# KLASIFIKASI PENYAKIT DAUN JAGUNG MENGGUNAKAN *CONVOLUTIONAL NEURAL NETWORK* (VGG16)

Tedi Maulana[1], Khurotul Aeni [2], Nurul Mega Saraswati[3]
[123]Program Studi Informatika, Fakultas Sains dan Teknologi, Universitas Peradaban
tedim765@gmail.com, khaeni988@gmail.com, nurul.mega.s@gmail.com
Jl. Raya Pagojengan KM.3, Paguyangan, Kabupaten Brebes, Jawa Tengah 52276

***Keywords:***

*Convolutional Neural Network (CNN), Corn Leaves, Detection, Classification, Visual Geometry Group 16*

**Abstract**

*Corn is the second most important food crop in Indonesia after rice, serving as a strategic source of carbohydrates, animal feed, and industrial raw materials. Its productivity is often disrupted by pests and diseases, which can cause significant losses for farmers. Technology-based solutions are needed to detect diseases early. This study developed a corn leaf disease detection system using the Convolutional Neural Network (CNN) method with the Visual Geometry Group 16 architecture through a digital image processing approach. The corn leaf image dataset was classified into several disease categories. The VGG16 model achieved an accuracy rate of 93%, a precision value of 90%, a recall of 92%, and an F1-score of 91% in the classification process. The results of this study are expected to help farmers detect diseases early and improve the effectiveness of sustainable pest control.*

***Kata Kunci:***

*Convolutional Neural Network* (CNN), Daun Jagung, Deteksi, Kelasifikasi, *Visual Geometry Group 16*

**Abstrak**

Jagung merupakan tanaman pangan penting kedua setelah padi di Indonesia yang memiliki peran strategis sebagai sumber karbohidrat, pakan ternak, dan bahan baku industri. Produktivitasnya sering terganggu oleh serangan hama dan penyakit yang dapat menyebabkan kerugian besar bagi petani. Dibutuhkan solusi berbasis teknologi untuk mendeteksi penyakit secara dini. Penelitian ini mengembangkan sistem deteksi penyakit daun jagung menggunakan metode *Convolutional Neural Network* (CNN) dengan arsitektur *Visual Geometry Group* 16 melalui pendekatan pengolahan citra digital. Dataset citra daun jagung diklasifikasikan ke dalam beberapa kategori penyakit. Model VGG16 berhasil mencapai tingkat akurasi sebesar 93%, nilai *precision* 90%, *recall* 92%, dan *F1-score* sebesar 91% dalam proses klasifikasi. Hasil penelitian ini diharapkan dapat membantu petani dalam mendeteksi penyakit lebih awal dan meningkatkan efektivitas pengendalian hama secara berkelanjutan.

## I. Pendahuluan

Jagung merupakan tanaman serealia dan bahan pangan penting. Setelah beras, jagung merupakan tanaman pangan terpenting di Indonesia. Tanaman ini menghasilkan banyak hasil dan dapat dimanfaatkan untuk berbagai keperluan. Tanaman ini secara strategis penting bagi perekonomian negara dan memiliki berbagai tujuan, termasuk pakan ternak dan sebagai pengganti beras sebagai makanan pokok. Jagung juga merupakan sumber daya industri yang layak[1]. Tanaman jagung dapat tumbuh dengan baik hampir diseluruh daerah di Indonesia, termasuk di daerah Bantarkawung sebagian petani menanam jagung juga selain menanam padi. Iklim yang cukup lembab sehingga mempercepat pertumbuhan hama yang meningkatkan penyebaran hama dari satu pohon ke pohon lain. Penyakit atau hama pada tanaman jagung adalah masalah yang tidak di inginkan oleh petani yang dapat menyebabkan kegagalan panen, mengakibatakan kerugian banyak bagi para petani dan mengurangi kualitas karena banyak buah jagung yang menghitam dan membusuk. Gejala tersebut

umumnya karena adanya hama pada tanaman jagung seringkali menunjukan beberapa gejala penyakit yang diderita tanaman tapi sering diabaikan karena ketidak pahaman dan keterbatasannya menganggap itu hal yang lumrah pada masa tanam, sampai gejala tersebut semakin meluas dan tidak bisa dikendalikan.

Banyak pakar pertanian, terutama di industri teknologi, telah mencari cara untuk mengatasi gejala-gejala ini. Salah satunya adalah mengidentifikasi bercak-bercak pada daun jagung menggunakan pemrosesan citra untuk menemukan penyakit pada tanaman jagung. Menurut data Badan Pusat Statistik (BPS), produksi jagung mencapai 2,17 juta ton pada tahun 2023 dan diperkirakan akan mencapai 2,59 juta ton pada tahun 2024. Angka ini kemungkinan akan menurun akibat buruknya pengelolaan hama. Strategi pengendalian hama yang efektif memerlukan penggunaan teknologi, seperti yang menggunakan pemrosesan gambar digital untuk mendeteksi penyakit pada tanaman jagung atau pendekatan Jaringan Saraf Konvolusional dengan arsitektur VGG16 untuk memproses gambar objek tanaman jagung.

