IDENTIFIKASI PENYAKIT PADA TANAMAN PADI MENGGUNAKAN MINIVGGNET BERBASIS *MOBILE APP*

SKRIPSI

Oleh :
AHMAD FAIZ
NIM. 210605110171

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2025

IDENTIFIKASI PENYAKIT PADA TANAMAN PADI MENGGUNAKAN MINIVGGNET BERBASIS *MOBILE APP*

SKRIPSI

Diajukan kepada:
Universitas Islam Negeri Maulana Malik Ibrahim Malang
Untuk memenuhi Salah Satu Persyaratan dalam
Memperoleh Gelar Sarjana Komputer (S.Kom)

Oleh :
AHMAD FAIZ
NIM. 210605110171

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2025

HALAMAN PERSETUJUAN

IDENTIFIKASI PENYAKIT PADA TANAMAN PADI MENGGUNAKAN MINIVGGNET BERBASIS *MOBILE APP*

SKRIPSI

Oleh :
AHMAD FAIZ
NIM. 210605110171

Telah Diperiksa dan Disetujui untuk Diuji:
Tanggal: 5 Desember 2025

Pembimbing I,	Pembimbing II,
Okta Qomaruddin Aziz, M. Kom	Dr. Cahyo Crysdian, M. Cs
NIP. 19911019 201903 1 013	NIP. 19740424 200901 1 008

Mengetahui,
Ketua Program Studi Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Supriyono, M. Kom
NIP. 19841010 201903 1 012

HALAMAN PENGESAHAN

IDENTIFIKASI PENYAKIT PADA TANAMAN PADI MENGGUNAKAN MINIVGGNET BERBASIS *MOBILE APP*

SKRIPSI

Oleh :
AHMAD FAIZ
NIM. 210605110171

Telah Dipertahankan di Depan Dewan Penguji Skripsi
dan Dinyatakan Diterima Sebagai Salah Satu Persyaratan
Untuk Memperoleh Gelar Sarjana Komputer (S.Kom)
Tanggal: 11 Desember 2025

Susunan Dewan Penguji

Ketua Penguji	: Dr. Irwan Budi Santoso, M. Kom NIP. 19770103 201101 1 004	( )
Anggota Penguji I	: Nur Fitriyah Ayu Tunjung Sari, M.Cs NIP. 19911226 202012 2 001	( )
Anggota Penguji II	: Okta Qomaruddin Aziz, M. Kom NIP. 19911019 201903 1 013	( )
Anggota Penguji III	: Dr. Cahyo Crysdian, M. Cs NIP. 19740424 200901 1 008	( )

Mengetahui dan Mengesahkan,
Ketua Program Studi Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Supriyono, M. Kom
NIP. 19841010 201903 1 012

PERNYATAAN KEASLIAN TULISAN

Saya yang bertanda tangan di bawah ini:

Nama	: AHMAD FAIZ
NIM	: 210605110171
Fakultas / Program Studi	: Sains dan Teknologi / Teknik Informatika
Judul Skripsi	: Identifikasi Penyakit Pada Tanaman Padi Menggunakan MiniVGGNet Berbasis *Mobile App*

Menyatakan dengan sebenarnya bahwa Skripsi yang saya tulis ini benar-benar merupakan hasil karya saya sendiri, bukan merupakan pengambil alihan data, tulisan, atau pikiran orang lain yang saya akui sebagai hasil tulisan atau pikiran saya sendiri, kecuali dengan mencantumkan sumber cuplikan pada daftar pustaka.

Apabila dikemudian hari terbukti atau dapat dibuktikan skripsi ini merupakan hasil jiplakan, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Malang, 11 Desember 2025
Yang membuat pernyataan,

Ahmad Faiz
NIM. 210605110171

MOTTO

”It is better to die for a dream, than to live asleep.”

”Sebutlah nama-Nya, tetap di jalan-Nya.”

~ 33x - Perunggu

“Di tanganmu kuasa”

~ Evaluasi - Hindia

HALAMAN PERSEMBAHAN

Segala puji syukur kehadirat Allah SWT atas limpahan rahmat dan karunianya yang telah diberikan kepada penulis, sehingga dapat menyelesaikan skripsi ini. Sholawat serta salam semoga selalu tercurahkan kepada Nabi Muhammad SAW yang telah memberikan syafaat dan karunianya kepada umat muslim.

Skripsi yang telah ditulis oleh penulis, tidak mungkin akan terselesaikan jika tidak ada dukungan baik secara jasmani, rohani, dan materi. Atas dasar itu, penulis memberikan persembahan yang sebesar-besarnya kepada:

Mama tercinta,

St. Nur Syamsudhuha, S. Sos dengan kasih dan cintanya yang selalu mendukung dalam pembuatan skripsi ini dari awal hingga akhir yang dukungannya tidak pernah hilang dan berkurang sedikitpun.

Ayah tercinta,

Dr. Muh. Yazid Abd. Rachim Gege, M, Pd dengan tekad dan wawasannya yang selalu menyemangati dan membimbing dalam proses pembuatan skripsi hingga skripsi ini berhasil selesai.

Kakak tercinta,

Sitti Amani Yazid, S. E dengan pengalaman dan cintanya kepada adiknya yang selalu memberikan motivasi untuk tetap bertahan dan melanjutkan skripsi hingga selesai.

Diri Sendiri,

Yang telah berhasil menyelesaikan pendidikan di perguruan tinggi dan memikul tanggung jawab sebagai mahasiswa

KATA PENGANTAR

Segala puji syukur senantiasa penulis panjatkan ke hadirat Allah Subhanahu wa Ta'ala atas limpahan rahmat, taufik, dan hidayah-Nya, sehingga penulis dapat menyelesaikan skripsi yang berjudul "Identifikasi Penyakit Pada Tanaman Padi Menggunakan MiniVGGNet berbasis *Mobile App*" dengan baik. Shalawat serta salam semoga senantiasa tercurah kepada Nabi Muhammad SAW, suri teladan umat manusia, beserta keluarga, sahabat, dan pengikutnya hingga akhir zaman.

Ucapan terima kasih sebanyak-banyaknya saya ucapkan kepada:

1. Prof. Dr. Hj. Ilfi Nur Diana, M.Si, CAHRM, CRMP selaku Rektor UIN Maulana Malik Ibrahim Malang.
2. Dr. Agus Mulyono, M. Kes selaku Dekan Fakultas Sains dan Teknologi UIN Maulana Malik Ibrahim Malang.
3. Supriyono, M. Kom selaku Kepala Program Studi Teknik Informatika Fakultas Sains dan Teknologi UIN Maulana Malik Ibrahim Malang.
4. Okta Qomaruddin Aziz, M. Kom Cs selaku dosen pembimbing kedua yang dengan tulus memberikan masukan, motivasi, serta pandangan kritis yang sangat membantu dalam penyelesaian karya ini.
5. Dr. Cahyo Crysdian, M. Cs selaku dosen pembimbing pertama skripsi yang telah mengarahkan dan membimbing di perkuliahan di bidang Teknik Informatika hingga dalam proses penyelesaian tugas akhir ini.
6. Dr. Irwan Budi Santoso, M. Kom selaku Ketua Dewan Penguji, atas koreksi dan saran yang membangun selama ujian skripsi berlangsung.
7. Nur Fitriyah Ayu Tunjung Sari, M.Cs selaku penguji kedua, atas evaluasi dan masukkan yang telah memperbaiki kualitas skripsi ini.
8. Kedua orang tua tercinta, Mama dan Ayah, serta Kakak saya yang telah menjadi sumber inspirasi, kekuatan, serta doa yang tidak pernah diputus.

Terima kasih atas cinta dan pengorbanan yang telah menjadi dasar dari setiap langkah perjalanan penulis.

9. Segenap dosen di Program Studi Teknik Informatika Fakultas Sains dan Teknologi UIN Maulana Malik Ibrahim Malang yang telah mendidik dan memberikan ilmu pengetahuan selama ini.
10. Teman-teman dan sahabat Jurusan Teknik Informatika 2021 ”ASTER” yang telah memberikan semangat dan bantuan dalam mengerjakan skripsi.
11. Serta semua pihak yang tidak dapat disebutkan satu persatu, yang telah bantu selama ini.

Saya telah berusaha sebaik mungkin dalam menyusun laporan ini, namun tidak menutup kemungkinan bahwa masih terdapat kekurangan atau kesalahan di dalamnya. Oleh karena itu, saya sangat mengharapkan adanya kritik dan saran yang konstruktif untuk memperbaiki dan menyempurnakan laporan penelitian skripsi ini.

Malang, 11 Desember 2025

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Faiz, Ahmad. 2025. **Identifikasi Penyakit Pada Tanaman Padi Menggunakan MiniVGGNet Berbasis *Mobile App***. Skripsi, Program Studi Teknik Informatika Fakultas Sains dan Teknologi, Universitas Islam Negeri Maulana Malik Ibrahim Malang. Pembimbing: (I) Okta Qomaruddin Aziz, M. Kom (II) Dr. Cahyo Crysdian, M. Cs.

Kata Kunci: *Convolutional Neural Network*, MiniVGGNet, Penyakit Padi, *Mobile App*.

Penyakit pada tanaman padi seringkali mengakibatkan petani mengalami gagal panen dan penurunan kualitas padi. Umumnya, penyakit tanaman padi disebabkan oleh serangan patogen seperti jamur, bakteri, dan virus. Penelitian ini bertujuan untuk mengidentifikasi 5 kelas yang terdiri dari 4 jenis penyakit pada tanaman padi, seperti hawar daun bakteri, blas, tungro, bercak coklat dan daun padi sehat dengan menggunakan metode *Convolutional Neural Network* (CNN). Arsitektur yang digunakan pada penelitian ini adalah MiniVGGNet dan kustomisasi MiniVGGNet yang dilatih dengan menggunakan 883 citra *no-background* dan 1104 citra *background*. Berdasarkan hasil pengujian, model MiniVGGNet menunjukkan performa yang lebih unggul dengan akurasi 98%, *precision* 98%, *recall* 98%, dan *F1-Score* 98%. Sedangkan model kustom MiniVGGNet mendapatkan nilai akurasi 88%, *precision* 88%, *recall* 88%, dan F1-Score 87%. Meskipun akurasi yang didapatkan pada model kustom lebih rendah, namun model tersebut memiliki jumlah parameter yang jauh lebih kecil (242.981 parameter) dibandingkan dengan model MiniVGGNet (2.167.797 parameter). Model kustom merupakan model yang unggul dalam efisiensi ukuran, sehingga model tersebut lebih efektif berjalan pada perangkat *mobile* berbasis Android. Kemudian dilakukan pengujian *usability* menggunakan SUS *Score*, didapatkan rata-rata skor 80,25 dengan *adjective rating* ”*Excellent*” dan *grade letter* ”B” pada perangkat *mobile* dengan model kustom. Namun demikian, kedua model tersebut tetap dapat diterapkan pada perangkat *mobile* dengan mempertimbangkan spesifikasi perangkat yang digunakan. Apabila ingin diterapkan pada perangkat yang memiliki spesifikasi lebih rendah, dapat menggunakan model kustom, namun, jika memiliki spesifikasi yang lebih tinggi, maka dapat menggunakan model MiniVGGNet.

ABSTRACT

Faiz, Ahmad. 2025. **Identification of Rice Plant Diseases Using MiniVGGNet Based on Mobile App**. Thesis, Department of Informatics Engineering, Faculty of Science and Technology, Maulana Malik Ibrahim State Islamic University of Malang. Supervisors: (I) Okta Qomaruddin Aziz, M. Kom (II) Dr. Cahyo Crysdian, M. Cs.

Keywords: Convolutional Neural Network, MiniVGGNet, Rice Diseases, Mobile App.

Rice plant diseases often lead to crop failure and a decline in rice quality for farmers. Generally, these diseases are caused by pathogens such as fungi, bacteria, and viruses. This study aims to classify 5 classes of rice leaves, consisting of 4 disease types (bacterial leaf blight, blast, tungro, brown spot) and healthy leaves, using the Convolutional Neural Network (CNN) method. The architectures employed in this study are MiniVGGNet and a customized MiniVGGNet, trained using 883 no-background images and 1104 background images. Based on the testing results, the MiniVGGNet model demonstrated superior performance with 98% accuracy, 98% precision, 98% recall, and a 98% F1-Score. Meanwhile, the custom MiniVGGNet model achieved an accuracy of 88%, precision of 88%, recall of 88%, and an F1-Score of 87%. Although the accuracy of the custom model is lower, it possesses a significantly smaller number of parameters (242,981 parameters) compared to the MiniVGGNet model (2,167,797 parameters). The custom model excels in size efficiency, making it more effective for deployment on Android-based mobile devices. Furthermore, usability testing using the System Usability Scale (SUS) yielded an average score of 80.25, with an "Excellent" adjective rating and a "B" grade letter on mobile devices using the custom model. Nevertheless, both models can be implemented on mobile devices by considering the device specifications. For devices with lower specifications, the custom model is recommended, whereas the MiniVGGNet model is suitable for devices with higher specifications.

مستخلص البحث

فائز، احمد. ٢٠٢٥. تحديد أمراض نبات الأرز باستخدام MiniVGGNet القائم على تطبيقات الهاتف المحمول. بحث جامعي . بحث تخرّج، برنامج دراسة تقنية المعلومات، كلية العلوم والتكنولوجيا، جامعة مولانا مالك إبراهيم الإسلامية الحكومية مالانغ.المشرفان: (١) أوكتا قمر الدين عزيز، املاجستر، (٢) د. جهيو كريسداين، املاجستر.

الكلمات المفتاحية: لشبكة العصبية التلافيفية ، MiniVGGNet، أمراض الأرز، تطبيق الهاتف المحمول.

غالبًا ما تؤدي أمراض نبات الأرز إلى فشل المحاصيل وانخفاض جودة الأرز لدى المزارعين. بشكل عام، تنجم أمراض الأرز عن هجمات مسببات الأمراض مثل الفطريات والبكتيريا والفيروسات. تهدف هذه الدراسة إلى تصنيف ٥ فئات تتكون من ٤ أنواع من أمراض نبات الأرز (لفحة الأرز البكتيرية، واللفحة، والتونغرو، والتبقع البني) والأوراق السليمة باستخدام طريقة الشبكة العصبية التلافيفية (CNN). البنية المستخدمة في هذه الدراسة هي MiniVGGNet وMiniVGGNet المخصصة (Custom) التي تم تدريبها باستخدام ٨٨٣ صورة بدون خلفية و ١١٠٤ صورة بخلفية. بناءً على نتائج الاختبار، أظهر نموذج MiniVGGNet أداءً فائقًا بدقة ٩٨٪، ودقة (Precision) ٩٨٪، واستدعاء (Recall) ٩٨٪، ودرجة F1-Score ٩٨٪. في حين حصل النموذج المخصص (Custom MiniVGGNet) على دقة ٨٨٪، ودقة (Precision) ٨٨٪، واستدعاء (Recall) ٨٨٪، ودرجة F1-Score ٨٧٪. على الرغم من أن الدقة التي تم الحصول عليها في النموذج المخصص أقل، إلا أن هذا النموذج يحتوي على عدد أصغر بكثير من المعلمات (٢٤٢،٩٨١ معلمة) مقارنة بنموذج MiniVGGNet (٢،١٦٧،٧٩٧ معلمة). يتفوق النموذج المخصص في كفاءة الحجم، مما يجعله أكثر فعالية للتشغيل على الأجهزة المحمولة التي تعمل بنظام أندرويد (Android). ثم تم إجراء اختبار قابلية الاستخدام باستخدام مقياس قابلية الاستخدام للنظام (SUS)، وتم الحصول على متوسط درجة ٨٠.٢٥ بتقدير وصفي "ممتاز" (Excellent) وحرف التقدير "B" على الأجهزة المحمولة باستخدام النموذج المخصص. ومع ذلك، يمكن تطبيق كلا النموذجين على الأجهزة المحمولة مع مراعاة مواصفات الجهاز المستخدم. إذا كان المراد التطبيق على أجهزة ذات مواصفات أقل، يمكن استخدام النموذج المخصص، أما إذا كانت المواصفات أعلى، فيمكن استخدام نموذج MiniVGGNet.

BAB I

PENDAHULUAN

1.1 Latar Belakang

Padi (*Oryza Sativa*) merupakan komoditas utama pada sektor pertanian Indonesia. Padi menjadi komoditas tanaman pangan yang penting dan juga memiliki peran dalam pembangunan di bidang pertanian (Saragi *et al*. 2023). Data oleh *Food and Agriculture Organization* (FAO) tahun 2025, menunjukkan bahwa Indonesia menempati urutan keempat sebagai produsen padi terbanyak di dunia sebanyak 34 juta ton (*Food Outlook – Biannual Report on Global Food Markets*, 2025). Tingginya tingkat produksi padi di Indonesia disebabkan oleh mayoritas penduduk Indonesia yang menjadikan beras sebagai makanan pokok. Menurut Badan Perakitan dan Modernisasi Pertanian (2024), beras dan jagung adalah makanan pokok sebagian masyarakat Indonesia. Namun, tingginya minat beras sebagai bahan makanan pokok tidak sesuai dengan jumlah produksi padi di Indonesia yang menurun pada tahun sebelumnya. Berdasarkan data Kementrian Pertanian Republik Indonesia, Indonesia mengalami penurunan jumlah produksi padi pada tahun 2024 sekitar 2,25%. Penurunan jumlah produksi tersebut salah satunya disebabkan oleh penyakit tanaman padi yang menyerang lahan pertanian.

Penyakit pada tanaman padi merupakan salah satu faktor yang menyebabkan penurunan jumlah dan mutu hasil pertanian (Prajapati *et al.* 2017). Umumnya penyakit yang menyerang daun tanaman padi disebabkan oleh bakteri dan jamur (Walascha *et al.* 2021). Data dari *website International Rice Research Institute*

(IRRI) *Rice Knowledge Bank*, Gejala umum yang ditimbulkan dari penyakit tanaman padi cenderung sama, seperti hawar daun bakteri, blas, tungro, dan bercak coklat. Beragam penyakit yang menyerang tanaman padi sering kali membuat petani kurang tepat dalam mengambil keputusan untuk menangani permasalahan tersebut. Hal itu disebabkan karena terbatasnya informasi yang didapatkan oleh petani dari penyuluhan dan pengetahuan yang dimiliki berdasarkan pengalaman pribadi (Sheila *et al.* 2023). Keterlambatan dalam mengidentifikasi jenis penyakit tanaman padi bisa mengakibatkan kegagalan panen dan dapat merugikan petani. Maka dari itu, diperlukan sistem yang dapat melakukan identifikasi jenis penyakit tanaman padi secara otomatis dan akurat.

Identifikasi jenis penyakit dapat dilakukan dengan menggunakan model pengenalan pola citra dengan memanfaatkan *artificial intelligence* (Rijal *et al.* 2024). Dalam konteks pengenalan pola citra, *deep learning* mengambil peranan penting. *Deep Learning* adalah sub-unit *machine learning* yang ada di *artificial intelligence*. Model *deep learning* lebih unggul dalam menyelesaikan tugas rumit. Salah satunya mengidentifikasi atau mendeskripsikan gambar. Karena di dalam model tersebut, *features extraction* dilakukan di dalam sistem secara otomatis yang tidak dilakukan secara manual di luar sistem seperti *machine learning*. Model ini menggunakan teknologi *neural network* atau jaringan saraf, yang memiliki input, *hidden layer* atau lapisan tersembunyi, dan output.

Metode *Convolutional Neural Network* (CNN) merupakan bagian dari *deep learning* yang memiliki tingkat akurasi yang tinggi dalam melakukan *image processing* dan pengenalan gambar visual (Yuliany *et al.* 2022). CNN merupakan

metode *supervised learning* untuk mengklasifikasikan atau mengidentifikasi citra gambar yang memiliki label. Selain itu, metode CNN dapat melakukan *features extraction* secara otomatis dan dapat mengklasifikasikan atau mengidentifikasikan citra gambar secara lebih akurat (Anggiratih *et al.* 2021). Tingkat ketepatan dalam melakukan identifikasi citra gambar bergantung pada arsitektur yang digunakan. Arsitektur harus memiliki tingkat akurasi yang tinggi dan ringan untuk digunakan pada aplikasi *mobile*, khususnya Android.

Dengan memanfaatkan kemajuan teknologi, CNN dapat membantu pekerjaan manusia, khususnya petani dalam menjaga dan merawat padi mereka agar tumbuh subur dengan kondisi yang sehat. Jika padi tidak dirawat dengan baik dan benar, maka akan mengakibatkan kerusakan pada tanaman padi tersebut sehingga padi menjadi tidak sehat dan kualitasnya berkurang. Di dalam Al-Qur'an, Allah SWT telah bersabda sebagai berikut:

وَإِذَا تَوَلَّىٰ سَعَىٰ فِي ٱلْأَرْضِ لِيُفْسِدَ فِيهَا وَيُهْلِكَ ٱلْحَرْثَ وَٱلنَّسْلَ ۗ وَٱللَّهُ لَا يُحِبُّ ٱلْفَسَادَ ﴿٢٠٥﴾

"Dan apa ia berpaling (dari kamu), ia berjalan di bumi untuk mengadakan kerusakan padanya, dan merusak tanam-tanaman dan binatang ternak, dan Allah tidak menyukai kebinasaan" (Q.S. Al-Baqarah : 205).

Menurut penjelasan tafsir Al-Mukhtashar, apabila ia (orang munafik) berpaling darimu dan berpisah denganmu, ia berusaha untuk membuat kerusakan di muka bumi dengan berbuat maksiat, merusak tanam-tanaman dan membunuh binatang ternak. Sedangkan Allah tidak menyukai kerusakan di muka bumi dan tidak mencintai orang-orang yang suka membuat kerusakan. Dari tafsir tersebut, dapat diambil pembelajaran kita tidak boleh menjadi orang yang merusak tanaman

dan membunuh binatang. Orang yang seperti itu adalah orang yang munafik atau perkataan dan perbuatannya saling bertentangan (Hazmi *et al.* 2024). Maka dari itu, untuk mencerminkan seorang mukmin yang taat dan patuh kita perlu menjaga dan merawat tanaman dengan tidak merusaknya dan memberikan perawatan dan penanganan yang tepat. Dalam penerapan dalam menjaga kualitas dan kesehatan tanaman, khususnya padi, diperlukan penelitian terdahulu yang menjadi berfungsi sebagai fondasi ilmiah.

Penelitian sebelumnya yang dilakukan oleh Hairani & Widiyaningtyas (2024) melakukan deteksi penyakit pada tanaman padi menggunakan metode CNN dan augmentasi data. Data citra yang diperoleh berasal dari *website kaggle* sebanyak 120 citra. Penyakit padi yang dideteksi adalah *bacterial leaf blight*, *brown spot*, dan *leaf smut*. Data dibagi menjadi 80 % data latih dan 20% data uji. Hasil yang didapatkan dari deteksi penyakit pada daun padi sebesar 99,7%. Penelitian yang dilakukan oleh Latifah *et al.* (2024) mengklasifikasikan sel darah merah pada pasien talasemia minor menggunakan metode CNN dengan arsitektur MiniVGGNet. Total data citra yang digunakan dalam penelitian tersebut sebanyak 7.108 citra. Data citra yang telah dikumpulkan, kemudian dilakukan proses *resize* dengan ukuran 32x32 piksel dan augmentasi data. Hasil nilai evaluasi yang didapatkan pada penelitian tersebut menggunakan MiniVGGNet adalah akurasi 96,00%, *recall* 96,05%, presisi 96,00%, dan F1-Score 95,95%. Penelitian lain yang dilakukan oleh Winanda *et al.* (2021) melakukan deteksi penyakit pada daun gambir menggunakan metode CNN dengan arsitektur MiniVGGNet. Data citra yang diperoleh berasal dari observasi langsung di Kecamatan Kapur IX Kabupaten

50 Kota, Provinsi Sumatera Barat. Data citra yang dikumpulkan berjumlah 60 dengan pembagian 30 gambir kualitas baik dan 30 kualitas buruk. Penelitian tersebut melakukan proses augmentasi data untuk menambahkan varian data. Kemudian, dilakukan pembagian data untuk data latih dan data uji. Data dibagi menjadi 70 % data latih dan 30% data uji. Hasil uji akurasi yang didapatkan sebesar 99,87%.

Berdasarkan penjelasan sebelumnya, penelitian ini menggunakan metode CNN dengan arsitektur MiniVGGNet. Arsitektur tersebut memiliki akurasi yang tinggi dalam mengidentifikasi citra. Akurasi sangat penting dalam metode CNN karena dapat menentukan tingkat keberhasilan suatu model arsitektur (Riza Agung Firmansyah *et al.* 2023). Namun untuk diterapkan pada perangkat *mobile*, dibutuhkan arsitektur yang memiliki jumlah parameter yang lebih kecil. Sehingga, pada penelitian ini ditambahkan arsitektur kustomisasi MiniVGGNet agar dapat berjalan secara efisien di perangkat *mobile*, khususnya Android.

1.2 Pernyataan Masalah

Seberapa baik performa metode CNN dengan menggunakan arsitektur MiniVGGNet dan kustomisasi MiniVGGNet dalam mengidentifikasi jenis penyakit pada daun padi?

1.3 Batasan Masalah

Batasan masalah pada penelitian ini untuk membatasi aspek penelitian agar penelitian lebih terfokus dan terarah. Beberapa batasan tersebut sebagai berikut:

1. Data citra yang dikumpulkan berasal dari Setiady (2021), vbookshelf (2020), Sankalana (2025), dan Anshul (2024). Data bersumber dari *website kaggle*.
2. Identifikasi dilakukan pada 5 kelas citra daun padi, meliputi 4 jenis penyakit, yaitu hawar daun bakteri (*bacterial leaf blight*), blas (*leaf blast*), tungro, dan bercak coklat *(brown spot*), serta kelas daun padi sehat (*healthy*)

1.4 Tujuan Penelitian

Mengukur tingkat performa arsitektur MiniVGGNet dan kustom MiniVGGNet dalam mengidentifikasi jenis penyakit tanaman padi.

1.5 Manfaat Penelitian

1. Bagi Kementrian Pertanian, diharapkan dapat mewujudkan visi dan misi dalam mewujudkan pertanian yang modern dan ketahanan pangan bagi negara.
2. Bagi *Non Govermental Organization* (NGO), diharapkan dapat memberikan wawasan atau informasi kepada Petani terkait penggunaan teknologi dalam mengidentifikasi penyakit yang menyerang tanaman padi.
3. Bagi petani, penelitian ini diharapkan dapat membantu dalam mengenali jenis penyakit secara akurat, sehingga petani dapat memberikan perawatan yang tepat pada padi.

