BAB II

LANDASAN TEORI

2.1 Tinjauan Pustaka

Tinjauan pusataka bertujuan untuk mengkaji penelitian dan literatur yang relevan dengan menggunakan *Vision Transformer* untuk klasifikasi penyakit daun padi

2.1.1 Klasifikasi Penyakit Daun Padi

Menurut (Irwan & Mu'min, 2020) Penyakit daun padi adalah salah satu ancaman utama yang dapat menurunkan hasil panen secara signifikan. Berbagai penyakit ini disebabkan oleh patogen seperti jamur, hama, bakteri, dan virus. Pentingnya diagnosis dini dan akurat tidak bisa diabaikan, karena hal ini sangat penting untuk menjaga ketahanan pangan. Beberapa jenis penyakit daun padi yang sering ditemukan di Indonesia antara lain:

1. **Hawar Daun Bakteri** (*Bacterial Leaf Blight*)

   Penyakit ini disebabkan oleh bakteri *Xanthomonas campestris pv. Oryzae* dan biasanya menyerang tanaman padi dari fase anakan hingga fase pemasakan. Gejalanya dimulai dengan munculnya bercak abu-abu kekuningan di tepi daun. Seiring waktu, bercak-bercak ini akan meluas dan membentuk hawar, yang akhirnya menyebabkan daun mengering sepenuhnya seperti gambar 2.1 dibawah ini.

Gambar 2. 1 Gejala Hawar Daun Bakteri

2. Blast

Penyakit ini disebabkan oleh jamur *Pyricularia oryzae*, dengan gejala awal berupa bercak coklat kehitaman berbentuk belah ketupat dengan ujung runcing, dan pusat bercak berwarna putih. Serangan yang parah dapat menghambat pertumbuhan tanaman dan mengurangi jumlah anakan produktif, menghasilkan malai kecil dengan sedikit gabah. Dalam kasus yang ekstrem, serangan dapat menyebabkan seluruh tanaman mati sebelum berbunga.

Gambar 2. 2 Gejala Blast

3. Bercak Cokelat (*Brown Spot)*

Penyakit ini disebabkan oleh jamur *Helminthosporium oryzae*. Gejala serangan penyakit bercak cokelat yang paling umum adalah berbentuk bulat

atau oval berwarna cokelat dengan tepian berwarna kuning seperti gambar 2.3 dibawah ini. Jika serangan berat, bercak merata diseluruh permukaan daun sehingga kering dan tanaman menjadi kerdil.

Gambar 2. 3 Gejala Bercak Cokelat

4. Tungro

Penyakit tungro pada tanaman padi disebabkan oleh infeksi dua virus berbeda, yaitu Rice Tungro Bacilliform Virus (RTBV) dan Rice Tungro Spherical Virus (RTSV), yang ditularkan oleh hama wereng hijau, terutama Nephotettix virescens. Gejala yang mencolok dari serangan tungro termasuk perubahan warna daun menjadi kuning atau jingga, serta pertumbuhan tanaman yang kerdil. Perubahan warna daun dimulai dari bagian ujung dan bisa meluas hingga ke bagian pangkal daun. Tingkat kekerdilan tanaman juga bervariasi, dari ringan hingga sangat kerdil, tergantung pada tingkat keparahan infeksi.

Gambar 2. 4 Gejala Tungro

2.1.1.1 Metode Klasifikasi Tradisional

Metode klasifikasi tradisional untuk penyakit daun padi umumnya dilakukan secara visual oleh para ahli atau petani. Metode ini memiliki beberapa keterbatasan, seperti:

1. **Keakuratan yang rendah:** Keakuratan metode klasifikasi tradisional dapat bervariasi tergantung pada pengalaman dan keahlian ahli serta kondisi pencahayaan dan lingkungan saat pengujian dilakukan.
2. **Subjektivitas:** Metode klasifikasi tradisional bersifat subjektif, karena diagnosis didasarkan pada interpretasi ahli terhadap gejala penyakit. Hal ini dapat menyebabkan perbedaan diagnosis antar ahli.
3. **Kecepatan yang lambat:** Proses klasifikasi tradisional dapat memakan waktu lama, terutama jika diperlukan pemeriksaan laboratorium atau mikroskopis.
4. **Ketidakmampuan untuk mendeteksi penyakit secara dini:** Metode klasifikasi tradisional mungkin tidak dapat mendeteksi penyakit secara dini, ketika gejala masih belum jelas.

2.1.1.2 Metode Klasifikasi Berbasis AI

Metode klasifikasi berbasis AI untuk penyakit tanaman padi menggunakan Artificial Intelligence (AI) untuk menganalisis data gambar, sensor, dan data lainnya untuk mendiagnosis penyakit pada tanaman padi. Metode ini menawarkan beberapa keunggulan dibandingkan metode klasifikasi tradisional, seperti:

1. **Keakuratan yang tinggi:** Metode berbasis AI dapat mencapai tingkat akurasi yang tinggi dalam mendiagnosis penyakit tanaman padi. Hal ini karena AI dapat menganalisis data gambar dengan lebih objektif dan konsisten dibandingkan manusia.
2. **Objektivitas:** Metode berbasis AI bersifat objektif, karena diagnosis didasarkan pada analisis data yang tidak bias.
3. **Kecepatan yang cepat**: Metode berbasis AI dapat mendiagnosis penyakit tanaman padi dengan cepat dan efisien.
4. **Kemampuan untuk mendeteksi penyakit secara dini:** AI dapat mendeteksi penyakit tanaman daun padi secara dini, ketika gejala masih belum jelas.

