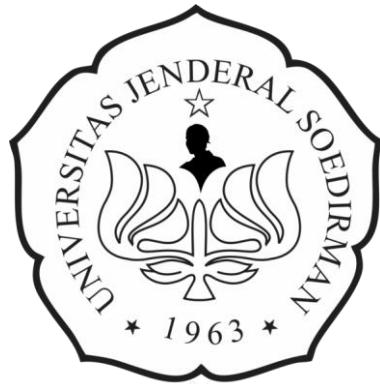


**RANCANG BANGUN KLASIFIKASI VARIETAS BERAS
BERDASARKAN CITRA MENGGUNAKAN METODE
*CONVOLUTIONAL NEURAL NETWORK (CNN) BERBASIS ANDROID***

LAPORAN TUGAS AKHIR

Disusun untuk memenuhi prasyarat memperoleh gelar Sarjana Teknik
di Jurusan Teknik Elektro Universitas Jenderal Soedirman



Disusun oleh:

Vidi Fitriansyah Hidarlan
H1A016042

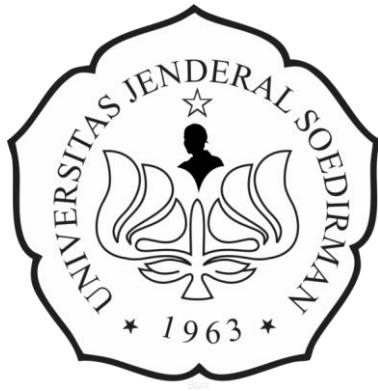
**KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN
UNIVERSITAS JENDERAL SOEDIRMAN
FAKULTAS TEKNIK
JURUSAN/PROGRAM STUDI TEKNIK ELEKTRO
PURBALINGGA
2020**



**RANCANG BANGUN KLASIFIKASI VARIETAS BERAS
BERDASARKAN CITRA MENGGUNAKAN METODE
CONVOLUTIONAL NEURAL NETWORK (CNN) BERBASIS ANDROID**

LAPORAN TUGAS AKHIR

Disusun untuk memenuhi prasyarat memperoleh gelar Sarjana Teknik
di Jurusan Teknik Elektro Universitas Jenderal Soedirman



Disusun oleh:

Vidi Fitriansyah Hidarlan
H1A016042

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

HALAMAN PENGESAHAN

Laporan Tugas Akhir dengan Judul:

RANCANG BANGUN KLASIFIKASI VARIETAS BERAS BERDASARKAN CITRA MENGGUNAKAN METODE *CONVOLUTIONAL NEURAL NETWORK (CNN) BERBASIS ANDROID*

Disusun oleh:

Vidi Fitriansyah Hidarlan
H1A016042

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/Lapangan

Imron Rosyadi, S.T., M.Sc.
(NIP : 197909242003121003)

Farida Asriani, S.Si., M.T.
(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 Akhir¹ dengan judul **“RANCANG BANGUN KLASIFIKASI VARIETAS BERAS BERDASARKAN CITRA 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, ... Februari 2020

[materai sesuai ketentuan uu]
Ttd.

Vidi Fitriansyah Hidarlan
NIM. H1A016042

¹ Hapus salah satu dan selanjutnya hapus nomor footnote

HALAMAN MOTTO DAN PERSEMBAHAN

MOTTO

“Belajarlah dari kesalahan-kesalahan yang telah dilakukan sebelumnya, karena dari kesalahan-kesalahan tersebut dapat memperkecil kesalahan-kesalahan berikutnya.”

PERSEMBAHAN

Penelitian ini dipersembahkan untuk:

1. Seluruh mahasiswa Teknik Elektro Unsoed,
2. Seluruh mahasiswa Teknik Unsoed, dan
3. Siapapun yang mungkin mendapatkan manfaat dari penelitian ini.

RINGKASAN

RANCANG BANGUN KLASIFIKASI VARIETAS BERAS BERDASARKAN CITRA MENGGUNAKAN METODE *CONVOLUTIONAL NEURAL NETWORK (CNN) BERBASIS ANDROID*

Vidi Fitriansyah Hidarlan

CNN memiliki berbagai macam arsitektur yang dapat digunakan untuk melakukan klasifikasi varietas beras berdasarkan citranya. Agar mempermudah pengguna untuk melakukan klasifikasi tersebut maka arsitektur CNN juga perlu diterapkan pada perangkat *Android* yang saat ini banyak digunakan.

Jenis arsitektur CNN yang digunakan pada klasifikasi varietas beras ini adalah *VGG16Net* dan *MobileNet* dengan menggunakan metode ekstraksi fitur (*feature extraction*). Kedua arsitektur tersebut digunakan untuk melakukan pelatihan dan pengujian pada infrastruktur *Google Colaboratory*. Dari data hasil pelatihan tersebut kemudian disimpan dan dikonversi ke dalam bentuk *file TensorFlow Lite* dan diimpor ke dalam *project* pada *Android Studio* agar dapat diimplementasikan pada aplikasi *Android*.

Berdasarkan hasil pelatihan dan pengujian pada *Google Colaboratory* dengan *dataset* citra varietas beras yang berukuran 224x224 piksel didapat bahwa pada arsitektur *VGG-16Net* dengan *dataset* kualitas baik diperoleh tingkat akurasi pelatihan sebesar 1.0 dan akurasi validasi sebesar 0.9556 sedangkan pada *MobileNet* tingkat akurasinya sama namun validasinya lebih rendah yaitu sebesar 0.9333. Pada *dataset* yang buruk dihasilkan nilai akurasi pelatihan dan validasi yang sama sebesar 1.0 pada arsitektur *VGG-16Net* akan tetapi pada *MobileNetV1* hanya mencapai 0.6889 akurasi validasinya. Selanjutnya dilakukan pengujian dari kedua arsitektur tersebut pada perangkat *Android*. Hasil pengujian arsitektur *VGG-16Net* pada kondisi beras disebar di alas hitam menggunakan *flashlight* pada perangkat *Android* menghasilkan tingkat akurasi sebesar 75,56% sedangkan pada *MobileNetV1* sebesar 95,56% karena model arsitektur *VGG-16Net* tidak cocok dikonversi ke *TensorFlow Lite*. Kemudian pengujian pada arsitektur *MobileNetV1* dilanjutkan menggunakan cahaya ruangan yang menghasilkan tingkat akurasi 71,11%. Selain itu dilakukan pengujian dengan kondisi beras yang dibungkus plastik bening baik menggunakan *flashlight* maupun cahaya ruangan. tingkat akurasi yang diperoleh pada kondisi beras dibungkus dan menggunakan cahaya *flashlight* sebesar 95,56% dan yang tanpa menggunakan *flashlight* sebesar 97,78%. Faktor yang mempengaruhi perbedaan hasil pengujian tersebut adalah jarak obyek yang dideteksi, kondisi pencahayaan, dan kualitas gambar.

Kata kunci : Klasifikasi varietas beras, *Android*, *Google Colaboratory*, CNN.

SUMMARY

DESIGN OF RICE VARIETIES CLASSIFICATION IMAGE-BASED USING CONVOLUTIONAL NEURAL NETWORK (CNN) METHOD ON ANDROID

Vidi Fitriansyah Hidarlan

CNN has a variety of architectures that can be used to classify rice varieties based on their image. To make it easier for users to do the classification, the CNN architecture also needs to be applied to Android devices that are currently widely used.

CNN architecture types used in the classification of rice varieties are VGG16Net and MobileNet using the feature extraction method. Both architectures are used for training and testing on the Google Collaboratory infrastructure. Then the training data is stored and converted into a TensorFlow Lite file and imported into a project in Android Studio so that it can be implemented in an Android application.

Based on the result of training and testing on Google Collaboration with the 224x224 pixel image datasets of rice variety were found that the VGG-16Net architecture with good quality image datasets obtained training accuracy levels of 1.0 and validation accuracy of 0.9556 whereas the accuracy level on MobileNet was the same but the validation accuracy was lower about 0.9333. In bad datasets the same accuracy and training results are obtained at 1.0 on the VGG-16Net architecture but on MobileNetV1 only reached the validation accuracy of 0.6889. Furthermore, to do the testing of these two architectures on Android devices. The results of the VGG-16Net architecture testing on the condition of rice being spread on a black mat using a flashlight on an Android device produce an accuracy rate of 75.56% while on MobileNetV1 of 95.56% because the VGG-16Net architectural model is not suitable converted to TensorFlow Lite. Then testing on the MobileNetV1 architecture was continued using room light which the result of the level of accuracy is 71.11%. Besides that, testing is done with the condition of rice wrapped in clear plastic using either a flashlight or room light. The level of accuracy obtained in the condition of rice wrapped and using a flashlight is 95.56% and without using a flashlight is 97.78%. Factors affecting the difference in the test results are the distance of the detected object, lighting condition, and image quality.

Keywords : Classification of rice varieties, Android, Google Colaboratory, CNN.

PRAKATA

Puji syukur kehadirat Allah S.W.T. yang telah melimpahkan berkah dan rahmat-Nya, sehingga penulis dapat menyelesaikan laporan tugas akhir ini dengan judul **“RANCANG BANGUN KLASIFIKASI VARIETAS BERAS BERDASARKAN CITRA MENGGUNAKAN METODE CONVOLUTIONAL NEURAL NETWORK (CNN) BERBASIS ANDROID”** ini dapat disusun. Penyusunan laporan tugas akhir ini telah melibatkan berbagai pihak, oleh karena itu penulis mengucapkan terima kasih kepada :

1. Allah Subhanahu Wata’ala yang senantiasa memberikan rahmat dan karunia-Nya.
2. Ibu, Bapak dan saudara-saudara penulis atas limpahan doa, dukungan dan semangat yang selalu mengalir.
3. Farida Asriani, S.Si., M.T. selaku kepala jurusan Teknik Elektro Universitas Jenderal Soedirman.
4. Hari Prasetijo, S.T., M.T. 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.

Penulis menyadari bahwa tugas akhir ini masih terdapat banyak kekurangan, tetapi penulis berharap semoga tugas ini dapat bermanfaat bagi semua pihak yang membutuhkan dan semoga yang disemogakan tersemogakan. Untuk itu penulis harapkan kritik dan sarannya.

Purbalingga, ... Februari 2020

Vidi Fitriansyah Hidarlan

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
DAFTAR SIMBOL.....	xvi
BAB 1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	2
1.3 Batasan Masalah.....	3
1.4 Tujuan dan Manfaat.....	3
1.4.1 Tujuan	3
1.4.2 Manfaat	4
1.5 Sistematika Penulisan.....	4
BAB 2 TINJAUAN PUSTAKA.....	5
2.1 Penelitian Terdahulu	5
2.2 <i>Deep Learning</i>	5
2.3 <i>Convolutional Neural Network (CNN)</i>	7
2.3.1 Konsep CNN.....	8
2.3.2 Arsitektur Jaringan CNN	10
2.3.3 Arsitektur <i>VGG16Net</i>	21
2.3.4 Arsitektur <i>MobileNet</i>	22
2.3.5 <i>Feature Extraction</i> (Ekstraksi Fitur).....	25
2.4 <i>Google Colaboratory</i>	28
2.5 <i>Keras</i> dan <i>TensorFlow</i>	29

2.6 <i>TensorFlow Lite</i>	31
2.7 <i>Android Studio</i>	33
2.7.1 Struktur Proyek	35
2.7.2 Tampilan Antarmuka	36
2.7.3 Sistem Pengembangan <i>Gradle (Gradle Build System)</i>	37
2.8 Beras	38
2.8.1 Beras IR 64	38
2.8.2 Beras Basmathi	39
2.8.3 Beras Ketan	40
BAB 3 METODOLOGI PENELITIAN	41
3.1 Waktu dan Tempat Penelitian	41
3.2 Alat dan Bahan	41
3.3 Metode Penelitian	42
3.3.1 Tahap Penyiapan dan <i>Pre-proses Dataset</i>	44
3.3.2 Tahap Desain Arsitektur	44
3.3.3 Tahap Pengujian	45
3.4 Waktu dan Jadwal Penelitian	46
BAB 4 HASIL DAN PEMBAHASAN	47
4.1 Perancangan Sistem dan <i>Dataset</i> Penelitian	47
4.1.1 <i>Dataset</i>	47
4.1.2 Pengambilan <i>Dataset</i>	48
4.1.3 Penyimpanan <i>Dataset</i>	49
4.1.4 Perancangan Program pada <i>Google Colaboratory</i>	50
4.1.5 Perancangan Program pada <i>Android Studio</i>	52
4.2 Pelatihan dan Pengujian pada <i>Google Colaboratory</i>	55
4.2.1 Varietas Beras	55
4.2.2 Variasi Kualitas Gambar <i>Dataset</i>	56
4.2.3 Arsitektur CNN yang digunakan	56
4.2.4 Hasil Pelatihan dan Pengujian pada <i>Google Colaboratory</i>	56
4.2.5 Kurva Perbandingan Hasil Pelatihan	73
4.3 Pengujian pada Perangkat <i>Android</i>	76
4.3.1 Hasil Pengujian pada Arsitektur <i>VGG-16Net</i> dengan <i>Flashlight</i>	76
4.3.2 Hasil Pengujian pada Arsitektur <i>MobileNetV1</i> dengan <i>Flashlight</i>	80
4.3.3 Hasil Pengujian pada Arsitektur <i>MobileNetV1</i> tanpa <i>Flashlight</i>	85
4.3.4 Hasil Pengujian pada Arsitektur <i>MobileNetV1</i> menggunakan <i>Flashlight</i> dengan Kondisi Beras dibungkus Plastik Bening	89
4.3.5 Hasil Pengujian pada Arsitektur <i>MobileNetV1</i> tanpa <i>Flashlight</i> dengan Kondisi Beras dibungkus Plastik Bening	92
BAB 5 PENUTUP	96
5.1 Kesimpulan	96
5.2 Saran	98

DAFTAR PUSTAKA	99
LAMPIRAN	101
Lampiran 1. Kode Sumber Klasifikasi Varietas Beras VGG-16Net.ipynb	101
Lampiran 2. Kode Sumber Klasifikasi Varietas Beras MobileNet.ipynb	113
Lampiran 3. Kode Sumber Klasifikasi Varietas Beras <i>Android Studio</i>	125
Lampiran 4. Sampel <i>Screenshot</i> Pengujian Arsitektur <i>VGG-16Net</i> dengan <i>Flashlight</i> pada Aplikasi <i>Android</i>	160
Lampiran 5. Sampel <i>Screenshot</i> Pengujian Arsitektur <i>MobileNetVI</i> dengan <i>Flashlight</i> pada Aplikasi <i>Android</i>	162
Lampiran 6. Sampel <i>Screenshot</i> Pengujian Arsitektur <i>MobileNetVI</i> tanpa <i>Flashlight</i> pada Aplikasi <i>Android</i>	164
Lampiran 7. Sampel <i>Screenshot</i> Pengujian Arsitektur <i>MobileNetVI</i> Menggunakan <i>Flashlight</i> dengan beras dibungkus pada Aplikasi <i>Android</i>	166
Lampiran 8. Sampel <i>Screenshot</i> Pengujian Arsitektur <i>MobileNetVI</i> tanpa <i>Flashlight</i> dengan beras dibungkus pada Aplikasi <i>Android</i>	168
BIODATA PENULIS	170

DAFTAR TABEL

Tabel 2.1 Arsitektur MobileNet.....	24
Tabel 3.1 Jadwal Penelitian.....	46
Tabel 4.1 Hasil pengujian arsitektur VGG-16Net dengan flashlight	77
Tabel 4.2 Hasil pengujian arsitektur MobileNetV1 dengan flashlight.....	80
Tabel 4.3 Hasil pengujian arsitektur MobileNetV1 tanpa flashlight.....	85
Tabel 4.4 Hasil pengujian arsitektur MobileNetV1 menggunakan flashlight pada beras dibungkus plastik bening	89
Tabel 4.5 Hasil pengujian arsitektur MobileNetV1 tanpa flashlight pada beras dibungkus plastik bening	92

DAFTAR GAMBAR

Gambar 2.1 Jaringan saraf mendalam untuk klasifikasi digit gambar	7
Gambar 2.2 Representasi mendalam yang dipelajari oleh model klasifikasi digit .	7
Gambar 2.3 Alur proses CNN dalam mengolah citra	8
Gambar 2.4 Arsitektur MLP Sederhana	9
Gambar 2.5 Proses konvolusi pada CNN.....	10
Gambar 2.6 Alur pada Convolution Layer.....	13
Gambar 2.7 Contoh input 7x7 filter 3x3 dengan stride 1	16
Gambar 2.8 Fungsi Aktivasi Rectified Linear Unit (ReLU)	17
Gambar 2.9 Matriks feature map 4x4 dengan proses pooling 2x2	19
Gambar 2.10 Contoh Fully Connected Layer	20
Gambar 2.11 Arsitektur VGG16	21
Gambar 2.12 Depthwise separable convolution.....	24
Gambar 2.13 Standard convolution (kiri), Depthwise separable convolution (kanan) dengan ReLU dan BN	25
Gambar 2.14 Pertukaran klasifikasi baru dengan mempertahankan basis konvolusi yang sama.....	27
Gambar 2.15 Antarmuka Google Colaboratory	28
Gambar 2.16 Minat pada pencarian Google untuk framework deep learning yang berbeda dari waktu ke waktu.....	30
Gambar 2.17 Susunan perangkat lunak dan perangkat keras deep learning	31
Gambar 2.18 Arsitektur TensorFlow Lite	32
Gambar 2.19 File proyek pada tampilan Android	35
Gambar 2.20 Jendela utama pada Android Studio	36
Gambar 2.21 Jenis beras IR 64	39
Gambar 2.22 Jenis Beras Basmathi.....	40
Gambar 2.23 Jenis beras ketan putih.....	40
Gambar 3.1 Desain arsitektur sistem	42
Gambar 3.2 Diagram alir sistem	43
Gambar 3.3 Diagram alir penelitian.....	44
Gambar 4.1 Tampilan direktori dataset train pada Github	49
Gambar 4.2 Tampilan direktori dataset test pada Github.....	50
Gambar 4.3 Tampilan program klasifikasi varietas beras pada Google Colaboratory.....	51
Gambar 4.4 Tampilan project aplikasi klasifikasi varietas beras di Android Studio	53
Gambar 4.5 File model (.tflite) beserta labelnya (.txt) pada folder assets	54
Gambar 4.6 Kurva tingkat akurasi hasil pelatihan dan validasi pada VGG-16Net57	57
Gambar 4.7 Kurva tingkat kesalahan pelatihan dan validasi pada VGG-16Net... ..	58
Gambar 4.8 Confusion matrix hasil pengujian pada VGG-16Net	60
Gambar 4.9 Kurva akurasi pelatihan dan validasi pada VGG-16Net pada gambar buruk	62
Gambar 4.10 Kurva kesalahan pelatihan dan validasi pada VGG-16Net pada gambar buruk	63

Gambar 4.11 Confusion matrix hasil pengujian pada VGG-16Net dengan gambar buruk	64
Gambar 4.12 Kurva akurasi hasil pelatihan dan validasi pada MobileNetV1	66
Gambar 4.13 Kurva tingkat kesalahan pelatihan dan validasi pada MobileNetV1	67
Gambar 4.14 Confusion matrix hasil pengujian pada MobileNetV1.....	68
Gambar 4.15 Kurva akurasi pelatihan dan validasi pada MobileNetV1 pada gambar buruk	70
Gambar 4.16 Kurva kesalahan pelatihan dan validasi pada MobileNetV1 pada gambar buruk	71
Gambar 4.17 Confusion matrix hasil pengujian pada MobileNetV1 dengan gambar buruk	72
Gambar 4.18 Kurva perbandingan tingkat akurasi dari hasil pelatihan	74
Gambar 4.19 Kurva perbandingan tingkat kesalahan dari hasil pelatihan	75
Gambar 4.20 Confusion matrix pengujian arsitektur VGG-16Net pada Android dengan flashlight	79
Gambar 4.21 Confusion matrix pengujian arsitektur MobileNetV1 pada Android dengan flashlight	84
Gambar 4.22 Confusion matrix pengujian arsitektur MobileNetV1 pada Android tanpa flashlight	88
Gambar 4.23 Confusion matrix pengujian arsitektur MobileNetV1 pada Android menggunakan flashlight dengan kondisi beras terbungkus.....	91
Gambar 4.24 Confusion matrix pengujian arsitektur MobileNetV1 pada Android tanpa flashlight dengan kondisi beras terbungkus	95

DAFTAR LAMPIRAN

Lampiran 1. Kode Sumber Klasifikasi Varietas Beras VGG-16Net.ipynb	101
Lampiran 2. Kode Sumber Klasifikasi Varietas Beras MobileNet.ipynb	113
Lampiran 3. Kode Sumber Klasifikasi Varietas Beras <i>Android Studio</i>	125
Lampiran 4. Sampel Screenshot Pengujian Arsitektur <i>VGG-16Net</i> dengan <i>Flashlight</i> pada Aplikasi <i>Android</i>	160
Lampiran 5. Sampel <i>Screenshot</i> Pengujian Arsitektur <i>MobileNetV1</i> dengan <i>Flashlight</i> pada Aplikasi <i>Android</i>	162
Lampiran 6. Sampel <i>Screenshot</i> Pengujian Arsitektur <i>MobileNetV1</i> tanpa <i>Flashlight</i> pada Aplikasi <i>Android</i>	164
Lampiran 7. Sampel <i>Screenshot</i> Pengujian Arsitektur <i>MobileNetV1</i> Menggunakan <i>Flashlight</i> dengan beras dibungkus pada Aplikasi <i>Android</i>	166
Lampiran 8. Sampel <i>Screenshot</i> Pengujian Arsitektur <i>MobileNetV1</i> tanpa <i>Flashlight</i> dengan beras dibungkus pada Aplikasi <i>Android</i>	168

DAFTAR ISTILAH DAN SINGKATAN

CNN : Convolutional Neural Network.

MLP : Multilayer Perceptron.

Python : Sebuah bahasa pemrograman tingkat tinggi yang bersifat *open source*.

JST : Jaringan Saraf Tiruan.

Overfitting : Kondisi model pembelajaran mesin sangat detail (berlebihan) yang dapat menurunkan tingkat akurasi.

Backpropagation : Pelatihan jenis terkontrol menggunakan pola penyesuaian bobot untuk mencapai nilai kesalahan yang minimum.

Dataset : Obyek yang merepresentasikan data dan relasinya di memori.

Hyperparameter : Parameter yang nilainya diatur sebelum proses pembelajaran.

ANN : *Artificial Neural Network*.

Jupyter Notebooks : Lingkungan pemrograman bahasa pemrograman *Python*.

CPU : *Central Processing Unit*.

GPU : *Graphical Processing Unit*.

API : *Application Programming Interface*.

MIT : *Massachusetts Institute of Technology*.

CNTK : *Microsoft Cognitive Toolkit*.

Java : Bahasa pemrograman yang multi platform dan multi perangkat.

SDK : *Android Software Development Kit*.

Gradle : *build tools* pada *Android Studio* yang digunakan untuk kompilasi dan menjalankan proyek aplikasi.

DAFTAR SIMBOL

W	:	Panjang atau tinggi <i>input</i>
F	:	Panjang atau tinggi <i>filter</i>
P	:	<i>Zero padding</i>
S	:	<i>Stride</i>
e	:	Bilangan <i>Euler</i> ($\approx 2,718281828459045235360287471352$)

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Beras adalah salah satu makanan pokok yang paling banyak dikonsumsi di Indonesia. Jenis beras yang paling banyak dikonsumsi oleh penduduk Indonesia adalah beras putih. Setiap varietas beras memiliki rasa, tekstur, manfaat, kandungan dan sifat unik tersendiri [1].

Tipe-tipe varietas beras dapat diklasifikasikan berdasarkan ukuran panjang dan bentuk, tekstur, dan warnanya. Klasifikasi varietas beras sangat penting karena setiap jenis beras memiliki kandungan nutrisi yang berbeda-beda. Teknik yang dilakukan untuk mengetahui varietas beras biasanya dilakukan dengan metode butiran atau dilihat dari morfologinya.

Deep learning adalah subbidang khusus dari pembelajaran mesin (*machine learning*) yang merupakan pandangan baru tentang representasi pembelajaran dari data yang menekankan pada lapisan-lapisan (*layers*) pembelajaran berturut turut dari representasi yang semakin informatif [2]. *Deep learning* banyak diimplementasikan untuk klasifikasi obyek berdasarkan citra atau gambar dengan cara mempelajari representasi atau fitur data secara otomatis seperti yang telah diterapkan pada identifikasi jenis tumbuhan dari citra daunnya. Model *deep learning* dapat mempelajari metode komputasinya sendiri, dengan 'otaknya' sendiri, apabila diibaratkan. Sebuah model *deep learning* dirancang untuk terus menganalisis data dengan struktur logika yang mirip dengan bagaimana manusia mengambil keputusan.

Convolutional Neural Network (CNN/ConvNet) adalah salah satu algoritma *deep learning* yang merupakan pengembangan dari *Multilayer Perceptron (MLP)* yang dirancang untuk mengolah data dalam bentuk dua dimensi, misalnya pada gambar [3]. CNN termasuk dalam jenis *Deep Neural Network* karena kedalaman jaringan yang tinggi dan banyak diaplikasikan pada data citra [4]. *CNN* dapat belajar langsung dari citra sehingga mengurangi beban dari pemrograman.

Seiring berkembangnya teknologi, *smartphone* dengan sistem operasi berbasis *Android* saat ini menjadi *smartphone* yang paling banyak dipakai sehingga metode *CNN* perlu diimplementasikan pada *smartphone Android*. Dengan menerapkan *CNN* pada *smartphone Android*, klasifikasi varietas beras menjadi lebih mudah dilakukan oleh sebagian besar pengguna hanya dengan menggunakan sebuah perangkat *Android*.

Oleh karena itu penting dilakukannya perancangan klasifikasi jenis beras menggunakan citra dengan perangkat *Android*. Sehingga pada tugas akhir ini, penulis mengambil bahan kajian tentang klasifikasi jenis beras dengan judul “RANCANG BANGUN KLASIFIKASI VARIETAS BERAS BERDASARKAN CITRA MENGGUNAKAN METODE CONVOLUTIONAL NEURAL NETWORK (*CNN*) BERBASIS *ANDROID*”.

1.2 Rumusan Masalah

Sesuai latar belakang tersebut, rumusan masalah dalam penelitian ini adalah sebagai berikut.

1. Bagaimana perancangan arsitektur CNN untuk klasifikasi varietas beras?

2. Bagaimana pelatihan dan pengujian untuk klasifikasi varietas beras dengan metode CNN?
3. Bagaimana pengujian klasifikasi varietas beras pada aplikasi *Android*?

1.3 Batasan Masalah

Batasan masalah untuk laporan ini adalah sebagai berikut.

1. Metode *deep learning* yang digunakan untuk klasifikasi varietas beras menggunakan *CNN (Convolutional Neural Network)*.
2. Program yang dibuat hanya untuk membedakan tiga varietas beras, yaitu varietas Beras Basmathi, Beras IR 64 dan Beras Ketan.
3. Program klasifikasi varietas beras dengan citra dibuat dengan Bahasa Pemrograman *Python* dengan antarmuka dan infrastruktur *Google Colaboratory*.
4. Pembuatan model *deep learning* pada klasifikasi varietas beras dengan citra menggunakan *Framework Keras dan TensorFlow*.
5. Aplikasi Android untuk deteksi varietas beras pada *smartphone* dibuat menggunakan *Android Studio*.
6. Penentuan jumlah *dataset* pelatihan, validasi serta pengujian tidak dibahas dalam penelitian ini.

1.4 Tujuan dan Manfaat

1.4.1 Tujuan

Tujuan pembuatan laporan tugas akhir ini adalah sebagai berikut.

1. Merancang arsitektur *CNN* untuk klasifikasi varietas beras dan penerapannya pada aplikasi *Android*.

2. Melatih arsitektur *CNN* untuk klasifikasi varietas beras.
3. Menguji arsitektur *CNN* klasifikasi varietas beras yang sudah dilatih sebelumnya.

1.4.2 Manfaat

Manfaat yang diharapkan dalam penyusunan tugas akhir ini adalah sebagai berikut.

1. Mampu menerapkan ilmu yang didapat pada mata kuliah yang bersangkutan untuk menyelesaikan tugas akhir.
2. Memudahkan dalam membedakan varietas beras dengan citra berdasarkan bentuk, ukuran, dan warnanya.

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

2.1 Penelitian Terdahulu

Dalam beberapa jurnal penelitian terhadulu 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. William Sugiarto, Yosi Kristian, dan Eka Rahayu Setyaningsih penelitiannya yang berjudul Estimasi Arah Tatapan Mata dengan Menggunakan *Average Pooling Convolutional Neural Network* membahas tentang pelacakan arah tatapan mata berfokus pada menginterpretasikan posisi mata pada layar yang nantinya akan diproses dan menghasilkan estimasi posisi dari apa yang dilihat oleh pengguna.
3. Sarirotul Ilahiyah dan Agung Nilogiri penelitiannya yang berjudul Implementasi *Deep Learning* Pada Identifikasi Jenis Tumbuhan Berdasarkan Citra Daun Menggunakan *Convolutional Neural Network* membahas tentang pengklasifikasian citra daun dari Neeraj Kumar yang mempunyai 20 jenis genus tumbuhan menggunakan arsitektur *AlexNet*.

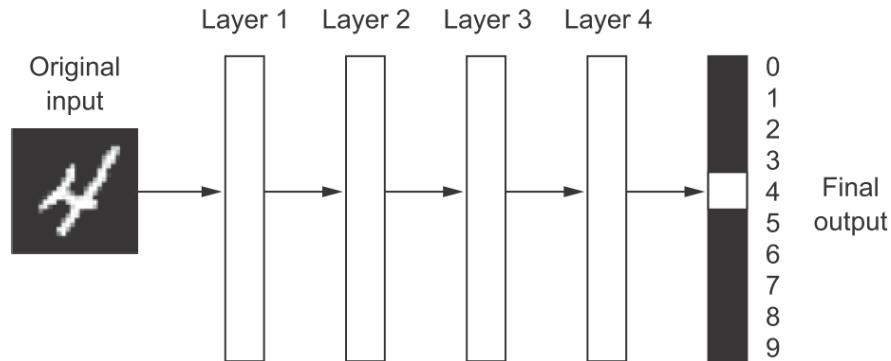
2.2 Deep Learning

Deep Learning atau pembelajaran mendalam adalah subbidang khusus dari pembelajaran mesin (*machine learning*) yaitu pandangan baru tentang

representasi pembelajaran dari data yang menekankan pada pembelajaran lapisan berturut-turut dari representasi yang semakin informatif [2]. Pendalaman dalam *Deep Learning* bukan merupakan referensi untuk segala jenis pemahaman yang lebih dalam yang dicapai oleh pendekatan, melainkan ia mewakili gagasan lapisan representasi yang berurutan. Banyak lapisan (*layer*) yang berkontribusi pada model data disebut kedalaman model. Nama-nama lain yang sesuai untuk bidang ini dapat berupa representasi pembelajaran berlapis dan pembelajaran representasi hierarkis. *Deep Learning* modern sering melibatkan puluhan atau bahkan ratusan lapisan representasi berturut-turut dan mereka semua belajar secara otomatis dari data pelatihan. Sementara itu, pendekatan lain untuk pembelajaran mesin cenderung fokus pada pembelajaran hanya satu atau dua lapisan representasi data. Oleh karena itu mereka kadang-kadang disebut pembelajaran dangkal.

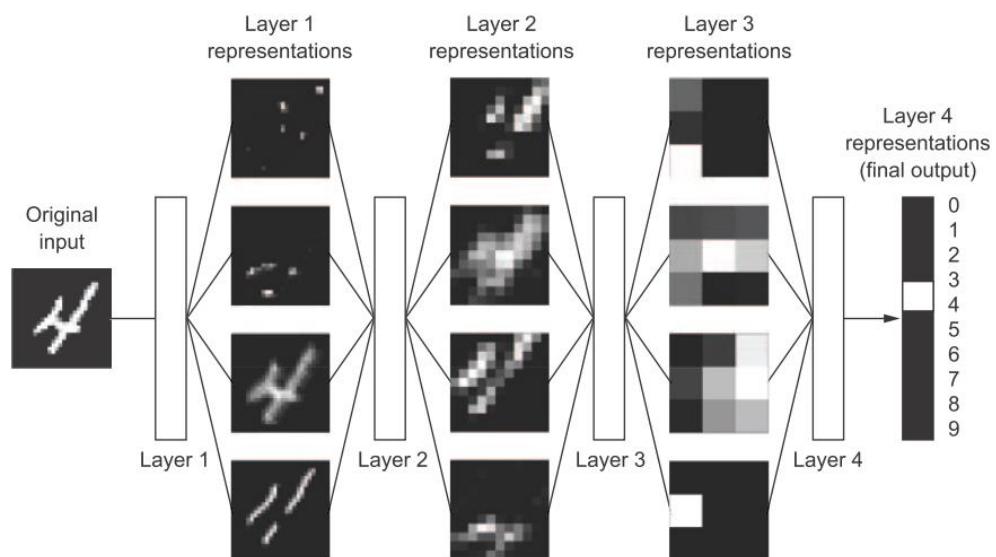
Dalam *deep learning*, representasi berlapis ini (hampir selalu) dipelajari melalui model yang disebut jaringan saraf, terstruktur dalam lapisan literal yang ditumpuk satu sama lain. Istilah jaringan saraf adalah referensi untuk neurobiologi, tetapi meskipun beberapa konsep dalam *deep learning* dikembangkan dalam bagian dengan menggambarkan inspirasi dari pemahaman kita tentang otak, model *deep learning* bukanlah model otak. Tidak ada bukti bahwa otak mengimplementasikan sesuatu seperti mekanisme pembelajaran yang digunakan dalam model *deep learning* modern.

Representasi yang dipelajari pada algoritma *deep learning* dapat dilihat bagaimana jaringan lapisan banyak mengubah gambar digit untuk mengenali digit apa itu seperti pada gambar 2.1.



Gambar 2.1 Jaringan saraf mendalam untuk klasifikasi digit gambar

Pada gambar 2.2, jaringan mengubah gambar digit menjadi representasi yang semakin berbeda dari gambar asli dan semakin informatif tentang hasil akhir. Kita dapat menganggap jaringan yang dalam sebagai operasi distilasi informasi yang bertingkat, di mana informasi melewati filter berturut-turut dan keluar semakin dimurnikan (yaitu, berguna berkaitan dengan beberapa tugas).



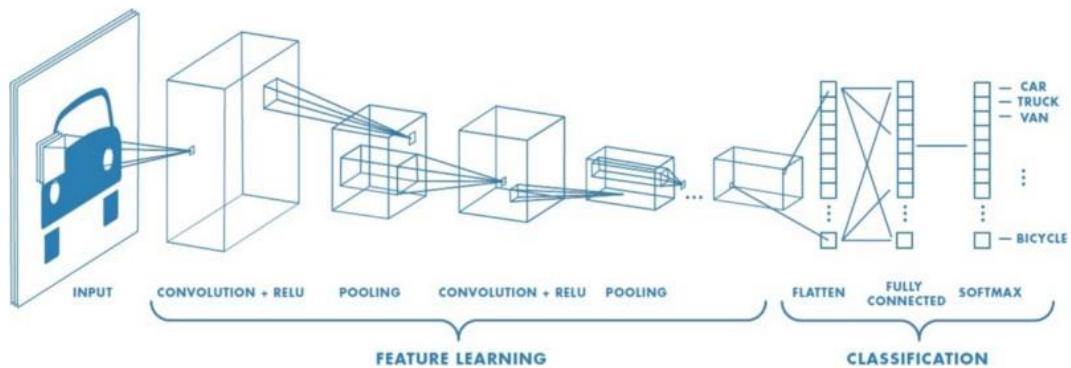
Gambar 2.2 Representasi mendalam yang dipelajari oleh model klasifikasi digit

2.3 Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) adalah pengembangan dari *Multilayer Perceptron* (MLP) yang didesain untuk mengolah data dua dimensi [4].

CNN termasuk dalam jenis *Deep Neural Network* karena kedalaman jaringan yang tinggi dan banyak diaplikasikan pada data citra. Pada klasifikasi citra, MLP kurang sesuai untuk digunakan karena tidak menyimpan informasi spasial dari data citra dan menganggap setiap piksel adalah fitur yang independen sehingga menghasilkan hasil yang kurang baik.

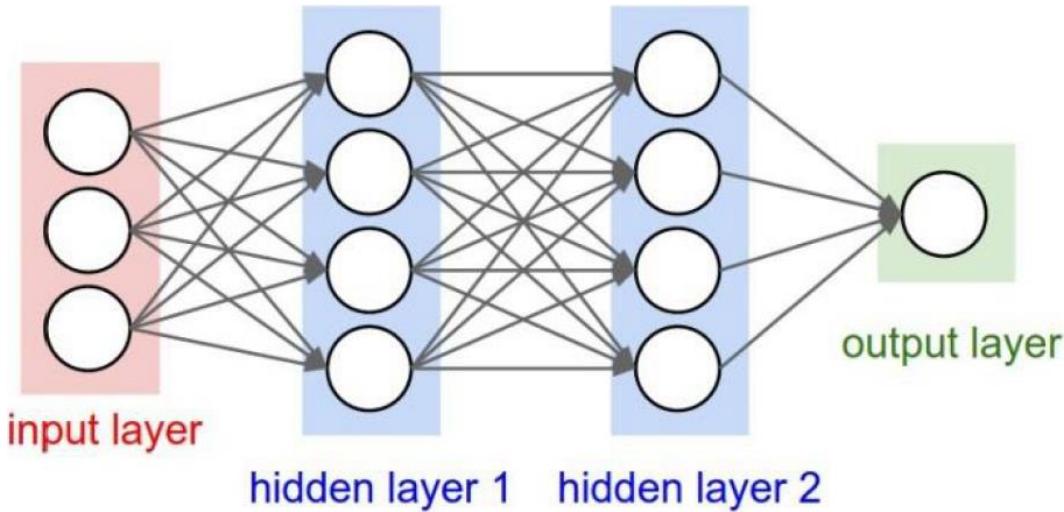
Pada gambar 2.3 bisa dilihat alur dari proses CNN dalam mengolah citra masukan sampai mengklasifikasikan citra tersebut ke kategori tertentu berdasarkan nilai keluarannya.



Gambar 2.3 Alur proses CNN dalam mengolah citra

2.3.1 Konsep CNN

Cara kerja CNN memiliki kesamaan pada MLP, namun dalam CNN setiap neuron dipresentasikan dalam bentuk dua dimensi, tidak seperti MLP yang setiap neuron hanya berukuran satu dimensi. Berikut gambar 2.3 merupakan arsitektur dari MLP.

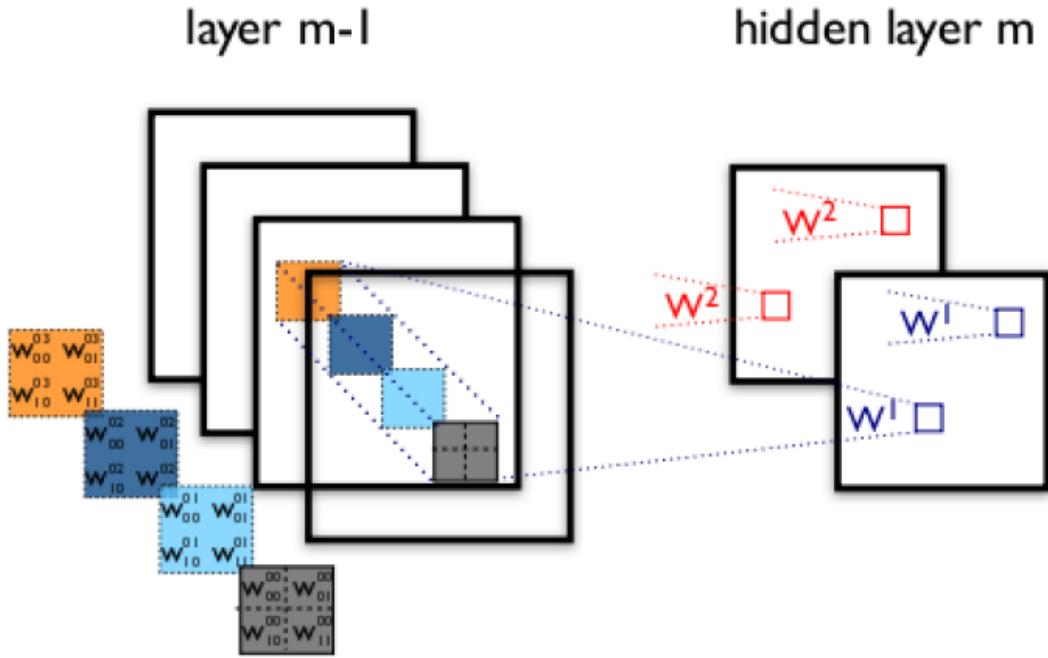


Gambar 2.4 Arsitektur MLP Sederhana

Sebuah MLP seperti pada Gambar 2.4 memiliki i layer (kotak merah dan biru) dengan masing-masing $layer$ berisi j_i neuron (lingkaran putih). MLP menerima input data satu dimensi dan mempropagasi data tersebut pada jaringan hingga menghasilkan output. Setiap hubungan antar neuron pada dua layer yang bersebelahan memiliki parameter bobot satu dimensi yang menentukan kualitas mode. Di setiap data masukkan pada $layer$ dilakukan operasi linear dengan nilai bobot yang ada, kemudian hasil komputasi akan ditransformasi menggunakan operasi non linear yang disebut sebagai fungsi aktivasi.

Pada CNN, data yang dipropagasi pada jaringan adalah data dua dimensi, sehingga operasi linear dan parameter bobot pada CNN berbeda. Pada CNN operasi linear menggunakan operasi konvolusi, sedangkan bobot tidak lagi satu dimensi saja, namun berbentuk empat dimensi yang merupakan kumpulan kernel konvolusi seperti pada gambar 2.5 dimensi bobot pada CNN adalah:

$$\text{neuron input} \times \text{neuron output} \times \text{tinggi} \times \text{lebar}$$



Gambar 2.5 Proses konvolusi pada CNN

Karena sifat proses konvolusi, maka CNN hanya dapat digunakan pada data yang memiliki struktur dua dimensi seperti citra dan suara.

2.3.2 Arsitektur Jaringan CNN

Jaringan Saraf Tiruan (JST) terdiri dari berbagai *layer* dan beberapa neuron pada masing-masing *layer*. Kedua hal tersebut tidak dapat ditentukan menggunakan aturan yang pasti dan berlaku berbeda-beda pada data yang berbeda [5].

