BAB 2
LANDASAN TEORI

2.1 Face Recognition

Wajah adalah salah satu cara untuk dapat mengenali atau mengingat seseorang. Setiap orang memiliki karakteristik wajah yang unik, sehingga sangat jarang orang yang memiliki karakteristik wajah yang sama [12]. Setiap orang dapat memiliki bentuk tulang wajah dan ciri khas yang berbeda, oleh karena itu wajah dapat dijadikan media untuk mendapatkan informasi.

Face Recognition adalah aplikasi biometrik yang secara matematis memetakan fitur wajah seseorang dan menyimpannya sebagai sebuah data [13]. Sebuah sistem *Face Recognition* biasanya memiliki struktur sebagai berikut:

1. *Image Preprocessing*, merupakan proses penyesuaian terhadap citra dari gambar wajah agar citra dapat diproses dan dapat menghasilkan hasil yang maksimal.

2. *Face Detection*, merupakan proses mendeteksi letak wajah pada citra yang telah disesuaikan pada tahap *preprocessing*.

3. *Face Normalization*, merupakan proses penyesuaian terhadap citra agar setiap citra memiliki perbedaan yang tidak terlalu signifikan.

4. *Feature Extraction*, merupakan proses ekstraksi fitur dari wajah untuk mendapatkan fitur-fitur penting dari wajah yang bisa digunakan untuk proses identifikasi pada wajah.

5. *Classification*, merupakan proses klasifikasi untuk memeriksa wajah berdasarkan hasil ekstraksi fitur.

2.2 Ekstraksi Fitur

Ekstraksi fitur melibatkan proses pengambilan informasi khas dari suatu objek yang dapat dibedakan dengan objek lain [14]. Informasi yang diekstraksi ini akan digunakan sebagai parameter dalam proses klasifikasi. Fitur-fitur ini mencerminkan karakteristik unik dari objek tersebut, sehingga terdapat beberapa kriteria yang perlu dipenuhi oleh fitur-fitur tersebut [6]:

1. Mampu membedakan objek dengan objek lainnya (*discrimination*).
2. Memperhatikan kompleksitas komputasi.
3. Bersifat *independent* (tidak terikat), artinya fitur-fitur tersebut invariant terhadap perubahan seperti rotasi, penskalaan, pergeseran, dan lainnya.
4. Jumlah fitur yang sedikit untuk mengurangi waktu komputasi dan kebutuhan ruang penyimpanan dalam proses berikutnya.

2.3 Discrete Cosine Transform (DCT)

Discrete Cosine Transform (DCT) digunakan untuk mengkompres data dengan mengelompokkan gambar 2D menjadi beberapa pita frekuensi. DCT adalah algoritma kompresi yang paling populer yang disebut JPEG, dan banyak diterapkan dalam pemrosesan gambar dan video [15]. DCT adalah sebuah metode untuk mengubah citra dari domain spasial menjadi sebuah sekumpulan blok frekuensi berukuran NxN [16]. DCT dapat dinyatakan pada persamaan 2.1.

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

Di mana:

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

2.4 Gaussian Mixture Model (GMM)

Gaussian Mixture Model (GMM) adalah model probabilistik yang digunakan untuk menganalisis data yang terdiri dari beberapa komponen *Gaussian*

yang saling tumpang tindih. Model ini dapat digunakan untuk *clustering data* dan juga dapat digunakan untuk mengidentifikasi distribusi data yang mendasarinya [17]. Rumus dasar untuk GMM adalah sebagai berikut [18]:

$$P(X|\Theta) = \sum_{k=1}^{K} \pi_k \cdot N(X|\mu_k, \Sigma_k) \tag{2.2}$$

Di mana:

- $P(X|\Theta)$ = probabilitas data X diberikan parameter Θ dalam GMM.
- K = jumlah komponen *Gaussian* dalam GMM.
- π_k = bobot untuk setiap komponen *Gaussian*, yang mengindikasikan proporsi atau probabilitas kemunculan komponen tersebut.
- $N(X|\mu_k, \Sigma_k)$ = fungsi kepadatan *Gaussian* untuk komponen k dengan rata-rata μ_k dan matriks kovarians Σ_k.

