

BAB II

TINJAUAN PUSTAKA

2.1 Penelitian Terkait

Untuk menunjukkan posisi dan kebaruan penelitian ini, disajikan ringkasan enam penelitian terdahulu yang relevan. Kajian ini disusun secara bertahap, mulai dari perbandingan akurasi arsitektur CNN, optimasi model untuk perangkat ringan, visualisasi keputusan model (interpretability), hingga implementasi pada platform digital. Tabel 2.1 memuat perbandingan setiap penelitian beserta research gap yang ada, sementara Tabel 2.2 menyajikan matriks perbandingan fitur untuk mempertegas posisi penelitian "BananaVision".

Tabel 2. 1 Perbandingan Penelitian Terdahulu

No	Peneliti (Tahun)	Fokus Penelitian	Teknologi & Metode	Keterbatasan / Research Gap
1	Syihad, dkk. (2023)[7].	Komparasi performa arsitektur CNN pada citra daun pisang.	CNN Arsitektur ResNet50 dan VGG-19 . Akurasi: ResNet50 mencapai 94% dan VGG-19 mencapai 91% .	Jumlah Kelas 4 kelas (Cordana, Sigatoka, Pestalotiopsis, dan Sehat). Riset terbatas pada uji akurasi di lingkungan pengembangan; belum diimplementasi kan ke sistem aplikasi web/mobile yang dapat diakses petani

No	Peneliti (Tahun)	Fokus Penelitian	Teknologi & Metode	Keterbatasan / Research Gap
				secara langsung.
2	Lase, dkk. (2025)[9].	Klasifikasi hibrida untuk menajamkan batas keputusan model.	Ekstraksi fitur CNN (MobileNetV2) yang dikombinasikan dengan klasifikator SVM . Akurasi: Mencapai 97,8% .	Jumlah Kelas 4 kelas (Cordana, Sigatoka, Pestalotiopsis, dan Sehat). Alur teknis klasifikasi hibrida (CNN-SVM) cenderung kompleks untuk diintegrasikan secara efisien ke lingkungan JavaScript (<i>MERN Stack</i>).
3	Ferdiansyah, dkk. (2024)[11].	Optimasi teknik <i>pooling</i> untuk menekan risiko <i>overfitting</i> .	CNN dengan modifikasi lapisan GlobalAveragePooling2D . Akurasi: Mencapai akurasi tertinggi sebesar 93,62% .	Jumlah Kelas 4 kelas (Cordana, Sigatoka, Pestalotiopsis, dan Sehat). Batasan: Fokus utama pada eksperimen struktur internal model; belum menyediakan antarmuka pengguna (<i>UI</i>) yang siap pakai bagi petani.
4	AR B, dkk. (2025)[14].	Transparansi diagnosis melalui <i>interpretability</i>	Arsitektur DenseNet201 yang diintegrasikan dengan teknik visualisasi Grad-CAM .	Jumlah Kelas 4 kelas (Cordana, Sigatoka, Pestalotiopsis, dan Sehat).

No	Peneliti (Tahun)	Fokus Penelitian	Teknologi & Metode	Keterbatasan / Research Gap
		<i>le deep learning.</i>	Akurasi: Mencapai di atas 97% (spesifik 98,12% pada dataset BananaLSD).	Arsitektur DenseNet201 memiliki parameter yang berat, menjadikannya kurang efisien untuk akses cepat melalui <i>browser web</i> standar.
5	Sanga, dkk. (2020)[12].	Evaluasi model <i>deep learning</i> untuk implementasi praktis petani.	CNN Arsitektur InceptionV3 (arsitektur ringan untuk <i>edge deployment</i>). Akurasi: Mencapai 95,41% pada citra <i>real-time</i> .	Jumlah Kelas Fokus pada Sigatoka dan <i>Fusarium Wilt</i> (min. 3 kelas) Berbasis aplikasi <i>mobile native</i> (Android APK); aksesibilitas terbatas bagi petani yang enggan melakukan instalasi aplikasi tambahan.
6	Penelitian Ini	Rancang bangun sistem deteksi berbasis <i>website</i> yang aksesibel dan edukatif.	CNN perbandingan dari Arsitektur ResNet50 dan MobileNetV2 dengan integrasi MERN STACK , FastAPI , dan PWA . Akurasi: Mencapai akurasi puncak sebesar 98,6% untuk Resnet50 dan untuk MobilenetV2 sebesar 96,9% .	Jumlah Kelas 7 (6 penyakit termasuk virus batang BBrMV + 1 Sehat). Implementasi dibatasi pada deteksi visual via browser; hasil sangat bergantung pada ketajaman citra (bebas <i>blur</i> atau silau) yang diunggah petani.

