2. TEORI PENUNJANG

2.1. Pengenalan Wajah

Proses pengenalan wajah yang dilakukan oleh komputer tidak semudah dan secepat dibandingkan dengan proses pengenalan yang dilakukan oleh manusia. Manusia dengan mudah dapat mengenali wajah seseorang dengan sangat cepat tanpa rasanya harus berpikir. Manusia juga tidak terpengaruh oleh orientasi wajah orang tersebut, misalnya orang tersebut dalam keadaan agak menoleh, menunduk, menengadah asal dalam batas-batas yang masih dapat dilihat. Sedangkan komputer selain lambat dalam pengenalan, juga kesulitan pada orientasi wajah yang berlainan, pencahayaan, latar belakang yang berbeda, potongan rambut, kumis atau jenggot, kacamata atau tidak dan lain sebagainya. Memang otak manusia lebih memiliki keuntungan dalam mengatasi masalah di mana aturan eksplisit tidak dapat dengan mudah diformulasikan, sedangkan komputer mempunyai keuntungan pada bidang seperti matematika dimana aturan-aturan mudah diformulasikan[1].

Oleh karena itu banyak dilakukan penelitian untuk mencari algoritma-algoritma yang tepat bagi komputer agar dapat mengenali suatu wajah yang diinputkan dengan memperhatikan faktor kecepatan dan akurasinya.

2.1.1. Berbagai Bidang tentang Penelitian Mengenai Wajah

Banyak penelitian yang dilakukan oleh para ahli komputer untuk menganalisa wajah orang. Penelitian tentang wajah ini terdiri dari beberapa bidang. Misalnya penelitian untuk pendeteksian wajah, maksudnya penelitian dilakukan untuk mencari lokasi wajah yang ada dalam *image*. Jadi dimasukkan sebuah foto ke dalam program komputer melalui *scanner*, dan nanti program akan menentukan lokasi-lokasi mana saja yang terdapat wajah.

---

[1] Bharath, Ramachandran dan Drosen, James, Neural Network Computing, (McGraw-Hill,Inc,1994), p.xv

Bidang yang lain yaitu pencarian *feature* dari wajah. Yang dimaksud *feature* dari wajah adalah hidung, mata, alis, telinga, mulut, dan lain sebagainya. Jadi program akan menerima *input* wajah gambar, dan setelah diproses akan menghasilkan lokasi-lokasi mana saja yang terdapat *feature-feature* tersebut.

Ada juga penelitian tentang ekspresi wajah. Dengan *training* yang cukup, maka program akan mengenali ekspresi *input testing* berupa wajah gambar yang dimasukkan, apakah wajah tersebut marah, sedih, tertawa dan lain sebagainya.

Selain pengklasifikasian ekspresi, para ahli juga berusaha mengklasifikasikan jenis kelamin. Program akan dapat mengenali jenis kelamin dari foto orang yang diinputkan ke dalamnya.

Kemudian juga dicari algoritma agar komputer dapat mengenali wajah *testing* yang diinputkan ke dalamnya berdasarkan wajah *training* yang telah disimpan. Pengenalan dilakukan dengan membandingkan dengan semua wajah *training* yang telah ada di *database*, dan dicari yang paling mirip.

Ada juga penelitian untuk menentukan posisi wajah *testing*, misalnya wajah *testing* yang diinputkan apakah sedang menoleh ke kiri, atau ke kanan, atau sedang memandang lurus.

2.1.2. Contoh Penelitian Mengenai Wajah

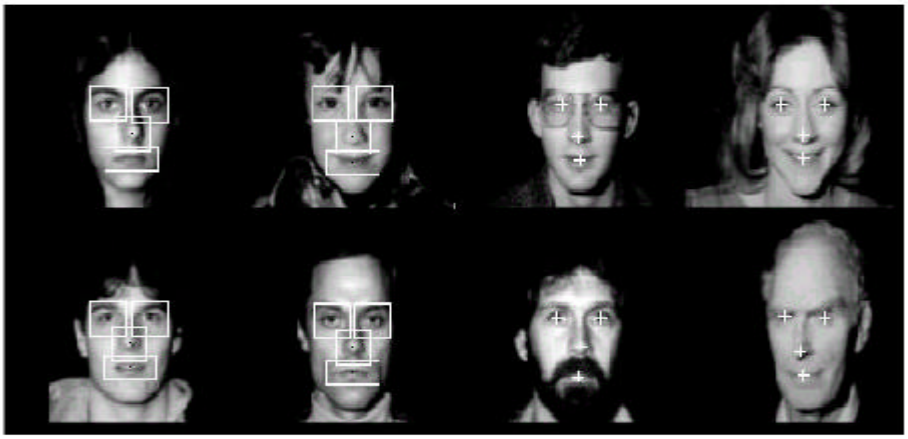
Baback Moghaddam[2] membuat program pengenalan wajah yang cukup komplit, yaitu ada tahap segmentasi, tahap pencarian *feature*, tahap normalisasi dan tahap terakhir adalah pengenalan wajah.. Ada beberapa tahap yang dilalui dalam program pengenalan wajahnya, yang pertama tahap segmentasi untuk mencari dan memisahkan wajah oval wajah dari latar belakangnya. Kemudian setelah wajah oval dari wajah ditemukan, akan dilakukan pencarian *feature* untuk menentukan letak mata, hidung, dan lain-lain, dan dilakukan rotasi berdasarkan *feature-feature* tersebut. Rotasi yang dilakukan mempunyai tujuan agar wajah oval dapat menjadi tegak lurus, tidak dalam keadaan miring ke kiri atau ke kanan.

[2] Moghaddam, Baback, F*ace Recognition*, http://www-white.media.mit.edu/vismod/demos/facerec/index.html , MIT.

Tahap keempat adalah pengenalan dari wajah *testing* berdasarkan dari wajah *training* yang telah ada di *database*.

Sami Romdhani[3] membuat penelitian untuk pengenalan wajah dengan menggunakan metode *Principal Component Analysis* (PCA). Dengan menggunakan *input* gambar dari *Manchester Face database* dan metode PCA untuk mereduksi dimensi data, maka pengenalan yang dilakukan menghasilkan prosentase kebenarannya 91%. Selain mengenali wajah, Sami Romdhani juga melakukan klasifikasi *sex* atau jenis kelamin dari wajah yang diinputkan.

Pentland, Moghaddam, dan Starnet[4] membuat pengenalan wajah dengan metode *view-based* dan *modular eigenspaces*. Pengenalan wajah menghasilkan akurasi 95%, dan orientasi wajah mempunyai *standard* deviasi 15°, maksudnya tingkat kemiringan orientasi wajah bisa mencapai 15°. Selain melakukan pengenalan wajah, juga dilakukan penelitian tentang pencarian *features* berdasarkan *eigenfeatures* yang akan menghasilkan *eigeneyes*, *eigennoses* dan *eigenmouths*.



Gambar 2.1. Pendeteksian *Feature* oleh Pentland, Maghaddam, Starnet

[3] Romdhani, Sami, *Face Recognition Using Principal Component Analysis*, http://www.elec.gla.ac.uk/~romdhani.

