



TUGAS AKHIR - EF234801

KLASIFIKASI DARAH PADA CITRA UJI SILANG SERASI MENGGUNAKAN PENDEKATAN DEEP LEARNING

IDA BAGUS KADE RAINATA PUTRA WIBAWA

NRP 5025201235

Dosen Pembimbing I

Dini Adni Navastara, S.Kom., M.Sc.

NIP 198510172015042001

Dosen Pembimbing II

Prof. Dr. Eng. Chastine Faticah, S.Kom., M.Kom.

NIP 197512202001122002

Program Studi S-1 Teknik Informatika

Departemen Teknik Informatika

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya

2024

[Halaman ini sengaja dikosongkan]



TUGAS AKHIR - EF234801

KLASIFIKASI DARAH PADA CITRA UJI SILANG SERASI MENGGUNAKAN PENDEKATAN DEEP LEARNING

IDA BAGUS KADE RAINATA PUTRA WIBAWA

NRP 5025201235

Dosen Pembimbing I

Dini Adni Navastara, S.Kom., M.Sc.

NIP 198510172015042001

Dosen Pembimbing II

Prof. Dr. Eng. Chastine Faticah, S.Kom., M.Kom.

NIP 197512202001122002

Program Studi S-1 Teknik Informatika

Departemen Teknik Informatika

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya

2024

[Halaman ini sengaja dikosongkan]



FINAL PROJECT - EF234801

BLOOD CLASSIFICATION ON CROSS-MATCHED TEST IMAGES USING A DEEP LEARNING APPROACH

IDA BAGUS KADE RAINATA PUTRA WIBAWA

NRP 5025201235

Advisor I

Dini Adni Navastara, S.Kom., M.Sc.

NIP 198510172015042001

Advisor II

Prof. Dr. Eng. Chastine Faticahah, S.Kom., M.Kom.

NIP 197512202001122002

Study Program Bachelor

Department of Informatics Engineering

Faculty of Intelligent Electrical and Informatics Technology

Institut Teknologi Sepuluh Nopember

Surabaya

2024

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

KLASIFIKASI DARAH PADA CITRA UJI SILANG SERASI MENGGUNAKAN PENDEKATAN DEEP LEARNING

TUGAS AKHIR

Diajukan untuk memenuhi salah satu syarat

memperoleh gelar Sarjana Komputer pada

Program Studi S-1 Teknik Informatika

Departemen Teknik Informatika

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Oleh: **IDA BAGUS KADE RAINATA PUTRA WIBAWA**

NRP. 5025201235

Disetujui oleh Tim Pengaji Tugas Akhir:

1. Dini Adni Navastara, S.Kom., M.Sc.


Pembimbing

2. Prof. Dr. Eng. Chastine Faticah, S.Kom., M.Kom.


Ko-pembimbing

3. Prof. Dr.Eng. Nanik Suciati, S.Kom., M.Kom.


Pengawas

5. Wijayanti Nurul Khotimah, S.Kom., M.Sc.


Pengaji

SURABAYA
Januari, 2024

[Halaman ini sengaja dikosongkan]

APPROVAL SHEET

BLOOD CLASSIFICATION ON CROSS-MATCHED TEST IMAGES USING A DEEP LEARNING APPROACH

FINAL PROJECT

Submitted to fulfill one of the requirements
for obtaining a degree in Computer Science at
Undergraduate Study Program of Informatics

Department of Informatics

Faculty of Intelligent Electrical and Informatics Technology

Institut Teknologi Sepuluh Nopember

By: **IDA BAGUS KADE RAINATA PUTRA WIBAWA**
NRP. 5025201235

Approved by Final Project Examiner Team:

1. Dini Adni Navastara, S.Kom., M.Sc.



Advisor

2. Prof. Dr. Eng. Chastine Faticahah, S.Kom., M.Kom.



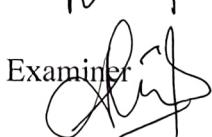
Co-Advisor

3. Prof. Dr.Eng. Nanik Suciati, S.Kom., M.Kom.



Examiner

5. Wijayanti Nurul Khotimah, S.Kom., M.Sc.



Examiner

SURABAYA
January, 2024

[Halaman ini sengaja dikosongkan]

PERNYATAAN ORISINALITAS

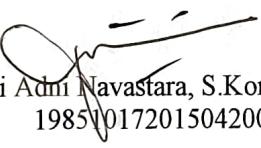
Yang bertanda tangan di bawah ini:

Nama mahasiswa / NRP : Ida Bagus Kade Rainata Putra Wibawa/5025201235
Program studi : S-1 Teknik Informatika
Dosen Pembimbing 1/ NIP : Dini Adni Navastara, S.Kom., M.Sc./
198510172015042001
Dosen Pembimbing 2/ NIP : Prof. Dr. Eng. Chastine Faticah, S.Kom., M.Kom./
197512202001122002

Dengan ini menyatakan bahwa Tugas Akhir dengan judul “Klasifikasi Darah pada Citra Uji Silang Serasi Menggunakan Pendekatan *Deep Learning*” adalah hasil karya sendiri, bersifat orisinal, dan ditulis dengan mengikuti kaidah penulisan ilmiah.
Bilamana di kemudian hari ditemukan ketidaksesuaian dengan pernyataan ini, maka saya bersedia menerima sanksi sesuai dengan ketentuan yang berlaku di Institut Teknologi Sepuluh Nopember.

Surabaya, 7 Februari 2024

Mengetahui
Dosen Pembimbing I


Dini Adni Navastara, S.Kom., M.Sc.
198510172015042001

Mahasiswa


Ida Bagus Kade Rainata Putra Wibawa
5025201235

Dosen Pembimbing II


Prof. Dr. Eng. Chastine Faticah, S.Kom.,
M.Kom.
197512202001122002

[Halaman ini sengaja dikosongkan]

STATEMENT OF ORIGINALITY

The undersigned below:

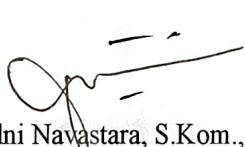
Name of student / NRP : Ida Bagus Kade Rainata Putra Wibawa/5025201235
Department : Informatics
Advisor 1/ NIP : Dini Adni Navastara, S.Kom., M.Sc./
198510172015042001
Advisor 2/ NIP : Prof. Dr. Eng. Chastine Fatichah, S.Kom., M.Kom./
197512202001122002

Hereby declare that the Final Project “Blood Classification on Cross-Matched Test Images Using a Deep Learning Approach” is the result of my own work, is original, and is written by following the rules of scientific writing.

If in the future there is a discrepancy with this statement, then I am willing to accept sanctions in accordance with the provisions that apply at Institut Teknologi Sepuluh Nopember.

Surabaya, 7 Februari 2024

Acknowledge
Advisor I



Dini Adni Navastara, S.Kom., M.Sc
198510172015042001

Student



Ida Bagus Kade Rainata Putra Wibawa
5025201235

Advisor II



Prof. Dr. Eng. Chastine Fatichah, S.Kom.,
M.Kom.
197512202001122002

[Halaman ini sengaja dikosongkan]

ABSTRAK

KLASIFIKASI DARAH PADA CITRA UJI SILANG SERASI MENGGUNAKAN PENDEKATAN DEEP LEARNING

Nama Mahasiswa / NRP : Ida Bagus Kade Rainata Putra Wibawa / 5025201235
Departemen : Teknik Informatika FTEIC - ITS
Dosen Pembimbing : Dini Adni Navastara, S.Kom, M.Sc.

Abstrak

Kecepatan dan akurasi dalam memberikan pelayanan kepada pasien merupakan faktor krusial dalam layanan kesehatan masyarakat. Pelayanan kesehatan, termasuk pelayanan transfusi darah, juga menghadapi tantangan khusus yang memerlukan solusi efektif. Dalam pelayanan transfusi darah, uji silang serasi (pratransfusi) memiliki peran penting untuk mencegah komplikasi pada pasien. Namun, permasalahan utama yang dihadapi adalah proses klasifikasi hasil uji silang serasi yang masih dilakukan secara manual. Hal ini dapat menghambat layanan rumah sakit jika tidak didukung oleh jumlah tenaga kesehatan yang memadai. Oleh karena itu, penelitian ini bertujuan untuk membantu mempercepat pelayanan rumah sakit dan mengalokasikan tenaga kesehatan secara lebih efisien melalui otomatisasi proses klasifikasi hasil uji silang serasi menggunakan *Deep Learning*. Pendekatan *deep learning* yang digunakan adalah *transfer learning* dengan arsitektur *pre-trained* model yang digunakan meliputi InceptionV3, ResNet50, MobileNetV3Small, EfficientNetB7, dan InceptionResNetV2. Tahapan proses pada penelitian ini adalah *preprocessing* citra hasil uji silang serasi, augmentasi citra, segmentasi citra hingga masuk ke tahap klasifikasi untuk setiap *pre-trained* model. *Preprocessing* citra melibatkan proses *cropping* citra dan anotasi citra untuk segmentasi. Sementara itu, arsitektur segmentasi yang digunakan adalah U-Net dengan *backbone* model VGG19. Citra terbagi menjadi *trainset* dan *testset*, dengan rasio 4:1. Berdasarkan hasil uji coba, model EfficientNetB7 memberikan performa terbaik dengan *hyperparameter* Adam sebagai *optimizer*, *learning rate* bernilai 0.01, *hidden units* sebesar 512, dan *dropout rate* bernilai 0.2. Performa model ini adalah nilai *precision* sebesar 0.9461, *recall* sebesar 0.9300, *f1-score* sebesar 0.9374 dan akurasi sebesar 0.9236 pada *testset*.

Kata kunci: *Klasifikasi Citra, Pelayanan Transfusi Darah, Uji Silang Serasi, Transfer Learning, U-Net*

[Halaman ini sengaja dikosongkan]

ABSTRACT

BLOOD CLASSIFICATION ON CROSS-MATCHED TEST IMAGES USING A DEEP LEARNING APPROACH

Student Name / NRP	: Ida Bagus Kade Rainata Putra Wibawa / 5025201235
Department	: Teknik Informatika FTEIC - ITS
Advisor	: Dini Adni Navastara, S.Kom, M.Sc.

Abstract

Speed and accuracy in providing services to patients are crucial factors in public health care. Health services, including blood transfusion services, also face specific challenges that require effective solutions. In blood transfusion services, crossmatch test (pre-transfusion) plays a crucial role in preventing complications in patients. However, the main issue faced is the manual classification process of crossmatch test results. This can hinder hospital services if not supported by an adequate number of healthcare professionals. Therefore, this research aims to help expedite hospital services and allocate healthcare personnel more efficiently through automating the classification process of crossmatch test results using Deep Learning. The deep learning approach used is transfer learning with pre-trained model architectures including InceptionV3, ResNet50, MobileNetV3Small, EfficientNetB7, and InceptionResNetV2. The research process includes preprocessing the crossmatch test result images, image augmentation, image segmentation, and entering the classification stage for each pre-trained model. Image preprocessing involves image cropping and annotation for segmentation. Meanwhile, the segmentation architecture used is U-Net with the VGG19 model as the backbone. The images are divided into a training set and a test set, with a ratio of 4:1. Based on the trial results, the EfficientNetB7 model provides the best performance with hyperparameters including Adam optimizer, a learning rate of 0.01, 512 hidden units, and a dropout rate of 0.2. The performance of this model is a precision value of 0.9461, recall of 0.9300, f1-score of 0.9374, and accuracy of 0.9236 on the test set.

Keywords: *Image Classification, Blood Transfusion Service, Crossmatch Testing, Transfer Learning, U-Net*

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Puji syukur dipanjatkan kepada Tuhan Yang Maha Esa yang telah melimpahkan rahmat-Nya yang melimpah sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul “Klasifikasi Darah pada Citra Uji Silang Serasi Menggunakan Pendekatan *Deep Learning*”.

Tugas Akhir ini dibuat sebagai salah satu persyaratan kelulusan untuk meraih gelar Sarjana Teknik Informatika program Strata Satu (S-1) di Fakultas Teknologi Elektro dan Informatika Cerdas, Institut Teknologi Sepuluh Nopember. Penulis menyadari bahwa dalam penulisan ini tidak terhindar dari kendala-kendala tertentu. Penyelesaian penulisan ini tidak akan mungkin tanpa dukungan dan bantuan yang berharga baik secara moral maupun materiil, serta bimbingan dari berbagai pihak. Oleh karena itu, penulis ingin mengucapkan terima kasih kepada semua pihak yang telah memberikan dukungan dan bantuan dalam penyelesaian perkuliahan dan Tugas Akhir ini, antara lain:

1. Tuhan Yang Maha Esa.
2. Kedua orang tua penulis, Ida Bagus Perang Wibawa dan Ida Ayu Ketut Suciati, kedua saudara penulis, Ida Bagus Gede Subawa Putra dan Ida Bagus Komang Raditya Udayana Putra, dan keluarga besar lainnya yang tidak bisa disebutkan satu per-satu yang senantiasa memberikan dukungan tak terbatas kepada penulis sehingga penulis mampu menyelesaikan perkuliahan ini.
3. Ibu Dini Adni Navastara, S.Kom., M.Sc. selaku pembimbing tugas akhir yang senantiasa dengan sabar memberikan arahan, motivasi, bantuan akademis dengan maksimal.
4. Prof. Dr. Eng. Chistine Fatichah, S.Kom., M.Kom. selaku pembimbing kedua tugas akhir ini yang berperan masif dalam keberhasilan penulisan tugas akhir ini.
5. Seluruh anggota TAOS (*The Art Of Science*) yang selalu hadir dalam pahitnya masalah, manisnya kebahagiaan, dan ikatan persaudaraan yang kuat.
6. Seluruh anggota Burongan TnT yang menjadi teman bertukar pikiran, teman makan, ladang tawa dan kawan patriot perjuangan.
7. Saudari Angela yang sudah menemani penulis dalam menulis memori, memberi energi, dan motivasi tanpa henti.

Saran dan kritik terhadap Tugas Akhir ini sangat diharapkan.

Demikian, semoga Tugas Akhir ini dapat memberikan manfaat bagi pembacanya.

Surabaya, 23 Januari 2024
Ida Bagus Kade Rainata Putra Wibawa

[Halaman ini sengaja dikosongkan]

DAFTAR ISI

LEMBAR PENGESAHAN	i
APPROVAL SHEET	iii
PERNYATAAN ORISINALITAS	v
ABSTRAK.....	ix
ABSTRACT.....	xi
KATA PENGANTAR	xiii
DAFTAR ISI.....	xv
DAFTAR GAMBAR	xvii
DAFTAR TABEL.....	xix
DAFTAR KODE SEMU	xxi
DAFTAR PERSAMAAN	xxiii
BAB 1 PENDAHULUAN	1
1.1 Latar Belakang.....	1
1.2 Rumusan Masalah.....	1
1.3 Batasan Masalah	1
1.4 Tujuan	2
1.5 Manfaat.....	2
BAB 2 TINJAUAN PUSTAKA	3
2.1 Hasil Penelitian Terdahulu	3
2.2 Dasar Teori	5
2.2.1 Image Classification	5
2.2.2 Uji Silang Serasi	5
2.2.3 Image Augmentation pada Citra Uji Silang Serasi.....	6
2.2.4 Image Segmentation pada Citra Uji Silang Serasi.....	7
2.2.5 Transfer Learning untuk Klasifikasi Citra Uji Silang Serasi.....	8
2.2.6 Evaluasi Performa Model Klasifikasi Uji Silang Serasi.....	11
BAB 3 METODOLOGI.....	13
3.1 Deskripsi Umum Sistem.....	13
3.2 Dataset yang Digunakan	13
3.3 Peralatan Pendukung	14
3.4 Perancangan Sistem.....	15
3.4.1 Preparasi Citra Mentah	15
3.4.2 Data Preprocessing	16
3.4.3 Segmentasi Data	18
3.4.4 Train-Test Splitting.....	19
3.4.5 Data Augmentation.....	20
3.4.6 Model Training	21
3.4.7 Evaluasi Performa Model	25
3.5 Implementasi	26
3.5.1 Implementasi Preparasi Raw Data.....	26
3.5.2 Implementasi Data Preprocessing	27
3.5.3 Implementasi Train-Test Splitting.....	29
3.5.4 Implementasi Data Augmentasi.....	30

3.5.5	Implementasi Segmentasi Data	30
3.5.6	Implementasi Model Training	34
3.5.7	Implementasi Testing dan Evaluasi Performa	36
BAB 4	HASIL DAN PEMBAHASAN	43
4.1	Skenario Uji Coba dan Analisis.....	43
4.1.1	Uji Coba Pada Hyperparameter Default.....	44
4.1.2	Uji Coba dengan Hyperparameter Tuning.....	45
4.1.3	Uji Coba Pengaruh Hyperparameter dengan Model Kontrol MobileNetV3Small	47
4.1.4	Uji Coba Perbandingan MobileNetV3Small vs Large	50
4.1.5	Uji Coba Model VGG19 pada Dataset Tersegmentasi.....	50
4.2	Pembahasan	51
4.2.1	Pembahasan Uji Skenario Hyperparameter Default.....	51
4.2.2	Pembahasan Uji Skenario Hyperparameter Tuning	56
4.2.3	Pembahasan Uji Skenario Pengaruh Hyperparameter Pada Model Kontrol MobileNetV3Small	62
4.2.4	Pembahasan Uji Skenario MobileNetV3Small vs MobileNetV3Large	64
4.2.5	Pembahasan Uji Skenario Model VGG19 pada Dataset Tersegmentasi	65
4.2.6	Pembahasan Performa Kelima Pre-Trained Model	66
4.2.7	Pembahasan Dataset Tersegmentasi dalam Peningkatan Performa Kelima Pre-Trained Model	68
BAB 5	KESIMPULAN DAN SARAN	71
5.1	Kesimpulan	71
5.2	Saran	71
DAFTAR PUSTAKA	73	
LAMPIRAN	77	
BIODATA PENULIS	105	

DAFTAR GAMBAR

Gambar 2.1 Contoh Proses Image Classification (Li, 2020)	5
Gambar 2.2 Petunjuk Kategori Penggumpalan Darah	6
Gambar 2.3 Hasil Sebelum dan Sesudah Proses Brightness Adjustment	7
Gambar 2.4 Hasil Segmentasi Sel Leukimia (Sel Original, Segmenasi Sitoplasma, Segmentasi Area dalam) (KHASHMAN & AL-ZGOUL, 2010).....	7
Gambar 2.5 Hasil Segmentasi Citra Uji Silang Serasi.....	8
Gambar 2.6 Arsitektur InceptionV3 (Zehtab-Salmasi, Derakhshi, & Khasmakhi, 2021) ...	8
Gambar 2.7 Arsitektur ResNet50 (serej, 2022).....	9
Gambar 2.8 Arsitektur MobileNetV3 (Yang & Kwon, 2023)	10
Gambar 2.9 Arsitektur EfficientNet (Baheti, 2021).....	10
Gambar 2.10 Arsitektur InceptionResNetV2 (Lin, 2018).....	11
Gambar 3.1 Dataset Mentah.....	14
Gambar 3.2 Dataset yang Sudah Di-crop Beserta Kelasnya (a) Negatif, (b) Positif 1, (c) Positif 2, (d) Positif 3, (e) Positif 4.....	14
Gambar 3.3 Diagram Alur Penelitian	16
Gambar 3.4 Isi file label.json	17
Gambar 3.5 Hasil Anotasi Citra Setiap Kelas (a) Negatif, (b) Positif 1, (c) Positif 2, (d) Positif 3, (e) Positif 4	17
Gambar 3.6 Arsitektur Segmentasi U-Net	19
Gambar 3.7 Augmentasi <i>Flip Horizontal</i> (a) Sebelum Augmentasi <i>Flip Horizontal</i> , (b) Sesudah Augmentasi <i>Flip Horizontal</i>	20
Gambar 3.8 Augmentasi Penambahan Brightness (a) Sebelum Augmentasi Brightness, (b) Sesudah Augmentasi Brightness	21
Gambar 3.9 Rancangan Arsitektur Klasifikasi Secara Umum.....	23
Gambar 3.10 Pratinjau Proses Cropping Data (a) Raw data, (b) Cropped data)	26
Gambar 3.11 Dataframe File label.json	28
Gambar 3.12 Visualisasi Dataset Non-Segmentasi.....	29
Gambar 3.13 Learning Curve Model U-Net pada Proses Segmentasi Data	33
Gambar 3.14 Hasil Citra Segmentasi yang Disimpan dalam Cloud Storage (a). Citra dalam cloud storage (b). Contoh salah satu citra tersegmentasi.....	34
Gambar 3.15 Learning Curve InceptionV3 pada Proses Klasifikasi	37
Gambar 3.16 Learning Curve ResNet50 pada Proses Klasifikasi	37
Gambar 3.17 Learning Curve MobileNetV3Small pada Proses Klasifikasi	38
Gambar 3.18 Learning Curve EfficientNetB7 pada Proses Klasifikasi.....	38
Gambar 3.19 Learning Curve InceptionResNetV2 pada Proses Klasifikasi	39
Gambar 4.1 Confusion Matrix EfficientNetB7 Uji Skenario Hyperparameter Default pada Dataset Non-Segmentasi	51
Gambar 4.2 Citra Kesalahan Prediksi EfficientNetB7 pada Dataset Non-Segmentasi Skenario Hyperparameter Default	52
Gambar 4.3 Confusion Matrix MobileNetV3Small Uji Skenario Hyperparameter Default pada Dataset Tersegmentasi	53
Gambar 4.4 Contoh Citra Kesalahan Prediksi MobileNetV3Small pada Dataset Tersegmentasi Skenario Hyperparameter Default.....	54

Gambar 4.5 Perbandingan Rata-Rata Nilai Metrics Evaluasi pada Dataset Tersegmentasi dan Non-Segmentasi.....	55
Gambar 4.6 Perbandingan F1-Score pada Dataset Tersegmentasi dan Non-Segmentasi..	55
Gambar 4.7 Perbandingan Akurasi pada Dataset Tersegmentasi dan Non-Segmentasi....	56
Gambar 4.8 Confusion Matrix Uji Skenario Hyperparameter Tuning Pada Dataset Tersegmentasi.....	56
Gambar 4.9 Citra Kesalahan Prediksi MobileNetV3Small pada Dataset Non-Segmentasi Skenario Hyperparameter Tuning	57
Gambar 4.10 Confusion Matrix EfficientNetB7 Uji Skenario Hyperparameter Tuning pada Dataset Tergmentasi	58
Gambar 4.11 Contoh Citra Kesalahan Prediksi EfficientNetB7 pada Dataset Tersegmentasi Skenario Hyperparameter Tuning	59
Gambar 4.12 Perbandingan Rata-Rata Nilai Metrics Evaluasi Hyperparameter Tuning pada Dataset Tersegmentasi dan Non-Segmentasi	60
Gambar 4.13 Perbandingan F1-Score Hyperparameter Tuning pada Dataset Tersegmentasi dan Non-Segmentasi.....	60
Gambar 4.14 Perbandingan F1-Score Hyperparameter Tuning pada Dataset Tersegmentasi dan Non-Segmentasi.....	61
Gambar 4.15 Perbandingan F1-Score pada Dataset Non-Segmentasi - Hyperparameter Default vs Tuning	61
Gambar 4.16 Perbandingan F1-Score pada Dataset Tersegmentasi - Hyperparameter Default vs Tuning	62
Gambar 4.17 Perbandingan F1-Score Terbaik pada Uji Evaluasi Hyperparameter antara Dataset Tersegmentasi vs Non-Segmentasi.....	64
Gambar 4.18 Perbandingan Akurasi Terbaik pada Uji Evaluasi Hyperparameter antara Dataset Tersegmentasi vs Non-Segmentasi.....	64
Gambar 4.19 Confusion Matrix Model VGG19 pada Dataset Tersegmentasi	65
Gambar 4.20 Citra Tersegmentasi Kelas Positif 3 yang Salah Diprediksi Model VGG19	65
Gambar 4.21 Perbedaan F1-Score Model VGG19 terhadap Model Lain pada Uji Hyperparameter Default Dataset Tersegmentasi	66
Gambar 4.22 Perbandingan F1-Score Setiap Model pada Skenario 1, 2, 8, 9	67
Gambar 4.23 Perbandingan Akurasi Setiap Model pada Skenario 1, 2, 8, 9	68

DAFTAR TABEL

Tabel 2.1 Ciri – Ciri Penggolongan Hasil Penggumpalan Darah pada Uji Silang Serasi....	6
Tabel 2.2 Representasi Confusion Matrix.....	11
Tabel 3.1 Spesifikasi Peralatan Pendukung	15
Tabel 3.2 Deskripsi Kategori Citra Uji Silang Serasi	15
Tabel 3.3 Layer-layer pada Encoder dan Decoder.....	18
Tabel 3.4 Distribusi Train Set dan Test Set	20
Tabel 3.5 Detail Overview Backbones Pre-trained Model yang Digunakan.....	22
Tabel 3.6 Deskripsi Layer-layer Umum pada CNN	22
Tabel 3.7 Hyperparameter yang Digunakan	25
Tabel 4.1 Rangkuman Skenario Uji Coba yang Dilakukan	43
Tabel 4.2 Variabel Konstan pada masing-masing Skenario	43
Tabel 4.3 Hasil Uji Dataset Non-Segmentasi pada Test Set Hyperparameter Default	45
Tabel 4.4 Hasil Uji Dataset Tersegmentasi pada Test Set Hyperparameter Default	45
Tabel 4.5 Hasil Hyperparameter Tuning Setiap Model pada Dataset Non-Segmentasi	46
Tabel 4.6 Hasil Uji pada Dataset Non-Segmentasi pada Model Hyperparamter Tuning ..	46
Tabel 4.7 Hasil Hyperparameter Tuning pada Setiap Model dengan Dataset Tersegmentasi	47
Tabel 4.8 Hasil Uji pada Dataset Tersegmentasi pada Model Hyperparamter Tuning	47
Tabel 4.9 Hasil Uji Optimizer pada Model MobileNetV3Small pada Dataset Non-Segmentasi.....	48
Tabel 4.10 Hasil Uji Hidden Units pada Model MobileNetV3Small pada Dataset Non-Segmentasi.....	48
Tabel 4.11 Hasil Uji Dropout Rate pada Model MobileNetV3Small pada Dataset Non-Segmentasi.....	48
Tabel 4.12 Hasil Uji Learning Rate pada Model MobileNetV3Small pada Dataset Non-Segmentasi.....	48
Tabel 4.13 Hasil Uji Optimizer Model MobileNetV3Small pada Dataset Tersegmentasi	49
Tabel 4.14 Hasil Uji Hidden Units Model MobileNetV3Small pada Dataset Tersegmentasi	49
Tabel 4.15 Hasil Uji Dropout Rate Model MobileNetV3Small pada Dataset Tersegmentasi	49
Tabel 4.16 Hasil Uji Learning Rate Model MobileNetV3Small pada Dataset Tersegmentasi	50
Tabel 4.17 Hasil Uji Coba Model MobileNetV3Large vs MobileNetV3Small	50
Tabel 4.18 Hasil Uji Model VGG19 pada Dataset Tersegmentasi	51
Tabel 4.19 Perbandingan Arsitektur MobileNetV3 Small vs Large	64
Tabel 4.20 Mean dan Standar Deviasi F1-Score Setiap Model pada Skenario 1, 2, 8, 9 ..	67
Tabel 4.21 Mean dan Standar Deviasi Akurasi Setiap Model pada Skenario 1, 2, 8, 9 ..	67
Tabel 4.22 Nilai Selisih F1-Score pada Performa Terbaik Kedua Dataset.....	69

[Halaman ini sengaja dikosongkan]

DAFTAR KODE SEMU

Kode Semu 3.1 Import Library	27
Kode Semu 3.2 Loading File label.json Menjadi Dataframe.....	27
Kode Semu 3.3 Visualisasi Citra yang Dikelompokkan Berdasarkan Kelasnya	28
Kode Semu 3.4 Implementasi Train-Test Splitting	30
Kode Semu 3.5 Implementasi Data Augmentasi Proses Klasifikasi	30
Kode Semu 3.6 Implementasi Downsampler pada Segmentasi Data	31
Kode Semu 3.7 Implementasi Upsampler pada Segmentasi Data	31
Kode Semu 3.8 Implementasi Kode Instansiasi Model U-Net	32
Kode Semu 3.9 Implementasi Training dan Evaluasi Proses Segmentasi	32
Kode Semu 3.10 Visualisasi Learning Curve Segmentasi.....	33
Kode Semu 3.11 Proses Segmentasi Seluruh Citra dan Penyimpanan ke Cloud Storage .	34
Kode Semu 3.12 Implementasi Model Building.....	35
Kode Semu 3.13 Implementasi Model Fitting	36
Kode Semu 3.14 Implementasi Visualisasi Learning Curve pada Proses Klasifikasi	36
Kode Semu 3.15 Proses Perhitungan Metric Evaluasi pada Proses Klasifikasi	39
Kode Semu 3.16 Implementasi Visualisasi Confusion Matrix	40
Kode Semu 3.17 Implementasi Pencarian Kesalahan Prediksi.....	40
Kode Semu 3.18 Implementasi Fungsi Utilitas Visualisasi Citra Kesalahan Prediksi	40
Kode Semu 3.19 Implementasi Layout Visualisasi Kesalahan Citra Non-Segmentasi....	41
Kode Semu 3.20 Implementasi Layout Visualisasi Kesalahan Citra Tersegmentasi	41

[Halaman ini sengaja dikosongkan]

DAFTAR PERSAMAAN

Persamaan 2.1 <i>Accuracy</i>	12
Persamaan 2.2 <i>Precision</i>	12
Persamaan 2.3 <i>Recall</i>	12
Persamaan 2.4 <i>F1-Score</i>	12

[Halaman ini sengaja dikosongkan]

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Dalam pelayanan kesehatan, kecepatan dan ketepatan dalam melayani pasien menjadi kunci utama (Tuzkaya, Sennaroglu, Kalender, & Mutlu, 2019). Salah satu pelayanan yang penting adalah transfusi darah, di mana uji kecocokan darah antara penerima dan pendonor harus dilakukan. Salah satu metode untuk menguji kecocokan darah adalah uji silang serasi. Uji silang serasi merupakan pemeriksaan yang dilakukan untuk mengevaluasi kompatibilitas antara darah pasien dengan darah donor dengan cara mereaksikan serum/plasma resipien dengan eritrosit donor. Namun, masalah terbesar yang dihadapi adalah membaca dan mendokumentasikan hasil uji silang serasi secara manual. Hal ini dapat menghambat pelayanan rumah sakit, terutama saat permintaan darah tinggi. Oleh karena itu, penelitian ini bertujuan untuk mempercepat pelayanan rumah sakit dengan otomatisasi proses klasifikasi hasil tes uji silang serasi dengan algoritma klasifikasi berbasis *deep learning*, yaitu InceptionV3, ResNet50, MobileNetV3Small, EfficientNetB7, dan InceptionResNetV2.

InceptionV3, ResNet50, MobileNetV3Small, EfficientNetB7, dan InceptionResNetV2 adalah model-model *deep learning* yang telah terbukti efektif dalam tugas klasifikasi citra. InceptionV3 memperkenalkan modul inception yang menggunakan operasi konvolusi paralel dengan ukuran kernel yang berbeda, mencapai kinerja yang sangat baik pada berbagai dataset klasifikasi citra. ResNet50, sebagai salah satu varian dari arsitektur ResNet, dapat menangani masalah vanishing gradient dengan menggunakan koneksi residual, memungkinkan *training* jaringan yang sangat dalam. MobileNetV3Small merupakan varian dari arsitektur MobileNet yang dirancang untuk menjadi model dengan akurasi tinggi dan efisien. MobileNetV3Small menggunakan konvolusi *depthwise separable* untuk mengurangi jumlah parameter dan kompleksitas komputasi (Howard, et al., 2019). EfficientNetB7 mengadopsi pendekatan penskalaan yang terkoordinasi dalam kedalaman, lebar, dan resolusi jaringan, mencapai kinerja yang luar biasa sambil tetap efisien secara komputasi. InceptionResNetV2 menggabungkan keunggulan modul Inception dan koneksi residual untuk meningkatkan performa *training* dan kinerja jaringan saraf yang dalam. Ini memperkenalkan gagasan koneksi residual dalam arsitektur Inception, memungkinkan optimasi yang lebih mudah dari jaringan yang sangat dalam dengan mengatasi masalah gradien yang menghilang (Szegedy, Ioffe, & Vanhoucke, 2016).

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam proposal ini adalah:

1. Bagaimana proses preparasi *dataset* pada klasifikasi citra berbasis *deep learning* untuk membaca hasil uji silang serasi?
2. Bagaimana proses penggunaan model klasifikasi citra berbasis *deep learning* untuk membaca hasil uji silang serasi?
3. Bagaimana performa lima *pre-trained* model klasifikasi citra berbasis *deep learning* (InceptionV3, ResNet50, MobileNetV3Small, EfficientNetB7, dan InceptionResNetV2) pada proses pembacaan hasil uji silang serasi?

1.3 Batasan Masalah

Batasan masalah yang dibahas dalam penelitian ini adalah sebagai berikut:

1. Hasil klasifikasi hanya mencakup - (negatif), +1 (positif satu), +2 (positif dua), +3 (positif tiga), dan +4 (positif empat).
2. Dataset citra yang dipakai diambil hanya menggunakan hasil kartu gel dari RSUD Dr. Soetomo.
3. *Pre-trained* model klasifikasi yang digunakan InceptionV3, ResNet50, MobileNetV3Small, EfficientNetB7, dan InceptionResNetV2.

1.4 Tujuan

Tujuan penelitian ini adalah:

1. Membandingkan performa dari *pre-trained* model klasifikasi citra berbasis *deep learning* (InceptionV3, ResNet50, MobileNetV3Small, EfficientNetB7, dan InceptionResNetV2) pada klasifikasi hasil tes uji silang serasi.
2. Mencari *pre-trained* model klasifikasi citra berbasis *deep learning* terbaik untuk proses klasifikasi hasil uji silang serasi.

1.5 Manfaat

Penelitian ini diharapkan menghasilkan suatu *deep learning* model yang efisien dan akurat untuk menentukan tipe darah pada rumah sakit yang ada di seluruh Indonesia dengan efisien dan akurat.

BAB 2 TINJAUAN PUSTAKA

2.1 Hasil Penelitian Terdahulu

Penyusunan proposal tentang klasifikasi hasil uji silang serasi darah ini melibatkan penelitian-penelitian terdahulu yang berkaitan dengan klasifikasi citra medis. Pertama, penelitian yang berjudul *Medical Image Classification Using Synergic Deep Learning* oleh Zhang, Xie, Wu, dan Xia ini, diusulkan sebuah model *Synergic Deep Learning* (SDL) untuk mengatasi tantangan dalam klasifikasi citra medis yang memiliki variasi intra-kelas yang signifikan dan kesamaan antar-kelas yang disebabkan oleh keragaman modalitas citra dan patologi klinis. SDL menggunakan beberapa *Deep Convolutional Neural Networks* (DCNNs) secara bersamaan dan memungkinkan mereka saling belajar. Setiap pasangan DCNN memiliki representasi citra yang dipelajari dan digabungkan sebagai input dari jaringan sinergis, yang memiliki struktur *fully connected* untuk memprediksi apakah pasangan citra input tersebut termasuk dalam kelas yang sama. Jika satu DCNN melakukan klasifikasi yang benar, kesalahan yang dilakukan oleh DCNN lain menghasilkan kesalahan sinergis yang berfungsi sebagai kekuatan tambahan untuk memperbarui model. Model ini dapat di-*train* secara *end-to-end* dengan supervisi dari kesalahan klasifikasi dari DCNNs dan kesalahan sinergis dari setiap pasangan DCNN. Hasil eksperimen pada dataset ImageCLEF-2015, ImageCLEF-2016, ISIC-2016, dan ISIC-2017 menunjukkan bahwa model SDL yang diusulkan mencapai performa terbaik dalam tugas klasifikasi citra medis ini (Zhang, Xie, Wu, & Xia, 2019).

Penelitian yang berjudul *Classification of Acute Lymphoblastic Leukemia Using Deep Learning* oleh Rehman, et al ini, membahas diagnosis Acute Lymphoblastic Leukemia (ALL) dengan sistem computer melalui teknik pengolahan citra digital dan *deep learning*. Teknik segementasi citra dan jaringan konvolusional digunakan untuk melatih citra sum-sum tulang belakang. Hasil eksperimen kemudian dibandingkan dengan hasil klasifikasi dari metode klasifikasi lainnya seperti Naïve Bayesian, KNN, dan SVM. Hasil eksperimen menunjukkan bahwa metode yang diusulkan mencapai akurasi sebesar 97,78%. Temuan ini menunjukkan bahwa pendekatan yang diusulkan dapat digunakan sebagai alat bantu dalam mendiagnosa Leukemia Akut Limfoblastik dan subtipenya, yang akan membantu para patolog (Rehman, et al., 2018).

Penelitian yang berjudul *White Blood Cells Image Classification Using Deep Learning with Canonical Correlation Analysis* oleh Patil dan Birajdar ini, digunakan Convolutional Neural Network (CNN) dan Recurrent Neural Network (RNN) yang digabungkan dengan metode Canonical Correlation Analysis (CCA) untuk melakukan klasifikasi sel darah. Pendekatan ini mampu mengatasi masalah tumpang tindih sel darah yang sering terjadi dalam citra sampel. Dengan menggunakan metode CCA, waktu klasifikasi dapat dipercepat, dimensi citra input dapat dikompresi, dan jaringan dapat mencapai konvergensi lebih cepat dengan parameter bobot yang lebih akurat. Hasil eksperimen menunjukkan bahwa model yang diusulkan mengungguli teknik klasifikasi sel darah lainnya dalam hal akurasi (Patil & Birajdar, 2020).

Penelitian yang berjudul *Detection of Subtype Blood Cells Using Deep Learning* oleh Tiwari, et al ini, menjelaskan tentang pentingnya sel darah putih WBC (White Blood Cell) dalam sistem kekebalan tubuh serta peran dan jenis-jenisnya. Deteksi dan perbedaan WBC memiliki makna klinis yang penting, namun proses manual yang dilakukan di bawah mikroskop cenderung rumit dan tidak efisien. Oleh karena itu, penelitian ini mengusulkan penggunaan model berbasis Convolutional Neural Network (CNN) untuk mengklasifikasikan jenis-jenis sel

darah secara otomatis. Model tersebut mencapai akurasi sebesar 94% untuk 2 label (polinuklear dan mononuklear) dan 78% untuk 4 label (eosinofil, limfosit, neutrofil, dan monosit) (Tiwari, et al., 2018).

Penelitian yang berjudul Deep Convolutional Neural Network Based Medical Image Classification for Disease Diagnosis oleh Yadav dan Jadhav ini, menjelaskan tentang fokus pada penggunaan algoritma berbasis Convolutional Neural Network (CNN) untuk mengklasifikasikan pneumonia menggunakan dataset sinar-X dada. Tiga teknik dievaluasi melalui eksperimen, yaitu klasifikasi dengan metode Support Vector Machine (SVM) linier menggunakan fitur bebas rotasi dan orientasi lokal, transfer learning pada model Convolutional Neural Network seperti VGG16 dan InceptionV3, serta jaringan kapsul yang di-*train* dari awal. Data augmentation digunakan sebagai metode pra-pemrosesan data untuk ketiga metode tersebut. Hasil eksperimen menunjukkan bahwa data augmentation secara umum efektif dalam meningkatkan performa ketiga algoritma. Transfer learning merupakan metode klasifikasi yang lebih efektif pada dataset kecil dibandingkan SVM dengan fitur ORB (Oriented Fast and Rotated Binary), serta jaringan kapsul. Pada transfer learning, penting untuk melatih ulang fitur-fitur tertentu pada dataset target baru untuk meningkatkan performa. Selain itu, faktor penting lainnya adalah kompleksitas jaringan yang sesuai dengan skala dataset (Yadav & Jadhav, 2019).

Penelitian yang berjudul Deep Convolution Neural Network for Big Data Medical Image Classification oleh Rehan Ashraf, et al ini menjelaskan tentang proses klasifikasi organ dalam tubuh menggunakan model *deep learning*. Penelitian ini menggunakan *pre-trained* model yang mengalami *fine-tuned* pada tiga layer terakhir dari *neural network*. *Pre-trained* model yang digunakan GoogleNet. Organ tubuh yang menjadi kelas target penelitian ini adalah *Prostate*, *Brain*, *Chest*, *Breast*, *Pancreas*, *Colon*, *Soft Tissue*, dan *Esophagus*. Metode yang diajukan pada penilitian ini memberikan akurasi yang cukup tinggi, yaitu 97.73%. Metode yang digunakan mengalahkan beberapa metode *state-of-the-art* yang tersedia (ASHRAF, HABIB, & AKRAM, 2020).

Penelitian yang berjudul Deep Learning Approach For Segmentation and Classification of Blood Cells Using Enhanced CNN oleh B Hemalatha, et al menjelaskan tentang metode untuk menganalisis sel darah dengan menggunakan ECNN (*Enhanced Convolutional Neural Network*). ECNN dalam jurnal ini dijelaskan sebagai kelas dari ANN yang digunakan untuk melatih model dan mengevaluasi ECNN model dengan testing data. *Dataset* yang digunakan dalam penelitian ini diambil dari Kaggle.com yang memiliki 1542 citra. Hal yang ingin disegmentasi pada penelitian ini adalah bagian sel darah pada citra. Selanjutnya, ketika citra sudah disegmentasi, citra diklasifikasi untuk menentukan apakah sel darah merupakan RBC (*Red Blood Cell*) atau sel darah merah. Hasil dari model ECNN yang digunakan pada penelitian ini memiliki akurasi sebesar 95.20% untuk 220 dataset (Hemalatha, Karthik, Reddy, & Latha, 2022).

Penelitian yang berjudul Deep Learning Techniques to Diagnose Lung Cancer oleh Lulu Wang menjelaskan tentang penggunaan *deep learning* untuk mendeteksi kanker pada paru-paru. Penelitian ini menunjukkan metode terkini dalam segmentasi, klasifikasi, dan deteksi kanker paru-paru. Tahapan dimulai dengan mengumpulkan citra CT (*computerized tomography*) pada pasien dengan kanker paru-paru. Selanjutnya dilakukan proses segmentasi untuk mendapatkan area paru paru. Segmentasi ini dilakukan dengan berbagai model terkini, seperti AWEU-Net. Berikutnya, hasil segmentasi akan dimasukkan ke dalam proses *long nodule detection*. *Long nodule detection* ini menggunakan berbagai macam model, salah satunya *Progressive Growing Channel Attentive Non-Local* (ProCAN) *network*. Terakhir, dilakukan klasifikasi *long nodule*

dengan model terkini untuk menentukan apakah kanker merupakan *benign* atau *malignant* (Wang, 2022).

Penelitian yang berjudul Segmentation of Bone Structure With The Use of Deep Learning Techniques oleh Zuzanna KRAWCZYK and Jacek STARZYŃSKI ini menjelaskan tentang proses segmentasi struktur tulang dengan data CT (*computerized tomography*) pada daerah panggul. Penelitian ini membandingkan arsitektur FCN, PSPNet, U-net, dan Segnet dalam segmentasi ini. Hasil segmentasi paling akurat diperoleh dengan menggunakan model U-Net, dengan nilai mIoU sebesar 93.20%. Hasil tersebut lebih ditingkatkan dengan modifikasi model U-net menggunakan ResNet50 sebagai encoder, di-*train* selama 30 epoch, yang menghasilkan nilai mIoU sebesar 96.92%, IoU kelas "tulang" sebesar 92.87%, koefisien mDice sebesar 98.41%, koefisien mDice untuk kelas "tulang" sebesar 96.31%, akurasi m sebesar 99.85%, dan akurasi untuk kelas "tulang" sebesar 99.92% (KRAWCZYK & STARZYŃSKI, 2021).

