BAB 2
LANDASAN TEORI

Terdapat beberapa telaah literatur dengan informasi yang berhubungan dengan Implementasi *Face Recognition* dengan algoritma DCT, GLCM, dan *Backpropagation*.

2.1 Face Recognition

Wajah manusia adalah salah satu bagian tubuh yang mampu menjadi pusat perhatian dalam interaksi sosial, karena wajah mampu menunjukkan identitas dan emosi manusia. Manusia mampu mengenali ribuan wajah manusia karena interaksi yang sering dilakukan ataupun hanya sekilas bertemu dan bertatapan. Ketika manusia sudah mengenal seseorang, manusia mampu mengenali perubahan yang terjadi pada mereka seperti bertambahnya usia, pemakaian kacamata, ataupun perubahan gaya rambut. Berdasarkan pernyataan demikian, wajah dapat dijadikan sebagai indikasi pengenalan wajah ataupun *face recognition* [13] untuk sistem keamanan sebagai teknologi biometrik. Pengenalan wajah sendiri telah melibatkan beberapa variabel, misalnya citra sumber, citra hasil pengolahan citra, citra hasil ekstrasi, dan data profil seseorang [14]. Perkenalan wajah memiliki beberapa tahapan proses, antara lain:

1. Modul akuisisi, berupa input untuk proses *face recognition* yang dapat diperoleh dari gambar kamera ataupun citra digital.

2. Modul *pre-processing*, diperlukan proses penyesuaian dari input citra. Tahapan-tahapan yang dilakukan antara lain normalisasi ukuran citra, penggunaan median *filtering* untuk menghilangkan *noise* akibat pergeseran frame atau kamera. Penggunaan histogram *equalization* untuk mempermudah proses pengenalan citra. Tujuan dari proses ini adalah meningkatkan kualitas citra tanpa menghilangkan informasi utamanya. Selain itu, digunakan juga *high pass filtering* untuk mendapatkan sisi tepi citra, penghilangan latar belakang sehingga hanya wajah yang diproses, dan terakhir dilakukan proses *grayscaling* untuk mengubah citra RGB menjadi citra skala abu-abu. Tujuan dari pre-processing ini adalah mengatasi masalah yang mungkin muncul saat proses pengenalan wajah.

3. Modul Ekstrasi Fitur, bertujuan untuk memperoleh bagian-bagian terpenting sebagai vektor yang mempresentasikan wajah dan bersifat unik.

4. Modul Klasifikasi, pemisahan pola, fitur wajah dibandingkan dan disamakan dengan fitur yang tersimpan dalam database untuk mengetahui citra wajah tersebut apakah dapat dikenali atau tidak.

5. Training Set, modul yang digunakan untuk proses training, proses identifikasi. Semakin kompleks dan sering proses pengenalan wajah maka semakin baik.

6. Database, berisikan kumpulan citra wajah.

2.2 Citra

Citra adalah gambaran, kemiripan, atau imitasi dari suatu objek. Secara keseluruhan citra adalah sistem perekaman dan dapat memiliki sifat optik seperti foto. Bersifat analog dari sinyal *video* seperti gambar pada monitor televisi, dan memiliki sifat digital yang dapat langsung disimpan pada media penyimpanan. Citra dapat dibedakan berdasarkan dua jenis, yaitu citra diam dan bergerak. Citra diam adalah citra tunggal yang tidak bergerak, sering dikenal dengan sebutan citra. Sementara untuk citra bergerak, merupakan rangkaian atau kumpulan dari citra diam yang ditampilkan secara berurutan dan beruntun (sekuensial) sehingga hal ini memberikan asumsi kepada pikiran bahwa citra atau gambar yang dimaksud bergerak [15].

2.3 Ekstrasi Fitur

Proses dalam mengekstrasi fitur dari objek yang dapat menggambarkan karakteristik dari objek yang diekstraksi tersebut. Fitur merupakan karakteristik yang unik dari suatu objek, sehingga fitur memiliki beberapa persyaratan yang harus dipenuhi [6] yaitu:

1. Mampu membedakan suatu objek dengan objek lainnya (*discrimination*).

2. Memperhatikan kompleksitas komputasi.

3. Independen (tidak terikat) artinya bersifat invarian terhadap berbagai perubahan seperti rotasi, penskalaan, dan pergeseran.

