

**KLASIFIKASI PENYAKIT DAUN TANAMAN JAGUNG
MENGGUNAKAN MODEL DEEP LEARNING EFFICIENTNET**

SKRIPSI



Oleh :

Rio Erfian

18.04.111.00040

Dosen Pembimbing 1 : Dr. Rima Tri Wahyuningrum, ST., MT.

Dosen Pembimbing 2 : Ari Kusumaningsih, ST., MT.

PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN TEKNIK INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS TRUNOJOYO MADURA

2023

HALAMAN JUDUL

KLASIFIKASI PENYAKIT DAUN TANAMAN JAGUNG MENGGUNAKAN MODEL *DEEP LEARNING EFFICIENTNET*

SKRIPSI

Diajukan untuk Memenuhi Persyaratan Penyelesaian
Studi Strata Satu (S1) dan Memperoleh Gelar Sarjana Komputer (S.Kom)
di Universitas Trunojoyo Madura

Rio Erfian

18.04.111.00040

PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN TEKNIK INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS TRUNOJOYO MADURA

2023

HALAMAN PENGESAHAN

KLASIFIKASI PENYAKIT DAUN TANAMAN JAGUNG MENGGUNAKAN MODEL DEEP LEARNING EFFICIENTNET

Skripsi ini disusun untuk memenuhi salah satu syarat memperoleh gelar Sarjana

Komputer (S.Kom) di Universitas Trunojoyo Madura

Oleh:

Nama : Rio Erfian

NIM 18.04.111.00040

Disetujui oleh Tim Pengaji Skripsi:

Tanggal Sidang

31 Maret 2023

Dr. Rima Tri Wahyuningrum, S.T, M.T
NIP. 19800820 200312 2 001

(Pembimbing I)

Ari Kusumaningsih, S.T., M.T.
NIP. 19790222 200501 2 003

(Pembimbing II)

Dr. Noor Ifada, S.T., M.ISD.
NIP. 19780317 200312 2 001

(Pengaji I)

Dr. Yeni Kustiyahningsih, S.Kom., M.Kom
NIP. 19770921 200812 2 002

(Pengaji II)

Dr. Fika Hastarita Rachman, S.T., M.Eng.
NIP. 19830305 200604 2 002

(Pengaji III)

Bangkalan, 24 Mei 2023

Mengefahui,

Ketua Jurusan Teknik Informatika

Dr. Yeni Kustiyahningsih, S.Kom., M.Kom.

NIP. 19770921 200812 2 002

HALAMAN PERNYATAAN ORISINALITAS

Saya yang bertanda tangan di bawah ini, menyatakan bahwa skripsi saya dengan judul:

“KLASIFIKASI PENYAKIT DAUN TANAMAN JAGUNG MENGGUNAKAN MODEL DEEP LEARNING EFFICIENTNET”

1. Adalah asli, bukan merupakan karya pihak lain serta belum pernah diajukan untuk mendapatkan gelar akademik Sarjana Komputer baik di Universitas Trunojoyo Madura maupun di Perguruan Tinggi yang lain di Indonesia.
2. Tidak terdapat karya atau pendapat pihak lain yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis telah diacu dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila dikemudian hari terbukti skripsi ini sebagian atau seluruhnya merupakan hasil plagiasi atau terdapat hal-hal yang tidak sesuai dengan pernyataan di atas, maka saya sanggup menerima sanksi akademis yang berlaku dengan segala akibat hukumnya sesuai peraturan Universitas Trunojoyo Madura dan atau peraturan perundang-undangan yang berlaku.

Bangkalan, 25 Mei 2023

Yang Menyatakan,

Rio Erfian

NIM. 180411100040

ABSTRAK

Jagung merupakan tanaman pangan utama setelah padi di Indonesia. Sehingga memiliki potensi besar di bidang pertanian. Seperti pada tanaman umumnya, tanaman ini juga memiliki penyakit yang beranekaragam dan dapat mengganggu kualitasnya. Gejala penyakit pada tanaman jagung dapat dilihat dari perubahan morfologi daun jagung. Metode *machine learning* klasik, di mana ekstraksi fitur manual harus sempurna untuk mencapai hasil yang optimal, kebutuhan akan model yang dapat melakukan klasifikasi dengan baik. Pada penelitian ini, model *deep learning EfficientNetB5* diajukan untuk mengklasifikasi penyakit daun tanaman jagung dan kinerja model dibandingkan antara *hyperparameter learning rate* dan *batch size*. Semua model di *training* dengan dataset publik yang diperoleh dari mendeley data dengan jumlah data sebanyak 3.852 citra yang terbagi menjadi 4 kelas yaitu, sehat (*healthy*) berjumlah 1.162, penyakit karat daun (*common rust*) berjumlah 1.192, penyakit daun hawar utara (*northern leaf blight*) berjumlah 985, dan penyakit bercak abu-abu (*cercospora grey leaf spot*) berjumlah 513. Kemudian dilakukan proses validasi model menggunakan *5-cross validation* untuk mendapatkan hasil model terbaik. Hasil yang diperoleh dalam pengujian dataset menunjukkan bahwa kombinasi *learning rate* = 0,0001 dan *batch size* = 32 mencapai nilai tertinggi dibandingkan dengan model lainnya. Nilai evaluasi yang diperoleh seperti, akurasi 96,27%, presisi 90,90%, spesifisitas 97,55%, dan sensitivitas 88,13%.

Kata kunci: Klasifikasi Citra, Penyakit daun jagung, *deep learning*, *EfficientNet*, *learning rate*, *batch size*.

KATA PENGANTAR

Puji syukur penulis kehadirat Allah SWT dengan rahmat, taufik, dan hidayah-Nya penulis mampu menyelesaikan Skripsi ini dengan judul “Klasifikasi Penyakit Daun Tanaman Jagung Menggunakan Model Deep learning EfficientNet”. Tidak lupa shalawat serta salam semoga terus tercurahkan kepada Nabi Muhammad SAW agar kita mendapatkan syafaat di hari akhir. Dalam proses penyelesaian skripsi, banyak pihak yang telah membantu penulis dalam penyusunan laporan ini, baik berupa bantuan materi ataupun berupa motivasi dan dukungan kepada penulis. Dengan terselesaikannya laporan skripsi ini, penulis tidak lupa menyampaikan terima kasih yang sebesar-besarnya kepada:

1. Ibu Dr. Rima Tri Wahyuningrum, S.T., M.T. selaku dosen pembimbing I dan Ibu Ari Kusumaningsih, S.T., M.T. selaku dosen pembimbing II yang selalu meluangkan waktu, memberikan nasehat selama perkuliahan, memberikan ilmu dan saran, serta sabar dalam membimbing penulis hingga skripsi ini selesai.
2. Ibu Dr. Noor Ifada, ST., M.ISD selaku Dosen Pengaji I, Ibu Dr. Yeni Kustyaningsih, S.Kom., M.Kom selaku Dosen Pengaji II, dan Ibu Dr. Fika Hastarita rachman, ST., M.Eng selaku Dosen Pengaji III yang telah banyak memberikan saran demi kemajuan dan penyempurnaan dalam penggerjaan Skripsi ini.
3. Seluruh Bapak dan Ibu Dosen Program Studi Teknik Informatika Universitas Trunojoyo Madura yang telah semangat dalam mendidik dengan ilmu pengetahuan yang bermanfaat.
4. Kedua orang tua penulis yang telah selalu memberikan do'a, kekuatan, motivasi, dukungan, nasihat-nasihat berharga dan seluruh kebaikan yang selalu diberikan sehingga telah sampai pada tahap ini.
5. Teman – teman yang selalu memberikan motivasi, memberikan bantuan dan dukungan moral selama proses penulisan dan penyusunan laporan skripsi.
6. Seluruh pihak yang belum penulis sebutkan, terima kasih atas dukungan baik material maupun spiritual.

Penulis juga ingin berterima kasih kepada para pembaca Skripsi dan menyadari bahwa penulisan skripsi ini tidak luput dari kesalahan. Oleh karena itu, penulis menyatakan permohonan maaf atas segala kekurangan yang ada. Dengan kerendahan hati, berbagai saran dan kritik yang membangun penulis sangat harapkan dari rekan-rekan pembaca.

Gresik, 25 Mei 2023

Penulis

DAFTAR ISI

HALAMAN JUDUL.....	ii
HALAMAN PENGESAHAN.....	iii
HALAMAN PERNYATAAN ORISINALITAS.....	iv
ABSTRAK	v
KATA PENGANTAR	vi
DAFTAR ISI.....	viii
DAFTAR GAMBAR	xi
DAFTAR TABEL.....	xiii
DAFTAR KODE PROGRAM.....	xv
DAFTAR PERSAMAAN	xvi
BAB 1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	3
1.2.1 Permasalahan	3
1.2.2 Solusi Permasalahan	3
1.2.3 Pertanyaan Penelitian.....	3
1.3 Tujuan dan Manfaat.....	3
1.3.1 Tujuan	3
1.3.2 Manfaat	4
1.4 Batasan Masalah.....	4
1.5 Sistematika Laporan	5
BAB 2 KAJIAN PUSTAKA.....	6
2.1 Klasifikasi Citra.....	6
2.2 <i>Deep Learning</i>	6
2.3 <i>Convolutional Neural Network</i>	7
2.3.1 <i>Convolution Layer</i>	8
2.3.2 <i>DepthWise Convolution</i>	9
2.3.3 <i>Pooling Layer</i>	9
2.3.4 <i>Activation Function</i>	10
2.3.5 <i>Batch Normalization</i>	11
2.3.6 <i>Fully Connected Layer</i>	11
2.3.7 <i>Global Average Pooling</i>	11
2.4 <i>EfficientNet</i>	12

2.4.1	<i>Compound Scalling</i>	13
2.4.2	Arsitektur	14
2.5	<i>Hyperparameter</i>	15
2.5.1	<i>Batch size</i>	15
2.5.2	<i>Learning rate</i>	15
2.6	<i>K-Fold Cross Validation</i>	16
2.7	Metrik Evaluasi	16
2.8	<i>Receiver Operating Characteristic (ROC) Curve</i>	18
2.9	Penelitian terkait.....	19
BAB 3	METODE USULAN.....	22
3.1	Dataset	22
3.2	Pembagian Dataset	23
3.3	Arsitektur Sistem <i>training</i> dan <i>testing</i>	23
3.4	<i>Pre-processing</i> Citra.....	24
3.5	Proses <i>Training</i>	26
3.6	Perhitungan Proses Algoritma <i>EfficientNetB5</i>	26
3.7	Proses <i>Testing</i>	31
3.8	Evaluasi Model.....	31
3.9	Skenario Ujicoba	33
BAB 4	HASIL DAN PEMBAHASAN.....	35
4.1	Lingkungan Uji Coba	35
4.2	Tahap-Tahap Pembuatan Program	36
4.2.1	Mendefinisikan nilai parameter	36
4.2.2	Melakukan <i>Preprocessing</i>	37
4.2.3	Pembagian Dataset <i>Training</i> dan <i>Testing</i>	38
4.2.4	Pembuatan Model <i>EfficientNet</i>	39
4.2.5	<i>Training</i> data.....	40
4.2.6	Pembuatan <i>Confusion matrix</i>	41
4.2.7	<i>Testing</i> dan Evaluasi Model.....	42
4.3	Hasil Skenario Uji Coba	45
4.3.1	Skenario Uji Coba 1	45
4.3.2	Skenario Uji Coba 2	47
4.3.3	Skenario Uji Coba 3	48
4.3.4	Skenario Uji Coba 4	50
4.3.5	Skenario Uji Coba 5	51
4.3.6	Skenario Uji Coba 6	53

4.4	Analisa Hasil Uji Coba.....	54
BAB 5	PENUTUP.....	64
5.1	Kesimpulan.....	64
5.2	Saran	64
REFERENSI		65
DAFTAR RIWAYAT HIDUP.....		68

DAFTAR GAMBAR

Gambar 2.1 Skema <i>deep learning</i>	6
Gambar 2.2 Ilustrasi CNN	7
Gambar 2.3 Operasi <i>convolution layer</i>	8
Gambar 2.4 <i>Pooling Layer</i>	9
Gambar 2.5 <i>ImageNet</i> akurasi vs <i>Model size</i>	12
Gambar 2.6 Penskalaan model.....	13
Gambar 2.7 Baseline <i>EfficientNet B0</i> [10].....	14
Gambar 2.8 Skema arsitektur <i>EfficientNetB5</i>	14
Gambar 2.9 <i>5-fold cross validation</i>	16
Gambar 2.10 <i>Multi-class confusion matrix</i> kelas 0.....	17
Gambar 2.11 Kurva ROC.....	18
Gambar 3.1 Contoh data citra jagung	22
Gambar 3.2 <i>Block diagram training</i> dan <i>testing</i>	23
Gambar 3.3 Proses <i>training</i>	26
Gambar 3.4 Ilustrasi <i>Block MBConv1</i> [24]	27
Gambar 3.5 Ilustrasi <i>Block SE</i> [24].....	28
Gambar 3.6 Illustrasi <i>Block MBConv6</i> [24]	30
Gambar 3.7 Proses <i>testing</i>	31
Gambar 4.1 <i>Confusion matrix</i> skenario 1 (<i>fold 4</i>)	46
Gambar 4.2 <i>Confusion matrix</i> skenario 2 (<i>fold 4</i>)	48
Gambar 4.3 <i>Confusion matrix</i> skenario 3 (<i>fold 4</i>)	49
Gambar 4.4 <i>Confusion matrix</i> skenario 4 (<i>fold 4</i>)	51
Gambar 4.5 <i>Confusion matrix</i> skenario 5 (<i>fold 4</i>)	52
Gambar 4.6 <i>Confusion matrix</i> skenario 6 (<i>fold 4</i>)	54
Gambar 4.7 Komparasi akurasi hasil uji coba	56
Gambar 4.8 Komparasi hasil rata-rata evaluasi model	56
Gambar 4.9 Contoh data kesalahan klasifikasi	59
Gambar 4.10 Kurva ROC skenario 1	60
Gambar 4.11 Kurva ROC skenario 2	61
Gambar 4.12 Kurva ROC skenario 3	61

Gambar 4.13 Kurva ROC skenario 4	62
Gambar 4.14 Kurva ROC skenario 5	62
Gambar 4.15 Kurva ROC skenario 6	63

DAFTAR TABEL

Tabel 2.1 Penelitian Terkait	20
Tabel 3.1 Penjabaran Dataset	22
Tabel 3.2 Pembagian Dataset	23
Tabel 3.3 Data fitur citra Asli 12x12	25
Tabel 3.4 Data fitur citra normalisasi 12x12.....	25
Tabel 3.5 Contoh Perhitungan <i>Convolution layer</i>	26
Tabel 3.6 Data fitur hasil <i>DepthWiseConv</i>	27
Tabel 3.7 Data Hasil <i>Batch Normalization</i>	28
Tabel 3.8 Data Hasil Aktivasi <i>Swish</i>	28
Tabel 3.9 Contoh Bobot	29
Tabel 3.10 Data hasil <i>Reshape</i>	29
Tabel 3.11 Data hasil <i>convolution 3x3</i>	30
Tabel 3.12 Data fitur hasil <i>Batch Normalization</i>	30
Tabel 3.13 Contoh Perbandingan data target dan hasil prediksi	32
Tabel 3.14 <i>Confusion matrix</i>	32
Tabel 3.15 Hasil perhitungan tiap kelas	32
Tabel 3.16 Hasil Evaluasi metrik	33
Tabel 3.17 Skenario Ujicoba.....	33
Tabel 4.1 Spesifikasi perangkat	35
Tabel 4.2 <i>Library</i> yang digunakan	35
Tabel 4.3 Hasil Akurasi tiap kelas pada skenario 1	45
Tabel 4.4 Hasil perhitungan metrik pengukuran skenario 1	46
Tabel 4.5 Hasil Akurasi tiap kelas pada skenario 2	47
Tabel 4.6 Hasil perhitungan metrik pengukuran skenario 2	47
Tabel 4.7 Hasil Akurasi tiap kelas pada skenario 3	48
Tabel 4.8 Hasil perhitungan metrik pengukuran skenario 3	49
Tabel 4.9 Hasil Akurasi tiap kelas pada skenario 4	50
Tabel 4.10 Hasil perhitungan metrik pengukuran skenario 4	50
Tabel 4.11 Hasil Akurasi tiap kelas pada skenario 5	51
Tabel 4.12 Hasil perhitungan metrik pengukuran skenario 5	52

Tabel 4.13 Hasil Akurasi tiap kelas pada skenario 6	53
Tabel 4.14 Hasil perhitungan metrik pengukuran skenario 6	53
Tabel 4.15 Rangkuman Hasil Akurasi Uji coba.....	54
Tabel 4.16 Hasil rata-rata evaluasi model setiap skenario.....	55
Tabel 4.17 Kinerja klasifikasi skenario 5 pada setiap kelas (<i>fold 1</i>).....	57
Tabel 4.18 Kinerja klasifikasi skenario 5 pada setiap kelas (<i>fold 2</i>).....	57
Tabel 4.19 Kinerja klasifikasi skenario 5 pada setiap kelas (<i>fold 3</i>).....	57
Tabel 4.20 Kinerja klasifikasi skenario 5 pada setiap kelas (<i>fold 4</i>).....	57
Tabel 4.21 Kinerja klasifikasi skenario 5 pada setiap kelas (<i>fold 5</i>).....	58
Tabel 4.22 Karakteristik data setiap kelas.....	58

DAFTAR KODE PROGRAM

Kode Program 4.1 Mendefinisikan parameter	36
Kode Program 4.2 <i>Preprocessing</i>	37
Kode Program 4.3 <i>k-Fold Cross Validation</i>	38
Kode Program 4.4 Model <i>EfficientNet</i>	39
Kode Program 4.5 <i>Training</i> data.....	40
Kode Program 4.6 <i>Plotting confusion matrix</i>	42
Kode Program 4.7 <i>Testing</i> dan evaluasi model.....	44

DAFTAR PERSAMAAN

2.1 Persamaan Konvolusi.....	8
2.2 Persamaan ReLU.....	10
2.3 Persamaan SiLU.....	10
2.4 Persamaan <i>Softmax</i>	10
2.5 Persamaan <i>Batch Normalization</i>	11
2.6 Persamaan <i>Fully Conneted Layer</i>	11
2.7 Persamaan <i>Coumpound Scalling</i>	13
2.8 Persamaan Akurasi.....	17
2.9 Persamaan Sensitivitas	17
2.10 Persamaan Spesifisitas	17
2.11 Persamaan Presisi.....	17
2.12 Persamaan Rata-rata Makro Metrik Evaluasi	18

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Indonesia merupakan salah satu negara yang memiliki potensi besar di bidang pertanian. Namun, seringkali banyak terjadi kerugian pada hasil panen akibat penyakit pada tanaman, salah satunya terjadi pada tanaman jagung. Jagung merupakan tanaman pangan utama kedua setelah padi di Indonesia dan menempati posisi ketiga setelah padi dan terigu di dunia [1]. Akan tetapi semakin berkembang pesatnya industri peternakan, jagung sebagai komponen utama dalam ransum pakan dengan persentase sebesar 60%. Kebutuhan jagung nasional diperkirakan lebih 55% diperuntukkan sebagai pakan, dan 30% sebagai konsumsi pangan, lalu sisanya sebagai kebutuhan industri dan benih[2].

Permasalahan yang sering terjadi sebagaimana tanaman pada umumnya, tanaman jagung tidak terlepas dari ancaman serangan penyakit. Sebagian besar penyakit menunjukkan gejala yang terlihat pada perubahan daunnya. Adapun penyakit pada daun jagung diantaranya daun hawar utara, daun bercak abu-abu, dan karat daun. Metode klasifikasi tradisional, seperti observasi mata telanjang dan tes laboratorium, memiliki banyak keterbatasan, seperti memakan waktu dan subjektif [3]. Kemajuan dalam teknologi kecerdasan buatan telah membuka jalan bagi pengembangan sistem otomatis yang dapat memperoleh hasil yang lebih cepat dan akurat dalam mendiagnosis penyakit. Salah satu bidang kecerdasan buatan adalah *machine learning*. Pada tahun 2015 Zhang Z, *et al.* [4] mengusulkan metode GA-SVM (*Genetic Algorithm Support Vector Machine*) mengkombinasikan algoritma evolusi dan algoritma *machine learning* untuk mengklasifikasikan penyakit daun tanaman jagung. Namun, saat ini model *deep learning* terutama yang berbasis *convolutional neural network* (CNN) lebih diunggulkan daripada metode *machine learning*. Karena CNN dalam belajar secara bersamaan mengekstrak fitur dan mengklasifikasikan data sampel. Karena itu, CNN tidak memerlukan proses ekstraksi fitur khusus dan pada umumnya hanya memerlukan *preprocessing* dasar untuk menormalisasi data[5].

Penelitian terkait klasifikasi penyakit daun tanaman jagung menggunakan model *deep learning* banyak diterapkan seperti pada tahun 2019, Sibya & Sumbwanyambe[6] menggunakan model *deep learning* Arsitektur CNN untuk mengklasifikasi 3 jenis penyakit daun tanaman jagung. Pada tahun 2020, Syarief & Setiawan[7] menganalisis klasifikasi citra penyakit daun jagung dengan 2 langkah: Ekstraksi fitur menggunakan 7 model *deep learning* (CNN) berbeda dan klasifikasi menggunakan 3 metode *machine learning*. Di tahun yang sama, Waheed A, et al. [8] mengusulkan arsitektur *Dense convolution neural network* (DenseNet) yang dioptimalkan untuk mengenali dan mengklasifikasi tiga penyakit daun jagung. Pada tahun berikutnya, Atila U, et al [9] mengklasifikasikan 39 jenis penyakit tanaman dengan membandingkan keluarga model *EfficientNets* dengan beberapa model *deep learning* lain seperti, *Alexnet*, *ResNet50*, *VGG16*, dan *InceptionV3*. Hasil yang diperoleh model *EfficientNetB5* terbukti mendapatkan akurasi lebih baik dari model-model tersebut.

