BAB 3
PENGENALAN WAJAH DENGAN DIMENSION BASED FNLVQ

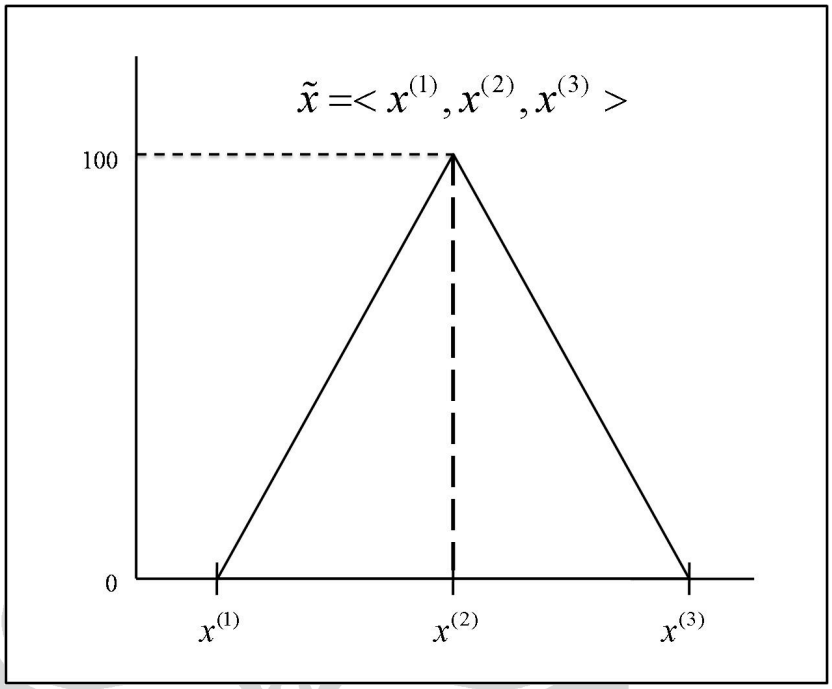
Bab ini menjelaskan tentang pemodelan data masukan yang diterapkan dalam sistem, algoritma FNLVQ secara umum, dan penjabaran mengenai modifikasi metode pengenalan FNLVQ *dimension based.*

Bab ini merupakan hasil penelitian dari Rahadianti, Laksmita (2009) yaitu Pengembangan Algoritma Pembelajaran Berbasiskan Dimensi serta Komparasinya terhadap Pembelajaran Berbasiskan Vektor pada *Fuzzy-Neuro Learning Vector Quantization* untuk Pengenalan Citra Wajah Frontal. Untuk metode pengenalan wajah pada tugas akhir ini, digunakan modifikasi berbasiskan dimensi dari algoritma *Fuzzy Neuro Learning Vector Quantization* (FNLVQ *Dimension Based*). Berikut adalah penjabaran dari algoritma FNLVQ *dimension based.*

3.1. Bilangan *Fuzzy*

3.1.1. Konsep Bilangan *Fuzzy*

Bilangan *fuzzy* adalah suatu konsep bilangan yang mengadaptasi konsep himpunan. Suatu bilangan *fuzzy* adalah suatu himpunan bilangan yang anggotanya tidak memiliki nilai keanggotaan yang sama (Mitsuishi, 2000). Representasi bilangan *fuzzy* pada garis bilangan akan membentuk sebuah segitiga yang pada sumbu *x* akan mewakili bilangan – bilangan anggotanya, serta sumbu *y* akan mewakili nilai keanggotaan suatu bilangan anggota dari himpunan bilangan *fuzzy* tersebut.



Gambar 3. 1 Gambaran Bilangan *Fuzzy*

Nilai keanggotaan dari suatu bilangan anggota bervariasi dari 0 sampai 1. Bilangan yang terletak pada tengah dari segitiga memiliki nilai keanggotaan tertinggi , dan semakin jauh letaknya dari tengah , semakin rendah pula nilai keanggotaan dari bilangan tersebut dalam himpunan *fuzzy*. Gambar 3.1 adalah gambaran dari bilangan *fuzzy*. Untuk kemudahan komputasi, rentang variasi nilai keanggotaan dibuat menjadi dari 0 sampai 100.

