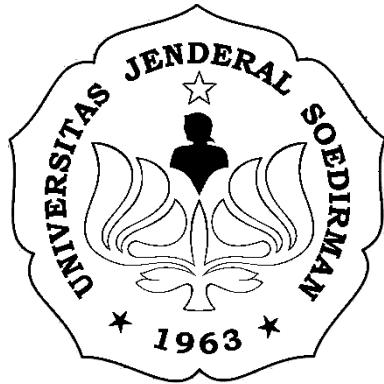


## LAPORAN TUGAS AKHIR

### **IDENTIFIKASI PENYAKIT PADA DAUN TOMAT DAN DAUN SINGKONG BERDASARKAN CITRA DAUN MENGGUNAKAN METODE *CONVOLUTIONAL NEURAL NETWORK (CNN)* BERBASIS ANDROID**

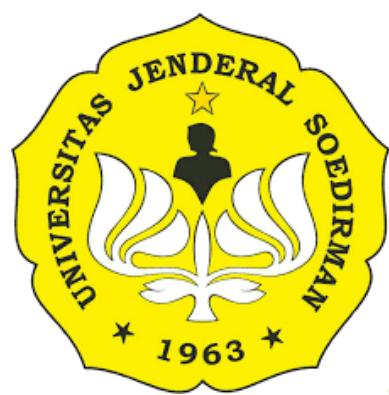
Disusun untuk memenuhi persyaratan memperoleh gelar sarjana  
di Jurusan Teknik Elektro Universitas Jenderal Soedirman



Disusun oleh:

Jepri  
H1A016025

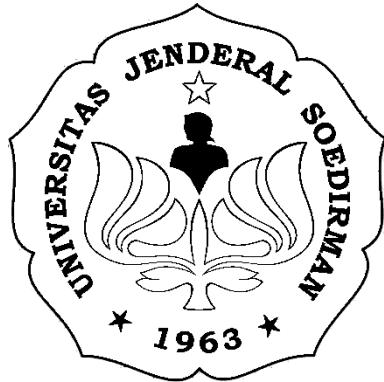
**KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN  
UNIVERSITAS JENDERAL SOEDIRMAN  
FAKULTAS TEKNIK  
JURUSAN/PROGRAM STUDI TEKNIK ELEKTRO  
PURBALINGGA  
2020**



## LAPORAN TUGAS AKHIR

# **IDENTIFIKASI PENYAKIT PADA DAUN TOMAT DAN DAUN SINGKONG BERDASARKAN CITRA DAUN MENGGUNAKAN METODE *CONVOLUTIONAL NEURAL NETWORK (CNN)* BERBASIS ANDROID**

Disusun untuk memenuhi persyaratan memperoleh gelar sarjana  
di Jurusan Teknik Elektro Universitas Jenderal Soedirman



Disusun oleh:

Jepri  
H1A016025

**KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN  
UNIVERSITAS JENDERAL SOEDIRMAN  
FAKULTAS TEKNIK  
JURUSAN/PROGRAM STUDI TEKNIK ELEKTRO  
PURBALINGGA  
2020**

## **HALAMAN PENGESAHAN**

Tugas Akhir dengan Judul:

### **IDENTIFIKASI PADA PENYAKIT DAUN TOMAT DAN DAUN SINGKONG BERDASARKAN CITRA DAUN MENGGUNAKAN METODE *CONVOLUTIONAL NEURAL NETWORK (CNN)* BERBASIS ANDROID**

Disusun oleh:

Jepri  
H1A016025

Diajukan untuk memenuhi salah satu persyaratan  
memperoleh gelar Sarjana Teknik pada  
Jurusang/Program Studi Teknik Elektro  
Fakultas Teknik  
Universitas Jenderal Soedirman

Diterima dan disetujui  
Pada Tanggal : \_\_\_\_\_

Pembimbing I

Pembimbing II

Imron Rosyadi, S.T., M.Sc                   Farida Asriani, S.Si., M.T.  
(NIP : 197909242003121003)                   (NIP : 197502012000032005)

Mengetahui:  
Dekan Fakultas Teknik

Dr. Eng Suroso, S.T., M.Eng.  
NIP. 197812242001121002

## **HALAMAN PERNYATAAN**

Dengan ini saya menyatakan bahwa dalam Laporan Tugas Akhir dengan judul "**IDENTIFIKASI PADA PENYAKIT DAUN TOMAT DAN DAUN SINGKONG BERDASARKAN CITRA DAUN MENGGUNAKAN METODE CONVOLUTIONAL NEURAL NETWORK (CNN) BERBASIS ANDROID**" ini tidak terdapat karya yang pernah diajukan untuk memperoleh gelar kesarjanaan di suatu Perguruan Tinggi, dan sepanjang pengetahuan saya juga tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis diacu dalam naskah ini dan disebutkan dalam daftar pustaka.

Purbalingga, 01 Februari 2020

[materai sesuai ketentuan UU]  
Ttd.

Jepri  
NIM. H1A016025

## **HALAMAN MOTTO DAN PERSEMBAHAN**

### **MOTTO**

*“sekarang atau tidak sama sekali”.*

### **PERSEMBAHAN**

Laporan tugas akhir ini dipersembahkan untuk:

1. Allah SWT atas berkat rahmat, karunia serta petunjuk-Nya selama pelaksanaan tugas akhir hingga pembuatan laporan dengan lancar.
2. Bapak, Ibu, Kakak dan saudara-saudara penulis atas limpahan doa, dukungan dan semangat.
3. Ibu Farida Asriani, S.Si M.T., selaku Ketua Jurusan Teknik Elektro Universitas Jenderal Soedirman dan dosen pembimbing II.
4. Azis Wisnu Widhi Nugraha, S.T., M.Eng. selaku dosen pembimbing akademik.
5. Ari Fadli, S.T., M.Eng. selaku komisi tugas akhir.
6. Bapak Imron Rosyadi, selaku Dosen Pembimbing I.
7. Bapak dan Ibu dosen Teknik Elektro Universitas Jenderal Soedirman
8. Semua teman - teman Teknik Elektro angkatan 2016, terutama Vidi Fitriansyah Hidarlan dan Fendy Prayogi selaku rekan seperjuangan selama menjalani penelitian Tugas Akhir
9. Teman-teman Pondok, IMMANT Purwokerto dan MGR.
10. Semua pihak yang telah membantu baik secara langsung maupun tidak langsung dalam pelaksanaan dan penulisan laporan tugas akhir.

## RINGKASAN

### **IDENTIFIKASI PENYAKIT PADA DAUN TOMAT DAN DAUN SINGKONG BERDASARKAN CITRA DAUN MENGGUNAKAN METODE *CONVOLUTIONAL NEURAL NETWORK (CNN)* BERBASIS ANDROID**

Jepri

Pertanian memiliki arti penting dalam pembangunan perekonomian bangsa. Pemerintah telah menetapkan pertanian sebagai prioritas utama pembangunan di masa mendatang. Sektor pertanian tidak saja sebagai penyedia kebutuhan pangan bagi penduduknya, tetapi juga merupakan sumber pendapatan ekspor (devisa) serta pendorong dan penarik bagi tumbuhnya sektor-sektor ekonomi lainnya. Misalnya pada produksi tanaman tomat dan singkong yang memiliki penjualan yang tinggi.

Salah satu penyebab utama penurunan produksi hasil tersebut yaitu munculnya berbagai macam penyakit. Identifikasi penyakit tanaman biasanya dilakukan di laboratorium. Dalam hal ini penulis membuat penelitian tentang identifikasi penyakit tanaman menggunakan *deep learning* berbasis android agar mudah digunakan menggunakan *Smartphone*. Cara penggunaannya hanya dengan memasukan obyek yang akan di identifikasi kedalam aplikasi *Plant Diseases* melalui media pengambilan gambar secara langsung menggunakan kamera atau *import gallery*.

Berdasarkan hasil pelatihan dan pengujian yang dilakukan menggunakan tiga arsitektur (VGG16, *InceptionV3*, dan *mobileNet*) proses identifikasi menggunakan arsitektur *mobileNet* memiliki akurasi yang lebih tinggi, *mobileNet* memiliki akurasi sebesar 95,33%, arsitektur *Inception* 92,67%, dan VGG16 76,67%. Selain itu *mobileNet* juga memiliki respon deteksi yang lebih cepat yaitu memiliki respon deteksi rata-rata sebesar 134,66, sedangkan pada arsitektur *Inception* dan VGG16 masing masing sebesar 643,6 dan 789,03.

Kata kunci : Penyakit Tanaman, *Deep Learning*, CNN.

## **SUMMARY**

### **DISEASES IDENTIFICATION OF TOMATO LEAVES AND CASSAVA LEAVES IMAGE-BASED USING CONVOLUTIONAL ARTIFICIAL NEURAL NETWORK (CNN) METHOD ON ANDROID**

Jepri

*Agriculture has an important role in the economic development of the nation. The government has set agriculture as priority development in the future. Agriculture not only as a provider of food for the population, but also a source of export revenue (foreign exchange), and push and pull of the growth sectors of the economy. For example in the production of tomatoes and potatoes that have high sales.*

*One of the main causes of the decline in production of these results is the emergence of various diseases. Identification of plant diseases is usually done in a laboratory. In this case the authors make research on the identification of plant diseases using Android-based deep learning so that it is easy to use using smartphones. How to use it only by entering the object that will be identified into the application Plant Diseases through media shooting directly using a camera or import gallery.*

*Based on the results of training and testing conducted using three architectures (VGG16, InceptionV3, and mobileNet) the approval process using mobileNet architecture has higher accuracy, mobileNet has an accuracy of 95.33%, Inception Architecture 92.67%, and VGG16 76.67 %. In addition, mobileNet also has a faster detection, which has an average detection of 134.66, while the Inception and VGG16 architectures are 643.6 and 789.03, respectively..*

*Keywords : Plant disease, Deep Learning, CNN*

## PRAKATA

Puji syukur kehadiratan Allah S.W.T. yang telah melimpahkan berkah dan rahmat-Nya laporan tugas akhir tentang “**IDENTIFIKASI PADA PENYAKIT DAUN TOMAT DAN DAUN SINGKONG BERDASARKAN CITRA DAUN MENGGUNAKAN METODE CONVOLUTIONAL NEURAL NETWORK (CNN) BERBASIS ANDROID**” ini dapat disusun. adapun maksud dan tujuan penyusunan laporan tugas akhir ini adalah untuk melengkapi salah satu syarat dalam menempuh Pendidikan sarjana pada Jurusan Teknik Elektro Fakultas Teknik Universitas Jenderal Soedirman.

Pada kesempatan ini, penulis mengucapkan terima kasih kepada seluruh pihak yang telah membantu dalam penyusunan laporan tugas akhir ini, yaitu:

1. Allah SWT atas berkat rahmat, karunia serta petunjuk-Nya selama pelaksanaan tugas akhir hingga pembuatan laporan dengan lancar.
2. Bapak, Ibu, Kakak dan saudara-saudara penulis atas limpahan doa, dukungan dan semangat.
3. Farida Asriani, S.Si., M.T. selaku kepala jurusan Teknik Elektro Universitas Jenderal Soedirman.
4. Azis Wisnu Widhi Nugraha, S.T., M.Eng. selaku dosen pembimbing akademik.
5. Ari Fadli, S.T., M.Eng. selaku komisi tugas akhir.
6. Imron Rosyadi, S.T., M.Sc selaku dosen pembimbing I tugas akhir.
7. Farida Asriani, S.Si., M.T. selaku dosen pembimbing II tugas akhir.
8. Dosen Teknik Eletro Universitas Jenderal Soedirman.

Penulis hanya bisa memanjatkan doa semoga Allah SWT berkenan untuk membalas budi baik semua pihak yang telah membantu penyusunan laporan tugas akhir ini. Semoga laporan tugas akhir ini dapat bermanfaat bagi penulis dan pembaca. Penulis menyadari bahwa laporan ini masih jauh dari kata sempurna, oleh karena itu kritik serta saran pembaca sangat diharapkan untuk menghasilkan laporan yang lebih baik.

Purbalingga, 01 Februari 2020

Jepri

## DAFTAR ISI

HALAMAN JUDUL.....	i
HALAMAN PENGESAHAN.....	ii
HALAMAN PERNYATAAN .....	iii
HALAMAN MOTTO DAN PERSEMBAHAN.....	iv
RINGKASAN .....	v
<i>SUMMARY</i> .....	vi
PRAKATA.....	vii
DAFTAR ISI .....	viii
DAFTAR TABEL.....	xi
DAFTAR GAMBAR .....	xii
DAFTAR LAMPIRAN .....	xiv
DAFTAR ISTILAH DAN SINGKATAN .....	xv
BAB 1 PENDAHULUAN .....	1
1.1 Latar Belakang .....	1
1.2 Rumusan Masalah .....	3
1.3 Batasan Masalah.....	3
1.4 Tujuan dan Manfaat.....	4
1.4.1 Tujuan .....	4
1.4.2 Manfaat.....	4
1.5 Sistematika Penulisan.....	5
BAB 2 TINJAUAN PUSTAKA.....	6
2.1 Penelitian Terdahulu .....	6
2.2 Pengolahan Citra .....	6
2.3 <i>Deep Leraning</i> .....	7
2.4 <i>Convolutional Neural Network (CNN)</i> .....	8
2.4.1 <i>Input Layer</i> .....	9
2.4.2 <i>Convolution Layer</i> .....	9
2.4.3 <i>Activation Layer</i> .....	10
2.4.4 <i>Pooling Layer</i> .....	11
2.4.5 <i>Fully Connected Layer</i> .....	12
2.5 Arsitektur CNN.....	14
2.5.1 Arsitektur VGG16.....	14
2.5.2 Arsitektur <i>MobileNet</i> .....	16

2.5.3 Arsitektur <i>InceptionV3</i> .....	18
2.6 <i>Google Colaboratory</i> .....	19
2.7 <i>Python</i> .....	20
2.8 <i>Framework Keras</i> .....	21
2.9 <i>Tensor Flow Lite</i> .....	21
2.10 <i>Android</i> .....	23
2.10.1 Struktur <i>Project</i> .....	24
2.10.2 Tampilan Antarmuka Pengguna.....	25
2.11 Penyakit Pada Tumbuhan .....	27
2.11.1 Penyakit Tanaman Tomat.....	27
2.11.2 Penyakit Tanaman Pada Daun Singkong .....	34
BAB 3 METODOLOGI PENELITIAN.....	38
3.1 Waktu dan Tempat Penelitian .....	38
3.2 Alat dan Bahan .....	38
3.2.1 Perangkat keras.....	38
3.2.2 Perangkat Lunak .....	38
3.2.3 <i>Dataset</i> .....	39
3.3 Metode Penelitian.....	39
3.3.1 Persiapan.....	40
3.3.2 Pelaksanaan.....	40
3.3.3 Evaluasi.....	41
3.3.4 Tahap Akhir.....	42
3.4 Alur Penelitian .....	42
3.5 Waktu dan Jadwal Penelitian.....	42
BAB 4 HASIL DAN PEMBAHASAN.....	43
4.1 Pengumpulan <i>Dataset</i> .....	43
4.1.1 <i>Dataset</i> Daun Tomat .....	43
4.1.2 <i>Dataset</i> Daun Singkong .....	44
4.2 Hasil Pelatihan dan Pengujian Pada <i>Google Colaboratory</i> .....	44
4.2.1 Arsitektur VGG16.....	44
4.2.2 Arsitektur <i>Inception</i> Versi 3 .....	48
4.2.3 Arsitektur <i>MobileNet</i> Versi 1 .....	51
4.3 Hasil Pengujian Menggunakan <i>Android</i> .....	55
4.3.1 Pengujian <i>Android</i> Menggunakan VGG16.....	55
4.3.2 Pengujian <i>Android</i> Menggunakan <i>Inception</i> Versi 3 .....	60
4.3.3 Pengujian <i>Android</i> Menggunakan <i>MobileNet</i> Versi 1 .....	66
4.4 Perbandingan Tiga Arsitektur.....	71
BAB 5 PENUTUP .....	75
5.1 Kesimpulan.....	75

5.2 Saran .....	76
DAFTAR PUSTAKA .....	77
LAMPIRAN .....	79
BIODATA PENULIS .....	134

## **DAFTAR TABEL**

Tabel 2.1 Arsitektur MobileNet.....	17
Tabel 3.1 Jadwal Penelitian.....	42
Tabel 4.1 Dataset Daun Tomat .....	43
Tabel 4.2Dataset Daun Singkong.....	44
Tabel 4.3 Data Uji Android Menggunakan Arsitektur VGG16.....	55
Tabel 4.4 Data Uji Android Menggunakan Arsitektur Inception .....	61
Tabel 4.5 Data Uji Android Menggunakan Arsitektur MobileNetV1 .....	66

## DAFTAR GAMBAR

Gambar 2.1 Perbedaan antara lapisan layer pada Jaringan Saraf Tiruan dengan Jaringan Deep Learning .....	7
Gambar 2.2 Alur Proses metode CNN .....	9
Gambar 2.3 Proses Input Layer.....	9
Gambar 2.4 Alur Convolution Layer .....	10
Gambar 2.5 Activation functions .....	11
Gambar 2.6 FeatMap.....	12
Gambar 2.7 Proses Pooling.....	12
Gambar 2.8 Fully Connected Layer.....	13
Gambar 2.9 Contoh CNN.....	13
Gambar 2.10 Arsitektur VGG16 .....	14
Gambar 2.11 Arsitektur Konversi model ke TensorFlow Lite .....	23
Gambar 2.12 File project dalam tampilan Android.....	25
Gambar 2.13 Tampilan Antarmuka Pengguna .....	26
Gambar 2.14 Bacterial Spot.....	28
Gambar 2.15 Early Blight .....	29
Gambar 2.16 Late Blight.....	29
Gambar 2.17 Leaf Mold.....	30
Gambar 2.18 Mosaic Virus .....	31
Gambar 2.19 Septoria Leaf Spot.....	32
Gambar 2.20 Two Spotted Spider Mite.....	32
Gambar 2.21 Target Spot .....	33
Gambar 2.22 Yellow Leaf Curl Virus.....	34
Gambar 2.23 Cassava Bacteial Blight.....	35
Gambar 2.24 Cassava Brown Streak Disease .....	35
Gambar 2.25 Cassava Green Mottle .....	36
Gambar 2.26 Cassava Mosaic Disease.....	37
Gambar 3.1 Tahapan Penelitian .....	39
Gambar 3.2 Tahap Pelaksanaan .....	40
Gambar 4.1 Kurva Akurasi Training dan Validasi menggunakan Arsitektur VGG16.....	45
Gambar 4.2 Kurva Loss training dan validasi menggunakan Arsiektur Vgg16 ..	46
Gambar 4.3 Confusion Matrix menggunakan arsitektur VGG16 .....	47
Gambar 4.4 Kurva akurasi training dan validasi Menggunakan Arsitektur InceptionV3.....	49
Gambar 4.5 Kurva Loss Training dan validasi Menggunakan arsitektur InceptionV3.....	50
Gambar 4.6 Confusion Matrix menggunakan Asritektur InceptionV3 .....	51
Gambar 4.7 Kurva Akurasi Training Dan Validasi Menggunakan Arsitektur MobileNetV1 .....	52
Gambar 4.8 Kurva Loss Training dan Validasi menggunakan Arsitektur MobileNetV1 .....	53
Gambar 4.9 Confusion Matrix Menggunakan Arsitektur MobileNetV1 .....	54

Gambar 4.10 Confusion Matrix Pengujian Pada Android Menggunakan Arsitektur VGG16.....	60
Gambar 4.11 Confusion Matrix Pengujian Pada Android Menggunakan Arsitektur InceptionV3.....	65
Gambar 4.12Confusion Matrix Pengujian Pada Android Menggunakan Arsitektur MobileNetV1 .....	71
Gambar 4.13 Kurva Akurasi Perbandingan Arsitektur VGG16, InceptionV3 dan MobileNetV1 .....	72
Gambar 4.14 Kurva kesalahan Perbandingan Arsitektur VGG16, InceptionV3 dan MobileNetV1 .....	73

## **DAFTAR LAMPIRAN**

Lampiran 1 Sourecode Identifikasi Penyakit Tanaman Menggunakan Arsitektur VGG16 .....	79
Lampiran 2 Sourcecode Identifikasi Penyakit Tanaman Menggunakan Arsitektur InceptionV3 .....	85
Lampiran 3 SourceCode Identifikasi Penyakit Tanaman Menggunakan Arsitektur MobileNet.....	92
Lampiran 4 SouceCode Pada Android Studio.....	98
Lampiran 5 Screenshot Data hasil uji Android menggunakan arsitektur VGG16 .....	111
Lampiran 6 Screenshot Data hasil uji Android menggunakan arsitektur InceptionV3 .....	118
Lampiran 7 Screenshot Data hasil uji Android menggunakan arsitektur MobileNet .....	126

## DAFTAR ISTILAH DAN SINGKATAN

*Activation Function:* Untuk membuat neural network menjadi non-linear.

<i>ANN</i>	: <i>Artificial Neural Network</i>
<i>API</i>	: Application Programming Interface
<i>Cloud</i>	: Metafora dari internet, sebagaimana awan yang sering digambarkan di diagram jaringan komputer
<i>CNN</i>	: <i>Convolutional Neural Network</i>
<i>Conv2D</i>	: untuk menjalankan operasi konvolusional pada gambar <i>train</i>
<i>CPU</i>	: Central Processing Unit
<i>Dense</i>	: untuk menjalankan full connection neural network
<i>Flatten</i>	: reshape feature map menjadi sebuah vector agar bisa kita gunakan sebagai input dari <i>fully-connected layer</i> .
<i>GPU</i>	: <i>Graphics Processing Unit</i>
<i>Jupyter Notebook</i>	: Sebuah aplikasi <i>open source</i> yang fungsinya untuk kita membuat dan berbagi dokumen yang berisi <i>live code</i> , persamaan, visualisasi dan teks penjelasan
<i>Lesi</i>	: Istilah kedokteran untuk merujuk pada keadaan jaringan yang abnormal
<i>MaxPooling2D</i>	: untuk mengaktifkan operasi <i>pooling</i>
<i>Pixel</i>	: Unsur gambar atau representasi sebuah titik terkecil dalam sebuah gambar grafis yang dihitung per inci
<i>Overfitting</i>	: Kondisi model pembelajaran mesin sangat detail (berlebihan) yang dapat menurunkan tingkat akurasi
<i>RStudio</i>	: Merupakan integrated development environment (IDE) khusus bagi bahasa pemrograman R
<i>Tungau</i>	: Sekelompok hewan kecil bertangkai delapan yang, bersama-sama dengan caplak, menjadi anggota <i>superordo Acarina</i> .

## **BAB 1**

### **PENDAHULUAN**

Bab ini akan membahas tentang latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, lokasi, waktu, dan metode pelaksanaan laporan tugas akhir.

#### **1.1 Latar Belakang**

Pertanian memiliki arti penting dalam pembangunan perekonomian bangsa. Pemerintah telah menetapkan pertanian sebagai prioritas utama pembangunan di masa mendatang. Sektor pertanian tidak saja sebagai penyedia kebutuhan pangan bagi penduduknya, tetapi juga merupakan sumber pendapatan ekspor (devisa) serta pendorong dan penarik bagi tumbuhnya sektor-sektor ekonomi lainnya. Pembangunan pertanian yang dikelola dengan baik dan bijak akan dapat meningkatkan pertumbuhan dan sekaligus pemerataan ekonomi secara berkelanjutan, mengatasi kemiskinan dan pengangguran yang pada akhirnya menyejahterakan masyarakat Indonesia secara keseluruhan [1].

Secara umum, sektor pertanian terdiri dari beberapa sub sektor, yaitu sub sektor pangan, hortikultura, dan perkebunan. Salah satu sub sektor pertanian yang cukup penting dan merupakan salah satu penyumbang peningkatan PDB Indonesia adalah sub sektor hortikultura ( buah dan sayuran). Tanaman tomat (*Lycopersicum esculentum Mill.*) merupakan salah satu komoditas sayuran yang berpotensi multiguna. Selain itu komoditas tomat juga salah satu komoditas sayuran yang menjadi penyumbang ekspor selain kol, wortel dan kentang [2].

Selain tomat, komoditas pertanian pangan yang mempunyai prospek untuk dikembangkan dalam rangka memenuhi kebutuhan pasar domestik maupun internasional adalah singkong. Singkong (*Manihot utilissima*) merupakan makanan pokok ketiga setelah padi dan jagung bagi masyarakat Indonesia. Pada tahun 2011 produksi singkong di Indonesia mencapai 24.044.025 ton, sedangkan pada tahun 2012 meningkat menjadi 24.177.327 ton [3].

Salah satu penyebab utama penurunan produksi hasil tersebut yaitu munculnya berbagai macam penyakit. Tanaman dikatakan sakit bila ada perubahan seluruh atau sebagian organ-organ tanaman yang menyebabkan terganggunya kegiatan fisiologis sehari-hari. Secara singkat, penyakit tanaman adalah penyimpangan dari keadaan normal. Penyebab sakitnya tanaman bermacam-macam. Ada yang disebabkan oleh cendawan, bakteri, virus, dan lain-lain [4].

Identifikasi penyakit tanaman biasanya dilakukan di laboratorium. Dalam hal ini penulis membuat penelitian tentang identifikasi penyakit tanaman menggunakan *deep learning* berbasis android agar mudah digunakan menggunakan *Smartphone*.

*Deep learning* adalah sub bidang khusus dari pembelajaran mesin (*machine learning*) yang merupakan pandangan baru tentang representasi pembelajaran dari data yang menekankan pada lapisan-lapisan (layer) pembelajaran berturut turut dari representasi yang semakin berarti [5]. *Deep learning* banyak digunakan untuk klasifikasi obyek berdasarkan citra atau gambar dengan cara mempelajari representasi atau fitur data secara otomatis.

*Convolutional Neural Network* (CNN) adalah salah satu jenis *neural network* yang biasa digunakan pada data *image*. CNN bisa digunakan untuk mendeksi dan mengenali *object* pada sebuah *image* [6].

Penanganan terhadap penyakit tanaman harus segera dilakukan. Oleh karena itu, identifikasi penyakit tanaman menggunakan metode *Deep Learning* sangat diperlukan sebagai langkah awal untuk mengetahui penyakit.

Pada Tugas Akhir ini penulis mengambil topik identifikasi penyakit tanaman. Namun, tanaman yang di ambil untuk dijadikan bahan uji baru hanya penyakit tanaman pada daun tomat dan daun Singkong. Sehingga diperoleh judul “IDENTIFIKASI PENYAKIT PADA DAUN TOMAT DAN DAUN SINGKONG BERDASARKAN CITRA DAUN MENGGUNAKAN METODE CONVOLUTIONAL NEURAL NETWORK (CNN) BERBASIS ANDROID”.

## 1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, rumusan masalah yang diajukan adalah:

1. Bagaimana perancangan arsitektur CNN untuk identifikasi penyakit tanaman?
2. Bagaimana pelatihan dan pengujian untuk identifikasi dengan metode CNN?
3. Bagaimana pengujian identifikasi pada aplikasi *Android*?

## 1.3 Batasan Masalah

Pada penelitian tugas akhir yang dilakukan memiliki batasan masalah sebagai berikut :

1. Identifikasi Penyakit tanaman hanya meliputi sepuluh penyakit tanaman pada daun tomat dan lima penyakit pada daun singkong.

2. Metode *deep learning* yang digunakan adalah CNN (*Convolutional Neural Network*).
3. Menggunakan Bahasa Pemrograman *Python*
4. Infrastruktur yang digunakan adalah *Google Colaboratory*.
5. Menggunakan *Framework Keras*.
6. Aplikasi dibuat menggunakan *Android Studio* untuk *smartphone*.

## **1.4 Tujuan dan Manfaat**

### **1.4.1 Tujuan**

Maksud penulis melaksanakan tugas akhir ini adalah sebagai berikut :

1. Merancang arsitektur *CNN* untuk identifikasi penyakit tanaman pada daun tomat dan daun singkong dan penerapannya pada aplikasi *Android*.
2. Melatih arsitektur *CNN* untuk identifikasi penyakit tanaman.
3. Menguji arsitektur *CNN* identifikasi penyakit tanaman yang sudah dilatih sebelumnya.

### **1.4.2 Manfaat**

Manfaat yang diharapkan dalam pelaksanaan tugas akhir ini adalah sebagai berikut :

1. Bagi Mahasiswa Mampu menerapkan ilmu yang didapat pada mata kuliah yang bersangkutan untuk menyelesaikan tugas akhir.
2. Memudahkan petani dalam mengidentifikasi penyakit tanaman sehingga bisa dilakukan penanganan.

### **1.5 Sistematika Penulisan**

Dalam penelitian yang akan dikerjakan penulis disusun sistematika penulisan yang terdiri dari BAB I Pendahuluan, BAB II Tinjauan pustaka, BAB III Metode Penelitian, BAB IV Hasil dan Pembahasan, dan BAB 5 Penutup.

## **BAB 2**

### **TINJAUAN PUSTAKA**

Tinjauan pustaka berisi materi-materi yang mendukung dalam penyusunan laporan tugas akhir ini.

#### **2.1 Penelitian Terdahulu**

Dalam beberapa jurnal penelitian terdahulu yang membahas tentang klasifikasi obyek dengan citra dapat dilihat dari beberapa jurnal sebagai berikut :

1. I Wayan Suartika E. P, Arya Yudhi Wijaya, dan Rully Soelaiman penelitiannya yang berjudul “Klasifikasi Citra Menggunakan *Convolutional Neural Network* (CNN) pada *Caltech 101*” membahas tentang melakukan klasifikasi 5 jenis unggas dengan 390 citra dari data Caltech 101 menggunakan 3 lapisan CNN, yaitu *Convolution Layer*, *Subsampling Layer*, dan *Fully Connected Layer*.
2. Laila Marifatul Azizah, Sitti Fadillah Umayah, Febriyana Fajar penelitiannya yang berjudul ‘‘Deteksi Kecacatan Permukaan Buah Manggis Menggunakan Metode *Deep Learning* dengan Konvolusi *Multilayer*’’ membahas data manggis dengan permukaan cacat dan gambar manggis dengan permukaan yang baik. Dilakukan *training* data dengan 120 data gambar yang terdiri dari 4 folder. Setiap folder memiliki 30 gambar dengan 24 gambar *fine* dan 6 *defect* dalam setiap folder.

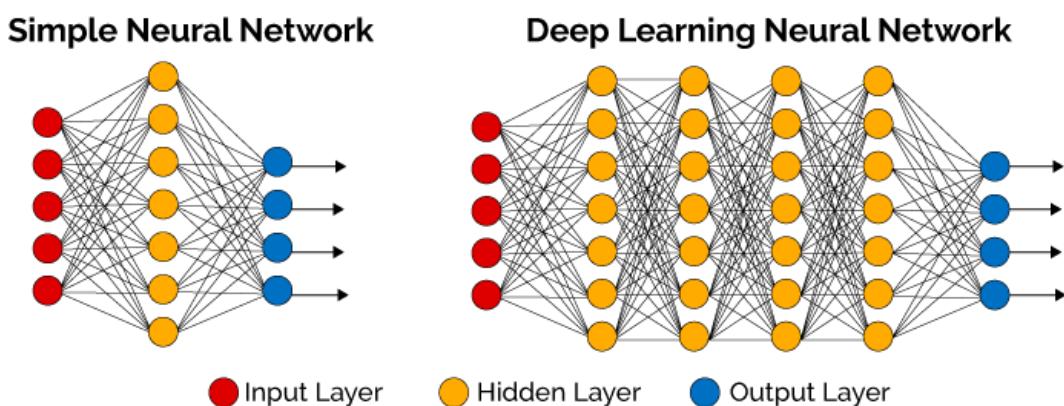
#### **2.2 Pengolahan Citra**

Pengolahan Citra adalah pemrosesan citra, khususnya dengan menggunakan komputer, menjadi citra yang kualitasnya lebih baik. Pengolahan citra bertujuan memperbaiki kualitas citra agar mudah diinterpretasi oleh manusia

atau mesin (dalam hal ini komputer). Teknik-teknik pengolahan citra mentransformasikan citra menjadi citra lain. Jadi, masukannya adalah citra dan keluarannya juga citra yang berkualitas lebih baik daripada citra masukan [7].

### 2.3 Deep Leraning

*Deep Learning* (Pembelajaran Dalam) atau sering dikenal dengan istilah Pembelajaran Struktural Mendalam (*Deep Structured Learning*) atau Pembelajaran Hierarki (*Hierarchical learning*) adalah salah satu cabang dari ilmu pembelajaran mesin (*Machine Learning*) yang terdiri algoritma pemodelan abstraksi tingkat tinggi pada data menggunakan sekumpulan fungsi transformasi non-linear yang ditata berlapis-lapis dan mendalam. Teknik dan algoritma dalam pembelajaran dalam dapat digunakan baik untuk kebutuhan pembelajaran terarah (*supervised learning*), pembelajaran tak terarah (*unsupervised learning*) dan semi-terarah (*semi-supervised learning*) dalam berbagai aplikasi seperti pengenalan citra, pengenalan suara, klasifikasi teks, dan sebagainya[8].



Gambar 2.1 Perbedaan antara lapisan layer pada Jaringan Saraf Tiruan dengan Jaringan Deep Learning

*Deep Learning* adalah salah satu jenis algoritma jaringan saraf tiruan yang menggunakan meta data sebagai *input* dan mengolahnya menggunakan sejumlah

lapisan tersembunyi (*hidden layer*) transformasi non linier dari data masukan untuk menghitung nilai *output*. Algoritma pada *Deep Learning* memiliki fitur yang unik yaitu sebuah fitur yang mampu mengekstraksi secara otomatis. Hal ini berarti algoritma yang dimilikinya secara otomatis dapat menangkap fitur yang relevan sebagai keperluan dalam pemecahan suatu masalah. Algoritma semacam ini sangat penting dalam sebuah kecerdasan buatan karena mampu mengurangi beban pemrograman dalam memilih fitur yang eksplisit.

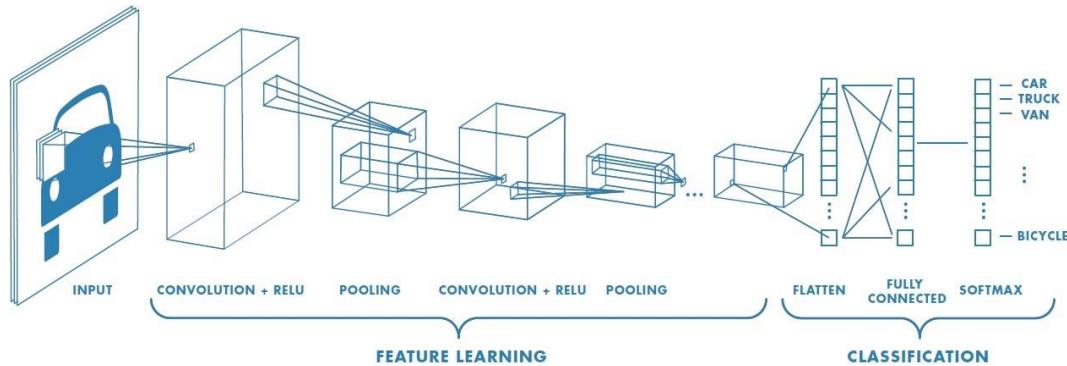
Dalam jaringan saraf tiruan tipe *Deep Learning* setiap lapisan tersembunyi bertanggung jawab untuk melatih serangkaian fitur unik berdasarkan *output* dari jaringan sebelumnya. Algoritma ini akan menjadi semakin kompleks dan bersifat abstrak ketika jumlah lapisan tersembunyi (*hidden layer*) semakin bertambah banyak. Jaringan saraf yang dimiliki oleh *Deep Learning* terbentuk dari hierarki sederhana dengan beberapa lapisan hingga tingkat tinggi atau banyak lapisan (*multi layer*). Berdasarkan hal itulah *Deep Learning* dapat digunakan untuk memecahkan masalah kompleks yang lebih rumit dan terdiri dari sejumlah besar lapisan transformasi non linier.

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

*Convolutional Neural Network* (CNN) adalah salah satu jenis neural network yang biasa digunakan pada data *image*. CNN bisa digunakan untuk mendekripsi dan mengenali *object* pada sebuah *image* [6].

Secara garis besar CNN tidak jauh beda dengan *neural network* biasanya.

CNN terdiri dari neuron yang memiliki *weight*, bias dan *activation function*.



Gambar 2.2 Alur Proses metode CNN

#### 2.4.1 Input Layer

*Input* layer menampung nilai *pixel* dari citra yang menjadi masukan [9].

Untuk citra dengan ukuran  $64 \times 64$  dengan 3 *channel* warna, RGB(Red, Green, Blue) maka yang menjadi masukan akan adalah *pixel array* yang berukuran  $64 \times 64 \times 3$ .

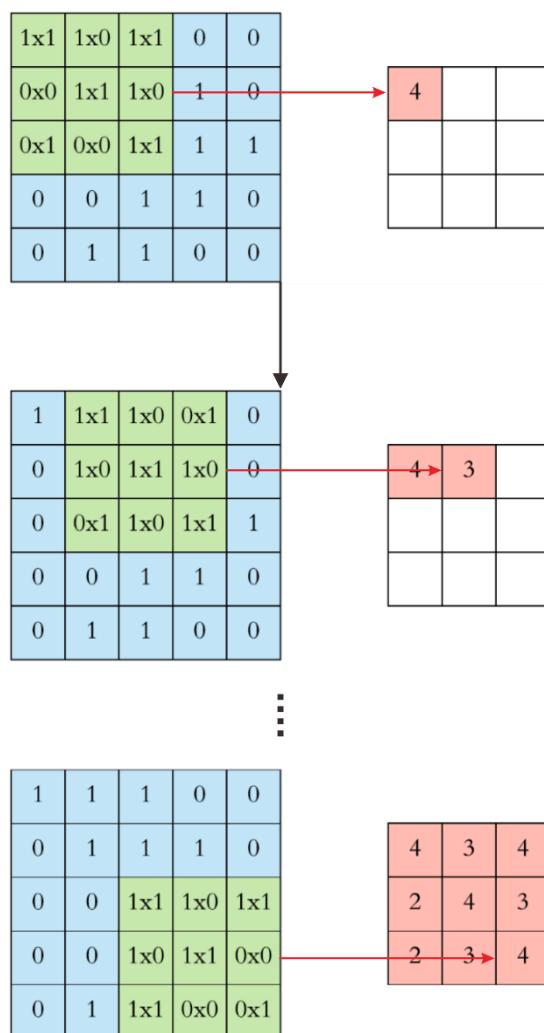


Gambar 2.3 Proses Input Layer

#### 2.4.2 Convolution Layer

*Convolution Layer* adalah inti dari CNN. *Convolution Layer* menghasilkan citra baru yang menunjukkan fitur dari citra *input* [9]. Dalam proses tersebut,

*Convolution Layer* menggunakan filter pada setiap citra yang menjadi masukan. Filter pada layer ini berupa *array* 2 dimensi bisa berukuran  $5\times 5$ ,  $3\times 3$  atau  $1\times 1$ . Proses *convolution* dengan menggunakan filter pada layer ini akan menghasilkan *feature map* yang akan digunakan pada *activation layer*. Gambar di bawah ini menunjukkan alur pada *Convolution Layer*.



Gambar 2.4 Alur Convolution Layer

#### 2.4.3 Activation Layer

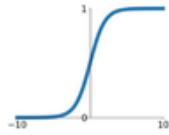
*Activation Layer* adalah layer di mana *feature map* dimasukkan ke dalam *activation Function*. *Activation function* digunakan untuk mengubah nilai-nilai

pada *feature* map pada *range* tertentu sesuai dengan *activation function* yang digunakan. ini bertujuan untuk meneruskan nilai yang menampilkan fitur dominan dari citra yang masuk ke layer berikutnya. *Activation function* yang umum digunakan bisa dilihat pada gambar berikut [9].

## Activation Functions

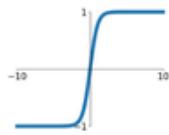
### Sigmoid

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



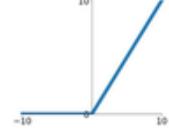
### tanh

$$\tanh(x)$$



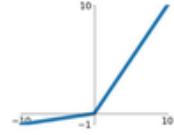
### ReLU

$$\max(0, x)$$



### Leaky ReLU

$$\max(0.1x, x)$$

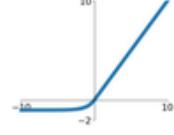


### Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

### ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Gambar 2.5 Activation functions

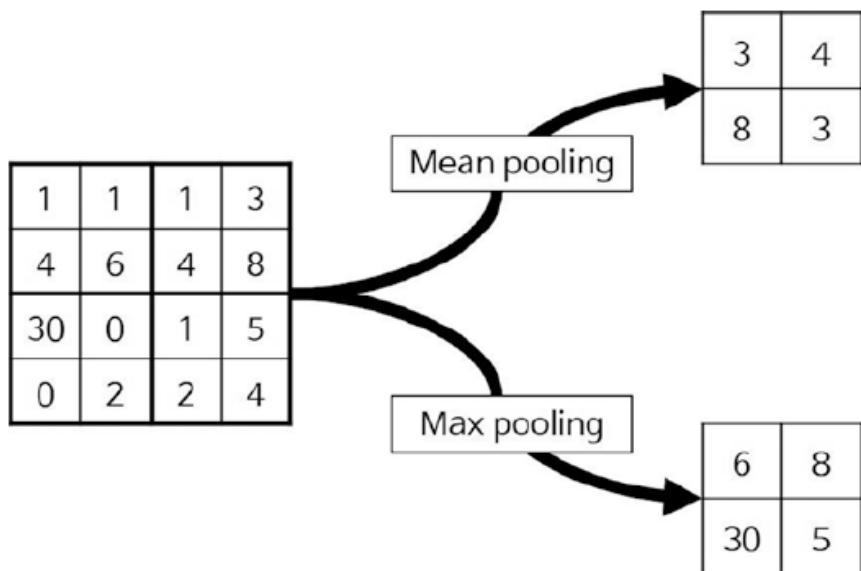
### 2.4.4 Pooling Layer

*Pooling layer* menerima *input* dari *activation layer* kemudian mengurangi jumlah parameternya. *Pooling* juga biasa disebut sub sampling atau *down sampling* yang mengurangi dimensi dari *feature* map tanpa menghilangkan informasi penting di dalamnya [9]. Proses dalam *pooling* layer cukup sederhana. pertama-tama kita menentukan ukuran *down sampling* yang akan digunakan pada *feature* map, misalnya  $2 \times 2$ . Setelah itu kita akan melakukan proses *pooling* pada *feature* map, sebagai contoh kita akan menggunakan *feature* map berukuran  $4 \times 4$  berikut.

1	1	1	3
4	6	4	8
30	0	1	5
0	2	2	4

Gambar 2.6 FeatMap

Setelah itu kita akan menggunakan matrix  $2 \times 2$  untuk melakukan proses *pooling*. proses *Pooling* sendiri ada beberapa macam seperti *Max pooling*, *Mean pooling*, dan *Sum pooling*.