EfficientNet mengungguli pada klasifikasi *ImageNet* dengan akurasi mencapai 84,4% dibanding dengan model lainnya seperti, *ResNet*, *Gpipe*, *DenseNet*, dan lain-lain [10]. Proses penskalaan CNN saat ini ada banyak cara untuk melakukannya. Cara yang paling umum adalah dengan meningkatkan CNN berdasarkan kedalamannya (*depth*) [11] atau lebarnya (*width*) [12]. Metode lain yang kurang umum, tetapi semakin populer, adalah meningkatkan model berdasarkan resolusi citra. *EfficientNet* menyeimbangkan semua dimensi lebar, kedalaman, dan resolusi menggunakan penskalaan majemuk (*coumpound scaling*)[10].

Dalam melatih model *deep learning* agar dapat mengklasifikasikan citra dengan baik, banyak *hyperparameter* perlu disesuaikan seperti, *batch size* dan *learning rate*. Penyesuaian *hyperparameter* diperlukan untuk mendapatkan akurasi yang optimal dengan menentukan nilai *batch size* dan *learning rate* yang baik untuk diterapkan pada model pembelajaran[13][14]. Dalam penelitian ini, Model *EfficientNetB5* diterapkan dalam klasifikasi penyakit daun tanaman jagung dengan penentuan *hyperparameter batch size* dan *learning rate* untuk mendapatkan akurasi yang optimal.

1.2 Rumusan Masalah

1.2.1 Permasalahan

Dalam mengklasifikasikan penyakit pada daun tanaman jagung banyak metode di gunakan, seperti metode *machine learning* dan *deep learning*. Akan tetapi model *deep learning* memiliki kinerja lebih baik. Setiap *training* model dibutuhkan pengoptimalan untuk menentukan seberapa baik kinerja jaringan. Bagaimana mengembangkan model yang memiliki kinerja yang baik dalam melakukan klasifikasi penyakit pada daun tanaman jagung dengan menentukan nilai *hyperparameter* tepat dalam arsitektur *EfficientNetB5*.

1.2.2 Solusi Permasalahan

Berdasarkan permasalahan yang telah diuraikan, Solusi yang digunakan untuk mengklasifikasi penyakit pada tanaman jagung menggunakan model *deep learning* *EfficientNetB5* dengan menentukan nilai optimal untuk *hyperparamater learning rate* dan *batch size*.

1.2.3 Pertanyaan Penelitian

Berapa besar tingkat akurasi pada setiap kombinasi *learning rate* dan *batch size* yang menghasilkan kinerja optimal dalam melakukan klasifikasi penyakit daun tanaman jagung dengan mengimplementasikan model *EfficientNetB5*?

1.3 Tujuan dan Manfaat

1.3.1 Tujuan

Mengetahui tingkat akurasi, presisi, sensitivitas, dan spesifisitas yang didapatkan dari hasil klasifikasi penyakit daun tanaman jagung menggunakan arsitektur EfficientNetB5 dengan kombinasi *hyperparameter learning rate* dan *batch size*.

1.3.2 Manfaat

Manfaat yang diharapkan dari pengimplementasian *EfficientNetB5* untuk identifikasi penyakit daun jagung adalah sebagai alat bantu untuk mempermudah dan mempercepat proses diagnosis penyakit tanaman jagung.

1.4 Batasan Masalah

Batasan-batasan masalah pada penelitian ini adalah sebagai berikut.

1. Implementasi model arsitektur *EfficientNetB5*.
2. Dataset yang digunakan adalah citra daun jagung yang diperoleh dari situs <https://data.mendeley.com/datasets/tywbtsjrjv/1>.
3. Data citra pada dataset sebanyak 3.852 citra dengan ukuran 256 x 256 piksel berekstensi jpg.
4. Klasifikasi citra dibagi menjadi 4 jenis, yaitu daun sehat (*healthy*), daun berkarat (*common rust*), daun hawar utara (*northern leaf blight*), dan daun bercak abu-abu (*cercospora leaf spot*).
5. Pemodelan yang dibandingkan sebagai berikut:
 - a. Klasifikasi citra dengan menerapkan arsitektur *EfficientNetB5* dengan nilai *hyperparameter batch size* = 16 & *learning rate* = 0,001.
 - b. Klasifikasi citra dengan menerapkan arsitektur *EfficientNetB5* dengan nilai *hyperparameter batch size* = 16 & *learning rate* = 0,0001.
 - c. Klasifikasi citra dengan menerapkan arsitektur *EfficientNetB5* dengan nilai *hyperparameter batch size* = 16 & *learning rate* = 0,00001.
 - d. Klasifikasi citra dengan menerapkan arsitektur *EfficientNetB5* dengan nilai *hyperparameter batch size* = 32 & *learning rate* = 0,001
 - e. Klasifikasi citra dengan menerapkan arsitektur *EfficientNetB5* dengan nilai *hyperparameter batch size* = 32 & *learning rate* = 0,0001.
 - f. Klasifikasi citra dengan menerapkan arsitektur *EfficientNetB5* dengan nilai *hyperparameter batch size* = 32 & *learning rate* = 0,00001.

1.5 Sistematika Laporan

Sistematika penulisan laporan skripsi ini adalah sebagai berikut:

BAB I PENDAHULUAN

Bab ini menjelaskan mengenai ruang lingkup masalah yang dibahas, usulan solusi untuk dapat menyelesaikan masalah, tujuan dan manfaat penerapan solusi yang diajukan, serta sistematika penulisan laporan.

BAB II KAJIAN PUSTAKA

Bab ini menjelaskan teori – teori pendukung yang dapat dijadikan bahan acuan untuk dapat memahami masalah yang akan dibahas pada penelitian ini. Teori – teori tersebut berhubungan dengan penyakit daun jagung, arsitektur *EfficientNet* untuk klasifikasi citra.

BAB III METODE USULAN

Bab ini menjelaskan mengenai metode penelitian yang akan dilakukan, yaitu mengenai dataset yang digunakan, arsitektur sistem, serta skenario yang akan di uji coba.

BAB IV HASIL DAN PEMBAHASAN

Bab ini berisikan tentang hasil dari implementasi dari sistem menggunakan arsitektur *EfficientNetB5* yang terdiri dari lingkungan uji coba, skenario uji coba, hasil uji coba, serta analisis hasil uji coba.

BAB V PENUTUP

Bab ini berisi kesimpulan dari penelitian yang disesuaikan dengan pertanyaan penelitian dan saran terkait pengembangan aplikasi selanjutnya.

BAB 2 KAJIAN PUSTAKA

2.1 Klasifikasi Citra

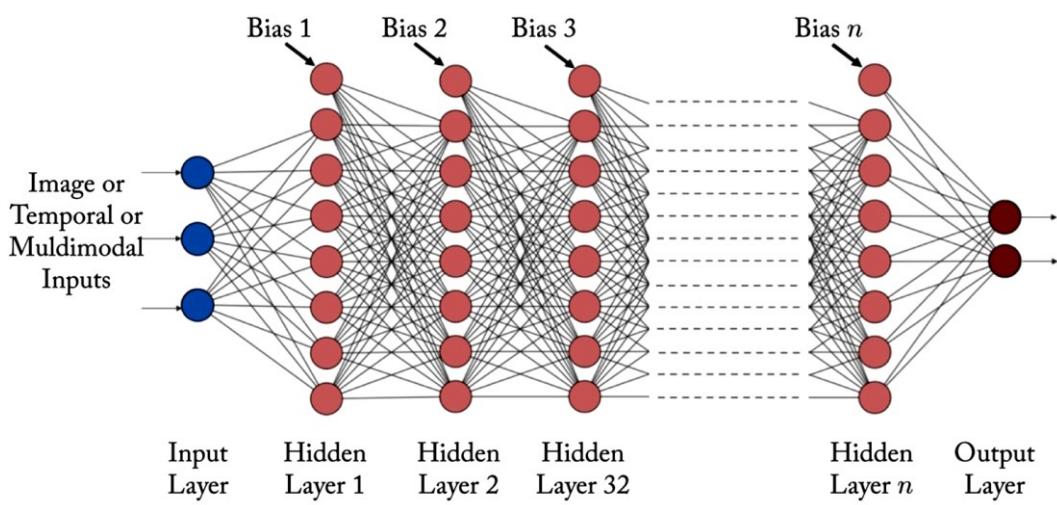
Klasifikasi citra adalah proses pengelompokan piksel pada citra ke dalam sejumlah kelas dengan keterbatasan masing-masing sehingga setiap kelas memiliki ciri atau label khusus yang merepresentasikan kelasnya [15]. Jenis klasifikasi terbagi menjadi 2, yaitu:

1. Klasifikasi Biner, adalah jenis klasifikasi hanya dengan 2 kelas dan *output* yang dihasilkan berupa *single label* (satu label).
2. Klasifikasi multi kelas, adalah jenis klasifikasi dengan jumlah kelas lebih dari dua dan *output* yang dihasilkan dapat berupa *single label* atau *multi label* (banyak label).

Pada penelitian ini klasifikasi citra yang akan digunakan yakni klasifikasi multi kelas dengan *output* berupa *single-label*.

2.2 Deep Learning

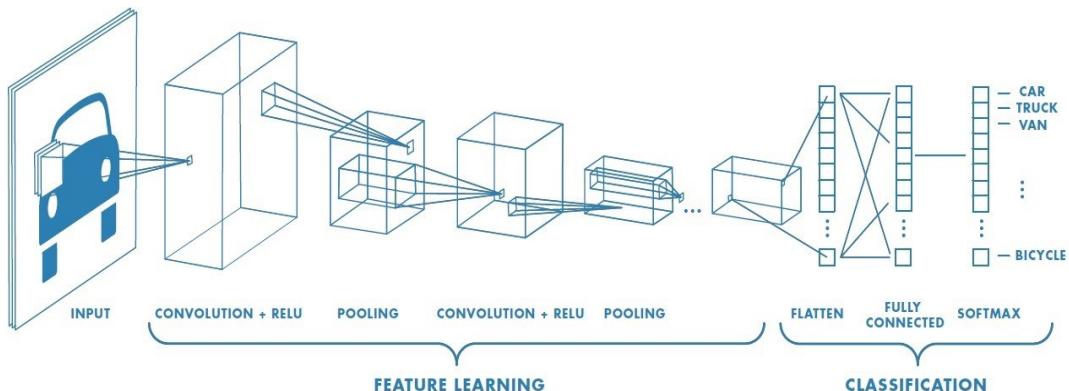
Deep learning atau pembelajaran mendalam adalah arsitektur jaringan saraf komputasional yang mengandung banyak lapisan tersembunyi. Karena *deep learning* adalah jaringan saraf yang lebih besar, dan merupakan sub bidang dari *machine learning* dan *artificial intelligence*.



Gambar 2.1 Skema *deep learning*

Deep learning adalah alat yang sangat kuat untuk pengenalan suara, mengklasifikasikan objek, dan pengenalan pola. Menggunakan banyak lapisan tersembunyi model *deep learning* belajar dari kumpulan data besar dengan berbagai tingkat abstraksi. Gambar 2.1 menunjukkan skema model pembelajaran mendalam dengan n lapisan tersembunyi dan bias, di mana setiap neuron dalam lapisan terhubung ke setiap neuron dari lapisan berikutnya [16]. Banyak arsitektur pembelajaran mendalam seperti AlexNet, ResNet, GoogleNet, DenseNet, dan *EfficientNet*. Namun, kinerja arsitektur ini tergantung pada banyaknya lapisan. Memiliki terlalu banyak lapisan tersembunyi tidak selalu efektif, dan itu harus dipilih berdasarkan jenis dataset *training* dan persyaratan output.

2.3 Convolutional Neural Network



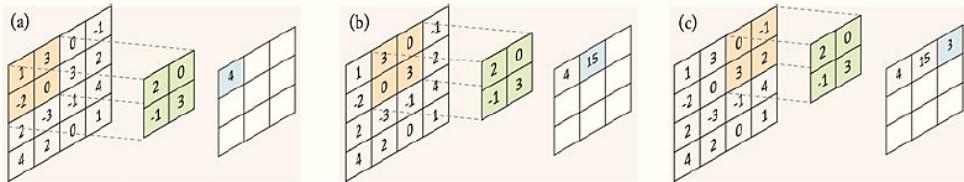
Gambar 2.2 Illustrasi CNN

Convolutional Neural Network (CNN) merupakan hasil pengembangan dari *Multi layer Perceptron* (MLP) yang dirancang untuk memproses data yang berdimensi dua seperti citra. Karena kedalaman jaringan yang tinggi, CNN termasuk dalam jenis *Deep Neural Network* dan sering diaplikasikan pada data citra [10]. CNN terdiri dari 2 bagian utama: ekstraksi fitur dan klasifikasi. Bagian ekstraksi fitur termasuk *input layer*, *convolution layer* dengan *stride* dan *padding*, *rectified linear unit* (ReLU), *pooling layer*, dan *batch normalization layer*. Sedangkan bagian klasifikasi terdiri dari *fully connected layer*, *softmax* dan *output layer*[7]. Illustrasi arsitektur CNN dapat dilihat pada Gambar 2.2.

2.3.1 Convolution Layer

Convolutional layer adalah komponen terpenting dari CNN. Ini terdiri dari satu set filter yang dikonvolusi dengan input yang diberikan untuk menghasilkan output berupa *feature map*[5]. *Convolution layer* memiliki 3 parameter yang digunakan untuk mengatur *output* dari konvolusi, yaitu:

1. *Filter/Kernel*, adalah matriks yang nilainya pada proses awal diperoleh secara random dan dikalikan dengan input citra untuk menghasilkan *feature map*.
2. *Stride*, digunakan untuk menentukan seberapa jauh filter bergeser secara vertikal dan horizontal dalam setiap kali proses konvolusi.
3. *Padding*, mengacu pada jumlah piksel yang ditambahkan ke gambar saat kernel memprosesnya.



Gambar 2.3 Operasi *convolution layer*

Gambar 2.3 menunjukkan perhitungan yang dilakukan pada setiap langkah, saat *filter* digeser ke *input feature map* untuk menghitung nilai yang sesuai dalam *output feature map*. Filter 2x2 (ditunjukkan dengan warna hijau) dikalikan dengan wilayah yang berukuran sama pada *input feature map* 4x4 (ditunjukkan dengan warna oranye) dan nilai yang dihasilkan dijumlahkan untuk mendapatkan nilai yang sesuai (ditunjukkan dengan warna biru) di *output feature map* pada setiap langkah konvolusi [5].

$$a_{ij} = (w * x)_{ij} + b \quad 2.1$$

a_{ij} = hasil perhitungan operasi konvolusi (i, j)

w = matriks *filter/kernel*

x = nilai setiap piksel dari citra

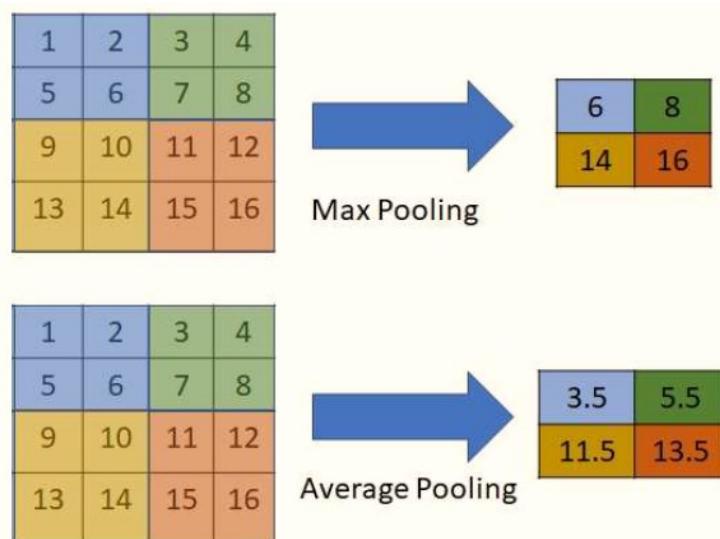
b = nilai bias

2.3.2 DepthWise Convolution

Depthwise convolution adalah memecah filter dan citra di *channel* yang berbeda dan kemudian menggabungkan piksel citra dengan *channel* dan kemudian menumpuknya kembali. *Depthwise convolution* digunakan untuk mengurangi biaya komputasi dan ruang paramater. Perbedaan utama antara *convolution layer* dan *depthwise convolution* adalah bahwa konvolusi kedalaman menerapkan konvolusi hanya pada satu dimensi spasial (yaitu *channel*) sedangkan konvolusi normal diterapkan pada semua dimensi spasial.

2.3.3 Pooling Layer

Pooling layer digunakan untuk memperkecil dimensi citra yang bertujuan memperoleh gambaran besar dari citra tersebut. *Pooling layer* terdiri dari 2 jenis, yaitu *Max pooling* dan *Average Pooling*. *Average pooling* adalah fungsi menemukan nilai rata-rata dari keseluruhan nilai pada pooling. Sedangkan *max pooling* adalah fungsi yang mencari nilai terbesar dalam ukuran pooling tersebut. Perbedaan antara kedua jenis *pooling* tersebut dapat ditunjukkan pada Gambar 2.4.



Gambar 2.4 Pooling Layer

2.3.4 Activation Function

Fungsi aktivasi (*Activation Function*) merupakan fungsi yang terdapat pada lapisan tersembunyi jaringan syaraf tiruan dan menjadi bagian dari simpul aktivasi. Fungsi ini menciptakan ketidaklinieran pada jaringan saraf karena tanpa fungsi aktivasi, jaringan saraf hanyalah fungsi linier. Fungsi non-linear dapat digunakan untuk menyelesaikan masalah yang kompleks karena fungsi-fungsi tersebut dapat secara bersamaan membentuk rancangan yang berbeda.

1. ReLU

Rectified Linear Unit (ReLU) merupakan fungsi aktivasi yang tidak menerima nilai negatif dan hanya menerima nilai riil pada input. Fungsi ReLU memetakan nilai negatif menjadi 0 dan tetap mempertahankan nilainya jika bernilai positif [5].

$$f_{ReLU}(x) = \max(0, x) \quad 2.2$$

2. Swish Activation (SiLU)

Swish adalah fungsi kontinu halus, tidak seperti ReLU yang merupakan fungsi linier sepotong-sepotong. Swish memungkinkan sejumlah kecil bobot negatif disebarluaskan, sementara ReLU membatasi semua bobot negatif ke nol.

$$f(x) = x * \frac{1}{1 + e^{-x}} \quad 2.3$$

3. Softmax

Fungsi aktivasi softmax biasanya terlihat pada output layer pada jaringan syaraf. Fungsi dari softmax adalah klasifikasi, fungsi *softmax* menghitung probabilitas dari setiap kelas target atas semua kelas target yang memungkinkan dan membantu untuk menentukan kelas target untuk input yang diberikan.

$$f(x)_i = \frac{\exp(x_i)}{\sum_{j=1}^k \exp(x_j)} \quad 2.4$$

2.3.5 Batch Normalization

Batch normalization adalah metode untuk menormalkan nilai input pada setiap layer, yang bertujuan untuk mengatasi masalah bias kovariat internal dan mempercepat *training deep learning*. *Batch normalization* merupakan salah satu metode populer untuk menghindari model menjadi *overfitting*. Metode ini menormalkan layer dan memungkinkan model melatih bobot yang dinormalisasi.

$$z = g(Wu + b) \quad 2.5$$

Di mana W dan b adalah parameter model yang dipelajari, dan g adalah nonlinier seperti sigmoid atau ReLU. Formulasi ini mencakup sepenuhnya terhubung dan lapisan berbelit-belit[17].

2.3.6 Fully Connected Layer

Fully connected layer merupakan lapisan terakhir yang terdapat pada CNN. Setiap neuron yang ada pada *layer* sebelumnya saling terhubung pada setiap neuron dalam *fully connected layer*. Tujuan dari layer ini adalah untuk melakukan transformasi dimensi pada data sehingga data dapat diklasifikasikan secara linear. *Fully Connected layer* biasanya melewati *feature extraction* yang dilakukan oleh *convolution layer* dan *pooling layer*. Hal tersebut digunakan untuk merancang kombinasi akhir dari fungsi nonlinier dan membuat prediksi jaringan.

$$y = f(W^T x + b) \quad 2.6$$

Di mana x dan y adalah vektor aktivasi input dan output. W menunjukkan matriks yang berisi bobot koneksi antara unit lapisan dan b mewakili vektor istilah bias. [5].

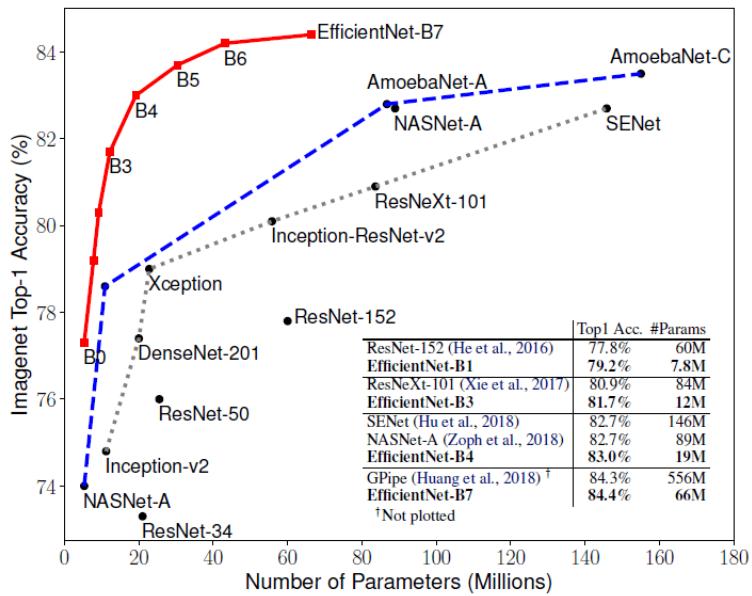
2.3.7 Global Average Pooling

Peta fitur dari *convolution layer* terakhir divektorisasi dan dimasukkan ke dalam lapisan yang terhubung sepenuhnya diikuti oleh lapisan regresi logistik *softmax*. Struktur ini menjembatani struktur konvolusional dengan jaringan saraf

tradisional. Memperlakukan *convolution layer* sebagai ekstraktor fitur. *Global average pooling* untuk mengganti *flatten* di CNN. Alih-alih menambahkan *fully connected layer* sepenuhnya di atas peta fitur, dibutuhkan rata-rata dari setiap peta fitur, dan vektor yang dihasilkan dimasukkan langsung ke layer *softmax*.

