PENGENALAN WAJAH MENGGUNAKAN METODE *PRINCIPAL COMPONENT ANALYSIS* DAN *ADAPTIVE RESONANCE THEORY 2*

SKRIPSI

Oleh :

NANDA GALIH ARDANA

(0410960040-96)

PROGRAM STUDI ILMU KOMPUTER

JURUSAN MATEMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS BRAWIJAYA

MALANG

2011

UNIVERSITAS BRAWIJAYA

PENGENALAN WAJAH MENGGUNAKAN METODE *PRINCIPAL COMPONENT ANALYSIS* DAN *ADAPTIVE RESONANCE THEORY 2*

SKRIPSI

Sebagai salah satu syarat untuk memperoleh gelar
Sarjana dalam bidang Ilmu Komputer

Oleh:
NANDA GALIH ARDANA
(0410960040-96)

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

2011

UNIVERSITAS BRAWIJAYA

LEMBAR PENGESAHAN SKRIPSI

PENGENALAN WAJAH MENGGUNAKAN METODE *PRINCIPAL COMPONENT ANALYSIS* DAN *ADAPTIVE RESONANCE THEORY 2*

Oleh :
Nanda Galih Ardana
0410960040-96

Setelah dipertahankan di depan Majelis Penguji
pada tanggal 2 Agustus 2011

dan dinyatakan memenuhi syarat untuk memperoleh gelar
Sarjana Komputer dalam bidang Ilmu Komputer

Pembimbing I,	Pembimbing II,
Drs. Marji, MT	Bayu Rahayudi, ST., MT
NIP. 19670801992031001	NIP. 197407122006041001

Mengetahui,
Ketua Jurusan Matematika
Fakultas MIPA Universitas Brawijaya

Dr. Abdul Rouf Alghofari, M.Sc
NIP. 19670907199203100l

UNIVERSITAS BRAWIJAYA

LEMBAR PERNYATAAN

Saya yang bertanda tangan di bawah ini :

Nama	: Nanda Galih Ardana
NIM	: 0410960040-96
Jurusan	: Matematika
Program Studi	: Ilmu Komputer
Penulis skripsi berjudul	: Pengenalan Wajah Menggunakan Metode *Principal Component Analysis* dan *Adaptive Resonance Theory 2*

Dengan ini menyatakan bahwa :

1. Isi dari Skripsi 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 Skripsi ini.
2. Apabila dikemudian hari ternyata Skripsi 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, 2 Agustus 2011
Yang menyatakan,

Nanda Galih Ardana
NIM. 0410960040-96

UNIVERSITAS BRAWIJAYA

PENGENALAN WAJAH MENGGUNAKAN METODE *PRINCIPAL COMPONENT ANALYSIS* DAN *ADAPTIVE RESONANCE THEORY 2*

ABSTRAK

Beberapa penelitian tentang pengenalan wajah pernah dibuat. Salah satunya adalah menggunakan PCA (*Principal Component Analysis*) dan ART-2A (*Adaptive Resonance Theory 2A*). Hasil penelitian menunjukkan bahwa menggunakan metode PCA dan ART-2A dapat mengenali wajah dengan tingkat keakuratan 97,8 %.

Pada penelitian akhir ini akan diimplementasikan pengenalan wajah menggunakan PCA (*Principal Component Analysis*) dan ART-2 (*Adaptive Resonance Theory 2*). Metode ART-2 dipilih karena kemampuannya untuk mengelompokkan pola input dengan jumlah acak dan memiliki karakteristik vektor input analog (kontinu). *Principal Component Analysis* (PCA) merupakan salah satu metode yang umum digunakan mendapatkan informasi ciri yang penting dari citra wajah dan mereduksi dimensi data. Reduksi dimensi digunakan untuk memudahkan kompleksitas perhitungan dengan tidak mengurangi hasil akhir pengenalan. Dengan melakukan ini, bisa mengurangi sensitivitas metode untuk varian substansial antara citra wajah disebabkan oleh ukuran besar, ekspresi atau variasi iluminasi.

Dari hasil penelitian didapatkan tingkat akurasi untuk proses klasifikasi terbaik sebesar 71,19 %. Dengan tingkat akurasi ini, maka metode ART-2 tidak memberikan hasil pengenalan wajah yang lebih baik dibandingkan menggunakan metode ART-2A.

Kata kunci : *pengenalan wajah, Adaptive Resonance Theory, ART-2, ART-2A, ekstraksi ciri, Principal Component Analysis (PCA)*.

UNIVERSITAS BRAWIJAYA

FACE RECOGNITION BASED ON PRINCIPAL COMPONENT ANALYSIS AND ADAPTIVE RESONANCE THEORY 2

ABSTRACT

A number of studies based on face recognition has been made. One of the studied was based on PCA (Principal Component Analysis) and ART-2A (Adaptive Resonance Theory 2A). The results showed that recognition rate accuracy when using the PCA and ART-2A to identify faces is 97,8 %.

In this research, ART-2 and PCA is applied to the problem of face recognition. ART-2 chosen because its ability to cluster arbitrary number input patterns combining with its characteristics of both analog and digital input vectors. Principal Component Analysis (PCA), is the one of most popular approaches to extract specific facial components with dimensionality reduction. Dimensionality reduction used to ease the complexity of the calculation and reduce methods sensitivity of variance substantial between face due to the size, expression or illumination.

The result obtained from the experiment show that the accuracy rate with the best classification was 71,19 %. From the accuracy archieved, the ART-2 is not give a better result compared with the ART-2A in a face recognition problem.

Keywords : *face recognition, Adaptive Resonance Theory, ART-2, ART-2A, feature extraction, Principal Component Analysis*.

UNIVERSITAS BRAWIJAYA

KATA PENGANTAR

Alhamdulillahi Robbil alamin, puji syukur penulis panjatkan kepada Allah SWT atas segala limpahan rahmat dan hidayah-Nya, sehingga penulis dapat menyelesaikan tugas akhir ini sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dalam bidang Ilmu Komputer.

Tugas akhir ini bertujuan untuk melakukan perancangan dan implementasi pengenalan wajah dengan menggunakan metode PCA dan ART-2. Dengan mengetahui hasil pengenalan dari metode ini, diharapkan dapat dijadikan sebagai sebuah masukan untuk membuat aplikasi pengenalan wajah dengan metode yang lebih baik.

Pada penyusunan tugas akhir ini, penulis ingin mengucapkan terima kasih kepada :

1. Drs. Marji MT., selaku pembimbing utama penulisan tugas akhir sekaligus Ketua Program Studi Ilmu Komputer, Jurusan Matematika, FMIPA Universitas Brawijaya.
2. Bayu Rahayudi, ST. MT, selaku pembimbing pendamping dalam penulisan tugas akhir.
3. Kasyful Amron, ST. selaku pembimbing akademik.
4. Dr. Abdul Rouf Alghofari, M.Sc., selaku Ketua Jurusan Matematika, FMIPA Universitas Brawijaya.
5. Segenap bapak dan ibu dosen yang telah mendidik dan mengajarkan ilmunya kepada penulis selama menempuh pendidikan di Program Studi Ilmu Komputer Jurusan Matematika FMIPA Universitas Brawijaya.
6. Segenap staf dan karyawan di Jurusan Matematika FMIPA Universitas Brawijaya yang telah banyak membantu penulis dalam pelaksanaan penyusunan tugas akhir ini.
7. Orang tua penulis atas dukungan materi dan doa restunya kepada Penulis.
8. Rekan-rekan di Program Studi Ilmu Komputer FMIPA Universitas Brawijaya yang telah banyak memberikan bantuannya demi kelancaran pelaksanaan penyusunan tugas akhir ini.
9. Dan semua pihak yang telah membantu dalam penyusunan tugas akhir ini yang tidak dapat penulis sebutkan satu per satu.

Penulis sadari bahwa masih banyak kekurangan dalam laporan ini, oleh karena itu penulis sangat menghargai saran dan kritik yang sifatnya membangun demi perbaikan penulisan dan mutu isi tugas akhir ini untuk kelanjutan penelitian serupa di masa mendatang.

Semoga laporan tugas akhir ini dapat bermanfaat. Amin.

Malang, 2 Agustus 2011

Penulis

DAFTAR ISI

BAB III METODOLOGI DAN PERANCANGAN

BAB IV HASIL DAN PEMBAHASAN

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR *SOURCE CODE*

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan dunia komputer sangat pesat seiring dengan semakin banyaknya ancaman terhadap keamanan sistem komputer. Penggunaan pengamanan konvensional seperti password ataupun kartu memang masih banyak digunakan namun tidak cukup handal karena password dan kartu dapat saja digunakan oleh pihak yang tidak berwenang.

Pemakaian identifikasi biometrik dapat dijadikan sebagai suatu alternatif untuk pengamanan sistem. Identifikasi biometrik didasarkan pada karakteristik alami manusia, yaitu karakteristik fisiologis dan karakteristik perilaku seperti wajah, sidik jari, suara, telapak tangan, iris dan retina mata, DNA, dan tandatangan.

Pengenalan wajah merupakan salah satu bidang identifikasi biometrik yang sering digunakan dalam berbagai aplikasi seperti identifikasi ahli, pengenalan wajah saksi, bank/toko dan departemen keamanan. Pengenalan wajah merupakan proses memberi label sebuah citra ke dalam kelas citra yang bersesuaian. Permasalan umum dalam bidang pengenalan pola, khususnya pengenalan wajah adalah dimensional data.

Principal Component Analysis (PCA) adalah salah satu metode yang umum digunakan dalam reduksi dimensi. PCA merupakan metode statistik yang mampu mengekstrak komponen wajah secara spesifik dan mengurangi besar dimensi data tanpa banyak kehilangan informasi dari data yang dipresentasikan. PCA juga dikenal sebagai ekspansi *Karhunen-Loève*, *eigenpicture*, dan *eigenface*.

Permasalahan lainnya dalam pengenalan pola adalah membutuhkan sistem klasifikasi yang cepat, tangguh dan handal. Jaringan saraf tiruan (JST) merupakan sistem pengklasifikasian yang paling sukses karena kemampuan non linier pada jaringannya. Tetapi jaringan saraf kompetitif standar tidak memiliki pola pembelajaran yang stabil karena permasalahan kemampuan adaptasi jaringan, sehingga informasi sebelumnya akan cenderung dilupakan. Permasalahan ini dipecahkan oleh metode *Adaptive Resonance Theory* (ART). ART memiliki kemampuan untuk menerima informasi baru tanpa melupakan informasi sebelumnya.

Dari beberapa penelitian sebelumnya tentang pengenalan wajah menggunakan PCA (*Principal Component Analysis*) dan ART-2A (*Adaptive Resonance Theory 2A*) didapatkan tingkat keakuratan pengenalan wajah sebesar 97,8%. (Mahyabadi, M.P, Soltanizadeh, H. dan Shokouhi, Sh.B, 2006).

Pada penelitian tugas akhir ini akan diaplikasikan pengenalan wajah menggunakan PCA (*Principal Component Analysis*) dan ART-2 (*Adaptive Resonance Theory 2*). Metode ART-2 dipilih karena kemampuannya untuk mengelompokkan pola input dengan jumlah acak dan memiliki karakteristik vektor input analog (kontinu). PCA digunakan untuk ekstraksi fitur dan mereduksi dimensi sehingga memudahkan kompleksitas perhitungan dengan tidak mengurangi hasil akhir pengenalan. Dengan melakukan ini, bisa mengurangi sensitivitas metode untuk varian substansial antara citra wajah disebabkan oleh ukuran besar, ekspresi atau variasi iluminasi. Metode ART-2 diharapkan dapat menghasilkan suatu sistem yang mampu memberikan *output* berupa pengenalan wajah dengan tingkat akurasi yang lebih baik.

1.2 Rumusan Masalah

Berdasarkan uraian pada latar belakang masalah, maka dalam skripsi ini dapat dirumuskan permasalahan sebagai berikut:

1. Bagaimana perancangan dan implementasi untuk pengenalan wajah dengan metode PCA dan ART-2
2. Bagaimana tingkat akurasi yang dihasilkan dari pengenalan wajah menggunakan metode PCA dan ART-2

1.3 Batasan Masalah

Dari permasalahan yang telah dijelaskan, berikut ini diberikan batasan masalah untuk menghindari melebarnya masalah yang akan diselesaikan :

1. Menggunakan *database* wajah manusia *AT&T Laboratories Cambridge* untuk citra latih dan uji.
2. Format citra adalah citra *grayscale* berukuran 92x112 piksel.
3. Ekstraksi fitur menggunakan *Principal Component Analysis* (PCA).
4. Klasifikasi dan pengenalan menggunakan *Adaptive Resonance Theory* 2 *(ART-2)*.

1.4 Tujuan Penelitian

Tujuan yang ingin dicapai dari adalah :

1. Merancang dan mengimplementasi metode PCA dan ART-2 dalam pengenalan wajah.
2. Mengetahui tingkat akurasi yang dihasilkan dari pengenalan wajah menggunakan PCA dan ART-2

1.5 Manfaat Penelitian

1. Menyediakan aplikasi yang mampu mengenali wajah manusia sehingga dapat digunakan untuk sistem keamanan dan absensi.
2. Mengetahui tingkat akurasi dari penerapan PCA dan ART-2.

1.6 Sistematika Penulisan

Skripsi ini disusun berdasarkan sistematika penulisan sebagai berikut :

BAB I PENDAHULUAN
Berisi latar belakang masalah, tujuan penulisan, perumusan masalah, batasan masalah, manfaat penulisan dan sistematika penulisan.

BAB II TINJAUAN PUSTAKA
Berisi penjelasan tentang pengenalan wajah secara umum, citra, PCA (*Principal Component Analysis*) dan ART-2 (*Adaptive Resonance Theory 2*).

BAB III METODOLOGI DAN PERANCANGAN
Pada bab ini akan dijelaskan mengenai metode-metode yang digunakan dalam pembuatan perangkat lunak pengenalan wajah menggunakan metode PCA dan ART-2.

BAB IV HASIL DAN PEMBAHASAN
Pada bab ini berisi tentang penjelasan implementasi sistem dan hasil pengujian yang dilakukan.

BAB V KESIMPULAN DAN SARAN
Berisi kesimpulan dari seluruh rangkaian penelitian dan saran untuk pengembangan lebih lanjut.


UNIVERSITAS BRAWIJAYA

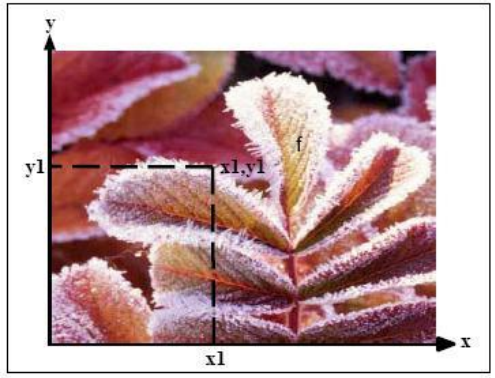
BAB II
TINJAUAN PUSTAKA

2.1 Citra

Citra *(image)* dapat diartikan suatu representasi, kemiripan atau imitasi dari suatu obyek atau benda. Sedangkan pengertian citra dari sudut pandang matematis, merupakan fungsi menerus (kontinu) dari intesitas cahaya pada bidang dua dimensi dimana x dan y adalah koordinat spasial pada bidang tersebut dan nilai $f(x,y)$ menyatakan intensitas atau tingkat kecerahan. Secara matematis persamaan sistematis untuk fungsi intensitas $f(x,y)$ adalah

$$0 < f(x,y) < \infty \qquad (2.1)$$

Gambar 2.1 merupakan ilustrasi dari persamaan diatas.

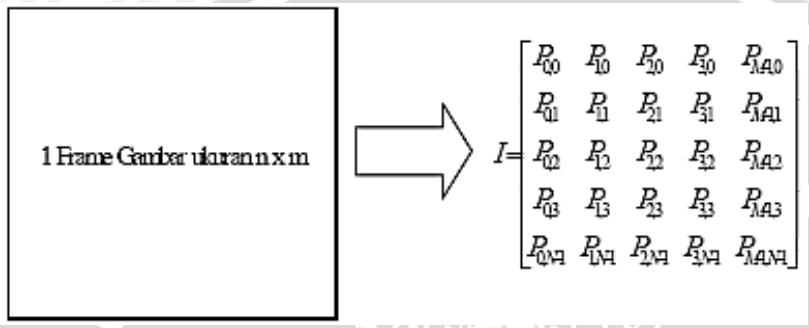


Gambar 2.1 Citra digital sebagai fungsi

Citra digital adalah gambar dua dimensi yang dapat ditampilkan pada layar monitor komputer sebagai himpunan berhingga (diskrit) nilai digital yang disebut *pixel (picture elements)*. Piksel dapat didefinisikan sebagai elemen terkecil dari sebuah citra digital yang menentukan resolusi citra tersebut.

Citra dinyatakan dalam bentuk data matriks 2 dimensi, dimana setiap titik data mewakili satu piksel. Dalam hubungannya dengan data video, maka satu gambar (*image*) dikenal sebagai satu frame. Misalnya sebuah gambar dikatakan resolusinya sebesar 800 x 600 maka berarti panjang piksel horisontalnya 800 dan panjang piksel

vertikalnya 600 dan jumlah total keseluruhan piksel dari gambar tersebut yaitu 480.000 atau dapat dikatakan bahwa gambar tersebut terdiri dari 480.000 piksel. Dimensi matriks yang mewakili 1 frame citra dengan ukuran *M x N* ditunjukkan pada Gambar 2.2.



Gambar 2.2 Data Matriks Dua Dimensi

2.2 Warna Tingkat Keabuan (*Grayscale*)

Kecerahan dari citra yang disimpan dengan cara pemberian nomor pada tiap-tiap piksel-nya. Semakin tinggi nomor piksel-nya maka makin terang (putih) piksel tersebut. Sedangkan semakin kecil nilai suatu piksel, mengakibatkan warna pada piksel tersebut menjadi gelap. Dalam sistem kecerahan yang umum terdapat 256 tingkat untuk setiap piksel. Skala kecerahan seperti ini dikenal sebagai *grayscale*.

Dalam sistem komputasi maupun fotografi, suatu citra *grayscale* atau tingkat keabuan citra digital adalah citra di mana nilai setiap piksel memiliki satu nilai sampel, yaitu hanya membawa informasi intensitas citra. Foto semacam ini, juga dikenal sebagai foto hitam-putih, yang tersusun secara eksklusif dari nuansa abu-abu yang bervariasi dari nilai hitam pada intensitas terlemah menuju nilai putih yang terkuat.

Citra *grayscale* memiliki perbedaan dari satu-bit citra hitam-putih, dalam konteks pencitraan komputer citra yang memilki hanya dua warna yaitu, hitam dan putih (juga disebut *bilevel* atau gambar biner). Citra *grayscale* juga disebut citra monokromatik yang menunjukkan tidak adanya variasi warna. Citra *grayscale* ditunjukan pada Gambar 2.3.

Gambar 2.3 Citra *Grayscale*

2.3 Ekstraksi Fitur (*Feature Extraction*)

Ekstraksi fitur adalah proses untuk mendapatkan ciri-ciri/fitur pembeda yang dapat digunakan untuk membedakan antara wajah antar individu yang berbeda, bagi sebagian besar aplikasi *pattern recognition*, teknik ekstraksi fitur yang handal merupakan kunci utama dalam penyelesaian masalah *pattern recognition* (Turk dan Pentland, 1991).

Banyak metode yang digunakan untuk ekstraksi fitur. Beberapa metode yang umum digunakan yaitu, PCA (*Principal Component Analysis*), DCT (*Discrete Cosinus Transform*), *Gabor Filter*, metode *moment*, dan *wavelet transform*.

2.3.1 Fitur Wajah

Dalam dunia matematika, *eigenvalue* (nilai eigen), *eigenvector* (vektor eigen), dan *eigenspace* (ruang eigen) memiliki keterkaitan dalam suatu ruang lingkup bidang aljabar linier. Kesemuanya itu merupakan hal yang penting dalam fitur wajah, khususnya dengan metode PCA. Kata *eigen* berasal dari Jerman yang berarti bawaan, berbeda, dan diri. Dalam aljabar linier, dipelajari mengenai transformasi linier yang diwakili oleh matriks yang bekerja pada vektor. *Eigenvalue, eigenvector* dan *eigenspaces* adalah properti dari sebuah matriks. Ketiganya dapat memberikan informasi penting tentang sebuah matriks dan dapat digunakan dalam faktorisasi matriks. Selain itu, ketiganya juga dapat digunakan dalam bidang matematika terapan yang beragam seperti keuangan dan mekanika kuantum.

Pada umumnya, sebuah matriks yang berperan sebagai vektor dapat diubah magnitude dan arah. Sebuah matriks dapat berperan pada sebuah vektor yang tepat dengan cara merubah nilai magnitude-nya dan tanpa merubah arahnya. Vektor inilah yang disebut *eigenvector*. Matriks juga dapat dihasilkan dari hasil perkalian nilai *eigenvector* dengan faktor pengali yang disebut *eigenvalue*. Jika nilai *eigenvalue* bernilai positif maka arahnya tidak berubah, sedangkan yang bernilai negatif arahnya terbalik. Sedangkan *eigenspace* merupakan himpunan *eigenvector* yang memiliki *eigenvalue* yang sama.

Secara formal jika matriks A adalah transformasi linier, vektor x adalah *eigenvector* dari matriks A dan λ adalah matriks skalar seperti yang ditunjukkan dengan formula pada persamaan 2.2 (Anton dan Rorres, 2005).

$$Ax = \lambda x \qquad \textbf{(2.2)}$$

Nilai matriks skalar λ dapat dikatakan sebagai *eigenvalue* dari matriks A yang bersesuaian dengan *eigenvector x*. Proses menghitung *eigenvalue* dan *eigenvector* dari matriks bujur sangkar dapat diturunkan menjadi tahap-tahap seperti yang ditunjukkan dalam persamaan 2. 3 berikut (Anton dan Rorres, 2005).

$$\begin{aligned} &a.\ Ax = \lambda Ix \\ &b.\ (\lambda I - A)x = 0 \\ &c.\ \det(\lambda I - A) = 0 \end{aligned} \qquad \textbf{(2.3)}$$

Keterangan :

1. *Eigenvalue* (λ) dari matriks bujur sangkar (A).
2. *I* adalah matriks identitas.

Setelah didapatkan nilai eigen, maka substitusikan kembali ke persamaan untuk mendapatkan vektor eigen.

2.3.2 *Principal Component Analysis* (PCA)

PCA juga dikenal dengan nama *Karhunen–Loève transform* (KLT), *Hotelling transform* atau *proper orthogonal decomposition* (POD) merupakan sebuah metode untuk mengambil ciri-ciri penting dari sekumpulan data dengan melakukan dekomposisi terhadap data

tersebut, sehingga menghasilkan koefisien-koefisien yang tidak saling berkolerasi.

Tujuan dari PCA adalah mengambil variasi total pada wajah-wajah yang dilatihkan dan menjelaskan variasi tersebut dengan variabel-variabel yang jumlahnya lebih sedikit. Dengan kata lain PCA digunakan untuk merepresentasikan data dalam dimensi yang rendah sehingga waktu komputasi dapat dikurangi dan kompleksitas dari wajah yang tidak perlu dapat dihilangkan (Silaen, 2006).

Teknik ekstraksi ciri yang sederhana menggunakan PCA yang digunakan untuk mengekstrak ciri dari citra yang ternormalisasi. PCA menghasilkan vektor-vektor eigen atau vektor-vektor karakteristik yang kemudian akan digunakan untuk membentuk ruang eigen. Langkah-langkah proses ekstraksi ciri dalam PCA (Turk dan Pentland, 1991) yaitu membentuk matrik wajah, mencari matrik rata-rata, menghitung matrik kovarian, menentukan fitur PCA.

2.3.2.1 Pembentukan Matrik Wajah

Sebuah wajah yang merupakan sebuah citra 2 dimensi dapat dilihat sebagai sebuah vektor 1 dimensi. Jika panjang dan lebar citra adalah *M* dan *N* piksel, maka jumlah komponen dari vektor ini adalah *n* (*n=M x N)*.

