuens Fitur
1	6.055238; 4.470664; 3.926347; 5.997019; 3.313190; 3.352858; 4.830296; 4.319752; 7.412091; 9.707383; 6.108831; 7.987217; 11.38372; 5.343916; 8.569565; 11.41522; 9.912857; 8.504855; 14.63382; 6.871521; 6.701410; 6.123724; 7.117216; 7.098415; 9.937862; 9.769965; 6.363961; 4.894718; 4.377802; 7.454809; 6.473372; 6.605551; 9.323452; 5.083821; 4.972421; 4.130649; 5.627233; 3.707107; 10.03055; 6.546499; 7.240370; 6.617741; 5.552693; 3.156597; 5.974691; 7.482121; 7.464102; 6.926699; 5.476380;
2	1.707107; 2.828427; 3.674235; 4.760279; 0.000000; 0.707107; 0.000000;1.414214; 1.224745; 2.000000; 4.030629; 4.236068; 0.707107; 0.000000;2.414214; 2.805884; 1.931852; 2.995352; 2.288246; 1.707107; 0.000000;2.581139; 3.095574; 1.707107; 2.121320; 4.000000; 1.414214; 0.000000;8.116769; 2.732051; 2.577935; 2.828427; 1.931852; 0.000000; 0.000000;0.000000; 7.873670; 2.638958; 3.968119; 1.707107; 0.000000; 0.000000;0.000000; 0.000000; 0.000000; 0.000000; 0.000000; 0.000000; 0.000000;
3	2.288246; 3.535534; 2.577935; 1.000000; 2.449490; 0.707107; 6.422493; 6.244485; 2.732051; 0.707107; 0.000000; 0.000000; 0.000000; 0.000000; 5.236068; 4.240370; 0.000000; 0.000000; 0.000000; 0.000000; 0.000000; 7.571844; 0.000000; 0.000000; 0.000000; 0.000000; 0.000000; 0.000000; 6.636314; 0.000000; 0.000000; 0.000000; 0.000000; 0.000000; 0.000000; 5.277917; 0.000000; 0.000000; 0.000000; 0.000000; 0.000000; 0.000000; 1.581139; 0.000000; 0.000000; 0.000000; 0.000000; 0.000000; 0.000000;
4	3.870829; 0.707107; 0.000000; 0.000000; 0.000000; 0.000000; 0.000000; 1.414214; 5.549510; 0.000000; 0.000000; 0.000000; 0.000000; 0.000000; 0.000000; 0.000000; 5.236068; 0.000000; 0.000000; 0.000000; 0.000000; 0.000000; 0.000000; 1.000000; 5.318275; 0.000000; 0.000000; 0.000000; 0.000000; 0.000000; 0.000000; 0.707107; 9.525273; 0.000000; 0.000000; 0.000000; 0.000000; 0.707107; 0.000000; 0.000000; 14.90347; 0.000000; 0.000000; 0.000000; 0.000000; 0.000000; 0.000000; 0.000000;
5	4.059965; 1.224745; 0.000000; 1.414214; 0.000000; 0.707107; 0.000000; 0.707107; 2.805884; 3.622583; 1.707107; 0.000000; 0.000000; 0.000000; 0.000000; 0.000000; 5.554787; 7.140825; 0.707107; 0.000000; 0.000000; 0.000000; 0.707107; 1.707107; 2.943175; 0.707107; 0.000000; 0.000000; 0.000000; 0.000000; 1.414214; 1.707107; 11.91010; 0.707107; 0.000000; 0.000000; 0.000000; 0.707107; 0.000000; 3.869384; 10.43212; 0.000000; 0.000000; 0.000000; 0.000000; 0.000000; 0.000000; 0.000000;



Gambar 3.24 Sebaran Nilai Fitur Citra

Tabel 3.3 Data*Target* untuk Pelatihan JST

Id Citra	Target pelatihan
1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 01
2	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 10
3	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 11
4	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 100
5	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 101

Menentukan parameter awal untuk pelatihan jaringan *optical backpropagation* yang meliputi:

- Jumlah *neuron* pada lapisan masukan: 49
- Jumlah *neuron* pada lapisan tersembunyi: 49
- Jumlah *neuron* pada lapisan keluaran: 30
- *Learning rate* (α) = 0.1
- $Error_{min}$ = 0.001
- $Epoch_{max}$ = 10000

Inisialisasi Bobot Awal

Pada proses pelatihan jaringan syaraf tiruan, hal pertama yang dilakukan adalah inisialisasi bobot. Proses inisialisasi bobot pada penilitan ini menggunakan metode *Nguyen-Widrow* (seperti yang telah dijelaskan pada sub-bab 2.10.1). Untuk menginisialisasi bobot menggunakan metode ini, digunakan persamaan 2.9, 2.10, dan 2.11.

Pertama, bobot awal diinisialisasi dengan cara membangkitkan nilai acak antara -0.5 sampai 0.5. Hasil dari pembangkitan ini dapat dilihat pada Gambar 3.25untuk bobot awal lapisan tersembunyi (*w*) dan Gambar 3.26untuk bobot awal lapisan keluaran (*v*). Karena keterbatasan tempat, maka isi matriks bobot tidak ditampilkan semua.

j / i	1	2	3	4	5	….	49
1	-0,418	-0,411	0,0426	-0,376	0,1888	….	-0,131
2	0,1088	0,3782	0,0948	0,1327	-0,274	….	-0,218
3	-0,367	0,0313	0,1723	-0,064	-0,292	….	0,1895
4	-0,399	-0,164	-0,198	0,2761	0,4664	….	-0,476
5	-0,157	0,363	0,1156	0,2145	-0,206	….	0,0214
….	….	….	….	….	….	….	….
49	-0,432	-0,273	0,4946	-0,057	-0,347	….	-0,037

Gambar 3.25 Bobot Awal Lapisan Tersembunyi

k / j	1	2	3	4	5	….	30
1	-0,247	0,4406	0,0288	-0,092	-0,365	….	-0,05
2	0,2809	0,4786	0,3856	0,0713	-0,109	….	-0,49
3	0,1851	0,0616	0,2681	0,3687	-0,013	….	0,366
4	-0,341	0,1518	-0,063	-0,477	-0,087	….	0,268
5	0,4361	-0,244	-0,157	0,2849	0,4441	….	-0,46
….	….	….	….	….	….	….	….
49	0,1229	0,0048	0,3722	-0,34	-0,391	….	0,435

Gambar 3.26 Bobot Awal Lapisan Keluaran

Setelah bobot awal didapatkan, langkah selanjutnya adalah menghitung nilai beta (β) dengan menggunakan persamaan 2.9

$\beta = 0.7 * \sqrt[49]{49}$

$= 0.7 * 1{,}083$

$= 0{,}758$

Kemudian nilai $\|V_i\|$ dan $\|W_j\|$ dihitung untuk masing-masing baris pada bobot lama menggunakan persamaan 2.10. Sebagai contoh akan dilakukan perhitungan untuk baris ke-0 dan ke-49 (masing-masing untuk bobot *v* dan *w*).

$$\|V_0\| = \sqrt{\sum_{i=0}^{49} {V_{i\,0}}^2}$$

$$= 2{,}04584305574616$$

$$\|V_{49}\| = \sqrt{\sum_{i=0}^{49} {V_{i\,49}}^2}$$

$$= 2{,}11954393093567$$

$$\|W_0\| = \sqrt{\sum_{i=0}^{49} {V_{i\,0}}^2}$$

$$= 1{,}60496860943744$$

$$\|W_{49}\| = \sqrt{\sum_{i=0}^{49} {V_{i\,49}}^2}$$

$$= 1{,}47152658775101$$

Hasil perhitungan $\|V_i\|$ dan $\|W_j\|$ tersebut digunakan untuk menghitung nilai bobot baru untuk setiap baris pada bobot lama.

Untuk melakukan perubahan bobot baru ini digunakan persamaan 2.11.

Setelah dilakukan perhitungan, maka didapatkan bobot dan bias baru untuk *v* dan *w*. Gambar 3.27menunjukkan bobot baru untuk v_{ij}, Gambar 3.28menunjukkan bias baru untuk v_{ij}, Gambar 3.29menunjukkan bobot baru untuk w_{jk} dan Gambar 3.30menunjukkan bias baru untuk w_{jk}. Karena ketebatasan tempat, maka isi matriks bobotdan bias tidak ditampilkan semua.

j / i	1	2	3	4	5	….	49
1	-0,155	-0,152	0,015	-0,13	0,06	….	-0,048
2	0,041	0,1446	0,036	0,050	-0,10	….	-0,083
3	-0,141	0,0121	0,066	-0,02	-0,11	….	0,073
4	-0,152	-0,062	-0,075	0,105	0,17	….	-0,181
5	-0,055	0,1267	0,0404	0,074	-0,07	….	0,0075
….	….	….	….	….	….	….	….
49	-0,154	-0,098	0,1769	-0,02	-0,12	….	-0,013

Gambar 3.27 Bobot Akhir Lapisan Tersembunyi

j / i	1	2	3	4	5	….	49
0	0,1766	0,0025	0,220	0,6045	-0,46	**….**	-0,653

Gambar 3.28 Bias Akhir Lapisan Tersembunyi

k / j	1	2	3	4	5	….	30
1	-0,115	0,206	0,013	-0,043	-0,171	….	-0,02
2	0,130	0,2218	0,178	0,033	-0,051	….	-0,23
3	0,085	0,0283	0,123	0,169	-0,006	….	0,168
4	-0,149	0,0662	-0,028	-0,208	-0,038	….	0,117
5	0,2253	-0,126	-0,081	0,1472	0,2294	….	-0,24
….	….	….	….	….	….	….	….
49	0,0627	0,0024	0,1898	-0,173	-0,2	….	0,221

Gambar 3.29Bobot Akhir Lapisan Keluaran

j \ k	1	2	3	4	5	….	30
0	0,1692	0,5129	0,6709	-0,64	0,737	….	0,2669

Gambar 3.30 Bias Akhir Lapisan Keluaran

Melatih jaringan dengan *Optical Backpropagation*

Proses pelatihan jaringan syaraf tiruan menggunakan metode *optical backpropagation* dilakukan dengan beberapa langkah, antara lain:

1. Setiap unit di lapisan masukan$(X_i, i = 1 \dots n)$menerima sinyal input dan sinyal tersebut diteruskan ke semua unit di lapisan tersembunyi.
2. Semua sinyal dari lapisan masukan dikalikan dengan bobot yang sudah dibangkitkan (v_{ij}) dan setiap hasil perkalian tersebut dijumlahkan dan kemudian dimasukkan pada setiap unit di lapisan tersembunyi $(Z_j, j = 1 \dots p)$ sehingga didapatkan nilai *z_in* seperti pada tabel 3.4. Operasi yang dilakukan pada langkah ini menggunakan persamaan 2.17.Sebagai contoh, akan ditunjukkan perhitungan untuk z_in_1.

$$z\_in_1 = v_{01} + \sum_{i=1}^{n} x_i v_{i1}$$

= 0,484 + (-0,2965)
= 0,1875

Hasil perhitungan untuk nilai *z_in* ini ditunjukkan pada Tabel 3.4.

Tabel 3.4Hasil Perhitungan *z_in*

j	z_in_j
1	0,1875
2	0,2445
3	0,2001
4	0,3877
5	-0,373

...	...
49	-0,707

Kemudian dihitung hasil dari fungsi aktivasinya sesuai dengan fungsi aktivasi yang digunakan yaitu fungsi aktivasi*binary sigmoid* sesuai dengan persamaan 2.18. Sebagai contoh, akan ditunjukkan perhitungan untuk z_1

$$z_1 = f(z\_in_1)$$
$$= \frac{1}{1 + e^{-0.1 * 0,1875}}$$
$$= 0,50468$$

Hasil perhitungan untuk nilai zini ditunjukkan padaTabel 3.5. Nilai *z* ini yang akan diteruskan ke semua unit pada lapisan keluaran.

Tabel 3.5 Hasil Perhitungan *z*

j	z_j
1	0,5047
2	0,5061
3	0,505
4	0,5097
5	0,4907
...	...
49	0,4823

3. Semua sinyal dari lapisan tersembunyi dikalikan dengan bobot yang sudah dibangkitkan (w_{jk}) dan setiap hasil perkalian tersebut dijumlahkan dan kemudian dimasukkan pada setiap unit di lapisan keluaran ($Y_k,\ k = 1 \dots m$) sesuai dengan persamaan 2.19 sehingga didapatkan nilai *y_in* seperti padaTabel 3.6. Sebagai contoh akan ditunjukkan perhitungan untuk mendapatkan nilai y_in_1.

$$y\_in_1 = w_{01} + \sum_{j=1}^{p} z_j w_{j1}$$

= 0,1692 + 0,6427
= 0,8119

Tabel 3.6 Hasil Perhitungan y_in

k	y_in_k
1	0,8119
2	1,1201
3	0,5747
4	-0,565
5	0,6592
...	...
30	0,0757

Setelah didapatkan nilai *y_in*, maka nilai ini dimasukkan pada fungsi aktivasi *binary sigmoid* sesuai dengan persamaan 2.20 sehingga didapatkan nilai dari hasil aktivasi seperti pada Tabel 3.7.

Tabel 3.7 Hasil Perhitungan y

k	y_k
1	0,5203
2	0,528
3	0,5144
4	0,4859
5	0,5165
...	...
30	0,5019

4. Setiap unit keluaran $(Y_k, k = 1 \dots m)$ menerima pola target sesuai dengan pola masukan saat pelatihan dan dihitung informasi *error* (δ_k) menggunakan persamaan 2.31 atau 2.32. Sebagai contoh, akan ditunjukkan perhitungan untuk mendapatkan nilai δ_1. Sesuai dengan persamaan 2.31 dan 2.32,untuk menghitung δ_1, dilakukan pengecekan terlebih dahulu apakah $(t_k - y_k) \geq 0$ atau $(t_k - y_k) < 0$. Setelah dilakukan pengecekan, ternyata nilai $(t_1 - y_1) < 0$, sehingga untuk menghitung nilai $\delta_1$digunakan persamaan 2.31.

$$\delta_1 = (1 + e^{(t_1 - y_1)^2}) y_1 (1 - y_1)$$
$$= \left(1 + e^{(-0{,}5203)^2}\right) * 0{,}5203 * (1 - 0{,}5203)$$
$$= -0{,}577$$

Hasil perhitungan *error* untuk setiap unit pada lapisan keluaran ditunjukkan pada Tabel 3.8.

Tabel 3.8 Hasil Perhitungan δ

K	δ_k
1	-0,577
2	-0,579
3	-0,575
4	-0,566
5	-0,576
...	...
30	0,5704

Kemudian menghitung nilai Δw yang akan digunakan untuk memperbaiki bobot pada lapisan tersembunyi (w_{jk}).

Untuk menghitung nilai koreksi bobot, digunakan persamaan 2.22. Sedangkan untuk menghitung nilai koreksi bias digunakan persamaan 2.23.

Sebagai contoh, akan dilakukan penghitungan untuk nilaiperbaikan bobot Δw_{11} dan nilai perbaikan biasΔw_{01}.

$$\Delta w_{jk} = \alpha \delta_k z_j$$
$$\Delta w_{11} = \alpha \delta_1 z_1$$
$$= 0{,}1 * -0{,}577 * 0{,}5047$$
$$= -0{,}0291$$

$$\Delta w_{01} = \alpha \delta_1$$
$$= 0.1 * -0{,}577$$
$$= -0{,}0577$$

Hasil perhitungan nilai Δw_{jk}untuk perbaikan bobot ditunjukkan padaGambar 3.31. Sedangkan hasil perhitungan nilai Δw_{0k}untuk perbaikan bias ditunjukkan padaGambar 3.32.

k / j	1	2	3	4	5	….	30
1	-0,029	-0,029	-0,029	-0,029	-0,029	….	0,0288
2	-0,029	-0,029	-0,029	-0,029	-0,029	….	0,0289
3	-0,029	-0,029	-0,029	-0,029	-0,029	….	0,0288
4	-0,029	-0,029	-0,029	-0,029	-0,029	….	0,0291
5	-0,028	-0,028	-0,028	-0,028	-0,028	….	0,028
….	….	….	….	….	….	….	….
49	-0,028	-0,028	-0,028	-0,027	-0,028	….	0,0275

Gambar 3.31 Hasil Perhitungan Perubahan Bobot Δw

k / j	1	2	3	4	5	….	30
0	-0,058	-0,058	-0,058	-0,057	-0,058	….	0,057

Gambar 3.32 Hasil Perhitungan Perubahan BiasΔw

5. Setiap unit tersembunyi $(Z_j, j = 1 \dots p)$menghitung informasi*error* dari lapisan keluaran dengan cara mencari nilai δ_in terlebih dahulu kemudian mengalikannya dengan turunan dari fungsi aktivasi yang digunakan. Untuk mendapatkan nilai δ_in, digunakan persamaan 2.24. Sebagai contoh, akan dilakukan perhitungan untuk δ_in_1 pada lapisan tersembunyi.

$$\delta\_in_1 = \sum_{k=1}^{m} \delta_k w_{1k}$$

= -0,2790

Hasil dari perhitungan $\delta_in$pada lapisan tersembunyi ini dapat dilihat pada Tabel 3.9.

Tabel 3.9 Hasil Perhitungan δ_in

j	δ_in_j
1	-0,2790
2	-0,3385
3	-0,2974
4	0,2580
5	0,7130
...	...
49	-0,1587

Selanjutnya dikalikan dengan turunan dari fungsi aktivasinya menggunakan persamaan 2.25, sehingga didapatkan nilai informasi *error* seperti pada Tabel 3.10.

Tabel 3.10 Hasil Perhitungan δ

j	δ_j
1	-0,0697
2	-0,0846
3	-0,0743
4	0,0645
5	0,1780
...	...
49	-0,0396

Kemudian menghitung nilai koreksi bobot (Δv) yang akan digunakan untuk memperbaiki nilai bobot v_{ij}menggunakan persamaan 2.26 dan menghitung nilai koreksi bias yang akan digunakan untuk memperbaiki nilai bias v_{0j}menggunakan persamaan 2.27. Sebagai contoh, akan dilakukan penghitungan untuk nilaiperbaikan bobot Δv_{11} dan nilai perbaikan bias Δv_{01}.

$$\Delta v_{ij} = \alpha \delta_j x_i$$

$$\Delta v_{11} = \alpha \delta_1 x_1$$

$$= 0.1 * -0{,}577 * 0.000055$$

$$= -0{,}0000031735$$

$$\Delta v_{0j} = \alpha \delta_j$$

$$\Delta v_{01} = \alpha\delta_1$$
$$= 0.1 * -0{,}577$$
$$= -0{,}0577$$

Hasil perhitungan nilai Δv_{ij} untuk perbaikan bobot ditunjukkan padaGambar 3.33. Sedangkan hasil perhitungan nilai Δv_{0j} untuk perbaikan bias ditunjukkan padaGambar 3.34.

i \ j	1	2	3	4	5	….	49
1	-0,000	-0,000	-0,000	0,000	0,000	….	-0,004
2	-0,000	-0,000	-0,000	0,000	0,000	….	-0,000
3	-0,002	-0,002	-0,002	0,0015	0,0043	….	-0,0002
4	-0,000	-0,001	-0,000	0,000	0,0023	….	-0,000
5	-0,000	-0,000	-0,000	0,000	0,001	….	-0,000
….	….	….	….	….	….	….	….
49	-0,000	-0,000	-0,000	0,000	0,0009	….	-0,0001

Gambar 3.33 Hasil Perhitungan Perubahan Bobot Δv

i \ j	1	2	3	4	5	….	49
0	-0,007	-0,008	-0,007	0,0065	0,0178	….	-0,004

Gambar 3.34 Hasil Perhitungan Perubahan Bias Δv

6. Setiap unit keluaran $(Y_k, j = 1 \dots m)$ memperbarui nilai bias dan bobot $(j = 0, \dots, p)$menggunakan persamaan 2.28. Sebagai contoh akan dilakukan penghitungan untuk nilai w_{jk} baru. Adapun hasil dari perbaikan bobot dan bias pada lapisan keluaran dapat dilihat pada Gambar 3.35danGambar 3.36.

$$w_{jk}(new) = w_{jk}(old) + \Delta w_{jk}$$
$$w_{11}(new) = w_{11}(old) + \Delta w_{11}$$
$$= -0{,}115 + -0{,}029$$
$$= -0{,}144$$

j \ k	1	2	3	4	5	….	49
1	-0,145	0,1768	-0,016	-0,072	-0,199	….	0,0024
2	0,101	0,1926	0,149	0,004	-0,079	….	-0,202
3	0,055	-0,0000	0,094	0,140	-0,035	….	0,1974
4	-0,178	0,0367	-0,057	-0,237	-0,067	….	0,1463
5	0,197	-0,1542	-0,109	0,119	0,201	….	-0,214
….	….	….	….	….	….	….	….
49	0,034	-0,025	0,162	-0,201	-0,227	….	0,2493

Gambar 3.35 Bobot baru *w*

j \ k	1	2	3	4	5	….	49
0	0,1116	0,4551	0,613	-0,70	0,679	….	0,324

Gambar 3.36 Bias Baru *w*

7. Setiap unit tersembunyi $(Z_j, j = 1 \dots p)$ memperbarui nilai bias dan bobot $(i = 0, \dots, n)$ menggunakan persamaan 2.29. Adapun hasil dari perbaikan bobot dan bias pada lapisan tersembunyi dapat dilihat pada Gambar 3.37danGambar 3.38.

$$v_{ij}(new) = v_{ij}(old) + \Delta v_{ij}$$
$$v_{11}(new) = v_{11}(old) + \Delta v_{11}$$
$$= -0{,}155 + -3{,}83661044462194 * 10^{-7}$$
$$= -0{,}155$$

i \ j	1	2	3	4	5	….	30
1	-0,155	-0,152	0,015	-0,139	0,069	….	-0,048
2	0,041	0,1442	0,035	0,051	-0,104	….	-0,084
3	-0,143	0,01	0,064	-0,023	-0,108	….	0,072
4	-0,153	-0,063	-0,076	0,106	0,179	….	-0,182
5	-0,055	0,1262	0,039	0,075	-0,071	….	0,007
….	….	….	….	….	….	….	….

49	-0,155	-0,098	0,176	-0,02	-0,123	….	-0,014

Gambar 3.37 Bobot Baru *v*

j / i	**1**	**2**	**3**	**4**	**5**	**….**	**30**
0	0,169	-0,006	0,212	0,610	-0,44	….	-0,657

Gambar 3.38 Bias Baru *v*

8. Langkahke-1 sampai ke-7 diulangi sampai semua data pola berhasil dilatih. Kemudian dilakukan pengecekan apakah konfigurasi jaringan sudah optimal (ditandai dengan sudah didapatkannya nilai *error* yang paling kecil). Jika sudah optimal maka proses pelatihan ini akan berhenti. Selain dikarenakan oleh hal tersebut, proses pelatihan jaringan juga bisa berhenti ketika iterasi yang dikerjakan sudah mencapai iterasi maksimal yang diperbolehkan.

Setelah jaringan syaraf tiruan terbentuk, maka proses selanjutnya adalah melakukan pengujian terhadap jaringan syaraf tiruan (JST) tersebut.Pada penerapan jaringan syaraf *optical backpropagation*, proses pengujian dilakukan dengan cara menjalankan proses perambatan maju (*feedforward*) dari proses pelatihan JST dengan menggunakan bias dan bobot yang sudah didapatkan dari proses pelatihan JST.

3.5. Perancangan Uji Coba

Setelah seluruh rancangan sistem sudah berhasil diimplementasikan, maka langkah selanjutnya adalah melakukan pengujian dan evaluasi terhadap sistem yang telah dikembangkan. Tujuan dari pengujian dan evaluasi ini adalah untuk mengetahui ketepatan hasil pemberian label yang dilakukan oleh sistem dan mengetahui tingkat akurasi yang didapatkan oleh sistem anotasi citra ini.

3.5.1. Skenario Evaluasi

Pada penelitian ini, untuk melakukan evaluasi terhadap sistem yang telah dikembangkan, digunakan koleksi citra yang terdiri dari 6 kelompok citra dimana masing-masing kelompok memiliki 90 citra.

Dari 90 citra pada masing-masing kelompok tersebut, 90% akan digunakan sebagai data latih untuk jaringan syaraf tiruan dan sisanya akan digunakan sebagai data uji untuk jaringan syaraf tiruan yang telah terbentuk.Secara keseluruhan terdapat 540 citraobyek yang akan digunakan pada penelitian ini.

Evaluasi yang dilakukan terhadap sistem anotasi ini terdiri dari evaluasi tingkat kesalahan pada saat melakukan proses pembelajaran jaringan syaraf tiruan dan evaluasi dari hasil pemberian label oleh sistem anotasi citra otomatistersebut.

Proses evaluasi pada pelatihan jaringan syaraf tiruan dilakukan dengan menggunakan metode perhitungan SSE dan RMSE untuk mengetahui konfigurasi terbaik pada jaringan syaraf tiruan tersebut. Evaluasi ini dilakukan dengan melakukan beberapa perubahan pada nilai parameter laju pembelajaran (α)dan jumlah *neuron* pada lapisan tersembunyi yang digunakan untuk membetuk jaringan syaraf tiruan tersebut.

Sedangkan untuk mengevaluasi sistem anotasi citra digunakan perhitungan nilai *recall (R), precision (P),* dan *F-measure (F)*. Nilai *F-measure* didapatkan dari perhitungan menggunakan nilai *recall* dan *precision* yang telah didapatkan. Adapun untuk menghitung nilai *recall*, *precision,* dan *F-measure* digunakan persamaan 2.33, persamaan 2.34 dan persamaan 2.35.Pengujian ini dilakukan dengan cara mengubah jumlah data latih yang digunakan untuk melakukan pelatihan jaringan syaraf tiruan. Hasil dari proses pengujian ini digunakan untuk mengetahui tingkat akurasi dari sistem anotasi citra yang telah dikembangkan. Selain pengujian dengan mengubah jumlah data latih, sistem yang akan dikembangkan juga diuji dengan mengubah variasi kelas yang ada pada data latih.