4. Jumlah fitur untuk meminimalkan waktu komputasi dan ruang penyimpanan saat masuk ke proses selanjutnya.

2.4 Discrete Cosine Transform (DCT)

DCT (Discrete Cosine Transform) adalah metode yang pertama kali diperkenalkan oleh Natarajan, Ahmed, dan Rao pada tahun 1974 melalui makalah yang dibuat dengan judul "On image processing and a Discrete Cosine Transform". DCT merupakan metode teknik yang dapat mengubah sinyal menjadi komponen frekuensi dasar. DCT dapat mempresentasikan citra dari penjumlahan *sinusoida* dari *magnitude* dan frekuensi yang berubah-ubah. DCT memiliki sifat yang dapat mengubah informasi citra yang signifikan dikonsentrasikan hanya pada beberapa koefisien DCT [15].

DCT memiliki kelebihan dalam ekstrasi fitur data, yaitu:

1. DCT dapat menghitung kuantitas bit-bit data citra dimana pesan tersebut tersembunyi di dalamnya. Gambar yang dikompresi dengan *lossy compression* mengakibatkan perubahan yang jelas pada gambar citra, pada metode DCT tidak terjadi hal demikian karena metode ini terjadi hanya pada frekuensi di dalam gambar citra, bukan pada domain spasial sehingga tidak mengakibatkan perubahan yang terlihat pada gambar citra.
2. Kokoh terhadap manipulasi pada *stego-object* (informasi atau data tersembunyi).

DCT juga memiliki kekurangan dalam ekstrasi fitur data, yaitu:

1. Tidak tahan dengan perubahan objek, karena hal ini dapat mengubah atau hilangnya informasi pesan.
2. Implementasi algoritma yang panjang dan diperlukannya perhitungan yang besar.

2.4.1 Discrete Cosine Transform Dua Dimensi (2D-DCT)

Pada penelitian ini dilakukan implementasi DCT 2 dimensi. Pada dua dimensi diperlukan versi dua dimensi dari DCT. matrik NxN pada 2D (2 Dimensi) DCT dapat dilakukan perhitungan cara yang sama seperti 1D (1 Dimensi) DCT. Diterapkan pada setiap baris dari C dilanjutkan perhitungan hasilnya pada setiap

kolomnya. Persamaan DCT 2 dimensi dapat ditampilkan dalam matriks NxN. Sehingga menghasilkan *output* berupa matriks NxN. Rumus pada transformasi 2D DCT untuk C , yaitu [16]:

$$C(u,v) = \frac{2}{N}\,\alpha(u)\,\alpha(v)\sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y)\cos\left(\frac{\pi(2x+1)u}{2N}\right)\cos\left(\frac{\pi(2y+1)u}{2N}\right) \tag{2.1}$$

Dengan,

C(u,v)= nilai DCT indeks ke-(u,v)

f(x,y)= nilai *pixel* pada indeks ke-(x,y)

N= ukuran baris dan kolom matriks

$\alpha(u)$, $\alpha(v)$ = 1 jika (u,v) > 0 (koefisien yang didapat dari C(u,v))

$\alpha(u)$, $\alpha(v)$ = $\frac{1}{\sqrt{2}}$ jika (u,v) = 0 (koefisien yang didapat dari C(u,v))

Namun, jika matriks berukuran MxN, maka persamaan yang digunakan dapat dituliskan sebagai berikut [16]:

$$C(u,v) = \sqrt{\frac{2}{M}}\sqrt{\frac{2}{N}}\,\alpha(u)\,\alpha(v)\sum_{x=0}^{N-1} \sum_{y=0}^{M-1} f(x,y)\cos\left(\frac{\pi(2x+1)u}{2M}\right)\cos\left(\frac{\pi(2y+1)u}{2N}\right) \tag{2.2}$$

Dengan,

C(u,v)= nilai DCT indeks ke-(u,v)

f(x,y)= nilai *pixel* pada indeks ke-(x,y)

N= ukuran baris matriks

M= ukuran kolom matriks

$\alpha(u)$, $\alpha(v)$ = 1 jika (u,v) > 0 (koefisien yang didapat dari C(u,v))