Misalkan terdapat *m* citra dengan masing-masing berdimensi 100 x 100 piksel = 10.000 piksel. Matriks yang mempresentasikan citra-citra tersebut berdimensi *10.000 x m*, adalah

$$X = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdot & \cdot & x_{1,m} \\ x_{2,1} & x_{2,2} & \cdot & \cdot & x_{2,m} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ x_{10.000,1} & x_{10.000,2} & \cdot & \cdot & x_{10.000,m} \end{bmatrix}$$

X = Matriks wajah

2.3.2.2 Mencari Matrik Rata-Rata

Citra rata-rata adalah rata-rata dari seluruh piksel citra inputan.

$$\bar{u} = \frac{1}{m}\sum_{i=1}^{m} X_i \qquad \textbf{(2.4)}$$

Dari matrik diatas, didapatkan matrik rata-rata citra dengan dimensi *1 x m*.

$$\bar{u} = [\bar{u}_1 \quad \bar{u}_2 \quad \bar{u}_3 \quad \bar{u}_4 \quad \dots \quad \bar{u}_m]$$

Kemudian mencari matrik rata-rata dengan cara matrik wajah dikurangi rata-rata citra

$$Y = X - \bar{u} \qquad \textbf{(2.5)}$$

Y = Matrik normalisasi
$\bar{u}$ = Citra rata-rata

2.3.2.3 Menghitung Matrik Kovarian

Matrik kovarian diperoleh dengan mengalikan matrik *Y* dengan matrik transposenya.

$$C = Y * Y^T \qquad \textbf{(2.6)}$$

Selanjutnya dilakukan dekomposisi *eigen* sehingga berlaku persamaan berikut

$$C * v = \lambda * v \qquad \textbf{(2.7)}$$

C = Matrik kovarian
λ = *Eigenvalue*
v = *Eigenvector*

Dimensi *v* dan λ adalah matrik berdimensi *10.000 x 10.000* (*n x n*) sebagai berikut

$$v = \begin{bmatrix} v_{1,1} & v_{1,2} & \cdot & v_{1,n} \\ v_{2,1} & v_{2,2} & \cdot & v_{2,n} \\ \cdot & \cdot & \cdot & \cdot \\ v_{n,1} & v_{n,2} & \cdot & v_{n,n} \end{bmatrix}, \lambda = \begin{bmatrix} \lambda_{1,1} & 0 & \cdot & 0 \\ 0 & \lambda_{2,2} & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \cdot & \lambda_{n,n} \end{bmatrix}$$

Komputasi yang dibutuhkan untuk menghitung *eigenvector* berdimensi *n* (10.000) akan memakan waktu yang sangat lama karena dimensinya yang sangat besar. Menurut Turk dan Pentland (Turk dan Pentland, 1991), permasalahan *n* dimensi *eigenvector* dapat diselesaikan dengan mencari *eigenvector* dari matrik kovarian berdimensi *m x m (m<n)*

$$C = Y^T * Y \quad \textbf{(2.8)}$$

Sehingga dekomposisi *eigen* berlaku sebagai berikut

$$Y^T Y * v = \lambda * v \quad \textbf{(2.9)}$$

Mengalikan kedua sisi dengan *Y*

$$YY^T * Yv = \lambda * Yv \quad \textbf{(2.10)}$$

Didapatkan *eigenvector* (v_{exp}) dari $C = YY^T$ adalah Yv

$$v_{exp} = Y * v \quad \textbf{(2.11)}$$

2.3.2.4 Menentukan Fitur PCA

Fitur adalah komponen-komponen dari citra-citra *training* yang didapatkan dari hasil proses *training*. Fitur inilah yang akan digunakan untuk mengidentifikasikan suatu citra yang akan dikenali.

Fitur diperoleh dengan cara mentransformasikan citra normalisasi ke dalam *eigenvector* yang terpilih, dengan menggunakan persamaan berikut

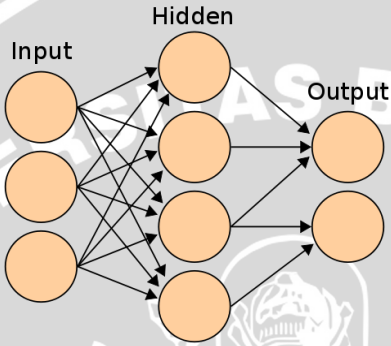
$$f_{PCA} = \sum_{k=1}^{m} (v_{exp})_{lk}^T * Y_k \quad l = 1, \ldots, M \quad \textbf{(2.12)}$$

f_{PCA} = Fitur PCA
M = Jumlah *eigenvector* yang terpilih

2.4 Jaringan Saraf Tiruan

Jaringan Saraf Tiruan (JST) adalah suatu sistem pengolahan informasi yang cara kerjanya menirukan cara kerja jaringan saraf

manusia. Jaringan saraf tiruan tersusun atas beberapa elemen pemroses yaitu neuron, unit, sel atau node, yang saling terhubung dalam bentuk grafik terarah *(directed graph)* melalui jalur sinyal searah yang disebut dengan koneksi. Struktur JST ditunjukkan pada Gambar 2.4



Gambar 2.4 Jaringan Saraf Tiruan
Sumber: Wikipedia. Tanggal akses: 14-10-2010

Suatu jaringan saraf minimum tersusun atas lapisan masukan dan lapisan keluaran. Dalam beberapa tipe jaringan diantara lapisan masukan (*input layer*) dan lapisan keluaran (*output layer*) terdapat lapisan tersembunyi (*hidden layer*). Hal ini berarti bahwa semua neuron pada lapisan masukan akan berhubungan ke semua neuron dalam lapisan tersembunyi yang selanjutnya setiap unit dalam lapisan tersembunyi nantinya akan dihubungkan ke semua neuron di lapisan keluaran. Pada setiap lapisan biasanya neuron mempunyai fungsi aktifasi serta pola hubungan ke neuron lain yang sama.

Seperti halnya jaringan saraf biologis, JST juga memiliki kemampuan untuk belajar dan beradaptasi terhadap masukan-masukannya. JST tidak perlu diprogram secara eksplisit, karena JST dapat belajar dari beberapa contoh pelatihan. Proses pembelajaran pada jaringan saraf tiruan dapat digolongkan menjadi tiga bagian, yaitu

1. Belajar dengan pengawasan (*Supervised Learning*)
   Pada metode *supervised learning* jaringan diberikan vektor target yang harus dicapai sebagai dasar untuk mengubah hubungan interkoneksi atau bobot pada jaringan.

Contoh jaringan yang belajar dengan pengawasan adalah *Backpropagation* dan *Perceptron*.

2. Belajar tanpa pengawasan (*Unsupervised Learning*)
   Berbeda dengan *supervised learning* yang masih membutuhkan pengawasan dan sangat tergantung pada *knowledge* yang diberikan pada pembangunan jaringan, metode *unsupervised learning* ini melatih jaringan terhadap suatu masukan tanpa adanya pengawasan, dimana vektor target tidak ditentukan. Vektor masukan dimasukkan ke dalam jaringan, kemudian sistem akan mengatur dirinya sendiri sedemikian rupa sehingga dihasilkan keluaran yang konsisten bilamana pola yang menyerupai vektor masukan tersebut diberikan.
   Contoh jaringan yang belajar tanpa pengawasan adalah *Adaptive Resonance Theory* (ART), *Self Organizing Map* (SOM), *Kohonen* dan *Learning Vektor Quantification* (LVQ).
3. Penguatan Belajar (*Reinforcement Learning*)
   Reinforcement learning adalah pengembangan metode supervised learning dimana memberikan beberapa umpan balik dari lingkungan yang telah ditentukan. Namun sinyal umpan balik tersebut bersifat evaluatif, bukan instruktif. Reinforcement learning sering disebut juga pembelajaran dengan kritik sebagai lawan untuk pembelajaran dengan seorang guru (supervised learning).
   Contoh jaringan yang belajar dengan kritik adalah *Markov Decision Process* (MDP).

2.5 *Adaptive Resonance Theory* (ART)

Pada metode JST yang umum, yaitu JST dengan pelatihan terbimbing, informasi-informasi lama cenderung untuk dilupakan, sehingga untuk mengenali pola yang baru diperlukan perubahan dengan memasukkan pola-pola baru tersebut dalam parameter-parameter belajarnya.

Hal ini disebut oleh Gossberg (Grossberg, 1976) sebagai *stability-plasticity dilemma*. Dilema ini dapat dinyatakan dengan deretan pertanyaan sebagai berikut

1. Bagaimana suatu sistem yang belajar tetap adaptif (plastis) terhadap masukan penting, dan tetap stabil terhadap masukan yang tidak relevan?

2. Bagaimana sistem mengetahui saatnya untuk beralih dari keadaan plastis ke keadaan stabil?
3. Bagaimana sistem dapat mempertahankan informasi terdahulu sementara sedang mempelajari hal-hal baru?

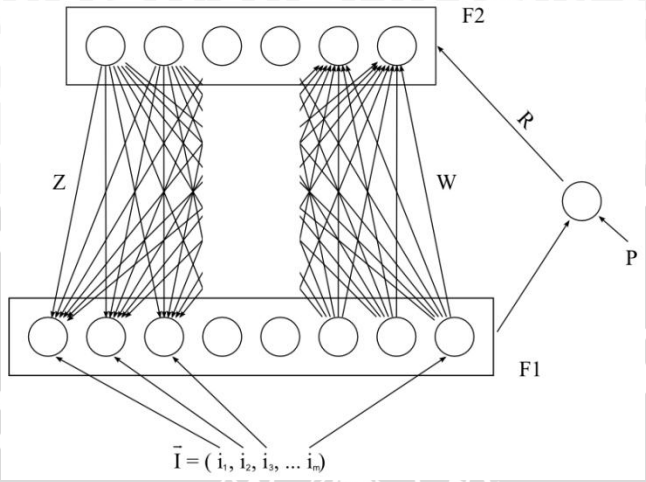
Masalah tersebut dapat diatasi dengan menambahkan mekanisme umpan balik (*feedback*) antara lapis kompetitif dengan lapis masukan pada jaringan. Mekanisme ini memberikan kemampuan pada jaringan untuk dapat mempelajari pola-pola baru tanpa kehilangan pola-pola yang telah dipelajari sebelumnya, peralihan secara otomatis dari keadaan stabil ke keadaan plastis, dan kestabilan kelas yang dilakukan oleh neuron-neuron.

Pendekatan ini menghasilkan arsitektur jaringan saraf tiruan model *Adaptive Resonance Theory* (ART) yang dikembangkan oleh Stephen Grossberg dan Gail A. Carpenter pada tahun 1987, (Carpenter dan Grossberg, 1987) yang didesain untuk meniru bagaimana cara otak memproses informasi. Struktur dasar ART dapat dilihat pada Gambar 2.5

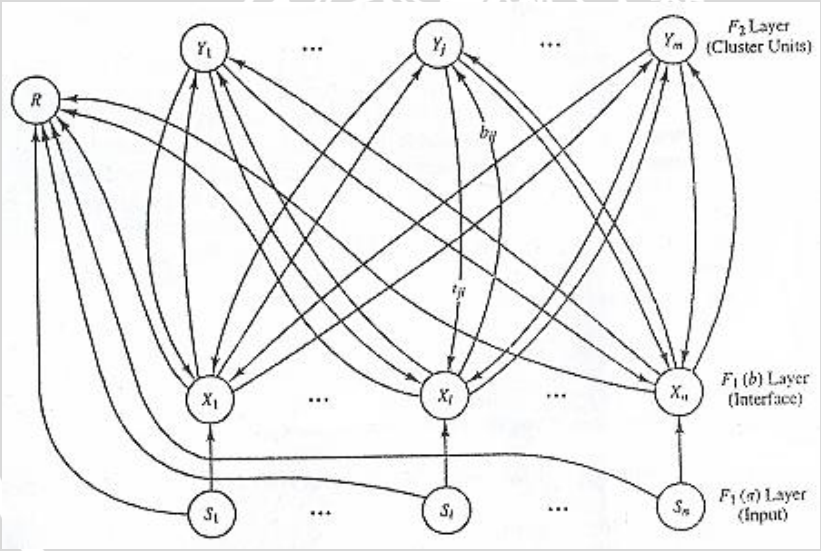
Sistem ART terbentuk dari dua sistem yaitu attentional system dan orienting system (Tanaka dan Weitzefeld, 2002). Attentional system terdiri dari jaringan kompetitif yaitu lapis masukan F1, lapis keluaran F2 dan mekanisme penguat sinyal (*gain control*). Sedang *orienting system* hanya terdiri dari mekanisme reset yang berfungsi untuk mengatur sub-sistem *attentional* agar berjalan secara dinamis.

Pada jaringan ART, informasi dialirkan kedepan dan kebelakang diantara lapisan sel-selnya (LTM). Jika telah terbentuk pola yang sesuai, maka terjadi osilasi yang stabil. Proses osilasi inilah yang dinamakan resonansi. Selama resonansi ini terjadi proses belajar dan adaptasi.

Kondisi resonansi dapat dicapai dengan dua cara, yaitu jika jaringan melakukan pengenalan terhadap masukan vektor masukan yang telah dipelajari sebelumnya, maka kondisi resonansi akan tercapai dengan cepat. Kedua, jika vektor masukan tidak segera dikenal, maka jaringan akan mencari kesesuaian dengan seluruh pola yang tersimpan didalamnya. Apabila kesesuaian tidak tercapai, maka jaringan akan memasuki kondisi resonansi dengan cara menyimpan pola baru tersebut untuk pertama kali. Dengan demikian, jaringan bereaksi cepat terhadap masukan yang pernah dipelajari dan juga mampu belajar jika masukan baru diberikan.



Gambar 2.5 Struktur ART
Sumber: Wikipedia. Tanggal akses: 24-11-2010



Gambar 2.6 Arsitektur ART-1
Sumber: Laurene Fausett, 1994

2.5.1 Arsitektur ART-1

ART-1 adalah arsitektur pertama dan paling mendasar dimana mampu mempelajari dan mengenali pola masukan berupa biner (digital) (Carpenter dan Grossberg, 1987).

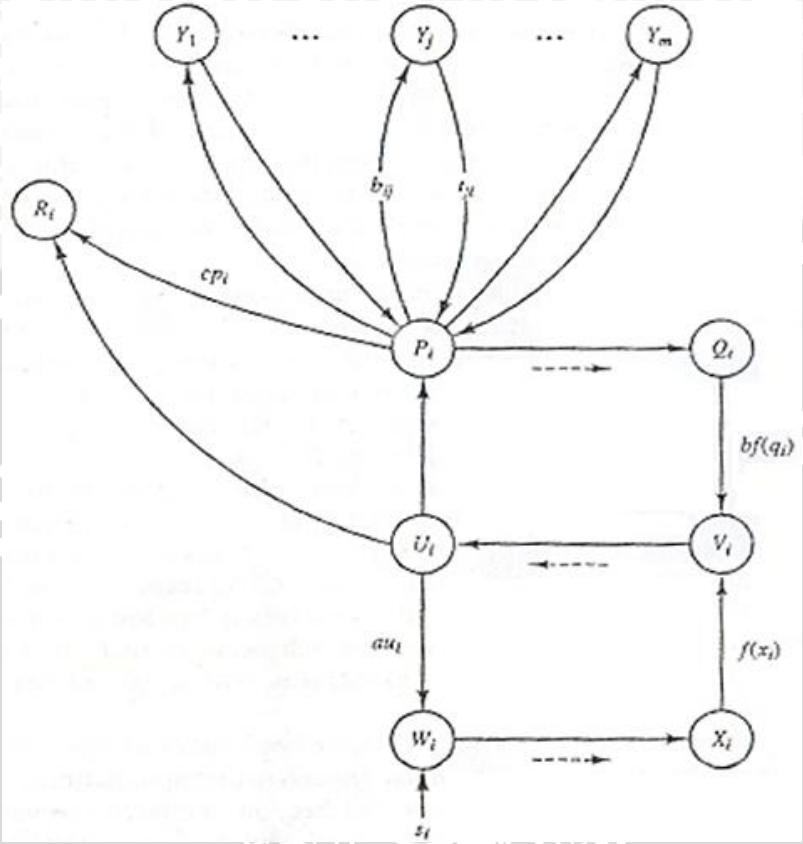
Arsitektur JST ART-1 terdiri dari sebuah lapisan masukan (F1), lapisan keluaran (F2) dan sebuah mekanisme untuk mengontrol derajat kesamaan pola-pola yang ditempatkan pada sel yang sama (mekanisme reset). Arsitektur ART-1 ditunjukkan pada Gambar 2.6

2.5.2 Arsitektur ART-2

ART-2 merupakan penyempurnaan dari arsitektur ART-1 (Carpenter dan Grossberg, 1987). Karakteristik yang membedakan antara keduanya adalah pada tipe masukan di masing-masing lapis F1. Pada ART-2 mempunyai vektor input biner (digital) dan kontinu (analog).

Struktur ART-2 secara umum sama dengan ART-1, yaitu terdiri dari sebuah lapisan masukan (F1), lapisan keluaran (F2) dan sebuah mekanisme untuk mengontrol derajat kesamaan pola-pola yang ditempatkan pada sel yang sama (mekanisme reset). Pada ART-2 terdapat penambahan sebuah kombinasi dari normalisasi dan penekanan derau pada lapisan F1 karena vektor masukannya bernilai kontinu, yang kemungkinan nilainya sangat berdekatan. Arsitektur ART-2 ditunjukkan pada Gambar 2.7.

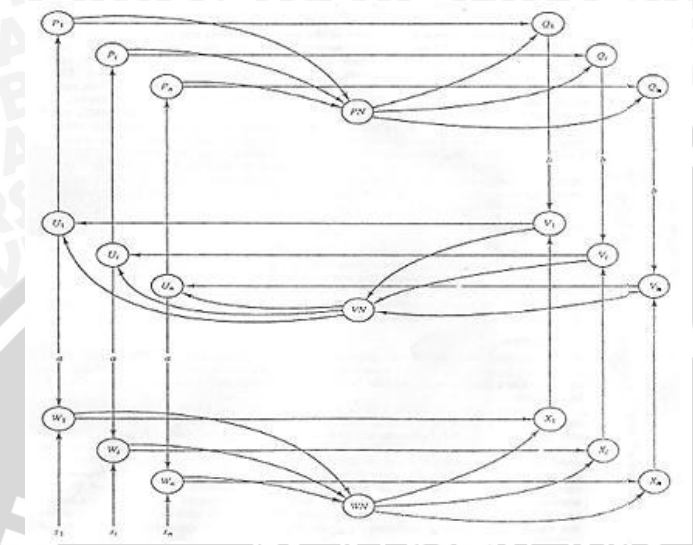
Lapisan F1 terdiri atas 6 jenis unit masukan yaitu **w**, **x**, **u**, **v**, **p** dan **q**. Setiap unit dari masing-masing jenis masukan terdiri atas n unit, dimana n menunjukkan dimensi dari pola masukan. Simbol-simbol yang ada pada jalur-jalur yang menghubungkan antara bermacam-macam unit pada lapisan F1 menunjukkan transformasi yang terjadi pada sinyal yang melalui tipe unit satu ke tipe unit selanjutnya, dan bukan menunjukkan perkalian untuk nilai yang diberikan. Kecuali untuk hubungan unit-unit $\boldsymbol{p_i}$, pada pada lapisan F1, dengan $\boldsymbol{y_j}$ pada lapisan F2 yang memang menunjukkan bobot yang dikalikan pada setiap sinyal yang ditransmisikan melalui jalur-jalur antara keduanya. Aktifasi dari unit di lapisan F2 yang menjadi pemenang adalah *d*, dimana nilainya antara 0 hingga 1 ($0 < d < 1$).



Gambar 2.7 Arsitektur ART-2
Sumber: Laurene Fausett, 1994

Sebuah unit sisipan diantara unit **w** dan **x** untuk menghitung normalisasi dari vektor **w**, dan mengirimkan sinyal ini pada setiap unit **x**, dimana unit **x** menerima sinyal dari semua unit **w**. Sisipan yang serupa dan fungsi yang sama juga terdapat antara unit **p** dan **q**, serta pada unit **v** dan **u**. Gambar 2.8 menunjukkan alur proses normalisasi tersebut

Kompetisi terjadi pada lapisan F2. Setiap unit y_j saling bersaing, dan pemenang berhak untuk melakukan proses belajar untuk setiap unit masukan dari p_i, lapisan F1. Proses belajar terjadi hanya jika bobot vektor dengan arah *top-down* untuk unit pemenang memiliki kemiripan yang mendekati dengan vektor masukannya.



Gambar 2.8 Proses Normalisasi ART-2
Sumber: Laurene Fausett, 1994

Fungsi aktifasi diberikan pada unit-unit masukan dari unit $\boldsymbol{x_i}$ $(f(x_i))$ dan unit $\boldsymbol{q_i}$ $(bf(q_i))$. Fungsi aktifasi ini berperan untuk menekan beberapa komponen dari vektor-vektor aktifasi yang berada pada tingkat dibawah nilai θ yang ditentukan. Sedangkan, pada jalur koneksi dari **u** ke **w** dan dari **q** ke **v** memiliki bobot-bobot tetap *a* dan *b*.

2.5.3 Algoritma Pelatihan ART-2

Terdapat dua teknik algoritma pelatihan dalam ART-2 (Tanaka dan Weitzefeld, 2002). Teknik yang pertama yaitu *fast learning*, pada teknik ini iterasi-iterasi dalam perubahan bobot diikuti dan dilakukan dengan memperbaharui hasil aktifasi-aktifasi F1 sehingga kondisi kesetimbangan tercapai. Teknik yang kedua yaitu *slow learning*, pada teknik ini hanya dilakukan satu iterasi dalam perubahan bobot yang dilakukan tetapi dibutuhkan sejumlah besar percobaan-percobaan belajar (*epoch*) dengan tujuan agar jaringan dapat menjadi stabil.

Teknik pelatihan *fast learning* umumnya lebih digunakan pada ART-1 sedangkan teknik *slow learning* lebih sesuai digunakan dalam ART-2 (Fauset, 1994).

Parameter pada ART-2 terdiri dari parameter-parameter ***a, b, o, c, d, e, α*** dan ***ρ***. Penjelasan lebih lanjut dari masing-masing parameter adalah sebagai berikut :

a, b Merupakan bobot tetap dalam lapisan F1. Nilai a dan b tidak boleh nol karena menghasilkan ketidakstabilan dalam jaringan, $(a, b > 0)$

c Merupakan bobot tetap yang digunakan untuk menguji kondisi reset. Nilai c yang kecil memberikan range efektif yang lebih lebar untuk parameter *vigilance (ρ)*. Nilai c antara 0 hingga 1, $(0 < c \leq 1)$

d Merupakan aktifasi unit pemenang F2. Nilai d harus memenuhi persamaan :

$$0 < d < 1 \quad \text{dan} \quad \frac{cd}{(1-d)} \leq 1$$

e merupakan parameter yang digunakan untuk mencegah pembagian oleh nol ketika normalisasi vektor adalah nol.

α *Learning rate* merupakan parameter untuk menentukan kecepatan belajar. Nilai α adalah :

$$0 < \alpha \leq 1/\sqrt{n}$$

ρ *Vigilance* parameter. Parameter ini menentukan seberapa banyak sel memori yang akan dibentuk. Jika nilai ρ besar maka memori akan semakin detail, dan sebaliknya jika nilai ρ kecil menghasilkan memori secara umum. Nilai ρ adalah $0 < \rho < 1$.

θ Merupakan parameter penekanan derau. Nilai θ adalah :

$$0 < \theta < 1/\sqrt{n}$$

$\|\boldsymbol{x}\|$ Merupakan nilai normalisasi vektor. Nilai $\|x\|$ adalah

$$\|x\| = \sqrt{\sum {x_i}^2}$$

n Jumlah masukan unit-unit pada lapisan F1

Fungsi aktifasi $f(x)$ yang digunakan pada saat pembaruan F1 adalah

$$f(x) = \begin{cases} 0, & 0 \leq x < \theta \\ x, & x \geq \theta \end{cases}$$

Inisialisasi bobot yang digunakan pada ART-2

$t_{ji}(0)$ Inisialisasi bobot *top-down* (nilai bobot harus kecil agar tidak terjadi reset saat pola pertama dipelajari oleh unit *cluster*).

