BAB II
LANDASAN TEORI

2.1. Citra Digital

Citra digital merupakan suatu fungsi yang memiliki nilai-nilai berupa intensitas cahaya pada tiap titik pada bidang yang telah diquantisasikan (diambil sampelnya pada interval diskrit). Titik-titik pada hasil sampling suatu citra digital tersebut disebut *pixel* (*picture element*). Sebuah citra merupakan kumpulan *pixel* yang membentuk sebuah matrik, dimana pada sebuah *pixel* mengandung suatu informasi. Citra digital merupakan fungsi intensitas cahaya f (*x*, *y*), dimana nilai *x* dan *y* adalah koordinat spasial dan nilai fungsi tersebut pada setiap titik (*x*, *y*) adalah tingkat kecemerlangan citra pada titik tersebut (Pratama, Mulya, dan Utami 2016). Terdapat tiga jenis citra yang biasanya digunakan dalam pemprosesan suatu citra. Diantranya yaitu citra berwarna (RGB), citra berskala keabuan (*grayscale*), dan citra biner.

2.1.1. Citra Berwarna (RGB)

Citra berwarna RGB merupakan jenis citra yang menyajikan warna dalam bentuk komponen warna R (*red*/merah), G (*green*/hijau), dan B (*blue*/biru). Setiap titik atau *pixel* pada citra warna memiliki nilai warna yang dinyatakan dalam nilai parameter R, G dan B. Setiap parameter warna menggunakan 8bit yang nilainya memiliki rentangan antara 0 sampai dengan 255. Berdasarkan jangkauan nilai parameter R, G dan B pada citra warna, maka kemungkinan warna yang bisa disajikan mencapai 16.777.216 warna (256 x 256 x 256) (Sujito dan Yunus 2016).

2.1.2. Citra Berskala Keabuan (*Grayscale*)

Citra berskala keabuan (*grayscale*) merupakan jenis citra yang direpresentasikan dengan nilai gradasi dari warna hitam ke warna putih. Gradasi warna dalam citra *grayscale* menghasilkan efek warna abu-abu. Warna keabuan tersebut dinyatakan dengan nilai intensitasnya yang memiliki rentangan antara 0

sampai dengan 255. Nilai 0 menyatakan hitam pekat dan nilai 255 menyatakan putih terang (Sujito dan Yunus 2016). Persamaan yang digunakan untuk mendapatkan nilai citra *grayscale* adalah sebagai berikut (Sholeh 2013):

$$I_{BW}(x,y) = (Red\ x\ 0{,}2126) + (Green\ x\ 0{,}7152) + (Blue\ x\ 0{,}0722) \quad (2.\ 1)$$

Dimana $I_{BW}(x,y)$ = nilai piksel *black and white* titik (x,y).

2.1.3. Citra Biner

Citra biner merupakan citra yang setiap *pixel* dinyatakan dengan nilai 0 atau 1. Nilai 0 menyatakan warna hitam pekat dan nilai 1 menyatakan warna putih terang (tidak mengenal gradasi warna keabuan). Citra jenis ini banyak dipakai dalam pemrosesan citra tertentu, misalnya untuk kepentingan memperoleh tepi bentuk, jumlah, keliling dan luasan suatu objek dalam suatu citra (Sujito dan Yunus 2016). Citra biner merupakan hasil dari pengolahan citra keabuan atau *grayscale* dengan menggunakan fungsi berikut (Kusumanto dan Tompunu 2011):

$$I_{Bin}(x,y) = \begin{cases} 0 & I_{BW}(x,y) < T \\ 1 & I_{BW}(x,y) \geq T \end{cases} \quad (2.\ 2)$$

Dimana:

1. $I_{BW}(x,y)$ = nilai piksel *gray* titik (x,y).
2. $I_{Bin}(x,y)$ = nilai piksel *binary* titik (x,y).
3. T adalah nilai *threshold*.

2.2. Pengolahan Citra Digital

Pengolahan citra digital (*Digital Image Processing*) merupakan sebuah ilmu yang mempelajari tentang teknik dalam mengolah suatu citra. Citra yang dimaksud yaitu berupa gambar diam (foto) maupun gambar bergerak (yang berasal dari *webcam*). Sedangkan digital yaitu memiliki maksud bahwa pengolahan citra/gambar ini dilakukan secara digital dengan menggunakan bantuan komputer (Kusumanto dan Tompunu 2011). Suatu citra digital yang melalui suatu proses citra digital nantinya menghasilkan suatu citra digital baru. Termasuk didalamnya yaitu perbaikan citra (*image restoration*) dan suatu peningkatan kualitas citra (*image*

enhancement). Sedangkan analisis citra digital (*digital image analysis*) yaitu menghasilkan suatu keputusan atau data yang dimana didalamnya termasuk yaitu pengenalan pola (*pattern recognation*) (Arsy, Nurhayati, dan Martono 2016).

