

**KLASIFIKASI DAN DETEKSI PENYAKIT KULIT MENGGUNAKAN  
METODE EFFICIENTNETB7 DAN YOLOV8 BERBASIS *WEBSITE***

**SKRIPSI**

**ALEX MARIO SIMANJUNTAK**

**201401034**



**PROGRAM STUDI S-1 ILMU KOMPUTER**

**FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI**

**UNIVERSITAS SUMATERA UTARA**

**2024**

**KLASIFIKASI DAN DETEKSI PENYAKIT KULIT MENGGUNAKAN  
METODE EFFICIENTNETB7 DAN YOLOV8 BERBASIS *WEBSITE***

**SKRIPSI**

Diajukan untuk melengkapi tugas dan memenuhi syarat memperoleh ijazah Sarjana  
Komputer

**ALEX MARIO SIMANJUNTAK**

**201401034**



**PROGRAM STUDI S-1 ILMU KOMPUTER**

**FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI**

**UNIVERSITAS SUMATERA UTARA**

**2024**

**PERSETUJUAN**

Judul : KLASIFIKASI DAN DETEKSI PENYAKIT KULIT  
MENGUNAKAN METODE EFFICIENTNETB7  
DAN YOLOV8 BERBASIS WEBSITE

Kategori : SKRIPSI

Nama : ALEX MARIO SIMANJUNTAK

Nomor Induk Mahasiswa : 201401034

Program Studi : SARJANA (S-1) ILMU KOMPUTER

Fakultas : ILMU KOMPUTER DAN TEKNOLOGI  
INFORMASI UNIVERSITAS SUMATERA  
UTARA

Telah diuji dan dinyatakan lulus di Medan, 15 Oktober 2024.

Dosen Pembimbing II

Handrizal S.Si., M.Comp.Sc

NIP. 197706132017061001

Dosen Pembimbing I

Dr. T. Henny Febriana Harumy S.Kom., M.Kom

NIP. 198802192019032016



Diketahui/Disetujui Oleh  
Ketua Program Studi S-1 Ilmu Komputer

Dr. Amalia, S.T., M.T

NIP. 197812212014042001

## **PERNYATAAN**

**KLASIFIKASI DAN DETEKSI PENYAKIT KULIT MENGGUNAKAN METODE  
EFFICIENTNETB7 DAN YOLOV8 BERBASIS *WEBSITE***

## **SKRIPSI**

Saya menyatakan bahwa skripsi ini adalah hasil penelitian saya sendiri, kecuali untuk kutipan dan ringkasan yang telah diberikan sumbernya..

Medan, 18 September 2024



Alex Mario Simanjuntak

NIM. 201401034

## PENGHARGAAN

Seluruh pujian dan syukur penulis persembahkan kepada Tuhan Yang Maha Esa atas segala rahmat, anugerah, dan petunjuk-Nya yang telah memberikan kekuatan dan kesehatan kepada penulis untuk menyelesaikan skripsi ini dengan baik. Tanpa bimbingan dan perlindungan-Nya, penulis tidak akan dapat menyelesaikan tugas akhir ini sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer di Program Studi S-1 Ilmu Komputer, Universitas Sumatera Utara.

Penulis juga mengungkapkan rasa terima kasih yang sebesar-besarnya kepada:

1. Bapak Dr. Muryanto Amin, S.Sos, M.Si. sebagai Rektor Universitas Sumatera Utara.
2. Ibu Dr. Maya Silvi Lydia, B.Sc., M.Sc. sebagai Dekan Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara.
3. Ibu Dr. Amalia ST., M.T. sebagai Ketua Program Studi S1 Ilmu Komputer Universitas Sumatera Utara.
4. Ibu Sri Melvani Hardi S.Kom., M.Kom sebagai Sekretaris Program Studi S1 Ilmu Komputer Universitas Sumatera Utara
5. Ibu Dr. T. Henny Febriana Harumy S.Kom., M.Kom. sebagai dosen pembimbing I yang telah membantu penulis dalam menyelesaikan skripsi dengan memberikan bimbingan, saran, kritik, serta motivasi.
6. Bapak Handrizal S.Si., M.Comp.Sc sebagai dosen pembimbing II yang telah membantu penulis dalam menyelesaikan skripsi dengan memberikan bimbingan, saran, kritik, serta motivasi.
7. Seluruh dosen, staf pengajar, dan pegawai di Program Studi S-1 Ilmu Komputer, Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Sumatera Utara yang telah membantu penulis dalam proses penyusunan skripsi.
8. Orang tua penulis yang telah memberikan semangat, motivasi, doa, serta kasih sayang kepada penulis dalam proses penyusunan skripsi ini.
9. Saudara kandung penulis, Mikhael Alan Perdana Simanjuntak, Reinhad Ericson Simanjuntak, dan Gilbert Halomoan Simanjuntak yang telah

memberikan dukungan, semangat, dan motivasi selama proses penyusunan skripsi ini.

10. Teman teman penulis, Abel Agustian Sidauruk, Albertman Putra Barasa, Andri Hasudungan, Daniel Tambunan, dan Ferdi Nolan Jeremy Nainggolan yang sudah menemani penulis, serta memberikan semangat dalam penyelesaian skripsi ini.

Semoga Tuhan Yang Maha Kuasa memberkati semua pihak yang terlibat yang telah memberikan motivasi, saran, saran dan bantuan kepada penulis dalam menyelesaikan karya ini. Penulis berharap semoga makalah ini bermanfaat dan memberikan kontribusi bagi perkembangan bidang ilmu pengetahuan khususnya ilmu komputer.

Medan, 18 September 2024

Penulis



Alex Mario Simanjuntak

## KLASIFIKASI DAN DETEKSI PENYAKIT KULIT MENGGUNAKAN METODE EFFICIENTNETB7 DAN YOLOV8 BERBASIS *WEBSITE*

### ABSTRAK

Penyakit kulit merupakan masalah kesehatan yang sering dialami oleh masyarakat di Indonesia, terutama di daerah dengan akses terbatas terhadap layanan medis. Deteksi dini penyakit kulit sangat penting, namun sering kali sulit dilakukan tanpa bantuan tenaga medis profesional. Oleh karena itu, penelitian ini bertujuan untuk mengembangkan sistem diagnosis berbasis kecerdasan buatan yang dapat membantu mendeteksi penyakit kulit secara lebih akurat. *Convolutional Neural Network* (CNN) telah terbukti efektif dalam klasifikasi gambar dan diimplementasikan dalam penelitian ini. Penelitian ini menghadirkan solusi diagnosis dini penyakit kulit dengan memanfaatkan CNN berbasis EfficientNetB7 untuk klasifikasi dan YOLOv8 untuk deteksi. Sistem ini dirancang untuk mengklasifikasikan lima jenis penyakit kulit: *Melanoma*, *Basal Cell Carcinoma* (BCC), *Melanocytic Nevi* (NV), *Benign Keratosis-like Lesions* (BKL), dan *Seborrheic Keratoses and other Benign Tumors*, serta mendeteksi apakah penyakit tersebut merupakan kanker atau bukan. Hasil penelitian menunjukkan bahwa model EfficientNetB7 berhasil mencapai akurasi 94% pada data pengujian, sementara YOLOv8 menunjukkan kinerja deteksi dengan mean average precision (mAP) sebesar 0.812. Sistem berbasis *website* yang dikembangkan mampu memproses gambar kulit dan memberikan hasil klasifikasi dan deteksi secara efisien, serta terbukti stabil dalam berbagai pengujian performa. Kombinasi EfficientNetB7 dan YOLOv8 dalam sistem diagnosis dini penyakit kulit memberikan hasil yang baik dan dapat menjadi alat bantu yang efektif, terutama bagi masyarakat dengan akses terbatas terhadap layanan kesehatan.

**Kata Kunci :** *Convolutional neural network* (CNN), EfficientNetB7, YOLOv8, Penyakit Kulit.

## **CLASSIFICATION AND DETECTION OF SKIN DISEASES USING WEBSITE-BASED EFFICIENTNETB7 AND YOLOV8 METHODS**

### **ABSTRACT**

Skin diseases are a common health problem experienced by people in Indonesia, especially in areas with limited access to medical services. Early detection of skin diseases is very important, but it is often difficult to do without the help of medical professionals. Therefore, this research aims to develop an artificial intelligence-based diagnosis system that can help detect skin diseases more accurately. Convolutional Neural Network (CNN) has been proven effective in image classification and is implemented in this research. This research presents a solution for early diagnosis of skin diseases by utilizing CNN based on EfficientNetB7 for classification and YOLOv8 for detection. The system is designed to classify five types of skin diseases: Melanoma, Basal Cell Carcinoma (BCC), Melanocytic Nevi (NV), Benign Keratosis-like Lesions (BKL), and Seborrheic Keratoses and other Benign Tumors, and detect whether the disease is cancer or not. The results showed that the EfficientNetB7 model achieved 94% accuracy on the test data, while YOLOv8 showed detection performance with a mean average precision (mAP) of 0.812. The web-based system developed was able to process skin images and provide classification and detection results efficiently, and proved stable in various performance tests. The combination of EfficientNetB7 and YOLOv8 in the early diagnosis system of skin diseases has led to the development of a new system.

**Keywords :** Convolutional neural networks (CNN), EfficientNetB7, YOLOv8, Skin Disease.



## DAFTAR ISI

PERSETUJUAN .....	iii
PERNYATAAN .....	iv
PENGHARGAAN .....	v
ABSTRAK .....	vii
ABSTRACT .....	viii
DAFTAR ISI .....	ix
DAFTAR GAMBAR .....	xiii
DAFTAR LAMPIRAN .....	xv
DAFTAR TABEL .....	xvi
BAB I PENDAHULUAN .....	1
1.1. Latar Belakang .....	1
1.2. Rumusan Masalah .....	2
1.3. Ruang Lingkup Penelitian .....	3
1.4. Tujuan Penelitian .....	3
1.5. Manfaat Penelitian .....	4
1.6. Metodologi Penelitian .....	4
1.7. Sistematika Penulisan .....	6
BAB II LANDASAN TEORI .....	7
2.1. Pembelajaran Mesin .....	7
2.2. <i>Framework</i> .....	7
2.3. <i>Data preprocessing</i> .....	7
2.4. <i>Convolutional neural network</i> .....	7
2.5. EfficientNetB7 .....	8

2.6. YOLOv8 .....	9
2.7. <i>Model Evaluation</i> .....	10
2.8. Penelitian Relevan .....	14
BAB III ANALISIS DAN PERANCANGAN SISTEM .....	17
3.1. Analisis Sistem.....	17
3.1.1. <i>Analisis masalah</i> .....	17
3.1.2. <i>Analisis kebutuhan</i> .....	17
3.2. Gambaran Umum Sistem .....	19
3.3. Pemodelan Sistem .....	20
3.3.1. <i>Use case diagram</i> .....	20
3.3.2. <i>Activity diagram</i> .....	22
3.3.3. <i>Sequence diagram</i> .....	22
3.3.4. <i>Flowchart</i> .....	23
3.4. Pengambilan <i>Dataset</i> .....	25
3.5. Pra-pemrosesan Data.....	25
3.6. Pembangunan Model .....	27
3.7. Pelatihan Model .....	28
3.7.1. <i>Pelatihan model klasifikasi</i> .....	28
3.7.2. <i>Pelatihan model deteksi</i> .....	29
3.8. Evaluasi dan Pengujian Model.....	30
3.9. Perancangan Sistem Berbasis <i>Website</i> Sebagai Antarmuka .....	30
3.9.1. <i>Halaman utama</i> .....	30
3.9.2. <i>Halaman prediksi</i> .....	31
3.9.3. <i>Halaman hasil prediksi</i> .....	32
BAB IV IMPLEMENTASI DAN PENGUJIAN SISTEM .....	33
4.1. Implementasi Tahap Pengambilan <i>Dataset</i> .....	33
4.1.1. <i>Pengambilan data klasifikasi</i> .....	33

4.1.2. Pengambilan data deteksi .....	34
4.2. Implementasi Tahap Pra-pemrosesan Data .....	34
4.2.1. Pra pemrosesan data klasifikasi .....	34
4.2.2. Pra-pemrosesan data deteksi .....	37
4.3. Implementasi Tahap Pembangunan Model .....	38
4.3.1. Model klasifikasi .....	38
4.3.2. Model deteksi .....	40
4.4. Implementasi Tahap Pelatihan Model .....	40
4.4.1. Pelatihan model klasifikasi .....	40
4.4.2. Pelatihan model deteksi .....	45
4.5. Implementasi Tahap Evaluasi dan Pengujian Model .....	47
4.5.1. Evaluasi model klasifikasi .....	47
4.5.2. Evaluasi model deteksi .....	52
4.6. Analisis Proses EfficientNetB7 .....	56
4.6.1. Feature extraction .....	56
4.6.2. Pooling .....	57
4.6.3. Klasifikasi gambar .....	57
4.7. Analisis Proses YOLOv8 .....	60
4.7.1. Feature extraction .....	60
4.7.2. Bounding box prediction .....	60
4.7.3. Intersection over union (IoU) .....	61
4.7.4. Non-maximum suppression (NMS) .....	62
4.7.5. Output bounding box dan label kelas .....	63
4.8. Implementasi Sistem .....	63
4.8.1. Halaman utama .....	63
4.8.2. Halaman prediksi .....	64
4.8.3. Halaman hasil klasifikasi dan deteksi .....	65

4.8.4. <i>Halaman evaluasi model</i> .....	66
4.9. Pengujian Sistem.....	67
4.9.1. <i>Black box testing</i> .....	67
4.9.2. <i>Percobaan klasifikasi dan deteksi</i> .....	68
BAB V PENUTUP .....	70
5.1. Kesimpulan .....	70
5.2. Saran .....	71
DAFTAR PUSTAKA .....	72
LAMPIRAN.....	75

## DAFTAR GAMBAR

<b>Gambar 1. 1.</b> Citra jenis-jenis penyakit kulit.....	2
<b>Gambar 2. 1.</b> Arsitektur EfficientNetB7 .....	8
<b>Gambar 2. 2.</b> Perbandingan Performa Model.....	9
<b>Gambar 2. 3.</b> Arsitektur YOLOv8.....	10
<b>Gambar 3. 1.</b> Diagram <i>Ishikawa</i> .....	18
<b>Gambar 3. 2.</b> Arsitektur Umum Sistem.....	20
<b>Gambar 3. 3.</b> <i>Use case diagram</i> .....	21
<b>Gambar 3. 4.</b> <i>Activity diagram</i> .....	22
<b>Gambar 3. 5.</b> <i>Sequence diagram</i> .....	23
<b>Gambar 3. 6.</b> <i>Flowchart</i> Sistem.....	24
<b>Gambar 3. 7.</b> Rancangan Halaman Utama .....	30
<b>Gambar 3. 8.</b> Rancangan Halaman Prediksi Gambar .....	31
<b>Gambar 3. 9.</b> Rancangan Halaman Hasil Prediksi .....	32
<b>Gambar 4. 1.</b> <i>Dataset Skin Disease</i> .....	33
<b>Gambar 4. 2.</b> Kelas Penyakit Kulit yang Digunakan.....	34
<b>Gambar 4. 3.</b> Kode Penyaringan Folder .....	35
<b>Gambar 4. 4.</b> Kode Pembagian <i>Dataset</i> .....	35
<b>Gambar 4. 5.</b> Konfigurasi Generator Data Pelatihan.....	36
<b>Gambar 4. 6.</b> Konfigurasi Generator Data Validasi dan Pengujian .....	37
<b>Gambar 4. 7.</b> Pembagian <i>Dataset</i> Deteksi .....	38
<b>Gambar 4. 8.</b> Konfigurasi <i>data.yaml</i> .....	38
<b>Gambar 4. 9.</b> Import <i>Library</i> Tensorflow.....	39
<b>Gambar 4. 10.</b> Model Dasar EfficientNetB7 .....	39
<b>Gambar 4. 11.</b> Menambahkan Lapisan Model .....	39
<b>Gambar 4. 12.</b> Import <i>library</i> Ultralytics .....	40
<b>Gambar 4. 13.</b> Konfigurasi Parameter YOLOv8.....	40
<b>Gambar 4. 14.</b> Proses Pelatihan Model (8:1:1).....	41
<b>Gambar 4. 15.</b> Proses Pelatihan Model (7:1.5:1.5).....	41
<b>Gambar 4. 16.</b> Proses Pelatihan Model (5:2.5:2.5).....	42

<b>Gambar 4. 17.</b> Grafik <i>Loss</i> Selama Pelatihan (8:1:1).....	42
<b>Gambar 4. 18.</b> Grafik <i>Loss</i> Selama Pelatihan (7:1.5:1.5).....	43
<b>Gambar 4. 19.</b> Grafik <i>Loss</i> Selama Pelatihan (5:2.5:2.5).....	43
<b>Gambar 4. 20.</b> Grafik Akurasi Selama Pelatihan (8:1:1).....	44
<b>Gambar 4. 21.</b> Grafik Akurasi Selama Pelatihan (7:1.5:1.5).....	44
<b>Gambar 4. 22.</b> Grafik Akurasi Selama Pelatihan (5:2.5:2.5).....	45
<b>Gambar 4. 23.</b> Proses Pelatihan YOLOv8 (1500 gambar) .....	46
<b>Gambar 4. 24.</b> Proses Pelatihan YOLOv8 (3000 gambar) .....	47
<b>Gambar 4. 25.</b> Kode Pengujian Model Klasifikasi.....	48
<b>Gambar 4. 26.</b> <i>Confusion Matrix</i> Model Klasifikasi (8:1:1) .....	48
<b>Gambar 4. 27.</b> <i>Confusion Matrix</i> Model Klasifikasi (7:1.5:1.5) .....	49
<b>Gambar 4. 28.</b> <i>Confusion Matrix</i> Model Klasifikasi (5:2.5:2.5) .....	49
<b>Gambar 4. 29.</b> <i>Classification Report</i> (8:1:1) .....	49
<b>Gambar 4. 30.</b> <i>Classification Report</i> (7:1.5:1.5) .....	50
<b>Gambar 4. 31.</b> <i>Classification Report</i> (5:2.5:2.5) .....	50
<b>Gambar 4. 32.</b> Kode Pengujian Model Deteksi.....	52
<b>Gambar 4. 33.</b> <i>Confusion Matrix</i> Model Deteksi 1500 Gambar .....	53
<b>Gambar 4. 34.</b> <i>Confusion Matrix</i> Model Deteksi 3000 Gambar .....	53
<b>Gambar 4. 35.</b> Grafik Pelatihan Model Deteksi 1500 Gambar .....	54
<b>Gambar 4. 36.</b> Grafik Pelatihan Model Deteksi 3000 Gambar .....	54
<b>Gambar 4. 37.</b> Feature Extraction.....	56
<b>Gambar 4. 38.</b> Teknik <i>Pooling</i> .....	57
<b>Gambar 4. 39.</b> Contoh Pencarian <i>Bounding Box</i> .....	60
<b>Gambar 4. 40.</b> BBox pred vs <i>ground truth</i> .....	61
<b>Gambar 4. 41.</b> Halaman Utama .....	63
<b>Gambar 4. 42.</b> Halaman Prediksi Gambar.....	64
<b>Gambar 4. 43.</b> Halaman Hasil Prediksi Gambar .....	65
<b>Gambar 4. 44.</b> Halaman evaluasi zip file .....	66
<b>Gambar 4. 45.</b> Halaman evaluasi multi <i>images</i> .....	66

**DAFTAR LAMPIRAN**

Lampiran 1. Kode Program.....	75
Lampiran 2. Biodata.....	82

**DAFTAR TABEL**

<b>Tabel 2.1.</b> Tabel <i>Confusion Matrix</i> untuk Klasifikasi Biner .....	11
<b>Tabel 2.2.</b> Tabel <i>Confusion Matrix</i> untuk Multi Kelas .....	11
<b>Tabel 4.1.</b> Perbandingan Model Klasifikasi .....	50
<b>Tabel 4.2.</b> Perbandingan Model Deteksi .....	53
<b>Tabel 4.3.</b> Pengujian Fungsional pada <i>Website</i> .....	67
<b>Tabel 4.4.</b> Percobaan Klasifikasi dan Deteksi .....	68



# **BAB I**

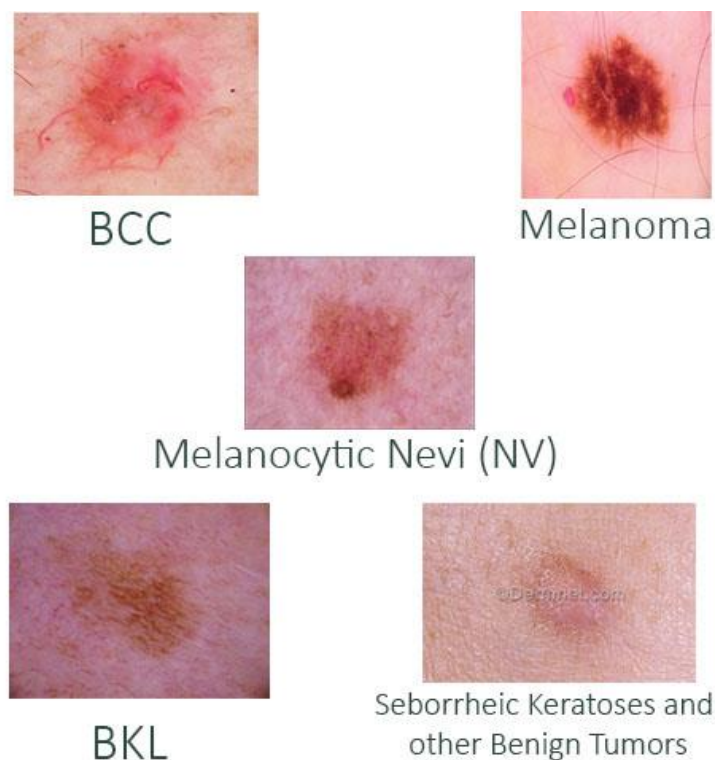
## **PENDAHULUAN**

### **1.1. Latar Belakang**

Penyakit kulit adalah suatu isu kesehatan yang umum dihadapi oleh masyarakat di Indonesia (Triyono et al., 2022). Berbagai jenis penyakit kulit, seperti dermatitis, campak, psoriasis, dan lainnya, sering kali sulit didiagnosis secara akurat tanpa bantuan tenaga medis profesional. Namun, akses ke layanan kesehatan profesional sering kali menjadi masalah, terutama di daerah-daerah yang terpencil. Oleh karena itu, memiliki alat yang dapat membantu dalam diagnosis dini penyakit kulit menjadi sangat penting.

*Convolutional neural network* (CNN) adalah teknologi kecerdasan buatan yang telah terbukti efektif dalam klasifikasi gambar (Lubis et al., 2023). Dalam beberapa penelitian, CNN telah digunakan untuk mengklasifikasikan berbagai jenis penyakit kulit dengan tingkat akurasi yang cukup tinggi (Lubis et al., 2023; Triyono et al., 2022). Misalnya, sebuah sistem telah dibuat untuk mengklasifikasikan tiga jenis penyakit kulit yang umum dihadapi oleh masyarakat Indonesia, yaitu panu, kadas, dan kudis, dengan tingkat akurasi mencapai 96,75% (Triyono et al., 2022). Namun, hingga saat ini, sebagian besar penelitian cenderung fokus pada klasifikasi sejumlah kecil jenis penyakit kulit. Oleh karena itu, ada ruang untuk peningkatan dalam hal ini.

Dalam skripsi ini, penulis merancang dan mengimplementasikan model deteksi dan klasifikasi gambar menggunakan CNN untuk mengklasifikasikan 5 kelas penyakit kulit yang berbeda, serta mendeteksi apakah penyakit tersebut termasuk kedalam kategori kanker atau bukan. Tujuan penelitian ini adalah untuk meningkatkan kemampuan mendeteksi penyakit kulit secara dini. Jenis penyakit kulit yang diklasifikasikan oleh penulis yaitu: *Melanoma*, *Basal Sel Carcinoma* (BCC), *Melanocytic Nevi* (NV), *Benign Keratosis-like Lesions* (BKL), dan *Seborrheic Keratoses and other Benign Tumors*, sebagaimana disajikan pada Gambar 1.1 di halaman 2.



**Gambar 1.1.** Citra jenis-jenis penyakit kulit

Dalam penelitian ini, penulis menggunakan *dataset* yang dapat diakses melalui Kaggle. Sebagai upaya untuk meningkatkan efisiensi dan efektivitas pembuatan model, penulis menerapkan teknik *Transfer Learning* menggunakan EfficientNetB7 dan YOLOv8. Penulis menggunakan Tensorflow dan Ultralytics, sebuah platform komprehensif untuk *machine learning*, untuk melatih model *Convolutional neural network* (CNN). Model CNN yang dibangun oleh penulis dirancang khusus untuk mengklasifikasikan gambar menjadi 5 penyakit kulit yang berbeda, serta mendeteksi apakah penyakit kulit tersebut masuk kedalam kategori kanker atau tidak.

## 1.2. Rumusan Masalah

Tantangan dalam diagnosis penyakit kulit, terutama bagi masyarakat di daerah dengan akses terbatas terhadap layanan kesehatan, memerlukan solusi inovatif untuk membantu mendiagnosis secara dini. Berbagai jenis penyakit kulit, seperti *Melanoma*, *BCC*, *NV*, *BKL*, dan *Seborrheic keratoses and other Benign Tumors*, sering kali sulit dideteksi tanpa bantuan tenaga medis profesional. Sehubungan dengan hal tersebut, studi ini bertujuan untuk memanfaatkan dan mengoptimalkan algoritma Convolutional Neural Network (CNN) berbasis EfficientNetB7 dan YOLOv8, guna meningkatkan

akurasi dan efisiensi dalam deteksi lima jenis penyakit kulit tersebut, sehingga dapat menjadi alat yang secara efektif dapat membantu proses diagnosis dini.