3.5.2. Hasil Evaluasi

Hasil dari semua proses evaluasi yang telah dilakukan disimpan dalam bentuk tabel sehingga dapat dilakukan analisis lebih lanjut tentang pengaruh perubahan beberapa parameter yang digunakan pada saat proses evaluasi sistem.

Tabel 3.11danTabel 3.12adalah tabel yang akan digunakan untuk menunjukkan hasil evaluasi dari arsitektur jaringan syaraf tiruan dan Tabel 3.13dan Gambar 3.15menunjukkan hasil evaluasi tingkat akurasi dari sistem anotasi citra ini.

Tabel 3.11 Evaluasi JST terhadap Laju Pembelajaran

Laju Pembelajaran	**RMSE Percobaan ke-**						$\overline{RMSE}$
	1	2	3	4	5	6	
0.1							
0.2							
0.3							
....							
0.9							

Tabel 3.12 Evaluasi JST terhadap Jumlah *Neuron Hidden Layer*

Neuron Hidden Layer	**RMSE Percobaan ke-**						$\overline{RMSE}$
	1	2	3	4	5	6	
49							
50							
51							
....							
98							

Tabel 3.13 Evaluasi Sistem terhadap Jumlah Data Latih

Learning Data	***Recall***	***Precission***	***F-Measure***	***Time***
20				
40				
60				
...				
160				

Tabel 3.14Evaluasi Sistem terhadap Jumlah Kelas

Jumlah kelas	**Kebenaran**	***Last RMSE***	***Time***
2			
3			
4			


UNIVERSITAS BRAWIJAYA

BAB IV
IMPLEMENTASI DAN PEMBAHASAN

Pada bab implementasi dan pembahasan ini, akan dibahas mengenai penerapan dan pembahasaan sistem yang telah dikembangkan serta evaluasinya. Secara umum proses-proses yang dilakukan oleh sistem adalah *preprocessing*data citra, mengekstraksi fitur dari citra, melatih jaringan syaraf tiruan dengan masukan fitur citra yang terekstraksi, kemudian memberikan label ke citra berdasarkan konten yang terkandung pada citra dengan menggunakan pengetahuan dari jaringan syaraf tiruan hasil pelatihan.

4.1. Lingkungan Implementasi

Agar proses-proses tersebut dapat diimplemetasikan dan berjalan sesuai dengan rancangan yang telah dibuat, maka dibutuhkan media atau lingkungan implementasi yang sesuai. Lingkungan implementasi ini terdiri dari dua, yaitu lingkungan perangkat keras dan lingkungan perangkat lunak.

4.1.1. Lingkungan Perangkat Keras

Perangkat keras yang digunakan dalam pengembangan dan pengujian aplikasi anotasi citra menggunakan *Content Based Image Retrieval* dan jaringan syaraf tiruan *Optical Backpropagation*adalah sebuah *notebook* dengan spesifikasi sebagai berikut :

1. Prosesor AMD Turion™ X2 Dual-Core Mobile RM-74 2.20 GHz.
2. *Chipset* NVIDIA nForce MCP77MH
3. *Video Card* NVIDIA GeForce 9100M
4. Memori 2 Gb
5. *Harddisk* 250 Gb
6. Monitor 14.1”
7. *Keyboard*
8. *Mouse*

4.1.2. Lingkungan Perangkat Lunak

Perangkat lunak yang digunakan untuk mengembangkan aplikasi anotasi citra otomatis dan untuk uji coba adalah sistem operasi *Microsoft Windows* 7 32-bit sebagai lingkungan kerja dari

aplikasi ini, dan *Microsoft Visual Studio* 2010 sebagai perangkat lunak yang digunakan untuk mengembangkan aplikasi dengan menggunakan bahasa pemrograman C#.

4.2. Implementasi Program

Berdasarkan analisa dan perancangan yang telah dipaparkan pada bab 3, maka pada bab ini akan dijelaskan tentang implementasi dari perancangan yang terdapat pada bab 3 tersebut.

4.2.1. Implementasi Struktur Data

Untuk menyimpan data-data yang dibutuhkan ketika sistem ini dijalankan, sistem ini menggunakan struktur data berupa kelas-kelas. Kelas-kelas tersebut digunakan untuk menyimpan nilai-nilai variabel yang digunakan pada sistem anotasi citra. Kelas-kelas ini juga menerapkan beberapa proses yang merupakan bagian dari sistem. Terdapat beberapa kelas utama dan *interface*pada sistem ini, antara lain kelas*HaarWavelet, ImageData, ImageExtraction, ImageFeature, Pixel, NeuralNet, TrainingData, ImageUtility, netUtility,* dan *interface iFile*.

a. Kelas *HaarWavelet*

Kelas *HaarWavelet* digunakan untuk melakukan proses transformasi *wavelet* yang mengubah citra pada domain spasial ke domain frekuensi. Di dalam kelas ini didefinisikan proses-proses yang dilakukan untuk melakukan transformasi citra ke domain frekuensi. Data masukan untuk kelas ini adalah data bertipe *Bitmap.* Kemudian dari data tersebut akan dilakukan proses transformasi citra. Adapun nilai-nilai yang dihasilkan oleh kelas ini yang dapat digunakan oleh kelas lain adalah data citra bertipe *Bitmap* yang telah ter-*filter* menjadi 4 domain frekuensi (*Low-low*, *High-Low*, *Low-High,* dan *High-High*), dan matriks F yang berisi data nilai warna dari citra yang sudah ter-*filter.*

b. **Kelas *ImageData***

Kelas ImageData digunakan untuk membaca citra dan menyimpan data citra, seperti metadata citra, *thumbnail* dari citra, dan beberapa data lain yang dimiliki oleh citra.

Variabel-variabel yang digunakan untuk menyimpan data pada kelas ImageData ini antara lain citra, lebar, tinggi, px, imgPath, dan imgFeature.

Sedangkan fungsi-fungsi yang ada pada kelas ini adalah *readImage*(membaca data citra), *updateImage* (mengubah data citra), *updateRGBPixel* (mengubah nilai komponen RGB pada citra), *binToDec* (mengubah bilangan biner ke desimal), *decToBin* (mengubah bilangan desimal ke biner), *ThumbnailCallBack* (menentukan status pengambilan data *thumbnail* dari citra), *doWavelet* (menjalankan proses transformasi *wavelet* pada citra), *getThumbnail* (mengembalikan data *thumbnail* dari citra), *getImageFeature* (mengembalikan nilai dari fitur citra yang sudah diperoleh), *getBitmap* (mengembalikan data citra bitmap yang sudah tersimpan), *getPixel* (mengembalikan nilai *pixel* yang sudah tersimpan), *getTag* (mengembalikan nilai label yang tersimpan pada metadata citra), *getTarget* (mengembalikan nilai *target* dari citra untuk proses pelatihan JST), *getHistorgram* (mengembalikan nilai *histogram* dari citra).

c. **Kelas *ImageExtraction***

Kelas ImageExtraction digunakan untuk menyimpan nilai dari hasil ekstraksi fitur citra. Adapun nilai-nilai yang dapat disimpan pada kelas ini adalah *filepath*, *feature*, dan *target*.

d. **Kelas *ImageFeature***

Kelas *ImageFeature* adalah kelas yang digunakan untuk melakukan proses ekstraksi fitur dari citra. Pada kelas ini didefiniskan beberapa *method* yang digunakan untuk mendapatkan nilai vektor fitur dari citra. Untuk mendapatkan nilai fitur dari citra, maka harus dilakukan proses transformasi wavelet sehingga kelas ini akan menjalankan proses transformasi wavelet yang ditangani oleh kelas lain (yaitu kelas *HaarWavelet*) untuk menghitung nilai fitur dari citra.

Adapun fungsi-fungsi yang ada pada kelas ini antara lain *doHaarTransform* (memanggil proses transformasi *wavelet* yang ada pada kelas *HaarWavelet*), *getFilteredPx* (mengembalikan nilai *pixel* hasil proses transformasi *wavelet*), *normalizePx* (m).

e. **Kelas *Pixel***

Kelas *Pixel* ini digunakan untuk menyimpan nilai komponen warna *Red, Green,* dan *Blue* yang dimiliki oleh setiap*pixel* pada citra.

f. **Kelas *NeuralNet***

Kelas *NeuralNet* ini adalah kelas yang berfungsi untuk melakukan proses pelatihan jaringan syaraf tiruan dengan menggunakan algoritma pelatihan *Optical Backpropagation*. Kelas ini menerapkan *interface* iFile untuk melakukan proses yang berkaitan dengan pemrosesan berkas.

g. **Kelas *TrainingData***

Kelas *TrainingData* ini digunakan untuk menyimpan hasil dari proses ekstraksi citra yang akan digunakan untuk proses pelatihan jaringan syaraf tiruan. Nilai-nilai yang disimpan pada kelas ini antara lain vektor Input dan vektor target.

h. **Kelas *ImageUtility***

Kelas *ImageUtility* adalah kelas yang digunakan untuk melakukan proses-proses tertentu yang berkaitan dengan citra. Kelas ini merupakan kelas *static* sehingga untuk memanggil *method-method* yang ada pada kelas ini tidak diperlukan proses inisialisasi terlebih dahulu. *Method-method* yang ada pada kelas ini antara lain *isImageNormal*, *resizeImg*, *normalizeRGB*, TampilMatriks, dan *AddBorder*. Semua *method* tersebut adalah *static method*.

i. **Kelas *netUtility***

Kelas *netUtility* digunakan untuk mendefinisikan beberapa *static method* yang berkaitan dengan jaringan syaraf tiruan. Karena kelas ini mengandung *method-method* yang *static*, maka kelas ini juga bersifat *static*, sehingga untuk menggunakan *method-method* yang ada pada kelas ini tidak diperlukan proses inisialisasi kelas terlebih dahulu. Adapun method yang ada pada kelas ini antara lain *getTrainedV*, *getTrainedW*, dan *getNearestT*.

j. *Interface iFile*

Interface iFile adalah kelas *interface* yang digunakan untuk mendefinisikan *method* yang berkaitan dengan pemrosesan berkas pada komputer. Semua kelas yang mengimplementasikan *interface* iFile ini, maka kelas tersebut harus mendefinisikan semua *method* yang ada pada *interface* iFile ini. Adapun method yang ada pada interface ini adalah *saveToFile*.

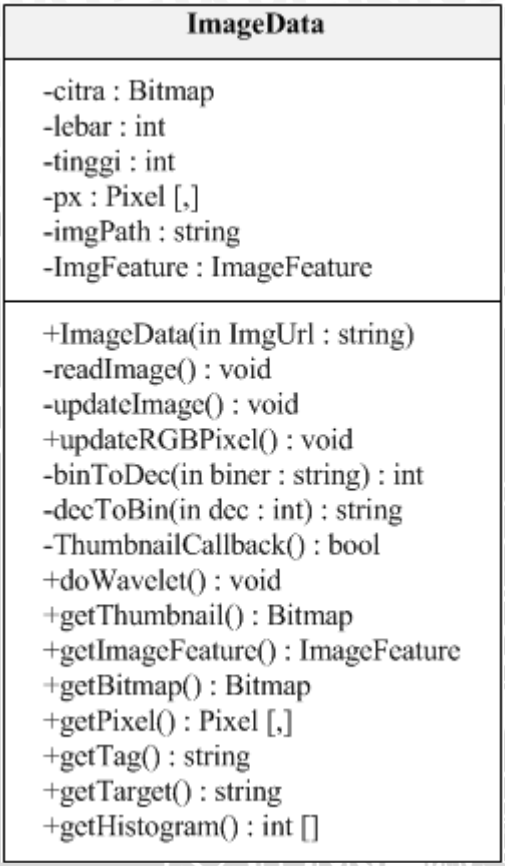
4.2.2. Implementasi Pelatihan Data Latih

Pada sub bab ini akan dijelaskan mengenai implementasi dari pelatihan data latih berupa fitur-fitur dari citra yang didapatkan dari proses ekstraksi fitur pada citra. Adapun proses-proses yang dilakukan pada pelatihan data latih ini adalah proses pengambilan data citra, proses normalisasi citra, proses transformasi *Wavelet*, proses pengubahan model warna, proses ekstraksi fitur, proses pelatihan jaringan syaraf tiruan, dan penyimpanan hasil pelatihan jaringan syaraf tiruan.

4.2.2.1. Implementasi Pengambilan Data Citra

Tahap awal yang dilakukan pada pelatihan aplikasi anotasi citra ini adalah mengambil semua data citra yang akan digunakan sebagai data latih. Data citra yang diambil adalah berkas bertipe jpg pada *folder* yang sudah ditentukan. Implementasi pengambilan dan penyimpanan data citra terdapat pada kelas *ImageData*.

Pada kelas ini terdapat beberapa fungsi, antara lain fungsi yang digunakan untuk membaca bit-bit data yang ada pada citra, fungsi memperbarui nilai warna yang terkandung pada citra, fungsi untuk menjalankan proses transformasi *Wavelet*, fungsi untuk mendapatkan data *thumbnail* untuk citra, fungsi untuk mengambil data tag citra yang tersimpan pada metadata, fungsi untuk mendapatkan nilai target yang digunakan untuk pelatihan jaringan syaraf tiruan, dan beberapa variabel yang digunakan untuk menyimpan data-data citra. Semua fungsi yang sudah disebutkan digambarkan dalam bentuk UML yang dapat dilihat pada Gambar 4.1.



Gambar 4.1 Diagram UML Kelas *ImageData*

Adapun implementasi konstruktor, method, dan variabel-variabel yang ada pada kelas *ImageData* ini dapat dilihat pada *Source Code4.1*.

ImageData.cs

```
using System;
using System.Drawing;
using System.Drawing.Imaging;
using System.IO;
using System.Linq;
using System.Windows.Forms;
using ExifLibrary;
```

```
using ImageAnnotation.Properties;
using ImageAnnotation.Utility;

namespace ImageAnnotation.ExtractingImage
{
classImageData
{
privateBitmap srcImg = null;
privateDouble[,] ImageGrid = null;
privateDouble[,] filteredGrid = null;
privateint height, width;
privatePixel[,] px = null;
privateString ImgPath = "";
privateImageFeature ImgFeature;

public ImageData(String s)
{
this.srcImg = newBitmap(s);
this.height = srcImg.Height;
this.width = srcImg.Width;
ImgPath = s;
}

public ImageData(Bitmap myBmp)
{
this.srcImg = myBmp;
this.height = srcImg.Height;
this.width = srcImg.Width;
}

privatevoid readImage()
{
// Normalize first
if (!ImageUtility.isImageNormal(this.srcImg))
{
int                    resizeSize                    =
ClassifySettings.Default.normalizationSize;
this.srcImg     =     ImageUtility.resizeImg(this.srcImg,
resizeSize);
this.height = this.srcImg.Height;
this.width = this.srcImg.Width;
}

px = newPixel[height, width];
```

```
for (int i = 0; i < height; i++)
{
for (int j = 0; j < width; j++)
{
px[i, j] = newPixel();
}
}

ImageGrid = newDouble[height, width];

BitmapData                    dataCitra                    =
this.srcImg.LockBits(newRectangle(0, 0, width, height),
ImageLockMode.ReadWrite,
PixelFormat.Format24bppRgb);
int stride = dataCitra.Stride;
int offset = stride - 3 * width;
System.IntPtr scan0 = dataCitra.Scan0;

unsafe
{
byte* p = (byte*)(void*)scan0;
for (int i = 0; i < height; i++)
{
for (int j = 0; j < width; j++)
{
px[i, j].Biru = p[0];
px[i, j].Hijau = p[1];
px[i, j].Merah = p[2];
int intensity = (p[0] + p[1] + p[2]) / 3;
//int luminance = (int)(0.114 * p[0] + 0.587 * p[1] +
0.299 * p[2]);
ImageGrid[i, j] = intensity;
p += 3;
}
p += offset;
}
}
this.srcImg.UnlockBits(dataCitra);
}

privateBitmap PixelToImage(Pixel[,] _px)
{
Bitmap tempImage = newBitmap(srcImg);
BitmapData                    dataCitra                    =
```

```
tempImage.LockBits(newRectangle(0, 0, width, height),
ImageLockMode.ReadWrite,
PixelFormat.Format24bppRgb);
int stride = dataCitra.Stride;
int offset = stride - 3 * width;
System.IntPtr scan0 = dataCitra.Scan0;

unsafe
{
byte* p = (byte*)(void*)scan0;
for (int i = 0; i < height; i++)
{
for (int j = 0; j < width; j++)
{
p[0] = px[i, j].Biru;
p[1] = px[i, j].Hijau;
p[2] = px[i, j].Merah;
p += 3;
}
p += offset;
}
}
tempImage.UnlockBits(dataCitra);

return tempImage;
}

privateint binToDec(string biner)
{
int result = 0;
char[] numbers = biner.ToCharArray();
for (int counter = numbers.Length; counter > 0; counter-
-)
{
int num = int.Parse(numbers[counter - 1].ToString());
int exp = numbers.Length - counter;
result += (Convert.ToInt32(Math.Pow(2, exp)) * num);
}
return result;
}

privateString decToBin(int dec)
{
string binary = Convert.ToString(dec, 2);
```

```
int leftZero = (30 - binary.Length);
for (int i = 0; i < leftZero; i++)
binary = "0" + binary;
return binary;
}

privatebool ThumbnailCallback()
{
returntrue;
}

publicvoid doWavelet(int l, Boolean b, int t)
{
this.readImage();
HaarWavelet hw = newHaarWavelet(l, b, t);
hw.Apply(ImageGrid);

this.filteredGrid = hw.getFilteredPixel();
}

publicvoid extractFeature()
{
int level = ClassifySettings.Default.transformLevel;
Boolean                    binarization                    =
ClassifySettings.Default.binarization;
int threshold = ClassifySettings.Default.threshold;

this.doWavelet(level, binarization, threshold);
this.ImgFeature = newImageFeature(this.filteredGrid);
}

/* GET METHOD */

publicBitmap getThumbnail()
{
Bitmap thumbnail = (Bitmap)srcImg.GetThumbnailImage(128,
128,                                                    new
System.Drawing.Bitmap.GetThumbnailImageAbort(ThumbnailCa
llback), IntPtr.Zero);
thumbnail   =   (Bitmap)ImageUtility.AddBorder(thumbnail,
SystemColors.ButtonShadow, 4);
return thumbnail;
}
```

```
publicBitmap getFilteredImage()
{
Pixel[,] filteredPx = newPixel[width, height];
for (int i = 0; i < height; i++)
{
for (int j = 0; j < width; j++)
{
px[i, j].Merah = px[i, j].Hijau = px[i, j].Biru =
(byte)filteredGrid[i, j];
}
}
return PixelToImage(filteredPx);
}

publicDouble[] getExtractedFeature()
{
returnthis.ImgFeature.getExtractedFeature();
}

publicString getExtractedFeatureStr()
{
returnthis.ImgFeature.ToString();
}

publicBitmap getBitmap()
{
returnthis.srcImg;
}

publicDouble[,] getImageGrid()
{
this.readImage();
returnthis.ImageGrid;
}

publicDouble[,] getFilteredGrid()
{
returnthis.filteredGrid;
}

publicString getTag()
{
String label = "unknown";
try
```

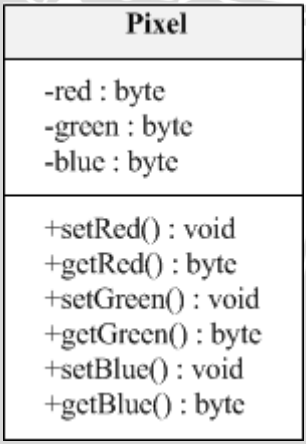
```
{
ImageFile data = ImageFile.FromFile(ImgPath);
label                                                    =
data.Properties[ExifLibrary.ExifTag.WindowsKeywords].ToS
tring();
data = null;
}
catch
{
}

return label;
}

publicString getTarget()
{
String _fromLbl = this.getTag();
String     target     =     "",     classInfoPath     =
Settings.Default.appDataLoc + "/db/label_0.ia";
String[] t;
try
{
StreamReader                streamReader                =
newStreamReader(classInfoPath);
String tmpStream = streamReader.ReadToEnd();
if (tmpStream.Length == 0)
{
streamReader.Close();
thrownewInvalidDataException("Label   file   can   not   be
empty");
}
t = tmpStream.Split('\n');
streamReader.Close();
}
catch (Exception ex)
{
System.IO.File.WriteAllText(classInfoPath, decToBin(0) +
">" + _fromLbl);
MessageBox.Show(ex.Message + Environment.NewLine
+    "We    have    created    needed    file",    "Error",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
// Baca lagi file baru
StreamReader                streamReader                =
newStreamReader(classInfoPath);
```

```
String tmpStream = streamReader.ReadToEnd();
t = tmpStream.Split('\n');
streamReader.Close();
}

foreach (String t2 in t)
{
if (t2.Contains('>'))
{
string tmp = t2.Trim();
string[] t3 = tmp.Split('>');

if (t3[1] == _fromLbl)
{
target = t3[0];
break;
}
}
}

// Definisikan kelas baru jika belum ada
if (target == "")
{
String lastTarget = t[t.Length - 1];
lastTarget = lastTarget.Substring(0,
lastTarget.IndexOf('>'));
int nextCount = binToDec(lastTarget) + 1;
String newTarget = decToBin(nextCount);
File.AppendAllText(classInfoPath, Environment.NewLine +
newTarget + ">" + _fromLbl);
target = newTarget;
}
return target;
}

publicint[] getHistogram()
{
int[] myHistogram = newint[256];

for (int i = 0; i < height; i++)
{
for (int j = 0; j < width; j++)
{
int Temp = 0;
```

```
Temp += px[i, j].Merah;
Temp += px[i, j].Hijau;
Temp += px[i, j].Biru;

Temp = (int)Temp / 3;

myHistogram[Temp]++;
}
}

// for (int i = 0; i < myHistogram.Length; i++)
// {
//myHistogram[i] = myHistogram[i] / (tinggi * lebar);
// }
return myHistogram;
}

}
}
```

*Source Code*4.1 Implementasi kelas *ImageData*

Pada kelas ini digunakan kelas *Pixel* untuk menyimpan nilai setiap *pixel* yang terdapat pada citra. Adapun implementasi dari kelas *pixel* ini dapat dilihat pada Gambar 4.2 dan *Source Code*4.2.



Gambar 4.2Diagram UML Kelas *Pixel*

```
Pixel.cs
namespace ImageAnnotation
{
classPixel{
publicbyte Biru { get; set; }
publicbyte Hijau { get; set; }
publicbyte Merah { get; set; }
}
}
```

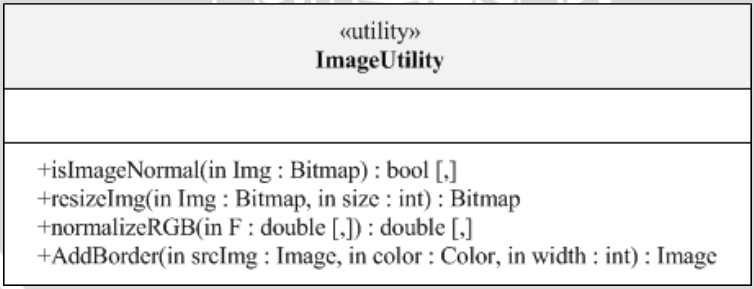
*Source Code*4.2Implementasi Kelas *Pixel*

4.2.2.2. Implementasi Normalisasi Citra

Setelah data citra yang akan digunakan sebagai data latih berhasil diambil, maka tahap berikutnya adalah melakukan normalisasi data citra masukan untuk dapat diolah pada tahap selanjutnya. Pada proses normalisasi ini dilakukan pengecekan apakah citra memiliki dimensi 128x128 atau tidak. Jika citra masukan tidak berdimensi 128x128 maka akan dilakukan proses pengubahan ukuran citra, dan sebaliknya jika citra masukan sudah berukuran 128x128 maka tidak dilakukan proses pengubahan citra.

Proses pengubahan ukuran citra ini menggunakan model interpolasi *HighQualityBicubic* yang sudah disediakan oleh *Microsoft Visual Studio*.

Kelas yang menangani proses pengecekan dimensi citra dan proses pengubahan dimensi citra ini adalah kelas *ImageUtility*. *Method* dan variabel yang ada pada kelas *ImageUtility* ini digambarkan dalam bentuk UML yang ditunjukkan pada Gambar 4.3.



Gambar 4.3Diagram UML Kelas *ImageUtility*

Adapun implementasi dari kelas ini ditunjukkan pada *Source Code*4.3yaitu ditunjukkan pada *method isImageNormal* untuk proses pengecekan dimensi citra dan *method resizeImg*untuk proses pengubahan dimensi citra.

