

**OPTIMASI *HYPERPARAMETER CONVOLUTIONAL NEURAL NETWORK*
DALAM DETEKSI PATOGNOMIK WAJAH *THALASSEMIA BETA MAYOR*
PADA ANDROID**

SKRIPSI

**HAJARANI SYADZWANA
201401082**



**PROGRAM STUDI S-1 ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA
MEDAN
2024**

**OPTIMASI *HYPERPARAMETER CONVOLUTIONAL NEURAL NETWORK*
DALAM DETEKSI PATOGNOMIK WAJAH *THALASSEMIA BETA MAYOR*
PADA ANDROID**

SKRIPSI

Diajukan untuk melengkapi tugas dan memenuhi syarat memperoleh ijazah
Sarjana Ilmu Komputer

HAJARANI SYADZWANA

201401082



**PROGRAM STUDI S-1 ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA
MEDAN
2024**

PERSETUJUAN

Judul : OPTIMASI *HYPERPARAMETER*
CONVOLUTIONAL NEURAL NETWORK
 DALAM DETEKSI PATOGNOMIK WAJAH
THALASSEMIA BETA MAYOR PADA
 ANDROID

Kategori : SKRIPSI

Nama : HAJARANI SYADZWANA

Nomor Induk Mahasiswa : 201401082

Program Studi : SARJANA (S1) ILMU KOMPUTER

Fakultas : ILMU KOMPUTER DAN TEKNOLOGI
 INFORMASI UNIVERSITAS SUMATERA
 UTARA

Medan, 26 Maret 2024

Komisi Pembimbing :


Pembimbing I

Pembimbing II



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

NIP. 197812212014042001



Sri Melvani Hardi S.Kom., M.Kom

NIP. 198805012015042006

Diketahui/disetujui oleh

Program Studi S1 Ilmu Komputer



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

NIP. 197812212014042001

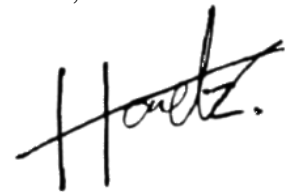
PERNYATAAN

**OPTIMASI *HYPERPARAMETER CONVOLUTIONAL NEURAL NETWORK*
DALAM DETEKSI PATOGNOMIK WAJAH *THALASSEMIA BETA MAYOR* PADA
ANDROID**

SKRIPSI

Saya mengakui bahwa skripsi ini adalah hasil karya saya sendiri, kecuali beberapa kutipan dan ringkasan yang masing-masing telah disebutkan sumbernya.

Medan, 24 Januari 2024



Hajarani Syadzwana

201401082

PENGHARGAAN

Semua pujian milik Allah Azza Wa Jalla, yang telah memberikan keberkahan, rahmat, taufik, inayah, dan hidayahNya kepada penulis sehingga mampu menyelesaikan salah satu syarat untuk memperoleh gelar Sarjana Komputer di Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara yaitu menyelesaikan skripsi ini. Shalawat mengucapkan salam kepada Nabi Muhammad, Nabi Muhammad Shallallahu ‘alaihi Wasallam, karena perantara beliau-lah kita bisa merasakan nikmat ilmu pengetahuan yang diridho Allah.

Penulis ingin mengucapkan rasa terima kasih dan penghargaan yang tulus kepada:

1. Orang tua dari penulis yang telah memberikan motivasi, dukungan, doa serta saran dan kasih sayang penuh kepada penulis dalam menyelesaikan pendidikan.
2. Ibu Dr. Maya Silvi Lydia, B.Sc., M.Sc., Dekan Fakultas Ilmu Komputer dan Teknologi Informasi di Universitas Sumatera Utara
3. Ibu Dr. Amalia ST., M.T. sebagai Ketua Program Studi S1 Ilmu Komputer di Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara Selain itu, sebagai dosen pembimbing I, telah membantu penulis menyelesaikan skripsi ini dengan memberikan bimbingan, kritik, motivasi, dan saran.
4. Ibu Sri Melvani Hardi S.Kom., M.Kom., dosen pembimbing II, telah membantu penulis menyelesaikan skripsi ini dengan memberikan bimbingan, kritik, motivasi, dan saran.
5. Bang Altaha yang penulis sayang, pemicu semangat, selalu mendukung dan menemani langkah-langkah penulis. Terima kasih telah menginspirasi dan selalu menolong penulis.
6. Seluruh Bapak dan Ibu Dosen Program Studi S-1 Ilmu Komputer yang telah memberikan waktu dan tenaga untuk mengajar dan membimbing sehingga penulis dapat sampai kepada tahap penyusunan skripsi ini.
7. Semua karyawan di Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara yang telah banyak membantu penulis selama perkuliahan hingga tahap penyusunan skripsi ini.

8. Saudara kandung penulis Mbak Wen, Mas Il, dan Dek Win, yang selalu mendukung serta mendoakan penulis dalam menjalankan aktivitas kuliah hingga akhirnya menyelesaikan tugas akhir.
9. Almarhum Mas Jarot dan almarhumah Teteh Bungsu selaku pejuang penyakit *thalassemia* yang telah menginspirasi penulis dalam pengambilan topik ini.
10. Teman-teman warga Dominates Wilson, Andrew, Rezha, Athika, Ariel, Iqbal, Imamul, Chalil, Ewaldo, Rio, Shafira, Yunisa, dan Anggi yang telah menemani penulis selama masa perkuliahan dan selalu menjadi penyemangat dan pendukung penulis terutama tempat penulis bertanya.
11. Teman-teman ciwik KOM B, Ultari, Difanie, Zahra, Devi, Vina, Salsa, Suki, Erlin, dan Holi yang selalu menemani penulis dan menyemangati satu sama lain
12. Tim SSA Teguh dan Wilbert, yang selama ini sepejuangan penulis dalam belajar pemrograman dan kompetisi.
13. Selain itu, semua pihak yang telah memberikan bantuan dan dukungan, yang tidak dapat disebutkan dengan nama satu per satu.

Semoga rahmat Allah selalu menemani langkah-langkah penulis serta diberikan keberkahan kepada semua orang yang telah memberikan motivasi, arahan, dan tindakan yang mendukung penulis dalam menyelesaikan skripsi ini. Semoga pribadi, keluarga, masyarakat, organisasi, dan negara mendapatkan manfaat dari skripsi ini.

Medan, 24 Januari 2024



Hajarani Syadzwana

201401082

**Optimasi *Hyperparameter Convolutional Neural Network* dalam Deteksi
Patognomik Wajah *Thalassemia Beta Mayor* pada Android**

ABSTRAK

Thalassemia merupakan kelompok kelainan genetik heterogen dimana penderita memiliki penurunan *beta hemoglobin* (Hb). Adanya manifestasi fisik pada wajah, yang umumnya dikenal sebagai "*cooley face*" atau "*mouse face*" menciptakan fitur khas yang menjadi indikator untuk *Beta Thalassemia Mayor* (β TM). Pengenalan wajah penderita β TM menjadi esensial untuk memicu intervensi medis yang sesuai, mengingat potensi penyakit ini dalam menyebabkan anemia berat, pembesaran organ, deformitas tulang, pertumbuhan terhambat, dan bahkan risiko kematian dalam dekade pertama kehidupan jika tidak ditangani secara adekuat. Pengklasifikasian dan pengenalan wajah penderita *Beta Thalassemia Mayor* menjadi penting sebagai langkah awal yang diperlukan dalam menggalakkan respons pengobatan yang cepat dan efektif. Penggunaan *Convolutional neural network* (CNN), khususnya arsitektur *DenseNet 121*, relevan dalam mengimplementasikan klasifikasi wajah. Fokus utama penelitian adalah optimalisasi *hyperparameter batch size* dan *learning rate* guna meningkatkan akurasi model. Penelitian ini mengadopsi metode *random search* untuk pemilihan kombinasi *hyperparameter* sebuah pendekatan yang mampu memberikan tingkat *accuracy* 97% pada model yang dibangun. Implementasi model pada platform aplikasi Android dirancang untuk meningkatkan aksesibilitas, portabilitas, kemudahan pengguna, dan kemampuan pemrosesan *real-time*. Hal ini memudahkan pengguna untuk secara mandiri mengambil gambar wajah dan memperoleh analisis cepat terkait potensi penderitaan *thalassemia*.

Kata Kunci: *Thalassemia Beta Mayor, Convolutional neural network, DenseNet 121, Hyperparameter, Random search*

Optimizing Convolutional Neural Network Hyperparameters for Pathognomonic Face Detection of Beta-Thalassemia Major on Android

ABSTRACT

Thalassemia is a heterogeneous group of genetic disorders where individuals have a reduction in beta hemoglobin (Hb). The presence of physical manifestations on the face, commonly known as "cooley face" or "mouse face," creates distinctive features that serve as indicators for Beta Thalassemia Major (β TM). Recognizing the facial characteristics of β TM patients becomes essential for triggering appropriate medical interventions, considering the potential of this condition to cause severe anemia, organ enlargement, bone deformities, stunted growth, and even the risk of death in the first decade of life if not adequately addressed. The classification and identification of the faces of Beta Thalassemia Major patients are crucial as the initial steps to encourage prompt and effective treatment responses. The use of Convolutional Neural Network (CNN), specifically the DenseNet 121 architecture, is relevant in implementing facial classification. The primary focus of the research is the optimization of hyperparameters such as batch size and learning rate to enhance the model's accuracy. This study adopts the random search method for selecting hyperparameter combinations, an approach that achieved a 97% accuracy rate in the constructed model. The implementation of the model on an Android application platform is designed to improve accessibility, portability, user-friendliness, and real-time processing capabilities. This facilitates users to independently capture facial images and obtain quick analyses related to the potential presence of thalassemia.

Keywords: *Beta Thalassemia Major, Convolutional neural network, DenseNet 121, Hyperparameter, Random search*

DAFTAR ISI

PERSETUJUAN	iii
PERNYATAAN	iv
PENGHARGAAN	v
ABSTRAK	vii
ABSTRACT	viii
DAFTAR ISI	ix
DAFTAR TABEL	xi
DAFTAR GAMBAR	xii
BAB I PENDAHULUAN	1
1.1. Latar Belakang	1
1.2. Rumusan Masalah	3
1.3. Batasan Masalah	3
1.4. Tujuan Penelitian	4
1.5. Manfaat Penelitian	4
1.6. Metodologi Penelitian	5
1.7. Sistematika Penulisan	6
BAB II LANDASAN TEORI	7
2.1. <i>Thalassemia Beta Mayor</i>	7
2.2. <i>Hyperparameter</i>	8
2.3. <i>DenseNet 121</i>	14
2.4. <i>Random search Optimization</i>	16
2.5. <i>Image Augmentation</i>	17
2.6. Penelitian yang Relevan	20
BAB III ANALISIS DAN PERANCANGAN SISTEM	22

3.1.	Analisis Sistem.....	22
3.1.1.	Analisis Masalah	22
3.1.2.	Analisis Kebutuhan	23
3.1.2.1.	Kebutuhan Fungsional	23
3.1.2.2.	Kebutuhan <i>Non-Fungsional</i>	24
3.1.3.	Diagram Umum Sistem	24
3.2.	Pemodelan Sistem	25
3.2.1.	<i>Use Case Diagram</i>	25
3.2.2.	<i>Activity Diagram</i>	26
3.2.3.	<i>Sequence Diagram</i>	28
3.3.	<i>Flowchart</i>	29
3.4.	<i>Work Flow</i>	30
3.5.	Perancangan <i>Interface</i>	35
3.5.1	Logo Aplikasi	35
3.5.2	Halaman Utama Aplikasi	36
BAB IV IMPLEMENTASI DAN PENGUJIAN SISTEM		37
4.1.	Implementasi	37
4.2.	Implementasi <i>Random Search Optimization</i>	41
4.3.	Perbandingan Optimasi	43
4.4.	Hasil Pengujian di Android	48
BAB V PENUTUP		51
5.1.	KESIMPULAN	51
5.2.	SARAN	52
DAFTAR PUSTAKA		53
LAMPIRAN		56

DAFTAR TABEL

Tabel 2. 1 Arsitektur DenseNet 121	15
Tabel 3. 1 Jumlah Data.....	33
Tabel 3. 2 Image Augmentation	34
Tabel 4. 1 Trial Training Model.....	42
Tabel 4. 2 Perbandingan Optimasi dan Default	44
Tabel 4. 3 Hasil Pengujian	48

DAFTAR GAMBAR

Gambar 2. 1 Wajah Pendertia Thalassemia.....	8
Gambar 2. 2 Kurva Learning Rate Besar	11
Gambar 2. 3 Kurva Learning Rate Kecil.....	12
Gambar 2. 4 Forward Propagation	13
Gambar 2. 5 Random Search Optimization.....	17
Gambar 3. 1 Ishikawa Diagram.....	23
Gambar 3. 2 Diagram Umum Sistem	25
Gambar 3. 3 Use Case Diagram	26
Gambar 3. 4 Activity Diagram	27
Gambar 3. 5 Sequence Diagram.....	28
Gambar 3. 6 Flow Chart Random search	29
Gambar 3. 7 Work Flow	30
Gambar 3. 8 Flow Data Gathering	32
Gambar 3. 9 Flow Data Cleaning	33
Gambar 3. 10 Pie Chart Split Data	33
Gambar 3. 11 Logo App.....	35
Gambar 3. 12 Halaman Utama App	36
Gambar 4. 1 Tampilan App pada Android	37
Gambar 4. 2 Halaman Utama	38
Gambar 4. 3 Open Camera	39
Gambar 4. 4 Setelah di Capture.....	39
Gambar 4. 5 Open Gallery.....	40
Gambar 4. 6 Terdeteksi Sehat	41
Gambar 4. 7 Terdeteksi Thalassemia	41
Gambar 4. 8 Grafik LR dan Val Accuracy	43
Gambar 4. 9 Grafik Validasi Optimasi.....	45
Gambar 4. 10 Grafik Validasi Default	45
Gambar 4. 11 Grafik Loss Optimasi	46
Gambar 4. 12 Grafik Loss Default	46

BAB I

PENDAHULUAN

1.1. Latar Belakang

Thalassemia merupakan kelainan genetic yang heterogen dihasilkan dari penurunan *beta hemoglobin* (Hb). *Thalassemia* adalah penyakit yang diturunkan, yang berarti salah satu dari dua orang tua harus mewarisi penyakit tersebut (Nooradi Praramdana dkk., 2023).

Keterlibatan tulang wajah dapat menciptakan fitur khas yang menyebabkan istilah "*Cooley face*" atau "*mouse face*" atau "*mongoloid*" atau "*rodent face*" digunakan untuk menggambarkan pasien *thalassemia* (Adamopoulos & Petrocheilou, 2020). Tulang *frontal* dan parietal menjadi menonjol sehingga tampak tonjolan di dahi, *zygoma* menonjol, jembatan hidung cekung dan mungkin melebar (hidung pelana), dan mata memiliki kemiringan mongoloid. *Hipertrofi* yang mencolok pada maksila atas, yang tampak sebagai tulang pipi yang membengkak, merupakan tanda patognomonik yang khas untuk β TM (*Beta Thalassemia Mayor*), terutama jika disertai dengan hipopneumatization antrum (Adamopoulos & Petrocheilou, 2020).

Jika tidak diobati, pasien β TM bisa mengalami anemia berat, pembesaran hati dan limpa, beberapa deformitas tulang, pertumbuhan yang terhambat, dan biasanya meninggal karena gagal jantung dalam dekade pertama kehidupan (Adamopoulos & Petrocheilou, 2020). Perjalanan dari penyakit *thalassemia* umumnya pendek karena jika pengidap tidak menerima bantuan transfusi, kematian dini akibat anemia berat bisa terjadi (Nooradi Praramdana dkk., 2023).

Mengklasifikasikan dan mengenali wajah penderita *Beta Thalassemia Mayor* menjadi penting sebagai langkah awal mengambil tindakan untuk keberlanjutan penanganan *thalassemia*. Untuk masalah klasifikasi gambar, model pembelajaran *optimization default* dalam *computer vision* adalah *convolutional neural networks* (CNN atau ConvNet) (Greeshma & Sreekumar, 2019).

Dense convolutional network (*DenseNet*) merupakan arsitektur baru dari CNN (Huang dkk., 2016). *DenseNet* cenderung memberikan peningkatan akurasi yang konsisten dengan bertambahnya jumlah parameter, tanpa adanya tanda-tanda

penurunan kinerja atau *overfitting* (Huang dkk., 2016). Peningkatan akurasi lebih lanjut pada *DenseNet* dapat diperoleh melalui *detailed tuning of hyperparameters* dan *learning rate schedules* (Huang dkk., 2016).

