BAB III
METODE PENELITIAN

Metodologi penelitian adalah penjelasan terstruktur yang memastikan proses penelitian dilakukan secara terarah dan sistematis. Sasaran penelitian dalam penyelesaian skripsi ini yaitu penggunaan *deep learning* dengan metode *convolutional neural network* dalam mendeteksi penyakit pada daun tanaman yang dipadukan dengan aplikasi berbasis android. Salah satu poin penting pada penelitian ini adalah kinerja dan akurasi dari model yang dibuat dalam mendeteksi penyakit pada daun tanaman. Selain itu, sisi fungsionalitas aplikasi juga menjadi poin penting lainnya apakah pengujian aplikasi berjalan lancar melalui metode *black box*.

Secara garis besar, Peneliti menerapkan metode pengembangan *Design and Development* (D&D). Metode ini digunakan untuk membuat aplikasi berbasis Android yang dikombinasikan dengan model *deep learning* dengan desain dan pengembangan yang dilakukan secara mandiri. Richey dan Klein dalam Hajidi 2019, metode *Design and Development* merupakan studi sistematis tentang proses pengembangan desain dan evaluasi, dengan tujuan untuk menciptakan produk dengan model baru atau menyempurnakan model yang sudah ada (J. Ellis & Levy, 2010).

Alur penelitian dalam pengembangan aplikasi ini terdiri dari 5 langkah secara garis besar seperti yang ditampilkan pada gambar 3.1 berikut

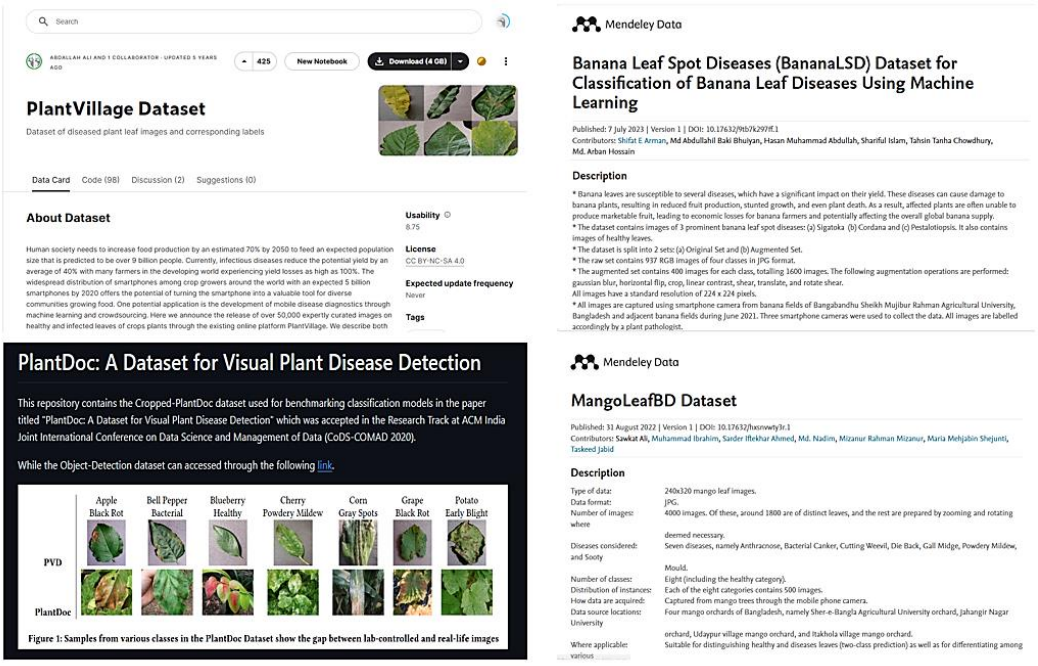
Gambar 3.1 Alur Penelitian (J. Ellis & Levy, 2010)

Penelitian diawali dengan analisis dimana pada bagian ini akan dijelaskan mengenai kebutuhan-kebutuhan yang diperlukan dalam penelitian ini. Kemudian dilanjutkan dengan desain yang menjelaskan tentang gambaran besar yang akan dilakukan. Pengembangan menjelaskan proses pembuatan model, API dan aplikasi yang dibangun. Pengujian dan Evaluasi menjelaskan proses evaluasi yang dilakukan setelah proses pengembangan selesai. Terakhir, pelaporan adalah proses dokumentasi dari penelitian yang telah dilakukan dimana ditulis pada skripsi ini.

3.1 Analisis

3.1.1 Analisis Model

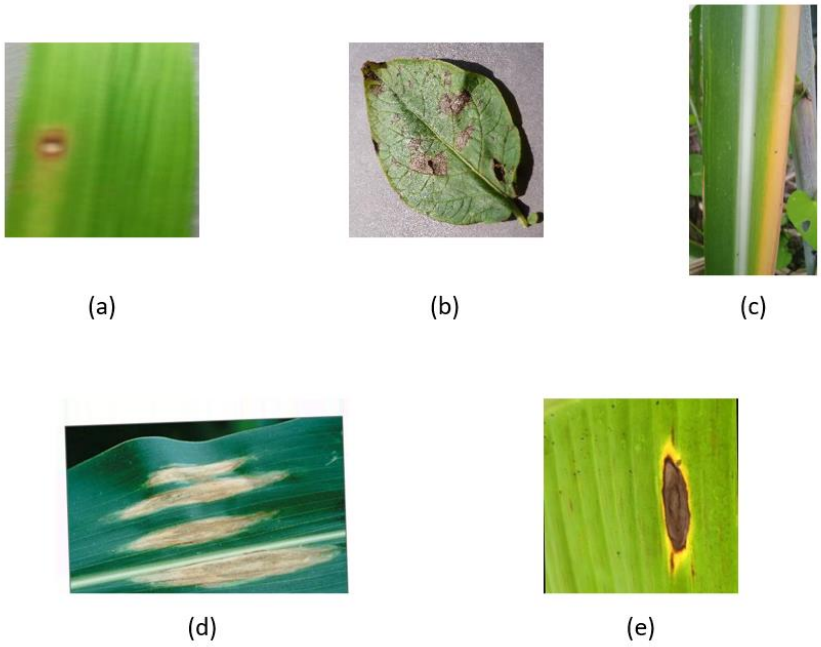
Analisis model diawali dengan proses pencarian dataset yang akan digunakan. Dataset untuk proses pembuatan model yang akan digunakan dalam penelitian ini adalah dataset yang bersifat *open source* dimana dataset ini tersedia untuk diunduh secara bebas di platform internet. Dataset ini diperoleh dari situs Kaggle, Mendeley Data, GitHub dan internet seperti yang terlihat pada gambar 3.2 di bawah.



Gambar 3.2 Sumber Dataset yang digunakan (PlantVillage, 2019), (Data Mendeley Banana, 2023), (PlantDoc Dataset, 2019), (Data Mendeley Mango, 2022)

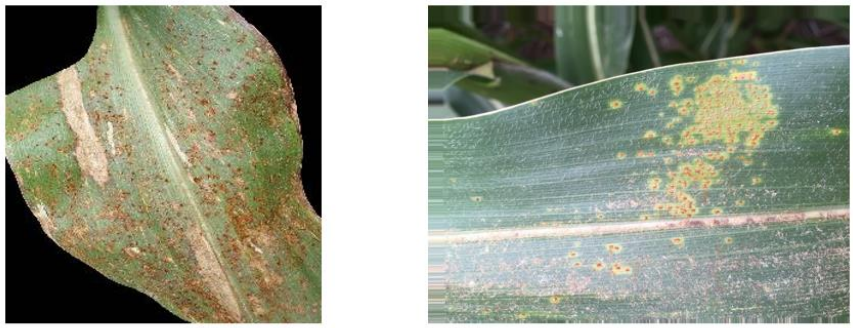
Kesemua dataset tersebut akan digabungkan menjadi sebuah satu folder baru dengan total dataset yang digunakan memiki jumlah gambar 19,986 yang terdiri dari 26 tipe penyakit pada daun tanaman diantaranya, Padi (Bacterialblight, Blast, Brownspot, Healthy), Mangga (Anthracnose, Bacterial Canker, Gall Midge, Powdery Mildew, Sooty Mould, Healthy), Tebu (Healthy, Mosaic, Redrot, Rust, Yellow), Jagung (Rust, Gray Leaf Spot, Blight, Healthy), Kentang (Early Blight, Late Blight, Healthy), dan Pisang (Cordana, Healthy,

Pestalotiopsis, Sigatoka). Gambar 3.3 merupakan beberapa contoh data dari dataset yang akan digunakan.



