BAB 2 FUZZY-NEURO LEARNING VECTOR QUANTIZATION (FNLVQ)

Bab ini akan menjelaskan algoritma pembelajaran FNLVQ konvensional yang dipelajari dari berbagai sumber referensi. Pada bab ini dijelaskan pula eksperimen yang dilakukan dengan algoritma ini beserta hasilnya.

2.1 Learning Vector Quantization (LVQ)

Algoritma LVQ adalah algoritma pembelajaran yang melakukan klasifikasi pola ke dalam beberapa kelas/kategori berdasarkan mekanisme kompetisi. Struktur jaringan LVQ adalah jaringan neural dua lapis yang terdiri dari lapisan masukan dan lapisan keluaran (Gambar 2.1). Lapisan masukan mengandung neuron sebanyak dimensi masukan, lapisan keluaran mengandung neuron sebanyak jumlah kelas.

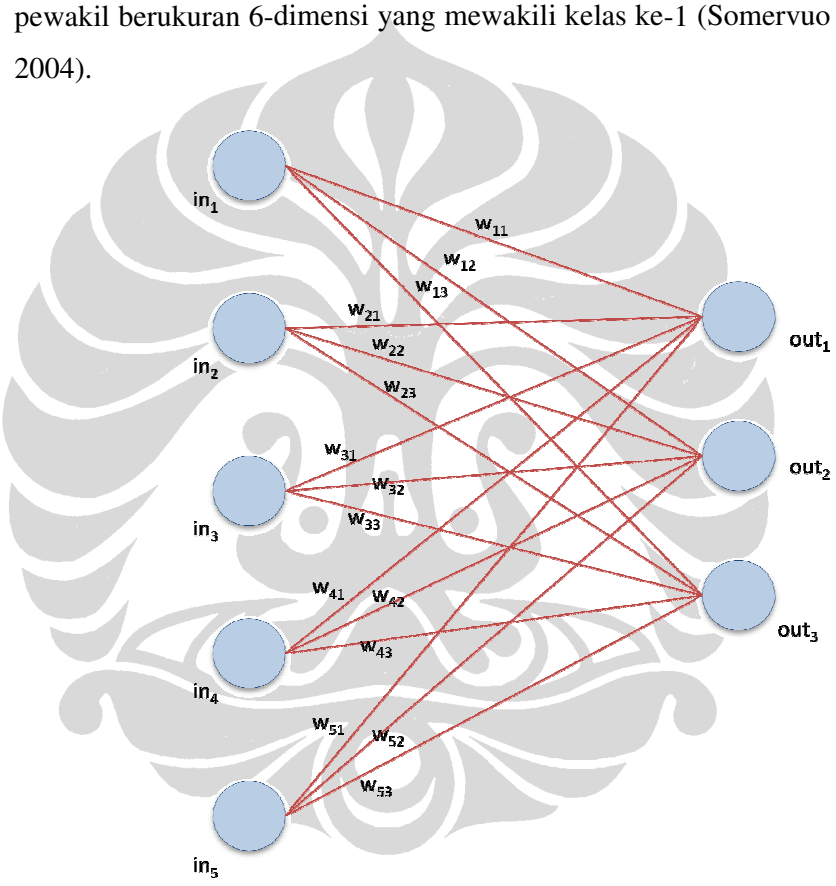
Kedua lapisan dihubungkan oleh penghubung antar setiap neuron yang memiliki suatu bobot tertentu. Bobot-bobot dari semua neuron pada lapisan masukan ke suatu neuron di lapisan keluaran mewakili dimensi-dimensi yang mewakili kelas tersebut. Misalkan saja terdapat 6 neuron masukan dan 3 neuron keluaran. Bobot dari keenam neuron masukan menuju neuron keluaran pertama merupakan vektor pewakil kelas pertama. Vektor pewakil terhadap neuron keluaran ini disebut juga sebagai *vector codebook* atau *reference vector* (Fausette, 1994).

Pada jaringan LVQ, pembelajaran yang dilakukan bersifat *supervised,* dimana pemberian masukan disertai dengan informasi keluaran yang diharapkan. Jaringan senantiasa diarahkan untuk mengeluarkan keluaran yang benar, dan jika jaringan mengeluarkan keluaran yang salah, algoritma ini akan mengetahui kesalahan pada keluaran tersebut dan mengarahkan kembali jaringan untuk bisa mengeluarkan keluaran yang sesuai.

LVQ adalah suatu jaringan yang dapat memproses masukan sesuai suatu fungsi, dan mengeluarkan keluaran berupa klasifikasinya. Dalam hal pengenalan citra wajah, LVQ mampu mengklasifikasikan input citra wajah ke dalam kelas yang

sesuai. Citra wajah dari masing-masing orang akan diklasifikasikan ke dalam 1 kelas yang berbeda.

Dalam contoh pada Gambar 2.1, masukan berupa vektor 6-dimensi yang akan diterima ke dalam jaringan melalui lapisan masukan dengan 6 neuron. Keluaran dari jaringan ini diwakilkan oleh 3 neuron pada lapisan keluaran yang mewakili ketiga kelas klasifikasi. Vektor-vektor berdimensi-6 akan dikelompokkan ke dalam 3 kelas. Vektor dengan elemen (w_{11}, w_{21}, w_{31}, w_{41}, w_{51}, w_{61}) adalah vektor pewakil berukuran 6-dimensi yang mewakili kelas ke-1 (Somervuo dan Kohonen, 2004).



Gambar 2.1 Jaringan LVQ

Pada tahap *training*, algoritma LVQ memproses masukan dengan menerima vektor masukan n-dimensi dan keterangan nomor kelas, kemudian vektor tersebut dihitung jaraknya terhadap semua vektor pewakil untuk k kelas yang ada. Perhitungan jarak antara 2 vektor ini menggunakan *euclidean distance* seperti pada persamaan 2.1. Vektor masukan tersebut lalu diklasifikasikan ke dalam kelas

dengan vektor pewakil yang jaraknya paling dekat dengan vektor masukan. Kelas pemenang pada hasil klasifikasi ini dinamakan *best matching unit* (BMU).

$$e(x, y) = \sum_{i=1}^{n} \sqrt{x_i - y_i} \tag{2.1}$$

dimana

$1 < i < n$

n = jumlah dimensi

Proses pembelajaran ini bersifat *supervised,* sehingga jika BMU yang didapatkan sesuai keterangan nomor kelas pada masukan, maka vektor pewakil kelas tersebut akan disesuaikan agar lebih dekat kepada vektor masukan sedangkan jika BMU tersebut salah, vektor pewakil akan disesuaikan agar lebih menjauh. Proses pengubahan vektor pewakil ini agar sesuai dengan masukan dilakukan sesuai persamaan 2.2 untuk klasifikasi yang benar dan persamaan 2.3 untuk klasifikasi yang salah.

$$w_{ij}(t+1) = v_{ij}(t) + \alpha.(v_{ij}(t) - in_i) \tag{2.2}$$

dimana

$1 < j < k$, k = jumlah kelas

α = laju pembelajaran

$$w_{ij}(t+1) = v_{ij}(t) - \alpha.(v_{ij}(t) - in_i) \tag{2.3}$$

Tahap *training* ini dilakukan secara iteratif untuk laju pembelajaran (α) yang semakin mengecil. Satu iterasi disebut juga sebagai satu *epoch*. Pada setiap *epoch,* semua data pada *training set* dijadikan masukan ke jaringan kemudian vektor pewakil disesuaikan sesuai *training set.* Pada *epoch* berikutnya, seluruh *training set* kembali diproses jaringan, tetapi dengan laju pembelajaran (α) yang lebih kecil. Batas proses pembelajaran ini dibatasi dengan nilai laju pembelajaran (α)

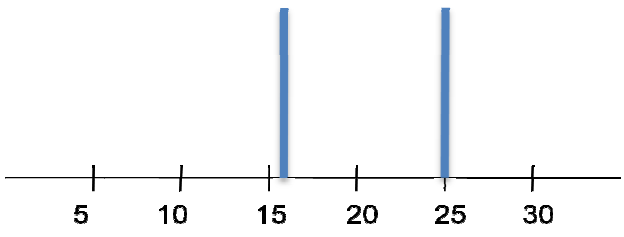
minimal. Setelah laju pembelajaran (α) mencapai nilai minimal, proses *training* dihentikan (Somervuo dan Kohonen, 2004).

Pada tahap *testing,* data diklasifikasikan dengan cara yang sama dengan pada tahap *training*, yaitu dengan cara penghitungan jarak antara vektor pewakil dari setiap kelas. Kelas dengan jarak vektor pewakil paling dekat dengan masukan dinyatakan sebagai BMU. Setelah proses *training* selesai, vektor pewakil akhir diharapkan sudah mewakili dengan baik setiap kelas, sehingga setiap vektor masukan akan dikategorikan ke dalam kelas yang sesuai. Tingkat pengenalan diukur dari jumlah masukan yang dikategorikan dengan benar terhadap keseluruhan data yang digunakan untuk *testing* (Somervuo dan Kohonen, 2004).