### 1.3. Ruang Lingkup Penelitian

Berikut ialah batasan masalah yang ditemukan pada studi ini:

1. *Dataset* yang digunakan dapat diakses melalui platform Kaggle.
2. Penelitian ini hanya fokus pada deteksi dan klasifikasi lima jenis penyakit kulit yang telah disebutkan sebelumnya.
3. Metode yang digunakan untuk pembuatan model adalah *Transfer Learning* menggunakan arsitektur EfficientNetB7 dan YOLOv8.
4. Implementasi model dilakukan menggunakan Tensorflow dan Ultralytics.
5. Evaluasi model dilakukan dengan mengukur akurasi klasifikasi dan mungkin metrik evaluasi tambahan seperti *precision*, *recall*, *F1-score*, dan mAP.
6. Penelitian ini mencakup pembuatan model deteksi dan klasifikasi menggunakan *Convolutional neural network* (CNN) berbasis EfficientNetB7 dan YOLOv8 untuk lima jenis penyakit kulit yang disebutkan, serta integrasi model ke dalam sebuah *website* sebagai antarmuka pengguna. Namun, penelitian ini tidak mengkaji secara rinci mengenai desain antarmuka pengguna, manajemen *database*, atau aspek teknis lainnya terkait dengan pengembangan *website*. Fokus utama akan tetap pada pembangunan dan evaluasi model deteksi dan klasifikasi penyakit kulit.

### 1.4. Tujuan Penelitian

Studi ini bertujuan untuk mengembangkan serta mengimplementasikan model klasifikasi dan deteksi gambar menggunakan Convolutional Neural Network (CNN) berbasis EfficientNetB7 dan YOLOv8, dengan fokus pada lima jenis penyakit kulit: *Melanoma*, *BCC*, *NV*, *BKL*, serta *Seborrheic Keratoses and other Benign Tumors*. Tujuan utama penelitian ini adalah meningkatkan kemampuan diagnosis dini penyakit kulit melalui evaluasi model menggunakan metrik akurasi klasifikasi, serta metrik tambahan seperti *precision*, *recall*, *F1-score*, dan mean Average Precision (mAP). Selain itu, model yang dikembangkan kemudian diintegrasikan ke dalam sebuah *website* sebagai antarmuka pengguna yang mudah diakses oleh masyarakat umum maupun tenaga medis untuk mempermudah penggunaan dan diagnosis.

### 1.5. Manfaat Penelitian

Berikut ialah beberapa manfaat yang dapat diperoleh dari studi ini:

1. Peningkatan Diagnosis Awal: Model klasifikasi yang diimplementasikan dapat membantu dalam diagnosis awal penyakit kulit, memungkinkan deteksi dini dan intervensi yang tepat waktu.
2. Aksesibilitas: Integrasi model ke dalam sebuah *website* sebagai antarmuka pengguna dapat meningkatkan aksesibilitasnya bagi masyarakat umum atau tenaga medis di wilayah terpencil atau yang memiliki akses terbatas ke layanan kesehatan.
3. Edukasi: *Website* yang menyediakan akses ke model klasifikasi dapat menjadi sumber pendidikan yang bernilai, membantu masyarakat untuk lebih memahami tentang berbagai jenis penyakit kulit dan pentingnya diagnosis dini.
4. Kontribusi untuk penelitian lanjutan: Penelitian ini dapat menjadi landasan untuk pengembangan model klasifikasi atau deteksi penyakit kulit yang lebih maju dan menyeluruh.
5. Potensi Penerapan di Dunia Nyata: Model yang dikembangkan dalam penelitian ini memiliki potensi untuk diterapkan dalam praktik medis sehari-hari.

### 1.6. Metodologi Penelitian

Metode-metode yang diterapkan dalam penelitian ini meliputi :

1. Studi Pustaka

Pada bagian ini, peneliti membahas tinjauan literatur mengenai teknik-teknik terbaru dalam deteksi dan klasifikasi gambar, terutama yang terkait dengan diagnosis penyakit kulit menggunakan CNN dan teknik *Transfer Learning*, juga memahami konsep dan metodologi yang digunakan dalam penelitian sebelumnya yang relevan dengan topik ini.

2. Pembuatan *Dataset*

Peneliti kemudian mengumpulkan *dataset* gambar yang sesuai untuk lima jenis penyakit kulit yang diklasifikasikan, kemudian menyusun *dataset* tersebut dengan membaginya menjadi kelas-kelas yang sesuai, kemudian peneliti memberikan *bounding box* pada *dataset* yang digunakan untuk

mendeteksi apakah penyakit tersebut masuk kedalam kategori kanker atau tidak.

### 3. Perancangan

Bagian ini melibatkan perancang arsitektur model klasifikasi menggunakan *Convolutional neural network* (CNN) berbasis EfficientNetB7 dan YOLOv8, kemudian peneliti menentukan parameter dan konfigurasi model, termasuk lapisan-lapisan yang diperlukan, fungsi aktivasi, dan fungsi kerugian yang tepat.

### 4. Implementasi Sistem

Pada tahap ini, peneliti mengimplementasikan model deteksi dan klasifikasi menggunakan Tensorflow dan Ultralytics dengan memanfaatkan teknik *Transfer Learning* untuk meningkatkan efisiensi dan akurasi model selama tahap pelatihan. Peneliti menggunakan *dataset* yang telah disiapkan sebelumnya untuk melatih model, kemudian melakukan proses pelatihan dengan menyesuaikan model ke *dataset* pelatihan dan memvalidasi kinerjanya menggunakan *dataset* validasi.

### 5. Pengujian Sistem

Bagian ini melibatkan pengujian kinerja sistem, dimana peneliti mengevaluasi kinerja model yang telah dilatih menggunakan *dataset* uji terpisah. Peneliti kemudian mengukur akurasi klasifikasi dan metrik evaluasi lainnya seperti *precision*, *recall*, *F1-score*, dan mAP.

### 6. Dokumentasi Sistem

Peneliti mendokumentasikan semua langkah yang diambil dalam proses merancang, menerapkan, dan menguji sistem. Setelah itu, peneliti menyusun laporan akhir yang memuat hasil temuan, kesimpulan, dan rekomendasi untuk penelitian berikutnya.

## 1.7. Sistematika Penulisan

### **BAB 1        PENDAHULUAN**

Bagian ini membahas latar belakang penelitian, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian, serta sistem penulisan.

### **BAB 2        LANDASAN TEORI**

Bagian ini menguraikan konsep teoretis yang mendukung penelitian ini, termasuk pembelajaran mesin, jaringan saraf konvolusional (CNN), EfficientNetB7, YOLOv8, dan topik terkait lainnya.

### **BAB 3        ANALISIS DAN PERANCANGAN**

Bagian ini menguraikan proses analisis masalah penelitian dan merancang solusi menggunakan metode EfficientNetB7 dan YOLOv8 untuk mengklasifikasikan dan mendeteksi penyakit kulit.

### **BAB 4        IMPLEMENTASI DAN PENGUJIAN**

Bagian ini menjelaskan implementasi model yang dirancang dalam penelitian dan proses pengujian kinerja model pada *dataset* yang telah disiapkan.

### **BAB 5        KESIMPULAN DAN SARAN**

Bagian ini menyampaikan ringkasan hasil penelitian dan beberapa saran bagi penelitian selanjutnya guna mengembangkan model klasifikasi dan deteksi.

## **BAB II**

### **LANDASAN TEORI**

#### **2.1. Pembelajaran Mesin**

Pembelajaran mesin merupakan suatu bidang kecerdasan buatan yang memungkinkan sistem komputer mempelajari dan meningkatkan kinerja tanpa pemrograman secara eksplisit. Algoritma pembelajaran mesin dapat menganalisis pola-pola kompleks dalam data yang diberikan dan membuat prediksi berdasarkan pemahaman tersebut. Dalam konteks penelitian ini, pembelajaran mesin digunakan untuk mendeteksi dan mengklasifikasikan 5 jenis penyakit kulit.

#### **2.2. Framework**

*Framework* adalah kerangka kerja perangkat lunak yang menyediakan alat, fungsi, dan struktur yang diperlukan untuk membangun, melatih, dan mengevaluasi model pembelajaran mesin secara efisien (Salvi et al., 2021). Dalam konteks pembangunan model deteksi dan klasifikasi, pemilihan *framework deep learning* yang tepat sangat penting. *Framework* seperti Tensorflow adalah contoh populer dari *framework deep learning* yang biasanya digunakan untuk pengembangan model *deep learning*, termasuk *Convolutional neural network* (CNN). Mereka menawarkan berbagai fungsi dan utilitas yang memudahkan pengguna untuk merancang, melatih, dan menguji model dengan berbagai skala dan kompleksitas.

#### **2.3. Data preprocessing**

Tahap pra-pemrosesan data sangat penting dalam mengembangkan model *Machine learning*. Ini meliputi langkah-langkah seperti normalisasi data, augmentasi data, dan pembagian *dataset* menjadi set pelatihan, validasi, dan pengujian. Dalam penelitian ini, prapemrosesan data diterapkan untuk menyiapkan *dataset* gambar yang digunakan dalam proses pelatihan dan pengujian model.

#### **2.4. Convolutional neural network**

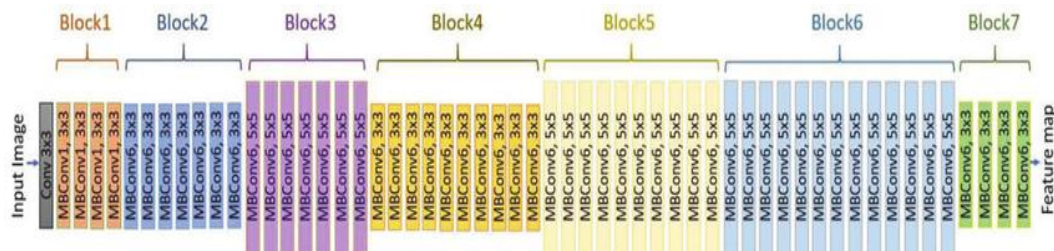
CNN adalah jenis arsitektur jaringan saraf tiruan yang sangat baik dalam mengolah data gambar. CNN memiliki beberapa lapisan khusus seperti lapisan

konvolusi, *pooling*, dan fully connected yang memungkinkannya untuk mengekstraksi fitur dari gambar dengan baik. Dalam penelitian ini, CNN digunakan sebagai model untuk deteksi dan klasifikasi gambar penyakit kulit.

## 2.5. EfficientNetB7

EfficientNet adalah serangkaian arsitektur CNN yang dirancang untuk memberikan performa yang tinggi dengan mempertimbangkan efisiensi komputasi. Model EfficientNet menggunakan pendekatan scaling untuk mengoptimalkan arsitektur dasar dan parameter-parameter jaringan untuk berbagai ukuran dan kebutuhan tugas. EfficientNetB7 adalah bagian dari keluarga model EfficientNet yang dirancang oleh Mingxing Tan dan Quoc V. Le (Tan & Le, 2019).

Model ini dirancang untuk mencapai tingkat ketepatan dan efisiensi yang lebih tinggi daripada *Convolutional neural networks* (ConvNets) sebelumnya (Tan & Le, 2019). Dalam penelitian ini, EfficientNetB7 digunakan sebagai arsitektur dasar untuk model klasifikasi gambar penyakit kulit. Arsitektur EfficientNetB7 ditampilkan pada Gambar 2.1.

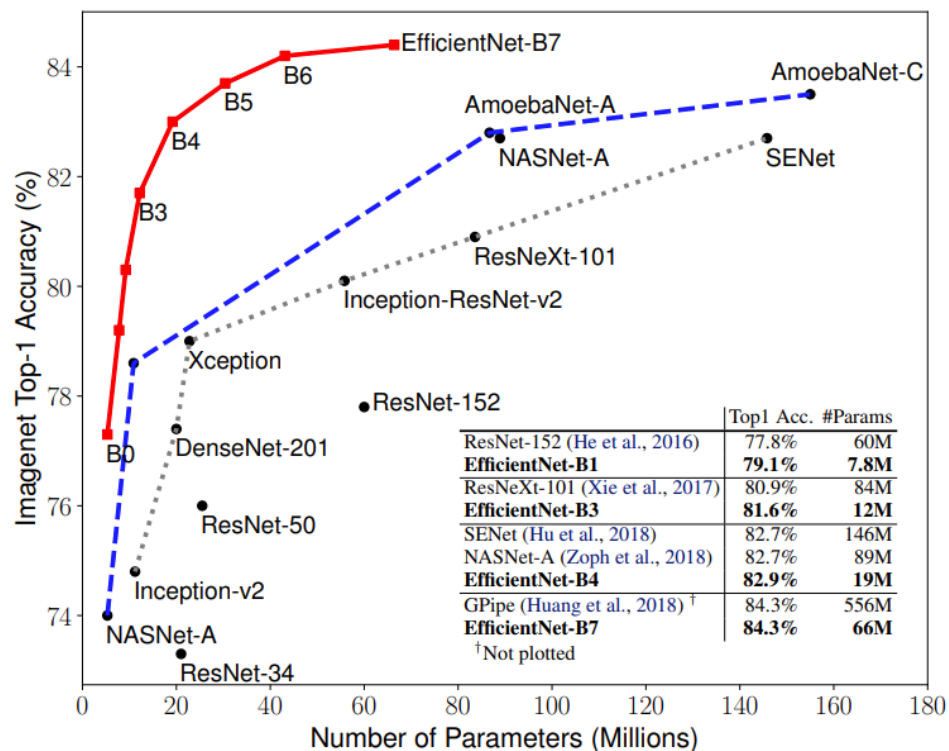


**Gambar 2.1.** Arsitektur EfficientNetB7

Sumber: Khalil et al., 2022

*Baseline model* EfficientNet ini yaitu EfficientNetB0, tidak didesain oleh manusia melainkan didesain oleh neural network lain, yaitu *neural architecture search* (NAS). Untuk mendapatkan EfficientNetB7, dilakukan yang namanya *compound scaling*. *Compound scaling* ini maksudnya ialah untuk menemukan nilai *depth* (banyaknya layer), *width* (banyaknya *feature map* / neuron) , dan *resolution* yang tepat.





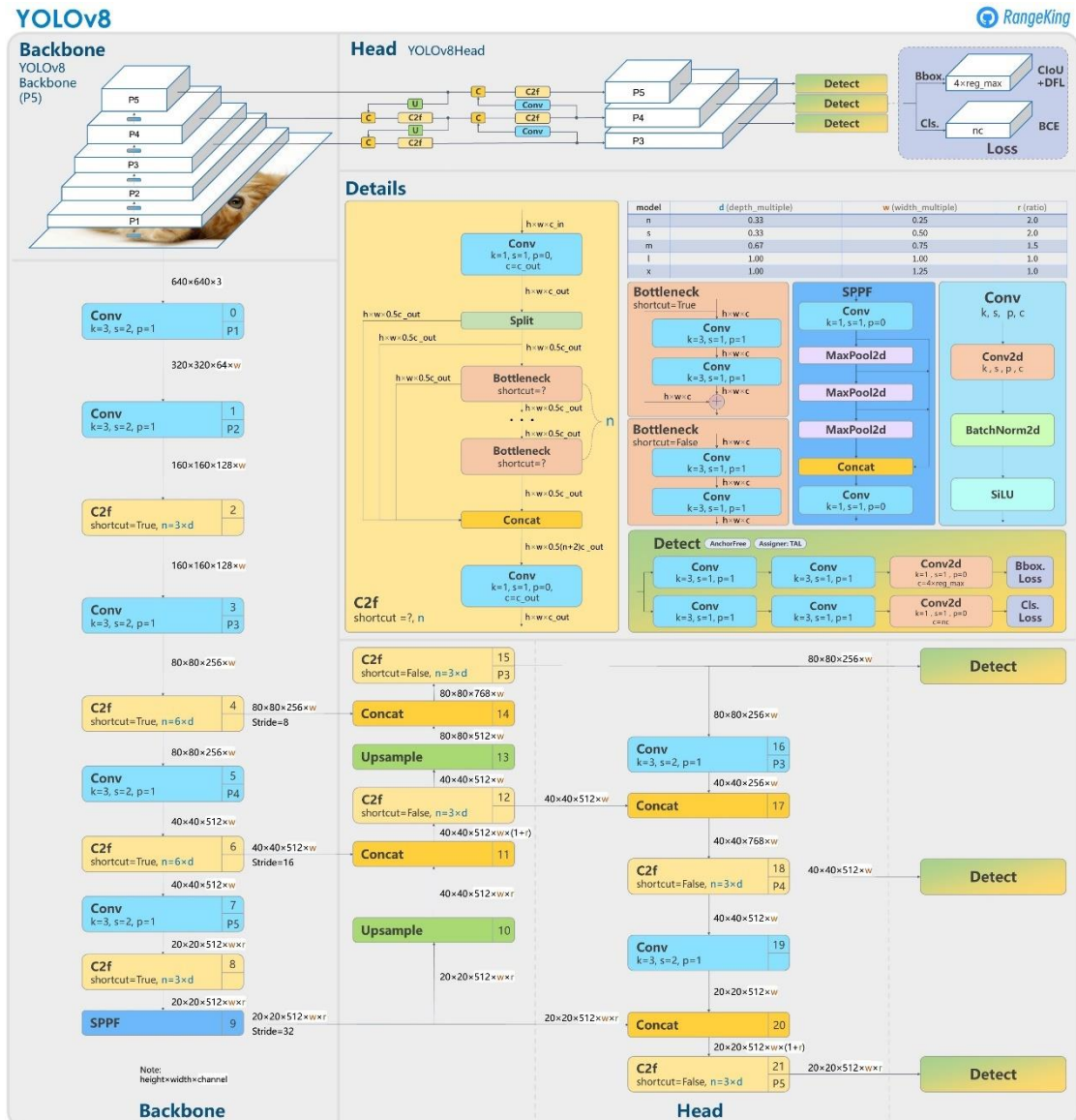
**Gambar 2. 2.** Perbandingan Performa Model

Sumber: Tan & Le, 2019

Seperti yang terlihat pada Gambar 2.2. Model EfficientNetB7 terbukti memiliki *accuracy* yang lebih unggul dibandingkan dengan model lainnya.

## 2.6. YOLOv8

YOLOv8 adalah model dalam seri YOLO (*You Only Look Once*), yang merupakan model deteksi objek *real-time* populer (Mupparaju et al., 2024). Dibangun oleh Ultralytics, pengembang YOLOv5, YOLOv8 menawarkan dukungan langsung untuk deteksi objek, klasifikasi, dan tugas segmentasi, yang dapat diakses melalui paket Python serta antarmuka baris perintah. YOLOv8 dirancang untuk menjadi cepat, akurat, dan mudah digunakan, menjadikannya pilihan yang sangat baik untuk berbagai tugas deteksi dan pelacakan objek, segmentasi instan, klasifikasi gambar, dan estimasi pose. Arsitektur YOLOv8 ditampilkan pada Gambar 2.3 di halaman 10.



Gambar 2.3. Arsitektur YOLOv8

Sumber: <https://github.com/RangeKing>

## 2.7. Model Evaluation

Evaluasi model merupakan fase krusial dalam pengembangan model *Machine learning*. Metrik evaluasi seperti *mAP*, *accuracy*, *recall*, *precision*, dan *F1-score* digunakan untuk mengevaluasi model dalam melakukan klasifikasi dan deteksi (Reis et al., 2023; Vickers et al., 2024). Evaluasi yang menyeluruh dapat membantu dalam memahami sejauh mana model mampu menerapkan pola dari data yang belum pernah ditemui sebelumnya (Hicks et al., 2022).

*Confusion matrix* seperti pada Tabel 2.1 merupakan tabel yang biasanya digunakan untuk mengevaluasi hasil dari performa model pada kumpulan data *testing* yang nilainya sudah diketahui. *Confusion matrix* biasanya menunjukkan berapa banyak data yang diklasifikasikan dengan benar dan juga sebaliknya.

Tabel *confusion matrix* umumnya terdiri dari empat sel atau elemen utama:

1. *True Positive* (TP): keadaan dimana model dengan benar memprediksi kelas positif.
2. *True Negative* (TN): keadaan dimana model dengan benar memprediksi kelas negatif.
3. *False Positive* (FP): keadaan dimana model keliru memprediksi kelas positif (mengklasifikasikan negatif sebagai positif).
4. *False Negative* (FN): keadaan dimana model keliru memprediksi kelas negatif (mengklasifikasikan positif sebagai negatif).

**Tabel 2.1.** Tabel *Confusion Matrix* untuk Klasifikasi Biner

<i>Actual Values</i>	<i>Predicted Values</i>		
		<i>Positive (1)</i>	<i>Negative(0)</i>
	<i>Positive (1)</i>	TP	FN
	<i>Negative (0)</i>	FP	TN

**Tabel 2.2.** Tabel *Confusion Matrix* untuk Multi Kelas

<i>Actual Values</i>	<i>Predicted Values</i>					
		1	2	3	4	5
	1	TP1 (sel 1)	FN/FP (sel 2)	FN/FP (sel 3)	FN/FP (sel 4)	FN/FP (sel 5)
	2	FN/FP (sel 6)	TP2 (sel 7)	FN/FP (sel 8)	FN/FP (sel 9)	FN/FP (sel 10)
	3	FN/FP (sel 11)	FN/FP (sel 12)	TP3 (sel 13)	FN/FP (sel 14)	FN/FP (sel 15)
	4	FN/FP (sel 16)	FN/FP (sel 17)	FN/FP (sel 18)	TP4 (sel 19)	FN/FP (sel 20)
	5	FN/FP (sel 21)	FN/FP (sel 22)	FN/FP (sel 23)	FN/FP (sel 24)	TP5 (sel 25)

Pada masalah klasifikasi multi kelas kita tidak akan mendapatkan nilai TP, FN, FP, dan TN seperti pada masalah klasifikasi biner. Saat melakukan validasi, kita perlu menghitung nilai tersebut untuk setiap kelas dimana :

- TP adalah nilai di mana nilai aktual dan nilai prediksi sesuai.
- FN untuk suatu kelas merupakan jumlah dari nilai dalam baris terkait kecuali nilai TP.
- FP untuk suatu kelas merupakan jumlah dari nilai dalam kolom terkait kecuali nilai TP.
- TN untuk suatu kelas merupakan total nilai dari semua kolom dan baris yang tidak termasuk dalam kelas yang sedang dihitung (TP/FP/FN).

Sebagai contoh untuk menghitung TP, FN, FP, dan TN kelas 1 seperti pada Tabel 2.2 pada halaman 11, maka :

- TP = Sel 1
- FN = Sel 2 + Sel 3 + Sel 4 + Sel 5
- FP = Sel 6 + Sel 11 + Sel 16 + Sel 21
- TN =  $\sum_{i=7}^{10} Sel_i + \sum_{i=12}^{15} Sel_i + \sum_{i=17}^{20} Sel_i + \sum_{i=22}^{25} Sel_i$

Adapun metrik evaluasi yang digunakan ialah:

1. *Accuracy*: Mengukur persentase hasil prediksi yang benar dari seluruh prediksi yang dibuat oleh model. Memberikan gambaran umum tentang seberapa sering model menghasilkan prediksi yang benar. Perhitungannya dapat dilihat pada Persamaan (1).

$$Accuracy = \frac{TP}{Total\ data} \quad (1)$$

2. *Precision*: Merupakan perbandingan antara prediksi positif yang benar (*true positive*) dibandingkan dengan semua prediksi positif (*true positive + false positive*). Mengukur seberapa akurat model dalam mengidentifikasi kelas positif. Perhitungannya dapat dilihat pada Persamaan (2).

$$Precision = \frac{TP}{TP+FP} \quad (2)$$

3. *Recall*: Merupakan perbandingan antara jumlah prediksi positif yang benar (*true positive*) dan total data aktual yang positif (*true positive + false negative*). Metrik ini mengukur sejauh mana model mampu menangkap semua data yang benar-benar positif. Perhitungannya dapat dilihat pada Persamaan (3).

$$Recall = \frac{TP}{TP+FN} \quad (3)$$

4. *F1-score*: Merupakan nilai rata-rata dari *precision* dan *recall*. Metrik ini menyediakan keseimbangan antara *precision* dan *recall*. Metrik ini berguna ketika keseimbangan antara keduanya penting. Perhitungannya dapat dilihat pada Persamaan (4).

$$F1-score = 2 \times \frac{Precision \times Recall}{Precision+Recall} \quad (4)$$

5. *Mean average precision* (mAP): merupakan rata-rata dari *Average Precision* (AP) untuk setiap kelas. AP sendiri adalah rata-rata dari *precision* di setiap titik di mana *recall* berubah dalam *Precision-Recall* curve. mAP memberikan gambaran tentang performa model secara keseluruhan dalam menangani semua kelas. Perhitungan mAP bisa cukup kompleks dan biasanya dilakukan dengan bantuan *library* atau fungsi khusus.

$$mAP = \frac{1}{N} \sum_{i=1}^N AP_i \quad (5)$$

Metrik-metrik ini memungkinkan evaluasi kinerja model secara komprehensif dan membantu mengidentifikasi *area* yang perlu diperbaiki untuk meningkatkan prediksi pada data baru

## 2.8. Penelitian Relevan

1. “*Classification of Skin Disease Using Deep learning Neural Networks with MobileNet V2 and LSTM*” (Srinivasu et al., 2021). Penelitian ini memperkenalkan pendekatan komputerisasi untuk klasifikasi penyakit kulit menggunakan *deep learning* berbasis MobileNet V2 dan *Long Short Term Memory* (LSTM). Model MobileNet V2 menunjukkan efisiensi tinggi dengan akurasi yang superior, cocok untuk perangkat komputasi ringan. Metode ini efisien dalam mempertahankan informasi stateful untuk prediksi yang akurat. Penggunaan matriks ko-kejadian level abu-abu untuk mengevaluasi perkembangan penyakit. Model yang diusulkan melampaui kinerja model-model state-of-the-art lainnya seperti FineTuned Neural Networks (FTNN), *Convolutional neural network* (CNN), dan Very Deep Convolutional Networks (VGG) dalam klasifikasi penyakit kulit pada *dataset* HAM10000, dengan akurasi melebihi 85%.
2. “*Six skin diseases classification using deep Convolutional neural network*” (Saifan & Jubair, 2022). Penelitian ini dilakukan oleh Ramzi Saifan dan Fahed Jubair dari Departemen Teknik Komputer, Universitas Yordania. Mereka mengembangkan sistem yang memanfaatkan *Convolutional neural network* (CNN) untuk mengklasifikasikan gambar lesi kulit berwarna. Sistem ini mengandalkan CNN yang telah dilatih sebelumnya untuk mengklasifikasikan antara enam penyakit kulit: jerawat, kaki atlet, cacar air, eksim, kanker kulit, dan vitiligo. Selain itu, mereka juga membuat *dataset* berisi 3000 gambar berwarna dari beberapa *dataset* online dan internet. Hasil eksperimen menunjukkan bahwa model yang diusulkan mencapai akurasi sebesar 81.75%, yang lebih tinggi dibandingkan dengan penelitian lain di bidang ini.
3. “*Transfer Learning with Ensembles of Deep Neural Networks for Skin Cancer Detection in Imbalanced Data Sets*” (Qureshi & Roos, 2023). Penelitian ini memperkenalkan pendekatan baru untuk deteksi kanker kulit menggunakan ensemble *Convolutional neural network* (CNN), yang terdiri dari model-model CNN yang telah dilatih sebelumnya dan beberapa dilatih khusus untuk data yang ada. Metode ini juga memanfaatkan metadata terkait gambar *input* untuk meningkatkan kemampuan model dalam menangani data terbatas dan tidak seimbang. Pendekatan ini diuji menggunakan *dataset* besar yang berisi

33.126 gambar dermoskopi dari 2056 pasien. Hasil evaluasi menunjukkan kinerja yang baik dalam berbagai metrik evaluasi seperti F1-measure, *area* di bawah kurva ROC (AUC-ROC), dan *area* di bawah kurva PR (AUC-PR), dibandingkan dengan tujuh metode benchmark yang berbeda, termasuk dua teknik terbaru berbasis CNN.