2.4 EfficientNet

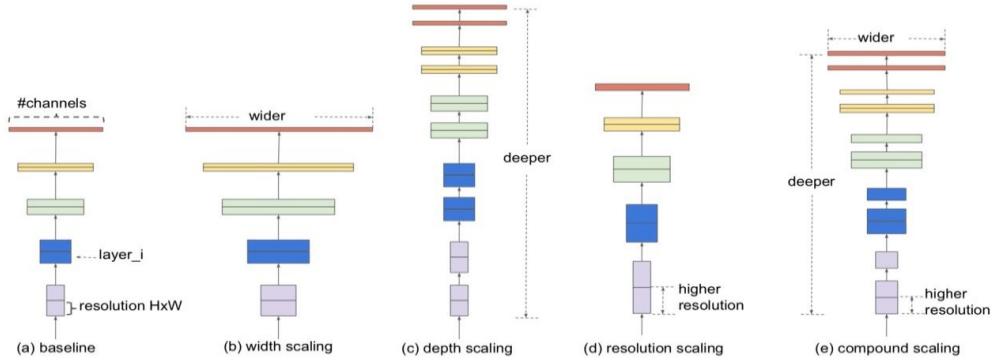
Arsitektur *EfficientNet* adalah salah satu model tercanggih dengan akurasi mencapai 84,4% dengan jumlah parameter 66 juta pada klasifikasi ImageNet[10]. *EfficientNet* terdiri dari 8 model antara B0 sampai B7. Seiring bertambahnya jumlah model, jumlah parameter tidak banyak meningkat. Sementara akurasi meningkat secara nyata. Berbeda dengan model CNN lainnya, arsitektur ini menggunakan fungsi aktivasi baru yang disebut SiLU (*Swish-1*). *EfficientNet* secara signifikan lebih unggul CNN lainnya seperti yang ditunjukkan pada Gambar 2.5.



Gambar 2.5 *ImageNet* akurasi vs Model size

EfficientNet merupakan arsitektur CNN yang menggunakan penskalaan ketiga dimensi kedalaman (*depth*), lebar (*width*), dan resolusi (*resolution*) jaringan secara seragam. Sehingga didapatkan jumlah parameter yang lebih sedikit yang membuat waktu proses menjadi lebih cepat namun juga didapatkan akurasi yang baik dari model sebelumnya. Sehingga model ini menawarkan efisiensi, dan performa yang lebih baik dari model lain. *EfficientNet* melakukan penskalaan

secara seragam pada semua dimensi dengan menggunakan penskalaan majemuk (*compound scaling*) [10]



Gambar 2.6 Penskalaan model

(a) adalah contoh jaringan dasar; (b)-(d) adalah penskalaan konvensional yang hanya meningkatkan satu dimensi jaringan lebar, kedalaman, atau resolusi. (e) adalah metode penskalaan majemuk (*compound scaling*) yang secara seragam menskalakan ketiga dimensi dengan rasio tetap [10].

2.4.1 Compound Scalling

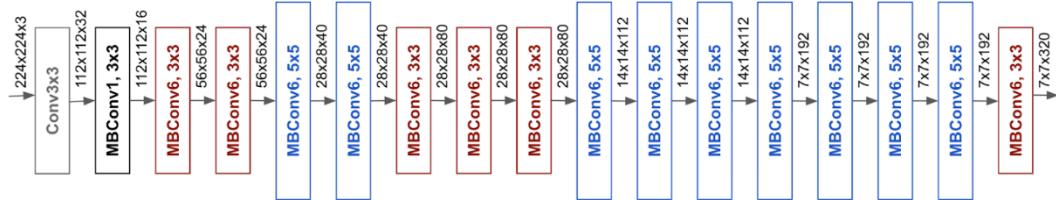
Metode penskalaan gabungan (*compound Scalling*) menggunakan *compound coefficient* ϕ untuk menskalakan *width*, *depth*, dan *resolution* secara bersamaan. Di bawah ini adalah rumus untuk atribut yang diskalakan:

$$\begin{aligned}
 \text{depth} : d &= \alpha^\phi \\
 \text{width} : w &= \beta^\phi \\
 \text{resolution} : r &= \gamma^\phi \\
 s.t \quad \alpha \cdot \beta^2 \cdot \gamma^2 &\approx 2 \\
 \alpha \geq 1, \beta \geq 1, \gamma \geq 1
 \end{aligned} \tag{2.7}$$

α , β , dan γ merupakan konstanta yang nilainya dapat ditentukan dengan menggunakan *grid search*. Sementara itu, ϕ merupakan koefisien yang nilainya ditentukan oleh pengguna yang digunakan untuk mengontrol besar sumber daya yang tersedia untuk melakukan penskalaan pada model. α , β , γ menentukan bagaimana sumber daya yang ditugaskan ke lebar, kedalaman, dan resolusi jaringan masing-masing. Kemudian memperbaiki α , β , γ sebagai konstanta dan

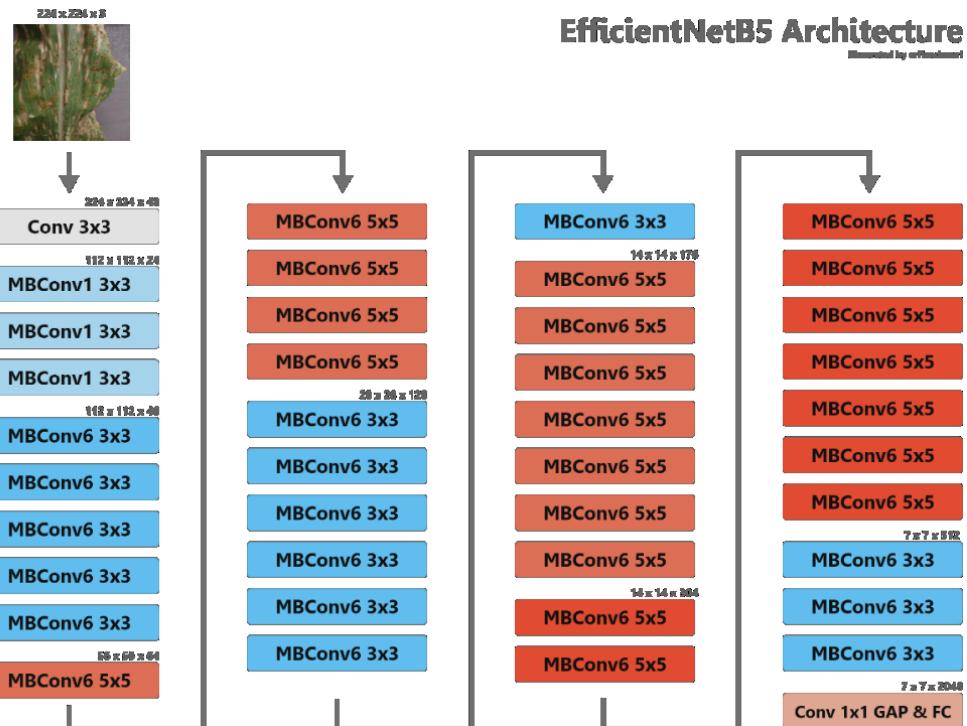
meningkatkan jaringan baseline dengan \emptyset berbeda menggunakan Persamaan 2.7, untuk mendapatkan *EfficientNet-B1* hingga B7 [10].

2.4.2 Arsitektur



Gambar 2.7 Baseline *EfficientNet* B0[10]

Baseline atau arsitektur dasar dari EfficientNetB0 yang memiliki layer-layer pada setiap *stage* nya ditunjukkan pada gambar Gambar 2.7. Layer pada *Convolutional neural network* biasanya dibagi menjadi beberapa *stages* dan semua layer pada setiap *stage* mempunyai arsitektur yang sama. Penskalaan tidak mengubah operasi lapisan, oleh karena itu lebih baik terlebih dahulu memiliki jaringan dasar yang baik dan kemudian menskalakannya di sepanjang dimensi yang berbeda menggunakan *compound scaling* yang diusulkan.



Gambar 2.8 Skema arsitektur *EfficientNetB5*

Meningkatkan jaringan dasar akan melahirkan keluarga model, yang disebut *EfficientNets*. Skema arsitektur *EfficientNetB5* yang ditunjukkan pada Gambar 2.8 termasuk dalam keluarga *EfficientNets* yang merupakan hasil dari penskalaan dari *baseline*/jaringan dasar *EfficientNetB0* menggunakan *compound scaling*. Arsitektur dikembangkan menggunakan *mobile inverted bottleneck convolution* (MBConv) mirip dengan MobileNetV2 dan MnasNet [10][18].

2.5 *Hyperparameter*

Banyak *hyperparameter* perlu disesuaikan sebelum melatih CNN untuk mengklasifikasikan gambar. *Hyperparameter* adalah parameter yang dapat disesuaikan yang memungkinkan mengontrol proses *training* model. Dalam kasus jaringan saraf, beberapa *hyperparameter* umum seperti: *learning rate* [19], *batch size* [13], jumlah *epoch*, dan fungsi aktivasi. *Hyperparameter* dipertimbangkan untuk analisis dan pengaruhnya terhadap akurasi *training* dan validasi. Proses mencari nilai optimal dari *hyperparameter* untuk memperbaiki performa model disebut *Hyperparameter tuning*. Hal ini dilakukan dengan mencoba berbagai nilai *hyperparameter* dan membandingkan hasilnya dengan metrik pengukuran.

2.5.1 *Batch size*

Salah satu *hyperparameter* utama yang perlu disesuaikan sebelum memulai proses *training* adalah *batch size*. *Batch size* adalah jumlah citra yang digunakan di setiap epoch untuk melatih jaringan. Mengatur *hyperparameter* ini terlalu tinggi dapat membuat jaringan membutuhkan waktu terlalu lama untuk mencapai konvergensi (tidak ada lagi peningkatan akurasi). Namun, jika terlalu rendah, membuat jaringan bolak-balik tanpa mencapai kinerja yang optimal [13].

2.5.2 *Learning rate*

Learning rate menentukan kecepatan model mempelajari bobot jaringan saraf dan memperbarui bobot selama proses *training*. *Learning rate* yang besar menghasilkan model pembelajaran yang cepat dengan *epoch* yang lebih sedikit tetapi mengarah pada bobot yang kurang optimal. Di sisi lain, *learning rate* yang

kecil membuat model menjadi lambat belajar tetapi membutuhkan lebih banyak epoch untuk dilatih dan menghasilkan bobot yang optimal. *Learning rate* juga memengaruhi kecepatan *training* sedemikian rupa sehingga semakin besar nilainya menghasilkan konvergensi yang lebih cepat [19].

2.6 K-Fold Cross Validation

Pada penilitian ini *k-fold cross validation* dipilih sebagai metode evaluasi untuk pembagian data. Metode ini membagi data menjadi k- bagian. Jika k bernilai 5, maka dataset akan dibagi menjadi 5 bagian sama besar dimana 4 bagian dijadikan sebagai data *training* dan satu sisanya dijadikan data *testing*. Dengan menggunakan *cross validation*, kami dapat memperoleh lebih banyak metrik dan menarik kesimpulan penting tentang algoritma dan dataset. Contoh *k-fold cross validation* dengan nilai k = 5 ditunjukkan pada Gambar 2.9.

	Data				
fold 1	test	train	train	train	train
fold 2	train	test	train	train	train
fold 3	train	train	test	train	train
fold 4	train	train	train	test	train
fold 5	train	train	train	train	test

Gambar 2.9 5-fold cross validation

2.7 Metrik Evaluasi

Evaluasi dilakukan untuk mengetahui seberapa kinerja dari suatu model atau arsitektur. Karena terdapat 4 kelas dalam dataset, maka dilakukan klasifikasi *multi-class*. Performa pada model yang dibahas dalam penelitian ini diukur menggunakan metrik yang berbeda seperti akurasi (*Acc*), presisi (*Pre*), spesifisitas (*Spe*) dan sensitivitas (*Sen*). Untuk mendapatkan nilai tersebut dibutuhkan *multi-class confusion matrix*. Contoh *multi-class confusion matrix* untuk kelas 0 ditunjukkan pada Gambar 2.10. Metrik dihitung menggunakan indeks seperti *True Positive* (*TP*), *True Negative* (*TN*), *False Positive* (*FP*) dan *False Negative* (*FN*). *TP* adalah jumlah citra yang diklasifikasikan dengan benar di setiap kategori. Sedangkan *TN*,

mewakili jumlah citra yang diklasifikasikan dengan benar di semua kategori lain kecuali untuk kategori yang relevan. FN adalah jumlah citra yang salah diklasifikasikan dari kategori yang relevan. FP adalah jumlah citra yang salah diklasifikasikan di semua kategori lain kecuali untuk kategori yang relevan.

		prediksi				
		kelas	0	1	2	3
target	0	TP	FN			
	1			TN		
	2	FP				
	3					

Gambar 2.10 *Multi-class confusion matrix* kelas 0

Akurasi menunjukkan tingkat data yang diklasifikasikan dengan benar dari semua data. Nilai dari akurasi dapat diperoleh dari persamaan 2.8 untuk kelas k .

$$Acc(k) = \frac{TP(k) + TN(K)}{TP(k) + FN(k) + TN(k) + FP(k)} \quad 2.8$$

Sensitivitas adalah rasio positif yang diprediksi dengan benar dari semua positif murni. Nilai dari sensitivitas dapat diperoleh dari persamaan 2.9.

$$Sen(k) = \frac{TP(k)}{TP(k) + FN(k)} \quad 2.9$$

Spesifisitas adalah rasio negatif yang diprediksi dengan benar dari semua negatif yang benar. Nilai dari spesifisitas dapat diperoleh dari persamaan 2.10.

$$Spe(k) = \frac{TN(k)}{TN(k) + FP(k)} \quad 2.10$$

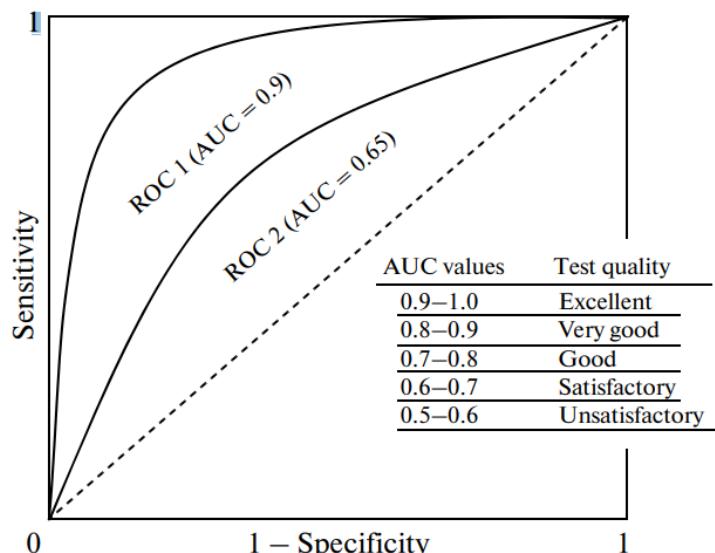
Sedangkan, presisi adalah proporsi positif yang diprediksi dengan benar dari semua identifikasi positif. Untuk memperoleh nilai presisi dengan persamaan 2.11.

$$Pre(k) = \frac{TP(k)}{TP(k) + FP(k)} \quad 2.11$$

Metrik dan perhitungannya diperluas untuk klasifikasi multi-kelas menggunakan rata-rata makro[20].

$$\begin{aligned}
 Avg\ Acc &= \frac{1}{class} \sum_{k=1}^{class} Acc(k) \\
 Avg\ Sen &= \frac{1}{class} \sum_{k=1}^{class} Sen(k) \\
 Avg\ Spe &= \frac{1}{class} \sum_{k=1}^{class} Spe(k) \\
 Avg\ Pre &= \frac{1}{class} \sum_{k=1}^{class} Pre(k)
 \end{aligned} \tag{2.12}$$

2.8 Receiver Operating Characteristic (ROC) Curve



Gambar 2.11 Kurva ROC

Kurva ROC adalah kurva dua dimensi hubungan antara *True Positive Rate* (TPR) atau sensitivitas (sumbu Y) dengan *false positive rate* (FPR) atau 1-Specifisitas (sumbu X). ROC merupakan sebuah kurva probabilitas dengan AUC (*Area Under the Curve*) sebagai area di bawah kurva tersebut. Nilai maksimum dari AUC adalah satu (1) yang mengindikasikan bahwa model klasifikasi tersebut memiliki nilai yang sempurna atau model klasifikasi tersebut seratus persen (100%) sensitif dan seratus persen (100%) spesifik [21]. Kurva yang berimpit dengan kurva diagonal, tidak memiliki kemampuan klasifikasi. Semakin kurva cembung dan

mendekati pojok kiri atas, semakin baik diskriminasi yang dimiliki model tersebut. Kebenaran tertinggi (sempurna) menempatkan pengklasifikasi dalam (0,1) [22].

2.9 Penelitian terkait

Zhang Z, *et al.* [4] pada tahun 2015 mengklasifikasikan enam jenis penyakit daun jagung menggunakan metode GA-SVM (*Genetic Algorithm Support Vector Machine*). Langkah yang digunakan mentransformasi citra ke format *bitmap*, lalu di konversi dari RGB ke HSI untuk mengekstrak fitur. Kemudian dilakukan segmentasi untuk mendapatkan citra biner. Teknik rotasi orthogonal digunakan untuk mendapatkan parameter yang sesuai untuk algoritma genetika. Dilaporkan hasil rata-rata tingkat klasifikasi mencapai 92,82%. Pada tahun 2019, Sibya & Sumbwanyambe [20] menggunakan 3 kelas klasifikasi penyakit yaitu daun hawar utara utara, karat biasa, dan *cercospora*. Arsitektur CNN yang digunakan tidak dijelaskan secara detail, namun hanya disebutkan menggunakan 50 *hidden layer* yang terdiri dari *convolution layer* dengan kernel filter yang memiliki median 24, *rectified linear unit* (ReLU) dan *pooling layer*. Seratus citra per kelas digunakan dengan rasio 70% untuk *training* dan 30% untuk *testing*.

Hasil *testing* menunjukkan akurasi 92,85%. Pada tahun 2020, Syarieff & Setiawan [7] menganalisis klasifikasi citra penyakit daun jagung dengan 2 langkah: Ekstraksi fitur menggunakan CNN dan klasifikasi menggunakan metode *machine learning*. 7 arsitektur CNN seperti, *AlexNet*, *virtual geometry group (VGG)* 16, *VGG19*, *residual network (ResNet)* 50, *ResNet101*, *GoogleNet*, dan *Inception-V3* digunakan untuk ekstraksi fitur dan metode klasifikasi seperti, *Support Vector Machine (SVM)*, *K-nearest Neighbour (KNN)*, dan *Decision Tree*. Berdasarkan hasil *testing*, klasifikasi terbaik adalah *AlexNet* dan *support vector machine* dengan akurasi, sensitivitas, spesifisitas masing-masing sebesar 93,5%, 95,08%, dan 93%. Di tahun yang sama, Waheed A, *et al.* [8] mengusulkan arsitektur *Dense convolution neural network* (*DenseNet*) yang dioptimalkan untuk mengenali dan mengklasifikasi tiga penyakit daun jagung. Menggunakan parameter yang jauh lebih rendah Performa arsitektur *DenseNet* yang dioptimalkan telah dibandingkan dengan arsitektur CNN lainnya dengan mempertimbangkan dua ukuran kualitas (waktu dan akurasi). Menggunakan kumpulan data besar dengan total 12.332 citra

dengan dimensi 250×250 piksel untuk eksperimen didapatkan hasil mencapai akurasi 98,06%.

Pada tahun 2021, Atila U, et al [9] mengklasifikasikan 39 jenis penyakit tanaman menggunakan menggunakan arsitektur *EfficientNet* B-0 sampai B-7 dan dikomparasi kan dengan model *deep learning* lainnya seperti, *AlexNet*, *VGG16*, *Resnet50*, dan *Inception V3*. Hasil yang diperoleh menunjukkan bahwa model arsitektur *EfficientNetB5* dan B4 mencapai nilai tertinggi dibandingkan model *deep learning* lainnya pada dataset asli dan *augmented* dengan masing-masing 99,91% dan 99,97% untuk akurasi dan 98,42% dan 99,39% untuk presisi. Penelitian-penelitian yang terkait pengklasifikasian penyakit daun tanaman jagung telah dilakukan dengan beberapa Metode berbeda-beda, diantaranya CNN. Metode *Deep learning* lebih baik daripada teknik *machine learning* konvensional. Dengan hasil yang di dapat sangat baik karenanya penggunaan metode ini diperluas dengan mengidentifikasi lebih banyak jenis penyakit daun jagung. Model *EfficientNet* yang mencapai akurasi dan efisiensi yang jauh lebih baik daripada CNN sebelumnya. *EfficientNet* mencapai *state-of-the-art* dengan akurasi 84,3% top-1 di *ImageNet*.