2.2 Aritmetika *Fuzzy*

Bilangan atau angka yang biasa digunakan untuk berbagai keperluan dalam kehidupan sehari-hari adalah bilangan *crisp.* Bilangan *crisp* ini adalah bilangan yang hanya memiliki 1 nilai pasti, dan berupa 1 garis pada garis bilangan (Gambar 2.2). Bilangan *crisp* ini adalah angka biasa yang sudah dikenal semua orang, contohnya bilangan 23, 56, atau 920.

Bilangan *fuzzy* adalah konsep bilangan yang mengadaptasi konsep himpunan. Suatu himpunan adalah kumpulan dari objek-objek tertentu, himpunan bilangan adalah sekumpulan bilangan. Suatu bilangan *fuzzy* adalah suatu himpunan (bilangan *fuzzy* bisa disebut juga sebagai himpunan *fuzzy / fuzzy set*) tetapi bilangan-bilangan anggotanya tidak memiliki nilai keanggotaan (μ) yang sama (Mitsuishi, 2004).



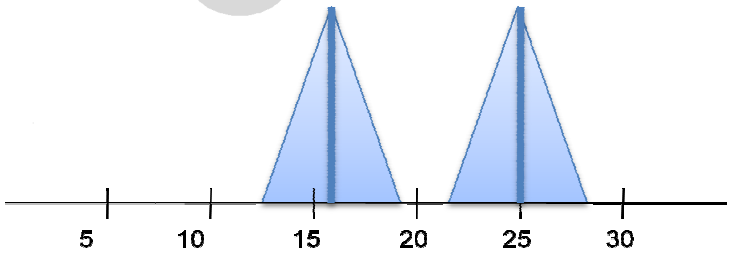
Gambar 2.2 Bilangan *Crisp*

2.2.1 Bilangan *Fuzzy*

Pada himpunan bilangan biasa, suatu bilangan hanya mungkin memiliki dua nilai keanggotaan terhadap himpunan tersebut, 0 atau 1. Nilai keanggotaan bernilai 0 jika bilangan tersebut bukan anggota himpunan, dan jika sebaliknya, bilangan tersebut akan memiliki semua nilai, dari 0 sampai 1. Oleh karena itu, semua bilangan anggota himpunan akan memiliki nilai keanggotaan (μ) yang sama, yaitu 1. Pada bilangan *fuzzy,* bilangan anggotanya memiliki μ yang bervariasi antara nilai 0 dan 1.

Representasi bilangan *fuzzy* pada garis bilangan akan menyerupai bentuk segitiga yang alasnya mewakili bilangan-bilangan anggotanya serta tingginya mewakili nilai μ yang bersesuaian dengan anggotanya. Nilai *crisp* pada Gambar 2.2 akan menjadi nilai-nilai *fuzzy* seperti pada Gambar 2.3. Nilai keanggotaan bilangan anggota suatu himpunan *fuzzy* sesuai konvensinya berada di antara nilai 0 hingga 1, namun untuk kemudahan komputasi pada penelitian ini, nilai μ dikonversi agar bernilai antara 0 sampai 100.

Semua bilangan pada himpunan *fuzzy* tersebut memiliki nilai μ yang berbeda sesuai dengan letaknya pada bilangan *fuzzy* tersebut. Untuk bilangan *fuzzy* yang memiliki rentang pada garis bilangan dari nilai i sampai nilai j, dengan nilai pada bagian tengah bilangan k, maka nilai keanggotaan tertinggi (100) dimiliki oleh bilangan k, sedangkan nilai keanggotaan terendah dimiliki oleh nilai i dan j. Semakin dekat letak bilangan tersebut ke tengah segitiga, semakin tinggi nilai μ yang dimilikinya.



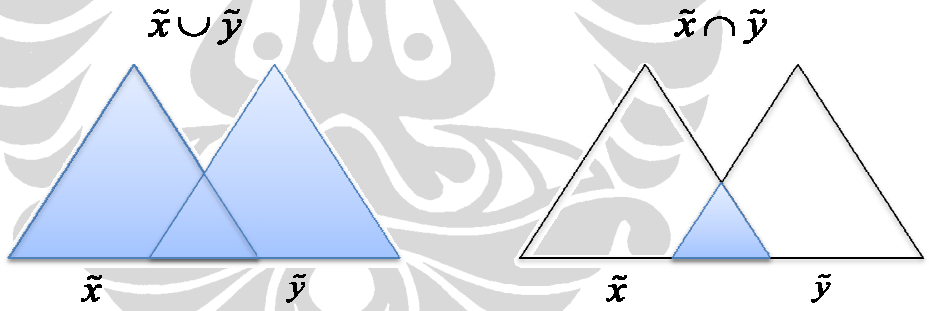
Gambar 2.3 Bilangan *Fuzzy*

Bilangan *fuzzy* dinyatakan sebagai suatu bilangan dengan 3 (tiga) komponen (anggota himpunan minimal yang memiliki nilai μ 0, anggota himpunan tengah yang memiliki nilai μ 1, serta anggota himpunan maksimal yang juga memiliki nilai μ 0) seperti pada persamaan 2.4. Nilai-nilai komponen bilangan *fuzzy* ini dapat disebut sebagai komponen $x^{(1)}$, $x^{(2)}$, dan $x^{(3)}$, selain itu dapat juga di-*refer* sebagai komponen minimal, rata-rata, dan maksimal; atau komponen kiri, tengah, dan kanan (Mitsuishi, 2004).

$$\tilde{x} =< x^{(1)}, x^{(2)}, x^{(3)} > \tag{2.4}$$

2.2.2 Operasi Matematika pada Bilangan *Fuzzy*

Operasi matematis pada bilangan *fuzzy* tentu saja tidak sama dengan operasi matematis pada bilangan *crisp* biasa. Pada bilangan *fuzzy* berlaku operasi-operasi himpunan seperti ∪(*union,* atau gabungan 2 bilangan *fuzzy*) dan ∩ (irisan 2 bilangan). Operasi gabungan dan irisan ini dapat dilihat pada Gambar 2.4. Operasi-operasi himpunan ini tidak dapat diterapkan pada bilangan *crisp.*



Gambar 2.4 Operasi Gabungan dan Irisan pada Bilangan *Fuzzy*

Selain itu, operasi dasar matematika yang semuanya dapat diterapkan pada bilangan *crisp* tidak semuanya dapat diterapkan pada bilangan *fuzzy.* Operasi matematis yang dapat diterapkan pada bilangan *fuzzy* adalah operasi penjumlahan antar bilangan *fuzzy* seperti pada persamaan 2.5, pengurangan antar bilangan *fuzzy* sesuai persamaan 2.6, serta perkalian antara bilangan *fuzzy* dengan bilangan *crisp* seperti pada persamaan 2.7 (Denceux dan Masson, 2004).

$$\tilde{x} + \tilde{y} =< x^{(1)} + y^{(1)}, x^{(2)} + y^{(2)}, x^{(3)} + y^{(3)} > \quad (2.5)$$

$$\tilde{x} - \tilde{y} =< x^{(1)} - y^{(3)}, x^{(2)} - y^{(2)}, x^{(3)} - y^{(1)} > \quad (2.6)$$

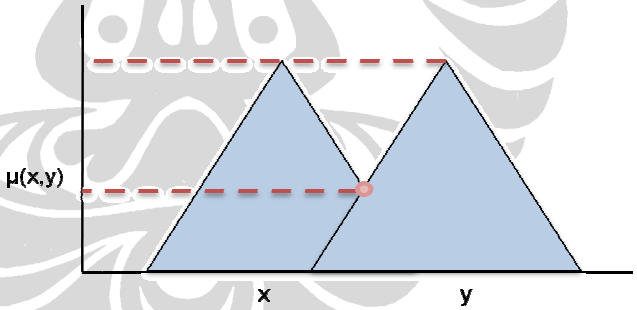
$$a.\tilde{x} =< a.x^{(1)}, a.x^{(2)}, a.x^{(3)} >, \text{jika a>0}$$

$$a.\tilde{x} =< a.x^{(3)}, a.x^{(2)}, a.x^{(1)} >, \text{jika a<0} \quad (2.7)$$

2.2.3 Nilai Similaritas

Misalkan x dan y adalah 2 buah bilangan *fuzzy*, maka dapat dihitung nilai similaritas (μ) antar keduanya sesuai dengan nilai komponen *fuzzy* yang mereka miliki. Nilai similaritas dari 2 buah bilangan *fuzzy* dapat dihitung sebagai nilai maksimum dari irisan kedua bilangan tersebut (persamaan 2.8). Nilai maksimum dari irisan kedua bilangan bisa didapat dari perpotongan tertinggi kedua bilangan *fuzzy* segitiga (Gambar 2.5).