$b_{ij}(0)$ Inisialisasi bobot *bottom-up* dimana harus memenuhi persamaan :

$$b_{ij}(0) \leq \frac{1}{(1-d)\sqrt{n}}$$

Untuk mencegah kemungkinan terjadinya pemilihan unit pemenang baru pada proses resonansi disaat nilai bobot berubah. Nilai bobot $b_{ij}(0)$ yang besar juga membuat jaringan membentuk lebih banyak cluster

Data diproses dengan algortima sebagai berikut:

1. Inisialisasi parameter :
   a, b, ɵ, c, d, e, α, ρ.
2. Melakukan langkah 3 sampai 13 sebanyak N kali *epoch*.
3. Untuk setiap vektor masukan s_i dilakukan langkah 4 sampai 12.
4. Memperbarui aktifasi-aktifasi unit F1:

$$u_i = 0 \qquad w_i = s_i \qquad \textbf{(2.13)}$$
$$p_i = 0 \qquad x_i = \frac{s_i}{e + \|s\|}$$
$$q_i = 0 \qquad v_i = f(x_i)$$

   Memperbarui kembali aktifitas-aktifitas F1 :

$$u_i = \frac{v_i}{e + \|v\|}$$
$$w_i = s_i + a u_i \qquad \textbf{(2.14)}$$
$$p_i = u_i$$
$$x_i = \frac{w_i}{e + \|w\|}$$
$$q_i = \frac{p_i}{e + \|p\|}$$
$$v_i = f(x_i) + b f(q_i)$$

5. Menghitung sinyal-sinyal untuk unit F2:

$$y_j = \sum_i b_{ij}\, p_i \qquad \textbf{(2.15)}$$

6. Apabila terjadi reset, melakukan langkah 7 dan 8

7. Mencari unit F2 y_J dengan sinyal terbesar (Menentukan J sehingga $y_J \geq y_j$ untuk $j = 1, \ldots, m$)
8. Memeriksa kondisi reset :

$$u_i = \frac{v_i}{e + \|v\|}$$
$$p_i = u_i + dt_{Ji} \quad \textbf{(2.16)}$$
$$r_i = \frac{u_i + cp_i}{e + \|u\| + c\|p\|}$$

Jika $\|r\| < p - e$ maka

$y_J = -1$ (menghalangi J)

Terjadi reset, kembali ke langkah 6

Jika $\|r\| \geq p - e$ maka

$$w_i = s_i + au_i \quad \textbf{(2.17)}$$
$$x_i = \frac{w_i}{e + \|w\|}$$
$$q_i = \frac{p_i}{e + \|p\|}$$
$$v_i = f(x_i) + bf(q_i)$$

Tidak terjadi reset, melanjutkan ke langkah 9

9. Melakukan langkah 10 sampai 12 sebanyak N kali iterasi pembelajaran
10. Memperbarui nilai bobot untuk unit pemenang J :

$$t_{Ji} = \alpha d u_i + \{1 + \alpha d(d-1)\} t_{Ji} \quad \textbf{(2.18)}$$
$$b_{iJ} = \alpha d u_i + \{1 + \alpha d(d-1)\} b_{iJ}$$

11. Memperbarui unit aktifasi F1 :

$$u_i = \frac{v_i}{e + \|v\|}$$
$$w_i = s_i + au_i \quad \textbf{(2.19)}$$
$$p_i = u_i + dt_{Ji}$$
$$x_i = \frac{w_i}{e + \|w\|}$$
$$q_i = \frac{p_i}{e + \|p\|}$$
$$v_i = f(x_i) + bf(q_i)$$

12. Menguji kondisi berhenti untuk pembaruan bobot.
13. Menguji kondisi berhenti untuk *epoch*.

2.6 Jarak *Euclidean*

Jarak *euclidean* digunakan dalam proses pengenalan untuk menghitung jarak antara fitur uji dengan fitur latih yang berada dalam *cluster* yang sama.

$$d = \sqrt{\sum_{i=1}^{n} (b_i - a_i)^2} \qquad \textbf{(2.20)}$$

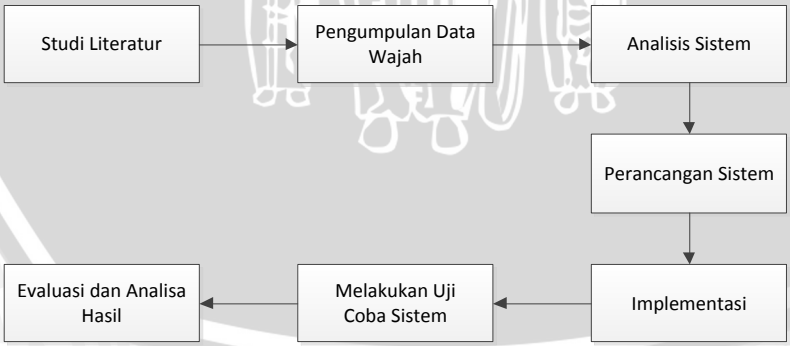
d = jarak *euclidean*
n = banyaknya data
a_i = data training ke-i

BAB III
METODOLOGI DAN PERANCANGAN

Pada bab ini dibahas tentang metodologi dan perancangan yang digunakan dan seluruh tahapan yang digunakan dalam penelitian ini. Tahapan-tahapan yang harus dilakukan dalam penelitian ini dapat diidentifikasikan sebagai berikut :

1. Mencari dan mempelajari literatur-literatur yang berkaitan dengan permasalahan pengenalan wajah melalui sebuah citra *grayscale* menggunakan metode PCA dan ART-2 yang berasal dari buku dan sumber lain dari internet.
2. Mempersiapkan data citra wajah yaitu berupa data latih dan data uji. Citra wajah yang digunakan berbentuk citra *grayscale*, berukuran 92 x 112 piksel dengan format JPG .
3. Mengimplementasikan metode ekstraksi ciri dengan metode PCA dan metode klasifikasi dengan metode ART-2 menggunakan data latih yang telah disiapkan sebagai proses pembelajaran yang akan digunakan dalam pengenalan ke sebuah sistem perangkat lunak.
4. Melakukan uji coba terhadap hasil pengenalan yang dilakukan oleh sistem. Hasil pengenalan merupakan informasi pengenalan dari individu yang diuji.
5. Mengevaluasi dan menganalisis hasil pengenalan yang dihasilkan oleh sistem berupa tingkat akurasi dari model pengklasifikasian yang dihasilkan dari data *training*.

Tahapan-tahapan yang telah dijelaskan dapat digambarkan dengan sebuah diagram alur yang ditunjukkan pada Gambar 3.1.



Gambar 3.1 Diagram Alur Penelitian

3.1 Deskripsi Umum

Pada sistem ini akan diimplementasikan mengenai proses yang digunakan untuk mengenali suatu wajah individu dengan metode PCA dan jaringan syaraf tiruan ART-2. Metode PCA berfungsi untuk mereduksi dimensi data dan mencari fitur ciri wajah, sedangkan jaringan syaraf tiruan berfungsi sebagai pengklasifikasi wajah. Model pembelajaran jaringan yang akan digunakan yaitu *slow learning*. Sistem yang dibangun ini bertujuan agar dapat digunakan untuk mengenali wajah seseorang dari masukan berupa citra wajah.

3.2 Deskripsi Data

Sistem yang dibangun ini menggunakan citra wajah yang berasal dari *AT&T Laboratories Cambridge Face Database*. *Database AT&T* diperoleh dari http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html. *Database* ini terdiri dari 400 citra dari 40 individu. Terdapat 10 citra wajah dalam setiap individunya. Format citra pada *database* ini yaitu citra *grayscale* berukuran 92x112 piksel dengan format PGM. Setiap citra memiliki variasi pencahayaan yang berbeda-beda, ekspresi yang berbeda (mata terbuka/tertutup, senyum/tidak senyum, menengok kesamping) dan aksesoris berupa kacamata dengan latar berwarna gelap

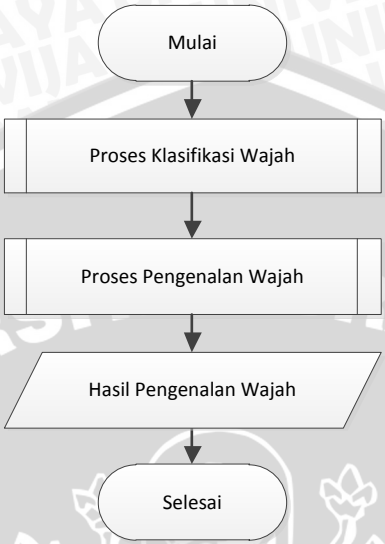
Citra *AT&T* yang nantinya akan digunakan adalah 320 citra, yaitu 200 untuk citra latih dan 120 untuk citra uji. Sistem akan menggunakan citra yang telah disesuaikan format data citranya menjadi format citra JPG.

3.3 Perancangan Proses

Pada perancangan sistem pengenalan wajah ini secara umum terdapat dua proses utama, yaitu proses klasifikasi wajah dan proses pengenalan wajah. Proses klasifikasi wajah ini adalah proses memasukkan citra-citra wajah kedalam kelompok/*cluster* yang akan menjadi rujukan sesuai dengan ciri/fiturnya.

Proses selanjutnya adalah proses pengenalan wajah. Pada proses ini citra uji akan dikelompokkan kedalam *cluster* hasil pelatihan. Kemudian akan dibandingkan tingkat kemiripan vektor ciri antara citra uji dengan citra latih dalam tiap-tiap *cluster*.

Perancangan sistem secara umum dapat dilihat pada Gambar 3.2.



Gambar 3.2 Alur Proses Sistem Pengenalan Wajah

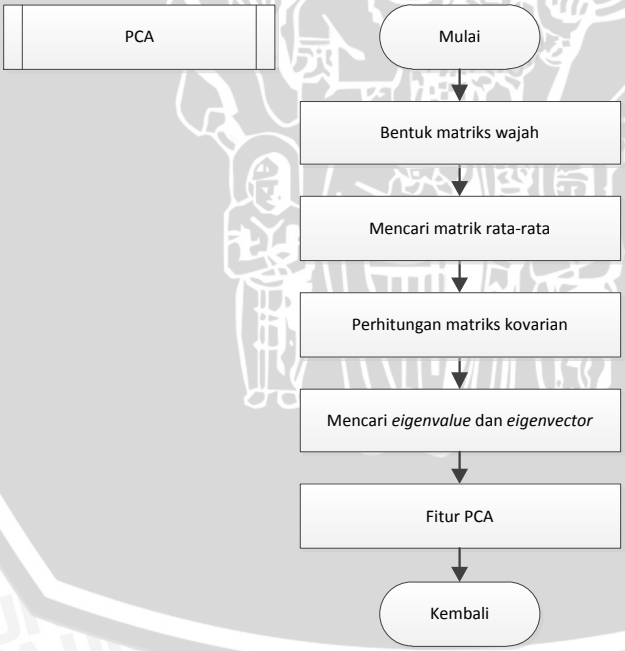
3.3.1 Proses Klasifikasi Wajah

Tahap yang pertama adalah proses klasifikasi wajah. Karena tanpa klasifikasi wajah maka tidak akan bisa dilakukan pengenalan sebab tidak ada acuan untuk pencocokannya. Proses klasifikasi wajah dimulai dengan memasukkan citra latih yang kemudian akan dicari nilai fitur wajahnya menggunakan PCA. Setelah didapat nilai fitur wajahnya, data dilatihkan dalam ART-2 yang menghasilkan *cluster-cluster* sebagai klasifikasi wajah. *Flowchart* proses klasifikasi wajah dapat dilihat pada Gambar 3.3

3.3.1.1 Proses PCA

Proses PCA dimulai dengan membentuk matrik wajah yang berisi nilai piksel citra. Kemudian mencari matrik nilai rata-rata dan dihitung matrik kovariannya. Dari matrik kovarian dicari *eigenvalue* dan *eigenvector* yang digunakan untuk menghitung nilai fitur PCA. *Flowchart* proses PCA dapat dilihat pada Gambar 3.4

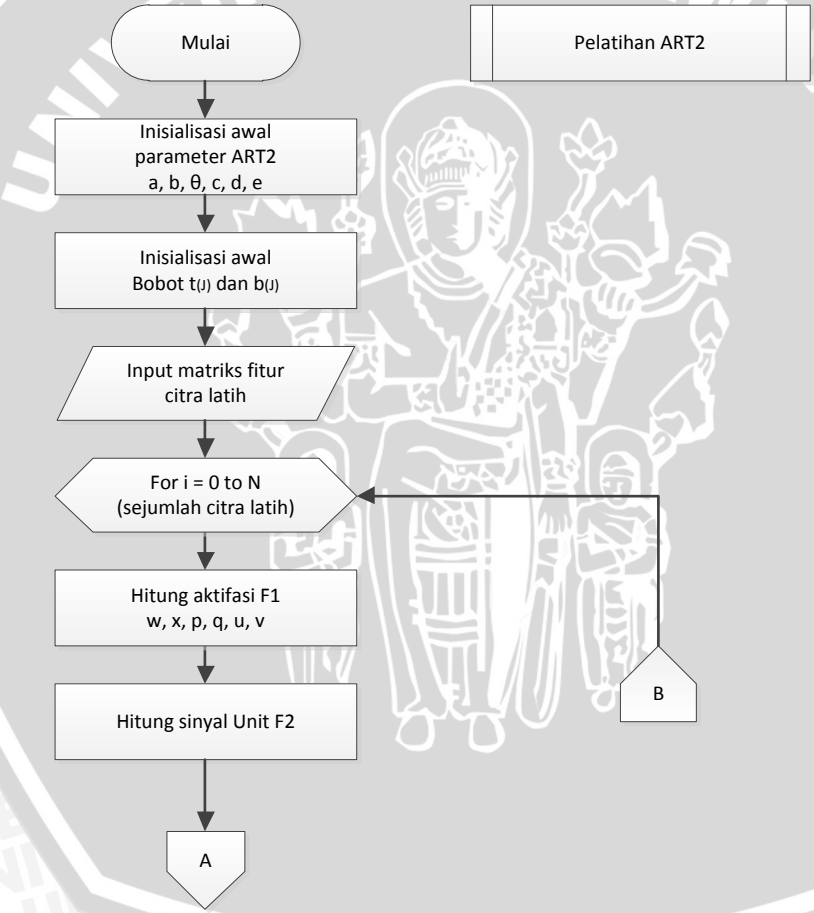
Gambar 3.3 *Flowchart* Proses Klasifikasi Wajah

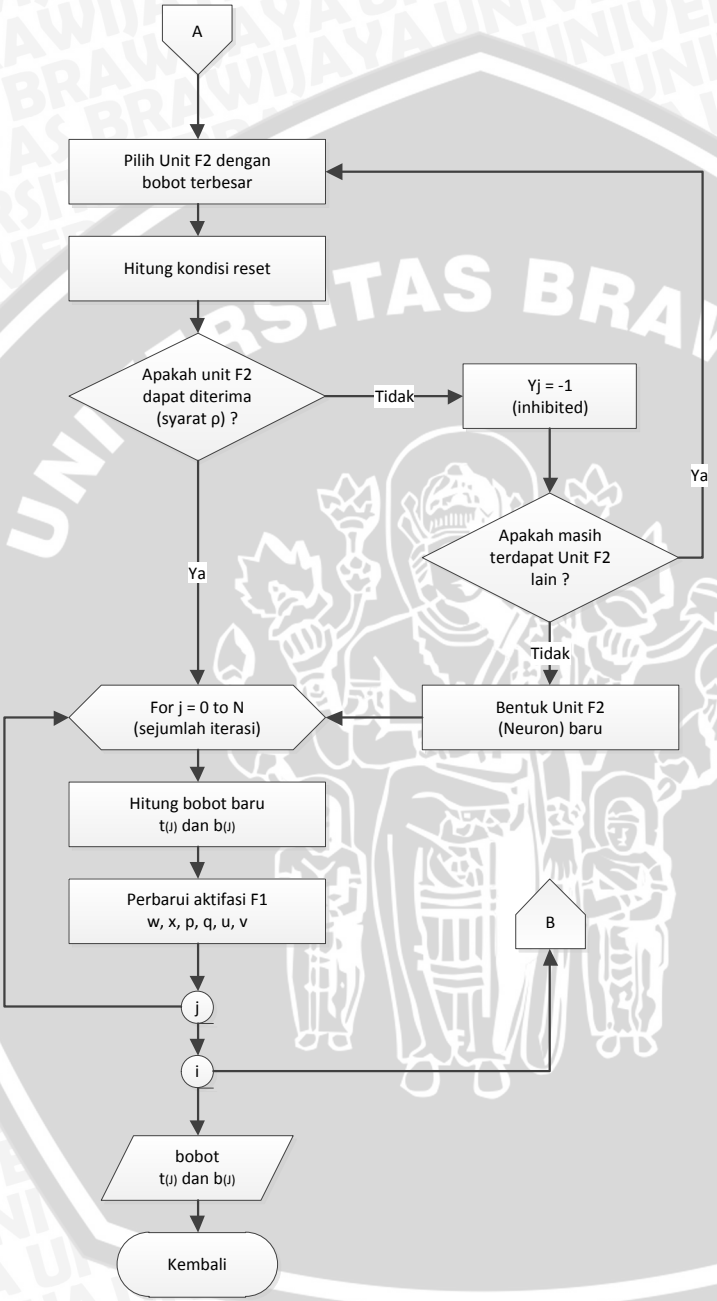


Gambar 3.4 *Flowchart* Proses PCA

3.3.1.2 Proses Pelatihan ART-2

Pada proses pelatihan ART-2 dilakukan pengklasifikasian citra-citra wajah sesuai dengan fiturnya. Proses pelatihan dimulai dengan menginisialisasi parameter ART-2 dan bobot awal. Kemudian untuk setiap citra input dilakukan aktifasi lapisan F1 dan lapisan F2. Unit pemenang F2 akan diperiksa nilai resetnya apakah memenuhi syarat. Apabila nilai reset memenuhi syarat maka bobot pemenang akan diperbarui dan jaringan dapat mempelajari inputan. *Flowchart* proses pelatihan ART-2 dapat dilihat pada Gambar 3.5

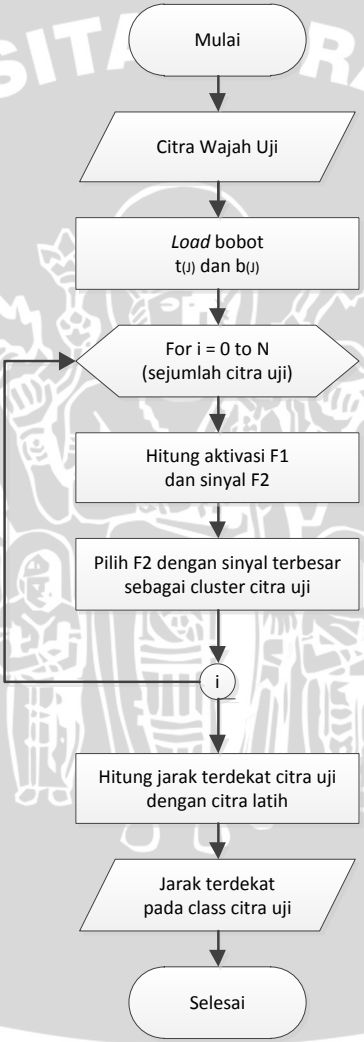




Gambar 3.5 *Flowchart* Proses Pelatihan ART-2

3.3.2 Proses Pengenalan Wajah

Tahap berikutnya adalah proses pengenalan wajah. Proses pengenalan wajah dimulai dengan menghitung fitur wajah uji. Kemudian dilakukan proses pengelompokkan citra uji kedalam *cluster-cluster* hasil pelatihan. Setelah itu dilakukan pencocokan dengan menghitung ciri fitur antara citra input dan masing-masing citra pada *cluster*.



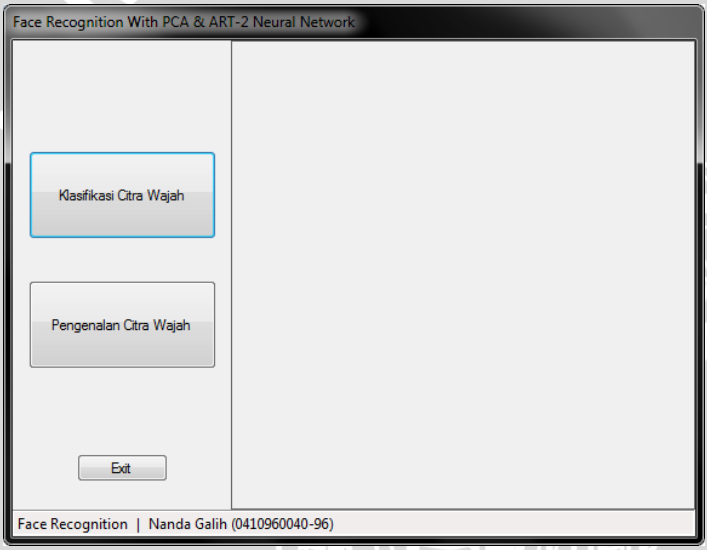
Gambar 3.6 *Flowchart* Proses Pengenalan Wajah

3.4 Perancangan Antarmuka

Pada subbab ini akan dijelaskan mengenai perancangan antar muka yang digunakan dalam aplikasi pengenalan wajah dengan metode PCA dan ART-2. Program terdiri dari tiga *form* yaitu *form* utama, *form training* dan *form* pengenalan wajah.

3.4.1 *Form* Utama

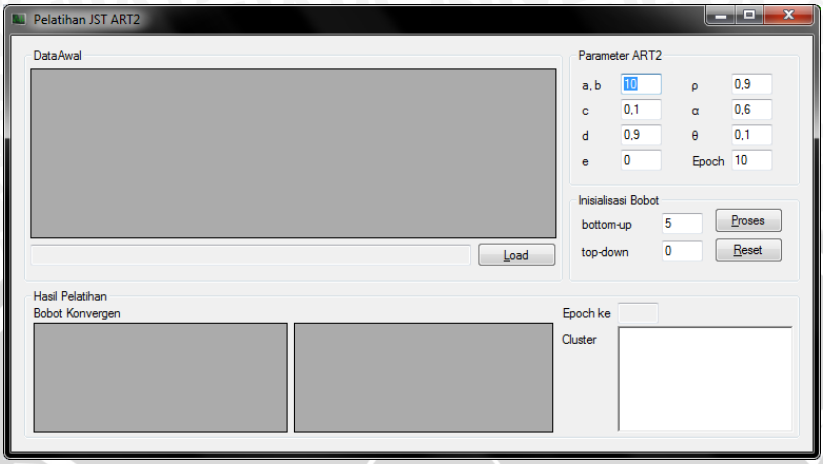
Pada *form* utama terdiri dari dua bagian yaitu bagian judul dan menu navigasi. Pada menu navigasi berisi *option* yang dapat dipilih oleh user untuk menampilkan *form training*, *form* pengenalan dan *exit*. Untuk rancangan *form* utama dapat dilihat pada Gambar 3.7



Gambar 3.7 *Form* Utama

3.4.2 *Form* Pelatihan

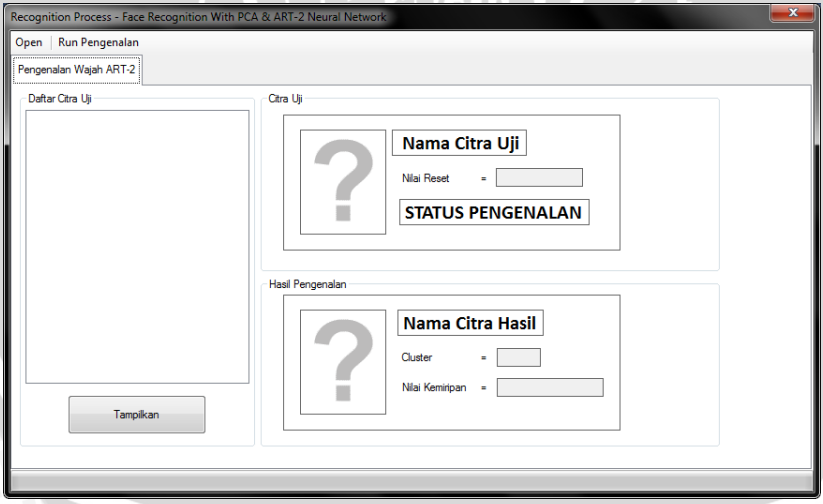
Pada *form* ini akan digunakan untuk melakukan pelatihan data latih menggunakan metode PCA. Hasil data PCA kemudian akan digunakan dalam pelatihan ART-2. Input citra latih merupakan citra berukuran 92 x 112 piksel berbentuk *grayscale* dan berformat JPG. Untuk rancangan *form* pelatihan dapat dilihat pada Gambar 3.8.



Gambar 3.8 *Form* Pelatihan

3.4.3 *Form* Pengenalan Wajah

Form digunakan untuk menguji data uji. Pengenalan ini berdasarkan pada data latih. Hasil dari menu dalam *form* ini akan menunjukkan informasi dari data yang diuji. Untuk rancangan *form* pengenalan wajah dapat dilihat pada Gambar 3.9.