[4] Pentland A., Moghaddam B., Starner T., *View-Based and Modular Eigenspaces for Face Recognition,IEEE Conf. on Computer Vision & Pattern Recognition*, Seattle, WA, July 1994

Raphael Cendrillon[5] membuat pengenalan wajah dengan *eigenfaces*. Pertama kali dilakukan algoritma untuk mendeteksi adanya wajah dalam suatu gambar dan kemudian dilakukan segmentasi terhadap gambar. Setelah itu dilakukan pencarian *feature*, antara lain mata. Dengan penemuan letak mata, dapat diukur sudut kemiringan, sehingga dapat mengetahui wajah yang diinputkan miring atau tidak. Hal ini berguna untuk menormalisasikan wajah sehingga wajah yang asalnya miring akan dirotasi sehingga menjadi tegak. Setelah itu akan dicari *eigenface*-nya untuk dibandingkan dan dicari wajah termiripnya pada wajah *training*. Cendrillon menjelaskan bahwa teori *eigenfaces* sensitif terhadap reduksi skala kurang dari 88% dan rotasi atau kemiringan wajah yang lebih dari 10°.

a) **b)** **c)**

Gambar 2.2. Pengenalan Wajah Oleh Raphael Cendrillon.a)PendeteksianWajah, b)Pencarian *Feature*, c)Normalisasi Wajah Berdasarkan *Feature*

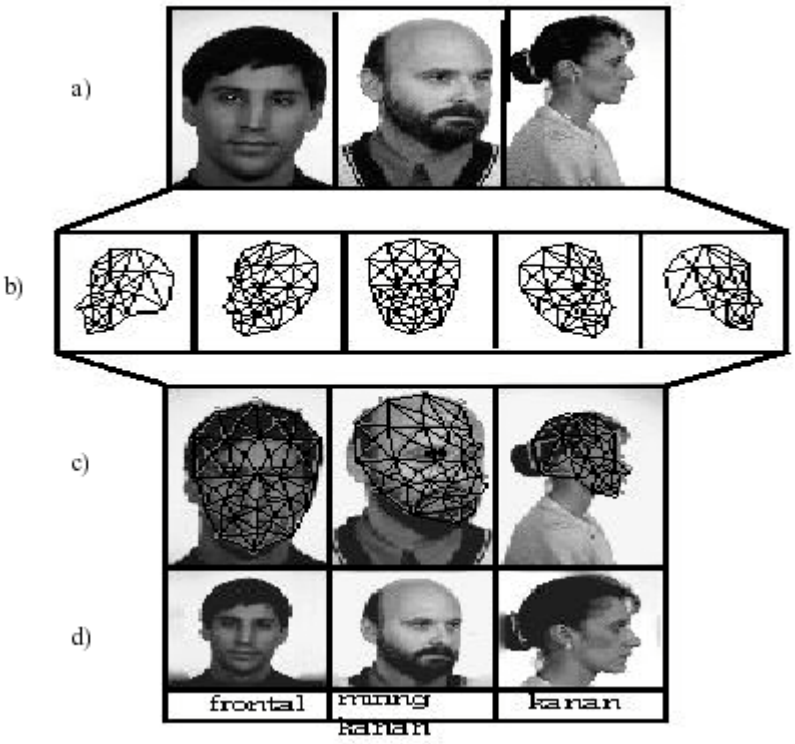
Pendeteksian wajah dapat mengenali beberapa jumlah dan lokasi dari wajah yang ada pada *image* yang diinputkan ke dalamnya. Jadi inputannya tidak hanya seperti pada gambar 2.2. a, tapi bisa merupakan gambar yang besar seperti pada gambar 2.3.

[5] Cendrillon, Raphael, *Face Recognition Using Eigenfaces*, http://student.uq.edu.au/~s341268/index.html, degree thesis, University of Queensland, Australia, Department of Computer Science and Electrical Engineering, 1999

Gambar 2.3. Pendeteksian Lokasi Semua Wajah Oval yang ada pada Sebuah *Image*

Kruger, Potzch,Maurer dan Rinne[6] dari Ruhr-Universitat Bochum, Institut Neuroinformatik di Jerman melakukan penelitian tentang estimasi dari posisi wajah dan pose dengan *graphs labeling*. Penelitian ini akan melakukan mencari posisi wajah dan mensegmentasi, kemudian akan menentukan pose wajah. Yang dimaksudkan pose wajah yaitu arah dari wajah, apakah wajah itu menoleh ke kiri atau ke kanan dan sebagainya. Klasifikasi pose berdasarkan lima kategori, yaitu menoleh ke kiri, menoleh ke kiri sebagian, frontal (menghadap ke depan), menoleh ke kanan sebagian, dan menoleh ke kanan. Jadi setiap wajah *testing* yang diinputkan akan dicari *feature*-nya terlebih dahulu, kemudian *feature* yang ditemukan dibandingkan dengan *template-template feature* dari lima kategori. *Template* yang paling mirip dengan *feature*-nya adalah pose dari *testing* wajah tersebut. Untuk lebih jelasnya dapat dilihat gambar 2.4.

[6] Norbert Kruger, Michael Potzch, Thomas Maurer, Michael Rinne, *Estimation of Face Position and Pose with Labeled Graphs*, Ruhr-Universitat Bochum



Gambar 2.4. Sistem Pendeteksian Wajah dan Pose.
a) Gambar *Input* dengan beberapa variasi. b) *Graph* yang merepresentasikan 5 pose. c) Gambar *Input* dengan *Grid*. d) Hasil *Output* Pose yang Termirip

Samaria dan Fallside[7], dari *Cambridge University*, membuat program untuk mengenali wajah dan mengekstrasi *feature* dengan *Hidden Markov models.* Wajah akan diperlukan sebagai obyek dua dimensi dan model ini akan otomatis mengekstrasi *statistical facial features*. *Features* ini dibagi menjadi 5 bagian, masing-masing merepresentasikan *feature band*. Penelitian ini mengasumsikan bahwa semua wajah yang dimasukkan dalam posisi frontal, atau memandang ke depan, sehingga tidak bisa dimasukkan wajah dalam keadaan menoleh. 5 bagian itu adalah dahi, mata, hidung, mulut dan dagu. Pencahayaan dan latar belakang dibuat sama. Penelitian ini menyimpulkan bahwa penggunaan *hidden markov* model ini dapat mengatasi variasi pada *facial feature*, tapi harus pada perubahan orientasi kepala yang kecil. Ini adalah problem yang akan ditemukan pada semua pengenalan wajah berbasis *template*. Pada gambar 2.5 dapat dilihat beberapa *training* wajah yang agak berbeda ekspresi wajahnya dan *feature band* yang dihasilkan.

[7] Samaria, F. dan Fallside, F., *Face Identification and Feature Extraction using Hidden Markov Models*, research report, Cambridge University, United Kingdom.

Gambar 2.5. *Training* data dan *Feature Band* dari Penelitian Pengidentifikasian Wajah dan Ekstrasi *Feature* dengan *Hidden Markov Models*

2.1.3. Pengenalan Wajah Sebagai Produk Komersial

Penelitian mengenai wajah bukan hanya sebagai produk penelitian dari universitas-universitas yang tidak dikomersialkan, tapi juga ada perusahaan yang menangani pembuatan program pengenalan wajah ini secara komersial.

Salah satu contoh adalah *software FaceIt* dari perusahaan *Visionics*,[8] perusahaan yang berdiri sejak tahun 1994 dan meneliti dan mengembangkan *software* untuk pengenalan wajah secara komersial.

Program ini mampu mendeteksi banyak wajah yang terdapat pada satu *image*, dan kemudian dapat melakukan segmentasi dan melepas gambar wajah oval tersebut dari latar belakangnya, dan dapat mengkompress gambar wajah sehingga menjadi 84 bytes per wajah.

Cara kerjanya yaitu menerima *input* sebuah gambar, dan kemudian dideteksi semua wajah oval, misal salah satu telah dideteksi, maka akan dilepas dari latas belakangnya dan kemudian masuk ke dalam tahap *preprocessing* untuk menyamakan ukuran, pencahayaan, ekspresi dan pose. Kemudian wajah dinormalisasi, dengan artian ditransformasi ke dalam representasi internal yang dinamakan *faceprint* dengan menggunakan teknik matematika yang dinamakan *local feature analysis.* Setelah ketemu *faceprint*, maka dibandingkan dengan *database* yang ada.Yang membuat *faceprint* bagus adalah ketahanannya terhadap perubahan pencahayaan, bentuk rambut, kacamata, ekspresi, dan pose.

Output-nya ada dua macam, yaitu menampilkan gambar *training* yang termirip dengan gambar *testing*-nya, atau langsung menunjukkan identitas gambar *testing*.

Software ini dapat digunakan untuk mengenali wajah *testing*. Selain itu juga dapat digunakan untuk verifikasi. Maksudnya, *faceprint* disimpan ke dalam

kartu kredit atau di komputer, kemudian *Faceit* bisa membandingkan *face print* dari wajah yang original dengan *image* wajah yang ada pada kartu kredit, dan dapat memberitahu apakah identitasnya sama. *Faceit* juga dapat dipakai untuk melakukan monitoring pada sebuah lingkungan, untuk mencari kehadiran dan lokasi dari suatu wajah yang diinginkan. Semua ini dilakukan secara otomatis, tidak memerlukan bantuan operasi dari manusia dan benar-benar secara *real time* (gambar yang hidup). Sehingga pada *airport* atau perbatasan negara dapat dipakai untuk memonitor apakah ada teroris atau kriminal, kemudian pada toko-toko retail dapat dipakai untuk mengecheck validitas dari *credit card*, bagi polisi dapat dipakai sebagai identifikasi orang yang hilang atau pencarian para kriminal.

