BAB II

KAJIAN LITERATUR

2.1 Citra Digital

Citra digital merupakan representasi visual suatu objek dalam bentuk dua dimensi yang secara matematis dinyatakan sebagai fungsi *f(x,y)* dimana x dan y menunjukkan koordinat spasial, sedangkan nilai fungsi tersebut menyatakan intensitas cahaya atau tingkat keabuan pada titik tersebut. Citra digital tersusun atas piksel (*picture element*), yaitu elemen terkecil yang memiliki nilai numerik sebagai representasi informasi visual. Pada citra *grayscale*, nilai piksel berada dalam rentang 0–255, sedangkan pada citra berwarna setiap piksel terdiri atas komponen RGB (*Red, Green, dan Blue*) yang membentuk variasi warna tertentu [9]. Representasi numerik inilah yang memungkinkan citra diproses menggunakan sistem komputer.

Gambar 2. 1 Citra Digital (RGB) [12]

Pengolahan citra digital merupakan teknik yang digunakan untuk memanipulasi dan menganalisis citra guna meningkatkan kualitas visual maupun mengekstraksi informasi penting di dalamnya. Proses ini meliputi peningkatan kontras, reduksi *noise*, segmentasi, serta ekstraksi fitur. Salah satu metode yang efektif dalam mereduksi gangguan khususnya *salt and pepper noise* adalah *Median Filter*, yang bekerja dengan mengganti nilai piksel dengan nilai median dari piksel di sekitarnya sehingga detail penting tetap terjaga. Kualitas hasil pemrosesan citra dapat dievaluasi menggunakan parameter seperti *Mean Squared Error* (MSE), *Peak Signal-to-Noise Ratio* (PSNR), dan *Structural Similarity Index Measure* (SSIM) untuk mengetahui tingkat kemiripan antara citra asli dan citra hasil perbaikan [12].

2.1.1 Jenis Citra

Dalam pengolahan citra digital, citra dapat diklasifikasikan berdasarkan jumlah tingkat intensitas dan representasi warnanya. Secara umum, citra digital direpresentasikan sebagai

fungsi dua dimensi *f(x,y)* yang terdiri atas elemen-elemen diskrit yang disebut piksel, dimana setiap piksel memiliki nilai intensitas tertentu [13]. Representasi citra dalam bentuk matriks memungkinkan setiap koordinat menyatakan posisi dan nilai keabuan atau warna pada titik tersebut.

Berdasarkan tingkat intensitasnya, citra digital dibedakan menjadi citra biner, *grayscale*, dan berwarna. Citra biner hanya memiliki dua nilai intensitas, yaitu 0 (hitam) dan 1 (putih), dengan kebutuhan 1 bit per piksel. Citra *grayscale* memiliki 256 tingkat keabuan dengan rentang 0–255 dan direpresentasikan dalam 8 bit per piksel. Sedangkan citra berwarna tersusun atas kombinasi tiga komponen dasar RGB (*Red, Green, Blue*) dengan rentang nilai 0–255 pada masing-masing kanal sehingga menghasilkan variasi warna yang lebih kompleks [13].

(a) (b) (c)

Gambar 2. 2 (a) Citra RGB, (b) Citra *Grayscale*, (c) Citra Biner

Dalam *computer vision* dan pengolahan citra digital, pemilihan jenis citra disesuaikan dengan kebutuhan analisis dan tujuan pemrosesan. Citra *grayscale* lebih sering digunakan dalam peningkatan kualitas dan reduksi *noise* karena lebih sederhana secara komputasi dibandingkan citra berwarna. Pada penerapan *Median Filter*, citra umumnya dikonversi terlebih dahulu ke *grayscale* untuk mempermudah proses pengolahan serta evaluasi menggunakan parameter MSE, PSNR, dan SSIM [13]. Dengan demikian, setiap jenis citra memiliki peran yang berbeda dalam mendukung analisis dan identifikasi objek.

2.1.2 Representasi Citra Digital

Citra digital dapat direpresentasikan sebagai fungsi dua dimensi *F(x,y)*, dimana *x* dan *y* menyatakan koordinat spasial, sedangkan nilai fungsi tersebut menunjukkan intensitas pada setiap titik. Ketika koordinat dan nilai intensitas bersifat diskrit, maka citra tersebut disebut citra digital. Secara komputasional, citra digital terdiri atas sejumlah elemen terbatas yang disebut piksel (*picture element*), dimana setiap piksel memiliki posisi dan nilai tertentu [14].

Secara matematis, citra digital dapat dinyatakan dalam bentuk matriks berukuran $M x N$ dengan M menyatakan jumlah baris dan N jumlah kolom. Setiap elemen matriks merepresentasikan nilai intensitas pada koordinat tertentu, sehingga dapat dituliskan sebagai:

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,0) & \cdots & f(M-1,N-1) \end{bmatrix}$$

Berdasarkan gambaran tersebut setiap nilai *F(x,y)* menyatakan tingkat keabuan atau warna dari piksel pada posisi tersebut. Dalam citra *grayscale*, nilai intensitas umumnya berada pada rentang 0–255, sedangkan pada citra berwarna setiap piksel terdiri atas tiga komponen utama yaitu RGB (*Red, Green, dan Blue*) yang masing-masing memiliki rentang nilai tertentu [15].

Dalam konteks *computer vision* dan *deep learning*, representasi citra dalam bentuk matriks numerik memungkinkan proses ekstraksi fitur dan pembelajaran pola secara otomatis. Nilai piksel yang tersusun dalam matriks menjadi input bagi model deteksi objek maupun segmentasi citra untuk mengenali batas bangunan atau objek lainnya. Transformasi seperti rotasi, translasi, perubahan kontras, maupun perspektif dilakukan dengan memodifikasi koordinat atau nilai intensitas piksel berdasarkan persamaan matematis tertentu, sehingga menghasilkan variasi data tanpa mengubah informasi utama citra [16]. Representasi numerik inilah yang menjadi dasar dalam seluruh proses pengolahan citra digital, mulai dari tahap akuisisi hingga analisis dan pengenalan objek.

2.1.3 Pengolahan Citra Digital

Pengolahan citra digital merupakan cabang ilmu yang mempelajari teknik untuk membentuk, memproses, dan menganalisis citra guna meningkatkan kualitas visual serta memperoleh informasi yang lebih bermakna. Dalam praktiknya, citra yang diperoleh dari proses akuisisi sering mengalami degradasi seperti *noise, blur*, atau penurunan kontras sehingga diperlukan metode perbaikan seperti *image enhancement* dan *image restoration*. Salah satu teknik yang umum digunakan dalam reduksi *noise* adalah *Median Filter*, yang bekerja dengan mengganti nilai piksel berdasarkan nilai median dari piksel di sekitarnya sehingga efektif menghilangkan *salt and pepper noise* tanpa merusak detail tepi objek. Selain peningkatan kualitas visual, hasil pemrosesan citra juga dievaluasi menggunakan parameter kuantitatif seperti *Mean Squared Error* (MSE), *Peak Signal-to-Noise Ratio* (PSNR), dan *Structural Similarity Index Measure* (SSIM) untuk mengukur tingkat kemiripan antara citra

asli dan citra hasil perbaikan, sehingga metode yang digunakan dapat dinilai efektivitasnya dalam mendukung analisis dan aplikasi *computer vision* [12].

a. *Resize*

Resize merupakan proses dalam pengolahan citra digital yang bertujuan untuk mengubah dimensi citra, baik melalui pembesaran (*up-sampling*) maupun pengecilan (*down-sampling*). Proses ini termasuk dalam teknik *image interpolation*, yaitu metode yang digunakan untuk menentukan nilai piksel pada citra hasil dengan cara memetakan setiap piksel baru ke posisi tertentu pada citra asli dan menghitung nilainya berdasarkan piksel di sekitarnya. Salah satu pendekatan yang digunakan adalah metode *nearest neighbor*, dimana nilai piksel ditentukan berdasarkan pembulatan ke piksel terdekat. Hubungan pemetaan tersebut dapat dinyatakan sebagai berikut [15]:

$$x = x' \cdot \frac{W_{lama}}{W_{baru}}, y = y' \cdot \frac{H_{lama}}{H_{baru}} \quad \text{..... (1)}$$

Dimana :

x = hasil *mapping* koordinat horizontal ke gambar *input*

y = hasil *mapping* koordinat vertikal ke gambar *input*

Wlama = lebar gambar sebelum *resize*

Wbaru = lebar gambar setelah *resize*

Hlama = tinggi gambar sebelum *resize*

Hbaru = tinggi gambar setelah *resize*

nilai piksel pada citra hasil diperoleh dengan:

$$f(x', y') = f(round(x), round(y)) \quad \text{..... (2)}$$

Dimana :

f(x', y') = nilai pixel pada gambar hasil (*output*)

f(x, y) = nilai pixel pada gambar asli (*input*)

x' = koordinat horizontal pada gambar *output*

y' = koordinat vertikal pada gambar *output*

round() = pembulatan ke pixel terdekat

b. *Brightness Adjustment*

Peningkatan kecerahan (*brightness adjustment*) merupakan salah satu teknik dasar dalam pengolahan citra yang bertujuan untuk memperbaiki kualitas visual dengan membuat

gambar tampak lebih terang atau lebih gelap. Variasi kecerahan pada citra sering terjadi akibat kondisi lingkungan saat pengambilan gambar, seperti perbedaan waktu, intensitas cahaya, maupun kondisi cuaca [9]. Oleh karena itu, diperlukan suatu metode untuk menyesuaikan tingkat kecerahan agar informasi dalam citra dapat terlihat lebih jelas.