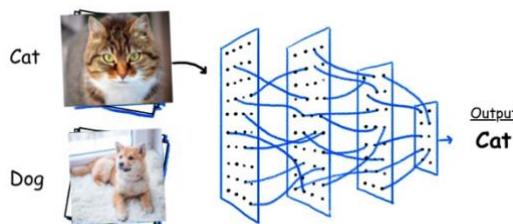
Penelitian yang berjudul Medical Image-Based Detection of COVID-19 Using Deep Convolutional Neural Networks oleh Loveleen Gaur, et al ini menjelaskan tentang proses klasifikasi citra paru-paru yang terkena COVID-19. Penelitian ini menggunakan 3 *pre-trained* model, yaitu EfficientNetB0, VGG16, dan InceptionV3. Hasil menunjukkan bahwa pendekatan yang diusulkan menghasilkan model berkualitas tinggi, dengan akurasi keseluruhan sebesar 92.93%, sensitivitas COVID-19 sebesar 94.79% (Gaur, Bhatia, Jhanjhi, Muhammad, & Masud, 2021).

2.2 Dasar Teori

Bagian ini memaparkan teori-teori yang digunakan sebagai dasar penggerjaan penelitian.

2.2.1 Image Classification

Image classification atau klasifikasi citra digital merupakan proses mengategorikan atau memberi label pada citra ke dalam berbagai kelas atau kategori yang telah ditentukan berdasarkan konten visualnya. Ini merupakan tugas dasar dalam *remote sensing* dan *computer vision*, dengan aplikasi di berbagai bidang seperti analisis citra satelit, pengenalan objek, citra medis, dan lainnya. Biasanya algoritma klasifikasi citra digital berbasis *machine learning* atau *deep learning* (Developers Google, 2022). **Gambar 2.1** menunjukkan proses dari *image classification*.



Gambar 2.1 Contoh Proses Image Classification (Li, 2020)

2.2.2 Uji Silang Serasi

Uji silang serasi atau *crossmatch testing* adalah tes yang dilakukan sebelum pemberian produk darah (pada umumnya kantong darah di RSUD Dr. Soetomo) kepada pasien. Tes ini dilakukan untuk menentukan kecocokan antara darah resipien dengan darah pendonor, lebih tepatnya kecocokan antara antibodi resipien dengan sel darah merah pendonor. Ketidakcocokan pemberian darah kepada resipien dapat mengakibatkan kegumpalan darah, anemia hemolitik, bahkan kematian. Terdapat dua variasi dalam *Crossmatch testing*, yaitu *Major Crossmatch*

Testing dan *Minor Crossmatch Testing*. Dalam *Major Crossmatch Testing*, sel darah donor dimasukkan dalam serum resipien, sedangkan *Minor Crossmatch Testing* melakukan sebaliknya, yaitu sel darah resipien dimasukkan ke dalam serum darah donor. *Crossmatch testing* dilakukan menggunakan kartu gel yang berisi gel dan larutan darah yang sudah sesuai dengan tipe tes yang dicoba. **Gambar 2.2** menunjukkan beberapa kategori hasil dari *crossmatch test*. **Tabel 2.1** mendeskripsikan masing-masing kategori penggumpalan darah dari *crossmatch test*.



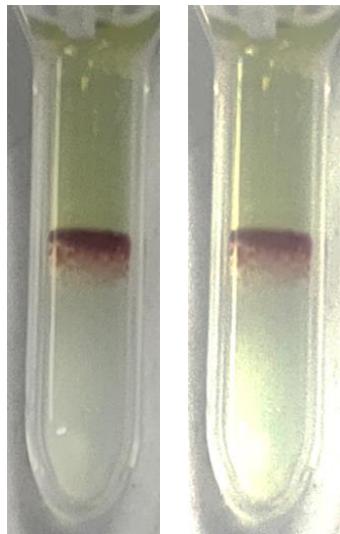
Gambar 2.2 Petunjuk Kategori Penggumpalan Darah

Tabel 2.1 Ciri – Ciri Penggolongan Hasil Penggumpalan Darah pada Uji Silang Serasi

Tipe Penggumpalan	Karakteristik
Negatif (-)	Semua darah berada di dasar tabung terkumpul
Positif satu (+1)	Sebagian darah berada pada dasar tabung, sebagian darah melayang, tidak ada ada darah berada pada permukaan gel
Positif dua (+2)	Tidak ada darah berada pada dasar tabung, semua dalam posisi melayang, tidak ada darah berada pada permukaan gel
Positif tiga (+3)	Sebagian darah berada dalam posisi melayang dan permukaan gel
Positif empat (+4)	Semua darah berada pada bagian permukaan gel

2.2.3 *Image Augmentation* pada Citra Uji Silang Serasi

Image augmentation merupakan salah satu teknik yang umum digunakan dalam *deep learning*, khususnya dalam permasalahan *computer vision* untuk meningkatkan jumlah dan variasi *training dataset*. Hal ini dilakukan dengan menerapkan berbagai jenis transformasi pada citra sehingga menghasilkan suatu variasi citra baru tanpa menghilangkan informasi asli pada citra. Beberapa contoh transformasi pada proses *image augmentation* adalah *flipping*, *rotating*, *scaling*, *brightness*, *contrast*, dan masih banyak lagi (Shorten & Khoshgoftaar, 2019). **Gambar 2.3** menunjukkan hasil citra setelah mengalami proses peningkatan *brightness*.



Gambar 2.3 Hasil Sebelum dan Sesudah Proses *Brightness Adjustment*

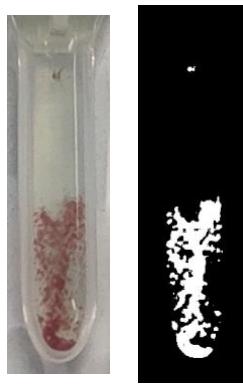
2.2.4 *Image Segmentation* pada Citra Uji Silang Serasi

Segmentasi citra yang merupakan permasalahan esensial dalam bidang *computer vision*, telah menjadi fokus utama sejak awal perkembangannya. Ini merujuk pada proses membagi citra atau *frame* video menjadi segmen-segmen yang berbeda, memisahkan objek-objek yang ada di dalamnya. Keberadaan segmentasi citra sangatlah vital dalam sistem-sistem *visual recognition* yang beragam, dan digunakan secara luas dalam berbagai aplikasi. Contohnya, dalam ranah medis, segmentasi ini digunakan untuk ekstraksi tepi tumor dan penentuan volume jaringan tubuh, lalu dalam domain kendaraan otonom, hal ini diperlukan untuk navigasi yang lebih aman.

Segmentasi citra dapat dilakukan dalam beberapa bentuk, yaitu segmentasi semantik, segmentasi instansi, atau keduanya. Segmentasi semantik melibatkan pelabelan piksel dengan kategori objek seperti manusia, mobil, pohon, dan langit. Ini adalah masalah yang lebih kompleks daripada klasifikasi citra keseluruhan yang hanya memprediksi satu label untuk seluruh citra. Segmentasi instansi memperluas segmentasi semantik dengan mendeteksi dan membatasi setiap objek individu dalam citra, seperti orang-orang secara terpisah. Hal ini telah mengubah paradigma dalam bidang segmentasi citra (Minaee, et al., 2022). **Gambar 2.4** menunjukkan contoh segmentasi pada sel leukimia. Salah satu hasil segmentasi citra uji silang serasi ditunjukkan pada **Gambar 2.5**.



Gambar 2.4 Hasil Segmentasi Sel Leukimia (Sel Original, Segmenasi Sitoplasma, Segmentasi Area dalam) (KHASHMAN & AL-ZGOUL, 2010)



Gambar 2.5 Hasil Segmentasi Citra Uji Silang Serasi

2.2.5 Transfer Learning untuk Klasifikasi Citra Uji Silang Serasi

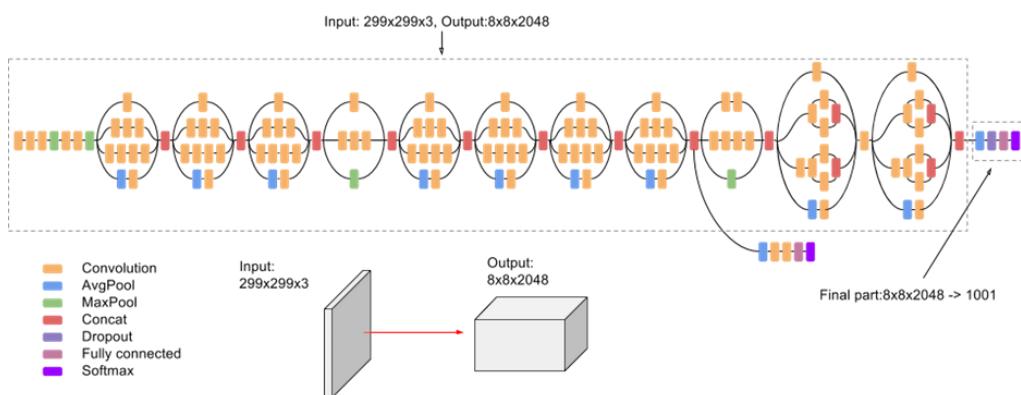
Terinspirasi dari manusia, *transfer learning* dalam konteks *machine learning* merupakan proses transfer pengetahuan suatu model untuk meningkatkan performa pada model lain dalam konteks permasalahan yang sama (WEI, Zhang, Huang, & Yang, 2018). Proses *transfer learning* melibatkan suatu model sumber yang biasa disebut sebagai “*pre-trained model*”.

Pre-trained model merupakan model yang sebelumnya sudah di-*train* dengan *dataset* berukuran besar, khususnya pada permasalahan *image classification* skala besar (Tensorflow, Transfer learning and fine-tuning, 2023). Pemilihan *pre-trained* model penting dilakukan untuk menghasilkan solusi permasalahan yang baik. Penelitian ini menggunakan berbagai *pre-trained* model sebagai berikut.

1. InceptionV3

InceptionV3 adalah model pembelajaran dalam kedalaman (*deep learning*) yang berbasis *Convolutional Neural Networks* (CNN) dan digunakan untuk klasifikasi citra. InceptionV3 merupakan versi yang lebih unggul dari model dasar InceptionV1 yang diperkenalkan sebagai GoogLeNet pada tahun 2014. Model InceptionV3 dikembangkan oleh tim di Google.

Model InceptionV3 terdiri dari beberapa modul Inception. Modul-modul ini memiliki struktur yang kompleks dan memungkinkan penggabungan informasi dari berbagai ukuran filter. Hal ini membantu model untuk memperoleh representasi fitur yang lebih baik dan meningkatkan kinerja dalam tugas klasifikasi citra. InceptionV3 telah terbukti efektif dalam berbagai aplikasi visi komputer dan menjadi salah satu model yang populer digunakan dalam penelitian dan implementasi pengenalan citra (OpenGenus, 2023). **Gambar 2.6** menunjukkan arsitektur InceptionV3.

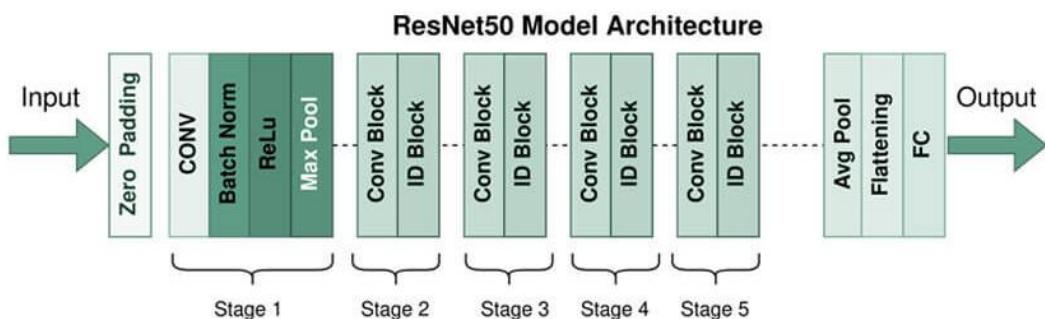


Gambar 2.6 Arsitektur InceptionV3 (Zehtab-Salmasi, Derakhshi, & Khasmakhi, 2021)

2. ResNet50

ResNet-50 merupakan variasi dari arsitektur Residual Network (ResNet), yang merupakan jenis jaringan saraf konvolusional (CNN) yang sangat populer untuk tugas klasifikasi citra. Diperkenalkan oleh Microsoft Research pada tahun 2015, ResNet-50 telah membuktikan keunggulannya dalam berbagai tantangan visi komputer. Keunggulan utama dari ResNet-50 terletak pada penggunaan blok residual, yang mengatasi masalah gradien yang menghilang pada jaringan yang dalam. Blok residual memungkinkan aliran gradien yang efektif melalui lapisan-lapisan jaringan dengan menggunakan koneksi *skip* atau *shortcut*. Dengan demikian, jaringan dapat belajar pemetaan residual dan fokus pada perbedaan antara input dan output, sehingga meningkatkan kemampuan belajar dan kinerja klasifikasi.

ResNet-50 terdiri dari 50 lapisan, termasuk lapisan konvolusi, lapisan *pooling*, lapisan terhubung penuh, dan blok residual. Lapisan awal bertugas mengekstrak fitur tingkat rendah dari citra input melalui operasi konvolusi dan *pooling*. Sementara itu, blok residual yang ditumpuk membentuk inti dari jaringan dalam menciptakan jalur residual yang memungkinkan pembelajaran fitur yang semakin kompleks saat informasi mengalir melalui lapisan-lapisan. Akhirnya, lapisan terakhir ResNet-50 umumnya terdiri dari *pooling* rata-rata global dan lapisan terhubung penuh untuk klasifikasi akhir (Kaushik, 2023). **Gambar 2.7** menunjukkan arsitektur ResNet50.

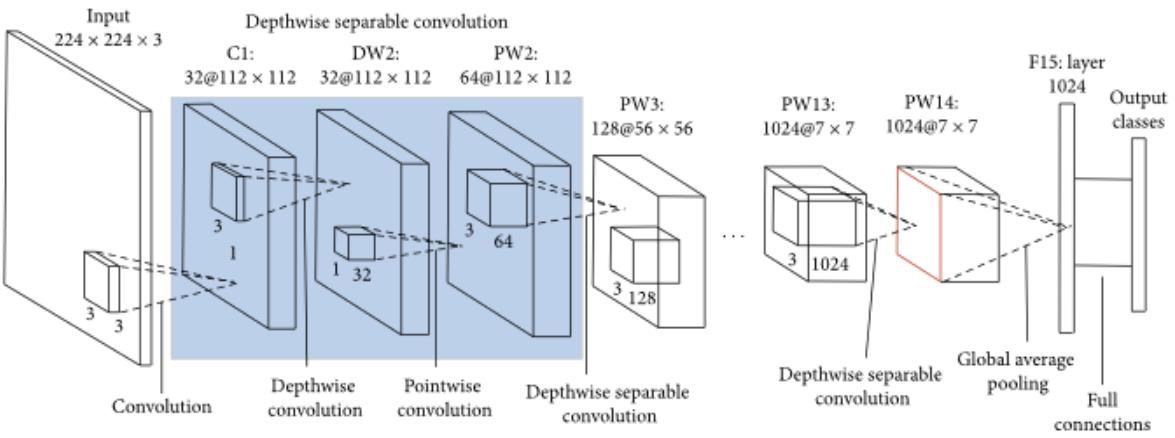


Gambar 2.7 Arsitektur ResNet50 (serej, 2022)

3. MobileNetV3

MobileNetV3 merupakan sebuah jenis arsitektur jaringan saraf konvolusi yang dirancang khusus untuk aplikasi visi pada perangkat *mobile* dan *embedded*. MobileNetV3 menggunakan konvolusi yang terpisah secara kedalaman (*depthwise separable convolutions*) untuk membangun jaringan saraf konvolusi yang ringan namun tetap dalam memori. Pendekatan ini memungkinkan MobileNetV3 memiliki kinerja yang efisien dengan *latency* rendah pada perangkat *mobile* dan *embedded*.

MobileNetV3 menggunakan struktur yang disederhanakan dengan memanfaatkan *depthwise separable convolutions*. *Depthwise separable convolutions* memisahkan proses konvolusi menjadi dua tahap terpisah, yaitu konvolusi pada setiap saluran masukan secara individu (*depthwise convolution*) dan konvolusi berukuran 1x1 untuk menggabungkan informasi dari saluran-saluran tersebut (*pointwise convolution*). Pendekatan ini mengurangi beban komputasi dan jumlah parameter dalam jaringan, sehingga membuat MobileNetV3 menjadi ringan dan efisien untuk aplikasi *mobile* dan *embedded*.



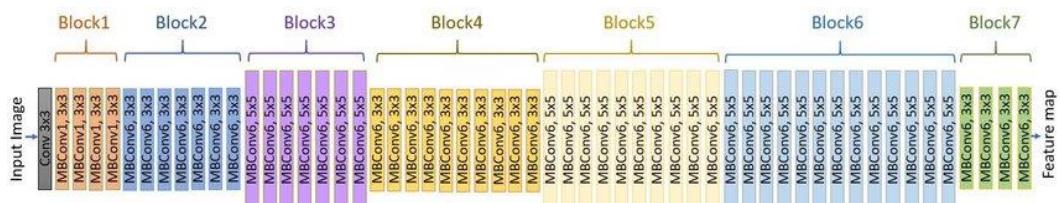
Gambar 2.8 Arsitektur MobileNetV3 (Yang & Kwon, 2023)

Dengan arsitektur yang dioptimalkan ini, MobileNetV3 memberikan solusi yang baik untuk tugas-tugas visi komputer pada perangkat dengan sumber daya terbatas, seperti ponsel cerdas dan perangkat *embedded* (Hindawi, 2023). **Gambar 2.8** menunjukkan arsitektur MobileNetV3.

4. EfficientNetB7

EfficientNet merupakan serangkaian model *Convolutional Neural Network* (CNN) yang dikembangkan dengan menggunakan metode skalabilitas model yang baru dan efisien. Dalam artikel ICML 2019 yang berjudul "*EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*", diusulkan metode skalabilitas model yang menggunakan koefisien gabungan sederhana namun sangat efektif untuk melakukan peningkatan skala pada CNN secara terstruktur. Berbeda dengan pendekatan konvensional yang secara sembarang meningkatkan dimensi jaringan, seperti lebar (*width*), kedalaman (*depth*), dan resolusi citra, metode ini secara seragam meningkatkan setiap dimensi dengan kumpulan koefisien skalabilitas yang tetap.

Metode skalabilitas tersebut, yang telah terbukti dapat meningkatkan akurasi dan efisiensi model secara konsisten, menjadi landasan utama bagi pengembangan teknologi AutoML terkini. Dengan memanfaatkan metode skalabilitas yang baru dan kemajuan terkini dalam AutoML, para peneliti berhasil menghasilkan keluarga model yang dikenal sebagai EfficientNets. Efisiensi dan akurasi yang telah teruji dari EfficientNets membuatnya unggul secara signifikan, dengan kemampuan untuk meningkatkan akurasi hingga 10 kali lebih baik dibandingkan dengan model-model sebelumnya, sambil tetap mempertahankan kecepatan dan ukuran yang lebih kecil (Yu, et al., 2023). **Gambar 2.9** menunjukkan arsitektur EfficientNetB7.

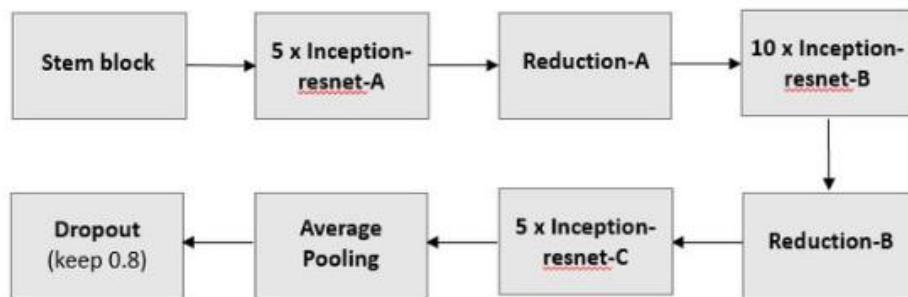


Gambar 2.9 Arsitektur EfficientNet (Baheti, 2021)

5. InceptionResNetV2

Inception-ResNet-v2 adalah arsitektur CNN (*Convolutional Neural Network*) yang di-*train* menggunakan lebih dari satu juta citra dari basis data ImageNet. Jaringan ini memiliki kedalaman 164 lapisan dan dapat mengklasifikasikan citra ke dalam 1000 kategori objek, seperti *keyboard*, *mouse*, pensil, dan banyak hewan lainnya. Sebagai hasilnya, jaringan ini telah belajar representasi fitur yang kaya untuk berbagai macam citra. Jaringan ini memiliki ukuran masukan citra 299 x 299, dan keluarannya berupa daftar probabilitas kelas yang diperkirakan.

Jaringan ini dirumuskan berdasarkan kombinasi struktur Inception dan koneksi residual, yang menggabungkan elemen-elemen kunci dari kedua pendekatan tersebut untuk menciptakan pendekatan yang lebih kuat dan efektif dalam mempelajari fitur-fitur yang kompleks dan hierarkis. Pada blok Inception-Resnet, pendekatan ini memanfaatkan gabungan beberapa *filter* konvolusi dengan ukuran yang berbeda, yang kemudian digabungkan dengan koneksi residual. Penggunaan koneksi residual tidak hanya bertujuan untuk mengatasi masalah degradasi performa yang terjadi akibat dari kedalaman struktur jaringan, tetapi juga memiliki dampak positif dalam mengurangi waktu *training* model karena memungkinkan informasi untuk mengalir lebih lancar melalui lapisan-lapisan jaringan dengan lebih sedikit hambatan.. **Gambar 2.10** menunjukkan arsitektur dasar jaringan InceptionResNetV2 (Elhamraouri, 2020).



Gambar 2.10 Arsitektur InceptionResNetV2 (Lin, 2018)

2.2.6 Evaluasi Performa Model Klasifikasi Uji Silang Serasi

Evaluasi performa pada model *machine learning* maupun *deep learning* adalah aspek yang sangat penting untuk dilakukan. Dalam permasalahan klasifikasi, metrik evaluasi yang umum digunakan adalah *precision*, *recall*, *f1-score* dan *accuracy*. Metriks evaluasi pada permasalahan klasifikasi juga dapat direpresentasikan dengan menggunakan *confusion matrix*. *Confusion matrix* merupakan sebuah matriks yang mendeskripsikan perbedaan antara hasil prediksi dengan hasil aktual. **Tabel 2.2** memberikan gambaran representasi *confusion matrix*.

Tabel 2.2 Representasi *Confusion Matrix*

		Kelas Prediksi	
Kelas Aktual	Benar	Benar	Salah
		True Positive (TP)	False Negative (FN)
	Salah	False Positive (FP)	True Negative (TN)

True Positive (TP) dan *True Negative (TN)* merupakan jumlah kelas positif dan negatif yang diklasifikasikan dengan tepat oleh model, *False Positive (FP)* dan *False Negative (FN)* merupakan jumlah kelas positif dan negatif yang tidak diklasifikasikan dengan tepat. Dari informasi yang diperoleh dari *confusion matrix*, *accuracy*, *precision*, *recall* dan *f1-score* dapat dihitung sesuai dengan **Persamaan 2.1**, **Persamaan 2.2**, **Persamaan 2.3**, dan **Persamaan 2.4** (Kulkarni, Chong, & Batarseh, 2020).

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \quad (2.1)$$

$$Precision = \frac{TP}{(TP + FP)} \quad (2.2)$$

$$Recall = \frac{TP}{(TP + FN)} \quad (2.3)$$

$$F1 = \frac{(2 \times Precision \times Recall)}{(Precision + Recall)} \quad (2.4)$$

Accuracy merupakan metrik yang mengukur sejauh mana model dapat mengklasifikasikan dengan tepat seluruh kelas dari *dataset* yang ada. *Accuracy* menjadi metrik yang optimal jika distribusi kelas pada *dataset* seimbang (Amer, 2022).

Precision merupakan metrik yang mengukur sejauh mana model mampu mengklasifikasi kelas positif dengan tepat dari seluruh kelas positif yang diprediksi oleh model. Dalam permasalahan *image classification*, *precision* penting jika *false negative* tidak menjadi permasalahan. Salah satu kasus dimana *precision* penting adalah *image search engine* yang memerlukan tingkat *precision* tinggi.

Recall (Sensitivity) merupakan metrik yang mengukur sejauh mana model mampu memprediksi kelas positif dengan tepat dari seluruh kelas positif yang sebenarnya. *Recall* penting jika *false positive* tidak menjadi permasalahan. Contoh kegunaan *recall* dalam permasalahan *medical image classification* adalah *tumor detection*. Model *tumor detection* penting untuk memiliki *recall* yang tinggi agar seluruh kasus *tumor* dapat terkласifikasi dengan tepat oleh model (Hasegawa, 2023).

F1-Score merupakan metrik gabungan dari *precision* dan *recall* dengan menggunakan formula rerata harmonik. *F1-Score* mengukur keseimbangan antara *precision* dan *recall* dalam permasalahan klasifikasi. *F1-Score* cocok digunakan untuk *dataset* dengan kelas yang *imbalance* dengan porsi kelas porsi kelas positif yang lebih sedikit (Kovačevića, 2021).

BAB 3 METODOLOGI

Bab ini berisi pembahasan tentang *dataset* yang digunakan, rancangan model, dan implementasi dalam penelitian ini. Pembahasan dimulai dengan deskripsi dan analisis *dataset*, peralatan pendukung, rancangan model dengan *transfer learning* disertai modifikasi *layer*, *hyperparameter tuning* yang digunakan, hingga proses pembuatan implementasi sistem dalam penelitian ini.

3.1 Deskripsi Umum Sistem

Penelitian ini diharapkan mampu menciptakan model yang mampu mengklasifikasikan hasil uji silang serasi dengan akurat sehingga dapat mengautomasi proses pengklasifikasian hasil uji silang serasi. Model ini di-*train* dengan data hasil uji silang serasi yang di dapat dari Bank Darah RSUD Dr Soetomo. Metode yang digunakan dalam penelitian ini adalah *deep learning* menggunakan *transfer learning* dari berbagai *pre-trained* model untuk mengklasifikasikan uji hasil uji silang serasi menjadi kelas Negatif, Positif 1, Positif 2, Positif 3, atau Positif 4. Sebelum, dilakukan proses klasifikasi, data yang diterima dari Bank Darah RSUD Dr Soetomo harus di-*crop* terlebih dahulu, lalu dilanjutkan dengan proses segmentasi menggunakan U-Net dengan encoder VGG19. Adapun ragam *pre-trained* model yang digunakan untuk proses klasifikasi adalah InceptionV3, ResNet50, MobileNetV3Small, EfficientNetB7, dan InceptionResnetV2. Detail spesifik pendekatan *transfer learning* yang diterapkan dijelaskan lebih lanjut pada **subbab 3.4**.

Untuk evaluasi performa, digunakan *accuracy*, *recall*, *precision*, dan *f1-score*. Prioritas evaluasi untuk permasalahan ini adalah *f1-score* dan juga *accuracy*. Pemilihan *f1-score* sebagai prioritas didasarkan karena *f1-score* memberikan keseimbangan antara *precision* dan *recall* yang cocok untuk data yang tidak seimbang (*imbalance dataset*). Jika prioritas utama adalah akurasi, model yang performanya dalam memprediksi suatu kelas rendah (misal Positif 1) tetapi dengan akurasi tinggi, dapat menjadi model terbaik. Model yang diinginkan adalah model yang memiliki performa baik dalam mengidentifikasi kelas-kelas yang ada pada *dataset*. Dengan pemilihan kedua metrik evaluasi ini, diharapkan menghasilkan model yang memiliki kemampuan baik dalam mengklasifikasi hasil uji silang serasi.

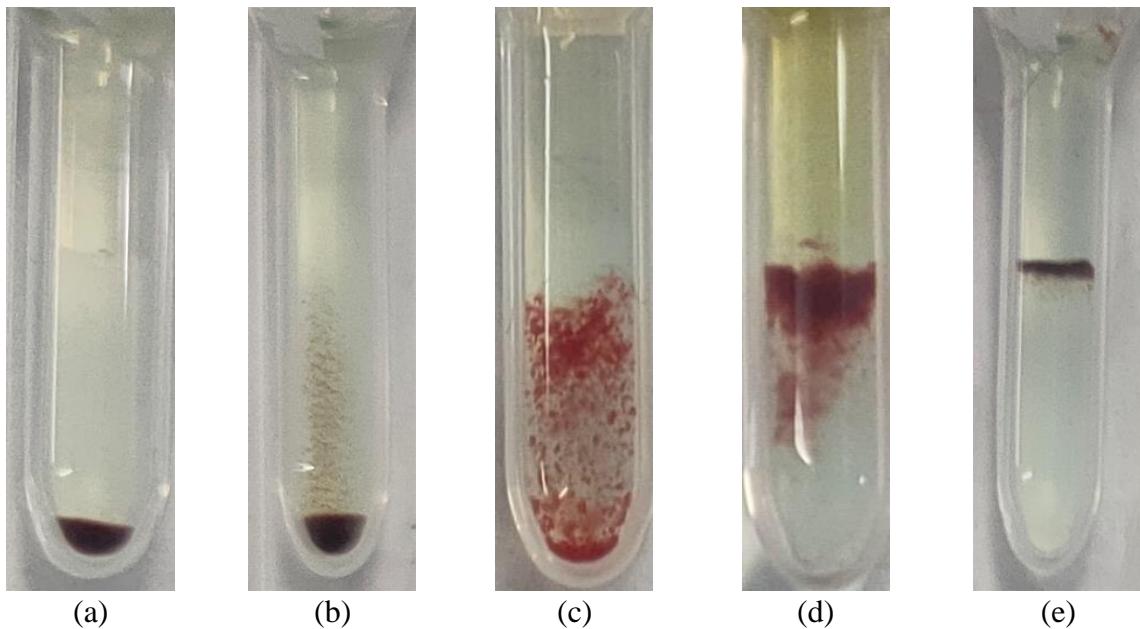
3.2 Dataset yang Digunakan

Dataset yang digunakan di sini merupakan *dataset* berjenis *multiclass* (memiliki kelas lebih dari 2). Kelas-kelas tersebut adalah Negatif, Positif 1, Positif 2, Positif 3, dan Positif 4. *Dataset* ini diambil dari hasil uji silang serasi pada Bank Darah RSUD Dr Soetomo. *Dataset* ini merupakan citra reaksi darah donor terhadap sel pasien pada suatu tabung. Pengambilan digunakan melalui kamera *smartphone*. Pengambilan citra mengharuskan di dalam ruangan dengan pencahayaan yang cukup dan setiap tabung terlihat secara jelas dan detil. Berikut adalah citra yang dijadikan sebagai bahan untuk *training* model *deep learning*. *Dataset* yang dikirim berbentuk tas yang terdiri dari beberapa tabung reaksi. Data ini berjumlah 131 citra yang nantinya mengalami proses *cropping* dan menghasilkan 719 citra. Penjelasan detail terkait dengan proses-proses yang berlangsung dari data mentah hingga tahap *training* model dijelaskan di **subbab 3.4**. **Gambar 3.1** dan **Gambar 3.2** menunjukkan citra mentah dan citra yang sudah mengalami proses *cropping*. *Dataset* yang sudah di-*crop* dibagi menjadi 2 bagian, yaitu data *train* dan data *test*. Data *train* digunakan untuk melatih model dan data *test* digunakan

untuk menguji model. Adapun persentase pembagian *dataset* ini ialah 80% untuk data *train* dan 20% untuk data *test*.



Gambar 3.1 Dataset Mentah



Gambar 3.2 Dataset yang Sudah Di-crop Beserta Kelasnya
(a). Negatif, (b). Positif 1, (c). Positif 2, (d). Positif 3, (e). Positif 4

3.3 Peralatan Pendukung

Tugas pemrograman dalam penelitian ini dilakukan pada sistem awan (*cloud system*) dan juga pada sistem lokal. Penggeraan pada sistem awan dilakukan karena sistem awan memberikan *resource* komputasi yang lebih baik dibandingkan dengan sistem lokal. Dengan alasan tersebut, sistem awan digunakan sebagai media komputasi *training* karena proses ini membutuhkan komputasi yang berat. Selain itu, penggunaan sistem awan juga mempermudah proses *loading* data karena dapat terhubung dengan sistem *cloud storage* secara langsung. **Tabel 3.1** menjelaskan spesifikasi, bahasa pemrograman, serta pustaka pemrograman yang digunakan untuk melakukan penelitian ini.

Tabel 3.1 Spesifikasi Peralatan Pendukung

No	Spesifikasi Sistem	Keterangan
1	Laptop	MacBook Pro M1
2	<i>Processor</i>	M1 Processor
3	Memori (RAM)	8.00 GB
4	<i>Operating System</i>	MacOs Ventura
5	IDE (<i>Integrated Development Env</i>)	<ul style="list-style-type: none">• Local Jupyter Notebook• Google Colab (<i>cloud based</i>)
6	Bahasa Pemrograman	Python 3.11
7	Pustaka Pemrograman	Tensorflow, Numpy, Matplotlib, Pandas

3.4 Perancangan Sistem

Proses penelitian ini meliputi beberapa tahap, yaitu preparasi *raw data*, *preprocessing* data, anotasi data, mengubah data ke dalam bentuk *dataframe*, segmentasi data, *splitting* data, proses augmentasi data, model *building*, model *fitting*, evaluasi performa dan mendapatkan model terbaik (*final model*). **Gambar 3.3** menunjukkan langkah-langkah yang dilakukan pada penelitian ini.

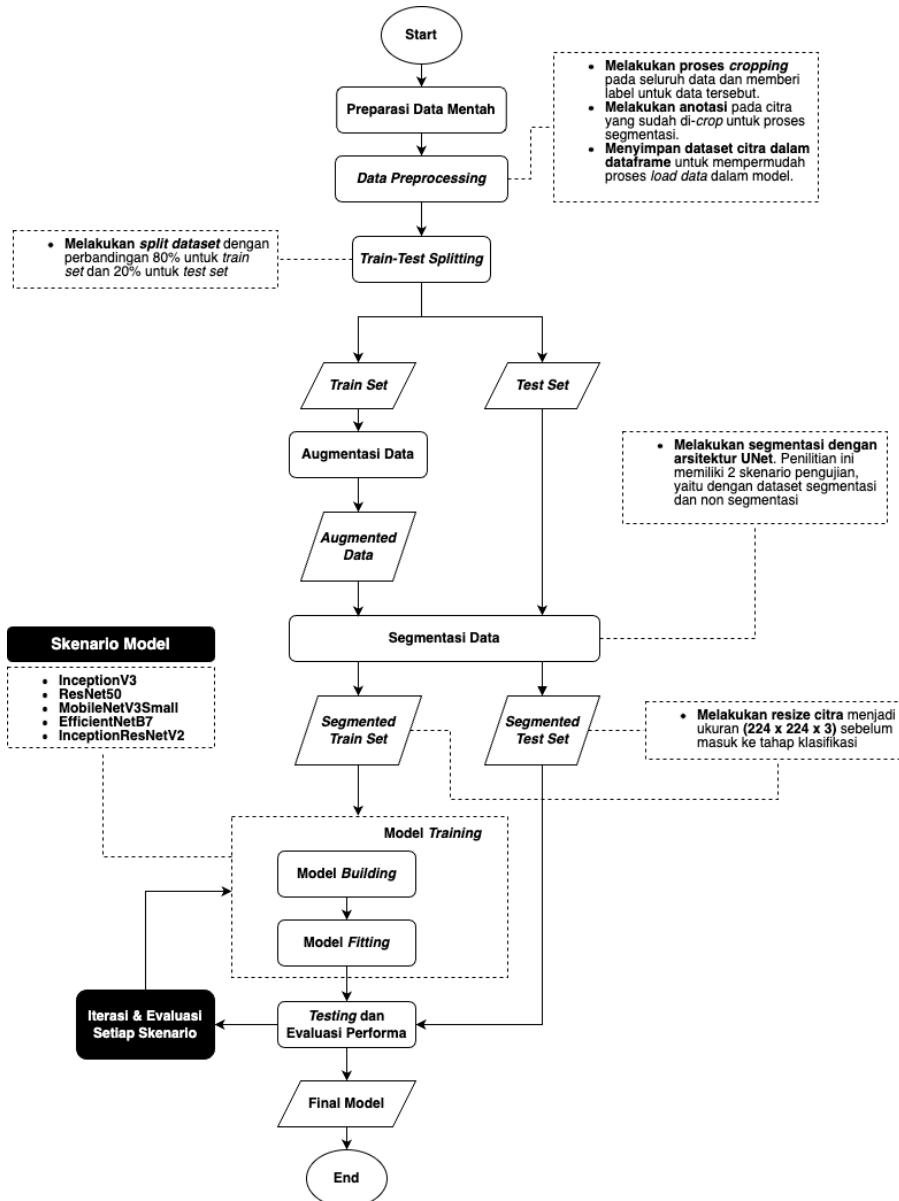
3.4.1 Preparasi Citra Mentah

Total *dataset* hasil uji silang serasi yang didapatkan dari Bank Darah Rumah Sakit Dr.Soetomo berjumlah 131 citra. **Gambar 3.1** pada **subbab 3.2** memperlihatkan citra hasil uji silang serasi yang dimaksud. Sebuah citra pada *dataset* biasanya memiliki 8 buah tabung kecil yang di dalamnya berisi reaksi darah pendonor dengan sel penerima. Reaksi-reaksi ini yang menentukan aman atau tidaknya penerima menerima darah dari pendonor.

Terdapat beberapa kategori yang dihasilkan berdasarkan reaksi darah pada tabung yang nantinya akan menjadi kelas target pada klasifikasi uji silang serasi ini. Kategori atau kelas tersebut diantaranya Negatif, Positif 1, Positif 2, Positif 3, dan Positif 4. Untuk melihat karakteristik citra dengan lebih baik, citra hasil uji silang serasi setiap kelas dapat dilihat pada **Gambar 3.2** pada **subbab 3.2**. **Tabel 3.2** menjelaskan karakteristik dari masing-masing kategori. Tahapan preparasi data ini melibatkan proses penyimpanan seluruh data pada penyimpanan *cloud* untuk mempermudah proses *training*. *Cloud storage* yang digunakan adalah Google Drive dengan alasan mempermudah proses integrasi dengan Google Colab saat penulisan kode program.

Tabel 3.2 Deskripsi Kategori Citra Uji Silang Serasi

Label	Deskripsi
Negatif	Tidak ada bercak darah menyebar. Darah hanya menggumpal pada bagian bawah tabung.
Positif 1	Terdapat bercak darah yang cukup banyak di bagian tengah tabung dengan mayoritas darah terdistribusi pada bagian bawah tabung.
Positif 2	Bercak darah yang terdistribusi secara merata dari bagian bawah hingga bagian tengah tabung. Distribusi bagian bawah dan bagian tengah tabung relatif sama.
Positif 3	Bercak darah terdistribusi pada bagian tengah tabung. Bagian bawah tabung tampak kosong (tidak ada darah). Bercak darah cukup merata.
Positif 4	Darah terdistribusi pada bagian tengah tabung dan jumlahnya lebih sedikit dibandingkan kelas Positif 3. Bagian bawah tabung kosong.

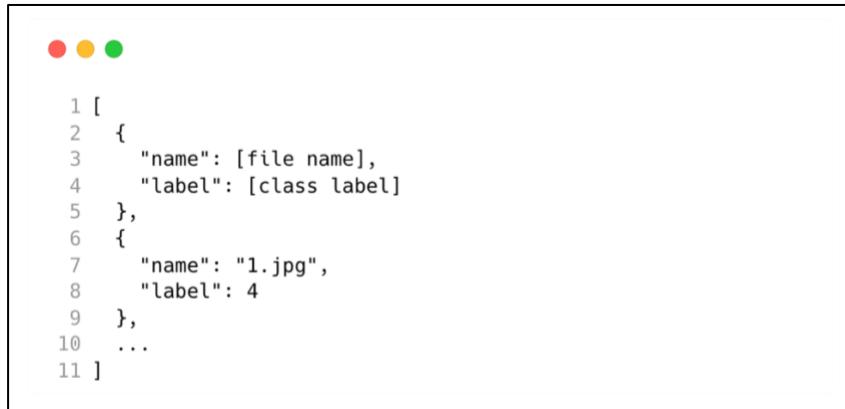


Gambar 3.3 Diagram Alur Penelitian

3.4.2 Data Preprocessing

Sebelum citra-citra di-*train* oleh model. Citra harus diproses terlebih dahulu. Proses pertama yang perlu dilakukan adalah memotong citra hasil uji silang serasi untuk memisahkan masing-masing tabung dan dijadikan sebagai *file* terpisah. Setiap citra pada **Gambar 3.2** menunjukkan sebuah citra yang sudah dipotong dan akan menjadi *file* citra. *File-file* ini berekstensi .jpg dan diberi nama berurutan, yaitu “1.jpg”, “2.jpg”, “3.jpg”, dst. Tujuan pemberian nama berurutan seperti ini adalah agar setiap nama *file* unik dan juga mempermudah proses *labelling*. Proses *labelling* dilakukan dengan cara menyimpan nama file beserta kelasnya dalam bentuk *file* yang berekstensi .json. Format ini menggunakan skema *key-value pairs* dengan *key*-nya yaitu “name” dan “label” dan *value*-nya yaitu nama *file* citra yang sudah dipotong serta angka 0 hingga 4 yang menunjukkan kelas dari *file* citra tersebut. Angka 0 berarti citra memiliki kelas “Negatif”, 1 untuk kelas “Positif 1”, 2 untuk kelas “Positif 2”, 3 untuk kelas “Positif 3”, dan 4 untuk kelas “Positif 4”. **Gambar 3.4** menunjukkan bagaimana penulisan nama *file* citra dan kelas dalam bentuk *file* yang bernama label.json. Tujuan dibuatnya *file*

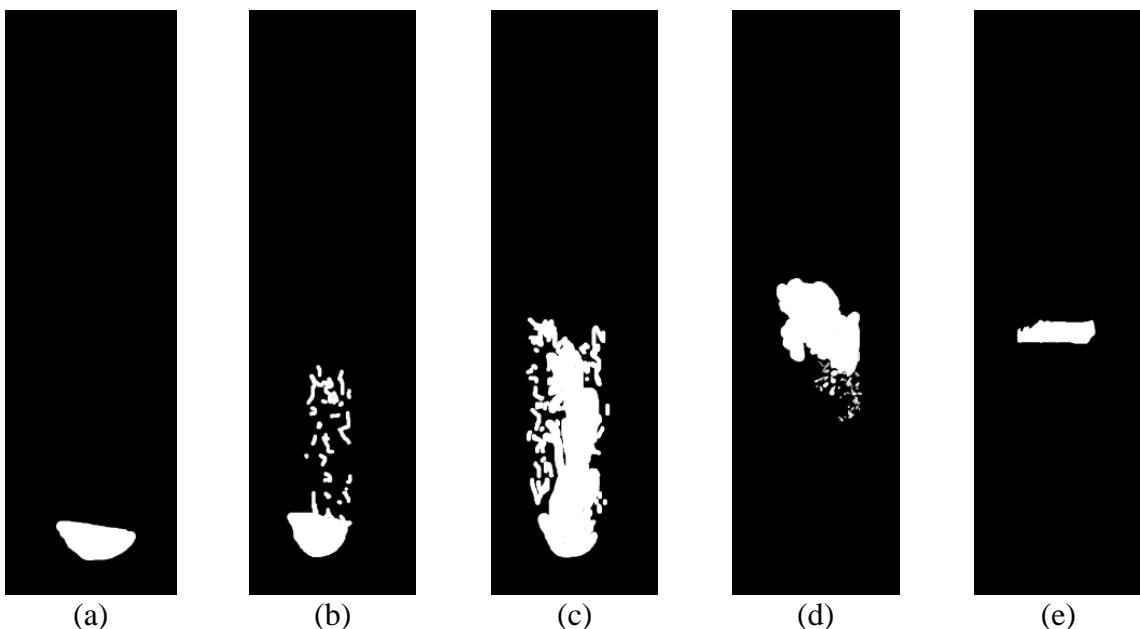
“label.json” ini adalah untuk mempermudah proses pembuatan *dataframe* data saat proses *training* model.



```
1 [
2 {
3     "name": [file name],
4     "label": [class label]
5 },
6 {
7     "name": "1.jpg",
8     "label": 4
9 },
10 ...
11 ]
```

Gambar 3.4 Isi file label.json

Selanjutnya, *file-file* citra yang sudah dipotong melalui proses anotasi piksel. Proses anotasi piksel adalah proses untuk memberi label pada setiap piksel citra. Label yang diberi pada piksel-piksel citra bernilai 0 atau 1 (*binary*) dengan nilai 1 berarti piksel yang mengandung darah dan nilai 0 berarti piksel yang tidak mengandung darah. Tujuan dari anotasi piksel ini adalah untuk memisahkan bagian darah dengan bagian lainnya selain darah (area tabung, area *background*, dsb). **Gambar 3.5** menunjukkan hasil anotasi citra yang setiap *piksel*-nya bernilai 0 atau 1. Citra-citra yang sudah dianotasi disimpan dalam direktori terpisah sebagai *file* berekstensi .jpg dengan nama yang sama seperti *file* sebelum anotasi. Citra-citra hasil anotasi ini menjadi *ground truth* untuk proses segmentasi. Tujuan penamaan yang sama ini adalah agar mempermudah referensi citra *ground truth* terhadap citra asli pada proses evaluasi segmentasi.