Gambar 2.6. Contoh *User Interface Software FaceIt*

2.1.4. Aplikasi dari Pengenalan Wajah oleh Komputer

Apabila komputer dapat mengenali suatu wajah, maka banyak aplikasi yang dapat diterapkan dan dikembangkan untuk membantu pekerjaan manusia. Walaupun kecepatan komputer masih kalah dengan manusia dalam mengenali wajah sampai saat ini, tapi manusia dapat lelah, sedangkan komputer tidak. Jadi misalkan diterapkan untuk pengawasan di bank, apabila algoritma pengenalan

[8] Software FaceIT dari http://www.faceit.com/About/default.htm, Visionics Corporation.

wajah betul-betul sempurna, maka komputer tentu akan lebih teliti dan tidak akan lelah dalam menjalankan tugasnya.

Beberapa contoh aplikasi dari pengenalan wajah oleh komputer adalah:

? Pengenalan kredit *card*, surat ijin mengemudi (SIM), passport
? Pengenalan wajah untuk sekuritas sebagai pengganti tanda tangan atau sidik jari, misal untuk verifikasi *credit card*.
? Pengenalan pasien yang tidak sadarkan diri.
? Pengenalan orang yang hilang atau seorang kriminal.
? Pengontrolan masyarakat, seperti kamera di bank sehingga dapat mengidentifikasi bila ada orang jahat yang masuk ke bank.
? Rekonstruksi wajah sesuai dengan bayangan dari saksi pada suatu peristiwa kejahatan.
? Klasifikasi jenis kelamin.

Dari aplikasi-aplikasi tersebut dapat diambil kesimpulan bahwa inputan gambar ke dalam komputer dapat berupa gambar biasa (foto) dan gambar dinamis yang sesungguhnya / gambar hidup. Inputan yang berupa gambar hidup lebih sulit dikenali sebab mempunyai latar belakang yang banyak dan kualitas gambar kurang bagus karena gambar diambil dalam keadaan bergerak.

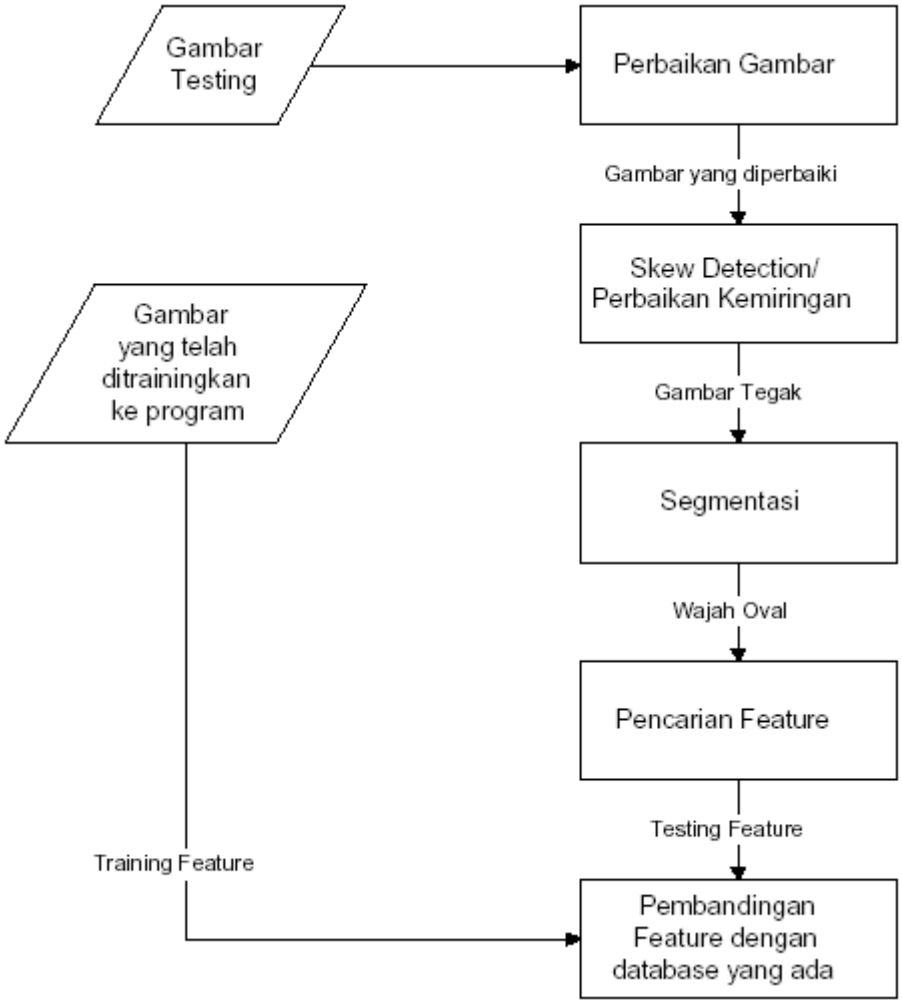
2.1.5. Tahapan Umum Pengenalan Wajah

Algoritma untuk mengenali wajah ada bermacam-macam. Tapi pada umumnya ada tahapan-tahapan umum yang dipakai dalam pengenalan wajah. Tahapan-tahapan umum dari pengenalan wajah yaitu :

? Perbaikan Gambar
? Segmentasi
? Pencarian *Feature*
? *Skew Detection* / perbaikan kemiringan
? Identifikasi

Tahapan ini bisa dilihat dalam gambar 2.7 untuk lebih jelasnya. Meskipun pada umumnya pengenalan wajah harus melalui tahapan-tahapan tersebut, tapi dalam kenyataannya banyak peneliti yang melompati beberapa tahap, misalkan tahap segmentasi dilompati sehingga wajah yang dimasukkan

harus sudah tersegmentasi manual. Hal ini mungkin disebabkan karena keterbatasan waktu atau biaya bagi peneliti.



Gambar 2.7. Tahap Utama Pengenalan Wajah

2.1.5.1. Perbaikan Gambar

Input gambar dari hasil *scanning* belum tentu bagus atau kabur, sehingga bukan merupakan *input* yang bagus bagi program. *Image* hasil *scanning* biasanya mengandung banyak *noise* yang harus dihilangkan. Kemudian harus diterapkan algoritma-algoritma untuk mempertajam gambar bila gambar kabur, seperti *filtering* yang teorinya ada pada *image processing*. Setelah itu dilakukan pengaturan pencahayaan pada foto, sehingga diharapkan semua foto mempunyai pencahayaan yang sama sehingga memudahkan dalam proses pengenalan nanti.

2.1.5.2. Segmentasi

Pada tahap ini, segmentasi digunakan dengan suatu algoritma untuk mengenali gambar mana yang merupakan wajah. Apabila ternyata gambar setelah dicek oleh algoritma bukan merupakan wajah, maka proses berhenti dan keluar. Tapi bila pada gambar tersebut ada bentuk oval dari wajah, maka foto tersebut akan disegmentasi untuk membuang latar belakangnya sehingga hasilnya adalah wajah oval saja tanpa latar belakangnya. Yang dimaksud wajah oval yaitu bentuk oval dari wajah, yaitu wajah tanpa rambut, telinga atau leher.

2.1.5.3. Pencarian *Feature*

Ada dua macam *feature* pada wajah, yaitu *holistic features* dan *partial features*[9]. Pada *partial feature*, dalam hal pengenalan wajah biasanya disebut sebagai *facial feature*, contoh *feature*-nya adalah warna dan bentuk rambut, besar dan letak hidung, mulut, mata, telinga dan lain-lain. Sedangkan pada *holistic feature* setiap *feature*-nya adalah merupakan suatu karakteristik dari seluruh wajah, maksudnya wajah dianggap sebagai suatu kesatuan yang utuh. *Principal Component Analysis* adalah teknik yang menggunakan *holistic feature*, jadi *Principal Component Analysis* menganggap suatu wajah adalah suatu kesatuan dan dibandingkan dengan wajah yang lain.

