MAKALAH SEMINAR HASIL TUGAS AKHIR

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

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

Vidi Fitriansyah Hidarlan[1], Imron Rosyadi, S.T., M.Sc.[2], Farida Asriani, S.Si., M.T.[3]

vidihidarlan@gmail.com[1], pak.imron@gmail.com[2], faridapamuji@gmail.com[2]

[1]Mahasiswa Pemakalah
[2]Dosen Pembimbing I
[3]Dosen Pembimbing II

[1,2,3]Jurusan Teknik Elektro, Fakultas Teknik, Universitas Jenderal Soedirman, Purwokerto

Abstrak— CNN memiliki berbagai macam arsitektur yang dapat digunakan untuk melakukan klasifikasi varietas beras berdasarkan citranya. Arsitektur CNN yang digunakan pada klasifikasi varietas beras ini adalah *VGG16Net* dan *MobileNet* dengan meggunakan metode ekstraksi fitur (*feature extraction*). Kedua arsitektur tersebut digunakan untuk melakukan pelatihan dan pengujian pada infrastruktur *Google Colaboratory*. 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%. 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 dan menggunakan cahaya *flashlight* diperoleh tingkat akurasi 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.

Abstract— *CNN has a variety of architectures that can be used to classify rice varieties based on their image. CNN architecture types used in the classification of rice varieties are VGG16Net and MobileNet using the feature extraction method. Both architectures are used to conduct training and testing on the Google Collaboratory infrastructure. 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%. Then testing on the MobileNetV1 architecture was continued using room light which the result of the level of accuracy is 71.11%. In addition, testing with the condition of rice wrapped in clear plastic and using a flashlight obtained an accuracy rate of 95.56% and without using a flashlight of 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*.

I. PENDAHULUAN

A. Latar Belakang

Beras adalah salah satu makanan pokok yang paling banyak dikonsumsi di Indonesia. Setiap varietas beras memiliki rasa, tekstur, manfaat, kandungan dan sifat unik tersendiri [1]. Klasifikasi varietas beras sangat penting karena setiap varietas 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.

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*".

B. 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*?

C. Batasan Masalah

Batasan masalah untuk laporan ini adalah sebagai berikut.

1. Metode *deep learning* yang digunakan untuk klasifikasi jenis 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.

D. Tujuan dan Manfaat

a. Tujuan

Tujuan pembuatan laporan kerja praktik 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.

b. Manfaat

Manfaat yang diharapkan dalam penyusunan laporan kerja praktik ini adalah sebagai berikut.

1. Mampu menerapkan ilmu yang didapat pada mata kuliah yang bersangkutan untuk menyelesaikan tugas akhir.

2. Memudahkan dalam membedakan jenis beras dengan citra berdasarkan bentuk, ukuran, dan warnanya.

II. TINJAUAN PUSTAKA

A. 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*.

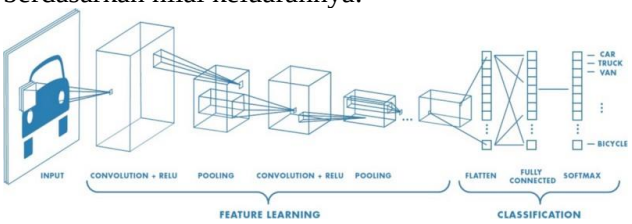
B. *Deep Learning*

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]. Banyaknya lapisan (*layer*) yang berkontribusi pada model data disebut kedalaman model.

C. *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 gambar 3 bisa dilihat alur dari proses CNN dalam mengolah citra masukan sampai mengklasifikasikan citra tersebut ke kategori tertentu berdasarkan nilai keluarannya.



Gambar- 1 Alur proses CNN dalam mengolah citra

a. 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].

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].

2. Lapisan Konvolusi (*Convolution Layer*)

Convolution Layer adalah inti dari dari CNN [6]. *Convolution Layer* menghasilkan citra baru yang menunjukkan fitur dari citra masukkan. *Convolutional layer* terdiri dari neuron yang tersusun sedemikian rupa sehingga membentuk sebuah *filter* dengan panjang dan tinggi (piksel). Alur pada *convolution layer* disajikan pada gambar 2.

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].

b. *Stride*

Stride adalah parameter yang menentukan berapa jumlah pergeseran *filter* [8]. Pada ilustrasi gambar 2, *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*

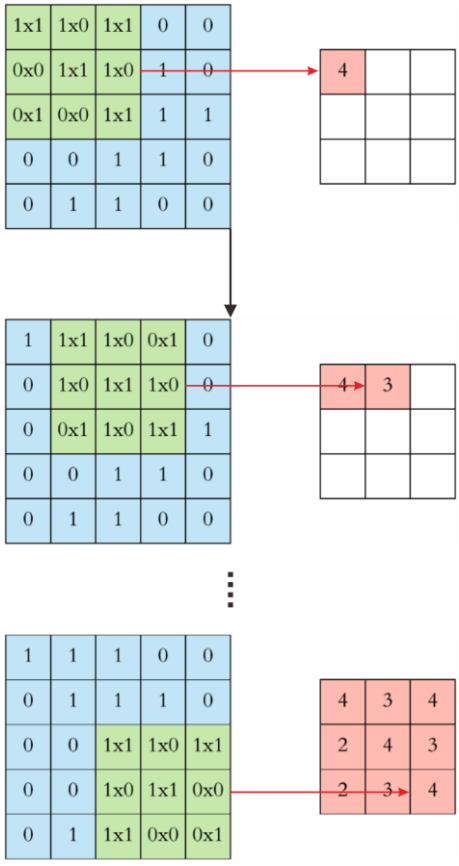
Padding atau *Zero Padding* adalah parameter yang menentukan jumlah piksel (berisi nilai 0) yang akan ditambahkan di setiap sisi dari masukkan [8].

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

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

dengan:

W = Panjang/Tinggi Input
F = Panjang/Tinggi Filter
P = *Zero Padding*
S = *Stride*



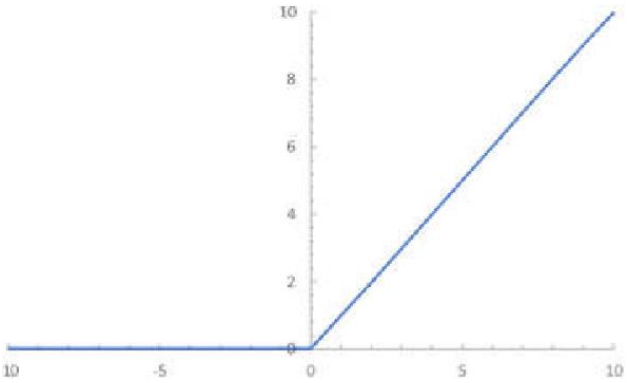
Gambar- 2 Alur pada *convolution layer*

3. Lapisan aktivasi (*Activation Layer*)

Activation Layer adalah *layer* dimana *feature map* dimasukkan ke dalam fungsi aktivasi [6]. 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 [9]. Berikut bentuk dari fungsi aktivasi ReLU disajikan pada gambar 3.



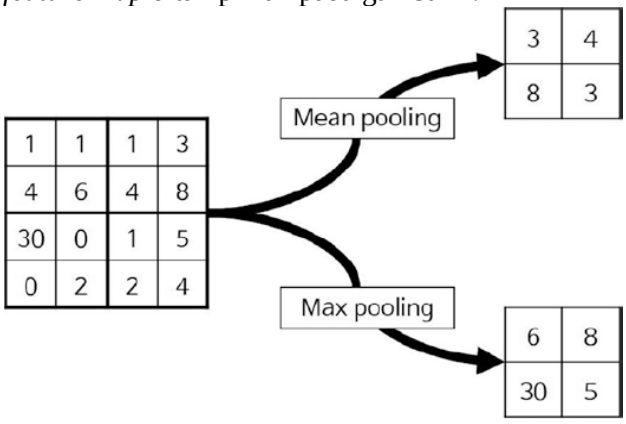
Gambar- 3 Fungsi aktivasi *Rectified Linear Unit* (ReLU)

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 [9].

4. *Pooling Layer*

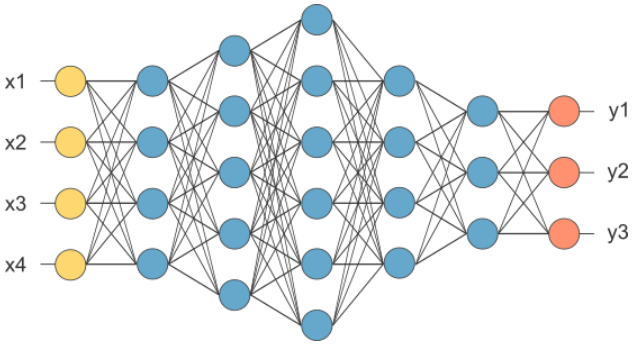
Pooling layer menerima masukkan dari *activation layer* kemudian mengurangi jumlah paramaternya. *Pooling* sendiri ada beberapa macam seperti *max pooling*, *mean pooling*, dan *sum pooling*. Berikut merupakan contoh proses *pooling* pada *feature map* ditampilkan pada gambar 4.



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

5. *Fully Connected Layer*

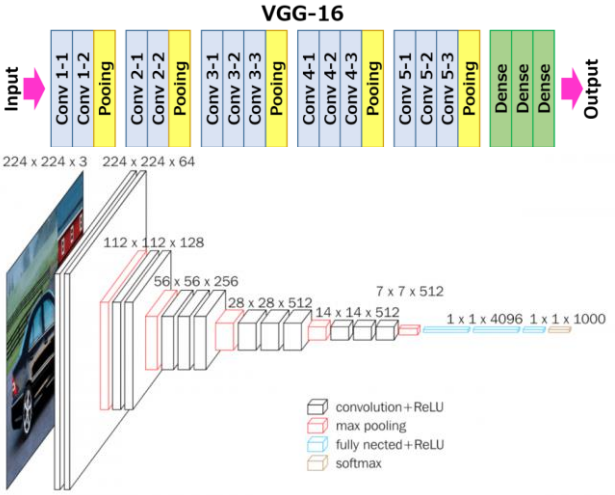
Setelah melewati proses-proses diatas, hasil dari *pooling layer* digunakan menjadi masukan untuk *Fully connected layer*. Gambar 5 ini merupakan contoh *Fully Connected Layer*.