*Multi Layer Perceptron* (MLP) melahirkan *Convolutional Neural Network* (CNN), sebuah teknik pembelajaran mesin yang dirancang khusus untuk menangani input dua dimensi, seperti gambar. Karena banyaknya lapisan jaringan yang rumit, CNN terkadang dikategorikan sebagai jaringan saraf dalam. VGG16 (*Visual Geometry Group 16*), sebuah arsitektur CNN yang populer untuk tugas identifikasi objek, memiliki 16 lapisan 1 pengklasifikasi, 2 lapisan tertaut penuh, dan 13 lapisan *konvolusional*. Keunggulan utama VGG16 adalah penggunaan 2x2 *max pooling* dan 3x3 filter di setiap segmen, yang telah terbukti menghasilkan akurasi yang lebih tinggi daripada arsitektur CNN sebelumnya. Bab selanjutnya akan memberikan penjelasan yang lebih menyeluruh tentang topik ini[2].

Penelitian klasifikasi terdahulu sering menggunakan metode *Support Vector Machine* (SVM) dan *Convolutional Neural Network* (CNN) untuk klasifikasi buah algoritma SVM menghasilkan akurasi sebesar 93,09%, sedangkan algoritma CNN memberikan akurasi sebesar 96,87% [3]. Dengan tingkat akurasi sebesar 87%, pendekatan KNN merupakan yang paling efektif dari kelima algoritma yang digunakan dalam penelitian deteksi penyakit daun tanaman padi dengan memanfaatkan algoritma *Decision Tree, Random Forest, Naïve Bayes*, SVM, dan KNN[4]. Selidiki klasifikasi penyakit daun tomat menggunakan fitur warna dan algoritma *Decision Tree*. Algoritma *Decision Tree* memiliki tingkat akurasi sebesar 78%[5]. Kemudian kinerja dua algoritma CNN, VGG16 dan VGG19, dibandingkan. Untuk klasifikasi *varietas* padi, algoritma VGG16 menghasilkan hasil dengan akurasi sebesar 98% setelah pelatihan selama 73,405 detik, sedangkan algoritma VGG19 menghasilkan hasil dengan akurasi sebesar 97% setelah pelatihan selama 78,098 detik[6].

Berdasarkan studi sebelumnya, pendekatan jaringan saraf tiruan *Konvolusional* (CNN), yang berfokus pada arsitektur VGG16, berkinerja baik dalam tugas klasifikasi dengan tingkat akurasi 98% dan periode pelatihan yang lebih singkat. Pendekatan jaringan saraf tiruan *konvolusional* (CNN), dengan fokus pada arsitektur VGG16, sangat tepat dan bermanfaat untuk digunakan dalam studi yang disarankan. Pendekatan jaringan saraf tiruan *Konvolusional* (CNN) akan digunakan dalam studi ini, dengan fokus pada arsitektur VGG16, untuk membantu petani mendeteksi hama pada tanaman jagung dengan teknologi otomatis dan presisi. Hal ini akan memungkinkan untuk mengambil tindakan lebih cepat dalam mengatasi hama.

Penelitian dilakukan dengan menggunakan latar belakang masalah yang dijelaskan sebelumnya sebagai dasar dengani judul "Klasifikasi Penyakit Daun Jagung Menggunakan *Convolutional Neural Network* (Vgg16)". Menekankan arsitektur VGG16, untuk membantu petani mendeteksi hama pada tanaman jagung dengan teknologi otomatis dan presisi. Hal ini akan memungkinkan untuk mengambil tindakan lebih cepat dalam mengatasi hama.

## II. Landasan Teori

### 1. *Deep Learning*

Salah satu aspek kecerdasan buatan yang pertama kali terinspirasi oleh arsitektur manusia adalah pembelajaran mendalam. Komputer dapat belajar berkat teknologi ini, yang memungkinkan

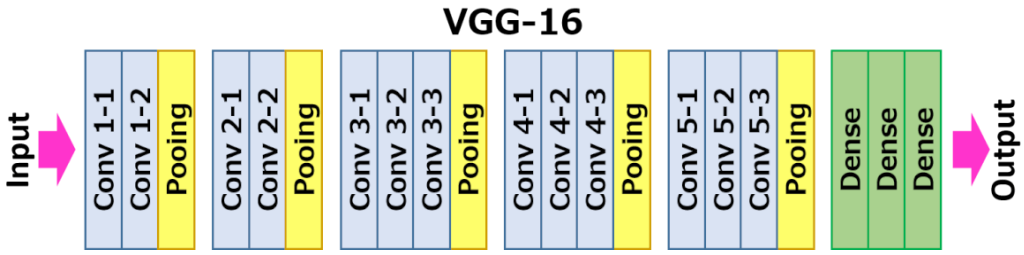
mereka untuk secara akurat mewakili data yang mereka lihat. Teks, gambar, dan video akan dikenali dengan benar oleh sistem sebagai hal yang terpisah. Beberapa jaringan saraf digunakan untuk memproses data dalam algoritma pembelajaran mendalam. Data yang telah disederhanakan kemudian dikirim ke lapisan berikutnya oleh jaringan saraf. Sementara algoritma pembelajaran mesin sering berkinerja baik pada data terstruktur dengan banyak baris dan kolom, mereka sering menghadapi kesulitan atau menjadi tidak layak ketika menangani data yang tidak terstruktur, seperti foto yang telah diproses sebelumnya. Untuk menganalisis dan mengedit gambar, algoritma pembelajaran mendalam telah dikembangkan. Algoritma ini menggunakan lapisan yang dibangun secara khusus. Lapisan selanjutnya akan menggabungkan aspek-aspek yang diambil dari tingkat sebelumnya untuk membuat representasi atau konfigurasi yang lebih canggih, sedangkan lapisan awal akan mendeteksi fitur-fitur halus dalam gambar[7]. Salah satu algoritma *Deep Learning* adalah *Convolutional Neural Network* (CNN).