4. “*Automated Monkeypox Skin Lesion Detection Using Deep learning and Transfer Learning Techniques*” (Jaradat et al., 2023). Penelitian ini menggunakan pendekatan *deep learning* untuk deteksi lesi kulit cacar monyet. Lima model telah dievaluasi (VGG19, VGG16, ResNet50, MobileNetV2, dan EfficientNetB3). Hasilnya menunjukkan bahwa *MobileNetV2* memiliki performa terbaik dengan tingkat presisi 98,16%, *recall* 0,96, dan skor F1 0,98. Model ini juga berhasil diverifikasi dengan akurasi 94% pada *dataset* terpisah. Kesimpulannya, *MobileNetV2* mengungguli model lainnya dalam mengklasifikasikan cacar monyet, menunjukkan potensi teknik pembelajaran mesin untuk deteksi dini. Algoritme ini menjanjikan sebagai alat diagnosis yang cepat dan akurat dalam praktik klinis.
5. “*Automatic Localization of Five Relevant Dermoscopic Structures Based on YOLOv8 for Diagnosis Improvement*” (Chabi Adjobo et al., 2023). Penelitian ini bertujuan untuk meningkatkan deteksi fitur dermoskopis pada lesi kulit secara otomatis melalui algoritma YOLOdermoscopic-features. Metode tersebut melibatkan pembuatan anotasi, penggunaan model YOLOv8, dan pelatihan model untuk lima fitur dermoskopis. Eksperimen dilakukan pada *dataset* ISIC 2018 task2. Hasil evaluasi menunjukkan performa yang memuaskan, dengan koefisien kesamaan Dice, koefisien kesamaan Jaccard, presisi, *recall*, dan presisi rata-rata masing-masing mencapai 0.9758, 0.954, 0.9724, 0.938, dan 0.9692. Algoritma ini mengungguli metode lain dalam koefisien kesamaan Jaccard, menunjukkan kesamaan yang lebih baik dengan anotasi spesialis, serta mengatasi masalah kurangnya anotasi fitur dalam *dataset*.
6. “*YOLOv8’s head-layer Performance Comparison for Skin Cancer Detection*”(Sutaji & Yildiz, 2023). Penelitian ini mengevaluasi kinerja YOLOv8 dalam deteksi kanker kulit dengan mempertimbangkan tiga lapisan jalur kepala berbeda pada *dataset* HAM10000. Hasil menunjukkan bahwa jalur

deteksi objek besar menghasilkan kinerja terbaik dan lebih cepat dibandingkan dengan jalur deteksi objek *medium* dan kecil. Penelitian ini merekomendasikan penggunaan jalur deteksi objek besar untuk mendeteksi kanker kulit pada tahap awal. Metode penelitian melibatkan modifikasi lapisan kepala YOLOv8, anotasi lesi kulit dari *dataset* HAM10000, dan pembelajaran transfer serta *fine-tuning*. Namun, keterbatasan yang diidentifikasi termasuk heterogenitas lesi kulit dan perluasan kerjasama dengan dermatolog untuk perbaikan kinerja deteksi.



## **BAB III**

### **ANALISIS DAN PERANCANGAN SISTEM**

#### **3.1. Analisis Sistem**

Analisis sistem bertujuan untuk memahami secara menyeluruh permasalahan yang dihadapi dan kebutuhan sistem yang akan dikembangkan dalam penelitian ini. Bagian ini mencakup identifikasi masalah utama yang menjadi fokus penelitian, analisis kebutuhan sistem dari sisi fungsional dan non-fungsional, serta pemetaan hubungan antar penyebab masalah menggunakan diagram *Ishikawa* (diagram sebab-akibat).

##### *3.1.1. Analisis masalah*

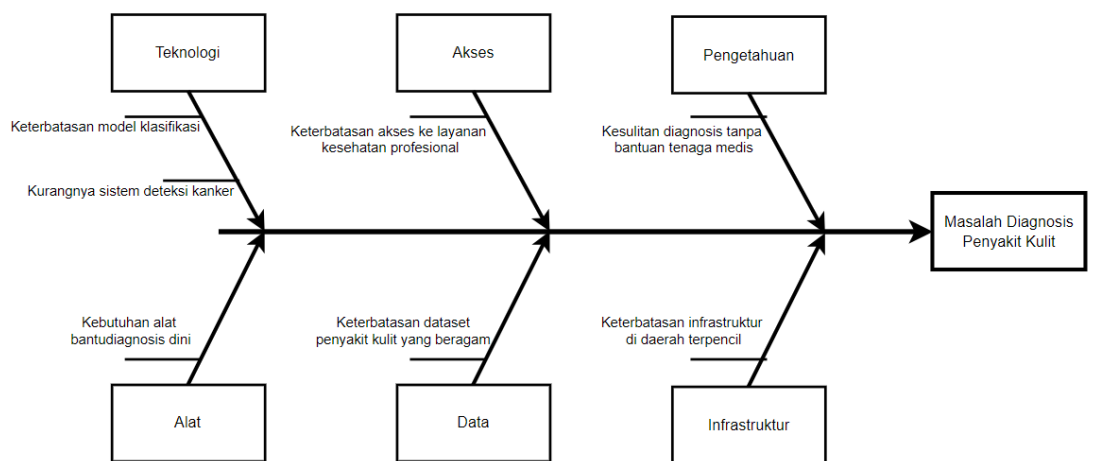
Masalah utama yang dihadapi dalam diagnosis penyakit kulit adalah:

1. Keterbatasan akses ke layanan kesehatan profesional, terutama di daerah terpencil.
2. Kesulitan dalam mendiagnosis penyakit kulit secara akurat tanpa bantuan tenaga medis.
3. Kebutuhan akan alat bantu diagnosis dini yang dapat diakses secara luas.
4. Keterbatasan model klasifikasi yang ada dalam mengenali berbagai jenis penyakit kulit.
5. Kurangnya sistem yang dapat mendeteksi apakah suatu penyakit kulit termasuk kanker atau tidak.

##### *3.1.2. Analisis kebutuhan*

1. Kebutuhan Fungsional :
  - a) Sistem harus mampu menerima input berupa gambar penyakit kulit.
  - b) Sistem harus dapat mengklasifikasikan gambar ke dalam 5 jenis penyakit kulit (*Melanoma, BCC, NV, BKL, dan Seborrheic Keratoses and other Benign Tumors*).
  - c) Sistem harus dapat mendeteksi apakah penyakit kulit termasuk kategori kanker atau tidak.
  - d) Sistem harus memberikan output berupa hasil klasifikasi dan deteksi.

- e) Sistem harus menyediakan antarmuka berbasis website untuk interaksi pengguna.
  - f) Sistem harus mampu menampilkan hasil klasifikasi dan deteksi dalam format yang mudah dipahami.
2. Kebutuhan Non-Fungsional :
- a) Performa: Sistem harus mampu memproses dan memberikan hasil dalam waktu yang relatif cepat.
  - b) Akurasi: Sistem harus mencapai tingkat akurasi yang baik dalam klasifikasi dan deteksi.
  - c) Keamanan: Sistem harus menjamin kerahasiaan data pasien dan gambar yang diunggah.
  - d) *Usability*: Antarmuka website harus intuitif dan mudah digunakan oleh pengguna awam maupun tenaga medis.
  - e) Kompatibilitas: Sistem harus dapat diakses melalui berbagai perangkat dan browser web umum.
  - f) *Maintainability*: Kode dan arsitektur sistem harus mudah dipelihara dan diperbarui.



**Gambar 3.1.** Diagram *Ishikawa*

Diagram *Ishikawa* pada Gambar 3.1 menggambarkan berbagai faktor yang berkontribusi terhadap masalah diagnosis penyakit kulit. Diagram ini membantu untuk memvisualisasikan hubungan antara berbagai penyebab dan masalah utama, sehingga memudahkan dalam mengidentifikasi area-area yang perlu ditangani dalam pengembangan sistem.

### 3.2. Gambaran Umum Sistem

Pada penelitian kali ini, dikembangkan sebuah sistem aplikasi yang mampu melakukan klasifikasi dan deteksi penyakit kulit menggunakan metode EfficientNetB7 dan YOLOv8 berbasis *website*. Sistem ini bertujuan untuk mengklasifikasikan lima jenis penyakit kulit dan mendeteksi apakah gambar kulit tersebut termasuk dalam kategori kanker atau non-kanker.

Langkah pertama dalam perancangan sistem ini adalah mengumpulkan *dataset* yang berisi gambar-gambar kulit dengan berbagai jenis penyakit kulit serta label yang menunjukkan jenis penyakit dan status kanker atau non-kanker. *Dataset* yang dipakai dalam penelitian ini didapatkan dari berbagai sumber di internet.

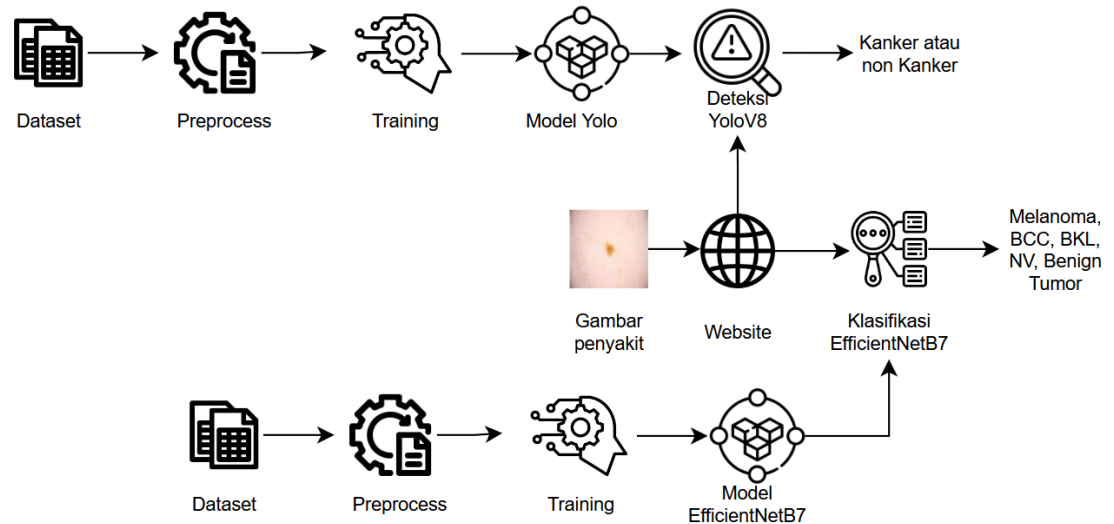
Setelah *dataset* terkumpul, akan dilakukan tahap pra-pemrosesan data, mencakup pembersihan data, augmentasi gambar, dan normalisasi gambar untuk memastikan data siap digunakan dalam proses pembelajaran model. Teknik augmentasi gambar diterapkan untuk memperbesar jumlah data dan meningkatkan variasi gambar guna meningkatkan performa model.

Model klasifikasi dibangun menggunakan metode EfficientNetB7. Model ini biasanya berisi beberapa lapisan konvolusi dan fully connected yang dirancang untuk mengenali pola-pola dalam gambar dan mengklasifikasikannya ke dalam lima jenis penyakit kulit. Proses pelatihan model akan melibatkan pembagian *dataset* menjadi data pelatihan, pengujian, dan validasi, serta penggunaan metrik evaluasi yang telah disebutkan sebelumnya untuk mengukur performa model.

Untuk deteksi kanker kulit, digunakan model YOLOv8 yang terkenal dengan kemampuannya dalam mendeteksi objek pada gambar. Model YOLOv8 dilatih untuk mengenali tanda-tanda kanker pada gambar kulit dan memberikan *output* apakah gambar tersebut termasuk kanker atau non-kanker. Model ini juga dievaluasi menggunakan metrik seperti *mean average precision* (mAP) untuk mengukur akurasi deteksi.

Model klasifikasi dan deteksi yang telah dilatih kemudian diintegrasikan ke dalam aplikasi berbasis *website*. Pengguna dapat mengunggah gambar kulit melalui antarmuka *website* kemudian sistem akan memproses gambar tersebut menggunakan model yang telah dilatih. Hasil klasifikasi dan deteksi kemudian ditampilkan kepada

pengguna bersama dengan deskripsi dan rekomendasi yang relevan. Urutan metode dan alur proses pengembangan sistem yang dilakukan pada studi ini ditampilkan pada Gambar 3.2.



**Gambar 3.2.** Arsitektur Umum Sistem

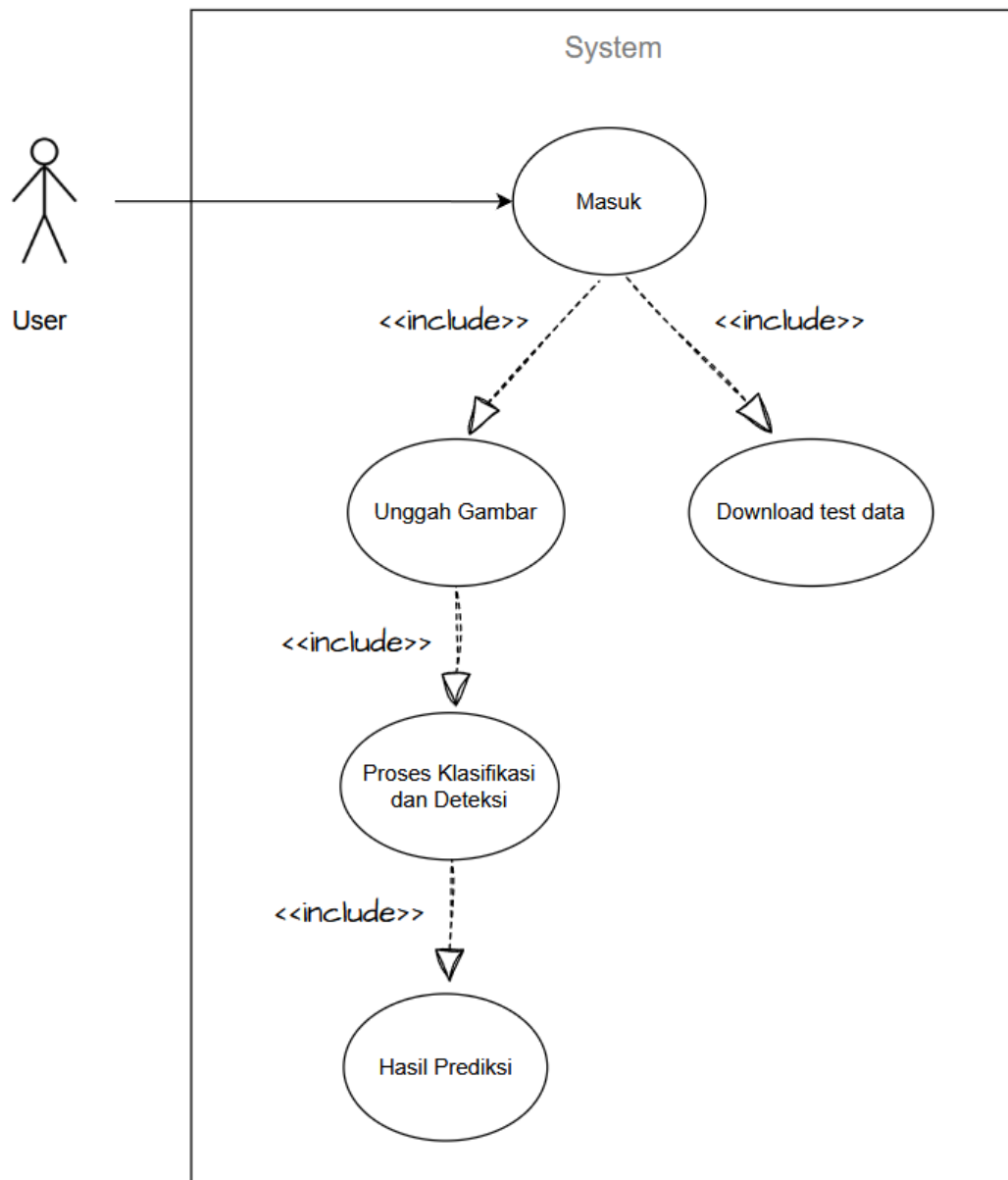
- Melakukan pengumpulan *dataset* gambar penyakit kulit.
- Melakukan *data preprocessing* pada gambar yg sudah dikumpulkan, termasuk pelabelan, pemberian *bounding box*, augmentasi, dan pembagian *dataset* (train, test, dan *validation*).
- Mengimplementasikan model dengan melakukan *training*, *testing*, dan *evaluation*.
- Selanjutnya mengintegrasikan model kedalam *website* sebagai antarmuka pengguna.
- Setelah *website* selesai dibangun, maka sistem sudah dapat digunakan oleh *user* yang membutuhkan.

### 3.3. Pemodelan Sistem

Tahapan ini merupakan cara untuk merepresentasikan sistem yang dibangun. Pemodelan sistem dapat ditampilkan dan dijelaskan melalui berbagai diagram, seperti *use case diagram*, *activity diagram*, *sequence diagram*, serta *flowchart* sistem.

#### 3.3.1. Use case diagram

Diagram ini berperan dalam menggambarkan konsep rancangan sistem, khususnya interaksi antara *user* dengan sistem yang direncanakan.

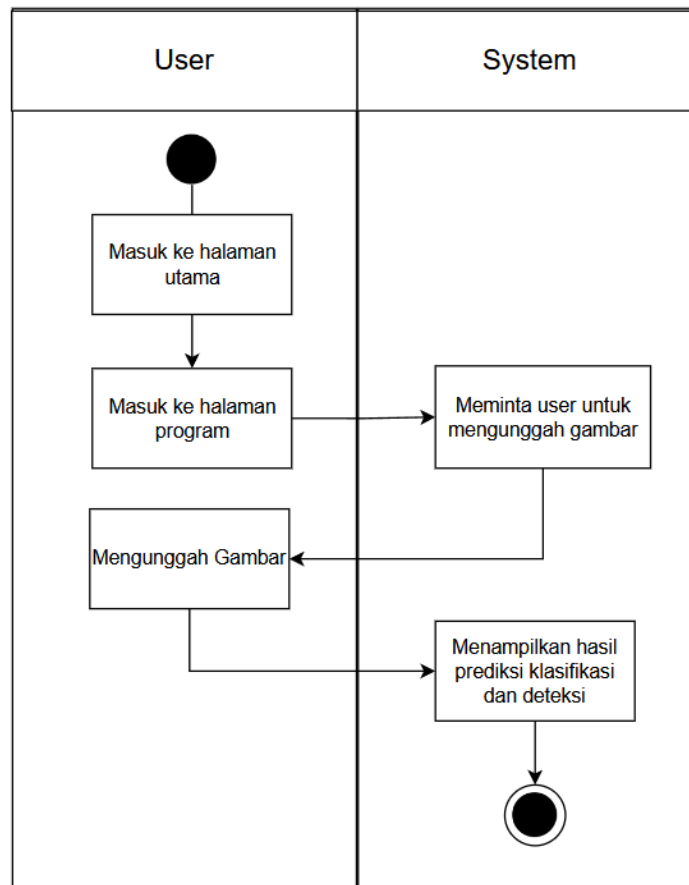


**Gambar 3. 3.** *Use case diagram*

Gambar 3.3 menyajikan diagram use case dari sistem klasifikasi dan deteksi penyakit kulit berbasis *website*. Diagram ini menggambarkan interaksi antara *user* dengan sistem yang direncanakan. Proses dimulai ketika *user* masuk kedalam halaman utama *website*. Setelah itu, *user* dapat mengunggah gambar kulit yang ingin didiagnosis melalui fitur Unggah Gambar. Gambar yang diunggah kemudian diproses oleh sistem menggunakan metode klasifikasi dan deteksi yang terintegrasi, yaitu EfficientNetB7 untuk klasifikasi dan YOLOv8 untuk deteksi. Hasil dari proses tersebut, yaitu prediksi klasifikasi dan deteksi kemudian ditampilkan kepada *user* pada halaman Hasil Prediksi.

### 3.3.2. Activity diagram

Diagram ini berfungsi untuk menunjukkan alur aktivitas dalam rancangan sistem, mulai dari tahap awal saat *website* dijalankan, kemudian melanjutkan ke proses yang dilakukan, hingga mencapai tahap akhir atau penyelesaian.



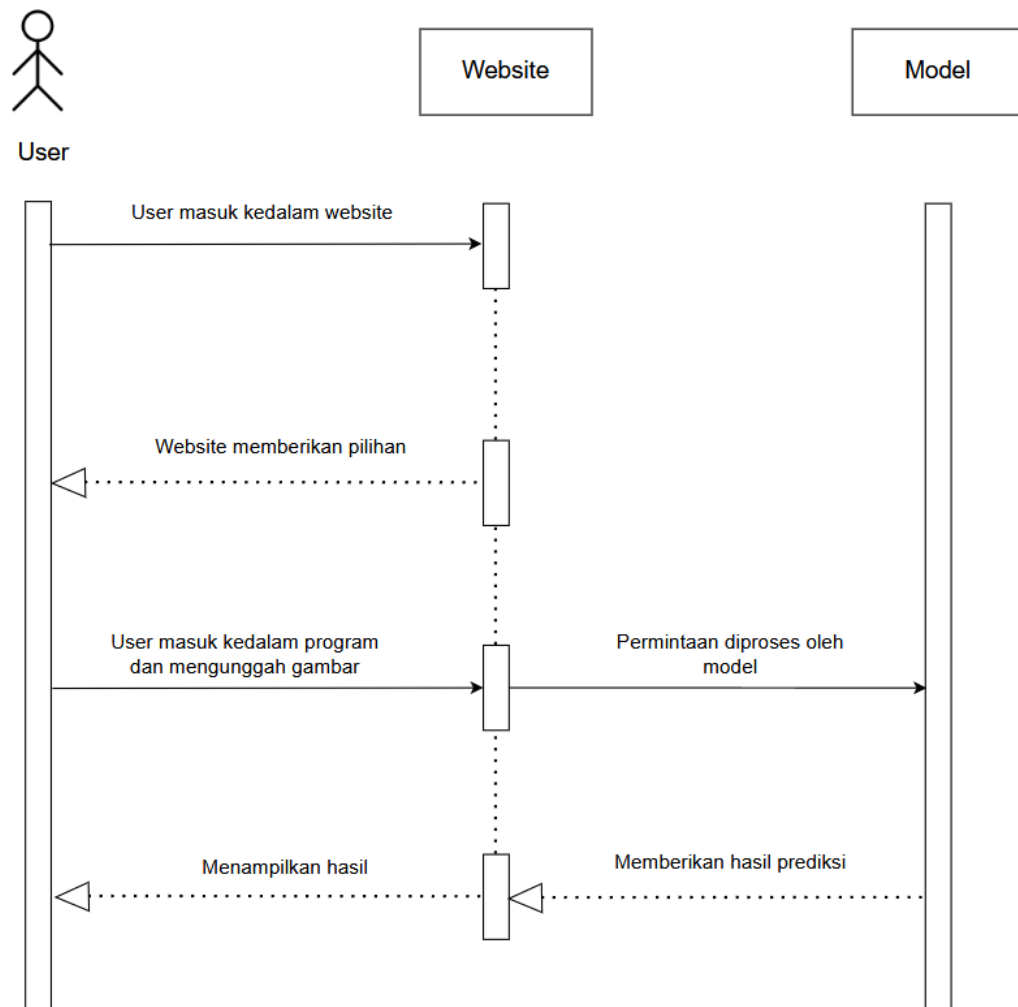
**Gambar 3. 4.** Activity diagram

Gambar 3.4 memaparkan alur dari aktivitas dalam sistem yang dirancang, mulai dari *user* yang mengakses halaman utama *website*. Setelah mengakses halaman utama, *user* diarahkan untuk masuk ke halaman program. Pada tahap ini, sistem meminta *user* untuk mengunggah gambar yang ingin diproses. Setelah gambar diunggah, sistem kemudian memprosesnya dan menampilkan hasil prediksi klasifikasi dan deteksi. Diagram ini menggambarkan urutan kegiatan dari awal hingga akhir yang terlibat dalam proses penggunaan sistem.

### 3.3.3. Sequence diagram

Diagram ini berfungsi untuk memvisualisasikan interaksi antar suatu objek atau komponen dalam sistem dalam urutan waktu yang spesifik. Diagram ini sangat

berguna untuk memperlihatkan bagaimana objek-objek saling berkomunikasi dalam konteks suatu skenario atau proses tertentu.