Optimized learning sering mengacu pada beberapa *hidden elements* yakni *hyperparameters* karena mereka merupakan salah satu komponen paling penting dari setiap aplikasi *optimized learning*. *Hyperparameters* adalah elemen *fine tuning* yang berada di luar model tetapi dapat sangat mempengaruhi perilakunya, dan performa model sangat bergantung pada pemilihan *hyperparameter* yang tepat (Greeshma & Sreekumar, 2019). *Hyperparameter* mengendalikan proses pembelajaran selama pelatihan jaringan saraf dalam *optimized learning* (Jafar & Lee, 2023). Salah satu masalah praktis utama yang dialami dalam proyek *optimized learning* adalah menemukan kombinasi *hyperparameter* yang ideal yang meminimalkan atau memaksimalkan fungsi objektif (Lacerda dkk., 2021). Keefektifan *optimized learning* pada akhirnya bergantung pada implementasi *hyperparameter* (Badriyah dkk., 2019).

Pada penelitian ini peneliti menggunakan arsitektur *DenseNet 121* karena terbukti efektif dalam tugas-tugas pengenalan gambar dan klasifikasi. Arsitektur ini memiliki keunggulan dalam memahami pola-pola kompleks pada gambar yang memungkinkan setiap *layer* berkomunikasi langsung dengan semua *layer* pada kedalaman yang sama. Hal ini bermanfaat dalam mengekstraksi fitur wajah yang khas pada penderita β TM. Bersama dengan kemampuan *DenseNet 121* untuk mengatasi masalah *vanishing gradient* mampu meningkatkan akurasi model klasifikasi. *Hyperparameter* yang digunakan penulis untuk melakukan optimalisasi yaitu *batch size* dan *learning rate*. Karena keduanya merupakan *hyperparameter* penting dan memiliki dampak langsung pada model pelatihan. *Batch size* yang terlalu besar bisa mempengaruhi konvergensi yang kurang stabil, *learning rate* yang salah mengakibatkan gagal konvergensi atau melompati nilai minimum. Untuk pemilihan kombinasi *hyperparameter* penulis menggunakan *random search*. Dibandingkan dengan eksperimen *grid search* yang dilakukan oleh Larochelle dkk. (2007), dalam kebanyakan kasus, pencarian acak menghasilkan model yang lebih baik dan memerlukan waktu komputasi yang lebih sedikit. *Random search* juga lebih mudah dilakukan daripada *grid search* karena alasan praktis yang terkait

dengan independensi statistik dari setiap percobaan (Bergstra dkk., 2012). Penelitian ini menggunakan *dataset* yang dikumpulkan sendiri oleh penulis melalui berbagai macam sumber seperti *google image*, *web thalassemia* maupun *social media*. Dan melakukan penyesuaian pada data dan model yang dibangun untuk mendapatkan tingkat akurasi yang dapat digunakan untuk deteksi wajah pada aplikasi Android.

1.2. Rumusan Masalah

Pengenalan wajah *Thalassemia Beta Mayor* yang harus cepat dilakukan sebagai tindakan untuk mengambil langkah pengobatan mengharuskan pengenalan wajah yang akurat dan cepat, untuk itu dilakukan penelitian lebih lanjut dengan optimasi *hyperparameter* pada *convolutional neural network* dan mengimplementasikannya pada aplikasi Android agar mendapatkan aksesibilitas yang luas, portabilitas, kemudahan pengguna, dan *real-time processing*. Yang mana pengguna dapat mengambil gambar dari kamera atau penyimpanan internal Android dan mengunggahnya ke aplikasi untuk mendapatkan hasil analisis apakah *thalassemia* atau tidak.

1.3. Batasan Masalah

Batasan yang terkait dengan topik penelitian ini adalah sebagai berikut:

1. Menggunakan dua klasifikasi gambar yaitu gambar wajah sehat dan gambar wajah penderita *Thalassemia Beta Mayor*. Deteksi hanya dilakukan pada dua jenis gambar tersebut.
2. *Batch size* dan *learning rate* adalah dua *hyperparameter* yang digunakan.
3. Untuk pencarian *hyperparameter* terbaik digunakan *random search*.
4. Sebagai perbandingan digunakan nilai default untuk *batch size* sebesar 32 dan *learning rate* sebesar 0,001 sebagai pembanding dengan hasil setelah optimasi.
5. Model yang dibangun diterapkan pada sistem Android. Dengan melakukan optimasi pada model tersebut diharapkan memiliki tingkat akurasi tinggi yang mampu memprediksi penderita *thalassemia* dengan baik, tetapi tidaklah cukup untuk menentukan apakah model tersebut berkinerja paling optimal di Android.

6. Pada aplikasi Android, untuk melakukan pengambilan gambar yang diuji dapat melalui kamera atau memilih gambar dari penyimpanan internal.
7. Model yang dibangun menggunakan bahasa pemrograman Python dengan menggunakan model CNN yaitu arsitektur *DenseNet 121*. Implementasi pada aplikasi Android menggunakan bahasa pemrograman yang umum untuk Android yaitu Java.

1.4. Tujuan Penelitian

Tujuan penelitian yang diharapkan adalah:

1. Melakukan pengenalan awal pada wajah penderita *Thalassemia Beta Mayor* pada aplikasi Android sebagai bentuk screening awal yang mudah diakses bagi masyarakat umum. Diharapkan, dengan pengenalan dini ini, dapat meningkatkan peluang untuk memberikan penanganan lebih cepat kepada para penderita, yang pada akhirnya diharapkan dapat meningkatkan angka harapan hidup penderita *Thalassemia Beta Mayor*.
2. Mengoptimasi model *optimized learning* untuk peningkatan kualitas model sehingga mendapatkan nilai *accuracy* tertinggi dan *loss function* (fungsi kerugian) yang terendah. Dimana *accuracy* dan *loss function* merupakan metrik evaluasi di dalam *machine learning* untuk mengukur seberapa baik model *machine learning* memetakan *input* ke *output* yang diharapkan.

1.5. Manfaat Penelitian

Penelitian ini diharapkan menghasilkan aplikasi yang dapat digunakan oleh masyarakat umum, khususnya yang belum mengetahui ciri wajah penderita *thalassemia*. Manfaat penelitian sebagai berikut:

1. Memudahkan masyarakat dalam melakukan pengenalan dini *thalassemia* secara sederhana melalui perangkat Android.
2. Ciri khusus pada wajah penderita *thalassemia* menjadi patokan utama untuk pengenalan penderita *thalassemia*. Sehingga deteksi hanya membutuhkan gambar wajah saja.
3. Meningkatkan kemungkinan diagnosis dini *thalassemia* oleh tenaga medis, yang berakibat pada peningkatan angka harapan hidup dan kualitas hidup penderita.

1.6. Metodologi Penelitian

Berikut ini adalah metode penelitian yang digunakan dalam penelitian ini:

1. Studi Pustaka

Hal awal yang dilakukan adalah melakukan observasi, meneliti, mengeksplor, mempelajari, dan menelaah berbagai referensi yang didasarkan pada judul penelitian untuk mengumpulkan berbagai informasi relevan dan memfilter data yang diperlukan untuk penelitian ini. Artikel-artikel ilmiah, buku-buku cetak, media cetak dan *online*, dan penelitian lain yang telah dipublikasikan, seperti jurnal dan prosiding, termasuk dalam daftar referensi yang berkaitan dengan optimisasi *random search*, CNN, *DenseNet 121*, *Hyperparameter*, *Thalassemia Beta Mayor*.

2. Analisis dan Perancangan

Pada tahap ini dilakukan analisis terhadap kebutuhan penelitian baik dari segi teori maupun perancangan sistem berupa diagram alir (*flowchart*) dan *Work flow* dan diagram Ishikawa terhadap penelitian yang dilakukan mengenai optimalisasi *hyperparameter*.

3. Implementasi Sistem

Dalam melakukan implementasi dibangun sistem menggunakan bahasa pemrograman Java pada sisi *front-end* untuk membuat aplikasi di Android dan bahasa pemrograman Python untuk membangun model.

4. Pengujian Sistem

Pada tahap ini, proses pengujian yang diperlukan dilakukan pada sistem untuk memastikan bahwa program berjalan sesuai dengan harapan. Pastikan bahwa sistem yang dibuat dapat mengidentifikasi pasien dengan *Thalassemia Beta Mayor* menggunakan gambar yang dimasukkan.

5. Dokumentasi Sistem

Melalui tahap ini, dilakukan proses dokumentasi terhadap keseluruhan tahap mulai dari analisis sampai ke tahap pengujian sistem dengan tujuan untuk menunjukkan hasil penelitian yang telah dilakukan.

1.7. Sistematika Penulisan

Sistematika penulisan dari skripsi ini terdiri dari lima bab, yakni:

BAB I PENDAHULUAN

Bab ini memberikan penjelasan lengkap tentang latar belakang masalah yang akan diteliti, termasuk rumusan masalah, tujuan penelitian, batasan masalah, tujuan, keuntungan, metode penelitian, dan sistematika penulisan.

BAB II LANDASAN TEORI

Mengumpulkan dan melakukan peninjauan teoritis yang berkaitan dengan *thalassemia*, *DenseNet 121* dan *random search* dibahas pada bab ini.

BAB III ANALISIS DAN PERANCANGAN

Analisis terhadap permasalahan dan sistem yang dibangun dibahas pada bab ini, lalu berlanjut ke tahapan perancangan sistem dengan menggunakan model yang paling optimal dibangun dalam sistem Android.

BAB IV IMPLEMENTASI DAN PENGUJIAN SISTEM

Pada bab ini dilakukan implementasi dan pengujian sistem yang didasarkan dari tahapan analisis dan perancangan.

BAB V KESIMPULAN DAN SARAN

Bab terakhir ini berisi kesimpulan dari penelitian yang telah dilaksanakan dan penulisan saran untuk penelitian selanjutnya.

BAB II

LANDASAN TEORI

2.1. *Thalassemia Beta Mayor*

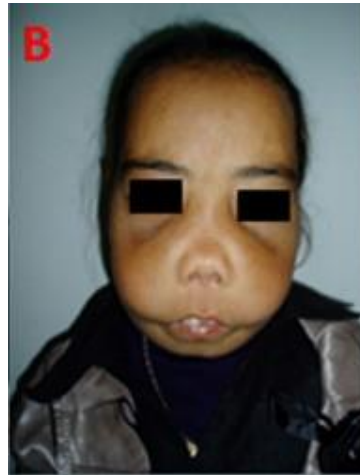
Thalassemia merupakan salah satu gangguan genetik paling umum di seluruh dunia. Ini ditemukan di lebih dari 60 negara, dengan distribusi tertinggi di wilayah Mediterania, sebagian Afrika, Timur Tengah, anak benua India, Asia Tenggara, dan Asia Timur (Khojastepour dkk., 2022)

Thalassemia merupakan kondisi *anemia hemolitik* turunan ditandai oleh kurangnya atau tidak terdapat produksi beberapa rantai *globin* dalam *hemoglobin*. Penurunan produksi rantai *globin* ini menyebabkan *hemoglobin* tidak dapat terbentuk dengan baik, menghasilkan rantai α dan β yang tidak berpasangan dan tidak efektif dalam mengikat oksigen. Akumulasi rantai yang tidak berpasangan ini mengganggu berbagai sistem organ tubuh, meningkatkan risiko kematian jika tidak diobati (Roussos dkk., 2021).

Thalassemia terbagi menjadi dua tipe, yaitu alfa dan *beta*, tergantung pada jenis gen yang terkena dampak. *Beta thalassemia* memiliki tiga klasifikasi, yaitu *mayor*, *intermediate*, dan *minor*, bergantung pada tingkat produksi rantai β . Gejala klinis dapat bervariasi, mulai dari kelainan morfologi yang ringan hingga kondisi yang mengancam nyawa. Perubahan pada kerangka tubuh terjadi karena sumsum tulang merah yang membesar, mengakibatkan perluasan rongga sumsum tulang dan deformitas kerangka, seperti osteoporosis dan patah tulang yang bersifat patologis. Bagian tengkorak, maksila, dan vertebra adalah struktur kraniofasial yang paling sering terpengaruh, menyebabkan bentuk wajah yang cembung. Terdapat perbedaan pandangan di dalam literatur mengenai hal ini, beberapa penelitian melaporkan adanya protrusi pada tulang frontal dan pipi, depresi pada jembatan hidung, serta kemiringan mata, namun penelitian antropometri terbaru tidak menunjukkan perbedaan statistik yang signifikan dalam daerah kepala dan mata antara pasien *thalassemia* dan kelompok kontrol yang sehat. Ukuran gigi dan dimensi lengkungan gigi seringkali menurun, perkembangan dan pertumbuhan gigi tertunda, dan perubahan warna gigi seringkali terlihat. Perluasan sumsum tulang dapat mempengaruhi posisi gigi, menyebabkan celah antar gigi dan insisivus

maksila yang melebar, hubungan antar rahang yang tidak seimbang, dan gigitan terbuka pada bagian depan (Roussos dkk., 2021).

Lihatlah gambar 2.1 yang menunjukkan wajah seorang penderita thalassemia, yang telah mengalami deformasi tulang pada wajah.



Gambar 2. 1 Wajah Pendertia Thalassemia

Penampilan wajah pada pasien dengan *beta-thalassemia mayor* telah diibaratkan sebagai "wajah rodent". Perbaikan dalam manajemen medis pasien *thalassemia* telah menyebabkan peningkatan harapan hidup (Khojastepour dkk., 2022).

2.2. *Hyperparameter*

Hyperparameter dalam *machine learning* (ML) merujuk pada pengaturan-pengaturan yang tidak dipelajari oleh model dari data, tetapi diatur sebelum proses pelatihan model. *Hyperparameter* memiliki pengaruh besar terhadap kinerja dan perilaku dari model ML. Proses optimasi *hyperparameter* (HPO) bertujuan untuk menemukan kombinasi pengaturan *hyperparameter* yang optimal untuk mencapai kinerja terbaik dari model dalam batasan atau *cost* yang ditetapkan.

Hyperparameter dapat dibagi menjadi beberapa jenis berdasarkan sifatnya:

1. *Continuous Hyperparameters*: Parameter yang memiliki rentang nilai yang kontinu, seperti tingkat pembelajaran (*learning rate*), kekuatan regularisasi, atau ambang batas (*threshold*). Rentang nilai dari *hyperparameter* ini bersifat

kontinu dan bisa memuat nilai-nilai dalam rentang tertentu tanpa batasan nilai diskrit atau spesifik.

2. *Discrete Hyperparameters*: *Hyperparameter* yang memiliki nilai-nilai diskrit yang spesifik. Contohnya adalah *decision tree leaf nodes* atau jumlah *cluster* dalam algoritma K-means yang hanya bisa bernilai bilangan bulat.
3. *Categorical Hyperparameters*: Parameter yang terbatas pada pilihan-pilihan tertentu. Contohnya adalah pilihan *activation functions* dalam jaringan saraf tiruan atau pemilihan *optimizer*.

Beberapa *hyperparameter* juga dapat bersifat kondisional, di mana nilai dari satu *hyperparameter* dapat bergantung pada nilai dari *hyperparameter* lainnya.

Optimisasi *hyperparameter* melibatkan beberapa komponen, termasuk estimator/fungsi tujuan (*objective function*) yang merupakan algoritma ML yang disesuaikan, ruang pencarian (*search space*) yang mendefinisikan rentang nilai dari *hyperparameter*, metode pencarian/optimasi yang digunakan, dan fungsi evaluasi yang mengevaluasi kinerja dari berbagai konfigurasi *hyperparameter*.

Tujuan utama dari HPO adalah untuk menemukan kombinasi *hyperparameter* yang dapat meminimalkan (atau memaksimalkan) fungsi tujuan yang telah ditentukan dalam batasan komputasi atau waktu yang ada. Karena proses evaluasi setiap konfigurasi *hyperparameter* memerlukan pelatihan dan validasi model yang membutuhkan sumber daya komputasi, maka optimisasi *hyperparameter* harus dilakukan secara efisien sesuai dengan batasan waktu yang tersedia.

Secara keseluruhan, optimisasi *hyperparameter* sangat penting dalam mencapai kinerja model yang optimal dengan menyesuaikan pengaturan *hyperparameter* secara efektif dalam batasan yang telah ditetapkan. Berbagai teknik optimisasi dan pertimbangan yang hati-hati terhadap ruang pencarian serta batasan sangat diperlukan untuk mendapatkan hasil HPO yang berhasil (Yang & Shami, 2020).

2.2.1 Batch size

Istilah "*batch size*" digunakan dalam pembelajaran mesin untuk menunjukkan berapa banyak sampel data pelatihan yang digunakan dalam satu iterasi. Hal ini merupakan salah satu parameter penting yang perlu disesuaikan dalam sistem *optimized learning* (Rochmawati dkk., 2021).

Dengan melakukan vektorisasi memungkinkan perhitungan yang efisien yang dapat menyederhanakan kode tetapi juga meningkatkan kinerja dan mengurangi beban komputasi.

