TUGAS AKHIR – EF234801

# PENGEMBANGAN APLIKASI BERBASIS ANDROID UNTUK KLASIFIKASI TANAMAN MENGGUNAKAN DEEP LEARNING


**ANTONIO TAIFAN MONTANA**

NRP 5025201219

Dosen Pembimbing I

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

NIP 197104281994122001

Dosen Pembimbing II

**Ir. Ary Mazharuddin Shiddiqi, S.Kom., M.Comp.Sc., Ph.D, IPM.**

NIP 198106202005011003

**Program Studi S1 Teknik Informatika**

Departemen Teknik Informatika

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya



2024

**TUGAS AKHIR – EF234801**

# PENGEMBANGAN APLIKASI BERBASIS ANDROID UNTUK KLASIFIKASI TANAMAN MENGGUNAKAN DEEP LEARNING

**ANTONIO TAIFAN MONTANA**

NRP 5025201219

Dosen Pembimbing I

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

NIP 197104281994122001

Dosen Pembimbing II

**Ir. Ary Mazharuddin Shiddiqi, S.Kom., M.Comp.Sc., Ph.D, IPM.**

NIP 198106202005011003

**Program Studi S1 Teknik Informatika**

Departemen Teknik Informatika

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya

2024

FINAL PROJECT – EF234801

# DEVELOPMENT OF AN ANDROID-BASED APPLICATION FOR FLOWER PLANT CLASSIFICATION USING DEEP LEARNING

**ANTONIO TAIFAN MONTANA**

NRP 5025201219

Advisor I

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

NIP 197104281994122001

Advisor II

**Ir. Ary Mazharuddin Shiddiqi, S.Kom., M.Comp.Sc., Ph.D, IPM.**

NIP 198106202005011003

**Study Program Bachelor of Informatics**

Department of Informatics

Faculty of Intelligent Electrical and Informatics Technology

Institut Teknologi Sepuluh Nopember

Surabaya

2024

*(Halaman ini sengaja dikosongkan)*

# LEMBAR PENGESAHAN

**PENGEMBANGAN APLIKASI BERBASIS ANDROID UNTUK KLASIFIKASI TANAMAN MENGGUNAKAN DEEP LEARNING**

**TUGAS AKHIR**

Diajukan untuk memenuhi salah satu syarat
Memperoleh gelar Sarjana Komputer pada
Program Studi S-1 Teknik Informatika
Departemen Teknik Informatika
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Oleh : **ANTONIO TAIFAN MONTANA**
NRP. 5025201219

Disetujui oleh Tim Penguji Tugas Akhir:

| | | |
|---|---|---|
| 1. | Prof. Dr. Eng. Nanik Suciati, S.Kom, M.Kom. | Pembimbing |
| 2. | Ir. Ary Mazharuddin Shiddiqi, S.Kom., M.Comp.Sc., Ph.D, IPM. | Ko-pembimbing |
| 3. | Prof. Dr.Eng. Chastine Fatichah, S.Kom., M.Kom. | Penguji I |
| 4. | Imam Mustafa Kamal, S.ST., Ph.D. | Penguji II |

**SURABAYA**
**Juli, 2024**

*(Halaman ini sengaja dikosongkan)*

# APPROVAL SHEET

**DEVELOPMENT OF AN ANDROID-BASED APPLICATION FOR FLOWER PLANT CLASSIFICATION USING DEEP LEARNING**

**FINAL PROJECT**

Submitted to fulfill one of the requirements
for obtaining a degree Bachelor of Computer Science at
Undergraduate Study Program of Informatics
Department of Informatics
Faculty of Intelligent Electrical and Informatics Technology
Institut Teknologi Sepuluh Nopember

By: **ANTONIO TAIFAN MONTANA**
NRP. 5025201219

Approved by Final Project Examiner Team:

| | | |
|---|---|---|
| 1. | Prof. Dr. Eng. Nanik Suciati, S.Kom, M.Kom. | Advisor |
| 2. | Ir. Ary Mazharuddin Shiddiqi, S.Kom., M.Comp.Sc., Ph.D, IPM. | Co-Advisor |
| 3. | Prof. Dr.Eng. Chastine Fatichah, S.Kom., M.Kom. | Examiner I |
| 4. | Imam Mustafa Kamal, S.ST., Ph.D. | Examiner II |

**SURABAYA**
**July, 2024**

*(Halaman ini sengaja dikosongkan)*

# PERNYATAAN ORISINALITAS

Yang bertandatangan di bawah ini:

| | |
|---|---|
| Nama mahasiswa / NRP | : Antonio Taifan Montana / 5025201219 |
| Departemen | : Teknik Informatika |
| Dosen Pembimbing I/ NIP | : Prof. Dr. Eng. Nanik Suciati, S.Kom, M.Kom. / 197104281994122001 |
| Dosen Pembimbing II/ NIP | : Ir. Ary Mazharuddin Shiddiqi, S.Kom., M.Comp.Sc., Ph.D, IPM. / 198106202005011003 |

Dengan ini menyatakan bahwa Tugas Akhir dengan judul "Pengembangan Aplikasi Berbasis Android untuk Klasifikasi Tanaman Menggunakan Deep Learning" adalah hasil karya sendiri, bersifat orisinal, dan ditulis dengan mengikuti kaidah penulisan ilmiah.

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

Surabaya, 24 Juli 2024

Mahasiswa

Antonio Taifan Montana

NRP. 5025201219

Mengetahui

| Dosen Pembimbing I | Dosen Pembimbing II |
|---|---|
| Prof. Dr. Eng. Nanik Suciati, S.Kom, M.Kom. | Ir. Ary Mazharuddin Shiddiqi, S.Kom., M.Comp.Sc., Ph.D, IPM. |
| NIP. 197104281994122001 | NIP. 198106202005011003 |

*(Halaman ini sengaja dikosongkan)*

# STATEMENT OF ORIGINALITY

The undersigned below:

| | |
|---|---|
| Name of student / NRP | : Antonio Taifan Montana / 5025201219 |
| Department | : Department of Informatics |
| Advisor / NIP | : Prof. Dr. Eng. Nanik Suciati, S.Kom, M.Kom. / 197104281994122001 |
| Advisor / NIP | : Ir. Ary Mazharuddin Shiddiqi, S.Kom., M.Comp.Sc., Ph.D, IPM. / 198106202005011003 |

Hereby declare that Final Project with the title of "Development of an Android-Based Application for Flower Plant Classification Using Deep Learning" is the result of my own work, is original, and is written by following the rules of scientific writing.
If in the future there is a discrepancy with this statement, then I am willing to accept sanctions in accordance with the provisions that apply at Institut Teknologi Sepuluh Nopember.

Surabaya, 24 July 2024
Student

Antonio Taifan Montana
NRP. 5025201219

Acknowledge

| Advisor I | Advisor II |
|---|---|
| Prof. Dr. Eng. Nanik Suciati, S.Kom, M.Kom. | Ir. Ary Mazharuddin Shiddiqi, S.Kom., M.Comp.Sc., Ph.D, IPM. |
| NIP. 197104281994122001 | NIP. 198106202005011003 |

*(Halaman ini sengaja dikosongkan)*

**PENGEMBANGAN APLIKASI BERBASIS ANDROID UNTUK KLASIFIKASI TANAMAN BUNGA MENGGUNAKAN DEEP LEARNING**


**Nama Mahasiswa / NRP : Antonio Taifan Montana / 5025201219**
**Departemen : Teknik Informatika FTEIC - ITS**
**Dosen Pembimbing 1 : Prof. Dr. Eng. Nanik Suciati, S.Kom, M.Kom**
**Dosen Pembimbing 2 : Ir. Ary Mazharuddin Shiddiqi, S.Kom., M.Comp.Sc., Ph.D, IPM.**



**Abstrak**

Bunga mempercantik lingkungan disekitarnya dengan keindahannya yang beragam. Selain itu, bunga juga dapat mempercantik lingkungan di sekitarnya dengan berbagai bentuk unik yang beragam. Sampai saat ini terdapat sekitar 390 ribu spesies tanaman bunga di dunia. Bunga dapat memberi makan serangga, burung, hewan dan manusia. Bunga juga digunakan sebagai obat untuk manusia dan juga hewan. Dari banyaknya jenis tanaman yang ada, hal ini menimbulkan permasalahan untuk mengetahui jenis tanaman hanya dengan berdasarkan bunganya. Oleh karena itu pentingnya membuat sistem yang dapat mengklasifikasikan jenis tanaman tersebut dengan mudah dan praktis. Deep Learning merupakan salah satu metodologi yang dapat menyelesaikan permasalahan klasifikasi tersebut. Pada penelitian ini, akan dilakukan pemodelan sistem klasifikasi dengan menggunakan metode Deep Learning. Hasil pemodelan dari sistem yang sudah buat kemudian akan diimplementasikan pada aplikasi berbasis Android sebagai sistem informasi. Pemodelan dibangun dengan membandingan beberapa arsitektur yang ringan yaitu: MobileNetV2, InceptionV3, Xception. Dataset yang digunakan untuk membangun model ialah Oxford 102-Flowers. Setelah mendapatkan model terbaik berdasarkan nilai loss dan akurasi, serta aplikasi selesai dibangun, akan dilakukan uji coba model dengan menggunakan data publik dan uji coba aplikasi. Hasil dari pengujian *pre-trained* model menunjukkan bahwa model terbaik untuk klasifikasi bunga yang diterapkan pada perangkat bergerak Android adalah Xception, dengan akurasi sebesar 82%. Lalu diikuti oleh MobileNetV2 dengan akurasi sebesar 78% dan yang terakhir InceptionV3 dengan akurasi sebesar 75%.



**Kata kunci: Klasifikasi, Deep Learning, Tanaman Bunga, Perangkat Bergerak.**

*(Halaman ini sengaja dikosongkan)*

# DEVELOPMENT OF AN ANDROID-BASED APPLICATION FOR FLOWER PLANT CLASSIFICATION USING DEEP LEARNING

**Student Name / NRP : Antonio Taifan Montana / 5025201219**
**Department : Informatics Engineering FTEIC - ITS**
**Advisor 1 : Prof. Dr. Eng. Nanik Suciati, S.Kom, M.Kom**
**Advisor 2 : Ir. Ary Mazharuddin Shiddiqi, S.Kom., M.Comp.Sc., Ph.D, IPM.**

**Abstract**

Flowers beautify the surrounding environment with their diverse beauty. In addition, flowers can also beautify the surrounding environment with a variety of unique shapes. To date, there are about 390 thousand species of flower plants in the world. Flowers can feed insects, birds, animals and humans. Flowers are also used as medicine for humans and animals. Of the many types of plants that exist, this raises the problem of knowing the type of plant based only on its flowers. Therefore, it is important to create a system that can classify the type of plant easily and practically. Deep Learning is one of the methodologies that can solve the classification problem. In this research, a classification system will be modeled using the Deep Learning method. The modeling results of the system that has been made will then be implemented in an Android-based application as an information system. Modeling is built by comparing several lightweight architectures, namely: MobileNetV2, InceptionV3, Xception. The dataset used to build the model is Oxford 102-Flowers. After getting the best model based on loss and accuracy values, and the application is completed, a model test will be carried out using public data and application trials. The results of testing pre-trained models show that the best model for flower classification applied to Android mobile devices is Xception, with an accuracy of 82%. Then followed by MobileNetV2 with an accuracy of 78% and finally InceptionV3 with an accuracy of 75%.

**Kata kunci: Classification, Deep Learning, Flower, Mobile-Based Application.**

*(Halaman ini sengaja dikosongkan)*

# KATA PENGANTAR

Puji syukur penulis panjatkan kepada Tuhan Yang Maha Esa atas berkat dan rahmat serta karunia-Nya sehingga penulis dapat menyelesaikan pengerjaan tugas akhir yang berjudul "Pengembangan Aplikasi Berbasis Android untuk Klasifikasi Tanaman Menggunakan Deep Learning". Penulisan tugas akhir ini tidak dapat terselesaikan tanpa dukungan dan bantuan dari berbagai pihak yang berarti. Oleh karena itu, penulis ingin mengucapkan terima kasih sebesar-besarnya kepada seluruh pihak yang telah membantu dan mendukung penulis dalam menyelesaikan tugas akhir ini, terutama kepada:

1. Tuhan Yang Maha Esa, atas berkat dan rahmat serta karunia-Nya sehingga penulis dapat menyelesaikan tugas akhir ini di Departemen Teknik Informatika, Institut Teknologi Sepuluh Nopember.
2. Kedua orang tua penulis yang telah mendukung secara fisik dan mental kepada penulis selama perkuliahan di Departemen Teknik Informatika, Institut Teknologi Sepuluh Nopember.
3. Ibu Prof. Dr. Eng. Nanik Suciati, S.Kom, M.Kom. dan Bapak Ir. Ary Mazharuddin Shiddiqi, S.Kom., M.Comp.Sc., Ph.D, IPM. sebagai dosen pembimbing yang telah memberi bimbingan dan arahan sepanjang penyelesaian tugas akhir ini.
4. Bapak dan Ibu dosen di Departemen Teknik Informatika, Institut Teknologi Sepuluh Nopember yang telah memberikan ilmu, pengetahuan, dan pengalaman selama masa perkuliahan.
5. Hafizh, Ariq, Pael, Pred, Cahyadi, Afan, Afdal, Barhan, Meisya, Nadya, dan teman-teman mahasiswa Informatika ITS Angkatan 2020 yang tidak bisa saya sebutkan namanya satu-persatu yang mana memberikan dukungan semangat dalam pengerjaan tugas akhir kepada penulis.
6. Jihan, Adika, Fakhri, Hany, Ghozi, Fathir, Dr. Lambert, Andre, Steven, Orchidta, Hio, Ms. Ellis, Ms. Aul, Intan, Puteri, Sabrina, Frisca, Rakhananta selaku rekan yang mana memberikan dukungan semangat dalam pengerjaan tugas akhir kepada penulis.
7. Pihak-pihak lain yang tidak dapat disebutkan satu persatu yang telah membantu penulis dalam penyusunan tugas akhir ini.
8. Peneliti persembahkan skripsi ini spesial untuk orang yang selalu bertanya "kapan kamu Wisuda?" dan "emang bisa Wisuda tahun ini". Wisuda hanyalah bentuk seremonial akhir setelah melewati beberapa proses, terlambat lulus atau tidak lulus tepat waktu bukanlah suatu kejahatan dan bukanlah sebuah aib. Alangkah kerdilnya jika kecerdasan seseorang diukur dari siapa yang paling cepat wisuda. Bukankah sebaik-baiknya skripsi adalah skripsi yang diselesaikan. Entah itu tepat waktu maupun tidak.

Penulis menyadari bahwa Tugas Akhir ini masih jauh dari kata sempurna, tetapi penulis berharap dengan penyusunan tugas akhir ini dapat memberikan kontribusi yang bermanfaat bagi pengembangan ilmu pengetahuan. Akhir kata, penulis menyampaikan terima kasih dan permohonan maaf atas segala kesalahan yang terdapat pada tugas akhir ini. Penulis sangat menghargai kritik dan saran dari berbagai pihak untuk perbaikan di masa mendatang.

*(Halaman ini sengaja dikosongkan)*

# DAFTAR ISI

*(Halaman ini sengaja dikosongkan)*

# DAFTAR GAMBAR

*(Halaman ini sengaja dikosongkan)*

# DAFTAR TABEL

*(Halaman ini sengaja dikosongkan)*

# DAFTAR KODE SUMBER

*(Halaman ini sengaja dikosongkan)*

# DAFTAR SINGKATAN

| | | |
|---|---|---|
| AI | : | *Artificial Intelligence* |
| CNN | : | *Convolutional Neural Network* |
| GAP | : | *Global Average Pooling* |
| IDE | : | *Integrated Development Environment* |
| ML | : | *Machine Learning* |
| MLP | : | *Multilayer Perceptron* |
| RGB | : | *Red, Green, Blue* |

*(Halaman ini sengaja dikosongkan)*

# BAB I PENDAHULUAN

## 1.1 Latar belakang

Bunga merupakan salah satu bagian tanaman yang dapat memikat orang yang melihatnya dengan keindahannya yang terlihat secara kasat mata. Selain itu, bunga juga dapat mempercantik lingkungan di sekitarnya dengan berbagai bentuk unik yang beragam. Sampai saat ini terdapat sekitar 390 ribu spesies tanaman bunga di dunia[1]. Bunga dapat memberi makan serangga, burung, hewan dan manusia. Bunga juga digunakan sebagai obat untuk manusia dan juga hewan [2].

Dibutuhkan pengetahuan tentang bunga yang baik untuk mengetahui spesiesnya. Jika tidak, banyak tanaman yang mungkin akan rusak karena dianggap hama dan berbahaya bagi tanaman lain atau lingkungan di sekitarnya. Bahkan, mungkin saja dijual dengan harga yang sangat murah [3]. Dan semua ini terjadi karena pengenalan yang tidak memadai terhadap spesies tanaman. Sampai saat ini masyarakat awam menghadapi kesulitan dalam mengklasifikasi spesies bunga [2]. Tidak lain disebabkan oleh adanya kemiripan di berbagai jenis tanaman dan juga keterbatasan kemampuan manusia dalam mengingat dan mengklasifikasi jenis tanaman bunga. Hal ini memunculkan permasalahan untuk mengklasifikasi tanaman bunga. Selain proses klasifikasi secara manual dapat memakan waktu dan sumber daya yang banyak serta rentan untuk melakukan kesalahan [4].

Dalam upaya menyelesaikan permasalahan klasifikasi tanaman bunga, telah dikembangkan berbagai teknologi seperti *Image Processing* dan *Deep Learning* yang memberikan kinerja yang lebih dari fungsi mata manusia dan mendapatkan hasil yang akurat [2]. Dari berbagai algoritma model *Deep Learning* yang sedang popular, *Convolutional Neural Network* (CNN) merupakan salah satu algoritma yang mapan dan sering kali digunakan untuk mengklasifikasikan tanaman bunga [5].

Di era teknologi yang terus berkembang, penggunaan *smartphone* telah menjadi suatu kebutuhan utama dalam kehidupan sehari-hari. Keberadaan *smartphone* memberikan kemudahan dalam berbagai aspek kehidupan, termasuk dalam mengenali tanaman bunga. Dengan adanya sistem operasi yang bersifat open source, seperti Android, pengguna memiliki akses ke aplikasi berbasis teknologi pengolahan citra digital yang memungkinkan mereka mengidentifikasi bunga secara langsung melalui perangkat mereka. Kelebihan ini mendukung pemanfaatan teknologi *Deep Learning* dan *Convolutional Neural Network* (CNN) dalam mengklasifikasikan tanaman bunga, menjadikan proses identifikasi lebih sederhana dan dapat diakses oleh masyarakat umum [6].

Pada penelitian ini difokuskan untuk pengembangan aplikasi klasifikasi tanaman berdasarkan citra bunga menggunakan *Deep Learning* pada perangkat berbasis Android untuk mempermudah proses klasifikasi tanaman bunga. Dengan menggunakan aplikasi ini, diharapkan klasifikasi tanaman bunga dapat dilakukan dengan lebih efisien dan akurat, serta menghemat waktu dan sumber daya yang diperlukan. Aplikasi ini juga dapat membantu masyarakat umum untuk lebih mengenal berbagai jenis tanaman bunga dengan cepat dan mudah.

## 1.2 Rumusan Permasalahan

Berdasarkan latar belakang tersebut, permasalahan utama yang akan diangkat dalam penelitian ini, yaitu :

1. Bagaimana merancang model klasifikasi tanaman bunga dengan menggunakan *deep learning*?
2. Bagaimana menerapkan aplikasi klasifikasi tanaman bunga menggunakan *deep learning* pada perangkat berbasis Android?

3. Bagaimana evaluasi kinerja aplikasi untuk klasifikasi jenis tanaman bunga pada data *real-time*?

## 1.3 Batasan Masalah

Penelitian ini disusun berdasarkan data-data yang diperoleh. Luasnya bidang yang dihadapi oleh peneliti membuat penelitian ini harus dibatasi. Ruang lingkup masalah yang menjadi batasan pada penelitian ini adalah:

1. Sistem klasifikasi tanaman bunga berupa aplikasi pada perangkat berbasis Android.
2. Sistem ini dirancang untuk mengidentifikasi seratus dua label atau kelas dari tanaman melalui pengenalan citra bunga. Data citra diambil dari 102 *Category Flower Dataset* berjumlah 8190 citra yang merupakan citra berwarna RGB, memiliki ukuran yang berbeda-beda, dan tidak memiliki *noise*.
3. Model dari *deep learning* dibuat dengan menggunakan bahasa pemrograman Python 3.11.0.
4. Performa saat pencarian model terbaik akan dilihat dari *training accuracy* dan *validation accuracy*.
5. Implementasi aplikasi pada perangkat berbasis Android akan menggunakan bahasa pemrograman Kotlin.

## 1.4 Tujuan

Tujuan tugas akhir ini adalah untuk mengembangan aplikasi berbasis Android untuk klasifikasi tanaman bunga menggunakan metode *Deep Learning* menggunakan *Convolutional Neural Network*. Hasil pemodelan tersebut kemudian digunakan untuk mengembangkan aplikasi yang berfungsi sebagai sistem klasifikasi tanaman bunga dengan efisien dan akurat. Model CNN yang digunakan dibangun dengan menggunakan arsitektur MobiletNetV2, InceptionV3, Xception dengan mempertimbangan *resource* komputasi yang terbatas pada perangkat Android.

## 1.5 Manfaat

Adapun manfaat dalam penelitian ini adalah :

1. Penekanan pada kemudahan dan efisiensi bagi pengguna dalam mengidentifikasi jenis tanaman bunga.
2. Penelitian ini diharapkan meningkatkan pengetahuan masyarakat umum mengenai berbagai jenis tanaman bunga.
3. Penelitian ini dapat memberikan informasi kepada mahasiswa tentang cara mengimplementasikan dan menerapkan metode *Deep Learning* menggunakan *Convolutional Neural Network* yang dilakukan dan menjadi acuan khususnya bagi mahasiswa yang dapat membantu mahasiswa dalam proses pembelajaran dan penelitian mahasiswa di masa yang akan datang.

# BAB II TINJAUAN PUSTAKA

## 2.1 Penelitian Terkait

Dalam penyusunan Tugas Akhir, terdapat beberapa penelitian terkait yang menjadi inspirasi dan referensi. Tabel 2.1 menyajikan perbandingan dari penelitian-penelitian yang relevan dengan penyusunan Tugas Akhir.

**Tabel 2.1** Perbandingan Penelitian Terkait

| Nama Peneliti | Judul | Metode | Dataset | Hasil |
|---|---|---|---|---|
| Yong Wu, Xiao Qin, Yonghua Pan, dan Changan Yuan | *Convolution Neural Network based Transfer Learning for Classification of Flowers* | *Transfer learning* VGG16, VGG19, InceptionV3, dan ResNet50 | Oxford-17 flowers dan Oxford-102 flowers | Akurasi ResNet50 95.29%, Akurasi InceptionV3 94.58%, Akurasi VGG19 84.71%, Akurasi VGG16 83.53% |
| Chhaya Narvekar, Madhuri Rao | *Flower classification using CNN and transfer learning in CNN-Agriculture Perspective* | *Transfer learning* VGG16, Resnet50, MobileNetV2 | Bunga dari Kaggle berjumlah 4323 data dan 5 label | Akurasi ResNet50 92.12%, Akurasi VGG16 89.35%, dan Akurasi MobileNetV2 71.75% |
| Neda Alipour, Omid Tarkhaneh, Mohammad Awrangjeb, dan Hongda Tian | *Flower Image Classification Using Deep Convolutional Neural Network* | DenseNet121 | Oxford-102 flowers | Akurasi 98.6% |
| Büşra Rümeysa Mete dan Tolga Ensari | *Flower Classification with Deep CNN and Machine Learning Algorithms* | *Feature Extraction* InceptionV3 dengan MLP Classifier | Oxford-102 flowers | Akurasi 99.8% |
| Christian A. Elinisa dan Neema Mduma | *Mobile-Based convolutional neural network model for the early identification of banana diseases* | CNN yang sudah di-*convert* ke TFLite | 27.360 data dengan 4 label dari peternakan di Tanzania. | Akurasi 91.17% |

Berikut adalah penelitian-penelitian yang terkait dengan penyusunan Tugas Akhir ini :

1. *Convolution Neural Network based Transfer Learning for Classification of Flowers*

Penelitian ini melakukan klasifikasi jenis bunga menggunakan *transfer learning* model pada CNN. Dataset yang digunakan adalah Dataset Oxford-17 yang terdiri dari 17 jenis bunga masing-masing terdapat 80 data dan Dataset Oxford-102 yang terdiri dari 102 jenis bunga

masing-masing terdapat 40-258 data. Penelitian ini menggunakan model *transfer learning* VGG16, VGG19, InceptionV3, dan ResNet50 untuk melakukan perbandingan akurasi pada klasifikasi jenis bunga. Hasil akurasi menggunakan InceptionV3-transfer dan ResNet50 transfer lebih besar daripada VGG-16-transfer dan VGG-19, yaitu 95,88% dan 96,57% [5].

2. *Flower classification using CNN and transfer learning in CNN- Agriculture Perspective*

Dalam penelitian ini, peneliti membangun model *pre-trained* CNN dengan menggunakan *transfer learning* untuk menyelesaikan permasalahan klasifikasi tanaman agrikultur dengan menggunakan dataset dari Kaggle dengan total citra 4323 dan 5 label. Peneliti menggunakan arsitektur *pre-trained* CNN yang berukuran ringan yaitu VGG16, Resnet50, MobileNetV2. Peneliti mengembangkan model ringan berukuran kecil agar dapat dikembangkan dengan baik pada perangkat bergerak yang memiliki *resource* komputasi yang terbatas. Model yang telah dibuat oleh peneliti mendapatkan akurasi sebesar 89.35% untuk VGG16, 92.12% untuk Resnet50, 71.75 untuk MobileNetV2 [2].

3. *Flower Image Classification Using Deep Convolutional Neural Network*

Penelitian serupa lainnya dengan judul "*Flower Image Classification Using Deep Convolutional Neural Network*". Penelitian ini menggunakan dataset dari Oxford-102 flower yang terdiri dari 102 jenis bunga dengan jumlah data 8189. Peneliti menggunakan pendekatan *transfer learning* dengan arsitektur DenseNet121. Peneliti membagi dataset menjadi 3 yaitu pelatihan, validasi, dan uji coba dengan perbandingan 80% pelatihan, 10% validasi, dan 10% uji coba. Model mendapatkan akurasi sebesar 98.6% [7].

4. *Flower Classification with Deep CNN and Machine Learning Algorithms*

Penelitian mengklasifikasikan bunga dengan menggunakan pendekatan *Deep* CNN dan *machine learning*. Arsitektur *Deep* CNN yang digunakan adalah InceptionV3. *Deep* CNN digunakan untuk melakukan ekstraksi fitur dan untuk mendapatkan hasil yang lebih baik, mereka menunjukkan pemanfaatan augmentasi gambar. Peneliti membandingkan beberapa pendekatan m*achine learning*. Mereka telah menggunakan dua dataset *Oxford-17 Flowers* dan *Oxford 102-Flowers*. Mereka telah mencapai akurasi tertinggi 98,5% dan 99,8% dengan menggunakan pengklasifikasi SVM dan MLP [3].

5. *Mobile-Based convolutional neural network model for the early identification of banana diseases*

Penelitian ini bertujuan untuk menerapkan model *deep learning* dalam aplikasi *mobile* untuk identifikasi awal penyakit Layu Fusarium dan Sigatoka Hitam pada pisang. Model CNN digunakan untuk klasifikasi penyakit pisang Black Sigatoka dan penyakit Layu Fusarium. Satu set data yang terdiri dari 27.360 gambar daun dan batang pisang yang sakit dan sehat yang dikumpulkan dari perkebunan menggunakan kamera seluler berfungsi sebagai data pelatihan untuk model ini. Model CNN mencapai akurasi 91,17% dan digunakan dalam aplikasi seluler untuk klasifikasi penyakit. Penelitian ini menunjukkan bahwa *deep learning* dapat diimplementasikan dan membantu dalam identifikasi awal penyakit pisang. Aplikasi ini dapat mendeteksi gambar daun dan batang pisang yang sehat dan sakit serta gambar yang bukan tanaman pisang dengan nilai kepercayaan lebih dari 90% dalam waktu kurang dari lima detik per gambar [8].

## 2.2 Dasar Teori

### *2.2.1 Machine Learning*

*Machine Learning* (ML) merupakan cabang dari *Artificial Intelligence* (kecerdasan

buatan) yang memungkinkan sistem komputer untuk belajar langsung dari contoh, data, dan pengalaman yang diperolehnya. Dengan kemampuan membuat komputer menjalankan tugas secara cerdas, ML dapat melibatkan algoritma kompleks dengan mempelajari data yang diberikan tanpa perlu mengikuti program yang dibuat atau diinstruksikan secara manual oleh pengguna [9].