Gambar 3.3 Contoh Gambar dari Dataset yang digunakan: (a) Padi Blast (b) Kentang Early Blight (c) Tebu Yellow (d) Jagung Blight (e) Pisang Cordana

Terdapat dua dataset yang akan digunakan dalam penelitian ini diambil dari sebuah dataset yang sudah besar tersedia secara bebas dengan nama *PlantVillage* dan *PlantDoc*. Meski kedua dataset tersebut memiliki jumlah tanaman dan kelas yang banyak, namun disini, penulis hanya menggunakan yang sesuai dengan penelitian ini saja dan tidak menggunakan keseluruhan dataset. Penulis menggabungkan kedua dataset ini dengan tujuan untuk memperbanyak gambar-gambar yang bisa dilatih dan agar model lebih "general" dan fleksibel dalam mengenali gambar-gambar masukan seperti yang terlihat pada gambar 3.4 di bawah.

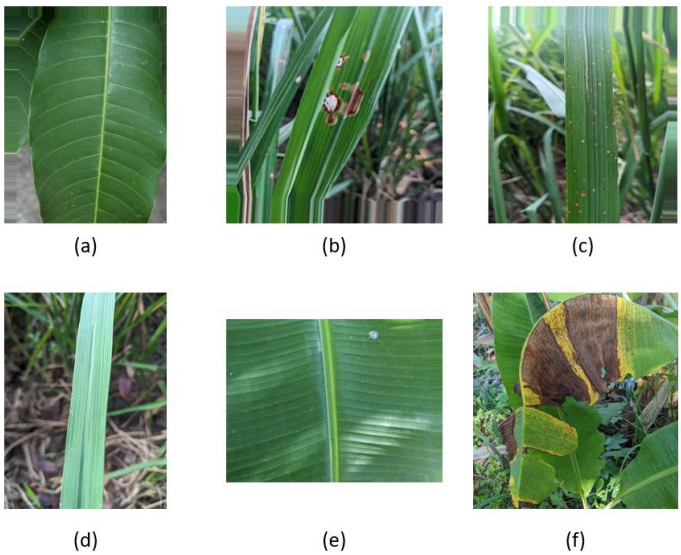


(a) (b)

Gambar 3.4 (a) Jagung Common Rust pada PlantVillage dan (b) Jagung Common Rust PlantDoc

Dataset PlantDoc yang digunakan dan digabungkan dengan dataset PlantVillage dan sesuai dengan penelitian ini yang digunakan adalah Jagung Blight, Jagung Common Rust, Jagung Gray Leaf Spot, Kentang Early Blight dan Kentang Late Blight.

Selain itu, penulis juga menambahkan beberapa gambar yang diambil dari *smartphone* penulis seperti yang terlihat pada gambar 3.5, untuk dijadikan sebagai data latih, validasi dan uji.



(a) (b) (c)

(d) (e) (f)

Gambar 3.5 (a) Mangga Healthy (b) Padi Blast (c) Padi Brownspor (d) Padi Healthy (e) Pisang Healthy (f) Pisang Cordana

Kelas pada Dataset yang ditambahkan antara lain Mangga Healthy, Padi Blast,

Padi Brownspot, Padi Healthy, Pisang Cordana dan Pisang Healthy.

3.1.2 Analisis API

Untuk API yang dibangun berfungsi sebagai komunikasi antara model dengan sebuah aplikasi android. Pada tahap pengembangan API, beberapa titik akses (endpoints) akan dibuat dan disesuaikan dengan fungsinya masing-masing. Beberapa fitur yang disediakan oleh API ini mencakup login, pembuatan akun baru, pemulihan password yang terlupa, tampilan halaman riwayat, proses deteksi penyakit daun, pengubahan profil pengguna, tampilan artikel, tampilan tanaman dan artikel. Setiap titik akses ini dibangun untuk menangani permintaan yang spesifik dari aplikasi Android dan dapat mengembalikan respons yang sesuai dengan permintaan sebelumnya. Selanjutnya, terdapat beberapa titik akses yang dibangun untuk adanya interaksi antara aplikasi dengan server, Misalnya, titik akses untuk masuk ke dalam aplikasi Android, titik akses untuk memanggil proses deteksi penyakit pada daun tanaman, titik akses untuk akses artikel, dan lainnya. Tabel 3.1 berikut ini daftar lengkap endpoint API yang dibangun beserta dengan fungsinya.

Tabel 3.1 Endpoint

Endpoints (titik akses)	Metode	Fungsi	Request
/register	POST	Proses pendaftaran akun oleh pengguna	username, password, nik, nama
/login	POST	Digunakan untuk login ke dalam aplikasi	username, password
/forgot	POST	Digunakan apabila pengguna lupa password	username, new_password

/update	POST	Digunakan apabila pengguna ingin merubah nama	nik,nama
/prediksiuploadopsi	POST	Proses deteksi penyakit	file_gambar
/updatecatatan	POST	Digunakan untuk menambah catatan setelah proses deteksi	id, catatan
/penyakit	GET	Digunakan untuk menampilkan penyakit daun tanaman yang bisa di deteksi pada penelitian ini	-
/artikel	GET	Digunakan untuk menampilkan artikel	-
/tanaman	GET	Digunakan untuk menampilkan tanaman	-
/riwayat	GET	Digunakan untuk menampilkan riwayat prediksi	nik

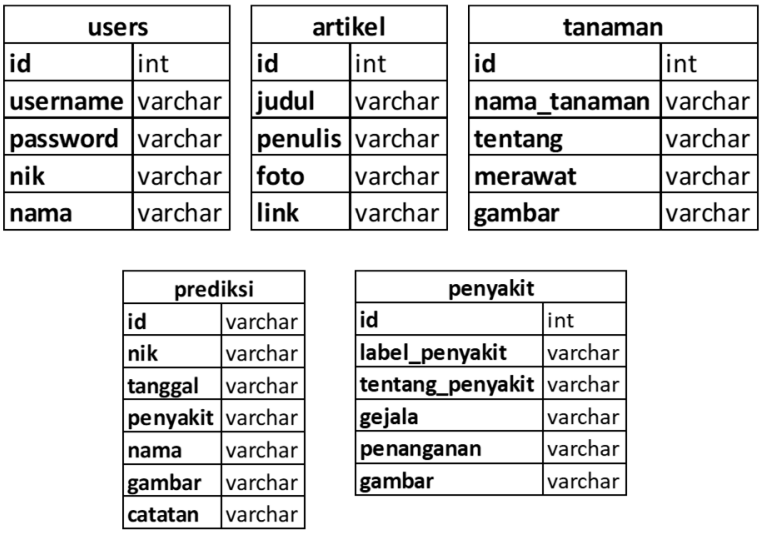
Setelah semua telah diuji dan berfungsi dengan baik, deploy API dilakukan dengan menggunakan aplikasi lainnya yaitu Ngrok sehingga dapat diakses dari aplikasi Android meskipun tidak dalam satu jaringan yang sama (localhost).

3.1.3 Analisis Aplikasi

Dalam tahap ini, kebutuhan untuk aplikasi deteksi penyakit daun

tanaman berbasis Android akan diidentifikasi dan dijabarkan. Aplikasi ini akan memiliki beberapa fitur pendukung untuk meningkatkan pengalaman pengguna, seperti login, pendaftaran akun baru, pengubahan kata sandi yang terlupa, halaman riwayat deteksi, deteksi penyakit daun, dan pengubahan profil pengguna. Aplikasi akan menggunakan database untuk menyimpan data pengguna, tanaman, penyakit, dan hasil klasifikasi, serta menghubungkan model deteksi penyakit melalui API yang telah dibangun. Analisis ini penting untuk memastikan semua kebutuhan dan fitur terdefinisi dengan jelas sebelum memulai tahap desain dan pengembangan. Pada penelitian ini, digunakan juga sebuah basis data untuk menyimpan data-data yang diperlukan oleh aplikasi yang dibangun. Basis data yang digunakan akan bersifat kumpulan data-data tabular yang disimpan dalam beberapa tabel. MySQL menggunakan bahasa SQL untuk melakukan operasi dan penghubung antara aplikasi dengan basis data yang berada pada server.