Secara matematis, proses peningkatan kecerahan dilakukan dengan menambahkan suatu konstanta terhadap setiap nilai piksel pada citra. Misalkan *f(y,x)* menyatakan nilai piksel pada koordinat *(y,x)* maka citra hasil peningkatan kecerahan dapat dinyatakan sebagai:

$$f'(y,x) = f(y,x) + \beta \quad \text{..... (3)}$$

Dimana :

$f'(y,x)$ adalah nilai piksel citra hasil

$f(y,x)$ = adalah nilai piksel citra asli

β = adalah konstanta penambah (*brightness offset*).

Nilai β berfungsi untuk mengatur tingkat kecerahan citra. Jika β bernilai positif, maka citra akan menjadi lebih terang. Sebaliknya, jika β bernilai negatif, maka citra akan menjadi lebih gelap. Proses ini dilakukan secara seragam terhadap seluruh piksel dalam citra, sehingga perubahan kecerahan terjadi secara *global* [14]. Dalam implementasinya, nilai piksel hasil harus tetap berada dalam rentang yang diperbolehkan (misalnya 0–255 untuk citra 8-bit). Oleh karena itu, diperlukan proses pembatasan (*clipping*) agar tidak terjadi *overflow* atau *underflow* nilai piksel. Melalui teknik ini, citra yang awalnya gelap dapat diperjelas, dan citra yang terang dapat diseimbangkan kembali.

c. *Global Thresholding optimal*

Ketika distribusi intensitas objek dan piksel latar belakang cukup berbeda, dimungkinkan untuk menggunakan ambang batas tunggal (*global*) yang berlaku untuk seluruh gambar. Dalam sebagian besar aplikasi, biasanya terdapat variabilitas yang cukup besar antar gambar sehingga, meskipun ambang batas *global* merupakan pendekatan yang sesuai, diperlukan algoritma yang mampu memperkirakan nilai ambang batas untuk setiap gambar [17].

Perhitungan nilai *global* thresholding optimal dinyatakan sebagai berikut:

$$T = \frac{1}{2}\,(m_1 + m_2) \quad \text{..... (4)}$$

$|T^{(t+1)} - T^{(t)}| < \Delta T$ ..... (5)

Dimana:

T = Nilai awal *Threshold* atau nilai rata-rata dari seluruh piksel dataset

m1 = Nilai rata-rata (*mean*) dari kelompok piksel pertama

m2 = Nilai rata-rata (*mean*) dari kelompok piksel kedua

ΔT = Nilai toleransi batas dengan rentang nilai 0 – 0,5

$T^{(t+1)}$ = nilai *Threshold* baru

$T^{(t)}$ = nilai *Threshold* lama

d. Perceptual Luminance

Perceptual luminance merupakan metode untuk mengukur tingkat kecerahan suatu citra berdasarkan persepsi visual manusia. Berbeda dengan rata-rata sederhana dari komponen warna merah (*Red*), hijau (*Green*), dan biru (*Blue*), metode ini memberikan bobot yang berbeda pada setiap kanal warna sesuai sensitivitas mata manusia terhadap cahaya. Mata manusia lebih sensitif terhadap warna hijau, diikuti warna merah, dan paling rendah terhadap warna biru [18]. Oleh karena itu, kontribusi setiap kanal warna terhadap kecerahan tidak sama.

Dalam pengolahan citra digital, *perceptual luminance* digunakan untuk mengonversi citra berwarna RGB menjadi representasi tingkat kecerahan (*luminance*) yang lebih sesuai dengan persepsi manusia. Nilai luminance ini sering digunakan pada tahap *preprocessing* karena mampu mempertahankan informasi penting mengenai intensitas cahaya pada citra sekaligus mengurangi kompleksitas data warna. Perhitungan nilai *perceptual luminance* dinyatakan sebagai berikut [18]:

$Y = INT [0.2126\ R + 0.7152\ G + 0.0722\ B$ ..... (6)

Dimana :

Y = nilai *luminance* (kecerahan),

R = komponen warna merah (*Red*),

G = komponen warna hijau (*Green*),

B = komponen warna biru (*Blue*),

INT = fungsi pembulatan ke bilangan bulat terdekat.

$\mu = \frac{\sum Y}{n}$ ..... (7)

μ = rata rata

y = total nilai elemen y

n = total elemen dari matrix y

2.2 *Deep Learning*

Deep Learning merupakan cabang dari *Machine Learning* yang menggunakan arsitektur *Artificial Neural Networks* (ANN) berlapis banyak (*multi-layer*) untuk memodelkan pola kompleks dalam data. Perkembangannya telah menjadi komponen inti dalam kemajuan *Artificial Intelligence* (AI) modern karena kemampuannya dalam mengolah data berskala besar dan melakukan komputasi kompleks secara efisien [19]. Berbeda dengan metode pembelajaran mesin tradisional yang memerlukan proses *feature extraction* secara manual, *deep learning* mampu melakukan ekstraksi fitur secara otomatis melalui representasi hierarkis pada setiap lapisan jaringan [20]. Struktur dasar *deep learning* terdiri atas *input layer*, beberapa *hidden layer*, dan *output layer*, dimana setiap lapisan menerapkan transformasi nonlinier sehingga model dapat mempelajari hubungan yang sangat kompleks.

Seiring perkembangannya, berbagai arsitektur *deep learning* telah dikembangkan sesuai dengan karakteristik data yang diproses. *Convolutional Neural Networks* (CNN) banyak digunakan dalam pengolahan citra dan terbukti efektif dalam tugas klasifikasi, deteksi objek, serta segmentasi . Untuk data sekuensial seperti teks dan suara, digunakan *Recurrent Neural Networks* (RNN) beserta variannya seperti *Long Short-Term Memory* (LSTM) dan *Gated Recurrent Unit* (GRU) yang mampu menangani ketergantungan jangka panjang [20]. Selain itu, inovasi arsitektur terbaru seperti *Transformers*, *Generative Adversarial Networks* (GAN), dan *Graph Neural Networks* (GNN) semakin memperluas kemampuan *deep learning* dalam bidang *Natural Language Processing* (NLP), visi komputer, dan pemodelan data berbasis graf.

Dalam bidang *computer vision*, *deep learning* telah menunjukkan performa yang sangat tinggi pada berbagai tugas seperti klasifikasi citra, pengenalan objek, dan segmentasi semantik. Namun demikian, model *deep learning* memiliki kerentanan terhadap gangguan kecil pada input yang dikenal sebagai *adversarial perturbations*, yang dapat menyebabkan kesalahan prediksi meskipun perubahan pada data tidak terlihat signifikan oleh manusia [21]. Oleh karena itu, penelitian terkini tidak hanya berfokus pada peningkatan akurasi, tetapi juga pada pengembangan *robust deep learning* untuk meningkatkan ketahanan model terhadap serangan *adversarial* dan gangguan lingkungan.

Kemajuan *deep learning* juga didukung oleh perkembangan perangkat keras seperti GPU dan TPU yang memungkinkan pelatihan model dalam skala besar serta mempercepat proses komputasi [20]. Selain itu, berbagai teknik pelatihan modern seperti *self-supervised learning*, *transfer learning*, dan *federated learning* telah dikembangkan untuk meningkatkan efisiensi dan adaptabilitas model dalam berbagai domain aplikasi [19]. Penerapan *deep learning* kini mencakup berbagai bidang seperti pengolahan bahasa alami, pengenalan suara, sistem rekomendasi, kendaraan otonom, serta layanan berbasis model bahasa seperti *ChatGPT*, yang memanfaatkan kombinasi pembelajaran terawasi dan penguatan untuk menghasilkan respons yang relevan.

2.3 *Convolutional Neural Network*

Convolutional Neural Network (CNN) merupakan salah satu metode *deep learning* yang secara khusus dirancang untuk memproses data berbentuk *grid* seperti citra digital. CNN termasuk dalam bagian dari *deep learning* yang merupakan cabang dari *machine learning* dan *Artificial Intelligence* (AI)[22]. Keunggulan utama CNN terletak pada kemampuannya dalam mengekstraksi fitur spasial secara otomatis tanpa memerlukan perancangan fitur manual seperti pada metode tradisional.

Sebelum berkembangnya CNN, klasifikasi citra umumnya mengandalkan ekstraksi fitur manual menggunakan metode seperti *Local Binary Pattern* (LBP), *Histogram of Oriented Gradients* (HOG), atau *Support Vector Machine* (SVM). Pendekatan tersebut memiliki keterbatasan dalam menangani data berskala besar dan kompleksitas variasi citra. CNN merevolusi bidang klasifikasi citra dengan kemampuan pembelajaran fitur secara hierarkis melalui proses konvolusi berlapis [23].

Dalam konteks klasifikasi ekspresi wajah, CNN terbukti efektif dalam mempelajari pola perubahan tekstur dan struktur wajah untuk mengidentifikasi emosi seperti *Happy, Sad, Angry, Fear, Disgust, Surprise* dan *Neutral* [24]. CNN mampu menangkap hubungan spasial antar piksel sehingga sangat sesuai untuk tugas pengenalan ekspresi wajah.