Sebagai contoh dengan 96 m , maka:

$$\begin{aligned}
 X_{(nx,m)} &= [x^{(1)}x^{(2)}x^{(3)} \dots x^{(m)}] \\
 X_{(nx,96)} &= [\underbrace{x^{(1)}x^{(2)}x^{(3)} \dots x^{(32)}}_{X^{(1)}} | \underbrace{x^{(33)}x^{(34)}x^{(35)} \dots x^{(64)}}_{X^{(2)}} | \underbrace{x^{(65)}x^{(66)}x^{(67)} \dots x^{(96)}}_{X^{(3)}}] \\
 Y_{(1,m)} &= [y^{(1)}y^{(2)}y^{(3)} \dots y^{(m)}] \\
 Y_{(1,96)} &= [\underbrace{y^{(1)}y^{(2)}y^{(3)} \dots y^{(32)}}_{Y^{(1)}} | \underbrace{y^{(33)}y^{(34)}y^{(35)} \dots y^{(64)}}_{Y^{(2)}} | \underbrace{y^{(65)}y^{(66)}y^{(67)} \dots y^{(96)}}_{Y^{(3)}}]
 \end{aligned}$$

Misalnya, dalam kasus di atas $X (nx, m)$ adalah matriks data masukan dengan dimensi nx (misalnya, fitur-fitur dari setiap contoh) dan m adalah jumlah total contoh dalam *dataset*.

Demikian pula, $Y (1, m)$ adalah vektor target atau label untuk m contoh dalam *dataset* tersebut. Dalam pelatihan model, Digunakan pasangan data x dan y untuk mengajarkan model agar dapat mempelajari hubungan antara data masukan (x) dan target keluaran (y).

Dari permasalahan contoh tersebut *batch size* adalah 32, jadi untuk sekali iterasi pelatihan, model memproses 32 sampel data secara bersamaan sebelum melakukan penyesuaian parameter (*update* bobot) berdasarkan *loss* yang dihitung.

2.2.2 Learning rate

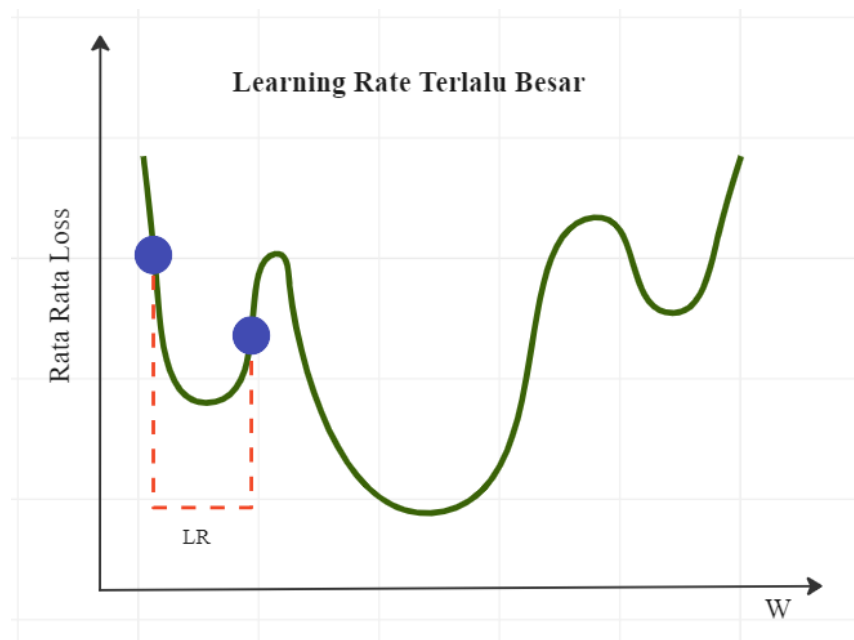
Learning rate (LR) digunakan untuk memperbarui bobot (*weights*) selama proses pelatihan. Dengan nilai LR yang tinggi, algoritma optimisasi melompat secara besar-besaran dan melewatkan informasi yang penting. Nilai LR yang terlalu tinggi dapat membuat proses optimisasi melewatkan nilai optimum dari fungsi yang ingin dioptimalkan. Sebaliknya, nilai optimal yang lebih rendah dari LR lebih baik untuk mengatur perubahan gradien saat

pembaruan bobot dalam pelatihan. Dengan nilai LR yang lebih rendah, proses optimisasi menjadi lebih stabil dan cenderung untuk menghadapi loncatan yang lebih kecil dalam pencarian nilai minimum dari fungsi yang ingin dioptimalkan (Jafar & Lee, 2023).

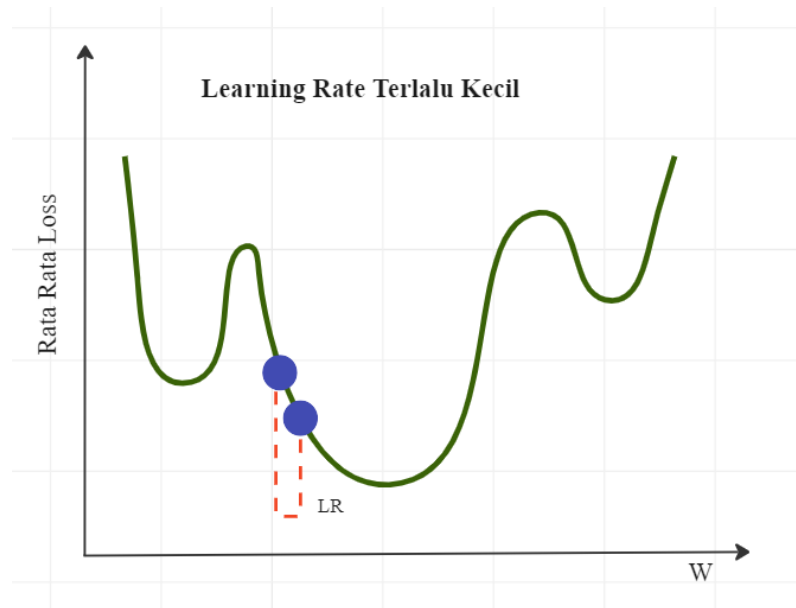
Setiap iterasi melangkah berdasarkan “*learning rate*”:

$$W = W - \eta (\delta L / \delta W)$$

Penentuan nilai LR yang tepat adalah penting dalam proses pelatihan model *machine learning*. Jika LR terlalu tinggi (lihat pada gambar 2.1), bisa menyebabkan proses pelatihan tidak stabil atau bahkan dapat melewati nilai minimum yang diinginkan. Sebaliknya, nilai LR yang terlalu rendah (lihat pada gambar 2.2) bisa menyebabkan proses pelatihan menjadi lambat dan membutuhkan lebih banyak iterasi untuk mencapai konvergensi pada nilai minimum yang optimal. Oleh karena itu, penyesuaian nilai LR dengan tepat sangat penting untuk mencapai keseimbangan antara kecepatan konvergensi dan stabilitas selama proses pelatihan model.



Gambar 2. 2 Kurva Learning Rate Besar

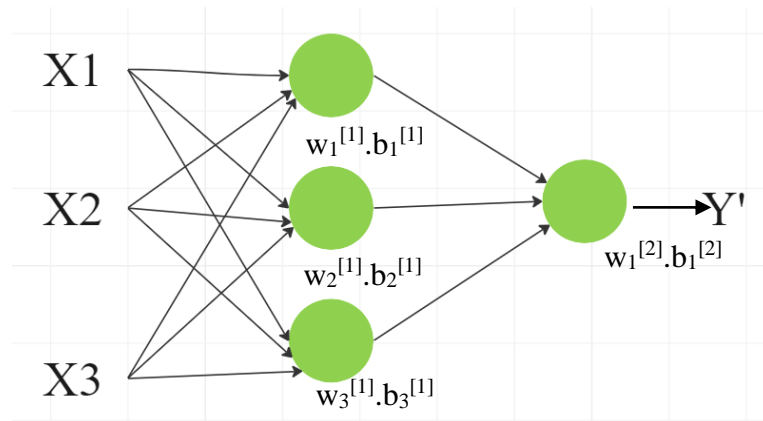


Gambar 2. 3 Kurva Learning Rate Kecil

2.2.3 Activation Function

Activation *function* adalah komponen penting dalam jaringan saraf tiruan yang memperkenalkan *non-linearitas*, memungkinkan jaringan untuk belajar pola dan hubungan yang kompleks dalam data. Fungsi ini beroperasi pada jumlah bobot *input* bersama dengan bias di setiap neuron dan menentukan apakah neuron harus diaktifkan, mempengaruhi aliran informasi melalui jaringan.

Secara sederhana, fungsi aktivasi mengambil masukan yang dijumlahkan dari lapisan sebelumnya, menerapkan transformasi padanya, dan menghasilkan keluaran (biasanya keluaran untuk lapisan berikutnya). Tanpa fungsi aktivasi, jaringan saraf, terlepas dari kedalaman, menjadi model *regresi linear* (Andrinandrasana David Rasamoelina dkk., 2020).



Gambar 2. 4 *Forward Propagation*

Seperti yang terlihat pada gambar *forward propagation* di atas, jika diberikan x , maka:

- $z[1] = w[1] \cdot x + b[1]$
 $a[1] = g(z[1])$
- $z[2] = w[2] \cdot a[1] + b[2]$
 $a[2] = g(z[2])$

Keterangan:

- Perhitungan pertama adalah menghitung nilai z yaitu perkalian bobot $w[1]$ dengan x ditambah dengan bobot pada neuron 1 $b[1]$.
 Untuk mendapatkan nilai $a[1]$ yaitu *output* pada layer pertama maka dilakukan aktivasi fungsi dengan fungsi $g(z[1])$.
- Dilanjutkan untuk layer kedua menghitung nilai z yang merupakan perkalian dari $w[2]$ dengan $a[1]$ (yang merupakan hasil dari layer sebelumnya) ditambah dengan bobot pada layer kedua $b[2]$ hingga mendapatkan nilai $a[2]$ dengan menghitung dari fungsi aktivasi $g(z[2])$. Maka $a[2]$ adalah *output* pada jaringan.

2.3. *DenseNet 121*

Dense convolutional network (DenseNet) sebagai sebuah arsitektur jaringan saraf tiruan yang dikembangkan untuk meningkatkan aliran informasi antar layer-layer dalam jaringan.

Dalam *DenseNet* setiap layer menerima *input* dari semua layer sebelumnya, berbeda dengan arsitektur konvensional yang hanya menghubungkan *output* layer sebelumnya ke *input* layer selanjutnya. Hal ini dilakukan dengan menggunakan koneksi langsung dari setiap layer ke semua layer berikutnya, memungkinkan aliran informasi yang lebih langsung dan lebih kuat melalui jaringan.

Arsitektur *DenseNet* menggunakan koneksi yang padat (*dense connectivity*), di mana setiap layer menerima *input* dari semua layer sebelumnya melalui konkatenasi dari *feature-maps* yang dihasilkan di setiap layer. Dengan demikian, aliran informasi dari *input* awal langsung terhubung ke setiap layer dalam jaringan, memungkinkan jaringan untuk memanfaatkan fitur-fitur yang dihasilkan pada berbagai tingkatan dalam proses pembelajaran.

Dalam *DenseNet* setiap layer diimplementasikan sebagai sebuah fungsi komposit yang terdiri dari tiga operasi secara berurutan: *batch normalization* (BN), *rectified linear unit* (ReLU), dan konvolusi 3x3. Selain itu, untuk memfasilitasi perubahan ukuran dari *feature-maps*, *DenseNet* menggunakan blok-blok padat yang dipisahkan oleh lapisan transisi yang melakukan konvolusi dan *pooling* untuk menurunkan ukuran *feature-maps*.

Arsitektur *DenseNet 121*, digambarkan melalui tabel 2.1:

Tabel 2. 1 *Arsitektur DenseNet 121*

Layers	Output Size	<i>DenseNet-121</i>
<i>Convolution</i>	112 x 112	7 x 7 conv
<i>Pooling</i>	56 x 56	3 x 3 max pool
<i>Dense Block 1</i>	56 x 56	$\begin{bmatrix} 1 \times 1 \\ 3 \times 3 \end{bmatrix} \times 6$
<i>Transition Layer 1</i>	56 x 56	1 x 1 conv
	28 x 28	2 x 2 average pool
<i>Dense Block 2</i>	28 x 28	$\begin{bmatrix} 1 \times 1 \\ 3 \times 3 \end{bmatrix} \times 12$
<i>Transition Layer 2</i>	28 x 28	1 x 1 conv
	14 x 14	2 x 2 average pool
<i>Dense Block 3</i>	14 x 14	$\begin{bmatrix} 1 \times 1 \\ 3 \times 3 \end{bmatrix} \times 24$
<i>Transition Layer 3</i>	14 x 14	1 x 1 conv
	7 x 7	2 x 2 average
<i>Dense Block 4</i>	7 x 7	$\begin{bmatrix} 1 \times 1 \\ 3 \times 3 \end{bmatrix} \times 16$
<i>Classification Layer</i>	1 x 1	7 x 7 global average pool
		1000D fully-connected, softmax

Dari arsitektur tersebut bisa disimpulkan bahwa *DenseNet 121* mempunyai layer sebagai berikut:

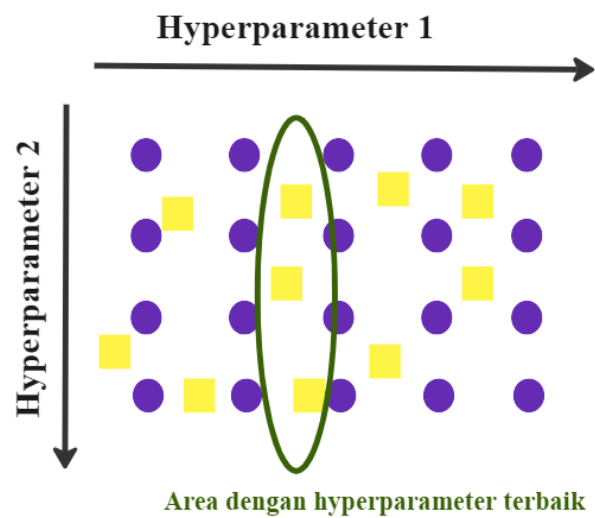
- 1 layer 7 x 7 convolution
- 58 layer 3 x 3 convolution
- 61 layer 1 x 1 convolution
- 4 layer AvgPool
- 1 layer Fully Connected Layer

Singkatnya *DenseNet-121* memiliki 120 layer Convolution dan 4 layer AvgPool.

2.4. *Random search Optimization*

Random search, seperti yang diperkenalkan oleh Bergstra dan Bengio pada tahun 2012, merupakan peningkatan dasar dari *grid search* (Bergstra dkk., 2012). Ini melibatkan eksplorasi secara acak terhadap *hyperparameter* dalam distribusi tertentu dari nilai-nilai parameter yang mungkin. Proses eksplorasi ini berlanjut hingga predetermined budget habis atau hingga tingkat akurasi yang diinginkan tercapai. Meskipun mirip dengan *grid search*, *random search* terbukti menghasilkan hasil yang lebih baik karena dua keunggulan utama yang disorot oleh Maladkar pada tahun 2018 (Kishan Maladkar, 2018) :

1. *Random search* memungkinkan penugasan budget secara independen berdasarkan distribusi dari ruang pencarian. Sebaliknya, dalam *grid search*, budget untuk setiap set *hyperparameter* adalah nilai tetap $B1/N$, di mana B adalah total anggaran dan N adalah jumlah *hyperparameter*. Oleh karena itu, *random search* cenderung memberikan hasil yang lebih baik, terutama ketika beberapa *hyperparameter* memiliki distribusi yang tidak merata. Pola pencarian ini menunjukkan bahwa *random search* memiliki kemungkinan lebih tinggi untuk menemukan konfigurasi optimal dibandingkan dengan *grid search*.
2. Meskipun kemungkinan untuk mendapatkan hasil optimal melalui *random search* tidak dijamin, investasi waktu yang lebih besar secara signifikan meningkatkan probabilitas untuk menemukan set *hyperparameter* terbaik. Konsep ini sejalan dengan teknik Monte Carlo, yang populer digunakan saat menangani *dataset* yang luas dalam skenario *optimized learning* multi-dimensi, sebagaimana yang dijelaskan oleh Harrison pada tahun 2010 (Harrison, 2009). Sebaliknya, untuk *grid search*, durasi pencarian yang lebih lama tidak selalu menjamin hasil yang lebih baik. Selain itu, *random search* menawarkan keuntungan dalam paralelisasi yang mudah dan alokasi sumber daya yang fleksibel, seperti yang ditekankan oleh Krivulin *et al.* pada tahun 2005 (Krivulin dkk., 2005).



Gambar 2. 5 *Random Search Optimization*

Mengambil rentang yang memungkinkan dan mengambil sampel dari rentang itu sesuai distribusi probabilitas. Itu sebabnya titik kuning pada gambar tersebut menyebar secara acak.