3.1.2. Perhitungan Nilai Similaritas

Misalkan ada 2 buah bilangan *fuzzy* x dan y, 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 3.1). Nilai maksimum dari irisan kedua bilangan bisa didapat dari perpotongan tertinggi kedua bilangan *fuzzy* segitiga.

$$\mu(\tilde{x}, \tilde{y}) = \max(\tilde{x} \cap \tilde{y}) \tag{3.1}$$

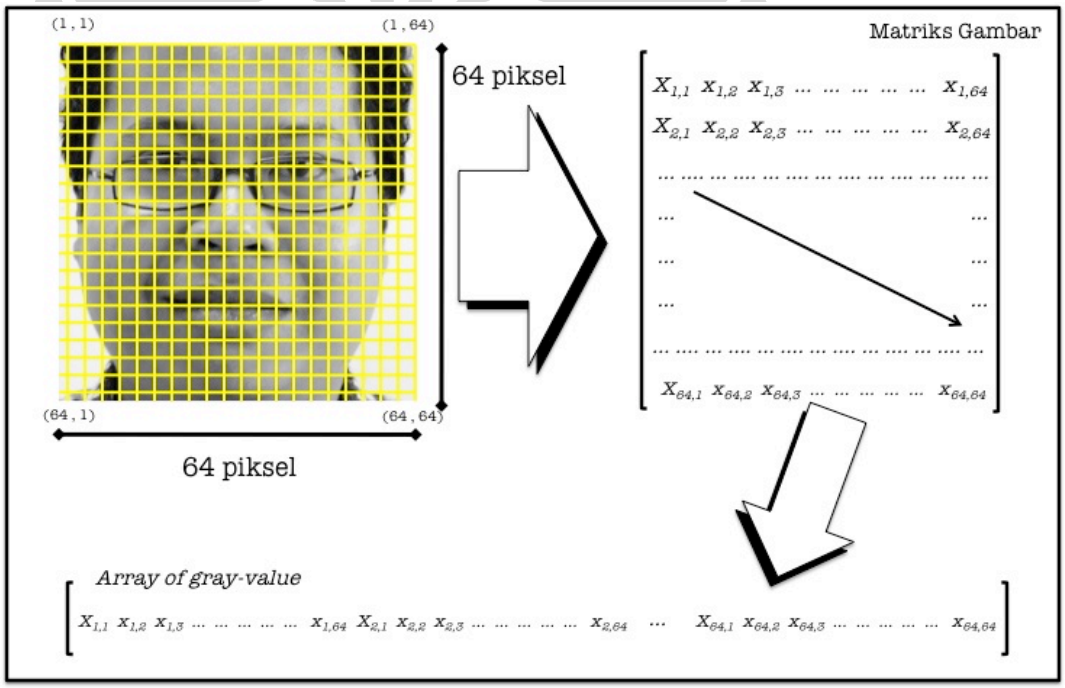
Pada penggunaan bilangan *fuzzy* pada penelitian tugas akhir ini, ada pembentukan bilangan *fuzzy* yang memiliki nilai komponen minimal dan maksimal yang sama

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. Nilai similaritas dihitung sesuai dengan persamaan 3.2 (Denceux dan Masson, 2004).

$$\mu(\tilde{x},\tilde{y}) = \frac{y - x^{(3)}}{x^{(2)} - x^{(3)}}.100 \qquad (3.2)$$

3.2. Pemodelan Masukan

Data masukan yang diolah adalah berkas gambar JPEG yang diorganisasikan dalam direktori sesuai dari userID dari pengguna yang terdaftar dalam sistem. Semua data masukan siap olah akan diproses seperti gambar 3.2



Gambar 3. 2 Pengolahan Data Masukan

Setiap data masukan siap olah berukuran 64 * 64 piksel akan dibaca *gray value* dari tiap pikselnya sebagai nilai *double* sehingga menghasilkan sebuah matriks dua dimensi berukuran 64 * 64 yang berisikan *gray value* dari tiap piksel. Dari

matriks dua dimensi tersebut, dibentuk sebuah array satu dimensi berukuran 1 * 4096 dari nilai *grayvalue* dari setiap piksel data masukan.