Pada MLP, sebuah jaringan tanpa lapisan tersembunyi (*hidden layer*) dapat memetakan persamaan linear apapun, sedangkan jaringan dengan satu atau dua *hidden layer* dapat memetakan sebagian besar persamaan pada data sederhana. Namun pada data yang lebih kompleks, MLP memiliki keterbatasan. Pada permasalahan jumlah *hidden layer* di bawah tiga *layer*, terdapat pendekatan untuk menentukan jumlah neuron pada masing-masing *layer* untuk mendekati

hasil optimal. Penggunaan *layer* diatas dua pada umumnya tidak disarankan karena akan menyebabkan *overfitting* serta kekuatan *backpropagation* berkurang secara signifikan.

Seiring berkembangnya *deep learning*, ditemukan bahwa untuk mengatasi kekurangan MLP dalam menangani data kompleks, diperlukan fungsi untuk mentransformasi data input menjadi bentuk yang lebih mudah dimengerti oleh MLP. Hal tersebut memicu berkembangnya *deep learning* dimana dalam satu model diberi beberapa *layer* untuk melakukan transformasi data diolah menggunakan metode klasifikasi. Hal tersebut memicu berkembangnya model jaringan saraf dengan jumlah *layer* diatas tiga. Namun dikarenakan fungsi *layer* awal sebagai metode ekstraksi fitur, maka jumlah *layer* dalam sebuah CNN tidak memiliki aturan universal dan berlaku berbeda-beda tergantung *dataset* yang digunakan.

Karena hal tersebut, jumlah *layer* pada jaringan serta jumlah neuron pada masing-masing *layer* dianggap sebagai *hyperparameter* dan dioptimasi menggunakan pendekatan *searching*.

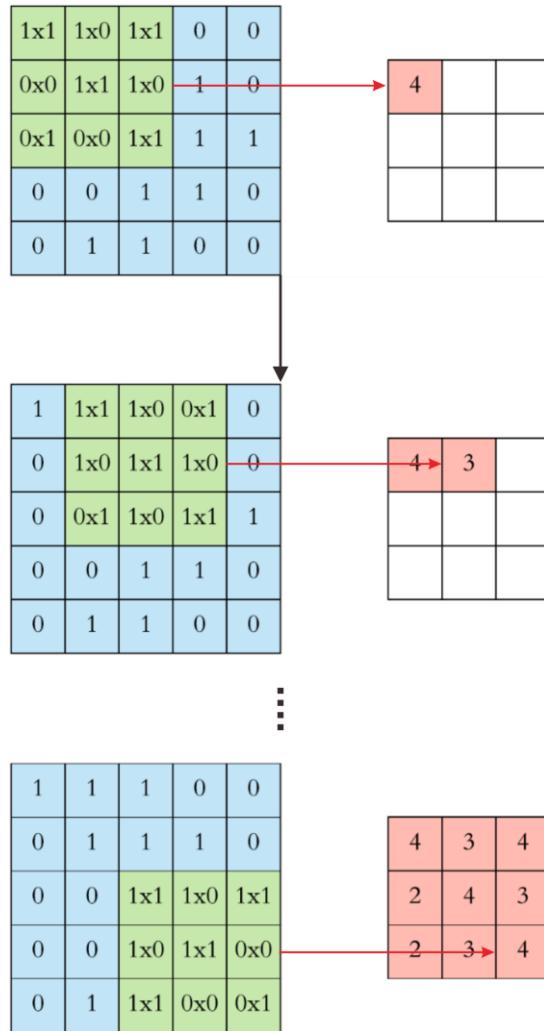
CNN memiliki lima komponen *layer* utama yaitu sebagai berikut:

1. Lapisan Masukkan (*Input Layer*)

Input layer menampung nilai piksel dari citra yang menjadi masukan [6]. Untuk citra dengan ukuran 64×64 dengan 3 *channel* warna, RGB (*Red*, *Green*, *Blue*) maka yang menjadi masukan akan adalah piksel *array* yang berukuran $64 \times 64 \times 3$.

2. Lapisan Konvolusi (*Convolution Layer*)

Convolution Layer adalah inti dari CNN [6]. *Convolution Layer* menghasilkan citra baru yang menunjukkan fitur dari citra masukkan. Dalam proses tersebut, *Convolution Layer* menggunakan *filter* pada setiap citra yang menjadi masukan. *Convolutional layer* terdiri dari neuron yang tersusun sedemikian rupa sehingga membentuk sebuah *filter* dengan panjang dan tinggi (piksel). Dalam CNN memiliki beberapa jumlah *filter* yang direpresentasikan sebagai kedalaman/jumlah dari *convolutional layer*. *Filter* ini akan digeser keseluruh bagian dari gambar. Setiap pergeseran akan dilakukan operasi “*dot*” antara masukkan (*input*) dan nilai dari *filter* tersebut sehingga menghasilkan sebuah keluaran (*output*) atau biasa disebut sebagai *activation map* atau *feature map*. Proses konvolusi dengan menggunakan *filter* pada *layer* ini akan menghasilkan *feature map* yang akan digunakan pada lapisan aktivasi (*activation layer*). Alur pada *convolution layer* disajikan pada gambar 2.6.



Gambar 2.6 Alur pada Convolution Layer

Dalam *convolution layer* ada tiga hyperparameter yang digunakan sebagai pengaturan ukuran volume keluaran neuron yaitu kedalaman (*depth*), langkah (*stride*), dan *zero-padding*.

a. *Depth*

Depth adalah *hyperparameter* volume keluaran yang sesuai dengan jumlah filter yang digunakan, masing-masing belajar mencari sesuatu yang berbeda dalam masukkan [7]. Misalnya, jika *convolution layer* pertama

mengambil masukkan gambar mentah, maka neuron yang berbeda di sepanjang dimensi kedalaman dapat diaktifkan dengan adanya berbagai tepi berorientasi, atau gumpalan warna. Kita akan merujuk pada satu set neuron yang semuanya melihat wilayah masukkan yang sama sebagai kolom kedalaman.

b. *Stride*

Stride adalah parameter yang menentukan berapa jumlah pergeseran filter [2]. Jika nilai *stride* adalah 1, maka filter konvolusi akan bergeser sebanyak 1 piksel secara horizontal lalu vertikal. Pada ilustrasi gambar 2.6, *stride* yang digunakan adalah 1.

Semakin kecil *stride* maka akan semakin detail informasi yang kita dapatkan dari sebuah masukkan, namun membutuhkan komputasi yang lebih jika dibandingkan dengan *stride* yang besar.

Namun perlu diperhatikan bahwa dengan menggunakan *stride* yang kecil kita tidak selalu akan mendapatkan performa yang bagus.

c. *Padding*

Padding atau *Zero Padding* adalah parameter yang menentukan jumlah piksel (berisi nilai 0) yang akan ditambahkan di setiap sisi dari masukkan [2]. Hal ini digunakan dengan tujuan untuk memanipulasi dimensi keluaran dari *conv. layer (Feature Map)*. Secara umum, pengaturan *zero padding* menjadi $P = (F-1)/2$ ketika *stride*-nya adalah $S = 1$ dengan P merupakan zero padding dan F merupakan ukuran matriks filternya. Pengaturan tersebut diperlukan untuk memastikan bahwa volume masukkan dan volume keluaran akan memiliki ukuran yang sama secara spasial.

Tujuan dari penggunaan *padding* yaitu sebagai berikut.

- 1) Dimensi keluaran dari *conv. layer* selalu lebih kecil dari masukkannya (kecuali penggunaan 1x1 filter dengan *stride* 1). Keluaran ini akan digunakan kembali sebagai masukkan dari *conv. layer* selanjutnya, sehingga semakin banyak informasi yang terbuang. Dengan menggunakan *padding*, kita dapat mengatur dimensi keluaran agar tetap sama seperti dimensi masukkan atau setidaknya tidak berkurang secara drastis. Sehingga kita bisa menggunakan *conv. layer* yang lebih dalam (*deep*) sehingga lebih banyak *features* yang berhasil diekstrak.
- 2) Meningkatkan performa dari model karena *conv. filter* akan fokus pada informasi yang sebenarnya yaitu yang berada di antara *zero padding* tersebut.

Untuk menghitung dimensi dari *feature map* kita dapat menggunakan persamaan seperti berikut:

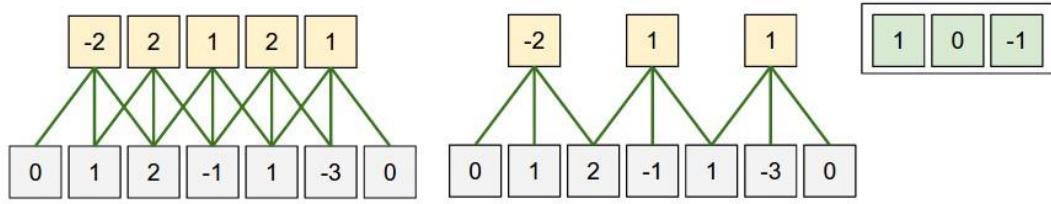
$$\text{Output} = \frac{W - F + 2P}{S} + 1 \quad (2.1)$$

dengan:

<i>W</i>	= Panjang/Tinggi Input
<i>F</i>	= Panjang/Tinggi Filter
<i>P</i>	= Zero Padding
<i>S</i>	= Stride

Pada *convolution layer* ada skema berbagi parameter (*parameter sharing*) yang digunakan untuk mengontrol jumlah parameter ketika melakukan konvolusi melewati input [7].

Seperti contoh berikut pada gambar 2.7, masukkan 7x7 dan filter 3x3 dengan langkah (*stride*) 1 dan *padding* 0 kita akan mendapatkan keluaran 5x5. Dengan langkah 2 kita akan mendapatkan keluaran 3x3.



Gambar 2.7 Contoh input 7×7 filter 3×3 dengan stride 1

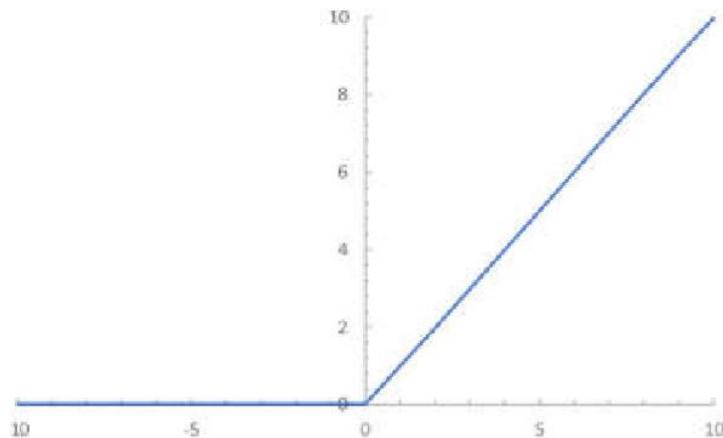
Dalam contoh ini hanya ada satu dimensi ruang (sumbu x), satu neuron dengan ukuran bidang reseptif $F = 3$, ukuran input adalah $W = 5$, dan ada *zero-padding* dari $P = 1$. Pada gambar sebelah kiri, neuron melangkah melintasi *input* dengan stride $S = 1$, memberikan *output* ukuran $(5-3+2)/1+1=5$. Sedangkan di sebelah kanannya, neuron menggunakan *stride* $S = 2$, memberikan output ukuran $(5-3+2)/2+1=3$. Filter/bobot neuron dalam contoh ini [1,0, -1] (ditampilkan di kanan), dan biasnya adalah nol. Perhatikan bahwa langkah $S = 3$ tidak dapat digunakan karena tidak sesuai dengan volumenya. Sesuai dengan rumus, hal itu dapat disimpulkan karena $(5-3+2) = 4$ tidak habis dibagi 3.

3. Lapisan aktivasi (*Activation Layer*)

Activation Layer adalah *layer* dimana *feature map* dimasukkan ke dalam fungsi aktivasi [6]. Fungsi aktivasi digunakan untuk mengubah nilai-nilai pada *feature map* pada jangkauan tertentu sesuai dengan fungsi aktivasi yang digunakan. ini bertujuan untuk meneruskan nilai yang menampilkan fitur dominan dari citra yang masuk ke *layer* berikutnya. Terdapat beberapa fungsi aktivasi yang sering digunakan, namun pada penelitian ini hanya menggunakan fungsi aktivasi *ReLU* dan *Softmax*.

a. Fungsi Aktivasi ReLU (*Rectified Linear Unit*)

ReLU merupakan fungsi aktivasi yang akan menghasilkan nilai nol apabila $x < 0$ dan kemudian linier dengan kemiringan 1 ketika $x > 0$ [8]. ReLU akan menghilangkan penghilangan/penyusutan gradien (*vanishing gradient*) dengan cara menerapkan fungsi aktivasi elemen sebagai $f(x) = \max[0, x]$ yaitu aktivasi elemen akan dilakukan saat berada di ambang batas 0. Berikut bentuk dari fungsi aktivasi ReLU disajikan pada gambar 2.8.



Gambar 2.8 Fungsi Aktivasi Rectified Linear Unit (ReLU)

Fungsi aktivasi ini memiliki kelebihan yaitu dapat mempercepat gradien stokastik dibandingkan dengan fungsi sigmoid/tan h karena ReLU berbentuk linear serta tidak menggunakan operasi eksponensial seperti sigmoid/tan h, sehingga bisa melakukan dengan pembuatan matriks aktivasi saat ambang batas berada pada nilai 0.

Namun terdapat kelemahan yang dapat rapuh saat masa training dan mati karena gradien besar yang mengalir melalui ReLU menyebabkan pembaruan bobot, sehingga neuron tidak aktif pada *datapoint* lagi. Jika ini terjadi, maka gradien yang mengalir melalui unit akan selamanya nol dari titik itu. Artinya, unit

ReLU dapat mati secara ireversibel (tidak dapat berubah) selama pelatihan karena mereka dapat melumpuhkan *data manifold*. Misalnya, Anda mungkin menemukan bahwa sebanyak 40% dari jaringan Anda dapat “mati” (yaitu neuron yang tidak pernah aktif di seluruh *dataset* pelatihan) jika tingkat pembelajaran ditetapkan terlalu tinggi. Dengan pengaturan tingkat pembelajaran yang tepat, ini lebih jarang menjadi masalah.

b. Fungsi Aktivasi *Softmax*

Fungsi *Softmax* menghitung probabilitas dari setiap kelas target atas semua kelas target yang memungkinkan dan akan membantu untuk menentukan kelas target untuk masukkan yang diberikan. Aktivasi *softmax* biasanya diterapkan pada lapisan terakhir pada *Convolutional Neural Network*. *Softmax* lebih sering digunakan daripada menggunakan ReLU, sigmoid, tanh, ataupun fungsi aktivasi lainnya. Persamaan dari fungsi aktivasi *softmax* yaitu.

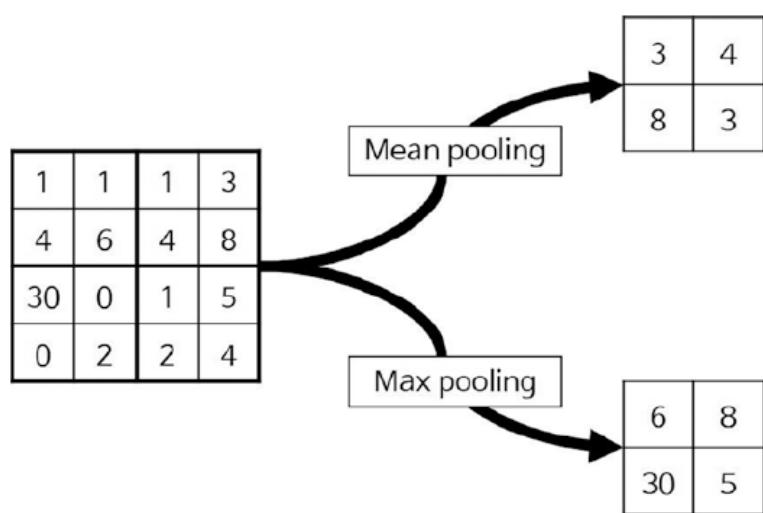
$$f_i(\vec{x}) = \frac{e^{x_i}}{\sum_{j=1}^k e^{x_j}} \text{ untuk } i = 1, 2, 3, \dots, k \quad (2.2)$$

Notasi f_i menunjukkan hasil fungsi untuk setiap elemen ke-i pada vektor keluaran kelas. Argumen x adalah hipotesis yang diberikan oleh model pelatihan agar dapat diklasifikasi oleh fungsi *Softmax*. Keuntungan utama menggunakan *Softmax* adalah rentang probabilitas keluaran dengan nilai 0 hingga 1, dan jumlah semua probabilitas akan sama dengan satu [9]. Jika fungsi *softmax* digunakan untuk model multi-klasifikasi, dia akan mengembalikan peluang dari masing-masing kelas dan kelas target akan memiliki probabilitas tinggi. *Softmax* menggunakan eksponensial (*e-power*) dari nilai masukkan yang diberikan dan

jumlah nilai eksponensial dari semua nilai dalam masukkan. Maka rasio eksponensial dari nilai masukkan dan jumlah nilai eksponensial adalah keluaran dari fungsi *softmax*.

4. Pooling Layer

Pooling layer menerima masukkan dari *activation layer* kemudian mengurangi jumlah paramaternya. *Pooling* juga biasa disebut *subsampling* atau *downsampling* yang mengurangi dimensi dari *feature map* tanpa menghilangkan informasi penting di dalamnya. Proses dalam *pooling layer* cukup sederhana. pertama-tama kita menentukan ukuran *downsampling* yang akan digunakan pada *feature map*. Setelah itu kita akan melakukan proses *pooling* pada *feature map*. *Pooling* sendiri ada beberapa macam seperti *Max pooling*, *Mean pooling*, dan *Sum pooling*. Berikut merupakan contoh proses *pooling* pada *feature map* ditampilkan pada gambar 2.9.



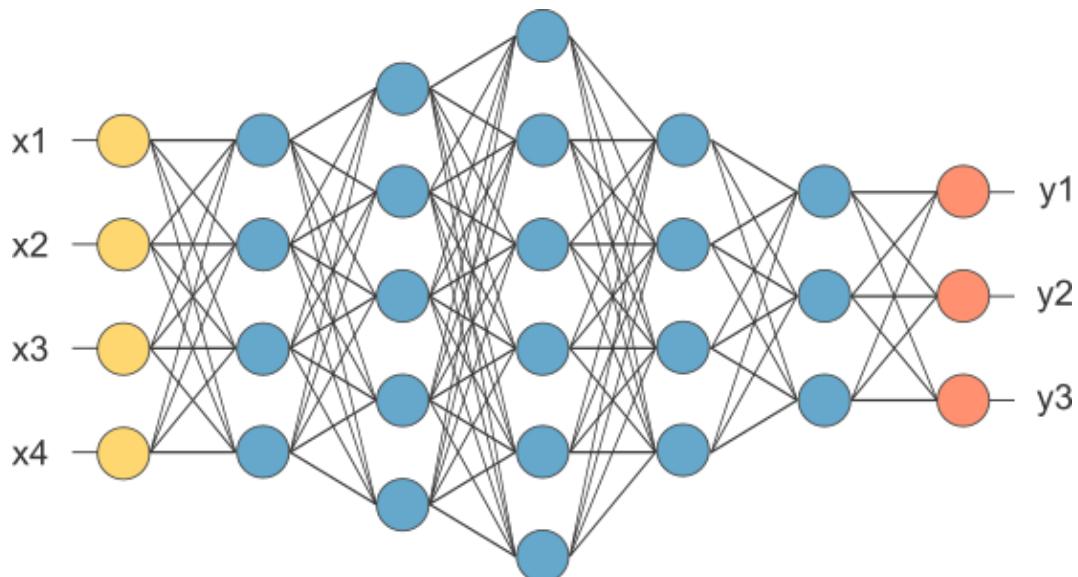
Gambar 2.9 Matriks *feature map* 4x4 dengan proses *pooling* 2x2

Dimensi keluaran dari *pooling layer* juga menggunakan rumus yang sama seperti pada *conv. ayer*. Tujuan dari penggunaan *pooling layer* adalah

mengurangi dimensi dari *feature map*, sehingga mempercepat komputasi karena parameter yang harus diperbarui semakin sedikit dan mengatasi *overfitting*.

5. Fully Connected Layer

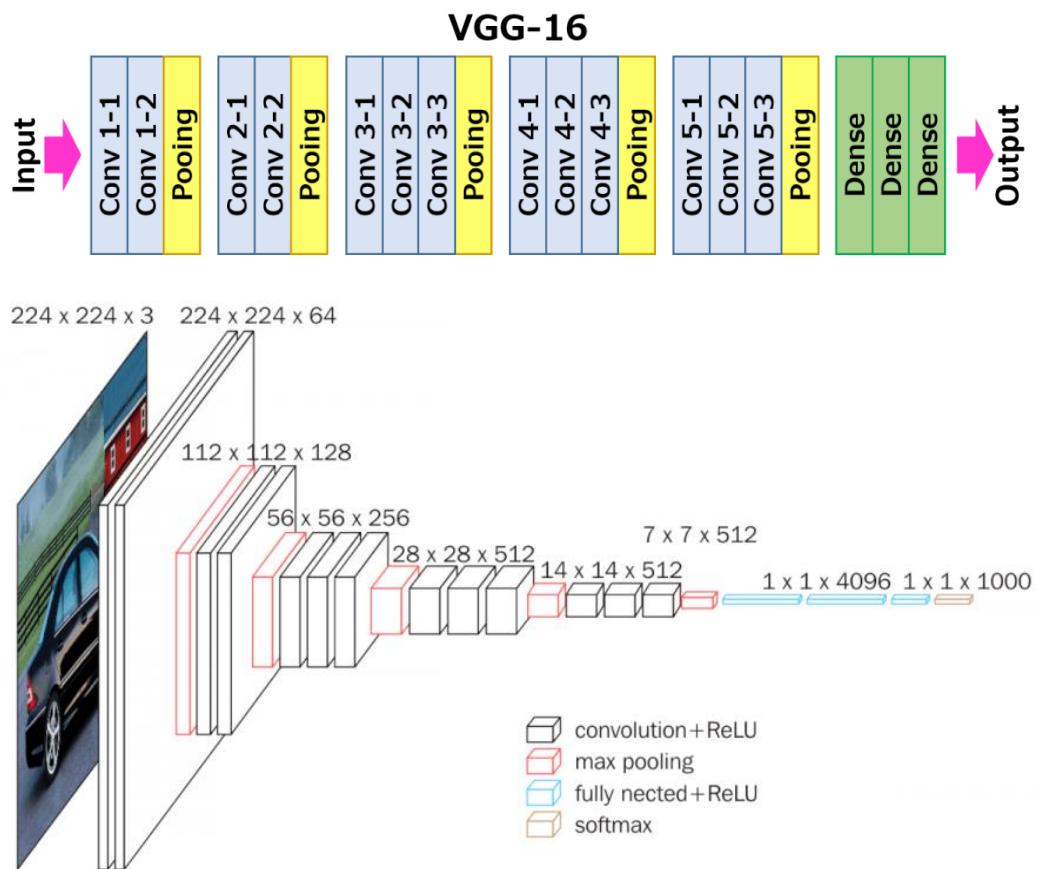
Setelah melewati proses-proses diatas, hasil dari *pooling layer* digunakan menjadi masukan untuk *Fully connected layer*. Namun, sebelum masuk ke *layer* ini terdapat lapisan *flatten* yang berfungsi untuk membentuk ulang *feature map* dari lapisan-lapisan yang sudah diekstrasi fitur sebelumnya menjadi sebuah vektor agar dapat digunakan sebagai masukkan dari *fully-connected layer*. *Fully Connected Layer* ini memiliki kesamaan struktur dengan *Artificial Neural Network* (ANN) pada umumnya yaitu memiliki *layer* masukkan, *hidden layer* dan *layer* keluaran yang masing-masing memiliki neuron-neuron yang saling terhubung dengan neuron-neuron di *layer* tetangganya. Gambar 2.10 ini merupakan contoh *Fully Connected Layer*.



Gambar 2.10 Contoh Fully Connected Layer

2.3.3 Arsitektur VGG16Net

VGG16 adalah model jaringan saraf konvolusi 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. Berikut arsitektur VGG16 pada gambar 2.11.



Gambar 2.11 Arsitektur VGG16

Masukan ke lapisan *conv1* berukuran tetap 224×224 gambar RGB. Gambar dilewatkan melalui tumpukan lapisan konvolusional (*conv.*), dimana *filter*

digunakan dengan bidang reseptif yang sangat kecil sebesar 3×3 . Dalam salah satu konfigurasi, ia juga menggunakan *filter* konvolusi 1×1 , 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 3×3 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 2×2 , dan 2 *stride*.

Tiga lapisan *Fully-Connected* (FC) mengikuti tumpukan lapisan konvolusional (yang memiliki kedalaman berbeda dalam arsitektur yang berbeda), yaitu 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.

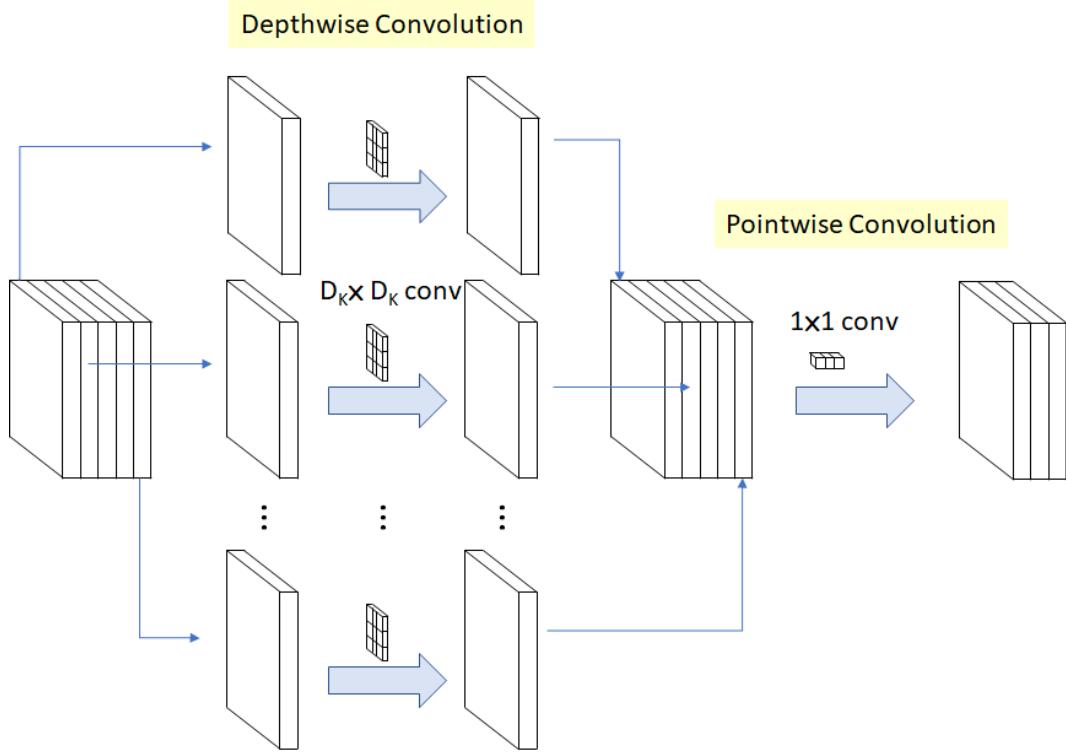
2.3.4 Arsitektur *MobileNet*

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 1×1 yang disebut konvolusi searah (*pointwise convolution*) [12]. Untuk *MobileNets*, *depthwise convolution* menerapkan satu *filter* ke setiap saluran masukkannya. *Pointwise convolution* kemudian menerapkan konvolusi 1×1 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. Arsitektur dari *depthwise separable convolution* dapat dilihat pada gambar 2.12.



Gambar 2.12 Depthwise separable convolution

2. Struktur Jaringan

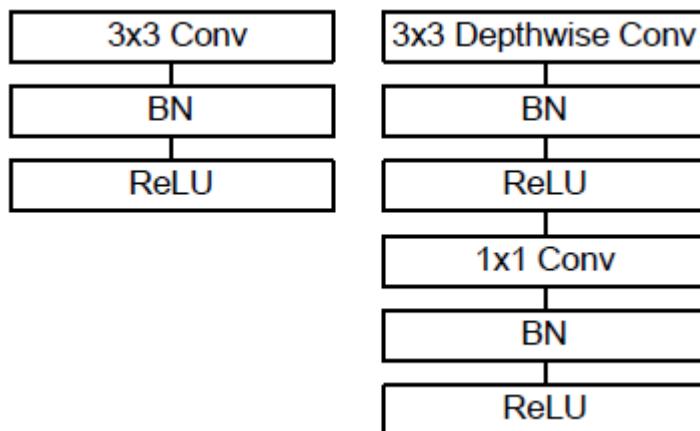
Arsitektur *MobileNet* didefinisikan dalam Tabel 2.1. Semua lapisan diikuti oleh *batchnorm* (*backnormalization*) dan ReLU dengan pengecualian lapisan akhir yang terhubung penuh yang tidak memiliki nonlinier dan dimasukkan ke dalam lapisan *softmax* untuk klasifikasi.

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	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$

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	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
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 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
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 di setiap lapisan konvolusinya seperti pada gambar 2.13 berikut.



Gambar 2.13 Standard convolution (kiri), Depthwise separable convolution (kanan) dengan ReLU dan BN

2.3.5 Feature Extraction (Ekstraksi Fitur)

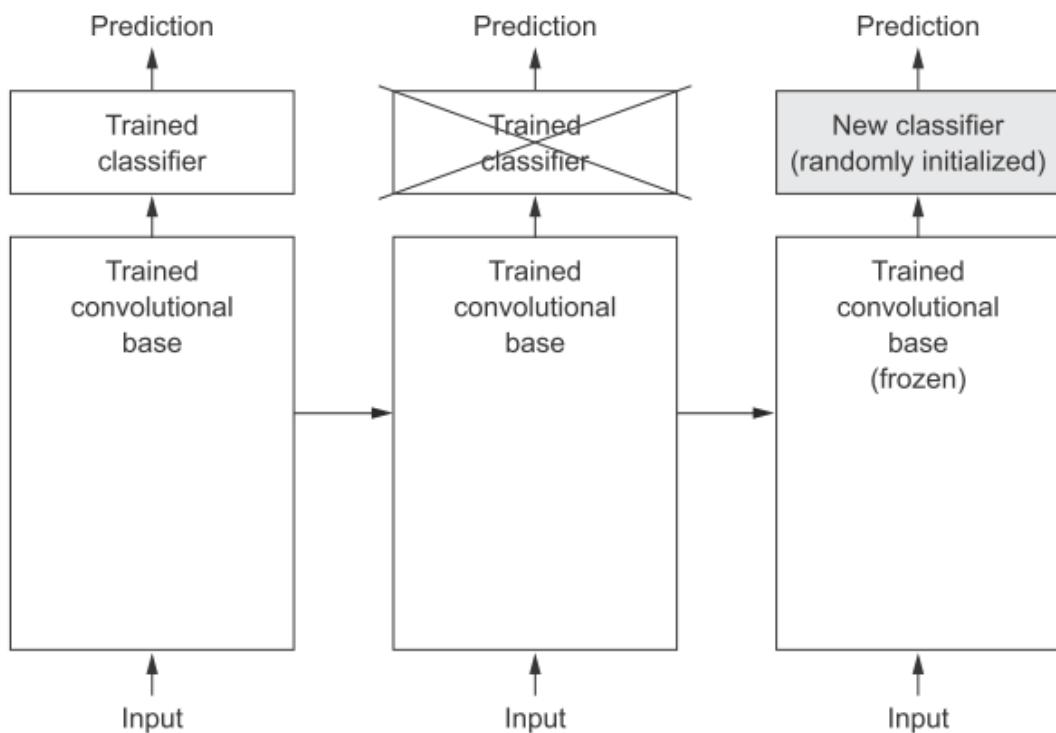
Pendekatan umum dan sangat efektif untuk *deep learning* pada dataset gambar dalam jumlah sedikit yaitu menggunakan jaringan yang sudah dilatih

sebelumnya (*pretrained network*). Jaringan *pretrained* adalah jaringan yang disimpan yang sebelumnya dilatih pada *dataset* yang besar, biasanya pada tugas klasifikasi gambar skala besar [2]. Jika dataset asli ini cukup besar dan cukup umum, maka hierarki spasial dari fitur yang dipelajari oleh jaringan yang belum dilatih dapat secara efektif bertindak sebagai model generik dari dunia visual, dan karenanya fitur-fiturnya dapat terbukti berguna untuk banyak masalah komputer visual, meskipun masalah-masalah baru ini mungkin melibatkan kelas-kelas yang benar-benar berbeda dari yang ada di tugas aslinya. Misalnya, digunakan untuk melatih jaringan di *ImageNet* (di mana kelas sebagian besar adalah binatang dan benda di kehidupan sehari-hari) dan kemudian menggunakan kembali jaringan yang terlatih ini untuk sesuatu yang jauh seperti mengidentifikasi item furnitur dalam gambar. Portabilitas fitur-fitur yang dipelajari di berbagai masalah yang berbeda merupakan keuntungan utama dari *deep learning* dibandingkan dengan banyak pendekatan yang lebih lama dan pembelajaran yang dangkal, dan itu membuat *deep learning* sangat efektif untuk masalah data kecil.

Feature extraction terdiri dari penggunaan representasi yang dipelajari oleh jaringan sebelumnya untuk mengekstraksi fitur menarik dari sampel baru [2]. Fitur-fitur ini kemudian dijalankan melalui *classifier* baru, yang dilatih dari awal.

CNN yang digunakan untuk klasifikasi gambar terdiri dari dua bagian yaitu dimulai dengan serangkaian *pooling layer* dan *convolution layer*, dan berakhir dengan pengklasifikasi yang terhubung erat (*densely connected classifier*). Bagian pertama disebut basis konvolusional (*convolutional base*) dari model. Dalam CNN, ekstraksi fitur mengambil basis konvolusional dari jaringan

yang sebelumnya dilatih, menjalankan data baru melaluinya, dan melatih pengklasifikasian baru di atas *output*-nya seperti yang digambarkan pada gambar 2.14.



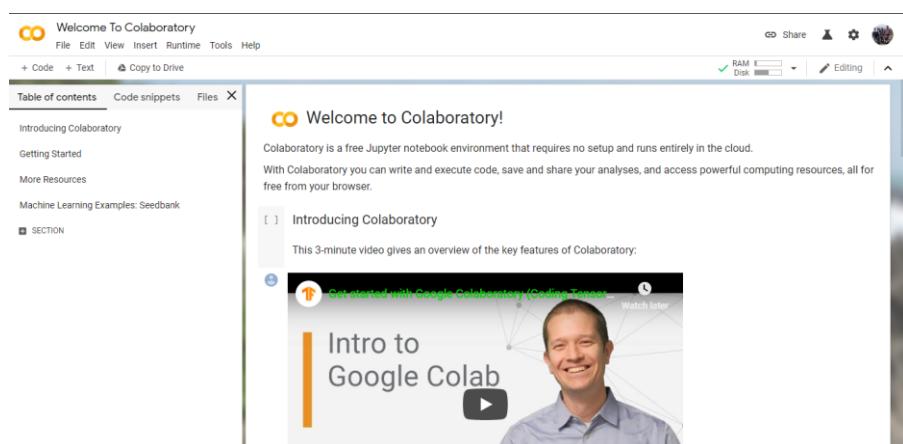
Gambar 2.14 Pertukaran klasifikasi baru dengan mempertahankan basis konvolusi yang sama

Dari gambar tersebut dapat dilihat bahwa hanya menggunakan kembali *convolutional base* dan tidak menggunakan *densely connected classifier*. Hal tersebut dikarenakan representasi yang dipelajari oleh *convolutional base* cenderung lebih generik dan oleh karena itu lebih dapat digunakan kembali. Peta fitur (*feature map*) dari sebuah CNN adalah peta keberadaan konsep-konsep umum pada gambar, yang mungkin berguna terlepas dari masalah visual komputer. Tetapi representasi yang dipelajari oleh pengklasifikasi tentu akan spesifik untuk *set* kelas di mana model dilatih. Mereka hanya akan berisi

informasi tentang kemungkinan kehadiran kelas ini atau itu di seluruh gambar. Selain itu, representasi yang ditemukan di lapisan yang terhubung erat tidak lagi berisi informasi tentang di mana objek berada pada gambar yang dimasukkan. Lapisan ini menghilangkan gagasan ruang, sedangkan lokasi obyek masih dapat digambarkan oleh *convolutional feature map*. Untuk masalah di mana lokasi obyek penting, fitur-fitur yang terhubung secara rapat sebagian besar tidak berguna.

2.4 Google Colaboratory

Google Colaboratory, kadang-kadang disebut *Colaboratory*, adalah layanan berbasis *cloud Google* yang mereplikasi *Jupyter Notebook* di-*cloud* [13]. Dengan menggunakan *Google Colaboratory* tidak perlu menginstal apa pun di sistem untuk menggunakannya. Dalam sebagian besar hal, penggunaan *Colaboratory* hampir sama seperti yang dilakukan pada instalasi *desktop Jupyter Notebook*. *Google Colaboratory* ditujukan untuk para pembaca yang menggunakan sesuatu selain dari pengaturan *desktop* standar untuk mengerjakan contoh-contoh. Berikut tampilan antarmukanya pada gambar 2.15.



Gambar 2.15 Antarmuka Google Colaboratory

Dalam menggunakan *Colaboratory*, diwajibkan memiliki akun *Google* untuk mengakses *Colaboratory* agar semua fitur yang ada pada *Colaboratory* dapat berfungsi dengan baik.

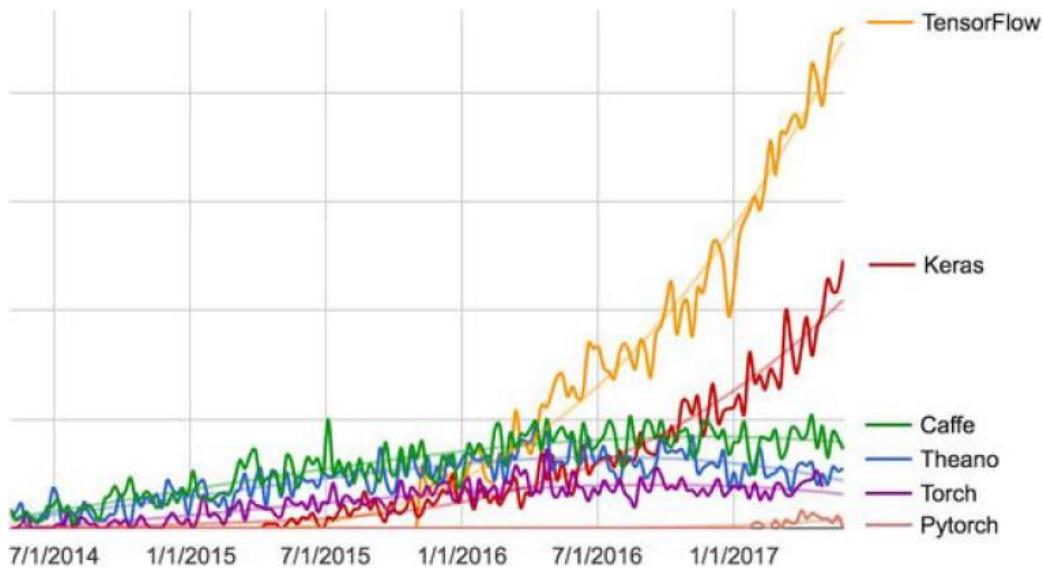
Seperti halnya dengan *Jupyter Notebook*, dengan menggunakan *Colaboratory* dapat melakukan tugas-tugas tertentu dalam paradigma berorientasi sel. Tampilan antarmuka antara *Jupyter Notebook* dengan *Colaboratory* sangat mirip. Pada *Colaboratory* dapat membuat berbagai jenis sel dan menggunakannya untuk membuat buku catatan.

2.5 Keras dan *TensorFlow*

Keras adalah sebuah *framework deep learning* untuk *Python* yang menyediakan cara mudah untuk mendefinisikan dan melatih hampir semua jenis model *deep learning* [1]. *Keras* awalnya dikembangkan untuk para peneliti, dengan tujuan memungkinkan eksperimen cepat. *Keras* memiliki fitur utama berikut:

- a. *Keras* memungkinkan kode yang sama berjalan mulus di CPU atau GPU.
- b. Memiliki API (*Application Programming Interface*) yang *user-friendly* yang membuatnya mudah untuk cepat prototipe model *deep learning*.
- c. Memiliki dukungan bawaan untuk *convolutional networks* (untuk visi komputer), *recurrent network* (untuk pemrosesan urutan), dan kombinasi keduanya.
- d. Mendukung arsitektur jaringan yang diinginkannya: model *multi-input* atau *multi-output*, berbagi lapisan, berbagi model, dan sebagainya. Ini berarti *Keras* tepat untuk membangun model *deep learning* apa pun.

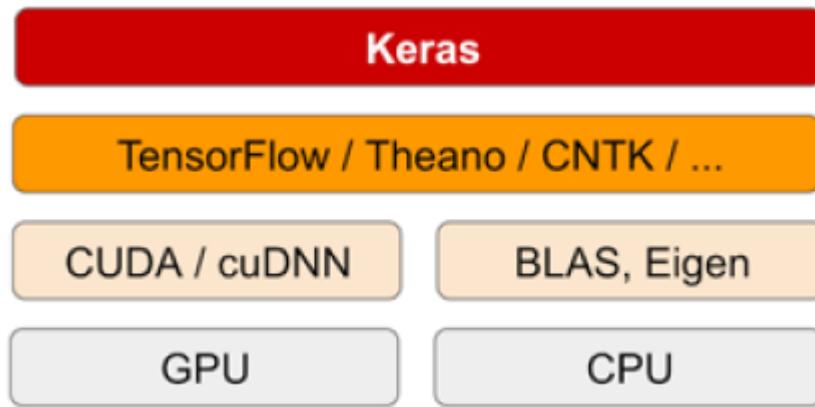
Keras didistribusikan di bawah lisensi MIT permisif, yang berarti dapat digunakan secara bebas dalam proyek komersial. *Framework* ini kompatibel dengan versi *Python* apa pun mulai 2,7 hingga 3,6.