2.3.1 *Forward Propagation*

Forward propagation merupakan proses pada jaringan saraf tiruan dimana data *input* dialirkan melalui setiap lapisan jaringan secara berurutan hingga menghasilkan *output* akhir. *Output* yang diperoleh kemudian dibandingkan dengan nilai target menggunakan fungsi loss untuk mengukur tingkat kesalahan. Pada model *Convolutional Neural Network* (CNN), proses *forward pass* melibatkan serangkaian tahapan yang terjadi secara berurutan [25].

a. *Convolutional Layer*

Convolutional layer merupakan salah satu komponen inti pada *Convolutional Neural Network* (CNN) yang berfungsi untuk mengekstraksi fitur dari citra masukan. Pada *layer* ini, sejumlah *filter* atau *kernel* yang dapat dipelajari digeser pada *input* untuk melakukan operasi konvolusi, sehingga menghasilkan *output* berupa *feature map* [23]. Selain itu, *convolutional layer* memiliki keunggulan berupa penggunaan *parameter sharing* dan *sparse connectivity*, yang dapat mengurangi jumlah parameter yang dilatih, menurunkan kompleksitas model dan beban komputasi, serta meningkatkan kemampuan generalisasi sehingga dapat meminimalkan risiko *overfitting* [26]. Ukuran output *feature map* dipengaruhi oleh beberapa parameter penting, yaitu sebagai berikut:

1. *Filter size* berfungsi ukuran kernel yang digunakan untuk mengekstraksi fitur dari input.
2. *Stride* adalah jarak atau langkah pergeseran *filter* saat *kernel* digeser pada *input*.
3. *Zero padding* merupakan penambahan nilai nol pada sisi-sisi *input* untuk mengatur ukuran *output feature map*.

Pada *convolution layer*, citra masukan diproses menjadi objek berdasarkan fitur-fitur yang terkandung di dalamnya. Pada lapisan ini, matriks keluaran *H* diperoleh dari hasil perkalian antara matriks masukan beserta elemennya dengan matriks *filter*. Matriks *filter* dalam proses ini digunakan untuk mengekstraksi fitur dari setiap citra masukan. Matriks *filter* merupakan matriks yang berisi bobot-bobot bernilai kecil. Ukuran matriks *filter* pada setiap proses konvolusi tidak selalu sama, karena perbedaan ukuran tersebut bertujuan untuk memperoleh sampel fitur yang lebih beragam. Sebagai contoh, penggunaan *filter* berukuran 7×7 yang kemudian dilanjutkan dengan ukuran 3×3 menunjukkan bahwa proses konvolusi dapat mengenali pola secara lebih rinci dan spesifik. Selain itu, ukuran matriks *H* juga dipengaruhi oleh nilai langkah yang digunakan. *Padding* dan *stride* merupakan parameter yang digunakan untuk menentukan penambahan nilai nol serta pergeseran *filter* pada citra masukan. Ukuran matriks *H* dapat ditentukan menggunakan rumus berikut [27]:

$$O_H = \frac{w-f+2p}{s} + 1 \quad \text{.....(8)}$$

maka nilai *padding* (p) dapat dicari dengan mengubah persamaan tersebut menjadi:

$$p = \frac{s(O_H-1)-w+f}{2} \quad \text{.....(9)}$$

Dimana:

O = ukuran matriks H.

w = ukuran matriks *input*.

f = ukuran matriks *filter*.

p = *padding*.

s = *stride*.

Persamaan tersebut digunakan untuk menentukan besar *padding* yang diperlukan agar ukuran output sesuai dengan yang diinginkan.

b. *Batch Normalization*

Batch Normalization merupakan teknik normalisasi yang diterapkan pada setiap *mini-batch* selama proses pelatihan. Teknik ini digunakan untuk menjaga kestabilan data masukan sehingga proses pembelajaran pada *neural network* dapat berlangsung dengan lebih baik. Dalam penerapannya, *Batch Normalization* memanfaatkan nilai rata-rata (μ), variansi (σ^2), serta konstanta kecil (ϵ) untuk menjaga stabilitas numerik. Secara umum, *Batch Normalization* dilakukan dengan menghitung nilai rata-rata (*mean*) dan variansi (*variance*) dari setiap *batch*, kemudian menggunakan nilai tersebut untuk menormalkan data masukan, rumus *Batch Normalization* dapat ditulis sebagai berikut [28]:

$$x^{\wedge} = \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} \quad \text{.....(10)}$$

dengan rata-rata (*mean*) dan variansi (*variance*) yang dihitung sebagai berikut:

rata-rata (*mean*)

$$\mu = \frac{\sum x}{N} \quad \text{.....(11)}$$

variansi (*variance*)

$$\sigma^2 = \frac{\sum (x-\mu)^2}{N} \quad \text{.....(12)}$$

Dimana :

x^ = nilai hasil normalisasi

x = nilai *input* sebelum normalisasi

μ = rata-rata (mean) dari *batch*

σ2 = variansi dari *batch*

σ = *standard deviation*

ϵ = nilai konstanta kecil untuk mencegah pembagian dengan nol

N = Jumlah elemen *input*

c. *Addition Layer*

Salah satu komponen utama yang membedakan arsitektur ResNet-50 dengan model arsitektur lain pada *Convolutional Neural Network* adalah *Addition Layer* yang terdapat pada mekanisme *residual learning yang* melibatkan *shortcut connection*. *Addition Layer* merupakan lapisan yang melakukan operasi penjumlahan antara matriks *input* dengan matriks hasil transformasi dari beberapa operasi konvolusi sebelumnya. Proses tersebut memungkinkan informasi dari *input* tetap diteruskan secara langsung ke lapisan berikutnya, sehingga membantu menjaga aliran informasi selama pelatihan jaringan [29].

Secara umum, hubungan tersebut dinyatakan sebagai:

$$y = F(x) + x \quad \text{.....(13)}$$

Dimana:

y = matriks *output* yang dihasilkan

x = matriks *input*

F(x) = matriks hasil transformasi

d. ReLU

ReLU merupakan fungsi aktivasi yang memungkinkan model mempelajari pola *non-linier* yang lebih kompleks. ReLU menjadi salah satu fungsi aktivasi yang paling umum digunakan karena mampu mengatasi masalah *vanishing gradient* pada fungsi *sigmoid* dan *tanh* serta mempercepat konvergensi model. Pada *feature map*, ReLU mengubah nilai negatif menjadi 0 dan mempertahankan nilai positif tetap sama, sehingga *output* yang dihasilkan bersifat *non*-negatif. Persamaan fungsi ReLU dinyatakan sebagai berikut [30]:

$$f(x) = \begin{cases} x, & if\ x > 0 \\ 0, & if\ x \leq 0 \end{cases} \quad \text{.....(14)}$$

e. *Pooling layer*

Pooling layer berfungsi untuk mengubah ukuran citra masukan (*feature map*) yang bertujuan untuk mengurangi reduksi dimensi melalui *downsampling* pada *feature map* yang dihasilkan oleh *convolutional layer*. Terdapat dua jenis operasi yang digunakan pada *pooling layer*, yaitu *max pooling* dan *average pooling*. Matriks keluaran pada *layer* ini diperoleh berdasarkan ukuran *filter* dan *stride* yang telah ditentukan, sehingga didapatkan persamaan sebagai berikut [27]:

$f(x) = \max(0, x)$ .....(15)

Pada metode *max pooling*, proses perhitungan dilakukan dengan mengambil nilai terbesar pada baris dan kolom yang telah ditentukan berdasarkan ukuran *filter* dan *stride*. Sementara itu, pada *average pooling*, proses perhitungan dilakukan dengan mengambil nilai rata-rata dari elemen-elemen pada area tersebut [31].

$$Ok = \frac{1}{H \, x \, W} \sum_{i=i}^{H} \sum_{j=1}^{W} X_k(i,j) \quad \text{.....(16)}$$