$\alpha(u)$, $\alpha(v)$ = $\frac{1}{\sqrt{2}}$ jika (u,v) = 0 (koefisien yang didapat dari C(u,v))

Citra yang telah ditransformasikan dengan DCT dibagi menjadi tiga komponen frekuensi, yaitu *low-frequency, mid-frequency, dan high-frequency*. Pada citra wajah manusia *low-frequency* terdapat informasi berupa tingkat variasi pencahayaan dan permukaan halus pada citra seperti dahi, pipi dan sejenisnya. Pada *mid-frequency* memiliki informasi berupa struktur dasar dari wajah pada citra. Sedangkan *high-frequency* memiliki informasi berupa *noise* dan tepian (*edge*) [17].

Citra yang sudah diproses pada domain frekuensi dapat dikembalikan ke domain spasial menggunakan rumus *inverse* DCT, persamaan *inverse* DCT pada matriks NxN sebagai berikut [16]:

$$f(x,y) = \frac{2}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} \alpha(u)\, \alpha(v)\, C(u,v) \cos\left(\frac{\pi(2x+1)u}{2N}\right) \cos\left(\frac{\pi(2y+1)u}{2N}\right) \tag{2.3}$$

Dengan,
C(u,v)= nilai DCT indeks ke-(u,v)
f(x,y)= nilai *pixel* pada indeks ke-(x,y)
N= ukuran baris dan kolom matriks
$\alpha(u)$, $\alpha(v)$ = 1 jika (u,v) > 0 (koefisien yang didapat dari C(u,v))
$\alpha(u)$, $\alpha(v)$ = $\frac{1}{\sqrt{2}}$ jika (u,v) $= 0$ (koefisien yang didapat dari C(u,v))

Jika pada ukuran matriks MxN, maka persamaan *inverse* DCT yang digunakaan adalah sebagai berikut [16]:

$$f(x,y) = \sqrt{\frac{2}{M}} \sqrt{\frac{2}{N}} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} \alpha(u)\, \alpha(v)\, C(u,v) \cos\left(\frac{\pi(2x+1)u}{2M}\right) \cos\left(\frac{\pi(2y+1)u}{2N}\right) \tag{2.4}$$

Dengan,
C(u,v)= nilai DCT indeks ke-(u,v)
f(x,y)= nilai *pixel* pada indeks ke-(x,y)
N= ukuran baris matriks
M= ukuran kolom matriks
$\alpha(u)$, $\alpha(v)$ = 1 jika (u,v) > 0 (koefisien yang didapat dari C(u,v))
$\alpha(u)$, $\alpha(v)$ = $\frac{1}{\sqrt{2}}$ jika (u,v) $= 0$ (koefisien yang didapat dari C(u,v))

2.5 Gray Level Co-occurrence Matrix (GLCM)

GLCM pertama kali diperkenalkan oleh Haralick el al pada tahun 1973. Metode GLCM adalah teknik untuk mengekstraksi ciri pada *contrast*, *correlation*, *energy*, *homogeneity*, dan *dissimilary* terhadap citra. GLCM merupakan metode yang memiliki tujuan untuk memperoleh ciri statistik *order* dua dengan

dilakukannya perhitungan probabilitas hubungan ketetanggaan antara dua piksel pada jarak orientasi sudut tertentu [18].

Berikut terdapat langkah-langkah yang dilakukan dalam melakukan perhitungan GLCM adalah sebagai berikut [6, 19]:

1. Dibentuk matriks awal GLCM dari pasangan dua piksel berjajar sesuai dengan arah 0°, 45°, 90°, dan 135°.
2. Membuat bentuk matriks yang simetris dengan menjumlahkan keseluruhan matriks awal GLCM dengan nilai transprosnya.
3. Melakukan normalisasi matriks GLCM dengna membagikan setiap elemen matriks dengan jumlah pasangan piksel.
4. Melakukan ekstrasi ciri, yaitu:
   (a) *Energy*, mengukur keseragaman atau sering dikenal sebagai *Angular Second Moment* (ASM). *Energy* adalah penjumlahan perpangkatan dari elemen matriks. GLCM nilai yang tinggi ketika citra mempunyai homogenitas yang baik atau mempunyai nilai piksel yang hampir serupa [20]. Rumus persamaan untuk menghitung *energy*, yaitu:

$$Energy = \sum_{i}^{N_g} \sum_{j}^{N_g} GLCM(i,j)^2 \tag{2.5}$$

   (b) *Homogeneity*, adalah perulangan ukuran struktur pada bobot nilai yang merupakan nilai *invers* dari *contrast* [21]. Berikut persamaan untuk menghitung *homgeneity*:

$$Homogeneity = \sum_{i}^{N_g} \sum_{j}^{N_g} \frac{GLCM(i,j)}{1+|i-j|} \tag{2.6}$$

