SISTEM PENGENALAN WAJAH DENGAN MENGGUNAKAN *BACKPROPAGATION ARTIFICIAL NEURAL NETWORK* DAN *PRINCIPAL COMPONENT ANALYSIS*

FACE RECOGNITION SYSTEM USING BACKPROPAGATION ARTIFICIAL NEURAL NETWORK AND PRINCIPAL COMPONENT ANALYSIS

Ikhwannul Kholis[1], Syah Alam[2]
Universitas 17 Agustus 1945 Jakarta
[1]ikhwannul.kholis@uta45jakarta.ac.id, [2]syah.alam@uta45jakarta.ac.id

Abstrak

Pengenalan wajah dapat dilakukan dengan menggunakan metode *Backpropagation Artificial Neural Network* (ANN) dan *Principal Component Analysis* (PCA). ANN dibuat menyerupai sistem syaraf manusia. Dengan beberapa parameter pada *Backpropagation*, dapat diketahui karakteristik *Backpropagation* sehingga dapat memperkecil *error* dan *epoch*, serta memperbesar *Recognition Rate*. Hasil percobaan menunjukkan hubungan antara parameter *eigenvalaue*, parameter *alpha*, dan koefisien momentum terhadap *Recognition Rate* yang diperoleh.

Kata kunci: ANN, *Backpropagation*, JST, *Recognition Rate*, *Face Recognition*, PCA.

Abstract

Face recognition can be performed using Backpropagation Artificial Neural Network (ANN) and Principal Component Analysis (PCA). ANN is made to resemble human neural system. Through several parameters on backpropagation, backpropagation characteristics could be known so that errors and epoch would be minimized and Recognition Rate would be enlarged. The experimental results show the relationship between the parameter of eigenvalues, the parameter of alpha, and the momentum coefficien, with the obtained recognition rate.

Keywords*: ANN, Backpropagation, JST, Recognition Rate, Face Recognition, PCA.*

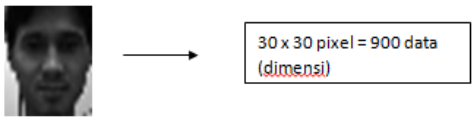
Tanggal Terima Naskah : 10 Juni 2015
Tanggal Persetujuan Naskah : 02 Juni 2016

1. PENDAHULUAN

Salah satu cabang dari *Artificial Intelligence* (AI) adalah Jaringan Saraf Tiruan (*Artificial Neural Network*). Jaringan saraf tiruan telah dikembangkan sejak tahun 1940. Pada tahun 1943, McCulloch dan W.H. Pitts memperkenalkan pemodelan matematis neuron. Tahun 1949, Hebb mencoba mengkaji proses belajar yang dilakukan oleh neuron. Teori ini dikenal sebagai Hebbian *Law*. Tahun 1958, Rosenblatt memperkenalkan konsep perseptron suatu jaringan yang terdiri atas beberapa lapisan yang saling berhubungan melalui umpan maju (*feedfoward*). Konsep ini dimaksudkan untuk memberikan ilustrasi tentang dasar-dasar intelejensia secara umum. Hasil kerja Rosenblatt yang sangat penting

adalah *perceptron convergence theorem* (tahun 1962), yang membuktikan bahwa bila setiap perseptron dapat memilah-milah dua buah pola yang berbeda maka siklus pelatihannya dapat dilakukan dalam jumlah yang terbatas.

Pengenalan wajah dapat dilakukan dengan menggunakan ANN *Backpropagation* dan *Principal Component Analysis* (PCA). Data *Face* diambil dari sekumpulan foto-foto hitam putih yang terdiri atas 10 orang dengan masing-masing sejumlah 10 foto. Data-data *Face* ini menunjukkan nilai pixel pada foto tersebut [1].



Gambar 1. *Data face*

Dalam hal ini, hanya 30x30 parameter yang digunakan sehingga pada proses *training* dibutuhkan *input* sebanyak 900 unit. Data IRIS ini terdiri atas 900 dimensi dan 10 *sample* dimana terdapat sepuluh kelas sehingga banyaknya unit keluaran adalah 10.

2. METODE PENELITIAN

Tujuan penelitian ini adalah untuk mempelajari tentang variasi parameter pembelajaran arsitektur jaringan saraf tiruan *backpropagation* dengan PCA untuk melakukan proses pengenalan wajah.

Tabel 1. Deskripsi dataset penelitian

Nama Data	Jumlah Data	Jumlah Kelas	Jumlah Dimensi	Metode Uji
Face	100	10	Foto 30x30 pixel yang dijadikan matriks 900x1	*Backpropagation* dan PCA

Gambar 2. Data *Face* yang digunakan pada penelitian

Dengan menggunakan arsitektur jaringan saraf tiruan dilakukan percobaan pelatihan dan pengenalan data, dengan rasio jumlah data *training* berbanding jumlah data *testing* adalah 50:50 dan 70:30. Pengamatan dilakukan pada seberapa banyak data yang berhasil dikenali (*recorgnition rate*) dengan parameter-parameter pembelajaran jaringan saraf tiruan yang divariasikan.