Gambar 3.9 *Form* Pengenalan Wajah

3.5 Perancangan Uji Coba

Pengujian sistem ini bertujuan untuk menguji kelayakan sistem jika diterapkan secara langsung untuk pengenalan. Pengujian dilakukan dengan memasukkan citra wajah yang akan dicari hasil identifikasinya. Pengujian sistem menggunakan acuan Tabel 3.1 dan Tabel 3.2

Tabel 3.1 Pengujian Identifikasi Wajah

Citra Uji	Status Pengenalan	Cluster	Hasil Identifikasi	Jarak Euclidean

Pengujian dilakukan untuk sejumlah citra latih yang berbeda, yaitu 80 citra latih (2 set), 120 citra latih (3 set), 160 citra latih (4 set) dan 200 citra latih (5 set). Sedangkan citra uji yang digunakan pada setiap pengujian sebanyak 120 citra uji (3 set). Citra uji dan citra latih yang digunakan berasal dari *database* yang sama.

Tabel 3.2 Tingkat Keakuratan Identifikasi

Jumlah citra latih	Jumlah yang teridentifikasi	Jumlah yang tidak teridentifikasi	Jumlah yang salah identifikasi	Persentase Keakuratan
80 (2 set)				
120 (3 set)				
160 (4 set)				
200 (5 set)				
Persentase Rata-Rata				

Pengujian data pada sistem pengenalan ini dilakukan dengan menghitung tingkat akurasi pada setiap pengujian. Tingkat akurasi diperoleh dari persentase antara gambar yang benar dikenali sebagai gambar input dan banyaknya gambar input yang dikenali.

$$\text{Tingkat Akurasi} = \frac{\text{Jumlah benar identifikasi}}{\text{Jumlah yang teridentifkasi}} \times 100\% \quad \textbf{(3.1)}$$

3.6 Contoh Perhitungan Manual

Pada perhitungan manual menunjukkan proses keseluruhan secara umum. Data citra untuk perhitungan manual berbentuk *grayscale*. Tahapan perhitungan dimulai dari reduksi dimensi sampai hasil pengenalan.

3.6.1 Perhitungan Data Latih

Dimisalkan menggunakan lima citra wajah yang diambil dari *database* AT&T sebagai data latih. Citra wajah tersebut masing-masing berukuran 2x5 piksel (panjang dimensi = 10), sehingga pembentukan matriks wajah sebagai berikut

A1	A2	B1	B2	C1
139	101	123	114	174
77	71	160	112	72
110	113	81	165	108
143	115	116	175	135
52	108	155	117	71
87	150	81	164	102
146	163	90	67	153
77	158	174	83	85
163	145	67	64	118
148	57	70	57	160

3.6.1.1 Reduksi Dimensi dengan PCA

Pada PCA nilai piksel dari citra diubah ke dalam ruang fitur. Langkah pertama adalah mencari rata-rata piksel dari setiap citra data *training*,

$$\overline{u}_1 = \frac{139 + 77 + 110 + 143 + \ \ldots \ + 148}{10} = 114{,}2$$

...

$$\overline{u}_5 = \frac{174 + 72 + 108 + 135 + \ \ldots \ + 160}{10} = 117{,}8$$

Hasil dari perhitungan rata-rata citra dapat dilihat sebagai berikut

$\overline{u}_1$	$\overline{u}_2$	$\overline{u}_3$	$\overline{u}_4$	$\overline{u}_5$
114,2	118,1	111,7	111,8	117,8

Selanjutnya mencari matriks rata-rata *Y* dengan cara mengurangi matriks wajah dengan rata-rata citra

$$Y = X - \overline{u}$$
$$Y_{1,1} = 139 - 114{,}2 = 24{,}8$$
$$\dots$$
$$Y_{5,10} = 160 - 117{,}8 = 42{,}2$$

Hasil dari perhitungan *Y* dapat dilihat sebagai berikut

Y_1	Y_2	Y_3	Y_4	Y_5
24,8	-17,1	11,3	2,2	56,2
-37,2	-47,1	48,3	0,2	-45,8
-4,2	-5,1	-30,7	53,2	-9,8
28,8	-3,1	4,3	63,2	17,2
-62,2	-10,1	43,3	5,2	-46,8
-27,2	31,9	-30,7	52,2	-15,8
31,8	44,9	-21,7	-44,8	35,2
-37,2	39,9	62,3	-28,8	-32,8
48,8	26,9	-44,7	-47,8	0,2
33,8	-61,1	-41,7	-54,8	42,2

Kemudian dari matrik rata-rata *Y* dicari matrik kovarian C

$$C = Y^T * Y$$
$$C_{(1,1)} = (24{,}8*24{,}8) + ((-37{,}2)*(-37{,}2)) + \dots + (33{,}8*33{,}8)$$
$$= 13373{,}6$$
$$\dots$$
$$C_{(5,5)} = (56{,}2*56{,}2) + ((-45{,}8)*(-45{,}8)) + \dots + (42{,}2*42{,}2)$$
$$= 12183{,}6$$

Hasil dari perhitungan matriks kovarian C dapat dilihat sebagai berikut

13373,6	211,8	-9720,4	-4637,6	10750,4
211,8	11730,9	-884,7	0,2	-1139,8
-9720,4	-884,7	14328,1	895,4	-7319,6
-4637,6	0,2	895,4	17705,6	-3342,4
10750,4	-1139,8	-7319,6	-3342,4	12183,6

Langkah selanjutnya yaitu mencari nilai *eigenvalue* dan *eigenvector* dari matriks kovarian C. Nilai *eigenvalue* yang didapat dapat dilihat sebagai berikut

33614,24	0	0	0	0
0	16555,29	0	0	0
0	0	12032,15	0	0
0	0	0	5567,165	0
0	0	0	0	1,55E+03

Eigenvector yang dipilih apabila nilai *eigenvalue* yang bersesuaian lebih besar dari nol. *Eigenvector* yang terpilih dapat dilihat sebagai berikut

0,593906	-0,08358	-0,23062	0,062358	-0,76369
0,578231	0,190791	0,732949	0,20442	0,224156
0,163392	0,078541	0,122231	-0,9758	0,001882
-0,09926	-0,92292	0,358402	-0,04618	-0,08818
0,525713	-0,3141	-0,51595	-0,00073	0,598964

Kemudian mencari nilai matrik V_{exp} dengan cara sebagai berikut

$$V_{exp} = Y * v$$

$$V_{exp_{[1,1]}} = (24{,}8*0{,}593906) + ((-17{,}1)*(-0{,}08358)) + (11{,}3*(-0{,}23062)) + (2{,}2*0{,}062358) + (56{,}2*(-0{,}76369)) = 36{,}01412214$$

...

$$V_{exp_{[5,10]}} = (33{,}8*(-0{,}76369)) + ((-61{,}1)*0{,}224156) + ((-41{,}7)*0{,}001882) + ((-54{,}8)*(-0{,}08818)) + (42{,}2*0{,}598964) = -9{,}478202132$$

Matrik V_{exp} yang terbentuk dapat dilihat sebagai berikut

V_{exp_1}	V_{exp_2}	V_{exp_3}	V_{exp_4}	V_{exp_5}
36,01412	-24,1307	-45,0797	-13,1182	10,71658
-65,5337	12,11777	3,663234	-59,055	-9,50791
-20,8924	-49,0545	17,60138	26,20297	-8,55471
18,78332	-66,3922	5,388231	-5,96497	-17,952
-60,8258	16,57343	38,24458	-48,4016	16,8287
-16,2127	-37,2653	52,76201	32,38285	13,79822
64,25533	34,49482	-11,2946	34,37966	10,7727
-3,22715	52,4976	52,03997	-53,6019	20,36383
42,08341	41,59543	-14,2363	54,36767	-26,9872
5,555434	19,56356	-99,0888	32,80857	-9,4782

Langkah selanjutnya yaitu mendapatkan fitur PCA dengan cara sebagai berikut

$f_{PCA} = V_{exp}{}^{T} * Y$

$f_{PCA[1,1]}$ = (36,01412*24,8) + ((-65,5337)*(-37,2)) + ((-20,8924) * (-4,2)) + ... + (5,555434*33,8)

= 12588,8712

...

$f_{PCA[5,5]}$ = (10,71658*56,2) + ((-9,50791)*(-45,8)) + ((-8,55471) * (-9,8)) + ... + ((-9,4782)*42,2)

= -886,91119

Hasil perhitungan f_{PCA} dapat dilihat sebagai berikut

f_{PCA_1}	f_{PCA_2}	f_{PCA_3}	f_{PCA_4}	f_{PCA_5}
12588,87	6165,182	-7880,36	-6122,56	11266,55
-937,411	2508,783	3241,718	-14833,1	-2433
-11325,9	9029,326	7442,086	9249,344	-11693,4
10568,77	3275,357	-14804,4	-1978,12	7725,336
-3335,98	1783,415	2788,849	-19,9247	-886,911

3.6.1.2 Klasifikasi dengan ART-2

Langkah pertama dalam proses pelatihan adalah menentukan parameter-parameter ART-2. Dimisalkan parameter yang digunakan sebagai berikut :

a, b	= 10	c	= 0,1
d	= 0,9	e	= 0
α	= 0,4	ρ	= 0,95
θ	= 0,1	*epoch*	= 5

Dalam pelatihan ART-2, nilai input yang digunakan adalah nilai fitur PCA data latih dan jenis pembelajaran yang digunakan adalah *slow learning*.

Inisialisasi bobot awal *bottom-up* ($\mathbf{b}_J$) dan *top-down* ($\mathbf{t}_J$)

$b_{[i,J]}$	i_1	i_2	i_3	i_4	i_5
J_1	4,47213595	4,47213595	4,47213595	4,47213595	4,47213595

$t_{[J,i]}$	i_1	i_2	i_3	i_4	i_5
J_1	0	0	0	0	0

Input Pertama (Citra Latih A1)

1. Menghitung aktifasi pertama pada unit F1

$$u_i = 0 \qquad w_i = s_i$$
$$p_i = 0 \qquad x_i = \frac{s_i}{e + \|s\|}$$
$$q_i = 0 \qquad v_i = f(x_i)$$

$x_1 = (12588{,}8712) / (0 + 20259{,}87775) = 0{,}621369554$
...
$x_5 = (-3335{,}981629) / (0 + 20259{,}87775) = -0{,}16465951$

Hasil aktifasi pertama unit F1 dapat dilihat sebagai berikut

u	0	0	0	0	0
w	12588,8712	-937,410537	-11325,922	10568,77224	-3335,98163
p	0	0	0	0	0

x	0,621369554	-0,046269309	-0,5590321	0,521660218	-0,16465951
q	0	0	0	0	0
v	0,621369554	0	0	0,521660218	0

2. Menghitung aktifasi kedua pada unit F1

$$u_i = \frac{v_i}{e + \|v\|} \qquad w_i = s_i + au_i$$

$$x_i = \frac{w_i}{e + \|w\|} \qquad p_i = u_i$$

$$q_i = \frac{p_i}{e + \|p\|}$$

$$v_i = f(x_i) + bf(q_i)$$

Hasil aktifasi kedua unit F1 dapat dilihat sebagai berikut

u	0,765881002	0	0	0,642982341	0
w	12596,53001	-937,410537	-11325,922	10575,20206	-3335,98163
p	0,765881002	0	0	0,642982341	0
x	0,621498675	-0,046250785	-0,5588083	0,52176862	-0,1645936
q	0,765881002	0	0	0,642982341	0
v	8,280308696	0	0	6,951592029	0

3. Menghitung sinyal F2 untuk *cluster* 1

$$Y_1 = \sum_i b_{i1} * p_i$$

$$= 4{,}47213595 * (0{,}765881002) + 4{,}47213595 * (0) + 4{,}47213595 * (0) + 4{,}47213595 * (0{,}642982341) + 4{,}47213595 * (0)$$

$$= 6{,}300628412$$

4. Dipilih *cluster* Y_J terbesar untuk *J=1,..,m* (*m* adalah banyaknya *cluster*). Nilai m = 1 karena jaringan baru mempelajari satu pola inputan. Nilai J pemenang adalah J=1
5. Menghitung kondisi reset
   Mencari nilai unit *u* dan *p*

u	0,765881002	0	0	0,642982341	0

p	0,765881002	0	0	0,642982341	0

Menghitung nilai reset

$$r_i = \frac{u_i + cp_i}{e + \|u\| + c\|p\|}$$

$r_1 = (0{,}765881002 + 0{,}1 * 0)/(0 + 1 + 0{,}1 * 1)$
$= 0{,}765881002$
...
$r_5 = (0 + 0{,}1 * 0)/(0 + 1 + 0{,}1 * 1)$
$= 0$

Nilai reset unit J=1 dapat dilihat sebagai berikut

r	0,765881002	0	0	0,642982341	0

$\|r\| = \sqrt{(0{,}765881002)^2 + (0)^2 + (0)^2 + (0{,}642982341)^2 + (0)^2}$
$= 1$

Unit pemenang J=1 diterima karena $\|r\| > \rho - e$, sehingga *cluster* 1 dapat mempelajari pola inputan A1. Menghitung kembali unit *w*, *x*, *q* dan *v* yang dapat dilihat sebagai berikut

w	12596,53001	-937,410537	-11325,922	10575,20206	-3335,98163
x	0,621498675	-0,046250785	-0,5588083	0,52176862	-0,1645936
q	0,765881002	0	0	0,642982341	0
v	8,280308696	0	0	6,951592029	0

6. Hitung bobot terbaru untuk unit pemenang J=1

Bobot *top-down* t_{ji}

$t_{ji} = \alpha d u_i + \{1 + \alpha d(d-1)\} t_{ji}$

$t_{1,1} = (0{,}4 * 0{,}9 * 0{,}765881002) + (1 + 0{,}4 * 0{,}9 * (0{,}9 - 1) * 0)$
$= 1{,}275717161$
...
$t_{1,5} = (0{,}4 * 0{,}9 * 0) + (1 + 0{,}4 * 0{,}9 * (0{,}9 - 1) * 0)$
$= 1$

Bobot *bottom-up* b_{ij}

$b_{ij} = \alpha d u_i + \{1 + \alpha d(d-1)\} b_{ij}$

$b_{1,1} = (0{,}4 * 0{,}9 * 0{,}765881002) + (1 + 0{,}4 * 0{,}9 * (0{,}9 - 1) * 4{,}47213595)$

$= 1{,}114720266$

...

$b_{1,5} = (0{,}4 * 0{,}9 * 0) + (1 + 0{,}4 * 0{,}9 * (0{,}9 - 1) * 4{,}47213595)$

$= 0{,}839003106$

Nilai bobot t_{1i} dan b_{i1} dapat dilihat sebagai berikut

t_{1i}	1,275717161	1	1	1,231473643	1
b_{i1}	1,114720266	0,839003106	0,839003106	1,070476748	0,839003106

Pola pelatihan yang terbentuk setelah pelatihan inputan pertama (A1) yaitu

Cluster (J)	Input yang dipelajari
1	A1

Input Kedua (Citra Latih A2)

Untuk data input yang kedua yaitu A2 dilakukan *training* menggunakan nilai bobot $\mathbf{t}_{ji}$ dan $\mathbf{bi}_{j}$ dari hasil training sebelumnya

t_{1i}	1,275717161	1	1	1,231473643	1
b_{i1}	1,114720266	0,839003106	0,839003106	1,070476748	0,839003106

1. Menghitung aktifasi pertama unit F1

u	0	0	0	0	0
w	6165,181529	2508,782962	9029,325819	3275,356898	1783,415103
p	0	0	0	0	0
x	0,521536121	0,212227479	0,763824964	0,27707488	0,150865857
q	0	0	0	0	0
v	0,521536121	0,212227479	0,763824964	0,27707488	0,150865857

2. Menghitung aktifasi kedua unit F1

u	0,521536121	0,212227479	0,763824964	0,27707488	0,150865857

w	6170,39689	2510,905237	9036,964068	3278,127647	1784,923761
p	0,521536121	0,212227479	0,763824964	0,27707488	0,150865857
x	0,521536121	0,212227479	0,763824964	0,27707488	0,150865857
q	0,521536121	0,212227479	0,763824964	0,27707488	0,150865857
v	5,736897329	2,33450227	8,402074607	3,047823678	1,659524426

3. Menghitung sinyal unit F2
   Y_1 = 1,823457054
4. Memilih *cluster* pemenang, J = 1
5. Menghitung kondisi reset
   Mencari nilai *u*, *p* dan reset yang dapat dilihat seperti berikut

u	0,521536121	0,212227479	0,763824964	0,27707488	0,150865857
p	1,669681565	1,112227479	1,663824964	1,385401158	1,050865857
r	0,524247711	0,246284659	0,70828773	0,316461665	0,194889831

Didapatkan nilai $||r||$= 0,987569408 sehingga J=1 diterima. Dihitung kembali unit *w*, *x*, *q* dan *v*

w	6170,39689	2510,905237	9036,964068	3278,127647	1784,923761
x	0,521536121	0,212227479	0,763824964	0,27707488	0,150865857
q	0,532902128	0,354982892	0,531032912	0,442170077	0,335398476
v	5,850557403	3,762056398	6,074154088	4,698775646	3,504850612

6. Hitung bobot terbaru untuk unit pemenang J=1
   Nilai bobot t_{1i} dan b_{i1} dapat dilihat seperti berikut

t_{1i}	1,141827186	1,040401892	1,238976987	1,055413906	1,018311708
b_{i1}	1,147623074	1,046197781	1,244772875	1,061209794	1,024107597

Pola pelatihan yang terbentuk setelah pelatihan inputan kedua (A2) yaitu

Cluster (J)	Input yang dipelajari
1	A1, A2

Input Ketiga (Citra Latih B1)

Bobot t_{ji} dan Bobot b_{ij}

t_{1K}	1,141827186	1,040401892	1,238976987	1,055413906	1,018311708
b_{K1}	1,147623074	1,046197781	1,244772875	1,061209794	1,024107597

1. Menghitung aktifasi pertama unit F1

u	0	0	0	0	0
w	-7880,36275	3241,718002	7442,085951	-14804,4058	2788,848786
p	0	0	0	0	0
x	-0,41828063	0,17206668	0,395017401	-0,78580091	0,1480289
q	0	0	0	0	0
v	0	0,17206668	0,395017401	0	0,1480289

2. Menghitung aktifasi kedua unit F1

u	0	0,377682432	0,867054173	0	0,324920055
w	-7880,36275	3245,494826	7450,756492	-14804,4058	2792,097987
p	0	0,377682432	0,867054173	0	0,324920055
x	-0,41817946	0,172225482	0,395381968	-0,78561085	0,148165518
q	0	0,377682432	0,867054173	0	0,324920055
v	0	3,949049801	9,065923695	0	3,397366069

3. Menghitung sinyal unit F2
   Y_1 = 1,807169134
4. Memilih *cluster* pemenang, J = 1
5. Menghitung kondisi reset
   Mencari nilai *u*, *p* dan reset

u	0	0,377682432	0,867054173	0	0,324920055
p	1,027644467	1,314044135	1,982133461	0,949872515	1,241400593
r	0,078893517	0,390832169	0,817818843	0,072922869	0,344748917

Didapatkan nilai $\|r\|$= 0,97569011 sehingga J=1 diterima. Dihitung kembali unit *w*, *x*, *q* dan *v*

w	-7880,36275	3245,494826	7450,756492	-14804,4058	2792,097987
x	-0,41817946	0,172225482	0,395381968	-0,78561085	0,148165518
q	0,339636898	0,434292101	0,655095884	0,313933238	0,410283382
v	3,396368984	4,515146489	6,946340811	3,139332378	4,250999333

6. Hitung bobot terbaru untuk unit pemenang J=1
   Nilai bobot t_{1i} dan b_{i1} dapat dilihat seperti berikut

t_{1i}	0,958894221	1,098511207	1,267536331	0,962005099	1,080311998
b_{i1}	0,958685569	1,098302555	1,267327679	0,961796447	1,080103346

Pola pelatihan yang terbentuk setelah pelatihan inputan ketiga (B1) yaitu

Cluster (J)	Input yang dipelajari
1	A1, A2, B1

Input Keempat (Citra Latih B2)

Bobot t_{Ji} dan Bobot b_{iJ}

t_{1i}	0,958894221	1,098511207	1,267536331	0,962005099	1,080311998
b_{i1}	0,958685569	1,098302555	1,267327679	0,961796447	1,080103346

1. Menghitung aktifasi pertama unit F1

u	0	0	0	0	0
w	-6122,55935	-14833,0552	9249,344151	-1978,12073	-19,9246541
p	0	0	0	0	0
x	-0,3286909	-0,79631571	0,496553002	-0,10619583	-0,00106966
q	0	0	0	0	0
v	0	0	0,496553002	0	0

2. Menghitung aktifasi kedua unit F1

u	0	0	1	0	0
w	-6122,55935	-14833,0552	9259,344151	-1978,12073	-19,9246541
p	0	0	1	0	0
x	-0,32860326	-0,79610340	0,496957323	-0,10616752	-0,00106937
q	0	0	1	0	0
v	0	0	10,49695732	0	0

3. Menghitung sinyal unit F2
   Y_1 = 1,267327679
4. Memilih *cluster* pemenang, J = 1
5. Menghitung kondisi reset
   Mencari nilai *u*, *p* dan reset

u	0	0	1	0	0
p	0,863004799	0,988660087	2,140782698	0,865804589	0,972280798
r	0,067272731	0,07706778	0,946395206	0,067490979	0,075790986

Didapatkan nilai $||r||$= 0,957302593 sehingga J=1 diterima. Dihitung kembali unit *w*, *x*, *q* dan *v* yang dapat dilihat seperti berikut

w	-6122,55935	-14833,05515	9259,344151	-1978,12073	-19,9246541
x	-0,32860326	-0,796103403	0,496957323	-0,10616752	-0,00106937
q	0,305115905	0,349541414	0,756875109	0,306105773	0,343750506
v	3,051159054	3,495414136	8,065708409	3,061057731	3,437505057

6. Hitung bobot terbaru untuk unit pemenang J=1
   Nilai bobot t_{1i} dan b_{i1} dapat dilihat seperti berikut

t_{1i}	0,847182633	0,673856371	1,133273329	0,92714751	0,960723793
b_{i1}	0,847190144	0,673863883	1,13328084	0,927155022	0,960731305

Pola pelatihan yang terbentuk setelah pelatihan inputan keempat (B2) yaitu

Cluster (J)	Input yang dipelajari
1	A1, A2, B1, B2

Input Kelima (Citra Latih C1)

Bobot t_{ji} dan Bobot b_{iJ}

t_{1i}	0,847182633	0,673856371	1,133273329	0,92714751	0,960723793
b_{i1}	0,847190144	0,673863883	1,13328084	0,927155022	0,960731305

1. Menghitung aktifasi pertama unit F1

u	0	0	0	0	0
w	11266,55441	-2432,99829	-11693,4334	7725,335642	-886,91119
p	0	0	0	0	0
x	0,62014841	-0,13392027	-0,64364524	0,425228019	-0,04881852
q	0	0	0	0	0
v	0,62014841	0	0	0,425228019	0

2. Menghitung aktifasi kedua unit F1

u	0,824739151	0	0	0,565513335	0
w	11274,8018	-2432,99829	-11693,4334	7730,990775	-886,91119
p	0,824739151	0	0	0,565513335	0
x	0,620345579	-0,13386486	-0,64337891	0,425363215	-0,04879832
q	0,824739151	0	0	0,565513335	0
v	8,867737092	0	0	6,080496568	0

3. Menghitung sinyal unit F2
   Y_1 = 1,223029409
4. Memilih *cluster* pemenang, J = 1
5. Menghitung kondisi reset
   Mencari nilai *u*, *p* dan reset

u	0,824739151	0	0	0,565513335	0
p	1,587203521	0,606470734	1,019945996	1,399946094	0,864651414

r	0,782024953	0,048225193	0,081103819	0,561004104	0,068755142

Didapatkan nilai $\|r\|$= 0,969494404 sehingga J=1 diterima. Dihitung kembali unit *w*, *x*, *q* dan *v*

w	11274,8018	-2432,998292	-11693,4334	7730,990775	-886,91119
x	0,620345579	-0,133864857	-0,64337891	0,425363215	-0,04879832
q	0,616196554	0,235448808	0,395971405	0,543498012	0,335681729
v	6,782311118	2,354488077	3,959714047	5,86034334	3,356817287

