TUGAS AKHIR – EF234801

PERANCANGAN DAN PEMBUATAN APLIKASI ANDROID DETEKSI KATARAK MATA MELALUI PONSEL KAMERA MENGGUNAKAN METODE TRANSFER LEARNING

IVAN ABDILLAH RAHMAN

NRP 05111840000137

Dosen Pembimbing 1

Dr. Dwi Sunaryono, S.Kom., M.Kom.

NIP. 197205281997021001

Dosen Pembimbing 2

Fajar Baskoro, S.Kom., M.T.

NIP. 197404031999031002

Program Studi S1 Teknik Informatika

Departemen Teknik Informatika

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya

2025

TUGAS AKHIR – EF234801

PERANCANGAN DAN PEMBUATAN APLIKASI ANDROID DETEKSI KATARAK MATA MELALUI PONSEL KAMERA MENGGUNAKAN METODE TRANSFER LEARNING

IVAN ABDILLAH RAHMAN

NRP 05111840000137

Dosen Pembimbing 1

Dr. Dwi Sunaryono, S.Kom., M.Kom.

NIP 197205281997021001

Dosen Pembimbing 2

Fajar Baskoro, S.Kom., M.T.

NIP 197404031999031002

Program Studi S1 Teknik Informatika

Departemen Teknik Informatika

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya

2025

FINAL PROJECT – EF234801

DESIGN AND DEVELOPMENT OF ANDROID-BASED APPLICATION FOR DETECTING EYE-CATARACT THROUGH CAMERA HANDPHONE USING TRANSFER LEARNING METHOD

IVAN ABDILLAH RAHMAN

NRP 05111840000137

Supervisor I

Dr. Dwi Sunaryono, S.Kom., M.Kom.

NIP 197205281997021001

Supervisor II

Fajar Baskoro, S.Kom., M.T.

NIP 197404031999031002

Study Program Informatics Engineering

Department of Informatics Engineering

Faculty of Electrical Technology and Informatics

Institut Teknologi Sepuluh Nopember

Surabaya

2025

LEMBAR PENGESAHAN

PERANCANGAN DAN PEMBUATAN APLIKASI ANDROID DETEKSI KATARAK MATA MELALUI PONSEL KAMERA MENGGUNAKAN METODE TRANSFER LEARNING

TUGAS AKHIR

Diajukan untuk memenuhi salah satu syarat

memperoleh gelar Sarjana Komputer pada

Program Studi S-1 Teknik Informatika

Departemen Teknik Informatika

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Oleh : **IVAN ABDILLAH RAHMAN**

NRP. 05111840000137

Disetujui oleh Tim Penguji Tugas Akhir :

1. Dr. Dwi Sunaryono, S.Kom., M.Kom. — Pembimbing
2. Fajar Baskoro, S.Kom., M.T. — Ko-pembimbing
3. Dr. Yudhi Purwananto, S.Kom., M.Kom. — Penguji
4. Dr. Wahyu Suadi, S.Kom., MM., M.Kom. — Penguji

SURABAYA

Juli, 2025

APPROVAL SHEET

DESIGN AND DEVELOPMENT OF ANDROID-BASED APPLICATION FOR DETECTING EYE-CATARACT THROUGH CAMERA HANDPHONE USING TRANSFER LEARNING METHOD

FINAL PROJECT

Submitted to fulfill one of the requirements

for obtaining a degree Bachelor of Informatics at

undergraduate Study Program of Informatics Engineering

Departement of Informatics

Faculty of Electrical Technology and Informatics

Institut Teknologi Sepuluh Nopember

By : **IVAN ABDILLAH RAHMAN**

NRP. 05111840000137

Approved by Final Project Examiner Team :

1. Dr. Dwi Sunaryono, S.Kom., M.Kom. — Advisor
2. Fajar Baskoro, S.Kom., M.T. — Co-Advisor
3. Dr. Yudhi Purwananto, S.Kom., M.Kom. — Examiner
4. Dr. Wahyu Suadi, S.Kom., MM., M.Kom. — Examiner

SURABAYA

July, 2025

PERNYATAAN ORISINALITAS

Yang bertanda tangan di bawah ini:

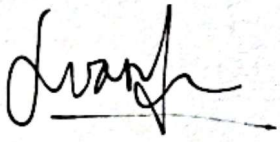
Nama mahasiswa / NRP	:	Ivan Abdillah Rahman / 05111840000137
Program studi	:	S-1 Teknik Informatika
Dosen Pembimbing I / NIP	:	Dr. Dwi Sunaryono, S.Kom., M.Kom. 197402092002121001
Dosen Pembimbing II / NIP		Fajar Baskoro, S.Kom., M.T. 197404031999031002

dengan ini menyatakan bahwa Tugas Akhir dengan judul "PERANCANGAN DAN PEMBUATAN APLIKASI ANDROID DETEKSI KATARAK MATA MELALUI PONSEL KAMERA MENGGUNAKAN METODE TRANSFER LEARNING" adalah hasil karya sendiri, bersifat orisinal, dan ditulis dengan mengikuti kaidah penulisan ilmiah.

Bilamana di kemudian hari ditemukan ketidaksesuaian dengan pernyataan ini, maka saya bersedia menerima sanksi sesuai dengan ketentuan yang berlaku di Institut Teknologi Sepuluh Nopember.

Surabaya, 11 Juli 2025

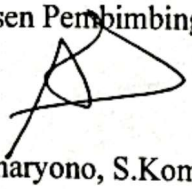
Mahasiswa



Ivan Abdillah Rahman
05111840000137

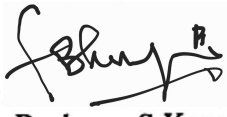
Mengetahui,

Dosen Pembimbing 1



Dr. Dwi Sunaryono, S.Kom., M.Kom.
197402092002121001

Dosen Pembimbing 2



Fajar Baskoro, S.Kom., M.T.
197404031999031002

ABSTRAK

PERANCANGAN DAN PEMBUATAN APLIKASI ANDROID DETEKSI KATARAK MATA MELALUI PONSEL KAMERA MENGGUNAKAN METODE TRANSFER LEARNING

Nama Mahasiswa / NRP	**:**	**Ivan Abdillah Rahman / 05111840000137**
Departemen	**:**	**Teknik Informatika FTEIC - ITS**
Dosen Pembimbing 1	**:**	**Dr. Dwi Sunaryono, S.Kom., M.Kom.**
Dosen Pembimbing 2		**Fajar Baskoro, S.Kom., M.T.**

Abstrak

Menurut laporan dari perwakilan Perhimpunan Dokter Spesialis Mata Indonesia (Perdami) dr. Aldiana Halim, beliau mengatakan bahwa di Indonesia dengan populasi pada tahun 2017 terdapat 8 juta orang dengan gangguan penglihatan. Sebanyak 1,6 juta orang buta ditambah dengan 6,4 juta orang dengan gangguan penglihatan sedang dan berat. Dari jumlah tersebut sebanyak 81,2% gangguan penglihatan disebabkan oleh katarak. Hingga saat ini, operasi katarak adalah cara yang paling mudah untuk mengembalikan penglihatan pada mata. Akan tetapi, untuk melakukan operasi mata tersebut dibutuhkan waktu dan peralatan operasi yang cukup banyak dan mahal untuk menentukan seberapa parah katarak pasien yang diderita. Pada tugas akhir ini, peneliti membuat aplikasi berbasis *android* untuk mendeteksi katarak pada mata melalui gambar foto mata pengguna. Foto mata kemudian dikirim ke *server* melalui API untuk dilakukan prediksi oleh model klasifikasi. Setelah menyelesaikan proses prediksi, hasilnya dikirim kembali ke aplikasi melalui *response* API untuk ditampilkan hasilnya kepada pengguna. Metode klasifikasi yang digunakan adalah model klasifikasi DenseNet201 yang mencapai angka akurasi 93,61%, tertinggi apabila dibanding dengan model klasifikasi yang lainnya.

Kata kunci: ***android, katarak, transfer learning, API, model klasifikasi***.

ABSTRACT

DESIGN AND DEVELOPMENT OF ANDROID-BASED APPLICATION FOR DETECTING EYE-CATARACT THROUGH CAMERA HANDPHONE USING TRANSFER LEARNING METHOD

Student Name / NRP	**:**	**Ivan Abdillah Rahman / 05111840000137**
Department	**:**	**Informatics Engineering, FTEIC - ITS**
Supervisor I	**:**	**Dr. Dwi Sunaryono, S.Kom., M.Kom.**
Supervisor II		**Fajar Baskoro, S.Kom., M.T.**

Abstract

According to a report from the representative of the Indonesian Ophthalmologist Association (Perdami) dr. Aldiana Halim, he said that in Indonesia with a population in 2017 there were 8 million people with visual impairments. As many as 1.6 million people are blind plus 6.4 million people with moderate and severe visual impairments. Of that number, 81.2% of visual impairments are caused by cataracts. Until now, cataract surgery is the easiest way to restore vision to the eye. However, to perform eye surgery requires a lot of time and surgical equipment and is expensive to determine how severe the patient's cataracts are. In this final project, researchers created an Android-based application to detect cataracts in the eye through photos of the user's eyes. The eye photos are then sent to the server via API to be predicted by the classification model. After completing the prediction process, the results are sent back to the application via the response API to display the results to the user. The classification method used is the DenseNet201 classification model which achieves an accuracy rate of 93.61%, the highest when compared to other classification models.

Keywords: ***android, cataract, transfer learning, API, classification model***.

KATA PENGANTAR

Pada tugas akhir ini, peneliti mengucapkan puji dan syukur kepada Allah SWT, karena atas limpahan rahmat dan karunia-Nya, peneliti dapat menyelesaikan tugas akhir ini yang berjudul "**PERANCANGAN DAN PEMBUATAN APLIKASI ANDROID DETEKSI KATARAK MATA MELALUI PONSEL KAMERA MENGGUNAKAN METODE TRANSFER LEARNING**" sebagai salah satu syarat untuk mendapatkan gelar Sarjana Komputer di bidang studi S1 Teknik Informatika Institut Teknologi Sepuluh Nopember Surabaya.

Dalam membuat tugas akhir ini, peneliti ingin mengucapkan terima kasih kepada beberapa pihak, karena telah memberikan dukungan dan bantuan untuk menyelesaikan tugas akhir ini. Pada kesempatan ini, peneliti ingin menyampaikan terima kasih kepada:

1. Kedua orang tua dan kedua adik penulis yang selalu memberikan dukungan berupa materi, doa, dan semangat,
2. Bapak Dr. Dwi Sunaryono, S.Kom., M.Kom. dan Bapak Fajar Baskoro, S.Kom., M.T. sebagai dosen pembimbing yang telah memberikan bimbingan, ajaran, dan saran agar tugas akhir yang peneliti buat mendapatkan hasil yang bagus,
3. Ibu Dini Adni Navastara, S.Kom., M.Sc. sebagai dosen wali yang telah memberikan bantuan dan saran selama peneliti menyelesaikan perkuliahan,
4. Seluruh dosen dan staf di Teknik Informatika ITS yang telah memberikan ilmu, nasihat, dan pengalaman selama menjalankan kuliah,
5. Seluruh teman satu angkatan 2018 (TransCendence) yang telah memberikan dukungan moral dan semangat dalam menyelesaikan tugas akhir,
6. Beberapa teman lama peneliti di SMP (Nada, Odi, Arya, Naily, Ayim, dan Idam) yang telah memberikan dukungan moral agar peneliti tetap semangat dalam menyelesaikan tugas akhir.

Peneliti menyadari bahwa masih banyak ditemukan kekurangan dan kesalahan selama mengerjakan tugas akhir. Oleh karena itu, peneliti mengharapkan kritik dan saran yang membangun sebagai langkah untuk menyempurnakan penelitian ini di masa yang akan datang. Semoga tugas akhir yang peneliti buat dapat memberikan manfaat kepada para pembaca.

Surabaya, 11 Juli 2025

Penulis,

Ivan Abdillah Rahman

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR KODE

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Menurut laporan Organisasi Kesehatan Dunia (WHO), Menurut laporan Organisasi Kesehatan Dunia (WHO), perkiraan jumlah penyandang tunanetra di seluruh dunia pada tahun 2021 setidaknya berjumlah 2,2 miliar. Lebih dari 1 miliar bisa dicegah atau belum teratasi. Penyebab utama gangguan penglihatan atau kebutaan adalah katarak, yang jumlahnya mencapai sekitar 94 juta jiwa. Deteksi dini dan operasi katarak dapat mengurangi kemungkinan kebutaan, menghindari kerusakan, dan memperbaiki penglihatan pasien [1].

Di Indonesia sendiri, Perwakilan dari Perhimpunan Dokter Spesialis Mata Indonesia (Perdami) dr. Aldiana Halim mengatakan di Indonesia dengan populasi pada tahun 2017 terdapat 8 juta orang dengan gangguan penglihatan. Sebanyak 1,6 juta orang buta ditambah dengan 6,4 juta orang dengan gangguan penglihatan sedang dan berat. Dari jumlah tersebut sebanyak 81,2% gangguan penglihatan disebabkan oleh katarak. Penyebab lainnya adalah refraksi atau glaukoma, atau kelainan mata hal-hal lainnya seperti kelainan refraksi, glaukoma atau kelainan mata yang berhubungan dengan diabetes [2].

Dari penjelasan di dua paragraf awal, dapat disimpulkan bahwa katarak menjadi salah satu penyakit mata yang banyak dialami oleh manusia. Sekarang ini, metode skrining katarak secara rutin adalah cara yang paling efektif untuk mencegah dari munculnya katarak pada mata [3]. Operasi katarak hingga saat ini adalah cara yang paling mudah untuk mengembalikan penglihatan pada mata [1]. Operasi ini dilakukan karena penglihatan yang dialami oleh pasien semakin memburuk sebagai akibat dari memburuknya katarak pada mata, dengan cara mengangkat lensa yang telah kabur pada mata. Pada bidang medis, dokter mata menggunakan *slit lamp* untuk melakukan penyaringan jaringan okular pasien dengan menggunakan intensitas cahaya yang tinggi [4] dan hasilnya dapat diklasifikasikan sesuai dengan kriteria diagnosis yang ada, contohnya *Lens Opacities Classification System III* (Sistem Klasifikasi Kekeruhan Lensa III) [5].

Melakukan operasi mata tersebut memakan waktu yang cukup banyak untuk menentukan seberapa parah katarak pasien yang diderita [6]. Selain itu, peralatan yang dibutuhkan cukup banyak untuk memeriksa satu pasien, sehingga memengaruhi pengeluaran biaya operasional yang mahal. Oleh karena itu, beberapa penelitian telah dilakukan untuk mengembangkan sistem deteksi katarak yang lebih cepat dan efisien dalam beberapa tahun terakhir agar dapat mengantisipasi penyakit katarak secara dini dan membantu meningkatkan akurasi diagnosis katarak pada mata. Salah satunya adalah menggunakan kecerdasan buatan untuk mendeteksi dan mengklasifikasikan katarak pada mata menggunakan beberapa gambar mata yang telah dikumpulkan untuk diklasifikasikan dan dapat didiagnosis apakah pasien ini mengidap penyakit katarak atau tidak, seperti yang telah dilakukan oleh Hossain dan kawan-kawan [7].

Pada tugas akhir ini, peneliti membentuk sebuah aplikasi deteksi katarak mata berbasis *android* dengan proses klasifikasi menggunakan metode *transfer learning* pada ponsel pintar atau kita lebih mengenal sebagai *smartphone*. Pada proses klasifikasi ini, aplikasi mengambil sebuah sampel gambar mata, lalu sampel gambar yang telah diambil ini diklasifikasikan menggunakan model *transfer learning* yang telah disiapkan pada aplikasi. Selanjutnya, hasil klasifikasi ditampilkan pada aplikasi.

Tugas akhir ini dibuat pada ponsel pintar karena menurut data pada Badan Pusat Statistik di Indonesia, penduduk yang memiliki atau menguasai ponsel pintar sejumlah 67,29% dari total penduduk Indonesia [8]. Hal ini menunjukkan bahwa dalam kesehariannya, ponsel pintar sangat banyak digunakan oleh penduduk Indonesia, sehingga penelitian menggunakan *android* pada ponsel pintar sebagai basis aplikasi. Harapan dari dibentuknya aplikasi ini adalah untuk memberikan kontribusi pada bidang kesehatan, khususnya pada penyakit katarak. Selain itu, aplikasi ini diharapkan dapat menjadi inspirasi bagi pengembangan teknologi *smartphone* dan *machine learning* di masa mendatang.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat pada tugas akhir ini adalah sebagai berikut:

1. Bagaimana cara mengolah *dataset* katarak mata untuk dilakukan klasifikasi terhadap model klasifikasi?
2. Bagaimana cara membuat model *deep learning* untuk mendeteksi adanya katarak pada mata berdasarkan hasil foto yang diambil?
3. Bagaimana cara membuat aplikasi berbasis *android* untuk mendeteksi adanya katarak pada mata berdasarkan hasil foto yang diambil?

1.3 Batasan Masalah

Batasan masalah yang dilakukan pada tugas akhir ini adalah sebagai berikut:

1. Aplikasi ini menggunakan Android 8 sebagai sistem operasi.
2. *Dataset* yang digunakan adalah *dataset cataract detection* yang berasal dari Kaggle, hasil kontribusi dari Hemoo Redaoo [9] dan Nandan Padiaa dkk [10].
3. Aplikasi ini dibangun menggunakan Android Studio, dengan bahasa pemrograman yang digunakan adalah Kotlin.
4. Proses klasifikasi katarak menggunakan *python* sebagai bahasa pemrograman.
5. Aplikasi ini hanya menyediakan hasil klasifikasi antara terdapat katarak pada mata atau tidak.

1.4 Tujuan

Tujuan dari tugas akhir ini adalah sebagai berikut:

1. Menemukan cara mengolah *dataset* katarak mata untuk dilakukan klasifikasi terhadap model klasifikasi.
2. Mendeteksi adanya katarak pada mata berdasarkan hasil foto yang diambil menggunakan model *deep learning*.
3. Membuat sebuah aplikasi berbasis *android* untuk mendeteksi adanya katarak pada mata berdasarkan hasil foto yang diambil.

1.5 Manfaat

Manfaat dari tugas akhir ini adalah agar aplikasi pendeteksi katarak mata berbasis *android* ini dapat menjadi upaya untuk mendeteksi katarak mata secara dini dan lebih cepat untuk dilakukan pengobatan ke rumah sakit. Aplikasi ini juga diharapkan dapat diimplementasikan dalam upaya deteksi penyakit yang lain.

BAB 2 TINJAUAN PUSTAKA

2.1 Hasil Penelitian Terdahulu

Pada penelitian terdahulu, Yusuf dan kawan-kawan membuat sebuah sistem deteksi katarak dengan berbasis web menggunakan metode *convolutional neural network* (CNN) dengan pengambilan gambar kamera digital. *Dataset* yang digunakan masing-masing berjumlah 100 untuk mata normal dan 100 untuk mata katarak. Proses klasifikasi menggunakan model ImageNet dikembangkan di ILSVRC2012 menggunakan pengklasifikasi *convolutional neural network* dan mentransfer pengetahuannya menggunakan *transfer learning* untuk melakukan *data training*. Model ini memperoleh kemampuan untuk mengklasifikasikan gambar mata menjadi "Normal" dan "Katarak". Sistem dirancang untuk mengambil gambar sebagai *input* dan mendapatkan sensitivitas 69%, spesifisitas 86%, presisi 86%, F-score 56%, dan AUC 84,56%. Skor akurasinya adalah 78% yang dipengaruhi oleh penggunaan model yang telah di-*training* selama klasifikasi gambar ImageNet menggunakan *deep convolutional neural network* [11].

Pada penelitian yang lain, Tawfik dan kawan-kawan menggunakan dua pengklasifikasi, yaitu *support vector machines* dan *artificial neural networks* untuk melakukan deteksi katarak. Transformasi *wavelet discrete* dan transformasi 2D Log Gabor digunakan untuk teknik ekstraksi fitur untuk *dataset* 120 gambar mata yang dilanjutkan dengan proses klasifikasi yang mengklasifikasikan kumpulan gambar menjadi tiga kelas; stadium normal, dini, dan lanjut. Hasil dari perbandingan ini adalah bahwa *support vector machines* memberikan hasil lebih baik dengan tingkat akurasi mencapai 96,8% dibandingkan dengan *artificial neural networks* yang mencapai tingkat akurasi sebesar 92,3% [12].

2.2 Dasar Teori

Berikut ini adalah beberapa dasar teori yang digunakan untuk mengerjakan tugas akhir pada kali ini.

2.2.1 Katarak

Katarak adalah hasil dari kekeruhan kristal yang mengaburkan lensa mata pada manusia. Lensa yang kabur fokus cahaya pada retina, sehingga mengakibatkan penglihatan menjadi buruk. Penyebab terbentuknya katarak adalah adanya penggumpalan protein hasil dari denaturasi protein pada kapsul lensa dan pigmen yang tersimpan pada kristal. Banyak faktor yang menyebabkan katarak, antara lain genetik, penuaan pada usia manusia, dan merokok [13].

Ada beberapa cara untuk mengklasifikasikan katarak. Standar yang paling umum yang digunakan adalah berdasarkan letak pengendapan protein pada lensa mata. Katarak diklasifikasikan menjadi tiga jenis, yaitu katarak katarak nukleus, katarak kortikal, dan katarak subkapsular posterior. Katarak nuklir berkembang di inti lensa, menjadi kuning dan coklat. Di sisi lain, katarak kortikal, yang sebagian besar dipicu oleh diabetes, terjadi di korteks lensa. Sementara itu, katarak subkapsular posterior, yang sering terjadi pada mereka yang mengonsumsi obat steroid dosis tinggi, terjadi di bagian belakang lensa [14].

2.2.2 Machine Learning

Machine learning adalah subbidang dari kecerdasan buatan yang melibatkan pengembangan algoritma dan model statistik yang memungkinkan komputer meningkatkan kinerjanya dalam tugas melalui pengalaman. Algoritma ini dirancang untuk mempelajari data-

data yang diterima dan membuat prediksi tanpa adanya instruksi secara eksplisit. *Machine learning* digunakan dalam berbagai macam aplikasi, seperti pengenalan gambar dan suara dan sistem rekomendasi [15].

2.2.3 Deep Learning

Deep learning adalah bidang baru dari *machine learning* yang populer belakangan ini. *Deep learning* mengacu pada arsitektur yang berisi beberapa *hidden layer* (*deep network*) untuk mempelajari berbagai fitur dengan berbagai level abstraksi. Algoritma *deep learning* berusaha untuk mengeksploitasi struktur yang tidak diketahui dalam distribusi masukan untuk menemukan representasi yang baik, sering kali pada berbagai level, dengan fitur yang dipelajari pada level yang lebih tinggi didefinisikan dalam bentuk fitur level yang lebih rendah. *Deep learning* memungkinkan memasukkan data mentah (piksel dalam hal data gambar) ke algoritma *learning* tanpa terlebih dahulu mengekstraksi fitur atau menentukan vektor fitur. *Deep learning* dapat mempelajari serangkaian fitur yang tepat, dan hal ini dilakukan dengan cara yang jauh lebih baik daripada mengekstraksi fitur-fitur tersebut menggunakan pengkodean tangan. Daripada membuat serangkaian aturan dan algoritma untuk mengekstrak fitur dari *raw data*, *deep learning* melibatkan pembelajaran fitur-fitur ini secara otomatis selama proses *training* [16].