BAB II

STUDI PUSTAKA

2.1 Identifikasi Penyakit Pada Tanaman Padi

Padi memiliki peranan penting di Indonesia. Padi adalah tanaman pangan yang menjadi sumber makanan pokok bagi masyarakat di Indonesia (Nurkayah *et al.* 2024). Namun, padi juga rentan terhadap berbagai penyakit. Serangan penyakit pada tanaman padi merupakan salah satu hal yang sering terjadi di Indonesia (Sukarta *et al.* 2018). Penyakit yang umum pada tanaman padi adalah hawar daun bakteri, blas, tungro, dan bercak coklat (*brown spot*). Penyakit tersebut menyebabkan kurangnya hasil panen padi dan membuat kerugian kepada petani (Zakaria *et al.* 2024).

Penyakit yang umum menyerang tanaman padi adalah hawar daun bakteri (*bacterial leaf blight*). Penyakit ini disebabkan oleh bakteri *xanthomonas campestris pv. Oryzae* (Yogaswara *et al.* 2023). Bakteri tersebut dapat menyebar karena angin dan percikan air hujan atau irigasi. Akibat dari penyakit tersebut adalah daun yang berubah warna menjadi kuning hingga berwarna jerami, serta bergaris-garis. sehingga membuat padi tersebut menjadi layu lalu mati.

Penyakit blas (*leaf blast*) merupakan salah satu penyakit yang menyerang tanaman padi. Penyakit ini disebabkan oleh jamur *pyricularia oryzae* (Marwan *et al.* 2021) yang menyerang tanaman padi di bagian daun, ruas batang, tangkai malai, cabang malai, dan bulir. Jamur ini menimbulkan bercak berwarna putih keabu-abuan dengan berbentuk belah ketupat yang memiliki tepi warna coklat.

Tungro adalah penyakit yang menyebabkan tanaman padi menjadi kerdil. Penyakit ini disebabkan oleh virus *Rice Tungro Bacilliform Virus* (RTBV) dan *Rice Tungro Spherical Virus* (RTSV), atau bisa salah satunya (Fiddin *et al.* 2021). Virus ini disebarkan oleh hama wereng yang disebut *nephotettix virescens*. Padi yang terjangkit virus tersebut menimbulkan gejala tanaman padi menjadi kerdil dan daunnya berubah menjadi kuning dari pangkal hingga bawah padi.

Selain ketiga penyakit tersebut, bercak coklat (*brown spot*) juga menjadi penyakit yang umum dijumpai pada tanaman padi. Penyakit ini disebabkan oleh jamur *cochliobolus miyabeanus* (Sunandar & Sutopo, 2024). Jamur ini dapat bertahan pada benih dan menyebar ke tanaman lainnya. Padi yang terpapar jamur tersebut mengalami gejala bercak coklat bundar atau oval dengan lingkar kuning. Jika bercaknya meluas maka tepiannya menjadi coklat dan memiliki warna abu-abu di tengahnya.

Penyakit-penyakit yang menyerang tanaman padi tentunya merugikan Petani dari berbagai aspek, terutama dalam hasil panen. Berdasarkan laporan *Food and Agriculture Organization* (FAO) tahun 2021, produksi tanaman global menurun hingga 40% disebabkan oleh hama dan penyakit tanaman (FAO, 2021). Ini diperburuk dengan laporan yang dikeluarkan oleh Badan Pusat Statistika (BPS) tahun 2024, yang menunjukkan penurunan hasil produksi padi sebesar 1,55% atau 0,84 juta ton Gabah Kering Giling (GKG) dibanding tahun 2023. Masalah tersebut sangat berdampak bagi masyarakat dan khususnya Petani dari segi ekonomi, kesulitan dalam memenuhi kebutuhan sehari-hari, dan potensi gagal panen. Salah

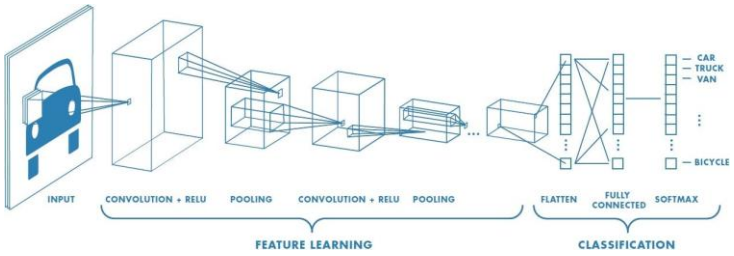
satu solusi untuk mengatasi permasalah yang dialami oleh Petani adalah dengan memanfaatkan perkembangan teknologi.

2.2 MiniVGGNet

Metode *Convolutional Neural Network* (CNN) merupakan salah satu algoritma *deep learning* yang banyak digunakan dalam beberapa tahun terakhir, khususnya untuk mengolah data berbentuk gambar, video, dan sinyal audio. Metode ini bekerja dengan cara meniru sistem penglihatan manusia dalam mengenali objek, yaitu dengan mengekstraksi fitur secara bertahap, mulai dari bentuk sederhana seperti garis atau tepi, hingga pola yang lebih kompleks seperti tekstur dan objek secara keseluruhan (O'Shea & Nash, 2015). Penelitian terdahulu menunjukkan tingkat keberhasilan yang tinggi dengan menggunakan metode CNN. Salah satu penelitian yang dilakukan oleh Sheila *et al.* (2023) adalah mengklasifikasikan penyakit pada daun padi menggunakan CNN yang mendapatkan nilai akurasi sebesar 93,75%. Hasil akurasi yang tinggi membuat CNN sebagai metode yang efektif dan efisien dalam mengenali objek tertentu.

Secara umum, metode CNN memiliki dua tahap, yaitu *feature extraction* dan *classification* (Nurilmiyanti Wardhani *et al.* 2024). *Feature extraction* berfungsi untuk mengektrasi fitur yang terdapat dari input yang berupa data citra. Fitur yang diekstrak bisa berupa tepi, garis, sudut, tekstur, atau fitur yang lainnya. Di dalam tahapan *feature extraction*, umumnya terdapat *convolutional layer*, *activation function*, dan *pooling layer*. Sedangkan di tahapan *classification*, terdapat *flatten*, *activation function*, dan *fully connected layer*. Tahapan ini memberikan output

berupa probabilitas terhadap kelas. Model CNN dapat dilihat pada Gambar 2. 1 yang bersumber dari Matlab (2021).



Gambar 2. 1 Model *Convolutional Neural Network* (CNN)

Convolution layer atau lapisan konvolusi adalah komponen utama dalam CNN, yang dimana filter berukuran kecil (biasanya 3x3 atau 5x5) digerakan melintasi data input untuk menghasilkan *feature map* (peta fitur). Setiap *filter* atau *kernel* dirancang untuk mengenali pola tertentu, seperti garis tepi atau tekstur (Li *et al.* 2020). Setelah lapisan konvolusi, umumnya diikuti dengan *activation function*, seperti ReLU. *Activation function* atau dikenal juga dengan non-linearitas digunakan untuk mengurangi linearitas yang dihasilkan dari *convolution layer*. *Pooling layer* dilakukan setelah melewati lapisan konvolusi beserta *activation function*. Lapisan *pooling* digunakan untuk mengurangi jumlah spasial (panjang dan lebar) pada peta fitur. *Pooling* yang biasa digunakan dalam CNN adalah *max-pooling* atau *Global Average Pooling* (GAP). Setelah melewati lapisan yang ada di *feature extraction*, dilanjutkan ke dalam tahapan *classification*. Namun, sebelum masuk ke dalam *classification*, peta fitur diubah menjadi sebuah vektor di dalam *flatten layer*. Di dalam tahapan *classification*, vektor saling terhubung menjadi *Fully Connected* (FC) *layer*. Lapisan ini, setiap neuron terhubung secara langsung dengan seluruh neuron di lapisan lainnya. Lapisan ini umumnya digunakan

dibagian akhir pada model CNN untuk menjalankan tugas klasifikasi atau prediksi (Qian *et al.* 2020).

Untuk mendapatkan prediksi atau nilai akurasi yang tinggi dibutuhkan arsitektur yang tepat. Arsitektur dalam metode CNN berperan penting dalam menentukan tingkat akurasi keberhasilan dalam mengenali objek. MiniVGGNet adalah salah satu arsitektur yang memiliki tingkat akurasi yang tinggi dalam proses identifikasi atau klasifikasi, arsitektur tersebut adalah versi modifikasi dari VGGNet yang lebih ringan namun tetap efektif. Beberapa penelitian telah menunjukkan performa dari arsitektur MiniVGGNet. Penelitian yang dilakukan oleh Latifah *et al*. (2024), dalam klasifikasi sel darah merah mendapatkan nilai akurasi sebesar 96,05%. Penelitian lain yang dilakukan oleh Winanda *et al.* (2021) mengklasifikasikan kualitas mutu dari daun gambir. Hasil dari penelitian tersebut mendapatkan nilai akurasi sebesar 99,87%. Dari beberapa penelitian tersebut, menunjukkan bahwa MiniVGGNet mampu menghasilkan akurasi yang tinggi dalam mengklasifikasikan dan mengidentifikasikan data citra.

2.3 Studi Terkait Klasifikasi

Penelitian yang dilakukan oleh Sheila *et al*. (2023) mendeteksi penyakit daun padi menggunakan *convolutional neural network* dengan menggunakan model arsitektur Inception V3. Dataset untuk penelitian tersebut berjumlah 240 data citra yang diambil dari situs Kaggle. Dari dataset tersebut kemudian dibagi menjadi 3 kelas yaitu, penyakit blas, penyakit hawar daun, dan penyakit tungro, yang setiap kelasnya berjumlah 80. Pembagian data dibagi sebesar 80% untuk data latihan dan 20% untuk data uji. Data tersebut memasuki proses *resize* dengan ukuran 299 x 299

piksel dan juga dilakukan augmentasi data. Hasil dari data latihan memiliki akurasi sebesar 98,44% dan untuk data uji sebesar 93.75%.

Penelitian lain yang dilakukan oleh Sutikno *et al.* (2024). mengklasifikasikan penyakit pada sawi pakcoy menggunakan CNN. Data citra diambil dari *smart greenhouse* di Universitas Nahdatul Ulama Indonesia (UNUSIA) sebanyak 1000 gambar. Dataset dibagi menjadi dua bagian, 500 gambar sawi pakcoy tanpa penyakit dan 500 gambar sawi pakcoy dengan penyakit. Tahap selanjutnya dilakukan *resize* dengan ukuran 512x512 piksel. Sebelum dilakukan proses klasifikasi data dibagi menjadi data latihan sebesar 80% dan data uji 20%. Dari 15 *epoch* didapatkan nilai akurasi yang hampir menyentuh angka 1 dengan tingkat *loss* mendekati nilai 0,1. Hasil uji coba didapatkan nilai akurasi sebesar 89,12% dan nilai loss 0,240.

Hawibowo & Muhimmmah (2024) melakukan penelitian untuk mendeteksi kematangan buah pepaya dengan metode CNN berbasis Android. Data citra yang didapatkan berjumlah 315 citra dari 3 kelas, yaitu matang, setengah matang, dan belum matang. Data diproses menjadi tiga bagian, yaitu proses pelatihan, validasi, dan pengujian dengan pembagian rasio 70:10:20. Pada proses pelatihan dengan 20 *epoch* hasil akurasi yang didapatkan sebesar 98,63%, lalu untuk proses validasi akurasi yang didapatkan sebesar 90%. Kemudian dilanjutkan dengan proses *testing* yang mendapatkan hasil akurasi sebesar 96,97% dan nilai *loss* 0,0932.

Widianto *et al.* (2023) melakukan penelitian dengan mengidentifikasi penyakit tanaman jagung berdasarkan citra daun menggunakan *convolutional neural network*. Data citra didapatkan dari *website* kaggle sebanyak 4188 data citra

yang terbagi menjadi 4 kelas yaitu *blight*, *common rust*, *grey leaf spot*, dan *healthy*. Data tersebut dibagi menjadi dua, data *training* sebesar 80% dan data uji sebesar 20%. Selanjutnya, dilakukan *preproccesing* data. Tahap pertama HSV (*Hue Saturation Value*), lalu *grayscale*, ROI (*Region of Interest*), terakhir GLCM (*Gray Level Co-Occurrence Matrix)*. Hasil dari penelitian tersebut mendapatkan Tingkat akurasi sebesar 93%.

Hairani & Widiyaningtyas (2024) melakukan deteksi penyakit pada tanaman padi metode CNN dan augmentasi data. Data citra yang diperoleh berasal dari *website* kaggle sebanyak 120 citra dengan pembagian 40 citra setiap kelas. Kelas terbagi menjadi 3 bagian yaitu *bacterial leaf blight*, *brown spot*, dan *leaf smut*. Kemudian data tersebut memasuki *image processing*. Proses tersebut mulai dari *resizing*, mengubah RGB ke *grayscale*, dan augmentasi data. Data dibagi menjadi 80 % data latih dan 20% data uji. Hasil yang didapatkan dari deteksi penyakit pada daun padi sebesar 99,7%.

Ilham Rahmana Syihad *et al.* (2023) mengidentifikasi penyakit pada tanaman daun pisang dengan metode CNN. Penelitian ini memanfaatkan model ResNet50 dan VGG-19. ResNet50 menunjukkan performa terbaik dengan akurasi sebesar 94%, *precision* 88%, *recall* 91%, dan skor F1 mencapai 89%. Di sisi lain, model VGG-19 juga memberikan hasil yang baik dengan akurasi 91%. Tingginya tingkat akurasi dari ResNet50 menjadikannya model unggulan dalam mendeteksi penyakit tanaman pisang berbasis CNN pada penelitian tersebut.

Kotta *et al.* (2022) penelitian tersebut melakukan identifikasi penyakit pada daun tomat menggunakan metode CNN. Metode *Convolutional Neural Network*

(CNN) telah berhasil diterapkan untuk mendeteksi penyakit pada tanaman tomat melalui citra daun. Aplikasi identifikasi penyakit ini dibangun dengan algoritma CNN dan mampu mengolah gambar yang diambil dari galeri maupun kamera secara langsung. Dalam pengujian normal menggunakan lima sampel untuk setiap kelas, aplikasi mencapai akurasi sebesar 94% dengan tingkat kesalahan 6%. Sistem ini mampu mengenali 10 kelas, terdiri dari 9 jenis penyakit seperti *bacterial spot*, *early blight*, *late blight*, *leaf mold*, *septoria leaf spot*, *tomato two-spotted spider mite*, *target spot*, *tomato mosaic virus*, dan *tomato yellow leaf curl virus*, serta satu kelas daun sehat.

Terakhir penelitian yang dilakukan oleh M.B. Gigih Baskoro Ashari (2024) mengidentifikasi penyakit pada tanaman padi menggunakan CNN. Dengan menerapkan teknik augmentasi data untuk memperluas dataset dan meminimalkan risiko *overfitting*, model CNN yang dikembangkan berhasil mencapai akurasi validasi sebesar 62%. Akurasi tertinggi tercatat pada kelas daun sehat, yaitu sebesar 83%. Hasil ini menunjukkan bahwa CNN memiliki potensi dalam mendeteksi penyakit pada tanaman durian. Namun, penelitian ini juga menggaris bawahi pentingnya optimasi model dan penambahan data pelatihan guna meningkatkan kinerja sistem secara keseluruhan.

2.4 Metode CNN Pada Aplikasi Android

Perkembangan teknologi *mobile* yang semakin maju, aplikasi Android tidak hanya digunakan untuk komunikasi dan hiburan, namun juga dapat dimanfaatkan sebagai sarana untuk mendukung pengolahan data berbasis kecerdasan buatan atau *artificial intelligence*. Aplikasi Android merupakan perangkat lunak yang terdapat

di dalam *smartphone* yang memungkinkan pengguna untuk menjalankan aktivitas secara praktis (Widia Rahma Minniarni *et al*., 2022).

Penelitian yang dilakukan oleh Alden & Sari (2023) mengklasifikasikan sampah sesuai jenisnya, di dalam penelitian tersebut sampah terbagi menjadi dua, organik dan anorganik. Arsitektur CNN yang digunakan adalah MobileNet yang diterapkan ke dalam android. Hasil pengujian sampah organik mendapatkan nilai akurasi sebesar 96% dan sampah anorganik sebesar 99%. Penelitian tersebut menunjukkan bahwa CNN yang diterapkan pada aplikasi *mobile* tetap memiliki akurasi yang tinggi dengan menggunakan sumber daya yang rendah. Penelitian lain yang dilakukan oleh Hawibowo & Muhimmmah (2024) mendeteksi tingat kematangan pepaya dengan metode CNN berbasis Android. Hasil dari uji coba mendapatkan tingkat akurasi yang tinggi sebesar 96,96% dan nilai *loss* sebesar 0,0932.

Dengan memanfaatkan aplikasi Android, metode CNN tetap efisien dalam mengidentifikasi atau untuk pengklasifikasian. Namun, karena diterapkan di Android, maka arsitektur yang dibutuhkan harus ringan dan tidak berat dalam komputasi, tapi tetap memiliki tingkat akurasi yang tinggi. Hal ini menjadi dasar pertimbangan dalam pemilihan arsitektur model yang tepat, agar keseimbangan antara ukuran *file*, kecepatan deteksi, dan akurasi dapat tercapai.

Beberapa penelitian yang telah dilakukan sebelumnya mendapatkan tingkat akurasi yang berbeda. Tingkat akurasi didapatkan bergantung pada jenis arsitektur dan data yang digunakan. Tabel 2. 1 menampilkan ringkasan beberapa penelitian terkait metode CNN.

Tabel 2. 1 Rangkuman Studi Pustaka

No.	Peneliti	Judul	Metode	Data	Hasil
1.	Sheila *et al.* (2023)	Deteksi Penyakit Pada Daun Padi Berbasis Pengolahan Citra Menggunakan Metode Convolutional Neural Network (CNN).	InceptionV3	Tiga dataset berupa blas, hawar daun, tungro. Jumlah data sebanyak 240 data citra.	Akurasi latihan sebesar 98,44% dan untuk data uji sebesar 93.75%.
2.	Hairani & Widiyaningtyas (2024)	*Augmented Rice Plant Disease Detection with Convolutional Neural Networks*	Arsitektur kostum	Kelas terbagi menjadi 3 yaitu *blight*, *brown spot*, dan *leaf smut*. Total 120 citra.	Hasil yang didapatkan dari deteksi penyakit pada daun padi sebesar 99,7%.
3.	Ilham Rahmana Syihad *et al.* (2023)	*CNN Method to Identify the Banana Plant Diseases based on Banana Leaf Images by Giving Models of ResNet50 and VGG-19*	ResNet50 dan VGG-19	Empat kelas dengan data *healthy, Cordana, Sigatoka,* dan *Pestalotiopsis*. Total 936 data citra.	ResNet50 akurasi sebesar 94%. VGG-19 akurasi sebesar 91%.
4.	Widianto *et al*. (2023)	Identifikasi Penyakit Tanaman Jagung Berdasarkan Citra Daun Menggunakan Convolutional Neural Network	Arsitektur kostum	Empat kelas dengan data *Blight*, *Common Rust*, *Grey Leaf Spot*, dan *Healthy*. Total 4188 data citra.	Hasil rata-rata yang didapatkan dari *Precision*, *Recall*, dan *F1-Score* sebesar 93%.
5.	Winanda *et al*. (2021)	Klasifikasi Kualitas Mutu Daun Gambir Ladang Rakyat Menggunakan Metode Convolutional Neural Network	Arsitektur MiniVGGNet	Kelas terbagi menjadi dua, kualitas bagus dan jelek, Total 6000 data citra.	Hasil akurasi yang didapatkan sebesar 99,87%
6.	Latifah *et al.* (2024)	Peningkatan Performa Klasifikasi Sel Darah Merah pada Pasien Talasemia Minor	*Support Vector Machine* (SVM) dan MiniVGGNet	Sembilan kelas berdasarkan jenis sel. Total 7.108 data citra.	Delapan kelas dengan nilai akurasi 96,05%, presisi 96,00%, sensitivitas 96,05%, dan F1-score 95,95%.

BAB III

DESAIN DAN IMPLEMENTASI SISTEM

3.1 Pengumpulan Data

Penelitian ini berfokus pada penyakit tanaman padi. Data yang dikumpulkan berasal dari *website Kaggle* berupa citra daun padi yang kemudian diubah menjadi dataset. Dataset terdiri dari 4 jenis penyakit dan daun padi sehat (*healthy*). Jenis penyakit tersebut terdiri dari hawar daun bakteri (*blight*), blas (*blast*), tungro, dan bercak coklat (*brown spot*). Total keseluruhan citra berjumlah 1.987 citra yang terdiri dari 883 data citra *no-background* dan 1104 data citra *background*. Tabel 3. 1 Dataset merupakan citra daun padi.

Tabel 3. 1 Dataset Padi

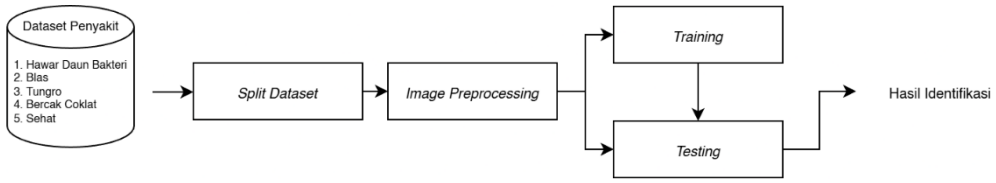
No	Nama	Citra Daun
1	Penyakit hawar daun bakteri (*blight*)	
2	Penyakit blas (*blast*)	
3	Penyakit tungro	
4	Penyakit bercak coklat (*brown spot*)	
5	Daun padi sehat (*healthy*)	

Citra daun pada Tabel 3. 1 merupakan daun padi yang terkena dari berbagai jenis penyakit yang berasal dari bakteri, jamur atau virus. Penyakit hawar daun bakteri atau *bacterial leaf blight* disebabkan oleh bakteri *xanthomonas oryzae pv.*

oryzae yang mengakibatkan daun berubah menjadi warna kuning, dan menyebabkan daun layu lalu mati. Penyakit blas atau *leaf blast* berasal dari jamur *pyricularia oryzae* yang menimbulkan gejala bercak berwarna putih keabu-abuan dengan berbentuk belah ketupat yang memiliki tepi warna coklat. Penyakit tungro disebabkan oleh Kombinasi virus *Rice Tungro Bacilliform Virus* (RTBV) dan *Rice Tungro Spherical Virus* (RTSV) atau salah satu dari keduanya. Akibatnya tanaman padi menjadi kerdil dan perubahan warna daun menjadi kuning dari pangkal hingga bawah padi. Bercak coklat atau *brown spot*, penyakit ini disebabkan oleh jamur *cochliobolus miyabeanus*. Akibat dari jamur tersebut adalah bercak coklat pada daun yang berbentuk bundar atau oval.

3.2 Desain Sistem

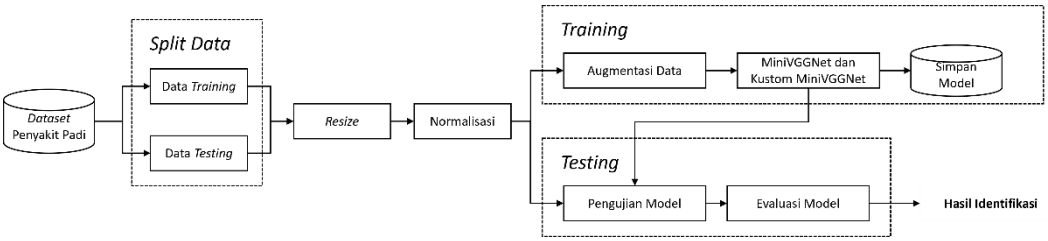
Desain sistem adalah gambaran sebuah sistem identifikasi atau klasifikasi dari awal proses hingga menghasilkan output. Proses ini dilakukan secara bertahap dari memasukkan data citra yang telah dikumpulkan hingga menghasilkan nilai probabilitas. Desain sistem pada penelitian ini dapat dilihat pada Gambar 3. 1.



Gambar 3. 1 Desain Sistem

Pada Gambar 3. 1 setelah data dikumpulkan menjadi dataset yang terdiri dari 5 kelas, yaitu hawar daun bakteri, blas, tungro, bercak coklat, dan padi normal atau sehat. Proses yang pertama dilakukan adalah membagi data menjadi dua bagian, data *training* dan data *testing*, kemudian dilakukan *input image* atau memasukkan

data *image* ke dalam *image preprocessing*. Di dalam proses tersebut, data citra melewati beberapa tahapan agar semua data seragam, tidak memberatkan komputasi, dan mudah diolah pada model CNN. Tahapan pertama, perubahan ukuran dari data citra agar semua data memiliki ukuran yang sama. Kedua, data dinormalisasikan ke rentang 0 hingga 1. Terakhir, masuk ke augmentasi data untuk menciptakan variasi *image*, tahapan tersebut adalah *rotate*, *horizontal flip*, dan *vertical flip*. Gambar 3. 2 menampilkan skema proses *training* dan *testing*.



Gambar 3. 2 Skema *Training* dan *Testing*

Pada Gambar 3. 2 menunjukkan bahwa pada penelitian ini terbagi menjadi dua proses yaitu *training* dan *testing*. Proses *training* memiliki tahapan yang lebih banyak dalam *image preprocessing* dibandingkan dengan proses *testing*. Perbedaan terdapat dalam augmentasi data berupa *rotate*, *horizontal flip*, dan *vertical flip*. Perbedaan tersebut terjadi karena dibutuhkan variasi data dalam proses *training* agar tidak terjadi *overfitting*. Selama proses pelatihan berlangsung nilai bobot dan bias juga selalu diperbarui agar model memiliki tingkat akurasi yang tinggi dan nilai *loss* yang rendah. Hasil *training* yang dilakukan kemudian diuji coba ke dalam proses *testing*. Di dalam proses *testing*, hanya terdapat *resize image* dan normalisasi. Tujuannya agar data memiliki ukuran yang sama dan tidak memberatkan komputasi.

3.2.1 *Input Image*

Data citra yang telah dikumpulkan kemudian dijadikan dataset pada penelitian ini. Semua data citra memiliki format .JPG, namun ukuran dan resolusi tiap data berbeda. Data citra tersebut digunakan sebagai *input* dalam sistem yang telah dibuat. Jumlah data citra yang digunakan sebagai *input* sebanyak 883 data citra *no-background* dan 1104 data citra *background* .

3.2.2 *Image Preprocessing*

Proses ini dilakukan untuk membuat data yang telah dimasukkan (*input*) ke dalam sistem menjadi seragam dan untuk menambahkan variasi data dalam proses *training* agar tidak terjadi *overfitting*. Proses ini memiliki beberapa tahapan yaitu *resize image*, normalisasi, *rotate*, *horizontal flip*, dan *vertical flip*.

1. *Resize Image*

Tahapan pertama yang dilakukan pada *image preprocessing* adalah melakukan *resize image*. Tahapan ini bertujuan untuk mengubah ukuran spasial gambar (lebar dan tinggi) agar tiap data citra memiliki ukuran yang sama. Ukuran tiap gambar akan diubah menjadi 32x32 piksel.