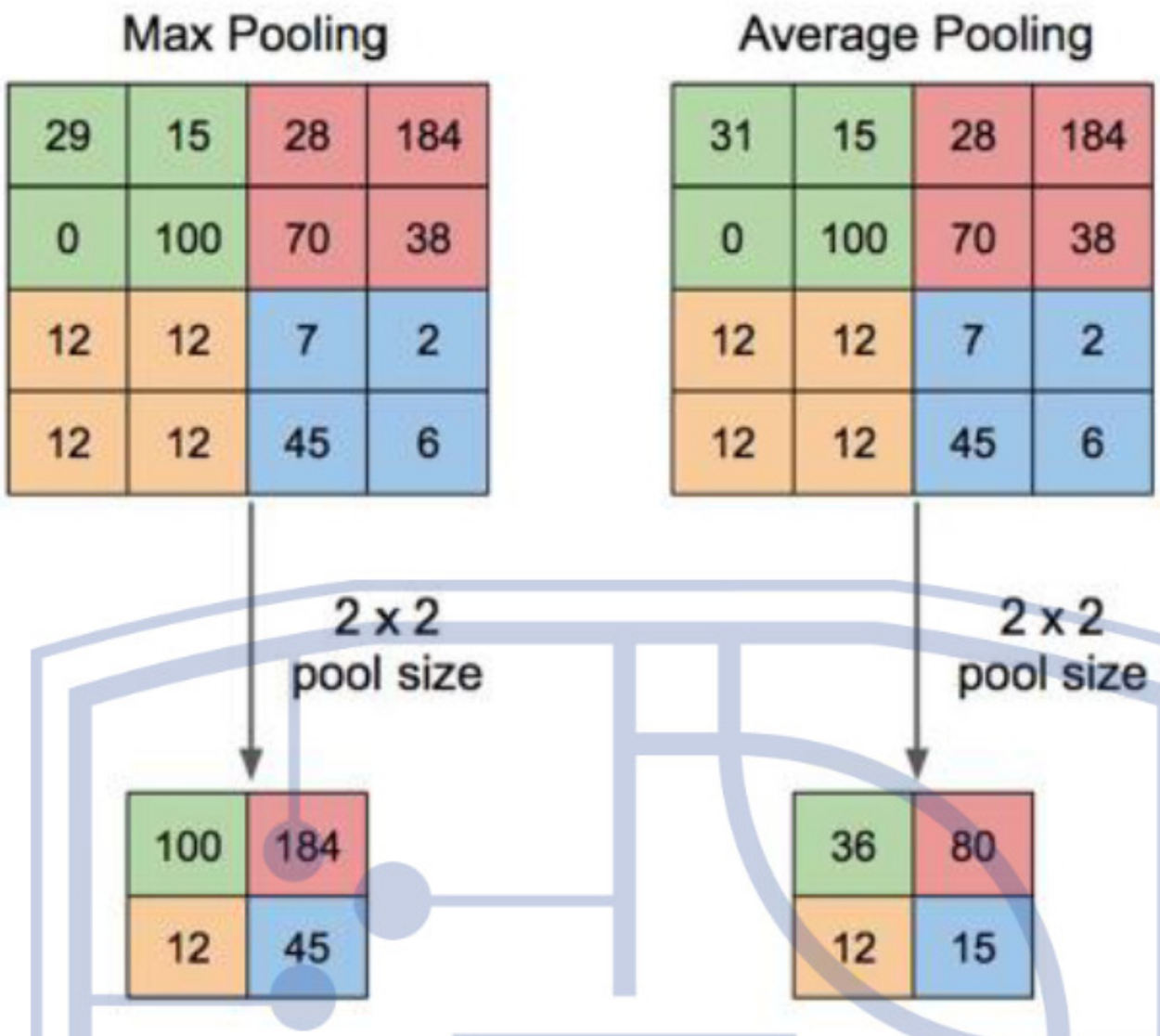
Dimana :

Ok = nilai *output* pada *feature map* ke-k

Xk(i,j) = nilai pada posisi i, j dari *feature map* ke-k

H = tinggi (*height*) *feature map*

W = lebar (*width*) *feature map*



Gambar 2. 3 Ilustrasi *Max Pooling* Dan *Average Pooling* [31]

f. Fully Connected Layer

Fully connected layer berfungsi untuk mentransformasikan hasil dari *layer* sebelumnya ke dalam bentuk vektor, kemudian mengklasifikasikan citra menggunakan fungsi aktivasi *softmax*. Hasil *output* dari tahap klasifikasi berupa nilai kelas dari citra masukan. Pada *fully connected layer* terdapat proses *flattening*, yaitu proses mengubah *output* dari *layer* sebelumnya menjadi bentuk satu dimensi atau vektor. Selanjutnya, vektor tersebut akan dihubungkan dengan jaringan saraf tiruan, berikut rumus yang didapatkan [27]:

$$Y_{(j)} = \sum_{i=1}^{n} w_{ij}\, x_i + b_j = 1,2,3,..,t \quad (17)$$

Dimana:

yj = nilai *output* pada *neuron* ke-j

xi = nilai *input* ke-i dari *layer* sebelumnya

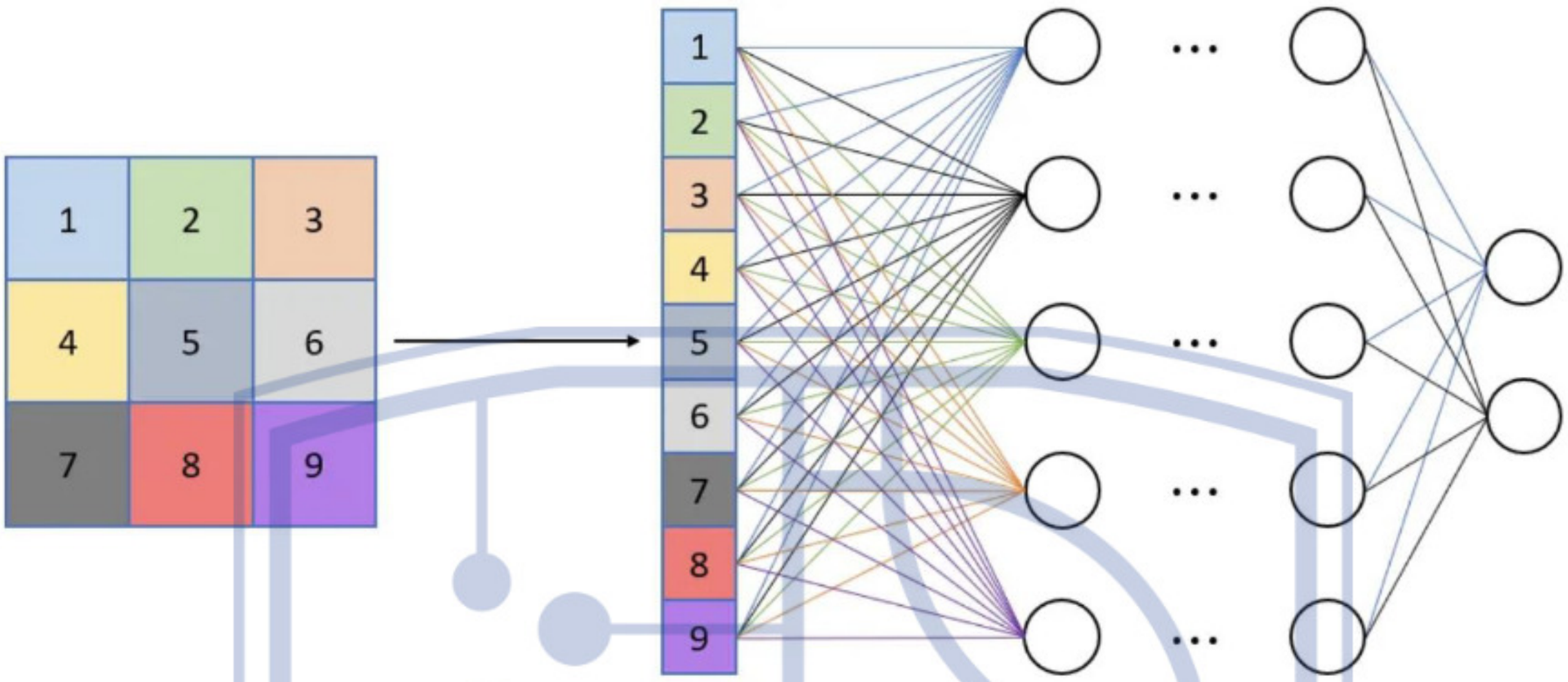
wij = bobot yang menghubungkan *input* ke-i dengan *neuron* ke-j

bj = bias pada *neuron* ke-j

n = jumlah input

Setiap nilai pada *pooled feature map* merepresentasikan sekumpulan fitur yang berbeda yang berasal dari bagian-bagian tertentu pada citra. Setelah itu, *pooled feature map* diubah

melalui proses *flattening* menjadi vektor kolom agar dapat dimasukkan ke dalam *fully connected network*.



Gambar 2. 4 Ilustrasi *Fully Connected Layer* [27]

g. *Softmax*

Softmax merupakan fungsi aktivasi yang digunakan untuk melakukan klasifikasi *linear* dengan mempertimbangkan nilai probabilitas pada setiap kelas. Nilai keluaran untuk masing-masing kelas berada pada rentang 0 hingga 1, dan jumlah seluruh nilai keluaran dari semua kelas sama dengan 1. Nilai probabilitas *softmax* dihitung menggunakan rumus berikut [27]:

$$softmax_j = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_i}}, j = 1,2,3,\dots,t \quad \text{.....(18)}$$

Dimana:

z = Nilai dari *fully connected layer*.

2.3.2 *Backpropagation*

Backpropagation merupakan metode dalam jaringan saraf tiruan yang digunakan untuk meminimalkan kesalahan (*error*) antara *output* prediksi dan nilai sebenarnya. Proses ini dilakukan dengan cara menyesuaikan parameter model, terutama bobot dan bias, melalui perhitungan *gradient* dari fungsi *loss*. Gradien tersebut dihitung menggunakan aturan rantai (*chain rule*) dan disebarkan kembali dari *output layer* ke *layer* sebelumnya. Dengan mekanisme ini, model dapat memperbaiki kesalahan secara bertahap pada setiap iterasi sehingga menghasilkan prediksi yang semakin akurat [32].

Metode *backpropagation* memiliki peran penting dalam pengembangan model *deep learning* karena mampu melakukan pembaruan parameter secara efisien serta dapat diterapkan pada jaringan dengan banyak lapisan dan struktur yang kompleks. Selain itu, proses pembelajaran menjadi lebih otomatis karena model dapat menyesuaikan parameternya sendiri untuk mengoptimalkan kinerja [32].

Dalam implementasinya, *backpropagation* melibatkan perhitungan turunan fungsi *loss*, baik untuk kasus klasifikasi maupun regresi, serta pembaruan bobot menggunakan metode optimasi seperti *Gradient Descent* hingga diperoleh nilai *error* yang minimum.