Dalam beberapa tahun terakhir, perkembangan *deep learning* mengalami perkembangan yang sangat pesat. *Deep learning* kini memiliki banyak metode mulai dari *artificial neural network* (ANN), *multilayer perceptron* (MLP) *neural network*, *backpropagation neural network* (BPNN), *convolutional neural network* (CNN), *recurrent neural network* (RNN). Metode-metode tersebut telah diterapkan untuk menyelesaikan berbagai tugas seperti klasifikasi gambar dan segmentasi gambar medis [17].

2.2.4 Transfer Learning

Transfer learning adalah *machine learning* yang modelnya dilakukan *training* dan dikembangkan untuk satu tugas dan digunakan lagi menggunakan *dataset* yang baru. Hal ini mengacu pada saat terdapat situasi ketika apa yang telah dipelajari dalam satu pengaturan dieksploitasi untuk meningkatkan optimalisasi dalam pengaturan yang baru. *Transfer learning* biasanya diterapkan ketika ada *dataset* baru yang lebih kecil dari *dataset* asli yang digunakan untuk melakukan *training model* sebelumnya [18].

2.2.5 CNN

Convolutional Neural Network (CNN) dengan kemampuan ekstraksi fitur dan klasifikasi dapat mengatasi masalah klasifikasi secara efektif dan efisien. Selain itu, pembagian bobot dan ekstraksi fitur dalam CNN meningkatkan efisiensi komputasi secara signifikan. Dengan demikian, CNN dapat menangani masalah dengan data berskala besar [19].

CNN terdiri dari *input layer*, beberapa *convolutional layers*, dan sebuah *output layer*. Fungsi dari *convolutional layers* adalah untuk melakukan *learning* dari representasi fitur *low-level, middle-level,* dan *high-level* dari masukan gambar melalui operasi *convolution* yang besar di berbagai tahapan [17].

2.2.6 VGG16

VGG16 adalah salah satu arsitektur *deep learning* yang dikenalkan oleh Oxford University. Arsitektur ini mencakup 41 *layer* yang terdistribusi ke 16 *weight layer*, 13 *layer* konvolusi, dan 3 *fully-connected layer*. VGG16 menggunakan 3x3 *kernel* yang kecil (filter) pada seluruh *layer*

konvolusi. *Max pooling layer* selalu mengikuti *layer* konvolusi. Masukan untuk VGG16 adalah gambar tiga *channel* gambar berukuran tetap 224x224. Dalam VGG16, tiga *fully-connected layer* memiliki kedalaman yang berbeda. Dua *layer* pertama memiliki ukuran *channel* yang sama (4096), sedangkan *fully-connected* terakhir memiliki ukuran 1000 *channel*, yang mewakili jumlah label kelas dalam dataset ImageNet. *Layer output* adalah *soft-max layer* yang bertanggung jawab atas probabilitas yang diberikan untuk masukan gambar [20].

Seperti model yang telah dilatih sebelumnya, VGG16 memerlukan pelatihan berat jika bobot dilakukan inisial secara acak. Jadi, secara umum, model CNN menggunakan teknik *transfer learning*. *Transfer learning* mengacu pada mekanisme di mana model yang dilatih pada satu tugas digunakan dengan cara tertentu pada tugas serupa kedua. Yaitu, model CNN dilatih pada masalah yang serupa dengan masalah yang sedang ditangani, di mana masukannya sama, tetapi keluarannya mungkin berbeda sifatnya. Dalam kasus ini, model VGG16 dilatih menggunakan *dataset* ImageNet, yang berisi banyak gambar objek dunia nyata [21].

Kemudian, bobot lapisan bermigrasi ke tugas klasifikasi ekstraksi fitur. Dengan demikian, waktu pelatihan berkurang. Selain itu, *transfer learning* merupakan teknik klasifikasi yang lebih kuat ketika *dataset* kecil dievaluasi. *Transfer learning* dapat digunakan baik untuk klasifikasi maupun ekstraksi fitur setelah adaptasi beberapa *layer* dari model yang telah dilatih sebelumnya. Dalam studi ini, kemampuan VGG16 dengan pembelajaran transfer digunakan untuk mengekstraksi fitur tingkat tinggi dari masukan gambar [22].

2.2.7 ResNet50

ResNet50 adalah salah satu pengembangan dari arsitektur CNN, ResNet50 adalah singkatan dari *Residual Network*-50. Arsitektur ini dikembangkan oleh He dkk. ResNet sendiri memanfaatkan *residual connection* dalam setiap *layer* untuk menangani masalah *gradient descent*, sehingga mempercepat konvergensi *deep network*. Arsitektur ini memiliki kompleksitas komputasi yang lebih rendah dibandingkan arsitektur lainnya, meskipun kedalamannya meningkat. *Deep network* pada arsitektur ResNet terbukti lebih baik dalam melakukan tugas klasifikasi karena dapat mengekstraksi fitur yang lebih representatif. Jumlah lapisan yang digunakan oleh arsitektur ResNet adalah 50, dengan 48 *layer* konvolusi, satu *layer max pooling*, dan satu *global average pooling layer* [23].

2.2.8 DenseNet 201

DenseNet diusulkan oleh Huang dkk., dan dikenal karena kinerjanya yang luar biasa pada empat kumpulan data tolok ukur pengenalan objek seperti CIFAR-100 dan ImageNet25. Untuk memaksimalkan aliran informasi antara lapisan-lapisan dalam jaringan, arsitektur DenseNet menggunakan pola konektivitas sederhana yang menghubungkan semua lapisan secara langsung satu sama lain dalam mode umpan maju, yaitu, setiap lapisan memperoleh masukan tambahan dari semua lapisan sebelumnya dan meneruskan peta fiturnya sendiri ke semua lapisan berikutnya. Dengan arsitektur ini, DenseNet memiliki beberapa keunggulan yang mengesankan, termasuk mengurangi masalah gradien yang menghilang, memperkuat propagasi fitur, mendorong penggunaan kembali fitur, dan secara substansial mengurangi jumlah parameter [24].

2.2.9 Python

Python adalah bahasa pemrograman yang mudah dalam penggunaannya. *Python* termasuk dalam bahasa pemrograman interpretatif yang dapat di gunakan di berbagai macam platform. *Python* termasuk jenis bahasa pemrograman yang memiliki level sangat tinggi sehingga

memiliki tipe data tingkat tinggi juga. Seperti bahasa program tingkat tinggi lainya, *python* dapat membagi program yang dibuat menjadi beberapa modul sehingga dapat di gunakan di program *python* yang lain. *Python* juga telah menyediakan beberapa modul yang dapat digunakan seperti file I/O, Sockets, hingga grafis serti Tkinter, bahkan karena komunitasnya yang bagus sampai ada beberapa *library open source* untuk mendukung modul-modul yang dimiliki *python* dibanyak bidang seperti *machine learning*, *data mining*, dan AI contohnya adalah Tensorflow, Keras dan lain-lainya [25].

2.2.10 TensorFlow

Versi 1.0.0 dari TensorFlow dirilis pada tanggal 11 Februari 2017. TensorFlow adalah *library* perangkat lunak atau *framework* yang dibuat oleh Google yang memungkinkan Anda untuk mengimplementasikan teknik *machine learning* dan *deep learning* dengan cepat. Banyak persamaan matematika yang dibuat sederhana untuk dihitung menggunakan kombinasi aljabar komputasional dan teknik pengoptimalan. Ini adalah *open-source library* untuk *machine learning* skala besar dan komputasi numerik. Model *machine learning* dikembangkan dan dilatih menggunakan TensorFlow. *Library* ini menggunakan pembelajaran mesin dan kecerdasan buatan untuk meningkatkan mesin pencari dan teks gambar, deteksi pola, dan fungsi lainnya [26].

Dalam *deep learning*, *tensor* adalah cara umum untuk mengodekan data. Ini memungkinkan programmer untuk merancang grafik aliran data, yang merupakan struktur yang menjelaskan bagaimana data mengalir melalui grafik atau serangkaian *node* pemrosesan. Ini memiliki sejumlah API (Antarmuka Pemrograman Aplikasi). Ini dapat dibagi menjadi dua kelompok: tingkat rendah dan tingkat tinggi. TensorFlow memudahkan pemula dan pengembang untuk membuat model *machine learning* untuk *cloud*, *mobile*, *desktop*, dan *web* [26].

2.2.11 Android

Android adalah sekumpulan *stack* perangkat lunak untuk perangkat seluler, yang mereferensikan terhadap sekumpulan sistem program atau sekumpulan program aplikasi yang membentuk suatu sistem yang lengkap. *Stack* perangkat lunak ini dibagi menjadi empat *layers* yang berbeda, yang mencakup 5 grup berbeda:

a. *Layer* aplikasi
   Platform perangkat lunak Android akan hadir dengan serangkaian aplikasi dasar seperti browser, klien email, program SMS, peta, kalender, kontak dan banyak lagi. Semua aplikasi ini ditulis menggunakan bahasa pemrograman Java. Perlu disebutkan bahwa aplikasi dapat dijalankan secara bersamaan, dimungkinkan untuk mendengarkan musik dan membaca email secara bersamaan. Lapisan ini sebagian besar akan digunakan oleh pengguna ponsel pada umumnya.
b. *Framework* aplikasi
   Framework aplikasi adalah kerangka perangkat lunak yang digunakan untuk mengimplementasikan struktur standar aplikasi untuk sistem operasi tertentu. Dengan bantuan pengelola, penyedia konten, dan pemrogram layanan lainnya, ia dapat menyusun kembali fungsi-fungsi yang digunakan oleh aplikasi lain yang sudah ada.
c. *Libraries*
   Libraries yang tersedia di Android, semuanya ditulis dalam C/C++. Mereka akan dipanggil melalui antarmuka Java.

d. *Runtime*

Runtime Android terdiri dari dua komponen. Pertama, sekumpulan *libraries* inti yang menyediakan fungsionalitas yang tersedia di *libraries* inti bahasa pemrograman Java. Kedua mesin virtual Dalvik yang beroperasi seperti penerjemah antara sisi aplikasi dan sistem operasi. Setiap aplikasi yang berjalan di Android ditulis dalam Java. Karena sistem operasi tidak dapat memahami bahasa pemrograman ini secara langsung, program Java akan diterima dan diterjemahkan oleh mesin virtual Dalvik. Kode yang diterjemahkan kemudian dapat dieksekusi oleh sistem operasi. Aplikasi akan dienkapsulasi di Dalvik. Untuk setiap program, mesin virtualnya sendiri tersedia meskipun beberapa program berjalan paralel.

e. *Kernel*

Android menggunakan Linux sebagai *kernel* untuk perangkat *driver*, manajemen memori, manajemen proses, dan jaringan [27].

BAB 3 METODOLOGI

3.1 Metode yang Digunakan

Pada bagian metodologi ini, akan dijelaskan beberapa tahap metode yang dirancang untuk menyelesaikan penelitian pada tugas akhir ini.

3.1.1 Pengembangan Aplikasi

Pada tugas akhir kali ini, aplikasi pada *smartphone* akan berperan sebagai proses antarmuka yang menjadi penghubung antara pengguna dengan sistem. Aplikasi deteksi katarak dibangun untuk membantu pengguna dalam mengambil gambar mata yang akan dideteksi baik melalui kamera atau mengambil dari penyimpanan di galeri *smartphone*.

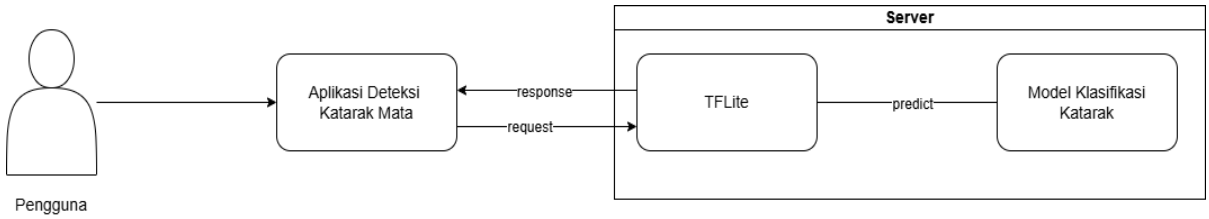
Ada beberapa langkah-langkah yang digunakan dalam mengembangkan aplikasi pada *smartphone*. Setelah peneliti mencari beberapa literatur tentang pengembangan aplikasi, peneliti menemukan bahwa menurut jurnal yang dipublikasikan oleh SoobiaSaeed dkk., *software development* adalah proses dari berbagai tahapan, namun saling berhubungan satu sama lain. Setiap tahapan memiliki spesifik waktu tertentu untuk menghasilkan sesuatu dari setiap tahapan. Tahapan-tahapan tersebut adalah *research, planning, design, development, testing, configuration,* dan *maintenance* (analisa, perencanaan, desain, pengembangan, uji coba, pengaturan, dan pemeliharaan [28].

3.1.1.1 Analisa

Pada tahap ini, peneliti melakukan analisa terhadap kebutuhan baik fungsional maupun non-fungsional yang digunakan untuk membangun aplikasi deteksi katarak mata. Kebutuhan fungsional mencakup beberapa *use case* yang digunakan dalam aplikasi. *Use case* ini berisi tentang interaksi antara pengguna dengan aplikasi. Lalu, kebutuhan non-fungsional mencakup tentang beberapa infrastruktur yang digunakan dalam merancang aplikasi seperti bahasa pemrograman yang digunakan, *library*, *framework* yang dimanfaatkan, dan beberapa versi perangkat pendukung untuk membangun aplikasi.

3.1.1.2 Perencanaan

Pada tahap ini, peneliti melakukan perencanaan terhadap aplikasi yang akan dibangun, sesuai dengan hasil analisa yang telah dilakukan pada tahap sebelumnya. Perancangan aplikasi meliputi desain arsitektur sistem yang digunakan.



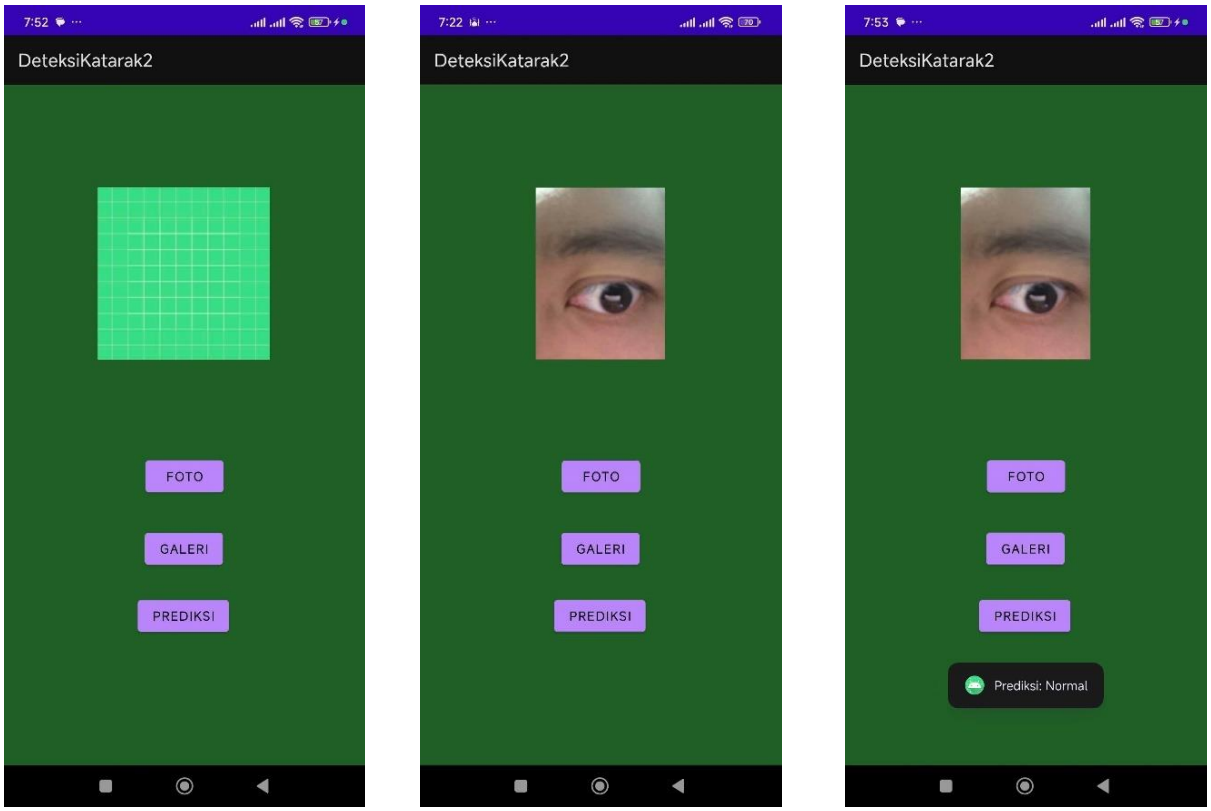
Gambar 3.1 Diagram Arsitektur Sistem

Gambaran dari desain arsitektur sistem yang digunakan pada tugas akhir ini ada pada Gambar 3.1. Desain arsitektur sistem ini bermula dari pengguna aplikasi yang mengakses aplikasi. Ketika pengguna melakukan deteksi gambar mata pada aplikasi, aplikasi akan melakukan aksi *request* kepada sebuah API yang dibuat menggunakan *framework* TFLite. Pada desain arsitektur ini, API berfungsi sebagai jembatan antara *request* dari aplikasi dan *response*

dari model klasifikasi katarak mata. Setelah aksi *request* dari aplikasi telah diterima oleh API, API meneruskan *request* ini kepada model klasifikasi katarak untuk dilakukan aksi *predict*, yaitu proses prediksi gambar mata pengguna aplikasi. Setelah model klasifikasi telah menyelesaikan hasil prediksinya, model klasifikasi akan mengirim aksi *response* kepada aplikasi melalui API terlebih dahulu, lalu API akan mengirim aksi *response* dari model klasifikasi kepada aplikasi yang akhirnya dapat dilihat hasilnya oleh pengguna. Hasil yang ditampilkan oleh aplikasi adalah "Katarak" atau "Normal".

3.1.1.3 Desain

Langkah selanjutnya dari pembangunan aplikasi adalah desain. Peneliti perlu melakukan desain terhadap aplikasi yang akan dibangun agar dapat membangun aplikasi sesuai dengan rumusan masalah dan perencanaan aplikasi yang telah dijabarkan pada subbab 3.1.1.2 tentang Perencanaan. Setelah melakukan analisa, diketahui bahwa aplikasi harus dapat menampilkan gambar yang didapatkan dari hasil ambil gambar melalui kamera atau dari galeri, untuk dilakukan prediksi terhadap model klasifikasi katarak pada mata. Kemudian, aplikasi juga harus memberikan informasi tentang keberhasilan prediksi gambar, kategori kelas gambar yang diprediksi, dan pesan *error* apabila gambar tidak dapat diambil. Aplikasi ini juga memberikan edukasi dan informasi tentang tujuan, kritik, dan saran aplikasi ini dibangun.



Gambar 3.2 Desain Aplikasi: (i) Tampilan Deteksi Katarak; (ii) Sebelum Deteksi; (iii) Sesudah Deteksi

3.1.1.4 Pengembangan

Pada tahap pengembangan, peneliti melakukan implementasi terhadap desain dan perencanaan yang telah disusun sebelumnya. Tahap ini membutuhkan penulisan program aplikasi deteksi katarak mata dan model klasifikasi katarak mata. Setelah itu, peneliti

melakukan *build* dan *deployment* aplikasi untuk dilakukan uji coba dan mencapai hasil akhir dari aplikasi dibangun.

3.1.1.5 Uji coba

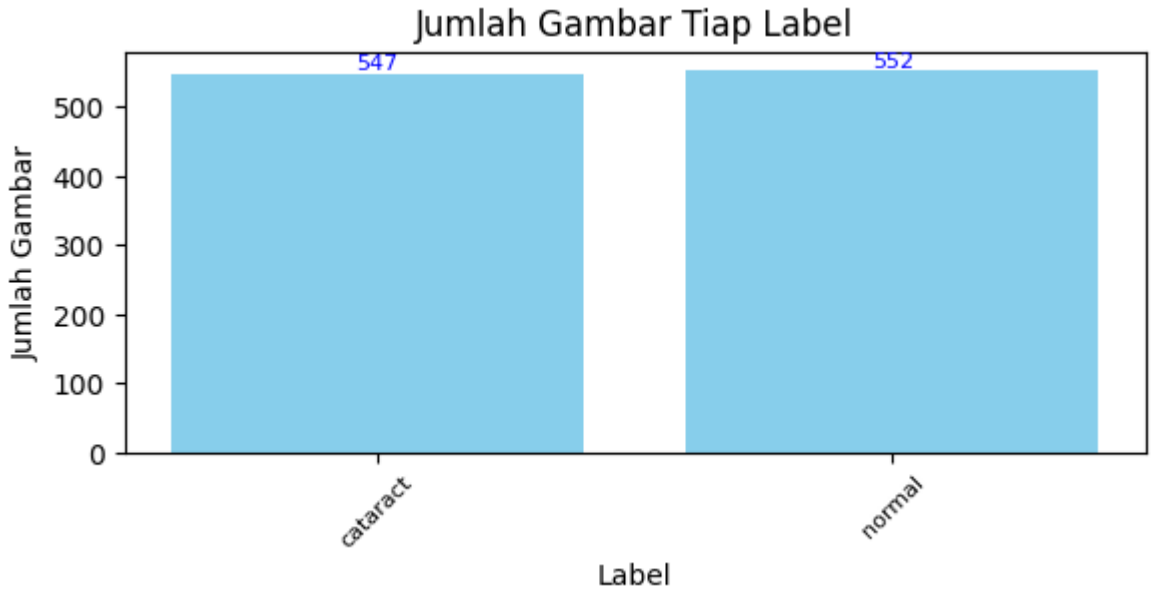
Pada tahap uji coba, peneliti melakukan uji coba terhadap aplikasi dengan beberapa kondisi yang ditentukan. Jika kondisi yang dilakukan uji coba terhadap aplikasi sukses, aplikasi bisa dirilis dan dapat digunakan oleh pengguna yang membutuhkan. Tahap ini juga mengecek apabila aplikasi mengalami *bug*, *glitch*, atau masalah yang lain saat mengembangkan aplikasi.

3.1.1.6 Pemeliharaan

Pada tahap pemeliharaan, peneliti melakukan pengawasan terhadap aplikasi yang sudah dirilis (*deploy*) untuk memeriksa jika ditemukan *error* ketika aplikasi sudah dirilis. Selain itu, tahap ini juga memungkinkan peneliti untuk meningkatkan performa dan kualitas aplikasi agar aplikasi berjalan lebih baik dan maksimal.