## 2. *Convolutional Neural Network*

Dalam pembelajaran mendalam, jaringan saraf *konvolusional*, atau CNN, merupakan teknik yang paling banyak digunakan. *GoogleNet*, *MobileNet*, *ResNet*, *AlexNet*, *VGG*, dan desain lainnya merupakan beberapa contoh CNN. *Multilayer Perceptron* (MLP), yang diciptakan khusus untuk menangani masukan dua dimensi, menjadi cikal bakal CNN. Karena arsitekturnya yang lebih rumit, CNN termasuk dalam kategori jaringan saraf dalam dan sering digunakan untuk analisis data gambar. Jaringan saraf *konvolusional* (CNN), salah satu jenis arsitektur jaringan saraf tiruan, diciptakan khusus untuk memproses masukan seperti gambar dan video[8]. *CNN* menerima input berupa gambar dan melalui serangkaian lapisan *convulation layer*, *pooling layer* dan *fully connected layer*, *CNN* dapat mempelajari representasi fitur yang signifikan dalam gambar.

## 3. *VGG16*

VGG16 terdiri dari total 16 lapisan, termasuk 13 lapisan *konvolusional* dan 3 lapisan terhubung penuh. VGG16 telah berhasil mencapai tingkat akurasi yang tinggi dalam pengenalan gambar pada berbagai tugas, termasuk dataset *ImageNet*, yang terdiri dari jutaan gambar dari ribuan kelas berbeda. Lapisan keluaran menggunakan fungsi aktivasi *sigmoid* jika terdapat 2 kategori atau kurang, dan fungsi aktivasi *softmax* jika terdapat 3 kategori atau lebih dalam dataset. Arsitekturnya seperti terlihat pada gambar 1.




Gambar 1 *Visualisai VGG16*

a. *Convolutional Layer*

*Convolutional layer* merupakan bagian utama dalam arsitektur CNN yang berfungsi melakukan operasi konvolusi terhadap *output* dari layer sebelumnya. Proses ini menggunakan filter untuk menghasilkan *feature map* dari *input* yang diberikan. Lapisan ini berperan penting dalam membentuk karakteristik jaringan. Selain itu, ukuran output yang dihasilkan dapat dikendalikan melalui penggunaan hyperparameter pada setiap lapisannya. Hasil oleh *layer convolutional* akan di teruskan ke *layer polling* untuk di kurangi dimensinya.

1) *Stride*

Parameter langkah pada lapisan konvolusi mengontrol seberapa besar pergeseran filter yang terjadi pada lapisan tersebut. Waktu komputasi yang lebih singkat tetapi informasi yang

diperoleh lebih sedikit ketika parameter langkah ditingkatkan karena menyebabkan pergeseran filter yang lebih besar.

2) *Padding*

*Padding* digunakan untuk memperbesar dimensi dari *convolution layer*. Tujuan dari penggunaan *padding* adalah supaya tiap pixel dari *convolution layer* akan dikunjungi dalam jumlah yang sama. Pengguanan *padding* dapat mengurangi resiko *over-fiiting* dikarenakan seluruh *pixel* dikunjungi secara merata, namun *output* dari *padding* juga akan menghasilkan *feature map* yang lebih besar. Rumus *Padding* dapat dilihat pada rumus 1.

$$Output = \frac{W - N + 2P}{S} \qquad 1.1.1.(1)$$

Dimana:

W = Panjang/Tinggi Input
N = Panjang/Tinggi Filter
P = Nilai Padding
S = Ukuran pergeseran (Stride)

b. *Pooling Layer*

*Pooling* adalah proses untuk mereduksi dimensi matriks fitur dengan menerapkan operasi tertentu, biasanya setelah lapisan konvolusi. Lapisan ini menggunakan filter berukuran dan langkah (stride) tertentu yang bergerak melintasi *feature map*. Umumnya, digunakan filter 2x2 dengan stride 2, yang bekerja pada tiap bagian dari input untuk menghasilkan representasi yang lebih ringkas.