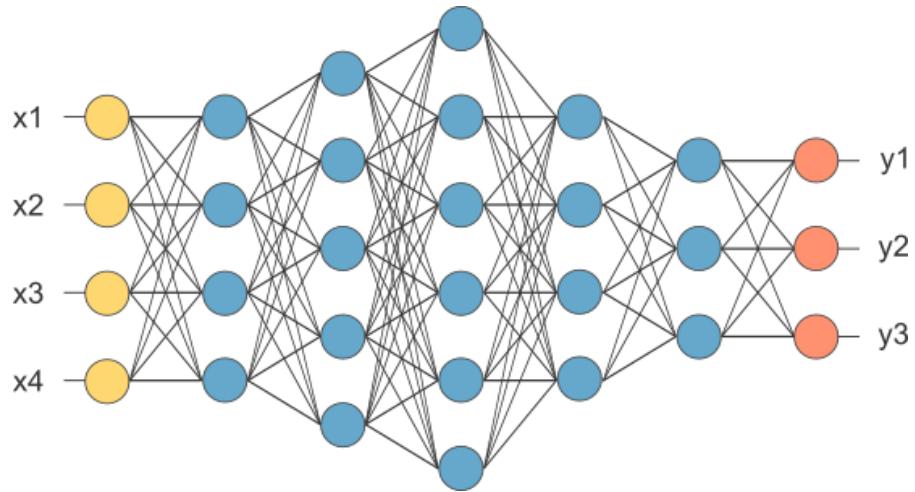


Gambar 2.7 Proses Pooling

#### 2.4.5 Fully Connected Layer

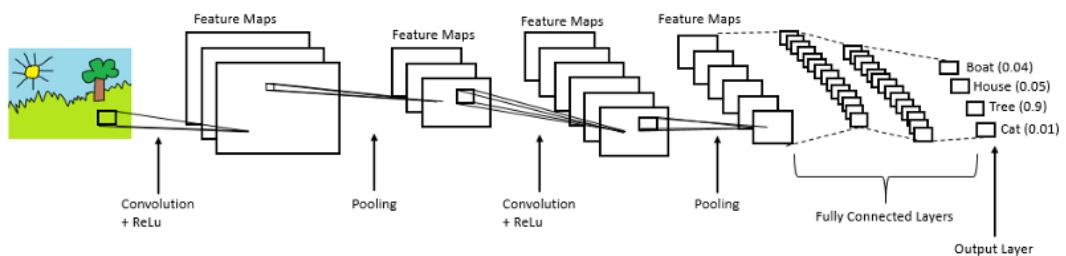
Setelah melewati proses-proses di atas, hasil dari *pooling* layer digunakan menjadi masukan untuk *Fully connected layer*. Layer ini memiliki kesamaan struktur dengan *Artificial Neural Network* (ANN) pada umumnya yaitu memiliki *input layer*, *hidden layer* dan *output layer* yang masing-masing memiliki *neuron*-

*neuron* yang saling terhubung dengan neuron-neuron di layer tetangganya. gambar di bawah ini merupakan contoh *Fully Connected Layer* [9].



Gambar 2.8 Fully Connected Layer

Pada gambar di atas dapat dilihat sebelum hasil *pooling* digunakan sebagai *input*, hasil *pooling* terlebih dahulu diubah menjadi vektor ( $x_1, x_2, x_3$ , dst) kemudian dari sini diproses ke dalam *Fully Connected Layer*. Pada layer terakhir di dalam *Fully Connected* layer akan digunakan *activation function sigmoid* atau *softmax* untuk menentukan klasifikasi dari citra masukan yang dari *Input Layer* CNN.

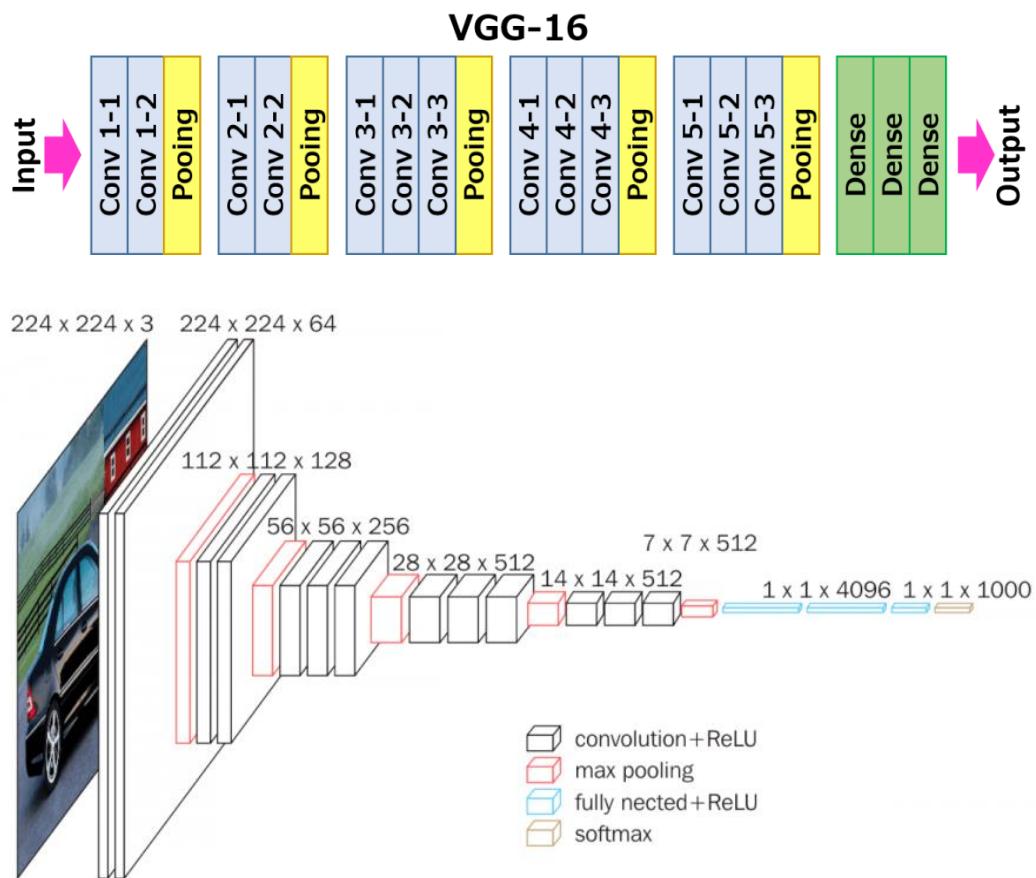


Gambar 2.9 Contoh CNN

## 2.5 Arsitektur CNN

### 2.5.1 Arsitektur VGG16

VGG16 adalah model jaringan saraf convolutional yang diusulkan oleh K. Simonyan dan A. Zisserman dari University of Oxford dalam makalah “Jaringan Konvolusional Sangat Dalam untuk Pengenalan Gambar Skala Besar” [10]. Model ini mencapai akurasi tes top-5 92,7% di ImageNet, yang merupakan dataset lebih dari 14 juta gambar dengan 1000 kelas. VGG16 adalah salah satu model terkenal yang diajukan ke ILSVRC-2014.



Gambar 2.10 Arsitektur VGG16

Masukan ke lapisan *conv1* berukuran tetap 224x224 gambar RGB. Gambar dilewatkan melalui tumpukan lapisan konvolusional (*conv.*), dimana filter digunakan dengan bidang reseptif yang sangat kecil sebesar  $3 \times 3$ . Dalam salah satu konfigurasi, ia juga menggunakan *filter* konvolusi  $1 \times 1$ , yang dapat dilihat sebagai transformasi linear dari saluran masukan (diikuti oleh non-linearitas). Langkah (*stride*) konvolusi ditetapkan pada 1 piksel, spasial *padding* dari masukan lapisan konvolusi sedemikian rupa sehingga resolusi spasial/ukuran tersebut dipertahankan setelah konvolusi, yaitu *padding* sebesar 1 piksel untuk  $3 \times 3$  lapisan konvolusi. Spasial *pooling* dilakukan oleh lima lapisan *max-pooling*, yang mengikuti beberapa lapisan konvolusi (tidak semua lapisan konv diikuti oleh *max-pooling*). *Max-pooling* dilakukan dengan ukuran piksel  $2 \times 2$ , dan 2 *stride*.

Tiga lapisan *Fully-Connected* (FC) mengikuti tumpukan lapisan konvolusional (yang memiliki kedalaman berbeda dalam arsitektur yang berbeda): dua yang pertama memiliki ukuran masing-masing 4096 kanal, yang ketiga melakukan klasifikasi ILSVRC 1000 keluaran dan dengan demikian memuat 1000 saluran (satu untuk setiap kelas). Lapisan terakhir adalah lapisan *soft-max*. Konfigurasi *fully connected layers* adalah sama di semua jaringan.

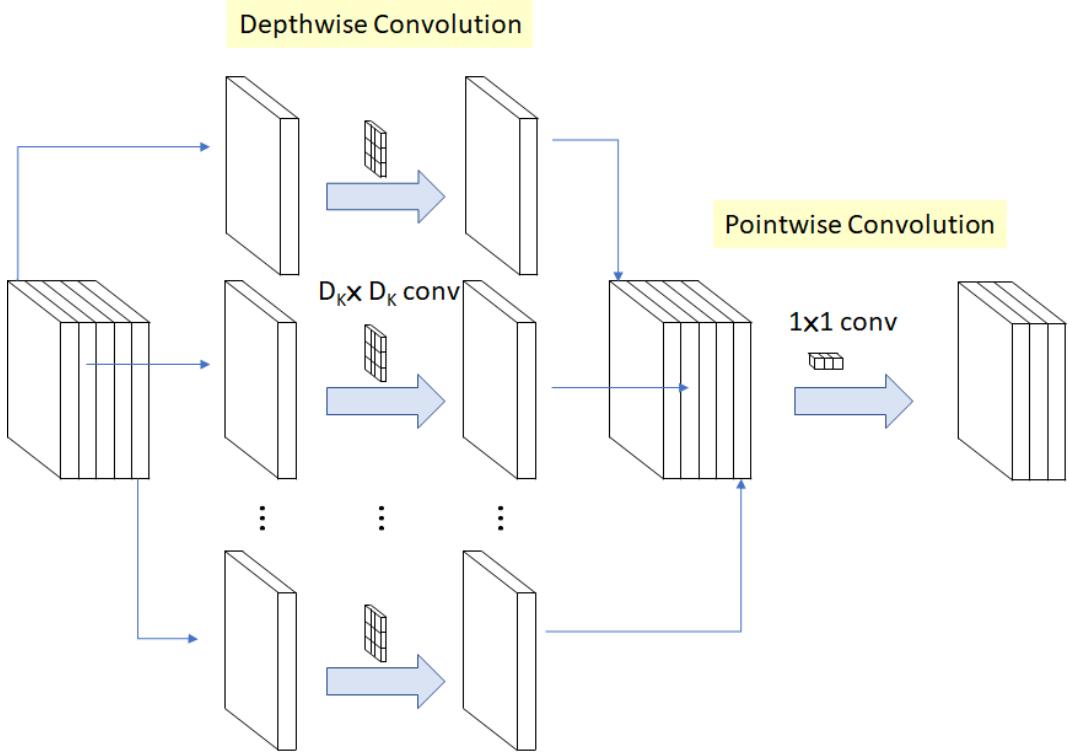
Setiap *hidden layers* dilengkapi dengan fungsi aktivasi ReLU. Juga dicatat bahwa tidak ada jaringan (kecuali satu) yang berisi Normalisasi Respon Lokal (LRN), normalisasi seperti itu tidak meningkatkan kinerja pada dataset ILSVRC, tetapi mengarah pada peningkatan konsumsi memori dan waktu komputasi.

## 2.5.2 Arsitektur *MobileNet*

*MobileNets*, merupakan salah satu arsitektur CNN yang dapat digunakan untuk mengatasi kebutuhan akan *computing resource* berlebih [11]. Seperti namanya, *Mobile*, para peneliti dari *Google* membuat arsitektur CNN yang dapat digunakan untuk ponsel. Perbedaan mendasar antara arsitektur *MobileNet* dan arsitektur CNN pada umumnya adalah penggunaan lapisan konvolusi dengan ketebalan *filter* yang sesuai dengan ketebalan dari gambar masukan. *MobileNet* membagi konvolusi menjadi *depthwise convolution* dan *pointwise convolution*.

### 1. *Depthwise Separable Convolution*

Model *MobileNet* didasarkan pada konvolusi mendalam yang dapat dipisahkan (*depthwise separable convolution*), yaitu bentuk konvolusi yang dibentuk dengan menguraikan konvolusi standar (*standard convolution*) menjadi konvolusi mendalam (*depthwise convolution*) dan konvolusi  $1 \times 1$  yang disebut konvolusi searah (*pointwise convolution*) [12]. Untuk *MobileNets*, *depthwise convolution* menerapkan satu *filter* ke setiap saluran masukkannya. *Pointwise convolution* kemudian menerapkan konvolusi  $1 \times 1$  untuk menggabungkan keluaran dari *depthwise convolution*. Konvolusi standar melakukan *filter* dan menggabungkan masukan ke dalam *set* keluaran baru dalam satu langkah. *Depthwise separable convolution* membaginya menjadi dua lapisan, lapisan terpisah untuk *filter* dan lapisan terpisah untuk menggabungkannya. Penguraian ini memiliki efek mengurangi komputasi dan ukuran model secara drastis.



Gambar 2.1 Depthwise separable convolution

## 2. Struktur Jaringan

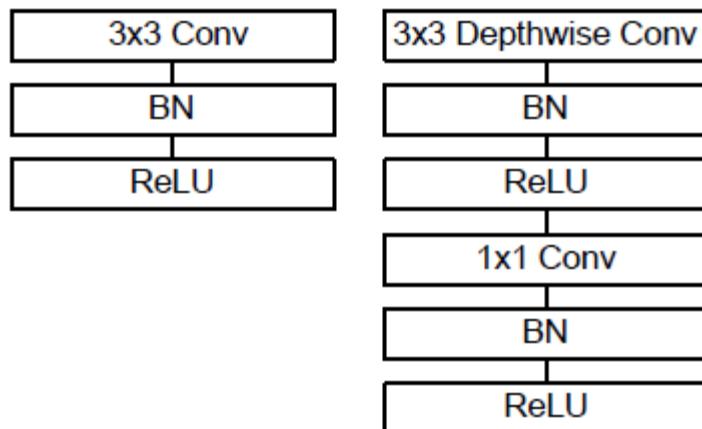
Arsitektur *MobileNet* didefinisikan dalam Tabel 2.1. Semua lapisan diikuti oleh *batchnorm* (*backnormalization*) dan ReLU nonlinier dengan pengecualian lapisan akhir yang terhubung penuh yang tidak memiliki nonlinier dan dimasukkan ke dalam lapisan *softmax* untuk klasifikasi.

Tabel 2.1 Arsitektur *MobileNet*

Type/Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 256$

Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 512$
5x Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
5x Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool $7 \times 7$	$7 \times 7 \times 1024$
FC / s1	$1024 \times 1000$	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Pada sebuah lapisan konvolusi standar hanya menggunakan lapisan konvolusi biasa sebesar 3x3. Sedangkan pada sebuah lapisan *depthwise separable convolution* dipisahkan menjadi dua konvolusi yaitu 3x3 *depthwise convolution* dan 1x1 *pointwise convolution* serta *batchnorm* dan ReLU nonlinier di setiap lapisan konvolusinya seperti pada gambar 2.13 berikut.



Gambar 2.2 Standard convolution (kiri), Depthwise separable convolution (kanan) dengan ReLU dan BN.

### 2.5.3 Arsitektur *Inception V3*

*Inception V3* adalah sebuah model *deep learning convolutional neural network* yang dikembangkan oleh *Google* memenuhi *ImageNet Large Visual*

*recognition challenge* pada tahun 2012. Model *Inception* menggunakan filter pada layer *convolutional*, tidak seperti *convolutional layer* biasa. Hasil dari beberapa filter tersebut dijadikan satu lagi menggunakan *channel Concat* sebelum masuk kedalam iterasi berikutnya [13].

Tujuan dari modul *inception* adalah untuk bertindak sebagai *multilevel feature extractor* dengan menghitung filter-filter *convolutional* dalam modul yang sama. Hasil dari filter-filter tersebut kemudian ditumpukan kedalam dimensi *channel* sebelum dimasukan kedalam lapisan selanjutnya.

## 2.6 Google Colaboratory

*Google Colaboratory* adalah salah satu produk Google berbasis *cloud* yang bisa kita gunakan secara gratis. Perbedaannya adalah Google Colab dibuat khusus untuk para *programmer* atau *researcher* yang mungkin kesulitan untuk mendapatkan akses komputer dengan spek tinggi. Google Colab adalah *coding environment* bahasa pemrograman *Python* dengan format “*notebook*” (mirip dengan *Jupyter notebook*), atau dengan kata lain Google seakan meminjami kita komputer secara gratis! untuk membuat program oleh Google [14].

Beberapa kelebihan dari Google Colab adalah sebagai berikut:

1. *Free GPU!* Google Colab memudahkan kita untuk menjalankan program pada komputer dengan spek tinggi (GPU Tesla, RAM ~12GB, Disk ~300GB yang masih bisa sambung dengan Google *Drive*, akses internet cepat untuk *download file* besar) dan *running* dalam waktu yang lama (Google *Colab* mengizinkan kita untuk menjalankan program hingga 12 jam).

2. Mudah berintegrasi! Google *Colab* terbilang sangat fleksibel dalam hal integrasi. Kita dapat dengan mudah menghubungkan Google *Colab* dengan *jupyter notebook* di komputer kita (*local runtime*), menghubungkan dengan Google *Drive*, atau dengan *Github*.
3. *Colaborate!* Google Colab juga memudahkan kita berkolaborasi dengan orang lain dengan cara membagi *kodingan* secara *online*.
4. Fleksibel! Salah satu yang saya favoritkan adalah kita bisa dengan mudah menjalankan *deep learning* program via HP! ya karena pada esensinya Google *Colab* hanya perlu *running* di browser, kita bisa mengawasi proses *training* (atau bahkan *coding*) via browser *smartphone* kita selama *smartphone* kita terhubung dengan Google Drive yang sama.

## 2.7 Python

*Python* adalah bahasa pemrograman interpretatif multiguna yang memakai filosofi perancangan dengan fokus kepada tingkat keterbacaan kode. Sebagai bahasa pemrograman, *Python* menggabungkan kemampuan, kapabilitas dan sintaksis kode serta fungsi pustaka yang berkualitas tinggi [15].

Banyak sekali fitur yang dimiliki *Python* sehingga menarik digunakan. Berikut fitur-fitur *Python* yang menjadi keunggulan darinya:

1. Berorientasi kepada objek.
2. Mudah dikembangkan dengan menciptakan modul-modul baru. Modul tersebut juga bisa dibangun dengan bahasa *Python*.
3. Memiliki tata bahasa yang mudah dipelajari.

4. Didukung sistem pengelolaan memori secara otomatis sehingga membutuhkan kinerja saat *coding*.
5. *Python* juga memiliki banyak fasilitas pendukung sehingga ketika mengoperasikannya, terhitung mudah dan cepat.

## 2.8 Framework Keras

Saat ini kita mengenal cukup banyak *framework* untuk penggunaan *deep learning*. Beberapa yang cukup populer di antaranya adalah: *Tensorflow*, Keras, *MXNet*, dan *PyTorch*.

Pada analisis ini, saya hanya akan menggunakan *framework* Keras dengan bantuan R *interface* dari RStudio. Keras adalah high-level neural *network* API yang dikembangkan dengan *Python* dengan fokus tujuan untuk mempercepat proses riset atau percobaan [16]. Beberapa fitur utama dari Keras:

1. Mampu menjalankan *source code* yang sama menggunakan CPU atau GPU dengan lancar.
2. API yang *user-friendly* sehingga mempermudah penggunanya dalam proses prototipe model *deep learning*.
3. Dukungan *built-in* untuk CNN atau *Convolutional Neural Networks* (*Computer Vision*), RNN atau *Recurrent Neural Networks* (untuk *sequence processing*), dan kombinasi keduanya.
4. Dapat digunakan untuk hampir semua jenis dari model *deep learning*.

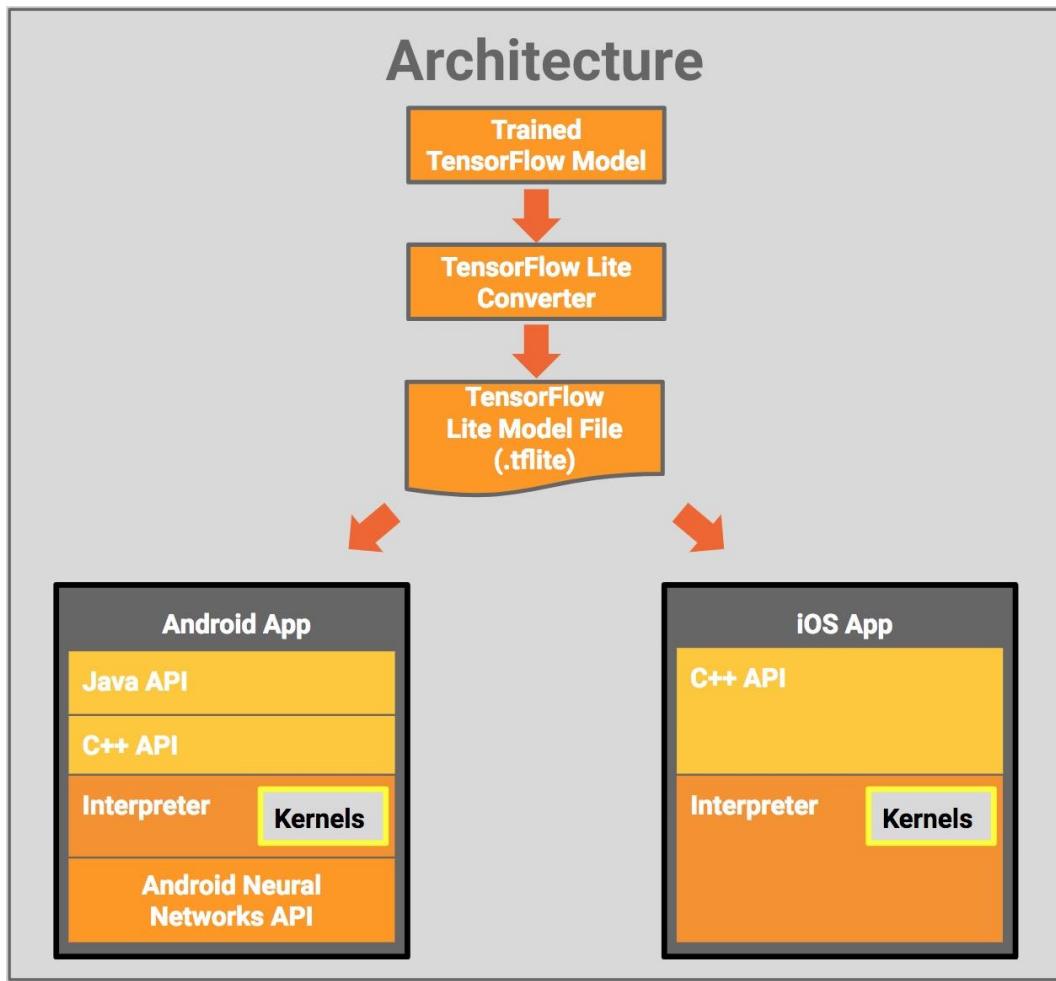
## 2.9 Tensor Flow Lite

*TensorFlow Lite* adalah solusi mudah untuk perangkat seluler dan *embedded devices*. Ini memungkinkan untuk menjalankan model yang dipelajari

mesin pada perangkat seluler dengan latensi rendah, sehingga dapat memanfaatkannya untuk melakukan klasifikasi, regresi, atau apa pun yang mungkin Anda inginkan tanpa harus melakukan perjalanan bolak-balik ke server [17].

TensorFlow Lite terdiri dari runtime tempat Anda dapat menjalankan model yang sudah ada sebelumnya, dan seperangkat alat yang dapat Anda gunakan untuk mempersiapkan model Anda untuk digunakan pada perangkat seluler dan *embedded devices*.

TensorFlow Lite saat ini dalam pratinjau pengembang, sehingga mungkin tidak mendukung semua operasi di semua model TensorFlow. Meskipun demikian, ini berfungsi dengan model Klasifikasi Gambar umum.



Gambar 2.11 Arsitektur Konversi model ke *TensorFlow Lite*

## 2.10 Android

Android Studio adalah Lingkungan Pengembangan Terpadu (*Integrated Development Environment/IDE*) resmi untuk pengembangan aplikasi Android, yang didasarkan pada IntelliJ IDEA [18]. Selain sebagai editor kode dan fitur developer IntelliJ yang andal, Android Studio menawarkan banyak fitur yang meningkatkan produktivitas dalam membuat aplikasi Android, seperti:

1. Sistem build berbasis Gradle yang fleksibel.
2. Emulator yang cepat dan kaya fitur.

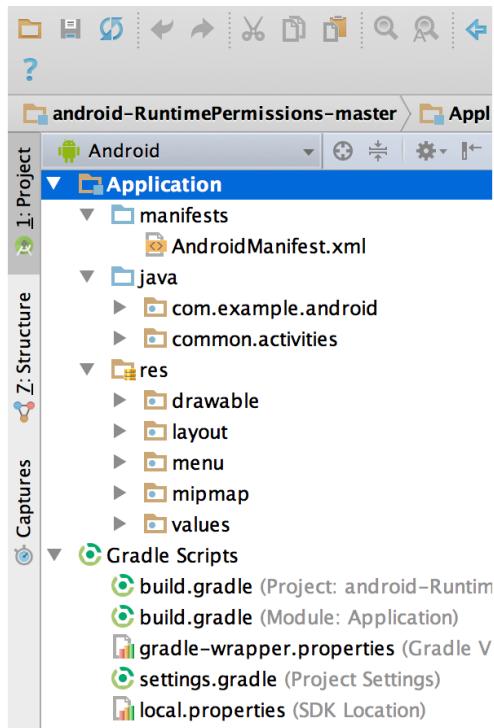
3. Lingkungan terpadu tempat Anda bisa mengembangkan aplikasi untuk semua perangkat Android.
4. Terapkan Perubahan untuk melakukan *push* pada perubahan kode dan *resource* ke aplikasi yang sedang berjalan tanpa memulai ulang aplikasi.
5. *Template* kode dan integrasi *GitHub* untuk membantu Anda membuat fitur aplikasi umum dan mengimpor kode sampel.
6. *Framework* dan fitur pengujian yang lengkap.
7. Fitur *lint* untuk merekam performa, kegunaan, kompatibilitas versi, dan masalah lainnya.
8. Dukungan C++ dan NDK.
9. Dukungan bawaan untuk Google Cloud Platform, yang memudahkan integrasi Google Cloud Messaging dan App Engine.

### **2.10.1 Struktur *Project***

Setiap project di Android Studio berisi satu atau beberapa modul dengan file kode sumber dan file resource. Jenis modul meliputi:

1. Modul aplikasi Android
2. Modul library
3. Modul Google App Engine

Secara default, Android Studio menampilkan file project Anda dalam tampilan project Android, seperti yang ditunjukkan pada Gambar 2.12. Tampilan ini disusun menurut modul untuk memberikan akses cepat ke file sumber utama project Anda.



Gambar 2.12 File project dalam tampilan Android

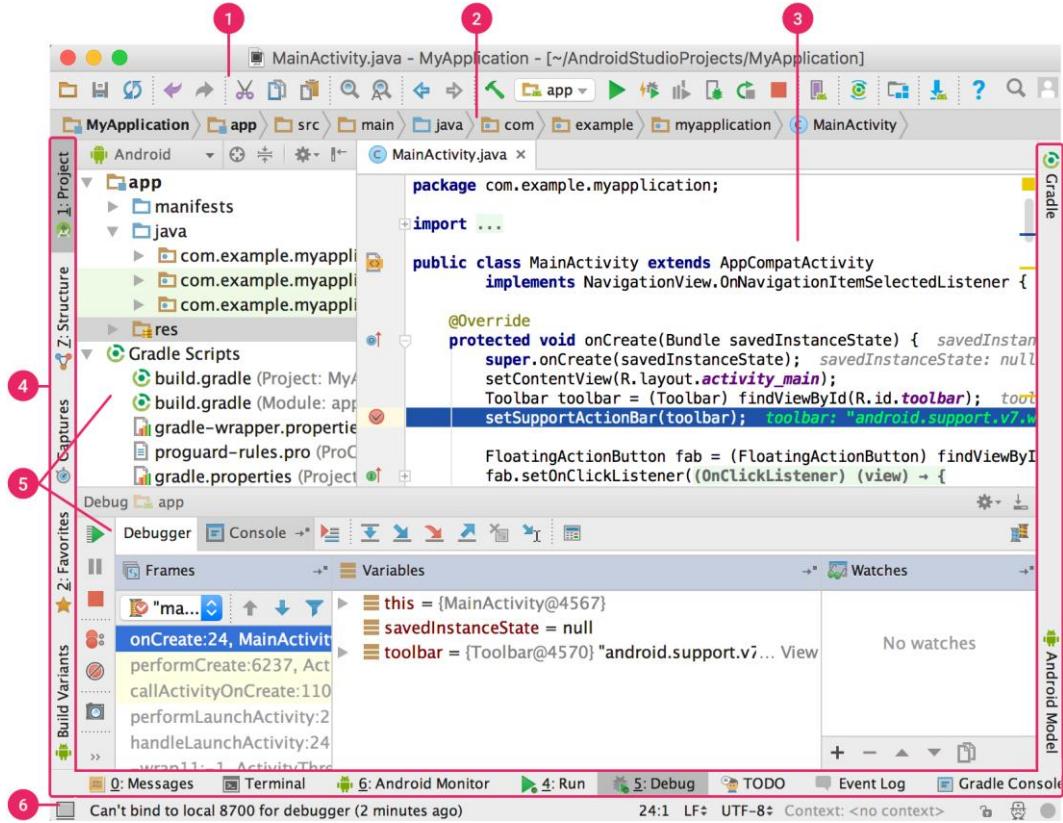
Semua file build terlihat di tingkat teratas di bagian *Gradle Script* dan

setiap modul aplikasi berisi folder berikut:

1. *Manifests* : Berisi file *AndroidManifest.xml*.
2. *Java* : Berisi file kode sumber Java, termasuk kode pengujian *JUnit*.
3. *Res* : Berisi semua *resource* non-kode, seperti tata letak XML, *string* UI, dan gambar *bitmap*.

### 2.10.2 Tampilan Antarmuka Pengguna

Jendela utama Android Studio terdiri dari beberapa area logis yang diidentifikasi dalam Gambar 2.13.



Gambar 2.13 Tampilan Antarmuka Pengguna

1. **Toolbar** memungkinkan Anda melakukan berbagai tindakan, termasuk menjalankan aplikasi dan meluncurkan fitur Android.
2. **Menu navigasi** membantu Anda menjelajah *project* dan membuka *file* untuk diedit. Menu ini memberikan tampilan struktur yang lebih ringkas yang terlihat di jendela Project.
3. **Jendela editor** adalah tempat Anda membuat dan memodifikasi kode. Tergantung jenis *file* yang ada, editor ini dapat berubah. Misalnya, saat menampilkan *file* tata letak, editor akan menampilkan *Layout Editor*.
4. **Panel jendela fitur** berada di sisi luar jendela IDE dan berisi tombol-tombol yang memungkinkan Anda memperluas atau mencuitkan setiap jendela fitur.

5. **Jendela fitur** memberi Anda akses ke tugas tertentu seperti pengelolaan *project*, penelusuran, kontrol versi, dan banyak lagi. Anda dapat memperluas dan menciutkan jendela ini.
6. **Status bar** menampilkan status *project* Anda dan IDE itu sendiri, serta semua peringatan atau pesan.

## 2.11 Penyakit Pada Tumbuhan

Penyakit tanaman dapat diartikan gangguan terhadap tanaman yang disebabkan oleh *pathogen* dan non *pathogen* yang menyebabkan terganggunya proses pertumbuhan pada bagian-bagian tertentu dari tanaman yang tidak dapat berjalan sesuai fungsinya dengan normal dan dengan baik sehingga menghambat pertumbuhan pada tanaman [19].

Penyebaran penyakit *phatogen* (biotik) dapat melalui jamur, bakteri, *riketsia*, *Mikloplasma*, *spiroplasma* dan hama yang membawa virus. Penyebaran penyakit non *pathogen* (abiotik) melalui bahan kimia, faktor alam ataupun kondisi tanaman itu sendiri.

### 2.11.1 Penyakit Tanaman Tomat

Tanaman tomat sangat rentan terhadap serangan penyakit. Jika lingkungan sekitar tanaman tomat tidak bersih maka patogen seperti jamur, bakteri atau virus akan sangat mudah untuk menyerang tanaman tomat [20].

Berikut ini adalah berapa jenis penyakit yang umumnya menyerang tanaman tomat.

### ***Bacterial Spot***

Penyakit ini disebabkan oleh bakteri *Xanthomonas vesicatoria* , yang menyerang tomat hijau tetapi tidak merah. Penyakit ini lebih banyak terjadi pada musim hujan. Kerusakan tanaman termasuk bercak daun dan buah, yang mengakibatkan berkurangnya hasil, defoliasi, dan buah yang terbakar matahari. Gejala-gejalanya terdiri dari banyak bintik-bintik kecil, bersudut hingga tidak beraturan, basah kuyup pada daun dan sedikit naik ke bintik-bintik kudis pada buah. Bintik-bintik daun mungkin memiliki lingkaran cahaya kuning. Pusat-pusat mengering dan sering sobek [21].



*Gambar 2.14 Bacterial Spot*

### ***Early Blight (Bercak Daun)***

Pada daun terdapat bercak kecil berwarna cokelat yang dapat meluas dan menyebabkan daun berlubang. Bercak dicirikan dengan adanya lingkaran yang konsentris. Penyebab bercak daun ini adalah karena serangan jamur *Alternaria solani* [20]. Jamur ini biasanya menyerang di saat musim hujan.



Gambar 2.15 Early Blight

#### **Late Blight (Busuk daun)**

Pada daun muncul bercak cokelat dengan tepi berwarna kuning. Di bagian bawah bercak akan terlihat adanya lapisan putih seperti beludru. Bercak ini kemudian meluas ke seluruh daun.

Penyakit ini disebabkan oleh jamur *Phytophthora infestans*. Penyebarannya sangat aktif di musim hujan, terutama pada suhu lingkungan 18-20 °C dan kelembapan 91-100%.



Gambar 2.16 Late Blight

### ***Leaf Mold (Jamur Daun)***

Jamur *Passalora fulva* menyebabkan jamur daun. Ini pertama kali diamati pada daun yang lebih tua di dekat tanah di mana gerakan udara buruk dan kelembaban tinggi [21]. Gejala awalnya adalah bintik-bintik hijau pucat atau kekuningan pada permukaan daun atas, yang membesar dan berubah menjadi kuning khas.

Jamur bertahan hidup di sisa tanaman dan di tanah. Spora disebarluaskan oleh hujan, angin atau alat.



*Gambar 2.17 Leaf Mold*

### ***Mosaic Virus***

Gejala infeksi virus dapat muncul sebagai bercak daun hijau terang dan gelap. *Tobacco mosaic virus* (TMV) menyebabkan bercak daun yang lebih tua dan dapat menyebabkan malformasi selebaran, yang dapat berbentuk seperti tali sepatu.

Virus ini sangat menular dan mudah ditularkan dengan cara memasukkan sedikit getah dari tanaman yang terinfeksi ke tanaman sehat.



Gambar 2.18 Mosaic Virus

#### ***Septoria Leaf Spot (Bintik Daun Septoria)***

Penyakit merusak daun tomat, daun dan batang (buah tidak terinfeksi) ini disebabkan oleh jamur *Septoria lycopersici* [21]. Infeksi biasanya terjadi pada daun bagian bawah dekat tanah, setelah tanaman mulai berbuah. Banyak bintik-bintik lingkar kecil dengan tepi gelap di sekitar pusat berwarna krem muncul di daun yang lebih tua. Bintik-bintik hitam kecil, yang merupakan tubuh penghasil spora, dapat dilihat di tengah-tengah bintik. Daun yang terlihat berbintik menguning, mati dan jatuh dari tanaman. Jamur ini paling aktif ketika suhu berkisar dari 68 hingga 77 ° F, kelembaban tinggi, dan curah hujan atau irigasi di kepala membasahi tanaman. Jamur tidak ditularkan melalui tanah, tetapi dapat menahan musim dingin pada residu tanaman dari tanaman sebelumnya, pembusukan vegetasi dan beberapa inang liar yang terkait dengan tomat.



Gambar 2.19 Septoria Leaf Spot

### **Two Spotted Spider Mite**

*Two Spotted Spider Mite* adalah spesies tungau paling umum yang menyerang tanaman sayuran dan buah-buahan di New England [23]. Tungau sering membuat permukaan daun atas tampak berbintik-bintik atau berbintik-bintik. Daun kemudian menguning dan jatuh. Populasi besar menghasilkan anyaman yang terlihat dapat menutupi daun. Biasanya penyakit ini muncul ketika cuaca kering yang panas.



Gambar 2.20 Two Spotted Spider Mite

### ***Target Spot (Bintik target mentimun)***

Banyak bintik bulat berwarna krem, hingga diameter 4 mm; sering kali, berbentuk tidak teratur atau bersudut, dibatasi oleh pembuluh darah. Daun mengering dan rontok sebelum waktunya [22].

Spora disebarluaskan oleh hujan yang ditiup angin, dan jika cuaca basah berangin berlanjut selama beberapa hari, penyebarannya cepat dan tanaman kehilangan daunnya dengan cepat.

Sumber jamur berasal dari tanaman lain yang terinfeksi, sisa-sisa tanaman sebelumnya dan, mungkin, spesies inang lainnya. Jamur sangat umum pada daun pepaya menyebabkan bintik-bintik sudut, coklat muda atau abu-abu, diameter 2 mm, kadang-kadang dikelilingi oleh margin kuning; pusat-pusat tempat sering rontok menghasilkan efek "lubang tembakan".



*Gambar 2.21 Target Spot*

### **Yellow Leaf Curl Virus**

Daun muda yang terinfeksi virus ini akan terlihat mengerut atau keriting dan berwarna kuning. Vektor utama virus ini adalah kutu kebul [20].

Pengendalian dapat dilakukan dengan cara mencabut tanaman terserang dan membakarnya. Pengendalian selanjutnya adalah dengan mengendalikan kutu kebul sebagai vektornya. Kutu kebul dapat diatasi dengan penyemprotan insektisida berbahan aktif *abamectin*.



Gambar 2.22 Yellow Leaf Curl Virus

### 2.11.2 Penyakit Tanaman Pada Daun Singkong

Gangguan hama, penyakit , dan gulma merupakan masalah yang dihadapi petani dalam budidaya ubi kayu karena selain dapat menurunkan hasil, juga dapat mengakibatkan penurunan kualitas ubi [24]. Berbagai tumbuhan pengganggu (gulma) baik berupa rerumputan ataupun tumbuhan berdaun sempit dan berdaun lebar sering tumbuh dan bersaing dengan tumbuhan ubi kayu dalam mendapatkan unsur hara, cahaya, maupun ruang tumbuh sehingga mengakibatkan penurunan hasil. Berikut beberapa penyakit pada daun tanaman ubi :

#### ***Cassava Bacterial Blight (CBB)***

CBB disebabkan oleh *Xanthomonas campestris* pv. *Manihotis* merupakan penyakit bakteri terpenting pada ubi kayu [24]. Gejala awal berupa lesi berwarna abu-abu mirip bekas tersiram air panas. Lesi dibatasi oleh tulang-tulang daun sehingga terbentuk lesi menyudut, terlihat lebih jelas pada sisi bawah daun.



Gambar 2.23 Cassava Bacterial Blight

#### **Cassava Brown Streak Disease (CBSD)**

CBSD yang diketahui disebabkan oleh *Cassava Brown Streak Virus* (CBSV). Gejala infeksi virus bergaris coklat ubi kayu sangat bervariasi antar varietas dan antar musim yang berbeda. Gejala infeksi CBSV dapat diamati pada daun, batang, buah, dan umbi [24]. Pada varietas yang peka, gejala dapat diamati pada semua bagian tanaman tersebut, sementara pada varietas yang toleran umumnya gejala hanya dapat diamati pada salah satu organ tanaman umumnya daun.



Gambar 2.24 Cassava Brown Streak Disease

### ***Cassava Green Mottle (CGM)***

Gejala tanaman ubi kayu yang terinfeksi Virus belang hijau ubi kayu adalah pada daun yang muncul menunjukkan gejala belang sistemis dengan nekrosis, namun pada daun-daun berikutnya tidak menunjukkan gejala meskipun mengandung virus [24]. Gejala yang paling jelas pada tanaman ubi kayu terdapat pada daun yang paling muda yang mengeriting, tepi daun mengalami distorsi dan menunjukkan pola belang hijau dan kuning. Umumnya tanaman seperti sembuh kembali, namun agak kerdil atau seperti tanaman sehat.



*Gambar 2.25 Cassava Green Mottle*

### ***Cassava Mosaic Disease (CMD)***

Penyakit ini disebabkan oleh spesies virus dalam genus *Begomovirus*. Tingkat paling parah pada *cassava Mosaic Disease* dipengaruhi oleh faktor lingkungan seperti intensitas cahaya, angin, curah hujan, kepadatan tanaman dan suhu. Mengingat bahwa virus ditularkan oleh kutu kebul, penyebaran virus akan sangat tergantung pada vektor [24].



*Gambar 2.26 Cassava Mosaic Disease*

## **BAB 3**

### **METODOLOGI PENELITIAN**

Pada bab ini akan dibahas metode yang digunakan untuk melakukan penelitian tugas akhir ini adalah sebagai berikut.

#### **3.1 Waktu dan Tempat Penelitian**

Penelitian dilaksanakan dimulai pada bulan Oktober 2019 sampai bulan Januari 2020. Bertempat di lingkungan kampus Fakultas Teknik Universitas Jenderal Soedirman, Kabupaten Purbalingga.

#### **3.2 Alat dan Bahan**

Dalam penelitian tugas akhir ini digunakan alat-alat dan bahan-bahan sebagai berikut:

##### **3.2.1 Perangkat keras**

Perangkat keras yang digunakan adalah sebagai berikut :

1. Laptop Lenovo B41-35 dengan spesifikasi *processor* AMD A8 7410 RAM 4 GB.
2. *Smartphone* OPPO A37fw dengan spesifikasi prosesor Snapdragon 410 dan RAM 2 GB.