2.1.5.4. Perbaikan Kemiringan

Foto oval dari wajah orang yang didapatkan dari hasil segmentasi dapat tegak, atau miring ke kiri atau ke kanan, atau bahkan terbalik. Oleh karena itu proses perbaikan kemiringan ini diperlukan, sehingga gambar yang miring atau terbalik bisa diperbaiki sehingga menjadi gambar yang benar-benar tegak.

Sebenarnya pada foto wajah, kemiringan bukan hanya sesederhana bila kita mengamati tulisan. Tulisan atau huruf hanya bisa miring ke kiri atau ke kanan. Tapi pada foto, selain miring ke kiri atau ke kanan, ada juga kemiringan pada orientasi wajah. Maksudnya foto orang tidak selalu menghadap ke depan, tapi bisa menoleh beberapa derajat ke kanan atau ke kiri. Maka seharusnya ada algoritma untuk mengatur orientasi wajah tersebut.

---

[9] Sami Romdhani, *Face Recognition Using Principal Component Analysis*, http://www.elec.gla.ac.uk/~romdhani

Facial feature biasanya dimanfaatkan untuk perbaikan kemiringan. Misalnya dengan mengamati keadaan lurus atau miringnya dari letak dua mata, maka dapat ditentukan sudut rotasi bagi wajah. Letak hidung juga harus diperhitungkan, agar rotasi wajah yang dihasilkan tidak terbalik, sehingga letak mata harus di atas hidung dan bukan sebaliknya. Berdasarkan *feature* ini, maka gambar hanya bisa dirotasi kemiringannya sehingga menjadi tegak, tapi tetap saja tidak ada algoritma untuk memperbaiki orientasi wajah.

2.1.5.5. Identifikasi

Pada tahap akhir akan diambil kesimpulan. Kesimpulan yang bisa diambil bisa berbeda tergantung pada aplikasi yang diinginkan, misalnya kesimpulan yang diperlukan yaitu untuk menggolongkan individu pada *image* berdasarkan jenis kelaminnya, atau melakukan pengenalan terhadap foto yang diinputkan dan langsung memberi identitas foto tersebut.

Setelah komputer tahu tentang *feature* dari *testing* foto yang diinputkan, maka dapat dilakukan pembandingan dengan *feature-feature* dari foto pelatihan. Tentu saja sebelumnya foto-foto pelatihan juga melalui tahap yang sama dari satu sampai lima, sehingga ketemu *feature-feature*-nya. Setelah dilakukan pembandingan, maka akan dipilih satu wajah *training* yang *feature*-nya paling mirip dengan *feature* dari foto *testing*. Wajah *training* inilah yang merupakan *output* dari program pengenalan wajah, yaitu sebagai wajah pelatihan yang termirip dengan wajah *testing* (bila aplikasi yang diinginkan adalah pengenalan wajah).

2.2. *Image Processing*

Image greyscale lebih mudah untuk dianalisa apabila dibandingkan dengan *image* berwarna. *Image* berwarna adalah kombinasi dari tiga warna utama merah, hijau, dan biru yang disebut dengan sistem warna RGB. Sistem RGB terdiri dari 24 bit, masing-masing 8 bit untuk merah, hijau, dan biru. Agar lebih mudah untuk dianalisa maka *image* terlebih dahulu diubah menjadi *image greyscale* dengan cara memberi nilai yang sama untuk masing-masing 8 bit itu.

2.3. *Principal Component Analysis* (PCA)

Metode *Principal Component Analysis* (PCA) dibuat pertama kali oleh para ahli statistik. Metode PCA pertama kali ditemukan oleh Karl Pearson pada tahun 1901, yang memakainya pada kontek biologi. Meskipun pertama kali ditemukan oleh Karl Pearson, sebenarnya prosedur umumnya seperti yang diketahui sekarang ini ditemukan oleh Harold Hotelling di mana paper yang menjadi pioner ini muncul pada tahun 1933, yang membahas pada bidang *psychometry*. Kemudian tidak ada perkembangan baru pada teknik ini, dan. perkembangannya baru mulai pesat pada akhir tahun 1930 dan awal 1940. Setelah itu perkembangannya berkurang sebentar sampai komputer telah berhasil didesain sehingga dapat mengaplikasikan teknik ini pada masalah-masalah yang masuk akal. Pada tahun 1947 teori ini muncul lagi dan cukup independen sebagai teori probabilitas yang ditemukan oleh Karhunen, dan kemudian dikembangkan oleh Loeve pada tahun 1963.[10] Sehingga teori ini juga dinamakan Karhunen-Loeve *transform* pada bidang ilmu telekomunikasi.

PCA adalah termasuk dalam bidang *multivariate analysis* pada ilmu statistik. Metode PCA ini mungkin teknik yang terlama dan terpopuler dalam bidang *multivariate analysis. Multivariate analysis* secara sederhana dapat dijelaskan sebagai metode yang berhubungan dengan variabel dalam jumlah besar pada satu atau banyak percobaan.

Beberapa bidang lain pada *multivariate analysis* adalah *common factor analysis*, *multiple regression*, *multiple discriminant analysis, multivariate analysis of variance* dan *covariance, conjoint analysis, canonical correlation, cluster analysis*, *multi dimensional scaling*, *correspondence analysis, linear probability model*, dan *simultaneous* / *structural equation modelling*.[11]

---

[10] Haykin, S, Neural Networks: A Comprehensive Foundation, (NY: Macmillan, 1994), p. 364.

[11] Hair Jr, Joseph F.Hair dan Anderson, Rolph E., dan Tatham, Ronald L, dan Black, William C., Multivariate Data Analysis (Prentice-Hall International, Inc, 1998), p.13.

Kesulitan yang sering timbul dalam aplikasi *Linear Discriminant Analysis* dan Jaringan Saraf Tiruan adalah lamanya proses belajar yang dikarenakan besarnya dimensi *input*. Tetapi dengan adanya algoritma PCA, dimensi *input* tersebut dapat direduksi sehingga proses belajar menjadi lebih efisien. Hal ini disebabkan karena algoritma PCA mereduksi perhitungan matriks berdimensi n x n (n adalah jumlah *pixel*) menjadi m x m (m adalah jumlah *image training*). Dari matriks ini didapatkan *feature* PCA yang digunakan sebagai *input* untuk algoritma LDA dan JST.

Sebagai contoh, misalnya ada 100 *image training* berdimensi 100 x 100 = 10.000 *pixel*. Dengan menggunakan metode PCA, didapatkan *feature* PCA yang berdimensi 100 x 100.

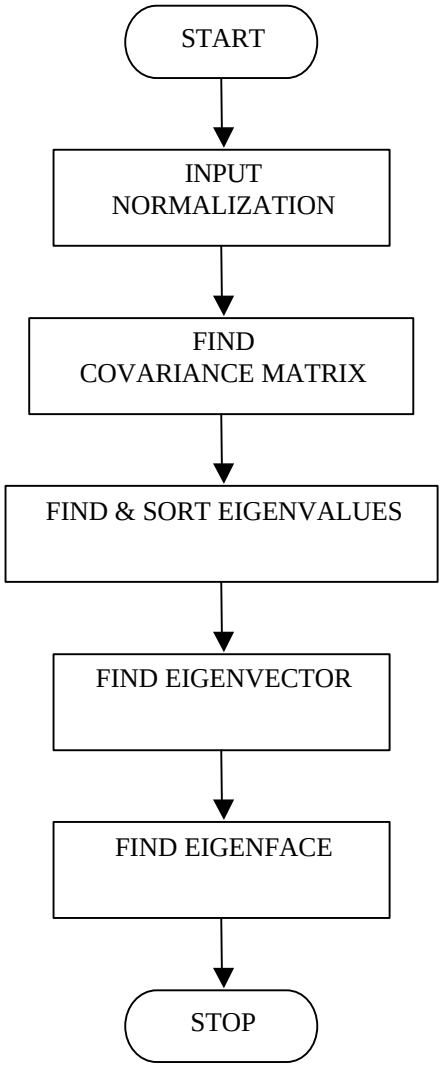
Kemudian matriks ini digunakan sebagai *input* bagi JST, sehingga seakan-akan ada 100 *image* dengan dimensi 10 x 10 saja

Prinsip dasar algoritma PCA adalah memproyeksikan *image* ke dalam bidang ruang *eigen*-nya dengan cara mencari *eigen vector* yang dimiliki setiap *image* dan memproyeksikannya ke dalam ruang *eigen* yang didapat tersebut. Besar ruang *eigen* tergantung dari jumlah *image training* yang dimiliki.