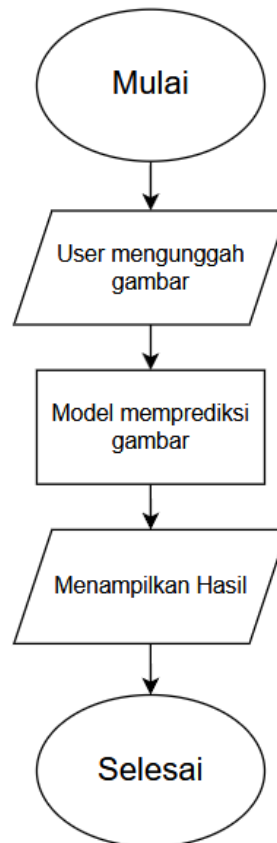
**Gambar 3. 5.** *Sequence diagram*

Gambar 3.5 menyajikan interaksi antara tiga komponen utama dalam sistem: *user*, *website*, dan *model*. Diagram ini memaparkan urutan proses dimulai dari *user* yang mengakses *website*, kemudian *website* memberikan pilihan kepada *user*. Setelah itu, *user* memasuki program dan mengunggah gambar, yang kemudian diproses oleh *model*. Setelah proses selesai, hasil prediksi dikembalikan oleh *model* dan ditampilkan oleh *website* kepada *user*. Diagram ini menggambarkan alur komunikasi yang terjadi antara komponen-komponen tersebut selama proses berlangsung.

### 3.3.4. Flowchart

Diagram ini digunakan untuk memvisualisasikan langkah-langkah dalam suatu proses secara urut. Dengan *flowchart*, kita bisa melihat bagaimana suatu proses

dimulai, keputusan apa saja yang perlu diambil, dan bagaimana proses itu berakhir. Diagram ini sangat membantu untuk memahami alur kerja atau prosedur secara lebih mudah dan jelas.



**Gambar 3. 6.** *Flowchart Sistem*

Gambar 3.6 menunjukkan alur proses dalam sistem yang dimulai dari pengguna yang mengunggah gambar. Setelah gambar diunggah, model kemudian memproses dan memprediksi gambar tersebut. Hasil prediksi kemudian ditampilkan kepada pengguna. *Flowchart* ini menggambarkan tahapan-tahapan yang terjadi dari awal proses hingga selesai, memudahkan dalam memahami bagaimana sistem bekerja secara keseluruhan.



### 3.4. Pengambilan *Dataset*

Pada studi ini, data yang digunakan untuk mengklasifikasikan serta mendeteksi penyakit kulit diperoleh dari *platform* Kaggle yang dapat diakses melalui link berikut:: <https://www.kaggle.com/datasets/ismailpromus/skin-diseases-image-dataset>. *Dataset* ini berisi gambar-gambar penyakit kulit yang dikategorikan ke dalam 10 kelas penyakit kulit yang berbeda.

Namun, untuk keperluan penelitian ini, hanya 5 kelas penyakit kulit yang dipilih, yaitu:

1. *Melanoma*: Ini adalah jenis kanker kulit paling berbahaya yang terjadi pada sel yang memproduksi melanin.
2. *Basal Sel Carcinoma* (BCC): Jenis kanker kulit yang sering terjadi, biasanya disebabkan oleh paparan sinar matahari berlebih.
3. *Melanocytic Nevi* (NV): Bintik-bintik kecil atau tahi lalat yang biasanya jinak.
4. *Benign Keratosis Like Lesion* (BKL): Lesi kulit yang biasanya tidak berbahaya dan mirip dengan keratosis.
5. *Seborrheic Keratoses and other Benign Tumors*: Merupakan tumor kulit jinak yang sering terjadi pada lansia.

Kemudian untuk deteksi, diambil beberapa sampel dari 5 kelas yang telah dipilih untuk klasifikasi di atas. Sampel-sampel ini digunakan untuk melatih model deteksi (YOLOv8) agar dapat mendeteksi apakah gambar tersebut termasuk dalam kategori kanker atau non-kanker.

### 3.5. Pra-pemrosesan Data

Tahapan ini merupakan langkah penting dalam mengembangkan model untuk mengklasifikasikan dan mendeteksi penyakit kulit. Tujuan dari dilakukannya pemrosesan data ini ialah untuk memastikan bahwa data yang digunakan untuk pelatihan dan pengujian sudah optimal. Tahapan yang dilakukan dalam *preprocessing* data pada studi ini ialah :

1. Memuat Data: Data gambar yang diambil dari *Skin Diseases Image Dataset* di Kaggle diimpor ke dalam lingkungan pemrograman. Data ini mencakup gambar dari lima kelas penyakit kulit yang telah dipilih sebelumnya.

2. Anotasi Data : Beberapa sampel gambar dari *Dataset* yang telah dimuat kemudian diambil dan dianotasi menggunakan platform *Roboflow*. Proses anotasi melibatkan pemberian *bounding box* dan label pada gambar untuk mendefinisikan *area* spesifik yang menunjukkan kanker kulit atau non kanker.
3. Resizing: Setiap gambar diubah ukurannya (resizing) ke dimensi yang konsisten untuk memastikan bahwa seluruh gambar memiliki *size* yang sama. Adapun ukuran yang dipilih sesuai dengan persyaratan *input* model EfficientNetB7 dan YOLOv8, yaitu 224x224 piksel untuk EfficientNetB7 dan 640x640 piksel untuk YOLOv8.
4. Normalisasi : Normalisasi dilakukan untuk mengubah nilai piksel gambar agar berada dalam rentang [0, 1]. Ini penting untuk mempercepat konvergensi model selama pelatihan.
5. Augmentasi Data : Augmentasi data dilakukan untuk meningkatkan variasi dalam *dataset* dan membantu mengurangi *overfitting*. Beberapa teknik augmentasi yang digunakan meliputi:
  - *Rotation Range*: Memutar gambar dengan sudut tertentu hingga 20 derajat.
  - *Width Shift Range*: Menggeser gambar secara horizontal sebesar 20% dari lebar gambar.
  - *Height Shift Range*: Menggeser gambar secara vertikal sebesar 20% dari lebar gambar.
  - *Shear Range*: Menerapkan transformasi geser (shear) hingga 20%.
  - *Zoom Range*: Memperbesar atau memperkecil gambar hingga 20%.
  - *Horizontal Flip*: Membalik gambar secara horizontal.
  - *Fill Mode*: Mengisi *area* kosong dengan metode '*nearest*'.

#### 6. Pembagian *Dataset*

Setelah itu, data kemudian dibagi menjadi tiga set:

- *Training Set*: Data yang dipakai untuk melatih model klasifikasi dan deteksi.
- *Validation Set*: Data yang dipakai untuk mengevaluasi kinerja model selama proses pelatihan.
- *Testing Set*: Data yang dipakai untuk menguji kinerja akhir dari model setelah pelatihan selesai.

### 3.6. Pembangunan Model

Pada bagian ini dilakukan pembangunan model untuk klasifikasi dan deteksi penyakit kulit menggunakan pendekatan yang telah ditentukan sebelumnya, yaitu EfficientNetB7 untuk klasifikasi dan YOLOv8 untuk deteksi. Berikut adalah langkah-langkah yang dilakukan dalam pembangunan model:

#### 1. Pembangunan Model Klasifikasi (EfficientNetB7):

- Instalasi Paket: Install paket yang diperlukan untuk membangun model, contohnya seperti tensorflow.
- *Input Layer*: Disesuaikan dengan dimensi yang dibutuhkan oleh EfficientNetB7 untuk menerima gambar dalam resolusi tertentu.
- Penggunaan *Pre-trained Model*: Menggunakan EfficientNetB7 yang telah dilatih sebelumnya pada *dataset ImageNet* untuk memanfaatkan representasi fitur yang kuat.
- Penyesuaian Layer: Menambahkan layer-layer klasifikasi seperti *Dense layer* dengan fungsi aktivasi yang sesuai, seperti ReLU untuk non-linearitas.
- Regularisasi: Menambahkan teknik regularisasi seperti *dropout* untuk menghindari *overfitting* pada model.
- *Output Layer*: Menggunakan *softmax* layer untuk menghasilkan probabilitas distribusi kelas.
- *Compile*: Kemudian model di *Compile* menggunakan *optimizer Adam*, *loss function categorical\_crossentropy*, dan *metrics evaluasi* yang telah disebutkan sebelumnya.

#### 2. Pembangunan Model Deteksi (YOLOv8):

- Instalasi Paket : Install paket Ultralytics untuk menggunakan model YOLOv8.
- Pengaturan Konfigurasi *Dataset*: Siapkan file konfigurasi *data.yaml*, yang mendefinisikan path ke direktori data pelatihan, validasi, dan pengujian.
- Inisialisasi Model YOLOv8: Muat model YOLOv8 yang sesuai, misalnya YOLOv8m (*medium*), yang dapat digunakan untuk deteksi objek.

### 3.7. Pelatihan Model

Sesudah model selesai dibangun, langkah berikutnya adalah melatihnya untuk meningkatkan kinerjanya. Berikut adalah rincian dari proses pelatihan model.

#### 3.7.1. Pelatihan model klasifikasi

1. Inisialisasi Model : model EfficientNetB7 yang telah dibangun diinisialisasi dengan parameter awal. Ini termasuk pengaturan layer klasifikasi yang ditambahkan, seperti *dense layer* dengan fungsi aktivasi ReLU dan *output layer* dengan *softmax*.
2. Pengaturan Hyperparameter
  - *Learning Rate*: 0.001, *learning rate* dapat disesuaikan selama pelatihan untuk memastikan konvergensi yang baik. *Learning rate* dapat diubah secara iteratif untuk meningkatkan performa model.
  - *Batch Size*: 16, pilih ukuran *batch* yang sesuai untuk pelatihan. Semakin besar ukurannya, maka stabilitas pelatihan akan semakin meningkat, tetapi memerlukan lebih banyak memori.
  - Jumlah *Epoch*: 15, tetapkan jumlah *epoch* yang cukup untuk pelatihan model.
3. Penggunaan *Callback*
  - *ReduceLROnPlateau*  
*Callback* ini digunakan untuk mengurangi *learning rate* secara otomatis ketika metrik yang dimonitor tidak menunjukkan perbaikan. Hal ini membantu model untuk melakukan *fine-tuning* saat sudah mendekati konvergensi.
  - *ModelCheckpoint*  
*Callback* ini menyimpan model pada interval tertentu atau ketika metrik kinerja terbaik tercapai selama pelatihan. Ini memastikan peneliti dapat menyimpan model dengan performa terbaik dan menghindari kehilangan kemajuan pelatihan.
  - *EarlyStopping*  
*Callback* ini menghentikan pelatihan jika tidak ada perbaikan dalam metrik validasi setelah sejumlah *epoch* tertentu. Ini membantu mencegah *overfitting* dan mengurangi waktu pelatihan yang tidak perlu.

4. Pelatihan Model: Model kemudian dilatih dengan menggunakan data latih serta data validasi yang telah disiapkan sebelumnya. Selama pelatihan, model memperbarui bobotnya untuk meminimalkan fungsi kerugian sambil memonitor metrik kinerja.
5. Penyimpanan Model: Model yang telah dilatih disimpan untuk digunakan pada tahap selanjutnya, yaitu implementasi dalam sistem deteksi dan klasifikasi berbasis *website*.

### 3.7.2. Pelatihan model deteksi

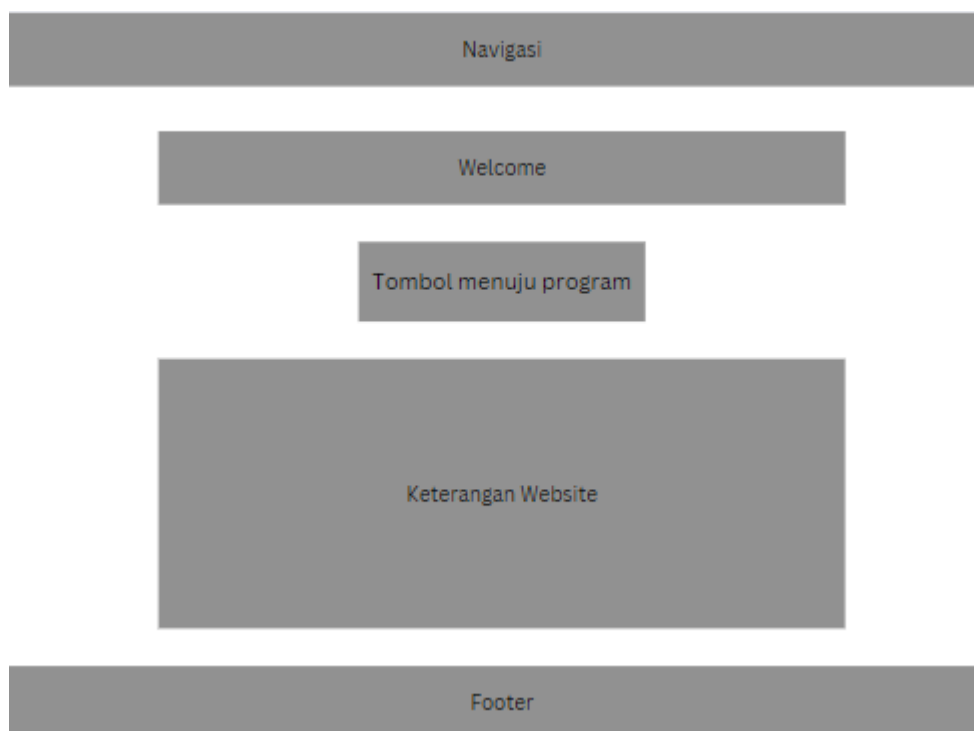
1. Inisialisasi Model : Model YOLOv8 diinisialisasi dengan parameter awal yang telah ditetapkan. Ini termasuk memilih model *pre-trained* YOLOv8m (*medium*) yang digunakan sebagai titik awal pelatihan.
2. Pengaturan Hyperparameter
  - *Mode*: Setel *Mode* ke *train* untuk memulai pelatihan model. Ini mengarahkan YOLOv8 untuk menjalankan pelatihan pada *dataset* yang telah disiapkan.
  - *Learning Rate*: 0.0001, ini mempengaruhi seberapa besar langkah model dalam memperbarui bobot selama pelatihan.
  - *Patience*: 10, ini menentukan jumlah *epoch* yang harus dilalui tanpa peningkatan signifikan dalam metrik yang dimonitor (biasanya *loss*) sebelum *learning rate* dikurangi atau pelatihan dihentikan.
  - *Batch Size*: 16, pilih ukuran *batch* yang optimal untuk pelatihan model deteksi. Ukuran *batch* yang lebih besar dapat membantu model mempelajari deteksi objek dengan lebih baik tetapi membutuhkan lebih banyak memori. Ukuran *batch* biasanya disesuaikan berdasarkan kapasitas GPU dan *dataset*.
  - Jumlah *Epoch*: 40, ini adalah jumlah iterasi penuh melalui seluruh *dataset*. Pastikan jumlah *epoch* cukup untuk model belajar dari data tanpa *overfitting*.
  - *Dropout*: 0.15, terapkan *dropout* untuk mengurangi risiko *overfitting*. *Dropout* membuat model tidak terlalu bergantung pada fitur tertentu dengan menonaktifkan neuron secara acak selama pelatihan.
3. Penyimpanan Model : Model yang telah dilatih disimpan untuk digunakan pada tahap selanjutnya, yaitu implementasi dalam sistem deteksi dan klasifikasi berbasis *website*.

### 3.8. Evaluasi dan Pengujian Model

Setelah model selesai dilatih, maka selanjutnya ialah melakukan evaluasi untuk memastikan kinerja model. Evaluasi model ini dilakukan selama proses validasi dan pengujian menggunakan *dataset* validasi dan *testing*. Proses evaluasi ini sangat penting untuk menilai sejauh mana model dapat menerapkan pengetahuan pada data yang belum pernah dihadapinya sebelumnya.

### 3.9. Perancangan Sistem Berbasis *Website* Sebagai Antarmuka

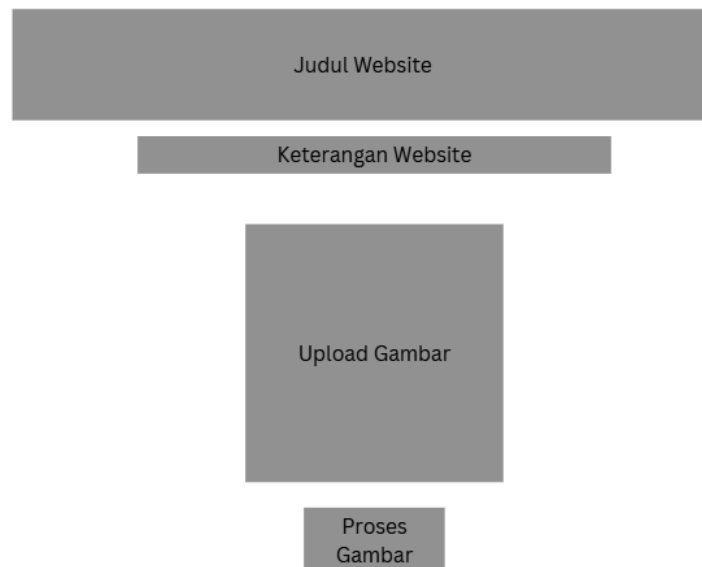
#### 3.9.1. Halaman utama



**Gambar 3.7.** Rancangan Halaman Utama

Gambar 3.6 memperlihatkan rancangan *website* untuk halaman utama. Pada halaman ini, pengguna dapat melihat keterangan informasi mengenai *website* ini, termasuk fitur-fitur utama, tujuan, serta berbagai layanan yang disediakan. Halaman ini dimaksudkan untuk memberikan pemahaman terkait *website* tersebut.

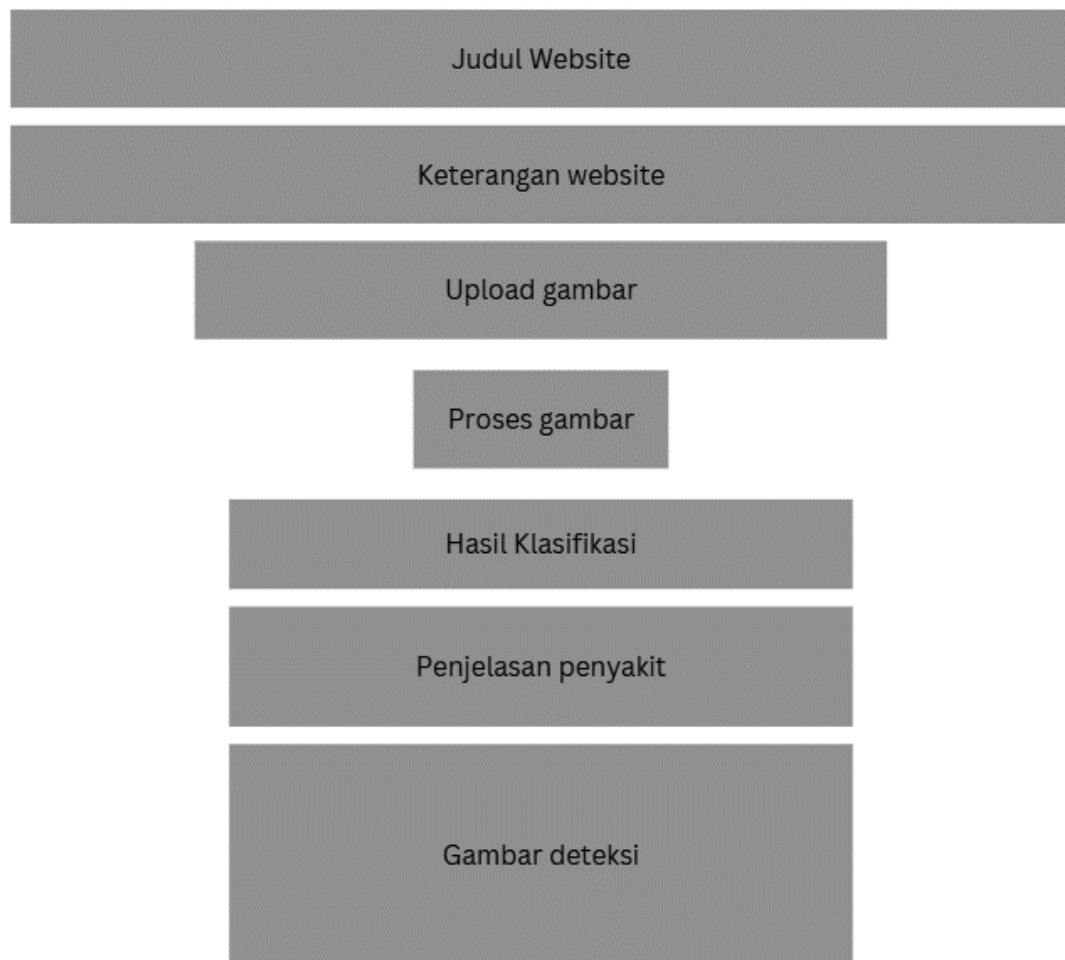
### 3.9.2. Halaman prediksi



**Gambar 3.8.** Rancangan Halaman Prediksi Gambar

Pada Gambar 3.7, ditunjukkan antarmuka *website* untuk halaman prediksi gambar. Pada halaman ini, pengguna dapat mengunggah gambar yang menunjukkan gejala penyakit kulit dan kemudian menekan tombol “Proses” untuk memperoleh hasil prediksi mengenai jenis penyakit kulit yang ada pada gambar tersebut.

### 3.9.3. Halaman hasil prediksi



**Gambar 3.9.** Rancangan Halaman Hasil Prediksi

Pada Gambar 3.8, ditampilkan antarmuka *website* untuk halaman hasil prediksi gambar penyakit kulit. Halaman ini menyajikan hasil klasifikasi dari gambar yang diunggah. Selain itu, halaman ini juga menampilkan hasil deteksi gambar untuk menentukan apakah gambar tersebut termasuk dalam kategori kanker atau bukan.



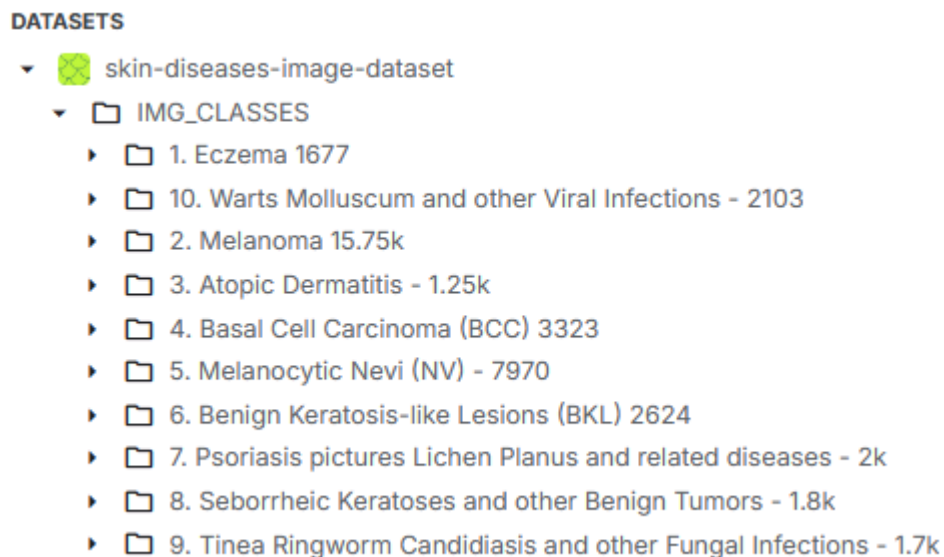
## BAB IV

### IMPLEMENTASI DAN PENGUJIAN SISTEM

#### 4.1. Implementasi Tahap Pengambilan *Dataset*





##### 4.1.1. Pengambilan data klasifikasi

*Dataset* yang digunakan untuk klasifikasi adalah *Skin Diseases Image Dataset*, yang diambil dari platform Kaggle. *Dataset* ini dapat diakses melalui tautan berikut: <https://www.kaggle.com/Datasets/ismailpromus/skin-diseases-image-Dataset>. *Dataset* ini terdiri dari 40.000 gambar yang berisi 10 kelas penyakit kulit sebagaimana ditampilkan pada Gambar 4.1.



**Gambar 4. 1.** *Dataset Skin Disease*

Seluruh data gambar dalam *dataset* ini dapat digunakan secara langsung pada Kaggle *Environment*. Untuk keperluan penelitian ini, digunakan 5 kelas dari 10 kelas yang tersedia, kemudian diambil 1000 gambar untuk masing masing kelas dengan total gambar sebanyak 5000 untuk 5 kelas, seperti pada Gambar 4.2 di halaman 34.

 BCC	8/30/2024 10:53 AM	File folder
 BKL	8/30/2024 10:53 AM	File folder
 Melanoma	8/30/2024 10:52 AM	File folder
 NV	8/30/2024 10:53 AM	File folder
 Seborrheic Keratoses and other Benign Tumors	8/30/2024 10:52 AM	File folder

**Gambar 4. 2.** Kelas Penyakit Kulit yang Digunakan

#### 4.1.2. Pengambilan data deteksi

Untuk tugas deteksi, data yang digunakan berasal dari *Skin Diseases Image Dataset* yang sama dengan *dataset* klasifikasi. Dimana sampel dari data klasifikasi kemudian diunduh untuk masing masing kelas dengan total 3000 data gambar yang berisi 5 kelas, kemudian dari 5 kelas tersebut dibagi menjadi kelas kanker dan non kanker, setelah itu dilakukan pra-pemrosesan.

