BAB 2
LANDASAN TEORI

Dalam deteksi penyakit kulit berbasis gambar, terdapat beberapa teori dan konsep dasar yang perlu dipahami, di antaranya adalah algoritma *Convolutional Neural Network* (CNN), arsitektur *ResNet152*, serta teori-teori pendukung lainnya.

2.1 Penyakit Kulit

Penyakit kulit merupakan gangguan pada bagian luar tubuh yang ditandai dengan gejala seperti gatal, nyeri, mati rasa, dan kemerahan. Penyebabnya dapat berasal dari paparan bahan kimia, sinar matahari, infeksi virus, kelemahan sistem imun, serta mikroorganisme seperti mikroba dan jamur, maupun faktor *personal hygiene* [20].

2.1.1 Eczema

Eczema atau *dermatitis atopik* (DA) merupakan penyakit kulit inflamasi kronis yang ditandai dengan gejala kulit kering, gatal, dan ruam merah yang muncul secara berulang. Penyakit ini tidak hanya memengaruhi kesehatan fisik, tetapi juga berdampak signifikan terhadap kualitas hidup pasien dan keluarganya [21]. Eczema biasanya mulai muncul dalam enam bulan pertama kehidupan dan bertahan hingga dewasa pada sekitar 25% penderitanya [22].

Gejala khasnya meliputi kulit yang sangat kering, rasa gatal hebat, dan ruam yang sering timbul di area lipatan kulit seperti siku, lutut, dan leher [7]. Faktor risiko utama termasuk riwayat keluarga dengan kondisi alergi seperti asma atau rinitis alergi, serta paparan terhadap faktor lingkungan seperti alergen, iritan, atau perubahan suhu yang ekstrem [2]. Contoh visual eczema ada pada gambar 2.1.

Gambar 2.1. Contoh gambar penyakit eczema

2.1.2 Psoriasis

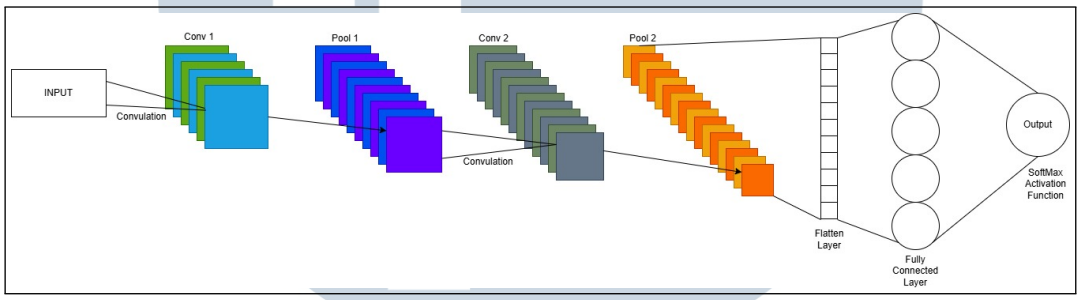
Psoriasis adalah penyakit autoimun kronis yang ditandai dengan percepatan siklus pergantian sel kulit, yang menyebabkan penumpukan sel-sel kulit di permukaan dan membentuk bercak merah, tebal, serta bersisik [23]. Gejala utamanya meliputi plak merah yang tertutup sisik keperakan, disertai rasa gatal, nyeri, dan peradangan, yang paling sering muncul di kulit kepala, siku, lutut, dan punggung bawah [8].

Psoriasis memiliki prevalensi global yang bervariasi dari 0,09% hingga 11,43%, dan memengaruhi sekitar 100 juta orang di seluruh dunia[12]. Pasien umumnya memerlukan pengobatan jangka panjang karena gejala dapat membaik atau memburuk secara periodik, tergantung pada berbagai faktor seperti gangguan sistem imun, infeksi, stres, konsumsi alkohol, dan kebiasaan merokok [24]. Contoh visual psoriasis ada pada gambar 2.2.

