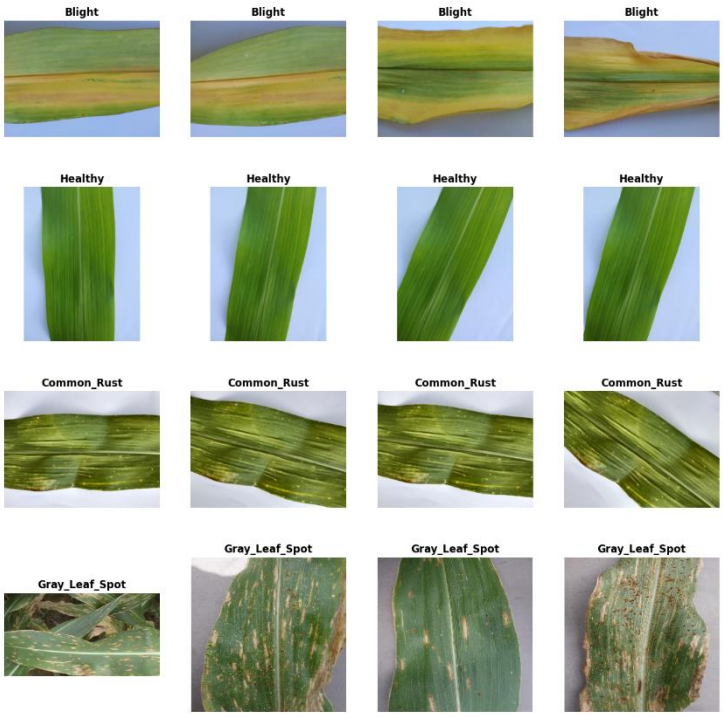
BAB IV
HASIL DAN PEMBAHASAN

4.1. *Dataset*

Dataset daun jagung diperoleh dari *open dataset* (dataset terbuka) dari situs *Kaggle*. *Dataset* tersebut merupakan *dataset* yang berisi gambar dengan total 4000 gambar. 4000 gambar tersebut terbagi kedalam 4 kelas yaitu, 1000 gambar tanaman jagung sehat (*healthy*), 1000 gambar tanaman jagung terkena hawar (*blight*), 1000 gambar tanaman jagung terkena karat daun jagung (*common rust*), 1000 gambar tanaman jagung terkena bercak daun abu-abu (*gray leaf spot*). Pada gambar 4.1 merupakan beberapa gambar dari hasil *dataset*.

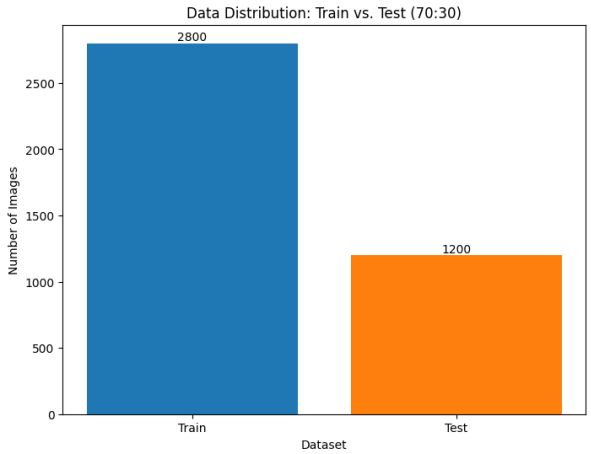


Gambar 4.1 *Dataset* daun jagung

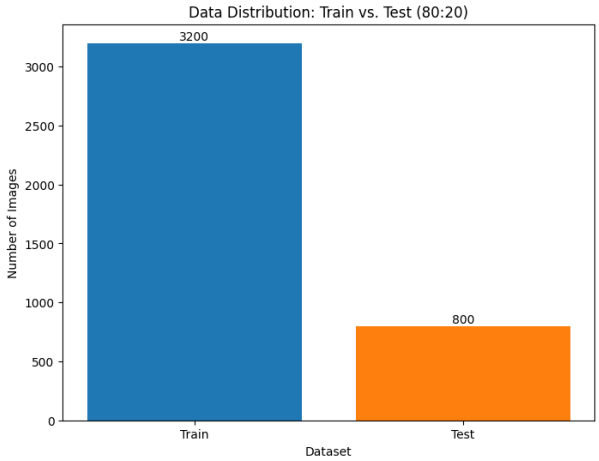
4.2. *Preprocessing*

Dataset akan dibagi menjadi dua *subset* yaitu *train set* dan *testing* set. Pembagian akan dilakukan menjadi dua yaitu 80:20 dan 70:30 sehingga 4000 gambar dengan

rincian 2800 gambar *train* dan 1200 gambar *test* untuk pembagian 70:30 dan 3200 gambar *train* dan 800 gambar *test* untuk pembagian 80:20. Hal tersebut dilakukan untuk mempermudah dalam melakukan pelatihan terhadap model dan mencari hasil terbaik dari 2 pembagian tersebut.

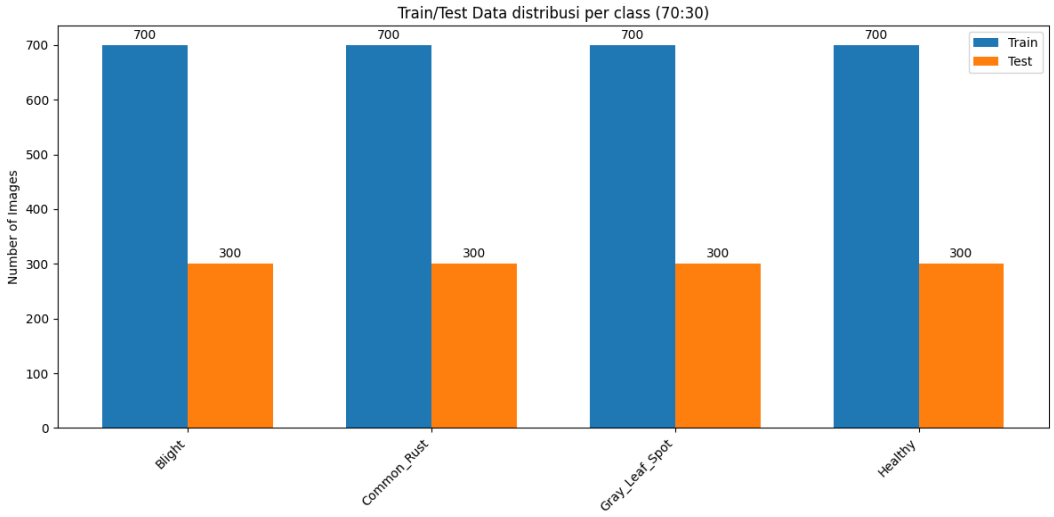


Gambar 4.2 Distribusi train dan test 70:30

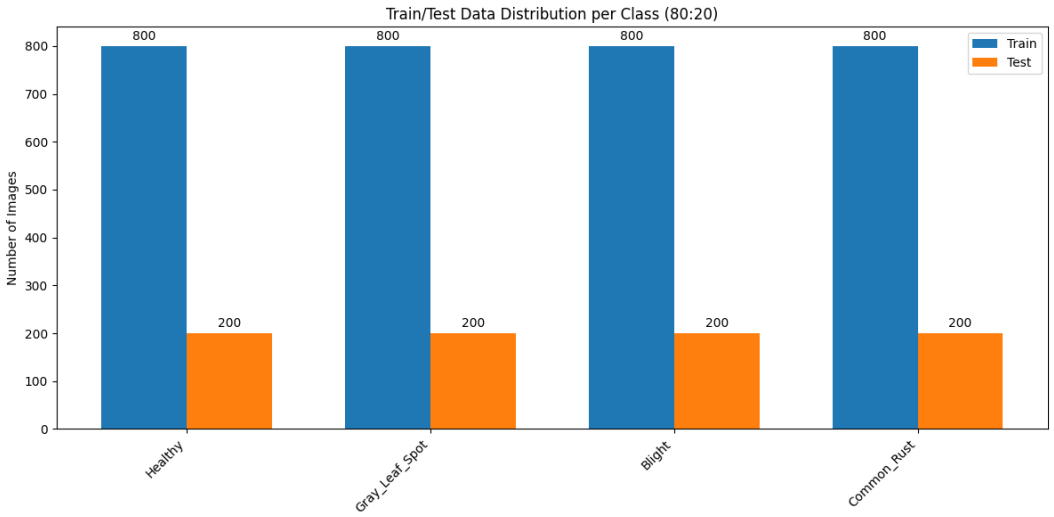


Gambar 4.3 Distribusi train dan test 80:20

Dengan menggunakan dua pembagian data yaitu 80:20 dan 70:30 maka pada masing-masing kelas yaitu *healthy*, *gray leaf spot*, *blight*, dan *common rust* akan terbagi dengan rincian 700 gambar *train* dan 300 gambar *test* setiap kelasnya pada pembagian 70:30 dan 800 gambar *train* dan 200 gambar *test* setiap kelasnya pada pembagian 80:20.



Gambar 4.4 Distribusi kelas 70:30



Gambar 4.5 Distribusi kelas 80:20

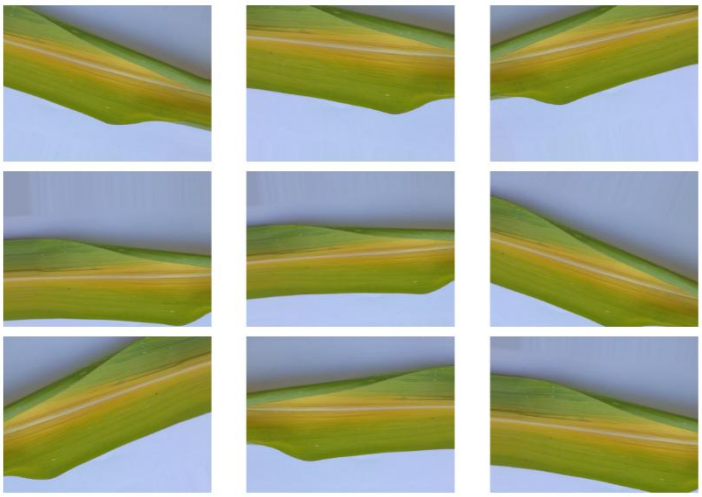
Data yang sudah terbagi menjadi dua selanjutnya akan dilakukan normalisasi untuk membantu model belajar lebih cepat dan akurat. Normalisasi dilakukan dengan mengubah nilai piksel yang sebelumnya berada pada rentang 0–255 menjadi rentang 0–1 menggunakan rescale=1./255. Selain itu, dilakukan *augmentasi* data untuk memperbanyak variasi gambar agar model lebih adaptif terhadap perubahan pada data nyata atau data input. Berikut ini kode yang digunakan dalam melakukan proses *augmentasi* gambar.

```
train_datagen = ImageDataGenerator(
    rescale=1./255,           # Mengubah skala piksel dari [0, 255] menjadi [0, 1].
    rotation_range=20,        # Memutar gambar acak hingga 20 derajat.
    width_shift_range=0.2,    # Menggeser gambar secara horizontal
    height_shift_range=0.2,   # Menggeser gambar secara vertical
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'       # Mengisi piksel kosong dengan nilai piksel terdekat.
)
```

Gambar 4.6 *Augmentasi* gambar

Pada kode tersebut, digunakan fungsi *ImageDataGenerator* yang berfungsi untuk melakukan *augmentasi* pada gambar, dan hasilnya disimpan dalam *variabel train_datagen*. Berikut adalah beberapa parameter yang diterapkan dalam proses *augmentasi* gambar :

- *rescale* = 1./255, berfungsi untuk merubah skala *pixel* RGB (*red,green,blue*) pada gambar yang semula memiliki rentang 0-255 dirubah menjadi 0-1 untuk mempercepat proses belajar model.
- *rotation_range* = 20, berfungsi utnuk mengacak gambar dengan memutar gambar hingga maksimal 20 derajat.
- *width_shift_range* = 0.2, berfungsi menggeser gambar secara *horizontal* (kanan atau kiri) sebesar 20% dari lebar gambar.
- *height_shift_range* = 0.2, berfungsi menggeser gambar secara *vertical* (atas atau bawah) sebesar 20% dari tinggi gambar.
- *shear_range* = 0.2, berfungsi untuk membuat gambar menjadi miring sebesar 20%.
- *zoom_range* = 0.2, berfungsi memperbesar atau memperkecil gambar secara acak hingga 20%
- *horizontal_flip* = True, berfungsi membalik gambar secara *horizontal*.
- *fill_mode* = ‘nearest’, memiliki tujuan untuk mengisi *pixel* kosong yang diakibatkan oleh *transformasi*.