##### **3.2.2 Perangkat Lunak**

Perangkat lunak yang digunakan untuk melakukan penelitian ini adalah :

1. Sistem Operasi *Windows* 10 64 bit.
2. Web *browser* *Google Chrome* versi 78.0.3904.96.
3. *Google Colaboratory*.
4. *Android Studio* versi 3.5.0.0

5. Sistem Operasi *Android Lollipop* 5.1.
6. Layanan Repositori *Web Development* pada *Platform Github*.

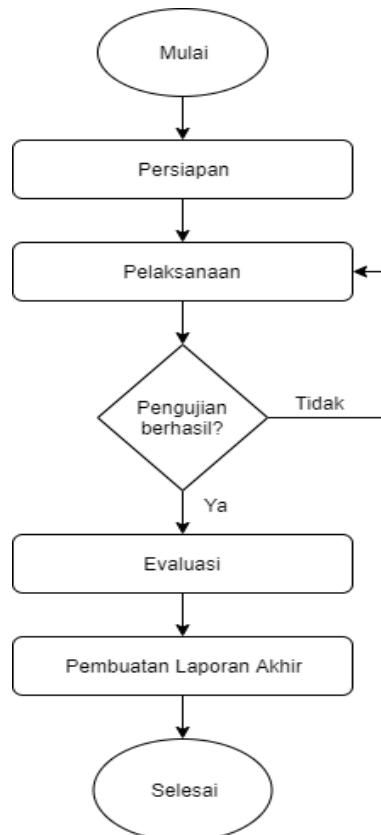
### 3.2.3 Dataset

Pada penelitian ini digunakan data set untuk melakukan *training* pada proses *Deep Learning* diperoleh dari kumpulan data set yang ada di **Github** dalam bentuk gambar \*.jpg.

1. Data set daun Tomat : <https://github.com/spMohanty/PlantVillage-Dataset>
2. Data set daun Singkong : <https://github.com/icassava/fgvcx-icassava>

### 3.3 Metode Penelitian

Penelitian ini dilaksanakan melalui beberapa tahapan sebagai berikut:



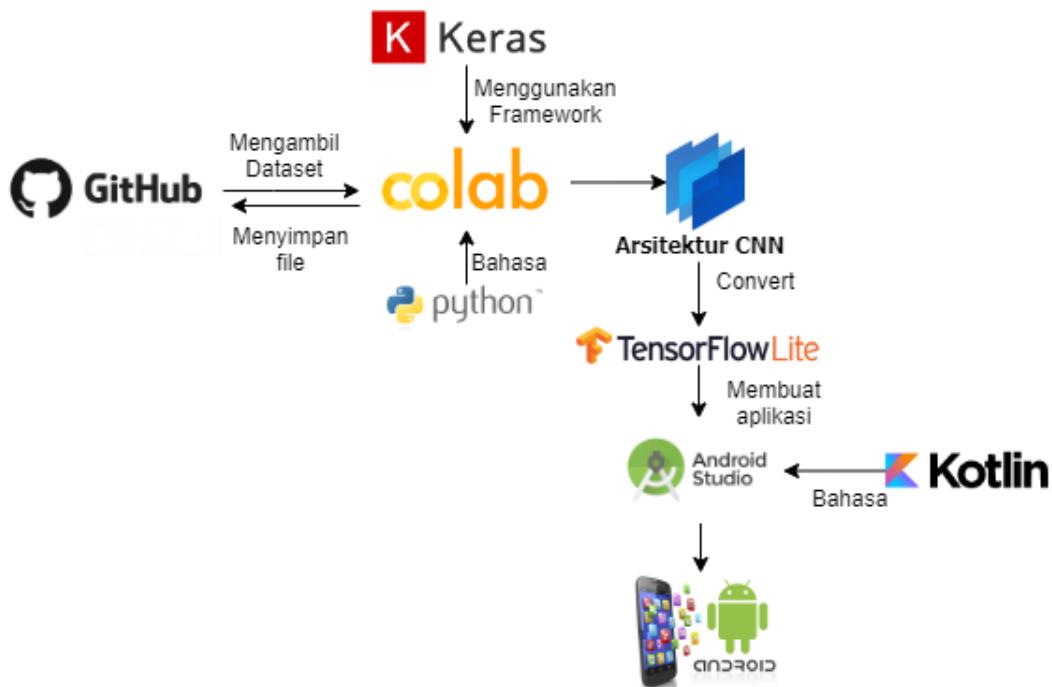
*Gambar 3.1 Tahapan Penelitian*

### 3.3.1 Persiapan

Pada tahap ini penulis melakukan studi pustaka melalui *website*, jurnal, buku, dan melakukan bimbingan langsung dengan dosen pembimbing. Penulis mencari materi-materi yang berkaitan dengan *deep learning*, *Convolutional Neural Network* (CNN), dan pengaplikasian dalam *smartphone*.

### 3.3.2 Pelaksanaan

Pada tahap ini penulis melakukan persiapan dan *preprocess*, mendesain arsitektur yang dipakai, dan melakukan pengujian.



Gambar 3.2 Tahap Pelaksanaan

#### Persiapan dan *Preprocess*

Pada tahap ini penulis melakukan pengumpulan data set dan menyiapkan beberapa penelitian sebelumnya sebagai panduan untuk melakukan penelitian ini. Dalam hal ini penulis memperoleh data set yang telah disediakan oleh Github.

## **Desain Arsitektur**

Pada tahap ini penulis membuat model *deep Learning* menggunakan *framework* keras dengan Google *Colaboratory* sebagai infrastrukturnya. Diawali dengan pengambilan data set pada Github, kemudian membuat model CNN (*Convolutional Neural Network*) dengan arsitektur yang berbeda-beda : VGG16, MobileNet V1, dan ResNet.

Setelah model terbentuk data set dilatih agar dapat mengidentifikasi penyakit tanaman pada daun tomat dan daun singkong mendapat data hasil pelatihan berupa *file* h5. Kemudian data pelatihan di *convert* menjadi *TensorFlow Lite* agar memudahkan dalam pemodelan aplikasi di android. Setelah di *convert*, lalu dilakukan pemodelan aplikasi menggunakan Android studio.

## **Tahap Pengujian**

Tahap ini penulis melakukan pengujian keseluruhan terhadap sistem yang telah dirancang. Sistem akan diuji apakah sudah sesuai yang diharapkan atau belum. Sistem dianggap sudah baik jika nilai dari kesalahan atau *error (loss)* semakin mendekati 0. Jika tidak sesuai maka peneliti melakukan pembenahan dan pengkajian ulang terhadap berbagai hal yang timbul selama pengujian.

### **3.3.3 Evaluasi**

Pada tahap ini penulis melakukan evaluasi mengenai hasil dari penelitian. Dari tiga arsitektur yang dipakai akan memperoleh hasil dan tingkat keakuratan yang berbeda. Karena setiap arsitektur memiliki kelebihan dan kekurangan masing-masing.

### **3.3.4 Tahap Akhir**

Tahap ini adalah tahapan terakhir yang di mana semua hasil penelitian sudah diperoleh kemudian penulisan laporan penelitian tersebut.

### **3.4 Alur Penelitian**

Penelitian ini dilakukan melalui beberapa tahapan yang dimulai dari tahap persiapan, dilanjutkan dengan tahap pengumpulan data set pelatihan dan data set pengujian. Setelah itu dilakukan tahap perancangan sistem yang dibuat. Kemudian dilakukan tahap pengujian dan evaluasi sistem. Pada tahap terakhir yaitu tahapan pembuatan laporan penelitian.

### **3.5 Waktu dan Jadwal Penelitian**

Penelitian dilaksanakan dalam waktu 4 bulan dimulai dari bulan Oktober 2019 sampai dengan bulan Januari 2020 dengan rencana jadwal kegiatan sebagai berikut.

*Tabel 3.1 Jadwal Penelitian*

No.	Kegiatan	Bulan 1				Bulan 2				Bulan 3				Bulan 4			
		I	II	III	IV												
1.	Persiapan dan Studi Pustaka																
2.	Pelaksanaan																
3.	Evaluasi																
3.	Pembuatan laporan																

## **BAB 4**

### **HASIL DAN PEMBAHASAN**

#### **4.1 Pengumpulan *Dataset***

*Dataset* yang digunakan pada identifikasi penyakit tanaman pada daun tomat dan daun singkong terdapat ada tiga bagian, yaitu *dataset* pelatihan, *dataset* validasi, dan *dataset* pengujian. *Dataset* pelatihan dan validasi merupakan *dataset* yang akan digunakan untuk melakukan pelatihan untuk memperoleh model dari kedua arsitektur tersebut. Sedangkan *dataset* pengujian digunakan untuk menguji tingkat probabilitas hasil yang benar pada kedua arsitektur tersebut pada *Google Colaboratory*. Seluruh dataset tersebut menggunakan gambar berwarna RGB (tiga saluran warna).

##### **4.1.1 *Dataset* Daun Tomat**

Dataset yang digunakan untuk melakukan proses *training* pendekripsi penyakit pada daun tomat diperoleh dari <https://github.com/spMohanty/PlantVillage-Dataset> milik spMohanty. Adapun jumlah *dataset* yang digunakan dapat dilihat pada tabel 4.1 berikut ini.

*Tabel 4.1 Dataset Daun Tomat*

No	Nama Penyakit	Jumlah
1	Tomat <i>Bacterial spot</i>	1.702
2	Tomat <i>Early blight</i>	1.920
3	Tomat <i>healthy</i>	1.926
4	Tomat <i>Late blight</i>	1.851
5	Tomat <i>Leaf Mold</i>	1.882
6	Tomat <i>Septoria leaf spot</i>	1.745
7	Tomat <i>Spider mites Two spotted spider mite</i>	1.741
8	Tomat <i>Target_Spot</i>	1.827
9	Tomat <i>Mosaic virus</i>	1.790
10	Tomat <i>Yellow Leaf Curl Virus</i>	1.961
<b>Jumlah</b>		18.345

#### **4.1.2 Dataset Daun Singkong**

Dataset yang digunakan untuk melakukan proses *training* pendekripsi penyakit pada daun singkong diperoleh dari <https://github.com/icassava/fgvx-icassava>. Adpun jumlah dataset pada masing-masing penyakit dapat dilihat pada tabel 4.2 berikut ini.

*Tabel 4.2Dataset Daun Singkong*

No	Nama Penyakit	Jumlah
1	<i>Cassava Bacteial Blight</i>	466
2	<i>Cassava Brown Streak Disease</i>	1.443
3	<i>Cassava Green Mottle</i>	773
4	<i>Cassava Mosaic Disease</i>	2.658
5	<i>Healty</i>	316
<b>Jumlah</b>		<b>5.656</b>

#### **4.2 Hasil Pelatihan dan Pengujian Pada Google Colaboratory**

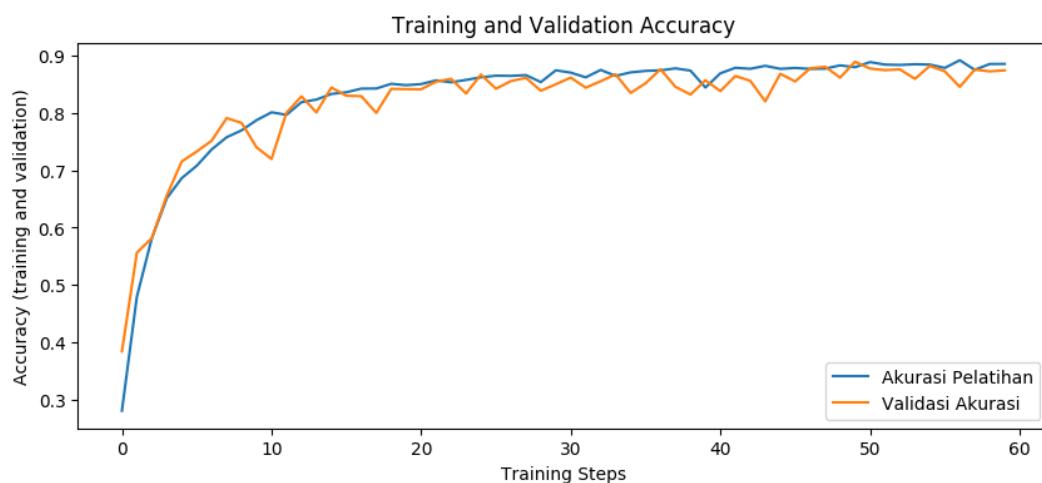
Proses pelatihan dan pengujian dilakukan pada *Google Colaboratory* dengan menggunakan arsitektur CNN yang berbeda-beda. Arsitektur CNN yang digunakan untuk pelatihan pada *Google Colaboratory* adalah *arsitektur VGG-16Net, InceptionV3* dan *MobileNetV1*.

##### **4.2.1 Arsitektur VGG16**

Pelatihan dan pengujian *dataset* menggunakan arsitektur VGG16 menggunakan gambar berukuran 64x64x3 *pixel*. Menggunakan *dataset train* Sebanyak 19.052, *dataset validasi* sebanyak 2.858, dan *dataset test* sebanyak 1.906.

Kemudian dataset dilatih menggunakan arsitektur VGG16 dengan tambahan pemodelan 3 kali *Conv2D dengan Activation 'Relu'*, 3 kali *MaxPool2D*,

*Global Average Pooling 2D, Flatten, Dense 512* menggunakan *activation ‘relu’*, *Dense 15* menggunakan *activation ‘softmax’* untuk menambah nilai akurasinya. Serta menggunakan *Early Stopping* untuk menghindari *overfitting* pada proses pelatihan dan validasi sehingga membentuk grafik seperti gambar 4.1.

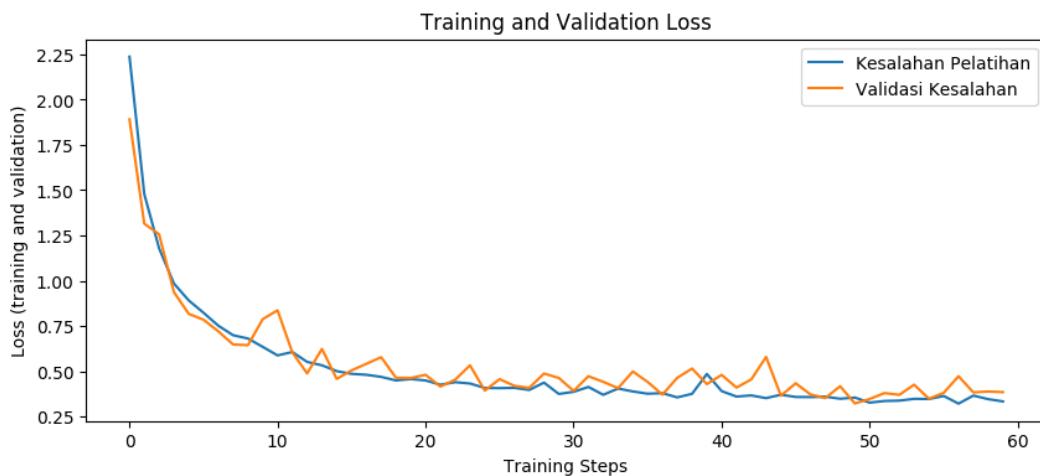


Gambar 4.1 Kurva Akurasi Training dan Validasi menggunakan Arsitektur VGG16

Pada gambar 4.1 diatas memperlihatkan kondisi kurva akurasi pelatihan yang mendekati nilai 1 di seiring bertambahnya epoch, baik pada *dataset* pelatihan maupun validasinya. Sehingga diperoleh nilai akurasi pada *epoch* terakhir (*epoch* ke-60) yaitu nilai akurasi pelatihan dan akurasi validasi masing-masing adalah 0.8860 dan 0.8747. Diketahui bahwa garis kurva yang berwarna biru merupakan akurasi pelatihan, sedangkan yang berwarna oranye merupakan akurasi validasi. Karena pada kurva dapat mencapai akurasi mendekati nilai 1, maka proses pelatihan dan validasi dinyatakan berhasil dalam melakukan klasifikasi. Selain itu akurasi pelatihan dan validasi berada pada posisi yang sama atau beriringan diantara keduanya, lalu tanpa adanya fluktuasi yang signifikan. Oleh karena itu, pada kurva

akurasi pelatihan pada dataset pelatihan dan validasi dengan arsitektur VGG16 Modifikasi disebut dengan kondisi *goodfit*.

Adapun ditampilkan hasil pelatihan pada dataset pelatihan dan validasi dalam kegagalannya pada Gambar 4.2 dibawah ini.

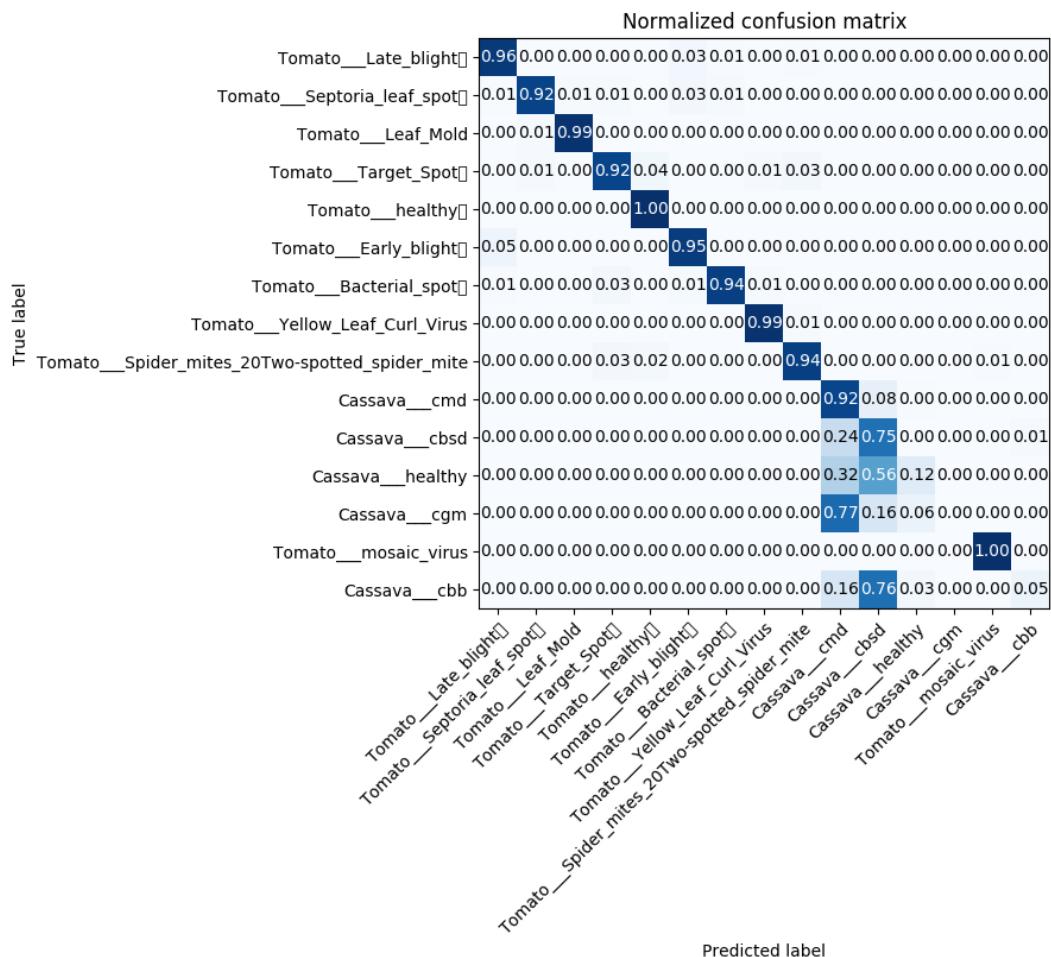


Gambar 4.2 Kurva Loss training dan validasi menggunakan Arsiektur Vgg16

Pada gambar 4.2 diatas memperlihatkan kondisi kurva kegagalan/kesalahan pelatihan yang mendekati nilai 0 seiring bertambahnya epoch, baik pada dataset pelatihan maupun validasinya. Nilai kegagalan pada epoch terakhir pada kegagalan pelatihan sebesar 0.3328, sedangkan pada kegagalan validasi sebesar 0.3848. Perbedaan kegagalan antara keduanya dibedakan pada warna garis kurva seperti yang tertera pada gambar. Karena pada kurva dapat mencapai kegagalan mendekati nilai 0, maka proses pelatihan dan validasi dinyatakan berhasil dalam melakukan klasifikasi dengan baik. Selain itu akurasi pelatihan dan validasi berada pada posisi yang sama atau beriringan diantara keduanya hingga akhir epoch yang ditentukan, meskipun kurva kegagalan validasi sempat berada pada kondisi fluktuasi diawal

epoch. Oleh karena itu, pada kurva kegagalan/kesalahan pelatihan pada dataset disebut dengan kondisi *goodfit*.

Kemudian terdapat *confusion matrix* yang digunakan untuk mengetahui tingkat keberhasilan dan kegagalan pada pengujian yang dilakukan. Dalam *confusion matrix* ini terdapat label benar dan label prediksi yang ditampilkan dengan masing-masing label terdapat nama penyakit-penyakit pada daun tomat dan daun singkong.



Gambar 4.3 Confusion Matrix menggunakan arsitektur VGG16

Pada gambar 4.3 diatas dapat diperhatikan hasil dari pengujian menggunakan arsitektur VGG16 menghasilkan matrik *confusion* yang cukup bagus meskipun ada beberapa penyakit yang memiliki nilai akurasi yang kurang dari 0,5. hal ini dipengaruhi karena jumlah dataset yang tidak seimbang pada penyakit-penyakit tertentu.

#### **4.2.2 Arsitektur *Inception* Versi 3**

Pelatihan dan pengujian *dataset* menggunakan arsitektur *Inception* Versi 3 menggunakan gambar berukuran 75x75x3 *pixel*. Menggunakan *dataset train* Sebanyak 19.052, *dataset validasi* sebanyak 2.858, dan *dataset test* sebanyak 1.906.

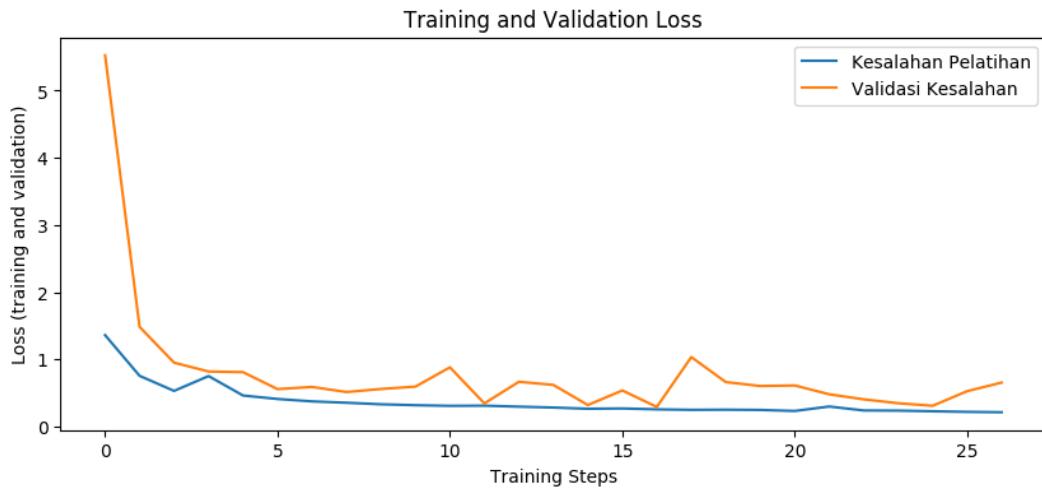
Kemudian dataset dilatih menggunakan arsitektur *Inceptionv3* dengan tambahan pemodelan *Flatten*, *Dropout* sebesar 0.2, *Dense* 512 menggunakan *activation ‘relu’*, *Dense* 15 menggunakan *activation ‘softmax’* untuk menambah nilai akurasinya. Serta menggunakan *Early Stopping* untuk menghindari *overfitting* pada proses pelatihan dan validasi sehingga membentuk grafik seperti gambar 4.4.



Gambar 4.4 Kurva akurasi training dan validasi Menggunakan Arsitektur InceptionV3

Pada gambar 4.4 diatas memperlihatkan kondisi kurva akurasi pelatihan yang tidak stabil seiring bertambahnya epoch, terutama pada akurasi validasinya. Sehingga diperoleh nilai akurasi pada *epoch* terakhir (*epoch* ke-27) yaitu nilai akurasi pelatihan dan akurasi validasi masing-masing adalah 0.9230 dan 0.7706. Diketahui bahwa garis kurva yang berwarna biru merupakan akurasi pelatihan, sedangkan yang berwarna oranye merupakan akurasi validasi. Karena pada kurva tidak stabil meskipun nilai akurasi mendekati nilai 1. Hasil pelatihan dan validasi dapat melakukan klasifikasi meskipun ada beberapa data yang salah.

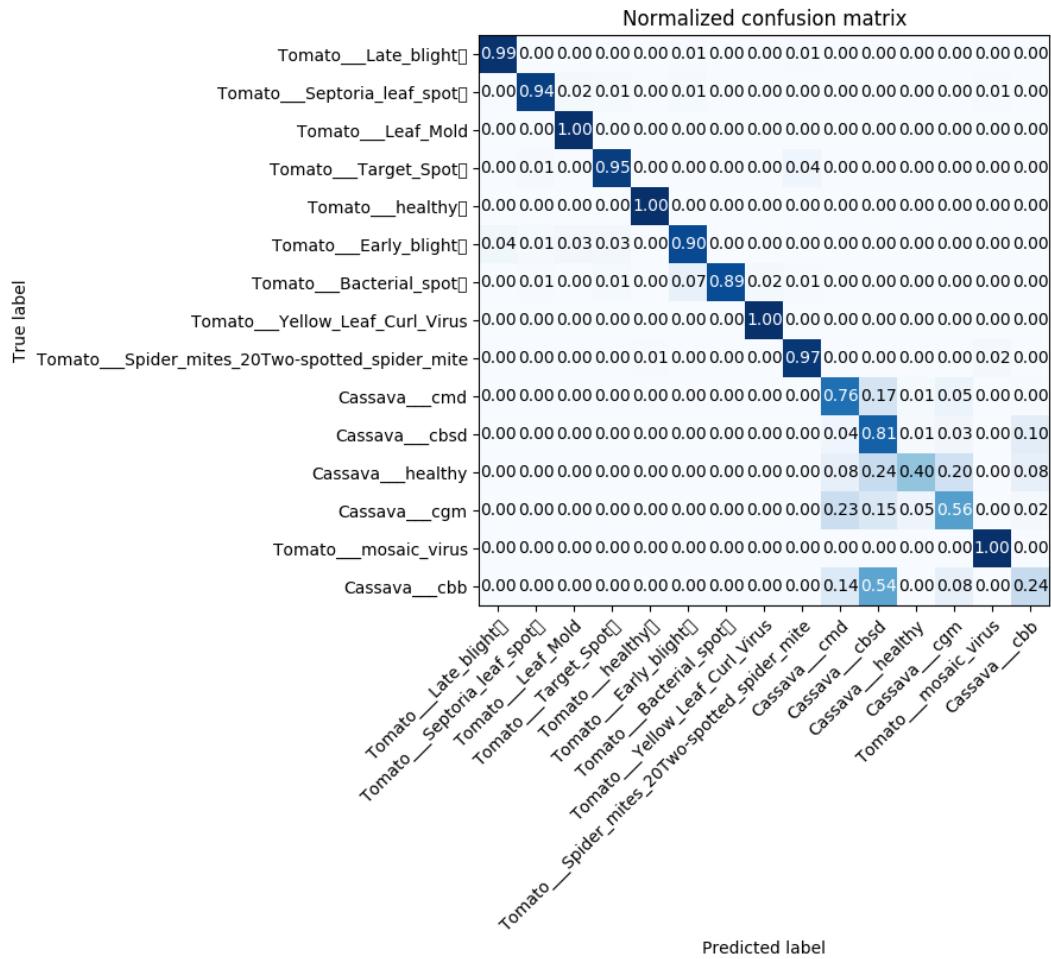
Adapun ditampilkan hasil pelatihan pada dataset pelatihan dan validasi dalam kegagalannya pada Gambar 4.5 dibawah ini.



Gambar 4.5 Kurva Loss Training dan validasi Menggunakan arsitektur InceptionV3

Pada gambar 4.5 diatas memperlihatkan kondisi kurva kegagalan/kesalahan pelatihan yang tidak stabil, bahkan pada *epoch* terakhir mengalami kenaikan *loss* pada data validasi. Nilai kegagalan pada epoch terakhir pada kegagalan pelatihan sebesar 0.2117, sedangkan pada kegagalan validasi sebesar 0,629. Perbedaan kegagalan antara keduanya dibedakan pada warna garis kurva seperti yang tertera pada gambar.

Kemudian terdapat *confusion matrix* yang digunakan untuk mengetahui tingkat keberhasilan dan kegagalan pada pengujian yang dilakukan. Dalam *confusion matrix* ini terdapat label benar dan label prediksi yang ditampilkan dengan masing-masing label terdapat nama penyakit-penyakit pada daun tomat dan daun singkong.



Gambar 4.6 Confusion Matrix menggunakan Arsitektur InceptionV3

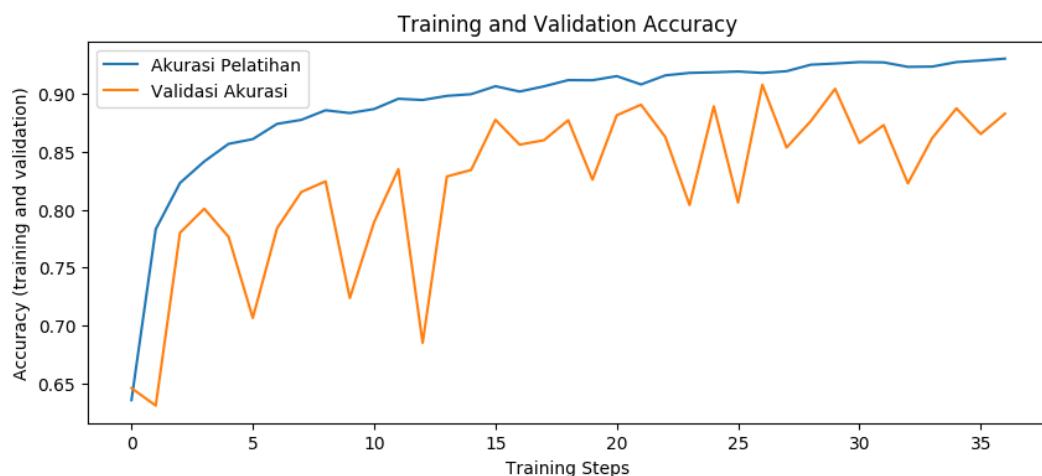
Pada gambar 4.6 diatas dapat diperhatikan hasil dari pengujian menggunakan arsitektur *InceptionV3* menghasilkan matrik confusion yang cukup bagus meskipun ada beberapa penyakit yang memiliki nilai akurasi yang kurang dari 0.5. hal ini dipengaruhi karena jumlah dataset yang tidak seimbang pada penyakit-penyakit tertentu

#### 4.2.3 Arsitektur *MobileNet* Versi 1

Pelatihan dan pengujian *dataset* menggunakan arsitektur *MobileNetV1* menggunakan gambar berukuran 64x64x3 pixel. Menggunakan *dataset train*

Sebanyak 19.052, *dataset* validasi sebanyak 2.858, dan *dataset test* sebanyak 1.906.

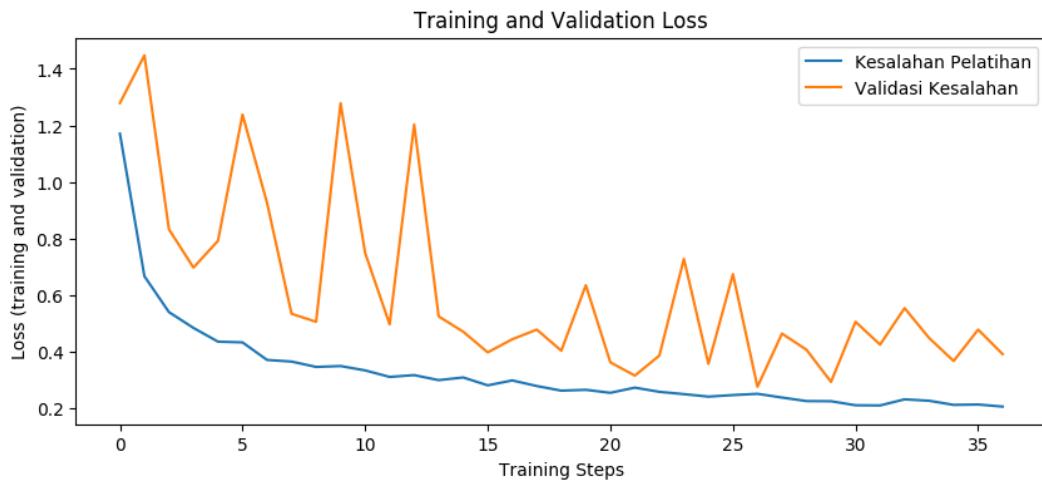
Kemudian dataset dilatih menggunakan arsitektur MobileNetV1 dengan tambahan pemodelan Conv2D, Dropout sebesar 0.2, Global Average Pooling 2D, Flatten, dan Dense 15 menggunakan activation ‘softmax’ untuk menambah nilai kurasinya. Serta menggunakan Early Stopping untuk menghindari overfitting pada proses pelatihan dan validasi sehingga membentuk grafik seperti gambar 4.7.



Gambar 4.7 Kurva Akurasi Training Dan Validasi Menggunakan Arsitektur MobileNetV1

Pada gambar 4.7 diatas memperlihatkan kondisi kurva akurasi pelatihan yang mendekati nilai 1 di seiring bertambahnya epoch, meskipun pada akurasi validasi tidak stabil. Diperoleh nilai akurasi pada epoch terakhir (epoch ke-37) yaitu nilai akurasi pelatihan dan akurasi validasi masing-masing adalah 0.9303 dan 0.8828. Diketahui bahwa garis kurva yang berwarna biru merupakan akurasi pelatihan, sedangkan yang berwarna oranye merupakan akurasi validasi. Karena pada kurva dapat mencapai akurasi mendekati nilai 1, maka proses pelatihan dan validasi dinyatakan berhasil dalam melakukan klasifikasi.

Adapun ditampilkan hasil pelatihan pada dataset pelatihan dan validasi dalam kegagalan pada Gambar 4.8 dibawah ini.

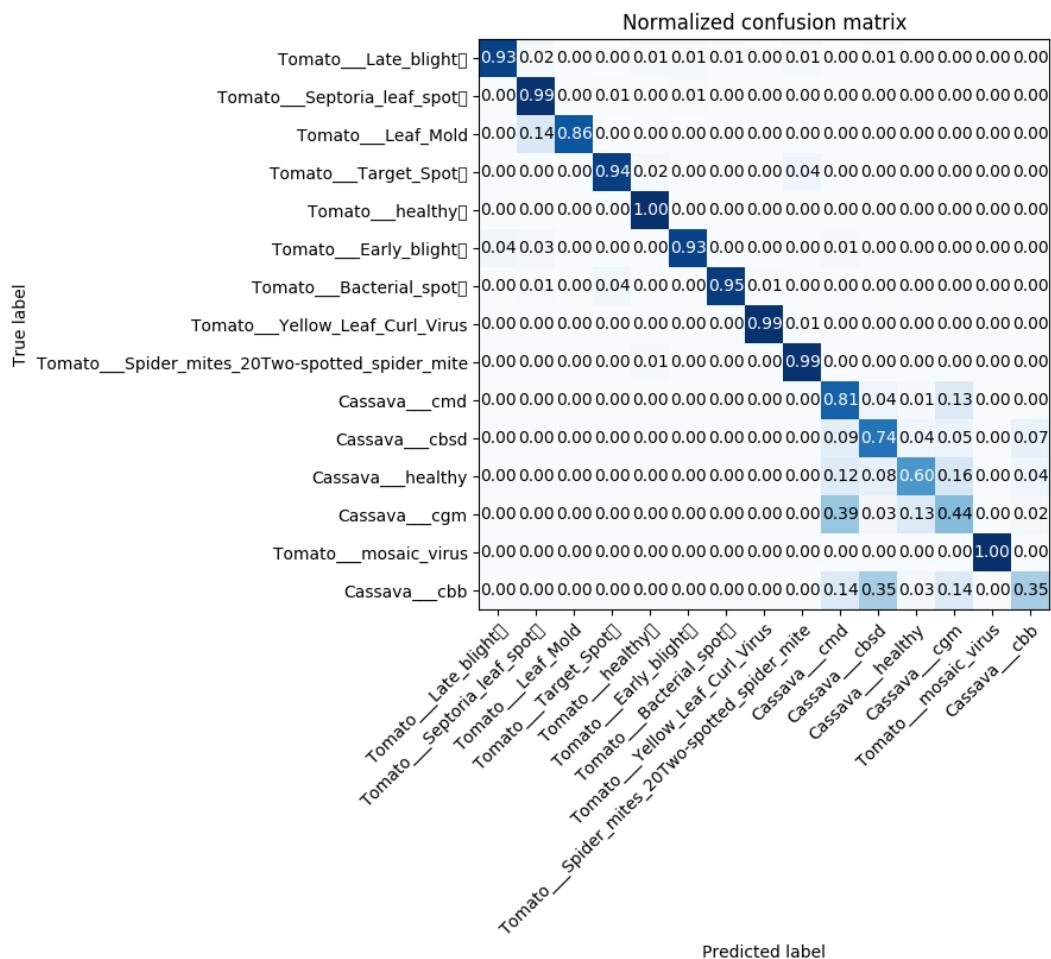


Gambar 4.8 Kurva Loss Training dan Validasi menggunakan Arsitektur MobileNetV1

Pada gambar 4.8 diatas memperlihatkan kondisi kurva kegagalan/kesalahan pelatihan yang mendekati nilai 0 seiring bertambahnya epoch, baik pada dataset pelatihan maupun validasinya. Nilai kegagalan pada epoch terakhir pada kegagalan pelatihan sebesar 0.2074, sedangkan pada kegagalan validasi sebesar 0.3919. Perbedaan kegagalan antara keduanya dibedakan pada warna garis kurva seperti yang tertera pada gambar. Karena pada kurva dapat mencapai kegagalan mendekati nilai 0, maka proses pelatihan dan validasi dinyatakan berhasil dalam melakukan klasifikasi dengan baik.

Kemudian terdapat confusion matrix yang digunakan untuk mengetahui tingkat keberhasilan dan kegagalan pada pengujian yang dilakukan. Dalam *confusion matrix* ini terdapat label benar dan label prediksi yang ditampilkan

dengan masing-masing label terdapat nama penyakit-penyakit pada daun tomat dan daun singkong.



Gambar 4.9 Confusion Matrix Menggunakan Arsitektur MobileNetV1

Pada gambar 4.9 diatas dapat diperhatikan hasil dari pengujian menggunakan arsitektur *MobileNetV1* menghasilkan matrik *confusion* yang cukup bagus meskipun ada beberapa penyakit yang memiliki nilai akurasi yang kurang dari 0.5. hal ini dipengaruhi karena jumlah dataset yang tidak seimbang pada penyakit-penyakit tertentu.

### 4.3 Hasil Pengujian Menggunakan Android

Proses pengujian dilakukan pada *Smartphone* yang telah terinstal Aplikasi *Plant Diseases*. Pengujian dilakukan menggunakan 10 gambar pada setiap penyakit. Pengujian juga terbagi dalam 3 arsitektur berdasarkan hasil pelatihan pada *Google Colaboratory*, dan telah di *convert* menjadi *file TFLite* sehingga bisa diaplikasikan pada Android.

#### 4.3.1 Pengujian Android Menggunakan VGG16

Pengujian dilakukan dengan memasukan gambar dengan cara memotret langsung atau melalui *import* galeri pada aplikasi *Plant Diseases*. Data yang digunakan sebanyak 10 pada setiap penyakit. Kemudian gambar yang telah dimasukan kedalam aplikasi akan dilakukan identifikasi sehingga dapat di prediksi keadaan daun tomat atau daun singkong dalam kedaan sehat atau berpenyakit. Pada aplikasi terdapat nilai propabilitas yang menunjukkan nilai presentase kebenaran berdasarkan citra yang diperoleh dari gambar tersebut. Berdasarkan hal tersebut maka diperoleh seperti pada tabel 4.3.