Gambar 2.16 Minat pada pencarian Google untuk framework deep learning yang berbeda dari waktu ke waktu

Keras adalah *library* tingkat model, menyediakan blok-blok tingkat tinggi untuk mengembangkan model *deep learning* [1]. *Keras* tidak dapat menangani operasi tingkat rendah seperti manipulasi tensor dan diferensiasi. Sebaliknya, ia bergantung pada *library tensor* khusus yang dioptimalkan dengan baik untuk melakukannya, berfungsi sebagai mesin *backend* dari *Keras*. Daripada memilih *library tensor* tunggal dan mengimplementasikan *Keras* ke *library* tersebut, *Keras* menangani masalah dengan cara modular sehingga beberapa mesin *backend* yang berbeda dapat dihubungkan dengan lancar ke *Keras*. Saat ini, tiga implementasi *backend* yang ada adalah *backend TensorFlow*, *backend Theano*, dan *backend Microsoft Cognitive Toolkit (CNTK)*. Di masa depan,

kemungkinan *Keras* akan diperluas untuk bekerja lebih dengan mesin eksekusi *deep learning*. Gambar 2.17 merupakan susunan perangkat lunak dan perangkat keras dari *deep learning*.



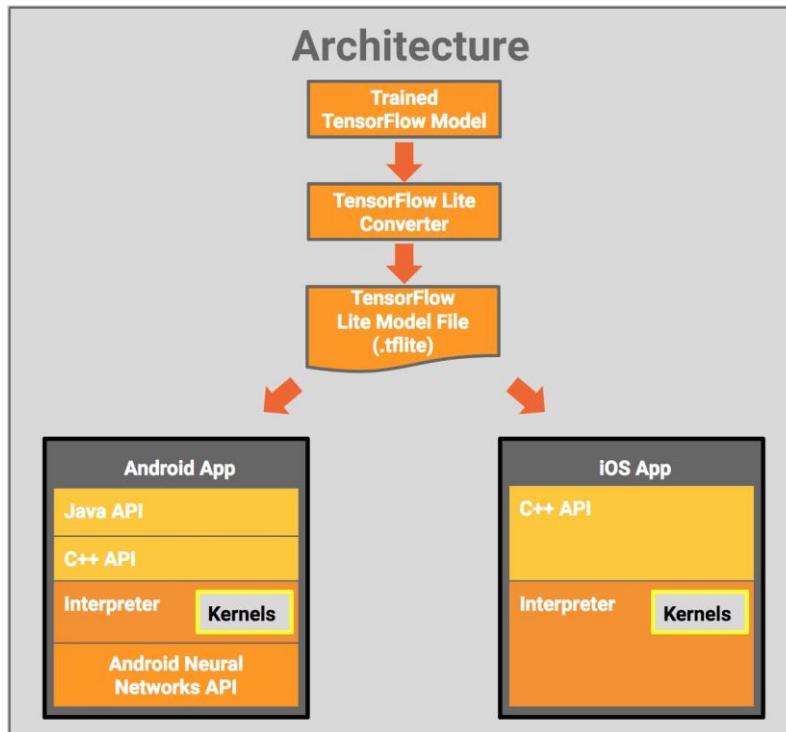
Gambar 2.17 Susunan perangkat lunak dan perangkat keras deep learning

Melalui *TensorFlow* (atau *Theano*, atau *CNTK*), *Keras* dapat berjalan dengan mulus di CPU dan GPU. Saat berjalan pada CPU, *TensorFlow* sendiri melibatkan *library* tingkat rendah untuk operasi *tensor* yang disebut *Eigen*. Pada GPU, *TensorFlow* melibatkan *library* operasi *deep learning* yang dioptimalkan dengan baik yang disebut *Library NVIDIA CUDA Deep Neural Network* (cuDNN).

2.6 *TensorFlow Lite*

TensorFlow Lite adalah solusi ringan *TensorFlow* untuk perangkat seluler dan tertanam [14]. *TensorFlow* itu sendiri adalah sebuah *end-to-end open source platform* yang digunakan untuk *machine learning*. Ini memungkinkan Anda menjalankan model dari mesin yang dipelajari (*machine-learned*) pada perangkat seluler dengan latensi yang rendah, sehingga dapat dimanfaatkan untuk melakukan klasifikasi, regresi, atau apa pun yang diinginkan tanpa harus

mengakses pengiriman dan penerimaan dari atau ke *server*. Arsitektur dari *TensorFlow Lite* dapat dilihat pada gambar 2.18 berikut.



Gambar 2.18 Arsitektur *TensorFlow Lite*

Saat ini *TensorFlow Lite* didukung pada Android dan iOS melalui C++ API (*Application Program Interface*), serta memiliki *Java Wrapper* untuk Pengembang Android. Selain itu, pada Perangkat Android yang mendukungnya, penerjemah (*interpreter*) juga dapat menggunakan *Android Neural Networks API* untuk akselerasi perangkat keras, jika tidak maka akan diproses dengan standar CPU (*Central Processing Unit*) untuk dieksekusi.

TensorFlow Lite terdiri dari *runtime* yang dapat digunakan untuk menjalankan model yang sudah ada sebelumnya, dan seperangkat alat (*tools*) yang dapat digunakan untuk menyiapkan model yang akan digunakan pada perangkat seluler dan tertanam.

TensorFlow Lite bukan dirancang untuk model pelatihan, akan tetapi untuk melatih modelnya dilakukan pada mesin bertenaga yang lebih tinggi, dan kemudian dikonversi modelnya ke dalam format “.TFLITE”, yang kemudian model tersebut dimuat ke dalam penerjemah seluler (*mobile interpreter*).

2.7 *Android Studio*

Android Studio merupakan sebuah *Integrated Development Environment* (IDE) resmi yang digunakan untuk pengembangan aplikasi Android, berdasarkan IntelliJ IDEA [15]. Selain editor kode dan alat pengembang IntelliJ yang baik, *Android Studio* menawarkan lebih banyak fitur yang meningkatkan produktivitas Anda saat membangun aplikasi Android, seperti:

1. Sistem pembangunan berbasis *Gradle* yang fleksibel,
2. *Emulator* yang cepat dan kaya fitur,
3. Lingkungan terpadu yang dapat mengembangkan untuk semua perangkat Android,
4. Menerapkan perubahan untuk memasukan kode dan perubahan sumber daya ke aplikasi yang sedang berjalan tanpa memulai ulang aplikasi tersebut,
5. *Code Templates* dan integrasi dengan GitHub yang dapat membantu untuk membangun fitur aplikasi umum dan mengimpor kode sampel,
6. Alat dan kerangka kerja (*framework*) pengujian yang luas,
7. *Lint tools* untuk menangkap kinerja, kegunaan, kompatibilitas versi, dan masalah lainnya,
8. Dukungan C++ dan NDK,

9. Dukungan bawaan untuk *Google Cloud Platform*, membuatnya mudah untuk mengintegrasikan *Google Cloud Messaging* dan *App Engine*.

Salah satu tugas Android sebagai *Integrated Development Environment* adalah menyediakan antarmuka untuk membuat aplikasi dan mengelola manajemen file yang bisa dibilang kompleks [16]. Bahasa pemrograman yang digunakan adalah *Java*. Beberapa hal yang dapat dilakukan di Android Studio ini yaitu dapat menulis, mengedit, dan menyimpan proyek beserta berbagai *file* yang berhubungan dengan proyek itu.

Tidak hanya itu, Android Studio juga memberi akses ke *Android Software Development Kit (SDK)*. SDK ini dapat dibilang sebagai ekstensi dari kode Java yang memperbolehkannya untuk berjalan dengan mulus di *Device Android*. Jadi, jika Java dibutuhkan untuk menulis programnya, Android SDK diperlukan untuk menjalankan programnya di Android. Untuk menggabungkan keduanya, maka diperlukannya Android Studio. Selain itu, jika menemukan *bug* pada aplikasi yang dibuat maka kita dapat memperbaikinya dengan Android Studio.

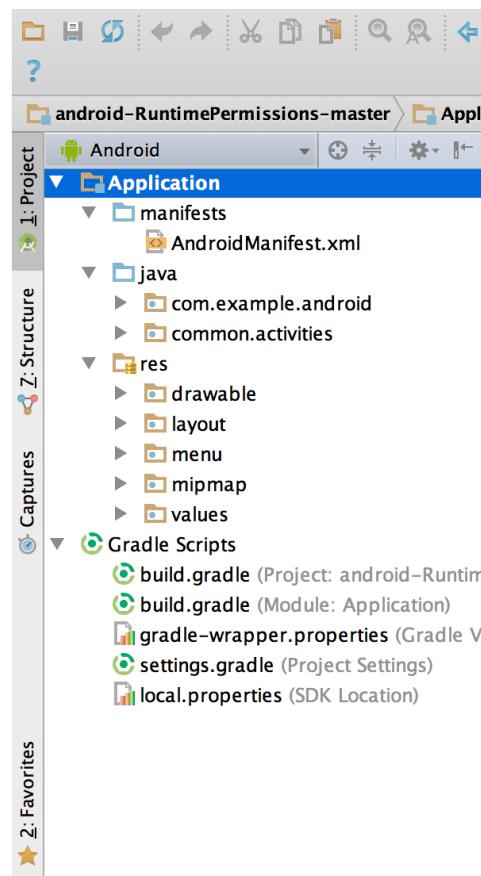
Google sendiri terus mencoba mengoptimasi Android Studio untuk memastikan bahwa perangkat lunak ini dapat membantu dalam membangun Aplikasi Android. Ketika kita sedang melakukan coding, mereka menawarkan *live hints* atau petunjuk langsung. Tidak hanya itu, mereka juga akan memberi saran jika menurut mereka ada yang salah atau ada hal yang bisa membuat coding kita lebih efisien. Selain itu ketika mulai mengetik sebuah baris coding, Android Studio juga akan memberikan saran kode-kode yang bisa digunakan untuk mempercepat proses coding atau jika kita lupa dengan sebuah kode.

2.7.1 Struktur Proyek

Setiap proyek di Android Studio berisi satu atau lebih modul dengan *file* kode sumber (*source code files*) dan *file* sumber (*resource files*) daya [15]. Jenis modul meliputi:

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

Secara umum, Android Studio menampilkan file proyek dalam tampilan proyek Android, seperti yang ditunjukkan pada gambar 2.19. Tampilan ini diatur oleh modul untuk memberikan akses cepat ke *file* sumber utama suatu proyek.



Gambar 2.19 File proyek pada tampilan Android

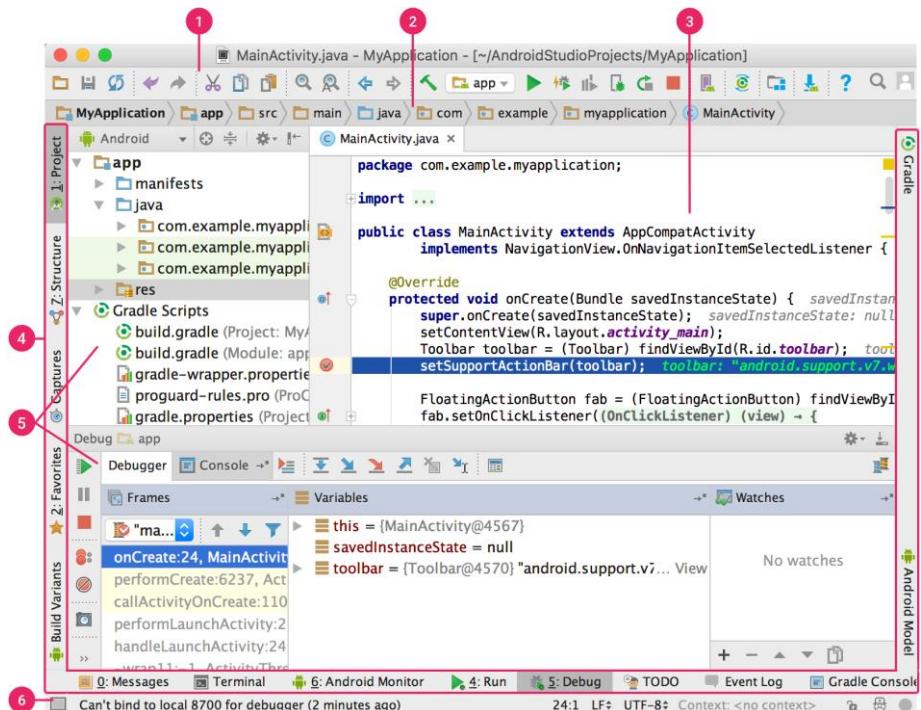
Semua *file build* terlihat di posisi atas di bawah *Script Gradle* dan setiap modul aplikasi berisi folder berikut:

1. Manifests: Berisi *file* “AndroidManifest.xml”.
2. Java: Berisi *file* kode sumber Java, termasuk kode uji JUnit.
3. Res: Berisi semua sumber daya non-kode, seperti tata letak XML, *UI strings*, dan gambar *bitmap*.

Struktur proyek Android pada disk berbeda dari representasi yang ada di sini. Untuk melihat struktur *file* sebenarnya dari proyek yang kita buat, pilih *Project* dari *dropdown Project* seperti yang ditunjukkan pada gambar 2.13.

2.7.2 Tampilan Antarmuka

Jendela utama Android Studio terdiri dari beberapa logika area yang ditunjukkan dalam gambar 2.20.



Gambar 2.20 Jendela utama pada Android Studio

1. *Toolbar* memungkinkan kita melakukan berbagai tindakan, termasuk menjalankan aplikasi dan meluncurkan *Android tools*.
2. *Navigation bar* membantu menavigasi proyek dan membuka *file* untuk diubah.
3. *Editor window* adalah tempat untuk membuat dan memodifikasi kode. Tergantung pada jenis *file* saat ini, *editor* dapat berubah. Misalnya, saat melihat *file* tata letak, *editor* menampilkan *editor* tata letak.
4. *Tool window bar* berjalan di luar jendela IDE dan berisi tombol yang memungkinkan kita untuk memperluas atau mencuatkan *tool window* individu.
5. *Tool windows* memberi akses ke tugas tertentu seperti manajemen proyek, pencarian, kontrol versi, dan banyak lagi.
6. *Status bar* menampilkan informasi proyek kita dan IDE itu sendiri, serta memberi peringatan atau pesan apa pun.

2.7.3 Sistem Pengembangan *Gradle* (*Gradle Build System*)

Android Studio menggunakan *Gradle* sebagai dasar dari pengembangan sistem, dengan lebih banyak kemampuan Android secara spesifik yang disediakan oleh *plugin* Android untuk *Gradle Build System*. *Build System* ini berjalan sebagai alat yang terintegrasi dari menu Android Studio, dan secara independen dari baris perintah. Dengan adanya fitur *build system* maka dapat dilakukan hal-hal seperti berikut:

1. Kustomisasi, konfigurasi, dan perluasan proses pembuatan.
2. Membuat beberapa APK, dengan berbagai fitur menggunakan proyek dan modul yang sama.
3. Menggunakan kembali kode dan sumber daya di seluruh sumber.

Dengan menggunakan fleksibilitas *Gradle*, kita dapat mencapai semua ini tanpa mengubah file sumber inti aplikasi Anda. *File build* *Android Studio* diberi nama “*build.gradle*”. Itu adalah *file* teks biasa yang menggunakan sintaks *Groovy* untuk mengatur *build* dengan elemen yang disediakan oleh *plugin Android* untuk *Gradle*. Setiap proyek memiliki satu *file build* tingkat atas untuk seluruh proyek dan *file build* modul terpisah untuk setiap modul. Ketika kita mengimpor proyek yang sudah ada, *Android Studio* secara otomatis menghasilkan *file build* yang diperlukan.

2.8 Beras

Beras adalah salah satu komoditas penting bagi Indonesia. Hal ini mengingat hampir seluruh masyarakat Indonesia mengkonsumsi beras sebagai makanan pokoknya. Itu sebabnya Indonesia merupakan konsumen pangan dengan bahan pangan beras terbesar. Selain itu, beras sangat berpengaruh bagi perekonomian Indonesia karena lebih dari 60 % penduduk Indonesia berprofesi sebagai petani penghasil beras. Dengan demikian beras tidak hanya dibutuhkan untuk dikonsumsi tetapi juga merupakan sumber pendapatan dan penyerapan tenaga kerja [17].

2.8.1 Beras IR 64

Beras IR 64 merupakan salah satu jenis varietas padi sawah yang memiliki bentuk tegak, tinggi sekitar 115-126 cm. Gabahnya berbentuk ramping dan panjang dan berwarna kuning bersih. Padi jenis ini cocok ditanam di daerah lahan sawah irigasi dataran rendah sampai sedang [18]. Beras IR 64 adalah jenis

beras yang pulen jika dimasak menjadi nasi karena memiliki kadar amilosa sebanyak 23%. Bobot beras per seribu butir beras IR64 adalah 24,1 gram.



Gambar 2.21 Jenis beras IR 64

2.8.2 Beras Basmathi

Beras basmathi bersal dari negara india/pakistan, beras basmathi berasal dari bahasa sanskerta ‘basmathi’ artinya berarti harum atau wangi dalam bahasa india ia juga mempunyai maksud ”soft rice” yaitu lembut [19].

Seperti pada umumnya beras, beras basmati juga mempunyai dua jenis, beras putih dan beras coklat ,Setelah dimasak menjadi nasi barulah keunikan beras ini terlihat. Beras ini akan melar memanjang butiran, sedikit pera, mudah terurai butiran berasnya dan aromanya sangat harum, beras ini memang aromanya sangat harum. Keistimewaan lainnya, butiran atau buliran beras ini panjang dan kecil melebihi ukuran beras yang biasanya agak pendek dan bulat.

Berdasarkan uji laboratorium, beras basmati mengandung 0.09 bagian 2-acetyl-1-pyrroline per juta, yaitu 12 kali lebih dari konsenstrasi pada beras biasa. Karena secara alami wangi makan beras ini tak bisa ditandingi aromanya dengan beras lain.



Gambar 2.22 Jenis Beras Basmathi

2.8.3 Beras Ketan

Beras ketan putih (*Oryza sativa glutinosa*) merupakan salah satu varietas padi yang termasuk dalam famili Graminae [20]. Butir beras sebagian besar terdiri dari zat pati sekitar 80-85% yang terdapat dalam endosperma yang tersusun oleh granula-granula pati yang berukuran 3-10 milimikron. Beras ketan juga mengandung vitamin (terutama pada bagian aleuron), mineral dan air. Dari komposisi kimiawinya diketahui bahwa karbohidrat penyusun utama beras ketan adalah pati. Pati merupakan karbohidrat polimer glukosa yang mempunyai dua struktur yakni amilosa dan amilopektin.



Gambar 2.23 Jenis beras ketan putih

BAB 3

METODOLOGI PENELITIAN

3.1 Waktu dan Tempat Penelitian

Penelitian dilaksanakan dalam waktu 4 bulan dimulai dari bulan Oktober 2019 hingga bulan Januari 2020 bertempat di Kampus Fakultas Teknik Universitas Jenderal Soedirman, Kabupaten Purbalingga.

3.2 Alat dan Bahan

Dalam penelitian ini, daftar alat dan bahan yang digunakan selama penelitian sebagai berikut.

1. Perangkat keras:

- a. Sebuah *Laptop* Dell Inspiron 14 3000 Series dengan spesifikasi prosesor Intel Celeron 3205 dan RAM 4 GB.
- b. Komputer virtual *Google Colaboratory* dengan spesifikasi GPU: 1x Tesla K80, komputasi 3.7, mempunyai inti CUDA 2496 , 12 GB GDDR5 VRAM, CPU: 1x *single core hyper threaded Xeon Processors* @2.3 Ghz, RAM: ~12.6 GB, *Disk*: ~33 GB.
- a. Sebuah *Smartphone Android* Oppo A37f dengan spesifikasi prosesor Qualcomm MSM8916 Snapdragon 410 *Quad-core* 1,21 GHz Cortex-A53 dan RAM 2 GB.

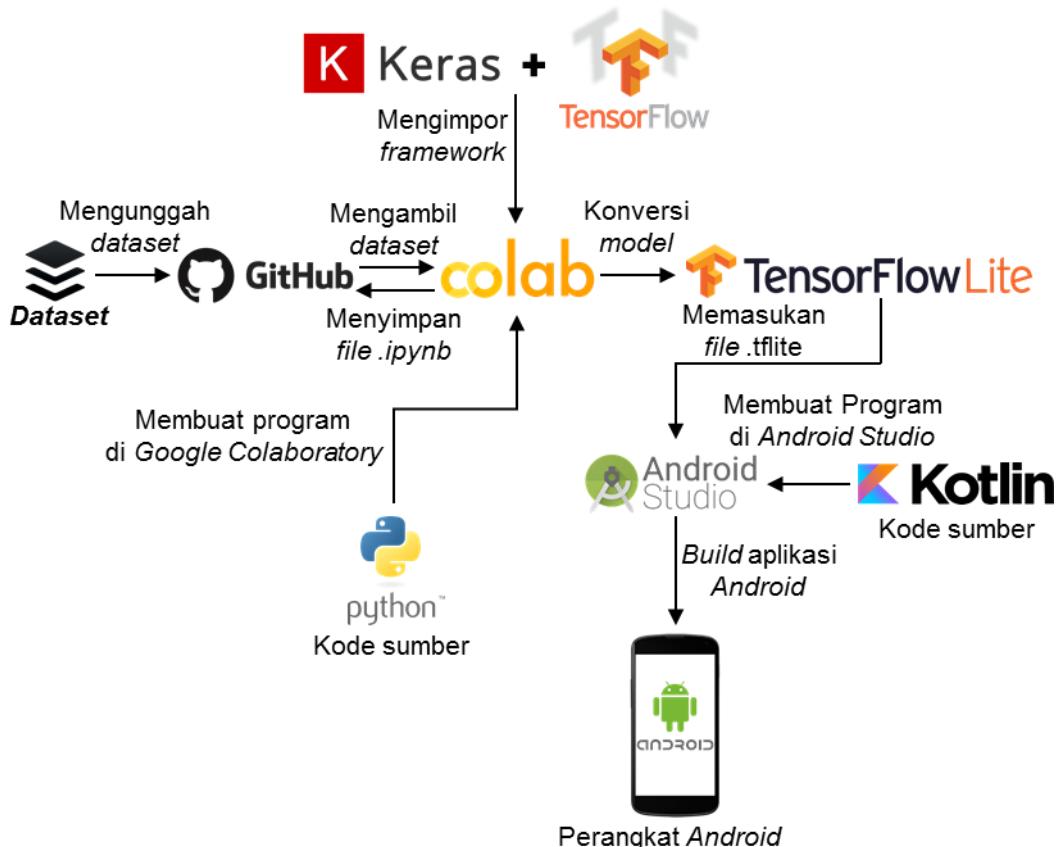
2. Perangkat lunak:

- a. Sistem Operasi *Windows* 10 64 bit
- b. Peramban Internet *Google Chrome* versi 77.0.3865.120 64 bit
- c. *Google Colaboratory (Jupyter Notebook versi cloud)*.

- d. *Android Studio* versi 3.5.0.0.
 - e. Sistem Operasi *Android Lollipop* 5.1.
 - f. Layanan Repository *Web Development* pada Platform *Github*.
3. *Dataset* pelatihan dan pengujian yang berupa gambar jenis beras IR 64, Basmathi, dan Ketan berformat “.jpg” yang diambil menggunakan kamera *smartphone*.

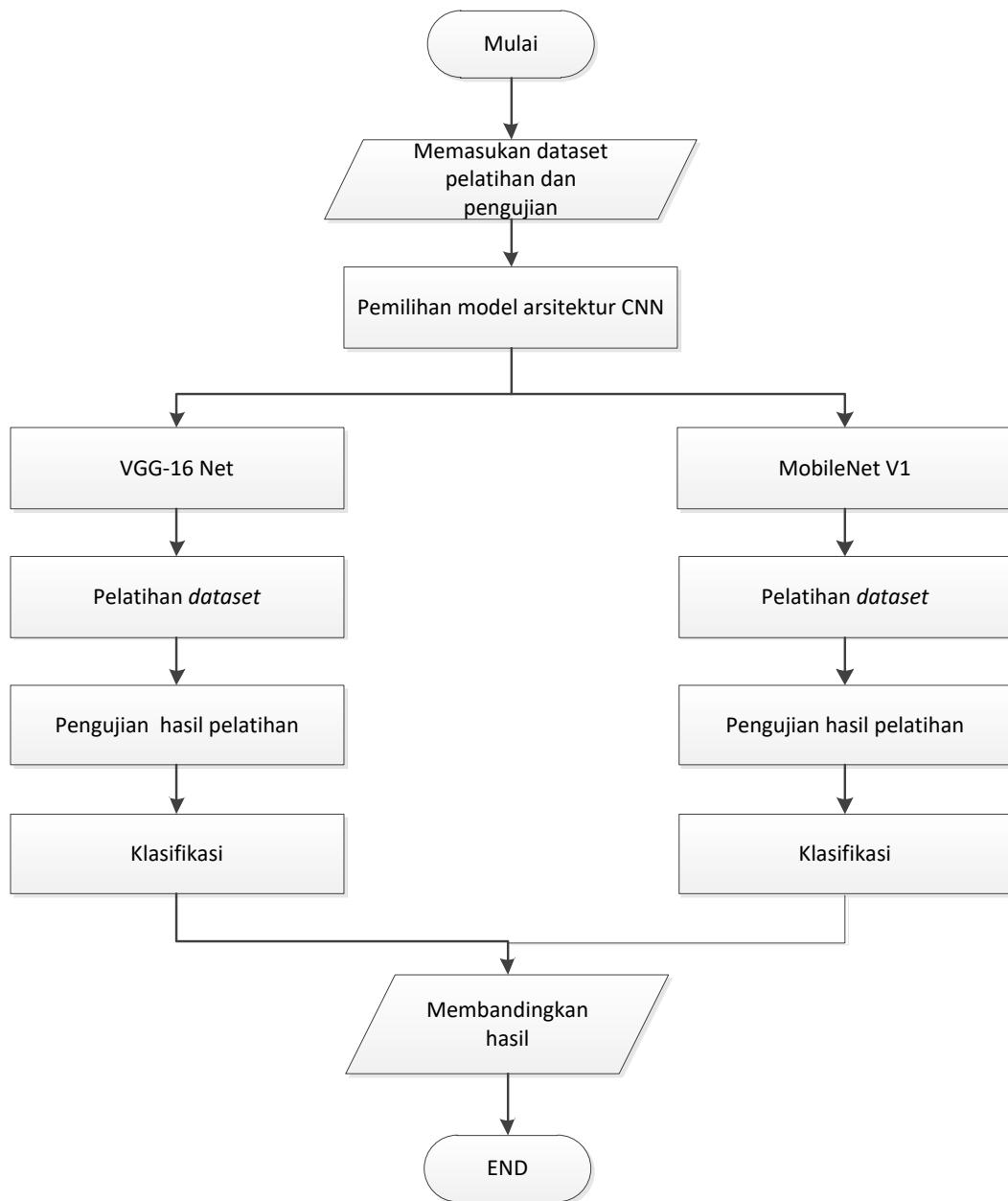
3.3 Metode Penelitian

Penelitian ini dilaksanakan melalui beberapa tahapan yaitu tahap penyiapan dan *pre-proses dataset*, tahap desain arsitektur, dan tahap pengujian. Desain arsitektur dari sistem yang akan dibuat disajikan pada gambar 3.1 berikut.



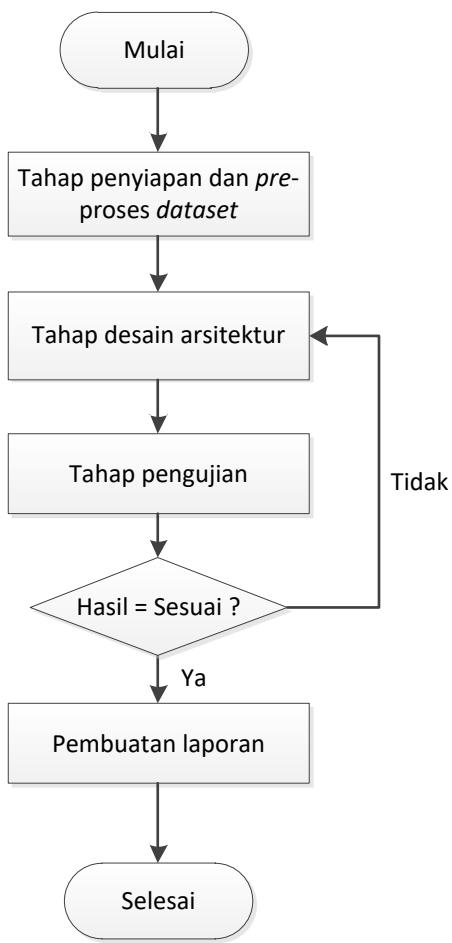
Gambar 3.1 Desain arsitektur sistem

Sedangkan untuk diagram alir sistem yang dirancang pada penelitian ini dapat dilihat pada gambar 3.2 berikut.



Gambar 3.2 Diagram alir sistem

Adapun diagram alir dari tahapan-tahapan penelitian yang dilakukan adalah seperti yang disajikan pada gambar 3.3 berikut.



Gambar 3.3 Diagram alir penelitian

3.3.1 Tahap Penyiapan dan Pre-proses Dataset

Pada tahap ini penyiapan dan *pre-proses dataset* dilakukan dengan mengambil gambar dari obyek tiga macam varietas beras menggunakan kamera *smartphone* dengan format gambar “*.jpg” yang kemudian di-*crop* dan dimasukkan sebagai *dataset* pelatihan dan pengujian pada *Github* yang selanjutnya akan dimasukkan ke *Google Colaboratory*.

3.3.2 Tahap Desain Arsitektur

Proses awal desain arsitektur dimulai dari membuat kode sumber untuk program CNN dan mengimpor *Framework Keras* dan *TensorFlow* yang dibuat

menggunakan infrastruktur *Google Colaboratory* dengan bahasa pemrograman *Python* dan disimpan dalam bentuk *file Jupyter Notebooks* “.ipynb” dan kemudian disimpan ke *Github*. Selanjutnya *dataset* pelatihan tersebut diambil dari *Github* dan disimpan ke dalam tempat penyimpanan sementara pada *Google Colaboratory*. Sistem yang dirancang ini menggunakan dua macam arsitektur jaringan CNN, yaitu *VGG16Net* dan *MobileNetV1*, sedangkan untuk metode yang digunakan adalah metode *transfer learning* dengan mengambil *file Keras* “*.h5” yang sudah dilatih sebelumnya pada kedua arsitektur tersebut dan melakukan *feature extraction* terhadap *file* tersebut. Selain itu juga ditambah satu *filter/feature map* tambahan sebagai keluaran untuk klasifikasi tersebut.

3.3.3 Tahap Pengujian

Pada tahap pengujian seluruh *dataset* pelatihan dilatih terhadap masing-masing arsitektur jaringan CNN yang digunakan sampai dengan panjangnya *epochs* yang ditentukan. Sesudah itu *dataset* pengujian diekstrak dan dibandingkan hasilnya dengan data pengujian untuk melakukan prediksi pada klasifikasi varietas beras serta membandingkan hasilnya dengan kedua arsitektur tersebut yang digunakan. Setelah pengujian selesai dilanjutkan dengan melakukan konversi hasil pelatihan masing-masing arsitektur tersebut ke dalam format *TensorFlow Lite* “*.tflite”. Setelah itu membuat program pada *Android Studio* dan memasukan *file* “*.tflite” yang didapat ke dalam folder “*assets*” pada direktori *project Android Studio* tersebut. Setelah program dikompilasi dan di-build menjadi *file* “*.apk”, maka aplikasi Android klasifikasi tiga varietas beras tersebut dapat dipasang pada *smartphone Android*. Kemudian diuji dan dibandingkan juga

hasil prediksi dari klasifikasinya pada aplikasi Android dengan menggunakan kedua arsitektur tersebut.

3.4 Waktu dan Jadwal Penelitian

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

Tabel 3.1 Jadwal Penelitian

No.	Kegiatan	Bulan 1				Bulan 2				Bulan 3				Bulan 4			
		I	II	III	IV												
1.	Studi Pustaka																
2.	Penyiapan dan <i>pre-proses</i> <i>dataset</i>																
3.	Desain arsitektur																
4.	Pengujian dan evaluasi sistem																
5.	Pembuatan laporan																

BAB 4

HASIL DAN PEMBAHASAN

4.1 Perancangan Sistem dan *Dataset* Penelitian

Penelitian ini menggunakan dua kondisi *dataset* dari ketiga varietas beras. *Dataset* tersebut digunakan untuk melakukan perancangan sistem pada *Google Colaboratory*. Model arsitektur CNN yang diperoleh dari pelatihan pada *Google Colaboratory* disimpan dalam bentuk *TensorFlow Lite* yang nantinya akan digunakan untuk merancang sistem pada perangkat Android.

4.1.1 *Dataset*

Dataset yang digunakan pada klasifikasi varietas beras ini ada tiga macam, 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) dan diubah ukurannya menjadi 224x224 piksel sesuai dengan masukkan pada arsitektur *VGG-16Net* dan *MobileNetV1*.

1. Jumlah *Dataset* Pelatihan

Pada penelitian ini jumlah *dataset* pelatihan yang digunakan sebanyak 60 gambar berformat “.jpg” di setiap varietas beras sehingga totalnya 180 gambar untuk 3 varietas beras. *Dataset* dari ketiganya disimpan di dalam *folder* yang terpisah. *Dataset* pelatihan ini diambil sebanyak 80% secara acak dari *folder train* yang berjumlah 75 gambar di setiap varietas beras.

2. Jumlah *Dataset* Validasi

Dataset validasi yang digunakan sebanyak 20% dari *folder train* dengan jumlah 75 gambar. Sehingga jumlahnya sebanyak 15 gambar di setiap varietas beras dan totalnya sebanyak 45 gambar. *Dataset* validasi digunakan untuk menguji dan membandingkan hasil pelatihan dengan *dataset* pelatihan di setiap *epoch*-nya. *Dataset* validasi dari ketiganya juga disimpan di dalam *folder* yang terpisah sesuai nama varietas berasnya.

3. Jumlah *Dataset* Pengujian

Jumlah *dataset* pengujian yang digunakan sama seperti dataset validasi yaitu sebanyak 15 gambar pada masing-masing varietas beras. Untuk dataset pengujian disimpan dalam *folder test*, terpisah dari *dataset* pelatihan dan validasi. *Dataset* pengujian ini digunakan untuk menguji hasil pelatihan pada jaringan CNN yang digunakan.

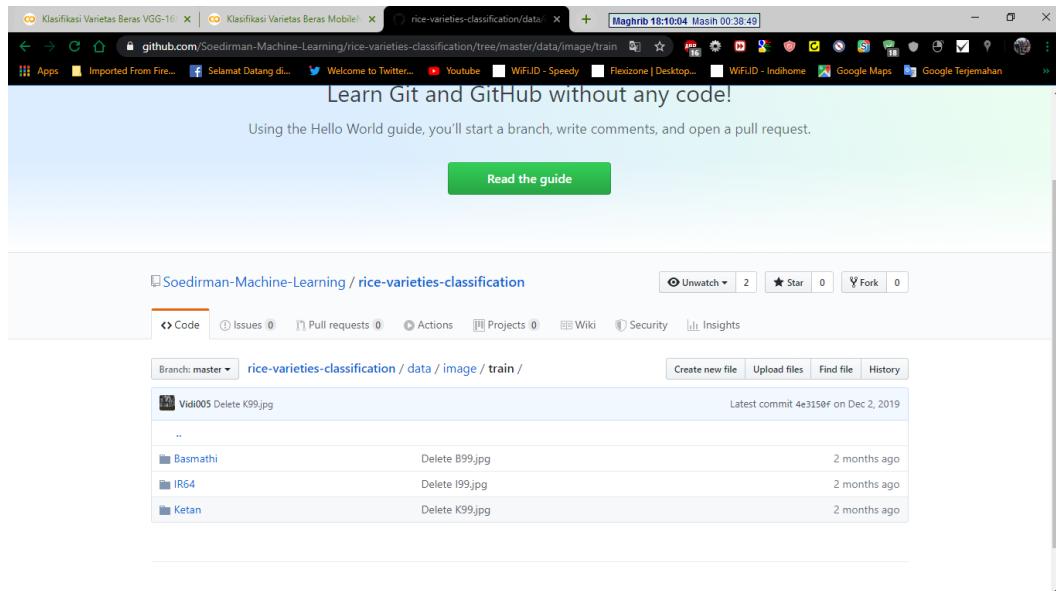
4.1.2 Pengambilan Dataset

Seluruh *dataset* dari varietas beras yang digunakan pada klasifikasi varietas beras diambil menggunakan sebuah kamera *smartphone Android*. Pengambilan *dataset* dilakukan dengan mengambil gambar dari varietas beras yang akan dilatih dan diuji pada alas berwarna hitam dengan jarak antara obyek (beras) dengan kamera antara 10 cm sampai dengan 15 cm. Hal tersebut dilakukan karena jarak tersebut sudah cukup dekat dan jelas hasilnya dan jika terlalu dekat maka kamera *smartphone* akan sulit untuk fokus terhadap obyek yang akan diambil. Alas berwarna hitam digunakan untuk mempermudah CNN melakukan

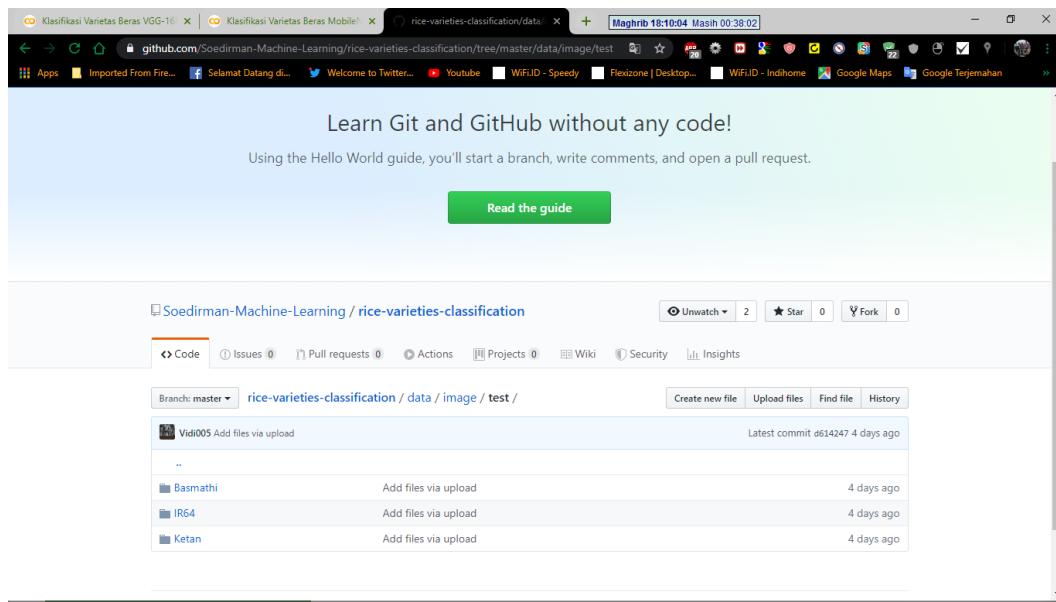
klasifikasi obyek beras yang terlihat berbeda warnanya. Semua *dataset* yang diambil menggunakan kamera *smartphone* disimpan dalam format gambar “.jpg”.

4.1.3 Penyimpanan *Dataset*

Seluruh *dataset* pelatihan, *dataset* validasi, dan *dataset* pengujian diunggah dan disimpan pada layanan Repositori *Web Development* pada *Platform Github*. *Dataset* tersebut disimpan di dalam sebuah repositori dengan nama “rice-varieties-classification”. *Dataset* pelatihan dan validasi disimpan di dalam *folder* “train” dan *dataset* pengujian disimpan di dalam *folder* “test”. Kedua *folder* tersebut disimpan di direktori *folder* “data” dengan *subfolder* “image”. Di setiap *folder* “train” dan “test” tersebut masing-masing terdapat 3 *folder* untuk menyimpan masing-masing *dataset* varietas beras, yaitu *folder* “Basmathi”, “IR64”, dan “Ketan”. Tampilan dataset train dan test dapat dilihat masing-masing pada gambar 4.1 dan gambar 4.2.



Gambar 4.1 Tampilan direktori dataset train pada Github

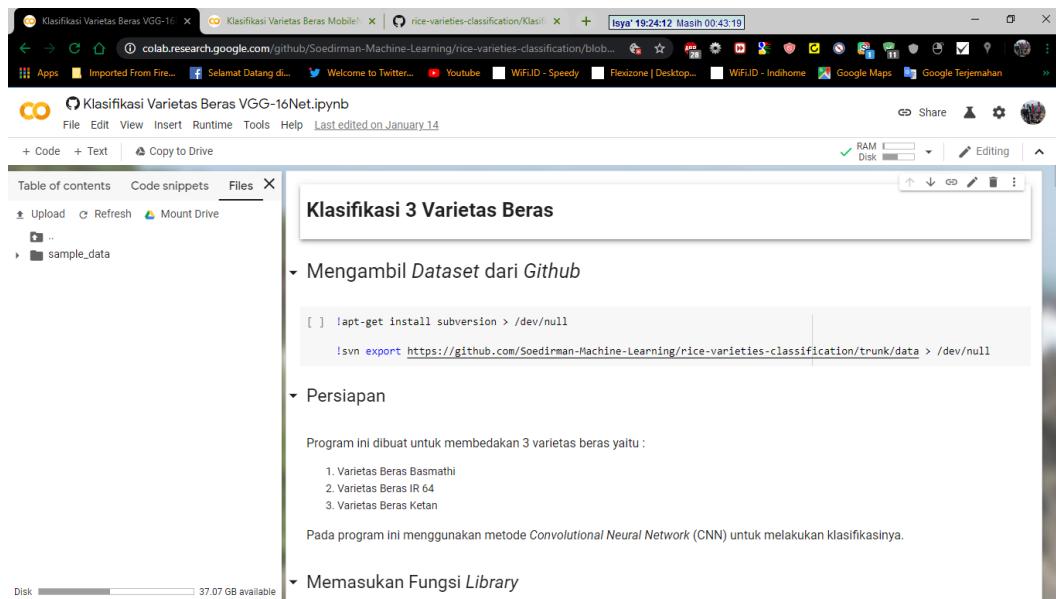


Gambar 4.2 Tampilan direktori dataset test pada Github

Selain untuk menyimpan *dataset*, pada repositori ini juga digunakan untuk menyimpan program klasifikasi varietas beras yang dibuat dengan *Google Colaboratory* serta program Aplikasi *Android* untuk klasifikasi varietas beras yang dibuat dengan *Android Studio*.

