# BAB 2
# LANDASAN TEORI

## 2.1 Pengenalan Wajah (Face Recognition)

Wajah merupakan bagian tubuh manusia yang menjadi pusat perhatian dalam interaksi sosial, dan berperan penting dalam menunjukkan identitas dan emosi. Kemampuan manusia untuk membedakan seseorang dari wajahnya memang luar biasa. Manusia bahkan dapat mengidentifikasi ribuan wajah dalam rentang waktu yang lama dan mampu mengenali orang itu meskipun telah berubah karena usia, memakai kacamata, atau mengubah gaya rambutnya. Oleh karena itu, wajah digunakan sebagai indikasi untuk merepresentasikan pengenalan wajah. [11].

Pengenalan wajah adalah salah satu teknologi biometrik yang paling banyak digunakan, selain pengenalan sidik jari, retina, dan pengenalan iris, yang juga digunakan dalam sistem keamanan. [12].Pengenalan wajah terdiri dari beberapa proses, antara lain [11]:

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

2. Modul *pre-processing*, berupa proses penyesuaian citra input seperti melakukan normalisasi *size* citra, median *filtering* untuk menghilangkan *noise* akibat pergeseran *frame* atau kamera, *histogram equalization* untuk memudahkan proses pengenalan citra dengan memperbaiki kualitas citra tanpa menghilangkan informasi utamanya, *high pass filtering* untuk mendapatkan sisi tepi suatu citra, *background removal* untuk menghilangkan bagian *background* sehingga bagian wajah saja yang diproses dan *grayscaling* untuk mengkonversi citra RGB menjadi citra skala abu-abu. *Preprocessing* dilakukan untuk menghilangkan masalah yang akan timbul saat proses *face recognition*.

3. *Modul ekstraksi fitur*, untuk mendapatkan bagian terpenting sebagai suatu vektor yang merepresentasikan wajah dan bersifat unik.

4. *Modul klasifikasi*, dengan adanya pemisahan pola, fitur wajah dibandingkan dengan fitur yang tersimpan dalam *database* untuk mengetahui apakah citra wajah tersebut dapat dikenali.

5. *Training set*, modul ini digunakan selama proses pembelajaran (*training*), proses identifikasi/pengenalan, dan semakin kompleks dan sering proses pengenalan wajah akan semakin baik.

6. *Database*, berisi kumpulan citra wajah.

## 2.2 Ekstraksi Fitur

Ekstraksi fitur adalah proses mengekstraksi fitur dari suatu objek yang dapat menggambarkan karakteristik dari objek tersebut. Fitur adalah karakteristik unik dari suatu objek. Fitur yang baik harus memenuhi persyaratan sebagai berikut[13]:

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

2. Memperhatikan kompleksitas komputasi.
   Independen (tidak terikat) artinya bersifat invarian terhadap berbagai perubahan seperti rotasi, penskalaan, pergeseran, dan lain-lain.

3. Jumlah fitur yang sedikit untuk meminimalkan waktu komputasi dan ruang penyimpanan saat masuk ke proses selanjutnya.

## 2.3 Standarisasi Zero-Mean

Data dalam *dataset* terkadang memiliki rentang nilai yang tidak sama. Tentunya hal ini akan memengaruhi hasil pengukuran analisis data, sehingga diperlukan suatu metode yaitu standarisasi Zero-Mean. Metode standarisasi Zero-Mean didasarkan pada mean dan standar deviasi. Standarisasi suatu dataset melibatkan pengubahan skala distribusi nilai, sehingga nilai rata-rata (mean) yang diamati adalah 0 dan standar deviasi adalah 1. Standarisasi Zero-Mean dihitung menggunakan persamaan berikut[14].

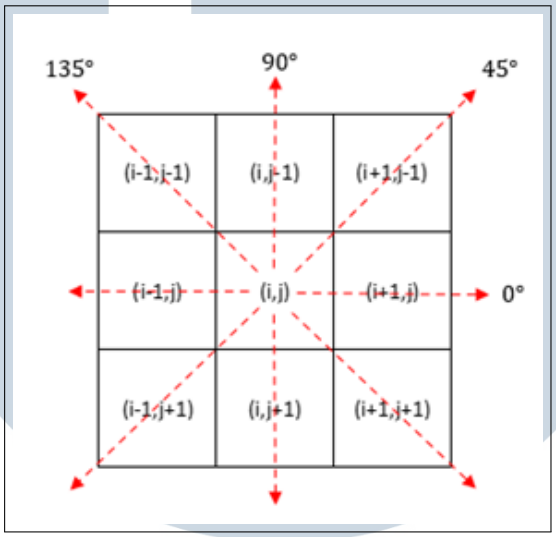
$$Z = \frac{x - \mu}{\sigma} \tag{2.1}$$

Dimana Z adalah nilai Zero-Mean, dan $x$ adalah nilai yang akan distandarisasi, $\mu$ adalah rata-rata seluruh kolom dan baris, serta $\sigma$ adalah standar deviasi.