Proses pelatihan dan pengujian dilakukan dengan perangkat lunak MATLAB R2009 dengan menggunakan fitur *editor* m.file pada MATLAB. Dengan MATLAB, dihasilkan pula grafik dari setiap eksperimen sehingga dapat diamati perbedaannya.

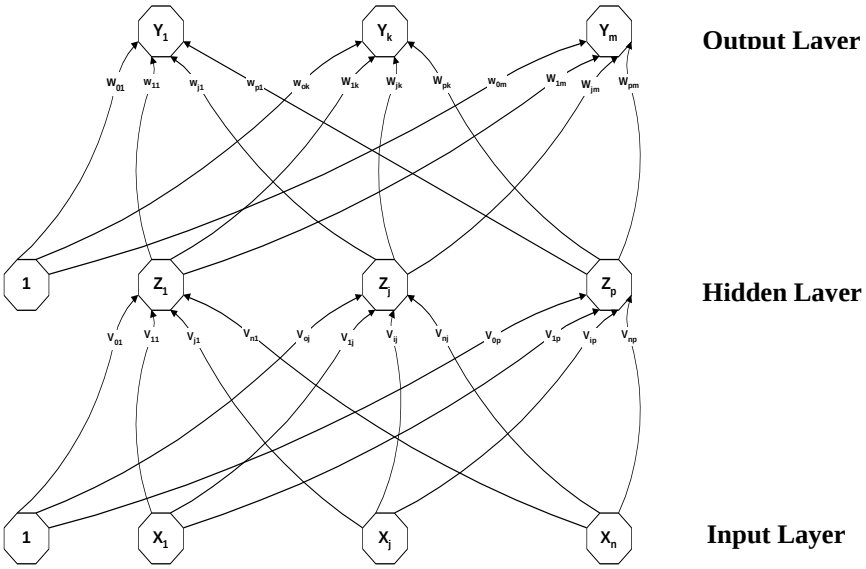
3. PEMBAHASAN

3.1 Pengertian *Backpropagation*

Jaringan Saraf Tiruan (JST) merupakan salah satu sistem pemrosesan informasi atau data yang didesain dengan menirukan cara kerja otak manusia dalam menyelesaikan suatu masalah dengan melakukan proses belajar melalui perubahan bobot sinapsisnya [2]. Salah satu metode yang digunakan dalam JST adalah *Backpropagation*. *Backpropagation* adalah algoritma pembelajaran untuk memperkecil tingkat *error* dengan cara menyesuaikan bobotnya berdasarkan perbedaan *output* dan target yang diinginkan. *Backpropagation* juga merupakan sebuah metode sistematik untuk pelatihan *multilayer* JST karena *Backpropagation* memiliki tiga *layer* dalam proses pelatihannya, yaitu input *layer*, *hidden layer*, dan *output layer*. *Backpropagation* ini merupakan perkembangan dari *single layer network* (Jaringan Lapisan Tunggal) yang memiliki dua layer, yaitu *input layer* dan *output layer*. Dengan adanya *hidden layer* pada *backpropagation* dapat menyebabkan tingkat *error* pada *backpropagation* lebih kecil dibanding tingkat *error* pada *single layer network*. karena *hidden layer* pada *backpropagation* berfungsi sebagai tempat untuk meng-*update* dan menyesuaikan bobot [3].

3.2 Arsitektur *Backpropagation*

Pada *input layer* tidak terjadi proses komputasi, hanya terjadi pengiriman sinyal input ke *hidden layer*. Pada *hidden* dan *output layer* terjadi proses komputasi terhadap bobot dan bias, serta dihitung pula besarnya *output* dari *hidden* dan *output layer* tersebut berdasarkan fungsi aktivasi. Dalam algoritma *backpropagation* ini digunakan fungsi aktivasi *sigmoid biner*, karena *output* yang diharapkan bernilai antara 0 sampai 1.



Gambar 3. Arsitektur ANN Backpropagation

dengan:

V_{ij} = Bobot pada lapisan tersembunyi (*hidden layer*)
V_{oj} = Bias pada lapisan tersembunyi (*hidden layer*)
W_{ij} = Bobot pada lapisan keluaran (*output layer*)
W_{oj} = Bias pada lapisan keluaran (*output layer*)
X = Lapisan masukan (*Input Layer*)
Y = Lapisan keluaran (*Output Layer*)
Z = Lapisan tersembunyi (*Hidden Layer*)

3.3 Algoritma *Backpropagation*

3.3.1 Inisialisasi Bobot

Terdapat dua cara untuk menginisialisasi bobot, yaitu inisialisasi secara *random* dan inisialisasi dengan menggunakan Nguyen-Widrow [4]. Inisialisasi *random* merupakan cara yang paling sering digunakan dalam inisialisasi bobot. Pada inisialisasi bobot secara *random*, bobot diinisialisasi secara acak tanpa menggunakan faktor skala. Pada inisialisasi Nguyen-Widrow, inisialisasi dilakukan dengan memodifikasi inisialisasi *random* dengan menggunakan faktor skala β dengan tujuan untuk mempercepat proses pelatihan.