Gambar 4.7 Hasil data *augmentasi*

Gambar 4.7 menunjukan hasil dari proses *augmentasi* gambar. Pada proses ini, setiap gambar mengalami transformasi secara acak berdasarkan parameter yang telah ditentukan, seperti rotasi, pergeseran, *shear* (bergeser), *zoom* (diperbesar), *flipping* (membalik), dan perubah skala piksel. Gambar dapat diputar, diperbesar, atau dimiringkan secara acak sesuai dengan parameter yang telah dikonfigurasikan sebelumnya.

Karena *augmentasi* dilakukan secara berkelanjutan pada setiap proses pelatihan, jumlah variasi yang dihasilkan satu gambar dapat dikatakan hampir tidak terbatas. Hal ini disebabkan oleh kombinasi transformasi yang diterapkan secara acak, sehingga setiap gambar yang sama dapat memiliki tampilan yang berbeda setiap kali proses dalam *batch* yang berbeda. Selain itu, pengolahan piksel akibat transformasi seperti rotasi dan shear juga menghasilkan nilai piksel baru yang bervariasi dalam skala angka desimal. Oleh karena itu, *augmentasi* ini membantu meningkatkan keberagaman data pelatihan tanpa harus menambahkan jumlah gambar secara langsung, yang pada akhirnya dapat meningkatkan kinerja model dalam mengenali pola pada berbagai kondisi.

4.3. *Arsitektur Convolutional Neural Network*

Pada pemodelan *Convolutional Neural Network* (CNN) akan menggunakan 4 layer *convolutional*, 4 layer *MaxPooling*, 2 layer *Batch Normalization*, 2 layer *Neural*

Network, dan 1 layer *dropout*. Permodelan secara keseluruhan dapat dilihat pada gambar 4.8 sebagai berikut :

```
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', padding='same', input_shape=(256, 256, 3))) # Input shape 256x256x3 # Kernel 3x3 # Filter layer 32
model.add(MaxPooling2D(2,2))
model.add(BatchNormalization())
model.add(Conv2D(32, (3,3), activation='relu', padding='same')) #Konvolusi kedua 32 filter ukuran kernel 3x3
model.add(MaxPooling2D(2,2))
model.add(BatchNormalization())
model.add(Conv2D(64,(3,3),activation='relu', padding='same')) # Konvolusi ketiga 64 filter ukuran kernel 3x3
model.add(MaxPooling2D(2,2))
model.add(Conv2D(64,(3,3),activation='relu', padding='same')) # Konvolusi keempat 64 filter ukuran kernel 3x3
model.add(MaxPooling2D(2,2))
model.add(Flatten())
model.add(Dense(126, activation='relu')) #Lapisan fully connected
model.add(Dropout(0.2))
model.add(Dense(4, activation='softmax')) #Klasifikasi multi-kelas (4 kelas)
```

Gambar 4.8 Permodelan *Convolutional Neural Network*

Pada permodelan *Convolutional Neural Network* diatas dapat dilihat bahwa terdapat 4 layer, dengan layer paling tinggi yaitu 256 yang selanjutnya selalu dibagi 2 yaitu 128, 64, dan 32. Pada masing-masing layer tersebut memiliki *padding* = '*same*' yang berguna untuk mempertahankan dimensi *output* agar sama dengan dimensi *input*, sehingga tidak akan kehilangan informasi *pixel* pada bagian tepi gambar. Pada masing-masing layer menggunakan *kernel* 3x3, selanjutnya *MaxPooling* memiliki ukuran 2x2. Jika keempat layer telah diproses maka akan masuk ke *array* 1 dimensi yaitu *Flatten*. Kemudian masuk ke proses *Dropout* dengan menonaktifkan sejumlah *neuron* secara acak sebanyak 0.2 (20%). Selanjutnya semua proses tersebut akan masuk ke *neural network* yang memiliki empat kelas. Hasil dari permodelan tersebut sebagai berikut :

Model: "sequential"

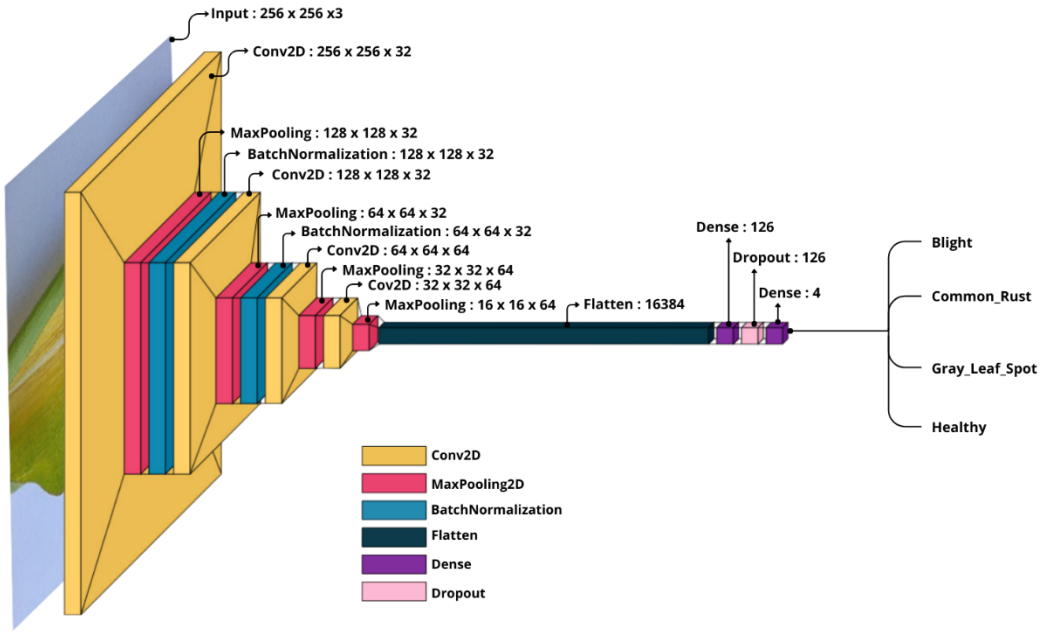
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 256, 256, 32)	896
max_pooling2d (MaxPooling2D)	(None, 128, 128, 32)	0
batch_normalization (BatchNormalization)	(None, 128, 128, 32)	128
conv2d_1 (Conv2D)	(None, 128, 128, 32)	9,248
max_pooling2d_1 (MaxPooling2D)	(None, 64, 64, 32)	0
batch_normalization_1 (BatchNormalization)	(None, 64, 64, 32)	128
conv2d_2 (Conv2D)	(None, 64, 64, 64)	18,496
max_pooling2d_2 (MaxPooling2D)	(None, 32, 32, 64)	0
conv2d_3 (Conv2D)	(None, 32, 32, 64)	36,928
max_pooling2d_3 (MaxPooling2D)	(None, 16, 16, 64)	0
flatten (Flatten)	(None, 16384)	0
dense (Dense)	(None, 126)	2,064,510
dropout (Dropout)	(None, 126)	0
dense_1 (Dense)	(None, 4)	508

Total params: 2,130,842 (8.13 MB)
Trainable params: 2,130,714 (8.13 MB)
Non-trainable params: 128 (512.00 B)

Gambar 4.9 Hasil permodelan CNN

Penggunaan 4 layer CNN memberikan keseimbangan yang optimal antara kompleksitas, akurasi, dan efisiensi komputasi. Dengan struktur ini, model mampu mengekstraksi fitur dari data secara bertahap, mulai dari pola sederhana hingga fitur yang lebih kompleks, sehingga meningkatkan kemampuan dalam mengenali karakteristik objek. Selain itu, jumlah layer yang digunakan membantu mengurangi risiko overfitting, sehingga model tetap dapat melakukan generalisasi dengan baik pada data baru. Penggunaan arsitektur ini juga memungkinkan proses pelatihan yang lebih cepat dan efisien tanpa mengorbankan performa model.

Adapun model visualisasi dalam bentuk 3D untuk memudahkan dalam pemahaman permodelan CNN yang terdapat pada gambar 4.10 Visualisasi arsitektur CNN.



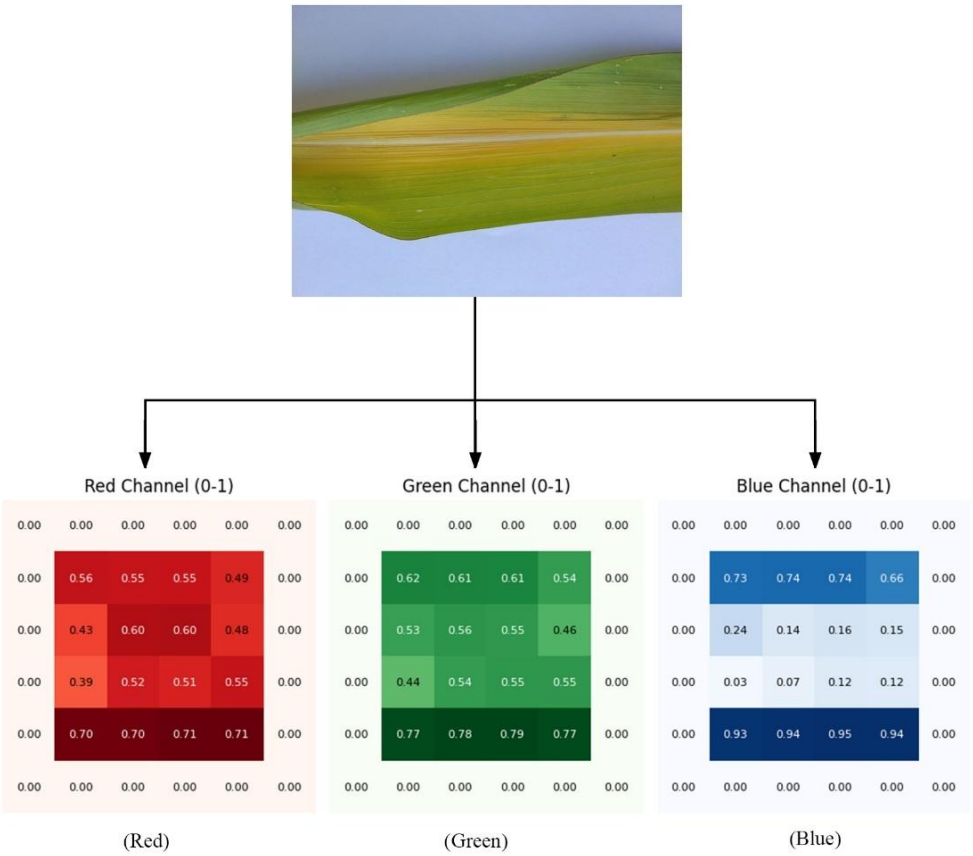
Gambar 4.10 Visualisasi arsitektur CNN

4.4. Training

Dalam proses *training*, sebuah gambar akan melalui empat layer *Convolutional Neural Network* (CNN), yaitu *input layer*, *convolution layer*, *pooling layer*, *flatten layer*, *dropout* dan juga *fully connected layer*, berikut adalah tahapan lengkapnya :

1. *Input layer*

Pada tahap pertama, gambar akan masuk ke dalam *input layer*. Dalam tahapan ini gambar akan dikonversi menjadi matriks tiga dimensi dengan ukuran panjang dan lebar, serta *channel* warna akan terdiri dari tiga nilai, yaitu RGB (*red*, *green*, *blue*). Berikut ini hasil input menggunakan salah satu daun jagung.