1. Fungsi *Loss Cross-Entropy*

   Fungsi *loss* yang digunakan untuk mengukur kesalahan prediksi model CNN dapat dirumuskan sebagai berikut :

   $L(\theta) = -\frac{1}{K}\sum_{i=1}^{K} y_i \, . log f_\theta(x_i)$ .....(19)

   Dimana:

   k = jumlah data pelatihan

   yi = label sebenarnya

   $f_\theta(x_i)$ = *output* model CNN

   θ = parameter model yang dioptimasi

   Rumus ini digunakan untuk mengukur perbedaan antara prediksi model dengan label sebenarnya sehingga nilai *loss* yang lebih kecil menunjukkan performa model yang lebih baik [33].

2. *Gradient* Parameter

   Gradient merupakan turunan dari fungsi *loss* terhadap parameter model yang menunjukkan arah dan besar perubahan *error*. Dalam proses *backpropagation, gradient* dihitung untuk setiap parameter dan digunakan dalam algoritma optimasi seperti *Gradient Descent* untuk memperbarui bobot dan bias agar nilai *loss* semakin kecil [33].

   $db = dz = P - y$ .....(20)

   $dW_t = dz.\ x^T$ .....(21)

   Dimana:

   dz = error pada *output layer*

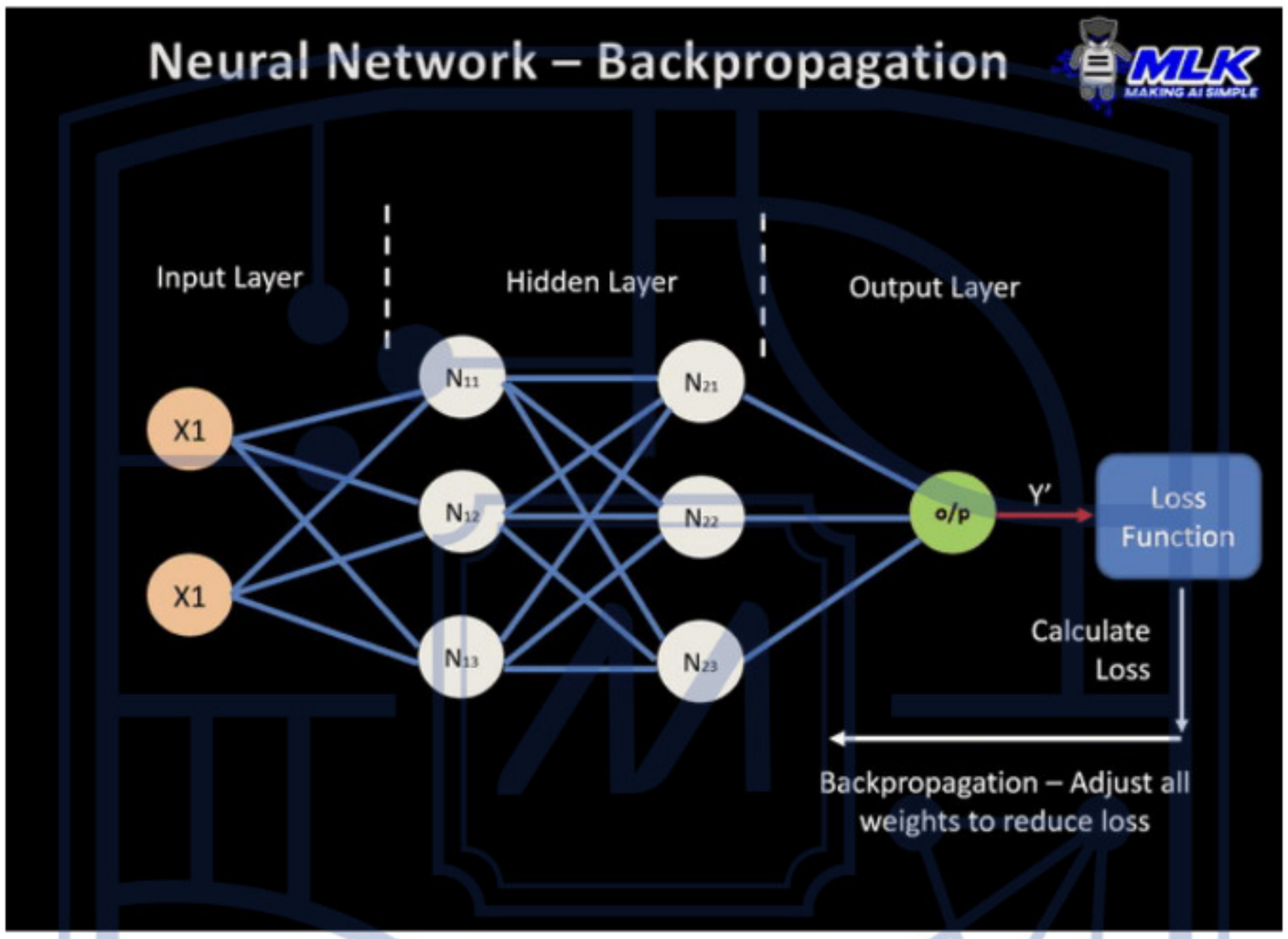
dw = *gradient* terhadap *weight*

db = *gradient* terhadap bias

x = *input*

P = probabilitas hasil dari *softmax*

y = label



Gambar 2. 5 Ilustrasi *Backpropagation*

2.3.3 *Hyperparameter*

Hyperparameter adalah parameter yang digunakan untuk mengontrol proses pelatihan dan perlu ditentukan oleh perancang model CNN. Dengan menyesuaikan *hyperparameter*, hasil akurasi prediksi dapat ditingkatkan [33].

Berikut beberapa *hyperparameter* yang bisa disesuaikan oleh perancang model CNN, yaitu sebagai berikut:

1. *Input Shape*

   Ukuran citra yang terlalu kecil dapat menyebabkan model CNN tidak dapat mempelajari fitur-fitur di dalam citra dengan baik. Oleh karena itu, ukuran citra yang digunakan sebagai *input* perlu diperhatikan. Ukuran citra yang disarankan adalah sebesar 224 x 224 [34].

2. *Epoch*

   Dalam pelatihan jaringan saraf tiruan, *epoch* merujuk pada jumlah siklus lengkap dimana seluruh dataset pelatihan melewati proses propagasi maju (*forward pass*) dan propagasi

balik (*backward pass*). Melatih model hanya dengan satu putaran data umumnya tidak memadai untuk mengekstraksi seluruh fitur yang ada, sehingga dataset perlu diproses secara berulang guna meningkatkan kemampuan generalisasi model. Meskipun demikian, penentuan jumlah *epoch* harus dilakukan secara cermat; iterasi yang berlebihan dapat memicu terjadinya *overfitting*, dimana model terlalu menghafal data latih namun gagal berkinerja baik pada data baru. Oleh karena itu, tidak ada angka standar untuk jumlah *epoch* yang ideal, melainkan harus diuji dan disesuaikan dengan karakteristik masing-masing dataset untuk mencapai keseimbangan model yang optimal [34], [35].

3. *Batch Size*

   Pelatihan (*training*) data keseluruhan pada dataset yang besar, terutama pada data citra, akan menjadi sangat berat secara komputasi. Oleh karena itu, penggunaan *batch size* dapat membantu memecah dataset menjadi bagian-bagian kecil untuk mempercepat proses pelatihan. Sebagai contoh, jika jumlah dataset adalah 3.200 dan menggunakan *batch size* 64, maka akan ada 50 iterasi yang dilakukan dalam satu putaran pelatihan (50 x 64 = 3.200). Terdapat beberapa nilai yang sering digunakan sebagai *batch size*, seperti 16, 32, 64, dan 128[34], [35] .

4. *Learning Rate*

   Parameter yang mengatur kecepatan model mempelajari masalah pada saat pelatihan (*training*). Jika *learning rate* terlalu rendah, model memerlukan waktu yang lama untuk mencapai titik optimal. Namun, jika terlalu tinggi, model mungkin melewatkan titik optimal tersebut, nilai *learning rate* yang umum untuk mencapai konvergensi yang stabil adalah 1×10^{-3} (1e-3) dan 1×10^{-4} (1e-4) [34], [36].

5. *Dropout*

   Dropout adalah teknik regularisasi yang digunakan untuk mencegah terjadinya *overfitting neural network* dimana beberapa *neuron* akan dipilih secara acak dan tidak dipakai selama *training* (dengan kata lain membuang fitur atau mengurangi variansi). *Neuron-neuron* ini dapat dibilang dibuang secara acak. Dengan cara ini, kontribusi dari *neuron* yang tidak digunakan tidak akan dihitung pada saat *forward pass* dan bobot baru juga tidak diterapkan pada saat melakukan backpropagation, nilai dropout yang dianjurkan untuk tugas umum adalah 0,5 [34], [37].

6. *Patience*

   parameter *patience* menentukan batas toleransi jumlah *epoch* berturut-turut tanpa adanya peningkatan performa pada data validasi sebelum proses pelatihan dihentikan secara otomatis, dimana nilai empiris yang umum dianjurkan adalah 5 [38].