   (c) *Contrast*, sering juga disebut sebagai *inertia*. *Contrast* merupakan ukuran intensitas tingkat keabuan antara piksel dengan piksel lainnya dengan lokasi relatif. *Contrast* memiliki batasan nilai dengan rentang dari 0 hingga pangkat 2 dari panjang matriks GLCM simetris. Citra yang memiliki eleman piksel bernilai sama secara keseluruhan, maka *contrast* bernilai 0 [22]. Rumus persamaan untuk menghitung nilai

contrast, yaitu:

$$Contrast = \sum_{i}^{N_g} \sum_{j}^{N_g} (i-j)^2 \, GLCM(i,j) \tag{2.7}$$

(d) *Correlation*, ukuran ketergantungan linear derajat keabuan pada citra sehingga memperoleh petunjuk jika terdapat struktur linear dalam citra [20]. Persamaan *correlation* dapat dijabarkan sebagai berikut.

$$Correlation = \sum_{i}^{N_g} \sum_{j}^{N_g} \frac{(i-\mu_i)(j-\mu_j) \, GLCM(i,j)}{\sigma_i \, \sigma_j} \tag{2.8}$$

(e) *Dissimiarity*, merupakan pengukuran ketidak miripan pada tekstur. *Dissimiarity* bernilai besar jika berbentuk tekstur acak dan bernilai kecil jika berbentuk tekstur sama atau seragam [20]. Berikut rumus persamaan *dissimiarity*, yaitu:

$$Dissimiarity = \sum_{i}^{N_g} \sum_{j}^{N_g} |i-j| \, GLCM(i,j) \tag{2.9}$$

(f) *Entropy*, merupakan sejumlah informasi dari citra yang dibutuhkan untuk kompresi pada citra, yaitu:

$$Entropy = -\sum_{i}^{N_g} \sum_{j}^{N_g} GLCM(i,j) \, log(GLCM(i,j)) \tag{2.10}$$

(g) *Autocorrelation*, merupakan pengukuran *correlation* yang ada di antara garis diagonal utama, yaitu:

$$Autocorrelation = \sum_{i}^{N_g} \sum_{j}^{N_g} i \, . \, j \, . \, GLCM(i,j) \tag{2.11}$$

Dengan keterangn,

i= Baris.

j= Kolom.

$GLCM$= Probabilitas (0-1) yaitu elemen matriks.

μ= Rata-rata dari matriks kookurensi yang telah dinormalisasi.

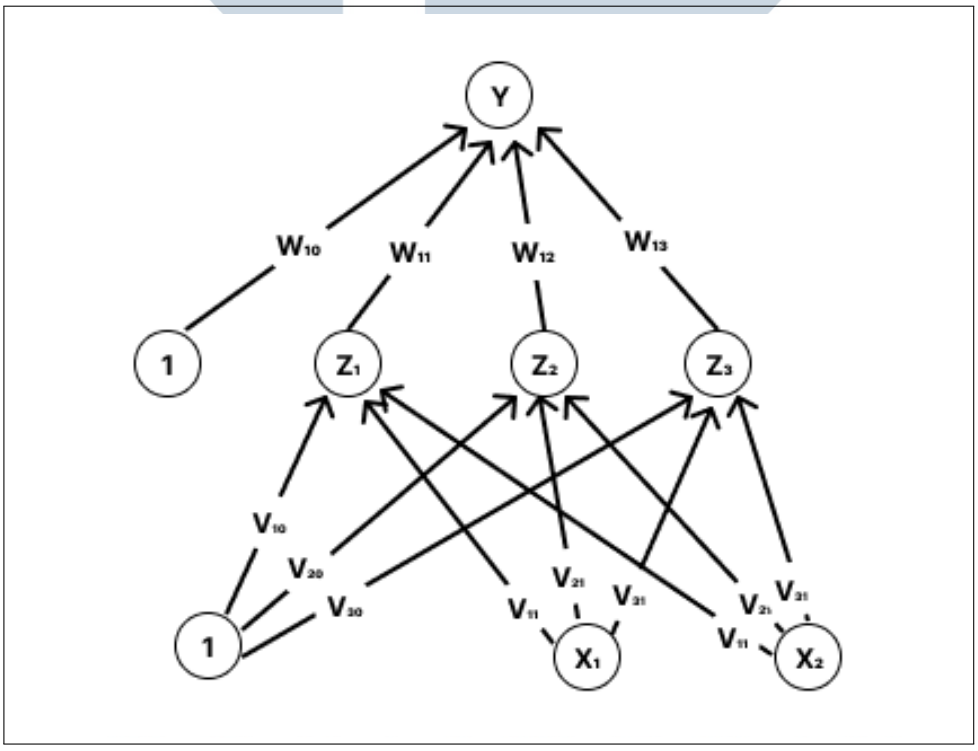
N_g= Ukuran matriks hasil kuantisasi.