## 2.4 Gray Level Co-occurrence Matrix (GLCM)

GLCM diusulkan oleh Haralick pada tahun 1979. GLCM adalah fitur statistik orde kedua yang mewakili hubungan kedekatan atau ketetanggaan antara dua

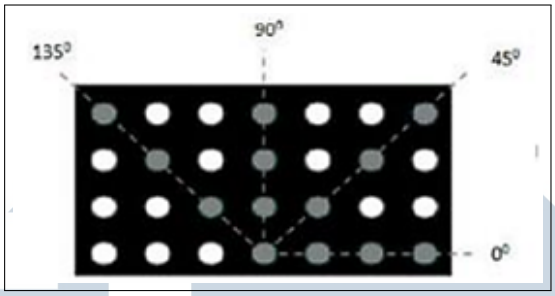
unit piksel dalam gambar berskala abu-abu. [13]. Koordinat pasangan piksel memiliki jarak d dan arah sudut $\theta$. Jarak dinyatakan dalam piksel dan sudut dinyatakan dalam derajat. Orientasi sudut dibentuk berdasarkan 4 arah sudut yaitu, 0°, 45°, 90° dan 135°, dengan jarak antar piksel sebesar 1 piksel [15]. Suatu piksel yang bertetanggaan memiliki d diantara keduanya, dapat terletak di delapan arah yang berlainan, hal ini ditunjukkan Gambar 2.1.




Gambar 2.1. Hubungan ketetanggaan antar piksel dan jarak spasial

Sumber: [16]

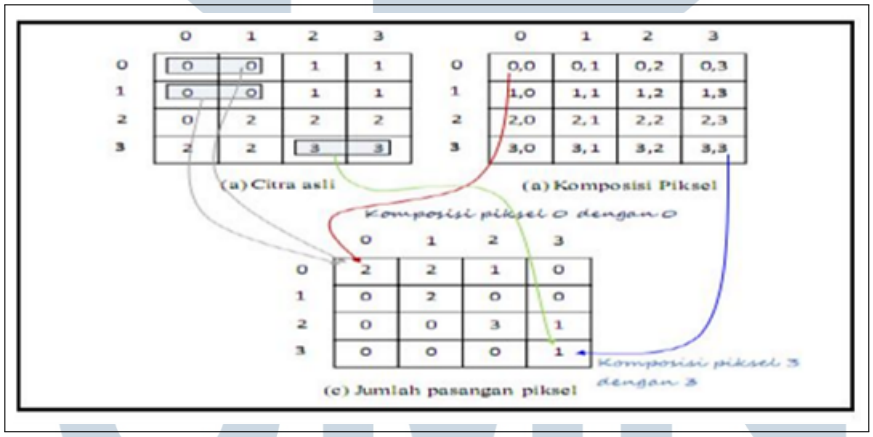
Langkah pertama dalam menghitung fitur-fitur tekstur dari GLCM adalah mengubah citra RGB menjadi citra *grayscale*. Langkah kedua adalah menciptakan *co-occurrence matrix* kemudian menentukan hubungan spasial antara piksel referensi dengan piksel tetangga berdasarkan arah sudut $\theta$ dan jarak d. Langkah selanjutnya adalah menambahkan *co-occurrence matrix* dengan *transpose matrix* untuk membentuk *symmetric matrix*. *Symmetric matrix* tersebut kemudian dinormalisasi dengan menghitung probabilitas setiap elemen dalam matriks tersebut. Langkah terakhir adalah menghitung fitur tekstur dari GLCM. Setiap fitur dihitung pada jarak satu piksel dalam empat arah, 0 °, 45 °, 90 °, dan 135 °, untuk mendeteksi kejadian bersama(co-occurrence) [17], seperti yang terlihat pada Gambar 2.2.