Algoritma inisialisasi dengan Nguyen-Widrow adalah sebagai berikut:

a. Menentukan besarnya skala β

$$\beta = 0.7(\mathrm{p})^{1/n} \quad \text{.....(1)}$$

dengan p = jumlah unit *hidden* dan n = jumlah unit input.

b. Inisialisasi bobot V_{ij} secara *random* dengan nilai inisialisasi V_{ij} adalah

$$-0.5 \le V_{ij} \le 0.5 \quad \text{.....(2)}$$

c. Menghitung besarnya magnitude bobot V_{ij}

$$\|V_{ij}\| = \sqrt{\left\{\sum_{i=1}^{p} (V_{ij})^2\right\}} \quad \text{.....(3)}$$

d. Meng-*update* bobot V_{ij}

$$V_{ij} = \frac{\beta . V_{ij}}{\|V_j\|} \quad \text{.....(4)}$$

e. Mengatur nilai bias V_{oj} sebesar

$$-\beta \le V_{oj} \le \beta \quad \text{.....(5)}$$

3.3.2 Proses *Feed Forward* dan *Backpropagation*

Pada dasarnya proses algoritma *backpropagation* terdiri dari komputasi maju (*feed forward*) dan komputasi balik (*backpropagation*).

Algoritma proses *feed forward* adalah sebagai berikut.

a. Unit *input* (X_i, i=1,2,…..,n)
   1. Menerima *input* X_i
   2. Mengirimkannya ke semua unit *layer* di atasnya (*Hidden layer*).

b. Unit *Hidden* (Z_j, j=1,2,….,n)
   1. Menghitung semua sinyal *input* dengan bobotnya:

$$z\_in_j = V_{oj} + \sum X_i V_{ij} \quad \text{.....(6)}$$

2. Menghitung nilai aktivasi setiap unit *hidden* sebagai *output* unit *hidden*

$$Z_j = f(z\_in_j)$$
$$f(z\_in_j) = \frac{1}{1+e^{-z\_in_j}} \quad \text{.....(7)}$$

3. Mengirim nilai aktivasi ke unit *output*.

c. Unit *Output* (Y_k, k=1,2,…..,n)

1. Menghitung semua sinyal *input* dengan bobotnya:

$$y\_in_k = W_{ok} + \sum Z_j W_{jk} \quad \text{..... (8)}$$

2. Menghitung nilai aktivasi setiap unit *output* sebagai *output* jaringan.

$$Y_k = f(y\_in_k)$$
$$f(y\_ink) = \frac{1}{1+e^{-y\_ink}} \quad \text{.....(9)}$$

Algoritma proses *backpropagation* adalah sebagai berikut:

a. Unit *Output* (Y_k, k=1,2,…..,m)

1. Menerima pola target yang bersesuaian dengan pola *input*
2. Menghitung informasi *error*:

$$\delta_k = (T_k - Y_k) f'(y\_in_k) \quad \text{.....(10)}$$

3. Menghitung besarnya koreksi bobot unit *output*:

$$\Delta W_{jk} = \alpha \frac{\partial E(W_{jk})}{\partial W_{jk}} = \alpha \partial_k z_j \quad \text{.....(11)}$$

4. Menghitung besarnya koreksi bias *output*:

$$\Delta W_{0k} = \alpha \delta_k \quad \text{.....(12)}$$

5. Mengirimkan δ_k ke unit-unit yang ada pada *layer* di bawahnya, yaitu ke *hidden layer.*

b. Unit *Hidden* (Z_j, j=1,2,….,p)

1. Menghitung semua koreksi *error*:

$$\delta\_in_j = \sum \delta_k W_{jk} \quad \text{.....(13)}$$

2. Menghitung nilai aktivasi koreksi *error*:

$$\delta_j = \delta\_in_j f'(z\_in_j) \quad \text{.....(14)}$$

3. Menghitung koreksi bobot unit *hidden*:

$$\Delta V_{ij} = \alpha \frac{\partial E(V_{ij})}{\partial V_{ij}} = \alpha \delta_j X_i \quad \text{.....(15)}$$

4. Menghitung koreksi *error* bias unit *hidden*:

$$\Delta V_{0j} = \alpha \delta_j \quad \text{.....(16)}$$

c. *Update* bobot dan bias

1. Unit *Output* (Y_k, k = 1,2,…,m)

- Meng-*update* bobot dan biasnya (j = 0,…,p):

$$\begin{aligned} W_{jk} &= W_{jk} + \Delta W_{jk} \\ W_{0k} &= W_{0k} + \Delta W_{0k} \end{aligned} \quad \text{.....(17)}$$

2. Unit *hidden* (Z_j, j = 1,…,p)

- Meng-*update* bobot dan biasnya (i = 0,…,n):

$$\begin{aligned} V_{ij} &= V_{ij} + \Delta V_{ij} \\ V_{0j} &= V_{0j} + \Delta V_{0j} \end{aligned} \quad \text{.....(18)}$$