Gambar 3.5 Hasil Anotasi Citra Setiap Kelas

- (a). Negatif, (b). Positif 1, (c). Positif 2, (d). Positif 3, (e). Positif 4

3.4.3 Segmentasi Data

Proses segmentasi pada penelitian ini merupakan bagian dari skenario pengujian. Proses segmentasi yang dilakukan dalam penelitian ini adalah *semantic segmentation*. *Semantic segmentation* adalah proses pengelompokan bagian pada citra ke dalam kelas tertentu (Liu, Deng, & Yang, 2018). Di penelitian ini, proses segmentasi berarti proses pemisahan area darah dari area lainnya sehingga hanya terdapat dua kemungkinan kelas pada tiap piksel citra, yaitu 0 untuk area yang tidak mengandung darah dan 1 untuk area yang mengandung darah. Proses segmentasi pada penelitian ini diharapkan dapat membuang area-area yang tidak relevan dalam citra sehingga mempermudah model dalam proses klasifikasi citra nantinya.

Segmentasi ini dilakukan dengan menggunakan arsitektur U-Net dengan model *backbone* VGG19. Arsitektur U-Net digunakan karena performanya yang baik dalam segmentasi citra medis (Chen Z. , 2023). Arsitektur U-Net sendiri merupakan arsitektur yang berbasis *encoder decoder*. Model VGG19 digunakan sebagai *encoder* karena arsitektur konvolusinya yang tidak terlalu rumit. Arsitektur konvolusi yang tidak rumit ini mampu menangkap informasi pada citra, tetapi tidak terlalu detil sampai menangkap informasi yang tidak relevan pada citra. Dalam konteks penelitian ini, diharapkan penggunaan arsitektur VGG19 mampu menangkap area darah tanpa harus menangkap detail-detail lain yang tidak relevan seperti refleksi sinar pada tabung, noda-noda pada *background*, dsb. *Layer-layer* dari model VGG19 yang dipilih sebagai *encoder* berjumlah 5 buah yang detailnya dapat dilihat pada **Tabel 3.3**. Untuk *decoder* pada arsitektur U-Net di penelitian ini, *layer* konvolusi *transpose* digunakan untuk mengembalikan ukuran citra saat fase *encoding* menjadi ukuran citra semula. **Tabel 3.3** menunjukkan detail dari *layer-layer* pada *encoder* dan *decoder* dalam arsitektur U-Net di proses segmentasi penelitian ini. *Layer-layer* pada *encoder* dan *decoder* terhubung satu sama lain dan membentuk *skip connections*. **Gambar 3.6** menunjukkan arsitektur U-Net yang digunakan pada proses segmentasi di penelitian ini.

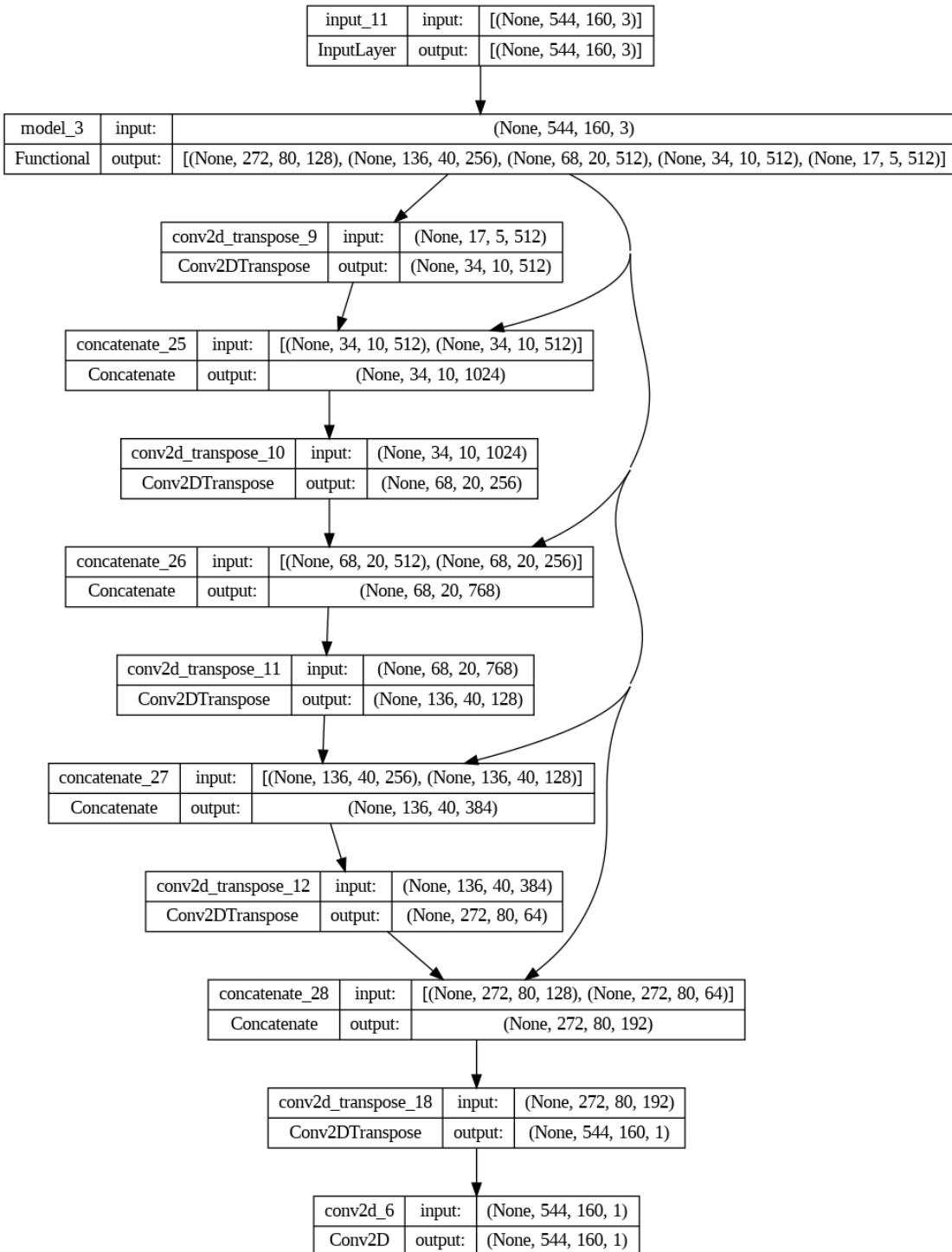
Tabel 3.3 Layer-layer pada *Encoder* dan *Decoder*

<i>Encoder</i>	<i>Decoder</i>
3x3 Conv2D 128 (block2_conv1)	2x2 Conv2D 64 Strides=2
3x3 Conv2D 256 (block3_conv1)	2x2 Conv2D 128 Strides=2
3x3 Conv2D 512 (block4_conv1)	2x2 Conv2D 256 Strides=2
3x3 Conv2D 512 (block5_conv1)	2x2 Conv2D 512 Strides=2

Sebelum data citra digunakan sebagai bahan dalam proses *training* model segmentasi, citra perlu diproses terlebih dahulu. Citra di-*resize* terlebih dahulu menjadi ukuran (544, 160, 1) dari yang sebelumnya berukuran (541, 155, 1). Selanjutnya citra di-*rescale* dengan cara membagi setiap nilai *piksel* dengan 255. Tujuan dari proses *scaling* ini adalah agar proses segmentasi cepat menyetuh titik konvergensi. Citra juga diubah dalam bentuk *grayscale* karena data *ground truth* bersifat *grayscale* sehingga antara data hasil segmentasi dan *ground truth* dapat dibandingkan. Proses *resizing* dan *scaling* dilakukan secara bersamaan dengan menggunakan pustaka “*ImageDataGenerator*” dari Tensorflow. Selain itu, *dataset* juga dibagi menjadi dua bagian, yaitu *train set* dan juga *test set*, dengan rasio 80% untuk *train set* dan 20% untuk *test set*. Proses pembagian ini juga dilakukan pengacakan agar distribusi kelas pada kedua *dataset* ini merata.

Model di-*compile* dengan *loss binary crossentropy* karena proses klasifikasi piksel hanya memiliki 2 kelas target. *Optimizer* yang digunakan adalah Adam dan metrik evaluasinya yaitu

akurasi serta *iou score*. Untuk proses *training*, model di-*train* dengan *epochs* sebesar 20. IOU *score* digunakan untuk melihat seberapa besar *overlap* hasil segmentasi dengan *ground truth*. Semakin besar hasil *overlap* maka semakin dekat juga hasil segmentasi dengan *ground truth*.



Gambar 3.6 Arsitektur Segmentasi U-Net

3.4.4 Train-Test Splitting

Data tersegmentasi dan non-segmentasi harus mengalami proses *splitting* sebelum digunakan dalam proses *training*. Proses *splitting* ini dilakukan dengan membagi *dataset* menjadi 2 bagian, yaitu *train set* dan *test set* dengan rasio 80% *train set* dan 20% *test set*. *Train*

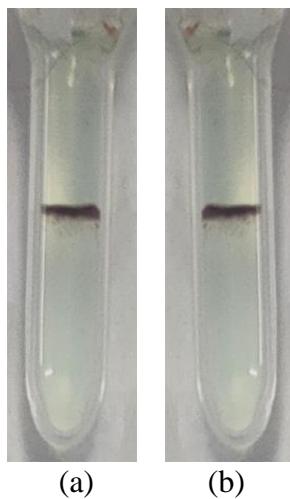
set digunakan sebagai bahan *training* model. Sementara itu, *test set* digunakan sebagai bahan evaluasi model sehabis *training*. Proses pembagian ini dilakukan secara acak dengan *seed* pengacakan yang identik antara data segmentasi dengan non-segmentasi. *Seed* yang sama penting untuk digunakan agar distribusi kelas pada kedua jenis *dataset* sama. **Tabel 3.4** menunjukkan distribusi kelas untuk *train set* dan *test set*.

Tabel 3.4 Distribusi *Train Set* dan *Test Set*

	<i>Train Set</i>	<i>Test Set</i>
Negatif	289	79
Positif 1	119	29
Positif 2	106	24
Positif 3	27	5
Positif 4	34	7
Total	575	144

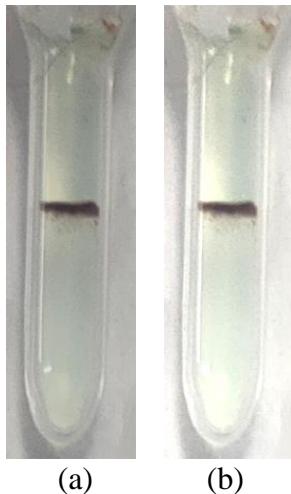
3.4.5 Data Augmentation

Proses augmentasi dilakukan di penelitian ini karena jumlah *dataset* yang relatif sedikit dan dilakukan sebagai strategi untuk menghindari *overfitting*. Augmentasi yang dilakukan pada penelitian ini yaitu *flip horizontal* dan penambahan *brightness* pada citra. *Flip horizontal* merupakan metode augmentasi yang mencerminkan citra terhadap sumbu x. Sementara itu, augmentasi *brightness* adalah pengaturan intensitas cahaya pada citra. Alasan untuk tidak menggunakan bentuk augmentasi lainnya seperti rotasi, *flip vertical*, *shear*, dsb adalah karena adanya kemungkinan ambiguitas *dataset* yang mempengaruhi performa model. Misalnya, pada kasus *flip vertical*, citra dengan kelas Negatif jika mengalami transformasi *flip vertical* mirip dengan citra dengan kelas Positif 4. Oleh karena itu, bentuk augmentasi yang dapat meningkatkan ambiguitas data dihindari. **Gambar 3.7** dan **Gambar 3.8** menunjukkan hasil citra sebelum dan sesudah augmentasi. Dapat dilihat distribusi darah tidak ambigu pada citra sebelum dan sesudah augmentasi.



Gambar 3.7 Augmentasi *Flip Horizontal*

(a) Sebelum Augmentasi *Flip Horizontal*, (b) Sesudah Augmentasi *Flip Horizontal*



Gambar 3.8 Augmentasi Penambahan *Brightness*

(a). Sebelum Augmentasi *Brightness*, (b). Sesudah Augmentasi *Brightness*

Proses augmentasi ini dilakukan dengan bantuan pustaka `ImageDataGenerator` yang dapat melakukan proses augmentasi langsung saat *training* sehingga mengurangi *overhead memory* (Bhandari, 2023). Proses-proses penting lainnya juga berlangsung pada pustaka `ImageDataGenerator` ini. Proses seperti *resizing* citra menjadi dimensi (224 x 224 x 3) piksel dilakukan di `ImageDataGenerator`. Pengubahan ukuran citra menjadi seragam penting untuk dilakukan agar proses *training* model dapat berjalan lancar. Pemilihan dimensi (224 x 224 x 3) piksel didasarkan pada *weights* di *pre-trained* model yang sebelumnya di-*train* dengan citra-citra dengan dimensi (224 x 224 x 3) piksel. *Preprocessing input* juga dilakukan di pustaka `ImageDataGenerator`. Penggunaan berbagai macam *pre-trained* model tentunya memerlukan fungsi *preprocessing input* yang sesuai. Dengan bantuan pustaka `ImageDataGenerator`, penulis mampu mengimplementasikan hal ini dengan mudah. Proses lainnya yang dilakukan di pustaka `ImageDataGenerator` adalah pemilihan *batch size*. *Batching* dilakukan dalam proses *training* model karena mempercepat proses komputasi dan menghemat penggunaan sumber daya komputasi. Nilai *batch size* yang dipilih adalah 32 karena nilai ini memiliki pada umumnya memiliki *error rate* yang rendah dengan waktu komputasi yang cukup efisien (Thakur, 2023).

3.4.6 Model Training

Proses model *training* melibatkan dua tahap, yaitu model *building* dan model *fitting*. Model *building* mendeskripsikan tentang proses penyusunan *pre-trained* model yang dipilih dengan *fully connected layer*. Sementara itu, model *fitting* mendeskripsikan tentang proses kompilasi *pre-trained* model hingga konfigurasi proses *fitting*.

3.4.6.1 Model Building

Penggunaan *transfer learning* dipilih dengan tujuan membangun model yang *robust* dalam mengklasifikasi hasil uji silang serasi. Model-model *pre-trained* yang dipilih adalah InceptionV3, ResNet50, MobileNetV3Small, EfficientNetB7, dan InceptionResNetV2. Pemilihan model-model ini didasari atas performanya yang baik pada citra medis. Detail *overview backbones* model-model ini ditunjukkan pada **Tabel 3.5** yang berisi pengujian dilakukan dengan data dari ImageNet.

Seluruh *pre-trained* model tersebut menggunakan arsitektur yang berbasis CNN (*Convolutional Neural Network*), tetapi dengan susunan dan konfigurasi layer konvolusi yang berbeda-beda. Dengan konfigurasi layer konvolusi yang berbeda-beda, diharapkan dapat

memberikan variasi performa dalam klasifikasi hasil uji silang serasi. **Tabel 3.6** memberikan deskripsi terkait layer-layer umum yang digunakan untuk menyusun CNN.

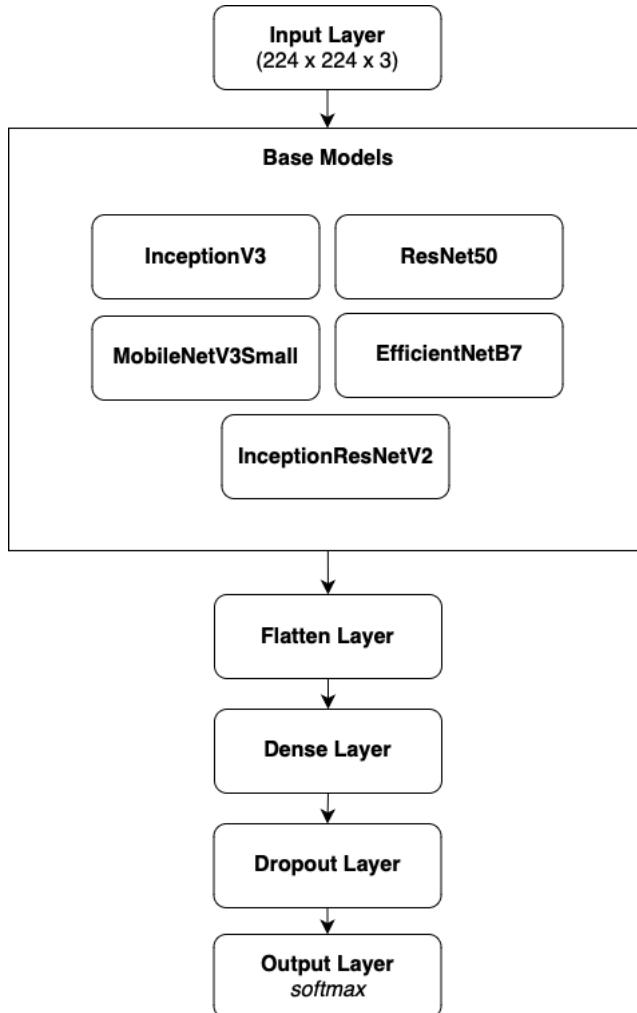
Sebelum *pre-trained* model ini digunakan, perlu dilakukan proses *freezing layer* (pembekuan *layer*) untuk mencegah *weights* dalam sebuah layer untuk di-update (Tensorflow, tensorflow.org, 2023). Proses *freezing layer* ini dilakukan pada seluruh *pre-trained* model yang digunakan. Alasan mengapa *weights* pada *pre-trained* model tidak diperbarui adalah untuk menjaga konsistensi dan memanfaatkan pengetahuan yang telah terakumulasi sebelumnya dari *weights* yang sudah ada. *Pre-trained* model ini memanfaatkan *weights* yang berasal dari data ImageNet yang berjumlah jutaan, yang sudah menjalani proses pembelajaran sebelumnya. Dengan mempertahankan *weights* yang sudah tersimpan dari pembelajaran pada *pre-trained* model tersebut, diharapkan model dapat mempertahankan dan mengoptimalkan pengetahuan yang telah dipelajari dari dataset yang besar tersebut, sehingga dapat menghasilkan performa yang baik dan konsisten dalam tugas yang diberikan.

Tabel 3.5 Detail Overview Backbones *Pre-trained* Model yang Digunakan

Model	Tahun Rilis	Size (MB)	Parameters	Number of Layers	Top-1 Accuracy	Top-5 Accuracy
InceptionV3	2015	92	21.80M	48	0.7790	0.9370
ResNet50	2015	98	25.60M	50	0.7490	0.9210
MobileNetV3Small	2019	9.83	3.58M	16	0.6740	0.8740
EfficientNetB7	2019	256	66.70M	664	0.8440	0.9700
InceptionResNetV2	2016	215	56.00M	164	0.8030	0.9530

Tabel 3.6 Deskripsi Layer-layer Umum pada CNN

Layer	Deskripsi
<i>Input</i>	<i>Layer</i> pertama dalam sebuah model yang digunakan sebagai pintu masuk data yang digunakan untuk proses <i>training</i> .
<i>Rescaling</i>	<i>Layer</i> yang digunakan untuk mengubah skala input. Proses <i>training</i> citra sering kali melibatkan proses ini untuk mempercepat konvergensi model.
<i>Convolutional</i>	<i>Layer</i> yang digunakan untuk melakukan konvolusi pada citra yang berfungsi sebagai pengekstrak fitur-fitur penting dalam citra.
<i>Max Pooling</i>	<i>Layer</i> yang digunakan untuk melakukan operasi pengambilan nilai maksimum dari sejumlah area yang ditentukan pada citra yang diolah, sehingga mengurangi dimensi dan menjaga invariansi terhadap pergeseran dalam citra.
<i>Global Average Pooling</i>	Mirip seperti <i>Max Pooling</i> , <i>layer</i> ini mengambil nilai rata-rata dari sejumlah area yang ditentukan pada citra.
<i>Batch Normalization</i>	<i>Layer</i> yang digunakan untuk proses normalisasi <i>mini-batch</i> dari input citra untuk mempercepat <i>training</i> dan memperbaiki stabilitas model.
<i>Dense</i>	<i>Layer</i> yang umum digunakan sebagai penghubung dari layer-layer ekstraksi fitur ke layer inferensi.
<i>Flatten</i>	<i>Layer</i> yang berfungsi untuk mengubah dimensi data menjadi data 1 dimensi (<i>one dimensional</i>). Biasanya layer ini digunakan sebelum data masuk ke <i>fully connected layer</i> .
<i>Dropout</i>	<i>Layer</i> yang digunakan untuk mengatur ketergantungan antar unit dengan secara acak menonaktifkan sejumlah unit selama <i>training</i> , mencegah overfitting dan meningkatkan generalisasi model.
<i>Output</i>	<i>Layer</i> yang digunakan untuk melakukan proses prediksi sesuai dengan masalah yang ditentukan. Dalam penelitian ini, layer ini memiliki aktivasi <i>softmax</i> karena permasalahan <i>multi-class classification</i> .



Gambar 3.9 Rancangan Arsitektur Klasifikasi Secara Umum

Pre-trained model yang dipilih merupakan model-model yang performanya baik dalam permasalahan klasifikasi citra medis. Hal ini relevan melihat jenis data yang digunakan bertipe citra medis juga. Berikut penjabaran alasan pemilihan setiap model dalam penelitian ini.

1. **InceptionV3:** Berdasarkan jurnal riset yang berjudul “*Fundus image classification using Inception V3 and ResNet-50 for the early diagnostics of fundus diseases*”, model InceptionV3 menunjukkan performa yang baik di angka 91.76% untuk permasalahan *multi-class classification*. Fundus sendiri merupakan bagian dalam mata yang terletak di belakang (Pan1, Liu, Cai, Yang, & Zhang, 2023).
2. **ResNet50:** ResNet50 merupakan model yang sangat lumrah digunakan dalam permasalahan klasifikasi citra. Jurnal yang berjudul “*Deep Learning in Image Classification using Residual Network (ResNet) Variants for Detection of Colorectal Cancer*”, model ResNet mampu mencapai akurasi diatas 80% (Sarwinda, Paradisa, Bustamam, & Anggia, 2021).
3. **MobileNetV3Small:** Model ini dipilih karena memiliki ukuran yang sangat ringan dengan performa yang cukup baik yaitu di angka 67.5% untuk akurasi pada data ImageNet. Model ini dipilih untuk melihat seberapa baik MobileNetV3Small dalam melakukan proses klasifikasi dengan ukuran yang sangat kecil. Dalam paper yang berjudul “*Medical Image Classification Using Transfer Learning and Chaos Game Optimization on the Internet of Medical Things*”, model MobileNetV3 mendapatkan

akurasi sekitar 88% (Mabrouk, Dahou, Elaziz, Redondo, & Kayed, 2022). Dengan alasan tersebut, ingin dilihat performa MobileNetV3Small yang merupakan versi lebih ringan dari MobileNetV3 dalam proses klasifikasi *medical image*.

4. **EfficientNetB7:** Model ini dipilih karena memiliki akurasi di angka 84.3 % pada data ImageNet. Model ini juga digunakan banyak riset medical imgaing dan terbukti memberikan performa yang sangat baik. Berdasarkan jurnal yang berjudul “*MRI-based Brain Tumour Classification Using EfficientNetB7 model with transfer learning*”, model ini mampu memberikan akurasi di angka 98.4% (A, et al., 2023).
5. **InceptionResNetV2:** Model ini dipilih karena memiliki akurasi di angka 80 % pada data ImageNet. Salah satu paper terbaru yang berjudul “*MRI-based Brain Tumor Classification with Inception Resnet V2*” menunjukkan model ini memiliki akurasi di angka 93% (Azzahra, Cerelia, Nugraha, & Pravitasari, 2023).

Selain proses *freezing layer*, *pre-trained* model juga dihubungkan ke *fully connected layer* untuk mendapatkan inferensi. *Fully connected layer* ini tersusun atas *layer Flatten*, *layer Dense* dengan nilai *default* 512 dengan aktivasi *ReLU*, *layer Dropout* dengan nilai *default* 0.5, dan terakhir *layer Dense* dengan jumlah unit 5, sesuai dengan jumlah kelas target dan aktivasi *softmax* untuk kasus *multi-class classification*. **Gambar 3.9** menunjukkan arsitektur *pre-trained* model secara keseluruhan.

3.4.6.2 Model Fitting

Setelah mendefinisikan arsitektur model, langkah selanjutnya adalah mengkompilasi model-model tersebut agar bisa digunakan untuk proses *training*. Proses kompilasi melibatkan penentuan *optimizer*, *loss function*, dan *metrics* yang digunakan. Dalam proses *fitting*, terdapat 2 skenario yang dilakukan, yaitu skenario pertama *fitting (training)* dengan data non-segmentasi dan data segmentasi serta skenario penggunaan *hyperparameter* untuk masing-masing model. *Hyperparameter-hyperparameter* yang digunakan beserta nilainya dijelaskan pada **Tabel 3.7**.

Proses *training* dimulai dengan memasukkan data *train* ke dalam masing-masing model. Citra hasil uji silang serasi mengalami serangkaian proses sesuai dengan arsitektur model yang digunakan. Sebelum masuk ke *layer* pertama suatu model, tentunya citra sudah diproses terlebih dahulu dengan masing-masing *pre-processing* *inp*. Selanjutnya, citra masuk ke *output layer* dan menghasilkan suatu prediksi kelas. Prediksi kelas ini yang digunakan sebagai bahan evaluasi oleh *loss function* untuk menghasilkan *loss score*. *Optimizer* melakukan *weights update* pada setiap *layer trainable* berdasarkan *loss score* dengan langkah *update* sesuai dengan *learning rate*. Proses ini disebut sebagai *backpropagation* dan berlangsung sebanyak 30 *epochs* (30 iterasi).

Selain itu, penulis juga melakukan uji skenario *hyperparameter tuning* menggunakan *library* KerasTuner. Konfigurasi *hyperparameter tuning* dilakukan dengan *cross validation dataset* sebanyak 3 *fold* dan dengan *max trial* pada KerasTuner yang berjumlah 20 *trials*. Ini berarti, proses iterasi pencarian kombinasi dilakukan sebanyak 20 kali. Proses pencarian kombinasi juga dioptimasi dengan BayesianOptimization yang merupakan fitur bawaan dari *library* KerasTuner. Adapun *hyperparameter* yang akan *di-tuning*, yaitu *optimizer*, *hidden units*, *dropout rate* dan *learning rate*. Untuk opsi dari masing-masing *hyperparameter* dijelaskan pada **Tabel 3.7**. Secara keseluruhan total kombinasi *hyperparameter* maksimal berjumlah 81 kombinasi.

Tabel 3.7 Hyperparameter yang Digunakan

Hyperparameter	Nilai dan Deskripsi
<i>Activation Function</i>	<p>[<i>ReLU</i>, <i>Softmax</i>]</p> <ul style="list-style-type: none"> • <i>ReLU</i> (Rectified Linear Unit) adalah fungsi aktivasi yang memiliki properti non-linier pada model <i>deep learning</i> dan menyelesaikan masalah <i>vanishing gradient</i>. <i>Vanishing gradient</i> adalah suatu fenomena nilai <i>gradient</i> nilainya mendekati 0 sehingga tidak terjadi <i>update</i> pada <i>weights</i>. $ReLU(x) = \max(0, x)$
	<ul style="list-style-type: none"> • <i>Softmax</i> adalah fungsi aktivasi yang mengubah skala nilai pada <i>output</i> menjadi probabilitas (seluruh kemungkinan <i>output</i> dijumlahkan menghasilkan nilai 1). $\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}$
<i>Loss Function</i>	<p>[<i>Sparse Categorical Crossentropy</i>]</p> <ul style="list-style-type: none"> • <i>Sparse Categorical Crossentropy</i> mengukur perbedaan antara distribusi probabilitas prediksi dengan distribusi probabilitas aktual (Rahman, 2023). Penggunaan <i>loss function</i> ini sudah sesuai karena representasi kelas target yang berupa <i>integer</i>, bukan <i>one-hot encoded value</i>.
<i>Optimizer</i>	<p>[Adam (<i>Default</i>), Adagrad, RMSProp]</p> <ul style="list-style-type: none"> • Adam merupakan <i>optimizer</i> hasil pengembangan dari <i>Stochastic Gradient Descent</i>. Adam mempertahankan <i>learning rate</i> yang berbeda pada setiap <i>network weights</i> dan memperbarui nilai <i>learning rate</i> secara terpisah saat proses <i>training</i> (Brownlee, 2021). • Adagrad merupakan <i>optimizer</i> yang bersifat <i>parameter-specific learning rates</i> yang proses adaptasi dilakukan relatif terhadap frekuensi <i>update</i> suatu parameter. Semakin sering suatu parameter di-<i>update</i>, semakin kecil perubahan yang terjadi pada parameter tersebut (Soni, 2023) • RMSProp (Root Mean Square Propagation) adalah <i>optimizer</i> yang menghitung perubahan bobot berdasarkan perbedaan rata-rata kuadrat dari <i>gradient</i> sebelumnya. Keunggulan RMSProp terlihat pada data yang telah diaugmentasi dengan variasi yang lebih besar.
<i>Epochs</i>	<p>[30]</p> <ul style="list-style-type: none"> • <i>Epochs</i> merupakan jumlah iterasi yang dilakukan saat proses <i>training</i> model dengan <i>dataset</i>. Penelitian ini menggunakan 30 <i>epochs</i> agar model memiliki waktu yang cukup untuk mempelajari pola-pola pada <i>dataset</i>.
<i>Learning Rate</i>	<p>[0.01 (<i>Default</i>), 0.001, 0.0001]</p> <ul style="list-style-type: none"> • <i>Learning rate</i> menentukan seberapa besar langkah perubahan bobot/<i>weights</i> yang dilakukan oleh <i>optimizer</i>. Nilai <i>learning rate</i> yang besar dapat menyebabkan model belajar dengan cepat, tetapi rentan terhadap divergensi. Sementara itu, nilai <i>learning rate</i> yang kecil menyebabkan model belajar dengan lambat.
<i>Dropout Rate</i>	<p>[0.2, 0.3, 0.5 (<i>Default</i>)]</p> <ul style="list-style-type: none"> • <i>Dropout rate</i> atau <i>dropout ratio</i> merupakan proporsi jumlah neuron yang dinonaktifkan pada suatu layer dalam proses <i>training</i>.
<i>Batch Size</i>	<p>[32]</p> <ul style="list-style-type: none"> • <i>Batch size</i> merupakan sampel data yang diproses dalam satu <i>epoch</i> (iterasi). Nilai 32 dipilih untuk menjaga kesimbangan antara waktu komputasi dan penggunaan memori.

3.4.7 Evaluasi Performa Model

Setelah proses *training* pada setiap *pre-trained* model, dilakukan uji performa terhadap masing-masing model pada setiap skenario yang sudah ditentukan. Pengujian dilakukan dengan *test dataset* yang diperoleh dari proses *train-test splitting*. Metrik performa yang digunakan

adalah *accuracy*, *recall*, *precision*, dan *f1-score*. Pada penelitian ini, nilai prioritas metrik dimulai dari *f1-score* lalu *accuracy*.

Proses evaluasi model juga melibatkan pembuatan *confusion matrix* sebagai bahan evaluasi kelas mana yang model tidak dapat prediksi dengan benar. Selain itu, proses inspeksi citra yang diprediksi salah oleh model juga dilakukan untuk melihat apakah kesalahan memang berasal dari model yang digunakan atau dari sisi label (nilai kelas target) atau dari sisi citra. Tentunya proses evaluasi ini juga dilakukan di seluruh skenario. Untuk kasus skenario dengan menggunakan data tersegmentasi, proses inspeksi citra dilakukan dengan cara mencetak nama file citra yang salah, diikuti dengan kelas target, prediksi kelas target dan dilanjutkan dengan mencetak citra non-segmentasi, citra *ground truth* segmentasi, dan citra prediksi segmentasi. Dengan melakukan hal ini, penulis dapat menginspeksi aspek citra/gambar mana yang salah. Sementara itu, untuk citra non-segmentasi, proses inspeksi dilakukan dengan mencetak nama file yang salah, diikuti dengan kelas target, kelas prediksi, lalu citra non-segmentasi yang diprediksi.

3.5 Implementasi

Pada subbab ini akan dijelaskan mengenai tahapan implementasi *multi-class classification* pada dataset uji silang serasi yang menggunakan pendekatan *deep learning*. Implementasi ini mengacu pada **Gambar 3.3** yaitu diagram penelitian. Dalam proses implementasi, mayoritas aktivitas yang terlibat adalah proses penulisan kode program. Dalam subbab ini kode program dituliskan dalam bentuk kode semu (*pseudocode*). Penulisan kode program dilakukan pada kegiatan *importing library*, *loading dataframe* citra, visualisasi citra, proses augmentasi, proses segmentasi, konfigurasi model, konfigurasi *hyperparameter* model, kompilasi model, hingga sampai pada tahap *training* dan evaluasi model.

3.5.1 Implementasi Preparasi Raw Data

Preparasi *raw data* melibatkan proses *cropping* pada citra uji silang serasi. *Cropping* dilakukan pada citra uji silang serasi hingga menjadi citra yang hanya berisi satu buah tabung gel. Pratinjau data sebelum dan sesudah *cropping* ditunjukkan pada **Gambar 3.10**. Setelah proses *cropping* selesai, citra hasil *crop* akan diberi label sesuai dengan kelasnya. Pelabelan dilakukan harus dilakukan pada proses klasifikasi. Pelabelan dilakukan dengan menulis nama file citra serta labelnya di dalam file berekstensi JSON. Hal ini dilakukan untuk mempermudah proses *loading dataframe* citra. *Dataframe* citra nantinya berisi nama file dan label citra tersebut.



Gambar 3.10 Pratinjau Proses *Cropping* Data

(a). *Raw data*, (b). *Cropped data*

3.5.2 Implementasi Data Preprocessing

Proses implementasi selanjutnya yang dilakukan adalah *importing library*. *Library* atau pustaka yang digunakan sesuai dengan uji skenario yang akan dilakukan. Secara umum, *library* yang digunakan ditunjukkan pada **Kode Semu 3.1**.

```
1 import numpy as np
2 import tensorflow as tf
3 import matplotlib.image as mpimg
4 import matplotlib.pyplot as plt
5 import seaborn as sns
6 import pandas as pd
7 import os
8 from tensorflow.keras.preprocessing.image import ImageDataGenerator
9 from datetime import datetime
10 from sklearn.metrics import f1_score, precision_score, recall_score,
11 accuracy_score
```

Kode Semu 3.1 Import Library

Kode mengimpor *library-library* utama yang digunakan dalam *load* data, analisis data, *preprocessing data*, segmentasi data, model *building*, model *training*, hingga evaluasi performa. *Library numpy* digunakan untuk mengolah matriks atau *array*. *Tensorflow* digunakan sebagai *library* untuk membangun dan melatih model. *Matplotlib* dan *seaborn* digunakan sebagai alat untuk visualisasi citra maupun grafik. *Pandas* digunakan untuk bekerja dengan data tabular yang nantinya akan berisi nama file dan target kelasnya. *Library os* atau *operating system* digunakan untuk berinteraksi dengan *file* pada sistem komputer yang digunakan. *Library* ini nantinya akan digunakan dalam tahap penyimpanan model. *ImageDataGenerator* digunakan untuk proses augmentasi dan *training* pada citra. *Datetime* digunakan sebagai alat untuk mencatat waktu *training*. Terakhir, *sklearn* digunakan sebagai pustaka untuk evaluasi performa model.

Setelah seluruh *library* sudah diimpor, tahap berikutnya dilakukan proses *loading file* *label.json* yang akan dimuat dalam bentuk *dataframe* menggunakan *library pandas*. **Kode Semu 3.2** menunjukkan kode pemuatan *file label.json* ke dalam *dataframe*. Baris ke-1 pada **Kode Semu 3.2** menunjukkan proses *load file* *label* dengan menggunakan fungsi “*read_json*” dari *library pandas*. Selanjutnya, pada baris ke-8 hingga ke-10, dilakukan proses *mapping* indeks sesuai dengan kelasnya. Misalnya, indeks 0 di-*map* dengan kelas Negatif, indeks 1 dengan kelas Positif 1, dan seterusnya. Kode ini akan menghasilkan *dataframe* yang tampak pada **Gambar 3.11**. *Dataframe* ini nanti akan digunakan pada *ImageDataGenerator* *library* untuk proses *fetching* citra saat *training*.

```
1 DF <- read_json("/content/drive/MyDrive/Image
2 Segmentation/label.json")
3 class_map <- {0: 'Negatif', 1: 'Positif 1', 2: 'Positif 2', 3:
4 'Positif 3', 4: 'Positif 4'}
5
6 # Add a new column 'class' to the data frame based on the 'label'
7 column
8 FOR index, row in DF.iterrows():DO
9     label <- row['label']
10    DF.at[index, 'class'] <- class_map[label]
11 ENDFOR
```

Kode Semu 3.2 Loading File label.json Menjadi Dataframe

	name	label	class
0	1.jpg	4	Positif 4
1	2.jpg	4	Positif 4
2	3.jpg	4	Positif 4
3	4.jpg	4	Positif 4
4	5.jpg	4	Positif 4

Gambar 3.11 Dataframe File label.json

Tahap selanjutnya adalah menampilkan *dataset* yang sudah di-*crop* dengan menggunakan *matplotlib*. Penampilan citra dilakukan dengan mengelompokkan citra sesuai dengan kelasnya. Hal ini dilakukan agar penulis dapat melihat karakteristik citra sesuai dengan kelasnya. Proses visualisasi ini dilakukan dengan tujuan untuk melihat karakteristik citra pada masing-masing kelas. Hal ini juga dapat berguna sebagai Tindakan awal dalam menginvestigasi adanya kesalahan *labelling* pada citra. **Kode Semu 3.3** menunjukkan proses visualisasi citra yang sudah dikelompokkan sesuai dengan kelasnya. Pada kode ini, jumlah citra yang diambil pada masing-masing kelas berjumlah 4 citra. Citra-citra tiap kelas akan di-*plot* pada suatu *subplot* dalam suatu *figure* pada *matplotlib*. Proses ini dilakukan di dalam *for loop* dengan *loop* pertama pada baris ke-9 berfungsi untuk instansiasi setiap *figure* kelas. *Loop* kedua yang ditunjukkan pada baris ke-20 berfungsi untuk menampilkan setiap citra pada masing-masing kelas. **Gambar 3.12** memperlihatkan hasil visualisasi citra yang dilakukan.

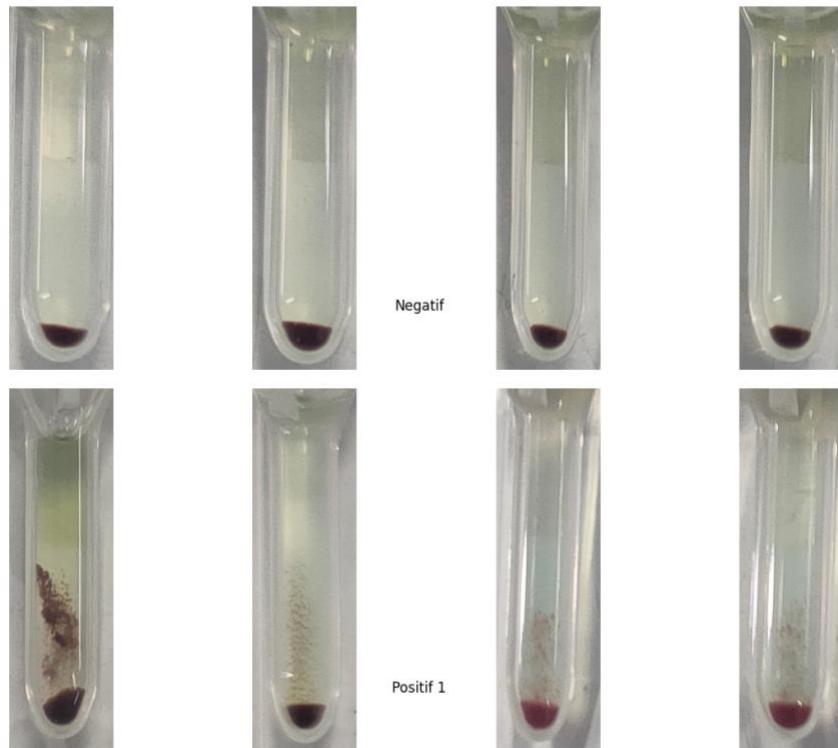
```

1 # Plot images for each class
2 # Loop through each class and its corresponding image names
3 FOR row, (cname, fname_list) in enumerate(class_pics_list):DO
4     # Create a figure for the class
5     fig, axes <- plt.subplots(1, nimages)
6     fig.set_size_inches(12, 5)
7     fig.tight_layout()
8     fig.subplots_adjust(hspace=0.4)
9     fig.suptitle(cname, y=0.25)
10
11    # Loop through each image in the class
12    FOR col, fname in enumerate(fname_list):DO
13        # Create a subplot for each image
14        ax <- axes[col]
15        ax.axis('Off')
16
17        # Load and display the image
18        imgpath <- os.path.join(DATA_DIR, fname)
19        img <- mpimg.imread(imgpath)
20        ax.imshow(img)
21    ENDFOR
22
23    # Show the plot
24    plt.show()
25 ENDFOR
26

```

Kode Semu 3.3 Visualisasi Citra yang Dikelompokkan Berdasarkan Kelasnya

Mengacu pada **Gambar 3.12**, citra dengan kelas Negatif dan kelas Positif 1 tampak sudah sesuai. Perlu diingat bahwa citra yang ditampilkan hanya berjumlah 4 citra sehingga hal ini tidak bisa menandakan bahwa seluruh citra terlabel dengan benar. Perlu dilakukan inspeksi secara menyeluruh dan keterlibatan para ahli terkait untuk memastikan kebenaran label seluruh citra. Dari **Gambar 3.12** juga dapat dilihat beberapa citra terdokumentasi sedikit miring (tabung tampak tidak lurus). Hal ini juga dapat menjadi pertimbangan penambahan fitur rotasi ketika sistem telah dibuat.