Jadi PCA merupakan metode untuk mengambil ciri penting dari sekumpulan *data set* dengan mereduksi data tersebut menjadi data yang otonormal dan tidak saling berkorelasi. Sasaran PCA adalah menangkap variasi total dan menjelaskannya dengan variabel lebih sederhana. Dengan variabel ini, ciri yang khusus dari kumpulan data tersebut dapat digambarkan dengan lebih baik.

2.3.1. *Flowchart* Pembentukan PCA

Flowchart Pembentukan PCA dapat digambarkan sebagai berikut:



Gambar 2.8. *Flowchart* PCA

2.3.2. Normalisasi *Input*

Hal pertama yang harus dilakukan adalah memasukkan *pixel* tiap *image* ke dalam matriks. Sistem penyimpanannya adalah dengan memasukkan semua kolom dalam satu baris sampai habis dahulu, baru kemudian pindah ke kolom pada baris berikutnya. Misalkan ada m *image* yang masing-masing berdimensi 100 x 100 = 10.000 *pixels*. Maka matrik baru yang mempresentasikan *image-image training* tersebut berdimensi jumlah wajah baris x 10.000 kolom.

2.3.3. Mencari *Covariance* Matriks

Setelah data dari tiap *pixel* dimasukkan, kemudian rata-rata dari matrik u dapat dicari. Langkah pertama adalah mencari jumlah total dari tiap baris matriks u, kemudian dirata-ratakan dengan dibagi 10.000. Kemudian semua pixel pada baris itu dikurangi dengan rata-ratanya.

$$\bar{u} ? \frac{1}{m} \sum_{k?1}^{m} u_{1,k} \quad (2.1)$$

Semua variasi yang memungkinkan diperoleh dari perpasangan vektor kolom dinyatakan sebagai *covariance* matriks. *Covariance* Matriks didapat dengan cara mengalikan matriks u dengan *transpose*-nya. Matriks baru yang dihasilkan berdimensi jumlah wajah baris x jumlah wajah kolom.

$$C ? u ? u^t \quad (2.2)$$

2.3.4. Mencari *EigenValue* Dan *EigenVector*

Setelah matriks *covariance* dihitung, langkah berikutnya adalah mencari *eigen value* dan *eigen vector*. Pencarian *eigen value* dan *eigen vector* ini dapat dibantu dengan menggunakan metode *Jacobi*. *Eigen value* yang didapat diurutkan mulai yang terbesar sampai yang terkecil, dan *eigen vector* yang bersesuaian dengan *eigen value* tersebut juga diurutkan. *Eigen value* yang dihasilkan merupakan matriks satu dimensi sebanyak jumlah wajah training. Sedangkan matriks *eigen vector* yang dihasilkan merupakan matriks berdimensi jumlah wajah baris x jumlah wajah kolom.

2.3.5. *Eigenface* PCA

Matriks Eigenface dihitung dengan cara mengalikan matriks u dengan matriks eigenvektor dan dengan satu dibagi akar eigenvaluenya.

$$\text{Eigenface} = \frac{1}{\sqrt{Eigenvalue}} * \text{Eigenvector} * u \quad (2.3)$$

Matriks *u* merupakan matriks berdimensi jumlah wajah baris x 10.000 kolom, sedangkan matriks eigenvector berdimensi jumlah wajah baris x jumlah wajah kolom. Untuk pengali ketiga yaitu seper akar eigenvalue, berdimensi

jumlah wajah x jumlah eigen, di mana 1 ? Jumlah Eigen ? Jumlah wajah. Jumlah eigen adalah jumlah principal component yang hendak dipakai untuk pengenalan. Nilai jumlah eigen paling bagus adalah sama dengan jumlah wajah, dimana akan mencakup hampir semua variance yang ada, tapi untuk mempersingkat waktu apabila data terlalu besar maka nilai jumlah eigen dapat dikecilkan, paling minimal jumlah eigen = 1. Hal ini diperbolehkan sebab dengan jumlah eigen ? jumlah wajah, sudah dapat mencakup variance yang besar. Sehingga matriks eigenface yang dihasilkan akan mempunyai dimensi jumlah eigen baris x 10.000 kolom.

Matriks *Eigenface* ini juga perlu disimpan supaya tidak perlu melakukan proses *training* berulang-ulang, cukup memanggil file yang bersangkutan dan meng-*upload* nya ke memori computer.

Secara umum, garis besar langkah-langkah dari metode PCA adalah sebagai berikut :

? cari matriks u

? cari matriks *covariance* : $C ? u^T ? u$

? cari *eigen values* (?) dan *eigen vector* (V) dari matriks C

? cari matriks *eigenface: eigenface* = $\frac{1}{\sqrt{Eigenvalue}}$ * Eigenvector * u

? matriks eigenface dapat digunakan untuk pengenalan image.

2.4. *Backpropagation Network*

Backpropagation adalah suatu teori *learning* dari *Artificial Neural Network* yang dipakai dalam tugas akhir ini. Oleh karena itu dalam sub bab ini akan dijelaskan sekilas tentang *Artificial Neural Network*, kelebihan dan kekurangannya, serta unsur – unsur yang ada pada *Neural Network*. Selain itu juga dijelaskan *Backpropagation Network*, algoritma dari *Backpropagation*, faktor-faktor yang dapat menunjang keberhasilannya, serta contoh pemakaiannya.

2.4.1. *Artificial Neural Network*

Berdasarkan pengetahuan tentang otak manusia, dikembangkanlah suatu metode baru yang dikenal dengan nama *Artificial Neural Network*, atau

singkatnya disebut *Neural Network*. Perjuangan untuk mengetahui struktur dan cara kerja otak ini pertama kali dimulai oleh Ramon y Cajal pada tahun 1911, yang memperkenalkan ide tentang *neuron* sebagai suatu struktur dari otak. *Neuron* lebih lambat lima sampai enam kali lipat dari gerbang *logic* silikon, suatu *event* dalam sebuah *chip* silikon terjadi pada *range nano* detik (10^{-9} detik), sedangkan *event* dalam suatu *neural* terjadi dalam milli detik (10^{-3} detik).[12] Diperkirakan ada 10 billiun *neuron* yang ada di otak manusia, dan 60 trilyun *synapses* atau koneksi antara *neuron*.

Otak adalah sistem yang *complex*, *nonlinear* dan merupakan *parallel* komputer. Otak mempunyai kemampuan untuk mengorganisasikan *neuron – neuron*-nya untuk melakukan komputasi tertentu (seperti pengenalan *pattern*, pengenalan suara,) lebih cepat beberapa kali daripada komputer *digital* yang ada sekarang.

Secara historis, komputer telah didominasikan dengan konsep perhitungan terprogram, dimana (biasanya secara prosedural) algoritma didisain dan diimplementasikan dengan menggunakan arsitektur yang dominan. Sedangkan komputasi pada otak manusia berbeda, yaitu pada:

?? Komputasinya terdistribusi dan paralel

? Proses belajar menggantikan pembuatan – pembuatan program sebelumnya.[13] (*neural network* mengandalkan proses belajar berdasarkan contoh – contoh *input* yang diberikannya, sedangkan program konvensional bekerja berdasarkan instruksi – instruksi yang diberikan kepadanya)

Meskipun *Neural Network* adalah suatu algoritma yang meniru cara kerja otak manusia, tapi tidak semuanya ditiru dan didesain secara persis dengan struktur otak yang asli. Misalnya pesawat atau dalam hal ini disebut *artificial bird* (burung buatan) yang terbang tanpa mengepakkan sayapnya, karena mengepakkan sayapnya bukan prinsip utama. Oleh karena itu mempunyai koneksi antar *processor* adalah prinsip utama, tapi menggunakan kimia dan elektrik proses

[12] Haykin, Simon, Neural Network : A Comprehensive Foundation, Macmillan 1994, p. l.
[13] Schalkoff, Robert J., Artificial Neural Network, McGraw-Hill Compani 1997, p. l.

untuk interaksi antar *processor*, seperti otak, bukanlah prinsip utama.[14] *Neural Network* mengadopsi cara kerja otak dalam dua hal :[15]

? Pengetahuan didapatkan oleh jaringan berdasarkan suatu proses belajar

? Koneksi antara *neuron* atau yang dikenal sebagai bobot *synaptic* digunakan untuk menyimpan pengetahuan tersebut.