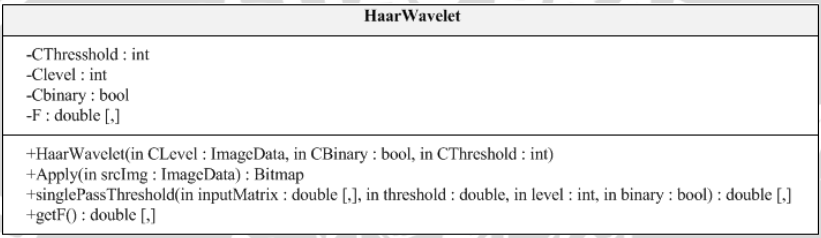
ImageUtility.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Drawing;
using System.Drawing.Drawing2D;

namespace ImageAnnotation
{
publicstaticclassImageUtility
{
publicstaticBoolean isImageNormal(Bitmap Img)
{
int h = Img.Height;
int w = Img.Width;
if (h != 128 || w != 128) returnfalse;
returntrue;
}

publicstaticBitmap resizeImg(Bitmap Img, int size)
{
int sourceWidth = Img.Width;
int sourceHeight = Img.Height;
int destWidth = size;
int destHeight = size;

Bitmap b = newBitmap(destWidth, destHeight);
Graphics g = Graphics.FromImage((Image)b);
g.InterpolationMode=InterpolationMode.HighQualityBicubic;
g.DrawImage(Img, 0, 0, destWidth, destHeight);
g.Dispose();

return b;
}
}}
```

*Source Code*4.3Impelementasi Kelas *ImageUtility*

4.2.2.3. Implementasi Transformasi Wavelet

Setelah data citra berhasil dinormalisasi menjadi ukuran yang sudah ditentukan. Maka tahap berikutnya adalah melakukan proses transformasi *Wavelet*.Proses transformasi *Wavelet* ini dilakukan menggunakan metode transformasi *Haar Wavelet*. Kelas yang menangani proses transformasi ini adalah kelas *HaarWavelet*. Kelas ini digambarkan dalam bentuk UML yang ditunjukkan padaGambar 4.4.



Gambar 4.4Diagram UML Kelas *HaarWavelet*

Adapun implementasi dari kelas *HaarWavelet* ini dapat dilihat pada *Source Code*4.4.

HaarWavelet.cs

```
using System;
using System.Drawing;
using System.Drawing.Imaging;

namespace ImageAnnotation
{
classHaarWavelet
{
privateint CThresshold;
privateint CLevel;
privatebool CBinary;
privatedouble[,] F;

// Constructor
public    HaarWavelet(int    CLevel,    bool    CBinary,    int
CThresshold)
{
this.CLevel = CLevel;
this.CBinary = CBinary;
this.CThresshold = CThresshold;
```

```
}

// Apply filter
publicBitmap Apply(Bitmap srcImg)
{
// get source image size
int width = srcImg.Width;
int height = srcImg.Height;

PixelFormat      fmt      =      (srcImg.PixelFormat      ==
PixelFormat.Format8bppIndexed) ?
PixelFormat.Format8bppIndexed : PixelFormat.Format24bppRgb;

// lock source bitmap data
BitmapData srcData = srcImg.LockBits(
newRectangle(0, 0, width, height),
ImageLockMode.ReadOnly, fmt);

// create new image
Bitmap dstImg = (fmt == PixelFormat.Format8bppIndexed) ?
   AForge.Imaging.Image.CreateGrayscaleImage(width, height)
:
newBitmap(width, height, fmt);

// lock destination bitmap data
BitmapData dstData = dstImg.LockBits(
newRectangle(0, 0, width, height),
ImageLockMode.ReadWrite, fmt);

int     offset     =     srcData.Stride     -     ((fmt     ==
PixelFormat.Format8bppIndexed) ? width : width * 3);

double[,] ImageGrid = newdouble[width, height];
double[,] WaveletImageGrid = newdouble[width, height];


// Getting the histogram
// do the job
unsafe
{
byte* src = (byte*)srcData.Scan0.ToPointer();
for (int y = 0; y < height; y++)
{
for (int x = 0; x < width; x++, src += 3)
{
int mean = (int)(0.114 * src[0] + 0.587 * src[1] + 0.299 *
```

```
src[2]);
ImageGrid[x, y] = mean;
}
src += offset;
}
}

// end of histogram

// Filter
for (int i = 1; i <= CLevel; i++)
{
WaveletImageGrid      =      singlePassThreshold(ImageGrid,
CThresshold, i, CBinary);
ImageGrid = WaveletImageGrid;
}

F = WaveletImageGrid;
// end of the filter

// do the job -> convert filtered data to image
unsafe
{
byte* src = (byte*)srcData.Scan0.ToPointer();
byte* dst = (byte*)dstData.Scan0.ToPointer();

// RGB invert
for (int y = 0; y < height; y++)
{
for (int x = 0; x < width; x++, src += 3, dst += 3)
{
dst[2] = (byte)WaveletImageGrid[x, y];
dst[1] = (byte)WaveletImageGrid[x, y];
dst[0] = (byte)WaveletImageGrid[x, y];
}
src += offset;
dst += offset;
}
}
// unlock both images
dstImg.UnlockBits(dstData);
srcImg.UnlockBits(srcData);
// end of filtering

return dstImg;
}
```

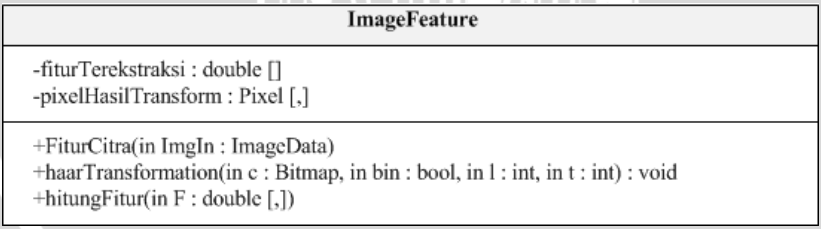
```
privatedouble[,] singlePassThreshold(double[,] inputMatrix,
double threshold, int level, bool binary)
{
level = (int)Math.Pow(2.0, level - 1);
double[,]                    resultMatrix                    =
newdouble[inputMatrix.GetLength(0),
inputMatrix.GetLength(1)];
int xOffset = inputMatrix.GetLength(0) / 2 / level;
int yOffset = inputMatrix.GetLength(1) / 2 / level;
int currentPixel = 0;
double      size      =      inputMatrix.GetLength(0)      *
inputMatrix.GetLength(1);
double multiplier = 0;

for (int y = 0; y < inputMatrix.GetLength(1); y++)
{
for (int x = 0; x < inputMatrix.GetLength(0); x++)
{
if ((y < inputMatrix.GetLength(1) / 2 / level) && (x <
inputMatrix.GetLength(0) / 2 / level))
{
currentPixel++;
resultMatrix[x,  y]  =  (inputMatrix[2  *  x,  2  *  y]  +
inputMatrix[2 * x + 1, 2 * y] + inputMatrix[2 * x, 2 * y +
1] + inputMatrix[2 * x + 1, 2 * y + 1]) / 4;
double  vertDiff  =  (-inputMatrix[2  *  x,  2  *  y]  -
inputMatrix[2 * x + 1, 2 * y] + inputMatrix[2 * x, 2 * y +
1] + inputMatrix[2 * x + 1, 2 * y + 1]);
double  horzDiff  =  (inputMatrix[2  *  x,  2  *  y]  -
inputMatrix[2 * x + 1, 2 * y] + inputMatrix[2 * x, 2 * y +
1] - inputMatrix[2 * x + 1, 2 * y + 1]);
double  diagDiff  =  (-inputMatrix[2  *  x,  2  *  y]  +
inputMatrix[2 * x + 1, 2 * y] + inputMatrix[2 * x, 2 * y +
1] - inputMatrix[2 * x + 1, 2 * y + 1]);

if (binary)
{
if (Math.Abs(vertDiff) > threshold)
{ resultMatrix[x + xOffset, y] = 255; }

if (Math.Abs(horzDiff) > threshold)
{ resultMatrix[x, y + yOffset] = 255; }

if (Math.Abs(diagDiff) > threshold)
{ resultMatrix[x + xOffset, y + yOffset] = 255; }
```

```
}
else
{
resultMatrix[x + xOffset, y] = multiplier +
Math.Abs(vertDiff);
resultMatrix[x, y + yOffset] = multiplier +
Math.Abs(horzDiff);
resultMatrix[x + xOffset, y + yOffset] = multiplier +
Math.Abs(diagDiff);
}}
else
{
if ((x >= inputMatrix.GetLength(0) / level) || (y >=
inputMatrix.GetLength(1) / level))
{
resultMatrix[x, y] = inputMatrix[x, y];
}}}}
return resultMatrix;
}

publicdouble[,] getF()
{
return F;
}}}
```

*Source Code*4.4Implementasi Kelas *HaarWavelet*

4.2.2.4. Implementasi Ekstraksi Fitur

Tahap ini dilakukan setelah didapatkan nilai *intensity* dari citra hasil proses transformasi *Wavelet*. Implementasi dari kelas ini dapat dilihat pada Gambar 4.5 dan *Source Code*4.5.



Gambar 4.5 Diagram UML Kelas *ImageFeature*

```
ImageFeature.cs
using System;
```

```
using System.Drawing;

namespace ImageAnnotation.ExtractingImage
{
classImageFeature
{
privateBitmap filteredImage;
privateDouble[] extractedFeature;

public  ImageFeature(Bitmap  mCitra,  Boolean  binarization,
int level, int threshold)
{
haarTransformation(mCitra,binarization,level,threshold);
}

privatevoid   haarTransformation(Bitmap   mCitra,   Boolean
binarization, int level, int threshold)
{
HaarWavelet   hw   =   newHaarWavelet(level,   binarization,
threshold);
filteredImage = hw.Apply(mCitra);
hitungFitur(hw.getF());
}

publicvoid hitungFitur(Double[,] F)
{
Double[,] mInput = getHHBand(F);
extractedFeature = slidingWindow(mInput);
}

privateDouble[] slidingWindow(Double[,] P)
{
Double[] fitur = newDouble[49];
int slideDistance = 8;
int maskSize = 16;

int m = P.GetUpperBound(0);
int k = 0;
for(int yB=maskSize-1; yB<=m; yB+=slideDistance)
{
int yA = yB - 1;
for(int xB=maskSize-1; xB<=m; xB+=slideDistance)
{
int xA = xB - 1;
// Mask yang bergeser2
Double[,] mask = newDouble[maskSize, maskSize];
```

```
int a = 0;
for (int i = yA; i <= yB; i++)
{
int b = 0;
for (int j = xA; j <= xB; j++)
{
mask[a, b] = P[i, j];
b++;
}
a++;
}
fitur[k] = extractFitur(mask);
k++;
}}
return fitur;
}

publicDouble extractFitur(Double[,] P)
{
int                extractingComponent                =
ImageAnnotation.Properties.ClassifySettings.Default.compToE
xtract;
Double fitur = 0;
Double px = 0;
switch (extractingComponent)
{
case 0: // Diagonal Moment
for (int i = 0; i < 2; i++)
{
for (int j = 0; j < 2; j++)
{
px = Math.Sqrt((Math.Abs(i - j) * P[i, j]) / 2);
fitur += px;
}}
break;
case 1:
break;
}
return fitur;
}

publicDouble[] getExtractedFeature()
{
return extractedFeature;
}
```

```
publicBitmap getFilteredImg()
{
return filteredImage;
}

privateDouble[,] getHHBand(Double[,] F)
{
int n = F.GetLength(0) / 2;
Double[,] t = newDouble[n, n];
for (int i = 0; i < n; i++)
{
for (int j = 0; j < n; j++)
{
t[i, j] = F[n + i, n + j];
}}
return t;
}

publicoverrideString ToString()
{
String temp = "";
for (int i = 0; i < extractedFeature.Length; i++)
{
temp += (extractedFeature[i] + ";");
}
return temp;
}}}
```

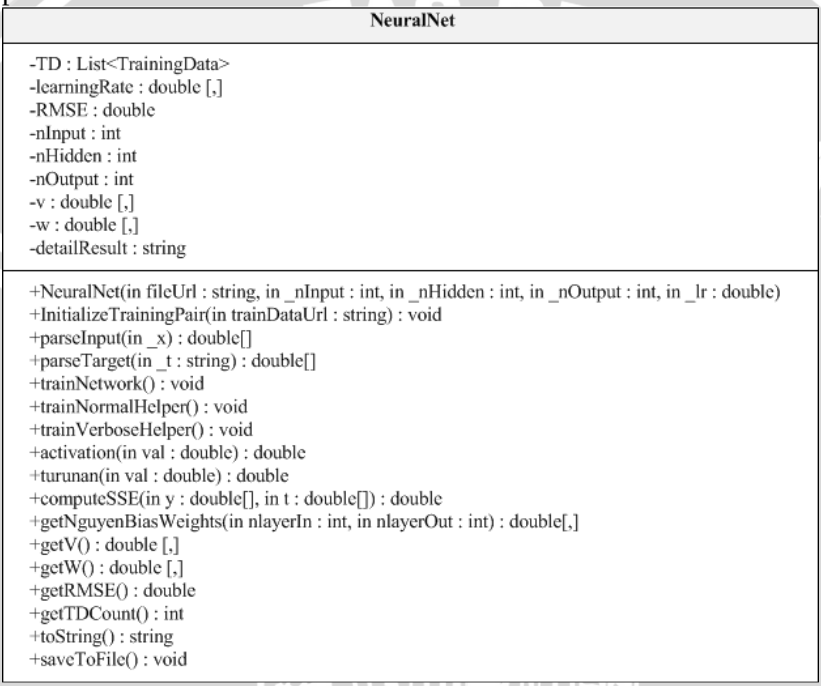
*Source Code*4.5Implementasi Kelas *ImageFeature*

Adapun contoh dari hasil proses transformasi *wavelet* untuk beberapa macam citra dapat dilihat pada

Lampiran 3.

4.2.2.5. Implementasi Pelatihan JST

Setelah nilai-nilai fitur yang merepresentasikan sebuah citra sudah didapatkan, maka nilai-nilai ini akan digunakan sebagai nilai masukan untuk melatih jaringan syaraf tiruan. Implementasi dari kelas yang digunakan untuk melakukan pelatihan JST dapat dilihat pada Gambar 4.6 dan *Source Code4.6*.



Gambar 4.6Diagram UML Kelas *NeuralNet*

NeuralNet.cs

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;
using System.Xml;
using ImageAnnotation.Properties;

namespace ImageAnnotation
{
classNeuralNet : iFile
```

```
{
privateList<TrainingData> TD= newList<TrainingData>();
privateDouble RMSE, learningRate;
privateint nInput, nHidden, nOutput;
privateDouble[,] v, w;
privateString detailResult = "";

public NeuralNet(String fileUrl, int _nInput, int _nHidden,
int _nOutput, Double _lr)
{
this.nInput = _nInput;
this.nHidden = _nHidden;
this.nOutput = _nOutput;
this.learningRate = _lr;

InitializeTrainingPair(fileUrl);
this.v = getNguyenBiasWeights(nInput, nHidden);
this.w = getNguyenBiasWeights(nHidden, nOutput);
}

privatevoid InitializeTrainingPair(String trainDataUrl)
{
XmlDocument doc = newXmlDocument();
doc.Load(trainDataUrl);
XmlNodeList                    filePathNode                =
doc.SelectNodes("traindata/data/feature");

foreach (XmlNode node in filePathNode)
{
TrainingData myTD = newTrainingData();
myTD.Input = parseInput(node.InnerText);
myTD.Target = parseTarget(node.NextSibling.InnerText);
this.TD.Add(myTD);
myTD = null;
}
}

privateDouble[] parseInput(String _x)
{
String[] tmp = _x.Split(';');
Double[] t = newDouble[tmp.Length - 1];
for (int i = 0; i < tmp.Length; i++)
{
if (tmp[i] != "")
{
t[i] = Double.Parse(tmp[i]);
```

```
}
}
return t;
}

privateDouble[] parseTarget(String _t)
{
char[] tmp = _t.ToCharArray();
Double[] t = newDouble[tmp.Length];
for (int i = 0; i < tmp.Length; i++)
{
t[i] = Double.Parse(tmp[i] + "");
}
return t;
}

publicvoid trainNetwork()
{
if (Settings.Default.showNeuVal) trainVerboseHelper();
else trainNormalHelper();
}

privatevoid trainNormalHelper()
{
Double[,] deltaW = newDouble[nHidden + 1, nOutput];
Double[,] deltaV = newDouble[nInput + 1, nHidden];
Double[] z_in = newDouble[nHidden];
Double[] z = newDouble[nHidden];
Double[] y_in = newDouble[nOutput];
Double[] y = newDouble[nOutput];
Double[] smallDeltaK = newDouble[nOutput];
Double[] smallDeltaJ = newDouble[nHidden];
Double[] smallDelta_inJ = newDouble[nHidden];
this.RMSE = 0;

/* Train untuk setiap pair */
foreach(TrainingData myTD in TD)
{
Double[] x = myTD.Input;
Double[] t = myTD.Target;

// FEEDFORWARD
/* INPUT ke HIDDEN */
for (int j = 0; j < nHidden; j++)
{
z_in[j] = 0;
```

```
for (int i = 1; i <= nInput; i++)
{
z_in[j] += x[i - 1] * v[i, j];
}
z_in[j] += v[0, j];
z[j] = activation(z_in[j]);
}

/* HIDDEN ke OUTPUT */
for (int k = 0; k < nOutput; k++)
{
y_in[k] = 0;
for (int j = 1; j <= nHidden; j++)
{
y_in[k] += z[j - 1] * w[j, k];
}
y_in[k] += w[0, k];
y[k] = activation(y_in[k]);
}

// BACKPROPAGATION ERROR

/* compute smallDelta k */
for (int k = 0; k < nOutput; k++)
{
if (NNsetting.Default.trainMethod == 0)
{
// do optical backprop
if   (t[k]   -   y[k]   >=   0)   smallDeltaK[k]   =   (1   +
Math.Exp(Math.Pow(t[k] - y[k], 2))) * y[k] * (1 - y[k]);
elseif  (t[k]  -  y[k]  <  0)  smallDeltaK[k]  =  -1  *  (1  +
Math.Exp(Math.Pow(t[k] - y[k], 2))) * y[k] * (1 - y[k]);
}
else
{
// do backprop
smallDeltaK[k] = (t[k] - y[k]) * (y[k] * (1 - y[k]));
}
}

/* compute delta w */
for (int k = 0; k < nOutput; k++)
{
for (int j = 1; j <= nHidden; j++)
{
deltaW[j, k] = learningRate * smallDeltaK[k] * z[j - 1];
```

```
}
deltaW[0, k] = learningRate * smallDeltaK[k];
}

/* compute smallDelta_in j and smallDelta j  */
for (int j = 0; j < nHidden; j++)
{
smallDelta_inJ[j] = 0;
for (int k = 0; k < nOutput; k++)
{
smallDelta_inJ[j] += smallDeltaK[k] * w[j, k];
}
smallDeltaJ[j]          =          smallDelta_inJ[j]          *
turunan(smallDelta_inJ[j]);
}

/* compute delta v */
for (int j = 0; j < nHidden; j++)
{
for (int i = 1; i <= nInput; i++)
{
deltaV[i, j] = learningRate * smallDeltaJ[j] * x[i - 1];
}
deltaV[0, j] = learningRate * smallDeltaJ[j];
}

/* RENEW ALL WEIGHTS AND BIASES */
for (int k = 0; k < nOutput; k++)
{
for (int j = 0; j <= nHidden; j++)
{
w[j, k] = w[j, k] + deltaW[j, k];
}
}
for (int j = 0; j < nHidden; j++)
{
for (int i = 0; i <= nInput; i++)
{
v[i, j] = v[i, j] + deltaV[i, j];
}
}
this.RMSE += computeSSE(y, t);
}
this.RMSE = Math.Sqrt(RMSE / TD.Count);
}
```

```
privatevoid trainVerboseHelper()
{
Double[,] deltaW = newDouble[nHidden + 1, nOutput];
Double[,] deltaV = newDouble[nInput + 1, nHidden];
Double[] z_in = newDouble[nHidden];
Double[] z = newDouble[nHidden];
Double[] y_in = newDouble[nOutput];
Double[] y = newDouble[nOutput];
Double[] smallDeltaK = newDouble[nOutput];
Double[] smallDeltaJ = newDouble[nHidden];
Double[] smallDelta_inJ = newDouble[nHidden];
this.RMSE = 0;

this.detailResult = Environment.NewLine;

/* Train untuk setiap pair */
for (int a = 0; a < TD.Count; a++)
{
Double[] x = TD[a].Input;
Double[] t = TD[a].Target;
this.detailResult    +=    ">    DATA    KE-"    +    a    +
Environment.NewLine;
this.detailResult += Environment.NewLine + "INPUT LAYER: "
+ Environment.NewLine;
foreach (Double myX in x)
this.detailResult += String.Format("{0:F6}", myX) + "   ";
// FEEDFORWARD
/* z_in^j */

for (int j = 0; j < nHidden; j++)
{
for (int i = 1; i <= nInput; i++)
{
z_in[j] += x[i - 1] * v[i, j];
}
z_in[j] += v[0, j];
}

this.detailResult       +=       Environment.NewLine       +
Environment.NewLine     +     "HIDDEN     LAYER:     "     +
Environment.NewLine;
/* z^j - fungsi aktivasi */
for (int j = 0; j < nHidden; j++)
{
z[j] = activation(z_in[j]);
this.detailResult += String.Format("{0:F6}", z[j]) + "   ";
```

```
}

/* y_in^k */
for (int k = 0; k < nOutput; k++)
{
for (int j = 1; j <= nHidden; j++)
{
y_in[k] += z[j - 1] * w[j, k];
}
y_in[k] += w[0, k];
}

this.detailResult       +=       Environment.NewLine       +
Environment.NewLine     +     "OUTPUT     LAYER:     "     +
Environment.NewLine;

for (int k = 0; k < nOutput; k++)
{
y[k] = activation(y_in[k]);
this.detailResult += String.Format("{0:F6}", y[k]) + "";
}
this.detailResult += Environment.NewLine;

// BACKPROPAGATION ERROR

/* compute smallDelta k */
for (int k = 0; k < nOutput; k++)
{
if (NNsetting.Default.trainMethod == 0)
{
// do optical backprop
if   (t[k]   -   y[k]   >=   0)   smallDeltaK[k]   =   (1   +
Math.Exp(Math.Pow(t[k] - y[k], 2))) * y[k] * (1 - y[k]);
elseif  (t[k]  -  y[k]  <  0)  smallDeltaK[k]  =  -1  *  (1  +
Math.Exp(Math.Pow(t[k] - y[k], 2))) * y[k] * (1 - y[k]);
}
else
{
// do backprop
smallDeltaK[k] = (t[k] - y[k]) * (y[k] * (1 - y[k]));
}
}

/* compute delta w */
for (int k = 0; k < nOutput; k++)
{
```

```
for (int j = 1; j <= nHidden; j++)
{
deltaW[j, k] = learningRate * smallDeltaK[k] * z[j - 1];
}
deltaW[0, k] = learningRate * smallDeltaK[k];
}

/* compute smallDelta_in j and smallDelta j  */
for (int j = 0; j < nHidden; j++)
{
for (int k = 0; k < nOutput; k++)
{
smallDelta_inJ[j] += smallDeltaK[k] * w[j, k];
}
smallDeltaJ[j]          =          smallDelta_inJ[j]          *
turunan(smallDelta_inJ[j]);
}

/* compute delta v */
for (int j = 0; j < nHidden; j++)
{
for (int i = 1; i <= nInput; i++)
{
deltaV[i, j] = learningRate * smallDeltaJ[j] * x[i - 1];
}
deltaV[0, j] = learningRate * smallDeltaJ[j];
}

/* RENEW ALL WEIGHTS AND BIASES */
for (int k = 0; k < nOutput; k++)
{
for (int j = 0; j <= nHidden; j++)
{
w[j, k] = w[j, k] + deltaW[j, k];
}
}
for (int j = 0; j < nHidden; j++)
{
for (int i = 0; i <= nInput; i++)
{
v[i, j] = v[i, j] + deltaV[i, j];
}
}
Double SSE = computeSSE(y, t);
this.RMSE += SSE;
this.detailResult += Environment.NewLine + "SSE : " + SSE +
```

```
Environment.NewLine + Environment.NewLine;
}
this.RMSE = Math.Sqrt(RMSE / TD.Count);
}

privateDouble activation(Double val)
{
return 1.0 / (1.0 + Math.Exp(-val));
}

privateDouble turunan(Double val)
{
Double f_x = activation(val);
return f_x * (1 - f_x);
}

privateDouble computeSSE(Double[] y, Double[] t)
{
Double SSE = 0.0;
for (int i = 0; i < y.Length; i++)
{
SSE += Math.Pow(y[i] - t[i], 2);
}

return SSE;
}

privateDouble[,] getNguyenBiasWeights(int nlayerIn, int
nlayerOut)
{
Random rd = newRandom();
Double[,] weight = newDouble[nlayerIn, nlayerOut];
Double[] bias = newDouble[nlayerOut];
Double[] absolut = newDouble[nlayerIn];
Double min, max, bheta;
min = -0.5; max = 0.5;

/* GENERATING RANDOM WEIGHT (-0.5,0.5) */

for (int i = 0; i < nlayerIn; i++)
{
for (int j = 0; j < nlayerOut; j++)
{
weight[i, j] = rd.NextDouble() * (max - min) + min;
if (weight[i, j] < -0.5 || weight[i, j] > 0.5)
MessageBox.Show("Melebihi batas");
```

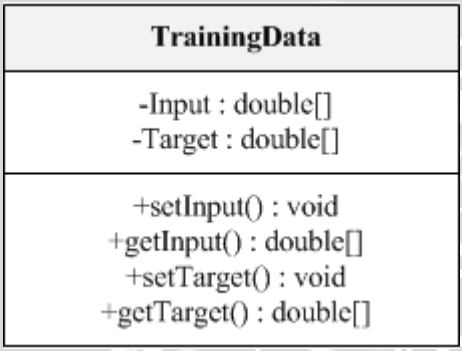
```
}
}

/* RENEW WEIGHT USING BHETA */
bheta = 0.7 * Math.Pow(nlayerOut, 1.0 / nlayerIn);

/* Count nilai |i| untuk semua bobot pada setiap baris */
for (int i = 0; i < nlayerIn; i++)
{
Double t = 0;
for (int j = 0; j < nlayerOut; j++)
{
t += Math.Pow(weight[i, j], 2);
}
absolut[i] = Math.Sqrt(t);
}

/* Update bobot */
for (int i = 0; i < nlayerIn; i++)
{
for (int j = 0; j < nlayerOut; j++)
{
weight[i, j] = (bheta * weight[i, j]) / absolut[i];
}
}

/* RANDOMIZE BIAS */
max = bheta;
min = -bheta;

for (int j = 0; j < nlayerOut; j++)
{
bias[j] = rd.NextDouble() * (max - min) + min;
}

/* COMBINE BIAS AND WEIGHT */
Double[,] biasWeight = newDouble[nlayerIn + 1, nlayerOut];
for (int i = 0; i < nlayerIn + 1; i++)
{
for (int j = 0; j < nlayerOut; j++)
{
if (i == 0) biasWeight[0, j] = bias[j];
else biasWeight[i, j] = weight[i - 1, j];
}
}
```

```
return biasWeight;
}

publicDouble[,] getV()
{
returnthis.v;
}

publicDouble[,] getW()
{
returnthis.w;
}

publicDouble getRMSE()
{
returnthis.RMSE;
}

publicint getTDCount()
{
returnthis.TD.Count;
}

publicoverridestring ToString()
{
return detailResult;
}

publicvoid saveToFile()
{
try
{
String tmp = "";
for (int i = 0; i < v.GetLength(0); i++)
{
for (int j = 0; j < v.GetLength(1); j++)
{
tmp += v[i, j] + ";";
}
}
tmp += "%";
for (int i = 0; i < w.GetLength(0); i++)
{
for (int j = 0; j < w.GetLength(1); j++)
{
tmp += w[i, j] + ";";
```

```
}}
System.IO.File.WriteAllText(Settings.Default.appDataLoc   +
"/db/trained_net.ia", tmp);
}
catch (Exception exc)
{
MessageBox.Show(exc.Message, "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
}}}}
```

*Source Code*4.6Implementasi kelas *NeuralNet*

Untuk menyimpan *training data*pada saat melakukan pelatihan JST, digunakan kelas *TrainingData*. Adapun implementasi dari kelas ini dapat dilihat pada Gambar 4.7 dan *Source Code*4.7.