Gambar 2.2. Contoh gambar penyakit psoriasis

2.2 Convolutional Neural Network (CNN)

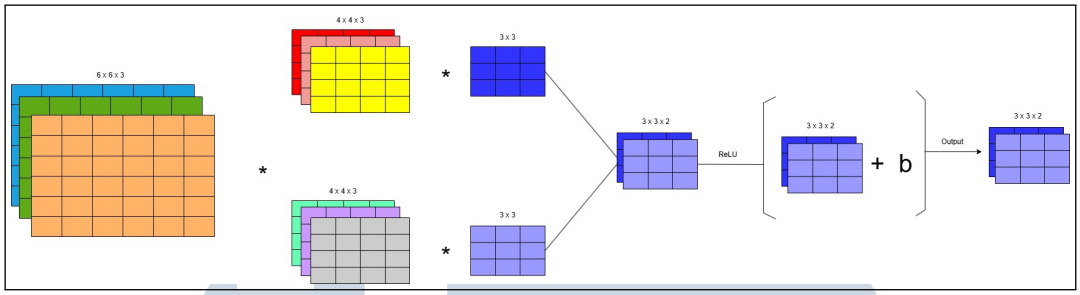
Convolutional Neural Network (CNN) adalah salah satu jenis jaringan saraf tiruan *deep feedforward* yang banyak diterapkan dalam bidang *computer vision*. Metode ini juga dikenal dengan istilah *ConvNet*. Mirip dengan jaringan saraf tiruan lainnya, CNN memiliki arsitektur yang terdiri dari node atau neuron yang saling terhubung satu sama lain dalam sebuah lapisan [25]. Lapisan-lapisan tersebut terdiri dari *hidden layer* yang mencakup *convolutional layers*, *pooling layers*, *fully connected layers*, dan *activation function*. Secara umum, proses kerja CNN dapat dilihat pada Gambar 2.3.



Gambar 2.3. Proses operasi pada arsitektur CNN

2.2.1 Convolutional Layer

Convolutional layer adalah lapisan yang terdiri dari beberapa filter konvolusi atau kernel yang digunakan untuk menghasilkan peta fitur keluaran. Masukan berupa gambar yang dimasukkan ke dalam jaringan akan diproses pada tahap ekstraksi fitur. Gambar akan dibagi menjadi beberapa bagian sesuai dengan piksel yang ditentukan oleh parameter tertentu [26]. Proses konvolusi ini akan menghasilkan gambar dengan ukuran yang lebih kecil atau tetap, namun dengan tingkat kedalaman (*depth*) yang berbeda tergantung pada jumlah filter yang digunakan [27]. Visualisasi proses convolutional layer ada pada gambar 2.4



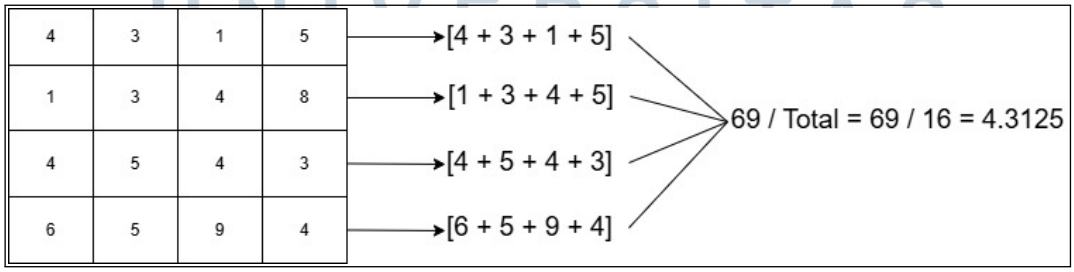
Gambar 2.4. Proses operasi pada *Convolutional Layer*

2.2.2 Pooling Layer

Pooling adalah operasi yang digunakan dalam CNN untuk mengurangi dimensi data sambil tetap mempertahankan informasi penting. Tujuan dari operasi ini adalah untuk mereduksi ukuran representasi fitur dengan menggabungkan informasi dari area-area piksel yang berdekatan [28]. Terdapat beberapa jenis metode *pooling* yang dapat digunakan dalam berbagai lapisan, antara lain: *tree pooling*, *gated pooling*, *average pooling*, *min pooling*, *max pooling*, *global average pooling* (GAP), dan *global max pooling* [29].