Tujuan utama dari GMM adalah untuk menemukan parameter Θ yang optimal yang memberikan probabilitas tertinggi untuk data yang diberikan. Untuk menentukan parameter-parameter model, GMM menggunakan algoritma *Expectation-Maximization* (EM), di mana pada tahap E (*Expectation*) dihitung nilai harapan dari setiap komponen *Gaussian* dalam campuran, dan pada tahap M (*Maximization*) dihitung ulang parameter-parameter model menggunakan nilai harapan tersebut [19]. Proses EM dalam GMM melibatkan dua tahap [18]:

1. Tahap Ekspektasi (*Expectation*): Mengestimasi probabilitas posterior dari setiap komponen *Gaussian* (kelompok) untuk setiap data poin. Rumus perhitungan probabilitas posterior (responsibilitas) untuk setiap komponen *Gaussian*:

$$r_{ik} = \frac{\pi_k \cdot N(x_i|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \cdot N(x_i|\mu_j, \Sigma_j)} \tag{2.3}$$

Di mana:

- r_{ik} = probabilitas posterior bahwa data ke-i berasal dari komponen *Gaussian* ke-k.
- x_i = data ke-i.
- μ_k = rata-rata komponen *Gaussian* ke-k.
- Σ_k = matriks kovarians komponen *Gaussian* ke-k.

- π_k = bobot komponen *Gaussian* ke-k.

2. Tahap Maksimisasi (*Maximization*): Menggunakan probabilitas posterior yang diestimasi pada tahap Ekspektasi untuk memperbarui parameter GMM, termasuk bobot, rata-rata, dan matriks kovarians.

   - Pembaruan bobot (π_k) untuk setiap komponen *Gaussian* :

   $$\pi_k^{new} = \frac{1}{N}\sum_{i=1}^{N} r_{ik} \tag{2.4}$$

   Di mana:

   – π_k^{new} = bobot yang diperbarui untuk komponen *Gaussian* ke-k.
   – N = jumlah total data.
   – r_{ik} = probabilitas posterior yang dihitung pada tahap Ekspektasi.

   - Pembaruan rata-rata (μ_k) untuk setiap komponen *Gaussian*:

   $$\mu_k^{new} = \frac{\sum_{i=1}^{N} r_{ik} \cdot x_i}{\sum_{i=1}^{N} r_{ik}} \tag{2.5}$$

   Di mana:

   – μ_k^{new} = rata-rata yang diperbarui untuk komponen *Gaussian* ke-k.
   – N = jumlah total data.
   – r_{ik} = probabilitas posterior yang dihitung pada tahap Ekspektasi.
   – x_i = data ke-i.

   - Pembaruan matriks kovarians (Σ_k) untuk setiap komponen *Gaussian*:

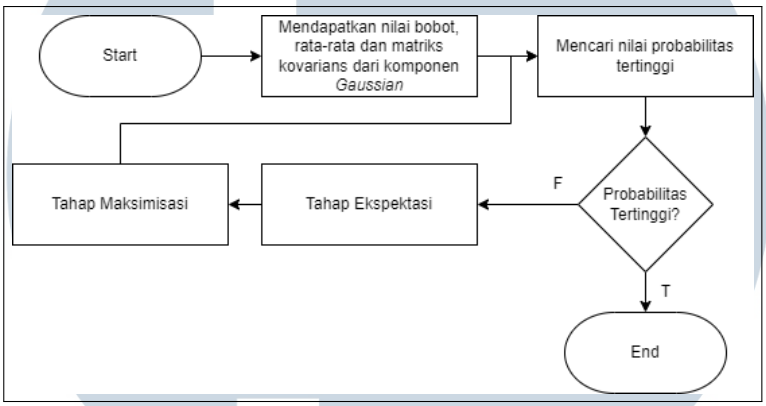
   $$\Sigma_k^{new} = \frac{\sum_{i=1}^{N} r_{ik} \cdot (x_i - \mu_k^{new})(x_i - \mu_k^{new})^T}{\sum_{i=1}^{N} r_{ik}} \tag{2.6}$$

   Di mana:

   – Σ_k^{new} = matriks kovarians yang diperbarui untuk komponen *Gaussian* ke-k.
   – N = jumlah total data.
   – r_{ik} = probabilitas posterior yang dihitung pada tahap Ekspektasi.
   – μ_k^{new} = rata-rata yang diperbarui.

– x_i = data ke-i.