Setiap data akan melalui proses fuzzifikasi untuk membentuk bilangan *fuzzy*. Fuzzifikasi adalah suatu proses pemetaan suatu nilai *crisp* menjadi bilangan *fuzzy* (Kusumoputro dan Irwanto, 2002). Data pelatihan dan data pengujian melewati proses fuzzifikasi yang berbeda. Untuk data pelatihan, proses fuzzifikasi yang dilalui adalah:

1. Untuk setiap vektor pewakil dihitung lebar segitiga kiri dan kanan pada setiap dimensi bilangan *fuzzy* segitiga pada vektor pewakil tersebut.
2. Misalkan data masukan berasal 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 masukan yang berupa 1 nilai, sehingga setiap dimensi nilai masukan dapat berupa nilai *fuzzy*.
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 satu buah citra). Melalui fuzzifikasi, maka nilai x_i dari kelas indeks j menjadi $\tilde{x}_i =< x_i - l_{ij}, x_i, x_i + r_{ij} >$.

Untuk data pengujian, proses fuzzifikasi yang dilalui adalah:

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

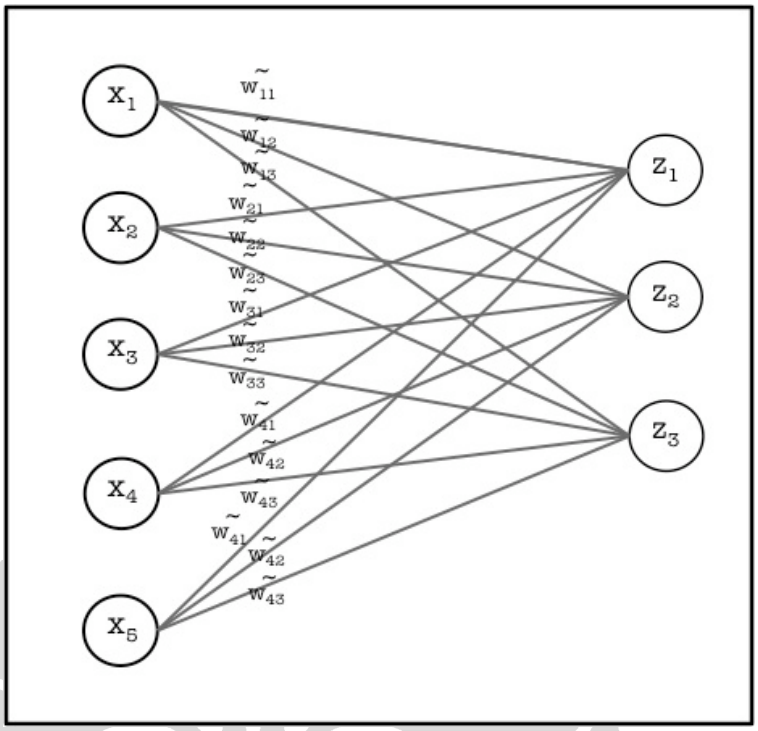
dimensi ke i (1 < i < n) adalah rata-rata dari semua nilai l_{ij} untuk semua kelas (1<j<k), atau $l_i = avg(l_{i1}, l_{i2}, ..l_{ik})$.

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 satu citra). Melalui fuzzifikasi, maka nilai x_i menjadi $\tilde{x}_i =< x_i - l_i, x_i, x_i + r_i >$.

3.3. Algoritma FNLVQ

Algoritma FNLVQ adalah algoritma pengenalan *supervised* yang berdasarkan algoritma LVQ dengan penerapan teori bilangan *fuzzy* didalamnya. Tujuan akhir jaringan ini adalah untuk melakukan klasifikasi data masukan, namun jaringan FNLVQ ini memiliki kelebihan dibandingkan algoritma lain, terutama untuk mengenali *outlier,* atau data tidak terdaftar

Struktur dari jaringan FNLVQ terdiri dari dua lapis. Lapisan pertama merupakan lapisan masukan yang mengandung *neuron* sebanyak dimensi dari data masukan dari jaringan. Lapisan kedua adalah lapisan keluaran yang mengandung *neuron* sebanyak jumlah kelas. Kedua lapisan dihubungkan oleh bobot yang mewakili vektor pewakil dari masing –masing kelas. Semua bobot yang terhubung ke satu *neuron* lapisan keluaran dari lapisan masukan adalah representasi dari dimensi – dimensi yang ada dalam kelas tersebut. Bobot – bobot yang menghubungkan lapisan masukan dan lapisan keluaran ini disebut sebagai vektor pewakil. Gambar 3.3 adalah gambaran dari struktur jaringan FNLVQ.