2.3. Pengenalan Pola

Pola merupakan entitas yang terdefinisi dan dapat didefinisikan melalui ciri-cirinya (*feature*). Ciri-ciri tersebut digunakan sebagai pembeda suatu pola dengan pola lainnya. Ciri yang bagus yaitu ciri yang mempunyai daya pembeda yang tinggi, sehingga pengelompokan pola berdasarkan ciri dapat dilakukan dengan tingkat keakuratan yang tinggi. Sebagai contoh beberapa pola dan cirinya dapat dilihat pada Tabel 2.1 sebagai berikut (Munir 2004):

Tabel 2. 1 Pola dan Ciri

Pola	Ciri
Huruf	Tinggi, tebal, titik sudut, lengkungan garis, dll
Suara	Amplitudo, frekuensi, nada, intonasi, warna, dll
Tanda tangan	Panjang, Kerumitan, tekanan, dll
Sidik jari	Lengkungan, jumlah garis, dll

Pengenalan pola (*pattern recognition*) merupakan proses klasifikasi dari objek atau pola menjadi beberapa kategori atau kelas. Tujuan dilakukannya pengenalan pola ini adalah untuk menentukan kelompok atau kategori dari suatu pola berdasarkan ciri-ciri yang dimilikinya. Ciri tersebut digunakan untuk membedakan antara pola yang satu dengan lainnya (Ananggadipa, Hidayatno, dan Zahra 2014). Pengenalan pola memiliki berbagai macam bagian. Salah satunya adalah pengenalan pola karakter seperti huruf dan angka, pengenalan suara, pengenalan sidik jari, pengenalan retina dan iris mata, pengenalan tulisan tangan, dan pengenalan telapak tangan.

2.4. *Portable Network Graphics* (.PNG)

Portable network graphics (PNG) merupakan salah satu format citra yang dirancang agar menjadi lebih baik daripada format yang sebelumnya. PNG mempunyai beberapa fitur diantaranya mendukung kedalaman warna hingga 48 bit (*true color images + alpha*), mendukung untuk *web browser*, mendukung *tranparency*, dan mendukung *interlacing*. Format PNG merupakan solusi kompresi yang *powerfull* dengan warna yang lebih banyak. Berbeda dengan JPG yang menggunakan teknik kompresi yang menghilangkan data, format PNG menggunakan kompresi yang tidak menghilangkan data (*lossles compression*). Kelebihan PNG lainnya adalah adanya warna transparan dan alpha. Warna alpha memungkinkan sebuah gambar transparan, tetapi gambar tersebut masih dilihat mata seperti samar-samar atau bening (Sujatmiko, Isnanto, dan Handoyo 2011).

2.5. *Preprocessing*

Preprocessing merupakan suatu proses yang bertujuan untuk memperbaiki kualitas dari suatu citra dengan menggunakan teknik-teknik pengolahan citra (Munir 2004). Tahapan pertama yang dilakukan yaitu dengan melakukan *cropping* pada citra yang bertujuan untuk mengambil citra huruf hiragana saja dengan ukuran 300 x 300 piksel. *Cropping* dilakukan dengan cara manual menggunakana aplikasi atau *tools* yang bernama *adobe photoshop CS3*. Berikutnya melakukan *resize* atau pengurangan terhadap ukuran citranya degan ukuran 40 x 40 piksel. Selanjutnya melakukan konversi warna citra menjadi RGB kemudian dikonversi lagi menjadi citra *grayscale* lalu dikonversi lagi menjadi citra biner, dan yang terakhir mengubah citra tersebut menjadi matrik 1 dimensi.

2.6. *Principal Component Analysis* (PCA)

PCA merupakan cara untuk mengidentifikasi pola-pola dalam data berkorelasi, dan mengekspresikan data sedemikian rupa sehingga menyoroti Persamaan dan perbedaan. PCA dapat digunakan sebagai mereduksi dimensi suatu data tanpa mengurangi karakteristik dari data tersebut secara signifikan (Mulyadi dan Suryanto 2016). PCA atau disebut juga transformasi *Karhunen loeve*

merupakan suatu teknik yang digunakan untuk menyederhanakan suatu data, dengan cara mentransformasi linear sehingga terbentuk sistem koordinat baru dengan variansi maksimum. (Ardiansyah 2009).