2. Normalisasi

Normalisasi adalah tahapan kedua yang dilakukan pada *preprocessing image*. Normalisasi berperan untuk meringankan beban komputasi. Berikut adalah rumus normalisasi yang ditunjukkan pada persamaan 3.1.

$$N_c = \frac{X_c}{225} \tag{3.1}$$

Keterangan:
N : Hasil Normalisasi
c : Channel pada gambar
X : Piksel yang akan dinormalisasikan
225 : Nilai maksimum piksel

3. *Rotate*

Tahapan ini merotasikan gambar sebesar 90 derajat. Proses ini merupakan bagian augmentasi data di dalam *preprocessing image*. Dengan begitu, gambar yang dihasilkan akan beragam atau bervariasi dan juga untuk menghindari *overfitting* pada model CNN.

4. *Horizontal Flip*

Horizontal flip termasuk ke dalam augmentasi data. *Image* yang masuk tahapan ini berfungsi untuk menambah variasi data agar tidak terjadi *overfitting* pada model CNN yang digunakan. Tahapan ini akan membalik gambar secara horizontal.

5. *Vertical Flip*

Tahapan terakhir adalah *vertical flip*. Tahapan ini juga masuk ke dalam augmentasi data yang berfungsi untuk menambahkan varisasi data pada pelatihan model. Tahapan ini akan membalik gambar secara vertikal, sehingga gambar akan kelihatan terbalik dari aslinya.

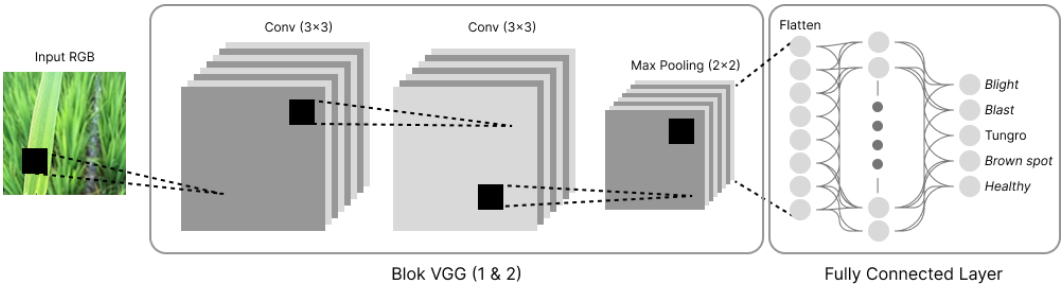
3.2.3 Model MiniVGGNet

Model MiniVGGNet adalah model yang dipopulerkan oleh Rosebrock (2021). MiniVGGNet adalah versi model yang lebih sederhana dan ringan dibandingkan dengan VGGNet. Model VGGNet merupakan model *Convolutional Neural Network* (CNN) yang dapat mengenali gambar dan video (Simonyan &

Zisserman, 2015). Terdapat aspek penting dari model VGGNet adalah kedalaman dari *convolutional layer* model. Model VGGNet memiliki jenis yang berbeda berdasarkan kedalaman *layer*-nya, mulai dari 11 hingga 19 *weight layer*. Selain itu, *filter* berukuran 3x3 untuk semua jenis model dengan *stride* 1. Sedangkan pada model MiniVGGNet, hanya memiliki 7 *weight layer* dan ukuran *filter* 3x3 dengan *stride* 1.

3.3 Implementasi Sistem

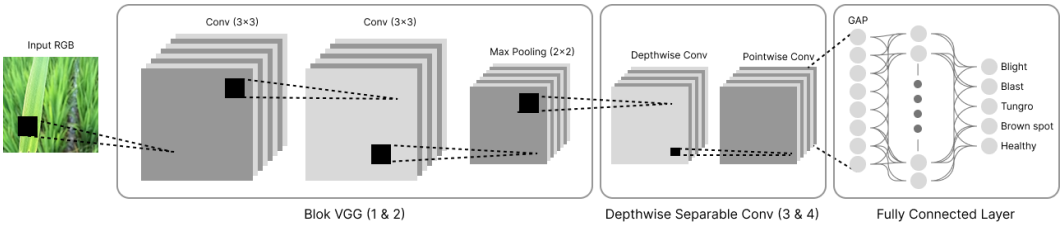
Sistem ini dibangun dengan menggunakan metode Convolutional Neural Network (CNN) dengan arsitektur MiniVGGNet. Setelah model dibuat dan melewati proses pelatihan (*training*) dan pengujian (*testing*), model tersebut diimplementasikan ke dalam aplikasi mobile, khususnya pada sistem operasi yang berbasis Android.



Gambar 3. 3 Arsitektur MiniVGGNet

Pada Gambar 3. 3, arsitektur MiniVGGNet tersusun dari 2 blok VGG dan lapisan *Fully Connected* (FC). Tiap blok dari arsitektur tersebut berisikan dua *convolution layer* dan satu *max pooling*. Pada blok pertama data mengelola data input. Di dalam blok tersebut terdapat *convolution layer* dengan ukuran *filter* atau *kernel* 3x3. Ukuran *channel* pada tiap blok berbeda-beda, 32 untuk blok pertama

dan 64 untuk blok kedua. Data input yang melewati *convolution layer* menghasilkan *feature maps*, kemudian masuk ke dalam *max pooling*. *Max pooling* akan mengambil nilai neuron yang tertinggi. Proses tersebut diulangi hingga blok terakhir. Sebelum masuk ke dalam lapisan FC, *feature maps* yang dihasilkan dari blok ketiga memasuki lapisan *flatten*. Ini bertujuan untuk mengubah *feature maps* menjadi bentuk vektor. Setelah melewati *flatten* dan lapisan FC, model akan memberikan output berupa probabilitas. Namun, pada penelitian ini terdapat tambahan dan pengurangan pada arsitektur MiniVGGNet. Penyesuaian ini dilakukan untuk mengurangi ukuran parameter model CNN agar lebih efisien ketika diterapkan pada perangkat Android tanpa membebani *device* dengan ukuran model yang besar. Gambar 3. 4 menunjukkan kustomisasi arsitektur MiniVGGNet dengan penambahan *depthwise separabel convolution* yang berisikan *depthwise convolution* dan *pointwise convolution*.



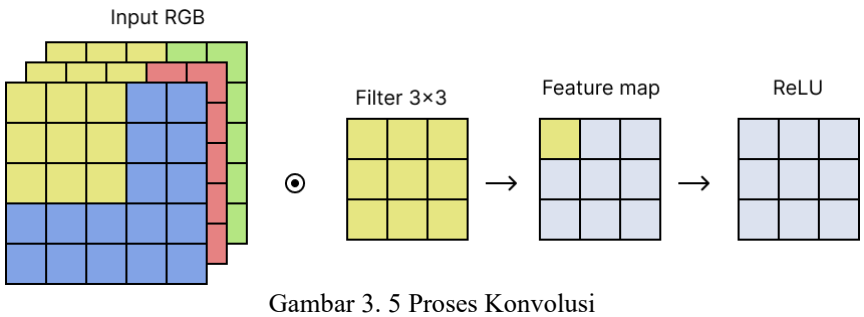
Gambar 3. 4 Kustomisasi Arsitektur MiniVGGNet

3.3.1 *Input Layer*

Lapisan awal pada arsitektur ini adalah input gambar yang berasal dari *dataset*. Sebelum masuk ke dalam lapisan input, data citra terlebih dahulu di proses di *image preprocessing*. Data citra yang masuk pada lapisan input berukuran 32x32 dan memiliki tiga channel RGB, yang memiliki nilai piksel yang telah dinormalisasikan dengan ukuran 0 hingga 1.

3.3.2 *Convolutional Layer*

Lapisan konvolusi terletak pada tiap blok VGG. Lapisan ini berfungsi untuk mengekstraksi fitur pada data citra dari lapisan input. Fitur yang diekstraksi dari data input bisa berupa tepi, sudut, garis, atau yang berhubungan dengan data citra. Untuk mendapatkan fitur-fitur tersebut dibutuhkan *filter* atau *kernel*. *Kernel* berbentuk matriks *N* x *N* tergantung dari arsitektur model CNN yang digunakan. *Kernel* akan melakukan perkalian pada permukaan data input di setiap *channel* yang bergerak secara horizontal dan vertikal. Pergeseran *kernel* ini ditentukan oleh parameter *stride*, sedangkan untuk menjaga agar dimensi output tetap sama dengan input, diterapkan teknik *padding*. Sebelum masuk ke fungsi aktivasi, hasil penjumlahan perkalian tersebut ditambahkan dengan nilai bias. Hasil dari perkalian tersebut kemudian dijumlahkan, lalu melewati fungsi aktivasi non linearitas ReLU. Hasil dari fungsi aktivasi menghasilkan sebuah *feature map*. Jumlah *feature maps* yang dihasilkan sama banyak dengan jumlah filter yang digunakan. Semakin banyak fitur yang digunakan, maka semakin banyak juga fitur yang dapat diekstrak atau didapatkan. Gambar 3. 5 menunjukkan visualisasi dari proses konvolusi.



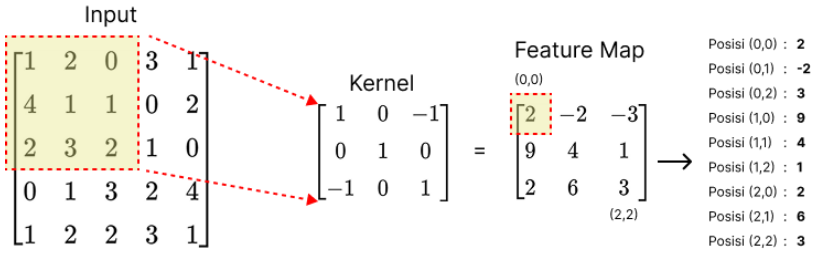
Gambar 3. 5 Proses Konvolusi

Operasi perhitungan dari proses konvolusi dapat dihitung dengan menggunakan persamaan 3.2.

$$\hat{z}_{(p,q)} = \sum_{i=0}^{k-1}\sum_{j=0}^{k-1} x_{(p+i,q+j)} w_{(i,j)} \quad (3.2)$$

Keterangan:
$\hat{z}_{(k,l)}$: Output feature map sementara
$x_{(k+i,l+j)}$: Input citra pada posisi (k,l)
p, q : Indeks spasial untuk input citra
i, j : Indeks spasial untuk filter

Berikut adalah contoh perhitungan sederhana dari operasi proses konvolusi. Gambar 3. 6 menunjukkan operasi proses konvolusi.



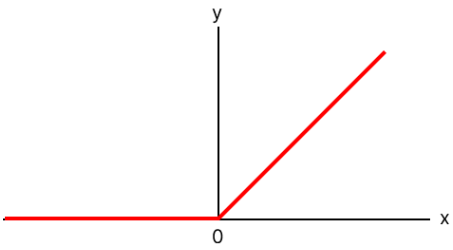
Gambar 3. 6 Operasi Proses Konvolusi

Setelah melakukan operasi konvolusi, maka dilanjutkan ke dalam perhitungan fungsi aktivasi. Pada penelitian ini, fungsi aktivasi yang digunakan adalah ReLU. Fungsi aktivasi ReLU adalah fungsi yang mengubah nilai minimum menjadi 0 dan maksimum menjadi x. Maka, jika terdapat nilai negatif diubah menjadi 0. Persamaan 3.3 adalah rumus aktivasi fungsi ReLU dan Gambar 3. 7 adalah grafik fungsi aktivasi ReLU.

$$ReLU(x) = \max(0, x) \quad (3.3)$$

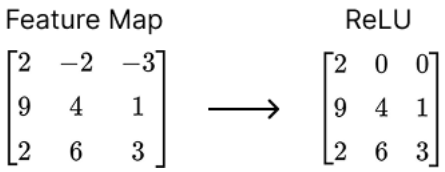
$$z = \mathrm{ReLU}(\hat{z}) \quad (3.4)$$

Keterangan:
$ReLU$: Fungsi aktivasi ReLU
x : Nilai input
z : Output *feature map*



Gambar 3. 7 Fungsi Aktivasi ReLU

Berikut adalah contoh penerapan fungsi aktivasi ReLU terhadap *feature maps* yang dapat dilihat pada Gambar 3. 8.



$$\begin{bmatrix} 2 & -2 & -3 \\ 9 & 4 & 1 \\ 2 & 6 & 3 \end{bmatrix} \longrightarrow \begin{bmatrix} 2 & 0 & 0 \\ 9 & 4 & 1 \\ 2 & 6 & 3 \end{bmatrix}$$

Gambar 3. 8 Contoh Penerapan ReLU

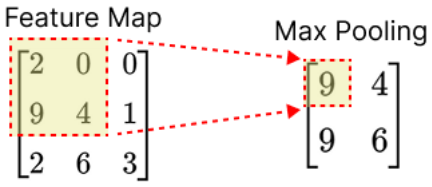
3.3.3 *Max Pooling Layer*

Penelitian ini menggunakan *max pooling layer*. Lapisan ini terdapat pada setiap blok VGG yang terletak setelah tahapan lapisan konvolusi. Lapisan *max pooling* berfungsi untuk mengurangi ukuran spasial dan mengambil nilai tertinggi dari *feature maps*. Ukuran *kernel max pooling* pada MiniVGGNet adalah 2x2 dengan *stride* 2. Perhitungan *max pooling* dapat dilihat pada persamaan 3.5.

$$m_{(i,j)} = \max(x_{2i:2i+2,2j:2j+2}) \tag{3.5}$$

Keterangan:
$m_{(i,j)}$: Output *max pooling*.
x : Nilai input.
i, j : Indeks spasial input.

Gambar 3. 9 adalah contoh penerapan *max pooling layer* terhadap *feature map* yang telah dilalui fungsi aktivasi atau non-linearitas ReLU. Pada Gambar 3. 9, *max pooling layer* menggunakan *kernel* 2x2 dan *stride* 1.



Gambar 3. 9 Contoh Penerapan *Max Pooling Layer*

3.3.4 *Depthwise Separable Convolution*

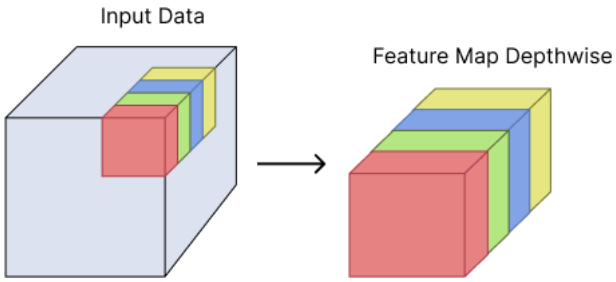
Depthwise separable convoltuion adalah lapisan yang tersusun dari *depthwise convolution* dan *pointwise convolution*. Lapisan konvolusi ini menghasilkan lapisan *filtering* (*depthwise*) dan lapisan kombinasi (*pointwise*). Gabungan kedua lapisan tersebut sangat efektif dalam menurunkan beban komputasi dan mengecilkan ukuran model.

1. *Depthwise Convolution*

Pada lapisan ini, dilakukan penerapan satu filter untuk satu *channel* input. Jika terdapat *channel* sejumlah *M* pada input, maka filter pada *depthwise convolution* juga akan berjumlah *M*. Hal ini menunjukkan bahwa setiap channel warna atau fitur diproses secara satu per satu. Berbeda dengan konvolusi konvensional yang satu *filter* diterapkan untuk semua *channel feature maps*. *Filter* pada lapisan *depthwise* berukuran 3x3. Output dari lapisan konvolusi ini mengurangi nilai spasial *feature maps* sebelumnya dan mempertahankan jumlah *channel*-nya. Proses pemisahan ini secara signifikan dapat mengurangi beban komputasi dan mengurangi jumlah parameter yang harus dilatih dibandingkan dengan konvolusi konvensional. Output dari lapisan ini diikuti dengan fungsi aktivasi ReLU. Operasi perhitungan dan proses *depthwise convolution* dapat dilihat pada persamaan 3.6 dan Gambar 3. 10.

$$\hat{G}_{k,l,m} = ReLU(\sum_{i,j} \hat{K}_{i,j,m} \cdot F_{k+i-1,l+j-1,m}) \quad (3.6)$$

Keterangan:
$\hat{G}_{k,l,m}$: Output posisi (k,l) untuk *channel* m setelah *depthwise convolution*.
$K_{i,j,m}$: *Kernel depthwise* pada posisi (i,j) untuk *channel* m.
$F_{k+i-1,l+j-1,m}$: Input posisi (k+i-1, l+j-1) untuk *channel* m.
i, j : Indeks spasial untuk kernel.
k, l : Indeks spasial (posisi piksel) pada *output feature map*.



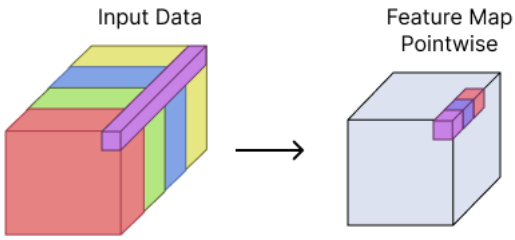
Gambar 3. 10 Proses *Depthwise Convolution*

2. *Pointwise Convolution*

Lapisan *pointwise convolution* ini dilakukan setelah *depthwise convolution*. Proses konvolusi ini menggunakan ukuran *filter* 1x1 yang berfungsi untuk memperbanyak atau mengurangi *channel* pada outputnya. Pada kustomisasi arsitektur MiniVGGNet, lapisan *pointwise convolution* digunakan untuk memperbanyak *channel feature maps* dan mempertahankan nilai spasial *feature maps*, Maka dari itu, model dapat meningkatkan kompleksitas fitur. Operasi perhitungan *pointwise convolution* dapat dilihat pada persamaan 3.7.

$$G_{k,l,n} = ReLU(\sum_{m} \tilde{K}_{m,n} \cdot \hat{G}_{k,l,m}) \quad (3.7)$$

Keterangan:
$G_{k,l,n}$: Output posisi (k,l) untuk *channel* n.
$\tilde{K}_{m,n}$: *Kernel pointwise* berukuran 1x1.
$\hat{G}_{k,l,m}$: Output dari *depthwise convolution*.
n : Indeks spasial (posisi piksel) pada *output feature map*.



Gambar 3. 11 Proses *Pointwise Convolution*

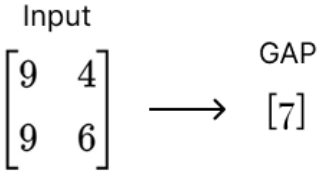
3.3.5 *Global Average Pooling Layer*

Lapisan *Global Average Pooling* (GAP) digunakan untuk menggantikan proses *flatten* pada model kustomisasi MiniVGGNet, namun pada model MiniVGGNet tetap menggunakan *flatten*. Lapisan GAP digunakan karena mampu mereduksi ukuran model dibandingkan *flatten*. GAP bekerja dengan mengubah lapisan input yang berukuran *H x W x C* (tinggi x lebar x *channel*) menjadi ukuran 1 x 1 x *C*. Setiap *channel* pada lapisan input diwakili dengan angka tunggal setelah proses GAP. Operasi perhitungan GAP dapat dilihat pada persamaan 3.8.

$$x_c = \frac{1}{H \times W} \sum_{i=0}^{H} \sum_{j=0}^{W} G_c(i,j) \tag{3.8}$$

Keterangan:
x_c : Output dari proses *squeeze*.
$H\ x\ W$: Tinggi dan lebar *feature map* input U.
$G_c(i,j)$: Nilai piksel pada posisi (i,j) untuk *channel* c pada fitur input U.

Berikut adalah contoh sederhana dari proses GAP pada Gambar 3. 12.



Gambar 3. 12 Proses GAP

Lapisan ini efektif untuk mengurangi ukuran model yang besar, tanpa mengorbankan nilai akurasi yang signifikan. Parameter adalah nilai yang dipelajari selama proses *training*, jumlah parameter pada setiap model juga menentukan jumlah ukuran *file* pada model. Persamaan berikut adalah operasi perhitungan parameter.

$$n_{in} = H \times W \times C \tag{3.9}$$

$$Param = (n_{in} \times n_{out}) + b_{out} \tag{3.10}$$

Keterangan:
n_{in} : Jumlah neuron pada lapisan input.
n_{out} : Jumlah neuron pada lapisan output.
b_{out} : Jumlah bias pada lapisan output.
$Param$: Output parameter.

Berikut adalah perbedaan parameter yang dihasilkan pada kedua model MiniVGGNet dan kustom MiniVGGNet yang ditampilkan pada Tabel 3. 2.

Tabel 3. 2 Perbedaan Jumlah Parameter antara Flatten dan GAP

Jenis	Model	Nin	Nout	Parameter
Flatten	MiniVGGNet	4096	512	2.097.664
GAP	Kustom	256	512	131.584

Berdasarkan Tabel 3. 2, penerapan lapisan *flatten* memiliki jumlah parameter yang lebih banyak dibandingkan dengan lapisan GAP. Perbedaan yang didapatkan antara kedua lapisan tersebut sebesar 93,73%. Ini yang membuat model kustom memiliki ukuran *file* yang lebih kecil dibandingkan dengan model MiniVGGNet.

3.3.6 *Fully Connected Layer*

Fully Connected (FC) *layer* merupakan tahapan terakhir yang berada di metode CNN. Pada lapisan FC, setiap neuron dari lapisan sebelumnya terhubung secara menyeluruh ke *neuron* lapisan setelahnya. Setiap *link* yang menghubungkan

neuron memiliki bobot (*weight*). Nilai bobot mulanya dimulai dengan angka yang acak, yang kemudian diperbarui seiring berjalannya proses *training*. Untuk mendapatkan nilai *neuron* pada lapisan setelahnya, dilakukan proses perkalian antara nilai *neuron* sebelumnya dengan nilai bobot. Hasil dari perkalian tersebut, kemudian dijumlahkan, lalu ditambahkan dengan nilai bias, dan juga diikuti dengan fungsi aktivasi. Lapisan ini menggunakan ReLU sebagai fungsi aktivasi. Lapisan FC memiliki tiga lapisan yaitu input *layer*, *hidden layer*, dan output *layer*. Lapisan tersebut memiliki perannya masing-masing dalam *fully connected layer*.

1. *Input Layer*

Pada umunya, *input layer* di *fully connected layer* didapatkan dari proses *flatten*. Namun, pada arsitektur MiniVGGNet proses *flatten* diganti dengan lapisan *Global Average Pooling* (GAP). Sehingga, jumlah *neuron* yang ada di lapisan FC sama dengan *channel* pada lapisan sebelumnya.

2. Hidden Layer

Lapisan ini berada di antara *input layer* dan *output layer*. Jumlah *Hidden layer* tergantung dari arsitektur yang digunakan. Dalam arsitektur MiniVGGNet dan kustomisasi MiniVGGNet, hanya terdapat satu *hidden layer*. Lapisan ini memiliki 512 *neuron*.

3. Output Layer

Lapisan terakhir pada FC layer adalah *output layer*. Lapisan ini memiliki jumlah *neuron* sebanyak 5. Banyak neuron sama dengan banyak kelas yang diprediksi. Lapisan ini memberikan hasil prediksi akhir berupa probabilitas dengan

menggunakan fungsi aktivasi *softmax*. Nilai probabilitas yang dihasilkan bernilai 0 hingga 1. Persamaan 3.11 adalah operasi perhintungan fungsi aktivasi *softmax*.

$$p_c = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} \tag{3.11}$$

Keterangan:
p_c : Output dari fungsi softmax.
z_i : Nilai yang didapatkan dari FC Layer.
e : Nilai eksponensial e (2.718).
k : Jumlah kelas (5).

Fully connected layer adalah lapisan yang bertujuan untuk mengklasifikasi atau mengidentifikasi sebuah kelas dari data input yang diterima. Lapisan ini dikenal juga dengan proses *classification*. Sebelum masuk ke dalam proses *classification* adalah *feature extraction*. Proses yang dilakukan secara maju (*forward*), maka disebut dengan *forward propagation*. Jika sebaliknya, disebut *backward propagation*. Kedua proses tersebut masuk ke dalam proses *training*.

3.4 Proses *Training*

Proses ini dilakukan agar sistem mampu dalam mengklasifikasi atau mengidentifikasi penyakit pada tanaman padi, seperti penyakit hawar daun bakteri, blas, tungro, dan *brown spot*. Selain penyakit juga dapat mengidentifikasi padi daun sehat (*healthy*). Proses pelatihan terus berlangsung hingga nilai *error* yang diperoleh menjadi seminimal mungkin atau hingga jumlah *epoch* telah tercapai. Proses ini menggunakan metode CNN yang disusun dengan arsitektur MiniVGGNet dan kustom yang dilatih dengan data *training*. Proses yang dilakukan terbagi menjadi dua tahapan yaitu *forward propagation* dan *backward propagation*. Kedua tahapan tersebut dilakukan secara berulang hingga *epoch* yang ditentukan.

Jumlah *epoch* pada proses *training* memengaruhi tingkat akurasi model CNN. Selain itu, tingkat akurasi juga dipengaruhi oleh *hyperparameter*. *Hyperparamer* tidak dapat berubah atau nilainya konstan sepanjang proses *training* dilakukan, berbeda dengan nilai bobot yang diperbarui selama proses *training* berlangsung. *Hyperparameter* yang digunakan berupa *batch size, epoch, learning rate, optimizer,* dan *dropout*. Tabel 3. 3 penjelasan dari parameter yang digunakan.

Tabel 3. 3 *Hyperparameter*

Parameter	Deskripsi
Batch Size	Banyaknya jumlah *batch size* memengaruhi banyaknya *batch* yang akan dilatih pada proses *training*
Epoch	Ketika semua *batch* telah melalui proses *training*, maka terhitung satu *epoch*.
Learning Rate	Seberapa besar jarak atau langkah yang diambil dari grafik *gradient desecent* untuk memperbarui bobot.
Dropout	Teknik *regularization* untuk mengurangi *overfitting* dalam model CNN.
Optimizer	Teknik untuk meningkatkan akurasi dan mengurangi nilai *loss* pada model CNN.

Proses *training* bergantung pada *hyperparameter* yang digunakan. Proses *training* menghasilkan sebuah model CNN yang telah dilatih dengan bobot terbaru, yang digunakan untuk melakukan proses *testing*. Namun, jika hasil akurasi yang didapatkan masih rendah, maka perlu dilakukan *fine-tuning* atau penyesuaian pada beberapa aspek, salah satunya *hyperparameter*.

3.4.1 *Forward Propagation*

Proses pelatihan yang dilakukan secara maju, atau disebut juga *forward propagation* (propagasi maju). Proses ini dilakukan dari data citra yang dimasukkan ke dalam model CNN hingga menghasilkan sebuah prediksi. Proses propagasi maju bergerak melewati blok VGG pertama sampai kedua, lalu melewati proses

depthwise separable convolution, GAP, dan terakhir FC *layer*. Berikut penjelasan terkait proses propagasi maju:

1. Ekstraksi Fitur

Pada arsitektur MiniVGGNet, ekstraksi fitur melalui beberapa lapisan yang ada. Lapisan tersebut berupa dua blok VGG dan dua *depthwise separable convolution*. Di dalam blok VGG terdapat lapisan konvolusi dan lapisan *max pooling*. Data input akan diproses melalui blok tersebut dan dilanjutkan ke dalam lapisan *depthwise separable convolution*. Secara umum, operasi perhitungan untuk mendapatkan *feature map* dapat dilihat pada persamaan berikut.

$$\widehat{Fm} = Maxpool(ReLU(Conv(x, k))) \tag{3.12}$$

$$Fm = Pointw(Depthw(\widehat{Fm}, \hat{k}), \tilde{k}) \tag{3.13}$$

Keterangan:
Fm : Output *Feature Map*.
x : Input Data Citra.
k : Kernel.