Gambar 2.2. Arah Sudut GLCM

Sumber: [17]

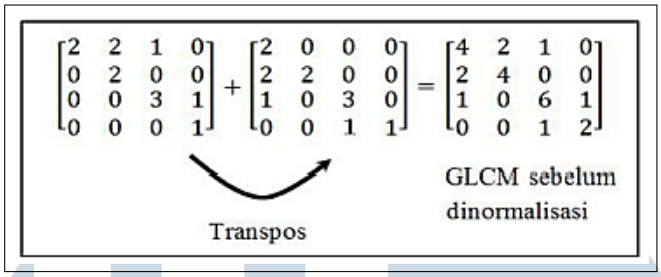
Untuk ilustrasi yang ditunjukkan dalam Gambar 2.2, ketetanggaan piksel dapat dipilih kearah timur(kanan). Salah satu cara untuk merepresentasikan hubungan ini yakni berupa (1,0), yang menyatakan hubungan dua piksel yang berjajar horizontal dengan piksel bernilai 1 diikuti dengan piksel bernilai 0. Berdasarkan komposisi tersebut, jumlah kelompok piksel yang memenuhi hubungan tersebut dihitung [17].




Gambar 2.3. Pasangan 2 Piksel GLCM Matrix

Sumber: [17]

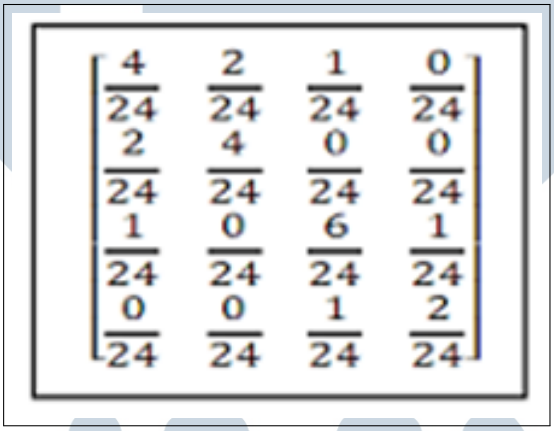
*Matrix* pada Gambar 2.3 merupakan *co-occurrence matrix*. *Matrix* ini perlu ditambahkan dengan hasil transposnya untuk membentuk *symmetric matrix* dengan cara seperti pada Gambar 2.4.




Gambar 2.4. Pembentukan Symmetric Matrix

Sumber: [17]

Untuk menghilangkan ketergantungan pada ukuran citra, nilai-nilai GLCM elemen perlu dinormalisasikan sehingga jumlahnya bernilai 1 seperti pada Gambar 2.5.



Gambar 2.5. Normalisasi Matrik Dari Citra

Sumber: [17]

Nilai-nilai dalam matriks *co-occurrence* yang sudah dinormalisasi, kemudian digunakan sebagai acuan untuk menghitung fitur-fitur tekstur dari GLCM. Haralick memperkenalkan 14 fitur GLCM yang terdiri dari 7 fitur utama dan 7 fitur tambahan yang merupakan turunan dari 7 fitur utama tersebut, yaitu [18] [19]:

1. Angular Second Moment (ASM) atau energy

   ASM atau *energy* menghitung homogenitas atau keseragaman citra. Ketika piksel citra sangat mirip, maka nilai ASM akan semakin tinggi.

$$f_l = \sum_i \sum_j \{p(i,j)\}^2 \tag{2.2}$$

   dengan:

   $f_l$: fitur *angular second moment*

$i$ : baris

$j$ : kolom

2. Contrast

   *Contrast* adalah ukuran intensitas atau perubahan skala abu-abu antara piksel yang berbeda dan piksel tetangganya. Penglihatan visual mungkin berbeda dengan munculnya dua bidang atau lebih yang terlihat secara serempak.

$$f_2 = \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} (i-j)^2 p(i,j) \tag{2.3}$$

   dengan:

   $f_2$: fitur contrast

   $N_g$: ukuran matriks hasil kuantisasi

3. Correlation

   *Correlation* menghitung wilayah linier dari derajat keabuan dalam *co-occurrence* matriks. Ini menunjukkan bagaimana piksel yang menjadi referensi dihubungkan dengan tetangganya.

$$f_3 = \sum_i \sum_j \frac{(i-\mu i)(j-\mu j)p_{(i,j)}}{\sigma i \sigma j} \tag{2.4}$$

   dengan:

   $f_3$: fitur *correlation*

$$\mu i = \sum_i \sum_j i p_{(i,j)} \tag{2.5}$$

$$\mu j = \sum_i \sum_j j p_{(i,j)} \tag{2.6}$$

$$\sigma i = \sqrt{\sum_i \sum_j (i-\mu i)^2 p_{(i,j)}} \tag{2.7}$$

$$\sigma j = \sqrt{\sum_i \sum_j (j-\mu j)^2 p_{(i,j)}} \tag{2.8}$$

4. Inverse Difference Moments (IDM) atau Homogenitas

   IDM sering disebut sebagai homogenitas, yang menghitung homogenitas

lokal sebuah citra digital. IDM membalik pengukuran jarak kedekatan dari penyebaran elemen-elemen GLCM menjadi GLCM diagonal.