3.1.2 Dataset

Untuk melakukan pelatihan model yang akan digunakan pada tugas akhir ini, peneliti memerlukan *dataset* gambar mata normal dan mata katarak yang didapatkan dari *website* penyedia *dataset* yang sifatnya *open source* bernama Kaggle. Di dalam *website* ini, peneliti dapat menemukan berbagai macam *dataset* yang diperlukan untuk penelitian. Pada tugas akhir ini, peneliti menggunakan *dataset* yang merupakan hasil kontribusi dari Hemoo Redaoo [9] dan Nandan Padiaa dkk [10]. Dari kedua kontributor tersebut, kedua *dataset* ini digabungkan sehingga berjumlah 1099 gambar mata normal dan mata katarak. Gambar-gambar mata tersebut kemudian diberi label dengan dua kelas, yaitu *cataract* dan *normal*. Perincian jumlah gambar pada *dataset* dijabarkan pada Gambar 3.3.



Gambar 3.3 Jumlah Gambar pada Dataset

Pada Gambar 3.3, dapat diketahui bahwa jumlah gambar mata katarak ada 547 gambar dan gambar mata normal ada 552 gambar. Karena peneliti merasa kedua jumlah gambar pada masing-masing kelas tidak ada perbedaan jumlah gambar yang jauh, akhirnya *dataset* tersebut digunakan pada tugas akhir ini.

3.1.3 Model Klasifikasi

Pada tugas akhir ini, penelitian melakukan proses pelatihan model klasifikasi dengan menggunakan *transfer learning*. Model ini digunakan untuk melakukan klasifikasi gambar pada *dataset* untuk dibagi-bagi ke dalam kelas yang ditentukan. Ada beberapa model yang diuji pada tugas akhir ini, yaitu CNN, VGG16, ResNet50, dan DenseNet201. Keempat model tersebut nantinya akan diuji untuk menentukan model yang memiliki hasil akurasi yang paling baik. Ketika sudah ditemukan hasil akurasi yang paling baik, model tersebut digunakan ke dalam aplikasi deteksi katarak pada mata.

3.1.4 Persiapan Data Klasifikasi

Pada tugas akhir ini, *dataset* yang digunakan pada proses pelatihan model klasifikasi menggunakan *dataset* yang telah dijelaskan pada subbab 3.1.2. Selanjutnya, *dataset* tersebut kemudian dibagi ke dalam dua bagian, yaitu *data training* dan *data validation* dengan menggunakan *ImageDataGenerator*. Setelah melakukan pembagian *dataset*, peneliti melakukan *Image Augmentation* yang berguna untuk memperbanyak jumlah data yang akan dilatih dalam melakukan proses pelatihan model klasifikasi.

3.1.5 Pelatihan Model Klasifikasi

Setelah *dataset* dibagi menjadi *data training* dan *data validation*, selanjutnya adalah melakukan pelatihan model klasifikasi. Pada bagian ini, beberapa model klasifikasi seperti CNN, VGG16, ResNet50, dan DenseNet201 kemudian dilakukan pelatihan pada *dataset*. Masing-masing hasil dari pelatihan model klasifikasi akan ditampilkan dalam bentuk grafik diagram yang berisi tentang akurasi model selama mengalami proses pelatihan. Selain itu, hasil pelatihan juga akan menunjukkan *loss model* selama proses pelatihan.

3.1.6 Evaluasi Model

Setelah pelatihan model klasifikasi pada *dataset* dilakukan, berikutnya adalah mengevaluasi model klasifikasi. Evaluasi ini dilakukan untuk mengetahui performa dan hasil akurasi dari model klasifikasi yang telah dijalankan. Untuk setiap model klasifikasi yang telah dilakukan, hasil akhir akan ditunjukkan hasil perhitungan *val_accuracy* dan *val_loss*.

Keempat hasil dari proses pelatihan model klasifikasi akan dibuat grafik akurasi untuk menentukan model klasifikasi yang memiliki akurasi yang terbaik. Akurasi yang terbaik digunakan pada aplikasi deteksi katarak mata.

3.1.7 Integrasi dengan Aplikasi Android

Pada tahap ini, model yang telah ditentukan hasil terbaiknya, kemudian diintegrasikan dengan aplikasi deteksi katarak yang berada pada *smartphone*. Integrasi dilakukan dengan cara menggunakan sebuah API yang bernama TensorFlow Lite. API yang dibangun menggunakan bahasa pemrograman *python* ini akan menerima gambar mata dari aplikasi di *smartphone* baik melalui kamera maupun dari penyimpanan pada galeri. Lalu, ketika API sudah menerima gambar mata yang akan diprediksi hasilnya, API akan melakukan proses deteksi menggunakan model klasifikasi yang sudah ditentukan sebelumnya. Ketika API selesai melakukan proses deteksi pada gambar mata, API akan mengirim hasil prediksi kepada aplikasi pada *smartphone* berupa tulisan yang menunjukkan bahwa gambar mata yang diprediksi masuk kelas *cataract* atau *normal*.

3.2 Perangkat yang Digunakan

Pada bagian ini, akan dijelaskan beberapa perangkat yang digunakan dalam membangun aplikasi untuk tugas akhir ini.

3.2.1 Perangkat Keras

Tugas akhir ini menggunakan perangkat keras seperti komputer dan ponsel pintar. Komputer digunakan untuk membangun aplikasi deteksi katarak dan melakukan klasifikasi *dataset* katarak, sedangkan ponsel pintar digunakan sebagai tempat peluncuran aplikasi deteksi katarak. Penjelasan tentang spesifikasi perangkat keras yang digunakan, dijelaskan pada Tabel 3.1

Komputer	
Tipe	Laptop ASUS TUF Gaming FX505DD
Operating System (OS)	Windows 10 Home Single Language 64-bit
Processor	AMD Ryzen 5 3550H
Memory	16384 MB RAM
GPU	AMD Radeon ™ Vega 8 Graphics
	NVIDIA GeForce GTX 1050

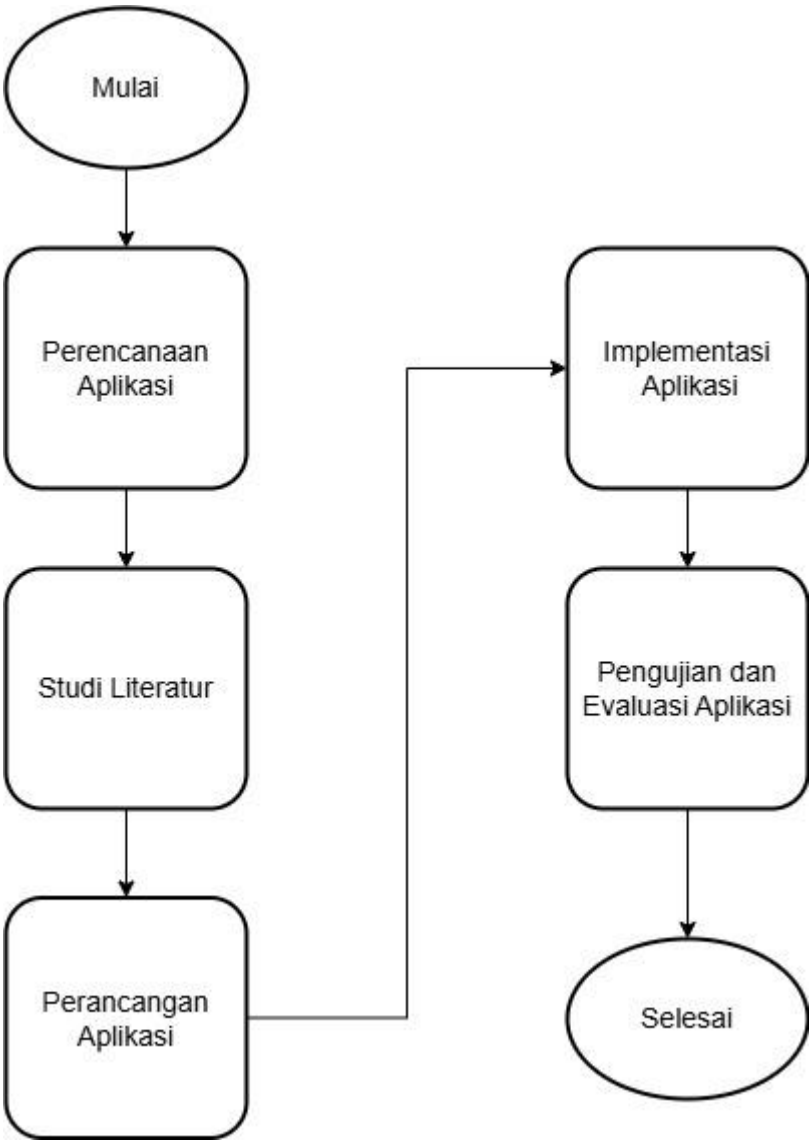
Tabel 3.1 Spesifikasi Perangkat Keras

3.2.2 Perangkat Lunak

Tugas akhir ini menggunakan beberapa perangkat lunak untuk melakukan pembuatan aplikasi deteksi katarak. Tugas akhir ini dibangun menggunakan IDE Android Studio Electric Eel versi 2022.1.1. Bahasa pemrograman yang digunakan untuk membangun aplikasi deteksi katarak adalah Kotlin. Kemudian, untuk melakukan klasifikasi pada *dataset* katarak, tugas akhir ini menggunakan Visual Studio Code dan beberapa *package* dari *python* seperti Tensorflow, Keras, Numpy, cv2, dan matplotlib untuk menampilkan grafik akurasi dari beberapa algoritma *machine learning* yang telah dijalankan.

3.3 Urutan Pelaksanaan Perancangan Aplikasi

Ada beberapa tahap yang dilakukan untuk melakukan perancangan aplikasi selama tugas akhir ini dibuat. Beberapa tahap tersebut digambarkan secara umum pada Gambar 3.4. Tahapan tersebut meliputi dari perencanaan, studi literatur, perancangan, implementasi, uji coba, dan evaluasi.



Gambar 3.4 Diagram Alur Perencanaan Aplikasi

Tahap pertama adalah perencanaan aplikasi. Di tahap ini, peneliti mengidentifikasi rumusan masalah yang ada, sehingga tujuan dari aplikasi deteksi katarak ini akhirnya ditemukan. Di tahap ini juga, perencanaan langkah-langkah pengerjaan tugas akhir juga dilakukan.

Tahap kedua adalah studi literatur. Di tahap ini, peneliti melakukan pencarian beberapa studi literatur dari berbagai jurnal penelitian terdahulu dan beberapa *website* pendukung untuk mempermudah pemahaman tentang aplikasi deteksi katarak yang akan peneliti bangun.

Tahap ketiga adalah perancangan aplikasi. Pada tahap ini, peneliti membuat rancangan aplikasi yang akan dibangun, mulai dari desain aplikasi, fitur-fitur yang digunakan, dan arsitektur sistem. Di tahap ini, proses pengumpulan dan klasifikasi *dataset* katarak dilakukan.

Tahap keempat adalah implementasi aplikasi. Di tahap ini, hasil perencanaan dan perancangan aplikasi yang telah disusun, diterapkan di tahap ini. Hasil dari pelatihan model pada *dataset* katarak mata juga diimplementasikan di tahap ini pada aplikasi yang dalam proses

pembangunan. Hasil dari implementasi aplikasi ini adalah sebuah aplikasi deteksi katarak mata yang siap diuji dan dievaluasi.

Tahap kelima, sekaligus menjadi tahap terakhir adalah pengujian dan evaluasi aplikasi. Pada tahap ini, aplikasi yang sudah dihasilkan kemudian dilakukan pengujian dan evaluasi apakah aplikasi tersebut telah memenuhi kriteria rumusan masalah dan tujuan aplikasi dibangun. Selain itu, di tahap ini, aplikasi juga akan diuji apakah terdapat beberapa *error* untuk menghasilkan aplikasi yang terbaik.

BAB 4 Hasil dan Pembahasan

4.1 Hasil Pembuatan Model Klasifikasi

Pada bagian ini, hasil dari pembuatan model klasifikasi katarak untuk aplikasi dijabarkan secara lengkap. Model klasifikasi katarak mata dibuat dengan memanfaatkan *machine learning*.

4.1.1 Instalasi Package

Sebelum melakukan pengolahan *dataset* untuk model klasifikasi yang digunakan pada tugas akhir ini, peneliti perlu melakukan instalasi pada beberapa *package* yang dibutuhkan untuk membantu proses klasifikasi *dataset*. Beberapa *package* yang dibutuhkan adalah cv2 untuk *environment* bekerja di dalam *python*, kemudian ada *Tensorflow*, Keras, dan Numpy untuk membantu melakukan model klasifikasi pada *dataset*. Terakhir, peneliti melakukan instalasi pada *package* matplotlib untuk menampilkan grafik akurasi dari hasil model klasifikasi.

4.1.2 Preparation dan Pre-Processing Data

Dataset yang digunakan adalah *dataset* yang berasal dari *website* penyedia *dataset* yang sifatnya *open source* bernama Kaggle. Pada tugas akhir ini, *dataset* yang digunakan merupakan hasil kontribusi dari Hemoo Redaoo dan Nandan Padiaa dkk. Dari kedua kontributor tersebut, kedua *dataset* ini digabungkan sehingga berjumlah 1099 gambar mata normal dan mata katarak. *Dataset* yang digunakan pada tugas akhir ini dibagi ke dalam dua kelas, yaitu katarak dan normal. Untuk jumlah gambar mata katarak ada 547 gambar dan gambar mata normal ada 552 gambar.

Dataset ini kemudian dilakukan *preparation data*. Tujuannya adalah untuk memastikan data yang digunakan untuk pelatihan data pada model klasifikasi bersih dari data yang tidak valid atau inkonsisten. Selain itu, *preparation data* juga dibutuhkan untuk menyeragamkan ukuran gambar data, sehingga data yang masuk dalam model klasifikasi memiliki ukuran gambar data yang sama dan konsisten. Pada tugas akhir ini, *preparation data* yang digunakan terlihat pada Kode 4.1

```
#Preparation Data
IMAGE_SIZE = (300,300)
BATCH_SIZE = 64
SEED = 999
```

Kode 4.1 *Preparation Data*

Pada Kode 4.1, *preparation data* yang digunakan adalah mengatur ukuran gambar, kumpulan data yang diproses selama pelatihan model klasifikasi dalam satu iterasi (*batch*), dan *seed*. Ukuran gambar yang digunakan pada proses pelatihan model klasifikasi adalah 300 x 300 piksel, kemudian *batch* yang digunakan ada 64 data, dan *seed* yang digunakan ada 999.

Setelah *Dataset* mengalami *preparation data*, selanjutnya *dataset* yang digunakan pada tugas akhir ini dilakukan *pre-processing* dengan menggunakan *ImageDataGenerator*. Setelah dilakukan *split validation*, *dataset* dibagi ke dalam *data training* dan *data validation*. Kode 4.2 menjelaskan cara *ImageDataGenerator* dapat melakukan *split data* sehingga terbagi menjadi *data training* dan *data validation*.

```
# Menggunakan ImageDataGenerator untuk preprocessing
import tensorflow as tf

datagen = tf.keras.preprocessing.image.ImageDataGenerator(
    validation_split=0.2
)

# Menyiapkan data train dan data validation
train_data = datagen.flow_from_directory(
    base_dir,
    class_mode='categorical',
    subset='training',
    target_size=IMAGE_SIZE,
    batch_size=BATCH_SIZE,
    seed=SEED
)

valid_data = datagen.flow_from_directory(
    base_dir,
    class_mode='categorical',
    subset='validation',
    target_size=IMAGE_SIZE,
    batch_size=BATCH_SIZE,
    seed=SEED
)
```

Kode 4.2 *Data Pre-Processing*

Setelah menjalankan Kode 4.2, hasil yang didapatkan adalah 880 gambar untuk *data training* dan 219 gambar untuk *data validation*. Masing-masing data dibagi ke dalam dua kelas, yaitu *cataract* dan *normal*.

Selanjutnya adalah *Image Augmentation*. *Image Augmentation* digunakan untuk memperbanyak jumlah data yang akan dilatih dalam melakukan model klasifikasi. Kode 4.3 menunjukkan *Image Augmentation* dilakukan pada *pre-processing* data.

```
# Image Augmentation
data_augmentation = tf.keras.Sequential(
  [
    tf.keras.layers.InputLayer(shape=(IMAGE_SIZE[0], IMAGE_SIZE[1], 3)),
    tf.keras.layers.RandomFlip("horizontal"),
    tf.keras.layers.RandomRotation(0.1),
    tf.keras.layers.RandomZoom(0.1),
    tf.keras.layers.Rescaling(1./255)
  ]
)
```

Kode 4.3 *Image Augmentation*

Pada Kode 4.3, dapat dilihat bahwa *Image Augmentation* yang digunakan selama melakukan pelatihan pada model klasifikasi adalah membalik gambar secara horizontal, memutar gambar secara acak hingga 10% kemiringan dari gambar *dataset*, dan memperbesar atau memperkecil gambar *dataset* secara acak hingga 10%. Hal-hal ini dilakukan agar model klasifikasi dapat meningkatkan akurasi hasil akhir.

4.1.3 Persiapan Model Klasifikasi

Pada bagian ini, *dataset* yang telah dilakukan *pre-processing* siap untuk dilakukan proses model klasifikasi. Pada tugas akhir ini, model klasifikasi yang digunakan ada empat macam, yaitu *Convolutional Neural Network* (CNN), VGG16, ResNet50, dan DenseNet201. Semua model klasifikasi akan digunakan untuk menentukan hasil akurasi dari model klasifikasi yang terbaik.

4.1.3.1 Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) adalah salah satu model klasifikasi yang paling sering digunakan dalam melakukan pelatihan model klasifikasi pada suatu *dataset*. Pada tugas akhir ini, model klasifikasi CNN pertama memasukkan dulu *dataset* yang telah mengalami *Image Augmentation*. Setelah itu, *dataset* diproses ke dalam beberapa *layer*. Beberapa *layer* yang dimaksud adalah sebagai berikut:

1. *Layer* Konvolusi
   Layer Konvolusi ditunjukkan pada baris Kode 4.4
   ```
   tf.keras.layers.Conv2D(32, 3, padding='same', activation='relu'),
   tf.keras.layers.MaxPooling2D(),
   tf.keras.layers.Conv2D(64, 3, padding='same', activation='relu'),
   tf.keras.layers.MaxPooling2D(),
   tf.keras.layers.Conv2D(64, 3, padding='same', activation='relu'),
   tf.keras.layers.MaxPooling2D(),
   ```
   Kode 4.4 *Layer* Konvolusi CNN
   Layer Konvolusi adalah inti dari CNN. Operasi konvolusi diterapkan pada gambar untuk mendeteksi fitur-fitur seperti tepi, sudut atau pola tertentu pada *dataset*. *Layer* konvolusi dibuat sebanyak tiga *layer* untuk menangkap fitur yang lebih kompleks pada *dataset*. Setelah itu, *layer-layer* ini diaktifkan menggunakan fungsi aktivasi ReLU untuk memperkenalkan non-linearitas ke dalam model, sehingga jaringan dapat belajar pola kompleks.
2. *Layer Pooling*
   Layer pooling yang digunakan adalah `tf.keras.layers.MaxPooling2D()`. *Layer* ini digunakan untuk mengurangi dimensi data sambil mempertahankan fitur penting, sehingga proses komputasi pada model klasifikasi menjadi lebih cepat. Dalam hal ini, nilai maksimum *pooling* yang digunakan adalah 2x2. *Layer pooling* digunakan setelah setiap *layer* konvolusi berjalan, untuk mengecilkan dimensi secara bertahap.
3. *Layer Dropout*
   Model klasifikasi ini menggunakan *layer dropout* untuk mencegah *overfitting*. *Layer dropout* pada model CNN ini menonaktifkan 30% neuron selama pelatihan, sehingga mendorong model untuk belajar fitur yang lebih umum daripada belajar terlalu banyak pola spesifik, termasuk data yang tidak relevan (*noise* data) hingga menyebabkan model mengalami kinerja buruk pada *data validation*,
4. *Layer Flatten*
   Layer ini dibutuhkan untuk mengubah keluaran data 2D menjadi vektor 1D, untuk menjadi penghubung antara *layer* konvolusi ke *layer fully connected*.
5. *Layer Fully Connected* (*Layer Dense*)
   Layer dense bertujuan untuk menghubungkan semua neuron dari *layer* sebelumnya ke neuron berikutnya. Pada model klasifikasi ini, *layer* yang digunakan adalah `tf.keras.layers.Dense(64, activation='relu')` sebanyak dua *layer*.
6. *Layer Output*

Layer output yang digunakan adalah `tf.keras.layers.Dense(2, activation='softmax')`. *Layer* ini memiliki dua neuron, sesuai dengan jumlah kelas (*cataract* dan *normal*). Karena menggunakan *activation softmax*, untuk setiap kelas yang dilakukan pelatihan model klasifikasi, *output* yang dikeluarkan bernilai antara 0 dan 1 untuk setiap kelas.

4.1.3.2 VGG16

VGG16 adalah model klasifikasi CNN yang lebih sederhana secara arsitektur. Model ini hanya menghitung *layer* konvolusi dan *dense* sebagai *layer* utama, sedangkan *layer pooling* tidak dihitung. VGG16 telah dilatih pada *dataset* besar seperti ImageNet. Model *pre-trained* dari VGG16 dapat dimanfaatkan untuk *transfer learning*, memanfaatkan bobot yang telah dilatih. Kode 4.5 menunjukkan VGG16 digunakan pada tugas akhir ini.

```
import tensorflow as tf
from tensorflow.keras.applications.vgg16 import VGG16

## Loading VGG16 model
base_vgg_model = VGG16(weights="imagenet", include_top=False,
input_shape=(IMAGE_SIZE[0], IMAGE_SIZE[1], 3))
base_vgg_model.trainable = False

# Preprocessing Input
vgg_preprocess = tf.keras.applications.vgg16.preprocess_input
train_data.preprocessing_function = vgg_preprocess

# Transfer learning dengan VGG16
vgg_model = tf.keras.models.Sequential([
  data_augmentation,
  base_vgg_model,
  tf.keras.layers.Dropout(0.7),
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(64, activation='relu'),
  tf.keras.layers.Dense(64, activation='relu'),
  tf.keras.layers.Dense(2, activation='softmax')
])
```

Kode 4.5 Model Klasifikasi VGG16

Berdasarkan pada Kode 4.5, *input* dari model klasifikasi VGG16 pada tugas akhir ini adalah *dataset* yang telah mengalami *Image Augmentation*. Model VGG16 memuat bobot yang sudah dilatih pada *dataset* ImageNet. Kemudian, pada kode `include_top=False`, model klasifikasi tugas akhir ini menghapus *layer dense* di bagian atas model, sehingga model klasifikasi tugas akhir ini hanya mengambil bagian *feature extractor*. Bagian tersebut berisi *layer* konvolusi dan *layer pooling* yang berguna untuk mengekstrak fitur dari gambar *dataset* peneliti.