Gambar 4.7Diagram UML Kelas *TrainingData*

TrainingData.cs

```
using System;

namespace ImageAnnotation
{
classTrainingData
{
publicDouble[] Input { get; set; }

publicDouble[] Target { get; set; }
}
}
```

*Source Code*4.7 Implementasi kelas *TrainingData*

4.2.2.6. Implementasi Penyimpanan Hasil JST

Tahap ini dilakukan setelah proses pelatihan jaringan syaraf tiruan (JST) berhasil dilakukan sehingga didapatkan nilai bobot akhir dari konfigurasi JST tersebut. Pada tahap ini, nilai bobot akhir tersebut disimpan pada sebuah berkas yang sudah ditentukan. Implementasi dari *interface* ini dapat dilihat pada Gambar 4.8 dan *Source Code*4.8.

Gambar 4.8 Diagram UML *Interface iFile*

iFile.cs

```
namespace ImageAnnotation{
interfaceiFile{
void saveToFile();
}
}
```

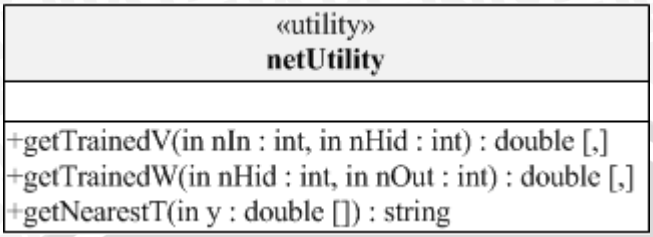
*Source Code*4.8Implementasi *Interface iFile*

4.2.3. Implementasi Pengujian JST

Setelah data latih berhasil dilatih, maka proses selanjutnya adalah menguji hasil pelatihan jaringan syaraf tiruan tersebut. Adapun proses-proses yang dilakukan pada pengujian hasil JST ini antara lain proses pengambilan data hasil pelatihan JST, proses klasifikasi citra, dan proses pemberian label pada citra.

4.2.3.1. Implementasi Pengambilan Data JST

Implementasi dari kelas yang digunakan untuk mengambil pengetahuan hasil proses pelatihan JST dapat dilihat pada Gambar 4.9 dan *Source Code*4.9.



Gambar 4.9 Diagram UML *netUtility*

netUtility.cs

```
using System;
using System.IO;
using System.Linq;
using System.Windows.Forms;
using ImageAnnotation.Properties;

namespace ImageAnnotation.Utility
{
classnetUtility
{
publicstaticDouble[,] getTrainedV(int nIn, int nHid)
{
Double[,] weight = newDouble[nIn, nHid];
try
{
StreamReader                    streamReader                    =
newStreamReader(Settings.Default.appDataLoc                     +
"/db/trained_net.ia");
String text = streamReader.ReadToEnd();
streamReader.Close();
int divider = text.IndexOf('%');
text = text.Substring(0, divider);
String[] my_V = text.Split(';');
int a = 0;

for (int i = 0; i < nIn; i++)
{
for (int j = 0; j < nHid; j++)
{
weight[i, j] = Double.Parse(my_V[a]);
a++;
}
}
}
```

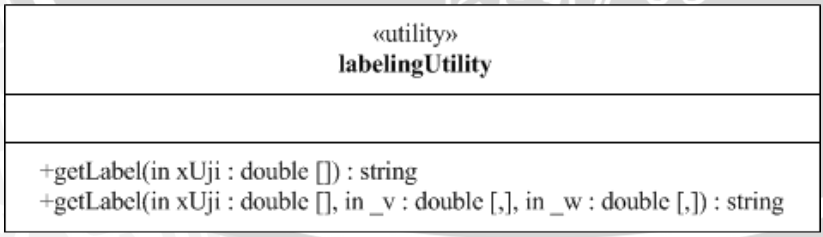
```
catch (Exception exc)
{
MessageBox.Show(exc.Message + Environment.NewLine +
"Application will exit...", "Fatal Error when finding
trained data", MessageBoxButtons.OK, MessageBoxIcon.Error);
Application.Exit();
}
return weight;
}

publicstaticDouble[,] getTrainedW(int nHid, int nOut)
{
Double[,] weight = newDouble[nHid, nOut];
try
{
StreamReader streamReader =
newStreamReader(Settings.Default.appDataLoc +
"/db/trained_net.ia");
String text = streamReader.ReadToEnd();
streamReader.Close();
int divider = text.IndexOf('%') + 1;
text = text.Substring(divider, text.Length - divider);
String[] tx = text.Split(';');
int a = 0;

for (int i = 0; i < nHid; i++)
{
for (int j = 0; j < nOut; j++)
{
weight[i, j] = Double.Parse(tx[a]);
a++;
}
}
}
catch
{
}
return weight;
}

publicstaticString getNearestT(Double[] y)
{
StreamReader
streamReader=newStreamReader(Settings.Default.appDataLoc+"/
db/label_0.ia");
String text = streamReader.ReadToEnd();
```

```
streamReader.Close();
String[] tx = text.Split('\n');
Double minDist = 9999;
String klas = "";

for (int i = 0; i < tx.Length; i++)
{
if (tx[i].Contains('>'))
{
String[] tmp = tx[i].Trim().Split('>');
Double tmpDist = 0;
for (int a = 0; a < tmp[0].Length; a++)
{
Double t = Double.Parse(tmp[0][a].ToString());
tmpDist += Math.Abs(y[a] - t);
}

//Console.WriteLine(i+"    "+tmpDist);
if (tmpDist < minDist)
{
minDist = tmpDist;
klas = tmp[1];
}
}
}
return klas;
}
}
}
```

*Source Code*4.9 Implementasi Kelas *netUtility*

4.2.3.2. Implementasi Klasifikasi Citra

Implementasi dari kelas yang digunakan untuk melakukan klasifikasi citra uji dapat dilihat pada Gambar 4.10 dan *Source Code*4.10.



Gambar 4.10Diagram UML *labelingUtility*

LabelingUtility.cs

```
namespace ImageAnnotation.Utility
{
staticclassLabelingUtility
{

publicstaticString getLabel(Double[] xUji)
{
// FEEDFORWARD
int nInput = 49, nHidden = NNsetting.Default.hiddenNeuron,
nOutput = 30;

Double[] z_in = newDouble[nHidden];
Double[] z = newDouble[nHidden];
Double[] y_in = newDouble[nOutput];
Double[] y = newDouble[nOutput];
Double[,] v=netUtility.getTrainedV(nInput + 1, nHidden);
Double[,]w=netUtility.getTrainedW(nHidden + 1, nOutput);

for (int j = 0; j < nHidden; j++)
{
z_in[j] = 0;
for (int i = 1; i <= nInput; i++)
{
z_in[j] += xUji[i - 1] * v[i, j];
}
z_in[j] += v[0, j];
z[j] = 1.0 / (1.0 + Math.Exp(z_in[j]));
}

for (int k = 0; k < nOutput; k++)
{
y_in[k] = 0;
for (int j = 1; j <= nHidden; j++)
{
y_in[k] += z[j - 1] * w[j, k];
}
y_in[k] += w[0, k];
y[k] = 1.0 / (1.0 + Math.Exp(y_in[k]));
}

// GET CLOSEST TARGET
String terdekat = (netUtility.getNearestT(y));
return terdekat;
```

```
}

publicstaticString  getLabel(Double[]  xUji,  Double[,]  _v,
Double[,] _w)
{
// FEEDFORWARD
int nInput = 49, nHidden = 49, nOutput = 30;
Double[] z_in = newDouble[nHidden];
Double[] z = newDouble[nHidden];
Double[] y_in = newDouble[nOutput];
Double[] y = newDouble[nOutput];
Double[,] v = _v;
Double[,] w = _w;

for (int j = 0; j < nHidden; j++)
{
z_in[j] = 0;
for (int i = 1; i <= nInput; i++)
{
z_in[j] += xUji[i - 1] * v[i, j];
}
z_in[j] += v[0, j];
z[j] = 1.0 / (1.0 + Math.Exp(z_in[j]));
}

for (int k = 0; k < nOutput; k++)
{
y_in[k] = 0;
for (int j = 1; j <= nHidden; j++)
{
y_in[k] += z[j - 1] * w[j, k];
}
y_in[k] += w[0, k];
y[k] = 1.0 / (1.0 + Math.Exp(y_in[k]));
}

// GET CLOSEST TARGET
String terdekat = (netUtility.getNearestT(y));
return terdekat;
}
}
}
```

*Source Code*4.10 Implementasi kelas *LabelingUtility*

4.3. Implementasi Antarmuka

Berdasarkan pada rancangan antarmukayang telah dikemukakan pada bab 3, maka dihasilkan antarmuka dari aplikasi sistem anotasi citra otomatis yang terdiri dari beberapa *form*, antara lain: *Form* Utama, *Form Folder Manager*, *Form Training Data Manager*, *Form Preferences*, *Form Verbose Mode*, *Form Wavelet Transformation*, *Form Neural Network Training*, *Form Labeling Progress*, *Form Labeling Result*, dan *Form Searching*.

Form utama pada aplikasi ini digunakan untuk menampilkan citra yang akan menjadi data uji dari hasil pelatihan JST. Pada *form* utama ini, terdapat tombol-tombol yang dapat digunakan untuk menjalankan beberapa perintah seperti melakukan ekstraksi citra, menjalankan proses pelatihan jaringan syaraf tiruan, dan lain-lain. Pada *form* utama ini juga terdapat tombol ataupun menu yang dapat digunakan untuk membuka beberapa form penunjang yang lainnya. *Form* utama juga digunakan untuk menampilkan beberapa informasi yang berkaitan dengan citra dan informasi tentang proses yang sudah dilakukan.*Form* utama pada rancangan antarmuka dari aplikasi ini dapat dilihat pada Gambar 4.11.

Gambar 4.11 Antarmuka *Form* Utama

Keterangan dari gambar antarmuka *form* utama adalah sebagai berikut:

1. *Menubar* yang memiliki beberapa menu dan sub menu untuk menjalankan beberapa perintah pada aplikasi ini.
2. *Toolbar*, untuk menempatkan beberapa perintah yang dapat dijalankan pada aplikasi.
3. *Container* yang digunakan untuk menampilkan semua citra berdasarkan *tag* yang dimiliki oleh masing-masing citra.
4. *Panel* yang digunakan untuk menampilkan informasi *tag* yang dimiliki citra.
5. *Groupbox* yang digunakan untuk menempatkan beberapa tombol dari proses pelatihan jaringan syaraf tiruan.
6. *Field* yang digunakan untuk menampilkan informasi tentang konfigurasi jaringan syaraf tiruan.
7. *Field* yang digunakan untuk menampilkan informasi hasil evaluasi sistem (*correctness*, *precision*, dan *recall*).
8. *Searchbox* yang digunakan untuk melakukan pencarian informasi yang terkandung pada citra.
9. *Field* yang digunakan untuk menampilkan informasi dari metadata yang dimiliki oleh citra.
10. *Box* yang digunakan untuk menampilkan informasi histogram dari setiap citra.

Semua tampilan dan cara kerja dari *form* utama akan diatur oleh sebuah pengaturan yang disimpan oleh pengguna. Untuk melakukan pengaturan pada aplikasi ini, maka disediakan form *Preferences*. Form ini digunakan untuk menangani nilai-nilai untuk mengatur tampilan dan cara kerja aplikasi. Pada form *Preferences*ini terdapat 4 *subform* antara lain *general* yang digunakan untuk menyimpan nilai pengaturan umum, *neural* digunakan untuk menyimpan nilai pengaturan yang berkaitan dengan jaringan syaraf tiruan, *extractions* digunakan untuk menyimpan nilai pengaturan untuk proses ekstraksi fitur dari citra, dan *advanced* digunakan untuk menyimpan nilai-nilai pengaturan yang lebih lanjut. Adapun tampilan antramuka dari form *Preferences*dapat dilihat pada Gambar 4.12.

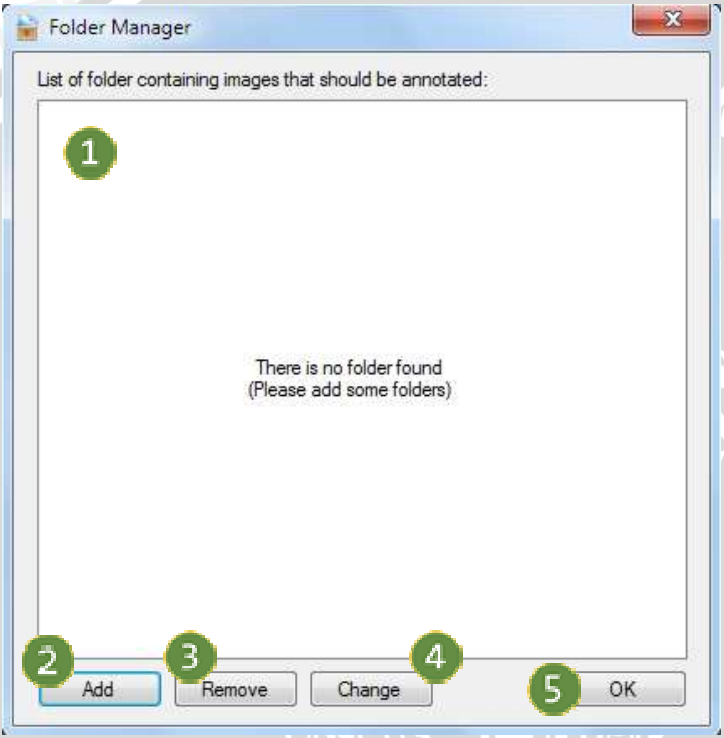
Gambar 4.12 Antarmuka *Form Preferences*

Keterangan:

1. Merupakan *panel* untuk menempatkan 4 menu (*general, neural, extractions*, dan *advanced*) yang ada pada form *preferences*.
2. Merupakan *panel* yang digunakan untuk menampilan beberapa *control* yang digunakan untuk menangani masukan dari pengguna sesuai dengan menu yang dipilih.
3. Merupakan *panel* yang digunakan untuk menempatkan tombol *help, OK,* dan *cancel*.

Pada aplikasi ini, lokasi dari *folder* tempat citra uji dan citra latih yang akan digunakan untuk proses pelatihan jaringan syaraf tiruan dan proses klasifikasi citra disimpan pada sebuah *file* dengan ekstensi tertentu yang sudah didefinisikan. Untuk menyimpan informasi tersebut, maka disediakan dua *form* yang dapat digunakan untuk memasukkan alamatdari *folder* tempat citra yang akan

dijadikan data latih atau data uji. Adapun *form* tersebut adalah *form folder manager*yang digunakan untuk memasukkan alamat citra uji dan *form training data manager*yang digunakan untuk memasukkan alamat citra latih. Adapun tampilan dari *form folder manager* dapat dilihat pada Gambar 4.13 dan untuk *form training data manager*dapat dilihat pada Gambar 4.14.

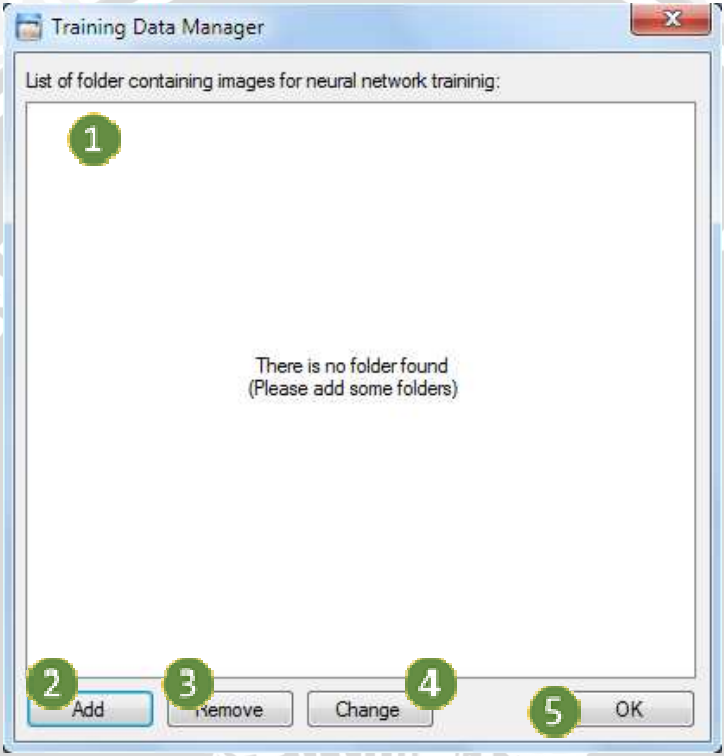


Gambar 4.13 Antarmuka *Form Folder Manager*

Keterangan:

1. *ListBox* yang digunakan untuk menampilkan alamat *folder* dari citra-citra yang digunakan sebagai data uji.
2. Tombol yang digunakan untuk menambahkan alamat *folder* dari citra-citra yang akan digunakan sebagai data uji.
3. Tombol yang digunakan untuk menghapus alamat *folder* dari citra uji yang akan disimpan oleh sistem.

4. Tombol yang digunakan untuk mengubah alamat *folder* yang sudah tersimpan.
5. Tombol yang digunakan untuk menyimpan semua perubahan alamat *folder* dan menyimpannya pada berkas tertentu.



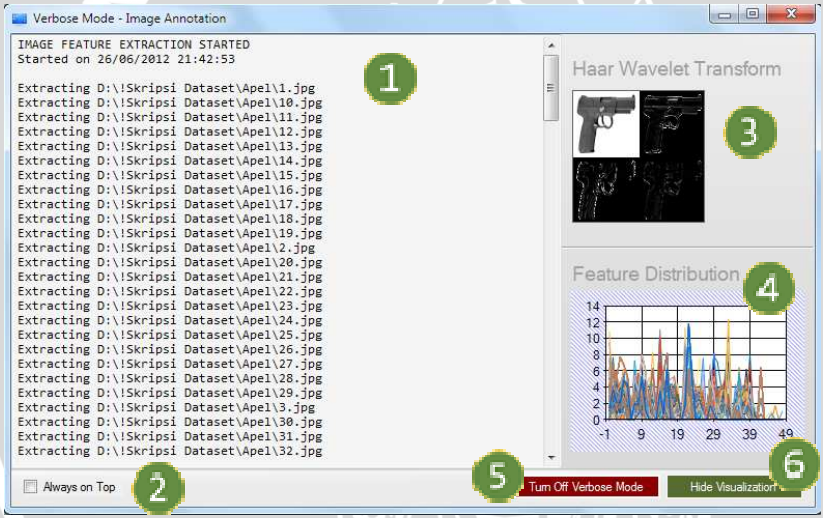
Gambar 4.14 Antarmuka *Form Training Data Manager*

Keterangan:

1. *ListBox* yang digunakan untuk menampilkan alamat *folder* dari citra-citra yang digunakan sebagai data latih.
2. Tombol yang digunakan untuk menambahkan alamat *folder* dari citra-citra yang akan digunakan sebagai data latih.
3. Tombol yang digunakan untuk menghapus alamat *folder* dari citra latih yang akan disimpan oleh sistem.
4. Tombol yang digunakan untuk mengubah alamat *folder* yang sudah tersimpan.

5. Tombol yang digunakan untuk menyimpan semua perubahan alamat *folder* dan menyimpannya pada berkas tertentu.

Seperti yang sudah dijelaskan pada bab 3, proses ekstraksi fitur dari citra adalah dengan cara mengubah domain citra dari domain spasial menjadi domain frekuensi kemudian menerapkan *Sliding-Window* untuk mendapatkan nilai fitur untuk masing-masing citra. Pada antarmuka aplikasi ini, ketika *verbose mode* diaktifkan, maka aplikasi akan menampilkan antramuka yang berisi detail informasi dari proses yang sedang dilakukan. Adapun antarmuka *verbose mode* dapat dilihat pada Gambar 4.15 untuk tampilan detil proses pada saat ekstraksi citra, dan Gambar 4.16 untuk tampilan detil proses pada saat pelatihan jaringan syaraf tiruan.

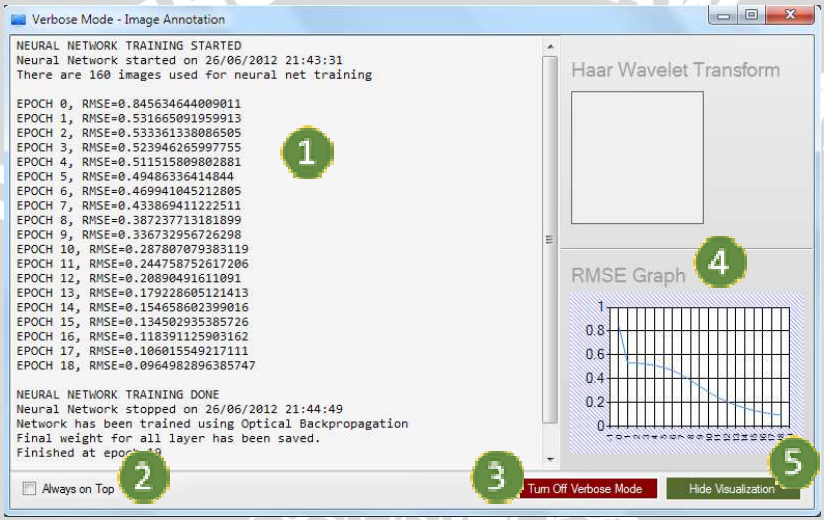


Gambar 4.15 Antarmuka *Form* DetilEkstraksi Fitur

Keterangan:

1. *Fieldtext* yang digunakan untuk menampilkan detil proses yang dilakukan ketika menjalankan proses ekstraksi fitur dari citra.
2. *Checkbox* yang digunakan untuk membuat *form verbose mode* berada pada paling atas dari form yang lainnya.

3. *Picturebox* yang digunakan untuk menampilkan citra hasil proses transformasi *Wavelet*.
4. *Graph*yang digunakan untuk menampilkan sebaran vektor dari fitur hasil proses ekstraksi fitur citra.
5. Tombol *Turn Off Verbose Mode*, digunakan untuk menonaktifkan *verbose mode* pada aplikasi.
6. Tombol *Hide Visualization*, digunakan untuk menyembunyikan visualisasi dari detil proses yang sedang berlangsung.



Gambar 4.16 Antarmuka *Form* Detil Pelatihan JST

Keterangan:

1. *Fieldtext* yang digunakan untuk menampilkan detil proses yang dilakukan ketika menjalankan proses pelatihan jaringan syaraf tiruan.
2. *Checkbox* yang digunakan untuk membuat *form verbose mode* berada pada paling atas dari form yang lainnya.
3. Tombol *Turn Off Verbose Mode*, digunakan untuk menonaktifkan *verbose mode*pada aplikasi
4. *Graph* yang digunakan untuk menampilkan visualisasi dari nilai RMSE yang didapatkan pada tiap *epoch*.