## 4.2. Implementasi Tahap Pra-pemrosesan Data

#### 4.2.1. Pra pemrosesan data klasifikasi

Pada tahap ini, dalam tugas klasifikasi gambar, dilakukan beberapa langkah untuk mempersiapkan *dataset* yang dipakai dalam pelatihan model.:

##### 1. Pemilihan dan Penyaringan Folder

*Dataset* gambar terdiri dari berbagai jenis lesi kulit yang dikelompokkan ke dalam beberapa folder. Untuk memastikan relevansi data, hanya folder-folder tertentu yang dipilih untuk diproses lebih lanjut. Folder yang dipilih meliputi:

- *Melanoma*
- *Basal Sel Carcinoma (BCC)*
- *Melanocytic Nevi (NV)*
- *Benign Keratosis-like Lesions (BKL)*
- *Seborrheic Keratoses and other Benign Tumors.*

Folder-folder tersebut kemudian disalin dari direktori sumber ke direktori target seperti pada Gambar 4.3 di halaman 35.

```

# Path direktori IMG_CLASSES
img_classes_dir = "/kaggle/input/skin-diseases-image-dataset/IMG_CLASSES"
# Path untuk menyimpan folder yang akan dipertahankan
target_dir = "/kaggle/working/selected_folders"

# Membuat direktori target jika belum ada
os.makedirs(target_dir, exist_ok=True)

# Iterasi melalui setiap folder di dalam IMG_CLASSES
for folder_name in os.listdir(img_classes_dir):
    folder_path = os.path.join(img_classes_dir, folder_name)
    # Periksa apakah folder harus dipertahankan
    if folder_name in folders_to_keep and os.path.isdir(folder_path):
        # Salin folder ke direktori target
        shutil.copytree(folder_path, os.path.join(target_dir, folder_name))
        print(f"Folder {folder_name} disalin.")
    else:
        print(f"Folder {folder_name} tidak disalin.")

# Print pesan setelah selesai
print("Proses selesai. Folder yang dipilih telah disalin ke direktori target.")

```

**Gambar 4. 3.** Kode Penyaringan Folder

## 2. Pembagian *Dataset*

Data yang telah disalin selanjutnya dibagi menjadi tiga subset: *training*, *validation*, dan *testing*. Pembagian ini dilakukan dengan rasio 80% untuk *training*, 10% untuk *validation*, dan 10% untuk *testing* menggunakan *library splitfolders*. Proses pembagian *dataset* ditampilkan pada Gambar 4.4.

```

[ ]: splitfolders.ratio(base_dir, seed=0, output="skin-splitted", ratio=(0.8, 0.1, 0.1))

[ ]: splitted_dir = "/kaggle/working/skin-splitted"

[ ]: train_dir = "/kaggle/working/skin-splitted/train"
    val_dir = "/kaggle/working/skin-splitted/val"
    test_dir = "/kaggle/working/skin-splitted/test"

```

**Gambar 4. 4.** Kode Pembagian *Dataset*

### 3. Pengaturan Generator Data

Untuk memproses gambar selama pelatihan, validasi, dan pengujian, digunakan *ImageDataGenerator* dari Tensorflow. Konfigurasi generator data adalah sebagai berikut:

- Generator Data Pelatihan: Menerapkan augmentasi pada gambar pelatihan, termasuk rotasi, pergeseran, pemotongan, zoom, dan *flip horizontal*, untuk meningkatkan variasi dan ketahanan model. Kode konfigurasi generator data pelatihan ditampilkan pada Gambar 4.5.

```
# Konfigurasi Data Generator untuk augmentasi
img_height = 224
img_width = 224
batch_size = 16
# epochs = 100

# Membuat generator untuk data pelatihan
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,|
    fill_mode='nearest')

train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical')
```

**Gambar 4. 5.** Konfigurasi Generator Data Pelatihan

- Generator Data Validasi dan Pengujian: Menggunakan *rescaling* tanpa augmentasi untuk data validasi dan pengujian. Kode konfigurasi generator data validasi dan pengujian disajikan pada Gambar 4.6 di halaman 37.

```

# Membuat generator untuk data validasi tanpa augmentasi
validation_datagen = ImageDataGenerator(rescale=1./255)

validation_generator = validation_datagen.flow_from_directory(
    val_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical')

# Membuat generator untuk data test tanpa augmentasi
test_datagen = ImageDataGenerator(rescale=1./255)

test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical',
    shuffle = False,
)

```

**Gambar 4. 6.** Konfigurasi Generator Data Validasi dan Pengujian

#### 4.2.2. Pra-pemrosesan data deteksi

Untuk tahap ini menggunakan YOLOv8, *dataset* yang digunakan diambil dari data pengujian klasifikasi. Proses ini melibatkan beberapa langkah penting yang meliputi pelabelan, augmentasi, dan konfigurasi *dataset*. Berikut adalah langkah-langkah yang dilakukan:

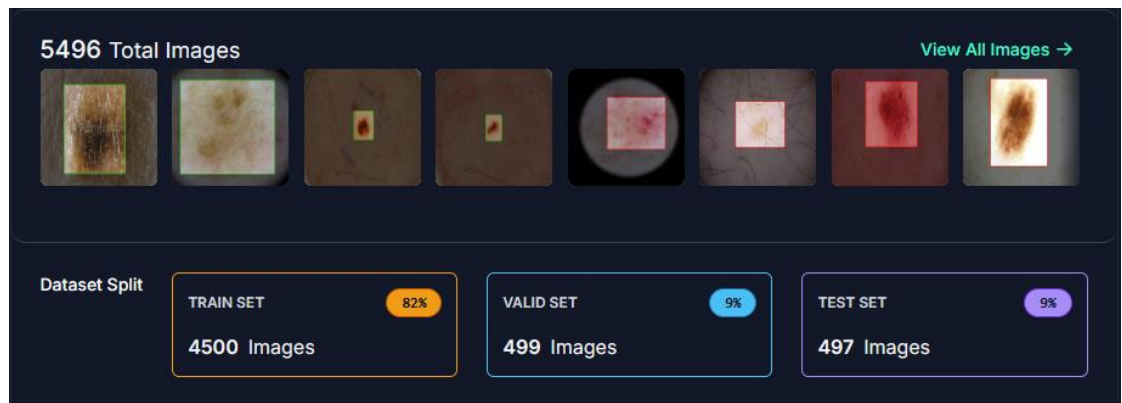
1. Pengambilan dan Pelabelan Data

*Dataset* yang sudah dipersiapkan tadi kemudian diberi label dan *bounding box* secara manual menggunakan platform *Roboflow*. *Roboflow* menyediakan fitur augmentasi yang mendukung pengolahan gambar secara otomatis.

2. Pembagian dan Augmentasi *Dataset*

*Dataset* yang telah dilabeli dan diberi *bounding box* kemudian di augmentasi. Augmentasi yang diterapkan, meliputi : *Flip Horizontal* dan *Vertical*, Rotasi: antara  $-15^{\circ}$  hingga  $+15^{\circ}$ , *Shear* :  $10^{\circ}$  *Vertical* dan *Horizontal*, *Brightness* : diantara  $-20\%$  dan  $20\%$ , *Noise* : up to  $0.5\%$  of pixel, dan

*Resize* gambar ke ukuran 640x640 piksel. Setelah itu data dibagi menjadi tiga subset: pelatihan, validasi, dan pengujian, seperti pada Gambar 4.7.



**Gambar 4. 7.** Pembagian Dataset Deteksi

*Dataset* ini kemudian diunduh dan disesuaikan dengan konfigurasi pada file `data.yaml`. Konfigurasi `data.yaml` disajikan pada Gambar 4.8.

```
import yaml

# Path to the data.yaml file
data_yaml_path = '/kaggle/working/skin-disease-new-12/data.yaml'

# Load data.yaml
with open(data_yaml_path, 'r') as file:
    data = yaml.safe_load(file)

# Ganti direktori val, test, dan train
data['test'] = '/kaggle/working/skin-disease-new-12/test/images' # Ganti dengan direktori test baru
data['train'] = '/kaggle/working/skin-disease-new-12/train/images' # Ganti dengan direktori train baru
data['val'] = '/kaggle/working/skin-disease-new-12/valid/images' # Ganti dengan direktori val baru

# Write back to data.yaml
with open(data_yaml_path, 'w') as file:
    yaml.dump(data, file)

print("Direktori val, test, dan train telah diubah di data.yaml.")
```

Direktori val, test, dan train telah diubah di `data.yaml`.

**Gambar 4. 8.** Konfigurasi `data.yaml`

### 4.3. Implementasi Tahap Pembangunan Model

Setelah melalui tahap pra-pemrosesan data, selanjutnya ialah tahap pembangunan model klasifikasi dan deteksi. Dimana tahap pembangunan model ini menggunakan *library* Tensorflow untuk klasifikasi, dan Ultralytics untuk deteksi.

#### 4.3.1. Model klasifikasi

1. Import *library* yang diperlukan seperti yang ditunjukkan pada Gambar 4.9 di halaman 39.

```
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.applications import EfficientNetB7
from tensorflow.keras.optimizers import Adam, Adamax
from tensorflow.keras.callbacks import ModelCheckpoint, ReduceLROnPlateau
```

**Gambar 4. 9.** Import *Library* Tensorflow

## 2. Membangun model dengan EfficientNetB7

EfficientNetB7 digunakan sebagai model dasar tanpa lapisan atas, sebagaimana terlihat pada Gambar 4.10.

```
# Membuat model dasar (base model)
base_model = EfficientNetB7(input_shape=(img_height, img_width, 3),
                           include_top=False,
                           weights='imagenet')

base_model.trainable = True
```

**Gambar 4. 10.** Model Dasar EfficientNetB7

Setelah *base model* dibangun, selanjutnya ialah menambahkan beberapa lapisan diatas model dasar dan menggunakan fungsi aktivasi *softmax* untuk lapisan *output*, seperti pada Gambar 4.11.

```
from tensorflow.keras import layers

# Gunakan output dari base_model sebagai input untuk layer berikutnya
x = base_model.output
x = layers.GlobalAveragePooling2D()(x) # Menambahkan Global Average Pooling
x = layers.Dropout(0.5)(x)
x = layers.Dense(256, activation="relu", kernel_regularizer=tf.keras.regularizers.l2(0.001))(x) #
Menambahkan regularisasi L2
x = layers.Dropout(0.5)(x)
outputs = layers.Dense(5, activation='softmax')(x) # Layer output dengan aktivasi softmax
model = tf.keras.Model(inputs=base_model.input, outputs=outputs) # Menggunakan input dari base_model
model.summary()
```

**Gambar 4. 11.** Menambahkan Lapisan Model

Kemudian model dikompilasi dengan *optimizer* Adam dan fungsi *loss categorical\_crossentropy*.

#### 4.3.2. Model deteksi

1. Import *library* yang diperlukan, sebagaimana ditunjukkan pada Gambar 4.12.

```
from ultralytics import YOLO
```

**Gambar 4. 12.** Import *library* Ultralytics

2. Selanjutnya ialah memanggil model YOLOv8 dari *library* Ultralytics, YOLO memiliki beberapa model seperti YOLOv8n, YOLOv8s, dll yang memiliki bobot parameter berbeda, untuk kebutuhan penelitian ini digunakan YOLOv8m.
3. Setelah itu dilakukan beberapa konfigurasi parameter seperti *epochs*, *imagesize*, *learning rate*, dll sesuai dengan yang diperlihatkan Gambar 4.13.

```
In [7]: # Training model
!yolo task = detect mode = train model = yolov8m.pt data={dataset.location}/data.yaml epochs = 40 imgsz=640 lr0=0.0001 patience = 10 optimizer="Adam" verbose=True dropout = 0.15
```

**Gambar 4. 13.** Konfigurasi Parameter YOLOv8

### 4.4. Implementasi Tahap Pelatihan Model

Setelah model selesai dibangun, selanjutnya ialah melakukan tahap pelatihan model untuk klasifikasi dan Deteksi. Pada tahap ini, model dilatih dengan skenario pembagian data *train*, *test*, dan *validation* yang berbeda untuk menilai performa secara komprehensif yaitu (8:1:1), (7:1.5:1.5), dan (5:2.5:2.5). Evaluasi dilakukan dengan menilai performa model melalui metrik *accuracy*, *precision*, *recall*, *F1-score*, serta *Mean average precision* (mAP) untuk deteksi objek.

#### 4.4.1. Pelatihan model klasifikasi

Model klasifikasi dilatih menggunakan data latih dan data validasi dengan jumlah *epochs* sebesar 15, kemudian digunakan *callbacks* seperti *ReduceLROnPlateau* dan *EarlyStopping* dari *tensorflow.keras.callbacks* untuk mengelola laju pembelajaran dan menghentikan pelatihan secara dini jika kriteria tertentu terpenuhi.



```

Epoch 12/100
250/250 ————— 124s 488ms/step - F1Score: 0.9287 - Precision: 0.9370 - Recall: 0.9253 - accuracy: 0.9300 - loss:
0.2858 - val_F1Score: 0.9336 - val_Precision: 0.9393 - val_Recall: 0.9280 - val_accuracy: 0.9340 - val_loss: 0.3235 - learni
ng_rate: 0.0010
Epoch 13/100
250/250 ————— 124s 488ms/step - F1Score: 0.9311 - Precision: 0.9343 - Recall: 0.9260 - accuracy: 0.9309 - loss:
0.2727 - val_F1Score: 0.8310 - val_Precision: 0.8424 - val_Recall: 0.8340 - val_accuracy: 0.8400 - val_loss: 0.5597 - learni
ng_rate: 0.0010
Epoch 14/100
250/250 ————— 124s 487ms/step - F1Score: 0.9397 - Precision: 0.9471 - Recall: 0.9342 - accuracy: 0.9403 - loss:
0.2660 - val_F1Score: 0.9062 - val_Precision: 0.9131 - val_Recall: 0.9040 - val_accuracy: 0.9040 - val_loss: 0.3535 - learni
ng_rate: 0.0010
Epoch 15/100
250/250 ————— 0s 471ms/step - F1Score: 0.9546 - Precision: 0.9571 - Recall: 0.9502 - accuracy: 0.9541 - loss:
0.2057
Training stopped as both training and validation accuracy reached 95% or above.
250/250 ————— 124s 487ms/step - F1Score: 0.9546 - Precision: 0.9571 - Recall: 0.9502 - accuracy: 0.9541 - loss:
0.2056 - val_F1Score: 0.9543 - val_Precision: 0.9540 - val_Recall: 0.9540 - val_accuracy: 0.9540 - val_loss: 0.1696 - learni
ng_rate: 1.0000e-04

```

**Gambar 4. 14.** Proses Pelatihan Model (8:1:1)

Pada Gambar 4.14 disajikan proses pelatihan model untuk skenario 1, yaitu perbandingan data (8:1:1). Kemudian didapatkan *accuracy training* = 0.9541 dan *validation* = 0.9540

```

Epoch 12/15
219/219 ————— 114s 514ms/step - F1Score: 0.9631 - Precision: 0.9652 - Recall: 0.
9618 - accuracy: 0.9633 - loss: 0.1756 - val_F1Score: 0.9431 - val_Precision: 0.9439 - val_Rec
all: 0.9427 - val_accuracy: 0.9427 - val_loss: 0.2542 - learning_rate: 1.0000e-04
Epoch 13/15
219/219 ————— 115s 514ms/step - F1Score: 0.9706 - Precision: 0.9751 - Recall: 0.
9686 - accuracy: 0.9710 - loss: 0.1541 - val_F1Score: 0.9445 - val_Precision: 0.9453 - val_Rec
all: 0.9440 - val_accuracy: 0.9440 - val_loss: 0.2462 - learning_rate: 1.0000e-05
Epoch 14/15
219/219 ————— 115s 515ms/step - F1Score: 0.9731 - Precision: 0.9747 - Recall: 0.
9716 - accuracy: 0.9730 - loss: 0.1464 - val_F1Score: 0.9470 - val_Precision: 0.9466 - val_Rec
all: 0.9453 - val_accuracy: 0.9467 - val_loss: 0.2421 - learning_rate: 1.0000e-05
Epoch 15/15
219/219 ————— 115s 515ms/step - F1Score: 0.9750 - Precision: 0.9778 - Recall: 0.
9730 - accuracy: 0.9753 - loss: 0.1363 - val_F1Score: 0.9444 - val_Precision: 0.9452 - val_Rec
all: 0.9427 - val_accuracy: 0.9440 - val_loss: 0.2431 - learning_rate: 1.0000e-05

```

**Gambar 4. 15.** Proses Pelatihan Model (7:1.5:1.5)

Pada Gambar 4.15 disajikan proses pelatihan model untuk skenario 2, yaitu perbandingan data (7:1.5:1.5). Kemudian didapatkan *accuracy training* = 0.9753 dan *validation* = 0.9440

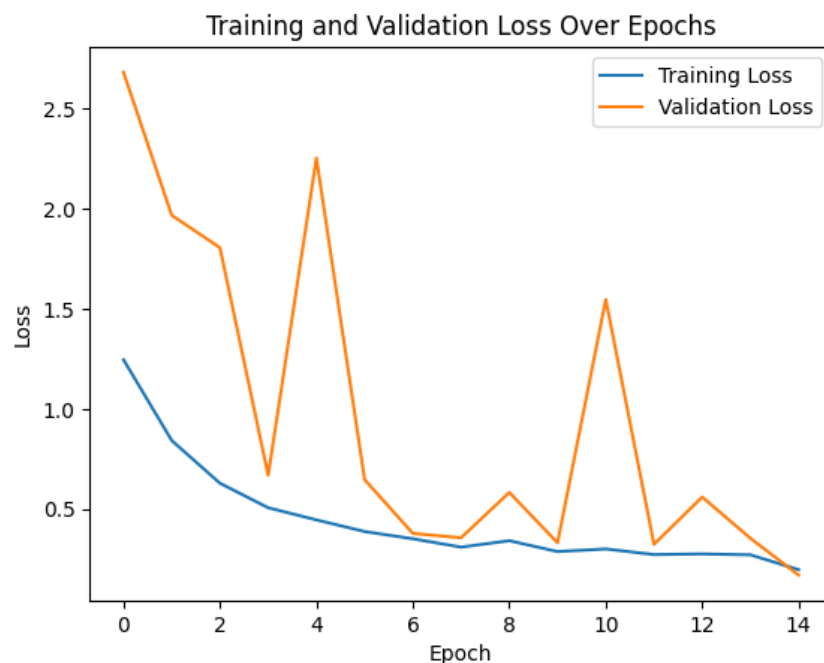
```

Epoch 12/15
157/157 ————— 84s 524ms/step - F1Score: 0.9537 - Precision: 0.9577 - Recall: 0.9460 - accuracy: 0.9533 - loss: 0.2319 - val_F1Score: 0.9321 - val_Precision: 0.9311 - val_Recall: 0.9296 - val_accuracy: 0.9312 - val_loss: 0.3030 - learning_rate: 1.0000e-04
Epoch 13/15
157/157 ————— 142s 527ms/step - F1Score: 0.9650 - Precision: 0.9658 - Recall: 0.9612 - accuracy: 0.9648 - loss: 0.1916 - val_F1Score: 0.9359 - val_Precision: 0.9358 - val_Recall: 0.9336 - val_accuracy: 0.9352 - val_loss: 0.2896 - learning_rate: 1.0000e-04
Epoch 14/15
157/157 ————— 84s 525ms/step - F1Score: 0.9696 - Precision: 0.9701 - Recall: 0.9663 - accuracy: 0.9699 - loss: 0.1824 - val_F1Score: 0.9320 - val_Precision: 0.9318 - val_Recall: 0.9296 - val_accuracy: 0.9312 - val_loss: 0.2970 - learning_rate: 1.0000e-05
Epoch 15/15
157/157 ————— 84s 525ms/step - F1Score: 0.9709 - Precision: 0.9719 - Recall: 0.9674 - accuracy: 0.9713 - loss: 0.1700 - val_F1Score: 0.9297 - val_Precision: 0.9303 - val_Recall: 0.9288 - val_accuracy: 0.9288 - val_loss: 0.3009 - learning_rate: 1.0000e-05

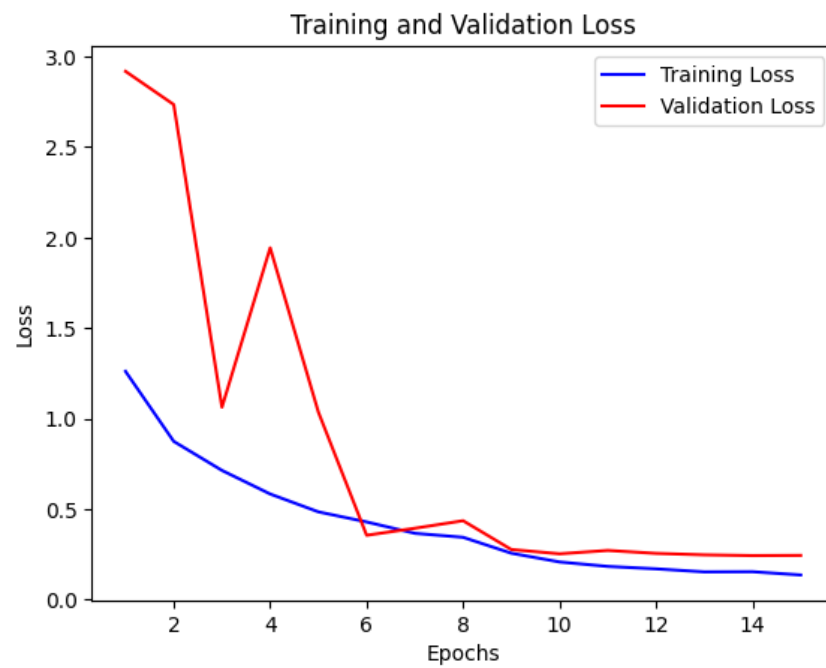
```

**Gambar 4. 16.** Proses Pelatihan Model (5:2.5:2.5)

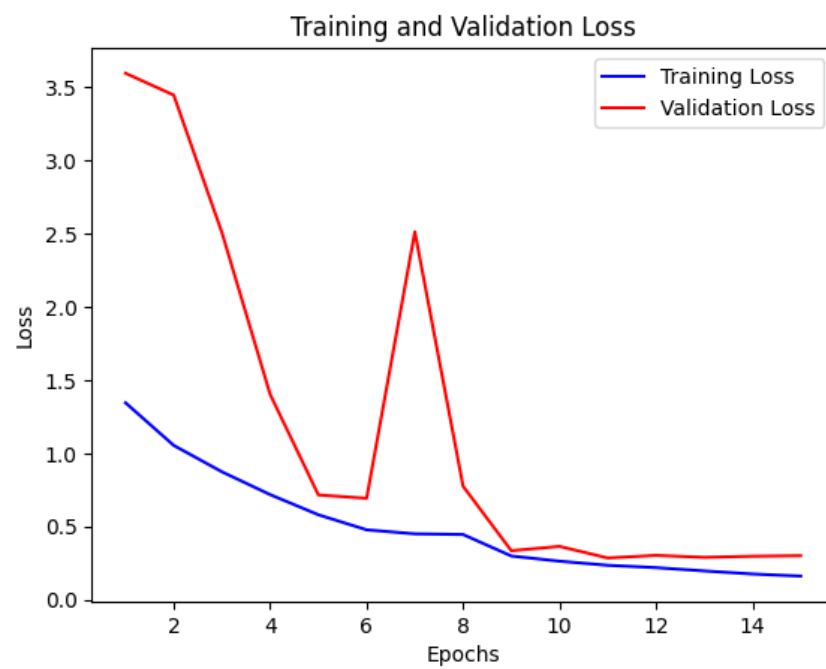
Pada Gambar 4.16, disajikan proses pelatihan model untuk skenario 3, yaitu perbandingan data (5:2.5:2.5). Kemudian didapatkan *accuracy training* = 0.9713 dan *validation* = 0.9288.



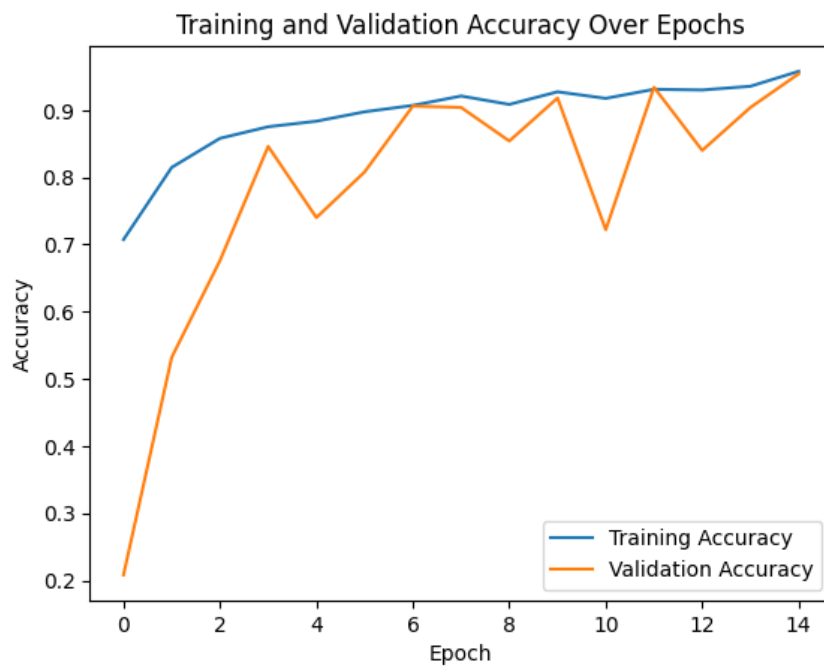
**Gambar 4. 17.** Grafik *Loss* Selama Pelatihan (8:1:1)



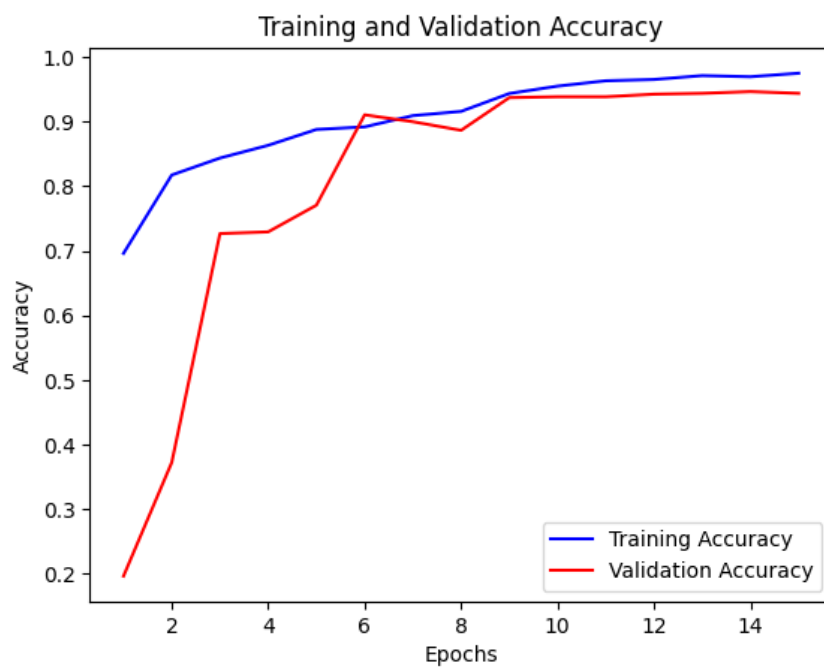
**Gambar 4. 18.** Grafik *Loss* Selama Pelatihan (7:1.5:1.5)



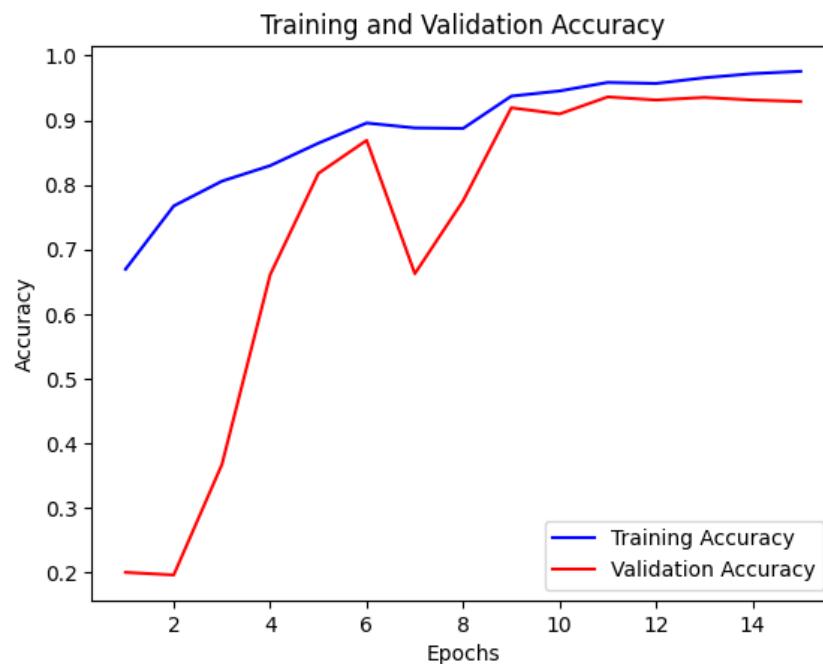
**Gambar 4. 19.** Grafik *Loss* Selama Pelatihan (5:2.5:2.5)



**Gambar 4. 20.** Grafik Akurasi Selama Pelatihan (8:1:1)



**Gambar 4. 21.** Grafik Akurasi Selama Pelatihan (7:1.5:1.5)



**Gambar 4. 22.** Grafik Akurasi Selama Pelatihan (5:2.5:2.5)

Berdasarkan grafik *loss* dan akurasi pelatihan pada Gambar 4.17 hingga Gambar 4.22, meskipun *accuracy* dan *loss* naik turun selama proses pelatihan, dapat dikatakan bahwa pelatihan model memberikan hasil yang cukup baik karena nilai *loss* mencapai titik terendah dan akurasi mencapai titik tertinggi pada akhir pelatihan.