Gambar 3.12 Visualisasi *Dataset* Non-Segmentasi

3.5.3 Implementasi *Train-Test Splitting*

Dataset tentunya dibagi menjadi *trainset* dan *testset*. *Trainset* digunakan sebagai bahan belajar oleh model. *Testset* digunakan sebagai bahan evaluasi performa model. **Kode Semu 3.4** menunjukkan implementasi kode untuk proses *train-test splitting*. Proses *train-test splitting* dilakukan dengan bantuan *library scikit-learn*. Berdasarkan **Kode Semu 3.4**, baris 2 mendefinisikan nilai *seed* yang penting agar proses pengacakkan fungsi identik pada seluruh skenario pengujian di *file notebook* terpisah. Nilai *seed* yang dipilih adalah 42. Selanjutnya, pada baris ke-3 hingga ke-6, dilakukan proses pembagian *dataset* menjadi *trainset* dan *testset* menggunakan fungsi “*train_test_split*” dari *library pandas*. Nilai parameter “*test_size*” yang diberikan adalah 0.2 dengan 20% pada *testset* dan 80% pada *trainset*. Hasil *splitting* selanjutnya dimasukkan ke dalam sebuah *dataframe* yang nantinya dimasukkan ke dalam *ImageDataGenerator*. Dapat dilihat pada baris ke-7 dan ke-10, dilakukan proses rekonstruksi *dataframe* untuk masing-masing *trainset* dan *testset*. Tidak lupa juga dilakukan perubahan tipe data menjadi *string* dengan fungsi “*astype*” agar pengolahan *dataframe* dapat dilakukan dengan mudah. Hasil dari proses *train-test splitting* ini ditunjukkan pada **Tabel 3.4** pada **subbab 3.4.4**. Selanjutnya, *trainset* akan masuk ke tahapan augmentasi sebelum masuk ke proses segmentasi citra uji silang serasi.

```

1 from sklearn.model_selection import train_test_split
2 SEED <- 42
3 Xtrain, Xtest, ytrain, ytest <- train_test_split(
4     DF['name'], DF['label'], test_size=.2, shuffle=True,
5     random_state=SEED)
6
7 # train_df and test_df to be fed to the ImageDataGenerator
8 train_df <- pd.DataFrame(
9     {'name': Xtrain, 'label': ytrain}).astype(str)
10
11 test_df <- pd.DataFrame(
12     {'name': Xtest, 'label': ytest}).astype(str)

```

Kode Semu 3.4 Implementasi *Train-Test Splitting*

3.5.4 Implementasi Data Augmentasi

Implementasi data augmentasi dilakukan pada *library* `ImageDataGenerator`. Data teraugmentasi akan *ter-generate* saat proses *training* berlangsung sehingga menghemat *resource* komputasi. **Kode Semu 3.5** menunjukkan implementasi kode data augmentasi. Augmentasi ditunjukkan pada baris ke-2 dan ke-3 dengan parameter “horizontal_flip” yang bernilai *true* yang menandakan dilakukannya proses *flip horizontal* dan parameter “brightness_range” bernilai di antara 0.5 hingga 1.5 yang menunjukkan tingkat *brightness* yang dilakukan.

```

1 train_generator <- ImageDataGenerator(
2     preprocessing_function=pre_func, horizontal_flip=True,
3     brightness_range=[0.5, 1.5]
4 ).flow_from_dataframe(dataframe=train_df,
5     x_col='name', y_col='label', directory=DATA_DIR,
6     batch_size=BATCH_SIZE, class_mode='sparse', target_size=TARGET_SIZE)

```

Kode Semu 3.5 Implementasi Data Augmentasi Proses Klasifikasi

Baris 2 terdapat variabel “*pre_func*” yang di *pass* sebagai argument untuk parameter *preprocessing_function*. Nilai “*pre_func*” ini merupakan fungsi *preprocess input*, sesuai dengan *pre-trained* model yang digunakan. Misalnya, model EfficientNetB7 memiliki fungsi *preprocessing input* agar citra dapat diproses oleh model EfficientNetB7. Nilai *BATCH_SIZE* yang digunakan adalah 32, *TARGET_SIZE* yang digunakan adalah (224, 224) dan variable *DATA_DIR* mengacu pada direktori dimana citra tersimpan.

3.5.5 Implementasi Segmentasi Data

Setelah proses *train-test splitting* dan augmentasi data dilakukan, tahap selanjutnya adalah implementasi segmentasi data. Proses segmentasi dilakukan dengan menggunakan arsitektur U-Net dengan *backbone* model VGG19. Proses ini diawali dengan mendefinisikan *downsampler (encoder)*. **Kode Semu 3.6** menunjukkan implementasi *downsampler* pada proses segmentasi ini. Bagian ini mengambil beberapa *layer* dari model VGG19 yang digunakan sebagai *blocks* pada *downsampler* U-Net. *Down blocks* akan menurunkan dimensi citra. *Downsampler* akan menangkap fitur-fitur esensial pada citra uji silang serasi. Baris ke-3 menunjukkan proses instansiasi model VGG19. Baris ke-6 menunjukkan pemilihan blok konvolusi yang akan digunakan dalam proses segmentasi. Baris ke-12 merupakan bagian dimana *downsampler* dibuat, sesuai dengan blok konvolusi yang sudah dipilih. Pada baris ke-15, *layer trainable* pada model dibekukan (*freeze*) agar *layer* tidak mengalami pembaharuan parameter (*weights*).

```

1 DIM <- (544, 160, 3)
2
3 base_model <- tf.keras.applications.VGG19(
4 include_top=False, input_shape=DIM)
5
6 down_blocks <- ['block2_conv1', 'block3_conv1', 'block4_conv1',
7 'block5_conv1', 'block5_pool']
8
9 outputs <- [base_model.get_layer(name).output FOR name in down_blocks
10 ENDFOR]
11
12 downsample <- tf.keras.Model(inputs=base_model.input,
13 outputs=outputs)
14
15 downsample.trainable = False

```

Kode Semu 3.6 Implementasi *Downsample* pada Segmentasi Data

```

1 FUNCTION decoder_block(num_filters):
2     return tf.keras.layers.Conv2DTranspose(
3         num_filters, (2,2), strides=2, padding="same"
4         )
5 ENDFUNCTION
6
7 up_blocks <- [decoder_block(512), decoder_block(256), _block(128),
8 decoder_block(64),]

```

Kode Semu 3.7 Implementasi *Upsampler* pada Segmentasi Data

Setelah mengimplementasikan *downsample*, langkah selanjutnya adalah menulis kode *upsampler* (*decoder*). **Kode Semu 3.7** menunjukkan implmentasi *upsampler* pada tahap segmentasi data. Proses ini melibatkan fungsi “decoder_block” yang mengembalikan *layer Conv2DTranspose*. *Decoder block* ini nantinya akan terkumpul dalam suatu *list* “*up_blocks*” yang akan mengembalikan dimensi citra ke ukuran semula. Fungsi dari *upsampler* sendiri adalah mengembalikan informasi setelah gambar melalui proses *downsampling* pada *downsample*. Beberapa *layer* pada *downsample* terhubung dengan *upsampler*. Hal ini disebut sebagai *skip connection*. *Skip connection* berfungsi untuk menyalurkan informasi penting pada *downsample* ke *upsampler* saat proses *upsampling* sehingga dihasilkan citra tersegmentasi yang akurat (Ronneberger, Fischer, & Brox, 2015).

Baris ke-2 pada **Kode Semu 3.7** menunjukkan proses pembuatan sebuah blok *upsampler*. Tahapan ini hanya membuat sebuah *layer* konvolusi transpos dengan *stride* bernilai 2 yang bertujuan untuk mengubah dimensi menjadi dua kali lipat lebih besar. Selanjutnya, pada baris ke-7 dilakukan proses instansiasi *upsampler* yang terdiri dari beberapa blok yang dibuat dengan menggunakan fungsi “decoder_block”. Di dalam fungsi tersebut, diberikan suatu parameter “*num_filters*” yang berfungsi sebagai jumlah *filter* yang akan diaplikasikan oleh *layer* konvolusi transpos. Dengan adanya *layer* konvolusi transpos, citra yang sebelumnya mengalami penurunan dimensi, dapat kembali seperti ukuran awal dengan area pada citra yang sudah tersegmentasi.

Selanjutnya dilakukan implementasi arsitektur U-Net yang merupakan gabungan dari *down blocks* dan *up blocks*. **Kode Semu 3.8** menunjukkan implementasi model segmentasi dengan arsitektur U-Net. Proses ini melibatkan pembuatan *skip connection* yang menghubungkan block pada *downsample* dengan *block* pada *upsampler*. Hal ini ditunjukkan pada baris 12-16. Berikutnya, model perlu di-*compile* sesuai dengan konfigurasi yang sudah ditentukan.

```

1 FUNCTION u_net(output_channels:int):
2
3     inputs <- tf.keras.layers.Input(shape=DIM)
4     # skips_connection
5     skips <- downampler(inputs)
6     # bridge
7     x <- skips[-1] # the bridge
8     skips <- skips[:-1][::-1]
9
10    # upsampling and skip connection
11    FOR up, skip in zip(up_blocks, skips):DO
12        x <- up (x)
13        concat <- tf.keras.layers.concatenate()
14        x <- concat([skip, x])
15    ENDFOR
16
17    # final layer
18    final <- tf.keras.layers.Conv2DTranspose(
19        filters=output_channels, kernel_size=3, strides=2,
20        padding='same')
21    final <- final(x)
22    final <- tf.keras.layers.Conv2D(
23        1, (1, 1), activation='sigmoid')(final)
24    return tf.keras.Model(inputs=inputs, outputs=final)
25
26 ENDFUNCTION
27
28 # Instantiate the model
29 NUM_OUTPUT <- 1
30 model <- u_net(NUM_OUTPUT)
31
32 # compile
33 model.compile(optimizer='adam', loss='binary_crossentropy',
34 metrics=['accuracy',tf.keras.metrics.IoU(num_classes=2,
35 target_class_ids=[0])])

```

Kode Semu 3.8 Implementasi Kode Instansiasi Model U-Net

Berikutnya dilakukan proses segmentasi pada seluruh *dataset* uji silang serasi dan disimpan dalam *cloud storage*. Model yang sudah di-*compile* dan siap masuk ke tahap *training*. **Kode Semu 3.9** menunjukkan implementasi kode *training* dan evaluasi pada *testset*. Baris ke-2 pada **Kode Semu 3.9** menunjukkan proses *fitting* model. Masih pada kode semu yang sama, baris ke-7 menunjukkan proses evaluasi model setelah melalui tahap *fitting*. Setelah proses *training* selesai, *learning curve* model dicetak untuk melihat apakah adanya *overfitting* pada model. Kode visualisasi *learning curve* ditunjukkan pada pada **Kode Semu 3.10**. Pada baris ke-3 **Kode Semu 3.10**, dilakukan instansiasi *subplots* yang nantinya akan menjadi wadah untuk visualisasi *learning curve*.

```

1 # Fitting
2 history <- model.fit(train_gen, steps_per_epoch=train_df.shape[0]//32,
3 validation_data=test_gen, validation_steps=test_df.shape[0]//32,
4 epochs=30)
5
6 # Evaluate
7 eva <- model.evaluate(test_gen, steps=test_df.shape[0]//32)

```

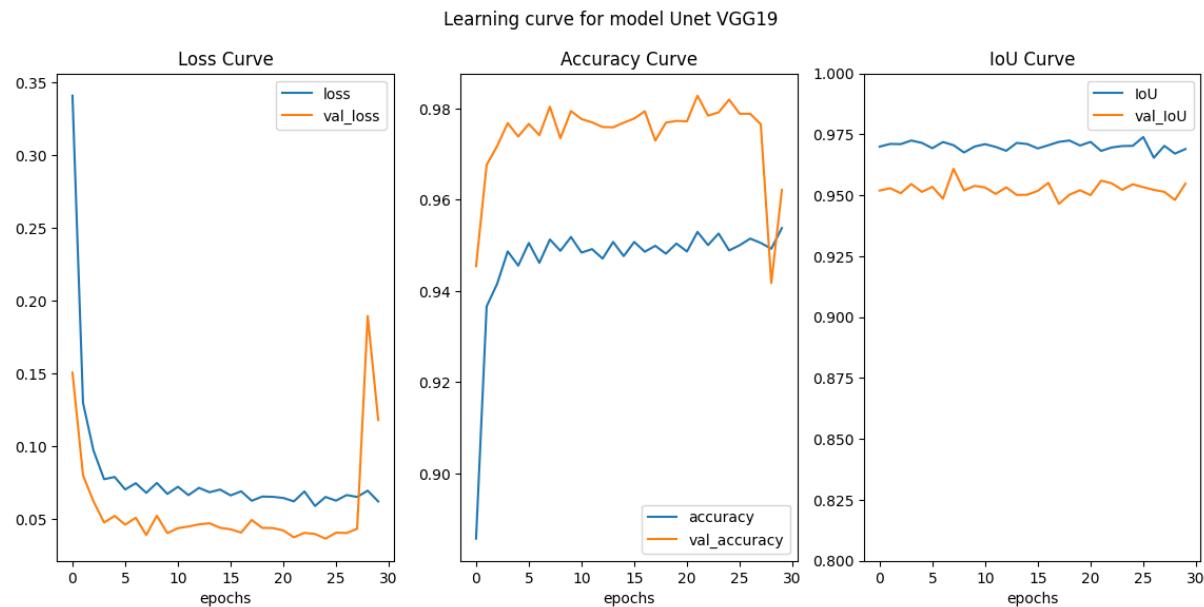
Kode Semu 3.9 Implementasi *Training* dan Evaluasi Proses Segmentasi

```

1 # Plot the learning curve
2 hist <- history.history
3 fig, ax <- plt.subplots(1, 3, figsize=(14, 6))
4 fig.suptitle(f'Learning curve for model U-Net VGG19')
5
6 ax[0].plot(hist['loss'])
7 ax[0].plot(hist['val_loss'])
8 ax[0].legend(['loss', 'val_loss'])
9 ax[0].set_title('Loss Curve')
10 ax[0].set_xlabel('epochs')
11
12 ax[1].plot(hist['accuracy'])
13 ax[1].plot(hist['val_accuracy'])
14 ax[1].legend(['accuracy', 'val_accuracy'])
15 ax[1].set_title('Accuracy Curve')
16 ax[1].set_xlabel('epochs')
17
18 ax[2].plot(hist['iou_1'])
19 ax[2].plot(hist['val_iou_1'])
20 ax[2].legend(['IoU', 'val_IoU'])
21 ax[2].set_ylim(0.8, 1)
22 ax[2].set_title('IoU Curve')
23 ax[2].set_xlabel('epochs')
24
25 plt.show()

```

Kode Semu 3.10 Visualisasi *Learning Curve* Segmentasi



Gambar 3.13 *Learning Curve* Model U-Net pada Proses Segmentasi Data

Hasil *learning curve* dalam proses *training* model segmentasi ditunjukkan pada **Gambar 3.13**. Pada *learning curve*, kurva yang ditampilkan adalah kurva *loss* (*binary crossentropy*), kurva akurasi, dan kurva *iou score*. *Binary crossentropy* digunakan karena mengklasifikasi piksel citra secara biner (0 atau 1). Metrik akurasi menunjukkan seberapa banyak piksel yang terkласifikasi dengan benar. Terakhir, *Iou (Intersection Over Union)* score adalah suatu metrik yang mengukur seberapa banyak piksel hasil prediksi segmentasi yang *overlap* dengan *ground truth* segmentasi. Nilai *iou score* berkisar antara 0 hingga 1. Semakin besar nilai *iou score*, semakin banyak piksel yang ter-*overlap* sehingga semakin bagus hasil prediksi segmentasi tersebut. **Kode Semu 3.11** menunjukkan penggunaan model segmentasi yang sudah dilatih untuk

mensegmentasi seluruh citra dan proses penyimpanan ke *cloud storage*. Baris ke-7 hingga ke-19 pada **Kode Semu 3.11** menunjukkan proses segmentasi seluruh citra sekaligus penyimpanan citra hasil segmentasi ke dalam *cloud storage*. **Gambar 3.14** menunjukkan citra hasil segmentasi yang disimpan dalam *cloud storage*.



Gambar 3.14 Hasil Citra Segmentasi yang Disimpan dalam *Cloud Storage*

(a). Citra dalam *cloud storage* (b). Contoh salah satu citra tersegmentasi

```

1 # check if the model exists
2 pred_model <- tf.keras.models.load_model(
3     '/content/drive/MyDrive/Rai_Model/Segmentation/U-Net_VGG19.h5'
4 )
5
6 DIR_SEGMENTED <- "/content/drive/MyDrive/Rai_Segmented_Images"
7 FOR fname in df['img_path']:DO
8     img <- mpimg.imread(fname)
9     img <- img[:, :, :-1] if img.shape[2] == 4 else img
10    img <- tf.image.resize(img, size=TARGET_SIZE).numpy()
11    pred_img <- pred_model.predict(
12        img.reshape(1, 544, 160, 3)
13        ).reshape(544,160).round()
14    pred_img <- PIL.Image.fromarray(np.uint8(pred_img*255))
15    pred_img.save(os.path.join(DIR_SEGMENTED, fname.split("/")[-1]))
16 ENDFOR

```

Kode Semu 3.11 Proses Segmentasi Seluruh Citra dan Penyimpanan ke *Cloud Storage*

3.5.6 Implementasi Model *Training*

Setelah citra segmentasi dilakukan, langkah selanjutnya adalah melakukan *training*. Proses *training* dilakukan sesuai dengan skenario yang didefinisikan pada **Bab 4**. Tahapan *training* diawali dengan proses *model building*. *Model building* merupakan proses dimana *pre-trained* model didefinisikan. **Kode Semu 3.12** menunjukkan implementasi *model building*. Pada *model building*, *pre-trained* model dihubungkan dengan *fully connected layers* dengan fungsi “*get_model*”. Baris ke-3 hingga ke-5 pada **Kode Semu 3.12** menunjukkan proses *freezing layer*

pada *pre-trained* model. Baris ke-7 hingga ke-19 menunjukkan proses penghubungan *pre-trained* model dengan *fully connected layer*.

```

1  FUNCTION get_model(base_model):
2      # freeze all trainable layers
3      FOR layer in base_model.layers:DO
4          layer.trainable <- False
5      ENDFOR
6
7      hidden <- base_model.output
8      hidden <- tf.keras.layers.Flatten()(hidden)
9      hidden <- tf.keras.layers.Dense(512, activation='relu')(hidden)
10     hidden <- tf.keras.layers.Dropout(0.5)(hidden)
11     output <- tf.keras.layers.Dense(NUM_CLASSES,
12         activation='softmax')(hidden)
13     model <- tf.keras.models.Model(inputs=base_model.inputs,
14         outputs=output)
15     model.compile(optimizer='adam',
16         loss='sparse_categorical_crossentropy', metrics=['accuracy'])
17     return model
18 ENDFUNCTION
19
20
21
22
23 # base models instantiations
24 model1 <- InceptionV3(weights='imagenet', include_top=False,
25     input_shape=DIM)
26
27 model2 <- ResNet50(weights='imagenet', include_top=False,
28     input_shape=DIM)
29
30 model3 <- MobileNetV3Small(weights='imagenet', include_top=False,
31     input_shape=DIM)
32
33 model4 <- EfficientNetB7(weights='imagenet', include_top=False,
34     input_shape=DIM)
35
36 model5 <- InceptionResNetV2(weights='imagenet', include_top=False,
37     input_shape=DIM)
38
39 BASE_MODELS <- {'InceptionV3':model1, 'ResNet50':model2,
40     'MobileNetV3Small':model3, 'EfficientNetB7':model4,
41     'InceptionResNetV2':model5}
42
43 MODELS <- {}
44
45 FOR model_name, base_model in BASE_MODELS.items():DO
46     MODELS[model_name] <- get_model(base_model)
47 ENDFOR
```

Kode Semu 3.12 Implementasi *Model Building*

Setelah proses *model building* selesai, tahap selanjutnya adalah melakukan *fitting* pada seluruh model. Secara umum, implementasi *model fitting* didefinisikan pada **Kode Semu 3.13**. *Fitting* dilakukan dengan *for loop* yang di dalamnya terdapat *instance ImageDataGenerator* yang ditunjukkan pada baris ke-4 hingga baris ke-22 pada **Kode Semu 3.13**. Hal ini dilakukan karena setiap model memiliki fungsi *preprocess input* yang berbeda sehingga *generator* berada di dalam *loop*. Selain itu, pada baris ke-6 dan ke-8, terdapat *train generator* dan *test generator* yang merupakan *instance* dari *ImageDataGenerator*. *Train generator* didapat dari proses augmentasi sebelumnya.

```

1 TRAIN_TIME <- {}
2 TRAINING_HISTORIES <- {}
3
4 FOR (model_name, model), pre_func in zip(MODELS.items(), PREPROCESS_FUNCS) :DO
5     train_generator <- Hasil Augmentasi
6
7     test_generator <- Test Dataframe
8
9     train_time_start <- datetime.now()
10    history <- model.fit(train_generator, epochs= 30,
11                           validation_data=test_generator, callbacks=[callback])
12    train_time_end <- datetime.now()
13
14    TRAIN_TIME[model_name] <- (
15        train_time_end - train_time_start
16        ).total_seconds() / 60
17    TRAINING_HISTORIES[model_name] <- history
18
19 ENDFOR

```

Kode Semu 3.13 Implementasi *Model Fitting*

3.5.7 Implementasi *Testing* dan Evaluasi Performa

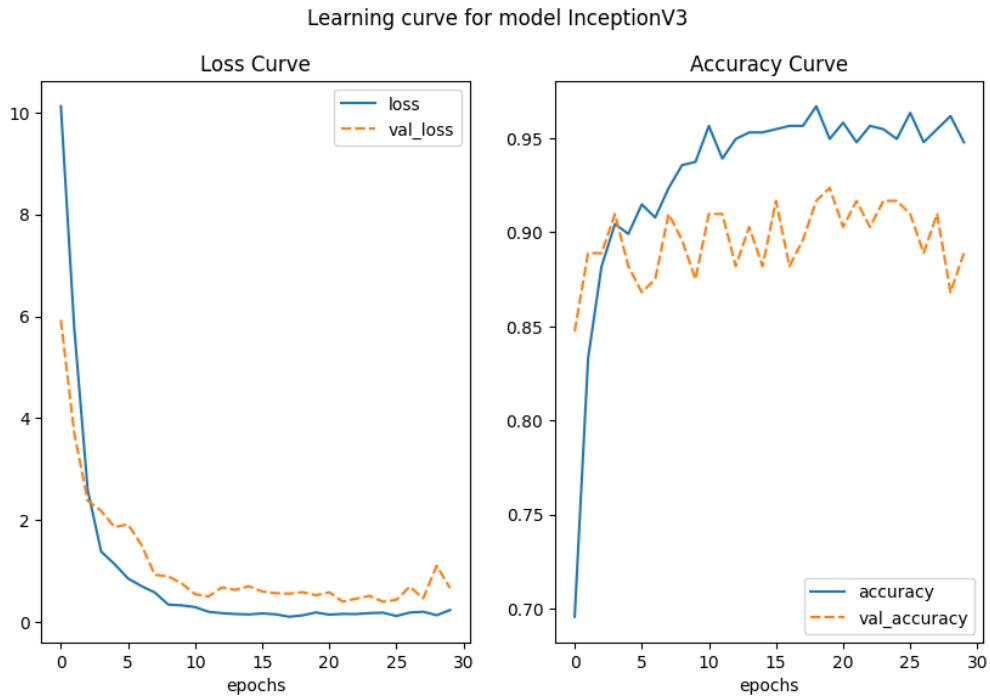
Setelah seluruh model melalui proses *model training*, selanjutnya performa masing-masing model dievaluasi. Proses evaluasi ini melibatkan visualisasi *learning curve* dan perhitungan *metrics* evaluasi yang diantaranya *f1-score*, *precision*, *recall*, dan *accuracy*. Perhitungan *metrics* evaluasi ini dibantu dengan *library scikit-learn* dan hasilnya disimpan dalam bentuk *dataframe*. Selanjutnya, dilakukan juga perhitungan *confusion matrix* dan visualisasinya dengan menggunakan *library seaborn*. Terakhir, proses inspeksi citra dilakukan dengan cara memvisualisasikan citra yang salah dengan bantuan *library matplotlib*. Visualisasi ini secara umum dibagi menjadi dua jenis, yaitu visualisasi pada *dataset* non-segmentasi dan *dataset* tersegmentasi. **Kode Semu 3.14** menunjukkan implementasi visualisasi *learning curve* setiap model pada proses klasifikasi. Baris ke-6 hingga ke-16 menunjukkan tahap pembuatan *subplots* hingga visualisasi. **Gambar 3.15**, **Gambar 3.16**, **Gambar 3.17**, **Gambar 3.18**, dan **Gambar 3.19** menunjukkan grafik *learning curve* pada model InceptionV3, ResNet50, MobileNetV3Small, EfficientNetB7 dan InceptionResNetV2 secara berurutan.

```

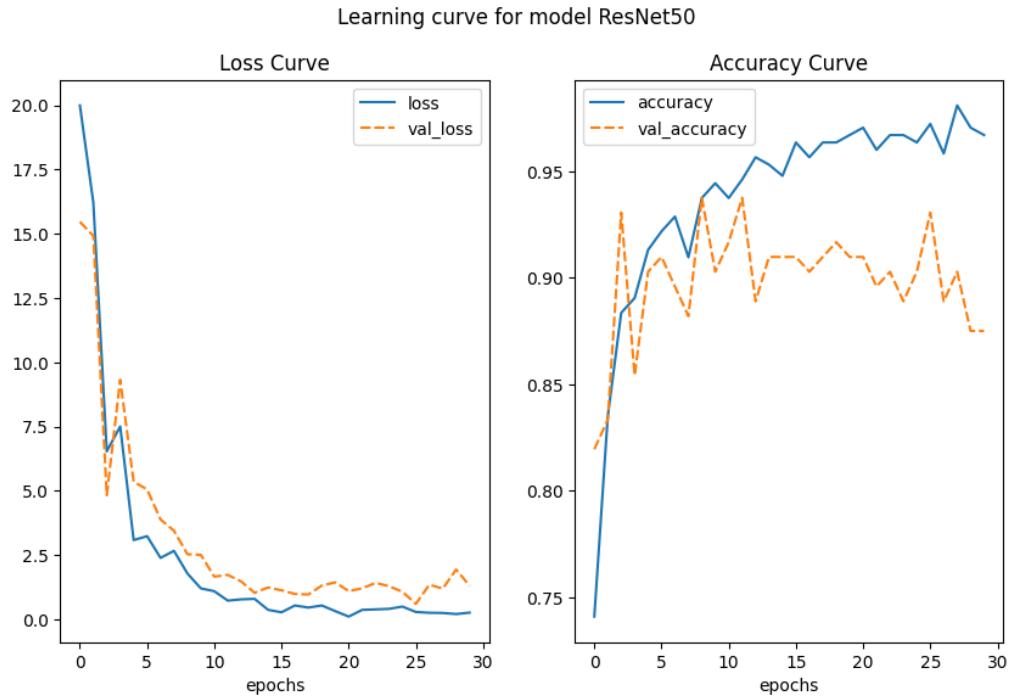
1 FOR mname, hist_obj in TRAINING_HISTORIES.items() :DO
2     # Get every single model's history object
3     hist <- pd.DataFrame(hist_obj.history)
4
5     # Figure initialization
6     fig, ax <- plt.subplots(1, 2, figsize=(10, 6))
7     fig.suptitle(f'Learning curve for model {mname}')
8     sns.lineplot(ax=ax[0], data=hist[['loss', 'val_loss']])
9     ax[0].set_title('Loss Curve')
10    ax[0].set_xlabel('epochs')
11    sns.lineplot(ax=ax[1], data=hist[['accuracy', 'val_accuracy']])
12    ax[1].set_title('Accuracy Curve')
13    ax[1].set_xlabel('epochs')
14
15    # Show plot
16    plt.show()
17
18 ENDFOR

```

Kode Semu 3.14 Implementasi Visualisasi *Learning Curve* pada Proses Klasifikasi



Gambar 3.15 Learning Curve InceptionV3 pada Proses Klasifikasi

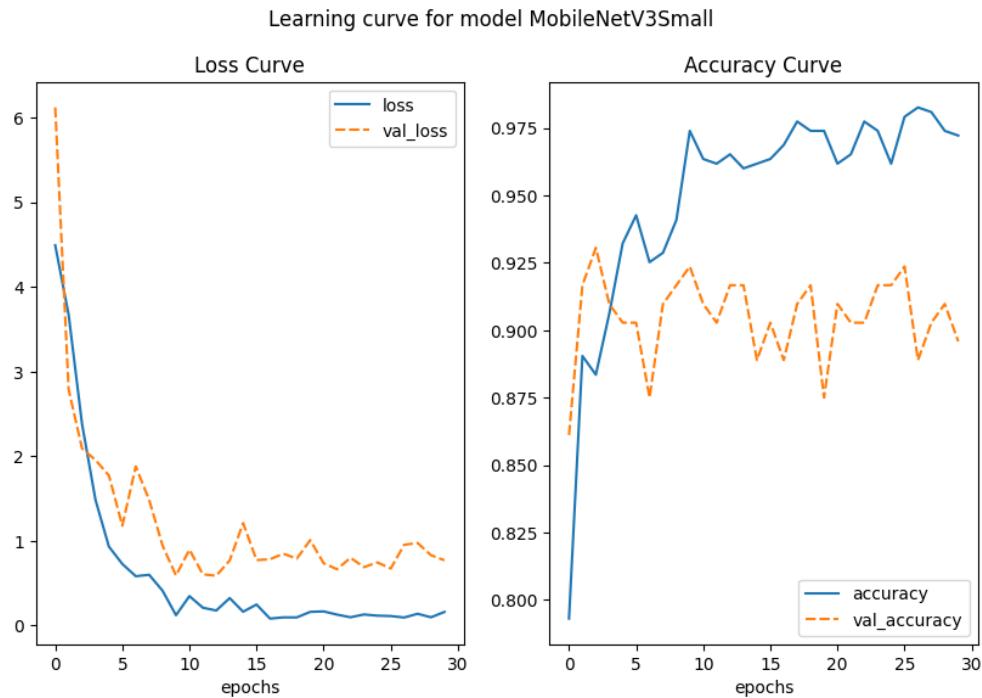


Gambar 3.16 Learning Curve ResNet50 pada Proses Klasifikasi

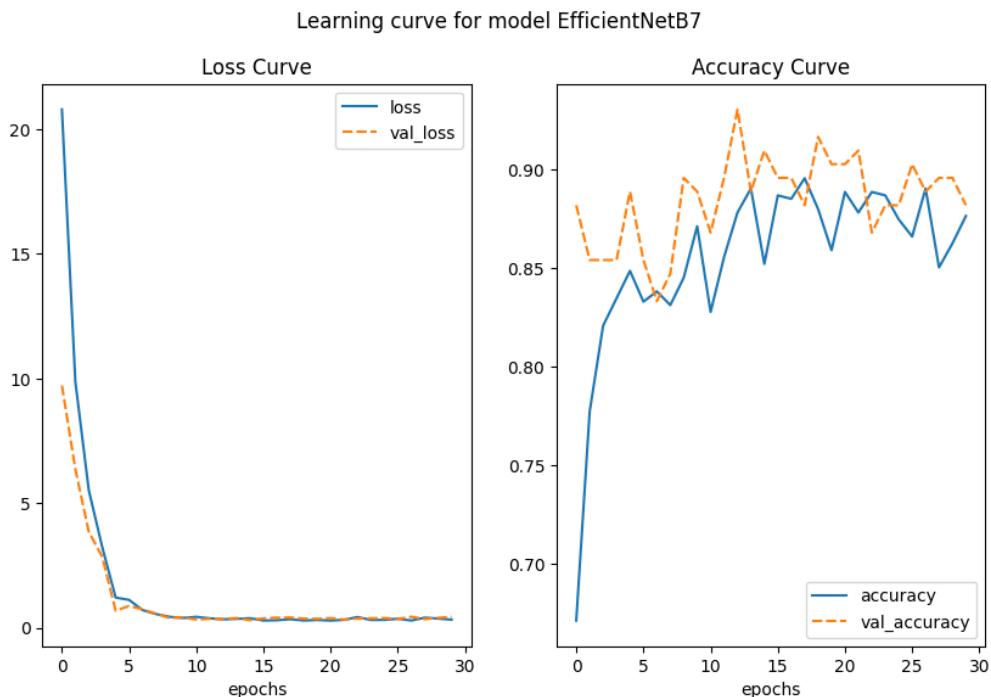
Setelah melihat *learning curve*, tahap selanjutnya adalah menghitung *metrics* evaluasi. **Kode Semu 3.15** menunjukkan implementasi perhitungan *metrics* implementasi pada tiap model. Proses ini dilakukan dengan cara menyimpan setiap *metric* model pada struktur data *dictionary* setiap *metric* yang ditunjukkan pada baris ke-6 hingga baris ke-23 pada **Kode Semu 3.15**.

Implementasi terakhir pada bagian *testing* dan evaluasi performa adalah menampilkan *confusion matrix* dan kesalahan prediksi pada citra. **Kode Semu 3.16** menunjukkan

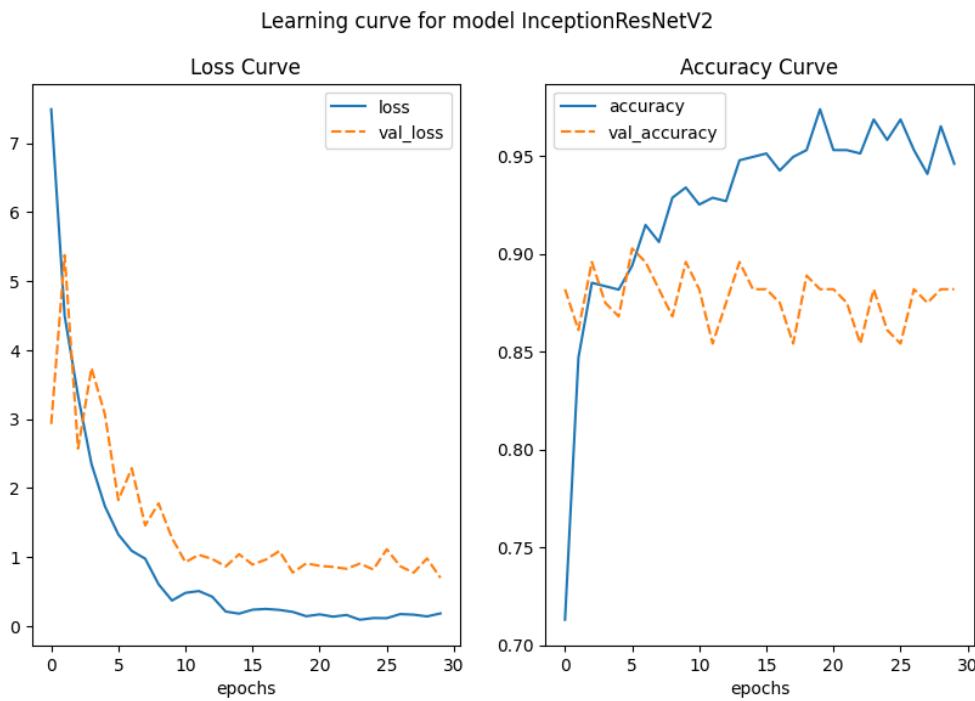
implementasi visualisasi *confusion matrix* dalam bentuk fungsi. **Kode Semu 3.17** menunjukkan implementasi pencarian kesalahan prediksi pada model. Hal ini dilakukan dengan menyimpan nama file citra yang salah diprediksi oleh suatu model pada *dictionary*, sesuai pada baris ke-3 hingga baris ke-10 pada **Kode Semu 3.17**.



Gambar 3.17 Learning Curve MobileNetV3Small pada Proses Klasifikasi



Gambar 3.18 Learning Curve EfficientNetB7 pada Proses Klasifikasi



Gambar 3.19 Learning Curve InceptionResNetV2 pada Proses Klasifikasi

```

1 FOR (model_name, model), pre_func in zip(MODELS.items(),
2 PREPROCESS_FUNCS):DO
3     prediction_labels <- np.argmax(
4         model.predict(validation_generator), axis=1)
5
6     ACCURACY_SCORES[model_name] <- accuracy_score(true_labels,
7         prediction_labels)
8     PREDICTION_TIME[model_name] <- (prediction_time_end -
9         prediction_time_start).total_seconds()
10    PRECISION_SCORES[model_name] <- precision_score(true_labels,
11        prediction_labels, average='macro')
12    RECALL_SCORES[model_name] <- recall_score(true_labels,
13        prediction_labels, ='macro')
14    F1_SCORES[model_name] <- f1_score(true_labels, prediction_labels,
15        average='macro')
16    MISTAKES[model_name] <- find_mistakes(true_labels,
17        prediction_labels)
18    CONFUSION_MATRIX[model_name] <-
19        tf.math.confusion_matrix(true_labels, prediction_labels,
20            num_classes=5)
21
22 ENDFOR

```

Kode Semu 3.15 Proses Perhitungan Metric Evaluasi pada Proses Klasifikasi

Kode Semu 3.18 menunjukkan implementasi fungsi utilitas visualisasi citra yang salah diprediksi oleh model. Fungsi ini akan menggambar citra sesuai dengan *subplots* yang diberikan. Hal ini ditunjukkan pada baris ke-12 hingga baris ke-20 yang akan menulis citra pada kanvas sesuai dengan *subplot* yang diberikan. **Kode Semu 3.19** dan **Kode Semu 3.20**, secara berurutan, menunjukkan implementasi *layout* visualisasi untuk *dataset* non-segmentasi dan tersegmentasi. Visualisasi kedua jenis *dataset* ini berbeda. Pada visualisasi citra non-segmentasi, hanya citra *original* saja yang ditampilkan.

Baris ke-10 hingga baris ke-18 pada **Kode Semu 3.19** menunjukkan proses penggambaran citra kesalahan prediksi *dataset* non-segmentasi yang dilakukan dengan cara mencetak gambar maksimal sebanyak 5 buah pada setiap *row*. Hal ini dilakukan dengan tujuan agar visualisasi yang dihasilkan terlihat rapi dan jelas.

```

1 FUNCTION draw_confusion_matrix(cm, model_name):
2     plt.figure(figsize=(8, 6))
3     sns.heatmap(cm, annot=True, cmap='Greens', fmt='d',
4                 xticklabels=cm.columns, yticklabels=cm.columns)
5     plt.xlabel('Predicted Label')
6     plt.ylabel('True Label')
7     plt.title(model_name)
8     plt.show()
9 ENDFUNCTION
```

Kode Semu 3.16 Implementasi Visualisasi *Confusion Matrix*

```

1 # Find mistakes for each true labels
2 FUNCTION find_mistakes(true_labels, predictions):
3     mistakes <- {0:[], 1:[], 2:[], 3:[], 4:[]}
4     # Dict structure
5     # { true_val: [file_no, predicted] }
6     FOR i, (true, pred) in enumerate(zip(true_labels, predictions)) :DO
7         IF(true != pred) :THEN
8             mistakes[true].append([i, pred])
9         ENDIF
10    ENDFOR
11    return mistakes
12 ENDFUNCTION
```

Kode Semu 3.17 Implementasi Pencarian Kesalahan Prediksi

```

1 FUNCTION draw_along_axes(axes, arr, val_dir, fname_list, cols):
2     i <- 0
3     IF(cols > 1) :THEN
4         FOR ax in axes:DO
5             ax.axis('Off')
6         ENDFOR
7     ELSE :THEN
8         axes.axis('Off')
9     ENDIF
10
11     # draw
12     FOR findex, pred_label in arr:DO
13         fname_path <- os.path.join(val_dir, fname_list[findex])
14         # subplot
15         ax <- axes[i] IF cols > 1 ELSE axes
16         i += 1
17         ax.title.set_text(f'{fname_list[findex]}\nDetected as
18         {pred_label}')
19         img <- mpimg.imread(fname_path)
20         ax.imshow(img)
21     ENDFOR
22 ENDFUNCTION
```

Kode Semu 3.18 Implementasi Fungsi Utilitas Visualisasi Citra Kesalahan Prediksi

Sementara itu, pada citra tersegmentasi, terdapat 3 jenis citra yang harus ditampilkan, yaitu citra *original*, citra *ground truth*, dan citra hasil segmentasi. Hal ini dapat dilihat pada **Kode Semu 3.20** di baris ke-4, terdapat inisialisasi variabel “cols” bernilai 3 yang masing-masing menjadi wadah pada ketiga citra yang sudah disebutkan. **Kode Semu 3.20** melibatkan banyak

proses pembacaan *file* karena melibatkan citra-citra yang berada pada direktori yang berbeda-beda.

```

1  FUNCTION draw_mistakes(val_dir, true_class, arr, fname_list):
2      nimages <- len(arr)
3      rows <- (nimages // 5) + (1 if nimages % 5 else 0)
4      cols <- nimages if nimages < 6 else 5
5
6      # fig class setting
7      fig, axes <- plt.subplots(rows, cols)
8      fig.set_size_inches(12, 5*rows)
9      fig.suptitle(f"True Value {true_class}")
10     IF(nimages > 5):THEN
11         i <- 0
12         FOR ax in axes:DO
13             draw_along_axes(ax, arr[i:i+5], val_dir, fname_list, cols)
14             i += 5
15         ENDFOR
16     ELSE:THEN
17         draw_along_axes(axes, arr, val_dir, fname_list, cols)
18     ENDIF
19     plt.show()
20 ENDFUNCTION
```

Kode Semu 3.19 Implementasi *Layout* Visualisasi Kesalahan Citra Non-Segmentasi

```

1  FUNCTION draw_mistakes_segmented(true_class, arr, fname_list):
2      nimages <- len(arr)
3      rows <- nimages
4      cols <- 3
5      # fig class setting
6      FOR findex, pred_label in arr:DO
7          path_segmented <- os.path.join(DATA_DIR, fname_list[findex])
8          path_true_segmented <- os.path.join(TRUE_SEGMENTED_DIR,
9              fname_list[findex])
10         path_true_image <- os.path.join(TRUE_DIR, fname_list[findex])
11         fig, (ax1, ax2, ax3) <- plt.subplots(1,3, figsize=(6,8))
12         fig.suptitle(f'\nFile name: {fname_list[findex]} with class
13             {true_class}\nBut predicted as {pred_label}')
14         img <- mpimg.imread(path_true_image)
15         img1 <- mpimg.imread(path_true_segmented)
16         img2 <- mpimg.imread(path_segmented)
17         ax1.imshow(img)
18         ax2.imshow(img1)
19         ax3.imshow(img2)
20         ax1.set_title('True \nNon Segmented')
21         ax2.set_title('True \nSegmented')
22         ax3.set_title('Predicted \nSegmented')
23         ax1.axis('Off')
24         ax2.axis('Off')
25         ax3.axis('Off')
26         plt.show()
27     ENDFOR
28 ENDFUNCTION
```

Kode Semu 3.20 Implementasi *Layout* Visualisasi Kesalahan Citra Tersegmentasi

[Halaman ini sengaja dikosongkan]

BAB 4 HASIL DAN PEMBAHASAN

Bab ini membahas hasil uji coba yang sudah dilakukan untuk mengevaluasi performa sistem berdasarkan lingkungan uji coba yang sudah ditentukan.

4.1 Skenario Uji Coba dan Analisis

Skenario uji coba dilakukan pada seluruh *pre-trained* model yang digunakan. **Tabel 4.1** menunjukkan rangkuman skenario uji coba yang dilakukan dan **Tabel 4.2** yang menggambarkan pemetaan variable konstan pada masing-masing skenario uji coba.