Tabel 4.3 Data Uji Android Menggunakan Arsitektur VGG16

Uji ke-	Respon Deteksi (ms)	Data uji	Hasil Prediksi (Tingkat Probabilitas)	Benar / Salah
1	713	<i>T. Bacterial spot</i>	<i>T. Bacterial spot (100%)</i>	Benar
2	650	<i>T. Bacterial spot</i>	<i>T. Bacterial spot (99.99%)</i>	Benar
3	729	<i>T. Bacterial spot</i>	<i>T. Bacterial spot (99.99%)</i>	Benar
4	659	<i>T. Bacterial spot</i>	<i>T. Bacterial spot (99.99%)</i>	Benar
5	1832	<i>T. Bacterial spot</i>	<i>T. Bacterial spot (100%)</i>	Benar
6	708	<i>T. Bacterial spot</i>	<i>T. Bacterial spot (100%)</i>	Benar
7	680	<i>T. Bacterial spot</i>	<i>T. Bacterial spot (100%)</i>	Benar
8	673	<i>T. Bacterial spot</i>	<i>T. Bacterial spot (100%)</i>	Benar
9	670	<i>T. Bacterial spot</i>	<i>T. Bacterial spot (100%)</i>	Benar
10	688	<i>T. Bacterial spot</i>	<i>T. Bacterial spot (99.99%)</i>	Benar
11	690	<i>T. Early blight</i>	<i>T. Early blight (99.93%)</i>	Benar
12	754	<i>T. Early blight</i>	<i>T. Early blight (51.46%)</i>	Benar

13	647	<i>T. Early blight</i>	<i>T. Early blight</i> (98.79%)	Benar
14	707	<i>T. Early blight</i>	<i>T. Early blight</i> (99.66%)	Benar
15	675	<i>T. Early blight</i>	<i>T. Early blight</i> (99.99%)	Benar
16	646	<i>T. Early blight</i>	<i>T. Early blight</i> (99.95%)	Benar
17	646	<i>T. Early blight</i>	<i>T. Early blight</i> (99.73%)	Benar
18	664	<i>T. Early blight</i>	<i>T. Early blight</i> (100%)	Benar
19	684	<i>T. Early blight</i>	<i>T. Early blight</i> (67.86%)	Benar
20	641	<i>T. Early blight</i>	<i>T. Early blight</i> (97.97%)	Benar
21	669	<i>T. healthy</i>	<i>T. healthy</i> (99.94%)	Benar
22	779	<i>T. healthy</i>	<i>T. healthy</i> (99.91%)	Benar
23	752	<i>T. healthy</i>	<i>T. healthy</i> (99.99%)	Benar
24	667	<i>T. healthy</i>	<i>T. healthy</i> (98.60%)	Benar
25	641	<i>T. healthy</i>	<i>T. healthy</i> (93.19%)	Benar
26	614	<i>T. healthy</i>	<i>T. healthy</i> (99.73%)	Benar
27	650	<i>T. healthy</i>	<i>T. healthy</i> (99.91%)	Benar
28	772	<i>T. healthy</i>	<i>T. healthy</i> (91.34%)	Benar
29	655	<i>T. healthy</i>	<i>T. healthy</i> (99.99%)	Benar
30	671	<i>T. healthy</i>	<i>T. healthy</i> (99.94%)	Benar
31	1929	<i>T. Late blight</i>	<i>T. Late blight</i> (100%)	Benar
32	648	<i>T. Late blight</i>	<i>T. Late blight</i> (99.99%)	Benar
33	663	<i>T. Late blight</i>	<i>T. Late blight</i> (99.96%)	Benar
34	711	<i>T. Late blight</i>	<i>T. Late blight</i> (99.99%)	Benar
35	638	<i>T. Late blight</i>	<i>T. Late blight</i> (96.74%)	Benar
36	666	<i>T. Late blight</i>	<i>T. Late blight</i> (99.98%)	Benar
37	674	<i>T. Late blight</i>	<i>T. Late blight</i> (99.99%)	Benar
38	698	<i>T. Late blight</i>	<i>T. Late blight</i> (99.99%)	Benar
39	672	<i>T. Late blight</i>	<i>T. Late blight</i> (99.93%)	Benar
40	689	<i>T. Late blight</i>	<i>T. Late blight</i> (99.99%)	Benar
41	614	<i>T. Leaf Mold</i>	<i>T. Leaf Mold</i> (99.93%)	Benar
42	704	<i>T. Leaf Mold</i>	<i>T. Leaf Mold</i> (99.91%)	Benar
43	633	<i>T. Leaf Mold</i>	<i>T. Leaf Mold</i> (99.99%)	Benar
44	2228	<i>T. Leaf Mold</i>	<i>T. Leaf Mold</i> (97.33%)	Benar
45	651	<i>T. Leaf Mold</i>	<i>T. Leaf Mold</i> (99.99%)	Benar
46	2188	<i>T. Leaf Mold</i>	<i>T. Leaf Mold</i> (99.93%)	Benar
47	723	<i>T. Leaf Mold</i>	<i>T. Leaf Mold</i> (99.79%)	Benar
48	715	<i>T. Leaf Mold</i>	<i>T. Leaf Mold</i> (99.99%)	Benar
49	705	<i>T. Leaf Mold</i>	<i>T. Leaf Mold</i> (99.30%)	Benar
50	708	<i>T. Leaf Mold</i>	<i>T. Leaf Mold</i> (99.20%)	Benar
51	715	<i>T. Septoria leaf spot</i>	<i>T. Septoria leaf spot</i> (99.99%)	Benar
52	674	<i>T. Septoria leaf spot</i>	<i>T. Septoria leaf spot</i> (99.99%)	Benar
53	668	<i>T. Septoria leaf spot</i>	<i>T. Septoria leaf spot</i> (99.99%)	Benar
54	666	<i>T. Septoria leaf spot</i>	<i>T. Septoria leaf spot</i> (99.99%)	Benar
55	686	<i>T. Septoria leaf spot</i>	<i>T. Septoria leaf spot</i> (99.99%)	Benar
56	731	<i>T. Septoria leaf spot</i>	<i>T. Septoria leaf spot</i> (100%)	Benar

57	713	<i>T. Septoria leaf spot</i>	<i>T. Septoria leaf spot (99.99%)</i>	Benar
58	727	<i>T. Septoria leaf spot</i>	<i>T. Septoria leaf spot (99.98%)</i>	Benar
59	656	<i>T. Septoria leaf spot</i>	<i>T. Septoria leaf spot (100%)</i>	Benar
60	736	<i>T. Septoria leaf spot</i>	<i>T. Septoria leaf spot (99.96%)</i>	Benar
61	616	<i>T. Spider mites T.S.S.M</i>	<i>T. Spider M.T.S.S.M (99.99%)</i>	Benar
62	681	<i>T. Spider mites T.S.S.M</i>	<i>T. Spider M.T.S.S.M (99.94%)</i>	Benar
63	688	<i>T. Spider mites T.S.S.M</i>	<i>T. Spider M.T.S.S.M (99.84%)</i>	Benar
64	718	<i>T. Spider mites T.S.S.M</i>	<i>T. Spider M.T.S.S.M (99.75%)</i>	Benar
65	698	<i>T. Spider mites T.S.S.M</i>	<i>T. Spider M.T.S.S.M (99.80%)</i>	Benar
66	641	<i>T. Spider mites T.S.S.M</i>	<i>T. Spider M.T.S.S.M (97.04%)</i>	Benar
67	697	<i>T. Spider mites T.S.S.M</i>	<i>T. Spider M.T.S.S.M (98.32%)</i>	Benar
68	766	<i>T. Spider mites T.S.S.M</i>	<i>T. Spider M.T.S.S.M (99.99%)</i>	Benar
69	1899	<i>T. Spider mites T.S.S.M</i>	<i>T. Target Spot (48.79%)</i>	Salah
70	644	<i>T. Spider mites T.S.S.M</i>	<i>T. Spider M.T.S.S.M (99.99%)</i>	Benar
71	643	<i>T. Target Spot</i>	<i>T. Target Spot (95.80%)</i>	Benar
72	623	<i>T. Target Spot</i>	<i>T. Target Spot (99.99%)</i>	Benar
73	701	<i>T. Target Spot</i>	<i>T. Target Spot (99.98%)</i>	Benar
74	648	<i>T. Target Spot</i>	<i>T. Target Spot (99.71%)</i>	Benar
75	692	<i>T. Target Spot</i>	<i>T. Target Spot (98.21%)</i>	Benar
76	658	<i>T. Target Spot</i>	<i>T. Spider M.T.S.S.M (55.56%)</i>	Salah
77	637	<i>T. Target Spot</i>	<i>T. Target Spot (93.07%)</i>	Benar
78	681	<i>T. Target Spot</i>	<i>T. Target Spot (99.98%)</i>	Benar
79	650	<i>T. Target Spot</i>	<i>T. Target Spot (99.77%)</i>	Benar
80	712	<i>T. Target Spot</i>	<i>T. Target Spot (99.91%)</i>	Benar
81	678	<i>T. Mosaic virus</i>	<i>T. Mosaic virus (51.79%)</i>	Benar
82	671	<i>T. Mosaic virus</i>	<i>T. Mosaic virus (99.17%)</i>	Benar
83	734	<i>T. Mosaic virus</i>	<i>T. Mosaic virus (96.23%)</i>	Benar
84	1975	<i>T. Mosaic virus</i>	<i>T. Septoria leaf spot (94.92%)</i>	Salah
85	692	<i>T. Mosaic virus</i>	<i>T. Mosaic virus (98.87%)</i>	Benar
86	687	<i>T. Mosaic virus</i>	<i>T. Mosaic virus (99.96%)</i>	Benar
87	653	<i>T. Mosaic virus</i>	<i>T. Septoria leaf spot (46.12%)</i>	Salah
88	693	<i>T. Mosaic virus</i>	<i>T. Mosaic virus (41.39%)</i>	Benar
89	1799	<i>T. Mosaic virus</i>	<i>T. Mosaic virus (99.20%)</i>	Benar
90	680	<i>T. Mosaic virus</i>	<i>T. Mosaic virus (97.29%)</i>	Benar
91	699	<i>T. Yellow Leaf C.V.</i>	<i>T. Yellow Leaf C.V. (100%)</i>	Benar
92	688	<i>T. Yellow Leaf C.V.</i>	<i>T. Yellow Leaf C.V. (100%)</i>	Benar
93	1747	<i>T. Yellow Leaf C.V.</i>	<i>T. Yellow Leaf C.V. (100%)</i>	Benar
94	608	<i>T. Yellow Leaf C.V.</i>	<i>T. Yellow Leaf C.V. (100%)</i>	Benar
95	665	<i>T. Yellow Leaf C.V.</i>	<i>T. Yellow Leaf C.V. (100%)</i>	Benar
96	695	<i>T. Yellow Leaf C.V.</i>	<i>T. Yellow Leaf C.V. (100%)</i>	Benar
97	601	<i>T. Yellow Leaf C.V.</i>	<i>T. Yellow Leaf C.V. (100%)</i>	Benar
98	687	<i>T. Yellow Leaf C.V.</i>	<i>T. Yellow Leaf C.V. (100%)</i>	Benar
99	644	<i>T. Yellow Leaf C.V.</i>	<i>T. Yellow Leaf C.V. (100%)</i>	Benar
100	749	<i>T. Yellow Leaf C.V.</i>	<i>T. Yellow Leaf C.V. (100%)</i>	Benar

<b>101</b>	1839	<i>CBB</i>	<i>CBSD (72.93%)</i>	Salah
<b>102</b>	672	<i>CBB</i>	<i>CBSD (72.29%)</i>	Salah
<b>103</b>	676	<i>CBB</i>	<i>CBSD (65.90%)</i>	Salah
<b>104</b>	751	<i>CBB</i>	<i>CBSD (70.81%)</i>	Salah
<b>105</b>	706	<i>CBB</i>	<i>CBSD (70.25%)</i>	Salah
<b>106</b>	651	<i>CBB</i>	<i>CBSD (69.85%)</i>	Salah
<b>107</b>	675	<i>CBB</i>	<i>CBSD (71.92%)</i>	Salah
<b>108</b>	676	<i>CBB</i>	<i>CBSD (72.64%)</i>	Salah
<b>109</b>	711	<i>CBB</i>	<i>CBSD (71.25%)</i>	Salah
<b>110</b>	629	<i>CBB</i>	<i>CBSD (63.73%)</i>	Salah
<b>111</b>	616	<i>CBSD</i>	<i>CBSD (87.39%)</i>	Benar
<b>112</b>	1729	<i>CBSD</i>	<i>CBSD (68.42%)</i>	Benar
<b>113</b>	650	<i>CBSD</i>	<i>CBSD (80.98%)</i>	Benar
<b>114</b>	1785	<i>CBSD</i>	<i>CMD (67.63%)</i>	Salah
<b>115</b>	1782	<i>CBSD</i>	<i>CBSD (86.45%)</i>	Benar
<b>116</b>	661	<i>CBSD</i>	<i>CBSD (93.36%)</i>	Benar
<b>117</b>	622	<i>CBSD</i>	<i>CBSD (72.70%)</i>	Benar
<b>118</b>	1642	<i>CBSD</i>	<i>CBSD (84.36%)</i>	Benar
<b>119</b>	648	<i>CBSD</i>	<i>CBSD (80.39%)</i>	Benar
<b>120</b>	672	<i>CBSD</i>	<i>CBSD (62.53%)</i>	Benar
<b>121</b>	2070	<i>CGM</i>	<i>CMD (60.89%)</i>	Salah
<b>122</b>	704	<i>CGM</i>	<i>CMD (76.88%)</i>	Salah
<b>123</b>	653	<i>CGM</i>	<i>CMD (77.33%)</i>	Salah
<b>124</b>	614	<i>CGM</i>	<i>CMD (77.24%)</i>	Salah
<b>125</b>	644	<i>CGM</i>	<i>CMD (47.30%)</i>	Salah
<b>126</b>	705	<i>CGM</i>	<i>CBSD (45.08%)</i>	Salah
<b>127</b>	660	<i>CGM</i>	<i>CMD (69.47%)</i>	Salah
<b>128</b>	732	<i>CGM</i>	<i>CBSD (43.71%)</i>	Salah
<b>129</b>	672	<i>CGM</i>	<i>CMD (68.69%)</i>	Salah
<b>130</b>	639	<i>CGM</i>	<i>CMD (77.33%)</i>	Salah
<b>131</b>	727	<i>CMD</i>	<i>CMD (69.32%)</i>	Benar
<b>132</b>	714	<i>CMD</i>	<i>CMD (86.53%)</i>	Benar
<b>133</b>	677	<i>CMD</i>	<i>CMD (86.48%)</i>	Benar
<b>134</b>	708	<i>CMD</i>	<i>CMD (88.29%)</i>	Benar
<b>135</b>	645	<i>CMD</i>	<i>CMD (92.78%)</i>	Benar
<b>136</b>	648	<i>CMD</i>	<i>CMD (97.11%)</i>	Benar
<b>137</b>	643	<i>CMD</i>	<i>CMD (97.85%)</i>	Benar
<b>138</b>	637	<i>CMD</i>	<i>CMD (91.69%)</i>	Benar
<b>139</b>	647	<i>CMD</i>	<i>CMD (79.84%)</i>	Benar
<b>140</b>	704	<i>CMD</i>	<i>CMD (92.88%)</i>	Benar
<b>141</b>	683	<i>Healty</i>	<i>CBSD (53.98%)</i>	Salah
<b>142</b>	634	<i>Healty</i>	<i>CMD (62.38%)</i>	Salah
<b>143</b>	698	<i>Healty</i>	<i>CBSD (55.44%)</i>	Salah
<b>144</b>	653	<i>Healty</i>	<i>CBSD (47.54%)</i>	Salah

<b>145</b>	624	<i>Healty</i>	<i>CBSD (58.36%)</i>	Salah
<b>146</b>	635	<i>Healty</i>	<i>CBSD (40.48%)</i>	Salah
<b>147</b>	634	<i>Healty</i>	<i>CBSD (73.42%)</i>	Salah
<b>148</b>	659	<i>Healty</i>	<i>CBSD (58.84%)</i>	Salah
<b>149</b>	667	<i>Healty</i>	<i>CBSD (44.02%)</i>	Salah
<b>150</b>	639	<i>Healty</i>	<i>CBSD (45.37%)</i>	Salah

Dari tabel pengujian tersebut dapat dilihat bahwa hasil prediksi menggunakan arsitektur VGG16 kurang cocok untuk digunakan pada android. Meskipun memiliki kurva yang baik pada saat proses pelatihan dan validasi namun VGG16 kurang cocok untuk diaplikasikan kedalam android. Berdasarkan tabel 4.3 dapat dihitung nilai akurasi dan *loss* dengan persamaan berikut.

$$\% \text{akurasi} = \frac{\text{jumlahprediksibenar}}{\text{jumlahdatapengujian}} \times 100\%$$

$$\% \text{akurasi} = \frac{115}{150} \times 100\% = 76,67\%$$

$$\% \text{kesalahan} = \frac{\text{jumlahprediksisalah}}{\text{jumlahdatapengujian}} \times 100\%$$

$$\% \text{kesalahan} = \frac{35}{150} \times 100\% = 23,33\%$$

Dari perhitungan diatas dapat dilihat nilai akurasi yang didapat dengan menggunakan arsitektur VGG16 sebesar 76,67% dan kesalahan (*Loss*) sebesar 23,33%. Hal ini dapat disimpulkan arsitektur VGG16 kurang cocok untuk diaplikasikan pada android. Untuk mengetahui tingkat akurasi prediksi pada setiap penyakit tanaman dapat dilihat pada *confusion matrix* yang disajikan pada gambar 4.5 berikut.

Normalized confusion matrix															
Label Benar	T. Bacterial spot	T. Early blight	T. healthy	T. Late blight	T. Leaf Mold	T. Septoria leaf spot	T. Spider mites T.S.S.M	T. Target Spot	T. Mosaic virus	T. Yellow Leaf C.V.	CBB	CBSD	CGM	CMD	C.Healthy
	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.10	0.00	0.00	0.00	0.00	0.00	0.00	
	0.00	0.00	0.00	0.00	0.00	0.00	0.10	0.90	0.00	0.00	0.00	0.00	0.00	0.00	
	0.00	0.00	0.00	0.00	0.00	0.20	0.00	0.00	0.80	0.00	0.00	0.00	0.00	0.00	
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.90	0.00	0.10	
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.20	0.00	0.80	
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.90	0.00	0.10	
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	

Gambar 4.10 Confusion Matrix Pengujian Pada Android Menggunakan Arsitektur VGG16

Berdasarkan *matrix Confusion* dapat dilihat hasil pengujian ada beberapa penyakit pada daun singkong tidak terdeteksi sama sekali. hal ini menandakan bahwa arsitektur VGG16 tidak direkomendasikan untuk android.

#### 4.3.2 Pengujian Android Menggunakan *Inception* Versi 3

Pengujian dilakukan dengan memasukan gambar dengan cara memotret langsung atau melalui *import* galeri pada aplikasi *Plant Diseases*. Data yang digunakan sebanyak 10 pada setiap penyakit. Kemudian gambar yang telah dimasukan kedalam aplikasi akan dilakukan identifikasi sehingga dapat di prediksi keadaan daun tomat atau daun singkong dalam kedaan sehat atau berpenyakit. Pada aplikasi terdapat nilai propabilitas yang menunjukkan nilai presentase kebenaran

berdasarkan citra yang diperoleh dari gambar tersebut. Berdasarkan hal tersebut maka diperoleh seperti pada tabel 4.4.

*Tabel 4.4 Data Uji Android Menggunakan Arsitektur Inception*

Uji ke-	Respon Deteksi (ms)	Data uji	Hasil Prediksi (Tingkat Probabilitas)	Benar/Salah
1	1001	<i>T. Bacterial spot</i>	<i>T. Bacterial spot (99.99%)</i>	Benar
2	2278	<i>T. Bacterial spot</i>	<i>T. Early blight (51.17%)</i>	Salah
3	2002	<i>T. Bacterial spot</i>	<i>T. Bacterial spot (99.93%)</i>	Benar
4	414	<i>T. Bacterial spot</i>	<i>T. Bacterial spot (99.48%)</i>	Benar
5	344	<i>T. Bacterial spot</i>	<i>T. Bacterial spot (100%)</i>	Benar
6	618	<i>T. Bacterial spot</i>	<i>T. Bacterial spot (99.98%)</i>	Benar
7	483	<i>T. Bacterial spot</i>	<i>T. Bacterial spot (100%)</i>	Benar
8	468	<i>T. Bacterial spot</i>	<i>T. Bacterial spot (100%)</i>	Benar
9	2191	<i>T. Bacterial spot</i>	<i>T. Bacterial spot (98.66%)</i>	Benar
10	1263	<i>T. Bacterial spot</i>	<i>T. Bacterial spot (100%)</i>	Benar
11	1709	<i>T. Early blight</i>	<i>T. Late blight (99.31%)</i>	Salah
12	353	<i>T. Early blight</i>	<i>T. Early blight (95.32%)</i>	Benar
13	552	<i>T. Early blight</i>	<i>T. Early blight (99.88%)</i>	Benar
14	522	<i>T. Early blight</i>	<i>T. Early blight (97.30%)</i>	Benar
15	1640	<i>T. Early blight</i>	<i>T. Early blight (99.92%)</i>	Benar
16	536	<i>T. Early blight</i>	<i>T. Early blight (97.90%)</i>	Benar
17	554	<i>T. Early blight</i>	<i>T. Early blight (96.53%)</i>	Benar
18	496	<i>T. Early blight</i>	<i>T. Late blight (98.06%)</i>	Salah
19	529	<i>T. Early blight</i>	<i>T. Early blight (99.76%)</i>	Benar
20	553	<i>T. Early blight</i>	<i>T. Early blight (99.92%)</i>	Benar
21	515	<i>T. healthy</i>	<i>T. healthy (99.99%)</i>	Benar
22	566	<i>T. healthy</i>	<i>T. healthy (99.99%)</i>	Benar
23	246	<i>T. healthy</i>	<i>T. healthy (99.73%)</i>	Benar
24	414	<i>T. healthy</i>	<i>T. healthy (99.99%)</i>	Benar
25	518	<i>T. healthy</i>	<i>T. healthy (99.98%)</i>	Benar
26	474	<i>T. healthy</i>	<i>T. healthy (99.96%)</i>	Benar
27	409	<i>T. healthy</i>	<i>T. healthy (99.99%)</i>	Benar
28	547	<i>T. healthy</i>	<i>T. healthy (99.98%)</i>	Benar
29	499	<i>T. healthy</i>	<i>T. healthy (99.87%)</i>	Benar
30	487	<i>T. healthy</i>	<i>T. healthy (99.99%)</i>	Benar
31	549	<i>T. Late blight</i>	<i>T. Late blight (99.99%)</i>	Benar
32	544	<i>T. Late blight</i>	<i>T. Late blight (99.99%)</i>	Benar
33	530	<i>T. Late blight</i>	<i>T. Late blight (99.99%)</i>	Benar
34	531	<i>T. Late blight</i>	<i>T. Late blight (99.99%)</i>	Benar
35	527	<i>T. Late blight</i>	<i>T. Late blight (99.99%)</i>	Benar
36	488	<i>T. Late blight</i>	<i>T. Late blight (99.99%)</i>	Benar

37	429	<i>T. Late blight</i>	<i>T. Late blight</i> (99.96%)	Benar
38	527	<i>T. Late blight</i>	<i>T. Late blight</i> (99.99%)	Benar
39	609	<i>T. Late blight</i>	<i>T. Late blight</i> (99.99%)	Benar
40	412	<i>T. Late blight</i>	<i>T. Late blight</i> (99.98%)	Benar
41	495	<i>T. Leaf Mold</i>	<i>T. Leaf Mold</i> (59.36%)	Benar
42	523	<i>T. Leaf Mold</i>	<i>T. Spider M.T.S.S.M</i> (63.25%)	Salah
43	512	<i>T. Leaf Mold</i>	<i>T. Leaf Mold</i> (99.82%)	Benar
44	531	<i>T. Leaf Mold</i>	<i>T. Leaf Mold</i> (85.37%)	Benar
45	1960	<i>T. Leaf Mold</i>	<i>T. Leaf Mold</i> (99.95%)	Benar
46	491	<i>T. Leaf Mold</i>	<i>T. Leaf Mold</i> (99.65%)	Benar
47	550	<i>T. Leaf Mold</i>	<i>T. Leaf Mold</i> (99.91%)	Benar
48	364	<i>T. Leaf Mold</i>	<i>T. Leaf Mold</i> (99.91%)	Benar
49	534	<i>T. Leaf Mold</i>	<i>T. Early blight</i> (45.35%)	Salah
50	441	<i>T. Leaf Mold</i>	<i>T. Leaf Mold</i> (95.19%)	Benar
51	528	<i>T. Septoria leaf spot</i>	<i>T. Septoria leaf spot</i> (99.95%)	Benar
52	613	<i>T. Septoria leaf spot</i>	<i>T. Septoria leaf spot</i> (99.99%)	Benar
53	547	<i>T. Septoria leaf spot</i>	<i>T. Septoria leaf spot</i> (99.99%)	Benar
54	553	<i>T. Septoria leaf spot</i>	<i>T. Septoria leaf spot</i> (99.70%)	Benar
55	541	<i>T. Septoria leaf spot</i>	<i>T. Septoria leaf spot</i> (99.99%)	Benar
56	512	<i>T. Septoria leaf spot</i>	<i>T. Septoria leaf spot</i> (99.92%)	Benar
57	2008	<i>T. Septoria leaf spot</i>	<i>T. Septoria leaf spot</i> (99.98%)	Benar
58	400	<i>T. Septoria leaf spot</i>	<i>T. Septoria leaf spot</i> (99.72%)	Benar
59	545	<i>T. Septoria leaf spot</i>	<i>T. Septoria leaf spot</i> (99.96%)	Benar
60	493	<i>T. Septoria leaf spot</i>	<i>T. Septoria leaf spot</i> (99.99%)	Benar
61	534	<i>T. Spider mites T.S.S.M</i>	<i>T. Spider M.T.S.S.M</i> (99.99%)	Benar
62	474	<i>T. Spider mites T.S.S.M</i>	<i>T. Spider M.T.S.S.M</i> (99.48%)	Benar
63	513	<i>T. Spider mites T.S.S.M</i>	<i>T. Spider M.T.S.S.M</i> (99.98%)	Benar
64	536	<i>T. Spider mites T.S.S.M</i>	<i>T. Spider M.T.S.S.M</i> (99.28%)	Benar
65	523	<i>T. Spider mites T.S.S.M</i>	<i>T. Spider M.T.S.S.M</i> (99.99%)	Benar
66	577	<i>T. Spider mites T.S.S.M</i>	<i>T. Spider M.T.S.S.M</i> (99.61%)	Benar
67	572	<i>T. Spider mites T.S.S.M</i>	<i>T. Spider M.T.S.S.M</i> (99.84%)	Benar
68	648	<i>T. Spider mites T.S.S.M</i>	<i>T. Spider M.T.S.S.M</i> (99.99%)	Benar
69	495	<i>T. Spider mites T.S.S.M</i>	<i>T. Spider M.T.S.S.M</i> (99.85%)	Benar
70	530	<i>T. Spider mites T.S.S.M</i>	<i>T. Spider M.T.S.S.M</i> (99.66%)	Benar
71	1104	<i>T. Target Spot</i>	<i>T. Target Spot</i> (98.52%)	Benar
72	518	<i>T. Target Spot</i>	<i>T. Target Spot</i> (99.57%)	Benar
73	548	<i>T. Target Spot</i>	<i>T. Target Spot</i> (99.58%)	Benar
74	534	<i>T. Target Spot</i>	<i>T. Target Spot</i> (99.81%)	Benar
75	528	<i>T. Target Spot</i>	<i>T. Target Spot</i> (99.07%)	Benar
76	523	<i>T. Target Spot</i>	<i>T. Target Spot</i> (78.94%)	Benar
77	497	<i>T. Target Spot</i>	<i>T. Target Spot</i> (99.94%)	Benar
78	514	<i>T. Target Spot</i>	<i>T. Target Spot</i> (99.05%)	Benar
79	558	<i>T. Target Spot</i>	<i>T. Target Spot</i> (99.21%)	Benar
80	497	<i>T. Target Spot</i>	<i>T. Target Spot</i> (98.47%)	Benar

81	526	<i>T. Mosaic virus</i>	<i>T. Mosaic virus (99.88%)</i>	Benar
82	561	<i>T. Mosaic virus</i>	<i>T. Mosaic virus (99.59%)</i>	Benar
83	523	<i>T. Mosaic virus</i>	<i>T. Early Blight (89.65%)</i>	Salah
84	405	<i>T. Mosaic virus</i>	<i>T. Spider M.T.S.S.M (48.77%)</i>	Salah
85	553	<i>T. Mosaic virus</i>	<i>T. Mosaic virus (99.69%)</i>	Benar
86	2020	<i>T. Mosaic virus</i>	<i>T. Mosaic virus (99.81%)</i>	Benar
87	554	<i>T. Mosaic virus</i>	<i>T. Early Blight (98.23%)</i>	Salah
88	528	<i>T. Mosaic virus</i>	<i>T. Mosaic virus (65.10%)</i>	Benar
89	522	<i>T. Mosaic virus</i>	<i>T. Mosaic virus (94.83%)</i>	Benar
90	575	<i>T. Mosaic virus</i>	<i>T. Mosaic virus (98.17%)</i>	Benar
91	510	<i>T. Yellow Leaf C.V.</i>	<i>T. Yellow Leaf C.V. (99.97%)</i>	Benar
92	499	<i>T. Yellow Leaf C.V.</i>	<i>T. Yellow Leaf C.V. (99.94%)</i>	Benar
93	501	<i>T. Yellow Leaf C.V.</i>	<i>T. Yellow Leaf C.V. (99.99%)</i>	Benar
94	513	<i>T. Yellow Leaf C.V.</i>	<i>T. Yellow Leaf C.V. (100%)</i>	Benar
95	556	<i>T. Yellow Leaf C.V.</i>	<i>T. Yellow Leaf C.V. (99.94%)</i>	Benar
96	506	<i>T. Yellow Leaf C.V.</i>	<i>T. Yellow Leaf C.V. (100%)</i>	Benar
97	538	<i>T. Yellow Leaf C.V.</i>	<i>T. Yellow Leaf C.V. (99.99%)</i>	Benar
98	560	<i>T. Yellow Leaf C.V.</i>	<i>T. Yellow Leaf C.V. (100%)</i>	Benar
99	534	<i>T. Yellow Leaf C.V.</i>	<i>T. Yellow Leaf C.V. (99.99%)</i>	Benar
100	503	<i>T. Yellow Leaf C.V.</i>	<i>T. Yellow Leaf C.V. (99.98%)</i>	Benar
101	532	<i>CBB</i>	<i>CBB (83.38%)</i>	Benar
102	525	<i>CBB</i>	<i>CBB (75.01%)</i>	Benar
103	536	<i>CBB</i>	<i>CBB (72.02%)</i>	Benar
104	501	<i>CBB</i>	<i>CBB (68.73%)</i>	Benar
105	546	<i>CBB</i>	<i>CBB (60.87%)</i>	Benar
106	470	<i>CBB</i>	<i>CBB (81.32%)</i>	Benar
107	357	<i>CBB</i>	<i>CBB (77.71%)</i>	Benar
108	572	<i>CBB</i>	<i>CBB (75.81%)</i>	Benar
109	542	<i>CBB</i>	<i>CBB (73.07%)</i>	Benar
110	532	<i>CBB</i>	<i>CBB (71.51%)</i>	Benar
111	2117	<i>CBSD</i>	<i>CBSD (79.62%)</i>	Benar
112	518	<i>CBSD</i>	<i>CBSD (96.20%)</i>	Benar
113	536	<i>CBSD</i>	<i>CBSD (91.21%)</i>	Benar
114	504	<i>CBSD</i>	<i>CBSD (72.82%)</i>	Benar
115	563	<i>CBSD</i>	<i>CBSD (56.35%)</i>	Benar
116	501	<i>CBSD</i>	<i>CBSD (92.78%)</i>	Benar
117	525	<i>CBSD</i>	<i>CBSD (80.37%)</i>	Benar
118	507	<i>CBSD</i>	<i>CBSD (56.00%)</i>	Benar
119	561	<i>CBSD</i>	<i>CBSD (92.78%)</i>	Benar
120	566	<i>CBSD</i>	<i>CBSD (65.75%)</i>	Benar
121	2161	<i>CGM</i>	<i>CGM (89.30%)</i>	Benar
122	518	<i>CGM</i>	<i>CGM (90.06%)</i>	Benar
123	519	<i>CGM</i>	<i>CMD (97.90%)</i>	Salah
124	523	<i>CGM</i>	<i>CGM (62.19%)</i>	Benar

<b>125</b>	458	<i>CGM</i>	<i>CGM (45.74%)</i>	Benar
<b>126</b>	517	<i>CGM</i>	<i>CGM (55.14%)</i>	Benar
<b>127</b>	577	<i>CGM</i>	<i>CGM (63.23%)</i>	Benar
<b>128</b>	553	<i>CGM</i>	<i>CGM (86.61%)</i>	Benar
<b>129</b>	502	<i>CGM</i>	<i>CGM (57.59%)</i>	Benar
<b>130</b>	489	<i>CGM</i>	<i>CMD (55.40%)</i>	Salah
<b>131</b>	505	<i>CMD</i>	<i>CGM (54.95%)</i>	Salah
<b>132</b>	444	<i>CMD</i>	<i>CMD (98.68%)</i>	Benar
<b>133</b>	548	<i>CMD</i>	<i>CMD (99.69%)</i>	Benar
<b>134</b>	510	<i>CMD</i>	<i>CMD (85.03%)</i>	Benar
<b>135</b>	513	<i>CMD</i>	<i>CMD (97.50%)</i>	Benar
<b>136</b>	521	<i>CMD</i>	<i>CMD (98.41%)</i>	Benar
<b>137</b>	581	<i>CMD</i>	<i>CMD (97.85%)</i>	Benar
<b>138</b>	531	<i>CMD</i>	<i>CMD (91.78%)</i>	Benar
<b>139</b>	553	<i>CMD</i>	<i>CMD (96.96%)</i>	Benar
<b>140</b>	2161	<i>CMD</i>	<i>CMD (85.87%)</i>	Benar
<b>141</b>	1662	<i>Healthy</i>	<i>Healthy (72.57%)</i>	Benar
<b>142</b>	468	<i>Healthy</i>	<i>Healthy (80.03%)</i>	Benar
<b>143</b>	501	<i>Healthy</i>	<i>Healthy (84.28%)</i>	Benar
<b>144</b>	502	<i>Healthy</i>	<i>Healthy (65.02%)</i>	Benar
<b>145</b>	528	<i>Healthy</i>	<i>Healthy (71.38%)</i>	Benar
<b>146</b>	406	<i>Healthy</i>	<i>Healthy (53.98%)</i>	Benar
<b>147</b>	521	<i>Healthy</i>	<i>Healthy (74.22%)</i>	Benar
<b>148</b>	537	<i>Healthy</i>	<i>Healthy (82.52%)</i>	Benar
<b>149</b>	552	<i>Healthy</i>	<i>Healthy (63.94%)</i>	Benar
<b>150</b>	490	<i>Healthy</i>	<i>Healthy (63.49%)</i>	Benar

Dari tabel pengujian tersebut dapat dilihat bahwa hasil prediksi *relative* bagus, hanya beberapa kesalahan prediksi yang terjadi pada beberapa penyakit. Hal ini dikarenakan jumlah *dataset* yang tidak seimbang dan penggunaan arsitektur, sehingga mempengaruhi proses pelatihan dan validasi. Berdasarkan tabel 4.4 dapat dihitung nilai akurasi dan *loss* dengan persamaan berikut.

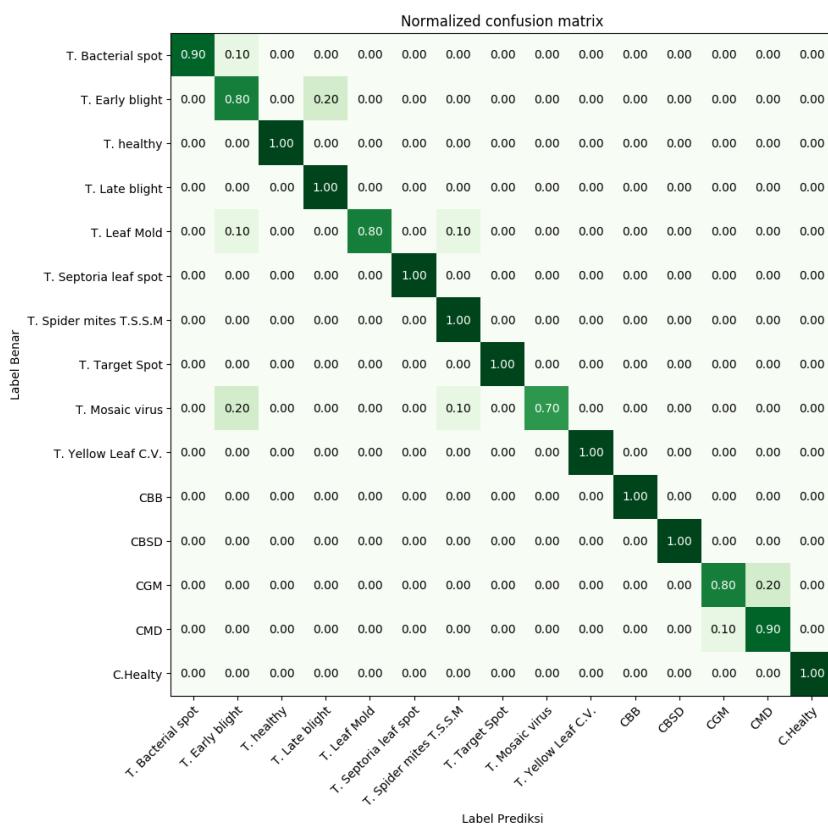
$$\% \text{akurasi} = \frac{\text{jumlahprediksibenar}}{\text{jumlahdatapengujian}} \times 100\%$$

$$\% \text{akurasi} = \frac{139}{150} \times 100\% = 92,67\%$$

$$\% \text{kesalahan} = \frac{\text{jumlah prediksi salah}}{\text{jumlah data pengujian}} \times 100\%$$

$$\% \text{kesalahan} = \frac{11}{150} \times 100\% = 7,33\%$$

Dari hasil perhitungan tersebut diperoleh persentase akurasi yang baik. Hal tersebut menandakan bahwa jaringan CNN dengan arsitektur *InceptionV3* mampu melakukan klasifikasi dengan baik. Untuk mengetahui tingkat akurasi prediksi pada setiap penyakit tanaman dapat dilihat pada *confusion matrix* yang disajikan pada gambar 4.11 berikut.



*Gambar 4.11 Confusion Matrix Pengujian Pada Android Menggunakan Arsitektur InceptionV3*

Berdasarkan *Confusion Matrix* diatas probabilitas akurasi yang didapat pada setiap penyakit tanaman memiliki nilai yang baik. Hal tersebut menandakan penggunaan arsitektur *InceptionV3* dapat diaplikasikan ke android.

#### **4.3.3 Pengujian Android Menggunakan *MobileNet* Versi 1**

Pengujian dilakukan dengan memasukan gambar dengan cara memotret langsung atau melalui *import* galeri pada aplikasi *Plant Diseases*. Data yang digunakan sebanyak 10 pada setiap penyakit. Kemudian gambar yang telah dimasukan kedalam aplikasi akan dilakukan identifikasi sehingga dapat di prediksi keadaan daun tomat atau daun singkong dalam kedaan sehat atau berpenyakit. Pada aplikasi terdapat nilai propabilitas yang menunjukkan nilai presentase kebenaran berdasarkan citra yang diperoleh dari gambar tersebut. Berdasarkan hal tersebut maka diperoleh seperti pada tabel 4.5.

*Tabel 4.5 Data Uji Android Menggunakan Arsitektur MobileNetV1*

<b>Uji ke-</b>	<b>Respon Deteksi (ms)</b>	<b>Data uji</b>	<b>Hasil Prediksi (Tingkat Probabilitas)</b>	<b>Benar/Salah</b>
1	99	<i>T. Bacterial spot</i>	<i>T. Bacterial spot (99.99%)</i>	Benar
2	204	<i>T. Bacterial spot</i>	<i>T. Bacterial spot (99.88%)</i>	Benar
3	160	<i>T. Bacterial spot</i>	<i>T. Bacterial spot (99.99%)</i>	Benar
4	104	<i>T. Bacterial spot</i>	<i>T. Bacterial spot (99.98%)</i>	Benar
5	417	<i>T. Bacterial spot</i>	<i>T. Bacterial spot (99.99%)</i>	Benar
6	123	<i>T. Bacterial spot</i>	<i>T. Bacterial spot (99.99%)</i>	Benar
7	97	<i>T. Bacterial spot</i>	<i>T. Bacterial spot (99.99%)</i>	Benar
8	117	<i>T. Bacterial spot</i>	<i>T. Bacterial spot (99.99%)</i>	Benar
9	179	<i>T. Bacterial spot</i>	<i>T. Bacterial spot (99.99%)</i>	Benar
10	328	<i>T. Bacterial spot</i>	<i>T. Bacterial spot (99.81%)</i>	Benar
11	427	<i>T. Early blight</i>	<i>T. Early blight (99.98%)</i>	Benar
12	147	<i>T. Early blight</i>	<i>T. Early blight (95.79%)</i>	Benar
13	120	<i>T. Early blight</i>	<i>T. Early blight (99.96%)</i>	Benar
14	106	<i>T. Early blight</i>	<i>T. Early blight (96.88%)</i>	Benar
15	159	<i>T. Early blight</i>	<i>T. Early blight (99.89%)</i>	Benar

<b>16</b>	98	<i>T. Early blight</i>	<i>T. Early blight (99.98%)</i>	Benar
<b>17</b>	176	<i>T. Early blight</i>	<i>T. Early blight (99.78%)</i>	Benar
<b>18</b>	110	<i>T. Early blight</i>	<i>T. Early blight (99.98%)</i>	Benar
<b>19</b>	152	<i>T. Early blight</i>	<i>T. Early blight (99.99%)</i>	Benar
<b>20</b>	141	<i>T. Early blight</i>	<i>T. Early blight (80.08%)</i>	Benar
<b>21</b>	135	<i>T. healthy</i>	<i>T. healthy (99.99%)</i>	Benar
<b>22</b>	76	<i>T. healthy</i>	<i>T. healthy (99.99%)</i>	Benar
<b>23</b>	84	<i>T. healthy</i>	<i>T. healthy (99.99%)</i>	Benar
<b>24</b>	102	<i>T. healthy</i>	<i>T. healthy (99.99%)</i>	Benar
<b>25</b>	150	<i>T. healthy</i>	<i>T. healthy (99.99%)</i>	Benar
<b>26</b>	94	<i>T. healthy</i>	<i>T. healthy (99.99%)</i>	Benar
<b>27</b>	130	<i>T. healthy</i>	<i>T. healthy (99.99%)</i>	Benar
<b>28</b>	89	<i>T. healthy</i>	<i>T. healthy (99.99%)</i>	Benar
<b>29</b>	136	<i>T. healthy</i>	<i>T. healthy (99.98%)</i>	Benar
<b>30</b>	146	<i>T. healthy</i>	<i>T. healthy (99.99%)</i>	Benar
<b>31</b>	214	<i>T. Late blight</i>	<i>T. Late blight (99.99%)</i>	Benar
<b>32</b>	126	<i>T. Late blight</i>	<i>T. Late blight (99.99%)</i>	Benar
<b>33</b>	115	<i>T. Late blight</i>	<i>T. Late blight (99.99%)</i>	Benar
<b>34</b>	155	<i>T. Late blight</i>	<i>T. Late blight (99.67%)</i>	Benar
<b>35</b>	127	<i>T. Late blight</i>	<i>T. Late blight (99.77%)</i>	Benar
<b>36</b>	112	<i>T. Late blight</i>	<i>T. Late blight (99.99%)</i>	Benar
<b>37</b>	126	<i>T. Late blight</i>	<i>T. Late blight (99.99%)</i>	Benar
<b>38</b>	118	<i>T. Late blight</i>	<i>T. Late blight (99.99%)</i>	Benar
<b>39</b>	139	<i>T. Late blight</i>	<i>T. Late blight (99.99%)</i>	Benar
<b>40</b>	163	<i>T. Late blight</i>	<i>T. Late blight (99.99%)</i>	Benar
<b>41</b>	110	<i>T. Leaf Mold</i>	<i>T. Leaf Mold (99.59%)</i>	Benar
<b>42</b>	116	<i>T. Leaf Mold</i>	<i>T. Spider M.T.S.S.M (57.73%)</i>	Salah
<b>43</b>	128	<i>T. Leaf Mold</i>	<i>T. Leaf Mold (99.66%)</i>	Benar
<b>44</b>	100	<i>T. Leaf Mold</i>	<i>T. Leaf Mold (98.17%)</i>	Benar
<b>45</b>	160	<i>T. Leaf Mold</i>	<i>T. Leaf Mold (99.98%)</i>	Benar
<b>46</b>	126	<i>T. Leaf Mold</i>	<i>T. Leaf Mold (99.99%)</i>	Benar
<b>47</b>	120	<i>T. Leaf Mold</i>	<i>T. Leaf Mold (100%)</i>	Benar
<b>48</b>	124	<i>T. Leaf Mold</i>	<i>T. Leaf Mold (99.99%)</i>	Benar
<b>49</b>	115	<i>T. Leaf Mold</i>	<i>T. Leaf Mold (99.87%)</i>	Benar
<b>50</b>	129	<i>T. Leaf Mold</i>	<i>T. Leaf Mold (99.90%)</i>	Benar
<b>51</b>	103	<i>T. Septoria leaf spot</i>	<i>T. Septoria leaf spot (99.80%)</i>	Benar
<b>52</b>	82	<i>T. Septoria leaf spot</i>	<i>T. Septoria leaf spot (99.99%)</i>	Benar
<b>53</b>	149	<i>T. Septoria leaf spot</i>	<i>T. Septoria leaf spot (99.99%)</i>	Benar
<b>54</b>	131	<i>T. Septoria leaf spot</i>	<i>T. Septoria leaf spot (99.99%)</i>	Benar
<b>55</b>	152	<i>T. Septoria leaf spot</i>	<i>T. Septoria leaf spot (99.99%)</i>	Benar
<b>56</b>	106	<i>T. Septoria leaf spot</i>	<i>T. Septoria leaf spot (99.99%)</i>	Benar
<b>57</b>	86	<i>T. Septoria leaf spot</i>	<i>T. Septoria leaf spot (99.99%)</i>	Benar
<b>58</b>	91	<i>T. Septoria leaf spot</i>	<i>T. Septoria leaf spot (99.99%)</i>	Benar
<b>59</b>	108	<i>T. Septoria leaf spot</i>	<i>T. Septoria leaf spot (99.99%)</i>	Benar