A Global Average Pooling

Global Average Pooling (GAP) merupakan salah satu jenis lapisan *pooling* yang menggunakan fungsi rata-rata. Lapisan ini mengambil nilai rata-rata dari seluruh hasil *feature map*, yang berdampak pada pengurangan jumlah parameter keluaran. *Global Average Pooling* memiliki kemiripan dengan struktur konvolusi karena melakukan operasi korespondensi antara *feature maps* dan kategori, sehingga memudahkan proses klasifikasi dan juga dapat mengurangi beban komputasi [30]. Pada gambar 2.5 merupakan cara kerja *global average pooling*.



Gambar 2.5. Proses *Global Average Pooling*

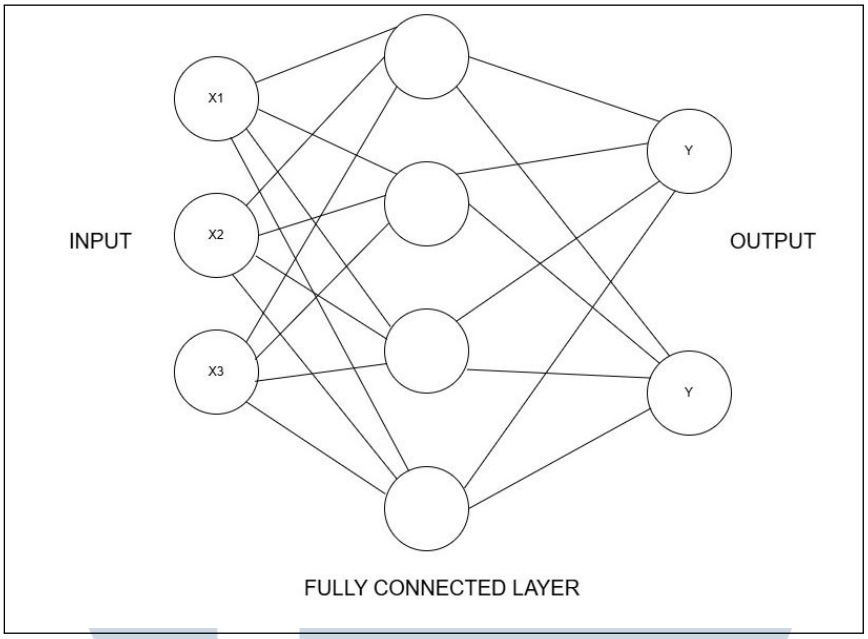
2.2.3 Flattened Layer

Feature map yang dihasilkan dari beberapa lapisan konvolusi dan *pooling* masih berbentuk *multidimensional array*, sehingga perlu dilakukan proses *flatten* atau *reshape* untuk mengubah *feature map* tersebut menjadi sebuah vektor agar dapat digunakan sebagai masukan ke dalam *fully-connected layer* [31]. Visualisasi proses flattening ada pada gambar 2.6.

Gambar 2.6. Proses flattening sebelum masuk ke *fully-connected layer*

2.2.4 Fully Connected Layer

Fully-connected layer adalah lapisan di mana semua neuron dari lapisan sebelumnya terhubung dengan semua neuron di lapisan berikutnya, mirip dengan arsitektur jaringan saraf tiruan tradisional [31]. *Fully-connected layer* berfungsi untuk menggabungkan fitur-fitur yang telah diekstraksi dari proses sebelumnya guna menentukan kelas atau kategori dari masukan yang diberikan [32]. Visualisasi proses fully connected layer ada pada gambar 2.7.