Misal:

1. *Learning rate* dalam rentang 0,001 sampai dengan 0,1
2. *Batch size* memiliki nilai 2^n dengan begitu 8,16,32,64

Maka:

$$P(\theta_1) = \text{LogUnifrom}(0,001; 0,1)$$

$$P(\theta_2) = \text{discreteValue}(8, 16, 32, 64)$$

Evaluasi:

$$\theta \sim p(\theta_1, \theta_2)$$

2.5. Image Augmentation

Image Data Augmentation merujuk pada serangkaian teknik atau metode yang digunakan dalam pengolahan citra atau pengenalan gambar untuk meningkatkan jumlah dan variasi data pelatihan. Tujuannya adalah untuk memperkenalkan invarian translasional ke dalam *dataset*, sehingga model pembelajaran mesin, khususnya model *optimized learning* dapat memperoleh

ketangguhan yang lebih baik terhadap variasi dan tantangan dalam data dunia nyata. Teknik augmentasi ini mencakup transformasi gambar seperti *flipping horizontal*, augmentasi ruang warna, *cropping* acak, serta berbagai teknik lainnya yang secara efektif meningkatkan keanekaragaman *dataset* untuk melatih model dengan lebih baik. *Image* augmentasi data menjadi penting terutama ketika pengumpulan dataset yang besar atau variasi data yang alami sulit dicapai. (Shorten & Khoshgoftaar, 2019).

Berdasarkan dokumentasinya pada *TensorFlow* (Keras Teams, 2024) berikut penjelasan tentang detail augmentasi:

2.5.1 *Rescale (scale_factor)*

Fungsi *rescale* mengubah skala nilai piksel dalam gambar dengan mengalikan setiap piksel dengan faktor skala. Nilai piksel yang dihasilkan kemudian menjadi dalam rentang baru. Faktor skala yang digunakan untuk mengalikan nilai piksel. Biasanya, nilai ini adalah bilangan pecahan (*float*) antara 0 dan 1.

2.5.2 *Rotation Range*

Fungsi ini mengatur rentang rotasi gambar secara acak. Setiap gambar yang dimuat dapat dirotasi dalam rentang ini secara acak. Rentang rotasi dalam derajat. Misalnya, jika *rotation_range=20*, gambar dapat dirotasi secara acak antara -20 dan 20 derajat.

2.5.3 *Width Shift Range (Fraction)*

Memungkinkan untuk menggeser gambar *horizontal* secara acak saat augmentasi data. Ini membantu model untuk belajar dari variasi posisi objek dalam gambar. Fraksi dari lebar gambar untuk menentukan rentang geser *horizontal*. Misalnya, jika *width_shift_range=0,2*, gambar dapat digeser secara acak antara -0,2 dan 0,2 kali lebar gambar.

2.5.4 *Height Shift Range (Fraction)*

Menggeser gambar secara vertikal (ke atas atau ke bawah) secara acak selama augmentasi data. Parameter ini memungkinkan model untuk belajar dari variasi posisi objek dalam gambar, bahkan jika objek berada di tempat yang berbeda di dalam gambar.

2.5.5 *Shear Range*

Memungkinkan untuk menerapkan transformasi *shear* atau perataan pada gambar saat augmentasi data. Transformasi *shear* dapat merubah bentuk atau sudut objek dalam gambar, yang dapat membantu model dalam memahami variasi deformasi geometris yang mungkin terjadi pada objek. Nilai yang dapat berupa *float*, yang menentukan rentang perataan. Misalnya, jika *shear_range=0,2*, gambar dapat mengalami perataan acak antara -0,2 dan 0,2.

2.5.6 *Zoom Range*

Memungkinkan untuk menerapkan transformasi *zoom* atau perbesaran pada gambar saat augmentasi data. Transformasi *zoom* dapat membantu model untuk belajar dari variasi ukuran objek dalam gambar. Nilai yang dapat berupa *float* atau tuple/list yang berisi dua *float*. Jika value adalah *float*, itu menentukan rentang *zoom* acak. Misalnya, jika *zoom_range=0,2*, gambar dapat mengalami *zoom* acak antara $[1-0,2; 1+0,2]$ atau $[0,8; 1,2]$. Jika value adalah tuple atau list, elemen pertama adalah faktor *zoom* minimum, dan elemen kedua adalah faktor *zoom* maksimum.

2.5.7 *Horizontal Flip*

Memungkinkan untuk menerapkan transformasi *flip horizontal* (pemutaran gambar secara *horizontal*) saat augmentasi data. Transformasi ini dapat membantu model untuk belajar dari variasi orientasi *horizontal* objek dalam gambar. Jika diatur sebagai “*True*”, gambar secara acak dapat di-*flip* secara *horizontal* (*mirrored*) saat augmentasi data.

2.5.8 Fill mode

Digunakan untuk menentukan strategi pengisian piksel saat terjadi transformasi seperti pergeseran, rotasi, atau *zoom*, yang dapat menyebabkan piksel baru diperlukan di bagian tepi gambar.

- a) constant: Mengisi piksel yang baru muncul dengan nilai konstan yang dapat ditentukan menggunakan *parameter cval*.
- b) nearest: Mengisi piksel yang baru muncul dengan nilai piksel terdekat dari gambar yang sudah ada.
- c) reflect: Mengisi piksel yang baru muncul dengan cara merefleksikan piksel di sepanjang tepi gambar.
- d) wrap: Mengisi piksel yang baru muncul dengan cara melingkupi gambar (wrap around).

2.6. Penelitian yang Relevan

Berikut ini adalah penelitian sebelumnya yang berkaitan dengan penelitian yang dilakukan, yakni:

1. Pada jurnal *Random search for Hyperparameter Optimization* (Bergstra dkk., 2012) Penelitian ini bertujuan untuk membandingkan strategi pencarian parameter optimal dalam proses optimasi *hyperparameter* untuk model-model *optimized learning*. Mereka membandingkan antara pencarian grid (*grid search*) dan pencarian acak (*random search*) serta teknik pencarian manual dalam menentukan parameter terbaik untuk jaringan saraf dan jaringan probabilitas dalam jaringan *Optimized Belief Networks*. Hasil penelitian ini menunjukkan bahwa secara empiris dan teoritis, percobaan yang dipilih secara acak lebih efisien dalam mengoptimalkan parameter-parameter untuk model-model *optimized learning* daripada percobaan pada grid. Mereka membandingkan hasil dari metode *grid search* dan manual *search* dengan pencarian acak, dan menemukan bahwa pencarian acak dalam domain yang sama mampu menemukan model yang sama baik atau bahkan lebih baik dalam sebagian kecil waktu komputasi. Hal ini terjadi karena pencarian acak mampu secara efektif mengeksplorasi ruang konfigurasi yang lebih luas, meskipun kurang menjanjikan. Penelitian ini juga menunjukkan bahwa analisis Gaussian

process dari hubungan antara *hyperparameter* dengan performa pada data validasi mengungkap bahwa hanya beberapa *hyperparameter* yang penting untuk sebagian besar *dataset*. Hal ini membuat pencarian grid menjadi pilihan yang kurang baik dalam mengkonfigurasi algoritma untuk *dataset* baru karena berbagai *hyperparameter* yang berbeda penting pada *dataset* yang berbeda.

2. Pada jurnal *Hyperparameter Optimization and Regularization on Fashion-MNIST Classification* (Greeshma & Sreekumar, 2019) terhadap *hyperparameter* pada data *Fashion-MNIST* yang dilakukan oleh Greeshma K V, Sreekumar K, menggunakan CNN2 dan mendapatkan akurasi maksimum di 93,99% dengan *hyperparameter* Adam, BS-64, dan *Epoch*-60. Penelitian ini melakukan perbandingan beberapa kombinasi *optimizer*, *batch size* dan *epoch* untuk menemukan kombinasi paling optimal. Lalu menggunakan 1 kombinasi yang terbaik untuk digunakan dalam penerapannya.
3. Pada Jurnal *Improving stroke diagnosis accuracy using hyperparameter optimized optimized learning* (Badriyah dkk., 2019) ini membahas tentang penggunaan teknik *optimized learning* untuk mendeteksi *stroke* menggunakan *Computerized Tomography* (CT) scan. Fokus utamanya adalah pada optimisasi *hyperparameter* dalam *optimized learning* dengan membandingkan dua pendekatan, yaitu *Random search* dan *Bayesian Optimization*, untuk menemukan kombinasi *hyperparameter* yang optimal. Penelitian ini menggunakan gambar CT scan yang diproses dengan langkah-langkah seperti penskalaan, konversi ke skala abu-abu, perataan (smoothing), thresholding, dan operasi morfologi. Selanjutnya, fitur-fitur dari gambar diambil menggunakan Gray Level Co-occurrence Matrix (GLCM). Dilakukan seleksi fitur untuk mengurangi biaya komputasi, dan *optimized learning* digunakan untuk klasifikasi data berdasarkan pengaturan *hyperparameter*. Hasil eksperimen menunjukkan bahwa *Random search* memberikan akurasi terbaik. Hasil optimisasi *hyperparameter* menggunakan *Random search* menghasilkan akurasi sebesar 1.0 atau 100%. Kombinasi *hyperparameter* terbaik yang menghasilkan akurasi sempurna tersebut adalah menggunakan fungsi aktivasi Tanh, ukuran batch 64, dropout rate 0,5, tingkat pembelajaran 0,004, 90 *epoch*, 7 lapisan tersembunyi, dan 200 node tersembunyi.

BAB III

ANALISIS DAN PERANCANGAN SISTEM

3.1. Analisis Sistem

Menganalisis sistem merupakan langkah penting dalam penelitian yang bertujuan untuk mengidentifikasi komponen-komponen yang diperlukan agar sistem dapat beroperasi secara efektif. Analisis sistem dapat dibagi menjadi dua jenis, yaitu analisis masalah untuk mengidentifikasi penyebab dan dampak suatu masalah, serta analisis kebutuhan untuk merinci data dan proses yang dibutuhkan dalam perancangan sistem.

3.1.1. Analisis Masalah

Dalam rangka penelitian ini, analisis masalah bertujuan untuk mengidentifikasi akar penyebab permasalahan, khususnya terkait dengan pengklasifikasian wajah penderita *thalassemia* dan wajah orang sehat agar penanganan medis bisa dilakukan dengan cepat dan tepat. Proses analisis masalah ini akan mendalam, dengan menggunakan metode Ishikawa diagram.

Ishikawa diagram, yang juga dikenal sebagai diagram tulang ikan, adalah alat visual yang membentuk struktur serupa tulang ikan untuk menunjukkan hubungan sebab-akibat dari suatu permasalahan. Struktur diagram ini terdiri dari kepala ikan yang berisi judul permasalahan utama yang akan diurai, serta tulang-tulang ikan yang menggambarkan kategori penyebab yang mungkin, seperti manusia, sistem, material, dan metode. Dalam penelitian ini, diagram Ishikawa yang dihasilkan dapat ditemukan dalam gambar 3.1.



Gambar 3. 1 Ishikawa Diagram

Gambar 3.1 memperlihatkan *Ishikawa diagram* yang memuat akar penyebab dari permasalahan yang ingin diselesaikan dalam penelitian ini.

3.1.2. Analisis Kebutuhan

Dalam konteks perancangan sistem, analisis kebutuhan menjadi langkah kritis untuk mengidentifikasi secara menyeluruh data dan proses yang diperlukan. Analisis ini melibatkan penentuan kebutuhan fungsional dan non-fungsional yang menjadi landasan utama dalam merancang sistem, sehingga sistem yang dikembangkan dapat optimal dalam memenuhi tujuan yang telah ditetapkan.

3.1.2.1. Kebutuhan Fungsional

Kebutuhan fungsional adalah kebutuhan yang berisi proses-proses apa saja yang nantinya dilakukan oleh sistem. Kebutuhan fungsional yang dibutuhkan dalam sistem ini adalah:

1. Sistem dapat berjalan dengan pada Android.
2. Sistem dapat mengambil gambar dari kamera Android maupun penyimpanan internal Android.
3. Sistem harus mampu mengenali dan mengklasifikasikan wajah sebagai indikator potensial penderita *Beta Thalassemia Mayor*.

4. Sistem dapat menampilkan hasil confidence dari persentase wajah penderita *thalassemia* dan wajah orang sehat.
5. Proses optimalisasi *hyperparameter batch size* dan *learning rate* pada model *convolutional neural network* (CNN) harus dilakukan untuk meningkatkan akurasi klasifikasi.
6. Model klasifikasi wajah harus diimplementasikan pada platform aplikasi Android untuk memungkinkan penggunaan yang lebih mudah dan aksesibilitas yang lebih luas.

3.1.2.2. Kebutuhan Non-Fungsional

Kebutuhan *non-fungsional* merupakan aspek-aspek seperti fitur, karakteristik, batasan-layanan, atau fungsi tambahan dalam sistem, seperti batasan waktu, proses pengembangan, dan standarisasi sistem yang menjadi pelengkap. Di bawah ini tercantum kebutuhan fungsional yang menjadi kebutuhan utama dalam perancangan sistem ini:

1. Batasan gambar yang bisa dideteksi hanya wajah penderita *thalassemia beta mayor* dan wajah orang sehat
2. Tampilan antar muka pada sistem mudah dipahami sehingga mudah digunakan oleh pengguna.
3. Tidak memerlukan perangkat tambahan sehingga tidak membutuhkan biaya yang besar.
4. Sistem memberikan respons yang cepat dalam pengenalan wajah penderita β TM untuk mendukung intervensi medis yang segera.

3.1.3. Diagram Umum Sistem

Diagram umum sistem adalah visualisasi dari alur sistem yang menggambarkan proses, alur, dan interaksi antara komponen-komponen di dalam suatu sistem. Rancangan keseluruhan aplikasi dijelaskan melalui gambar yang disajikan di gambar 3.2.



Gambar 3. 2 Diagram Umum Sistem

Berikut penjelasan alur proses diagram umum sistem pada gambar 3.2:

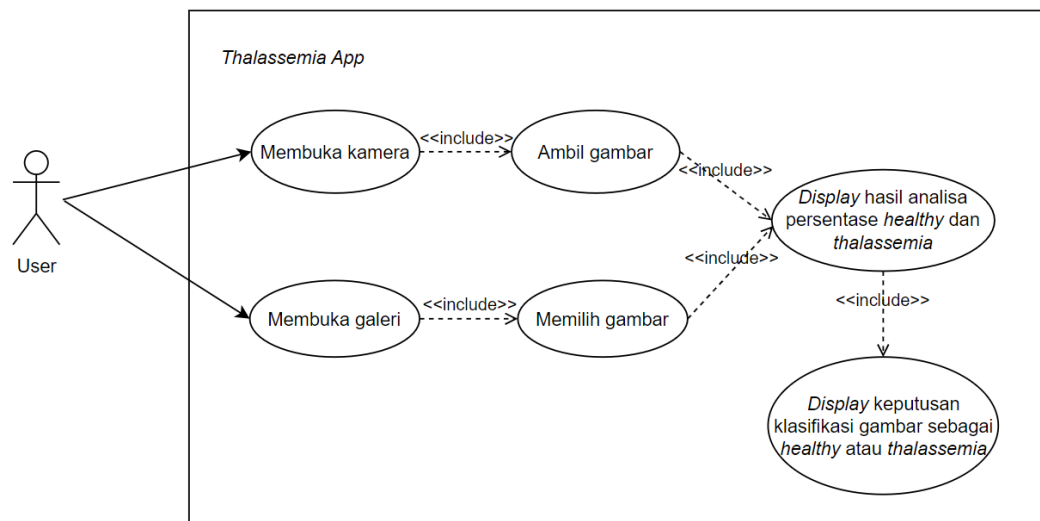
1. *User* menginputkan data baik dari kamera maupun dari penyimpanan internal Android
2. Aplikasi di Android mengirimkan data gambar tersebut ke model yang sudah di muat di Android
3. Model melakukan prediksi berdasarkan data yang diberikan
4. *Output* diberikan kepada pengguna

3.2. Pemodelan Sistem

Pemodelan sistem adalah proses penjelasan langkah-langkah interaksi antara pengguna dan aplikasi yang dirancang untuk memastikan optimalitas sistem. Pemodelan sistem pada penelitian ini direpresentasikan dalam bentuk *Unified Modeling Language* (UML), yang merupakan bahasa pemodelan umum untuk menggambarkan hubungan antar komponen di dalam suatu sistem, memungkinkan interaksi melalui pengguna. Dalam penelitian ini, model UML yang diterapkan mencakup *use case diagram*, *activity diagram*, dan *sequence diagram*.

3.2.1. Use Case Diagram

Use case diagram adalah bentuk pemodelan yang memperinci interaksi antara aktor-aktor dan sistem yang sedang dikembangkan. Gambar 3.3 adalah diagram use case yang menggambarkan interaksi antara aktor dan sistem dalam konteks sistem yang sedang dibangun.