Ciri khas dari ML adalah adanya proses pembelajaran atau *training*. Oleh karena itu, ML membutuhkan data *training* untuk dipelajari. Setelah berhasil melakukan *training*, ML dapat melakukan proses klasifikasi dan prediksi terhadap data baru sesuai dengan hasil *training* yang telah dilakukan. Klasifikasi adalah metode dalam ML yang digunakan oleh mesin untuk memilah atau mengklasifikasikan objek berdasarkan ciri tertentu, seperti manusia membedakan benda satu dengan yang lain. Sementara itu, prediksi atau regresi digunakan oleh mesin untuk menerka keluaran dari suatu data masukan [10]. Beberapa contoh implementasi dari ML antara lain adalah *image processing, search recommendation, text,* dan *speech recognition,* dan *health diagnosis*

### *2.2.2 Deep Learning*

*Deep learning* adalah bagian dari *machine learning* yang difokuskan pada pemanfaatan arsitektur jaringan saraf tiruan yang mendalam, atau *deep neural networks,* untuk mengekstrak fitur dari data. Konsep utama dalam *deep learning* adalah penggunaan jaringan saraf tiruan dengan banyak lapisan (*layer*), memungkinkannya memahami dan merepresentasikan struktur yang semakin kompleks dari data masukan. Ini merupakan perbedaan utama dengan *machine learning* konvensional yang memiliki keterbatasan dalam memproses data alami secara langsung. Selama bertahun-tahun, pembuatan sistem pengenalan pola atau pembelajaran mesin membutuhkan keterampilan teknis dan pemahaman yang mendalam di bidangnya untuk merancang alat ekstraksi fitur. Alat ini mengubah data mentah, seperti nilai piksel dalam gambar, menjadi representasi internal atau vektor fitur yang sesuai [11].

*Deep learning* telah mencapai kemajuan signifikan dalam menyelesaikan masalah yang sebelumnya sulit dipecahkan oleh komunitas kecerdasan buatan. Teknik ini sangat efektif dalam menemukan pola rumit dalam data yang memiliki banyak dimensi, dan dapat diterapkan dalam berbagai bidang ilmu pengetahuan, bisnis, dan pemerintahan. Bahkan, *deep learning* dapat memberikan hasil yang sangat baik dalam tugas-tugas tertentu dalam pemahaman bahasa alami, seperti klasifikasi topik, analisis sentimen, pertanyaan dan jawaban, serta terjemahan bahasa [11].

### *2.2.3 Image Processing*

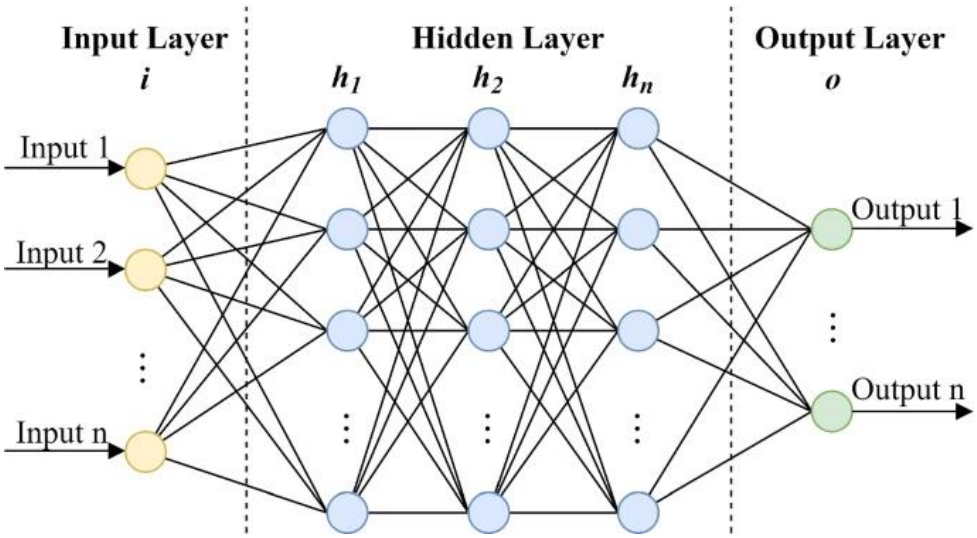
*Image Processing* adalah teknologi modern yang mengubah informasi gambar menjadi informasi digital, sehingga dapat dikenali dan diproses dengan lebih baik oleh komputer. Pemrosesan gambar secara kasar dapat dibagi menjadi serangkaian proses seperti klasifikasi gambar, kompresi, peningkatan, pengkodean, dan ekstraksi fitur. Untuk memastikan bahwa gambar cukup jelas dan mengidentifikasi informasi secara efektif, pemrosesan gambar harus didukung oleh berbagai teknologi untuk meningkatkan resolusi dan kualitas gambar. Perkembangan teknologi kecerdasan buatan juga mendorong perkembangan teknologi pemrosesan gambar, yang membuat teknologi pemrosesan gambar banyak digunakan di berbagai bidang seperti pengenalan pola, visi mesin, dan teknologi multimedia [12].

### 2.2.4 *Convolutional Neural Network* (CNN)

*Convolutional Neural Network* (CNN) merupakan perkembangan *dari Multilayer Perceptron* (MLP) yang dirancang khusus untuk memproses data dua dimensi. CNN termasuk dalam kategori *Deep Neural Network* karena memiliki kedalaman yang signifikan dan sering

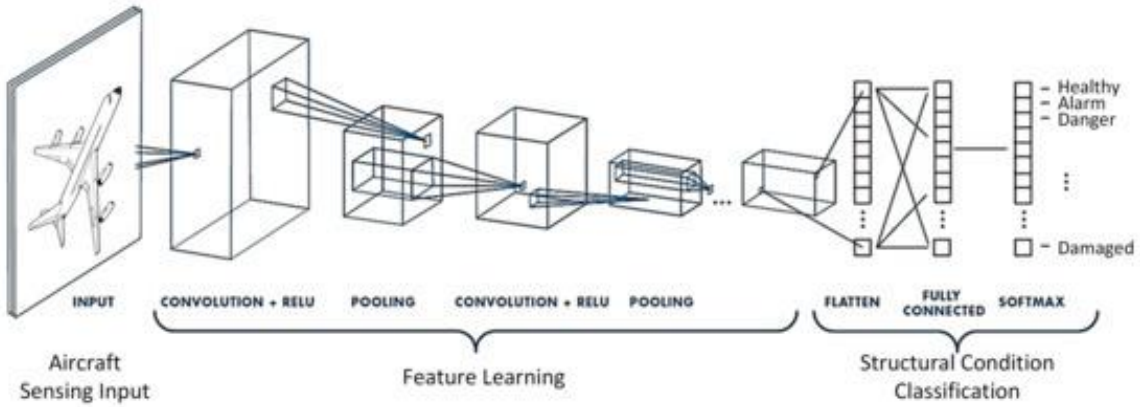
digunakan dalam konteks pengolahan data citra. Ketika digunakan untuk klasifikasi citra, MLP kurang cocok karena tidak dapat menyimpan informasi spasial dari citra dan menganggap setiap piksel sebagai fitur independen, mengakibatkan hasil yang kurang memuaskan [13].

CNN terbentuk oleh berlapis-lapis neuron yang memiliki berat (*weight*) dan bisa yang dapat diatur. Arsitektur *neural network* sendiri terdiri dari 2 atau lebih dari 2 *hidden layer*. Setiap neuron menerima input dari *layer* input, melakukan operasi *dot-product* pada *layer* berikutnya, dan menghasilkan *output* pada layer *output*. Pada umumnya, peningkatan jumlah *layer* dalam jaringan saraf dapat meningkatkan akurasi dan kompleksitas kemampuan jaringan. Namun, ada suatu titik di mana penambahan lapisan tidak lagi memberikan peningkatan kinerja yang signifikan [14]. *Network* ini dilatih untuk mengolah data mentah dan menemukan korelasi antara data tersebut dengan skor akhir. Semakin tinggi skor akhir, semakin cenderung jaringan untuk memprioritaskan konfigurasi tertentu yang menghasilkan skor tersebut. Arsitektur *neural network* diilustrasikan pada Gambar 2.1




**Gambar 2.1** Ilustrasi arsitektur *neural network* [14]

Secara teknis, CNN adalah arsitektur yang dapat di-*training* dan terdiri dari beberapa tahap. Input dan *output* dari setiap tahap berupa *array* yang disebut *feature map* atau peta fitur. *Output* dari setiap tahap merupakan *feature map* hasil pengolahan dari semua lokasi pada input. Struktur CNN dibangun dari tiga jenis *layer* utama, yaitu *Convolution Layer*, *Pooling Layer*, dan *Fully-Connected Layer* [15]. Ilustrasi alur kerja *layer* CNN dapat dilihat pada Gambar 2.2.




**Gambar 2.2** Ilustrasi alur kerja CNN [16]

Pada *layer* pertama *Convolution Layer*, terdapat filter berukuran kecil yang

dikonvolusikan dengan gambar (direpresentasikan dalam bentuk matriks). Ketika filter berada di atas piksel tertentu, hasil konvolusi menghasilkan matriks citra. Pada lapisan kedua, *Pooling Layer* digunakan untuk melakukan *down-sampling* pada data, mengurangi beban komputasi dalam memproses matriks. Sementara itu, *Fully-Connected Layer* pada lapisan ketiga memiliki koneksi penuh dengan lapisan-lapisan sebelumnya, mirip dengan struktur pada *neural network* biasanya.

### 2.2.5 *Layer-layer* pada CNN

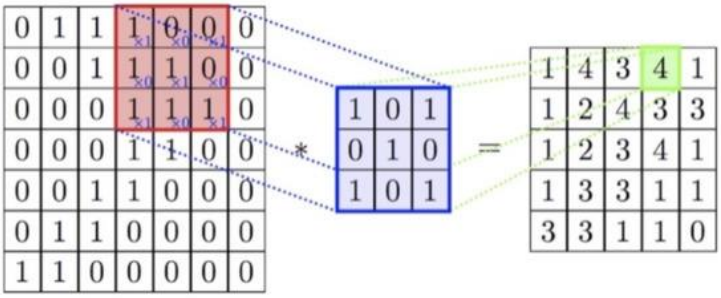
Pada CNN, terdapat beberapa *layer* yang digunakan, yaitu :

1. *Convolution Layer*

   *Layer* konvolusi merupakan tulang belakang dari CNN. Pada *layer* ini, data citra diambil, dan sebuah *kernel* atau filter diterapkan dengan pergeseran di atas citra. *Kernel* ini kemudian melakukan ekstraksi fitur melalui operasi *dot-product*. Hasil *dot-product* dapat disalurkan ke *layer* berikutnya. Tiap *kernel* dalam *layer* konvolusi memiliki *weight* seukuran *kernel*, yang dapat diubah melalui proses *backpropagation*.

   Satu *layer* konvolusi dapat memiliki lebih dari satu jenis *kernel*. Setiap *kernel* memiliki *weight* yang berbeda dan mengekstraksi fitur yang berbeda pula. Keseluruhan *kernel* ini membentuk citra baru melalui hasil perkalian *dot-product*, dengan kedalaman *channel* sesuai dengan jumlah *kernel* [16].

   *Layer* konvolusi menerima input dengan ukuran spasial 2-dimensi (*matrix*), sementara *layer fully-connected* hanya menerima input dengan ukuran spasial 1-dimensi (*array*). Ilustrasi cara kerja *filter* pada *layer* konvolusi dapat dilihat pada Gambar 2.3.



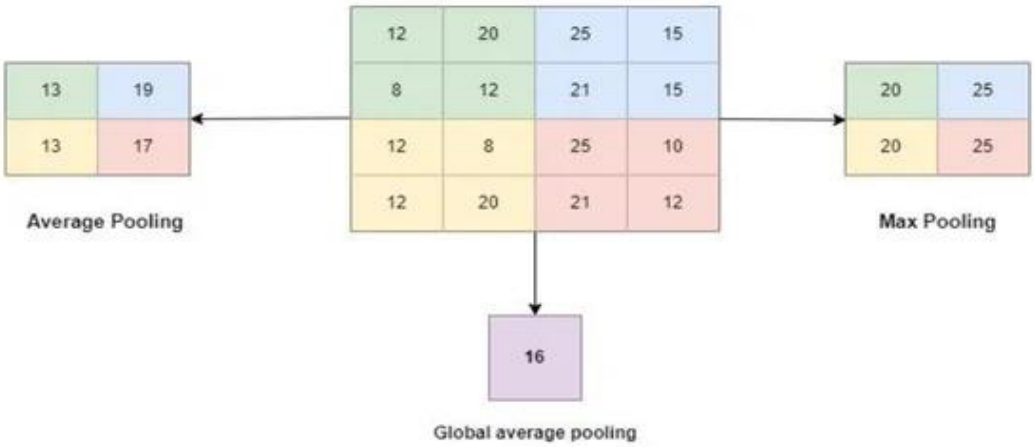
**Gambar 2.3** Filter pada *Convolution layer* [17]

2. *Pooling Layer*

   Lapisan *pooling*, juga dikenal sebagai lapisan *down-sampling*, digunakan untuk mengurangi dimensi matriks sambil mempertahankan data yang paling penting. Sebuah filter menerapkan operasi *pooling* pada data input dengan menggesernya di lapisan *pooling*.

   Bagian penting dari *pooling*, yang digunakan untuk mengurangi kompleksitas lapisan atas, adalah *down-sampling*. Dalam hal pemrosesan gambar, ini dapat dibandingkan dengan mengurangi resolusi. Jumlah filter tidak terpengaruh oleh *pooling*. *Max-pooling* adalah salah satu metode *pooling* yang paling sering digunakan. Gambar dibagi menjadi *sub-region* persegi panjang, dan hanya nilai terbesar yang ditemukan di dalam setiap *sub-region* yang dikembalikan. Salah satu ukuran *max-pooling* yang paling umum adalah 2 × 2.

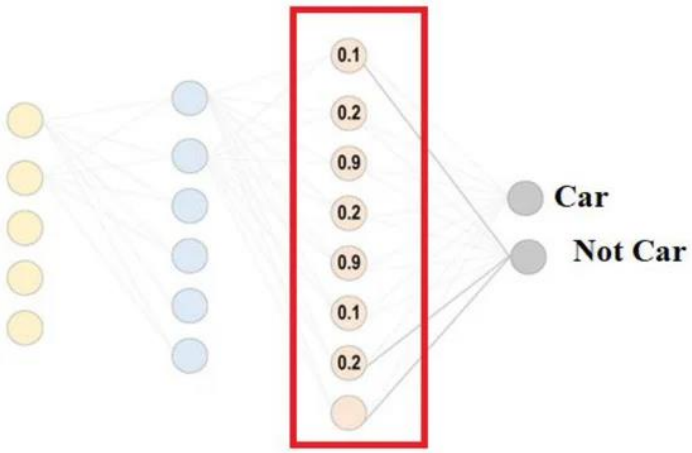
   Pada berbagai tingkat *pooling*, berbagai teknik *pooling* dapat diterapkan. Beberapa metode tersebut termasuk *global average pooling* (GAP), *global max pooling*, *average pooling*, *min pooling*, dan *gated pooling*. Gambar 2.4 menggambarkan masing-masing dari tiga teknik *pooling* ini. [18]




**Gambar 2.4** *Layer pooling* [18]

3. *Fully-Connected Layer*

*Fully-Connected Layer* pada CNN pada dasarnya mirip *dengan Fully-Connected* yang terdapat pada *neural network* lainnya. *Fully-Connected* dan *Convolution layer* memiliki fungsi yang serupa dalam proses *training* dan penyimpanan *weight* dari model. Namun, karena *Fully-Connected Layer* hanya mendukung data dengan 1 dimensi spasial, sebaiknya *Fully-Connected Layer* hanya diterapkan di akhir arsitektur. Hal ini dilakukan untuk mencegah kehilangan informasi data akibat mengubah data gambar 2-dimensi spasial menjadi data 1-dimensi spasial [18]. Gambar 2.5 menggambarkan *Fully-Connected Layer*.




**Gambar 2.5** *Fully-connected layer* [18]

### 2.2.6 Fungsi Aktivasi

Fungsi aktivasi adalah suatu mekanisme yang mengambil nilai input dari suatu neuron dan menghasilkan nilai *output* yang selanjutnya diteruskan ke neuron lain. Fungsi ini menambahkan properti nonlinier pada algoritma, yang penting untuk menangani permasalahan kompleks [16]. Keberadaan sifat nonlinier ini krusial agar algoritma dapat berhasil menyelesaikan tugas yang rumit. Tanpa properti ini, algoritma cenderung menghasilkan akurasi yang rendah, dan model CNN menjadi kurang fleksibel, mirip dengan perilaku *single-layer perceptron* [3]. Beberapa fungsi aktivasi yang umum digunakan dalam CNN termasuk:

*a. Sigmoid*

Fungsi aktivasi *sigmoid* memiliki bentuk seperti pada persamaan 2.1.

$$\sigma(x) = \frac{1}{(1 + e^{-x})} \quad (2.1)$$

Fungsi *sigmoid* adalah fungsi aktivasi yang memiliki rentang antara 0 dan 1. Fungsi ini memiliki bentuk kurva S yang mirip dengan grafik fungsi logaritma [16]. Fungsi *sigmoid* mudah dimengerti dan diterapkan, namun sudah jarang digunakan karena beberapa alasan, yaitu:

1. Titik tengah kurva dari fungsi tidak tepat di 0 sehingga *output* tidak konsisten dan bisa bergeser terlalu jauh ke atas atau ke bawah. Ini mengakibatkan fungsi *sigmoid* sulit dioptimasi.
2. Gradien data hilang (*Vanishing Gradient Problem*)
3. Konvergensi yang lambat

b. Tanh *(Hyperbolic Tangent)*

Fungsi Tanh merupakan peningkatan dari fungsi sigmoid. Fungsi Tanh memiliki titik tengah 0, sehingga menyelesaikan salah satu permasalahan fungsi *sigmoid* yaitu di permasalahan optimasi fungsi aktivasi [16]. Fungsi aktivasi Tanh memiliki bentuk seperti pada persamaan 2.2.

$$\sigma(x) = tanh(x) \quad (2.2)$$

Fungsi aktivasi Tanh lebih mudah dioptimasi, namun memiliki kekurangan yaitu permasalahan gradien tersaturasi dan *vanishing gradient*. Oleh karena itu, fungsi Tanh jarang digunakan.

c. ReLU *(Rectified Linear Unit)*

Fungsi ReLU merupakan singkatan dari *rectified liner unit* dan merupakan fungsi aktivasi nonlinier yang banyak digunakan di jaringan saraf tiruan. Fungsi aktivasi ReLU memiliki bentuk seperti pada persamaan 2.3.

$$\sigma(x) = \max(0, x) \quad (2.3)$$

Dikarenakan fungsinya yang sangat sederhana, fungsi ReLU memiliki performa yang lebih tinggi dibandingkan fungsi aktivasi lainnya. Selain itu ada beberapa alasan mengapa ReLU sering kali digunakan, yaitu:

1. *Cost-effective*
2. Optimasi yang mudah
3. Konvergensi data yang cepat
4. Pergerakan data stabil, tidak mengalami permasalahan gradien hilang

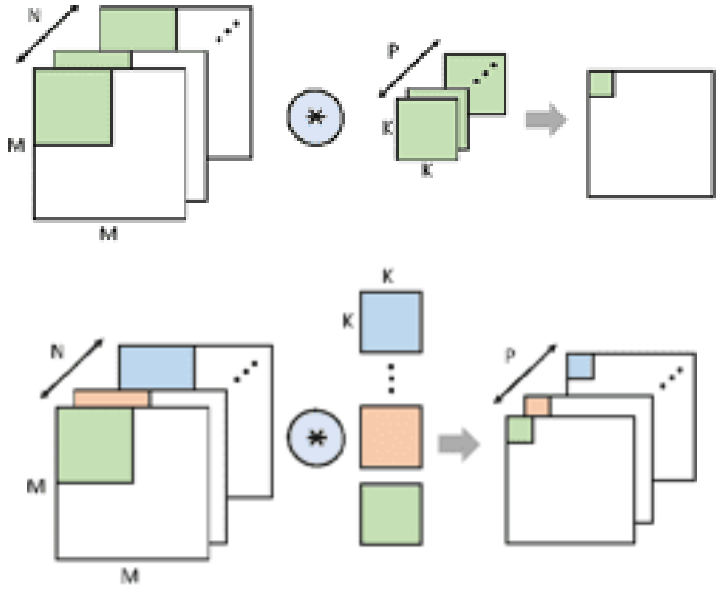
*d. Softmax*

Fungsi aktivasi *softmax* adalah fungsi yang digunakan untuk menghitung probabilitas kelas target dari hasil *output* sebuah *neural network*. Fungsi *softmax* menghasilkan *output* yang merupakan kisaran nilai antara 0 dan 1, dengan jumlah probabilitasnya sama dengan 1. Fungsi *softmax* umumnya digunakan pada layer terakhir dari sebuah *neural network*, yang merupakan layer yang menghasilkan output akhir dari model. Fungsi softmax digunakan untuk menghitung probabilitas dari setiap kelas target, sehingga model dapat menentukan kelas target dari input yang diberikan[16]. Fungsi aktivasi *softmax* memiliki bentuk seperti pada persamaan 2.4.

$$\sigma(Xi) = \frac{e^{Xi}}{\sum_{j=0}^{k} e^{Xj}} \quad (2.4)$$

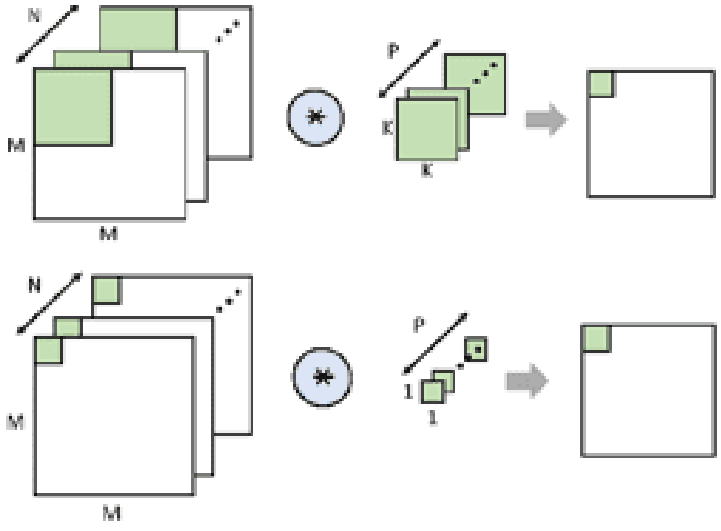
### 2.2.7 MobileNetV2

MobileNetV2 merupakan pengembangan dari MobileNetV1. MobileNetV2 dibangun menggunakan *Depthwise Separable Convolution* yang daya komputasinya lebih rendah untuk konvolusi gambar tidak seperti CNN biasa. *Depthwise Separable Convolution* membagi konvolusi standar menjadi dua tahapan: *depthwise convolution* dan *pointwise convolution.* Hal ini bertujuan untuk mengurangi terjadinya multiplisitas, yang akan mempengaruhi biaya komputasi [19]. Gambar perbedaan *normal convolution* dan *depthwise convolution* dapat dilihat pada gambar 2.6.




**Gambar 2.6** Gambar *Normal Convolution dan Depthwise Convolution* [20]

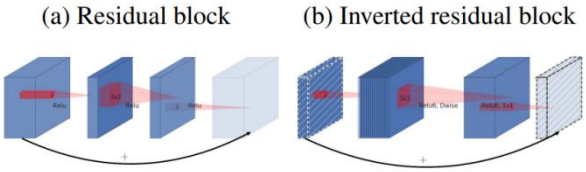
Pada *normal convolution,* semua *channel* pada sebuah *kernel* digunakan untuk menghasilkan *feature map*. Sedangkan pada *Depthwise Convolution*, setiap *channel* pada sebuah *kernel* menghasilkan *feature map. Pointwise Convolution* dikenal sebagai 1x1 *Convolution*, adalah operasi konvolusi yang menggunakan *kernel* berukuran 1x1. *Pointwise Convolution* digunakan untuk mengurangi dimensi saluran dari hasil *Depthwise Convolution*. Hal ini membantu mengurangi jumlah parameter dan komputasi yang diperlukan[19]. Gambar perbedaan *normal convolution* dan *pointwise Convolution* dapat dilihat pada gambar 2.7.




**Gambar 2.7** Gambar *Normal Convolution* dan *Pointwise Convolution*[20]

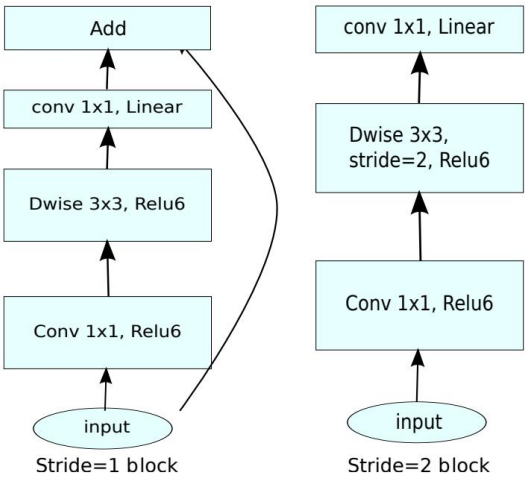
Pada gambar 2.7, *normal convolution* disimulasikan memiliki ukuran *spatial* satuannya adalah 8x8 piksel, ukuran kernel konvolusi yang digunakan adalah 5x5 piksel, jumlah saluran (*channel*) pada input atau output lapisan konvolusi adalah 3, dan jumlah filter atau saluran *output* pada lapisan konvolusi adalah 256. Sehingga jumlah operasinya perhitungan jumlah operasi adalah 8x8x5x5x3x256 = 1.228.800 operasi [20]. Sedangkan pada *pointwise convolution* disimulasikan memiliki jumlah operasi 8x8x1x1x3x256 = 49.152 operasi dan jumlah operasi *depthwise convolution* 8x8x5x5x3 = 4.800 operasi, Sehingga total jumlah operasi *depthwise separable convolution* adalah 4.800 + 49.152 = 53.952 yang berarti *depthwise separable convolution* lebih sedikit komputasi sekitar 22,7% dibandingkan dengan *normal convolution*.

Dalam MobileNetV2 memperkenalkan *inverted residual blocks* dan *Linear Bottlenecks*. Pada *inverted residual blocks*, konvolusi dapat mengakses aktivasi yang sebelumnya tidak dimodifikasi melalui penambahan input sebelum operasi penyaringan ke output yang difilter. Namun, ini hanya dapat dilakukan jika kontraksi menghasilkan M saluran sama dengan input dan tidak ada *down-sampling* dari *feature maps* [21]. *Residual blocks* memperluas jumlah *feature maps* nilai input dari dimensi M ke $M_{exp}$ di mana $M_{exp} > M$ dan biasanya $M_{exp} = 6 \times M$. Jumlah filter kemudian disingkat menjadi $M_{crt} < M_{exp}$ untuk membentuk sebuah *bottleneck*. Gambar *residual block* dan *inverted residual block* ditunjukkan pada gambar 2.8.




**Gambar 2.8** Gambar Residual block dan Inverted residual block [22]

Fungsi aktivasi nonlinier pada beberapa bagian lapisan digantikan dengan fungsi aktivasi linier sehingga disebut *linear bottleneck*. Hal ini dapat mengurangi resiko kehilangan kehilangan informasi penting selama proses pelatihan dan meningkatkan kemampuan model untuk mempertahankan informasi yang relevan [22]. Gambar blok konvolusi dapat dilihat pada gambar 2.9.




**Gambar 2.9** Gambar blok konvolusi [22]

Arsitektur MobileNetV2 terdiri dari lapisan konvolusi penuh awal dengan 32 filter, diikuti oleh 19 lapisan *bottleneck residual* seperti yang terdapat dalam Tabel 2.1. MobileNetV2

menggunakan ReLU6 sebagai nonlinieritas karena kekokohan penggunaannya dengan komputasi presisi rendah. Pada arsitektur ini *kernel* selalu menggunakan ukuran 3 × 3 sesuai standar untuk konvolusi modern, dan menggunakan *dropout* dan normalisasi *batch* selama pelatihan.