2.1.1.3 Dampak Penyakit Daun Padi terhadap Hasil Panen

Penyakit daun pada padi mempunyai dampak yang signifikan terhadap hasil panen dan dapat terjadi tanpa terdeteksi pada waktunya. Salah satu dampak utamanya adalah berkurangnya produksi biji-bijian akibat kerusakan daun, sehingga menurunkan kapasitas fotosintesis tanaman.

Penyakit seperti blast, hawar daun, bercak coklat, dan tungro pada padi dapat menyebabkan daun menguning, kering, atau bahkan rontok sehingga

menyebabkan tanaman kehilangan unsur hara penting yang dibutuhkan untuk membentuk bunga dan biji sehingga menghambat tanaman dalam memproduksi cukup energi untuk berproduksi bahan organik. Tanaman yang tidak sehat pada akhirnya menurunkan produksi gabah dan menurunkan kualitas.

2.1.2 *Computer Vision*

Computer Vision adalah cabang ilmu komputer yang memikat, yang memusatkan perhatian pada pengembangan sistem cerdas untuk memahami dan menginterpretasi informasi dari dunia visual. Sistem ini umumnya menggunakan kamera digital dan algoritma pengolahan gambar untuk menangkap, menganalisis, dan memahami citra dan video. Prinsip *computer vision* terutama mencakup akuisisi gambar, *preprocessing* gambar, fitur ekstraksi, pengenalan gambar dan pemahaman gambar. Akuisisi gambar adalah dasar dari *computer vision*, yang menggunakan kamera, sensor dan perangkat lain untuk mengubah gambar di dunia nyata menjadi sinyal digital untuk pemrosesan komputer. (Chen et al., 2024)

Computer Vision memiliki cakupan yang luas dan mencakup berbagai teknik dan algoritma untuk analisis dan interpretasi data visual, seperti gambar dan video. Ini meliputi:

1. **Pengenalan Objek:** Pengidentifikasian dan klasifikasi objek dalam gambar atau video.
2. **Deteksi Objek**: Mendeteksi keberadaan dan lokasi objek dalam gambar atau video.
3. **Segmentasi Citra**: Memisahkan gambar menjadi bagian-bagian yang saling terkait, seperti objek dan latar belakang.

4. **Pengenalan Wajah**: Identifikasi individu berdasarkan fitur wajah mereka.
5. **Pelacakan Objek**: Melacak pergerakan objek dalam ruang dan waktu dalam video.
6. **Augmented Reality (AR)**: Menggabungkan elemen digital dengan dunia nyata dalam waktu nyata.
7. **Klasifikasi Gambar**: Mengklasifikasikan gambar ke dalam kategori-kategori tertentu.
8. **Pengenalan Aksi**: Mendeteksi dan mengidentifikasi tindakan atau gerakan dalam video.
9. **Pengenalan Karakter Optik**: Mengenali karakter dalam dokumen yang dipindai atau gambar.
10. **Pengolahan Citra dan Video**: Manipulasi gambar dan video untuk berbagai tujuan seperti peningkatan kualitas, pengurangan noise, atau kompresi.

2.1.3 *Machine Learning*

Menurut (Id, 2021) *Machine Learning (ML)* adalah salah satu cabang dari kecerdasan buatan (AI) yang menarik perhatian banyak, fokus pada kemampuan sistem untuk belajar dari data tanpa perlu diprogram ulang oleh manusia secara berulang. Proses pembelajaran ML dimulai dengan data valid selama tahap pelatihan (training) untuk menghasilkan *output* optimal saat diujikan (*testing)*. Jenis-jenis ML dapat dikelompokkan berdasarkan metode pembelajarannya, yang umumnya terbagi menjadi tiga kelompok utama:

1. Supervised Learning

Supervised Learning adalah model diberi data yang sudah diberi label dan belajar untuk membuat prediksi atau klasifikasi berdasarkan data tersebut. Contoh penggunaan termasuk klasifikasi gambar dan deteksi objek.

2. Unsupervised Learning

Unsupervised Learning adalah konsep yang menarik dalam machine learning di mana model belajar dari data tanpa label untuk menemukan pola atau struktur tersembunyi. Contohnya termasuk pengelompokkan data pelanggan berdasarkan perilaku atau preferensi yang tidak diketahui sebelumnya.

3. Reinforcement Learning

Reinforcement Learning memperkenalkan ide belajar melalui interaksi dengan lingkungan, menerima umpan balik positif atau negatif berdasarkan tindakan yang diambil. Contoh aplikasinya sangat beragam, seperti pengembangan mobil *self-driving* yang belajar dari pengalaman berinteraksi dengan jalan dan lalu lintas.