6. Hitung bobot terbaru untuk unit pemenang J=1
   Nilai bobot t_{1i} dan b_{i1} dapat dilihat seperti berikut

t_{1i}	1,26640752	0,975741171	0,95920216	1,17020749	0,965413943
b_{i1}	1,266407249	0,9757409	0,95920189	1,17020722	0,965413673

Pola pelatihan yang terbentuk setelah pelatihan inputan kelima (C1) yaitu

Cluster (J)	Input yang dipelajari
1	A1, A2, B1, B2, C1

Setelah dilakukan pelatihan sebanyak 3 *epoch* didapatkan jaringan yang telah konvergen. Nilai bobot dan hasil pola pelatihan seperti berikut ini

Nilai bobot jaringan t_{Ji} dan b_{iJ}

t_{1i}	1,262148745	0,965423612	0,952682669	1,168831489	0,965400026
b_{i1}	1,262148745	0,965423612	0,952682669	1,168831489	0,965400026

Hasil pola pelatihan

Cluster (J)	Input yang dipelajari
1	A1, A2, B1, B2, C1

3.6.2 Perhitungan Data Uji

Data uji dicontohkan menggunakan lima citra wajah yang diambil dari *database* AT&T yang ditunjukkan sebagai berikut

A3	A4	B3	B4	C2
135	169	52	73	111
94	79	150	123	161
71	152	67	65	159
105	118	58	69	82
156	154	137	116	163
66	138	165	61	79
147	145	101	66	111
67	91	122	168	70
57	66	80	92	82
115	69	163	56	164

3.6.2.1 Menghitung Fitur PCA Uji

Sebelum dilakukan pengenalan wajah terlebih dahulu harus menghitung fitur data uji. Langkah pertama yaitu mencari matrik normalisasi data uji dengan cara mengurangi matrik uji dengan nilai rata-ratanya. Matrik rata-rata data uji ditunjukkan sebagai berikut

Y_6	Y_7	Y_8	Y_9	Y_{10}
33,7	50,9	-57,5	-15,9	-7,2
-7,3	-39,1	40,5	34,1	42,8
-30,3	33,9	-42,5	-23,9	40,8
3,7	-0,1	-51,5	-19,9	-36,2
54,7	35,9	27,5	27,1	44,8
-35,3	19,9	55,5	-27,9	-39,2
45,7	26,9	-8,5	-22,9	-7,2
-34,3	-27,1	12,5	79,1	-48,2
-44,3	-52,1	-29,5	3,1	-36,2
13,7	-49,1	53,5	-32,9	45,8

Kemudian mencari fitur PCA data uji dengan cara mengalikan *transpose* matrik V_{epx} data latih dengan matrik normalisasi data uji. Hasil perhitungan fitur PCA dapat dilihat sebagai berikut

f_{PCA_1}	f_{PCA_2}	f_{PCA_3}	f_{PCA_4}	f_{PCA_5}
898,7197	529,686	-8907,61	-5656,86	-8262,04
762,1549	-7127,42	5952,443	7627,275	-90,8474
-4857,82	4474,165	1560,135	7469,05	-5800,39
-3167	-629,02	-2788,49	-10466	-2715,29
1916,311	3117,099	1967,439	1730,433	-486,497

3.6.2.2 Pengenalan dengan ART-2

Dalam pengenalan wajah, nilai input yang digunakan adalah nilai fitur PCA data uji. Bobot jaringan dan parameter yang dipakai menggunakan bobot dan parameter dari proses pelatihan

Input Pertama (Citra Uji A3)

Menghitung aktifasi pada lapisan F1 sebanyak dua tahap

u	0,399497049	0,338791525	0	0	0,851834732
w	902,7147045	765,5428045	-4857,81689	-3166,9997	1924,829583
p	0,399497049	0,338791525	0	0	0,851834732
x	0,145045171	0,12300485	-0,78053772	-0,50886289	0,309275162
q	0,399497049	0,338791525	0	0	0,851834732
v	4,14001566	3,510920099	0	0	8,827622481

Kemudian menghitung sinyal F2

Y_1 = 1,653663309

Dipilih *cluster* pemenang J = 1, kemudian dihitung kembali nilai unit *u*, *p* dan reset

u	0,399497049	0,338791525	0	0	0,851834732
p	1,535430919	1,207672776	0,857414403	1,05194834	1,720694755
r	0,427528815	0,355262874	0,066282596	0,081321082	0,791531247

$||r|| = 0{,}972893012$, memenuhi syarat $||r|| > \rho$ sehingga citra uji A3 dikenali sistem

Input Kedua (Citra Uji A4)

Menghitung aktifasi pada *loop* F1

u	0	0	0,820506394	0	0,571637348
w	529,6860057	-7127,424337	4482,369854	-629,01978	3122,815414
p	0	0	0,820506394	0	0,571637348
x	0,05873808	-0,790376211	0,497060135	-0,06975343	0,346296067
q	0	0	0,820506394	0	0,571637348
v	0	0	8,702124075	0	6,062669544

Kemudian menghitung sinyal F2
$Y_1 = 1{,}333540932$

Dipilih *cluster* pemenang J = 1, kemudian dihitung kembali nilai unit *u*, *p* dan reset

u	0	0	0,820506394	0	0,571637348
p	1,135933871	0,868881251	1,677920797	1,05194834	1,440497371
r	0,088496607	0,067691478	0,769948529	0,08195359	0,557566599

$||r|| = 0{,}960641154$, memenuhi syarat $||r|| > \rho$ sehingga citra uji A4 dikenali sistem

Input Ketiga (Citra Uji B3)

Menghitung aktifasi pada *loop* F1

u	0	0,921377691	0,241492965	0	0,304539487
w	-8907,60749	5961,656862	1562,54944	-2788,49469	1970,483915
p	0	0,921377691	0,241492965	0	0,304539487
x	-0,78431095	0,524921285	0,137581796	-0,24552574	0,173500249
q	0	0,921377691	0,241492965	0	0,304539487
v	0	9,738698191	2,552511451	0	3,218895114

Kemudian menghitung sinyal F2
Y_1 = 1,413588369

Dipilih *cluster* pemenang J = 1, kemudian dihitung kembali nilai unit *u*, *p* dan reset

u	0	0,921377691	0,241492965	0	0,304539487
p	1,135933871	1,790258942	1,098907368	1,05194834	1,17339951
r	0,088322582	0,855599855	0,273212346	0,081792431	0,328025091

$||r||$ = 0,963736187, memenuhi syarat $||r|| > \rho$ sehingga citra uji B3 dikenali sistem

Input Keempat (Citra Uji B4)

Menghitung aktifasi pada *loop* F1

u	0	0,705273139	0,690642539	0	0,160008383
w	-5656,85875	7634,327462	7475,956488	-10465,9756	1732,033057
p	0	0,705273139	0,690642539	0	0,160008383
x	-0,35169738	0,47464027	0,464794054	-0,65068908	0,107683701
q	0	0,705273139	0,690642539	0	0,160008383
v	0	7,527371662	7,371219442	0	1,70776753

Kemudian menghitung sinyal F2
Y_1 = 1,493322616

Dipilih *cluster* pemenang J = 1, kemudian dihitung kembali nilai unit *u*, *p* dan reset

u	0	0,705273139	0,690642539	0	0,160008383
p	1,135933871	1,57415439	1,548056941	1,05194834	1,028868406
r	0,088151424	0,669468781	0,656089825	0,081633928	0,204013533

$||r||$ = 0,966798118, memenuhi syarat $||r|| > \rho$ sehingga citra uji B4 dikenali sistem

Input Kelima (Citra Uji C2)

Menghitung aktifasi pada *loop* F1

u	0	0	0	0	0
w	-8262,03601	-90,84738429	-5800,38831	-2715,28959	-486,496601
p	0	0	0	0	0
x	-0,78946609	-0,008680781	-0,55424714	-0,2594553	-0,04648643
q	0	0	0	0	0
v	0	0	0	0	0

Input kelima (C2) tidak dikenali oleh sistem karena semua nilai input dari proses normalisasi pada lapisan F1 bernilai 0.

Pola pengenalan yang terbentuk adalah

Cluster (J)	Input yang dikenali
1	A3, A4, B3, B4

Kemudian mencari jarak terdekat antara fitur citra uji dengan fitur citra *training* dalam setiap *cluster* dengan menggunakan jarak *Euclidean*.

Citra Uji	Status Pengenalan	Cluster	Hasil Identifikasi	Jarak Euclidean
A3	DIKENALI	1	A2	16284
A4	DIKENALI	1	B2	11751
B3	DIKENALI	1	B1	13713
B4	DIKENALI	1	B1	6642
C2	TIDAK DIKENALI	-	-	-

Dari hasil perhitungan jarak tersebut didapatkan 3 citra yang benar di identifikasi, yaitu citra uji A3, B3 dan B4, dari 4 citra yang berhasil teridentifikasi oleh sistem, sehingga tingkat akurasi dari pengenalan wajah sebesar

$$\text{Tingkat Akurasi} = \frac{\text{Jumlah benar identifikasi}}{\text{Jumlah yang teridentifkasi}} \times 100\%$$
$$= \frac{3}{4} \times 100\%$$
$$= 75\ \%$$

BAB IV
HASIL DAN PEMBAHASAN

Pada bab ini akan dijelaskan seluruh proses yang sudah dirancang pada bab sebelumnya, tampilan antarmuka dan bagian-bagian *source code* yang dibuat, serta analisa terhadap data yang dihasilkan sistem.

4.1 Lingkungan Implementasi

Implementasi merupakan proses transformasi representasi rancangan ke dalam bahasa pemrograman yang dapat dimengerti oleh komputer. Pada bab ini, lingkungan implementasi yang akan dijelaskan meliputi lingkungan implementasi perangkat keras dan perangkat lunak.

4.1.1 Lingkungan Perangkat Keras

Perangkat keras yang digunakan dalam pengembangan sistem pengenalan wajah dengan metode PCA dan ART-2 adalah sebuah *Notebook* dengan spesifikasi sebagai berikut :

1. *Processor* Intel® Core™ i3-370M, @2.40GHz
2. Memori 2 GB
3. *Harddisk* 320 GB
4. VGA nVIDIA GeForce 310M
5. Monitor 14“
6. *Keyboard*
7. *Mouse*

4.1.2 Lingkungan Perangkat Lunak

Perangkat lunak yang digunakan dalam pengembangan sistem pengenalan wajah dengan metode ART2 adalah sebagai berikut:

1. Sistem operasi *Windows 7 Ultimate.*
2. *Microsoft Visual C# 2010 Ultimate*
3. *ACDSee 10 Photo Manager*

4.2 Lingkungan Program

Berdasarkan analisa dan perancangan proses yang telah dipaparkan pada Bab III, maka pada bab ini akan dijelaskan proses-proses implementasinya.

4.2.1 Implementasi *Load* Citra *Grayscale*

Tahap awal yang dari sistem pengenalan wajah adalah memanggil file citra *grayscale* ke dalam perangkat lunak dan kemudian dibentuk matrik wajah dari nilai piksel citra grayscale tersebut yang nantinya diperlukan untuk pemrosesan klasifikasi dan pengenalan wajah. Prosedur yang digunakan ditunjukkan pada *source code* 4.1

```
MatriksLatih = new double[JumCitra][];
for (int i = 0; i < JumCitra; i++)
    MatriksLatih[i] = new double[JumKomponen];

int x = 0;
foreach (string namaCitra in ofdImage.FileNames)
{
    int y = 0;
    Bitmap gambar = new Bitmap(@namaCitra);
    for (int i = 0; i < gambar. Width; i++)
        for (int j = 0; j < gambar.Height; j++)
        {
            //mengambil nilai piksel dari citra asli
            Color pixelColor = gambar.GetPixel(i, j);

            //membuat nilai grayscale dari piksel
            int grayScale = (int)( pixelColor.R * .299 +
pixelColor.G * .587 + pixelColor.B * .114);

             //menyimpan nilai grayscale kedalam array
            MatriksLatih[x][y] = grayScale;
            y++;
        }
    x++;
}
```

***Source code* 4.1** Inisialisasi Matrik dari Citra *Grayscale*

4.2.2 Implementasi *Transpose* Matrik

Fungsi *Transpose* digunakan untuk mendapatkan matrik *transpose* yang nantinya akan digunakan untuk perhitungan matrik

lainnya. Fungsi *transpose* ditunjukkan pada *source code* 4.2

```
public static double[][] Transpose(double[][] input)
{
    double[][] temp = new double[input[0].Length][];
    for (int i = 0; i < input[0].Length; i++)
        temp[i] = new double[input.Length];
    for (int i = 0; i < input.Length; i++)
        for (int j = 0; j < input[0].Length; j++)
            temp[j][i] = input[i][j];
    return temp;
}
```

***Source code* 4.2** Fungsi *Transpose* Matrik

4.2.3 Implementasi *Multiply* Matrik

Fungsi *Multiply* merupakan fungsi operasi perkalian dua matrik. Fungsi *Multiply* ditunjukkan pada *source code* 4.3

```
public static double[][] Multiply(double[][] a, double[][] b)
{
    //matrik harus memenuhi syarat pjg col A = pjg row B
    int m = a.Length;       // col A
    int n = a[0].Length;    // row A
    int p = b.Length;       // col B

    double[][] matM = new double[p][];
    for (int i = 0; i < p; i++)
        matM[i] = new double[n];

    for (int i = 0; i < p; i++)
    {
        double[] rowB = b[i];
        for (int j = 0; j < n; j++)
        {
            double s = 0;
            for (int k = 0; k < m; k++)
                s += a[k][j] * rowB[k];
            matM[i][j] = s;
        }
    }
    return matM;
}
```

***Source code* 4.3** Fungsi *Multiply* Matrik

4.2.4 Implementasi *Principal Component Analysis*

Tahap selanjutnya adalah melakukan ekstraksi fitur dan reduksi dimensi menggunakan metode *Principal Component Analysis*. Langkah awal pada proses PCA yaitu membentuk matrik normalisasi wajah. Matrik normalisasi wajah terbentuk dari hasil pengurangan matrik wajah dengan nilai rata-rata dari setiap citra. Proses pembentukan matrik normalisasi ditunjukkan pada *source code* 4.4.

```
//mencari matrik normalisasi
for (int i = 0; i < MatNorm.Length; i++)
{
    var RataRata = PCA.Mean(MatriksLatih[i]);
    for (int j = 0; j < MatNorm[i].Length; j++)
        MatNorm[i][j] = MatriksLatih[i][j] - RataRata;
}
```

***Source code* 4.4** Inisialisasi Matrik Normalisasi

Kemudian dari matrik normalisasi dibentuk matrik kovarian menggunakan fungsi *CovarianceMatrix*, yang ditunjukkan pada *Source code* 4.5.

```
public static double[,] CovarianceMatrix(double[][] input)
{
    var temp = Common.Transpose(input);
    double[,] Cov = new double[input.Length, input.Length];
    for (int i = 0; i < input.Length; i++)
        for (int j = 0; j < input.Length; j++)
        {
            double tempo = 0;
            for (int z = 0; z < input[0].Length; z++)
                tempo += temp[z][j] * input[i][z];
            Cov[j, i] = tempo / (input[0].Length - 1);
        }
    return Cov;
}
```

***Source code* 4.5** Fungsi Matrik Kovarian

Setelah terbentuk matrik kovarian, dilakukan proses dekomposisi eigen untuk mendapatkan matrik *eigenvalue* dan *eigenvector*. Dari hasil proses dekomposisi eigen, dilakukan

pembentukan matrik *Vexp* dengan cara perkalian matrik normalisasi dengan matrik *eigenvector* yang bersesuaian dengan *eigenvalue* yang terpilih. Langkah terakhir adalah menghitung matrik fitur yang didapat dari perkalian antara matrik normalisasi dengan matrik *Vexp* yang sudah di-*transpose*. Proses dekomposisi eigen, proses pembentukan *Vexp*, dan proses pembentukan matrik fitur PCA ditunjukkan pada *source code* 4.6.

```
//perhitungan dekomposisi eigen menggunakan tridiagonal
PCA.EigenDecomposition(MatCov, out MatEigenVec, out MatEigenVal);

//memilih eigenvector yang terpilih
for (int i = 0; i < JumPCA; i++)
    EigenSelected[i] = MatEigenVec[i];
//menghitung matrik Vexp
frmUtama.Vexp = Common.Multiply(MatNorm, EigenSelected);
var VexpT = Common.Transpose(frmUtama.Vexp);
//menghitung matrik fitur
frmUtama.FiturLatih = Common.Multiply(VexpT, MatNorm);
```

***Source code* 4.6** Perhitungan Dekomposisi Eigen, Matrik *Vexp* dan Matrik Fitur

4.2.5 Implementasi Klasifikasi ART-2

Pada tahap ini, citra akan diklasifikasi ke dalam *cluster-cluster* yang bersesuai berdasarkan nilai kedekatan fiturnya. Nilai input yang digunakan pada proses ini adalah nilai fitur hasil ekstraksi PCA. Langkah pertama dalam tahap ini adalah menghitung aktifasi lapisan F1 sebanyak dua kali. Pada lapisan F1 terdapat proses normalisasi nilai input sehingga nilai yang dihasilkan tidak acak, dengan cara memilih hanya nilai input yang yang lebih besar dari nilai *choice parameter* (θ). Proses aktifasi lapisan F1 ditunjukkan pada *source code* 4.7.

```
//Aktivasi Lapisan F1 - Loop 1
w = ART2.UpdateUnit(frmUtama.FiturLatih[j]);
NormW = ART2.Magnitude(w);
x = ART2.UpdateUnit(w, NormW);
v = ART2.ActivationF(x, frmUtama.paramTheta);

//Aktifasi Lapisan F1 - Loop 2
NormV = ART2.Magnitude(v);
u = ART2.UpdateUnit(v, NormV);
for (int k = 0; k < frmUtama.FiturLatih[j].Length; k++)
  w[k] = frmUtama.FiturLatih[j][k] + frmUtama.varA * u[k];
p = ART2.UpdateUnit(u);
NormP = ART2.Magnitude(p);
q = ART2.UpdateUnit(p, NormP);
NormW = ART2.Magnitude(w);
x = ART2.UpdateUnit(w, NormW);
//normalisasi nilai input
var tmpX = ART2.ActivationF(x, frmUtama.varTheta);
var tmpQ = ART2.ActivationF(q, frmUtama.varTheta);
for (int k = 0; k < frmUtama.FiturLatih[j].Length; k++)
  v[k] = tmpX[k] + frmUtama.varB * tmpQ[k];
```

***Source code* 4.7** Aktifasi Lapisan F1

Langkah berikutnya adalah menghitung sinyal pada lapisan F2 dari tiap-tiap input sinyal yang dikirim oleh lapisan F1. Proses aktifasi lapisan F2 ditunjukkan pada *source code* 4.8.

```
//Aktifasi Lapisan F2
for (int k = 0; k < MaxCluster; k++)
{
  double s = 0;
  for (int l = 0; l < tmpBobotBj[k].Length; l++)
    s += tmpBobotBj[k][l] * p[l];
  YCluster[k] = s;
}
```

***Source code* 4.8** Aktifasi Lapisan F2

Dari hasil perhitungan sinyal pada lapisan F2 kemudian dipilih *cluster* dengan sinyal terbesar (*cluster* pemenang). Setelah itu dilakukan perhitungan kondisi reset untuk *cluster* pemenang, dimana apabila nilai reset lebih kecil dari nilai *vigilance* maka *cluster* tersebut akan ditolak (*inhibited*) dan dilakukan pemilihan *cluster* dengan sinyal terbesar lainnya. Sedangkan jika nilai reset lebih besar dari nilai *vigilance*, maka *cluster* pemenang berhak untuk

mempelajari nilai input (resonansi). Apabila seluruh *cluster* ditolak maka akan dibentuk *cluster* baru. Proses pemilihan *cluster* dan perhitungan kondisi reset ditunjukkan pada *source code* 4.9.

```
//Memilih Winner & Cek Kondisi Reset
winner = ART2.ChooseF2(YCluster, Inhibited);
if (winner != -1)
{
  NormV = ART2.Magnitude(v);
  u = ART2.UpdateUnit(v, NormV);
  for (int k = 0; k < frmUtama.FiturLatih[j].Length; k++)
    p[k] = u[k] + frmUtama.varD * tmpBobotTj[winner][k];
  NormU = ART2.Magnitude(u);
  NormP = ART2.Magnitude(p);
  for (int k = 0; k < frmUtama.FiturLatih[j].Length; k++)
    r[k] = (u[k] + (frmUtama.varC * p[k])) / (NormU +
(frmUtama.varC * NormP));
  NormR = ART2.Magnitude(r);
  if (NormR < frmUtama.varVigilance)
    Inhibited--;
  else
    Resonance = true;
}
else
{
  if (MaxCluster != JumCitra)
  {
    MaxCluster += 1;
    tmpBobotBj = ART2.CreateCluster(tmpBobotBj, MaxCluster,
frmUtama.varBobotB);
    tmpBobotTj = ART2.CreateCluster(tmpBobotTj, MaxCluster,
frmUtama.varBobotT);
    YCluster = new double[MaxCluster];
    winner = MaxCluster - 1;
    Resonance = true;
  }
  else
    Exhausted = true;
}
```

***Source code* 4.9** Pemilihan *Cluster* dan Perhitungan Kondisi Reset

Setelah proses perhitungan cluster, dilakukan proses pembaruan bobot untuk *cluster* pemenang (kondisi resonansi) yang ditunjukkan pada *source code* 4.10.

```
//Kondisi resonansi & update bobot
if (Resonance)
{
  for (int k = 0; k < frmUtama.FiturLatih[j].Length; k++)
    w[k] = frmUtama.FiturLatih[j][k] + frmUtama.varA * u[k];
  NormW = ART2.Magnitude(w);
  x = ART2.UpdateUnit(w, NormW);
  NormP = ART2.Magnitude(p);
  q = ART2.UpdateUnit(p, NormP);
  tmpQ = ART2.ActivationF(q, frmUtama.varTheta);
  tmpX = ART2.ActivationF(x, frmUtama.varTheta);
  for (int k = 0; k < frmUtama.FiturLatih[j].Length; k++)
    v[k] = tmpX[k] + frmUtama.varB * tmpQ[k];

  //Update Bobot Unit Pemenang J
  for (int k = 0; k < frmUtama.FiturLatih[j].Length; k++)
  {
    tmpBobotBj[winner][k] = (frmUtama.varAlpha * frmUtama.varD *
u[k]) + (1 + frmUtama.varAlpha * frmUtama.varD * (frmUtama.varD -
1)) * tmpBobotBj[winner][k];
    tmpBobotTj[winner][k] = (frmUtama.paramAlpha *
frmUtama.paramD * u[k]) + (1 + frmUtama.paramAlpha *
frmUtama.paramD * (frmUtama.paramD - 1)) * tmpBobotTj[winner][k];
  }

  //simpan ke dalam cluster pemenang
  Array.Resize(ref ClusterPelatihan[j], ep + 1);
  ClusterPelatihan[j][ep] = winner.ToString();
}
```

***Source code* 4.10** Pembaruan Bobot

Langkah yang terakhir adalah memeriksa kondisi berhenti iterasi *epoch* yaitu membandingkan nilai bobot pada *epoch* sebelum dan hasil bobot *epoch* sekarang apakah terdapat perubahan. Apabila tidak terdapat perubahan maka jaringan telah stabil dan iterasi dihentikan. Proses pemeriksaan kondisi *epoch* ditunjukkan pada *source code* 4.11

```
//Pemeriksaan kondisi epoch & bobot
bool Condition = ART2.CheckIteration(frmUtama.BobotBj,
tmpBobotBj, tmpBobotTj);
if (Condition)
{
    ep++;
    Convergen = true;
}
else
{
    MaxCluster = tmpBobotBj.Length;
    frmUtama.BobotBj = ART2.UpdateCluster(tmpBobotBj);
    frmUtama.BobotTj = ART2.UpdateCluster(tmpBobotTj);
    ep++;
}
```