Gambar 2.7. Proses *Fully Connected Layer* pada arsitektur

2.2.5 Activation Function

Fungsi aktivasi dalam jaringan saraf tiruan merupakan suatu persamaan matematis yang digunakan untuk menentukan output dari setiap neuron. Efisiensi komputasi dari fungsi aktivasi sangat penting karena fungsi ini harus dihitung pada banyak neuron untuk setiap data yang diproses. Selain itu, fungsi aktivasi juga berperan dalam menormalkan output neuron agar berada dalam rentang tertentu, seperti antara 0 dan 1 atau antara -1 dan 1 [33]. Terdapat berbagai jenis fungsi aktivasi yaitu:

A ReLU

Rectified Linear Unit (ReLU) merupakan fungsi aktivasi yang umum digunakan dalam jaringan saraf karena kesederhanaan dalam perhitungannya. Fungsi ini bekerja dengan mengatur nilai output menjadi nol apabila input bernilai negatif, dan membiarkan input tetap apabila bernilai positif. Keunggulan ReLU terletak pada efisiensi komputasi karena tidak melibatkan operasi kompleks seperti eksponensial, melainkan hanya keputusan kondisi sederhana. Pendekatan ini mempercepat proses pelatihan jaringan baik dalam proses *forward* maupun *backward* propagation [26].

B Sigmoid

Sigmoid merupakan salah satu fungsi aktivasi non-linear yang digunakan dalam jaringan saraf tiruan. Fungsi ini menerima input berupa satu nilai numerik dan mengubahnya menjadi nilai dalam rentang antara 0 hingga 1 [34]. Karakteristik ini membuat sigmoid cocok untuk tugas klasifikasi biner, karena dapat menginterpretasikan output sebagai probabilitas. Namun, kelemahan dari fungsi ini adalah adanya potensi *vanishing gradient* ketika nilai input sangat besar atau sangat kecil, yang dapat memperlambat proses pelatihan model.

B.1 Binary Cross Entropy

Binary cross-entropy adalah bentuk khusus dari fungsi *cross-entropy* yang digunakan dalam tugas klasifikasi biner, yaitu ketika target label bernilai 0 atau 1 [35]. Fungsi ini bekerja dengan mengukur jarak antara output prediksi model dan label sebenarnya. Umumnya, *binary cross-entropy* digunakan bersama dengan fungsi aktivasi *sigmoid* pada layer output untuk menghasilkan nilai prediksi dalam bentuk probabilitas. Semakin dekat hasil prediksi terhadap label yang benar, semakin kecil nilai loss-nya, sehingga fungsi ini sangat efektif dalam melatih model klasifikasi biner.

2.3 ResNet152

Deep Residual Network atau yang lebih dikenal sebagai *ResNet* merupakan salah satu arsitektur *Convolutional Neural Network* (CNN) yang diperkenalkan oleh He et al. pada tahun 2016. Arsitektur ini dirancang untuk mengatasi masalah penurunan performa pada jaringan yang sangat dalam, yang sering mengalami kesulitan dalam proses pelatihan karena fenomena *vanishing gradient*. Solusi yang ditawarkan oleh ResNet adalah dengan menerapkan *skip connection* atau jalur pintas, yang memungkinkan sebagian input dilewatkan secara langsung ke lapisan yang lebih dalam tanpa mengalami transformasi. Keunggulan utama dari ResNet dibandingkan arsitektur CNN konvensional lainnya adalah kemampuannya mempertahankan akurasi bahkan ketika jumlah lapisan semakin bertambah. Beberapa lapisan dalam arsitektur ini berfungsi untuk menyederhanakan representasi visual dari gambar masukan sambil mempertahankan informasi penting [36].