5. Tombol *Hide Visualization*, digunakan untuk menyembunyikan visualisasi dari detil proses yang sedang berlangsung.

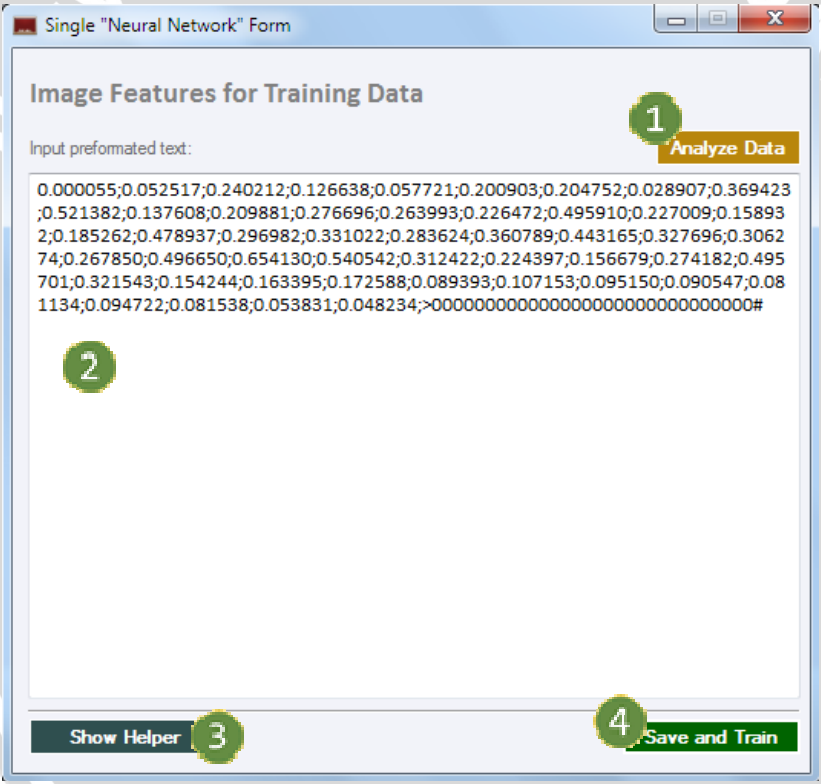
Pada aplikasi ini, juga diberikan fasilitas tambahan untuk menjalankan proses-proses tertentu secara terpisah sehingga disediakan *form* tersendiri untuk menjalankan proses-proses tersebut. Adapun proses-proses yang dimaksud adalah: Proses transformasi *wavelet*, proses pelatihan JST, melihat sebaran fitur hasil ekstraksi citra, dan pengujian performa dari arsitektur JST yang dibuat.

Gambar 4.17 Antarmuka *Form* Transformasi *Wavelet*

Keterangan:

1. Tombol *OpenImage*, digunakan untuk membuka citra yang akan dilakukan proses transformasi *Wavelet*

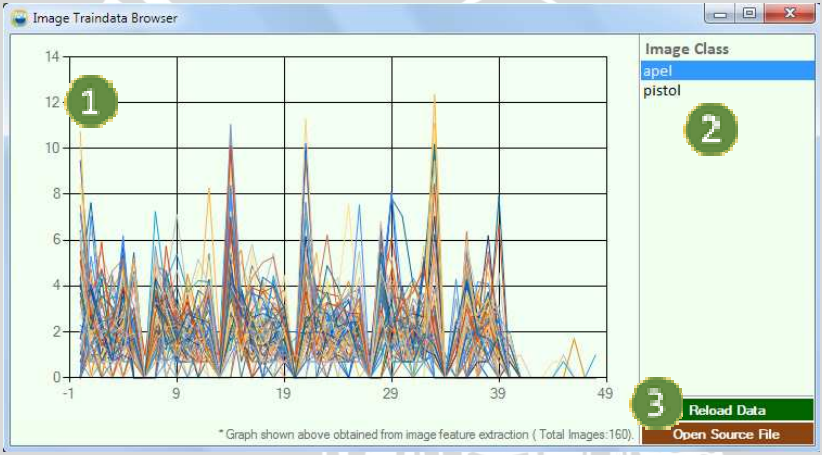
2. *Field* yang digunakan untuk menampilkan citra, dan juga untuk menerima masukan *drag and drop* dari pengguna.
3. Tombol *Transform*, digunakan untuk menjalankan proses transformasi wavelet terhadap citra yang dimasukkan.
4. *Combobox* yang digunakan untuk memilih level transformasi dari proses transformasi *Wavelet*yang akan dilakukan.
5. *Checkbox* yang digunakan untuk memilih apakah akan mengaktifkan mode *binarization* pada saat melakukan proses transformasi *Wavelet*.
6. *Trackbar* yang digunakan untuk menentukan *threshold* dari *binarization* ketika mode*binarization* diaktifkan.



Gambar 4.18 Antarmuka *Form* Pelatihan JST

Keterangan:

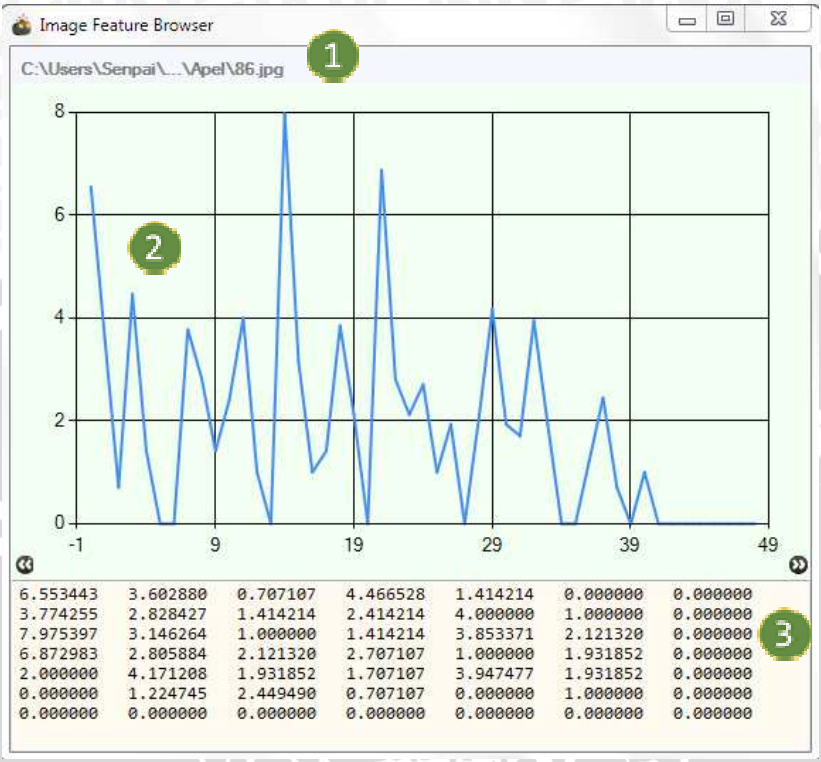
1. Tombol *AnalyzeData*, digunakan untuk melakukan analisa dari data input yang dimasukkan untuk proses pelatihan jaringan syaraf tiruan.
2. *Fieldtext* yang digunakan untuk memasukkan data input untuk proses pelatihan jaringan syaraf tiruan.
3. Tombol *Show Helper*, digunakan untuk menampilkan *helper* yang bisa digunakan untuk mempermudah pemasukan data input.
4. Tombol yang digunakan untuk menyimpan data input dan kemudian menjalankan proses pelatihan jaringan syaraf tiruan.



Gambar 4.19 Antarmuka *Form* untuk Melihat Sebaran Fitur

Keterangan:

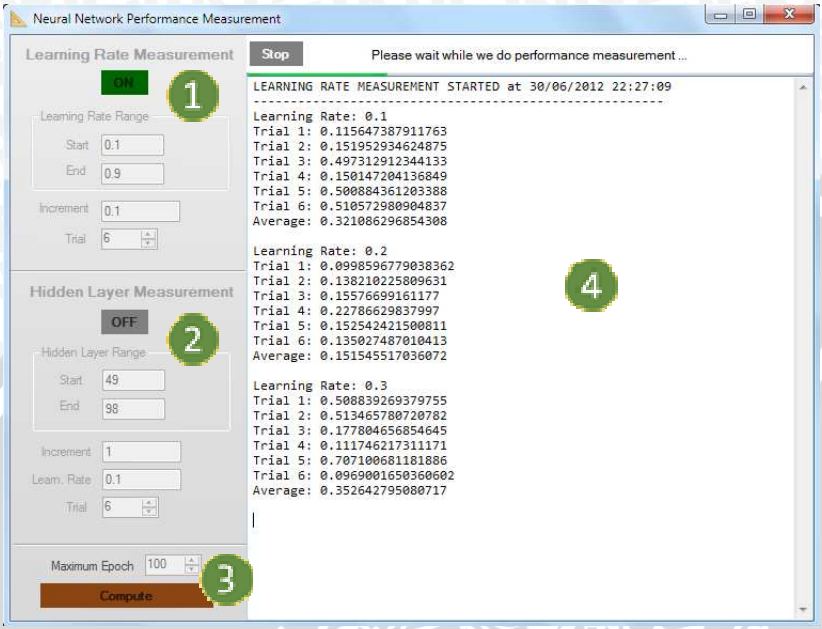
1. *Graph* yang digunakan untuk menampilkan sebaran vektor dari fitur semua citra untuk kelas tertentu.
2. *Listbox Image Class*, digunakan untuk memilih kelas yang sebaran fitur citra-citranya akan ditampilkan dalam bentuk grafik.
3. Tombol-tombol yang digunakan untuk memuat ulang data dan membuka data dari sumber yang berbeda.



Gambar 4.20Antarmuka *Image Feature Browser*

Keterangan:

1. *Label* yang digunakan untuk menampilkan alamat berkas dari citra yang sedang ditampilkan sebaran vektor fiturnya.
2. *Graph* yang digunakan untuk menampilkan sebaran vektor fitur dari sebuah citra tertentu dalam bentuk grafik.
3. *Fieldtext* yang digunakan untuk menampilkan vektor fitur dari sebuah citra tertentu dalam bentuk teks.



Gambar 4.21Antarmuka *Performance Measurement*

Keterangan:

1. *Groupbox Learning Rate Measurement*, digunakan untuk menempatkan beberapa *user control*yang berfungsi untuk menerima nilai masukan untuk proses pengujian laju pembelajaran.
2. *GroupboxHidden Layer Measurement*, digunakan untuk menempatkan beberapa *user control* yang berfungsi untuk menerima nilai masukan untuk proses pengujian jumlah *neuron* pada *hidden layer*.
3. Tombol yang digunakan untuk menjalankan proses *measurement*.
4. *Fieldtext* yang digunakan untuk menampilkan hasil dari proses *measurement* yang sedang dilakukan.

Gambar 4.22 Antarmuka Hasil Pelabelan

Keterangan:

1. Tombol yang digunakan untuk menyetujui hasil dari proses klasifikasi citra. Ketika tombol ini diklik, maka hasil klasifikasi citra akan disimpan pada *metadata* citra tersebut.
2. Tombol yang digunakan untuk menolak hasil dari proses klasifikasi citra. Ketika tombol ini diklik, maka hasil klasifikasi citra tidak akan disimpan pada metadata citra tersebut.
3. *Container* yang digunakan untuk menampilkan citra dan hasil klasifikasi dari citra tersebut.
4. *Panel* yang digunakan untuk menampilkan tingkat akurasi dari hasil klasifikasi citra.

Gambar 4.23 Antarmuka Pencarian Citra

Keterangan:

1. *Fieldtext* yang digunakan untuk memasukkan kata kunci pencarian dari citra.
2. *Container* yang digunakan untuk menampilkan citra-citra yang telah dimasukkan pada sistem.

4.4. Sistematika Pengujian

Sesuai dengan penjelasan pada bab 3 tentang rancangan pengujian, maka pada sub bab ini akan dijelaskan mengenai sistematika pengujian dari3 macam pengujian untuk sistem yang telah dikembangkan. Adapun ketiga pengujian tersebut antara lain pengujian laju pembelajaran, pengujian jumlah *neuron* pada lapisan tersembunyi (*hidden layer*), dan pengujian tingkat akurasi dari sistem anotasi citra.

4.4.1. Sistematika Uji Laju Pembelajaran

Pengujian yang pertama adalah menguji pengaruh pengubahan nilai laju pembelajaran terhadap nilai RMSE yang didapatkan. Pengujian ini bertujuan untuk mengetahui nilai laju pembelajaran terbaikyang dapat digunakan untuk melakukan pelatihan data latih dari fitur-fitur yang telah diekstraksi dari citra.

Pada pengujian ini akan digunakan 160 citra yang terdiri dari dua kelas (yaitu kelas apel dan pistol) yang digunakan sebagai data latih.

Contoh citra dari kelas yang digunakan sebagai data latih ini dapat dilihat pada Gambar 4.24, sedangkan untuk semua gambar yang ada pada kelas apel dan pistol ini dapat dilihat pada Lampiran 1 dimana citra dijadikan sebagai data latih dan pada

Lampiran 2 dimana citra dijadikan sebagai data uji.

Adapun proses pengujian yang dilakukan adalah membandingkan nilai RMSE yang didapatkan dari proses pelatihan jaringan syaraf tiruan dengan menggunakan nilai laju pembelajaran(α) yang berbeda-beda. Percobaan pengujian laju pembelajaran dilakukan sebanyak 6 kali dengan nilai laju pembelajaran terletak antara rentang0.0 - 1.0. Pada masing-masing percobaan tersebut, akan dilakukan proses pelatihan jaringan syaraf tiruan sebanyak 100*epoch* dengan nilai RMSE minimalharapan sebesar 0.0001.

Gambar 4.24 Data Latih Kelas Apel dan Pistol

Pada pengujian ini, faktor kecepatan jaringan syaraf tiruan untuk mencapai nilai minimum *error* tidak dijadikan faktor yang akan dianalisis. Nilai yang digunakan untuk menguji laju pembelajaran adalah nilai RMSE akhir yang diperoleh pada saat *epoch* ke-100.

Dari hasil keenam percobaan yang telah dilakukan, akan dihitung nilai rata-ratanya dan dibandingkan satu persatu, sehingga dapat diketahui nilai laju pembelajaran yang menghasilkan nilai RMSE rata-rata yang paling kecil.

4.4.2. Sistematika Uji Lapisan Tersembunyi

Pengujian yang kedua adalah menguji pengaruh pengubahan jumlah*neuron*pada konfigurasi lapisan tersembuyi.Parameter yang digunakan pada pengujian ini adalah jumlah *neuron* pada lapisan tersembunyi dan nilai *Root Mean Square Error(RMSE)* yang didapatkan dari proses pelatihan jaringan syaraf tiruan.

Pengujian ini dilakukan untuk mengetahui pengaruh pengubahan jumlah *neuron* pada lapisan tersembunyi terhadap nilai RMSE yang didapatkan dari proses pelatihan jaringan syaraf tiruan.

Data latih yang digunakan pada pengujian ini adalah sama dengan data latih yang digunakan pada pengujian laju pembelajaran, yaitu 160 citra latih yang terdiri dari dua kelas (yaitu kelas apel dan kelas pistol).

Proses pengujian dilakukan dengan cara membandingkan nilai RMSE yang didapatkan dari proses pelatihan JST dengan menggunakan konfigurasi nilai jumlah *neuron* yang berbeda-beda pada lapisan tersembunyi. Percobaan pengujian jumlah neuron ini akan dilakukan sebanyak 6 kali dengan nilai jumlah neuron pada lapisan tersembunyi terletak pada tentang 49 – 98. Nilai rentang tersebut didapatkan dari jumlah neuron pada lapisan masukan dan dua kali dari nilai tersebut.

Pada masing-masing percobaan untuk jumlah neuron yang berbeda-beda, akan dilakukan proses pelatihan JST sebanyak 100 *epoch* dengan nilai RMSE minimal harapan sebesar 0.0001, dan nilai laju pembelajaran sebesar 0.1. Nilai yang digunakan untuk analisa pada pengujian ini adalah nilai RMSE yang didapatkan pada *epoch* ke-100.

Dari hasil keenam percobaan yang telah dilakukan, akan dihitung nilai rata-ratanya,kemudian dibandingkan satu persatu sehingga akan diketahui nilai rata-rata RMSE terbaik untuk masing-masing parameter pengujian.

4.4.3. Sistematika Uji Akurasi Sistem Anotasi Citra

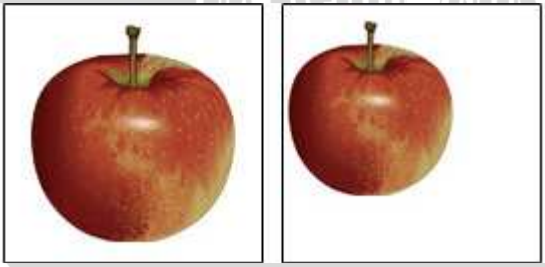
Pengujian yang terakhir adalah menguji tingkat akurasi dari sistem anotasi citra yang telah dibangun.Pada pengujian tingkat akurasi sistem anotasi citra ini akan dilakukan 2 pengujian, yaitu pengujian pengaruh jumlah data latih terhadap tingkat akurasi sistem dan pengujian pengaruh jumlah kelas pada data latih terhadap tingkat akurasi sistem.

Pada pengujian pertama, yaitu pengujian pengaruh jumlah data latih, pengujian dilakukan dengan cara mengubah jumlah data latih yang digunakan untuk melakukan pelatihan pada jaringan syaraf tiruan. Adapun parameter yang digunakan pada pengujian ini adalah jumlah data latih, *recall*, *precision*, *F-Measure*, dan waktu yang

digunakan untuk pelatihan. Pengujian ini dilakukan dengan melakukan percobaan beberapa kali menggunakan jumlah data latih yang berbeda-beda yang terletak pada rentang 20 – 160.

Pengujian nilai *F-Measure*ini akan dilakukan menggunakan variasi kelas yang berbeda-beda. Variasi kelas yang pertama adalah kelas dari citra yang memiliki perbedaan mencolok antar citra pada kelas yang lain, tetapi memiliki kesamaan antar citra pada kelas yang sama. Variasi kelas yang kedua adalah kelas dari citra hasil proses pengeditan (pengubahan ukuran dan penggeseran). Pada variasi kelas yang kedua ini, citra memiliki perbedaan yang mencolok baik itu pada kelas yang sama maupun pada kelas lain. Variasi kelas yang terakhir adalah kelas dari citra yang memiliki kemiripan yang mencolok, baik itu pada kelas yang sama maupun pada kelas yang lain. Adapun contoh citra dari variasi kelas pertama adalah citra apel dan pistol yang dapat dilihat pada Gambar 4.24, sedangkan contoh citra dari variasi kelas yang kedua adalah citra apel dan citra apel hasil pengeditan yang dapat dilihat pada Gambar 4.25, dan contoh citra dari kelas citra ketiga, yaitu citra yang memiliki kemiripan bentuk, adalah citra bolpoin dan pisau yang dapat dilihat pada Gambar 4.26.

Pada awal pengujian, jumlah data latih yang digunakan adalah sebanyak 20 citra, kemudian data latih tersebut ditambah sebanyak 20 citra untuk percobaan selanjutnya dan nilai *F-Measure* dari masing-masing percobaan tersebut dicatat. Setelah semua percobaan berhasil dilakukan, maka nilai *F-Measure*yang didapatkan pada masing-masing percobaan dibandingkan satu persatu sehingga akan diketahui jumlah data latih dan variasi kelas yang menghasilkan tingkat akurasi terbaik.



Gambar 4.25 Data Latih Kelas Citra Hasil Pengeditan

Gambar 4.26 Data Latih Kelas Bolpoin dan Pisau

Setelah pengujian pertama selesai dilakukan, maka akan dilakukan pengujian yang kedua yaitu pengujian pengaruh jumlah kelas pada data latih terhadap tingkat kebenaran sistem.Pengujian ini dilakukan dengan cara mengubah jumlah kelas pada data latih. Parameter-parameter yang digunakan pada pengujian ini antara lain jumlah kelas pada data latih, tingkat kebenaran dari pengenalan, nilai RMSE yang didapatkan pada *epoch* terakhir, dan waktu yang digunakan untuk melakukan pelatihan JST.

Pada pengujian ini akan dilakukan percobaan dengan menggunakan jumlah kelasdata latih yang berbeda-beda, yaitu menggunakan jumlah kelas sebanyak 2,3, dan 4. Adapun kelas yang digunakan pada pengujian ini adalah kelas apel, pistol, bolpoin, dan pisau. Setelah semua percobaan berhasil dilakukan maka nilai semua parameter hasil pengujian dicatat dan dibandingkan satu persatu untuk masing-masing parameter, sehingga akan diketahui pengaruh dari pengubahan jumlah kelas terhadap parameter-parameter pada pengujian ini.

4.5. Implementasi Uji Coba

Pada sub bab ini akan dibahas mengenai implementasi dari uji coba untuk sistem anotasi citra sesuai dengan sistematikapengujian yang telah dipaparkan.

4.5.1. Pengujian Laju Pembelajaran

Sesuai dengan sistematika pengujian laju pembelajaran (α) yang telah dijelaskan, pengujian dilakukan sebanyak 6 kali untuk masing-masing nilai laju pembelajaran. Proses pengujian dimulai dari nilai laju pembelajaran paling kecil sampai laju pembelajaran paling besar, yaitu dari 0.1 – 0.9, dengan perubahan laju

pembelajaran sebesar 0.1 untuk setiap percobaan. Hasil dari pengujian laju pembelajaran ditunjukkan pada Tabel 4.1.

Tabel 4.1Pengujian Laju Pembelajaran

α	RMSE Percobaan ke-						$\overline{RMSE}$
	1	2	3	4	5	6	
0.1	0.0005	0.0012	0.0007	0.0005	0.0007	0.0009	0.00075
0.2	0.0005	0.0003	0.0003	0.0006	0.0002	0.0006	0.00039
0.3	0.0004	0.0003	0.0002	0.0002	0.0002	0.0000	0.00023
0.4	0.0006	0.0006	0.0002	0.006	0.0003	0.0003	0.00134
0.5	0.7071	0.7071	0.0004	0.7071	0.7071	0.7071	0.58932
0.6	0.7064	0.7071	0.7071	0.7071	0.7071	0.7071	0.70698
0.7	0.7071	0.7071	0.7071	0.7071	0.7071	0.7071	0.70711
0.8	0.7071	0.7071	0.7071	0.7072	0.7071	0.7071	0.70712
0.9	0.7071	0.7071	0.7071	0.7072	0.7073	0.7071	0.70715

Keterangan:
α = laju pembelajaran
$\overline{RMSE}$ = rata-rata RMSE dari semua percobaan

Pada Tabel 4.1dapat dilihat bahwa nilai *Root Mean Square Error (RMSE)* terkecil adalah 0.00023. Nilai RMSE terkecil inididapatkan pada saat nilai laju pembelajaran (α) 0.3. Nilai RMSE yang dihasilkan pada saat laju pembelajaran 0.1, 0.2, dan 0.4 mendekati nilai RMSE yang terkecil, dan mulai dari nilai laju pembelajaran 0.5 sampai 0.9 nilai RMSE yang didapatkan semakin menjauhi nilai RMSE terkecil, dan ketika nilai laju pembelajaran berada pada rentang 0.7 – 0.8 nilai RMSE yang dihasilkan adalah relatif sama sebelum dilakukan pembulatan hasil RMSE.

4.5.2. Pengujian Jumlah *Neuron*Lapisan Tersembunyi

Sesuai dengan sistematika pengujian jumlah lapisan tersembunyi, pengujian dilakukan sebanyak 6 percobaan untuk masing-masing nilai jumlah *neuron.* Proses pengujian dimulai dari jumlah *neuron* 49, sampai dengan dua kali nilai jumlah *neuron* pada lapisan masukan pada JST yaitu 98. Adapun hasil dari proses pengujian ini dapat dilihat pada Tabel 4.2

Tabel 4.2 Pengujian Jumlah Lapisan Tersembunyi

h	RMSE Percobaan ke-						$\overline{RMSE}$
	1	2	3	4	5	6	
49	0.0015	0.0012	0.0007	0.0008	0.0011	0.0011	0.0011
50	0.0007	0.0022	0.0019	0.0006	0.0007	0.0018	0.0013
51	0.0007	0.0011	0.0012	0.0009	0.0009	0.0000	0.0009
52	0.0006	0.002	0.0005	0.0017	0.0005	0.0026	0.0013
53	0.0007	0.001	0.0006	0.0008	0.0006	0.0015	0.0009
54	0.0006	0.0005	0.0006	0.0006	0.0004	0.001	0.0006
55	0.0007	0.0016	0.002	0.001	0.0005	0.0017	0.0012
56	0.0004	0.0007	0.0006	0.0002	0.0004	0.0009	0.0005
57	0.002	0.0009	0.0024	0.0004	0.0012	0.0004	0.0012
58	0.0004	0.0003	0.0008	0.0012	0.0006	0.0011	0.0007
59	0.0012	0.0008	0.0004	0.0008	0.0005	0.0009	0.0008
60	0.0008	0.0006	0.0005	0.0006	0.0011	0.0006	0.0007
61	0.0005	0.0004	0.0006	0.0006	0.0005	0.0008	0.0006
62	0.0005	0.0003	0.0012	0.0005	0.0006	0.0004	0.0006
63	0.0011	0.001	0.0004	0.0005	0.0009	0.0005	0.0007
64	0.0005	0.0015	0.0007	0.0009	0.0003	0.0004	0.0007
65	0.0005	0.0005	0.0014	0.0011	0.0004	0.0002	0.0007
66	0.0003	0.0007	0.0005	0.0004	0.001	0.0016	0.0007
67	0.0008	0.0003	0.0004	0.0003	0.0005	0.0005	0.0005
68	0.0005	0.0005	0.0006	0.0006	0.0007	0.0007	0.0006
69	0.0011	0.0011	0.0004	0.0005	0.0006	0.0005	0.0007
70	0.0008	0.0006	0.0008	0.0006	0.0006	0.0006	0.0007
71	0.0006	0.0003	0.0003	0.0004	0.0005	0.0006	0.0004
72	0.0009	0.0005	0.0004	0.0003	0.0013	0.0004	0.0007
73	0.0004	0.0004	0.0005	0.0007	0.001	0.0004	0.0006
74	0.0005	0.0003	0.0004	0.0008	0.0006	0.0003	0.0005
75	0.0005	0.0008	0.001	0.0004	0.0006	0.001	0.0007
76	0.0008	0.0003	0.0005	0.0004	0.0003	0.0004	0.0005
77	0.0004	0.0004	0.0007	0.0007	0.0003	0.0004	0.0005
78	0.0003	0.0003	0.0003	0.0005	0.0009	0.0005	0.0005
79	0.0007	0.0002	0.0004	0.0004	0.0006	0.0005	0.0004
80	0.0004	0.0006	0.0008	0.0007	0.0007	0.0006	0.0006
81	0.0002	0.0004	0.0004	0.0005	0.0005	0.0006	0.0004
82	0.0004	0.0005	0.0003	0.0006	0.0002	0.0007	0.0004

83	0.0004	0.0003	0.0005	0.0002	0.0004	0.0003	0.0003
84	0.0004	0.0003	0.0003	0.0004	0.0004	0.0004	0.0004
85	0.0016	0.0004	0.0003	0.0004	0.0003	0.0004	0.0006
86	0.0003	0.0004	0.0004	0.0005	0.0002	0.0004	0.0004
87	0.0004	0.0005	0.0003	0.0003	0.0003	0.0005	0.0004
88	0.0006	0.0003	0.0003	0.0006	0.0005	0.0006	0.0005
89	0.0003	0.0005	0.0009	0.0003	0.0004	0.0003	0.0005
90	0.0004	0.0004	0.0003	0.0006	0.0002	0.0003	0.0004
91	0.0002	0.0003	0.0003	0.0002	0.0004	0.0003	0.0003
92	0.0003	0.0004	0.0004	0.0006	0.0003	0.0003	0.0004
93	0.0005	0.0003	0.0003	0.0004	0.0004	0.0007	0.0004
94	0.0003	0.0003	0.0003	0.0005	0.0003	0.0003	0.0003
95	0.0005	0.0006	0.0002	0.0004	0.0004	0.0007	0.0005
96	0.0004	0.0004	0.0004	0.0003	0.0002	0.0004	0.0004
97	0.0003	0.0005	0.0003	0.0005	0.0003	0.0003	0.0004
98	0.0004	0.0003	0.0004	0.0003	0.0003	0.0003	0.0003

Keterangan:
h = jumlah *neuron*lapisan tersembunyi pada JST
$\overline{RMSE}$ = rata-rata RMSE dari semua percobaan

Pada Tabel 4.2 dapat dilihat bahwa nilai rata-rata RMSE terkecil adalah 0.000305 yang didapatkan pada saat nilai jumlah *neuron* sebesar 91, sedangkan nilai rata-rata RMSE terbesar adalah 0.00134yang didapatkan pada saat nilai jumlah *neuron* sebesar 52. Dari data hasil pengujian, dapat dilihat juga bahwa nilai RMSE yang didapatkan untuk masing-masing jumlah *neuron* memiliki kecenderungan menurun ketika jumlah *neuron* dinaikkan.