***Source code* 4.11** Pemeriksaan Kondisi *Epoch*

4.2.6 Implementasi Pengenalan ART-2

Tahap yang terakhir adalah tahap pengenalan (identifikasi) ART-2. Sebelum dilakukan proses pengenalan (identifikasi), dilakukan perhitungan fitur uji terlebih dahulu. Fitur uji didapatkan dari perkalian antara matrik normalisasi uji dengan *transpose* matrik *Vexp* latih. Proses perhitungan fitur uji ditunjukkan pada *source code* 4.12.

```
var MatNorm = new double[JumCitra][];
for (int i = 0; i < JumCitra; i++)
    MatNorm[i] = new double[MatriksUji[i].Length];
for (int i = 0; i < MatNorm.Length; i++)
{
    var RataRata = PCA.Mean(MatriksUji[i]);
    for (int j = 0; j < MatNorm[i].Length; j++)
        MatNorm[i][j] = MatriksUji[i][j] - RataRata;
}
var VexpT = Common.Transpose(frmUtama.Vexp);
FiturUji = Common.Multiply(VexpT, MatNorm);
```

***Source code* 4.12** Inisialisasi Fitur Uji

Proses pengenalan ART-2 memiliki struktur tahap yang sama dengan proses klasifikasi ART-2. Hal yang yang membedakan antara kedua proses tersebut adalah pada proses pengenalan ART-2 tidak dilakukan proses pembaruan bobot dan terdapat penambahan

proses pengambilan keputusan. Proses pengenalan ART-2 ditunjukkan pada *source code* 4.13.

```
//Aktivasi Lapisan F1 - Loop 1
w = ART2.UpdateUnit(FiturUji[j]);
NormW = ART2.Magnitude(w);
x = ART2.UpdateUnit(w, NormW);
v = ART2.ActivationF(x, frmUtama.paramTheta);

//Aktifasi Lapisan F1 - Loop 2
NormV = ART2.Magnitude(v);
u = ART2.UpdateUnit(v, NormV);
for (int k = 0; k < FiturUji[j].Length; k++)
    w[k] = FiturUji[j][k] + frmUtama.paramA * u[k];
p = ART2.UpdateUnit(u);
NormP = ART2.Magnitude(p);
q = ART2.UpdateUnit(p, NormP);
NormW = ART2.Magnitude(w);
x = ART2.UpdateUnit(w, NormW);
var tmpX = ART2.ActivationF(x, frmUtama.paramTheta);
var tmpQ = ART2.ActivationF(q, frmUtama.paramTheta);
for (int k = 0; k < FiturUji[j].Length; k++)
    v[k] = tmpX[k] + frmUtama.paramB * tmpQ[k];

//Aktifasi Lapisan F2
for (int k = 0; k < frmUtama.BobotBj.Length; k++)
{
    double s = 0;
    for (int l = 0; l < frmUtama.BobotBj[k].Length; l++)
        s += frmUtama.BobotBj[k][l] * p[l];
    YCluster[k] = s;
}

//Memilih Winner & Cek Kondisi Reset
winner = ART2.ChooseF2(YCluster, Inhibited);
if (winner != -1)
{
  NormV = ART2.Magnitude(v);
  u = ART2.UpdateUnit(v, NormV);
  for (int k = 0; k < frmUtama.FiturLatih[j].Length; k++)
    p[k] = u[k] + frmUtama.varD * tmpBobotTj[winner][k];
  NormU = ART2.Magnitude(u);
  NormP = ART2.Magnitude(p);  for (int k = 0; k <
frmUtama.FiturLatih[j].Length; k++)
    r[k] = (u[k] + (frmUtama.varC * p[k])) / (NormU +
(frmUtama.varC * NormP));
  NormR = ART2.Magnitude(r);
  if (NormR < frmUtama.varVigilance)
    Inhibited--;
  else
    Resonance = true;
```

```
}
 else
  Exhausted = true;

ClusterPengenalan[j] = winner;
ArrayReset[j] = NormR;
```

***Source code* 4.13** Pengenalan ART-2

Pada proses pengambilan keputusan, citra uji akan "dikenali" apabila citra uji masuk pada salah satu *cluster* hasil klasifikasi dan "tidak dikenali" apabila tidak termasuk pada seluruh *cluster*. Dalam proses pengambilan keputusan, citra hasil pengenalan di identifikasi dengan cara menghitung jarak kedekatan antara fitur uji dengan fitur-fitur latih pada *cluster* tersebut menggunakan jarak *Euclidean*. Proses pengambilan keputusan ditunjukkan pada *source code* 4.14

```
//Proses Pengambilan Keputusan
for (int i = 0; i < JumCitra; i++)
{
  hasil[i][0] = ArrayReset[i].ToString();
  if (ClusterPengenalan[i] != -1)
  {
    hasil[i][1] = "DIKENALI";
    hasil[i][2] = (ClusterPengenalan[i] + 1).ToString();
    if (frmUtama.Cluster[ClusterPengenalan[i]].Length == 1)
    {
      hasil[i][3] = frmUtama.Cluster[ClusterPengenalan[i]][0];
      hasil[i][4] = frmUtama.Cluster[ClusterPengenalan[i]][0];
      var indeks = Array.IndexOf(frmUtama.NamaFile, frmUtama.
Cluster[ClusterPengenalan[i]][0]);
      hasil[i][5] = Common.EuclideanDistance(FiturUji[i],
frmUtama.FiturLatih[indeks]).ToString();
    }
    else
    {
      var res = ofdImage.SafeFileNames[i].Remove(ofdImage.
SafeFileNames[i].IndexOf("."));
      res = res.Trim(EndsWith);
      var pjg = frmUtama.Cluster[ClusterPengenalan[i]].Length;
      int[] indeks = new int[0];
      for (int j = 0; j < pjg; j++)
      {
        var res2 = frmUtama.Cluster[ClusterPengenalan[i]][j].Trim
(EndsWith);
        if (res2.Equals(res))
        {
          int pos = indeks.Length;
```

```
            Array.Resize(ref indeks, pos + 1);
            indeks[pos] = Array.IndexOf(frmUtama.NamaFile,
frmUtama.Cluster[ClusterPengenalan[i]][j]);
          }
          hasil[i][3] += frmUtama.Cluster[ClusterPengenalan[i]][j]
+ ", ";
        }
        double[] jarak;
        if (indeks.Length < 1)
        {
          jarak = new double[pjg];
          for (int j = 0; j < pjg; j++)
          {
            int ind = Array.IndexOf(frmUtama.NamaFile, frmUtama.
Cluster[ClusterPengenalan[i]][j]);
            jarak[j] = Common.EuclideanDistance(FiturUji[i],
frmUtama.FiturLatih[ind]);
          }
          hasil[i][4] = frmUtama.Cluster[ClusterPengenalan[i]]
[Array.IndexOf(jarak, jarak.Min())];
        }
        else
        {
          jarak = new double[indeks.Length];
          for (int j = 0; j < indeks.Length; j++)
            jarak[j] = Common.EuclideanDistance(FiturUji[i],
frmUtama.FiturLatih[indeks[j]]);
          hasil[i][4] = frmUtama.NamaFile[indeks[Array.IndexOf
(jarak, jarak.Min())]];
        }
        hasil[i][5] = jarak.Min().ToString();
      }
    }
    else
    {
      hasil[i][1] = "TIDAK DIKENALI";
      hasil[i][2] = "-";
      hasil[i][3] = "-";
      hasil[i][4] = "-";
      hasil[i][5] = "0";
    }
  }
```

***Source code* 4.14** Pengambilan Keputusan

4.3 Implementasi Antarmuka

4.3.1 Antarmuka Utama

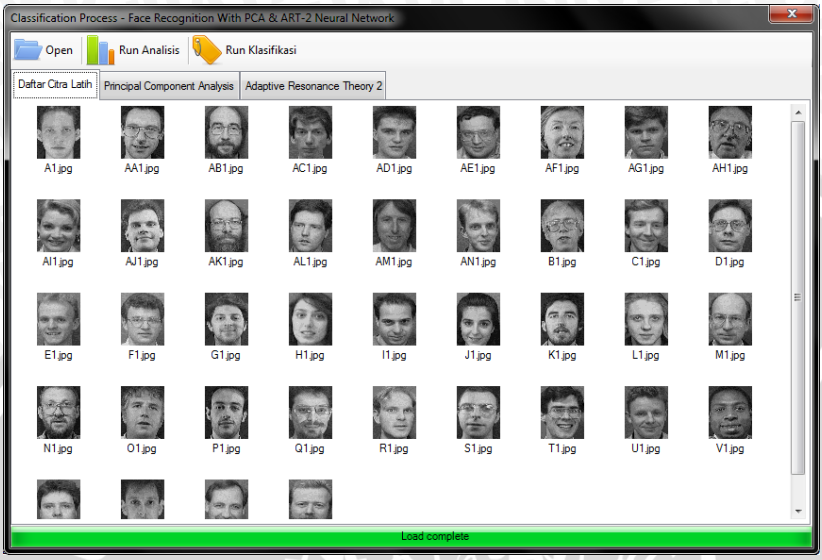
Berdasarkan rancangan antarmuka yang telah dikemukakan pada Bab III maka dihasilkan antarmuka utama pada Gambar 4.1.

Pada tampilan Gambar 4.1 antarmuka terdapat tiga tombol yaitu tombol *face classification* yang digunakan untuk menampilkan *form* klasifikasi, tombol *face recognition* yang digunakan untuk menampilkan *form* pengenalan wajah, dan tombol *exit* untuk keluar dari aplikasi.

Gambar 4.1 Antarmuka Utama

4.3.2 Antarmuka Klasifikasi Citra Wajah

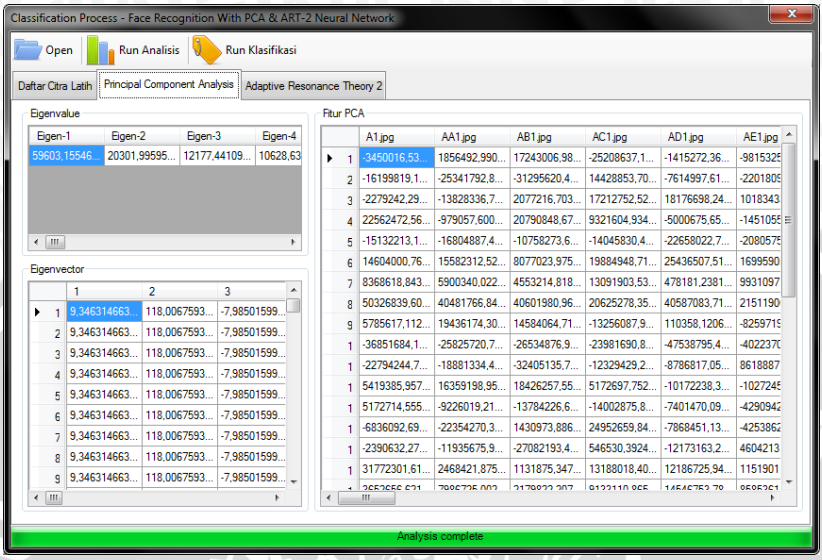
Pada gambar 4.2 merupakan antarmuka dari *form* klasifikasi citra wajah. Pada *form* klasifikasi wajah, antarmuka dibagi menjadi beberapa *tabControl* antara lain *tabPage* daftar citra latih yang digunakan untuk menampilkan daftar citra secara *thumbnails*, *tabPage Principal Component Analysis* yang digunakan untuk menampilkan proses perhitungan ekstraksi fitur PCA, dan *tabPage Adaptive Resonance Theory 2* yang digunakan untuk menampilkan proses klasifikasi citra wajah menggunakan metode ART-2.



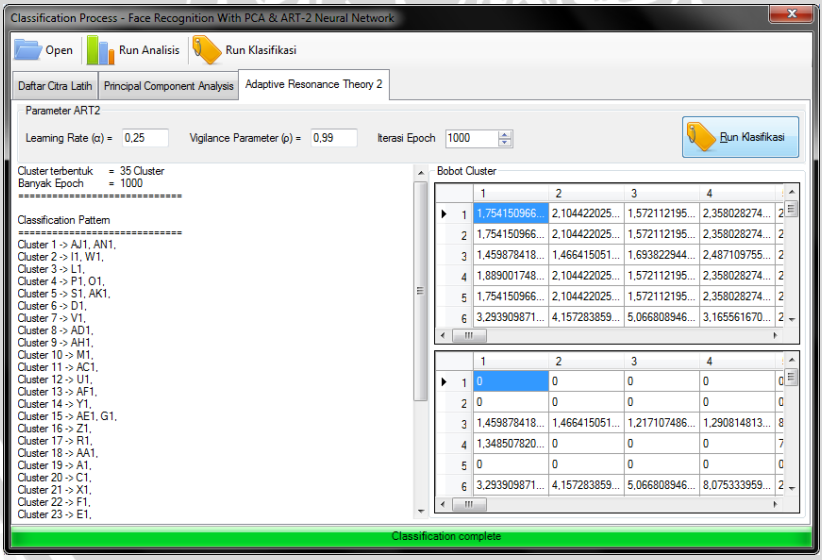
Gambar 4.2 Antarmuka *tabPage* Daftar Citra Latih

Pada Gambar 4.3 merupakan *tabPage Principal Component Analysis* disertai dengan tampilan hasil ekstraksi fitur PCA. Untuk menjalankan proses ekstraksi fitur memerlukan file citra *grayscale* dengan format JPG dan tombol *Run Analisis* digunakan untuk melakukan eksekusi proses ekstraksi fitur PCA. Hasil proses dekomposisi eigen dalam proses ekstraksi fitur akan ditampilkan pada tampilan *eigenvalue* dan *eigenvector*. Dari tampilan *eigenvalue* dan *eigenvector* dapat diketahui berapa *eigenvalue* yang terbentuk dan berapa *eigenvector* yang terpilih untuk digunakan dalam proses perhitungan fitur PCA, yang hasilnya akan ditampilkan pada tampilan fitur PCA dalam sistem.

Gambar 4.4 merupakan *tabPage Adaptive Resonance Theory 2* disertai dengan tampilan hasil proses klasifikasi. Input yang diminta sistem adalah hasil ekstraksi fitur PCA, sedangkan input *learning parameter* dengan nilai default 0,25, input *vigilance parameter* dengan nilai default 0,97, input iterasi *epoch* dengan nilai default 1000, dan tombol *Run Klasifikasi* digunakan untuk melakukan eksekusi proses klasifikasi ART-2. Dari tampilan hasil proses klasifikasi dapat diketahui berapa *cluster* yang terbentuk, pola hasil klasifikasi, nilai bobot cluster, dan *epoch* pada saat jaringan konvergen.



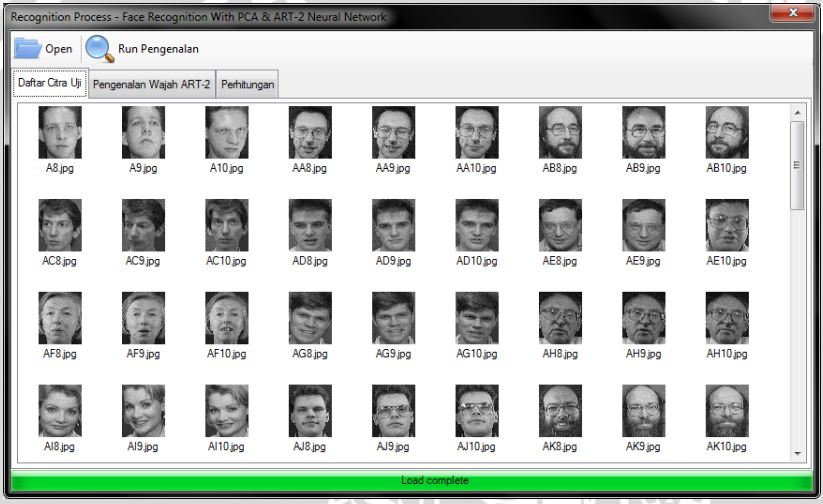
Gambar 4.3 Antarmuka *tabPage Principal Component Analysis* disertai hasil proses ekstraksi fitur



Gambar 4.4 Antarmuka *tabPage Adaptive Resonance Theory 2* disertai hasil proses klasifikasi wajah

4.3.3 Antarmuka Pengenalan Citra Wajah

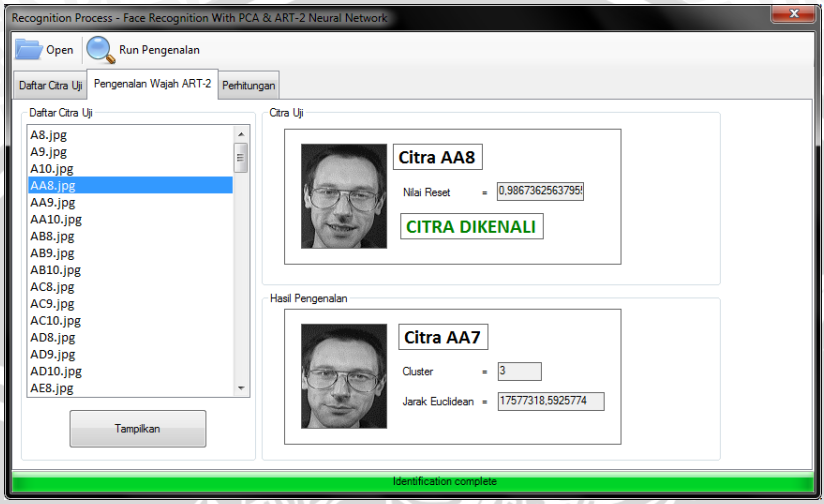
Pada Gambar 4.5 merupakan antarmuka dari pengenalan citra wajah. Pada *form* pengenalan wajah, antarmuka dibagi menjadi beberapa *tabControl* antara lain *tabPage* daftar citra uji yang digunakan untuk menampilkan citra-citra uji secara *thumbnails*, *tabPage* pengenalan wajah yang digunakan untuk menampilkan proses pengenalan wajah, dan *tabPage* perhitungan yang digunakan untuk menampilkan hasil perhitungan fitur uji dan uraian pengenalan.



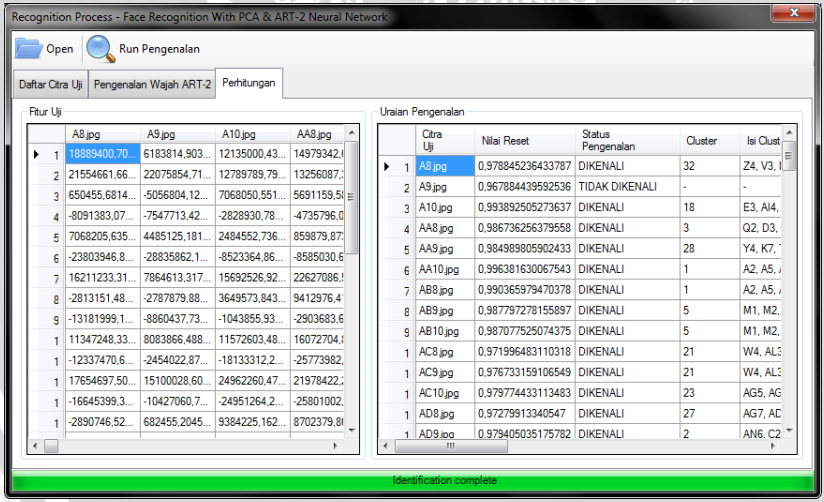
Gambar 4.5 Antarmuka *tabPage* Daftar Citra Uji

Pada Gambar 4.6 merupakan antarmuka *tabPage* pengenalan wajah disertai dengan tampilan hasil pengenalan wajah. Untuk menjalankan proses pengenalan wajah memerlukan file citra *grayscale* dengan format JPG yang akan dibentuk menjadi fitur uji dan tombol *Run Pengenalan* yang digunakan untuk melakukan eksekusi proses pengenalan. Pada bagian daftar citra uji akan ditampilkan nama-nama citra yang diujikan dan tombol tampilkan yang digunakan untuk melihat hasil pengenalan citra uji yang dipilih. Kemudian pada bagian citra uji akan ditampilkan atribut seperti nama citra uji, nilai reset, dan status pengenalan. Pada bagian hasil pengenalan akan ditampilkan nama citra hasil pengenalan, *cluster* hasil pengenalan dan nilai kemiripan. Sedangkan untuk uraian fitur

uji dan uraian hasil proses pengenalan yang lebih lengkap akan ditampilkan pada *tabPage* perhitungan yang ditunjukkan pada gambar 4.7.



Gambar 4.6 Antarmuka *tabPage* Pengenalan Wajah



Gambar 4.7 Antarmuka *tabPage* Perhitungan

4.4 Skenario Pengujian

Pada sistem pengenalan wajah dengan menggunakan metode PCA dan ART-2 akan dilakukan beberapa proses pengujian. Proses pengujian yang pertama adalah pengujian terhadap parameter proses belajar jaringan untuk memperoleh parameter yang tepat untuk identifikasi. Parameter yang diujikan yaitu parameter *vigilance* dan *learning rate*. Jumlah citra yang akan digunakan pada pengujian pertama sebanyak 40 citra (1 set) dari 40 individu.

Pengujian kedua adalah pengujian identifikasi JST ART-2 terhadap masukan citra uji guna mengetahui tingkat keakuratan metode ini. Citra latih yang digunakan adalah citra wajah dari 40 individu yang kemudian dicobakan untuk klasifikasi dengan beberapa set citra latih, yaitu 80 citra (2 set), 120 citra (3 set), 160 citra (4 set), dan 200 citra (5 set). Citra uji terdiri dari 120 citra (3 set) dari 40 individu yang sama.

Keseluruhan citra pengujian menggunakan *database* wajah *AT&T Cambrigde*. Citra-citra yang digunakan pada saat pengujian dapat dilihat pada bagian lampiran 1 dan lampiran 2.

4.5 Hasil Pengujian

Hasil pengujian aplikasi pengenalan wajah dengan menggunakan metode PCA & ART-2 didapatkan dari hasil skenario pengujian yang telah disebutkan pada subbab sebelumnya

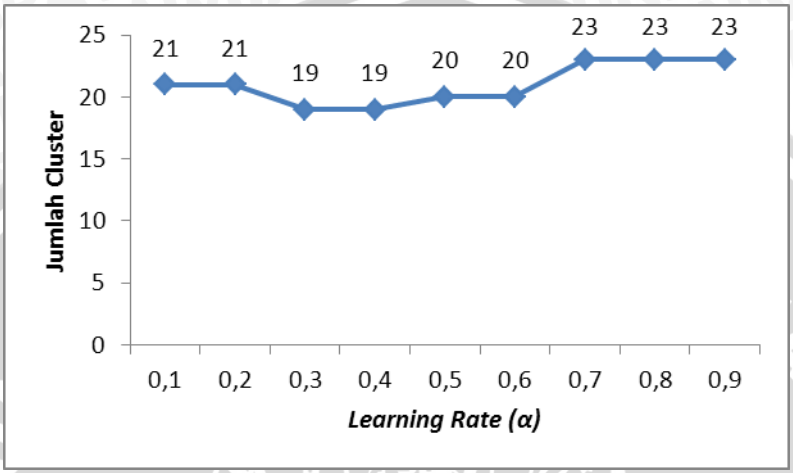
4.5.1 Pengujian Klasifikasi dengan ART-2

4.5.1.1 Pengaruh Nilai *Learning Rate*

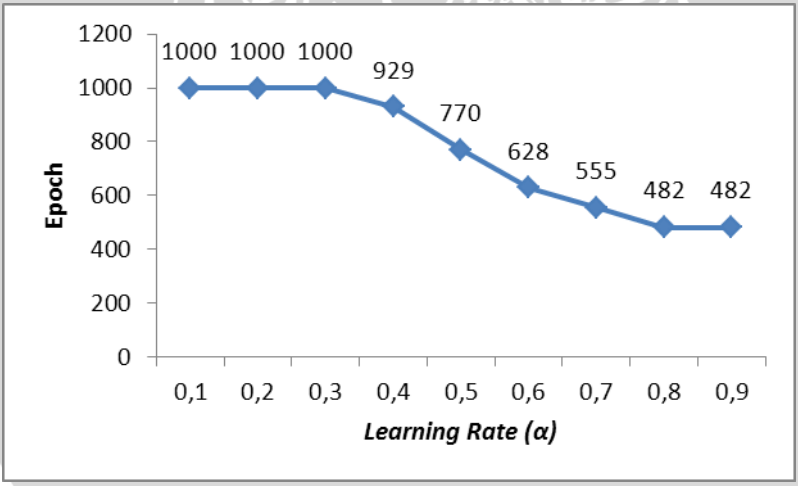
Learning rate adalah parameter yang menentukan kecepatan belajar dari jaringan. Pada penelitian ini dilakukan pengujian dengan untuk nilai *learning rate* antara α = 0.1 sampai 0.9. Hasil dari pengujian untuk beberapa nilai *learning rate* yang berbeda dapat dilihat pada lampiran 7.