7. *Weight decay*

Weight decay merupakan salah satu teknik regularisasi yang digunakan dalam proses pelatihan model *deep learning* untuk mengurangi risiko *overfitting* dengan membatasi besarnya nilai bobot (*weights*) pada model. Teknik ini membantu model menghasilkan parameter yang lebih stabil sehingga tidak terlalu menyesuaikan diri dengan data pelatihan dan memiliki kemampuan generalisasi yang lebih baik terhadap data baru. Pemilihan nilai *weight decay* perlu dilakukan secara tepat karena nilai yang terlalu kecil dapat menyebabkan *overfitting*, sedangkan nilai yang terlalu besar dapat mengakibatkan *underfitting*. Oleh karena itu, *weight decay* menjadi salah satu *hyperparameter* penting yang berperan dalam meningkatkan performa model selama proses pelatihan [34].

8. *Optimizer* Adam

Metode Adam (*Adaptive Moment Estimation*) memperbarui parameter model dengan memanfaatkan estimasi momen pertama dan kedua dari *gradient*. Rumus pembaruan parameter pada metode Adam adalah sebagai berikut [33]:

Mencari Momentum *weight*:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) dW, \quad (22)$$

Mencari Momentum bias:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) dB, \quad (23)$$

Mencari *Variance weight*:

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) dW^2, \quad (24)$$

Mencari *Variance* bias:

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) dB^2, \quad (25)$$

Mencari bias correction:

$$\hat{m} = \frac{m^t}{1 - \beta_1^t} \quad (26)$$

$$\hat{v} = \frac{v^t}{1 - \beta_2^t} \quad (27)$$

Mencari W_{baru}:

$$W_{t+1} = W_{t-1} - \alpha \frac{\hat{m}}{\sqrt{\hat{v} + \epsilon}} \quad (28)$$

Mencari b_{baru}:

$$b_{t+1} = b_{t-1} - \alpha \frac{\hat{m}}{\sqrt{\hat{v} + \epsilon}} \quad (29)$$

Dimana:

dw = *gradient* terhadap *weight*

db = *gradient* terhadap bias

mt = estimasi momen pertama (rata - rata *gradient* / momentum)

vt = estimasi momen kedua (rata - rata kuadrat *gradient* / *variance*)

β1 , β2 = parameter *decay rate*

α = *learning rate*

ϵ = konstanta kecil untuk menghindari pembagian dengan nol

Metode Adam dirancang untuk menggabungkan kelebihan metode *AdaGrad* dan *RMSProp*, sehingga mampu menyesuaikan *learning rate* secara adaptif pada setiap parameter serta meningkatkan efisiensi pelatihan model *deep learning* [33], [34].

2.4 ResNet

ResNet (*Residual Network*) diperkenalkan pada tahun 2015 untuk mengatasi permasalahan *degradation problem*, yaitu menurunnya akurasi ketika jaringan semakin dalam [29]. Permasalahan ini bukan disebabkan oleh *overfitting*, melainkan kesulitan optimasi akibat *vanishing gradient* pada jaringan yang sangat dalam. ResNet mengusulkan pendekatan *residual learning*, yaitu mempelajari fungsi *residual* dari pada langsung mempelajari fungsi pemetaan utama. Jika pemetaan yang diinginkan adalah *H(x)* maka ResNet memodelkannya sebagai *F(x) := H(x) - x*. Secara umum, *block residual* dinyatakan sebagai [29]:

$$y = f(x, \{w_i\}) + x \quad \text{.....(30)}$$

Dimana :

x = *input*

$f(x, \{w_i\})$ = *residual function*

y = *output block residual*

Jika dimensi tidak sama, digunakan proyeksi *linear* $y = f(x, \{w_i\}) + w_s x$. Pendekatan ini memungkinkan jaringan sangat dalam (hingga $\pm$152 layer) tetap stabil saat dilatih

2.4.1 Arsitektur ResNet-50

ResNet-50 merupakan arsitektur *deep convolutional neural network* dengan 50 *layer* yang mengadopsi konsep *residual learning* untuk mengatasi permasalahan *vanishing gradient* pada jaringan yang sangat dalam [29].

Tabel 2. 1 Arsitektur Resnet-50 [29]

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		[3×3, 64; 3×3, 64] ×2	[3×3, 64; 3×3, 64] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3
conv3_x	28×28	[3×3, 128; 3×3, 128] ×2	[3×3, 128; 3×3, 128] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×8
conv4_x	14×14	[3×3, 256; 3×3, 256] ×2	[3×3, 256; 3×3, 256] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×23	[1×1, 256; 3×3, 256; 1×1, 1024] ×36
conv5_x	7×7	[3×3, 512; 3×3, 512] ×2	[3×3, 512; 3×3, 512] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Arsitektur ini diawali dengan *layer* konvolusi 7×7 (64 *filter*, *stride* 2) dan *max pooling* 3 x 3 dengan *stride* 2, kemudian dilanjutkan dengan empat tahap *residual* utama yaitu Conv2_x, Conv3_x, Conv4_x, dan Conv5_x yang masing-masing terdiri dari 3, 4, 6, dan 3 blok *residual*. Setiap tahap melakukan ekstraksi fitur secara bertahap dengan pengurangan ukuran spasial (56×56 hingga 7×7) dan peningkatan jumlah channel (256 hingga 2048). Setelah setiap operasi konvolusi diterapkan *Batch Normalization* dan fungsi aktivasi ReLU. *Batch Normalization* dan fungsi aktivasi ReLU memastikan bahwa sinyal yang dipropagasikan ke depan memiliki variansi yang tidak nol serta menjaga *gradient* pada proses *backpropagation* tetap stabil, sehingga membantu mengurangi permasalahan *vanishing gradient* pada jaringan yang sangat dalam. Di akhir jaringan digunakan *Global Average Pooling* dan *Fully Connected layer* untuk proses klasifikasi. Keunggulan utama ResNet-50 terletak pada penggunaan *shortcut connection* dengan persamaan $y = F(x) + x$ yang memungkinkan aliran *gradient* tetap stabil sehingga jaringan dapat dilatih lebih dalam dan menghasilkan performa yang lebih baik dibanding CNN konvensional [29].

Tabel 2. 2 Struktur Blok Residual *(Bottleneck Block)* [29]

Jenis Konvolusi	Fungsi Sederhana
1×1 Conv	Mengurangi jumlah channel (kompres fitur)
3×3 Conv	Mengambil fitur utama (ekstraksi pola)
1×1 Conv	Mengembalikan jumlah channel
Shortcut Connection	Menambahkan input awal agar tidak terjadi *vanishing gradient*

Struktur utama dalam ResNet-50 adalah *bottleneck residual block* yang terdiri dari tiga lapisan konvolusi berurutan, yaitu 1×1, 3×3, dan 1×1. Konvolusi 1×1 pertama berfungsi untuk mengurangi jumlah *channel* (*feature compression*), konvolusi 3×3 melakukan ekstraksi pola spasial utama, dan konvolusi 1×1 terakhir mengembalikan jumlah *channel* ke dimensi semula (*feature expansion*). Input awal kemudian dijumlahkan dengan *output* blok melalui *shortcut connection* untuk membentuk fungsi *residual*. Desain *bottleneck* ini membuat ResNet-50 tetap efisien secara komputasi meskipun memiliki kedalaman jaringan yang besar, karena mampu menekan jumlah parameter tanpa mengurangi kemampuan ekstraksi fitur tingkat tinggi.

2.4.2 Pengenalan Ekspresi Wajah Dengan Metode ResNet-50

Metode ResNet-50 merupakan salah satu arsitektur *deep learning* berbasis *Convolutional Neural Network* (CNN) yang banyak digunakan dalam pengenalan ekspresi wajah karena kemampuannya dalam mengekstraksi fitur visual secara mendalam. ResNet-50 memanfaatkan konsep *residual learning* dengan mekanisme *skip connection* yang memungkinkan jaringan mempelajari fungsi *residual* dari suatu *input* sehingga dapat mengatasi permasalahan *vanishing gradient* pada jaringan yang sangat dalam. Dengan pendekatan ini, jaringan dapat memiliki banyak lapisan tanpa mengalami penurunan performa, sehingga mampu mengekstraksi pola ekspresi wajah secara lebih kompleks dan akurat. Proses pengenalan ekspresi wajah biasanya melibatkan tahap praproses citra, ekstraksi fitur menggunakan CNN, serta proses klasifikasi untuk menentukan kategori emosi seperti *Happy, Sad, Angry, Fear, Surprise, Disgust,* dan *Neutral* [39].

Dalam beberapa penelitian, ResNet-50 juga sering dikombinasikan dengan teknik *transfer learning* untuk meningkatkan performa model, terutama ketika jumlah dataset yang tersedia terbatas. *Transfer learning* memungkinkan model memanfaatkan pengetahuan dari jaringan yang telah dilatih sebelumnya pada dataset besar, kemudian menyesuaikannya untuk tugas pengenalan ekspresi wajah. Pendekatan ini terbukti mampu meningkatkan akurasi pengenalan ekspresi wajah secara signifikan karena model dapat menangkap fitur wajah yang lebih representatif serta lebih stabil terhadap variasi pencahayaan, sudut pandang, dan karakteristik wajah individu. Penggunaan ResNet-50 dengan *transfer learning* telah menunjukkan hasil yang sangat baik dalam klasifikasi ekspresi wajah dengan tingkat akurasi yang tinggi pada berbagai dataset emosi wajah [40], [41].