**Tabel 2.2** Arsitektur MobileNetV2[22]

| Input | Operator | *T* | *C* | *N* | *s* |
|---|---|---|---|---|---|
| $224^2$ x 3 | conv2d | - | 32 | 1 | 2 |
| $112^2$ x 32 | bottleneck | 1 | 16 | 1 | 1 |
| $112^2$ x 16 | bottleneck | 6 | 24 | 2 | 2 |
| $56^2$ x 24 | bottleneck | 6 | 32 | 3 | 2 |
| $28^2$ x 32 | bottleneck | 6 | 64 | 4 | 2 |
| $14^2$ x 64 | bottleneck | 6 | 96 | 3 | 1 |
| $14^2$ x 96 | bottleneck | 6 | 160 | 3 | 2 |
| $7^2$ x 160 | bottleneck | 6 | 320 | 1 | 1 |
| $7^2$ x 320 | conv2d 1x1 | - | 1280 | 1 | 1 |
| $7^2$ x 1280 | avgpool 7x7 | - | - | 1 | - |
| 1 x 1 x 1280 | conv2d 1x1 | - | K | - | |

### 2.2.8 InceptionV3

InceptionV3 adalah arsitektur *Deep Convolutional Neural Network GoogleNet* yang tersusun atas 42 *layer*, mengutamakan efisiensi komputasi dengan jumlah parameter yang lebih rendah. InceptionV3 merupakan penyempurnaan yang terdapat pada versi sebelumnya, di mana terdapat beberapa tambahan fitur, termasuk penggunaan RMSProp sebagai *optimizer*, penerapan konvolusi *factorized* 7x7, *BatchNorm*, dan penerapan label *smooting*. Terdapat penerapan faktorisasi untuk mengurangi jumlah parameter yang digunakan[23]

**Tabel 2.3** Arsitektur InceptionV3[23]

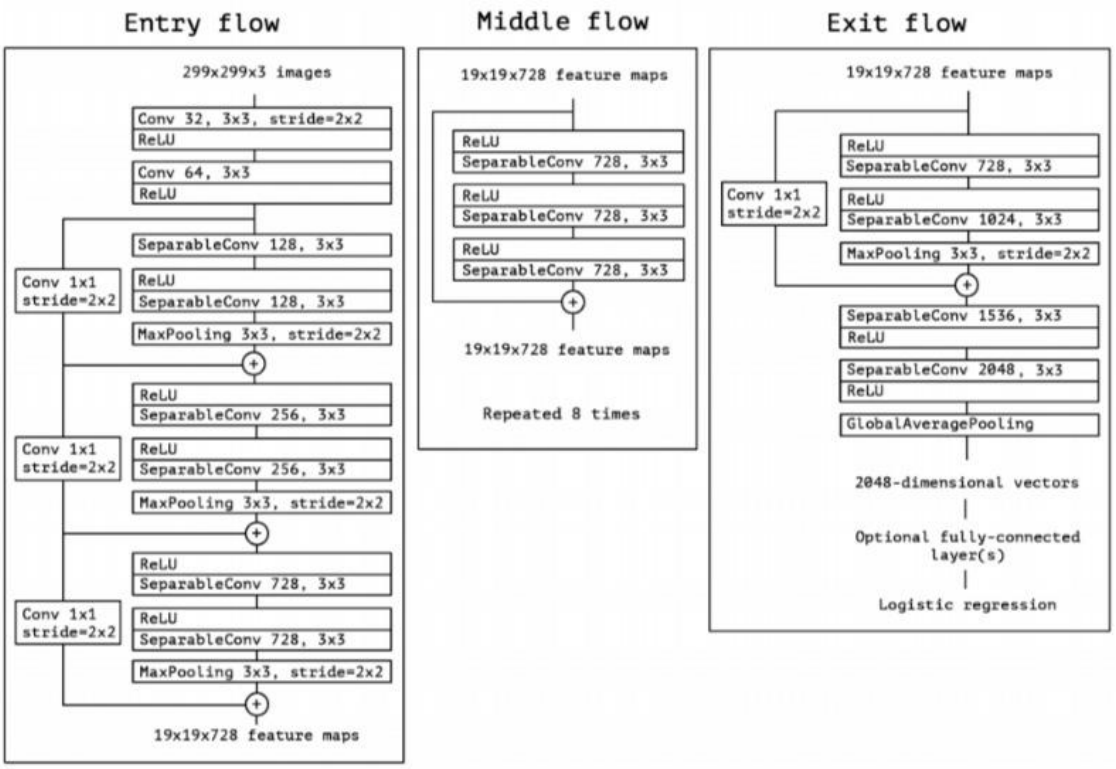
| ***Type*** | ***Patch / Stride Size*** | ***Input Size*** |
|---|---|---|
| Conv | 3×3/2 | 299×299×3 |
| Conv | 3×3/1 | 149×149×32 |
| Conv padded | 3×3/1 | 147×147×32 |
| Pool | 3×3/2 | 147×147×64 |
| Conv | 3×3/1 | 73×73×64 |
| Conv | 3×3/2 | 71×71×80 |
| Conv | 3×3/1 | 35×35×192 |
| 3 × Inception | Module 1 | 35×35×288 |
| 5 × Inception | Module 2 | 17×17×768 |
| 2 × Inception | Module 3 | 8×8×1280 |
| Pool | 8 × 8 | 8 × 8 × 2048 |
| Linear | Logits | 1 × 1 × 2048 |

| *Type* | *Patch / Stride Size* | *Input Size* |
|---|---|---|
| Softmax | Classifier | 1 × 1 × 1000 |

### 2.2.9 Xception

Xception, juga dikenal sebagai *Extreme Inception*, merupakan peningkatan dari arsitektur Inception sebelumnya. Arsitektur Xception terdiri dari 36 lapisan konvolusional yang berfungsi sebagai basis ekstraksi fitur dalam jaringan [24]. Struktur ini dicirikan oleh adanya lapisan konvolusional yang dipisahkan oleh jaringan residual, yang saling terhubung satu sama lain.

Pada arsitektur Xception, setelah lapisan konvolusional pertama dijalankan, setiap saluran dipisahkan secara spasial dengan menggunakan konvolusi 1×1. Keluaran dari proses ini kemudian disimpan untuk mendapatkan korelasi lintas saluran (*cross-channel correlation*)[25]. Dengan struktur yang lebih dalam dan pemisahan spasial pada saluran, Xception diharapkan dapat meningkatkan kemampuan ekstraksi fitur dan menghasilkan representasi yang lebih kaya dibandingkan dengan arsitektur Inception sebelumnya.




**Gambar 2.10** Arsitektur Xception[24]

### 2.2.10 Evaluasi

Evaluasi merupakan langkah penting untuk mengetahui seberapa baik model tersebut dapat memprediksi kategori atau label yang benar. Terdapat beberapa metrik evaluasi yang dapat digunakan untuk mengukur kinerja model, termasuk *loss function*, *accuracy*, *precision*, *recall*, dan *F-1Score*. Metrik-metrik ini memberikan gambaran yang lebih lengkap tentang seberapa baik model dapat membedakan kelas-kelas yang berbeda dalam data.

Akurasi adalah metrik yang mengukur sejauh mana model dapat mengklasifikasikan data dengan benar secara keseluruhan [7]. Untuk menghitung akurasi dengan rumus pada persamaan 2.5.

$$Akurasi = \frac{True\ Positives + True\ Negatives}{Jumlah\ Total\ Data} \quad (2.5)$$

Presisi merupakan persentase data positif yang diklasifikasikan dengan benar oleh model [17]. Presisi dapat dihitung dengan menggunakan rumus pada persamaan 2.6.

$$Presisi = \frac{True\ Positives}{True\ positives + False\ Positives} \quad (2.6)$$

*Recall* (atau *Sensitivity*) adalah metrik yang mengukur seberapa baik model dapat mengidentifikasi semua data positif yang seharusnya positif [17]. *Recall* dapat dihitung dengan menggunakan rumus pada persamaan 2.7.

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \quad (2.7)$$

*F-1 Score* adalah metrik yang menggabungkan *precision* dan *recall* untuk memberikan gambaran yang lebih seimbang tentang kinerja model. *F-1 Score* adalah rata-rata harmonik dari *precision* dan *recall*, *F-1 Score* dapat dihitung dengan menggunakan rumus pada persamaan 2.8.

$$F1\ Score = \frac{2 \times Presisi \times Recall}{Presisi + Recall} \quad (2.8)$$

dengan *True Positives* adalah jumlah data positif yang benar-benar diklasifikasikan dengan benar sebagai positif. *True Negatives* adalah jumlah data negatif yang benar-benar diklasifikasikan dengan benar sebagai negatif. *False Positives* adalah jumlah data negatif yang salah diklasifikasikan sebagai positif. *False Negatives* adalah jumlah data positif yang salah diklasifikasikan sebagai negatif [31].

*Loss function* adalah metrik yang digunakan untuk mengukur sejauh mana prediksi model CNN mendekati nilai yang sebenarnya pada tugas yang sedang dijalankan. *Loss function* menggambarkan perbedaan antara prediksi model dengan nilai yang sebenarnya, dan menjadi acuan untuk menyesuaikan bobot dan parameter model selama proses pelatihan. Salah satu *loss function* yang umum digunakan dalam CNN adalah *categorical cross-entropy*.

Fungsi *categorical cross-entropy loss* digunakan pada tugas klasifikasi multikelas. Cara kerjanya adalah dengan mengukur perbedaan antara distribusi probabilitas prediksi model dan distribusi probabilitas yang sebenarnya [16]. Kalkulasi nilai *cross-entropy loss* ditunjukkan pada Persamaan 2.9.

$$CGE = -\sum_{i=1}^{n} y_i \cdot \log\left(f(X_i)\right) \quad (2.9)$$

Menggunakan metrik-metrik ini secara bersama-sama membantu dalam mengevaluasi sejauh mana model dapat memprediksi dengan baik dan mengidentifikasi kelas-kelas yang berbeda dalam data.

### *2.2.11 Transfer Learning*

*Transfer learning* merupakan model arsitektur CNN yang sudah dilatih oleh suatu dataset sebelumnya yang bisa digunakan untuk klasifikasi pada dataset lain. *Transfer learning*

memiliki arsitektur lapisan *convolution* dan *pooling* yang lebih dalam dibandingkan arsitektur CNN sederhana, sehingga dapat melakukan ekstraksi tekstur citra lebih banyak dan menghasilkan informasi dari citra yang lebih baik. Kedalaman lapisan arsitektur pada *Transfer Learning* membuat suatu metode bisa mengatasi *overfitting* pada data yang sedikit[2].

### 2.2.12 Python

Python, sebagai bahasa pemrograman, semakin populer di kalangan ilmuwan data dan pengembang perangkat lunak. Python menyediakan *libraries* yang dapat digunakan dalam membuat model CNN. Beberapa dari libraries tersebut adalah Pandas, NumPy, Keras, dan TensorFlow. Python memiliki aplikasi yang lebih luas, termasuk pengembangan web, akses basis data, GUI desktop, komputasi ilmiah, dan pengembangan perangkat lunak. Antarmuka pengguna yang populer adalah Jupyter Notebook [26].

### 2.2.13 Tensorflow

TensorFlow merupakan salah satu *library* yang sangat terkemuka yang dikembangkan oleh Google untuk aplikasi *machine learning* dan *deep learning*. Perpustakaan ini dirancang untuk dijalankan pada CPU, GPU, serta prosesor IoT seperti Jetson Nano dan *neural network stick*. Pengumuman pertama mengenai *library* ini dilakukan pada tahun 2015. Meskipun begitu, versi yang stabil telah tersedia sejak tahun 2017 di bawah Lisensi Sumber Terbuka Apache. Dengan berada di bawah lisensi ini, perpustakaan tersebut dapat digunakan, dimodifikasi, dan didistribusikan tanpa adanya royalti yang harus diberikan kepada Google [27].

### 2.2.14 Keras

Keras merupakan API jaringan saraf berbasis Python yang digunakan untuk mengeksekusi TensorFlow, CNTK, dan Theano. Dalam konteks TensorFlow 2.0, Keras diimplementasikan dengan versi terbaru, yaitu Keras 2.3.0. Penggunaan Keras dalam Python menggunakan notasi *tf.keras*. API ini didasarkan pada empat prinsip kerja, meliputi kemudahan penggunaan, modularitas, kemudahan perluasan, dan kompatibilitas yang baik dengan bahasa pemrograman Python [27].

### 2.2.15 Android Studio

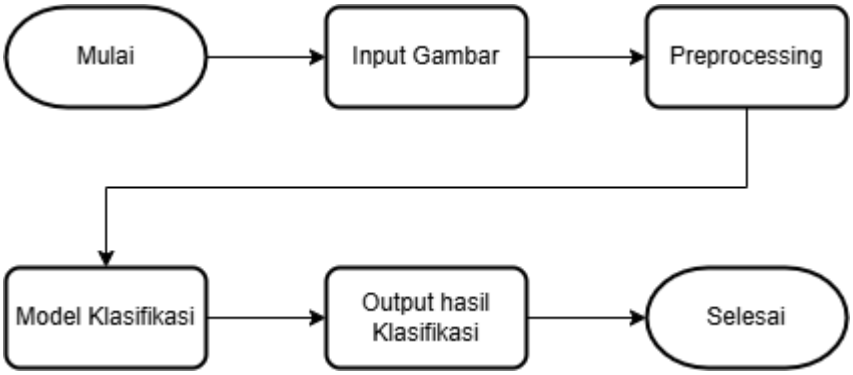
Android Studio merupakan perangkat lunak resmi dari Google untuk pengembangan aplikasi Android. Ini adalah perangkat lunak sumber terbuka yang meningkatkan efisiensi biaya dan menyediakan berbagai alat untuk pengembangan aplikasi di berbagai perangkat Android. Keutamaan dari perangkat lunak ini adalah ketersediaan *platform real-time*, yaitu *emulator*, untuk menguji aplikasi secara langsung. Selain itu, aplikasi juga dapat diuji pada *smartphone* melalui fitur USB *debugging*. Android Studio menyediakan beragam tampilan, seperti tombol, tampilan gulir, tampilan daftar, kotak centang, dan tampilan pencarian, untuk mendukung pengembangan aplikasi. *Editor* tata letak yang mendukung fungsi seret dan lepas komponen membuat pengembangan aplikasi lebih ramah pengguna [28].

*(Halaman ini sengaja dikosongkan)*

# BAB III METODOLOGI

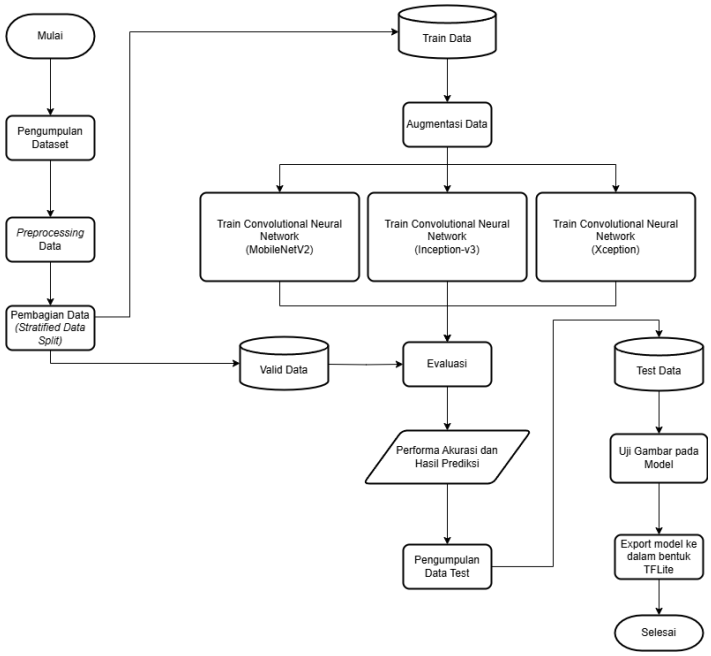
## 3.1 Metode yang dirancang

Rancangan pengembangan klasifikasi Tanaman Bunga pada aplikasi berbasis android dimulai dari membuat model CNN untuk klasifikasi tanaman, kemudian men-*deploy* model CNN tersebut ke dalam aplikasi dengan menggunakan TensorFlow Lite (TFLite). Rancangan aplikasi klasifikasi yang dikembangkan digambarkan pada Gambar 3.1.




**Gambar 3.1** Rancangan Aplikasi Klasifikasi

Tahap pertama dalam rancangan pengembangan klasifikasi pada aplikasi adalah input gambar. Gambar yang diinput adalah gambar yang terdapat satu jenis tanaman bunga di dalamnya. Aplikasi dapat menerima input gambar dengan dua cara, yaitu dengan memilih salah satu gambar yang terdapat di galeri perangkat, atau mengambil gambar melalui kamera perangkat secara langsung. Setelah aplikasi menerima input gambar, selanjutnya dilakukan proses *preprocessing* gambar. Pada tahap ini, gambar di-*resize* sesuai dengan ukuran yang diambil pada model. Gambar yang telah melalui proses *preprocessing* berikutnya digunakan untuk klasifikasi dengan menggunakan model. Adapun rancangan untuk model dijelaskan pada Gambar 3.2. Setelah gambar diklasifikasi menggunakan model, aplikasi akan menuju laman *output* hasil klasifikasi. Pada laman ini, ditampilkan gambar yang sebelumnya telah diinput dan jenis tanaman bunga hasil klasifikasi.




**Gambar 3.2** Perancangan Model Klasifikasi

Detil pengembangan klasifikasi tanaman bunga terdiri dari tahap pengumpulan dataset, *preprocessing* data, pembagian data, augmentasi data, pelatihan model, evaluasi model dan ekspor model dalam bentuk TFLite. Adapun detil setiap tahapan dijelaskan pada subbab 3.1.1 sampai 3.1.8.

### 3.1.1 Pengumpulan Dataset

Sebelum melakukan pelatihan model *image classification*, dilakukan terlebih dahulu pencarian dataset gambar tanaman bunga. Dataset yang dicari menyesuaikan dengan tanaman bunga yang terdapat di Indonesia sehingga dapat di uji coba secara langsung. Dataset diambil dari *Visual Geometry Group University of Oxford* dengan nama dataset *102 Category Flower Dataset* yang diterbitkan oleh Maria-Elena Nilsback dan Andrew Zisserman pada tahun 2008. *102 Category Flower Dataset* memiliki 102 label dengan jumlah gambar 8189. Setiap label memiliki minimal 40 gambar [29]. Adapun detil citra yang terdapat pada *dataset 102 Category Flower* beserta jumlahnya ditampilkan pada Tabel 3.1.

**Tabel 3.1** Data Tanaman Bunga *102 Category Flower*

| Label | Jumlah | Label | Jumlah | Label | Jumlah |
|---|---|---|---|---|---|
| alpine sea holly | 43 | fire lily | 40 | peruvian lily | 82 |
| anthurium | 105 | foxglove | 162 | petunia | 258 |
| artichoke | 78 | frangipani | 166 | pincushion flower | 59 |
| azalea | 96 | fritillary | 91 | pink primrose | 40 |
| ball moss | 46 | garden phlox | 45 | pink-yellow dahlia | 109 |
| balloon flower | 49 | gaura | 67 | poinsettia | 93 |
| barbeton daisy | 127 | gazania | 78 | primula | 93 |
| bearded iris | 54 | geranium | 114 | prince of wales feathers | 40 |
| bee balm | 66 | giant white arum lily | 56 | purple coneflower | 85 |
| bird of paradise | 85 | globe thistle | 45 | red ginger | 42 |
| bishop of llandaff | 109 | globe-flower | 41 | rose | 171 |
| black-eyed susan | 54 | grape hyacinth | 41 | ruby-lipped cattleya | 75 |
| blackberry lily | 48 | great masterwort | 56 | siam tulip | 41 |
| blanket flower | 49 | hard-leaved pocket orchid | 60 | silverbush | 52 |
| bolero deep blue | 40 | hibiscus | 131 | snapdragon | 87 |
| bougainvillea | 128 | hippeastrum | 76 | spear thistle | 48 |
| bromelia | 63 | japanese anemone | 55 | spring crocus | 42 |
| buttercup | 71 | king protea | 49 | stemless gentian | 66 |
| californian poppy | 102 | lenten rose | 67 | sunflower | 61 |
| camellia | 91 | lotus lotus | 137 | sweet pea | 56 |
| canna lily | 82 | love in the mist | 46 | sweet william | 85 |
| canterbury bells | 40 | magnolia | 63 | sword lily | 130 |

| Label | Jumlah | Label | Jumlah | Label | Jumlah |
|---|---|---|---|---|---|
| cape flower | 108 | mallow | 66 | thorn apple | 120 |
| carnation | 52 | marigold | 67 | tiger lily | 45 |
| cautleya spicata | 50 | mexican aster | 40 | toad lily | 41 |
| clematis | 112 | mexican petunia | 82 | tree mallow | 58 |
| colt's foot | 87 | monkshood | 46 | tree poppy | 62 |
| columbine | 86 | moon orchid | 40 | trumpet creeper | 58 |
| common dandelion | 92 | morning glory | 107 | wallflower | 196 |
| corn poppy | 41 | orange dahlia | 67 | water lily | 194 |
| cyclamen | 154 | osteospermum | 61 | watercress | 184 |
| daffodil | 59 | oxeye daisy | 49 | wild pansy | 85 |
| desert-rose | 63 | passion flower | 251 | windflower | 54 |
| english marigold | 65 | pelargonium | 71 | yellow iris | 49 |

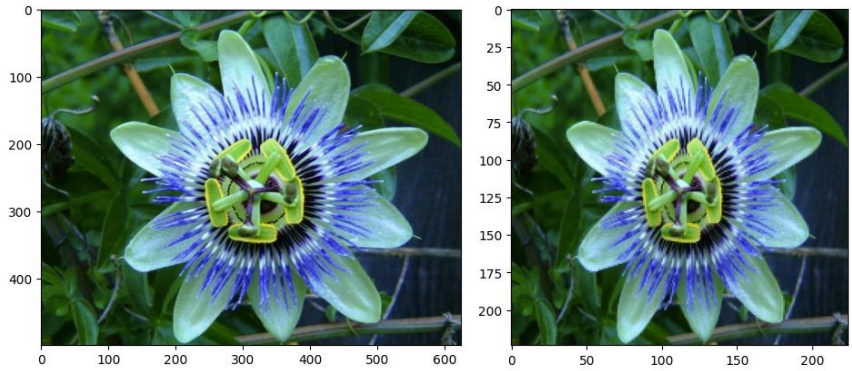
Pada dataset tersebut setiap label terdiri dari 40 sampai 258 gambar dan setiap gambar memiliki ukuran yang berbeda-beda. Beberapa contoh data gambar dari *102 Category Flower* ditunjukkan pada Gambar 3.3.

**Gambar 3.3** Contoh Data *102 Category Flower Dataset*

### 3.1.2 *Preprocessing* Data

Pada tahapan *preprocessing* data, semua citra pada dataset yang sebelumnya berukuran sembarang akan di-*resize* menjadi berukuran 224 x 224 piksel untuk MobileNetV2, dan 299 x 299 piksel untuk InceptionV3 dan Xception. Ukuran *resizing* mengikuti ukuran input gambar *default* pada setiap arsitektur dengan menggunakan pustaka *Image* yang dimiliki Tensorflow. *Resizing* ini bertujuan untuk mengurai harga komputasi sehingga mempersingkat waktu training. Contoh citra sebelum dilakukan *resize* dan sudah dilakukan *resize* ditunjukkan pada

gambar 3.4.




**Gambar 3.4** Gambar yang di-*resize* dengan *preprocessing data*

### 3.1.3 Pembagian Data

Ketika seluruh gambar dalam dataset telah di-*preprocessing*, kemudian gambar akan dibagi menjadi *training data* dan *valid data*. *Training data* merupakan dataset yang digunakan untuk melatih model. Sedangkan *valid data* digunakan untuk proses evaluasi model dan mencegah terjadinya *overfitting*. Metode yang akan digunakan untuk pembagian data adalah *Stratified Shuffle Split* (*Stratified Random Sampling*) yang diimplementasikan menggunakan pustaka Scikit-Learn. Metode *Stratified Shuffle Split* mengambil seluruh data, melakukan *shuffle* pada data, dan kemudian membagi data menjadi *training data* dan *valid data* untuk setiap kelas dengan mempertahankan rasio antara *training data* dan *valid data* pada setiap label [30].

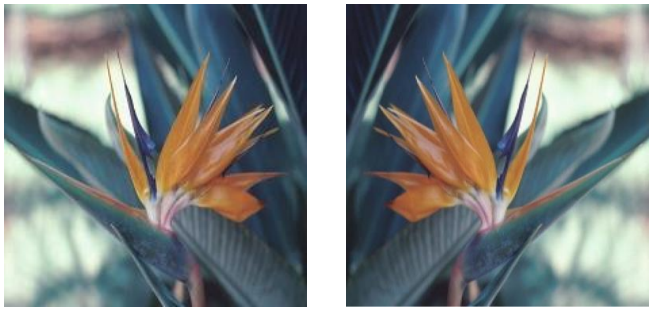
Pemisahan dataset umumnya dilakukan dengan perbandingan 70:30 atau 80:20, di mana 70% atau 80% data digunakan untuk melatih model, dan sisanya untuk validasi. Dengan perbandingan ini, data yang cukup tersedia untuk melatih model, dan juga terdapat data yang cukup untuk menguji dan mengevaluasi kinerja model. Untuk dataset yang besar, umumnya disarankan untuk mengalokasikan porsi yang lebih besar untuk pelatihan, yaitu sekitar 80% atau lebih. Hal ini dilakukan untuk memastikan bahwa model dapat mempelajari pola yang ada dalam data dengan baik, dan dapat melakukan generalisasi dengan baik terhadap data baru [31]. Pada penelitian ini pembagian data akan menggunakan perbandingan 80:20.

### 3.1.4 Augmentasi Data

Augmentasi data adalah suatu teknik yang dibuat untuk meningkatkan variasi data dengan cara memodifikasi dari data yang sudah peroleh sebelumnya. Augmentasi data dilakukan dengan menggunakan pustaka *Image* yang dimiliki TensorFlow. Pada tahapan ini gambar di dataset dirotasi, dibalik secara horizontal, digeser secara horizontal, didistorsi, dan diperbesar. Contoh rotasi gambar dengan augmentasi data ditunjukkan pada Gambar 3.5.

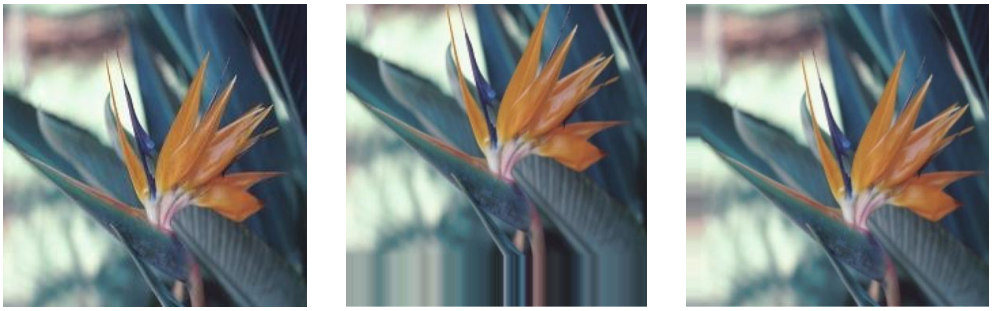
**Gambar 3.5** Gambar yang Dirotasi dengan Augmentasi Data

Selanjutnya gambar balik secara horizontal. Hal ini digunakan untuk melatih model pada kasus dengan gambar yang objeknya menghadap berbagai arah. Contoh gambar yang dibalik secara horizontal dengan augmentasi data ditunjukkan pada Gambar 3.6.



**Gambar 3.6** Gambar yang Dibalik dengan Augmentasi Data

Selanjutnya gambar digeser secara horizontal dan vertikal. Hal ini digunakan untuk melatih model pada kasus dengan gambar yang objeknya tidak berada di tengah. Contoh gambar yang digeser secara horizontal dan vertikal dengan augmentasi data ditunjukkan pada Gambar 3.7.



**Gambar 3.7** Gambar yang Digeser secara horizontal dan vertikal dengan Augmentasi Data

Berikutnya gambar diperbesar secara acak sampai dengan 20%. Hal ini bertujuan agar model mempelajari berbagai objek dengan ukuran yang bervariasi pada gambar. Contoh gambar yang dilakukan pembesaran dengan augmentasi data ditunjukkan pada gambar 3.8.

**Gambar 3.8** Gambar yang Diperbesar dengan Augmentasi Data

Lalu gambar didistorsi pada sumbu x dan y. Hal ini dilakukan agar model mempelajari berbagai objek yang diambil dengan sudut yang berbeda-beda. Contoh gambar yang didistorsi dengan augmentasi data ditunjukkan pada Gambar 3.9.

**Gambar 3.9** Gambar yang Didistorsi dengan Augmentasi Data

### 3.1.5 Pelatihan Model

Model *Neural Network* dilatih untuk mengklasifikasi gambar tanaman bunga. Tahap pelatihan model bertujuan untuk meminimalkan nilai *loss function*. Semakin rendah nilai *loss* yang didapat semakin baik model bekerja. *Loss* adalah nilai yang menunjukkan jumlah kesalahan dalam model. Dan tentunya mengukur seberapa baik atau buruk kinerja pada model. Jika nilai *loss* tinggi, maka kesalahannya tinggi. Sehingga semakin rendah nilai *loss* yang didapat oleh model menunjukkan bahwa model bekerja semakin baik.

Arsitektur yang digunakan pada penelitian ini adalah arsitektur MobiletNetV2, InceptionV3, Xception. Pada pengujian akan menggunakan *batch size* sebanyak 32 karena keterbatasan memori GPU. Jumlah berapa kali model melatih satu putaran lengkap dari seluruh data pada dataset pelatihan ditentukan berdasarkan jumlah *epoch*. Dengan kata lain *epoch* merupakan jumlah berapa kali algoritma melihat dataset secara keseluruhan. Jumlah *epoch* mula-mula adalah 50. Namun untuk menghemat waktu pelatihan model, pelatihan akan dihentikan sesuai kebutuhan jika sudah terjadi konvergensi, walaupun jumlah *epoch* belum tercapai. Jika setelah *epoch* 50 belum terjadi konvergensi, pelatihan akan dihentikan dan mengambil *epoch* dengan nilai akurasi *training* (*plateau*) tertinggi. Pelatihan disebut konvergen ketika sudah tidak terjadi perubahan *plateau* atau *plateau* sudah mencapai 100%. *Optimizer* yang akan digunakan adalah *optimizer* Adam dengan parameter *learning rate* 0.001, momentum 0, dan nilai *decay* 0. Parameter Adam ini tidak diubah selama pengujian dilakukan.

Parameter arsitektur yang digunakan dapat dilihat pada Tabel 3.2, arsitektur MobileNetV2 Tabel 2.1, arsitektur InceptionV3 pada Tabel 2.2, dan arsitektur Xception pada Gambar 2.10.

**Tabel 3.2** Parameter Arsitektur

| Parameter | Nilai |
|---|---|
| Ukuran *batch* | 32 |
| Jumlah *epoch* | 50 |
| *Optimizer* | Adam (*learning rate* 0.001, momentum=0.0, *decay*=0.0) |
| Fungsi Aktivasi | Softmax |

### 3.1.6 Evaluasi Model

Evaluasi model dilakukan dengan memanggil fungsi *evaluate()*. Fungsi ini disediakan oleh kerangka kerja *machine learning*, seperti TensorFlow atau PyTorch, untuk mengukur kinerja model pada kumpulan data yang tidak digunakan selama proses pelatihan. Fungsi

*evaluate()* akan mengembalikan nilai *loss* dan akurasi pada dataset yang diuji.