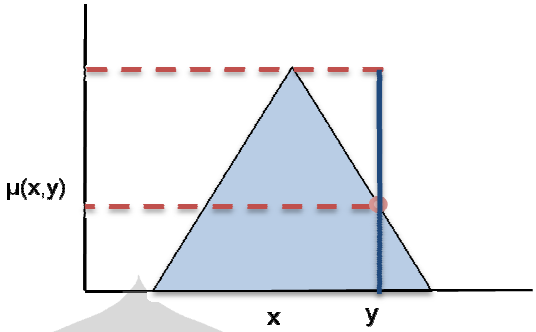
$$\mu(\tilde{x}, \tilde{y}) = \max(\tilde{x} \cap \tilde{y}) \quad (2.8)$$



Gambar 2.5 Nilai Similaritas (1)

Pada penggunaan bilangan *fuzzy* pada penelitian tugas akhir ini, ada pembentukan bilangan *fuzzy* yang memiliki nilai komponen minimal dan maksimal yang sama dengan komponen tengahnya, sehingga bentuk bilangan *fuzzy* berupa garis lurus, seperti bilangan *crisp.* Dalam kasus demikian, nilai similaritas dapat digambarkan juga sebagai perpotongan garis dengan segitiga seperti pada Gambar 2.6. Nilai similaritas dihitung sesuai dengan persamaan 2.9 (Denceux dan Masson, 2004).

$$\mu(\tilde{x},\tilde{y}) = \frac{y - x^{(3)}}{x^{(2)} - x^{(3)}}.100 \tag{2.9}$$



Gambar 2.6 Nilai Similaritas (2)

2.3 FNLVQ

Fuzzy-Neuro Learning Vector Quantization atau FNLVQ memiliki konsep yang sama dengan algoritma LVQ biasa, namun jaringan ini menggunakan bilangan *fuzzy*. Algoritma pembelajaran ini juga merupakan algoritma kompetitif seperti yang dijabarkan oleh Kusumoputro, Jatmiko, dan Krisnadhi (2002). Tujuan akhir jaringan ini adalah untuk melakukan klasifikasi data input, namun jaringan FNLVQ ini memiliki kelebihan dibandingkan algoritma lain, terutama untuk mengenali *outlier,* atau data tidak teregistrasi (Supriyadi, Rahmatullah, dan Sulistio, 2008).

Struktur jaringan FNLVQ kurang lebih sama dengan struktur jaringan LVQ biasa (Gambar 2.1), namun memiliki perbedaan hanya pada jenis bilangan yang digunakan pada bobot (w) atau representasi vektor pewakil. Pada penggunaan bilangan *fuzzy* pada vektor pewakil, masing-masing dimensi akan berupa bilangan *fuzzy,* sehingga vektor pewakil akan berupa sebuah vektor *fuzzy,* yaitu vektor dengan elemen-elemen vektor yang berupa bilangan *fuzzy.*

Jumlah neuron pada lapisan masukan adalah sebanyak dimensi data yang diproses (n) serta jumlah neuron pada lapisan keluaran adalah sebanyak kelas yang akan digunakan untuk klasifikasi data (k). Bobot (w) yang menghubungkan kedua lapisan adalah representasi dari vektor pewakil dari masing-masing kelas pada

lapisan keluaran. Semua bobot penghubung dari n buah dimensi di lapisan masukan menuju kelas ke i, merupakan dimensi-dimensi vektor pewakil untuk kelas i tersebut (persamaan 2.10). Setiap elemen vektor adalah sebuah bilangan *fuzzy* seperti pada persamaan 2.11.

$$\vec{w}_i = < \tilde{w}_{1i}, \tilde{w}_{2i}, ... \tilde{w}_{ni} > \quad (2.10)$$

dimana

i= indeks kelas pada lapisan keluaran

n = indeks dimensi pada lapisan masukan

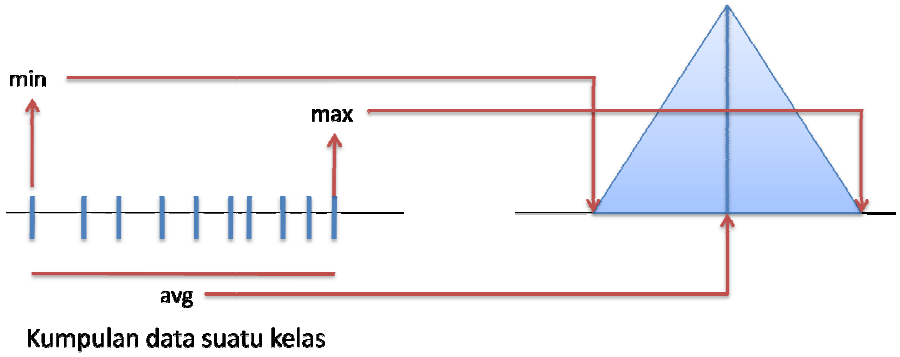
$$\tilde{w}_{ij} = < w_{ij}^{(1)}, w_{ij}^{(2)}, w_{ij}^{(3)} > \quad (2.11)$$

2.4.1 Proses Pembentukan Vektor Pewakil

Data yang dapat diproses menjadi vektor pewakil inisialisasi berupa vektor berdimensi n, sebanyak p buah. Pada penelitian tugas akhir ini, jumlah vektor yang digunakan adalah sebanyak 15 (lima belas) buah untuk setiap kelas. Berikutnya untuk setiap kelas, ke-15 vektor diproses per dimensi dan disimpan nilai-nilai dimensinya, sehingga untuk setiap dimensi akan ada 15 buah data nilai. Selanjutnya untuk setiap dimensi, semua dari data nilai dimensi itu digunakan untuk membentuk suatu bilangan *fuzzy,* kemudian proses ini diulangi untuk setiap dimensi yang ada (sebanyak n).

Pembentukan bilangan *fuzzy* dari 15 buah data adalah dengan cara mencari nilai minimal, rata-rata, dan maksimal, yang kemudian akan menjadi nilai minimal, rata-rata, dan maksimal, menjadi 1 bilangan *fuzzy* (Gambar 2.7).

Pada akhir proses akan ada n buah bilangan *fuzzy* untuk masing-masing dimensi, yang kemudian dijadikan elemen dari sebuah vektor berdimensi n. Vektor ini yang menjadi vektor pewakil awal dari kelas tersebut. Bentuk akhir dari vektor pewakil dapat dilihat pada Gambar 2.8.

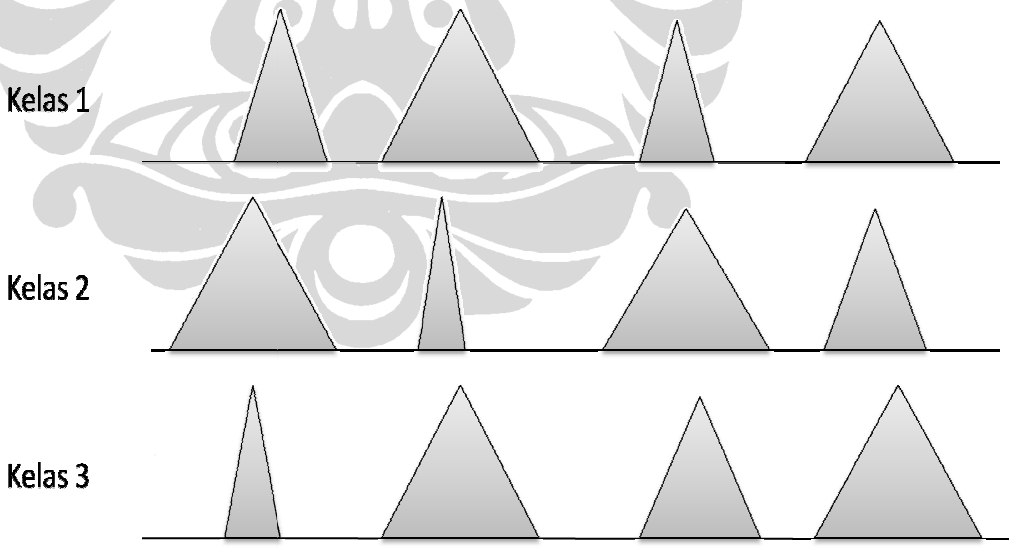


Gambar 2.7 Pembentukan Vektor Pewakil

2.4.2 *Training*

Proses *training* pada jaringan FNLVQ mengikuti langkah-langkah sebagai berikut:

1. Semua data yang digunakan untuk *training* membentuk suatu *training set* yang akan digunakan untuk melatih. *Training* set ini akan dimasukkan ke jaringan berulang kali sampai terpenuhi syarat henti jaringan.