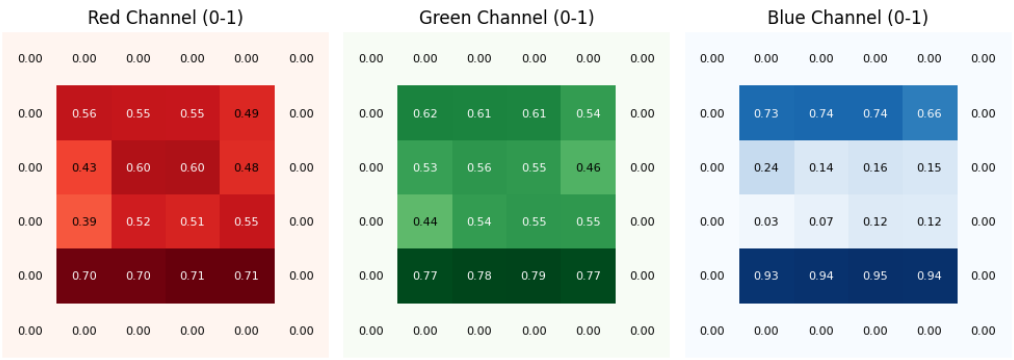


Gambar 4.11 Hasil *input layer*

Pada gambar 4.11 pada gambar daun jagung dirubah menjadi ukuran 4 x 4 *pixel* agar dapat dilakukan ilustrasi. Selanjutnya gambar tersebut di rescale yang sebelumnya berada pada rentang 0–255 menjadi rentang 0–1 Pada bagian tepi daripada *channel* terdapat angka 0.00 karena menggunakan *padding* = '*same*' supaya mempertahankan dimensi ouput.

2. *Convolution layer*

Setelah melakukan *input* pada gambar selanjutnya akan masuk ke layer atau lapisan konvolusi, pada layer ini akan mengekstrak nilai RGB (*red*, *green*, *blue*) pada setiap *pixel*nya sehingga akan mendapatkan *output* berupa *feature map*. Berikut ini merupakan hasil dari *output* berupa *feature map* pada nilai RGB.



Gambar 4.12 Citra warna RGB

Pada tahapan ini selanjutnya akan dilakukan perhitungan menggunakan sebuah filter yang berukuran 3x3 dengan nilai acak yaitu sebagai berikut.

1	0	-1
1	0	-1
1	0	-1

Gambar 4.13 Nilai acak *kernel* 3x3

Selanjutnya pada setiap titik dilakukan perhitungan untuk mencari sebuah *feature map*, berikut adalah perhitungan pada citra warna RGB.

Channel Red

Kolom 1,1 = ((1 * 0) + (0 * 0) + ((-1)*0) + (1 * 0) + (0 * 0.56) + ((-1)*0.55) + (1 * 0) + (0 * 0.43) + ((-1)*0.60) = -1.15

Kolom 1,2 = ((1 * 0) + (0 * 0) + ((-1)*0) + (1 * 0.56) + (0 * 0.55) + ((-1)*0.55) + (1 * 0.43) + (0 * 0.60) + ((-1)*0.60) = -0.16

Pada kolom 1,1 memiliki hasil -1.15 dan kolom 1,2 adlaah -0.16, sehingga hasil keseluruhan pada *channel red* terlihat pada gambar 4.14.

-1.15	-0.16	0.17	1.15
-1.67	-0.27	0.14	1.65
-1.82	-0.30	0.07	1.82
-1.22	-0.13	-0.04	1.22

Gambar 4.14 Hasil perhitungan *channel red*

Channel Green

Kolom 1,1 = ((1 * 0) + (0 * 0) + ((-1)*0) + (1 * 0) + (0 * 0.62) + ((-1)*0.61) + (1 * 0) + (0 * 0.53) + ((-1)*0.56) = -1.17

Kolom 1,2 = ((1 * 0) + (0 * 0) + ((-1)*0) + (1 * 0.62) + (0 * 0.61) + ((-1)*0.61) + (1 * 0.53) + (0 * 0.56) + ((-1)*0.55) = -0.01

Pada kolom 1,1 memiliki hasil -1.17 dan kolom 1,2 adalah -0.01, sehingga hasil keseluruhan pada *channel green* terlihat pada gambar 4.15.

-1.17	-0.01	0.17	1.15
-1.71	-0.13	0.16	1.71
-1.88	-0.15	0.09	1.89
-1.32	-0.13	-0.00	1.34

Gambar 4.15 Hasil perhitungan *channel green*

Channel Blue

Kolom 1,1 = ((1 * 0) + (0 * 0) + ((-1)*0) + (1 * 0) + (0 * 0.73) + ((-1)*0.74) + (1 * 0) + (0 * 0.24) + ((-1)*0.14) = -0.88

Kolom 1,2 = ((1 * 0) + (0 * 0) + ((-1)*0) + (1 * 0.73) + (0 * 0.74) + ((-1)*0.74) + (1 * 0.24) + (0 * 0.14) + ((-1)*0.16) = 0.07

Pada kolom 1,1 memiliki hasil -0.88 dan kolom 1,2 adalah 0.07, sehingga hasil keseluruhan pada *channel blue* dapat dilihat pada gambar 4.16.

-0.88	0.07	0.07	0.91
-0.95	-0.04	0.02	1.03
-1.14	-0.04	-0.06	1.23
-1.00	-0.11	-0.05	1.07

Gambar 4.16 Hasil perhitungan *channel blue*

Setiap nilai daripada channel RGB selanjutnya dilakukan penjumlahan sehingga akan menghasilkan *output* total seperti yang ditunjukkan pada gambar 4.17.

-3.2	-0.1	0.41	3.21
-4.33	-0.44	0.32	4.39
-4.84	-0.49	0.1	4.94
-3.54	-0.37	-0.09	3.63

Gambar 4.17 Total perhitungan *convolutional*

Setelah proses penjumlahan pada *channel* selesai maka selanjutnya yaitu melakukan fungsi aktivasi, pada permodelan penelitian ini menggunakan *ReLU* yaitu jika nilai *input* positif maka akan mengembalikan nilai *input* tersebut, namun

jika nilai *input* tersebut negatif, maka akan mengembalikannya dengan nilai nol. Berikut ini merupakan rumus dari *ReLU*.

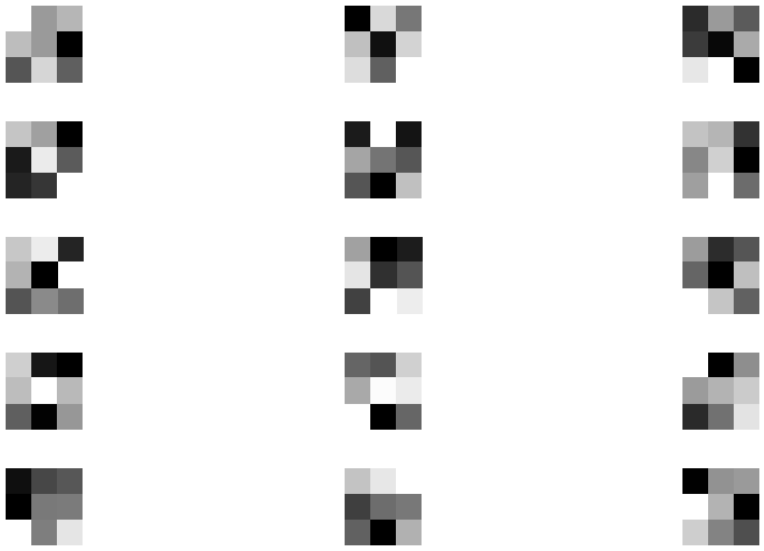
$$f(x) = maks(0,x)$$

Dengan demikian, hasil penggunaan ReLU ditampilkan pada gambar 4.18.

0	0	0.41	3.21
0	0	0.32	4.39
0	0	0.1	4.94
0	0	0	3.63

Gambar 4.18 Hasil *ReLU*

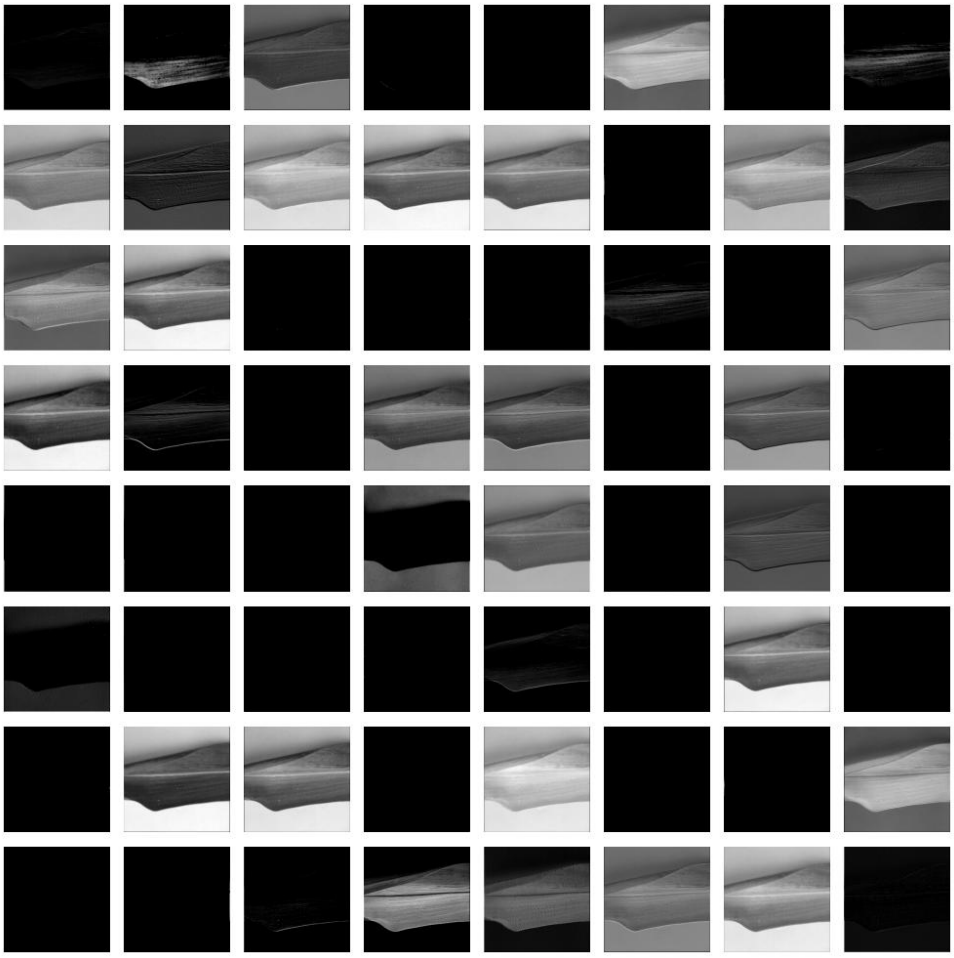
Setelah proses konvolusi, terjadi proses ekstraksi fitur yang membentuk *feature map* pada sebuah gambar. Dalam pengenalan gambar, struktur dalam gambar dikenali dengan bantuan filter. Di bawah ini merupakan filter konvolusi atau visualiasi filter dari sebuah gambar.