Pada penelitian ini terdapat satu buah basis data yang dibangun dan terdiri dari 5 tabel untuk masing-masing peruntukannya. Kelima tabel tersebut adalah users, artikel, tanaman, prediksi dan penyakit seperti yang terlihat pada gambar 3.6.



Gambar 3.6 Tabel yang digunakan pada Database

Tabel users, dipergunakan untuk menyimpan data-data users yang terdaftar pada aplikasi seperti yang terlihat pada Tabel 3.2

Tabel 3.2 Tabel Users pada Database

id	username	password	nik	nama
1	mdhkrmd	dhk	3215030	Andhika R
2	fadhli	dli	3216020	Fadhli Jabbar
3	abdi	abdi	3,22E+08	Abdi Perdana
4	ghalbin	gha	6349138	Ghalbin A
5	aksyal	syal	32167340	Aksyal Suseno

Tabel ini digunakan untuk proses melakukan login pada aplikasi, menyimpan data user baru dan mengubah password. Kemudian terdapat sebuah tabel artikel yang terlihat pada Tabel 3.3

Tabel 3.3 Tabel Artikel pada Database

id	judul	penulis	foto	link
1	uang: Tip	anduria.c	www.skin	rming/ber
2	dengan M	ia Riskita	www.skin	l/magazine
3	Awal Hin	Berkebun	www.skin	.co.id/me
4	rtani yang	ngkalanja	www.skin	trategi-be
5	i yang Bai	ahan.co.i	www.skir	co.id/cara-

Tabel artikel, dipergunakan untuk menyimpan data-data artikel yang akan ditampilkan pada aplikasi, terdiri dari judul, penulis, foto dan link. Terdapat tabel tanaman yang terlihat pada Tabel 3.4.

Tabel 3.4 Tabel Tanaman pada Database

id	ma_tanam	tentang	merawat	gambar
1	Jagung	Jagung	agar mere	ww.skinn
2	Kentang	Tanaman	Penanam	ww.skinni
3	Mangga	Tanaman	Penanam	ww.skinni
4	Padi	Tanaman	Penanam	www.skin
5	Pisang	Tanaman	Penanam	ww.skinni
6	Tebu	Tanaman	Penanam	w.skinnie.

Tabel tanaman, dipergunakan untuk menyimpan data 6 tanaman yang ada pada penelitian ini. Tabel ini akan menampilkan mengenai nama tanaman terkait, tentang, cara merawat dan gambar tanaman. Kemudian terdapat tabel prediksi

yang terlihat pada gambar 3.5 di bawah.

Tabel 3.5 Tabel Prediksi pada Database

id	nik	tanggal	penyakit	nama	gambar	catatan
652	3215030	23/07/2024 17:11	li_Browns	Andhika R	skinnie.m	Asli
653	3215030	23/07/2024 17:11	ngga_Hea	Andhika R	skinnie.m	Asli
654	3215030	23/07/2024 17:13	ang_Healt	Andhika R	skinnie.m	Asli
655	3215030	23/07/2024 17:19	ang_Cord	Andhika R	skinnie.m	Asli
656	3215030	23/07/2024 17:20	adi_Healt	Andhika R	skinnie.m	Asli

Tabel prediksi, dipergunakan untuk menyimpan hasil deteksi yang dilakukan oleh user. Ini dilakukan untuk membuat sebuah fitur riwayat pada aplikasi sehingga nantinya riwayat pengambilan gambar yang dilakukan oleh user akan tersimpan dan bisa dilihat pada halaman riwayat di aplikasi yang dibangun. Terakhir, tabel 3.6 menampilkan contoh sampel data pada tabel penyakit.

Tabel 3.6 Tabel Penyakit pada Database

id	label_penyakit	tentang_penyakit	gejala	penanganan	gambar
16	Padi_Brownspot	inthosporium ory	hingga m	1. Gunakan	nie.my.id/
17	Padi_Healthy	ehat, tidak terdetek	tidak terd	- Memilih	nie.my.id/
18	Pisang_Cordana	h dua spesies jam	elilingi ole	1. Gunakan	kinnie.my.
19	sang_Pestalotiop	neluas dan menya	nya terjad	1. Lesi dan ke	nnie.my.id/
20	Pisang_Sigatoka	yebabkan hilangn	r-angsur n	1.	kinnie.my.

Tabel terakhir adalah tabel penyakit. Ini digunakan untuk memberi informasi kepada user mengenai penyakit atau hasil dari deteksi yang dilakukan sebelumnya. Informasi yang diberikan berupa tentang penyakit yang dideteksi, gejala dan penanganan penyakit.

3.2 Desain

3.2.1 Desain Model

Setelah data untuk proses pembuatan model sudah didapatkan, augmentasi data merupakan langkah yang dilakukan selanjutnya. Augmentasi dilakukan agar dataset bervariasi dan tidak monoton sesuai dengan dataset yang ada. Selain membuat dataset menjadi bervariasi, augmentasi juga dapat meminimalisir terjadinya *overfitting*. Proses augmentasi yang dilakukan pada penelitian ini adalah dengan menerapkan horizontal flip atau membalikkan

gambar di sepanjang sumbu Y seperti yang terlihat pada gambar 3.7.

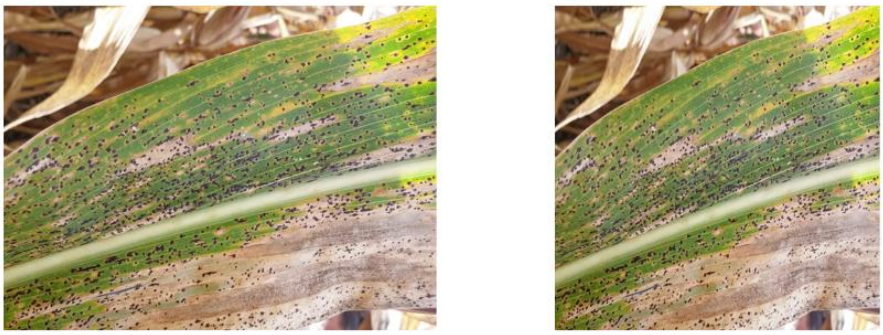
Gambar 3.7 Horizontal Flip

Berikut kode program untuk melakukan proses horizontal flip pada gambar 3.8.

```
Image_datagen = ImageDataGenerator(
                    preprocessing_function=effnetprep,
                    horizontal_flip=True,
                    )
```

Gambar 3.8 Kode Proses Horizontal Flip

Hal lain yang dilakukan pada pra-proses ini adalah dengan melakukan perubahan ukuran gambar menjadi satu ukuran seragam. Hal ini dilakukan agar struktur jaringan konvolusi bisa memproses data gambar yang masuk secara konsisten dari sisi ukuran gambar. Jika ukuran suatu pixel pada gambar dibuat lebih kecil. Maka ukuran data gambar tersebut akan semakin kecil pula, ini menyebabkan penggunaaan memori ketika proses pelatihan akan semakin lebih kecil. Pada penelitian ini ukuran gambar pelatihan yang berbagai macam akan diseragamkan ukuran gambar yang digunakan menjadi ukuran 456x456. Proses *resize* gambar akan dilakukan ketika pembuatan model berlangsung dengan menggunakan tools yang tersedia pada Tensorflow. Gambar 3.9 memperlihatkan contoh gambar yang di *resize*



(a) (b)

Gambar 3.9 Resize Gambar, (a) Sebelum resize, (b) resize 456x456

Berikut ini merupakan kode program untuk melakukan proses *resizing* pada gambar 3.10.