3.3.3 *Stopping Condition*

Terdapat dua kondisi *stopping* pada algoritma *backpropagation*, yaitu:

a. *Error* < *Error* maksimum
Error adalah perbedaan yang terjadi antara *ouput* terhadap target yang diinginkan. Proses ANN akan berhenti jika besarnya *error* yang terjadi telah bernilai lebih kecil dari nilai *error* maksimum yang telah ditetapkan. Besarnya nilai *error* dihitung dengan menggunakan fungsi *error* kuadratis.

$$E = 0.5 \sum_{k=0}^{k} (T_k - Y_k)^2 \quad \text{.....(19)}$$

b. *Epoch* > *Epoch* maksimum
Epoch adalah suatu langkah yang dilakukan dalam pembelajaran pada ANN. Jika besarnya *epoch* lebih besar dari besarnya *epoch* maksimum yang telah ditetapkan, maka proses pembelajaran akan berhenti.

Kedua kondisi *stopping* tersebut digunakan dengan logika OR. Jadi, kondisi *stopping* terjadi jika besarnya *Error* < *Error* maksimum atau *Epoch* > *Epoch* maksimum.

3.4 Faktor-Faktor dalam Pembelajaran *Backpropagation*

Beberapa faktor yang mempengaruhi keberhasilan algoritma *backpropagation*, antara lain:

1. Inisialisasi bobot
Bobot awal menentukan apakah jaringan akan mencapai *global minima* atau *local minima* kesalahan, dan seberapa cepat jaringan akan konvergen.

2. Laju pembelajaran (*alpha*)
   Laju pembelajaran merupakan parameter jaringan dalam mengendalikan proses penyesuaian bobot. Nilai laju pembelajaran yang optimal bergantung pada kasus yang dihadapi. Laju pembelajaran yang terlalu kecil menyebabkan konvergensi jaringan menjadi lebih lambat, sedangkan laju pembelajaran yang terlalu besar dapat menyebabkan ketidakstabilan pada jaringan.
3. Momentum (*miu*)
   Momentum digunakan untuk mempercepat pelatihan jaringan. Metode momentum melibatkan penyesuaian bobot ditambah dengan faktor tertentu dari penyesuaian sebelumnya. Penyesuaian ini dinyatakan sebagai berikut:

$$\begin{aligned}
\Delta w_{jk}(t+1) &= \alpha\delta_k z_j + \mu\Delta w_{jk}(t) \\
\Delta w_{0k}(t+1) &= \alpha\delta_k z_j + \mu\Delta w_{0k}(t) \\
\Delta v_{ij}(t+1) &= \alpha\delta_j X_i + \mu\Delta v_{ij}(t) \\
\Delta v_{0j}(t+1) &= \alpha\delta_j X_i + \mu\Delta v_{0j}(t)
\end{aligned} \quad \text{.....(20)}$$

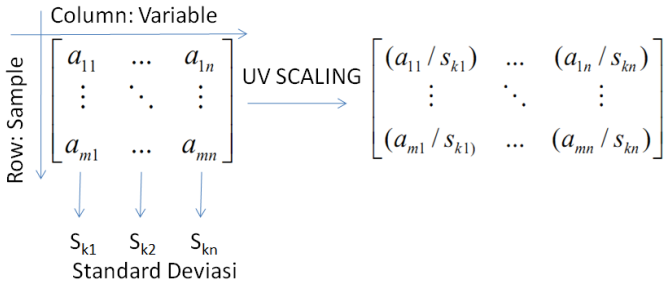
Dengan menggunakan persamaan 17, 18, dan 20, *update* bobot dengan momentum dirumuskan sebagai berikut:

$$\begin{aligned}
W_{jk}(t+1) &= W_{jk}(t) + \alpha\delta_k z_j + \mu\Delta W_{jk}(t-1) \\
W_{0k}(t+1) &= W_{0k}(t) + \alpha\delta_k z_j + \mu\Delta W_{0k}(t-1) \\
V_{ij}(t+1) &= V_{ij}(t) + \alpha\delta_j X_i + \mu\Delta V_{ij}(t-1) \\
V_{0j}(t+1) &= V_{0j}(t) + \alpha\delta_j X_i + \mu\Delta V_{0j}(t-1)
\end{aligned} \quad \text{.....(21)}$$

3.5 *Principal Component Analysis* (PCA)