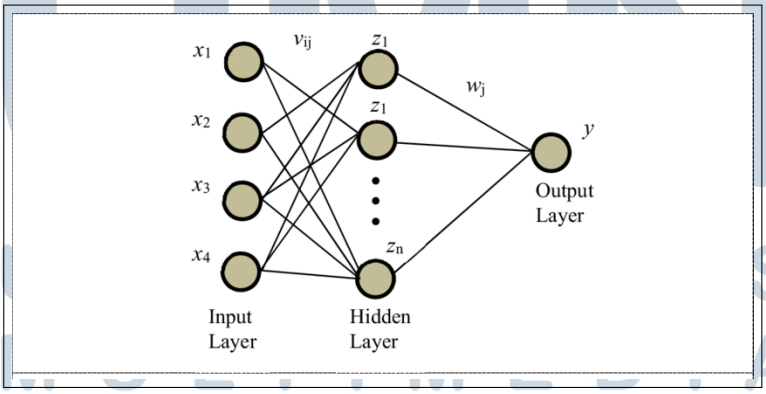
Proses iterasi *Expectation-Maximization* berlanjut hingga konvergensi, yaitu ketika parameter GMM stabil atau perbedaan antara iterasi yang berurutan menjadi sangat kecil. Dapat dilihat pada Gambar 2.1 yang merupakan *flowchart* untuk proses dari GMM.



Gambar 2.1. *Flowchart* alur sistem GMM
Sumber: [19]

2.5 BackPropagation (BackProp)

Backpropagation (BackProp) merupakan teknik jaringan saraf tiruan yang memiliki 3 macam layer [20], antara lain:



Gambar 2.2. Arsitektur diagram backpropagation
Sumber: [21]

1. *Input Layer*, merupakan bagian yang terdiri dari unit di mana unit dimulai dari 1 sampai ke-*n*.

2. *Hidden Layer*, merupakan *layer* yang setidaknya terdiri dari satu *layer* di mana setiap *layer*-nya terdiri dari beberapa unit.

3. *Output Layer*, merupakan setiap unit *neuron* pada *layer input* terhubung terhadap semua unit pada *layer* tersembunyi di bawahnya. Begitu juga sebaliknya, setiap unit pada *layer* tersembunyi terhubung ke semua unit pada *layer output*.

Pada BackProp terdapat suatu siklus yaitu siklus *Epoch* yang merupakan siklus proses pelatihan pada jaringan saraf tiruan. Proses pelatihannya meliputi sebagai berikut [22]:

1. Propagasi Maju
   Pada proses ini, sinyal input dari *input layer* dipropagasikan ke seluruh unit pada *hidden layer*. Selama proses berjalan, bobot atau *weight* dan nilai *offset* dari *neural network* dijaga agar tetap stabil dan setiap *neuron layer* hanya dapat memberikan efek pada *neuron layer* berikutnya. Jika hasil proses propagasi dari *Output* atau *Output* harapan dari seluruh unit *hidden layer* pada *output layer* tidak dapat dicapai, maka proses tersebut akan dimasukkan ke proses propagasi mundur.

2. Propagasi Mundur
   Pada proses ini, perbedaan setiap *output* sebenarnya dan *output* harapan dapat didefinisikan sebagai *error signal*. Pada proses ini, *error signal* akan disebarkan dengan cara berjalan mundur melalui *hidden layer*, propagasi akan berjalan dari *output layer* hingga ke *input layer*. Selama propagasi mundur berjalan, bobot atau *weight* dari jaringan saraf tiruan akan diatur berdasarkan *error feedback* dan kemudian masuk kedalam proses modifikasi bobot.

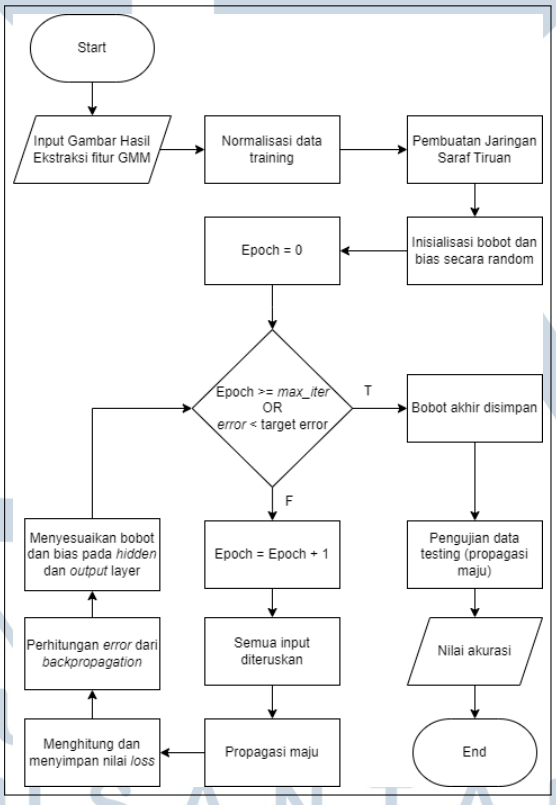
3. Modifikasi Bobot
   Setelah seluruh faktor didapatkan, maka akan dilanjutan dengan memodifikasi bobot seluruh jaringan secara bersamaan. Proses modifikasi bobot akan berjalan secara terus menerus dari bobot atau *weight* dan nilai *offset* dengan tujuan untuk mendapatkan *output* sebenarnya yang ada pada jaringan saraf tiruan dapat mendekati hasil output harapannya.