4.5.3. Pengujian Tingkat Akurasi Sistem Anotasi Citra

Seperti yang telah dipaparkan pada sistematika pengujian tingkat akurasi sistem, pada pengujian ini akan dilakukan dua macam pengujian yaitu pengujian pengaruh jumlah data latih terhadap akurasi sistem dan pengujian pengaruh jumlah kelas pada data latih terhadap akurasi sistem. Untuk proses pelatihan jaringan syaraf tiruan yang dilakukan pada kedua pengujian tersebut, digunakan nilai laju pembelajaran (α) sebesar 0.2, jumlah neuron pada lapisan

tersembunyi sebesar 49, nilai *error* harapan sebesar 0.0001, dan nilai maksimum *epoch* sebesar 100.

Pada pengujian pertama, yaitu pengujian pengaruh jumlah data latih terhadap akurasi sistem, sistem diuji menggunakan beberapa parameter seperti yang sudah dijelaskan pada sistematika pengujian. Proses pengujian dimulai dengan menggunakan data latih sebanyak 20 citra yang terdiri dari dua kelas (yaitu kelas apel dan pistol), kemudian ditambah 20 citra dengan variasi kelas yang sama untuk percobaan selanjutnya sampai jumlah citra yang digunakan sebagai data latih berjumlah 160. Adapun contoh citra dari kelas yang digunakan pada pengujian ini dapat dilihat pada Gambar 4.24.

Pada pengujian ini, parameter yang akan dianalisa adalah nilai *F-Measure* dan waktu pelatihan yang dibutuhkan untuk melatih JST. Nilai *F-Measure* didapatkan dari perhitungan menggunakan nilai *recall* dan *precision* sesuai dengan persamaan 2.35. Kemudian nilai *F-Measure* dan waktu pelatihan tersebut dicatat untuk setiap percobaan menggunakan jumlah data latih yang berbeda-beda. Adapun hasil dari proses pengujian ini dapat dilihat pada Tabel 4.3.

Tabel 4.3 Pengujian *F-Measure*Kelas Apel Dan Pistol.

Learning Data	***Recall***	***Precission***	***F-Measure***	***Time***
20	1	0.56	0.717948718	00:01.138
40	1	0.62	0.765432099	00:02.168
60	1	0.67	0.80239521	00:03.213
80	1	0.77	0.870056497	00:04.368
100	1	0.67	0.80239521	00:05.647
120	1	0.91	0.952879581	00:06.380
140	1	1	1	00:07.363
160	1	1	1	00:08.497

Jika dilihat dari nilai parameter *F-Measure*, pada Tabel 4.3 dapat dilihat bahwa nilai *F-Measure* terbesar adalah 1 yang didapatkan pada saat nilai jumlah data latih adalah 140 dan 160. Sedangkan nilai *F-Measure* terkecil adalah 0.717948718 yang didapatkan pada saat nilai jumlah data latih adalah 20. *Nilai F-Measure* ini memiliki kecenderungan naik ketika jumlah data latih dibesarkan, dan begitu juga sebaliknya.

Kemudian jika dilihat dari nilai parameter waktu, pada Tabel 4.3 dapat dilihat bahwa waktu pelatihan terlama adalah 8 detik 497 milidetik yaitu ketika jumlah data latih sebanyak 160 citra, sedangkan waktu pelatihan tersingkat adalah 1 detik 138 milidetik yaitu ketika jumlah data latih sebanyak 20 citra. Waktu pelatihan ini mengalami kenaikkan ketika jumlah data latih diperbesar, dan begitu juga sebaliknya.

Seperti yang sudah dipaparkan pada sistematika uji akurasi sistem, pengujian pengaruh jumlah data latih terhadap nilai *F-Measure* juga dilakukan untuk variasi kelas yang berbeda. Pertama dilakukan pengujian dengan menggunakan data dari kelas citra hasil pengeditan (pengubahan ukurandan penggeseran), dimana hasil dari pengujian ini dapat dilihat pada Tabel 4.4. Dan untuk pengujian variasi kelas yang kedua, dilakukan pengujian menggunakan data dari kelas citra yang mirip, dimana hasil pengujiaannya dapat dilihat pada Tabel 4.5.

Tabel 4.4 Pengujian *F-Measure*Kelas Citra Hasil Pengeditan.

Learning Data	***Recall***	***Precission***	***F-Measure***	***Time***
20	0.5	0.83	0.62406015	00:01.920
40	1	0.5	0.666666667	00:02.184
60	1	0.5	0.666666667	00:03.166
80	1	0.5	0.666666667	00:04.196
100	1	0.5	0.666666667	00:05.210
120	1	0.5	0.666666667	00:06.318
140	1	0.5	0.666666667	00:07.300
160	1	0.5	0.666666667	00:08.330

Tabel 4.5 Pengujian *F-Measure* Kelas Citra Mirip

Learning Data	***Recall***	***Precission***	***F-Measure***	***Time***
20	1	0.5	0.666666667	0:0:1.138
40	1	0.59	0.742138365	00:02.106
60	1	0.59	0.742138365	00:03.135
80	1	0.56	0.717948718	00:04.149
100	1	0.59	0.742138365	00:05.132
120	1	0.59	0.742138365	00:06.146

140	1	0.59	0.742138365	00:07.176
160	1	0.62	0.765432099	00:08.174

Pada pengujian kedua, yaitu pengujian pengaruh jumlah kelas pada data latih terhadap akurasi sitem, sistem diuji menggunakan beberapa parameter seperti yang telah dijelaskan pada sistematika pengujian. Proses pengujian dilakukan dengan menggunakan variasi kelas sebanyak 2, 3, dan 4 variasi kelas.Adapun kelas-kelas yang digunakan untuk pengujian ini antara lain kelas pistol, apel, bolpoin, dan pisau. Contoh citra dari keempat kelas tersebut dapat dilihat pada Gambar 4.24 dan Gambar 4.25.Untuk setiap kelas yang digunakan sebagai data pengujian, kelas tersebut terdiri dari 80 citra.

Adapun hasil dari proses pengujian pengaruh jumlah kelas terhadap tingkat akurasi sistem dapat dilihat pada Tabel 4.6Tabel 4.6.

Tabel 4.6Pengujian Sistem dengan Pengubahan Jumlah Kelas

Jumlah kelas	**Kebenaran**	***Last RMSE***	**Waktu**
2	100.00%	0.000597904	00:08.268
3	86.67%	0.018786285	00:12.667
4	80.00%	0.030650549	00:16.660

Keterangan:
Jumlah kelas 2 → kelas apel dan pistol
Jumlah kelas 3 →kelas apel, pistol, dan bolpoin
Jumlah kelas 4 → kelas apel, pistol, bolpoin, dan pisau

Jika dilihat dari nilai parameter kebenaran, maka Tabel 4.6menunjukkan bahwa nilai kebenaranterbesar adalah 100% yang didapatkan pada saat jumlah variasi kelas yang ada pada data latih adalah 2. Sedangkan nilai kebenaranterkecil adalah 80% yang didapatkan pada saat jumlah variasi kelas adalah 4. Nilai kebenaranini memiliki kecenderungan turun ketika jumlah kelas yang digunakan ditambah, dan begitu juga sebaliknya.

Kemudian jika dilihat dari parameter nilai RMSE terakhir yang didapatkan pada saat proses pelatihan JST, nilai RMSE terakhir yang terkecil didapatkan pada saat variasi kelas berjumlah 3 yaitu sebesar 0.000379237, sedangkan nilai RMSE yang terbesar didapatkan pada saat variasi kelas berjumlah 4 yaitu sebesar

0.076660161. Nilai RMSE terakhir ini memiliki kecenderungan naik ketika jumlah kelas ditambah, tetapi tidak selalu naik ketika jumlah kelas bertambah. Hal ini dapat dilihat dari hasil pengujian yang sudah didapatkan, seperti terlihat pada Tabel 4.6, nilai RMSE terakhir pada saat jumlah variasi kelas adalah 2 ternyata lebih besar daripada nilai RMSE terakhir yang didapatkan ketika jumlah variasi kelas adalah 3.

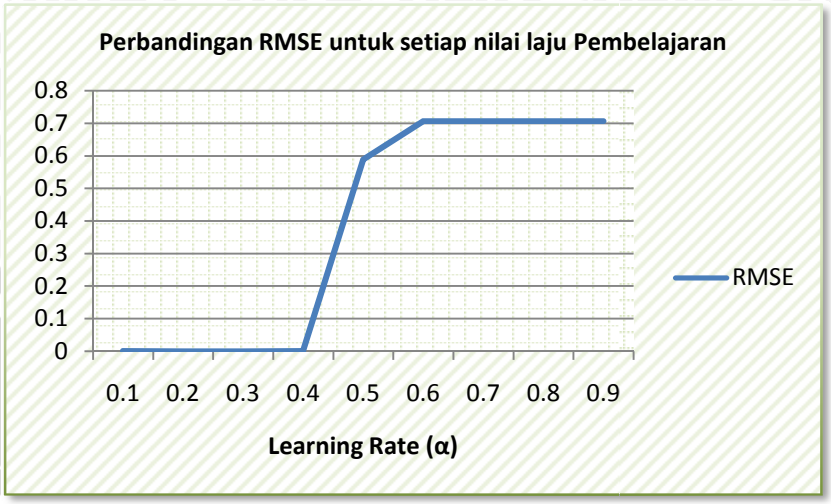
Kemudian jika dilihat dari parameter waktu pelatihan yang digunakan untuk melatih jaringan syaraf tiruan, jumlah kelas yang memerlukan waktu pelatihan JST paling singkat adalah 2 yaitu memerlukan waktu pelatihan selama 8 detik 268 milidetik dan jumlah kelas yang memerlukan waktu pelatihan JST paling lama adalah 4 yaitu memerlukan waktu pelatihan selama 16 detik 660 milidetik. Lama waktu pelatihan ini mengalami kenaikan ketika jumlah kelas pada data latih ditambah, dan begitu juga sebaliknya.

4.6. Analisis Hasil

4.6.1. Analisis Laju Pembelajaran

Dari hasil pengujian yang sudah dilakukan, dapat diketahui bahwa nilai rata-rata nilai RMSE yang didapatkan oleh JST pada *epoch* ke-100 dengan nilai laju pembelajaran yang berbeda-beda memiliki nilai yang bervariasi.

Nilai rata-rata RMSE yang paling kecil didapatkan pada saat nilai laju pembejalaran = 0,2 sedangkan nilai rata-rata RMSE paling besar didapatkan pada saat nilai laju pembelajaran = 0,9. Adapun grafik pengaruh laju pembelajaran terhadap nilai RMSE hasil pelatihan jaringan syaraf tiruan dapat dilihat pada Gambar 4.27.

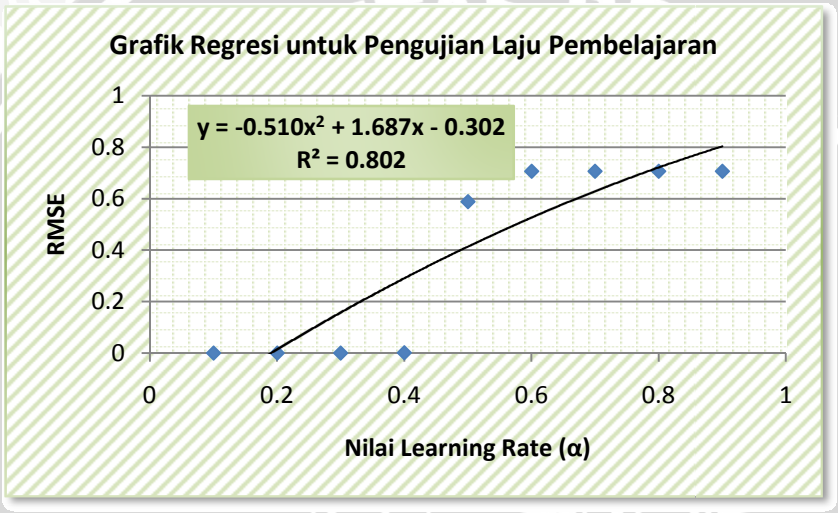


Gambar 4.27 Grafik Pengujian Nilai Laju Pembelajaran

Grafik pada Gambar 4.27 menunjukkan bahwa nilai laju yang kecil akan menghasilkan nilai *error* yang kecil. Tetapi tidak selalu nilai laju pembelajaran yang lebih kecil menghasilkan error yang lebih kecil juga. Hal ini seperti terlihat pada Tabel 4.1, nilai RMSE untuk laju pembelajaran 0.1 ternyata lebih besar dari nilai RMSE untuk laju pembelajaran 0.2. Hal ini bisa terjadi karena nilai RMSE yang didapatkan dari proses pelatihan jaringan syaraf tiruan (JST)sangat tergantung dari bobot awal yang digunakan pada saat proses awal pelatihan. Penentuan bobot awal pada pembentukan JST dilakukan secara *random*. Walaupun terdapat beberapa metode yang dapat digunakan untuk memperbaiki bobot yang didapatkan secara *random*, pembangkitan bobot awal untuk setiap percobaan pelatihan JST tetap akan menghasilkan bobot awal yang berbeda-beda sehingga nilai RMSE yang didapatkan untuk suatu nilai laju pembelajaran tertentu bisa berbeda-beda untuk setiap percobaan yang dilakukan, tergantung dari bobot awal yang didapatkan pada suatu percobaan.

Dari grafik pada Gambar 4.27dapat dilakukan analisis lebih lanjut untuk mengetahui kecenderungan pergerakan grafik pengaruh nilai laju pembelajaran terhadap nilai RMSE. Adapun analisis yang dapat dilakukan adalah analisis regresi. Pada Tabel 4.1 dapat

diketahui bahwa perubahan nilai RMSE diikuti dengan perubahan yang tidak tetap pada nilai *learning rate*, sehingga pendekatan analisis regresi yang digunakan adalah regresi non-linear yaitu menggunakan regresi *polynomial order* 2 (*quadratic*). Analisis ini digunakan untuk mengetahui kecenderungan pergerakan grafik.Grafik analisis regresi untuk pengujian pengaruh nilai laju pembelajaran terhadap nilai RMSE dapat dilihat pada Gambar 4.28.



Gambar 4.28 Grafik Regresi Pengujian Laju Pembelajaran

Grafik pada Gambar 4.28 menunjukkan bahwa nilai RMSE cenderung mengalami kenaikan ketika nilai laju pembelajaran dinaikkan. Persamaan regresi yang didapatkan dari grafik tersebut adalah $y= -0.51x^2 + 1.6877x - 0.3022$. Dari grafik regresi juga diketahui bahwa nilai R^2 adalah 0.802, hal ini menunjukkan bahwa sebesar 80.2% nilai x berkontribusi dalam mempengaruhi nilai y dan sisa19.8% lainnya, nilai y dipengaruhi oleh variabel lain yang tidak dibahas dalam penelitian ini.

Dari percobaan pengujian laju pembelajaran yang telah dilakukan, nilai RMSE yang didapatkan mengalami kenaikan yang drastis ketika nilai laju pembelajaran bernilai 0.4. Pada saat nilai laju pembelajaran 0.1 – 0.4, nilai RMSE yang didapatkan terletak pada rentang 0 – 0.01. Tetapi pada saat nilai laju pembelajaran 0.5 – 0.9,

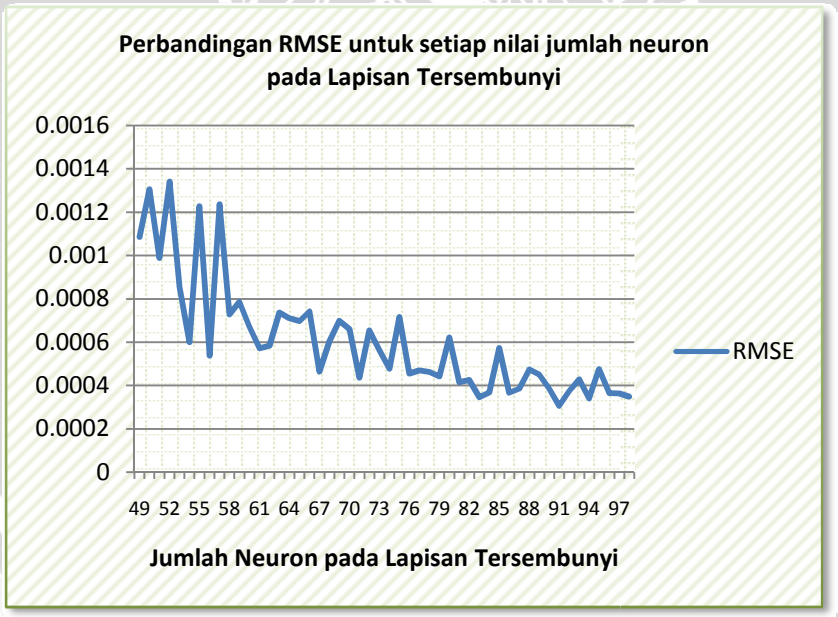
nilai RMSE yang didapatkan terletak pada rentang 0 – 1. Sehingga untuk memaksimalkan hasil dari proses pelatihan jaringan syaraf tiruan yang akan digunakan untuk melakukan proses pelabelan citra, maka laju pembelajaran yang digunakan pada saat melakukan pelatihan JST terletak pada rentang0.1 – 0.4.

4.6.2. Analisis Jumlah Neuron Lapisan Tersembunyi

Dari hasil pengujian jumlah *neuron* pada lapisan tersembunyi yang telah dilakukan, dapat diketahui bahwa nilai rata-rata RMSE dari hasil percobaan memiliki nilai yang bervariasi.

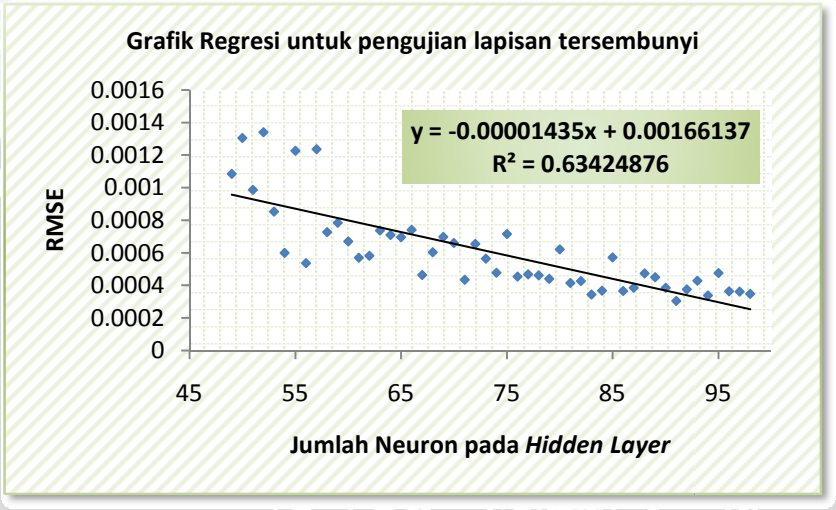
Nilai rata-rata RMSE terkeciladalah 0.000305976 yang didapatkan pada saat nilai jumlah neuron sebesar 91, sedangkan nilai rata-rata RMSE terbesar adalah 0.00134 yang didapatkan pada saat nilai jumlah neuron sebesar 52.

Adapun grafik yang menunjukkan pengaruh jumlah neuron pada lapisan tersembunyi terhadap nilai RMSE hasil pelatihan jaringan syaraf tiruan dapat dilihat pada Gambar 4.29.



Gambar 4.29 Grafik Pengujian Jumlah Lapisan Tersembunyi

Dari grafik pada Gambar 4.29dapat dilakukan analisa regresi untuk mengetahui kecenderungan pergerakan grafik pengaruh jumlah *neuron* terhadap nilai RMSEyang dihasilkan. Setelah dilakukan perhitungan dengan menggunakan analisis regresi maka didapatkan grafik seperti pada Gambar 4.30.



Gambar 4.30 Grafik Regresi Pengujian Lapisan Tersembunyi

Dari grafik regresi seperti yang terlihat pada Gambar 4.30, terlihat bahwa nilai nilai RMSE yang didapatkan untuk setiap jumlah neuron pada *hidden layer* mengalami kecenderungan turun. Dari persamaan regresi yang didapatkan, yaitu *y = -0.00001435x + 0.00166137*, dapat disimpulkan bahwa setiap kenaikan nilai jumlah neuron pada lapisan tersembunyisebesar satu satuan akan menurunkan nilai RMSE sebesar 0.00001435. Sehingga dapat disimpulkan bahwa semakin banyak jumlah neuron pada lapisan tersembunyi, maka nilai RMSE yang dihasilkan oleh jaringan syaraf tiruan cenderung mengalami penurunan. Dari grafik regresi juga diketahui bahwa nilai R^2 adalah 0.63, hal ini menunjukkan bahwa sebesar 63% nilai *x* berkontribusi dalam mempengaruhi nilai *y* dan sisa 37% lainnya nilai *y* dipengaruhi oleh variabel lain yang tidak dibahas dalam penelitian ini.

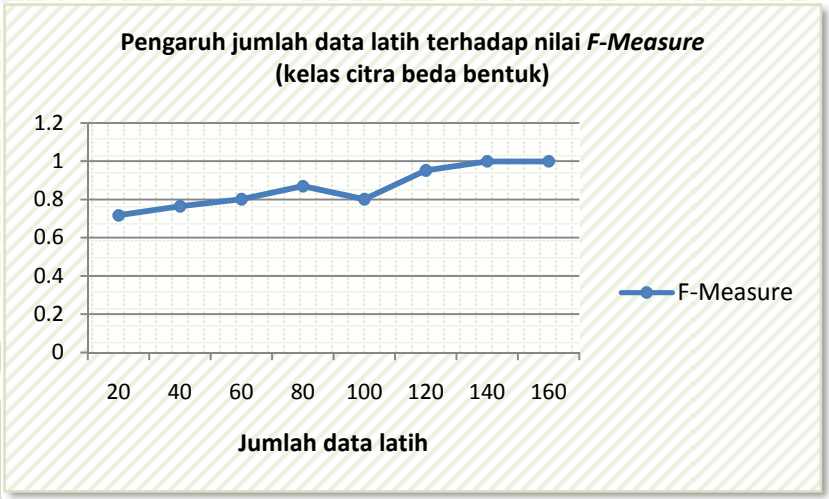
Dari pengujian jumlah *neuron* pada lapisan tersembunyi yang telah dilakukan, maka dapat diketahui bahwa nilai jumlah *neuron* yang digunakan untuk membentuk jaringan syaraf tiruan (JST) memiliki pengaruh terhadap nilai RMSE yang didapatkan oleh sebuah JST tersebut. Semakin banyak jumlah *neuron* pada lapisan tersembunyi, maka nilai RMSE yang dihasilkan oleh JST tersebut akan menjadi semakin kecil, dan begitu juga sebaliknya.