Principal Component Analysis (PCA) merupakan suatu teknik untuk menyederhanakan suatu *dataset*, dengan mengurangi *dataset* multidimensional ke dimensi yang lebih rendah dengan cara mengambil bagian-bagian dimensi yang penting (*orthogonal* secara vektor). Tujuan dari PCA adalah mengurangi dimensionalitas data sambil tetap memelihara sebanyak mungkin variansi yang muncul dalam *dataset*.
Berikut adalah langkah-langkah dalam melakukan PCA:

a. *Scaling*
   Proses *Scaling* bertujuan untuk membuat suatu *dataset* memiliki persebaran data yang lebih baik. *Output* dari proses *scaling* adalah suatu *Matrix Auto Scaling*, yang merupakan pengurangan dari *Matrix UV Scaling* dengan *Matrix Mean Centering*. *Matrix UV Scaling* adalah *matrix dataset* yang telah dibagi dengan standar deviasi dari setiap dimensi (dalam hal ini dimensi diwakili oleh kolom). Berikut adalah ilustrasi *matrix UV Scaling*:



Gambar 4. Proses *scaling*

Proses *mean centering* adalah proses pencarian nilai rata-rata setiap dimensi (dalam program yang dibuat diwakili dengan nama *variable*). Setelah melakukan proses *mean centering*, *matrix UV scaling* dikurangkan dengan nilai rata-rata sehingga terbentuk *matrix autoscaling*. Berikut adalah potongan program pca.m terkait proses *scaling*.

```
function [SD,X_ave,Xi_auto,T] = pca(Xi,L)
%Xi         matrix input
%L          dimensi akhir
%SD         Standar Deviasi
%X_ave      Rata-rata X
%Xi_auto    Auto Scaling Marix

m = size(Xi,1);     %m x n matrix, dapet m (baris)
n = size(Xi,2);     %m x n matrix, dapet n (kolom)

%Scaling
SD = std(Xi);       %standar deviasi

%UV Scaling process
for sampel_ke = 1:n
    Xi_scaled(:,sampel_ke) = Xi(:,sampel_ke)/SD(sampel_ke);
end

%Mean Centering
X_ave = mean(Xi_scaled);

%Matrix auto
for sampel_ke = 1:n
    Xi_auto(:,sampel_ke) = Xi_scaled(:,sampel_ke)-X_ave(sampel_ke);
end
```

Gambar 5. Program mfile proses *scaling*

Pada program pca.m yang telah dibuat, *matrix* Xi_scaled merupakan *matrix autoscaling*, X_ave merupakan nilai rata-rata untuk proses *mean centering*, dan Xi_auto adalah *matrix autoscaling*.

b. Menghitung *Covariance Matrix*

Covariance matrix dihitung dengan persamaan:

$$S = \left(\frac{1}{m-1}\right) X^T X = \mathrm{cov}(X) \quad \text{.....(22)}$$

Pada MATLAB, nilai *covariance matrix* dapat dicari dengan menggunakan fungsi S=cov(X), dengan X adalah *matrix input* dan S adalah *covariance matrix*.

c. Menghitung Nilai *Eigen* dan Vektor *Eigen*

Pencarian Nilai *eigen* dan vektor *eigen* bertujuan untuk mengetahui sejauh mana dimensionalitas *dataset* dapat dipotong, dimensi yang memiliki *eigen value* yang besar adalah dimensi yang dinilai sangat penting dalam *dataset*. Pencarian *eigen value* dalam MATLAB dapat dilakukan dengan fungsi [P,A]=eig(S), dengan S adalah *matrix input*, P merupakan vektor *eigen* dan A merupakan nilai *eigen*.

d. Menghitung *Score*

Score merupakan nilai akhir *dataset* yang dapat mewakili *dataset*. *Score* dihitung dengan persamaan:

$$[X]([P]^T)^{-1} = [T] \quad \text{.....(23)}$$

dimana X adalah *matrix autoscaling*, P merupakan *eigen vector*, dan T merupakan *score* dari proses PCA. Berikut merupakan potongan program dari proses perhitungan *covariance matrix* sampai mendapatkan *score*.

```
%Covariance Matrix
S = cov(Xi_auto)
[P,A] = eig(S)          %P = eigen vector, A = eigen values
A=sum(A,1);             %biar jadi 1 baris, cuma buat plot, ngeliat eigenval yg nilainya gede
figure;
plot(A,'o');
title('eigenvalue');

%Get Scores
A=A(:,n:-1:1);          %dibalik, supaya bisa motong bagian yang penting
P=P(:,n:-1:1);
M = inv(P');
%cek=size(M)
M = M(:,1:L);           %pemotongan eigen vektor
T = (Xi_auto*M);
end
```

Gambar 6. Program mfile perhitungan *covariance matrix*

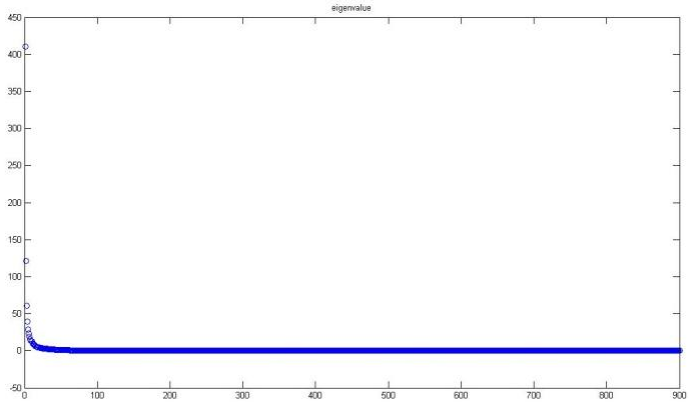
4. HASIL DAN PEMBAHASAN

Pada penelitian ini digunakan data wajah dari 10 orang (10 kelas) dengan masing-masing orang terdiri atas 10 foto yang berbeda. Setiap foto orang memiliki dimensi gambar (foto) 30x30 pixel. Dimensi 30x30 pixel ini akan direntangkan menjadi sebuah matriks 900x1 yang akan diolah oleh *input layer*. Namun, karena dimensi yang cukup besar, akan dilakukan pereduksian dimensi dengan menggunakan metode PCA.