Dari grafik pada gambar 4.8 dapat diketahui bahwa jumlah *cluster* yang terbentuk fluktuatif berdasarkan perbedaan nilai *learning rate*. Pada nilai α = 0.1 sampai 0.4, jumlah *cluster* yang terbentuk semakin sedikit. Sedangkan pada nilai α = 0.5 hingga 0.9, jumlah *cluster* yang terbentuk semakin banyak. Nilai *learning rate* juga mempengaruhi jumlah *epoch* yang dibutuhkan. Dari grafik pada gambar 4.9 dapat diketahui bahwa semakin besar nilai *learning rate*

maka semakin sedikit *epoch* yang dibutuhkan agar jaringan konvergen.



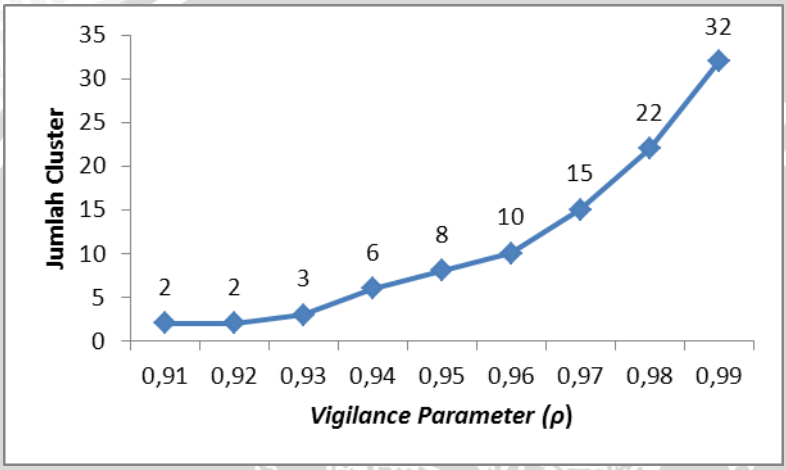
Gambar 4.8 Grafik pengaruh perubahan nilai *Learning Rate (α)* terhadap jumlah *cluster*



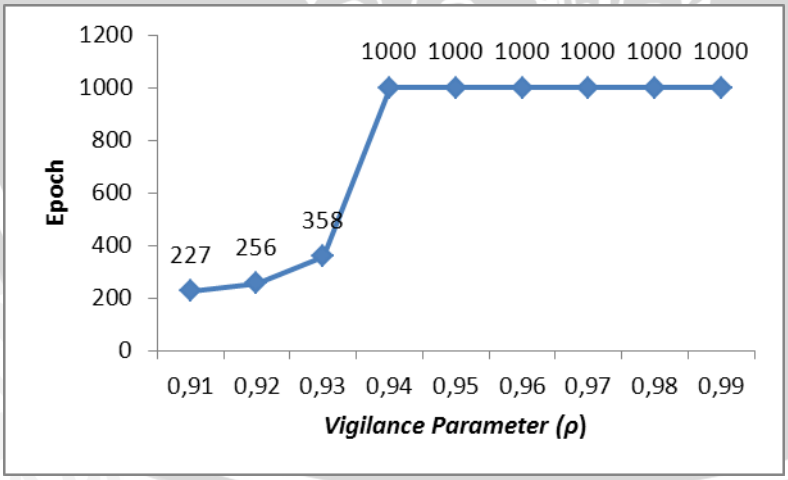
Gambar 4.9 Grafik pengaruh perubahan nilai *Learning Rate (α)* terhadap jumlah *epoch*

4.5.1.2 Pengaruh Nilai *Vigilance Parameter*

Parameter *vigilance (ρ)* merupakan parameter yang mempengaruhi pembentukan *cluster-cluster*. Pada penelitian ini dilakukan pengujian dengan menggunakan beberapa nilai *vigilance*, yaitu ρ = 0.91 sampai 0.99. Hasil dari pengujian untuk beberapa nilai *vigilance* dapat dilihat pada lampiran 8.



Gambar 4.10 Grafik pengaruh perubahan nilai *Vigilance (ρ)* terhadap jumlah *cluster*



Gambar 4.11 Grafik pengaruh perubahan nilai *Vigilance (ρ)* terhadap jumlah *epoch*

Dari grafik pada gambar 4.10 dan gambar 4.11 dapat diketahui bahwa semakin besar nilai *vigilance* maka jumlah *cluster* yang terbentuk semakin banyak. Demikian juga pada epoch yang dibutuhkan, semakin besar nilai *vigilance* maka *epoch* yang dibutuhkan juga semakin banyak.

4.5.2 Pengujian Identifikasi dengan ART-2

Pada pengujian identifikasi dengan ART-2, parameter *vigilance* dan *learning* yang digunakan yaitu $\rho = 0.97$ dan $\alpha = 0.25$.

4.5.2.1 Identifikasi untuk 80 citra latih dan 120 citra uji

Pada pengujian untuk 80 citra latih dan 120 citra uji, citra latih diklasifikasi oleh sistem ke dalam 15 *cluster*. Hasil klasifikasi untuk citra 80 latih dapat dilihat pada bagian lampiran 3.

Hasil pengujian identifikasi untuk citra uji dapat dilihat pada Tabel 4.1. Dari hasil pengujian 120 citra uji, sebanyak 3 citra tidak dikenali oleh sistem, 57 citra salah dalam identifikasi dan 60 citra tepat dalam identifikasi. Tingkat keakuratan yang didapatkan sebesar 51,28 %.

Tabel 4.1 Beberapa hasil identifikasi untuk 120 citra uji dengan 80 citra latih

Citra Uji	Status Pengenalan	Cluster	Hasil Identifikasi	Jarak Euclidean
A8	DIKENALI	2	B2	46494131,24
A9	DIKENALI	4	AM1	54852116,64
A10	DIKENALI	1	G2	45086747,91
AA8	DIKENALI	1	AA1	46828210,17
AA9	DIKENALI	1	AA1	38035151,1
AA10	DIKENALI	1	AA1	26135233,87
AB8	DIKENALI	1	AB1	50579834,52
AB9	DIKENALI	1	AB1	52748692,79
AB10	DIKENALI	2	S1	43668516,47
AC8	DIKENALI	1	G2	52560196,9
AC9	DIKENALI	13	AG2	48369083,34
AC10	DIKENALI	1	K1	50825174,16

AD8	DIKENALI	3	AD1	37150154,23
AD9	DIKENALI	14	I2	48566252,46
AD10	DIKENALI	3	AD1	28925553,39
AE8	DIKENALI	14	Z2	47711050,47
AE9	DIKENALI	14	I2	45148981,69
AE10	DIKENALI	2	AK2	43887202,59
AF8	DIKENALI	2	AF2	44619949,05
AF9	DIKENALI	1	K1	41015648,4
AF10	DIKENALI	1	K1	42264391,47
AG8	DIKENALI	13	AG2	43542074,32
AG9	DIKENALI	8	Q2	39197839,9
AG10	DIKENALI	13	AG2	16112635,38
AH8	DIKENALI	1	AK1	39502691,85
AH9	DIKENALI	1	AK1	40025730,13
AH10	DIKENALI	1	AK1	39520944,3
AI8	DIKENALI	14	L1	40199906,05
AI9	DIKENALI	4	AM1	45249181,01
AI10	DIKENALI	8	Q2	36192412,61
AJ8	DIKENALI	3	C1	54445226,25
AJ9	DIKENALI	8	Q2	33069250,01
AJ10	DIKENALI	8	Q2	37102218,93

4.5.2.2 Identifikasi untuk 120 citra latih dan 120 citra uji

Pada pengujian untuk 120 citra latih dan 120 citra uji, terbentuk 24 *cluster* pada proses klasifikasi. Hasil klasifikasi untuk 120 citra latih dapat dilihat pada bagian lampiran 4.

Hasil pengujian identifikasi untuk citra uji dapat dilihat pada Tabel 4.2. Dari hasil pengujian 120 citra uji, sebanyak 5 citra tidak dikenali oleh sistem, 48 citra salah dalam identifikasi dan 67 citra tepat dalam identifikasi. Tingkat keakuratan yang didapatkan sebesar 58,26 %.

Tabel 4.2 Beberapa hasil identifikasi untuk 120 citra uji dengan 120 citra latih

Citra Uji	Status Pengenalan	Cluster	Hasil Identifikasi	Jarak Euclidean
A8	DIKENALI	20	N1	44997481,2
A9	DIKENALI	10	O1	45991646,03
A10	DIKENALI	1	Q1	31987319,29
AA8	DIKENALI	1	AA1	28960890,11
AA9	DIKENALI	1	AA1	24754826,07
AA10	DIKENALI	7	AA2	18895758,4
AB8	DIKENALI	7	AB2	24263615,78
AB9	DIKENALI	7	AB2	28173334,36
AB10	DIKENALI	7	AB2	24551434,38
AC8	DIKENALI	2	W1	35589017,89
AC9	DIKENALI	9	AC3	36848938,82
AC10	TIDAK DIKENALI	-	-	0
AD8	TIDAK DIKENALI	-	-	0
AD9	DIKENALI	6	AD2	39454986,83
AD10	DIKENALI	2	AD1	25007534,54
AE8	DIKENALI	11	AE1	31791716,44
AE9	DIKENALI	2	I1	32356205,06
AE10	DIKENALI	4	AF3	47461193,35
AF8	DIKENALI	4	AF3	32374757,1
AF9	DIKENALI	3	V3	37867231,63
AF10	DIKENALI	3	AE3	34791038,55
AG8	DIKENALI	13	AG2	36211482,25
AG9	TIDAK DIKENALI	-	-	0
AG10	DIKENALI	13	AG2	16185244,68
AH8	DIKENALI	4	AH1	36192726,8
AH9	DIKENALI	3	AE3	32868801,65
AH10	DIKENALI	3	AE3	30706921,05
AI8	DIKENALI	5	AI2	41032036,96

AI9	DIKENALI	3	AI3	46035723,27
AI10	DIKENALI	1	AN3	45940154,63
AJ8	DIKENALI	3	G2	31669068,15
AJ9	DIKENALI	24	L1	37721202,12
AJ10	DIKENALI	3	AE3	34331594,82

4.5.2.3 Identifikasi untuk 160 citra latih dan 120 citra uji

Pada pengujian untuk 160 citra latih dan 120 citra uji, citra latih diklasifikasi kedalam 25 *cluster*. Hasil klasifikasi untuk 160 citra latih dapat dilihat pada bagian lampiran 5.

Hasil pengujian identifikasi untuk citra uji dapat dilihat pada Tabel 4.3. Dari hasil pengujian 120 citra uji, sebanyak 3 citra tidak dikenali oleh sistem, 36 citra salah dalam identifikasi dan 81 citra tepat dalam identifikasi. Tingkat keakuratan yang didapatkan sebesar 69,23 %.

Tabel 4.3 Beberapa hasil identifikasi untuk 120 citra uji dengan 160 citra latih

Citra Uji	Status Pengenalan	Cluster	Hasil Identifikasi	Jarak Euclidean
A8	DIKENALI	3	A2	27487751,96
A9	DIKENALI	9	AB4	38975936,8
A10	DIKENALI	12	Q2	24917360,43
AA8	DIKENALI	1	D1	27403896,11
AA9	DIKENALI	1	D3	32167718,05
AA10	DIKENALI	3	AA1	13768891,82
AB8	DIKENALI	3	AB3	21527036,9
AB9	DIKENALI	3	AB3	25673782,52
AB10	DIKENALI	3	AB3	15454743,26
AC8	TIDAK DIKENALI	-	-	0
AC9	DIKENALI	6	AC2	30487378,61
AC10	DIKENALI	24	AC4	28148168,84
AD8	DIKENALI	1	W1	34897776,29
AD9	DIKENALI	2	AD1	25767585,12

AD10	DIKENALI	1	W1	30213451,68
AE8	DIKENALI	1	AE2	56389406,46
AE9	DIKENALI	1	AE2	67421692,8
AE10	DIKENALI	3	AB3	28669833,51
AF8	DIKENALI	3	AF1	32694557,65
AF9	DIKENALI	3	AF1	35017969,46
AF10	DIKENALI	3	AF1	36435177,77
AG8	DIKENALI	4	AG3	38701405,4
AG9	DIKENALI	3	X3	31593818,19
AG10	DIKENALI	4	AG4	23590013,4
AH8	DIKENALI	3	AH1	32901863,74
AH9	DIKENALI	3	AH1	37095821,52
AH10	DIKENALI	3	AH1	35624652,61
AI8	DIKENALI	1	AI4	30287498,04
AI9	DIKENALI	3	AB3	30997653,58
AI10	DIKENALI	1	AI3	39759771,96
AJ8	DIKENALI	1	AJ3	22970803,53
AJ9	DIKENALI	1	AJ3	32963643,08
AJ10	DIKENALI	3	AB3	29996272,66

4.5.2.4 Identifikasi untuk 200 citra latih dan 120 citra uji

Pada pengujian untuk 200 citra latih dan 120 citra uji, citra latih diklasifikasi kedalam 18 *cluster*. Hasil klasifikasi untuk 160 citra latih dapat dilihat pada bagian lampiran 6.

Hasil pengujian identifikasi untuk citra uji dapat dilihat pada Tabel 4.4. Dari hasil pengujian 120 citra uji, sebanyak 2 citra tidak dikenali oleh sistem, 34 citra salah dalam identifikasi dan 84 citra tepat dalam identifikasi. Tingkat keakuratan yang didapatkan sebesar 71,19 %.

Tabel 4.4 Beberapa hasil identifikasi untuk 120 citra uji dengan 200 citra latih

Citra Uji	Status Pengenalan	Cluster	Hasil Identifikasi	Jarak Euclidean
A8	DIKENALI	1	A2	29037811,6
A9	DIKENALI	1	A5	32257790,5
A10	DIKENALI	2	Z5	33118435,44
AA8	DIKENALI	1	AA5	27903221,23
AA9	DIKENALI	1	AA5	21019414,59
AA10	DIKENALI	1	AA1	11729530,51
AB8	DIKENALI	1	AB3	28681942,4
AB9	DIKENALI	1	AB1	29200601,78
AB10	DIKENALI	1	AB3	27311930,03
AC8	DIKENALI	5	AC3	25360081,08
AC9	DIKENALI	14	F5	45025782,79
AC10	DIKENALI	5	AC4	23942575
AD8	DIKENALI	4	AD4	25646522,56
AD9	TIDAK DIKENALI	-	-	0
AD10	DIKENALI	14	F5	32925398,11
AE8	DIKENALI	2	AE1	22510312,78
AE9	DIKENALI	10	I3	32978898,54
AE10	DIKENALI	1	AE3	20659770,24
AF8	DIKENALI	1	AF2	27087111,75
AF9	DIKENALI	3	K3	30262628,43
AF10	DIKENALI	3	K3	30993516,5
AG8	DIKENALI	1	A2	33208700,15
AG9	DIKENALI	3	AG5	20754749,54
AG10	DIKENALI	8	AG4	21523258,86
AH8	DIKENALI	1	AH1	29030892,61
AH9	DIKENALI	1	AH3	34506767,41
AH10	DIKENALI	1	AH1	33053792,01
AI8	DIKENALI	2	AI4	27504894,22

AI9	DIKENALI	1	AB4	32928231,84
AI10	DIKENALI	10	AM5	29616283
AJ8	DIKENALI	10	AJ5	19992507,64
AJ9	DIKENALI	3	AJ1	27926474,85
AJ10	DIKENALI	3	AJ4	43607163,9

4.6 Analisis Hasil

Dari hasil pengujian yang dilakukan maka dapat dilakukan beberapa analisa.

4.6.1 Analisis Proses Pembelajaran (Klasifikasi) ART-2

Pemilihan nilai *learning rate (α)* dan nilai *vigilance (ρ)* yang tepat mempengaruhi hasil klasifikasi dalam ART-2. *Learning rate (α)* adalah parameter yang menentukan kecepatan pembelajaran jaringan. *Learning rate* berpengaruh pada unit *cluster* pemenang hasil kompetisi pada lapis F2. Hal ini terlihat pada pembaruan bobot *top-down* dan *bottom-up* dalam persamaan 2.18. Dari persamaan tersebut dapat diketahui bahwa semakin besar harga *learning rate (α)* makin besar pula perubahan bobot pada setiap *epoch* sehingga jaringan akan memerlukan sedikit *epoch* dan cepat beresonansi. Sebaliknya bila harga parameter tersebut cukup kecil, maka jaringan akan cenderung membutuhkan lebih banyak *epoch* dalam mencapai kestabilan jaringan. *Cluster* klasifikasi yang terbentuk juga bervariasi terhadap nilai *learning*. Pada nilai *learning rate* tertentu jaringan agak mengalami kesulitan dalam mencapai kestabilan karena hasil operasi lapisan F1-F2 yang tidak sesuai sehingga mempengaruhi proses kompetisi yang terjadi sampai terjadi reset.

Parameter *vigilance* juga turut menentukan hasil klasifikasi. Pemilihan nilai *vigilance (ρ)* yang terlalu rendah menyebabkan jumlah cluster yang terbentuk sedikit. Hal ini disebabkan vektor-vektor input yang memiliki perbedaan yang cukup besar akan ditempatkan dalam *cluster* yang sama. Sebaliknya, pemilihan nilai *vigilance (ρ)* yang terlalu tinggi maka jumlah cluster yang terbentuk semakin banyak. Hal ini disebabkan jaringan akan menjadi sensitif sehingga menyebabkan jaringan mengaktifkan *cluster* baru untuk perbedaan vektor input yang kecil sekalipun. Dari hasil pengujian, nilai *vigilance* (ρ) > 0.97 menghasilkan pembelajaran jaringan yang

cukup baik karena jumlah *cluster* klasifikasi mendekati jumlah kelas vektor input sehingga setiap vektor citra wajah dapat diklasifikasi sesuai pemiliknya.

Nilai *learning rate* dan *vigilance* yang cukup tepat dalam menghasilkan pembelajaran jaringan yang baik, yaitu nilai (ρ) > 0.97 dan nilai (α) < 0.25

4.6.2 Analisis Proses Pengenalan (Identifikasi) ART-2

Keberhasilan sistem yang dirancang ditentukan oleh tingkat keakuratan identifikasi citra wajah yang dimasukan ke sistem. Dari identifikasi citra wajah menunjukkan bahwa sistem mampu memutuskan apakah citra wajah tersebut dapat dikenali (*recognize*) atau tidak dikenali (*unknown*). Persentase keakuratan identifikasi dapat dilihat pada tabel 4.5.

Tabel 4.5 Persentase Keakuratan Identifikasi

Jumlah citra latih	Jumlah yang teridentifikasi	Jumlah yang tidak teridentifikasi	Jumlah yang salah identifikasi	Persentase Keakuratan
80 (2 set)	117	3	57	51,28 %
120 (3 set)	115	5	48	58,26 %
160 (4 set)	117	3	36	69,23 %
200 (5 set)	118	2	34	71,19 %
Persentase Rata-Rata				62,49 %

Pada hasil pengujian terdapat citra yang salah identifikasi, yaitu wajah seseorang yang di identifikasi sebagai wajah orang lain dan citra yang tidak teridentifikasi, yaitu wajah yang tidak dikenali oleh sistem. Pada citra yang salah identifikasi, hal ini disebabkan karena pola vektor uji dan pola vektor pada *cluster* yang ditempati memiliki jarak yang lebih dekat dibandingkan antara pola vektor uji dengan pola vektor yang sebenarnya pada *cluster* yang diinginkan. Sedangkan pada citra yang tidak teridentifikasi, disebabkan karena nilai unit **r** dari hasil operasi pada lapisan F1-F2 tidak memenuhi syarat parameter *vigilance* sehingga citra akan ditolak oleh sistem.

Jumlah pemilihan sampel citra latih turut mempengaruhi tingkat keakuratan pengenalan. Semakin banyak citra yang dipelajari, maka semakin peka (adaptif) jaringan dalam melakukan proses identifikasi. Hal tersebut dikarenakan jarak pola-pola vektor

ciri citra wajah orang yang berbeda pada *cluster* hasil klasifikasi nilainya cukup besar sehingga perbedaan variasi antar individu semakin lebar.

Dari hasil analisa identifikasi diketahui bahwa sistem yang dirancang ini memiliki hasil identifikasi terbaik untuk pengujian 200 citra latih dengan tingkat akurasi sebesar 71,19% dan tingkat keakuratan rata-rata sebesar 62,49 %.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari hasil analisis terhadap pengujian yang dilakukan, maka dapat ditarik kesimpulan sebagai berikut :

1. Pengenalan wajah menggunakan ART-2 tidak memberikan tingkat akurasi yang lebih baik dari pada menggunakan ART-2A.
2. Tingkat keakuratan dari proses klasifikasi terbaik sebesar 71,19 % dan tingkat keakuratan rata-rata dari sistem pengenalan wajah sebesar 62,49 %.
3. Keberhasilan sistem dalam pengenalan citra wajah pada dasarnya sangat dipengaruhi oleh jarak antar pola-pola vektor ciri citra wajah yang dimasukan. Jika jarak antar pola wajah dari orang yang berbeda cukup dekat maka dapat terjadi kesalahan pengenalan citra.
4. Dengan penentuan kombinasi parameter-parameter jaringan yang sesuai, jaringan syaraf tiruan ART-2 dapat memberikan hasil klasifikasi yang optimal, sehingga dapat digunakan untuk pengenalan. Nilai parameter yang memberikan hasil yang optimal, yaitu parameter *vigilance* (ρ) > 0.97 dan parameter *learning* (α) < 0.25
5. Pemilihan sampel citra wajah pada proses pembelajaran dapat mempengaruhi tingkat akurasi proses pengenalan. Semakin banyak citra wajah yang dipelajari semakin baik sistem dapat melakukan proses pengenalan.

5.2 Saran

Pengembangan lebih lanjut yang dapat dilakukan dari skripsi ini antara lain :

1. Penambahan proses pengolahan citra sebelum proses ekstraksi fitur seperti penambahan proses deteksi tepi dan pengkontrasan, sehingga dapat menghasilkan fitur yang semakin baik.
2. Penggunaan jenis jaringan syaraf tiruan yang lain untuk diterapkan dalam sistem pengenalan ini, agar mendapatkan tingkat keakuratan yang lebih tinggi.

3. Proses pengenalan tidak hanya memperhatikan variasi pose tapi juga letak wajah.
4. Penggunaan ektraksi ciri dengan metode yang lain yang menghasilkan ciri yang berbeda antar citra-citra yang dimasukkan, sehingga memiliki jarak antar pola ciri yang cukup jauh.
5. Citra yang didukung lebih dikembangkan untuk mendukung citra berwarna dan format citra yang lain seperti BMP, PNG atau PGM.

DAFTAR PUSTAKA

Anton, H. dan Rorres C. 2005. *Elementary Linier Algebra, Applications Version*. John Wiley & Sons, Inc. New York.

Carpenter, G.A dan Grossberg, S. 1987. *ART 2: Self-organization of Stable Category Recognition Codes for Analog Input Patterns*. *Applied Optics*. 1987. 26 : 4919-4930.

Fausett, L. 1994. *Fundamentals of Neural Networks: Architectures, Algorithms and Applications, 2nd Edition*. Prentice-Hall International Inc. 218-287.

Grossberg, S. 1976. *Adaptive pattern classification and universal recording, I : Parallel development and coding of neural feature detectors*. *Biological Cybernetics*. 23 : 121-134.

Leon, S.J. 2001. *Aljabar Linier Dan Aplikasinya, Edisi Kelima*. Erlangga. Jakarta

Mahyabadi, M.P, Soltanizadeh, H., Shokouhi, Sh.B. 2006. *Facial Detection based on PCA and Adaptive Resonance Theory 2A Neural Network. The 2006 IJME –INTERTEC Conference (Proceedings)*.

Turk, M. dan Pentland, A. *Eigenfaces for Recognition*. *Journal of Cognitive Neuroscience*. 1991. 3 (1) : 71-86.

Silaen, A.M. 2006. *Pengenalan Citra Wajah Manusia Menggunakan Principle Component Analysis Dan Adaptive Resonance Theory (ART)*. STT Telkom. Bandung.

Smith, L.I. 2002. *A tutorial on Principal Component Analysis*. http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_component.pdf. Tanggal akses : 05-10-2010.

Tanaka T. dan Weitzenfeld. A. 2002. *The Neural Simulation Language : A System for Brain Modeling*. Weitzenfeld, A., Arbib, M., Alexander, A. MIT Press. 157-170.