### 3.1.7 Uji Gambar Pada Model

Uji coba gambar dilakukan pada setiap model yang telah dibuat. Gambar yang diambil secara acak dari berbagai *search engine*, dan gambar yang dites tidak digunakan pada tahap *training* sebelumnya. Jumlah gambar yang dites pada setiap kelas ada 10 gambar. Dengan jumlah kelas yang terdapat pada dataset sebanyak 102 kelas, jumlah seluruh gambar yang disediakan untuk *testing* ada 1020 gambar. Dari uji coba tersebut, dicatat jumlah sukses dan gagal model dapat mengklasifikasi tanaman bunga sesuai dengan jenisnya. Dari jumlah sukses dan gagal, bisa dihitung persentase akurasi sistem klasifikasi pada jenis tanaman bunga tersebut.

### 3.1.8 *Export* Model dalam TFLite

Model yang telah dibuat diekspor dalam bentuk TFLite. TFLite adalah *framework* yang mengonversi *trained* model dari TensorFlow ke format khusus yang dapat dioptimalkan untuk kecepatan atau penyimpanan. Model format tersebut dapat diterapkan pada perangkat lunak ponsel yang menggunakan Android [8].

### 3.1.9 Uji Gambar pada Format TFLite

Uji coba gambar dilakukan pada setiap model yang telah diekspor ke dalam bentuk TFLite dan telah di-*deploy* pada aplikasi. Gambar yang diuji sama dengan gambar yang digunakan untuk uji gambar pada model. Dari uji coba tersebut, dicatat jumlah sukses dan gagal model dapat mengklasifikasi tanaman bunga sesuai dengan jenisnya. Dari jumlah sukses dan gagal, bisa dihitung persentase akurasi sistem klasifikasi pada jenis tanaman bunga tersebut. Sehingga dapat dihitung nilai delta model yang belum di-*convert* ke dalam format TFLite dan model yang sudah di-*convert* ke dalam format TFLite.

### 3.1.10 Uji Coba Aplikasi

Pengujian aplikasi berfokus pada pengujian persyaratan fungsional dari aplikasi untuk mendapatkan kondisi masukan yang sesuai dengan kebutuhan fungsional. Keluaran dari masukan akan diperiksa apakah sesuai dengan hasil yang diharapkan. Berikut adalah tabel pengujian aplikasi yang digunakan:

**Tabel 3.3** Uji Coba Aplikasi

| No | Skenario Pengujian | Testcase | Hasil Uji yang diharapkan | Kesimpulan yang diharapkan |
|---|---|---|---|---|
| 1 | Mengambil gambar dari kamera perangkat | Masukan:<br>- Menekan tombol "*Take Image*"<br>- Memposisikan kamera pada objek bunga<br>- Menekan tombol "*Capture*" | Gambar berhasil diambil. | Pengujian sukses |
| 2 | Memilih gambar dari galeri perangkat | Masukan:<br>- Menekan tombol "*Load Image*"<br>- Memilih foto pada galeri | Gambar berhasil dipilih. | Pengujian sukses |

| No | Skenario Pengujian | Testcase | Hasil Uji yang diharapkan | Kesimpulan yang diharapkan |
|---|---|---|---|---|
| | | - Menekan tombol "*Choose*" | | |
| 3 | Menjalankan proses klasifikasi | Masukan:<br>- Memilih foto yang digunakan<br>- Menekan tombol "*Choose*" atau tombol "*Capture*" | Proses klasifikasi berhasil berjalan. | Pengujian sukses |
| 4 | Melihat hasil klasifikasi | - | Sistem berhasil menampilkan hasil klasifikasi | Pengujian sukses |

## 3.2 Peralatan pendukung

1. Tensorflow
   Tensorflow merupakan *library open-source* yang sering digunakan dalam *training* model *Machine Learning* dan *Deep Learning*. Pada proyek ini digunakan Tensorflow Lite yang merupakan versi ringan dari Tensorflow, yang dirancang khusus untuk *mobile devices.* Tensorflow Library digunakan untuk klasifikasi gambar
2. Google Colab
   Google Colaboratory adalah lingkungan *notebook* Jupyter berbasis *cloud online* gratis yang memungkinkan pengguna melakukan *train machine learning* dan *deep learning* pada CPU, GPU, dan TPU.
3. Android Studio
   Android Studio adalah *Integrated Development Environment* (IDE) resmi untuk pengembangan aplikasi Android. Android Studio berbasiskan dari IntelliJ IDEA, sebuah Java IDE untuk perangkat lunak, dan menggabungkan pengeditan kode dan alat pengembangnya.

## 3.3 Perancangan Aplikasi

### 3.3.1 Pseudocode *Preprocessing* Data

Pseudocode adalah deskripsi terstruktur mengenai urutan dari suatu algoritma pemrograman yang dituliskan secara sederhana agar mudah dipahami oleh pembaca yang awam akan pemrograman. Pada implementasi aplikasi klasifikasi, tahap pengembangan yang terjadi dapat dibilang cukup kompleks sehingga dibutuhkan pseudocode untuk mempermudah para pengembang aplikasi dalam mengerjakan kode aplikasinya. Sebelum dataset digunakan untuk pembuatan model, dataset dilakukan *preprocessing* terlebih dahulu. Dikarenakan ukuran gambar yang terdapat pada dataset berbeda-beda, semua gambar di-*resize* dengan ukuran 224×224 piksel untuk MobileNetV2 dan 299×299 piksel untuk InceptionV3 dan Xception. Pseudocode *preprocessing* dataset ditunjukkan pada gambar 3.10.

```
resized_img = tf.image.resize(image, target_size).numpy()
```

**Gambar 3.10** Pseudocode *Preprocessing* Dataset

### 3.3.2 Pseudocode Augmentasi Data

Augmentasi data menggunakan *Image Data Generator* yang dimiliki Tensorflow. Adapun pseudocode terdapat pada gambar 3.11.

```
train_datagen = ImageDataGenerator(rescale=1./255,
                                   rotation_range=30,
                                   width_shift_range=0.2,
                                   height_shift_range=0.2,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True,
                                   fill_mode='nearest')
```

**Gambar 3.11** Pseudocode Augmentasi Data

### 3.3.3 Pseudocode Meluncurkan *Pre-Trained* Model

Untuk menggunakan *Pre-Trained* Model, model perlu di-*import* terlebih dahulu dari Keras. Pada tugas akhir ini, model yang digunakan adalah MobileNetV2, InceptionV3, dan Xception. Pseudocode meluncurkan *Pre-Trained* Model dapat dilihat pada gambar 3.12.

```
base_model = Pre-Trained(input_shape=(weight, height, 3),
                         include_top=False,
                         weights='imagenet')
```

**Gambar 3.12** Pseudocode Meluncurkan *Pre-Trained* Model

Adapun parameter yang digunakan yaitu ukuran input yang digunakan 224×224 untuk MobileNetV2 dan 299×299 untuk InceptionV3, dan Xception dengan 3 bentuk warna RGB, tetap menggunakan lapisan *output* dari *Pre-Trained* Model, dan lapisan *output* yang digunakan adalah *softmax*.

### 3.3.4 Pseudocode Meluncurkan Kamera di Perangkat Keras

Untuk meluncurkan kamera pada Android, digunakan fungsi *TakePicturePreview()* yang disediakan Kotlin. Fungsi tersebut membantu meluncurkan kamera untuk mengambil foto. Untuk memeriksa apakah *user* mengizinkan aplikasi untuk mengakses kamera, digunakan fungsi *checkSelfPermission()*. Jawaban dari *user* mengenai permohonan akses diterima pada *Manifest.permission.CAMERA*, yang juga merupakan parameter dari fungsi *checkSelfPermission()*. Gambar 3.13 menunjukkan pseudocode memeriksa apakah *user* mengizinkan aplikasi untuk mengakses kamera dan juga meluncurkan kamera pada perangkat Android.

```
buttonCamera.setOnClickListener {
    if (ContextCompat.checkSelfPermission(this, android.Manifest.
        permission.CAMERA)
        == PackageManager.PERMISSION_GRANTED
    ) {
        TakePicturePreview.launch(null)
    } else {
        requestPermission.launch(android.Manifest.permission.
            CAMERA)
    }
}
```

**Gambar 3.13** Pseudocode Meluncurkan Kamera di Perangkat Keras

### 3.3.5 Pseudocode Meluncurkan Galeri di Perangkat Keras

Untuk meluncurkan galeri pada perangkat android, digunakan *intent Images.Media.EXTERNAL_CONTENT_URI* yang disediakan MediaStore. *Intent* tersebut untuk mengakses galeri dari perangkat lunak. Untuk memeriksa apakah *user* mengizinkan aplikasi untuk mengakses galeri, digunakan fungsi *checkSelfPermission()*. Jawaban dari *user* mengenai permohonan akses diterima pada *Manifest.permission. READ_MEDIA_IMAGES*, yang juga merupakan parameter dari fungsi *checkSelfPermission(). Pseudocode* untuk memeriksa apakah *user* mengizinkan aplikasi untuk mengakses galeri dan juga meluncurkan galeri pada perangkat Android ditunjukkan pada Gambar 3.14.

```
buttonLoadGalery.setOnClickListener {
    if (ContextCompat.checkSelfPermission(this, android.Manifest.
        permission.READ_MEDIA_IMAGES)
        == PackageManager.PERMISSION_GRANTED
    ) {
        val intent = Intent(Intent.ACTION_PICK, MediaStore.Images.
            Media.EXTERNAL_CONTENT_URI)
        intent.type = "image/*"
        val mimeTypes = arrayOf("image/jpeg", "image/jpg", "image/
            png")
        intent.putExtra(Intent.EXTRA_MIME_TYPES, mimeTypes)
        intent.flags = Intent.FLAG_GRANT_READ_URI_PERMISSION
        onresult.launch(intent)
    } else {
        requestPermission.launch(android.Manifest.permission.
            READ_MEDIA_IMAGES)
    }
}
```

**Gambar 3.14** Pseudocode Meluncurkan Galeri di Perangkat Keras

### 3.3.6 Pseudocode *Preprocessing* Gambar pada Perangkat Keras

*Preprocessing* gambar untuk keperluan prediksi dalam bahasa Kotlin. Pertama, *bitmap* diubah ukurannya menjadi dimensi *size* x *size* menggunakan fungsi *Bitmap.createScaledBitmap*. Ukuran akan diubah menyesuikan ukuran *default* model yang digunakan. Untuk *MobileNetV2* akan di-*resize* menjadi 224 x 224 piksel, Xception, dan InceptionV3 akan di-*resize* 299 x 299 piksel. Hal ini dilakukan karena ukuran *bitmap* yang didapatkan berukuran sembarang.

Setelah itu, bitmap yang telah diubah ukurannya dikonversi menjadi objek *ByteBuffer* melalui fungsi *convertBitmapToByteBuffer*. Fungsi ini mengalokasikan *ByteBuffer* dengan ukuran tertentu yang ditentukan oleh parameter *size* dan mengatur urutan *byte* sesuai dengan urutan asli perangkat. Kemudian, semua *pixel* dari *bitmap* diambil dan disimpan dalam *array intValues*. Selanjutnya, nilai RGB dari setiap *pixel* dinormalisasi dengan mengurangi 127.5 dan membaginya dengan 127.5, lalu hasil normalisasi tersebut dimasukkan ke dalam *ByteBuffer* sebagai nilai *float*. Tujuan dari proses ini adalah untuk menyiapkan data gambar yang telah dinormalisasi untuk digunakan dalam model prediksi.

Fungsi yang sudah dibuat akan mengembalikan nilai *byteBuffer* yang merupakan nilai *bitmap* yang sudah *preprocessing* dan normalisasi. Kode sumber untuk *preprocessing* gambar pada perangkat keras dapat lihat pada gambar 3.15.

```
val resizedbitmap = Bitmap.createScaledBitmap(bitmap, size, size,
     true)
val bytebuffer = convertBitmapToByteBuffer(resizedbitmap, size)

private fun convertBitmapToByteBuffer(bitmap: Bitmap, size: Int):
     ByteBuffer {
    val byteBuffer = ByteBuffer.allocateDirect(4 * size * size *
        3)
    byteBuffer.order(ByteOrder.nativeOrder())

    val intValues = IntArray(size * size)
    bitmap.getPixels(intValues, 0, bitmap.width, 0, 0, bitmap.
        width, bitmap.height)

    var pixelIndex = 0
    for (i in 0 until size) {
        for (j in 0 until size) {
            val pixelValue = intValues[pixelIndex++]

            // Normalize RGB values
            byteBuffer.putFloat((Color.red(pixelValue) - 127.5f) /
                 127.5f)
            byteBuffer.putFloat((Color.green(pixelValue) - 127.5f)
                 / 127.5f)
            byteBuffer.putFloat((Color.blue(pixelValue) - 127.5f)
                / 127.5f)
        }
    }

    return byteBuffer
}
```

**Gambar 3.15** Pseudocode *Preprocessing* Gambar pada Perangkat Keras

### 3.3.7 Pseudocode Klasifikasi Gambar

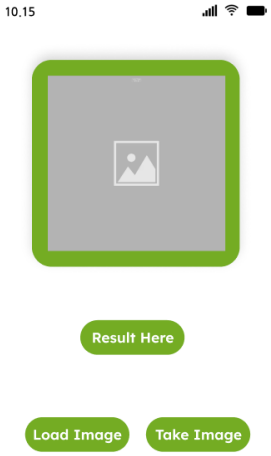
Untuk melakukan klasifikasi gambar dengan model yang telah dilatih sebelumnya. Pertama, model diinisialisasi dengan memanggil *Model.newInstance(this)*. Input untuk model didefinisikan menggunakan *TensorBuffer* dengan ukuran tetap sesuai dengan dimensi gambar yang diinginkan (1, *size*, *size*, 3) dan tipe data FLOAT32. *TensorBuffer* kemudian diisi dengan data dari *bytebuffer* yang berisi nilai-nilai *pixel* gambar yang telah dinormalisasi. Setelah itu, input tersebut diproses oleh model dengan memanggil *model.process(inputFeature)*, menghasilkan *output* yang disimpan dalam *TensorBuffer*. Nilai-nilai hasil prediksi diambil dari *TensorBuffer* sebagai *array float*. Fungsi *getBestPrediction* digunakan untuk menentukan nilai prediksi terbaik (*bestValue*) dan indeksnya. Indeks tersebut kemudian digunakan untuk mendapatkan kata yang sesuai dari daftar *allWords*. Nilai prediksi terbaik diformat menjadi dua angka desimal. Terakhir, model ditutup dengan memanggil *model.close()*.

```
val model = Model.newInstance(this)
val inputFeature = TensorBuffer.createFixedSize(intArrayOf(1,
    size, size, 3), DataType.FLOAT32)
inputFeature.loadBuffer(bytebuffer)
val outputs = model.process(inputFeature)
val outputFeature = outputs.outputFeature0AsTensorBuffer
val outputArray = outputFeature.floatArray
val (bestValue, bestIndex) = getBestPrediction(outputArray)
val bestWord = allWords.elementAt(bestIndex)
val formattedValue = "%.2f".format(bestValue)
model.close()
```

**Gambar 3.16** Pseudocode Klasifikasi Gambar

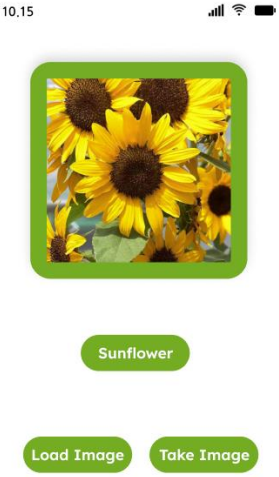
### 3.3.8 Perancangan Antarmuka Pengguna

Tampilan saat halaman utama ditunjukkan pada Gambar 3.17. Di dalamnya terdapat komponen tombol untuk memilih foto pada galeri atau mengambil foto lewat kamera.




**Gambar 3.17** Halaman Utama

Pada Gambar 3.18 adalah tampilan setelah melewati proses klasifikasi bunga. Laman ini menampilkan hasil foto yang diambil oleh *user*, diikuti nama bunga.




**Gambar 3.18** Halaman Hasil Klasifikasi

## 3.4 Implementasi

### 3.4.1 Pengambilan Data

Sebelum memasuki tahap implementasi, dilakukan pengumpulan data bunga *Oxford 102 Flowers* untuk dianalisis sebagai bahan dasar Tugas Akhir. Jumlah keseluruhan data setelah dilakukan *filtering* adalah sebanyak 8189 data, yang terbagi menjadi 102 kelas. Implementasi pengunduhan dataset ditunjukkan pada Kode Sumber 3.1.

```
import os
import requests

url = "https://www.robots.ox.ac.uk/~vgg/data/flowers
    /102/102flowers.tgz"
file_name = "102flowers.tgz"

if not os.path.exists(file_name):
    response = requests.get(url, stream=True)
    with open(file_name, 'wb') as file:
        for chunk in response.iter_content(chunk_size
            =8192):
            if chunk:
                file.write(chunk)
```

**Kode Sumber 3.1** Pengunduhan Dataset

Pada setiap kelas, gambar terbagi menjadi dua direktori, yaitu *training set* untuk melatih model, dan *validation set* untuk mengevaluasi model. Dataset *training* dan validasi tersebut dibagi dengan rasio 80:20. Implementasi pembagian dataset ditunjukkan pada Kode Sumber 3.2.

```
import os
import shutil
from sklearn.model_selection import
    StratifiedShuffleSplit
import pandas as pd

df = pd.read_csv('dataset.csv')
X = df['Image']
y = df['Label']

sss = StratifiedShuffleSplit(n_splits=1, test_size=0.2,
    random_state=4)

for train_index, valid_index in sss.split(X, y):
    train_data, valid_data = df.iloc[train_index], df.
        iloc[valid_index]

train_dir = 'train'
valid_dir = 'valid'
os.makedirs(train_dir, exist_ok=True)
os.makedirs(valid_dir, exist_ok=True)

for image_id in train_data['Image']:
    source_path = os.path.join('dataset', image_id)
    dest_path = os.path.join(train_dir, image_id)
    shutil.copy(source_path, dest_path)

for image_id in valid_data['Image']:
    source_path = os.path.join('dataset', image_id)
    dest_path = os.path.join(valid_dir, image_id)
    shutil.copy(source_path, dest_path)
```

**Kode Sumber 3.2** Pembagian Dataset

### 3.4.2 Implementasi Struktur Dataset

Implementasi struktur dataset terbagi menjadi *training set* dan *validation set* dengan perbandingan rasio 80:25. Visualisasi dari struktur dataset ditunjukkan oleh Gambar 3.19.

```
Dataset
+---train
|    +---class 1
|    +---class 2
|    +---class n
|    +---class 102
+---val
     +---class 1
     +---class 2
     +---class n
     +---class 102
```

**Gambar 3.19** Struktur Dataset

Untuk Strukturisasi dataset yang sudah terbagi sesuai dengan masing-masing kelas ditunjukkan pada Kode Sumber 3.3.

```
import os
import shutil
import pandas as pd

source_folder = '/content/train'
destination_folder = '/content/train_structured'

for index, row in train_df.iterrows():
    image_filename = row['Image']
    label = row['Label']

    source_path = os.path.join(source_folder,
        image_filename)

    destination_label_folder = os.path.join(
        destination_folder, label)
    os.makedirs(destination_label_folder, exist_ok=True)

    destination_path = os.path.join(
        destination_label_folder, image_filename)

    shutil.move(source_path, destination_path)
```

**Kode Sumber 3.3** Strukturisasi Dataset Sesuai Kelas

Dataset yang telah dikumpulkan memiliki 102 kelas, dengan jumlah seluruh data gambar adalah 8189. Detail nama kelas pada dataset beserta jumlah data latihnya ditunjukkan pada Tabel 3.1.

### 3.4.3 Implementasi Data *Preprocessing*

Data *Preprocessing* bertujuan untuk memperbanyak keragaman data yang dilatih tanpa mengumpulkan data baru. *Preprocessing* dilakukan dengan augmentasi dataset menggunakan *ImageDataGenerator* yang dimiliki Keras. Teknik proses yang digunakan pada augmentasi data adalah membuat nilai *pixel* pada setiap gambar dibuat dengan skala 1/255, merotasi gambar secara acak dengan rotasi maksimal sebesar 30 derajat, menggeser gambar secara vertikal dan horizontal sebesar -20% sampai 20%, distorsi pada sumbu x dan y sampai 20%, *zoom* sampai perbesaran 20%, dan membalikkan gambar secara horizontal. Implementasi data *preprocessing* dengan augmentasi data ditunjukkan pada Kode Sumber 3.4.

```
from keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(rescale=1./255,
                                   rotation_range=30,
                                   width_shift_range=0.2,
                                   height_shift_range
                                       =0.2,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True,
                                   fill_mode='nearest')
```

**Kode Sumber 3.4** Augmentasi Data

Gambar orisinil terdiri dari koefisien RGB (*Red*, *Green*, *Blue*) dalam rentang [0,255]. Nilai pada rentang tersebut sangat besar untuk diproses oleh model. Untuk mengatasi hal tersebut, rentang piksel pada gambar diperkecil. Pada kode sumber baris ke-1, piksel yang sebelumnya pada rentang [0,255] dikonversi menjadi rentang [0,1]. Proses ini disebut juga dengan normalisasi input. Berikutnya pada kode sumber baris ke-2, gambar dirotasi secara acak dengan sudut rotasi maksimal 30 derajat. Pada sumber kode baris ke-3, gambar digeser secara horizontal dengan parameter *width_shift_range* sebesar -20% sampai 20%, dan pada sumber kode baris ke-4, gambar digeser secara vertikal dengan parameter *height_shift_range* sebesar -20% sampai 20%. Hal ini dilakukan karena objek gambar yang akan diambil tidak selalu berada di tengah, maka dari itu gambar digeser untuk mengatasi masalah tersebut. Pada sumber kode baris ke-5, gambar distorsi pada sumbu x dan y dengan parameter *shear_range*. Hal ini dilakukan karena objek gambar akan diambil dari sudut yang bervariasi. Nilai 0.2 pada parameter *shear_range* menentukan besaran distorsi adalah sampai 20%. Pada sumber kode baris ke-6, gambar dilakukan pembesaran sampai 20%. Hal ini dilakukan untuk menangani ukuran objek yang tertangkap kamera bervariasi. Parameter yang digunakan untuk pembesaran gambar adalah *zoom_range*. Pada sumber kode baris ke-7, gambar dibalikkan secara horizontal untuk memperbanyak data latih. Parameter yang digunakan untuk membalikkan gambar adalah *horizontal_flip* dengan nilai yang dimasukkan adalah *True*. Ketika semua parameter tersebut digabungkan, hasil variasi gambar dari data augmentasi ditampilkan pada Gambar 3.20.

**Gambar 3.20** Visualisasi Gambar Setelah Augmentasi Data

### 3.4.4 Implementasi Menggunakan *Pre-Trained* Model

Untuk menggunakan *Pre-Trained* Model, model perlu di-*import* terlebih dahulu dari Keras. Pada Tugas Akhir ini, model yang digunakan adalah MobileNetV2, InceptionV3, dan Xception. Model yang digunakan hanya dasarnya saja, dan pada lapisan akhirnya dibuat perubahan untuk menyesuaikan dengan jumlah kelas yang ada. Parameter yang digunakan ada bentuk input di mana ukuran yang digunakan adalah 224×224 dan 299×299 dengan 3 bentuk warna RGB (*Red*, *Green*, *Blue*), dan meninggalkan lapisan *output* dari *pre-trained* model. Lapisan pada model dasar dibuat agar tidak melatih dengan *non_trainable*, setiap *layer* pada *pre-trained* model dimasukkan nilai *False* pada parameter *layer.trainable*. Implementasi

penggunaan *pre-trained* model ditunjukkan pada Kode Sumber 3.5, Kode Sumber 3.6, dan Kode Sumber 3.7.

```
from tensorflow.keras.applications import MobileNetV2

base_model = MobileNetV2(input_shape=(224, 224, 3),
                         include_top=False,
                         weights='imagenet')

for layer in base_model.layers:
    layer.trainable = False
```

**Kode Sumber 3.5** Menggunakan *Pre-Trained* Model MobileNetV2

```
from tensorflow.keras.applications import InceptionV3

base_model = InceptionV3(input_shape=(299, 299, 3),
                         include_top=False,
                         weights='imagenet')

for layer in base_model.layers:
    layer.trainable = False
```

**Kode Sumber 3.6** Menggunakan *Pre-Trained* Model InceptionV3

```
from tensorflow.keras.applications import Xception

base_model = Xception(input_shape=(299, 299, 3),
                      include_top=False,
                      weights='imagenet')

for layer in base_model.layers:
    layer.trainable = False
```

**Kode Sumber 3.7** Menggunakan *Pre-Trained* Model Xception

### 3.4.5 Implementasi Membuat Lapisan Akhir

Pada lapisan akhir, diawali dengan GlobalAveragePooling2D *layer* yang berfungsi untuk mengubah input multidimensi menjadi satu dimensi dan menghubungkan lapisan konvolusi ke lapisan terhubung penuh. Berikutnya diikuti dengan *layer* yang bervariasi untuk setiap percobaan *pre-trained* model. Pada percobaan menggunakan *pre-trained* model, setelah dengan GlobalAveragePooling2D layer diikuti dengan lapisan *dense* akhir yang menggunakan aktivasi *softmax*. Dengan model MobileNetV2 yang ditambahkan dengan konfigurasi lapisan seperti ini pada Kode Sumber 3.8 Penambahan Lapisan pada Model.

```
from tensorflow.keras import layers, models
model = models.Sequential()
model.add(base_model)
model.add(layers.GlobalAveragePooling2D())
model.add(layers.Dense(102, activation='softmax'))
```

**Kode Sumber 3.8** Penambahan Lapisan pada Model

Model kemudian dikompilasi dengan adam *optimizer*, loss *function Categorical Cross Entropy* untuk klasifikasi multikelas, dan *acc* sebagai metrik akurasi. Kompilasi model ditunjukkan pada Kode Sumber 3.9.

```
model.compile(optimizer='adam', loss='
    categorical_crossentropy', metrics=['accuracy'])
```

**Kode Sumber 3.9** Kompilasi Model

### 3.4.6 Implementasi Meluncurkan Kamera dan Galeri

Untuk meluncurkan kamera pada Android, digunakan fungsi *TakePicturePreview()* yang akan disediakan Kotlin. Fungsi tersebut membantu meluncurkan kamera untuk mengambil foto yang akan disimpan dalam bentuk *bitmap*. Untuk memeriksa apakah *user* mengizinkan aplikasi untuk mengakses kamera, digunakan fungsi *checkSelfPermission()*. Jawaban dari *user* mengenai permohonan akses diterima pada *Manifest.permission.CAMERA*, yang juga merupakan parameter dari fungsi *checkSelfPermission()*. Kode Sumber 3.10 menunjukkan implementasi memeriksa apakah user mengizinkan aplikasi untuk mengakses kamera dan juga meluncurkan kamera pada perangkat Android.

```
import android.content.pm.PackageManager
import androidx.core.content.ContextCompat

button.setOnClickListener {
    if (ContextCompat.checkSelfPermission(this, android.
        Manifest.permission.CAMERA)
        == PackageManager.PERMISSION_GRANTED
    ) {
        TakePicturePreview.launch(null)
    } else {
        requestPermission.launch(android.Manifest.
            permission.CAMERA)
    }
}
    private val TakePicturePreview =
        registerForActivityResult(ActivityResultContracts.
        TakePicturePreview()){ bitmap ->
        if(bitmap != null){
            imageView.setImageBitmap(bitmap)
            outputGenerator(bitmap)
        }
    }
```

**Kode Sumber 3.10** Meluncurkan Kamera di Perangkat Keras

Pada Kode Sumber 3.10 adalah fungsi untuk meluncurkan kamera. Fungsi tersebut akan terpanggil ketika pengguna menekan tombol "*Take Image*". Pada fungsi tersebut, aplikasi akan memeriksa apakah pengguna sudah memberikan akses untuk kamera atau belum. Jika sudah, aplikasi akan meluncurkan kamera dan hasil foto yang tertangkap kamera akan tersimpan dalam bentuk bitmap. Selanjutnya akan memanggil fungsi *outputGenerator* untuk mengklasifikasi gambar.

Untuk meluncurkan galeri pada perangkat anndroid, digunakan *intent Images.Media.EXTERNAL_CONTENT_URI* yang disediakan MediaStore. *Intent* tersebut untuk mengakses galeri dari perangkat lunak. Untuk memeriksa apakah user mengizinkan aplikasi untuk mengakses galeri, digunakan fungsi *checkSelfPermission()*. Jawaban dari *user* mengenai permohonan akses diterima pada *Manifest.permission. READ_MEDIA_IMAGES*, yang juga merupakan parameter dari fungsi *checkSelfPermission()*. Implementasi untuk memeriksa apakah *user* mengizinkan aplikasi untuk mengakses galeri dan juga meluncurkan galeri pada perangkat Android ditunjukkan pada Kode Sumber 3.11.

```
import android.content.Intent
import android.content.pm.PackageManager
import android.provider.MediaStore
import android.util.Log
import androidx.core.content.ContextCompat

buttonLoad.setOnClickListener {
    if (ContextCompat.checkSelfPermission(this, android.
        Manifest.permission.READ_MEDIA_IMAGES)
        == PackageManager.PERMISSION_GRANTED
    ) {
        Log.i("TAG", "Masuk")
        val intent = Intent(Intent.ACTION_PICK,
            MediaStore.Images.Media.EXTERNAL_CONTENT_URI)
        intent.type = "image/*"
        val mimeTypes = arrayOf("image/jpeg", "image/jpg"
            , "image/png")
        intent.putExtra(Intent.EXTRA_MIME_TYPES,
            mimeTypes)
        intent.flags = Intent.
            FLAG_GRANT_READ_URI_PERMISSION
        onresult.launch(intent)
    } else {
        requestPermission.launch(android.Manifest.
            permission.READ_MEDIA_IMAGES)
    }
}

imageView.setOnLongClickListener {
    requestPermissionLauncer.launch(android.Manifest.
        permission.WRITE_EXTERNAL_STORAGE)
    return@setOnLongClickListener true
}
```

**Kode Sumber 3.11** Meluncurkan Kamera di Perangkat Keras

Pada Kode Sumber 3.11 adalah fungsi untuk meluncurkan galeri. Fungi tersebut akan terpanggil ketika *user* menekan tombol "*Load Image*". Pada fungsi tersebut, aplikasi memeriksa apakah *user* sudah memberikan izin aplikasi untuk mengakses galeri atau belum. Jika sudah, aplikasi akan meluncurkan galeri, jika *user* belum memberikan izin aplikasi untuk mengakses galeri, aplikasi akan mengirim pesan meminta izin untuk mengakses galeri.