$$f_4 = \sum_i \sum_j \frac{1}{1+(i-j)^2} p(i,j) \tag{2.9}$$

dengan:
$f_4$ : fitur *inverse difference moment*

5. Dissimilarity
   *Dissimilarity* mengukur ketidakmiripan tekstur, jika acak akan memiliki nilai yang lebih besar, jika seragam akan memiliki nilai yang lebih kecil.

$$f_5 = \sum_i \sum_j |i-j| \cdot p(i,j) \tag{2.10}$$

dengan:
$f_5$ : fitur *dissimilarity*

6. Entropy
   *Entropy* menunjukkan sejumlah informasi dari citra yang dibutuhkan untuk kompresi citra.

$$f_6 = -\sum_i \sum_j p(i,j) \log(p(i,j)) \tag{2.11}$$

dengan:
$f_6$ : fitur *entropy*

7. Autocorrelation
   *Autocorrelation* mengukur correlation diantara garis diagonal utama.

$$f_7 = \sum_i \sum_j i \cdot j \cdot p(i,j) \tag{2.12}$$

dengan:
$f_7$ : fitur *autocorrelation*

8. Variance
   Rumus menghitung variasi warna keabuan.

$$f_8 = \sum_i \sum_j (i-\mu)^2 p(i,j) \tag{2.13}$$

dengan:

$f_8$: fitur *variance*

$\mu$ : rata-rata dari matriks kookurensi yang telah dinormalisasi

9. Sum Average (Mean)

$$f_9 = \sum_{i=2}^{2N_g} i p_{x+y}(i) \tag{2.14}$$

dengan:

$f_9$ : fitur *sum average*

$$p_{x+y}(k) = \sum_{\substack{i=1 \\ i+j=k}}^{Ng} \sum_{j=1}^{Ng} p(i,j) \quad k = 2,3,\ldots 2Ng \tag{2.15}$$

$$p_{x-y}(k) = \sum_{\substack{i=1 \\ |i-j|=k}}^{Ng} \sum_{j=1}^{Ng} p(i,j) \quad k = 0,1,\ldots Ng-1 \tag{2.16}$$

10. Sum Variance

$$f_{10} = \sum_{i=2}^{2Ng} (i - f_{11})^2 \, p_{x+y}(i) \tag{2.17}$$

dengan:

$f_{10}$ : fitur *sum variance*

11. Sum Entropy

$$f_{11} = -\sum_{i=2}^{2N_g} p_{x+y} \log \{p_{x+y}(i)\} \tag{2.18}$$

dengan:

$f_{11}$ : fitur *sum entropy*

12. Difference Variance

$$f_{12} = \text{ variance dari } p_{x-y}$$

dengan:

$f_{12}$ : fitur *difference variance*

13. Difference Entropy

$$f_{13} = -\sum_{i=0}^{N_{g-1}} p_{x-y}(i) \log \{p_{x-y}(i)\} \tag{2.19}$$

dengan: $f_{13}$ : fitur *difference entropy*

14. Information Measure of Correlation (IMC)

$$f_{14} = \frac{HXY - HXY1}{\max\{HX, HY\}} \tag{2.20}$$

dengan:
$f_{14}$ : fitur IMC
$HX$ dan $HY$ : *entropy* dari $p_x$ dan $p_y$

$$HXY = -\sum_{i}\sum_{\mathrm{j}} p(i,j) \log p(i,j) \tag{2.21}$$

$$HXYI = -\sum_{i}\sum_{j} p(i,j) \log \{p_x(i) p_y(j)\} \tag{2.22}$$

Langkah-langkah perhitungan GLCM adalah sebagai berikut: [15]:

1. Pembentukan matriks awal GLCM dari pasangan dua piksel yang berjajar sesuai dengan orientasi sudut 0°, 45°, 90° atau 135°.
2. Tambahkan matriks GLCM awal dan transposnya untuk membentuk matriks simetris.
3. Normalisasi matriks GLCM dengan membagi tiap elemen matriks dengan jumlah pasangan piksel.
4. Melakukan ekstraksi fitur dengan menghitung sesuai rumusnya, fitur yang digunakan yaitu: ASM, *contrast, correlation, homogeneity, dissimiliarity, entropy* dan *autocorrelation*.