Gambar- 5 Contoh *fully connected layer*

b. 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]. Berikut arsitektur VGG16 pada gambar 6.



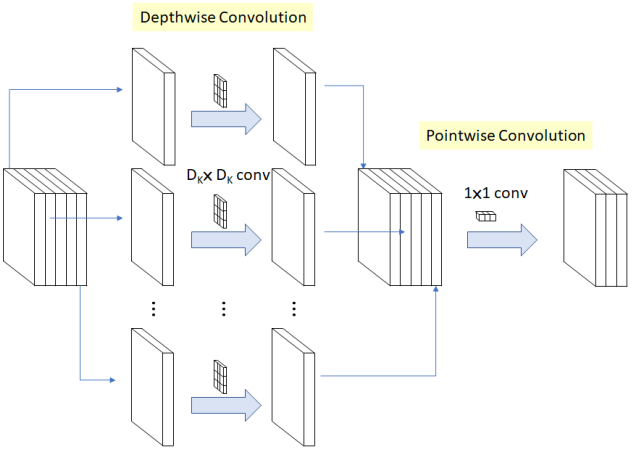
Gambar- 6 Arsitektur VGG16

c. Arsitektur *MobileNet*

MobileNets, merupakan salah satu arsitektur CNN yang dapat digunakan untuk mengatasi kebutuhan akan *computing resource* berlebih [11]. *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]. Arsitektur dari *dephwise separable convolution* dapat dilihat pada gambar 7.



Gambar- 7 *Depthwise separable convolution*

2. Struktur Jaringan

Arsitektur *MobileNet* didefinisikan dalam Tabel 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- 1 Arsitektur *MobileNet*

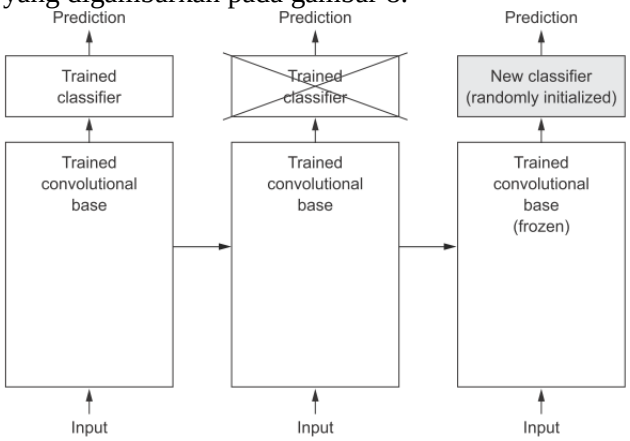
Type/Stride	*Filter Shape*	*Input Size*
Conv / s2	3 × 3 × 3 × 32	224 × 224 × 3
Conv dw / s1	3 × 3 × 32 dw	112 × 112 × 32
Conv / s1	1 × 1 × 32 × 64	112 × 112 × 32
Conv dw / s2	3 × 3 × 64 dw	112 × 112 × 64
Conv / s1	1 × 1 × 64 × 128	56 × 56 × 64
Conv dw / s1	3 × 3 × 128 dw	56 × 56 × 128
Conv / s1	1 × 1 × 128 × 128	56 × 56 × 128
Conv dw / s2	3 × 3 × 128 dw	56 × 56 × 128
Conv / s1	1 × 1 × 128 × 256	28 × 28 × 128
Conv dw / s1	3 × 3 × 256 dw	28 × 28 × 256
Conv / s1	1 × 1 × 256 × 256	28 × 28 × 256
Conv dw / s2	3 × 3 × 256 dw	28 × 28 × 256
Conv / s1	1 × 1 × 256 × 512	14 × 14 × 256
5x Conv dw / s1	3 × 3 × 512 dw	14 × 14 × 512
5x Conv / s1	1 × 1 × 512 × 512	14 × 14 × 512
Conv dw / s2	3 × 3 × 512 dw	14 × 14 × 512
Conv / s1	1 × 1 × 512 × 1024	7 × 7 × 512
Conv dw / s2	3 × 3 × 1024 dw	7 × 7 × 1024
Conv / s1	1 × 1 × 1024 × 1024	7 × 7 × 1024
Avg Pool / s1	Pool 7 × 7	7 × 7 × 1024
FC / s1	1024 × 1000	1 × 1 × 1024
Softmax / s1	Classifier	1 × 1 × 1000

d. *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*) [2].

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 8.



Gambar- 8 Pertukaran klasifikasi baru dengan mempertahankan basis konvolusi yang sama

D. *Google Colaboratory*

Google Colaboratory adalah layanan berbasis *cloud Google* yang mereplikasi *Jupyter Notebook* di-*cloud* [13].

Pada *Colaboratory* dapat membuat berbagai jenis sel dan menggunakannya untuk membuat buku catatan.

E. *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* [2].

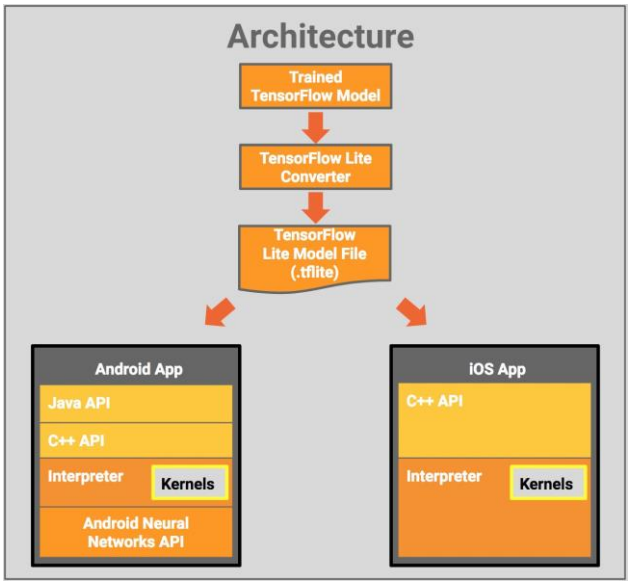
Keras didistribusikan di bawah lisensi MIT permisif, yang berarti dapat digunakan secara bebas dalam proyek komersial. *Framework* ini kompatibel dengan versi *Python* apa pun mulai 2,7 hingga 3,6.

Keras adalah *library* tingkat model, menyediakan blok-blok tingkat tinggi untuk mengembangkan model *deep learning* [2]. *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*.

Melalui *TensorFlow* (atau *Theano*, atau CNTK), *Keras* dapat berjalan dengan mulus di CPU dan GPU.

F. *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*. Arsitektur dari *TensorFlow Lite* dapat dilihat pada gambar 9 berikut.



Gambar- 9 Arsitektur *TensorFlow Lite*

G. *Android Studio*

Android Studio merupakan sebuah *Integrated Development Environment* (IDE) resmi yang digunakan untuk pengembangan aplikasi Android, berdasarkan IntelliJ IDEA [15].

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* atau *Kotlin*.

H. Beras

a. 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[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- 10 Jenis Beras IR 64

b. 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].

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.

Gambar- 11 Jenis Beras Basmathi

c. 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- 12 Jenis Beras Ketan Putih

III. METODE PENELITIAN

A. 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.

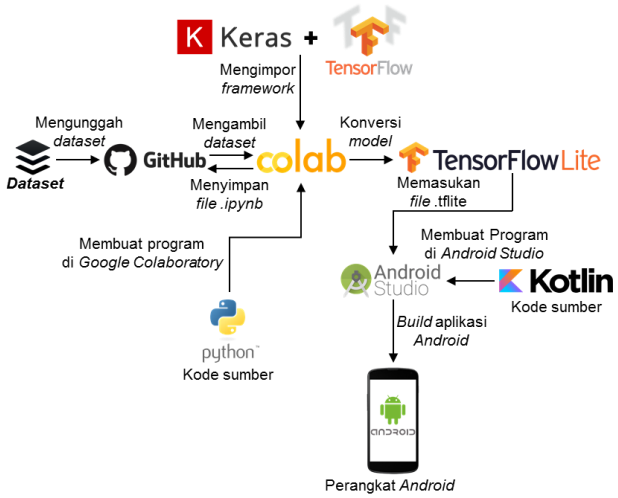
B. Alat dan Bahan

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

1. Perangkat keras:
   a. *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
   c. *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 Repositori *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*.

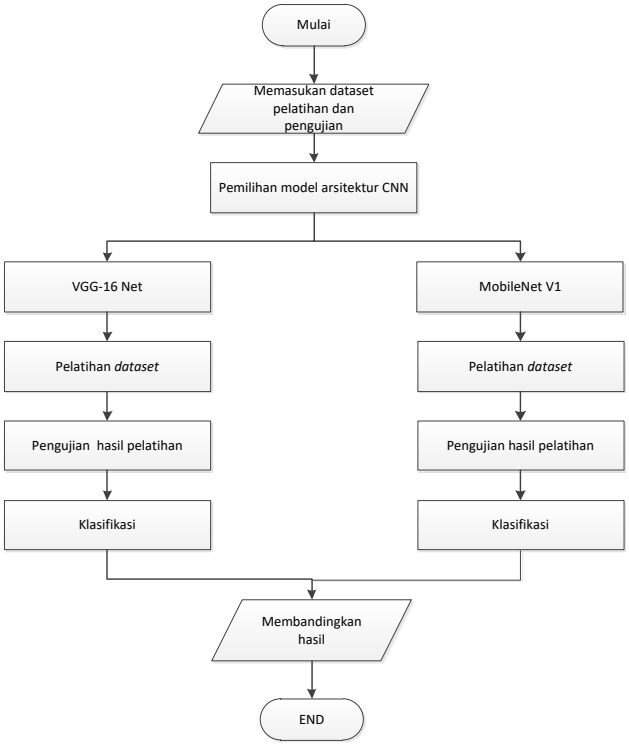
C. Tahapan Penelitian

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 13 berikut.