4.1.4 Perancangan Program pada *Google Colaboratory*

Perancangan program pada infrastuktur *Google Colaboratory* dilakukan dengan membuat beberapa sel/baris *text* untuk memberi judul, keterangan dan penjelasan program di setiap barisnya serta membuat sel/baris *code* untuk mengimpor fungsi *library* dan membuat program di setiap baris yang dibuat. Program tersebut dibuat dengan *Framework Keras* dengan *backend TensorFlow*. Berikut tampilan program klasifikasi varietas beras pada *Google Colaboratory* dapat dilihat pada gambar 4.3.



Gambar 4.3 Tampilan program klasifikasi varietas beras pada Google Colaboratory

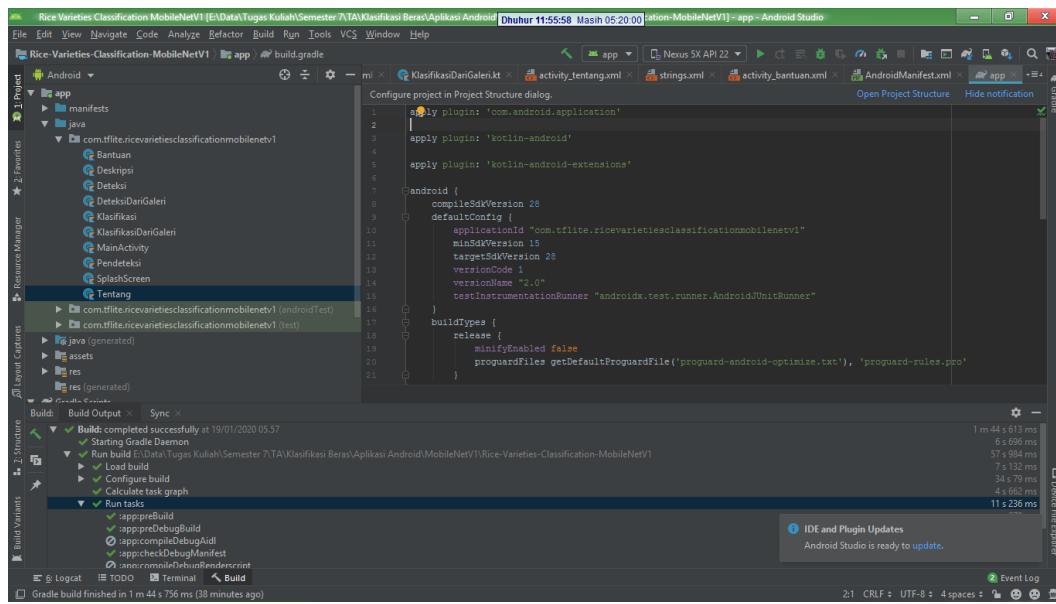
Pembuatan program yang pertama adalah membuat program untuk mengambil *dataset* dari *Github*. Kemudian memasukan/mengimpor fungsi *library* yang nantinya akan digunakan untuk menjalankan program yang membutuhkan *library* tambahan tersebut. Selanjutnya terdapat program yang digunakan untuk mengubah ukuran 224x224 piksel sesuai pada masukkan dari arsitektur *VGG-16Net* dan *MobileNetV1*, memecah *folder train* menjadi *dataset* pelatihan dan validasi serta memuat seluruh *dataset* pengujian secara urut. Setelah itu dibuat program untuk menentukan nama label varietas berasnya urut secara abjad dan menyimpannya dalam bentuk *text* “.txt”. Selain itu membuat program untuk mengambil *base model* untuk melakukan *feature extraction* dari file model “.h5” yang sudah dilatih pada arsitektur *VGG-16Net* dan *MobileNetV1*. Setelah itu membuat program untuk menambahkan lapisan klasifikasi untuk 3 kelas varietas beras dengan menggunakan teknik *Global Average Pooling*. Kemudian membuat program untuk melakukan pelatihan dari kedua arsitektur tersebut dengan *epochs*

sebanyak 100 kali. Selain itu dibuat program untuk menampilkan hasil pelatihan dalam bentuk grafik tingkat akurasi dan tingkat kesalahannya. Sesudah itu terdapat program yang dibuat untuk menguji model hasil pelatihannya dengan *dataset* pengujian. Selanjutnya dibuat juga program untuk menampilkan hasil pengujian dalam bentuk *confusion matrix* yang berfungsi untuk mengetahui tingkat keberhasilan hasil pengujian dari setiap varietas beras. Pada tahap yang terakhir terdapat program yang dibuat untuk menyimpan dan mengkonversi model hasil pelatihan ke dalam bentuk file *TensorFlow Lite* “.tflite”. Kode sumber (*source code*) program *Jupyter Notebooks* “.ipynb” untuk klasifikasi varietas beras dengan CNN dapat dilihat pada lampiran 1 dan lampiran 2.

4.1.5 Perancangan Program pada *Android Studio*

Perancangan program untuk perangkat *Android* dilakukan setelah membuat program pada *Google Colaboratory* dan mendapatkan hasil model pelatihannya yang disimpan dalam bentuk “.tflite”. Program dalam Bahasa *Kotlin* yang dibuat meliputi penggunaan fitur kamera pada *smartphone Android*, mengimpor *library TensorFlow Lite*, membuat *file class* untuk klasifikasi obyek beras yang diambil dengan kamera maupun mengimpor gambar dari galeri foto, memasukkan *file* “.tflite” beserta labelnya dengan format “.txt” ke dalam *folder assets* serta membuat *class* lainnya untuk mendukung dalam membuat antarmuka aplikasi *Android*. Kode sumber dari program tersebut dapat dilihat pada lampiran 3 di penelitian ini. Adapun pembuatan *layout* dalam format “.xml” yang berfungsi untuk membuat tampilan antarmuka pada aplikasi *Android* dari setiap *class* yang akan ditampilkan pada aplikasi *Android* untuk klasifikasi varietas beras. Gambar

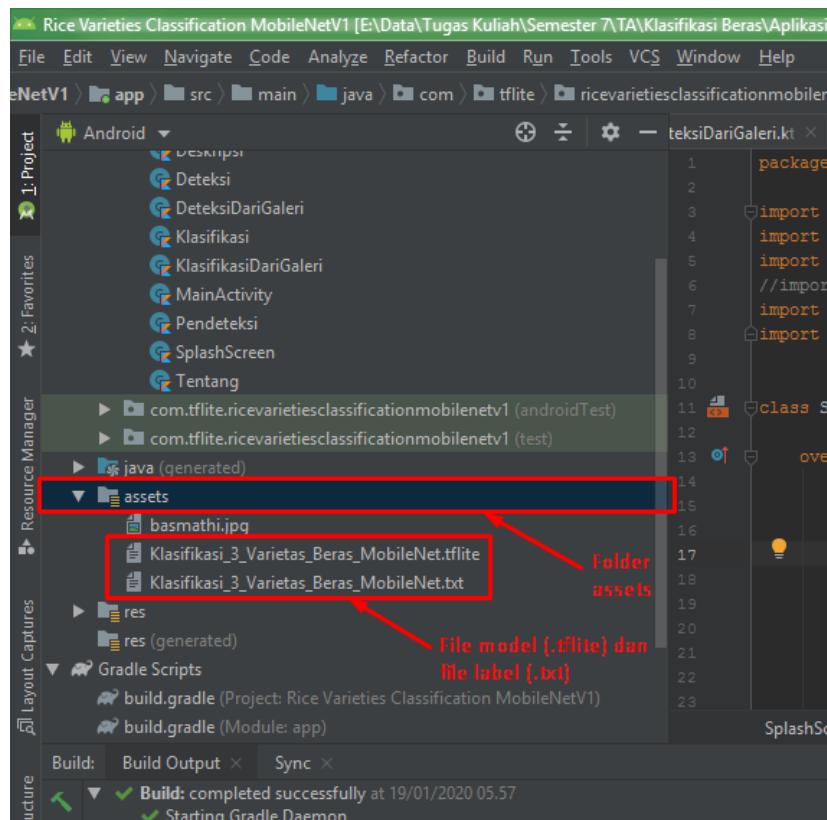
4.4 berikut merupakan tampilan dari *file project* klasifikasi varietas beras pada *Android Studio* IDE.



Gambar 4.4 Tampilan project aplikasi klasifikasi varietas beras di Android Studio

Sebelum membuat *class* untuk klasifikasi varietas beras dengan kamera, maka perlu dimasukan fitur penggunaan kamera dan perijinan penggunaan kamera pada *file* “AndroidManifest.xml”. Selanjutnya untuk menggunakan *mobile interpreter* sebagai penerjemah *file* “.tflite” maka diharuskan mengimpor *library* TensorFlow Lite dengan cara memasukan implementasi “org.tensorflow:tensorflow-lite:1.13.1” di dalam *dependencies* pada *build.gradle*. Pada pembuatan program untuk klasifikasi varietas beras dengan kamera dibuat dengan 3 *file class*. Kelas yang pertama digunakan untuk membuat antarmuka kamera dan mengambil gambar yang selanjutnya akan diekstrak dalam bentuk *array* agar dapat diolah oleh model CNN. Kelas selanjutnya dibuat untuk memuat *file* model (.tflite) dan melakukan klasifikasi varietas beras serta memotong ukuran gambar RGB menjadi 224x224 piksel, sedangkan kelas yang ketiga untuk

mengubah nilai data hasil klasifikasi dalam bentuk persentase. Pada pembuatan program untuk klasifikasi beras dengan mengimpor gambar beras yang sudah tersimpan di dalam galeri foto *Android* dibuat dengan 2 file *class*. Kelas pertama dibuat untuk mengimpor gambar dari galeri dan mengubah ukurannya menjadi 224x224 piksel serta menampilkan hasil prediksi dan probabilitasnya dalam persentase. Kelas berikutnya dibuat untuk melakukan klasifikasi, yaitu dengan membuat program untuk memuat *file model* (.tflite), mengekstrak gambar menjadi bentuk *array*, dan menghitung probabilitas hasil prediksi. Setelah seluruh program dibuat maka *file model CNN* yang sudah dilatih dan disimpan dalam bentuk “.tflite” beserta labelnya dalam bentuk “.txt” dimasukan ke dalam *folder assets* di dalam *project Android Studio*.



Gambar 4.5 File model (.tflite) beserta labelnya (.txt) pada folder assets

4.2 Pelatihan dan Pengujian pada *Google Colaboratory*

Pelatihan dan pengujian pada infrastruktur *Google Colaboratory* berfungsi untuk melatih *dataset* dan memperoleh modelnya serta melakukan pengujian dari hasil pelatihan tersebut. Pelatihan dan pengujian tersebut dilakukan menggunakan metode *feature extraction* dengan cara mentransfer hasil pembelajaran) menggunakan arsitektur *VGG-16Net* dan *MobileNetV1*. Sedangkan untuk melakukan klasifikasinya menggunakan teknik *Global Average Pooling*. Pelatihan dan pengujian pada *Google Colaboratory* ini dilakukan dengan menjalankan setiap baris program yang sudah dibuat. Pada saat pelatihan dan pengujian harus dipastikan bahwa *laptop* yang kita gunakan terhubung ke internet dan juga sudah terkoneksi dengan *server Google Colaboratory*. Sesudah terhubung ke *server* maka perlu mengubah pengaturan jenis *runtime*-nya menjadi GPU agar proses pelatihan dapat dilakukan lebih cepat. Setelah itu semua baris program dijalankan maka akan ditampilkan hasil pelatihan dan pengujianya dalam bentuk kurva tingkat akurasi dan tingkat kesalahan dari pelatihan, validasi, dan pengujianya. Selain itu juga model pada hasil pelatihan tersebut disimpan ke dalam direktori *folder* sementara pada *Google Colaboratory* yang nantinya akan diunduh dan digunakan pada aplikasi *Android*. Pengujian tersebut dilakukan terhadap tiga varietas beras, variasi kualitas gambar *dataset*, dan arsitektur yang digunakan.

4.2.1 Varietas Beras

Varietas beras yang akan dilatih dan diuji adalah varietas Beras Basmathi, Beras IR-64, dan Beras Ketan. Ketiga varietas tersebut masing-masing disimpan

di dalam *folder* yang berbeda pada *Github* agar tidak tercampur pada saat pelatihan.

4.2.2 Variasi Kualitas Gambar *Dataset*

Dataset yang diambil untuk klasifikasi varietas beras terdapat dua kondisi, yaitu menggunakan *flashlight* pada kamera *smartphone* dan tanpa menggunakan *flashlight*. Kedua kondisi *dataset* tersebut diambil di dalam ruangan yang sama. Pada kondisi menggunakan *flashlight*, intensitas cahaya yang mengenai obyek beras pada jarak 10 – 15 cm sekitar 1000 *lux*. Pada kondisi tanpa *flashlight*, intensitas cahaya yang mengenai obyek sekitar 20 *lux*. Pengukuran intensitas cahaya dilakukan menggunakan aplikasi *Light Meter* pada perangkat *Android*.

4.2.3 Arsitektur CNN yang digunakan

Arsitektur CNN yang digunakan untuk pelatihan pada *Google Colaboratory* adalah arsitektur *VGG-16Net* dan *MobileNetV1*. Metode CNN yang digunakan pada kedua arsitektur tersebut adalah *Feature Extraction*.

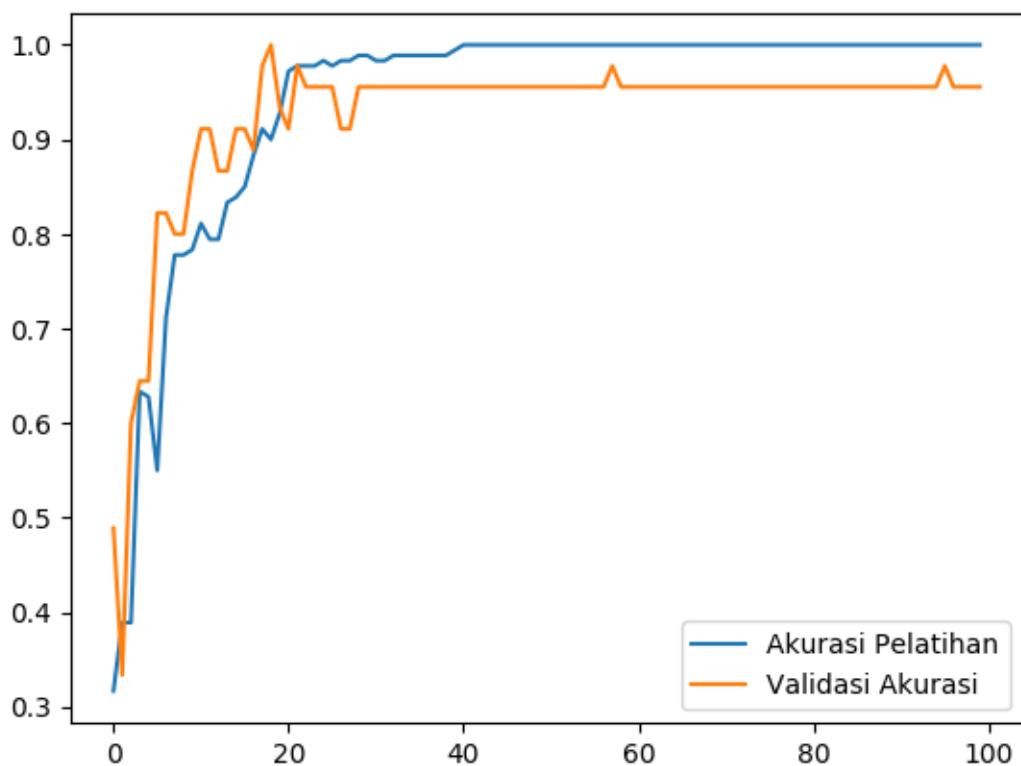
4.2.4 Hasil Pelatihan dan Pengujian pada *Google Colaboratory*

Pada penelitian ini pengujian yang dilakukan pada infrastruktur *Google Colaboratory* terdapat empat macam kondisi, yaitu dengan menggunakan *dataset* pelatihan dan pengujian dengan kualitas gambar yang baik (menggunakan *flashlight*) menggunakan arsitektur *VGG-16Net*, *dataset* dengan kualitas gambar yang baik menggunakan arsitektur *MobileNetV1*, *dataset* dengan kualitas gambar buruk (tanpa *flashlight*) menggunakan arsitektur *VGG-16Net* dan *dataset* kualitas buruk menggunakan arsitektur *MobileNetV1*. Hal tersebut dilakukan untuk menguji kinerja arsitektur yang digunakan apakah mampu melakukan klasifikasi

varietas beras yang dapat dilihat dari tingkat akurasi keberhasilan dalam memprediksi varietas beras. Pada masing-masing pengujian dilakukan sebanyak 100 *epochs*.

1. Hasil pelatihan dan pengujian pada arsitektur *VGG-16Net* dengan kualitas gambar baik

Berikut hasil pelatihan (*training*) *dataset* pelatihan dan validasi yang dapat dilihat pada gambar 4.6.

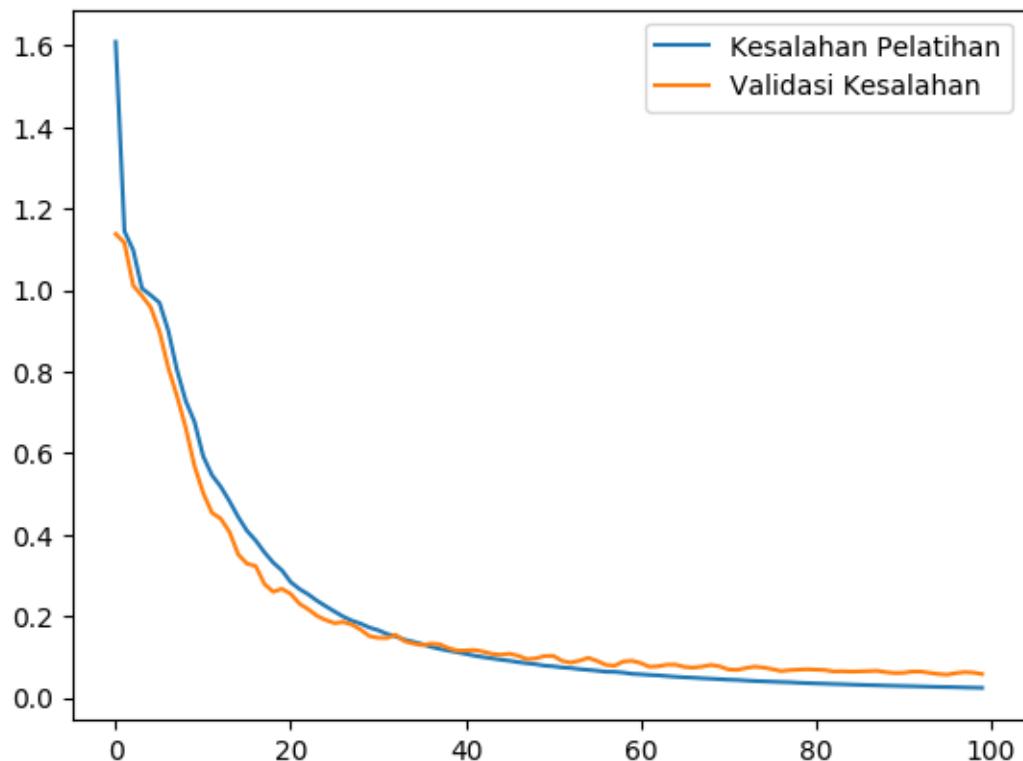


Gambar 4.6 Kurva tingkat akurasi hasil pelatihan dan validasi pada *VGG-16Net*

Dari hasil pelatihan tersebut dapat dilihat bahwa kurva hasil pelatihan dan validasinya semakin meningkat mendekati nilai 1.0 seiring bertambahnya *epochs* (lembar kerja). Pada *epoch* yang terakhir nilai akurasi pelatihan mencapai 1.0 dan nilai akurasi validasi mencapai 0.9556. Kurva akurasi pelatihan yang berwarna

biru dan kurva akurasi validasi yang berwarna oranye merupakan nilai akurasi dari setiap *epoch*. Jika nilainya semakin mendekati 1 selama bertambahnya *epoch* maka proses pelatihan dan validasinya dinyatakan berhasil melakukan klasifikasi. Jika nilai akurasi dari pelatihan dan validasinya tidak meningkat atau menurun maka jaringan CNN yang dilatih tidak dapat melakukan klasifikasi dengan baik. Dari kurva tersebut juga dapat dilihat bahwa kurva akurasi pelatihan hampir sama dengan kurva validasi akurasi. Hal tersebut menunjukkan hasil pelatihan dengan keadaan yang *goodfit*.

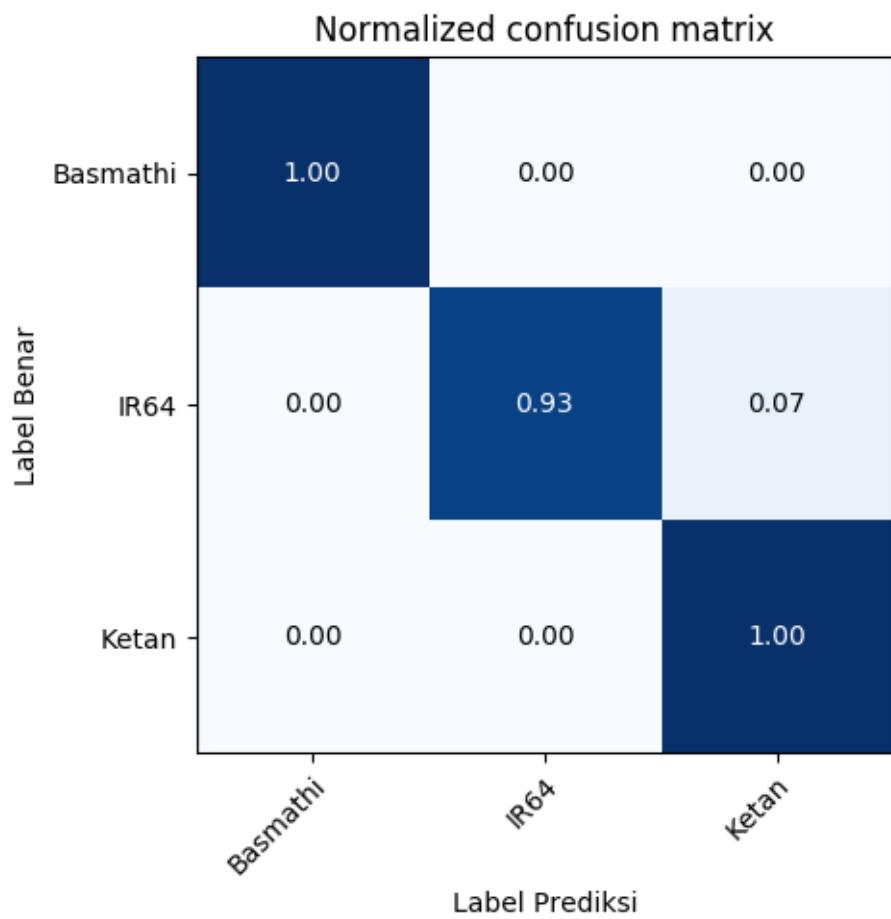
Adapun grafik kesalahan (*error*) yang dihasilkan selama pelatihan seperti pada gambar 4.7 berikut.



Gambar 4.7 Kurva tingkat kesalahan pelatihan dan validasi pada VGG-16Net

Dari hasil pelatihan tersebut dapat diketahui bahwa grafik dari kesalahan pelatihan dan validasinya semakin menurun mendekati 0.0 seiring bertambahnya *epochs*. Pada *epoch* yang terakhir nilai kesalahan pelatihan mencapai 0.024 dan nilai kesalahan validasinya mencapai 0.058. Kurva kesalahan pelatihan yang berwarna biru dan kurva kesalahan validasi yang berwarna oranye merupakan nilai kesalahan yang dihasilkan dari setiap *epoch*. Jika nilainya semakin mendekati 0 maka jaringan CNN yang dilatih dapat melakukan klasifikasinya dengan baik dan jika nilainya tidak menurun atau bertambah maka tidak dapat melakukan klasifikasi dengan baik. Dari kurva tersebut juga dapat dilihat bahwa kurva kesalahan pelatihan hampir sama penurunannya dengan kurva validasi kesalahan sehingga dapat disimpulkan hasil pelatihannya *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 varietas beras Basmathi, IR-64, dan Ketan. Label prediksi merupakan label hasil pengujian pada arsitektur *VGG-16Net*, sedangkan label benar adalah dari *dataset* pengujian yang sudah ditandai label di setiap varietas berasnya.



Gambar 4.8 Confusion matrix hasil pengujian pada VGG-16Net

Berdasarkan *confusion matrix* dari hasil pengujian tersebut dapat disimpulkan bahwa hasil prediksi dari varietas Beras Basmathi bernilai 1.00 pada label benar Basmathi, 0.00 pada label benar IR64 dan 0.00 pada label benar Ketan. Hal tersebut menandakan bahwa seluruh hasil prediksi dari *dataset* pengujian sesuai dengan varietas beras yang diberi label. Untuk hasil prediksi varietas Beras IR64 bernilai 0.00 pada label benar Basmathi, 0.93 pada label benar IR64 dan 0.07 pada label benar Ketan. Dari hasil tersebut disimpulkan bahwa nilai 0.93 merupakan tingkat kebenaran hasil prediksi dari *dataset* pengujian yang diprediksi benar sebanyak 14 dari 15 *dataset* pengujian pada

varietas IR64. Sedangkan nilai 0.07 merupakan hasil prediksi Beras Ketan pada label benar IR64. Pada hasil prediksi Beras Ketan sama seperti pada Beras Basmathi dengan nilai 1.00 pada label benar Ketan dan bernilai 0.00 pada label benar Basmathi dan IR64 yang menandakan seluruh hasil prediksi dari dataset pengujian Beras Ketan benar.

Dari *confusion matrix* tersebut juga dapat dihitung persentase tingkat akurasi dan tingkat kesalahan dari hasil prediksinya seperti berikut.

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

$$\% \text{akurasi} = \frac{44}{45} \times 100\% = 97,78\%$$

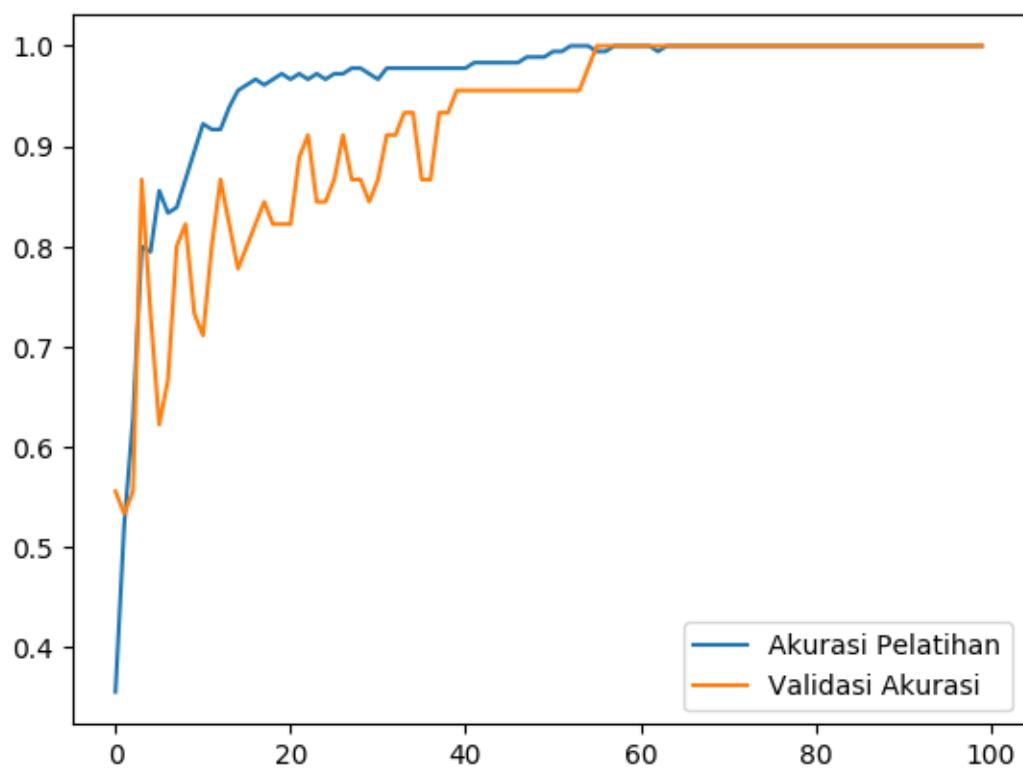
$$\% \text{kesalahan} = \frac{\text{jumlahprediksisalah}}{\text{jumlahdatapengujian}} \times 100\%$$

$$\% \text{kesalahan} = \frac{1}{45} \times 100\% = 2,22\%$$

Berdasarkan persentase akurasi dan kesalahan dapat disimpulkan bahwa arsitektur *VGG-16Net* mampu melakukan klasifikasi varietas Beras Basmathi, IR-64, dan Ketan dengan tingkat akurasi yang tinggi dengan nilai kesalahan yang rendah.

2. Hasil pelatihan dan pengujian pada arsitektur *VGG-16Net* dengan kualitas gambar buruk

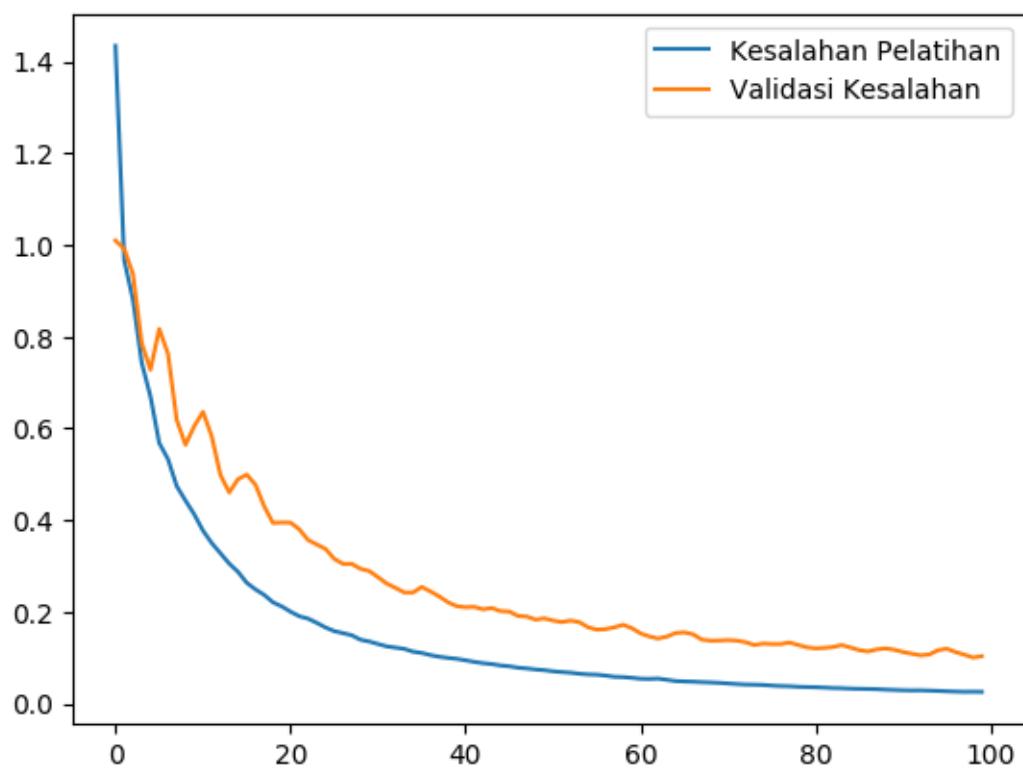
Berikut hasil pelatihan (*training*) *dataset* pelatihan dan validasi yang ditampilkan pada gambar 4.9 berikut.



Gambar 4.9 Kurva akurasi pelatihan dan validasi pada VGG-16Net pada gambar buruk

Dari hasil pelatihan tersebut terlihat sama pada *epoch* terakhir nilainya sebesar 1.0 baik pada akurasi pelatihan maupun validasinya. Namun peningkatan akurasinya sedikit lebih lama dibandingkan dengan data kualitas baik. Hal tersebut dikarenakan *dataset* dengan kualitas buruk terdapat sedikit *noise* seperti adanya bayangan dan kurangnya pencahayaan. Namun pada arsitektur ini masih dapat membedakan klasifikasinya karena *VGG-16Net* merupakan jaringan yang sangat mendalam dan sangat luas penggunaannya untuk klasifikasi gambar sehingga masih mampu untuk melakukan klasifikasi dengan baik.

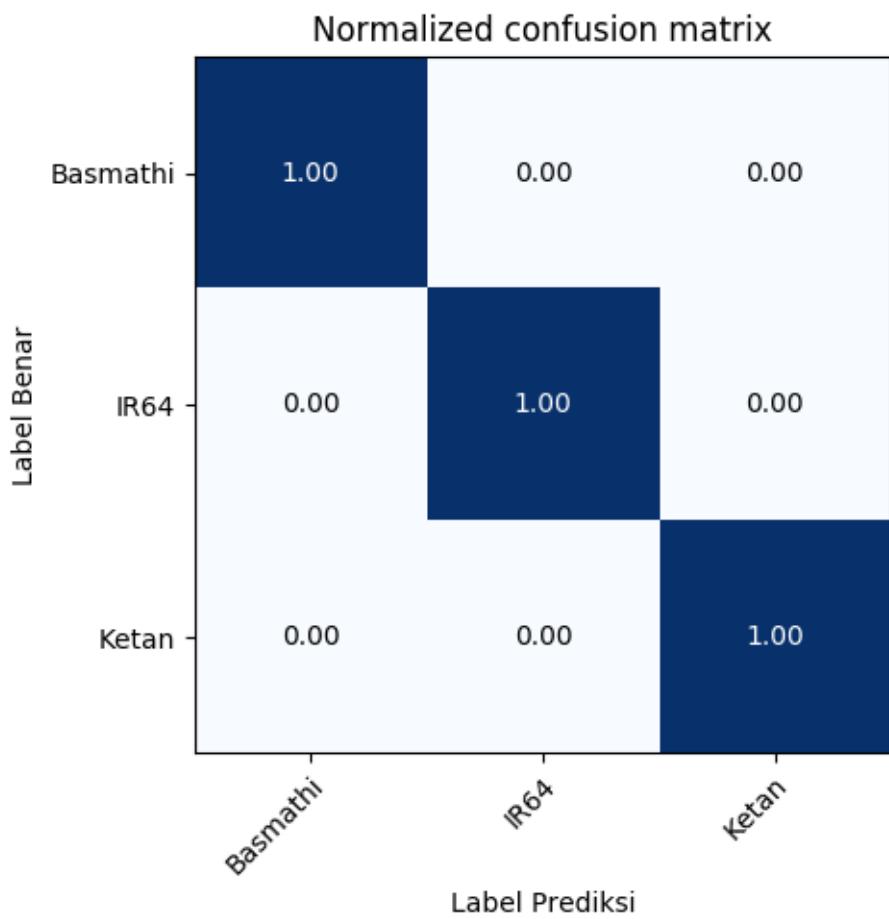
Adapun grafik kesalahan yang dihasilkan selama pelatihan seperti pada gambar 4.10 berikut.



Gambar 4.10 Kurva kesalahan pelatihan dan validasi pada VGG-16Net pada gambar buruk

Dari hasil pelatihan arsitektur VGG-16Net dengan *dataset* yang buruk menghasilkan nilai kesalahan pelatihan sebesar 0.0235 dan kesalahan validasinya sebesar 0.1037 dengan nilai kesalahan validasi yang sedikit di atas kesalahan pelatihan. Berdasarkan kurva kesalahan tersebut disimpulkan bahwa ada sedikit kondisi *underfitting* dimana kesalahan validasi sedikit lebih besar dibandingkan kesalahan pelatihannya.

Selanjutnya dapat dilihat juga hasil prediksinya dengan *confusion matrix* pada gambar 4.11.



Gambar 4.11 Confusion matrix hasil pengujian pada VGG-16Net dengan gambar buruk

Berdasarkan *confusion matrix* tersebut dapat dilihat nilai pada seluruh label prediksi pada Basmathi, IR64, dan Ketan bernilai 1.00 yang berarti bahwa seluruh hasil prediksi sesuai dengan label varietas berasnya. Hasilnya hanya sedikit berbeda dengan *dataset* kualitas gambar baik. Namun pada kualitas gambar yang buruk hasilnya lebih baik daripada dengan *dataset* gambar buruk. Hal tersebut kemungkinan terjadi karena *dataset* pengujian yang diambil masih cukup mirip dibandingkan *dataset* pengujian pada kualitas gambar baik sehingga dapat diklasifikasikan sesuai dengan labelnya.

Selain itu dapat diketahui persentase akurasi dan kesalahan dari hasil prediksi ketiga varietas beras sebagai berikut.

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

$$\% \text{akurasi} = \frac{45}{45} \times 100\% = 100\%$$

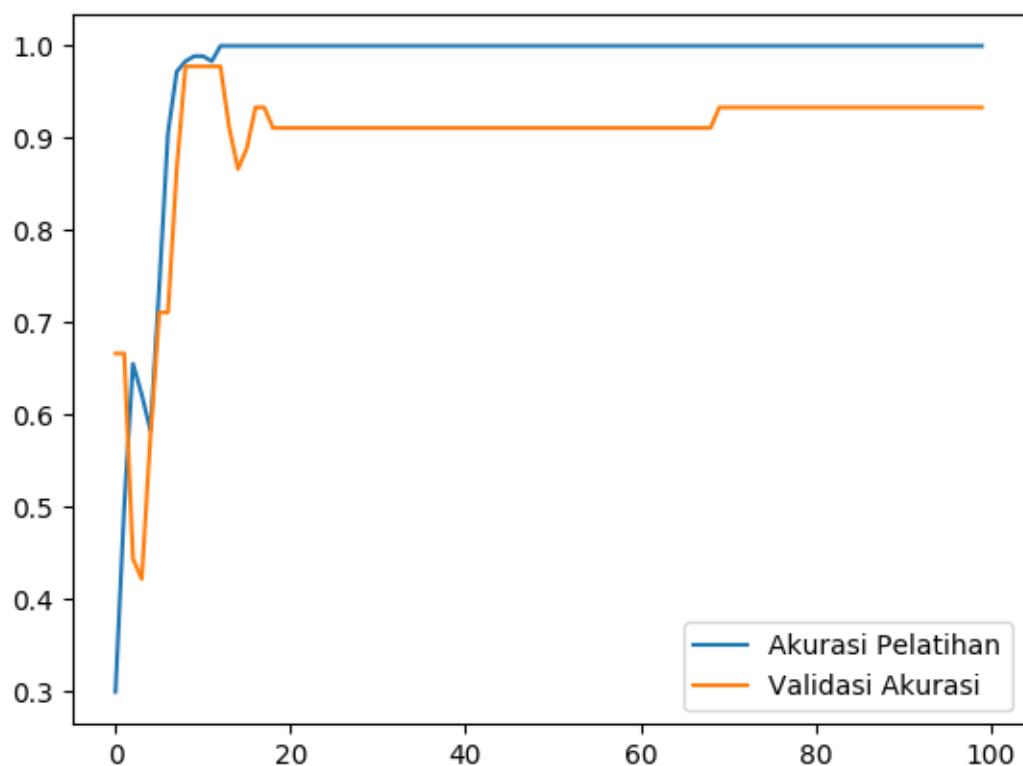
$$\% \text{kesalahan} = \frac{\text{jumlahprediksisalah}}{\text{jumlahdatapengujian}} \times 100\%$$

$$\% \text{kesalahan} = \frac{0}{45} \times 100\% = 0\%$$

Berdasarkan persentase akurasi dan kesalahan dapat disimpulkan bahwa arsitektur *VGG-16Net* masih mampu melakukan klasifikasi varietas Beras Basmathi, IR-64, dan Ketan dengan tingkat akurasi yang tinggi dengan nilai kesalahan yang rendah walaupun dengan *dataset* yang tidak bagus kualitasnya.

3. Hasil pelatihan dan pengujian pada arsitektur *MobileNeV1* dengan kualitas gambar baik

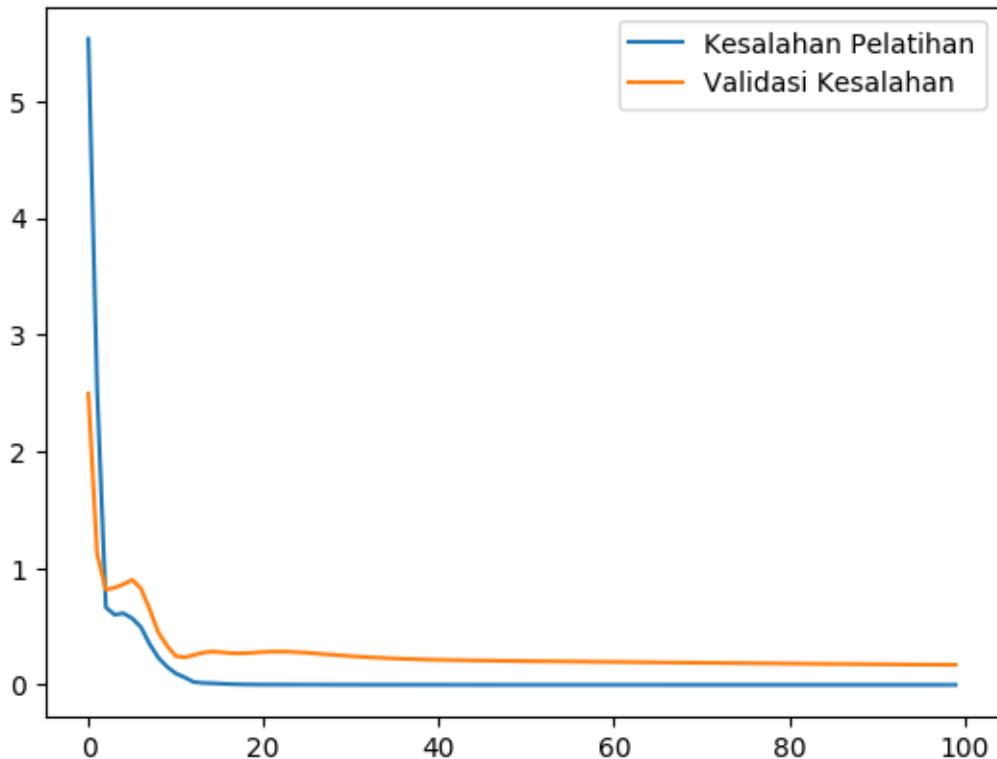
Berikut hasil pelatihan dari *dataset* pelatihan dan validasi yang dapat dilihat pada gambar 4.12.