Pada kode `base_vgg_model.trainable = False`, model klasifikasi tugas akhir ini membekukan *layer* VGG16 agar bobot dari *dataset* ImageNet tidak mengalami pembaruan selama pelatihan. Hal ini bertujuan untuk memanfaatkan fitur yang dipelajari oleh model VGG16 tanpa melatih ulang bagian ini.

Untuk melakukan pelatihan model klasifikasi VGG16 pada tugas akhir ini, model perlu melakukan *pre-processing* ulang khusus VGG16 untuk mengubah nilai piksel gambar *dataset*

peneliti menjadi format yang sesuai dengan *dataset* ImageNet pada VGG16. Setelah melakukan normalisasi *dataset*, model klasifikasi VGG16 pada tugas akhir ini masuk ke bagian *transfer learning*. Sebelum masuk ke *layer*, *dataset Image Augmentation* masuk sebagai *input* dari *transfer learning*. Selanjutnya, proses dilanjutkan ke beberapa *layer* yang dijelaskan seperti berikut:

1. `base_vgg_model`
   Layer ini digunakan untuk memasukkan model VGG16 tanpa bagian atas model, menjadikan *layer* ini berfungsi sebagai *feature extractor* untuk *input* gambar.
2. *Layer Dropout*
   Layer ini menggunakan 70% probabilitas *dropout*, tujuannya adalah membantu mencegah *overfitting* dengan menonaktifkan 70% neuron secara acak selama pelatihan berlangsung.
3. *Layer Flatten*
4. *Layer* ini dibutuhkan untuk mengubah *output* 3D dari *layer* `base_vgg_model` menjadi vektor 1D untuk menjadi penghubung ke *layer fully connected*.
5. *Layer Fully Connected* (*Dense*)
   Layer ini bertujuan untuk menghubungkan semua neuron dari *layer* sebelumnya ke neuron berikutnya. Pada model klasifikasi ini, *layer* yang digunakan sebanyak dua *layer*.
6. *Layer Output*
   Layer ini memiliki dua neuron, sesuai dengan jumlah kelas (*cataract* dan *normal*). Karena menggunakan *activation softmax*, untuk setiap kelas yang dilakukan pelatihan model klasifikasi, *output* yang dikeluarkan bernilai antara 0 dan 1 untuk setiap kelas.

4.1.3.3 ResNet50

ResNet50 adalah salah satu bagian dari arsitektur CNN. Model klasifikasi ini menggunakan *residual connection* untuk membantu jaringan mempelajari perbedaan antara *input* dengan *output* target. Perbedaan ini disebut residu. Untuk mempelajari residu, ResNet50 melakukan dengan cara menambahkan *shortcut* atau *skip connection* yang melewati beberapa *layer*. Tujuannya adalah untuk mengatasi masalah kehilangan informasi gradien saat jaringan melakukan pembelajaran pada jaringan yang sangat dalam (*vanishing gradient*). Kode 4.6 menunjukkan model klasifikasi ResNet50 digunakan pada tugas akhir ini.

```
from tensorflow.keras.applications import ResNet50

# Loading ResNet50 model
base_resnet_model = ResNet50(include_top=False,
                    input_shape=(IMAGE_SIZE[0],IMAGE_SIZE[1],3),
                    pooling='max',classes=5,
                    weights='imagenet')

base_resnet_model.trainable = False

train_data.preprocessing_function = tf.keras.applications.resnet50.preprocess_input


# Transfer learning ResNet50
resnet_model = tf.keras.models.Sequential([
    data_augmentation,
    base_resnet_model,
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(64, activation="relu"),
    tf.keras.layers.Dense(64, activation="relu"),
    tf.keras.layers.Dense(2, activation="softmax")
])
```

Kode 4.6 Model Klasifikasi ResNet50

Model klasifikasi ResNet50 pada tugas akhir ini diawali dengan memuat model ResNet50 terlebih dahulu. Kemudian, model yang dirancang pada tugas akhir ini menghapus *layer dense* (*fully connected layer*) pada bagian atas yang digunakan untuk klasifikasi pada *dataset* ImageNet. Hal ini ditunjukkan pada kode `include_top=False` yang bertujuan untuk mengambil bagian *layer* konvolusi pada model saja, yang akhirnya digunakan sebagai *feature extractor*. Kemudian, setelah *layer* konvolusi selesai menjalankan pembelajaran, hasil fitur akan direduksi menjadi 1D dengan adanya `pooling='max'`. Rancangan model klasifikasi ResNet50 juga menggunakan bobot hasil pelatihan model ResNet50 pada *dataset* ImageNet.

Selanjutnya, model klasifikasi ResNet50 pada tugas akhir ini masuk ke tahap *pre-processing input*. Sebelum masuk ke *pre-processing*, model klasifikasi ResNet50 pada tugas akhir ini membekukan *layer base model* pada kode `base_resnet_model.trainable = False`. Tujuannya adalah agar semua bobot pada *base model* ResNet50 tidak berubah selama proses pelatihan. Hal ini dilakukan supaya model klasifikasi ResNet50 pada tugas akhir ini hanya melatih *layer* baru di atasnya, sehingga model klasifikasi ResNet50 pada tugas akhir ini dapat memanfaatkan fitur-fitur umum yang sudah dipelajari dari *dataset ImageNet*. Lalu, masuk ke tahap *pre-processing input*, fungsi ini diterapkan ke data sebelum dimasukkan ke model klasifikasi, dengan tujuan untuk memproses secara otomatis gambar agar sesuai dengan normalisasi yang digunakan oleh model ResNet50.

Setelah melalui proses *pre-processing*, model klasifikasi ResNet50 pada tugas akhir ini masuk ke tahap *transfer learning*. Ada beberapa *layer* yang digunakan, yaitu sebagai berikut:

1. `base_resnet_model`
   Layer ini berisi tentang *base* model dari ResNet50, digunakan untuk melakukan ekstraksi fitur dari gambar *input*. *Base* model ini sudah dilatih pada *dataset* ImageNet.
2. *Layer Flatten*
3. *Layer* ini mengubah *output* dibutuhkan untuk mengubah *output* 3D dari *layer* `base_resnet_model` menjadi vektor 1D untuk menjadi penghubung ke *layer fully connected* (*layer dense*).
4. *Layer Dense*

Layer ini bertujuan untuk menghubungkan semua neuron dari *layer* sebelumnya ke neuron berikutnya. Pada model klasifikasi ini, *layer* yang digunakan sebanyak dua *layer*.

4.1.3.4 DenseNet201

DenseNet201 adalah salah satu pengembangan dari model klasifikasi CNN. DenseNet201 adalah model yang dirancang untuk meningkatkan efisiensi aliran informasi dan gradien di dalam jaringan CNN. Setiap *layer* dalam DenseNet terhubung secara langsung dengan semua *layer* sebelumnya. *Output* dari setiap *layer* digunakan sebagai *input* untuk semua *layer* berikutnya di dalam blok yang sama, sehingga DenseNet membutuhkan lebih sedikit parameter karena tidak melakukan duplikasi fitur yang sama di beberapa *layer*. Kode 4.7 menunjukkan model klasifikasi DenseNet201 digunakan pada tugas akhir ini.

```
# Loading DenseNet201 model
base_densenet_model = tf.keras.applications.DenseNet201(include_top=False,
                                                         weights='imagenet',
                                                         input_shape=(IMAGE_SIZE[0],
                                                         IMAGE_SIZE[1], 3),
                                                         pooling='max')
base_densenet_model.trainable=False
train_data.preprocessing_function = tf.keras.applications.densenet.preprocess_input

# Transfer learning DenseNet201
densenet_model = tf.keras.models.Sequential([
  data_augmentation,
  base_densenet_model,
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(64, activation='relu'),
  tf.keras.layers.Dense(64, activation='relu'),
  tf.keras.layers.Dense(2, activation='softmax')
])
```

Kode 4.7 Model Klasifikasi DenseNet201

Berdasarkan pada Kode 4.7, model klasifikasi DenseNet201 pada tugas akhir ini dimulai dari memuat model DenseNet201 terlebih dahulu. Kemudian model klasifikasi menghapus *layer fully connected* (ditunjukkan pada kode `include_top=False`) untuk menyesuaikan dengan *dataset* yang peneliti gunakan. Lalu, model klasifikasi DenseNet201 menggunakan bobot yang telah dilakukan *pre-trained* dari *dataset* ImageNet, yang dapat memungkinkan model klasifikasi memanfaatkan fitur yang telah dipelajari sebelumnya. Setelah *layer* selesai menjalankan pembelajaran, hasil fitur akan direduksi menjadi 1D dengan adanya `pooling='max'`.

Model klasifikasi DenseNet201 kemudian masuk ke tahap *pre-processing*. Namun sebelum masuk ke tahap *pre-processing*, *layer base* model DenseNet201 dibekukan terlebih dahulu, agar bobot pada *base* model DenseNet201 tidak diperbarui selama pelatihan. Hal ini dilakukan untuk mempertahankan fitur-fitur yang telah dipelajari dari *dataset* ImageNet, mengurangi risiko *overfitting*, dan dapat mempercepat pelatihan. Selanjutnya, model klasifikasi DenseNet201 melakukan *pre-processing*. Fungsi ini diterapkan ke data sebelum dimasukkan ke model klasifikasi, dengan tujuan untuk memproses secara otomatis gambar agar sesuai dengan normalisasi yang digunakan oleh model DenseNet201.

Setelah melalui proses *pre-processing*, model klasifikasi DenseNet201 pada tugas akhir ini masuk ke tahap *transfer learning*. Ada beberapa *layer* yang digunakan, yaitu sebagai berikut:

1. `base_densenet_model`

Layer ini berisi tentang *base* model dari DenseNet201, digunakan untuk melakukan ekstraksi fitur dari gambar *input*. *Base* model ini sudah dilatih pada *dataset* ImageNet.

2. *Layer Dropout*
   Layer ini menggunakan 20% *dropout* untuk mengurangi *overfitting* dengan secara acak menonaktifkan beberapa neuron selama pelatihan.
3. *Layer Flatten*
   Layer ini mengubah *output* dibutuhkan untuk mengubah *output* 2D dari *layer* `base_densenet_model` menjadi vektor 1D untuk menjadi penghubung ke *layer fully connected* (*layer dense*).
4. *Layer Dense*
   Layer ini bertujuan untuk menghubungkan semua neuron dari *layer* sebelumnya ke neuron berikutnya. Pada model klasifikasi ini, *layer* yang digunakan sebanyak dua *layer* dengan fungsi aktivasi ReLU untuk mempelajar pola non-linear.
5. *Layer Output*
   Layer ini memiliki dua neuron, sesuai dengan jumlah kelas (*cataract* dan *normal*). Karena menggunakan *activation softmax*, untuk setiap kelas yang dilakukan pelatihan model klasifikasi, *output* yang dikeluarkan bernilai antara 0 dan 1 untuk setiap kelas.

4.1.4 Pelatihan Data pada Model Klasifikasi

Setelah mempersiapkan model klasifikasi yang digunakan pada tugas akhir ini, tahap selanjutnya adalah melakukan pelatihan model klasifikasi. Tugas akhir ini menggunakan beberapa parameter yang digunakan untuk semua model klasifikasi yang dilatih, yaitu *loss function*, *optimizer*, dan *metric*. Parameter selengkapnya dapat dilihat pada Tabel 4.1.

Parameter	Nilai Parameter
Loss function	Categorical_crossentropy
Optimizer	Adam
Metric	Metric accuracy

Tabel 4.1 Parameter Model Klasifikasi

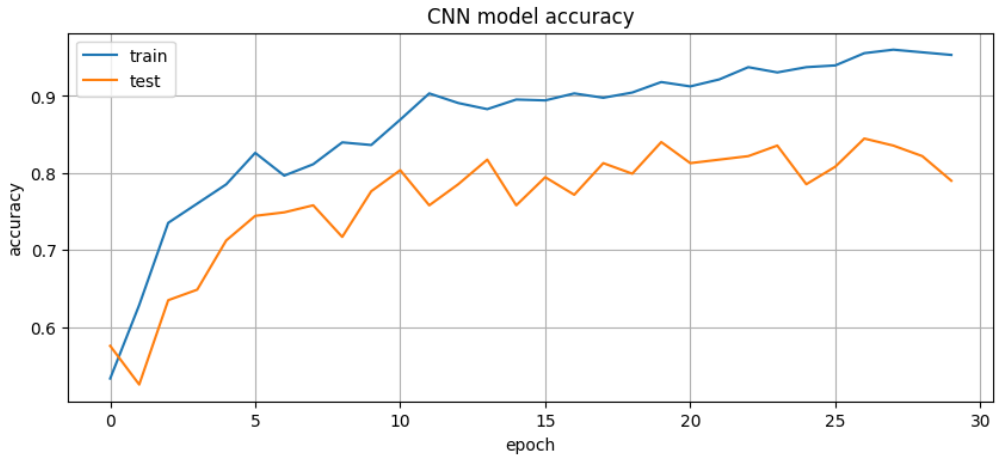
Semua model klasifikasi yang digunakan pada tugas akhir ini akan dilakukan pelatihan dengan *epoch* sejumlah 30. Hasil dari pelatihan model klasifikasi akan menunjukkan sejumlah variabel tentang *accuracy, loss, val_accuracy,* dan *val_loss*. Selain itu, peneliti akan menggunakan *library* matplotlib untuk menampilkan grafik model akurasi dan *loss model* dari semua model klasifikasi.

Tabel 4.2 menunjukkan hasil dari pelatihan model klasifikasi pada tugas akhir ini. Nilai yang ditunjukkan adalah nilai tertinggi untuk akurasi dan nilai terendah untuk *loss model* yang didapatkan dari setiap *epoch* yang telah dilakukan.

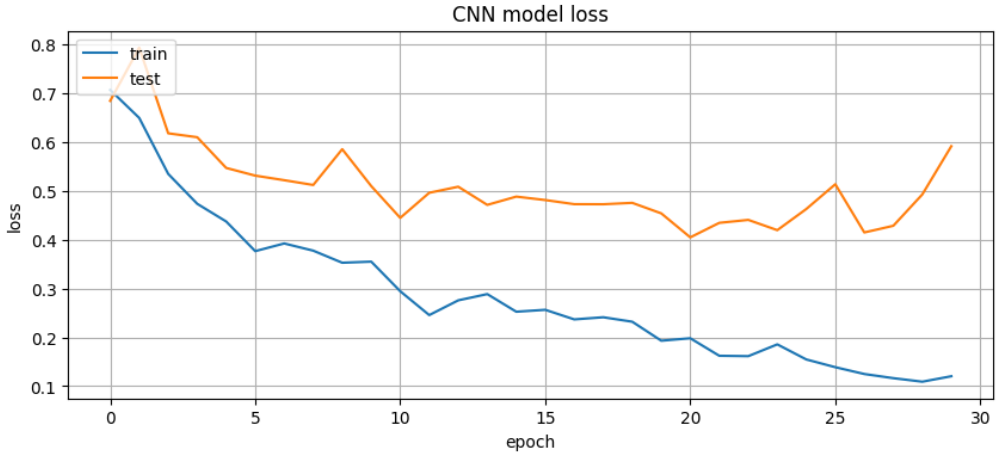
No.	Model Klasifikasi	val_accuracy	val_loss
1.	CNN	0.8219	0.4448
2.	VGG16	0.9315	0.2191
3.	ResNet50	0.5936	0.7840
4.	DenseNet201	0.9361	0.1899

Tabel 4.2 Hasil Pelatihan Model Klasifikasi

Berikut ini adalah grafik hasil akurasi model dan *loss model* dari model klasifikasi CNN, ditunjukkan pada Gambar 4.1 dan Gambar 4.2.

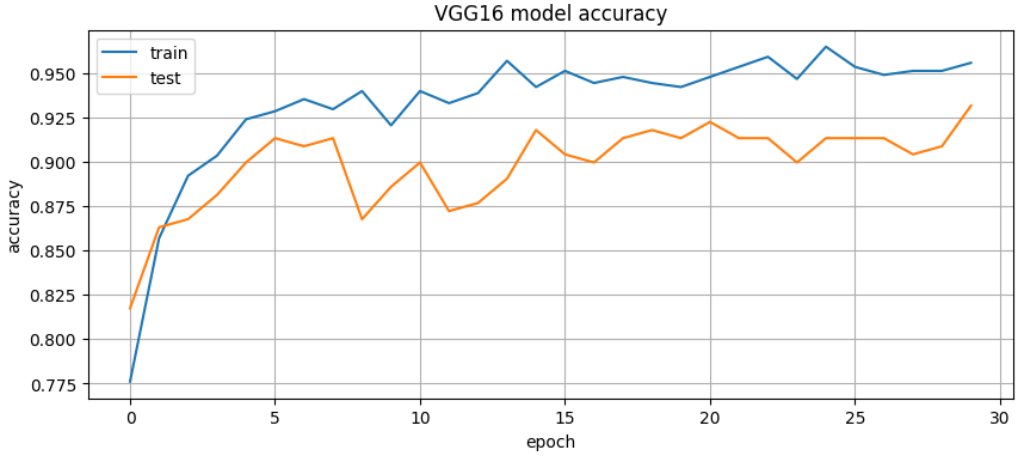


Gambar 4.1 Akurasi Model Klasifikasi CNN

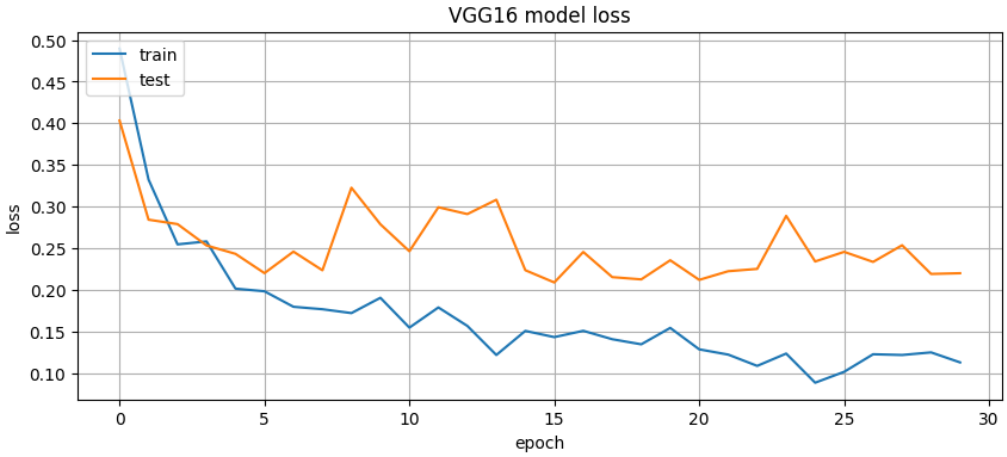


Gambar 4.2 *Loss Model* dari Model Klasifikasi CNN

Selanjutnya, berikut ini adalah grafik hasil akurasi model dan *loss model* dari model klasifikasi VGG16, ditunjukkan pada Gambar 4.3 dan Gambar 4.4.

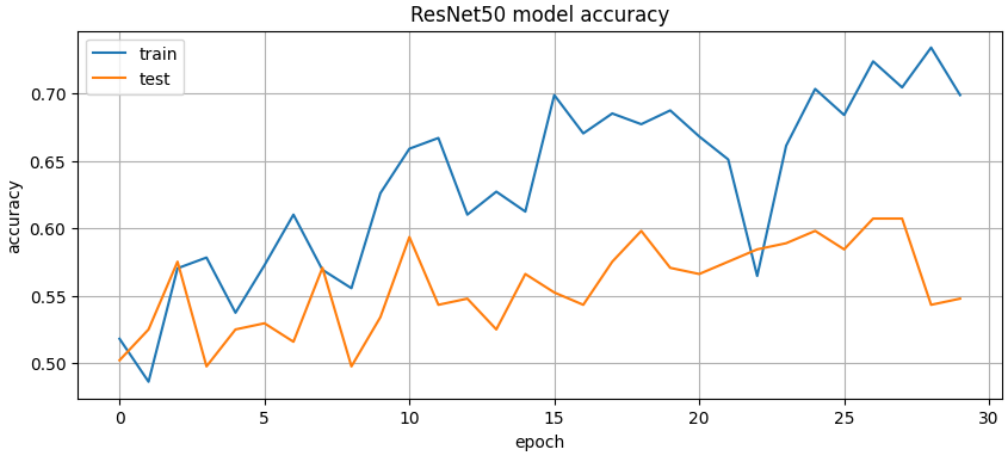


Gambar 4.3 Akurasi Model Klasifikasi VGG16



Gambar 4.4 *Loss Model* dari Model Klasifikasi VGG16

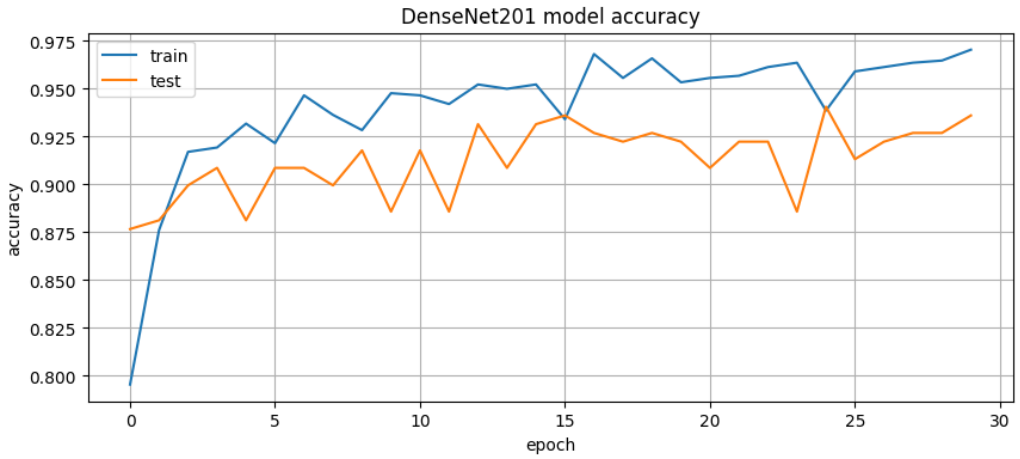
Selanjutnya, berikut ini adalah grafik hasil akurasi model dan *loss model* dari model klasifikasi ResNet50, ditunjukkan pada Gambar 4.5 dan Gambar 4.6.