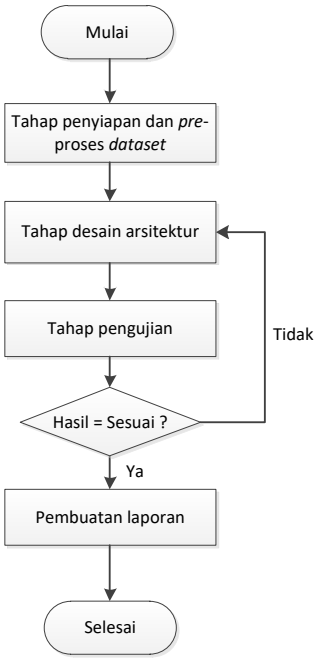
Gambar- 13 Desain arsitektur sistem

Sedangkan untuk diagram alir sistem yang dirancang pada penelitian ini dapat dilihat pada gambar 14 berikut.



Gambar- 14 Diagram alir sistem

Adapun diagram alir dari tahapan-tahapan penelitian yang dilakukan adalah seperti yang disajikan pada gambar 15 berikut.



Gambar- 15 Diagram alir penelitian

a. Tahap Penyiapan dan *Pre*-proses *Dataset*

Pada tahap ini penyiapan dan *pre*-proses *dataset* dilakukan dengan mengambil gambar dari obyek tiga macam jenis 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*.

b. 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.

c. 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 jenis 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 jenis beras tersebut dapat dipasang pada *smartphone Android*. Kemudian diuji dan dibandingkan juga hasil prediksi dari klasifikasinya pada aplikasi Android dengan menggunakan kedua arsitektur tersebut.

D. 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- 2 Jadwal penelitian

No.	Kegiatan	Bulan 1				Bulan 2				Bulan 3				Bulan 4			
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1.	Studi Pustaka																
2.	Penyiapan dan *pre*-proses *dataset*																
3.	Desain arsitektur																
4.	Pengujian dan evaluasi sistem																
5.	Pembuatan laporan																

IV. HASIL DAN PEMBAHASAN

A. Perancangan Sistem dan *Dataset* Penelitian

a. *Dataset*

Dataset yang digunakan pada klasifikasi varietas beras ini ada tiga macam, yaitu *dataset* pelatihan, *dataset* validasi, dan *dataset* pengujian. 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*.

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.

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.

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.

b. Pengambilan *Dataset*

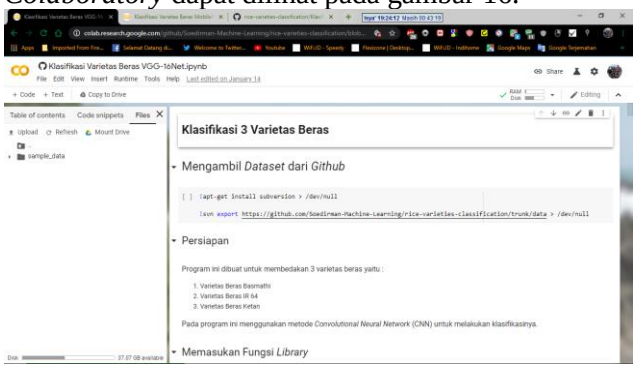
Seluruh *dataset* dari varietas beras yang digunakan pada klasifikasi varietas beras diambil menggunakan sebuah kamera *smartphone Android*. Pengambilan *dataset* dilakukan dengan mengambil gambar dari varietas beras yang akan dilatih dan diuji pada alas berwarna hitam dengan jarak antara obyek (beras) dengan kamera antara 10 cm sampai dengan 15 cm.

c. Penyimpanan *Dataset*

Seluruh *dataset* pelatihan, *dataset* validasi, dan *dataset* pengujian diunggah dan disimpan pada layanan Repositori *Web Development* pada *Platform Github*.

d. Perancangan Program pada *Google Colaboratory*

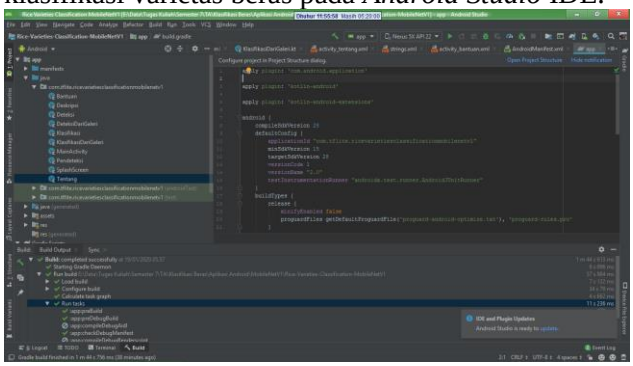
Perancangan program pada infrastuktur *Google Colaboratory* dilakukan dengan membuat beberapa sel/baris *text* untuk memberi judul, keterangan dan penjelasan program di setiap barisnya serta membuat sel/baris *code* untuk mengimpor fungsi *library* dan membuat program di setiap baris yang dibuat. Program tersebut dibuat dengan *Framework Keras* dengan *backend TensorFlow*. Berikut tampilan program klasifikasi varietas beras pada *Google Colaboratory* dapat dilihat pada gambar 16.



Gambar- 16 Tampilan program klasifikasi varietas beras pada *Google Colaboratory*

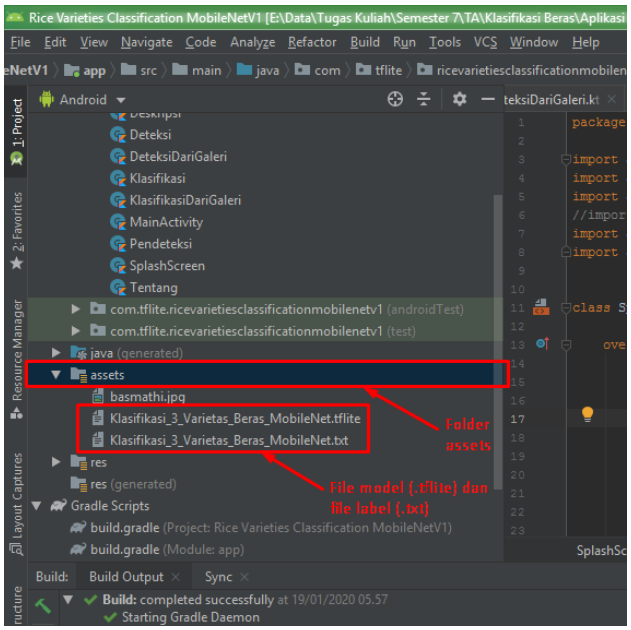
e. Perancangan Program pada *Android Studio*

Program untuk perangkat Android yang dibuat dalam Bahasa Kotlin 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, serta membuat *class* lainnya untuk mendukung dalam membuat antarmuka aplikasi *Android*. Gambar 17 berikut merupakan tampilan dari *file project* klasifikasi varietas beras pada *Android Studio* IDE.



Gambar- 17 Tampilan *project* aplikasi klasifikasi varietas beras di *Android Studio*

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- 18 *File* model (.tflite) beserta labelnya (.txt) pada *folder assets*

B. 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.

a. 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*.

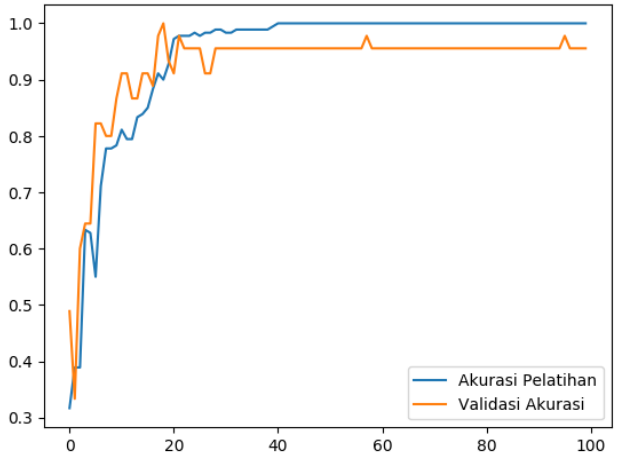
b. Arsitektur CNN yang digunakan

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*.

c. Hasil Pelatihan dan Pengujian pada *Google Colaboratory*

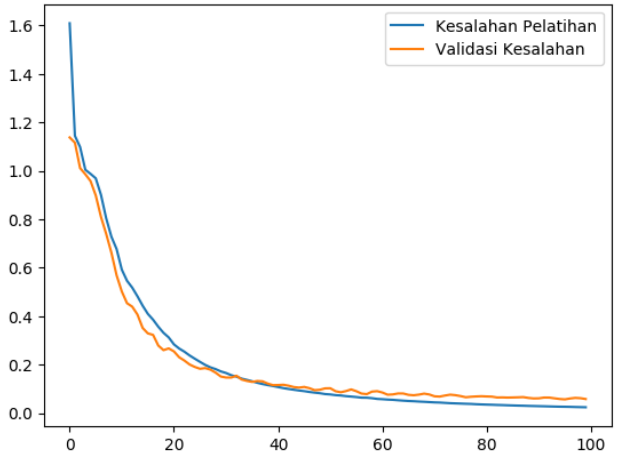
1. Hasil pelatihan dan pengujian pada arsitektur *VGG-16Net* dengan kualitas gambar baik

Berikut kurva akurasi hasil pelatihan (*training*) *dataset* pelatihan dan validasi yang dapat dilihat pada gambar 19.



Gambar- 19 Kurva tingkat akurasi hasil pelatihan dan validasi pada *VGG-16Net*

Berdasarkan 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. dapat dilihat juga bahwa kurva akurasi pelatihan hampir sama peningkatannya dengan kurva validasi akurasi yang menunjukkan hasil pelatihan dengan keadaan *goodfit*.