PCA merupakan teknik handal untuk mengekstraksi suatu citra dengan dimensi yang banyak. Sebuah citra memiliki kumpulan piksel yang mengandung sebuah informasi tertentu. Piksel pada citra membentuk sebuah matriks yang akan diolah menggunakan metode PCA dengan mengambil informasi dari citra tersebut sehingga informasi yang diperoleh menjadi lebih sederhana (Puspitaningrum, Kemala Sari, dan Susilo 2014). Pada penelitian ini proses ekstraksi ciri menggunakan PCA terbagi menjadi dua bagian yaitu pertama ekstraksi ciri pada citra data latih dan ekstraksi ciri pada data uji. Berikut langkah-langkah dari metode PCA untuk mengekstraksi data latih (Puspitaningrum, Kemala Sari, dan Susilo 2014):

1. Proses menghitung matriks rata-rata (ψ).
   Menggunakan Persamaan (2.3) sebagai berikut:

$$\psi = \frac{\sum_{i=1}^{M} \Gamma_i}{M} \qquad (2.3)$$

   Keterangan:
   ψ = Matriks rata-rata (*means*)
   M = Banyak data di data set
   Γ_i = Nilai pada setiap piksel.
   Tujuan dilakukannya tahapan ini untuk mengurangi dimensi yang akan dihitung pada proses selanjutnya.

2. Proses menghitung matriks normalisasi (Φ).
   Menggunakan Persamaan (2.4) sebagai berikut:

$$\phi_i = \Gamma_i - \psi \qquad (2.4)$$

   Masing-masing nilai pada setiap piksel dari (Γ_i) dikurangkan dengan nilai rata-rata (ψ).

3. Proses menghitung matriks kovarian (C).
   Menggunakan Persamaan (2.5) sebagai berikut:

$$C = \Phi \times \Phi^T \quad (2.5)$$

Keterangan:

C = Matriks kovarian.

Φ = Matriks selisih / normalisasi.

Φ^T = Matriks *transpose* dari matriks selisih / normalisasi.

Tujuan dilakukannya pencarian nilai matrik kovarian untuk mempermudah proses pencarian nilai *eigen* dan *eigen vector*.

4. Proses menghitung *eigen vector* (υ) dan *eigen value* (λ) dari matriks C.

   Pada matriks kovarian (C) yang berisi ciri utama, nantinya akan didapatkan nilai *eigen* dan *eigen vector* yang selanjutnya disebut dengan *eigen face*. Nilai *eigen* atau *eigen value* (λ) merupakan nilai karakteristik dari suatu matrik berukuran n x n. Sedangkan *eigen vector* (υ) merupakan vektor kolom bukan nol yang apabila dikalikan dengan suatu matrik berukuran n x n, akan menghasilkan vektor lain yang memiliki nilai kelipatan dari *eigen vector* itu sendiri. Berikut Persamaan (2.6) untuk menghitung nilai *eigen* atau *eigen value* dari matrik kovarian (C):

$$C\,\upsilon = \lambda\,\upsilon$$
$$C\,\upsilon - \lambda\,\upsilon = 0$$
$$(C - \lambda\, I)\; atau\; \det(\lambda I - C) = 0 \quad (2.6)$$

   Sedangkan Persamaan yang digunakan dalam menghitung nilai *eigen vector* (υ) harus dilakukan terlebih dahulu proses subtitusi dari *eigen value* (λ) ke dalam Persamaan (2.7) sebagai berikut:

$$(\lambda I - C)\upsilon = 0 \quad (2.7)$$

   Keterangan:

   λ = Nilai *eigen* atau *eigen value*

   υ = Vektor *eigen* atau *eigen vector*

   C = Matrik kovarian

   I = Matriks identitas

5. Mengurutkan mengurutkan nilai *eigen value* (λ) dan *eigen vektor* (υ) dari besar ke kecil berdasarkan urutan nilai eigen.