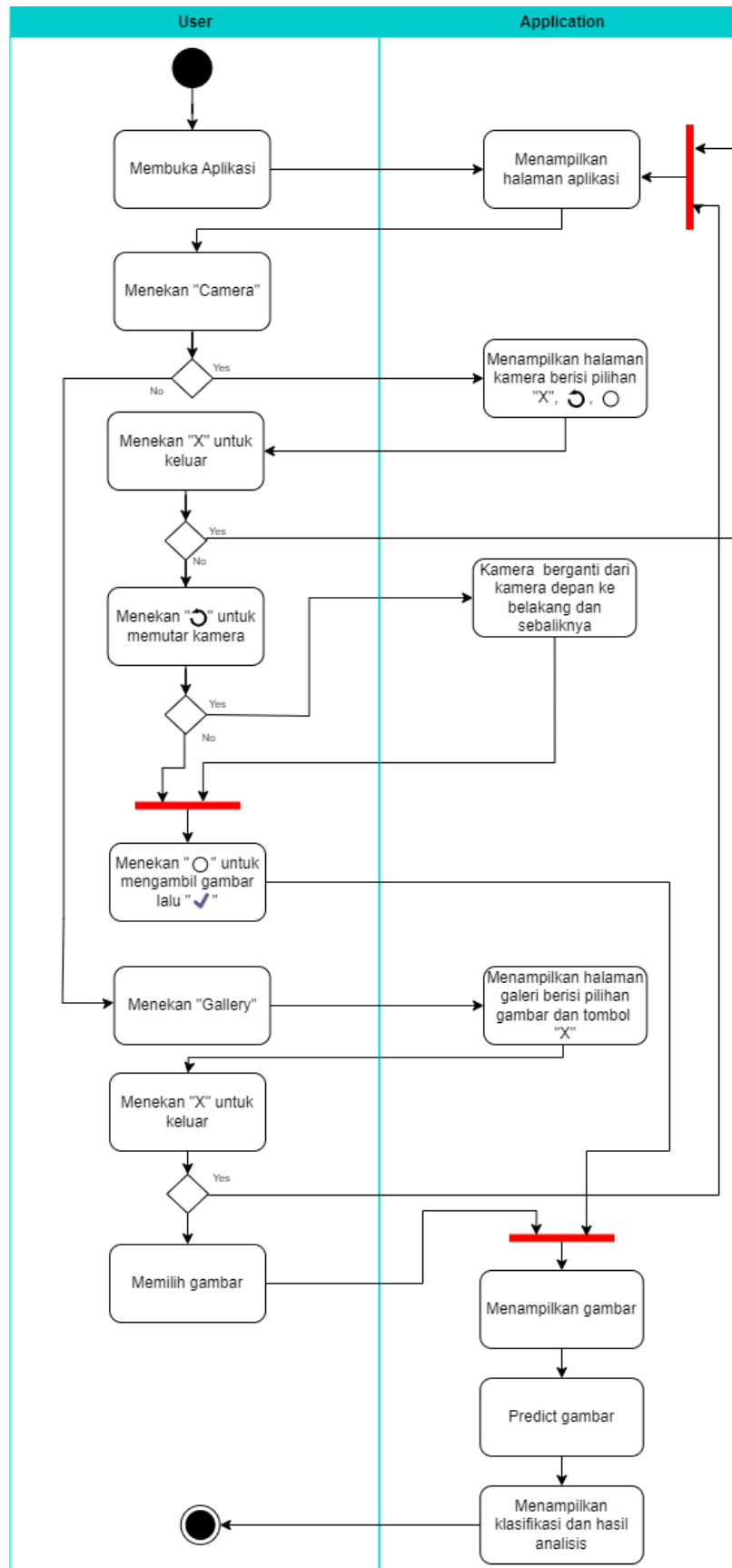


Gambar 3. 3 Use Case Diagram

Gambar 3.3 menunjukkan hubungan interaksi antara aktor yakni *user* dengan sistem. Saat masuk kedalam aplikasi *user* memiliki dua akses yaitu membuka kamera dan membuka galeri. Saat kamera dibuka *user* bisa mengambil gambar dan langsung tampil hasil analisis dan pengklasifikasiannya. Saat *user* membuka galeri maka *user* bisa memilih gambar dari penyimpanan internal setelah itu tampil hasil analisis dan pengklasifikasian.

3.2.2. Activity Diagram

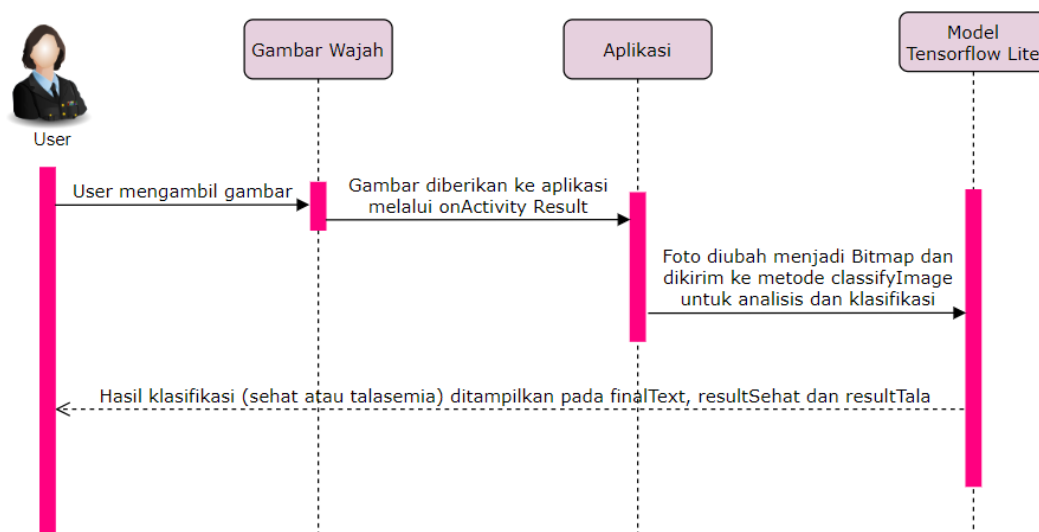
Activity Diagram merupakan visualisasi dari alur kerja atau proses aktivitas dalam suatu sistem, mulai dari inisiasi hingga penyelesaian. Diagram Aktivitas juga berguna untuk menggambarkan segmen komponen dari diagram use case. Seperti pada gambar 3.4 dapat dilihat bahwa *user* pertama sekali membuka aplikasi lalu ditampilkan tampilan utama dari aplikasi. *User* bisa memilih untuk mengambil gambar dari kamera atau galeri. Jika dari kamera maka ada pilihan “X” untuk Kembali ke halaman utama, ada pilihan memutar kamera dan mengambil gambar. Jika sudah diambil maka gambar ditampilkan di halaman utama. Namun pada saat *user* memilih galeri, *user* diberikan pilihan “X” untuk kembali atau melanjutkan pemilihan gambar. Gambar yang dipilih ditampilkan pada menu utama. Aplikasi melakukan predict terhadap gambar baru yang diberikan. Dengan cepat result tampil dari hasil analisis itu maka ditampilkan juga pengklasifikasiannya.



Gambar 3. 4 Activity Diagram

3.2.3. Sequence Diagram

Diagram urutan (sequence diagram) adalah jenis diagram dalam Unified Modeling Language (UML) yang menggambarkan interaksi antar objek dengan memperhatikan urutan waktu di dalam sistem. Penggunaan panah penuh pada diagram urutan menunjukkan hubungan atau interaksi antar objek, sementara penggunaan garis putus-putus mengindikasikan respon atau tanggapan terhadap *input* yang diberikan oleh pengguna.

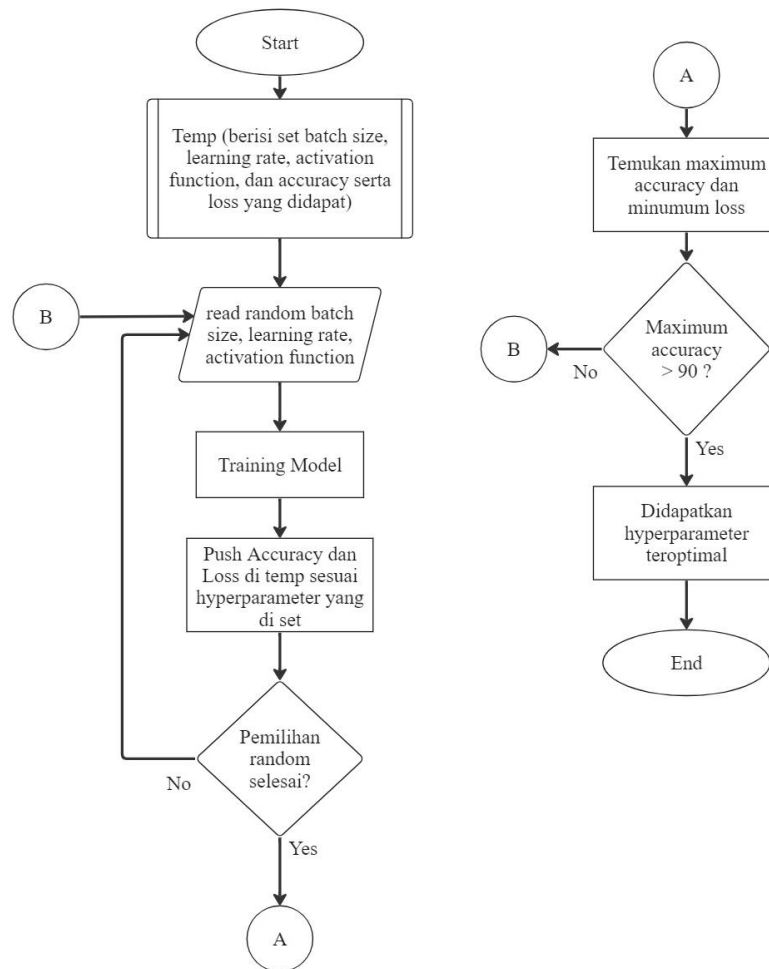


Gambar 3. 5 Sequence Diagram

Pada *sequence* diagram gambar 3.5, terdapat 4 komponen yang saling berinteraksi. *Sequence diagram* di atas menjabarkan proses dari awal *user* melakukan pengambilan gambar wajah. Gambar diberikan ke aplikasi melalui *onActivityResult* Dimana gambar diubah menjadi *Bitmap* dan dikirim ke metode *classifyImage* untuk analisis dan klasifikasi. Hasil klasifikasi (sehat atau *thalassemia*) ditampilkan pada *finalText*, *resultSehat* dan *resultTala*.

3.3. Flowchart

Flowchart atau diagram alir adalah representasi grafis atau simbolik yang menggambarkan urutan proses dengan bentuk simbol-simbol dan hubungan antara suatu proses dengan proses lainnya dalam bentuk tanda panah di dalam suatu sistem.



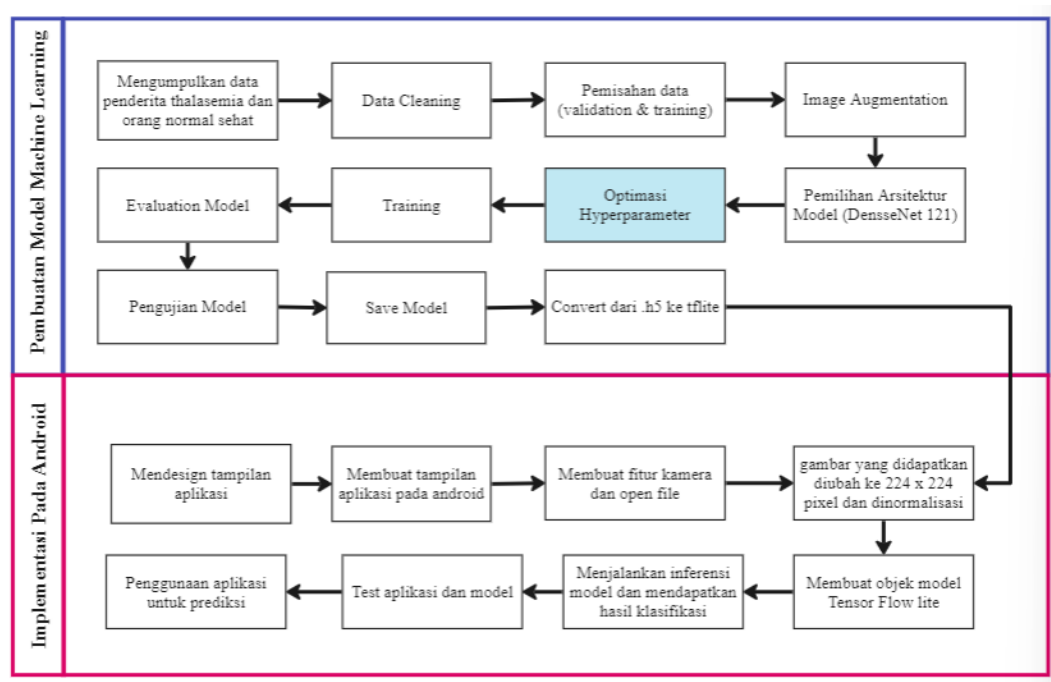
Gambar 3. 6 Flow Chart Random search

Pada *flowchart* gambar 3.6, diberikan langkah dalam proses penentuan *hyperparameter*. Yang pertama menetapkan satu variabel untuk menyimpan nilai random yang kita set pada *batch size*, *learning rate*, *activation function* serta *accuracy* dan *loss*. Kemudian model di train, hasil dari *training* di simpan pada temp. selanjutnya dilakukan pengecekan apakah random yang dilakukan sudah cukup atau belum jika belum maka lakukan random lagi. Jika sudah cukup

didapatkan *accuracy* paling maksimum dan *loss* paling minimum sebagai model terbaik. Setelah ditemukan yang terbaik lihat apakah maximum *accuracy* > 90%. Jika belum maka lakukan random lagi. Jika sudah sesuai maka didapatkan model dan *hyperparameter* yang sesuai

3.4. Work Flow

Work flow atau alur kerja adalah urutan langkah-langkah atau serangkaian tindakan yang dirancang untuk menyelesaikan tugas tertentu, proses, atau aktivitas. Ini menggambarkan serangkaian langkah yang harus diikuti untuk menyelesaikan suatu pekerjaan atau mencapai tujuan tertentu. Berikut ini *work flow* yang digunakan:



Gambar 3. 7 Work Flow

Pada pembuatan model *machine learning* pertama sekali dilakukan adalah Gathering data dikumpulkan data dari *google image* yang terdiri dari 2 class yaitu gambar wajah penderita *thalassemia beta mayor* dan gambar wajah orang normal sehat, kemudian dilakukan *data cleaning* dengan pemilihan data yang sesuai dan tidak sesuai (relevan). Kemudian data di split menjadi data untuk *validation* dan data untuk *training*. Jumlah data yang terkumpul untuk data *image training* adalah

354 mencakup 2 kelas. Sedangkan untuk *data validation* sebanyak 152 gambar mencakup 2 kelas. Dengan arsitektur yang dipilih dalam hal ini *DenseNet 121* dilakukan optimasi *hyperparameter* dengan *random search optimization* setelah itu didapatkan hasil *training* yang paling optimal. Lakukan evaluasi model dan uji model tersebut. Setelah itu *save* model. *Convert* model yang di *save* dalam bentuk .h5 ke bentuk tf lite

Dalam pembuatan aplikasi Android, pertama sekali dibuat desain aplikasi lalu implementasikan design yang sudah dibuat ke dalam codingan Android. Mengaktifkan fitur kamera dan open file pada penyimpanan Android. Gambar yang didapatkan diubah menjadi ukuran 224 x 224 pixel dan dilakukan normalisasi untuk pixel implementasi konversi nilai piksel dari format RGB ke format *float* dalam rentang 0 hingga 1. Integrasikan model *machine learning* ke dalam Android dan lakukan test aplikasi dan model. Saat test selesai aplikasi dapat digunakan untuk prediksi.

3.5.1 *Data Gathering*

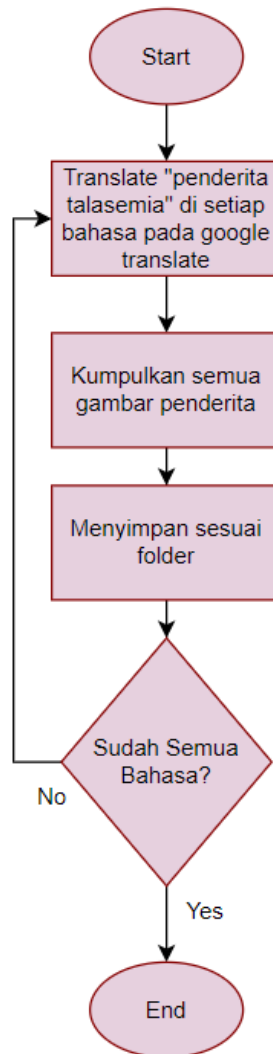
Proses pengumpulan data dari *Google images* dan media social dengan memastikan data yang diperoleh akurat, sah, dan sesuai dengan tujuan *training*.

Penggunaan pencarian efektif yang dilakukan peneliti melalui:

1. *Google image*
 - a) Menggunakan kata kunci yang spesifik untuk menghindari hasil yang terlalu umum
 - b) Menggunakan kata kunci “penderita *thalassemia*” dalam beberapa negara yang ada di *google translate* kemudian mengumpulkan gambarnya
2. Media Sosial
 - a) Menggunakan *hashtag* untuk mencari konten terkait di platform seperti *Instagram*, *Twitter* dan *Youtube*
 - b) Memanfaatkan fitur pencarian lanjutan di setiap *platform*
 - c) Melihat melalui profil dan daftar teman untuk mengidentifikasi sumber-sumber potensial.