σ_i= Nilai standar deviasi baris.

σ_j= Nilai standar deviasi kolom.

2.6 Backpropagation

Backpropagation adalah salah satu model Jaringan Saraf Tiruan (JST) yang mampu mencapai keseimbangan antara kemampuan jaringan untuk mengenali pola yang telah dilatih dan memberikan respons yang tepat terhadap pola masukan yang serupa, meskipun tidak identik dengan pola pelatihan. *Backpropagation* terdiri dari tiga lapisan, yaitu lapisan *input*, lapisan tersembunyi, dan lapisan *output*. Lapisan *input* berfungsi sebagai pengirim sinyal *input* ke lapisan tersembunyi, tanpa melibatkan perhitungan atau proses komputasi. Lapisan tersembunyi dan lapisan *output* juga terdapat dalam model ini.



Gambar 2.1. Arsitektur diagram backpropagation
sumber: [23]

Epoch merupakan satu siklus proses pelatihan pada jaringan saraf tiruan. Proses pelatihan pada *backpropagation* meliputi [9]:

1. Propagasi Maju, Selama tahap ini, sinyal *input* (X_i) dipropagasikan ke seluruh unit pada *layer* tersembunyi menggunakan fungsi aktivasi yang telah ditentukan. *Output* dari seluruh unit *layer* tersembunyi (Z_j) tersebut kemudian dipropagasikan kembali ke *layer* tersembunyi yang terdapat dibawahnya (jika ada) menggunakan fungsi aktivasi yang ditentukan. Demikian seterusnya hingga akhirnya menerima hasil keluaran jaringan (Y_k) dengan k=1, 2, 3, ...

   Setelah dihasilkan keluaran jaringan, dilanjutkan dengan membandingkan keluaran jaringan dengan target (t_k) yang ingin dicapai. Selisih (t_k - Y_k) adalah kesalahan yang terjadi. Namun jika kesalahan lebih kecil daripada batas toleransi yang diperkenankan maka iterasi dihentikan, tapi jika selisihnya lebih besar dari *error* dari yang diperkenankan maka bobot dimodifikasi untuk mengurangi kesalahan yang terjadi.

2. Propagasi Mundur, Berdasakan kesalahan t_k - T_k, dihitung faktor σ_k (k = 1, 2, ..., m) yang digunakan untuk mendistribusikan kesalahan di unit keluaran (Y_k) ke semua unit tersembunyi yang terhubung langsung dengan keluaran (Y_k). σ_k juga biasa digunakan untuk mengubah nilai bobot garis yang berhubungan langsung dengan unit keluaran.

   Dengan cara yang sama, faktor dari σ yang terdapat pada *layer* tersembunyi juga dicari untuk mengubah nilai bobot pada *layer* tersembunyi yang ada dibawahnya (jika ada). Demikian seterusnya hingga mendapatkan faktor σ pada *layer* tersembunyi yang berhubungan langsung dengan *layer input*.