Penggabungan digunakan untuk meminimalkan dimensionalitas lapisan konvolusi keluaran, yang sering dikenal sebagai "peta fitur", untuk mempercepat proses pelatihan model dan mengurangi jumlah parameter yang perlu disesuaikan.

c. *Fully-Conected Layer*

*Fully-Conected (FC) Layer* merupakan layer yang bertugas untuk mengklasifikasikan data berdasarkan *output* yang di hasilakan dari *convoluation* dan *pooling layer*. *FC layer* menerima input 1D *array* sehingga *output* dari *layer* sebelumnya harus diubah ke dalam bentuk 1D *array* terlebih dahulu. Perbedaan utama antara layer konvolusi dan *fully connected* terletak pada pola koneksi neuronnya. Pada layer konvolusi, setiap *neuron* hanya terhubung ke area tertentu dari *input* (*receptive field*), sedangkan pada *fully connected*, setiap *neuron* terhubung ke seluruh *input*. Meski demikian, keduanya tetap menggunakan operasi *dot product*, sehingga fungsi dasarnya tetap serupa.

Berikut adalah fungsi yang ada di dalam neuron pada dense layer pada rumus 2.

$$Output = x.W + b \qquad (2)$$

Dimana:

*W* = *Weight*
*x* = *Input dari neuron sebelumnya*
*b* = *Bias*

Ada beberapa *activation function* pada *FC layer*, 3 diantaranya yang sering digunakan adalah *Sigmoid, Softmax,* dan *ReLu*.

1. ***Sigmoid***

*Sigmoid* memiliki fungsi mengubah *logits* menjadi 0 dan 1, sehingga s*igmoid* cocok untuk mengklasifikasikan dataset binary.

Rumus *Sigmoid* dapat dilihat pada rumus 3.

$$S(x)^n = \frac{1}{1 + e^{-x}} \quad (3)$$

Dimana:

- *S(x)* adalah nilai *sigmoid* dari *input x*.
- Fungsi *sigmoid* mengubah input *x* menjadi nilai antara 0 dan 1.
- Memangkatkan *S(x)* dengan *n* akan menghasilkan kurva yang lebih "tajam" atau lebih "landai" tergantung nilai n.

**2. *Softmax***

*Softmax* dapat digunakan sebagai fungsi aktivasi untuk kumpulan data dengan lebih dari tiga kategori karena dapat mengubah *logit* menjadi tiga keluaran. Selain itu, *Softmax* menghasilkan *probabilitas* dengan jumlah 1.

Rumus *Softmax* dapat dilihat pada rumus 4.

$$S(x_i) = \frac{e^{xi}}{\Sigma_{j=1}^{n} + e^{xj}} \quad (4)$$

Dimana:

- $S(x_i)$ = Nilai softmax dari elemen ke-i.
- $x_i$ = Elemen ke-i dari vektor input $x = (x_1, x_2, \ldots, x_n)$.
- $e$ = Bilangan Euler (~2.718), basis dari logaritma natural.
- $\Sigma_{j=1}^{n}$ = Jumlah dari eksponensial seluruh elemen dalam vektor x.

**3. *ReLu***

*Rectified Linear Unit* adalah *activation function* yang populer digunakan. *ReLu* bisa digunakan pada *convolution layer* ataupun *FC layer*. *ReLu* membatasi *output negatif* menjadi 0 sehingga *ReLu* hanya mengeluarkan *output positif*. Fungsi *ReLu* selain mudah dipahami juga cepat dalam peforma sehingga banyak dipilih dalam penelitian model *AI*.

Rumus *ReLu* dapat dilihat pada rumus 5.

$$f(x) = max(0,x) \quad (5)$$

Dimana:

- *f(x)* : fungsi aktivasi *ReLu*
- *x* : input ke neuron dalam jaringan saraf

.

**4. *Confusion Matrix***

Salah satu alat untuk mengevaluasi dan menilai efektivitas model klasifikasi dalam penambangan data adalah matriks konfusi. Matriks ini menawarkan ringkasan menyeluruh tentang kemampuan model untuk meramalkan kelas atau kategori dari sekumpulan data uji tertentu. Prediksi *True Positive* (TP) dan *True Negative* (TN) yang akurat berfungsi sebagai dasar pengukuran dalam matriks konfusi. Ketika prediksi positif terbukti salah, prediksi tersebut disebut *False Positive* (FP), dan ketika prediksi negatif terbukti salah, prediksi tersebut disebut *False Negative* (FN). Akurasi, presisi, *recall*, dan *F1 score* membentuk matriks tersebut[9]. *Confusion matrix* seperti pada Tabel 1.

Tabel 1 *Confusion Matrix*

| **Label** | **Terklasifikasi Positif** | **Terklasifikasi Negatif** |
|---|---|---|
| Positif | *True Positive* | *False Negative* |
| Negatif | *False Positive* | *True Negative* |

Keterangan Tabel 1 yaitu:

1) *True Positive* (TP) Hasil tes yang benar-benar menunjukkan keberadaan suatu kondisi atau fakta.
2) *False Positive* (FP) Hasil tes yang secara keliru menunjukkan adanya suatu kondisi atau fakta, padahal sebenarnya tidak.
3) *True Negative* (TN) adalah hasil tes yang sebenarnya menunjukkan bahwa suatu kondisi atau fakta tidak ada.
4) *False Negative* (FN) Hasil tes yang secara keliru menunjukkan bahwa suatu kondisi atau fakta tidak ada, padahal sebenarnya ada.