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				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	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

Gambar 2.8. Arsitektur ResNet

Sumber:[37]

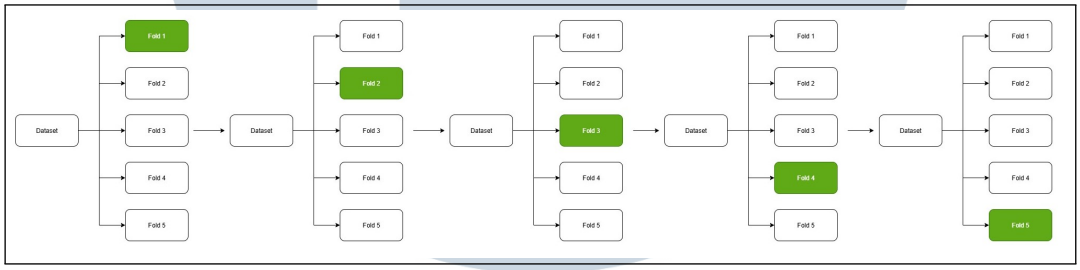
Salah satu varian *ResNet* yang populer adalah ResNet152, yang memiliki 152 lapisan dan dirancang untuk menangani *dataset* kompleks dengan memberikan performa tinggi tanpa mengorbankan efisiensi pelatihan. Dalam implementasinya, *ResNet* sering dimodifikasi dengan menambahkan lapisan seperti *reshape*, *flattened*, *dense*, *dropout*, dan *SoftMax* agar sesuai dengan tugas spesifik, seperti klasifikasi gambar[38].

Arsitektur *ResNet152* terdiri dari beberapa lapisan yang tersusun secara hierarkis. Komposisi awal dimulai dengan lapisan **conv1** yang berisi operasi *convolution* berukuran 7×7 dan menghasilkan output berdimensi 112×112. Selanjutnya, lapisan **conv2** terdiri dari *max pooling* berukuran 3×3, diikuti dengan urutan *batch normalization* 1×1, fungsi aktivasi *ReLU* 1×1, dan *convolution* 3×3. Rangkaian ini diulang sebanyak 3 kali sehingga menghasilkan output berdimensi 56×56.

Kemudian, pada lapisan **conv3**, terdapat komposisi *batch normalization* 1×1, *ReLU* 1×1, dan *convolution* 3×3 yang diulang sebanyak 8 kali untuk menghasilkan output berdimensi 28×28. Lapisan berikutnya, yaitu **conv4**, terdiri dari struktur yang sama namun diulang sebanyak 36 kali dan menghasilkan output berdimensi 14×14. Selanjutnya, lapisan **conv5** juga memiliki struktur serupa yang diulang sebanyak 3 kali dan menghasilkan output berdimensi 7×7. Terakhir, output diproses melalui lapisan *average pooling* dan *softmax* yang menghasilkan output akhir berdimensi 1×1. Seluruh arsitektur ini digambarkan secara ilustratif pada Gambar 2.8.

2.4 K-Fold Cross Validation

K-Fold Cross Validation merupakan teknik validasi di mana kumpulan data dibagi menjadi K bagian (*fold*) yang berukuran sama. Setiap fold secara bergantian digunakan sebagai data pengujian, sementara fold lainnya digunakan untuk melatih model. Misalnya, pada skenario validasi silang 5-fold ($K = 5$) yang di ilustrasikan pada gambar 2.9. Dataset dibagi menjadi lima subset, Pada iterasi pertama, fold ke-1 digunakan sebagai data uji dan empat fold lainnya sebagai data latih. Pada iterasi kedua, fold ke-2 berperan sebagai data uji dan empat fold lainnya sebagai data latih. Proses ini diulang hingga seluruh fold pernah digunakan sebagai data uji tepat satu kali. Pendekatan ini bertujuan untuk memperoleh evaluasi model yang lebih umum dan tidak bergantung pada satu pembagian data tertentu [39].



Gambar 2.9. Proses 5 *Fold Cross Validation*

2.5 Augmentasi Data

Data Augmentation ialah proses memperkaya data pelatihan yang bertujuan untuk menghindari munculnya overfitting[40]. Proses data augmentation terdiri dari beberapa metode yaitu flip, rotate, brightness.