<b>60</b>	114	<i>T. Septoria leaf spot</i>	<i>T. Septoria leaf spot (99.99%)</i>	Benar
<b>61</b>	110	<i>T. Spider mites T.S.S.M</i>	<i>T. Spider M.T.S.S.M (99.84%)</i>	Benar
<b>62</b>	82	<i>T. Spider mites T.S.S.M</i>	<i>T. Spider M.T.S.S.M (99.21%)</i>	Benar
<b>63</b>	190	<i>T. Spider mites T.S.S.M</i>	<i>T. Spider M.T.S.S.M (99.99%)</i>	Benar
<b>64</b>	97	<i>T. Spider mites T.S.S.M</i>	<i>T. Spider M.T.S.S.M (99.24%)</i>	Benar
<b>65</b>	129	<i>T. Spider mites T.S.S.M</i>	<i>T. Spider M.T.S.S.M (99.95%)</i>	Benar
<b>66</b>	128	<i>T. Spider mites T.S.S.M</i>	<i>T. Spider M.T.S.S.M (99.99%)</i>	Benar
<b>67</b>	115	<i>T. Spider mites T.S.S.M</i>	<i>T. Spider M.T.S.S.M (99.99%)</i>	Benar
<b>68</b>	147	<i>T. Spider mites T.S.S.M</i>	<i>T. Spider M.T.S.S.M (99.99%)</i>	Benar
<b>69</b>	126	<i>T. Spider mites T.S.S.M</i>	<i>T. Spider M.T.S.S.M (99.95%)</i>	Benar
<b>70</b>	119	<i>T. Spider mites T.S.S.M</i>	<i>T. Spider M.T.S.S.M (99.99%)</i>	Benar
<b>71</b>	85	<i>T. Target Spot</i>	<i>T. Target Spot (98.26%)</i>	Benar
<b>72</b>	140	<i>T. Target Spot</i>	<i>T. Target Spot (99.73%)</i>	Benar
<b>73</b>	165	<i>T. Target Spot</i>	<i>T. Target Spot (99.97%)</i>	Benar
<b>74</b>	81	<i>T. Target Spot</i>	<i>T. Target Spot (99.99%)</i>	Benar
<b>75</b>	146	<i>T. Target Spot</i>	<i>T. Target Spot (99.99%)</i>	Benar
<b>76</b>	160	<i>T. Target Spot</i>	<i>T. Target Spot (78.97%)</i>	Benar
<b>77</b>	133	<i>T. Target Spot</i>	<i>T. Target Spot (99.84%)</i>	Benar
<b>78</b>	142	<i>T. Target Spot</i>	<i>T. Target Spot (99.99%)</i>	Benar
<b>79</b>	88	<i>T. Target Spot</i>	<i>T. Target Spot (99.99%)</i>	Benar
<b>80</b>	171	<i>T. Target Spot</i>	<i>T. Target Spot (99.99%)</i>	Benar
<b>81</b>	143	<i>T. Mosaic virus</i>	<i>T. Mosaic virus (99.99%)</i>	Benar
<b>82</b>	153	<i>T. Mosaic virus</i>	<i>T. Mosaic virus (99.99%)</i>	Benar
<b>83</b>	80	<i>T. Mosaic virus</i>	<i>T. Mosaic virus (99.99%)</i>	Benar
<b>84</b>	136	<i>T. Mosaic virus</i>	<i>T. Mosaic virus (99.62%)</i>	Benar
<b>85</b>	96	<i>T. Mosaic virus</i>	<i>T. Mosaic virus (99.99%)</i>	Benar
<b>86</b>	87	<i>T. Mosaic virus</i>	<i>T. Mosaic virus (99.99%)</i>	Benar
<b>87</b>	91	<i>T. Mosaic virus</i>	<i>T. Mosaic virus (99.99%)</i>	Benar
<b>88</b>	116	<i>T. Mosaic virus</i>	<i>T. Mosaic virus (99.98%)</i>	Benar
<b>89</b>	86	<i>T. Mosaic virus</i>	<i>T. Mosaic virus (99.99%)</i>	Benar
<b>90</b>	153	<i>T. Mosaic virus</i>	<i>T. Mosaic virus (99.99%)</i>	Benar
<b>91</b>	116	<i>T. Yellow Leaf C.V.</i>	<i>T. Yellow Leaf C.V. (100%)</i>	Benar
<b>92</b>	134	<i>T. Yellow Leaf C.V.</i>	<i>T. Yellow Leaf C.V. (100%)</i>	Benar
<b>93</b>	88	<i>T. Yellow Leaf C.V.</i>	<i>T. Yellow Leaf C.V. (100%)</i>	Benar
<b>94</b>	150	<i>T. Yellow Leaf C.V.</i>	<i>T. Yellow Leaf C.V. (100%)</i>	Benar
<b>95</b>	126	<i>T. Yellow Leaf C.V.</i>	<i>T. Yellow Leaf C.V. (100%)</i>	Benar
<b>96</b>	127	<i>T. Yellow Leaf C.V.</i>	<i>T. Yellow Leaf C.V. (100%)</i>	Benar
<b>97</b>	140	<i>T. Yellow Leaf C.V.</i>	<i>T. Yellow Leaf C.V. (100%)</i>	Benar
<b>98</b>	104	<i>T. Yellow Leaf C.V.</i>	<i>T. Yellow Leaf C.V. (100%)</i>	Benar
<b>99</b>	178	<i>T. Yellow Leaf C.V.</i>	<i>T. Yellow Leaf C.V. (100%)</i>	Benar
<b>100</b>	92	<i>T. Yellow Leaf C.V.</i>	<i>T. Yellow Leaf C.V. (100%)</i>	Benar
<b>101</b>	142	<i>CBB</i>	<i>CBB (81.79%)</i>	Benar
<b>102</b>	377	<i>CBB</i>	<i>CBB (80.47%)</i>	Benar
<b>103</b>	123	<i>CBB</i>	<i>CBB (82.41%)</i>	Benar

<b>104</b>	122	<i>CBB</i>	<i>CBB (78.40%)</i>	Benar
<b>105</b>	94	<i>CBB</i>	<i>CGM (67.13%)</i>	Salah
<b>106</b>	90	<i>CBB</i>	<i>CBB (70.14%)</i>	Benar
<b>107</b>	120	<i>CBB</i>	<i>CBB (79.48%)</i>	Benar
<b>108</b>	108	<i>CBB</i>	<i>CBB (78.36%)</i>	Benar
<b>109</b>	115	<i>CBB</i>	<i>CBSD (60.88%)</i>	Salah
<b>110</b>	141	<i>CBB</i>	<i>CBB (76.90%)</i>	Benar
<b>111</b>	121	<i>CBSD</i>	<i>CBSD (97.19%)</i>	Benar
<b>112</b>	142	<i>CBSD</i>	<i>CBSD (87.59%)</i>	Benar
<b>113</b>	123	<i>CBSD</i>	<i>CBSD (97.15%)</i>	Benar
<b>114</b>	126	<i>CBSD</i>	<i>CBSD (97.76%)</i>	Benar
<b>115</b>	162	<i>CBSD</i>	<i>CBSD (90.79%)</i>	Benar
<b>116</b>	93	<i>CBSD</i>	<i>CBSD (91.81%)</i>	Benar
<b>117</b>	97	<i>CBSD</i>	<i>CBSD (79.64%)</i>	Benar
<b>118</b>	208	<i>CBSD</i>	<i>CBSD (97.06%)</i>	Benar
<b>119</b>	88	<i>CBSD</i>	<i>CBSD (96.14%)</i>	Benar
<b>120</b>	94	<i>CBSD</i>	<i>CGM (56.45%)</i>	Salah
<b>121</b>	110	<i>CGM</i>	<i>CMD (52.75%)</i>	Salah
<b>122</b>	139	<i>CGM</i>	<i>CGM (71.09%)</i>	Benar
<b>123</b>	166	<i>CGM</i>	<i>CGM (90.14%)</i>	Benar
<b>124</b>	109	<i>CGM</i>	<i>CGM (89.77%)</i>	Benar
<b>125</b>	159	<i>CGM</i>	<i>CMD (58.20%)</i>	Benar
<b>126</b>	129	<i>CGM</i>	<i>CGM (87.18%)</i>	Benar
<b>127</b>	223	<i>CGM</i>	<i>CGM (86.70%)</i>	Benar
<b>128</b>	123	<i>CGM</i>	<i>CGM (86.70%)</i>	Benar
<b>129</b>	206	<i>CGM</i>	<i>CGM (74.84%)</i>	Benar
<b>130</b>	111	<i>CGM</i>	<i>CGM (69.05%)</i>	Benar
<b>131</b>	176	<i>CMD</i>	<i>CMD (99.35%)</i>	Benar
<b>132</b>	134	<i>CMD</i>	<i>CMD (96.35%)</i>	Benar
<b>133</b>	165	<i>CMD</i>	<i>CMD (99.31%)</i>	Benar
<b>134</b>	104	<i>CMD</i>	<i>CMD (92.35%)</i>	Benar
<b>135</b>	122	<i>CMD</i>	<i>CMD (99.62%)</i>	Benar
<b>136</b>	120	<i>CMD</i>	<i>CMD (99.36%)</i>	Benar
<b>137</b>	125	<i>CMD</i>	<i>CMD (99.59%)</i>	Benar
<b>138</b>	141	<i>CMD</i>	<i>CMD (99.17%)</i>	Benar
<b>139</b>	147	<i>CMD</i>	<i>CMD (91.05%)</i>	Benar
<b>140</b>	138	<i>CMD</i>	<i>CMD (93.91%)</i>	Benar
<b>141</b>	149	<i>Healty</i>	<i>Healty (74.99%)</i>	Benar
<b>142</b>	112	<i>Healty</i>	<i>Healty (78.03%)</i>	Benar
<b>143</b>	169	<i>Healty</i>	<i>CGM (54.24%)</i>	Salah
<b>144</b>	112	<i>Healty</i>	<i>Healty (71.67%)</i>	Benar
<b>145</b>	158	<i>Healty</i>	<i>Healty (73.44%)</i>	Benar
<b>146</b>	163	<i>Healty</i>	<i>CMD (43.97%)</i>	Salah
<b>147</b>	120	<i>Healty</i>	<i>Healty (72.51%)</i>	Benar

<b>148</b>	123	<i>Healty</i>	<i>Healty (77.05%)</i>	Benar
<b>149</b>	133	<i>Healty</i>	<i>Healty (72.94%)</i>	Benar
<b>150</b>	126	<i>Healty</i>	<i>Healty (71.15%)</i>	Benar

Dari tabel pengujian tersebut dapat dilihat bahwa hasil prediksi untuk daun tomat relative bagus hanya ada satu kesalahan prediksi pada penyakit *leaf mold*. Selanjutnya pada daun singkong terdapat beberapa kesalahan yang terjadi. Hal ini dikarenakan jumlah *dataset* yang tidak seimbang. Sehingga mempengaruhi proses pelatihan dan validasi. Berdasarkan tabel 4.5 dapat dihitung nilai akurasi dan *loss* dengan persamaan berikut.

$$\% \text{akurasi} = \frac{\text{jumlahprediksibenar}}{\text{jumlahdatapengujian}} \times 100\%$$

$$\% \text{akurasi} = \frac{143}{150} \times 100\% = 95,33\%$$

$$\% \text{kesalahan} = \frac{\text{jumlahprediksisalah}}{\text{jumlahdatapengujian}} \times 100\%$$

$$\% \text{kesalahan} = \frac{7}{150} \times 100\% = 4,67\%$$

Dari hasil perhitungan tersebut diperoleh persentase akurasi yang baik. Hal tersebut menandakan bahwa jaringan CNN dengan arsitektur *MobileNetVI* mampu melakukan klasifikasi dengan baik. Untuk mengetahui tingkat akurasi prediksi pada setiap penyakit tanaman dapat dilihat pada *confusion matrix* yang disajikan pada gambar 4.11 berikut.

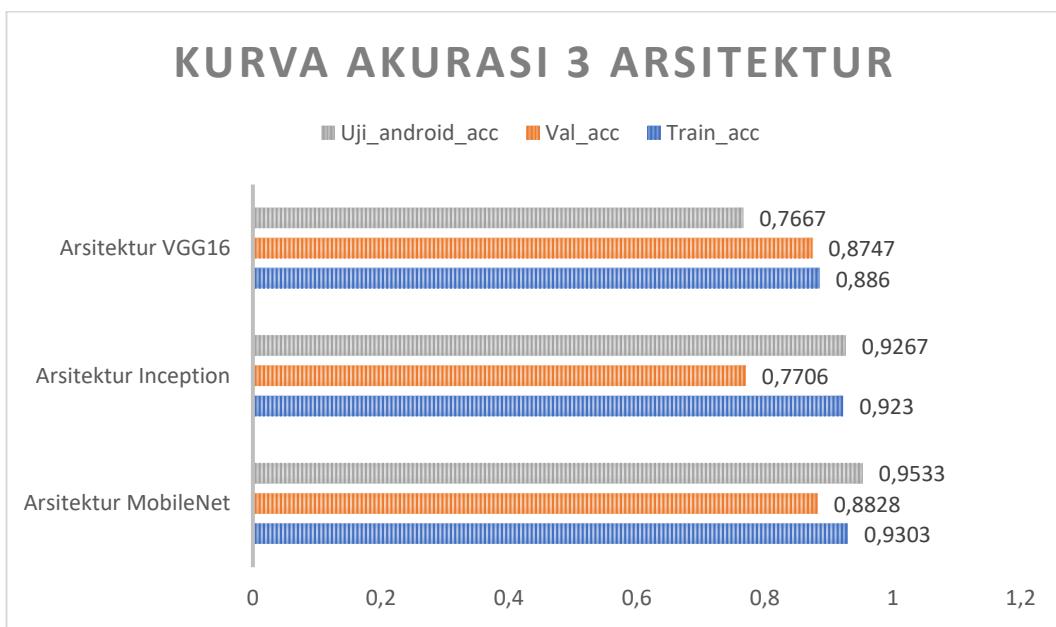
Normalized confusion matrix															
Label Benar	T. Bacterial spot	T. Early blight	T. healthy	T. Late blight	T. Leaf Mold	T. Septoria leaf spot	T. Spider mites T.S.S.M	T. Target Spot	T. Mosaic virus	T. Yellow Leaf C.V.	CBB	CBSD	CGM	CMD	C.Healthy
	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
	0.00	0.00	0.00	0.00	0.90	0.00	0.10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.80	0.10	0.10	0.00	
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.90	0.10	0.00	
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.90	0.10	
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.10	0.10	0.80	

Gambar 4.12 Confusion Matrix Pengujian Pada Android Menggunakan Arsitektur MobileNetV1

Berdasarkan Confusion Matrix diatas probabilitas akurasi yang didapat pada setiap penyakit tanaman memiliki nilai yang baik. Hal tersebut menandakan penggunaan arsitektur MobileNetV1 dapat diaplikasikan ke android.

#### 4.4 Perbandingan Tiga Arsitektur

Berdasarkan dari hasil penelitian ketiga arsitektur yang digunakan memiliki perbedaan masing-masing. Adapun perbedaannya sebagai berikut.

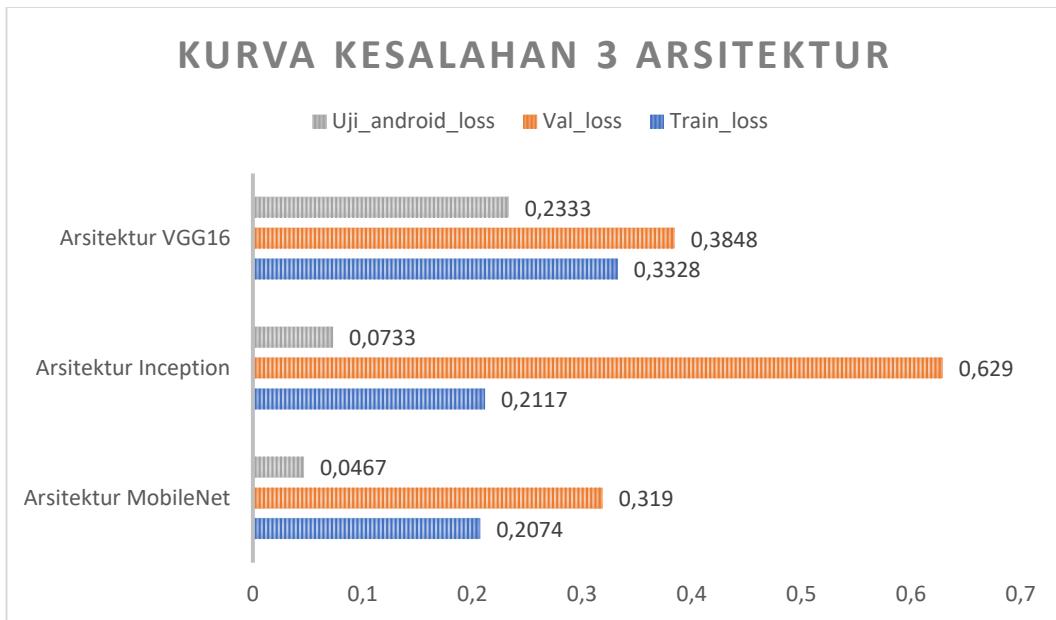


*Gambar 4.13 Kurva Akurasi Perbandingan Arsitektur VGG16, InceptionV3 dan MobileNetV1*

Kurva diatas menunjukan perbandingan nilai akurasi antara tiga arsitektur yaitu VGG16, *InceptionV3* dan *MobilNetV1*. Dimana ketiganya memiliki perbedaan pada nilai akurasinya. Pada pengujian android *MobileNetV1* lebih unggul daripada *Inceptionv3* dan VGG16 dengan akurasi untuk *MobileNetV1* yaitu sebesar 0.9533, sedangkan pada arsitektur *InceptionV3* sebesar 0.927 dan pada arsitektur VGG16 memiliki nilai akurasi sebesar 0.7667.

Pada proses *training* dan *validation* ketiganya memiliki nilai akurasi dan kesalahan yang hampir sama. Nilai akurasi *training* dan *validation* pada arsitektur VGG16 adalah 0.8860 dan 0.8747. Nilai akurasi *training* dan *validation* pada arsitektur *InceptionV3* adalah 0.9230 dan 0.7706, Nilai akurasi pada arsitektur *MobileNetV1* pada proses *training* dan *validation* adalah 0.933 dan 0.8828.

Pada ketiga arsitektur yang digunakan memiliki kekurangan yaitu berupa nilai kesalahan pada masing-masing proses, baik pada proses *training*, *validation*, dan pengujian pada android. Berikut kurva kesalahan yang terjadi pada tiga arsitektur ini.



Gambar 4.14 Kurva kesalahan Perbandingan Arsitektur VGG16, InceptionV3 dan MobileNetV1

Pada gambar 4.14 diatas menunjukan kurva kesalahan tiga arsitektur (VGG16, *InceptionV3*, dan *MobileNetV1*) yang terjadi pada proses *training*, *validation*, dan pengujian android. Pada pengujian android nilai kesalahan terkecil terdapat pada arsitektur *MobilenetV1* yaitu sebesar 0.0467, selanjutnya pada arsitektur *InceptionV3* yaitu sebesar 0.0733, dan terakhir pada arsitektur VGG16 yaitu sebesar 0.233.

Kemudian pada proses *training* dan *validation*, arsitektur VGG16 memiliki nilai kesalahan yaitu sebesar 0.3328 dan 0.3848. pada arsitektur *InceptionV3*

memiliki nilai kesalahan sebesar 0.2117 dan 0.629. Pada Arsitektur *MobilenetV1* memiliki nilai kesalahan sebesar 0.2074 dan 0.319.

## **BAB 5**

### **PENUTUP**

#### **5.1 Kesimpulan**

Berdasarkan hasil penelitian yang telah dilakukan maka diperoleh kesimpulannya sebagai berikut

1. *Deep learning* dapat mengklasifikasikan sesuatu secara otomatis. Sehingga tidak harus memberikan fitur-fitur yang diperlukan untuk mengenali sesuatu tersebut.
2. Jumlah dan kualitas *dataset* memberi pengaruh besar terhadap proses *training* dan *validation*. Jumlah *dataset* yang tidak seimbang pada setiap jenis klasifikasi dapat menjadikan proses *training* dan *validation* tidak stabil.
3. Arsitektur VGG16 memiliki kurva *training* dan *validation* yang lebih baik dibandingkan arsitektur *MobileNet* dan *Inception*.
4. Identifikasi penyakit tanaman menggunakan *goolge colaboratory* untuk melakukan proses *training* dan *validation*, lalu kemudian mengkonversi dalam bentuk *tensorflow lite* agar bisa diaplikasikan pada android.
5. Pengujian identifikasi penyakit tanaman dilakukan pada aplikasi *Plant Diseases* yang terinstal pada *smartphone*, dengan cara memotret langsung obyek atau dengan cara *import Gallery*.
6. Arsitektur *MobileNet* memiliki nilai akurasi lebih tinggi yaitu sebesar 95,33% dibandingkan dengan arsitektur *Inception* (92,67%) dan VGG16 (76,67%) saat pengujian melalui aplikasi android.

7. Respon deteksi *MobileNet* lebih cepat dibandingkan dengan respon deteksi arsitektur lain. Dapat dilihat pada perhitungan nilai rata-rata respon deteksi yang dibutuhkan pada *MobileNet* yaitu sebesar 134,66, sedangkan pada arsitektur *Inception* dan VGG16 masing masing sebesar 643,6 dan 789,03.

## 5.2 Saran

Adapun beberapa hal yang disarankan untuk pengembangan penelitian ini adalah sebagai berikut.

1. Menambah jumlah *dataset* sehingga jumlah *dataset* pada setiap jenis penyakit tanaman tidak jauh berbeda.
2. Mengembangkan arsitektur agar memperoleh nilai akurasi yang baik dan mengurangi jumlah *loss* pada proses *training* maupun *validation*.
3. Melakukan modifikasi untuk klasifikasi obyek lain agar penggunaan metode *Deep learning* dapat diaplikasikan lebih luas lagi.

## DAFTAR PUSTAKA

- [1] Apriyantono, A., D. Fardiaz, N. L. Puspitasari, Sedamawati dan S. Budiyanto, 1989. Analisis Pangan. PAU Pangan dan Gizi. IPB Press.
- [2] Departemen Kesehatan RI. 2003. Pedoman Penanggulangan Masalah Gizi dalam Keadaan Darurat. Direktorat Bina Gizi Masyarakat.
- [3] Badan Pusat Statistik Jakarta Pusat , 2012. Statistik Indonesia Tahun 2012. Jakarta Pusat : Badan Pusat Statistik.
- [4] J. Pertanian Agros. “*Penyakit Tanaman*”. Universitas Janabadra Yogyakarta.
- [5] Chollet, François. 2018. *Deep Learning with Python*. Shelter Island: Manning Publications Co.
- [6] Sena, Samuel. 2017. “Pengenalan Deep Learning Part 7 : Convolutional Neural Network (CNN)”. [Daring]. Tersedia pada: <https://medium.com/@samuelsena/pengenalan-deep-learning-part-7-convolutional-neural-network-cnn-b003b477dc94>. [Diakses: 10-Nov-2019].
- [7] Munir, Rinaldi. 2004. Pengolahan Citra Digital dengan Pendekatan Algoritmik. Bandung : Informatika
- [8] XenonStack. 2017. “Log Analytics With Deep Learning And Machine Learning”. [Daring]. Tersedia pada <https://medium.com/@xenonstack/log-analytics-with-deep-learning-and-machine-learning-20a1891ff70e> [Diakses Pada : 10-Nov-2019]
- [9] Tandungan, Sofyan. 2019. “Pengenalan Convolutional Neural Network – Part 1”. [daring] tersedia pada: <http://sofyantandungan.com/pengenalan-convolutional-neural-network-part-1/> [Diakses Pada : 10-Nov-2019]
- [10] Neurohive. “*VGG16 – Convolutional Network for Classification and Detection*”. [Daring]. Tersedia pada: <https://neurohive.io/en/popular-networks/vgg16/>. [Diakses: 29-Nov-2019].
- [11] Ekoputris, Rizqi Okta. “*MobileNet: Deteksi Objek pada Platform Mobile*”. [Daring]. Tersedia pada: <https://medium.com/nodeflux/mobilenet-deteksi-objek-pada-platform-mobile-bbbf3806e4b3>. [Diakses: 29-Nov-2019].
- [12] Google Inc. 2017. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. arXiv:1704.04861v1 [cs.CV] 17 Apr 2017.
- [13] K. He, X. Zhang, S. Ren, and J. Sun. 2016. “Deep residual learning for image recognition”. Salt Lake City:IEEE.
- [14] Adam, Rian. 2019. “Mengenal Google Colab”. [Daring]. Tersedia pada <https://structilmy.com/2019/05/mengenal-google-colab/> [Diakses Pada : 17-Nov-2019]
- [15] Dewaweb. 2019. “Keunggulan Memahami Bahasa Pemrograman Python”. [Daring]. Tersedia pada <https://www.dewaweb.com/blog/keunggulan-memahami-bahasa-pemrograman-python/> [Diakses Pada : 17-Nov-2019]
- [16] Tjioe, Enlik. 2019. “Klasifikasi Gambar menggunakan Keras”. [Daring] tersedia pada : <https://rpubs.com/enlik/keras> [Diakses Pada : 17-Nov.2019]
- [17] Moroney, Laurence. 2018. “*Using TensorFlow Lite on Android*”. [Daring] tersedia pada : <https://medium.com/tensorflow/using-tensorflow-lite-on-android-9bbc9cb7d69d> [Diakses Pada 17-Nov-2019]

- [18] Google. “Mengenal Android Studio” . [Daring] tersedia Pada : <https://developer.android.com/studio/intro/?hl=id> [Diakses Pada 17-Nov-2019]
- [19] Anonim. 2016. “Pengertian Penyakit tanaman”. [Daring] tersedia pada : <https://www.sampulpertanian.com/2016/10/pengertian-penyakit-tanaman.html> [Diakses Pada 17-Nov-2019]
- [20] Layanan Informasi Desa. 2018. “Penyakit pada Tanaman Tomat”. [Daring] tersedia pada : <https://8villages.com/full/petani/article/id/5b5af73bec9e5e7b5cd05e28> [Diakses Pada 17-Nov-2019]
- [21] Cle on University. 2018. “*Tomato Diseases & Disorders*”. [Daring] tersedia pada : <https://hgic.cle on.edu/factsheet/tomato-diseases-disorders/> [Diakses Pada 17-Nov-2019]
- [22] Jackson, Grahame. 2010. “*Pacific Pests and Pathogens - Fact Sheets*”. [Daring] tersedia pada : [http://www.pestnet.org/fact\\_sheets/cucumber\\_target\\_spot\\_189.htm](http://www.pestnet.org/fact_sheets/cucumber_target_spot_189.htm) [Diakses pada : 17-Nov-19]
- [23] R. Hazzard, UMass Extension. 2013. “*Two-spotted Spider Mite*”. [Daring] tersedia pada : <https://ag.umass.edu/vegetable/fact-sheets/two-spotted-spider-mite> [diakses pada : 17-Nov-19]
- [24] Badan Penelitian Dan Pengembangan Pertanian. 2013. “Hama, Penyakit, dan Gulma pada Tanaman Ubi Kayu Identifikasi dan Pengendaliannya”. Jakarta : IAARD Press.

## LAMPIRAN

### Lampiran 1 Sourecode Identifikasi Penyakit Tanaman Menggunakan Arsitektur VGG16

```
import numpy as np
import os
import matplotlib.pyplot as plt
plt.style.use('default')
from skimage.io import imread
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import train_test_split
from __future__ import absolute_import, division, print_function, unicode_literals

import tensorflow as tf
#tf.logging.set_verbosity(tf.logging.ERROR)
#tf.enable_eager_execution()

import tensorflow_hub as hub
import os
from tensorflow.keras.layers import Dense, Flatten, Conv2D
from tensorflow.keras import Model
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam
from tensorflow.keras import layers
#from keras import optimizers

import keras
import keras.backend as K
from keras.preprocessing.image import load_img, img_to_array,
ImageDataGenerator
from keras.utils.np_utils import to_categorical

from keras.utils.data_utils import get_file

from keras import layers
from keras.models import Sequential, Model
from keras.callbacks import EarlyStopping, ModelCheckpoint

from sklearn.model_selection import train_test_split

!apt-get install subversion > /dev/null

#Retreive specific diseases of tomato for training
```

```
!svn export https://github.com/spMohanty/PlantVillage-Dataset/
trunk/raw/color/Tomato__Bacterial_spot image/Tomato__Bacteri
al_spot > /dev/null
!svn export https://github.com/spMohanty/PlantVillage-Dataset/
trunk/raw/color/Tomato__Early_blight image/Tomato__Early_bli
ght > /dev/null
!svn export https://github.com/spMohanty/PlantVillage-Dataset/
trunk/raw/color/Tomato__Late_blight image/Tomato__Late_bli
ght > /dev/null
!svn export https://github.com/spMohanty/PlantVillage-Dataset/
trunk/raw/color/Tomato__Septoria_leaf_spot image/Tomato__Sep
toria_leaf_spot > /dev/null
!svn export https://github.com/spMohanty/PlantVillage-Dataset/
trunk/raw/color/Tomato__Target_Spot image/Tomato__Target_Spo
t > /dev/null
!svn export https://github.com/spMohanty/PlantVillage-Dataset/
trunk/raw/color/Tomato__healthy image/Tomato__healthy > /de
v/null
!svn export https://github.com/spMohanty/PlantVillage-Dataset/
trunk/raw/color/Tomato__Tomato_mosaic_virus image/Tomato__mo
saic_virus > /dev/null
!svn export https://github.com/spMohanty/PlantVillage-Dataset/
trunk/raw/color/Tomato__Spider_mites%20Two-spotted_spider_mit
e image/Tomato__Spider_mites_20Two-spotted_spider_mite > /de
v/null
!svn export https://github.com/spMohanty/PlantVillage-Dataset/
trunk/raw/color/Tomato__Leaf_Mold image/Tomato__Leaf_Mold >
/dev/null
!svn export https://github.com/spMohanty/PlantVillage-Dataset/
trunk/raw/color/Tomato__Tomato_Yellow_Leaf_Curl_Virus image/T
omato__Yellow_Leaf_Curl_Virus > /dev/null
!svn export https://github.com/icassava/fgvcx-icassava/trunk/d
ata/train/cbb image/Cassava__cbb > /dev/null
!svn export https://github.com/icassava/fgvcx-icassava/trunk/d
ata/train/cbsd image/Cassava__cbsd > /dev/null
!svn export https://github.com/icassava/fgvcx-icassava/trunk/d
ata/train/cgm image/Cassava__cgm > /dev/null
!svn export https://github.com/icassava/fgvcx-icassava/trunk/d
ata/train/cmd image/Cassava__cmd > /dev/null
!svn export https://github.com/icassava/fgvcx-icassava/trunk/d
ata/train/healthy image/Cassava__healthy > /dev/null
```

```
plt.figure(figsize=(15,10))

#visualize several images

parent_directory = "image"

for i, folder in enumerate(os.listdir(parent_directory)):
    print(folder)
```

```

folder_directory = os.path.join(parent_directory,folder)
files = os.listdir(folder_directory)
#will inspect only 1 image per folder
file = files[0]
file_path = os.path.join(folder_directory,file)

image = imread(file_path)
plt.subplot(1,15,i+1)
plt.imshow(image)
plt.axis("off")

name = folder.split("__") [1] [-1]
plt.title(name)
plt.show()

#load everything into memory
x = []
y = []
class_names = []
parent_directory = "image"

for i, folder in enumerate(os.listdir(parent_directory)):
    print(i, folder)
    class_names.append(folder)
    folder_directory = os.path.join(parent_directory,folder)
    files = os.listdir(folder_directory)
    #will inspect only 1 image per folder
    for file in files:
        file_path = os.path.join(folder_directory,file)
        image = load_img(file_path,target_size=(64,64))
        image = img_to_array(image)/255.
        x.append(image)
        y.append(i)

x = np.array(x)
y = to_categorical(y)

#check the data shape
print(x.shape)
print(y.shape)
print(y[0])

x_train, _x, y_train, _y = train_test_split(x,y,test_size=0.2,
stratify = y, random_state = 1)
x_valid,x_test, y_valid, y_test = train_test_split(_x,_y,test_
size=0.4, stratify = _y, random_state = 1)

print("train data:",x_train.shape,y_train.shape)
print("validation data:",x_valid.shape,y_valid.shape)
print("test data:",x_test.shape,y_test.shape)

import tensorflow as tf

```

```

IMG_SHAPE = (64, 64, 3)
# Membuat model dasar (base model) dari pre-trained model MobileNet
base_model = tf.keras.applications.VGG16(input_shape=IMG_SHAPE,
                                         include_top=False,
                                         weights='imagenet')

base_model.trainable = True
base_model.summary()

model = tf.keras.Sequential([
    base_model,
    tf.keras.layers.Conv2D(16, (3, 3), name="conv1", input_shape=(64, 64, 3), activation="relu", padding="same"),
    tf.keras.layers.MaxPool2D((2, 2), name="pool1", padding="same"),
    #tf.keras.layers.Dropout(0.05),
    tf.keras.layers.Conv2D(32, (3, 3), name="conv2", padding="same"),
    tf.keras.layers.Activation("relu"),
    tf.keras.layers.MaxPool2D((2, 2), name="pool2", padding="same"),
    #tf.keras.layers.Dropout(0.05),
    tf.keras.layers.Conv2D(32, (3, 3), name="conv3", padding="same"),
    tf.keras.layers.Activation("relu"),
    tf.keras.layers.MaxPool2D((2, 2), name="pool3", padding="same"),
    #tf.keras.layers.Dropout(0.05),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(64, activation="relu"),
    #tf.keras.layers.Dropout(0.05),
    tf.keras.layers.Dense(15, activation='softmax')
])

model.compile("adam", loss="categorical_crossentropy", metrics=[ "acc"])
model.summary()

#utilize early stopping function to stop at the lowest validation loss
es = EarlyStopping(monitor='val_loss', patience=10, verbose=1, mode='auto')
#utilize save best weight model during training
ckpt = ModelCheckpoint("DeteksiPenyakitTanaman.hdf5", monitor='val_loss', verbose=1, save_best_only=True, save_weights_only=False, mode='auto', period=1)

```

```

#we will define a generator class for training data and validation data separately, as no augmentation is not required for validation data
t_gen = ImageDataGenerator(rotation_range=90, horizontal_flip=True)
v_gen = ImageDataGenerator()
train_gen = t_gen.flow(x_train,y_train,batch_size=98)
valid_gen = v_gen.flow(x_valid,y_valid,batch_size=98)

history = model.fit_generator(
    train_gen,
    steps_per_epoch = train_gen.n // 98,
    callbacks = [ckpt],
    callbacks = [es, ckpt],
    validation_data = valid_gen,
    validation_steps = valid_gen.n // 98,
    epochs=100 #@param {type:"integer"}
)

plt.figure(figsize=(10, 4))
# plt.subplot(1, 5, 4)
plt.plot(history.history["acc"],label="Akurasi Pelatihan")
plt.plot(history.history["val_acc"],label="Validasi Akurasi")
plt.legend()
plt.title('Training and Validation Accuracy')
plt.ylabel("Accuracy (training and validation)")
plt.xlabel("Training Steps")
plt.show()
plt.figure(figsize=(10, 4))
# plt.subplot(1, 3, 2)
plt.plot(history.history["loss"],label="Kesalahan Pelatihan")
plt.plot(history.history["val_loss"],label="Validasi Kesalahan")
plt.legend()
plt.title('Training and Validation Loss')
plt.ylabel("Loss (training and validation)")
plt.xlabel("Training Steps")
plt.show()

#load the model weight file with lowest validation loss
model.load_weights("DeteksiPenyakitTanaman.hdf5")

#check the model metrics
print(model.metrics_names)
#evaluate training data
print(model.evaluate(x= x_train, y = y_train))
#evaluate validation data
print(model.evaluate(x= x_valid, y = y_valid))

```

```

#evaluate test data
print(model.evaluate(x= x_test, y = y_test))

#true label
y_true = np.argmax(y_test, axis=1)

#prediction label
Y_pred = model.predict(x_test)
y_pred = np.argmax(Y_pred, axis=1)

print(y_true)
print(y_pred)

from sklearn.metrics import confusion_matrix
from sklearn.utils.multiclass import unique_labels

def plot_confusion_matrix(y_true, y_pred, classes,
                           normalize=False,
                           title=None,
                           cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    if not title:
        if normalize:
            title = 'Normalized confusion matrix'
        else:
            title = 'Confusion matrix, without normalization'

    # Compute confusion matrix
    cm = confusion_matrix(y_true, y_pred)
    # Only use the labels that appear in the data
    #classes = classes[unique_labels(y_true, y_pred)]
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
    print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)

    fig, ax = plt.subplots(figsize=(9,9))
    im = ax.imshow(cm, interpolation='nearest', cmap=cmap)
    #ax.figure.colorbar(im, ax=ax)
    # We want to show all ticks...
    ax.set(xticks=np.arange(cm.shape[1]),
           yticks=np.arange(cm.shape[0]),
           # ... and label them with the respective list entries
           xticklabels=classes, yticklabels=classes,

```

```

        title=title,
        ylabel='True label',
        xlabel='Predicted label')

    # Rotate the tick labels and set their alignment.
    plt.setp(ax.get_xticklabels(), rotation=45, ha="right",
              rotation_mode="anchor")

    # Loop over data dimensions and create text annotations.
    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i in range(cm.shape[0]):
        for j in range(cm.shape[1]):
            ax.text(j, i, format(cm[i, j], fmt),
                    ha="center", va="center",
                    color="white" if cm[i, j] > thresh else "black")
    fig.tight_layout()
    return ax

np.set_printoptions(precision=2)

plot_confusion_matrix(y_true, y_pred, classes=class_names, normalize=True,
                      title='Normalized confusion matrix')

keras_file = "DeteksiPenyakitTanaman.hdf5"
tf.keras.models.save_model(model, keras_file)

# Convert to TensorFlow Lite model.
converter = tf.lite.TFLiteConverter.from_keras_model_file(keras_file)
tflite_model = converter.convert()
open("DeteksiPenyakitTanamanVGG16.tflite", "wb").write(tflite_model)

```

### **Lampiran 2 Sourcecode Identifikasi Penyakit Tanaman Menggunakan Arsitektur InceptionV3**

```

import numpy as np
import os
import matplotlib.pyplot as plt
plt.style.use('default')
from skimage.io import imread
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import train_test_split
from __future__ import absolute_import, division, print_function, unicode_literals

import tensorflow as tf

```

```

#tf.logging.set_verbosity(tf.logging.ERROR)
#tf.enable_eager_execution()

import tensorflow_hub as hub
import os
from tensorflow.keras.layers import Dense, Flatten, Conv2D
from tensorflow.keras import Model
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam
from tensorflow.keras import layers
#from keras import optimizers

import keras
import keras.backend as K
from keras.preprocessing.image import load_img, img_to_array,
ImageDataGenerator
from keras.utils.np_utils import to_categorical

from keras.utils.data_utils import get_file

from keras import layers
from keras.models import Sequential, Model
from keras.callbacks import EarlyStopping, ModelCheckpoint

from sklearn.model_selection import train_test_split

```

```

!apt-get install subversion > /dev/null

#Retreive specific diseases of tomato for training
!svn export https://github.com/spMohanty/PlantVillage-Dataset/
trunk/raw/color/Tomato__Bacterial_spot image/Tomato__Bacteri
al_spot > /dev/null
!svn export https://github.com/spMohanty/PlantVillage-Dataset/
trunk/raw/color/Tomato__Early_blight image/Tomato__Early_bli
ght > /dev/null
!svn export https://github.com/spMohanty/PlantVillage-Dataset/
trunk/raw/color/Tomato__Late_blight image/Tomato__Late_bli
ght > /dev/null
!svn export https://github.com/spMohanty/PlantVillage-Dataset/
trunk/raw/color/Tomato__Septoria_leaf_spot image/Tomato__Sep
toria_leaf_spot > /dev/null
!svn export https://github.com/spMohanty/PlantVillage-Dataset/
trunk/raw/color/Tomato__Target_Spot image/Tomato__Target_Spo
t > /dev/null
!svn export https://github.com/spMohanty/PlantVillage-Dataset/
trunk/raw/color/Tomato__healthy image/Tomato__healthy > /de
v/null

```

```
!svn export https://github.com/spMohanty/PlantVillage-Dataset/
trunk/raw/color/Tomato__Tomato_mosaic_virus image/Tomato__mo
saic_virus > /dev/null
!svn export https://github.com/spMohanty/PlantVillage-Dataset/
trunk/raw/color/Tomato__Spider_mites%20Two-spotted_spider_mit
e image/Tomato__Spider_mites_20Two-spotted_spider_mite > /de
v/null
!svn export https://github.com/spMohanty/PlantVillage-Dataset/
trunk/raw/color/Tomato__Leaf_Mold image/Tomato__Leaf_Mold >
/dev/null
!svn export https://github.com/spMohanty/PlantVillage-Dataset/
trunk/raw/color/Tomato__Tomato_Yellow_Leaf_Curl_Virus image/T
omato__Yellow_Leaf_Curl_Virus > /dev/null
!svn export https://github.com/icassava/fgvcx-icassava/trunk/d
ata/train/cbb image/Cassava__cbb > /dev/null
!svn export https://github.com/icassava/fgvcx-icassava/trunk/d
ata/train/cbsd image/Cassava__cbsd > /dev/null
!svn export https://github.com/icassava/fgvcx-icassava/trunk/d
ata/train/cgm image/Cassava__cgm > /dev/null
!svn export https://github.com/icassava/fgvcx-icassava/trunk/d
ata/train/cmd image/Cassava__cmd > /dev/null
!svn export https://github.com/icassava/fgvcx-icassava/trunk/d
ata/train/healthy image/Cassava__healthy > /dev/null
```

```
plt.figure(figsize=(15,10))

#visualize several images

parent_directory = "image"

for i, folder in enumerate(os.listdir(parent_directory)):
    print(folder)
    folder_directory = os.path.join(parent_directory,folder)
    files = os.listdir(folder_directory)
    #will inspect only 1 image per folder
    file = files[0]
    file_path = os.path.join(folder_directory,file)

    image = imread(file_path)
    plt.subplot(1,15,i+1)
    plt.imshow(image)
    plt.axis("off")

    name = folder.split("__")[1][:-1]
    plt.title(name)
    plt.show()

#load everything into memory
x = []
y = []
```

```

class_names = []
parent_directory = "image"

for i, folder in enumerate(os.listdir(parent_directory)):
    print(i, folder)
    class_names.append(folder)
    folder_directory = os.path.join(parent_directory, folder)
    files = os.listdir(folder_directory)
    #will inspect only 1 image per folder
    for file in files:
        file_path = os.path.join(folder_directory, file)
        image = load_img(file_path, target_size=(75, 75))
        image = img_to_array(image)/255.
        x.append(image)
        y.append(i)

x = np.array(x)
y = to_categorical(y)

#check the data shape
print(x.shape)
print(y.shape)
print(y[0])

x_train, x, y_train, y = train_test_split(x, y, test_size=0.2,
stratify = y, random_state = 1)
x_valid,x_test, y_valid, y_test = train_test_split(x, y, test_
size=0.4, stratify = y, random_state = 1)

print("train data:",x_train.shape,y_train.shape)
print("validation data:",x_valid.shape,y_valid.shape)
print("test data:",x_test.shape,y_test.shape)

import tensorflow as tf
IMG_SHAPE = (75, 75, 3)
# Membuat model dasar (base model) dari pre-trained model MobileNet
base_model = tf.keras.applications.InceptionV3(input_shape=IMG_
_SHAPE,
                                               include_top=False
e,
                                               weights='imagenet')

base_model.trainable = True
base_model.summary()

model = tf.keras.Sequential([
    base_model,
    tf.keras.layers.Flatten(),