Tabel 2.1 Penelitian Terkait

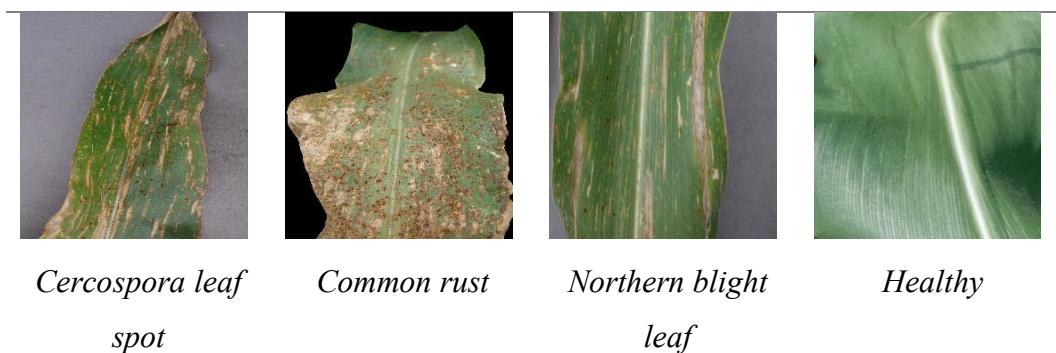
Peneliti, tahun	Permasalahan & Metode/solusi	Hasil	Kelemahan
Zhang Z, et al.[4] (2015)	Mengklasifikasikan enam jenis penyakit daun jagung menggunakan gabungan metode GA-SVM (<i>Genetic Algorithm Support Vector Machine</i>).	Dilaporkan hasil rata-rata tingkat klasifikasi mencapai 92,82%	Algoritma SVM tidak cocok untuk kumpulan data yang besar.
Sibiya & Sumbwanyambe [6](2019)	Mengklasifikasi 3 jenis penyakit daun jagung menggunakan Arsitektur CNN yang menggunakan 50 <i>hidden layer</i> yang terdiri dari <i>convolution layer</i> dengan kernel	Seratus citra per kelas digunakan dengan rasio 70% untuk <i>training</i> dan 30% untuk <i>testing</i> . Hasil <i>testing</i>	Arsitektur CNN yang digunakan tidak dijelaskan secara detail, namun hanya disebutkan jumlah layer yang digunakan

Peneliti, tahun	Permasalahan & Metode/solusi	Hasil	Kelemahan
	filter yang memiliki median 24, <i>rectified linear unit</i> (ReLU) dan <i>pooling layer</i> .	menunjukkan akurasi 92,85%	
Syarief & Setiawan [7] (2020)	Menganalisis klasifikasi citra penyakit daun jagung menggunakan 7 arsitektur CNN (<i>AlexNet</i> , <i>VGG16</i> , <i>VGG19</i> , <i>ResNet50</i> , <i>ResNet110</i> , <i>GoogleNet</i> , dan <i>Inception-V3</i>) untuk ekstraksi fitur dan 3 metode <i>machine learning</i> (SVM, KNN, <i>Decision tree</i>) untuk klasifikasi	Hasil <i>testing</i> klasifikasi terbaik adalah <i>AlexNet</i> dan <i>support vector machine</i> dengan akurasi, sensitivitas, spesifisitas masing-masing sebesar 93,5%, 95,08%, dan 93%.	Kombinasi arsitektur antara CNN untuk ekstraksi fitur dan metode <i>Machine learning</i> untuk <i>classifier</i> belum tentu mendapatkan hasil yang maksimal dibandingkan dengan metode CNN asli ataupun yang dioptimalkan.
Waheed A, et al. [8] (2020)	Mengenali dan mengklasifikasi tiga penyakit daun jagung menggunakan arsitektur <i>DenseNet</i> yang dioptimalkan.	Arsitektur <i>DenseNet</i> yang dioptimalkan mencapai akurasi 98,06%	Perbandingan akurasi dengan model <i>EfficientNet B0</i> didapatkan lebih tinggi daripada model yang disusulkan.
Atila U, et al. [9] (2021)	Klasifikasi 39 kelas penyakit daun tanaman menggunakan model <i>EfficientNet</i> dan model <i>deep learning</i> lainnya.	Hasil yang diperoleh model <i>EfficientNetB5</i> dan <i>EfficientNet B4</i> lebih unggul dari model <i>deep learning</i> lainnya.	Penggunaan <i>hyperparameter learning rate</i> 0,001 dan <i>batch size</i> 16. Tanpa adanya <i>hyperparameter tuning</i>

BAB 3 METODE USULAN

3.1 Dataset

Pada penelitian ini, dataset yang digunakan adalah citra daun jagung yang dikumpulkan dari website <https://data.mendeley.com/datasets/tywbtsjrjv/1>[23]. Berisikan 60.000-an citra dari berbagai tanaman, pada penelitian ini menggunakan mengambil data penyakit tanaman jagung yang diklasifikasikan menjadi 4 kelas yaitu, kelas daun sehat(*healthy*), kelas daun berkarat(*common rust*), kelas daun hawar utara(*northern leaf blight*), dan kelas daun bercak abu-abu (*Cercospora leaf spot*). Contoh data pada tiap kelasnya ditunjukkan pada Gambar 3.1.



Gambar 3.1 Contoh data citra jagung

Dataset citra daun yang digunakan dalam penelitian ini sebanyak 3.852 citra daun jagung yang terbagi pada masing-masing kelas yang dapat dilihat pada Tabel 3.1

Tabel 3.1 Penjabaran Dataset

Kelas	Jumlah citra
Healthy	1.162
Common rust	1.192
Northern blight leaf	985
Cercospora leaf spot	513
Total	3.852

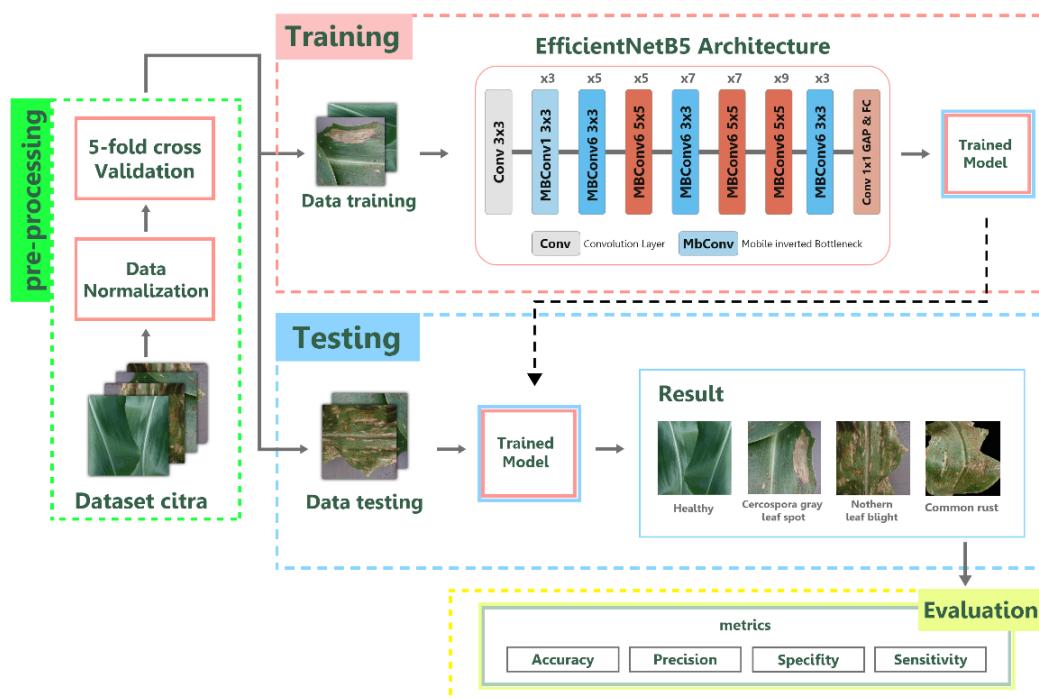
3.2 Pembagian Dataset

Dataset dibagi menjadi data *training* dan data *testing*. Pembagian menggunakan *5-fold cross validation* dimana dataset dibagi menjadi 5 bagian acak untuk *training* dan *testing*. Penggunaan metode ini dapat menguji kinerja model agar lebih efisien. Penjabaran pembagian data pada penelitian ini dapat dilihat pada Tabel 3.2.

Tabel 3.2 Pembagian Dataset

Kelas	Dataset	
	Pelatihan	Pengujian
<i>Healthy</i>	930	232
<i>Common rust</i>	954	238
<i>Northern blight leaf</i>	788	197
<i>Cercospora leaf spot</i>	410	103
Total	3.082	770

3.3 Arsitektur Sistem *training* dan *testing*



Gambar 3.2 *Block diagram training* dan *testing*

Arsitektur *EfficientNetB5* diimplementasikan untuk membangun model pada tahap *training*. Sedangkan pada tahap *testing* digunakan untuk menguji model yang telah dibangun pada tahap *training* sebelumnya. Alur *training* dan *testing* ditunjukkan oleh Gambar 3.2 yang terdiri dari beberapa proses, yaitu:

1. Dataset citra daun jagung dilakukan *pre-processing* terlebih dahulu, melakukan normalisasi pada data citra menjadi data *array*.
2. Setelah dinormalisasi data dibagi menjadi dua bagian yaitu, data *training* dan data *testing* menggunakan *5-fold cross validation*.
3. Tahap *training*, dilakukan dengan langkah-langkah berikut :
 - a. Memasukkan data *training* hasil normalisasi.
 - b. Data kemudian dilatih dalam arsitektur *EfficientNetB5* meliputi proses ekstraksi fitur dan klasifikasi dan menghasilkan model klasifikasi.
 - c. Hasil yang didapatkan berupa model klasifikasi hasil proses *training*.
4. Tahap *testing*, dilakukan dengan langkah-langkah berikut :
 - a. Memasukkan data *testing* hasil normalisasi
 - b. Data kemudian diuji dengan model yang telah dihasilkan dari proses *training*.
 - c. Hasil *testing* berupa evaluasi klasifikasi citra penyakit daun jagung dari kelas *healthy*, *common rust*, *northern blight leaf*, dan daun *cercospora leaf spot*.
5. Proses selanjutnya adalah evaluasi model untuk mengetahui tingkat kinerja dari model yang dihasilkan.

3.4 *Pre-processing* Citra

Sebelum melakukan proses *training* dan *testing*, dilakukan proses awal pada data citra. Mengubah input citra sesuai dengan persyaratan ukuran input model. Ukuran citra dataset semula 256x256 diubah menjadi 224x224. Kemudian, dilakukan normalisasi setiap citra untuk menskalakan data ke kisaran yang diterima jaringan. Menormalkan nilai piksel sehingga setiap nilai piksel memiliki nilai antara 0 dan 1. Hal ini dapat dicapai dengan membagi semua nilai piksel dengan nilai

piksel terbesar (255). Setelah dataset dilakukan normalisasi, dataset dibagi menjadi 5 bagian terlebih dahulu menggunakan *k-fold cross validation* untuk membaginya menjadi data *training* dan data *testing* sebanyak 5 *fold*.

Tabel 3.3 Data fitur citra Asli 12x12

134	133	128	100	98	102	116	105	97	85	96	98
142	144	133	83	116	81	134	110	131	93	78	86
146	143	137	143	94	92	154	95	109	98	125	90
148	146	154	102	91	81	162	160	113	107	91	92
150	147	56	52	70	134	136	122	169	154	84	98
149	142	104	78	97	170	115	131	109	146	93	103
155	141	83	94	85	167	90	142	97	146	105	111
149	133	104	113	143	190	184	163	116	99	106	112
148	104	128	103	150	132	148	128	111	95	108	107
141	94	82	92	116	130	121	94	135	103	106	102
128	99	98	127	149	107	109	87	107	113	113	115
96	93	90	127	136	116	108	114	123	126	125	121

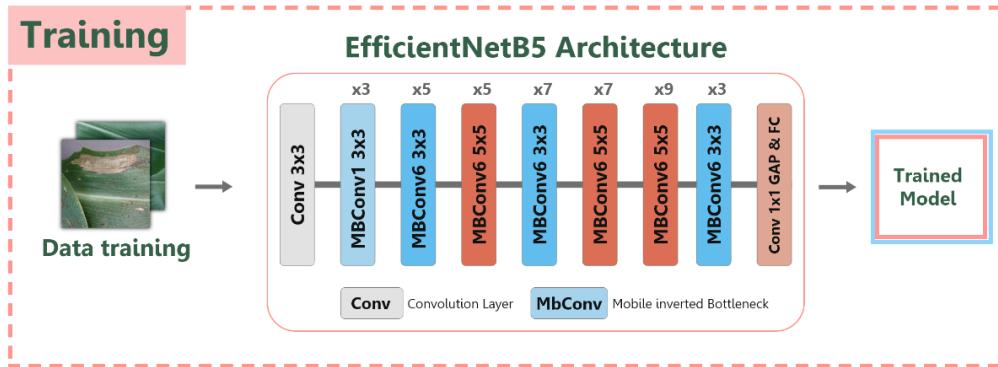
Contoh perhitungan normalisasi data fitur citra:

- $134 / 255 = 0,53$
- $133 / 255 = 0,52$
- $128 / 255 = 0,50$
- $142 / 255 = 0,56$
- $144 / 255 = 0,56$
- $133 / 255 = 0,52$
- $144 / 255 = 0,56$
- Dst.

Tabel 3.4 Data fitur citra normalisasi 12x12

0,53	0,52	0,50	0,39	0,38	0,40	0,45	0,41	0,38	0,33	0,38	0,38
0,56	0,56	0,52	0,33	0,45	0,32	0,53	0,43	0,51	0,36	0,31	0,34
0,57	0,56	0,54	0,56	0,37	0,36	0,60	0,37	0,43	0,38	0,49	0,35
0,58	0,57	0,60	0,40	0,36	0,32	0,64	0,63	0,44	0,42	0,36	0,36
0,59	0,58	0,22	0,20	0,27	0,53	0,53	0,48	0,66	0,60	0,33	0,38
0,58	0,56	0,41	0,31	0,38	0,67	0,45	0,51	0,43	0,57	0,36	0,40
0,61	0,55	0,33	0,37	0,33	0,65	0,35	0,56	0,38	0,57	0,41	0,44
0,58	0,52	0,41	0,44	0,56	0,75	0,72	0,64	0,45	0,39	0,42	0,44
0,58	0,41	0,50	0,40	0,59	0,52	0,58	0,50	0,44	0,37	0,42	0,42
0,55	0,37	0,32	0,36	0,45	0,51	0,47	0,37	0,53	0,40	0,42	0,40
0,50	0,39	0,38	0,50	0,58	0,42	0,43	0,34	0,42	0,44	0,44	0,45
0,38	0,36	0,35	0,50	0,53	0,45	0,42	0,45	0,48	0,49	0,49	0,47

3.5 Proses Training



Gambar 3.3 Proses *training*

Proses *training* seperti yang di tunjukkan pada Gambar 3.3. Data citra *training* yang telah dilakukan *pre-processing* sebelumnya dengan nilai input citra 224x224. Kemudian, dilakukan proses *training* dengan menggunakan arsitektur *EfficientNetB5* hingga dihasilkan model yang dapat digunakan pada proses *testing*.

3.6 Perhitungan Proses Algoritma *EfficientNetB5*

Perhitungan proses *training* model menggunakan persamaan yang digunakan pada setiap *layer* model mulai dari *Convolution layer*, *MBConv layer*, dan *Fully Connected layer* yang ada dalam model *EfficientNetB5*.

1. *Input* data fitur citra yang diperoleh dari pre-processing pada Tabel 3.4 dilakukan *stage 1 conv layer* $k = 3 \times 3$. Berikut adalah hasil perhitungan pada *Convolution layer* $k = 3 \times 3$ dengan menggunakan 1 *channel*.

Tabel 3.5 Contoh Perhitungan *Convolution layer*

$k = 3 \times 3$, stride = 2

0,3	0,3	0,0
0,4	0,2	0,1
-0,2	-0,2	-0,2

(a)

Hasil :

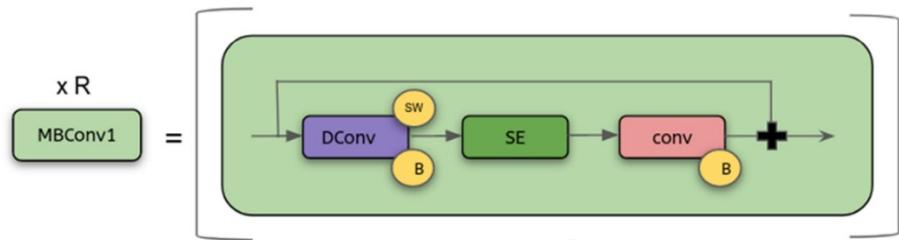
0,30	0,24	0,22	0,27	0,22	0,21
0,40	0,49	0,17	0,32	0,18	0,28
0,36	0,14	0,24	0,32	0,36	0,22
0,37	0,17	0,32	0,36	0,29	0,29
0,33	0,19	0,33	0,35	0,26	0,27
0,49	0,52	0,60	0,50	0,55	0,51

(b)

Pada Tabel 3.5(a) adalah kernel 3x3, sedangkan Tabel 3.5(b) hasil data fitur dikonvolusi dengan kernel pada Tabel 3.5(a). Contoh perhitungan *convolution layer* dengan persamaan 2.1:

- $(0,53 \times 0,3) + (0,52 \times 0,3) + (0,5 \times 0) + (0,56 \times 0,4) + (0,56 \times 0,2) + (0,52 \times 0,1) + (0,57 \times -0,2) + (0,56 \times -0,2) + (0,54 \times -0,2) = 0,30$
- $(0,54 \times 0,3) + (0,56 \times 0,3) + (0,37 \times 0) + (0,60 \times 0,4) + (0,40 \times 0,2) + (0,36 \times 0,1) + (0,22 \times -0,2) + (0,20 \times -0,2) + (0,27 \times -0,2) = 0,49$
- Dst.

2. Operasi *MBConv1*



Gambar 3.4 Ilustrasi *Block MBConv1* [24]

- Pada tahap ini dilakukan operasi *DepthwiseConv* dengan filter 3x3. Dihasilkan data fitur hasil *DepthwiseConv* seperti pada Tabel 3.6. Perhitungannya mirip dengan *convolution layer* karena pada contoh ini hanya menggunakan 1 *channel*.

Tabel 3.6 Data fitur hasil *DepthWiseConv*

0,0	0,2	0,1	0,40	0,30	0,38	0,35	0,25	0,06
-0,1	0,2	0,2	0,41	0,37	0,44	0,40	0,21	0,09
0,4	0,1	0,2	0,35	0,41	0,45	0,42	0,22	0,07
			0,52	0,60	0,61	0,53	0,35	0,16
			0,32	0,34	0,32	0,31	0,14	-0,04
			0,22	0,23	0,21	0,22	0,13	0,01

- Dilanjutkan proses *batch normalization* menggunakan persamaan 2.5. Fungsi ReLU dari mengubah nilai negatif ke positif (karena 1 *channel* jadi nilai ReLU tetap).

Tabel 3.7 Data Hasil *Batch Normalization*

0,40	0,60	0,38	0,35	0,50	0,06
0,41	0,74	0,44	0,40	0,43	0,09
0,35	0,82	0,45	0,42	0,43	0,07
0,52	1,21	0,61	0,53	0,70	0,16
0,32	0,68	0,32	0,31	0,27	0,00
0,22	0,45	0,21	0,22	0,26	0,01

Contoh perhitungan *Batch normalization* dengan persamaan 2.1:

- $z = g(Wu + b) = 0,40 (0+1) = 0,40$
- $z = g(Wu + b) = 0,60 (0+1) = 0,60$
- $z = g(Wu + b) = 0,38 (0+1) = 0,38$

- c. Kemudian diaktivasi menggunakan aktivasi *swish* dengan persamaan 2.3.

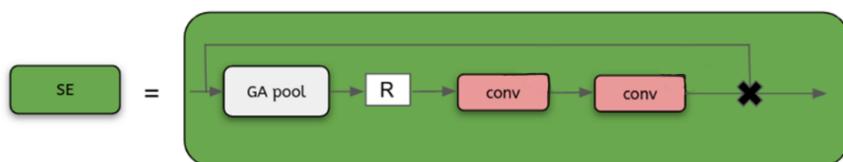
Tabel 3.8 Data Hasil Aktivasi *Swish*

0,239	0,387	0,228	0,206	0,310	0,030
0,246	0,504	0,268	0,241	0,258	0,046
0,206	0,570	0,276	0,256	0,262	0,038
0,325	0,928	0,393	0,333	0,466	0,088
0,182	0,452	0,183	0,179	0,154	0,000
0,123	0,276	0,117	0,119	0,148	0,006

Contoh perhitungan aktivasi *swish* menggunakan persamaan 2.3:

- $0,40 * 1 / (1 + e^{-0,40}) = 0,239$
- $0,60 * 1 / (1 + e^{-0,60}) = 0,387$
- $0,38 * 1 / (1 + e^{-0,38}) = 0,228$
- Dst.

- d. Selanjutnya dilakukan proses *Squeeze and Excitation*.



Gambar 3.5 Ilustrasi *Block SE* [24]

- e. Pada tahap ini dilakukan operasi *Global average pooling*, di lakukan pooling tiap *channel*.

GAP	0,1779
-----	--------

Nilai Global average pooling dapat dihitung sebagai berikut:

$$\frac{0,239 + 0,172 + 0,228 + 0,206 + 0,140 \dots \dots + 0,006}{6} = 0,1779$$

- f. Kemudian, dilakukan *Reshape*, perkalian antara GAP dengan filter 6x6

Tabel 3.9 Contoh Bobot

0,02	0,23	0,15	0,02	0,02	0,23
-0,09	0,23	0,23	-0,09	-0,09	0,23
0,35	0,08	0,02	0,23	0,08	0,19
0,02	0,23	-0,09	0,23	0,23	0,15
0,02	0,23	0,23	-0,09	0,02	0,23
-0,09	0,23	0,19	0,35	-0,09	0,23

Tabel 3.10 Data hasil *Reshape*

0,004	0,041	0,027	0,004	0,004	0,041
-0,016	0,041	0,041	-0,016	-0,016	0,041
0,063	0,015	0,004	0,041	0,015	0,034
0,004	0,041	-0,016	0,041	0,041	0,027
0,004	0,041	0,041	-0,016	0,004	0,041
-0,016	0,041	0,034	0,063	-0,016	0,041

Melakukan *Reshape* data dari *Global Average Pooling* dengan perkalian bobot. Contoh perhitungan sebagai berikut:

- $0,1779 \times 0,02 = 0,004$
 - $0,1779 \times 0,23 = 0,041$
 - $0,1779 \times 0,15 = 0,027$
 - Dst.
- g. Selanjutnya, dilakukan 2 kali *convolution* dengan filter 3x3, masing-masing ditunjukan pada tabel 3.11.

Tabel 3.11 Data hasil *convolution* 3x3

0,017	0,021	0,018	0,010	0,011	0,007
0,021	0,059	0,023	-0,001	0,037	0,029
0,029	0,016	0,038	0,013	0,033	0,033
0,036	0,024	0,024	0,047	0,024	0,016
0,024	0,030	0,035	0,023	0,056	0,014
0,013	0,035	0,027	0,005	0,007	0,021

(a)

0,022	0,024	0,027	0,018	0,011	0,016
0,031	0,044	0,017	0,031	0,033	0,019
0,032	0,047	0,037	0,030	0,048	0,022
0,031	0,036	0,044	0,048	0,033	0,031
0,032	0,036	0,040	0,041	0,030	0,006
0,021	0,026	0,017	0,015	0,021	0,009

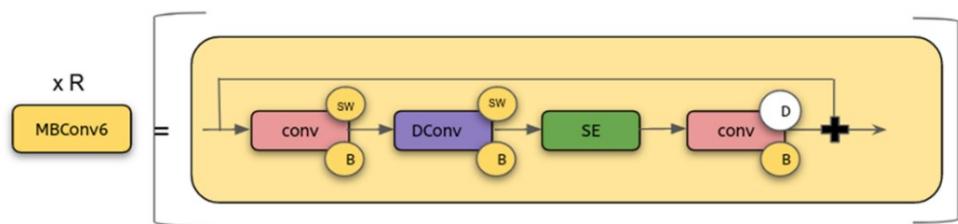
(b)

- h. Selanjutnya dilakukan konvolusi dengan filter 3x3 dan *batch normalization*.