3. Modifikasi Bobot, setelah seluruh faktor didapatkan, dilanjutan dengan σ memodifikasi bobot seluruh jaringan secara bersamaan. Memodifikasi suatu garis bergantung pada faktor σ unit *layer* yang ada dibawahnya. Sebagai contoh, perubahan pada bobot garis yang menuju ke *layer* keluaran didasarkan atas σ yang ada di unit keluaran.

Keseluruhan fase ini terus dilakukan hingga kondisi penghentian dipenuhi. Kondisi penghentian secara umumnya sering dipakai adalah jumlah iterasi atau kesalahan. Iterasi diberhentikan jika jumlah iterasi yang dilaksanakan melampaui jumlah maksimum iterasi yang telah ditetapkan, atau kesalahan yang sudah terjadi sudah lebih kecil dari batas toleransi yang diberikan.

Berikut algoritma pelatihan untuk jaringan saraf tiruan *backpropagation*, yaitu [12]:

1. Lakukan tahap propagasi maju, seluruh unit menerima sinyal dan meneruskannya keunit yang tersembunyi diatasnya. Hitung keseluruhan keluaran di *layer* tersembunyi z_j(j = 1, 2, ..., p).

$$z\_net_j = v_{0j} + \sum_{i=1}^{n} x_i \, v_{ij} \tag{2.12}$$

$$z_j = f(z\_net_j) = \frac{1}{1 + e^{-z\_net_j}} \tag{2.13}$$

Dengan keterangan,
x = Layer masukkan
z = Layer tersembunyi
y = Layer keluaran
z_net_j = Faktor keluaran pada layer tersembunyi
y_net_k = Faktor keluaran pada layer keluaran
v_{ij} = Bobot pada layer tersembunyi
v_{0j} = Bias pada layer tersembunyi
w_{0k} = Bias pada layer keluar
w_{jk} = Bobot pada layer keluar
σ_j = Faktor kesalahan layer tersembunyi
σ_k = Faktor kesalahan layer keluaran
α = Laju pembelajaran

Hitung semua keluaran jaringan diunit y_k (k=1, 2, ..., m).

$$y\_net_k = w_{0k} + \sum_{j=1}^{p} z_j \, w_{jk} \tag{2.14}$$

$$y_k = f(y\_net_k) = \frac{1}{1 + e^{-y\_net_k}} \tag{2.15}$$

2. Dilanjutkan ketahap propagasi mundur, menghitung faktor σ unit keluaran berdasarkan kesalahan pada seluruh unit keluaran $y_k (k = 1, 2, ..., m)$.

$$\sigma_k = (t_k - y_k) \, f'(y\_net_k) = (t_k - y_k) \, y_k (1 - y_k) \tag{2.16}$$

σ_k adalah unit kesalahan yang digunakan dalam perubahan bobot *layer* yang

ada dibawahnya. Menghitung suku perubahan bobot w_{kj} (yang nantinya digunakan untuk merubah bobot w_{kj}) dengan laju pembelajaran $a(k = 1,2,...,m; j = 0,1,...,p)$.

$$\Delta w_{jk} = \alpha\ \sigma_k\ z_j \tag{2.17}$$

Menghitung faktor σ unit tersembunyi berdasarkan kesalahan di keseluruhan unit tersembunyi $z_j(j = 1,2,...,p)$.

$$\sigma\_net_j = \sum_{k=1}^{m} \sigma_k\ w_{jk} \tag{2.18}$$

Dengan faktor σ unit tersembunyi adalah sebagai berikut.

$$\sigma_j = \sigma\_net_j\ \ f'(z\_net_j) = \sigma\_net_j\ z_j\ (1 - z_j) \tag{2.19}$$

3. Melakukan tahapan modifikasi bobot. Menghitung keseluruhan perubahan bobot yang menuju keunit keluaran (k = 1, 2, ..., m; j=1, 2, ..., p).

$$w_{jk}(baru) = w_{jk}(lama) + \Delta w_{jk} \tag{2.20}$$

Menghitung keseluruhan perubahan bobot yang menuju ke unit tersembunyi (j=1, 2, ..., p; i = 0, 1, ..., n).