Gambar 3. 3 Struktur Jaringan FNLVQ

Jumlah *neuron* yang berada pada lapisan masukan adalah sebanyak dimensi data yang akan diproses (n), jumlah neuron pada lapisan keluaran adalah sebanyak kelas (k) dan bobot (w) yang menghubungkan kedua lapisan adalah representasi dari vektor pewakil dari masing-masing kelas pada lapisan keluaran. Semua bobot penghubung dari n pewakil untuk kelas i tersebut (persamaan 3.3). Setiap elemen vektor adalah sebuah bilangan *fuzzy* seperti pada persamaan 3.4.

$$\vec{w}_i =< \tilde{w}_{1i}, \tilde{w}_{2i}, ...\tilde{w}_{ni} > \quad (3.3)$$

dimana

- $1 < i < k$ = indeks kelas pada lapisan keluaran
- n = indeks dimensi pada lapisan masukan

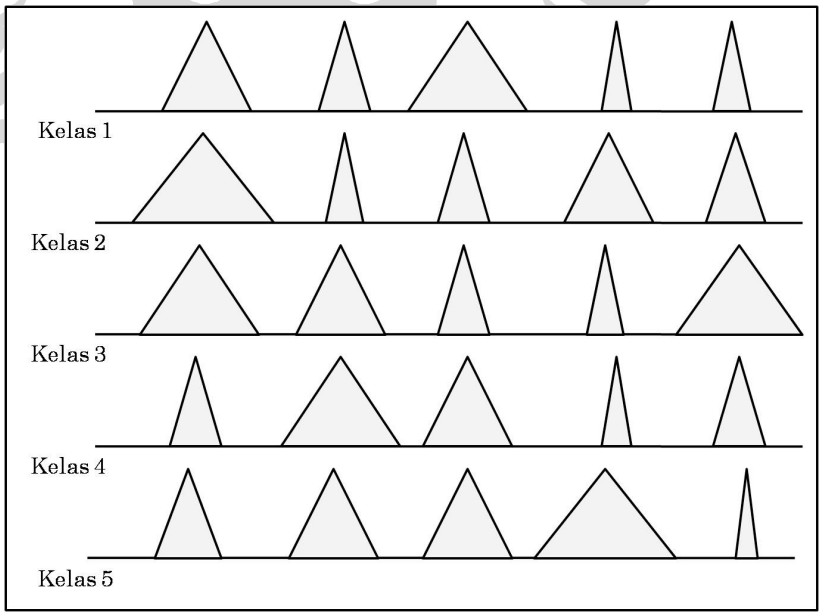
$$\tilde{w}_{ij} =< w_{ij}^{(1)}, w_{ij}^{(2)}, w_{ij}^{(3)} > \quad (3.4)$$

Dimana setiap elemen vektor pewakil adalah bilangan *fuzzy* dengan tiga elemen

fuzzy.

Tahap selanjutnya adalah 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 lima puluh buah untuk setiap kelas. Berikutnya untuk setiap kelas, keseluruhan vektor diproses per dimensi dan disimpan nilai-nilai dimensinya, sehingga untuk setiap dimensi akan ada lima puluh 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 lima puluh buah data tersebut adalah dengan cara mencari nilai minimal, rata – rata dan nilai maksimal. Ketiga nilai tersebut akan menjadi 1 buah bilangan *fuzzy*. 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.



Gambar 3. 4 Gambaran Vektor Pewakil pada FNLVQ

Pada proses pelatihan, akan mengikuti langkah – langkah sebagai berikut:

1. Hitung nilai similaritas per dimensi vektor masukan tersebut terhadap vektor pewakil setiap kelas. Dari setiap kelas disimpan nilai similaritas yang terkecil dalam kelas tersebut.