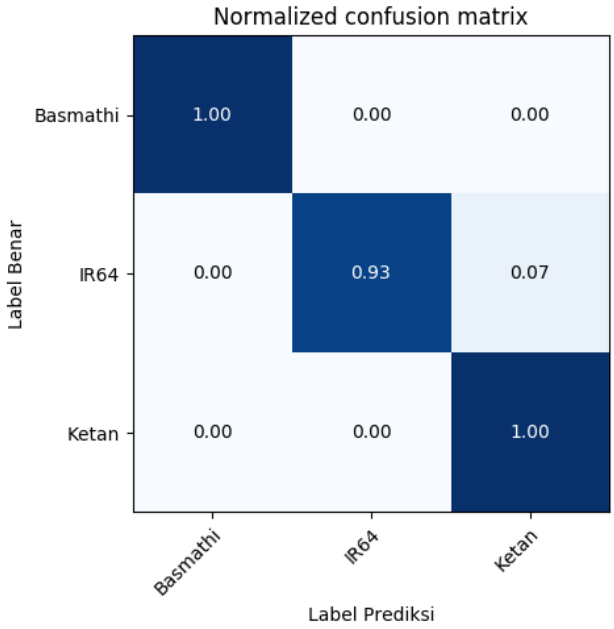


Gambar- 20 Kurva tingkat kesalahan pelatihan dan validasi pada *VGG-16Net*

Kurva kesalahan (*error*) yang dihasilkan selama pelatihan ditampilkan pada gambar 20.

Dari kurva tersebut diperoleh pada *epoch* yang terakhir nilai kesalahan pelatihan mencapai 0.024 dan nilai kesalahan validasinya mencapai 0.058. Karena nilainya semakin mendekati 0 dan kurva kesalahan pelatihan hampir sama penurunannya dengan kurva validasi kesalahan sehingga dapat disimpulkan hasil pelatihannya *goodfit*.

Kemudian diperoleh *confusion matrix* yang digunakan untuk mengetahui tingkat keberhasilan dan kegagalan pada pengujian yang dilakukan.



Gambar- 21 *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, hasil prediksi Beras IR64 bernilai 0.93 pada label benar IR64, dan hasil prediksi Beras Ketan dengan nilai 1.00 pada label benar Ketan. Nilai pada *confusion matrix* tersebut merupakan persentase hasil prediksi dari masing-masing *dataset* varietas beras yang berjumlah 15.

Kemudian dihitung persentase tingkat akurasi dan tingkat kesalahannya seperti berikut.

$$\%akurasi = \frac{jumlahprediksibenar}{jumlahdatapengujian} x100\%$$

$$\%akurasi = \frac{44}{45} x100\% = 97{,}78\%$$

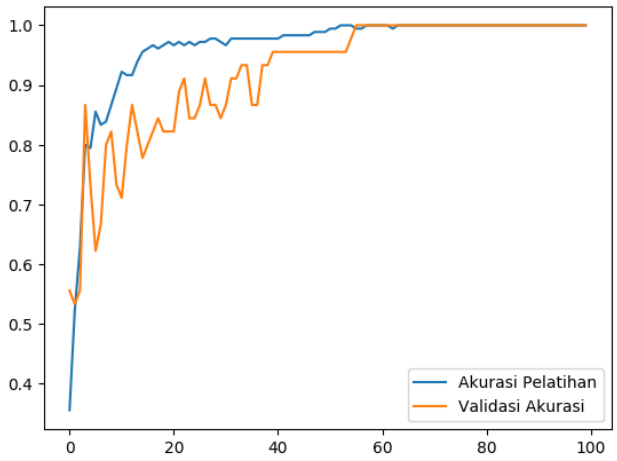
$$\%kesalahan = \frac{jumlahprediksisalah}{jumlahdatapengujian} x100\%$$

$$\%kesalahan = \frac{1}{45} x100\% = 2{,}22\%$$

Berdasarkan persentase akurasi dan kesalahan tersebut dapat disimpulkan bahwa arsitektur *VGG-16Net* dapat melakukan klasifikasi ketiga varietas beras dengan baik.

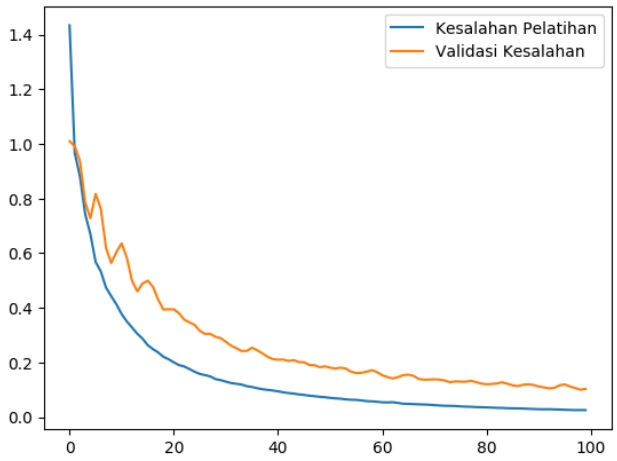
2. Hasil pelatihan dan pengujian pada arsitektur *VGG-16Net* dengan kualitas gambar buruk

Kurva akurasi hasil pelatihan dan pengujiannya dapat dilihat pada gambar 22.



Gambar- 22 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.

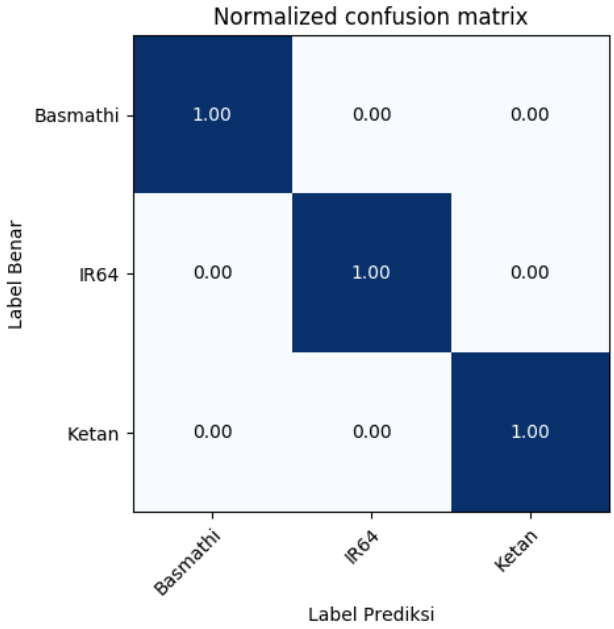


Gambar- 23 Kurva kesalahan pelatihan dan validasi pada *VGG-16Net* pada gambar buruk

Kurva kesalahan hasil pelatihan dan pengujiannya ditampilkan pada gambar 23.

Dari kurva kesalahan tersebut diperoleh nilai kesalahan pelatihan sebesar 0.0235 dan kesalahan validasinya sebesar 0.1037 yang sedikit mengalami *underfitting* karena nilai kesalahan validasi yang sedikit di atas kesalahan pelatihan.

Selanjutnya diperoleh hasil prediksi pengujiannya dalam *confusion matrix* pada gambar 24. Berdasarkan gambar tersebut diperoleh bahwa hasilnya sedikit lebih baik dibandingkan *dataset* kualitas gambar yang baik. Hal tersebut kemungkinan terjadi karena *dataset* pengujian yang diambil masih cukup mirip dibandingkan *dataset* pengujian pada kualitas gambar baik.



Gambar- 24 *Confusion matrix* hasil pengujian pada *VGG-16Net* dengan gambar buruk

Persentase akurasi dan kesalahan yang diperoleh adalah sebagai berikut.

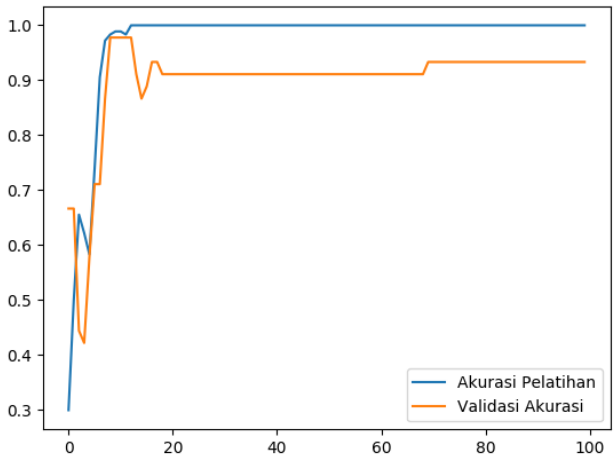
$$\%akurasi = \frac{45}{45} x100\% = 100\%$$

$$\%kesalahan = \frac{0}{45} x100\% = 0\%$$

Dari persentase tersebut dapat disimpulkan bahwa arsitektur *VGG-16Net* masih dapat melakukan klasifikasi dengan baik.

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

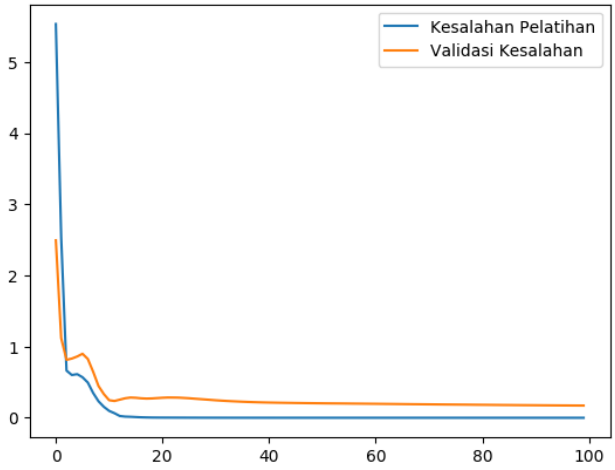
Kurva akurasi hasil pelatihan dan pengujiannya dapat dilihat pada gambar 25.



Gambar- 25 Kurva akurasi hasil pelatihan dan validasi pada *MobileNetV1*

Berdasarkan kurva tersebut nilai akurasi pelatihan yang diperoleh mencapai 1.0 dan nilai akurasi validasi mencapai sekitar 0.9333. Dapat disimpulkan bahwa arsitektur *MobileNetV1* cukup baik dalam melakukan klasifikasi namun sedikit lebih rendah daripada *VGG-16Net*.