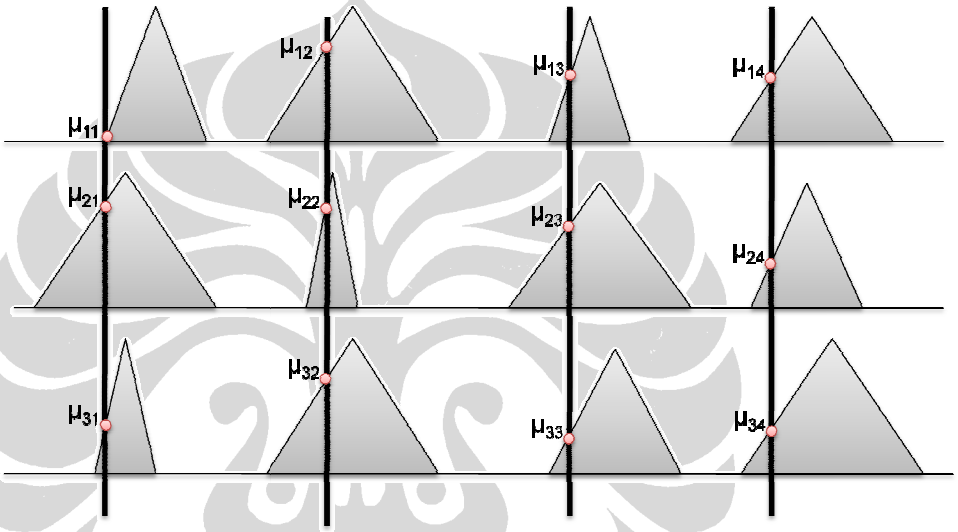


Gambar 2.8 Vektor Pewakil

2. Data sepanjang n dimensi ini berupa data dari 1 kamera saja (1 foto) sehingga setiap dimensi hanya memiliki 1 nilai. Dengan demikian

bilangan *fuzzy* yang dibentuk memiliki nilai yang sama untuk komponen minimal, rata-rata, serta maksimalnya (menyerupai garis lurus seperti bilangan *crisp*). Masukan adalah sebuah vektor *fuzzy* x berdimensi n $\bar{x} = (\tilde{x}_1, \tilde{x}_2, ...\tilde{x}_n)$.

3. Hitung nilai similaritas per dimensi vektor masukan tersebut terhadap vektor pewakil setiap kelas. Penghitungan μ terhadap vektor-vektor pewakil dapat dilihat pada Gambar 2.9.



Gambar 2.9 Pemotongan Data Masukan Terhadap Vektor Pewakil

4. Untuk setiap kelas, masukan $\bar{x} = (\tilde{x}_1, \tilde{x}_2, ...\tilde{x}_n)$ dihitung terhadap vektor pewakil kelas ke-i tersebut $\vec{w}_i =< \tilde{w}_{1i}, \tilde{w}_{2i}, ...\tilde{w}_{ni} >$, akan menghasilkan n buah nilai μ untuk kelas tersebut ($M_i = \{\mu_{i1}, \mu_{i2}, ...\mu_{in}\}$). Setelah itu, untuk kelas tersebut, perlu dicatat nilai similaritas terkecil yang ada (persamaan 2.12).

$$\mu(i) = \min(\mu_{i1}, \mu_{i2}, ...\mu_{in}) \quad (2.12)$$

5. Setelah dilakukan penghitungan nilai similaritas untuk setiap kelas, akan didapat nilai similaritas minimal setiap kelas. Langkah berikutnya adalah menghitung nilai similaritas tertinggi dari kumpulan nilai minimal per kelas (persamaan 2.13). Kelas dengan nilai similaritas

terbesar adalah kelas pemenang. Jika nilai similaritas terbesar adalah 0, maka data *training* dianggap sebagai data tidak teregistrasi.

$$\mu_{final} = \max(\mu(1), \mu(2), ...\mu(k)) \quad (2.13)$$

6. Setelah terjadi klasifikasi, dilakukan pengubahan terhadap vektor pewakil sesuai hasil yang diperoleh. Terdapat 3 kemungkinan yang bisa terjadi, yaitu:

   a. Nilai similaritas terbesar adalah 0. Jika hal ini terjadi, maka data *training* dianggap tidak masuk ke dalam kelas manapun. Setiap dimensi pada vektor pewakil untuk semua kelas, dilebarkan dengan menggunakan sebuah konstanta beta (β) yang bernilai lebih dari 1.

      Jika setiap dimensi berupa bilangan *fuzzy* seperti $\tilde{w}_{ij} =< w_{ij}^{(1)}, w_{ij}^{(2)}, w_{ij}^{(3)} >$, maka setiap bilangan w_{ij} dilebarkan sesuai persamaan 2.14.

$$w_{ij}^{(1)} = w_{ij}^{(2)} - \beta.(w_{ij}^{(2)} - w_{ij}^{(1)})$$

$$w_{ij}^{(2)} = w_{ij}^{(2)}$$

$$w_{ij}^{(3)} = w_{ij}^{(2)} + \beta.(w_{ij}^{(3)} - w_{ij}^{(2)}) \quad (2.14)$$

dimana

β >1

1<i<n, n = jumlah dimensi

1<j<k, k = jumlah kelas

   b. Jika hasil klasifikasi benar,contohnya ketika data masukan adalah data dari kelas 1, kemudian nilai miu terbesar ada pada kelas 1 sehingga hasil kategorisasi jaringan juga kelas 1. Untuk setiap dimensi $\tilde{w}_{ij}$, nilai tengah $w_{ij}^{(2)}$ didekatkan kepada data masukan (persamaan 2.15). Selanjutnya, nilai $w_{ij}^{(1)}$ dan $w_{ij}^{(3)}$ mengikuti.

Selanjutnya w_{ij} dilebarkan seperti pada persamaan 2.14 tetapi hanya untuk j=indeks kelas pemenang. Ini dilakukan untuk semua dimensi pada vektor pewakil pemenang (1<i<n).

$$w_{ij}^{(2)} = w_{ij}^{(2)}.\alpha.((100-\mu_{ij})/100).(x_i - w_{ij}^{(2)}) \quad (2.15)$$

dimana

$\beta > 1$

1<i<n, n = jumlah dimensi

j = indeks kelas pemenang

c. Jika hasil klasifikasi contohnya ketika data masukan adalah data dari kelas 1, tetapi nilai μ terbesar ada pada kelas 2 sehingga hasil kategorisasi jaringan kelas 2. Untuk setiap dimensi $\tilde{w}_{ij}$, nilai tengah $w_{ij}^{(2)}$ dijauhkan dari data masukan (persamaan 2.16) dan nilai $w_{ij}^{(1)}$ dan $w_{ij}^{(3)}$ mengikuti.

   Selanjutnya w_{ij} dipersempit seperti pada persamaan 2.14 tetapi hanya untuk j=indeks kelas pemenang menggunakan konstanta β bernilai kurang dari 1. Ini dilakukan untuk semua dimensi pada vektor pewakil pemenang (1<i<n).

$$w_{ij}^{(2)} = w_{ij}^{(2)}.\alpha.(\mu_{ij}/100).(x_i - w_{ij}^{(2)}) \quad (2.16)$$

dimana

$\beta < 1$

1<i<n

j=indeks kelas pemenang

Ketika semua data pada *training set* sudah dimasukkan ke dalam jaringan dan dilakukan pembelajaran, maka dikatakan telah dilakukan pembelajaran 1 *epoch.*

Pembelajaran dilakukan secara berulang dengan laju pembelajaran (α) yang semakin diperkecil. Ketika nilai α sudah memenuhi syarat henti (ditentukan nilai α minimal sebagai batas pembelajaran), maka fase *training* sudah selesai (Supriyadi, Rahmatullah, dan Sulistio, 2008).

2.4.3 *Testing*

Pada fase *testing,* data yang membentuk *testing set* dimasukkan ke dalam jaringan dengan cara yang sama seperti pada tahap *training*. Setiap data *testing* yang dimasukkan ke dalam jaringan harus diproses dengan cara yang sama dengan data masukan pada tahap *training.* Perbedaan yang ada adalah pada tahap *testing* tidak dilakukan perubahan pada vektor pewakil. Data masukan hanya dikategorikan ke dalam kelas sesuai nilai similaritas terhadap vektor pewakil (Supriyadi, Rahmatullah, dan Sulistio, 2008).

2.4 Fuzzifikasi Data

Fuzzifikasi adalah proses pemetaan suatu nilai *crisp* menjadi bilangan *fuzzy* (Kusumoputro dan Irwanto, 2002). Terdapat beberapa cara yang berbeda untuk melakukan fuzzifikasi. Fuzzifikasi yang dilakukan pada penelitian tugas akhir ini akan dijelaskan pada subbab ini.