#### 4.4.2. Pelatihan model deteksi

Untuk model deteksi, model dilatih dengan dua skenario pembagian data sebelum augmentasi dilakukan:

1. Menggunakan 1500 gambar (900 *train*, 150 *validation*, 150 *testing*).
2. Menggunakan 3000 gambar (1800 *train*, 300 *validation*, 300 *testing*).

Model deteksi dilatih menggunakan data latih masing masing skenario dengan jumlah *epochs* sebesar 32, *learning rate* sebesar 0.0001, *patience* 10, *dropout* 0.15, dan *optimizer* Adam.

```

Epoch   GPU_mem   box_loss   cls_loss   dfl_loss   Instances   Size
39/40    6.82G    0.6955     0.3219     1.292       12          640: 1
          Class   Images  Instances  Box(P       R          mAP50  m
          all     300      303       0.806       0.794       0.842   0.455

Epoch   GPU_mem   box_loss   cls_loss   dfl_loss   Instances   Size
40/40    6.82G    0.6812     0.311      1.277       12          640: 1
          Class   Images  Instances  Box(P       R          mAP50  m
          all     300      303       0.792       0.819       0.847   0.459

40 epochs completed in 1.013 hours.
Optimizer stripped from runs/detect/train/weights/last.pt, 52.0MB
Optimizer stripped from runs/detect/train/weights/best.pt, 52.0MB

Validating runs/detect/train/weights/best.pt...
Ultralytics YOLOv8.2.90 Python-3.10.14 torch-2.4.0 CUDA:0 (Tesla T4, 15095MiB)
Model summary (fused): 218 layers, 25,840,918 parameters, 0 gradients, 78.7 GFLOPs
          Class   Images  Instances  Box(P       R          mAP50  m
          all     300      303       0.795       0.822       0.849   0.459
          Cancer   150       150       0.781       0.893       0.858   0.471
          Non-Cancer 150       153       0.81       0.751       0.84    0.447
Speed: 0.2ms preprocess, 10.1ms inference, 0.0ms loss, 2.5ms postprocess per image
Results saved to runs/detect/train
💡 Learn more at https://docs.ultralytics.com/modes/train

```

**Gambar 4. 23.** Proses Pelatihan YOLOv8 (1500 gambar)

Pada Gambar 4.23, disajikan proses pelatihan model untuk skenario 1, yaitu menggunakan 1500 data gambar. Kemudian didapatkan nilai mAP untuk semua kelas sebesar 0.849.

```

Epoch   GPU_mem   box_loss   cls_loss   dfl_loss   Instances   Size
39/40    6.85G    0.7266    0.3207    1.315      8           640: 1
          Class   Images   Instances   Box(P       R          mAP50  m
          all     599      604        0.843      0.81       0.871   0.486

Epoch   GPU_mem   box_loss   cls_loss   dfl_loss   Instances   Size
40/40    6.83G    0.7271    0.3164    1.321      8           640: 1
          Class   Images   Instances   Box(P       R          mAP50  m
          all     599      604        0.855      0.802      0.864   0.485

40 epochs completed in 1.959 hours.
Optimizer stripped from runs/detect/train/weights/last.pt, 52.0MB
Optimizer stripped from runs/detect/train/weights/best.pt, 52.0MB

Validating runs/detect/train/weights/best.pt...
Ultralytics YOLOv8.2.90 Python-3.10.14 torch-2.4.0 CUDA:0 (Tesla T4, 15095MiB)
Model summary (fused): 218 layers, 25,840,918 parameters, 0 gradients, 78.7 GFLOPs
          Class   Images   Instances   Box(P       R          mAP50  m
          all     599      604        0.828      0.823      0.868   0.488
        Cancer     300      300        0.79      0.897      0.876   0.486
      Non-Cancer     299      304        0.866      0.75      0.86   0.489

Speed: 0.2ms preprocess, 9.5ms inference, 0.0ms loss, 1.9ms postprocess per image
Results saved to runs/detect/train
💡 Learn more at https://docs.ultralytics.com/modes/train

```

**Gambar 4. 24.** Proses Pelatihan YOLOv8 (3000 gambar)

Pada Gambar 4.24, disajikan proses pelatihan model untuk skenario 2, yaitu menggunakan 3000 data gambar. Kemudian didapatkan nilai mAP untuk semua kelas sebesar 0.868.

#### 4.5. Implementasi Tahap Evaluasi dan Pengujian Model

Setelah pelatihan, model kemudian diuji menggunakan skenario pembagian data yang berbeda untuk menilai performa secara komprehensif. Evaluasi dilakukan dengan menilai performa model melalui metrik *accuracy*, *precision*, *recall*, *F1-score*, serta *mean average precision* (mAP) untuk deteksi objek.

##### 4.5.1. Evaluasi model klasifikasi

Model klasifikasi diuji menggunakan tiga skenario pembagian data:

1. 80% data untuk pelatihan (4000 data), 10% untuk validasi (500 data), dan 10% untuk pengujian (500 data).
2. 70% data untuk pelatihan (3500 data), 15% untuk validasi (750 data), dan 15% untuk pengujian (750 data).
3. 50% data untuk pelatihan (2500 data), 25% untuk validasi (1250 data), dan 25% untuk pengujian (1250 data).

Untuk setiap skenario setelah model selesai dilatih, dilakukan evaluasi dengan *confusion matrix* dan *classification report* untuk mengukur kinerjanya. Setiap skenario memberikan gambaran bagaimana model dapat menggeneralisasi pada data uji yang berbeda proporsi. Kode pengujian disajikan dalam Gambar 4.25.

```
import numpy as np
from sklearn.metrics import classification_report, confusion_matrix
from keras.models import load_model

# Prediksi menggunakan test generator
predictions = model.predict(test_generator)
y_pred = np.argmax(predictions, axis=1)
y_true = test_generator.classes

# Tampilkan confusion matrix
conf_matrix = confusion_matrix(y_true, y_pred)
print("Confusion Matrix:")
print(conf_matrix)

# Tampilkan classification report
class_labels = list(test_generator.class_indices.keys())
class_report = classification_report(y_true, y_pred, target_names=class_labels)
print("\nClassification Report:")
print(class_report)
```

**Gambar 4. 25.** Kode Pengujian Model Klasifikasi

```
Confusion Matrix:
[[92  8  0  0  0]
 [11 89  0  0  0]
 [ 0  0 98  2  0]
 [ 2  5  0 93  0]
 [ 0  1  0  1 98]]
```

**Gambar 4. 26.** *Confusion Matrix* Model Klasifikasi (8:1:1)



Confusion Matrix:

```
[[133  17   0   0   0]
 [ 12 137   0   1   0]
 [  1   1 147   1   0]
 [  2   8   0 140   0]
 [  0   1   1   3 145]]
```

**Gambar 4. 27.** *Confusion Matrix* Model Klasifikasi (7:1.5:1.5)

Confusion Matrix:

```
[[211  39   0   0   0]
 [ 14 234   0   1   1]
 [  0   1 248   1   0]
 [  1  10   1 238   0]
 [  3   4   2   3 238]]
```

**Gambar 4. 28.** *Confusion Matrix* Model Klasifikasi (5:2.5:2.5)

Classification Report:

	precision	recall	f1-score	support
BCC	0.88	0.92	0.90	100
BKL	0.86	0.89	0.88	100
Melanoma	1.00	0.98	0.99	100
NV	0.97	0.93	0.95	100
Seborrheic Keratoses and other Benign Tumors	1.00	0.98	0.99	100
accuracy			0.94	500
macro avg	0.94	0.94	0.94	500
weighted avg	0.94	0.94	0.94	500

**Gambar 4. 29.** *Classification Report* (8:1:1)

Classification Report:

	precision	recall	f1-score	support
BCC	0.90	0.89	0.89	150
BKL	0.84	0.91	0.87	150
Melanoma	0.99	0.98	0.99	150
NV	0.97	0.93	0.95	150
Seborrheic Keratoses and other Benign Tumors	1.00	0.97	0.98	150
accuracy			0.94	750
macro avg	0.94	0.94	0.94	750
weighted avg	0.94	0.94	0.94	750

**Gambar 4. 30.** *Classification Report (7:1.5:1.5)*

Classification Report:

	precision	recall	f1-score	support
BCC	0.92	0.84	0.88	250
BKL	0.81	0.94	0.87	250
Melanoma	0.99	0.99	0.99	250
NV	0.98	0.95	0.97	250
Seborrheic Keratoses and other Benign Tumors	1.00	0.95	0.97	250
accuracy			0.94	1250
macro avg	0.94	0.94	0.94	1250
weighted avg	0.94	0.94	0.94	1250

**Gambar 4. 31.** *Classification Report (5:2.5:2.5)*

Confusion matrix dan *classification report* ditampilkan dalam bentuk yang terperinci pada beberapa gambar berikut: Gambar 4.25 hingga Gambar 4.31. Selanjutnya ialah pembuktian nilai *classification report*-nya menggunakan hasil dari *confusion matrix*. Sebagai contoh, kita mengambil skenario 1, yaitu perbandingan (8:1:1):

- *Precision* 1 =  $92/92+13 = 92/105 = 0.88$
- *Precision* 2 =  $89/89+14 = 89/103 = 0.86$
- *Precision* 3 =  $98/98+0 = 98/98 = 1$
- *Precision* 4 =  $93/93+3 = 93/96 = 0.97$
- *Precision* 5 =  $98/98+0 = 98/98 = 1$

$$\text{Average Precision} = (0.88 + 0.86 + 1 + 0.97 + 1) / 5 = 0.94$$

- $Recall\ 1 = 92/92+8 = 92/100 = 0.92$
- $Recall\ 2 = 89/89+11 = 89/100 = 0.89$
- $Recall\ 3 = 98/98+2 = 98/100 = 0.98$
- $Recall\ 4 = 93/93+7 = 93/100 = 0.93$
- $Recall\ 5 = 98/98+2 = 98/100 = 0.98$

$$Average\ Recall = (0.92+0.89+0.98+0.93+0.98) / 5 = 0.94$$

- $F1-score\ 1 = 2 \times \frac{0.88 \times 0.92}{0.88+0.92} = 2 \times \frac{0.8096}{1.8} = 0.89 = 0.90$
- $F1-score\ 2 = 2 \times \frac{0.86 \times 0.89}{0.86+0.89} = 2 \times \frac{0.7654}{1.75} = 0.8747 = 0.88$
- $F1-score\ 3 = 2 \times \frac{1 \times 0.98}{1+0.98} = 2 \times \frac{0.98}{1.98} = 0.8995 = 0.989 = 0.99$
- $F1-score\ 4 = 2 \times \frac{0.97 \times 0.93}{0.97+0.93} = 2 \times \frac{0.9021}{1.9} = 0.949 = 0.95$
- $F1-score\ 5 = 2 \times \frac{1 \times 0.98}{1+0.98} = 2 \times \frac{0.98}{1.98} = 0.989 = 0.99$

$$Average\ F1-score = (0.90 + 0.88 + 0.99 + 0.95 + 0.99) / 5 = 0.94$$

- $Accuracy\ 1 = 92/100 = 0.92$
- $Accuracy\ 2 = 89/100 = 0.89$
- $Accuracy\ 3 = 98/100 = 0.98$
- $Accuracy\ 4 = 93/100 = 0.93$
- $Accuracy\ 5 = 98/100 = 0.98$

$$Average\ Accuracy = (0.92 + 0.89 + 0.98 + 0.93 + 0.98) / 5 = 0.94$$

Berdasarkan hasil pembuktian, nilai *precision*, *recall*, *F1-score*, dan *accuracy* dari *confusion matrix* sesuai dengan hasil yang ditampilkan pada *classification report*. Sebagai contoh, pada skenario 1 dengan perbandingan data 8:1:1, perhitungan manual untuk setiap metrik menghasilkan nilai yang konsisten dengan laporan klasifikasi.

**Tabel 4.1.** Perbandingan Model Klasifikasi

	<i>Accuracy</i>		
	Model 1 (8:1:1)	Model 2 (7:1.5:1.5)	Model 3 (5:2.5:2.5)
<i>Training</i>	0.9541	0.9753	0.9713
<i>Validation</i>	0.9540	0.9440	0.9288
<i>Testing</i>	0.94	0.94	0.94

Berdasarkan hasil perbandingan seluruh model yang ditampilkan pada Tabel 4.1, model 1 terbukti menjadi yang terbaik. Model ini menunjukkan kestabilan yang lebih tinggi, tanpa tanda-tanda *overfitting* yang signifikan, serta mampu memberikan generalisasi yang baik terhadap data pengujian.

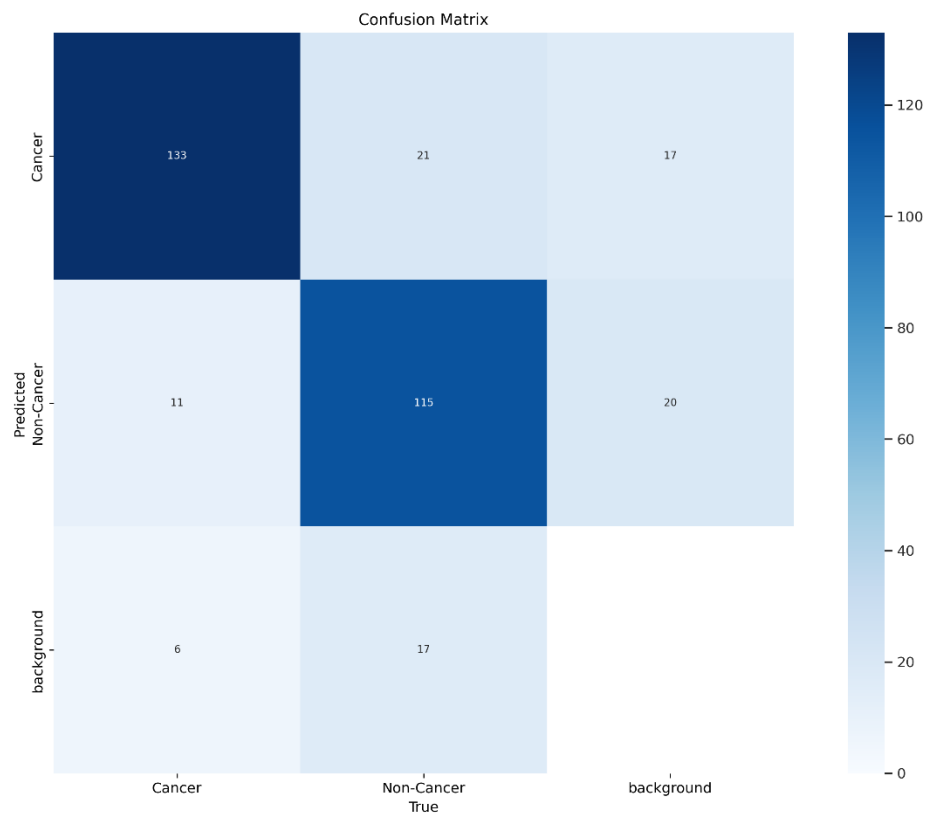
#### 4.5.2. Evaluasi model deteksi

Model deteksi diuji menggunakan data uji. Kode pengujian, *confusion matrix*, dan grafik pelatihan diperlihatkan pada Gambar 4.32 hingga Gambar 4.36.

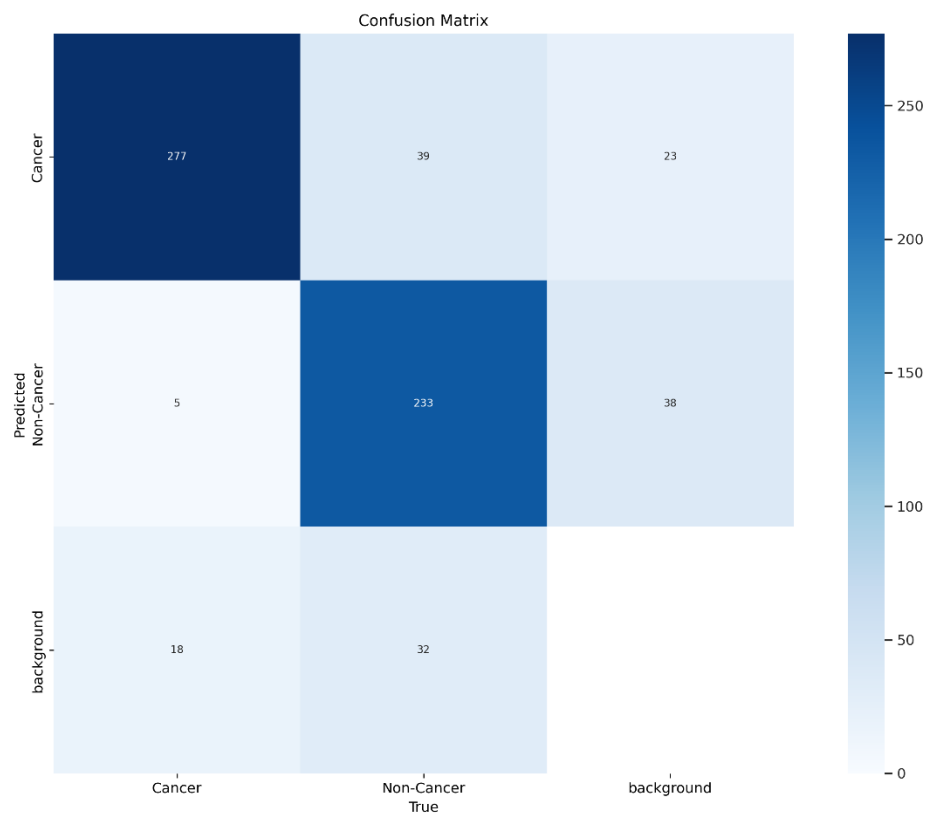
```
# Load a pretrained YOLOv8 model
model = YOLO('/kaggle/working/runs/detect/train/weights/best.pt')

# Run validation on a set specified as 'val' argument
metrics = model.val(data='/kaggle/working/skin-disease-new-12/data.yaml', split="test")
```

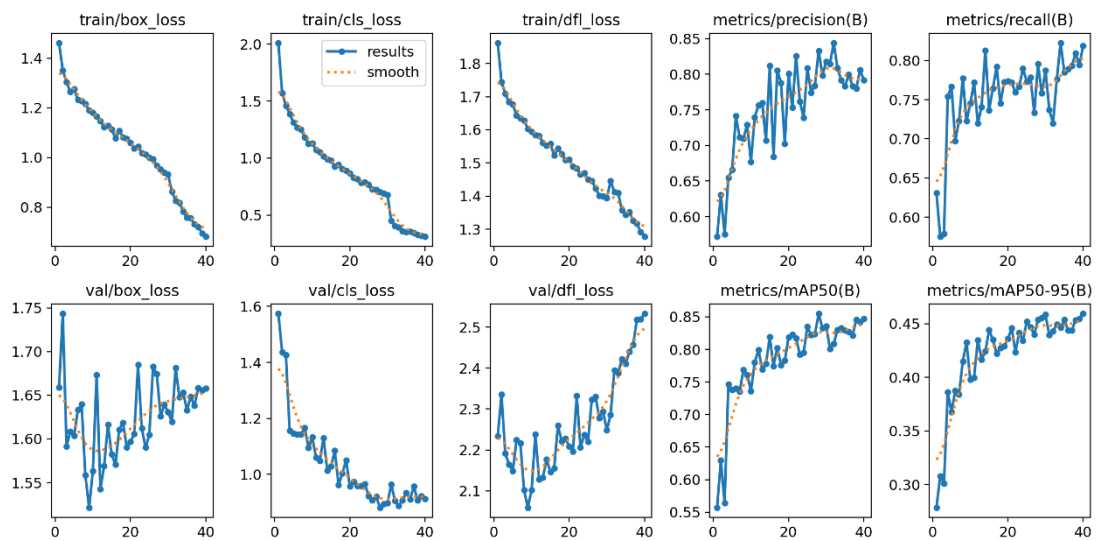
**Gambar 4. 32.** Kode Pengujian Model Deteksi



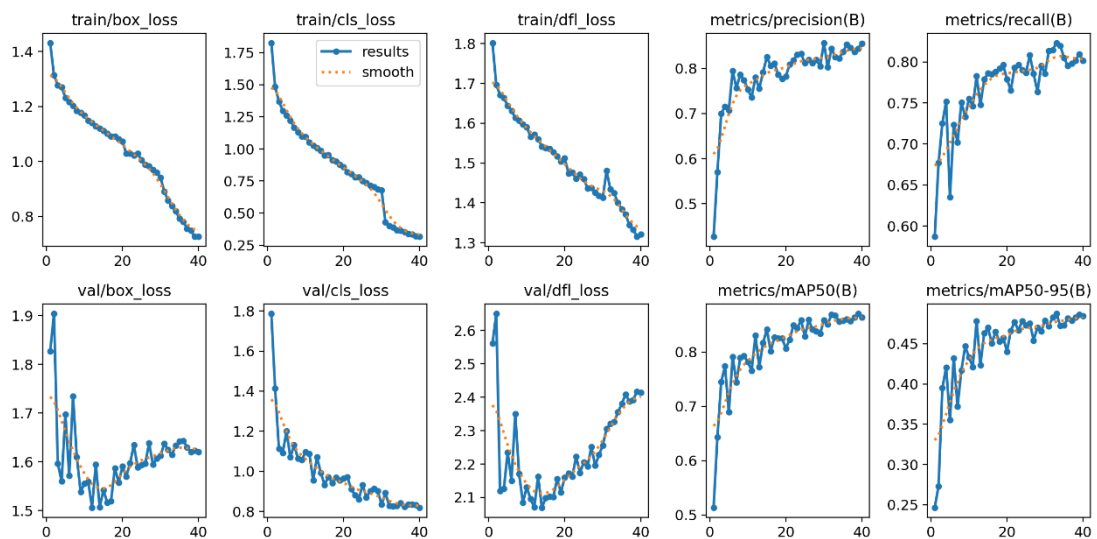
**Gambar 4. 33.** *Confusion Matrix* Model Deteksi 1500 Gambar



**Gambar 4. 34.** *Confusion Matrix* Model Deteksi 3000 Gambar



**Gambar 4. 35.** Grafik Pelatihan Model Deteksi 1500 Gambar



**Gambar 4. 36.** Grafik Pelatihan Model Deteksi 3000 Gambar

Adapun beberapa keterangan dari grafik pelatihan dan validasi model diatas ialah :

1. *box\_loss* : Box loss mengukur seberapa baik model dalam memprediksi lokasi *bounding box* yang benar untuk objek yang terdeteksi.
2. *cls loss*: *Classification loss* mengukur seberapa akurat model dalam mengklasifikasikan objek ke kategori yang tepat.
3. *dfl loss*: Ukuran dari seberapa baik model mempelajari fitur spasial dengan mempertimbangkan arah dan distribusi spasial objek.

4. *precision* (B): Mengukur seberapa tepat *bounding box* yang diprediksi sesuai dengan objek yang ada. *Precision* tinggi berarti sedikit kesalahan positif palsu (FP).
5. *recall* (B): Mengukur seberapa baik model mendeteksi semua objek yang relevan. *Recall* tinggi berarti sedikit objek yang terlewat atau kesalahan negatif palsu (FN).