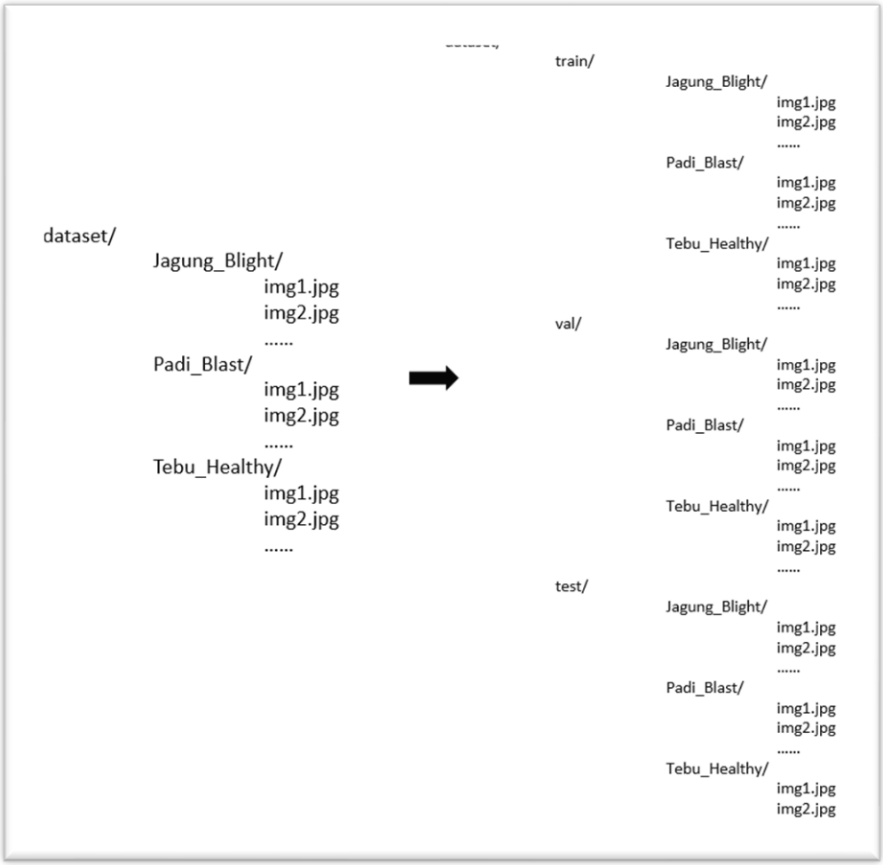
```
target=(456,456)
batchS = 8

train_generator = image_datagen.flow_from_directory(
        train_dir,
        target_size=target,
        batch_size=batchS,
        color_mode='rgb',
        shuffle=True,
        class_mode='categorical'
        )

validation_generator = image_datagen.flow_from_directory(
        val_dir,
        target_size=target,
        batch_size=batchS,
        color_mode='rgb',
        shuffle=False,
        class_mode='categorical'
        )
```

Gambar 3.10 Proses *Resizing* gambar

Terlihat bahwa proses gambar akan dilakukan *resize* menjadi 456x456 yang terlihat pada variabel target = (456,456).

Setelah praproses data selesai dilakukan, pada penelitian ini, data yang

digunakan akan dibagi menjadi 3 bagian yaitu data latih, validasi dan uji. Data latih menggunakan 70% dari total gambar, kemudian data validasi dan data uji masing-masing sekitar 15% dari total gambar. Dari total 19,986 yang terdiri dari 26 tipe penyakit pada daun tanaman, pada penelitian ini, dataset dibagi menjadi 3 bagian yaitu data latih, validasi dan uji. Proses pembagian dataset dilakukan dengan menggunakan library *splitfolders* dengan menggunakan bahasa Python. Total gambar latih yang digunakan adalah sebesar 13977 gambar, 2987 untuk data validasi dan 3022 untuk data test, terlihat pada gambar 3.11 di bawah.



Gambar 3.11 Pembagian Dataset

Terlihat bahwa dari yang awalnya folder dataset terdiri dari nama-nama kelasnya saja, setelah dilakukan proses split, folder tersebut dipecah menjadi folder train, val dan test dimana masing-masing folder tersebut akan digunakan

sebagaimana penggunaannya. Berikut ini merupakan kode terkait untuk melakukan proses splitting gambar seperti terlihat pada gambar 3.12.

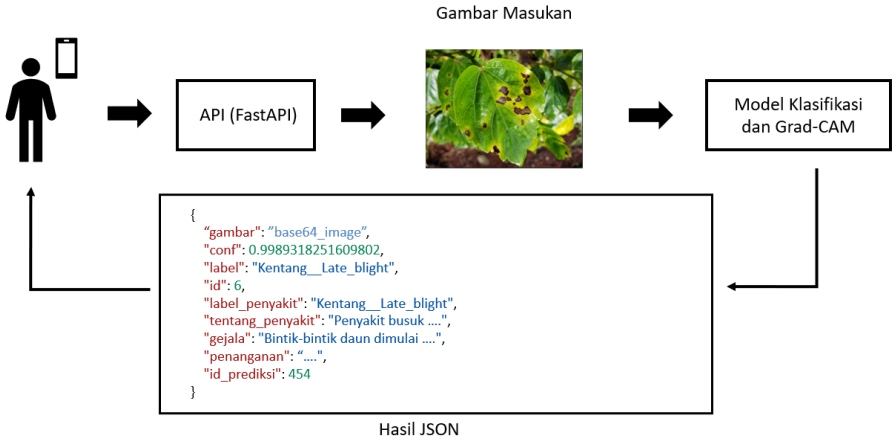
```
import splitfolders
input_folder = r"F:\Kuliah\Semester 8\Dataset - Terpakai\Pl
antDoc Aug"
output = r"F:\Kuliah\Semester 8\Dataset - Terpakai\PlantDoc
Aug - Split"

splitfolders.ratio(input_folder, output=output, seed=42, ra
tio=(0.7, 0.15, 0.15))
```

Gambar 3.12 Kode *splitting* gambar

Pada kode ini diperlukan library tambahan bernama *splitfolders* untuk melakukan proses splitting di atas. Data latih digunakan untuk melatih model sehingga model dapat mengenali pola-pola penting yang ada dalam gambar. Data validasi digunakan untuk memantau kinerja model selama proses pelatihan dan untuk menyesuaikan *hyperparameter* agar model tidak *overfitting*. Sementara itu, data uji digunakan untuk mengukur performa model setelah proses pelatihan selesai, data uji akan memberikan gambaran seberapa baik model tersebut dapat mengenali data yang belum pernah dilihat sebelumnya.

3.2.2 Desain API

Fungsi API adalah untuk menghubungkan model yang berada pada sebuah server dengan sebuah aplikasi berbasis android. API yang dibangun pada penelitian ini akan diletakan dan dijalankan pada laptop milik pribadi. Alasan penulis menggunakan API dibandingkan dengan menerapkan langsung (model) di android adalah dikarenakan akses kepada database MySQL dan juga proses model deteksi yang pada akhirnya tidak hanya untuk mengklasifikasikan penyakit saja, tetapi juga proses Grad-CAM dan deteksi (kotak pembatas). API ini dikembangkan memanfaatkan *framework* FastAPI dengan bahasa pemrograman Python serta MySQL Connector. Gambaran mengenai proses prediksi melalui akses API dijelaskan pada gambar 3.13 di bawah ini.



Gambar 3.13 Proses API

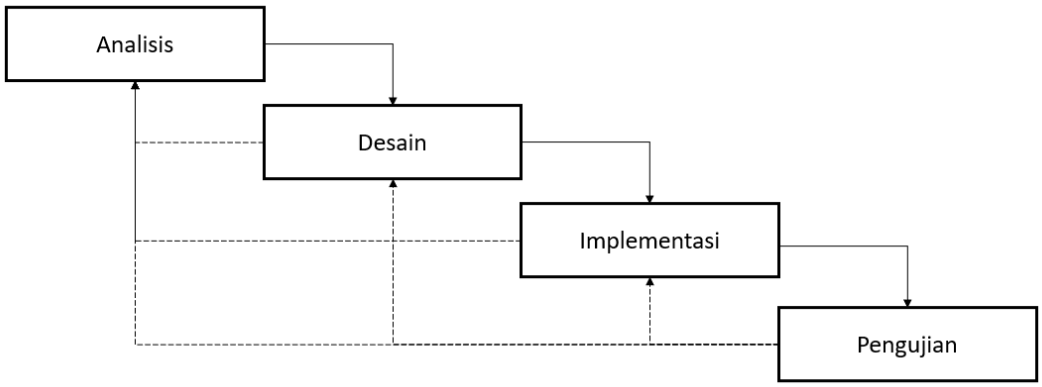
Proses dimulai ketika pengguna membuka aplikasi dimana aplikasi tersebut mengakses API yang sudah dibangun. Ketika pengguna memasukan gambar yang ingin diprediksi, gambar tersebut dikirimkan melalui jaringan internet untuk nantinya diproses di dalam server (dalam hal ini laptop penulis). Setelah proses selesai, hasil prediksi akan dikembalikan ke dalam aplikasi melalui file yang berformat JSON seperti pada gambar di atas..