Karena mengadopsi cara kerja otak itulah, maka nama *Artificial Neural Network* diberikan, yang artinya jaringan *neural* buatan.

Prosedur yang digunakan dalam proses belajar disebut sebagai *learning algorithm* (algoritma belajar), sebagai suatu fungsi untuk memodifikasi bobot pada *network* sedemikian rupa sehingga mencapai tujuan yang diinginkan.

Artificial Neural Network ini sangatlah menarik untuk dikembangkan karena kenyataan bahwa model yang mirip dengan sistem jaringan otak, yang ternyata dapat dipergunakan untuk melakukan hal-hal yang berguna, yaitu antara lain dapat memecahkan permasalahan yang sebelumnya tidak dapat dipecahkan dengan pemrograman tradisional.

2.4.1.1. Kelebihan dan Kekurangan dari *Neural Network*

Sebagai suatu algoritma, tentu *neural network* mempunyai kelebihan dan kekurangan. *Neural Network* cocok bila dipakai untuk aplikasi seperti pengenalan suatu pola berdasarkan contoh-contoh pelatihan yang diberikan.

Kelebihan dari *Neural Network* adalah:

? *Parallel*. Kemampuan bekerja secara *parallel* inilah yang bila dioptimalkan secara maksimal bisa mempercepat pekerjaan dalam pengenalan suatu corak, misalnya dengan menerapkan algoritma *neural network* pada beberapa komputer yang bekerja bersamaan.

? Pemetaan *Input Output* . *Neural Network* dapat dibayangkan sebagai suatu '*black box*', yang hanya menerima contoh-contoh *input* dan *output* yang diinginkan tanpa harus tahu bagaimana operasi dari *input* menjadi *output*, dan melalui proses *training neural network* akan bisa mengubah suatu *input* dari *testing* menjadi *output* yang diinginkan. Pada *network* akan dimasukkan

[14] Bharath, Ramachandran dan Drosen, James, Neural Network Computing, (McGraw-Hill,Inc,1994),p.xix

contoh, dan kemudian meng-*update* bobot *synaptic* sehingga meminimalkan perbedaan antara *output* yang diinginkan dengan *output* sesungguhnya.

? Mudah teradaptasi. *Neural Network* mempunyai kemampuan secara built sehingga dapat mengadaptasi bobot *synaptic*-nya terhadap perubahan-perubahan pada lingkungan sekitarnya. *Neural network* yang dilatih di suatu bidang tertentu dapat dengan mudah dilatih kembali untuk menghadapi perubahan-perubahan kecil yang terjadi, misalnya *statistic* yang berubah berdasarkan waktu.

Selain kelebihan-kelebihan tersebut, kekurangan-kekurangan dari *Neural Network* adalah

? Tidak ada aturan-aturan atau petunjuk disain yang baik dan benar pada aplikasi. Misal tidak ada petunjuk tentang bagaimana menentukan beberapa *layer* yang harus dipunyai pada sebuah *network*, atau bagaimana *topology* jaringannya dan lain sebagainya, dan semuanya ini harus dicari yang terbaik melalui percobaan.

? *Training* yang dimasukkan dapat sulit atau tidak masuk akal.

? Tidak ada langkah umum untuk memperkirakan operasi internal pada *network*

? Sulit untuk memprediksikan performa dari *network* selanjutnya.

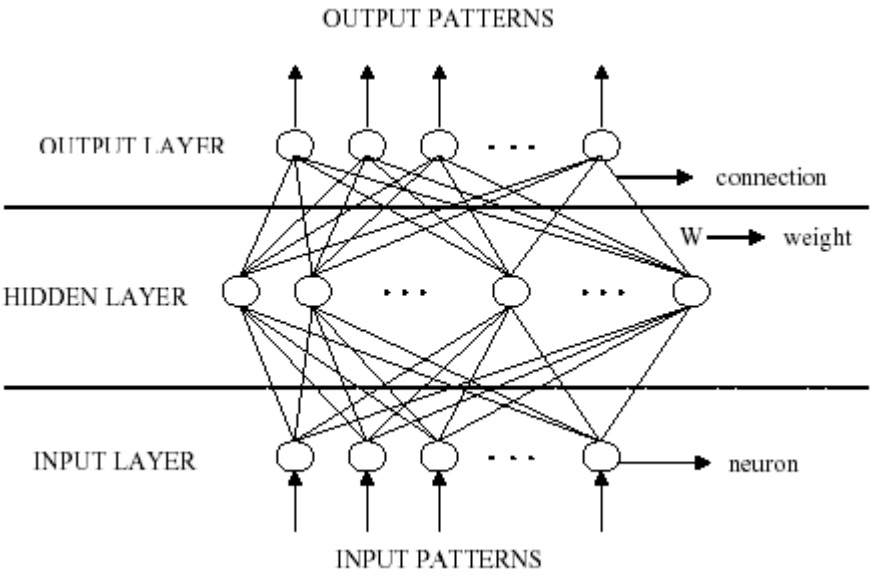
2.4.1.2. Unsur – Unsur *Neural Network*

Neural Network memang mempunyai berbagai macam algoritma, susunan *network*, rumus-rumus dan variabel yang berbeda-beda. Meskipun demikian, masing-masing algoritma tersebut mempunyai unsur– unsur yang umum sebagai bagian dari *Neural Network*.

Neural Network juga mempunyai struktur secara umum bagi algoritma-algoritmanya. Struktur umum dari *neural network* adalah seperti yang terdapat pada gambar 2.9.

---

[15] Haykin, p. 2, Op Cit.



Gambar 2.9. Struktur Umum *Neural Network*

Adapun unsur-unsur umum tersebut adalah :

? *Input Pattern*

Input pattern adalah sejumlah pola yang akan dipakai dan dimasukkan ke dalam *input layer*. Pola ini dapat digunakan dalam proses pelatihan maupun dalam proses pengujian suatu jaringan. Pola ini merupakan salah satu *input* yang berasal dari luar jaringan.

? *Output Pattern*

Output pattern adalah hasil dari *output layer* yang membentuk suatu pola tertentu. Pada proses pelatihan yang menggunakan metode *supervised learning*, *output pattern* akan dibuat semirip mungkin dengan *output* yang diinginkan. Sedangkan pada proses pelatihan yang menggunakan metode *unsupervised learning*, *output pattern* dapat berupa pengklasifikasian *input pattern*.

? *Input Layer*

Input layer adalah lapisan pertama dari sebuah jaringan yang menerima *input* berdasarkan *input pattern* yang dimasukkan oleh *user*. Lapisan ini dapat disebut sebagai *input* bagi sebuah jaringan. *Input layer* ini adalah suatu lapisan yang selalu ada pada setiap model *neural network*.

? *Hidden Layer*

Hidden layer adalah lapisan yang terletak di antara *input layer* dan *output layer*, yang menerima *input* dari *input layer*. Terdapat beberapa model jaringan yang memiliki lebih dari satu *hidden layer* dan ada juga model yang tidak memerlukan *hidden layer* sama sekali. Penentuan berapa *hidden layer* yang harus dibuat disesuaikan dengan kebutuhan dan harus direncanakan pada saat perancangan sebuah jaringan.

? *Output Layer*

Output layer adalah lapisan yang terakhir dari sebuah jaringan. Lapisan ini menerima *input* dari *hidden layer* (pada model jaringan yang memiliki satu atau lebih *hidden layer*) atau dari *input layer* (pada model jaringan yang tidak memiliki *hidden layer*). Lapisan ini menghasilkan *output* dari suatu jaringan, yang kemudian akan disebut sebagai *output pattern*. Pada beberapa model jaringan, lapisan ini akan melakukan kalkulasi kesalahan dengan cara membandingkan *output* yang dihasilkannya dengan *output* yang diinginkan. Selanjutnya nilai kesalahan ini dapat dipergunakan untuk memperbaiki nilai *weight* atau bobot.