Berdasarkan grafik hasil pelatihan model, bisa dikatakan bahwa model deteksi cukup baik, karena mengalami peningkatan performa selama pelatihan. Hal ini bisa dilihat dari metrik *loss* yang mengalami penurunan, juga metrik presisi, *recall*, serta mAP mengalami peningkatan.

**Tabel 4.2.** Perbandingan Model Deteksi

	<i>Mean average precision</i> (mAP)	
	Model 1 (1500 Gambar)	Model 2 (3000 Gambar)
<i>Training</i>	0.849	0.868
<i>Testing</i>	0.771	0.812

Berdasarkan hasil perbandingan seluruh model yang ditampilkan pada Tabel 4.2, model 2 dengan 3000 gambar merupakan model yang terbaik. Hal ini karena model 2 memiliki mAP yang lebih tinggi pada data pengujian (0.812) dibandingkan dengan model 1 (0.771), serta mAP pada data pelatihan yang juga lebih baik (0.868 dibandingkan 0.849). Berdasarkan nilai mAP, bisa dikatakan bahwa model 2 mampu mendeteksi objek dengan lebih akurat, baik pada data pelatihan maupun data pengujian, dikarenakan model 2 memiliki nilai mAP yang lebih tinggi dibandingkan dengan model 1.

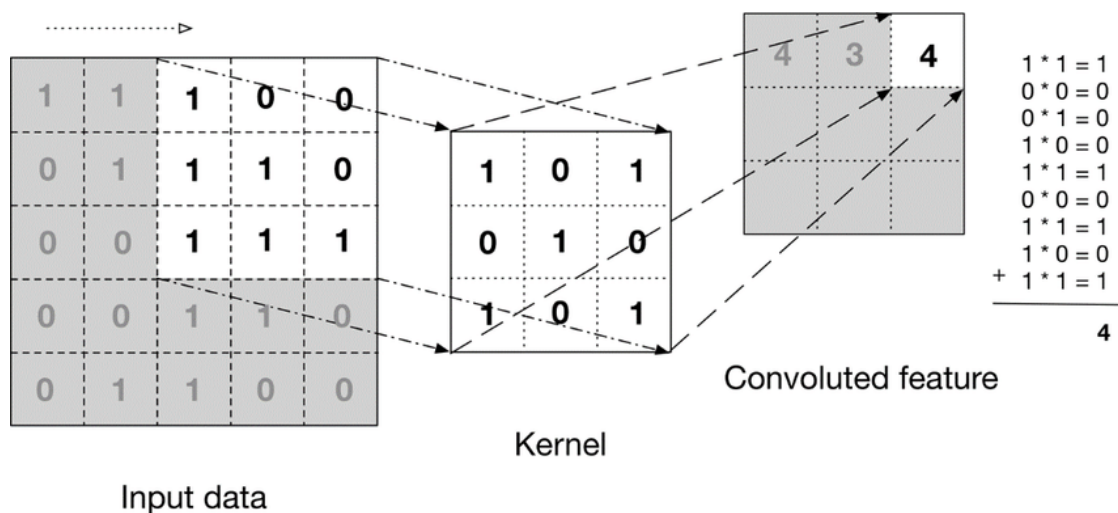
Model ini juga menyediakan informasi tambahan mengenai ukuran penyakit yang terdeteksi, yang dibagi menjadi tiga kategori: kecil, sedang, dan besar. Kategorisasi ini didasarkan pada luas *bounding box* hasil deteksi, dengan ambang batas sebagai berikut: kecil jika luasnya kurang dari 66.677 piksel, sedang antara 66.677 hingga 302.013 piksel, dan besar jika lebih dari 302.013 piksel.

#### 4.6. Analisis Proses EfficientNetB7

Secara konsep dasar, YOLOv8 dan EfficientNetB7 memiliki persamaan dalam hal klasifikasi, yaitu keduanya sama-sama menggunakan jaringan syaraf convolutional (CNN). Namun, tujuan penggunaannya berbeda. EfficientNetB7 digunakan untuk klasifikasi gambar, sementara YOLOv8 digunakan untuk deteksi objek. Proses yang dilakukan pada keduanya mencakup beberapa tahapan penting, yaitu:

##### 4.6.1. Feature extraction

*Feature extraction* adalah proses ekstraksi fitur dari gambar *input* melalui serangkaian *convolutional layers*. Misalnya, kita memiliki sebuah gambar penyakit kulit berukuran 6x6 piksel (36 piksel total), dengan skala warna *grayscale*. Setelah parameter *kernel* dan *stride* ditentukan, misalnya *kernel* 3x3 dan *stride* 1, *kernel* tersebut diterapkan pada gambar untuk melakukan operasi konvolusi. *Stride* menentukan seberapa jauh *kernel* (filter) bergerak melintasi gambar *input* selama operasi konvolusi. Pada setiap langkah konvolusi, *kernel* kemudian melakukan operasi *dot product* terhadap bagian gambar yang dicakup oleh kernel, mengikuti nilai *stride*, hingga seluruh piksel telah diproses. Hasilnya adalah sejumlah *feature map* yang menyoroti fitur-fitur spesifik dari gambar tersebut, seperti yang diperlihatkan pada Gambar 4.37.



**Gambar 4. 37.** Feature Extraction

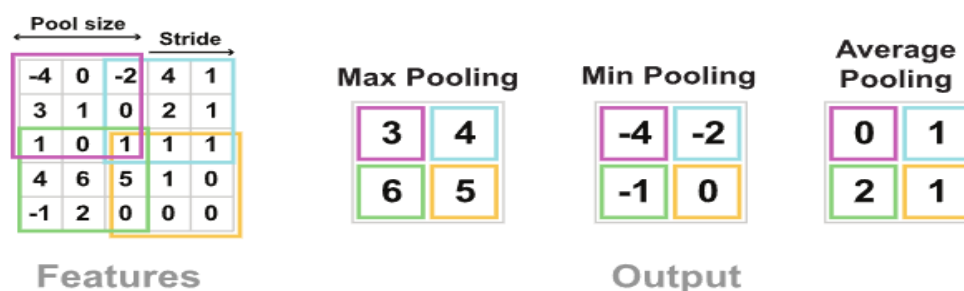
Sumber: Fronzetti, 2019



*Feature map* ini memungkinkan model untuk mempelajari representasi fitur yang berbeda dari gambar, yang kemudian digunakan untuk tugas-tugas seperti klasifikasi objek oleh EfficientNetB7 atau deteksi objek oleh YOLOv8.

#### 4.6.2. Pooling

Setelah *feature map* dihasilkan oleh lapisan konvolusi, mereka diproses lebih lanjut oleh *Pooling layer* untuk merangkum informasi dan mengurangi ukuran *feature map*. Ini membantu mengurangi kompleksitas model sekaligus mempertahankan informasi penting. Beberapa teknik *pooling* yang umum digunakan meliputi *Max Pooling*, *Min Pooling*, dan *Average Pooling*, seperti pada Gambar 4.38.



**Gambar 4. 38.** Teknik *Pooling*

Sumber: <https://epynn.net/>

Setelah *pooling*, hasilnya adalah *feature map* dengan resolusi lebih rendah namun dengan informasi penting yang telah dirangkum. Misalkan setelah dilakukan *max pooling*, didapatkan matrix 2x2 sebagai hasilnya :

$$\text{Matrix setelah max pooling} = \begin{bmatrix} 3 & 4 \\ 6 & 5 \end{bmatrix}$$

Matriks ini kemudian diproses lebih lanjut tergantung pada tugas model.

#### 4.6.3. Klasifikasi gambar

1. *Flattening*: Selanjutnya matriks 2x2 diratakan (flattened) menjadi vektor satu dimensi: Flattened Vector = [ 3 , 4 , 6 , 5 ]
2. *Fully Connected Layers*: Pada tahap ini, flattened vector tadi kemudian dimasukkan kedalam *fully connected layers*, menggunakan rumus  $z = Wx + b$ , dimana :
  - W adalah matriks bobot yang menghubungkan setiap elemen dari vektor rata dengan neuron di layer fully connected.

- $x$  adalah vektor rata yang diperoleh dari flattening *feature map*.
- $b$  adalah bias yang ditambahkan untuk setiap neuron di fully connected layer.
- $z$  adalah *output* dari layer tersebut, yang kemudian biasanya diikuti oleh fungsi aktivasi seperti *softmax*.

Bobot dan bias dalam jaringan saraf tiruan dioptimalkan secara otomatis selama proses pelatihan. Sebagai contoh, karena klasifikasi ini memiliki 5 kelas, maka bobot  $W$  (berukuran  $5 \times 4$ ) dan bias  $b$  (berukuran  $5 \times 1$ ) bisa diberikan seperti:

$$W = \begin{bmatrix} 0.1 & -0.2 & 0.3 & 0.4 \\ 0.2 & 0.1 & -0.1 & -0.3 \\ -0.4 & 0.5 & 0.2 & -0.1 \\ 0.3 & -0.4 & 0.1 & 0.2 \\ -0.1 & 0.2 & -0.3 & 0.4 \end{bmatrix} \quad b = \begin{bmatrix} 0.1 \\ -0.1 \\ 0.2 \\ -0.2 \\ 0.1 \end{bmatrix}$$

Kemudian kita dapat menghitung nilai  $z$  menggunakan rumus  $Wx+b$ .

$$W.x = \begin{bmatrix} 0.1 & -0.2 & 0.3 & 0.4 \\ 0.2 & 0.1 & -0.1 & -0.3 \\ -0.4 & 0.5 & 0.2 & -0.1 \\ 0.3 & -0.4 & 0.1 & 0.2 \\ -0.1 & 0.2 & -0.3 & 0.4 \end{bmatrix} \cdot \begin{bmatrix} 3 \\ 4 \\ 6 \\ 5 \end{bmatrix}$$

$$W.x = \begin{bmatrix} 0.3 & -0.8 & +1.8 & +2 \\ 0.6 & +0.4 & -0.6 & -1.5 \\ -1.2 & +2 & +1.2 & -0.5 \\ 0.9 & -1.6 & +0.6 & +1 \\ -0.3 & +0.8 & -1.8 & +2 \end{bmatrix} = \begin{bmatrix} 3.3 \\ -1.1 \\ 1.5 \\ 0.9 \\ 0.7 \end{bmatrix}$$

Selanjutnya tambahkan nilai bias.

$$z = \begin{bmatrix} 3.3 \\ -1.1 \\ 1.5 \\ 0.9 \\ 0.7 \end{bmatrix} + \begin{bmatrix} 0.1 \\ -0.1 \\ 0.2 \\ -0.2 \\ 0.1 \end{bmatrix} = \begin{bmatrix} 3.4 \\ -1.2 \\ 1.7 \\ 0.7 \\ 0.8 \end{bmatrix}$$

Setiap elemen dalam vektor  $z$  mewakili masing masing kelas.

- $z_1 = 3.4$  (skor untuk kelas 1)
- $z_2 = -1.2$  (skor untuk kelas 2)
- $z_3 = 1.7$  (skor untuk kelas 3)
- $z_4 = 0.7$  (skor untuk kelas 4)
- $z_5 = 0.8$  (skor untuk kelas 5)

### 3. Activation Function Softmax

Untuk menghitung fungsi aktivasi *softmax*, kita perlu mengubah skor mentah  $z$  menjadi probabilitas untuk masing-masing kelas. Fungsi *softmax* mengubah skor  $z$  menjadi nilai probabilitas dengan cara berikut.

1. Eksponensiasi skor: Hitung eksponen dari setiap skor  $z_i$
2. Normalisasi: Bagikan eksponen setiap skor dengan jumlah total eksponen untuk mendapatkan probabilitas.

Rumus *softmax* dapat dibuat sebagai :

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^n e^{z_j}}$$

Di mana:

- $e^{z_i}$  adalah eksponen dari skor ke- $i$
- $\sum_{j=1}^n e^{z_j}$  adalah jumlah eksponen dari semua skor dalam vektor  $z$

Skor  $z = [3.4, -1.2, 1.7, 0.7, 0.8]$

Maka

$$e^{z_1} = e^{3.4} \approx 30.12$$

$$e^{z_2} = e^{-1.2} \approx 0.30$$

$$e^{z_3} = e^{1.7} \approx 5.47$$

$$e^{z_4} = e^{0.7} \approx 2.01$$

$$e^{z_5} = e^{0.8} \approx 2.23$$

$$\text{Total} = 30.12 + 0.30 + 5.47 + 2.01 + 2.23 = 40.13$$

Selanjutnya kita menghitung probabilitasnya.

- Probabilitas 1 =  $\frac{30.12}{40.13} = 0.752$
- Probabilitas 2 =  $\frac{0.30}{40.13} = 0.0075$
- Probabilitas 3 =  $\frac{5.47}{40.13} = 0.136$
- Probabilitas 4 =  $\frac{2.01}{40.13} = 0.050$
- Probabilitas 5 =  $\frac{2.23}{40.13} = 0.056$

Jadi, setelah menerapkan fungsi *softmax* pada skor  $z$ , kita mendapatkan probabilitas untuk masing-masing kelas, dimana probabilitas tertinggi berada pada kelas 1 yaitu 0.752. Maka gambar *input* tersebut kemudian diprediksi sebagai kelas 1.

## 4.7. Analisis Proses YOLOv8

### 4.7.1. Feature extraction

Proses ini sangat mirip dengan yang ada di EfficientNetB7. YOLOv8 menggunakan jaringan konvolusi untuk mengekstrak fitur dari gambar *input*. Seperti pada EfficientNetB7, gambar *input* diproses melalui beberapa lapisan konvolusi, menghasilkan *feature maps* yang mewakili fitur-fitur tertentu dari gambar. Pada setiap lapisan konvolusi, *kernel* menggeser gambar dan menghasilkan *output* berupa *feature maps* yang menangkap berbagai pola dari gambar.

### 4.7.2. Bounding box prediction

YOLOv8 tidak hanya melakukan klasifikasi tetapi juga memprediksi *bounding box* untuk setiap objek dalam gambar. *Bounding box* ini diwakili oleh empat parameter:  $(x,y,w,h)$  di mana  $(x,y)$  adalah titik pusat koordinat dari *bounding box*, dan  $w$  serta  $h$  ialah lebar dan tinggi dari *bounding box*.



**Gambar 4. 39.** Contoh Pencarian *Bounding Box*

Sumber: Redmon et al., 2015

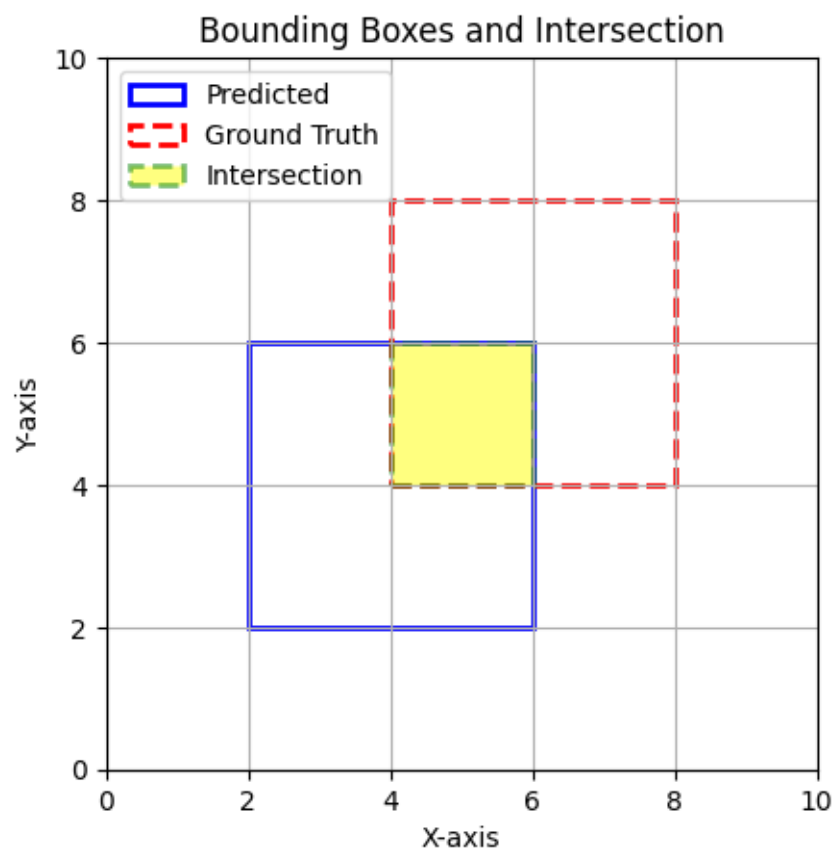
Proses pencarian *bounding box* ditampilkan pada Gambar 4.39 di halaman 60, dimana model mencari *bounding box* menggunakan *anchor box* dengan berbagai ukuran. Pada tahap ini model memprediksi *bounding box* untuk setiap objek, serta *confidence score* dan kelas objek.

#### 4.7.3. Intersection over union (IoU)

Setelah *bounding boxes* diprediksi, YOLOv8 menghitung *Intersection over Union* (IoU) untuk menilai kesesuaian antara *bounding box* yang diprediksi dan *bounding box ground truth*. IoU merupakan rasio antara *area* tumpang tindih (*intersection*) dari kedua *bounding boxes* dan *area* gabungan (*union*) mereka. Metrik ini digunakan untuk mengukur seberapa akurat prediksi *bounding box* dibandingkan dengan *ground truth*. Nilai IoU berkisar antara 0 hingga 1, dan semakin tinggi nilainya, semakin baik tingkat akurasi prediksi.

$$\text{IoU} = \frac{\text{Area of Intersection}}{\text{Area of Union}}$$

Misalkan kita memiliki dua *bounding boxes*, seperti pada gambar 4.40.



**Gambar 4. 40.** BBox pred vs *ground truth*

Kita mendapatkan nilai koordinat dari *bounding\_box* :

- *Bounding box* prediksi (BB\_pred) : (2,2) hingga (6,6)
- *Bounding box ground truth* (BB\_gt) : (4,4) hingga (8,8)

Kemudian kita menghitung *Area of Intersection* :

- Koordinat sudut kiri atas *area* tumpang tindih (*intersection*) : (4,4)
- Koordinat sudut kanan bawah *area* tumpang tindih (*intersection*) : (6,6)
- Lebar *area intersection* :  $6 - 4 = 2$
- Tinggi *area intersection* :  $6 - 4 = 2$
- *Area intersection* :  $2 \times 2 = 4$

Selanjutnya kita menghitung *Area of Union* :

- *Area* BB\_pred :  $4 \times 4 = 16$
- *Area* BB\_gt :  $4 \times 4 = 16$
- *Area* gabungan =  $16 + 16 - 4 = 28$

Lalu kita hitung nilai IoUnya:

$$\text{IoU} = \frac{\text{Area of Intersection}}{\text{Area of Union}} = \frac{4}{28} = 0.143$$

Jadi, nilai IoU untuk *bounding boxes* ini adalah sekitar 0.143.

Biasanya, kita menetapkan ambang batas IoU untuk menentukan apakah prediksi *bounding box* dianggap benar atau tidak. Misalnya, jika ambang batas IoU adalah 0.5, maka prediksi dengan  $\text{IoU} \geq 0.5$  dianggap benar, sedangkan yang lebih rendah dianggap salah.

#### 4.7.4. Non-maximum suppression (NMS)

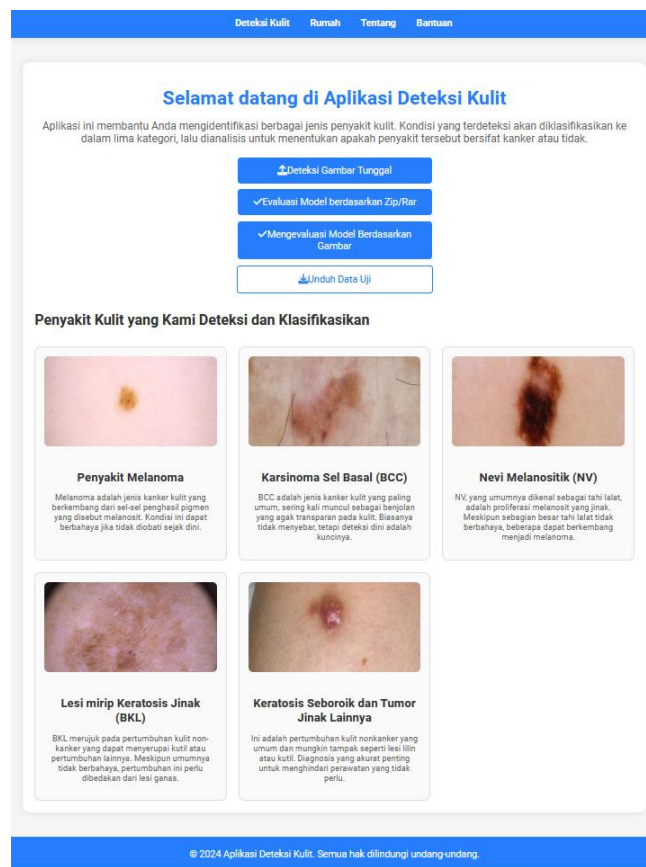
YOLO biasanya memprediksi banyak *bounding box*, kemudian memilih *bounding box* dengan nilai IoU tertinggi, yang biasa disebut sebagai *Non-maximum suppression* (Jiang et al., 2021). YOLOv8 menggunakan teknik *Non-Maximum Suppression* (NMS) untuk menyaring *bounding boxes* yang tumpang tindih. NMS mempertahankan *bounding box* dengan *confidence score* tertinggi dan menghapus yang lainnya jika IoU melebihi ambang batas tertentu.

#### 4.7.5. Output bounding box dan label kelas

Setelah NMS diterapkan, YOLOv8 memberikan *output* berupa *bounding boxes* final yang telah disaring, bersama dengan label kelas untuk setiap *bounding box*. Setiap *bounding box* yang dihasilkan menunjukkan posisi dan ukuran objek yang terdeteksi dalam gambar, serta kelas objek tersebut.

### 4.8. Implementasi Sistem

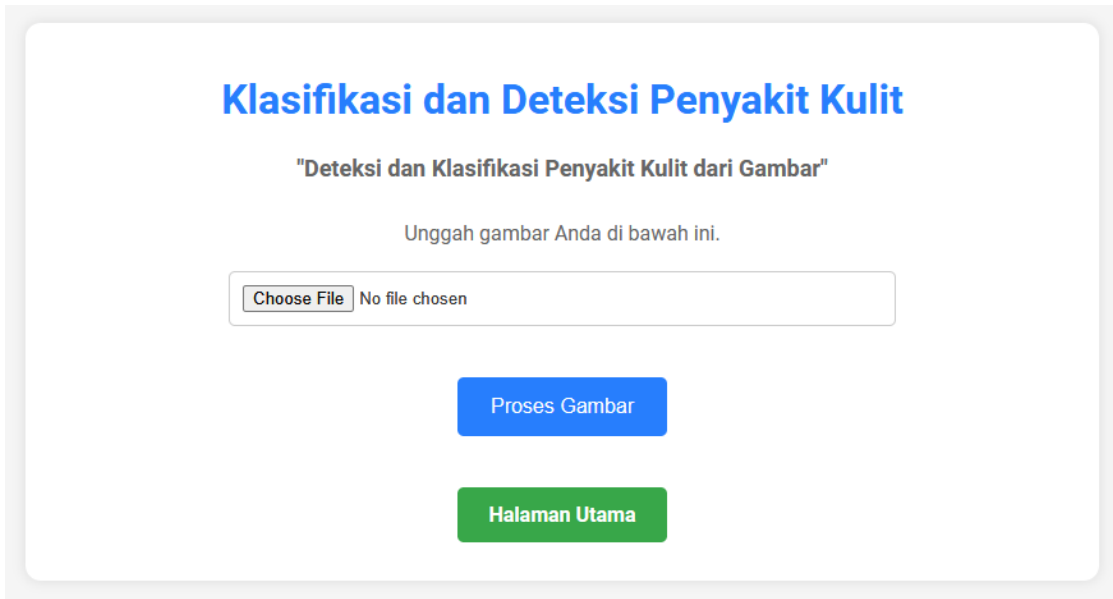
#### 4.8.1. Halaman utama



**Gambar 4. 41.** Halaman Utama

Hasil implementasi pada halaman utama dimana pengguna dapat memulai program dengan menekan tombol menuju program. Halaman utama ditampilkan pada Gambar 4.41.

#### 4.8.2. Halaman prediksi



**Klasifikasi dan Deteksi Penyakit Kulit**

"Deteksi dan Klasifikasi Penyakit Kulit dari Gambar"

Unggah gambar Anda di bawah ini.

[Choose File](#) No file chosen

[Proses Gambar](#)

[Halaman Utama](#)

**Gambar 4. 42.** Halaman Prediksi Gambar

Gambar 4.42 menunjukkan implementasi dari halaman prediksi yang memungkinkan pengguna untuk mengunggah gambar. Setelah gambar diunggah, *website* kemudian memprosesnya dan menampilkan hasil klasifikasi serta deteksi terkait penyakit kulit yang telah disebutkan sebelumnya.



#### 4.8.3. Halaman hasil klasifikasi dan deteksi

**Klasifikasi & Deteksi Penyakit Kulit**

Bahasa Indonesia English

**"Mendeteksi dan Mengklasifikasikan Penyakit Kulit dari Gambar"**

Unggah gambar Anda di bawah.

Choose File No file chosen

Proses Gambar

**Hasil Klasifikasi:**

Penyakit kulit yang diprediksi adalah: **Seborrheic Keratoses and other Benign Tumors**

**Tentang Seborrheic Keratoses and other Benign Tumors:**

Keratososis Seboroik dan tumor jinak lainnya adalah pertumbuhan kulit non-kanker (jinak) yang umum terjadi. Seringkali muncul sebagai pertumbuhan berwarna coklat, hitam, atau coklat muda di wajah, dada, bahu, atau punggung. Pertumbuhannya seperti lilin, bersisik, dan sedikit meninggi.