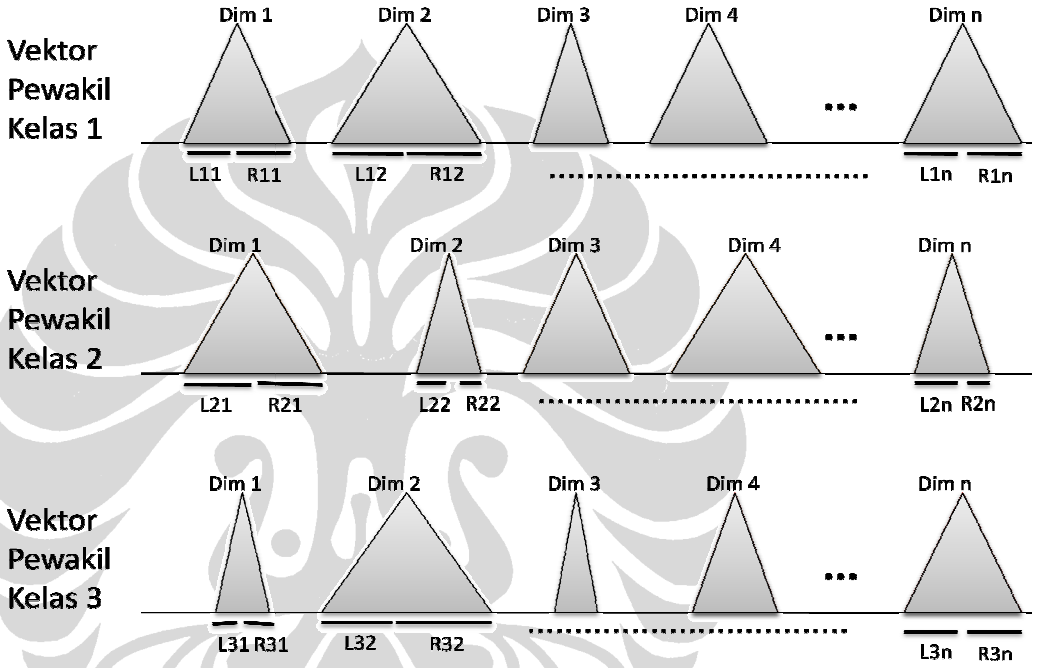
Untuk data *training* dan *testing* pada penelitian tugas akhir ini, data masukan yang digunakan hanya satu untuk setiap data (hanya berasal dari 1 kamera), maka pada vektor *fuzzy* masukan y , elemen y_1, y_2, dan y_3 memiliki nilai yang sama, sehingga bilangan *fuzzy* segitiga yang terbentuk menyerupai suatu bilangan *crisp* biasa, karena sama saja dengan garis lurus. Sebagai variasi untuk pembelajaran, dapat juga dilakukan fuzzifikasi terhadap data *training* dan/atau data *testing* yang hanya memiliki 1 nilai tersebut, agar dapat berbentuk segitiga.

2.3.1 Fuzzifikasi Data *Training*

Dalam melakukan fuzzifikasi pada data *training,* nilai indeks kelas data yang masuk diketahui. FNLVQ adalah sebuah algoritma pembelajaran *supervised,* oleh

karena itu setiap data masukan diketahui kelasnya. Dengan pengetahuan tersebut, fuzzifikasi pada data *training* dilakukan sebagai berikut:

1. Untuk vektor pewakil kelas yang sesuai (karena ini merupakan *supervised learning*, maka setiap masukan *training* diketahui kelas yang benar), dihitung lebar segitiga kiri dan kanan pada setiap dimensi bilangan *fuzzy* segitiga pada vektor pewakil tersebut (Gambar 2.10).



Gambar 2.10 Penghitungan Lebar Kiri dan Kanan Setiap Dimensi pada Setiap Vektor

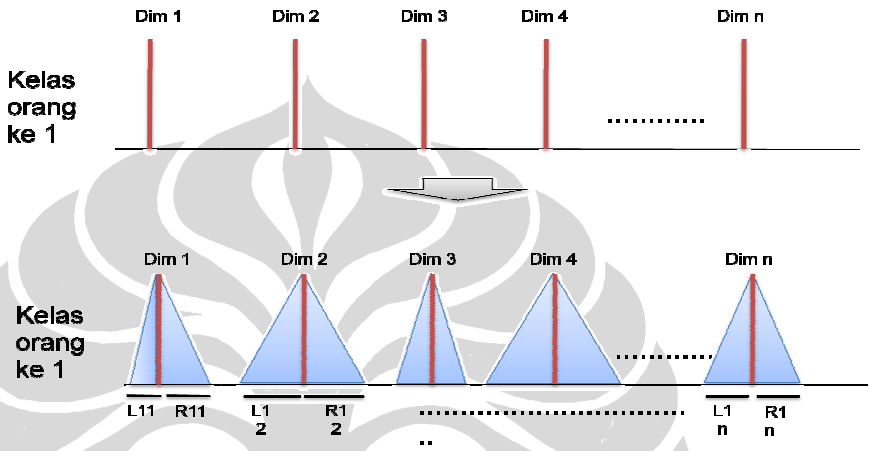
2. Misalkan data masukan berasa dari kelas dengan indeks j, maka vektor pewakil indeks j $\vec{w}_j = (\tilde{w}_{1j}, \tilde{w}_{2j}, ...\tilde{w}_{nj})$, untuk setiap elemen w_{ij} (1<i<n) dihitung $l_{ij} = w_{ij}^{(2)} - w_{ij}^{(1)}$ dan $r_{ij} = w_{ij}^{(3)} - w_{ij}^{(2)}$.

3. Nilai lebar kanan dan kiri pada setiap dimensi itu ditambahkan pada nilai input yang berupa 1 nilai, sehingga setiap dimensi nilai masukan dapat berupa nilai *fuzzy* (Gambar 2.11).

4. Jika data masukan berupa $\vec{x} = (\tilde{x}_1, \tilde{x}_2, ...\tilde{x}_n)$, maka setiap komponen $\tilde{x}_i =< x_i, x_i, x_i >$, karena nilai minimal, maksimal, dan rata-rata adalah

sama (dari 1 citra). Melalui fuzzifikasi, maka nilai x_i dari kelas indeks j menjadi $\tilde{x}_i =< x_i - l_{ij}, x_i, x_i + r_{ij} >$.

Selanjutnya data *training* ini dimasukkan ke dalam jaringan dan mengikuti langkah-langkah yang sama, hanya saja dengan data berbentuk bilangan *fuzzy,* penghitungan nilai similaritas menggunakan persamaan 2.8 pada Gambar 2.5.



Gambar 2.11 Hasil Fuzzifikasi Terhadap Data *Traning* Masukan *Crisp*

2.3.2 Fuzzifikasi Data *Testing*

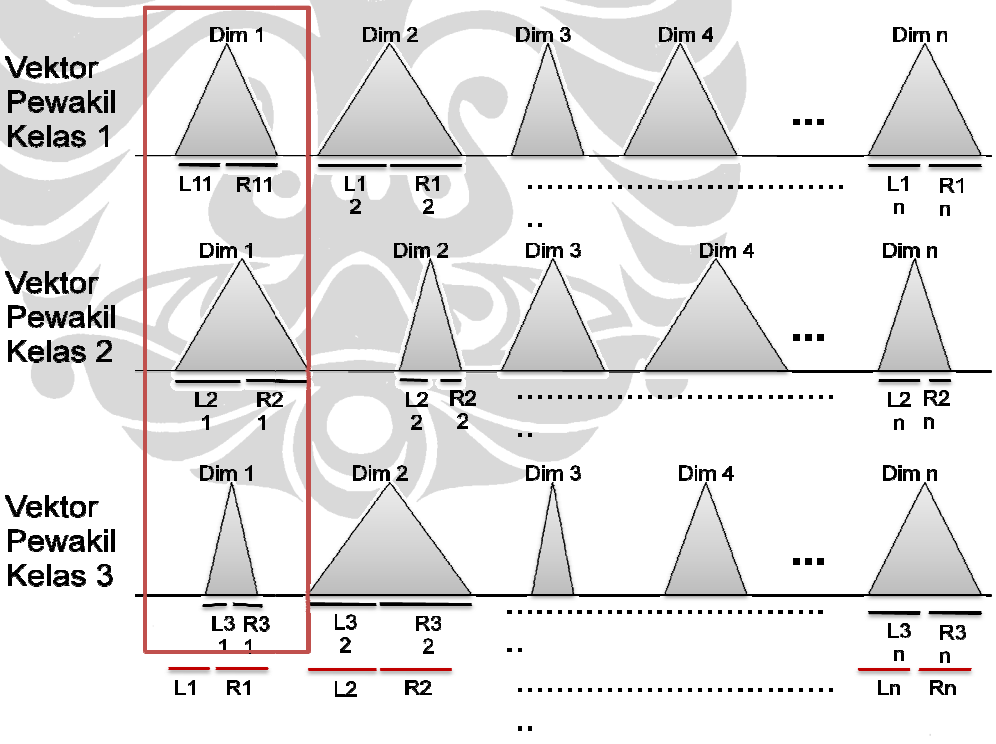
Fuzzifikasi pada data *testing* kurang lebih sama seperti fuzzifikasi pada data *training,* hanya saja pada penerimaan masukan data *testing* tidak diketahui indeks kelas yang benar dari data tersebut. Oleh karena itu, pada fuzzifikasi pada data *testing* dilakukan penghitungan rata-rata nilai kanan dan kiri dari setiap dimensi dari setiap kelas. Data masukan yang *crisp* kemudian dijadikan bilangan *fuzzy* dengan cara yang sama, tidak bergantung pada kelasnya.