Tabel 3.12 Data fitur hasil *Batch Normalization*

0,022	0,049	0,027	0,018	0,023	0,016
0,031	0,088	0,017	0,031	0,066	0,019
0,032	0,094	0,037	0,030	0,097	0,022
0,031	0,072	0,044	0,048	0,066	0,031
0,032	0,071	0,040	0,041	0,059	0,006
0,021	0,052	0,017	0,015	0,043	0,009

3. *Stage MBConv6*, Pada Operasi dilakukan beberapa tahapan:

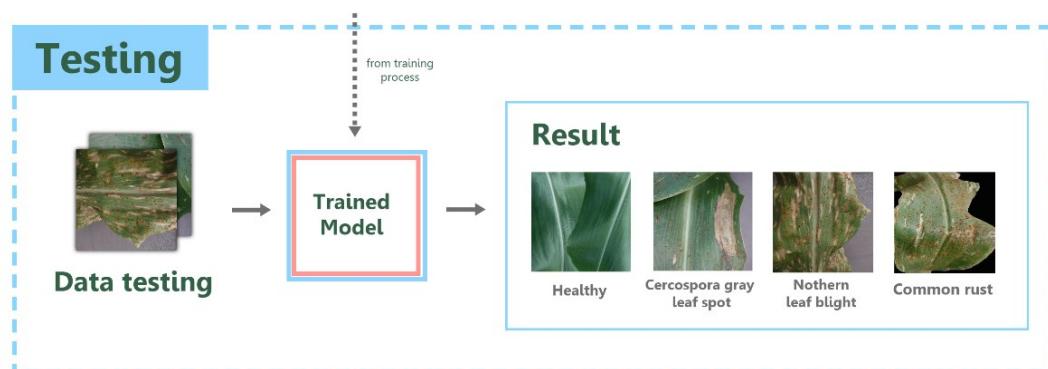


Gambar 3.6 Illustrasi *Block MBConv6* [24]

- a. Pada tahap dilakukan *convolution* dengan filter 3x3
 b. Kemudian dilakukan proses *batch normalization*.

- c. Setelah itu, proses aktivasi menggunakan *swish*.
 - d. Selanjutnya, dilakukan sama seperti proses *MBConv1* dengan filter 3x3.
 - e. Proses *MBConv6* dilakukan berulang sebanyak *layer* pada setiap *stage*.
4. *Stage Conv 1x1, Pooling, dan Fully Connected Layer*
- a. Pada Tahap ini dilakukan conv 1x1,
 - b. Kemudian data dilakukan *pooling* menggunakan *Global average pooling* yang menggantikan fungsi *flatten* menjadi bentuk 1 dimensi.

3.7 Proses *Testing*



Gambar 3.7 Proses *testing*

Proses *testing* dilakukan dengan menggunakan input data *testing* yang akan diprediksi menggunakan model yang telah dibangun pada proses *training*. *Output* yang diperoleh dari proses ini yakni hasil klasifikasi masing-masing data citra. Proses *testing* dapat dilihat pada Gambar 3.7.

3.8 Evaluasi Model

Evaluasi dilakukan untuk mengetahui kinerja dari sebuah model. Evaluasi dilakukan dengan membandingkan data target dan data hasil prediksi yang disimpan dalam *confusion matrix*. Dari matriks inilah didapatkan nilai akurasi, presisi, spesifisitas, dan sensitivitas. Tabel 3.13 menunjukkan contoh perbandingan data target dan data hasil prediksi pada data yang diuji.

Tabel 3.13 Contoh Perbandingan data target dan hasil prediksi

	Citra 1	Citra 2	Citra 3	Citra 4	Citra 5	Citra 6	Citra 7	Citra 8
Target	0	3	1	2	1	2	0	1
Prediksi	0	3	1	2	2	2	1	1

Tabel 3.13 adalah perbandingan hasil klasifikasi *multi-class* dengan banyak kelas sebanyak 4 kelas. Maka *confusion matrix* yang dibentuk mengikuti aturan *multi-class*. Tabel 3.14 menunjukkan *confusion matrix* dari data pada Tabel 3.13.

Tabel 3.14 *Confusion matrix*

		Prediksi				
		kelas	0	1	2	3
Target	0	1	1	0	0	0
	1	0	2	1	0	0
	2	0	0	2	0	0
	3	0	0	0	0	1

Selanjutnya adalah menghitung nilai pada indeks TP, FP, FN, dan TN masing-masing kelas. Hasil perhitungan dapat ditunjukkan pada Tabel 3.15.

Tabel 3.15 Hasil perhitungan tiap kelas

	TP	FP	FN	TN
Kelas 0	1	0	1	6
Kelas 1	2	1	1	4
Kelas 2	2	1	0	5
Kelas 3	1	0	0	7

Setelah itu, menghitung nilai evaluasi metrik akurasi, presisi, spesifisitas, dan sensitivitas. Contoh hasil perhitungan evaluasi metrik dapat ditunjukkan pada Tabel 3.16.

Tabel 3.16 Hasil Evaluasi metrik

Kelas	Acc	Sen	Spe	Pre
Kelas 0	0,875	0,500	1,000	1,000
Kelas 1	0,750	0,667	0,800	0,667
Kelas 2	0,875	1,000	0,833	0,667
Kelas 3	1,000	1,000	1,000	1,000
Rata-rata	0,875	0,792	0,908	0,833

3.9 Skenario Ujicoba

Skenario uji coba dilakukan untuk mengetahui jawaban dari rumusan masalah terkait dengan tingkat akurasi yang diperoleh dari model terhadap klasifikasi penyakit daun tanaman jagung. Informasi mengenai skenario uji coba yang dilakukan dalam penelitian ini dapat dilihat pada Tabel 3.17. Dimana tiap skenarionya bertujuan untuk mengetahui tingkat akurasi yang dihasilkan dataset dengan model *EfficientNetB5*.

Tabel 3.17 Skenario Ujicoba

No.	Percobaan	<i>Learning rate (α)</i> (adam)	<i>Batch size</i>
1		0,001 [19]	16 [13]
2		0,0001 [19]	16
3	Ujicoba dataset menggunakan	0,00001 [19]	16
4	model <i>EfficientNetB5</i>	0,001	32 [13]
5		0,0001	32
6		0,00001	32

Pada setiap skenario masing-masing bertujuan untuk mengetahui pengaruh kombinasi *learning rate* dan *batch size* yang tepat terhadap model kinerja model dalam mengklasifikasi penyakit daun tanaman jagung dengan mengan menggunakan model *EfficientNetB5*. Selain itu, pada Tabel 3.17 pemilihan nilai *learning rate* berdasarkan penelitian chamarty (2020) didapatkan akurasi terbaik

diperoleh pada 3 *learning rate* 0.001, 0.0001, dan 0.00001 [19]. Sedangkan pada *batch size* ukuran yang dipertimbangkan menurut Kandel & Casteli (2020) yang dalam percobaannya memberikan performa dan hasil yang optimal pada ukuran *batch size* 16 dan 32 [13]. Dalam studi eksperimental ini, resolusi citra input harus diubah ukurannya dari semula 256 x 256 diubah ke ukuran default yang diterima oleh model yaitu 224 x 224. Nilai parameter lain seperti jumlah *epoch* sebesar 20 dan optimasi dengan menggunakan Adam disesuaikan seperti pada penelitian Atila U, et al. (2021) [9]. Dalam menentukan model klasifikasi terbaik digunakan evaluasi yakni akurasi, presisi, spesifitas dan sensitivitas. Hasil dari masing-masing skenario dibandingkan dan dianalisa, kemudian untuk menentukan hasil akhir kinerja model terbaik dilakukan perhitungan rata-rata hasil nilai pada setiap skenario.

BAB 4 HASIL DAN PEMBAHASAN

4.1 Lingkungan Uji Coba

Bagian ini dijelaskan mengenai spesifikasi perangkat yang digunakan pada saat *training* dan *testing* sistem. Perangkat keras maupun perangkat lunak yang digunakan dapat dilihat pada Tabel 4.1 dan Tabel 4.2

Tabel 4.1 Spesifikasi perangkat

no	Kebutuhan	Jenis
1	vCPU	Intel(R) Xeon(R) CPU @ 2.20GHz
2	vGPU	Tesla T4 16GB
3	Web IDE	Google Colab Pro
4	Bahasa Pemrograman	Python 3
5	Web browser	Google Chrome

Tabel 4.2 *Library* yang digunakan

No	Library	versi	Fungsi
1.	Keras	2.8.0	Menyediakan <i>toolkit</i> untuk pembelajaran mesin serta melakukan implementasi metode <i>EfficientNet</i>
2.	Numpy	1.22.4	Menyediakan fungsi yang dapat digunakan untuk proses pengolahan atau penyimpanan data sehingga dapat dengan melakukan modifikasi terhadap data untuk <i>training</i> dan <i>testing</i> .
3.	Scikit Learn	1.2.1	Dapat digunakan untuk mendapatkan hasil dalam proses evaluasi model terutama <i>confusion matrix</i>
4.	Matplotlib	3.5.3	Dapat digunakan untuk membantu menampilkan hasil visualisasi data dan menghasilkan suatu gambar yang dapat disimpan
5.	OpenCV	4.6.0	OpenCV digunakan untuk membaca dan menyimpan citra serta melakukan <i>pre-processing</i> .

4.2 Tahap-Tahap Pembuatan Program

Ada beberapa tahapan yang dilakukan untuk membangun program *training* dan *testing* yakni sebagai berikut:

4.2.1 Mendefinisikan nilai parameter

Pada tahap ini mempersiapkan nilai parameter seperti, nama kelas, jumlah kelas, jumlah epoch, nilai *learning rate*, dan nilai *batch size* sesuai skenario.

```
1 dataset_path = '/content/gdrive/My Drive/dataset'
2
3 #Define parameter
4 labels = os.listdir(dataset_path)
5 class_labels = labels.copy()
6 num_class = len(class_labels)
7
8 img_width, img_height, img_num_channels = 224, 224, 3
9 epoch = 20
10
11 #EXPERIMENTS SKENARIO
12 skenario = 1
13
14 if skenario == 1:
15     str_skenario = "skenario_1"
16     learning_rate = 0.001
17     batch_size = 16
18 elif skenario == 2:
19     str_skenario = "skenario_2"
20     learning_rate = 0.0001
21     batch_size = 16
22 elif skenario == 3:
23     str_skenario = "skenario_3"
24     learning_rate = 0.00001
25     batch_size = 16
26 elif skenario == 4:
27     str_skenario = "skenario_4"
28     learning_rate = 0.001
29     batch_size = 32
30 elif skenario == 5:
31     str_skenario = "skenario_5"
32     learning_rate = 0.0001
33     batch_size = 32
34 elif skenario == 6:
35     str_skenario = "skenario_6"
36     learning_rate = 0.00001
37     batch_size = 32
```

Kode Program 4.1 Mendefinisikan parameter

Penjelasan dari kode program 4.1 adalah sebagai berikut:

1. Baris 1: variable dataset_path adalah lokasi menunjukkan direktori dataset

2. Baris 3-9: Mendefinisikan parameter seperti, nama kelas, jumlah kelas, epoch, dan ukuran input citra.
3. Baris 11-37: *If condition* skenario berguna mendefinisikan skenario program akan dimuat dengan nilai variabel *str_skenario* untuk merekayasa lokasi direktori. Selanjutnya nilai *learning rate* dan *batch size* disesuaikan seperti pada skenario ujicoba.

4.2.2 Melakukan *Preprocessing*

Pada tahap ini citra dataset dimuat kedalam program menjadi sebuah *array*. Hal ini dimaksudkan agar dalam memudahkan dalam melakukan normalisasi. Tahapan melakukan *preprocessing* dapat dilihat pada Kode Program 4.2.

```

1 def preprocessing(data, img_shape):
2     dataset_path = data
3     labels = os.listdir(dataset_path)
4     class_labels = labels.copy()
5
6     X = []
7     y = []
8     label = 0
9     for classname in class_labels:
10         count = 0
11         for img in glob.glob(data + "/" + classname + '/*'):
12             im = cv2.imread(img)
13             im = cv2.resize(im, (img_shape, img_shape))
14             im = image.img_to_array(im)/255
15             X.append(im)
16             y.append(label)
17             count += 1
18             print("Jumlah "+str(classname)+"["+str(label)+"] "+"
: "+str(count))
19             label += 1
20     X = np.array(X)
21     y = np.array(y)
22
23     return X, y

```

Kode Program 4.2 *Preprocessing*

Penjelasan dari Kode Program 4.2 adalah sebagai berikut:

1. Baris 1: Membuat fungsi *preprocessing* dengan parameter ukuran dimensi citra dan lokasi dataset yang akan dilakukan *preprocessing*.
2. Baris 2-4: Variabel *dataset_path* sebagai deklarasi dimana direktori dataset disimpan dan variabel *class_labels* digunakan sebagai target *classname* pada direktori dataset.

3. Baris 6-8: Inisialisasi variabel *list* X dan y, dimana digunakan untuk menyimpan data hasil normalisasi di variable X dan y sebagai data label yang didapat dari variabel label.
4. Baris 9: Perulangan pertama adalah perulangan untuk setiap kelasnya
5. Baris 11-13: Perulangan kedua adalah perulangan setiap citra di setiap kelas pada direktori, kemudian diubah ukuran sesuai ukuran input.
6. Baris 14: Melakukan normalisasi data dengan cara mengubahnya menjadi data array kemudian membaginya nilai piksel dengan nilai piksel terbesar (255).
7. Baris 15-16: Menyimpan setiap data array ke variabel list X dan y adalah data labelnya.
8. Baris 17: Variabel *count* dimaksudkan sebagai penghitung jumlah data pada tiap kelasnya.
9. Baris 20-23: Setelah proses berjalan data X dan y di konversi menjadi sebuah *array*.

4.2.3 Pembagian Dataset *Training* dan *Testing*

Setelah data dirubah menjadi *array* maka selanjutnya data akan dibagi menjadi data *training* dan data *testing* menggunakan metode *K-Fold Cross Validation* dimana jumlah *k-fold* adalah 5 seperti yang ditunjukan pada Kode Program 4.3.

```

1 def split_data(k, X, y):
2     kfolds = StratifiedKFold(n_splits=k)
3     yc = to_categorical(y, 4)
4     for num, (train_index, test_index) in enumerate(kfolds.split(X, y)):
5         num += 1
6         X_train, X_test = X[train_index], X[test_index]
7         y_train, y_test = yc[train_index], yc[test_index]
8         #saving data per fold with numpy compressed
9         savez_compressed(f'/content/gdrive/My Drive/data_folds/data_fold_{num}', X_train, X_test, y_train, y_test)
10
11         print("Saving data fold "+str(num)+" to directory")

```

Kode Program 4.3 *k-Fold Cross Validation*

Penjelasan dari Kode Program 4.3 adalah sebagai berikut:

1. Baris 1: Fungsi *split_data* dengan parameter nilai fold, data array, dan data label.
2. Baris 2-3: Variabel *kfold* digunakan mendeklarasikan modul *StratifiedKFold* yang digunakan untuk mengambil sampel dalam data fitur dan data label dan membaginya sebanyak k. Variabel *yc* digunakan untuk mengubah data label y menjadi bentuk data kategori.
3. Baris 4-9: Perulangan yang digunakan membagi data menjadi 5 bagian data *training* dan *testing* menggunakan *5-fold cross validation*, kemudian data fitur dan label disimpan dalam paket *numpy compressed (npz)*.

4.2.4 Pembuatan Model *EfficientNet*

```
1 def build_model():
2
3     base_model = EfficientNetB5(weights=None, include_top=False,
4     input_shape=(img_width, img_height, img_num_channels))
5     model = Sequential()
6     model.add(base_model)
7     model.add(GlobalAveragePooling2D())
8     model.add(layers.Dense(num_class, activation="softmax",
9     name="fc_out"))
10
11 return model
```

Kode Program 4.4 Model *EfficientNet*

Penjelasan dari Kode Program 4.4 adalah sebagai berikut:

1. Baris 1: Membuat fungsi untuk membangun model.
2. Baris 3: Pembuatan *base model* dengan library EfficientNetB5 dengan parameter ukuran input, tanpa *pre-trained* model atau *weights=none*,
3. Baris 4: Model dibangun secara sekuisial menambahkan *base_model* sebagai dasar model.
4. Baris 5-7: Kemudian penambahan *output layer Global Average Pooling* sebagai *flatten layer* yang menghasilkan *output* 1 dimensi. Selanjutnya, adalah *classification layer* dengan output sebanyak jumlah kelas dengan fungsi aktivasi *softmax*.

4.2.5 *Training* data

Tahapan ini melakukan proses *training* data yang telah dibagi melalui proses *5-fold cross validation*. Untuk proses *training* data dapat dilihat pada Kode Program 4.5.

```
1 def training(fold):
2
3     for k in range(1,fold+1):
4         print("fold-"+str(k))
5         # load data_fold from npz file (dict of arrays)
6         dict_data = load("/content/gdrive/My Drive/data_folds/data_fold_"+str(k)+".npz")
7
8         # extract arrays
9         X_train = dict_data['arr_0']
10        y_train = dict_data['arr_2']
11
12        file_path = "/content/gdrive/My Drive/"+str(str_skenario)+"/models/model_fold_"+str(k)+".h5"
13        model = build_model()
14        model.compile(optimizer=Adam(learning_rate=learning_rate), loss = "categorical_crossentropy", metrics=["accuracy"])
15
16        history = model.fit(X_train, y_train, batch_size=batch_size,verbose=1, epochs=epoch)
17        model.save(file_path)
18
19        # convert the history.history dict to a pandas Data
20        Frame:
21            hist_df = pd.DataFrame(history.history)
22
23            # save to json:
24            hist_json_file = "/content/gdrive/My Drive/"+str(str_skenario)+"/history_folds/history_fold_"+str(k)+".json"
25            hist_df.to_json(hist_json_file)
```

Kode Program 4.5 *Training* data

Penjelasan dari kode program 4.5 adalah sebagai berikut:

1. Baris 1: Membuat fungsi untuk melatih data *training* tiap *fold* data dengan model yang telah dibangun sebelumnya.
2. Baris 3: Melakukan perulangan sesuai jumlah *fold*.
3. Baris 6-10: Memuat data *training* (*X_train*, *y_train*) dari file *numpy compressed* yang telah disimpan sebelumnya.
4. Baris 12: Variabel *file_path* adalah lokasi file model yang nantinya akan disimpan.

5. Baris 13-16: Memanggil fungsi yang membangun model sebelumnya yaitu *build_model*. Model dilakukan compile dengan parameter learning rate yang telah ditentukan, categorical cross-entropy sebagai fungsi loss, dan metric akurasi sebagai evaluasinya. Kemudian, melakukan proses *training* dengan modul fit dengan parameter masukan berupa data *training*, data label, jumlah *epoch* dan *batch size*.
6. Baris 17: menyimpan model hasil *training* di lokasi sesuai variabel *file_path*.
7. Baris 20-24: Setelah di lakukan *training*, didapatkan hasil loss dan akurasi selama *training* yang disimpan dalam bentuk *dictionary* yang dikonversi menjadi dataframe untuk disimpan dalam file json.

4.2.6 Pembuatan *Confusion matrix*

Pembuatan fungsi plot *confusion matrix* dengan tujuan untuk visualisasi dan mengevaluasi model dapat dilihat pada Kode Program 4.6.

```

1 def plot_confusion_matrix(cm, target_names, title, cmap=None,
2                           normalize=True):
3     if cmap is None:
4         cmap = plt.get_cmap('YlOrRd')
5
6     plt.figure(figsize=(8, 8))
7     plt.imshow(cm, interpolation='nearest', cmap=cmap)
8     plt.title(title)
9
10    if target_names is not None:
11        tick_marks = np.arange(len(target_names))
12        plt.xticks(tick_marks, target_names, rotation=45)
13        plt.yticks(tick_marks, target_names)
14
15    if normalize:
16        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
17
18        thresh = cm.max() / 1.5 if normalize else cm.max() / 2
19        for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
20            if normalize:
21                plt.text(j, i, "{:0.4f}".format(cm[i, j]), horizontalalignment="center", color="white" if cm[i, j] > thresh else "black")
22            else:
23                plt.text(j, i, "{:,}").format(cm[i, j]), horizontalalignment="center", color="white" if cm[i, j] > thresh else "black")
24
25    plt.tight_layout()

```

24	plt.ylabel('True label')
25	plt.show()

Kode Program 4.6 Plotting confusion matrix

Kode Program 4.6 dijelaskan sebagai berikut:

1. Baris 1: Pembuatan fungsi plot *confusion matrix* untuk memvisualisasi *confusion matrix* setiap fold dimulai dengan parameter modul *confusion matrix*, kelas target, *cmap* (peta konsep), dan kondisi normalisasi.
2. Baris 2-7: Deklarasi tema, ukuran, dan judul *cmap* (peta konsep) menggunakan *library matplotlib*.
3. Baris 9-12: Mengatur koordinat titik sumbu x dan titik sumbu y dengan nama label/target.
4. Baris 14-15: Mengatur nilai pada matriks apakah sudah dalam kondisi normalisasi atau tidak.
5. Baris 17-22: Nilai *threshold* yang digunakan untuk mengatur kondisi teks pada matriks.
6. Baris 23-25: Menampilkan gambar matriks menggunakan *matplotlib*.