*Confusion matrix* digunakan untuk menghitung nilai *accuracy, precision, recall, dan F1-score.* Metode evaluasi ini sangat baik jiga digunakan untuk menilai kinerja classifer atau algoritma machine learning dalam melakukan sebuah prediksi. Perhitungan metode evaluasi ini dilakukan dengan memanfaatkan *Confusion matrix.*

1) *Accuracy*

Akurasi dapat dihitung dengan membandingkan Perbandingan antara jumlah klasifikasi yang benar dan total klasifikasi yang dilakukan. Klasifikasi tepat dianggap sebagai *Positif* Asli untuk setiap jenis atau kelas. Rumus matematis *accuracy* dapat dilihat pada rumus 6.

$$Accuracy = \frac{TP}{Total\ Seluruh\ Data} \quad (6)$$

2) *Precision*

*Precision* mengukur sejauh mana prediksi *positif* yang dilakukan oleh model benar-benar sesuai dengan kelas yang dimaksud, dan rumus matematis dapat dilihat dalam rumus 7.

$$Precision = \frac{TP}{(TP + FP)} \quad (7)$$

3) *Recall*

*Recall* mengukur rasio prediksi *true positif* dibandingkan dengan keseluruhan data *true positif* dengan nilai mendekati 1. Penulisan kembali matematis, seperti yang ditunjukkan dalam rumus 8.

$$Recall = \frac{TP}{(TP + FN)} \quad (8)$$

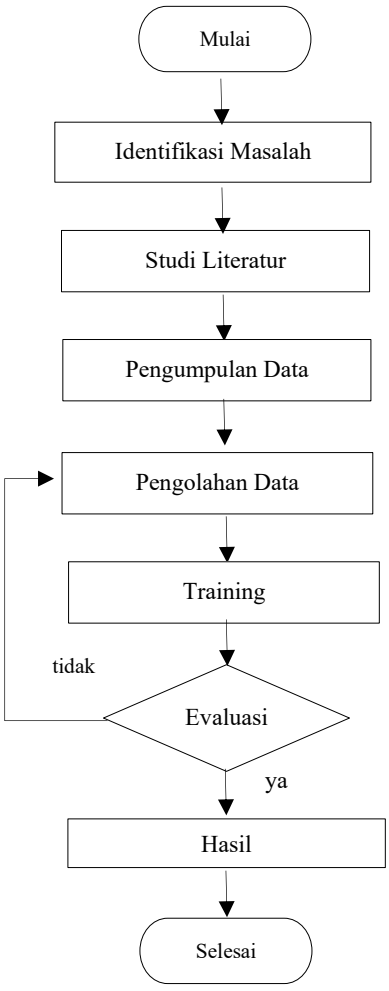
4) *F1-Score*

Merupakan hasil dari perhitungan rata-rata antara *precision* dan *recall*, yang berfungsi sebagai metrik evaluasi gabungan untuk menilai keseimbangan antara keduanya. Berikut penulisan matematis pada rumus 9.

$$F1 - Score \ = 2\,x\ \frac{precision\ x\ recall}{recall + precision} \quad (9)$$

## III. Metode

### 1. Tahapan Penelitian




Gambar 1 Tahap Penelitian

1) Identifikasi Masalah

Menemukan permasalahan penelitian merupakan langkah pertama dalam proses ini. Permasalahan yang teridentifikasi adalah keberadaan penyakit pada tanaman jagung, yang perlu dideteksi. Tujuan dari penelitian ini adalah untuk mengetahui seberapa baik algoritma CNN menggunakan arsitektur VGG16 dalam mengidentifikasi penyakit pada daun jagung.

2) Studi Literatur

Memahami jurnal dan website terkait metode dan masalah terkait penalitian, manfaat dari studi literatur yang didapat peneliti yaitu memeperdalam dan memperkuat pemahaman terhadap konsep dan teori terkait penelitian tentang metode CNN dengan aristektur *VGG16* yang akan diterapkan peneliti dalam mendeteksi hama pada tanaman jagung.

3) Pengumpulan Data

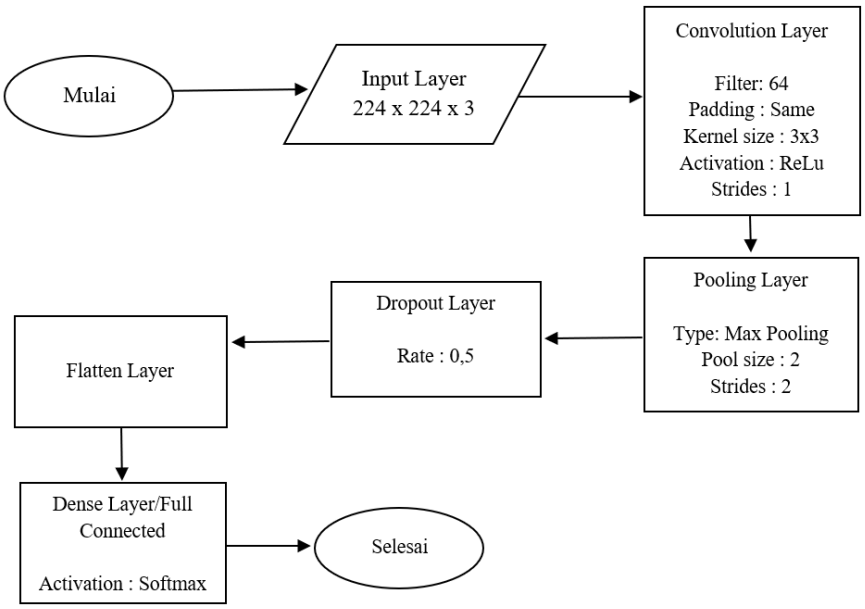
Salah satu tahap penting dalam proses penelitian yang akan digunakan untuk menemukan berbagai jenis penyakit pada tanaman jagung adalah tahap pengumpulan data. Penelitian ini dimana *dataset* penyakit tanaman jagung diperoleh dari *kaggle*. Fokus utama dari penelitian ini untuk mengklasifikasikan citra ke dalam kategori terdampak penyakit dan kategori kondisi sehat. Hasil yang diharapkan dapat diperoleh data yang konferhensif dari kondisi kesehatan tanaman jagung. Pengumpulan data juga perlu memperhatikan kelas yang sesuai dengan penelitian, pengolahan data diperlukan juga untuk memastikan data sesuai dengan kriteria supaya hasilnya akurat.