6. Proses perhitungan matriks *eigen face* dengan menggunakan Persamaan (2.8) sebagai berikut:

$$Eig_F = \upsilon \times \phi \quad (2.8)$$

Keterangan:

Eig_f = *Eigen face*

7. Melakukan proses penghitungan *project images* dengan menggunakan Persamaan (2.9) sebagi berikut:

$$Project_{images} = \phi \times {Eig_f}^T \quad (2.9)$$

Keterangan:

Eig_f^T = Matriks *transpose* dari nilai *eigen face*

Setelah melewati tahapan ekstraksi ciri pada PCA dengan mendapatkan nilai dari *project image*, maka proses berikutnya yaitu mereduksi matrik *project image* sebanyak nilai *n* yang dapat disesuaikan dengan kebutuhan. Maksud dari mereduksi matrik sebanyak nilai *n* yaitu membuang sebagian kolom dari sisi kanan matrik dan hanya mengambil beberapa kolom dari kolom matrik paling kiri dari matrik 2 dimensi tanpa mempengaruhi baris pada matrik tersebut. Selanjutnya matrik *project image* yang telah melalui tahap reduksi, selanjutnya memasukkan matrik tersebut kedalam proses klasifikasi pada algoritma LVQ3.

Setelah melakukan proses ekstraksi terhadap citra data latih, maka tahapan selanjutnya melakukan proses ekstraksi ciri terhadap data uji. Berikut langkah-langkah metode PCA dalam mengekstraksi data uji (Puspitaningrum, Kemala Sari, dan Susilo 2014):

1. Menghitung matriks normalisasi dari citra data uji.

   Pada tahapan ini membutuhkan nilai *mean* dari citra data latih dan nilai biner dari data uji yang akan digunakan untuk mencari matriks normalisasi. Berikut Persamaan (2.10) dan mencari matriks normalisasi:

$$\phi_i = \Gamma_i - \psi \quad (2.10)$$

Dimana Γ_i merupakan nilai setiap piksel dari citra data uji kemudian dikurangkan dengan *mean* (ψ) yang merupakan hasil dari proses ekstraksi ciri dari citra data latih.

2. Menghitung nilai *project image* dari citra data uji.

   Pada tahapan ini yaitu menghitung nilai *project image* dari citra data uji dengan mengalikan nilai *eigen face* yang didapat dari hasil ekstraksi ciri pada data latih dengan matrik normalisasi data uji. Berikut Persamaan (2.11) untuk mencari nilai *project image* pada ekstraksi ciri data uji:

$$Project_{images} = \phi \times Eig_f{}^T \qquad (2.\ 11)$$

Setelah proses perhitungan selesai dan nilai *project image* telah didapatkan, maka langkah selanjutnya mereduksi matriks *project image* tersebut dengan nilai *n* yang dapat disesuaikan dengan kebutuhan. Kemudian setelah matriks *project image* direduksi, masukkan nilai *project image* yang berupa matriks 2 dimensi tesebut ke dalam proses LVQ3 untuk dilakukan identifikasi.

2.7. Normalisasi

Sebelum melakukan proses ke tahap LVQ3 ada baiknya data yang telah melalui tahap ekstraksi ciri harus melawati tahap normalisasi. Tujuan normalisasi ini agar nilai masukan dan target berada dalam range 0.1 sampai dengan 0.9. Normalisasi data dapat dihitung dengan Persamaan (2.12) sebagai berikut:

$$x' = (0.8\ (\ x - a\ ))/\ (\ b - a))\ + 0.1 \qquad (2.\ 12)$$

Keterangan :

x' = hasil normalisasi data latih.

x = nilai data latih yang akan dinormalisasi.

a = nilai minimal dari semua data latih yang akan dinormalisasi.

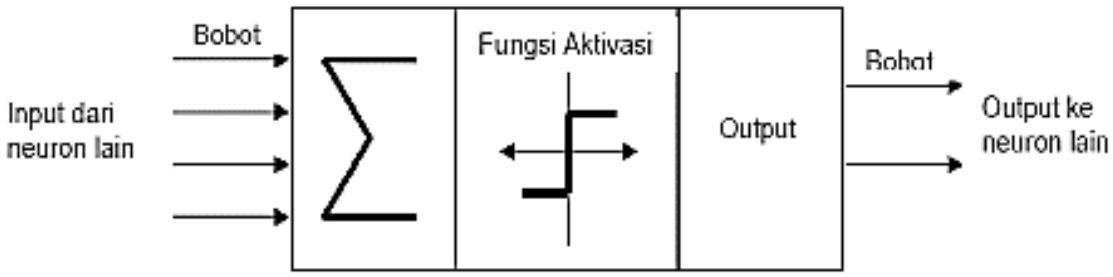
b = nilai maksimal dari semua data latih yang akan dinormalisasi.