## 2.5 Jaringan Saraf Tiruan Backpropagation

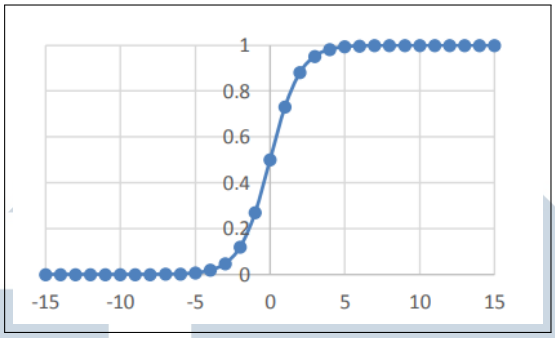
Jaringan saraf tiruan banyak digunakan dalam berbagai kebutuhan yang memiliki hubungan kompleks antara *input* dan *output* dengan kemampuan untuk melakukan pemrosesan jaringan pada hubungan antar *unit* atau bobot yang diperoleh dari proses adaptasi, melalui pembelajaran dari serangkaian pola yang telah

terlatih[9]. Salah satu algoritma jaringan saraf tiruan yang akan digunakan adalah *Backpropagation*.

*Backpropagation* adalah algoritma pembelajaran yang mengurangi tingkat kesalahan dengan menyesuaikan bobotnya berdasarkan perbedaan antara *output* dan perolehan target yang diharapkan[20]. Algoritma *Backpropagation* biasanya digunakan untuk memecahkan masalah prediksi. Hal ini dimungkinkan karena *Backpropagation* merupakan metode pelatihan jaringan saraf tiruan terawasi(supervisi)[21]. *Backpropagation* juga merupakan pendekatan sistematis untuk melatih jaringan saraf tiruan *multi-layer* karena *Backpropagation* memiliki tiga lapisan dalam proses pelatihannya, yaitu lapisan *input*, lapisan tersembunyi, dan lapisan keluaran[20]. Jaringan diberikan sepasang pola yang terdiri dari pola masukkan dan pola yang diinginkan. Ketika sebuah pola ditetapkan ke jaringan, bobot diubah untuk meminimalkan perbedaan antara pola keluaran dan pola yang diinginkan. Pelatihan dilakukan berulang-ulang sampai pola yang dihasilkan dapat memenuhi pola yang diinginkan.

Pada lapisan *input* tidak ada proses komputasi, hanya mengirimkan sinyal input ke lapisan tersembunyi. Pada lapisan tersembunyi dan lapisan keluaran, terdapat proses perhitungan bobot(*weight*) dan bias, dan dihitung juga besarnya *output* dari lapisan tersembunyi dan lapisan keluaran(*output*) menggunakan fungsi aktivasi [20]. Tujuan penggunaan fungsi aktivasi yaitu untuk melakukan pengaktifan terhadap *neuron*[22]. Sifat yang harus dimiliki oleh fungsi aktivasi pada jaringan Backpropagation yaity: kontinu, terdiferensiasi, dan tidak menurun secara monoton.[23]. Fungsi aktivasi terbagi menjadi dua macam yaitu fungsi linear, dan non-linear. Pada kasus yang lebih kompleks fungsi aktivasi non-linear lebih disarankan untuk digunakan karena dapat menyusun konsep yang berbeda secara bersamaan. Sehingga mampu memecahkan masalah yang kompleks[22]. Ada tiga fungsi aktivasi yang banyak digunakan yaitu fungsi aktivasi sigmoid, fungsi aktivasi tanh, dan fungsi aktivasi ReLU[24].

(a) Fungsi aktivasi Sigmoid
Fungsi aktivasi sigmoid mengubah rentang nilai input x antara 0 dan 1 dengan distribusi fungsi seperti pada Gambar 2.6.




Gambar 2.6. Distribusi Fungsi Sigmoid
(Sumber: [24])

Fungsi aktivasi sigmoid memiliki persamaan sebagai berikut :