Sama seperti pada tahap *training* juga, data *testing* juga berasal dari 1 buah citra saja, sehingga nilai setiap dimensi input hanya terdiri dari 1 nilai saja. Untuk dijadikan masukan pada jaringan, bisa berupa nilai *fuzzy* dengan nilai ketiga komponen yang sama (menyerupai bilangan *crisp*), atau dapat dilakukan fuzzifikasi serupa pada proses *training*, namun karena pada tahap *testing* tidak diketahui data berasal dari kelas yang mana, fuzzifikasi dilakukan melalui langkah-langkah berikut ini:

1. Pada tahap *testing*, tidak dimiliki informasi mengenai indeks kelas asal data, maka tidak dapat diambil data lebar kiri dan kanan setiap dimensi pada vektor pewakil kelas yang bersesuaian.

2. Oleh karena itu, dihitung nilai rata-rata lebar kiri dan nilai rata-rata nilai kanan untuk setiap dimensi dari setiap vektor pewakil. Nilai l_i pada dimensi ke i (1<i<n) adalah rata-rata dari semua nilai l_{ij} untuk semua kelas (1<j<k), atau $l_i = avg(l_{i1}, l_{i2}, ..l_{ik})$ yang dapat dilihat pada Gambar 2.12.

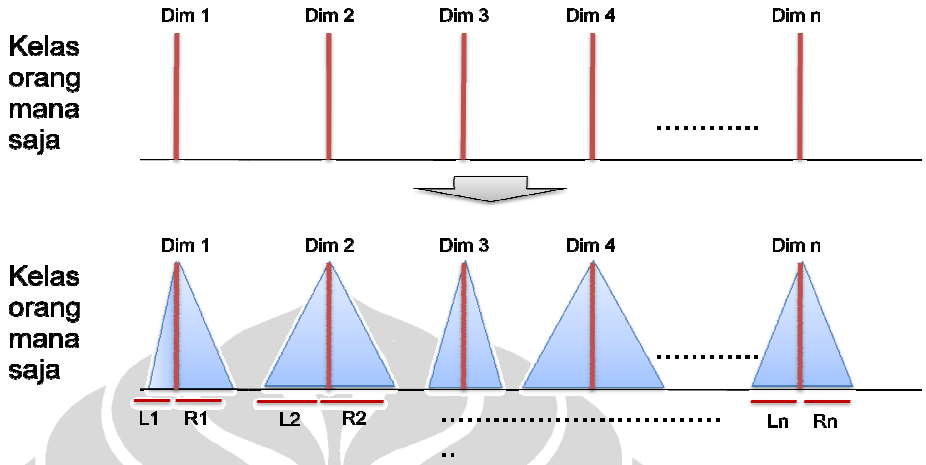
3. Jika data masukan berupa $\vec{x} = (\tilde{x}_1, \tilde{x}_2, ...\tilde{x}_n)$, maka setiap komponen $\tilde{x}_i =< x_i, x_i, x_i >$, karena nilai minimal, maksimal, dan rata-rata adalah sama (dari 1 citra). Melalui fuzzifikasi, maka nilai x_i menjadi $\tilde{x}_i =< x_i - l_i, x_i, x_i + r_i >$ (Gambar 2.13).



Gambar 2.12 Penghitungan Rata-Rata Lebar Kiri dan Kanan Setiap Dimensi

Sama seperti data *training,* selanjutnya data *testing* ini dimasukkan ke dalam jaringan dan mengikuti langkah-langkah yang sama, hanya saja dengan data

berbentuk bilangan *fuzzy,* penghitungan nilai similaritas menggunakan persamaan 2.8 pada Gambar 2.5.



Gambar 2.13 Fuzzifikasi Data *Training*

2.5 Eksperimen terhadap Data Murni

Pada penelitian tugas akhir ini, dilakukan eksperimen pada algoritma pembelajaran FNLVQ yang sudah ada dengan tujuan melihat dan menganalisa hasilnya. Eksperimen dilakukan dengan melatih 10 kelas data citra wajah orang, kemudian menguji dengan 12 kelas data citra wajah orang. Dengan 10 kelas data yang dilatihkan, maka pada lapisan keluaran akan ada 10 neuron dan nilai k pada eksperimen ini adalah 10.

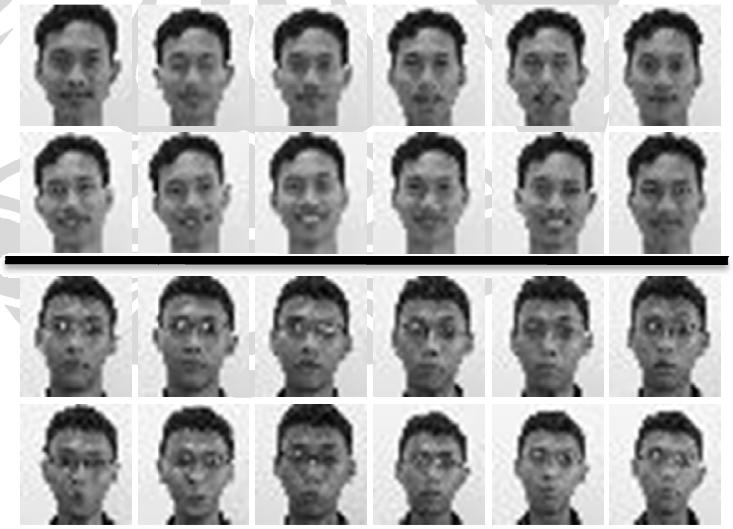
2.5.1 Spesifikasi Data

Pada penelitian tugas akhir ini, eksperimen dilakukan untuk menghitung tingkat pengenalan terhadap data yang berbentuk citra digital. Pengenalan yang dilakukan adalah pengenalan orang (kelas) sesuai citra wajah frontal. Citra wajah frontal yang digunakan adalah citra wajah frontal dengan format citra .jpg, yang diambil dari 12 (dua belas) orang yang berbeda. Citra wajah diambil dari depan wajah orang dengan posisi kepala tidak berubah, tetapi ekspresi bisa berbeda-beda.

Sebelum data tersebut bisa diproses oleh algoritma pembelajaran, data tersebut harus disiapkan dengan menggunakan *image editor* apa saja atau menggunakan *mathematical tools* seperti MATLAB melalui langkah-langkah berikut ini:

1. Citra tersebut diubah menjadi citra *greyscale* (hitam putih).
2. Citra asli yang masih mengandung informasi tidak penting seperti latar di-*crop* agar yang terlihat hanya wajah orang tersebut, mulai dari dahi hingga leher serta dari telinga kiri ke telinga.
3. Citra di-*resize* agar berukuran 30x30 piksel.

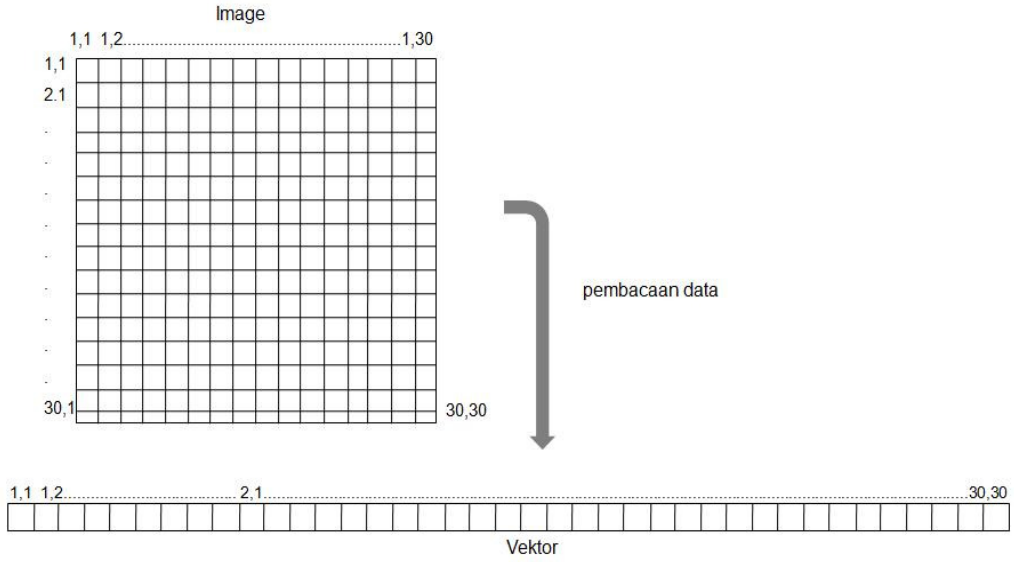
Contoh data akhir yang didapatkan adalah seperti pada Gambar 2.14. Data seperti inilah yang dimaksud sebagai citra asli. Data yang digunakan untuk eksperimen ini adalah data citra wajah 12 orang dengan 50 citra per orang. Citra ini akan dilatihkan untuk 10 orang (10 kelas), dan citra 2 orang lainnya dipergunakan untuk pengujian pengenalan *outlier*. Pada Gambar 2.14 ini diberikan contoh data untuk 2 kelas, masing-masing sebanyak 12 citra.