2.8. Jaringan Syaraf Tiruan

Jaringan Syaraf Tiruan (JST) merupakan suatu paradigma pengolahan informasi yang kinerjanya mirip dengan sistem saraf yang secara biologis, seperti halnya proses informasi pada otak manusia. Kunci dari paradigma ini yaitu struktur dari sistem pengolahan informasi yang terdiri dari elemen pemrosesan yang saling berhubungan (*neuron*), bekerja secara bersamaan untuk menyelesaikan masalah tertentu. Cara kerja dari JST seperti cara kerja manusia, yaitu belajar melalui contoh-contoh (Salim dan Jauhari 2016). JST tercipta sebagai suatu generalisasi model matematis dari pemahaman manusia (*human cognition*) yang didasarkan pada asumsi seperti berikut ini (Dessy dan Irawan 2012) :

1. Pemrosesan terjadi di bagian *neuron*.
2. Sinyal mengalir diantara sel saraf/*neuron* melalui penghubung.
3. Setiap sambungan penghubung memiliki Vektor Pewakil yang berguna untuk menggandakan/ mengalikan sinyal yang dikirim melaluinya.
4. Setiap sel syaraf akan menerapkan fungsi aktivasi kepada sinyal yang telah melakukan penjumlahan berVektor Pewakil yang masuk kepadanya, dimana fungsinya untuk menentukan sinyal keluaran.

Model struktur neuron pada JST dapat digambarkan seperti Gambar 2.1 dibawah ini.



Gambar 2. 1 Model struktur JST

2.8.1. Arsitektur Jaringan

Arsitektur jaringan merupakan pengaturan neuron dalam layer dan hubungan-hubungannya. Dalam memecahkan suatu permasalahan, dibutuhkan sebuah arsitektur jaringan yang tepat, dan oleh karena itu arsitektur akan menentukan suatu keberhasilan sebuah pola target yang ingin dicapai. Terdapat tiga jenis dari arsitektur JST sebagai berikut (Sari, Agustina, dan Darusalam 2015):

1. Jaringan Layar Tunggal (*Single Layer Network*)

   Pada jaringan ini, kumpulan *input neuron* dihubungkan langsung dengan kumpulan *output*nya. Dalam beberapa model misalnya perceptron, hanya ada sebuah unit *neuron output*.

2. Jaringan Layar Jamak (*Multi Layer Network*)

   Jaringan layar jamak merupakan perluasan dari layar tunggal. Dalam jaringan ini, selain unit masukan dan keluaran, terdapat unit-unit lain yang sering disebut layer tersembunyi (*hidden layer*).

3. Jaringan *Reccurent*

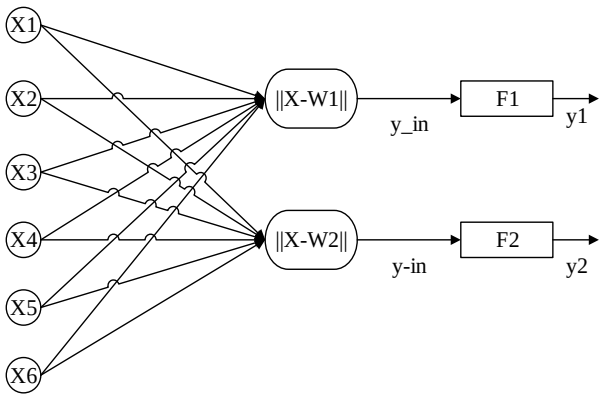
   Model jaringan *recurrent* ini mirip dengan jaringan layer tunggal maupun ganda. Hanya saja yang membuat sedikit berbeda yaitu, terdapat *neuron* keluaran yang memberikan sinyal pada unit masukan yang sering disebut *feedback loop*.

2.8.2. *Learning Vector Quantization 3* (LVQ3)

LVQ3 merupakan pengembangan algoritma dari metode LVQ atau bisa disebut LVQ1. LVQ merupakan suatu metode pengelompokan pola yang keluarannya mewakili kelompok tertentu. Proses pembelajaran pada setiap *neuron* yaitu untuk mencari jarak terdekat antara suatu vektor masukan ke Vektor Pewakil yang dipilih. Selama proses pembelajaran unit keluaran dikondisikan dengan memperbarui Vektor Pewakil melalui pembelajaran terarah (*supervised*) untuk memperkirakan hasil klasifikasi (Budianita dan Prijodiprodjo 2012). LVQ

merupakan tipe arsitektur jaringan menggunakan lapisan tunggal umpan maju (*Single-Layer Feed Forward*) yang terdiri dari unit input dan output. Suatu lapisan kompetitif akan mengelompokkan vektor-vektor masukan secara otomatis (Dessy dan Irawan 2012).