4) Pengolahan Data

Tahap ini merupakan tahap persiapan klasifikasi data. Pembagian setiap data pada jenis penyakit dibagi menjadi data *training*, *testing* dan *validation*. Data dibagi dengan perbandingan tertentu untuk mengetahui model terbaik dalam pelatihan. Data *training* digunakan untuk menjalankan *CNN* dalam pencarian model yang terbaik, Data validasi digunakan untuk menunjukkan seberapa baik performa model selama pelatihan, sementara data uji digunakan untuk menguji model yang dibuat selama fase pelatihan.

5) Training Model

Tahapan selanjutnya yaitu training model menggunakan *VGG16* untuk klassifikasi penyakit pada tanaman jagung yang telah dirancang. Dilakukan pengkodean model menggunakan bahasa pemerograman python. Pengkodean ini berupa model *CNN* dengan arsitektur *VGG16*. Arsitektur algoritma CNN adalah VGG16. 13 lapisan *konvolusional* dan 3 lapisan tertaut penuh membentuk 16 lapisan yang membentuk VGG16. Berikut Gambar 2. Proses Permodelan algoroitma *CNN* dengan Arsitektur *VGG16.*




Gambar 2. Permodelan Algoritma VGG16

Berikut penjelasan dari gambar 2 alur permodelan VGG16:

1) *Input Layer*

Citra dengan dimensi 224 x 224 x 3. Kemudian data itu diproses kedalam *input layer* dengan nilai pixel dan channel RGB.

2) *Convolutionl Layer*

Arsitektur CNN mencakup lapisan *konvolusional*. Keluaran dari lapisan sebelumnya mengalami operasi konvolusi pada langkah ini. Arsitektur jaringan CNN sebagian besar dibentuk oleh lapisan ini. Penerapan berulang suatu fungsi kekeluaran fungsi lain dikenal sebagai konvolusi dalam matematika. Setelah lapisan ini, masukan dikirim ke filter, yang menghasilkan peta masa depan sebagai keluaran. *Hyperparameter* juga dapat digunakan untuk menentukan volume keluaran untuk setiap lapisan.[1]. Tahapan ini akan memanfaatkan input yang telah diperoleh sebelumnya untuk melakukan operasi konvolusi. Konvolusi dalam tahap ini akan menggunakan 64 filter dengan ukuran karnel 3x3, serta menerapkan *padding* 'same' (0) untuk memastikan ukuran *output* tetap sama dengan ukuran *input*. Selain itu, fungsi aktivasi yang digunakan adalah *ReLu*, dan pergeseran (stride) yang diterapkan adalah 1. Dengan pengaturan ini, konvolusi akan mengoptimalkan proses ekstraksi fitur dari *input*, memungkinkan model untuk menangkap pola-pola penting yang ada dalam citra secara efisien. Proses ini merupakan langkah kunci dalam mendeteksi dan mengklasifikasikan informasi yang relevan dari citra untuk mencapai akurasi yang lebih baik dalam aplikasi yang kembangkan. *Output* yang dihasilkan oleh *convolutional layer* akan di teruskan ke *polling layer* untuk di kurangi dimensinya.

3) *Pooling Layer*

*Pooling* adalah mengurangi ukuran matriks dengan menggunakan operasi *pooling*. Lapisan penyatuan biasanya muncul setelah konvolusi. Pada dasarnya, *pooling layer* terdiri dari filter dengan ukuran dan langkah tertentu yang akan berpindah secara bergantian di seluruh area peta fitur. Dengan Menerapkan filter 2x2 dalam dua tahap, bekerja pada setiap irisan masukan. Untuk mengurangi jumlah parameter yang dibutuhkan dan mempercepat proses pelatihan model, penggabungan digunakan untuk meminimalkan dimensionalitas lapisan konvolusi

keluaran, yang juga disebut sebagai peta fitur. Prosedur ini mempertahankan aspek data yang paling signifikan sekaligus membantu mengurangi kompleksitas komputasi.