$$\sigma(x) = \frac{1}{(1+e^{-x})} \tag{2.23}$$

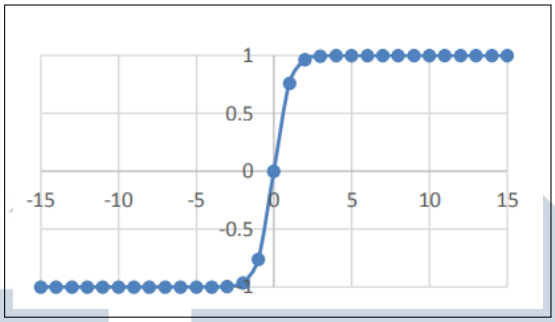
Fungsi sigmoid tidak banyak digunakan dalam praktek sekarang, karena memiliki kelemahan besar yaitu rentang nilai output dari fungsi sigmoid tidak terpusat pada nol. Hal tersebut menyebabkan terjadinya proses *backpropagation* yang tidak ideal dikarenakan bobot pada JST tidak terdistribusi rata antara nilai *positif* dan *negatif* serta nilai bobot akan banyak mendekati ekstrim 0 atau 1. Dikarenakan komputasi nilai propagasi menggunakan perkalian, maka nilai ekstrim tersebut akan menyebabkan efek *saturating gradients* dimana jika nilai bobot cukup kecil, maka lama kelamaan nilai bobot akan mendekati salah satu ekstrim sehingga memiliki gradien yang mendekati nol. Jika hal tersebut terjadi, maka neuron tersebut tidak akan dapat mengalami *update* yang signifikan dan akan nonaktif.

(b) Fungsi aktivasi Tanh

Fungsi aktivasi tanh memiliki distribusi yang sangat mirip dengan fungsi sigmoid namun dalam *range* antara -1 sampai 1 dan distribusi yang lebih sempit. Fungsi tanh memiliki bentuk sebagai berikut :

$$\sigma(x) = \tanh(x) \tag{2.24}$$

Sifat *zero-center* tersebut mengatasi kelemahan utama pada fungsi aktivasi sigmoid, sehingga dalam prakteknya fungsi aktivasi tanh selalu mengungguli fungsi aktivasi sigmoid.

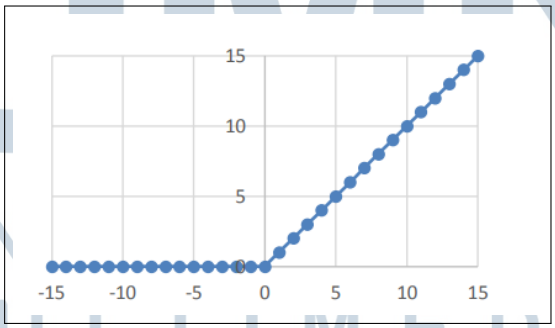



Gambar 2.7. Distribusi Fungsi Tanh
(Sumber: [24])

(c) Fungsi aktivasi ReLU (*Rectified Linear Unit*)
Fungsi ReLU termasuk salah satu fungsi aktivasi baru. Fungsi ReLU melakukan operasi *thresholding* nilai linear pada nol. Secara matematis fungsi ReLU memiliki bentuk:

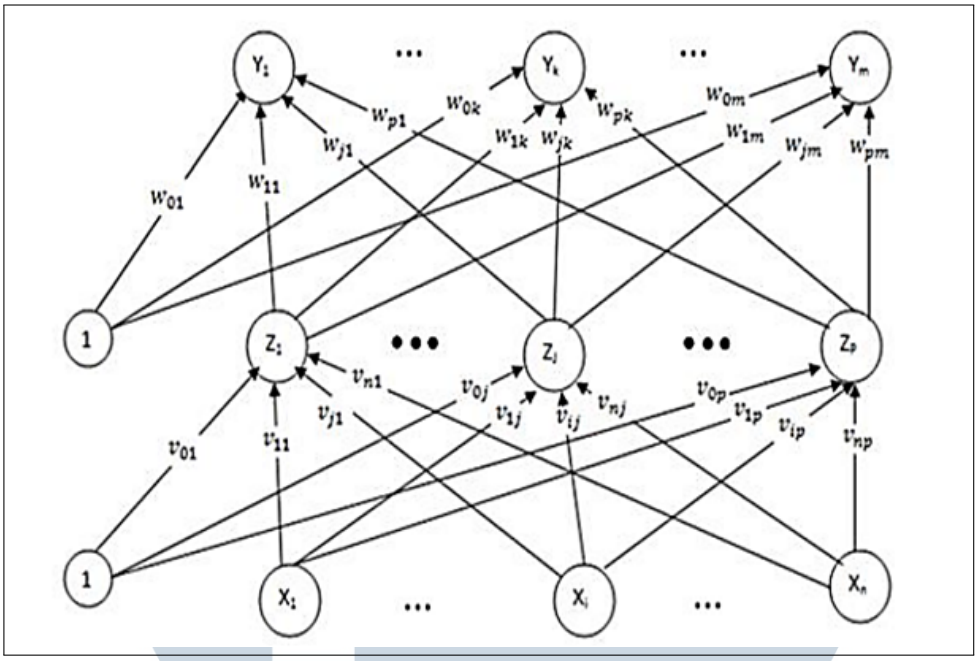
$$\sigma(x) = \max(0, x) \tag{2.25}$$

ReLU adalah operasi antara sebuah *input* dengan jangkauan 0-max (x,0). Memiliki properti dengan aturan pada bagian ujung dari garis *gradient* tidak menyentuh nilai 0, sehingga membantu kecepatan konvergensi dalam pelatihan jaringan saraf [22]. Menurut Krizhevsky et. al., konvergensi proses pelatihan jika menggunakan fungsi ReLU lebih cepat hingga 6 kali dibandingkan fungsi tanh.