Berdasarkan tinjauan terhadap berbagai penelitian terdahulu, terlihat adanya tren positif dalam pencapaian akurasi model, di mana penggunaan arsitektur mendalam seperti ResNet50 (Syihad, 2023) dan DenseNet201 (ARB, 2025) mampu menyentuh angka di atas 94%. Namun, terdapat celah di mana model-model dengan akurasi tinggi tersebut sering kali belum ditranslasikan ke dalam platform yang praktis bagi petani, atau masih bersifat aplikasi native (Sanga, 2020). BananaVision hadir untuk menyempurnakan batasan tersebut dengan mengadopsi ketangguhan arsitektur ResNet50 (akurasi 98,1%) ke dalam ekosistem web modern *MERN STACK* yang dioptimasi dengan PWA. Hal ini memungkinkan petani mendapatkan diagnosis seakurat pakar secara instan tanpa perlu melakukan instalasi aplikasi rumit, sekaligus mengatasi kendala subjektivitas identifikasi manual di lapangan

Tabel 2. 2 Matriks Perbandingan

No	Fitur Utama	Syihad (2023)	Lase (2025)	Ferdiansyah (2024)	ARB (2025)	Sanga (2020)	Penelitian Ini
1	Klasifikasi Citra CNN	Ya	Ya	Ya	Ya	Ya	Ya
2	Integrasi Web (MERN)	Tidak	Tidak	Tidak	Tidak	Tidak	Ya
3	Teknologi PWA	Tidak	Tidak	Tidak	Tidak	Tidak	Ya
4	Fitur Riwayat Deteksi	Tidak	Tidak	Tidak	Tidak	Ya	Ya
5	Katalog Edukasi Penyakit	Tidak	Tidak	Tidak	Tidak	Tidak	Ya

No	Fitur Utama	Syihad (2023)	Lase (2025)	Ferdiansyah (2024)	ARB (2025)	Sanga (2020)	Penelitian Ini
6	Akses Real-time (API)	Tidak	Tidak	Tidak	Tidak	Tidak	Ya

2.2 Landasan Teori

2.2.1 Komoditas Pisang dan Permasalahan Penyakit

1. Signifikansi Komoditas Pisang di Indonesia

Pisang merupakan salah satu komoditas hortikultura unggulan yang memiliki peran vital dalam perekonomian Indonesia dan ketahanan pangan. Indonesia secara konsisten menduduki peringkat tinggi sebagai produsen pisang global, yang menjadikan komoditas ini sebagai sumber pendapatan penting bagi jutaan petani kecil (Badan Pusat Statistik, 2023).

Secara botani, tanaman pisang termasuk dalam keluarga *Musaceae*. Klasifikasi dan jenis-jenis pisang komersial yang umum dibudidayakan di Indonesia mencakup varietas meja (misalnya Pisang Cavendish, Pisang Ambon) dan varietas olahan (misalnya Pisang Kepok, Pisang Tanduk). Kebutuhan pasar, baik domestik maupun ekspor, menunjukkan permintaan yang stabil dan terus meningkat, sehingga menjaga produktivitas dari ancaman penyakit menjadi sangat krusial[15][16].