3.2.3 Desain Aplikasi

Untuk pembuatan aplikasi dalam penelitian ini, penulis akan menerapkan model SDLC (*Software Development Life Cycle*) untuk mengembangkan aplikasi. SDLC merupakan metode sistematis yang digunakan dalam pembuatan perangkat lunak. Pendekatan ini membantu mengatur proses pembuatan, pengembangan, pengujian, dan pemeliharaan perangkat lunak secara lebih efisien dan terstruktur (Permana, 2023).

Pada penelitian ini, metode SDLC yang digunakan adalah metode *Waterfall*. Metode ini disebut *Waterfall* karena setiap tahapannya harus dilalui secara berurutan, seperti air terjun yang mengalir dari satu tingkat ke tingkat berikutnya. Setiap tahapan harus diselesaikan sepenuhnya sebelum melanjutkan ke tahap berikutnya (Pargaonkar, 2023). Dengan metode ini, penulis dapat mengurangi risiko kesalahan yang dapat terjadi dan proses pembuatan aplikasi jelas dan terarah. Gambar 3.14 memvisualisasikan proses

dari SDLC *Waterfall* yang digunakan



Gambar 3.14 Proses *Waterfall* (Pargaonkar, 2023)

Proses ini dimulai dengan analisis kebutuhan untuk memahami apa saja yang dibutuhkan oleh aplikasi yang ingin dibangun, kemudian desain aplikasi yang akan dijelaskan dengan *use case diagram*, implementasi aplikasi yang akan menjelaskan alat dan bahasa Pemrograman yang digunakan dan ditutup dengan skenario pengujian aplikasi dengan menggunakan Black box.

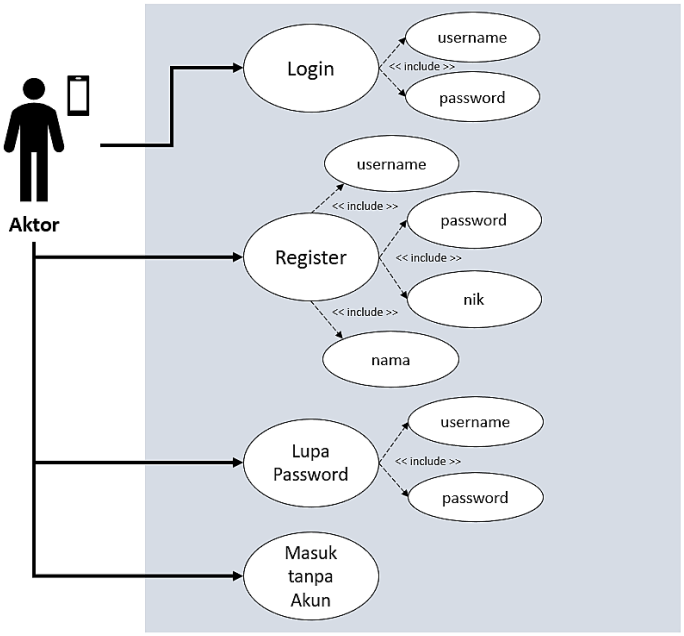
a. Analisis

Pada metode *Waterfall*, tahap pertama yang harus dilakukan adalah proses analisis. Dalam tahap ini, semua kebutuhan untuk aplikasi deteksi penyakit pada daun tanaman berbasis Android akan diidentifikasi dan dijabarkan lebih lanjut. Aplikasi yang dibangun akan terdiri dari beberapa fitur pendukung yang bertujuan untuk meningkatkan pengalaman pengguna ketika memakainya. Aplikasi ini akan memerlukan database untuk menyimpan informasi penting seperti data pengguna, data tanaman, data penyakit dan hasil klasifikasi. Selain itu, aplikasi ini juga akan menggunakan model yang telah dibangun sebelumnya untuk mendeteksi penyakit pada daun yang akan dihubungkan melalui API yang dibangun. Fitur-fitur yang akan disertakan dalam aplikasi ini meliputi login untuk akses pengguna, pendaftaran akun baru, fitur untuk mengubah kata sandi yang terlupa, halaman riwayat untuk melihat hasil deteksi sebelumnya, proses deteksi penyakit daun, dan fitur untuk mengubah profil pengguna.

Proses analisis ini sangat penting untuk memastikan bahwa semua kebutuhan dan fitur yang akan dibangun terjabarkan dengan jelas sebelum tahap desain dan pengembangan dimulai.

b. Desain

Proses desain ini akan menggambarkan hasil analisis sebelumnya ke dalam bentuk *Use Case Diagram* sebagai bentuk representasi antara sistem dengan pengguna. *Use Case Diagram* adalah diagram yang mendefinisikan desain hubungan antara pengguna dan sistem. Terdapat dua *use case diagram* yang akan dijelaskan pada bagian ini, yaitu sebelum melakukan login aplikasi dan sesudah melakukan login aplikasi. Gambar 3.15 menjelaskan bagaimana *use case diagram* sebelum melakukan login aplikasi.

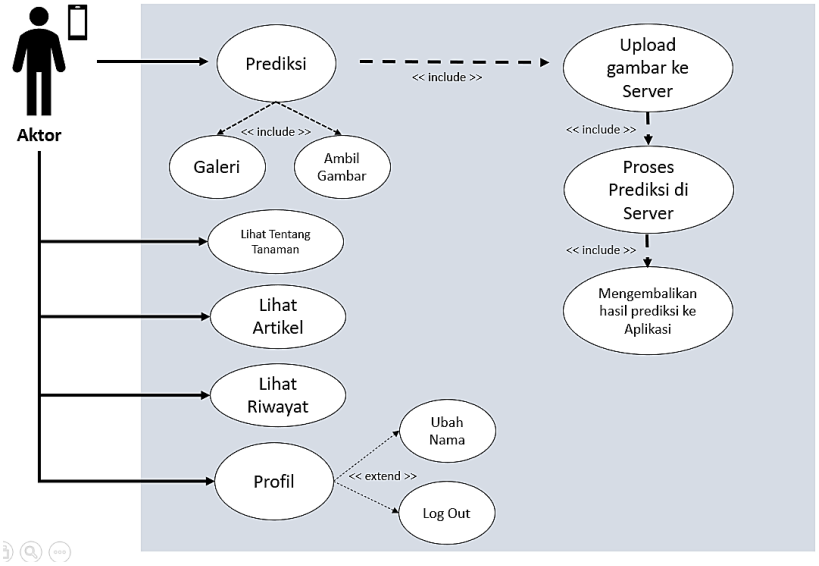


Gambar 3.15 *Use Case Diagram* Sebelum Masuk Halaman Utama

Ketika pengguna belum masuk ke dalam halaman utama, maka pengguna akan dihadapkan dengan beberapa pilihan yaitu Login, Register, Lupa Password, dan Masuk tanpa Akun. Fitur pertama adalah Login, di mana pengguna harus memasukkan username dan password untuk masuk ke dalam sistem. Kedua atribut ini dihubungkan dengan relasi <<include>>, yang menunjukkan bahwa username dan password diperlukan dalam proses

login. Fitur kedua adalah Register, yang memungkinkan pengguna untuk membuat akun baru. Proses registrasi ini memerlukan beberapa informasi dari pengguna, yaitu username, password, nik, dan nama. Setiap atribut ini juga dihubungkan dengan relasi <<include>>, menunjukan bahwa data ini diperlukan ketika proses pembuatan akun baru. Fitur ketiga adalah Lupa Password, yang memungkinkan pengguna untuk mengganti password yang terlupa. Proses ini juga memerlukan masukan username dan password baru untuk mengubah password lama yang tersimpan. Terakhir, fitur Masuk tanpa Akun, memungkinkan pengguna untuk mengakses aplikasi tanpa perlu melakukan login atau daftar akun terlebih dahulu, meskipun akses ini akan terbatas dibandingkan dengan pengguna yang sudah terdaftar.