Pada penelitian ini *variable* terikat yang diamati adalah nilai persentase tingkat pengenalan rata-rata (*average recognition rate*), nilai *error*, *epoh*, dan waktu pelatihan. *Variable* bebas yang diubah-ubah adalah jumlah dimensi PCA, *alpha*, koefisien momentum, mode inisialisasi, jumlah *hidden* neuron, jumlah *epoch*, dan nilai *error* minimum. Penelitian dilakukan dengan variasi data *training* : data *testing* 50:50 dan 70:30. Nilai awal untuk masing-masing parameter adalah:
alpha = 0.2; *hidden* neuron = 30; koefisien Momentum=0.2; epoh max= 10000; *error* min= 0.01; metode inisialisasi = nguyen widrow.

Berikut adalah hasil dan analisis dari penelitian yang dilakukan:

a. Pengaruh perubahan dimensi pemotongan PCA
   Pada bagian ini dilakukan variasi perubahan nilai dimensi PCA dengan rentang dimensi dari 10-50 dengan interval 10 dimensi. Pemotongan data ini dilakukan berdasarkan nilai *eigen value* yang semakin kecil ketika dimensi semakin besar. Berikut grafik *eigen value* yang dihasilkan data *Face Recognition*.



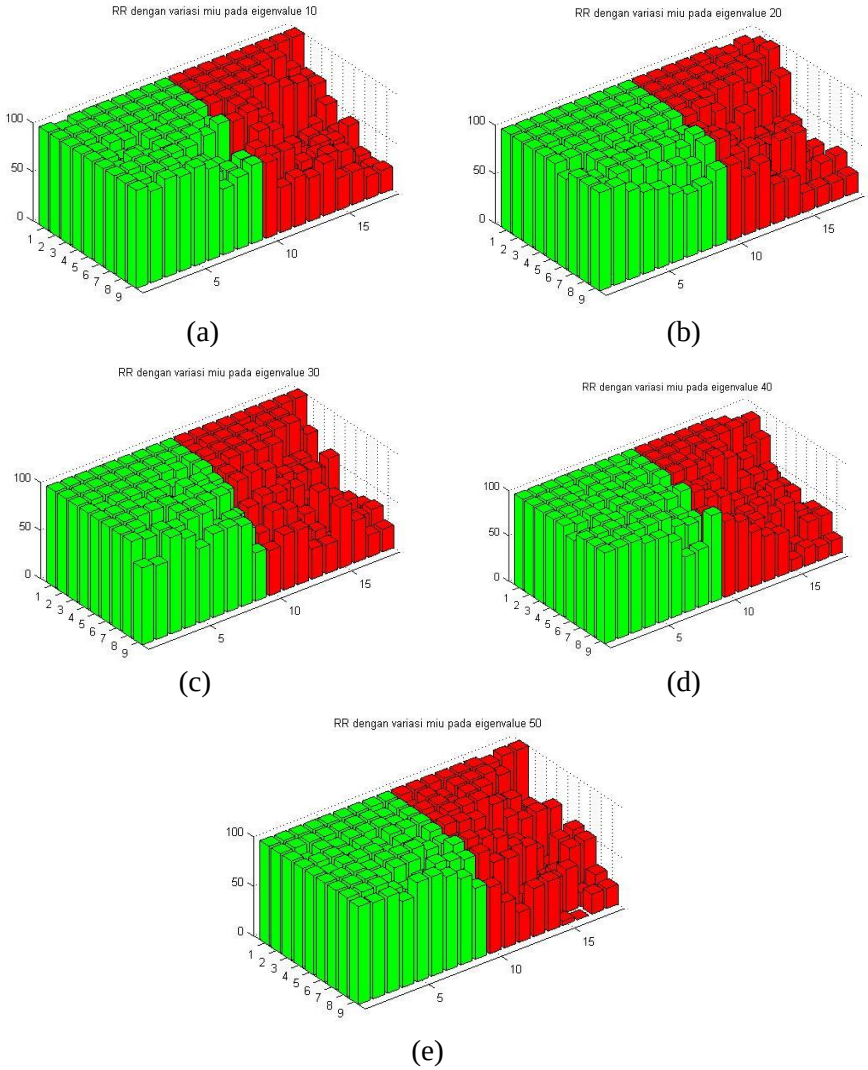
Gambar 7. *Eigen value*

Selanjutnya, diperoleh hasil dari pelatihan dan pengujian sebagai berikut.