2. Penyakit Tumbuhan Pisang

Meskipun memiliki nilai ekonomi tinggi, budidaya pisang rentan terhadap serangan patogen yang dapat menyebabkan kerugian hasil panen[17]. Penjelasan rinci mengenai penyakit-penyakit yang menjadi fokus penelitian ini disajikan sebagai berikut:

- a. Penyakit Layu Fusarium (*Fusarium Wilt*) : Penyakit *Fusarium wilt* atau bisa disebut juga penyakit panama pada tanaman pisang, yang disebabkan oleh *Fusarium oxysporum f.sp. cubense* (Foc), merupakan ancaman serius bagi produksi pisang di seluruh dunia, termasuk Indonesia. Penyakit ini mengakibatkan gejala layu daun, penguningan, dan kematian tanaman, dengan kerugian ekonomi yang signifikan, mencapai 50% hingga 100% pada varietas yang rentan[18].
- b. Penyakit Layu Bakteri (*Bacterial Wilt*) : Penyakit Layu Bakteri atau disebabkan oleh bakteri *Ralstonia solanacearum* (secara historis dikenal sebagai *Pseudomonas solanacearum*) dan *Xanthomonas campestris* pv. *musacearum* Di Asia, bakteri yang relevan seperti *Moko Disease*. Penyakit ini merupakan salah satu penyakit paling merusak pada tanaman pisang, terutama di wilayah tropis. Infeksi bakteri menyebabkan terganggunya aliran air dan

nutrisi dalam jaringan tanaman sehingga menimbulkan kelayuan mendadak [19].

- c. Penyakit Daun Sigatoka : Penyakit Sigatoka yang disebabkan oleh jamur *Mycosphaerella musicola* (Sigatoka Kuning) dan *Mycosphaerella fijiensis* (Black Sigatoka) ini menyerang daun pisang dengan menimbulkan bercak kecil berwarna kuning yang kemudian berkembang menjadi garis coklat atau hitam, menyebabkan daun mengering dan mengalami nekrosis hingga akhirnya mati, sehingga kemampuan fotosintesis tanaman menurun secara signifikan[20].
- d. Penyakit *Bract Mosaic (Banana Bract Mosaic)* : Penyakit ini disebabkan oleh *Banana bract mosaic virus* (BBrMV) yang termasuk dalam kelompok *Potyvirus*. Penyakit ini menginfeksi tanaman pisang dan menimbulkan gejala khas berupa pola mosaik atau garis-garis sejajar berwarna coklat kemerahan hingga gelap pada seludang jantung pisang (*bract*), tangkai daun, dan batang semu (*pseudostem*). Infeksi virus ini yang umumnya ditularkan melalui bahan tanam (bibit) yang terinfeksi maupun melalui vektor kutu daun (seperti *Aphis gossypii*) dapat menyebabkan malformasi pada tandan, pengkerdilan tanaman, hingga penurunan kuantitas serta kualitas hasil panen secara signifikan[21].

2.2.2 Konsep Machine Learning dan Convolutional Neural Network (CNN)

1. *Machine Learning* dan *Deep Learning*

Machine Learning merupakan cabang kecerdasan buatan yang memungkinkan komputer belajar dari data untuk menghasilkan prediksi atau keputusan tanpa harus diprogram secara eksplisit. Di dalamnya terdapat berbagai algoritma yang mempelajari pola dari data yang diberikan. Sementara itu, *Deep Learning* adalah bagian khusus dari *Machine Learning* yang menggunakan arsitektur jaringan saraf tiruan berlapis-lapis untuk memproses data dalam jumlah besar dan kompleks.

Perbedaan utama keduanya terletak pada tingkat kompleksitas model dan kemampuan dalam mengekstraksi fitur. *Machine Learning* tradisional masih memerlukan proses ekstraksi fitur secara manual, sedangkan *Deep Learning* mampu mempelajari fitur secara otomatis melalui lapisan-lapisan representasi. Kemampuan inilah yang membuat *Deep Learning* sangat efektif untuk tugas *Computer Vision*, karena dapat mengenali pola visual seperti bentuk, tekstur, dan objek secara lebih akurat meskipun data gambar memiliki variasi yang tinggi.