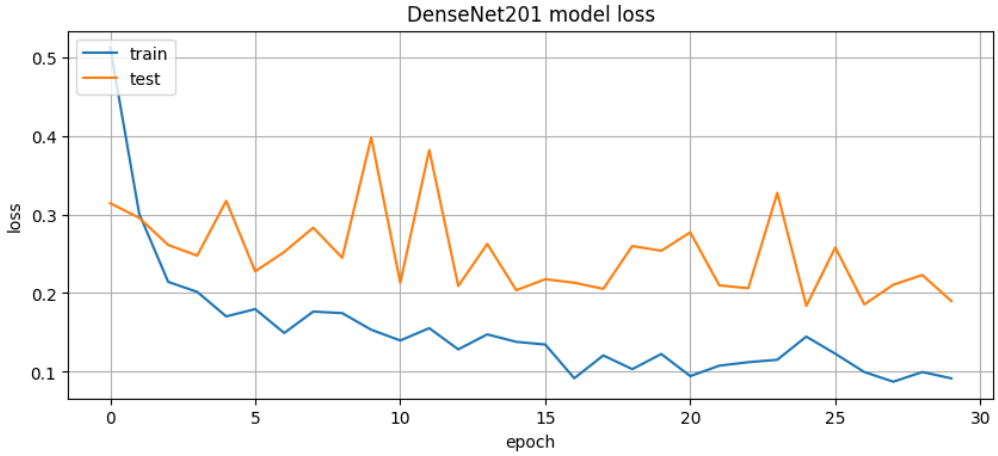
Gambar 4.5 Akurasi Model Klasifikasi ResNet50

Gambar 4.6 *Loss Model* dari Model Klasifikasi ResNet50

Terakhir, berikut ini adalah grafik hasil akurasi model dan *loss model* dari model klasifikasi DenseNet201, ditunjukkan pada Gambar 4.7 dan Gambar 4.8.

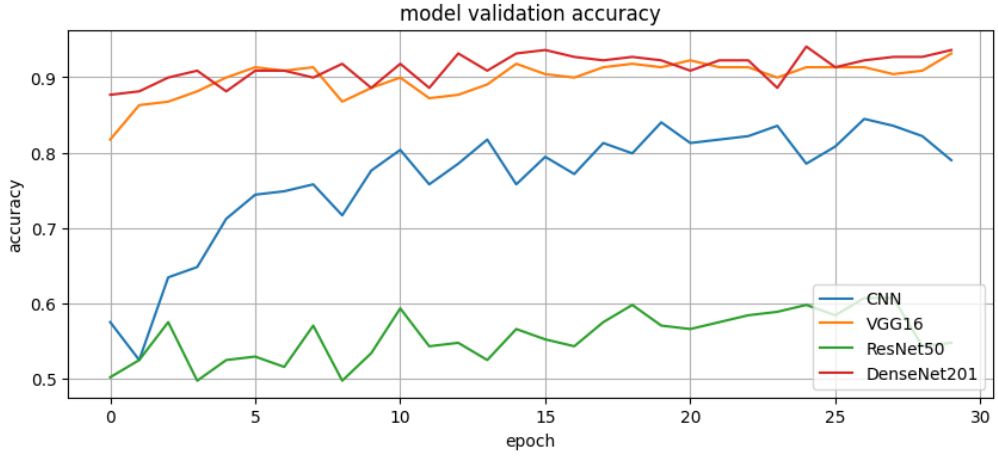


Gambar 4.7 Akurasi Model Klasifikasi DenseNet201



Gambar 4.8 *Loss Model* dari Model Klasifikasi DenseNet201

Berikut ini adalah perbandingan hasil akurasi dari semua model klasifikasi yang telah dilakukan pelatihan, digambarkan melalui grafik pada Gambar 4.9.



Gambar 4.9 Perbandingan Akurasi Model Klasifikasi

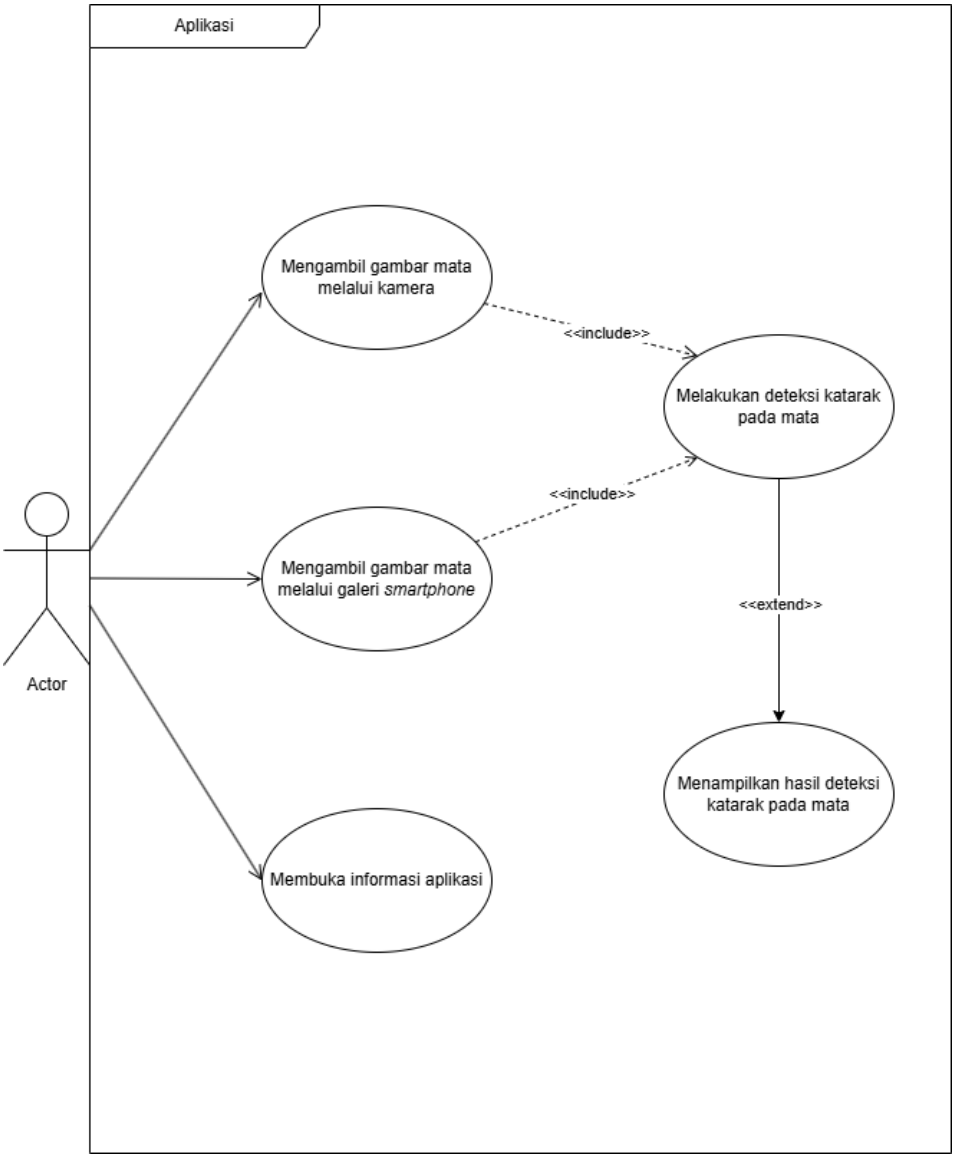
Berdasarkan hasil pelatihan dari semua model klasifikasi yang digunakan pada tugas akhir ini, ditunjukkan bahwa model klasifikasi DenseNet201 memiliki hasil akurasi yang terbaik dan konsisten jika dilihat dari grafik pada Gambar 4.9. Model klasifikasi DenseNet201 mencapai *val_accuracy* sebesar 93,61% dan *val_loss* terkecil sebesar 18,99%. Oleh karena itu, model klasifikasi DenseNet201 dipilih sebagai model klasifikasi untuk aplikasi deteksi katarak mata pada tugas akhir ini.

4.2 Hasil Pembuatan Aplikasi

Pada bagian ini, hasil dari pembuatan aplikasi deteksi katarak pada mata dijabarkan secara lengkap. Aplikasi ini berbasis *android* dengan menggunakan hasil perancangan yang telah dijelaskan pada Bab 3 tentang Metodologi.

4.2.1 Hasil Analisa

Pada tahap analisa, peneliti melakukan analisa terhadap kebutuhan yang dibutuhkan dalam membangun aplikasi. Analisa ini menghasilkan sebuah diagram *use case* yang dibutuhkan dalam aplikasi. Gambar 4.10 akan menjelaskan diagram *use case* yang dibutuhkan.

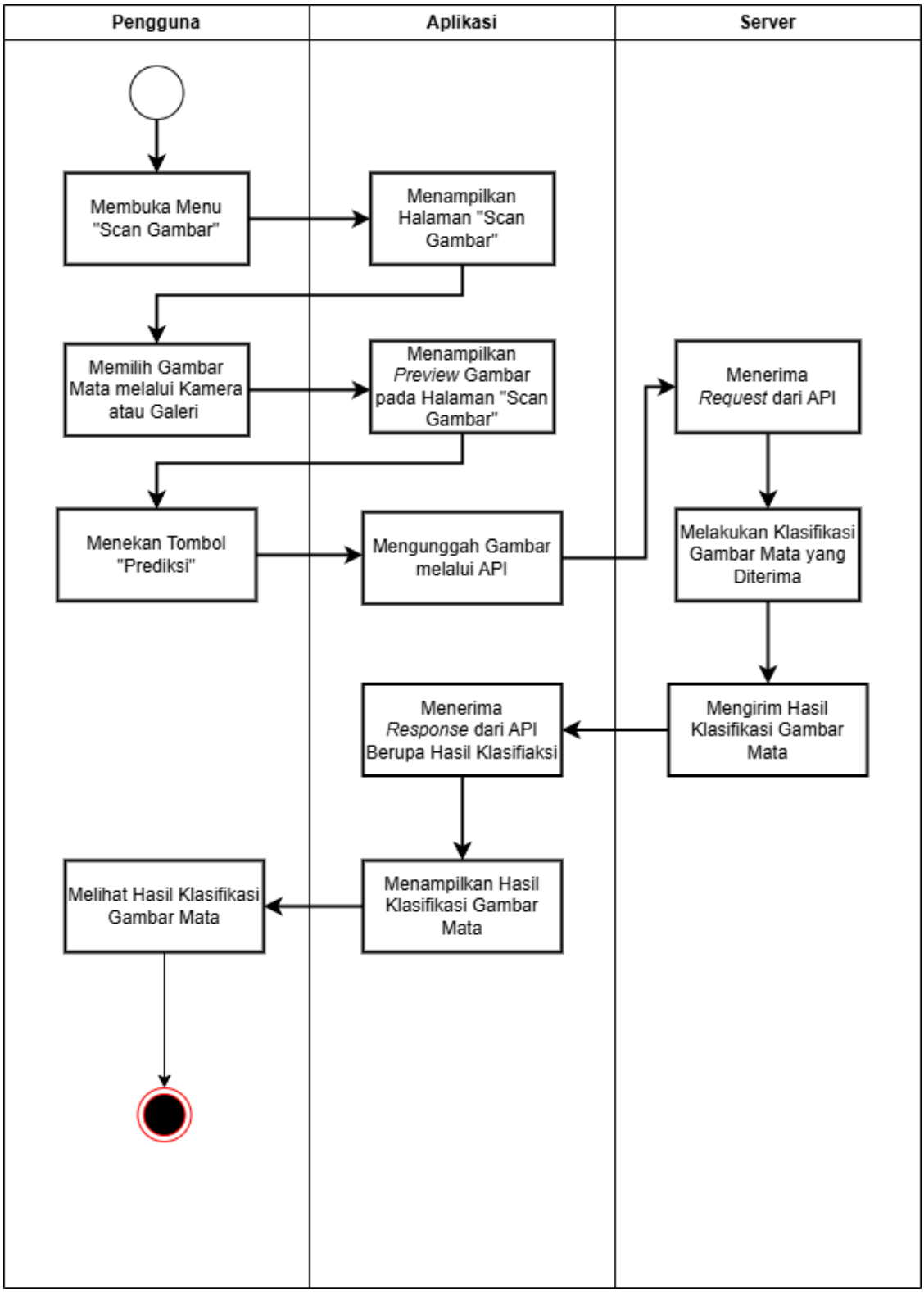


Gambar 4.10 Diagram *Use Case* Aplikasi

Gambar 4.10 menunjukkan bahwa terdapat 3 *use case* yang digunakan pada tugas akhir ini, yaitu mengambil gambar melalui kamera, mengambil gambar melalui galeri *smartphone*, dan membuka informasi aplikasi. Penjelasan tentang diagram *use case* adalah sebagai berikut:

1. Mengambil Gambar Mata melalui Kamera
   Pengguna aplikasi mengambil gambar mata melalui kamera *smartphone* pengguna ke aplikasi. Kemudian, aplikasi mengirimkan hasil gambar mata pengguna menggunakan API ke *server* yang telah dirancang untuk mendeteksi katarak.
2. Mengambil Gambar Mata melalui Galeri *Smartphone*
   Pengguna aplikasi mengambil gambar mata melalui galeri *smartphone* pengguna ke aplikasi. Kemudian, aplikasi mengirimkan hasil gambar mata pengguna menggunakan API ke *server* yang telah dirancang untuk mendeteksi katarak.
3. Melakukan Deteksi Katarak
   Sistem pada aplikasi menerima gambar mata yang telah dikirim oleh pengguna. Kemudian, sistem akan melakukan klasifikasi pada gambar mata yang telah dikirim.
4. Menampilkan Hasil Deteksi Katarak pada Mata
   Pengguna dapat melihat hasil klasifikasi gambar mata yang telah dilakukan prediksi oleh aplikasi dan mendapatkan hasil berupa katarak atau normal.
5. Membuka Informasi Aplikasi
   Pengguna dapat melihat informasi tentang aplikasi yang dibangun, untuk memberikan informasi tentang tujuan aplikasi ini dibuat dan beberapa himbauan kepada pengguna apabila aplikasi mendeteksi adanya katarak pada mata.

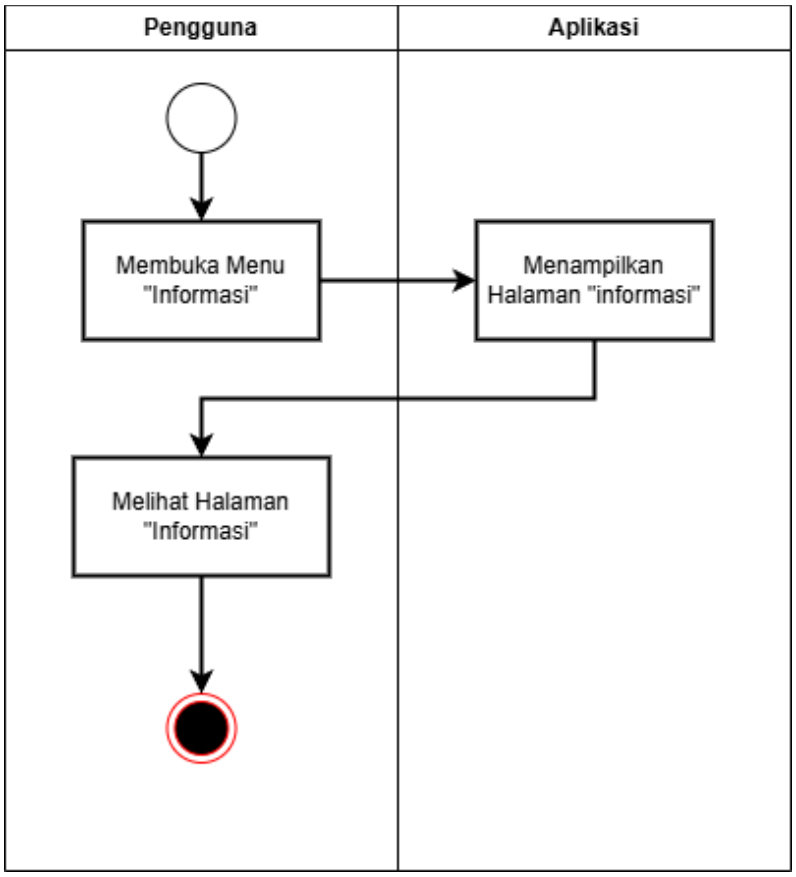
Setelah menggambarkan diagram *use case* untuk aplikasi, pada tugas akhir ini juga membuat gambar tentang diagram alur aplikasi yang digunakan. Penjelasan dari diagram alur aplikasi terdapat pada Gambar 4.11.



Gambar 4.11 Diagram Alur Aplikasi Menu Scan Gambar

Berdasarkan pada Gambar 4.11, digambarkan bahwa alur aplikasi dimulai dari pengguna yang membuka menu “Scan Gambar” terlebih dahulu. Kemudian, aplikasi mengarahkan pengguna ke halaman prediksi gambar. Di halaman ini, pengguna dapat memulai prediksi mata pengguna dengan cara mengirimkan gambar mata yang ingin dilakukan deteksi ke aplikasi. Setelah itu, aplikasi menampilkan *preview* gambar pada halaman prediksi gambar untuk

memastikan pengguna bahwa gambar mata yang diinginkan untuk dilakukan deteksi adalah gambar yang benar. Kemudian, pengguna menekan tombol “Prediksi” untuk melakukan deteksi gambar mata pada aplikasi. Gambar mata yang telah diterima oleh aplikasi kemudian diteruskan kepada *server* melalui *request* kepada API. Setelah *server* menerima gambar mata yang akan dilakukan prediksi, *server* melakukan klasifikasi pada model klasifikasi yang terdapat pada *server*. Hasil dari klasifikasi pada gambar mata kemudian dikirim ke pengguna melalui *response* API. Aplikasi yang telah menerima *response* dari API berupa hasil klasifikasi gambar mata lalu menampilkan hasilnya pada aplikasi, sehingga pengguna dapat melihat hasil klasifikasi gambar mata yang diinginkan untuk dilakukan prediksi.



Gambar 4.12 Diagram Alur Aplikasi Menu Informasi

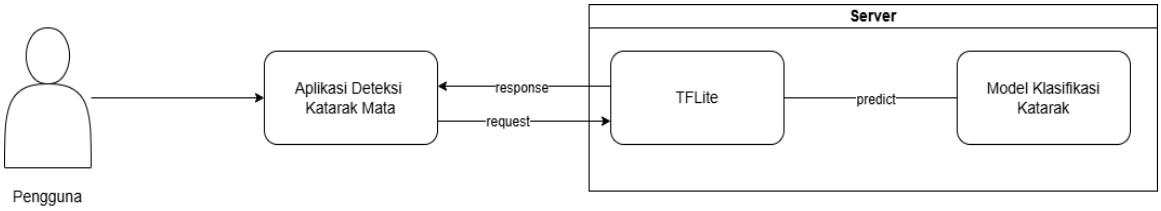
Berdasarkan pada Gambar 4.12, diagram alur aplikasi pada bagian menu informasi dimulai ketika pengguna membuka menu “Informasi”. Kemudian, aplikasi merespons dengan menampilkan halaman informasi, sehingga pengguna dapat melihat informasi tentang aplikasi.

4.2.2 Hasil Perencanaan

Pada tahap perencanaan, peneliti melakukan perencanaan terhadap perencanaan terhadap aplikasi yang akan dibangun, sesuai dengan hasil analisa yang telah dilakukan pada tahap sebelumnya. Perencanaan di tahap ini meliputi desain arsitektur sistem yang digunakan.

Pada tugas akhir ini, desain arsitektur sistem yang digunakan adalah *client-server*. Tugas akhir ini terdapat *server* yang berfungsi sebagai pengolah data yang diminta oleh *client*. Pada kasus ini, aplikasi *android* bertindak sebagai *client* yang akan melakukan *request* ke *server*. Kemudian, *server* mengelola *request* dengan hasilnya adalah hasil klasifikasi gambar mata.

Hasil ini kemudian dikirim ke *client* sebagai *response*. Gambar 4.13 menunjukkan desain arsitektur sistem yang digunakan pada tugas akhir ini.



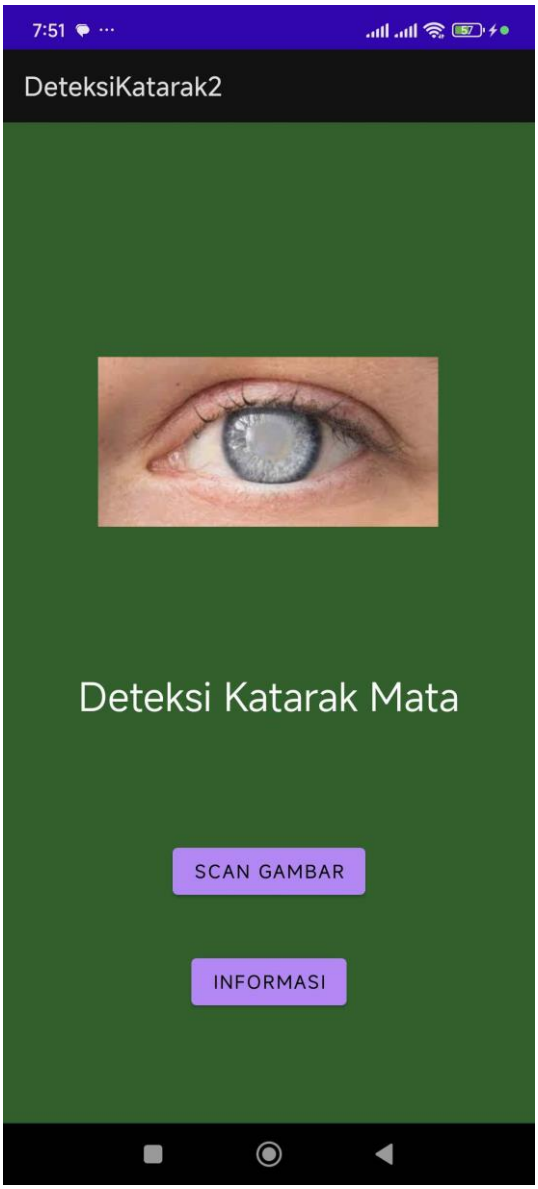
Gambar 4.13 Desain Arsitektur Sistem yang Digunakan

Berdasarkan pada Gambar 4.13, diketahui bahwa aplikasi *android* yang bertugas sebagai *client* akan melakukan *upload file* gambar mata ke *server* sebagai *request* melalui API. Pada tugas akhir ini, API yang digunakan adalah TFLite sebagai penghubung antara aplikasi dengan model klasifikasi katarak. Ketika API telah menerima *request* dari aplikasi, API meneruskan ke model klasifikasi katarak untuk dilakukan prediksi. Setelah model klasifikasi katarak menyelesaikan hasil prediksinya, *server* menyampaikan hasilnya ke aplikasi *android* melalui API sebagai *response*. Pada akhirnya, pengguna dapat melihat hasil *response* dari *server* berupa hasil klasifikasi yang terdiri dari "Katarak" atau "Normal".

4.2.3 Hasil Desain Aplikasi

Pada tahap desain aplikasi, peneliti melakukan desain terhadap aplikasi yang akan dibangun agar dapat membangun aplikasi sesuai dengan hasil analisa dan perencanaan yang telah dibahas pada subbab 4.2.1 tentang Hasil Analisa dan 4.2.2 tentang Hasil Perencanaan. Diketahui bahwa aplikasi harus dapat menampilkan gambar yang didapatkan dari hasil ambil gambar melalui kamera atau dari galeri, untuk dilakukan prediksi terhadap model klasifikasi katarak pada mata. Kemudian, aplikasi juga harus memberikan informasi tentang keberhasilan prediksi gambar, kategori kelas gambar yang diprediksi, dan pesan *error* apabila gambar tidak dapat diambil. Aplikasi ini juga memberikan edukasi dan informasi tentang tujuan, kritik, dan saran aplikasi ini dibangun.