Tabel 4.1 Rangkuman Skenario Uji Coba yang Dilakukan

No	Dataset	Optimizer	Learning Rate	Hidden Units	Dropout Rate	Variabel Konstan (Lihat Tabel 4.2)
1	Non Segmentasi	Adam	0.01	512	0.5	1
2		Hasil dari Keras Tuner	Hasil dari Keras Tuner	Hasil dari Keras Tuner	Hasil dari Keras Tuner	1
3		Adam, Adagrad, RMSProp	0.01	512	0.5	2
4		Terbaik dari skenario 3	0.1, 0.01, 0.001	512	0.5	2
5		Terbaik dari skenario 3	Terbaik dari skenario 4	128, 256, 512	0.5	2
6		Terbaik dari skenario 3	Terbaik dari skenario 4	Terbaik dari skenario 5	0.2, 0.3, 0.5	2
7		Adam	0.01	512	0.5	3
8	Tersegmentasi	Adam	0.01	512	0.5	1
9		Hasil dari Keras Tuner	Hasil dari Keras Tuner	Hasil dari Keras Tuner	Hasil dari Keras Tuner	1
10		Adam, Adagrad, RMSProp	0.01	512	0.5	2
11		Terbaik dari skenario 10	0.1, 0.01, 0.001	512	0.5	2
12		Terbaik dari skenario 10	Terbaik dari skenario 11	128, 256, 512	0.5	2
13		Terbaik dari skenario 10	Terbaik dari skenario 11	Terbaik dari skenario 12	0.2, 0.3, 0.5	2
14		Adam	0.01	512	0.5	4

Tabel 4.2 Variabel Konstan pada masing-masing Skenario

No	Deskripsi
1	<ul style="list-style-type: none"> Variasi Model <i>Pre-trained</i> yang digunakan: <ol style="list-style-type: none"> InceptionV3 ResNet50 MobileNetV3Small EfficientNetB7 InceptionResNetV2 Epochs: 30 Loss Function: Sparse Categorical Crossentropy Batch Size: 32 Image Size: 224 x 224 piksel

No	Deskripsi
2	<ul style="list-style-type: none"> Variasi Model <i>Pre-trained</i> yang digunakan adalah MobileNetV3Small <i>Epochs</i>: 30 <i>Loss Function</i>: Sparse Categorical Crossentropy <i>Batch Size</i>: 32 <i>Image Size</i>: 224 x 224 piksel
3	<ul style="list-style-type: none"> Variasi Model <i>Pre-trained</i> yang digunakan: <ol style="list-style-type: none"> MobileNetV3Large MobileNetV3Small <i>Epochs</i>: 30 <i>Loss Function</i>: Sparse Categorical Crossentropy <i>Batch Size</i>: 32 <i>Image Size</i>: 224 x 224 piksel
4	<ul style="list-style-type: none"> Variasi Model <i>Pre-trained</i> yang digunakan adalah VGG19 <i>Epochs</i>: 30 <i>Loss Function</i>: Sparse Categorical Crossentropy <i>Batch Size</i>: 32 <i>Image Size</i>: 224 x 224 piksel

4.1.1 Uji Coba Pada *Hyperparameter Default*

Uji coba pertama menggunakan *hyperparameter default* pada kedua data non-segmentasi dan tersegmentasi. Skenario yang diuji pada tahap ini dideskripsikan pada **Tabel 4.1** nomor 1 (data non-segmentasi) dan nomor 8 (data tersegmentasi). Tujuan dilakukannya pengujian ini adalah untuk melihat performa *pre-trained* model pada kedua dataset yang berbeda dengan konfigurasi *hyperparameter default*. *Hyperparameter* tentunya berdampak pada hasil *training* suatu model untuk *dataset* yang berbeda-beda. Misalnya, untuk *dataset* tersegmentasi, ada kemungkinan penggunaan *hidden units* yang berlebihan menyebabkan *overfit* karena model tidak memerlukan unit yang banyak untuk mempelajari data yang sudah disederhanakan. Penggunaan *hyperparameter default* di awal memberikan petunjuk untuk langkah selanjutnya yang perlu dilakukan dalam meningkatkan performa model. Dengan uji coba ini, penulis dapat mendapatkan gambaran pengaruh data tersegmentasi terhadap performa model.

A. Dataset Non-Segmentasi

Pengujian pertama menggunakan *dataset* non-segmentasi yang terdiri dari 719 buah citra yang diambil dari bank darah RSUD. Dr Soetomo. *Dataset* ini dibagi menjadi dua jenis, yaitu *train set* dan *test set*. Model di-*train* dengan *train set* diuji dengan *test set*. Pengujian dengan *test set* penting dilakukan karena data ini adalah data yang belum dilihat oleh model (*unseen data*). Hal ini bertujuan untuk menghindari adanya data tumpang tindih atau yang telah dipelajari sebelumnya oleh model. Distribusi data yang digunakan dalam uji coba ini dapat dilihat pada **Tabel 3.4**.

Dari hasil uji coba yang dilakukan, *pre-trained* model EfficientNetB7 memberikan performa terbaik dengan nilai *f1-Score* 0.9062 dan akurasi sebesar 0.9097. Angka ini sudah baik. Berdasarkan **Tabel 4.3**, performa model InceptionV3 dan InceptionResNetV2 masih belum optimal, yaitu 0.7338 untuk *f1-score* InceptionV3 dan 0.7666 untuk *f1-score* InceptionResNetV2. Meskipun akurasinya berada di atas 0.8, *f1-score* model yang berada di angka 0.7 ini menandakan bahwa ada beberapa kelas yang tidak dapat diprediksi dengan baik oleh model. Penelitian ini menggunakan data *imbalance*, dimana distribusi kelas pada *dataset* tidak merata. Oleh karena itu, metrik akurasi menjadi prioritas kedua dibandingkan dengan *f1-score*.

Nilai *f1-score* rata-rata model adalah 0.8161. Angka ini berarti beberapa model memiliki nilai *f1-score* di atas 0.8. Angka ini sudah cukup baik, tetapi masih bisa ditingkatkan.

Peningkatan dapat dilakukan dengan berbagai cara, seperti *hyperparameter tuning*, inspeksi citra, dsb.

Tabel 4.3 Hasil Uji Dataset Non-Segmentasi pada *Test Set Hyperparameter Default*

Model	Precision	Recall	F1-Score	Accuracy
InceptionV3	0.8185	0.7353	0.7338	0.8333
ResNet50	0.8800	0.8202	0.8323	0.8681
MobileNetV3Small	0.9037	0.8278	0.8418	0.8889
EfficientNetB7	0.9170	0.9020	0.9062	0.9097
InceptionResNetV2	0.8165	0.7838	0.7666	0.8403
Rata-rata	0.8671	0.8138	0.8161	0.8681

B. Dataset Tersegmentasi

Pengujian selanjutnya dilakukan dengan data yang sudah tersegmentasi. Sama seperti data non-segmentasi, pengujian dengan data tersegmentasi dilakukan dengan *test set* pada model yang di-*train* dengan *train set*. Jika mengacu pada **Tabel 4.1**, skenario ini adalah skenario nomor 8. **Tabel 4.4** menunjukkan hasil uji coba yang dilakukan dengan data tersegmentasi.

Tabel 4.4 Hasil Uji Dataset Tersegmentasi pada *Test Set Hyperparameter Default*

Model	Precision	Recall	F1-Score	Accuracy
InceptionV3	0.8703	0.7933	0.8002	0.8889
ResNet50	0.8829	0.8704	0.8736	0.8750
MobileNetV3Small	0.9056	0.8809	0.8878	0.8958
EfficientNetB7	0.8813	0.8102	0.8405	0.8819
InceptionResNetV2	0.8439	0.8311	0.8341	0.8819
Rata-rata	0.8768	0.8372	0.8472	0.8847

Berdasarkan hasil uji yang sudah dilakukan, *pre-trained* model MobileNetV3Small memiliki performa terbaik dari keseluruhan model dengan nilai *f1-score* di angka 0.8878 dan akurasi di angka 0.8958. Dari sisi, *f1-Score* dan akurasi, model ResNet50 menyusul MobileNetV3Small di angka 0.8736 untuk *f1-score* dan 0.8750 untuk akurasi. Nilai *f1-score* MobileNetV3Small di angka 0.8878 sudah baik. Ini berarti model memiliki nilai *precision* dan *recall* yang baik. *Precision* sendiri menjelaskan seberapa akurat kelas hasil prediksi model.

Sementara itu, *recall* mendeskripsikan seberapa banyak citra dengan kelasnya yang diprediksi dengan benar, sesuai dengan kelas sebenarnya. Penggunaan *dataset* tersegmentasi dalam penilitian ini berhasil mendapatkan nilai 0.8847 untuk rata-rata akurasi, 0.8472 untuk rata-rata *f1-Score*, 0.8768 untuk rata-rata *precision*, dan 0.8372 untuk rata-rata *recall*. Angka dari 4 *metric* tersebut menunjukkan bahwa performa *pre-trained* model yang digunakan berada di angka yang cukup baik.

4.1.2 Uji Coba dengan *Hyperparameter Tuning*

Pengujian kedua ini menggunakan konfigurasi *hyperparameter* yang sudah mengalami proses *tuning*. *Hyperparameter* di-*tuning* dengan menggunakan modul KerasTuner di setiap model. Proses detail *hyperparameter tuning* dijelaskan pada **subbab 3.4.6.2**. Tujuan dilakukannya proses *hyperparameter tuning* ini adalah untuk mendapatkan konfigurasi *hyperparameter* yang terbaik pada setiap model. Adapun *hyperparameter* yang di-*tuning*, yaitu

optimizer, *learning rate*, *hidden units*, dan *dropout rate*. Uji coba ini menerapkan skenario nomor 2 dan nomor 9 pada **Tabel 4.1**.

A. Dataset Non-Segmentasi

Pengujian pertama dilakukan pada *dataset* non-segmentasi. **Tabel 4.5** menunjukkan hasil *hyperparameter tuning* pada setiap model. Dari sisi *optimizer*, RMSProp menunjukkan performa yang kurang baik dibandingkan dengan Adam dan Adagrad. Dari sisi *learning rate* dan *hidden units*, seluruh model memiliki nilai seragam, yaitu 0.001 dan 512. Untuk *dropout rate* sendiri, nilainya bervariasi di angka 0.3 dan 0.2. Nilai ini lebih rendah dari nilai skenario *default* yang menggunakan *dropout rate* di angka 0.5. Ini menunjukkan bahwa pada proses *hyperparameter tuning*, model menunjukkan performa yang baik dengan 30% atau 20% neuron yang dinonaktifkan secara acak.

Tabel 4.5 Hasil *Hyperparameter Tuning* Setiap Model pada *Dataset* Non-Segmentasi

Model	Optimizer	Learning Rate	Hidden Units	Dropout Rate
InceptionV3	Adam	0.001	512	0.3
ResNet50	Adagrad	0.001	512	0.3
MobileNetV3Small	Adam	0.001	512	0.2
EfficientNetB7	Adam	0.001	512	0.2
InceptionResNetV2	Adagrad	0.001	512	0.2

Tabel 4.6 menunjukkan hasil *training* setiap model dengan *hyperparameter* yang sudah di-tuning. Berdasarkan **Tabel 4.6**, model yang memiliki performa terbaik adalah MobileNetV3Small dengan nilai *f1-score* 0.8692 dan nilai akurasi 0.9236. Model ini disusul oleh model InceptionResNetV2 yang memiliki *f1-score* di angka 0.8444 dan akurasi di angka 0.9028. Model InceptionV3, ResNet50, dan EfficientNetB7 memiliki performa yang kurang optimal. Hal ini dapat dilihat dari nilai *f1-score* yang masih berada di bawah 0.8. Dari sisi akurasi, beberapa model ini memiliki angka yang cukup baik, seperti pada model InceptionV3 yang akurasinya di angka 0.8472. Hal ini berarti bahwa suatu kelas pada *dataset* gagal diprediksi dengan baik oleh model InceptionV3. Rata-rata yang didapat dari setiap *metrics* evaluasi pada skenario ini adalah 0.7803 untuk *recall*, 0.8348 untuk *precision*, 0.7895 untuk *f1-score*, dan 0.8639 untuk akurasi. Secara umum, performa *pre-trained* model masih belum baik dan masih bisa ditingkatkan.

Tabel 4.6 Hasil Uji pada *Dataset* Non-Segmentasi pada Model *Hyperparameter Tuning*

Model	Precision	Recall	F1-Score	Accuracy
InceptionV3	0.7691	0.7230	0.7297	0.8472
ResNet50	0.8208	0.8579	0.7081	0.7917
MobileNetV3Small	0.9192	0.6852	0.8692	0.9236
EfficientNetB7	0.8136	0.7877	0.7963	0.8542
InceptionResNetV2	0.8515	0.8475	0.8444	0.9028
Rata-rata	0.8348	0.7803	0.7895	0.8639

B. Dataset Tersegmentasi

Pengujian selanjutnya dilakukan dengan menggunakan data tersegmentasi. **Tabel 4.7** menunjukkan hasil *hyperparamter* yang diperoleh saat melakukan *tuning* dengan KerasTuner.

Hal yang menarik dari hasil *hyperparameter tuning* adalah nilai *hidden units* yang bervariasi pada tiap model. Dari sisi optimizer, RMSProp masih tidak unggul pada *dataset* tersegmentasi.

Tabel 4.7 Hasil *Hyperparameter Tuning* pada Setiap Model dengan *Dataset* Tersegmentasi

Model	Optimizer	Learning Rate	Hidden Units	Dropout Rate
InceptionV3	Adagrad	0.001	128	0.3
ResNet50	Adam	0.001	512	0.3
MobileNetV3Small	Adagrad	0.01	128	0.3
EfficientNetB7	Adam	0.001	512	0.2
InceptionResNetV2	Adam	0.001	256	0.3

Tabel 4.8 Hasil Uji pada *Dataset* Tersegmentasi pada Model *Hyperparameter Tuning*

Model	Precision	Recall	F1-Score	Accuracy
InceptionV3	0.9052	0.8654	0.8816	0.9167
ResNet50	0.9260	0.9052	0.9120	0.9306
MobileNetV3Small	0.9240	0.8624	0.8878	0.8889
EfficientNetB7	0.9461	0.9300	0.9374	0.9236
InceptionResNetV2	0.8882	0.7868	0.8093	0.8750
Rata-rata	0.9179	0.8700	0.8856	0.9069

Mengacu pada **Tabel 4.8**, model terbaik dari skenario ini adalah EfficientNetB7 dengan nilai *f1-score* sebesar 0.9374 dan nilai akurasi sebesar 0.9236. Di posisi selanjutnya, model ResNet50 memberikan performa yang tidak terlalu berbeda, yaitu di angka 0.9120 untuk *f1-score* dan 0.9306 untuk akurasi. Kedua model ini mampu menunjukkan performa *f1-score* dan akurasi diatas 0.9. Ini berarti model-model ini mampu memprediksi kelas dengan akurat meskipun pada data yang *imbalance*. Hal yang menarik adalah model InceptionResNetV2 yang mengalami penurunan performa di skenario dengan *dataset* tersegmentasi ini.

4.1.3 Uji Coba Pengaruh *Hyperparameter* dengan Model Kontrol MobileNetV3Small

Pengujian ketiga bertujuan untuk melihat pengaruh *optimizer*, *learning-rate*, *dropout-rate* dan *hidden-units* (*hyperparameter*). Tujuan dari pengujian ini adalah untuk melihat bagaimana pengaruh *hyperparameter* pada proses klasifikasi citra uji silang serasi. Pengujian ini dilakukan dengan model kontrol MobileNetV3Small. Alasan dipilihnya model ini adalah karena ukurannya yang ringan sehingga proses evaluasi bisa berjalan lebih cepat. *Dataset* yang digunakan masih sama, yaitu *dataset* tersegmentasi dan *dataset* non-segmentasi. Untuk *dataset* non-segmentasi, skenario yang dilakukan adalah skenario nomor 3, 4, 5 dan 6. Sementara itu, skenario pada *dataset* tersegmentasi adalah skenario nomor 10, 11, 12, dan 13. Skenario ini mengacu pada **Tabel 4.1**. Pengujian ini dilakukan dari *optimizer*, *hidden units*, *dropout rate*, dan *learning rate* secara berurutan. Detail variabel independent dan variabel kontrol dapat dilihat pada **Tabel 4.1**.

A. Dataset Non-Segmentasi

Tabel 4.9 menunjukkan hasil uji *optimizer* pada *dataset* non-segmentasi. Sesuai tabel, *optimizer* yang memberikan performa terbaik adalah Adagrad dengan nilai *f1-score* di angka 0.9212 dan angka akurasi di 0.9028. Adagard memberikan performa yang cukup berbeda dengan *optimizer* lainnya. *Optimizer* Adagrad digunakan sebagai *optimizer* kontrol pada uji *hidden units* selanjutnya.

Tabel 4.9 Hasil Uji *Optimizer* pada Model MobileNetV3Small pada *Dataset Non-Segmentasi*

MobileNetV3Small				
Optimizer	Precision	Recall	F1-Score	Accuracy
Adam	0.8611	0.8612	0.8583	0.8542
Adagrad	0.9401	0.9107	0.9212	0.9028
RMSProp	0.9021	0.8235	0.8495	0.8889

Tabel 4.10 Hasil Uji *Hidden Units* pada Model MobileNetV3Small pada *Dataset Non-Segmentasi*

MobileNetV3Small				
Hidden Units	Precision	Recall	F1-Score	Accuracy
128	0.8916	0.8534	0.8698	0.8958
256	0.9283	0.9010	0.9127	0.8958
512	0.9426	0.9231	0.9317	0.9167

Tabel 4.10 menunjukkan hasil uji *hidden units*. Performa terbaik diberikan oleh *hidden units* di angka 512 dengan nilai *f1-score* 0.9317 dan akurasi 0.9167. Jika mengacu pada **Tabel 4.10**, performa model meningkat seiring bertambahnya nilai *hidden units*. Selanjutnya *hidden units* 512 digunakan sebagai *hidden units* kontrol di uji coba *dropout rate*.

Tabel 4.11 menunjukkan hasil uji *dropout rate*. Nilai 0.5 memberikan performa terbaik dengan *f1-score* di angka 0.9374 dan akurasi di angka 0.9236. Perbedaan performa pada tiap *dropout rate* tidak terlalu signifikan sehingga dapat dikatakan ketiga nilai *dropout rate* ini masih cocok digunakan pada permasalahan ini. Hal ini dapat berarti bahwa model tetap belajar dengan baik pada penonaktifan neuron sebanyak 20%, 30% atau 50%. Selanjutnya nilai *dropout rate* 0.5 digunakan sebagai kontrol uji coba terakhir, yaitu uji coba *learning rate*.

Tabel 4.11 Hasil Uji *Dropout Rate* pada Model MobileNetV3Small pada *Dataset Non-Segmentasi*

MobileNetV3Small				
Dropout Rate	Precision	Recall	F1-Score	Accuracy
0.2	0.9358	0.9148	0.9243	0.9097
0.3	0.936	0.9206	0.9276	0.9097
0.5	0.9461	0.9300	0.9374	0.9236

Tabel 4.12 Hasil Uji *Learning Rate* pada Model MobileNetV3Small pada *Dataset Non-Segmentasi*

MobileNetV3Small				
Learning Rate	Precision	Recall	F1-Score	Accuracy
0.1	0.8364	0.7831	0.8008	0.8542
0.01	0.9396	0.9275	0.9332	0.9167
0.001	0.9091	0.8649	0.8797	0.8958

Tabel 4.12 menunjukkan hasil uji *learning rate*. Nilai *learning rate* yang memberikan performa terbaik adalah 0.01 yang merupakan nilai *default* pada *optimizer* Adagrad. Jika mengacu pada **Tabel 4.12**, nilai *learning rate* 0.1 memberikan performa terburuk. Berdasarkan ujian yang sudah dilakukan, konfigurasi *hyperparameter* terbaik yang didapatkan untuk model MobileNetV3Small pada *dataset* non-segmentasi adalah Adagrad untuk *optimizer*, 512 untuk

hidden units, 0.5 untuk *dropout rate*, dan 0.01 untuk *learning rate*. **Tabel 4.12** menunjukkan hasil uji *learning rate* yang dilakukan.

B. Dataset Tersegmentasi

Pengujian berikutnya dilakukan pada *dataset* tersegmentasi. **Tabel 4.13** menunjukkan hasil uji *optimizer* pada data ini. *Optimizer* yang memberikan hasil terbaik adalah Adagrad dengan nilai *f1-score* di angka 0.9390 dan nilai akurasi di angka 0.9236. Nilai ini tidak berbeda jauh dengan *optimizer* RMSProp yang memiliki nilai *f1-score* di angka 0.9259 dan akurasi di angka 0.9306. Kedua *optimizer* ini memiliki sifat yang sama-sama menggunakan *adaptive learning rate mechanism*, yaitu penyesuaian *learning rate* pada tiap parameter. Selanjutnya, *optimizer* Adagrad digunakan pada uji coba *hidden unit*.

Tabel 4.14 menunjukkan hasil uji *hidden units* pada *dataset* tersegmentasi. Performa terbaik pada uji ini diberikan oleh *hidden units* dengan nilai 128 dengan nilai *f1-score* 0.9112 dan nilai akurasi sebesar 0.9236. Nilai ini tidak berbeda jauh dengan nilai *hidden units* 256 dan *hidden units* 512. *Hidden units* 256 memberikan nilai *f1-score* sebesar 0.9112 dan 0.8958 untuk akurasi. Sementara itu, *hidden units* 512 memberikan nilai *f1-score* sebesar 0.9106 dan akurasi sebesar 0.8958. Secara umum, nilai *f1-score* yang diberikan oleh seluruh *hidden units* cukup baik. Selanjutnya, *hidden units* 128 digunakan dalam uji coba *dropout rate*.

Tabel 4.13 Hasil Uji *Optimizer* Model MobileNetV3Small pada *Dataset* Tersegmentasi

MobileNetV3Small				
<i>Optimizer</i>	<i>Precision</i>	<i>Recall</i>	<i>F1-Score</i>	<i>Accuracy</i>
Adam	0.8713	0.8217	0.8331	0.8681
Adagrad	0.9613	0.9285	0.9390	0.9236
RMSProp	0.9535	0.9067	0.9259	0.9306

Tabel 4.14 Hasil Uji *Hidden Units* Model MobileNetV3Small pada *Dataset* Tersegmentasi

MobileNetV3Small				
<i>Hidden Units</i>	<i>Precision</i>	<i>Recall</i>	<i>F1-Score</i>	<i>Accuracy</i>
128	0.9199	0.9098	0.9112	0.9236
256	0.926	0.9024	0.9112	0.8958
512	0.9318	0.8966	0.9106	0.8958

Tabel 4.15 menunjukkan hasil uji *dropout rate* pada *dataset* tersegmentasi. Nilai *dropout rate* yang memberikan performa terbaik adalah 0.5 dengan nilai *f1-score* 0.9210 dan nilai akurasi sebesar 0.9236. Selanjutnya, *dropout rate* sebesar 0.5 digunakan pada uji *learning rate*.

Tabel 4.15 Hasil Uji *Dropout Rate* Model MobileNetV3Small pada *Dataset* Tersegmentasi

MobileNetV3Small				
<i>Dropout Rate</i>	<i>Precision</i>	<i>Recall</i>	<i>F1-Score</i>	<i>Accuracy</i>
0.2	0.9032	0.9028	0.9011	0.9028
0.3	0.8873	0.8889	0.8830	0.8889
0.5	0.9238	0.9236	0.9210	0.9236

Tabel 4.16 menunjukkan hasil uji *learning rate* pada *dataset* tersegmentasi. Nilai *learning rate* yang memberikan performa terbaik adalah 0.001 dengan nilai *f1-score* sebesar 0.9386 dan nilai akurasi sebesar 0.9236. Nilai ini sudah sangat baik dengan angka *f1-score* dan akurasi

yang diatas 0.9. Berdasarkan hasil uji *hyperparameter* yang sudah dilakukan, didapatkan hasil akhir yaitu Adagrad untuk *optimizer*, 128 untuk *hidden units*, 0.5 untuk *dropout rate*, dan 0.001 untuk *learning rate*.

Tabel 4.16 Hasil Uji *Learning Rate* Model MobileNetV3Small pada *Dataset Tersegmentasi*

MobileNetV3Small				
<i>Learning Rate</i>	<i>Precision</i>	<i>Recall</i>	<i>F1-Score</i>	<i>Accuracy</i>
0.1	0.8782	0.7121	0.7409	0.8333
0.01	0.9213	0.8958	0.9049	0.9167
0.001	0.9432	0.9344	0.9386	0.9236

4.1.4 Uji Coba Perbandingan MobileNetV3Small vs Large

Percobaan ini bertujuan untuk membandingkan performa antara MobileNetV3Small dengan MobileNetV3Large. Skenario yang diujikan ini adalah skenario nomor 7 pada **Tabel 4.1**. Aspek yang ingin dilihat dari uji coba ini adalah *metrics* evaluasi dan juga efisiensi dari kedua model. Aspek efisiensi tersebut berupa waktu prediksi untuk 144 data *test* dan ukuran kedua model tersebut.

Tabel 4.17 menunjukkan hasil *training* dari kedua model. Dari sisi *f1-score*, model MobileNetV3Small memberikan hasil yang lebih baik dari model MobileNetV3Large, yaitu 0.8418 untuk MobileNetV3Small dan 0.8127 untuk MobileNetV3Large. Selanjutnya, dari sisi akurasi, model MobileNetV3Small memberikan akurasi yang lebih baik, tetapi berbeda tipis dengan MobileNetV3Large, yaitu 0.8889 untuk MobileNetV3Small dan 0.8819 untuk MobileNetV3Large. Sementara itu, dari aspek efisiensi, model MobileNetV3Small unggul di waktu prediksi dengan angka 2.68 detik dibandingkan dengan MobileNetV3Large yang memiliki waktu prediksi di angka 2.88 detik. Dari ukuran model, tentu MobileNetV3Small memiliki ukuran yang lebih kecil, yaitu 170 MB dibandingkan MobileNetV3Large yang berukuran 287.9 MB.

Tabel 4.17 Hasil Uji Coba Model MobileNetV3Large vs MobileNetV3Small

<i>Model</i>	<i>Precision</i>	<i>Recall</i>	<i>F1-Score</i>	<i>Accuracy</i>	<i>Prediction Time (s)</i>	<i>Model Size (MB)</i>
MobileNetV3Large	0.8326	0.8029	0.8127	0.8819	2.88	287.9
MobileNetV3Small	0.9037	0.8278	0.8418	0.8889	2.68	169.7

Berdasarkan **Tabel 4.13**, model MobileNetV3Small unggul dari seluruh sisi *metric* evaluasi dan efisiensi sehingga dapat disimpulkan model MobileNetV3Small memiliki performa yang lebih baik pada skenario ini. Perlu diingat bahwa, *hyperparameter* yang digunakan pada kedua model di skenario ini adalah identik.

4.1.5 Uji Coba Model VGG19 pada *Dataset Tersegmentasi*

Percobaan ini dilakukan untuk melihat performa model VGG19 pada klasifikasi citra yang disegmentasi dengan *backbone* VGG19 juga. Uji coba ini merupakan skenario nomor 14 pada **Tabel 4.1**. *Hyperparameter* yang digunakan pada model ini adalah Adam untuk *optimizer*, 512 untuk *hidden units*, 0.5 untuk *dropout rate*, dan 0.01 untuk *learning rate*. **Tabel 4.18** menunjukkan hasil klasifikasi pada *dataset tersegmentasi* dengan model VGG19. Hasil uji ini memberikan nilai *f1-score* sebesar 0.8260 dan nilai akurasi sebesar 0.8611.

Tabel 4.18 Hasil Uji Model VGG19 pada *Dataset* Tersegmentasi

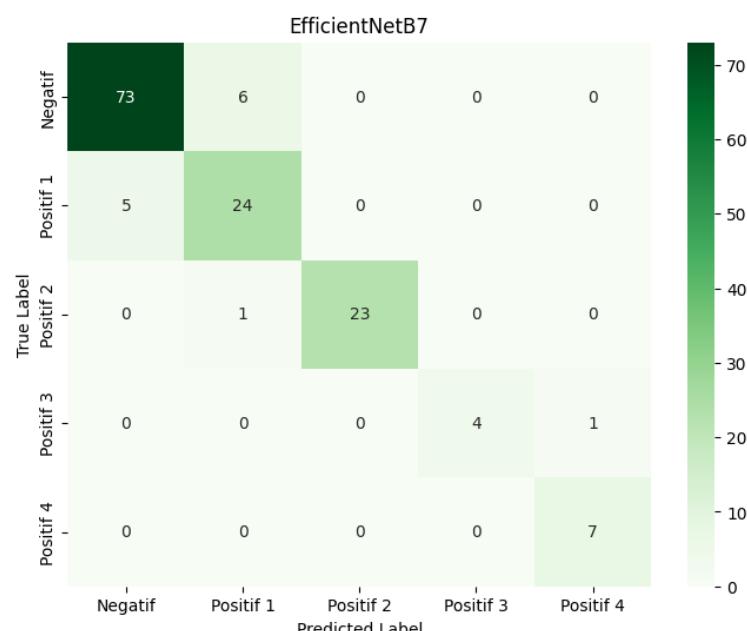
Model VGG19			
Precision	Recall	F1-Score	Accuracy
0.8134	0.8655	0.8260	0.8611

4.2 Pembahasan

Bab ini akan berisi tentang pembahasan dari seluruh skenario yang sudah dilakukan serta beberapa temuan-temuan yang didapat melalui skenario-skenario yang sudah dilakukan.

4.2.1 Pembahasan Uji Skenario *Hyperparameter Default*

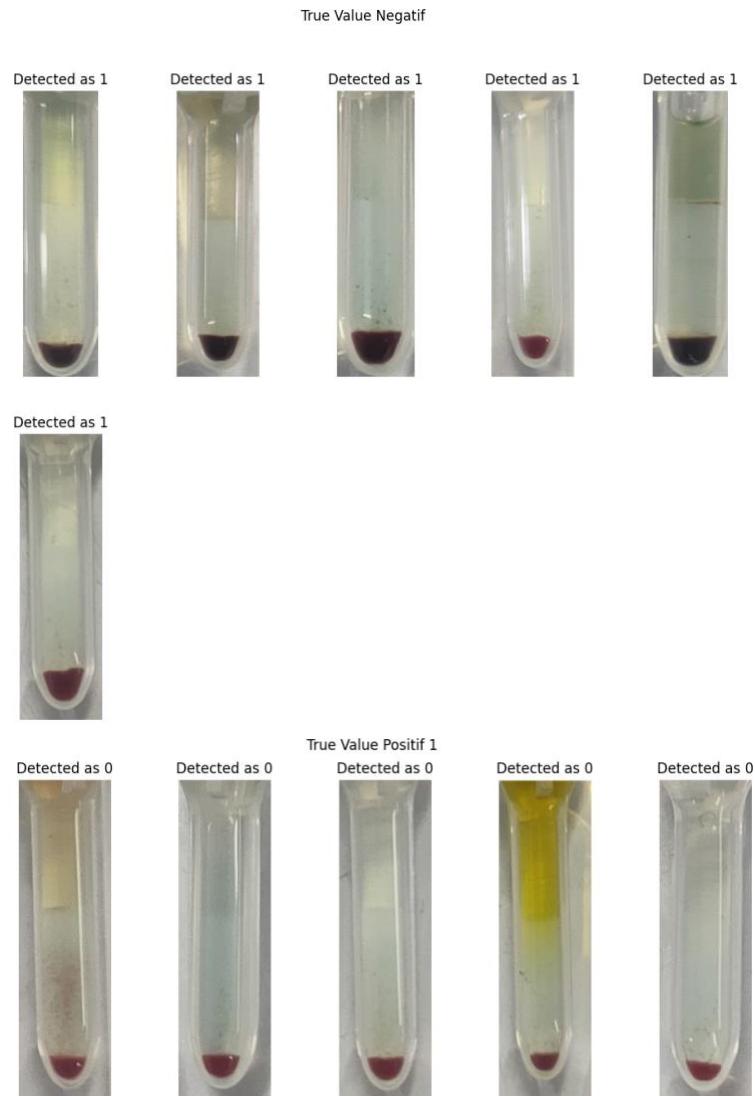
Berdasarkan hasil uji skenario *hyperparameter default* yang dilakukan pada kedua *dataset*, beberapa hal dibahas dalam subbab ini. Mengacu pada **Tabel 4.3** yang memberikan hasil uji pada *dataset* non-segmentasi, model yang memberikan performa terbaik adalah EfficientNetB7 dengan *f1-score* sebesar 0.9062. Meskipun nilai *f1-score* yang diberikan cukup tinggi, model ini masih memiliki beberapa kesalahan prediksi. **Gambar 4.1** Menunjukkan *confusion matrix* pada model EfficientNetB7 yang di-train dengan *dataset* non-segmentasi pada skenario ini. Berdasarkan *confusion matrix*, kelas Positif 1 dan Negatif merupakan kelas yang paling banyak gagal diprediksi dengan benar oleh model EfficientNetB7 dengan kesalahan prediksi sebanyak 6 citra pada kelas Negatif dan 5 citra pada kelas Positif 1. **Gambar 4.2** menunjukkan citra kelas Negatif dan Positif 1 yang gagal diprediksi oleh model EfficientNetB7.



Gambar 4.1 *Confusion Matrix* EfficientNetB7 Uji Skenario *Hyperparameter Default* pada *Dataset* Non-Segmentasi

Jika diperhatikan, seluruh kesalahan prediksi citra Negatif diprediksi sebagai kelas Positif 1. Sebaliknya, seluruh kesalahan prediksi citra Positif 1 diprediksi sebagai Negatif. Mengacu pada gambar, citra-citra yang salah diprediksi terlihat memiliki pola yang mirip. Pada kelas Negatif, beberapa citra memiliki *noise* (noda) yang tidak relevan sehingga model dapat menganggap bahwa hal tersebut merupakan bercak darah yang akhirnya menyebabkan citra

tampak seperti kelas Positif 1. Mengingat model EfficientNetB7 yang kompleks, masuk akal apabila model menangkap detail yang tidak relevan tersebut dan memberikan prediksi yang salah. Dari sisi, kelas Positif 1, ada kecendrungan kesalahan prediksi diakibatkan oleh detail *irrelevant* lain yang mirip dengan citra kelas Negatif, seperti distribusi darah yang pekat pada bagian bawah tabung, warna gel yang mirip, kontras citra yang mirip, dan lain sebagainya.

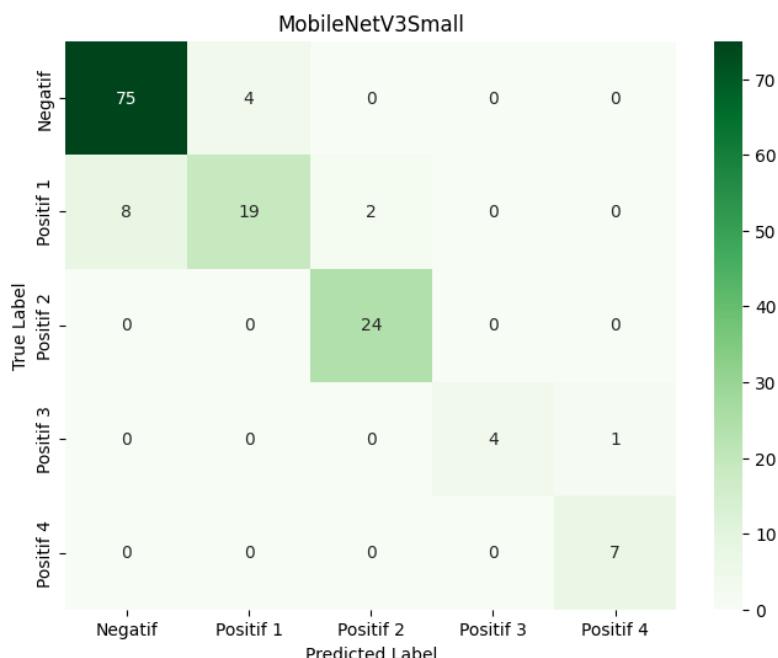


Gambar 4.2 Citra Kesalahan Prediksi EfficientNetB7 pada *Dataset* Non-Segmentasi Skenario *Hyperparameter Default*

Pada penggunaan *dataset* tersegmentasi, model yang memberikan performa terbaik adalah MobileNetV3Small, sesuai dengan **Tabel 4.4**. Jika diperhatikan pada **Gambar 4.3** yang menunjukkan *confusion matrix* model, mayoritas kesalahan prediksi model MobileNetV3Small berada pada citra dengan kelas Positif 1 dengan kesalahan prediksi berjumlah 10 citra. Dari 10 citra tersebut, 8 diprediksi sebagai kelas Positif 1 dan 2 diprediksi sebagai kelas Positif 2. Seluruh *confusion matrix* pada skenario *hyperparameter default* dapat dilihat pada **lampiran L1** hingga **L8**. Gambar **4.4** menunjukkan beberapa citra yang salah diprediksi model MobileNetV3Small pada skenario ini. Struktur dari citra ini adalah citra kiri menunjukkan citra *original* (yang tidak tersegmentasi), citra tengah menunjukkan *ground truth* segmentasi, dan citra kanan menunjukkan hasil segmentasi (citra yang digunakan untuk *training*).

Mengacu pada **Gambar 4.4**, citra pertama memiliki citra original yang sudah sesuai dengan *ground truth* segmentasi yang sesuai, tetapi hasil segmentasi (*predicted segmented*) yang tidak sesuai. Dapat dilihat bahwa hasil segmentasi yang di dapat dari arsitektur U-Net dengan *backbone* VGG19 masih menangkap bercak-bercak yang *irrelevant* sehingga masuk akal apabila model mengira bahwa citra 522.jpg memiliki kelas Positif 1. Nilai *iou score* citra 522.jpg ini adalah 0.8307. Selanjutnya, untuk citra 532.jpg dengan kelas asli Negatif, citra *original* tampak sudah benar, tetapi *ground truth* segmentasi tampak masih memberikan bercak darah. Hal ini menyebabkan citra hasil segmentasi memiliki bercak darah di bagian atas sehingga model MobileNetV3Small mengira bahwa citra 532.jpg merupakan kelas Positif 1. *Iou score* dari citra 532.jpg adalah 0.6719.

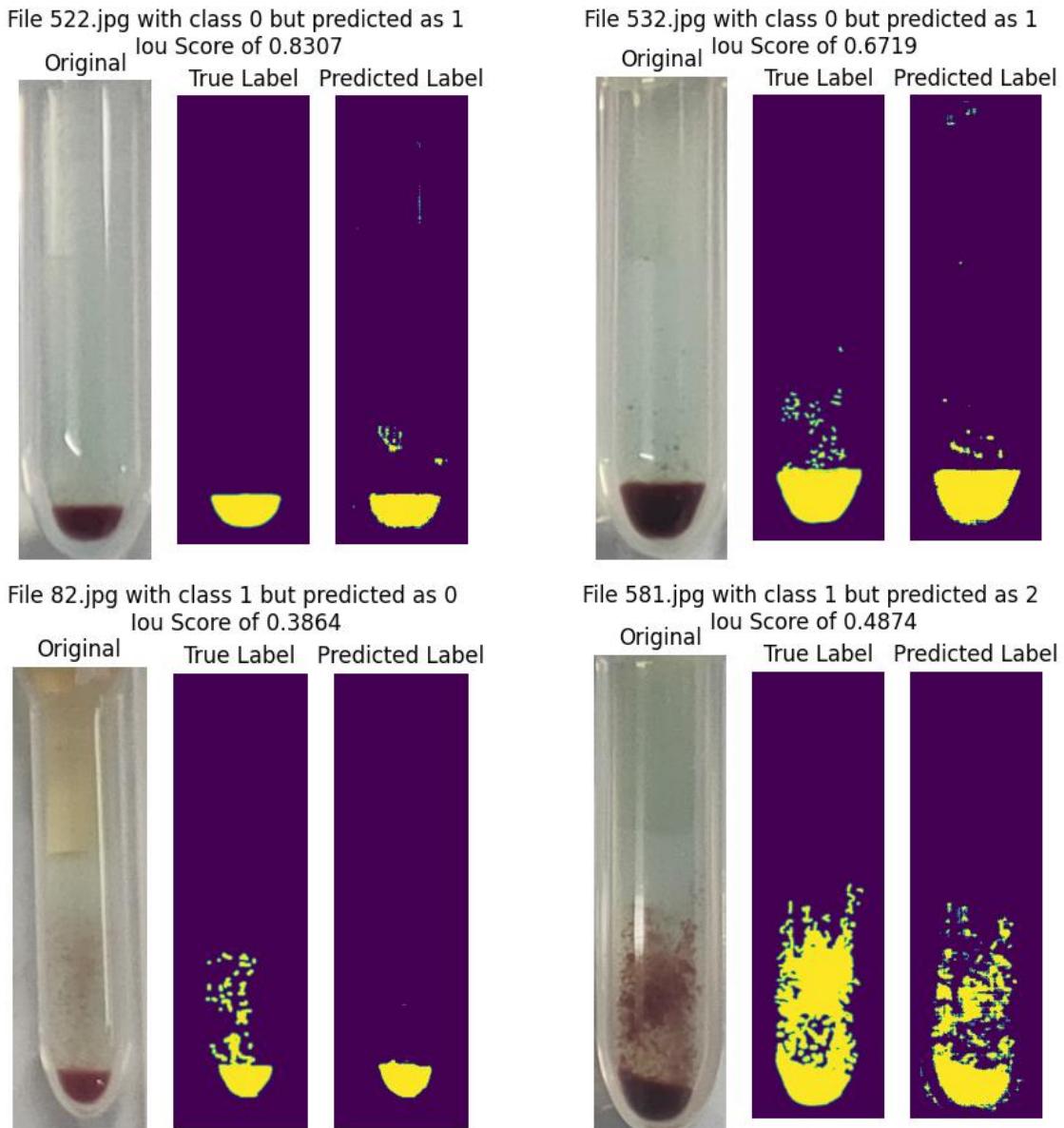
Disisi lain, citra 82.jpg dan 224.jpg memiliki kelas Positif 1, tetapi citra hasil segmentasi yang digunakan untuk *training* tampak seperti data dengan kelas Negatif sehingga wajar jika model memprediksi citra-citra tersebut sebagai kelas Negatif. Citra 581.jpg dengan kelas Positif 1 sedikit berbeda dengan kasus-kasus sebelumnya, dimana citra ini diprediksi oleh model sebagai citra dengan kelas Positif 2. Jika diperhatikan, citra ini sangat mirip dengan citra kelas Positif 2, dimana distribusi darahnya tampak merata. Namun, terdapat distribusi darah yang tebal pada bagian dasar tabung sehingga citra ini benar dikategorikan sebagai Positif 2. Kedua citra 82.jpg dan 224.jpg secara Secara umum, kesalahan prediksi terjadi akibat model segmentasi yang kurang *robust*, utamanya dalam mensegmentasi citra dengan kelas Positif 1.



Gambar 4.3 Confusion Matrix MobileNetV3Small Uji Skenario *Hyperparameter Default* pada *Dataset* Tersegmentasi

Mengacu pada hasil evaluasi yang sudah dilakukan pada kedua *dataset*, performa *pre-trained* model yang di-*train* dengan *dataset* tersegmentasi memiliki performa keseluruhan yang lebih baik dibandingkan dengan *dataset* non-segmentasi. Hal ini dapat dilihat pada **Gambar 4.5** yang menunjukkan perbandingan nilai *metrics* evaluasi rata-rata pada kedua *dataset*, dimana *dataset* tersegmentasi unggul dari setiap *metric* penilaian. Meskipun *dataset* tersegmentasi unggul dari setiap *metric*, hasil dari *dataset* non-segmentasi tidak jauh berbeda dengan *dataset* tersegmentasi. Hal ini menunjukkan pada skenario ini, *dataset* tersegmentasi

memberikan peningkatan tetapi tidak terlalu signifikan. Skenario ini juga menunjukkan bahwa performa rata-rata kedua *dataset* tidak ada yang menyentuh angka 0.9 ke atas.