2. Arsitektur *Convolutional Neural Network* (CNN)

Convolutional Neural Network (CNN) merupakan arsitektur *Deep Learning* yang dirancang khusus untuk mengolah

data berbentuk *grid*, seperti citra. CNN terdiri dari beberapa lapisan utama, yaitu *convolutional layer* untuk mengekstraksi fitur visual, *pooling layer* untuk mengecilkan dimensi dan mengurangi kompleksitas, serta *fully connected layer* yang berfungsi melakukan klasifikasi berdasarkan fitur yang telah dipelajari. Struktur berlapis ini memungkinkan CNN mengenali pola visual mulai dari fitur sederhana seperti tepi dan tekstur hingga fitur kompleks seperti bentuk objek. Kemampuan hierarkis dalam mengekstraksi fitur secara otomatis menjadikan CNN sangat efektif pada berbagai tugas *Computer Vision*, termasuk deteksi objek, segmentasi citra, dan klasifikasi gambar [22].

3. *Transfer Learning* dan Model Basis

Transfer Learning merupakan pendekatan dalam pembelajaran mesin yang memanfaatkan pengetahuan dari model yang telah dilatih pada dataset besar untuk menyelesaikan tugas baru dengan jumlah data yang lebih sedikit. Melalui metode ini, model tidak perlu belajar dari awal, melainkan hanya menyesuaikan kembali (*fine-tuning*) bobot yang sudah terbentuk agar sesuai dengan permasalahan yang sedang ditangani. Model basis seperti VGG, *ResNet50*, atau *MobileNetV2* adalah model CNN yang telah dilatih secara mendalam pada dataset berskala besar seperti *ImageNet* dan berfungsi sebagai fondasi awal dalam

Transfer Learning. Pendekatan ini terbukti meningkatkan akurasi, mempercepat proses pelatihan, serta mengurangi kebutuhan komputasi, sehingga sangat efektif digunakan pada proyek *Computer Vision* yang membutuhkan performa tinggi dengan sumber daya terbatas.

2.2.3 Pengembangan Sistem Berbasis Website (MERN Stack)

MERN STACK merupakan kombinasi teknologi pengembangan *website modern* yang terdiri dari *MongoDB*, *Express.js*, *React.js*, dan *Node.js*. Keempat teknologi ini bekerja secara terintegrasi untuk membangun aplikasi web *full-stack* yang responsif, cepat, dan mudah dikembangkan. *MongoDB* berfungsi sebagai *database* NoSQL yang fleksibel dalam menyimpan data tidak terstruktur. *Express.js* dan *Node.js* membentuk bagian *backend* yang menangani logika server, pemrosesan data, serta penyediaan API. Sementara itu, *React.js* digunakan pada sisi *frontend* untuk membangun antarmuka pengguna yang interaktif dan dinamis melalui konsep komponen. Arsitektur ini banyak digunakan karena mendukung pengembangan skala besar, mudah dipelihara, serta memungkinkan penggunaan satu bahasa pemrograman utama *JavaScript* di semua sisi aplikasi, sehingga mempercepat proses pengembangan dan meningkatkan efisiensi.

2.2.4 Progressive Web App (PWA)

Pendekatan pengembangan sistem deteksi penyakit tanaman pisang ini akan memanfaatkan *Progressive Web App* (PWA) untuk

menjamin penyediaan layanan yang handal dan *user-friendly* bagi para petani. Dengan mengadopsi PWA, sistem berbasis web ini dapat memberikan pengalaman pengguna yang sangat mirip dengan aplikasi *native (mobile app)*, termasuk kemampuan instalasi langsung ke layar utama (*home screen*) perangkat pengguna tanpa melalui toko aplikasi. Hal ini memungkinkan akses cepat melalui ikon khusus, peluncuran dalam mode *standalone* tanpa bilah alamat *browser*, dan waktu muat yang sangat cepat berkat penggunaan *Service Workers* untuk *caching* aset antarmuka. Meskipun proses deteksi utama tetap memerlukan koneksi internet untuk memproses model *Convolutional Neural Network (CNN)* pada server, arsitektur PWA memastikan antarmuka selalu responsif dan tersedia, bahkan saat koneksi lambat atau terputus, sehingga secara signifikan meningkatkan aksesibilitas dan adopsi di area pertanian pedesaan.