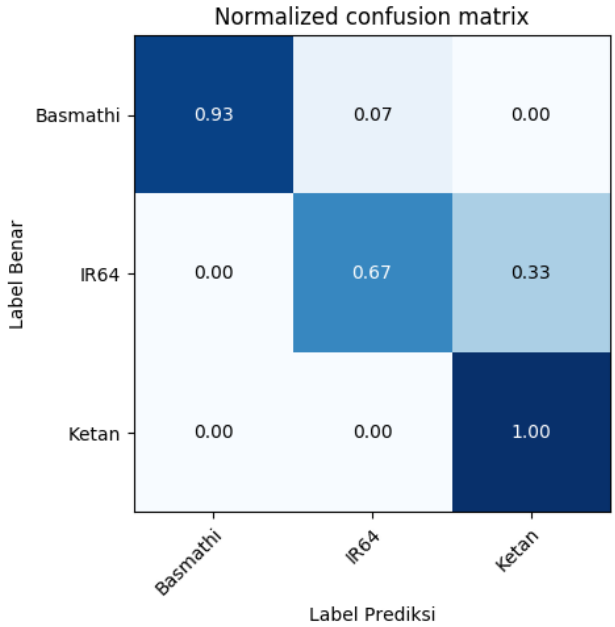
Kemudian kurva kesalahan yang dihasilkan selama pelatihan ditampilkan pada gambar 26.



Gambar- 26 Kurva tingkat kesalahan pelatihan dan validasi pada *MobileNetV1*

Dari kurva hasil pelatihan tersebut diperoleh nilai kesalahan pelatihan 0.0002 dan nilai validasi kesalahan sedikit di atas kesalahan pelatihan yaitu sebesar 0.1722 sehingga dapat disimpulkan hanya sedikit terjadi *underfitting*.

Untuk hasil pengujiannya ditampilkan dengan *confusion matrix* pada gambar 27.



Gambar- 27 *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.

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

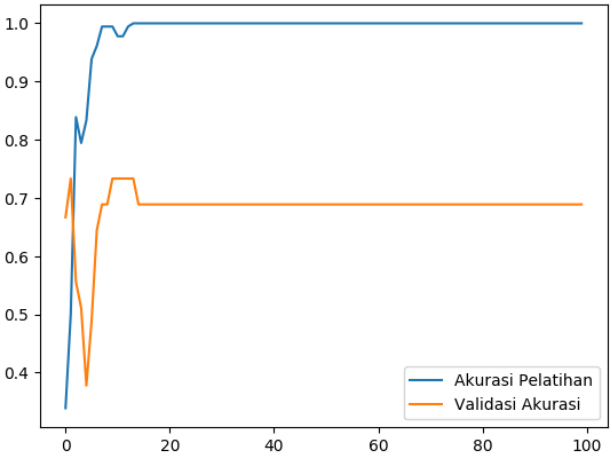
$$\%akurasi = \frac{39}{45} x100\% = 86{,}67\%$$

$$\%kesalahan = \frac{3}{45} x100\% = 13{,}33\%$$

Berdasarkan persentase tersebut disimpulkan bahwa arsitektur *MobileNetV1* mampu melakukan klasifikasi varietas beras dengan baik. Namun masih lebih rendah daripada arsitektur *VGG-16Net*.

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

Kurva akurasi hasil pelatihan dan pengujiannya dapat dilihat pada gambar 28.

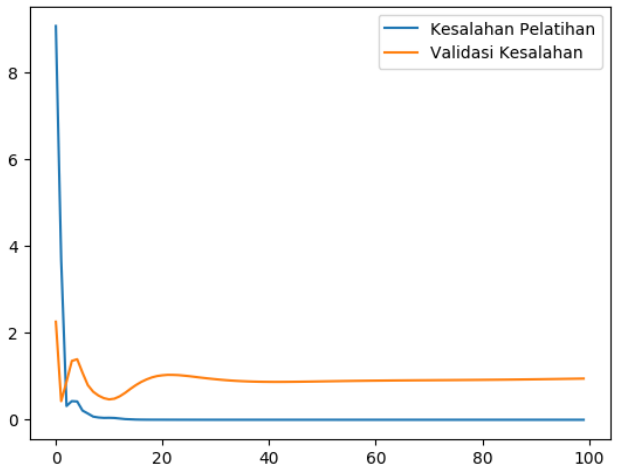


Gambar- 28 Kurva akurasi pelatihan dan validasi pada *MobileNetV1* pada gambar buruk

Dari hasil pelatihan tersebut diperoleh nilai akurasi pelatihan mencapai 1.0 tetapi nilai akurasi validasinya jauh di bawah akurasi pelatihannya yaitu hanya mencapai 0.6889 sehingga dipastikan terjadinya *underfitting*.

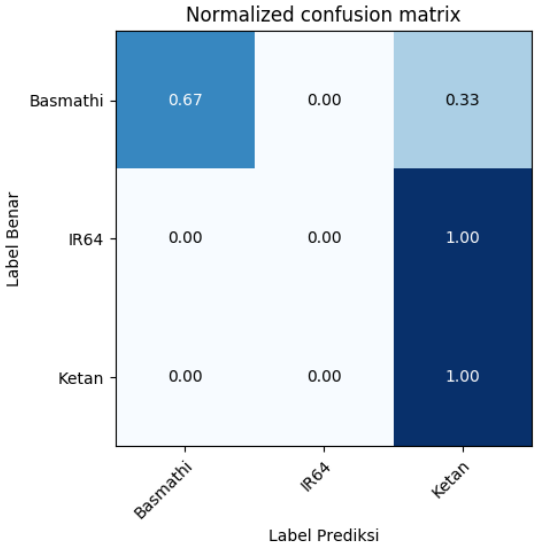
Kemudian kurva kesalahan yang dihasilkan selama pelatihan ditampilkan pada gambar 29.

Dari hasil pelatihan tersebut diperoleh nilai kesalahan sebesar 0.0001 dan nilai validasi kesalahan yang cukup besar yaitu sebesar 0.9485 sehingga dapat disimpulkan pada kondisi tersebut arsitektur CNN tidak dapat melakukan klasifikasi dengan baik.



Gambar- 29 Kurva kesalahan pelatihan dan validasi pada *MobileNetV1* pada gambar buruk

Untuk hasil pengujiannya ditampilkan dengan *confusion matrix* pada gambar 30.



Gambar- 30 *Confusion matrix* hasil pengujian pada *MobileNetV1* dengan gambar buruk

Berdasarkan hasil pengujian diperoleh hasil prediksi Basmathi yang benar 67%, IR64 tidak ada yang benar melainkan memprediksi Ketan 100% dan prediksi pada Ketan yang benar 100%.

Persentase tingkat keberhasilan dan kesalahannya diperoleh sebagai berikut.

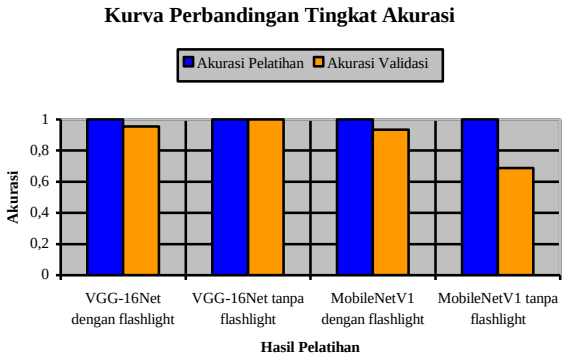
$$\%akurasi = \frac{25}{45} x100\% = 55{,}56\%$$

$$\%kesalahan = \frac{20}{45} x100\% = 44{,}44\%$$

Dari persentase tersebut dapat disimpulkan bahwa arsitektur *MobileNetV1* pada kondisi *dataset* tersebut tidak mampu melakukan klasifikasi dengan baik.

d. Kurva Perbandingan Hasil Pelatihan

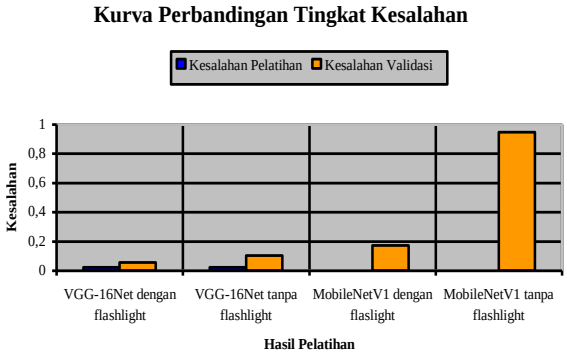
Kurva perbandingan tingkat akurasi hasil pelatihannya disajikan pada gambar 31.



Gambar- 31 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.

Untuk kurva perbandingan tingkat kesalahan yang dihasilkan ditampilkan pada gambar 32.



Gambar- 32 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.

C. Pengujian pada Perangkat *Android*

Pengujian ini dilakukan menggunakan model hasil pelatihan dari *Google Colaboratory* pada masing-masing arsitektur yang digunakan. Hasil

pengujian ini mengambil hasil prediksi dari persentase probabilitas yang tertinggi.

a. 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 menyebarkan beras pada alas berwarna hitam di ruangan yang cukup redup agar *flashlight* pada perangkat *Android* aktif secara otomatis. Jarak antara obyek beras dengan kamera ditetapkan sekitar 10 cm dalam pengujian agar hasil pengujiannya konsisten.