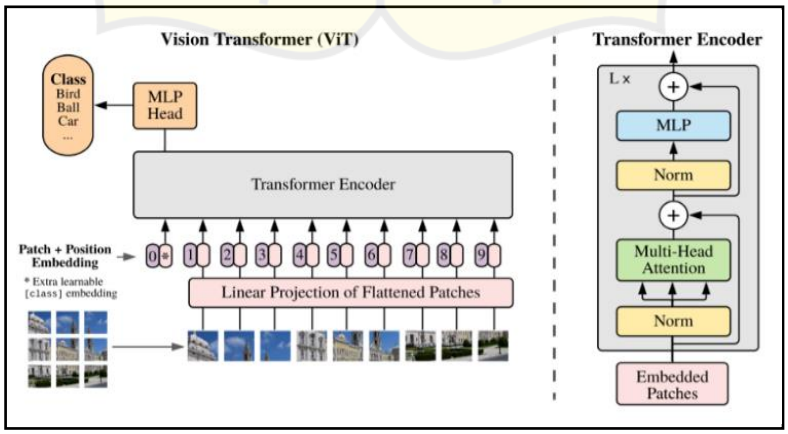
2.1.4 *Deep Learning*

Deep learning, sebagai sub-bidang penting dari *machine learning*, menggabungkan konsep jaringan saraf tiruan (JST) yang berlapis-lapis untuk memahami dan memodelkan representasi data yang kompleks. Inspirasi dari struktur dan fungsi otak manusia membuatnya sangat kuat dalam memproses data dari berbagai domain.(Kelleher, 2019).

2.1.5 *Vision Transformer*

Transformer diperkenalkan di bidang NLP untuk menerjemahkan tugas bahasa Inggris ke bahasa Jerman. *Transformer* ini semata-mata didasarkan pada mekanisme attention, bukan lapisan konvolusi. Ini terdiri dari modul *encoder* dan *decoder*. Kedua modul terdiri dari lapisan perhatian dengan jaringan *feed forward*. Saat ini, ViT telah muncul dan menunjukkan hasil yang lebih menjanjikan, yang dapat mengungguli CNN. Struktur ViT didasarkan pada mekanisme *self attention* dan melakukan tugas visual dengan sangat baik.(Islam, 2022)

Vision Transformer (ViT) mewakili evolusi terbaru dalam deep learning dengan mengadopsi arsitektur *Transformer*, yang awalnya dikembangkan untuk pemrosesan bahasa alami. ViT menggunakan pendekatan *patch embeddings, multi-head attention, dan multi-layer perceptron* untuk mengubah cara kita memahami dan menganalisis data visual secara mendalam dan efisien. Gambar *input* dibagi menjadi beberapa *patch* berdasarkan ukuran arsitekturnya selama proses penerapan *patch*. Setelah itu, gambar diberi *position embeddings*, yang menunjukkan lokasi *patch* pada gambar. Setelah itu, *patch* akan dimasukkan ke *multi-head attention* seperti gambar 2.5.(Dosovitskiy et al., 2020)



Gambar 2. 5 Arsitektur *Vision Transformer* (Sumber : Dosovitsky 2020)

2.1.6 *Patch Embedding*

Patch embedding adalah teknik yang digunakan untuk pemrosesan gambar, khususnya dalam konteks pengolahan gambar oleh model jaringan saraf tiruan (*neural networks*). Dalam konteks pengolahan gambar, *patch embedding* mengacu pada membagi gambar menjadi bagian-bagian kecil yang disebut *patch* dan mengonversi setiap *patch* menjadi sekumpulan vektor fitur, yang memungkinkan model jaringan saraf untuk mengoperasikan gambar sebagai sekumpulan vektor fitur daripada sekumpulan grid piksel. Persamaan 2.1 akan menjelaskan rumus perubahan vektor dua dimensi menjadi satu dimensi pada proses *patch embeddings*.(Figo et al., 2023)

$$X \in RH \times W \times C \rightarrow XP \in RN \times (P2.C) \quad (2.1)$$

Keterangan :

X = Citra masukan.

Xp = Citra masukan setelah transformasi.

H = Tinggi citra masukan.

W = Lebar citra masukan.

C = Jumlah channel citra masukan.

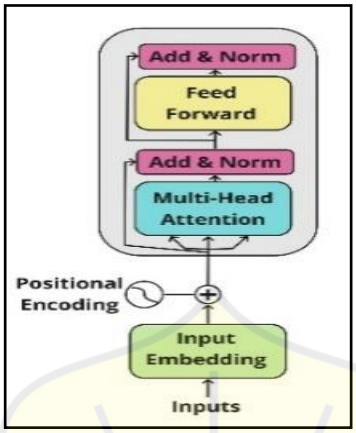
P = Ukuran patch.

N = $HW/P2$.

2.1.7 *Transformer Encoder*

Pada *Vision Transformer* (ViT), komponen terpentingnya adalah *transformer encoder* yang berisi blok MHSA dan MLP. *Transformer encoder* dalam ViT bertanggung jawab atas pemrosesan representasi fitur gambar, mengubah setiap piksel gambar menjadi *vektor embedding*, dan kemudian

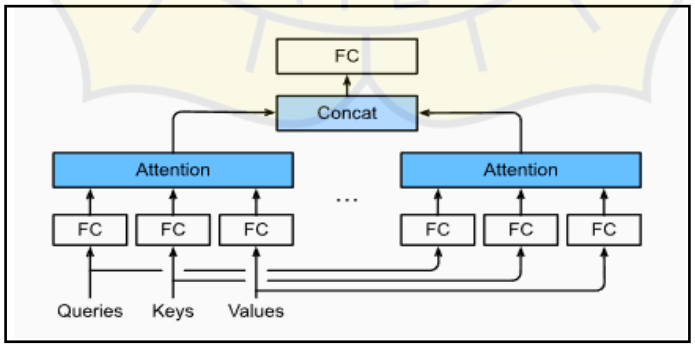
menggabungkan informasi dari seluruh gambar melalui multi-*head self-attention* seperti gambar 2.6.(Vaswani et al., 2017)