2. *Flatten* dengan GAP

Sebelum masuk ke dalam *fully connected layer*, hasil ekstraksi fitur berupa *feature maps* perlu diubah menjadi sebuah vektor. *Feature maps* diubah menjadi sebuah vektor dengan menggunakan *Global Average Pooling* (GAP) *layer*. Lapisan ini mengubah setiap *channel* menjadi nilai tunggal. Persamaan 3.14 adalah proses fungsi GAP.

$$Flatten = GAP(Fm) \tag{3.14}$$

Keterangan:
$Flatten$: Output berupa vektor.
GAP : Fungsi *Global Average Pooling*.
Fm : Input *Feature map.*

3. Proses *input layer* menuju *hidden layer*

Hasil proses GAP dijadikan sebagai *input layer* pada *fully connected layer*. Setiap neuron yang berada di lapisan *input* terhubung dengan seluruh neuron pada lapisan *hidden*. *Link* yang menghubungkan kedua lapisan tersebut memiliki nilai bobot (*weight*). Nilai pada lapisan *hidden* didapatkan dengan cara mengalikan nilai neuron dengan nilai bobot dan ditambahkan nilai bias. Persamaan 3.15 menunjukkan operasi perhitungan untuk mendapatkan nilai *neuron hidden layer*.

$$\hat{z} = \sum_{i=1}^{n} x_{in_{(i)}} w_{in_{(i)}} + b \tag{3.15}$$

Keterangan:
$\hat{z}$: Output *neuron hidden layer*.
x_{in} : Input dari *neuron input layer*.
w_{in} : Nilai bobot.
i : indeks untuk *input* dan dan bobot.
n : Jumlah *neuron input layer.*
b : Nilai bias

Setelah mendapatkan hasil output *neuron* untuk lapisan *hidden*, dilanjutkan dengan menggunakan fungsi aktivasi ReLU. Persamaan 3.16 menunjukkan operasi perhitungan untuk melakukan fungsi aktivasi ReLU.

$$z = \ max(0, \hat{z}) \tag{3.16}$$

Keterangan:
z : Output neuron *hidden layer*.
$max()$: Fungsi aktivasi ReLU.

4. Proses *hidden layer* menuju *output layer*

Jumlah neuron yang berada di *hidden layer* adalah 512 *neuron*. Setiap *neuron* tersebut dikalikan dengan nilai bobot dan ditambahkan dengan nilai bias untuk mendapatkan nilai *neuron* pada *output layer*. Kombinasi linear ini bertujuan untuk memetakan fitur yang diekstraksi dari *hidden layer* ke dalam skor mentah (*logits*)

untuk setiap kelas. Perhitungan untuk mendapatkan nilai *neuron* pada *output layer* sama seperti persamaan 3.15. *Neuron* yang berada di *output layer* sebanyak 5 *neuron*. Jumlahnya disesuaikan dengan kelas yang diprediksi. *Neuron* pada *output layer* kemudian dilakukan perhitungan dengan fungsi aktivasi *softmax*. Fungsi aktivasi *softmax* menghasilkan nilai probabilitas antara 0 hingga 1 pada tiap neuronnya, dan jika semua *neuron* dijumlahkan menghasilkan nilai 1. Fungsi aktivasi ini juga cocok untuk *multi-class*, seperti yang dilakukan pada penelitian ini. Berikut adalah rumus fungsi aktivasi *softmax* yang dapat dilihat pada persamaan 3.17.

$$y = softmax(z) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} \tag{3.17}$$

Keterangan:
y : Output dari fungsi aktivasi *softmax*.
z : Input dari *hidden layer*.
e : Nilai eksponensial (2,718).
k : Jumlah kelas (5).

5. Menghitung nilai *loss*

Proses terakhir adalah menghitung nilai *loss*. Proses ini berfungsi untuk mengetahui nilai *loss* atau *error* dari sebuah data yang diprediksi dari model CNN. Semakin sedikit nilai *loss*-nya semakin akurat model yang digunakan, jika sebaliknya, maka model belum akurat, dan harus terus dilatih. Nilai *loss* ini selanjutnya menjadi dasar bagi algoritma optimasi untuk memperbarui bobot jaringan melalui proses *back propagation*. Tujuannya adalah untuk meminimalkan kesalahan tersebut secara iteratif hingga model mencapai performa optimal Untuk menghitung nilai *loss* digunakan fungsi *cross-entropy loss*. Rumus operasi fungsi *cross-entropy loss* dapat dilihat pada persamaan 3.18.

$$L = -\sum_{c=1}^{c} \hat{y}_c \, log(y_c) \tag{3.18}$$

Keterangan:
L : Output berupa nilai *loss*.
c : Jumlah kelas (5).
$\hat{y}_c$: Label untuk kelas c (1 jika benar dan 0 jika salah)
y_c : Input dari output fungsi *softmax.*

3.4.2 *Backward Propagation*

Proses pelatihan *backward propagation* dilakukan dari belakang atau secara mundur. *Backward propagation* dilakukan setelah melalui proses *forward propagation*. Tujuan utama dari proses pelatihan mundur adalah untuk memperbarui nilai bobot dan bias yang ada pada model. Nilai bobot diperbarui dengan menghitung nilai *gradient* dari fungsi kehilangan (*loss function*) terhadap nilai bobot dan bias yang terdapat pada jaringan model.

1. Menghitung *gradient error* pada *output layer*

Gradient error pada *output layer* dihitung dengan turunan yang didapatkan dari nilai prediksi (*softmax*) dan nilai *loss* yang berupa label aktual. Berikut adalah rumus perhitungan turunan dari *softmax* dan nilai *loss*:

$$\frac{\partial L}{\partial z_c} = y_c - \hat{y}_c \tag{3.19}$$

$$\delta z_c = \frac{\partial L}{\partial z} \tag{3.20}$$

Keterangan:
δz_c : *Output* probabilitas yang diprediksi model.
$\frac{\partial L}{\partial z_c}$: Turunan nilai *loss output layer*.
$\hat{y}_c$: Target label (1 jika benar dan 0 jika salah)
y_c : Nilai probabilitas dari fungsi *softmax*.

Setelah mendapatkan nilai turunan dari *output layer*, selanjutnya mencari nilai turunan dari nilai bobot (*weight*). Berikut adalah persamaan untuk mendapatkan turunan nilai bobot (*weight*):

$$\frac{\partial L}{\partial W_c} = \delta z_c \cdot \hat{z} \tag{3.21}$$

$$\frac{\partial L}{\partial W_c} = \frac{\partial L}{\partial z_c} \cdot \hat{z} \tag{3.22}$$

Keterangan:
δz_c : Nilai *gradient output layer*.
$\frac{\partial L}{\partial W_c}$: Turunan nilai bobot (*weight*).
$\hat{z}$: Nilai *hidden layer* dari persamaan 3.15 .

2. *Gradient error* pada *hidden layer*

Sama seperti perhitungan sebelumnya, bagian ini menghitung *gradient error* pada *hidden layer* dan nilai bobot (*weight*). Berikut adalah rumus perhitungan untuk mendapatkan nilai *gradient error* pada *hidden layer*:

$$\delta \hat{z}_c = \frac{\partial L}{\partial \hat{z}_c} = \frac{\partial L}{\partial W_c} \cdot \hat{z}_c \tag{3.23}$$

Keterangan:
$\delta \hat{z}_c$: Nilai *gradient hidden layer*.
$\hat{z}_c$: Nilai *hidden layer*.

Selanjutnya menghitung turunan nilai bobot (*weight*). Berikut adalah persamaan untuk menghitung nilai bobot:

$$\frac{\partial L}{\partial W_c} = \delta \hat{z}_c \cdot x_c = \frac{\partial L}{\partial \hat{z}_c} \cdot x_c \tag{3.24}$$

Keterangan:
δx_c : Nilai *gradient output layer*.
x_c : Nilai *input layer*.

Proses tersebut dilanjutkan sampai *input layer* pada *fully connected layer*. Kemudian proses turunan dilakukan hingga *feature extraction*. Proses ini disebut dengan *chain rule*.

3. *Update* nilai bobot dan bias

Setelah *gradient* dihitung, selanjutnya dilakukan pembaruan bobot dan bias menggunakan Adam *optimizer*. Berikut adalah operasi perhitungan pembaruan bobot dan bias menggunakan Adam:

$$g_t = \frac{\partial L}{\partial W_t} \tag{3.25}$$

Perbarui bias momen pertama:

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t \tag{3.26}$$

Perbarui bias momen kedua:

$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2 \tag{3.27}$$

Koreksi bias untuk momen pertama:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{3.28}$$

Koreksi bias untuk momen kedua:

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{3.29}$$

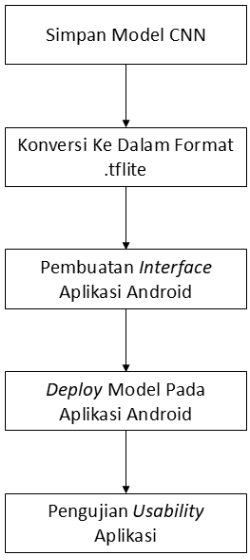
Pembaruan nilai bobot (*weight*):

$$W_{t(new)} = W_t - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{3.30}$$

Keterangan:
W_t : Bobot (*weight*).
η : *Learning rate*.
g_t : *Gradient* dari *loss function* terhadap W_t.
β_1, β_2 : Parameter *decay rate*.
ϵ : Konstanta kecil untuk mencegah pembagian dengan nol.

3.5 Integrasi *Mobile*

Integrasi sistem identifikasi dengan Android dilakukan dengan cara *on-device*. Cara ini menjalankan seluruh proses identifikasi penyakit pada tanaman padi secara lokal pada perangkat Android tanpa perlu menggunakan internet. Sehingga, proses dijalankan tanpa adanya gangguan jaringan dan tanpa perlu mengirimkan data ke server eksternal atau *cloud*. Model terbaik dari hasil pengujian skenario, kemudian dikonversi ke dalam format .tflite menggunakan *framework* TensorFlow Lite. *Framework* tersebut dapat membuat aplikasi *machine learning* berjalan pada perangkat *mobile* atau *embedded device* (Alden & Sari, 2023). Sebelum di-*deploy* ke dalam aplikasi Android, perlu dilakukan pembuatan *user interface* (UI) aplikasi Android. *Interface* adalah tampilan ketika *user* atau pengguna membuka aplikasi. Setelah *interface* aplikasi telah diselesaikan, maka model akan di-*deploy* ke dalam aplikasi. Proses integrasi *mobile* dapat dilihat pada Gambar 3. 13.



Gambar 3. 13 Proses Integrasi *Mobile*

1. Simpan Model CNN

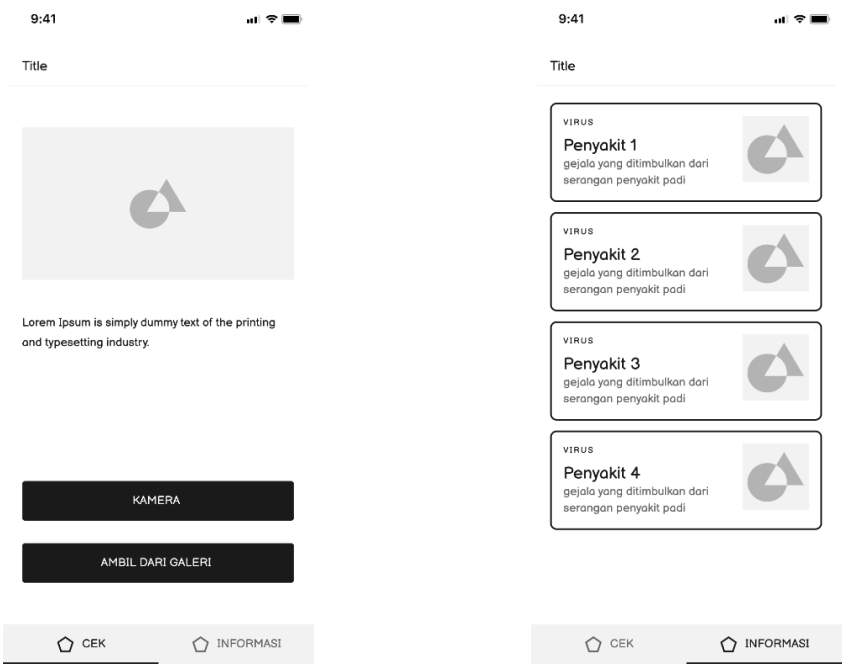
Model *Convolutional Neural Network* (CNN) MiniVGGNet dan kustomisasi MiniVGGNet yang telah dilatih dengan beberapa skenario yang berbeda, kemudian disimpan ke dalam format .h5. Format tersebut memiliki kemampuan untuk menyimpan arsitektur model, bobot (*weights*), dan konfigurasi pelatihan secara utuh. Model yang memiliki tingkat akurasi yang tinggi dari semua skenario pengujian akan dilanjutkan ke tahap selanjutnya untuk dilakukan konversi ke dalam format .tflite.

2. Konversi Model ke Format .tflite

Model yang telah dipilih dari tahapan sebelumnya kemudian diubah atau dikonversi menjadi format .tflite. Format tersebut dipilih karena *compatible* untuk diterapkan di *code editor* seperti Android Studio atau Flutter. Dalam proses konversi, model dioptimalkan agar dapat berjalan pada prosesor *smartphone* (CPU/GPU). Format tersebut merupakan bagian dari *framework* TensorFlow Lite, yang dirancang khusus untuk menjalankan model *machine learning* pada perangkat *mobile* dan *embedded system* (Alden & Sari, 2023).

3. Pembuatan *User Interface* Aplikasi Android

Sebelum model dijalankan di aplikasi Android, terlebih dahulu dilakukan pembuatan *User Interface* (UI) pada aplikasi Android. Ini bertujuan agar aplikasi yang dibuat memiliki tampilan yang menarik, tanpa melupakan fungsi sebagai aplikasi untuk pendeteksi penyakit pada tanaman padi. Komponen *user interface* disusun agar pengguna dapat memilih citra hingga menampilkan prediksi penyakit padi. Gambar 3. 14 menampilkan *Low-Fidelity* (Lo-Fi) aplikasi android.



Gambar 3. 14 *User Interface* Aplikasi, (kiri) Beranda, (kanan) informasi

4. *Deployment* Model CNN Pada Aplikasi Android

Tampilan pengguna atau *User Interface* (UI) aplikasi Android yang telah dibuat, kemudian diintegrasikan dengan model *Convolutional Neural Network* (CNN). Proses *deployment* dilakukan dengan mengintegrasikan TensorFlow Lite Interpreter ke dalam aplikasi untuk memuat model dari *folder assets*. Selain itu, aplikasi juga menjalankan tahapan *pre-processing* citra, seperti mengubah ukuran gambar dan normalisasikan nilai piksel agar sesuai dengan bentuk input yang diperlukan oleh model.

5. Pengujian *Usability* Aplikasi Android

Pengujian aplikasi dilakukan dengan menghitung skor *System Usability Scale* (SUS) (Brooke, 1995). Tahapan ini untuk menilai apakah aplikasi yang telah dibuat sudah sesuai dengan kebutuhan pengguna. Tabel 3. 4 menunjukkan evaluasi dengan

skor SUS yang terdiri dari 10 pertanyaan dalam bahasa indonesia oleh Ardhana (2021).

Tabel 3. 4 Pertanyaan SUS

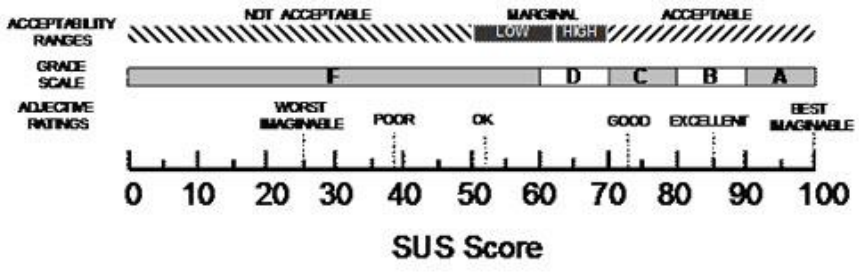
Kode	Pernyataan
Q1	Saya akan sering menggunakan aplikasi ini
Q2	Saya menilai aplikasi ini terlalu kompleks (memuat banyak hal yang tidak perlu)
Q3	Saya menilai aplikasi ini mudah untuk digunakan
Q4	Saya membutuhkan bantuan orang lain untuk menggunakan aplikasi ini
Q5	Saya menilai fitur yang disediakan pada aplikasi ini dirancang dan disiapkan dengan baik
Q6	Saya menilai terlalu banyak inkonsistensi pada aplikasi ini
Q7	Saya merasa kebanyakan orang akan mudah menggunakan aplikasi ini dengan cepat
Q8	Saya menilai aplikasi ini sangat rumit untuk digunakan
Q9	Saya merasa tidak ada hambatan dalam menggunakan aplikasi ini
Q10	Saya perlu belajar banyak hal sebelum saya dapat menggunakan aplikasi ini dengan baik

Perhitungan skor SUS penting untuk memastikan hasil yang akurat. Prinsip dasar perhitungannya adalah menjumlahkan konstribusi skor dari setiap item. Skala perhitungan menggunakan skala likert 1-5, lalu setiap item diubah menjadi nilai antara 0 dan 4. Langkah detail perhitungan skor sebagai berikut:

a. Untuk item bernomor ganjil (1, 3, 5, 7, 9), kontribusi skor didapatkan dari posisi skala responden dikurangi 1. Misalnya, jika respoden menandai 3, kontribusinya adalah 3 – 1 = 2.
b. Untuk item bernomor genap (2, 4, 6, 8, 10), kontribusi skor didapatkan dari nilai 5 dikurangi dengan posisi skala responden. Misalnya, jika responden menandai 1, kontribusinya adalah 5 – 1 = 4.

Setelah semua kontribusi item didapatkan, lalu nilai kontribusi dijumlahkan. Hasil penjumlahan dikalikan dengan 2,5 untuk mendapatkan skor SUS. Misalnya, jika kontribusi skor adalah 30, maka skor SUS adalah 30 x 2,5 = 75. Hasil dari skor

SUS memiliki rentang 0 hingga 100. Skor sus yang didapatkan kemudian dikonversi menjadi *adjective rating*. Ini berfungsi untuk memberikan makna atau maksud dari hasil yang didapatkan dari nilai skor. Menunjukkan nilai yang *adjective* berdasarkan nilai yang didapatkan (Bangor, 2009)



Gambar 3. 15 Peringkat *Adjective* Skor SUS

BAB IV

UJI COBA DAN PEMBAHASAN

4.1 Skenario Pengujian

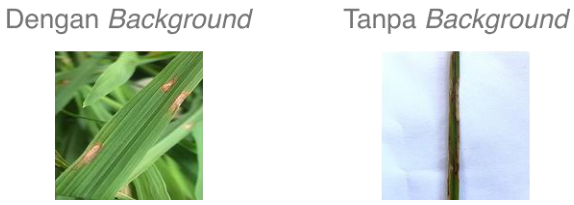
Penelitian ini menggunakan beberapa skenario penyusun. Skenario ini berfungsi untuk menguji model yang telah dibuat. Penerapan skenario diharapkan mampu membuat model menjadi optimal dalam mengidentifikasi atau mengklasifikasikan jenis penyakit pada tanaman padi berbasis Android. Di dalam skenario terdapat pembagian skenario mayor dan skenario minor. Skenario mayor adalah skenario yang skalanya mencakup dataset dan model. Skenario minor adalah skenario yang mencakup nilai *hyperparameter* yang diterapkan pada model.

Skenario penyusun awal dari penelitian ini adalah penggunaan model CNN yang merupakan skenario mayor. Model CNN yang digunakan dibagi menjadi dua, yaitu arsitektur MiniVGGNet (Gambar 3. 3) dan kustomisasi arsitektur MiniVGGNet (Gambar 3. 4). Perbedaan pada kedua model terdapat pada jumlah parameter dan banyaknya lapisan konvolusi. Model MiniVGGNet memiliki parameter yang lebih besar dibandingkan dengan kustom MiniVGGNet. Ini disebabkan oleh penggunaan *flatten* pada model MiniVGGNet dan model kustom menggunakan *Global Average Pooling* (GAP). Tabel 4. 1 menunjukkan skenario model CNN.

Tabel 4. 1 Model CNN

Skenario	Model CNN
Ori	MiniVGGNet *Original* (Gambar 3. 3)
Add	Kustomisasi MiniVGGNet (Gambar 3. 4)

Skenario kedua adalah penggunaan dataset, yang juga termasuk skenario mayor. Skenario ini menggunakan dataset yang berbeda dalam proses *training* dan *testing*. Pembeda dalam dataset tersebut adalah data yang tanpa *background* dan dengan *background*. Hal ini dilakukan untuk mengetahui apakah model lebih mudah memahami gambar dengan *background* atau dengan *no-background* Gambar 4. 1 adalah perbedaan dataset dengan dan tanpa *backgroud* dan Tabel 4. 2 adalah skenario penggunaan dataset.



Gambar 4. 1 Perbedaan Dataset

Tabel 4. 2 Penggunaan Dataset

Skenario	**Data *Train***	**Data *Test***
X	*No-background*	*No-background*
Y	*Background*	*Background*

Skenario ketiga adalah perubahaan nilai pada *dropout regularization*. Skenario ini termasuk ke dalam skenario minor. Skenario ini berpengaruh untuk menghilangkan *overfitting* pada model CNN. Berdasarkan Srivastava *et al.* (2014), nilai optimal probabilitas dari *dropout regularization* adalah 0,5. Pada penelitian ini nilai yang digunakan dibawah 0,5 karena model yang digunakan tidak memiliki kompleksitas yang tinggi. Tabel 4. 3 menunjukkan skenario *dropout regularization*.

Tabel 4. 3 *Dropout Regularization*

Skenario	***Dropout***
a	0,2
b	0,4

Skenario terakhir adalah jumlah *batch size*, yang termasuk ke dalam skenario minor. Jumlah *batch size* memengaruhi seberapa cepat proses *training* selesai dalam satu *epoch*. Pada penelitian ini, batch size yang digunakan tidak terlalu besar karena disesuaikan dengan jumlah dataset yang tersedia. Pemilihan ukuran *batch size* juga mempertimbangkan keterbatasan memori agar proses *training* tetap berjalan stabil. Tabel 4. 4 adalah skenario *batch size*.

Tabel 4. 4 *Batch Size*

Skenario	***Batch size***
m	32
n	64

Dari beberapa skenario penyusun tersebut, dilakukan kombinasi terhadap skenario tersebut untuk memperoleh model yang paling baik, optimal, dan mampu mencapai titik konvergensi. Pendekatan ini memungkinkan evaluasi yang lebih menyeluruh terhadap performa model pada berbagai skenario penyusun. Tabel 4. 5 menunjukkan kombinasi skenario.

Tabel 4. 5 Kombinasi Skenario

No. Skenario	**Skenario Pengujian**
1	Ori-X-a-m
2	Ori-X-a-n
3	Ori-X-b-m
4	Ori-X-b-n
5	Ori-Y-a-m
6	Ori-Y-a-n
7	Ori-Y-b-m
8	Ori-Y-b-n
9	Add-X-a-m
10	Add-X-a-n
11	Add-X-b-m
12	Add-X-b-n
13	Add-Y-a-m
14	Add-Y-a-n
15	Add-Y-b-m
16	Add-Y-b-n

Nilai dari beberapa parameter juga ditetapkan terlebih dahulu. Gabungan parameter ini disebut dengan *hyperparameter*. Berikut adalah nilai dari *hyperparameter* pada Tabel 4. 6.

Tabel 4. 6 *Nilai Hyperparameter*

Parameter	**Nilai/Jenis Parameter**
Epoch	100
Learning Rate	1e-4
Optimizer	Adam

Pemilihan *hyperparameter* berupa epoch sebanyak 100 dan optimizer Adam mengacu pada prinsip yang digunakan oleh Abasi *et al.* (2023). Dalam penelitian tersebut, dilakukan percobaan dengan dua jumlah epoch, yaitu 30 dan 100, dan diperoleh bahwa penggunaan 100 epoch menghasilkan akurasi yang lebih baik. Oleh karena itu, penelitian ini juga menerapkan 100 epoch. Sementara itu, pemilihan learning rate sebesar 1e-4 didasarkan pada rekomendasi dari paper Adam oleh Kingma & Ba (2017), yang menetapkan nilai default sebesar 1e-3. Namun, ketika nilai *default* tersebut digunakan dalam penelitian ini, model mengalami stagnasi. Oleh sebab itu, dilakukan penyesuaian dengan menurunkan *learning rate* menjadi 1e-4 agar proses penurunan *gradient error* berjalan lebih stabil.

Evaluasi model pada penelitian ini menggunakan *confusion matrix* dan *K-Fold cross validation*. *Confusion matrix* adalah tabel yang berisikan jumlah prediksi yang benar dan jumlah prediksi yang salah. Tujuan dari tabel ini untuk membandingkan nilai sebenarnya dengan nilai prediksi. Matriks ini tidak hanya menunjukkan tingkat kebenaran model, tetapi juga memetakan jenis kesalahan spesifik yang dilakukan oleh model dalam mengklasifikasikan data. *Confusion matrix* terbagi menjadi empat kategori seperti berikut:

1. *True Positive* (TP), model memprediksi positif dan nilai sebenarnya adalah positif.
2. *True Negative* (TN), model memprediksi negatif dan nilai sebenarnya adalah negatif.
3. *False Positive* (FP), model memprediksi positif, namun nilai sebenarnya adalah negatif
4. *False Negative* (FN), model memprediksi negatif, namun nilai sebenarnya positif

Kategori yang didapatkan dari tabel *confusion matrix* digunakan untuk mendapatkan *classification report* yang berupa nilai *accuracy*, *precision*, *recall*, dan *F1-Score*. *Accuracy* menunjukkan seberapa akurat model yang digunakan dalam mengidentifikasi atau mengklasifikasi dengan benar. *Precision* menunjukkan akurasi antara data yang diminta dengan hasil prediksi model. *Recall* menunjukkan keberhasilan model dalam menemukan kembali sebuah informasi. *F1-Score* menunjukkan perbandingan antara *precision* dan *recall* yang dibobotkan. Berikut adalah rumus untuk mendapatkan nilai-nilai tersebut.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

$$Precision = \frac{TP}{TP + FP} \tag{4.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{4.3}$$

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4.4}$$

K-Fold cross validation adalah metode yang digunakan untuk mengevaluasi kinerja dan generalisasi model. Teknik ini bertujuan untuk menilai seberapa baik model akan bekerja pada data baru yang belum pernah dilihat sebelumnya, serta untuk menghindari terjadinya *overfitting* (model hanya menghafal data latih) atau *underfitting*.