Ketika pertama kali membuka aplikasi, pengguna akan diarahkan ke halaman utama dari aplikasi yang terdiri dari *button* "Scan Gambar" dan "Informasi". *Button* ini berfungsi untuk mengarahkan pengguna ke halaman prediksi gambar atau halaman informasi sesuai kebutuhan pengguna. Desain aplikasinya dapat dilihat pada Gambar 4.14.

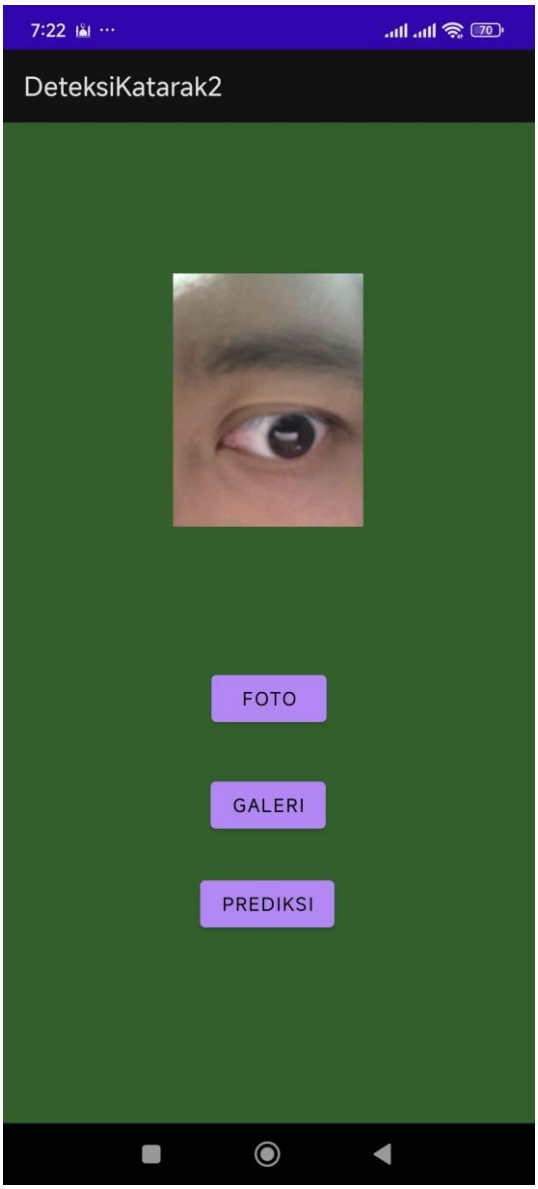


Gambar 4.14 Halaman Utama Aplikasi

Ketika masuk ke halaman prediksi gambar, aplikasi akan menunjukkan sebuah kotak kosong dan tiga *button*, yaitu “Foto”, “Galeri”, dan “Prediksi”. Kotak kosong yang posisinya di tengah dan di atas *button* “Foto” berfungsi sebagai *preview* gambar yang akan dilakukan prediksi oleh aplikasi. Kemudian, *button* “Foto” berfungsi untuk mengambil gambar mata yang akan diprediksi melalui kamera *smartphone*. *Button* “Galeri” berfungsi untuk mengambil gambar mata yang akan diprediksi melalui galeri *smartphone*. *Button* “Prediksi” berfungsi untuk melakukan prediksi gambar mata yang dikirim oleh pengguna. Desain untuk halaman prediksi gambar dapat dilihat pada Gambar 4.15.

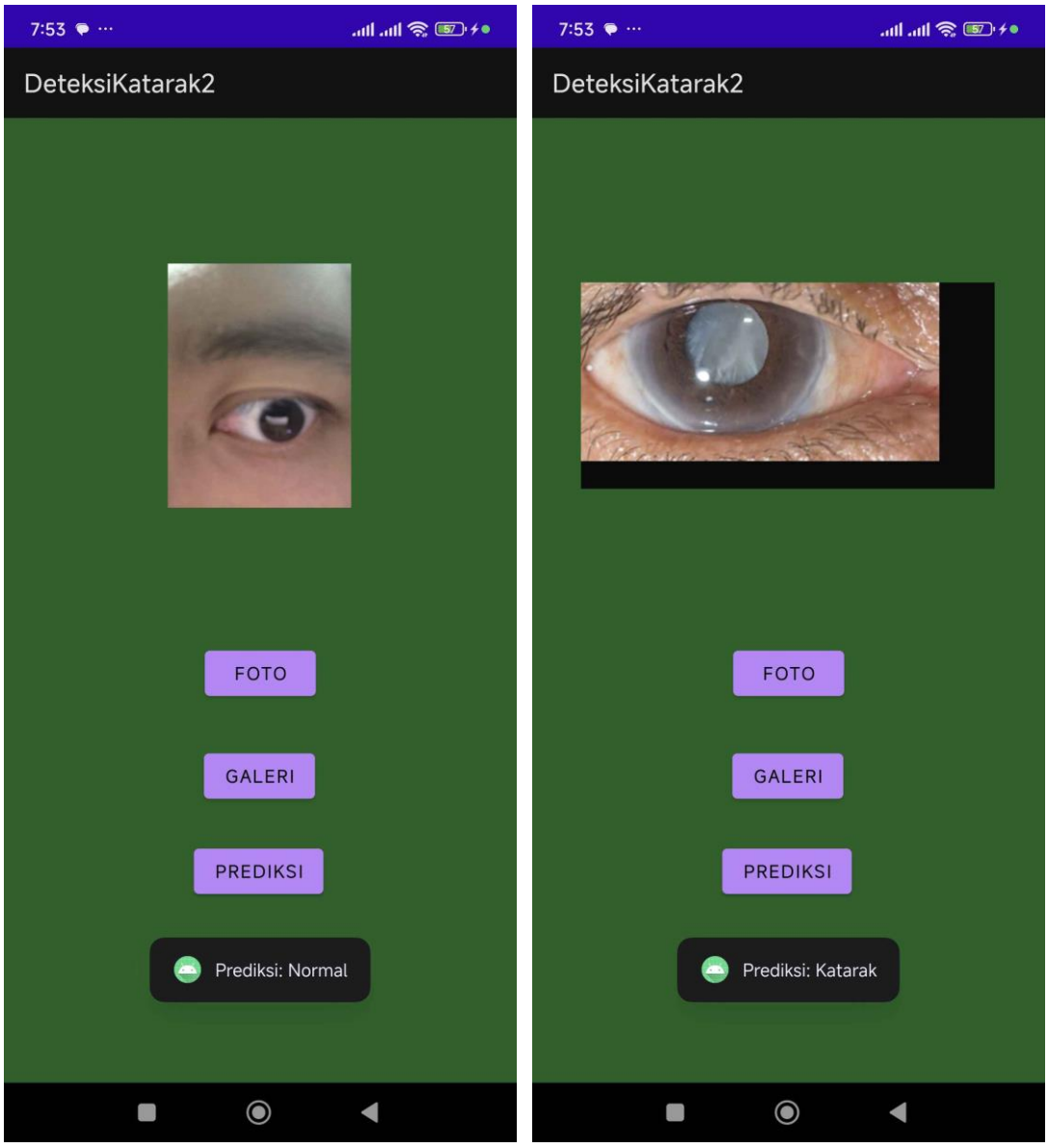
Gambar 4.15 Halaman Prediksi Gambar

Ketika gambar yang ingin dilakukan prediksi telah dimasukkan ke aplikasi oleh pengguna, aplikasi akan menampilkan *preview* gambar terlebih dahulu, untuk memastikan pengguna apakah gambar yang ingin dilakukan prediksi sesuai dengan keinginan pengguna. Gambar 4.16 menggambarkan aplikasi ketika menampilkan *preview* gambar mata yang akan dilakukan prediksi.



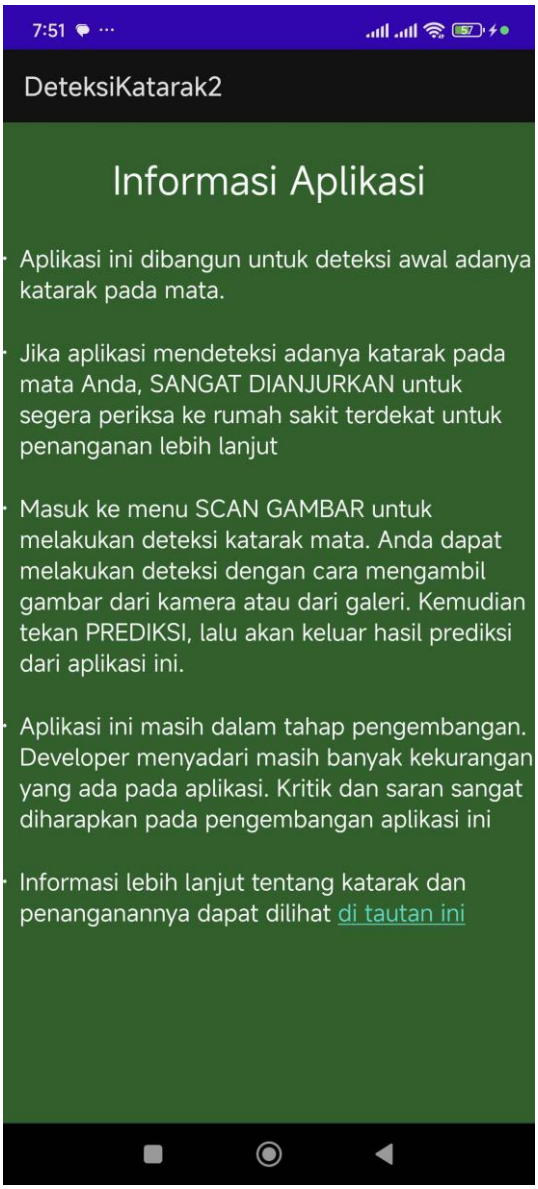
Gambar 4.16 Halaman Prediksi Gambar dengan Preview Gambar Mata

Setelah pengguna yakin dengan gambar yang akan dilakukan prediksi, pengguna menekan *button* "Prediksi". Kemudian, aplikasi akan melakukan klasifikasi gambar mata. Setelah selesai, aplikasi menampilkan pesan notifikasi kepada pengguna berupa hasil klasifikasi. Pesan tersebut berisi tentang gambar mata yang diprediksi merupakan gambar mata katarak atau normal. Gambar 4.17 menjelaskan jika aplikasi telah melakukan prediksi gambar mata.



Gambar 4.17 Hasil Prediksi Gambar Mata

Kemudian, aplikasi juga menyediakan halaman informasi yang berisi tentang informasi aplikasi dan himbauan apabila aplikasi mendeteksi adanya katarak pada pengguna aplikasi. Halaman informasi ini juga terdapat tautan *link* yang mengarahkan pengguna ke *website* Halodoc tentang informasi katarak, apabila pengguna menekan tautan *link* tersebut. Gambar 4.18 menunjukkan halaman informasi yang dimaksud.



Gambar 4.18 Halaman Informasi

Terakhir, aplikasi menyediakan fitur menu keluar aplikasi. Tujuan dari fitur ini adalah untuk memastikan kembali kepada pengguna jika benar-benar ingin keluar aplikasi atau masih tetap pada aplikasi. Gambar 4.19 menunjukkan menu keluar aplikasi yang terdapat pada aplikasi jika pengguna melakukan aktivitas keluar aplikasi.

Gambar 4.19 Menu Keluar Aplikasi

4.2.4 Hasil Pengembangan

Pada tahap pengembangan, hasil dari desain yang telah dipaparkan pada subbab 4.2.3 diimplementasikan di tahap ini menjadi sebuah aplikasi. Dengan beberapa *class*, aplikasi dibangun sehingga dapat digunakan sebagaimana mestinya.

4.2.4.1 Halaman Utama

Halaman utama adalah halaman yang ditampilkan pertama kali ketika pengguna membuka aplikasi. Halaman ini memberikan akses kepada pengguna untuk membuka halaman prediksi gambar dan halaman informasi. Halaman ini terdapat dua *button*, yaitu "Scan Gambar" dan "Informasi". Kedua *button* ini dapat mengarahkan pengguna ke halaman prediksi gambar atau halaman informasi, sesuai dengan kebutuhan pengguna. Implementasinya terdapat pada Kode 4.8.

```
<ConstraintLayout>
        <ImageView id='cataractimage' />
        <TextView id='nameapp' />
        <Button id='BtnCapture' />
        <Button id='BtnHelp' />
</ConstraintLayout>
```

Kode 4.8 Pseudocode XML activity_main.xml

Kode 4.8 digunakan untuk melakukan implementasi terhadap tampilan halaman utama. Seluruh komponen yang digunakan dalam halaman utama ditampilkan dalam satu *layout* menggunakan *ConstraintLayout*. Isi dari *layout* tersebut adalah *ImageView* untuk menampilkan gambar untuk *cover* halaman utama aplikasi, *TextView* untuk menampilkan judul aplikasi, dan dua buah *Button* untuk memindahkan pengguna ke halaman yang lain. *Button* yang bertuliskan "Scan Gambar" digunakan untuk memindahkan pengguna ke halaman prediksi gambar, sedangkan *Button* yang bertuliskan "Informasi" digunakan untuk memindahkan pengguna ke halaman informasi.

Setelah desain pada *file* XML dibuat, selanjutnya adalah membuat sebuah *activity* yang bernama MainActivity.kt. *Activity* ini digunakan supaya komponen di dalam *file* XML dapat dioperasikan. Di dalam MainActivity.kt terdapat dua buah variabel *button* yang berfungsi untuk menghubungkan *button* di *file* XML. Ada sebuah fungsi yang digunakan pada *activity* ini, yaitu fungsi *onCreate*(). Fungsi ini adalah yang pertama kali dipanggil ketika halaman utama dibuka. Kode 4.9 menjelaskan *activity* pada MainActivity.kt dibangun.

```
Kelas MainActivity extend AppCompatActivity() {
  fungsi onCreate(){
    super
      contentView := activity_main.xml

      variabel BtnCapture = view dengan id BtnCapture
      variabel BtnHelp = view dengan id BtnHelp

      BtnCapture ditambahkan OnClickListener {
        variabel intent1 = intent dari MainActivity.kt ke CaptureActivity.kt
        start
      }

      BtnHelp ditambahkan OnClickListener {
        variabel intent2 = intent dari MainActivity.kt ke HelpActivity.kt
        start
      }
  }
}
```

Kode 4.9 Pseudocode MainActivity.kt

Di dalam fungsi *onCreate*(), terdapat proses pembuatan *activity* dan implementasi seluruh variabel yang digunakan. Fungsi ini juga dapat menampilkan desain yang telah dibuat dalam *file* XML untuk tampilan halaman, termasuk untuk dua *button* yang akan digunakan pada halaman ini. Kedua *button* tersebut dihubungkan pada MainActivity dengan cara memberikan sebuah *onClickListener* kepada setiap *button*. Hal ini akan membuat *button* dapat menjalankan operasi untuk memindahkan aplikasi ke halaman yang lain. Ketika *BtnCapture* dijalankan atau ditekan oleh pengguna, aplikasi akan berpindah ke halaman prediksi gambar melalui *intent* yang menghubungkan antara MainActivity.kt dengan CaptureActivity.kt. Hal ini berlaku juga

pada *BtnHelp*, jika *button* tersebut dijalankan, aplikasi akan berpindah ke halaman informasi melalui *intent* yang menghubungkan antara MainActivity.kt dengan HelpActivity.kt.

4.2.4.2 Halaman Prediksi Gambar

Halaman prediksi gambar adalah halaman yang digunakan untuk melakukan prediksi gambar mata pengguna. Di dalam halaman ini, terjadi aktivitas berupa proses *upload* gambar mata ke aplikasi dan tentunya melakukan prediksi gambar mata yang telah di-*upload*. Pada halaman ini, terdapat tiga *button* dan sebuah *ImageView*. Beberapa *button* yang digunakan di halaman ini adalah sebagai berikut:

1. *Button* Foto : Berfungsi untuk membuka kamera *smartphone* pengguna.
2. *Button* Galeri : Berfungsi untuk membuka galeri *smartphone* pengguna.
3. *Button* Prediksi : Berfungsi untuk melakukan prediksi gambar mata yang telah di-*upload*

Untuk *ImageView* yang digunakan di halaman ini, bagian tersebut digunakan sebagai *preview* gambar mata yang telah di-*upload* oleh pengguna. Tujuannya adalah untuk memastikan gambar mata yang akan dilakukan prediksi adalah gambar yang sesuai dengan keinginannya, sekaligus memberikan informasi kepada aplikasi bahwa gambar itulah yang akan dilakukan prediksi. *ImageView* ini terletak di tengah halaman dan di atas *button* Foto, sedangkan untuk ketiga *button* yang berada di halaman ini tersusun secara vertikal ke bawah, berurutan mulai dari *button* Foto, Galeri, dan Prediksi. Implementasinya dapat dilihat pada Kode 4.10.

```
<ConstraintLayout>
  <ImageView id='captureView' />
  <Button id='photobutton' />
  <Button id='gallerybutton' />
  <Button id='predictbutton' />
</ConstraintLayout>
```

Kode 4.10 Pseudocode XML activity_capture.xml

Setelah desain pada *file* XML dibuat, selanjutnya adalah membuat *activity* pada CaptureActivity.kt. *Activity* ini digunakan supaya komponen di dalam *file* XML dapat dioperasikan. Di dalam CaptureActivity.kt terdapat tiga buah variabel *button* yang berfungsi untuk menghubungkan *button* di *file* XML dan beberapa variabel bebas seperti *captureImg*, *imageUri*, dan *imageBitmap*. Kemudian ada beberapa variabel tetap seperti *galleryLauncher* dan *cameraLauncher*. Ada beberapa fungsi yang digunakan pada *activity* ini, yaitu fungsi *onCreate*(), *createImageUri*(), dan *uriToBitmap*(). Implementasinya dapat dilihat pada Kode 4.11.

```
variabel captureImg := ImageView
variabel imageUrl := Uri
variabel imageBitmap := Bitmap dengan nilai null //sebagai tempat simpan

variabel galleryLauncher =
  fungsi registerActivityResult(ActivityResultContracts.GetContent()){
      callback parameter 'uri'
      jika uri != null {
        captureImg tampilkan gambar dari imageUrl
        convert imageUrl ke Bitmap dan simpan di imageBitmap
      } jika tidak = tampilkan tulisan "Tidak ada gambar yang dipilih"
    }

variabel cameraLauncher =
  fungsi registerForActivityResult(ActivityResultContracts.TakePicture()){
    callback fungsi 'isSuccess'
    jika isSuccess = TRUE {
      captureImg tampilkan gambar dari imageUrl
      convert imageUrl ke Bitmap dan simpan di imageBitmap
    }
    jika tidak {
      tampilkan tulisan "Gagal mengambil foto"
    }
  }

fungsi onCreate(){
  contentView := activity_capture
  imageUrl = fungsi createImageUri()
  captureImg = view dengan id captureView

  variabel photobutton = view dengan id photobutton
  photobutton ditambahkan onClickListener {
    jalankan cameraLauncher dengan input imageUrl
  }

  variabel gallerybutton = view dengan id gallerybutton
  gallerybutton ditambahkan onClickListener {
    jalankan galleryLauncher dengan input image/*
  }

  variabel predictbutton = view dengan id predictbutton
  predictbutton ditambahkan onClickListener {
    jalankan proses TFLite di android
  }
```

Kode 4.11 Pseudocode CaptureActivity.kt

Berdasarkan Kode 4.11, *activity* dimulai dari mendefinisikan variabel *galleryLauncher* untuk mengaktifkan fitur mengambil gambar dari galeri *smartphone* dan variabel *cameraLauncher* sebagai cara untuk mengaktifkan fitur mengambil gambar dari kamera *smartphone*. Kedua variabel ini mendaftarkan fungsi *registerForActivityResult* untuk menangani hasil aktivitas, dalam hal ini *GetContent*() untuk mengambil gambar melalui kamera dan *TakePicture*() untuk mengambil gambar dari galeri *smartphone*. Kedua variabel ini nanti dipanggil di dalam fungsi *onCreate* pada setiap *button* untuk dijalankan pada aplikasi.

Variabel galleryLauncher kemudian melakukan *callback* parameter *uri*. Parameter ini adalah hasil dari memilih konten (dalam hal ini hasil URI dari gambar galeri) yang dipilih. Jika nilai dari *uri* tidak sama dengan *null*, variabel *captureImg* menampilkan gambar dari *imageUrl* yang didefinisikan sebagai fungsi dari *createImageUri*(). Setelah itu, gambar yang berada di *imageUrl* dikonversikan ke *Bitmap*. Hasil konversi tadi disimpan di dalam variabel

imageBitmap. Jika nilai dari *uri* adalah *null*, aplikasi akan mengirimkan pesan "Tidak ada gambar yang dipilih".

Selanjutnya, masuk ke variabel *cameraLauncher*. Variabel ini melakukan *callback* fungsi *isSuccess*. Fungsi ini adalah sebuah *boolean* yang menunjukkan operasi mengambil gambar melalui kamera berhasil atau tidak. Jika *isSuccess* dinyatakan berhasil (*value*-nya adalah *TRUE*), variabel *captureImg* menampilkan gambar dari *imageUrl* yang didefinisikan sebagai fungsi dari *createImageUri*(). Setelah itu, gambar yang berada di *imageUrl* dikonversikan ke *Bitmap*. Hasil konversi tadi disimpan di dalam variabel *imageBitmap*. Jika *value* dari *isSuccess* adalah *FALSE*, aplikasi akan mengirimkan pesan "Gagal mengambil foto".

Langkah selanjutnya adalah masuk ke fungsi *onCreate*(). Di fungsi ini, tampilan dari seluruh desain yang telah dibangun pada *file* XML ditampilkan di fungsi ini. *Button-button* yang didesain sebelumnya, diimplementasikan kegunaannya di fungsi ini. Pada variabel *photobutton*, setelah ditambahkan *onClickListener*, aplikasi dapat membuka aplikasi kamera *smartphone*. Untuk variabel *galleryLauncher*, setelah ditambahkan *onClickListener*, aplikasi dapat membuka galeri *smartphone*. Sedangkan untuk variabel *predictbutton*, aplikasi dapat menjalankan proses prediksi yang akan dijelaskan pada subbab 4.2.5 tentang Hasil Integrasi Model Klasifikasi dengan Aplikasi.