Gambar 4.12 Kurva akurasi hasil pelatihan dan validasi pada MobileNetV1

Berdasarkan hasil tersebut nilai akurasi pelatihan yang diperoleh menggunakan arsitektur *MobileNetV1* mencapai 1.0 dan nilai akurasi validasi mencapai sekitar 0.9333. Dari kurva tersebut dapat disimpulkan bahwa klasifikasi yang dilakukan dengan *MobileNetV1* cukup baik karena hasil akurasi validasi terus meningkat mendekati nilai akurasi pelatihan dan hanya terdapat sedikit perbedaan. Hasil tersebut dapat dilihat bahwa tingkat akurasi yang sedikit lebih rendah dibandingkan menggunakan arsitektur *VGG-16Net*. Hal tersebut disebabkan oleh *depthwise separable convolution* yang dapat menghilangkan sebagian informasi penting dari konvolusi standar sehingga dapat menurunkan tingkat akurasi.

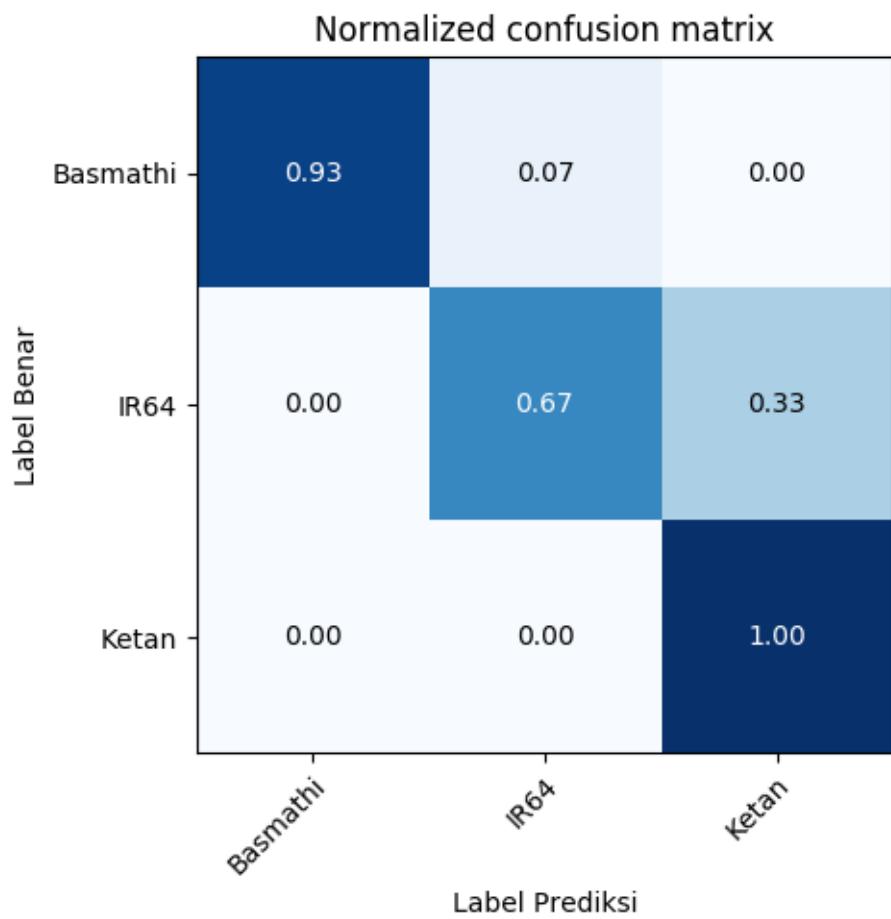
Terdapat juga kurva kesalahan yang dihasilkan selama pelatihan yang divisualisasikan pada gambar 4.13.



Gambar 4.13 Kurva tingkat kesalahan pelatihan dan validasi pada MobileNetV1

Dari hasil pelatihan arsitektur MobileNetV1 menghasilkan nilai kesalahan pelatihan 0.0002 dan nilai validasi kesalahan sedikit di atas kesalahan pelatihan yaitu sebesar 0.1722. Dari kurva tersebut juga dapat dilihat bahwa kurva kesalahan pelatihan hampir sama penurunannya dengan kurva validasi kesalahan sehingga dapat disimpulkan hasil pelatihannya hanya sedikit terjadi *underfitting*.

Untuk *confusion matrix* dari hasil pengujiannya disajikan pada gambar 4.14 sebagai berikut.



Gambar 4.14 Confusion matrix hasil pengujian pada MobileNetV1

Berdasarkan *confusion matrix* tersebut dapat dilihat nilai pada label prediksi Basmathi sebesar 0.93 pada label benar Basmathi, IR64 sebesar 0.67 pada label benar IR64 dan Ketan sebesar 1.00 pada label benar Ketan. Masing-masing nilai label prediksi tersebut merupakan tingkat keberhasilan prediksi dari 15 *dataset* pengujian pada setiap varietas berasnya. Pada pengujian Beras Basmathi terdapat kesalahan yang memprediksi Beras Ketan sebesar 0.07 atau 7% dari seluruh *dataset* pengujinya, pengujian Beras IR64 terdapat kesalahan yang memprediksi Beras Ketan sebesar 0.33 atau 33% dari dataset pengujinya, dan untuk Beras Ketan tidak terdapat kesalahan prediksi. Kesalahan prediksi Beras

IR-64 yang lebih besar dari Beras Basmathi dan Ketan disebabkan karena pada dataset pengujinya, penyebaran setiap varietas beras yang diubah-ubah bentuknya, pengambilan gambar yang kurang tepat jaraknya dan kurang bagus kualitasnya, serta struktur beras IR-64 dan Ketan yang agak mirip.

Untuk mengetahui tingkat keberhasilan dan tingkat kesalahan prediksi ketiga varietas beras tersebut dapat ditentukan persentasenya seperti berikut.

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

$$\% \text{akurasi} = \frac{39}{45} \times 100\% = 86,67\%$$

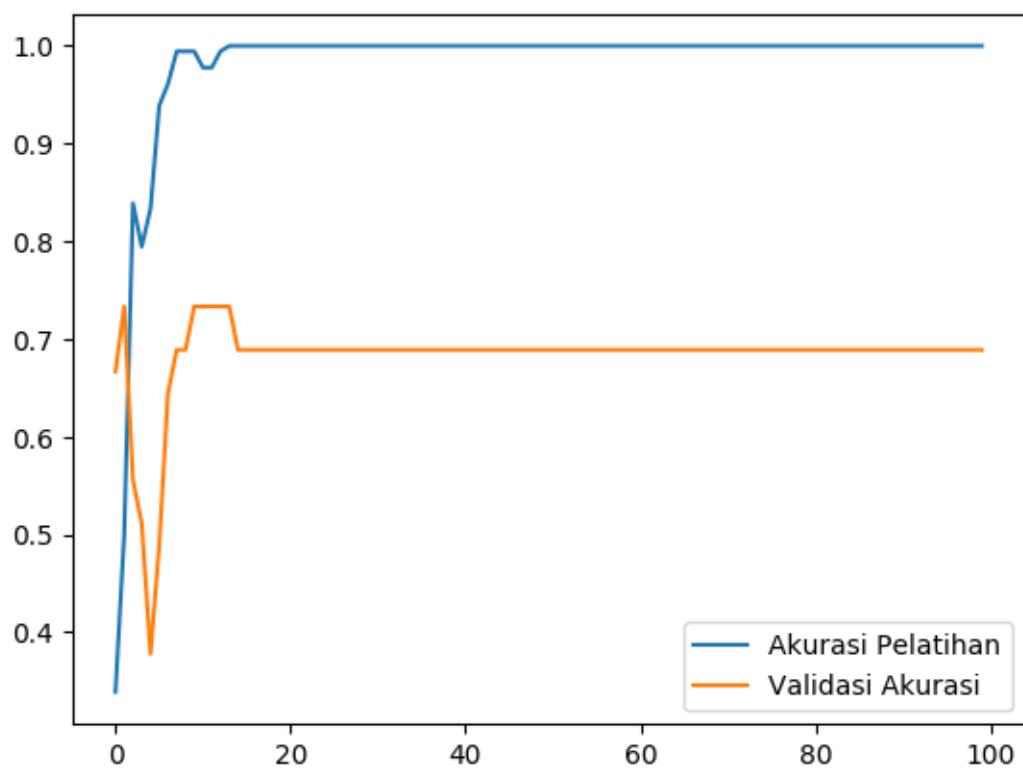
$$\% \text{kesalahan} = \frac{\text{jumlahprediksisalah}}{\text{jumlahdatapengujian}} \times 100\%$$

$$\% \text{kesalahan} = \frac{3}{45} \times 100\% = 13,33\%$$

Berdasarkan persentase akurasi dan kesalahan dapat disimpulkan bahwa arsitektur *MobileNetV1* mampu melakukan klasifikasi varietas Beras Basmathi, IR-64, dan Ketan dengan tingkat akurasi yang tinggi dengan nilai kesalahan yang cukup rendah. Namun masih lebih rendah daripada arsitektur *VGG-16Net*.

4. Hasil pelatihan dan pengujian pada arsitektur *MobileNeV1* dengan kualitas gambar buruk

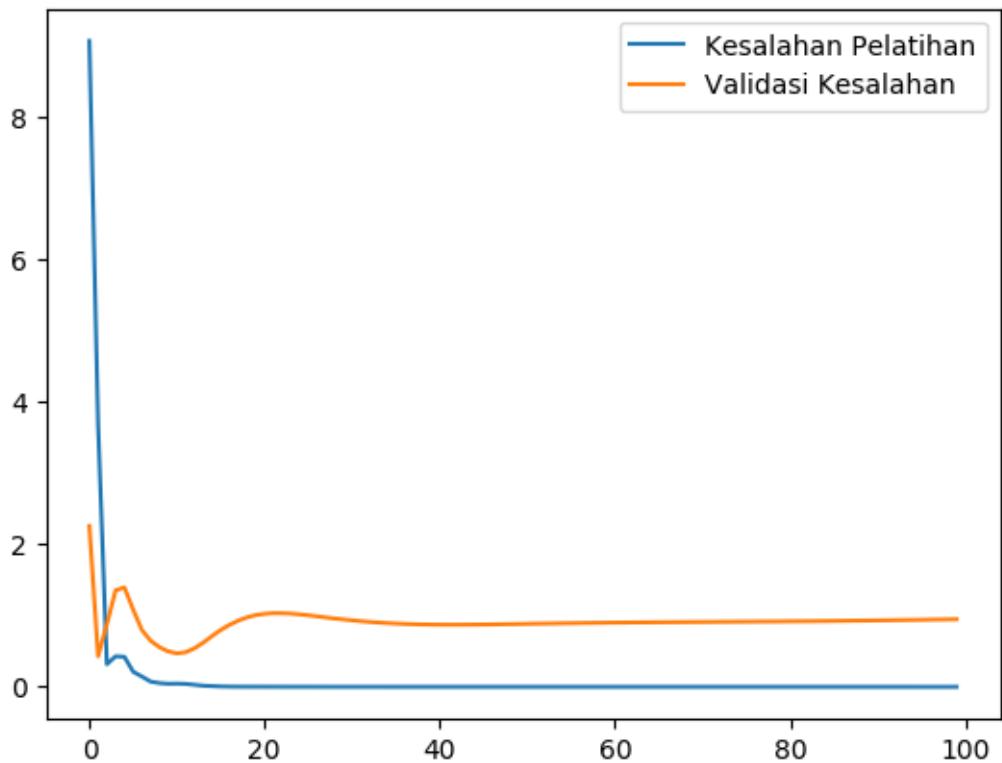
Berikut hasil pelatihan dari *dataset* pelatihan dan validasi yang dapat dilihat pada gambar 4.15.



Gambar 4.15 Kurva akurasi pelatihan dan validasi pada MobileNetV1 pada gambar buruk

Dari hasil pelatihan arsitektur MobileNetV1 dengan kualitas *dataset* yang buruk menghasilkan nilai akurasi pelatihan mencapai 1.0 tetapi nilai akurasi validasinya jauh di bawah akurasi pelatihannya yaitu hanya mencapai 0.6889 di *epoch* terakhir. Hal itu menunjukkan hasil pelatihan yang tidak baik dan terjadinya *underfitting* sehingga dapat dikatakan jaringan CNN tersebut tidak dapat melakukan klasifikasinya dengan baik.

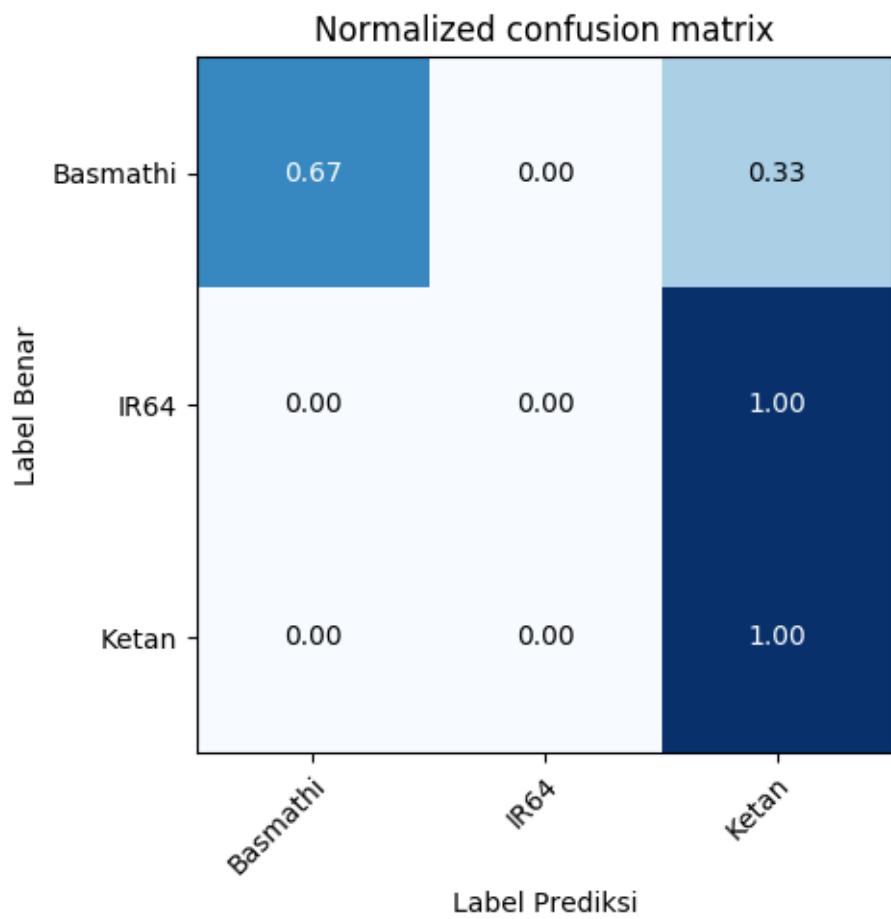
Kemudian diperoleh juga kurva kesalahan pelatihan dan validasinya pada gambar 4.16 berikut.



Gambar 4.16 Kurva kesalahan pelatihan dan validasi pada MobileNetV1 pada gambar buruk

Dari hasil pelatihan arsitektur MobileNetV1 menghasilkan nilai kesalahan sebesar 0.0001 dan nilai validasi kesalahan yang cukup besar yaitu sebesar 0.9485. Dari kurva tersebut juga dapat dilihat bahwa kurva kesalahan pelatihan semakin berkurang mendekati 0, namun pada kurva kesalahan validasinya tidak menurun mendekati 0 seiring bertambahnya *epoch*. Sehingga diperoleh bahwa MobileNetV1 tidak dapat melakukan klasifikasi dengan baik pada dataset yang buruk dan terjadinya *underfitting*, yaitu nilai kesalahan validasi yang jauh lebih besar dibandingkan nilai kesalahan pelatihannya.

Selanjutnya terdapat hasil prediksi dari pengujian yang ditampilkan dengan *confusion matrix* pada gambar 4.17.



Gambar 4.17 Confusion matrix hasil pengujian pada MobileNetV1 dengan gambar buruk

Berdasarkan *confusion matrix* tersebut dapat dilihat pada label benar Basmathi hasil prediksi yang benar 67% dan kesalahan memprediksi Ketan sebesar 33%, pada label benar IR64 tidak ada hasil prediksi yang benar melainkan memprediksi Ketan 100%, sedangkan pada Ketan seluruh hasil prediksinya sesuai dengan label benar. Masing-masing nilai label prediksi tersebut merupakan tingkat keberhasilan prediksi dari 15 *dataset* pengujian pada setiap varietas berasnya.

Persentase tingkat keberhasilan dan kesalahan dalam pengujian tersebut dapat dihitung dengan cara sebagai berikut.

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

$$\% \text{akurasi} = \frac{25}{45} \times 100\% = 55,56\%$$

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

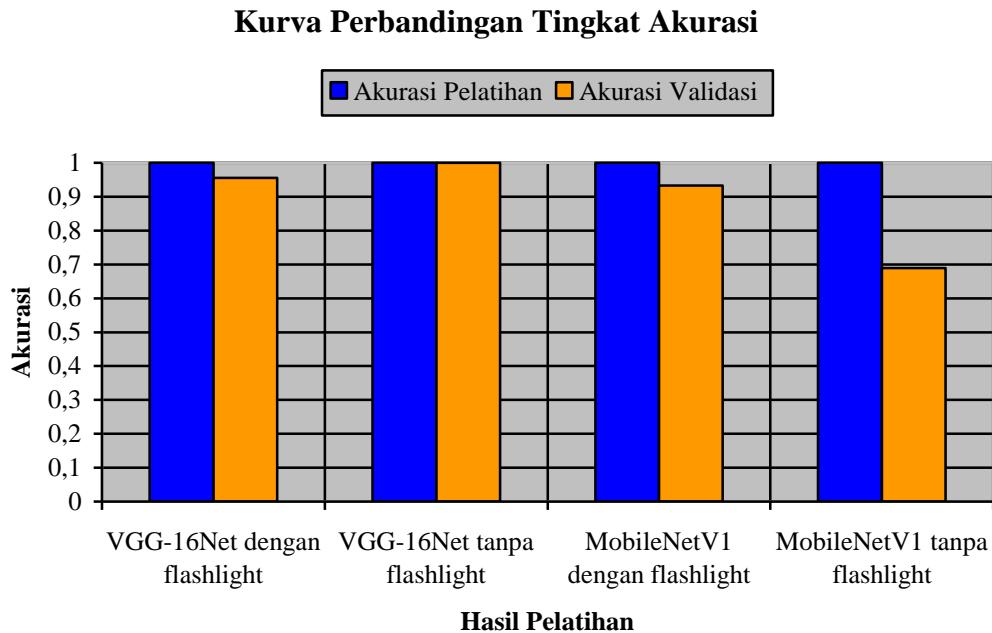
$$\% \text{kesalahan} = \frac{20}{45} \times 100\% = 44,44\%$$

Berdasarkan persentase akurasi dan kesalahan dapat dilihat persentase akurasi yang jauh lebih rendah dari persentase kesalahannya. Sehingga dapat disimpulkan bahwa arsitektur *MobileNetV1* tidak mampu melakukan klasifikasi varietas Beras Basmathi, IR-64, dan Ketan dengan dataset kualitas gambar yang tidak baik. Karena arsitektur *MobileNetV1* tidak dapat melakukan klasifikasi yang cukup mendalam pada obyek yang hanya terdapat sedikit perbedaan serta banyaknya *noise* pada gambar karena arsitektur tersebut hanya ditujukan untuk kebutuhan komputasi yang cepat. Oleh karena itu hasil model pelatihan pada pengujian arsitektur *MobileNetV1* dengan *dataset* yang buruk tidak dilanjutkan pada perangkat Android karena sudah dipastikan tidak akan akurat hasil prediksi klasifikasinya.

4.2.5 Kurva Perbandingan Hasil Pelatihan

Untuk mengetahui perbedaan tingkat akurasi pelatihan dan validasi serta tingkat kesalahan pelatihan dan validasi dari hasil pelatihan yang dilakukan maka dibuatlah grafik perbandingan tingkat akurasi dan tingkat kesalahan dari setiap

hasil pelatihannya. Kurva tingkat akurasi hasil pelatihannya dapat dilihat pada gambar 4.18 sebagai berikut.

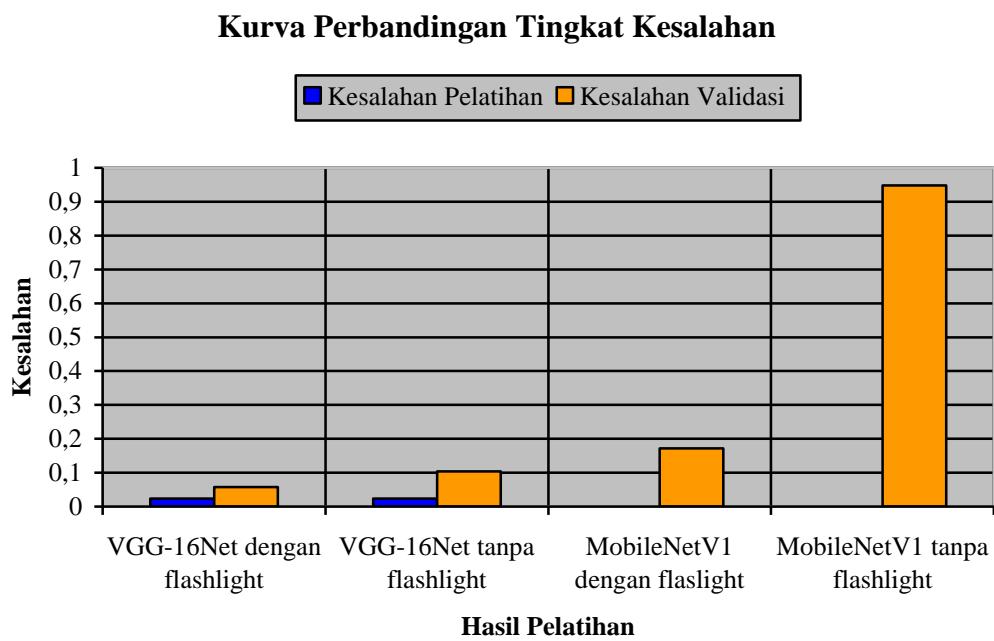


Gambar 4.18 Kurva perbandingan tingkat akurasi dari hasil pelatihan

Berdasarkan kurva tersebut dapat dilihat bahwa seluruh tingkat akurasi

pelatihan mencapai nilai maksimumnya sebesar 1. Tingkat akurasi validasi yang cukup stabil dan baik dihasilkan oleh arsitektur *VGG-16Net* sedangkan pada *MobileNetV1* sangat rendah pada kualitas *dataset* yang buruk. Semakin tinggi tingkat akurasi dan semakin sedikit perbedaan antara akurasi pelatihan dan validasinya, maka arsitektur CNN dapat melakukan klasifikasi dengan baik dan dapat disimpulkan hasil pelatihan tersebut dalam kondisi *goodfit*. Jika nilai akurasi pelatihan jauh lebih rendah daripada nilai akurasi validasinya maka terjadinya kondisi *overfitting* dan sebaliknya jika akurasi validasi jauh lebih rendah daripada nilai akurasi pelatihannya maka terjadinya kondisi *underfitting*.

Kemudian diperoleh juga kurva tingkat kesalahan yang dihasilkan dari pelatihan tersebut pada gambar 4.19.



Gambar 4.19 Kurva perbandingan tingkat kesalahan dari hasil pelatihan

Berdasarkan kurva tersebut dapat dilihat bahwa perbedaan antara tingkat kesalahan pelatihan dan validasi pada arsitektur *VGG-16Net* relatif lebih kecil dibandingkan *MobileNetV1*. Selain itu pada *MobileNetV1* dengan *dataset* yang buruk diperoleh tingkat kesalahan validasi yang paling tinggi. Semakin kecil tingkat kesalahan dan semakin sedikit perbedaan antara kesalahan pelatihan dan validasinya, maka arsitektur CNN dapat melakukan klasifikasi dengan baik dan dapat disimpulkan hasil pelatihan tersebut dalam kondisi *goodfit*. Jika nilai kesalahan pelatihan jauh lebih rendah daripada nilai akurasi validasinya maka terjadinya kondisi *overfitting* dan sebaliknya jika nilai kesalahan validasi jauh lebih rendah daripada nilai akurasi pelatihannya maka terjadinya kondisi *underfitting*.

4.3 Pengujian pada Perangkat *Android*

Pengujian pada perangkat *Android* dilakukan setelah melakukan pengujian pada *Google Colaboratory*. Pengujian ini dilakukan menggunakan model hasil pelatihan dari *Google Colaboratory* pada masing-masing arsitektur dan variasi *dataset* yang digunakan. Model hasil pelatihan tersebut kemudian diimpor ke dalam *folder assets* pada *file* proyek *Android Studio* yang akan dibuat. Setelah diimpor dilanjutkan membuat program untuk aplikasi *Android*-nya pada *Android Studio*. Pengujian tersebut dilakukan terhadap tiga varietas beras, arsitektur yang digunakan, menggunakan cahaya *flashlight* dan cahaya ruangan, serta kondisi beras yang disebarluaskan dengan alas berwarna hitam seperti pada *dataset* dan dibungkus dengan plastik bening. Beberapa *screenshot* hasil pengujian tersebut dapat dilihat pada lampiran yang tersedia di penelitian ini.

4.3.1 Hasil Pengujian pada Arsitektur *VGG-16Net* dengan *Flashlight*

Pengujian yang pertama dilakukan menggunakan model *TensorFlow Lite* arsitektur *VGG-16Net* dengan *dataset* kualitas gambar yang baik menggunakan cahaya *flashlight* pada *smartphone*. Pengujian ini dilakukan dengan menyebarluaskan beras pada alas berwarna hitam di ruangan yang cukup redup agar *flashlight* pada perangkat *Android* aktif secara otomatis. Hal tersebut dilakukan karena aplikasi yang dibuat menggunakan fitur *flashlight* otomatis pada kamera *smartphone*. Pada pengujian yang pertama ini setiap varietas beras disebarluaskan pada alas berwarna hitam dengan mengubah-ubah bentuk sebaran varietas berasnya. Jarak antara obyek beras dengan kamera yang diatur dalam pengujian ini sekitar 10 cm untuk menjaga konsistensi hasil pengujiannya. Hasil pengujian pada perangkat *Android*

ini mengambil hasil prediksi dari persentase probabilitas yang tertinggi untuk hasil prediksi yang benar. Berikut hasil pengujian pada tabel 4.1.

Tabel 4.1 Hasil pengujian arsitektur VGG-16Net dengan flashlight

Pengujian ke-	Respon Deteksi	Intensitas Cahaya Flashlight/Ruang	Varietas Beras	Hasil Prediksi (Tingkat Probabilitas)	Benar/Salah
1	6419 ms	1070 lux	Basmathi	Basmathi (91,41%)	Benar
2	4976 ms	1070 lux	Basmathi	IR-64 (94,25%)	Salah
3	4827 ms	1070 lux	Basmathi	Basmathi (87,72%)	Benar
4	4924 ms	1070 lux	Basmathi	Basmathi (95,3%)	Benar
5	4930 ms	1070 lux	Basmathi	Basmathi (93,63%)	Benar
6	5203 ms	1070 lux	Basmathi	Basmathi (89,28%)	Benar
7	4930 ms	1070 lux	Basmathi	Basmathi (99,72%)	Benar
8	5107 ms	1070 lux	Basmathi	Basmathi (96,27%)	Benar
9	4930 ms	1070 lux	Basmathi	Basmathi (77,12%)	Benar
10	4974 ms	1070 lux	Basmathi	Basmathi (99,95%)	Benar
11	4879 ms	1070 lux	Basmathi	Basmathi (78,28%)	Benar
12	5139 ms	1070 lux	Basmathi	IR-64 (93,75%)	Salah
13	4993 ms	1070 lux	Basmathi	Basmathi (94,33%)	Benar
14	4998 ms	1070 lux	Basmathi	Basmathi (96,54%)	Benar
15	4936 ms	1070 lux	Basmathi	Basmathi (89,5%)	Benar
16	6209 ms	1070 lux	IR-64	IR-64 (94,58%)	Benar
17	5058 ms	1070 lux	IR-64	IR-64 (99,74%)	Benar
18	5020 ms	1070 lux	IR-64	IR-64 (98,22%)	Benar
19	5029 ms	1070 lux	IR-64	IR-64 (99,59%)	Benar
20	5275 ms	1070 lux	IR-64	IR-64 (99,75%)	Benar
21	5186 ms	1070 lux	IR-64	IR-64 (96,18%)	Benar
22	5046 ms	1070 lux	IR-64	IR-64 (99,91%)	Benar
23	5428 ms	1070 lux	IR-64	IR-64 (70,51%)	Benar
24	4892 ms	1070 lux	IR-64	IR-64 (99,63%)	Benar
25	5274 ms	1070 lux	IR-64	IR-64 (97,46%)	Benar
26	4903 ms	1070 lux	IR-64	IR-64 (97,72%)	Benar
27	5022 ms	1070 lux	IR-64	IR-64 (99,92%)	Benar
28	4911 ms	1070 lux	IR-64	IR-64 (84,25%)	Benar
29	5056 ms	1070 lux	IR-64	IR-64 (99,37%)	Benar
30	5172 ms	1070 lux	IR-64	IR-64 (99,94%)	Benar
31	6116 ms	1070 lux	Ketan	Ketan (97,24%)	Benar
32	5009 ms	1070 lux	Ketan	IR-64 (98,83%)	Salah
33	4894 ms	1070 lux	Ketan	Ketan (57,51%)	Benar
34	4936 ms	1070 lux	Ketan	IR-64 (73,96%)	Salah
35	5004 ms	1070 lux	Ketan	IR-64 (75,4%)	Salah
36	4913 ms	1070 lux	Ketan	Ketan (69,02%)	Benar
37	4941 ms	1070 lux	Ketan	IR-64 (99,92%)	Salah

38	5010 ms	1070 lux	Ketan	Basmathi (45,14%)	Salah
39	5029 ms	1070 lux	Ketan	Ketan (52,73%)	Benar
40	5234 ms	1070 lux	Ketan	IR-64 (90,8%)	Salah
41	4980 ms	1070 lux	Ketan	Ketan (94,4%)	Benar
42	4887 ms	1070 lux	Ketan	IR-64 (97,75%)	Salah
43	4979 ms	1070 lux	Ketan	IR-64 (93,51%)	Salah
44	5037 ms	1070 lux	Ketan	Ketan (58,43%)	Benar
45	4994 ms	1070 lux	Ketan	IR-64 (90,82%)	Salah

Dari tabel pengujian tersebut dapat diketahui bahwa hasil prediksi untuk Beras Basmathi dan IR-64 cukup akurat karena pada Beras Basmathi hanya terdapat 2 kesalahan dan pada Beras IR-64 benar semua. Selain itu juga tingkat probabilitas dari Beras Basmathi dan IR-64 sudah akurat dan stabil di angka 90%. Akan tetapi untuk Beras Ketan kesalahan prediksi yang dihasilkan cukup besar dan tingkat probabilitasnya yang tidak stabil. Respon deteksi pengujian dengan arsitektur *VGG-16Net* pada perangkat *Android* sangat lama yaitu sekitar 4 sampai dengan 6 detik. Hal itu dikarenakan arsitektur *VGG-16Net* memiliki pembelajaran yang sangat mendalam sehingga membutuhkan komputasi yang sangat banyak yang menyebabkan prosesor bekerja lebih lama untuk melakukan klasifikasinya. Kemudian untuk intensitas cahaya yang dihasilkan oleh *flashlight* pada jarak 10 cm sekitar 1070 lux. Setelah diperoleh tabel hasil pengujian maka dapat dihitung persentase akurasi dan persentase kesalahan pengujian untuk seluruh varietas berasnya sebagai berikut.

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

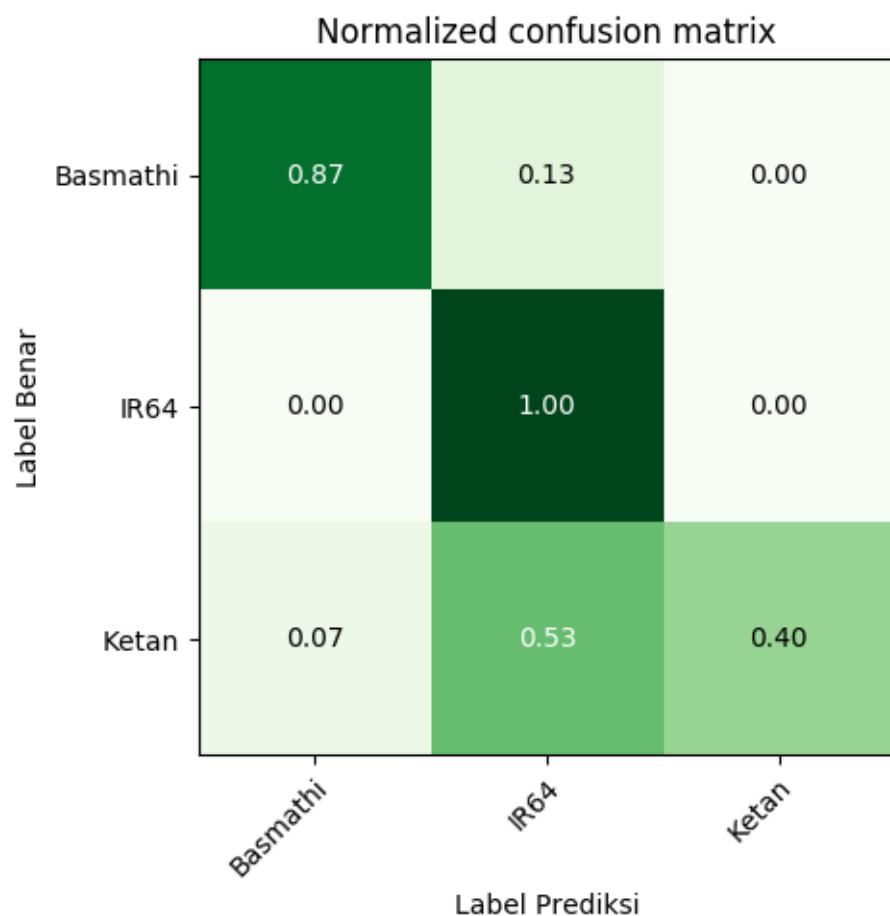
$$\% \text{akurasi} = \frac{34}{45} \times 100\% = 75,56\%$$

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

$$\% \text{kesalahan} = \frac{11}{45} \times 100\% = 24,44\%$$

Dari hasil perhitungan tersebut persentase akurasi cukup tinggi dan persentase kesalahannya cukup rendah, namun hal tersebut belum dapat memastikan tingkat akurasi klasifikasi pada setiap varietas berasnya.

Oleh karena itu perlu ditampilkan hasilnya dengan *confusion matrix* yang disajikan pada gambar 4.20 untuk mengetahui tingkat akurasi hasil prediksi dari setiap varietas beras.



Gambar 4.20 Confusion matrix pengujian arsitektur VGG-16Net pada Android dengan flashlight

Berdasarkan *confusion matrix* tersebut tingkat akurasi hasil prediksi Basmathi yang benar mencapai 87%, IR64 mencapai 100%, dan Ketan hanya 40% dari jumlah pengujian setiap varietas berasnya. Sehingga dapat disimpulkan bahwa arsitektur *VGG-16Net* pada perangkat *Android* kurang akurat dan respon deteksinya yang sangat lambat. Tingkat akurasi yang menurun dibandingkan dengan hasil pengujian pada *Google Colaboratory* disebabkan karena model hasil pelatihan *VGG-16Net* tidak cukup bagus untuk dikonversi ke dalam bentuk *file* ekstensi *TensorFlow Lite* akibat adanya beberapa fungsi yang tidak cocok untuk dikonversi ke format “.tflite” sehingga fungsi-fungsi tersebut diabaikan dan tidak disimpan ke dalam format tersebut. Oleh karena itu pengujian dengan arsitektur *VGG-16Net* pada perangkat *Android* tidak dilanjutkan pada pengujian selanjutnya karena sudah dapat dipastikan hasilnya tidak akan akurat jika modelnya sudah dikonversi dengan *format file* “.tflite”.

4.3.2 Hasil Pengujian pada Arsitektur *MobileNetV1* dengan *Flashlight*

Pada pengujian ketiga dilakukan menggunakan model *TensorFlow Lite* arsitektur *MobileNetV1* dengan *dataset* kualitas yang baik menggunakan cahaya *flashlight* pada *smartphone*. Pengujian ini dilakukan sama seperti pada percobaan pertama pada arsitektur *VGG-16Net*. Hasil pengujian dapat dilihat pada tabel 4.2 berikut.

*Tabel 4.2 Hasil pengujian arsitektur *MobileNetV1* dengan *flashlight**

Pengujian ke-	Respon Deteksi	Intensitas Cahaya <i>Flashlight/Ruang</i>	Varietas Beras	Hasil Prediksi (Tingkat Probabilitas)	Benar/Salah
1	1344 ms	1070 lux	Basmathi	Basmathi (99,29%)	Benar
2	983 ms	1070 lux	Basmathi	Basmathi (92,65%)	Benar
3	957 ms	1070 lux	Basmathi	Basmathi (92,22%)	Benar
4	954 ms	1070 lux	Basmathi	Basmathi (76%)	Benar

5	976 ms	1070 lux	Basmathi	Basmathi (99,65%)	Benar
6	997 ms	1070 lux	Basmathi	Basmathi (57,12%)	Benar
7	972 ms	1070 lux	Basmathi	Basmathi (98,21%)	Benar
8	963 ms	1070 lux	Basmathi	Basmathi (87,28%)	Benar
9	1000 ms	1070 lux	Basmathi	Basmathi (90,92%)	Benar
10	1043 ms	1070 lux	Basmathi	Basmathi (83,49%)	Benar
11	994 ms	1070 lux	Basmathi	Basmathi (59,38%)	Benar
12	952 ms	1070 lux	Basmathi	Basmathi (86,66%)	Benar
13	1004 ms	1070 lux	Basmathi	Basmathi (92,53%)	Benar
14	990 ms	1070 lux	Basmathi	Basmathi (99,18%)	Benar
15	1055 ms	1070 lux	Basmathi	Basmathi (97,22%)	Benar
16	1380 ms	1070 lux	IR-64	IR-64 (53,55%)	Benar
17	975 ms	1070 lux	IR-64	IR-64 (97,73%)	Benar
18	998 ms	1070 lux	IR-64	Ketan (67,19%)	Salah
19	1011 ms	1070 lux	IR-64	IR-64 (94,69%)	Benar
20	996 ms	1070 lux	IR-64	IR-64 (94,92%)	Benar
21	957 ms	1070 lux	IR-64	IR-64 (73,82%)	Benar
22	993 ms	1070 lux	IR-64	IR-64 (98,52%)	Benar
23	975 ms	1070 lux	IR-64	IR-64 (99,47%)	Benar
24	1079 ms	1070 lux	IR-64	IR-64 (92,3%)	Benar
25	1080 ms	1070 lux	IR-64	IR-64 (98,78%)	Benar
26	948 ms	1070 lux	IR-64	IR-64 (85,53%)	Benar
27	974 ms	1070 lux	IR-64	IR-64 (99,75%)	Benar
28	966 ms	1070 lux	IR-64	IR-64 (95,28%)	Benar
29	999 ms	1070 lux	IR-64	IR-64 (91,99%)	Benar
30	1056 ms	1070 lux	IR-64	Ketan (68,59%)	Salah
31	1391 ms	1070 lux	Ketan	Ketan (98,84%)	Benar
32	955 ms	1070 lux	Ketan	Ketan (72,66%)	Benar
33	969 ms	1070 lux	Ketan	Ketan (99,49%)	Benar
34	978 ms	1070 lux	Ketan	Ketan (95,18%)	Benar
35	1005 ms	1070 lux	Ketan	Ketan (86,3%)	Benar
36	962 ms	1070 lux	Ketan	Ketan (99,14%)	Benar
37	960 ms	1070 lux	Ketan	Ketan (79,32%)	Benar
38	1004 ms	1070 lux	Ketan	Ketan (93,6%)	Benar
39	1071 ms	1070 lux	Ketan	Ketan (99,68%)	Benar
40	1066 ms	1070 lux	Ketan	Ketan (99,92%)	Benar
41	988 ms	1070 lux	Ketan	Ketan (98,58%)	Benar
42	956 ms	1070 lux	Ketan	Ketan (83,58%)	Benar
43	1003 ms	1070 lux	Ketan	Ketan (70,94%)	Benar
44	993 ms	1070 lux	Ketan	Ketan (99,19%)	Benar
45	1056 ms	1070 lux	Ketan	Ketan (99,61%)	Benar

Dari tabel pengujian tersebut dapat dilihat bahwa hasil prediksi untuk ketiga varietas beras sangat akurat dan hanya terdapat 2 kesalahan prediksi pada

varietas Beras IR-64. Kedua kesalahan tersebut tidak terlalu tinggi hanya sekitar 68% saja yang menunjukan bahwa model arsitektur *MobileNet* dengan kualitas gambar baik dapat melakukan klasifikasinya dengan tingkat akurasi yang tinggi pada perangkat *Android*. Tingkat probabilitasnya pun sudah cukup akurat sekitar 90%, hanya sedikit prediksi yang benar dengan tingkat probabilitas di bawah 70%. Hal itu disebabkan oleh pengaturan jarak antar obyek dengan kamera yang kurang tepat. Respon deteksi pengujian dengan arsitektur *MobileNetV1* pada perangkat *Android* sangat cepat hanya sekitar 1 detik tetapi respon deteksi juga dipengaruhi oleh prosesor yang digunakan. Hal itu dikarenakan adanya metode konvolusi *depthwise separable convolution* pada arsitektur *MobileNet*. Selain itu hasil prediksi yang diperoleh cukup baik hampir sama dengan menggunakan *Google Colaboratory* karena arsitektur *MobileNet* memang didesain untuk kebutuhan komputasi yang rendah sehingga model pelatihannya dapat dikonversi ke dalam *format TensorFlow Lite* dengan baik tanpa adanya penurunan tingkat akurasi yang signifikan. Hasil pengujian tersebut sedikit lebih baik dari pada di *Google Colaboratory* karena adanya kemungkinan jarak kamera dengan obyek beras yang relatif tepat. Setelah diperoleh tabel hasil pengujiannya maka dapat dihitung persentase akurasi dan persentase kesalahan pengujian untuk seluruh varietas berasnya sebagai berikut.