Gambar 4.19 Filter konvolusi / visualisasi filter

Pada gambar 4.19 dapat dilihat pada setiap filter yang berukuran 3x3 memiliki warna berbeda-beda mulai dari hitam, abu-abu hingga putih. Dimana pada setiap warna memiliki maksudnya masing-masing yaitu pada warna hitam menunjukan area dimana filter tidak mendeteksi pola, warna putih menunjukkan area filter mendeteksi pola tertentu, dan warna abu-abu menunjukan intesitas menengah. Dengan warna filter yang berbeda, CNN (*Convolutional Neural* Network) melakukan konvolusi pada data *input* untuk menghasilkan fitur-fitur dasar seperti

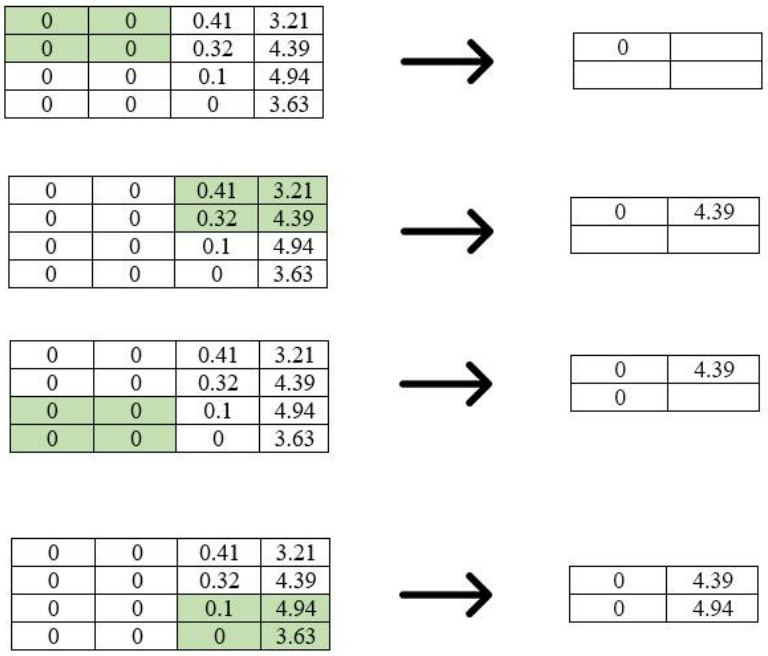
tepi (*edges*), tekstur, dan pola-pola sederhana yang diperlihatkan dalam gambar 4.20.



Gambar 4.20 Hasil proses konvolusi

3. *Pooling layer*

Dari hasil *ReLU* pada gambar 4.18 akan dijadikan sebagai *inputan* untuk *pooling layer*, pada tahapan *pooling layer* menggunakan *maxpooling* dengan *kernel* 2x2, Gambar 4.21 menunjukkan ilustrasi dari proses *maxpooling*.

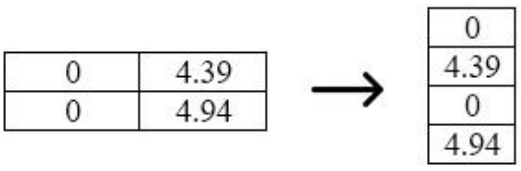


Gambar 4.21 Hasil *pooling layer*

Pada gambar 4.21, dilakukan tahapan *pooling layer* dengan menggunakan *kernel* 2x2, pada tahapan ini nilai diambil dari nilai tertinggi yaitu 0 , 4.39, 0, 4.94.

4. *Flatten Layer*

Flatten layer digunakan untuk mengubah matriks kebentuk 1 dimensi, hal tersebut akan mendapatkan bentuk *vector baru*, ilustrasi proses *flatten layer* ditunjukkan pada gambar 4.22.



Gambar 4.22 Hasil *flatten layer*

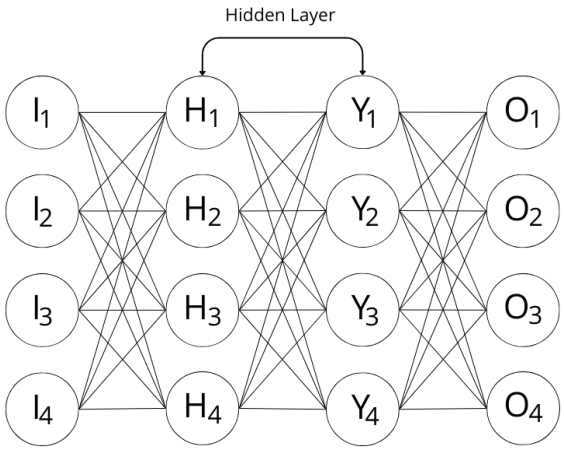
Hasil yang diperoleh dari *flatten layer* yang mana *matrix* yang sebelumnya memiliki dimensi 2x2 diubah menjadi 1x4. *Output* dari *flatten layer* ini yang nantinya digunakan dalam proses *input* di *fully connected layer*.

5. *Dropout*

Dropout digunakan untuk mengurangi kompleksitas model dengan menonaktifkan sejumlah *neuron* secara acak sebanyak 0.2 (20%). Hal tersebut tidak akan mempengaruhi ukuran model.

6. *Fully connected layer*

Dalam proses *fully connected layer*, *input* yang digunakan berasal dari *flatten layer* yang diubah kedalam bentuk dimensi 1 x 4, pada gambar 4.23 merupakan ilustrasi *fully connected layer*.



Gambar 4.23 Ilustrasi *dense + softmax*

Lapisan terakhir dari model CNN yang dikembangkan adalah *dense* dengan fungsi aktivasi *softmax*. Berikut ini merupakan perhitungan dari setiap *hidden layer*.

$I_{1,}\, I_{2,}\, I_{3,}\, I_4$ merupakan nilai dari proses *flatten* yang terdapat pada gambar 4.22 yang selajutnya dilakukan proses *fully connected layer* dengan menghitung setiap *hidden layer*, berikut hasil perhitungan pada *hidden layer* pertama :

$H_1 = (0 * 0.2) + (4.39 * 0.2) + (0 * 0.2) + (4.94 * 0.2) = 1.866$

$H_2 = (0 * 0.3) + (4.39 * 0.3) + (0 * 0.3) + (4.94 * 0.3) = 2.799$

$H_3 = (0 * 0.5) + (4.39 * 0.5) + (0 * 0.5) + (4.94 * 0.5) = 4.665$

$H_4 = (0 * 0.4) + (4.39 * 0.4) + (0 * 0.4) + (4.94 * 0.4) = 3.732$

Hasil diatas merupakan nilai *hidden layer* pertama yang mana setiap *neuron* dari I_1, I_2, I_3, I_4 dikalikan dengan bobot (*weight*) yang berbeda pada yang meghasilkan nilai $H_1 = 1.866$, $H_2 = 2.799$, $H_3 = 4.665$ dan $H_4 = 3.732$. Untuk *layer*

selanjutnya nilai pada *neuron* H_1, H_2, H_3, dan H_4 akan kembali dikalikan dengan bobot (*weight*) yang berbeda untuk menghasilkan nilai *hidden layer* selanjutnya.

$Y_1 = (1.866 * 0.3) + (2.799 * 0.3) + (4.665 * 0.3) + (3.732 * 0.3) = 3.919$

$Y_2 = (1.866 * 0.4) + (2.799 * 0.4) + (4.665 * 0.4) + (3.732 * 0.4) = 5.225$

$Y_3 = (1.866 * 0.2) + (2.799 * 0.2) + (4.665 * 0.2) + (3.732 * 0.2) = 2.612$

$Y_4 = (1.866 * 0.5) + (2.799 * 0.5) + (4.665 * 0.5) + (3.732 * 0.5) = 6.531$

Pada setiap *neuron* Y1, Y2, Y3, dan Y4 akan kembali dikalikan dengan bobot (*weight*) untuk menghasilkan nilai O_1, O_2, O_3, dan O_4 sebagai berikut.

$O_1 = (3.919 * 0.5) + (5.225 * 0.5) + (2.612 * 0.5) + (6.531 * 0.5) = 9.144$

$O_2 = (3.919 * 0.3) + (5.225 * 0.3) + (2.612 * 0.3) + (6.531 * 0.3) = 5.486$

$O_3 = (3.919 * 0.1) + (5.225 * 0.1) + (2.612 * 0.1) + (6.531 * 0.1) = 1.829$

$O_4 = (3.919 * 0.4) + (5.225 * 0.4) + (2.612 * 0.4) + (6.531 * 0.4) = 7.315$

Berikutnya adalah perhitungan *softmax* dengan menggunakan rumus eksponensial dengan cara menghitung terlebih dahulu eksponensial pada masing-masing nilai yang selanjutnya dijumlahkan, dan tahap terakhir membagi nilai eksponensial masing-masing nilai dengan hasil yang dijumlahkan. Berikut ini rumus dan ilustrasi perhitungannya :

$$S_i = \frac{e^{O_i}}{\sum e^{O_j}}$$

Hitung nilai eksponensial pada masing-masing nilai :

$$e^{O_1} = e^{9.1435} = 9366.56$$

$$e^{O_2} = e^{5.4861} = 241.96$$

$$e^{O_3} = e^{1.8287} = 6.23$$

$$e^{O_4} = e^{7.3148} = 1948.44$$

Nilai eksponensial tersebut kemudian dijumlahkan :

$$\sum e^{O_j} = 9366.56 + 241.96 + 6.23 + 1948.44 = 11113.19$$

Kemudian dilakukan penghitungan pada masing-masing *softmax* :

$$S_1 = \frac{e^{O_1}}{\sum e^{O_j}} = \frac{9366.56}{11113.19} = 0.842$$

$$S_2 = \frac{e^{O_2}}{\sum e^{O_j}} = \frac{241.96}{11113.19} = 0.0217$$

$$S_3 = \frac{e^{O_3}}{\sum e^{O_j}} = \frac{6.23}{11113.19} = 0.00056$$

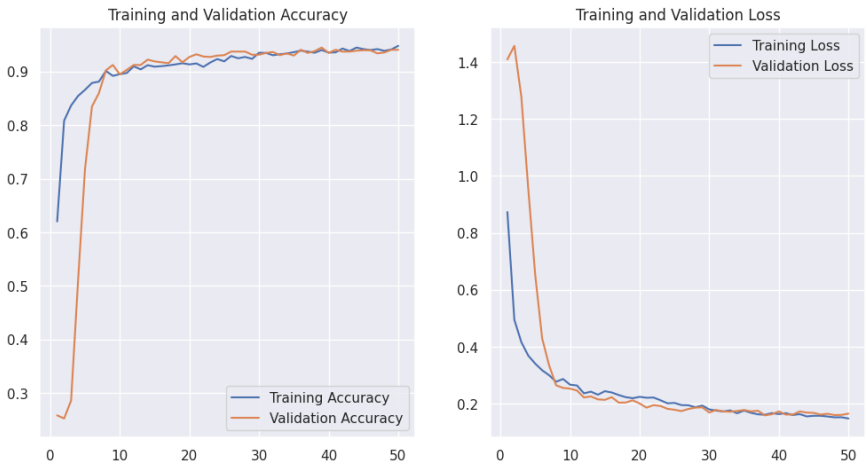
$$S_4 = \frac{e^{O_4}}{\sum e^{O_j}} = \frac{1498.44}{11113.19} = 0.135$$

Sehingga dari hasil perhitungan tersebut didapatkan bobot nilai probabilitas terbesar yaitu pada S_1 yaitu 0.842 atau 84,2% yang berarti input citra yang dimasukkan diprediksi penyakit tanaman jagung jenis *Blight*. Adapun hasil probabilitas untuk kategori lainnya, yaitu S_2 (*Common Rust*) sebesar 0.0217 atau 2,17%, S_3 (*Gray Leaf Spot*) sebesar 0.00056 atau 0,056%, dan S_4 (*Healthy*) sebesar 0.135 atau 13,5%.