4) *Dropout*

Dengan menghindari *overfitting*, lapisan *dropout* dapat digunakan untuk meningkatkan kinerja model. Lapisan *dropout* ditambahkan, yang secara acak memilih *neuron* dan menetapkan nilainya ke nol. Selama pelatihan, *neuron* yang telah dinolkan ini tidak digunakan maupun dihilangka[10].

5) *Flatten Layer*

*Flatten* merupakan tahapan yang mengubah matriks multi-dimensi menjadi vektor atau matriks 1 dimensi. Konteks jaringan saraf tiruan *flatten* digunakan untuk meratakan (flattening) *output* dari *layer konvolusional* atau *pooling* agar bisa dihubungkan ke *fully connected layer* (dense layer).

6) *Fully-Conected Layer*

*Output* dari *Fletten* tadi akan masuk *Fully-Conected (FC) Layer* merupakan layer yang bertugas untuk mengklasifikasikan data berdasarkan *output* yang di hasilakan dari *convoluation* dan *pooling layer*. *FC layer* menerima *input 1D array* sehingga *output* dari *layer* sebelumnya harus diubah ke dalam bentuk *1D array* terlebih dahulu. *Konvolusional* biasa karena neuron pada layer terhubung penuh, sedangkan neuron pada layer *konvolusional* biasa hanya terhubung ke area tertentu dari input. Namun, karena kedua larik masih menggunakan perkalian titik (dot product), perbedaan di antara keduanya tidak terlalu signifikan. Tahap aktivasi *softmax* dilakukan untuk memastikan nilai yang diharapkan setelah perhitungan koneksi penuh selesai.

7) Evaluasi

Proses testing merupakan tahapan penelitian sebagai proses uji dalam mengetahui hasil evalusi kinerja metode dalam klasifikasi penyakit, menguji akurasi model menggunakan *confusion matrix*. *Confusion matrix* adalah melatih model pada dataset tentu melalui proses pralatih dan penyetelan lanjut, serta evaluasi kinerja model untuk pemahaman terhadap data berupa citra digital atau gambar menggunakan bahasa pemrograman *python*. Proses evaluasi telah mencapai akurasi yang telah sesuai dan tepat dalam mendeteksi hama maka akan berlanjut ke proses selanjutnya, Jika proses evaluasi tidak mencapai akurasi yang sesuai dan kurang tepat dalam mendeteksi hama maka akan kembali ke proses pengolahan data.

8) Hasil

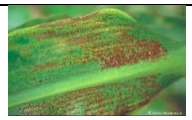
Tahapan ini merupakan tahap terakhir penelitian yang telah mencapai akhir sesuai yang diharapkan peneliti dan dapat diimplementasikan kedalam program menggunakan *GUI* yang menampilkan tingkat akurasi model *VGG16* dalam mendeteksi hama atau penyakit pada tanaman Jagung.

## IV. Hasil dan Pembahasan

### 1. Pengumpulan Data

Hasil dari pengumpulan data berupa citra hama pada daun tanaman jagung. Dataset yang digunakan berjumlah 4188 gambar hama atau penyakit pada daun tanaman jagung dimana data di ambil dari kaggle. Data terbagi menjadi 4 kelas penyakit daun tanaman jagung yaitu 3 jenis hama, hawar daun (blight), bercak daun (gray leaf spot) dan karat daun (common rust) dan daun sehat (healthy) pada daun tanaman jagung. Data asli di ambil menggunakan HP dengan kamera 12 Mega Pixsel. Untuk data sehat mendapatkan ukuran 2067x2023 dan 2683x 4000 pixel dan data terdampak penyakit mendapatkan ukuran asli 3000x 4000 dan 4000x3000. Dataset dapat dilihat pada tabel 1, detail dataset bisa di lihat dilampiran.

Tabel 1. Contoh *Dataset*

| Dataset *Kaggle* | | | | | |
|---|---|---|---|---|---|
| *Blight* |  |  |  |  |  |
| *Common Rust* |  |  |  |  |  |
| *Gray Leaft Spot* |  |  |  |  |  |
| *Healthy* |  |  |  |  |  |
| Data Asli | | | | | |
| Daun sehat | |  | |  | |
| Daun terkena penyakit | |  | |  | |

**2. Pembagian Dataset**

Penelitian menggunakan 4188 dataset. Data dibagi menjadi tiga bagian dengan perbandingan 80% data latih , 10% data validasi dan 10% data uji.

Tabel 2. Pembagian *Dataset*

| Dataset | Total | Training | Validation | Testing |
|---|---|---|---|---|
| *Blight* | 1146 | 916 | 115 | 115 |
| *Common_Rust* | 1306 | 1044 | 131 | 131 |
| *Gray_Leaft_Spot* | 574 | 460 | 57 | 57 |
| *Healthy* | 1162 | 930 | 116 | 116 |
| **Total** | **4188** | **3350** | **419** | **419** |

## 3. Analisis dan Pembahasan

Penelitian ini memulai eksperimen dengan mengidentifikasi komponen-komponen awal yang akan digunakan dalam proses pelatihan sebagai eksperimen dengan harapan menghasilkan model dengan akurasi terbaik dan optimal. Komponen-komponen tersebut akan dipilih sesuai dengan jenis arsitektur yang telah dirancang. Hal ini dilakukan sebelum proses pelatihan dan pengembangan model dilakukan. Proses penelitian menggunakan model VGG16, menggunakan beberapa komponen, pada Tabel 3.