Gambar 2.8. Distribusi Fungsi ReLU
(Sumber: [24])

Arsitektur jaringan *Backpropagation* terdiri atas lapisan masukan (*input layer*), lapisan tersembunyi (*hidden layer,*) dan lapisan keluaran(*output layer*)[21]. Arsitektur *Backpropagation* dapat dilihat pada Gambar 2.9.




Gambar 2.9. Arsitektur Jaringan Backpropagation dengan Satu Hidden Layer

Sumber: [21]

dengan keterangan:

$X$ = *Input layer*

$Z$ = *Hidden layer*

$Y$ = *Output layer*

$V_{ij}$ = Bobot pada *hidden layer*

$V_{0j}$ = Bias pada *hidden layer*

$W_{jk}$ = Bobot pada *output layer*

$W_{0k}$ = Bias pada *output layer*

$Z\_in_j$ = Faktor keluaran pada unit tersembunyi

$Y\_in_k$ = Faktor keluaran pada unit *output*

$\delta_j$= Faktor kesalahan pada lapisan tersembunyi

$\delta_k$= Faktor kesalahan pada lapisan *output*

$\alpha$= Laju pembelajaran (Learning rate)

Algoritma dari jaringan saraf tiruan backpropagation dapat dijelaskan sebagai berikut[25]:

1. Menentukan nilai parameter seperti: *learning rate*, *epoch*, *error tolerance* atau target error dan jumlah lapisan yang akan digunakan.

2. Inisialisasi bobot(*weight*) dan bias dengan nilai acak (Algoritma Nguyen Widrow).

3. Jika kondisi penghentian belum terpenuhi, lakukan langkah 4 hingga 8.

4. Untuk masing-masing pasangan data *training* lakukan langkah 5 hingga 7.

5. Proses *feed forward* atau propagasi maju:

   (a) Setiap unit masukan ($X_i$, $i$ =1,2... total masukan) menerima sinyal masukan $X_i$ kemudian meneruskannya ke semua unit pada *hidden layer*.

   (b) Perhitungan tiap sinyal masukan yang sudah terboboti termasuk biasnya pada tiap unit tersembunyi ($Z_j$, $j$ =1,2..,p)

   $$Z\_in_j = V_{0j} + \sum X_i V_{ij} \tag{2.26}$$

   (c) Perhitungan sinyal keluaran dari unit tersembunyi dengan fungsi aktivasi ReLu.

   $$Z_j = f\left(Z\_in_j\right) = f_{\max}\left(0, Z\_in_j\right) \tag{2.27}$$

   Sinyal keluaran ini, kemudian diteruskan ke unit pada *output layer*.

   (d) Setiap unit hidden $Z_j$ menyebarkan sinyal ke setiap unit output ($Y_k$, $k$ =1,2..,m):

   $$Y\_in_k = W_{0k} + \sum Z_j \cdot W_{jk} \tag{2.28}$$

   Lalu, dilakukan perhitungan nilai *output* menggunakan fungsi aktivasi ReLu:

   $$Y_k = f\left(Y\_in_k\right) = f_{\max}\left(0, Y\_in_k\right) \tag{2.29}$$

6. Proses *Backpropagation* atau propagasi mundur:

   (a) Setiap unit output ($Y_k$, $k$ =1,2..,m) menerima pola target yang sesuai dengan pola input (data citra). Untuk menghitung eror antara target yang di input-kan dengan output yang dihasilkan oleh jaringan digunakan rumus:

   $$\delta_k = \left(t_k - Y_k\right) f'\left(Y\_in_k\right) \tag{2.30}$$

   (b) Error tersebut digunakan untuk memperbaiki nilai bobot ($\Delta W_{jk}$) dan bias ($\Delta W_{0k}$) di lapisan bawahnya (hidden layer) dengan laju pembelajaran

atau *learning rate*($\alpha$) dengan persamaan berikut:

$$\Delta W_{jk} = \alpha \delta_k Z_j \tag{2.31}$$

$$\Delta W_{0k} = \alpha \delta_k \tag{2.32}$$

(c) Setiap unit tersembunyi $Z_j$ menerima input delta dari proses sebelumnya. Kemudian dilakukan perhitungan galat di setiap unit tersembunyi berdasarkan kesalahan tiap unit tersembunyi.