Dalam penelitian ini, diterapkan 5-*Fold Cross Validation* (k=5) yang dikhususkan pada model dengan hasil evaluasi terbaik di tahap pengujian. Metode ini dilakukan dengan merotasi data latih (*training*) dan data uji (*testing*) yang berbeda pada setiap *fold*-nya. Tujuannya adalah untuk memvalidasi konsistensi akurasi model, untuk memastikan bahwa performa tinggi yang dihasilkan bukan disebabkan oleh kebetulan dalam pembagian data (data splitting bias), namun karena model memiliki kemampuan generalisasi yang baik.

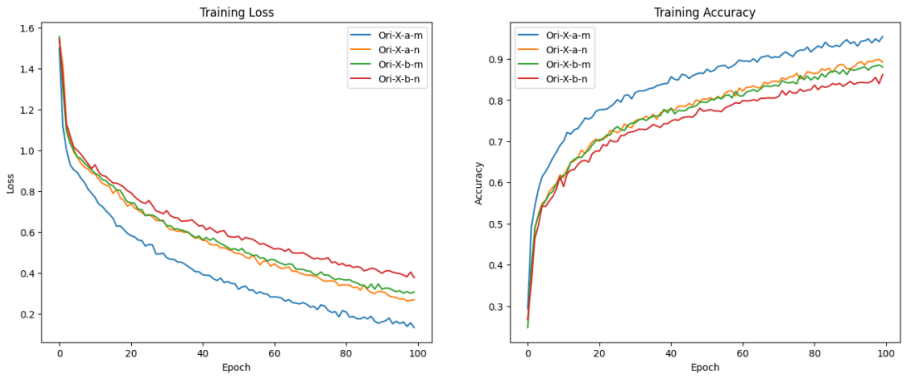
4.2 Hasil Uji Coba

Data citra yang dikumpulkan pada penelitian ini berjumlah 1.987 citra, dengan data *background* 1104 citra dan data *no-background* 883 citra. Dari data yang dikumpulkan tersebut, kemudian dilakukan pembagian rasio data *train* dan data *testing*. Rasio perbandingan diterapkan secara konsisten pada dua jenis skenario dataset yang digunakan. Skenario pertama, atau disebut dengan skenario X, menggunakan citra *no-background* baik pada data *train* dan data *test* dengan menerapkan rasio pembagian 90:10. Skenario kedua, atau disebut dengan skenario Y, menggunakan citra *background* pada kedua set data tersebut dengan rasio yang sama, yaitu 90:10. Pada kedua skenario tersebut diterapkan perbandingan rasio yang sama agar pembagian data yang digunakan untuk proses pelatihan model

menjadi konsisten, sekaligus menyediakan data *testing* yang cukup untuk mengukur performa secara objektif.

Hasil uji coba yang telah dilakukan dari 16 kombinasi skenario pengujian, terdapat hasil yang berbeda pada setiap skenario. Dilakukan beberapa rangkuman untuk meringkas dan mempermudah dalam memberikan hasil *training*. Rangkuman dari hasil *training* dibagi menjadi dua kelompok berdasarkan model CNN yang digunakan, yaitu Model MiniVGGNet untuk skenario pertama hingga kedelapan dan kustomisasi MiniVGGNet untuk skenario kesembilan hingga ke-16.

Gambar 4. 2 menampilkan perbedaan hasil pelatihan pada model MiniVGGNet yang menggunakan dataset *no-background*. Secara keseluruhan, model sudah dapat belajar dengan baik. Secara spesifik, skenario 1 (Ori-X-a-m) memiliki performa model yang lebih baik dari ketiga skenario lainnya. Hasil nilai akurasi yang didapatkan sebesar 96% dan nilai *loss* 0,1241 pada *epoch* ke 100.



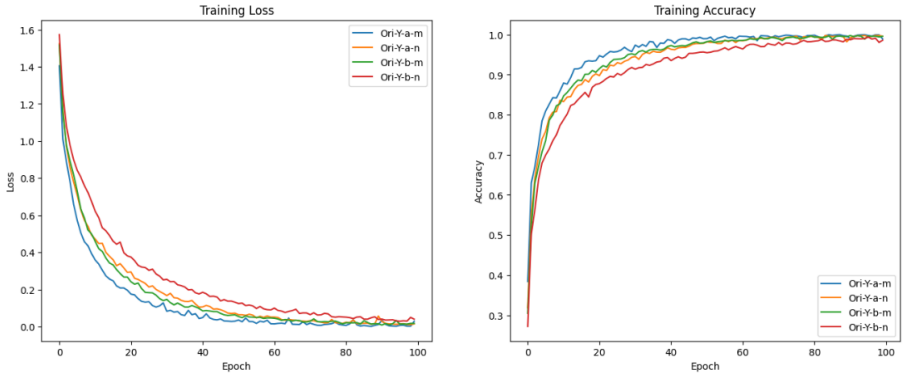
Gambar 4. 2 Hasil *Training* Skenario ke-1 sampai ke-4

Pengujian skenario yang telah dilakukan pada Gambar 4. 2, mendapatkan nilai *accuracy* dan nilai *loss* yang berbeda pada setiap skenario. Nilai *accuracy* dan nilai *loss* yang didapatkan pada *epoch* ke-100 ditampilkan pada Tabel 4. 7.

Tabel 4. 7 Nilai *Accuracy* dan Nilai *Loss* pada Skenario ke-1 sampai ke-4

Skenario	**Nilai *Loss***	**Nilai *Accuracy***
Ori-X-a-m	0,1241	96,00%
Ori-X-a-n	0,2528	89,91%
Ori-X-b-m	0,3026	88,28%
Ori-X-b-n	0,3815	85,76%

Gambar 4. 3 menampilkan hasil pelatihan pada model MiniVGGNet yang menggunakan dataset *background*. Secara umum, setiap pelatihan sudah berhasil konvergen lebih cepat dibandingkan skenario sebelumnya. Secara khusus, skenario 7 (Ori-Y-b-m) adalah skenario yang terbaik dari ketiga skenario lainnya. Skenario tersebut mendapatkan nilai akurasi 99,76% dan nilai *loss* 0,0110 pada *epoch* ke-100.



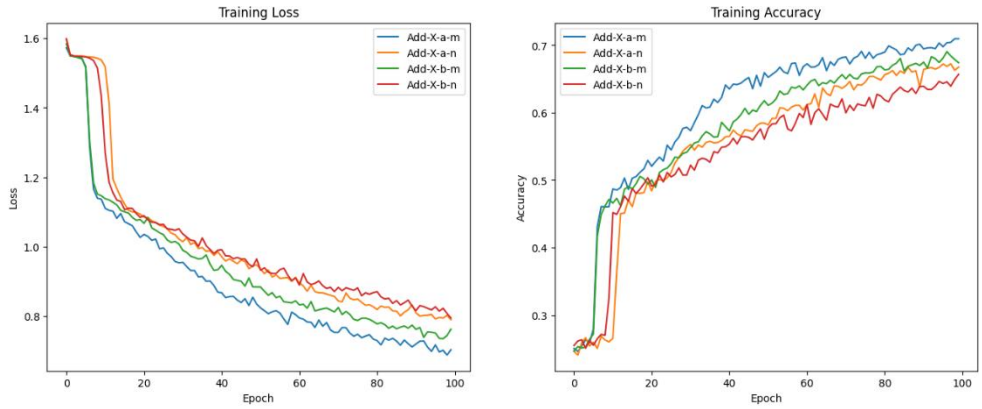
Gambar 4. 3 Hasil *Training* Skenario ke-5 sampai ke-8

Pengujian skenario yang telah dilakukan pada Gambar 4. 3, mendapatkan nilai *accuracy* dan nilai *loss* yang berbeda pada setiap skenario. Nilai *accuracy* dan nilai *loss* yang didapatkan pada *epoch* ke-100 ditampilkan pada Tabel 4. 8.

Tabel 4. 8 Nilai *Accuracy* dan Nilai *Loss* pada Skenario ke-5 sampai ke-8

Skenario	**Nilai *Loss***	**Nilai *Accuracy***
Ori-Y-a-m	0,0274	98,99%
Ori-Y-a-n	0,0112	99,73%
Ori-Y-b-m	0,0110	99,76%
Ori-Y-b-n	0,0432	98,41%

Gambar 4. 4 menampilkan hasil pelatihan pada model kustom MiniVGGNet yang menggunakan dataset *no-background*. Seluruh model awalnya mengalami kesulitan yang grafiknya cenderung datar (*plateau*), namun setelah itu model tetap berhasil meningkatkan nilai akurasi secara bertahap. Skenario 9 (Add-X-a-m) memiliki tingkat akurasi 71% yang tinggi dan nilai loss 0,7061 yang lebih rendah dibandingkan skenario yang lain.



Gambar 4. 4 Hasil *Training* Skenario ke-9 sampai ke-12

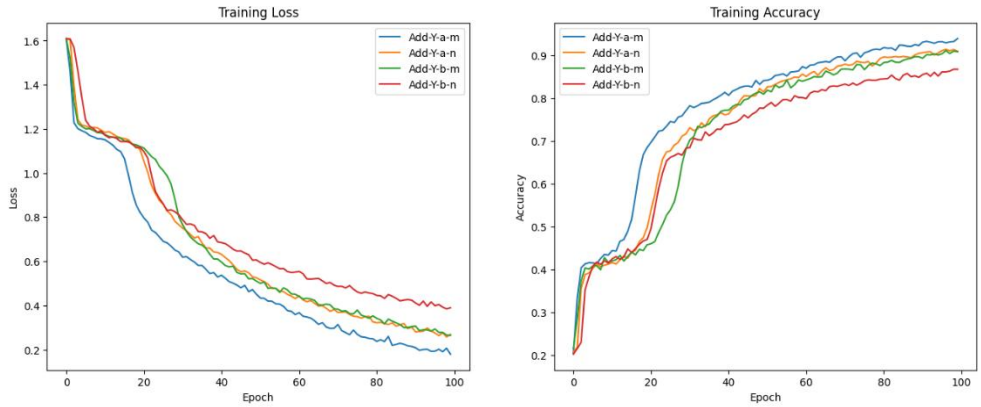
Pengujian skenario yang telah dilakukan pada Gambar 4. 4, mendapatkan nilai *accuracy* dan nilai *loss* yang berbeda pada setiap skenario. Nilai *accuracy* dan nilai *loss* yang didapatkan pada *epoch* ke-100 ditampilkan pada Tabel 4. 9.

Tabel 4. 9 Nilai *Accuracy* dan Nilai *Loss* pada Skenario ke-9 sampai ke-12

Skenario	**Nilai *Loss***	**Nilai *Accuracy***
Add-X-a-m	0,7061	71,26%
Add-X-a-n	0,7821	67,68%
Add-X-b-m	0,7821	66.94%
Add-X-b-n	0,8040	65,44%

Gambar 4. 5 menampilkan hasil pelatihan pada model kustom MiniVGGNet yang menggunakan dataset *background*. Seluruh model menampilkan konvergen yang terlambat atau grafiknya cenderung datar di awal seperti skenario 9-12, namun

setelah itu, model tetap berhasil mampu belajar dan meningkatkan nilai akurasi secara bertahap. Grafik yang ditampilkan pada setiap skenario lebih stabil dibandingkan dengan model yang menggunakan dataset *no-background*. Skenario 13 (Add-Y-a-m) memiliki tingkat akurasi 93,80% dan nilai *loss* 0,1896. Hal ini membuktikan bahwa skenario tersebut memiliki kemampuan generalisasi yang paling baik di antara skenario lainnya.



Gambar 4. 5 Hasil *Training* Skenario ke-13 sampai ke-16

Pengujian skenario yang telah dilakukan pada Gambar 4. 5, didapatkan nilai *accuracy* dan nilai *loss* yang berbeda pada setiap skenario. Nilai *accuracy* dan nilai *loss* yang didapatkan pada epoch ke-100 ditampilkan pada Tabel 4. 10.

Tabel 4. 10 Nilai *Accuracy* dan Nilai *Loss* pada Skenario ke-13 sampai ke-16

Skenario	**Nilai *Loss***	**Nilai *Accuracy***
Add-Y-a-m	0,1896	93,80%
Add-Y-a-n	0,2793	90,77%
Add-Y-b-m	0,2587	91,77%
Add-Y-b-n	0,4065	85,81%

Berdasarkan hasil *training* yang didapatkan dari semua skenario pengujian, bahwa model MiniVGGNet memiliki tingkat akurasi yang lebih tinggi dibandingkan dengan model kustom. MiniVGGNet mendapatkan rata-rata nilai

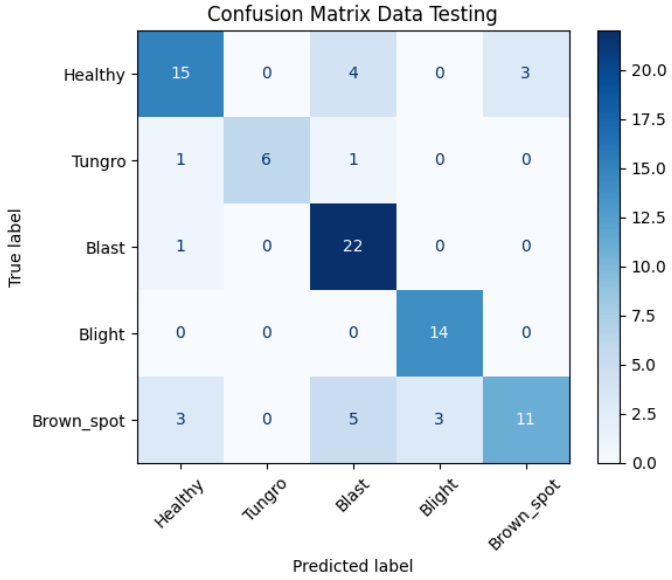
akurasi 94% dan rata-rata nilai *loss* 0,1442. Sedangkan model kustom, nilai akurasi yang didapatkan 79% dan nilai *loss* 0,5261.

Selain model, penggunaan dataset juga sangat berpengaruh pada nilai akurasi dan nilai *loss* yang didapatkan. Pada model MiniVGGNet yang menggunakan dataset Y memiliki nilai akurasi rata-rata 99% dan nilai *loss* yang didapatkan rata-rata 0,0232. Sedangkan model kustom mendapatkan nilai akurasi 90% dan nilai loss 0,2835. Hasil yang berbeda didapatkan pada setiap model yang menggunakan dataset X. Model MiniVGGNet mendapatkan rata-rata akurasi 89% dan nilai loss 0,2653, sedangkan model kustom mendapatkan nilai akurasi 67% dan nilai loss 0,7686. Perbedaan ini menunjukkan bahwa model MiniVGGNet dan model kustom MiniVGGNet lebih efektif ketika menggunakan skenario Y.

Selain itu, pengaturan hyperparameter seperti *dropout* dan *batch size* juga memengaruhi performa model. Pada Model MiniVGGNet yang menggunakan *dropout* 0,2 mendapatkan nilai akurasi 3% lebih tinggi dibandingkan dengan *dropout* 0,4 dan *batch size* 32 mendapatkan nilai akurasi 2% lebih tinggi dibandingkan dengan *batch size* 64. Pola yang sama terjadi pada model kustom MiniVGGNet, ketika menggunakan *dropout* 0,2 mendapatkan nilai akurasi 4% lebih tinggi dibandingkan dengan *dropout* 0,4, dan *batch size* 32 mendapatkan nilai akurasi 3% lebih tinggi dibandingkan dengan *batch size* 64. Hasil ini menunjukkan bahwa kombinasi arsitektur yang tepat, penggunaan dataset, dan pemilihan *hyperparameter* yang optimal sangat berperan dalam meningkatkan performa model. Analisis selanjutnya adalah melakukan hasil *testing*, untuk melihat performa model pada data yang belum pernah dilihat sebelumnya.

4.2.1 Skenario Ori-X-a-m

Hasil *testing* menggunakan skenario ini mendapatkan nilai yang kurang baik atau *overfitting*, karena jumlah prediksi kelas yang benar masih sedikit dan perbedaan hasil akurasi *testing* dan *training* cukup jauh. Gambar 4. 6 menunjukkan hasil evaluasi menggunakan *confusion matrix*.



Gambar 4. 6 *Confusion Matrix* Skenario 1 (Ori-X-a-m)

Berdasarkan *confusion matrix* pada Gambar 4. 6, didapatkan *classification report* berupa *precision*, *recall*, dan *F1-Score* pada Tabel 4. 11.

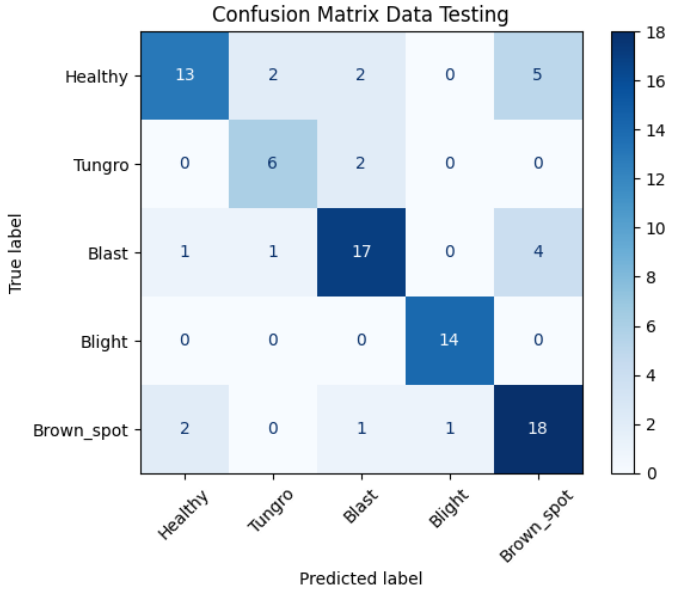
Tabel 4. 11 Nilai *Precision*, *Recall*, *F1-Score* Skenario 1 (Ori-X-a-m)

Kelas Penyakit	***Precision***	***Recall***	***F1-Score***
Blast	69%	96%	80%
Blight	82%	100%	90%
Healthy	75%	68%	71%
Tungro	100%	75%	86%
Brown Spot	79%	50%	61%
Rata-Rata	81%	78%	78%

Hasil *testing* menggunakan *confusion matrix* pada skenario Ori-X-a-m mendapatkan nilai akurasi sebesar 76%.

4.2.2 Skenario Ori-X-a-n

Hasil *testing* menggunakan skenario ini mendapatkan nilai yang kurang baik atau *overfitting*, karena jumlah prediksi kelas yang benar masih sedikit dan perbedaan hasil akurasi *testing* dan *training* cukup jauh. Gambar 4. 7 menunjukkan hasil evaluasi menggunakan *confusion matrix*.



Gambar 4. 7 *Confusion Matrix* Skenario 2 (Ori-X-a-n)

Berdasarkan *confusion matrix* pada Gambar 4. 7, didapatkan classification report berupa *precision*, *recall*, dan *F1-Score* pada Tabel 4. 12.

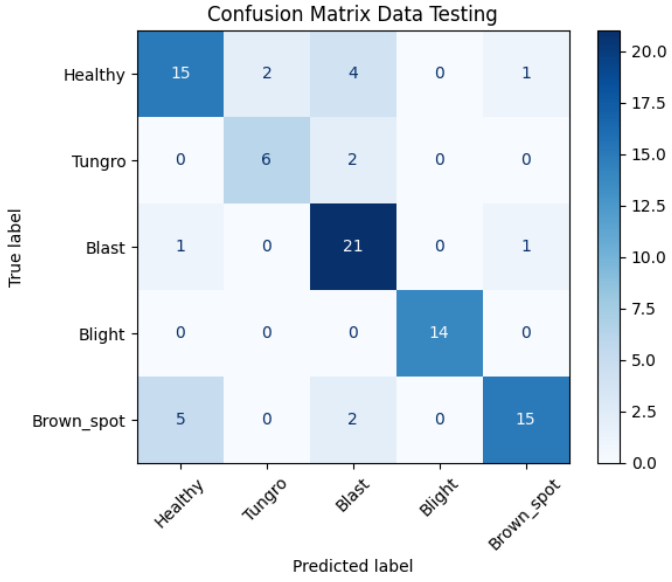
Tabel 4. 12 Nilai *Precision*, *Recall*, *F1-Score* Skenario 2 (Ori-X-a-n)

Kelas Penyakit	***Precision***	***Recall***	***F1-Score***
Blast	77%	74%	76%
Blight	93%	100%	97%
Healthy	81%	59%	68%
Tungro	67%	75%	71%
Brown Spot	67%	82%	73%
Rata-Rata	77%	78%	77%

Hasil dari *testing* menggunakan *confusion matrix* pada skenario Ori-X-a-n mendapatkan nilai akurasi sebesar 76%.

4.2.3 Skenario Ori-X-b-m

Hasil *testing* menggunakan skenario ini mendapatkan nilai yang cukup baik, walaupun jumlah prediksi kelas yang benar masih sedikit, namun perbedaan hasil akurasi *testing* dan *training* tidak jauh berbeda. Gambar 4. 8 menunjukkan hasil evaluasi menggunakan *confusion matrix*.



Gambar 4. 8 *Confusion Matrix* Skenario 3 (Ori-X-b-m)

Berdasarkan *confusion matrix* pada Gambar 4. 8, didapatkan *classification report* berupa *precision*, *recall*, dan *F1-Score* pada Tabel 4. 13.

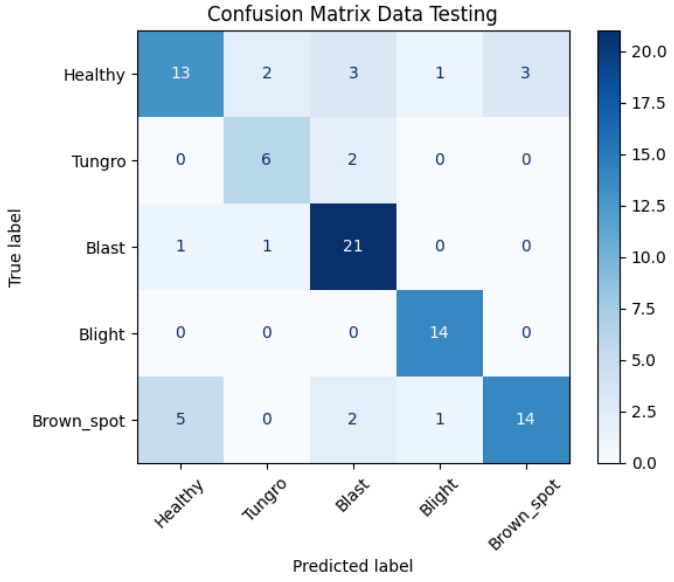
Tabel 4. 13 Nilai *Precision*, *Recall*, *F1-Score* Skenario 3 (Ori-X-b-m)

Kelas Penyakit	***Precision***	***Recall***	***F1-Score***
Blast	72%	91%	81%
Blight	100%	100%	100%
Healthy	71%	68%	70%
Tungro	75%	75%	75%
Brown Spot	88%	68%	77%
Rata-Rata	81%	80%	81%

Hasil dari *testing* menggunakan *confusion matrix* pada skenario Ori-X-b-m mendapatkan nilai akurasi sebesar 80%.

4.2.4 Skenario Ori-X-b-n

Hasil *testing* menggunakan skenario ini mendapatkan nilai yang kurang baik atau *overfitting*, karena jumlah prediksi kelas yang benar masih sedikit dan perbedaan hasil akurasi *testing* dan *training* cukup jauh. Gambar 4. 9 menunjukkan hasil evaluasi menggunakan *confusion matrix*.



Gambar 4. 9 *Confusion Matrix* Skenario 4 (Ori-X-b-n)

Berdasarkan *confusion matrix* pada Gambar 4. 9, didapatkan *classification report* berupa *precision*, *recall*, dan *F1-Score* pada Tabel 4. 14.

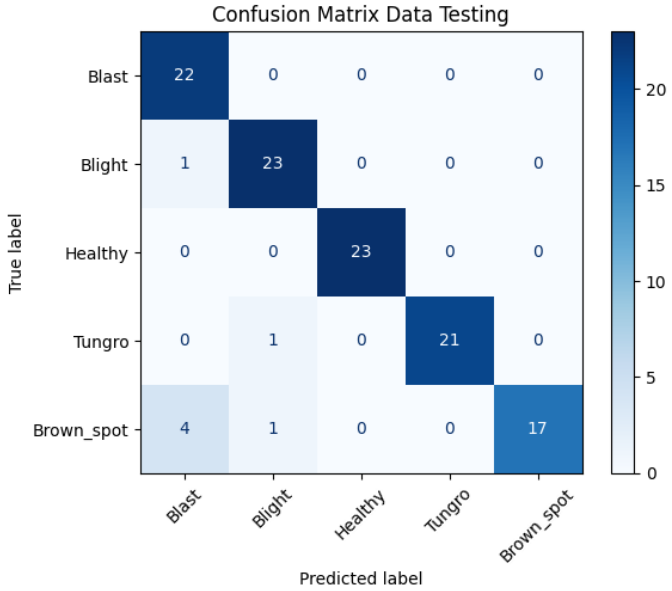
Tabel 4. 14 Nilai *Precision*, *Recall*, *F1-Score* Skenario 4 (Ori-X-b-n)

Kelas Penyakit	***Precision***	***Recall***	***F1-Score***
Blast	75%	91%	82%
Blight	88%	100%	93%
Healthy	68%	59%	63%
Tungro	67%	75%	71%
Brown Spot	82%	64%	72%
Rata-Rata	76%	78%	76%

Hasil dari *testing* menggunakan *confusion matrix* pada skenario Ori-X-b-n mendapatkan nilai akurasi sebesar 76%.

4.2.5 Skenario Ori-Y-a-m

Hasil *testing* menggunakan skenario ini mendapatkan nilai yang sangat baik, karena model dapat memprediksi penyakit setiap kelas secara benar dengan jumlah yang banyak, selain itu perbedaan hasil akurasi *testing* dan *training* tidak jauh berbeda. Gambar 4. 10 menunjukkan hasil evaluasi menggunakan *confusion matrix*.



Gambar 4. 10 *Confusion Matrix* Skenario 5 (Ori-Y-a-m)

Berdasarkan *confusion matrix* pada Gambar 4. 10, didapatkan *classification report* berupa *precision*, *recall*, dan *F1-Score* pada Tabel 4. 15.