Tabel- 3 Hasil pengujian arsitektur *VGG-16Net* dengan *flashlight*

Pengu-jian ke-	Waktu (ms)	Intensitas Cahaya (*lux*)	Varietas Beras	Hasil Prediksi (Tingkat Probabilitas)	Benar/ Salah
1	6419	1070	Basmathi	Basmathi (91,41%)	Benar
2	4976	1070	Basmathi	IR-64 (94,25%)	Salah
3	4827	1070	Basmathi	Basmathi (87,72%)	Benar
4	4924	1070	Basmathi	Basmathi (95,3%)	Benar
5	4930	1070	Basmathi	Basmathi (93,63%)	Benar
6	5203	1070	Basmathi	Basmathi (89,28%)	Benar
7	4930	1070	Basmathi	Basmathi (99,72%)	Benar
8	5107	1070	Basmathi	Basmathi (96,27%)	Benar
9	4930	1070	Basmathi	Basmathi (77,12%)	Benar
10	4974	1070	Basmathi	Basmathi (99,95%)	Benar
11	4879	1070	Basmathi	Basmathi (78,28%)	Benar
12	5139	1070	Basmathi	IR-64 (93,75%)	Salah
13	4993	1070	Basmathi	Basmathi (94,33%)	Benar
14	4998	1070	Basmathi	Basmathi (96,54%)	Benar
15	4936	1070	Basmathi	Basmathi (89,5%)	Benar
16	6209	1070	IR-64	IR-64 (94,58%)	Benar
17	5058	1070	IR-64	IR-64 (99,74%)	Benar
18	5020	1070	IR-64	IR-64 (98,22%)	Benar
19	5029	1070	IR-64	IR-64 (99,59%)	Benar
20	5275	1070	IR-64	IR-64 (99,75%)	Benar
21	5186	1070	IR-64	IR-64 (96,18%)	Benar
22	5046	1070	IR-64	IR-64 (99,91%)	Benar
23	5428	1070	IR-64	IR-64 (70,51%)	Benar
24	4892	1070	IR-64	IR-64 (99,63%)	Benar
25	5274	1070	IR-64	IR-64 (97,46%)	Benar
26	4903	1070	IR-64	IR-64 (97,72%)	Benar
27	5022	1070	IR-64	IR-64 (99,92%)	Benar
28	4911	1070	IR-64	IR-64 (84,25%)	Benar
29	5056	1070	IR-64	IR-64 (99,37%)	Benar
30	5172	1070	IR-64	IR-64 (99,94%)	Benar
31	6116	1070	Ketan	Ketan (97,24%)	Benar
32	5009	1070	Ketan	IR-64 (98,83%)	Salah
33	4894	1070	Ketan	Ketan (57,51%)	Benar
34	4936	1070	Ketan	IR-64 (73,96%)	Salah
35	5004	1070	Ketan	IR-64 (75,4%)	Salah
36	4913	1070	Ketan	Ketan (69,02%)	Benar
37	4941	1070	Ketan	IR-64 (99,92%)	Salah
38	5010	1070	Ketan	Basmathi (45,14%)	Salah
39	5029	1070	Ketan	Ketan (52,73%)	Benar
40	5234	1070	Ketan	IR-64 (90,8%)	Salah
41	4980	1070	Ketan	Ketan (94,4%)	Benar
42	4887	1070	Ketan	IR-64 (97,75%)	Salah
43	4979	1070	Ketan	IR-64 (93,51%)	Salah
44	5037	1070	Ketan	Ketan (58,43%)	Benar
45	4994	1070	Ketan	IR-64 (90,82%)	Salah

Dari tabel pengujian tersebut dapat diketahui bahwa hasil prediksi untuk Beras Basmathi dan IR-64 cukup akurat dengan tingkat probabilitas yang cukup stabildi angka 90%, tetapi tidak untuk Beras Ketan dengan kesalahan 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. intensitas cahaya yang dihasilkan oleh *flashlight* pada jarak 10 cm sekitar 1070 *lux*. Dari tabel pengujian tersebut dapat diketahui persentase akurasi dan kesalahannya.

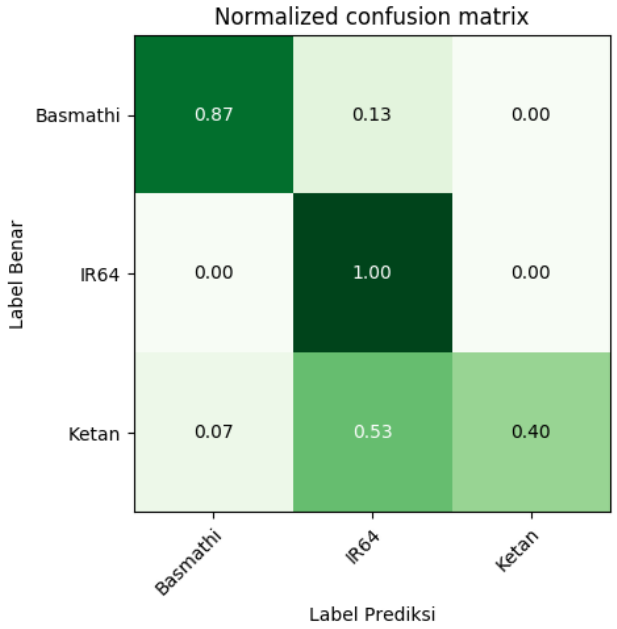
$$\%akurasi = \frac{jumlahprediksibenar}{jumlahdatapengujian} x100\%$$

$$\%akurasi = \frac{34}{45} x100\% = 75{,}56\%$$

$$\%kesalahan = \frac{jumlah prediksi salah}{jumlah data pengujian} x100\%$$

$$\%kesalahan = \frac{11}{45} x100\% = 24{,}44\%$$

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 33.



Gambar- 33 *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. Hal itu disebabkan model hasil pelatihan *VGG-16Net* tidak bagus dikonversi ke dalam format *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 untuk arsitektur *VGG-16Net* tidak dilanjutkan ke pengujian selanjutnya karena sudah dapat dipastikan hasilnya tidak akurat.

b. Hasil Pengujian pada Arsitektur *MobileNetV1* dengan *Flashlight*

Pengujian ini dilakukan sama dengan pengujian sebelumnya hanya menggunakan arsitektur yang berbeda.

Tabel- 4 Hasil pengujian arsitektur *MobileNetV1* dengan *flashlight*

Pengu-jian ke-	Waktu (ms)	Intensitas Cahaya (*lux*)	Varietas Beras	Hasil Prediksi (Tingkat Probabilitas)	Benar/ Salah
1	1344	1070	Basmathi	Basmathi (99,29%)	Benar
2	983	1070	Basmathi	Basmathi (92,65%)	Benar
3	957	1070	Basmathi	Basmathi (92,22%)	Benar
4	954	1070	Basmathi	Basmathi (76%)	Benar
5	976	1070	Basmathi	Basmathi (99,65%)	Benar
6	997	1070	Basmathi	Basmathi (57,12%)	Benar
7	972	1070	Basmathi	Basmathi (98,21%)	Benar
8	963	1070	Basmathi	Basmathi (87,28%)	Benar
9	1000	1070	Basmathi	Basmathi (90,92%)	Benar
10	1043	1070	Basmathi	Basmathi (83,49%)	Benar
11	994	1070	Basmathi	Basmathi (59,38%)	Benar
12	952	1070	Basmathi	Basmathi (86,66%)	Benar
13	1004	1070	Basmathi	Basmathi (92,53%)	Benar
14	990	1070	Basmathi	Basmathi (99,18%)	Benar
15	1055	1070	Basmathi	Basmathi (97,22%)	Benar
16	1380	1070	IR-64	IR-64 (53,55%)	Benar
17	975	1070	IR-64	IR-64 (97,73%)	Benar
18	998	1070	IR-64	Ketan (67,19%)	Salah
19	1011	1070	IR-64	IR-64 (94,69%)	Benar
20	996	1070	IR-64	IR-64 (94,92%)	Benar
21	957	1070	IR-64	IR-64 (73,82%)	Benar
22	993	1070	IR-64	IR-64 (98,52%)	Benar
23	975	1070	IR-64	IR-64 (99,47%)	Benar
24	1079	1070	IR-64	IR-64 (92,3%)	Benar
25	1080	1070	IR-64	IR-64 (98,78%)	Benar
26	948	1070	IR-64	IR-64 (85,53%)	Benar
27	974	1070	IR-64	IR-64 (99,75%)	Benar
28	966	1070	IR-64	IR-64 (95,28%)	Benar
29	999	1070	IR-64	IR-64 (91,99%)	Benar
30	1056	1070	IR-64	Ketan (68,59%)	Salah
31	1391	1070	Ketan	Ketan (98,84%)	Benar
32	955	1070	Ketan	Ketan (72,66%)	Benar

33	969	1070	Ketan	Ketan (99,49%)	Benar
34	978	1070	Ketan	Ketan (95,18%)	Benar
35	1005	1070	Ketan	Ketan (86,3%)	Benar
36	962	1070	Ketan	Ketan (99,14%)	Benar
37	960	1070	Ketan	Ketan (79,32%)	Benar
38	1004	1070	Ketan	Ketan (93,6%)	Benar
39	1071	1070	Ketan	Ketan (99,68%)	Benar
40	1066	1070	Ketan	Ketan (99,92%)	Benar
41	988	1070	Ketan	Ketan (98,58%)	Benar
42	956	1070	Ketan	Ketan (83,58%)	Benar
43	1003	1070	Ketan	Ketan (70,94%)	Benar
44	993	1070	Ketan	Ketan (99,19%)	Benar
45	1056	1070	Ketan	Ketan (99,61%)	Benar

Dari tabel pengujian tersebut terlihat bahwa hasil prediksi untuk ketiga varietas beras sangat akurat dan hanya terdapat 2 kesalahan prediksi pada varietas Beras IR-64. Tingkat probabilitasnya cukup baik di atas 90% dan hanya beberapa saja yang di bawah 70%. Respon deteksi yang diperoleh sangat cepat hanya sekitar 1 detik. Hasil tersebut sedikit lebih baik dibandingkan pengujian pada *Google Colaboratory* karena kemungkinan jarak antara obyek beras dengan kamera relatif tepat. Selanjutnya dihitung tingkat akurasi dan kesalahannya.

$$\%akurasi = \frac{jumlahprediksibenar}{jumlahdatapengujian} x100\%$$

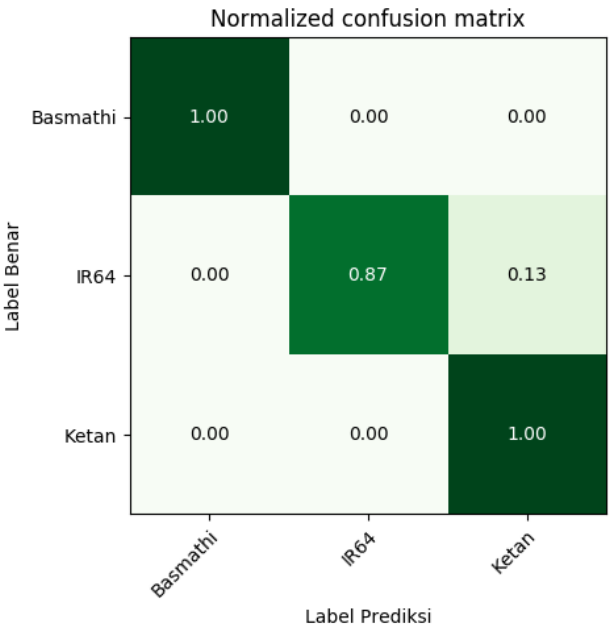
$$\%akurasi = \frac{43}{45} x100\% = 95{,}56\%$$

$$\%kesalahan = \frac{jumlahprediksisalah}{jumlahdatapengujian} x100\%$$

$$\%kesalahan = \frac{2}{45} x100\% = 4{,}44\%$$

Dari persentase tersebut diperoleh nilai persentase akurasi yang tinggi dan persentase kesalahan yang rendah. Selanjutnya ditampilkan hasilnya dalam *confusion matrix* pada gambar 34.