2. Dari kumpulan nilai similaritas minimal seluruh kelas, dicari nilai tertinggi dari seluruh nilai tersebut. Kelas pemilik nilai similaritas minimal tertinggi adalah kelas pemenang. Jika nilai similaritas tertinggi yang didapat adalah nol, maka data pelatihan dianggap sebagai data tidak terdaftar.

3. Setelah klasifikasi selesai, maka tahap selanjutnya adalah proses pengubahan vektor pewakil. Ada tiga kondisi yang perlu diperhatikan:

    a. Nilai similaritas minimal tertinggi bernilai nol. Dalam kasus ini setiap dimensi pada setiap vektor pewakil untuk semua kelas, dilebarkan dengan menggunakan sebuah konstanta beta (β) yang bernilai lebih besar dari 1.

    b. Jika klasifikasi benar, maka untuk setiap dimensi vektor pewakil tersebut, nilai tengah vektor pewakil didekatkan kepada data masukan dan nilai keanggotaan minimal dan maksimal dari vektor pewakil mengikuti. Selanjutnya vektor pewakil tersebut dilebarkan ,tetapi hanya untuk yang memiliki indeks kelas pemenang. Pelebaran dilakukan dengan menggunakan konstanta beta (β) yang bernilai lebih besar dari satu. Hal ini dilakukan untuk semua dimensi pada vektor pewakil pemenang

    c. Jika klasifikasi salah, maka untuk setiap dimensi vektor pewakil tersebut, nilai tengah vektor pewakil dijauhkan kepada data masukan dan nilai keanggotaan minimal dan maksimal dari vektor pewakil mengikuti. Selanjutnya vektor pewakil tersebut dipersempit, tetapi hanya untuk yang memiliki indeks kelas pemenang. Penyempitan dilakukan dengan menggunakan konstanta beta (β) yang bernilai lebih kecil dari satu.

Langkah – langkah tersebut diulang sampai syarat henti (*stopping condition*) sudah terpenuhi.

Untuk proses pengujian, tahapan yang dilalui mirip dengan proses pelatihan, hanya tidak terjadi langkah pengubahan vektor pewakil. Data masukan dikategorisasikan ke dalam kelas sesuai nilai similaritas terhadap vektor pewakil.

3.4. Modifikasi FNLVQ menjadi FNLVQ *Dimension Based*

Demi mencapai tingkat pengenalan yang lebih tinggi, modifikasi dilakukan pada algoritma FNLVQ. Pada algoritma pembelajaran yang sebelumnya dilakukan dengan memperbaharui vektor pewakil per vektor, dimodifikasi agar terjadi pembaharuan per satuan dimensi, dimana setiap dimensinya berbentuk bilangan *fuzzy* segitiga. Oleh karena itu algoritma hasil modifikasi ini akan disebut dengan nama *dimension-based* FNLVQ

Perbedaan utama dari FNLVQ *dimension based* dengan FNLVQ biasa adalah pada tahap pengubahan vektor pewakil. Untuk memungkinkan proses ini, maka dimensi yang akan diubah disimpan nilainya. Berikut adalah langkah – langkah yang terdapat pada tahap pengubahan vektor pewakil FNLVQ *dimension based:*

a. Nilai similaritas minimal tertinggi adalah nol. Jika hal ini terjadi, maka data *training* dianggap tidak masuk ke dalam kelas manapun. Pada FNLVQ *dimension based* hanya dilebarkan vektor pewakil untuk kelas yang sesungguhnya saja.

b. Jika klasifikasi benar, maka pendekatan dan pelebaran vektor pewakil hanya pada dimensi yang bersangkutan. Pelebaran dilakukan dengan menggunakan konstanta beta (β) yang bernilai lebih dari satu.

c. Jika klasifikasi salah, maka penjauhan dan penyempitan vektor pewakil hanya pada dimensi yang bersangkutan. Penyempitan dilakukan dengan menggunakan konstanta beta (β) yang bernilai kurang dari satu.