Tabel 2. Hasil pengujian variasi parameter *Eigen Value*

eigenvalue = 10		eigenvalue = 20		eigenvalue = 30		eigenvalue = 40		eigenvalue = 50	
RRTR	RRTS	RRTR	RRTS	RRTR	RRTS	RRTR	RRTS	RRTR	RRTS
100	100	100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	96,667	100	100
100	100	100	100	100	100	100	93,333	100	96,667
100	96,667	100	100	100	100	100	100	100	96,667
91	90	100	96,667	100	96,667	100	93,333	100	100
100	93,333	100	96,667	100	100	100	96,667	100	96,667
100	100	100	100	85,714	80	100	90	100	96,667
100	100	100	100	100	96,667	100	100	100	90
100	96,667	90	83,333	100	93,333	100	93,333	97,143	86,667

Keterangan :
RRTR : *Recognition Rate data Training* (Grafik Hijau)
RRTS : *Recognition Rate data Testing* (Grafik Merah)



Gambar 8. Grafik hasil pengujian variasi parameter Eigen value (a) *Eigen value* = 10, (b) *Eigen value* = 20, (c) *Eigen value* = 30, (d) *Eigen value* = 40, (e) *Eigen value* = 50

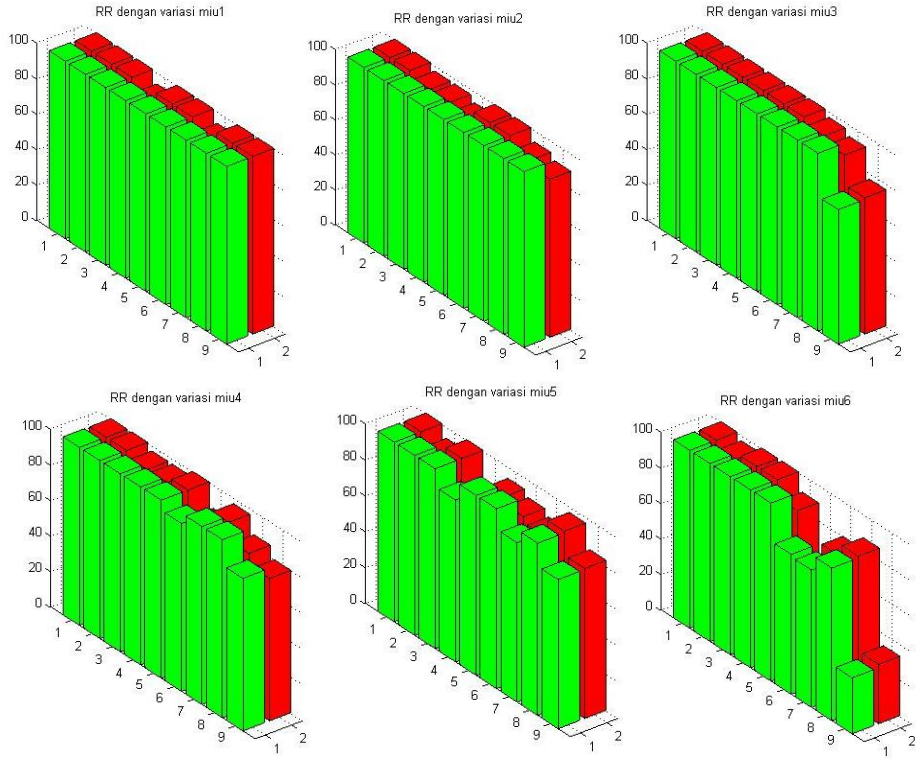
Dari hasil nilai *recognition rate* dapat diamati bahwa nilai pemotongan dimensi PCA ini optimal pada pemotongan 30 dimensi. Hal ini berarti 30 dimensi hasil pemotongan PCA cukup untuk mewakili keseluruhan dimensi (900 dimensi). Dapat dilihat pula bahwa semakin tinggi dimensi PCA waktu pelatihan semakin bertambah. Hal ini sangat logis, karena semakin tinggi dimensi pemotongan, komputasi pada neuron-neuron lebih banyak. Karena pada percobaan didapatkan nilai dimensi optimal adalah 30, untuk percobaan selanjutnya akan digunakan nilai dimensi PCA 30.

b. Pengaruh perubahan *alpha* (Laju Pembelajaran) dan *miu* (momentum)
Pada percobaan ini dilakukan variasi perubahan nilai *alpha* dan *miu* dengan rentang dari 0,1-0,9 dengan interval 0,1. Berikut adalah data hasil dari eksperimen.