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

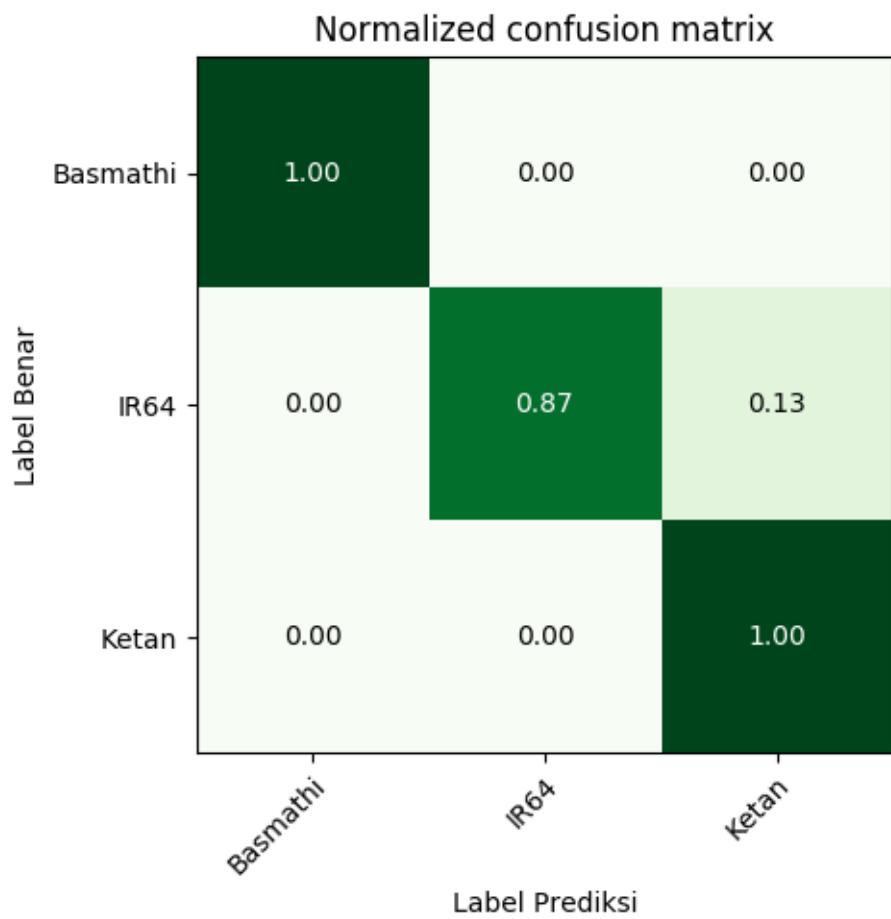
$$\% \text{akurasi} = \frac{43}{45} \times 100\% = 95,56\%$$

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

$$\% kesalahan = \frac{2}{45} \times 100\% = 4,44\%$$

Dari hasil perhitungan tersebut persentase akurasi cukup tinggi dan persentase kesalahannya cukup rendah, namun hal tersebut belum dapat memastikan tingkat akurasi klasifikasi pada setiap varietas berasnya.

Oleh karena itu perlu ditampilkan hasilnya dengan *confusion matrix* yang disajikan pada gambar 4.21 untuk mengetahui tingkat akurasi hasil prediksi dari setiap varietas beras.



Gambar 4.21 Confusion matrix pengujian arsitektur MobileNetV1 pada Android dengan flashlight

Berdasarkan *confusion matrix* tersebut tingkat akurasi hasil prediksi Basmathi dan Ketan yang benar mencapai 100%, dan untuk IR64 mencapai 87% dari jumlah pengujian setiap varietas berasnya. Sehingga dapat disimpulkan bahwa arsitektur *MobileNetV1* pada perangkat *Android* akurat untuk klasifikasi ketiga varietas beras tersebut dengan menggunakan *flashlight*. Berbeda dengan hasil pengujian pada *VGG-16Net* yang tingkat akurasinya jauh lebih rendah jika diterapkan pada perangkat *Android*.

4.3.3 Hasil Pengujian pada Arsitektur MobileNetV1 tanpa Flashlight

Pada pengujian ketiga dilakukan menggunakan model yang sama seperti pengujian kedua. Perbedaannya pengujian ini dilakukan pada kondisi intensitas cahaya ruangan yang terang tanpa menggunakan *flashlight*. Hasil pengujian dapat dilihat pada tabel 4.3 berikut.

Tabel 4.3 Hasil pengujian arsitektur MobileNetV1 tanpa flashlight

Pengujian ke-	Respon Deteksi	Intensitas Cahaya Flashlight/Ruang	Varietas Beras	Hasil Prediksi (Tingkat Probabilitas)	Benar/Salah
1	1416 ms	2110 lux	Basmathi	Basmathi (99,38%)	Benar
2	1056 ms	2110 lux	Basmathi	Basmathi (50,17%)	Benar
3	1012 ms	2110 lux	Basmathi	Basmathi (98,76%)	Benar
4	1033 ms	2110 lux	Basmathi	Basmathi (67,46%)	Benar
5	1047 ms	2110 lux	Basmathi	Basmathi (57,32%)	Benar
6	1224 ms	2110 lux	Basmathi	Basmathi (99,74%)	Benar
7	1051 ms	2110 lux	Basmathi	Basmathi (95,14%)	Benar
8	1049 ms	2110 lux	Basmathi	Basmathi (78,37%)	Benar
9	967 ms	2110 lux	Basmathi	Basmathi (56,02%)	Benar
10	1087 ms	2110 lux	Basmathi	Basmathi (78,77%)	Benar
11	1039 ms	2110 lux	Basmathi	Basmathi (99,38%)	Benar
12	1090 ms	2110 lux	Basmathi	Basmathi (97,83%)	Benar
13	1023 ms	2110 lux	Basmathi	Basmathi (83,2%)	Benar
14	980 ms	2110 lux	Basmathi	Basmathi (74,34%)	Benar
15	1054 ms	2110 lux	Basmathi	Ketan (52,96%)	Salah
16	1935 ms	2110 lux	IR-64	IR-64 (67,29%)	Benar
17	1169 ms	2110 lux	IR-64	IR-64 (99,74%)	Benar
18	1023 ms	2110 lux	IR-64	IR-64 (96,12%)	Benar
19	1066 ms	2110 lux	IR-64	IR-64 (99,41%)	Benar
20	1149 ms	2110 lux	IR-64	IR-64 (51,69%)	Benar
21	1085 ms	2110 lux	IR-64	IR-64 (45,23%)	Benar
22	1043 ms	2110 lux	IR-64	IR-64 (99,26%)	Benar
23	1035 ms	2110 lux	IR-64	IR-64 (99,42%)	Benar
24	1053 ms	2110 lux	IR-64	IR-64 (96,69%)	Benar
25	1003 ms	2110 lux	IR-64	IR-64 (51,33%)	Benar
26	1049 ms	2110 lux	IR-64	IR-64 (69,87%)	Benar
27	1032 ms	2110 lux	IR-64	IR-64 (99,52%)	Benar
28	1012 ms	2110 lux	IR-64	IR-64 (99,89%)	Benar
29	1047 ms	2110 lux	IR-64	IR-64 (99,77%)	Benar
30	1088 ms	2110 lux	IR-64	IR-64 (79,67%)	Benar
31	2798 ms	2110 lux	Ketan	IR-64 (76,91%)	Salah
32	1032 ms	2110 lux	Ketan	IR-64 (98,99%)	Salah

33	1084 ms	2110 lux	Ketan	IR-64 (87,55%)	Salah
34	1049 ms	2110 lux	Ketan	IR-64 (88,38%)	Salah
35	1246 ms	2110 lux	Ketan	Ketan (92,25%)	Benar
36	1535 ms	2110 lux	Ketan	IR-64 (78,12%)	Salah
37	1084 ms	2110 lux	Ketan	IR-64 (96,22%)	Salah
38	1114 ms	2110 lux	Ketan	IR-64 (95,13%)	Salah
39	1068 ms	2110 lux	Ketan	IR-64 (85,5%)	Salah
40	1027 ms	2110 lux	Ketan	Ketan (71,49%)	Benar
41	1072 ms	2110 lux	Ketan	IR-64 (87,36%)	Salah
42	1103 ms	2110 lux	Ketan	IR-64 (99,4%)	Salah
43	1021 ms	2110 lux	Ketan	IR-64 (93,39%)	Salah
44	1031 ms	2110 lux	Ketan	IR-64 (72,64%)	Salah
45	1036 ms	2110 lux	Ketan	Ketan (52,34%)	Benar

Dari tabel pengujian tersebut dapat dilihat bahwa hasil prediksi untuk Basmathi hanya terdapat 1 kesalahan, IR-64 benar semua dan Ketan hanya terdapat 3 yang benar. Tingkat probabilitas dari ketiganya tidak stabil yang bervariasi antara 50% sampai 90%. Hal tersebut disebabkan oleh tidak meratanya cahaya yang mengenai beras sehingga terdapat bayangan yang menjadi faktor penurunan tingkat akurasi dibanding dengan cahaya *flashlight*. Penurunan tingkat akurasi pada Beras Ketan yang cukup signifikan disebabkan karena bayangan yang dihasilkan membuat jaringan CNN memprediksi klasifikasinya ke Beras IR-64 yang memang tekstur warnanya lebih gelap daripada Beras Ketan. Respon deteksi pada pengujian tersebut tergolong cepat antara 1 sampai 2 detik dan bergantung pada proses yang sedang dilakukan oleh prosesor. Setelah diperoleh tabel hasil pengujinya maka dapat dihitung persentase akurasi dan persentase kesalahan pengujian untuk seluruh varietas berasnya sebagai berikut.

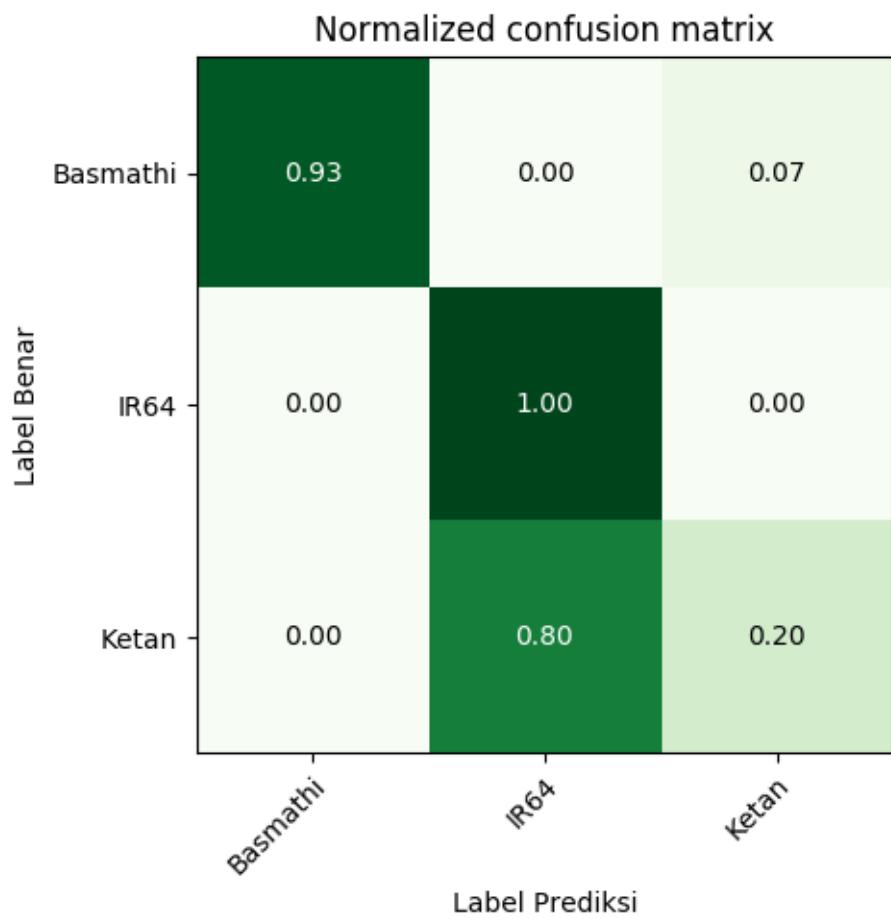
$$\% \text{ akurasi} = \frac{\text{jumlah prediksibenar}}{\text{jumlah datapengujian}} \times 100\%$$

$$\% \text{ akurasi} = \frac{32}{45} \times 100\% = 71,11\%$$

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

$$\% kesalahan = \frac{13}{45} \times 100\% = 28,89\%$$

Dari hasil perhitungan tersebut persentase akurasinya lebih rendah daripada menggunakan *flashlight* dan persentase kesalahannya juga lebih tinggi dari pengujian sebelumnya. Hal tersebut disebabkan karena pencahayaan ruangan yang tidak merata dan terdapatnya bayangan pada obyek, tidak seperti menggunakan *flashlight* yang cukup merata pencahayaannya. Untuk mengetahui tingkat akurasi prediksi pada setiap varietas beras dapat dilihat pada *confusion matrix* yang disajikan pada gambar 4.22 berikut.



Gambar 4.22 Confusion matrix pengujian arsitektur MobileNetV1 pada Android tanpa flashlight

Berdasarkan *confusion matrix* tersebut tingkat akurasi hasil prediksi yang benar Beras Basmathi mencapai 93% Beras IR-64 mencapai 100% dan Beras Ketan hanya 20% dari jumlah pengujian setiap varietas berasnya. Sehingga dapat disimpulkan bahwa hasil pengujian arsitektur *MobileNetV1* dengan cahaya ruangan pada perangkat *Android* kurang akurat untuk memprediksi hasilnya. Hal itu disebabkan tidak menggunakan *dataset* dengan gambar yang terdapat bayangan serta intensitas cahaya yang mengenai obyek beras tidak merata sehingga menghasilkan bayangan yang dapat mempengaruhi perbedaan hasil ketika diproses citranya oleh jaringan CNN tersebut.

4.3.4 Hasil Pengujian pada Arsitektur *MobileNetV1* menggunakan *Flashlight* dengan Kondisi Beras dibungkus Plastik Bening

Pada pengujian keempat ini dilakukan menggunakan model *TensorFlow Lite* arsitektur *MobileNetV1* yang masih sama pada pengujian sebelumnya. Pengujian ini dilakukan sama seperti percobaan sebelumnya bedanya hanya dengan mengganti arsitektur CNN. Berikut hasil pengujianya disajikan pada tabel 4.4.

*Tabel 4.4 Hasil pengujian arsitektur *MobileNetV1* menggunakan *flashlight* pada beras dibungkus plastik bening*

Pengujian ke-	Respon Deteksi	Intensitas Cahaya <i>Flashlight/Ruang</i>	Varietas Beras	Hasil Prediksi (Tingkat Probabilitas)	Benar/Salah
1	972 ms	1070 lux	Basmathi	Basmathi (85,34%)	Benar
2	975 ms	1070 lux	Basmathi	Basmathi (78,49%)	Benar
3	1013 ms	1070 lux	Basmathi	Basmathi (65,85%)	Benar
4	990 ms	1070 lux	Basmathi	Basmathi (52,4%)	Benar
5	970 ms	1070 lux	Basmathi	Basmathi (87,69%)	Benar
6	948 ms	1070 lux	Basmathi	Basmathi (76,94%)	Benar
7	975 ms	1070 lux	Basmathi	IR-64 (79,89%)	Salah
8	1000 ms	1070 lux	Basmathi	IR-64 (59,82%)	Salah
9	972 ms	1070 lux	Basmathi	Basmathi (67,02%)	Benar
10	968 ms	1070 lux	Basmathi	Basmathi (95,91%)	Benar
11	997 ms	1070 lux	Basmathi	Basmathi (84,43%)	Benar
12	955 ms	1070 lux	Basmathi	Basmathi (97,99%)	Benar
13	1001 ms	1070 lux	Basmathi	Basmathi (74,19%)	Benar
14	976 ms	1070 lux	Basmathi	Basmathi (50,93%)	Benar
15	959 ms	1070 lux	Basmathi	Basmathi (54,19%)	Benar
16	1315 ms	1070 lux	IR-64	IR-64 (99,88%)	Benar
17	986 ms	1070 lux	IR-64	IR-64 (99,44%)	Benar
18	1009 ms	1070 lux	IR-64	IR-64 (99,77%)	Benar
19	996 ms	1070 lux	IR-64	IR-64 (99,89%)	Benar
20	982 ms	1070 lux	IR-64	IR-64 (91,49%)	Benar
21	978 ms	1070 lux	IR-64	IR-64 (89,47%)	Benar
22	996 ms	1070 lux	IR-64	IR-64 (99,86%)	Benar
23	971 ms	1070 lux	IR-64	IR-64 (99,75%)	Benar
24	972 ms	1070 lux	IR-64	IR-64 (99,98%)	Benar
25	956 ms	1070 lux	IR-64	IR-64 (73,32%)	Benar
26	1032 ms	1070 lux	IR-64	IR-64 (99,97%)	Benar
27	975 ms	1070 lux	IR-64	IR-64 (99,99%)	Benar
28	988 ms	1070 lux	IR-64	IR-64 (95,04%)	Benar

29	971 ms	1070 lux	IR-64	IR-64 (98,88%)	Benar
30	950 ms	1070 lux	IR-64	IR-64 (99,88%)	Benar
31	1305 ms	1070 lux	Ketan	Ketan (90,4%)	Benar
32	969 ms	1070 lux	Ketan	Ketan (99,98%)	Benar
33	948 ms	1070 lux	Ketan	Ketan (97,57%)	Benar
34	990 ms	1070 lux	Ketan	Ketan (99,45%)	Benar
35	975 ms	1070 lux	Ketan	Ketan (99,96%)	Benar
36	986 ms	1070 lux	Ketan	Ketan (99,99%)	Benar
37	986 ms	1070 lux	Ketan	Ketan (99,53%)	Benar
38	917 ms	1070 lux	Ketan	Ketan (99,87%)	Benar
39	945 ms	1070 lux	Ketan	Ketan (95,17%)	Benar
40	940 ms	1070 lux	Ketan	Ketan (99,01%)	Benar
41	949 ms	1070 lux	Ketan	Ketan (97,35%)	Benar
42	985 ms	1070 lux	Ketan	Ketan (81,9%)	Benar
43	962 ms	1070 lux	Ketan	Ketan (99,8%)	Benar
44	988 ms	1070 lux	Ketan	Ketan (99,96%)	Benar
45	976 ms	1070 lux	Ketan	Ketan (94,28%)	Benar

Dari tabel pengujian tersebut dapat dilihat bahwa hasil prediksi untuk IR-64 dan Ketan benar semua, dan Basmathi hanya terdapat 2 kesalahan yang memprediksi IR-64. Tingkat probabilitas dari hasil prediksi IR-64 dan Ketan cukup stabil di angka 90% namun untuk Basmathi bervariasi dari 50% sampai 90%. Hasil tersebut sedikit berbeda dengan beras yang disebarluaskan pada arsitektur dan kondisi cahaya yang sama dikarenakan adanya sedikit pantulan cahaya dari bungkus plastik berasnya. Setelah diperoleh tabel tersebut maka dapat dihitung persentase akurasi dan persentase kesalahan pengujian untuk seluruh varietas berasnya sebagai berikut.

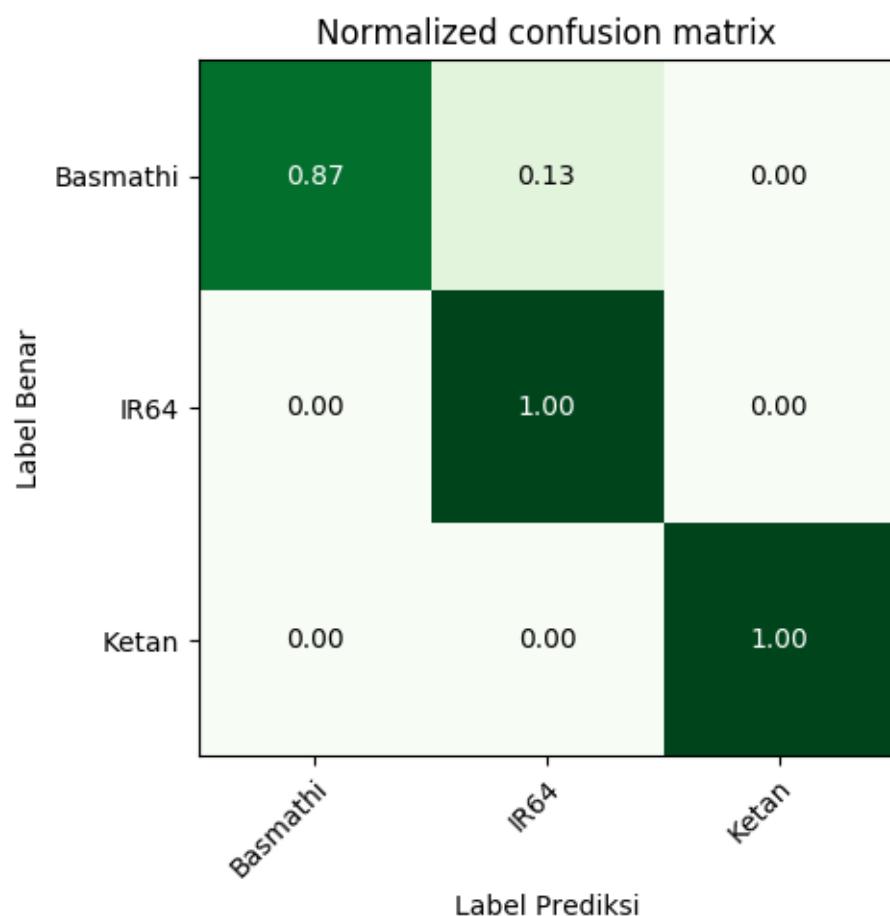
$$\% \text{akurasi} = \frac{\text{jumlahprediksibenar}}{\text{jumlahdatapengujian}} \times 100\%$$

$$\% \text{akurasi} = \frac{43}{45} \times 100\% = 95,56\%$$

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

$$\% \text{kesalahan} = \frac{2}{45} \times 100\% = 4,44\%$$

Dari hasil perhitungan tersebut diperoleh persentase akurasi dan kesalahan yang sama dengan percobaan ketiga pada kondisi beras yang disebarluaskan pada arsitektur *MobileNetV1* dengan *flashlight*. Hal tersebut menandakan bahwa jaringan CNN masih mampu melakukan klasifikasi dengan baik dengan kondisi obyek beras yang terbungkus dengan plastik bening. Untuk mengetahui tingkat akurasi prediksi pada setiap varietas beras dapat dilihat pada *confusion matrix* yang disajikan pada gambar 4.23 berikut.



Gambar 4.23 Confusion matrix pengujian arsitektur MobileNetV1 pada Android menggunakan flashlight dengan kondisi beras terbungkus

Berdasarkan *confusion matrix* tersebut tingkat akurasi hasil prediksi Basmathi yang benar mencapai 87% dengan hanya terdapat kesalahan prediksi pada IR64 13%, sedangkan IR64 dan Ketan mencapai 100%. Selain itu juga persentase hasil prediksi tiap varietas berasnya tidak berbeda jauh dengan beras yang disebarluaskan menggunakan cahaya *flashlight*. Sehingga dapat disimpulkan bahwa klasifikasi dari ketiga varietas tersebut berhasil dilakukan dengan baik karena intensitas cahaya yang merata dan pantulan cahaya dari plastik yang tidak begitu banyak.

4.3.5 Hasil Pengujian pada Arsitektur *MobileNetV1* tanpa *Flashlight* dengan Kondisi Beras dibungkus Plastik Bening

Pengujian ini dilakukan menggunakan model *TensorFlow Lite* arsitektur *MobileNetV1* yang sama, yaitu dengan *dataset* kualitas yang baik. Pengujian ini dilakukan sama seperti percobaan sebelumnya bedanya hanya pada kondisi pencahayaan. Berikut hasil pengujian disajikan pada tabel 4.8.

*Tabel 4.5 Hasil pengujian arsitektur *MobileNetV1* tanpa *flashlight* pada beras dibungkus plastik bening*

Pengujian ke-	Respon Deteksi	Intensitas Cahaya <i>Flashlight/Ruang</i>	Varietas Beras	Hasil Prediksi (Tingkat Probabilitas)	Benar/Salah
1	983 ms	2110 lux	Basmathi	Basmathi (49,96%)	Benar
2	969 ms	2110 lux	Basmathi	Basmathi (72,34%)	Benar
3	984 ms	2110 lux	Basmathi	Basmathi (97,03%)	Benar
4	1002 ms	2110 lux	Basmathi	Basmathi (79,11%)	Benar
5	966 ms	2110 lux	Basmathi	Basmathi (99,03%)	Benar
6	993 ms	2110 lux	Basmathi	Basmathi (98,87%)	Benar
7	974 ms	2110 lux	Basmathi	Basmathi (89,93%)	Benar
8	923 ms	2110 lux	Basmathi	Basmathi (86,05%)	Benar
9	994 ms	2110 lux	Basmathi	IR-64 (52,84%)	Salah
10	1010 ms	2110 lux	Basmathi	Basmathi (96,68%)	Benar
11	965 ms	2110 lux	Basmathi	Basmathi (68,25%)	Benar
12	1026 ms	2110 lux	Basmathi	Basmathi (97,4%)	Benar
13	1066 ms	2110 lux	Basmathi	Basmathi (97,5%)	Benar
14	953 ms	2110 lux	Basmathi	Basmathi (88,67%)	Benar

15	940 ms	2110 lux	Basmathi	Basmathi (50,15%)	Benar
16	1541 ms	2110 lux	IR-64	IR-64 (99,44%)	Benar
17	993 ms	2110 lux	IR-64	IR-64 (84,62%)	Benar
18	956 ms	2110 lux	IR-64	IR-64 (99,19%)	Benar
19	965 ms	2110 lux	IR-64	IR-64 (98,32%)	Benar
20	973 ms	2110 lux	IR-64	IR-64 (99,02%)	Benar
21	1109 ms	2110 lux	IR-64	IR-64 (80,65%)	Benar
22	956 ms	2110 lux	IR-64	IR-64 (99,86%)	Benar
23	1024 ms	2110 lux	IR-64	IR-64 (88,59%)	Benar
24	968 ms	2110 lux	IR-64	IR-64 (98,02%)	Benar
25	1038 ms	2110 lux	IR-64	IR-64 (99,4%)	Benar
26	930 ms	2110 lux	IR-64	IR-64 (96,26%)	Benar
27	993 ms	2110 lux	IR-64	IR-64 (99,77%)	Benar
28	1026 ms	2110 lux	IR-64	IR-64 (95,64%)	Benar
29	994 ms	2110 lux	IR-64	IR-64 (85,93%)	Benar
30	977 ms	2110 lux	IR-64	IR-64 (99,47%)	Benar
31	1729 ms	2110 lux	Ketan	Ketan (97,62%)	Benar
32	1012 ms	2110 lux	Ketan	Ketan (92,49%)	Benar
33	1052 ms	2110 lux	Ketan	Ketan (98,79%)	Benar
34	1058 ms	2110 lux	Ketan	Ketan (98,57%)	Benar
35	1038 ms	2110 lux	Ketan	Ketan (97,44%)	Benar
36	1026 ms	2110 lux	Ketan	Ketan (99,83%)	Benar
37	1007 ms	2110 lux	Ketan	Ketan (99,7%)	Benar
38	1003 ms	2110 lux	Ketan	Ketan (99,97%)	Benar
39	1011 ms	2110 lux	Ketan	Ketan (98,79%)	Benar
40	1043 ms	2110 lux	Ketan	Ketan (99,53%)	Benar
41	1047 ms	2110 lux	Ketan	Ketan (98,34%)	Benar
42	1041 ms	2110 lux	Ketan	Ketan (99,99%)	Benar
43	1014 ms	2110 lux	Ketan	Ketan (89,78%)	Benar
44	1022 ms	2110 lux	Ketan	Ketan (99,87%)	Benar
45	1002 ms	2110 lux	Ketan	Ketan (99,81%)	Benar

Dari tabel pengujian tersebut dapat dilihat bahwa hasil prediksi untuk IR-64 dan Ketan benar semua, dan Basmathi hanya terdapat 1 kesalahan yang memprediksi IR-64. Tingkat probabilitas dari hasil prediksi IR-64 dan Ketan cukup stabil di angka 90% namun untuk Basmathi bervariasi dari 50% sampai 90%. Setelah diperoleh tabel tersebut maka dapat dihitung persentase akurasi dan persentase kesalahan pengujian untuk seluruh varietas berasnya sebagai berikut.

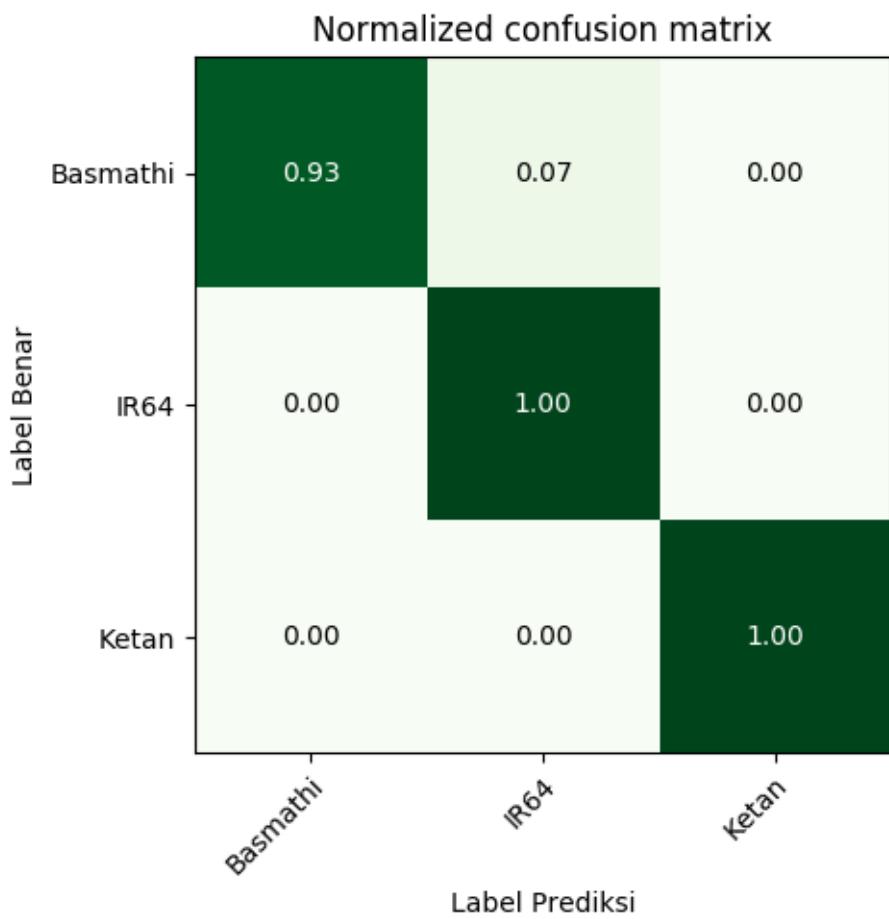
$$\% \text{akurasi} = \frac{\text{jumlahprediksibenar}}{\text{jumlahdatapengujian}} \times 100\%$$

$$\% \text{akurasi} = \frac{44}{45} \times 100\% = 97,78\%$$

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

$$\% \text{kesalahan} = \frac{1}{45} \times 100\% = 2,22\%$$

Dari hasil perhitungan tersebut diperoleh persentase akurasi dan kesalahan yang sedikit lebih baik dengan percobaan kedua pada kondisi beras yang disebarluaskan pada arsitektur *MobileNetV1* dengan *flashlight*. Hal tersebut menandakan bahwa jaringan CNN masih mampu melakukan klasifikasi dengan baik dengan kondisi obyek beras yang terbungkus dengan plastik bening pada kondisi cahaya ruangan. Selain itu kondisi sebaran pada beras yang dibungkus plastik lebih merata dibandingkan dengan beras yang disebar di alas. Untuk mengetahui tingkat akurasi prediksi pada setiap varietas beras dapat dilihat pada *confusion matrix* yang disajikan pada gambar 4.24 berikut.



Gambar 4.24 Confusion matrix pengujian arsitektur MobileNetV1 pada Android tanpa flashlight dengan kondisi beras terbungkus

Berdasarkan *confusion matrix* tersebut tingkat akurasi hasil prediksi Basmathi yang benar mencapai 93% dengan hanya terdapat kesalahan prediksi pada IR64 7%, sedangkan IR64 dan Ketan mencapai 100%. Hasil tersebut sedikit lebih baik pada prediksi varietas Beras Basmathi dan jauh lebih baik daripada pengujian dengan menyebarluaskan beras dengan kondisi cahaya ruang. Penyebabnya adalah penyebarluasan beras yang dibungkus cenderung lebih merata dan luas sehingga cahaya bayangan yang dihasilkan sedikit, selain itu juga adanya kemungkinan jarak antara kamera dengan objek beras yang relatif tepat sesuai *dataset* pelatihan yang dibuat.

BAB 5

PENUTUP

5.1 Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan maka diperoleh kesimpulannya sebagai berikut.

1. Perancangan arsitektur CNN dari *VGG-16Net* dan *MobileNetV1* untuk klasifikasi varietas beras dilakukan dengan metode *Feature Extraction* karena metode tersebut cocok digunakan pada jumlah *dataset* yang sedikit dengan hasil yang cukup baik.
2. Proses pelatihan dan pengujian klasifikasi varietas beras dengan CNN dilakukan pada infrastruktur *Google Colaboratory* agar menghemat waktu pelatihan serta mudah untuk mengimpor *framework Keras* dan *TensorFlow* sehingga tidak perlu memasangnya secara lokal.
3. Pengujian klasifikasi varietas beras dengan perangkat *Android* dilakukan dengan cara mengambil obyek beras yang akan dideteksi menggunakan fitur kamera dengan keadaan beras yang disebarluaskan di alas berwarna hitam dan dibungkus plastik bening.
4. Arsitektur *VGG-16Net* jauh lebih akurat dan lebih konsisten hasilnya dibandingkan dengan arsitektur *MobileNetV1* untuk membedakan obyek yang sangat kecil dan mirip.
5. Faktor yang mempengaruhi tingkat akurasi dan tingkat kesalahan dari hasil pengujian pada CNN adalah jarak obyek yang akan diklasifikasi, kualitas gambar obyek serta kondisi pencahayaan.

6. CNN dapat melakukan klasifikasi obyek gambar dengan baik jika obyek yang akan diklasifikasi sangat mirip dengan *dataset* yang sudah dilatih.
7. Tingkat akurasi validasi hasil pelatihan arsitektur *VGG-16Net* lebih tinggi dibandingkan arsitektur *MobileNetV1*, yaitu sebesar 1.0 baik pada *dataset* kualitas baik maupun yang buruk sedangkan pada *MobileNetV1* sebesar 0.9333 pada *dataset* kualitas baik dan 0.6889 pada *dataset* kualitas buruk dalam rentang nilai dari 0 hingga 1.
8. Tingkat kesalahan validasi hasil pelatihan arsitektur *VGG-16Net* lebih rendah daripada arsitektur *MobileNetV1*, dengan nilai 0.058 pada *dataset* kualitas baik dan 0.1037 pada *dataset* kualitas buruk sedangkan pada *MobileNetV1* sebesar 0.1722 untuk *dataset* kualitas yang baik dan 0.9485 pada *dataset* kualitas yang buruk.
9. Hasil pengujian arsitektur *VGG-16Net* pada perangkat *Android* menghasilkan nilai akurasi sebesar 75,56%, lebih rendah dari pada arsitektur *MobileNetV1* yang mencapai 95,56% karena model hasil pelatihan arsitektur *VGG-16Net* tidak dapat dikonversi ke *TensorFlow Lite* dengan baik sehingga hasil klasifikasi pada aplikasi *Android* menjadi tidak akurat.
10. Pengujian arsitektur *MobileNetV1* pada beras yang disebar menggunakan *flashlight* pada *Android* tingkat akurasinya mencapai 95,56%, jauh lebih baik dibandingkan pada kondisi intensitas cahaya ruangan sebesar 2110 *lux* dengan tingkat akurasi hanya sebesar 71,11% karena disebabkan oleh cahaya yang mengenai obyek beras yang disebar tidak merata.

11. Tingkat akurasi hasil pengujian arsitektur *MobileNetV1* tanpa menggunakan *flashlight* pada Android dengan keadaan beras dibungkus plastik bening sebesar 97,78% jauh lebih baik dibandingkan dengan keadaan beras disebarlu yang hanya mencapai 71,11% karena cahaya yang mengenai beras yang dibungkus lebih merata dibandingkan dengan beras yang disebarlu.
12. Waktu respon deteksi arsitektur *MobileNetV1* jauh lebih cepat yaitu sekitar 1000 ms daripada arsitektur *VGG-16Net* yang sekitar 5000 ms di perangkat *Android* karena adanya *depthwise separable convolution*.

5.2 Saran

Saran yang perlu dikembangkan untuk tugas akhir ini agar lebih baik sebagai berikut.

1. *Dataset* pelatihan ditambahkan lagi jumlah dan variasi kondisi berasnya agar hasil prediksinya semakin meningkat.
2. *Dataset* pengujian perlu ditambahkan lagi agar tingkat akurasi hasil pengujian tiap varietas berasnya tidak terlalu berbeda jauh.
3. Perlu adanya penambahan varietas beras yang lebih beragam untuk dilakukan klasifikasi menggunakan CNN.
4. Proses pendekripsi varietas beras pada aplikasi *Android* dikembangkan lagi agar dapat mendekripsi secara *realtime* sehingga pengguna tidak perlu menekan tombol “Deteksi” pada aplikasi tersebut.

DAFTAR PUSTAKA

- [1] Webstaurant Store. “*Types of Rice*”. [Daring]. Tersedia pada: <https://www.webstaurantstore.com/guide/658/types-of-rice.html>. [Diakses: 23-Okt-2019].
- [2] Chollet, François. 2018. *Deep Learning with Python*. Shelter Island: Manning Publications Co.
- [3] Sarirotul Ilahiyah dan Agung Nilogiri. 2018. Implementasi *Deep Learning* Pada Identifikasi Jenis Tumbuhan Berdasarkan Citra Daun Menggunakan *Convolutional Neural Network*. JUSTINDO (Jurnal Sistem & Teknologi Informasi Indonesia) Vol. 3, No. 2, (2018) : Agustus 2018.
- [4] I Wayan Suartika E. P, dkk. 2016. Klasifikasi Citra Menggunakan *Convolutional Neural Network* (CNN) pada Caltech 101. Jurnal Teknik ITS Vol. 5, No. 1 : 2016.
- [5] Stathakis, D. 2009. *How many hidden layers and nodes?*. International Journal of Remote Sensing Vol. 30, No. 8, (2009) : April 2009.
- [6] Tandungan, Sofyan. 2019. “Pengenalan *Convolutional Neural Network – Part 1*”. [Daring]. Tersedia pada: <http://sofyantandungan.com/pengenalan-convolutional-neural-network-part-1/>. [Diakses: 24-Okt-2019].
- [7] Anonim. “*CS231n Convolutional Neural Networks for Visual Recognition*”. [Daring]. Tersedia pada: <http://cs231n.github.io/convolutional-networks/>. [Diakses: 12-Nov-2019].
- [8] 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: 23-Okt-2019].
- [9] William Sugiarto, dkk. 2017. Estimasi Arah Tatapan Mata dengan Menggunakan *Average Pooling Convolutional Neural Network*. Dinamika Teknologi Vol. 9, No. 2 : 2017.
- [10] Neurohive. “*VGG16 – Convolutional Network for Classification and Detection*”. [Daring]. Tersedia pada: <https://neurohive.io/en/popular-networks/vgg16/>. [Diakses: 28-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: 28-Nov-2019].
- [12] Google Inc. 2017. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. arXiv:1704.04861v1 [cs.CV] 17 Apr 2017.
- [13] John Paul Mueller dan Luca Massaron. “*What is Google Colaboratory?*”. [Daring]. Tersedia pada: <https://www.dummies.com/programming/python/what-is-google-colaboratory/>. [Diakses: 24-Okt-2019].
- [14] TensorFlow. “*Using TensorFlow Lite on Android*”. [Daring]. Tersedia pada: <https://medium.com/tensorflow/using-tensorflow-lite-on-android-9bbc9cb7d69d>. [Diakses: 16-Nov-2019].
- [15] Google. “*Android Studio User Guide*”. [Daring]. Tersedia pada: <https://developer.android.com/studio/intro>. [Diakses: 12-Nov-2019].

- [16]John Paul Mueller dan Luca Massaron. “*Android Studio*: Panduan Untuk Pemula”. [Daring]. Tersedia pada: <https://www.dewaweb.com/blog/android-studio/>. [Diakses: 12-Nov-2019].
- [17]Aji, J.M.M dan Widodo, A. 2010. Perilaku Konsumen pada Pembelian Beras di Kabupaten Jember dan Faktor yang Memenaruhinya. *Jurnal Sosial Ekonomi Pertanian* Vol. 1, No. 3 : 2016.
- [18]Suprihatno, B, Daradjat, dkk. 2010. Deskripsi Varietas Padi. Balai Besar Penelitian Tanaman Padi.
- [19]As Shidiq Cater Indonesia. “Kandungan-kandungan beras basmati”. [Daring]. <https://www.asshidiqaqiqah.com/kandungan-kandungan-beras-basmati/>. [Diakses 27-Nov-2019].
- [20]Priyanto, T. 2012. “Beras Ketan & Sifat Fisika-Kimianya”. [Daring]. <http://www.alatcetakrengginang.com/2012/02/beras-ketan-sifat-fisika-kimianya.html>. [Diakses: 24-Okt-2019].