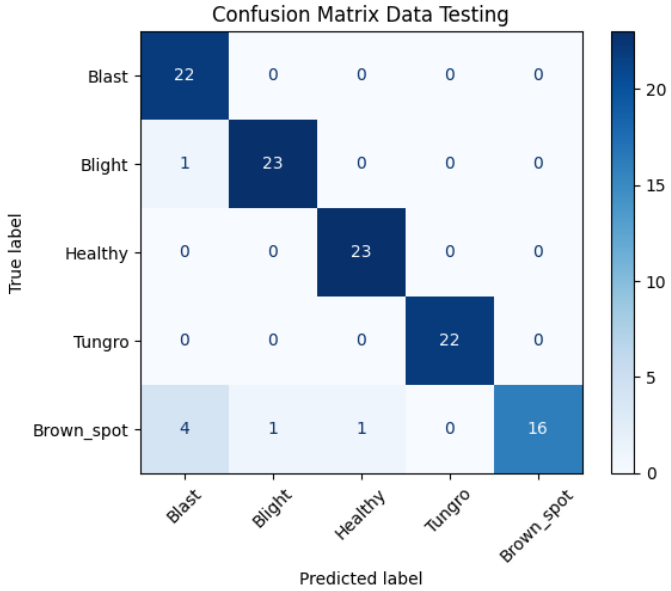
Tabel 4. 15 Nilai *Precision*, *Recall*, *F1-Score* Skenario 5 (Ori-Y-a-m)

Kelas Penyakit	***Precision***	***Recall***	***F1-Score***
Blast	81%	100%	90%
Blight	92%	96%	94%
Healthy	100%	100%	100%
Tungro	100%	95%	98%
Brown Spot	100%	77%	87%
Rata-Rata	95%	94%	94%

Hasil dari *testing* menggunakan *confusion matrix* pada skenario Ori-Y-a-m mendapatkan nilai akurasi sebesar 94%.

4.2.6 Skenario Ori-Y-a-n

Hasil *testing* menggunakan skenario ini mendapatkan nilai yang sangat baik, karena model dapat memprediksi penyakit setiap kelas secara benar dengan jumlah yang banyak, selain itu perbedaan hasil akurasi *testing* dan *training* tidak jauh berbeda. Gambar 4. 11 menunjukkan hasil evaluasi menggunakan *confusion matrix*.



Gambar 4. 11 *Confusion Matrix* Skenario 6 (Ori-Y-a-n)

Berdasarkan *confusion matrix* pada Gambar 4. 11, didapatkan *classification report* berupa *precision*, *recall*, dan *F1-Score* pada Tabel 4. 16.

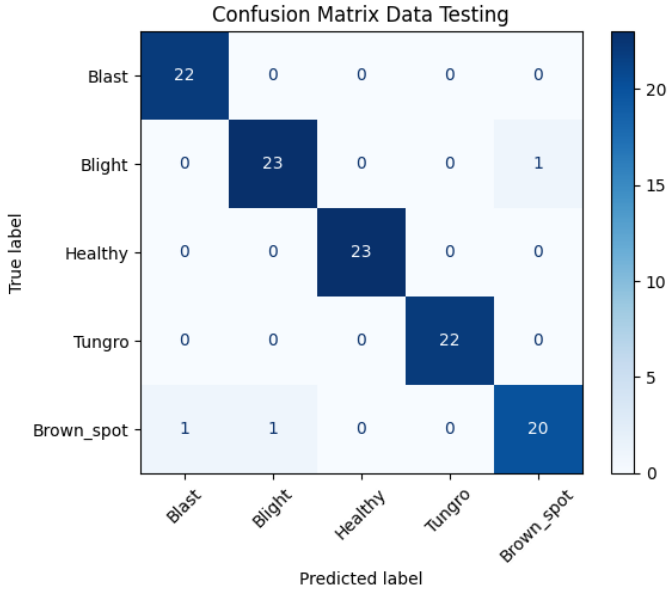
Tabel 4. 16 Nilai *Precision*, *Recall*, *F1-Score* Skenario 6 (Ori-Y-a-n)

Kelas Penyakit	***Precision***	***Recall***	***F1-Score***
Blast	81%	100%	90%
Blight	96%	96%	96%
Healthy	96%	100%	98%
Tungro	100%	100%	100%
Brown Spot	100%	73%	84%
Rata-Rata	95%	94%	94%

Hasil dari *testing* menggunakan *confusion matrix* pada skenario Ori-Y-a-n mendapatkan nilai akurasi sebesar 94%.

4.2.7 Skenario Ori-Y-b-m

Hasil *testing* menggunakan skenario ini mendapatkan nilai yang sangat baik, karena model dapat memprediksi penyakit setiap kelas secara benar dengan jumlah yang banyak, selain itu perbedaan hasil akurasi *testing* dan *training* tidak jauh berbeda. Gambar 4. 12 menunjukkan hasil evaluasi menggunakan *confusion matrix*.



Gambar 4. 12 *Confusion Matrix* Skenario 7 (Ori-Y-b-m)

Berdasarkan *confusion matrix* pada Gambar 4. 12, didapatkan *classification report* berupa *precision*, *recall*, dan *F1-Score* pada Tabel 4. 17.

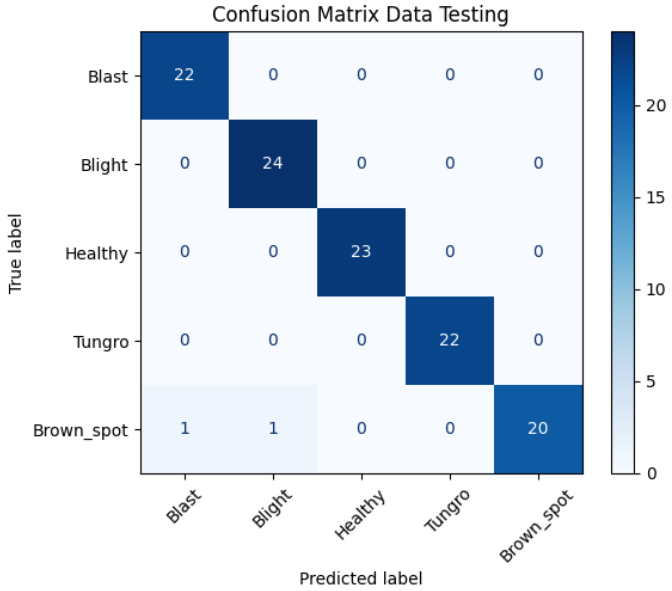
Tabel 4. 17 Nilai *Precision*, *Recall*, *F1-Score* Skenario 7 (Ori-Y-b-m)

Kelas Penyakit	***Precision***	***Recall***	***F1-Score***
Blast	96%	100%	98%
Blight	96%	96%	96%
Healthy	100%	100%	100%
Tungro	100%	100%	100%
Brown Spot	95%	91%	93%
Rata-Rata	97%	97%	97%

Hasil dari *testing* menggunakan *confusion matrix* pada skenario Ori-Y-b-m mendapatkan nilai akurasi sebesar 97%.

4.2.8 Skenario Ori-Y-b-n

Hasil *testing* menggunakan skenario ini mendapatkan nilai yang sangat baik, karena model dapat memprediksi penyakit setiap kelas secara benar dengan jumlah yang banyak, selain itu perbedaan hasil akurasi *testing* dan *training* tidak jauh berbeda. Gambar 4. 13 menunjukkan hasil evaluasi menggunakan *confusion matrix*.



Gambar 4. 13 *Confusion Matrix* Skenario 8 (Ori-Y-b-n)

Berdasarkan *confusion matrix* pada Gambar 4. 13, didapatkan *classification report* berupa *precision*, *recall*, dan *F1-Score* pada Tabel 4. 18.

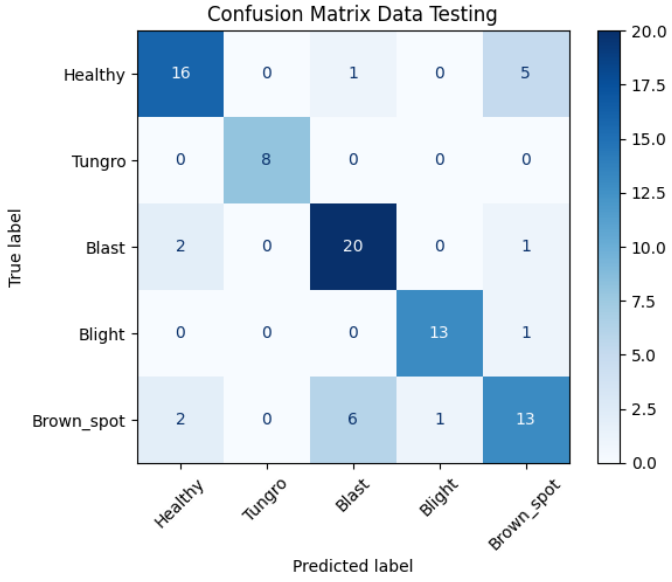
Tabel 4. 18 Nilai *Precision*, *Recall*, *F1-Score* Skenario 8 (Ori-Y-b-n)

Kelas Penyakit	***Precision***	***Recall***	***F1-Score***
Blast	96%	100%	98%
Blight	96%	100%	98%
Healthy	100%	100%	100%
Tungro	100%	100%	100%
Brown Spot	100%	91%	95%
Rata-Rata	98%	98%	98%

Hasil dari *testing* menggunakan *confusion matrix* pada skenario Ori-Y-b-n mendapatkan nilai akurasi sebesar 98%.

4.2.9 Skenario Add-X-a-m

Hasil *testing* menggunakan skenario ini mendapatkan nilai yang cukup baik, karena nilai akurasi yang didapatkan antara proses *training* dan proses *testing* tidak jauh berbeda, walaupun nilai akurasi yang didapatkan masih belum optimal. Gambar 4. 14 menunjukkan hasil evaluasi menggunakan *confusion matrix*.



Gambar 4. 14 *Confusion Matrix* Skenario 9 (Add-X-a-m)

Berdasarkan *confusion matrix* pada Gambar 4. 14, didapatkan *classification report* berupa *precision*, *recall*, dan *F1-Score* pada Tabel 4. 19.

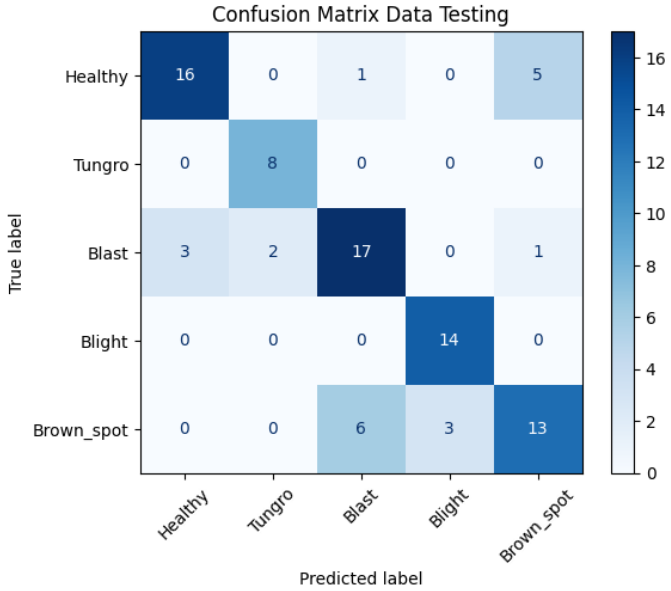
Tabel 4. 19 Nilai *Precision*, *Recall*, *F1-Score* Skenario 9 (Add-X-a-m)

Kelas Penyakit	***Precision***	***Recall***	***F1-Score***
Blast	74%	87%	80%
Blight	93%	93%	93%
Healthy	80%	73%	76%
Tungro	100%	100%	100%
Brown Spot	65%	59%	62%
Rata-Rata	82%	82%	82%

Hasil dari *testing* menggunakan *confusion matrix* pada skenario Add-X-a-m mendapatkan nilai akurasi sebesar 79%.

4.2.10 Skenario Add-X-a-n

Hasil *testing* menggunakan skenario ini mendapatkan nilai yang cukup baik, karena nilai akurasi yang didapatkan antara proses *training* dan proses *testing* tidak jauh berbeda, walaupun nilai akurasi yang didapatkan masih belum optimal. Gambar 4. 15 menunjukkan hasil evaluasi menggunakan *confusion matrix*.



Gambar 4. 15 *Confusion Matrix* Skenario 10 (Add-X-a-n)

Berdasarkan *confusion matrix* pada Gambar 4. 15, didapatkan *classification report* berupa *precision*, *recall*, dan *F1-Score* pada Tabel 4. 20.

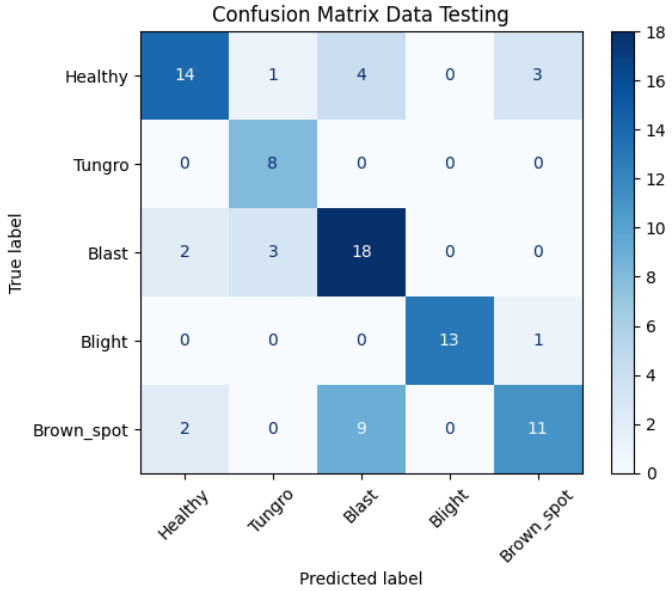
Tabel 4. 20 Nilai *Precision*, *Recall*, *F1-Score* Skenario 10 (Add-X-a-n)

Kelas Penyakit	***Precision***	***Recall***	***F1-Score***
Blast	71%	74%	72%
Blight	82%	100%	90%
Healthy	84%	73%	78%
Tungro	80%	100%	89%
Brown Spot	68%	59%	63%
Rata-Rata	77%	81%	78%

Hasil dari *testing* menggunakan *confusion matrix* pada skenario Add-X-a-n mendapatkan nilai akurasi sebesar 76%.

4.2.11 Skenario Add-X-b-m

Hasil *testing* menggunakan skenario ini mendapatkan nilai yang cukup baik, karena nilai akurasi yang didapatkan antara proses *training* dan proses *testing* tidak jauh berbeda, walaupun nilai akurasi yang didapatkan masih belum optimal. Gambar 4. 16 menunjukkan hasil evaluasi menggunakan *confusion matrix*.



Gambar 4. 16 *Confusion Matrix* Skenario 11 (Add-X-b-m)

Berdasarkan *confusion matrix* pada Gambar 4. 16, didapatkan *classification report* berupa *precision*, *recall*, dan *F1-Score* pada Tabel 4. 21.

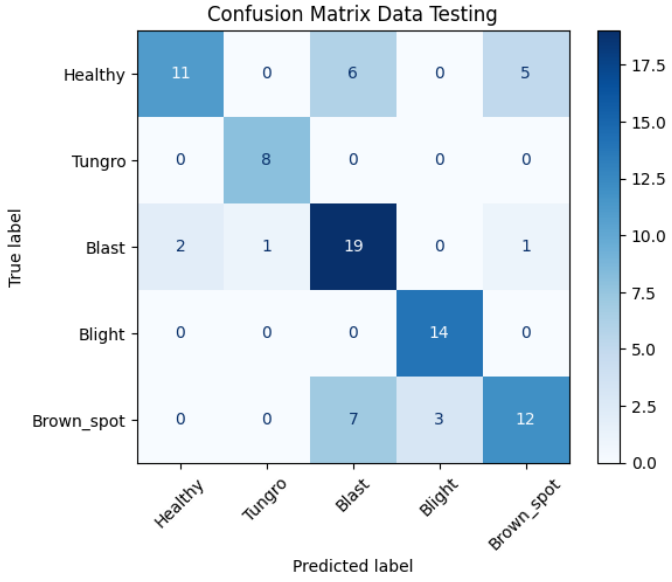
Tabel 4. 21 Nilai *Precision*, *Recall*, *F1-Score* Skenario 11 (Add-X-b-m)

Kelas Penyakit	***Precision***	***Recall***	***F1-Score***
Blast	58%	78%	67%
Blight	100%	93%	96%
Healthy	78%	64%	70%
Tungro	67%	100%	80%
Brown Spot	73%	50%	59%
Rata-Rata	75%	77%	74%

Hasil dari *testing* menggunakan *confusion matrix* pada skenario Add-X-b-m mendapatkan nilai akurasi sebesar 72%.

4.2.12 Skenario Add-X-b-n

Hasil *testing* menggunakan skenario ini mendapatkan nilai yang cukup baik, karena nilai akurasi yang didapatkan antara proses *training* dan proses *testing* tidak jauh berbeda, walaupun nilai akurasi yang didapatkan masih belum optimal. Gambar 4. 17 menunjukkan hasil evaluasi menggunakan *confusion matrix*.



Gambar 4. 17 *Confusion Matrix* Skenario 12 (Add-X-a-n)

Berdasarkan *confusion matrix* pada Gambar 4. 17, didapatkan *classification report* berupa *precision*, *recall*, dan *F1-Score* pada Tabel 4. 22.

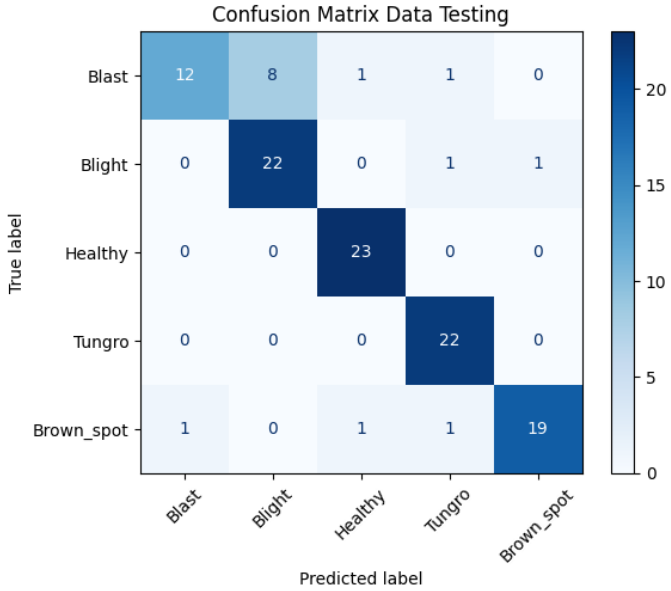
Tabel 4. 22 Nilai *Precision*, *Recall*, *F1-Score* Skenario 12 (Add-X-b-n)

Kelas Penyakit	***Precision***	***Recall***	***F1-Score***
Blast	59%	83%	69%
Blight	82%	100%	90%
Healthy	85%	50%	63%
Tungro	89%	100%	94%
Brown Spot	67%	55%	60%
Rata-Rata	76%	78%	75%

Hasil dari *testing* menggunakan *confusion matrix* pada skenario Add-X-b-n mendapatkan nilai akurasi sebesar 72%.

4.2.13 Skenario Add-Y-a-m

Hasil *testing* menggunakan skenario ini mendapatkan nilai yang baik, karena model dapat memprediksi penyakit setiap kelas secara benar dengan jumlah yang banyak, selain itu perbedaan hasil akurasi *testing* dan *training* tidak jauh berbeda. Gambar 4. 18 menunjukkan hasil evaluasi menggunakan *confusion matrix*.



Gambar 4. 18 Confusion Matrix Skenario 13 (Add-Y-a-m)

Berdasarkan *confusion matrix* pada Gambar 4. 18, didapatkan *classification report* berupa *precision*, *recall*, dan *F1-Score* pada Tabel 4. 23.

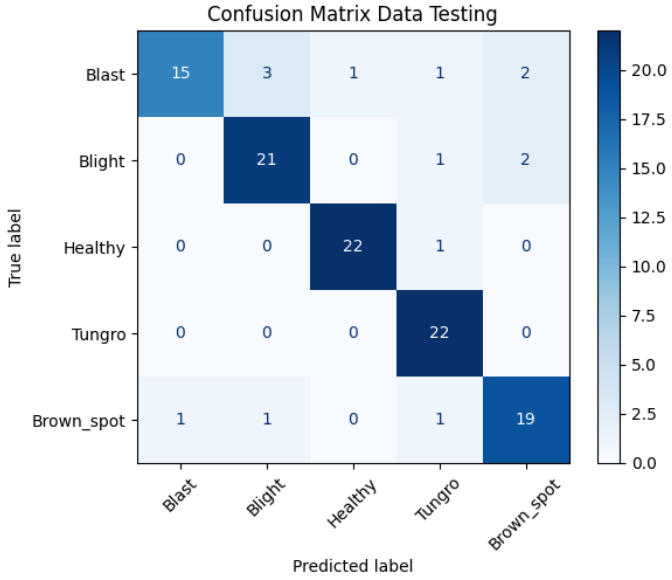
Tabel 4. 23 Nilai *Precision*, *Recall*, *F1-Score* Skenario 13 (Add-Y-a-m)

Kelas Penyakit	***Precision***	***Recall***	***F1-Score***
Blast	92%	55%	69%
Blight	73%	92%	81%
Healthy	92%	100%	96%
Tungro	88%	100%	94%
Brown Spot	95%	86%	90%
Rata-Rata	88%	87%	86%

Hasil dari *testing* menggunakan *confusion matrix* pada skenario Add-Y-a-m mendapatkan nilai akurasi sebesar 87%.

4.2.14 Skenario Add-Y-a-n

Hasil *testing* menggunakan skenario ini mendapatkan nilai yang baik, karena model dapat memprediksi penyakit setiap kelas secara benar dengan jumlah yang banyak, selain itu perbedaan hasil akurasi *testing* dan *training* tidak jauh berbeda. Gambar 4. 19 menunjukkan hasil evaluasi menggunakan *confusion matrix*.



Gambar 4. 19 *Confusion Matrix* Skenario 14 (Add-Y-a-n)

Berdasarkan *confusion matrix* pada Gambar 4. 19, didapatkan *classification report* berupa *precision*, *recall*, dan *F1-Score* pada Tabel 4. 24.

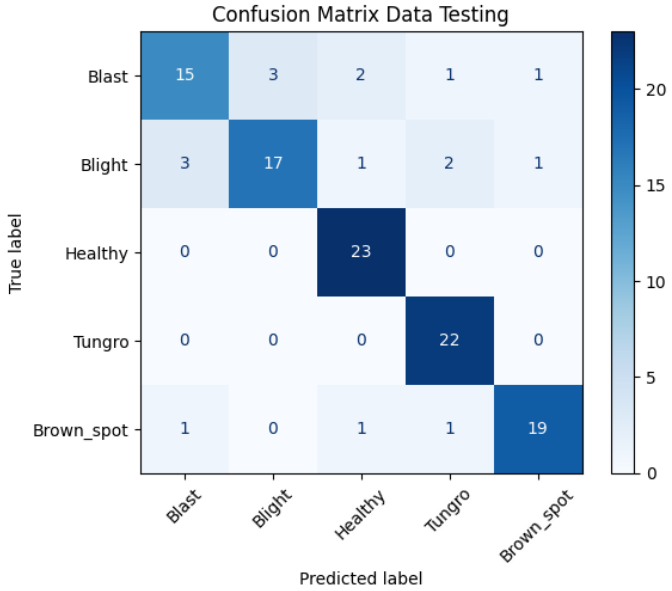
Tabel 4. 24 Nilai *Precision*, *Recall*, *F1-Score* Skenario 14 (Add-Y-a-n)

Kelas Penyakit	***Precision***	***Recall***	***F1-Score***
Blast	94%	68%	79%
Blight	84%	88%	86%
Healthy	96%	96%	96%
Tungro	85%	100%	92%
Brown Spot	83%	86%	84%
Rata-Rata	88%	88%	87%

Hasil dari *testing* menggunakan *confusion matrix* pada skenario Add-Y-a-n mendapatkan nilai akurasi sebesar 88%.

4.2.15 Skenario Add-Y-b-m

Hasil *testing* menggunakan skenario ini mendapatkan nilai yang baik, karena model dapat memprediksi penyakit setiap kelas secara benar dengan jumlah yang banyak, selain itu perbedaan hasil akurasi *testing* dan *training* tidak jauh berbeda. Gambar 4. 20 menunjukkan hasil evaluasi menggunakan *confusion matrix*.



Gambar 4. 20 *Confusion Matrix* Skenario 15 (Add-Y-b-m)

Berdasarkan *confusion matrix* pada Gambar 4. 20, didapatkan *classification report* berupa *precision*, *recall*, dan *F1-Score* pada Tabel 4. 25.

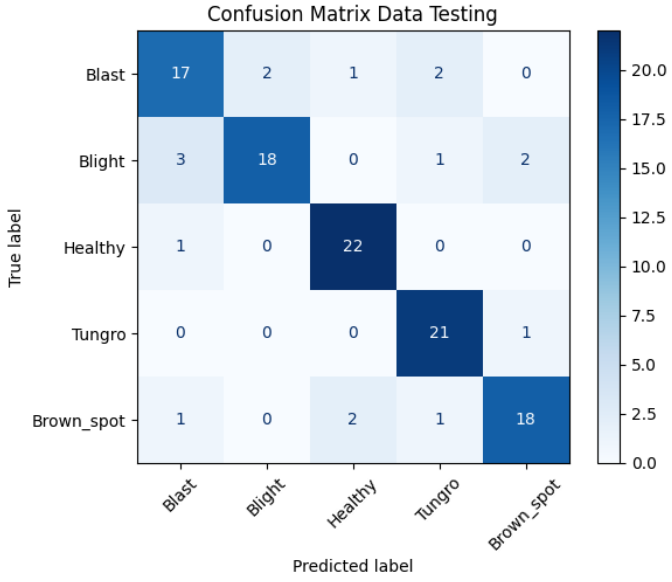
Tabel 4. 25 Nilai *Precision*, *Recall*, *F1-Score* Skenario 15 (Add-Y-b-m)

Kelas Penyakit	***Precision***	***Recall***	***F1-Score***
Blast	79%	68%	73%
Blight	85%	71%	77%
Healthy	85%	100%	92%
Tungro	85%	100%	92%
Brown Spot	90%	86%	88%
Rata-Rata	85%	85%	84%

Hasil dari *testing* menggunakan *confusion matrix* pada skenario Add-Y-b-m mendapatkan nilai akurasi sebesar 85%.

4.2.16 Skenario Add-Y-b-n

Hasil *testing* menggunakan skenario ini mendapatkan nilai yang baik, karena model dapat memprediksi penyakit setiap kelas secara benar dengan jumlah yang banyak, selain itu perbedaan hasil akurasi *testing* dan *training* tidak jauh berbeda. Gambar 4. 21 menunjukkan hasil evaluasi menggunakan *confusion matrix*.



Gambar 4. 21 *Confusion Matrix* Skenario 16 (Add-Y-b-n)

Berdasarkan *confusion matrix* pada Gambar 4. 21, didapatkan *classification report* berupa *precision*, *recall*, dan *F1-Score* pada Tabel 4. 26.