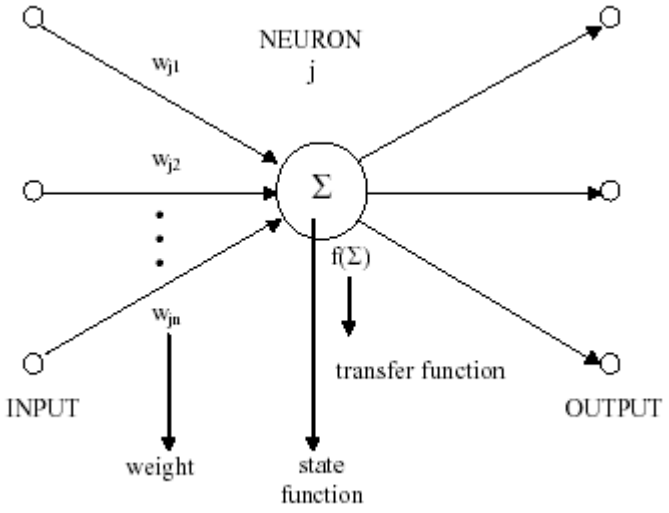
? *Weight*

Weight merupakan suatu bilangan *real* yang terdapat pada tiap-tiap *connection*, yang merupakan bobot dari *connection* tersebut. Setiap *connection* yang menjadi *input* dari suatu *neuron* pasti memiliki *weight*, demikian juga dengan bias yang selalu memiliki *weight* sama dengan 1. *Weight* digunakan agar suatu jaringan dapat belajar (*learning*), yaitu dengan cara mengubah nilai dari bilangan *real* yang diberikan pada *weight* tersebut, sampai didapatkan nilai *output* yang paling sesuai dengan nilai *output* yang diinginkan.

? *Neuron*

Neuron atau *node* atau *cell* ialah tempat jaringan melakukan perhitungan matematik. Sebuah *neuron* terdiri dari sejumlah *input connection* (koneksi *input*) dengan *neuron* yang lain termasuk *bias input*, sebuah *state function*, sebuah *transfer function*, dan *output*. *Input connection* memberikan suatu nilai *input* tertentu yang dapat berasal dari *neuron* yang terdapat pada lapisan sebelumnya atau untuk *input layer* berasal dari *input pattern*. *Input* dari

neuron ini berupa bilangan *real* dengan skala antara –1.0 sampai dengan 1.0. Bias *input* adalah *input* yang tidak berasal dari *neuron* jaringan lain dan dalam perhitungan bias selalu bernilai 1. *State function* digunakan untuk melakukan perhitungan terhadap *input-input* yang didapat oleh *neuron* tersebut. *State function* yang umum digunakan ialah fungsi penjumlahan biasa, yaitu penjumlahan terhadap nilai-nilai *weight* yang terdapat dalam *input connection*. *Output* dari *state function* menjadi *input* bagi *transfer function*. *Transfer function* berupa fungsi matematika yang *nonlinear*, yang digunakan untuk mengkonversikan *input* yang diterimanya ke dalam suatu skala tertentu. Sehingga *output* dari fungsi ini juga berupa bilangan *real* dengan skala tergantung dari jenis fungsi aktivasi yang digunakan. Namun secara umum *output* fungsi ini juga mempunyai skala yang sama dengan *input* yaitu –1.0 sampai 1.0. Dua tipe *transfer function* adalah *continuous* dan *discrete*. Fungsi *continuous* yang biasa digunakan ialah *Ramp*, *Sigmoid*, *Arc Tangent*, dan *Hyperbolic Tangent*. Fungsi *continuous* ini biasa disebut sebagai *squashing function.* Sedangkan fungsi *discrete* yang umum dipakai ialah *Step* dan *Threshold*. Fungsi *discrete* ini biasa disebut sebagai *activation function*. *Output* dari *transfer function* ini sekaligus juga merupakan *output* dari *neuron*. Gambaran umum tentang suatu *neuron* dengan *input* dan *output* yang dimilikinya adalah seperti yang terdapat pada gambar 2.10.



Gambar 2.10. Struktur *Neuron* dalam *Neural Network*

? *Connection*

Connection merupakan penghubung antara suatu *neuron* dengan *neuron* yang lain. *Connection* ini terkadang disebut juga dengan nama *synapsis* atau *synapses*. Kekuatan suatu *neural network* terletak pada banyaknya jumlah *connection* yang dimilikinya. Semakin banyak jumlah *connection* yang dimiliki suatu jaringan, semakin besarlah kekuatannya.

Pada gambar 2.10, dijelaskan struktur tentang sebuah *neuron* dalam *neural network*. Sebuah *neuron* dalam *neural network* dapat mempunyai inputan-inputan dari *neuron* lain, yang mempunyai bobot yang disimbulkan dengan w_{j1} – w_{jn} pada gambar 2.10. *Neuron* kemudian akan melakukan *transfer function* untuk menentukan bobot-bobot yang akan keluar dari *neuron* tersebut.

2.4.2. *Backpropagation*

Backpropagation adalah metode yang paling dikenal dan paling sering digunakan diantara tipe-tipe dari *neural network* yang ada sekarang.

Backpropagation network adalah *multilayer feedforward network* dan termasuk dalam *supervised learning networks*.

Backpropagation hanya dapat mengenali suatu pola yang mirip dengan yang telah dipelajari, tapi tidak bisa mengenali bila dimasukkan pola yang baru. Agar dapat mengenali pola yang baru maka pada *network* harus dimasukkan pola baru sebagai *input training*. Tapi bila hanya pola baru yang dimasukkan untuk *training* ulang, maka pola-pola yang lama akan dilupakan, sehingga proses belajar tidak dapat bertambah otomatis. Ini adalah kelemahan utama bagi *supervised learning networks*.[16] Kelemahan lainnya adalah bahwa *backpropagation network* dapat memasuki *local* minimal sehingga tidak akan menemukan solusi optimal.

Backpropagation network terdiri dari satu lapisan *input*, satu lapisan *output* dan satu atau lebih lapisan *hidden*. Umumnya *network* adalah *full connected* (terhubung penuh) antara *adjacent layers*.

[16] Fu, Li Min, Neural Network in Computer Intelligence, McGraw-Hill, 1994, p. 80.

2.4.2.1. Algoritma *Backpropagation*

Algoritma *Backpropagation* mempunyai 2 tahap utama, yaitu *learning* dan *testing*. Untuk *learning*, proses bekerja secara *forward*, dan kemudian *backward*. *Forward* maksudnya perhitungan dihitung mulai dari *layer* yang paling depan, yaitu *input layer*, ke belakang sampai *output layer*. Pada tahap *forward* ini dilakukan perhitungan *output* dari *network*. Setelah itu ada tahap *backward*, yang dilakukan dari *output layer* sampai ke *input layer* yang bertujuan meng-*update* bobot dalam setiap *network*.

Berikut adalah algoritma dari *Backpropagation Network* (algoritma untuk mentraining *network* sehingga menghasilkan *output* yang diinginkan):

? Inisialisasi Bobot

Tentukan semua bobot pada setiap koneksi ke suatu bilangan *random* yang kecil.

? Perhitungan aktivasi

Aktivasi level dari *input* unit ditentukan berdasarkan contoh-contoh yang dimasukan ke dalam *network*.