Pada Kode 4.12, ada fungsi yang digunakan untuk memuat gambar dalam bentuk URI, yaitu *createImageUri*() dan cara mengonversi URI yang telah dipilih menjadi bentuk *bitmap*, yaitu *uriToBitmap*(). Hal ini dilakukan untuk membantu melakukan deteksi gambar pada aplikasi.

```
fungsi createImageUri(){
  variabel image = File(membuat direktori file camera_phones.png)
  buat URI untuk File dengan FileProvide.getUriForFile
  return URI yg telah dibuat
}

fungsi uriToBitmap(parameter -> uri, tipedata -> Uri){
  TRY {
    variabel inputStream = openInputStream dari parameter uri
    Decode inputStream ke Bitmap
    return
  }
  CATCH (exception stack trace){
    print stack trace
    return null
  }
}
```

Kode 4.12 Pseudocode Fungsi *createImageUri*() dan *uriToBitmap*()

4.2.4.3 Halaman Informasi

Untuk memuat halaman informasi, aplikasi harus membuat *activity* pada HelpActivity.kt. *Activity* ini digunakan supaya komponen di dalam *file* XML dapat dioperasikan. HelpActivity.kt terdapat beberapa informasi yang memuat tentang tujuan dibangunnya aplikasi dan himbauan apabila aplikasi mendeteksi katarak pada pengguna. Halaman ini juga mengandung *website* yang ditautkan pada aplikasi, yaitu pada bagian "di tautan ini". Tujuannya

adalah agar pengguna bisa langsung menuju ke *website* Halodoc tentang katarak untuk mengetahui katarak lebih lanjut. Kode 4.13 menjelaskan halaman informasi diimplementasikan.

```
<ConstraintLayout>
  <TextView id='title_info' />
  <TextView id='bullet1' />
  <TextView id='bullet2' />
  <TextView id='bullet3' />
  <TextView id='bullet4' />
  <TextView id='bullet5' />
</ConstraintLayout>
```

Kode 4.13 Pseudocode XML activity_help.xml

Berdasarkan Kode 4.13, halaman informasi tersusun dari beberapa *TextView* yang terdiri dari satu judul (*title_info*) dan beberapa *bullet* sebagai wadah untuk menampilkan informasi yang akan disampaikan pada aplikasi. Implementasinya ada di Kode 4.14.

```
Kelas HelpActivity extend AppCompatActivity(){
  fungsi onCreate(){
    contentView := activity_help.xml

    variabel bullet1: TextView = view dengan id bullet1
    variabel bulletText1 = string("Aplikasi ini....pada mata")
    edit bullet di bulletText1
    bullet1 tampilkan teks bulletText1

    variabel bullet2: TextView = view dengan id bullet2
    variabel bulletText2 = string("Jika Aplikasi....lebih lanjut")
    edit bullet di bulletText2
    bullet2 tampilkan teks bulletText2

    variabel bullet3: TextView = view dengan id bullet3
    variabel bulletText3 = string("Masuk ke menu....aplikasi ini")
    edit bullet di bulletText3
    bullet3 tampilkan teks bulletText3

    variabel bullet4: TextView = view dengan id bullet4
    variabel bulletText4 = string("Aplikasi ini....aplikasi ini")
    edit bullet di bulletText4
    bullet4 tampilkan teks bulletText4

    variabel bullet5: TextView = view dengan id bullet5
    variabel bulletText5 = string("Informasi lebih....tautan ini")
    edit bullet di bulletText5
    tambah URL pada kata "di tautan ini"
    bullet5 tampilkan teks bulletText5
  }
}
```

Kode 4.14 Pseudocode HelpActivity.kt

Kode 4.14 menjelaskan implementasi dari halaman informasi dijalankan pada aplikasi. Terdapat beberapa *string* berupa beberapa kalimat yang memuat informasi tentang aplikasi dan himbauan apabila aplikasi mendeteksi katarak pada pengguna. Setiap *bullet* mengandung satu

informasi pada aplikasi. Variabel *bullet5* menunjukkan cara URL *website* Halodoc dapat ditampilkan pada aplikasi, sekaligus diakses langsung oleh pengguna aplikasi.

4.2.4.4 Menu Keluar Aplikasi

Menu keluar aplikasi muncul ketika pengguna ingin keluar dari aplikasi. Ketika pengguna menekan tombol *back* di bawah layar *smartphone*, aplikasi akan memunculkan sebuah *pop up* menu yang berisi tentang pertanyaan "Anda yakin ingin keluar aplikasi?". Menu ini membutuhkan dua *button*, "Ya" dan "Tidak". *Button* "Ya" akan membuat aplikasi tertutup, artinya pengguna telah keluar dari aplikasi. Sedangkan *button* "Tidak" akan membuat pengguna akan diarahkan kembali ke tampilan utama aplikasi.

Menu keluar aplikasi diletakkan pada MainActivity.kt, karena pengguna biasanya memutuskan untuk keluar aplikasi jika sudah berada di halaman pertama aplikasi. Di samping itu, tombol *back* di layar *smartphone* digunakan untuk kembali ke *activity* sebelumnya, sehingga menu keluar aplikasi diletakkan di activity paling awal, yaitu MainActivity.kt. Kode 4.15 menunjukkan menu keluar aplikasi dapat digunakan pada aplikasi.

```
fungsi onBackPressedMethod():
  Alert Dialog Builder:
    judul: "Keluar"
    pesan: "Anda yakin ingin keluar aplikasi?"
    TIDAK cancelable

    positivebutton("Ya"):
      ketika diclick:
        pindahkan app task ke background
        killProcess(android)
        exitProcess(1)

    negativebutton("Tidak"):
      ketika diclick:
        jalankan null (tutup dialog)

  tampilkan dialog
```

Kode 4. 15 Pseudocode Fungsi *onBackPressedMethod*()

4.2.5 Hasil Integrasi Model Klasifikasi dengan Aplikasi

Bagian ini akan menjelaskan cara model klasifikasi dapat diintegrasikan pada aplikasi *android*. Hasil dari model klasifikasi dikonversikan ke dalam sebuah API dengan *framework* TFLite. *Framework* ini adalah salah satu API yang sering digunakan dalam dunia klasifikasi dalam aplikasi berbasis *android*. Dengan memanggil *library* yang diperlukan pada *python*, TFLite akan mengonversikan model klasifikasi yang diinginkan menjadi sebuah *file*. Kode 4.16 menunjukkan TFLite melakukan konversi pada model klasifikasi.

```
converter = tf.lite.TFLiteConverter.from_keras_model(densenet_model)
tflite_model = converter.convert()
with tf.io.gfile.GFile('eyes-cataract-recognition.tflite', 'wb') as f:
  f.write(tflite_model)
```

Kode 4.16 Konversi Model Klasifikasi ke TFLite

Karena hasil model klasifikasi yang terbaik adalah model klasifikasi DenseNet201, model tersebut kemudian digunakan untuk konversi ke TFLite. Hasil dari konversi tersebut adalah sebuah *file* yang bernama eyes-cataract-recognition.tflite. *File* ini akan dilakukan *import* ke aplikasi *android* untuk diintegrasikan dengan aplikasi, sehingga aplikasi bisa melakukan deteksi gambar. Kode 4.17 menjelaskan cara TFLite diintegrasikan pada aplikasi.

```
variabel imageProcessor = build ImageProcessor
  .add(ResizeOperation 300x300, ResizeMethod BILINEAR)
  .build()

predictbutton ditambahkan onClickListener {
  variabel tensorImage = inisiasi TensorImage dengan tipedata FLOAT32
  memuat imageBitmap ke tensorImage
  tensorImage = proses menggunakan imageProcessor

  variabel model = memuat EyesCataractRecognitionV2 //file TFLite

  variabel inputFeature0 = buat TensorBuffer dengan fixed size [1, 300, 300, 3],
                           tipe data FLOAT32
  memuat tensorImage.buffer ke inputFeature0

  variabel outputs = gunakan model untuk proses inputFeature0
  variabel outputFeature0 = output.ekstrak outputFeature0AsTensorBuffer, floatArray

  variabel maxIdx = cari index max value outputFeature0
  variabel result = when (maxIdx){
    jika 0, print "Katarak"
    jika 1, print "Normal"
    jika tidak, print "Tidak Terdeteksi"
  }

  Toast tampilkan hasil klasifikasi("Prediksi: $result")

  tutup model
}
```

Kode 4.17 Pseudocode Integrasi *File* TFLite pada Aplikasi

Integrasi antara *file* TFLite pada aplikasi diletakkan pada variabel *predictbutton*. Tujuannya adalah ketika tombol 'Prediksi' ditekan oleh pengguna, fungsi prediksi langsung bekerja. Prediksi dimulai dari membuat fungsi yang bernama *imageProcessor* terlebih dahulu. Hal ini dilakukan untuk menyeragamkan ukuran gambar yang dideteksi dengan model klasifikasi yang dimasukkan ke aplikasi. Karena model klasifikasi menggunakan *input* ukuran gambar 300x300, aplikasi juga harus disamakan *input* ukuran gambarnya menjadi 300x300 piksel dengan menggonakan metode *bilinear*.

Setelah mengatur *imageProcessor*, berikutnya adalah membuat fungsi prediksi pada *predictbutton*. Hal yang dilakukan pertama adalah membuat variabel *tensorImage* sebagai inisiasi dari *TensorImage* yang merupakan sebuah *library* dari *android* untuk mengimplementasi TFLite pada aplikasi berbasis *android*. Variabel ini bertipe data FLOAT32. Selanjutnya, *tensorImage* memuat *imageBitmap* sebagai gambar yang akan dilakukan prediksi dalam bentuk *Bitmap*. *tensorImage* yang telah diisi nilai *Bitmap* kemudian diproses menggunakan *imageProcessor*, sehingga ukuran piksel gambar seragam dengan model klasifikasi.

Setelah persiapan gambar yang akan dilakukan deteksi sudah diatur, langkah selanjutnya adalah memuat model klasifikasi. Sesuai dengan Kode 4.16, model klasifikasi dimuat dalam variabel *model* dengan nama model klasifikasi yang telah dikonversi adalah EyesCataractRecognitionV2. Kemudian, fungsi prediksi menyiapkan persiapan *input* dalam variabel *inputFeature0*. Persiapan ini dilakukan sebagai *data input* prediksi. Nilai dari variabel tersebut adalah sebuah *TensorBuffer* dengan ukuran *array* [1, 300, 300, 3] menggunakan tipe data FLOAT32. Lalu, variabel *inputFeature0* memuat variabel *tensorImage* yang telah diproses dengan diberikan *buffer* pada variabel.

Setelah memuat model klasifikasi dan persiapan data telah dilakukan, berikutnya adalah melakukan prediksi. Prediksi dilakukan pada variabel *outputs* yang memuat variabel *model* untuk melakukan proses terhadap variabel *inputFeature0*. Hasil dari prediksi kemudian dilakukan ekstraksi yang nilai prediksinya disimpan dalam variabel *outputFeature0* sebagai *float array*. Hasil ini kemudian ditampilkan berupa pesan kepada pengguna. Untuk mendapatkan hasil prediksi, variabel *maxIdx* digunakan untuk mencari indeks maksimal dari nilai *outputFeature0*. Ketika nilai *maxIdx* adalah 0, pesan yang disampaikan adalah "Prediksi: Katarak". Sebaliknya, jika nilai *maxIdx* adalah 1, pesan yang disampaikan adalah "Prediksi: Normal". Dari pesan yang disampaikan ini, pengguna dapat mengetahui kondisi mata sedang dalam keadaan normal atau terdapat katarak.

4.3 Hasil Uji Coba

Berikut ini adalah hasil uji coba dari aplikasi yang telah dibuat. Penjelasan hasil uji coba dengan berbagai skenario dibahas di sini.

4.3.1 Uji Coba Fitur Ambil Gambar menggunakan Kamera

Fitur ambil gambar menggunakan kamera adalah salah satu fitur yang paling penting dalam aplikasi. Fitur ini membutuhkan kemampuan aplikasi mendapatkan gambar mata pengguna untuk dilakukan prediksi keadaan mata pengguna aplikasi mengalami katarak atau normal. Skenario pengujian fitur ini ada pada Tabel 4.3.

No.	Skenario	Kondisi Berhasil
1.	Pengguna menekan tombol "Scan Gambar"	Halaman aplikasi berpindah ke halaman prediksi gambar
2.	Pengguna menekan tombol "Foto"	Halaman aplikasi berpindah ke kamera *smartphone* pengguna
3.	Pengguna selesai mengambil gambar dari kamera *smartphone*	Halaman prediksi gambar menampilkan *preview* tangkapan gambar dari kamera

Tabel 4.3 Skenario Uji Coba Fitur Ambil Gambar menggunakan Kamera

Setelah dilakukan uji coba sesuai dengan skenario pada Tabel 4.3, hasil dari uji coba ditampilkan pada Tabel 4.4.

No.	Skenario	Kondisi Berhasil	Hasil
1.	Pengguna menekan tombol "Scan Gambar"	Halaman aplikasi berpindah ke halaman prediksi gambar	Berhasil
2.	Pengguna menekan tombol "Foto"	Halaman aplikasi berpindah ke kamera *smartphone* pengguna	Berhasil
3.	Pengguna selesai mengambil gambar dari kamera *smartphone*	Halaman prediksi gambar menampilkan *preview* tangkapan gambar dari kamera	Berhasil

Tabel 4.4 Hasil Uji Coba Fitur Ambil Gambar menggunakan Kamera

4.3.2 Uji Coba Fitur Ambil Gambar dari Galeri Smartphone

Fitur ambil gambar dari galeri *smartphone* adalah salah satu fitur yang paling penting dalam aplikasi. Fitur ini membutuhkan kemampuan aplikasi mengambil gambar mata pengguna yang tersimpan di dalam galeri *smartphone* pengguna untuk dilakukan prediksi keadaan mata pengguna aplikasi mengalami katarak atau normal. Skenario pengujian fitur ini ada pada Tabel 4.5.

No.	Skenario	Kondisi Berhasil
1.	Pengguna menekan tombol "Scan Gambar"	Halaman aplikasi berpindah ke halaman prediksi gambar
2.	Pengguna menekan tombol "Galeri"	Halaman aplikasi berpindah ke galeri *smartphone* pengguna
3.	Pengguna selesai mengambil gambar dari galeri *smartphone*	Halaman prediksi gambar menampilkan *preview* tangkapan gambar dari kamera

Tabel 4.5 Skenario Uji Coba Fitur Ambil Gambar dari Galeri *Smartphone*

Setelah dilakukan uji coba sesuai dengan skenario pada Tabel 4.5, hasil dari uji coba ditampilkan pada Tabel 4.6.

No.	Skenario	Kondisi Berhasil	Hasil
1.	Pengguna menekan tombol "Scan Gambar"	Halaman aplikasi berpindah ke halaman deteksi gambar	Berhasil
2.	Pengguna menekan tombol "Galeri"	Halaman aplikasi berpindah ke galeri *smartphone* pengguna	Berhasil
3.	Pengguna selesai mengambil gambar dari galeri *smartphone*	Halaman prediksi gambar menampilkan *preview* pilihan gambar dari galeri	Berhasil

Tabel 4.6 Hasil Uji Coba Fitur Ambil Gambar dari Galeri Smartphone

4.3.3 Uji Coba Fitur Prediksi Gambar

Fitur prediksi gambar adalah salah satu fitur yang paling penting dalam aplikasi. Fitur ini membutuhkan kemampuan aplikasi melakukan prediksi gambar sesuai dengan gambar yang telah di-*upload* oleh pengguna. Skenario pengujian fitur ini ada pada Tabel 4.7.

No.	Skenario	Kondisi Berhasil
1.	Pada halaman prediksi gambar yang telah terdapat *preview* gambar yang akan diprediksi, pengguna menekan tombol "Prediksi".	Aplikasi menampilkan pesan "Prediksi: Normal" jika gambar mata yang diprediksi normal.
2.	Pada halaman prediksi gambar yang telah terdapat *preview* gambar yang akan diprediksi, pengguna menekan tombol "Prediksi".	Aplikasi menampilkan pesan "Prediksi: Katarak" jika gambar mata yang diprediksi katarak.

Tabel 4.7 Skenario Uji Coba Fitur Prediksi Gambar

Setelah dilakukan uji coba sesuai dengan skenario pada Tabel 4.7, hasil dari uji coba ditampilkan pada Tabel 4.8.

No.	Skenario	Kondisi Berhasil	Hasil
1.	Pada halaman prediksi gambar yang telah terdapat *preview* gambar yang akan diprediksi, pengguna menekan tombol "Prediksi".	Aplikasi menampilkan pesan "Prediksi: Normal" jika gambar mata yang diprediksi normal.	Berhasil
2.	Pada halaman prediksi gambar yang telah terdapat *preview* gambar yang akan diprediksi, pengguna menekan tombol "Prediksi".	Aplikasi menampilkan pesan "Prediksi: Katarak" jika gambar mata yang diprediksi katarak.	Berhasil

Tabel 4.8 Hasil Uji Coba Fitur Prediksi Gambar

4.3.4 Uji Coba Fitur Informasi

Fitur informasi adalah fitur tambahan dalam aplikasi. Fitur ini membutuhkan kemampuan aplikasi dalam menyediakan informasi tentang aplikasi deteksi katarak mata. Isi informasi ini adalah cara penanganan yang seharusnya dilakukan jika aplikasi mendeteksi adanya katarak, permintaan saran dan kritik tentang aplikasi, dan informasi lebih lanjut tentang katarak yang dapat diakses pengguna melalui sebuah *website* yang sudah ditautkan pada informasi di aplikasi. Skenario pengujian fitur ini dijelaskan pada Tabel 4.9.

No.	Skenario	Kondisi Berhasil
1.	Pengguna menekan tombol "Informasi".	Halaman aplikasi berpindah ke halaman informasi.
2.	Pengguna menekan tautan *link website.*	Pengguna dialihkan ke *website* Halodoc yang berisi tentang penjelasan katarak.

Tabel 4.9 Skenario Uji Coba Fitur Informasi Aplikasi

Setelah dilakukan uji coba sesuai dengan skenario pada Tabel 4.9, hasil dari uji coba ditampilkan pada Tabel 4.10.

No.	Skenario	Kondisi Berhasil	Hasil
1.	Pengguna menekan tombol "Informasi".	Halaman aplikasi berpindah ke halaman informasi.	Berhasil
2.	Pengguna menekan tautan *link website.*	Pengguna dialihkan ke *website* Halodoc yang berisi tentang penjelasan katarak.	Berhasil

Tabel 4.10 Hasil Uji Coba Fitur Informasi Aplikasi

4.3.5 Uji Coba Fitur Menu Keluar Aplikasi

Fitur menu keluar aplikasi membutuhkan kemampuan untuk mengarahkan pengguna tetap berada di aplikasi atau keluar dari aplikasi. Jika pengguna memilih "Ya", aplikasi akan tertutup. Sedangkan, jika pengguna memilih "Tidak", pengguna akan tetap berada di aplikasi. Skenario pengujian fitur ini dijelaskan pada Tabel 4.11.

No.	Skenario	Kondisi Berhasil
1.	Pengguna menekan tombol *back* pada *smartphone* ketika di halaman utama.	Kotak dialog muncul pada layar *smartphone* pengguna, berisi pertanyaan "Anda yakin ingin keluar aplikasi?" dan dua jawaban "Ya" dan "Tidak".
2.	Pengguna memilih "Ya" pada kotak dialog yang muncul.	Pengguna keluar dari aplikasi.
3.	Pengguna memilih "Tidak" pada kotak dialog yang muncul.	Pengguna tetap di aplikasi.

Tabel 4.11 Skenario Uji Coba Fitur Menu Keluar Aplikasi

Setelah dilakukan uji coba sesuai dengan skenario pada Tabel 4.11, hasil dari uji coba ditampilkan pada Tabel 4.12.

No.	Skenario	Kondisi Berhasil	Hasil
1.	Pengguna menekan tombol *back* pada *smartphone* ketika di halaman utama.	Kotak dialog muncul pada layar *smartphone* pengguna, berisi pertanyaan "Anda yakin ingin keluar aplikasi?", diikuti dua jawaban "Ya" dan "Tidak".	Berhasil
2.	Pengguna memilih "Ya" pada kotak dialog yang muncul.	Pengguna keluar dari aplikasi.	Berhasil
3.	Pengguna memilih "Tidak" pada kotak dialog yang muncul.	Pengguna tetap di aplikasi.	Berhasil

Tabel 4.12 Hasil Uji Coba Fitur Menu Keluar Aplikasi

4.3.6 Uji Coba Aplikasi menggunakan Dataset

Selanjutnya adalah menguji coba aplikasi apabila mengambil foto menggunakan *dataset* yang ada. Selain menggunakan *dataset* milik Hemoo Redaoo dan Nandan Padia dkk., uji coba ini menggunakan beberapa gambar yang diambil dari Google Image. Tujuan dari uji coba ini adalah untuk mengetahui seberapa akurat aplikasi dalam mendeteksi katarak pada mata. Gambar 4.20 menunjukkan beberapa gambar yang diuji pada aplikasi.

Gambar 4.20 Sampel Gambar dari *Dataset* untuk Pengujian pada Aplikasi (urutan gambar dimulai dari kiri ke kanan dan dari atas ke bawah)

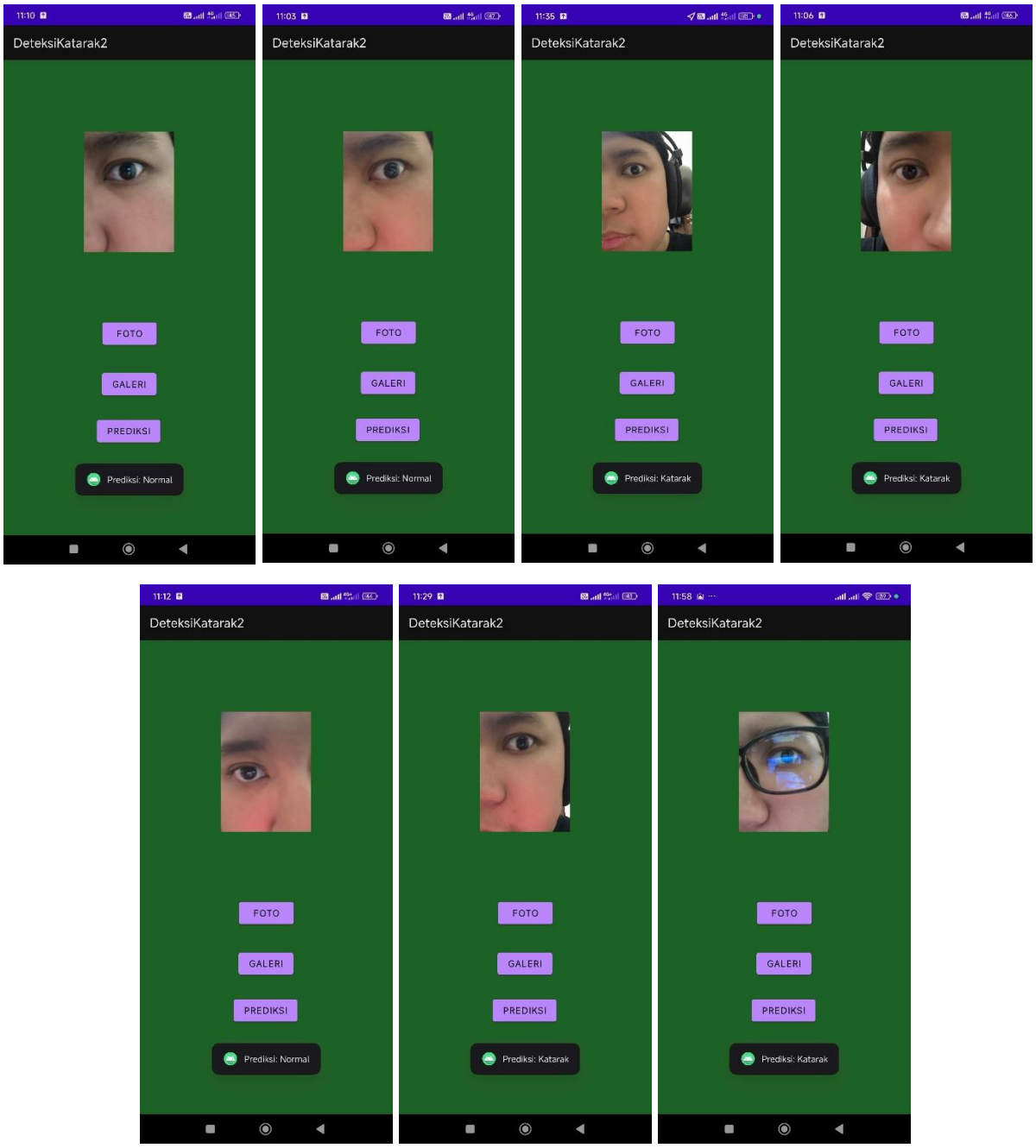
Setelah melakukan pengujian gambar terhadap aplikasi, hasilnya ada pada Tabel 4.13. Tabel tersebut menunjukkan bahwa aplikasi dapat mengeluarkan hasil sesuai yang diinginkan.