4.5. Validasi

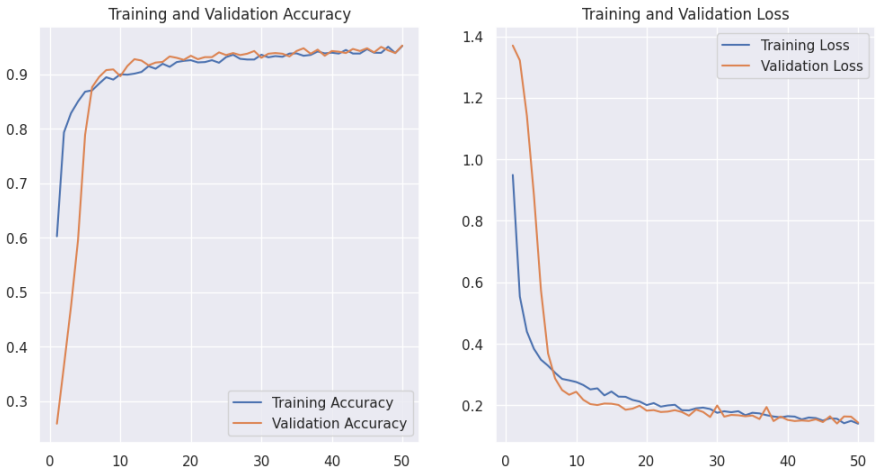
Pada tahap validasi dilakukan pengukuran akurasi daripada model yang sudah menjalankan proses *training* data, pada penelitian ini menggunakan 2 *split* data yang berbeda sehingga akan menghasilkan 2 data hasil *training* data epoch yang dijalankan pada 2 data tersebut ialah 50 epoch. Hasil akurasi dan juga *loss* digunakan sebagai proses apakah *taining* model sudah berjalan dengan baik atau belum, selain itu dari data akurasi dan *loss* dari 2 data akan terlihat perbandingan akurasi dan *loss*.

Pada gambar 4.24 menunjukan hasil daripada akurasi pada *split* data 70:30, akurasi tertinggi yang didapatkan pada data ini terjadi pada *epoch* ke 50 yaitu dengan akurasi 94,82% dan *loss* terendah terjadi pada *epoch* ke 50 yaitu 14,81%. Dari hasil akurasi dan *loss* tersebut model sudah dapat mendeteksi data *train* dengan sangat baik.



Gambar 4.24 Akurasi dan *loss split* data 70:30

Pada gambar 4.25 menunjukan hasil daripada akurasi pada *split* data 80:20, akurasi tertinggi pada *split* data 80:20 terjadi pada *epoch* ke 50 dengan total akurasi yaitu 95,16%, dan data *loss* terendah terjadi pada *epoch* ke 50 yaitu 14,04%. *Split* data 80:20 memiliki akurasi dan *loss* yang lebih unggul dibandingkan dengan *split* data 70:30 yaitu dengan unggul 0,34% pada akurasi. Oleh karena itu *split* data sudah dapat mengenal dengan baik data *train*.

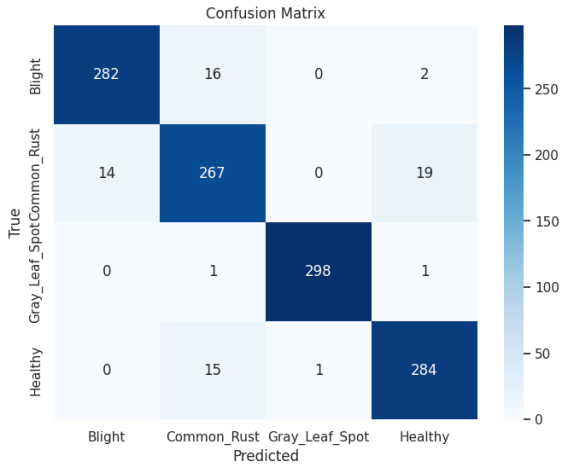


Gambar 4.25 Akurasi dan *loss split* data 80:20

Selain melakukan pemeriksaan pada akurasi dan *loss*, model juga harus diperiksa menggunakan beberapa tahap seperti *confusion matrix*, *F1Score*, *precision*, dan *recall*.

Pada *split* data 70:30 *confusion matrix* yaitu pada gambar 4.26 menunjukan pengetesan pada data *test* dengan cukup baik, namun pada penyakit *common rust*

jagung memiliki kesalahan deteksi yang sekitar 33 gambar. Selain itu pada daun *healthy* dan daun *blight* memiliki kesalahan yang berkisar 16-18 gambar. Percobaan terbaik untuk mendeteksi gambar *test* terjadi pada pendeteksian jenis daun *gray leaf spot* yang hanya memiliki kesalahan deteksi pada 2 daun.



Gambar 4.26 *Confusion matrix split* data 70:30

Dari hasil *confusion matrix* diatas maka *F1Score*, *precision*, dan *recall* ditunjukkan pada gambar 4.27.

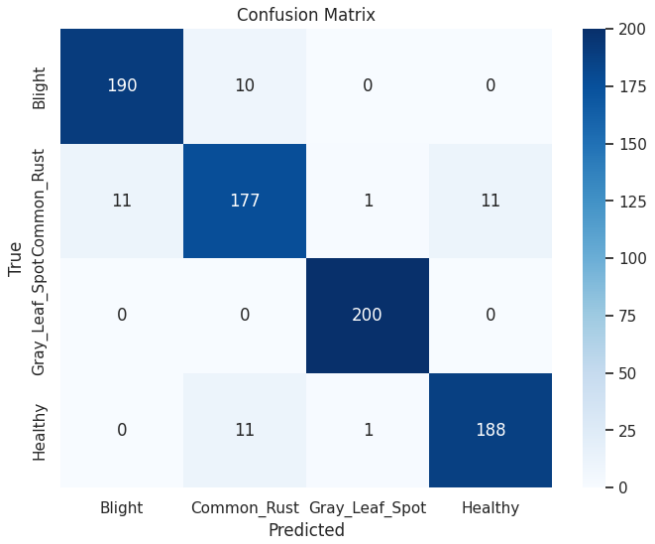
	precision	recall	f1-score	support
Blight	0.96	0.93	0.95	300
Common_Rust	0.89	0.89	0.89	300
Gray_Leaf_Spot	0.99	1.00	1.00	300
Healthy	0.93	0.96	0.94	300
accuracy			0.94	1200
macro avg	0.94	0.94	0.94	1200
weighted avg	0.94	0.94	0.94	1200

Gambar 4.27 *F1Score*, *precision*, dan *recall* 70:30

Dari hasil *F1Score*, *precision*, dan *recall* dapat dilihat hasil yang kurang terdapat pada *common rust* yang hanya mendapatkan akurasi sekitar 89%, berbanding dari *blight*, *gray leaf spot*, dan *healthy* yang mendapatkan akurasi diatas 90%.

Pada *split* data 80:20 memiliki kesamaan dengan *split* data 70:30 yaitu pada daun *common rust* memiliki akurasi yang cukup kurang, dapat diamati pada gambar 4.28, kesalahan yang muncul pada daun *common rust* mencapai 23 gambar, untuk *blight* dan *healthy* memiliki kesalahan berkisar 10-12, hasil terbaik diraih oleh daun

gray leaf spot yang benar 100% tanpa ada kesalahan dalam mendeteksi gambar *train*.



Gambar 4.28 *Confusion matrix split* data 80:20

Dari hasil *confusion matrix* diatas maka *F1Score*, *precision*, dan *recall* ditunjukkan pada gambar 4.29.

	precision	recall	f1-score	support
Blight	0.95	0.95	0.95	200
Common_Rust	0.88	0.92	0.90	200
Gray_Leaf_Spot	0.99	0.99	0.99	200
Healthy	0.97	0.93	0.95	200
accuracy			0.95	800
macro avg	0.95	0.95	0.95	800
weighted avg	0.95	0.95	0.95	800

Gambar 4.29 *F1Score*, *precision*, dan *recall* 80:20

Sesuai dengan *confusion matrix* daun yang terkena *common rust* memiliki akurasi yang paling kecil yaitu 88%, sedangkan untuk 3 daun lainnya memiliki akurasi yang cukup baik yaitu 93% keatas.

4.6. *Testing/Deployment*

Setelah melihat hasil akurasi data, model akan diuji untuk menentukan apakah model tersebut dapat mendeteksi atau memprediksi gambar dengan tepat atau tidak, sama seperti validasi data *testing* akan memiliki 2 data yaitu *split* data 70:30 dan 80:20. Hasil dari *testing* data 70:30 ditunjukkan pada gambar 4.30.

Gambar 4.30 Hasil *testing* data 70:30

Hasil pengujian model menggunakan *split* data 70:30 dan menggunakan 50 gambar terlihat pada tabel 4.1.

Tabel 4.1 Pengujian data 70:30

No	Data asli	Prediksi	Probabilitas
1	*Blight*	*Blight*	95.67%
2	*Blight*	*Blight*	81.08%
3	*Blight*	*Blight*	99.33%
4	*Blight*	*Blight*	99.83%
5	*Blight*	*Blight*	99.51%
6	*Blight*	*Blight*	99.92%
7	*Blight*	*Blight*	78.94%
8	*Blight*	*Blight*	96.98%
9	*Blight*	*Blight*	96.68%
10	*Blight*	*Blight*	99.67%
11	*Blight*	*Blight*	99.94%
12	*Blight*	*Common Rust*	96.91%
13	*Gray Leaf Spot*	*Gray Leaf Spot*	100%
14	*Gray Leaf Spot*	*Gray Leaf Spot*	100%
15	*Gray Leaf Spot*	*Gray Leaf Spot*	100%
16	*Gray Leaf Spot*	*Gray Leaf Spot*	99.99%
17	*Gray Leaf Spot*	*Gray Leaf Spot*	100%
18	*Gray Leaf Spot*	*Gray Leaf Spot*	100%
19	*Gray Leaf Spot*	*Gray Leaf Spot*	100%
20	*Gray Leaf Spot*	*Gray Leaf Spot*	100%
21	*Gray Leaf Spot*	*Gray Leaf Spot*	100%
22	*Gray Leaf Spot*	*Gray Leaf Spot*	100%
23	*Gray Leaf Spot*	*Gray Leaf Spot*	99.71%
24	*Gray Leaf Spot*	*Gray Leaf Spot*	100%
25	*Gray Leaf Spot*	*Gray Leaf Spot*	100%
26	*Healthy*	*Common Rust*	78.94%
27	*Healthy*	*Healthy*	89.96%
28	*Healthy*	*Healthy*	91.01%
29	*Healthy*	*Healthy*	66.90%
30	*Healthy*	*Healthy*	88.99%
31	*Healthy*	*Common Rust*	73.54%
32	*Healthy*	*Healthy*	97.56%
33	*Healthy*	*Healthy*	92.49%
34	*Healthy*	*Healthy*	95.22%
35	*Healthy*	*Healthy*	64.96%
36	*Healthy*	*Healthy*	56.33%
37	*Healthy*	*Healthy*	91.67%
38	*Healthy*	*Healthy*	83.69%
39	*Common Rust*	*Common Rust*	97.74%
40	*Common Rust*	*Common Rust*	99.40%
41	*Common Rust*	*Blight*	74.10%
42	*Common Rust*	*Common Rust*	99.72%
43	*Common Rust*	*Common Rust*	89.32%
44	*Common Rust*	*Common Rust*	90.28%
45	*Common Rust*	*Common Rust*	79.02%
46	*Common Rust*	*Common Rust*	99.92%
47	*Common Rust*	*Common Rust*	99.46%
48	*Common Rust*	*Common Rust*	99.39%
49	*Common Rust*	*Common Rust*	98.09%
50	*Common Rust*	*Common Rust*	99.91%

Pada data tabel 4.1 menunjukan data yang memiliki warna kuning merupakan data yang memiliki kesalahan prediksi. Hasil daripada *split* data 70:30 menunjukan tingkat akurasi mencapai 92%.

$$akurasi = \frac{46}{50}\ x\ 100\%$$

$$akurasi = 92\%$$

Setelah melakukan uji *testing* pada *split* data 70:30 selanjutnya dilakukan uji *testing* pada data 80:20, Hasil dari *testing* data 80:20 dapat dilihat pada gambar 4.31 dan hasil dari pengujian benar atau salah dapat terlihat pada tabel 4.2.