4.2.7 Testing dan Evaluasi Model

Setelah melakukan proses *training* pada model maka proses selanjutnya adalah mengujinya menggunakan data *testing* dan mengevaluasinya menggunakan metrik pengukuran. Proses *testing* data dapat dilihat pada Kode Program 4.7 berikut:

1	def testing(fold, scenario):
2	for j in range(1, scenario+1):
3	print("skenario - ", j)
4	scenario_acc = []
5	scenario_pre = []
6	scenario_spe = []
7	scenario_sen = []
8	str_scenario = "scenario_" + str(j)
9	for k in range(1, fold+1):
10	print("fold - ", k)
11	model_path = "/content/gdrive/My Drive/" + str(str_scenario) + "/models/model_fold_" + str(k) + ".h5"
12	
13	# load data_fold from npz file (dict of arrays)
14	dict_data = load("/content/gdrive/My Drive/data_folds /data_fold_" + str(k) + ".npz")
15	

```

16     # extract arrays
17     X_train = dict_data['arr_0']
18     X_test = dict_data['arr_1']
19     y_train = dict_data['arr_2']
20     y_test = dict_data['arr_3']
21
22     target_test = np.argmax(y_test, axis=1)
23
24     mod = load_model(model_path)
25     model = mod.predict(X_test)
26     y_pred = np.argmax(model, axis=1)
27
28     cm = confusion_matrix(target_test, y_pred)
29     plot_confusion_matrix(cm, target_names=class_labels,
30                           title="Confusion matrix -fold"+str(k), normalize=False)
31
32     TP = np.diag(cm)
33     FP = cm.sum(axis=0) - TP
34     FN = cm.sum(axis=1) - TP
35     TN = cm.sum() - (FP + FN + TP)
36
37     print("Hasil K-Fold-", str(k))
38     print("FP,FN,TP,TN = ", FP, FN, TP, TN)
39     temp_acc = []
40     temp_pre = []
41     temp_spe = []
42     temp_sen = []
43
44     for i in range(len(TP)):
45         accuracy = (TP[i] + TN[i]) / (TP[i] + FN[i] + TN[i]
46 + FP[i])
47         precision = TP[i] / (TP[i] + FP[i])
48         specificity = TN[i] / (TN[i] + FP[i])
49         sensitivity = TP[i] / (TP[i] + FN[i])
50         temp_acc.append(accuracy)
51         temp_pre.append(precision)
52         temp_spe.append(specificity)
53         temp_sen.append(sensitivity)
54
55     print(class_labels)
56     print(" accuracy per class : ", temp_acc)
57     print(" precision per class : ", temp_pre)
58     print(" specificity per class : ", temp_spe)
59     print(" sensitivity per class : ", temp_sen)
60
61     print("")
62     print("==== Fold-"
63 "+str(k)+" metrics performance ===")
64     print("Accuracy : {:.4f};".format(sum(temp_acc) / len(temp_acc)))
65     print("Precision : {:.4f};".format(sum(temp_pre) / len(temp_pre)))
66     print("specificity : {:.4f};".format(sum(temp_spe) / len(temp_spe)))
67     print("sensitivity : {:.4f};".format(sum(temp_sen) / len(temp_sen)))
68
69     print("=====")
70
71     scenario_acc.append(sum(temp_acc) / len(temp_acc))

```

```

68     scenario_pre.append(sum(temp_pre) / len(temp_pre))
69     scenario_spe.append(sum(temp_spe) / len(temp_spe))
70     scenario_sen.append(sum(temp_sen) / len(temp_sen))
71
72     print("")
73     print("Average metrics performane per skenario")
74     print("Accuracy : ", scenario_acc)
75     print("AVG Accuracy : {:.0f};".format(sum(scenario_acc
76 )/len(scenario_acc)))
76     print("Precision : ", scenario_pre)
77     print("AVG Precision : {:.0f};".format(sum(scenario_pr
78 e)/len(scenario_pre)))
78     print("specificity : ", scenario_spe)
79     print("AVG Specificity : {:.0f};".format(sum(scenario_sp
80 e)/len(scenario_spe)))
80     print("sensitivity : ", scenario_sen)
81     print("AVG Sensitivity : {:.0f};".format(sum(scenario_
81 sen)/len(scenario_sen)))

```

Kode Program 4.7 *Testing* dan evaluasi model

Kode program 4.7 dapat dijelaskan sebagai berikut:

1. Baris 1: Pembuatan fungsi *testing* sekaligus mengevaluasinya dengan parameter input nilai jumlah *fold* dan jumlah skenario.
2. Baris 2-8: Fungsi ini mengandung perulangan bertingkat (*nested looping*) dengan skema perulangan setiap skenario dan setiap *fold* di dalamnya. Perulangan pertama (*j*) adalah perulangan skenario dimana memiliki 4 *list* inisiasi evaluasi metrik setiap *fold* dan variabel *string str_skenario* sebagai rekayasa lokasi direktorinya.
3. Baris 9-14: Perulangan kedua (*k*), melakukan perulangan sebanyak *fold*. Selanjutnya memuat model dan data *fold* yang telah disimpan sebelumnya.
4. Baris 16-20: Mengekstrak data array yang ada dalam *fold* seperti data *training* (*X_train*), data target *training* (*y_train*), data *testing*(*X_test*), dan data target *testing*(*y_test*).
5. Baris 23-26: Melakukan prediksi dengan model yang telah dimuat.
6. Baris 28-29: Setelah dilakukan *testing* maka *confusion matrix* hal yang wajib untuk mengevaluasi suatu model memanggil fungsi untuk melakukan *plotting confusion matrix*.
7. Baris 31-34: Inisiasi perhitungan TP, FP, FN, dan TN untuk mendapatkan nilainya setiap kelas.
8. Baris 38-41: Deklarasi 4 variabel list sementara berisi 4 metrik pengukuran.

9. Baris 43-51: perulangan ketiga (i) digunakan untuk mengevaluasi model menggunakan metrik pengukuran seperti akurasi, presisi, sensitivitas, dan spesifisitas sesuai dengan persamaan 2.8 -2.11 pada setiap kelasnya. Kemudian, Setiap nilai metrik pengukuran dimasukkan kedalam list sementara pada baris 38-41.
10. Baris 53-57: Menampilkan nilai tiap list pada variable list sementara dimana perindeksan sesuai dengan *class_labels*.
11. Baris 59-64: Setelah *looping* evaluasi per kelas (i) selesai, menghitung nilai evaluasi metrik keseluruhan satu *fold* yang didapatkan dari nilai-nilai evaluasi per kelas sebelumnya.
12. Baris 67-70: Merangkum nilai rata-rata evaluasi setiap *fold*.
13. Baris 73-81: Menampilkan hasil nilai evaluasi rata-rata pada setiap skenario.

4.3 Hasil Skenario Uji Coba

Uji coba dilakukan untuk mengetahui kinerja model dengan arsitektur *EfficientNetB5* dengan dataset citra penyakit daun jagung. Melakukan *training* model yang kemudian dilakukan proses *testing* dan mengevaluasinya untuk mendapatkan nilai akurasinya.

4.3.1 Skenario Uji Coba 1

Hasil uji coba *EfficientNetB5* dengan kombinasi *learning rate* = 0,001 dan *batch size* = 16 didapatkan hasil akurasi tiap kelas sebagai berikut yang dapat ditujukan pada Tabel 4.3.

Tabel 4.3 Hasil Akurasi tiap kelas pada skenario 1

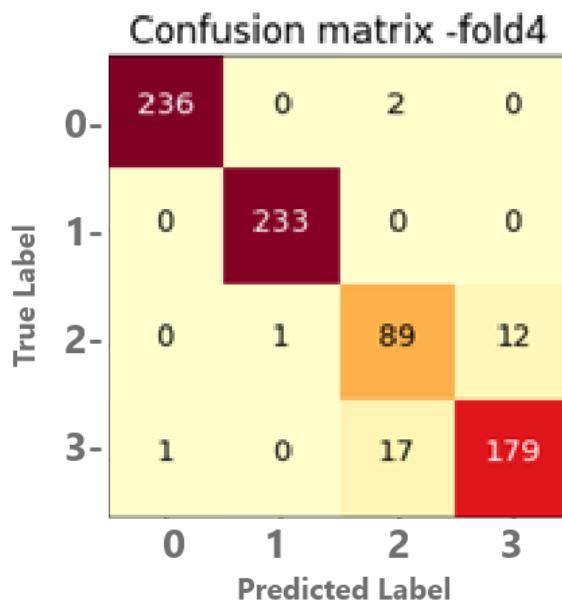
	<i>Common rust</i> (%)	<i>Healthy</i> (%)	<i>Cercospora leaf spot</i> (%)	<i>Northern Leaf Blight</i> (%)
<i>fold 1</i>	96,76	99,74	84,95	83,27
<i>fold 2</i>	73,41	86,25	82,49	94,03
<i>fold 3</i>	99,87	99,87	96,49	96,49
<i>fold 4</i>	99,61	99,87	95,84	96,10
<i>fold 5</i>	100,00	99,61	95,84	95,97

Pada Tabel 4.4 menunjukkan hasil perhitungan akurasi, presisi, spesifisitas, dan sensitivitas yang didapat setiap *fold* pada skenario 1.

Tabel 4.4 Hasil perhitungan metrik pengukuran skenario 1

	<i>fold 1</i> (%)	<i>fold 2</i> (%)	<i>fold 3</i> (%)	<i>fold 4</i> (%)	<i>fold 5</i> (%)	Rata-rata (%)
Akurasi	91,18	84,05	98,18	97,86	97,86	93,82
Presisi	83,32	74,76	95,29	93,82	94,91	88,42
spesifisitas	94,65	89,70	98,83	98,67	98,59	96,09
sensitivitas	81,85	71,27	94,32	94,32	92,92	86,94

Pada skenario 1 didapatkan nilai metrik pengukuran baik akurasi, presisi, spesifisitas, dan sensitivitas mendapatkan nilai optimal pada *fold* 3 masing-masing 98,18%, 95,29%, 98,83%, dan 94,32%. Nilai rata-rata makro pada skenario didapatkan dari hasil dari rata-rata pengukuran setiap *fold*. Nilai rata-rata makro metrik pengukuran pada skenario ini yakni, akurasi = 93,82%, presisi = 88,42%, spesifisitas = 96,09%, dan sensitivitas = 86,94%. Gambar 4.1 menunjukkan hasil *plotting confusion matrix* dari *fold* 4 pada skenario 1. Dimana pada *fold* ini nilai akurasi setiap kelas mencapai 95,84% hingga 99,87%.



Gambar 4.1 *Confusion matrix* skenario 1 (*fold* 4)

Kelas 0 mewakili kelas *common rust*, kelas 1 adalah daun *healthy*, kelas 2 adalah *cercospora leaf spot*, dan kelas 3 adalah *northern blight leaf*.

4.3.2 Skenario Uji Coba 2

Hasil uji coba *EfficientNetB5* dengan kombinasi *learning rate* = 0,0001 dan *batch size* = 16 didapatkan hasil akurasi tiap kelas sebagai berikut yang dapat ditujukan pada Tabel 4.5.

Tabel 4.5 Hasil Akurasi tiap kelas pada skenario 2

	<i>Common rust</i> (%)	<i>Healthy</i> (%)	<i>Cercospora leaf spot</i> (%)	<i>Northern Leaf Blight</i> (%)
<i>fold 1</i>	99,48	85,99	90,92	78,73
<i>fold 2</i>	99,87	98,57	89,88	89,11
<i>fold 3</i>	99,35	99,22	95,19	95,06
<i>fold 4</i>	99,74	99,74	93,51	93,77
<i>fold 5</i>	99,74	99,74	92,21	91,95

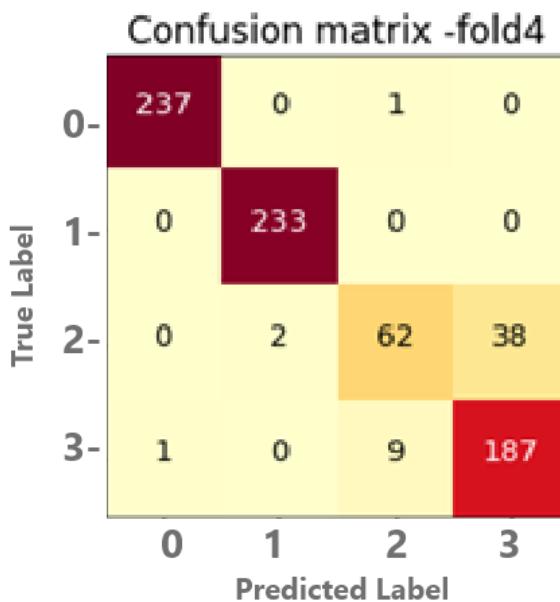
Pada Tabel 4.6 menunjukkan hasil perhitungan akurasi, presisi, spesifisitas, dan sensitivitas yang didapat setiap *fold* pada skenario 2.

Tabel 4.6 Hasil perhitungan metrik pengukuran skenario 2

	<i>fold 1</i> (%)	<i>fold 2</i> (%)	<i>fold 3</i> (%)	<i>fold 4</i> (%)	<i>fold 5</i> (%)	Rata-rata (%)
Akurasi	88,78	94,36	97,21	96,69	95,91	94,59
Presisi	83,56	86,18	92,46	91,99	90,66	88,97
spesifisitas	92,53	96,30	98,23	97,83	97,30	96,44
sensitivitas	73,35	81,90	92,56	88,82	86,14	84,55

Pada skenario 2 didapatkan nilai metrik pengukuran baik akurasi, presisi, spesifisitas, dan sensitivitas mendapatkan nilai optimal pada *fold 3* masing-masing 97,21%, 92,46%, 98,23%, dan 92,56%. Nilai rata-rata makro pada skenario didapatkan dari hasil dari rata-rata pengukuran setiap *fold*. Nilai rata-rata makro metrik pengukuran pada skenario ini yakni, akurasi = 94,59%, presisi = 88,97%, spesifisitas = 96,44%, dan sensitivitas = 84,55%. Gambar 4.2 menunjukkan hasil

plotting confusion matrix dari *fold 4* pada skenario 2. Dimana pada *fold* ini nilai akurasi setiap kelas mencapai 93,51% hingga 99,74%.



Gambar 4.2 *Confusion matrix* skenario 2 (*fold 4*)

Kelas 0 mewakili kelas *common rust*, kelas 1 adalah daun *healthy*, kelas 2 adalah *cercospora leaf spot*, dan kelas 3 adalah *northern blight leaf*.

4.3.3 Skenario Uji Coba 3

Hasil uji coba *EfficientNetB5* dengan kombinasi *learning rate* = 0,00001 dan *batch size* = 16 didapatkan hasil akurasi tiap kelas sebagai berikut yang dapat ditujukan pada Tabel 4.7.

Tabel 4.7 Hasil Akurasi tiap kelas pada skenario 3

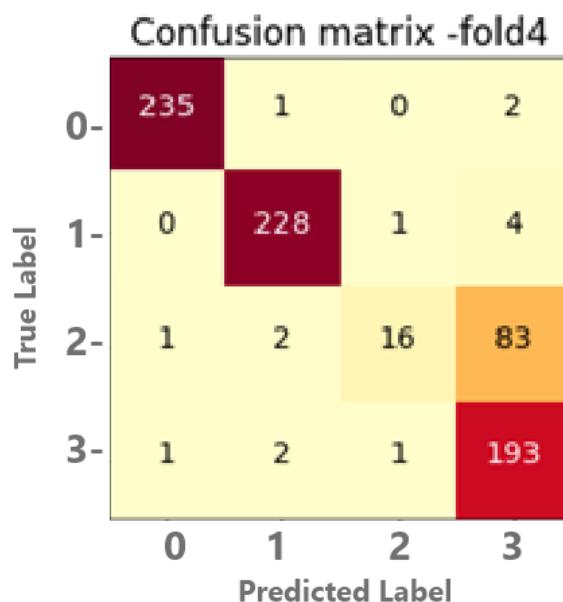
	<i>Common rust</i> (%)	<i>Healthy</i> (%)	<i>Cercospora leaf spot</i> (%)	<i>Northern Leaf Blight</i> (%)
<i>fold 1</i>	98,70	97,02	89,88	89,75
<i>fold 2</i>	98,70	97,80	86,77	86,38
<i>fold 3</i>	98,44	98,44	87,53	88,31
<i>fold 4</i>	99,35	98,70	88,57	87,92
<i>fold 5</i>	98,31	97,14	87,01	87,40

Pada Tabel 4.8 menunjukkan hasil perhitungan akurasi, presisi, spesifisitas, dan sensitivitas yang didapat setiap *fold* pada skenario 3.

Tabel 4.8 Hasil perhitungan metrik pengukuran skenario 3

	<i>fold 1</i> (%)	<i>fold 2</i> (%)	<i>fold 3</i> (%)	<i>fold 4</i> (%)	<i>fold 5</i> (%)	Rata-rata (%)
Akurasi	93,84	92,41	93,18	93,64	92,47	93,11
Presisi	89,44	81,57	83,15	88,58	80,86	84,72
spesifisitas	95,82	94,87	95,41	95,72	94,91	95,35
sensitivitas	79,47	73,72	76,18	77,56	74,51	76,29

Pada skenario 3 didapatkan nilai metrik pengukuran baik akurasi, presisi, spesifisitas, dan sensitivitas mendapatkan nilai optimal pada *fold* 1 masing-masing 93,84%, 89,44%, 95,82%, dan 79,47%. Nilai rata-rata makro pada skenario didapatkan dari hasil dari rata-rata pengukuran setiap *fold*. Nilai rata-rata makro metrik pengukuran pada skenario ini yakni, akurasi = 93,11%, presisi = 84,72%, spesifisitas = 95,35%, dan sensitivitas = 76,29%. Gambar 4.3 menunjukkan hasil *plotting confusion matrix* dari *fold* 4 pada skenario 3. Dimana pada *fold* ini nilai akurasi setiap kelas mencapai 87,92% hingga 99,35%.



Gambar 4.3 *Confusion matrix* skenario 3 (*fold* 4)

Kelas 0 mewakili kelas *common rust*, kelas 1 adalah daun *healthy*, kelas 2 adalah *cercospora leaf spot*, dan kelas 3 adalah *northern blight leaf*.

4.3.4 Skenario Uji Coba 4

Hasil uji coba *EfficientNetB5* dengan kombinasi *learning rate* = 0,001 dan *batch size* = 32 didapatkan hasil akurasi tiap kelas sebagai berikut yang dapat ditujukan pada Tabel 4.9.

Tabel 4.9 Hasil Akurasi tiap kelas pada skenario 4

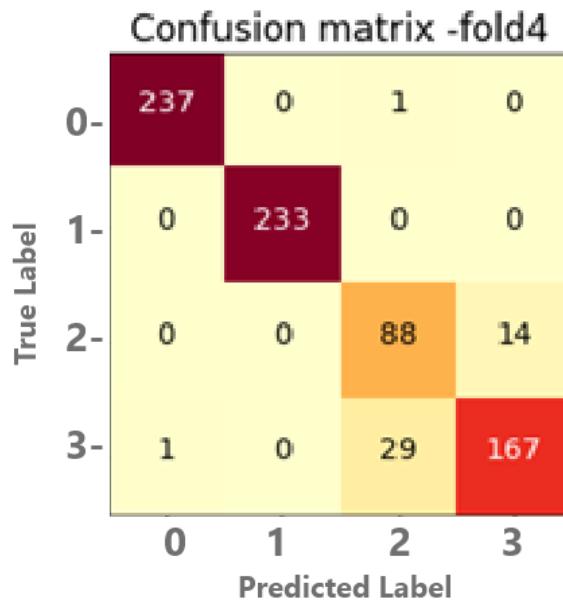
	Common rust (%)	Healthy (%)	Cercospora leaf spot (%)	Northern Leaf Blight (%)
<i>fold 1</i>	100,00	100,00	95,98	95,98
<i>fold 2</i>	100,00	99,09	94,68	94,81
<i>fold 3</i>	99,35	99,74	95,71	95,84
<i>fold 4</i>	99,74	100,00	94,29	94,29
<i>fold 5</i>	99,22	81,69	92,60	77,92

Pada Tabel 4.10 menunjukkan hasil perhitungan akurasi, presisi, spesifisitas, dan sensitivitas yang didapat setiap *fold* pada skenario 4. Nilai yang didapatkan dari metrik pengukuran baik akurasi, presisi, spesifisitas, dan sensitivitas mendapatkan nilai optimal pada *fold 1* masing-masing 97,99%, 94,76%, 98,71%, dan 93,63%.

Tabel 4.10 Hasil perhitungan metrik pengukuran skenario 4

	<i>fold 1</i> (%)	<i>fold 2</i> (%)	<i>fold 3</i> (%)	<i>fold 4</i> (%)	<i>fold 5</i> (%)	Rata-rata (%)
Akurasi	97,99	97,15	97,66	97,08	87,86	95,55
Presisi	94,76	92,55	93,59	91,61	81,14	90,73
Spesifisitas	98,71	98,15	98,52	98,22	91,99	97,12
Sensitivitas	93,63	91,29	93,53	92,66	75,49	89,32

Nilai rata-rata makro metrik pengukuran pada skenario ini yakni, akurasi = 95,55%, presisi = 90,73%, spesifisitas = 97,12%, dan sensitivitas = 89,32%. Gambar 4.4 menunjukkan hasil *plotting confusion matrix* dari *fold 4* pada skenario 4. Dimana pada *fold* ini nilai akurasi setiap kelas mencapai 94,29% hingga 100%.



Gambar 4.4 Confusion matrix skenario 4 (*fold 4*)

Kelas 0 mewakili kelas *common rust*, kelas 1 adalah daun *healthy*, kelas 2 adalah *cercospora leaf spot*, dan kelas 3 adalah *northern blight leaf*.

4.3.5 Skenario Uji Coba 5

Hasil uji coba *EfficientNetB5* dengan kombinasi *learning rate* = 0,0001 dan *batch size* = 32 didapatkan hasil akurasi tiap kelas sebagai berikut yang dapat ditujukan pada Tabel 4.11

Tabel 4.11 Hasil Akurasi tiap kelas pada skenario 5

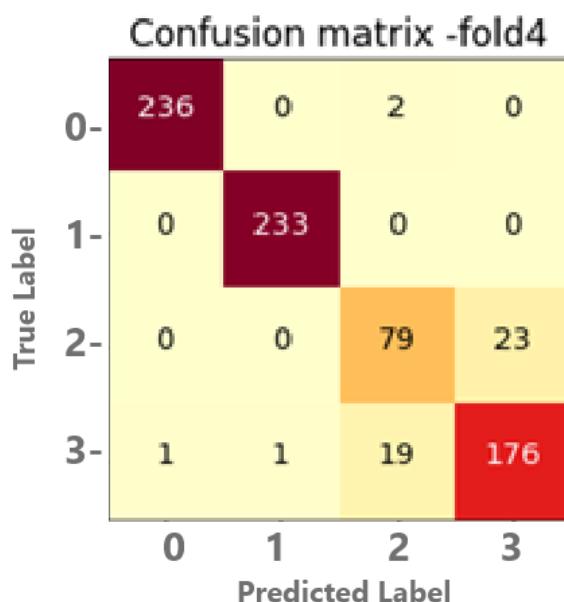
	<i>Common rust</i> (%)	<i>Healthy</i> (%)	<i>Cercospora leaf spot</i> (%)	<i>Northern Leaf Blight</i> (%)
<i>fold 1</i>	99,61	99,48	96,24	96,11
<i>fold 2</i>	99,74	97,80	88,59	87,94
<i>fold 3</i>	99,74	99,22	93,12	93,12
<i>fold 4</i>	99,61	99,87	94,29	94,29
<i>fold 5</i>	99,48	99,22	93,90	94,16

Pada Tabel 4.12 menunjukkan hasil perhitungan akurasi, presisi, spesifisitas, dan sensitivitas yang didapat setiap *fold* pada skenario 5.