### 3.4.7 Implementasi *Preprocessing Image*

Gambar yang akan diinput pada model perlu dilakukan *preprocessing* agar ukurannya sesuai dengan input model. Implementasi Image *Preprocessing* pada Android ditunjukkan Kode Sumber 3.16. Kelas *Bitmap* adalah jenis organisasi memori atau format file gambar yang digunakan untuk menyimpan gambar digital. bitmap asli diubah ukurannya menjadi 224×224 piksel menggunakan *Bitmap.createScaledBitmap*(*bitmap*, 224, 224, *true*). Ukuran 224×224 ini menyesuaikan untuk input gambar sesuai dengan model yang dipilih dalam hal ini model MobileNetV2, untuk Xception dan InceptionV3 akan menggunakan ukuran 299×299. Setelah diubah ukurannya, bitmap tersebut dikonversi menjadi objek *ByteBuffer* melalui fungsi *convertBitmapToByteBuffer* yang mengambil dua parameter: *bitmap* yang telah diubah ukurannya dan ukuran (224 dalam hal ini).

```
import android.graphics.Bitmap
import android.graphics.Color
import java.nio.ByteBuffer
import java.nio.ByteOrder

val resizedbitmap224 = Bitmap.createScaledBitmap(bitmap,
    224, 224, true)

val bytebuffer224 = convertBitmapToByteBuffer(
    resizedbitmap224, 224)

private fun convertBitmapToByteBuffer(bitmap: Bitmap,
    size: Int): ByteBuffer {
    val byteBuffer = ByteBuffer.allocateDirect(4 * size *
        size * 3)
    byteBuffer.order(ByteOrder.nativeOrder())

    val intValues = IntArray(size * size)
    bitmap.getPixels(intValues, 0, bitmap.width, 0, 0,
        bitmap.width, bitmap.height)

    var pixelIndex = 0
    for (i in 0 until size) {
        for (j in 0 until size) {
            val pixelValue = intValues[pixelIndex++]

            byteBuffer.putFloat((Color.red(pixelValue) -
                127.5f) / 127.5f)
            byteBuffer.putFloat((Color.green(pixelValue)
                - 127.5f) / 127.5f)
            byteBuffer.putFloat((Color.blue(pixelValue) -
                127.5f) / 127.5f)
        }
    }

    return byteBuffer
}
```

**Kode Sumber 3.12** *Image Preprocessing*

Dalam fungsi *convertBitmapToByteBuffer*, *ByteBuffer* dialokasikan dengan ukuran yang cukup untuk menampung data gambar dalam format RGB dengan tipe data *float* (4 byte per *channel*). *ByteBuffer* ini diatur dalam urutan *byte* asli perangkat (*ByteOrder.nativeOrder*()). Selanjutnya, piksel dari bitmap diambil dan disimpan dalam array intValues. Setiap piksel kemudian diekstraksi nilai RGB-nya dan dinormalisasi dengan mengurangi 127.5 dan membaginya dengan 127.5. Nilai-nilai yang dinormalisasi ini dimasukkan ke dalam *ByteBuffer* sebagai nilai *float*. Proses normalisasi ini bertujuan untuk menskalakan nilai piksel ke dalam rentang [-1, 1], untuk meningkatkan performa prediksi. Akhirnya, *ByteBuffer* yang berisi data gambar yang telah diproses dikembalikan untuk digunakan dalam tahap inferensi model.

### 3.4.8 Implementasi Klasifikasi Gambar

Setelah meng-*import* file TFLite ke dalam *project* Android, pada file TFLite tersedia kode sampel yang bisa digunakan untuk mengklasifikasi gambar dengan TFLite, kode sampel yang terdapat pada berkas TFLite ditunjukkan pada Kode Sumber 3.13 untuk MobileNetV2, Kode Sumber 3.14 untuk InceptionV3, dan Kode Sumber 3.15 untuk Xception.

```
import org.tensorflow.lite.support.image.TensorImage
import org.tensorflow.lite.support.tensorbuffer.
    TensorBuffer
import org.tensorflow.lite.DataType
import org.tensorflow.lite.task.vision.detector.
    ObjectDetector

val model = MobileNetV2.newInstance(context)

// Creates inputs for reference.
val inputFeature0 = TensorBuffer.createFixedSize(
    intArrayOf(1, 224, 224, 3), DataType.FLOAT32)
inputFeature0.loadBuffer(byteBuffer)

// Runs model inference and gets result.
val outputs = model.process(inputFeature0)
val outputFeature0 = outputs.outputFeature0AsTensorBuffer

// Releases model resources if no longer used.
model.close()
```

**Kode Sumber 3.13** Kode Sampel MobileNetV2

```
import org.tensorflow.lite.support.image.TensorImage
import org.tensorflow.lite.support.tensorbuffer.
    TensorBuffer
import org.tensorflow.lite.DataType

val model = InceptionV3.newInstance(context)

// Creates inputs for reference.
val inputFeature0 = TensorBuffer.createFixedSize(
    intArrayOf(1, 299, 299, 3), DataType.FLOAT32)
inputFeature0.loadBuffer(byteBuffer)

// Runs model inference and gets result.
val outputs = model.process(inputFeature0)
val outputFeature0 = outputs.outputFeature0AsTensorBuffer

// Releases model resources if no longer used.
model.close()
```

**Kode Sumber** 3.14 Kode Sampel InceptionV3

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

val model = Xception.newInstance(context)

// Creates inputs for reference.
val inputFeature0 = TensorBuffer.createFixedSize(
    intArrayOf(1, 299, 299, 3), DataType.FLOAT32)
inputFeature0.loadBuffer(byteBuffer)

// Runs model inference and gets result.
val outputs = model.process(inputFeature0)
val outputFeature0 = outputs.outputFeature0AsTensorBuffer

// Releases model resources if no longer used.
model.close()
```

**Kode Sumber** 3.15 Kode Sampel Xception

Sampel kode tersebut kemudian dimasukkan ke dalam fungsi *generateOutput()*, yaitu merupakan fungsi di mana proses klasifikasi dilakukan. Salah satu implementasi untuk klasifikasi gambar menggunakan TFLite MobileNetV2 ditunjukkan pada Kode Sumber 3.16.

```
import android.graphics.Bitmap
import org.tensorflow.lite.support.tensorbuffer.
    TensorBuffer
import org.tensorflow.lite.DataType

private fun outputGenerator(bitmap: Bitmap) {
    val resizedbitmap224 = Bitmap.createScaledBitmap(
        bitmap, 224, 224, true)
    val bytebuffer224 = convertBitmapToByteBuffer(
        resizedbitmap224, 224)

    val modelMobilenetv2 = MobileNetV2.newInstance(this)

    val inputFeatureMobilenetv2 = TensorBuffer.
        createFixedSize(intArrayOf(1, 224, 224, 3),
        DataType.FLOAT32)
    inputFeatureMobilenetv2.loadBuffer(bytebuffer224)

    val outputsMobilenetv2 = modelMobilenetv2.process(
        inputFeatureMobilenetv2)
    val outputFeatureMobilenetv2 = outputsMobilenetv2.
        outputFeature0AsTensorBuffer

    val outputArrayMobilenetv2 = outputFeatureMobilenetv2
        .floatArray
    val (bestValueMobilenetv2, bestIndexMobilenetv2) =
        getBestPrediction(outputArrayMobilenetv2)
    val bestWordMobilenetv2 = allWords.elementAt(
        bestIndexMobilenetv2)
    val formattedValueMobilenetv2 = "%.2f".format(
        bestValueMobilenetv2)

    modelMobilenetv2.close()

    tvoutput.text = "${bestWordMobilenetv2} - ${
        formattedValueMobilenetv2}"
}
```

**Kode Sumber 3.16** Klasifikasi Gambar Menggunakan TFLite

*TensorBuffer* adalah kelas yang mengambil bentuk *array* yang diinginkan dan tipe datanya. Variabel untuk menampung data input dan *output* adalah *inputFeature0* dan *outputFeature0*. Parameter uang digunakan *inputFeature0* untuk memuat data ke model berupa nilai piksel. Nilai piksel yang sudah dikecilkan kemudian dimuat ke *byteBuffer*, lalu dipakai sebagai parameter untuk melakukan proses pada model. Hasil *output* model disimpan di variabel *outputFeature0* dalam bentuk *TensorBuffer*.

### 3.4.9 Implementasi Menampilkan Hasil Klasifikasi

*Output* dari proses klasifikasi adalah berupa nilai akurasi pada setiap kelas, yang disimpan pada variabel *outputFeature()*. Untuk menampung hasil *output*, dibuatkan sebuah variabel *confidence* dengan bentuk *float array*. Nilai yang dimasukkan ke variabel *confidences* adalah hasil *output* dari model yang diambil dalam bentuk *float array*. Untuk mengambil hasil *output* model dalam bentuk *float array* variabel *outputFeature0* yang merupakan nilai *output* dari model dioperasikan dengan fungsi *outputFeature0AsTensorBuffer()*, yaitu fungsi untuk mendapatkan nilai dalam bentuk *float array*. Setelah hasil disimpan, berikutnya akan memanggil fungsi *getBestPrediction()* di mana dilakukan iterasi pada *array confidence* untuk menemukan indeks dengan nilai tertinggi. Setiap indeks pada *array* merujuk pada nama bunga, dengan urutan bunga adalah berdasarkan urutan huruf awal dari kecil ke besar. Kode Sumber 3.17 menunjukkan implementasi untuk menampilkan hasil klasifikasi bunga.

```
import android.graphics.Bitmap
import org.tensorflow.lite.support.tensorbuffer.
    TensorBuffer
import org.tensorflow.lite.DataType

private fun outputGenerator(bitmap: Bitmap) {
    ...

    val outputsMobilenetv2 = modelMobilenetv2.process(
        inputFeatureMobilenetv2)
    val outputFeatureMobilenetv2 = outputsMobilenetv2.
        outputFeature0AsTensorBuffer

    val outputArrayMobilenetv2 = outputFeatureMobilenetv2
        .floatArray
    val (bestValueMobilenetv2, bestIndexMobilenetv2) =
        getBestPrediction(outputArrayMobilenetv2)
    val bestWordMobilenetv2 = allWords.elementAt(
        bestIndexMobilenetv2)

    val formattedValueMobilenetv2 = "%.2f".format(
        bestValueMobilenetv2)
    tvoutput.text = "${bestWordMobilenetv2} ${
        formattedValueMobilenetv2}"
    ...
}

private fun getBestPrediction(outputArray: FloatArray):
    Pair<Float, Int> {
    var bestValue = outputArray[0]
    var bestIndex = 0

    for (i in outputArray.indices) {
        if (outputArray[i] > bestValue) {
            bestValue = outputArray[i]
            bestIndex = i
        }
    }