Gambar 2. 6 *Transformer Encoder* (Sumber :Vaswani 2017)

2.1.8 *Multi Head Attention*

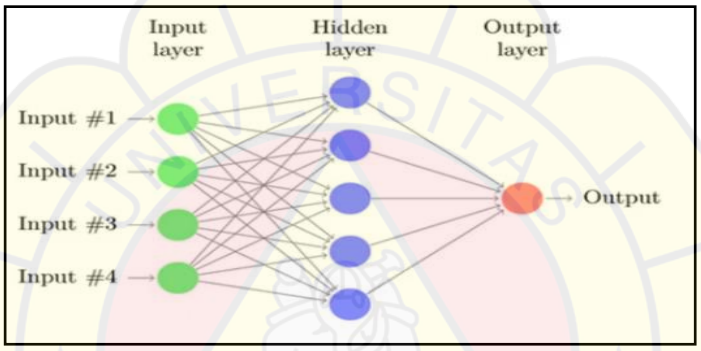
Multi Head Attention adalah salah satu mekanisme perhatian (*attention mechanism*) yang digunakan dalam arsitektur *transformer* untuk memungkinkan model fokus pada beberapa bagian *input* secara bersamaan. Hal ini memungkinkan model memperoleh representasi data yang lebih baik dengan mempertimbangkan hubungan yang lebih kompleks antar elemennya seperti gambar 2.7.(Vaswani et al., 2017)



Gambar 2. 7 *Multi Head Attention* (Sumber : Vaswani 2017)

2.1.9 *Multi Layer Perceptron*

Menurut (Figo et al., 2023) *Multi Layer Perceptron (MLP)* adalah model jaringan saraf tiruan yang terinspirasi dari sistem saraf manusia. MLP terdiri dari tiga bagian utama: lapisan masukan, lapisan tersembunyi, dan lapisan keluaran. Seperti halnya sistem saraf manusia, setiap neuron dalam MLP memiliki bobot yang penting dalam proses pembelajaran. Fungsi aktivasi nonlinier hadir di setiap neuron, kecuali pada lapisan masukan seperti gambar 2.8.



Gambar 2. 8 *Multi Layer Perceptron* (Sumber : Figo 2023)

2.1.10 *Layer Normalization*

Menurut (Figo et al., 2023) Proses pelatihan dalam arsitektur *deep learning* sering memerlukan banyak sumber daya komputasi. Untuk mengurangi konsumsi energi dan waktu yang diperlukan, penggunaan teknik seperti *layer normalization* dapat sangat bermanfaat. *Layer normalization* mengelola nilai input dengan menggunakan rata-rata dan deviasi standar, menjaga agar nilai input mendekati nol semakin dekat dengan rata-rata dan standar deviasi.. Persamaan 2.2 memuat rumus menghitung normalisasi kelas.

$$x'_{i,k} = \frac{x_{i,k} - \mu^i}{\sqrt{\sigma_i^2 + \epsilon}} \tag{2.2}$$

Keterangan :

xi,k = Nilai inputan / vektor

μi = Rata-rata nilai inputan / vektor

σ_i^2 = Standar deviasi nilai inputan / vektor

ϵ = Denominator

2.1.11 *CRISP-DM*

Menurut (Schröer et al., 2021) CRISP-DM (Cross-Industry Standard Process for Data Mining) adalah model proses yang independen untuk data mining, terdiri dari enam langkah utama:

1. ***Business Understanding***

   Business understanding adalah fase awal dari proses data mining di mana penting untuk memahami masalah yang akan diselesaikan. Pada fase ini, peneliti fokus memahami tujuan bisnis dan masalah bisnis yaitu untuk mendeteksi dini penyakit pada daun padi masih terbatas menggunakan metode tradisional untuk mengidentifikasi penyakit, yang seringkali tidak akurat dan membutuhkan waktu yang cukup lama. Setelah memahami masalah, peneliti menentukan solusi bisnis dari jenis *data mining* yang ada berdasarkan masalah yang akan diselesaikan.

2. ***Data Understanding***

   Data understanding adalah fase dari proses *data mining* di mana penting untuk memahami data yang tersedia untuk digunakan dalam menciptakan

solusi terhadap masalah bisnis. Pada fase ini, peneliti menelaah data yang telah dikumpulkan untuk melihat kecocokan data terhadap masalah yang akan diselesaikan. Data yang dikumpulkan adalah data gambar dengan 5 kelas penyakit daun padi yaitu bercak coklat, blast, hawar daun bakteri, tungro dan sehat dengan total keseluruhan sebesar 1.785 gambar.

3. Data Preparation

Data preparation adalah fase dari proses *data mining* yang berfokus membersihkan, mengubah, dan menyiapkan data agar sesuai untuk proses *modelling*. Pada tahap ini, peneliti menentukan label data sesuai dengan jenis penyakit daun padi.