Gambar 4.31 Hasil *testing* data 80:20

Tabel 4.2 Pengujian data 80:20

No	Data asli	Prediksi	Probabilitas
1	*Blight*	*Blight*	98.08%
2	*Blight*	*Blight*	94.11%
3	*Blight*	*Blight*	99.98%
4	*Blight*	*Blight*	99.98%
5	*Blight*	*Blight*	99.59%
6	*Blight*	*Blight*	99.98%
7	*Blight*	*Blight*	97.92%
8	*Blight*	*Blight*	99.46%
9	*Blight*	*Blight*	99.41%
10	*Blight*	*Blight*	99.91%
11	*Blight*	*Blight*	99.95%
12	*Blight*	*Common Rust*	96.61%
13	*Gray Leaf Spot*	*Gray Leaf Spot*	100%
14	*Gray Leaf Spot*	*Gray Leaf Spot*	100%
15	*Gray Leaf Spot*	*Gray Leaf Spot*	100%
16	*Gray Leaf Spot*	*Gray Leaf Spot*	100%
17	*Gray Leaf Spot*	*Gray Leaf Spot*	100%
18	*Gray Leaf Spot*	*Gray Leaf Spot*	100%
19	*Gray Leaf Spot*	*Gray Leaf Spot*	100%
20	*Gray Leaf Spot*	*Gray Leaf Spot*	100%
21	*Gray Leaf Spot*	*Gray Leaf Spot*	100%
22	*Gray Leaf Spot*	*Gray Leaf Spot*	100%
23	*Gray Leaf Spot*	*Gray Leaf Spot*	99.15%
24	*Gray Leaf Spot*	*Gray Leaf Spot*	100%
25	*Gray Leaf Spot*	*Gray Leaf Spot*	100%
26	*Healthy*	*Common Rust*	60.88%
27	*Healthy*	*Healthy*	83.34%
28	*Healthy*	*Healthy*	82.34%
29	*Healthy*	*Healthy*	77.04%
30	*Healthy*	*Healthy*	78.17%
31	*Healthy*	*Common Rust*	91.12%
32	*Healthy*	*Healthy*	98.51%
33	*Healthy*	*Healthy*	87.13%
34	*Healthy*	*Healthy*	96.72%
35	*Healthy*	*Healthy*	68.33%
36	*Healthy*	*Healthy*	80.24%
37	*Healthy*	*Healthy*	94.04%
38	*Healthy*	*Healthy*	59.06%
39	*Common Rust*	*Common Rust*	96.50%
40	*Common Rust*	*Common Rust*	98.78%
41	*Common Rust*	*Blight*	84.91%
42	*Common Rust*	*Common Rust*	99.87%
43	*Common Rust*	*Common Rust*	90.85%
44	*Common Rust*	*Common Rust*	97.25%
45	*Common Rust*	*Common Rust*	92.56%
46	*Common Rust*	*Common Rust*	98.24%
47	*Common Rust*	*Common Rust*	99.09%
48	*Common Rust*	*Common Rust*	98.30%
49	*Common Rust*	*Common Rust*	94.75%
50	*Common Rust*	*Common Rust*	99.55%

Pada tabel 4.2 memiliki kesalahan deteksi yang sama dengan data 70:30, namun pada data 80:20 memiliki akurasi yang cukup tinggi ketika berhasil mendeteksi, untuk tingkat akurasi sampai dengan 92%.

$$akurasi = \frac{46}{50}\ x\ 100\%$$

$$akurasi = 92\%$$

4.7. Implementasi Fitur Deteksi Pada Android

Pada saat pengimplementasian fitur deteksi ke Android, model harus drubah ke dalam format *tensorflow lite*, hal tersebut dilakukan agar model dapat diimplementasikan ke dalam aplikasi Android guna mendeteksi hasil dari gambar daun jagung.

Sebelum melakukan pembuatan aplikasi peneliti melakukan perancangan aplikasi terlebih dahulu dengan menggunakan *Unified Modelling Language* (UML), *diagram* yang diterapkan untuk perancangan diantaranya *use case diagram*, *activity diagram*, *class diagram*, dan *sequence diagram*.

4.8. Tampilan Aplikasi

4.8.1. Tampilan *Splash Screen*

Splash screen ialah tampilan yang pertama tampil saat aplikasi dibuka, yang menampilkan logo dan versi aplikasi. Tampilan ini berdurasi 2 detik sebelum masuk ke menu *home*, berikut tampilan *splash screen* yang teralihat pada gambar 4.32.

Gambar 4.32 Tampilan *splash screen*

4.8.2. Tampilan *Home*

Tampilan *home* adalah tampilan pertama dari aplikasi, di mana pada tampilan ini memiliki fitur untuk melihat info daun tanaman jagung dengan menekan tombol pada salah satu jenis daun.

Gambar 4.33 Tampilan *home*

4.8.3. Tampilan Jenis Penyakit

Jenis penyakit merupakan lanjutan dari tampilan *home* ketika menekan salah satu jenis daun, pada tampilan ini berisi penjelasa jenis daun yang dipilih seperti cara penanganan, sebab, dan lain-lain. Lebih jelasnya dapat terlihat pada gambar 4.34.

Gambar 4.34 Tampilan jenis penyakit

4.8.4. Tampilan Tutorial Penggunaan

Tutorial penggunaan merupakan salah satu fitur dari aplikasi CornAnalyze yang berguna membantu para pengguna baru yang tidak tahu bagaimana cara menggunakan aplikasi.

Gambar 4.35 Tampilan tutorial penggunaan

4.8.5. Tampilan *Scan*

Scan merupakan fitur utama daripada aplikasi CornAnalyze, pada menu ini memiliki beberapa tombol yaitu diantaranya :

- Galeri, tombol galeri berguna bagi pengguna yang ingin mengakses galeri ketika sudah memiliki foto daun jagung.
- Kamera, tombol kamera merupakan tombol yang berguna untuk membuka kamera pada aplikasi CornAnalyze, pengguna dapat melakukan foto langsung dengan menggunakan fitur kamera ini.
- Analisa 70:30, analisa 70:30 merupakan tombol untuk melakukan deteksi pada daun yang sebelumnya sudah dimasukkan oleh pengguna dengan menggunakan model deteksi perbandingan 70:30.
- Analisa 80:80, analisa 80:20 merupakan tombol untuk melakukan deteksi pada daun yang sebelumnya sudah dimasukkan oleh pengguna dengan menggunakan model deteksi perbandingan 80:20.

Gambar 4.36 Tampilan *scan*

4.8.6. Tampilan Kamera

Pada gambar 4.37 menunjukan tampilan kamera yang dapat digunakan oleh pengguna untuk mengambil gambar daun jagung, berikut ini adalah tampilan dari kamera CornAnalyze.

Gambar 4.37 Tampilan kamera

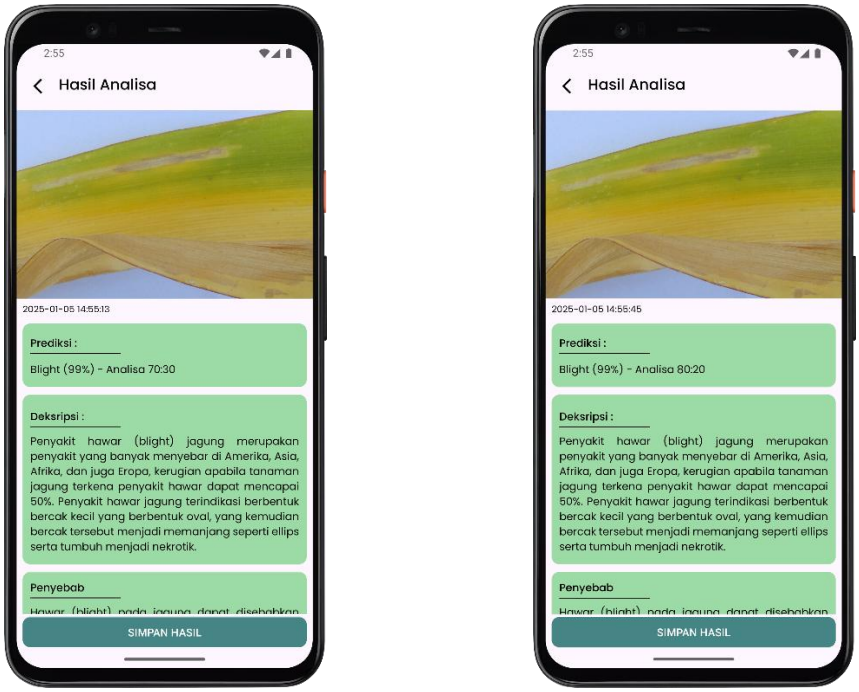
4.8.7. Tampilan *Crop*

Crop merupakan suatu fitur yang dapat digunakan untuk memotong gambar yang sebelumnya dipilih melalui galeri ataupun dengan menggunakan kamera CornAnalyze, untuk tampilan dari *crop* seperti yang dapat dilihat pada gambar 4.38.

Gambar 4.38 Tampilan *crop*

4.8.8. Tampilan Hasil

Tampilan hasil ialah tampilan yang apabila sudah menekan tombol “analisa” pada menu sebelumnya, pada saat ini aplikasi merespon model yang sebelumnya dimasukkan kedalam aplikasi, aplikasi akan mendeteksi jenis penyakit pada daun jagung. Pada hasil tersebut memiliki satu *button* yaitu simpan hasil, *button* tersebut berfungsi untuk menyimpan hasil deteksi.



Gambar 4.39 Tampilan hasil analisa 70:30 dan 80:20

4.8.9. Tampilan *History* Tersimpan

Tampilah *history* tersimpan merupakan hasil dari tombol simpan pada tampilan hasil, semua hasil deteksi yang tersimpan akan tampil di menu tersebut.

Gambar 4.40 Tampilan *history* tersimpan

4.8.10. Tampilan Detail Tersimpan

Pada tampilan ini merupakan tampilan ketika mengklik salah satu hasil deteksi yang sudah tersimpan, isinya yaitu hasil analisa sebelumnya yang sudah disimpan.

Gambar 4.41 Tampilan detail tersimpan

4.8.11. Tampilan Info Aplikasi

Info aplikasi adalah menu yang menyajikan penjelasan dari aplikasi CornAnalyze dan juga berisi kontak pembuat aplikasi.

Gambar 4.42 Tampilan info aplikasi

4.9. Hasil Implementasi Model

Setelah selesai dalam pengembangan fitur deteksi pada aplikasi yang memiliki fokus untuk mendeteksi penyakit pada tanaman jagung. Maka selanjutnya aplikasi tersebut diuji coba dengan menggunakan *dataset* yang sebelumnya digunakan untuk model *tensorflow lite*. Data uji yang digunakan yaitu berupa 30 gambar dengan rincian 7 gambar *healthy*, 8 gambar *gray leaf spot*, 7 gambar *blight*, 8 gambar *common rust*. Penulis akan menggunakan tipe model yaitu 70:30 dan 80:20 yang selanjutnya penulis akan memperlihatkan hasil dari data hasil dan prediksi kemudian menampilkan hasil probabilitasnya..