    return Pair(bestValue, bestIndex)
}
```

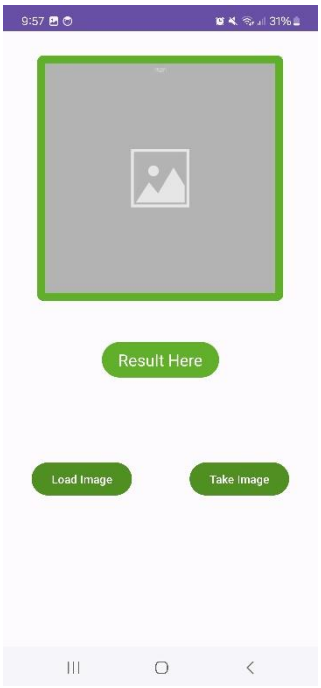
**Kode Sumber 3.17** Menampilkan Hasil Klasifikasi

### 3.4.10 Implementasi Antarmuka

Berikut merupakan hasil dari implementasi antarmuka aplikasi untuk tugas akhir ini. Terdapat dua implementasi antarmuka, yaitu antarmuka menu utama dan antarmuka hasil klasifikasi.

#### 3.4.10.1 Antarmuka Menu Utama

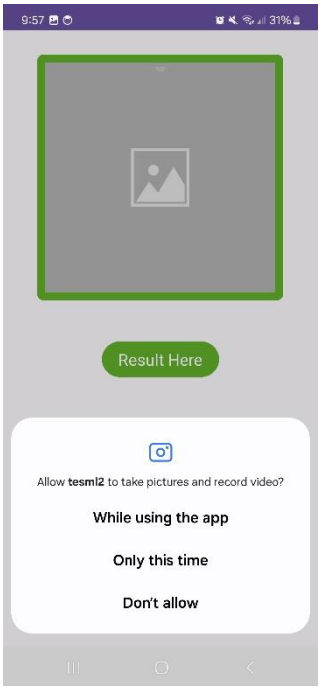
Pada halaman menu utama terdapat *placeholder* yang digunakan sebagai tempat foto yang akan diklasifikasi ditambahkan dua tombol yaitu tombol "*Load Image*" yang digunakan untuk memilih foto dari galeri dan tombol "*Take Image*" yang digunakan menangkap foto melalui kamera. Selain itu juga terdapat *textfield* yang nantinya akan menampilkan hasil klasifikasi berupa nama bunga dan nilai *confidence*. Tampilan antarmuka menu utama aplikasi ditunjukkan pada Gambar 3.21.




**Gambar 3.21** Antarmuka Menu Utama

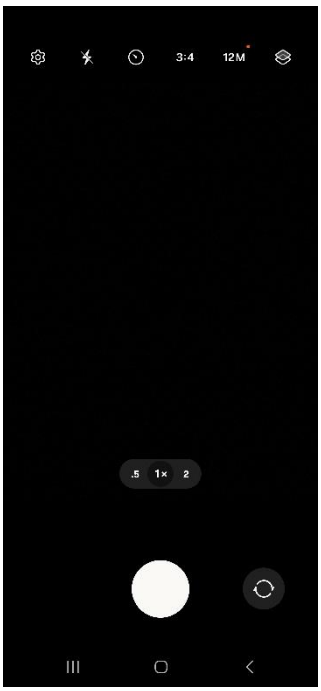
#### 3.4.10.2 Antarmuka Hasil Klasifikasi

Antarmuka Hasil Klasifikasi muncul saat *user* menekan tombol "*Take Image*" aplikasi akan membuka kamera pada perangkat keras, dengan syarat *user* telah mengizinkan aplikasi untuk mengakses kamera pada perangkatnya. Apabila *user* belum memberi izin aplikasi untuk mengakses kamera, aplikasi akan menampilkan pertanyaan *request* untuk mengakses fitur kamera. Tampilan *request* untuk mengakses kamera perangkat ditunjukkan pada Gambar 3.22.



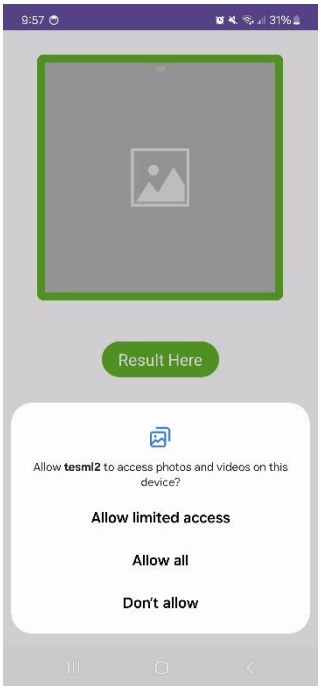

**Gambar 3.22** Tampilan *Request* untuk Mengakses Kamera

Tampilan fitur kamera akan pada perangkat Android yang diakses oleh aplikasi setelah *user* menekan tombol "*Take Image*". Contoh tampilan saat mengakses fitur kamera ditunjukkan pada Gambar 3.23



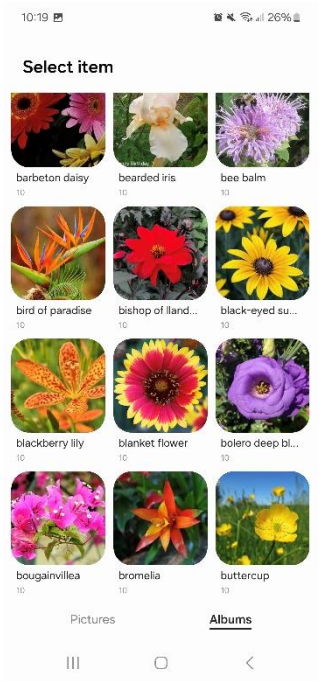

**Gambar 3.23** Contoh Tampilan Saat Mengakses Fitur Kamera

Saat *user* memencet tombol "*Load Image*" aplikasi akan membuka galeri pada perangkat, dengan syarat *user* telah mengizinkan aplikasi untuk mengakses galeri pada perangkatnya. Apabila *user* belum memberi izin aplikasi untuk mengakses galeri, aplikasi akan menampilkan pertanyaan *request* untuk mengakses fitur galeri. Tampilan *request* untuk mengakses galeri pada perangkat ditunjukkan pada gambar Gambar 3.24




**Gambar 3.24** Tampilan *Request* untuk Mengakses Kamera

Setelah menekan tombol "*Load Image*" dan memberikan akses ke galeri aplikasi akan membuka fitur galeri pada perangkat, salah satu contoh tampilannya ditunjukkan pada Gambar 3.25. Di sini user dapat memilih gambar manapun yang terdapat pada galeri untuk diklasifikasi.




**Gambar 3.25** Contoh Tampilan Saat Mengakses Fitur Galeri

Setelah *user* mengambil atau memilih gambar untuk diklasifikasi, aplikasi akan berpindah ke halaman hasil klasifikasi. Tampilan dari implementasi hasil klasifikasi bunga pada aplikasi ditunjukkan pada Gambar 3.26.


**Gambar 3.26** Antarmuka Hasil Klasifikasi Bunga

Pada antarmuka klasifikasi bunga, termuat komponen foto bunga yang diambil untuk diklasifikasi, nama bunga hasil klasifikasi, dan diikuti nilai *confidence* hasil klasifikasi. Foto yang ditampilkan pada halaman tersebut adalah foto yang diambil atau dipilih oleh *user* untuk diklasifikasi. Di bawahnya, terdapat dua tombol yaitu tombol "*Load Image*" dan "*Take Image*" dengan posisi *fixed* di bawah komponen sebelumnya.

# BAB IV HASIL DAN PEMBAHASAN

## 4.1 Lingkungan Uji Coba

Lingkungan uji coba yang digunakan pada Tugas Akhir ini memakai perangkat laptop ASUS TUF Gaming A15 FA506IH dengan sistem operasi Windows 11 64-bit. Laptop yang digunakan memiliki spesifikasi processor AMD Ryzen™ 7 4800H Mobile Processor (8-core/16-thread, 12MB Cache, 4.2 GHz max boost) dengan Random Access Memory (RAM) sebesar 32 GB. Spesifikasi detail dapat dilihat pada Tabel 4.1.

**Tabel 4.1** Kebutuhan Perangkat Keras

| Perangkat keras | Spesifikasi |
|---|---|
| **Sistem Operasi** | *Windows 11* |
| ***Central Processing Unit (CPU)*** | *AMD Ryzen™ 7 4800H Mobile Processor (8-core/16-thread, 12MB Cache, 4.2 GHz max boost)* |
| ***Graphics Processing Unit (GPU)*** | *NVIDIA® GeForce GTX™ 1650, 4GB GDDR6* |
| ***Memory (RAM)*** | *32.0 GB* |
| ***Internal Memory*** | *1.5 TB* |

Selain itu adapun memakai perangkat keras bergerak Samsung A55 5G dengan sistem operasi Android 14 untuk melakukan proses uji coba dari fitur aplikasi yang telah dibangun sebelumnya. Spesifikasi dari perangkat keras bergerak yang digunakan ditunjukkan pada Tabel 4.2.

**Tabel 4.2** Kebutuhan Perangkat Keras Bergerak

| Perangkat keras | Spesifikasi |
|---|---|
| **Sistem Operasi** | *Android 14* |
| ***Central Processing Unit (CPU)*** | *Exynos 1480 (5G)* |
| ***Memory (RAM)*** | *12.0 GB* |
| ***Internal Memory*** | *256.0 GB* |

## 4.2 Hasil Eksperimen

### 4.2.1 Pengujian Training *Pre-Trained* Model

Untuk mendapatkan model dengan akurasi yang terbaik, dilakukan uji coba *training* pada tiga *pre-trained* model. *Pre-Trained* model yang dicoba adalah InceptionV3, MobileNetV2, dan Xception. Pada setiap percobaan *training* model, dilakukan berbagai percobaan *transfer learning* dengan melakukan perubahan pada lapisan terakhir untuk dicatat evaluasi akurasi dan *loss* pada setiap *pre-trained* model, dan untuk mendapatkan akurasi tertinggi dari setiap model. Performa model tiap *epoch* dapat dilihat pada tabel 4.3.

**Tabel 4.3** Performa Model tiap *Epoch*

| ***Epoch*** | **MobilenetV2** | | | **Xception** | | | **InceptionV3** | | |
|---|---|---|---|---|---|---|---|---|---|
| | ***Accuracy*** | ***Loss*** | **Waktu Pelatihan** | ***Accuracy*** | ***Loss*** | **Waktu Pelatihan** | ***Accuracy*** | ***Loss*** | **Waktu Pelatihan** |
| 1 | 0.498 | 2.327 | 104 detik | 0.435 | 2.835 | 200 detik | 0.432 | 2.704 | 207 detik |

| *Epoch* | MobilenetV2 | | | Xception | | | InceptionV3 | | |
|---|---|---|---|---|---|---|---|---|---|
| | *Accuracy* | *Loss* | Waktu Pelatihan | *Accuracy* | *Loss* | Waktu Pelatihan | *Accuracy* | *Loss* | Waktu Pelatihan |
| 2 | 0.805 | 0.890 | 97 detik | 0.745 | 1.373 | 180 detik | 0.736 | 1.222 | 187 detik |
| 3 | 0.866 | 0.600 | 95 detik | 0.825 | 0.934 | 190 detik | 0.812 | 0.844 | 183 detik |
| 4 | 0.890 | 0.483 | 94 detik | 0.852 | 0.756 | 180 detik | 0.838 | 0.686 | 186 detik |
| 5 | 0.908 | 0.399 | 95 detik | 0.878 | 0.626 | 188 detik | 0.859 | 0.585 | 183 detik |
| 6 | 0.922 | 0.339 | 96 detik | 0.897 | 0.534 | 181 detik | 0.883 | 0.487 | 184 detik |
| 7 | 0.925 | 0.304 | 96 detik | 0.905 | 0.479 | 178 detik | 0.891 | 0.438 | 182 detik |
| 8 | 0.938 | 0.260 | 98 detik | 0.915 | 0.424 | 179 detik | 0.900 | 0.406 | 181 detik |
| 9 | 0.945 | 0.238 | 94 detik | 0.922 | 0.388 | 179 detik | 0.911 | 0.364 | 186 detik |
| 10 | 0.948 | 0.220 | 95 detik | 0.926 | 0.355 | 189 detik | 0.913 | 0.338 | 181 detik |
| 11 | 0.949 | 0.207 | 94 detik | 0.935 | 0.331 | 180 detik | 0.924 | 0.310 | 183 detik |
| 12 | 0.952 | 0.191 | 95 detik | 0.941 | 0.302 | 188 detik | 0.927 | 0.288 | 183 detik |
| 13 | 0.960 | 0.175 | 97 detik | 0.944 | 0.280 | 180 detik | 0.932 | 0.275 | 183 detik |
| 14 | 0.963 | 0.160 | 95 detik | 0.942 | 0.274 | 179 detik | 0.940 | 0.248 | 181 detik |
| 15 | 0.965 | 0.147 | 94 detik | 0.949 | 0.246 | 190 detik | 0.939 | 0.232 | 185 detik |
| 16 | 0.962 | 0.149 | 95 detik | 0.955 | 0.227 | 179 detik | 0.939 | 0.231 | 183 detik |
| 17 | 0.970 | 0.126 | 94 detik | 0.958 | 0.214 | 179 detik | 0.943 | 0.214 | 187 detik |
| 18 | 0.971 | 0.125 | 101 detik | 0.957 | 0.206 | 178 detik | 0.947 | 0.206 | 185 detik |
| 19 | 0.968 | 0.131 | 101 detik | 0.959 | 0.199 | 177 detik | 0.942 | 0.209 | 188 detik |
| 20 | 0.971 | 0.119 | 100 detik | 0.963 | 0.185 | 178 detik | 0.949 | 0.189 | 183 detik |
| 21 | 0.972 | 0.113 | 105 detik | 0.963 | 0.179 | 187 detik | 0.950 | 0.183 | 188 detik |
| 22 | 0.973 | 0.110 | 97 detik | 0.963 | 0.177 | 179 detik | 0.950 | 0.186 | 183 detik |
| 23 | 0.974 | 0.103 | 96 detik | 0.968 | 0.165 | 187 detik | 0.953 | 0.173 | 184 detik |
| 24 | 0.977 | 0.095 | 95 detik | 0.971 | 0.157 | 178 detik | 0.955 | 0.172 | 186 detik |
| 25 | 0.977 | 0.094 | 95 detik | 0.976 | 0.146 | 183 detik | 0.959 | 0.152 | 185 detik |
| 26 | 0.980 | 0.084 | 94 detik | 0.972 | 0.146 | 181 detik | 0.955 | 0.158 | 184 detik |
| 27 | 0.980 | 0.089 | 95 detik | 0.972 | 0.141 | 182 detik | 0.964 | 0.143 | 185 detik |
| 28 | 0.977 | 0.087 | 95 detik | 0.972 | 0.135 | 185 detik | 0.961 | 0.142 | 185 detik |
| 29 | 0.975 | 0.093 | 95 detik | 0.974 | 0.128 | 184 detik | 0.964 | 0.139 | 183 detik |
| 30 | **0.980** | **0.084** | 93 detik | **0.974** | **0.127** | 182 detik | **0.964** | **0.136** | 185 detik |

Tabel 4.3 menunjukkan akurasi dan *loss*, dan waktu pelatihan pada 3 *pre-trained* model di setiap *epoch*. Secara keseluruhan, akurasi dan *loss* terbaik ada pada epoch 30. Waktu pelatihan yang membutuhkan waktu paling lama ada pada model InceptionV3 sementara waktu pelatihan tercepat adalah saat menggunakan model MobileNetV2.

Hasil training model dengan susunan lapisan akhir seperti pada Kode Sumber 3.8, untuk model MobileNetV2 didapatkan nilai *loss* sebesar 0.0837 dan akurasi sebesar 0.9799. Lalu untuk model InceptionV3 mendapatkan nilai *loss* sebesar 0.1359 dan akurasi sebesar 0.9641, Dan untuk model Xception mendapatkan nilai *loss* sebesar 0.9744. Hasil evaluasi akurasi tertinggi untuk setiap *pre-trained* model ditunjukkan pada Tabel 4.4.

**Tabel 4.4** Skor *Loss* dan *Accuracy* Setiap *Pre-Trained* Model

| ***Pre-Trained Model*** | ***Accuracy*** | ***Loss*** |
|---|---|---|
| MobileNetV2 | 0.9799 | 0.0837 |
| InceptionV3 | 0.9641 | 0.1359 |
| Xception | 0.9744 | 0.1267 |

Dari Tabel 4.4 terlihat bahwa *pre-trained* model MobileNetV2 menghasilkan nilai akurasi tertinggi dan *loss value* terendah. Untuk mengetahui tingkat akurasi model pada setiap kelas, dilakukan uji coba klasifikasi pada setiap kelas bunga sebanyak 10 percobaan. Hasil percobaan pada setiap kelas bunga ditunjukkan pada Tabel 4.5.

**Tabel 4.5** Uji Coba Setiap Kelas Bunga Menggunakan Model

| **Bunga** | **Jumlah Data *Test*** | **Jumlah *True* pada MobilenetV2** | **Jumlah *True* pada InceptionV3** | **Jumlah *True* pada Xception** |
|---|---|---|---|---|
| alpine sea holly | 10 | 10 | 10 | 8 |
| anthurium | 10 | 8 | 8 | 10 |
| artichoke | 10 | 9 | 9 | 9 |
| azalea | 10 | 7 | 6 | 7 |
| ball moss | 10 | 7 | 8 | 9 |
| balloon flower | 10 | 7 | 9 | 9 |
| barbeton daisy | 10 | 3 | 7 | 7 |
| bearded iris | 10 | 7 | 10 | 7 |
| bee balm | 10 | 9 | 7 | 7 |
| bird of paradise | 10 | 10 | 10 | 9 |
| bishop of llandaff | 10 | 8 | 10 | 9 |
| black-eyed susan | 10 | 8 | 9 | 10 |
| blackberry lily | 10 | 10 | 10 | 9 |
| blanket flower | 10 | 10 | 8 | 9 |
| bolero deep blue | 10 | 8 | 8 | 9 |
| bougainvillea | 10 | 8 | 7 | 9 |
| bromelia | 10 | 10 | 10 | 10 |
| buttercup | 10 | 10 | 7 | 9 |

| Bunga | Jumlah Data *Test* | Jumlah *True* pada MobilenetV2 | Jumlah *True* pada InceptionV3 | Jumlah *True* pada Xception |
|---|---|---|---|---|
| californian poppy | 10 | 7 | 8 | 7 |
| camellia | 10 | 7 | 9 | 10 |
| canna lily | 10 | 7 | 6 | 4 |
| canterbury bells | 10 | 7 | 7 | 5 |
| cape flower | 10 | 8 | 10 | 9 |
| carnation | 10 | 10 | 10 | 9 |
| cautleya spicata | 10 | 3 | 8 | 4 |
| clematis | 10 | 7 | 6 | 9 |
| colt's foot | 10 | 9 | 9 | 10 |
| columbine | 10 | 8 | 6 | 9 |
| common dandelion | 10 | 9 | 10 | 8 |
| corn poppy | 10 | 10 | 7 | 9 |
| cyclamen | 10 | 6 | 7 | 7 |
| daffodil | 10 | 8 | 4 | 4 |
| desert-rose | 10 | 5 | 7 | 8 |
| english marigold | 10 | 8 | 7 | 8 |
| fire lily | 10 | 10 | 9 | 10 |
| foxglove | 10 | 9 | 9 | 10 |
| frangipani | 10 | 10 | 10 | 10 |
| fritillary | 10 | 9 | 8 | 10 |
| garden phlox | 10 | 9 | 7 | 5 |
| gaura | 10 | 10 | 10 | 9 |
| gazania | 10 | 9 | 8 | 9 |
| geranium | 10 | 8 | 5 | 7 |
| giant white arum lily | 10 | 9 | 8 | 10 |
| globe thistle | 10 | 10 | 9 | 9 |
| globe-flower | 10 | 9 | 8 | 9 |
| grape hyacinth | 10 | 10 | 10 | 10 |
| great masterwort | 10 | 10 | 10 | 10 |
| hard-leaved pocket orchid | 10 | 10 | 10 | 10 |
| hibiscus | 10 | 9 | 10 | 10 |
| hippeastrum | 10 | 8 | 9 | 10 |
| japanese anemone | 10 | 7 | 7 | 8 |

| Bunga | Jumlah Data *Test* | Jumlah *True* pada MobilenetV2 | Jumlah *True* pada InceptionV3 | Jumlah *True* pada Xception |
|---|---|---|---|---|
| king protea | 10 | 8 | 8 | 10 |
| lenten rose | 10 | 6 | 7 | 5 |
| lotus lotus | 10 | 7 | 8 | 8 |
| love in the mist | 10 | 8 | 8 | 10 |
| magnolia | 10 | 5 | 7 | 6 |
| mallow | 10 | 7 | 4 | 5 |
| marigold | 10 | 6 | 3 | 8 |
| mexican aster | 10 | 8 | 9 | 9 |
| mexican petunia | 10 | 8 | 8 | 9 |
| monkshood | 10 | 9 | 8 | 6 |
| moon orchid | 10 | 9 | 10 | 10 |
| morning glory | 10 | 8 | 10 | 8 |
| orange dahlia | 10 | 9 | 4 | 7 |
| osteospermum | 10 | 5 | 5 | 9 |
| oxeye daisy | 10 | 10 | 10 | 10 |
| passion flower | 10 | 10 | 10 | 10 |
| pelargonium | 10 | 8 | 5 | 3 |
| peruvian lily | 10 | 8 | 10 | 10 |
| petunia | 10 | 9 | 7 | 7 |
| pincushion flower | 10 | 9 | 9 | 10 |
| pink primrose | 10 | 9 | 10 | 7 |
| pink-yellow dahlia | 10 | 3 | 7 | 8 |
| poinsettia | 10 | 9 | 7 | 7 |
| primula | 10 | 8 | 8 | 8 |
| prince of wales feathers | 10 | 6 | 7 | 7 |
| purple coneflower | 10 | 10 | 10 | 10 |
| red ginger | 10 | 9 | 9 | 10 |
| rose | 10 | 10 | 10 | 8 |
| ruby-lipped cattleya | 10 | 9 | 8 | 9 |
| siam tulip | 10 | 5 | 7 | 10 |
| silverbush | 10 | 6 | 5 | 7 |
| snapdragon | 10 | 6 | 7 | 8 |
| spear thistle | 10 | 10 | 10 | 10 |

| Bunga | Jumlah Data *Test* | Jumlah *True* pada MobilenetV2 | Jumlah *True* pada InceptionV3 | Jumlah *True* pada Xception |
|---|---|---|---|---|
| spring crocus | 10 | 3 | 9 | 7 |
| stemless gentian | 10 | 10 | 9 | 8 |
| sunflower | 10 | 9 | 10 | 10 |
| sweet pea | 10 | 5 | 4 | 5 |
| sweet william | 10 | 8 | 8 | 8 |
| sword lily | 10 | 8 | 8 | 5 |
| thorn apple | 10 | 5 | 7 | 7 |
| tiger lily | 10 | 9 | 10 | 9 |
| toad lily | 10 | 10 | 10 | 9 |
| tree mallow | 10 | 6 | 8 | 9 |
| tree poppy | 10 | 10 | 9 | 10 |
| trumpet creeper | 10 | 8 | 6 | 5 |
| wallflower | 10 | 10 | 10 | 10 |
| water lily | 10 | 10 | 10 | 10 |
| watercress | 10 | 8 | 8 | 9 |
| wild pansy | 10 | 7 | 6 | 8 |
| windflower | 10 | 9 | 9 | 10 |
| yellow iris | 10 | 8 | 8 | 8 |

Pada pengujian klasifikasi setiap kelas bunga, hasil menunjukkan bahwa terdapat beberapa bunga yang memiliki akurasi yang berbeda-beda di setiap *pre-trained* model yang telah dilatih. Seperti pada bunga clematis pada MobileNetV2 mendapat jumlah benar 7 gambar, pada InceptionV3 mendapat jumlah benar 6 gambar, dan pada Xception mendapat jumlah benar 9 gambar.

### 4.2.2 Hasil Pengujian Aplikasi

Pengujian aplikasi berfokus pada pengujian kebutuhan fungsional untuk mendapatkan keluaran yang diharapkan dari kondisi masukan yang sesuai dengan menggunakan metode Black Box. Apliksai dilakukan uji coba kemudian keluaran akan diperiksa apakah sesuai dengan hasil yang diharapkan. Hasil uji coba dapat dilihat pada Tabel 4.6.

**Tabel 4.6** Hasil Uji Coba Aplikasi

| No | Skenario Pengujian | Testcase | Hasil Uji yang | Kesimpulan |
|---|---|---|---|---|
| 1 | Mengambil gambar dari kamera perangkat | Masukan:<br>- Menekan tombol "*Take Image*"<br>- Memposisikan kamera pada objek bunga | Gambar berhasil diambil. | Pengujian sukses |

| No | Skenario Pengujian | Testcase | Hasil Uji yang | Kesimpulan |
|---|---|---|---|---|
| | | - Menekan tombol "*Capture*" | | |
| 2 | Memilih gambar dari galeri perangkat | Masukan:<br>- Menekan tombol "*Load Image*"<br>- Memilih foto pada galeri<br>- Menekan tombol "*Choose*" | Gambar berhasil dipilih. | Pengujian sukses |
| 3 | Menjalankan proses klasifikasi | Masukan:<br>- Memilih foto yang digunakan<br>- Menekan tombol "*Choose*" atau tombol "*Capture*" | Proses klasifikasi berhasil berjalan. | Pengujian sukses |
| 4 | Melihat hasil klasifikasi | - | Berhasil menampilkan hasil klasifikasi | Pengujian sukses |

### 4.2.3 Pengujian Model pada Aplikasi

Setelah mengkonversi model ke dalam bentuk TFLite agar dapat diimplementasikan ke dalam aplikasi perangkat bergerak terdapat penurunan ukuran berkas model. Untuk penurunan ukuran berkas model yang dikonversi ke dalam TFLite dapat dilihat pada Tabel 4.7.

**Tabel 4.7** Ukuran Berkas Model

| Model | Format H5 | Format TFLite |
|---|---|---|
| MobileNetV2 | 10,4 MB | 8,95 MB |
| InceptionV3 | 86,1 MB | 83,8 MB |
| Xception | 82,2 MB | 80,1 MB |

Terdapat penurunan pada setiap model yang sudah di konvesi ke dalam TFLite, Seperti MobileNetV2 yang sebelumnya berukuran 10,4 MB menjadi 8,95 MB, InceptionV3 yang sebelumnya berukuran 86,1 MB menjadi 83,8 MB, dan Xception yang sebelumnya berukuran 82,2 MB menjadi 80,1 MB. Untuk mengetahui tingkat akurasi model yang sudah dikonversi ke dalam TFLite pada setiap kelas, dilakukan uji coba klasifikasi pada setiap kelas bunga sebanyak 10 percobaan. Hasil percobaan pada setiap kelas bunga ditunjukkan pada Tabel 4.8.

**Tabel 4.8** Uji Coba Setiap Kelas Bunga Menggunakan Model TFLite pada Aplikasi

| Bunga | Jumlah Data *Test* | Jumlah *True* pada MobilenetV2 | Jumlah *True* pada InceptionV3 | Jumlah *True* pada Xception |
|---|---|---|---|---|
| alpine sea holly | 10 | 10 | 8 | 10 |
| anthurium | 10 | 8 | 10 | 8 |

| Bunga | Jumlah Data *Test* | Jumlah *True* pada MobilenetV2 | Jumlah *True* pada InceptionV3 | Jumlah *True* pada Xception |
|---|---|---|---|---|
| artichoke | 10 | 9 | 9 | 9 |
| azalea | 10 | 7 | 7 | 6 |
| ball moss | 10 | 6 | 8 | 7 |
| balloon flower | 10 | 7 | 9 | 9 |
| barbeton daisy | 10 | 3 | 7 | 7 |
| bearded iris | 10 | 7 | 7 | 10 |
| bee balm | 10 | 9 | 7 | 7 |
| bird of paradise | 10 | 10 | 9 | 10 |
| bishop of llandaff | 10 | 8 | 9 | 10 |
| black-eyed susan | 10 | 8 | 10 | 9 |
| blackberry lily | 10 | 10 | 9 | 10 |
| blanket flower | 10 | 10 | 9 | 8 |
| bolero deep blue | 10 | 8 | 9 | 8 |
| bougainvillea | 10 | 8 | 9 | 7 |
| bromelia | 10 | 10 | 10 | 10 |
| buttercup | 10 | 10 | 9 | 7 |
| californian poppy | 10 | 7 | 7 | 8 |
| camellia | 10 | 7 | 10 | 9 |
| canna lily | 10 | 7 | 4 | 6 |
| canterbury bells | 10 | 7 | 5 | 7 |
| cape flower | 10 | 8 | 9 | 10 |
| carnation | 10 | 10 | 9 | 10 |
| cautleya spicata | 10 | 3 | 4 | 8 |
| clematis | 10 | 7 | 9 | 6 |
| colt's foot | 10 | 9 | 10 | 9 |
| columbine | 10 | 8 | 9 | 6 |
| common dandelion | 10 | 9 | 8 | 10 |
| corn poppy | 10 | 10 | 9 | 7 |
| cyclamen | 10 | 6 | 7 | 7 |
| daffodil | 10 | 8 | 4 | 4 |
| desert-rose | 10 | 5 | 8 | 7 |
| english marigold | 10 | 8 | 8 | 7 |
| fire lily | 10 | 10 | 10 | 9 |

| **Bunga** | **Jumlah Data *Test*** | **Jumlah *True* pada MobilenetV2** | **Jumlah *True* pada InceptionV3** | **Jumlah *True* pada Xception** |
|---|---|---|---|---|
| foxglove | 10 | 9 | 10 | 9 |
| frangipani | 10 | 10 | 10 | 10 |
| fritillary | 10 | 9 | 10 | 8 |
| garden phlox | 10 | 9 | 5 | 7 |
| gaura | 10 | 10 | 9 | 10 |
| gazania | 10 | 9 | 9 | 8 |
| geranium | 10 | 8 | 7 | 5 |
| giant white arum lily | 10 | 9 | 10 | 8 |
| globe thistle | 10 | 10 | 9 | 9 |
| globe-flower | 10 | 9 | 9 | 8 |
| grape hyacinth | 10 | 10 | 10 | 10 |
| great masterwort | 10 | 10 | 10 | 10 |
| hard-leaved pocket orchid | 10 | 10 | 10 | 10 |
| hibiscus | 10 | 9 | 10 | 10 |
| hippeastrum | 10 | 8 | 10 | 9 |
| japanese anemone | 10 | 7 | 8 | 7 |
| king protea | 10 | 8 | 10 | 8 |
| lenten rose | 10 | 6 | 5 | 7 |
| lotus lotus | 10 | 7 | 8 | 8 |
| love in the mist | 10 | 8 | 10 | 8 |
| magnolia | 10 | 5 | 6 | 7 |
| mallow | 10 | 7 | 5 | 4 |
| marigold | 10 | 6 | 8 | 3 |
| mexican aster | 10 | 8 | 9 | 9 |
| mexican petunia | 10 | 8 | 9 | 8 |
| monkshood | 10 | 9 | 6 | 8 |
| moon orchid | 10 | 9 | 10 | 10 |
| morning glory | 10 | 8 | 8 | 10 |
| orange dahlia | 10 | 9 | 7 | 4 |
| osteospermum | 10 | 5 | 9 | 5 |
| oxeye daisy | 10 | 10 | 10 | 10 |
| passion flower | 10 | 10 | 10 | 10 |

| Bunga | Jumlah Data *Test* | Jumlah *True* pada MobilenetV2 | Jumlah *True* pada InceptionV3 | Jumlah *True* pada Xception |
|---|---|---|---|---|
| pelargonium | 10 | 8 | 3 | 5 |
| peruvian lily | 10 | 8 | 10 | 10 |
| petunia | 10 | 9 | 7 | 7 |
| pincushion flower | 10 | 9 | 10 | 9 |
| pink primrose | 10 | 9 | 7 | 10 |
| pink-yellow dahlia | 10 | 3 | 8 | 7 |
| poinsettia | 10 | 9 | 7 | 7 |
| primula | 10 | 8 | 8 | 8 |
| prince of wales feathers | 10 | 6 | 7 | 7 |
| purple coneflower | 10 | 10 | 10 | 10 |
| red ginger | 10 | 9 | 10 | 9 |
| rose | 10 | 10 | 8 | 10 |
| ruby-lipped cattleya | 10 | 9 | 9 | 8 |
| siam tulip | 10 | 5 | 10 | 7 |
| silverbush | 10 | 6 | 7 | 5 |
| snapdragon | 10 | 6 | 8 | 7 |
| spear thistle | 10 | 10 | 10 | 10 |
| spring crocus | 10 | 3 | 7 | 9 |
| stemless gentian | 10 | 10 | 8 | 9 |
| sunflower | 10 | 9 | 10 | 10 |
| sweet pea | 10 | 5 | 5 | 4 |
| sweet william | 10 | 8 | 8 | 8 |
| sword lily | 10 | 8 | 5 | 8 |
| thorn apple | 10 | 5 | 7 | 7 |
| tiger lily | 10 | 9 | 9 | 10 |
| toad lily | 10 | 10 | 9 | 10 |
| tree mallow | 10 | 6 | 9 | 8 |
| tree poppy | 10 | 10 | 10 | 9 |
| trumpet creeper | 10 | 8 | 5 | 6 |
| wallflower | 10 | 10 | 10 | 10 |
| water lily | 10 | 10 | 10 | 10 |
| watercress | 10 | 8 | 9 | 8 |

| Bunga | Jumlah Data *Test* | Jumlah *True* pada MobilenetV2 | Jumlah *True* pada InceptionV3 | Jumlah *True* pada Xception |
|---|---|---|---|---|
| wild pansy | 10 | 7 | 8 | 6 |
| windflower | 10 | 9 | 10 | 9 |
| yellow iris | 10 | 8 | 8 | 8 |

Pada pengujian klasifikasi setiap kelas bunga dengan menggunakan model yang sudah dikonversi ke dalam format TFLite, hasil menunjukkan bahwa terdapat beberapa bunga yang memiliki akurasi yang berbeda-beda di setiap *pre-trained* model yang telah dilatih. Seperti pada bunga wild pansy pada MobileNetV2 mendapat jumlah benar 7 gambar, pada InceptionV3 mendapat jumlah benar 8 gambar, dan pada Xception mendapat jumlah benar 6 gambar. Terdapat beberapa kelas bunga yang mendapat akurasi baik di MobileNetV2 namun rendah pada model InceptionV3 dan Xception. Begitu juga terdapat kelas yang mendapat akurasi baik pada model Xception namun rendah pada model yang lain. Selain itu terdapat kelas yang mendapat hasil akurasi tinggi di semua model seperti pada kelas wallflower dan water lily yang mendapat jumlah benar 10 gambar pada semua MobileNetV2, InceptionV3, dan Xception.

Adapun waktu yang diperlukan untuk proses klasifikasi pada setiap model yang dapat dilihat pada tabel 4.9.

**Tabel 4.9** Waktu Klafikasi Setiap Model

| Model | Waktu Klasifikasi |
|---|---|
| MobilenetV2 | 2.01 detik |
| Xception | 2.31 detik |
| InceptionV3 | 2.44 detik |

Selain melihat performa. Waktu klasifikasi juga merupakan faktor penting yang perlu dipertimbangkan saat memilih model deep learning. Berdasarkan hasil percobaan, MobileNetV2 memerlukan waktu klasifikasi sebesar 2,01 detik, Xception 2,31 detik, dan InceptionV3 2,44 detik. MobileNetV2 menunjukkan keunggulan dalam hal kecepatan klasifikasi, dengan waktu yang paling cepat di antara ketiga model. Ini menjadikannya pilihan yang sangat menarik untuk aplikasi yang memerlukan *inferensi* cepat atau *real-time*, seperti pada perangkat mobile atau aplikasi dengan keterbatasan sumber daya komputasi.

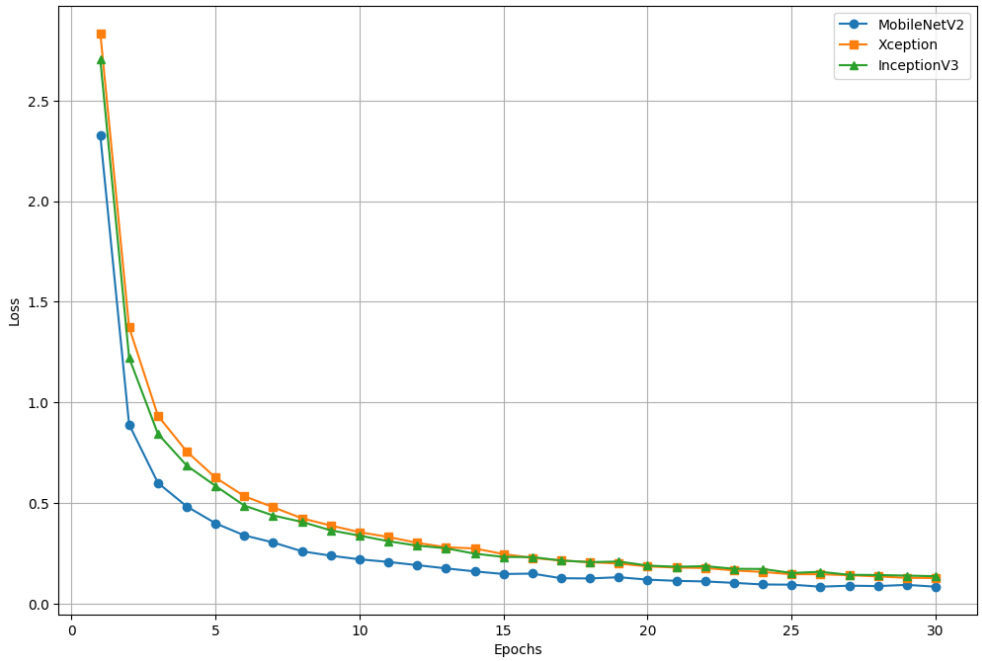
Xception, dengan waktu klasifikasi sebesar 2,31 detik, berada di posisi tengah. Meskipun sedikit lebih lambat dibandingkan MobileNetV2, Xception masih menunjukkan waktu klasifikasi yang cukup cepat. Dengan akurasi yang tinggi. Xception bisa menjadi pilihan yang baik untuk aplikasi yang mengutamakan akurasi sambil tetap mempertahankan waktu klasifikasi yang relatif singkat. InceptionV3, meskipun memiliki waktu klasifikasi yang paling lambat di antara ketiga model yaitu 2,44 detik.