2.5 Ekspresi Wajah

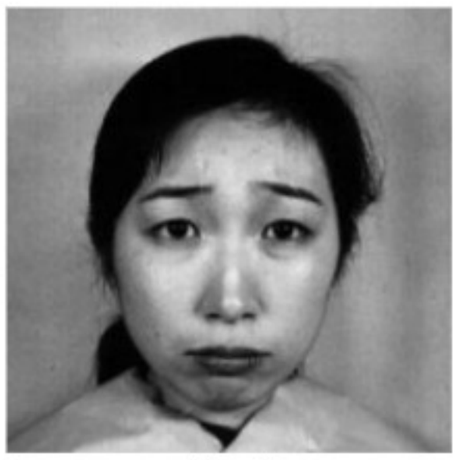
Dalam kehidupan sehari-hari, khususnya dalam komunikasi interpersonal, wajah digunakan sebagai media penyampaian pesan melalui ekspresi. Ekspresi wajah merupakan gerakan permukaan wajah yang terlihat dan dipengaruhi oleh aktivitas otot tertentu, sehingga dapat dirasakan dan diinterpretasikan oleh orang lain. Ekspresi wajah berfungsi sebagai sinyal sosial yang memungkinkan individu memahami perasaan dan niat orang lain serta mendukung terjalinnya interaksi sosial yang efektif. Selain itu, penelitian menunjukkan bahwa tidak seluruh bagian wajah memiliki kontribusi yang sama dalam mengenali emosi. Area mata diketahui memiliki peran penting dalam mengenali emosi takut (*Fear*), marah (*Anger*), dan sedih (*Sad*), sedangkan area mulut lebih dominan dalam mengenali emosi senang (*Happy*). Sementara itu, area hidung memiliki peran khusus dalam mengenali emosi jijik (*Disgust*) . Kesesuaian dan kesamaan emosional antara ekspresi wajah dan konteks penyajiannya juga terbukti dapat meningkatkan memori terhadap wajah maupun memori asosiatif antara wajah dan konteks, sehingga memperkuat proses pemahaman dan interaksi sosial. Temuan lain menunjukkan bahwa valensi emosional ekspresi wajah memodulasi proses berbagi emosi, dimana meskipun ekspresi bahagia dan marah sama-sama membangkitkan respons emosional, hanya ekspresi bahagia yang secara konsisten mendorong terjadinya berbagi emosi pada pengamat [1].

Ekspresi wajah tidak hanya merefleksikan emosi internal, tetapi juga berfungsi sebagai alat sosial yang mengkomunikasikan motivasi, niat, atau potensi tindakan yang tidak selalu berkaitan langsung dengan emosi. Dalam perspektif ekologi perilaku, ekspresivitas wajah dipandang sebagai strategi adaptif dalam interaksi sosial yang mempengaruhi bagaimana seseorang dipersepsikan oleh orang lain. Individu yang menampilkan ekspresi wajah secara jelas cenderung lebih mudah dipahami dan memiliki keuntungan dalam komunikasi serta hubungan interpersonal [42].

Dalam sistem pengenalan emosi otomatis, ekspresi wajah umumnya dikategorikan ke dalam beberapa kelas emosi dasar, yaitu *Happy, Sad, Angry, Fear, Disgust, Surprise, Neutral*. Proses pengenalan dilakukan dengan mempelajari pola visual wajah menggunakan pendekatan berbasis fitur maupun pembelajaran mendalam (*deep learning*) [43]. Namun, pengenalan ekspresi wajah masih menghadapi tantangan akibat kemiripan visual antar ekspresi serta pengaruh variasi pencahayaan, pose wajah, kualitas citra, dan perbedaan individu yang dapat menurunkan akurasi sistem.

Tabel 2. 3 Deskripsi Ekspresi Wajah [44]

Ekspresi	Keterangan
 *Angry*	Ekspresi marah biasanya ditunjukkan dengan mulut terbuka seolah olah berteriak dan mata menatap tajam ke arah lawan bicara.
	Angry expressions are usually shown with an open mouth as if shouting and eyes staring sharply at the person you are talking to.
 *Disgust*	Ekspresi seseorang digambarkan dengan sudut bibir yang ditarik kebawah dan mata yang menatap tajam ke objek yang menyebabkan rasa jijik (*Disgust*).
	A person expression is depicted with the corners of the lips pulled down and eyes staring intently at an object that causes disgust.
 *Fear*	Ekspresi seseorang ditandai dengan mata yang terlihat ketakutan dan ragu ragu saat menatap objek.
	A person expression is characterized by eyes that look scared and hesitant when looking at an object.
 *Happy*	Ekspresi seseorang yang diuji untuk klasifikasi ini adalah wajah yang menunjukkan senyuman.
	The expression of a person tested for this classification is a face showing a smile.
 *Neutral*	Ekspresi seseorang menunjukkan bahwa seseorang tidak merasakan emosi apapun, sehingga wajah yang ditampilkan memiliki ekspresi datar.
	A person expression shows that the person does not feel any emotion, so the face shown has a flat expression.

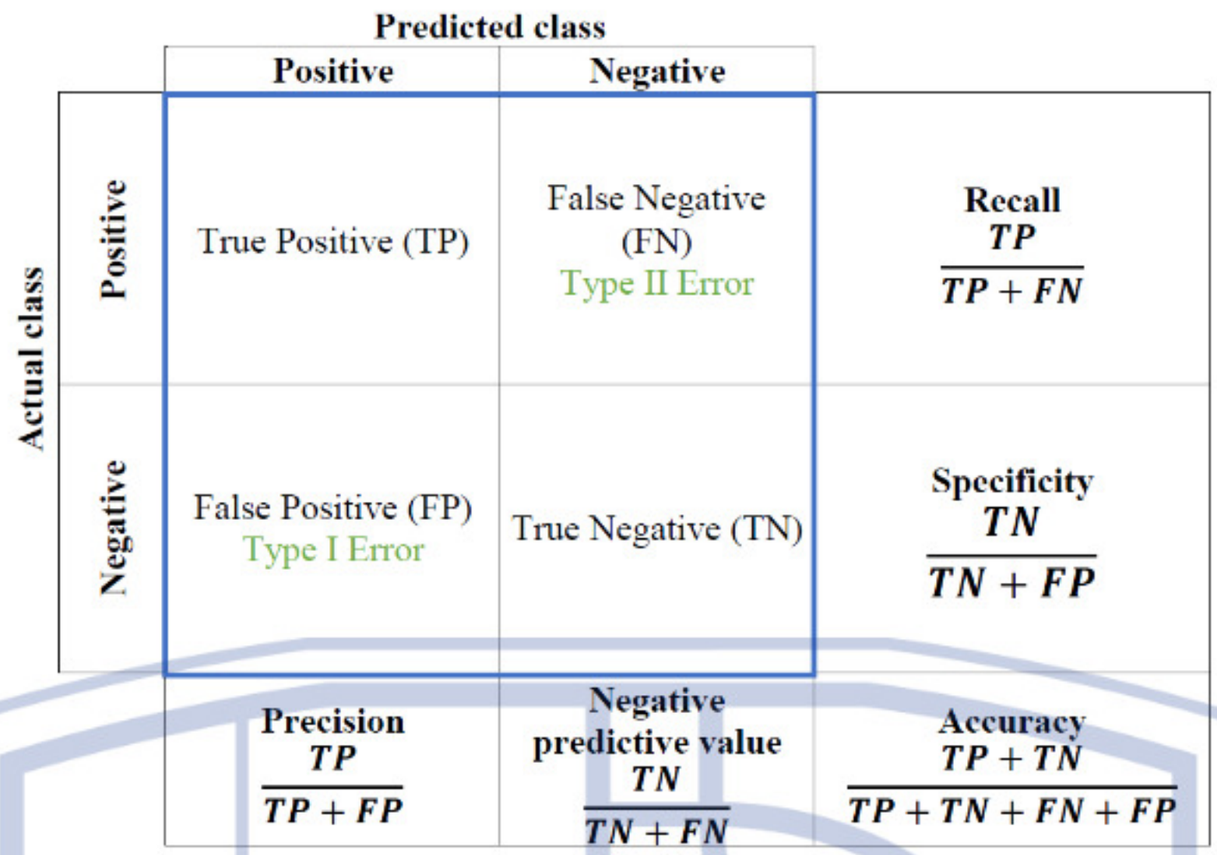
 *Sad*	Ekspresi seseorang ditunjukkan dengan mata yang tidak fokus menatap objek dan sudut bibir yang sedikit ditarik ke bawah. A person expression is shown by eyes that are not focused on an object and the corners of the lips are slightly pulled down.
 Surprise	Ekspresi seseorang yang ditunjukkan dengan tatapan mata tajam ke arah objek dan mulut yang menganga. A person expression is shown by a sharp gaze towards an object and an open mouth.

2.6 Metode Evaluasi

Metode evaluasi merupakan tahap untuk menilai kinerja model yang telah dibangun, dengan membandingkan hasil prediksi model terhadap data sebenarnya (*ground truth*). Evaluasi dilakukan untuk mengetahui seberapa baik model dalam melakukan klasifikasi, sekaligus mengidentifikasi kesalahan prediksi yang terjadi [45]. Hasil evaluasi dinyatakan dalam bentuk nilai metrik tertentu, sehingga performa model dapat diukur secara objektif dan dijadikan dasar untuk perbaikan maupun perbandingan dengan metode lain.

2.6.1 *Confusion Matrix*

Confusion Matrix merupakan metode evaluasi yang digunakan untuk mengukur kinerja model klasifikasi dengan membandingkan hasil prediksi terhadap nilai aktual. Matriks ini berbentuk tabel berukuran $N \times N$, dimana baris menunjukkan kelas prediksi dan kolom menunjukkan kelas aktual, sehingga dapat diketahui jumlah prediksi yang benar maupun salah. Metode ini umum digunakan pada *supervised learning* karena memerlukan label sebenarnya sebagai acuan evaluasi [45].