Peneliti memvalidasi sumber data yang dapat diandalkan dan memiliki kredibilitas. Menghindari pengumpulan data yang melanggar privasi orang lain. Kemudian menyimpan data dengan cara yang terorganisir, sehingga mudah di akses

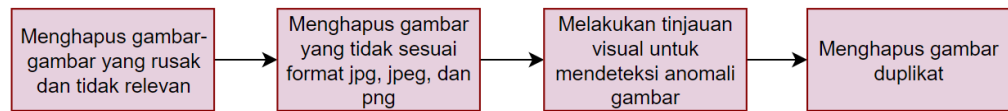
dan dikelola. Data dikelompokkan sesuai dengan kelasnya masing-masing. Untuk lebih jelasnya perhatikan alur *data gathering* dibawah pada gambar 3.8 dibawah:



Gambar 3. 8 *Flow Data Gathering*

3.5.2 Data Cleaning

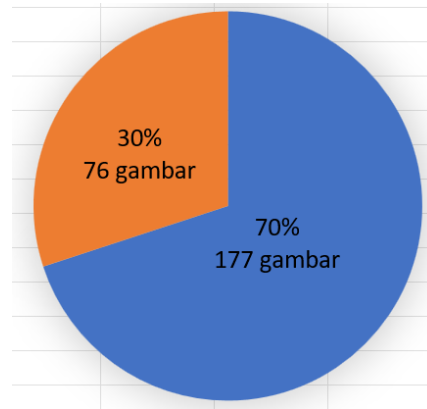
Pada proses ini dilakukan penghapusan gambar-gambar yang rusak, tidak relevan. Dilanjutkan dengan tinjauan visual beberapa gambar untuk mendeteksi masalah atau anomaly gambar yang terjadi. Mendeteksi duplikat gambar. Memperhatikan format gambar yang dimiliki. Untuk lebih detail lihat gambar alur pada gambar 3.9.



Gambar 3. 9 *Flow Data Cleaning*

3.5.3 Split Data

Data telah dibagi menjadi dua subset, yaitu data pelatihan (*training*) dan data validasi, dengan proporsi 70% untuk pelatihan dan 30% untuk validasi. Total data gambar yang dikumpulkan untuk pelatihan adalah sebanyak 354, yang terdiri dari dua kelas yang berbeda. Setiap gambar dalam subset ini akan digunakan untuk melatih model agar dapat mengenali dan memahami karakteristik masing-masing kelas. Sementara itu, untuk subset data validasi, terdapat sebanyak 152 gambar yang juga mencakup dua kelas. Data validasi berperan penting dalam menguji sejauh mana model mampu melakukan generalisasi pada data yang belum pernah dilihat sebelumnya. Pembagian data ini menjadi pelatihan dan validasi merupakan langkah kritis dalam pengembangan model untuk memastikan performa yang baik dan ketangguhan terhadap data baru. Lihat pada *pie chart* gambar 3.10.



Gambar 3. 10 *Pie Chart Split Data*

Untuk lebih detailnya mengenai jumlah data yang digunakan perhatikan tabel 3.1:

Tabel 3. 1 *Jumlah Data*

Jenis Data	Sehat	<i>Thalassemia</i>
Train	177	177
Val	76	76

3.5.4 Image Augmentation

Image augmentation bertujuan untuk meningkatkan variasi gambar agar model dapat lebih tangguh terhadap data baru yang berbeda dengan data yang digunakan selama pelatihan. Dalam penelitian ini, digunakan beberapa teknik augmentasi, sebagaimana tercantum dalam tabel 3.1 di bawah:

Tabel 3. 2 Image Augmentation

Parameter	Value
<i>rescale</i>	1.0/255
<i>rotation_range</i>	20
<i>Width_shift_range</i>	0,2
<i>Height_shift_range</i>	0,2
<i>Shear_range</i>	0,2
<i>Zoom_range</i>	0,2
<i>Horizontal_flip</i>	True
<i>Fill_mode</i>	'nearest'

Dalam konteks penggunaan `ImageDataGenerator` dengan parameter-parameter yang disebutkan, berikut adalah penjelasan dari masing-masing parameter:

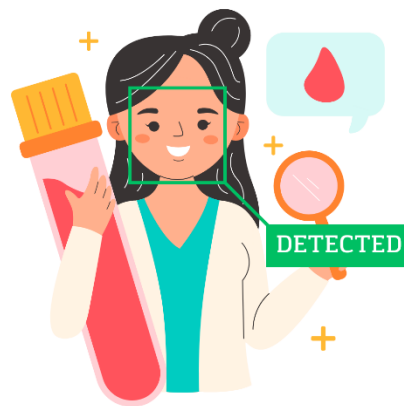
1. *rescale*=1.0/255: Mengubah skala nilai piksel dalam gambar sehingga nilai piksel berada dalam rentang antara 0 dan 1. Ini adalah praktek umum untuk mengoptimalkan kinerja model.
2. *rotation_range*=20: Menentukan rentang rotasi gambar secara acak antara -20 dan 20 derajat. Hal ini membantu model untuk belajar dari variasi orientasi objek dalam gambar.
3. *width_shift_range*=0,2: Mengatur rentang pergeseran *horizontal* secara acak antara -20% dan +20% dari lebar gambar. Ini membantu model belajar dari variasi posisi objek dalam gambar.
4. *height_shift_range*=0,2: Mengatur rentang pergeseran vertikal secara acak antara -20% dan +20% dari tinggi gambar. Ini memungkinkan model belajar dari variasi posisi objek dalam gambar.

5. *shear_range=0,2*: Mengatur rentang perataan (*shear*) acak. Ini dapat merubah bentuk objek dalam gambar.
6. *zoom_range=0,2*: Menentukan rentang *zoom* acak antara $[1-0,2; 1+0,2]$ atau $[0,8; 1,2]$. Ini membantu model belajar dari variasi ukuran objek dalam gambar.
7. *horizontal_flip=True*: Mengaktifkan *flip horizontal* secara acak pada gambar. Ini dapat membantu model belajar dari variasi orientasi *horizontal* objek.
8. *fill_mode='nearest'*: Menentukan metode pengisian piksel yang baru muncul sebagai hasil dari transformasi. Dalam hal ini, nilai piksel baru diisi dengan nilai piksel terdekat dari gambar yang sudah ada.

3.5. Perancangan *Interface*

Perancangan *interface* adalah tahap pembuatan kerangka desain dari sistem yang akan dibangun. Perancangan *interface* diperlukan agar tahap proses pembuatan sistem dapat berlangsung lancar dengan patokan dari rancangan *interface* yang sudah dibuat.

3.5.1 Logo Aplikasi

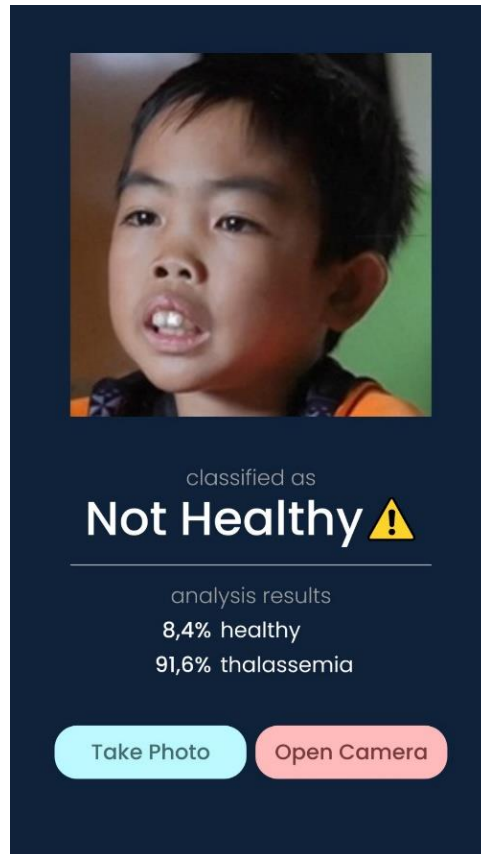


Gambar 3. 11 Logo App

Logo aplikasi dibuat sebagai identitas visual yang khas dan memudahkan pengenalan aplikasi kepada Masyarakat. Logo menjadi elemen desain yang mencerminkan nilai, citra, dan pesan yang ingin disampaikan oleh aplikasi kepada pengguna. Pada logo tersebut di *highlight* bagian wajah menunjukkan

pendeteksian dilakukan pada gambar wajah. Tabung darah yang erat hubungannya dengan penyakit *thalassemia*

3.5.2 Halaman Utama Aplikasi



Gambar 3. 12 Halaman Utama App

Halaman pada gambar di atas hanya bisa diakses oleh *user*. *User* memiliki hak khusus untuk menggunakan fitur yang tersedia. Berikut keterangan dari gambar 3.12 :

1. Fitur kamera untuk *user* mengambil gambar wajah secara langsung
2. Fitur galeri untuk *user* memilih foto langsung dari galeri
3. Tulisan yang langsung muncul jika gambar sudah di tampilkan

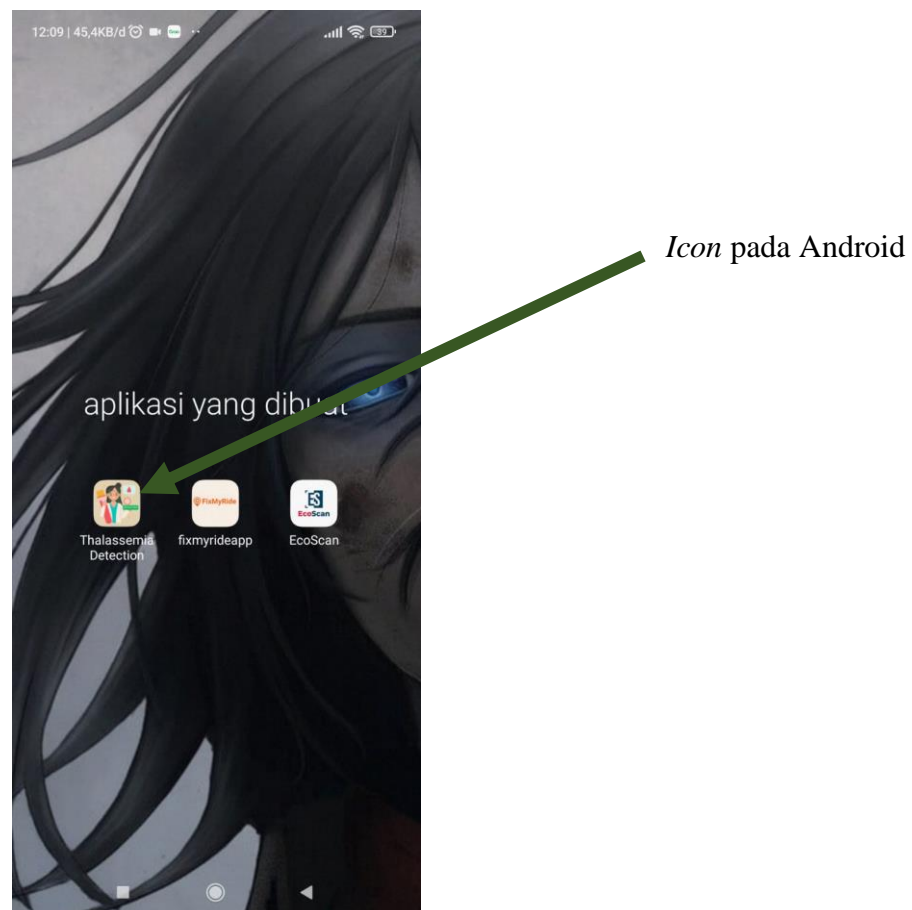
BAB IV

IMPLEMENTASI DAN PENGUJIAN SISTEM

4.1. Implementasi

Pada penelitian yang dilakukan, sistem dibangun dengan mengimplementasikan arsitektur *DenseNet 121* yang di optimasi menggunakan teknik random dengan bahasa pemrograman Python. Model di *training* sampai mencapai hasil yang paling optimal mencapai *accuracy* 97% kemudian di convert ke bentuk .tflite untuk dapat di predict pada Android menggunakan bahasa pemrograman *Java*.

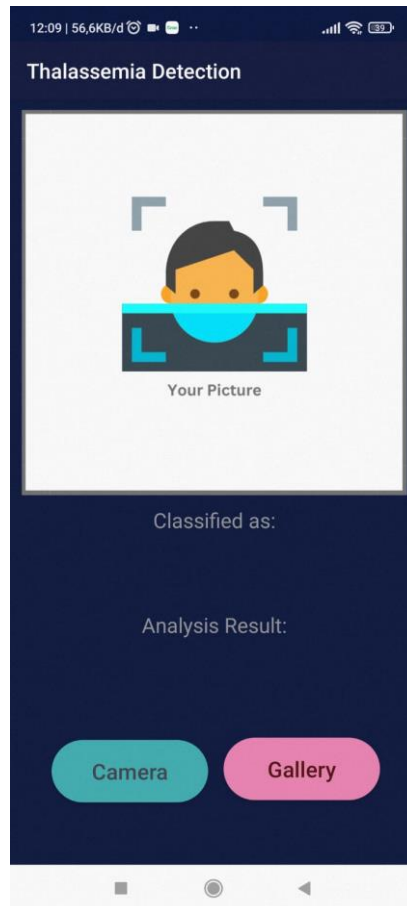
4.1.1. Tampilan Aplikasi pada Android



Gambar 4. 1 Tampilan App pada Android

Saat membuka Android tampilan aplikasi terlihat seperti gambar 4.1, dengan nama *Thalassemia Detection*. Untuk bisa menggunakannya *user* pengguna hanya perlu melakukan klik pada antarmuka yang telah disediakan untuk mengakses fungsionalitas aplikasi.

4.1.2 Halaman Utama



Gambar 4. 2 Halaman Utama

Pada halaman utama ditampilkan *frame* sebagai tempat gambar di letakkan. Terdapat tulisan klasifikasi dan analisis untuk menampilkan hasil dan analisis. Pada bagian paling bawah terdapat 2 *button* yaitu *camera* untuk mengambil gambar langsung dari kamera dan *gallery* untuk mengambil gambar dari penyimpanan internal

4.1.3 Fitur Camera



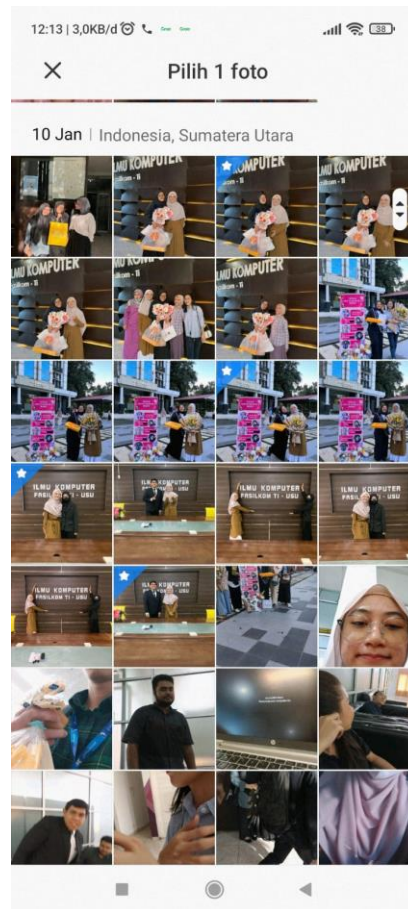
Gambar 4. 3 *Open Camera*



Gambar 4. 4 *Setelah di Capture*

Ketika *user* memilih tombol *camera* maka diarahkan untuk pengambilan gambar seperti gambar 4.3 terdapat 3 pengaturan penting yaitu *X* untuk Kembali ke halaman utama dan membatalkan pengambilan gambar. Lingkaran putih di tengah untuk pengambilan gambar dan terakhir *flip* untuk membalik kamera dari kamera depan ke kamera belakang maupun sebaliknya. Jika sudah di ambil gambar tampilan akan berubah seperti gambar 4.4 terdapat pilihan *checklist* maupun *cross* untuk memvalidasi gambar.