**Hasil Deteksi:**

Non-Cancer

**Kategori Ukuran Lesi:**

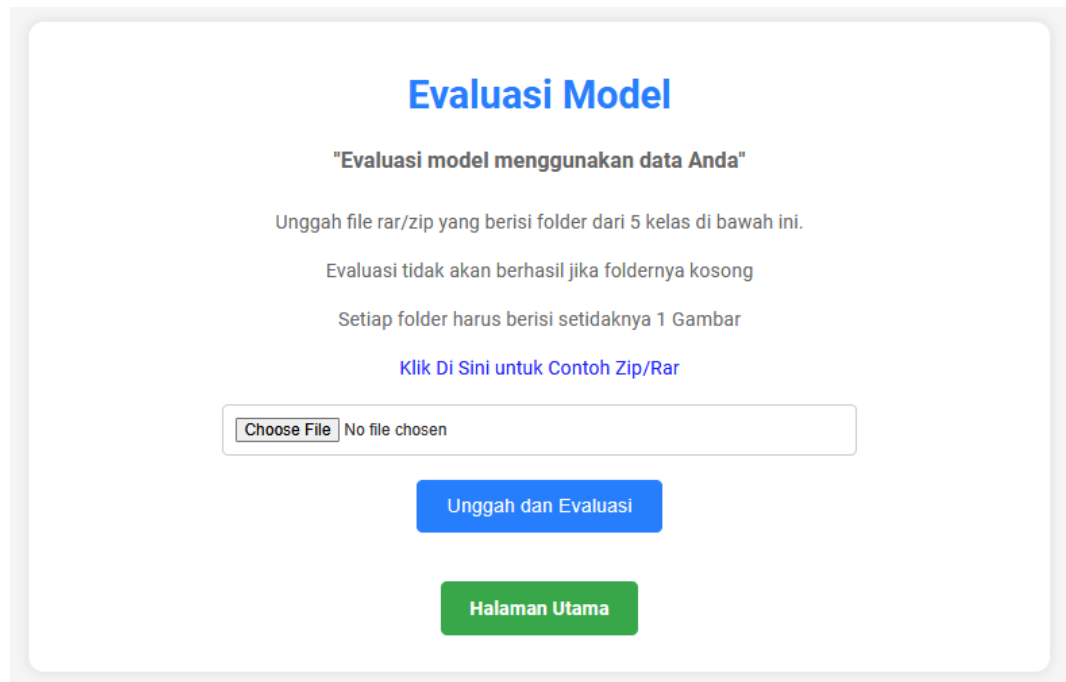
Kecil

Mulai Ulang Halaman Utama

**Gambar 4. 43.** Halaman Hasil Prediksi Gambar

Setelah gambar yang diunggah oleh pengguna selesai diproses, hasilnya kemudian diberikan pada halaman hasil prediksi gambar, seperti yang ditampilkan pada Gambar 4.43. Halaman ini menyajikan informasi seperti kelas penyakit kulit yang diprediksi. Selain itu, halaman ini juga menampilkan hasil deteksi gambar yang telah diberi *bounding box* dan prediksi apakah kondisi tersebut adalah kanker atau non-kanker.

#### 4.8.4. Halaman evaluasi model



**Evaluasi Model**

"Evaluasi model menggunakan data Anda"

Unggah file rar/zip yang berisi folder dari 5 kelas di bawah ini.

Evaluasi tidak akan berhasil jika foldernya kosong

Setiap folder harus berisi setidaknya 1 Gambar

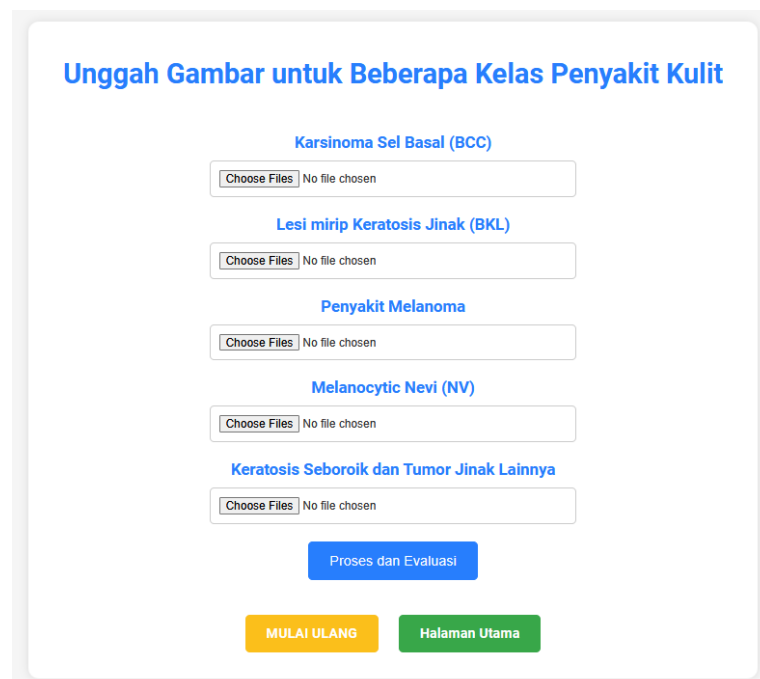
[Klik Di Sini untuk Contoh Zip/Rar](#)

Choose File No file chosen

Unggah dan Evaluasi

Halaman Utama

**Gambar 4. 44.** Halaman evaluasi zip file



**Unggah Gambar untuk Beberapa Kelas Penyakit Kulit**

Karsinoma Sel Basal (BCC)

Choose Files No file chosen

Lesi mirip Keratosis Jinak (BKL)

Choose Files No file chosen

Penyakit Melanoma

Choose Files No file chosen

Melanocytic Nevi (NV)

Choose Files No file chosen

Keratosi Seboroik dan Tumor Jinak Lainnya

Choose Files No file chosen

Proses dan Evaluasi

MULAI ULANG Halaman Utama

**Gambar 4. 45.** Halaman evaluasi multi *images*

Halaman evaluasi model ditampilkan pada Gambar 4.44 dan 4.45. Pada halaman ini, user bisa memilih untuk mengupload berupa file zip, atau *images*. Kemudian model memproses *input* dari user dan memberikan hasil evaluasi berupa *precision*, *recall*, *F1-score*, dan *accuracy*.

## 4.9. Pengujian Sistem

Selanjutnya dilakukan pengujian sistem pada aplikasi berbasis *website*, tujuan utama dalam pengujian ini ialah untuk memastikan bahwa implementasi yang dilakukan sudah berhasil.

### 4.9.1. Black box testing

*Black box testing* ialah suatu metode pengujian pada suatu perangkat lunak yang biasanya memfokuskan pada fungsionalitas sistem tanpa memperhatikan bagaimana sistem tersebut diimplementasikan secara internal. Tujuannya ialah untuk memastikan bahwa semua fungsi dalam aplikasi berbasis *website* seperti pengolahan gambar, klasifikasi dan deteksi, dan lainnya bekerja sesuai dengan yang diharapkan. Hasil pengujian ditampilkan pada Tabel 4.3.


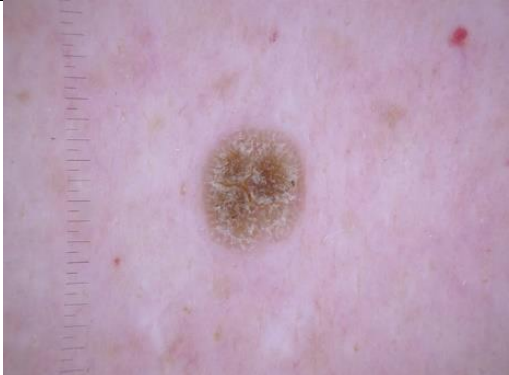

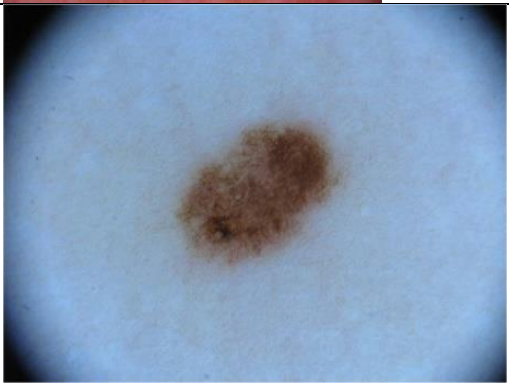
**Tabel 4.3.** Pengujian Fungsional pada *Website*

No	Fitur	Fitur Bekerja (Ya / Tidak)	Keterangan
1	<i>Button</i> menuju program	Ya	Apabila diklik, user akan diarahkan menuju program yang diinginkan
2	<i>Upload image</i>	Ya	Tombol untuk mengupload gambar penyakit kulit yang ingin di klasifikasi serta deteksi oleh pengguna
3	<i>Process image</i>	Ya	Tombol untuk memproses gambar yang telah diupload
4	<i>Button home</i>	Ya	Tombol untuk kembali ke halaman utama
5	<i>Button restart</i>	Ya	Tombol untuk memulai ulang program
6	<i>Translate indonesia</i>	Ya	Tombol untuk mengubah bahasa pada website ke bahasa indonesia
7	<i>Translate english</i>	Ya	Tombol untuk mengubah bahasa pada website ke bahasa inggris
8	<i>Navigation bar</i>	Ya	Fitur navigasi untuk mengarahkan user ke tujuan yang diinginkan

#### 4.9.2. Percobaan klasifikasi dan deteksi

Hasil percobaan klasifikasi dan deteksi pada *website* ditampilkan pada Tabel 4.4.

**Tabel 4.4.** Percobaan Klasifikasi dan Deteksi

No	Gambar	Jenis Penyakit	Kategori
1		Basal Cell Carcinoma (BCC)	Cancer
2		<i>Benign Keratosis-like Lesions</i> (BKL)	Non-Cancer
3		<i>Melanoma</i>	Cancer
4		<i>Melanocytic Nevi</i> (NV)	Non-Cancer

5		<i>Seborrheic Keratoses and other Benign Tumors</i>	Non-Cancer
---	---	---	------------

## **BAB V**

### **PENUTUP**

#### **5.1. Kesimpulan**

Studi ini bertujuan untuk mengembangkan sistem berbasis website yang mampu mendeteksi penyakit kulit secara dini, terutama bagi masyarakat di daerah dengan akses terbatas terhadap layanan kesehatan, menggunakan model EfficientNetB7 untuk klasifikasi dan YOLOv8 untuk deteksi. Berdasarkan temuan penelitian dan analisis perbandingan model, dapat ditarik kesimpulan bahwa:

1. Optimalisasi Algoritma CNN (EfficientNetB7 dan YOLOv8) sebagai Solusi Diagnosis Dini:
  - Kombinasi model EfficientNetB7 untuk klasifikasi dan YOLOv8 untuk deteksi berhasil memberikan solusi yang baik dalam mendiagnosis lima jenis penyakit kulit, seperti *Melanoma*, BCC, NV, BKL, dan *Seborrheic keratoses and other Benign Tumors*. Dimana berdasarkan hasil dari perbandingan model klasifikasi, Model 1 ( dengan pembagian data 8:1:1) terbukti menjadi yang terbaik dengan akurasi pada data pelatihan sebesar 95.41%, validasi sebesar 95.40%, dan testing sebesar 94%, menunjukkan kestabilan dan kemampuan generalisasi yang baik. Model ini tidak menunjukkan tanda-tanda overfitting yang signifikan, menjadikannya lebih unggul dibandingkan Model 2 dan Model 3. Sedangkan pada deteksi, perbandingan model menunjukkan bahwa Model 2 (3000 gambar) merupakan model yang terbaik, dengan mean average precision (mAP) pada data pengujian sebesar 0.812, lebih tinggi dibandingkan Model 1 yang hanya mencapai 0.771. Hal ini menunjukkan bahwa Model 2 lebih akurat dalam mendeteksi objek penyakit kulit pada gambar.
2. Implementasi Berbasis *Website*:
  - Sistem yang dikembangkan, memungkinkan pengguna untuk mengunggah gambar kulit melalui antarmuka *website* dan mendapatkan hasil klasifikasi serta deteksi.
  - Integrasi model klasifikasi dan deteksi ke dalam aplikasi *website* berjalan dengan baik dan responsif terhadap *input* pengguna.

### 3. Pengujian Sistem:

- Pengujian fungsional menggunakan black box testing menunjukkan bahwa semua fitur pada *website* bekerja sesuai dengan fungsinya, termasuk pengunggahan gambar, prediksi klasifikasi dan deteksi, serta penampilan hasil prediksi.
- Pengujian performa model menunjukkan bahwa sistem yang dibangun memiliki kinerja yang baik dan dapat diandalkan dalam mengklasifikasikan dan mendeteksi penyakit kulit.

## 5.2. Saran

Berdasarkan temuan dari hasil studi ini, ada beberapa rekomendasi yang bisa dipertimbangkan untuk pengembangan di masa depan.

1. Mengembangkan aplikasi mobile memungkinkan pengguna untuk mengambil foto kulit langsung menggunakan kamera smartphone mereka dan mendapatkan hasil klasifikasi serta deteksi secara lebih mudah dan cepat. Ini juga meningkatkan aksesibilitas bagi pengguna yang tidak selalu memiliki akses ke komputer.
2. Kerjasama dengan dokter dan tenaga medis profesional sangat penting untuk memastikan bahwa sistem ini dapat digunakan secara efektif dalam lingkungan klinis. Kolaborasi ini juga dapat membantu dalam validasi dan peningkatan akurasi model melalui feedback dan data klinis yang lebih kaya.
3. Teknologi dan metode dalam bidang pembelajaran mesin terus berkembang. Menggunakan arsitektur model yang lebih baru dan teknik pelatihan yang mutakhir dapat meningkatkan performa sistem secara keseluruhan. Penelitian lebih lanjut harus mempertimbangkan untuk selalu mengadopsi inovasi terbaru dalam bidang ini.
4. Kualitas *dataset* sangat mempengaruhi kinerja model. Menggunakan *dataset* yang lebih besar, lebih beragam, dan lebih representatif dapat membantu dalam meningkatkan generalisasi model dan membuatnya lebih *robust* terhadap berbagai kondisi kulit dan variasi gambar.

## DAFTAR PUSTAKA

- Chabi Adjobo, E., Sanda Mahama, A. T., Gouton, P., & Tossa, J. (2023). Automatic Localization of Five Relevant Dermoscopic Structures Based on YOLOv8 for Diagnosis Improvement. *Journal of Imaging*, 9(7). <https://doi.org/10.3390/jimaging9070148>
- Fronzetti, N. (2019). *Predictive Neural Network Applications for Insurance Processes*. <https://www.researchgate.net/publication/335609766>
- Hicks, S. A., Strümke, I., Thambawita, V., Hammou, M., Riegler, M. A., Halvorsen, P., & Parasa, S. (2022). On evaluation metrics for medical applications of artificial intelligence. *Scientific Reports*, 12(1). <https://doi.org/10.1038/s41598-022-09954-8>
- Jaradat, A. S., Al Mamlook, R. E., Almakayeel, N., Alharbe, N., Almuflih, A. S., Nasayreh, A., Gharaibeh, H., Gharaibeh, M., Gharaibeh, A., & Bzizi, H. (2023). Automated Monkeypox Skin Lesion Detection Using Deep Learning and Transfer Learning Techniques. *International Journal of Environmental Research and Public Health*, 20(5). <https://doi.org/10.3390/ijerph20054422>
- Jiang, P., Ergu, D., Liu, F., Cai, Y., & Ma, B. (2021). A Review of Yolo Algorithm Developments. *Procedia Computer Science*, 199, 1066–1073. <https://doi.org/10.1016/j.procs.2022.01.135>
- Khalil, M. I., Tehsin, S., Humayun, M., Jhanjhi, N. Z., & AlZain, M. A. (2022). Multi-scale network for thoracic organs segmentation. *Computers, Materials and Continua*, 70(2), 3251–3265. <https://doi.org/10.32604/cmc.2022.020561>
- Lubis, C., Yulianto, D., Tarumanagara Jakarta, U., Sakit Tiara Tangerang, R., & Kunci, K. (2023). *KLASIFIKASI PENYAKIT KULIT MENGGUNAKAN CONVOLUTIONAL NEURAL NETWORK (CNN) DENGAN ARSITEKTUR VGG16*. 8(1).



- Mupparaju, S., Thotakura, S., & Ch., V. R. R. (2024). A Review on YOLOv8 and Its Advancements. In S. and F.-G. P. Jacob I. Jeena and Piramuthu (Ed.), *Data Intelligence and Cognitive Informatics* (pp. 529–545). Springer Nature Singapore.
- Qureshi, A. S., & Roos, T. (2023). Transfer Learning with Ensembles of Deep Neural Networks for Skin Cancer Detection in Imbalanced Data Sets. *Neural Processing Letters*, 55(4), 4461–4479. <https://doi.org/10.1007/s11063-022-11049-4>
- Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2015). *You Only Look Once: Unified, Real-Time Object Detection*. <http://arxiv.org/abs/1506.02640>
- Reis, D., Kupec, J., Hong, J., & Daoudi, A. (2023). *Real-Time Flying Object Detection with YOLOv8*. <http://arxiv.org/abs/2305.09972>
- Saifan, R., & Jubair, F. (2022). Six skin diseases classification using deep convolutional neural network. *International Journal of Electrical and Computer Engineering*, 12(3), 3072–3082. <https://doi.org/10.11591/ijece.v12i3.pp3072-3082>
- Salvi, M., Acharya, U. R., Molinari, F., & Meiburger, K. M. (2021). The impact of pre- and post-image processing techniques on deep learning frameworks: A comprehensive review for digital pathology image analysis. In *Computers in Biology and Medicine* (Vol. 128). Elsevier Ltd. <https://doi.org/10.1016/j.compbimed.2020.104129>
- Srinivasu, P. N., Sivasai, J. G., Ijaz, M. F., Bhoi, A. K., Kim, W., & Kang, J. J. (2021). Classification of skin disease using deep learning neural networks with mobilenet v2 and lstm. *Sensors*, 21(8). <https://doi.org/10.3390/s21082852>
- Sutaji, D., & Yildiz, O. (2023). *YOLOv8's head-layer Performance Comparison for Skin Cancer Detection*. <https://doi.org/10.46491/icateas.v2i1>
- Tan, M., & Le, Q. V. (2019). *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. <http://arxiv.org/abs/1905.11946>
- Triyono, L., Nur, A., Thohari, A., Hestningsih, I., & Yobioktabera, A. (2022). *KLASIFIKASI PENYAKIT KULIT MENGGUNAKAN METODE CONVOLUTIONAL NEURAL NETWORK*.

Vickers, P., Barrault, L., Monti, E., & Aletras, N. (2024). *We Need to Talk About Classification Evaluation Metrics in NLP*. <http://arxiv.org/abs/2401.03831>

## LAMPIRAN

### Lampiran 1. Kode Program

#### App.py

```
app = Flask(__name__)

# Load models
yolo_model = YOLO("yolo.pt")
skin_model = load_model('efficientnet.h5')

# Class names for skin disease classification
skin_class_names = {
    0: "Basal Cell Carcinoma (BCC)",
    1: "Benign Keratosis-like Lesions (BKL)",
    2: "Melanoma",
    3: "Melanocytic Nevi (NV)",
    4: "Seborrheic Keratoses and other Benign Tumors",
}

@app.route('/upload_image', methods=['GET', 'POST'])
def upload_image():
    if request.method == 'POST':
        y_true = []
        y_pred = []
        errors = []

        for class_id, class_name in skin_class_names.items():
            field_name = f'images_{class_id}'
            if field_name in request.files:
```

```

images = request.files.getlist(field_name)
if len(images) == 0:
    continue # Tidak ada gambar yang diunggah untuk kelas ini

for image_file in images:
    if image_file.filename == "":
        continue # Lewati file kosong

    try:
        predicted_class_name = predict_skin_from_file(image_file)
        predicted_class_index =
list(skin_class_names.values()).index(predicted_class_name)

        y_true.append(class_id)
        y_pred.append(predicted_class_index)
    except Exception as e:
        errors.append(f"Error processing {image_file.filename}: {str(e)}")

if not y_true:
    return render_template('upload_image.html', error='Tidak ada gambar yang
diunggah.', skin_class_names=skin_class_names)

if errors:
    return render_template('upload_image.html', error='; '.join(errors),
skin_class_names=skin_class_names)

# Hitung metrik
precision = precision_score(y_true, y_pred, average='weighted',
labels=list(range(len(skin_class_names))))
recall = recall_score(y_true, y_pred, average='weighted',
labels=list(range(len(skin_class_names))))
f1 = f1_score(y_true, y_pred, average='weighted',
labels=list(range(len(skin_class_names))))

```

```

accuracy = accuracy_score(y_true, y_pred)

metrics = {
    'precision': round(precision, 4),
    'recall': round(recall, 4),
    'f1_score': round(f1, 4),
    'accuracy': round(accuracy, 4)
}

conf_matrix = confusion_matrix(y_true, y_pred)
class_labels = [name for _, name in sorted(skin_class_names.items())]

plt.figure(figsize=(10, 8))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
            xticklabels=class_labels, yticklabels=class_labels)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Confusion Matrix')
conf_matrix_path = 'static/confusion_matrix.png'
plt.savefig(conf_matrix_path)
plt.close()

# Tambahkan path Confusion Matrix ke metrics
metrics['confusion_matrix'] = conf_matrix_path

return render_template('upload_image.html',
                      metrics=metrics,
                      skin_class_names=skin_class_names)

return render_template('upload_image.html', skin_class_names=skin_class_names)

def clear_folder(folder_path):

```

```

for root, dirs, files in os.walk(folder_path, topdown=False):
    for name in files:
        os.remove(os.path.join(root, name))
    for name in dirs:
        os.rmdir(os.path.join(root, name))

# Allowed file extensions
ALLOWED_EXTENSIONS = {'zip', 'rar'}

def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in
ALLOWED_EXTENSIONS

def extract_file(file_path, extract_to):
    if file_path.endswith('.zip'):
        with zipfile.ZipFile(file_path, 'r') as zip_ref:
            zip_ref.extractall(extract_to)
    elif file_path.endswith('.rar'):
        with rarfile.RarFile(file_path, 'r') as rar_ref:
            rar_ref.extractall(extract_to)

def evaluate_model(test_dir):
    # Assuming you already defined img_height, img_width, and batch_size
    test_datagen = ImageDataGenerator(rescale=1. / 255)
    test_generator = test_datagen.flow_from_directory(
        test_dir,
        target_size=(224, 224), # Assuming you are using 224x224 images
        batch_size=32,
        class_mode='categorical',
        shuffle=False,
    )

```

```

predictions = skin_model.predict(test_generator)
y_pred = np.argmax(predictions, axis=1)
y_true = test_generator.classes

conf_matrix = confusion_matrix(y_true, y_pred)
class_labels = list(test_generator.class_indices.keys())
class_report = classification_report(y_true, y_pred, target_names=class_labels)

return conf_matrix, class_report, class_labels

@app.route('/evaluation', methods=['GET', 'POST'])
def evaluate():
    if request.method == 'POST':
        if 'file' not in request.files:
            return render_template('evaluation.html', error='No file part')
        file = request.files['file']
        if file.filename == "":
            return render_template('evaluation.html', error='No selected file')
        if file and allowed_file(file.filename):
            filename = 'testing.' + file.filename.rsplit('.', 1)[1].lower()
            file_path = os.path.join('uploads', filename)
            file.save(file_path)

            # Clear the folder before extraction
            extract_to = 'uploads/extracted/'
            clear_folder(extract_to)

            # Extract the file
            extract_file(file_path, extract_to)

            # Perform evaluation
            conf_matrix, class_report, class_labels = evaluate_model(extract_to)

```

```

# Plot confusion matrix
plt.figure(figsize=(10, 8))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
xticklabels=class_labels,
            yticklabels=class_labels)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.savefig('static/confusion_matrix.png')
plt.close()

return render_template('evaluation.html', class_report=class_report,
                        confusion_matrix='confusion_matrix.png')
return render_template('evaluation.html')

def predict_skin_from_file(file):
    img = Image.open(file)
    img = img.resize((224, 224))
    img_array = image.img_to_array(img)
    img_batch = np.expand_dims(img_array, axis=0)
    img_preprocessed = img_batch / 255.
    prediction = skin_model.predict(img_preprocessed)
    predicted_class_index = np.argmax(prediction)
    predicted_class_name = skin_class_names[predicted_class_index]
    return predicted_class_name

def detect_object_from_image(image_file):
    image = Image.open(image_file)
    results = yolo_model(image, stream=True) # return a generator of Results objects
    # Process results generator
    for result in results:
        result.save(filename='static/result.jpg') # save to disk
    return 'result.jpg'

```



```

@app.route('/')
def landing_page():
    return render_template('landing_page.html')

@app.route('/about')
def about_page():
    return render_template('about.html')

@app.route('/skin_detection', methods=['GET', 'POST'])
def detect():
    if request.method == 'POST':
        if 'image_file' not in request.files:
            return render_template('skin_detection.html', error='No file part')
        image_file = request.files['image_file']
        if image_file.filename == "":
            return render_template('skin_detection.html', error='No selected file')
        if image_file:
            classification_result = predict_skin_from_file(image_file)
            detection_result = detect_object_from_image(image_file)
            return render_template('skin_detection.html',
                                   classification_result=classification_result,
                                   detection_result=detection_result)
    return render_template('skin_detection.html')

if __name__ == "__main__":
    app.run(debug=True)

```

**Lampiran 2. Biodata****BIODATA PENULIS****A. Data Pribadi**

- |                           |                            |
|---------------------------|----------------------------|
| 1. Nama                   | : Alex Mario Simanjuntak   |
| 2. Tempat / Tanggal Lahir | : Medan / 14 April 2002    |
| 3. Jenis Kelamin          | : Laki Laki                |
| 4. Alamat Asal            | : Karya Pembangunan no.79  |
| 5. Telepon / HP           | : 081270005608             |
| 6. Email                  | : alex.mario1404@gmail.com |

**B. Riwayat Pendidikan Formal**

- |  |                   |
|--|-------------------|
| • SD Santa Maria Pekanbaru                       | (2008 – 2014)     |
| • SMP Santa Maria Pekanbaru                      | (2014 – 2017)     |
| • SMA Negeri 1 Pekanbaru                         | (2017 – 2020)     |
| • Universitas Sumatera Utara<br>S1-Ilmu Komputer | (2020 – Sekarang) |