```

```

        tf.keras.layers.Dense(512, activation='relu'),
        tf.keras.layers.Dropout(rate=0.2),
        tf.keras.layers.Dense(15, activation='softmax')
    ])

model.compile("adam", loss="categorical_crossentropy", metrics=[ "acc"])
model.summary()

#utilize early stopping function to stop at the lowest validation loss
es = EarlyStopping(monitor='val_loss', patience=10, verbose=1,
mode='auto')
#utilize save best weight model during training
ckpt = ModelCheckpoint("DeteksiPenyakitTanaman.hdf5", monitor=
'val_loss', verbose=1, save_best_only=True, save_weights_only=
False, mode='auto', period=1)

#we will define a generator class for training data and validation data seperately, as no augmentation is not required for validation data
t_gen = ImageDataGenerator(rotation_range=90, horizontal_flip=True)
v_gen = ImageDataGenerator()
train_gen = t_gen.flow(x_train,y_train,batch_size=98)
valid_gen = v_gen.flow(x_valid,y_valid,batch_size=98)

history = model.fit_generator(
    train_gen,
    steps_per_epoch = train_gen.n // 98,
#callbacks = [ckpt],
    callbacks = [es,ckpt],
    validation_data = valid_gen,
    validation_steps = valid_gen.n // 98,
    epochs=100 #@param {type:"integer"}
)

```

```

plt.figure(figsize=(10, 4))
#plt.subplot(1, 5, 4)
plt.plot(history.history["acc"],label="Akurasi Pelatihan")
plt.plot(history.history["val_acc"],label="Validasi Akurasi")
plt.legend()
plt.title('Training and Validation Accuracy')
plt.ylabel("Accuracy (training and validation)")
plt.xlabel("Training Steps")
plt.show()
plt.figure(figsize=(10, 4))
#plt.subplot(1, 3, 2)
plt.plot(history.history["loss"],label="Kesalahan Pelatihan")

```

```

plt.plot(history.history["val_loss"],label="Validasi Kesalahan")
")
plt.legend()
plt.title('Training and Validation Loss')
plt.ylabel("Loss (training and validation)")
plt.xlabel("Training Steps")
plt.show()

#load the model weight file with lowest validation loss
model.load_weights("DeteksiPenyakitTanaman.hdf5")

#check the model metrics
print(model.metrics_names)
#evaluate training data
print(model.evaluate(x= x_train, y = y_train))
#evaluate validation data
print(model.evaluate(x= x_valid, y = y_valid))
#evaluate test data
print(model.evaluate(x= x_test, y = y_test))

#true label
y_true = np.argmax(y_test, axis=1)

#prediction label
Y_pred = model.predict(x_test)
y_pred = np.argmax(Y_pred, axis=1)

print(y_true)
print(y_pred)

from sklearn.metrics import confusion_matrix
from sklearn.utils.multiclass import unique_labels

def plot_confusion_matrix(y_true, y_pred, classes,
                           normalize=False,
                           title=None,
                           cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    if not title:
        if normalize:
            title = 'Normalized confusion matrix'
        else:
            title = 'Confusion matrix, without normalization'

    # Compute confusion matrix
    cm = confusion_matrix(y_true, y_pred)
    # Only use the labels that appear in the data
    #classes = classes[unique_labels(y_true, y_pred)]

```

```

if normalize:
    cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
]
    print("Normalized confusion matrix")
else:
    print('Confusion matrix, without normalization')

print(cm)

fig, ax = plt.subplots(figsize=(9, 9))
im = ax.imshow(cm, interpolation='nearest', cmap=cmap)
#ax.figure.colorbar(im, ax=ax)
# We want to show all ticks...
ax.set(xticks=np.arange(cm.shape[1]),
       yticks=np.arange(cm.shape[0]),
       # ... and label them with the respective list entries
       xticklabels=classes, yticklabels=classes,
       title=title,
       ylabel='True label',
       xlabel='Predicted label')

# Rotate the tick labels and set their alignment.
plt.setp(ax.get_xticklabels(), rotation=45, ha="right",
         rotation_mode="anchor")

# Loop over data dimensions and create text annotations.
fmt = '.2f' if normalize else 'd'
thresh = cm.max() / 2.
for i in range(cm.shape[0]):
    for j in range(cm.shape[1]):
        ax.text(j, i, format(cm[i, j], fmt),
                ha="center", va="center",
                color="white" if cm[i, j] > thresh else "black")
fig.tight_layout()
return ax

np.set_printoptions(precision=2)

plot_confusion_matrix(y_true, y_pred, classes=class_names, normalize=True,
                      title='Normalized confusion matrix')

keras_file = "DeteksiPenyakitTanaman.hdf5"
tf.keras.models.save_model(model, keras_file)

# Convert to TensorFlow Lite model.
converter = tf.lite.TFLiteConverter.from_keras_model_file(keras_file)
tflite_model = converter.convert()

open("DeteksiPenyakitTanamanIncept.tflite", "wb").write(tflite_model)

```

*Lampiran 3 SourceCode Identifikasi Penyakit Tanaman Menggunakan Arsitektur MobileNet*

```

import numpy as np
import os
import matplotlib.pyplot as plt
plt.style.use('default')
from skimage.io import imread
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import train_test_split
from __future__ import absolute_import, division, print_function, unicode_literals

import tensorflow as tf
#tf.logging.set_verbosity(tf.logging.ERROR)
#tf.enable_eager_execution()

import tensorflow_hub as hub
import os
from tensorflow.keras.layers import Dense, Flatten, Conv2D
from tensorflow.keras import Model
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam
from tensorflow.keras import layers
#from keras import optimizers

import keras
import keras.backend as K
from keras.preprocessing.image import load_img, img_to_array,
ImageDataGenerator
from keras.utils.np_utils import to_categorical

from keras.utils.data_utils import get_file

from keras import layers
from keras.models import Sequential, Model
from keras.callbacks import EarlyStopping, ModelCheckpoint

from sklearn.model_selection import train_test_split

!apt-get install subversion > /dev/null

#Retreive specific diseases of tomato for training

```

```
!svn export https://github.com/spMohanty/PlantVillage-Dataset/
trunk/raw/color/Tomato__Bacterial_spot image/Tomato__Bacteri
al_spot > /dev/null
!svn export https://github.com/spMohanty/PlantVillage-Dataset/
trunk/raw/color/Tomato__Early_blight image/Tomato__Early_bli
ght > /dev/null
!svn export https://github.com/spMohanty/PlantVillage-Dataset/
trunk/raw/color/Tomato__Late_blight image/Tomato__Late_bli
ght > /dev/null
!svn export https://github.com/spMohanty/PlantVillage-Dataset/
trunk/raw/color/Tomato__Septoria_leaf_spot image/Tomato__Sep
toria_leaf_spot > /dev/null
!svn export https://github.com/spMohanty/PlantVillage-Dataset/
trunk/raw/color/Tomato__Target_Spot image/Tomato__Target_Spo
t > /dev/null
!svn export https://github.com/spMohanty/PlantVillage-Dataset/
trunk/raw/color/Tomato__healthy image/Tomato__healthy > /de
v/null
!svn export https://github.com/spMohanty/PlantVillage-Dataset/
trunk/raw/color/Tomato__Tomato_mosaic_virus image/Tomato__mo
saic_virus > /dev/null
!svn export https://github.com/spMohanty/PlantVillage-Dataset/
trunk/raw/color/Tomato__Spider_mites%20Two-spotted_spider_mit
e image/Tomato__Spider_mites_20Two-spotted_spider_mite > /de
v/null
!svn export https://github.com/spMohanty/PlantVillage-Dataset/
trunk/raw/color/Tomato__Leaf_Mold image/Tomato__Leaf_Mold >
/dev/null
!svn export https://github.com/spMohanty/PlantVillage-Dataset/
trunk/raw/color/Tomato__Tomato_Yellow_Leaf_Curl_Virus image/T
omato__Yellow_Leaf_Curl_Virus > /dev/null
!svn export https://github.com/icassava/fgvcx-icassava/trunk/d
ata/train/cbb image/Cassava__cbb > /dev/null
!svn export https://github.com/icassava/fgvcx-icassava/trunk/d
ata/train/cbsd image/Cassava__cbsd > /dev/null
!svn export https://github.com/icassava/fgvcx-icassava/trunk/d
ata/train/cgm image/Cassava__cgm > /dev/null
!svn export https://github.com/icassava/fgvcx-icassava/trunk/d
ata/train/cmd image/Cassava__cmd > /dev/null
!svn export https://github.com/icassava/fgvcx-icassava/trunk/d
ata/train/healthy image/Cassava__healthy > /dev/null
```

```
plt.figure(figsize=(15,10))

#visualize several images

parent_directory = "image"

for i, folder in enumerate(os.listdir(parent_directory)):
    print(folder)
```

```

folder_directory = os.path.join(parent_directory,folder)
files = os.listdir(folder_directory)
#will inspect only 1 image per folder
file = files[0]
file_path = os.path.join(folder_directory,file)

image = imread(file_path)
plt.subplot(1,15,i+1)
plt.imshow(image)
plt.axis("off")

name = folder.split("__") [1] [-1]
plt.title(name)
plt.show()

#load everything into memory
x = []
y = []
class_names = []
parent_directory = "image"

for i, folder in enumerate(os.listdir(parent_directory)):
    print(i, folder)
    class_names.append(folder)
    folder_directory = os.path.join(parent_directory,folder)
    files = os.listdir(folder_directory)
    #will inspect only 1 image per folder
    for file in files:
        file_path = os.path.join(folder_directory,file)
        image = load_img(file_path,target_size=(64,64))
        image = img_to_array(image)/255.
        x.append(image)
        y.append(i)

x = np.array(x)
y = to_categorical(y)

#check the data shape
print(x.shape)
print(y.shape)
print(y[0])

x_train, _x, y_train, _y = train_test_split(x,y,test_size=0.2,
stratify = y, random_state = 1)
x_valid,x_test, y_valid, y_test = train_test_split(_x,_y,test_
size=0.4, stratify = _y, random_state = 1)

print("train data:",x_train.shape,y_train.shape)
print("validation data:",x_valid.shape,y_valid.shape)
print("test data:",x_test.shape,y_test.shape)

import tensorflow as tf

```

```

IMG_SHAPE = (64, 64, 3)
# Membuat model dasar (base model) dari pre-trained model MobileNet
base_model = tf.keras.applications.MobileNet(input_shape=IMG_SHAPE,
                                              include_top=False
                                              ,
                                              weights='imagenet')

base_model.trainable = True
base_model.summary()

model = tf.keras.Sequential([
    base_model,
    tf.keras.layers.Conv2D(32, 2, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dense(15, activation='softmax')
])

model.compile("adam", loss="categorical_crossentropy", metrics=[ "acc"])
model.summary()

#utilize early stopping function to stop at the lowest validation loss
es = EarlyStopping(monitor='val_loss', patience=10, verbose=1, mode='auto')
#utilize save best weight model during training
ckpt = ModelCheckpoint("DeteksiPenyakitTanaman.hdf5", monitor='val_loss', verbose=1, save_best_only=True, save_weights_only=False, mode='auto', period=1)

#we will define a generator class for training data and validation data seperately, as no augmentation is not required for validation data
t_gen = ImageDataGenerator(rotation_range=90, horizontal_flip=True)
v_gen = ImageDataGenerator()
train_gen = t_gen.flow(x_train,y_train,batch_size=98)
valid_gen = v_gen.flow(x_valid,y_valid,batch_size=98)

history = model.fit_generator(
    train_gen,
    steps_per_epoch = train_gen.n // 98,
    #callbacks = [ckpt],
    callbacks = [es,ckpt],
    validation_data = valid_gen,
)

```

```

    validation_steps = valid_gen.n // 98,
    epochs=100 #@param {type:"integer"}
)

plt.figure(figsize=(10, 4))
# plt.subplot(1, 5, 4)
plt.plot(history.history["acc"],label="Akurasi Pelatihan")
plt.plot(history.history["val_acc"],label="Validasi Akurasi")
plt.legend()
plt.title('Training and Validation Accuracy')
plt.ylabel("Accuracy (training and validation)")
plt.xlabel("Training Steps")
plt.show()
plt.figure(figsize=(10, 4))
# plt.subplot(1, 3, 2)
plt.plot(history.history["loss"],label="Kesalahan Pelatihan")
plt.plot(history.history["val_loss"],label="Validasi Kesalahan")
plt.legend()
plt.title('Training and Validation Loss')
plt.ylabel("Loss (training and validation)")
plt.xlabel("Training Steps")
plt.show()

#load the model weight file with lowest validation loss
model.load_weights("DeteksiPenyakitTanaman.hdf5")

#check the model metrics
print(model.metrics_names)
#evaluate training data
print(model.evaluate(x= x_train, y = y_train))
#evaluate validation data
print(model.evaluate(x= x_valid, y = y_valid))
#evaluate test data
print(model.evaluate(x= x_test, y = y_test))

#true label
y_true = np.argmax(y_test,axis=1)

#prediction label
Y_pred = model.predict(x_test)
y_pred = np.argmax(Y_pred, axis=1)

print(y_true)
print(y_pred)

from sklearn.metrics import confusion_matrix
from sklearn.utils.multiclass import unique_labels

```

```

def plot_confusion_matrix(y_true, y_pred, classes,
                           normalize=False,
                           title=None,
                           cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    if not title:
        if normalize:
            title = 'Normalized confusion matrix'
        else:
            title = 'Confusion matrix, without normalization'

    # Compute confusion matrix
    cm = confusion_matrix(y_true, y_pred)
    # Only use the labels that appear in the data
    #classes = classes[unique_labels(y_true, y_pred)]
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
    ]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)

    fig, ax = plt.subplots(figsize=(9, 9))
    im = ax.imshow(cm, interpolation='nearest', cmap=cmap)
    #ax.figure.colorbar(im, ax=ax)
    # We want to show all ticks...
    ax.set(xticks=np.arange(cm.shape[1]),
           yticks=np.arange(cm.shape[0]),
           # ... and label them with the respective list entries
           xticklabels=classes, yticklabels=classes,
           title=title,
           ylabel='True label',
           xlabel='Predicted label')

    # Rotate the tick labels and set their alignment.
    plt.setp(ax.get_xticklabels(), rotation=45, ha="right",
             rotation_mode="anchor")

    # Loop over data dimensions and create text annotations.
    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i in range(cm.shape[0]):
        for j in range(cm.shape[1]):
            ax.text(j, i, format(cm[i, j], fmt),
                    ha="center", va="center",
                    color="white" if cm[i, j] > thresh else "black")

```

```

fig.tight_layout()
return ax

np.set_printoptions(precision=2)

plot_confusion_matrix(y_true, y_pred, classes=class_names, normalize=True,
                      title='Normalized confusion matrix')

keras_file = "DeteksiPenyakitTanaman.hdf5"
tf.keras.models.save_model(model, keras_file)

# Convert to TensorFlow Lite model.
converter = tf.lite.TFLiteConverter.from_keras_model_file(keras_file)
tflite_model = converter.convert()

open("DeteksiPenyakitTanamanMoNet.tflite", "wb").write(tflite_model)

```

#### *Lampiran 4 SouceCode Pada Android Studio*

##### 1. AndroidManifest.xml

```

<?xml version="1.0" encoding="utf-8"?>
<manifest
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:dist="http://schemas.android.com/apk/distribution"
    package="com.tflite.Deteksipenyakittanaman">

    <dist:module dist:instant="true"/>

    <uses-permission android:name="android.permission.CAMERA" />
    <uses-permission
        android:name="android.permission.WRITE_EXTERNAL_STORAGE" />

    <uses-permission
        android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission
        android:name="android.permission.WRITE_INTERNAL_STORAGE" />
    <uses-permission
        android:name="android.permission.READ_INTERNAL_STORAGE" />

    <uses-feature android:name="android.hardware.camera" />
    <uses-feature android:name="android.hardware.camera.autofocus" />
</>

    <!--<uses-feature android:name="android.hardware.camera.flash" />-->

    <uses-permission
        android:name="android.permission.FLASHLIGHT" />

```

```

<application
    android:allowBackup="true"
    android:icon="@mipmap/icon"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/icon"
    android:supportsRtl="true"
    android:theme="@style/AppTheme"
    tools:ignore="GoogleAppIndexingWarning">
    <activity
        android:name="com.tflite.Deteksipenyakittanaman.SplashScreen"
        android:label="@string/app_name"
        android:noHistory="true"
        android:screenOrientation="portrait"
        android:theme="@style/AppThemeSplashScreen">
        <intent-filter>
            <action android:name="android.intent.action.MAIN"
/>
        <category
            android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <activity
        android:name="com.tflite.Deteksipenyakittanaman.Tentang" />
    <activity
        android:name="com.tflite.Deteksipenyakittanaman.Bantuan" />
    <activity
        android:name="com.tflite.Deteksipenyakittanaman.DeteksiDariGaleri"
        android:screenOrientation="portrait">
    </activity>
    <activity
        android:name="com.tflite.Deteksipenyakittanaman.Pendeteksi"
        android:screenOrientation="portrait"
        android:colorMode="wideColorGamut">
    </activity>
    <activity
        android:name="com.tflite.Deteksipenyakittanaman.MainActivity"
        android:screenOrientation="portrait">
    </activity>
</application>

</manifest>

```

## 2. MainActivity.kt

```

package com.tflite.Deteksipenyakittanaman

import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.View
import android.widget.Button

class MainActivity : AppCompatActivity(), View.OnClickListener {

```

```

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
    //memperkenalkan button yang sudah ditambahkan di layout
    activity_main.xml
    val btnPendeteksi: Button =
        findViewById(R.id.btn_move_detection)
        //menambahkan event onClick pada button btnMoveDetection
        btnPendeteksi.setOnClickListener(this)

    val btnImporGaleri: Button =
        findViewById(R.id.btn_move_import)
        btnImporGaleri.setOnClickListener(this)

    val btnBantuan: Button = findViewById(R.id.btn_move_help)
    btnBantuan.setOnClickListener(this)

    val btnTentang: Button = findViewById(R.id.btn_move_about)
    btnTentang.setOnClickListener(this)
}

override fun onClick(v: View) {
    when (v.id) {
        R.id.btn_move_detection -> {
            //menambahkan suatu Intent pada method onClick()
            val moveIntent = Intent(this@MainActivity,
Pendeteksi::class.java)
                startActivity(moveIntent)
            }
        R.id.btn_move_import -> {
            val moveIntent = Intent(this@MainActivity,
DeteksiDariGaleri::class.java)
                startActivity(moveIntent)
            }
        R.id.btn_move_help -> {
            val moveIntent = Intent(this@MainActivity,
Bantuan::class.java)
                startActivity(moveIntent)
            }
        R.id.btn_move_about -> {
            val moveIntent = Intent(this@MainActivity,
Tentang::class.java)
                startActivity(moveIntent)
            }
    }
}
}

```

### 3. Klasifikasi.kt

```

package com.tflite.Deteksipenyakittanaman

import android.content.res.AssetManager
import android.graphics.Bitmap

```

```

import android.graphics.BitmapFactory
import org.tensorflow.lite.Interpreter
import java.io.BufferedReader
import java.io.FileInputStream
import java.io.IOException
import java.io.InputStreamReader
import java.nio.ByteBuffer
import java.nio.ByteOrder
import java.nio.channels.FileChannel
import java.util.*

class Klasifikasi(assetManager: AssetManager) {

    private val labels: List<String>
    private val model: Interpreter

    init {
        model = Interpreter(getModelByteBuffer(assetManager,
MODEL_PATH))
        labels = getLabels(assetManager, LABELS_PATH)
    }

    fun recognize(data: ByteArray): List<Deteksi> {
        val result = Array(BATCH_SIZE) { FloatArray(labels.size) }

        val unscaledBitmap = BitmapFactory.decodeByteArray(data,
0, data.size)
        val bitmap =
            Bitmap.createScaledBitmap(unscaledBitmap,
MODEL_INPUT_SIZE, MODEL_INPUT_SIZE, false)

        val byteBuffer = ByteBuffer
            .allocateDirect(
                BATCH_SIZE *
                MODEL_INPUT_SIZE *
                MODEL_INPUT_SIZE *
                BYTES_PER_CHANNEL *
                PIXEL_SIZE
            )
            .apply { order(ByteOrder.nativeOrder()) }

        val pixelValues = IntArray(MODEL_INPUT_SIZE *
MODEL_INPUT_SIZE)
        bitmap.getPixels(pixelValues, 0, bitmap.width, 0, 0,
bitmap.width, bitmap.height)

        var pixel = 0
        for (i in 0 until MODEL_INPUT_SIZE) {
            for (j in 0 until MODEL_INPUT_SIZE) {
                val pixelValue = pixelValues[pixel++]
                byteBuffer.putFloat((pixelValue shr 16 and 0xFF) /
255f)
                byteBuffer.putFloat((pixelValue shr 8 and 0xFF) / 255f)
                byteBuffer.putFloat((pixelValue and 0xFF) / 255f)
            }
        }
    }
}

```

```

    }

    model.run(byteBuffer, result)
    return parseResults(result)
}

private fun parseResults(result: Array<FloatArray>):
List<Deteksi> {

    val recognitions = mutableListOf<Deteksi>()

    labels.forEachIndexed { index, label ->
        val probability = result[0][index]
        recognitions.add(Deteksi(label, probability))
    }

    return recognitions.sortedByDescending { it.probability }
}

@Throws(IOException::class)
private fun getModelByteBuffer(assetManager: AssetManager,
modelPath: String): ByteBuffer {
    val fileDescriptor = assetManager.openFd(modelPath)
    val inputStream =
FileInputStream(fileDescriptor)
        val fileChannel = inputStream.channel
        val startOffset = fileDescriptor.startOffset
        val declaredLength = fileDescriptor.declaredLength
        return fileChannel.map(FileChannel.MapMode.READ_ONLY,
startOffset, declaredLength)
            .asReadOnlyBuffer()
}

@Throws(IOException::class)
private fun getLabels(assetManager: AssetManager, labelPath:
String): List<String> {
    val labels = ArrayList<String>()
    val reader =
BufferedReader(InputStreamReader(assetManager.open(labelPath)))
        while (true) {
            val label = reader.readLine() ?: break
            labels.add(label)
        }
    reader.close()
    return labels
}

companion object {
    private const val BATCH_SIZE = 1 // process only 1 image
at a time
    private const val MODEL_INPUT_SIZE = 75
    private const val BYTES_PER_CHANNEL = 4 // float size
    private const val PIXEL_SIZE = 3 // rgb

    private const val LABELS_PATH = "label.txt"
    private const val MODEL_PATH =

```

```

    "DeteksiPenyakitTanamanIncNew.tflite"
}

}

```

#### 4. Klasifikasi darigaleri.kt

```

package com.tflite.DeteksiPenyakittanaman

import android.content.res.AssetManager
import android.graphics.Bitmap
import android.util.Log
import org.tensorflow.lite.Interpreter
import java.io.FileInputStream
import java.nio.ByteBuffer
import java.nio.ByteOrder
import java.nio.MappedByteBuffer
import java.nio.channels.FileChannel
import java.util.*
import kotlin.Comparator
import kotlin.collections.ArrayList

class KlasifikasiDariGaleri(assetManager: AssetManager, modelPath: String, labelPath: String, inputSize: Int) {
    private var interpreter: Interpreter
    private var labellist: List<String>
    private val inputsize: Int = inputSize
    private val pixelsize: Int = 3
    private val imagemean = 0
    private val imagestd = 255.0f
    private val maxresults = 3
    private val threshold = 0.4f

    data class Recognition(
        var id: String = "",
        var title: String = "",
        var confidence: Float = 0F,
        val percent: Float = confidence*100
    ) {
        override fun toString(): String {
            return "Title = $title, Prediksi = $percent"
        }
    }

    init {
        interpreter = Interpreter(loadModelFile(assetManager,
modelPath))
        labellist = loadLabelList(assetManager, labelPath)
    }

    private fun loadModelFile(assetManager: AssetManager,
modelPath: String): MappedByteBuffer {
        val fileDescriptor = assetManager.openFd(modelPath)
        val inputStream =

```

```

InputStream(fileDescriptor.fileDescriptor)
    val fileChannel = inputStream.channel
    val startOffset = fileDescriptor.startOffset
    val declaredLength = fileDescriptor.declaredLength
    return fileChannel.map(FileChannel.MapMode.READ_ONLY,
startOffset, declaredLength)
}

private fun loadLabelList(assetManager: AssetManager,
labelPath: String): List<String> {
    return
assetManager.open(labelPath).bufferedReader().useLines {
it.toList() }

}

fun recognizeImage(bitmap: Bitmap): List<Recognition> {
    val scaledBitmap = Bitmap.createScaledBitmap(bitmap,
inputsize, inputsize, false)
    val byteBuffer = convertBitmapToByteBuffer(scaledBitmap)
    val result = Array(1) { FloatArray(labellist.size) }
    interpreter.run(byteBuffer, result)
    return getSortedResult(result)
}

private fun convertBitmapToByteBuffer(bitmap: Bitmap):
ByteBuffer {
    val byteBuffer = ByteBuffer.allocateDirect(4 * inputsize *
inputsize * pixelsize)
    byteBuffer.order(ByteOrder.nativeOrder())
    val intValues = IntArray(inputsize * inputsize)

    bitmap.getPixels(intValues, 0, bitmap.width, 0, 0,
bitmap.width, bitmap.height)
    var pixel = 0
    for (i in 0 until inputsize) {
        for (j in 0 until inputsize) {
            val `val` = intValues[pixel++]

                byteBuffer.putFloat((((`val`.shr(16) and 0xFF) -
imagemean) / imagestd))
                byteBuffer.putFloat((((`val`.shr(8) and 0xFF) -
imagemean) / imagestd))
                byteBuffer.putFloat((((`val` and 0xFF) -
imagemean) / imagestd))
        }
    }
    return byteBuffer
}

private fun getSortedResult(labelProbArray:
Array<FloatArray>): List<Recognition> {
    Log.d("KlasifikasiDariGaleri", "List Size: (%d, %d,

```

```
%d)".format(labelProbArray.size,labelProbArray[0].size,labellist.size))

    val pq = PriorityQueue(
        maxresults,
        Comparator<Recognition> {
            (_, _, confidence1), (_, _, confidence2)
                -> confidence1.compareTo(confidence2) * -1
        })

    for (i in labellist.indices) {
        val confidence = labelProbArray[0][i]
        if (confidence >= threshold) {
            pq.add(Recognition("") + i,
                if (labellist.size > i) labellist[i] else
                "Unknown", confidence)
        }
    }
    Log.d("KlasifikasiDariGaleri",
    "pqsize: (%d)".format(pq.size))

    val recognitions = ArrayList<Recognition>()
    val recognitionsSize = pq.size.coerceAtMost(maxresults)
    for (i in 0 until recognitionsSize) {
        recognitions.add(pq.poll())
    }
    return recognitions
}
}
```

## 5. Deteksi.kt

```
package com.tflite.Deteksipenyakittanaman

data class Deteksi(
    val name: String,
    val probability: Float
) {
    override fun toString() =
        "$name : ${probability*100}%"
}
```

## 6. deteksiDariGaleri.kt

```
package com.tflite.Deteksipenyakittanaman

import android.annotation.SuppressLint
import android.content.Intent
import android.content.pm.ActivityInfo
import android.graphics.Bitmap
import android.graphics.BitmapFactory
import android.graphics.Matrix
```

```

import android.os.Build
import android.os.Bundle
import android.os.SystemClock
import android.provider.MediaStore
import android.widget.Toast
import androidx.annotation.RequiresApi
import androidx.appcompat.app.ActionBar
import androidx.appcompat.app.AppCompatActivity
import kotlinx.android.synthetic.main.activity_import_gallery.*
import java.io.IOException

class DeteksiDariGaleri : AppCompatActivity() {
    private lateinit var mClassifier: KlasifikasiDariGaleri
    private lateinit var mBitmap: Bitmap

    private val mGalleryRequestCode = 2

    private val mInputSize = 75
    private val mModelPath = "DeteksiPenyakitTanamanIncNew.tflite"
    private val mLabelPath = "label.txt"
    private val mSamplePath = "yellow.JPG"

    private var lastProcessingTimeMs: Long = 0

    @SuppressLint("SetTextI18n")
    @RequiresApi(Build.VERSION_CODES.O)
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        if (supportActionBar != null) {
            supportActionBar as ActionBar.title = "Pendeteksi
Penyakit Tanaman"
        }

        requestedOrientation =
ActivityInfo.SCREEN_ORIENTATION_PORTRAIT
        setContentView(R.layout.activity_import_gallery)
        mClassifier = KlasifikasiDariGaleri(assets, mModelPath,
mLabelPath, mInputSize)

        resources.assets.open(mSamplePath).use {
            mBitmap = BitmapFactory.decodeStream(it)
            mBitmap = Bitmap.createScaledBitmap(mBitmap,
mInputSize, mInputSize, true)
            mPhotoImageView.setImageBitmap(mBitmap)
        }

        mGalleryButton.setOnClickListener {
            val callGalleryIntent = Intent(Intent.ACTION_PICK)
            callGalleryIntent.type = "image/*"
            startActivityForResult(callGalleryIntent,
mGalleryRequestCode)
        }
        mDetectButton.setOnClickListener {
            val startTime = SystemClock.uptimeMillis() //menghitung
waktu awal
        }
    }
}

```

```

        val results =
mClassifier.recognizeImage(mBitmap).firstOrNull()
    mResultTextView.text= results?.title+"\n Probabilitas:
"+results?.percent+"%"
        lastProcessingTimeMs = SystemClock.uptimeMillis() -
startTime//menghitung lamanya proses
    val waktu = lastProcessingTimeMs.toString() //konversi
ke string
    delaytime.text = "$waktu ms"

    }
}

@SuppressLint("MissingSuperCall", "SetTextI18n")
override fun onActivityResult(requestCode: Int, resultCode:
Int, data: Intent?) {
    if(requestCode == mGalleryrequestCode) {
        if (data != null) {
            val uri = data.data

            try {
                mBitmap =
MediaStore.Images.Media.getBitmap(this.contentResolver, uri)
            } catch (e: IOException) {
                e.printStackTrace()
            }

            println("Selesai!")
            mBitmap = scaleImage(mBitmap)
            mPhotoImageView.setImageBitmap(mBitmap)

        }
    } else {
        Toast.makeText(this, "Unrecognized request code",
Toast.LENGTH_LONG).show()
    }
}

fun scaleImage(bitmap: Bitmap?): Bitmap {
    val originalWidth = bitmap!!.width
    val originalHeight = bitmap.height
    val scaleWidth = mInputSize.toFloat() / originalWidth
    val scaleHeight = mInputSize.toFloat() / originalHeight
    val matrix = Matrix()
    matrix.postScale(scaleWidth, scaleHeight)
    return Bitmap.createBitmap(bitmap, 0, 0, originalWidth,
originalHeight, matrix, true)
}
}

```

## 7. Pendekripsi.kt

```

package com.tflite.Deteksipenyakittanaman

import android.Manifest
import android.annotation.SuppressLint
import android.content.pm.PackageManager
import android.os.AsyncTask
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.os.SystemClock
import android.widget.TextView
import android.widget.Toast
import androidx.appcompat.app.ActionBar
import androidx.core.app.ActivityCompat
import com.tflite.Deteksipenyakittanaman.Klasifikasi
import kotlinx.android.synthetic.main.activity_pendeteksi.*

class Pendekripsi : AppCompatActivity() {

    private lateinit var classifier: Klasifikasi
    //mendeklarasikan komponen TextView resultbar yang akan dimanipulasi
    private lateinit var resultbar: TextView
    private lateinit var processtime: TextView
    //deklarasi variabel lastprocessingtime bertipe data long
    private var lastProcessingTimeMs: Long = 0

    @SuppressLint("MissingPermission")
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_pendeteksi)

        //mengganti nama title bar tiap activity
        if (supportActionBar != null) {
            (supportActionBar as ActionBar).title = "Pendeteksi
Penyakit Tanaman"
        }

        //obyek TextView resultbar disesuaikan (cast) dengan komponen TextView ber-ID result_bar di layout activity_pendeteksi.xml melalui metode findViewById().
        resultbar = findViewById(R.id.result_bar)
        processtime = findViewById(R.id.process_time_bar)

        classifier = Klasifikasi(assets)

        if (!canUseCamera()) {
            requestCameraPermissions()
        } else {
            setupCamera()
        }
    }

    private fun requestCameraPermissions() {
        ActivityCompat.requestPermissions(

```

```

        this,
        arrayOf(Manifest.permission.CAMERA),
        REQUEST_CAMERA_CODE
    )
}

@SuppressLint("MissingPermission", "SetTextI18n")
private fun setupCamera() {
    camera.addPictureTakenListener {
        AsyncTask.execute {
            val startTime =
                SystemClock.uptimeMillis() //menghitung waktu awal
            val recognitions = classifier.recognize(it.data)
            val txt = recognitions.joinToString(separator =
                "\n")
            lastProcessingTimeMs = SystemClock.uptimeMillis()
            - startTime //menghitung lamanya proses
            val waktu =
                lastProcessingTimeMs.toString() //konversi ke string
            runOnUiThread {
                /*menampilkan hasil pada UI*/
                //Toast.makeText(this, txt,
                Toast.LENGTH_LONG).show()
                /*menampilkan hasil pada layout TextView*/
                resultbar.text = txt
                processtime.text = "$waktu ms "
            }
        }
    }

    capturePhoto.setOnClickListener {
        camera.capture()
    }
}

override fun onRequestPermissionsResult(
    requestCode: Int,
    permissions: Array<out String>,
    grantResults: IntArray
) {
    super.onRequestPermissionsResult(requestCode, permissions,
    grantResults)

    if (REQUEST_CAMERA_CODE == requestCode) {
        if (grantResults.isNotEmpty() && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
            setupCamera()
        } else {
            Toast.makeText(this, "App needs camera in order to
work.", Toast.LENGTH_LONG).show()
            requestCameraPermissions()
        }
    }
}

```

```
@SuppressLint("MissingPermission")
override fun onResume() {
    super.onResume()
    if (canUseCamera()) {
        camera.start()
    }
}

override fun onPause() {
    if (canUseCamera()) {
        camera.stop()
    }
    super.onPause()
}

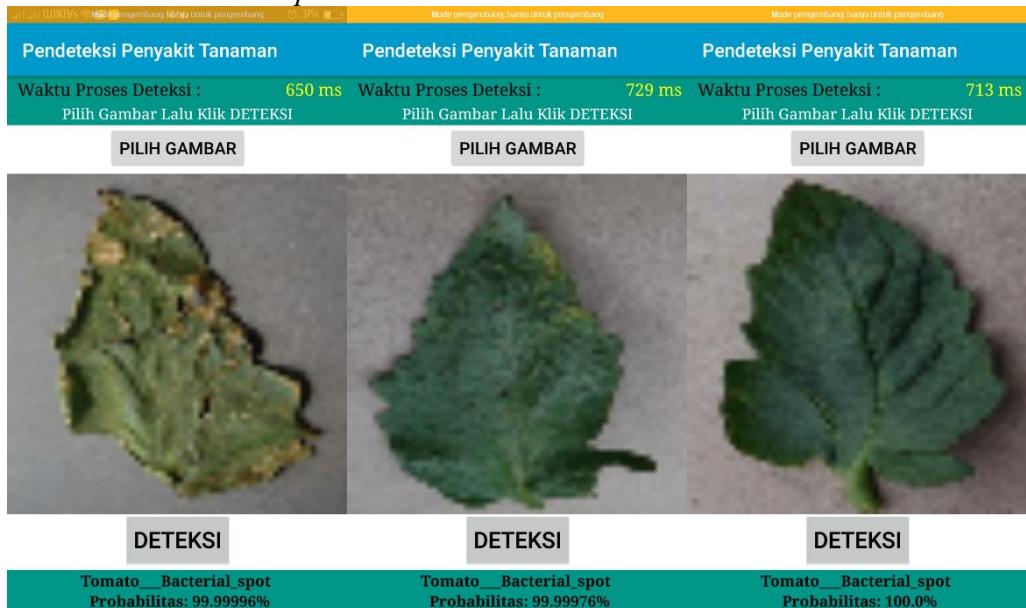
override fun onDestroy() {
    if (canUseCamera()) {
        camera.destroy()
    }
    super.onDestroy()
}

private fun canUseCamera() =
    ActivityCompat.checkSelfPermission(
        this,
        Manifest.permission.CAMERA
    ) == PackageManager.PERMISSION_GRANTED

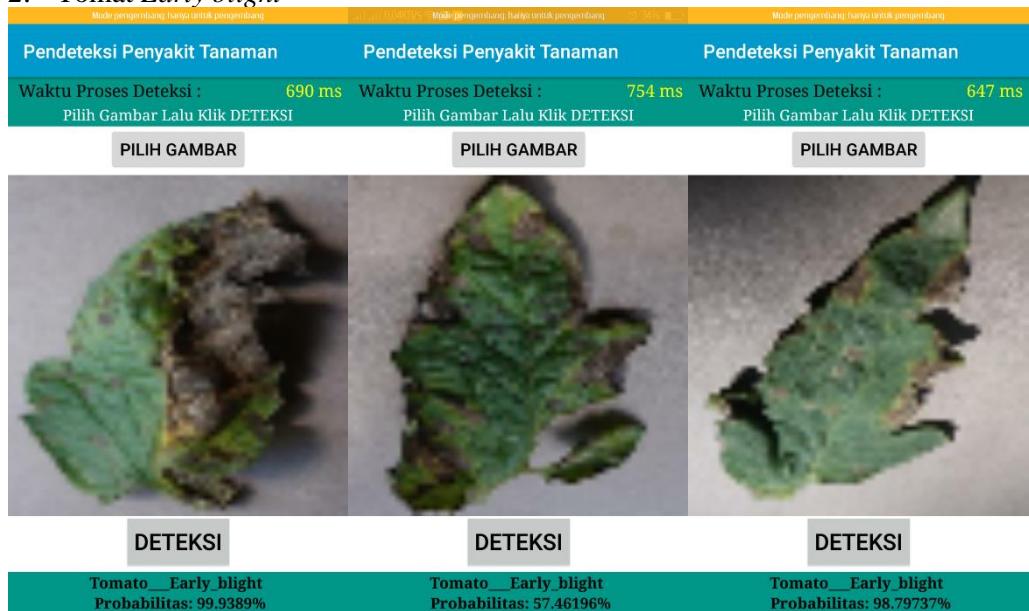
companion object {
    private const val REQUEST_CAMERA_CODE = 1
}
}
```