LAMPIRAN

Lampiran 1. Kode Sumber Klasifikasi Varietas Beras VGG-16Net.ipynb

1. Menggunakan *Dataset* Kualitas Baik

```
Mengambil Dataset dari Github
!apt-get install subversion > /dev/null

!svn export https://github.com/Soedirman-Machine-
Learning/rice-varieties-classification/trunk/data > /dev/null

Memasukan Fungsi Library
from __future__ import absolute_import, division,
print_function, unicode_literals

try:
    # The %tensorflow_version magic only works in colab.
    %tensorflow_version 2.x
except Exception:
    pass
import numpy as np
import math, os, sys
import itertools

import matplotlib.pyplot as plt
plt.style.use('default')
from scipy import ndimage

from skimage import measure, morphology
from skimage.io import imsave, imread
from skimage.filters import threshold_otsu
from skimage.transform import resize

import tensorflow as tf
from sklearn import svm, datasets
from sklearn.metrics import confusion_matrix
import pandas as pd

Mengimpor Dataset
#Perintah di bawah ini untuk melihat list bagian dari file
yang sudah diunduh dari Github
!ls data/image
!ls data/image/train
!ls data/image/train/Basmathi
!ls data/image/train/IR64
!ls data/image/train/Ketan

#Perintah untuk memvisualisasikan satu gambar dari data train
```

```

image = imread("data/image/train/IR64/I17.jpg")
plt.figure(figsize=(3,3))
plt.imshow(image)
#Memuat semua gambar ke memori untuk pertama kali

#Memuat dataset pelatihan
IMAGE_SIZE = 224
BATCH_SIZE = 128
base_dir = os.path.join('data/image/train')

datagen = tf.keras.preprocessing.image.ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2)

train_generator = datagen.flow_from_directory(
    base_dir,
    target_size=(IMAGE_SIZE, IMAGE_SIZE),
    batch_size=BATCH_SIZE,
    subset='training')

val_generator = datagen.flow_from_directory(
    base_dir,
    target_size=(IMAGE_SIZE, IMAGE_SIZE),
    batch_size=BATCH_SIZE,
    subset='validation')

#Memuat dataset pengujian
X_test = []
y_test = []
labels = ['Basmathi', 'IR64', 'Ketan']

for i,label in enumerate(labels):
    folder = os.path.join("data/image/test",label)
    files = sorted(os.listdir(folder))
    files = [x for x in files if x.endswith(".jpg")]
    for k,file in enumerate(files):
        image_path = os.path.join(folder, file)

        image = imread(image_path)/255.
        image = resize(image, (224,224))
        X_test.append(image)
        category = os.path.split(folder)[-1]
        y_test.append(i)

X_test = np.array(X_test)
y_test = np.array(y_test)

#Menampilkan bentuk dari masing-masing dataset
for image_batch, label_batch in train_generator:
    break
print("Bentuk array dari dataset train (pelatihan) adalah:",
      image_batch.shape,label_batch.shape)
for image_batch, label_batch in val_generator:
    break

```

```

print("Bentuk array dari dataset validation (validasi) adalah:", image_batch.shape,label_batch.shape)
print("Bentuk array dari dataset test (pengujian) adalah:", X_test.shape,y_test.shape)

Menyimpan Label
print (train_generator.class_indices)

labels_txt =
'\n'.join(sorted(train_generator.class_indices.keys()))

with open('labels.txt', 'w') as f:
    f.write(labels_txt)

!cat labels.txt

Membuat Model dari Jaringan CNN yang Sudah dipelajari Sebelumnya (pre-trained convnets)
IMG_SHAPE = (224, 224, 3)
# Membuat model dasar (base model) dari pre-trained model VGG-16Net
base_model =
tf.keras.applications.VGG16(input_shape=IMG_SHAPE,
                               include_top=False,
                               weights='imagenet')

Feature Extraction
base_model.trainable = False
base_model.summary()

Mengelompokkan Klasifikasi Tiga Varietas Beras dengan Deep learning (Convolutional Neural Network)
import keras
from keras import backend as K
from keras.models import Sequential
from keras import layers
from keras.utils.np_utils import to_categorical

from sklearn.model_selection import train_test_split

Mempersiapkan Data Masukkan Pengujian
y_test2 = to_categorical(y_test)
X_test3, y_test3 = (X_test, y_test2)

Pembuatan Model (Menambah Model)
base_model,
tf.keras.layers.Conv2D(32, 3, activation='relu'),
#tf.keras.layers.Dropout(0.2),
tf.keras.layers.GlobalAveragePooling2D(),
tf.keras.layers.Dense(3, activation='softmax')
])

```

```

model.compile("adam", loss="categorical_crossentropy", metrics=[  

    "acc"])
model.summary()

Pelatihan  

#from keras.callbacks import EarlyStopping, ModelCheckpoint  
  

#Menyimpan file model bobot yang terbaik selama pelatihan  

(dalam format keras ".h5")  

#ckpt = ModelCheckpoint("Klasifikasi_Beras_Tumpukan.h5",  

monitor='val_loss', verbose=1, save_best_only=True,  

save_weights_only=False, mode='auto', period=1)  

history = model.fit_generator(train_generator,  

                                epochs=100,  

                                validation_data=val_generator)  

#history = model.fit(x = X_train3, y = y_train3,  

batch_size=120, epochs=  

100,validation_data=(X_valid3,y_valid3),callbacks = [ckpt])  
  

Menggambarkan Hasil Pelatihan  

plt.plot(history.history["acc"],label="Akurasi Pelatihan")  

plt.plot(history.history["val_acc"],label="Validasi Akurasi")  

plt.legend()  

plt.show()  
  

plt.plot(history.history["loss"],label="Kesalahan Pelatihan")  

plt.plot(history.history["val_loss"],label="Validasi  
Kesalahan")  

plt.legend()  

plt.show()  
  

print('Number of trainable variables =  

{}'.format(len(model.trainable_variables)))  
  

Menggunakan Model  

print(train_generator)  
  

#Prediksi Label Validasi dengan Pelatihan  

n = 44  

input_image = image_batch[n][np.newaxis,...]  

print("Labelnya adalah: ", label_batch[n])  
  

predictions = model.predict(input_image)
print("Prediksinya adalah", predictions[0])  
  

Evaluasi  

#Memeriksa matriks model
print(model.metrics_names)
#Evaluasi data test
print(model.evaluate(x= X_test3, y = y_test3))  
  

#Menampilkan matriks yang benar dan matriks hasil prediksi

```

```

#Label yang benar
y_true = np.argmax(y_test2, axis=1)
#Label prediksi
Y_pred = model.predict(X_test)
y_pred = np.argmax(Y_pred, axis=1)

print(y_true)
print(y_pred)

Prediksi Gambar Individual
n = 44 #Jangan melampaui (nilai dari gambar test - 1)

plt.imshow(X_test[n])
plt.show()

true_label = np.argmax(y_test2, axis=1)[n]
print("Label yang benar
adalah:",true_label,":",labels[true_label])
prediction = model.predict(X_test[n][np.newaxis,...])[0]
print("Nilai yang diprediksi adalah:",prediction)
predicted_label = np.argmax(prediction)
print("Label yang diprediksi
adalah:",predicted_label,":",labels[predicted_label])

if true_label == predicted_label:
    print("Prediksi benar")
else:
    print("Prediksi salah")

Confusion Matrix
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=(5,5))
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='Label Benar',
       xlabel='Label Prediksi')

# 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=labels,
normalize=True,
                    title='Normalized confusion matrix')

Menyimpan dan Konversi ke ".tflite"
saved_model_dir = 'save/model'
tf.saved_model.save(model, saved_model_dir)

converter =
tf.lite.TFLiteConverter.from_saved_model(saved_model_dir)
tflite_model = converter.convert()

with open('Klasifikasi_3_Varietas_Beras_VGG16Net.tflite',
'wb') as f:
    f.write(tflite_model)

```

2. Menggunakan Dataset Kualitas Rendah (tanpa *flashlight*)

Mengambil Dataset dari Github

```
!apt-get install subversion > /dev/null
```

```
!svn export https://github.com/Soedirman-Machine-
Learning/rice-varieties-classification/trunk/data_nf >
/dev/null
```

Memasukan Fungsi Library

```
from __future__ import absolute_import, division,
print_function, unicode_literals
```

try:

```
# The %tensorflow_version magic only works in colab.
%tensorflow_version 2.x
```

except Exception:

```
pass
```

```
import numpy as np
import math, os, sys
import itertools
```

```
import matplotlib.pyplot as plt
```

```
plt.style.use('default')
```

```
from scipy import ndimage
```

```
from skimage import measure, morphology
```

```
from skimage.io import imsave, imread
```

```
from skimage.filters import threshold_otsu
```

```
from skimage.transform import resize
```

```
import tensorflow as tf
```

```
from sklearn import svm, datasets
```

```
from sklearn.metrics import confusion_matrix
```

```
import pandas as pd
```

Mengimpor Dataset

```
#Perintah di bawah ini untuk melihat list bagian dari file
yang sudah diunduh dari Github
```

```
!ls data_nf/image
```

```
!ls data_nf/image/train
```

```
!ls data_nf/image/train/Basmathi
```

```
!ls data_nf/image/train/IR64
```

```
!ls data_nf/image/train/Ketan
```

```
#Perintah untuk memvisualisasikan satu gambar dari data train
```

```
image = imread("data_nf/image/train/IR64/I_nf017.jpg")
```

```
plt.figure(figsize=(3,3))
```

```
plt.imshow(image)
```

```
#Memuat semua gambar ke memori untuk pertama kali
```

```

#Memuat dataset pelatihan
IMAGE_SIZE = 224
BATCH_SIZE = 128
base_dir = os.path.join('data_nf/image/train')

datagen = tf.keras.preprocessing.image.ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2)

train_generator = datagen.flow_from_directory(
    base_dir,
    target_size=(IMAGE_SIZE, IMAGE_SIZE),
    batch_size=BATCH_SIZE,
    subset='training')

val_generator = datagen.flow_from_directory(
    base_dir,
    target_size=(IMAGE_SIZE, IMAGE_SIZE),
    batch_size=BATCH_SIZE,
    subset='validation')

#Memuat dataset pengujian
X_test = []
y_test = []
labels = ['Basmathi', 'IR64', 'Ketan']

for i,label in enumerate(labels):
    folder = os.path.join("data_nf/image/test",label)
    files = sorted(os.listdir(folder))
    files = [x for x in files if x.endswith(".jpg")]
    for k,file in enumerate(files):
        image_path = os.path.join(folder, file)

        image = imread(image_path)/255.
        image = resize(image, (224,224))
        X_test.append(image)
        category = os.path.split(folder)[-1]
        y_test.append(i)

X_test = np.array(X_test)
y_test = np.array(y_test)

#Menampilkan bentuk dari masing-masing dataset
for image_batch, label_batch in train_generator:
    break
print("Bentuk array dari dataset train (pelatihan) adalah:", 
image_batch.shape,label_batch.shape)
for image_batch, label_batch in val_generator:
    break
print("Bentuk array dari dataset validation (validasi) adalah:", 
image_batch.shape,label_batch.shape)
print("Bentuk array dari dataset test (pengujian) adalah:", 
X_test.shape,y_test.shape)

```

```

Menyimpan Label
print (train_generator.class_indices)

labels_txt =
'\n'.join(sorted(train_generator.class_indices.keys()))

with open('labels.txt', 'w') as f:
    f.write(labels_txt)

!cat labels.txt

Membuat Model dari Jaringan CNN yang Sudah dipelajari Sebelumnya (pre-trained convnets)
IMG_SHAPE = (224, 224, 3)
# Membuat model dasar (base model) dari pre-trained model VGG-16Net
base_model =
tf.keras.applications.VGG16(input_shape=IMG_SHAPE,
include_top=False,
weights='imagenet')

Feature Extraction
base_model.trainable = False
base_model.summary()

Mengelompokkan Klasifikasi Tiga Varietas Beras dengan Deep learning (Convolutional Neural Network)
import keras
from keras import backend as K
from keras.models import Sequential
from keras import layers
from keras.utils.np_utils import to_categorical

from sklearn.model_selection import train_test_split

Mempersiapkan Data Masukkan Pengujian
y_test2 = to_categorical(y_test)
X_test3, y_test3 = (X_test, y_test2)

Pembuatan Model (Menambah Model)
base_model,
tf.keras.layers.Conv2D(32, 3, activation='relu'),
#tf.keras.layers.Dropout(0.2),
tf.keras.layers.GlobalAveragePooling2D(),
tf.keras.layers.Dense(3, activation='softmax')
[])

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

```

Pelatihan

```
#from keras.callbacks import EarlyStopping, ModelCheckpoint

#Menyimpan file model bobot yang terbaik selama pelatihan
(dalam format keras ".h5")
#ckpt = ModelCheckpoint("Klasifikasi_Beras_Tumpukan.h5",
monitor='val_loss', verbose=1, save_best_only=True,
save_weights_only=False, mode='auto', period=1)
history = model.fit_generator(train_generator,
epochs=100,
validation_data=val_generator)
#history = model.fit(x = X_train3, y = y_train3,
batch_size=120, epochs=
100, validation_data=(X_valid3,y_valid3), callbacks = [ckpt])
```

Menggambarkan Hasil Pelatihan

```
plt.plot(history.history["acc"],label="Akurasi Pelatihan")
plt.plot(history.history["val_acc"],label="Validasi Akurasi")
plt.legend()
plt.show()

plt.plot(history.history["loss"],label="Kesalahan Pelatihan")
plt.plot(history.history["val_loss"],label="Validasi
Kesalahan")
plt.legend()
plt.show()

print('Number of trainable variables =
{}'.format(len(model.trainable_variables)))
```

Menggunakan Model

```
print(train_generator)

#Prediksi Label Validasi dengan Pelatihan
n = 44
input_image = image_batch[n][np.newaxis,...]
print("Labelnya adalah: ", label_batch[n])

predictions = model.predict(input_image)
print("Prediksinya adalah",predictions[0])
```

Evaluasi

```
#Memeriksa matriks model
print(model.metrics_names)
#Evaluasi data test
print(model.evaluate(x= X_test3, y = y_test3))

#Menampilkan matriks yang benar dan matriks hasil prediksi
#Label yang benar
y_true = np.argmax(y_test2, axis=1)
#Label prediksi
Y_pred = model.predict(X_test)
y_pred = np.argmax(Y_pred, axis=1)
```

```

print(y_true)
print(y_pred)

Prediksi Gambar Individual
n = 44 #Jangan melampaui (nilai dari gambar test - 1)

plt.imshow(X_test[n])
plt.show()

true_label = np.argmax(y_test2, axis=1)[n]
print("Label yang benar
adalah:", true_label, ":", labels[true_label])
prediction = model.predict(X_test[n][np.newaxis, ...])[0]
print("Nilai yang diprediksi adalah:", prediction)
predicted_label = np.argmax(prediction)
print("Label yang diprediksi
adalah:", predicted_label, ":", labels[predicted_label])

if true_label == predicted_label:
    print("Prediksi benar")
else:
    print("Prediksi salah")

Confusion Matrix
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=(5,5))
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='Label Benar',
       xlabel='Label Prediksi')

# 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=labels,
normalize=True,
           title='Normalized confusion matrix')

Menyimpan dan Konversi ke ".tflite"
saved_model_dir = 'save/model'
tf.saved_model.save(model, saved_model_dir)

converter =
tf.lite.TFLiteConverter.from_saved_model(saved_model_dir)
tflite_model = converter.convert()

with open('Klasifikasi_3_Varietas_Beras_VGG16Net_nf.tflite',
'wb') as f:
    f.write(tflite_model)

```

Lampiran 2. Kode Sumber Klasifikasi Varietas Beras MobileNet.ipynb

1. Menggunakan *Dataset* Kualitas Baik

```
Mengambil Dataset dari Github
!apt-get install subversion > /dev/null

!svn export https://github.com/Soedirman-Machine-
Learning/rice-varieties-classification/trunk/data > /dev/null

Memasukan Fungsi Library
from __future__ import absolute_import, division,
print_function, unicode_literals

try:
    # The %tensorflow_version magic only works in colab.
    %tensorflow_version 2.x
except Exception:
    pass
import numpy as np
import math, os, sys
import itertools

import matplotlib.pyplot as plt
plt.style.use('default')
from scipy import ndimage

from skimage import measure, morphology
from skimage.io import imsave, imread
from skimage.filters import threshold_otsu
from skimage.transform import resize

import tensorflow as tf
from sklearn import svm, datasets
from sklearn.metrics import confusion_matrix
import pandas as pd

Mengimpor Dataset
#Perintah di bawah ini untuk melihat list bagian dari file
yang sudah diunduh dari Github
!ls data/image
!ls data/image/train
!ls data/image/train/Basmathi
!ls data/image/train/IR64
!ls data/image/train/Ketan

#Perintah untuk memvisualisasikan satu gambar dari data train
image = imread("data/image/train/IR64/I17.jpg")
plt.figure(figsize=(3,3))
```

```

plt.imshow(image)
#Memuat semua gambar ke memori untuk pertama kali

#Memuat dataset pelatihan
IMAGE_SIZE = 224
BATCH_SIZE = 128
base_dir = os.path.join('data/image/train')

datagen = tf.keras.preprocessing.image.ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2)

train_generator = datagen.flow_from_directory(
    base_dir,
    target_size=(IMAGE_SIZE, IMAGE_SIZE),
    batch_size=BATCH_SIZE,
    subset='training')

val_generator = datagen.flow_from_directory(
    base_dir,
    target_size=(IMAGE_SIZE, IMAGE_SIZE),
    batch_size=BATCH_SIZE,
    subset='validation')

#Memuat dataset pengujian
X_test = []
y_test = []
labels = ['Basmathi', 'IR64', 'Ketan']

for i,label in enumerate(labels):
    folder = os.path.join("data/image/test",label)
    files = sorted(os.listdir(folder))
    files = [x for x in files if x.endswith(".jpg")]
    for k,file in enumerate(files):
        image_path = os.path.join(folder, file)

        image = imread(image_path)/255.
        image = resize(image,(224,224))
        X_test.append(image)
        category = os.path.split(folder)[-1]
        y_test.append(i)

X_test = np.array(X_test)
y_test = np.array(y_test)

#Menampilkan bentuk dari masing-masing dataset
for image_batch, label_batch in train_generator:
    break
print("Bentuk array dari dataset train (pelatihan) adalah:",image_batch.shape,label_batch.shape)
for image_batch, label_batch in val_generator:
    break
print("Bentuk array dari dataset validation (validasi) adalah:", image_batch.shape,label_batch.shape)

```

```

print("Bentuk array dari dataset test (pengujian) adalah:",
X_test.shape,y_test.shape)

Menyimpan Label
print (train_generator.class_indices)

labels_txt =
'\n'.join(sorted(train_generator.class_indices.keys()))

with open('labels.txt', 'w') as f:
    f.write(labels_txt)

!cat labels.txt

Membuat Model dari Jaringan CNN yang Sudah dipelajari Sebelumnya (pre-trained convnets)
IMG_SHAPE = (224, 224, 3)
# Membuat model dasar (base model) dari pre-trained model
MobileNetV1
base_model =
tf.keras.applications.MobileNet(input_shape=IMG_SHAPE,
include_top=False,
weights='imagenet')

Feature Extraction
base_model.trainable = False
base_model.summary()

Mengelompokkan Klasifikasi Tiga Varietas Beras dengan Deep learning (Convolutional Neural Network)
import keras
from keras import backend as K
from keras.models import Sequential
from keras import layers
from keras.utils.np_utils import to_categorical

from sklearn.model_selection import train_test_split

Mempersiapkan Data Masukkan Pengujian
y_test2 = to_categorical(y_test)
X_test3, y_test3 = (X_test, y_test2)

Pembuatan Model (Menambah Model)
base_model,
tf.keras.layers.Conv2D(32, 3, activation='relu'),
#tf.keras.layers.Dropout(0.2),
tf.keras.layers.GlobalAveragePooling2D(),
tf.keras.layers.Dense(3, activation='softmax')
[])

```

```

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

Pelatihan
#from keras.callbacks import EarlyStopping, ModelCheckpoint

#Menyimpan file model bobot yang terbaik selama pelatihan
(dalam format keras ".h5")
#ckpt = ModelCheckpoint("Klasifikasi_Beras_Tumpukan.h5",
monitor='val_loss', verbose=1, save_best_only=True,
save_weights_only=False, mode='auto', period=1)
history = model.fit_generator(train_generator,
                               epochs=100,
                               validation_data=val_generator)
#history = model.fit(x = X_train3, y = y_train3,
batch_size=120, epochs=
100,validation_data=(X_valid3,y_valid3),callbacks = [ckpt])

Menggambarkan Hasil Pelatihan
plt.plot(history.history["acc"],label="Akurasi Pelatihan")
plt.plot(history.history["val_acc"],label="Validasi Akurasi")
plt.legend()
plt.show()

plt.plot(history.history["loss"],label="Kesalahan Pelatihan")
plt.plot(history.history["val_loss"],label="Validasi
Kesalahan")
plt.legend()
plt.show()

print('Number of trainable variables =
{}'.format(len(model.trainable_variables)))

Menggunakan Model
print(train_generator)

#Prediksi Label Validasi dengan Pelatihan
n = 44
input_image = image_batch[n][np.newaxis,...]
print("Labelnya adalah: ", label_batch[n])

predictions = model.predict(input_image)
print("Prediksinya adalah", predictions[0])

Evaluasi
#Memeriksa matriks model
print(model.metrics_names)
#Evaluasi data test
print(model.evaluate(x= X_test3, y = y_test3))

#Menampilkan matriks yang benar dan matriks hasil prediksi
#Label yang benar

```

```

y_true = np.argmax(y_test2, axis=1)
#Label prediksi
Y_pred = model.predict(X_test)
y_pred = np.argmax(Y_pred, axis=1)

print(y_true)
print(y_pred)

Prediksi Gambar Individual
n = 44 #Jangan melampaui (nilai dari gambar test - 1)

plt.imshow(X_test[n])
plt.show()

true_label = np.argmax(y_test2, axis=1)[n]
print("Label yang benar
adalah:", true_label, ":", labels[true_label])
prediction = model.predict(X_test[n][np.newaxis, ...])[0]
print("Nilai yang diprediksi adalah:", prediction)
predicted_label = np.argmax(prediction)
print("Label yang diprediksi
adalah:", predicted_label, ":", labels[predicted_label])

if true_label == predicted_label:
    print("Prediksi benar")
else:
    print("Prediksi salah")

Confusion Matrix
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=(5,5))
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='Label Benar',
       xlabel='Label Prediksi')

# 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=labels,
normalize=True,
                    title='Normalized confusion matrix')

Menyimpan dan Konversi ke ".tflite"
saved_model_dir = 'save/model'
tf.saved_model.save(model, saved_model_dir)

converter =
tf.lite.TFLiteConverter.from_saved_model(saved_model_dir)
tflite_model = converter.convert()

with open('Klasifikasi_3_Varietas_Beras_MobileNet.tflite',
'wb') as f:
    f.write(tflite_model)

```

2. Menggunakan Dataset Kualitas Rendah (tanpa *flashlight*)

```
Mengambil Dataset dari Github
!apt-get install subversion > /dev/null

!svn export https://github.com/Soedirman-Machine-Learning/rice-varieties-classification/trunk/data_nf > /dev/null

Memasukan Fungsi Library
from __future__ import absolute_import, division,
print_function, unicode_literals

try:
    # The %tensorflow_version magic only works in colab.
    %tensorflow_version 2.x
except Exception:
    pass
import numpy as np
import math, os, sys
import itertools

import matplotlib.pyplot as plt
plt.style.use('default')
from scipy import ndimage

from skimage import measure, morphology
from skimage.io import imsave, imread
from skimage.filters import threshold_otsu
from skimage.transform import resize

import tensorflow as tf
from sklearn import svm, datasets
from sklearn.metrics import confusion_matrix
import pandas as pd

Mengimpor Dataset
#Perintah di bawah ini untuk melihat list bagian dari file yang sudah diunduh dari Github
!ls data_nf/image
!ls data_nf/image/train
!ls data_nf/image/train/Basmathi
!ls data_nf/image/train/IR64
!ls data_nf/image/train/Ketan

#Perintah untuk memvisualisasikan satu gambar dari data train
image = imread("data_nf/image/train/IR64/I_nf017.jpg")
plt.figure(figsize=(3,3))
plt.imshow(image)
#Memuat semua gambar ke memori untuk pertama kali
```

```

#Memuat dataset pelatihan
IMAGE_SIZE = 224
BATCH_SIZE = 128
base_dir = os.path.join('data_nf/image/train')

datagen = tf.keras.preprocessing.image.ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2)

train_generator = datagen.flow_from_directory(
    base_dir,
    target_size=(IMAGE_SIZE, IMAGE_SIZE),
    batch_size=BATCH_SIZE,
    subset='training')

val_generator = datagen.flow_from_directory(
    base_dir,
    target_size=(IMAGE_SIZE, IMAGE_SIZE),
    batch_size=BATCH_SIZE,
    subset='validation')

#Memuat dataset pengujian
X_test = []
y_test = []
labels = ['Basmathi', 'IR64', 'Ketan']

for i,label in enumerate(labels):
    folder = os.path.join("data_nf/image/test",label)
    files = sorted(os.listdir(folder))
    files = [x for x in files if x.endswith(".jpg")]
    for k,file in enumerate(files):
        image_path = os.path.join(folder, file)

        image = imread(image_path)/255.
        image = resize(image, (224,224))
        X_test.append(image)
        category = os.path.split(folder)[-1]
        y_test.append(i)

X_test = np.array(X_test)
y_test = np.array(y_test)

#Menampilkan bentuk dari masing-masing dataset
for image_batch, label_batch in train_generator:
    break
print("Bentuk array dari dataset train (pelatihan) adalah:", 
image_batch.shape,label_batch.shape)
for image_batch, label_batch in val_generator:
    break
print("Bentuk array dari dataset validation (validasi) adalah:", 
image_batch.shape,label_batch.shape)
print("Bentuk array dari dataset test (pengujian) adalah:", 
X_test.shape,y_test.shape)

```

```

Menyimpan Label
print (train_generator.class_indices)

labels_txt =
'\n'.join(sorted(train_generator.class_indices.keys()))

with open('labels.txt', 'w') as f:
    f.write(labels_txt)

!cat labels.txt

Membuat Model dari Jaringan CNN yang Sudah dipelajari Sebelumnya (pre-trained convnets)
IMG_SHAPE = (224, 224, 3)
# Membuat model dasar (base model) dari pre-trained model MobileNetV1
base_model =
tf.keras.applications.MobileNet(input_shape=IMG_SHAPE,
include_top=False,
weights='imagenet')

Feature Extraction
base_model.trainable = False
base_model.summary()

Mengelompokkan Klasifikasi Tiga Varietas Beras dengan Deep learning (Convolutional Neural Network)
import keras
from keras import backend as K
from keras.models import Sequential
from keras import layers
from keras.utils.np_utils import to_categorical

from sklearn.model_selection import train_test_split

Mempersiapkan Data Masukkan Pengujian
y_test2 = to_categorical(y_test)
X_test3, y_test3 = (X_test, y_test2)

Pembuatan Model (Menambah Model)
base_model,
tf.keras.layers.Conv2D(32, 3, activation='relu'),
#tf.keras.layers.Dropout(0.2),
tf.keras.layers.GlobalAveragePooling2D(),
tf.keras.layers.Dense(3, activation='softmax')
[])

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

```

Pelatihan

```
#from keras.callbacks import EarlyStopping, ModelCheckpoint

#Menyimpan file model bobot yang terbaik selama pelatihan
(dalam format keras ".h5")
#ckpt = ModelCheckpoint("Klasifikasi_Beras_Tumpukan.h5",
monitor='val_loss', verbose=1, save_best_only=True,
save_weights_only=False, mode='auto', period=1)
history = model.fit_generator(train_generator,
epochs=100,
validation_data=val_generator)
#history = model.fit(x = X_train3, y = y_train3,
batch_size=120, epochs=
100, validation_data=(X_valid3,y_valid3), callbacks = [ckpt])
```

Menggambarkan Hasil Pelatihan

```
plt.plot(history.history["acc"],label="Akurasi Pelatihan")
plt.plot(history.history["val_acc"],label="Validasi Akurasi")
plt.legend()
plt.show()

plt.plot(history.history["loss"],label="Kesalahan Pelatihan")
plt.plot(history.history["val_loss"],label="Validasi
Kesalahan")
plt.legend()
plt.show()

print('Number of trainable variables =
{}'.format(len(model.trainable_variables)))
```

Menggunakan Model

```
print(train_generator)

#Prediksi Label Validasi dengan Pelatihan
n = 44
input_image = image_batch[n][np.newaxis,...]
print("Labelnya adalah: ", label_batch[n])

predictions = model.predict(input_image)
print("Prediksinya adalah",predictions[0])
```

Evaluasi

```
#Memeriksa matriks model
print(model.metrics_names)
#Evaluasi data test
print(model.evaluate(x= X_test3, y = y_test3))
```

```
#Menampilkan matriks yang benar dan matriks hasil prediksi
#Label yang benar
y_true = np.argmax(y_test2,axis=1)
#Label prediksi
Y_pred = model.predict(X_test)
y_pred = np.argmax(Y_pred, axis=1)
```

```

print(y_true)
print(y_pred)

Prediksi Gambar Individual
n = 44 #Jangan melampaui (nilai dari gambar test - 1)

plt.imshow(X_test[n])
plt.show()

true_label = np.argmax(y_test2, axis=1)[n]
print("Label yang benar
adalah:", true_label, ":", labels[true_label])
prediction = model.predict(X_test[n][np.newaxis, ...])[0]
print("Nilai yang diprediksi adalah:", prediction)
predicted_label = np.argmax(prediction)
print("Label yang diprediksi
adalah:", predicted_label, ":", labels[predicted_label])

if true_label == predicted_label:
    print("Prediksi benar")
else:
    print("Prediksi salah")

Confusion Matrix
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=(5,5))
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='Label Benar',
       xlabel='Label Prediksi')

# 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=labels,
normalize=True,
           title='Normalized confusion matrix')

Menyimpan dan Konversi ke ".tflite"
saved_model_dir = 'save/model'
tf.saved_model.save(model, saved_model_dir)

converter =
tf.lite.TFLiteConverter.from_saved_model(saved_model_dir)
tflite_model = converter.convert()

with open('Klasifikasi_3_Varietas_Beras_MobileNet_nf.tflite',
'wb') as f:
    f.write(tflite_model)

```

Lampiran 3. Kode Sumber Klasifikasi Varietas Beras *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.ricevarietiesclassificationmobilenetv1">

    <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.ricevarietiesclassificationmobilenetv1.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.ricevarietiesclassificationmobilenetv1.Te
        ntang" />
        <activity
            android:name="com.tflite.ricevarietiesclassificationmobilenetv1.Ba
            ntuhan" />
            <activity
                android:name="com.tflite.ricevarietiesclassificationmobilenetv1.De
                skripsi" />
                <activity android:name=".DeteksiDariGaleri"
                    android:screenOrientation="portrait">
                </activity>
                <activity
                    android:name="com.tflite.ricevarietiesclassificationmobilenetv1.Pe
                    ndekksi"
                        android:screenOrientation="portrait"
                        android:colorMode="wideColorGamut">
                    </activity>
                    <activity
                        android:name="com.tflite.ricevarietiesclassificationmobilenetv1.Ma
                        inActivity"
                            android:screenOrientation="portrait">
                        </activity>
                    </application>
    </manifest>

```

2. Bantuan.kt

```

package com.tflite.ricevarietiesclassificationmobilenetv1

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import androidx.appcompat.app.ActionBar

class Bantuan : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_bantuan)

        if (supportActionBar != null) {
            supportActionBar as ActionBar).title = "Halaman
Bantuan"
        }
    }
}

```

3. Deskripsi.kt

```
package com.tflite.ricevarietiesclassificationmobilenetv1

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import androidx.appcompat.app.ActionBar

class Deskripsi : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_deskripsi)

        if (supportActionBar != null) {
            supportActionBar as ActionBar.title = "Deskripsi
Varietas Beras"
        }
    }
}
```

4. Deteksi.kt

```
package com.tflite.ricevarietiesclassificationmobilenetv1

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

5. DeteksiDariGaleri.kt

```
package com.tflite.ricevarietiesclassificationmobilenetv1

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 = 224
    private val mModelPath =
        "Klasifikasi_3_Varietas_Beras_MobileNet.tflite"
        private val mLabelPath =
        "Klasifikasi_3_Varietas_Beras_MobileNet.txt"
        private val mSamplePath = "basmathi.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
Varietas Beras"
        }

        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)
}
}

```

6. Klasifikasi.kt

```

package com.tflite.ricevarietiesclassificationmobilenetv1

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.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 = 224 // 224x224
    private const val BYTES_PER_CHANNEL = 4 // float size
    private const val PIXEL_SIZE = 3 // rgb

    private const val LABELS_PATH =
"Klasifikasi_3_Varietas_Beras_MobileNet.txt"
    private const val MODEL_PATH =
"Klasifikasi_3_Varietas_Beras_MobileNet.tflite"
}

```

```

    }
}
```

7. KlasifikasiDariGaleri.kt

```

package com.tflite.ricevarietiesclassificationmobilenetv1

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, Hasil 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 =
FileInputStream(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,
inputszie, inputszie, false)
    val byteBuffer = convertBitmapToByteBuffer(scaledBitmap)
    val result = Array(1) { FloatArray(labelist.size) }
    interpreter.run(byteBuffer, result)
    return getSortedResult(result)
}

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

    bitmap.getPixels(intValues, 0, bitmap.width, 0, 0,
bitmap.width, bitmap.height)
    var pixel = 0
    for (i in 0 until inputszie) {
        for (j in 0 until inputszie) {
            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, labelist.s

```

```
ize))
```

```
    val pq = PriorityQueue<Recognition>()
        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("".plus(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
}
```

8. MainActivity.kt

```
package com.tflite.ricevarietiesclassificationmobilenetv1

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 btnDeskripsi: Button =
    findViewById(R.id.btn_move_description)
        btnDeskripsi.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_description -> {
                val moveIntent = Intent(this@MainActivity,
Deskripsi::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)
            }
        }
    }
}

```

9. Pendeksi.kt

```

package com.tflite.ricevarietiesclassificationmobilenetv1

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.klasifikasivarietasberas.Klasifikasi
import kotlinx.android.synthetic.main.activity_pendeteksi.*

class Pendekesi : 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
Varietas Beras"
        }

        //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
            prosesstime.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
}
}
}

```

10. SplashScreen.kt

```

package com.tflite.ricevarietiesclassificationmobilenetv1

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.content.Intent
//import android.R
import android.os.Handler
import android.view.Window

class SplashScreen : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        //menghilangkan ActionBar
        this.requestWindowFeature(Window.FEATURE_NO_TITLE)
        setContentView(R.layout.activity_splash_screen)

        //tvSplash = findViewById(R.id.tvSplash) as TextView

        //setelah loading maka akan langsung berpindah ke
        MainActivity
        val handler = Handler()
        handler.postDelayed(Runnable {
            startActivity(Intent(applicationContext,
            MainActivity::class.java))
            finish()
        }, 3000L) //3000 L = 3 detik
    }
}

```

```

    }
}
}
```

11. Tentang.kt

```

package com.tflite.ricevarietiesclassificationmobilenetv1

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import androidx.appcompat.app.ActionBar

class Tentang : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_tentang)

        if (supportActionBar != null) {
            supportActionBar as ActionBar).title = "Tentang
Applikasi"
        }
    }
}
```

12. gradient_pendeteksi.xml

```

<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
    <gradient
        android:type="linear"
        android:startColor="@android:color/holo_blue_bright"
        android:centerColor="#FFFF66"
        android:endColor="@android:color/holo_green_light"
        android:angle="315"/>
</shape>
```

13. gradient_splash_screen.xml

```

<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
    <gradient
        android:type="linear"
        android:startColor="@android:color/holo_blue_bright"
        android:endColor="@android:color/holo_green_light"
        android:angle="-90"/>
</shape>
```

14. gradient_tentang.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
    <gradient
        android:type="linear"
        android:startColor="@android:color/holo_blue_bright"
        android:centerColor="#FFFF66"
        android:endColor="@android:color/holo_green_light"
        android:angle="225"/>
</shape>
```

15. activity_bantuan.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/background"
    tools:context=".Bantuan">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textSize="20sp"
            android:textStyle="bold"
            android:fontFamily="times"
            android:text="1. Cara Kerja Aplikasi"
            android:layout_marginLeft="16dp"
            android:layout_marginRight="16dp"
            android:layout_marginTop="16dp"
            android:layout_marginBottom="8dp"
            android:textColor="@android:color/holo_green_dark"/>
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/content_work"
            android:background="#80FF6600"
            android:layout_marginLeft="16dp"
            android:layout_marginRight="16dp"
            android:layout_marginBottom="16dp"
            android:lineSpacingMultiplier="1"
            android:textColor="#FFFFFF"/>
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textSize="20sp"
```

```
        android:textStyle="bold"
        android:fontFamily="times"
        android:text="2. Cara Menggunakan"
        android:layout_marginLeft="16dp"
        android:layout_marginRight="16dp"
        android:layout_marginTop="16dp"
        android:layout_marginBottom="8dp"
        android:textColor="@android:color/holo_green_dark"/>
<TableLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="16dp"
    android:layout_marginRight="16dp"
    android:layout_marginBottom="16dp">
<TableRow
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp">
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="14sp"
    android:layout_marginRight="16dp"
    android:textColor="#FFFFFF"
    android:text="1. "/>
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textSize="14sp"
    android:layout_weight="1"
    android:text="@string/content_step_1"
    android:textColor="#FFFFFF"/>
</TableRow>
<TableRow
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp">
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="14sp"
    android:layout_marginRight="16dp"
    android:textColor="#FFFFFF"
    android:text="2. "/>
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textSize="14sp"
    android:layout_weight="1"
    android:text="@string/content_step_2"
    android:textColor="#FFFFFF"/>
</TableRow>
<TableRow
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp">
```

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="14sp"
    android:layout_marginRight="16dp"
    android:textColor="#FFFFFF"
    android:text="3. "/>
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textSize="14sp"
    android:layout_weight="1"
    android:text="@string/content_step_3"
    android:textColor="#FFFFFF"/>
</TableRow>
<TableRow
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp">
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="14sp"
    android:layout_marginRight="16dp"
    android:textColor="#FFFFFF"
    android:text="4. "/>
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textSize="14sp"
    android:layout_weight="1"
    android:text="@string/content_step_4"
    android:textColor="#FFFFFF"/>
</TableRow>
<TableRow
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp">
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="14sp"
    android:layout_marginRight="16dp"
    android:textColor="#FFFFFF"
    android:text="5. "/>
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textSize="14sp"
    android:layout_weight="1"
    android:text="@string/content_step_5"
    android:textColor="#FFFFFF"/>
</TableRow>
<TableRow
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
```

```

        android:layout_marginBottom="8dp">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="14sp"
        android:layout_marginRight="16dp"
        android:textColor="#FFFFFF"
        android:text="6. "/>
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="14sp"
        android:layout_weight="1"
        android:text="@string/content_step_6"
        android:textColor="#FFFFFF"/>
</TableRow>
<TableRow
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="14sp"
        android:layout_marginRight="16dp"
        android:textColor="#FFFFFF"
        android:text="7. "/>
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="14sp"
        android:layout_weight="1"
        android:text="@string/content_step_7"
        android:textColor="#FFFFFF"/>
</TableRow>
</TableLayout>
</LinearLayout>
</ScrollView>

```

16. activity_deskripsi.xml

```

<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/background"
    tools:context=".Deskripsi">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"

```

```
    android:orientation="vertical">
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textSize="26sp"
    android:fontFamily="times"
    android:text="Deskripsi Tiga Varietas Beras (Basmathi,
IR-64, dan Ketan)"
    android:layout_marginLeft="16dp"
    android:layout_marginRight="16dp"
    android:layout_marginTop="16dp"
    android:layout_marginBottom="8dp"
    android:textColor="@android:color/holo_blue_dark"/>
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textStyle="bold"
    android:text="1. Varietas Beras Basmathi"
    android:textSize="20sp"
    android:layout_marginLeft="16dp"
    android:layout_marginRight="16dp"
    android:layout_marginBottom="16dp"
    android:textColor="@android:color/holo_green_dark"/>
<FrameLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
    <ImageView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:adjustViewBounds="true"
        android:scaleType="fitCenter"
        android:src="@drawable/basmathi" />
</FrameLayout>
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/content_basmathi"
    android:background="#60FFFFFF"
    android:layout_marginLeft="16dp"
    android:layout_marginRight="16dp"
    android:layout_marginBottom="16dp"
    android:lineSpacingMultiplier="1"
    android:textColor="#FFFFFF"
    android:autoLink="all"
    android:linksClickable="true"/>
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textStyle="bold"
    android:text="2. Varietas Beras IR-64"
    android:textSize="20sp"
    android:layout_marginLeft="16dp"
    android:layout_marginRight="16dp"
    android:layout_marginBottom="16dp"
```

```
        android:textColor="@android:color/holo_green_dark" />
<FrameLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <ImageView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:adjustViewBounds="true"
        android:scaleType="fitCenter"
        android:src="@drawable/ir64" />
</FrameLayout>
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/content_ir64"
    android:background="#60FFFFFF"
    android:layout_marginLeft="16dp"
    android:layout_marginRight="16dp"
    android:layout_marginBottom="16dp"
    android:lineSpacingMultiplier="1"
    android:textColor="#FFFFFF"/>

<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textStyle="bold"
    android:text="3. Varietas Beras Ketan"
    android:textSize="20sp"
    android:layout_marginLeft="16dp"
    android:layout_marginRight="16dp"
    android:layout_marginBottom="16dp"
    android:textColor="@android:color/holo_green_dark" />
<FrameLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <ImageView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:adjustViewBounds="true"
        android:scaleType="fitCenter"
        android:src="@drawable/ketan" />
</FrameLayout>
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/content_ketan"
    android:background="#60FFFFFF"
    android:layout_marginLeft="16dp"
    android:layout_marginRight="16dp"
    android:layout_marginBottom="16dp"
    android:lineSpacingMultiplier="1"
    android:textColor="#FFFFFF"
    android:autoLink="all"
    android:linksClickable="true" />
```

```
</LinearLayout>
</ScrollView>
```

17. activity_import_gallery

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/gradient_pendeteksi"
    tools:context=".DeteksiDariGaleri">

    <TextView
        android:id="@+id/delaytime"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#CCFF6633"
        android:padding="2dp"
        android:fontFamily="times"
        android:gravity="right"
        android:text="@string/delay_time"
        android:textAlignment="gravity"
        android:textColor="#FFFF00"
        android:textSize="18sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.0"
        tools:layout_editor_absoluteX="0dp" />

    <TextView
        android:id="@+id/process_name_bar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="2dp"
        android:fontFamily="times"
        android:gravity="left"
        android:text=" Waktu Proses Deteksi : "
        android:textAlignment="gravity"
        android:textColor="#FFFF00"
        android:textSize="18sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.0"
        tools:layout_editor_absoluteX="0dp" />

    <TextView
        android:id="@+id/hints_bar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#80000000"
        android:fontFamily="times"
```

```

        android:gravity="center"
        android:padding="2dp"
        android:text="@string/hints"
        android:textAlignment="gravity"
        android:textColor="#FFFFFF"
        android:textSize="16sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.05"
        tools:layout_editor_absoluteX="0dp" />

<Button
        android:id="@+id/mGalleryButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="0dp"
        android:text="@string/buttonSelectPhoto"
        android:textColor="@android:color/holo_green_dark"
        android:textSize="18sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/hints_bar" />

<ImageView
        android:id="@+id/mPhotoImageView"
        android:layout_width="360dp"
        android:layout_height="360dp"
        android:contentDescription="@string/descriptionImage"
        app:layout_constraintBottom_toTopOf="@+id/mDetectButton"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/mGalleryButton"
        app:layout_constraintVertical_chainStyle="packed"
        app:srcCompat="@android:color/darker_gray" />

<Button
        android:id="@+id/mDetectButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:background="#8000CC33"
        android:padding="7dp"
        android:shadowColor="@android:color/darker_gray"
        android:text="Deteksi"
        android:textColor="@android:color/black"
        android:textSize="24sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintBottom_toTopOf="@+id/mResultTextView"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" />

<TextView
        android:id="@+id/mResultTextView"