Tabel 4.12 Hasil perhitungan metrik pengukuran skenario 5

	<i>fold 1</i> (%)	<i>fold 2</i> (%)	<i>fold 3</i> (%)	<i>fold 4</i> (%)	<i>fold 5</i> (%)	Rata-rata (%)
Akurasi	97,86	93,51	96,30	97,01	96,69	96,27
Presisi	94,53	85,79	91,00	91,65	91,54	90,90
spesifisitas	98,63	95,63	97,56	98,12	97,84	97,55
sensitivitas	93,87	77,74	87,82	91,49	89,75	88,13

Pada skenario 5 didapatkan nilai metrik pengukuran baik akurasi, presisi, spesifisitas, dan sensitivitas mendapatkan nilai optimal pada *fold* 1 masing-masing 97,86%, 94,53%, 98,63%, dan 93,87%. Nilai rata-rata makro pada skenario didapatkan dari hasil dari rata-rata pengukuran setiap *fold*. Nilai rata-rata makro metrik pengukuran pada skenario ini yakni, akurasi = 96,27%, presisi = 90,90%, spesifisitas = 97,55%, dan sensitivitas = 88,13%. Gambar 4.5 menunjukkan hasil *plotting confusion matrix* dari *fold* 4 pada skenario 5. Dimana pada *fold* ini nilai akurasi setiap kelas mencapai 94,29% hingga 99,87%.



Gambar 4.5 *Confusion matrix* skenario 5 (*fold* 4)

Kelas 0 mewakili kelas *common rust*, kelas 1 adalah daun *healthy*, kelas 2 adalah *cercospora leaf spot*, dan kelas 3 adalah *northern blight leaf*.

4.3.6 Skenario Uji Coba 6

Hasil uji coba *EfficientNetB5* dengan kombinasi *learning rate* = 0,00001 dan *batch size* = 32 didapatkan hasil akurasi tiap kelas sebagai berikut yang dapat ditujukan pada Tabel 4.13.

Tabel 4.13 Hasil Akurasi tiap kelas pada skenario 6

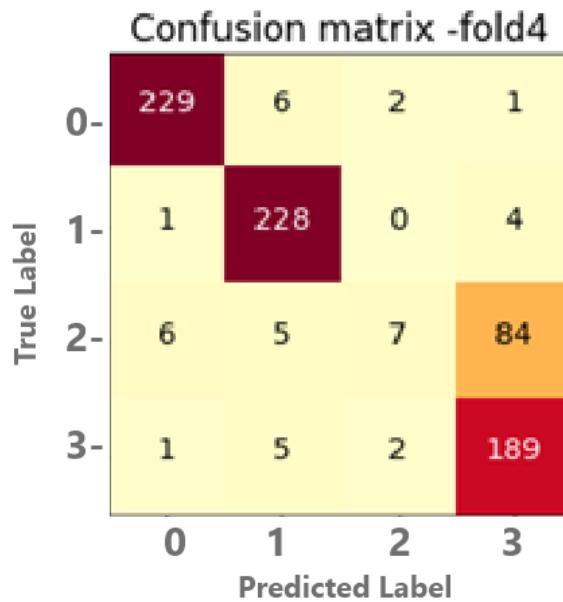
	<i>Common rust</i> (%)	<i>Healthy</i> (%)	<i>Cercospora leaf spot</i> (%)	<i>Northern Leaf Blight</i> (%)
<i>fold 1</i>	98,05	97,67	87,68	88,07
<i>fold 2</i>	99,61	99,09	87,55	87,55
<i>fold 3</i>	97,79	95,97	87,14	86,62
<i>fold 4</i>	97,79	97,27	87,14	87,40
<i>fold 5</i>	97,53	95,84	87,66	87,53

Pada Tabel 4.14 menunjukkan hasil perhitungan akurasi, presisi, spesifisitas, dan sensitivitas yang didapat setiap *fold* pada skenario 6 dan didapatkan nilai metrik pengukuran baik akurasi, presisi, spesifisitas, dan sensitivitas mendapatkan nilai optimal pada *fold 2* masing-masing 93,45%, 86,72%, 95,60%, dan 76,38%.

Tabel 4.14 Hasil perhitungan metrik pengukuran skenario 6

	<i>fold 1</i> (%)	<i>fold 2</i> (%)	<i>fold 3</i> (%)	<i>fold 4</i> (%)	<i>fold 5</i> (%)	Rata-rata (%)
Akurasi	92,87	93,45	91,88	92,40	92,14	92,55
Presisi	84,94	86,72	80,35	80,42	79,75	82,44
spesifisitas	95,18	95,60	94,48	94,85	94,74	94,97
sensitivitas	75,71	76,38	73,18	74,22	75,90	75,08

Nilai rata-rata makro pada skenario didapatkan dari hasil dari rata-rata pengukuran setiap *fold*. Nilai rata-rata makro metrik pengukuran pada skenario ini yakni, akurasi = 92,55%, presisi = 82,44%, spesifisitas = 94,97%, dan sensitivitas = 75,08%. Gambar 4.6 menunjukkan hasil *plotting confusion matrix* dari *fold 4* pada skenario 6. Dimana pada *fold* ini nilai akurasi setiap kelas mencapai 87,14% hingga 97,79%.



Gambar 4.6 Confusion matrix skenario 6 (fold 4)

Kelas 0 mewakili kelas *common rust*, kelas 1 adalah daun *healthy*, kelas 2 adalah *cercospora leaf spot*, dan kelas 3 adalah *northern blight leaf*.

4.4 Analisa Hasil Uji Coba

Tujuan utama penelitian ini adalah seberapa keberhasilan model *deep learning EfficientNetB5* dalam mengklasifikasikan penyakit daun tanaman jagung dengan menentukan nilai *hyperparameter learning rate* dan *batch size* yang optimal. Dalam penelitian ini, nilai rata-rata akurasi, sensitivitas, spesifisitas, dan presisi yang diperoleh dari semua model pada setiap skenario masing-masing.

Tabel 4.15 Rangkuman Hasil Akurasi Uji coba

	<i>fold 1</i>	<i>fold 2</i>	<i>fold 3</i>	<i>fold 4</i>	<i>fold 5</i>	Rata-rata
skenario 1	91,18	84,05	98,18	97,86	97,86	93,82
skenario 2	88,78	94,36	97,21	96,69	95,91	94,59
skenario 3	93,84	92,41	93,18	93,64	92,47	93,11
skenario 4	97,99	97,15	97,66	97,08	87,86	95,55
skenario 5	97,86	93,51	96,30	97,01	96,69	96,27
skenario 6	92,87	93,45	91,88	92,40	92,14	92,55

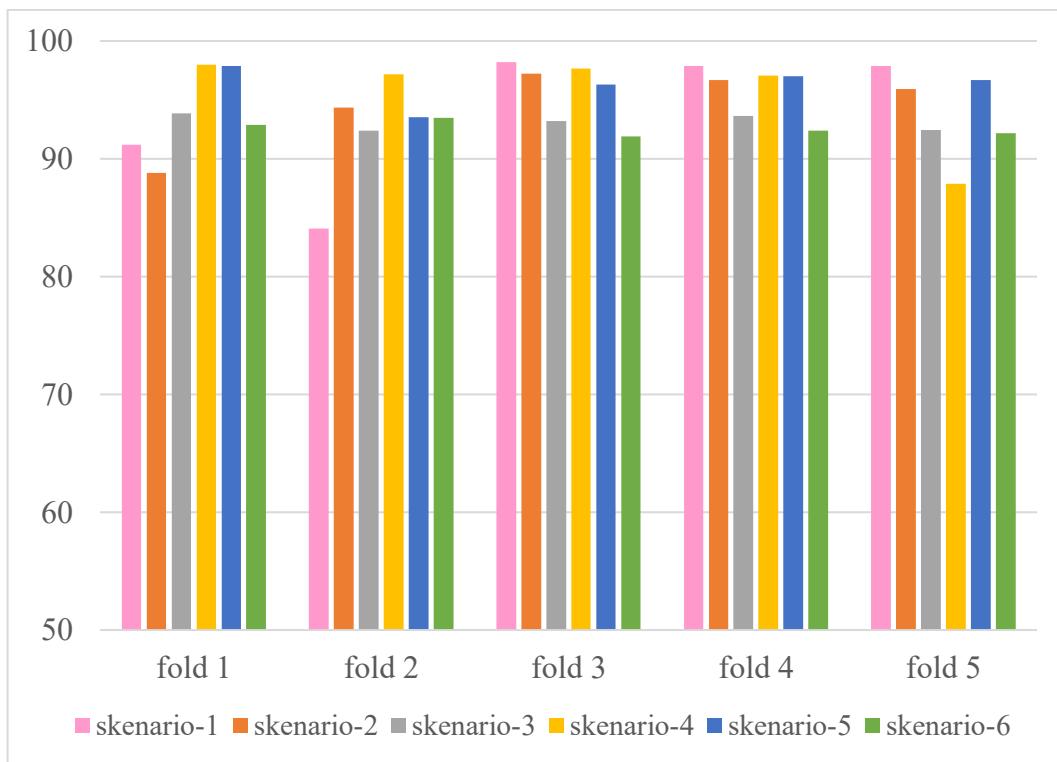
Sebelum melakukan hasil rata-rata evaluasi model terlebih dahulu merangkum hasil akurasi dari setiap skenario hasil uji coba pada sub-bab sebelumnya yang disajikan dalam bentuk Tabel 4.15. Semua model memperoleh nilai akurasi yang dekat satu sama lain. Pada *fold* 1 dan *fold* 2 nilai tertinggi didapatkan pada skenario 4 masing-masing, 97,99% dan 97,15%. Pada *fold* 3, 4, dan 5 skenario 1 mendapatkan nilai tertinggi pada ketiga *fold* tersebut masing-masing, 98,18%, 97,86%, dan 97,86%.

Tabel 4.16 Hasil rata-rata evaluasi model setiap skenario

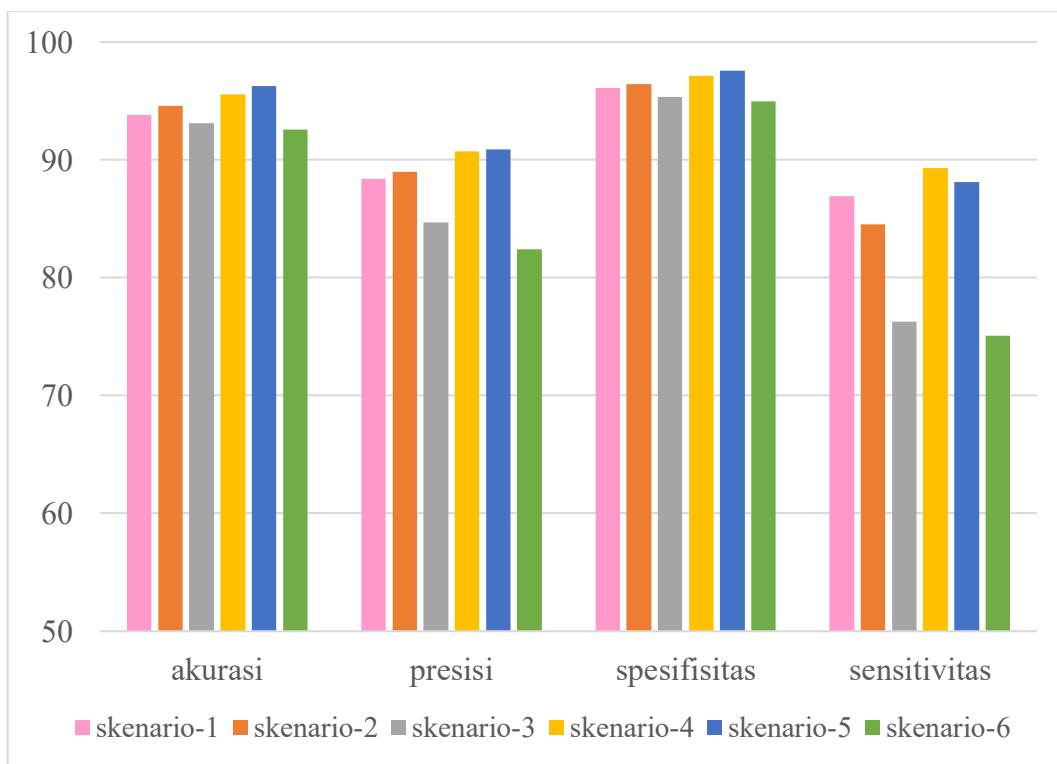
	Rata-rata Akurasi (%)	Rata-rata Presisi (%)	Rata-rata Spesifisitas (%)	Rata-rata Sensitivitas (%)
skenario 1	93,82	88,42	96,09	86,94
skenario 2	94,59	88,97	96,44	84,55
skenario 3	93,11	84,72	95,35	76,29
skenario 4	95,55	90,73	97,12	89,32
skenario 5	96,27	90,90	97,55	88,13
skenario 6	92,55	82,44	94,97	75,08

Hasil evaluasi rata-rata model setiap skenario dapat dilihat pada Tabel 4.16. Skenario 5 secara berturut-turut unggul dalam nilai rata-rata akurasi, presisi, dan spesifisitas. Disisi lain, skenario 4 mendapatkan nilai rata-rata sensitivitas terbaik dari semua skenario. Hasil rata-rata uji coba model klasifikasi dengan kombinasi nilai *learning rate* dan *batch size* yang ditunjukkan pada tabel hasil rata-rata evaluasi didapati bahwa skenario 5 memperoleh akurasi tertinggi dengan nilai akurasi sebesar 96,27% dengan kombinasi *hyperparameter* *learning rate* = 0,0001 dan *batch size* = 32. Sedangkan hasil akurasi rata-rata terendah didapatkan pada skenario 6 dengan nilai sebesar 92,55% dengan kombinasi *hyperparameter* *learning rate* 0,00001 dan *batch size* = 32.

Penggambaran kinerja model juga dapat dilihat pada grafik visualisasi perbandingan akurasi hasil uji coba dan perbandingan hasil rata-rata evaluasi model menggunakan diagram batang masing-masing ditunjukkan pada Gambar 4.7 dan Gambar 4.8. Model skenario 5 mencapai kinerja terbaik pada dataset berkisar antara 70% hingga 100%. Sedangkan nilai akurasi pada model skenario 5 untuk setiap kelas pada setiap *fold* berkisar antara 87,94% hingga 99,87%.



Gambar 4.7 Komparasi akurasi hasil uji coba



Gambar 4.8 Komparasi hasil rata-rata evaluasi model

TP, TN, FP, FN, akurasi, presisi, spesifisitas, dan sensitivitas adalah nilai-nilai yang diperoleh oleh model skenario 5 untuk masing-masing kelas dalam setiap *fold* disajikan pada Tabel 4.17, Tabel 4.18, Tabel 4.19, Tabel 4.20, dan Tabel 4.21.

Tabel 4.17 Kinerja klasifikasi skenario 5 pada setiap kelas (*fold 1*)

	FP	FN	TP	TN	ACC	PRE	SPE	SEN
<i>Common_rust</i>	0	3	236	532	99,61	100,0	100,0	98,74
<i>healthy</i>	0	4	228	539	99,48	100,0	100,0	98,28
<i>Cercospora leaf spot</i>	11	18	85	657	96,24	88,54	98,35	82,52
<i>Northern Leaf Blight</i>	22	8	189	552	96,11	89,57	96,17	95,94

Tabel 4.18 Kinerja klasifikasi skenario 5 pada setiap kelas (*fold 2*)

	FP	FN	TP	TN	ACC	PRE	SPE	SEN
<i>Common_rust</i>	0	2	237	532	99,74	100,0	100,0	99,16
<i>healthy</i>	17	0	232	522	97,80	93,17	96,85	100,0
<i>Cercospora leaf spot</i>	5	83	20	663	88,59	80,00	99,25	19,42
<i>Northern Leaf Blight</i>	78	15	182	496	87,94	70,00	86,41	92,39

Tabel 4.19 Kinerja klasifikasi skenario 5 pada setiap kelas (*fold 3*)

	FP	FN	TP	TN	ACC	PRE	SPE	SEN
<i>Common_rust</i>	1	1	237	531	99,74	99,58	99,81	99,58
<i>healthy</i>	6	0	233	531	99,22	97,49	98,88	100,0
<i>Cercospora leaf spot</i>	11	42	60	657	93,12	84,51	98,35	58,82
<i>Northern Leaf Blight</i>	39	14	183	534	93,12	82,43	93,19	92,89

Tabel 4.20 Kinerja klasifikasi skenario 5 pada setiap kelas (*fold 4*)

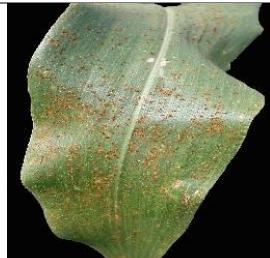
	FP	FN	TP	TN	ACC	PRE	SPE	SEN
<i>Common_rust</i>	1	2	236	531	99,61	99,58	99,81	99,16
<i>healthy</i>	1	0	233	536	99,87	99,57	99,81	100,0
<i>Cercospora leaf spot</i>	21	23	79	647	94,29	79,00	96,86	77,45
<i>Northern Leaf Blight</i>	23	21	176	550	94,29	88,44	95,99	89,34

Tabel 4.21 Kinerja klasifikasi skenario 5 pada setiap kelas (*fold 5*)

	FP	FN	TP	TN	ACC	PRE	SPE	SEN
<i>Common_rust</i>	2	2	236	530	99,48	99,16	99,62	99,16
<i>healthy</i>	6	0	232	532	99,22	97,48	98,88	100,0
<i>Cercospora leaf spot</i>	14	33	70	653	93,90	83,33	97,90	67,96
<i>Northern Leaf Blight</i>	29	16	181	544	94,16	86,19	94,94	91,88

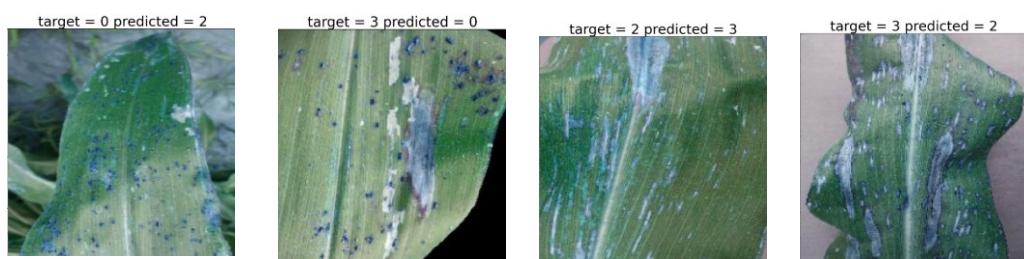
Pada Tabel 4.17 - Tabel 4.21 menunjukkan kinerja klasifikasi memiliki nilai FP atau FN tinggi pada kelas *cercospora leaf spot* dan *northern leaf blight* berarti banyak terjadi kesalahan klasifikasi dikarenakan adanya karakter data pada setiap kelasnya seperti yang ditunjukkan pada Tabel 4.22.

Tabel 4.22 Karakteristik data setiap kelas

Data citra	Kelas	Karakteristik kelas
	<i>Common rust</i>	Memiliki ciri daun berkarat dengan latar belakang citra berwarna hitam yang telah tersegmentasi
	<i>Healthy</i>	Memiliki ciri daun sehat penuh pada citra
	<i>Cercospora leaf spot</i>	Memiliki ciri daun persegi panjang berwarna coklat hingga abu-abu yang sejajar dan membentang di antara urat daun serta dengan latar belakang citra berwarna ungu

Data citra	Kelas	Karakteristik kelas
	<i>Nothern leaf blight</i>	Memiliki bercak berbentuk oval dan memanjang seperti bentuk <i>ellips</i> dengan latar belakang citra berwarna ungu seperti kelas <i>cercospora leaf spot</i> .

Dari Tabel 4.22 dapat disimpulkan bahwa kelas cercospora leaf spot dan northern leaf blight saling tumpang tindih dalam mengklasifikasinya. Pada kelas *healthy* dan kelas *common rust* memiliki tingkat keberhasilan klasifikasi tertinggi dikarenakan ciri atau karakteristik data tersebut. *Fold 4* menjadi klasifikasi terbaik dengan kesalahan klasifikasi sebanyak 46 data. Contoh data yang diklasifikasi salah dapat ditunjukkan pada Gambar 4.10.