4.6.3. Analisis Akurasi Sistem Anotasi Citra

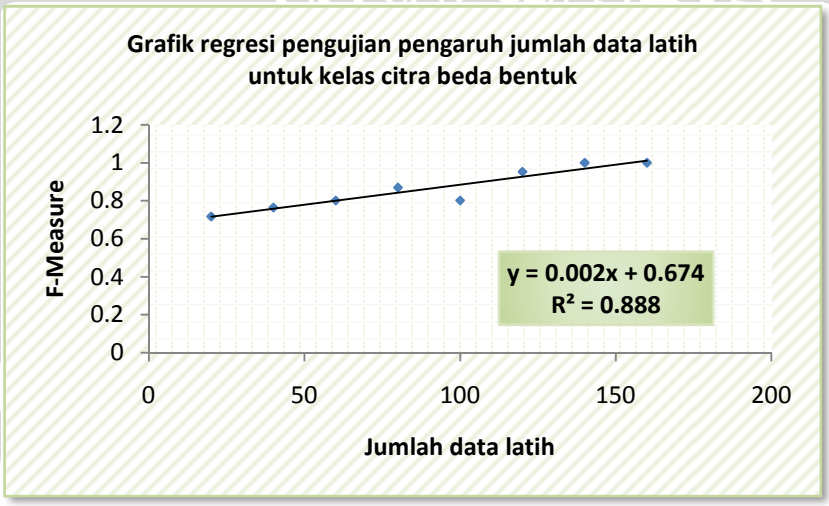
Seperti yang telah dijabarkan pada sistematika pengujian akurasi sistem, terdapat dua pengujian yang akan dilakukan yaitu pengujian pengaruh jumlah data latih terhadap tingkat akurasi sistem dan pengujian pengaruh jumlah kelas pada data latih terhadap tingkat akurasi sistem.

Pada pengujian pengaruh jumlah data latih terhadap tingkat akurasi sistem, dilakukan pengujian terhadap tiga variasi kelas yang berbeda yaitu variasi kelas citra beda bentuk, variasi kelas citra hasil pengeditan, dan variasi kelas citra mirip. Pada pengujian ini terdapat dua parameter yang akan dianalisa yaitu parameter *F-Measure* dan parameter waktu pelatihan.

Dari hasil pengujian tingkat akurasi kelas beda bentuk (kelas apel dan pistol), dapat diketahui bahwa nilai *F-Measure* memiliki nilai yang bervariasi.Adapun grafik yang menunjukkan pengaruh jumlah data latih terhadap nilai *F-Measure*untuk kelas beda bentuk dapat dilihat pada Gambar 4.31.Dari grafik yang ada padaGambar 4.31 tersebut, dapat dilakukan analisis regresi untuk mengetahui kecenderungan pergerakan grafik. Adapun grafik regresi yang digunakan untuk mengetahui kecenderungan pergerakan grafik pengaruh jumlah data latih terhadap nilai *F-Measure* dapat dilihat pada Gambar 4.32.



Gambar 4.31 Grafik Perubahan*F-Measure*Citra Beda Bentuk

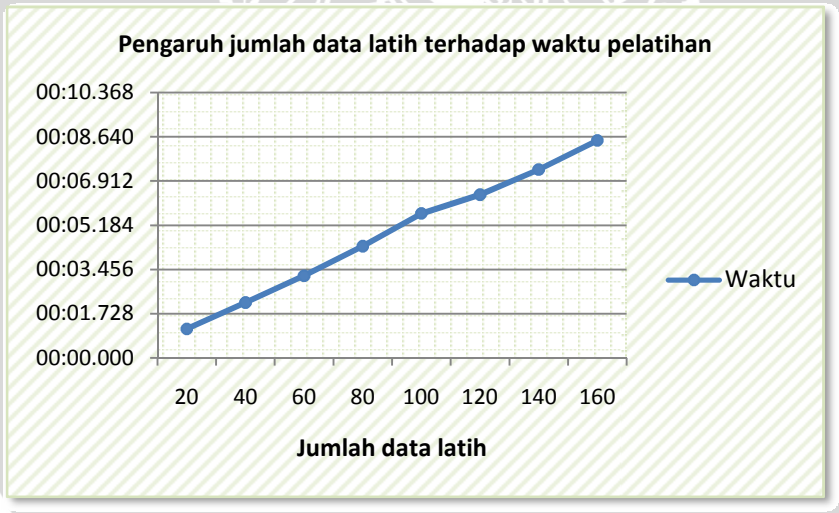


Gambar 4.32Grafik Regresi *F-Measure*Citra Beda Bentuk

Dari grafik pada Gambar 4.31dapat diketahui bahwa nilai *F-Measure* yang paling kecil didapatkan ketika jumlah data latih yang digunakan pada pelatihan JST sebanyak 20, sedangkan nilai *F-Measure* yang paling besar didapatkan ketika jumlah data latih

sebanyak 140 dan 160. Dari grafik regresi seperti yang terlihat pada Gambar 4.32, diketahui bahwa persamaan regresinya adalah *y = 0.0021x + 0.6747*, artinya adalah setiap kenaikan nilai *x* sebesar satu satuan, maka akan menaikan nilai *y* sebesar 0.0021 sehingga dapat disimpulkan bahwa semakin banyak data latih yang digunakan untuk pelatihan JST, maka nilai *F-measure* yang didapatkan oleh sistem cenderung mengalami kenaikan. Dari grafik regresi juga diketahui bahwa nilai R^2adalah 0.88. Hal ini menunjukkan bahwa sebanyak 88% nilai *x* mempengaruhi nilai *y*, dan sisa 12% lainnya, nilai *y* dipengaruhi oleh variabel lain yang tidak dibahas dalam penelitian ini.

Pengujian selanjutnya adalah pengujian berdasarkan parameter waktu pelatihan. Dari hasil pengujian, dapat diketahui bahwa waktu pelatihan mengalami kenaikan ketika jumlah data latih bertambah.Adapun grafik yang menunjukkan pengaruh jumlah data latih terhadap nilai waktu pelatihan JST dapat dilihat padaGambar 4.33.

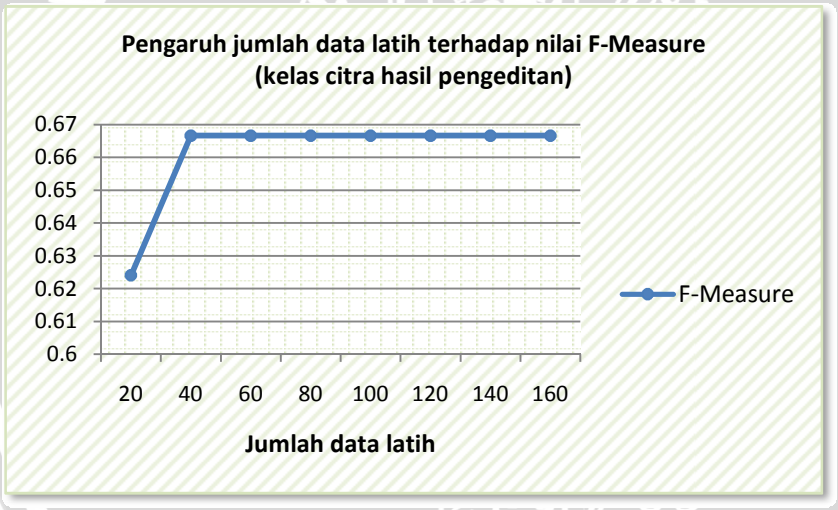


Gambar 4.33 Grafik Pengaruh Data Latih terhadap Waktu

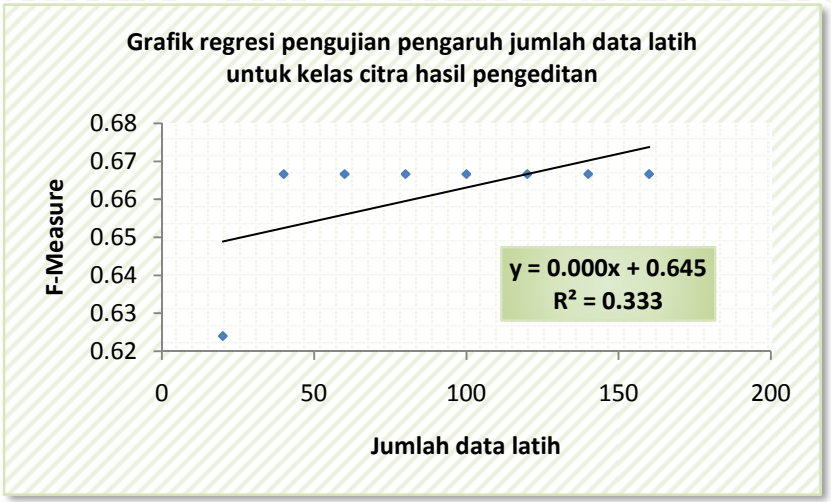
Seperti yang terlihat dari grafik pada Gambar 4.33, jumlah data latih yang digunakan untuk melatih JST sangat berpengaruh terhadap waktu yang digunakan untuk melatih jaringan syaraf

tersebut. Hal ini dapat dilihat bahwa semakin banyak data yang digunakan untuk melatih JST, maka waktu yang diperlukan untuk melakukan pelatihan JST menjadi semakin lama, dan begitu juga sebaliknya.

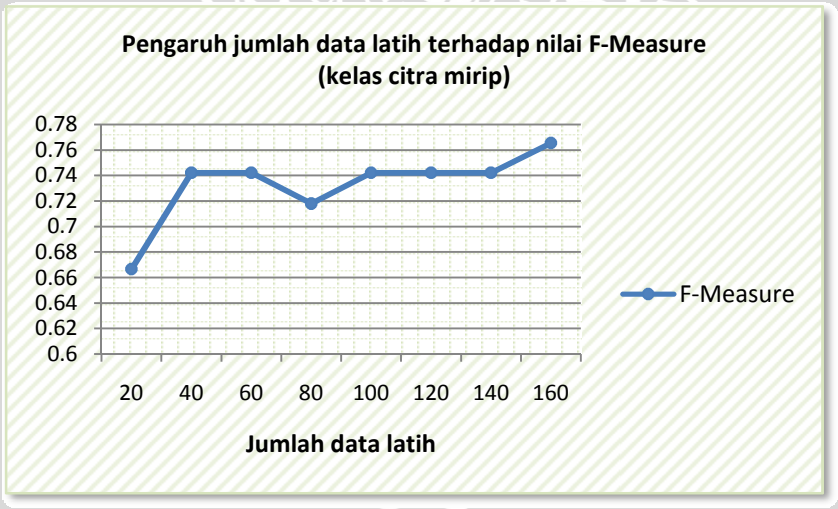
Pengujian pengaruh jumlah data latih terhadapa nilai *F-measure* selanjutnya adalah pengujian dengan menggunakan data latih dari kelas citra hasil pengeditan (pengubahan ukuran dan penggeseran citra) dan data latih dari kelas citra mirip. Untuk pengujian dengan menggunakan masing-masing kelas tersebut, pelatihan juga dilakukan dengan mengubah jumlah citra yang digunakan sebagai data latih. Adapun grafik yang menunjukkan pengaruh pengubahan jumlah kelas terhadap nilai *F-Measure* untuk citra hasil pengeditan dan citra mirip dapat dilihat pada Gambar 4.34 dan Gambar 4.36, sedangkan analisis regresi dari kedua grafik tersebut dapat dilihat pada Gambar 4.35 danGambar 4.37.



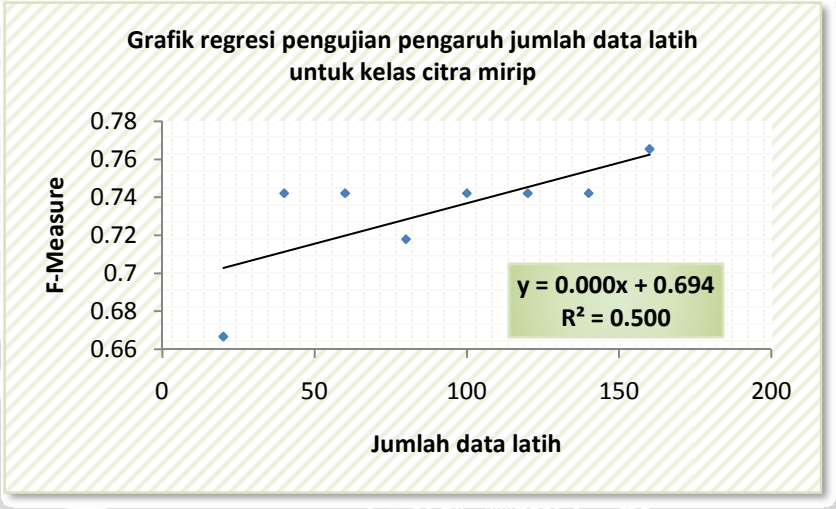
Gambar 4.34Grafik *F-Measure*Citra Hasil Pengeditan



Gambar 4.35 Grafik Regresi *F-Measure*Citra Hasil Pengeditan



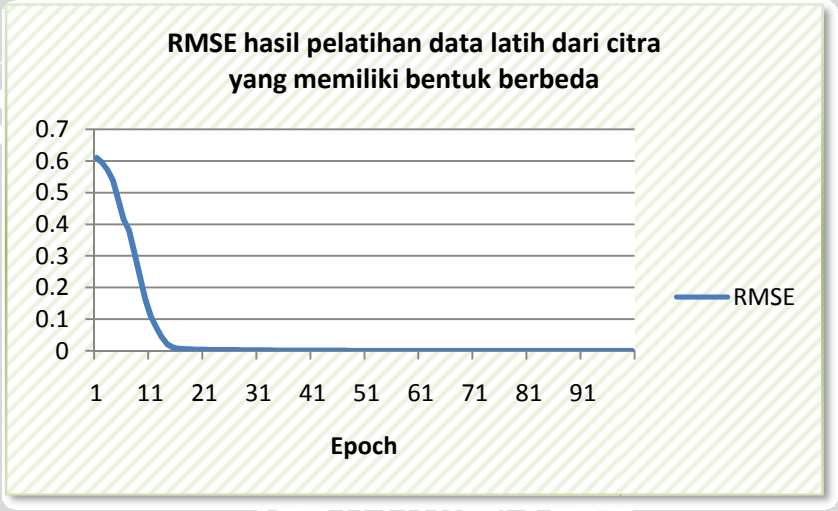
Gambar 4.36 Grafik Perubahan *F-Measure*Citra Mirip



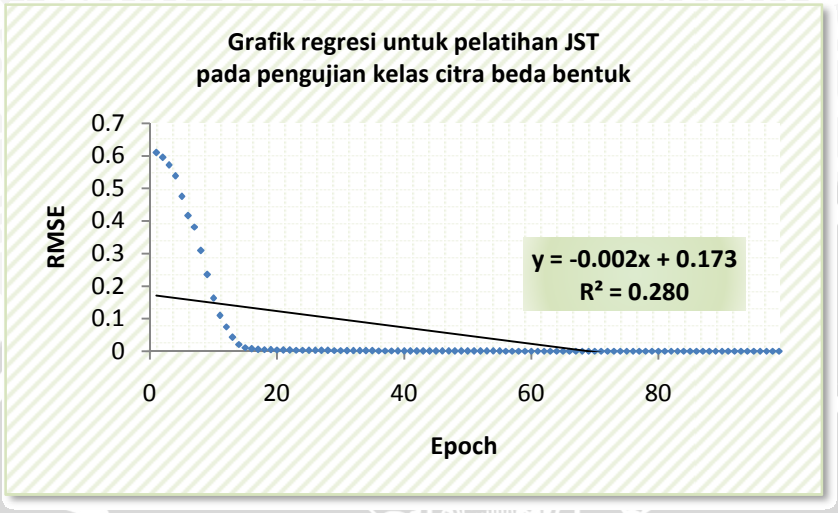
Gambar 4.37Grafik Regresi *F-Measure* Citra Mirip

Dari grafik analisis regresi pada Gambar 4.35 dan Gambar 4.37, dapat disimpulkan bahwa semakin banyak data latih yang digunakan untuk proses pelatihan jaringan syaraf tiruan (JST), maka nilai *F-Measure* untuk menguji tingkat akurasi sistem cenderung mengalami kenaikan. Dari kedua grafik regresi tersebut didapatkan persamaan regresi y = 0.0002x + 0.6454 dengan R^2 = 0.3333 untuk variasi kelas citra hasil pengeditan, dan persamaan regresi y = 0.0004x + 0.6943 dengan R^2 = 0.5001 untuk variasi kelas citra mirip. Dari kedua persamaan tersebut dapat diketahui bahwa setiap kenaikan nilai *x* pada grafik variasi kelas citra hasil pengeditan, maka nilai *y* akan naik sebesar 0.0002 dan setiap kenaikan nilai *x* pada grafik variasi kelas citra mirip, maka nilai *y* akan naik sebesar 0.0004. Kedua nilai ini relatif kecil jika dibandingkan dengan hasil dari analisis regresi untuk variasi kelas beda bentuk. Pada hasil analisis regresi untuk variasi kelas beda bentuk, setiap kenaikan *x* sebesar satu satuan, maka nilai *y* akan naik sebesar 0.0021. Hal ini menunjukkan bahwa variasi data latih sangat mempengaruhi tingkat akurasi dari sistem. Kelas data latih yang menghasilkan tingkat akurasi paling baik adalah kelas citra yang memiliki bentuk yang berbeda. Sedangkan kedua variasi kelas yang lainnya, yaitu kelas citra hasil pengeditan dan citra mirip, menghasilkan tingkat akurasi

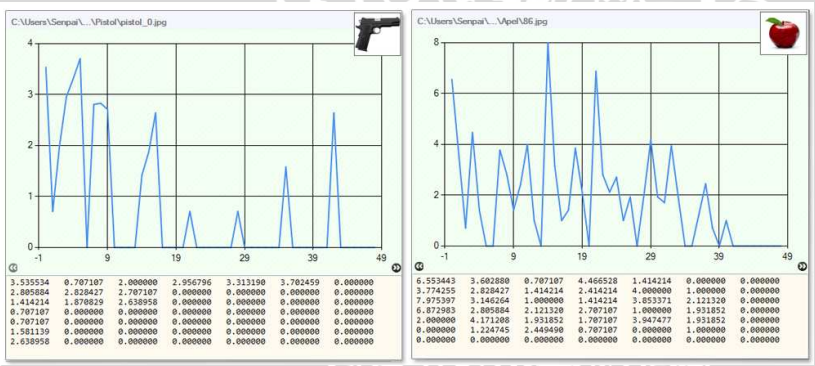
yang kurang baik. Hal ini dikarenakan kelas yang terdiri dari citra-citra beda bentuk, dimana suatu citra tertentu memiliki perbedaan dengan citra lain yang berbeda kelas tetapi memiliki kemiripan dengan citra lain yang sama kelas, menghasilkan nilai RMSE yang kecil dengan waktu yang relatif cepat seperti terlihat pada grafik nilai RMSE padaGambar 4.38. Adapun perbedaan fitur dari citra yang terdapat pada variasi kelas beda bentuk ini dapat dilihat pada Gambar 4.39. Dari perbandingan tersebut, terlihat bahwa citra pada suatu kelas tertentu memiliki perbedaan yang jelas dengan citra pada kelas lain jika dilihat dari nilai fitur yang didapatkan dari proses ekstraksi fitur dari masing-masing citra tersebut.



Gambar 4.38Pelatihan JST Kelas Citra Beda Bentuk



Gambar 4.39Grafik Regresi Pengujian Kelas Citra Beda Bentuk



Gambar 4.40Perbandingan Fitur dari 2 Citra yang Berbeda

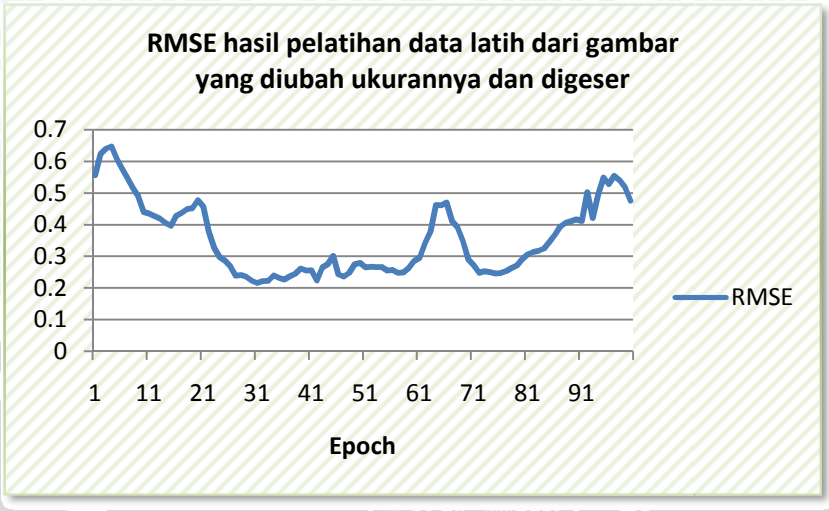
Dari grafik analisis regresi pada Gambar 4.39 dapat disimpulkan bahwa semakin banyak *epoch*, maka nilai RMSE yang dihasilkan mengalami penurunan sebesar 0.0025. Hal ini menunjukkan bahwa data latih yang digunakan berhasil mendapatkan nilai *error* yang relatif baik dengan kecenderungan turun untuk setiap *epoch*.

Adapun untuk pengujian nilai *F-Measure* menggunakan variasi kelas citra hasil pengeditan seperti terlihat pada Gambar 4.34,

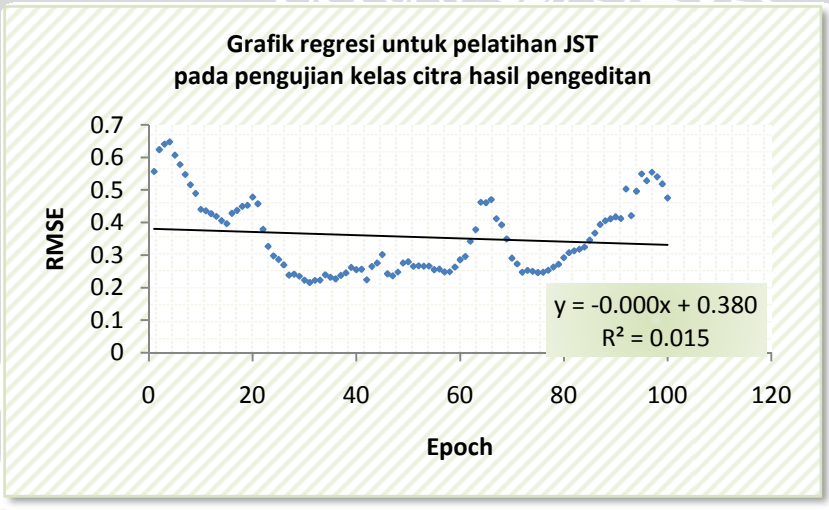
dapat diketahui bahwa tingkat akurasi yang didapatkan sistem adalah relatif sama untuk semua jumlah data latih yang digunakan sebagai parameter pengujian. Hal ini menunjukkan bahwa posisi dari suatu obyek sangat mempengaruhi tingkat akurasi dari sistem ini. Tingkat akurasi yang kurang baik ini dipengaruhi oleh data latih yang digunakan untuk melatih JST. Adapun grafik yang menunjukkan pergerakan nilai RMSE pada saat dilakukan pelatihan JST dari data citra hasil pengeditan ini dapat dilihat pada Gambar 4.41 dan analisis regresi dari pergerakan nilai RMSE ini dapat dilihat pada Gambar 4.42.

Grafik pada Gambar 4.41 menunjukkan bahwa nilai RMSE mengalami fluktuasi dan RMSE yang dihasilkan adalah besar yaitu terletak disekitar nilai 0.5, padahal nilai RMSEyang didapatkan dari variasi kelas beda bentuk adalah terletak di sekitar nilai 0.0002. Sehingga hal ini menyebabkan tingkat akurasi yang didapatkan dari variasi kelas citra hasil pengeditan ini menjadi kurang baik.

Adapun perbedaan nilai fitur citra dari variasi kelas citra hasil pengeditan dapat dilihat padaGambar 4.43.Dari perbandingan tersebut, terlihat bahwa suatu citra pada suatu kelas tertentu memiliki perbedaan dengan citra pada kelas itu sendiri, dan juga memiliki perbedaan dengan citra pada kelas lain. Sehingga hal itulah yang menyebabkan proses pelatihan JST menghasilkan nilai RMSE yang tinggi.



Gambar 4.41Pelatihan JST Kelas Citra Hasil Pengeditan



Gambar 4.42Grafik Regresi Pengujian Citra Hasil Pengeditan

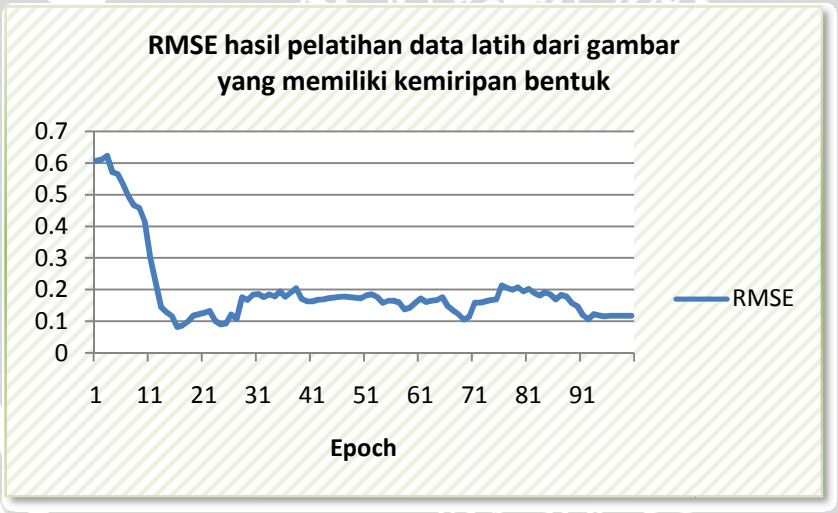
Gambar 4.43Perbandingan Fitur dari Citra Hasil Pengeditan

Dari grafik analisis regresi pada Gambar 4.42dapat disimpulkan bahwa semakin banyak epoch, maka nilai RMSE yang dihasilkan mengalami penurunan sebesar 0.0005. Penurunan nilai RMSE ini lebih kecil dari nilai penurunan RMSE pada variasi kelas beda bentuk, sehingga tingkat akurasi yang dihasilkan oleh variasi kelas ini relatif kurang baik.