Gambar 2.14 Contoh Citra untuk Data Masukan

Langkah selanjutnya adalah pembacaan data. Citra adalah sebuah matriks piksel berukuran 30x30. Citra kemudian dibaca oleh program per piksel, dan ketika mencapai akhir dari 1 barisan piksel, pembacaan dilanjutkan langsung ke piksel

pertama pada baris selanjutnya. Hasil akhir yang didapat adalah suatu vektor sepanjang 900 dimensi. Ilustrasi proses ini adalah seperti pada Gambar 2.15.



Gambar.2.15 Pembacaan Citra Menjadi Vektor

2.5.2 Rancangan Eksperimen

Variabel-variabel yang digunakan pada eksperimen adalah:

1. Penerapan fuzzifikasi terhadap data *training* dan *testing*, dengan 3 skenario yang berbeda:

    a. Data *crisp*. Dengan skenario ini bilangan *fuzzy* hanya digunakan untuk dimensi-dimensi pada vektor pewakil. Data *training* dan *testing* hanya berasal dari 1 kamera, sehingga bilangan *fuzzy* yang terbentuk hanya memiliki 1 nilai dan menyerupai bilangan *crisp.*

    b. Data *testing fuzzy*. Data *testing* yang hanya memiliki 1 buah nilai dilebarkan hingga berbentuk bilangan *fuzzy* segitiga melalui fuzzifikasi. Data *training* yang digunakan masih menyerupai bilangan *crisp.* Fuzzifikasi pada data *testing* dimaksudkan untuk memberi kelonggaran pada penghitungan nilai similaritas (Supriyadi, Rahmatullah, dan Sulistio, 2008).

c. Data *training* dan data *testing fuzzy* Fuzzifikasi dilakukan pada data *training* maupun data *testing.* Data *training* dan *testing* yang hanya memiliki 1 nilai dilebarkan hingga berbentuk bilangan *fuzzy* segitiga.

2. Laju pembelajaran (α). Nilai laju pembelajaran ini digunakan sebagai konstanta untuk melakukan perubahan terhadap dimensi-dimensi vektor pewakil. Dengan nilai α yang semakin tinggi, maka pembelajaran akan lebih banyak pada setiap *epoch,* sedangkan sebaliknya pembelajaran akan lebih sedikit namun lebih teliti. Nilai laju pembelajaran yang digunakan adalah: 0.5, 0.4, 0.3, 0.2, dan 0.1.

3. Perbandingan jumlah data yang digunakan masing-masing untuk *training* dan *testing.* Perbandingan yang digunakan adalah *training*:*testing* sebesar 50%:50% dan 70%:30%.

Pada eksperimen ini yang diukur adalah tingkat pengenalan yang mampu dicapai jaringan FNLVQ terhadap citra wajah frontal yang dimasukkan ke dalam jaringan. Satuan pengukur yang digunakan adalah persentase citra yang dikenali terhadap jumlah citra total yang digunakan untuk *testing*.

Tingkat pengenalan yang diukur dibagi menjadi 2, yaitu:

1. Tingkat Identifikasi

Tingkat identifikasi ini adalah kemampuan jaringan untuk mengenali citra sesuai dengan kelasnya. Jaringan dianggap berhasil melakukan identifikasi jika citra masukan milik orang ke-1 diklasifikasikan benar sebagai orang kelas ke-1.

2. Tingkat Klasifikasi

Tingkat klasifikasi ini adalah kemampuan jaringan untuk mengelompokkan citra menjadi data teregistrasi dan data tidak teregistrasi. Jaringan dianggap berhasil melakukan identifikasi jika citra masukan orang yang pernah diregistrasi ke dalam jaringan dikenali sebagai salah satu kelas (tidak harus dikenali sebagai kelas yang benar atau identifikasinya bisa salah), atau data yang tidak pernah teregistrasi

dalam jaringan dapat dipisahkan dan dikategorikan sebagai *outlier.* Oleh karena ini, tingkat klasifikasi dibagi menjadi 2 juga, yaitu pengenalan data teregistrasi dan data tidak teregistrasi.

2.5.3 Hasil Eksperimen

Tingkat identifikasi terhadap citra asli yang dapat dicapai oleh algoritma FNLVQ ini adalah seperti pada Tabel 2.1. Tingkat identifikasi paling tinggi yang bisa dicapai oleh algoritma ini adalah 33% yang diperoleh pada beberapa skenario eksperimen.

Tabel 2.1 Tingkat Identifikasi FNLVQ terhadap Citra Asli

VARIABEL				
Training*: *Testing	**α**	***Crisp***	**Data *Testing Fuzzy***	**Data *Training* dan Data *Testing Fuzzy***
50:50	0.1	30%	30%	10%
	0.2	31%	32%	12%
	0.3	29%	31%	15%
	0.4	30%	13%	12%
	0.5	30%	33%	30%
70:30	0.1	21%	30%	18%
	0.2	21%	21%	2%
	0.3	21%	24%	**0%**
	0.4	19%	13%	8%
	0.5	33%	33%	31%

Untuk skenario yang menggunakan data *crisp* tanpa fuzzifikasi, tingkat identifikasi tertinggi yang dicapai adalah sebesar 33%, yang diperoleh dengan perbandingan data *training:testing* 70%:30% dan laju pembelajaran 0.5. Tingkat identifikasi pada skenario yang lainnya stabil pada kisaran 19% hingga 30%.

Ketika digunakan data *testing fuzzy,* tingkat identifikasi maksimal yang bisa dicapai juga sebesar 33% ketika digunakan variabel eksperimen yang sama. Tingkat identifikasi pada skenario yang lain juga sekitar kisaran yang sama.

Terakhir adalah dengan data *training* dan data *testing fuzzy,* tingkat identifikasi tertinggi yang dapat dicapai tidak setinggi pada skenario data yang lainnya, yaitu sebesar 31%. Keadaan ini dicapai juga pada saat 70%:30% dan laju pembelajaran 0.5. Pada skenario lainnya, tingkat identifikasi juga cenderung menurun, bahkan mencapai 0% pada skenario tertentu.

Berikutnya untuk tingkat klasifikasi terhadap citra asli yang bisa dicapai oleh algoritma FNLVQ adalah seperti pada Tabel 2.2. Pada penggunaan data *crisp,* tingkat klasifikasi data teregistrasi cukup tinggi yaitu pada kisaran 44-78%. Walaupun demikian, tingkat pengenalan data tak teregistrasi hanya mencapai maksimal 22%. Untuk membantu melihat keseimbangan diantara pengenalan keduanya, dihitung rata-rata pengenalannya. Tingkat pengenalan paling seimbang adalah sebesar 61% untuk data teregistasi dan 22% untuk data tidak teregistrasi, dengan tingkat pengenalan rata-rata 42%.

Untuk data *testing* yang berupa bilangan *fuzzy,* tingkat klasifikasi data teregistrasi yang bisa dicapai lebih tinggi dan mencapai keadaan maksimum, yaitu sebesar 83%. Akan tetapi, secara umum tingkat klasifikasi data yang tidak teregistrasi rendah yang mencapai 0% pada beberapa skenario.

Melalui pengamatan rata-rata tingkat klasifikasi data teregistrasi dan tidak teregistrasi, terdapat rata-rata tertinggi yaitu sebesar 47%, dimana tingkat klasifikasi data teregistrasi 82% dan 11% untuk data tidak teregistrasi. Tingkat klasifikasi 11% ini adalah tingkat pengenalan maksimum yang dicapai untuk data tidak teregistrasi, yang masih rendah. Secara umum tingkat pengenalan data tidak teregistrasi masih rendah.