**Lampiran 5 Screenshot Data hasil uji Android menggunakan arsitektur VGG16**

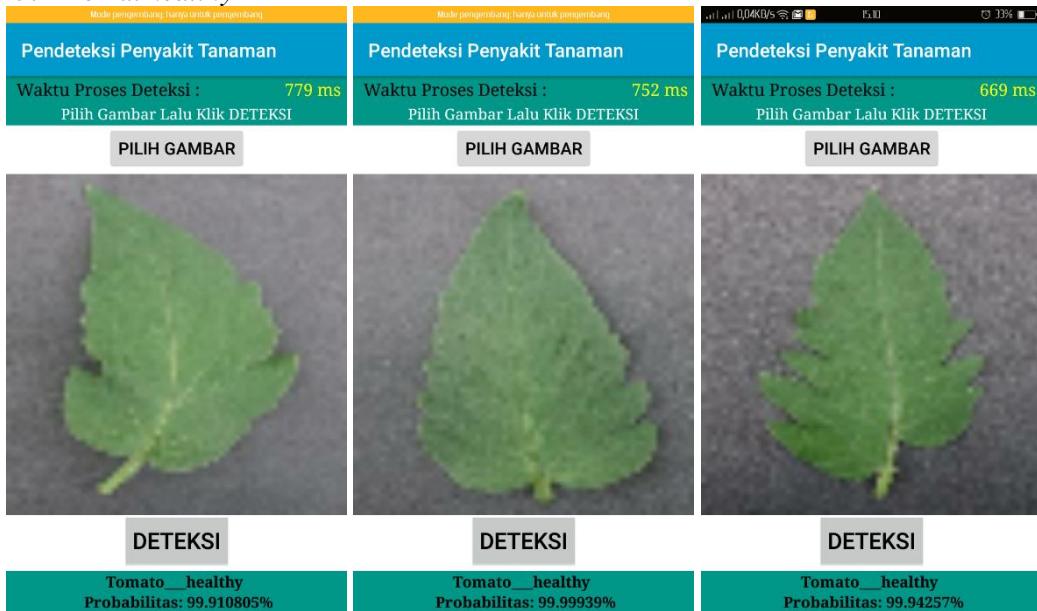
**1. Tomat Bacterial Spot**



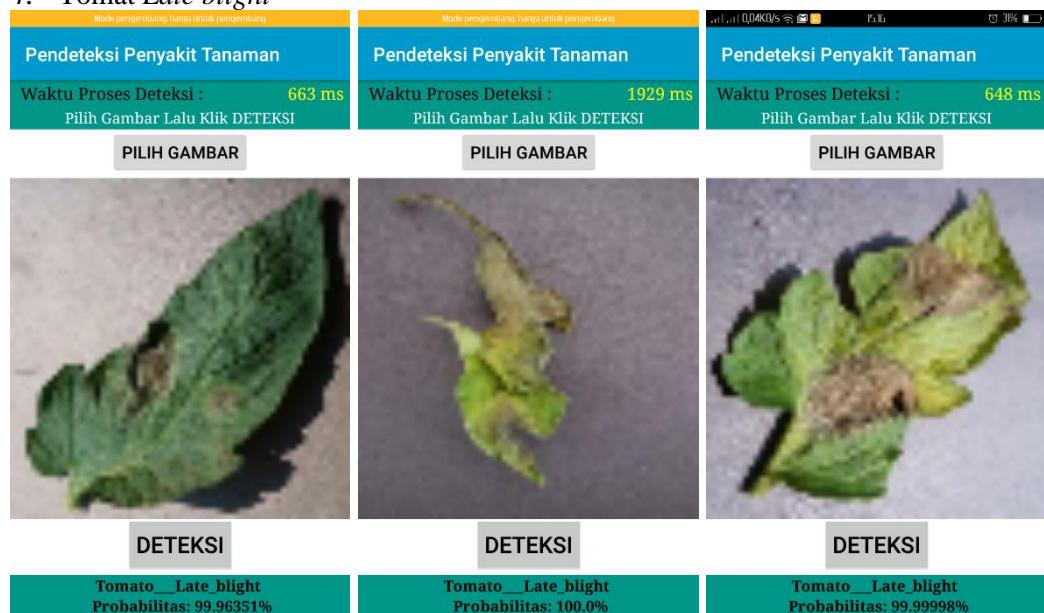
**2. Tomat Early blight**



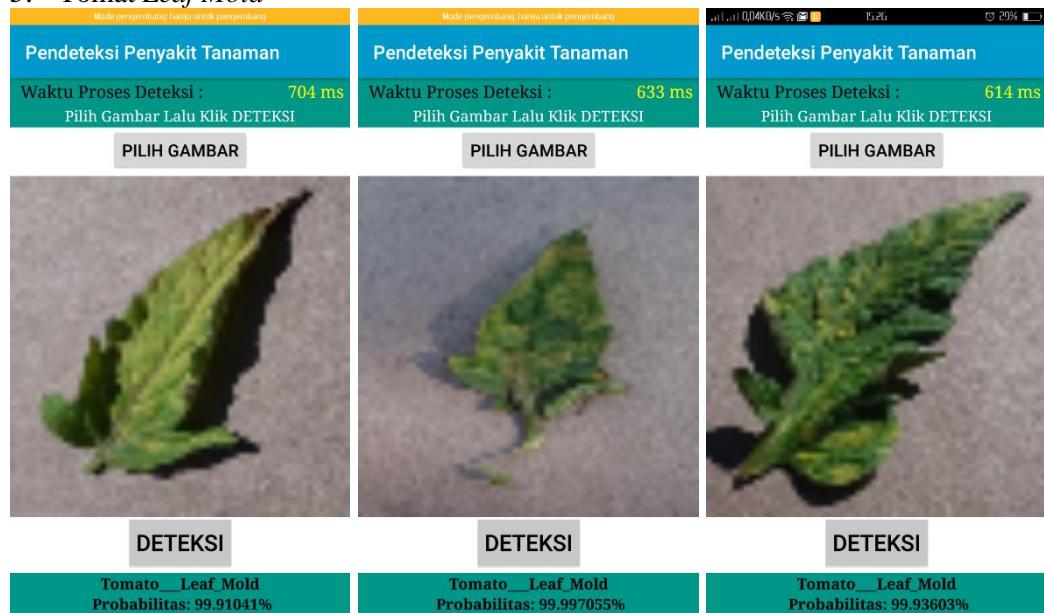
### 3. Tomat healthy



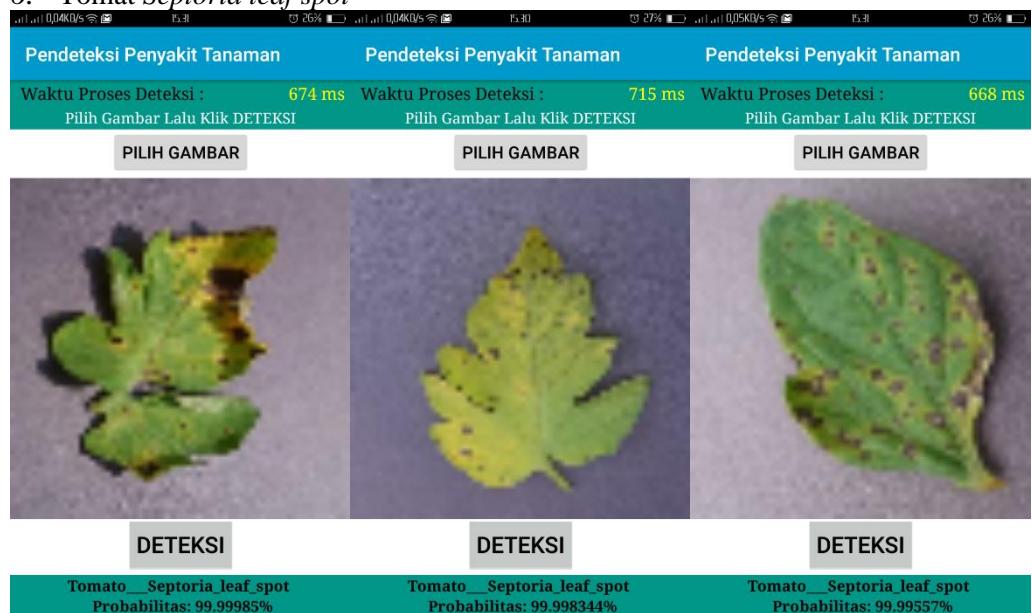
### 4. Tomat Late blight



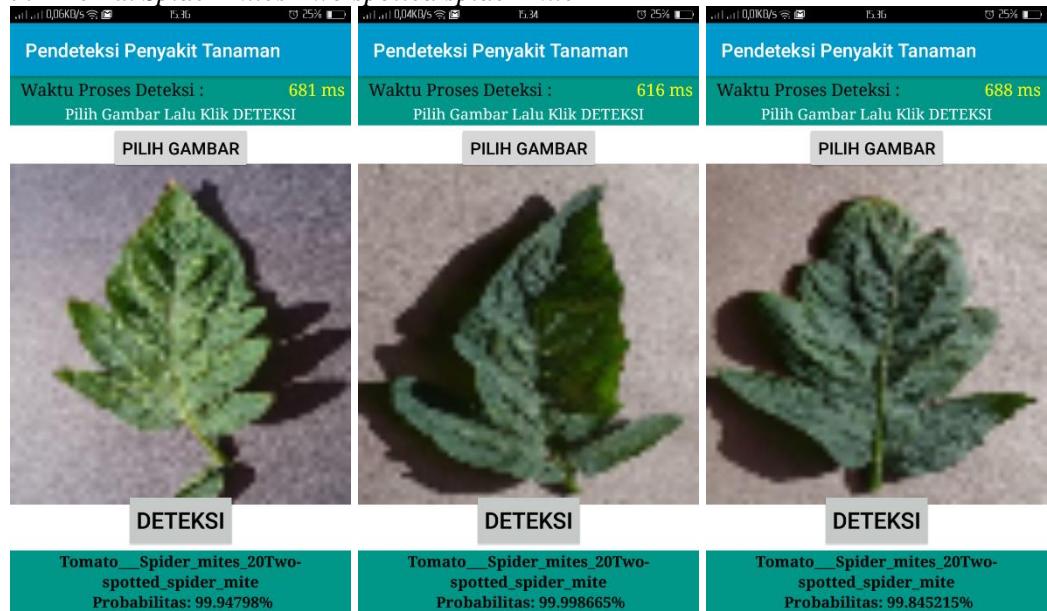
### 5. Tomat Leaf Mold



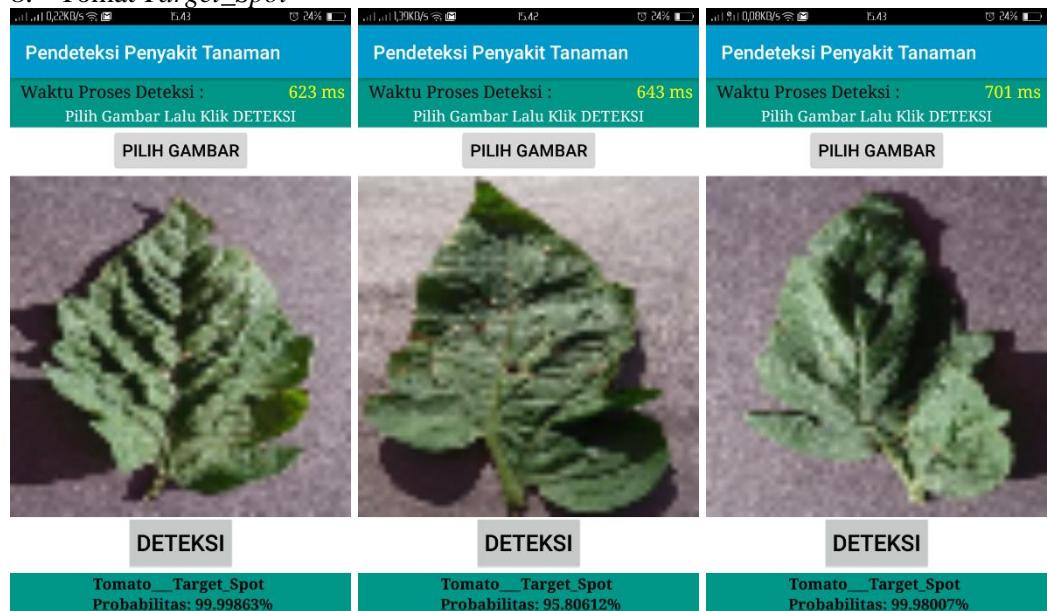
### 6. Tomat Septoria leaf spot



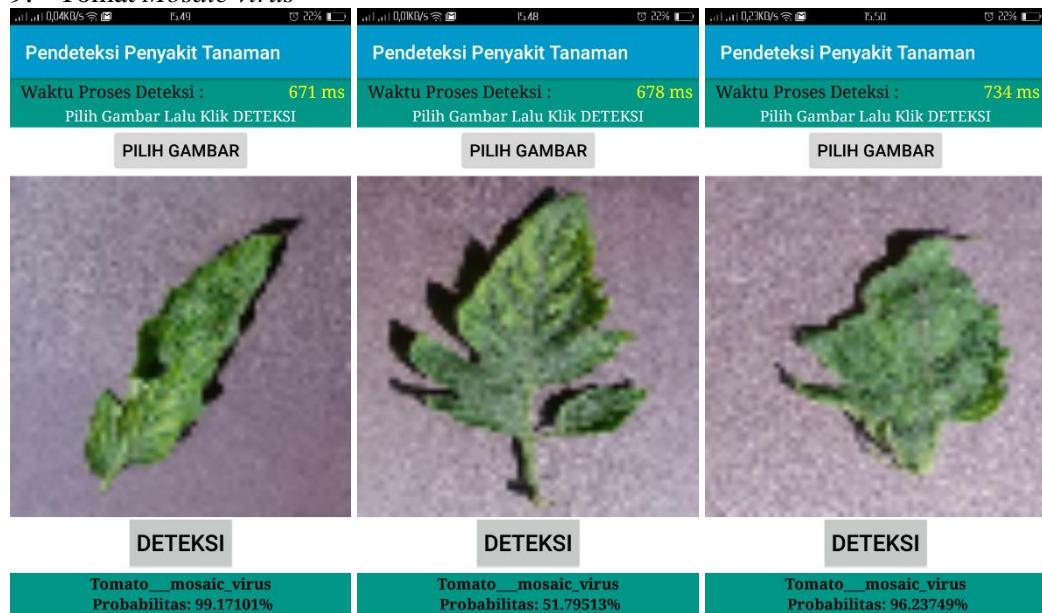
### 7. Tomat *Spider mites Two spotted spider mite*



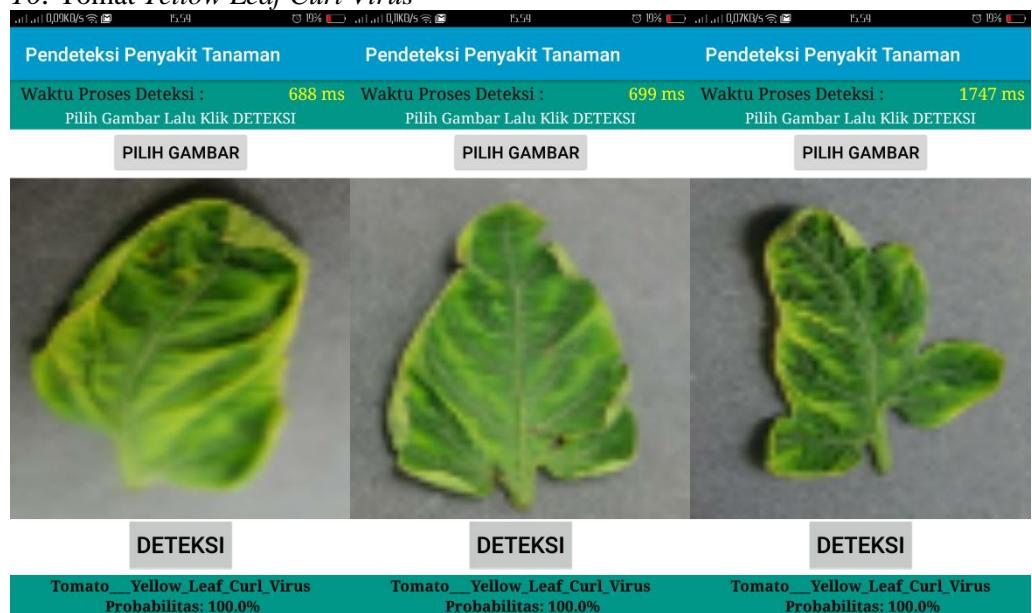
### 8. Tomat *Target\_Spot*



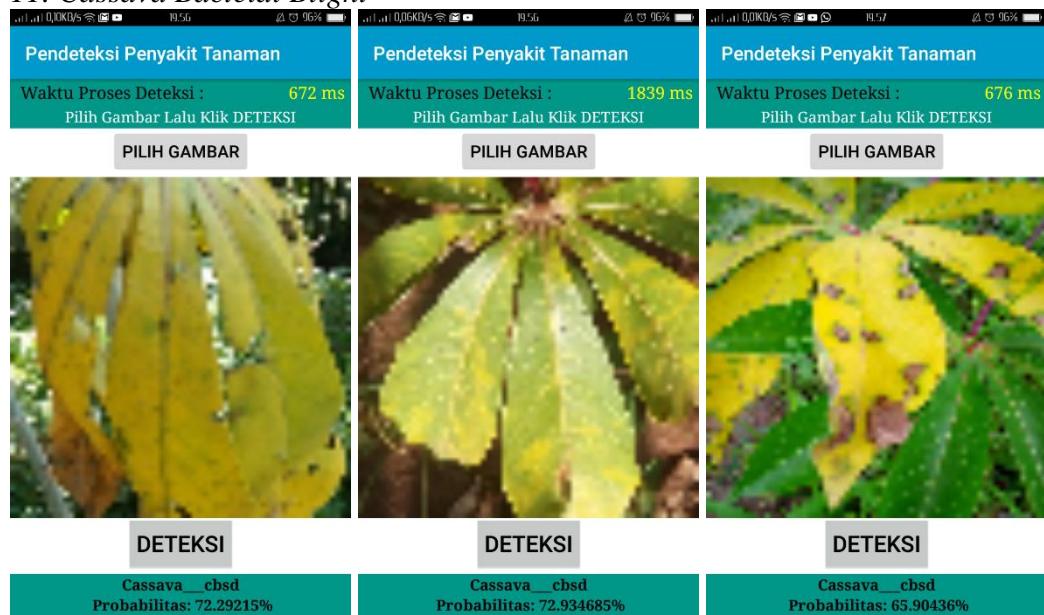
### 9. Tomat Mosaic virus



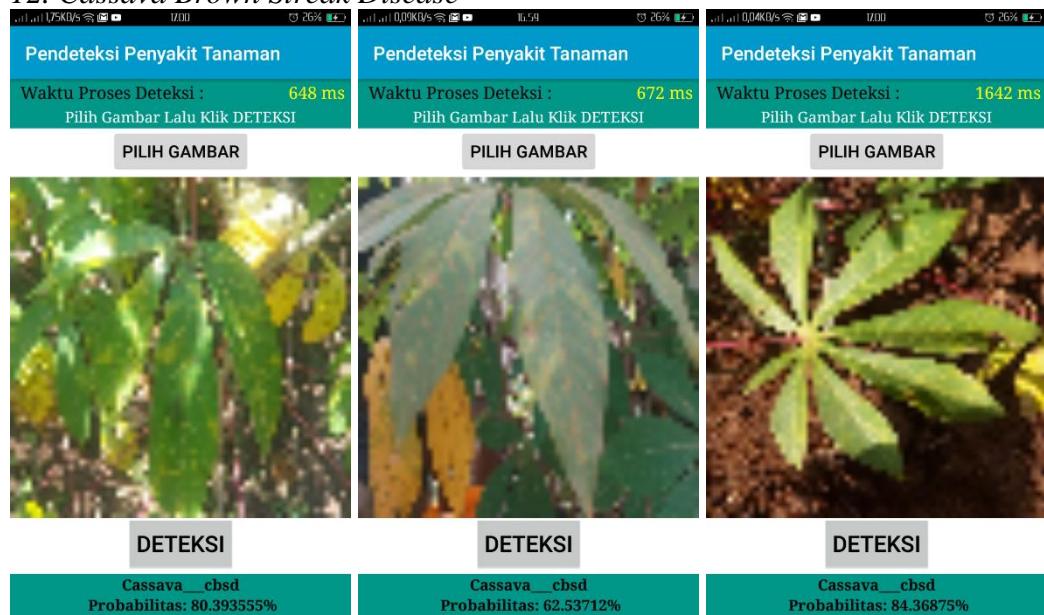
### 10. Tomat Yellow Leaf Curl Virus



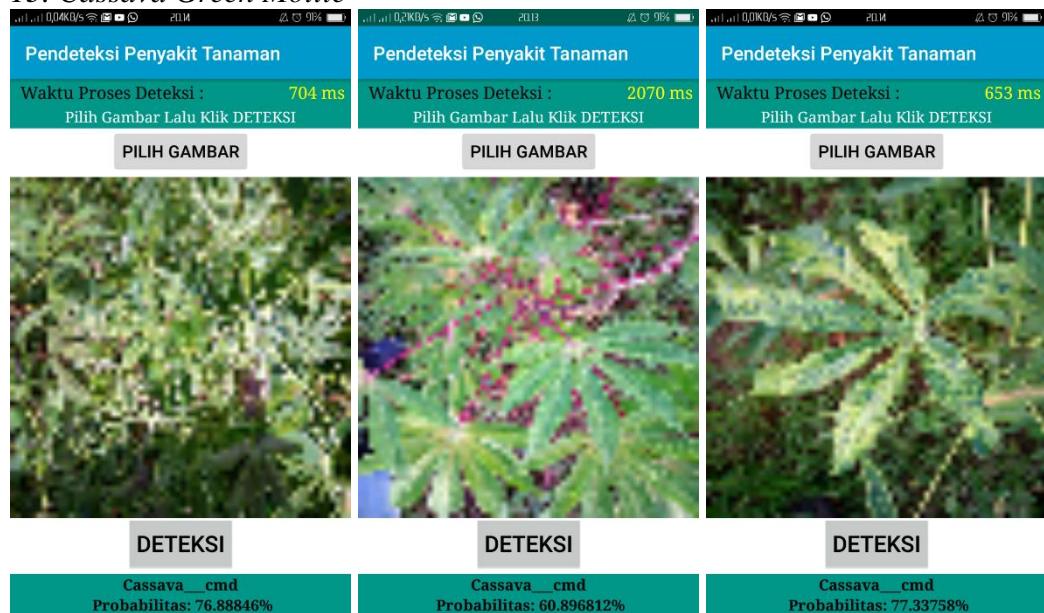
### 11. Cassava Bacterial Blight



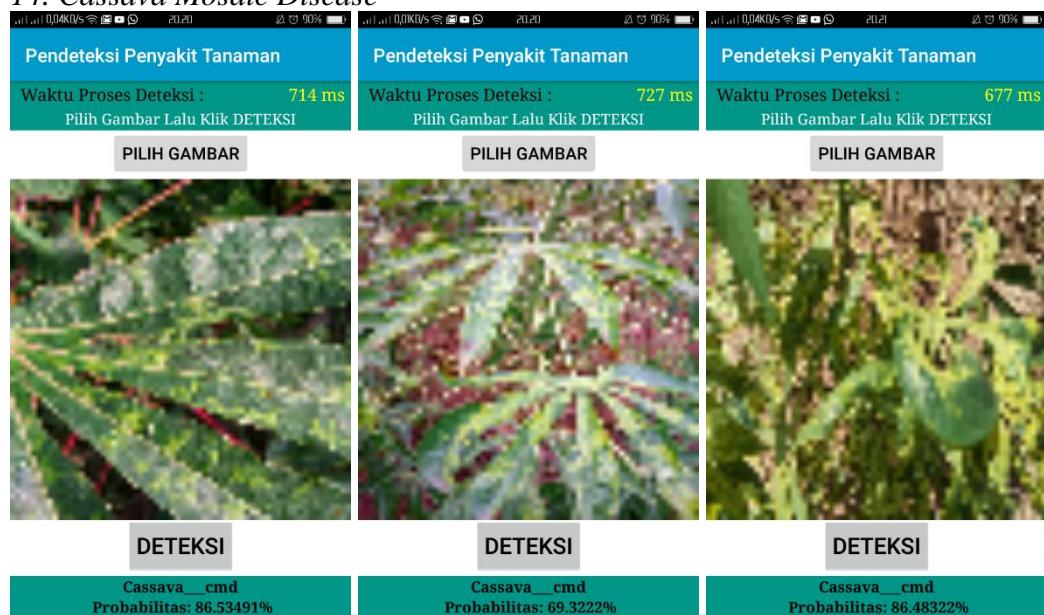
### 12. Cassava Brown Streak Disease



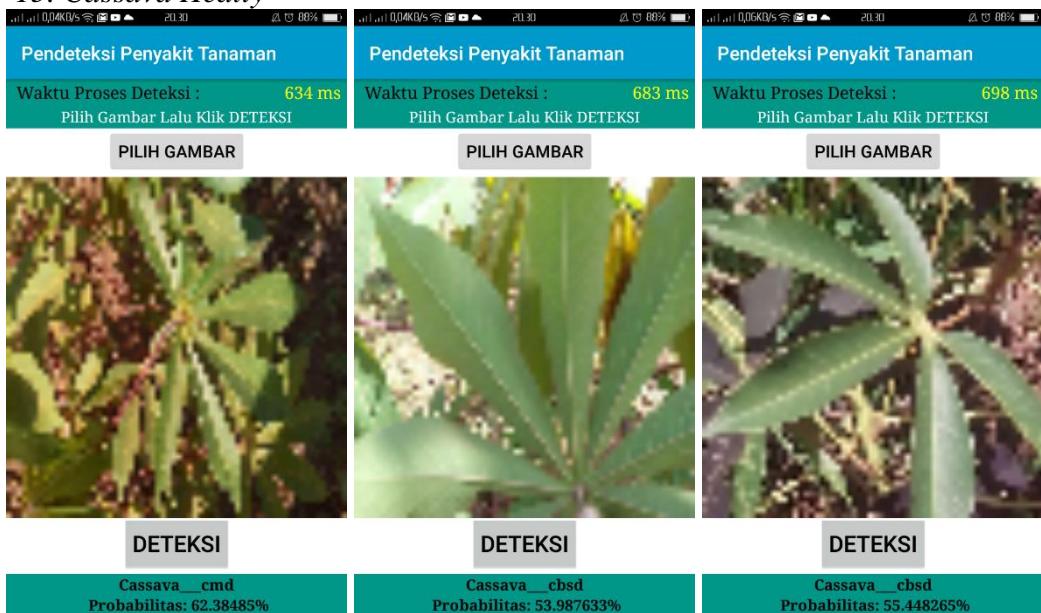
### 13. Cassava Green Mottle



### 14. Cassava Mosaic Disease

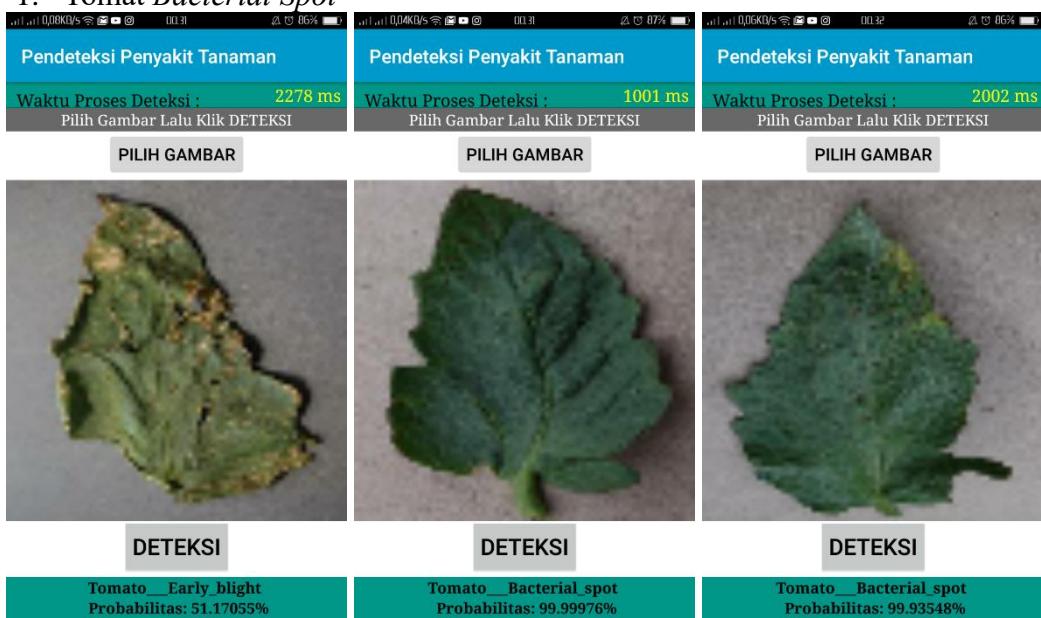


### 15. Cassava Healty

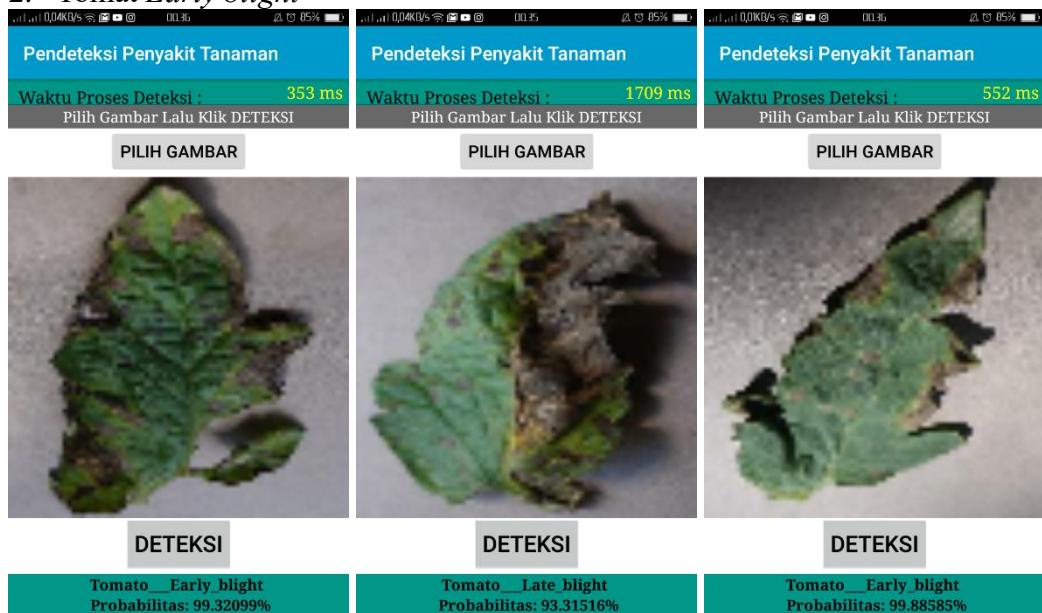


*Lampiran 6 Screenshot Data hasil uji Android menggunakan arsitektur InceptionV3*

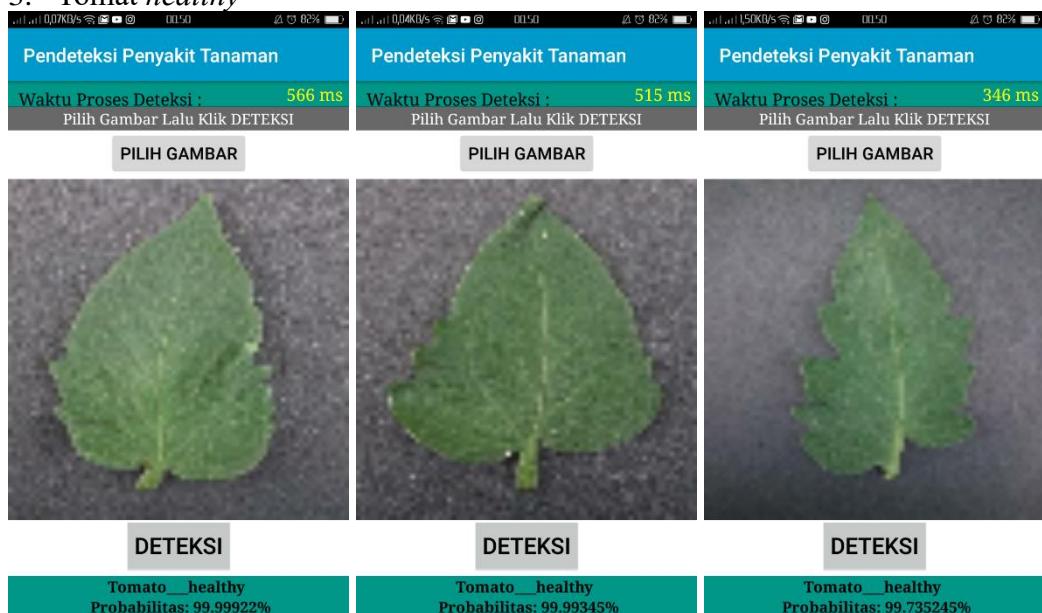
#### 1. Tomat Bacterial Spot



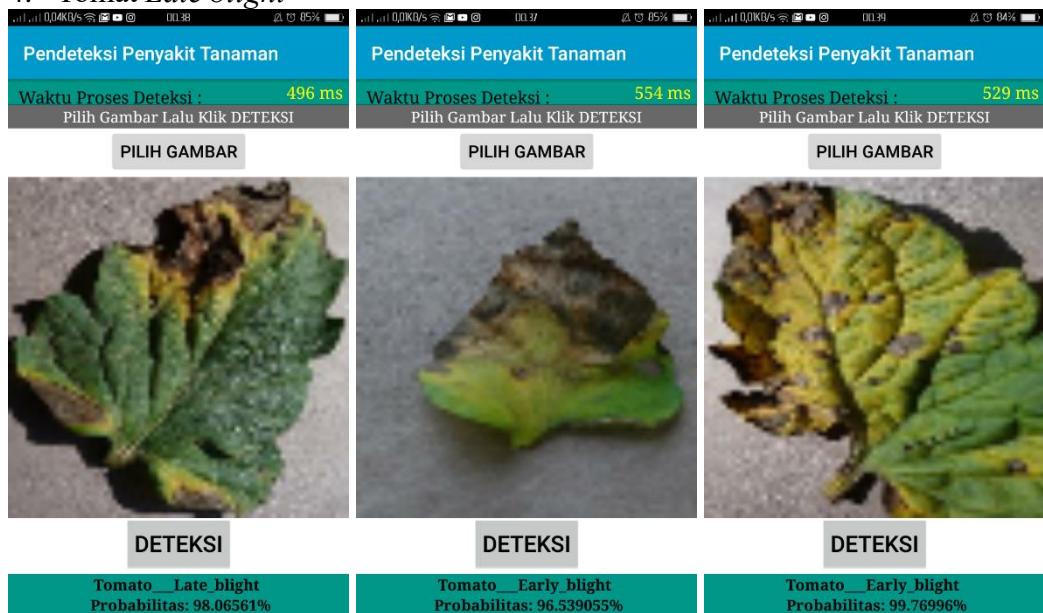
## 2. Tomat *Early blight*



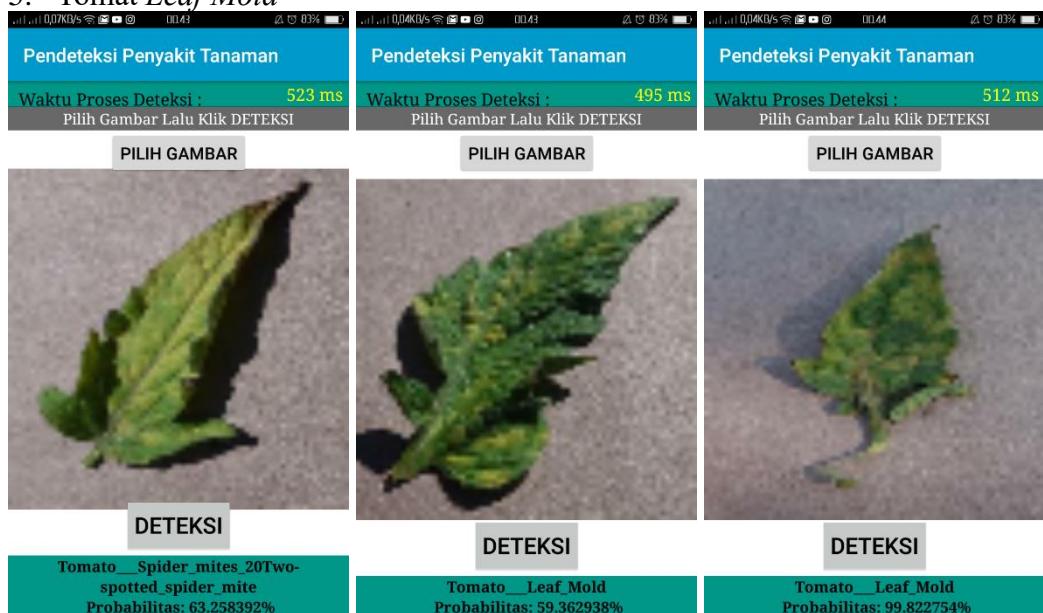
## 3. Tomat *healthy*



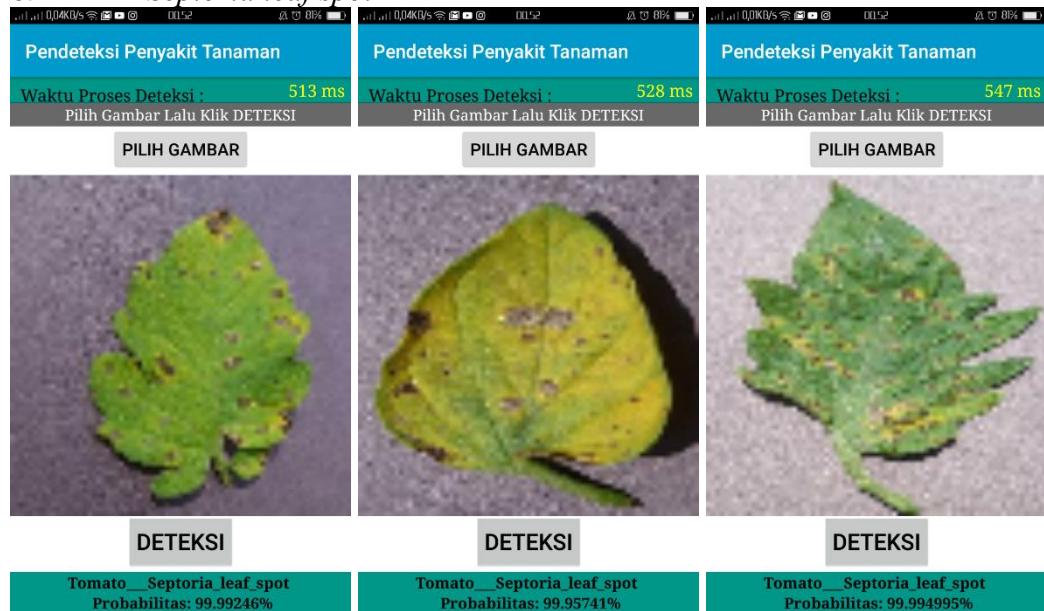
#### 4. Tomat Late blight



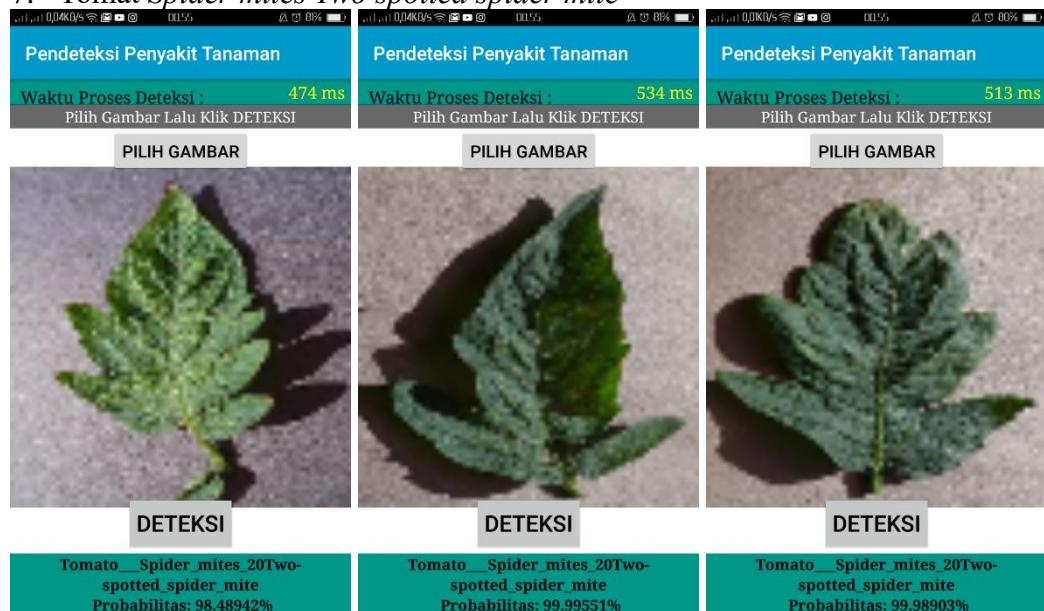
#### 5. Tomat Leaf Mold



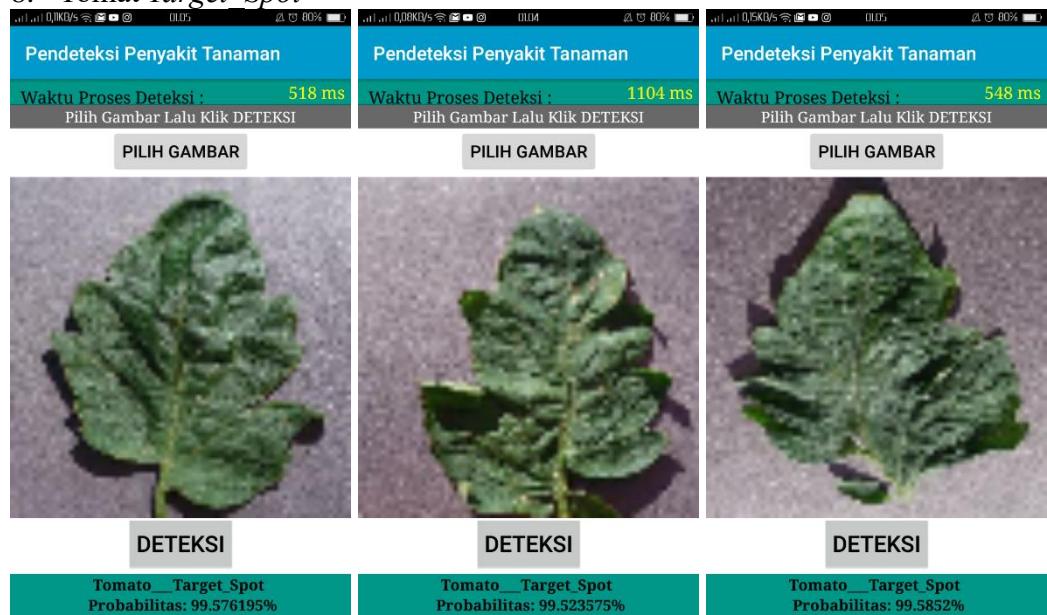
### 6. Tomat *Septoria leaf spot*



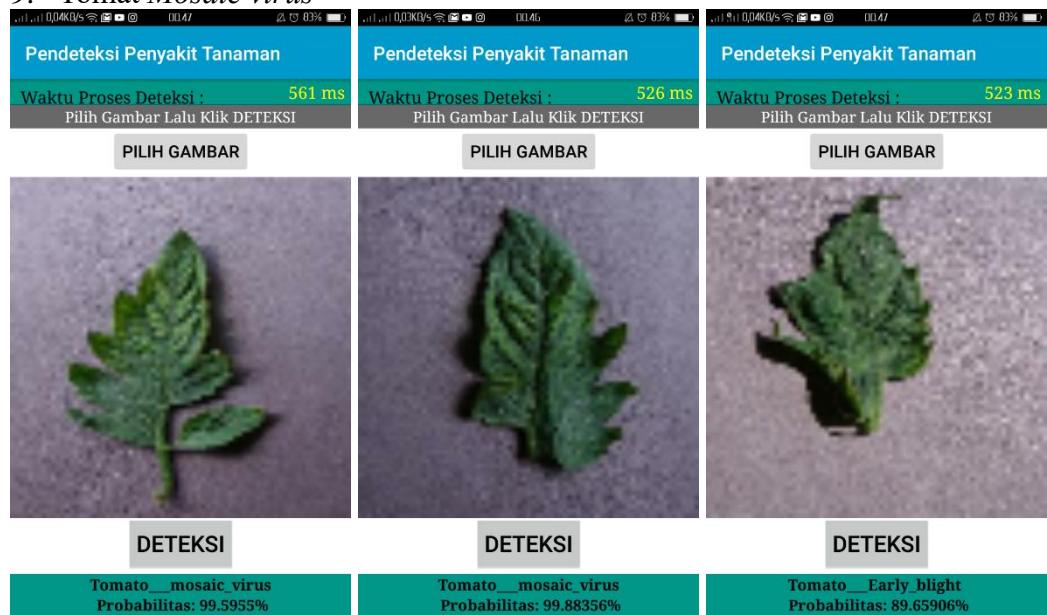
### 7. Tomat *Spider mites Two spotted spider mite*