No.	Gambar	Hasil Prediksi
1.	Gambar 1 (baris 1 kolom 1)	Katarak
2.	Gambar 2 (baris 1 kolom 2)	Katarak
3.	Gambar 3 (baris 1 kolom 3)	Katarak
4.	Gambar 4 (baris 1 kolom 4)	Normal
5.	Gambar 5 (baris 2 kolom 1)	Normal
6.	Gambar 6 (baris 2 kolom 2)	Katarak
7.	Gambar 7 (baris 2 kolom 3)	Katarak
8.	Gambar 8 (baris 2 kolom 4)	Normal
9.	Gambar 9 (baris 3 kolom 1)	Normal
10.	Gambar 10 (baris 3 kolom 2)	Normal

Tabel 4.13 Hasil Uji Coba menggunakan Gambar dari Dataset dan Google Image

4.3.7 Uji Coba Aplikasi menggunakan Mata Peneliti

Selanjutnya adalah menguji coba aplikasi apabila mengambil foto menggunakan mata peneliti. Tujuan dari pengujian ini adalah untuk memastikan aplikasi dapat mendeteksi mata pengguna dengan baik dan benar. Hasil dari uji coba ini ditampilkan pada Gambar 4.21.



Gambar 4.21 Hasil Uji Coba Pengujian pada Aplikasi menggunakan Mata Peneliti (urutan gambar dimulai dari kiri ke kanan dan dari atas ke bawah)

Berdasarkan hasil uji coba menggunakan mata peneliti, hasil yang didapatkan adalah data mata peneliti yang dimasukkan seperti pada Gambar 4.21 yang pertama, kedua, dan kelima mendapatkan hasil prediksi normal. Kemudian, untuk data mata peneliti yang dimasukkan seperti pada Gambar 4.21 yang ketiga, keempat, keenam dan ketujuh mendapatkan hasil prediksi katarak.

4.4 Pembahasan

Pada uji coba yang telah dilakukan pada tugas akhir ini, penelitian menggunakan *dataset* katarak mata normal dan katarak, hasil kontribusi dari Hemoo Redaoo dan Nanda Padiaa dkk yang dapat ditemukan di *website* Kaggle, dengan total gambar sejumlah 1099. Kemudian,

dataset dilakukan pengolahan data dengan mengategorikan gambar ke dalam 2 macam, yaitu gambar mata katarak yang berjumlah 547 gambar dan gambar mata normal yang berjumlah 552 gambar. Langkah selanjutnya dengan *dataset* adalah melakukan *preparation data* untuk menyeragamkan ukuran gambar data dan *pre-processing data* agar mendapatkan *data training* dan *data validation* yang digunakan untuk proses klasifikasi pada model. Angka yang digunakan *dataset* untuk melakukan *preparation data* dan *pre-processing data* adalah IMAGE_SIZE sebesar 300x300 piksel, 64 *batch* dan 999 *seed*. Sedangkan *data training* yang digunakan sejumlah 880 gambar dan *data validation* yang digunakan sejumlah 219 gambar. Terakhir, *dataset* mengalami proses *Image Augmentation* untuk menambah jumlah data yang dilatih dalam melakukan model klasifikasi.

Kemudian, setelah melakukan pengolahan terhadap *dataset*, tugas akhir ini melakukan klasifikasi dengan menggunakan model yang telah disebutkan pada subbab 4.1.3, yaitu *Convolutional Neural Network* (CNN), VGG16, ResNet50, dan DenseNet 201. Tujuannya adalah untuk menentukan model klasifikasi yang memiliki tingkat akurasi yang terbaik. Ada parameter yang digunakan untuk melakukan klasifikasi, yaitu *loss function*, *optimizer*, dan *metric*. Dengan *epoch* sejumlah 30, keempat model klasifikasi dilakukan pelatihan. Hasil dari pelatihan model klasifikasi akan menunjukkan sejumlah variabel tentang *accuracy, loss, val_accuracy,* dan *val_loss*. Hasilnya adalah bahwa CNN memiliki *val_accuracy* sebesar 82,19% dan *val_loss* sebesar 44,48%, VGG16 memiliki *val_accuracy* sebesar 93,15% dan *val_loss* sebesar 21,91%, ResNet50 memiliki *val_accuracy* sebesar 59,36% dan *val_loss* sebesar 78,40%, dan DenseNet201 memiliki *val_accuracy* sebesar 93,61% dan *val_loss* sebesar 18,99%. Karena DenseNet201 memiliki *val_accuracy* yang paling besar dan *val_loss* yang paling kecil, model klasifikasi DenseNet201 akhirnya digunakan untuk aplikasi deteksi katarak mata.

Setelah menemukan model klasifikasi yang digunakan pada aplikasi, selanjutnya adalah proses pembuatan aplikasi. Dapat diketahui bahwa aplikasi berbasis *android* ini dibangun sebagai antarmuka bagi para pengguna dalam melakukan deteksi pada katarak mata. Berdasarkan fitur-fitur yang diberikan untuk uji coba, aplikasi harus dapat mengambil gambar melalui kamera maupun galeri *smartphone*. Hal ini dibutuhkan karena aplikasi membutuhkan gambar dari pengguna untuk melakukan prediksi terhadap model klasifikasi yang akan dikirimkan melalui API dengan *framework* TFLite. Kemudian, hasil dari model klasifikasi akan dikirim kembali melalui API kepada aplikasi untuk ditampilkan kepada pengguna tentang kondisi gambar mata yang telah dilakukan prediksi. Aplikasi juga harus menampilkan informasi tentang tujuan dibangunnya aplikasi dan himbauan apabila aplikasi mendeteksi adanya katarak pada mata pengguna.

Setelah melakukan pembuatan aplikasi, selanjutnya adalah melakukan pengujian fitur-fitur yang terdapat pada aplikasi. Hal yang diuji pertama adalah mengambil gambar menggunakan kamera dan galeri *smartphone*. Skenario yang dijalankan adalah pengguna menekan tombol "Scan Gambar" untuk masuk ke halaman prediksi gambar, lalu menekan tombol "Foto" atau "Galeri" untuk mengambil gambar baik dari kamera maupun galeri *smartphone*. Ketika berhasil mengambil gambar, aplikasi akan menampilkan *preview* gambar yang telah dipilih atau diambil untuk dilakukan prediksi gambar. Setelah melalui proses uji coba, aplikasi berhasil melakukan skenario tersebut.

Hal yang diuji selanjutnya adalah melakukan prediksi gambar pada aplikasi. Skenario yang dijalankan adalah pengguna menekan tombol "Prediksi" setelah aplikasi menampilkan *preview* gambar yang akan dilakukan prediksi. Kemudian, aplikasi akan melakukan prediksi gambar terhadap model klasifikasi yang terdapat pada aplikasi. Ketika selesai melakukan prediksi, aplikasi akan menampilkan pesan bahwa gambar mata yang diprediksi adalah Normal atau Katarak. Setelah melalui proses uji coba, aplikasi berhasil melakukan skenario tersebut.

Hal yang diuji selanjutnya adalah membuka halaman informasi. Skenarionya adalah pengguna menekan tombol "Informasi". Setelah itu, aplikasi akan berpindah ke halaman informasi yang berisi tentang himbauan dan tujuan aplikasi dibuat. Halaman informasi juga menyediakan tautan *link website* ke Halodoc yang berisi tentang penjelasan katarak. Setelah dilakukan uji coba, aplikasi berhasil melakukan skenario tersebut.

Sebagai penutup uji coba, hal yang diuji terakhir adalah menu keluar aplikasi. Skenarionya adalah pengguna menekan tombol *back* pada layar *smartphone* ketika aplikasi berada di halaman utama. Lalu, aplikasi akan memunculkan kotak dialog berisi pertanyaan Anda yakin ingin keluar aplikasi?" dengan diikuti dua jawaban "Ya" dan "Tidak". Jika pengguna memilih jawaban "Ya", aplikasi akan tertutup, artinya pengguna telah keluar dari aplikasi. Sebaliknya, jika pengguna memilih jawaban "Tidak", pengguna akan tetap berada di aplikasi, yang artinya aplikasi masih berjalan.

Kemudian, aplikasi melakukan pengujian deteksi gambar katarak mata menggunakan *dataset* atau Google Image dan mata peneliti. Setelah dilakukan uji coba, ketika aplikasi mendeteksi gambar katarak mata yang bersumber dari *dataset* atau Google Image, aplikasi dapat mendeteksi gambar dengan hasil yang tepat dan sesuai yang diinginkan. Namun, ketika aplikasi mendeteksi gambar mata melalui mata sendiri, aplikasi terkadang tidak dapat mendeteksi gambar dengan hasil yang diinginkan. Alasan yang dapat mengakibatkan hasil yang tidak diinginkan ini adalah sebagai berikut:

1. Posisi kamera kurang dekat dengan mata, sehingga aplikasi sulit mendeteksi mata dengan baik. Hal ini dapat dilihat pada Gambar 4.21 yang ketiga, keempat, dan keenam. Karena objek yang diambil bukan hanya mata, menyebabkan aplikasi kesulitan melakukan prediksi gambar.
2. Adanya pantulan cahaya kadang dapat mengganggu fokus aplikasi terhadap mata. Hal ini dapat dilihat pada Gambar 4.21 yang ketujuh. Terlalu banyak cahaya yang terpantul di mata dapat dianggap sebagai titik katarak oleh aplikasi, sehingga mempengaruhi hasil prediksi gambar.

Setelah mencoba beberapa skenario yang telah direncanakan, telah terbukti bahwa aplikasi dapat dianggap lolos uji coba setelah aplikasi berhasil melaksanakan beberapa skenario yang dibutuhkan. Artinya, aplikasi telah memenuhi *use case* dari kebutuhan pengguna dan siap dikembangkan di masa depan.

BAB 5 Kesimpulan dan Saran

5.1 Kesimpulan

Berdasarkan pada penelitian yang telah dilakukan, tugas akhir ini dapat menarik beberapa kesimpulan. Poin-poin kesimpulan dijabarkan seperti berikut:

1. Dengan jumlah *epoch* sebanyak 30, keempat model klasifikasi dilakukan pelatihan data. Hasilnya adalah model klasifikasi DenseNet201 memiliki *val_accuracy* paling tinggi di angka 93.61% dan *val_loss* paling rendah di angka 18,99%. Selain itu, grafik akurasi dari DenseNet201 juga memiliki hasil yang terbaik dan konsisten. Oleh karena itu, model klasifikasi DenseNet201 digunakan pada tugas akhir ini.
2. Aplikasi ini berhasil memasukkan gambar mata pengguna ke dalam aplikasi, baik melalui kamera atau galeri *smartphone*. Kemudian, aplikasi juga dapat mengeluarkan hasil prediksi berupa pesan "Prediksi: Katarak" atau "Prediksi: Normal".
3. Proses klasifikasi gambar yang dijalankan oleh aplikasi berhasil dijalankan dengan melibatkan *framework* TFLite sebagai API yang menjembatani antara *client* (diperankan oleh aplikasi) dan *server* (diperankan oleh model klasifikasi) untuk melakukan aktivitas *request* (mengirim gambar ke model klasifikasi untuk dimintakan hasil klasifikasi pada gambar katarak) dan *receive* (menerima hasil klasifikasi pada gambar katarak).
4. Aplikasi deteksi katarak mata dapat menyampaikan informasi tentang aplikasi dan himbauan apabila aplikasi mendeteksi adanya katarak pada pengguna.
5. Aplikasi dapat mendeteksi gambar mata yang bersumber dari *dataset* atau Google Image dengan hasil yang sesuai diinginkan.
6. Aplikasi dapat mendeteksi gambar mata yang bersumber dari mata peneliti. Namun, jika ada gangguan seperti posisi kamera yang kurang dekat ke mata atau ada banyak pantulan cahaya pada mata, hasil prediksi gambar aplikasi tidak sesuai dengan yang diinginkan.

5.2 Saran

Saran dari pembuatan tugas akhir Perancangan dan Pembuatan Aplikasi Android Deteksi Katarak Mata melalui Ponsel Kamera menggunakan Metode Transfer Learning adalah untuk meningkatkan performa model klasifikasi yang digunakan pada aplikasi deteksi katarak. Selain itu, tampilan antarmuka pengguna dapat dilakukan improvisasi sehingga dapat memiliki fitur yang lebih berguna dan menarik.

DAFTAR PUSTAKA

[1] World Health Organization, “Blindness and vision impairment.” Accessed: Dec. 01, 2023. [Online]. Available: https://www.who.int/en/news-room/fact-sheets/detail/blindness-and-visual-impairment

[2] Sehat Negeriku Kemkes, “Katarak Penyebab Terbanyak Gangguan Penglihatan di Indonesia.” Accessed: Nov. 25, 2024. [Online]. Available: https://sehatnegeriku.kemkes.go.id/baca/umum/20211012/5738714/katarak-penyebab-terbanyak-gangguan-penglihatan-di-indonesia/

[3] S. Hu *et al.*, “Unified diagnosis framework for automated nuclear cataract grading based on smartphone slit-lamp images,” *IEEE Access*, vol. 8, pp. 174169–174178, 2020, doi: 10.1109/ACCESS.2020.3025346.

[4] S. Pathak and B. Kumar, “A robust automated cataract detection algorithm using diagnostic opinion based parameter thresholding for telemedicine application,” *Electronics (Switzerland)*, vol. 5, no. 3, Sep. 2016, doi: 10.3390/electronics5030057.

[5] L. T. Chylack Jr *et al.*, “The Lens Opacities Classification System III,” *Archives of Ophthalmology*, vol. 111, no. 6, pp. 831–836, Jun. 1993, doi: 10.1001/archopht.1993.01090060119035.

[6] C. J. Lai, P. F. Pai, M. Marvin, H. H. Hung, S. H. Wang, and D. N. Chen, “The Use of Convolutional Neural Networks and Digital Camera Images in Cataract Detection,” *Electronics (Switzerland)*, vol. 11, no. 6, Mar. 2022, doi: 10.3390/electronics11060887.

[7] M. R. Hossain, S. Afroze, N. Siddique, and M. M. Hoque, “Automatic Detection of Eye Cataract using Deep Convolution Neural Networks (DCNNs),” in *2020 IEEE Region 10 Symposium (TENSYMP)*, 2020, pp. 1333–1338. doi: 10.1109/TENSYMP50017.2020.9231045.

[8] Badan Pusat Statistik, “Proporsi Individu yang Menguasai/Memiliki Telepon Genggam Menurut Provinsi (Persen), 2021-2023.” Accessed: Nov. 25, 2024. [Online]. Available: https://www.bps.go.id/id/statistics-table/2/MTIyMSMy/proporsi-individu-yang-menguasai-memiliki-telepon-genggam-menurut-provinsi.html

[9] Hemoo Redaoo, “cataract.” Accessed: Sep. 05, 2024. [Online]. Available: https://www.kaggle.com/datasets/hemooredaoo/cataract

[10] Nandan Padia, Siddharth Patel, Amit Hirpara, and Dhaivat Jani, “Cataract dataset.” Accessed: Sep. 05, 2024. [Online]. Available: https://www.kaggle.com/datasets/nandanp6/cataract-image-dataset

[11] M. Yusuf, S. Theophilous, J. Adejoke, and A. B. Hassan, “Web-Based Cataract Detection System Using Deep Convolutional Neural Network,” in *2019 2nd International Conference of the IEEE Nigeria Computer Chapter (NigeriaComputConf)*, 2019, pp. 1–7. doi: 10.1109/NigeriaComputConf45974.2019.8949636.

[12] Hadeer R. M. Tawfik, Rania A. K. Birry, and Amani A. Saad, “Early Recognition and Grading of Cataract Using a Combined Log Gabor/Discrete Wavelet Transform with

ANN and SVM," *World Academy of Science, Engineering and Technology International Journal of Computer and Information Engineering*, vol. 12, pp. 1–6, 2018.

[13] X. Luo, J. Li, M. Chen, X. Yang, and X. Li, "Ophthalmic Disease Detection via Deep Learning With a Novel Mixture Loss Function," *IEEE J Biomed Health Inform*, vol. 25, no. 9, pp. 3332–3339, 2021, doi: 10.1109/JBHI.2021.3083605.

[14] D. Patil, A. Nair, N. Bhat, R. Chavan, and D. Jadhav, "Analysis and study of cataract detection techniques," in *2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)*, 2016, pp. 516–519. doi: 10.1109/ICGTSPICC.2016.7955355.

[15] GeeksforGeeks, "An introduction to Machine Learning." Accessed: Dec. 10, 2023. [Online]. Available: https://www.geeksforgeeks.org/introduction-machine-learning/

[16] M. Arif, W. Farooq, A. B. Saduf, A. Asif, and I. Khan, "Studies in Big Data 57 Advances in Deep Learning," 2020. [Online]. Available: http://www.springer.com/series/11970

[17] X.-Q. Zhang *et al.*, "Machine Learning for Cataract Classification/Grading on Ophthalmic Imaging Modalities: A Survey," *Machine Intelligence Research*, vol. 19, no. 3, pp. 184–208, 2022, doi: 10.1007/s11633.

[18] M. Hussain, J. J. Bird, and D. R. Faria, "A study on CNN transfer learning for image classification," in *Advances in Intelligent Systems and Computing*, Springer Verlag, 2019, pp. 191–202. doi: 10.1007/978-3-319-97982-3_16.

[19] M. S. M. Khan, M. Ahmed, R. Z. Rasel, and M. M. Khan, "Cataract Detection Using Convolutional Neural Network with VGG-19 Model," in *2021 IEEE World AI IoT Congress (AIIoT)*, 2021, pp. 209–212. doi: 10.1109/AIIoT52608.2021.9454244.

[20] D. Albashish, R. Al-Sayyed, A. Abdullah, M. H. Ryalat, and N. Ahmad Almansour, "Deep CNN Model based on VGG16 for Breast Cancer Classification," in *2021 International Conference on Information Technology, ICIT 2021 - Proceedings*, Institute of Electrical and Electronics Engineers Inc., Jul. 2021, pp. 805–810. doi: 10.1109/ICIT52682.2021.9491631.

[21] Z. Han, B. Wei, Y. Zheng, Y. Yin, K. Li, and S. Li, "Breast Cancer Multi-classification from Histopathological Images with Structured Deep Learning Model," *Sci Rep*, vol. 7, no. 1, Dec. 2017, doi: 10.1038/s41598-017-04075-z.

[22] S. S. Yadav and S. M. Jadhav, "Deep convolutional neural network based medical image classification for disease diagnosis," *J Big Data*, vol. 6, no. 1, Dec. 2019, doi: 10.1186/s40537-019-0276-2.

[23] L. I. Kesuma and R. Rudiansyah, "Classification of Covid-19 Diseases Through Lung CT-Scan Image Using the ResNet-50 Architecture," *Computer Engineering and Applications*, vol. 12, no. 1, 2023, [Online]. Available: https://www.kaggle.com/maedemaftouni/large-

[24] T. Lu, B. Han, L. Chen, F. Yu, and C. Xue, "A generic intelligent tomato classification system for practical applications using DenseNet-201 with transfer learning," *Sci Rep*, vol. 11, no. 1, Dec. 2021, doi: 10.1038/s41598-021-95218-w.

[25] Guido van Rossum, “Python Tutorial Release 3.8.1 Guido van Rossum and the Python development team,” 2020.

[26] M. Ramchandani *et al.*, “Survey: Tensorflow in Machine Learning,” in *Journal of Physics: Conference Series*, Institute of Physics, 2022. doi: 10.1088/1742-6596/2273/1/012008.

[27] Benjamin Speckmann, “The Android Mobile Platform,” *A Review Paper Submitted to the Eastern Michigan University Department of Computer Science In Partial Fulfillment of the Requirements for the Master of Science in Computer Science*, 2008.

[28] S. Saeed, N. Z. Jhanjhi, M. Naqvi, and M. Humayun, “Analysis of software development methodologies,” *International Journal of Computing and Digital Systems*, vol. 8, no. 5, pp. 445–460, 2019, doi: 10.12785/ijcds/080502.

BIODATA PENULIS

Penulis dilahirkan di Gresik, 1 Mei 2000, merupakan anak pertama dari 3 bersaudara. Penulis telah menempuh pendidikan formal yaitu di TK Islam Bakti 3 Randuagung, SDN Petrokimia Gresik (sekarang UPT SD Negeri 4 Gresik), SMPN 1 Gresik dan SMAN 1 Gresik. Setelah lulus dari SMAN tahun 2018, Penulis mengikuti Program Kemitraan dan Mandiri dan diterima di Departemen Teknik Informatika FTIK pada tahun 2018 dan terdaftar dengan NRP 05111840000137.

Di Departemen Teknik Informatika Penulis sempat aktif di beberapa kegiatan seperti pelatihan LKMM Pra-TD, LKMM TD dan LKMW-TD yang diselenggarakan oleh departemen. Di lingkungan departemen, Penulis juga sempat aktif sebagai salah satu asisten laboratorium di Laboratorium Komputasi Berbasis Jaringan (KBJ) dan menjadi salah satu staf di Himpunan Mahasiswa Teknik-Computer Informatika (HMTC). Di luar lingkungan departemen, Penulis sempat aktif di UKM Badminton ITS dan pernah menjadi Koordinator Divisi WebApps ITS EXPO tahun 2021.

Berikut email dari Penulis: ivanabdillahrahman@gmail.com