Tabel 2.2 Tingkat Klasifikasi FNLVQ terhadap Citra Asli

VARIABEL		*Crisp*			Data *Testing Fuzzy*			Data *Training* dan Data *Testing Fuzzy*		
Training*: *Testing	**α**	**Data Teregistrasi**	**Data Tak Teregistrasi**	**Rata-Rata**	**Data Teregistrasi**	**Data Tak Teregistrasi**	**Rata-Rata**	**Data Teregistrasi**	**Data Tak Teregistrasi**	**Rata-Rata**
	0.1	78%	9%	44%	85%	3%	44%	27%	73%	50%
	0.2	76%	7%	42%	83%	3%	43%	61%	44%	53%
	0.3	65%	19%	42%	82%	11%	47%	28%	31%	30%
	0.4	62%	18%	40%	83%	**0%**	42%	44%	**0%**	22%
50:50	0.5	61%	22%	42%	68%	2%	35%	70%	20%	45%
	0.1	52%	1%	27%	70%	**0%**	35%	34%	100%	67%
	0.2	44%	2%	23%	62%	2%	32%	10%	100%	55%
	0.3	50%	2%	26%	66%	2%	34%	2%	100%	51%
	0.4	66%	**0%**	33%	83%	**0%**	42%	34%	100%	67%
70:30	0.5	66%	2%	34%	68%	2%	35%	59%	2%	31%

Terakhir adalah untuk data *training* dan data *testing fuzzy,* tingkat pengenalan data tidak teregistrasi cenderung meningkat, bahkan mencapai 100% untuk beberapa skenario. Walaupun demikian tingkat pengenalan data terdaftar pada umumnya menurun. Tingkat pengenalan rata-rata tertinggi adalah sebesar 67%, tetapi pada skenario ini tingkat klasifikasi data teregistrasi cenderung masih rendah. Skenario yang lebih seimbang adalah ketika perbandingan data *training* dan data *testing* 50%:50% dan laju pembelajaran 0.2, dimana tingkat klasifikasi data teregistrasi sebesar 61%, tingkat klasifikasi data tidak teregistrasi sebesar 44%, dan rata-ratanya 53%.

2.6 Eksperimen Terhadap Data yang Mengandung *Noise*

Pada eksperimen sebelum ini, data citra yang digunakan adalah citra yang murni dan ideal. Pada kenyataannya pada implementasi sistem pengenalan wajah, citra yang diinputkan tidak selalu ideal, sering kali citra mengandung *noise* akibat pencahayaan, gerakan objek, dan lain-lain. Oleh karena itu algoritma pengenalan sebaiknya bersifat *robust* terhadap adanya *noise* pada citra.

2.6.1 Noise pada Citra

Citra yang sudah ada merupakan citra yang berada dalam keadaan ideal. *Noise* ditambahkan 6terhadap citra yang ideal tersebut dengan menggunakan *mathematical tools* MATLAB. *Noise* yang digunakan adalah (1) *gaussian blur* 10%, (2) *poisson,* dan (3) *salt & pepper*. Citra yang didapatkan kemudian adalah citra yang terdistorsi (Gambar 2.16).

2.6.2 Rancangan Eksperimen

Pada skenario eksperimen ini, *training* jaringan masih dilakukan dengan data biasa. Pada tahap *testing,* jaringan diuji dengan menggunakan data yang dimanipulasi agar mengandung *noise*. Dengan demikian, dapat diuji kemampuan jaringan untuk mengenali data *noise* dari pembelajaran data biasa. Variabel eksperimen lain yang digunakan adalah (1) perbandingan data *training: testing* dan (2) laju pembelajaran atau α.

Citra Asli

Citra dengan
Gaussian Noise

Citra dengan
Poisson Noise

Citra dengan
Salt & Pepper Noise

Gambar 2.16 Contoh Citra yang Terdistorsi

Untuk variabel penelitian perbandingan data *training:testing,* dibuat dengan mengatur jumlah data yang digunakan agar memenuhi perbandingan 50:50 dan 70:30, tetapi data yang digunakan untuk *testing* semuanya adalah data yang mengandung *noise.* Untuk variabel penerapan fuzzifikasi pada data, untuk eksperimen ini tidak digunakan, dan data yang digunakan adalah data tanpa fuzzifikasi.

2.6.3 Hasil Ekperimen

Tingkat pengenalan yang dicapai oleh algoritma FNLVQ yang konvensional ini dalam pengenalan citra *noisy* adalah seperti pada Tabel 2.3. Tingkat identifikasi yang dicapai sebesar 0% pada banyak skenario.

Walaupun demikian, pada beberapa skenario algoritma FNLVQ ini dapat mencapai tingkat identifikasi yang cukup baik. Untuk citra dengan *gaussian noise,* tingkat identifikasi yang dapat dicapai sebesar 43% pada keadaan optimal. Untuk citra dengan *poisson noise,* tingkat identifikasi maksimal yang dicapai sebesar 68%, dan untuk citra dengan *salt&pepper noise* dicapai tingkat identifikasi maksimal sebesar 54%.

Tingkat klasifikasi terhadap citra *noisy* adalah seperti pada Tabel 2.4. Dari tabel tersebut dapat dilihat bahwa tingkat klasifikasi data tidak teregistrasi bisa

mencapai 100%, tetapi hal ini terjadi ketika tingkat klasifikasi data teregistrasi sebesar 0%, sedangkan ketika data teregistrasi dapat mencapai tingkat klasifikasi diatas 60%, data tidak teregistrasi tidak dapat dikenali. Oleh karena itu walaupun tingkat pengenalan rata-rata sebesar 50%, tetapi sesungguhnya tidak terdapat keseimbangan pengenalan.

Tabel 2.3 Tingkat Identifikasi FNLVQ terhadap Citra *Noisy*

VARIABEL				
Training*: *Testing	**α**	**GAUSSIAN NOISE**	**POISSON NOISE**	**SALT & PEPPER NOISE**
50:50	0.1	**0%**	**0%**	**0%**
	0.2	**0%**	**0%**	**0%**
	0.3	**0%**	**0%**	**0%**
	0.4	43%	67%	54%
	0.5	43%	67%	54%
70:30	0.1	**0%**	**0%**	**0%**
	0.2	**0%**	**0%**	**0%**
	0.3	**0%**	**0%**	**0%**
	0.4	**0%**	**0%**	**0%**
	0.5	40%	68%	47%

2.7 Analisis FNLVQ

Dari hasil yang diperoleh dari eksperimen-eksperimen pada Subbab 2.5 dan 2.6, tingkat pengenalan yang mampu dicapai oleh algoritma FNLVQ seperti yang dapat dilihat pada Tabel 2.1 dan Tabel 2.2 masih belum optimal. Tingkat identifikasi tertinggi untuk hanya sebesar 33%. Walaupun tingkat klasifikasi data sebagai data teregistrasi cukup tinggi (mencapai 85% pada skenario tertentu), namun pada skenario tersebut tingkat klasifikasi data yang tidak teregistrasi (*outlier*) tidak mencapai 10%.

Tabel 2.4 Tingkat Klasifikasi FNLVQ terhadap Citra *Noisy*

VARIABEL		GAUSSIAN NOISE			POISSON NOISE			SALT & PEPPER NOISE		
Training*: *Testing	**α**	**Data Teregistrasi**	**Data Tak Teregistrasi**	**Rata-Rata**	**Data Teregistrasi**	**Data Tak Teregistrasi**	**Rata-Rata**	**Data Teregistrasi**	**Data Tak Teregistrasi**	**Rata-Rata**
	0.1	**0%**	100%	50%	**0%**	100%	50%	**0%**	100%	50%
	0.2	**0%**	100%	50%	**0%**	100%	50%	**0%**	100%	50%
	0.3	**0%**	100%	50%	**0%**	100%	50%	**0%**	100%	50%
	0.4	71%	**0%**	38%	74%	**0%**	56%	62%	**0%**	39%
50:50	0.5	71%	**0%**	38%	74%	**0%**	56%	62%	**0%**	39%
	0.1	**0%**	100%	50%	**0%**	100%	50%	**0%**	100%	50%
	0.2	**0%**	100%	50%	**0%**	100%	50%	**0%**	100%	50%
	0.3	**0%**	100%	50%	**0%**	100%	50%	**0%**	100%	50%
	0.4	**0%**	100%	50%	**0%**	100%	50%	**0%**	100%	50%
70:30	0.5	54%	**0%**	31%	75%	0%	48%	62%	0%	36%

Pada skenario lain, walaupun tingkat klasifikasi terhadap data tidak teregistrasi sempat mencapai 100%, namun pada skenario tersebut tingkat klasifikasi data teregistrasi masih sangat rendah. Jaringan yang diinginkan adalah jaringan yang mampu mengklasifikasikan data teregistrasi maupun data tidak teregistrasi dengan baik. Pada eksperimen ini tidak didapat tingkat klasifikasi yang seimbang antara data teregistrasi dan tidak teregistrasi.

Berikutnya untuk pengenalan citra yang memiliki *noise,* walaupun terdapat beberapa skenario yang mampu mencapai tingkat identifikasi yang tinggi (terutama untuk *poisson noise*), masih diharapkan terjadi peningkatan lagi secara lebih merata. Di samping itu, dalam tingkat klasifikasi antara data tidak teregistrasi yang memiliki *noise,* sistem masih belum mampu membedakan keduanya. Oleh karena itu, pada penelitian tugas akhir ini, dilakukan modifikasi pada algoritma FNLVQ dalam rangka mencapai tingkat pengenalan yang lebih baik. Modifikasi yang dilakukan akan dijelaskan secara rinci pada Bab 3.