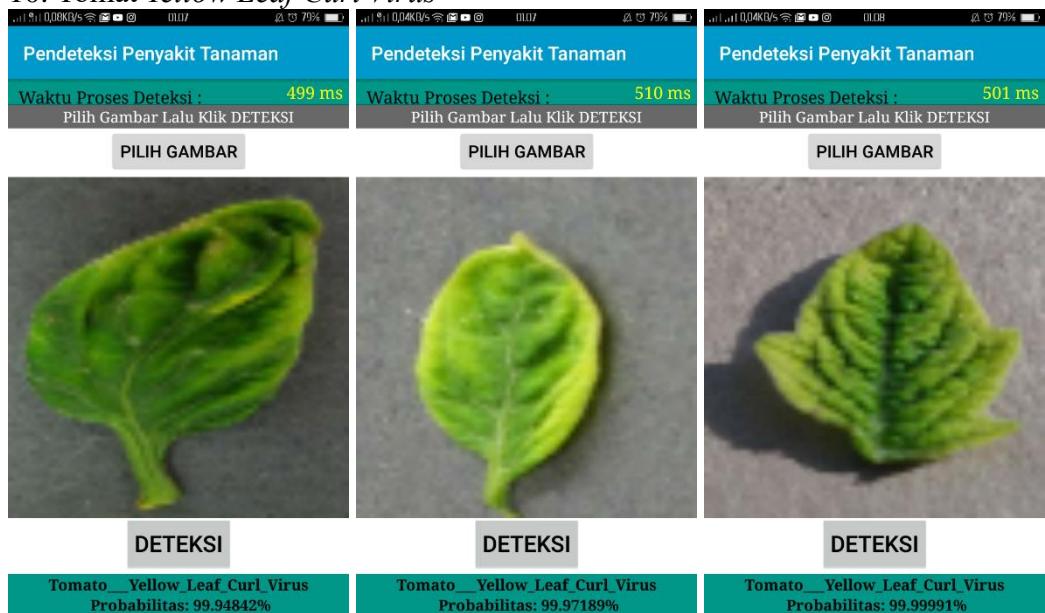
### 8. Tomat Target Spot



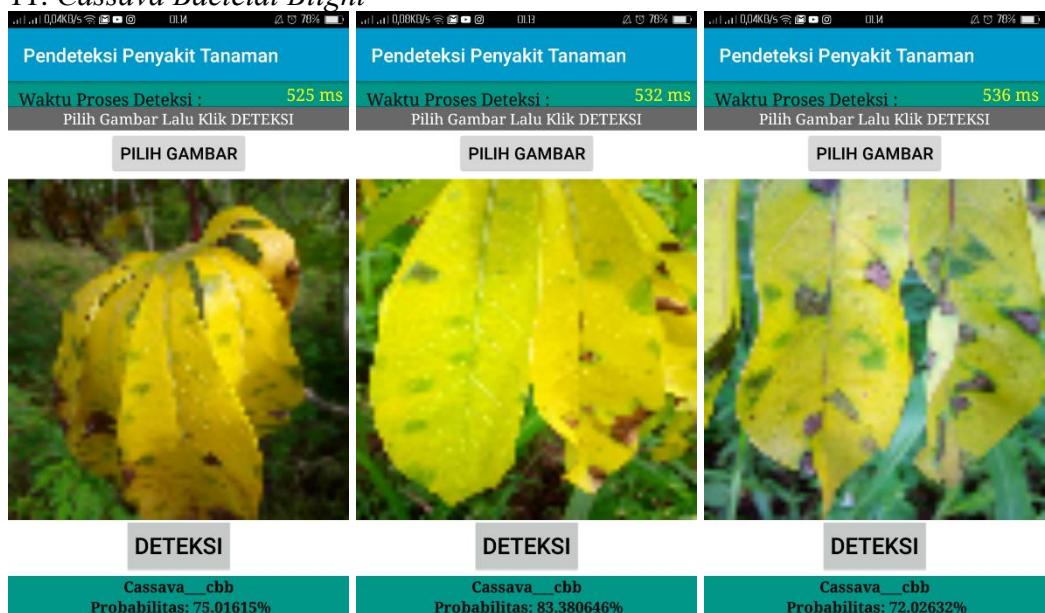
### 9. Tomat Mosaic virus



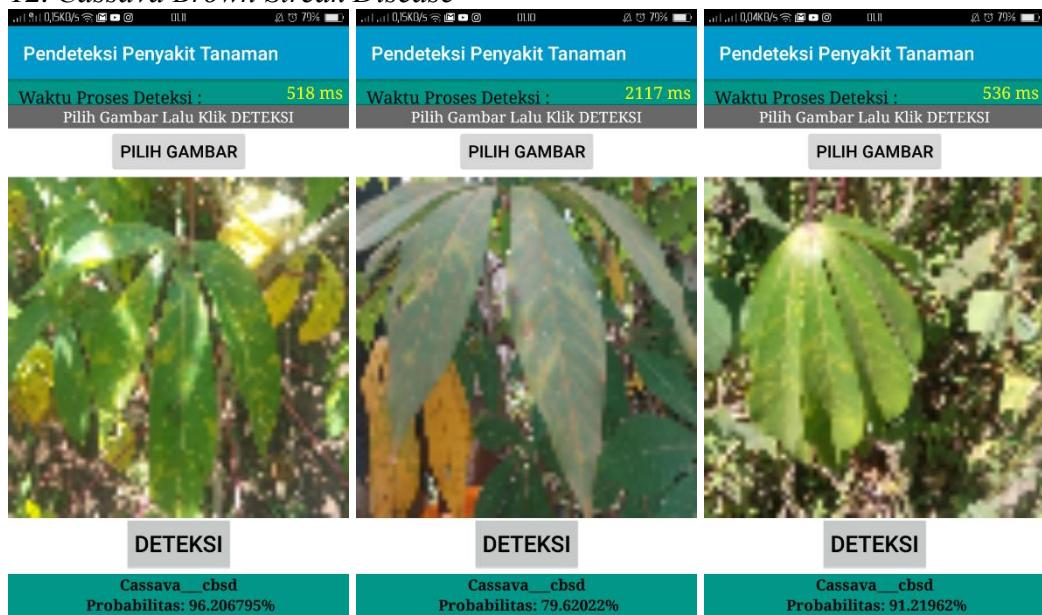
### 10. Tomat Yellow Leaf Curl Virus



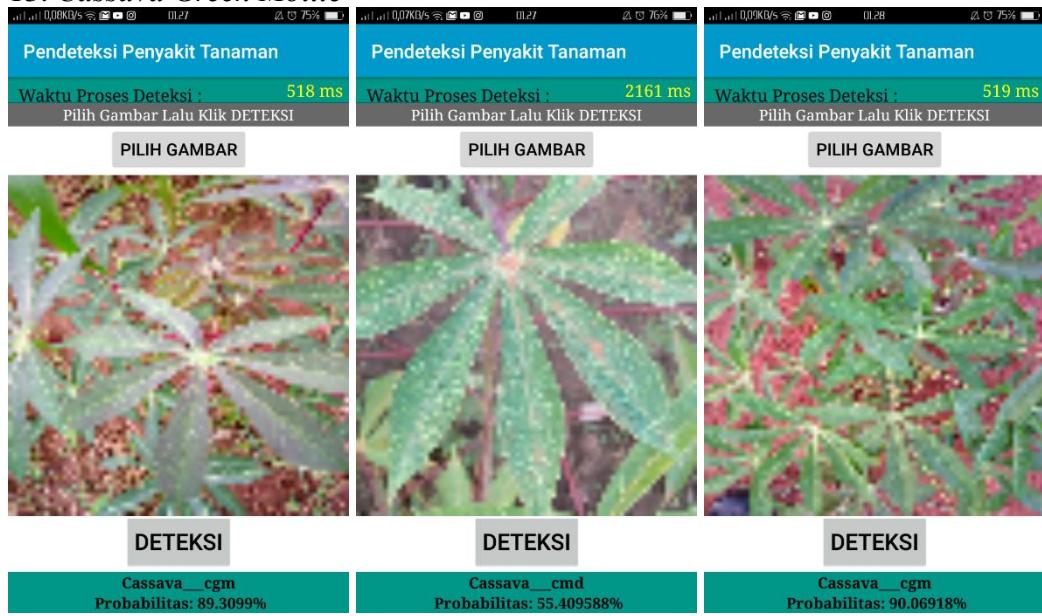
### 11. Cassava Bacterial Blight



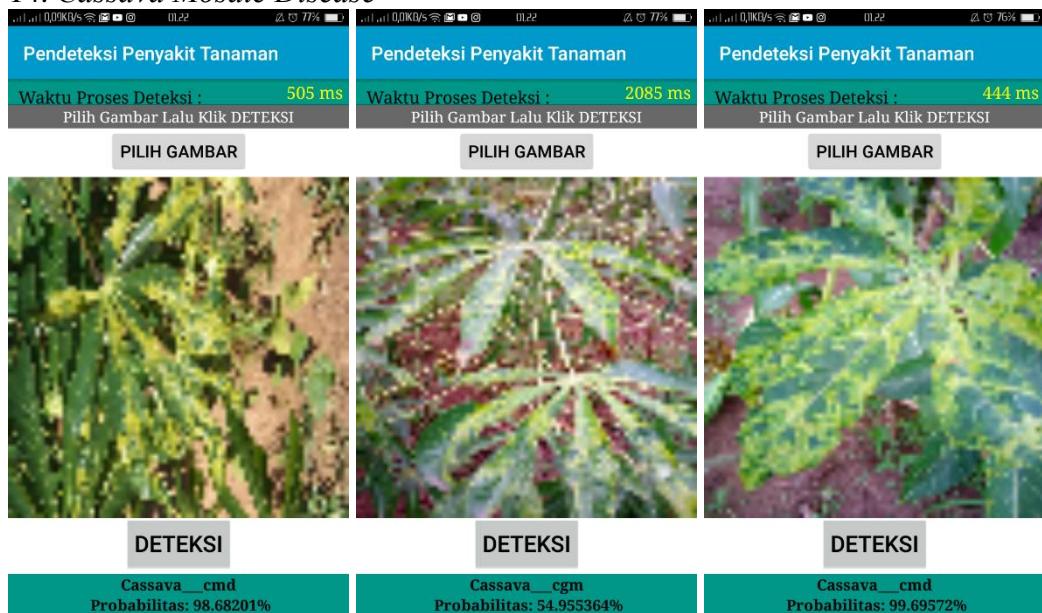
### 12. Cassava Brown Streak Disease



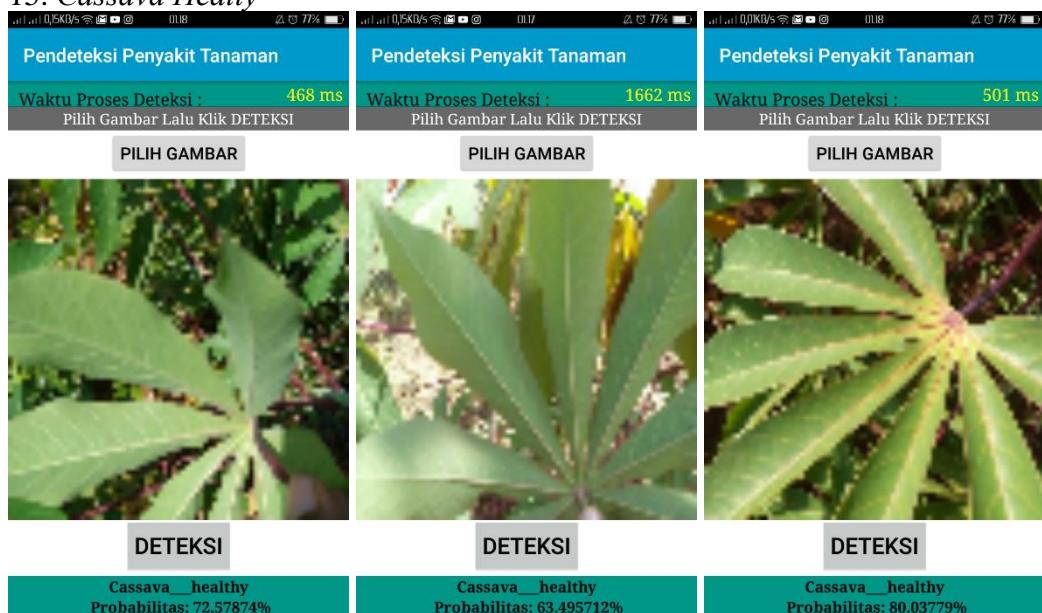
### 13. Cassava Green Mottle



#### 14. Cassava Mosaic Disease

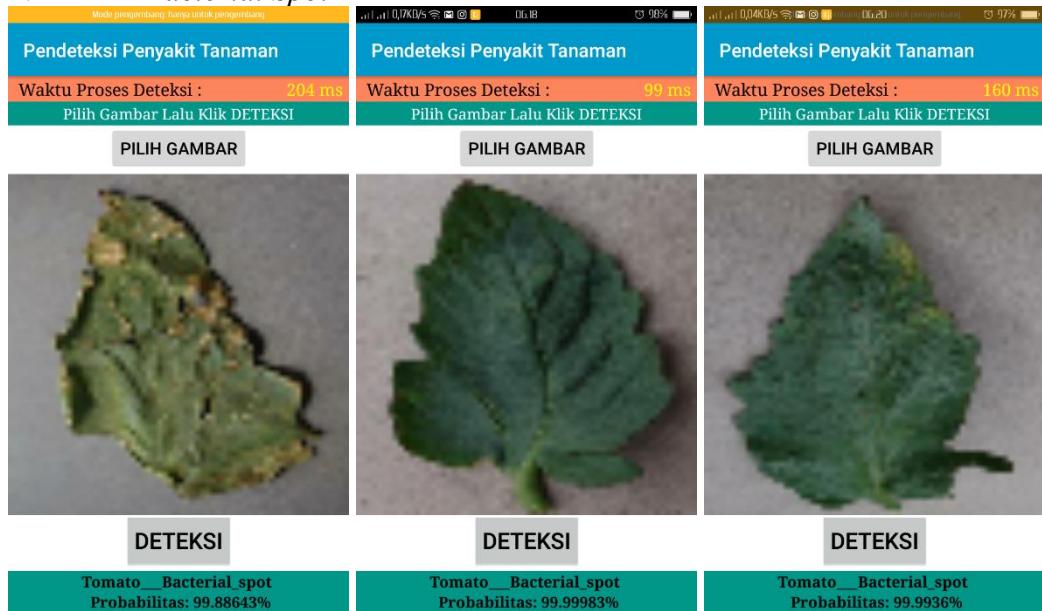


#### 15. Cassava Healthy

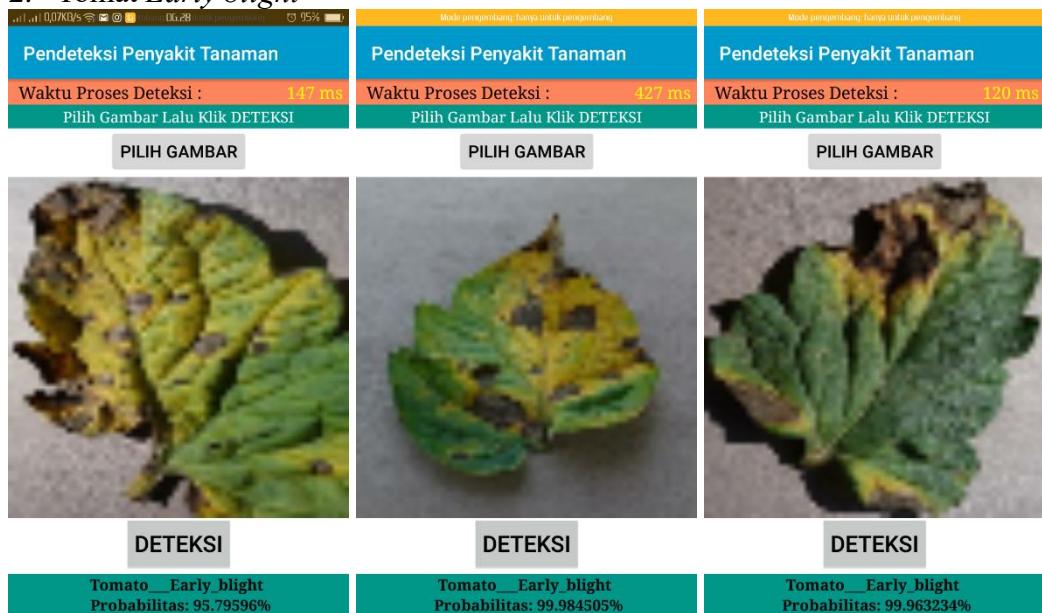


**Lampiran 7 Screenshot Data hasil uji Android menggunakan arsitektur MobileNet**

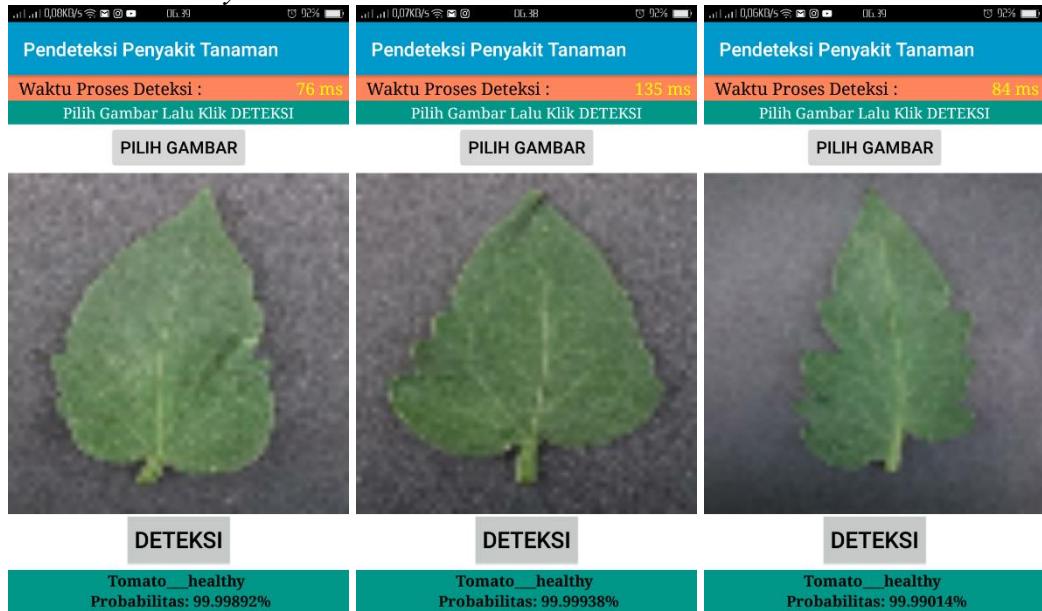
**1. Tomat Bacterial Spot**



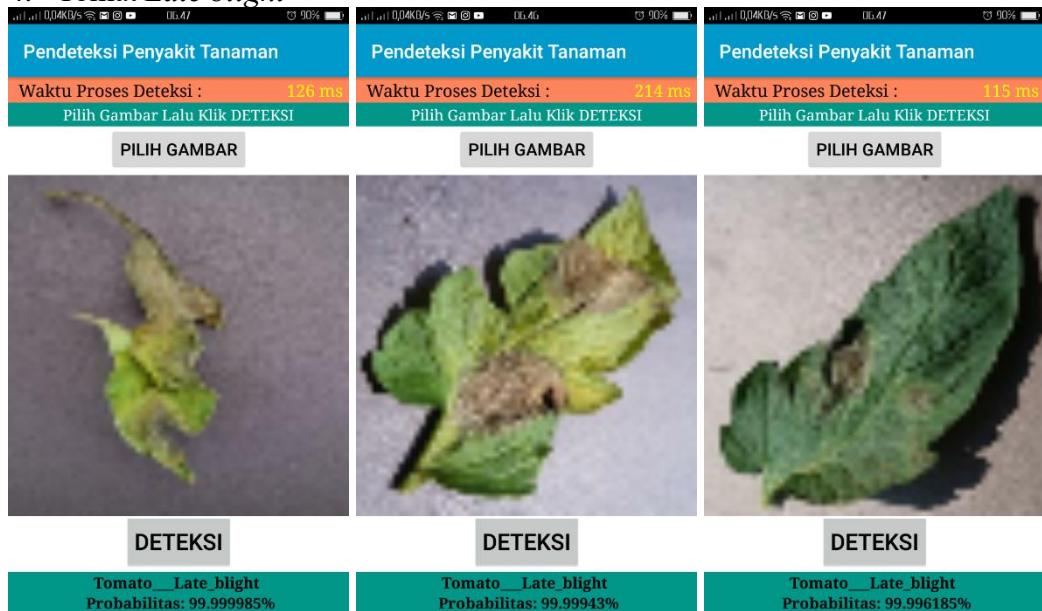
**2. Tomat Early blight**



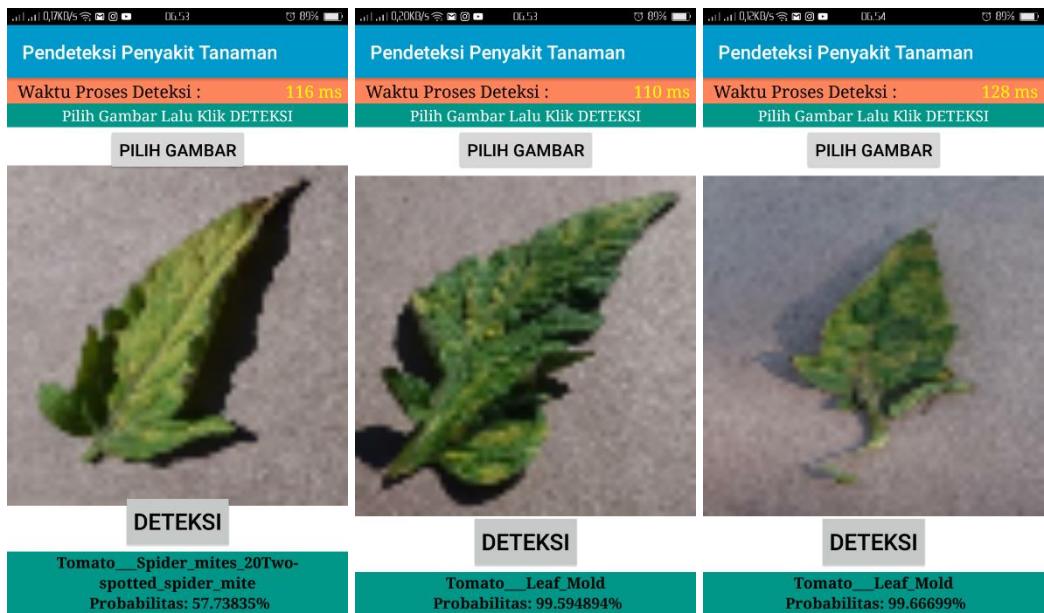
### 3. Tomat *healthy*



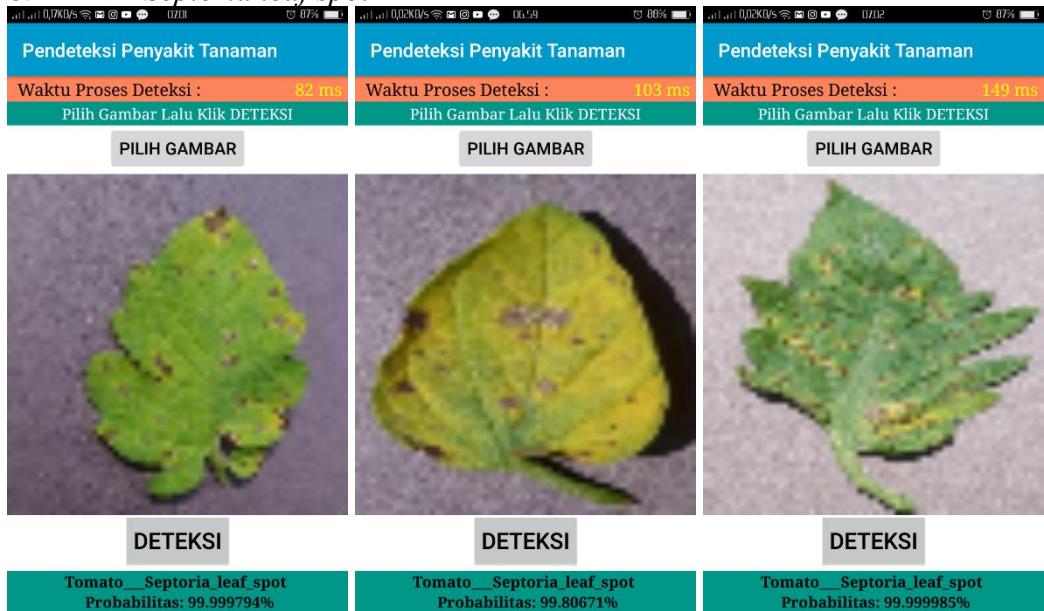
### 4. Tomat *Late blight*



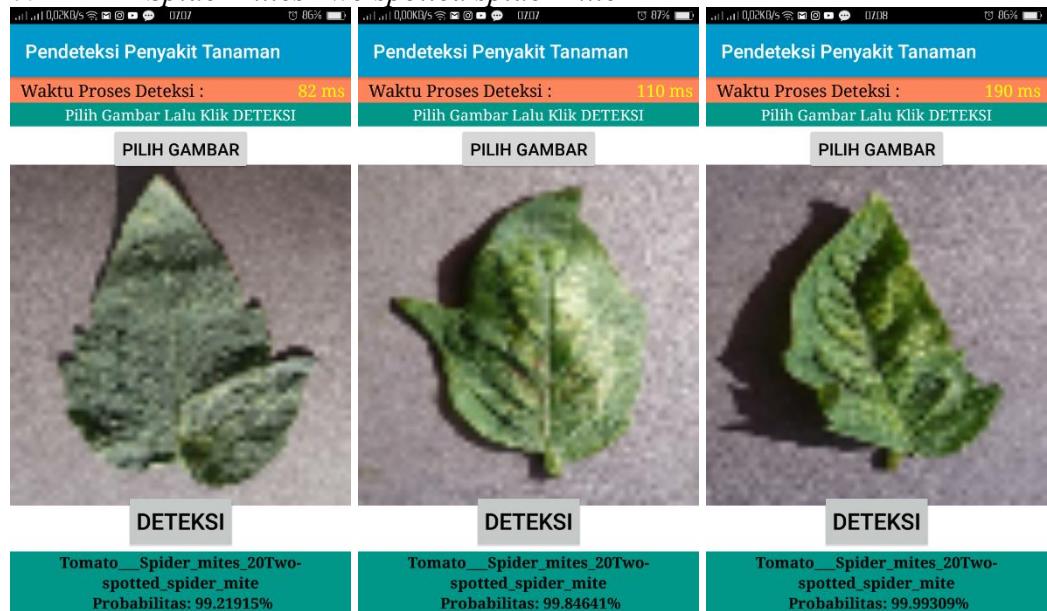
### 5. Tomat Leaf Mold



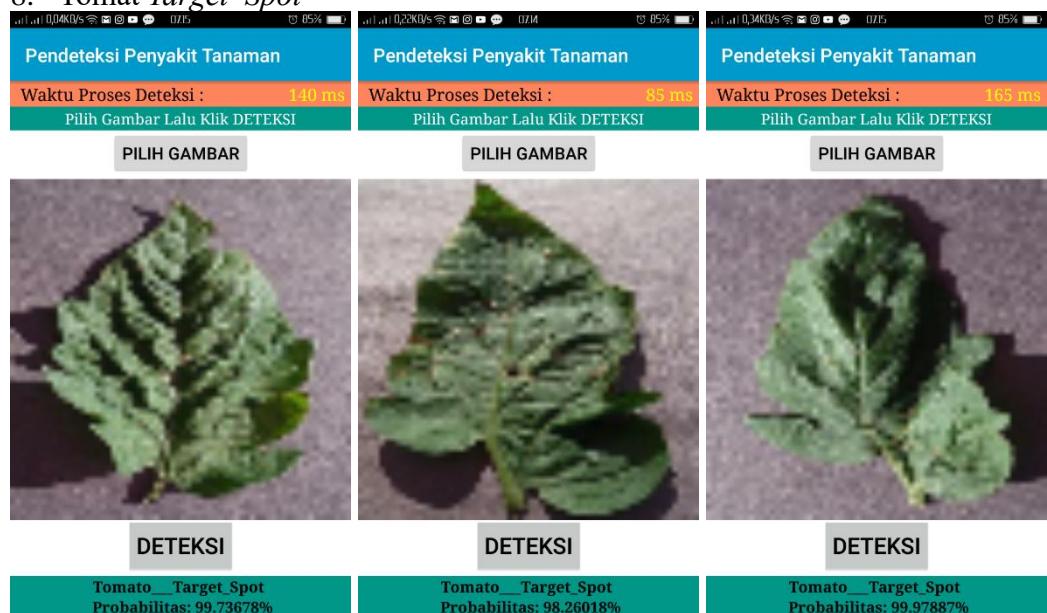
### 6. Tomat Septoria leaf spot



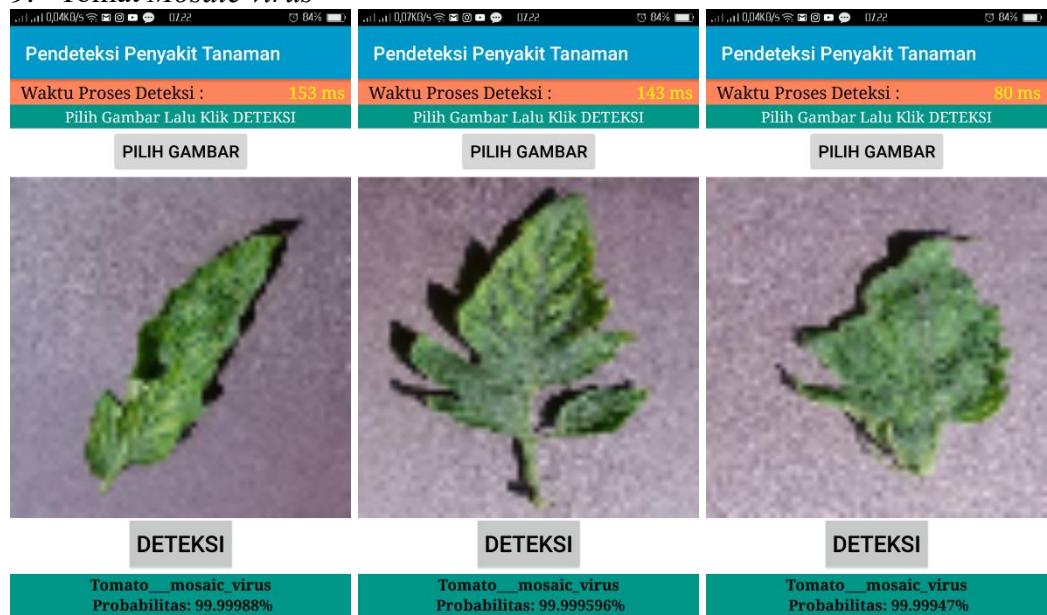
### 7. Tomat *Spider mites Two spotted spider mite*



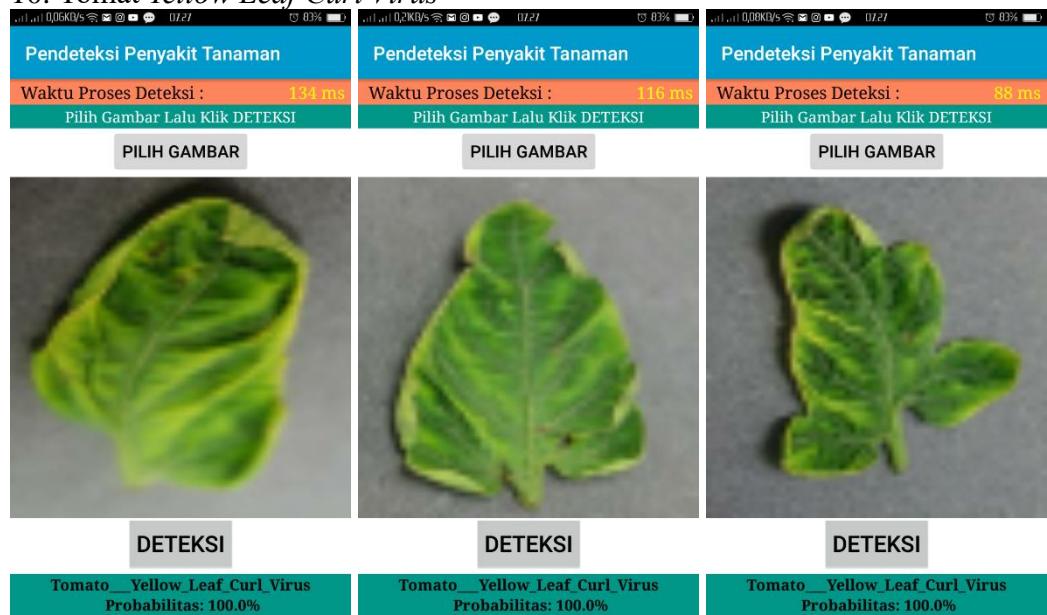
### 8. Tomat *Target Spot*



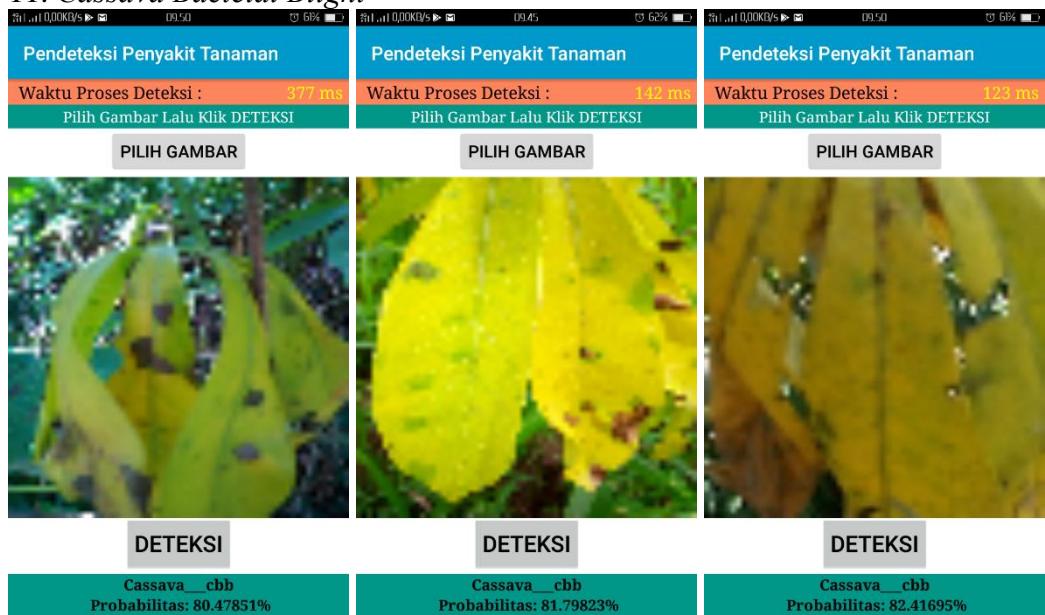
### 9. Tomat *Mosaic virus*



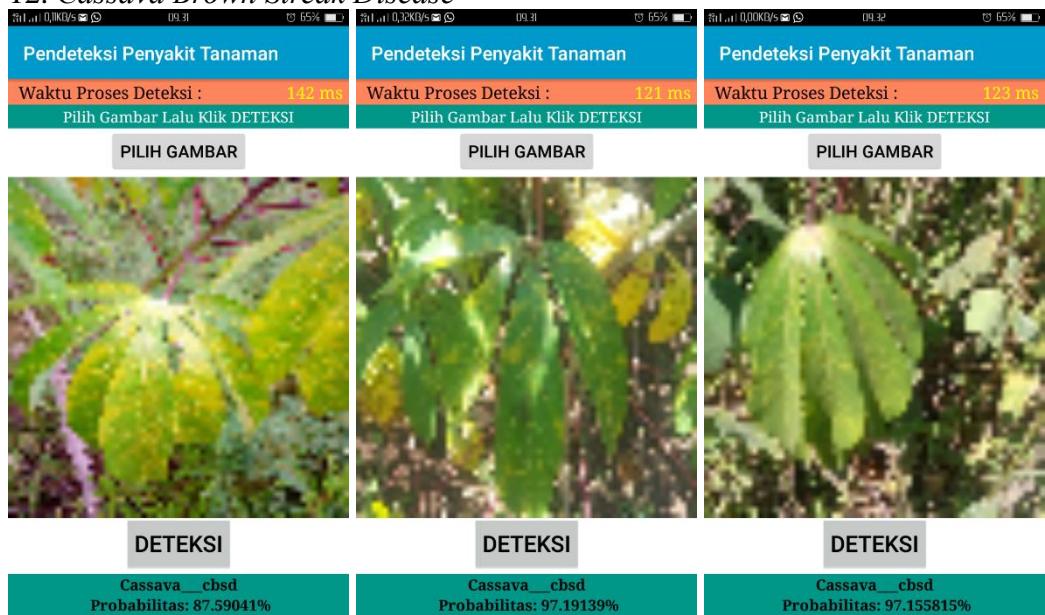
### 10. Tomat *Yellow Leaf Curl Virus*



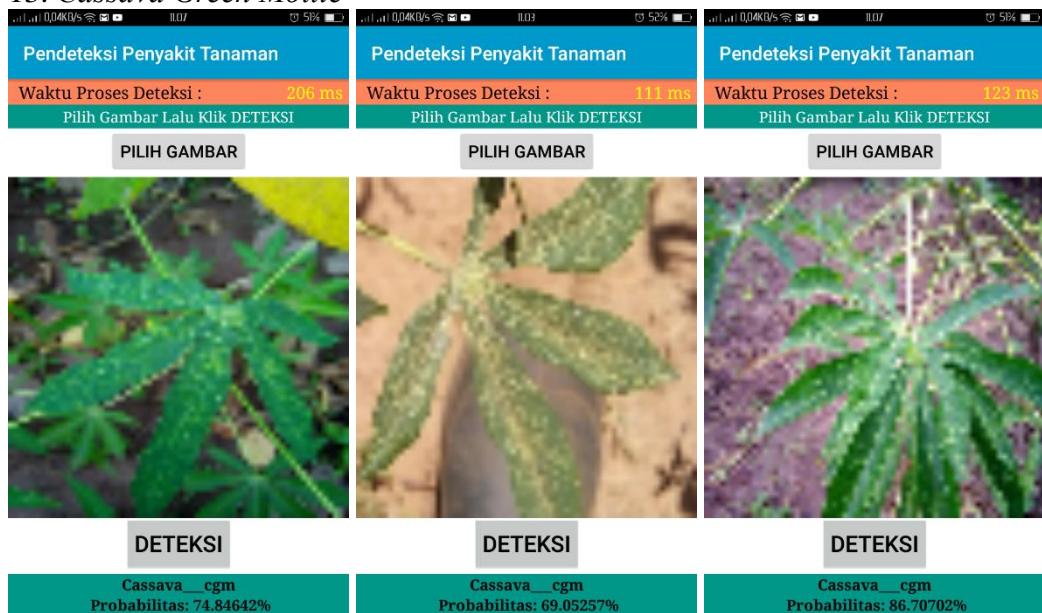
### 11. Cassava Bacterial Blight



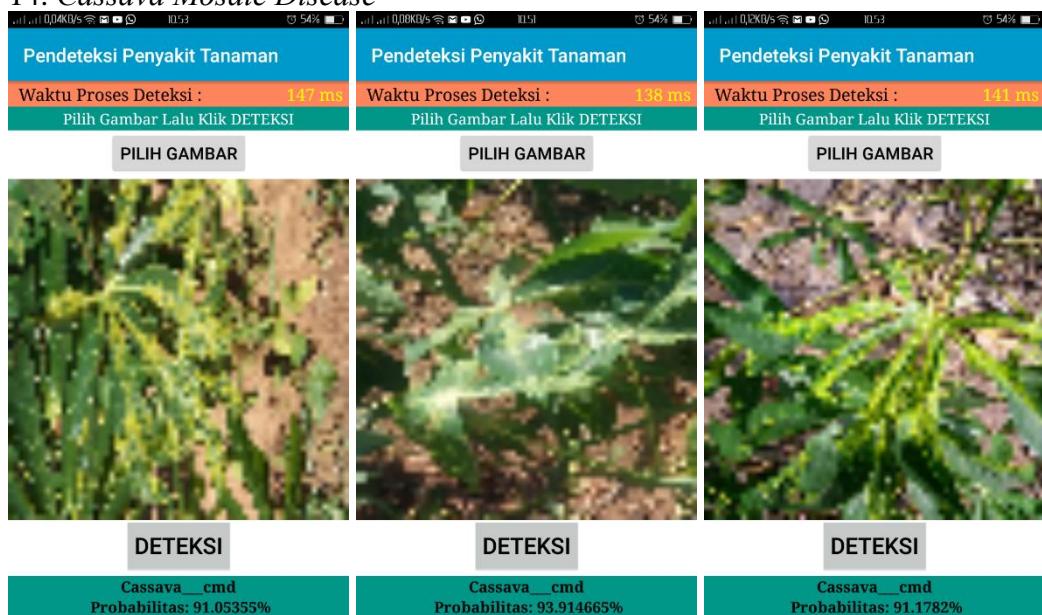
### 12. Cassava Brown Streak Disease

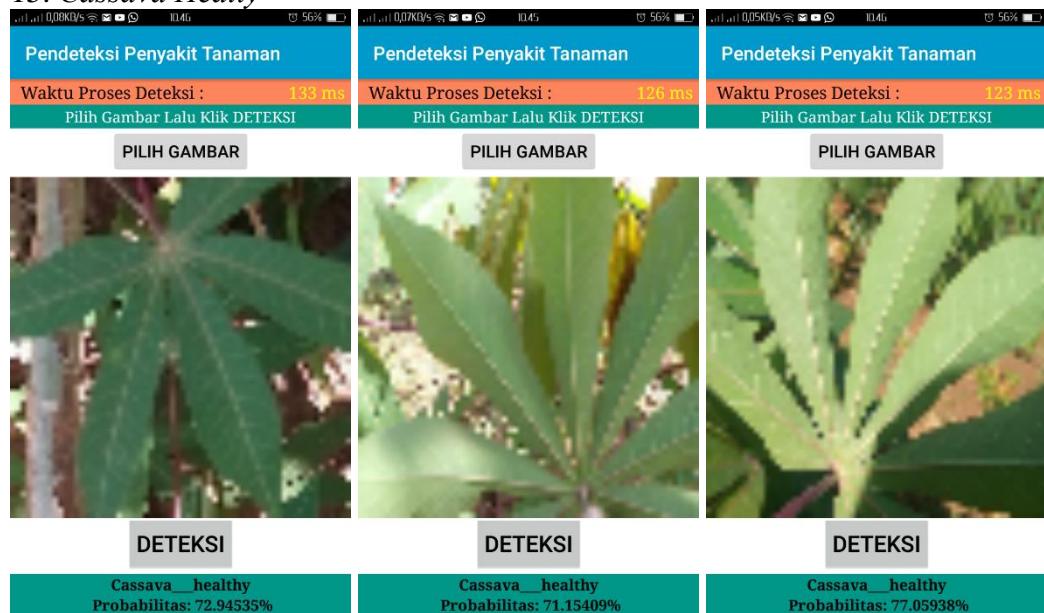


### 13. Cassava Green Mottle



### 14. Cassava Mosaic Disease



**15. Cassava Healthy**

## **BIODATA PENULIS**



### **A. Identitas**

Nama : Jepri  
 NIM : H1A016025  
 Tempat, tanggal lahir : Indramayu, 18 April 1997  
 Alamat : Ds. Rancahan Kec. Gabuswetan Kab. Indramayu, Jawa Barat  
 No. Telp. : 083823292238  
 Alamat e-mail : [jepri.tugas@gmail.com](mailto:jepri.tugas@gmail.com)

### **B. Riwayat Pendidikan Akademik**

Periode	Jenjang	Institusi
2016 – Sekarang	S1	Teknik Elektro Universitas Jenderal Soedirman
2013 – 2016	SMA	MAN Babakan Ciwaringin Cirebon
2010 – 2013	SMP	MTsN Babakan Ciwaringin Cirebon

### **C. Keahlian**

Memiliki minat di bidang pengembangan perangkat sistem informasi. Mampu membuat aplikasi berbasis web. Terlibat secara aktif dalam kegiatan asisten Laboratorium Sistem Telekomunikasi dan Informasi sebagai asisten praktikum Algoritma dan Struktur Data dan Dasar Pemrograman.