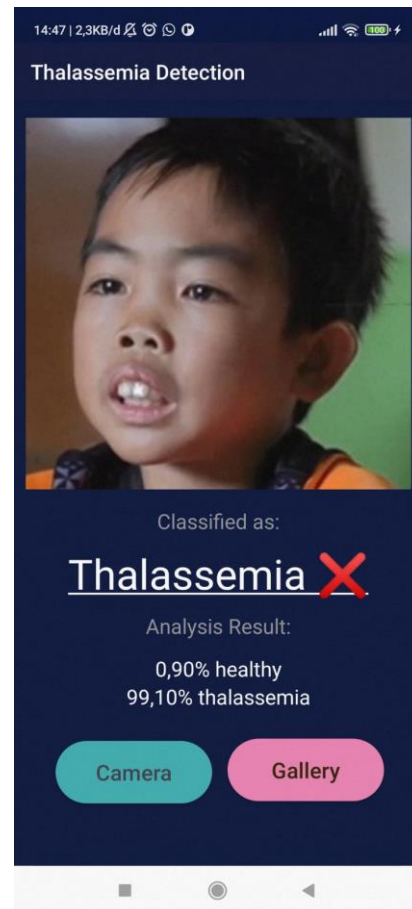
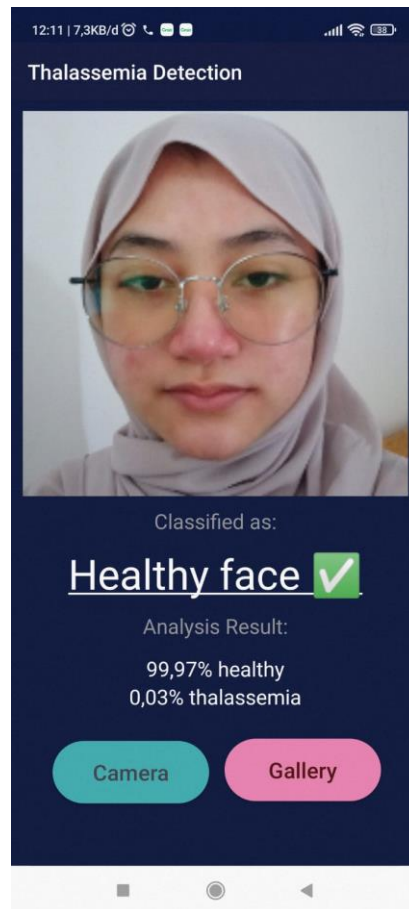
4.1.4 Open Gallery



Gambar 4. 5 Open Gallery

Jika *user* memilih untuk mengambil gambar melalui *gallery*, maka tampilan terlihat seperti gambar 4.5 *user* bisa menyesuaikan gambar wajah yang di ambil.

4.1.5 Result



Gambar 4. 6 Terdeteksi Sehat **Gambar 4. 7** Terdeteksi Thalassemia

Hasil gambar dari *camera* maupun *gallery* yang di ambil *user* ditampilkan kembali pada halaman utama diikuti penjelasan klasifikasi dan hasil analisis. Untuk orang sehat bisa dilihat seperti pada gambar 4.7. Untuk orang yang terindikasi sebagai penderita bisa dilihat pada gambar kanan

4.2. Implementasi *Random Search Optimization*

Impelementasi *Random Search Optimization* pada *hyperparameter* bertujuan mencari kombinasi *hyperparameter* yang memberikan performa terbaik pada model *machine learning* yang dibangun. Melibatkan pengujian sejumlah kombinasi *hyperparameter* secara acak dari rentang nilai yang ditentukan. Proses *Random search* yang dilakukan pada penelitian ini sebanyak 14 *training* dengan kombinasi yang berbeda. Menggunakan *activation function softmax* yang terdiri dari 2 class dan setiap *training* yang dilakukan adalah

50 *Epoch* untuk mencapai hasil maksimal yang masing-masing model bisa berikan. Lihat hasil random pada tabel 4.1:

Tabel 4. 1 Trial Training Model

Trial	Batch size	Learning rate	Accuracy	Val Accuracy	Loss	Val Loss
1	4	0,001	0,9517	0,9079	0,1178	0,2871
2	8	0,0009	0,9574	0,8750	0,1248	0,3070
3	8	0,0001	0,9261	0,8816	0,2632	0,2977
4	8	0,002	0,9773	0,8882	0,0821	0,3086
5	8	0,001	0,9716	0,9342	0,1064	0,2448
6	8	0,03	0,9659	0,9013	0,2779	1,2733
7	8	0,05	0,9517	0,9013	0,3632	2,4675
8	8	0,01	0,9716	0,9145	0,0889	0,5438
9	8	0,1	0,9318	0,9079	1,5154	4,6760
10	16	0,001	0,9545	0,9079	0,1335	0,2571
11	32	0,001	0,9432	0,9013	0,1885	0,2655
12	32	0,05	0,8920	0,7895	0,4991	1,7295
13	64	0,001	0,9290	0,9079	0,2053	0,2674
14	128	0,0001	0,7330	0,7171	0,5877	0,5636

Pada tabel jika dilihat trial yang ketiga menghasilkan validasi *accuracy* yang tertinggi sedangkan untuk validasi *accuracy* yang paling rendah adalah trial kelima. Metrik *val accuracy* memberikan informasi seberapa baik model dapat memprediksi data yang belum pernah dilihat sebelumnya (data validasi). Tujuannya untuk menilai generalisasi model sehingga memberikan gambaran yang lebih objektif tentang kinerja model di luar data pelatihan. Untuk itu disini peneliti menggunakan *validation accuracy* sebagai acuan pemilihan model. Untuk lebih

jelasnya perhatikan gambar 4.8 untuk melihat *validation accuracy* paling tinggi. Dengan menggunakan *batch size* 8 dan berbagai *learning rate*:



Gambar 4. 8 Grafik LR dan Val Accuracy

Dapat dilihat yang mencapai *validation accuracy* tertinggi ada pada *learning rate* 0,001. Sehingga model yang digunakan dalam membangun sistem adalah model trial 3 yang memiliki *accuracy* 97%, *val accuracy* 93%, *loss* 10%, dan *val loss* 24%.

4.3. Perbandingan Optimasi

Perbandingan optimasi dilakukan melalui evaluasi kinerja final model yang telah mengalami penyetelan *hyperparameter*. Pada satu sisi, model tersebut dikonfigurasi menggunakan Adam *Optimizer* dengan *learning rate* yang dioptimalkan sebesar 0,001 dan *batch size* sebesar 8. Sementara itu, pada sisi lain, model dibangun dengan mengadopsi *hyperparameter* default, yaitu *learning rate* pada Adam *Optimizer* tetap 0,001 dan *batch size* 32.

Dengan membandingkan keduanya, tujuan utama adalah untuk mengamati perbedaan dalam performa akhir model ketika menggunakan konfigurasi *hyperparameter* yang dioptimalkan dan default. Analisis perbandingan ini diharapkan dapat memberikan wawasan yang lebih mendalam terkait dampak penyetelan *hyperparameter* terhadap hasil akhir model, serta mengevaluasi apakah

peningkatan performa yang signifikan dapat dicapai melalui proses optimasi tersebut.

Untuk melihat lebih jelas antara perbandingan *accuracy*, *validation accuracy*, *loss*, dan *validation loss* keduanya perhatikan *trial* 5 untuk optimasi dan *trial* 11 untuk *default* dari tabel 4.1, lebih lanjut perbandingan keduanya disajikan tabel 4.2:

Tabel 4. 2 Perbandingan Optimasi dan Default

Hasil Metric	<i>Hyperparameter</i> Optimasi (<i>Batch size</i> = 8, <i>Lr</i> = 0,001)	<i>Hyperparameter</i> Default (<i>Batch size</i> = 32, <i>Lr</i> = 0,001)
<i>Accuracy</i>	0,9716	0,9432
<i>Validation Accuracy</i>	0,9342	0,9013
<i>Loss</i>	0,1064	0,1885
<i>Validation Loss</i>	0,2448	0,2655

Model dengan *hyperparameter* optimasi (*Batch size* = 8, *Lr* = 0,001) menunjukkan hasil yang superior dalam hal *accuracy* (0,9716) dan *validation accuracy* (0,9342) dibandingkan dengan model yang menggunakan *hyperparameter default* (*batch size* = 32, *Lr* = 0,001).

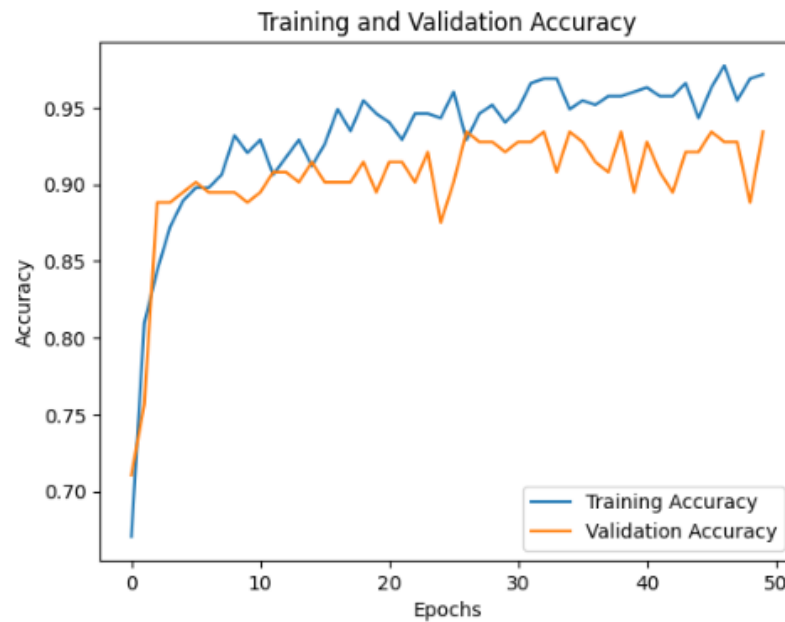
Secara konsisten, model dengan *hyperparameter* optimasi menghasilkan nilai *loss* yang lebih rendah, baik pada data pelatihan maupun data validasi, menunjukkan bahwa model tersebut cenderung lebih baik dalam menangani pola data.

Perbedaan signifikan dalam *accuracy* dan *validation accuracy* menunjukkan bahwa penyetelan *hyperparameter* pada model dengan *batch size* 8 memberikan hasil yang lebih baik dalam mengenali pola-pola yang kompleks pada data pelatihan dan data validasi.

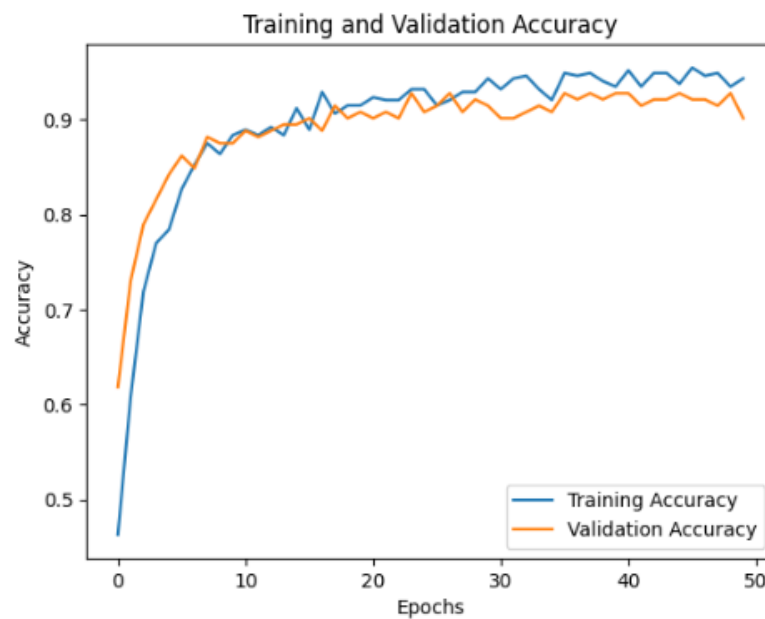
Penyimpangan yang rendah antara *accuracy* dan *validation accuracy* pada model dengan *hyperparameter* optimasi menandakan bahwa model tersebut tidak cenderung mengalami *overfitting* pada data pelatihan.

Dengan demikian, berdasarkan hasil ini, model dengan *hyperparameter* optimasi (*batch size* = 8, *lr* = 0,001) dapat dianggap sebagai pilihan yang lebih baik untuk konfigurasi *hyperparameter* pada penelitian ini.

Untuk melihat dan membandingkan model bisa juga dilihat dari grafik pelatihan yang dilakukan masing-masing sebanyak 50 *epoch*.

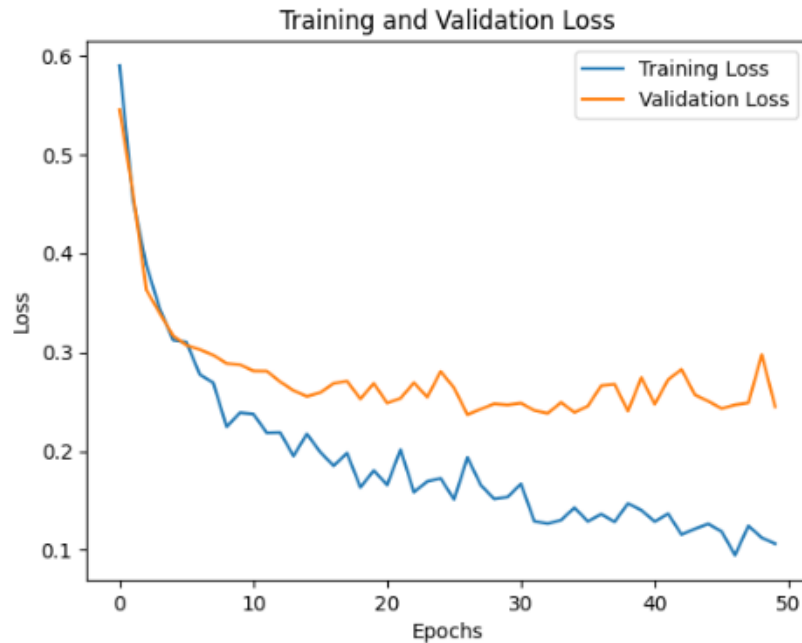


Gambar 4. 9 Grafik Validasi Optimasi



Gambar 4. 10 Grafik Validasi Default

Pada gambar 4.9 dan 4.10 terlihat bahwa pada data validasi mencapai puncaknya dan kemudian menurun pada kedua grafik namun untuk grafik yang telah dioptimalkan terdapat gap yang terlihat lebih acak.



Gambar 4. 11 Grafik Loss Optimasi



Gambar 4. 12 Grafik Loss Default

Jika terlihat bahwa kurva *loss* di kedua grafik pada data pelatihan menurun. *Loss* pada data pelatihan terus menurun tetapi *loss* pada data validasi malah stagnan

atau bahkan meningkat, ini menunjukkan indikasi *overfitting*. Dan pada model yang di optimasi terlihat lebih besar untuk itu peneliti melakukan analisis dengan melihat semua aspek sebagai berikut:




1. Perbedaan yang lebih besar antara *loss* pada data pelatihan dan data validasi bisa menjadi indikasi *overfitting*. *Overfitting* terjadi ketika model terlalu memfokuskan pada data pelatihan dan gagal menggeneralisasi dengan baik pada data yang belum pernah dilihat sebelumnya. Model tidak tahan terhadap variasi pada data baru.
2. Meskipun terdapat gap perbedaan antara *loss* pada data pelatihan dan data validasi, peneliti melihat lagi akurasi pada data validasi. Akurasi pada data validasi masih baik mencapai 93% dan memiliki akurasi 97% yang harus di pertimbangkan, dan *overfitting* tidak terlalu signifikan, maka model mampu memberikan hasil yang memuaskan.






Oleh karena itu melihat *validation accuracy* dan nilai *loss* yang lebih kecil peneliti tetap men-*deploy* dengan model yang sudah di optimasi karena lebih memberikan hasil yang memuaskan untuk data baru.




4.4. Hasil Pengujian di Android

Dalam rangka menguji model pada aplikasi yang telah dikembangkan, peneliti telah melakukan serangkaian eksperimen menggunakan sepuluh gambar wajah yang belum pernah di *training* sama sekali. Hasil observasi menunjukkan bahwa aplikasi berhasil mendeteksi patognomik wajah sesuai dengan persentasenya masing-masing. Perhatikan tabel 4.3 hasil pengujian:

Tabel 4. 3 Hasil Pengujian

No	Gambar	Jenis Gambar	Klasifikasi Prediksi	Analisis	
				Sehat	<i>Thalassemia</i>
1		Sehat	Sehat	99,93%	0,07%
2		<i>Thalassemia</i>	<i>Thalassemia</i>	0,35%	99,65%
3		Sehat	Sehat	99,85%	0,15%

4		Sehat	Sehat	99,64%	0,36%
5		Sehat	Sehat	99,73%	0,27%
6		Sehat	Sehat	93,18%	6,82%
7		Sehat	Sehat	94,11%	5,89%
8		<i>Thalassemia</i>	<i>Thalassemia</i>	0,11%	99,89%

9		Sehat	Sehat	84,51%	15,49%
10		<i>Thalassemia</i>	<i>Thalassemia</i>	13,15%	86,85%
11		<i>Thalassemia</i>	<i>Thalassemia</i>	0,76%	99,24%

Secara spesifik, untuk individu yang menderita *thalassemia beta mayor*, aplikasi menghadirkan persentase deteksi *thalassemia* yang lebih tinggi, memberikan gambaran yang akurat terkait dengan keberadaan kondisi tersebut. Sebaliknya, pada individu yang sehat, aplikasi menampilkan persentase deteksi yang lebih tinggi terhadap orang sehat, mencerminkan kemampuan aplikasi untuk mengenali keadaan normal.

Dengan hasil ini, dapat disimpulkan bahwa aplikasi memberikan respons yang memuaskan dalam mengenali perbedaan antara gambar-gambar yang mencerminkan *thalassemia beta mayor* dan kondisi sehat. Penerapan persentase deteksi sebagai metrik memperkuat ketepatan dan keandalan aplikasi dalam memberikan informasi yang relevan sesuai dengan kondisi kesehatan yang diamati.