Aktivasi antara *input* layer dan *hidden* layer ditentukan oleh:

$$Z\_in_j = v_{0j} + \sum_i x_i v_{ij} \quad (2.4)$$

dimana v_{oj} adalah bobot bias antara input layer dan *hidden* layer, x_i adalah *vector* input, dan v_{ij} adalah bobot antara input layer dan *hidden* layer.

$$z_j = f(z\_in_j) \quad (2.5)$$

Fungsi aktivasi yang dipakai adalah *binary sigmoid*, sehingga persamaan diatas menjadi:

$$z_j = \frac{1}{1 + \exp(-z\_in_j)} \quad (2.6)$$

Aktivasi antara *hidden* layer dan *output* layer ditentukan oleh:

$$Y\_in_k = w_{0k} + \sum_j z_j w_{jk} \quad (2.7)$$

dimana w_{ok} adalah bobot bias antara *hidden* layer dan *output* layer, w_{jk} adalah bobot antara *hidden* layer dan *output* layer.

$$y_k = f(y\_in_k) \quad (2.8)$$

Fungsi aktivasi yang dipakai adalah *binary sigmoid*, sehingga persamaan diatas menjadi:

$$Y_k = \frac{1}{1 ? \exp(? y\_in_k)} \quad (2.9)$$

? Perhitungan *Backward*

Untuk *output* unit, error *gradient* diberikan dengan:

$$?_k ? (t_k ? y_k) f'(y\_in_k) \quad (2.10)$$

$$? (t_k ? y_k) y_k (1 ? y_k)$$

dimana t_k adalah target *output* yang diinginkan dan y_k adalah *output* yang sebenarnya dari aktivasi antara *hidden layer* dan *output layer*.

Perubahan bobot antara *hidden layer* dan *output layer* diberikan dengan:

$$?w_{jk}(t ? 1) ? ??_k z_j ? (?w_{jk} * momentum) \quad (2.11)$$

dimana ? adalah *learning rate* ($0 < ? < 1$)

Perubahan bobot bias antara *hidden layer* dan *output layer* diberikan dengan:

$$?w_{0k}(t ? 1) ? ??_k ? (?w_{0k} * momentum) \quad (2.12)$$

Untuk *hidden* unit, *error gradient* diberikan dengan:

$$?\_in_j ? \sum_{k?1}^{m} ?_k w_{jk} \quad (2.13)$$

$$?_j ? (?\_in_j) f'(z\_in_j) \quad (2.14)$$

$$? (?\_in_j) z_j (1 ? z_j)$$

Perubahan bobot antara *input layer* dan *hidden layer* diberikan dengan:

$$?v_{ij}(t ? 1) ? ??_j x_i ? (?v_{ij} * momentum) \quad (2.15)$$

Perubahan bobot bias antara *input layer* dan *hidden layer* diberikan dengan:

$$?v_{0j}(t ? 1) ? ??_j ? (?v_{0j} * momentum) \quad (2.16)$$

? Update bobot dan bobot bias antar layer

Update bobot antara hidden layer dan output layer dengan :

$$w_{jk}(new) ? ? w_{jk}(old) ? ? ?w_{jk}(t ? 1) \quad (2.17)$$

Update bobot bias antara hidden layer dan output layer dengan:

$$w_{0k}(new) ? ? w_{0k}(old) ? ? ?w_{0k}(t ? 1) \quad (2.18)$$

Update bobot antara input layer dan hidden layer dengan:

$$v_{ij}(new) = v_{ij}(old) + \Delta v_{ij}(t+1) \quad (2.19)$$

Update bobot bias antara input layer dan hidden layer dengan:

$$v_{0j}(new) = v_{0j}(old) + \Delta v_{0j}(t+1) \quad (2.20)$$

- Ulangi iterasi sampai *convergen*

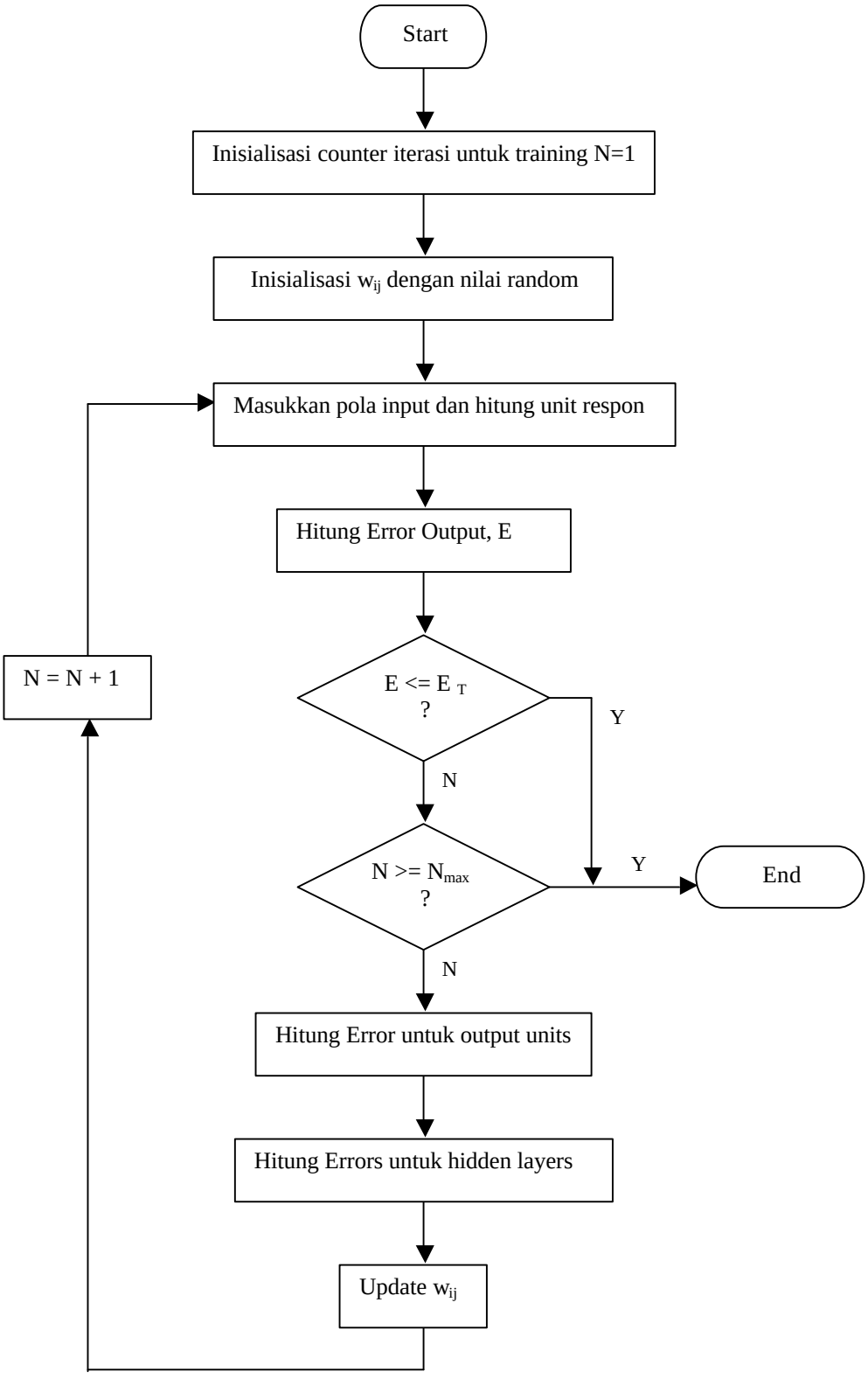
  Ulangi langkah diatas sampai *error* yang didapatkan sesuai dengan kriteria *error* yang dipilih. Sebuah iterasi terdiri dari penginputan contoh, menghitung aktivasi dan memodifikasi bobot.

2.4.2.2. Faktor Penunjang Keberhasilan

Neural Network merupakan suatu metode yang penerapannya tidak sama pada setiap kasus. Tidak ada teori untuk menentukan tentang bagaimana struktur dari *network*, jumlah *hidden* unit, angka *learning rate* dan momentum, dan sebagainya. *Error* dari *neural network* dapat dipengaruhi oleh :[17]

- Algoritma *learning* dan jumlah iterasi. Hal ini menentukan seberapa bagus *error* dari *training* set dapat diminimalkan.
- Jumlah dari contoh *learning*. Hal ini menentukan seberapa bagus contoh *training* dalam merepresentasikan fungsi aktualnya.
- Jumlah dari *hidden units.* Hal ini menentukan *expressive power* dari *network*. Untuk fungsi yang 'lembut' diperlukan beberapa *hidden units* saja, sedangkan untuk fungsi yang berfluktuatif memerlukan banyak *hidden units.*

[17] Krose, Ben dan Patrick van der Smagt, An Introduction to Neural Networks, University of Amsterdam, http://www.science.uva.nl/research/ias/ , November 1996, p.42.



Gambar 2.11. *Flowchart* dari algoritma *Backpropagation* [18]