LAMPIRAN

Lampiran 1. Citra Latih


A1 A2 A3 A4 A5


AA1 AA2 AA3 AA4 AA5


AB1 AB2 AB3 AB4 AB5


AC1 AC2 AC3 AC4 AC5


AD1 AD2 AD3 AD4 AD5

AE1 AE2 AE3 AE4 AE5

AF1 AF2 AF3 AF4 AF5

AG1 AG2 AG3 AG4 AG5

AH1 AH2 AH3 AH4 AH5

AI1 AI2 AI3 AI4 AI5

AJ1 AJ2 AJ3 AJ4 AJ5

AK1 AK2 AK3 AK4 AK5

AL1 AL2 AL3 AL4 AL5

AM1 AM2 AM3 AM4 AM5

AN1 AN2 AN3 AN4 AN5

B1 B2 B3 B4 B5

C1 C2 C3 C4 C5

D1 D2 D3 D4 D5

E1 E2 E3 E4 E5

F1 F2 F3 F4 F5

G1 G2 G3 G4 G5

H1 H2 H3 H4 H5

I1 I2 I3 I4 I5

J1 J2 J3 J4 J5

K1 K2 K3 K4 K5

L1 L2 L3 L4 L5

M1 M2 M3 M4 M5

N1 N2 N3 N4 N5

O1 O2 O3 O4 O5

P1 P2 P3 P4 P5

Q1 Q2 Q3 Q4 Q5

R1 R2 R3 R4 R5

S1 S2 S3 S4 S5

T1 T2 T3 T4 T5

U1 U2 U3 U4 U5

V1 V2 V3 V4 V5

W1 W2 W3 W4 W5

X1 X2 X3 X4 X5

Y1 Y2 Y3 Y4 Y5

Z1 Z2 Z3 Z4 Z5

Lampiran 2. Citra Uji

AH9 AH10 AI8 AI9 AI10

AJ8 AJ9 AJ10 AK8 AK9

AK10 AL8 AL9 AL10 AM8

AM9 AM10 AN8 AN9 AN10

B8 B9 B10 C8 C9

C10 D8 D9 D10 E8

E9 E10 F8 F9 F10

G8 G9 G10 H8 H9

H10 I8 I9 I10 J8

J9 J10 K8 K9 K10

L8 L9 L10 M8 M9

M10 N8 N9 N10 O8

O9 O10 P8 P9 P10

Q8 Q9 Q10 R8 R9

R10 S8 S9 S10 T8

T9 T10 U8 U9 U10

V8 V9 V10 W8 W9

W10 X8 X9 X10 Y8

Y9 Y10 Z8 Z9 Z10

Lampiran 3. Hasil klasifikasi untuk 80 citra latih

ρ = 0.97	α = 0.25
Classification Pattern	
Cluster 1 -> AA1, AN1, AN2, D2, E1, E2, I2, K1, M1, M2, O2, P2, T1, X1, Y1 Cluster 2 -> G2, AK1, AK2, V2, AM1, AF2 Cluster 3 -> D1, AI1, U1, AD2, AG1, C1 Cluster 4 -> AC1, F1 Cluster 5 -> L1, Q1, Q2, AJ1, Z1, Z2 Cluster 6 -> N2, AH1, AB2 Cluster 7 -> R1, T2 Cluster 8 -> B1, B2, S1, S2 Cluster 9 -> U2, AI2 Cluster 10 -> Y2, AB1, A1 Cluster 11 -> H1, P1, H2 Cluster 12 -> AE1, AL1, AL2, AD1, W1 Cluster 13 -> AM2, X2 Cluster 14 -> C2, J2, I1 Cluster 15 -> AE2, V1 Cluster 16 -> AG2, F2 Cluster 17 -> AA2, L2 Cluster 18 -> G1 Cluster 19 -> R2 Cluster 20 -> N1, AH2 Cluster 21 -> W2 Cluster 22 -> AJ2, A2 Cluster 23 -> AC2 Cluster 24 -> J1 Cluster 25 -> K2 Cluster 26 -> AF1 Cluster 27 -> O1	
Jumlah Cluster	27 Cluster
Banyak Epoch	1000 Epoch

Lampiran 4. Hasil klasifikasi untuk 120 citra latih

ρ = 0.97	α = 0.25
Classification Pattern	
Cluster 1 -> AA1, AA3, AN2, AN3, D1, D3, E1, M2, P2, Q1, R3 Cluster 2 -> AL1, AL2, AL3, U2, U3, W1 Cluster 3 -> R2, M1, P3, L2, R1, Z3 Cluster 4 -> AK2, B2, T2, T3, AA2, A3, AB2, AK1 Cluster 5 -> AK3, AI3, AE2, V1, V2, S1, AE3 Cluster 6 -> F1, W3, F3 Cluster 7 -> G3, Q3, AJ1 Cluster 8 -> P1, O1, O3 Cluster 9 -> AF2, S2, S3, AF3 Cluster 10 -> Y2, C1 Cluster 11 -> K1, AB1, AD1 Cluster 12 -> K2, M3, K3, B1, B3 Cluster 13 -> AC2, AC3, W2 Cluster 14 -> O2, I3, Y1, H1, Y3, I2 Cluster 15 -> AM2, A1, Cluster 16 -> AF1, AG3 Cluster 17 -> Z1, C3 Cluster 18 -> N2, N3 Cluster 19 -> X2, T1 Cluster 20 -> H2, AJ2, AE1 Cluster 21 -> AH1, AH2, AH3 Cluster 22 -> AD3, I1, J3 Cluster 23 -> AB3, AN1 Cluster 24 -> AJ3, G2 Cluster 25 -> A2, N1, X1 Cluster 26 -> E2, Z2, C2 Cluster 27 -> AG1 Cluster 28 -> AM3, H3 Cluster 29 -> E3, L3 Cluster 30 -> Q2, D2 Cluster 31 -> U1, J2 Cluster 32 -> AD2, F2 Cluster 33 -> AI2, AI1 Cluster 34 -> J1 Cluster 35 -> V3, AM1 Cluster 36 -> G1 Cluster 37 -> AG2 Cluster 38 -> AC1 Cluster 39 -> L1, X3	
Jumlah Cluster	39 Cluster
Banyak Epoch	1000 Epoch

Lampiran 5. Hasil klasifikasi untuk 160 citra latih

ρ = 0.97	α = 0.25
Classification Pattern	
Cluster 1 -> A2, AA1, AA2, AA3, AA4, AB3, AH1, AK1, K1, K2, K3, P1, P2, R1, R2, T2, V2, X1, X2, X3 Cluster 2 -> AE2, AN4, L2, L4, M1, M2, C4, D1, Q2, Q4, D2, D3, E1, U2, E2, E3, E4, AI4, Y1, Z1, Z2, Z3 Cluster 3 -> C1, R3, C2, AJ3, A3, W1, G2, AL4, P3, AJ1, Q3, D4, L1, Z4 Cluster 4 -> I3, I4, AD1, U4, AE1, I1, I2 Cluster 5 -> AH2, N4, AB4, S1, S3, AM4 Cluster 6 -> AG4, AG2, V4, AG3 Cluster 7 -> AN2, AN3, AN1, B1, B3 Cluster 8 -> AF4, AK2, AF3, AH3 Cluster 9 -> G4, AC2, C3 Cluster 10 -> O2, P4, O3, O4, AM2, O1 Cluster 11 -> AJ4, G1, B2, G3 Cluster 12 -> AF1, L3, H1 Cluster 13 -> T1, A1, T3, Q1 Cluster 14 -> F2, F3, F1 Cluster 15 -> AB1, AK3, AB2, K4 Cluster 16 -> AL2, AI2, AI1, Y2 Cluster 17 -> H3, AI3 Cluster 18 -> AL3, AD2, AD4 Cluster 19 -> N3, N2 Cluster 20 -> W2, AC4 Cluster 21 -> AE3, AE4 Cluster 22 -> AH4, AG1 Cluster 23 -> M3, M4, H4, Y3 Cluster 24 -> AL1, AJ2 Cluster 25 -> S4, S2 Cluster 26 -> W3 Cluster 27 -> U1, U3 Cluster 28 -> AM1, H2 Cluster 29 -> J2, J3 Cluster 30 -> T4, B4 Cluster 31 -> AC1, W4 Cluster 32 -> AF2 Cluster 33 -> AD3, R4 Cluster 34 -> F4, Y4 Cluster 35 -> AM3 Cluster 36 -> V3 Cluster 37 -> N1 Cluster 38 -> J4 Cluster 39 -> A4	

Cluster 40 -> V1, X4 Cluster 41 -> J1 Cluster 42 -> AC3 Cluster 43 -> AK4	
Jumlah Cluster	43 Cluster
Banyak Epoch	1000 Epoch

Lampiran 6. Hasil klasifikasi untuk 200 citra latih

ρ = 0.97	α = 0.25
Classification Pattern	

Cluster 1 -> A2, A3, A5, AA1, AA2, AA3, AA4, AA5, AB1, AB2, AB4, AB5, AE2, AE3, AF2, AF3, AF5, AH2, AH3, AH4, AH5, AK1, AK2, AK4, AK5, M3, M4, M5, N1, P1, P4, P5, S1, S3, S4, S5, V1, V2, V3, V4, V5, X1, X2
Cluster 2 -> AL4, AL5, U1, U3, U4, U5, AN1, AN5, I3, Q3, Q4, W1, AJ3, AL2
Cluster 3 -> H2, H4, AG5, K1, K3, T1, T2, T4, T5, AM2, AJ1, A1, B1, O1, O3, O4, B3, P2, B4, B5, H1, X4, X5
Cluster 4 -> AN4, U2, I2, AE1, C1, C4, L4, L5, E1, E2, E3, E4, E5, AJ2, AI4, Y1, Y2, Y3, Y4, Z1, Z2, Z3, Z4, Z5
Cluster 5 -> W4, P3, D3, X3, F4
Cluster 6 -> O5, AC2, AC5, F3, O2, AM5, AD1
Cluster 7 -> R4, D5, Q2, AN2, D4, Q5, R1, T3, R2
Cluster 8 -> I1, I4, I5
Cluster 9 -> AF4, N3, N4, N5, AK3, AE4, AE5
Cluster 10 -> AL1, J1, J3, R3, J5, AD3, D1, AL3
Cluster 11 -> G5, G2
Cluster 12 -> J4, L2
Cluster 13 -> S2, H5, H3, Q1, AM4
Cluster 14 -> AC3, AC4, AG1
Cluster 15 -> AI3, AI5, K2
Cluster 16 -> G4, C3, C2
Cluster 17 -> M1, M2
Cluster 18 -> G3, G1, B2
Cluster 19 -> AN3
Cluster 20 -> K5, K4
Cluster 21 -> W3, W5
Cluster 22 -> AJ4, AG3
Cluster 23 -> J2, AC1
Cluster 24 -> C5, AM3
Cluster 25 -> AJ5, R5, F1, F2
Cluster 26 -> AG2, AD2, AG4
Cluster 27 -> F5, L1
Cluster 28 -> D2, AB3, AH1, AF1, AM1
Cluster 29 -> AI2, AI1
Cluster 30 -> Y5
Cluster 31 -> L3
Cluster 32 -> W2
Cluster 33 -> A4
Cluster 34 -> AD4, AD5
Cluster 35 -> N2

Jumlah Cluster	35 Cluster
Banyak Epoch	1000 Epoch

Lampiran 7. Hasil klasifikasi untuk nilai *Learning Rate* (α) yang berbeda

α = 0,1	α = 0,2	α = 0,3	α = 0,4	α = 0,5	α = 0,6	α = 0,7	α = 0,8	α = 0,9
Cluster 1 -> AD1, AJ1, AN1, I1, W1 Cluster 2 -> P1, S1, V1, O1 Cluster 3 -> Q1, T1 Cluster 4 -> L1, C1, Z1 Cluster 5 -> Y1, F1 Cluster 6 -> AB1, AK1 Cluster 7 -> G1, AE1, A1 Cluster 8 -> AH1, B1 Cluster 9 -> E1, D1 Cluster 10 -> AF1 Cluster 11	Cluster 1 -> AE1, AJ1, AN1, W1 Cluster 2 -> Q1, C1, Z1 Cluster 3 -> P1, O1 Cluster 4 -> AC1, T1 Cluster 5 -> K1, V1 Cluster 6 -> F1 Cluster 7 -> AH1, X1, B1 Cluster 8 -> AB1, AK1, AA1 Cluster 9 -> I1, AD1 Cluster 10 -> E1, L1 Cluster 11 -> AG1, U1	Cluster 1 -> AD1, AJ1, AN1, I1, W1 Cluster 2 -> L1, C1, Z1 Cluster 3 -> O1, P1 Cluster 4 -> Q1, T1 Cluster 5 -> AB1, AK1, AA1 Cluster 6 -> X1, B1 Cluster 7 -> V1, K1 Cluster 8 -> G1, AE1, A1 Cluster 9 -> F1, N1 Cluster 10 -> AF1, H1 Cluster 11 -> E1, D1	Cluster 1 -> AD1, AJ1, AN1, I1, W1 Cluster 2 -> Q1, Z1 Cluster 3 -> S1, V1, P1, O1 Cluster 4 -> AH1, X1, B1 Cluster 5 -> L1, E1, D1 Cluster 6 -> AK1, AB1, AA1 Cluster 7 -> F1, N1 Cluster 8 -> H1, AF1 Cluster 9 -> Y1 Cluster 10 -> T1, M1 Cluster 11	Cluster 1 -> AD1, AJ1, AN1, I1, W1 Cluster 2 -> Q1, D1, Z1 Cluster 3 -> P1, O1 Cluster 4 -> T1, M1, AA1 Cluster 5 -> AK1, AB1, S1 Cluster 6 -> Y1, F1 Cluster 7 -> AM1, X1 Cluster 8 -> E1, L1 Cluster 9 -> AG1, U1 Cluster 10 -> H1, AF1 Cluster 11 -> J1	Cluster 1 -> A1, AE1, G1 Cluster 2 -> O1, P1 Cluster 3 -> AD1, I1, AJ1, AN1, W1 Cluster 4 -> T1, Q1, Z1 Cluster 5 -> S1, V1 Cluster 6 -> R1, D1 Cluster 7 -> U1, B1 Cluster 8 -> AB1, AA1, AK1 Cluster 9 -> C1, L1 Cluster 10 -> X1, AC1 Cluster 11 -> Y1, F1	Cluster 1 -> AD1, AE1, AJ1, G1, W1 Cluster 2 -> P1, O1 Cluster 3 -> L1, C1, Z1 Cluster 4 -> T1 Cluster 5 -> Q1 Cluster 6 -> X1, AC1 Cluster 7 -> AK1, S1, AB1 Cluster 8 -> D1, AN1, I1 Cluster 9 -> J1, M1 Cluster 10 -> R1 Cluster 11 -> AI1	Cluster 1 -> AB1, AK1, S1 Cluster 2 -> AD1, AN1, I1, AJ1, W1 Cluster 3 -> O1, P1 Cluster 4 -> C1, L1 Cluster 5 -> F1, Y1 Cluster 6 -> AC1, K1 Cluster 7 -> D1, T1 Cluster 8 -> H1, AF1 Cluster 9 -> E1, M1 Cluster 10 -> B1, Q1 Cluster 11 -> A1, G1 Cluster 12 -> J1	Cluster 1 -> AB1, AK1, S1 Cluster 2 -> AD1, AN1, I1, AJ1, W1 Cluster 3 -> O1, P1 Cluster 4 -> C1, L1 Cluster 5 -> F1, Y1 Cluster 6 -> AC1, K1 Cluster 7 -> D1, T1 Cluster 8 -> H1, AF1 Cluster 9 -> E1, M1 Cluster 10 -> B1, Q1 Cluster 11 -> A1, G1 Cluster 12 -> J1

-> M1, J1 Cluster 12 -> AC1, X1 Cluster 13 -> AI1 Cluster 14 -> AM1 Cluster 15 -> R1 Cluster 16 -> U1, AG1 Cluster 17 -> N1 Cluster 18 -> AA1 Cluster 19 -> AL1 Cluster 20 -> H1 Cluster 21 -> K1	Cluster 12 -> AF1, H1 Cluster 13 -> M1, J1 Cluster 14 -> Y1 Cluster 15 -> R1, D1 Cluster 16 -> N1 Cluster 17 -> A1, G1 Cluster 18 -> AI1 Cluster 19 -> S1 Cluster 20 -> AL1 Cluster 21 -> AM1	Cluster 12 -> Y1, AC1 Cluster 13 -> AM1, AH1 Cluster 14 -> AI1 Cluster 15 -> U1, AG1 Cluster 16 -> J1 Cluster 17 -> R1 Cluster 18 -> S1, M1 Cluster 19 -> AL1	-> K1 Cluster 12 -> U1, AG1 Cluster 13 -> G1, A1, AE1 Cluster 14 -> AM1 Cluster 15 -> J1 Cluster 16 -> C1, R1 Cluster 17 -> AI1 Cluster 18 -> AC1 Cluster 19 -> AL1	Cluster 12 -> K1, V1 Cluster 13 -> AI1, C1 Cluster 14 -> G1, AE1, A1 Cluster 15 -> AH1 Cluster 16 -> R1 Cluster 17 -> AC1 Cluster 18 -> B1 Cluster 19 -> N1 Cluster 20 -> AL1	Cluster 12 -> AF1, H1 Cluster 13 -> AG1 Cluster 14 -> J1, M1 Cluster 15 -> AI1 Cluster 16 -> AM1, AH1 Cluster 17 -> E1 Cluster 18 -> AL1 Cluster 19 -> N1 Cluster 20 -> K1	Cluster 12 -> V1, K1 Cluster 13 -> A1, B1 Cluster 14 -> AG1 Cluster 15 -> AH1, AM1 Cluster 16 -> H1, AF1 Cluster 17 -> U1 Cluster 18 -> Y1 Cluster 19 -> E1 Cluster 20 -> N1 Cluster 21 -> AA1 Cluster 22 -> AL1 Cluster 23 -> F1	Cluster 13 -> V1 Cluster 14 -> AI1 Cluster 15 -> U1 Cluster 16 -> AH1, X1, AM1 Cluster 17 -> R1 Cluster 18 -> AE1 Cluster 19 -> N1 Cluster 20 -> Z1 Cluster 21 -> AL1 Cluster 22 -> AG1 Cluster 23 -> AA1	Cluster 13 -> V1 Cluster 14 -> AI1 Cluster 15 -> U1 Cluster 16 -> AH1, X1, AM1 Cluster 17 -> R1 Cluster 18 -> AE1 Cluster 19 -> N1 Cluster 20 -> Z1 Cluster 21 -> AL1 Cluster 22 -> AG1 Cluster 23 -> AA1
Jumlah Cluster								
21	21	19	19	20	20	23	23	23
Banyak epoch								
1000	1000	1000	929	770	628	555	482	482

Lampiran 8. Hasil klasifikasi untuk nilai *Vigilance* (ρ) yang berbeda

ρ = 0,91	ρ = 0,92	ρ = 0,93	ρ = 0,94	ρ = 0,95	ρ = 0,96	ρ = 0,97	ρ = 0,98	ρ = 0,99
Cluster 1 -> A1, AA1, AB1, AC1, AD1, AE1, AF1, AG1, AH1, AJ1, AK1, AM1, AN1, B1, D1, F1, G1, H1, I1, K1, M1, N1, O1, P1, Q1, R1, S1, T1, U1, V1, W1, X1, Z1 Cluster 2 -> AL1, E1, J1, AI1, L1, Y1, C1	Cluster 1 -> A1, AA1, AB1, AC1, AD1, AE1, AG1, AH1, AJ1, AK1, AM1, AN1, B1, C1, D1, E1, F1, G1, I1, K1, L1, M1, O1, P1, Q1, R1, S1, T1, U1, V1, W1, X1, Y1, Z1 Cluster 2 -> AL1, J1, AF1, AI1, H1, N1	Cluster 1 -> A1, AA1, AB1, AC1, AD1, AE1, AG1, AH1, AJ1, AK1, AN1, B1, C1, D1, E1, G1, I1, K1, L1, M1, O1, P1, Q1, S1, T1, U1, V1, W1, X1, Z1 Cluster 2 -> R1, AM1, H1, AF1 Cluster 3 -> AI1, AL1, F1, N1, Y1, J1	Cluster 1 -> A1, AC1, AD1, AE1, AJ1, AN1, D1, F1, G1, I1, Q1, T1, U1, W1 Cluster 2 -> C1, R1, AI1, E1, L1, Y1 Cluster 3 -> AL1, N1, AG1, J1 Cluster 4 -> O1, P1, AK1, AB1, S1, K1, AA1, V1, M1, AF1, Z1 Cluster 5 -> H1 Cluster 6 -> B1, X1, AM1, AH1	Cluster 1 -> A1, AD1, AE1, AJ1, AN1, B1, D1, G1, I1, Q1, T1, U1, W1 Cluster 2 -> L1, M1, C1, AK1, AB1, AA1 Cluster 3 -> AF1, H1, R1 Cluster 4 -> N1, F1, AG1 Cluster 5 -> AI1, AL1, E1, Y1 Cluster 6 -> AH1, AM1, O1, P1, X1, AC1 Cluster 7 -	Cluster 1 -> A1, AD1, AE1, AJ1, AN1, D1, G1, I1, Q1, T1, W1, Z1 Cluster 2 -> AA1, S1, AK1, AB1, M1 Cluster 3 -> AH1, K1, AC1, O1, X1, P1 Cluster 4 -> AI1, E1, L1, Y1, C1 Cluster 5 -> H1, AF1 Cluster 6 -> AG1, U1 Cluster 7 -> R1, B1 Cluster 8 -> F1, J1 Cluster 9 -> AM1, V1	Cluster 1 -> AD1, AE1, AJ1, AN1, D1, I1, Q1, W1 Cluster 2 -> AK1, M1, AB1, AA1 Cluster 3 -> O1, P1, K1, AC1, X1 Cluster 4 -> C1, E1, L1, Z1 Cluster 5 -> U1, AG1 Cluster 6 -> H1, AF1 Cluster 7 -> Y1, F1 Cluster 8 -> B1, AM1, AH1 Cluster 9 -> J1 Cluster 10	Cluster 1 -> AD1, AJ1, AN1, I1, W1 Cluster 2 -> P1, S1, V1, O1 Cluster 3 -> AH1, B1, X1 Cluster 4 -> C1, L1 Cluster 5 -> AK1, AB1 Cluster 6 -> AC1, F1 Cluster 7 -> D1, Q1, Z1 Cluster 8 -> AE1, T1 Cluster 9 -> K1 Cluster 10 -> E1 Cluster 11	Cluster 1 -> AJ1, AN1 Cluster 2 -> Q1 Cluster 3 -> D1, Z1 Cluster 4 -> O1, P1 Cluster 5 -> AB1, AK1 Cluster 6 -> AC1 Cluster 7 -> V1 Cluster 8 -> I1 Cluster 9 -> B1, AH1 Cluster 10 -> L1, C1 Cluster 11 -> M1 Cluster 12 -> R1 Cluster 13 -> U1

				> J1 Cluster 8 -> K1, S1, V1, Z1	Cluster 10 -> AL1, N1	-> A1, G1 Cluster 11 -> N1 Cluster 12 -> AL1, AI1 Cluster 13 -> S1, V1 Cluster 14 -> R1 Cluster 15 -> T1	-> AF1, H1 Cluster 12 -> M1, AA1 Cluster 13 -> AG1 Cluster 14 -> AL1 Cluster 15 -> U1 Cluster 16 -> J1 Cluster 17 -> AM1 Cluster 18 -> R1 Cluster 19 -> N1 Cluster 20 -> AI1 Cluster 21 -> A1, G1 Cluster 22 -> Y1	Cluster 14 -> AE1, F1 Cluster 15 -> AF1 Cluster 16 -> AA1 Cluster 17 -> A1, G1 Cluster 18 -> X1 Cluster 19 -> Y1 Cluster 20 -> AL1 Cluster 21 -> N1 Cluster 22 -> E1 Cluster 23 -> S1 Cluster 24 -> W1 Cluster 25 -> J1 Cluster 26 -> AM1 Cluster 27 -> K1 Cluster 28 -> AG1

								Cluster 29 -> AI1 Cluster 30 -> H1 Cluster 31 -> T1 Cluster 32 -> AD1
Jumlah Cluster								
2	2	3	6	8	10	15	22	32
Banyak epoch								
227	256	358	1000	1000	1000	1000	1000	1000