Gambar 2. 6 *Confusion Matrix* [22]

a. *True Positive* (TP): Kategori *true positive* pada *confusion matrix* menunjukkan jumlah data ketika model berhasil memprediksi kelas atau kejadian positif dengan benar dari seluruh data yang memang bernilai positif dalam dataset.
b. *True Negative* (TN): Pada *confusion matrix*, *true negative* menunjukkan jumlah data ketika model berhasil memprediksi kelas negatif dengan benar dari seluruh data yang memang bernilai negatif dalam dataset.
c. *False Positive* (FP): *False positive* menunjukkan kesalahan model ketika model memprediksi adanya suatu kondisi atau kejadian tertentu, padahal kondisi atau kejadian tersebut sebenarnya tidak ada. Jenis kesalahan ini juga disebut sebagai *type I error* dan dapat menyebabkan pengambilan keputusan yang keliru apabila tidak ditangani dengan baik.
d. *False Negative* (FN): *False negative* pada *confusion matrix* terjadi ketika model salah memprediksi bahwa suatu kejadian tidak ada, padahal sebenarnya kejadian tersebut ada dalam data positif. Kondisi ini menunjukkan kecenderungan model untuk melewatkan kasus positif atau melakukan *type II error*.

Pada klasifikasi biner, *confusion matrix* terdiri atas empat komponen utama yaitu *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN). TP dan TN menunjukkan prediksi yang benar, sedangkan FP (Type I error) dan FN (Type II error) menunjukkan kesalahan klasifikasi. Keempat nilai tersebut menjadi dasar dalam menghitung berbagai metrik evaluasi performa model [45], [46].

Berdasarkan nilai-nilai tersebut, dapat dihitung beberapa metrik evaluasi, yaitu *precision, recall, dan F1-score*. *Precision* digunakan untuk mengukur tingkat ketepatan prediksi positif oleh model, *recall* digunakan untuk mengukur kemampuan model dalam mendeteksi seluruh data positif, sedangkan *F1-score* merupakan rata-rata harmonis antara *precision* dan *recall*. Berikut rumus yang dapat digunakan untuk menghitung *precision, recall, dan F1-score* [22]:

$$Precision = \frac{TP}{TP+FP} \quad (31)$$

$$Recall = \frac{TP}{TP+FN} \quad (32)$$

$$F1 - score = 2\ x\ \frac{Precision\ x\ Recall}{Precision + Recall} \quad (33)$$

2.6.2 ROC dan AUC

ROC (*Receiver Operating Characteristic*) merupakan salah satu metode evaluasi yang digunakan untuk mengukur kinerja model klasifikasi dengan menggambarkan hubungan antara *True Positive Rate* (TPR) dan *False Positive Rate* (FPR). Kurva ROC digunakan untuk menunjukkan kemampuan model dalam membedakan kelas positif dan negatif, dimana model dengan performa yang baik akan menghasilkan kurva yang mendekati sudut kiri atas. Sementara itu, AUC (*Area Under The Curve*) merupakan nilai yang menyatakan luas area di bawah kurva ROC dan digunakan sebagai indikator kuantitatif dalam menilai performa model secara keseluruhan [47].

Secara matematis, ROC dan AUC didasarkan pada perhitungan TPR (*Recall*) dan FPR. Nilai TPR menunjukkan perbandingan antara jumlah prediksi positif yang benar terhadap seluruh data positif, sedangkan FPR menunjukkan perbandingan antara jumlah prediksi positif yang salah terhadap seluruh data negatif. Perhitungan TPR dan FPR dapat dinyatakan sebagai berikut [47]:

$$TPR = \frac{TP}{TP+FN} \quad (34)$$

$$FPR = \frac{FP}{FP+TN} \quad (35)$$

Nilai AUC diperoleh dari luas area di bawah kurva ROC yang dihitung menggunakan metode numerik, salah satunya adalah *trapezoidal rule*, dengan rumus sebagai berikut:

$$AUC = \sum \frac{(TPR_{i+1})}{2} \times (FPR_{i+1} - FPR_i) \quad (36)$$

Nilai AUC berada pada rentang 0 hingga 1, dimana:

a. AUC = 1 menunjukkan model sempurna
b. AUC = 0,5 menunjukkan model tidak lebih baik dari tebakan acak

Semakin tinggi nilai AUC, maka semakin baik kemampuan model dalam membedakan kelas positif dan negatif. Sebaliknya, nilai AUC yang mendekati 0,5 menunjukkan bahwa model tidak memiliki kemampuan diskriminatif yang baik atau setara dengan prediksi acak [45], [46]

2.7 Penanganan Ketidakseimbangan Data

Selain arsitektur model dan metrik evaluasi, karakteristik distribusi data juga memengaruhi kinerja model klasifikasi. Oleh karena itu, diperlukan strategi khusus untuk menangani kondisi data yang tidak seimbang antarkelas.

2.7.1 Ketidakseimbangan Kelas pada Klasifikasi Citra

Ketidakseimbangan kelas (*class imbalance*) merupakan kondisi ketika jumlah citra pada setiap kelas dalam suatu dataset tidak terdistribusi secara merata. Pada dataset pengenalan ekspresi wajah, kondisi ini umumnya membentuk distribusi ekor panjang (*long-tailed distribution*), yaitu sebagian kecil kelas menguasai mayoritas data, sedangkan sebagian besar kelas lainnya hanya memiliki sedikit sampel. Ketimpangan tersebut muncul karena sebagian ekspresi wajah secara alami lebih sering terjadi dibandingkan ekspresi lainnya [46]. Kondisi ini juga ditemukan pada dataset RAF-DB dan AffectNet yang dilaporkan memiliki tingkat ketidakseimbangan yang tinggi [45].

Ketidakseimbangan kelas menyebabkan model cenderung mengalami bias terhadap kelas mayoritas sehingga kemampuannya dalam mengenali kelas minoritas menurun [46]. Hal ini terbukti pada dataset RAF-DB, dimana tiga kelas dengan akurasi terendah merupakan tiga kelas dengan jumlah citra paling sedikit akibat tidak dilakukannya penyeimbangan data saat pelatihan [45]. Untuk menanganinya, pendekatan yang umum digunakan adalah pengambilan sampel ulang (*resampling*), yaitu mengurangi sampel kelas mayoritas (*undersampling*) atau menambah sampel kelas minoritas (*oversampling*) agar distribusi antarkelas menjadi lebih seimbang [46]. Meskipun demikian, pendekatan tersebut memiliki keterbatasan, sehingga diperlukan alternatif yang mampu menyeimbangkan proporsi kelas tanpa mengubah jumlah data secara fisik.

2.7.2 *WeightedRandomSampler*

WeightedRandomSampler merupakan teknik penanganan ketidakseimbangan kelas (*class imbalance*) yang bekerja pada tahap pembentukan *mini-batch* saat pelatihan, bukan dengan mengubah jumlah data secara fisik pada media penyimpanan. Teknik ini melakukan *oversampling* terhadap kelas minoritas secara probabilistik sehingga setiap kelas memiliki peluang yang relatif seimbang untuk muncul dalam satu *batch* pelatihan [48].

Prinsip kerjanya adalah pemberian bobot pengambilan sampel yang berbanding terbalik dengan frekuensi kemunculan kelas. Probabilitas p_i untuk mengambil sampel dari kelas i dirumuskan sebagai [48]:

$$p_i = \frac{1/f_i}{\sum_{j=1}^{k} 1/f_i} \quad \text{.....(37)}$$

Dengan f_i adalah frekuensi kemunculan kelas i dan k adalah jumlah kelas. Secara implementatif pada *framework* PyTorch, bobot kelas didefinisikan sebagai $w_k = 1/c_k$, dengan $c_k$adalah jumlah data kelas $k$pada himpunan pelatihan, kemudian setiap data ke-i diberi bobot sampling $s_i = w_{y_i}$. *Sampler* selanjutnya menarik indeks data dengan probabilitas proporsional terhadap s_i dan dilakukan *with replacement*, sehingga frekuensi harapan kemunculan kelas minoritas dalam *mini-batch* meningkat tanpa menduplikasi data secara fisik [49].

Pendekatan ini dipilih karena *undersampling* pada kelas mayoritas akan membuang banyak data dan mengurangi keragaman sinyal pelatihan, sementara *oversampling* fisik memperbesar ukuran dataset dan biaya komputasi. Perlu ditekankan bahwa pembobotan hanya diterapkan pada *data loader* tahap pelatihan, sedangkan *data loader* validasi dan pengujian dibentuk tanpa pembobotan agar hasil evaluasi tetap merefleksikan distribusi kelas yang alami [48]. Efektivitas teknik ini dibuktikan pada penelitian klasifikasi citra radiografi dada, dimana model *baseline* tanpa penyeimbangan menghasilkan sensitivitas nol karena model belajar memprediksi seluruh citra sebagai kelas mayoritas. Setelah *WeightedRandomSampler* diterapkan, model mulai mengenali kelas positif dengan tingkat sensitivitas mencapai 97–100% [49]. Dengan demikian, *WeightedRandomSampler* berperan penting dalam mencegah model terjebak pada prediksi kelas mayoritas dan meningkatkan *recall* pada kelas yang jumlah datanya sedikit