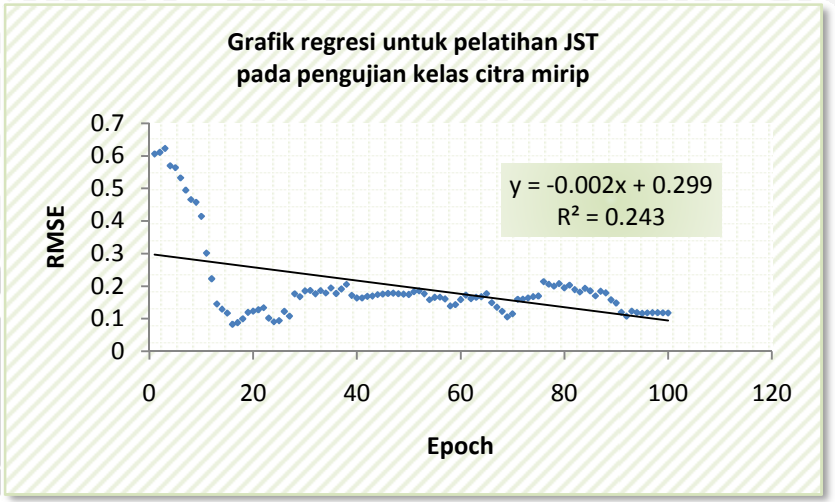
Pada pengujian nilai *F-Measure* menggunakan variasi kelas citra yang mirip seperti terlihat padaGambar 4.36, dapat diketahui bahwa tingkat akurasi yang didapatkan sistem adalah bervariasi untuk setiap jumlah data latih.

Tingkat akurasi yang didapatkan pada pengujian menggunakan variasi kelas citra yang mirip ini masih dibawah dari tingkat akurasi yang didapatkan pada pengujian menggunakan variasi kelas citra yang berbeda. Hal ini dikarenakan pada variasi kelas ini, citra pada suatu kelas memiliki sebaran nilai fitur yang

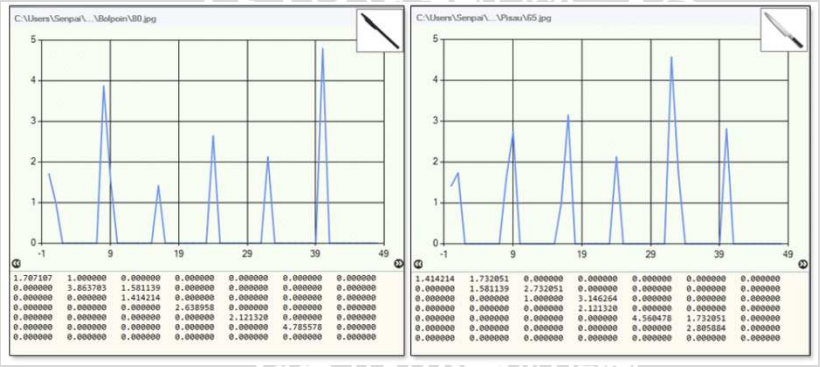
hampir sama dengan citra yang berada pada kelas lain. Seperti terlihat pada grafik sebaran fitur citra pada Gambar 4.45, terdapat dua citra dari dua kelas yang berbeda, yaitu citra bolpoin dan citra pisau, memiliki sebaran fitur yang hampir sama. Artinya bahwa suatu citra pada suatu kelas tertentu, selain memiliki kemiripan dengan citra lain pada kelas tersebut, citra tersebut juga memiliki kemiripan dengan citra lain pada kelas lain. Hal inilah yang menyebabkan proses pelatihan JST menghasilkan nilai RMSE yang relatif besar yaitu di sekitar nilai 0.1 jika dibandingkan dengan nilai RMSE yang dihasilkan oleh variasi kelas beda bentuk yang terletak di sekitar nilai 0.0002. Adapun grafik yang menunjukkan pergerakan nilai RMSE untuk setiap epoch pada pelatihan JST menggunakan data latih dari variasi kelas citra mirip dapat dilihat pada Gambar 4.44 dan untuk analisis regresi dari grafik tersebut dapat dilihat pada Gambar 4.45.



Gambar 4.44 Pelatihan JST dari Kelas Citra Mirip



Gambar 4.45Grafik Regresi Pengujian Kelas Citra Mirip



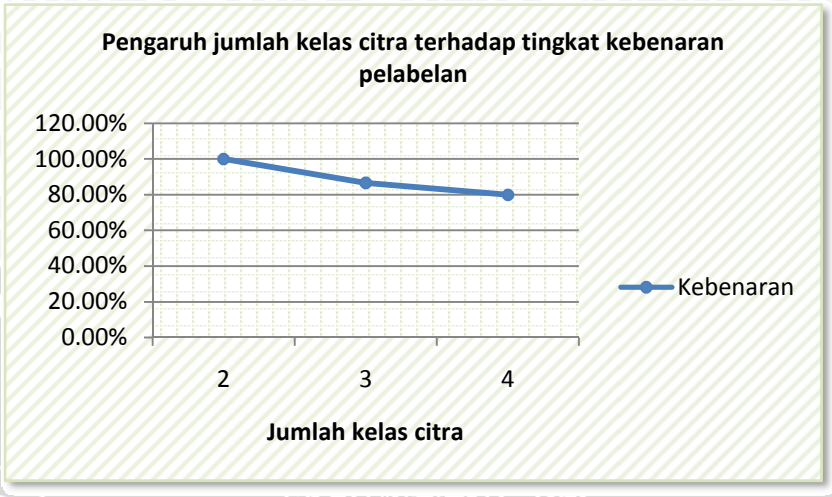
Gambar 4.46Perbandingan Fitur dari 2 Citra yang Mirip

Dari grafik analisis regresi seperti yang terlihat pada Gambar 4.45, dapat disimpulkan bahwa semakin banyak *epoch*pada proses pelatihan jaringan syaraf tiruan maka nilai RMSE yang dihasilkan mengalami penurunan sebesar 0.0005. Penurunan nilai RMSE ini lebih kecil dari nilai penurunan RMSE pada variasi kelas beda bentuk, sehingga tingkat akurasi yang dihasilkan oleh variasi kelas ini tidak lebih baik dari tingkat akurasi yang dihasilkan oleh variasi kelas beda bentuk.

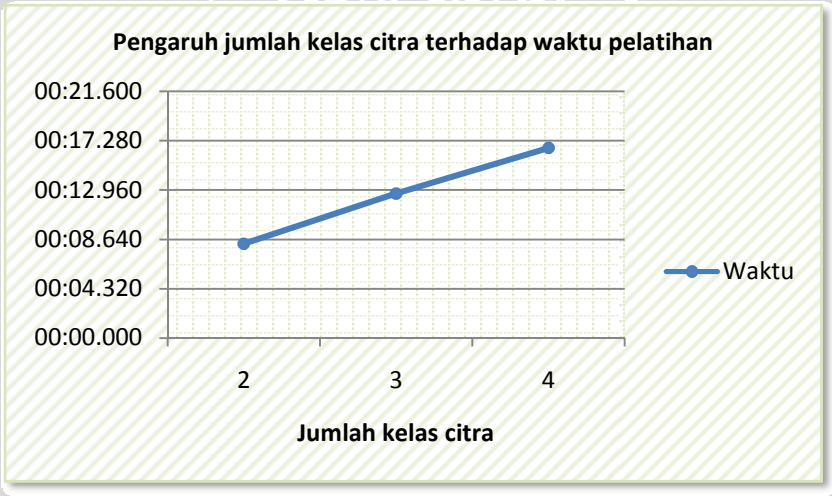
Dari hasil semua pengamatan pada proses pengujian pertama, yaitu pengamatan perubahan nilai *F-Measure* dan perubahan lama waktu pelatihan, dapat disimpulkan bahwa semakin banyak data latih yang digunakan maka proses pelatihan JST menghasilkan hasil yang relatif lebih baik, sehingga mampu meningkatkan tingkat akurasi dari sistem anotasi citra, tetapi membutuhkan waktu yang relatif lebih lama untuk melatih semua data tersebut.

Nilai *F-Measure*sangat dipengaruhi oleh variasi kelas. Jika data latih memiliki banyak perbedaan antar kelas dan banyak persamaan pada kelas yang sama, maka tingkat akurasi yang dihasilkan sistem relatif tinggi. Jika data latih memiliki banyak perbedaan antar kelas dan pada kelas yang sama, maka tingkat akurasi yang dihasilkan menjadi kurang baik. Jika data latih memiliki banyak persamaan antar kelas dan pada kelas yang sama, maka tingkat akurasi yang dihasilkan juga kurang baik.

Adapun untuk proses pengujian yang kedua, yaitu pengujian pengaruh jumlah kelas pada data latih terhadap akurasi sitem, akan dianalisa pengaruh dari perubahan jumlah kelas citra yang digunakan terhadap tingkat kebenaran sistem, waktu pelatihan, dan nilai RMSE terakhir yang didapatkan pada saat pelatihan. Adapun grafik yang menunjukkan hasil pengujian yang telah dilakukan pada implementasi uji coba dapat dilihat pada Gambar 4.47, Gambar 4.48, dan Gambar 4.49.



Gambar 4.47 Grafik Pengaruh Kelas terhadapNilai Kebenaran



Gambar 4.48 Grafik Pengaruh Kelas Citra terhadap Waktu

Gambar 4.49Grafik Pengaruh Kelas terhadap Nilai RMSE

Dari ketiga grafik yang terdapat pada Gambar 4.47, Gambar 4.48, dan Gambar 4.49, dapat dilihat bahwa nilai tingkat kebenaran yang didapatkan akan mengalami penurunan ketika jumlah kelas pada data latih dan data uji ditambah, dan semakin banyak jumlah kelas yang digunakan pada data latih dan data uji, maka waktu pelatihan akan semakin meningkat, serta semakin sedikit jumlah kelas yang digunakan pada data latih dan data uji, maka waktu yang dibutuhkan untuk melakukan pelatihan JST akan semakin singkat.


UNIVERSITAS BRAWIJAYA

BAB V
KESIMPULAN DAN SARAN

5.1. Kesimpulan

Setelah dilakukan penelitian tentang implementasi *content-based image retrieval* menggunakan jaringan syaraf tiruan *optical backpropagation*untuk sistem anotasi citra otomatis, dapat disimpulkan beberapa hal, antara lain:

1. Metode pelatihan Jaringan Syaraf Tiruan (JST)*optical backpropagation* dapat digunakan dengan baik untuk melakukan pelatihan JST karena menghasilkan nilai *Root Mean Square Error* (RMSE) yang cenderung turun untuk setiap *epoch*. Hal ini menjadikan metode *content-based image retrieval* yang digunakan untuk membuat sistem anotasi citra otomatis menghasilkan hasil pelabelan citra yang baik.
2. Tingkat akurasi dari hasil pengenalan citra menggunakan sistem anotasi citra sangat dipengaruhi oleh data latih yang digunakan untuk melakukan pelatihan jaringan syaraf tiruan.Dengan menggunakan data latih dari variasi kelas beda bentuk (yang terdiri dari data latih sebanyak 160 citra, dan data uji sebanyak 20 citra), tingkat akurasi yang didapatkan adalah 100%. Tingkat akurasi ini diperoleh dengan menggunakan nilai laju pembelajaran yang paling optimal yang didapatkan dari proses evaluasi sistem yaitu laju pembelajaran 0.2 dan menggunakan konfigurasi jumlah *hidden layer* sebanyak 49 *neuron*. Dengan menggunakan konfigurasi tersebut, nilai RMSE yang didapatkan dari hasil pelatihan JST menunjukkan kecenderungan turun untuk setiap *epoch*. Pada sebuah arsitektur jaringan syaraf tiruan, nilai RMSE yang dihasilkan oleh jaringan syaraf tiruan dipengaruhi oleh nilai laju pembelajaran dan jumlah *neuron* pada lapisan tersembunyi. Semakin besar nilai laju pembelajaran yang digunakan pada sebuah jaringan syaraf tiruan maka nilai RMSE yang dihasilkan cenderung mengalami kenaikan dan begitu juga sebaliknya. Semakin banyak jumlah *neuron* yang digunakan untuk melatih jaringan syaraf tiruan maka nilai RMSE yang dihasilkan cenderung mengalami penurunan dan begitu juga sebaliknya.

5.2. Saran

Beberapa saran untuk pengembangan lebih lanjut dari penelitian ini antara lain:

1. Dapat melakukan penelitian lebih lanjut terhadap teknik transformasi *wavelet* yang lain, seperti menggunakan *daubechies wavelet transformation.*
2. Dapat melakukan pengenalan citra lebih lanjut, dimana pada citra tersebut terdapat 2 obyek atau lebih.

DAFTAR PUSTAKA

Barrios, J.M., Diaz-Espinoza, D. & Bustos, B., 2009. Text-based and content-based image retrieval on Flickr: DEMO. In *Second International Workshop on Similarity Search and Applications*., 2009. IEEE Computer Society.

Chen, Z., Fu, H., Chi, Z. & Feng, D., 2010. A neural network model with adaptive structure for image annotation. In *11th International conference control, automation, robotics and vision*. Singapore, 2010. IEEE.

Chu, H., 2003. *Information Representation and Retrieval in the Digital Age*. New Jersey: Information Today, Inc.

D.A., G. & A.P., G., 2007. *Digital Computer Fundamentals*. Technical Publications.

Eakins, J. & Graham, M., 1999. *Content-based Image Retrieval: A report to the JISC Technology Applications Programme*. Newcastle: University of Northumbria.

Efford, N., 2000. *Digital Image Processing: A Practical Introduction Using Java*. Essex: Pearson Education Limited.

Fausset, L., 1994. *Fundamentals of neural networks*. Unknown: Prentice Hall.

Gonzales, R.C. & Woods, R.E., 2002. *Digital Image Processing*. 2nd ed. New Jersey: Prentice Hall.

Latha, Y.M., Jinaga, B.C. & Reddy, V.S.K., 2007. Content based color image retrieval via wavelet transforms. *International Journal of Computer Science and Network Security*, 7.

Lippmann, R.P., 1987. An Intoduction to computing with neural nets. *IEEE ASSP Magazine*, 3(4), pp.4-22.

Luo, B., Wang, X. & Tang, X., 2003. A World Wide Web Based Image Search Engine Using Text and Image Content Features. *Proceedings of SPIE Internet Imaging IV*, 5018(13), pp.123-30.

Munir, R., 2004. *Pengolahan citra digital dengan pendekatan algoritmik*. Bandung: Informatika.

Nguyen, D. & Widrow, B., 1990. Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights. In *International Joint Conference on Neural Networks*. San Diego, 1990. IEEE.

Olson, D.L. & Delen, D., 2008. *Advanced Data Mining Techniques.* Heidelberg: Springer-Verlag.

Otair, M.A. & Salameh, W.A., 2005. Speeding Up Back-Propagation Neural Networks. In *2005 Informing Science and IT Education Joint Conference*. Arizona, 2005. Informing Science.

Park, S.B., Lee, J.W. & Kim, S.K., 2004. Content-based image classification using a neural network. *Pattern Recognition Letters*, 25(3), pp.287-300.

Ramadijanti, N., 2006. Content Based Image Retrieval Berdasarkan Ciri Tekstur Menggunakan Wavelet. In *Prosiding Seminar Nasional Aplikasi Teknologi Informasi (SNATI) 2006*. Yogyakarta, 2006. Universitas Islam Indonesia.

Santini, S. & Jain, R.C., 1997. The Graphical Specification of Similarity Queries. *Journal of Visual Languages and Computing*, 7(4), pp.403-21.

Soto, P.J. & Derado, D.J., 2004. *Wavelets and Image Processing*. [Online] Diakses dari: http://ksuweb.kennesaw.edu/~jderado/hurricane.htm [Diakses pada 18 Mei 2012].

Stollnitz, E.J., DeRose, T.D. & Salesin, D.H., 1995. Wavelets for Computer Graphics: A Primer, part 1. *IEEE Computer Graphics and Applications*, 15(3), pp.76-84.

Terras, M.M., 2008. *Digital Images for the Information Professional*. Hampshire: Ashgate Publishing Limited.

Tratz, C., 2001. *Extracting IPTC header information from JPEG images*. [Online] Diakses dari: http://www.codeproject.com/Articles/1208/Extracting-IPTC-header-information-from-JPEG-image [Diakses pada 21 April 2012].

Tsai, D. dkk., 2011. Large-scale image annotation using visual synset. In *International Conference on Computer Vision*. Barcelona, 2011. IEEE Computer Society.

Yang, C.C., 2004. Content-based image retrieval: a comparison between query by example and image browsing map approaches. *Journal of Information Science*, 30(3), pp.254-67.

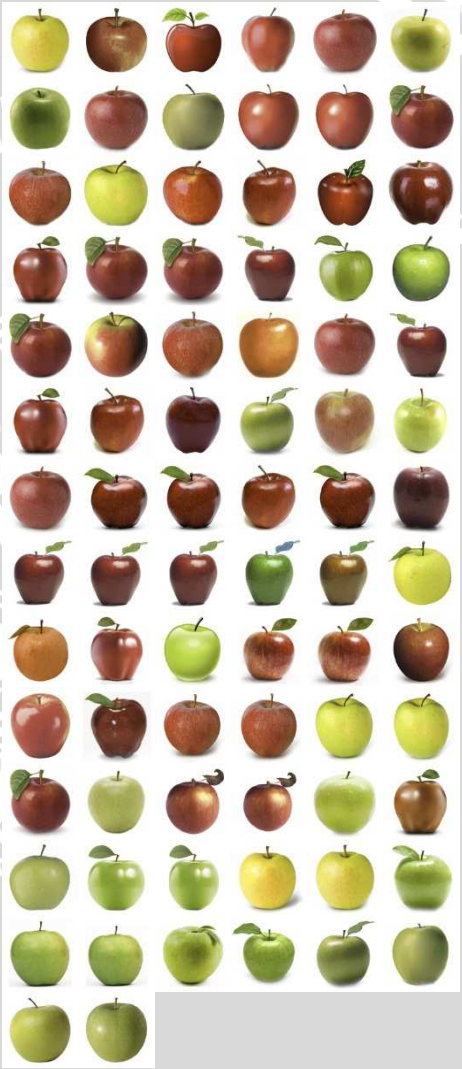
Yavlinsky, A., 2007. *Image indexing and retrieval using automated annotation*. PhD Thesis. London: University of London.

Yavlinsky, A., Schofield, E. & Ruger, S., 2005. Automated Image Annotation Using Global Features and Robust Nonparametric Density Estimation. In *Image and Video Retrieval*. Berlin: Springer. pp.507-17.


UNIVERSITAS BRAWIJAYA

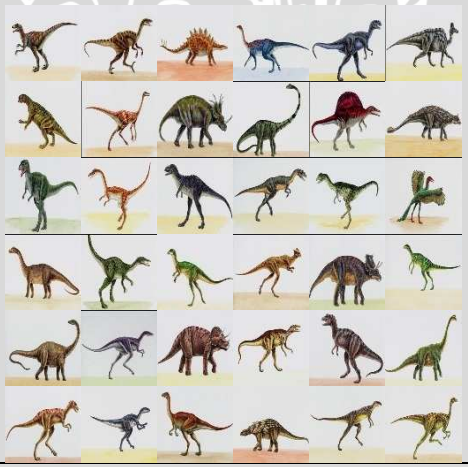
LAMPIRAN

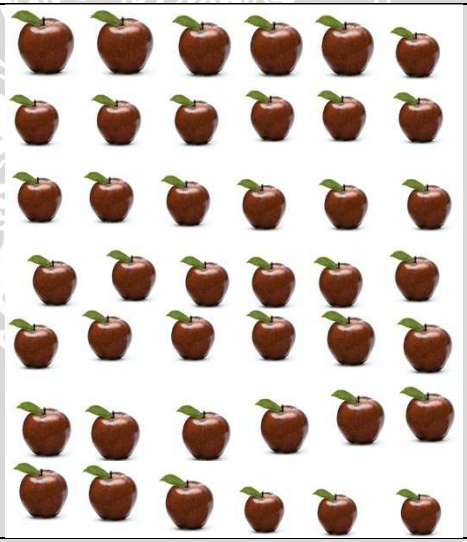
Lampiran 1Citra Data Latih

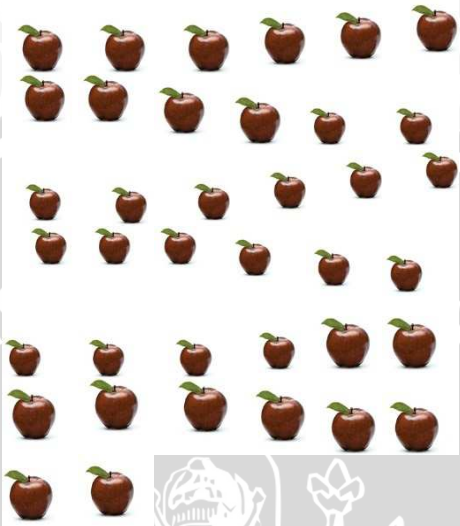
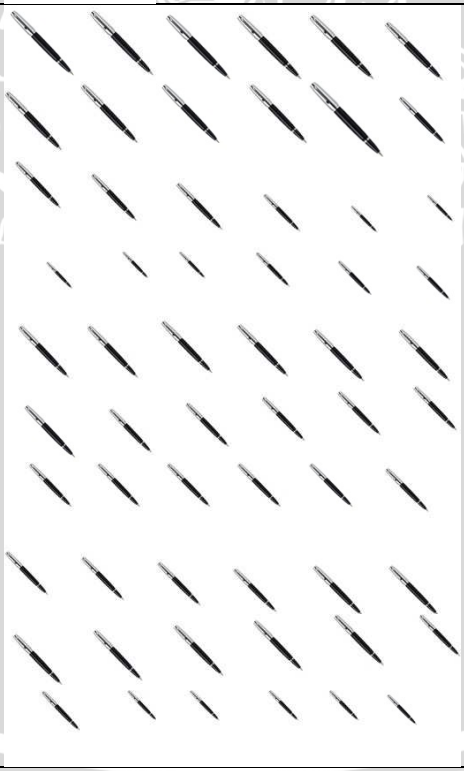
No	*Tag Metadata*	Citra
1	apel	

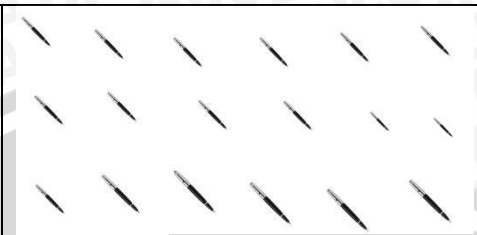
2	bolpoin	
3	pisau	

		
4	pistol	

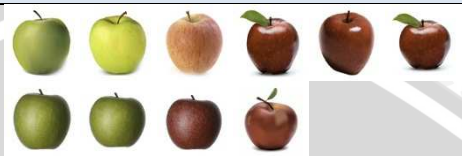
		
5	dinosaurus	

		
6	Apel hasil pengeditan	

		
7	Pisau hasil pengeditan	

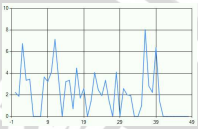
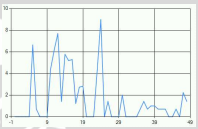
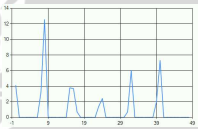
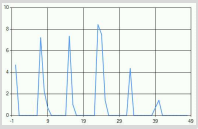
		

Lampiran 2Citra Data Uji

No	*Tag Metadata*	Citra
1	apel	
2	bolpoin	
3	pisau	
4	pistol	
5	dinosaurus	


UNIVERSITAS BRAWIJAYA

Lampiran 3 Contoh Transformasi *Wavelet* dan Ekstraksi Fitur

No	Citra Asli	Citra Hasil Transformasi	Sebaran Fitur
1			
2			
3			
4			

OTOBIOGRAFI

Nama **Andri Santoso**, lahir di Kediri 22 tahun yang lalu adalah anak pertama dari dua bersaudara. Masa kecil dihabiskan di kota yang berjuluk “Kampung Inggris”, kota Pare, Kediri.

Umur 6 tahun, menjalani pendidikan Sekolah Dasar di sebuah SD di kota Pare yaitu SDN Tulungrejo II Pare . Enam tahun kemudian, lulus dan memutuskan untuk melanjutkan studi di sebuah SMP di lingkungan Pondok Pesantren di Jombang, yaitu Pondok PesantrenDarul Ulum.

Selama menjalani pendidikan di pondok pesantren, ia bertemu dengan teman-teman dari berbagai daerah. Di pondok pesantren inilah ia bertemu dengan suasana dan teman-teman baru yang luar biasa. Setelah lulus SMP dengan predikat memuaskan, ia meneruskan pendidikannya di sebuah SMK Telekomunikasi yang masihberada di lingkungan pondok pesantren tersebut.

Setelah lulus dari SMK, ia memutuskan untuk mengikuti SNMPTN dan bersaing dengan siswa-siswa SMA se-Indonesia dengan harapan dapat menjadi mahasiswa di salah satu PTN favorit di Indonesia. Karena materi pada saat menjalani studi di SMK sangat berbeda jika dibandingkan dengan SMA, ia harus mengejar materi-materi tersebutdan berusaha dengan keras untuk mempelajari materi-materi SMA yang akan diujikan pada SNMPTN.Dengan waktu yang singkat, sekitar 3 bulan, hari-harinya dilalui dengan belajar dan latihan. Hasil dari usaha keras tersebut membuahkan hasil yang manis. Ia diterima di jurusan Ilmu Komputer di Universitas Brawijaya, Malang.

Setelah lulus nantinya, dengan moto “**Make the world a better place!**” ia ingin berkontribusi positif pada dunia dan membuat dunia menjaditempat yang lebih nyaman dan menyenangkan.


UNIVERSITAS BRAWIJAYA