Setelah menjelaskan proses *use case diagram* ketika pengguna belum melakukan proses login, gambar 3.16 di bawah menjelaskan bagaimana proses *use case diagram* setelah pengguna melakukan proses login pada aplikasi.

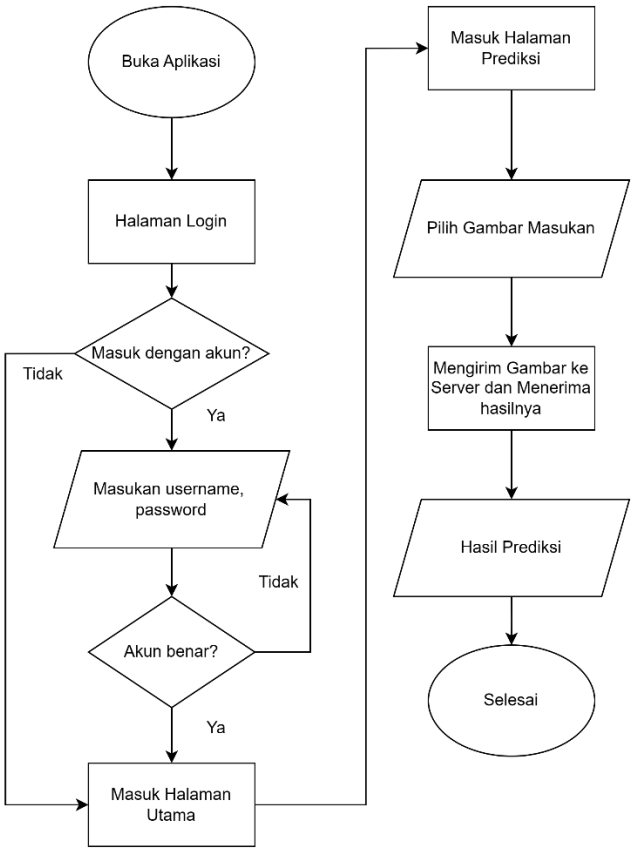


Gambar 3.16 *Use Case Diagram* Setelah Login

Kemudian setelah login, penggunaan untuk aplikasi ini dimulai dengan aktor membuka aplikasi kemudian aktor terdapat tiga fitur utama yang bisa dilakukan. Aktor bisa melakukan prediksi, lihat riwayat dan profil. Proses

prediksi bisa dilakukan dengan dua cara yaitu memilih gambar yang sudah ada di dalam galeri atau dengan mengambilnya secara langsung menggunakan kamera yang ditandai dengan relasi <<include>> dimana proses prediksi memerlukan gambar masukan. Proses prediksi selanjutnya adalah dengan mengirimkan gambar yang dipilih ke sebuah server untuk dilakukannya proses prediksi dan proses *bounding box* dengan Grad-CAM. Setelah selesai hasil prediksi tersebut akan dikembalikan kembali dalam bentuk format data JSON. Fitur lihat riwayat akan menampilkan beberapa hasil prediksi yang pernah dilakukan sebelumnya oleh pengguna dan fitur profil akan menampilkan pilihan seperti ubah nama dan *logout* dari aplikasi yang ditandai dengan relasi <<extend>>. Kemudian terdapat dua fitur pendukung lainnya seperti lihat artikel dan lihat tentang tanaman. Fitur melihat artikel memberikan akses kepada pengguna untuk membaca berbagai artikel terkait dengan tanaman pada penelitian ini, sedangkan fitur melihat informasi tentang tanaman menyediakan informasi mengenai berbagai jenis tanaman yang digunakan pada penelitian ini.

Setelah proses *use case diagram* yang telah dijelaskan sebelumnya, Gambar 3.17 menjelaskan proses flowchart untuk prediksi penyakit.



Gambar 3.17 Flowchart Proses Prediksi Penyakit

Flowchart sistem di atas menggambarkan proses rangkaian ketika pengguna akan melakukan proses prediksi di aplikasi. Proses diawali dengan pengguna membuka aplikasi yang dilanjutkan dengan menampilkan halaman untuk proses login. Pada bagian ini pengguna terdapat pilihan yaitu pengguna diminta untuk memasukan username dan password atau "masuk tanpa akun". Keterbatasan akses ketika memilih "masuk tanpa akun" adalah pada nantinya pengguna tidak bisa melihat riwayat hasil proses prediksi yang pernah dilakukan sebelumnya. Kemudian pengguna akan diarahkan ke halaman utama, disini pengguna mengakses halaman prediksi untuk melakukan proses deteksi. Pada halaman prediksi, disini pengguna diminta untuk memasukan gambar yang akan dilakukan proses deteksi. Setelah gambar sudah dipilih, proses selanjutnya adalah mengirimkan gambar

tersebut ke server dan akan diproses untuk nantinya dibalikan hasilnya kembali ke dalam aplikasi. Hasil deteksi yang diterima antara lain nama penyakit, probabilitas prediksi, tentang penyakit, gejala yang terlihat dan penanganan yang bisa dilakukan.

c. Implementasi

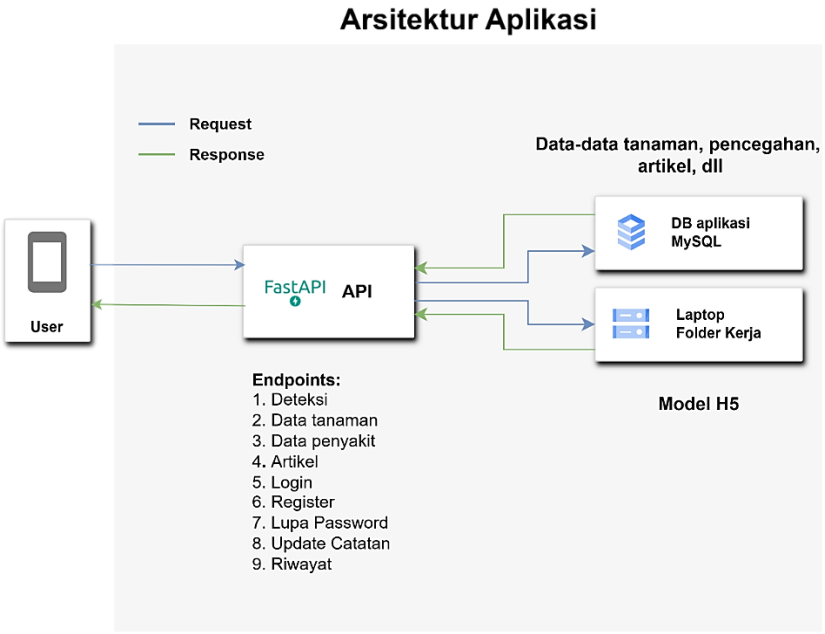
Proses implementasi aplikasi ini dilakukan menggunakan *Integrated Development Environment* (IDE) Android Studio, yang merupakan IDE umum untuk pengembangan aplikasi Android. Bahasa pemrograman yang digunakan adalah Java, yang masih bisa diandalkan untuk pengembangan aplikasi berbasis android. Selain itu, beberapa library pendukung akan digunakan untuk mempermudah dan mempercepat proses pengembangan aplikasi, seperti Retrofit untuk mengelola dan menghubungkan aplikasi dengan API, serta Glide untuk memuat dan menampilkan gambar dari sebuah URL.

d. Pengujian

Proses pengujian akan dilakukan menggunakan metode black box, yang fokus pada pengujian fungsionalitas aplikasi tanpa memeriksa kode yang sudah dibangun. Pengujian black box akan memastikan bahwa semua fitur aplikasi, seperti deteksi penyakit daun, login, pendaftaran akun, lupa password, halaman riwayat, proses deteksi, dan ubah profil, berfungsi sesuai dengan skenario yang telah ditentukan. Dalam pengujian ini, aplikasi akan diuji dengan berbagai input untuk memastikan bahwa output yang dihasilkan sesuai dengan yang diharapkan. Setiap kali adanya bug atau kesalahan yang ditemukan selama pengujian akan diperbaiki, sehingga aplikasi dapat berjalan dengan lancar dan memberikan pengalaman pengguna yang optimal.