## 4.3 Pembahasan/Diskusi

### 4.3.1 Pembahasan Pengujian *Training Pre-Trained* Model

Pada pengujian *training* yang dilakukan terhadap tiga *pre-trained* model, nilai *loss* dan *accuracy* terbaik dari setiap model tersebut dicatat pada Tabel 4.3. Nilai *loss* atau kerugian adalah nilai yang merepresentasikan jumlah kesalahan pada model. Nilai *loss* mengukur

seberapa baik atau buruk kinerja model. Jika kesalahannya tinggi, nilai *loss* akan tinggi, jika tingkat kesalahannya rendah, nilai *loss* semakin rendah. Nilai *loss* yang semakin rendah menunjukkan kinerja model yang semakin baik. Nilai *accuracy* atau akurasi adalah ukuran seberapa baik model melakukan prediksi dengan membandingkan jumlah prediksi benar dengan jumlah seluruh data prediksi, ditulis dalam bentuk persentase.




Gambar 4.1 Grafik *Loss* pada Pengujian Beberapa *Pre-trained* Model

Gambar 4.1 menunjukkan perbandingan *loss* dari tiga model deep learning yang berbeda, yaitu MobileNetV2, Xception, dan InceptionV3, selama 30 *epoch*. Pada *epoch* pertama, semua model memiliki nilai *loss* yang relatif tinggi, dengan InceptionV3 dan Xception di atas 2, sementara MobileNetV2 di atas 2,3. Namun, seiring berjalannya *epoch*, ketiga model menunjukkan penurunan yang signifikan dalam kehilangan validasi mereka.

MobileNetV2 menunjukkan penurunan yang sangat tajam dalam lima *epoch* pertama, turun dari 2,327 menjadi 0,339 pada *epoch* keenam. Setelah itu, nilai *loss* terus menurun dengan tingkat yang lebih lambat, mencapai nilai serendah 0,084 pada epoch ke-30. Hal ini menunjukkan bahwa MobileNetV2 memiliki kemampuan adaptasi yang baik dalam fase awal pelatihan, dengan kemampuan yang cukup konsisten dalam menurunkan *loss* hingga akhir pelatihan.

Xception, di sisi lain, memulai dengan *loss* yang lebih tinggi pada epoch pertama (2,835), tetapi menunjukkan penurunan yang konsisten dan cepat hingga epoch kesebelas, mencapai nilai 0,302. Penurunan ini terus berlanjut hingga mencapai nilai 0,127 pada epoch ke-30. Ini menunjukkan bahwa Xception juga cukup efisien dalam mengurangi *loss*, meskipun awalnya lebih tinggi dibandingkan dengan MobileNetV2.

InceptionV3 memulai dengan *loss* yang sedikit lebih rendah dari Xception pada epoch pertama (2,704) dan menunjukkan penurunan yang stabil sepanjang 30 epoch, mencapai nilai 0,136 pada epoch ke-30. Penurunan ini cukup konsisten, dengan beberapa fluktuasi kecil, tetapi secara keseluruhan menunjukkan tren yang positif dalam penurunan *loss*.

Secara keseluruhan, ketiga model menunjukkan kemampuan yang kuat dalam mengurangi *loss* selama pelatihan. Namun, MobileNetV2 menunjukkan performa yang lebih unggul dalam fase awal pelatihan, sementara Xception dan InceptionV3 menunjukkan kinerja

yang lebih baik dalam jangka panjang, dengan nilai *loss* yang sangat rendah pada akhir epoch.

Dari hasil pengujian pada tiga *pre-trained* model, didapatkan MobileNetV2 bahwa memberikan hasil terbaik dari *pre-trained* model lainnya, dengan nilai *loss* sebesar 0.0837 dan nilai akurasi sebesar 0.9799. Model dengan hasil terbaik berikutnya adalah Xception dengan nilai akurasi sebesar 0.9744, dan hasil terendah diperoleh InceptionV3 dengan nilai akurasi 0.9641. Hasil dari model tersebut diperoleh dengan konfigurasi lapisan *output* sesuai urutan berikut:

1. Base Model
2. GlobalAveragePooling2D *layer*
3. *Dense layer* dengan jumlah *node* 102 dan aktivasi *softmax*

Pada hasil pengujian menggunakan data uji yang berjumlah 10 gambar setiap kelas pada Tabel 4.4. Didapatkan nilai akurasi, presisi, *recall*, dan *F1-Score* dengan menggunakan fungsi *Classification_Report*() yang ditunjukkan pada Tabel 4.10.

**Tabel 4.10** Perbandingan *Classification Report* pada Setiap Model

| Model | Akurasi | Presisi | Recall | F1-Score |
|---|---|---|---|---|
| MobileNetV2 | 0.807 | 0.85 | 0.81 | 0.81 |
| InceptionV3 | 0.808 | 0.85 | 0.81 | 0.81 |
| Xception | 0.830 | 0.85 | 0.83 | 0.83 |

Tabel 4.10 menunjukan hasil perbandingan nilai evaluasi dengan menggunakan data uji. Dari hasil tersebut didapatkan bahwa nilai akurasi tertinggi didapat oleh model Xception dengan nilai akurasi 0.830. Model dengan hasil terbaik selanjutnya didapatkan oleh InceptionV3 dengan nilai akurasi 0.808, dan hasil akurasi terendah didapatkan oleh model MobileNetV2 dengan akurasi 0.807. Hasil ini menunjukan bahwa model Xception dapat lebih baik dalam mengenali jenis bunga pada gambar di luar dataset. Berbanding terbalik dengan model MobileNetV2 berdasarkan Tabel 4.3, model MobileNetV2 mendapatkan hasil akurasi terendah dalam mengenali jenis bunga pada gambar di luar dataset.

### 4.3.2 Pembahasan Uji Coba Aplikasi

Hasil uji coba pada Tabel 4.5 dengan metode *Black Box* menunjukkan aplikasi telah berhasil melakukan semua skenario pengujian yang telah ditetapkan. Berdasarkan hasil tersebut, dapat aplikasi sudah memenuhi persyaratan fungsional dan dapat digunakan untuk melakukan proses klasifikasi bunga.

### 4.3.3 Pembahasan Pengujian Model pada Aplikasi

Pada hasil pengujian menggunakan data uji yang berjumlah 10 gambar setiap kelas pada Tabel 4.9. Didapatkan nilai akurasi, presisi, *recall*, dan *F1-Score* dengan menggunakan fungsi *Classification_Report*() dengan perbandingan Tabel 4.10. Hasil perbandingan dapat dilihat pada Tabel 4.11.

**Tabel 4.11** Perbandingan *Classification Report* Model sebelum dan sudah dikonversi kedalam bentuk TFLite

| Model | Format H5 | | | | Format TFLite | | | |
|---|---|---|---|---|---|---|---|---|
| | Akurasi | Presisi | Recall | F1-Score | Akurasi | Presisi | Recall | F1-Score |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| MobileNetV2 | 0.807 | 0.85 | 0.81 | 0.81 | 0.78 | 0.84 | 0.78 | 0.78 |
| InceptionV3 | 0.808 | 0.85 | 0.81 | 0.81 | 0.75 | 0.82 | 0.75 | 0.74 |
| Xception | 0.830 | 0.85 | 0.83 | 0.83 | 0.82 | 0.85 | 0.82 | 0.82 |

Berdasarkan Tabel 4.11 didapatkan penurunan performa model setelah dikonversi ke dalam format TFLite. Dari hasil tersebut, Xception mendapatkan nilai performa terbaik dengan akurasi 0.83 sebelum dikonversi dan akurasi sebesar 0.82 setelah dikonversi. Model dengan nilai performa terbaik selanjutnya diperoleh MobileNetV2 dengan nilai akurasi 0.78, lalu dengan performa terendah didapatkan oleh InceptionV3 dengan nilai akurasi 0.75.

Dari Tabel 4.11 dapat diperoleh delta penurunan akurasi, presisi, *recall*, dan *F1-Score* pada setiap model yang ditunjukkan pada Tabel 4.12.

**Tabel 4.12** Perbandingan Delta Performa Model sebelum dan sudah dikonversi kedalam bentuk TFLite

| Model | Delta Akurasi | Delta Presisi | Delta Recall | Delta F1-Score |
|---|---|---|---|---|
| MobileNetV2 | 0.027 | 0.01 | 0.03 | 0.03 |
| InceptionV3 | 0.058 | 0.03 | 0.06 | 0.07 |
| Xception | 0.010 | 0.00 | 0.01 | 0.01 |

Pada Tabel 4.12 didapatkan Xception nilai delta terendah dengan nilai delta akurasi 0.01, delta presisi 0.0, delta *recall* 0.01, dan delta *F1-Score* 0.01 yang berarti model Xception lebih dapat mengenali jenis bunga pada perangkat keras maupun perangkat bergerak Android dibandingan dengan MobileNetV2 dan InceptionV3. Model MobileNetV2 mendapatkan nilai delta akurasi 0.027, delta presisi 0.01, delta *recall* 0.03, dan delta *F1-Score* 0.03, lalu dengan nilai delta tertinggi didapatkan model InceptionV3 dengan nilai delta akurasi 0.058, delta presisi 0.03, delta *recall* 0.06, dan delta *F1-Score* 0.07. Berdasarkan perbandingan nilai delta tersebut, MobileNetV2 dapat berkerja lebih baik daripada InceptionV2 pada perangkat bergerak Android.

# BAB V KESIMPULAN DAN SARAN

Bab ini membahas mengenai kesimpulan yang diperoleh dari pengerjaan tugas akhir dan saran terkait pengembangan dari tugas akhir ini yang dapat dilakukan di masa yang akan mendatang.

## 5.1 Kesimpulan

Berdasarkan uji coba dan evaluasi yang telah dilaksanakan, kesimpulan yang didapatkan adalah sebagai berikut:

1. Perancangan model klasifikasi tanaman bunga dengan menggunakan *deep learning* dimulai dari pengumpulan dataset bunga dan data uji bunga. Selanjutnya praproses data yang melibatkan serangkaian tahapan, seperti *resize*,pembagian dataset bunga menjadi dua yaitu data latih dan data validasi dengan perbandingan 80:20, lalu augmentasi data untuk memperbanyak variasi data latih. Setelah praproses data dilanjutkan dengan pelatihan model, evaluasi model yang sudah dilatih dengan memperhatikan nilai akurasi dan *loss*. Diakhiri dengan uji coba model dengan menggunakan data tes yang berisi data di luar dataset. Dari perancangan model didapatkan tiga model klasifikasi menggunakan *pre-trained* model dengan akurasi tertinggi yaitu Xception dengan nilai akurasi 0.83, InceptionV3 dengan nilai akurasi 0.808, dan MobileNetV2 dengan nilai akurasi 0.807
2. Penerapan aplikasi tanaman bunga menggunakan *deep learning* pada perangkat berbasis Android dapat diterapkan dengan memilih model dengan arsitektur yang berukuran ringan mengingat perangkat bergerak Android memiliki keterbatasan memori, penyimpanan, dan daya komputasi. Setelah itu mengkonversi model menjadi format TFLite agar dapat diterapkan pada perangkat bergerak Android.
3. Evaluasi kinerja aplikasi untuk klasifikasi jenis tanaman bunga pada data *real-time* dilakukan dengan mengukur kinerja menggunakan beberapa metrik evaluasi seperti akurasi, presisi, *recall*, dan *F1-score*. Hasil evaluasi ini memberikan gambaran tentang seberapa baik performa model dalam mengklasifikasi jenis tanaman bunga pada data *real-time*. Dilakukan juga perbandingan performa antara model Xception, InceptionV3, dan MobileNetV2 dengan format TFLite dengan menggunakan data uji. Didapatkan bahwa Xception memiliki nilai akurasi sebesar 0.82, MobileNetV2 dengan nilai akurasi 0.78, sementara InceptionV3 memiliki nilai akurasi terendah yaitu 0.75. Selain akurasi perlu juga mempertimbangkan berat model dan waktu *inference* dikarenakan perangkat Android memiliki keterbatasan memori, penyimpanan, dan daya komputasi. Didapatkan berat model MobileNetv2 setelah di konversi ke dalam format TFLite sebesar 8.95 *MB* dan memerlukan waktu klasifikasi sebesar 2,01 detik, Xception sebesar 80.1 *MB* dan memerlukan waktu klasifikasi sebesar 2,31 detik, sementara InceptionV3 memiliki berat model terbesar yaitu 83.8 *MB* dan memerlukan waktu klasifikasi sebesar 2,44 detik.

Dengan demikian, kesimpulan tersebut memberikan gambaran tentang perancangan model klasifikasi tanaman bunga dengan menggunakan *deep learning*, penerapan aplikasi tanaman bunga menggunakan *deep learning* pada perangkat berbasis android, dan cara melakukan evaluasi kinerja aplikasi untuk klasifikasi jenis tanaman bunga pada data r*eal-time* yang dibuat dalam Tugas Akhir ini.

## 5.2 Saran

Berdasarkan kesimpulan yang diambil dari Tugas Akhir ini, terdapat beberapa saran yang dapat diberikan sebagai berikut:

1. Meningkatkan kualitas data yang digunakan dengan menggabungkan sumber data dari berbagai sumber untuk mendapatkan representasi yang lebih luas tentang jenis bunga.
2. Melakukan *tuning* pada proses pembangunan model untuk mendapatkan hasil klasifikasi bunga yang lebih baik pada perangkat bergerak Android.

*(Halaman ini sengaja dikosongkan)*

# DAFTAR PUSTAKA


[1] M. J. M. Christenhusz and J. W. Byng, "The number of known plants species in the world and its annual increase," *Phytotaxa*, vol. 261, no. 3. Magnolia Press, pp. 201–217, May 20, 2016. doi: 10.11646/phytotaxa.261.3.1.

[2] C. Narvekar and M. Rao, "Flower classification using CNN and transfer learning in CNN- Agriculture Perspective," in *2020 3rd International Conference on Intelligent Sustainable Systems (ICISS)*, 2020, pp. 660–664. doi: 10.1109/ICISS49785.2020.9316030.

[3] B. R. Mete and T. Ensari, "Flower Classification with Deep CNN and Machine Learning Algorithms," in *2019 3rd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, 2019, pp. 1–5. doi: 10.1109/ISMSIT.2019.8932908.

[4] S. Desai, C. Gode, and P. Fulzele, "Flower Image Classification Using Convolutional Neural Network," in *2022 First International Conference on Electrical, Electronics, Information and Communication Technologies (ICEEICT)*, 2022, pp. 1–4. doi: 10.1109/ICEEICT53079.2022.9768635.

[5] Y. Wu, X. Qin, Y. Pan, and C. Yuan, "Convolution Neural Network based Transfer Learning for Classification of Flowers," in *2018 IEEE 3rd International Conference on Signal and Image Processing (ICSIP)*, 2018, pp. 562–566. doi: 10.1109/SIPROCESS.2018.8600536.

[6] O. A. Malik, N. Ismail, B. R. Hussein, and U. Yahya, "Automated Real-Time Identification of Medicinal Plants Species in Natural Environment Using Deep Learning Models—A Case Study from Borneo Region," *Plants*, vol. 11, no. 15, 2022, doi: 10.3390/plants11151952.

[7] N. Alipour, O. Tarkhaneh, M. Awrangjeb, and H. Tian, "Flower Image Classification Using Deep Convolutional Neural Network," in *2021 7th International Conference on Web Research (ICWR)*, 2021, pp. 1–4. doi: 10.1109/ICWR51868.2021.9443129.

[8] C. A. Elinisa and N. Mduma, "Mobile-Based convolutional neural network model for the early identification of banana diseases," *Smart Agricultural Technology*, vol. 7, p. 100423, 2024, doi: https://doi.org/10.1016/j.atech.2024.100423.

[9] R. Geirhos, D. H. J. Janssen, H. H. Schütt, J. Rauber, M. Bethge, and F. A. Wichmann, "Comparing deep neural networks against humans: object recognition when the signal gets weaker." 2018.

[10] M. Toğaçar, B. Ergen, and Z. Cömert, "BrainMRNet: Brain tumor detection using magnetic resonance images with a novel convolutional neural network model," *Med Hypotheses*, vol. 134, p. 109531, 2020, doi: https://doi.org/10.1016/j.mehy.2019.109531.

[11] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015, doi: 10.1038/nature14539.

[12] Y. Zhang and X. Zheng, "Development of Image Processing Based on Deep Learning Algorithm," in *2022 IEEE Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC)*, 2022, pp. 1226–1228. doi: 10.1109/IPEC54454.2022.9777479.

[13] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biol Cybern*, vol. 36, no. 4, pp. 193–202, 1980, doi: 10.1007/BF00344251.

[14] V. Balakrishnan, Z. Shi, C. L. Law, R. Lim, L. L. Teh, and Y. Fan, "A deep learning approach in predicting products' sentiment ratings: a comparative analysis," *Journal of*

*Supercomputing*, vol. 78, no. 5, pp. 7206–7226, Apr. 2022, doi: 10.1007/s11227-021-04169-6.
[15] M. Toğaçar, B. Ergen, and Z. Cömert, "Classification of flower species by using features extracted from the intersection of feature selection methods in convolutional neural network models," *Measurement*, vol. 158, p. 107703, 2020, doi: https://doi.org/10.1016/j.measurement.2020.107703.
[16] I. Tabian, H. Fu, and Z. S. Khodaei, "A convolutional neural network for impact detection and characterization of complex composite structures," *Sensors (Switzerland)*, vol. 19, no. 22, Nov. 2019, doi: 10.3390/s19224933.
[17] X. Sun, J. Peng, Y. Shen, and H. Kang, "Tobacco plant detection in RGB aerial images," *Agriculture (Switzerland)*, vol. 10, no. 3, Mar. 2020, doi: 10.3390/agriculture10030057.
[18] M. M. Taye, "Theoretical Understanding of Convolutional Neural Network: Concepts, Architectures, Applications, Future Directions," *Computation*, vol. 11, no. 3. MDPI, Mar. 01, 2023. doi: 10.3390/computation11030052.
[19] K. Dong, C. Zhou, Y. Ruan, and Y. Li, "MobileNetV2 Model for Image Classification," in *2020 2nd International Conference on Information Technology and Computer Application (ITCA)*, 2020, pp. 476–480. doi: 10.1109/ITCA52113.2020.00106.
[20] L. Bai, Y. Zhao, and X. Huang, "A CNN Accelerator on FPGA Using Depthwise Separable Convolution," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 65, pp. 1415–1419, 2018, [Online]. Available: https://api.semanticscholar.org/CorpusID:52164186
[21] S. F. Cotter, "MobiExpressNet: A Deep Learning Network for Face Expression Recognition on Smart Phones," in *2020 IEEE International Conference on Consumer Electronics (ICCE)*, 2020, pp. 1–4. doi: 10.1109/ICCE46568.2020.9042973.
[22] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks." 2019.
[23] C. Szegedy, V. Vanhoucke, S. Ioffe, and J. Shlens, "Rethinking the Inception Architecture for Computer Vision."
[24] F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions," Oct. 2016, [Online]. Available: http://arxiv.org/abs/1610.02357
[25] M. M. Leonardo, T. J. Carvalho, E. Rezende, R. Zucchi, and F. A. Faria, "Deep Feature-Based Classifiers for Fruit Fly Identification (Diptera: Tephritidae)," in *2018 31st SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, 2018, pp. 41–47. doi: 10.1109/SIBGRAPI.2018.00012.
[26] J. Hao and T. K. Ho, "Machine Learning Made Easy: A Review of Scikit-learn Package in Python Programming Language," *Journal of Educational and Behavioral Statistics*, vol. 44, no. 3, pp. 348–361, Feb. 2019, doi: 10.3102/1076998619832248.
[27] F. J. J. Joseph, S. Nonsiri, and A. Monsakul, "Correction to: Keras and TensorFlow: A Hands-On Experience," in *Advanced Deep Learning for Engineers and Scientists: A Practical Approach*, K. B. Prakash, R. Kannan, S. A. Alexander, and G. R. Kanagachidambaresan, Eds., Cham: Springer International Publishing, 2021, pp. C1–C1. doi: 10.1007/978-3-030-66519-7_12.
[28] J. Pardede and A. S. Purohita, "Hyperparameter Search for CT-Scan Classification Using Hyperparameter Tuning in Pre-Trained Model CNN With MLP," in *2022 IEEE International Conference of Computer Science and Information Technology (ICOSNIKOM)*, 2022, pp. 1–8. doi: 10.1109/ICOSNIKOM56551.2022.10034878.
[29] M.-E. Nilsback and A. Zisserman, "Automated flower classification over a large

number of classes."
[30] Scikit-Learn, "StratifiedShuffleSplit." Accessed: Jun. 23, 2024. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedShuffleSplit.html
[31] V. R. Joseph, "Optimal ratio for data splitting," *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 15, no. 4, pp. 531–538, Apr. 2022, doi: 10.1002/sam.11583.

*(Halaman ini sengaja dikosongkan)*

# LAMPIRAN A

## Perbandingan *Classification Report*

**Tabel 6.1** *Classification Report* Model MobileNetV2

| Kelas | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| alpine sea holly | 1.00 | 1.00 | 1.00 | 10 |
| anthurium | 1.00 | 0.80 | 0.89 | 10 |
| artichoke | 1.00 | 0.90 | 0.95 | 10 |
| azalea | 0.78 | 0.70 | 0.74 | 10 |
| ball moss | 1.00 | 0.70 | 0.82 | 10 |
| balloon flower | 1.00 | 0.70 | 0.82 | 10 |
| barbeton daisy | 0.43 | 0.30 | 0.35 | 10 |
| bearded iris | 1.00 | 0.70 | 0.82 | 10 |
| bee balm | 0.50 | 0.90 | 0.64 | 10 |
| bird of paradise | 1.00 | 1.00 | 1.00 | 10 |
| bishop of llandaff | 0.89 | 0.80 | 0.84 | 10 |
| black-eyed susan | 1.00 | 0.80 | 0.89 | 10 |
| blackberry lily | 0.77 | 1.00 | 0.87 | 10 |
| blanket flower | 0.67 | 1.00 | 0.80 | 10 |
| bolero deep blue | 1.00 | 0.80 | 0.89 | 10 |
| bougainvillea | 0.67 | 0.80 | 0.73 | 10 |
| bromelia | 0.71 | 1.00 | 0.83 | 10 |
| buttercup | 0.83 | 1.00 | 0.91 | 10 |
| californian poppy | 1.00 | 0.70 | 0.82 | 10 |
| camellia | 0.78 | 0.70 | 0.74 | 10 |
| canna lily | 0.78 | 0.70 | 0.74 | 10 |
| canterbury bells | 1.00 | 0.70 | 0.82 | 10 |
| cape flower | 0.80 | 0.80 | 0.80 | 10 |
| carnation | 0.71 | 1.00 | 0.83 | 10 |
| cautleya spicata | 1.00 | 0.30 | 0.46 | 10 |
| clematis | 0.70 | 0.70 | 0.70 | 10 |
| colt's foot | 1.00 | 0.90 | 0.95 | 10 |
| columbine | 0.38 | 0.80 | 0.52 | 10 |
| common dandelion | 0.90 | 0.90 | 0.90 | 10 |
| corn poppy | 0.91 | 1.00 | 0.95 | 10 |

| Kelas | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| cyclamen | 0.75 | 0.60 | 0.67 | 10 |
| daffodil | 0.80 | 0.80 | 0.80 | 10 |
| desert-rose | 1.00 | 0.50 | 0.67 | 10 |
| english marigold | 0.67 | 0.80 | 0.73 | 10 |
| fire lily | 0.91 | 1.00 | 0.95 | 10 |
| foxglove | 0.75 | 0.90 | 0.82 | 10 |
| frangipani | 1.00 | 1.00 | 1.00 | 10 |
| fritillary | 1.00 | 0.90 | 0.95 | 10 |
| garden phlox | 0.56 | 0.90 | 0.69 | 10 |
| gaura | 1.00 | 1.00 | 1.00 | 10 |
| gazania | 1.00 | 0.90 | 0.95 | 10 |
| geranium | 1.00 | 0.80 | 0.89 | 10 |
| giant white arum lily | 1.00 | 0.90 | 0.95 | 10 |
| globe thistle | 1.00 | 1.00 | 1.00 | 10 |
| globe-flower | 1.00 | 0.90 | 0.95 | 10 |
| grape hyacinth | 0.91 | 1.00 | 0.95 | 10 |
| great masterwort | 0.91 | 1.00 | 0.95 | 10 |
| hard-leaved pocket orchid | 1.00 | 1.00 | 1.00 | 10 |
| hibiscus | 0.75 | 0.90 | 0.82 | 10 |
| hippeastrum | 0.67 | 0.80 | 0.73 | 10 |
| japanese anemone | 0.88 | 0.70 | 0.78 | 10 |
| king protea | 1.00 | 0.80 | 0.89 | 10 |
| lenten rose | 0.86 | 0.60 | 0.71 | 10 |
| lotus lotus | 0.78 | 0.70 | 0.74 | 10 |
| love in the mist | 0.89 | 0.80 | 0.84 | 10 |
| magnolia | 1.00 | 0.50 | 0.67 | 10 |
| mallow | 1.00 | 0.70 | 0.82 | 10 |
| marigold | 1.00 | 0.60 | 0.75 | 10 |
| mexican aster | 0.73 | 0.80 | 0.76 | 10 |
| mexican petunia | 1.00 | 0.80 | 0.89 | 10 |
| monkshood | 0.75 | 0.90 | 0.82 | 10 |
| moon orchid | 1.00 | 0.90 | 0.95 | 10 |
| morning glory | 0.80 | 0.80 | 0.80 | 10 |
| orange dahlia | 1.00 | 0.90 | 0.95 | 10 |

| Kelas | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| osteospermum | 0.71 | 0.50 | 0.59 | 10 |
| oxeye daisy | 1.00 | 1.00 | 1.00 | 10 |
| passion flower | 0.63 | 1.00 | 0.77 | 10 |
| pelargonium | 0.80 | 0.80 | 0.80 | 10 |
| peruvian lily | 0.73 | 0.80 | 0.76 | 10 |
| petunia | 0.41 | 0.90 | 0.56 | 10 |
| pincushion flower | 0.69 | 0.90 | 0.78 | 10 |
| pink primrose | 1.00 | 0.90 | 0.95 | 10 |
| pink-yellow dahlia | 1.00 | 0.30 | 0.46 | 10 |
| poinsettia | 0.90 | 0.90 | 0.90 | 10 |
| primula | 0.50 | 0.80 | 0.62 | 10 |
| prince of wales feathers | 1.00 | 0.60 | 0.75 | 10 |
| purple coneflower | 0.83 | 1.00 | 0.91 | 10 |
| red ginger | 0.75 | 0.90 | 0.82 | 10 |
| rose | 0.56 | 1.00 | 0.71 | 10 |
| ruby-lipped cattleya | 1.00 | 0.90 | 0.95 | 10 |
| siam tulip | 1.00 | 0.50 | 0.67 | 10 |
| silverbush | 1.00 | 0.60 | 0.75 | 10 |
| snapdragon | 1.00 | 0.60 | 0.75 | 10 |
| spear thistle | 0.91 | 1.00 | 0.95 | 10 |
| spring crocus | 0.75 | 0.30 | 0.43 | 10 |
| stemless gentian | 1.00 | 1.00 | 1.00 | 10 |
| sunflower | 1.00 | 0.90 | 0.95 | 10 |
| sweet pea | 0.50 | 0.50 | 0.50 | 10 |
| sweet william | 0.62 | 0.80 | 0.70 | 10 |
| sword lily | 0.89 | 0.80 | 0.84 | 10 |
| thorn apple | 1.00 | 0.50 | 0.67 | 10 |
| tiger lily | 1.00 | 0.90 | 0.95 | 10 |
| toad lily | 0.91 | 1.00 | 0.95 | 10 |
| tree mallow | 0.67 | 0.60 | 0.63 | 10 |
| tree poppy | 0.77 | 1.00 | 0.87 | 10 |
| trumpet creeper | 0.73 | 0.80 | 0.76 | 10 |
| wallflower | 0.53 | 1.00 | 0.69 | 10 |
| water lily | 0.71 | 1.00 | 0.83 | 10 |

| Kelas | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| watercress | 1.00 | 0.80 | 0.89 | 10 |
| wild pansy | 0.78 | 0.70 | 0.74 | 10 |
| windflower | 0.75 | 0.90 | 0.82 | 10 |
| yellow iris | 1.00 | 0.80 | 0.89 | 10 |
| | | | | |
| **macro avg** | 0.85 | 0.81 | 0.81 | 1020 |
| **weighted avg** | 0.85 | 0.81 | 0.81 | 1020 |

**Tabel 6.2** *Classification Report* Model Xception

| Kelas | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| alpine sea holly | 1.00 | 0.80 | 0.89 | 10 |
| anthurium | 1.00 | 1.00 | 1.00 | 10 |
| artichoke | 1.00 | 0.90 | 0.95 | 10 |
| azalea | 1.00 | 0.70 | 0.82 | 10 |
| ball moss | 1.00 | 0.90 | 0.95 | 10 |
| balloon flower | 0.75 | 0.90 | 0.82 | 10 |
| barbeton daisy | 1.00 | 0.70 | 0.82 | 10 |
| bearded iris | 1.00 | 0.70 | 0.82 | 10 |
| bee balm | 0.78 | 0.70 | 0.74 | 10 |
| bird of paradise | 1.00 | 0.90 | 0.95 | 10 |
| bishop of llandaff | 0.82 | 0.90 | 0.86 | 10 |
| black-eyed susan | 0.91 | 1.00 | 0.95 | 10 |
| blackberry lily | 0.75 | 0.90 | 0.82 | 10 |
| blanket flower | 0.82 | 0.90 | 0.86 | 10 |
| bolero deep blue | 0.75 | 0.90 | 0.82 | 10 |
| bougainvillea | 0.69 | 0.90 | 0.78 | 10 |
| bromelia | 1.00 | 1.00 | 1.00 | 10 |
| buttercup | 0.82 | 0.90 | 0.86 | 10 |