1. Flip bekerja untuk meningkatkan jumlah data pelatihan dengan cara memutar gambar secara horizontal[41].
2. Rotate melakukan rotasi pada gambar dengan besaran sudut[41].
3. Brightness mengubah pencahayaan brightness atau kecerahan pada gambar dengan besaran skala[41].

2.6 Optimizer

Optimizer merupakan algoritme yang digunakan untuk memperbarui bobot dan bias dalam proses pembelajaran jaringan saraf tiruan, dengan tujuan untuk

meminimalkan *error* atau selisih antara keluaran jaringan dan target yang diinginkan. Beberapa di antaranya adalah *Adam*, dan *Nadam*. Algoritme-algoritme ini umumnya menggabungkan pendekatan adaptif terhadap laju pembelajaran dan momentum, sehingga dapat mempercepat proses pelatihan dan meningkatkan performa model[42].

2.6.1 Adam

Adam optimizer adalah algoritma optimasi yang sering digunakan dalam *deep learning*. Adam optimizer memperbarui bobot dan learning rate dapat berubah–ubah selama proses training. Adam optimizer menggabungkan kelebihan dari *Momentum* dan *RMSProp optimizer* yang dapat memecahkan masalah gradien renggang untuk mencapai kinerja yang baik. Adam optimizer menggunakan rata-rata momen gradien pertama dan kedua untuk menyesuaikan parameter *learning rate*[43].

2.6.2 Nadam

Nadam (*Nesterov-accelerated Adaptive Moment Estimation*) merupakan modifikasi dari algoritma Adam yang menggabungkan komponen momentum dari Adam dengan pendekatan *Nesterov accelerated gradient*. Tujuan utama dari algoritma ini adalah untuk meningkatkan kecepatan konvergensi dan kualitas model yang dipelajari [44].

2.7 Evaluasi Model

Evaluasi dan pembahasan dilakukan dengan menerapkan perbandingan untuk tiap model skenario yang sudah diuji. Perbandingan diterapkan pada nilai *accuracy*, *precision*, *recall*, dan *f1-score* pada hasil prediksi dalam proses pengujian yang dihitung menggunakan *confusion matrix* seperti yang ditunjukan pada Tabel 2.1. *True Positive* (TP) menunjukan banyak data aktual dan prediksi yang sama-sama bernilai positif, dan sebaliknya *True Negative* (TN) menunjukan banyak data aktual dan prediksi yang sama-sama bernilai negatif. *False Positive* (FP) menunjukan banyak data aktual yang negatif namun diprediksi positif, sebaliknya *False Negative* (FN) menunjukan data aktual yang bernilai positif namun diprediksi bernilai negatif [45].

Tabel 2.1. Confusion Matrix

		Predict	
		Positive	**Negative**
Actual	**Positive**	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

Precision (P) adalah rasio antara jumlah data positif yang diprediksi dengan benar terhadap seluruh data yang diprediksi positif [46], seperti yang ditunjukkan pada Persamaan 2.1.

$$Precision = \frac{TP}{TP + FP} \tag{2.1}$$

Recall (R) adalah rasio antara jumlah data positif yang diprediksi dengan benar terhadap seluruh data aktual yang positif [46], seperti yang ditunjukkan pada Persamaan 2.2.

$$Recall = \frac{TP}{TP + FN} \tag{2.2}$$

F1 Score, seperti yang ditunjukkan pada Persamaan 2.3, merupakan rata-rata harmonis dari Precision dan Recall, dan digunakan untuk mengukur performa model [46].

$$F1\ Score = \frac{2 \cdot (Precision \cdot Recall)}{Precision + Recall} \tag{2.3}$$

Persentase data yang berhasil diklasifikasikan dengan benar disebut **akurasi**, yang ditunjukkan pada Persamaan 2.4. Akurasi merupakan ukuran performa yang paling intuitif, yaitu rasio antara jumlah prediksi yang benar terhadap total seluruh data [46].

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.4}$$