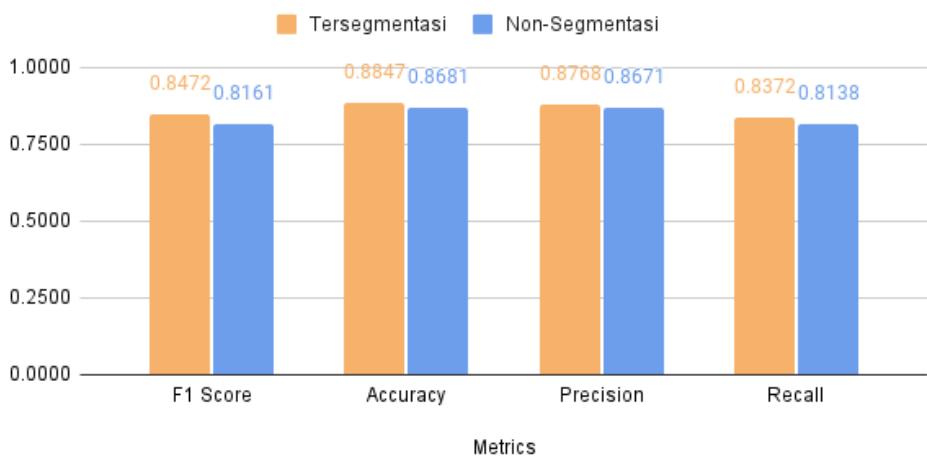
Gambar 4.4 Contoh Citra Kesalahan Prediksi MobileNetV3Small pada *Dataset* Tersegmentasi Skenario *Hyperparameter Default*

Berdasarkan **Gambar 4.6**, *F1-Score* pada *dataset* tersegmentasi unggul pada setiap model dibandingkan dengan *dataset* non-segmentasi kecuali di model EfficientNetB7 yang berhasil menyentuh angka 0.9062 pada *dataset* non-segmentasi. **Gambar 4.7** menunjukkan perbandingan nilai akurasi pada kedua *dataset*. Nilai akurasi pada *dataset* tersegmentasi masih lebih unggul dibandingkan dengan *dataset* non-segmentasi kecuali pada model EfficientNetB7 yang berada di angka 0.9097 pada *dataset* tersegmentasi.

Model EfficientNetB7 mengalami penurunan dari sisi akurasi dan *f1-score* saat diuji pada *dataset* tersegmentasi. Hasil ini menunjukkan adanya kecenderungan model EfficientNetB7 untuk mengalami *overfitting* dalam skenario tersebut, di mana model cenderung terlalu fokus pada detail-detail yang spesifik dari *testset*. Secara arsitektur, EfficientNetB7 memiliki struktur yang sangat kompleks yang secara efisien menangkap detail-detail yang rumit pada citra.

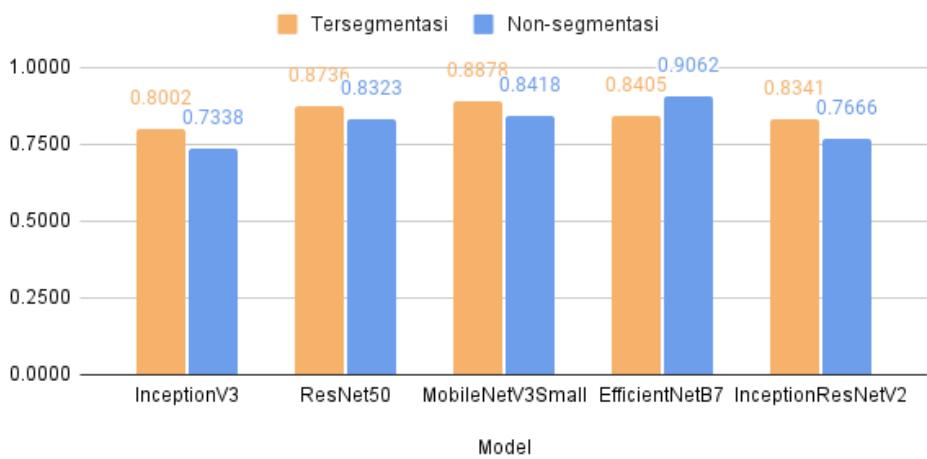
Namun, pada *dataset* tersegmentasi yang lebih sederhana, kompleksitas ini mungkin menjadi terlalu besar, menyebabkan model sulit untuk men-generalize prediksi dengan baik. Oleh karena itu, hasil prediksi yang diberikan oleh model EfficientNetB7 pada *dataset* tersegmentasi di kasus ini tidak mencapai tingkat performa yang diharapkan. Selain itu, *hyperparameter* yang digunakan pada skenario ini tidak ter-tuned sehingga ada kecendrungan kombinasi *hyperparameter* yang digunakan tidak cocok dengan proses klasifikasi uji silang serasi dengan *dataset* tersegmentasi. Oleh karena itu, perlu dilakukan uji *hyperparameter* untuk mendapatkan hasil yang lebih optimal pada model EfficientNetB7 dalam mengklasifikasi citra uji silang serasi.

Perbandingan Rata-rata Nilai Metrics Evaluasi - Data Tersegmentasi vs Non-Segmentasi



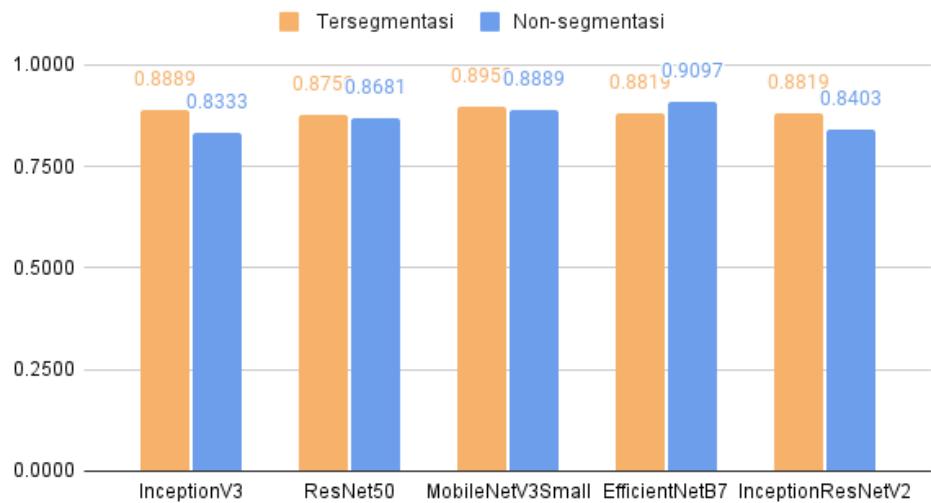
Gambar 4.5 Perbandingan Rata-Rata Nilai *Metrics* Evaluasi pada *Dataset* Tersegmentasi dan Non-Segmentasi

Perbandingan F1Score - Data Tersegmentasi vs Non-Segmentasi



Gambar 4.6 Perbandingan *F1-Score* pada *Dataset* Tersegmentasi dan Non-Segmentasi

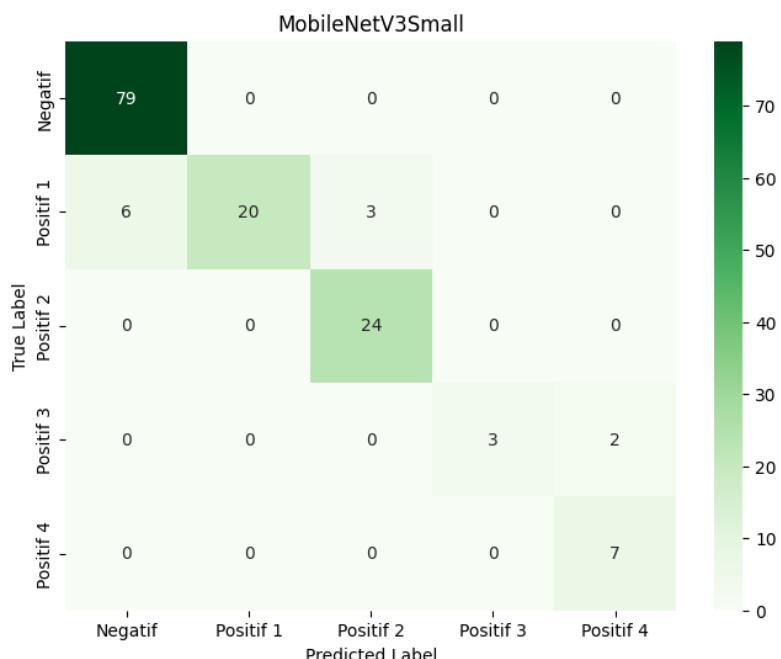
Perbandingan Akurasi - Data Tersegmentasi vs Non-Segmentasi



Gambar 4.7 Perbandingan Akurasi pada *Dataset* Tersegmentasi dan Non-Segmentasi

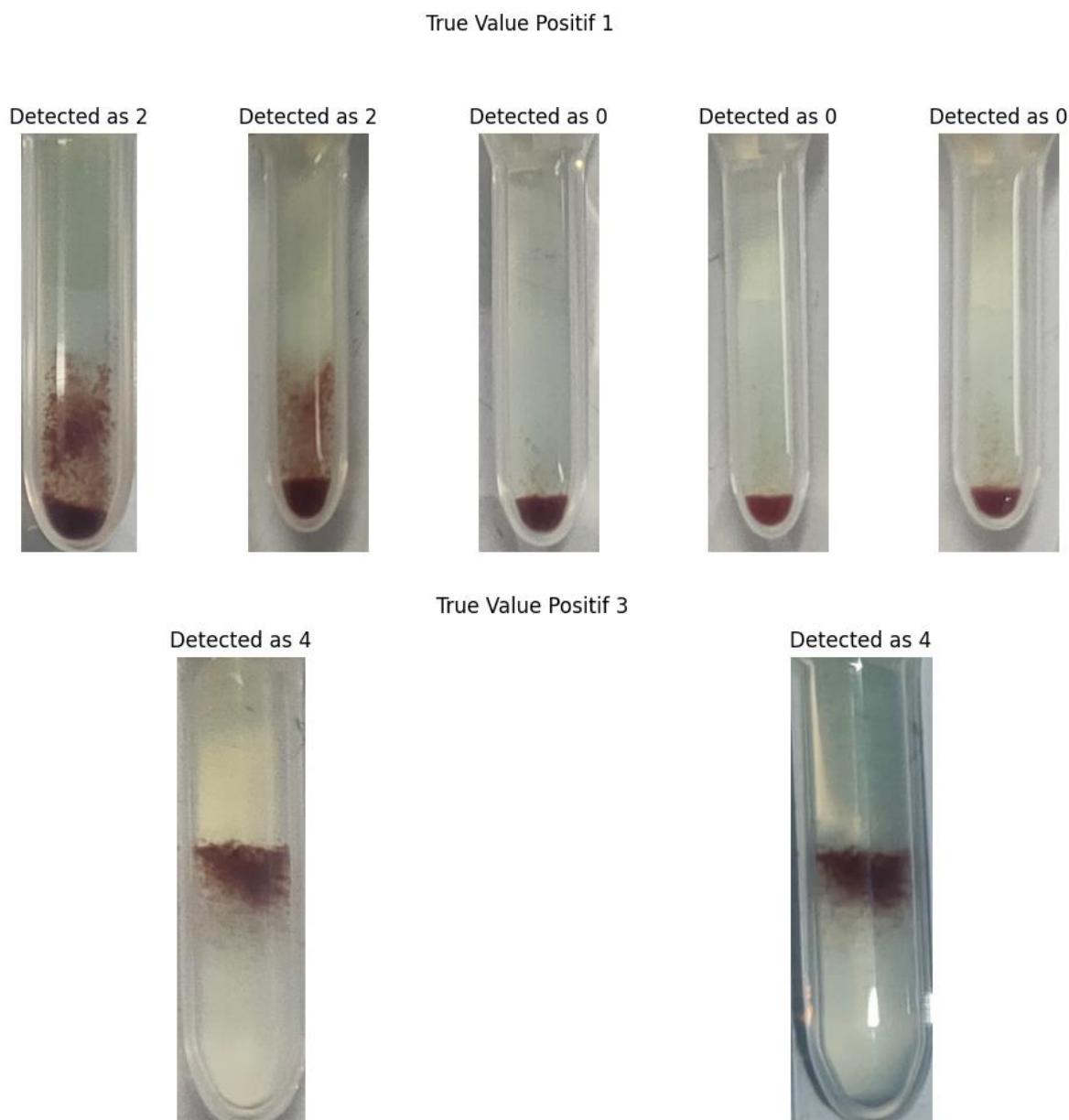
4.2.2 Pembahasan Uji Skenario *Hyperparameter Tuning*

Uji skenario *hyperparameter tuning* dilakukan pada kedua *dataset*. Pada **Tabel 4.5** yang menunjukkan hasil uji skenario ini pada *dataset* non-segmentasi, model MobileNetV3Small memberikan performa terbaik. **Gambar 4.8** menunjukkan *confusion matrix* model MobileNetV3Small pada *dataset* non-segmentasi. Mengacu pada *confusion matrix*, kesalahan prediksi utamanya terjadi pada kelas Positif 1 dan kelas Positif 3. Pada kelas Positif 1, kesalahan prediksi berjumlah 9 citra dengan 6 citra diprediksi sebagai kelas Negatif dan 3 citra diprediksi sebagai kelas Positif 2. Sementara itu pada kelas Positif 3, kesalahan prediksi berjumlah 2 citra yang semuanya diprediksi sebagai kelas Positif 4. **Gambar 4.9** menunjukkan beberapa contoh citra yang kelas Positif 1 dan Positif 3 yang salah diprediksi oleh model MobileNetV3Small.



Gambar 4.8 *Confusion Matrix* Uji Skenario *Hyperparameter Tuning* Pada *Dataset* Tersegmentasi

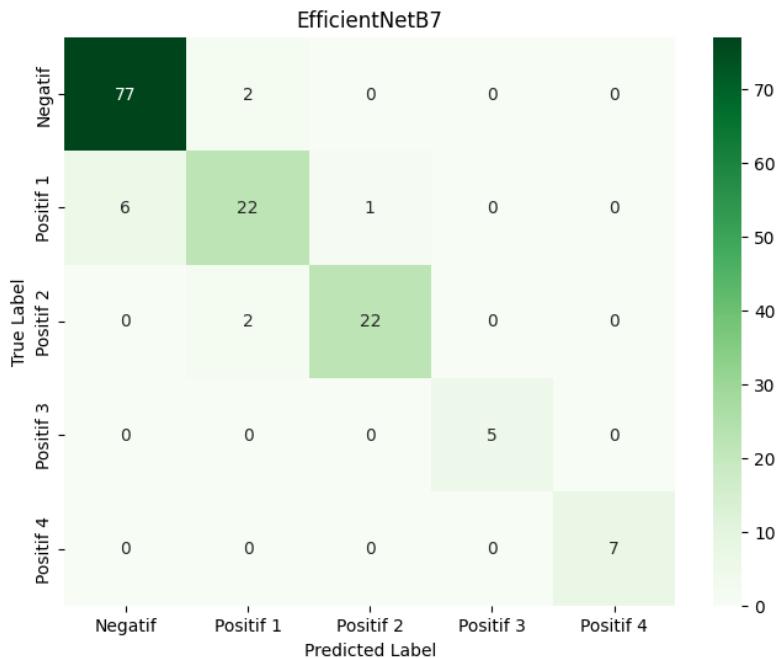
Mengacu pada **Gambar 4.9**, citra-citra kelas Positif 1 yang terdeteksi sebagai Positif 2 memiliki bercak darah yang cukup banyak. Hal ini cenderung membuat model mengira bahwa citra yang diberikan merupakan kelas Positif 2. Sementara itu, pada citra kelas Positif 1 yang diprediksi sebagai kelas Negatif, bercak darah muncul sangat sedikit sehingga bentuknya sangat mirip dengan kelas Negatif. Selanjutnya, pada citra kelas Positif 3 yang diprediksi sebagai Positif 4, citra memang selayaknya tampak seperti citra Positif 3 dengan distribusi darah tebal pada bagian tengah tabung. Berdasarkan inspeksi yang dilakukan, model MobileNetV3 tampak gagal dalam menangkap detail-detail relevan pada citra, utamanya pada citra kelas Positif 1. Hal ini cenderung dikarenakan struktur MobileNetV3Small yang tidak kompleks sehingga kemampuannya dalam menangkap detail pada skenario ini kurang baik.



Gambar 4.9 Citra Kesalahan Prediksi MobileNetV3Small pada *Dataset Non-Segmentasi Skenario Hyperparameter Tuning*

Pada *dataset* tersegmentasi, model yang memberikan performa terbaik adalah EfficientNetB7, sesuai dengan **Tabel 4.6**. Model ini memiliki skore *f1-score* yang tinggi, yaitu

0.9373. Meskipun model EfficientNetB7 memiliki nilai *f1-score* yang baik, model ini masih salah dalam memprediksi beberapa citra. Berdasarkan **Gambar 4.10** yang menunjukkan *confusion matrix* model EfficientNetB7, model EfficientNetB7 masih gagal dalam memprediksi citra dengan kelas Positif 1, dengan 6 citra diprediksi sebagai kelas Negatif dan 1 citra diprediksi sebagai Positif 2. Seluruh *confusion matrix* pada skenario *hyperparameter tuning* dapat dilihat pada **lampiran L9** hingga **L16**.



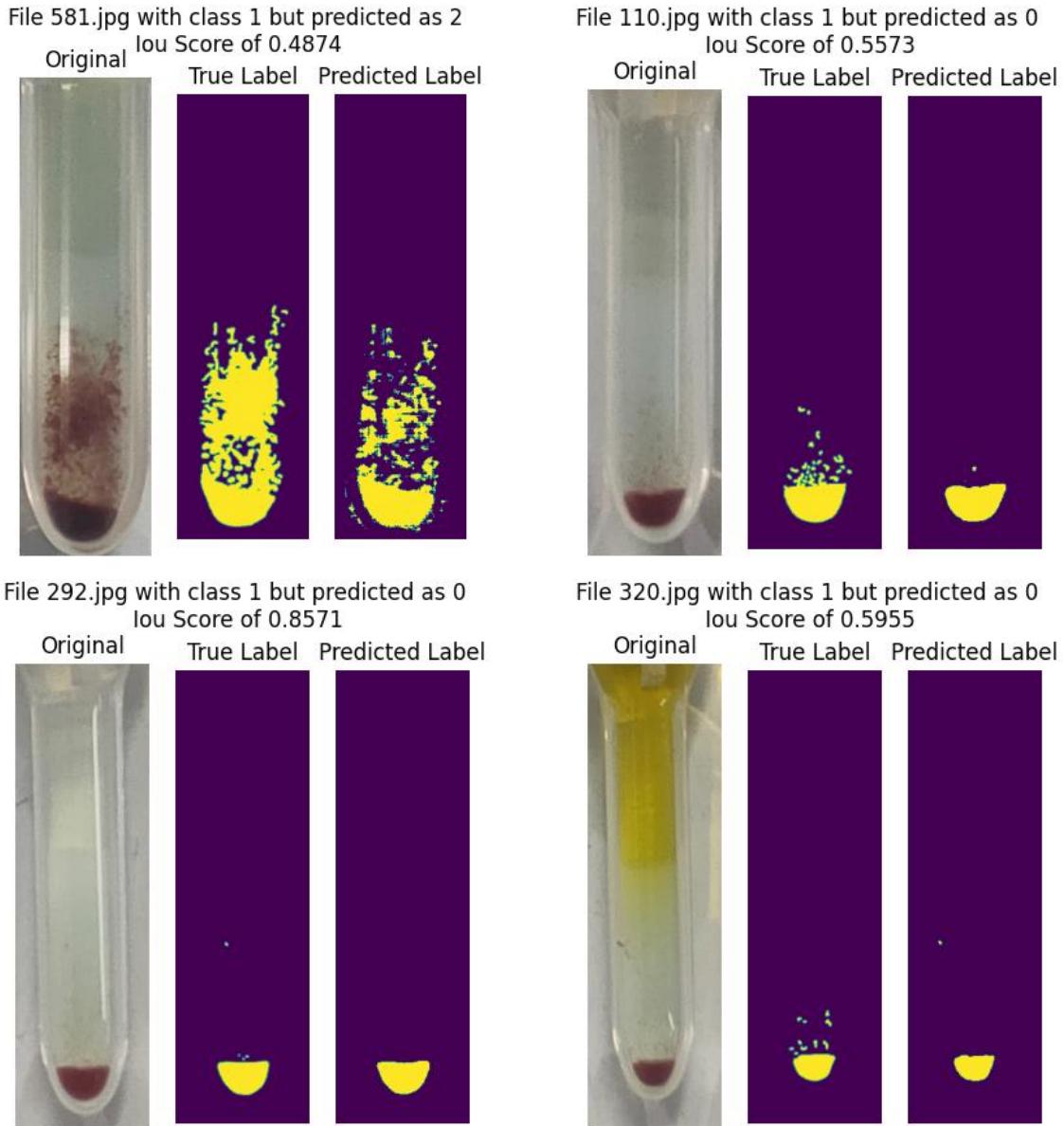
Gambar 4.10 *Confusion Matrix* EfficientNetB7 Uji Skenario *Hyperparameter Tuning* pada *Dataset* Tergmentasi

Jika mengacu pada **Gambar 4.11** yang menunjukkan beberapa contoh citra yang salah diprediksi oleh model EfficientNetB7, seluruh citra segmentasi kelas Positif 1 yang diprediksi sebagai Negatif memang nampak seperti citra dengan kelas Negatif. Jika diperhatikan, file 110.jpg 320.jpg memiliki *ground truth* yang sudah benar, tetapi hasil segmentasi yang diberikan masih tidak tepat. Bercak-bercak darah pada *ground truth* tidak tertangkap oleh citra hasil segmentasi yang digunakan dalam proses training sehingga masuk akal jika model klasifikasi mengira citra sebagai kelas Negatif. Kesalahan ini tentu berasal dari model segmentasi yang tidak mampu menangkap detail bercak darah pada *ground truth*. Untuk file 292.jpg, citra *ground truth* tampak terlihat seperti kelas Negatif sehingga kesalahan ini berasal dari *ground truth*. Sementara itu, untuk file 581.jpg, citra ini memiliki distribusi darah yang mirip dengan kelas Positif 2 hanya saja konsentrasi darah masih berada pada dasar tabung. Ini termasuk kesalahan model karena sisi citra tersegmentasi sudah sesuai.

Disisi lain, nilai *iou score* yang dimiliki citra yang salah prediksi menunjukkan nilai yang relative rendah. Contohnya pada citra 581.jpg, 110.jpg, dan 320.jpg yang memberikan nilai *iou score* secara berurutan, yaitu 0.4874, 0.5573, dan 0.5955. Nilai ini menunjukkan adanya area pada citra hasil segmentasi yang berbeda dengan citra *ground truth*. Oleh karena itu, masuk akal jika model masih keliru dalam memprediksi citra-citra ini.

Setelah melakukan uji coba pada kedua *dataset* di skenario ini, dapat disimpulkan bahwa *dataset* tersegmentasi memberikan performa yang lebih baik dibandingkan dengan *dataset* non-segmentasi pada skenario ini. Hal ini dapat dilihat pada **Gambar 4.12** yang menunjukkan

perbandingan rata-rata performa keseluruhan model antara *dataset* tersegmentasi dengan non-segmentasi. *Dataset* tersegmentasi unggul disetiap *metric* penilaian meskipun tidak berbeda jauh dengan *dataset* non-segmentasi. Ini berarti bahwa *dataset* tersegmentasi memberikan performa yang lebih baik, tetapi tidak signifikan.

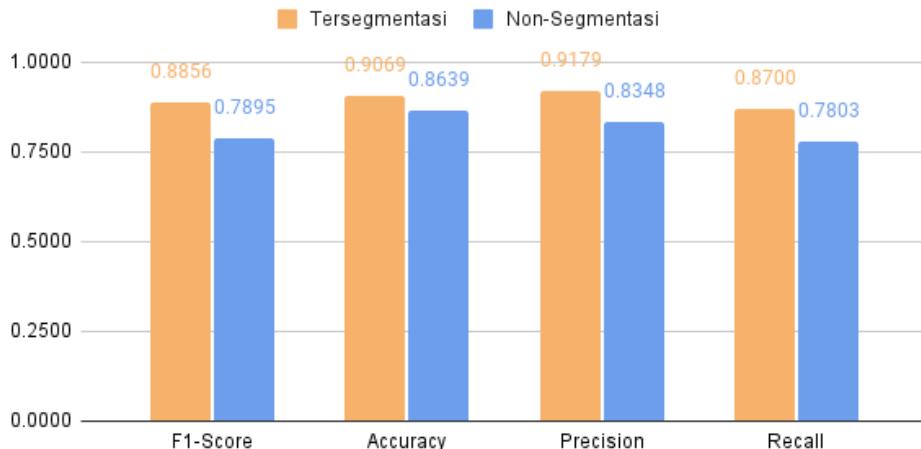


Gambar 4.11 Contoh Citra Kesalahan Prediksi EfficientNetB7 pada *Dataset* Tersegmentasi Skenario Hyperparameter Tuning

Beberapa model juga sudah menyentuh *f1-score* dan akurasi di angka 0.9, dimana pada skenario *default*, hanya model EfficientNetB7 yang mampu menyentuh angka ini. Hal ini dapat dilihat pada **Gambar 4.13** dan **Gambar 4.14** yang menunjukkan perbandingan *f1-score* dan akurasi pada kedua *dataset*. Dari sisi *f1-score*, model ResNet50 dan EfficientNetB7 berhasil menyentuh angka 0.9120 dan 0.9374. Nilai ini menunjukkan bahwa nilai harmonik antara *precision* dan *recall* berada di angka yang baik. Dengan nilai tersebut, dapat dikatakan bahwa kedua model ini sudah *robust* dalam mengatasi permasalahan *imbalance classification*. Dari *metric* akurasi, model ResNet50 dan EfficientNetB7 memiliki nilai 0.9306 dan 0.9236 untuk *dataset* tersegmentasi. Hal ini juga sudah menunjukkan performa model yang baik. Hal menarik

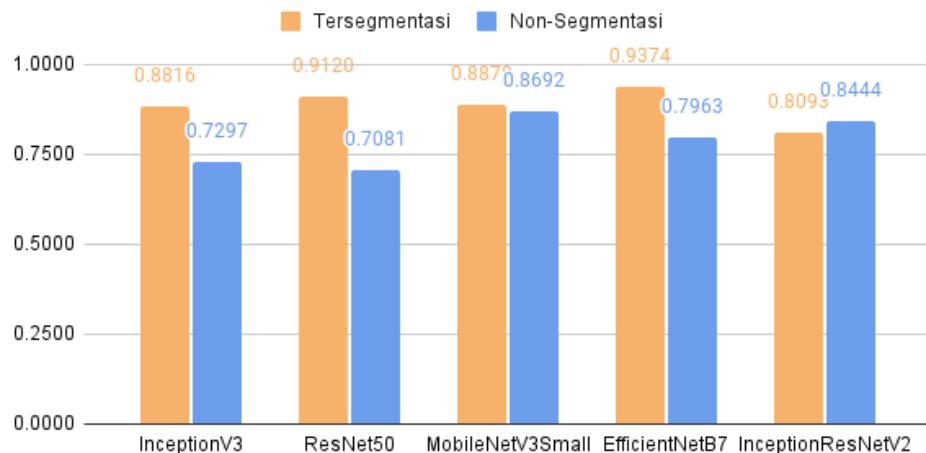
lainnya yang dapat dilihat adalah model InceptionResNetV2 yang *f1-score* data non-segmentasinya lebih besar dibandingkan dengan data tersegmentasi. Nilai ini tidak terlalu signifikan, yaitu pada *dataset* tersegmentasi, *f1-score* berada di angka 0.8093 dan pada *dataset* non-segmentasi berada di angka 0.8444.

Perbandingan Rata-Rata Nilai Metrics Evaluasi Hyperparameter Tuning - Data Tersegmentasi vs Non-Segmentasi



Gambar 4.12 Perbandingan Rata-Rata Nilai *Metrics* Evaluasi *Hyperparameter Tuning* pada *Dataset* Tersegmentasi dan Non-Segmentasi

Perbandingan F1-Score Hyperparameter Tuning - Data Tersegmentasi vs Non-Segmentasi

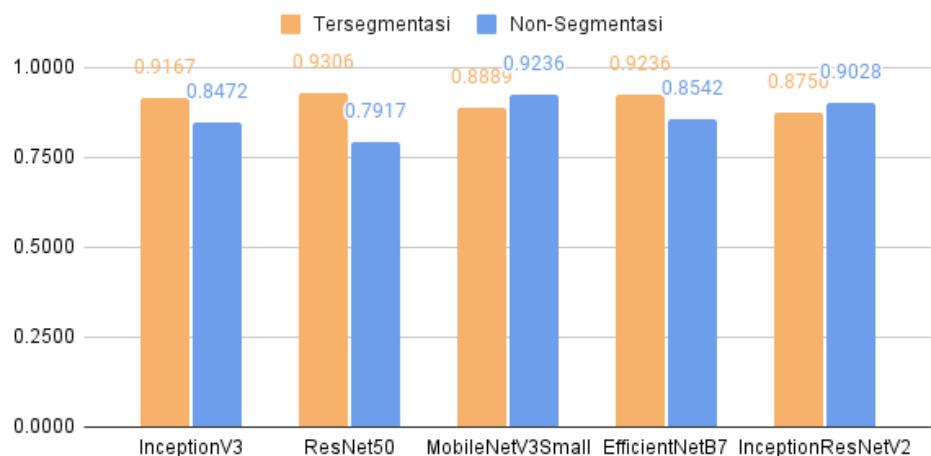


Gambar 4.13 Perbandingan *F1-Score* *Hyperparameter Tuning* pada *Dataset* Tersegmentasi dan Non-Segmentasi

Gambar 4.15 menunjukkan perbandingan performa *f1-score* antara *hyperparameter default* dengan *hyperparameter turning* pada *dataset* non-segmentasi. Berdasarkan **Gambar 4.15**, *hyperparameter tuning* justru menurunkan performa pada model ResNet50 dan EfficientNetB7. Nilai *max trial* 20 berpotensi menjadi penyebab penurunan performa kedua model ini. Hal ini disebabkan karena adanya kombinasi parameter yang belum dieksplorasi oleh KerasTuner. Kombinasi *hyperparameter* maksimal berdasarkan opsi parameter yang ada

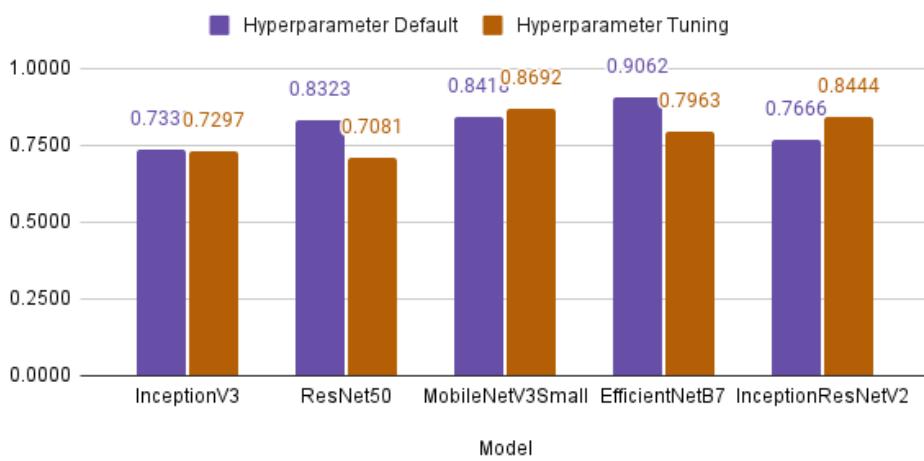
adalah 81. Dengan *max trial* sejumlah 20, masih ada kombinasi parameter yang lebih cocok untuk kedua model yang belum dieksplorasi.

Perbandingan Akurasi Hyperparameter Tuning - Data Tersegmentasi vs Non-Segmentasi



Gambar 4.14 Perbandingan *F1-Score Hyperparameter Tuning* pada *Dataset Tersegmentasi* dan *Non-Segmentasi*

Perbandingan F1-Score pada Dataset Non-Segmentasi - Hyperparameter Default vs Tuning

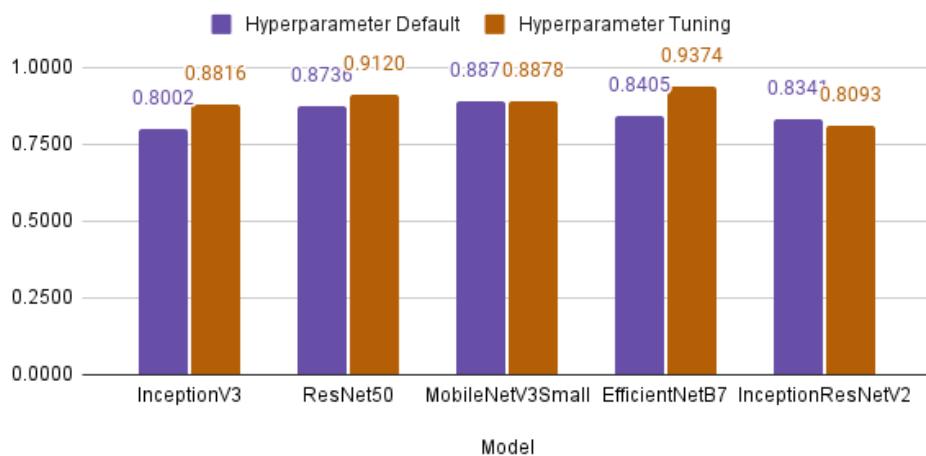


Gambar 4.15 Perbandingan *F1-Score* pada *Dataset Non-Segmentasi - Hyperparameter Default vs Tuning*

Dengan demikian, *hyperparameter tuning* hanya bekerja pada sebagian model di *dataset* non-segmentasi pada kasus ini. Selanjutnya, pada **Gambar 4.16** yang menampilkan performa *f1-score* pada *dataset* tersegmentasi, seluruh model menunjukkan peningkatakan performa kecuali pada model InceptionResNetV2. Hal ini menunjukkan bahwa secara umum, *hyperparameter tuning* meningkatkan performa pada *dataset* tersegmentasi. Penuruan performa pada model InceptionResNetV2 dapat diakibatkan karena belum tereksplorinya kombinasi *hyperparameter* yang sesuai untuk model tersebut pada *dataset* tersegmentasi. Perlu

dilakukannya uji kombinasi *hyperparameter* menyeluruh pada model InceptionResNetV2 untuk memastikan hal tersebut.

Perbandingan F1-Score pada Dataset Tersegmentasi - Hyperparameter Default vs Tuning



Gambar 4.16 Perbandingan *F1-Score* pada Dataset Tersegmentasi - *Hyperparameter Default vs Tuning*

4.2.3 Pembahasan Uji Skenario Pengaruh *Hyperparameter* Pada Model Kontrol MobileNetV3Small

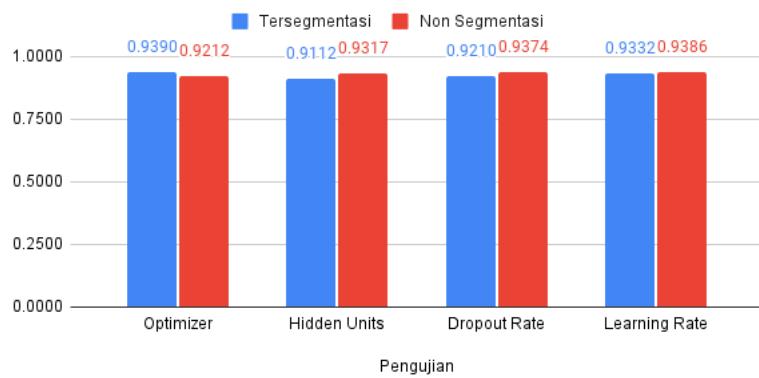
Pengujian pengaruh *hyperparameter* ini dilakukan pada kedua *dataset*. Pada *dataset* non-segmentasi, hasil uji *optimizer* menunjukkan Adagrad sebagai *optimizer* dengan performa terbaik sesuai dengan **Tabel 4.9**. Adagrad memiliki kecendrungan yang baik dalam permasalahan klasifikasi dengan data yang bersifat *sparse* (Ault, 2020). Data *sparse* berarti bahwa data memiliki fitur-fitur yang tidak beraturan atau berbeda-beda pada sampel. Dalam penilitian ini, citra darah pada tiap kelas maupun intra-kelas memiliki pola bercak darah yang rumit, khususnya pada kelas Positif 1, Positif 2, dan Positif 3. Fitur-fitur yang sifatnya tidak beraturan ini mengacu pada bercak-bercak darah pada citra. Adagrad memperbarui parameternya dengan memberikan *weights* yang lebih besar pada parameter yang jarang diperbarui atau parameter yang terasosiasi dengan fitur yang tidak beraturan sehingga hasil yang didapatkan optimal. *Optimizer-optimizer* lain sejatinya memiliki sifat yang sama seperti Adagrad, yaitu *adaptive learning*, tetapi dengan konfigurasi *hyperparameter* pada pengujian ini, Adagrad cenderung memberikan hasil yang lebih optimal. Berikutnya, pada hasil uji *hidden units* yang ditunjukkan **Tabel 4.10**, *hidden units* 512 menunjukkan performa terbaik pada *dataset* non-segmentasi. Hal ini dapat berarti bahwa model membutuhkan *hidden units* yang lebih untuk mempelajari pola pada citra. Hal ini masuk akal karena data yang digunakan pada skenario ini adalah data non-segmentasi sehingga pola-pola pada citra belum disederhanakan dan membutuhkan unit neuron lebih untuk mempelajari pola. Selain itu, model yang digunakan pada skenario ini adalah MobileNetV3Small yang secara arsitektur kurang begitu kompleks. Oleh karena itu, masuk akal jika *hidden units* bernilai 512 memberikan performa yang terbaik. Masuk ke tahap uji *dropout rate*, rata-rata performa yang diberikan relatif sama sehingga setiap nilai *dropout rate* memberikan performa yang sama. Terakhir pada *dataset* non-segmentasi, hasil uji *learning rate* memberikan nilai 0.01 sebagai *learning rate* terbaik, sesuai dengan **Tabel 4.12**. Hal ini menunjukkan bahwa *learning rate* ini memiliki

ukuran *step size* yang tepat untuk model dalam mempelajari citra. Sementara itu, *learning rate* 0.1 memberikan performa terburuk. Hal ini dapat disebabkan karena model gagal mencapai titik konvergensi dikarenakan *step* yang diberikan oleh *learning rate* terlalu besar. Hal yang sama juga dapat dikatakan pada *learning rate* 0.001 yang nilainya terlalu kecil. Hal ini dapat menyebabkan model mempelajari citra dengan lambat. Akibatnya, ketika *epoch* sudah menyentuh angka 30, model belum menyentuh titik konvergensi.

Pada kasus *dataset* tersegmentasi, uji *optimizer* lagi-lagi memberikan Adagrad sebagai *optimizer* dengan performa terbaik. Sesuai dengan **Tabel 4.13**, Adagrad memberikan nilai *f1-score* sebesar 0.9390 disusul dengan RMSProp dengan nilai yang berbeda tipis, yaitu 0.9259. Adagrad menghitung pembaruan *learning rate* dengan akar kuadrat dari *sum of squared gradient* sebelumnya. Sementara itu, RMSProp menggunakan *moving average* dari gradient sebelumnya. Kedua *optimizer* ini sama-sama cocok digunakan untuk permasalahan data yang sifatnya *sparse*. Meskipun proses segmentasi dilakukan, pola-pola yang tidak beraturan tidak hilang karena pola tersebut penting untuk mebedakan citra antar kelas. Masuk ke pembahasan uji *hidden units*, neuron berjumlah 128 tampak memberikan performa terbaik sesuai dengan **Tabel 4.14**. Performa terbaik di *hidden units* 128 menunjukkan bahwa model dapat mempelajari pola-pola pada citra dengan baik di nilai ini. Hal ini masuk akal dengan jenis *dataset* yang digunakan dalam pada uji skenario ini, yaitu *dataset* tersegmentasi. *Dataset* tersegmentasi memiliki pola citra yang lebih sederhana dibandingkan dengan citra non-segmentasi sehingga tidak dibutuhkan jumlah neuron yang banyak untuk mempelajari pola pada citra. Selanjutnya, uji *dropout rate* menunjukkan bahwa nilai 0.5 memberikan performa terbaik sesuai dengan **Tabel 4.15**. Nilai 0.5 pada *dropout rate* berarti terdapat 50% neuron yang sementara waktu dinonaktifkan secara acak. Nilai ini cenderung memberikan nilai yang lebih baik dibandingkan *dropout rate* 0.2 dan 0.3. Hal ini dapat berarti bahwa model mampu *generalize* prediksi pada *dataset* dengan nilai 0.5 untuk *dropout rate*. Pada uji terakhir yaitu uji *learning rate*, model memberikan performa terbaik pada *learning rate* 0.001, sesuai dengan **Tabel 4.16**. *Learning rate* 0.001 memberikan nilai yang lebih baik dibandingkan 0.01. Ini berarti, pada skenario ini, model memerlukan *step* yang lebih kecil untuk memperbarui parameter agar menghasilkan performa yang optimal. Sementara itu, *learning rate* 0.1 memberikan performa yang kurang baik, yaitu 0.7409 untuk *f1-score*. Hal ini menunjukkan bahwa *step size* saat proses *training* terlalu besar sehingga model gagal menyentuh titik konvergensi.

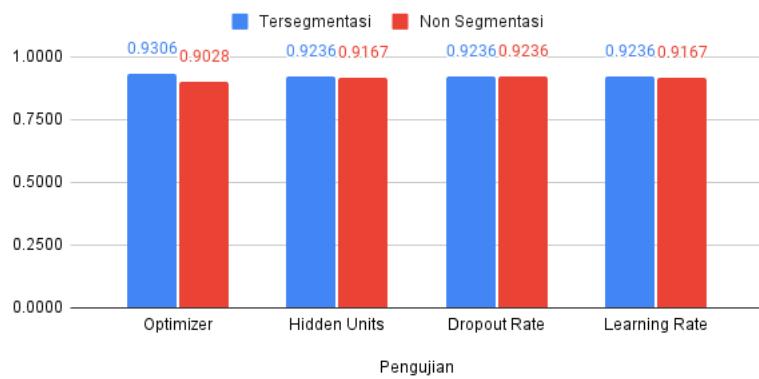
Setelah melakukan uji coba pada kedua *dataset*, didapatkan beberapa bahasan. **Gambar 4.17** menunjukkan perbandingan performa *f1-score* terbaik dari setiap uji *hyperparameter* antara *dataset* tersegmentasi dengan non-segmentasi. Mengacu pada **Gambar 4.17**, *dataset* non-segmentasi unggul dibandingkan dengan *dataset* tersegmentasi dengan perbedaan yang sangat minim. *Dataset* non-segmentasi unggul pada 3 dari 4 uji skenario yang dilakukan. Hal yang serupa juga ditunjukkan pada **Gambar 4.18** yang menunjukkan perbandingan akurasi terbaik dari setiap uji *hyperparameter* di kedua *dataset*. Hasil akurasi menunjukkan bahwa *dataset* tersegmentasi unggul dari 3 dari 4 uji skenario yang dilakukan. Ini menunjukkan bahwa penggunaan *dataset* tersegmentasi tidak memiliki pengaruh yang signifikan pada uji skenario evaluasi *hyperparameter*. Hal yang menarik lainnya yang dapat dilihat adalah performa *f1-score* dan akurasi yang konsisten berada diatas 0.9 pada model MobileNetV3Small. Pada uji *hyperparameter default* dan uji *hyperparameter tuning*, model MobileNetV3Small belum dapat menyentuh angka 0.9 ke atas. Pengaruh *optimizer* dapat menjadi jawaban atas fenomena ini. Pada 2 skenario sebelumnya Adam digunakan sebagai *optimizer*, bukan Adagrad.