Berdasarkan *confusion matrix* tersebut dapat disimpulkan bahwa arsitektur *MobileNetV1* pada perangkat *Android* akurat untuk klasifikasi ketiga varietas beras tersebut dengan menggunakan *flashlight*.



Gambar- 34 *Confusion matrix* pengujian arsitektur *MobileNetV1* pada *Android* dengan *flashlight*

c. Hasil Pengujian pada Arsitektur *MobileNetV1* tanpa *Flashlight*

Pada pengujian ketiga dilakukan menggunakan model yang sama seperti pengujian kedua, yang membedakan hanyala kondisi pencahayaannya. Berikut hasil pengujiannya diperoleh pada tabel 5.

Tabel- 5 Hasil pengujian arsitektur *MobileNetV1* tanpa *flashlight*

Pengu-jian ke-	Waktu (ms)	Intensitas Cahaya (*lux*)	Varietas Beras	Hasil Prediksi (Tingkat Probabilitas)	Benar/ Salah
1	1416	2110	Basmathi	Basmathi (99,38%)	Benar
2	1056	2110	Basmathi	Basmathi (50,17%)	Benar
3	1012	2110	Basmathi	Basmathi (98,76%)	Benar
4	1033	2110	Basmathi	Basmathi (67,46%)	Benar
5	1047	2110	Basmathi	Basmathi (57,32%)	Benar
6	1224	2110	Basmathi	Basmathi (99,74%)	Benar
7	1051	2110	Basmathi	Basmathi (95,14%)	Benar
8	1049	2110	Basmathi	Basmathi (78,37%)	Benar
9	967	2110	Basmathi	Basmathi (56,02%)	Benar
10	1087	2110	Basmathi	Basmathi (78,77%)	Benar
11	1039	2110	Basmathi	Basmathi (99,38%)	Benar
12	1090	2110	Basmathi	Basmathi (97,83%)	Benar
13	1023	2110	Basmathi	Basmathi (83,2%)	Benar
14	980	2110	Basmathi	Basmathi (74,34%)	Benar

15	1054	2110	Basmathi	Ketan (52,96%)	Salah
16	1935	2110	IR-64	IR-64 (67,29%)	Benar
17	1169	2110	IR-64	IR-64 (99,74%)	Benar
18	1023	2110	IR-64	IR-64 (96,12%)	Benar
19	1066	2110	IR-64	IR-64 (99,41%)	Benar
20	1149	2110	IR-64	IR-64 (51,69%)	Benar
21	1085	2110	IR-64	IR-64 (45,23%)	Benar
22	1043	2110	IR-64	IR-64 (99,26%)	Benar
23	1035	2110	IR-64	IR-64 (99,42%)	Benar
24	1053	2110	IR-64	IR-64 (96,69%)	Benar
25	1003	2110	IR-64	IR-64 (51,33%)	Benar
26	1049	2110	IR-64	IR-64 (69,87%)	Benar
27	1032	2110	IR-64	IR-64 (99,52%)	Benar
28	1012	2110	IR-64	IR-64 (99,89%)	Benar
29	1047	2110	IR-64	IR-64 (99,77%)	Benar
30	1088	2110	IR-64	IR-64 (79,67%)	Benar
31	2798	2110	Ketan	IR-64 (76,91%)	Salah
32	1032	2110	Ketan	IR-64 (98,99%)	Salah
33	1084	2110	Ketan	IR-64 (87,55%)	Salah
34	1049	2110	Ketan	IR-64 (88,38%)	Salah
35	1246	2110	Ketan	Ketan (92,25%)	Benar
36	1535	2110	Ketan	IR-64 (78,12%)	Salah
37	1084	2110	Ketan	IR-64 (96,22%)	Salah
38	1114	2110	Ketan	IR-64 (95,13%)	Salah
39	1068	2110	Ketan	IR-64 (85,5%)	Salah
40	1027	2110	Ketan	Ketan (71,49%)	Benar
41	1072	2110	Ketan	IR-64 (87,36%)	Salah
42	1103	2110	Ketan	IR-64 (99,4%)	Salah
43	1021	2110	Ketan	IR-64 (93,39%)	Salah
44	1031	2110	Ketan	IR-64 (72,64%)	Salah
45	1036	2110	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%.

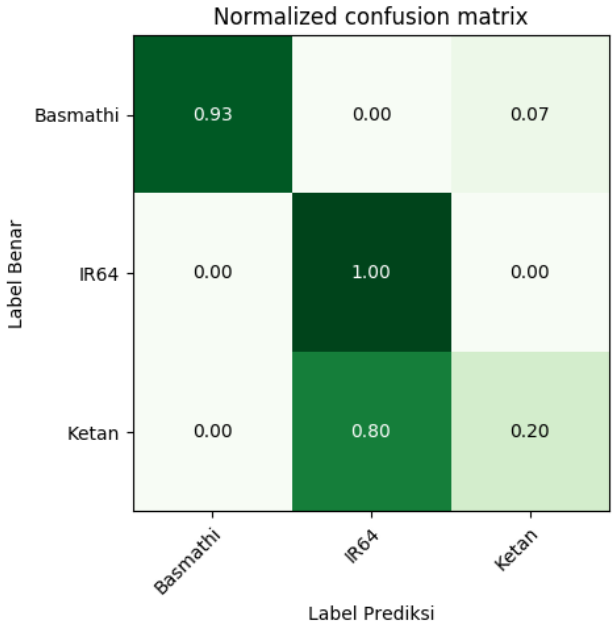
Persentase tingkat akurasi dan kesalahan dari pengujian tersebut adalah sebagai berikut.

$$\%akurasi = \frac{32}{45} x100\% = 71{,}11\%$$

$$\%kesalahan = \frac{13}{45} x100\% = 28{,}89\%$$

Berdasarkan persentase tersebut diperoleh tingkat akurasi yang lebih rendah daripada pengujian sebelumnya. Tingkat akurasi tiap varietas berasnya ditampilkan dalam *confusion matrix* pada gambar 35.

Berdasarkan *confusion matrix* tersebut disimpulkan bahwa tingkat akurasi hasil prediksi Beras Ketan sangat rendah yang disebabkan intensitas cahaya yang mengenai obyek beras tidak merata sehingga terdapat bayangan yang mempengaruhi hasil tersebut.



Gambar- 35 *Confusion matrix* pengujian arsitektur *MobileNetV1* pada *Android* tanpa *flashlight*

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

Pengujian ini masih sama menggunakan model *TensorFlow Lite* arsitektur *MobileNetV1* dari pengujian sebelumnya. Berikut hasil pengujiannya pada tabel 6.

Tabel- 6 Hasil pengujian arsitektur *MobileNetV1* menggunakan *flashlight* pada beras dibungkus plastik

Pengu-jian ke-	Waktu (ms)	Intensitas Cahaya (*lux*)	Varietas Beras	Hasil Prediksi (Tingkat Probabilitas)	Benar/ Salah
1	972	1070	Basmathi	Basmathi (85,34%)	Benar
2	975	1070	Basmathi	Basmathi (78,49%)	Benar

3	1013	1070	Basmathi	Basmathi (65,85%)	Benar
4	990	1070	Basmathi	Basmathi (52,4%)	Benar
5	970	1070	Basmathi	Basmathi (87,69%)	Benar
6	948	1070	Basmathi	Basmathi (76,94%)	Benar
7	975	1070	Basmathi	IR-64 (79,89%)	Salah
8	1000	1070	Basmathi	IR-64 (59,82%)	Salah
9	972	1070	Basmathi	Basmathi (67,02%)	Benar
10	968	1070	Basmathi	Basmathi (95,91%)	Benar
11	997	1070	Basmathi	Basmathi (84,43%)	Benar
12	955	1070	Basmathi	Basmathi (97,99%)	Benar
13	1001	1070	Basmathi	Basmathi (74,19%)	Benar
14	976	1070	Basmathi	Basmathi (50,93%)	Benar
15	959	1070	Basmathi	Basmathi (54,19%)	Benar
16	1315	1070	IR-64	IR-64 (99,88%)	Benar
17	986	1070	IR-64	IR-64 (99,44%)	Benar
18	1009	1070	IR-64	IR-64 (99,77%)	Benar
19	996	1070	IR-64	IR-64 (99,89%)	Benar
20	982	1070	IR-64	IR-64 (91,49%)	Benar
21	978	1070	IR-64	IR-64 (89,47%)	Benar
22	996	1070	IR-64	IR-64 (99,86%)	Benar
23	971	1070	IR-64	IR-64 (99,75%)	Benar
24	972	1070	IR-64	IR-64 (99,98%)	Benar
25	956	1070	IR-64	IR-64 (73,32%)	Benar
26	1032	1070	IR-64	IR-64 (99,97%)	Benar
27	975	1070	IR-64	IR-64 (99,99%)	Benar
28	988	1070	IR-64	IR-64 (95,04%)	Benar
29	971	1070	IR-64	IR-64 (98,88%)	Benar
30	950	1070	IR-64	IR-64 (99,88%)	Benar
31	1305	1070	Ketan	Ketan (90,4%)	Benar
32	969	1070	Ketan	Ketan (99,98%)	Benar
33	948	1070	Ketan	Ketan (97,57%)	Benar
34	990	1070	Ketan	Ketan (99,45%)	Benar
35	975	1070	Ketan	Ketan (99,96%)	Benar
36	986	1070	Ketan	Ketan (99,99%)	Benar
37	986	1070	Ketan	Ketan (99,53%)	Benar
38	917	1070	Ketan	Ketan (99,87%)	Benar
39	945	1070	Ketan	Ketan (95,17%)	Benar
40	940	1070	Ketan	Ketan (99,01%)	Benar
41	949	1070	Ketan	Ketan (97,35%)	Benar
42	985	1070	Ketan	Ketan (81,9%)	Benar
43	962	1070	Ketan	Ketan (99,8%)	Benar
44	988	1070	Ketan	Ketan (99,96%)	Benar
45	976	1070	Ketan	Ketan (94,28%)	Benar