Berikut Gambar 2.2 dari arsitektur LVQ:



Gambar 2. 2 Arsitektur LVQ

Dimana:

1. x_1 - x_6 = nilai *input*
2. || x-w_1 || - || x-w_2 || = jarak Vektor Pewakil
3. F_1 – F_2 = lapisan *output*
4. y_1 – y_2 = nilai *output*

Algoritma LVQ3 memiliki perbedaan terhadap versi LVQ sebelumnya seperti LVQ1. Hal yang membedakannya yaitu terdapat pada nilai *beta* (β). Nilai *beta* (β) yaitu suatu nilai yang digunakan sebagai daerah yang harus dipenuhi untuk memperbaharui vektor referensi pemenang (Dc1) dan *runner-up* (Dc2), jika kedua pemenang berada dikelas yang sama. Ide pengembangan pada algoritma LVQ ini adalah jika nilai masukan memiliki taksiran jarak yang sama dengan vektor pemenang dan *runner-up*, maka masing-masing vektor tersebut harus melakukan pembelajaran (Budianita dan Prijodiprodjo 2012). Fungsi aktivasi yang digunakan pada LVQ3 yaitu fungsi linier. Tujuannya adalah agar diperoleh keluaran yang sama dengan masukan, sesuai dengan rumus fungsi linier yaitu $y = x$ (Hidayati dan Warsito 2010). Berikut adalah langkah-langkah dari algoritma pelatihan LVQ3 (Kohonen 1990):

1. Tentukan vektor pewakil (Wj) dari *variable input* ke-j menuju ke kelas ke-i, vektor pelatihan (Xi), nilai *epoch*, target (T), parameter *learning rate* (α), pengurangan *learning rate*, nilai minimum *learning rate* (min α), nilai *window* (ω), dan nilai *second learning rate* (ε).
2. Masuk ketahap iterasi.
3. Kerjakan jika α > min α atau *epoch* kecil dari nilai *epoch*.
   a. Hitung jarak *euclidean* antara Vektor Pewakil (Wj) dengan vektor (Xi) dengan menggunakan Persamaan (2.13) :

$$d = \sqrt{(X - W)^2} \qquad (2.\ 13)$$

   b. Tentukan jarak terkecil / terdekat pertama (dc1) dan jarak terdekat kedua (dc2).
   c. Cek kondisi dc1 dan dc2 menggunakan persamaan *window* (Persamaan 2.14) dimana:

$$Min\left(\frac{dc1}{dc2}, \frac{dc2}{dc1}\right) > (1 - \omega)/(1 + \omega) \qquad (2.\ 14)$$

   - Jika persamaan *window* terpenuhi, maka cek kondisi kembali dimana jika T ≠ dc1 dan T = dc2 dan kondisi tersebut terpenuhi, maka lakukan perubahan vektor pewakil dengan menggunakan Persamaan (2.16) dan Persamaan (2.17) berikut:

$$Wdc1(c1) = Wdc1(c1) - \alpha\,(Xi - Wdc1(c1)) \qquad (2.\ 15)$$

$$Wdc2(c2) = Wdc2(c2) + \alpha\,(Xi - Wdc2(c2)) \qquad (2.\ 16)$$

   - Jika kondisi T ≠ dc1 dan T = dc2 tidak terpenuhi, maka lakukan pengecekan kondisi kembali dimana jika T = dc1 dan T = dc2 dan kondisi tersebut terpenuhi, maka lakukan perubahan vektor pewakil. Sebelum melakukan perubahan vektor pewakil, lakukan pencarian nilai *beta* (β) dengan menggunakan Persamaan (2.17), setelah di dapatkan nilai beta, lakukan perubahan vektor pewakil dengan menggunakan Persamaan (2.18) dan Persamaan (2.19) berikut:

$$\beta = \varepsilon \times \alpha \quad (2.17)$$

$$Wdc1(c1) = Wdc1(c1) + \beta\,(Xi - Wdc1(c1)) \quad (2.18)$$

$$Wdc2(c2) = Wdc2(c2) + \beta\,(Xi - Wdc2(c2)) \quad (2.19)$$

4. Kurangi nilai α dan tambah nilai *epoch*, dengan menggunakan Persamaan (2.20) dan Persamaan (2.21) berikut:

$$\alpha = \alpha - (0.1 \times \alpha) \quad (2.20)$$

$$Epoch = epoch + 1 \quad (2.21)$$

Setelah dilakukannya pelatihan, maka akan diperoleh vektor pewakil akhir (W) yang nantinya akan digunakan untuk melakukan pengujian. Adapun langkah-langkah pengujian pada LVQ3 sebagai berikut:

1. Inisialisasi vektor pewakil akhir (W) dari hasil pelatihan dan data uji (X)
2. Hitung jarak *euclidean* antara W dan X.
3. Tentukan jarak terkecil (J).
4. J adalah kelas untuk X.

2.9. Huruf *Hiragana* Jepang

Tulisan Jepang awalnya berasal dari tulisan Cina karena sebelumnya orang Jepang tidak memiliki sistem penulisan sendiri. Tulisan Jepang terbagi menjadi tiga yaitu *Kanji* (漢字), *Hiragana* (ひらがな), dan *Katakana* (カタカナ). Huruf *Hiragana* umumnya digunakan untuk menulis kata-kata asli bahasa Jepang seperti menulis akhiran kata, kata keterangan, dalam situasi formal, bacaan anak-anak seperti komik Jepang, juga dalam membaca huruf *kanji* (Darjat 2015). Huruf *Hiragana* memiliki ciri khas tersendiri dari huruf Jepang lainnya. Huruf *Hiragana* memiliki bentuk sangat halus. Huruf *Hiragana* memiliki tiga golongan bunyi yaitu bunyi *Seion*, bunyi *Dakuon*, dan bunyi *Yoon*. Huruf golongan bunyi *Seion* merupakan huruf *Hiragana* dasar yang masing-masing berjumlah 46 huruf. Bunyi tersebut adalah bunyi *Dakuon* dan bunyi *Yoon* (Gede et al. 2015). Berikut adalah Tabel 2.2 Huruf dasar *Hiragana*.

Tabel 2. 2 *Hiragana Seion*

	A	I	U	E	O
	あ	い	う	え	お
K	か	き	く	け	こ
S	さ	し**(shi)**	す	せ	そ
T	た	ち**(chi)**	つ**(tsu)**	て	と
N	な	に	ぬ	ね	の
H	は	ひ	ふ**(fu)**	へ	ほ
M	ま	み	む	め	も
Y	や		ゆ		よ
R	ら	り	る	れ	ろ
W	わ				を
N	ん**(ng)**				

2.9.1. Huruf *Hiragana Dakuon*

Bunyi *Hiragana Dakuon* adalah bunyi huruf *Hiragana* dasar dengan menambahkan tanda *tenten* (``) yaitu tanda titik dua yang diletakkan di sebelah kanan atas huruf *Hiragana* dasar dan tanda *maru* (o) yaitu tanda lingkaran kecil yang diletakkan disebelah kanan atas huruf *Hiragana* dasar. Huruf-huruf dasar yang menggunakan tanda *tenten* adalah huruf ka menjadi ga, sa menjadi za, ta menjadi da dan ha menjadi ba. Sedangkan huruf dasar yang menggunakan [o] tanda *maru* adalah huruf ha menjadi pa (Gede et al. 2015). Berikut Tabel 2.3 *Hiragana Dakuon*:

Tabel 2. 3 *Hiragana Dakuon*

	A	I	U	E	O
G	が	ぎ	ぐ	げ	ご
Z	ざ	じ**(ji)**	ず	ぜ	ぞ
D	だ	ぢ**(ji)**	づ**(zu)**	で	ど
B	ば	び	ぶ	べ	ぼ
P	ぱ	ぴ	ぷ	ぺ	ぽ

2.9.2. Huruf *Hiragana Yoon*

Bunyi *Yoon* merupakan bunyi huruf *Hiragana* baik bunyi *Seiyon* dan bunyi *Dakuon* dengan menambahkan huruf ya, yu dan yo yang ditulis disebelah kanan huruf dasar dengan ukuran yang lebih kecil. Penulisan antara huruf ya, yu dan yo yang ditulis dengan ukuran yang sama dan berbeda dengan huruf dasar memiliki perbedaan, misalnya huruf ひや dibaca hiya sementara huruf ひゃ dibaca hya. Huruf びや dibaca biya dan びゃ dibaca bya. Huruf yang menggunakan huruf ya, yu dan yo yaitu huruf dasar urutan kedua, seperti huruf ki, shi, chi, ni, hi, mi dan ri (Gede et al. 2015). Berikut Tabel 2.4 *Hiragana Yoon (Seion)*:

Tabel 2. 4 *Hiragana Yoon (Seion)*

	YA	**YU**	**YO**
K	きゃ	きゅ	きょ
S	しゃ**(sha)**	しゅ**(shu)**	しょ**(sho)**
C	ちゃ	ちゅ	ちょ
N	にゃ	にゅ	にょ
H	ひゃ	ひゅ	ひゅ
M	みゃ	みゅ	みゅ
R	りゃ	りゅ	りょ

Berikut Tabel 2.5 *Hiragana Yoon (Dakuon)*:

Tabel 2. 5 *Hiragana Yoon (Dakuon)*

	YA	**YU**	**YO**
G	ぎゃ	ぎゅ	ぎょ
J	じゃ**(ja)**	じゅ**(ju)**	じょ**(jo)**
J	ぢゃ**(ja)**	ぢゅ**(ju)**	ぢょ**(jo)**
B	びゃ	びゅ	びょ
P	ぴゃ	ぴゅ	ぴょ

2.10. *Confusion Matrix*

Confusion matrix merupakan alat untuk menganalisis seberapa baik pengklasifikasi tersebut dapat mengenali record dalam kelas-kelas yang berbeda. Evaluasi dengan *confusion matrix* menghasilkan nilai *accuracy*. *Accuracy* dalam klasifikasi adalah persentase ketepatan *record* data yang diklasifikasikan secara benar setelah dilakukan pengujian pada hasil klasifikasi (Hana 2013). Berikut adalah Tabel 2.6 *confusion matrix*:

Tabel 2. 6 *Confusion Matrix*

Classification		*Predicted Class*	
Actual Class	*Class*	*True*	*False*
	True	*True Positive*	*False Negative*
	False	*False Positive*	*True Negative*

Adapun untuk menghitung akurasi, digunakan Persamaan (2.22) sebagai berikut (Hana 2013):

$$Akurasi = \frac{TP+TN}{TP+FN+FP+TN} \times 100\% \quad (2.\ 22)$$

Dimana:

1. TP = jumlah *true positive* yaitu jumlah data dari kelas *true* yang benar dan diklasifikasikan sebagai kelas *true*.
2. TN = jumlah *true negative* yaitu jumlah data dari kelas *false* yang benar dan diklasifikasikan sebagai kelas *false*.
3. FP = jumlah *false positive* yaitu jumlah data dari kelas *false* yang benar dan diklasifikasikan sebagai kelas *true*.
4. TN = jumlah *false negative* yaitu jumlah data dari kelas *true* yang benar dan diklasifikasikan sebagai kelas *false*.

2.11. Penelitian Terkait

Beberapa penelitian sebelumnya yang berkaitan tentang penerapan metode PCA dan LVQ3 yang dapat dilihat pada Tabel 2.7 berikut ini:

Tabel 2. 7 Penelitian Terkait

Peneliti	Tahun	Judul	Metode	Akurasi
Darma Setiawan Putra, Adhi Dharma Wibawa, Mauridhi Hery Purnomo	2016	*Classification of EMG during Walking using PCA and LVQ for Biometrics Study*	PCA, LVQ	88,8%
Rizal Isnanto, Ajub Zahra, Eko Widianto	2015	Analisis Kinerja Pengenalan Telapak Tangan Menggunakan Ekstraksi Ciri PCA dan *Overlapping Block*	PCA, *Overlapping Block*	90%
Diyah Puspitaningrum, Dyan Kemala sari, Boko Susilo	2014	Dampak Reduksi Sampel Menggunakan PCA Pada Pelatihan Jaringan Saraf Tiruan Terawasi	PCA, BPNN	86,75%
Mohammed Azara, Tamer Fatayer, Alaa El-Halees	2012	*Arabic Text Classification Using* LVQ	LVQ, LVQ2.1,LVQ3, OLVQ1,OLVQ3	84,09%
Yasser Fouad Hassan, Nora Habeb	2012	*Hybrid System of* PCA, *Rough Sets and Neural Networks for Dimensionality Reduction and Classification in Human Face Recognition*	PCA, *Rough Sets*, LVQ	93%
Elvia Budianita, Widodo Prijodiprodjo	2012	Penerapan LVQ untuk Klasifikasi Status Gizi Anak	LVQ,LVQ3	95,2%