4. Modelling

Modelling adalah fase yang berfokus pada pengembangan model *data mining* yang akan digunakan untuk memprediksi atau menjelaskan pola dalam data. Pada tahap ini, peneliti membangun model menggunakan arsitektur *vision transformer* untuk mendeteksi penyakit daun padi.

5. Evaluation

Evaluation adalah Fase ini berfokus pada evaluasi kinerja model *data mining* dan pemilihan model terbaik untuk digunakan.. Pada tahap ini, peneliti mengevaluasi hasil pemodelan dengan arsitektur *vision transformer*.

6. Deployment

Deployment adalah fase terakhir dari CRISP-DM, Fase ini berfokus pada penerapan model data mining ke dalam sistem produksi. Pada tahap ini, peneliti merencanakan *deployment* model dalam bentuk aplikasi *mobile*.

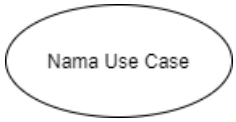
2.1.12 Pemodelan Sistem UML

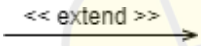
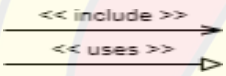
Menurut (Hasanah & Untari, 2020) Bahasa pemodelan grafis yang disebut pemodelan UML digunakan untuk merancang sistem perangkat lunak. Untuk memvisualisasikan berbagai aspek sistem, seperti struktur, perilaku, dan interaksi, UML menyediakan berbagai diagram dan simbol. Untuk membuat rancangan yang dapat memberikan informasi yang teratur, peneliti menggunakan rancangan UML yang terdiri dari:

2.1.12.1 *Use Case Diagram*

Menurut (Hasanah & Untari, 2020) *Use case* diagram menunjukkan interaksi antara aktor dan sistem dan menekankan "apa" yang dilakukan sistem daripada "bagaimana". Teknik ini digunakan untuk merekam persyaratan fungsional sebuah sistem dan menggambarkan fungsionalitas yang diharapkan dari sistem. Sebuah tugas tertentu, seperti mengakses sistem, membuat daftar belanja, dan sebagainya, disebut use case. Sebuah aktor dapat berupa manusia atau mesin yang berinteraksi dengan sistem untuk melakukan tugas tertentu.

Tabel 2. 1 Simbol Use Case Diagram (Sumber : Hasanah & Untari 2020)

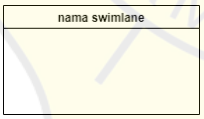
No	Simbol	Deskripsi
1	*Use Case* 	Fungsi yang disediakan oleh sistem kepada aktor. Menggambarkan interaksi antara aktor dan sistem.

2	*Actor*  	Entitas yang berinteraksi dengan sistem dengan Aktor sebagai pengguna.
3	*Association* 	Hubungan antara aktor dan *use case* yang menunjukkan keterlibatan atau penggunaan *use case* tertentu oleh aktor.
4	*Extend*  	Hubungan bahwa *use case* dapat memperluas atau mengubah perilaku dari *use case* lain dalam kondisi tertentu.
5	*Include*  	Hubungan bahwa *use case* mencakup atau mengandalkan fungsionalitas dari use case lain.

2.1.12.2 *Activity Diagram*

Menurut (Hasanah & Untari, 2020) berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, keputusan yang mungkin terjadi, dan bagaimana mereka berakhir, digambarkan dalam *activity diagram*.

Tabel 2. 2 Simbol Activity Diagram (Sumber : Hasanah & Untari 2020)

No	Simbol	Deskripsi
1.	 Status awal	Titik awal pada suatu diagram aktivitas.
2.	 Aktivitas	Merupakan proses atau aktivasi yang sedang dilakukan oleh sistem.
3.	 Percabangan / *decision*	Percabangan asosiasi dimana terdapat lebih dari satu pilihan kegiatan.
4.	Pengabungan / *join*	Menggabungkan aliran dari beberapa aktivitas menjadi satu.
5.	 Status akhir	Titik akhir pada suatu diagram aktivitas
6.	 *Swimlane*	Mengelompokkan aktivitas yang terjadi berdasarkan entitas misalnya aktor, dan system.

2.1.13 *Google Drive*

Google Drive, layanan penyimpanan milik *Google, Inc*. yang diluncurkan sejak April 2012, telah menjadi pilihan utama bagi pengguna yang membutuhkan fleksibilitas dan ketersediaan dalam menyimpan berbagai jenis file. Dengan kapasitas penyimpanan gratis sebesar 15 *Gigabyte* (GB), *Google Drive*

memungkinkan pengguna untuk mengakses file mereka dengan mudah di mana saja dan kapan saja, menggunakan berbagai perangkat. Dari foto dan video hingga dokumen teks, spreadsheet, dan presentasi, Google Drive menyediakan platform yang terintegrasi dengan berbagai layanan Google lainnya, memastikan pengalaman pengguna yang terhubung dan efisien dalam pengelolaan fil.. (Rio Trilaksono et al., 2018).