```

```

        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="0dp"
        android:background="#9000CC33"
        android:fontFamily="times"
        android:text="@string/display_result"
        android:textAlignment="center"
        android:textColor="#000099"
        android:textSize="16sp"
        android:textStyle="bold"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>

```

18. activity_main.xml

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
    android:orientation="vertical"
    android:background="@drawable/background"
    android:padding="16dp">

    <Button
        android:id="@+id/btn_move_detection"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="64dp"
        android:layout_marginBottom="32dp"
        android:textColor="@android:color/holo_green_dark"
        android:text="@string/pendeksi"
        android:textStyle="bold"/>

    <Button
        android:id="@+id/btn_move_import"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="32dp"
        android:textColor="@android:color/holo_green_dark"
        android:text="@string/import_gallery"
        android:textStyle="bold"/>

    <Button
        android:id="@+id/btn_move_description"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="32dp"
        android:textColor="@android:color/holo_green_dark"

```

```

        android:text="@string/deskripsi"
        android:textStyle="bold"/>

<Button
        android:id="@+id	btn_move_help"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="32dp"
        android:textColor="@android:color/holo_green_dark"
        android:text="@string/bantuan"
        android:textStyle="bold"/>

<Button
        android:id="@+id	btn_move_about"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="32dp"
        android:textColor="@android:color/holo_green_dark"
        android:text="@string/tentang"
        android:textStyle="bold"/>

</LinearLayout>

```

19. activity_pendeteksi.xml

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/gradient_pendeteksi"
    tools:context=".Pendeteksi">

    <com.priyankvasa.android.cameralayout.CameraView
        android:id="@+id/camera"
        android:layout_width="480dp"
        android:layout_height="640dp"
        android:layout_marginBottom="0dp"
        android:adjustViewBounds="true"
        android:keepScreenOn="true"
        app:aspectRatio="4:3"
        app:autoFocus="continuous_picture"
        app:awb="auto"
        app:cameraMode="single_capture"
        app:facing="back"
        app:flash="auto"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:opticalStabilization="true"
        app:outputFormat="jpeg"

```

```
    app:pinchToZoom="true"
    app:shutter="short_time"
    app:touchToFocus="true"
    app:zsl="true" />

<TextView
    android:id="@+id/process_time_bar"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#CCFF6633"
    android:padding="2dp"
    android:fontFamily="times"
    android:gravity="right"
    android:text="@string/process_time"
    android:textAlignment="gravity"
    android:textColor="#FFFF00"
    android:textSize="18sp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.0"
    tools:layout_editor_absoluteX="0dp" />

<TextView
    android:id="@+id/process_name_bar"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="2dp"
    android:fontFamily="times"
    android:gravity="left"
    android:text=" Waktu Proses Deteksi : "
    android:textAlignment="gravity"
    android:textColor="#FFFF00"
    android:textSize="18sp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.0"
    tools:layout_editor_absoluteX="0dp" />

<TextView
    android:id="@+id/info_bar"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#80000000"
    android:fontFamily="times"
    android:gravity="center"
    android:padding="2dp"
    android:text="@string/info"
    android:textAlignment="gravity"
    android:textColor="#FFFFFF"
    android:textSize="16sp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.05"
    tools:layout_editor_absoluteX="0dp" />

<TextView
```

```

        android:id="@+id/capturePhoto"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="84dp"
        android:background="#8000CC33"
        android:padding="7dp"
        android:shadowColor="@android:color/darker_gray"
        android:text="Deteksi"
        android:textColor="@android:color/black"
        android:textSize="30sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent" />

<TextView
        android:id="@+id/prediction_bar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="63dp"
        android:background="#9000CC33"
        android:fontFamily="times"
        android:text="HASIL PREDIKSI :"
        android:textAlignment="center"
        android:textColor="#000099"
        android:textSize="16sp"
        app:layout_constraintBottom_toBottomOf="parent" />

<TextView
        android:id="@+id/result_bar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="0dp"
        android:background="#9000CC33"
        android:fontFamily="times"
        android:text="@string/hasil_1_nhasil_2_nhasil_3"
        android:textAlignment="center"
        android:textColor="#000099"
        android:textSize="16sp"
        app:layout_constraintBottom_toBottomOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>

```

20. activity_splash_screen.xml

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/activity_splash_screen"
    android:background="@drawable/gradient_splash_screen"
    android:gravity="center" />

```

```

tools:context=".SplashScreen">

<TextView
    android:id="@+id/tvSplash"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_alignParentStart="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentTop="true"
    android:layout_marginTop="280dp"
    android:fontFamily="sans-serif-smallcaps"
    android:text="Klasifikasi Varietas Beras"
    android:textAlignment="center"
    android:textColor="@android:color/holo_green_dark"
    android:textSize="24sp"
    android:textStyle="bold" />

<ImageView
    android:id="@+id/logo_kvb"
    android:layout_width="match_parent"
    android:layout_height="100dp"
    android:layout_alignParentStart="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentTop="true"
    android:layout_marginTop="176dp"
    android:scaleType="centerInside"
    android:src="@drawable/logo_kvb" />

<ImageView
    android:id="@+id/logo_tflite"
    android:layout_width="match_parent"
    android:layout_height="20dp"
    android:layout_alignParentStart="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentTop="true"
    android:layout_marginTop="320dp"
    android:scaleType="centerInside"
    android:src="@drawable/tflite" />

<ProgressBar
    android:id="@+id/progresBar"
    style="@style/Widget.AppCompat.ProgressBar"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_centerHorizontal="true"
    android:layout_marginBottom="60dp" />

</RelativeLayout>

```

21. activity_tentang.xml

```

<?xml version="1.0" encoding="utf-8"?>
<ScrollView

```

```
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:background="@drawable/gradient_tentang"
android:textAlignment="center"
tools:context=".Tentang">

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="24sp"
        android:fontFamily="times"
        android:text="Klasifikasi Varietas Beras"
        android:background="#20000000"
        android:textAlignment="center"
        android:layout_marginLeft="0dp"
        android:layout_marginRight="0dp"
        android:layout_marginTop="16dp"
        android:layout_marginBottom="8dp"
        android:textColor="@android:color/holo_blue_dark"/>
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Versi 2.0"
        android:fontFamily="times"
        android:background="#80FFFFFF"
        android:textSize="16sp"
        android:textAlignment="center"
        android:layout_marginLeft="16dp"
        android:layout_marginRight="16dp"
        android:layout_marginBottom="16dp"
        android:textColor="@android:color/holo_green_dark"/>
    <TextView
        android:textSize="14dp"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/update"
        android:layout_marginLeft="16dp"
        android:layout_marginRight="16dp"
        android:layout_marginBottom="16dp"
        android:lineSpacingMultiplier="1"
        android:textColor="#000099"/>
    <TableLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="16dp"
        android:layout_marginRight="16dp"
        android:layout_marginBottom="16dp">
        <TableRow
            android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="14sp"
        android:layout_marginRight="16dp"
        android:textColor="#000099"
        android:text="1. "/>
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="14sp"
        android:layout_weight="1"
        android:text="@string/content_update_1"
        android:textColor="#000099"/>
</TableRow>
<TableRow
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="14sp"
        android:layout_marginRight="16dp"
        android:textColor="#000099"
        android:text="2. "/>
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="14sp"
        android:layout_weight="1"
        android:text="@string/content_update_2"
        android:textColor="#000099"/>
</TableRow>
</TableLayout>
<TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/content_about"
        android:layout_marginLeft="16dp"
        android:layout_marginRight="16dp"
        android:layout_marginBottom="16dp"
        android:lineSpacingMultiplier="1"
        android:textColor="@android:color/black"
        android:autoLink="all"
        android:linksClickable="true"/>
<TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textAlignment="center"
        android:text="Author : Vidi Fitriansyah Hidarlan"
        android:layout_marginTop="50dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
```

```

        app:layout_constraintRight_toRightOf="parent"
        android:textColor="@android:color/black"/>
    </LinearLayout>
</ScrollView>
```

22. strings.xml

```

<resources>
    <string name="app_name">KVB MobileNetV1</string>
    <string name="pendekripsi">Deteksi Beras dengan Kamera</string>
    <string name="import_gallery">Deteksi Beras dari Galeri</string>
    <string name="deskripsi">Deskripsi Varietas Beras</string>
    <string name="content_basmathi">Beras basmathi berasal dari negara india/pakistan, beras basmathi berasal dari bahasa sanskerta 'basmathi' artinya berarti harum atau wangi dalam bahasa india ia juga mempunyai maksud "soft rice" yaitu lembut.\n
        Beras ini akan melar memanjang butiran, sedikit pera, mudah terurai butiran berasnya dan aromanya sangat harum, beras ini memang aromanya sangat harum. Keistimewaan lainnya, butiran atau buliran beras ini panjang dan kecil melebihi ukuran beras yang biasanya agak pendek dan bulat.\n
        Sumber : https://www.asshidiqaqiqah.com/kandungan-kandungan-beras-basmati/</string>
    <string name="content_ir64">Beras IR 64 merupakan salah satu jenis varietas padi sawah yang memiliki bentuk tegak, tinggi sekitar 115-126 cm. Gabahnya berbentuk ramping dan panjang dan berwarna kuning bersih. Padi jenis ini cocok ditanam di daerah lahan sawah irigasi dataran rendah sampai sedang.\n
        Beras IR 64 adalah jenis beras yang pulen jika dimasak menjadi nasi karena memiliki kadar amilosa sebanyak 23%. Bobot beras per seribu butir beras IR64 adalah 24,1 gram.\n
        Sumber : Suprihatno, B, Daradjat, dkk. 2010. Deskripsi Varietas Padi. Balai Besar Penelitian Tanaman Padi.</string>
    <string name="content_ketan">Beras ketan putih (Oryza sativa glutinosa) merupakan salah satu varietas padi yang termasuk dalam famili Graminae.\n
        Butir beras sebagian besar terdiri dari zat pati sekitar 80-85% yang terdapat dalam endosperma yang tersusun oleh granula-granula pati yang berukuran 3-10 milimikron. Beras ketan juga mengandung vitamin (terutama pada bagian aleuron), mineral dan air. Dari komposisi kimiawinya diketahui bahwa karbohidrat penyusun utama beras ketan adalah pati. Pati merupakan karbohidrat polimer glukosa yang mempunyai dua struktur yakni amilosa dan amilopektin.\n
        Sumber :
http://www.alatcetakrengginang.com/2012/02/beras-ketan-sifat-fisika-kimianya.html</string>
    <string name="tentang">Tentang Aplikasi</string>
    <string name="content_about">Aplikasi ini dibuat hanya untuk klasifikasi tiga varietas beras (Basmathi, IR-64, dan Ketan) menggunakan sebuah metode CNN (<i>Convolutional Neural Network</i>) dengan model arsitektur MobileNetV1. Arsitektur tersebut dilatih pada <i>Google Colaboratory</i>. Setelah dilatih
```

disimpan modelnya dan dikonversi menggunakan <i>TensorFlow Lite</i>. Setelah dikonversi diunduh <i>file</i> modelnya dan di-<i>import</i> untuk diimplementasikan pada <i>smartphone</i> Android agar dapat melakukan klasifikasi varietas beras secara langsung menggunakan sebuah perangkat Android.\n\n

Pada aplikasi ini juga diatur secara otomatis penggunaan <i>flash</i> kameranya agar perubahan cahaya yang ada tidak berbeda signifikan ketika berada di luar ruangan yang banyak cahaya maupun di dalam ruangan yang minim cahaya karena cahaya <i>flash</i> kamera <i>smartphone</i> dapat menyala dan mati secara otomatis berdasarkan intensitas cahaya yang ada. Hal itu dapat mempermudah pengguna untuk mendeteksi varietas berasnya.\n\n

<i>Sourcecode</i> program pada <i>Google Colaboratory</i> dan <i>Android Studio</i> : <https://github.com/Soedirman-Machine-Learning/rice-varieties-classification>\natau <https://github.com/Vidi005/Klasifikasi-3-Varietas-Beras>

<string name="bantuan">Bantuan</string>

<string name="content_work">Aplikasi ini bekerja dengan cara mengambil citra/gambar obyek dari varietas beras menggunakan kamera <i>smartphone</i> atau mengimpor gambar dari galeri foto yang sudah tersimpan. Setelah itu diekstrak fitur dari piksel obyek tersebut dan diubah ke dalam bentuk <i>array</i> dengan ukuran piksel 224 x 224 dengan format gambar berwarna (RGB). Kemudian dilatih dan diproses menggunakan model yang sudah dilatih sebelumnya sehingga dapat diprediksi hasil keluaran dari varietas beras yang dideteksi.</string>

<string name="content_step_1">Buka Aplikasi Klasifikasi Varietas Beras kemudian pilih menu "DETEKSI BERAS DENGAN KAMERA" untuk klasifikasi secara langsung</string>

<string name="content_step_2">Setelah terbuka menu tersebut arahkan kamera ke obyek varietas beras yang ingin dideteksi (jarak kamera dengan obyek sekitar 10 cm), kemudian klik/tap tombol "Deteksi"</string>

<string name="content_step_3">Hasil prediksi akan muncul beberapa saat setelah menekan tombol "Deteksi" di bagian bawah layar dalam persentase (% diurutkan dari yang tertinggi). Persentase tertinggi merupakan hasil prediksi yang paling mendekati/mirip</string>

<string name="content_step_4">Buka menu "DETEKSI BERAS DARI GALERI" jika ingin mengklasifikasi secara manual dengan mengimpor gambar yang sudah tersimpan di galeri foto</string>

<string name="content_step_5">Klik tombol "PILIH GAMBAR" dan cari gambar obyek beras yang ingin dideteksi</string>

<string name="content_step_6">Selanjutnya tekan tombol DETEKSI dan hasil prediksinya akan muncul beberapa saat setelah menekannya di bagian bawah layar</string>

<string name="content_step_7">Jika Anda ingin mengetahui penjelasan singkat tiap varietas beras pilih menu "DESKRIPSI VARIETAS BERAS", Menu "BANTUAN" untuk mengetahui cara penggunaan aplikasi ini, dan Menu "TENTANG APLIKASI" sebagai informasi pembuatan aplikasi ini.</string>

<string name="hasil_1_nhasil_2_nhasil_3">Hasil 1\nHasil 2\nHasil 3</string>

<string name="process_time">... ms </string>

<string name="delay_time">... ms </string>

```

<string name="info">Klik DETEKSI untuk Klasifikasi</string>
<string name="buttonSelectPhoto">Pilih Gambar</string>
<string name="descriptionImage">Lihat untuk Menampilkan
Gambar</string>
<string name="display_result"><b>Hasil Prediksi</b>\n</string>
<string name="hints">Pilih Gambar Lalu Klik DETEKSI</string>
<string name="update"><b>Pembaruan Fitur :</b></string>
<string name="content_update_1">Versi 1.0\n- Klasifikasi 3
varietas beras dengan kamera\n- Memperbaiki <i>bug force
closed</i>\n- Mengubah aspek rasio kamera menjadi 4:3\n- Menambah
fitur waktu respon deteksi</string>
<string name="content_update_2">Versi 2.0\n- Menambah fitur
deteksi dengan impor galeri</string>
</resources>

```

23. styles.xml

```

<resources>

    <!-- Base application theme. -->
    <style name="AppThemeSplashScreen"
parent="Theme.AppCompat.Light.NoActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item
name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>
    <!-- Main application theme. -->
    <style name="AppTheme"
parent="Theme.AppCompat.Light.DarkActionBar">
        <!-- Customize your theme here. -->
        <item
name="colorPrimary">@android:color/holo_blue_dark</item>
        <item
name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>
</resources>

```

24. build.gradle (Module: app)

```

apply plugin: 'com.android.application'

apply plugin: 'kotlin-android'

apply plugin: 'kotlin-android-extensions'

android {
    compileSdkVersion 28
    defaultConfig {

```

```

        applicationId
"com.tflite.ricevarietiesclassificationmobilenetv1"
        minSdkVersion 15
        targetSdkVersion 28
        versionCode 2
        versionName "2.0"
        testInstrumentationRunner
"androidx.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-
android-optimize.txt'), 'proguard-rules.pro'
        }
    }
    aaptOptions {
        noCompress "tflite"
        noCompress "lite"
    }
    packagingOptions {//mencegah duplikasi modul kotlin saat build
aplikasi
        pickFirst 'META-INF/kotlinx-io.kotlin_module'
        pickFirst 'META-INF/atomicfu.kotlin_module'
        pickFirst 'META-INF/kotlinx-coroutines-io.kotlin_module'
    }
}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation "org.jetbrains.kotlin:kotlin-stdlib-
jdk7:$kotlin_version"
    implementation 'androidx.appcompat:appcompat:1.1.0'
    implementation 'androidx.core:core-ktx:1.1.0'
    implementation
'androidx.constraintlayout:constraintlayout:1.1.3'
    //menambah tampilan kamera
    implementation "com.priyankvasa.android:cameraview-ex:3.5.1-
alpha"
    //menambah library tensorflowlite
    implementation 'org.tensorflow:tensorflow-lite:1.13.1'
    //menampilkan sebuah custom ImageView dalam bentuk lingkaran
    implementation 'de.hdodenhof:circleimageview:3.0.0'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test.ext:junit:1.1.1'
    androidTestImplementation 'androidx.test.espresso:espresso-
core:3.2.0'
}

```

25. settings.gradle (Project Settings)

```
include ':app'  
rootProject.name='Rice Varieties Classification MobileNetV1'
```

Selengkapnya dapat dibuka pada *Github* dengan *link* sebagai berikut:

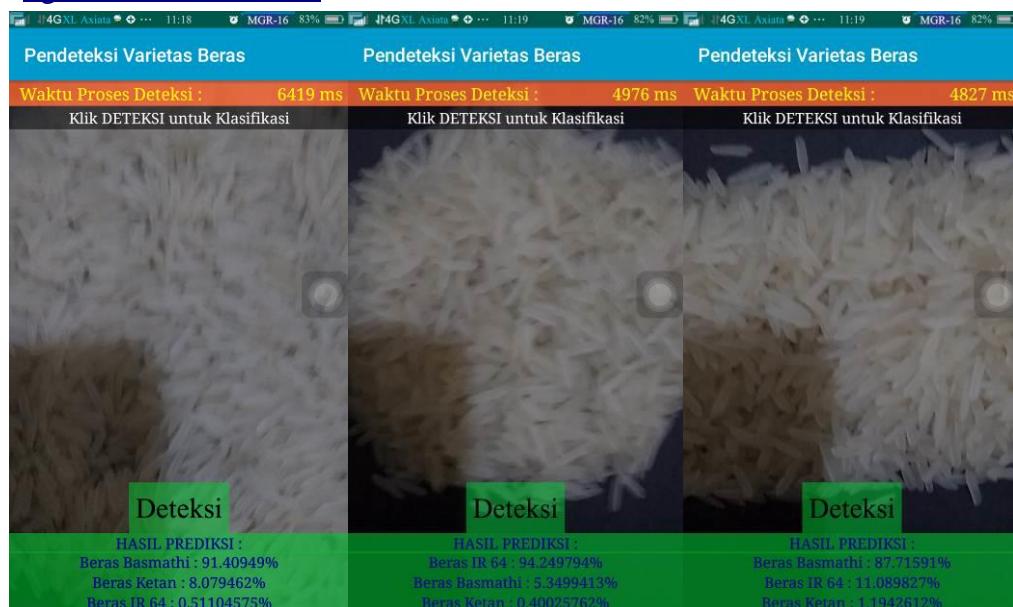
<https://github.com/Soedirman-Machine-Learning/rice-varieties-classification>

Lampiran 4. Sampel Screenshot Pengujian Arsitektur VGG-16Net dengan Flashlight pada Aplikasi Android

1. Varietas Beras Basmathi

Selengkapnya dapat dilihat pada *link* berikut :

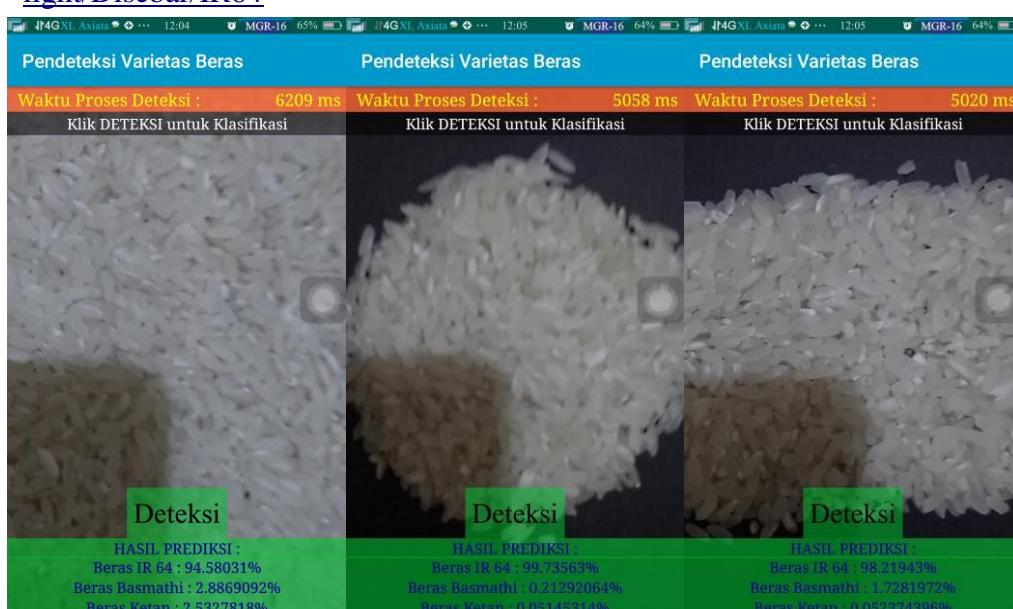
<https://github.com/Soedirman-Machine-Learning/rice-varieties-classification/tree/master/Screenshot%20Hasil%20Pengujian/VGG16Net/Flashlight/Disebar/Basmathi>



2. Varietas Beras IR-64

Selengkapnya dapat dilihat pada *link* berikut :

<https://github.com/Soedirman-Machine-Learning/rice-varieties-classification/tree/master/Screenshot%20Hasil%20Pengujian/VGG16Net/Flashlight/Disebar/IR64>



3. Varietas Beras Ketan

Selengkapnya dapat dilihat pada *link* berikut :

<https://github.com/Soedirman-Machine-Learning/rice-varieties-classification/tree/master/Screenshot%20Hasil%20Pengujian/VGG16Net/Flashlight/Disebar/Ketan>

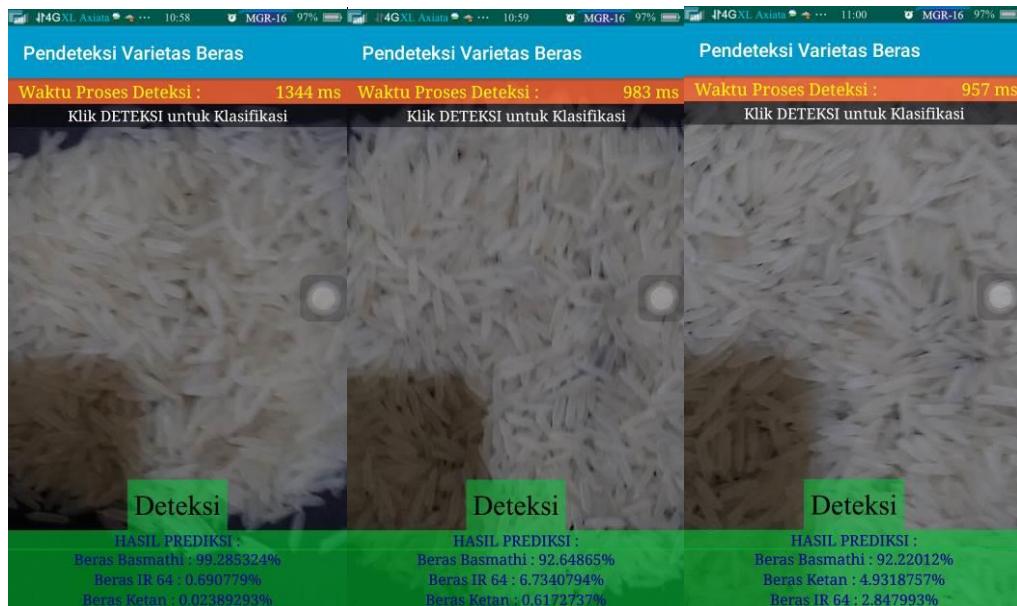


Lampiran 5. Sampel Screenshot Pengujian Arsitektur MobileNetV1 dengan Flashlight pada Aplikasi Android

1. Varietas Beras Basmathi

Selengkapnya dapat dilihat pada *link* berikut :

<https://github.com/Soedirman-Machine-Learning/rice-varieties-classification/tree/master/Screenshot%20Hasil%20Pengujian/MobileNetV1/Flashlight/Disebar/Basmathi>



2. Varietas Beras IR-64

Selengkapnya dapat dilihat pada *link* berikut :

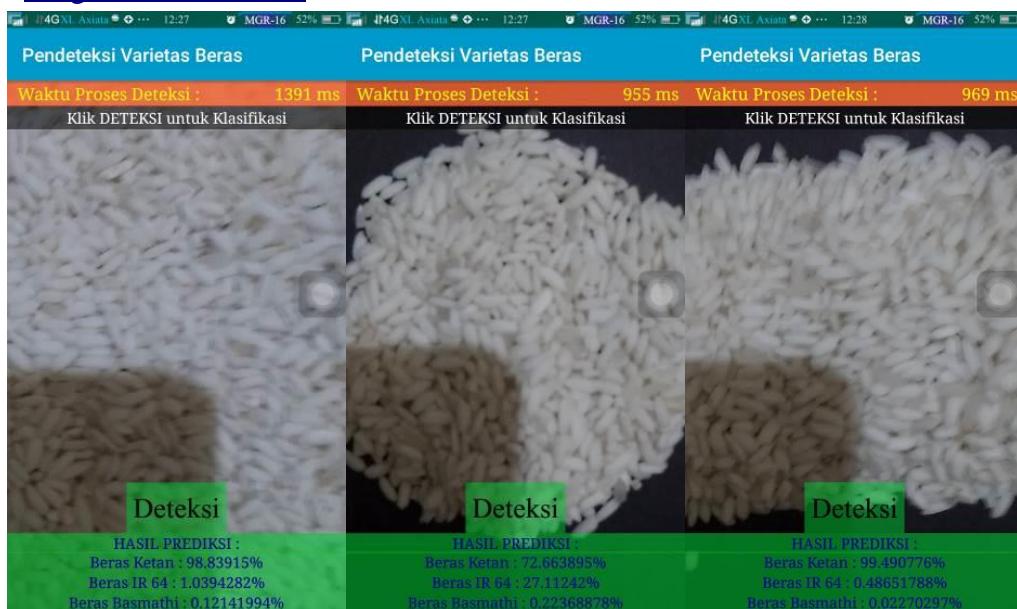
<https://github.com/Soedirman-Machine-Learning/rice-varieties-classification/tree/master/Screenshot%20Hasil%20Pengujian/MobileNetV1/Flashlight/Disebar/IR64>



3. Varietas Beras Ketan

Selengkapnya dapat dilihat pada *link* berikut :

<https://github.com/Soedirman-Machine-Learning/rice-varieties-classification/tree/master/Screenshot%20Hasil%20Pengujian/MobileNetV1/Flashlight/Disebar/Ketan>



Lampiran 6. Sampel Screenshot Pengujian Arsitektur MobileNetV1 tanpa Flashlight pada Aplikasi Android

1. Varietas Beras Basmuthi

Selengkapnya dapat dilihat pada link berikut :

<https://github.com/Soedirman-Machine-Learning/rice-varieties-classification/tree/master/Screenshot%20Hasil%20Pengujian/MobileNetV1/No%20Flashlight/Disebar/Basmuthi>



2. Varietas Beras IR-64

Selengkapnya dapat dilihat pada link berikut :

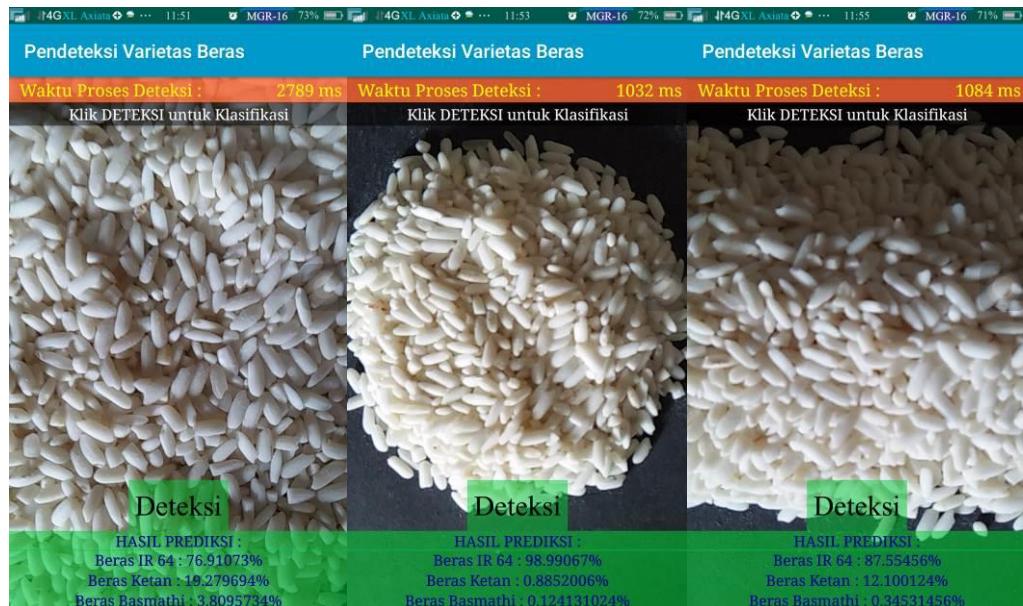
<https://github.com/Soedirman-Machine-Learning/rice-varieties-classification/tree/master/Screenshot%20Hasil%20Pengujian/MobileNetV1/No%20Flashlight/Disebar/IR64>



3. Varietas Beras Ketan

Selengkapnya dapat dilihat pada *link* berikut :

<https://github.com/Soedirman-Machine-Learning/rice-varieties-classification/tree/master/Screenshot%20Hasil%20Pengujian/MobileNetV1/No%20Flashlight/Disebar/Ketan>



Lampiran 7. Sampel Screenshot Pengujian Arsitektur MobileNetV1 Menggunakan Flashlight dengan beras dibungkus pada Aplikasi Android

1. Varietas Beras Basmathi

Selengkapnya dapat dilihat pada *link* berikut :

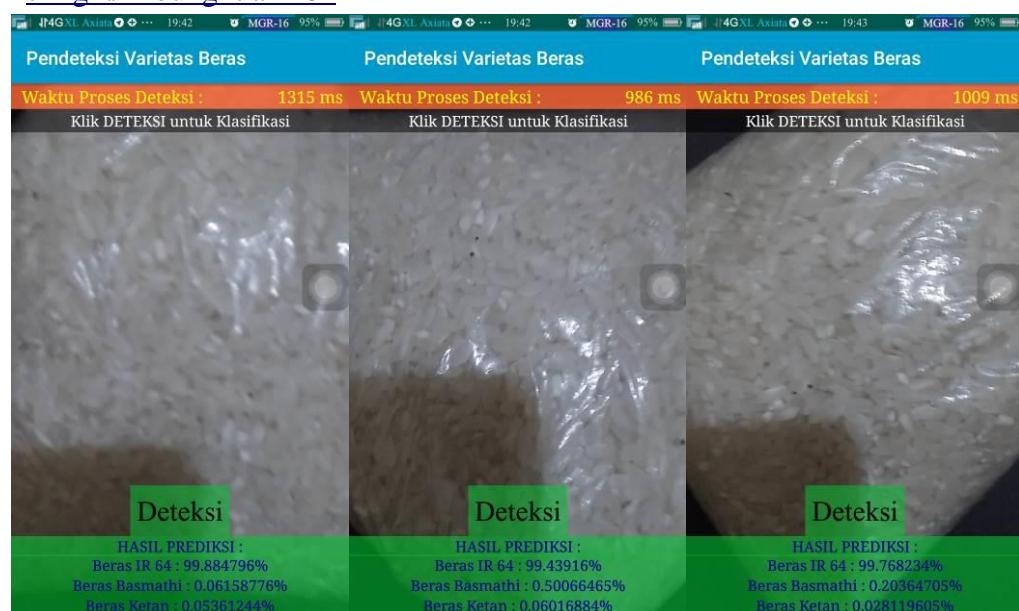
<https://github.com/Soedirman-Machine-Learning/rice-varieties-classification/tree/master/Screenshot%20Hasil%20Pengujian/MobileNetV1/Flashlight/Dibungkus/Basmathi>



2. Varietas Beras IR-64

Selengkapnya dapat dilihat pada *link* berikut :

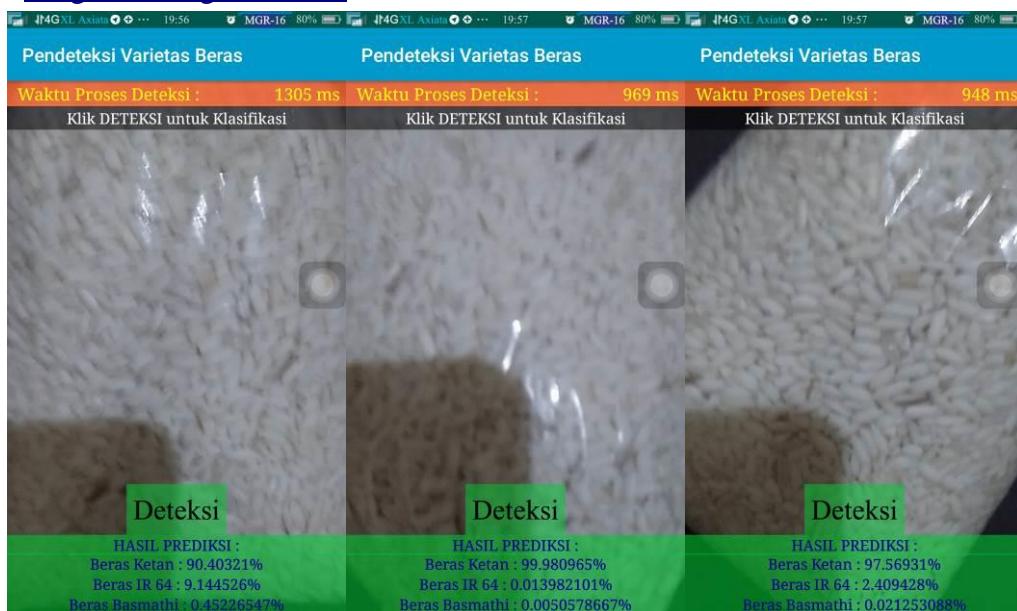
<https://github.com/Soedirman-Machine-Learning/rice-varieties-classification/tree/master/Screenshot%20Hasil%20Pengujian/MobileNetV1/Flashlight/Dibungkus/IR64>



3. Varietas Beras Ketan

Selengkapnya dapat dilihat pada link berikut :

<https://github.com/Soedirman-Machine-Learning/rice-varieties-classification/tree/master/Screenshot%20Hasil%20Pengujian/MobileNetV1/Flashlight/Dibungkus/Ketan>



Lampiran 8. Sampel Screenshot Pengujian Arsitektur MobileNetV1 tanpa Flashlight dengan beras dibungkus pada Aplikasi Android

1. Varietas Beras Basmathi

Selengkapnya dapat dilihat pada *link* berikut :

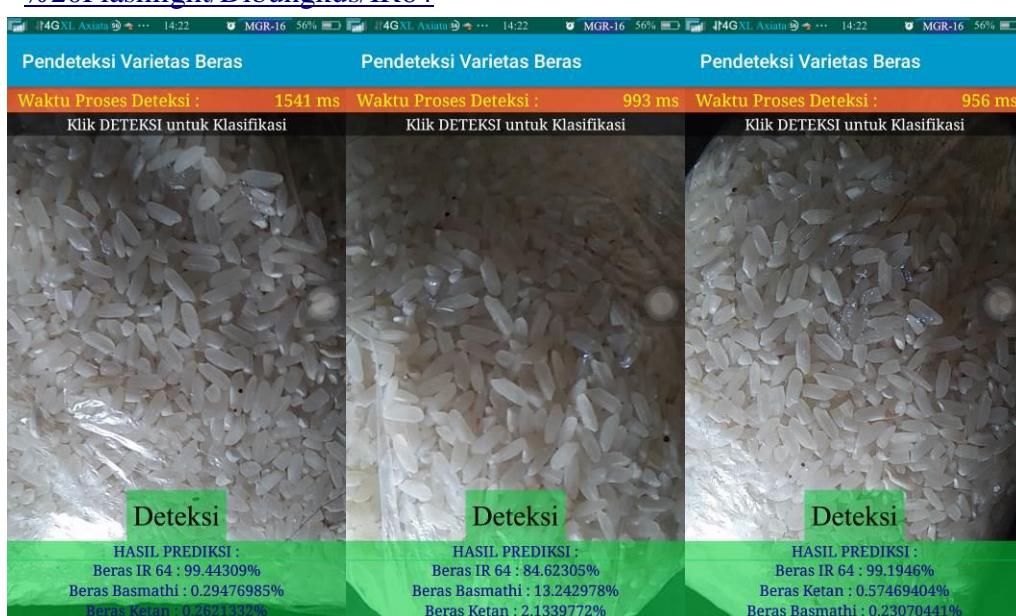
<https://github.com/Soedirman-Machine-Learning/rice-varieties-classification/tree/master/Screenshot%20Hasil%20Pengujian/MobileNetV1/No%20Flashlight/Dibungkus/Basmathi>



2. Varietas Beras IR-64

Selengkapnya dapat dilihat pada *link* berikut :

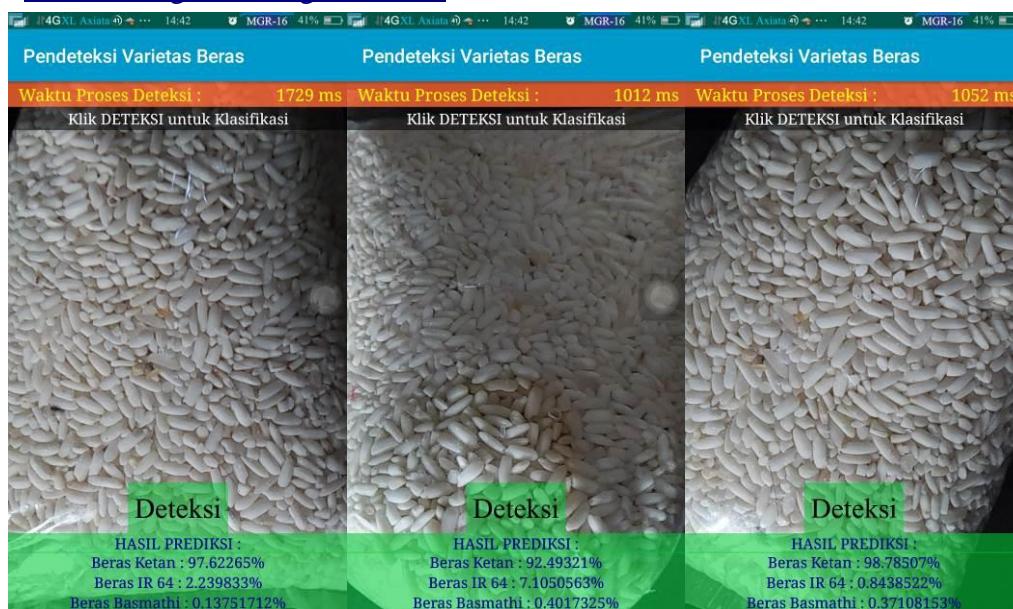
<https://github.com/Soedirman-Machine-Learning/rice-varieties-classification/tree/master/Screenshot%20Hasil%20Pengujian/MobileNetV1/No%20Flashlight/Dibungkus/IR64>



3. Varietas Beras Ketan

Selengkapnya dapat dilihat pada *link* berikut :

<https://github.com/Soedirman-Machine-Learning/rice-varieties-classification/tree/master/Screenshot%20Hasil%20Pengujian/MobileNetV1/No%20Flashlight/Dibungkus/Ketan>



BIODATA PENULIS



Biodata penulis berisi terkait dengan identitas penulis (nama, kontak email), riwayat akademis (pendidikan) penulis ditulis dari yang paling , skill, serta prestasi penulis.

A. Identitas

Nama	:	Vidi Fitriansyah Hidarlan
NIM	:	H1A016042
Tempat, tanggal lahir	:	Cirebon, 11 Februari 1997
Alamat	:	Jl. Gn. Guntur III/015 RT/RW. 04/08 Perumnas, Cirebon, 45142
No. Telp.	:	083823775820
Alamat e-mail	:	vidihidarlan@gmail.com

B. Riwayat Pendidikan Akademik

Periode	Jenjang	Institusi
2016 – 2020	S1	Teknik Elektro Universitas Jenderal Soedirman
2012 – 2015	SMA	SMAN 6 Cirebon
2009 – 2012	SMP	SMPN 5 Cirebon

C. Riwayat Pendidikan Non Formal (jika ada)

Tahun	Keahlian	Penyelenggara	Kota
-	-	-	-

D. Prestasi

Tahun	Tingkat	Prestasi
-	-	-

E. Keahlian (tuliskan secara diskriptif)

Memiliki minat di bidang perancangan dengan mikrokontroler. Mampu merancang sistem kontrol berbasiskan mikro kontroler atmega. Memiliki keahlian dalam bidang desain grafis dengan perangkat lunak *Corel Draw*. Terlibat secara aktif dalam kegiatan asisten Laboratorium Sistem Telekomunikasi dan Informasi sebagai asisten praktikum Dasar Teknik Elektro dan Algoritma dan Struktur Data, kegiatan asisten Laboratorium Kendali sebagai asisten praktikum Teknik Digital, dan kegiatan asisten Laboratorium Elektronika sebagai asisten praktikum Pengukuran Besaran Listrik, asisten praktikum Elektronika dan asisten praktikum Sistem Kendali.