$$\delta\_in_j = \sum \delta_k W_{jk} \tag{2.33}$$

(d) Kemudian nilai tersebut dikalikan dengan hasil turunan dari fungsi aktivasi untuk mendapatkan informasi kesalahan :

$$\delta_j = \delta\_in_j f'(Y\_in_k) \tag{2.34}$$

(e) Perhitungan koreksi nilai bobot yang digunakan untuk memperbaiki bobot($\Delta V_{ij}$):

$$\Delta V_{ij} = \alpha \delta_j X_i \tag{2.35}$$

(f) Perhitungan koreksi nilai bias yang digunakan untuk memperbaiki bias($\Delta V_{0j}$):

$$\Delta V_{0j} = \alpha \delta_j \tag{2.36}$$

7. Perbaikan nilai bobot dan bias:

(a) Perhitungan perubahan bobot dan bias di unit tersembunyi menuju unit output:

$$W_{jk(baru)} = W_{jk(lama)} + \Delta W_{jk} \tag{2.37}$$

(b) Perhitungan perubahan bobot dan bias di unit input menuju unit tersembunyi:

$$V_{ij(baru)} = V_{ij(lama)} + \Delta V_{ij} \tag{2.38}$$

8. Pengecekan apakah kondisi berhenti sudah terpenuhi yaitu saat iterasi mencapai maksimum iterasi(epoch) atau error lebih kecil daripada target error atau *error tolerance*, namun jika belum terpenuhi maka lakukan langkah 4 sampai

8.

## 2.6 Evaluasi Performa

Salah satu cara mengevaluasi performa model *machine learning* yang telah dibuat adalah menggunakan *confusion matrix*. *Confusion matrix* adalah matriks yang sering digunakan untuk mengukur kinerja suatu algoritma klasifikasi. *Confusion matrix* berisi informasi tentang klasifikasi aktual dan prediksi dilakukan dengan klasifikasi sistem yang dievaluasi menggunakan data dalam matriks[26]. Berikut bentuk *confusion matrix* serta penjelasannya dapat dilihat pada Tabel 2.1 [27].

Tabel 2.1. Tabel Confusion Matrix

| | Positive (Aktual) | Negative (Aktual) |
|---|---|---|
| Positive (Prediksi) | TP | FP |
| Negative (Prediksi) | FN | TN |

Sumber: [27]

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

Dalam *Confusion Matrix* terdapat beberapa metrik yang dapat diukur seperti *accuracy, recall, precision,* dan *F1-score*,adapun penjelasan mengenai metrik tersebut adalah sebagai berikut [28]:

1. Accuracy
   *Accuracy* adalah metrik yang menyatakan persentase pengamatan yang diprediksi dengan benar sebagai TP(*True Positive*) ataupun TN(*True Negative*) dibandingkan dengan semua data. Untuk menghitung metrik *accuracy* digunakan rumus berikut.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \times 100\% \tag{2.39}$$

2. Recall

   *Recall* merepresentasikan rasio prediksi *true positive* dibandingkan dengan jumlah keseluruhan data *positive*. Untuk mengkalkulasikan metrik *recall* digunakan rumus sebagai berikut.

$$\text{Recall} = \frac{TP}{TP+FN} \times 100\% \tag{2.40}$$

3. Precision

   *Precision* merupakan metrik yang mewakili rasio dari TP(*True Positive*) dibandingkan dengan seluruh hasil yang bernilai positif. Untuk mengkalkulasikan metrik *precision* digunakan rumus sebagai berikut.

$$\text{Precision} = \frac{TP}{TP+FP} \times 100\% \tag{2.41}$$

4. F1-Score

   *F1-score* mewakili kombinasi nilai *precision* dan *recall*. *F1-score* menghitung nilai *mean* antara kedua nilai tersebut. Untuk mengkalkulasikan metrik *F1-score* digunakan rumus sebagai berikut.

$$F1-\text{Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \times 100\% \tag{2.42}$$

Perbedaan antara *macro average, micro average,* dan *weighted average* adalah *macro average* menghitung matriks secara bebas untuk setiap kelas kemudian mengambil rata-ratanya dan *micro average* mengitung matriks rata-rata dengan mengumpulkan kontribusi dari semua kelas. Sedangkan *weighted average*, akan menghitung rata-rata dengan memperhitungkan bobot pada setiap datanya [26].