2.1.14 *Google Colaboratory*

Menurut (Gelar Guntara, 2023) *Google Colaboratory* (Colab) adalah wujud nyata dari platform *cloud* milik Google yang mengusung popularitas riset dalam bidang *machine learning*. Dengan Colab, pengguna dapat menjalankan pemrograman *Python* di server *Google* melalui internet, membuatnya sangat menguntungkan untuk pengembangan dan eksperimen machine learning. Didukung oleh *Jupyter Notebook*, Colab tidak hanya berfungsi seperti *Google Docs* dalam hal berbagi dan kolaborasi, tetapi juga menyediakan runtime Python 2 dan 3 yang sudah siap pakai, dilengkapi dengan berbagai pustaka utama seperti *TensorFlow*, *Matplotlib*, *Keras*, *Pytorch*, dan *scikit-learn.*

Selain itu, Colab menawarkan integrasi yang mulus dengan *Google Drive* sebagai media penyimpanan, serta mendukung penggunaan prosesor CPU, GPU, dan TPU. Dengan jaminan operasional server yang stabil, Colab menjadi pilihan ideal untuk menjalankan pemrosesan yang kompleks dan tidak akan mengalami kendala signifikan selama koneksi internet tetap lancar.

Gambar 2. 9 *Google Colaboratory*

2.1.15 *Python*

Menurut (Mambang et al., 2022) Python, bahasa pemrograman yang pertama kali muncul pada tahun 1991 oleh Guido van Rossum, terus berkembang di bawah naungan *Python Software Foundation* hingga saat ini. Keunikan *Python* terletak pada kemampuannya sebagai bahasa pemrograman interpretatif multiguna yang sangat mudah dibaca dan dipahami.

Selain itu, *Python* juga menjadi pilihan utama untuk pengembangan aplikasi *web*. Dengan menggunakan *framework* seperti *Django* dan *Flask*, *Python* mampu menangani pengembangan *backend* dengan mudah, memungkinkan pengembang untuk membangun aplikasi *web* yang kuat dan skalabel dengan cepat. Kemampuan *Python* untuk menyelaraskan antara kekuatan analisis data, machine learning, dan pengembangan *web* menjadikannya bahasa pemrograman yang sangat diminati dan relevan dalam dunia teknologi modern.

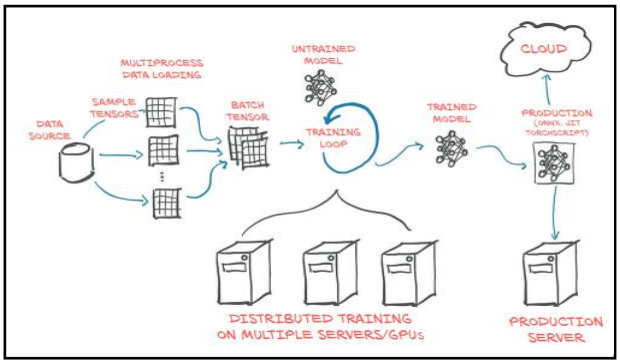
2.1.16 *PyTorch*

Menurut (Stevens et al., 2020) *PyTorch* adalah library program *Python* yang membantu membuat proyek *deep learning* dengan mudah. Hal ini menekankan fleksibilitas dan memungkinkan model *deep learning* diekspresikan dalam bahasa

Python. Aksesibilitas dan kemudahan penggunaan ini segera diadopsi oleh komunitas riset, dan pada tahun-tahun sejak rilis pertamanya, ini telah menjadi salah satu alat *deep learning* yang paling penting dalam banyak aplikasi. Seperti halnya *Python* untuk pemrograman, *PyTorch* adalah pengantar yang bagus untuk *deep learning*.

PyTorch menawarkan dua hal yang membuatnya sangat relevan untuk *deep learning*: pertama, ia menyediakan komputasi yang dipercepat menggunakan *Graphic Processing Unit* (GPU), sering kali menghasilkan percepatan dalam kisaran 50x lebih dari melakukan perhitungan yang sama pada *Central Processing Unit* (CPU). Kedua, *PyTorch* menyediakan fasilitas yang mendukung optimasi numerik pada ekspresi matematika generik, yang digunakan *deep learning* untuk pelatihan.

Modul inti *PyTorch* untuk membangun jaringan saraf terletak di *torch.nn*, yang menyediakan lapisan jaringan saraf umum dan komponen arsitektur lainnya. Komponen-komponen ini dapat digunakan untuk membangun dan menginisialisasi model yang tidak terlatih. Untuk melatih model memerlukan beberapa hal tambahan seperti: sumber data pelatihan, pengoptimal untuk menyesuaikan model dengan data pelatihan, dan cara untuk mendapatkan model dan data ke perangkat keras yang benar-benar akan melakukan perhitungan yang diperlukan untuk melatih model seseuai dengan gambar 2.10.



Gambar 2. 10 Struktur Proyek *Pytorch* (Sumber: Stevens 2020)

2.1.17 *Flask*

Menurut (Grinberg, 2018) *Flask* adalah seperti penyihir kecil dalam dunia pengembangan *web*, dikenal sebagai "*microframework*" yang menakjubkan dalam bahasa pemrograman *Python*. Meskipun sederhana, *Flask* memiliki keajaiban tersendiri dalam membangun aplikasi *web* dengan cepat dan efisien. Dibandingkan dengan kerangka kerja lain yang kompleks, *Flask* menawarkan pendekatan yang lebih ringkas tanpa kekacauan komponen yang membingungkan. Ini memungkinkan pengembang untuk menyesuaikan struktur aplikasi sesuai kebutuhan, baik untuk proyek sederhana maupun skala besar, menjadikan *Flask* pilihan ideal untuk membangun solusi *web* yang tangguh dan mudah dikelola.