$$v_{ij}(baru) = v_{ij}(lama) + \Delta v_{ij} \tag{2.21}$$

4. Memeriksa apakah kondisi berhenti sudah terpenuhi atau belum. kondisi berhenti terpenuhi jika *epoch* sudah mencapai maksimal *epoch* atau toleransi *error* yang diberikan sudah terpenuhi maka iterasi berhenti dengan sendirinya. Jika belum, maka dilanjutkan ke tahap berikutnya dan seterusnya. Rumus yang digunakan untuk toleransi *error* yaitu MSE.

$$SquareError = \sum (t_k - y_k)^2 \quad dan \quad MSE = \frac{jumlah\ square\ error\ setiap\ data}{jumlah\ data} \tag{2.22}$$

5. Kembali lagi ke langkah yang pertama, hingga kondisi penghentian pada langkah yang kelima sudah terpenuhi.

Setelah pelatihan selesai dilakukan, jaringan dapat dipakai untuk perkenalan

pola. Dalam hal ini, hanya langkah propagesi maju (langkah pertama) yang digunakan untuk keluaran jaringan.

Fungsi sigmoid adalah fungsi asimtotik yang nilainya tidak pernah mencapai 0 ataupun 1, sehingga hasil dari ekstrasi ciri diubah menjadi bilangan antara [0.1, 0.9] dengan menggunakan normalisasi [23]. Rumus normalisasi adalah sebagai berikut.

$$x' = \frac{0.8(x - nilai\ terkecil)}{nilai\ terbesar - nilai\ terkecil} + 0.1 \tag{2.23}$$

2.7 Evaluasi Performa

Confusion Matrix adalah salah satu cara evaluasi peforma dari model machine learning. Sering digunakan dalam mengukur kinerja suatu algoritma klasifikasi. *Confusion Matrix* memiliki informasi tentang klasifikasi aktual dan prediksi dilakukan dengan sistem klasifikasi yang dievaluasi menggunakan data dan matrix. Berikut bentuk Tabel *confusion matrix* dan penjelasannya [9].

Tabel 2.1. *Random Index*

	Positif (Aktual)	**Negatif (Aktual)**
Positif (Prediksi)	TP	FP
Negatif (Prediksi)	FN	TN

1. TP (*True Positif*, data positif yang diprediksi benar sebagai data positif).
2. FP (*False Positif*, data negatif yang diprediksi salah sebagai data positif).
3. FN (*False Negative*, data positif yang diprediksi salah sebagai data negatif).
4. TN (*True Negative*, data negatif yang diprediksi benar sebagai data positif).

Confusion Matrix memiliki beberapa metrik yang memungkinkan untuk diukur seperti *accuracy, recall, precision,* dan *F1-score*. Terdapat penjelasan mengenai metrik tersebut, yaitu [9]:

1. *Accuracy*, metrik yang memberikan persentase pengamatan yang diprediksi sebagai TP ataupun FN dibandingkan dengan semua data. Berikut rumus yang digunakan untuk menghitung nilai metrik *accuracy*:

$$Accuracy = \frac{TP + FN}{TP + TN + FP + FN} \tag{2.24}$$

2. *Recall*, memberikan rasio prefiksi TP dibandingkan dengan jumlah keseluruhan data positif. Berikut rumus yang digunakan untuk menghitung nilai metrik *recall*:

$$Recall = \frac{TP}{TP + FP} \tag{2.25}$$

3. *Precision*, metrik yang mewakili rasio dari TP dibandingkan dengan seluruh hasil yang bernilai positif. Berikut rumus yang digunakan untuk menghitung nilai metrik *precision*:

$$Precision = \frac{TP}{TP + FN} \tag{2.26}$$

4. *F1-Score*, kombinasi dari nilai *precision* dan *recall*, *F1-Score* menghitung nilai mean antara kedua nilai. Berikut rumus yang digunakan untuk menghitung nilai metrik *F1-Score*:

$$F1 - Score = 2X\frac{PrecisionXRecall}{Precision + Recall} \tag{2.27}$$

Perbedaan antara *macro average, micro average,* dan *weighted average*, yaitu:

1. *Macro average* menghitung matriks secara bebas untuk setiap kelas kemudian mengambil rata-ratanya
2. *Micro average* mengitung matriks rata-rata dengan mengumpulkan kontribusi dari semua kelas.
3. *weighted average* menghitung rata-rata dengan memperhitungkan bobot pada setiap datanya.