Kemudian, terdapat sebuah desain arsitektur aplikasi (*backend)* yang dibangun. Ini memberikan gambaran bagaimana sebuah proses aplikasi yang akan

dirancang dan diimplementasikan dari penelitian ini. Penelitian ini akan mengkombinasikan perangkat android dengan model *deep learning* yang dibangun kemudian akan disambungkan dengan sebuah *Application Programming Interface* atau API seperti yang terlihat pada gambar 3.18.



Gambar 3.18 Arsitektur Aplikasi yang dibangun

Pada android, diperlukan sebuah kode untuk menghubungkan aplikasi dengan API yang berada di server eksternal. Kemudian, API akan terhubung dengan aplikasi Android untuk bertukar data antara server dengan aplikasi. Model *deep learning* akan berada pada sebuah server dimana dalam hal ini server eksternal merupakan laptop penulis sehingga model tidak langsung dibenamkan pada *smartphone*. Kemudian API juga berfungsi untuk mengakses data-data yang berada di *database.* Database ini terletak di server eksternal, dalam hal ini laptop penulis, sehingga tidak terintegrasi langsung dengan aplikasi Android. Sebagai perantara, API memungkinkan aplikasi Android untuk berkomunikasi dengan database, mengambil, dan menyimpan data. Dengan kata lain, API bertindak sebagai jembatan yang memungkinkan aplikasi Android berinteraksi dengan informasi yang ada di server eksternal.

3.3 Pengembangan

3.3.1 Pengembangan Model

Proses pembuatan model akan dilakukan dengan menggunakan dukungan *Graphical Processing Unit (GPU)* berbasis *cloud* dimana penelitian ini menggunakan *Kaggle* sebagai platform untuk melakukan pembuatan model tersebut yang nantinya akan digunakan dalam aplikasi berbasis android. Spesifikasi dari GPU yang digunakan adalah dengan menggunakan Nvidia T4 yang berjumlah 2 yang memiliki total memori sebesar 16GB. Pembuatan model dilakukan dengan menggunakan bahasa pemrograman Python dan library Tensorflow versi 2.9.1.

Transfer Learning akan digunakan sebagai metode untuk mengurangi beban komputasi dan model yang dihasilkan cukup baik dari sisi akurasi dan meminimalisir kesalahan. Proses pelatihan menggunakan bobot model yang telah dilatih sebelumnya yaitu ImageNet. Dataset ImageNet berisi sekitar 1,2 juta gambar dan 1000 kategori kelas. Bobot ImageNet ini digunakan sehingga model tidak perlu melakukan proses kalkulasi bobot dari awal. Lapisan *Fully Connected* terakhir diubah dari 1000 keluaran menjadi 26 keluaran sesuai dengan kelas atau tipe penyakit pada penelitian ini. *Softmax* dipilih sebagai fungsi aktivasi di lapisan terakhir dan loss function menggunakan categorical cross entropy. Selama pelatihan, teknik *early stopping* juga digunakan apabila validation loss tidak mengalami penurunan. Karena teknik *early stopping* digunakan, epoch maksimum tidak ditentukan dalam pelatihan model. Parameter lainnya adalah dengan menggunakan Adamax *optimizer* dengan *learning rate* 0.001 diawal yang nantinya berubah menjadi 0.0001 hingga 0.00005. Terakhir, gambar masukan untuk model pada penelitian ini adalah dengan berukuran 456x456 pixels untuk EfficientNet B5. Tabel 3.7 merangkum lengkap *hyperparameter tuning* yang akan digunakan dalam penelitian ini.

Tabel 3.7 *Hyperparameter Tuning* yang akan digunakan

Model	Ukuran Gambar	Optimizer	Loss	*Learning rate*
EfficientNet B5	456x456	Adamax(beta_1=0.9,beta_2=0.999)	Categorical Cross Entropy	0.001, 0.0001, 0.00005

Proses perancangan model diawali dengan mendefinisikan gambar masukan sesuai dengan pra-proses sebelumnya, dimana gambar masukan akan berukuran 456x456 pixel seperti yang terlihat pada gambar 3.19

```
# Create Model Structure
img_size = (456,456)
channels = 3
img_shape = (img_size[0], img_size[1], channels)
class_count = len(list(train_generator.class_indices.keys()
)) # to define number of classes in dense layer
```

Gambar 3.19 Proses Perancangan Awal Model

Gambar yang dimasukan ini terdapat 3 channel yang mendefinisikan Red, Green, Blue pada gambar. Kemudian proses selanjutnya adalah dengan mendefinisikan model yang akan digunakan yang terlihat pada gambar 3.20.

```
# create pre-trained model
# we will use efficientnetb5 from EfficientNet family.
base_model = tf.keras.applications.efficientnet.EfficientNe
tB5(include_top= False, weights='imagenet', input_shape= im
g_shape, pooling= 'max')
```

Gambar 3.20 Proses Mendefinisikan Model EfficientNet

Pada penelitian ini digunakan model EfficientNetB5 dan menggunakan bobot *imagenet*. Kemudian *include_top=False*, ini dimaksudkan agar tidak menggunakan lapisan *Fully Connected* yang berjumlah 1000 keluaran sementara pada penelitian ini hanya terdapat 26 keluaran sesuai dengan kelas atau tipe penyakit pada penelitian ini seperti yang terlihat pada gambar 3.21 di

bawah.

```
# Build the model
x = base_model.output
x = BatchNormalization(axis=-1, momentum=0.99, epsilon=0.00
1)(x)
x = Dense(256, kernel_regularizer=regularizers.l2(l=0.016),
activity_regularizer=regularizers.l1(0.006),
          bias_regularizer=regularizers.l1(0.006), activati
on='relu')(x)
x = Dropout(rate=0.45, seed=123)(x)
x = Dense(128, kernel_regularizer=regularizers.l2(l=0.016),
activity_regularizer=regularizers.l1(0.006),
          bias_regularizer=regularizers.l1(0.006), activati
on='relu')(x)
x = Dropout(rate=0.30, seed=123)(x)
predictions = Dense(class_count, activation='softmax')(x)

# Create the final model
model = Model(inputs=base_model.input, outputs=predictions)
```

Gambar 3.21 Proses Perancangan Model *Fully Connected*

Pada bagian ini, didefinisikan model utuh yang akan digunakan dalam penelitian. Disini model EfficientNetB5 akan disambungkan dengan lapisan *Fully Connected* yang digunakan dalam penelitian ini yaitu fully connected layer dengan 256 unit neuron yang kemudian dilanjutkan dengan 123 unit neuron. Adapun lapisan pendukung lainnya seperti *Batch Normalization* dan *Dropout* yang digunakan dalam penelitian agar model yang dibangun terhindar dari masalah *overfitting.* Gambar 3.22 melihatkan proses *compile* model.

```
# Compile the model
model.compile(optimizer=Adamax(learning_rate=0.001), loss='
categorical_crossentropy', metrics=['accuracy'])
```

Gambar 3.22 Proses *Compile* model

Selanjutnya, dilakukan proses *compile* (fiksasi model dengan menambahkan konfigurasi nilai *hyperparameter tuning*) terhadap model yang akan dibangun, dengan menggunakan optimizer *Adamax* dan *learning_rate* sebesar 0.001. Kemudian, setelah model didefinisikan, proses pelatihan akan dilakukan

dengan menambahkan kode model.fit terlihat pada gambar 3.23.

```
history = model.fit(x= train_generator, epochs= epochs, ver
bose= 0, callbacks= callbacks,
                    validation_data= validation_generator,
validation_steps= None, shuffle= False)
```

Gambar 3.23 Proses Fit Model

Pada bagian ini, parameter-parameter yang diberikan kepada model.fit telah disesuaikan dengan kebutuhan. Misalnya, pada train_generator yang digunakan sebagai memasukan data pelatihan dan epochs menentukan jumlah epoch yang digunakan dalam pelatihan. Callbacks juga dapat ditambahkan untuk melakukan tindakan tertentu selama pelatihan, seperti menyimpan model atau menghentikan pelatihan jika tercapai kondisi tertentu. Selain itu, data validasi dapat diberikan menggunakan parameter validation_data dengan validation_generator sebagai generator data validasi. Proses pelatihan ini akan memakan waktu yang cukup lama dimana proses melatih model ini dilakukan untuk mengidentifikasi pola-pola dalam data pelatihan dan meningkatkan akurasi model seiring berjalannya waktu. Pada bagian terakhir ini, berfungsi untuk melakukan simpan model yang sudah dilatih sebelumnya. Gambar 3.24 memperlihatkan proses penyimpanan model.