2.1.17.1 Fitur-fitur *Flask*

Menurut (Irsyad, 2018) *Flask* adalah seperti bintang kecil dalam dunia pengembangan *web*, dikenal sebagai "*microframework*" yang ringkas namun sangat fleksibel. Dengan inti yang sederhana dan kecil, *Flask* mampu tumbuh dan berkembang sesuai kebutuhan proyek. Dengan demikian, beberapa fitur bawaan *Flask* termasuk sedikit jumlahnya, termasuk:

1. Memiliki *server* pengembangan terintegrasi.

2. *Debugger* yang cepat.
3. Mendukung pengetesan unit terintegrasi.
4. Kompatibel dengan mesin aplikasi *Google.*
5. Pengiriman permintaan *RESTful.*
6. *Templating Jinja2*.
7. Dukungan untuk *cookies* yang aman.
8. Berbasis *unicode*.
9. Mengikuti *Web Server Gateway*.
10. *Interface* (WSGI) 1.0.

Kelebihannya tidak hanya terletak pada strukturnya yang minimalis, tetapi juga didukung oleh dokumentasi yang sangat baik dan komunitas forum yang aktif di seluruh internet. Ini menjadikan *Flask* pilihan yang ideal untuk pengembang yang ingin membangun aplikasi *web* dengan cara yang efisien dan efektif.

2.1.17.2 Keunggulan dan Kekurangan *Flask*

Menurut (Irsyad, 2018) *Flask* memancarkan kelebihan dan kekurangan saat dibandingkan dengan *framework web Python* lainnya seperti *Django, CherryPy*, dan sebagainya. Sebagai pendekatan yang "*micro*", *Flask* menonjol dengan desain modular yang sederhana dan ringan, ideal untuk aplikasi yang berjalan cepat. Kemampuannya dalam mengelola permintaan HTTP dengan mudah dan menyediakan API yang konsisten membuatnya disukai oleh pengembang yang mengutamakan kejelasan dan kemudahan. Dukungan dokumentasi yang komprehensif dan mudah dipahami memperkuat daya tariknya, serta kemudahan pemasangan dan penyebaran untuk produksi yang tak tertandingi.

Namun, seperti pedang bermata dua, *Flask* juga memiliki kelemahan. Ketergantungannya pada aplikasi pihak ketiga untuk ORM dan lapisan *database* membatasi fungsionalitas bawaannya, sementara kurangnya dukungan *async* menunjukkan batasan dalam kesiapan untuk skala besar dan interaksi *real-time*. Meskipun demikian, fleksibilitas dan konfigurasi yang mudah tetap menjadikan *Flask* pilihan yang menarik bagi pengembang yang menghargai kontrol dan adaptabilitas dalam pembangunan aplikasi *web.*

2.1.18 *Android Studio*

Menurut (Prasetyo et al., 2023) *Android Studio* adalah IDE Android resmi, yang disajikan kepada publik pada tahun 2013 sebagai pratinjau dan akhirnya dirilis pada tahun 2014. *Android Studio* diimplementasikan sebagai plugin melalui *IntelliJ IDEA6*, IDE Java yang dibuat oleh *Jetbrains7*, perusahaan yang juga berada di belakang Kotlin. Jadi, seperti yang Anda lihat, semuanya terhubung erat. Gradle sekarang menjadi sistem build resmi untuk Android, membuka banyak peluang baru untuk pembuatan dan implementasi versi. Dua fungsi yang paling menonjol dari fungsi ini adalah membangun sistem dan rasa, yang memungkinkan Anda dengan mudah membuat versi aplikasi yang tidak terbatas (atau bahkan aplikasi yang berbeda) sambil menggunakan basis kode yang sama. Berikut logo android studio seperti gambar 2.11.

Gambar 2. 11 *Android Studio*

2.1.19 Bahasa Pemrograman *Kotlin*

Menurut (Prasetyo et al., 2023) sejak rilis IntelliJ 15, *plugin Kotlin* telah dimasukkan ke dalam IDE. Untuk *Android Studio*, ini berbeda tergantung pada versinya. Jika memiliki Android Studio 3.0 atau yang lebih baru. Sampai saat ini, 3.0 masih merupakan rilis *Canary*. Untuk menghindari *bug* yang tidak terduga dalam proses, disarankan untuk menggunakan versi stabil.

2.1.20 *Firebase*

Menurut (Khawas & Shah, 2018)Salah satu *platform Backend as a Service* (BaaS) yang dikembangkan oleh *Google* adalah *Firebase*. BaaS menawarkan berbagai layanan yang membantu dan mengelola aplikasi *mobile* dan *web* lebih mudah dan lebih efektif. Berikut layanan yang tersedia:

1. *Firebase Auth*

   Firebase Auth memungkinkan integrasi login sosial dengan *Facebook, Google, GitHub,* dan *Twitter,* memberikan fleksibilitas otentikasi langsung dari sisi klien. Layanan berbayar ini tidak hanya menyederhanakan proses autentikasi dengan kode, tetapi juga menyertakan sistem manajemen pengguna. Pengembang dapat

mengaktifkan otentikasi melalui email dan kata sandi yang tersimpan di Firebase, memastikan keamanan dan kenyamanan dalam mengelola akses pengguna.