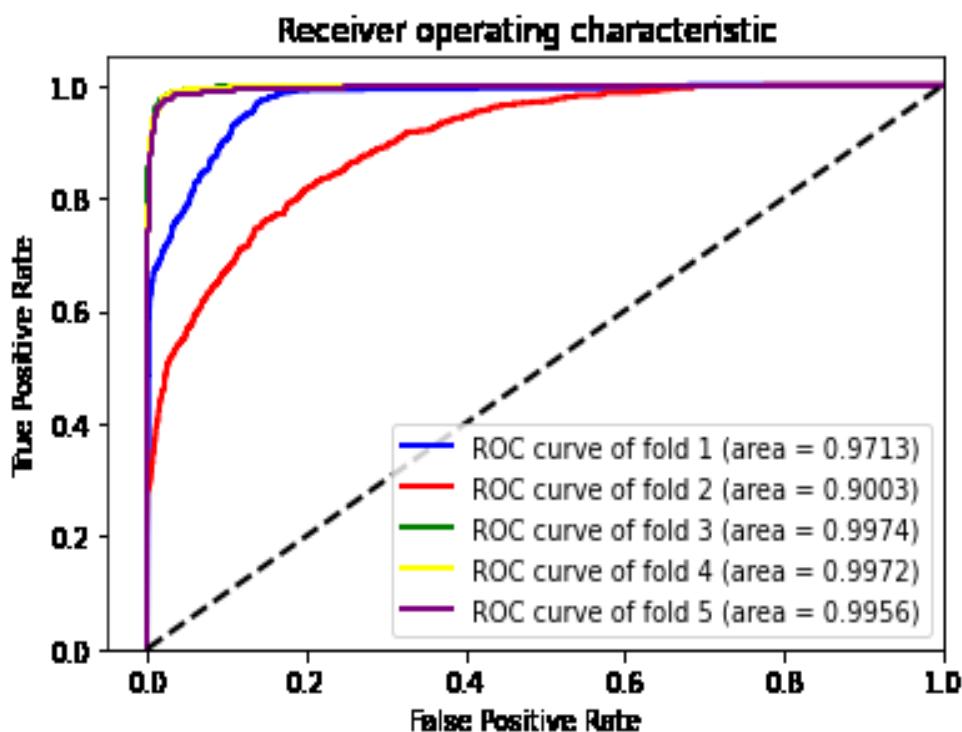


Gambar 4.9 Contoh data kesalahan klasifikasi

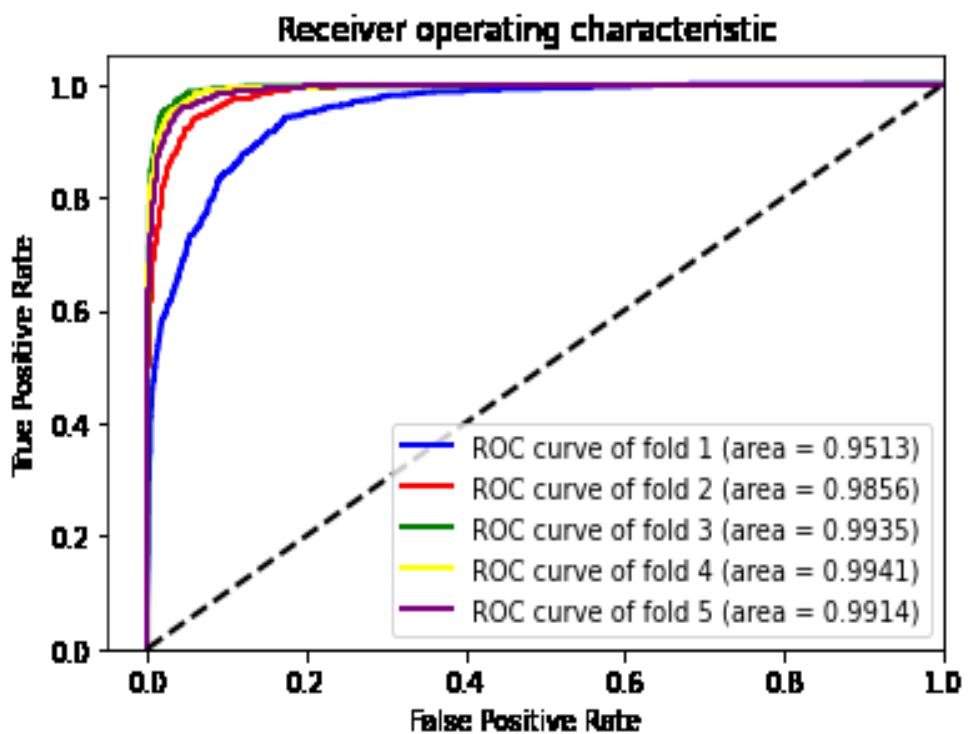
Keempat gambar pada Gambar 4.9 berturut-turut menunjukkan kesalahan seperti, gambar pertama kelas *common rust* yang terdeteksi sebagai kelas *cercospora leaf spot* dikarenakan gambar tidak tersegmentasi seperti data pada kelas nya. Kemudian, gambar kedua kelas *northern leaf blight* yang terdeteksi sebagai kelas *common rust* karena cirinya citra tersegmentasi dan berkarat layaknya *common rust*. Kedua gambar terakhir secara umum memiliki ciri yang mirip karena keduanya memiliki kemiripan pada morfologis daun dan juga latar belakang citra yang berwarna sama dibandingkan dengan kelas *healthy* dan kelas *common rust*. Bagi mata orang awam, kedua gambar tersebut tampaknya mewakili penyakit yang sama. Namun, dataset menyediakannya dalam kelas penyakit yang berbeda. Keasalahan ini merupakan kesalahan dalam kumpulan data tetapi hanya ahli

patologi profesional yang dapat mengkonfirmasinya. Seringkali kelas *cercospora leaf spot* dideteksi kelas *northern leaf blight* ataupun sebaliknya.

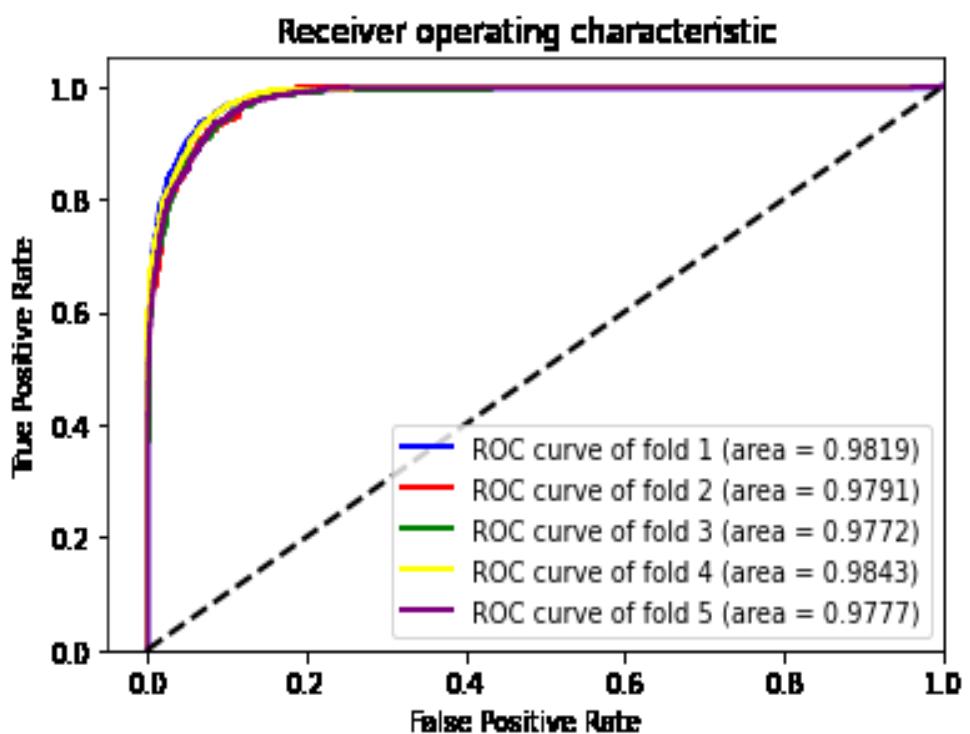
Sementara itu, komparasi kurva ROC untuk setiap *fold* pada tiap skenario dapat dilihat pada gambar-gambar dibawah. Gambar 4.10 menggambarkan kurva ROC pada skenario 1 semua *fold* mendapatkan hasil yang sangat baik (*excellent classification*) kecuali pada *fold* 2 yang termasuk di kategori *very good classification*. Gambar 4.11 menggambarkan kurva ROC pada skenario 2 rata-rata setiap *fold* berada di *excellent classification*. Gambar 4.12 menggambarkan kurva ROC pada skenario 3 semua *fold* berada pada tingkat *excellent classification*. Skenario 4 ditunjukkan pada Gambar 4.13 dengan masing-masing fold memperoleh kategori *excellent classification*. Skenario 5 ditunjukkan pada Gambar 4.14 dimana mendapatkan nilai AUC tertinggi tiap *fold* nya daripada skenario lainnya. Skenario 6 ditunjukkan pada Gambar 4.15 dapat dilihat semua *fold* mendapatkan kategori *excellent classification*.



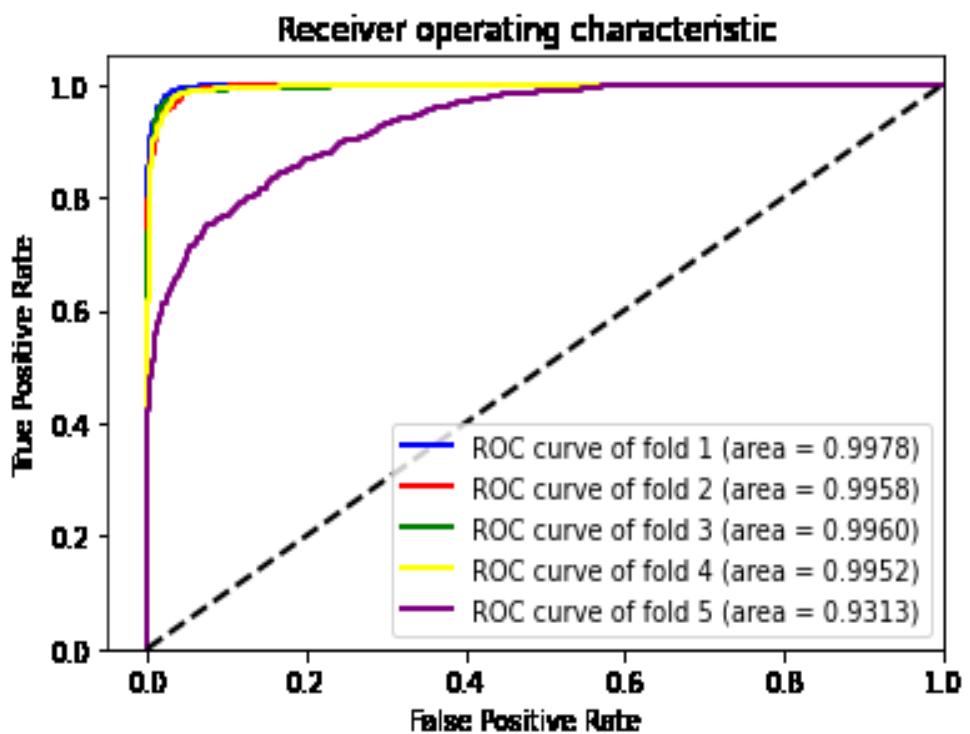
Gambar 4.10 Kurva ROC skenario 1



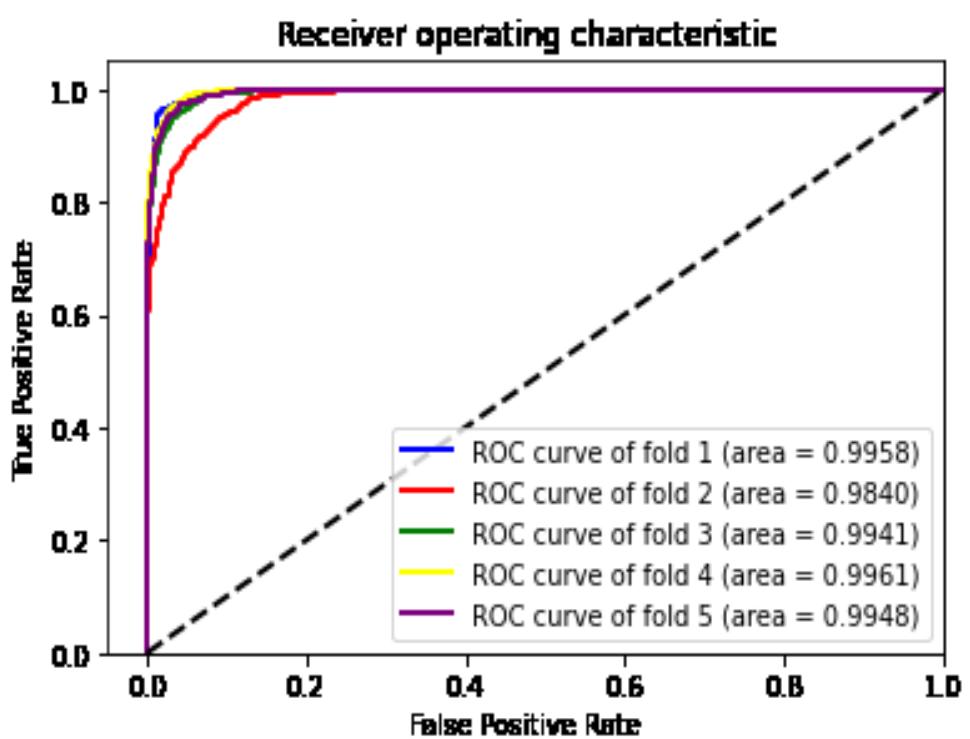
Gambar 4.11 Kurva ROC skenario 2



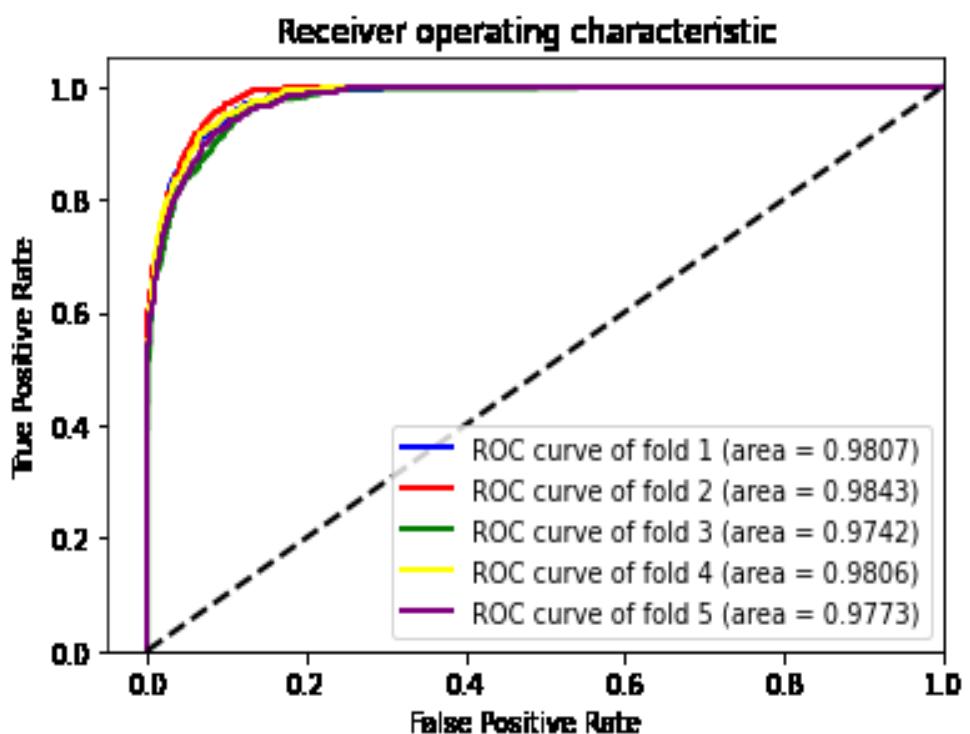
Gambar 4.12 Kurva ROC skenario 3



Gambar 4.13 Kurva ROC skenario 4



Gambar 4.14 Kurva ROC skenario 5



Gambar 4.15 Kurva ROC skenario 6

BAB 5 PENUTUP

5.1 Kesimpulan

Dalam Penilitian ini, model *deep learning EfficientNetB5* diusulkan untuk mengklasifikasi citra penyakit daun tanaman jagung yang terbagi menjadi 4 kelas. Berdasarkan hasil analisis yang telah dilakukan diperoleh skenario 5 dengan *hyperparameter learning rate = 0,0001* dan *batch size = 32* mendapatkan hasil kinerja yang terbaik dengan mendapatkan nilai akurasi sebesar 96,27%, presisi sebesar 90,90%, spesifisitas sebesar 97,55%, dan sensitivitas sebesar 88,13%. Disisi lain, nilai AUC yang didapatkan skenario 5 mengungguli dari rata-rata skenario lainnya dengan nilai setiap *fold* masing-masing sebesar 99,58%, 98,40%, 99,41%, 99,61%, dan 99,48%. Berdasarkan penelitian dapat disimpulkan bahwa nilai *learning rate = 0,0001* dan *batch size = 32* mendapatkan hasil kinerja yang lebih optimal daripada kombinasi lainnya. Oleh karena itu, penentuan nilai yang tepat pada *hyperparameter learning rate* dan *batch size* dapat mempengaruhi hasil kinerja model yang dilatih.

5.2 Saran

Saran yang dapat penulis berikan beberapa agar dapat dijadikan bahan untuk mengembangkan penelitian ini pada penelitian selanjutnya. Saran tersebut adalah sebagai berikut:

1. Menggunakan kombinasi parameter yang lebih banyak seperti *optimizer*, fungsi aktivasi, jumlah *epoch* dan lain-lain.
2. Memperbanyak data dengan cara melakukan augmentasi untuk mendapatkan data yang lebih variatif.
3. Menggunakan arsitektur dari keluarga *EfficientNet* lainnya seperti *EfficientNetB7* yang memiliki parameter lebih tinggi dari *EfficientNet* lainnya
4. Melakukan penambahan proses segmentasi citra seperti penghapusan latar belakang.

REFERENSI

- [1] Sudjono, M. S., “Penyakit Jagung dan Pengendaliannya,” in *Buku Jagung*, Bogor: Puslitbang Tanaman Pangan, pp. 381–394.
- [2] Amzeri, A., (2018), “Tinjauan Perkembangan Pertanian Jagung di Madura dan Alternatif Pengolahan Menjadi Biomaterial,” *Rekayasa*, vol. 11, pp. 74–86.
- [3] Lu, J., Tan, L., and Jiang, H., (2021), “Review on Convolutional Neural Network (CNN) Applied to Plant Leaf Disease Classification,” *Agriculture*, vol. 11, p. 707.
- [4] Zhang, Z. Y., He, X. Y., Sun, X. H., Guo, L. M., Wang, J. H., and Wang, F. S., (2015), “Image Recognition of Maize Leaf Disease Based on GA-SVM,” *Chem. Eng. Trans.*, vol. 46, pp. 199–204.
- [5] Khan, S., Rahmani, H., Shah, S. A. A., Bennamoun, M., Medioni, G., and Dickinson, S., (2018), *A Guide to Convolutional Neural Networks for Computer Vision*. Morgan & Claypool.
- [6] Sibiya, M. and Sumbwanyambe, M., (2019), “A Computational Procedure for the Recognition and Classification of Maize Leaf Diseases Out of Healthy Leaves Using Convolutional Neural Networks,” *AgriEngineering*, vol. 1, pp. 119–131.
- [7] Syarief, M. and Setiawan, W., (2020), “Convolutional neural network for maize leaf disease image classification,” *Telkomnika (Telecommunication Comput. Electron. Control.)*, vol. 18, pp. 1376–1381.
- [8] Waheed, A., Goyal, M., Gupta, D., Khanna, A., Hassanien, A. E., and Pandey, H. M., (2020), “An optimized dense convolutional neural network model for disease recognition and classification in corn leaf,” *Comput. Electron. Agric.*, vol. 175, p. 105456.
- [9] Atila, Ü., Uçar, M., Akyol, K., and Uçar, E., (2021), “Plant leaf disease classification using EfficientNet deep learning model,” *Ecol. Inform.*, vol.

61, p. 101182.

- [10] Tan, M. and Le, Q., (2019), “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks,” in *Proceedings of the 36th International Conference on Machine Learning*, pp. 6105–6114, PMLR.
- [11] He, K., Zhang, X., Ren, S., and Sun, J., (2016), “Deep Residual Learning for Image Recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, IEEE.
- [12] Zagoruyko, S. and Komodakis, N., (2016), “Wide Residual Networks,” in *Proceedings of the British Machine Vision Conference (BMVC)*, pp. 87.1–87.12, BMVA Press.
- [13] Kandel, I. and Castelli, M., (2020), “The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset,” *ICT Express*, vol. 6, pp. 312–315.
- [14] Nugroho, A. and Suhartanto, H., (2020), “Hyper-Parameter Tuning based on Random Search for DenseNet Optimization,” in *2020 7th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE)*, pp. 96–99.
- [15] Chang, C.-I. and Ren, H., (2000), “An experiment-based quantitative and comparative analysis of target detection and image classification algorithms for hyperspectral imagery,” *Geosci. Remote Sensing, IEEE Trans.*, vol. 38, pp. 1044–1063.
- [16] Arif, T. M., (2020), *Introduction to Deep Learning for Engineers: Using Python and Google Cloud Platform*. Morgan & Claypool.
- [17] Ioffe, S. and Szegedy, C., (2015), “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” in *International conference on machine learning*, pp. 448–456, PMLR.
- [18] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C., (2018), “Mobilenetv2: Inverted residuals and linear bottlenecks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–

4520, IEEE.

- [19] Chamarty, A., (2020), “Fine-Tuning of Learning Rate for Improvement of Object Detection Accuracy,” in *2020 IEEE India Council International Subsections Conference (INDISCON)*, pp. 135–141.
- [20] Sokolova, M. and Lapalme, G., (2009), “A systematic analysis of performance measures for classification tasks,” *Inf. Process. Manag.*, vol. 45, pp. 427–437.
- [21] Fan, J., Upadhye, S., and Worster, A., (2006), “Understanding receiver operating characteristic (ROC) curves,” *CJEM*, vol. 8, pp. 19–20.
- [22] Gajowniczek, K., Z\k{a}bkowski, T., and Szupiluk, R., (2014), “ESTIMATING THE ROC CURVE AND ITS SIGNIFICANCE FOR CLASSIFICATION MODELS’ASSESSMENT,” *Metod. Ilościowe w Badaniach Ekon.*, vol. 15, pp. 382–391.
- [23] Geetharamani, G. and Arun Pandian, J., “Data for: Identification of Plant Leaf Diseases Using a 9-layer Deep Convolutional Neural Network,” *Mendeley Data*, vol. 1. 2019.
- [24] Mom, F., “Summary & Implementation of Deep Learning research paper in Tensorflow/Pytorch.,” 2020. https://github.com/3outeille/Research-Paper-Summary/blob/master/src/architecture/efficientnet/tensorflow_2/efficientnet_tf2.ipynb (accessed Mar. 13, 2023).