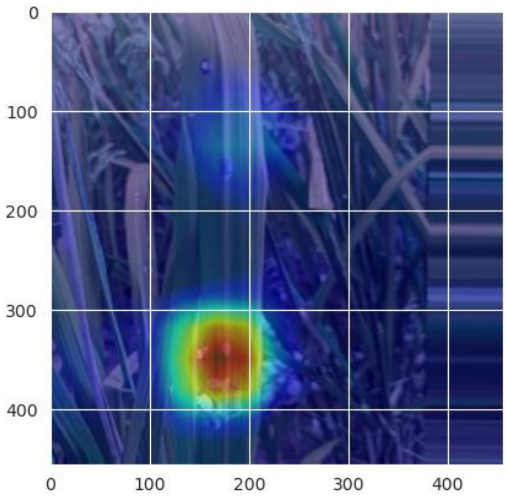
```
model_name = model.input_names[0][:-6]
subject = 'Real Data W PlantDoc - Cleaned - 456'
acc = test_score[1] * 100
save_path = ''

# Save model
save_id = str(f'{model_name}-{subject}-{"%.2f" %round(acc,
2)}.h5')
model_save_loc = os.path.join(save_path, save_id)
model.save(model_save_loc)
print(f'model was saved as {model_save_loc}')
```

Gambar 3.24 Proses Penyimpanan Model

Tujuan penyimpanan model ini adalah untuk nantinya digunakan dan dihubungkan ke sebuah aplikasi berbasis android melalui API yang dibangun.

Adapun algoritma Grad-CAM yang digunakan nantinya untuk

membangun *bounding box*, digunakan setelah proses pelatihan selesai dengan memanfaatkan bobot-bobot pada pelatihan model yang telah dilakukan. Grad-CAM akan digunakan untuk menghasilkan *heatmap* yang menunjukkan area-area penting pada gambar yang diperhatikan oleh model saat membuat hasil prediksi. *Heatmap* ini yang kemudian digunakan untuk membangun *bounding box*. Tujuan penelitian ini memanfaatkan Grad-CAM adalah untuk membuat sebuah kotak pembatas pada penyakit daun tanaman sehingga penelitian membutuhkan sebuah dataset daun tanaman yang sudah melewati proses labelling untuk proses evaluasi nantinya. Gambar 3.25 menunjukan contoh penggunaan Grad-CAM untuk membentuk sebuah kotak pembatas yang menunjukan penyakit pada sebuah daun tanaman.



Gambar 3.25 Contoh Pengaplikasian Grad-CAM

Pada gambar tersebut terlihat bahwa model terfokus pada suatu area, dimana semakin merah bagian tersebut menandakan bahwa bagian tersebutlah yang menjadi fokus utama model dalam membuat sebuah prediksi kelas penyakit. Pada penelitian ini, akan diberikan sebuah *threshold* sebesar 80%, dimana *threshold* tersebut digunakan untuk membuat sebuah *bounding box* disekitar *heatmap* pada bagian daun yang terdeteksi penyakit. Pemberian *threshold* = 0.8 menentukan seberapa signifikan suatu area harus dianggap penting untuk dideteksi dan disorot oleh *bounding box*. Area dengan aktivasi di bawah nilai *threshold* ini akan diabaikan, sementara area di atasnya akan ditandai. Tujuan

penggunaan threshold ini adalah untuk membatasi jumlah *bounding box* yang ditampilkan nantinya, sehingga menghindari adanya *bounding box* yang tidak relevan atau tidak seharusnya ditampilkan.

3.3.2 Pengembangan API

API ini dibangun menggunakan bahasa pemrograman Python dan memanfaatkan beberapa *library* seperti TensorFlow, Keras untuk model CNN, FastAPI untuk pembuatan API yang cepat dan efisien, serta MySQL Connector untuk menghubungkan dan mengelola basis data. Untuk memastikan API berfungsi dengan baik dan sesuai dengan kebutuhan aplikasi, Postman akan digunakan sebagai alat pengujian untuk memverifikasi setiap titik akses yang telah dibangun.

3.3.3 Pengembangan Aplikasi

Proses pengembangan aplikasi ini dilakukan menggunakan Android Studio, IDE umum untuk pengembangan aplikasi Android. Bahasa pemrograman yang digunakan adalah Java.

Lingkungan pengembangan yang digunakan dalam penelitian ini adalah sebuah laptop merek Acer untuk menjalankan seluruh proses penelitian. Spesifikasi laptop tersebut adalah sebagai berikut:

1. Intel Core i3 6006u
2. Memory RAM sebesar 12GB
3. Kartu Grafis Nvidia MX130, serta kartu grafis lainnya yaitu Intel HD Graphics 520
4. Sistem Operasi Windows 10.

3.4 Pengujian dan Evaluasi

3.4.1 Pengujian dan Evaluasi Model

Pengujian dan evaluasi model dilakukan setelah proses pelatihan model selesai. Pengujian dilakukan dengan dua skenario yaitu dengan menggunakan data uji dan uji langsung menggunakan kamera pada *smartphone*. Evaluasi dilakukan untuk mencari tahu apakah model yang dihasilkan dari proses

pelatihan sudah baik atau masih perlu adanya perbaikan. Proses evaluasi ini melibatkan hasil akurasi pada data uji dan juga metrik yang telah dijelaskan sebelumnya yaitu *Precision, Accuracy, Recall dan F1 Score*.

3.4.2 Pengujian dan Evaluasi API

Pengujian API yang telah dibangun akan menggunakan aplikasi tambahan yaitu Postman. Postman akan sangat berguna untuk menguji dan memvalidasi API dengan mengirimkan permintaan HTTP ke titik akses yang telah dibuat. Postman memungkinkan untuk melihat respon yang diberikan oleh API, memastikan bahwa data yang dikirim dan diterima sesuai dengan skenario yang diharapkan. Postman dapat mendeteksi apabila terdapat kesalahan lebih awal, sehingga memastikan API yang dibuat berfungsi dengan baik sebelum diintegrasikan ke dalam aplikasi Android

3.4.3 Pengujian dan Evaluasi Aplikasi

Seperti yang sudah dijelaskan pada model *waterfall* sebelumnya, pengujian aplikasi akan dilakukan dengan metode *black box,* yang fokus pada pengujian fungsionalitas tanpa memeriksa kode. Metode ini memastikan semua fitur, seperti deteksi penyakit daun, login, pendaftaran akun, lupa password, halaman riwayat, proses deteksi, dan ubah profil, berfungsi sesuai skenario yang ditentukan

3.5 Pelaporan

Setelah proses-proses sebelumnya telah dilakukan dan proses pengujian dan evaluasi telah dilakukan, pelaporan dilakukan dengan menulis hasil-hasil penelitian ini ke dalam bentuk skripsi yang mencakup pendahuluan, kajian pustaka, metode penelitian, pembahasan dan hasil penelitian, kesimpulan dan saran. Pada bagian pendahuluan, diuraikan latar belakang masalah, rumusan masalah, tujuan penelitian, serta manfaat penelitian. Tinjauan pustaka berisi landasan teori yang mendukung penelitian ini. Metode penelitian menjelaskan secara rinci prosedur yang digunakan dalam penelitian ini. Pembahasan dan Hasil Penelitian memaparkan hasil-hasil yang diperoleh dari penelitian ini. Kesimpulan menjelaskan hasil utama penelitian, dan saran memberikan rekomendasi untuk penelitian selanjutnya dari penelitian ini. Penulisan ini

disajikan secara sistematis dan jelas agar harapannya dapat mudah dipahami oleh pembaca.