2. *Realtime Database*

   Firebase menawarkan fitur seperti *backend* dan *basis data real-time*, serta API untuk pengembang aplikasi yang memungkinkan data aplikasi disinkronkan dengan klien dan disimpan di awan. Pustaka klien, yang tersedia untuk aplikasi berbasis *Android, iOS,* dan *JavaScript*, memungkinkan integrasi dengan aplikasi.

3. *Firebase Notifications*

   Memungkinkan pengembang aplikasi seluler dan layanan ini mengakses pemberitahuan pengguna yang ditargetkan secara gratis.

2.2 Kajian Penelitian Terdahulu

Dibawah ini adalah beberapa hasil penelitian yang relevan yang dijadikan acuan riset ini, yang terangkum dalam tabel 2.3. berikut.

Tabel 2. 3 Penelitian Terkait

No.	Penulis	Judul	Metode	Publikasi
1	Amanda Caecilia Milano, Achmad Yasid dan Rima Tri Wahyuningrum	Klasifikasi Penyakit Daun Padi Menggunakan Model Deep Learning Efficientnet-B6	*Convolutional Neural Network* (CNN)	JITET (Jurnal Informatika dan Teknik Elektro Terapan) Vol. 12 No. 1, pISSN: 2303-

				0577 eISSN: 2830-7062
	Hasil : Dalam eksperimen ini, penggunaan *Convolutional Neural Network* (CNN) dengan arsitektur *EfficientNet-B6* menonjol dalam skenario 2. Dengan menggunakan *input size* 224 dan menjalankan pelatihan selama 50 *epoch*, hasilnya mengungkapkan pencapaian tertinggi pada fold kelima dengan akurasi mencapai 77.05%. Presisi dan *recall* juga mencapai 77.11% dan 77.05% secara berturut-turut, sementara *f1 score* mencapai 76.29%. Secara keseluruhan, nilai *Area Under the Curve* (AUC) untuk skenario 2 menunjukkan konsistensi yang optimal, dengan rentang antara 0.90 hingga 0.93 untuk setiap foldnya.			
	Keterbatasan Penelitian : Penelitian ini perlu memperhatikan metode validasi model yang lebih komprehensif untuk memastikan keandalan dan generalisasi model yang dikembangkan.			
2.	Afis Julianto, Andi Sunyoto, dan Ferry Wahyu Wibowo	Optimasi Hyperparameter Convolutional Neural Network Untuk Klasifikasi Penyakit Tanaman Padi	*Convolutional Neural Network* (CNN)	TEKNIMEDIA - Volume 3, Nomor 2, Desember 2022: 98 – 105
	Hasil :			

	Dalam eksperimen ini, penggunaan *Convolutional Neural Network* (CNN) dengan arsitektur *MobileNet-V2* menunjukkan bahwa penentuan *hyperparameter* yang optimal sangat mempengaruhi performa model. Dataset yang digunakan terdiri dari 3 kelas penyakit pada daun tanaman padi: blast, blight, dan tungro. Dengan menggunakan *hyperparameter* seperti jumlah *epoch* 100, *batch size* 32, *learning rate* 0.001, dan *optimizer RMSProp*, hasilnya menunjukkan performa yang sangat baik. Model mencapai akurasi 97.56%, presisi 97.64%, *recall* 97.57%, dan *f1-score* 97.57%, menggarisbawahi pentingnya pengaturan *hyperparameter* yang tepat dalam pengembangan model CNN untuk klasifikasi penyakit tanaman.			
	Keterbatasan Penelitian : Keterbatasan dalam variasi penyakit yang dipelajari hanya menggunakan tiga kelas penyakit daun padi yaitu blast, blight dan tungro. Hal ini dapat membatasi kemampuan model untuk mengenali penyakit yang lebih kompleks atau langka pada tanaman padi.			
3	Endang Anggiratih, Sri Siswanti, Saly Kurnia Octaviani dan Arumsari	Klasifikasi Penyakit Tanaman Padi Menggunakan Model Deep Learning Efficientnet B3 dengan Transfer Learning	*Convolutional Neural Network* (CNN)	Jurnal Ilmiah Sinus (JIS) Vol : 19, No. 1, Januari 2021 ISSN (Print) : 1693-1173, ISSN (Online): 2548-4028

	Hasil : Penelitian ini bertujuan untuk mengembangkan model *Deep Learning* dengan menggunakan arsitektur *EfficientNet B3* dan *Transfer Learning* untuk mengidentifikasi penyakit pada tanaman padi, khususnya *brown spot* dan *bacterial leaf*. Hasil penelitian menunjukkan bahwa model yang dikembangkan mampu mengklasifikasi penyakit dengan akurasi tinggi. Penggunaan *EfficientNet B3* dengan *Transfer Learning* berhasil mencapai akurasi sebesar 79.53% dengan nilai *loss/error* sebesar 0.012, menegaskan efektivitas pendekatan ini dalam meningkatkan kemampuan model dalam mengenali kondisi kesehatan tanaman padi secara presisi dan efisien..
	Keterbatasan Penelitian : Kelemahan dan keterbatasan penelitian ini masih perlu dipertimbangkan. Salah satunya berkaitan dengan pengembangan model yang menggunakan arsitektur *Efficientnet B3* dengan *Transfer Learning*. Penelitian tambahan diperlukan untuk meningkatkan kinerja model yang telah dikembangkan.