Tabel 4.3 Hasil pengujian model pada aplikasi dengan model 70:30

No	Data asli	Prediksi	Probabilitas
1	*Blight*	*Blight*	97%
2	*Blight*	*Blight*	99%
3	*Blight*	*Blight*	99%
4	*Blight*	*Blight*	81%
5	*Blight*	*Blight*	97%
6	*Blight*	*Blight*	97%
7	*Blight*	*Blight*	99%
8	*Common Rust*	*Common Rust*	99%
9	*Common Rust*	*Common Rust*	96%

Tabel 4.3 Hasil pengujian model pada aplikasi dengan model 70:30 (lanjutan)

10	*Common Rust*	*Common Rust*	93%
11	*Common Rust*	*Common Rust*	99%
12	*Common Rust*	*Common Rust*	99%
13	*Common Rust*	*Common Rust*	99%
14	*Common Rust*	*Common Rust*	97%
15	*Common Rust*	*Common Rust*	99%
16	*Gray Leaf Spot*	*Gray Leaf Spot*	100%
17	*Gray Leaf Spot*	*Gray Leaf Spot*	100%
18	*Gray Leaf Spot*	*Gray Leaf Spot*	99%
19	*Gray Leaf Spot*	*Gray Leaf Spot*	100%
20	*Gray Leaf Spot*	*Gray Leaf Spot*	100%
21	*Gray Leaf Spot*	*Gray Leaf Spot*	99%
22	*Gray Leaf Spot*	*Gray Leaf Spot*	99%
23	*Gray Leaf Spot*	*Gray Leaf Spot*	100%
24	*Healthy*	*Healthy*	54%
25	*Healthy*	*Healthy*	98%
26	*Healthy*	*Healthy*	93%
27	*Healthy*	*Healthy*	96%
28	*Healthy*	*Healthy*	67%
29	*Healthy*	*Healthy*	62%
30	*Healthy*	*Healthy*	92%

Pada data tabel 4.3 menunjukan hasil dari deteksi menggunakan model 70.30. Hasil daripada *split* data 70:30 menunjukan tingkat akurasi mencapai 100% dengan menggunakan 30 gambar acak.

$$akurasi = \frac{30}{30}\ x\ 100\%$$

$$akurasi = 100\%$$

Tabel 4.4 Hasil pengujian model pada aplikasi dengan model 80:20

No	**Data asli**	**Prediksi**	**Probabilitas**
1	*Blight*	*Blight*	99%
2	*Blight*	*Blight*	99%
3	*Blight*	*Blight*	99%
4	*Blight*	*Blight*	98%
5	*Blight*	*Blight*	99%
6	*Blight*	*Blight*	99%
7	*Blight*	*Blight*	99%
8	*Common Rust*	*Common Rust*	99%
9	*Common Rust*	*Common Rust*	98%
10	*Common Rust*	*Common Rust*	96%
11	*Common Rust*	*Common Rust*	97%
12	*Common Rust*	*Common Rust*	98%
13	*Common Rust*	*Common Rust*	98%
14	*Common Rust*	*Common Rust*	93%
15	*Common Rust*	*Common Rust*	99%

Tabel 4.4 Hasil pengujian model pada aplikasi dengan model 80:20 (lanjutan)

16	*Gray Leaf Spot*	*Gray Leaf Spot*	100%
17	*Gray Leaf Spot*	*Gray Leaf Spot*	100%
18	*Gray Leaf Spot*	*Gray Leaf Spot*	99%
19	*Gray Leaf Spot*	*Gray Leaf Spot*	100%
20	*Gray Leaf Spot*	*Gray Leaf Spot*	100%
21	*Gray Leaf Spot*	*Gray Leaf Spot*	100%
22	*Gray Leaf Spot*	*Gray Leaf Spot*	99%
23	*Gray Leaf Spot*	*Gray Leaf Spot*	100%
24	*Healthy*	*Common Rust*	82%
25	*Healthy*	*Healthy*	98%
26	*Healthy*	*Healthy*	91%
27	*Healthy*	*Healthy*	96%
28	*Healthy*	*Healthy*	73%
29	*Healthy*	*Healthy*	77%
30	*Healthy*	*Healthy*	92%

Pada data tabel 4.4 menunjukan hasil dari deteksi menggunakan model 80:20. Hasil daripada *split* data 80:20 menunjukan tingkat akurasi mencapai 100% dengan menggunakan 30 gambar acak.

$$akurasi = \frac{29}{30}\ x\ 100\%$$

$$akurasi = 96{,}66\%$$

4.10. Uji Coba Aplikasi Dengan Data Lapangan

Setelah selesai melakukan uji coba dengan *dataset* model yang digunakan, selanjutnya penulis akan menguji coba aplikasi dengan menggunakan data lapangan langsung yang telah diambil pada lahan pertanian jagung, dengan rincian pada gambar 3.3 – 3.5 pada proses *sampling*.

Pada proses pengambilan *sampling* menggunakan 3 *smartphone* yang memiliki spesifikasi berbeda-beda, dengan rincian pada tabel 4.5.

Tabel 4.5 Jenis *smartphone*

No	Jenis HP	Spesifikasi
1	Samsung A21s	• RAM : 3 GB • Memori : 32 GB • Kamera : 48 MP
2	Xiaomi Redmi 12	• RAM : 8 GB • Memori : 256 GB • Kamera : 50 MP

Tabel 4.5 Jenis *smartphone* (lanjutan)

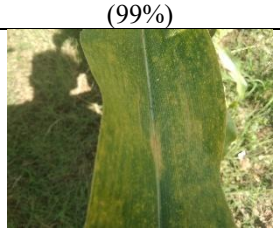
3	Xiaomi Redmi 5A	• RAM : 2 GB • Memori : 8 GB • Kamera : 13 MP

Dengan jenis *smartphone* yang berbeda, akan mencoba model apakah hasil yang dihasilkan akan sama atau diakibatkan oleh spesifikasi yang berbeda akan menyebabkan berbeda hasilnya. Pengujian akan menggunakan 2 model yaitu 70:30 dan 80:20. Hasil uji coba dapat dilihat pada tabel 4.6 dan 4.7.

Tabel 4.6 Pengujian 3 *smartphone* pada split data 70:30

No	Samsung A21s	Xiaomi Redmi 12	Xiaomi Redmi 5A
1	 Hasil : Gray Leaf Spot (99%)	 Hasil : Gray Leaf Spot (100%)	 Hasil : Gray Leaf Spot (99%)
2	 Hasil : Gray Leaf Spot (99%)	 Hasil : Gray Leaf Spot (99%)	 Hasil : Gray Leaf Spot (97%)
3	 Hasil : Gray Leaf Spot (65%)	 Hasil : Gray Leaf Spot (99%)	 Hasil : Blight (52%)
4	 Hasil : Gray Leaf Spot (99%)	 Hasil : Gray Leaf Spot (100%)	 Hasil : Gray Leaf Spot (97%)

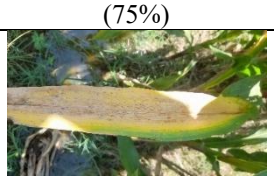
Tabel 4.6 Pengujian 3 *smartphone* pada split data 70:30 (lanjutan)

5	 Hasil : Gray Leaf Spot (96%)	 Hasil : Gray Leaf spot (92%)	 Hasil : Gray Leaf Spot (95%)
6	 Hasil : Gray Leaf Spot (99%)	 Hasil : Gray Leaf Spot (99%)	 Hasil : Gray Leaf Spot (99%)
7	 Hasil : Healthy (60%)	 Hasil : Gray Leaf Spot (88%)	 Hasil : Healthy (60%)
8	 Hasil : Blight (52%)	 Hasil : Blight (99%)	 Hasil : Blight (99%)

Tabel 4.7 Pengujian 3 *smartphone* pada split data 80:20

No	Samsung A21s	Xiaomi Redmi 12	Xiaomi Redmi 5A
1	 Hasil : Gray Leaf Spot (99%)	 Hasil : Gray Leaf Spot (100%)	 Hasil : Gray Leaf Spot (94%)
2	 Hasil : Gray Leaf Spot (99%)	 Hasil : Gray Leaf Spot (99%)	 Hasil : Gray Leaf Spot (58%)
3	 Hasil : Gray Leaf Spot (53%)	 Hasil : Gray Leaf Spot (99%)	 Hasil : Blight (53%)
4	 Hasil : Gray Leaf Spot (99%)	 Hasil : Gray Leaf Spot (99%)	 Hasil : Blight (87%)
5	 Hasil : Gray Leaf Spot (84%)	 Hasil : Gray Leaf spot (99%)	 Hasil : Gray Leaf Spot (99%)
6	 Hasil : Gray Leaf Spot (99%)	 Hasil : Gray Leaf Spot (99%)	 Hasil : Gray Leaf Spot (99%)

Tabel 4.7 Pengujian 3 *smartphone* pada split data 80:20 (lanjutan)

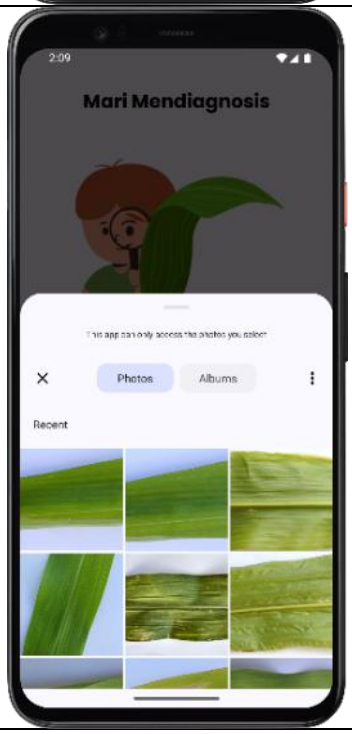
7	 Hasil : Common Rust (75%)	 Hasil : Gray Leaf Spot (57%)	 Hasil : Healthy (79%)
8	 Hasil : Blight (50%)	 Hasil : Blight (99%)	 Hasil : Blight (99%)

Pada tabel 4.6 dan 4.7 perbedaan *split data* 70:30 dan 80:20 menghasilkan hasil deteksi berbeda pada beberapa deteksi yang terjadi pada gambar nomor 3 pada *handphone* Redmi 5A yang pada data pengujian 70:30 mendeteksi *gray leaf spot* namun pada data pengujian 80:20 mendeteksi *blight* hal tersebut juga terjadi pada gambar no 4 dan 7. Selain itu perbedaan *split data* berpengaruh pada hasil probabilitas. Pada gambar nomor 7 terjadi perbedaaan pengambilan sudut gambar juga berpengaruh kepada hasil yang dari 3 tipe *handphone* memiliki 3 hasil yang berbeda.

4.11. Pengujian Aplikasi

Berikut ini merupakan pengujian aplikasi CornAnalyze yang dapat dilihat dalam tabel 4.8.