Setelah pelatihan selesai dilakukan, jaringan dapat dipakai untuk perkenalan pola. Dalam hal ini, hanya langkah propagesi maju yang digunakan untuk keluaran jaringan. Kemudian dilakukan normalisasi dengan menggunakan fungsi sigmoid.

Fungsi sigmoid adalah fungsi asimtotik yang nilainya tidak pernah mencapai 0 ataupun 1. Fungsi sigmoid tidak banyak digunakan dalam praktik saat ini karena memiliki kelemahan besar yaitu rentang keluaran fungsi sigmoid tidak berpusat di sekitar 0. Hal ini menyebabkan proses backpropagation kurang optimal, karena bobot pada JST tidak terdistribusi secara merata antara nilai positif dan negatif, dan nilai bobot akan mendekati nilai ekstrim 0 atau 1. Karena perhitungan nilai propagasi menggunakan perkalian, maka nilai ekstrim tersebut akan menyebabkan gradien menjadi jenuh, dan jika nilai bobot cukup kecil, maka lama kelamaan nilai bobot akan mendekati salah satu nilai ekstrim tersebut, memberikannya gradien mendekati 0. Jika ini terjadi, neuron tidak akan dapat melakukan pembaruan yang signifikan dan akan dinonaktifkan. Rumus normalisasi adalah sebagai berikut [21].

$$x' = \frac{0.8(x - nilai\,terkecil)}{nilai\,terbesar - nilai\,terkecil} + 0.1 \tag{2.7}$$

Flowchart untuk proses *Backpropagation* dapat dilihat pada Gambar 2.3.



Gambar 2.3. *Flowchart* alur proses *backpropagation*

Sumber: [6]

2.6 Confusion Matrix

Confusion Matrix merupakan salah satu cara untuk melakukan evaluasi performa dari sebuah model *machine learning*. *Confusion Matrix* memiliki infomrasi tentang klasifikasi aktual dan prediksi dilakukan dengan sistem klasifikasi yang dievaluasi menggunakan data dan matrix. Berikut bentuk tabel *confusion matrix* dan penjelasannya [23].

Tabel 2.1. Tabel *Confusion matrix*

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

Berikut merupakan penjelasan dari Tabel 2.1,

1. TP (*True Positive*), merupakan data yang diprediksikan positif dan memiliki hasil klasifikasi aktual yang positif.
2. FP (*False Positive*), merupakan data yang diprediksikan negatif tetapi memiliki hasil klasifikasi aktual yang positif.
3. TN (*True Negative*), merupakan data yang diprediksikan negatif dan memiliki hasil klasifikasi aktual yang negatif.
4. FN (*False Negative*), merupakan data yang diprediksikan positif tetapi memiliki hasil klasifikasi aktual yang negatif.

Confusion Matrix memiliki beberapa matriks yang memungkinkan untuk diukur seperti *Accuracy, Recall, Precision,* dan *F1-Score*. Berikut merupakan rumus yang digunakan [24]:

1. *Accuracy*, merupakan matriks yang dapat memberikan persentasi prediksi dari nilai yang benar atau "*True*" yang terdiri dari TP dan FN. Persamaan untuk mencari akurasi dapat dilihat pada persamaan 2.8.

$$Accuracy = \frac{TP+FN}{TP+TN+FP+FN} X100\% \tag{2.8}$$

2. *Recall*, merupakan rasio dari nilai data TP *(True Positive)* terhadap keseluruhan data positif. Persamaan untuk mencari nilai *Recall* dapat dilihat

pada persamaan 2.9.

$$Recall = \frac{TP}{TP + FN} X100\% \tag{2.9}$$

3. *Precision*, merupakan rasio dari nilai data TP *(True Positive)* terhadap keseluruhan data yang diprediksi positif. Persamaan untuk mencari nilai *Precision* dapat dilihat pada persamaan 2.10.

$$Precision = \frac{TP}{TP + FP} X100\% \tag{2.10}$$

4. *F1-Score*, merupakan mean dari nilai *Precision* dan *Recall*. Persamaan untuk mencari nilai *F1-Score* dapat dilihat pada persamaan 2.11.

$$F1 - Score = 2X \frac{PrecisionXRecall}{Precision + Recall} X100\% \tag{2.11}$$