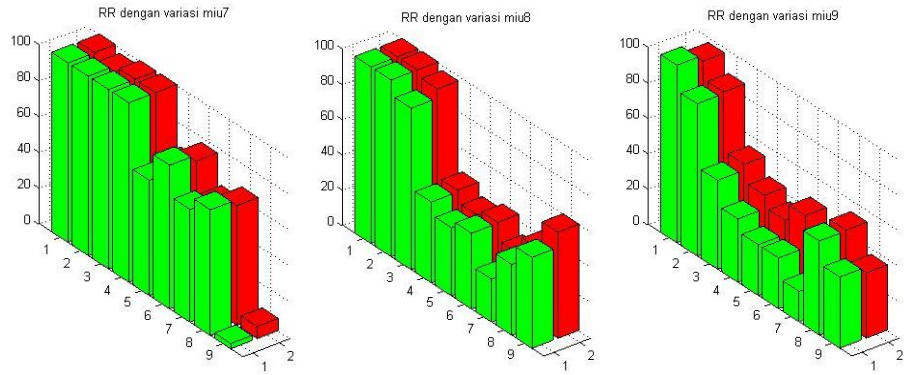
Tabel 3. Hasil Pengujian Pengaruh Alpha dan Miu

miu	0.1		0.2		0.3		0.4		0.5		0.6		0.7		0.8		0.9	
alpha	RRTR	RRTS	RRTR	RRTS	RRTR	RRTS	RRTR	RRTS	RRTR	RRTS	RRTR	RRTS	RRTR	RRTS	RRTR	RRTS	RRTR	RRTS
0.1	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	96,67
0.2	100	100	100	100	100	100	100	100	100	93,33	100	96,67	100	96,67	100	100	86	86,67
0.3	100	100	100	96,67	100	100	100	96,67	100	100	100	100	100	100	91,43	97	50	53,33
0.4	100	93,33	100	96,67	100	100	100	96,67	90	83,33	100	100	100	100	46	46,67	36	43,33
0.5	100	100	100	93,33	100	100	100	100	100	93,33	100	90	64,29	67	39	40	29	36,67
0.6	100	100	100	100	100	100	94,29	86,67	100	90	78,57	60	80	77	42,86	43	29	46,67
0.7	100	93,33	100	100	100	97	100	96,67	88,57	87	77,14	80	63	60	24	33	17	13,33
0.8	100	100	100	93,33	100	93,33	100	87	96	97	85,71	87	70	67	40	43	53	53,33
0.9	100	100	100	90	76	76,67	86	80	83	83	31	33,33	3	6,667	51	60	40	36,67

Keterangan:
RRTR : *Recognition Rate data Training* (grafik hijau)
RRTS : *Recognition Rate data Testing* (grafik merah)



Gambar 9. Grafik Hasil Pengujian variasi parameter laju pembelajaran dan koefisien momentum



Gambar 10. Grafik Hasil Pengujian variasi parameter laju pembelajaran dan koefisien momentum (lanjutan)

Dari hasil nilai *recognition rate* dapat diamati bahwa nilai laju pembelajaran ini optimal pada nilai 0,2. Pada variasi *miu*, terlihat bahwa nilai momentum yang paling optimal terdapat pada nilai 0,2. Dapat dilihat pula bahwa semakin tinggi dimensi PCA waktu pelatihan semakin berkurang. Hal ini sangat logis, karena semakin tinggi laju pembelajaran, kecepatan data untuk menuju konvergen lebih cepat, tetapi kurang stabil. Semakin besar nilai *Alpha* dan momentum, walaupun proses waktu pelatihan semakin cepat, namun *Recognition Rate* akan semakin turun karena *error* dapat masuk ke *local minima.*

5. KESIMPULAN

Berdasarkan percobaan yang telah dilakukan untuk Pengenalan wajah dengan menggunakan metode *Backpropagation* ANN dan PCA, diperoleh kesimpulan sebagai berikut:

a. Nilai *recognition* bervariasi mulai dari 40 % hingga 100%, namun sebagian besar nilai *recognition rate* untuk tiap-tiap kelas adalah 100 %. Nilai optimal berada pada nilai *eigen value* 30, *alpha* 0,2, dan *miu* 0,2.
b. Untuk memperoleh nilai *eigen value* terbaik, diperlukan perhitungan *score* sehingga diperoleh grafik *eigen value*. Pemotongan dimensi dilakukan sesuai dengan banyaknya nilai *eigen value* yang berarti.
c. Semakin kecil *alpha*, semakin besar nilai *recognition rate*.
d. Semakin kecil *miu*, semakin besar nilai *recognition rate*.

REFERENSI

[1]. Kim, Jong Min., Kang, Myung-A. (2010). "A Study of Face Recognition using the PCA and Error-Backpropagation." IEEE.

[2]. Kusumadewi, Sri. "Analisis Jaringan Syaraf Tiruan dengan Metode Backpropagation untuk Mendeteksi Gangguan Psikologi." *Media Informatika*, 2004: 1-14.

[3]. Kusumoputro, Benyamin. (2001). Jaringan Neural Buatan, Ed. 1. Jakarta: Universitas Indonesia.

[4]. Marzuki. "Multilayer Neural Network and the Backpropagation Algorithm." *Lecture Material.* Kuala Lumpur: UTM, n.d.