BAB V

PENUTUP

5.1. KESIMPULAN

Berikut adalah kesimpulan berdasarkan analisis, perancangan, implementasi dan pengujian optimasi *hyperparameter* dengan *random search* dalam klasifikasi patognomik wajah penderita *thalassemia* yang telah dilakukan.

1. Dengan melakukan optimasi menggunakan *random search* selama 14 trial didapatkan 1 kombinasi *batch size* dan *learning rate* terbaik yaitu *batch size* 8 dan *learning rate* 0,001. Memberikan akurasi yang tinggi mencapai 97% dengan *validation accuracy* mencapai 93%. Untuk *loss* yaitu 10% dan *validation loss* 24%.
2. Model yang sudah di optimalkan memiliki tingkat *accuracy* dan *validation accuracy* yang lebih tinggi dari pada *hyperparameter* default perbedaan *accuracy* dari 94% ke 97%, perbedaan *validation accuracy* 90% ke 93%.
3. Dalam pelatihan model dengan arsitektur *DenseNet 121*, *hyperparameter* bisa dioptimalkan lagi mengikuti kesesuaian data.
4. Model *machine learning DenseNet 121* yang telah dioptimalkan dapat diterapkan langsung pada Android dengan bahasa pemrograman *Java* setelah dikonversi ke dalam format *tensorflow lite (tflite)*.
5. Pengoptimalan yang dilakukan mampu mendeteksi gambar yang tepat dan melakukan analisis *predict confidence* setiap gambar yang diberikan. Ketika gambar orang penderita *thalassemia* diberikan maka aplikasi memberikan *predict thalassemia* dengan hasil analisis confidence *thalassemia* lebih besar daripada orang sehat. Sebaliknya apabila gambar orang sehat diberikan ke aplikasi maka aplikasi mampu memberikan *predict sehat* kepada *user* dengan confidence sehat lebih tinggi.
6. Aplikasi sudah bisa digunakan dan diinstal langsung pada Android untuk dapat digunakan dalam mendeteksi patognomik wajah penderita *thalassemia*.

5.2. SARAN

Berikut adalah saran yang dapat dipertimbangkan untuk penelitian selanjutnya.

1. Dalam penelitian ini, fokus pada identifikasi dua kelas gambar, yaitu *thalassemia* dan gambar wajah sehat. Meski sudah terkumpul 504 gambar, namun untuk memperluas variasi dan meningkatkan keakuratan model diperlukan penambahan gambar lebih lanjut. Hal ini mendukung peningkatan performa model terutama dalam mengenali kasus *thalassemia* dan wajah sehat dengan lebih baik.
2. Penting untuk menambahkan lebih banyak data gambar wajah sehat terutama yang berasal dari individu Asia. Hal ini disebabkan karena penyakit *thalassemia* memiliki keterkaitan dengan ciri-ciri wajah mongoloid. Menambahkan variasi wajah sehat yang mencakup representasi dari berbagai etnis akan membantu meningkatkan keandalan model. Keberagaman ini mendukung presisi deteksi tanpa dipengaruhi oleh ciri-ciri etnis tertentu, menjaga akurasi identifikasi.
3. Meskipun aplikasi Android saat ini hanya memiliki satu halaman utama, perlu dipertimbangkan pengembangan fitur agar aplikasi lebih komprehensif. Beberapa ide pengembangan meliputi:
 - a) pembuatan halaman baru yang menyediakan edukasi mengenai *thalassemia*.
 - b) implementasi halaman pendaftaran donor darah dapat menjadi fitur berikutnya, yang memberikan nilai tambah dan dukungan positif untuk masyarakat.

Dengan demikian, aplikasi dapat menjadi alat yang lebih bermanfaat dan informatif bagi pengguna, sejalan dengan tujuan meningkatkan kesadaran masyarakat dan partisipasinya dalam kegiatan donor darah.

DAFTAR PUSTAKA

- Adamopoulos, S. G., & Petrocheilou, G. M. (2020). Skeletal radiological findings in thalassemia major. *Journal of Research and Practice on the Musculoskeletal System*, 04(03), 76–85. <https://doi.org/10.22540/jrpms-04-076>
- Andrinandrasana David Rasamoelina, Fouzia Adjailia, & Peter Sincák. (2020). A Review of Activation Function for Artificial Neural Network. *SAMI 2020 • IEEE 18th World Symposium on Applied Machine Intelligence and Informatics • January 23–25 • Herl'any, Slovakia*, 23–25.
- Badriyah, T., Santoso, D. B., Syarif, I., & Syarif, D. R. (2019). Improving stroke diagnosis accuracy using hyperparameter optimized optimized learning. *International Journal of Advances in Intelligent Informatics*, 5(3), 256–272. <https://doi.org/10.26555/ijain.v5i3.427>
- Bergstra, J., Ca, J. B., & Ca, Y. B. (2012). Random Search for Hyperparameter Optimization Yoshua Bengio. Dalam *Journal of Machine Learning Research* (Vol. 13). <http://scikit-learn.sourceforge.net>.
- Greeshma, K. V., & Sreekumar, K. (2019). Hyperparameter optimization and regularization on fashion-MNIST classification. *International Journal of Recent Technology and Engineering*, 8(2), 3713–3719. <https://doi.org/10.35940/ijrte.B3092.078219>
- Harrison, R. L. (2009). Introduction to Monte Carlo simulation. *AIP Conference Proceedings*, 1204, 17–21. <https://doi.org/10.1063/1.3295638>
- Huang, G., Liu, Z., van der Maaten, L., & Weinberger, K. Q. (2016). *Densely Connected Convolutional Networks*. <http://arxiv.org/abs/1608.06993>

- Jafar, A., & Lee, M. (2023). High-speed *hyperparameter* optimization for *optimized* ResNet models in image recognition. *Cluster Computing*, 26(5), 2605–2613. <https://doi.org/10.1007/s10586-021-03284-6>
- Keras Teams. (2024, Januari 11). *Tensor Flow Image Data Generator Dokumentation*.
https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator.
- Khojastepour, L., Naderi, A., Akbarizadeh, F., Movahhedian, N., & Ahrari, F. (2022). Symphysis morphology *and* mandibular alveolar bone thickness in patients with β -thalassemia major *and* different growth patterns. *Dental Press Journal of Orthodontics*, 27(2). <https://doi.org/10.1590/2177-6709.27.2.e22205.oar>
- Kishan Maladkar. (2018, Juni). *Why Is Random Search Better Than Grid Search For Machine Learning*. Mystery Vault.
- Krivulin, N. K., Guster, D., Hall, C., & Herberger, G. R. (2005). *Parallel Implementation of a Random Search Procedure: An Experimental Study*.
- Lacerda, P., Barros, B., Albuquerque, C., & Conci, A. (2021). *Hyperparameter* optimization for COVID-19 pneumonia diagnosis based on chest CT. *Sensors*, 21(6), 1–11. <https://doi.org/10.3390/s21062174>
- Nooradi Praramdana, M., Rusydi, M. A., Rizky, M., Praramdana, M. N., Dokter, P., & Kedokteran, F. (2023). *SEBUAH TINJUAN PUSTAKA: PENATALAKSANAAN BETA THALASEMIA*.
<http://jurnalmedikahutama.com>
- Rochmawati, N., Hidayati, H. B., Yamasari, Y., Peni, H., Tjahyaningtjas, A., Yustanti, W., & Prihanto, A. (2021). *Analisa Learning rate dan Batch size Pada Klasifikasi Covid Menggunakan Optimized Learning dengan Optimizer Adam*.
- Roussos, P., Mitsea, A., Halazonetis, D., & Sifakakis, I. (2021). Craniofacial shape in patients with beta thalassaemia: *a* geometric morphometric

analysis. *Scientific Reports*, 11(1). <https://doi.org/10.1038/s41598-020-80234-z>

Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on *Image Data Augmentation for Optimized Learning*. *Journal of Big Data*, 6(1). <https://doi.org/10.1186/s40537-019-0197-0>

Yang, L., & Shami, A. (2020). On *hyperparameter optimization of machine learning algorithms: Theory and practice*. *Neurocomputing*, 415, 295–316. <https://doi.org/10.1016/j.neucom.2020.07.061>

LAMPIRAN

LISTING PROGRAM

Split Data.ipynb

```

from PIL import Image
from sklearn.model_selection import train_test_split
import os
import shutil

data_directory = 'data2'
train_directory = 'splitdata7/train'
validation_directory = 'splitdata7/validation'

os.makedirs(train_directory, exist_ok=True)
os.makedirs(validation_directory, exist_ok=True)

train_dir = ''
val_dir = ''
index = 0
for image_class in os.listdir(data_directory):
    train_dir = os.path.join(train_directory, image_class)
    val_dir = os.path.join(validation_directory, image_class)
    os.makedirs(train_dir, exist_ok=True)
    os.makedirs(val_dir, exist_ok=True)
    files = os.listdir(os.path.join(data_directory, image_class))
    # Split the files into training and validation sets
    train_files, val_files = train_test_split(files, test_size=0.3, random_state=123)
    # Copy the training files to the train directory
    for file in train_files:
        image = Image.open(os.path.join(data_directory, image_class, file))
        # Check if the image mode is 'P' (paletted)
        if image.mode == 'P':
            # Convert the paletted image to RGBA
            image = image.convert('RGBA')
        src = os.path.join(data_directory, image_class, file) #datadir/namaclass, file
        dst = os.path.join(train_directory, image_class, file)
        shutil.copy(src, dst)

    # Copy the validation files to the validation directory
    for file in val_files:
        image = Image.open(os.path.join(data_directory, image_class, file))
        # Check if the image mode is 'P' (paletted)
        if image.mode == 'P':
            # Convert the paletted image to RGBA
            image = image.convert('RGBA')
        src = os.path.join(data_directory, image_class, file)
        dst = os.path.join(validation_directory, image_class, file)
        shutil.copy(src, dst)

```

Training Model.ipynb

```
import tensorflow as tf
from tensorflow.keras.applications import DenseNet121
from tensorflow.keras.layers import Dense
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam
import matplotlib.pyplot as plt
import numpy as np
from tensorflow.keras.preprocessing import image

train_data_directory = 'splitdata7/train'
validation_data_directory = 'splitdata7/validation'
batch_size = 8

# Create data generators for training and validation with augmentation
train_datagen = ImageDataGenerator(
    rescale=1.0/255.,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

val_datagen = ImageDataGenerator(rescale=1.0/255.)

# Prepare the training and validation data with labels
train_generator = train_datagen.flow_from_directory(
    train_data_directory,
    target_size=(224, 224),
    batch_size=batch_size,
    class_mode='categorical' # Change class_mode to 'categorical' for binary classification
)

validation_generator = val_datagen.flow_from_directory(
    validation_data_directory,
    target_size=(224, 224),
    batch_size=batch_size,
    class_mode='categorical' # Change class_mode to 'categorical' for binary classification
)

# Load the pre-trained DenseNet model without top layers
base_model = DenseNet121(include_top=False, weights='imagenet', input_shape=(224, 224, 3))

# Freeze the base model layers
base_model.trainable = False

# Add your own classification head on top of the base model
model = tf.keras.Sequential([
    base_model,
    tf.keras.layers.GlobalAveragePooling2D(),
    Dense(2, activation='softmax') # Change the units to 2 and use softmax activation for binary classification
])

custom_optimizer = Adam(learning_rate=0.001)

# Compile the model
model.compile(optimizer=custom_optimizer, loss='categorical_crossentropy', metrics=['accuracy'])
```

```

# Train the model using the generators for initial training
history = model.fit(
    train_generator,
    epochs=50,
    validation_data=validation_generator
)

# Get the training and validation accuracy values
train_acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

# Get the training and validation loss values
train_loss = history.history['loss']
val_loss = history.history['val_loss']

# Plot the accuracy values
plt.plot(train_acc, label='Training Accuracy')
plt.plot(val_acc, label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()

# Plot the loss values
plt.plot(train_loss, label='Training Loss')
plt.plot(val_loss, label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

# Save the model in the Keras HDF5 format
model.save("model/modelFinal_v3.2.h5")

```

Convert to tflite .ipynb

```

import tensorflow as tf
from keras.models import load_model

model = tf.keras.models.load_model('model/modelFinal_v3.2.h5')
converter = tf.lite.TFLiteConverter.from_keras_model(model)
lite_model = converter.convert()
with open("talasemia6.tflite", "wb") as f:
    f.write(lite_model)

```

MainActivity.java

```
package app.ij.mlwithtensorflowlite;

import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;

import android.Manifest;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.graphics.Bitmap;
import android.media.ThumbnailUtils;
import android.net.Uri;
import android.os.Bundle;
import android.provider.MediaStore;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;

import org.tensorflow.lite.DataType;
import org.tensorflow.lite.support.tensorbuffer.TensorBuffer;

import java.io.IOException;
import java.nio.ByteBuffer;
import java.nio.ByteOrder;
import java.text.DecimalFormat;

import app.ij.mlwithtensorflowlite.ml.Talasemia6;
```

```
public class MainActivity extends AppCompatActivity {

    Button camera, gallery;
    ImageView imageView;
    TextView resultSehat;
    TextView resultTala;
    int imageSize = 224;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        camera = findViewById(R.id.button);
        gallery = findViewById(R.id.button2);

        resultSehat = findViewById(R.id.resultSehat);
        resultTala = findViewById(R.id.resultTala);
        imageView = findViewById(R.id.imageView);

        camera.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                if (checkSelfPermission(Manifest.permission.CAMERA) == PackageManager.PERMISSION_GRANTED) {
                    Intent cameraIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
                    startActivityForResult(cameraIntent, 3);
                } else {
                    requestPermissions(new String[]{Manifest.permission.CAMERA}, 100);
                }
            }
        });
    }
}
```

```

        gallery.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent cameraIntent = new Intent(Intent.ACTION_PICK, MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
                startActivityForResult(cameraIntent, 1);
            }
        });
    }

    public void classifyImage(Bitmap image){
        try {
            Talasemia6 model = Talasemia6.newInstance(getApplicationContext());

            // Creates inputs for reference.
            TensorBuffer inputFeature0 = TensorBuffer.createFixedSize(new int[]{1, 224, 224, 3}, DataType.FLOAT32);
            ByteBuffer byteBuffer = ByteBuffer.allocateDirect(4 * imageSize * imageSize * 3);
            byteBuffer.order(ByteOrder.nativeOrder());

            int[] intValues = new int[imageSize * imageSize];
            image.getPixels(intValues, 0, image.getWidth(), 0, 0, image.getWidth(), image.getHeight());
            int pixel = 0;
            //iterate over each pixel and extract R, G, and B values. Add those values individually to the byte buffer.
            for(int i = 0; i < imageSize; i++){
                for(int j = 0; j < imageSize; j++){
                    int val = intValues[pixel++]; // RGB
                    byteBuffer.putFloat(((val >> 16) & 0xFF) * (1.f / 255.0f));
                    byteBuffer.putFloat(((val >> 8) & 0xFF) * (1.f / 255.0f));
                    byteBuffer.putFloat((val & 0xFF) * (1.f / 255.0f));
                }
            }

            inputFeature0.loadBuffer(byteBuffer);

            // Runs model inference and gets result.
            Talasemia6.Outputs outputs = model.process(inputFeature0);
            TensorBuffer outputFeature0 = outputs.getOutputFeature0AsTensorBuffer();
            // Accessing probabilities directly (assuming 2 classes)
            float confidenceSehat = outputFeature0.getFloatArray()[0];
            float confidenceTalasemia = outputFeature0.getFloatArray()[1];

            DecimalFormat decimalFormat = new DecimalFormat("0.00%");
            String outputTextSehat = "Sehat: " + decimalFormat.format(confidenceSehat);
            String outputTextTala = "Talasemia: " + decimalFormat.format(confidenceTalasemia);
            resultSehat.setText(outputTextSehat);
            resultTala.setText(outputTextTala);

            // Releases model resources if no longer used.
            model.close();
        } catch (IOException e) {
            // TODO Handle the exception
        }
    }
}

```



```

@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    if(resultCode == RESULT_OK){
        if(requestCode == 3){
            Bitmap image = (Bitmap) data.getExtras().get("data");
            int dimension = Math.min(image.getWidth(), image.getHeight());
            image = ThumbnailUtils.extractThumbnail(image, dimension, dimension);
            imageView.setImageBitmap(image);

            image = Bitmap.createScaledBitmap(image, imageSize, imageSize, false);
            classifyImage(image);
        }else{
            Uri dat = data.getData();
            Bitmap image = null;
            try {
                image = MediaStore.Images.Media.getBitmap(this.getContentResolver(), dat);
            } catch (IOException e) {
                e.printStackTrace();
            }
            imageView.setImageBitmap(image);

            image = Bitmap.createScaledBitmap(image, imageSize, imageSize, false);
            classifyImage(image);
        }
    }
    super.onActivityResult(requestCode, resultCode, data);
}
}

```