Perbandingan F1-Score Terbaik pada Uji Evaluasi
Hyperparameter - Dataset Tersegmentasi vs Non-Segmentasi



Gambar 4.17 Perbandingan *F1-Score* Terbaik pada Uji Evaluasi *Hyperparameter* antara *Dataset* Tersegmentasi vs Non-Segmentasi

Perbandingan Accuracy pada Uji Learning Rate - Dataset Tersegmentasi vs Non-Segmentasi



Gambar 4.18 Perbandingan Akurasi Terbaik pada Uji Evaluasi *Hyperparameter* antara *Dataset* Tersegmentasi vs Non-Segmentasi

4.2.4 Pembahasan Uji Skenario MobileNetV3Small vs MobileNetV3Large

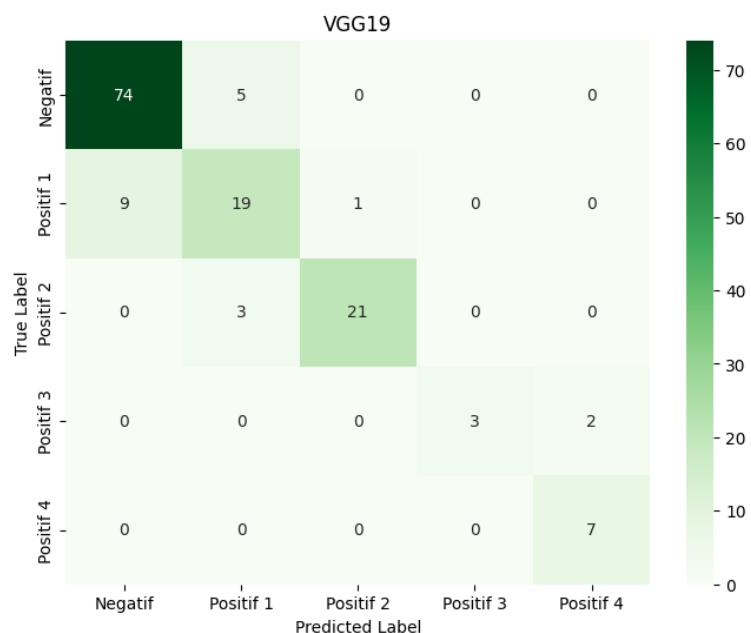
Pengujian ini dilakukan pada *dataset* non-segmentasi dan memberikan hasil yang ditunjukkan pada **Tabel 4.17**. Mengacu pada tabel, model MobileNetV3Small memberikan hasil yang lebih baik pada setiap *metric* evaluasi serta unggul dari sisi efisiensi. Pengujian ini dilakukan dengan *hyperparameter* yang identik sehingga terdapat potensi adanya ketidakcocokan *hyperparameter* yang digunakan pada MobileNetV3Large dan mengakibatkan performa tidak optimal. Namun, perlu dilakukan uji yang lebih komprehensif untuk membuktikan hal ini. **Tabel 4.19** menunjukkan perbandingan arsitektur model MobileNetV3Small dengan MobileNetV3Large (Howard, et al., 2019). Jika mengacu pada **Tabel 4.19**, model MobileNetV3Large memang memiliki jumlah *layer* dan parameter yang lebih banyak daripada MobileNetV3Small sehingga masuk akal jika ukurannya lebih kecil.

Tabel 4.19 Perbandingan Arsitektur MobileNetV3 Small vs Large

Model	Top-1 Accuracy	Parameters	Layers
MobileNetV3Small	0.6740	2.5M	16
MobileNetV3Large	0.7520	5.4M	20

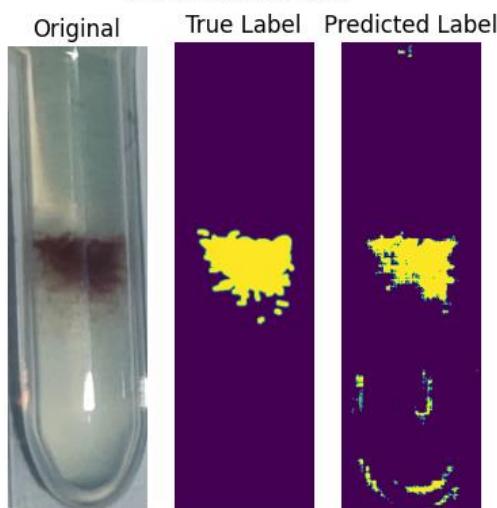
4.2.5 Pembahasan Uji Skenario Model VGG19 pada Dataset Tersegmentasi

Setelah dilakukan pengujian pada model VGG19 dengan *dataset* tersegmentasi, didapatkan beberapa bahasan. **Gambar 4.19** menunjukkan *confusion matrix* model VGG19. Permasalahan yang dihadapi masih sama yaitu kurang baiknya model dalam memprediksi citra dengan kelas Positif 1. Tidak hanya itu, model VGG19 juga gagal memprediksi dengan baik citra kelas Positif 3. Hal ini dapat dilihat pada *confusion matrix*, model memprediksi 3 dari 5 citra (60 % akurat). Jika mengacu pada **Gambar 4.20**, yaitu gambar kesalahan prediksi citra kelas Positif 3. Kesalahan ini jelas merupakan kesalahan model. Citra tersegmentasi yang digunakan sudah menunjukkan karakteristik kelas Positif 3 sehingga kesalahan merupakan kesalahan model. Perlu dilakukan *hyperparameter tuning* untuk mendapatkan hasil yang lebih baik. Penggunaan model VGG19 tidak berdampak signifikan pada performa klasifikasi. Hal ini dapat dilihat pada

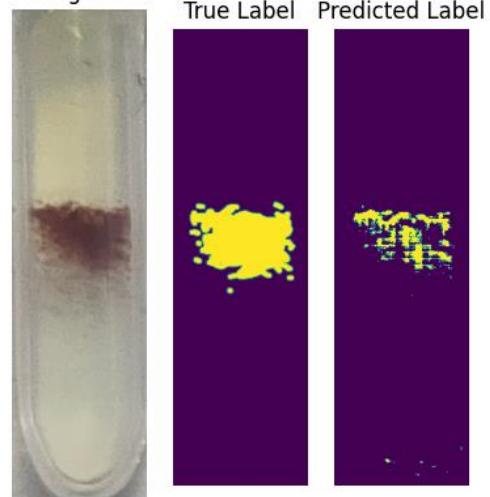


Gambar 4.19 Confusion Matrix Model VGG19 pada Dataset Tersegmentasi

File 667.jpg with class 3 but predicted as 4
IoU Score of 0.5416



File 668.jpg with class 3 but predicted as 4
IoU Score of 0.3090

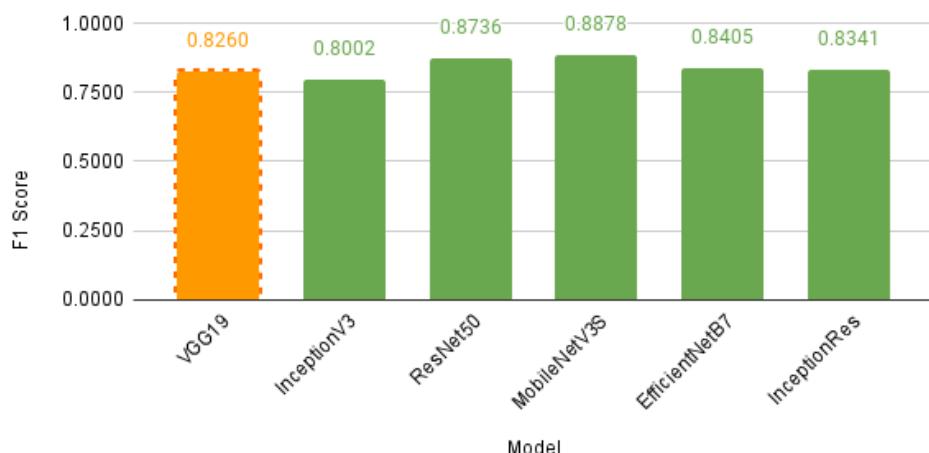


Gambar 4.20 Citra Tersegmentasi Kelas Positif 3 yang Salah Diprediksi Model VGG19

Selain itu, mengacu pada **Gambar 4.20**, nilai *iou score* yang diberikan pada citra 667.jpg dan citra 668.jpg secara berurutan adalah 0.5416 dan 0.3090. Nilai ini termasuk nilai yang rendah sehingga model segmentasi U-Net yang digunakan masih belum mampu menangkap detail relevan pada citra. Meskipun demikian, citra hasil segmentasi yang diberikan masih terlihat sesuai dengan kelasnya.

Gambar 4.21 yang menunjukkan perbandingan nilai *f1-score* antara model VGG19 dengan model lainnya pada uji skenario *hyperparameter default* dengan *dataset tersegmentasi* (*hyperparameter* yang digunakan sama). Hasil yang didapatkan tidak jauh berbeda dengan model-model lainnya, yaitu berkisar di angka 0.8. Jika dihitung rata-rata perbedaan absolut (*mean absolute error*) *f1-score* model VGG19 dengan model lain, didapatkan hasil 0.037. Ini menunjukkan error yang sangat kecil sehingga dapat disimpulkan bahwa model VGG19 tidak memberikan performa yang jauh lebih baik dibandingkan dengan model-model lainnya.

Perbedaan F1 Score VGG19 dengan Pre-Trained Model Lainnya pada Dataset Tersegmentasi



Gambar 4.21 Perbedaan *F1-Score* Model VGG19 terhadap Model Lain pada Uji *Hyperparameter Default Dataset Tersegmentasi*

4.2.6 Pembahasan Performa Kelima *Pre-Trained Model*

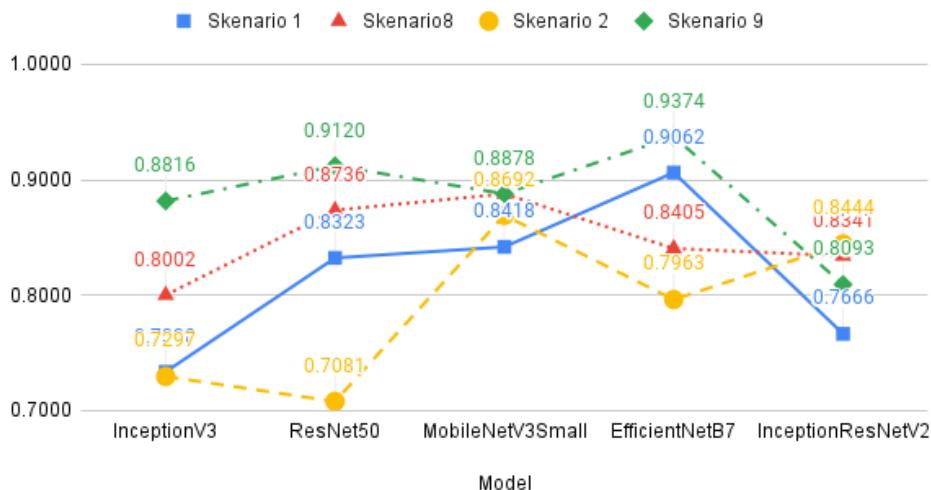
Berdasarkan uji coba skenario pada **subbab 4.1.1** dan **subbab 4.1.2**, model yang memiliki performa paling stabil adalah MobileNetV3Small. Dari sisi *f1-score*, model MobileNetV3Small sangatlah stabil sesuai pada **Gambar 4.22**.

Hal ini juga dapat dilihat pada **Tabel 4.20** yang menunjukkan rata-rata dan standar deviasi *f1-score* bahwa model MobileNetV3Small memiliki rata-rata nilai *f1-score* yang baik, yaitu di angka 0.8717 dan standar deviasi *f1-score* yang paling rendah di antara semua model, yaitu di angka 0.0217. Hal ini berarti bahwa pada uji skenario 1, 2, 8 dan 9, model MobileNetV3Small memiliki performa yang sangat stabil. Hal ini juga bisa dilihat dari sisi akurasi pada **Gambar 4.23** dan **Tabel 4.21**. MobileNetV3Small memiliki standar deviasi yang paling rendah, di angka 0.0165 dan rata-rata akurasi di angka 0.8993. Nilai yang rendah ini kembali menekankan bahwa model MobileNetV3Small memiliki performa yang sangat stabil selama uji skenario 1, 2, 8 dan 9.

Dari skenario yang disebutkan, model dengan performa terbaik dipegang oleh EfficientNetB7 dengan performa puncak di skenario 9. Di skenario ini, model EfficientNetB7 berhasil mendapatkan nilai *f1-score* di angka 0.9374 dan nilai akurasi di angka 0.9236. Hal

yang cukup menarik adalah peforma model EfficientNetB7 terburuk justru berada pada skenario *Hyperparameter Tuning* Non-Segmentasi dengan nilai *f1-score* 0.7963. Ada beberapa faktor yang dapat menyebab hal ini, salah satunya eksplorasi kombinasi parameter yang dilakukan sebanyak 20 *trials* sehingga tidak seluruh kombinasi parameter diuji coba. Hasil klasifikasi model terbaik EfficientNetB7 dapat dilihat pada **lampiran L17 hingga L21**.

Perbandingan F1-Score Setiap Model pada Skenario 1, 2, 8, 9



Gambar 4.22 Perbandingan *F1-Score* Setiap Model pada Skenario 1, 2, 8, 9

Tabel 4.20 Mean dan Standar Deviasi *F1-Score* Setiap Model pada Skenario 1, 2, 8, 9

Model	Mean	STD
InceptionV3	0.7863	0.0713
ResNet50	0.8315	0.0885
MobileNetV3Small	0.8717	0.0217
EfficientNetB7	0.8701	0.0637
InceptionResNetV2	0.8136	0.0346

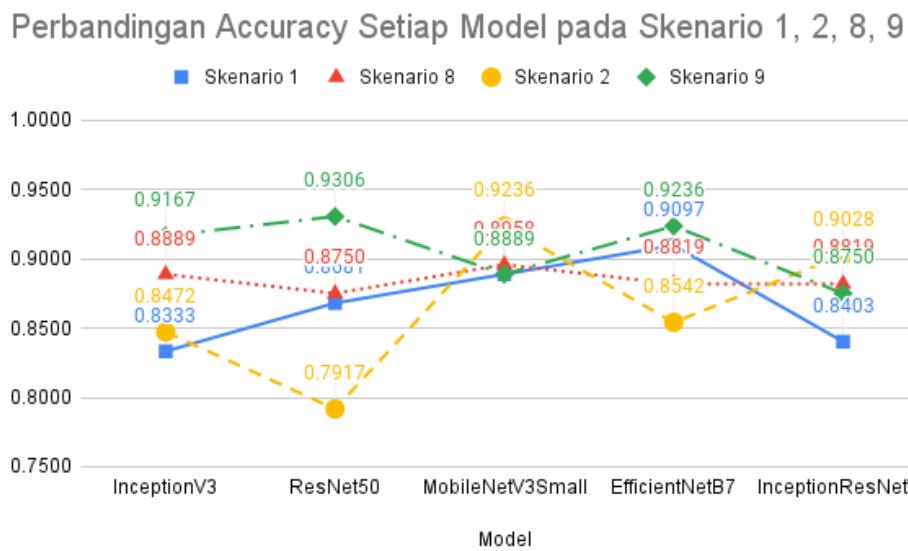
Tabel 4.21 Mean dan Standar Deviasi Akurasi Setiap Model pada Skenario 1, 2, 8, 9

Model	Mean	STD
InceptionV3	0.8715	0.0382
ResNet50	0.8663	0.0571
MobileNetV3Small	0.8993	0.0165
EfficientNetB7	0.8924	0.0308
InceptionResNetV2	0.8750	0.0260

Hal menarik lainnya dapat dilihat dari model InceptionV3 dan ResNet50. Model-model ini mengalami peningkatan akurasi dan *f1-score* pada skenario 1, 8, 2, 9 secara berurutan. Kedua model ini mengalami puncak performa pada skenario *hyperparameter tuning* dengan *dataset* tersegmentasi. Hal ini menunjukkan proses *hyperparameter tuning* berhasil mencapai tujuannya, yaitu meningkatkan performa model. Selain itu, *dataset* tersegmentasi selalu memberikan performa yang lebih baik dibandingkan dengan *dataset* non-segmentasi. Ini menunjukkan bahwa *data tersegmentasi* cocok digunakan pada kedua model ini. Sementara itu,

dataset tersegmentasi tidak bisa dikatakan berdampak dalam peningkatan performa pada model InceptionResNetV2. Hal ini dibuktikan pada **Gambar 4.22** yang menunjukkan performa terbaik model InceptionResNetV2 berada di skenario 2, dimana *hyperparameter tuning* dilakukan dan model di-train dengan data non-segmentasi.

Beberapa uji skenario melibatkan penggunaan *dataset* tersegmentasi dan non-segmentasi. Jika diperhatikan pada **Gambar 4.22** dan **Gambar 4.23**, puncak performa model didapatkan pada skenario 9, dimana dalam skenario tersebut, *dataset* tersegmentasi digunakan. Pada **Gambar 4.22**, 4 dari 5 *pre-trained* model memiliki performa terbaik di skenario 9. Sementara itu, pada **Gambar 4.23**, 3 dari 5 *pre-trained* model memiliki performa terbaik di skenario 9. Hal ini menunjukkan bahwa, secara umum, pada penilitian klasifikasi uji silang serasi ini, *dataset* tersegmentasi memberikan performa yang lebih baik dibandingkan *dataset* non-segmentasi.



Gambar 4.23 Perbandingan Akurasi Setiap Model pada Skenario 1, 2, 8, 9

4.2.7 Pembahasan *Dataset* Tersegmentasi dalam Peningkatan Performa Kelima *Pre-Trained* Model

Pembahasan selanjutnya yang dilakukan adalah apakah penggunaan *dataset* tersegmentasi memberikan efek yang signifikan terhadap performa *pre-trained* model. **Tabel 4.22** menunjukkan selisih *f1-score* pada performa terbaik dari kelima model di kedua *dataset*. Berdasarkan tabel, model yang mengalami peningkatan yang signifikan hanyalah InceptionV3, yaitu mengalami 20% peningkatan saat menggunakan *dataset* tersegmentasi. Sementara itu, beberapa model lainnya mengalami peningkatan yang tidak signifikan seperti MobileNetV3Small dan EfficientNetB7 yang hanya mengalami peningkatan 2.14 % dan 3.44 %. Hal yang menarik adalah model InceptionResNetV2 yang justru mengalami penurunan sebesar -1.22%, tetapi nilai ini tidak terlalu signifikan.

Berdasarkan **Tabel 4.22**, rerata persentase yang didapat adalah 6.82%. Berdasarkan nilai ini, model InceptionV3 dan ResNet50 mengalami peningkatan performa yang signifikan karena laju peningkatan *f1-score*-nya lebih besar dari laju peningkatan *f1-score* rata-rata. InceptionV3 memberikan nilai laju peningkatan yang tinggi, yaitu di angka 20.14% dan ResNet50 memberikan laju peningkatan di angka 9.58%. Oleh karena itu, *dataset* tersegmentasi memberikan peningkatan performa yang signifikan pada kedua model ini.

Untuk mempermudah penggunaan proses klasifikasi uji silang serasi, dibuatlah suatu sistem berbasis *website* yang penggunaanya ditunjukkan pada **lampiran L.22**. Sistem dibuat dalam bentuk *website* untuk mempermudah aksesibilitas.

Tabel 4.22 Nilai Selisih *F1-Score* pada Performa Terbaik Kedua *Dataset*

Model	Tersegmentasi	Non-Segmentasi	Selisih	Percentase (%)
InceptionV3	0.8816	0.7338	0.1478	20.14
ResNet50	0.9120	0.8323	0.0797	9.58
MobileNetV3Small	0.8878	0.8692	0.0186	2.14
EfficientNetB7	0.9374	0.9062	0.0312	3.44
InceptionResNetV2	0.8341	0.8444	-0.0103	-1.22
Rata-Rata			0.0534	6.82

[Halaman ini sengaja dikosongkan]

BAB 5 KESIMPULAN DAN SARAN

5.1 Kesimpulan

Setelah melalui tahap perancangan sistem, implementasi metode, dan uji coba pada penelitian ini, didapatkan kesimpulan sebagai berikut.

1. Proses preparasi *dataset* citra uji silang serasi pada penelitian ini dimulai dari proses *cropping* citra gel yang terdiri dari beberapa tabung menjadi citra yang menampilkan satu tabung. Selanjutnya dilakukan proses pelabelan dalam bentuk *file json* sebagai bahan klasifikasi dan proses anotasi *ground truth* untuk tahap segmentasi. Terakhir dilanjutkan dengan proses augmentasi sebelum data masuk ke proses *training*.
2. Proses klasifikasi citra uji silang serasi dengan metode *deep learning* melibatkan berbagai macam tahap, dimulai dari proses pengolahan citra, proses segmentasi, pembagian dataset, penentuan model, proses *training* hingga evaluasi model. Secara umum, proses klasifikasi citra uji silang serasi dengan metode *deep learning* memberikan hasil yang baik dengan *f1-score* tertinggi di angka 0.9374 dan akurasi di angka 0.9236 pada *dataset* tersegmentasi dan *f1-score* di angka 0.9062 dan akurasi di angka 0.9097 pada *dataset* non-segmentasi.
3. Performa kelima *pre-trained* model pada klasifikasi uji silang serasi memiliki performa yang beragam. Performa yang beragam ini dapat dilihat dari beberapa pengujian skenario yang dilakukan. Seluruh model mengalami puncak performa pada *dataset* tersegmentasi kecuali InceptionResNetV2 yang mengalami penurunan performa meski tidak signifikan. Selain itu, pemilihan *hyperparameter* berpengaruh pada performa model. Model yang memiliki performa terbaik adalah model EfficientNetB7 dengan nilai *f1-score* di angka 0.9374. Mayoritas model memiliki performa *f1-score* lebih besar dan mendekati angka 0.9 sehingga performa kelima *pre-trained* model cenderung baik.

5.2 Saran

Dalam penelitian yang berjudul “Klasifikasi Darah pada Citra Uji Silang Serasi Menggunakan Pendekatan *Deep Learning*”, penulis memiliki beberapa saran untuk penelitian selanjutnya, yaitu:

1. Penambahan pada citra pada penelitian selanjutnya. Hal ini bertujuan untuk mengurangi kemungkinan *overfitting* sehingga menciptakan model yang lebih *robust*.
2. Proses *hyperparameter tuning* dengan KerasTuner pada *max trial* seluruh kombinasi. Penelitian ini menggunakan *max trial* senilai 20. Ini berarti terdapat 20 kali uji coba *hyperparameter* pada kombinasi kumpulan *hyperparamter* yang tersedia. Kemungkinan kombinasi *hyperparameter* yang tersedia 81 buah sehingga belum suam kombinasi *hyperparameter* dieksplorasi.
3. Pada uji coba perbandingan MobileNetV3Large dan MobileNetV3Small, tidak digunakan *dataset* tersegmentasi. Hal ini dilakukan untuk melihat performa kedua model pada *dataset* tersegmentasi. Selain itu, perlu juga dilakukan *hyperparameter tuning* pada kedua model. Hal ini bertujuan untuk mendapatkan konfigurasi terbaik pada kedua model. Dengan demikian, hasil yang diberikan kedua model adalah hasil yang terbaik.
4. Eksplorasi efek *hyperparameter* pada **subbab 4.1.3** akan lebih baik jika diinspeksi *learning curve*. *Learning curve* dapat berupa grafik akurasi dan grafik *loss*. Hal ini dapat menunjukkan pengaruh *hyperparameter* pada proses belajar model dalam setiap *epoch*.

5. Dilakukannya *hyperparameter tuning* pada model VGG19 pada uji skenario di **subbab 4.1.5.** Hal ini dapat menunjukkan performa terbaik dari VGG19. Setelah itu, dapat dilakukan perbandingan performa VGG19 dengan model-model lainnya.

DAFTAR PUSTAKA

- (2019, January 12). Retrieved from sciencedirect:
<https://www.sciencedirect.com/science/article/pii/S2215098618321050>
- A, A., S.Bhuvaneswari, Gowri, B. S., Narayanan, S., N, S., & M, S. (2023). MRI-based Brain Tumour Classification Using EfficientNetB7 model with transfer learning. 1.
- Amer, M. (2022, June 07). Classification Evaluation Metrics: Accuracy, Precision, Recall, and F1 Visually Explained. Retrieved from cohoreblog:
<https://txt.cohere.com/classification-eval-metrics/>
- ASHRAF, R., HABIB, M. A., & AKRAM, M. (2020). Deep Convolution Neural Network for Big Data Medical Image Classification. IEEE.
- Ault, R. (2020). Optimization Study of an Image Classification Deep Neural Network. Honors Projects.
- Azzahra, T. S., Cerelia, J. J., Nugraha, F. A., & Pravitasari, A. A. (2023). MRI-based Brain Tumor Classification with Inception Resnet V2. amcs, 1.
- Baheti, B. (2021, June). researchgate. Retrieved from researchgate:
https://www.researchgate.net/figure/Architecture-of-EfficientNetB7-as-encoder-which-is-build-up-using-MBconv-blocks-The_fig2_354297151
- Bhandari, A. (2023, May 19). Analytics Vidhya. Retrieved from Analytics Vidhya:
<https://www.analyticsvidhya.com/blog/2020/08/image-augmentation-on-the-fly-using-keras-imagedatagenerator/>
- Brownlee, J. (2021, January 13). Machine Learning Mastery. Retrieved from Machine Learning Mastery: <https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/>
- Chen, H., Wang, Y., Guo, T., Xu, C., Deng, Y., Liu, Z., . . . Gao, W. (2021). Pre-Trained Image Processing Transformer.
- Chen, Z. (2023). Medical Image Segmentation Based on U-Net. iopscience, 2.
- Developers Google. (2022, July 18). Retrieved June 20, 2023, from
<https://developers.google.com/machine-learning/practica/image-classification#:~:text=Image%20classification%20is%20a%20supervised,the%20input%20to%20the%20model.>
- Elhamraouri, Z. (2020, May 17). Medium. Retrieved from Medium:
<https://medium.com/@zahraelhamraoui1997/inceptionresnetv2-simple-introduction-9a2000edc6b6>
- Gaur, L., Bhatia, U., Jhanjhi, N. Z., Muhammad, G., & Masud, M. (2021). Medical image-based detection of COVID-19 using Deep Convolution Neural Networks. Multimedia Systems.
- Hasegawa, H. (2023). Brain Tumor Detection Based on Deep Learning Approaches and Magnetic Resonance Imaging. PudMed.
- Hemalatha, B., Karthik, B., Reddy, C. K., & Latha, A. (2022). Deep learning approach for segmentation and classification of blood cells using enhanced CNN. Measurement Sensor.
- Hindawi. (2023, June 27). Retrieved from Hindawi:
<https://www.hindawi.com/journals/misy/2020/7602384/>
- Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., . . . Adam, H. (2019). Searching for MobileNetV3. arxiv, 1-2.

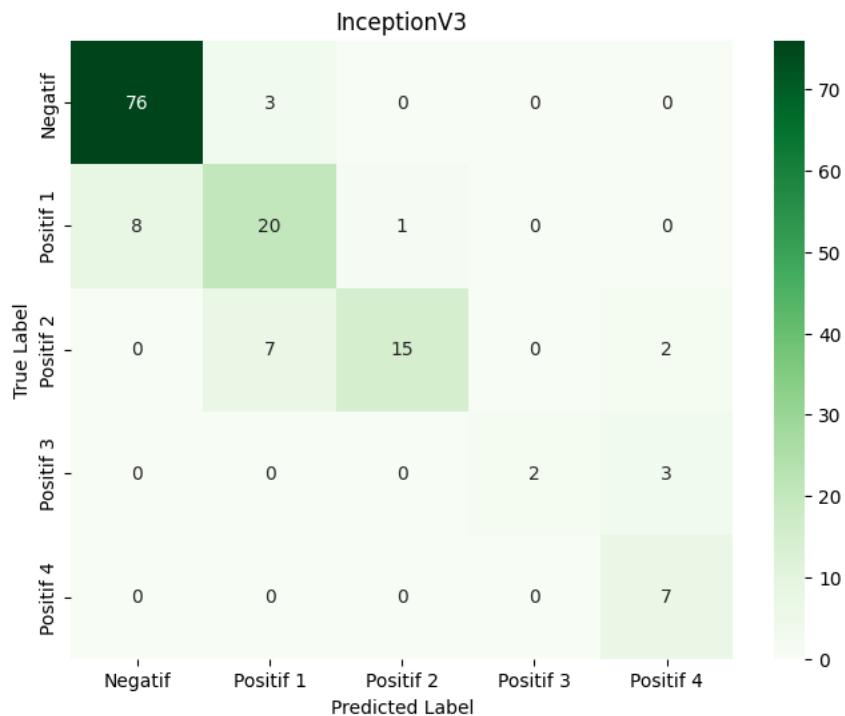
- Kaushik, A. (2023, 6 20). Opengenus. Retrieved June 20, 2023, from <https://iq.opengenus.org/resnet50-architecture#:~:text=ResNet50%20is%20a%20variant%20of,explored%20ResNet50%20architecture%20in%20depth>.
- Keras ResNet50. (2023, June 27). Retrieved from educba: <https://www.educba.com/keras-resnet50/>
- KHASHMAN, A., & AL-ZGOUL, E. (2010). Image Segmentation of Blood Cells in Leukemia Patients. researchgate.
- Kovačevića, B. S. (2021). Classification Model Evaluation Metrics . IJACSA.
- KRAWCZYK, Z., & STARZYŃSKI, J. (2021). Segmentation of bone structures with the use of deep learning techniques. PAN.
- Kulkarni, A., Chong, D., & Batarseh, F. A. (2020). Foundations of data imbalance and solutions for a data democracy. sciencedirect. Retrieved from sciencedirect: <https://www.sciencedirect.com/topics/engineering/confusion-matrix#:~:text=A%20confusion%20matrix%20is%20a,performance%20of%20a%20classification%20algorithm>
- Li, E. Y. (2020, August 01). 10 Papers You Should Read to Understand Image Classification in the Deep Learning Era. Retrieved from towardsdatascience: <https://towardsdatascience.com/10-papers-you-should-read-to-understand-image-classification-in-the-deep-learning-era-4b9d792f45a7>
- Lin, D. (2018, May). researchgate. Retrieved from researchgate: https://www.researchgate.net/figure/The-basic-architecture-of-Inception-Resnet-v2_fig3_324961229
- Liu, X., Deng, Z., & Yang, Y. (2018). Recent Progress in Semantic Image Segmentation. arxiv, 1.
- Mabrouk, A., Dahou, A., Elaziz, M. A., Redondo, R. P., & Kayed, M. (2022). Medical Image Classification Using Transfer Learning and Chaos Game Optimization on the Internet of Medical Things. Hindawi.
- Minaee, S., Boykov, Y., Porikli, F., Plaza, A., Kehtarnavaz, N., & Terzopoulos, D. (2022). Image Segmentation Using Deep Learning: A Survey.
- OpenGenus. (2023, 6 20). Retrieved June 20, 2023, from <https://iq.opengenus.org/inception-v3-model-architecture/>
- Pan1, Y., Liu, J., Cai, Y., Yang, X., & Zhang, Z. (2023). Fundus image classification using Inception V3 and ResNet-50 for the early diagnostics of fundus diseases. Frontiers.
- Paperswithcode. (2023, June 27). Retrieved from paperswithcode: <https://paperswithcode.com/method/inception-v3>
- Patil, A., & Birajdar, G. (2020). White Blood Cells Image Classification Using Deep Learning with Canonical Correlation Analysis.
- Rahman, M. (2023, March 7). Medium. Retrieved from Medium: <https://rmoklesur.medium.com/what-you-need-to-know-about-sparse-categorical-cross-entropy-9f07497e3a6f#:~:text=Sparse%20Categorical%20Cross%20Entropy%20is,used%20for%20binary%20classification%20problems>
- Rehman, A., Abbas, N., Saba, T., Rahman, S. I., Mehmood, Z., & Kolivand, H. (2018). Classification of Acute Lymphoblastic Leukemia Using Deep Learning.
- Ronneberger, O., Fischer, P., & Brox, T. (2015). Segmentation, U-Net: Convolutional Networks for Biomedical Image. arxiv.

- Sarwinda, D., Paradisa, R. H., Bustamam, A., & Anggia, P. (2021). Deep Learning in Image Classification using Residual Network (ResNet) Variants for Detection of Colorectal Cancer. *Science Direct*, 423-431.
- Section IO. (2022, April 7). Retrieved from Section: <https://www.section.io/engineering-education/building-a-multiclass-image-classifier-using-mobilenet-v2-and-tensorflow/>
- serej, a. d. (2022, December 23). ResNet-50. Retrieved from medium: <https://medium.com/@arashserej/resnet-50-83b3ff33be7d>
- Shorten, C., & Khoshgoftaar, T. (2019). A survey on Image Data Augmentation for Deep Learning. *J Big Data*.
- Soni, B. (2023, August 04). Understanding the AdaGrad Optimization Algorithm: An Adaptive Learning Rate Approach. Retrieved from Medium: https://medium.com/@brijesh_soni/understanding-the-adagrad-optimization-algorithm-an-adaptive-learning-rate-approach-9dfaee2077bb
- Szegedy, C., Ioffe, S., & Vanhoucke, V. (2016). Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. arxiv.
- Tensorflow. (2023, December 07). tensorflow.org. Retrieved from tensorflow.org: https://www.tensorflow.org/tutorials/images/transfer_learning
- Tensorflow. (2023, December 07). Transfer learning and fine-tuning. Retrieved from Tensorflow: https://www.tensorflow.org/tutorials/images/transfer_learning#:~:text=A%20pre%2Dt,rained%20model%20is,model%20to%20a%20given%20task
- Thakur, A. (2023, July 9). Wandb.ai. Retrieved from Wandb.ai: <https://wandb.ai/ayush-thakur/dl-question-bank/reports/What-s-the-Optimal-Batch-Size-to-Train-a-Neural-Network---VmldzoyMDkyNDU>
- Tiwari, P., Qian, J., Li, Q., Wang, B., Gupta, D., Khanna, A., & Albuquerque, V. H. (2018). Detection of Subtype Blood Cells Using Deep Learning.
- Tuzkaya, G., Sennaroglu, B., Kalender, Z. T., & Mutlu, M. (2019). Hospital Service Quality Evaluation with IVIF-promethee and a case study. *Socio-Economic Planning Sciences*, 68.
- Wang, L. (2022). Deep Learning Techniques to Diagnose Lung Cancer. PubMed.
- WEI, Y., Zhang, Y., Huang, J., & Yang, Q. (2018). Transfer Learning via Learning to Transfer. *PMLR*, 1.
- wisdomml. (2023, June 27). EfficientNet and its Performance Comparison with Other Transfer Learning Networks. Retrieved from Wisdom ML: <https://wisdomml.in/efficientnet-and-its-performance-comparison-with-other-transfer-learning-networks/v>
- Yadav, S. S., & Jadhav, S. M. (2019). Deep convolutional neural network based medical image classification for disease diagnosis. *Big Data*.
- Yang, J., & Kwon, Y. (2023). Novel CNN-Based Approach for Reading Urban Form Data in 2D Images: An Application for Predicting Restaurant Location in Seoul, Korea. MDPI.
- Yu, H., Pang, R., Vasudevan, V., Aggarwal, A., Zoph, B., Du, X., . . . team., G. B. (2023, June 20). Ai Google Blog. Retrieved June 20, 2023, from <https://ai.googleblog.com/2019/05/efficientnet-improving-accuracy-and.html>
- Zehtab-Salmasi, A., Derakhshi, A. R., & Khasmaki, N. N. (2021, April). researchgate. Retrieved from researchgate: https://www.researchgate.net/figure/Architecture-of-the-proposed-Inception-based-method-Model-2_fig4_350579351
- Zhang, J., Xie, Y., Wu, Q., & Xia, Y. (2019). Medical image classification using synergic deep learning. Elsevier.

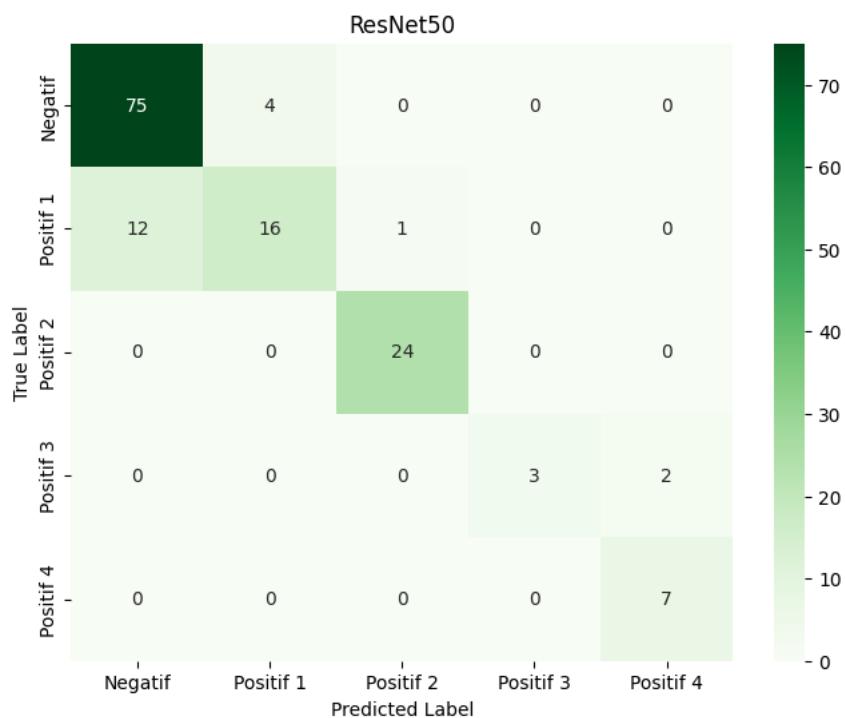
[Halaman ini sengaja dikosongkan]

LAMPIRAN

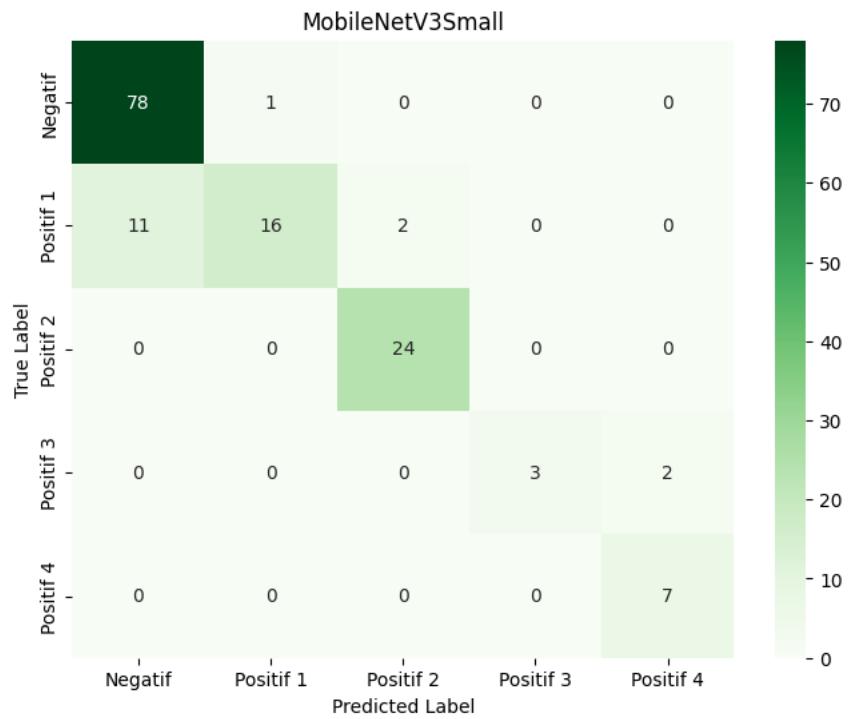
L1. Confusion Matrix InceptionV3 Skenario Hyperparameter Default Dataset Non-Segmentasi



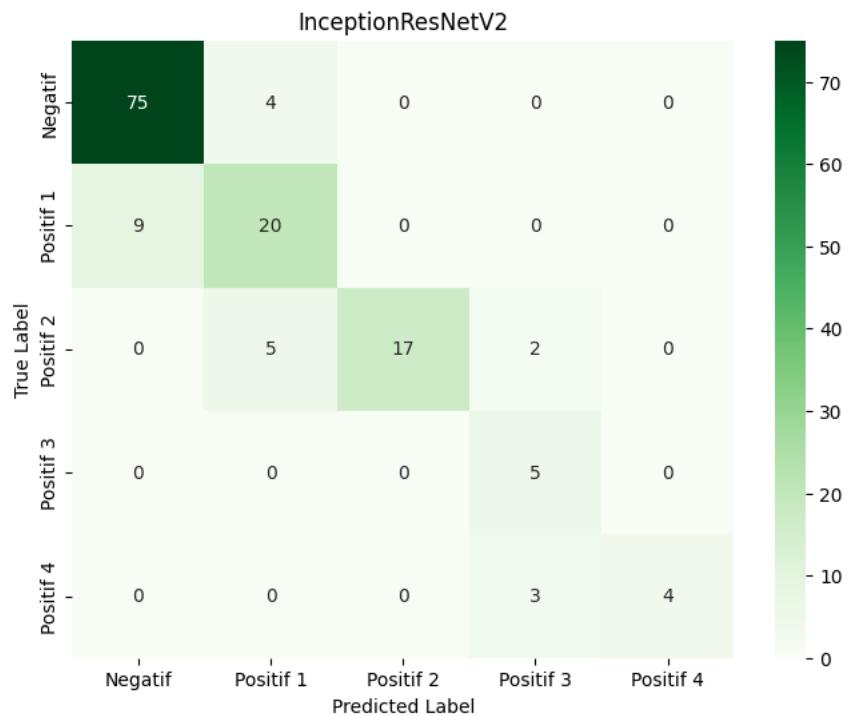
L2. Confusion Matrix ResNet50 Skenario Hyperparameter Default Dataset Non-Segmentasi



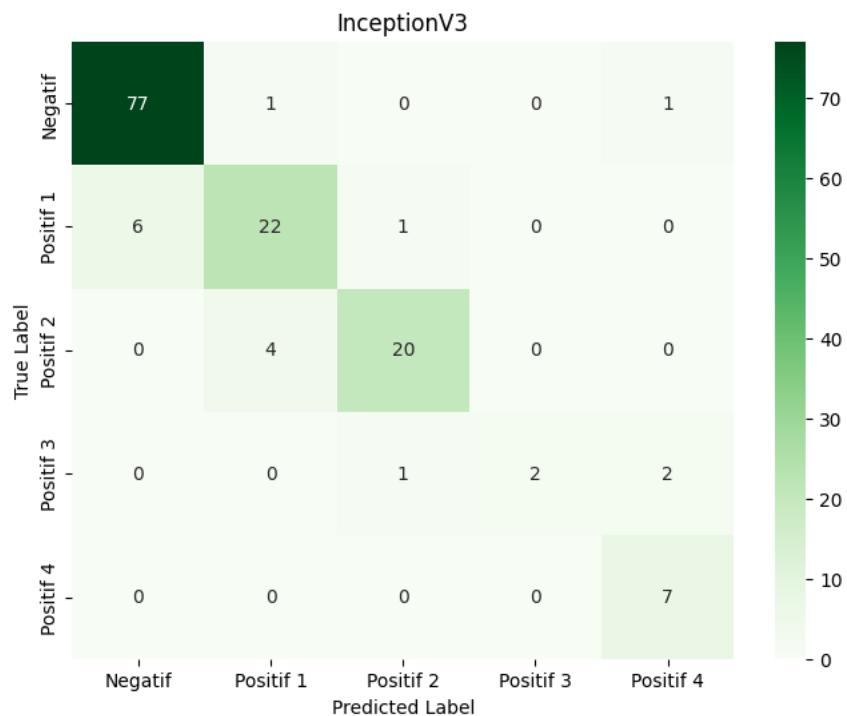
L3. Confusion Matrix MobileNetV3Small Skenario Hyperparameter Default Dataset Non-Segmentasi