2.2.5 Tensorflow

TensorFlow adalah sebuah *framework open-source* yang dikembangkan oleh Google untuk membangun, melatih, dan menerapkan model *machine learning* serta *deep learning*. *Framework* ini menyediakan berbagai API yang memudahkan pengembang dalam merancang arsitektur jaringan saraf, mengelola data, dan melakukan komputasi numerik secara efisien dengan dukungan CPU maupun GPU. *TensorFlow* juga memiliki ekosistem yang luas, seperti *TensorFlow Lite* untuk perangkat *mobile* dan *embedded*, serta

TensorFlow.js untuk implementasi langsung pada aplikasi web. Dengan fleksibilitas dan skalabilitas yang tinggi, *TensorFlow* menjadi salah satu platform yang banyak digunakan dalam pengembangan aplikasi berbasis pembelajaran mesin, termasuk pada bidang *Computer Vision*, pemrosesan citra, dan klasifikasi objek.

2.2.6 FastApi

FastAPI adalah *framework backend* berbasis *Python* yang memiliki performa tinggi dan sangat cocok untuk mengimplementasikan model *machine learning* pada aplikasi berbasis web. Dalam konteks tugas akhir ini, *FastAPI* digunakan sebagai layanan server untuk memuat (*load*) model yang telah dilatih sebelumnya dan menyediakan *endpoint API* yang dapat menerima input berupa citra, memprosesnya melalui model, kemudian mengembalikan hasil prediksi kepada sistem *frontend*. *FastAPI* mendukung pemrograman *asynchronous*, struktur routing yang sederhana, serta dokumentasi otomatis melalui *OpenAPI*, sehingga mempercepat proses integrasi antara model *machine learning* dan aplikasi *website*. Berkat efisiensi dan kemudahan penggunaannya, *FastAPI* menjadi pilihan tepat untuk membangun *backend* yang stabil, cepat, dan skalabel pada proyek deteksi penyakit tanaman pisang ini.

2.2.7 TailwindCss

TailwindCSS merupakan *framework CSS utility-first* yang menyediakan kelas-kelas siap pakai untuk mengatur tampilan

antarmuka secara cepat dan konsisten. Berbeda dengan *framework* tradisional yang menawarkan komponen siap pakai, *Tailwind* memberikan kebebasan penuh kepada pengembang untuk merancang desain sesuai kebutuhan dengan memanfaatkan *utility* seperti margin, *padding*, warna, ukuran *font*, hingga sistem *grid*. Pendekatan ini mempermudah proses *styling* tanpa harus menulis CSS dari nol dan menjaga konsistensi desain di seluruh halaman. *Tailwind* juga mendukung proses *build* yang efisien melalui fitur *purge* yang menghapus kelas yang tidak digunakan, sehingga menghasilkan file CSS berukuran kecil dan optimal untuk performa web. *Framework* ini banyak digunakan dalam pengembangan *frontend* modern karena fleksibel, cepat, dan mudah disesuaikan.

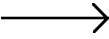
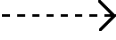
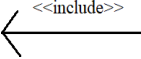
2.2.8 Unified Modeling Language (UML)

Unified Modeling Language (UML) adalah suatu bahasa pemodelan visual yang digunakan untuk menggambarkan, merancang, dan mendokumentasikan sistem perangkat lunak. UML menyediakan berbagai diagram untuk menggambarkan struktur, perilaku, dan interaksi dalam sistem, dan sering digunakan dalam pengembangan perangkat lunak berbasis objek, UML membantu pengembangan perangkat lunak, arsitek, dan analisis sistem untuk merencanakan dan memahami sistem yang sedang dibangun dengan cara yang lebih terstruktur dan mudah dipahami. Terdapat beberapa

diagram UML yang sering digunakan dalam pengembangan sebuah sistem yaitu:

1. *Use Case* : merupakan gambaran dari fungsionalitas yang diharapkan dari sebuah sistem dan mempresentasikan sebuah interaksi antara *actor* dan sistem. Didalam *use case* terdapat actor yang merupakan sebuah sistem yang melakukan pekerjaan di sistem.