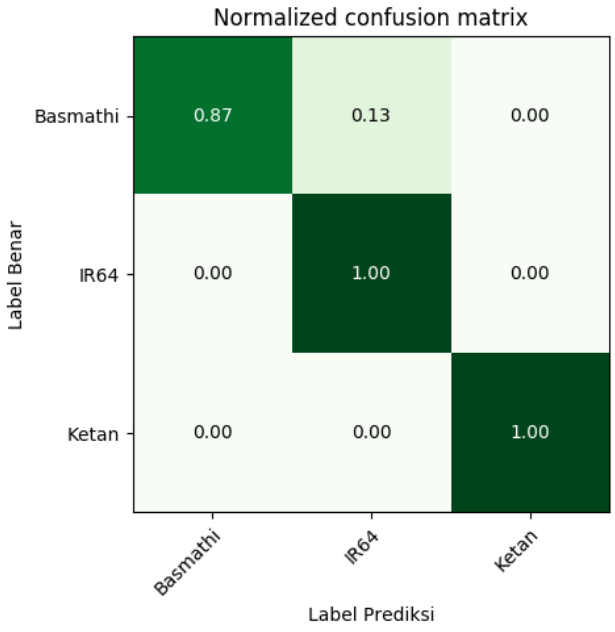
Dari tabel pengujian tersebut dapat dilihat bahwa hasil prediksi untuk IR-64 dan Ketan benar semua, dan Basmathi hanya terdapat 2 kesalahan yang memprediksi IR-64.

Persentase tingkat akurasi dan kesalahan dari pengujian tersebut adalah sebagai berikut.

$$\%akurasi = \frac{43}{45} x100\% = 95{,}56\%$$

$$\%kesalahan = \frac{2}{45} x100\% = 4{,}44\%$$

Dari persentase tersebut dapat disimpulkan bahwa jaringan CNN tersebut masih mampu melakukan klasifikasi dengan baik dengan kondisi penempatan obyek beras yang berbeda. Tingkat akurasi tiap varietas berasnya ditampilkan dalam *confusion matrix* pada gambar 36.



Gambar- 36 *Confusion matrix* pengujian arsitektur *MobileNetV1* dengan *flashlight* pada beras terbungkus

Berdasarkan *confusion matrix* tersebut disimpulkan bahwa hasil pengujian jaringan CNN

tersebut dapat melakukan klasifikasi dengan baik karena intensitas cahaya yang merata dan pantulan cahaya dari plastik hanya sedikit.

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

Hasil pengujian yang kelima dapat dilihat pada tabel 7.

Tabel- 7 Hasil pengujian arsitektur *MobileNetV1* tanpa *flashlight* pada beras dibungkus plastik bening

Pengu-jian ke-	Waktu (ms)	Intensitas Cahaya (*lux*)	Varietas Beras	Hasil Prediksi (Tingkat Probabilitas)	Benar/ Salah
1	983	2110	Basmathi	Basmathi (49,96%)	Benar
2	969	2110	Basmathi	Basmathi (72,34%)	Benar
3	984	2110	Basmathi	Basmathi (97,03%)	Benar
4	1002	2110	Basmathi	Basmathi (79,11%)	Benar
5	966	2110	Basmathi	Basmathi (99,03%)	Benar
6	993	2110	Basmathi	Basmathi (98,87%)	Benar
7	974	2110	Basmathi	Basmathi (89,93%)	Benar
8	923	2110	Basmathi	Basmathi (86,05%)	Benar
9	994	2110	Basmathi	IR-64 (52,84%)	Salah
10	1010	2110	Basmathi	Basmathi (96,68%)	Benar
11	965	2110	Basmathi	Basmathi (68,25%)	Benar
12	1026	2110	Basmathi	Basmathi (97,4%)	Benar
13	1066	2110	Basmathi	Basmathi (97,5%)	Benar
14	953	2110	Basmathi	Basmathi (88,67%)	Benar
15	940	2110	Basmathi	Basmathi (50,15%)	Benar
16	1541	2110	IR-64	IR-64 (99,44%)	Benar
17	993	2110	IR-64	IR-64 (84,62%)	Benar
18	956	2110	IR-64	IR-64 (99,19%)	Benar
19	965	2110	IR-64	IR-64 (98,32%)	Benar
20	973	2110	IR-64	IR-64 (99,02%)	Benar
21	1109	2110	IR-64	IR-64 (80,65%)	Benar
22	956	2110	IR-64	IR-64 (99,86%)	Benar
23	1024	2110	IR-64	IR-64 (88,59%)	Benar
24	968	2110	IR-64	IR-64 (98,02%)	Benar
25	1038	2110	IR-64	IR-64 (99,4%)	Benar
26	930	2110	IR-64	IR-64 (96,26%)	Benar
27	993	2110	IR-64	IR-64 (99,77%)	Benar
28	1026	2110	IR-64	IR-64 (95,64%)	Benar
29	994	2110	IR-64	IR-64 (85,93%)	Benar
30	977	2110	IR-64	IR-64 (99,47%)	Benar
31	1729	2110	Ketan	Ketan (97,62%)	Benar
32	1012	2110	Ketan	Ketan (92,49%)	Benar
33	1052	2110	Ketan	Ketan (98,79%)	Benar
34	1058	2110	Ketan	Ketan (98,57%)	Benar
35	1038	2110	Ketan	Ketan (97,44%)	Benar
36	1026	2110	Ketan	Ketan (99,83%)	Benar
37	1007	2110	Ketan	Ketan (99,7%)	Benar
38	1003	2110	Ketan	Ketan (99,97%)	Benar
39	1011	2110	Ketan	Ketan (98,79%)	Benar
40	1043	2110	Ketan	Ketan (99,53%)	Benar
41	1047	2110	Ketan	Ketan (98,34%)	Benar
42	1041	2110	Ketan	Ketan (99,99%)	Benar
43	1014	2110	Ketan	Ketan (89,78%)	Benar
44	1022	2110	Ketan	Ketan (99,87%)	Benar
45	1002	2110	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.

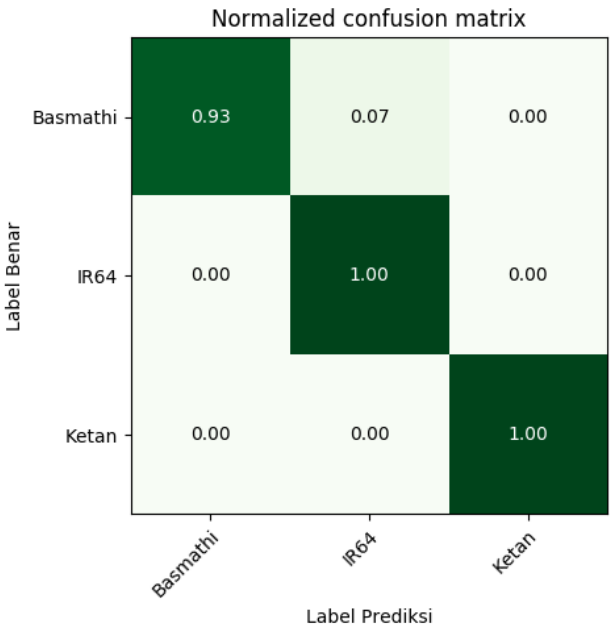
Persentase tingkat akurasi dan kesalahan dari pengujian tersebut adalah sebagai berikut.

$$\%akurasi = \frac{44}{45} x100\% = 97{,}78\%$$

$$\%kesalahan = \frac{1}{45} x100\% = 2{,}22\%$$

Berdasarkan persentase tersebut disimpulkan arsitektur *MobileNetV1* masih mampu melakukan klasifikasi dengan baik pada kondisi beras tersebut. Tingkat akurasi tiap varietas berasnya ditampilkan dalam *confusion matrix* pada gambar 37.

Berdasarkan *confusion matrix* tersebut diperoleh bahwa hasil klasifikasi tiap varietas berasnya sangat akurat dikarenakan penyebaran beras yang dibungkus dan intensitas cahaya yang lebih merata daripada beras yang disebar.



Gambar- 37 *Confusion matrix* pengujian arsitektur *MobileNetV1* tanpa *flashlight* pada beras terbungkus

V. KESIMPULAN DAN SARAN

A. Kesimpulan

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 disebarkan 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 disebarkan 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 disebarkan yang hanya mencapai 71,11% karena cahaya yang mengenai beras yang dibungkus lebih merata dibandingkan dengan beras yang disebarkan.
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*.

B. Saran

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 pendeteksian varietas beras pada aplikasi *Android* dikembangkan lagi agar dapat mendeteksi 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] A 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:// eveloper.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].