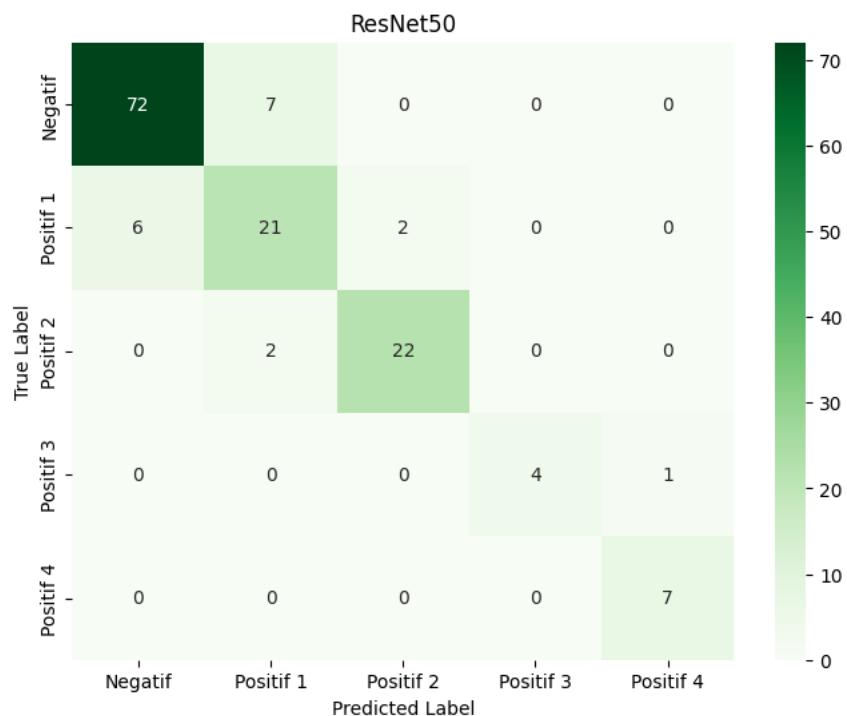
L4. Confusion Matrix InceptionResNetV2 Skenario Hyperparameter Default Dataset Non-Segmentasi



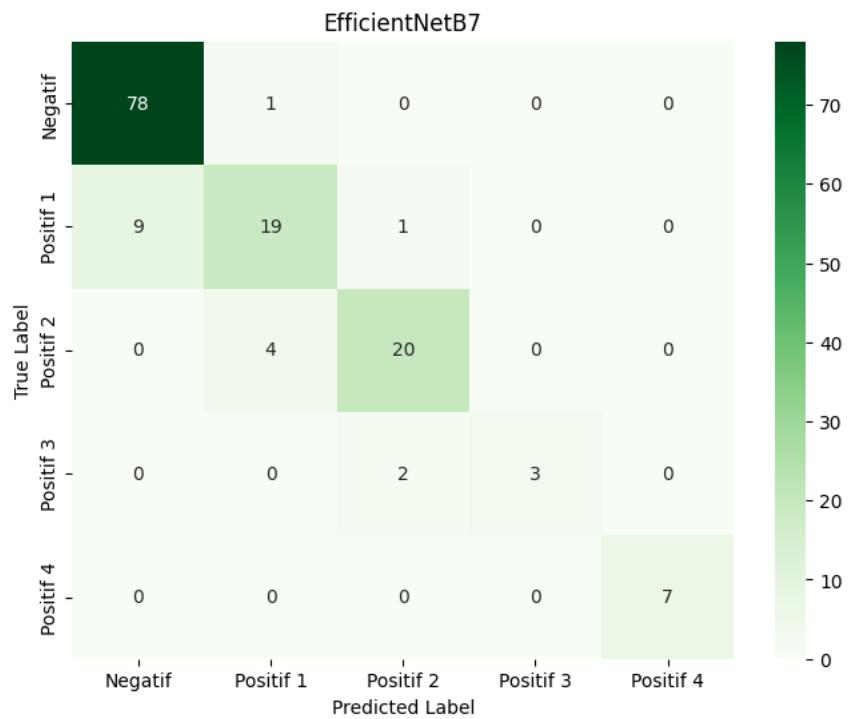
L5. Confusion Matrix InceptionV3 Skenario Hyperparameter Default Dataset Tersegmentasi



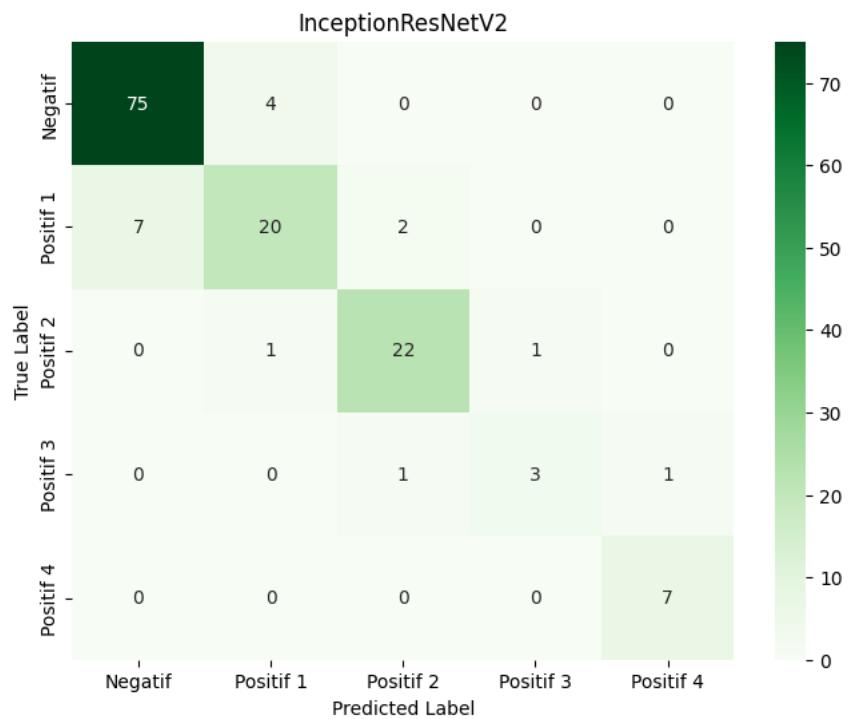
L6. Confusion Matrix ResNet50 Skenario Hyperparameter Default Dataset Tersegmentasi



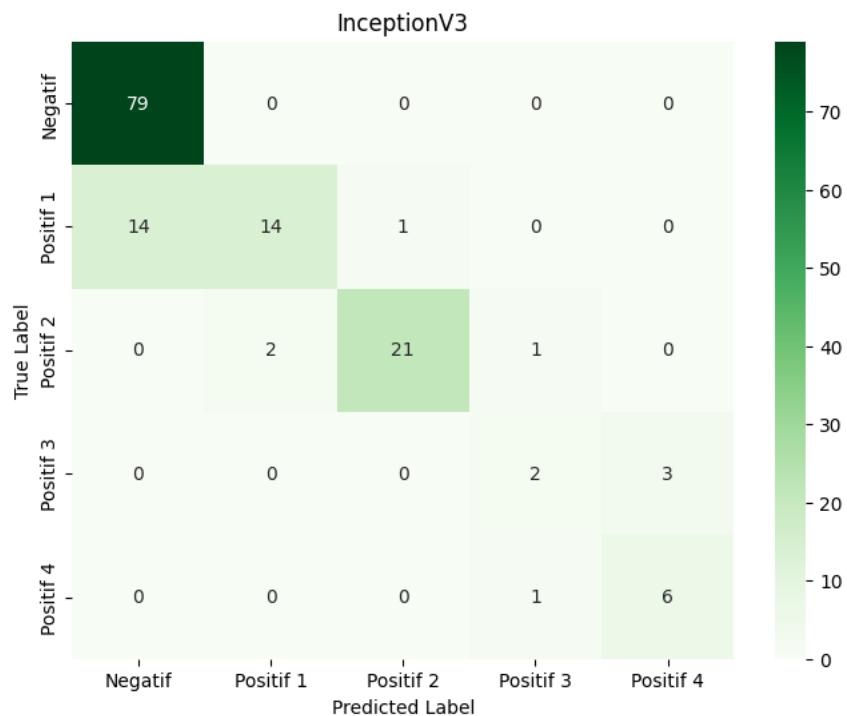
L7. Confusion Matrix EfficientNetB7 Skenario Hyperparameter Default Dataset Tersegmentasi



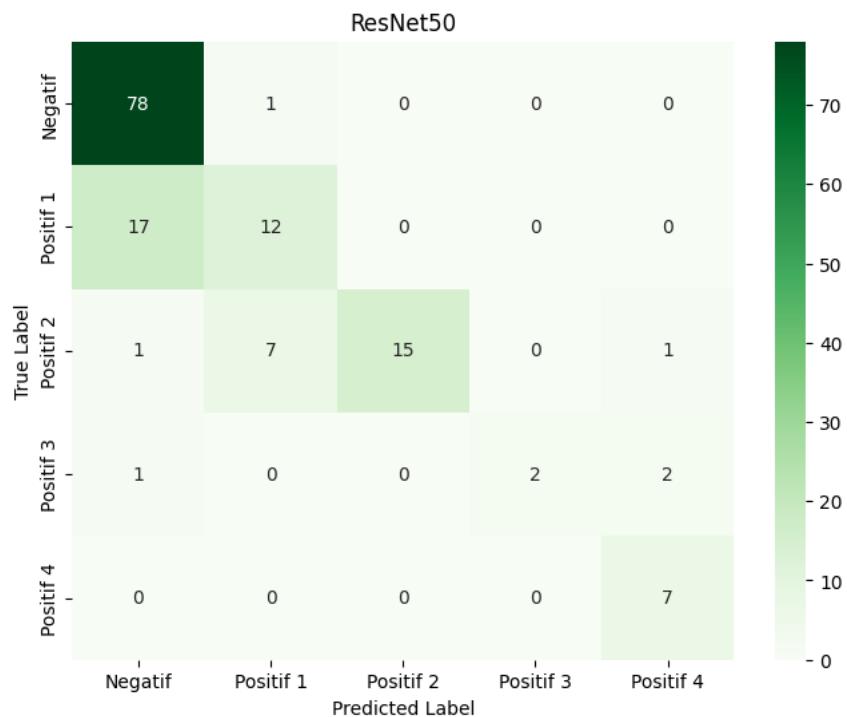
L8. Confusion Matrix InceptionResNetV2 Skenario Hyperparameter Default Dataset Tersegmentasi



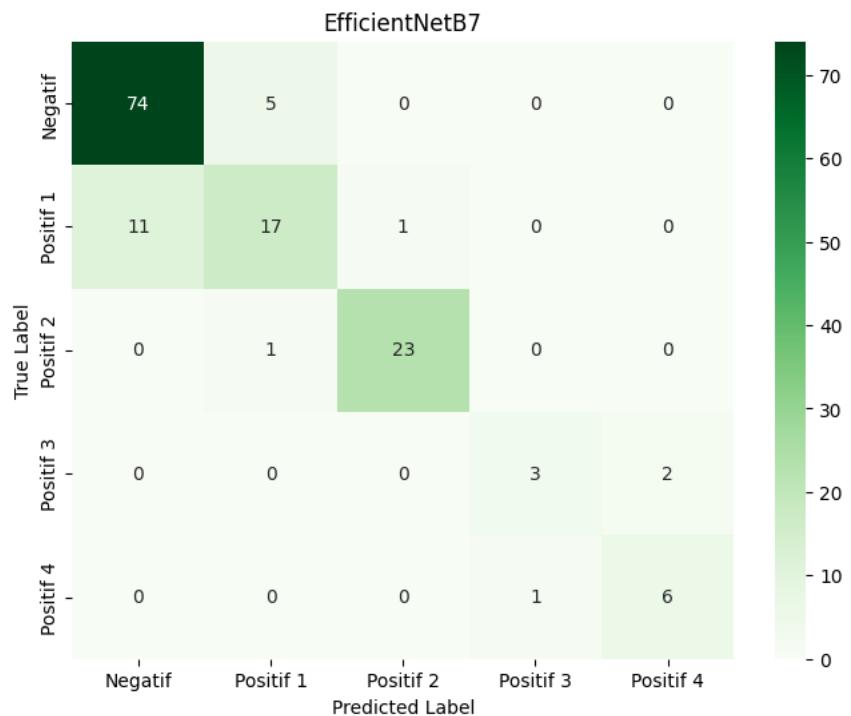
L9. Confusion Matrix InceptionV3 Skenario Hyperparameter Tuning Dataset Non-Segmentasi



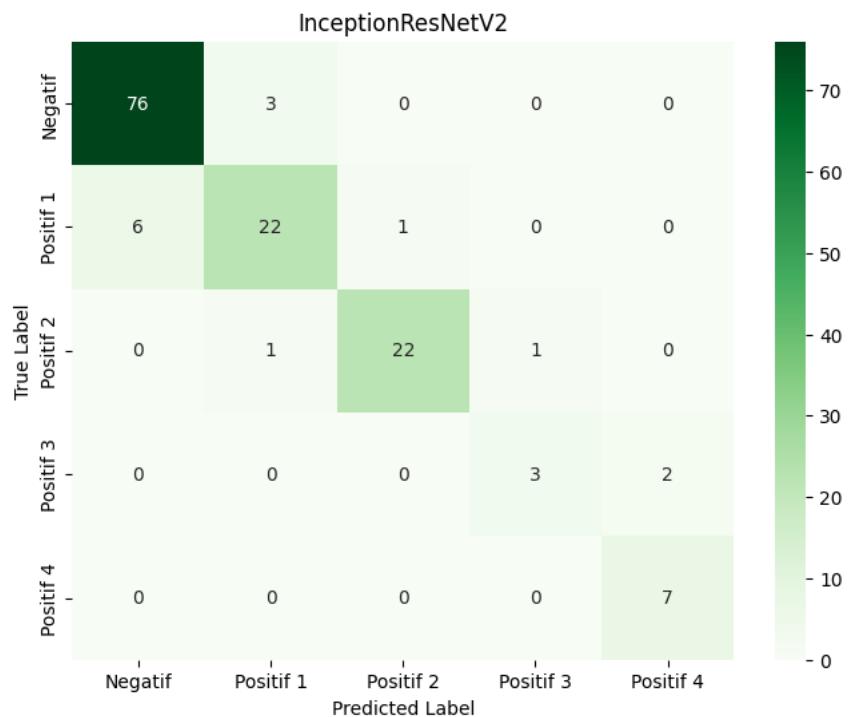
L10. Confusion Matrix ResNet50 Skenario Hyperparameter Tuning Dataset Non-Segmentasi



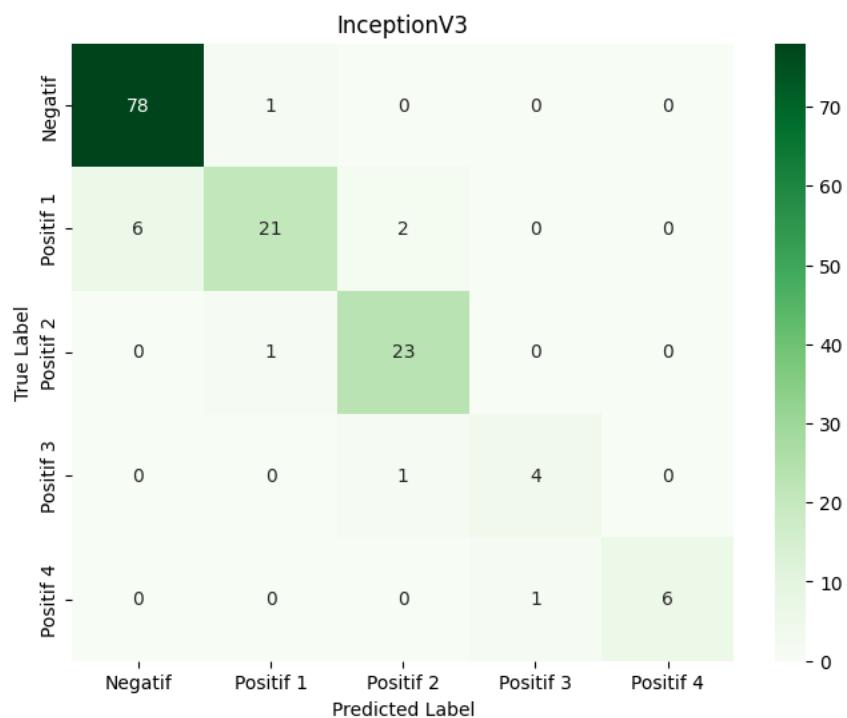
L11. Confusion Matrix EfficientNetB7 Skenario Hyperparameter Tuning Dataset Non-Segmentasi



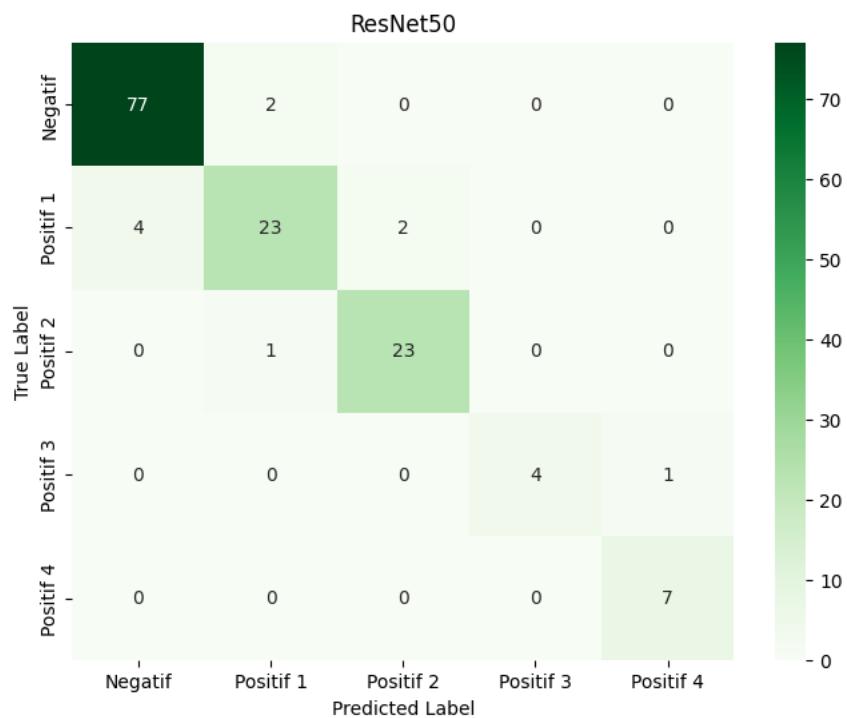
L12. Confusion Matrix InceptionResNetV2 Skenario Hyperparameter Tuning Dataset Non-Segmentasi



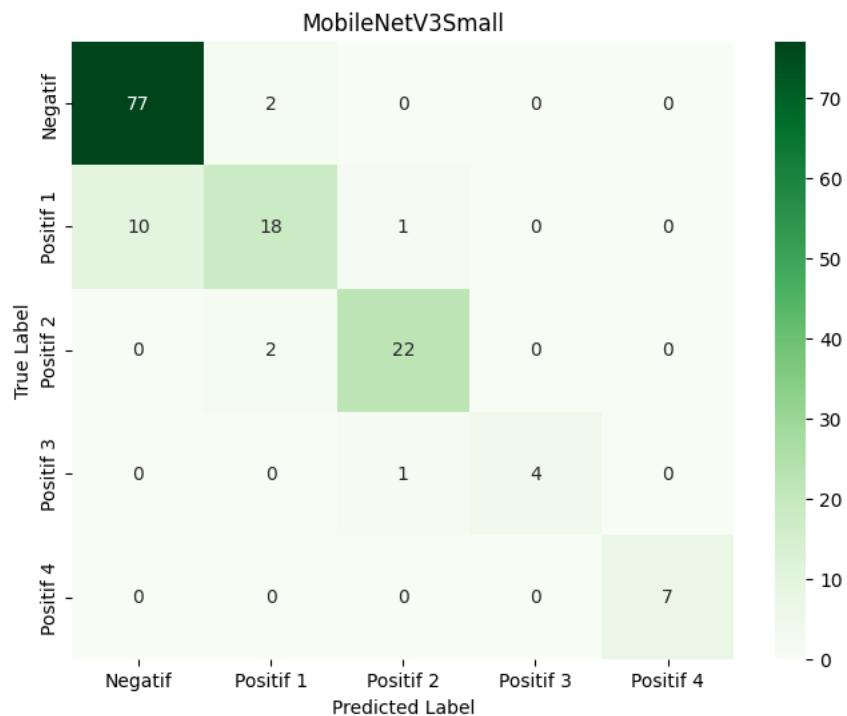
L13. Confusion Matrix InceptionV3 Skenario Hyperparameter Tuning Dataset Tersegmentasi



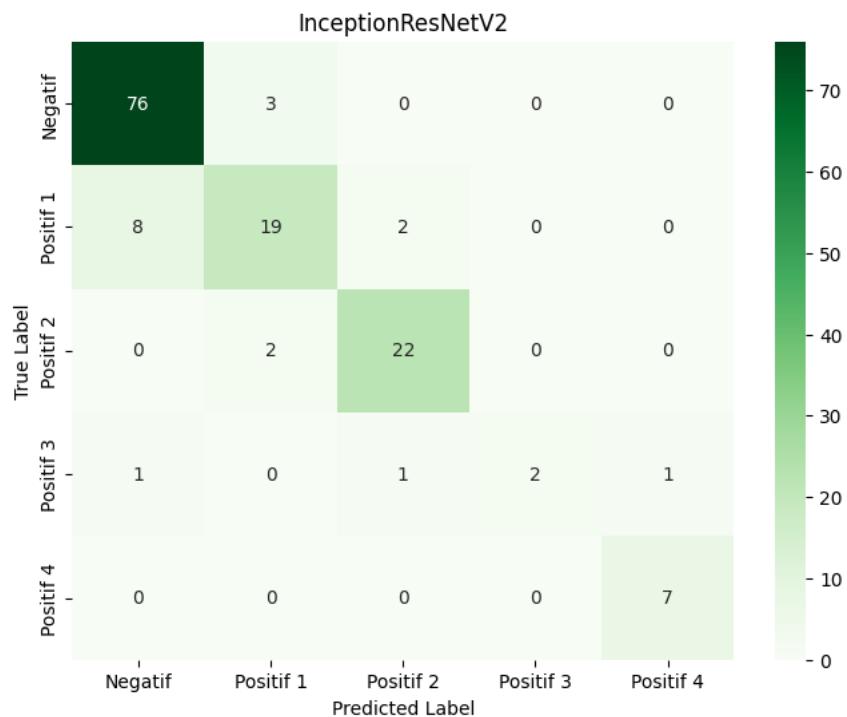
L14. Confusion Matrix ResNet50 Skenario Hyperparameter Tuning Dataset Tersegmentasi



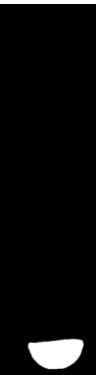
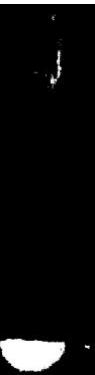
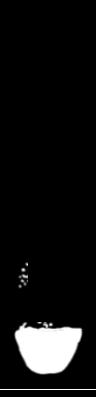
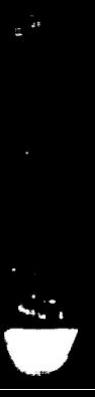
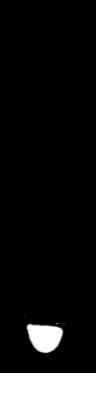
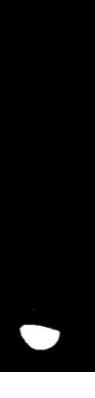
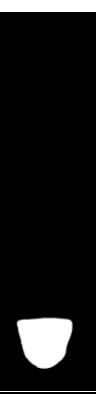
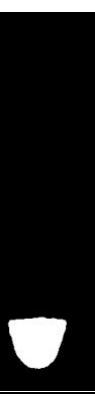
L15. Confusion Matrix MobileNetV3Small Skenario Hyperparameter Tuning Dataset Tersegmentasi



L16. Confusion Matrix InceptionResNetV2 Skenario Hyperparameter Tuning Dataset Tersegmentasi



L17. Hasil Prediksi EfficientNetB7 pada Citra Kelas Negatif

<i>Original</i>	<i>Ground Truth</i>	<i>Segmented Result</i>	<i>Actual Class</i>	<i>Predicted Class</i>
			Negatif	Negatif
			Negatif	Positif 1
			Negatif	Negatif
			Negatif	Negatif

<i>Original</i>	<i>Ground Truth</i>		<i>Segmented Result</i>		<i>Actual Class</i>	<i>Predicted Class</i>
					Negatif	Negatif
					Negatif	Negatif
					Negatif	Negatif
					Negatif	Negatif

<i>Original</i>	<i>Ground Truth</i>	<i>Segmented Result</i>		<i>Actual Class</i>	<i>Predicted Class</i>
				Negatif	Negatif
				Negatif	Negatif
				Negatif	Negatif
				Negatif	Negatif

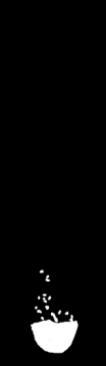
<i>Original</i>	<i>Ground Truth</i>	<i>Segmented Result</i>	<i>Actual Class</i>	<i>Predicted Class</i>
			Negatif	Negatif
			Negatif	Negatif
			Negatif	Negatif
			Negatif	Negatif

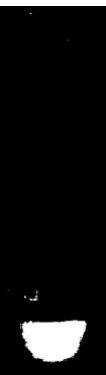
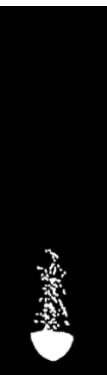
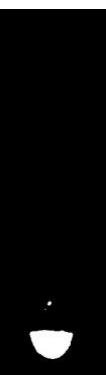
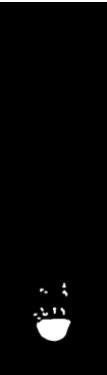
<i>Original</i>	<i>Ground Truth</i>	<i>Segmented Result</i>	<i>Actual Class</i>	<i>Predicted Class</i>
			Negatif	Negatif
			Negatif	Negatif
			Negatif	Negatif
			Negatif	Negatif

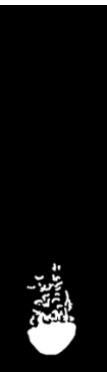
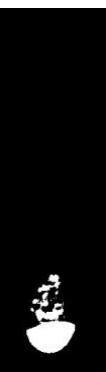
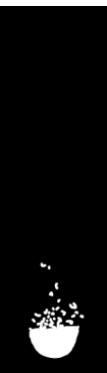
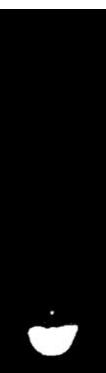
L18. Hasil Prediksi EfficientNetB7 pada Citra Kelas Positif 1

<i>Original</i>	<i>Ground Truth</i>	<i>Segmented Result</i>	<i>Actual Class</i>	<i>Predicted Class</i>
			Positif 1	Positif 1
			Positif 1	Negatif
			Positif 1	Positif 1
			Positif 1	Positif 1

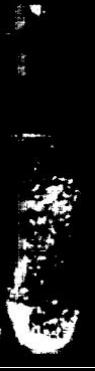
<i>Original</i>	<i>Ground Truth</i>	<i>Segmented Result</i>	<i>Actual Class</i>	<i>Predicted Class</i>
			Positif 1	Positif 1
			Positif 1	Positif 2
			Positif 1	Negatif
			Positif 1	Positif 1

<i>Original</i>	<i>Ground Truth</i>	<i>Segmented Result</i>	<i>Actual Class</i>	<i>Predicted Class</i>
			Positif 1	Positif 1
			Positif 1	Negatif
			Positif 1	Positif 1
			Positif 1	Positif 1

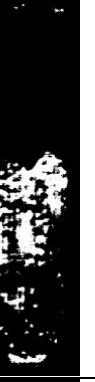
<i>Original</i>	<i>Ground Truth</i>	<i>Segmented Result</i>	<i>Actual Class</i>	<i>Predicted Class</i>
			Positif 1	Positif 1
			Positif 1	Negatif
			Positif 1	Positif 1
			Positif 1	Negatif

<i>Original</i>	<i>Ground Truth</i>	<i>Segmented Result</i>	<i>Actual Class</i>	<i>Predicted Class</i>
			Positif 1	Positif 1
			Positif 1	Negatif
			Positif 1	Positif 1

L19. Hasil Prediksi EfficientNetB7 pada Citra Kelas Positif 2

<i>Original</i>	<i>Ground Truth</i>	<i>Segmented Result</i>	<i>Actual Class</i>	<i>Predicted Class</i>
			Positif 2	Positif 2
			Positif 2	Positif 2
			Positif 2	Positif 2
			Positif 2	Positif 2

<i>Original</i>	<i>Ground Truth</i>	<i>Segmented Result</i>	<i>Actual Class</i>	<i>Predicted Class</i>
			Positif 2	Positif 2
			Positif 2	Positif 2
			Positif 2	Positif 2
			Positif 2	Positif 2

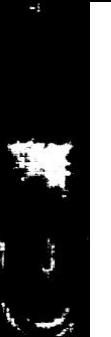
<i>Original</i>	<i>Ground Truth</i>	<i>Segmented Result</i>	<i>Actual Class</i>	<i>Predicted Class</i>
			Positif 2	Positif 2
			Positif 2	Positif 2
			Positif 2	Positif 2
			Positif 2	Positif 2

<i>Original</i>	<i>Ground Truth</i>	<i>Segmented Result</i>		<i>Actual Class</i>	<i>Predicted Class</i>
				Positif 2	Positif 2
				Positif 2	Positif 2
				Positif 2	Positif 2
				Positif 2	Positif 2

<i>Original</i>	<i>Ground Truth</i>	<i>Segmented Result</i>		<i>Actual Class</i>	<i>Predicted Class</i>
				Positif 2	Positif 1
				Positif 2	Positif 2
				Positif 2	Positif 2

L20. Hasil Prediksi EfficientNetB7 pada Citra Kelas Positif 3

<i>Original</i>	<i>Ground Truth</i>	<i>Segmented Result</i>	<i>Actual Class</i>	<i>Predicted Class</i>
			Positif 3	Positif 3
			Positif 3	Positif 3
			Positif 3	Positif 3
			Positif 3	Positif 3

<i>Original</i>	<i>Ground Truth</i>	<i>Segmented Result</i>	<i>Actual Class</i>	<i>Predicted Class</i>
			Positif 3	Positif 3

L21. Hasil Prediksi EfficientNetB7 pada Citra Kelas Positif 4

<i>Original</i>	<i>Ground Truth</i>	<i>Segmented Result</i>	<i>Actual Class</i>	<i>Predicted Class</i>
			Positif 4	Positif 4
			Positif 4	Positif 4
			Positif 4	Positif 4

<i>Original</i>	<i>Ground Truth</i>	<i>Segmented Result</i>	<i>Actual Class</i>	<i>Predicted Class</i>
			Positif 4	Positif 4
			Positif 4	Positif 4
			Positif 4	Positif 4
			Positif 4	Positif 4

L22. Gambar Interface Web untuk Klasifikasi Uji Silang Serasi

Welcome to Atomics

Recent Test

Rainata Putra Wib

Quick Actions

Upload Doc

Discussions

Blood Storage

Blood Type

3 Bags

Blood Type

2 Bags

Blood Type

1 Bags

First Step.

Please fill out the required information below.

Donor Identity

NIK * 51012313041403120

Full Name * Ida Bagus Kade Rainata Putra Wibawa

Address * Jalan Sriandi Gang Durian Blok C 14

Born Date * 11/11/2001

Blood Type * O

Recipient Identity

NIK * 51012313041403120

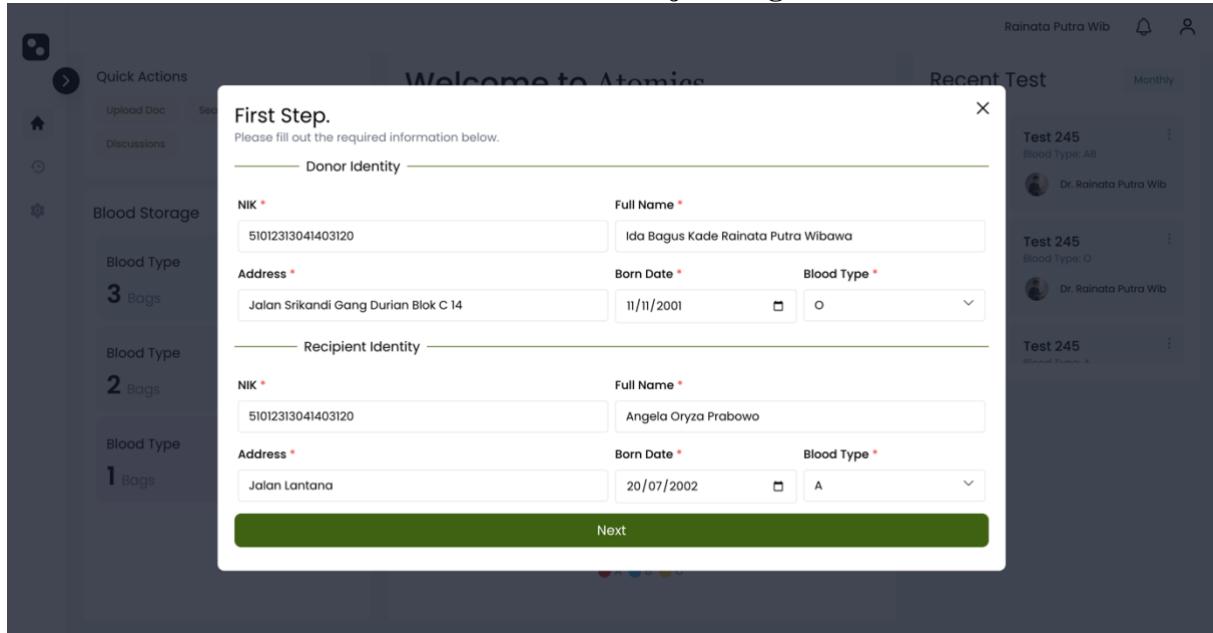
Full Name * Angela Oryza Prabowo

Address * Jalan Lantana

Born Date * 20/07/2002

Blood Type * A

Next



Rainata Putra Wib

Test

Monthly

Test 245 Blood Type: AB Dr. Rainata Putra Wib

Test 246 Blood Type: O Dr. Rainata Putra Wib

Test 247 Blood Type: A Dr. Rainata Putra Wib

Second Step.

Please upload the test images.

Crop your image first.

Scroll on the area of the image to zoom in and zoom out. Drag around the area of the image to change the crop position

CROP USE ANOTHER IMAGE

Zoom

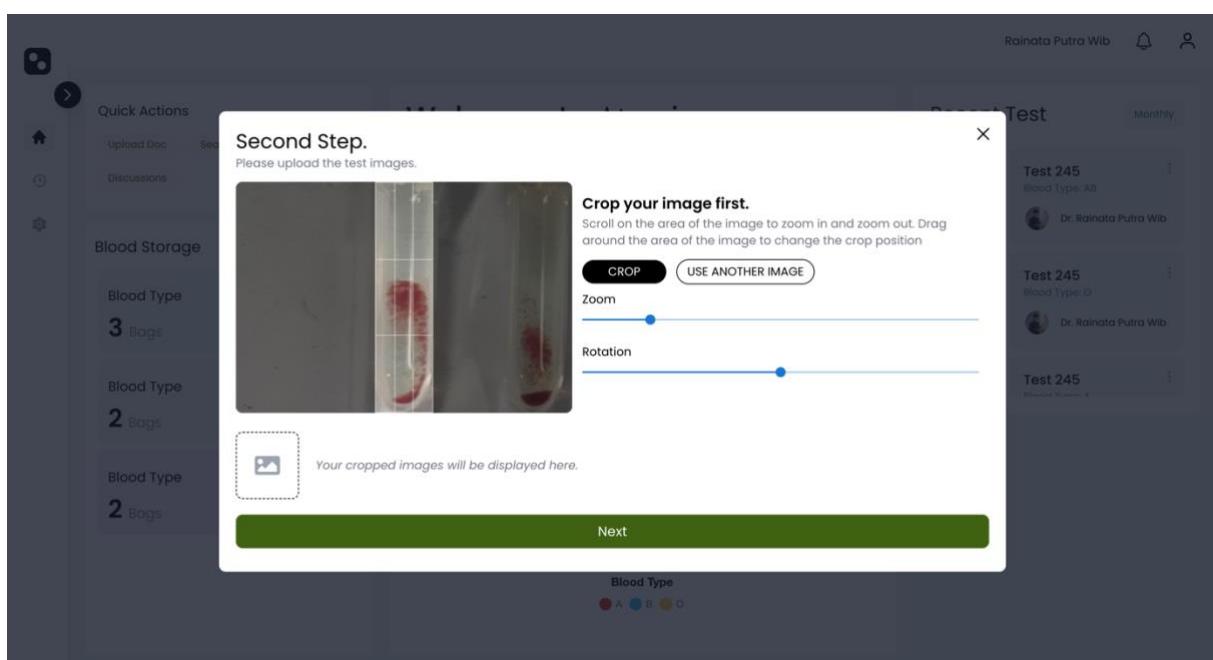
Rotation

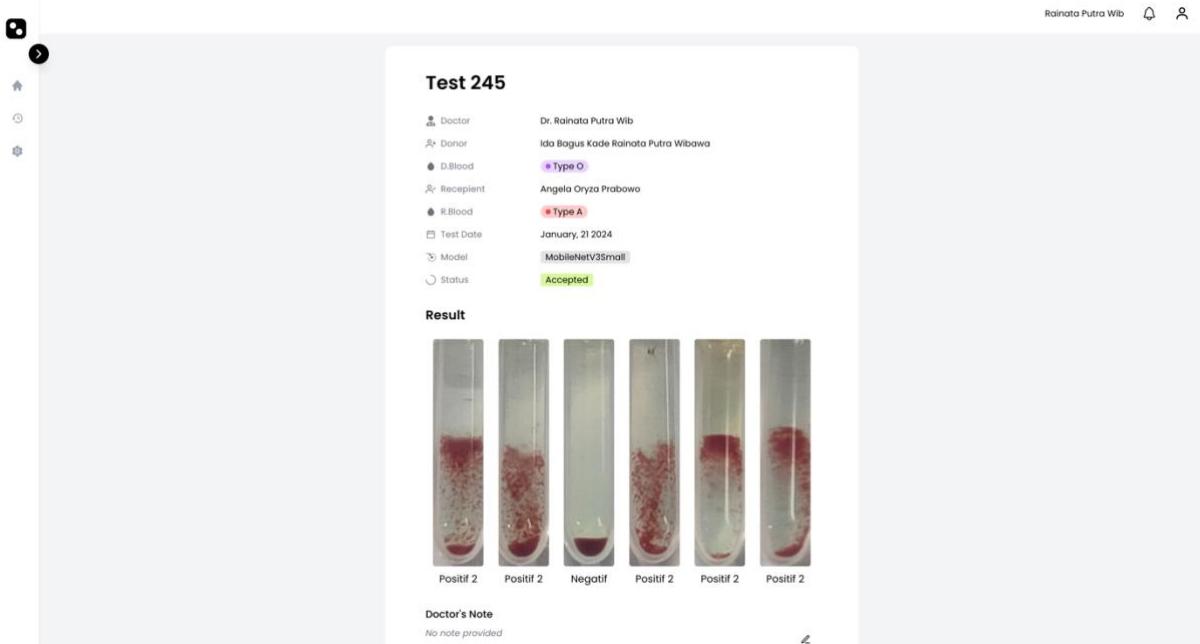
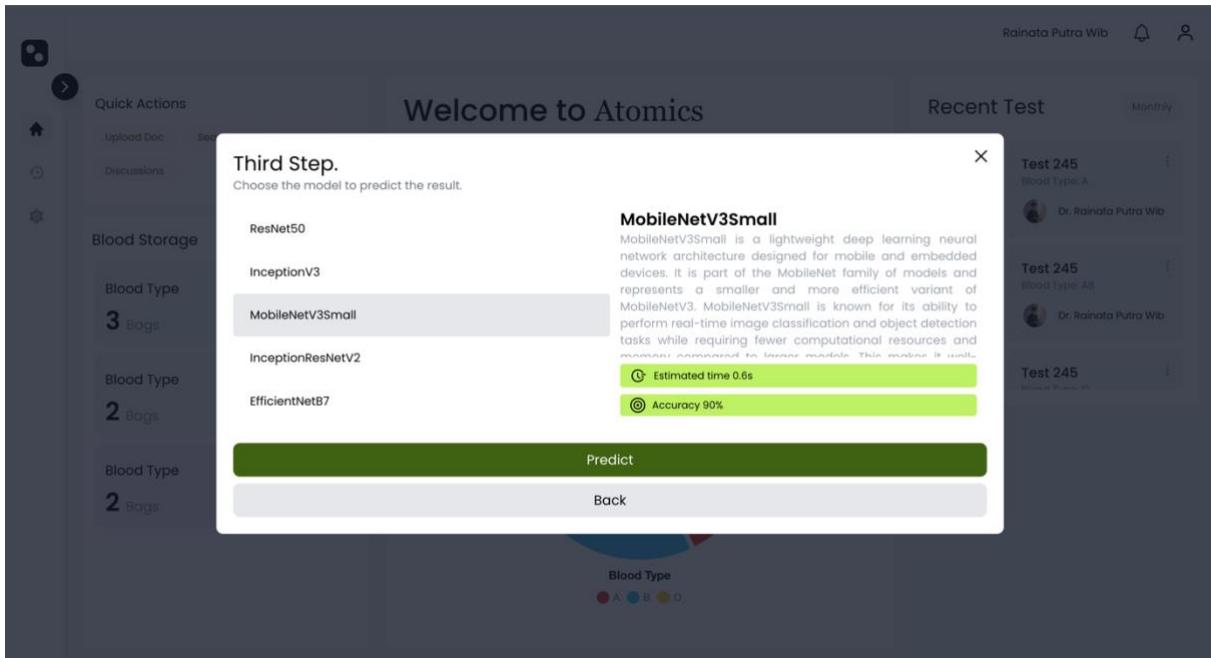
Your cropped images will be displayed here.

Next

Blood Type

A B O





BIODATA PENULIS



Penulis dilahirkan di Denpasar, 11 Novermber 2001, merupakan anak kedua dari 3 bersaudara. Penulis telah menempuh pendidikan formal yaitu di TK Santa Maria, SDN 1 Banjar Jawa, SMPN 4 Singaraja dan SMAN 1 Singaraja. Penulis mengikuti SBMPTN dan diterima di Departemen Teknik Informatika FTEIC pada tahun 2020 dan terdaftar dengan NRP 5025201235.

Di Departemen Teknik Informatika, penulis aktif dalam kegiatan belajar mengajar sebagai asisten dosen. Penulis merupakan seseorang yang tertarik dengan *deep learning*. Penulis dapat dikatakan sebagai seseorang yang cukup ambisius. Kekurangan penulis adalah sikap yang sering menunda-nunda.