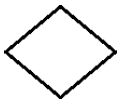
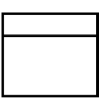
Tabel 2. 3 Simbol Use Case Diagram

No	Simbol	Nama	Keterangan
1		<i>Actor</i>	Mewakili peran orang, sistem yang lain, atau alat ketika berkomunikasi dengan <i>use case</i> .
2		<i>Use Case</i>	Mewakili peran orang, sistem yang lain, atau alat ketika berkomunikasi dengan <i>use case</i> .
3		<i>Association</i>	Abstraksi dari penghubung antara aktor dan <i>use case</i> .
4		<i>Generalisasi</i>	Menunjukkan spesialisasi aktor untuk dapat berpartisipasi dengan <i>use case</i> .
5		<i>include</i>	Menunjukkan bahwa suatu use case seluruhnya merupakan

2. *Activity Diagram* : sebuah diagram aktivitas UML menggambarkan perilaku dinamis dari suatu sistem atau bagian

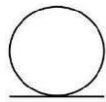
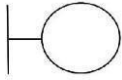
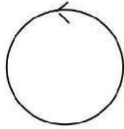
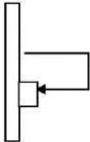
dari sistem melalui aliran control antara aksi yang dilakukan sistem.

Tabel 2. 4 Simbol Activity Diagram

No	Simbol	Nama	Keterangan
1		Status awal	Sebuah diagram aktivitas memiliki sebuah status awal.
2		Aktivitas	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.
3		<i>Decision</i> atau percabang	Percabangan dimana ada pilihan aktivitas yang lebih dari satu.
4		Penggabungan atau <i>Join</i>	Penggabungan dimana yang mana lebih dari satu aktivitas lalu digabungkan jadi satu.
5		Status Akhir	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir.
6		<i>Swimlane</i>	<i>Swimlane</i> memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi
7		<i>Flow Final Node</i>	Digunakan untuk menghentikan satu jalur aliran (<i>flow</i>) aktivitas spesifik saja di dalam diagram tanpa menghentikan seluruh proses aktivitas sistem secara keseluruhan. Jalur aliran lain yang

3. *Sequence Diagram* : menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* biasa digunakan untuk menggambarkan scenario atau langkah-langkah yang dilakukan sebagai *respons* dari sebuah *event* untuk menghasilkan *output* tertentu.

Tabel 2. 5 Simbol sequence diagram

No	Simbol	Nama	Keterangan
1		<i>Entity Class</i>	Gambaran sistem sebagai landasan dalam menyusun basidata.
2		<i>Boundary Class</i>	Menangani komunikasi antar lingkungan sistem.
3		<i>Control Class</i>	Bertanggung jawab terhadap kelas-kelas terhadap objek yang berisi logika.
4		<i>Recursive</i>	Pesan untuk dirinya.
5		<i>Activation</i>	Mewakili proses durasi aktivasi sebuah operasi.
6		<i>Life Line</i>	Komponen yang digambarkan garis putus terhubung dengan objek.

4. *Class Diagram* :merupakan gambaran struktur dan deskripsi dari *class*, *package*, dan objek yang saling berhubungan seperti diantaranya pewarisan, asosiasi, dan lainnya.

Tabel 2. 6 Simbol Class Diagram

No	Gambar	Simbol	Keterangan
1		<i>Class</i>	Menggambarkan kumpulan objek-objek dengan atribut dan operasi yang sama
2		<i>Nary Association</i>	Sebagai upaya menghindari asosiasi dengan lebih dari 2 objek
3		<i>Collaboration</i>	Urutan aksi-aksi yang ditampilkan oleh sistem yang menghasilkan suatu hasil yang tertukar bagi suatu <i>actor</i>
4		<i>Generalization</i>	Menggambarkan dimana hubungan objek <i>descendent</i> berperilaku dan struktur data objek yang di atasnya objek induk <i>ancestor</i>
5		<i>Realization</i>	Operasi yang benar-benar dilakukan oleh suatu objek

No	Gambar	Simbol	Keterangan
6		<i>Dependency</i>	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri akan mempengaruhi elemen yang tidak mandiri.