Tabel 4. 26 Nilai *Precision, Recall, F1-Score* Skenario 16 (Add-Y-b-n)

Kelas Penyakit	***Precision***	***Recall***	***F1-Score***
Blast	77%	77%	77%
Blight	90%	75%	82%
Healthy	88%	96%	92%
Tungro	84%	95%	89%
Brown Spot	86%	82%	84%
Rata-Rata	85%	85%	85%

Hasil dari *testing* menggunakan *confusion matrix* pada skenario Add-Y-b-n mendapatkan nilai akurasi sebesar 85%.

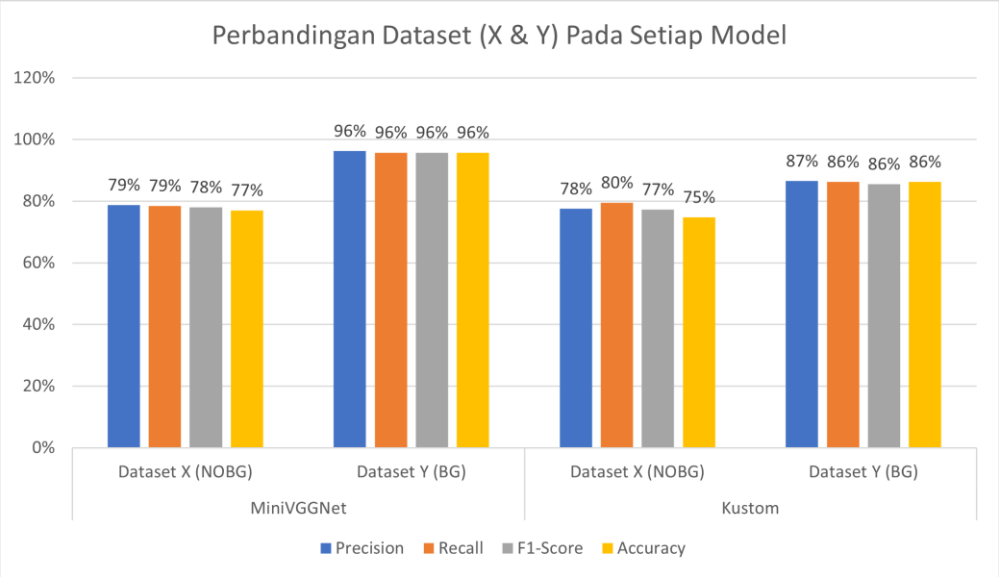
4.3 Pembahasan

Berdasarkan hasil uji coba yang telah dilakukan, bahwa penggunaan dataset sangat berpengaruh pada hasil akurasi yang didapatkan pada setiap skenario. Dari hasil *training* sebelumnya, grafik menunjukkan bahwa dataset Y (*background*) pada model MiniVGGNet dan kustom MiniVGGNet berhasil mencapai konvergen lebih cepat dan stabil dibandingkan dengan dataset X (*no-background*). Perbedaan dataset tersebut dibuktikan dengan melakukan *testing*, yang menunjukkan bahwa kedua model yang menggunakan dataset Y memiliki tingkat akurasi yang lebih tinggi. Model MiniVGGNet mendapatkan rata-rata akurasi 96% dan model kustom MiniVGGNet 86%. Selain nilai akurasi, dataset Y juga unggul pada nilai evaluasi seperti *precision*, *recall*, dan *F1-Score*. Tabel 4. 27 menampilkan rata-rata nilai evaluasi pada setiap model berdasarkan dataset yang digunakan.

Tabel 4. 27 Perbandingan Rata-rata Nilai Evaluasi Dataset

Model	**Skenario**	***Precision***	***Recall***	***F1-Score***	***Accuracy***
MiniVGGNet	Dataset X (NOBG)	79%	79%	78%	77%
	Dataset Y (BG)	96%	96%	96%	96%
Kustom	Dataset X (NOBG)	78%	80%	77%	75%
	Dataset Y (BG)	87%	86%	86%	86%

Berdasarkan Tabel 4. 27, bahwa model yang menggunakan dataset Y memiliki nilai evaluasi yang tinggi pada semua aspek dibandingkan dataset X. Model yang menggunakan dataset X memiliki rata-rata nilai evaluasi disekitaran 70%. Sedangkan model yang menggunakan dataset Y memiliki rata-rata nilai evaluasi berkisar antara 86% hingga 96%. Perbedaan nilai akurasi pada kedua model mencapai 10% hingga 20%. Untuk mempermudah dalam melihat perbedaan nilai evaluasi berdasarkan dataset, Gambar 4. 22 menampilkan dalam bentuk diagram batang.



Gambar 4. 22 Rata-rata Nilai Evaluasi Dataset

Perbedaan nilai evaluasi yang didapatkan pada kedua dataset cukup signifikan, namun model yang digunakan juga memiliki pengaruh pada peningkatan akurasi maupun nilai evaluasi lainnya seperti *precision*, *recall*, dan *F1-Score*. Analisis yang dilakukan pada dataset X dan dataset Y menunjukkan bahwa model MiniVGGNet memiliki rata-rata nilai evaluasi yang lebih unggul dibandingkan model kustom. Hal ini terjadi karena jumlah parameter pada kedua model tersebut sangat jauh berbeda. Model MiniVGGNet memiliki jumlah parameter sebanyak 2 jutaan parameter, sedangkan model kustom hanya 200 ribuan parameter, perbedaan mencapai 89%. Perbedaan jumlah parameter berdampak pada model untuk mengamati detail rumit atau *noise* pada dataset. Berikut adalah perbedaan jumlah parameter dan ukuran model yang ditampilkan pada Tabel 4. 28.

Tabel 4. 28 Perbandingan Jumlah Parameter dan Ukuran Model

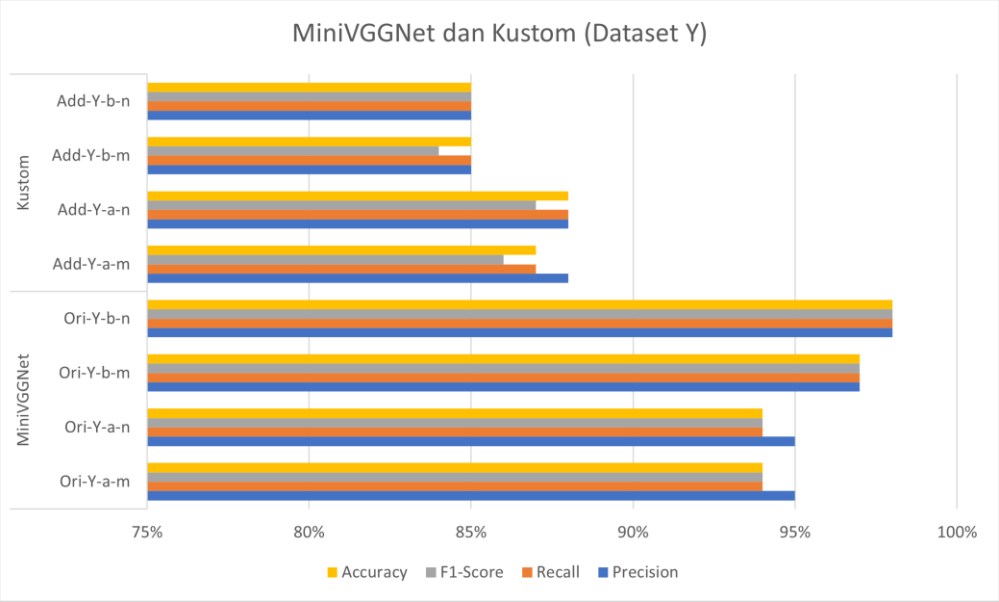
Model	Parameter	Ukuran (MB)
MiniVGGNet	2.167.797	8,26
Kustom MiniVGGNet	242.981	0,95

Pada analisis sebelumnya, bahwa model yang menggunakan dataset Y memiliki nilai evaluasi yang lebih tinggi dibandingkan dengan model yang tidak menggunakan dataset tersebut. Maka dari itu, dilakukan perbandingan nilai evaluasi pada setiap model dengan menggunakan dataset Y. Berikut adalah hasil yang didapatkan setiap skenario dengan menggunakan dataset Y pada Tabel 4. 29

Tabel 4. 29 Rata-rata Nilai Evaluasi Skenario Menggunakan Dataset Y

Model	**Skenario**	***Precision***	***Recall***	***F1-Score***	***Accuracy***
MiniVGGNet	Ori-Y-a-m	95%	94%	94%	94%
	Ori-Y-a-n	95%	94%	94%	94%
	Ori-Y-b-m	97%	97%	97%	97%
	Ori-Y-b-n	98%	98%	98%	98%
Kustom	Add-Y-a-m	88%	87%	86%	87%
	Add-Y-a-n	88%	88%	87%	88%
	Add-Y-b-m	85%	85%	84%	85%
	Add-Y-b-n	85%	85%	85%	85%

Pada Tabel 4. 29 menunjukkan bahwa model MiniVGGNet dan model kustom memiliki nilai akurasi yang berbeda, namun keduanya tetap memiliki nilai yang tinggi. Untuk melihat perbedaan secara grafik dapat dilihat pada Gambar 4. 23 yang menampilkan diagram batang.

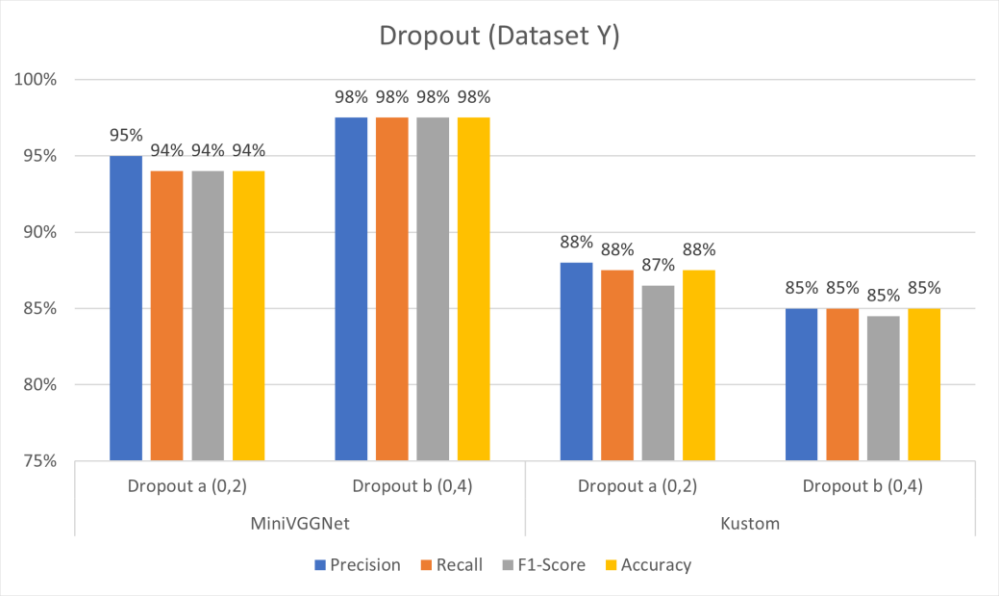


Gambar 4. 23 Rata-rata Nilai Evaluasi Skenario Pada Dataset Y

Pada Gambar 4. 23, setiap model MiniVGGNet dan model kustom memiliki nilai skenario yang terbaik, yang masing-masing memiliki nilai *dropout* yang berbeda. *Dropout* 0,4 memiliki nilai evaluasi yang lebih unggul pada model MiniVGGNet dan *dropout* 0,2 lebih unggul pada model kustom. Nilai *dropout* yang besar lebih dibutuhkan pada model yang mempunyai parameter yang banyak seperti model MiniVGGNet, sebaliknya, model kustom menggunakan nilai *dropout* yang lebih kecil karena jumlah parameternya lebih sedikit. Hal ini menunjukkan bahwa model dengan kompleksitas yang tinggi memerlukan regularisasi yang lebih besar untuk mencegah *overfitting* dan meningkatkan nilai evaluasi. Hasil perbedaan nilai *dropout* dapat dilihat pada Tabel 4. 30 dan Gambar 4. 24.

Tabel 4. 30 Rata-rata Nilai Evaluasi *Dropout* Menggunakan Dataset Y

Model	**Skenario**	***Precision***	***Recall***	***F1-Score***	***Accuracy***
MiniVGGNet	*Dropout* a (0,2)	95%	94%	94%	94%
	Dropout b (0,4)	98%	98%	98%	98%
Kustom	*Dropout* a (0,2)	88%	88%	87%	88%
	Dropout b (0,4)	85%	85%	85%	85%



Gambar 4. 24 Rata-rata Nilai Evaluasi *Dropout* Pada Dataset Y

Berdasarkan analisis yang telah dilakukan, bahwa model yang menggunakan dataset Y memiliki nilai evaluasi yang lebih tinggi dibandingkan dengan dataset X, karena pada dataset Y memiliki lebih banyak fitur yang dapat diekstrak. Selain itu, nilai *hyperparamer* yang digunakan juga memiliki pengaruh pada nilai evaluasi, khusunya penggunaan nilai *dropout*.

Pada model MiniVGGNet, skenario 8 (Ori-Y-b-n) adalah skenario yang menggunakan dataset Y, *dropout* 0,4 dan *batch size* 64, yang memiliki nilai akurasi 98%, tertinggi daripada semua skenario. Berbeda dengan model kustom MiniVGGNet, skenario 14 (Add-Y-a-n) adalah skenario yang menggunakan dataset Y, *dropout* 0,4 dan *batch size* 64, yang memiliki nilai akurasi 88%, tertinggi daripada skenario lain dengan menggunakan model yang sama.

Pada analisis yang telah dilakukan, setiap model mendapatkan skenario terbaiknya. Model MiniVGGNet mendapatkan nilai evaluasi yang terbaik pada skenario 8 (Ori-Y-b-n) dan model kustom MiniVGGNet pada skenario 14 (Add-Y-a-n). Dari kedua skenario terbaik tersebut, kemudian dilakukan tahapan *K-Fold cross validation*. Tahapan ini bertujuan untuk membuktikan skenario terbaik yang didapatkan bukan karena kebetulan mendapatkan data *training* atau *testing* yang bagus. Berikut adalah hasil yang didapatkan dari skenario 8 (Ori-Y-b-n) dengan 5 lipatan (fold) yang ditampilkan pada Tabel 4. 31.

Tabel 4. 31 Hasil K-Fold Skenario 8 (Ori-Y-b-n)

Fold	***Val loss***	***Val Acc***
1	0,2750	95,36%
2	0,3102	95,81%
3	0,3851	94,45%
4	0,2994	92,19%
5	0,3180	95,70%

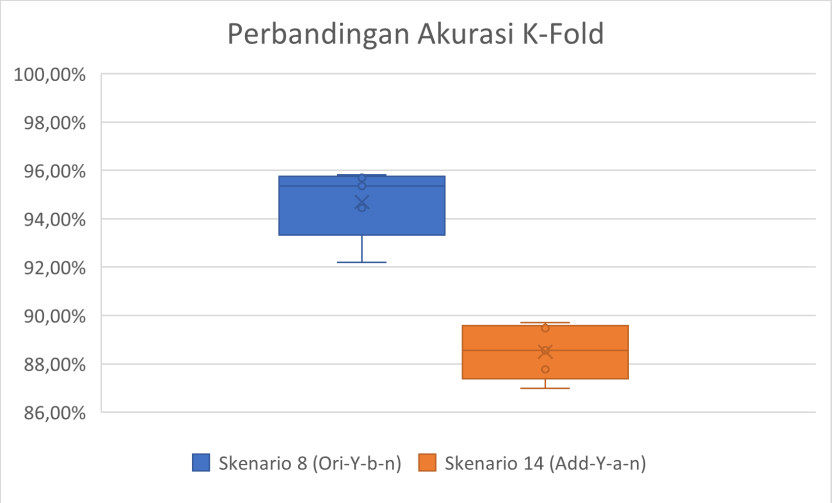
Berdasarkan Tabel 4. 31, hasil yang didapatkan dengan pengujian *K-Fold cross validation* pada skenario 8 (Ori-Y-b-n), model menunjukkan performa yang stabil pada setiap lipatan (*fold*). Rata-rata akurasi yang didapatkan adalah sebesar 94,07%, dengan rentan akurasi yang didapatkan antara *fold* adalah 92,19% hingga 95,81%. Nilai akurasi tersebut tidak jauh berbeda dari hasil *testing* sebesar 98%. Kemiripan nilai akurasi yang didapatkan dari rata-rata *K-Fold* dan nilai akurasi *testing*, menunjukkan bahwa model mampu menghasilkan performa yang tinggi secara konsisten, walaupun dengan menggunakan data *training* dan *testing* yang berbeda.

Pada skenario 14 (Add-Y-a-n) dilakukan *K-Fold cross validation* dengan 5 *fold*, berikut adalah hasil yang ditampilkan pada Tabel 4. 32.

Tabel 4. 32 Hasil K-Fold Skenario 14 (Add-Y-a-n)

Fold	***Val loss***	***Val Acc***
1	0,1707	89,71%
2	0,1414	87,77%
3	0,1676	86,98%
4	0,3068	89,47%
5	0,1684	88,56%

Berdasarkan Tabel 4. 32, hasil yang didapatkan dengan pengujian *K-Fold cross validation* pada skenario 14 (Add-Y-a-n), model menunjukkan performa yang stabil pada setiap *fold*. Rata-rata akurasi yang didapatkan adalah sebesar 88,50%, dengan rentan akurasi yang didapatkan antara *fold* adalah 86,98% hingga 89,71%. Nilai akurasi tersebut tidak jauh berbeda dari hasil *testing* sebesar 88%. Kemiripan nilai akurasi yang didapatkan dari rata-rata *K-Fold* dan nilai akurasi *testing*, menunjukkan bahwa model mampu menghasilkan performa yang tinggi secara konsisten, walaupun dengan menggunakan data *training* dan *testing* yang berbeda.



Gambar 4. 25 *Box Plot* Skenario 8 (Ori-Y-b-n) dan Skenario 14 (Add-Y-a-n)

Berdasarkan Gambar 4. 25, menunjukkan bahwa distribusi akurasi skenario 8 lebih tinggi dibandingkan dengan skenario 14. Skenario 8 memiliki akurasi minimum sebesar 92,19% dan maksimum mencapai 95,81%, dengan mayoritas data (*interquartil range*) berkumpul di atas 94%. Sementara itu, skenario 14 memiliki rentang yang lebih rendah antara 86,98% sampai 89,71%. Hal ini menunjukkan bahwa skenario 8 memiliki performa yang lebih unggul dan konsisten pada setiap *fold* pengujian.

Dari hasil *box plot* tersebut, dilakukan uji signifikansi menggunakan *paired t-test* dengan taraf signifikansi sebesar 0,05. Berdasarkan hasil perhitungan, diperoleh nilai *p-value* sebesar 0,0029. Hasil ini menunjukkan bahwa terdapat perbedaan performa yang signifikan secara statistik antara skenario 8 dan skenario 14.

Dalam penelitian ini, penerapan metode *Convolutional Neural Network* (CNN) untuk mengidentifikasi jenis penyakit pada tanaman padi diintegrasikan dengan nilai-nilai Islam. Integrasi tersebut mencakup tiga aspek utama, yaitu

hubungan manusia dengan alam (*muamalah ma'a al-'alam*), hubungan manusia dengan sesama manusia (*muamalah ma'a an-nas*), dan hubungan manusia dengan Allah (*muamalah ma'a Allah*). Pendekatan ini dimaksudkan untuk memadukan ajaran Islam dengan perkembangan ilmu pengetahuan pada bidang *machine learning*, sehingga sistem identifikasi yang dikembangkan tidak hanya berfungsi secara teknis, tetapi juga memiliki landasan etis dan spiritual. Dengan demikian, penelitian ini diharapkan mampu memberikan manfaat yang lebih luas, tidak hanya dalam peningkatan produktivitas pertanian, tetapi juga dalam menumbuhkan kesadaran bahwa teknologi harus dibangun dan diterapkan dengan tanggung jawab moral serta nilai-nilai keislaman. Integrasi ini juga menjadi wujud bahwa ilmu pengetahuan dan iman dapat berjalan selaras untuk menghadirkan solusi yang membawa kemaslahatan bagi lingkungan, masyarakat, dan sebagai bentuk pengabdian kepada Allah SWT.

1. Muamalah Ma'a Al-Alam

Penelitian ini dilakukan untuk mengidentifikasikan jenis penyakit pada tanaman padi menggunakan model CNN yang berbasis Android. Tujuannya untuk mempermudah petani dalam mengenali jenis penyakit yang menyerang tanaman padi mereka. Dengan menggunakan model CNN, petani dapat merawat tanaman padi mereka agar tetap sehat dan subur, yang akan menghasilkan padi yang berkualitas. Hal ini sesuai dengan ajaran agama Islam yang menyerukan umat Islam untuk selalu menjaga dan merawat tanaman mereka. Allah SWT berfirman,

وَلَا تُفْسِدُوا۟ فِى ٱلْأَرْضِ بَعْدَ إِصْلَٰحِهَا وَٱدْعُوهُ خَوْفًا وَطَمَعًا ۚ إِنَّ رَحْمَتَ ٱللَّهِ قَرِيبٌ مِّنَ ٱلْمُحْسِنِينَ ﴿٥٦﴾

"Janganlah kamu berbuat kerusakan di bumi setelah diatur dengan baik. Berdoalah kepada-Nya dengan rasa takut dan penuh harap. Sesungguhnya rahmat Allah sangat dekat dengan orang-orang yang berbuat baik" (Q.S. Al-A'raf: 56).

Berdasarkan tafsir Ibnu Katsri, Allah melarang perbuatan yang menimbulkan kerusakan di muka bumi dan hal-hal yang membahayakan kelestariannya sesudah diperbaiki. Karena sesungguhnya apabila segala sesuatunya berjalan sesuai dengan kelestariannya, kemudian terjadilah pengrusakan padanya, hal tersebut akan membahayakan semua hamba Allah. Maka Allah melarang hal tersebut, dan memerintahkan kepada mereka untuk menyembah-Nya dan berdoa kepada-Nya serta berendah diri dan memohon belas kasihan-Nya. Yakni dengan perasaan takut terhadap azab yang ada di sisi-Nya dan penuh harap kepada pahala berlimpah yang ada di sisi-Nya. Rahmat Allah selalu berada sangat dekat kepada orang-orang yang berbuat kebaikan, yaitu mereka yang mengikuti perintah-perintah-Nya dan menjauhi larangan-larangan-Nya.

2. Muamalah Ma'a An-Nas

Integrasi ini menekankan pentingnya hubungan antar sesama manusia dalam melakukan kebaikan yang membawa manfaat bagi masyarakat. Penerapan metode *Convolutional Neural Network* (CNN) berbasis Android dirancang untuk membantu masyarakat, khususnya para petani, dalam memantau kesehatan tanaman padi sehingga dapat meningkatkan kualitas dan kuantitas hasil panen. Upaya ini sejalan dengan ajaran Islam yang mendorong umatnya untuk saling membantu dalam kebaikan, karena sebaik-baik manusia adalah yang memberikan manfaat bagi orang lain. Dengan demikian, pengembangan sistem ini bukan hanya memenuhi kebutuhan teknologi pertanian, tetapi juga merefleksikan nilai-nilai

sosial dan kemanusiaan dalam Islam. Dari Jabir bin ‘Abdillah *radhiyallahu ‘anhu*, Rasulullah *shallallahu ‘alahi wa sallam* bersabda,

المؤْمِنُ يَأْلَفُ وَيُؤْلَفُ وَلاَ خَيْرَ فِيْمَنْ لاَ يَأْلَفُ وَلاَ يُؤْلَفُ وَخَيْرُ النَّاسِ أَنْفَعُهُمْ لِلنَّاسِ

”Seorang mukmin itu adalah orang yang bisa menerima dan diterima orang lain, dan tidak ada kebaikan bagi orang yang tidak bisa menerima dan tidak bisa diterima orang lain. Dan sebaik-baik manusia adalah yang paling bermanfaat bagi manusia lainnya” (HR. Thabrani dalam Al-Mu’jam Al-Awsath, no. 5949. Syaikh Al-Albani menghasankan dalam Silsilah Al-Ahadits Ash-Shahihah, no. 426).

Dalam hadis tersebut, Rasulullah *shallallahu ‘alahi wa sallam* menekankan bahwa orang yang baik adalah orang yang bermanfaat antara sesama manusia. Dalam penelitian ini, salah satu manfaat yang dapat diberikan adalah menerapkan metode CNN untuk meningkatkan hasil dan kualitas panen padi. Hadis tersebut menjelaskan tentang amalan *muta’addi*. Amalan tersebut adalah amalan yang manfaatnya untuk orang lain, seperti manfaat *ukhrawi* (seperti mengajarkan ilmu dan dakwah ilallah), bisa juga manfaat *duniawi* (seperti menunaikan hajat orang lain, menolong orang yang dizalimi).

3. Muamalah Ma’a Allah

Muamalah ma’a Allah menekankan bahwa segala usaha ilmiah adalah bentuk ibadah selama diniatkan untuk kebaikan. Pengembangan penelitian ini dilakukan sebagai upaya menunaikan amanah ilmu, memanfaatkan akal, dan berkontribusi pada kemaslahatan umat. Allah SWT berfirman,

وَمَا خَلَقْتُ ٱلْجِنَّ وَٱلْإِنسَ إِلَّا لِيَعْبُدُونِ ﴿٥٦﴾

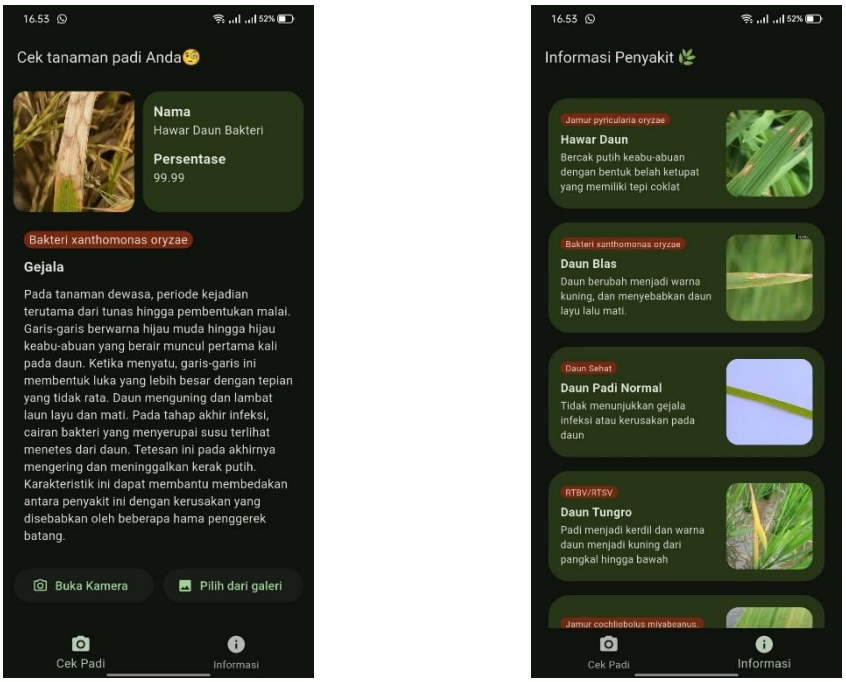
”Dan aku tidak menciptakan jin dan manusia melainkan supaya mereka beribadah kepada-Ku” (Q.S. Adz-Dzariyat: 56).