| californian poppy | 0.78 | 0.70 | 0.74 | 10 |
| camellia | 0.63 | 1.00 | 0.77 | 10 |
| canna lily | 1.00 | 0.40 | 0.57 | 10 |
| canterbury bells | 1.00 | 0.50 | 0.67 | 10 |
| cape flower | 1.00 | 0.90 | 0.95 | 10 |
| carnation | 0.82 | 0.90 | 0.86 | 10 |

| Kelas | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| cautleya spicata | 1.00 | 0.40 | 0.57 | 10 |
| clematis | 1.00 | 0.90 | 0.95 | 10 |
| colt's foot | 0.71 | 1.00 | 0.83 | 10 |
| columbine | 0.64 | 0.90 | 0.75 | 10 |
| common dandelion | 0.73 | 0.80 | 0.76 | 10 |
| corn poppy | 1.00 | 0.90 | 0.95 | 10 |
| cyclamen | 1.00 | 0.70 | 0.82 | 10 |
| daffodil | 0.80 | 0.40 | 0.53 | 10 |
| desert-rose | 1.00 | 0.80 | 0.89 | 10 |
| english marigold | 0.89 | 0.80 | 0.84 | 10 |
| fire lily | 0.91 | 1.00 | 0.95 | 10 |
| foxglove | 0.91 | 1.00 | 0.95 | 10 |
| frangipani | 1.00 | 1.00 | 1.00 | 10 |
| fritillary | 0.91 | 1.00 | 0.95 | 10 |
| garden phlox | 0.56 | 0.50 | 0.53 | 10 |
| gaura | 0.75 | 0.90 | 0.82 | 10 |
| gazania | 1.00 | 0.90 | 0.95 | 10 |
| geranium | 0.78 | 0.70 | 0.74 | 10 |
| giant white arum lily | 1.00 | 1.00 | 1.00 | 10 |
| globe thistle | 0.90 | 0.90 | 0.90 | 10 |
| globe-flower | 0.82 | 0.90 | 0.86 | 10 |
| grape hyacinth | 1.00 | 1.00 | 1.00 | 10 |
| great masterwort | 1.00 | 1.00 | 1.00 | 10 |
| hard-leaved pocket orchid | 1.00 | 1.00 | 1.00 | 10 |
| hibiscus | 0.63 | 1.00 | 0.77 | 10 |
| hippeastrum | 0.91 | 1.00 | 0.95 | 10 |
| japanese anemone | 0.80 | 0.80 | 0.80 | 10 |
| king protea | 0.91 | 1.00 | 0.95 | 10 |
| lenten rose | 0.63 | 0.50 | 0.56 | 10 |
| lotus lotus | 0.67 | 0.80 | 0.73 | 10 |
| love in the mist | 1.00 | 1.00 | 1.00 | 10 |
| magnolia | 1.00 | 0.60 | 0.75 | 10 |
| mallow | 0.83 | 0.50 | 0.63 | 10 |
| marigold | 1.00 | 0.80 | 0.89 | 10 |

| Kelas | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| mexican aster | 0.82 | 0.90 | 0.86 | 10 |
| mexican petunia | 1.00 | 0.90 | 0.95 | 10 |
| monkshood | 0.75 | 0.60 | 0.67 | 10 |
| moon orchid | 1.00 | 1.00 | 1.00 | 10 |
| morning glory | 0.80 | 0.80 | 0.80 | 10 |
| orange dahlia | 1.00 | 0.70 | 0.82 | 10 |
| osteospermum | 0.82 | 0.90 | 0.86 | 10 |
| oxeye daisy | 0.91 | 1.00 | 0.95 | 10 |
| passion flower | 0.91 | 1.00 | 0.95 | 10 |
| pelargonium | 0.43 | 0.30 | 0.35 | 10 |
| peruvian lily | 0.83 | 1.00 | 0.91 | 10 |
| petunia | 0.58 | 0.70 | 0.64 | 10 |
| pincushion flower | 0.83 | 1.00 | 0.91 | 10 |
| pink primrose | 0.70 | 0.70 | 0.70 | 10 |
| pink-yellow dahlia | 0.89 | 0.80 | 0.84 | 10 |
| poinsettia | 1.00 | 0.70 | 0.82 | 10 |
| primula | 0.73 | 0.80 | 0.76 | 10 |
| prince of wales feathers | 0.88 | 0.70 | 0.78 | 10 |
| purple coneflower | 0.91 | 1.00 | 0.95 | 10 |
| red ginger | 0.77 | 1.00 | 0.87 | 10 |
| rose | 0.50 | 0.80 | 0.62 | 10 |
| ruby-lipped cattleya | 0.82 | 0.90 | 0.86 | 10 |
| siam tulip | 0.91 | 1.00 | 0.95 | 10 |
| silverbush | 1.00 | 0.70 | 0.82 | 10 |
| snapdragon | 0.73 | 0.80 | 0.76 | 10 |
| spear thistle | 0.77 | 1.00 | 0.87 | 10 |
| spring crocus | 1.00 | 0.70 | 0.82 | 10 |
| stemless gentian | 0.62 | 0.80 | 0.70 | 10 |
| sunflower | 1.00 | 1.00 | 1.00 | 10 |
| sweet pea | 0.71 | 0.50 | 0.59 | 10 |
| sweet william | 0.67 | 0.80 | 0.73 | 10 |
| sword lily | 1.00 | 0.50 | 0.67 | 10 |
| thorn apple | 0.70 | 0.70 | 0.70 | 10 |
| tiger lily | 1.00 | 0.90 | 0.95 | 10 |

| Kelas | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| toad lily | 1.00 | 0.90 | 0.95 | 10 |
| tree mallow | 0.90 | 0.90 | 0.90 | 10 |
| tree poppy | 0.91 | 1.00 | 0.95 | 10 |
| trumpet creeper | 0.71 | 0.50 | 0.59 | 10 |
| wallflower | 0.43 | 1.00 | 0.61 | 10 |
| water lily | 0.71 | 1.00 | 0.83 | 10 |
| watercress | 0.75 | 0.90 | 0.82 | 10 |
| wild pansy | 1.00 | 0.80 | 0.89 | 10 |
| windflower | 0.83 | 1.00 | 0.91 | 10 |
| yellow iris | 0.89 | 0.80 | 0.84 | 10 |
| | | | | |
| **macro avg** | 0.85 | 0.83 | 0.83 | 1020 |
| **weighted avg** | 0.85 | 0.83 | 0.83 | 1020 |

**Tabel 6.3** *Classification Report* Model InceptionV3

| Kelas | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| alpine sea holly | 0.83 | 1.00 | 0.91 | 10 |
| anthurium | 1.00 | 0.80 | 0.89 | 10 |
| artichoke | 1.00 | 0.90 | 0.95 | 10 |
| azalea | 1.00 | 0.60 | 0.75 | 10 |
| ball moss | 1.00 | 0.80 | 0.89 | 10 |
| balloon flower | 1.00 | 0.90 | 0.95 | 10 |
| barbeton daisy | 0.70 | 0.70 | 0.70 | 10 |
| bearded iris | 0.71 | 1.00 | 0.83 | 10 |
| bee balm | 0.88 | 0.70 | 0.78 | 10 |
| bird of paradise | 1.00 | 1.00 | 1.00 | 10 |
| bishop of llandaff | 0.59 | 1.00 | 0.74 | 10 |
| black-eyed susan | 0.90 | 0.90 | 0.90 | 10 |
| blackberry lily | 0.91 | 1.00 | 0.95 | 10 |
| blanket flower | 1.00 | 0.80 | 0.89 | 10 |
| bolero deep blue | 1.00 | 0.80 | 0.89 | 10 |
| bougainvillea | 0.64 | 0.70 | 0.67 | 10 |
| bromelia | 0.83 | 1.00 | 0.91 | 10 |
| buttercup | 0.78 | 0.70 | 0.74 | 10 |

| Kelas | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| californian poppy | 0.89 | 0.80 | 0.84 | 10 |
| camellia | 0.69 | 0.90 | 0.78 | 10 |
| canna lily | 1.00 | 0.60 | 0.75 | 10 |
| canterbury bells | 0.88 | 0.70 | 0.78 | 10 |
| cape flower | 1.00 | 1.00 | 1.00 | 10 |
| carnation | 0.83 | 1.00 | 0.91 | 10 |
| cautleya spicata | 0.67 | 0.80 | 0.73 | 10 |
| clematis | 0.46 | 0.60 | 0.52 | 10 |
| colt's foot | 1.00 | 0.90 | 0.95 | 10 |
| columbine | 0.60 | 0.60 | 0.60 | 10 |
| common dandelion | 0.77 | 1.00 | 0.87 | 10 |
| corn poppy | 0.88 | 0.70 | 0.78 | 10 |
| cyclamen | 0.64 | 0.70 | 0.67 | 10 |
| daffodil | 0.80 | 0.40 | 0.53 | 10 |
| desert-rose | 1.00 | 0.70 | 0.82 | 10 |
| english marigold | 0.88 | 0.70 | 0.78 | 10 |
| fire lily | 1.00 | 0.90 | 0.95 | 10 |
| foxglove | 1.00 | 0.90 | 0.95 | 10 |
| frangipani | 0.91 | 1.00 | 0.95 | 10 |
| fritillary | 1.00 | 0.80 | 0.89 | 10 |
| garden phlox | 0.58 | 0.70 | 0.64 | 10 |
| gaura | 0.83 | 1.00 | 0.91 | 10 |
| gazania | 0.80 | 0.80 | 0.80 | 10 |
| geranium | 1.00 | 0.50 | 0.67 | 10 |
| giant white arum lily | 1.00 | 0.80 | 0.89 | 10 |
| globe thistle | 0.90 | 0.90 | 0.90 | 10 |
| globe-flower | 1.00 | 0.80 | 0.89 | 10 |
| grape hyacinth | 0.91 | 1.00 | 0.95 | 10 |
| great masterwort | 0.83 | 1.00 | 0.91 | 10 |
| hard-leaved pocket orchid | 1.00 | 1.00 | 1.00 | 10 |
| hibiscus | 0.83 | 1.00 | 0.91 | 10 |
| hippeastrum | 0.75 | 0.90 | 0.82 | 10 |
| japanese anemone | 0.88 | 0.70 | 0.78 | 10 |
| king protea | 0.80 | 0.80 | 0.80 | 10 |

| Kelas | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| lenten rose | 0.70 | 0.70 | 0.70 | 10 |
| lotus lotus | 0.67 | 0.80 | 0.73 | 10 |
| love in the mist | 1.00 | 0.80 | 0.89 | 10 |
| magnolia | 1.00 | 0.70 | 0.82 | 10 |
| mallow | 1.00 | 0.40 | 0.57 | 10 |
| marigold | 1.00 | 0.30 | 0.46 | 10 |
| mexican aster | 0.60 | 0.90 | 0.72 | 10 |
| mexican petunia | 0.80 | 0.80 | 0.80 | 10 |
| monkshood | 0.80 | 0.80 | 0.80 | 10 |
| moon orchid | 0.91 | 1.00 | 0.95 | 10 |
| morning glory | 0.71 | 1.00 | 0.83 | 10 |
| orange dahlia | 0.80 | 0.40 | 0.53 | 10 |
| osteospermum | 1.00 | 0.50 | 0.67 | 10 |
| oxeye daisy | 0.83 | 1.00 | 0.91 | 10 |
| passion flower | 0.83 | 1.00 | 0.91 | 10 |
| pelargonium | 0.83 | 0.50 | 0.63 | 10 |
| peruvian lily | 1.00 | 1.00 | 1.00 | 10 |
| petunia | 0.64 | 0.70 | 0.67 | 10 |
| pincushion flower | 0.90 | 0.90 | 0.90 | 10 |
| pink primrose | 0.83 | 1.00 | 0.91 | 10 |
| pink-yellow dahlia | 1.00 | 0.70 | 0.82 | 10 |
| poinsettia | 0.78 | 0.70 | 0.74 | 10 |
| primula | 0.50 | 0.80 | 0.62 | 10 |
| prince of wales feathers | 1.00 | 0.70 | 0.82 | 10 |
| purple coneflower | 0.83 | 1.00 | 0.91 | 10 |
| red ginger | 1.00 | 0.90 | 0.95 | 10 |
| rose | 0.45 | 1.00 | 0.63 | 10 |
| ruby-lipped cattleya | 1.00 | 0.80 | 0.89 | 10 |
| siam tulip | 0.88 | 0.70 | 0.78 | 10 |
| silverbush | 1.00 | 0.50 | 0.67 | 10 |
| snapdragon | 0.54 | 0.70 | 0.61 | 10 |
| spear thistle | 0.91 | 1.00 | 0.95 | 10 |
| spring crocus | 0.64 | 0.90 | 0.75 | 10 |
| stemless gentian | 1.00 | 0.90 | 0.95 | 10 |

| Kelas | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| sunflower | 0.83 | 1.00 | 0.91 | 10 |
| sweet pea | 1.00 | 0.40 | 0.57 | 10 |
| sweet william | 0.67 | 0.80 | 0.73 | 10 |
| sword lily | 0.67 | 0.80 | 0.73 | 10 |
| thorn apple | 0.78 | 0.70 | 0.74 | 10 |
| tiger lily | 0.91 | 1.00 | 0.95 | 10 |
| toad lily | 1.00 | 1.00 | 1.00 | 10 |
| tree mallow | 1.00 | 0.80 | 0.89 | 10 |
| tree poppy | 1.00 | 0.90 | 0.95 | 10 |
| trumpet creeper | 0.86 | 0.60 | 0.71 | 10 |
| wallflower | 0.31 | 1.00 | 0.48 | 10 |
| water lily | 0.83 | 1.00 | 0.91 | 10 |
| watercress | 0.89 | 0.80 | 0.84 | 10 |
| wild pansy | 0.75 | 0.60 | 0.67 | 10 |
| windflower | 1.00 | 0.90 | 0.95 | 10 |
| yellow iris | 1.00 | 0.80 | 0.89 | 10 |
| | | | | |
| **macro avg** | 0.85 | 0.81 | 0.81 | 1020 |
| **weighted avg** | 0.85 | 0.81 | 0.81 | 1020 |

# LAMPIRAN B

## Perbandingan *Confusion Matrix*

**Tabel 6.4** *Confusion Matrix* pada Pengujian Model MobileNetV2

| **Kelas** | ***True Positives*** **(TP)** | ***False Positives*** **(FP)** | ***False Negatives*** **(FN)** | ***True Negatives*** **(TN)** |
|---|---|---|---|---|
| alpine sea holly | 10 | 0 | 0 | 1010 |
| anthurium | 8 | 0 | 2 | 1010 |
| artichoke | 9 | 0 | 1 | 1010 |
| azalea | 7 | 2 | 3 | 1008 |
| ball moss | 7 | 0 | 3 | 1010 |
| balloon flower | 7 | 0 | 3 | 1010 |
| barbeton daisy | 3 | 4 | 7 | 1006 |
| bearded iris | 7 | 0 | 3 | 1010 |
| bee balm | 9 | 9 | 1 | 1001 |
| bird of paradise | 10 | 0 | 0 | 1010 |
| bishop of llandaff | 8 | 1 | 2 | 1009 |
| black-eyed susan | 8 | 0 | 2 | 1010 |
| blackberry lily | 10 | 3 | 0 | 1007 |
| blanket flower | 10 | 5 | 0 | 1005 |
| bolero deep blue | 8 | 0 | 2 | 1010 |
| bougainvillea | 8 | 4 | 2 | 1006 |
| bromelia | 10 | 4 | 0 | 1006 |
| buttercup | 10 | 2 | 0 | 1008 |
| californian poppy | 7 | 0 | 3 | 1010 |
| camellia | 7 | 2 | 3 | 1008 |
| canna lily | 7 | 2 | 3 | 1008 |
| canterbury bells | 7 | 0 | 3 | 1010 |
| cape flower | 8 | 2 | 2 | 1008 |
| carnation | 10 | 4 | 0 | 1006 |
| cautleya spicata | 3 | 0 | 7 | 1010 |
| clematis | 7 | 3 | 3 | 1007 |
| colt's foot | 9 | 0 | 1 | 1010 |
| columbine | 8 | 13 | 2 | 997 |
| common dandelion | 9 | 1 | 1 | 1009 |

| **Kelas** | ***True Positives* (TP)** | ***False Positives* (FP)** | ***False Negatives* (FN)** | ***True Negatives* (TN)** |
|---|---|---|---|---|
| corn poppy | 10 | 1 | 0 | 1009 |
| cyclamen | 6 | 2 | 4 | 1008 |
| daffodil | 8 | 2 | 2 | 1008 |
| desert-rose | 5 | 0 | 5 | 1010 |
| english marigold | 8 | 4 | 2 | 1006 |
| fire lily | 10 | 1 | 0 | 1009 |
| foxglove | 9 | 3 | 1 | 1007 |
| frangipani | 10 | 0 | 0 | 1010 |
| fritillary | 9 | 0 | 1 | 1010 |
| garden phlox | 9 | 7 | 1 | 1003 |
| gaura | 10 | 0 | 0 | 1010 |
| gazania | 9 | 0 | 1 | 1010 |
| geranium | 8 | 0 | 2 | 1010 |
| giant white arum lily | 9 | 0 | 1 | 1010 |
| globe thistle | 10 | 0 | 0 | 1010 |
| globe-flower | 9 | 0 | 1 | 1010 |
| grape hyacinth | 10 | 1 | 0 | 1009 |
| great masterwort | 10 | 1 | 0 | 1009 |
| hard-leaved pocket orchid | 10 | 0 | 0 | 1010 |
| hibiscus | 9 | 3 | 1 | 1007 |
| hippeastrum | 8 | 4 | 2 | 1006 |
| japanese anemone | 7 | 1 | 3 | 1009 |
| king protea | 8 | 0 | 2 | 1010 |
| lenten rose | 6 | 1 | 4 | 1009 |
| lotus lotus | 7 | 2 | 3 | 1008 |
| love in the mist | 8 | 1 | 2 | 1009 |
| magnolia | 5 | 0 | 5 | 1010 |
| mallow | 7 | 0 | 3 | 1010 |
| marigold | 6 | 0 | 4 | 1010 |
| mexican aster | 8 | 3 | 2 | 1007 |
| mexican petunia | 8 | 0 | 2 | 1010 |
| monkshood | 9 | 3 | 1 | 1007 |
| moon orchid | 9 | 0 | 1 | 1010 |

| **Kelas** | ***True Positives* (TP)** | ***False Positives* (FP)** | ***False Negatives* (FN)** | ***True Negatives* (TN)** |
|---|---|---|---|---|
| morning glory | 8 | 2 | 2 | 1008 |
| orange dahlia | 9 | 0 | 1 | 1010 |
| osteospermum | 5 | 2 | 5 | 1008 |
| oxeye daisy | 10 | 0 | 0 | 1010 |
| passion flower | 10 | 6 | 0 | 1004 |
| pelargonium | 8 | 2 | 2 | 1008 |
| peruvian lily | 8 | 3 | 2 | 1007 |
| petunia | 9 | 13 | 1 | 997 |
| pincushion flower | 9 | 4 | 1 | 1006 |
| pink primrose | 9 | 0 | 1 | 1010 |
| pink-yellow dahlia | 3 | 0 | 7 | 1010 |
| poinsettia | 9 | 1 | 1 | 1009 |
| primula | 8 | 8 | 2 | 1002 |
| prince of wales feathers | 6 | 0 | 4 | 1010 |
| purple coneflower | 10 | 2 | 0 | 1008 |
| red ginger | 9 | 3 | 1 | 1007 |
| rose | 10 | 8 | 0 | 1002 |
| ruby-lipped cattleya | 9 | 0 | 1 | 1010 |
| siam tulip | 5 | 0 | 5 | 1010 |
| silverbush | 6 | 0 | 4 | 1010 |
| snapdragon | 6 | 0 | 4 | 1010 |
| spear thistle | 10 | 1 | 0 | 1009 |
| spring crocus | 3 | 1 | 7 | 1009 |
| stemless gentian | 10 | 0 | 0 | 1010 |
| sunflower | 9 | 0 | 1 | 1010 |
| sweet pea | 5 | 5 | 5 | 1005 |
| sweet william | 8 | 5 | 2 | 1005 |
| sword lily | 8 | 1 | 2 | 1009 |
| thorn apple | 5 | 0 | 5 | 1010 |
| tiger lily | 9 | 0 | 1 | 1010 |
| toad lily | 10 | 1 | 0 | 1009 |
| tree mallow | 6 | 3 | 4 | 1007 |
| tree poppy | 10 | 3 | 0 | 1007 |

| Kelas | *True Positives* (TP) | *False Positives* (FP) | *False Negatives* (FN) | *True Negatives* (TN) |
|---|---|---|---|---|
| trumpet creeper | 8 | 3 | 2 | 1007 |
| wallflower | 10 | 9 | 0 | 1001 |
| water lily | 10 | 4 | 0 | 1006 |
| watercress | 8 | 0 | 2 | 1010 |
| wild pansy | 7 | 2 | 3 | 1008 |
| windflower | 9 | 3 | 1 | 1007 |
| yellow iris | 8 | 0 | 2 | 1010 |

**Tabel 6.5** *Confusion Matrix* pada Pengujian Model Xception

| Kelas | *True Positives* (TP) | *False Positives* (FP) | *False Negatives* (FN) | *True Negatives* (TN) |
|---|---|---|---|---|
| alpine sea holly | 8 | 0 | 2 | 1010 |
| anthurium | 10 | 0 | 0 | 1010 |
| artichoke | 9 | 0 | 1 | 1010 |
| azalea | 7 | 0 | 3 | 1010 |
| ball moss | 9 | 0 | 1 | 1010 |
| balloon flower | 9 | 3 | 1 | 1007 |
| barbeton daisy | 7 | 0 | 3 | 1010 |
| bearded iris | 7 | 0 | 3 | 1010 |
| bee balm | 7 | 2 | 3 | 1008 |
| bird of paradise | 9 | 0 | 1 | 1010 |
| bishop of llandaff | 9 | 2 | 1 | 1008 |
| black-eyed susan | 10 | 1 | 0 | 1009 |
| blackberry lily | 9 | 3 | 1 | 1007 |
| blanket flower | 9 | 2 | 1 | 1008 |
| bolero deep blue | 9 | 3 | 1 | 1007 |
| bougainvillea | 9 | 4 | 1 | 1006 |
| bromelia | 10 | 0 | 0 | 1010 |
| buttercup | 9 | 2 | 1 | 1008 |
| californian poppy | 7 | 2 | 3 | 1008 |
| camellia | 10 | 6 | 0 | 1004 |
| canna lily | 4 | 0 | 6 | 1010 |
| canterbury bells | 5 | 0 | 5 | 1010 |

| **Kelas** | ***True Positives* (TP)** | ***False Positives* (FP)** | ***False Negatives* (FN)** | ***True Negatives* (TN)** |
|---|---|---|---|---|
| cape flower | 9 | 0 | 1 | 1010 |
| carnation | 9 | 2 | 1 | 1008 |
| cautleya spicata | 4 | 0 | 6 | 1010 |
| clematis | 9 | 0 | 1 | 1010 |
| colt's foot | 10 | 4 | 0 | 1006 |
| columbine | 9 | 5 | 1 | 1005 |
| common dandelion | 8 | 3 | 2 | 1007 |
| corn poppy | 9 | 0 | 1 | 1010 |
| cyclamen | 7 | 0 | 3 | 1010 |
| daffodil | 4 | 1 | 6 | 1009 |
| desert-rose | 8 | 0 | 2 | 1010 |
| english marigold | 8 | 1 | 2 | 1009 |
| fire lily | 10 | 1 | 0 | 1009 |
| foxglove | 10 | 1 | 0 | 1009 |
| frangipani | 10 | 0 | 0 | 1010 |
| fritillary | 10 | 1 | 0 | 1009 |
| garden phlox | 5 | 4 | 5 | 1006 |
| gaura | 9 | 3 | 1 | 1007 |
| gazania | 9 | 0 | 1 | 1010 |
| geranium | 7 | 2 | 3 | 1008 |
| giant white arum lily | 10 | 0 | 0 | 1010 |
| globe thistle | 9 | 1 | 1 | 1009 |
| globe-flower | 9 | 2 | 1 | 1008 |
| grape hyacinth | 10 | 0 | 0 | 1010 |
| great masterwort | 10 | 0 | 0 | 1010 |
| hard-leaved pocket orchid | 10 | 0 | 0 | 1010 |
| hibiscus | 10 | 6 | 0 | 1004 |
| hippeastrum | 10 | 1 | 0 | 1009 |
| japanese anemone | 8 | 2 | 2 | 1008 |
| king protea | 10 | 1 | 0 | 1009 |
| lenten rose | 5 | 3 | 5 | 1007 |
| lotus lotus | 8 | 4 | 2 | 1006 |
| love in the mist | 10 | 0 | 0 | 1010 |

| **Kelas** | ***True Positives* (TP)** | ***False Positives* (FP)** | ***False Negatives* (FN)** | ***True Negatives* (TN)** |
|---|---|---|---|---|
| magnolia | 6 | 0 | 4 | 1010 |
| mallow | 5 | 1 | 5 | 1009 |
| marigold | 8 | 0 | 2 | 1010 |
| mexican aster | 9 | 2 | 1 | 1008 |
| mexican petunia | 9 | 0 | 1 | 1010 |
| monkshood | 6 | 2 | 4 | 1008 |
| moon orchid | 10 | 0 | 0 | 1010 |
| morning glory | 8 | 2 | 2 | 1008 |
| orange dahlia | 7 | 0 | 3 | 1010 |
| osteospermum | 9 | 2 | 1 | 1008 |
| oxeye daisy | 10 | 1 | 0 | 1009 |
| passion flower | 10 | 1 | 0 | 1009 |
| pelargonium | 3 | 4 | 7 | 1006 |
| peruvian lily | 10 | 2 | 0 | 1008 |
| petunia | 7 | 5 | 3 | 1005 |
| pincushion flower | 10 | 2 | 0 | 1008 |
| pink primrose | 7 | 3 | 3 | 1007 |
| pink-yellow dahlia | 8 | 1 | 2 | 1009 |
| poinsettia | 7 | 0 | 3 | 1010 |
| primula | 8 | 3 | 2 | 1007 |
| prince of wales feathers | 7 | 1 | 3 | 1009 |
| purple coneflower | 10 | 1 | 0 | 1009 |
| red ginger | 10 | 3 | 0 | 1007 |
| rose | 8 | 8 | 2 | 1002 |
| ruby-lipped cattleya | 9 | 2 | 1 | 1008 |
| siam tulip | 10 | 1 | 0 | 1009 |
| silverbush | 7 | 0 | 3 | 1010 |
| snapdragon | 8 | 3 | 2 | 1007 |
| spear thistle | 10 | 3 | 0 | 1007 |
| spring crocus | 7 | 0 | 3 | 1010 |
| stemless gentian | 8 | 5 | 2 | 1005 |
| sunflower | 10 | 0 | 0 | 1010 |
| sweet pea | 5 | 2 | 5 | 1008 |

| Kelas | *True Positives* (TP) | *False Positives* (FP) | *False Negatives* (FN) | *True Negatives* (TN) |
|---|---|---|---|---|
| sweet william | 8 | 4 | 2 | 1006 |
| sword lily | 5 | 0 | 5 | 1010 |
| thorn apple | 7 | 3 | 3 | 1007 |
| tiger lily | 9 | 0 | 1 | 1010 |
| toad lily | 9 | 0 | 1 | 1010 |
| tree mallow | 9 | 1 | 1 | 1009 |
| tree poppy | 10 | 1 | 0 | 1009 |
| trumpet creeper | 5 | 2 | 5 | 1008 |
| wallflower | 10 | 13 | 0 | 997 |
| water lily | 10 | 4 | 0 | 1006 |
| watercress | 9 | 3 | 1 | 1007 |
| wild pansy | 8 | 0 | 2 | 1010 |
| windflower | 10 | 2 | 0 | 1008 |
| yellow iris | 8 | 1 | 2 | 1009 |

**Tabel 6.6** *Confusion Matrix* pada Pengujian Model InceptionV3

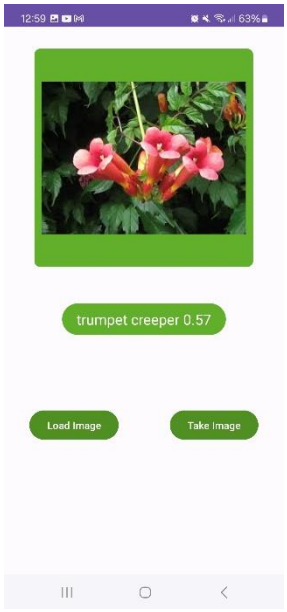
| Kelas | *True Positives* (TP) | *False Positives* (FP) | *False Negatives* (FN) | *True Negatives* (TN) |
|---|---|---|---|---|
| alpine sea holly | 10 | 2 | 0 | 1008 |
| anthurium | 8 | 0 | 2 | 1010 |
| artichoke | 9 | 0 | 1 | 1010 |
| azalea | 6 | 0 | 4 | 1010 |
| ball moss | 8 | 0 | 2 | 1010 |
| balloon flower | 9 | 0 | 1 | 1010 |
| barbeton daisy | 7 | 3 | 3 | 1007 |
| bearded iris | 10 | 4 | 0 | 1006 |
| bee balm | 7 | 1 | 3 | 1009 |
| bird of paradise | 10 | 0 | 0 | 1010 |
| bishop of llandaff | 10 | 7 | 0 | 1003 |
| black-eyed susan | 9 | 1 | 1 | 1009 |
| blackberry lily | 10 | 1 | 0 | 1009 |
| blanket flower | 8 | 0 | 2 | 1010 |
| bolero deep blue | 8 | 0 | 2 | 1010 |

| **Kelas** | ***True Positives* (TP)** | ***False Positives* (FP)** | ***False Negatives* (FN)** | ***True Negatives* (TN)** |
|---|---|---|---|---|
| bougainvillea | 7 | 4 | 3 | 1006 |
| bromelia | 10 | 2 | 0 | 1008 |
| buttercup | 7 | 2 | 3 | 1008 |
| californian poppy | 8 | 1 | 2 | 1009 |
| camellia | 9 | 4 | 1 | 1006 |
| canna lily | 6 | 0 | 4 | 1010 |
| canterbury bells | 7 | 1 | 3 | 1009 |
| cape flower | 10 | 0 | 0 | 1010 |
| carnation | 10 | 2 | 0 | 1008 |
| cautleya spicata | 8 | 4 | 2 | 1006 |
| clematis | 6 | 7 | 4 | 1003 |
| colt's foot | 9 | 0 | 1 | 1010 |
| columbine | 6 | 4 | 4 | 1006 |
| common dandelion | 10 | 3 | 0 | 1007 |
| corn poppy | 7 | 1 | 3 | 1009 |
| cyclamen | 7 | 4 | 3 | 1006 |
| daffodil | 4 | 1 | 6 | 1009 |
| desert-rose | 7 | 0 | 3 | 1010 |
| english marigold | 7 | 1 | 3 | 1009 |
| fire lily | 9 | 0 | 1 | 1010 |
| foxglove | 9 | 0 | 1 | 1010 |
| frangipani | 10 | 1 | 0 | 1009 |
| fritillary | 8 | 0 | 2 | 1010 |
| garden phlox | 7 | 5 | 3 | 1005 |
| gaura | 10 | 2 | 0 | 1008 |
| gazania | 8 | 2 | 2 | 1008 |
| geranium | 5 | 0 | 5 | 1010 |
| giant white arum lily | 8 | 0 | 2 | 1010 |
| globe thistle | 9 | 1 | 1 | 1009 |
| globe-flower | 8 | 0 | 2 | 1010 |
| grape hyacinth | 10 | 1 | 0 | 1009 |
| great masterwort | 10 | 2 | 0 | 1008 |
| hard-leaved pocket orchid | 10 | 0 | 0 | 1010 |

| **Kelas** | ***True Positives* (TP)** | ***False Positives* (FP)** | ***False Negatives* (FN)** | ***True Negatives* (TN)** |
|---|---|---|---|---|
| hibiscus | 10 | 2 | 0 | 1008 |
| hippeastrum | 9 | 3 | 1 | 1007 |
| japanese anemone | 7 | 1 | 3 | 1009 |
| king protea | 8 | 2 | 2 | 1008 |
| lenten rose | 7 | 3 | 3 | 1007 |
| lotus lotus | 8 | 4 | 2 | 1006 |
| love in the mist | 8 | 0 | 2 | 1010 |
| magnolia | 7 | 0 | 3 | 1010 |
| mallow | 4 | 0 | 6 | 1010 |
| marigold | 3 | 0 | 7 | 1010 |
| mexican aster | 9 | 6 | 1 | 1004 |
| mexican petunia | 8 | 2 | 2 | 1008 |
| monkshood | 8 | 2 | 2 | 1008 |
| moon orchid | 10 | 1 | 0 | 1009 |
| morning glory | 10 | 4 | 0 | 1006 |
| orange dahlia | 4 | 1 | 6 | 1009 |
| osteospermum | 5 | 0 | 5 | 1010 |
| oxeye daisy | 10 | 2 | 0 | 1008 |
| passion flower | 10 | 2 | 0 | 1008 |
| pelargonium | 5 | 1 | 5 | 1009 |
| peruvian lily | 10 | 0 | 0 | 1010 |
| petunia | 7 | 4 | 3 | 1006 |
| pincushion flower | 9 | 1 | 1 | 1009 |
| pink primrose | 10 | 2 | 0 | 1008 |
| pink-yellow dahlia | 7 | 0 | 3 | 1010 |
| poinsettia | 7 | 2 | 3 | 1008 |
| primula | 8 | 8 | 2 | 1002 |
| prince of wales feathers | 7 | 0 | 3 | 1010 |
| purple coneflower | 10 | 2 | 0 | 1008 |
| red ginger | 9 | 0 | 1 | 1010 |
| rose | 10 | 12 | 0 | 998 |
| ruby-lipped cattleya | 8 | 0 | 2 | 1010 |
| siam tulip | 7 | 1 | 3 | 1009 |

| **Kelas** | ***True Positives* (TP)** | ***False Positives* (FP)** | ***False Negatives* (FN)** | ***True Negatives* (TN)** |
|---|---|---|---|---|
| silverbush | 5 | 0 | 5 | 1010 |
| snapdragon | 7 | 6 | 3 | 1004 |
| spear thistle | 10 | 1 | 0 | 1009 |
| spring crocus | 9 | 5 | 1 | 1005 |
| stemless gentian | 9 | 0 | 1 | 1010 |
| sunflower | 10 | 2 | 0 | 1008 |
| sweet pea | 4 | 0 | 6 | 1010 |
| sweet william | 8 | 4 | 2 | 1006 |
| sword lily | 8 | 4 | 2 | 1006 |
| thorn apple | 7 | 2 | 3 | 1008 |
| tiger lily | 10 | 1 | 0 | 1009 |
| toad lily | 10 | 0 | 0 | 1010 |
| tree mallow | 8 | 0 | 2 | 1010 |
| tree poppy | 9 | 0 | 1 | 1010 |
| trumpet creeper | 6 | 1 | 4 | 1009 |
| wallflower | 10 | 22 | 0 | 988 |
| water lily | 10 | 2 | 0 | 1008 |
| watercress | 8 | 1 | 2 | 1009 |
| wild pansy | 6 | 2 | 4 | 1008 |
| windflower | 9 | 0 | 1 | 1010 |
| yellow iris | 8 | 0 | 2 | 1010 |

# LAMPIRAN C
# Contoh Hasil Uji Klasifikasi pada Perangkat Android

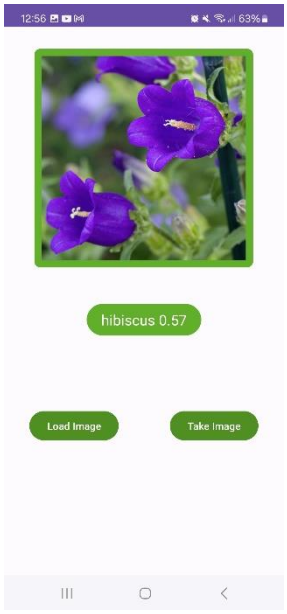



**Gambar 6.1** Contoh Hasil Uji Klasifikasi Bunga *Trumpet Creeper*

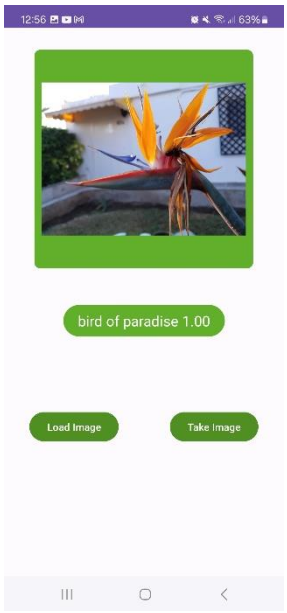

**Gambar 6.2** Contoh Hasil Uji Klasifikasi Bunga *Fire Lily*


**Gambar 6.3** Contoh Hasil Uji Klasifikasi Bunga *Petunia*


**Gambar 6.4** Contoh Hasil Uji Klasifikasi Bunga *Frangipani*




**Gambar 6.5** Contoh Hasil Uji Klasifikasi Bunga *Canterbury Bells*




**Gambar 6.6** Contoh Hasil Uji Klasifikasi Bunga *Bird of Paradise*


**Gambar 6.7** Contoh Hasil Uji Klasifikasi Bunga *Snapdragon*


**Gambar 6.8** Contoh Hasil Uji Klasifikasi Bunga *Bromelia*


**Gambar 6.9** Contoh Hasil Uji Klasifikasi Bunga *Artichoke*


**Gambar 6.10** Contoh Hasil Uji Klasifikasi Bunga *Pink-Yellow Dahlia*

*(Halaman ini sengaja dikosongkan)*

# BIODATA PENULIS

Penulis dilahirkan di Jombang, 28 September 2001, merupakan anak ketiga dari 3 bersaudara. Penulis telah menempuh pendidikan formal yaitu di SD Katolik Wijana, SMP Negeri 2 Jombang, dan SMA Negeri 6 Surabaya. Setelah lulus dari SMA tahun 2020, Penulis mengikuti SBMPTN dan diterima di Departemen Teknik Informatika FTEIC - ITS pada tahun 2020 dan terdaftar dengan NRP 5025201219.

Di Departemen Teknik Informatika penulis sempat aktif di beberapa kegiatan mahasiswa seperti staff departemen Kaderisasi dan Pemetaan Mahasiswa Himpunan Mahasiswa Teknik Computer – Informatika (HMTC) tahun 2022, ketua departemen Pengembangan Sumber Daya Mahasiswa Himpunan Mahasiswa Teknik Computer – Informatika (HMTC) tahun 2023, Staf Ahli SCHEMATIC 2022, asisten dosen pada mata kuliah dasar pemrograman, kecerdasan buatan, sistem operasi, dan jaringan komputer. Selain itu, penulis pernah melakukan studi independent sebagai Cloud Computing Cohort pada Bangkit Academy 2023 Batch 1.