Tabel 4.8 Pengujian aplikasi

No	Pengujian	Deskripsi	Hasil	Status
1	Pengguna menekan tombol untuk membuka menu scan	Tampilan layar menu scan akan ditampilkan		sukses
2	Pengguna menekan tombol untuk membuka galeri	Sistem akan membuka galeri		Sukses

Tabel 4.8 Pengujian aplikasi (lanjutan)

3	Pengguna menekan tombol untuk membukakamera	Sistem akan membuka kamera		Sukses
4	Pengguna melakukan *crop* gambar	Sistem melakukan pemotongan gambar		Sukses
5	Pengguna melihat hasil *crop*	Sistem menampilkan hasil *crop*		Sukses

Tabel 4.8 Pengujian aplikasi (lanjutan)

6	Pengguna menekan tombol deteksi analisa 70:30	Sistem melakukan pendeteksian menggunakan *split* data 70:30		Sukses
7	Pengguna menekan tombol deteksi analisa 80:20	Sistem melakukan pendeteksian menggunakan *split* data 80:20		Sukses
8	Pengguna menyimpan hasil	Sistem menyimpan hasil		Sukses

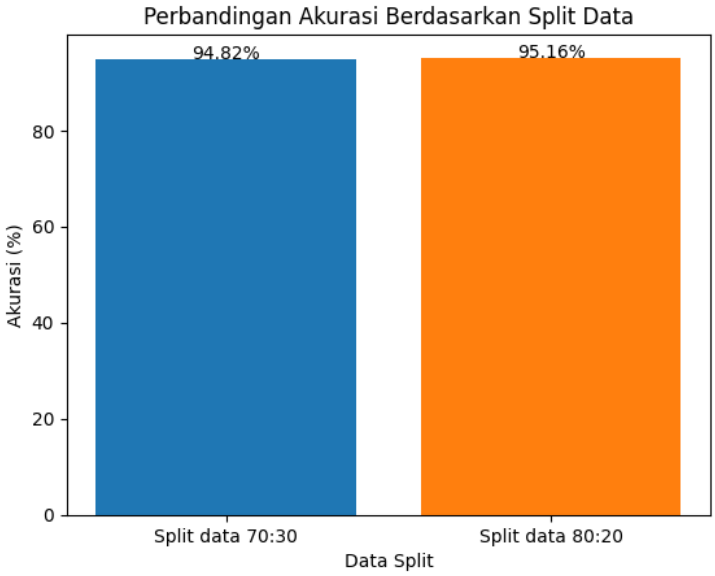
Tabel 4.8 Pengujian aplikasi (lanjutan)

9	Pengguna mengakses hasil yang sudah tersimpan	Sistem membuka hasil yang sebelumnya tersimpan		Sukses

4.12. Pembahasan

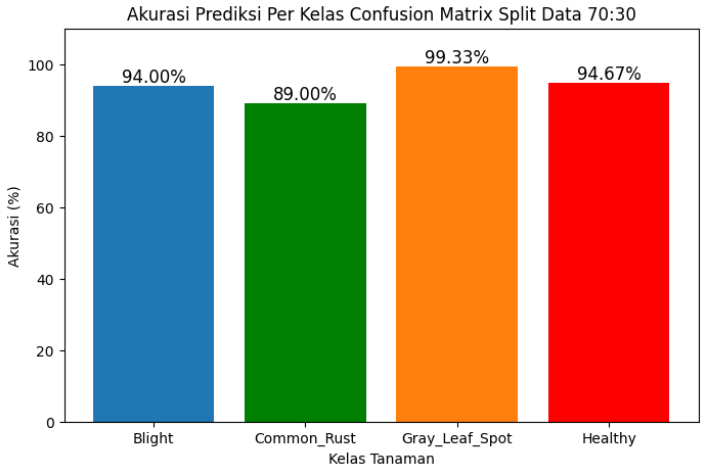
Total data yang digunakan dalam pembelajaran model yaitu 4000 gambar dengan 4 jenis yaitu 1 sehat dan 3 sakit, pada prosesnya menggunakan 2 tipe *split* data yaitu 70:30 dan 80:20 untuk mencari nilai akurasi yang tinggi dan nilai *loss* yang kecil, selain itu dalam prosesnya menggunakan pengecekan *F1Score*, *precision*, dan *recall*. Hal tersebut dilakukan untuk mengecek apakah model yang dilatih tersebut mengalami *overfitting* ataukah tidak dan memastikan model dapat digunakan dengan baik dalam melakukan pendeteksian penyakit tanaman jagung.

Pada penelitian ini, hasil akurasi yang dihasilkan pada kedua *split* data mencapai nilai 94.82% pada perbandingan 70:30 dan 95.16% pada perbandingan 80:20 yang dapat dilihat pada gambar 4.43.



Gambar 4.43 Perbandingan akurasi berdasarkan *split* data

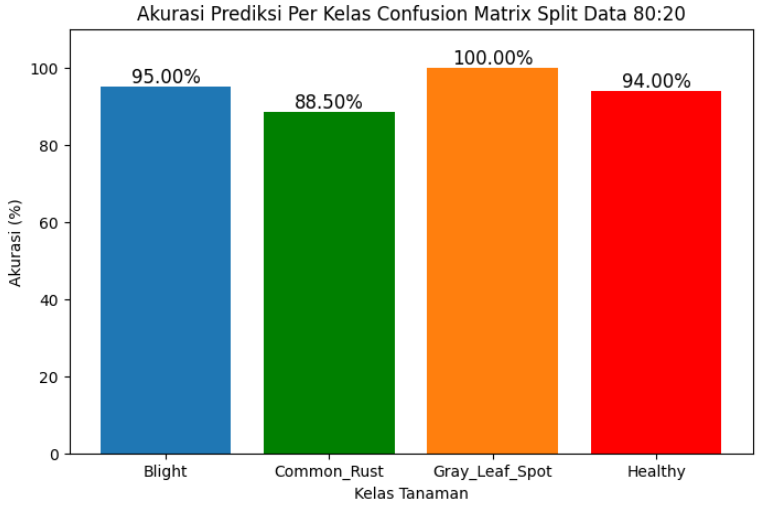
Pada akurasi prediksi pada *confusion matrix* pada *split data* 70:30 dengan data uji yaitu 300 gambar, nilai terbaik terdapat pada jenis penyakit *gray leaf spot* dengan nilai 99.33%, dan nilai lainnya yaitu *healthy* dengan nilai 94,67%, *blight* 94.00%, dan *common rust* 89.00%, dapat dilihat pada gambar 4.44.



Gambar 4.44 Akurasi prediksi *confusion matrix* 70:30

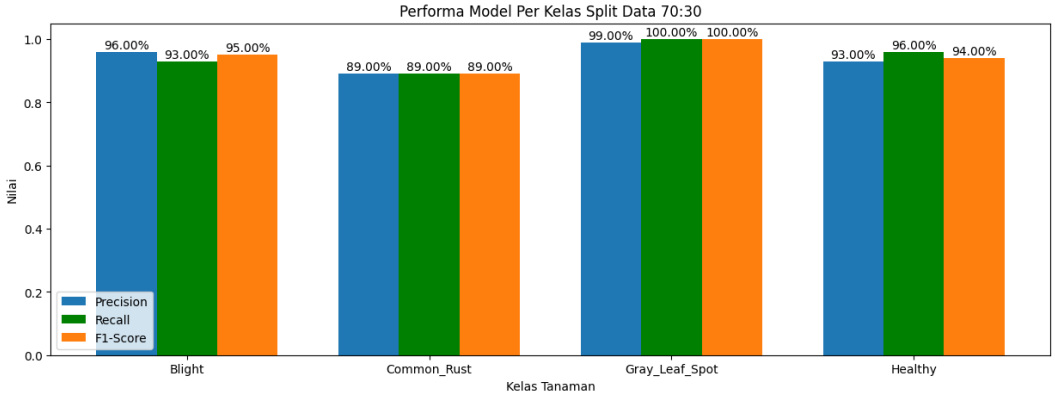
Pada akurasi prediksi pada *confusion matrix* pada *split* data 80:20 dengan data uji 200 gambar, nilai terbesar terdapat pada jenis penyakit *gray leaf spot* yaitu

100%, dan dilanjutkan dengan *blight* 95.00%, *healthy* 94.00%, dan *common rust* 88.50%, dapat dilihat pada gambar 4.45.



Gambar 4.45 Akurasi prediksi *confusion matrix* 80:20

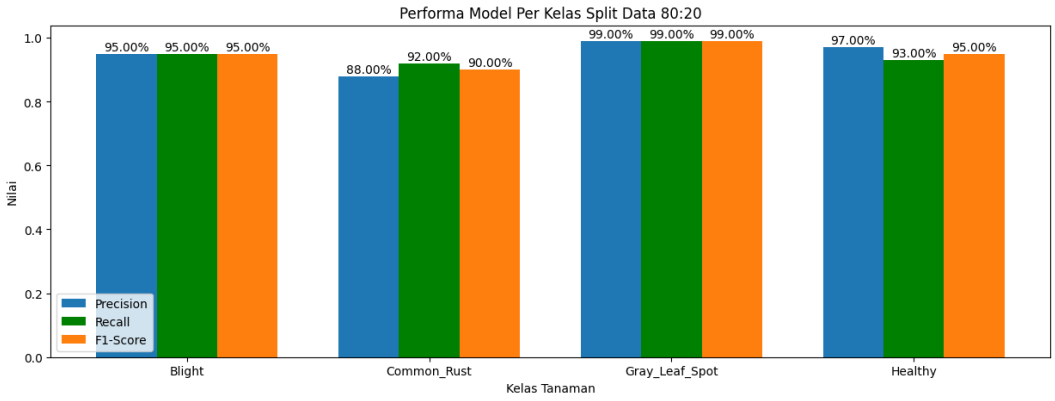
Pada *F1 Score split* data 70:30 dengan data uji 300 gambar pada masing-masing kelas, memiliki nilai 100% pada jenis penyakit *gray leaf spot* namun pada penyakit *common rust* memiliki nilai paling rendah yaitu 89%, namun pada nilai *precision* dan *recall* memiliki nilai yang seimbang pada semua jenis penyakit, yang berarti model cukup seimbang dalam performanya, dapat dilihat pada gambar 4.46.



Gambar 4.46 Performa model *split* data 70:30

Sedangkan pada *F1 Score split* data 80:20 dengan data uji 200 gambar pada masing-masing kelas, memiliki nilai 99% pada penyakit *gray leaf spot* namun pada

penyakit *common rust* memiliki nilai 90% yang lebih rendah dari kelas lainnya. Pada nilai *precision* dan *recall* memiliki nilai yang seimbang dari segi performa model, dapat dilihat pada gambar 4.47.



Gambar 4.47 Performa model *split* data 80:20

Dalam prosesnya model tersebut dirubah menjadi model *tensorflow lite* agar model tersebut dapat ditanam atau dimasukkan kedalam Android. Fitur yang disediakan pada aplikasi CornAnalyze yaitu :

1. Fitur deteksi penyakit pada daun jagung menggunakan kamera ataupun galeri.
2. Menyediakan cara penanganan dan penyebab setelah melakukan deteksi
3. Fitur penyimpanan hasil deteksi sehingga dapat dilihat kapan saja.
4. Memiliki artikel mengenai daun jagung.

Namun aplikasi CornAnalyze juga memiliki kekurangan yaitu :

1. Variasi hasil berdasarkan sudut pengambilan gambar menunjukkan bahwa model masih sensitif terhadap perubahan persepektif yang ditunjukan pada tabel 4.9 dan 4.10. Hal ini dapat disebabkan oleh keterbatasan jumlah sampel dalam dataset yang memiliki variasi sudut yang cukup beragam. Selain itu perbedaan hasil dapat dipengaruhi dari kualitas kamera masing-masing *smartphone*, resolusi, dan pencahayaan.

Tabel 4.9 Pengaruh pengambilan sudut gambar *split* data 70:30

Samsung A21s	Xiaomi Redmi 12	Xiaomi Redmi 5A
 Hasil : Healthy (60%)	 Hasil : Gray Leaf Spot (88%)	 Hasil : Healthy (60%)

Tabel 4.10 Pengaruh pengambilan sudut gambar *Split* data 80:20

Samsung A21s	Xiaomi Redmi 12	Xiaomi Redmi 5A
 Hasil : Common Rust (75%)	 Hasil : Gray Leaf Spot (57%)	 Hasil : Healthy (79%)

2. Penggunaan *split* data 70:30 dan 80:20 cukup berpengaruh kepada hasil yang memiliki perbedaan pada hasil probabilitas pada setiap gambar yang dideteksi.