Menurut tafsir Kementrian Agama Republik Indonesia, Ayat ini menegaskan bahwa Allah tidaklah menjadikan jin dan manusia melainkan untuk mengenal-Nya dan supaya menyembah-Nya. Pendapat tersebut sama dengan pendapat az-Zajjaj, tetapi ahli tafsir yang lain berpendapat bahwa maksud ayat tersebut ialah bahwa Allah tidak menjadikan jin dan manusia kecuali untuk tunduk kepada-Nya dan untuk merendahkan diri. Maka setiap makhluk, baik jin atau manusia wajib tunduk kepada peraturan Tuhan, merendahkan diri terhadap kehendak-Nya. Menerima apa yang Dia takdirkan, mereka dijadikan atas kehendak-Nya dan diberi rezeki sesuai dengan apa yang telah Dia tentukan. Tak seorang pun yang dapat memberikan manfaat atau mendatangkan mudarat karena kesemuanya adalah dengan kehendak Allah. Ayat tersebut menguatkan perintah mengingat Allah dan memerintahkan manusia supaya melakukan ibadah kepada Allah.

4.4 *Deployment*

Berdasarkan hasil evaluasi pengujian, skenario 14 (Add-Y-a-n) mendapatkan ukuran yang jauh lebih ringan dibandingkan dengan skenario 8 (Ori-Y-b-n), namun akurasi yang didapatkan skenario 14 memiliki akurasi yang lebih rendah dibandingkan skenario 8. Walaupun memiliki akurasi yang lebih rendah, skenario 14 memiliki keunggulan pada ukuran *file* yang jauh lebih ringan. Sehingga, untuk efisiensi dalam penyimpanan perangkat *mobile*, maka model yang diterapkan pada perangkat *mobile* adalah skenario 14. Skenario tersebut awalnya disimpan dalam format .h5. Agar dapat dioperasikan pada perangkat *mobile*, format tersebut dikonversi menjadi .tflite (TensorFlow Lite) yang kompatibel dengan arsitektur aplikasi Android. Setelah dikonversi ke dalam format .tflite, model tersebut

diintegrasikan ke dalam lingkungan pengembangan aplikasi Android. Aplikasi ini telah dirancang dengan antarmuka pengguna (*User Interface*) untuk memfasilitasi interaksi sistem. Implementasi tampilan antarmuka pada aplikasi Android tersebut dapat dilihat pada Gambar 4. 26.



Gambar 4. 26 *User Interface* (UI) Aplikasi

Aplikasi Android yang telah berisikan model CNN perlu dilakukan pengujian *usability*. Pengujian dilakukan untuk mengetahui apakah aplikasi yang dibuat telah berjalan dengan lancar dan tidak memiliki hambatan, serta sesuai dengan kebutuhan pengguna. Pengujian *usability* dilakukan dengan menggunakan *System Usability Scale* (SUS). Pengujian tersebut melibatkan beberapa responden yang berasal dari Mahasiswa UIN Malang untuk mengetahui aplikasi berjalan dengan baik dan mendapatkan nilai skor yang tinggi atau rendah. Skor yang didapatkan dari responden sudah dilakukan perhitungan skor SUS dengan skala 0 hingga 4. Nilai

skor dijumlahkan lalu dikalikan 2,5. Berikut adalah hasil yang didapatkan dari responden berdasarkan perhitungan nilai skor yang ditampilkan pada Tabel 4. 33.

Tabel 4. 33 Rata-rata Skor SUS Responden

Responden	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**	**Q6**	**Q7**	**Q8**	**Q9**	**Q10**	**Skor SUS**
1	3	4	4	3	3	3	4	4	4	3	87,5
2	3	4	4	3	3	4	4	4	4	4	92,5
3	4	3	3	1	3	3	3	3	3	3	72,5
4	2	4	3	2	4	3	3	3	3	1	70
5	2	3	3	3	3	3	3	3	4	3	75
6	2	3	4	3	3	1	4	4	4	4	80
7	2	4	4	4	4	1	4	4	3	4	85
8	4	2	3	3	3	3	3	4	4	0	72,5
9	3	4	3	3	3	0	4	3	3	4	75
10	4	4	4	3	4	3	4	4	4	3	92,5
Rata-rata											**80,25**

Berdasarkan Tabel 4. 33, skor rata-rata dengan menggunakan pengujian SUS adalah sebesar 80,25. Dari skor tersebut, maka didapatkan *adjective rating* "*Excellent*" dan *grade letter* "B". Angka ini mengindikasikan bahwa aplikasi Android yang telah dikembangkan memiliki tingkat *usability* yang baik dan mampu memenuhi kebutuhan pengguna dalam konteks mengidentifikasi jenis penyakit tanaman daun padi.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan terhadap 16 skenario pengujian, bahwa terdapat dua skenario yang terbaik pada masing-masing model CNN. Skenario 8 (Ori-Y-b-n) yang dilatih dengan menggunakan model MiniVGGNet dan dataset Y (*background*) dengan konfigurasi *dropout* 0,4 dan *batch size* 64 mendapatkan nilai *precision* 98%, *recall* 98%, *F1-Score* 98%, dan nilai akurasi 98%. Selain itu, Skenario 14 (Add-Y-a-n) yang dilatih menggunakan model kustom MiniVGGNet dan dataset Y dengan konfigurasi *dropout* 0,4 dan *batch size* 64 mendapatkan nilai *precision* 88%, *recall* 88%, *F1-Score* 87%, dan nilai akurasi 88%.

Meskipun skenario 14 memiliki akurasi yang lebih rendah dibandingkan dengan skenario 8, model ini memiliki keunggulan berupa ukuran *file* yang jauh lebih kecil dan efisien. Maka dari itu, dengan mempertimbangkan performa dan efisiensi sumber daya penyimpanan pada perangkat *mobile*, skenario 14 (Add-Y-a-n) dipilih untuk diimplementasikan ke dalam aplikasi *mobile*. Aplikasi yang ditanamkan model tersebut kemudian dilakukan pengujian *usability*. Pengujian menggunakan *System Usability Scale* (SUS). Rata-rata skor SUS yang didapatkan sebesar 80,25, *adjective rating* "*Excellent*" dan *grade letter* "B". Dari hasil tersebut menunjukkan bahwa aplikasi yang dikembangkan dapat diterima dengan baik dan mudah oleh pengguna.

5.2 Saran

Berdasarkan pengujian dan evaluasi yang telah dilakukan, penulis menyadari masih terdapat beberapa kekurangan dari penelitian ini. Maka dari itu, penulis memberikan saran kepada peneliti selanjutnya untuk penyempurnaan sistem, antara lain:

1. Menambahkan jenis penyakit pada tanaman padi, sehingga model lebih mampu untuk mendeteksi jenis penyakit yang lebih beragam.
2. Menambahkan jumlah *epoch* pada proses *training* atau menambahkan lapisan konvolusi pada model kustom MiniVGGNet agar nilai *loss* dan akurasi mencapai konvergen.
3. Menambahkan beberapa fitur pada aplikasi *mobile* untuk meningkatkan tingkat fungsionalitas, seperti fitur riwayat deteksi (*history*) dan kemampuan mendeteksi secara langsung (*real-time detection*) menggunakan kamera.

DAFTAR PUSTAKA

Abasi, A. K., Makhadmeh, S. N., Alomari, O. A., Tubishat, M., & Mohammed, H. J. (2023). Enhancing Rice Leaf Disease Classification: A Customized Convolutional Neural Network Approach. *Sustainability*, *15*(20), 15039. https://doi.org/10.3390/su152015039

Alden, S., & Sari, B. N. (2023). Implementasi Algoritma CNN Untuk Pemilahan Jenis Sampah Berbasis Android Dengan Metode CRISP-DM. *Jurnal Informatika*, *10*(1), 62–71. https://doi.org/10.31294/inf.v10i1.14985

Anggiratih, E., Siswanti, S., Octaviani, S. K., & Sari, A. (2021). Klasifikasi Penyakit Tanaman Padi Menggunakan Model Deep Learning Efficientnet B3 dengan Transfer Learning. *Jurnal Ilmiah SINUS*, *19*(1), 75. https://doi.org/10.30646/sinus.v19i1.526

Anshul. (2024). *Rice Disease Dataset* [Dataset]. https://www.kaggle.com/datasets/anshulm257/rice-disease-dataset

Ardhana, V. Y. P. (2021). Pengujian Usability Aplikasi Halodoc Menggunakan Metode System Usability Scale (SUS). *Jurnal Kesehatan Qamarul Huda*, *9*(2), 132–136. https://doi.org/10.37824/jkqh.v9i2.2021.311

Bangor, A. (2009). *Determining What Individual SUS Scores Mean: Adding an Adjective Rating Scale*. *4*(3).

Brooke, J. (1995). SUS: A quick and dirty usability scale. *Usability Eval. Ind.*, *189*.

FAO. (2021). *The State of Food and Agriculture 2021*. FAO. https://doi.org/10.4060/cb4476en

Fiddin, A., Sutrawati, M., Bustamam, H., Ganefianti, D. W., & Sipriyadi, S. (2021). PENYAKIT TUNGRO PADA TANAMAN PADI (Oryza sativa) DI KECAMATAN TABA PENANJUNG: INSIDENSI PENYAKIT DAN DETEKSI VIRUS SECARA MOLEKULER. *Jurnal Ilmu-Ilmu Pertanian Indonesia*, *23*(1), 37–45. https://doi.org/10.31186/jipi.23.1.37-45

Food Outlook – Biannual report on global food markets. (2025). FAO. https://doi.org/10.4060/cd5655en

Hairani, H., & Widiyaningtyas, T. (2024). Augmented Rice Plant Disease Detection with Convolutional Neural Networks. *INTENSIF: Jurnal Ilmiah Penelitian Dan Penerapan Teknologi Sistem Informasi*, *8*(1), 27–39. https://doi.org/10.29407/intensif.v8i1.21168

Hawibowo, M. S., & Muhimmmah, I. (2024). Aplikasi Pendeteksi Tingkat Kematangan Pepaya menggunakan Metode Convolutional Neural Network (CNN) Berbasis Android. *Jurnal Edukasi Dan Penelitian Informatika (JEPIN)*, *10*(1), 162. https://doi.org/10.26418/jp.v10i1.77819

Hazmi, M. A. A., Azizah, F. H. N., Hajar, S., Ahmad, H. J., & Abrar, M. R. A. (2024). Kerusakan Alam dan Mitigasi Krisis Lingkungan (Kajian Surat Al-Baqarah Ayat 205-207 dalam Tafsir Al-Maraghi). *Ulumul Qur'an: Jurnal Kajian Ilmu Al-Qur'an Dan Tafsir*, *4*(1), 75–92.

Ilham Rahmana Syihad, Muhammad Rizal, Zamah Sari, & Yufis Azhar. (2023). CNN Method to Identify the Banana Plant Diseases based on Banana Leaf Images by Giving Models of ResNet50 and VGG-19. *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi)*, *7*(6), 1309–1318. https://doi.org/10.29207/resti.v7i6.5000

Kingma, D. P., & Ba, J. (2017). *Adam: A Method for Stochastic Optimization* (arXiv:1412.6980). arXiv. https://doi.org/10.48550/arXiv.1412.6980

Kotta, C. R., Paseru, D., & Sumampouw, M. (2022). Implementasi Metode Convolutional Neural Network untuk Mendeteksi Penyakit Pada Citra Daun Tomat. *Jurnal Pekommas*, *7*(2). https://doi.org/10.56873/jpkm.v7i2.4961

Latifah, H., Tyas, D. A., & Harjoko, A. (2024). Peningkatan Performa Klasifikasi Sel Darah Merah pada Pasien Talasemia Minor. *IJEIS (Indonesian Journal of Electronics and Instrumentation Systems)*, *14*(2). https://doi.org/10.22146/ijeis.93025

Li, Z., Yang, W., Peng, S., & Liu, F. (2020). *A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects* (arXiv:2004.02806). arXiv. https://doi.org/10.48550/arXiv.2004.02806

Marwan, H., Nusifera, S., & Mulyati, S. (2021). Potensi Bakteri Endofit sebagai Agens Hayati untuk Mengendalikan Penyakit Blas pada Tanaman Padi. *Jurnal Ilmu Pertanian Indonesia*, *26*(3), 328–333. https://doi.org/10.18343/jipi.26.3.328

Matlab. (2021). *What Is a Convolutional Neural Network?* https://www.mathworks.com/discovery/convolutional-neural-network.html

M.B. Gigih Baskoro Ashari. (2024). Implementasi Algoritma Convolutional Neural Network untuk Meningkatkan Identifikasi Penyakit Tanaman Durian. *Jupiter: Publikasi Ilmu Keteknikan Industri, Teknik Elektro Dan Informatika*, *2*(4), 162–172. https://doi.org/10.61132/jupiter.v2i4.418

Nurilmiyanti Wardhani, Asrul, B. E. W., Antonius Riman Tampang, Sitti Zuhriyah, & Abdul Latief Arda. (2024). Classification of Toraja Wood Carving Motif Images Using Convolutional Neural Network (CNN). *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi)*, *8*(4), 486–495. https://doi.org/10.29207/resti.v8i4.5897

Nurkayah, N., Sidiq, A. N., & Murdini, L. A. (2024). Identifikasi Prevalensi dan Karakterisasi Penyakit Tanaman Padi (Oryza Sativa L) di Kecamatan Sumber Harta, Kabupaten Musi Rawas. *Indonesian Research Journal on Education*, *4*(3), Article 3. https://doi.org/10.31004/irje.v4i3.847

O'Shea, K., & Nash, R. (2015). *An Introduction to Convolutional Neural Networks* (arXiv:1511.08458). arXiv. https://doi.org/10.48550/arXiv.1511.08458

Prajapati, H. B., Shah, J. P., & Dabhi, V. K. (2017). Detection and classification of rice plant diseases. *Intelligent Decision Technologies*, *11*(3), 357–373. https://doi.org/10.3233/IDT-170301

Qian, Z., Hayes, T. L., Kafle, K., & Kanan, C. (2020). *Do We Need Fully Connected Output Layers in Convolutional Networks?* (arXiv:2004.13587). arXiv. https://doi.org/10.48550/arXiv.2004.13587

Rijal, M., Yani, A. M., & Rahman, A. (2024). Deteksi Citra Daun untuk Klasifikasi Penyakit Padi menggunakan Pendekatan Deep Learning dengan Model CNN. *Jurnal Teknologi Terpadu*, *10*(1), Article 1. https://doi.org/10.54914/jtt.v10i1.1224

Riza Agung Firmansyah, Tri Arief Sardjono, & Ronny Mardiyanto. (2023). Peningkatan Akurasi Adaptive Monte Carlo Localization Menggunakan Convolutional Neural Network. *Jurnal Nasional Teknik Elektro Dan Teknologi Informasi*, *12*(3), 167–174. https://doi.org/10.22146/jnteti.v12i3.7432

Rosebrock, A. (2021, May 22). MiniVGGNet: Going Deeper with CNNs. *PyImageSearch*. https://pyimagesearch.com/2021/05/22/minivggnet-going-deeper-with-cnns/

Sankalana, N. (2025). *Plant Leaf Diseases Training Dataset* [Dataset]. https://www.kaggle.com/datasets/nirmalsankalana/plant-diseases-training-dataset

Saragi, C. Ph., Aulia, M. R., & Manihuruk, R. A. (2023). Analisis Pendapatan Usahatani Padi Sawah di Desa Simpang Panei Raya, Kecamatan Panei, Kabupaten Simalungun. *Jurnal Agriust*, 26–31. https://doi.org/10.54367/agriust.v3i1.2580

Setiady, T. (2021). *Leaf Rice Disease* [Dataset]. https://www.kaggle.com/datasets/tedisetiady/leaf-rice-disease-indonesia

Sheila, S., Sari, I. P., Saputra, A. B., Anwar, M. K., & Pujianto, F. R. (2023). Deteksi Penyakit Pada Daun Padi Berbasis Pengolahan Citra Menggunakan Metode Convolutional Neural Network (CNN). *MULTINETICS*, *9*(1), Article 1. https://doi.org/10.32722/multinetics.v9i1.5255

Simonyan, K., & Zisserman, A. (2015). *Very Deep Convolutional Networks for Large-Scale Image Recognition* (arXiv:1409.1556). arXiv. https://doi.org/10.48550/arXiv.1409.1556

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*.

Sukarta, A. I. N., Sugiarto, Y., & Koesmaryono, Y. (2018). Projection of Rice Blast Diseases in West Java Region based on Climate Change Scenario. *Agromet*, *32*(2), 62. https://doi.org/10.29244/j.agromet.32.2.62-70

Sunandar, N., & Sutopo, J. (2024). Pemanfaatan Artificial Neural Network Terhadap Klasifikasi Penyakit Tanaman Padi Menggunakan Citra Daun. *J-SAKTI (Jurnal Sains Komputer Dan Informatika)*, *8*(1), 1–11.

Sutikno, A., Pohan, S. D., & Aljabar, A. (2024). Klasifikasi Penyakit Pada Sawi Pakcoy Dengan Menggunakan Metode Convolutional Neural Network (CNN). *Jurnal Tika*, *9*(2), Article 2. https://doi.org/10.51179/tika.v9i2.2665

vbookshelf. (2020). *Rice Leaf Diseases Dataset* [Dataset]. https://www.kaggle.com/datasets/vbookshelf/rice-leaf-diseases

Walascha, A., Febriana, A., Saputri, D., Haryanti, D. S. N., Tsania, R., Sanjaya, Y., & Priyanti. (2021). Review Artikel: Inventarisasi Jenis Penyakit yang Menyerang Daun Tanaman Padi (Oryza sativa L.). *Prosiding Seminar Nasional Biologi*, *1*(2), Article 2. https://doi.org/10.24036/prosemnasbio/vol1/150

Widia Rahma Minniarni, A., Novriadi, D., & Sepika, S. (2022). EDUKASI PEMANFAATAN SMARTPHONE SEBAGAI MEDIA PENGHASIL UANG PADA REMAJA DI DESA MUARA KALANGAN KECAMATAN ULU MUSI KABUPATEN EMPAT LAWANG. *Jurnal Ilmiah Mahasiswa Kuliah Kerja Nyata (JIMAKUKERTA)*, *2*(1), 137–142. https://doi.org/10.36085/jimakukerta.v2i1.3386

Widianto, B., Utami, E., & Ariatmanto, D. (2023). Identifikasi Penyakit Tanaman Jagung Berdasarkan Citra Daun Menggunakan Convolutional Neural

Network. *Techno.Com*, *22*(3), 599–608. https://doi.org/10.33633/tc.v22i3.8425

Winanda, T., Yuhandri, Y., & Hendrick, H. (2021). Klasifikasi Kualitas Mutu Daun Gambir Ladang Rakyat Menggunakan Metode Convolutional Neural Network. *Jurnal Sistim Informasi Dan Teknologi*, 102–107. https://doi.org/10.37034/jsisfotek.v3i3.51

Yogaswara, M. A., Sondari, N., & Ulfah, I. (2023). Uji Efikasi Berbagai Media Agens Antagonis Paenibacillus polymyxa Terhadap Penyakit Bacterial Leaf Blight Pada Tanaman Padi (Oryza sativa). *OrchidAgro*, *3*(2), 22–28. https://doi.org/10.35138/orchidagro.v3i2.598

Yuliany, S., Aradea, & Andi Nur Rachman. (2022). Implementasi Deep Learning pada Sistem Klasifikasi Hama Tanaman Padi Menggunakan Metode Convolutional Neural Network (CNN). *Jurnal Buana Informatika*, *13*(1), 54–65. https://doi.org/10.24002/jbi.v13i1.5022

Zakaria, M., Affandi, E., & Pane, D. H. (2024). Sistem Pakar Dalam Mendiagnosa Penyakit Blast Pada Tanaman Padi Dengan Metode Teorema Bayes. *Jurnal Sistem Informasi Triguna Dharma (JURSI TGD)*, *3*(5), 738–747. https://doi.org/10.53513/jursi.v3i5.8346

LAMPIRAN

Kode NN

#	Memisahkan fitur dan label pada data yang telah dipreprocessing
1	`print(f'Jumlah Data Train: {len(prepro_DataTrain)}\n')`
2	
3	`X_train = []`
4	`y_train = []`
5	
6	`for fitur, label in tqdm(prepro_DataTrain):`
7	`  img = np.array(fitur)`
8	`  img = cv2.resize(img,(32,32))`
9	`  X_train.append(img)`
10	`  y_train.append(label)`
11	
12	`X_train = np.array(X_train)`
13	`y_train = np.array(y_train)`
14	`print(X_train.shape)`
15	`print(y_train.shape)`
#	Ekstraksi fitur dari Global Average Pooling (3946, 254)
16	`layer = 'global_average_pooling2d_1'`
17	`gap = models.Model(inputs=model.input, outputs=model.get_layer(layer).output)`
18	`fitur_gap = gap.predict(X_train)`
19	
20	`# print(input)`
21	`print('Berhasil mengekstraksi fitur dari GAP\n')`
22	`print(f"Fitur terkstrak, Shape: {fitur_gap.shape}")`
#	Pembuatan Kelas Neural Network
23	`class ManualNN:`
24	`    def __init__(self, input_size, hidden_size, output_size, learning_rate=1e-4):`
25	`        self.lr = learning_rate`
26	
27	`        # Inisialisasi Bobot (He Initialization)`
28	`        self.W1 = np.random.randn(input_size, hidden_size)`
29	`        self.W2 = np.random.randn(hidden_size, output_size)`
30	
31	`        # Inisialisasi Bias`
32	`        self.b1 = np.zeros((1, hidden_size))`
33	`        self.b2 = np.zeros((1, output_size))`
34	
35	`        # Inisialisasi Adam Optimizer`
36	`        self.mW1, self.vW1 = np.zeros_like(self.W1), np.zeros_like(self.W1)`

```
        self.mW2, self.vW2 = np.zeros_like(self.W2), np.zeros_like(self.W2)
        self.mb1, self.vb1 = np.zeros_like(self.b1), np.zeros_like(self.b1)
        self.mb2, self.vb2 = np.zeros_like(self.b2), np.zeros_like(self.b2)

        # Hyperparameter Adam
        self.beta1 = 0.9
        self.beta2 = 0.999
        self.epsilon = 1e-8
        self.t = 0

    #  AKTIVASI
    def ReLU(self, x):
        return np.maximum(0, x)

    def ReLU_derivative(self, x):
        return (x > 0).astype(float)

    def softmax(self, z):
        shift = z - np.max(z, axis=1, keepdims=True)
        exp = np.exp(shift)
        sum_exp = np.sum(exp, axis=1, keepdims=True)
        return exp / sum_exp

    #  FORWARD
    def forward(self, X):
        self.input = X
        # Hidden Layer
        self.z1 = np.dot(X, self.W1) + self.b1
        self.a1 = self.ReLU(self.z1)
        # Output Layer
        self.z2 = np.dot(self.a1, self.W2) + self.b2
        self.a2 = self.softmax(self.z2)
        return self.a2

    #  BACKWARD
    def backward(self, y_true):
        m = y_true.shape[0]
        # Error Output
        dz2 = self.a2 - y_true
        # Gradient Output
        self.dW2 = (1/m) * np.dot(self.a1.T, dz2)
        self.db2 = (1/m) * np.sum(dz2, axis=0, keepdims=True)
        # Error Hidden
```

```
        dz1 = np.dot(dz2, self.W2.T) *
self.ReLU_derivative(self.z1)
        # Gradient Hidden
        self.dW1 = (1/m) * np.dot(self.input.T, dz1)
        self.db1 = (1/m) * np.sum(dz1, axis=0,
keepdims=True)

    #  UPDATE BOBOT (ADAM)
    def update_weights(self): # Saya ganti namanya jadi
update_weights biar cocok sama loop di bawah
        self.t += 1

        def apply_adam(param, grad, m, v):
            m = self.beta1 * m + (1 - self.beta1) * grad
            v = self.beta2 * v + (1 - self.beta2) * (grad
** 2)
            m_hat = m / (1 - self.beta1 ** self.t)
            v_hat = v / (1 - self.beta2 ** self.t)
            param_new = param - self.lr * m_hat /
(np.sqrt(v_hat) + self.epsilon)
            return param_new, m, v

        self.W1, self.mW1, self.vW1 = apply_adam(self.W1,
self.dW1, self.mW1, self.vW1)
        self.b1, self.mb1, self.vb1 = apply_adam(self.b1,
self.db1, self.mb1, self.vb1)
        self.W2, self.mW2, self.vW2 = apply_adam(self.W2,
self.dW2, self.mW2, self.vW2)
        self.b2, self.mb2, self.vb2 = apply_adam(self.b2,
self.db2, self.mb2, self.vb2)

# Melakukan pelatihan dengan 100 epoch
model = ManualNN(256, 512, 5, learning_rate=1e-3)

print("Mulai Training Manual...")

for i in range(100): # 100 Epoch
    # 1. Forward
    pred = model.forward(fitur_gap)

    # 2. Backward
    model.backward(y_oneHot)

    # 3. Update (Pakai Adam)
    model.update_weights()

    #  PERHITUNGAN LOSS & AKURASI
    # Hitung Loss (Categorical Cross Entropy)
```

```
    loss = -np.mean(np.sum(y_oneHot * np.log(pred + 1e-8), axis=1))

    # Hitung Akurasi
    pred_label = np.argmax(pred, axis=1)
    true_label = np.argmax(y_oneHot, axis=1)

    # Membandingkan prediksi dengan label asli (Hasilkan True/False), lalu dirata-rata
    accuracy = np.mean(pred_label == true_label)

 if i % 10 == 0:
        print(f"Epoch {i}: Loss = {loss:.4f} | Accuracy = {accuracy*100:.2f}%")

print(f"Training Selesai. Akurasi Akhir: {accuracy*100:.2f}%")

# Melakukan prediksi penyakit tanaman padi
print("Urutan Kelas:", classes)

# Load Data Test
X_sample = X_test[0:1]
y_sample = y_test[0:1]

# Ekstraksi Fitur
fitur_gap = gap.predict(X_sample)

# Prediksi
prediksi_manual = model.forward(fitur_gap)

# Ambil Index Prediksi (Angka terbesar)
hasil = np.argmax(prediksi_manual, axis=1)[0]
asli = int(y_sample[0])

# Ambil String Nama Kelas dari List
nama_prediksi = classes[hasil]
nama_asli = classes[asli]

# TAMPILKAN HASIL
print("\n--- HASIL PREDIKSI ---")
print(f"Probabilitas   : {np.max(prediksi_manual)*100:.2f}%")
print(f"Index Prediksi : {hasil} -> {nama_prediksi}")
print(f"Index Asli     : {asli} -> {nama_asli}")
print("Urutan Kelas:", classes)
```