Tabel 3. Komponen Pelatihan

| Nama Komponen | Jenis dan Nilai Komponen |
|---|---|
| Arsitektur | VGG16 |
| Ukuran Input | 224 x 224 piksel |
| *Learning Rate* | 0.001 |
| *Loss Function* | *Categorical Crossentropy* |
| *Batch size* | 64 |
| *Optimizer* | *Adam* |

Tabel 3. Merupakan komponen percobaan, dengan arsitektur VGG16, ukuran *input* 224 x 224, *learning rate* 0.001, *loss function* menggunakan *categorical crossenttopy* karena klasifikasi bersifat *multi-class* dengan format label *one-hot encoding*, pelatihan menggunakan *batch size* 64, dan *optimizer* yang digunakan yaitu *Adam*.

Tabel 4. Model VGG16

| Layer (Type) | Output Shape | Param |
|---|---|---|
| input_layer (inputlayer) | (None, 224, 224, 3) | 0 |
| block1_conv1 (conv2d) | (None, 224, 224, 64) | 1,792 |
| block1_conv2 (conv2d) | (None, 224, 224, 64) | 36,928 |
| block1_pool (maxpooling2d) | (None, 112, 112, 64) | 0 |
| block2_conv1 (conv2d) | (None, 112, 112, 128) | 73,856 |
| block2_conv2 (conv2d) | (None, 112, 112, 128) | 147,584 |

| Layer (Type) | Output Shape | Param |
|---|---|---|
| block2_pool (maxpooling2d) | (None, 56, 56, 128) | 0 |
| block3_conv1 (conv2d) | (None, 56, 56, 256) | 295,168 |
| block3_conv2 (conv2d) | (None, 56, 56, 256) | 590,000 |
| block3_conv3 (conv2d) | (None, 56, 56, 256) | 590,000 |
| block3_pool (maxpooling2d) | (None, 28, 28, 256) | 0 |
| block4_conv1 (conv2d) | (None, 28, 28, 512) | 1,180,160 |
| block4_conv2 (conv2d) | (None, 28, 28, 512) | 2,359,808 |
| block4_conv3 (conv2d) | (None, 28, 28, 512) | 2,359,808 |
| block4_pool (maxpooling2d) | (None, 14, 14, 512) | 0 |
| block5_conv1 (conv2d) | (None, 14, 14, 512) | 2,359,808 |
| block5_conv2 (conv2d) | (None, 14, 14, 512) | 2,359,808 |
| block5_conv3 (conv2d) | (None, 14, 14, 512) | 2,359,808 |
| block5_pool (maxpooling2d) | (None, 7, 7, 512) | 0 |

Tabel 4. merupakan arsitektur VGG16, penelitian ini terdiri dari serangkaian layer konvolusi (*Conv2D*) dan *pooling* yang dirancang untuk mengekstraksi fitur dari citra *input* secara bertahap dan mendalam. Model diawali dengan *layer input* berukuran *(224, 224, 3)* yang menunjukkan bahwa citra yang digunakan berwarna (RGB) dengan resolusi 224x224 piksel. Terdapat lima blok utama yang masing-masing terdiri dari beberapa *Conv2D* dan satu *MaxPooling2D*. Blok pertama terdiri dari dua layer konvolusi, yaitu block1_conv1 dan block1_conv2, yang masing-masing menghasilkan 64 fitur map dan mempertahankan ukuran spasial *(224x224)*. Setelah itu, block1_pool digunakan untuk menurunkan resolusi menjadi *(112x112)*. Blok kedua memperdalam ekstraksi fitur menggunakan dua layer konvolusi (block2_conv1 dan block2_conv2) dengan 128 filter, lalu diikuti oleh *pooling* yang mereduksi ukuran menjadi *(56x56)*. Blok ketiga terdiri dari tiga layer konvolusi dengan 256 filter per layer, yang secara bertahap menangkap pola visual yang lebih kompleks, dan ukuran spasial kembali dikurangi menjadi *(28x28)* setelah proses pooling. Blok keempat dan kelima masing-masing terdiri dari tiga layer konvolusi yang menggunakan 512 filter. Ukuran fitur map terus dipertahankan melalui *padding* 'same', sementara kedalaman (jumlah filter) meningkat secara signifikan. Setiap blok ditutup dengan *MaxPooling2D* yang mengurangi ukuran spasial secara bertahap dari *(28x28)* menjadi *(14x14)* pada blok keempat, dan dari *(14x14)* menjadi *(7x7)* pada blok kelima.

Tabel 5. Model *Sequential*

| Layer (Type) | Output Shape | Param |
|---|---|---|
| Vgg16 (Functional) | (None, 7, 7, 512) | 14,714,688 |
| Conv2d (Conv2d) | (None, 7, 7, 32) | 147,488 |
| Max_Pooling2d (Maxpooling2d) | (None, 3, 3, 32) | 0 |
| Dropout (Dropout) | (None, 3, 3, 32) | 0 |
| Flatten (Flatten) | (None, 288) | 0 |
| Dense (Dense) | (None, 4) | 1,156 |

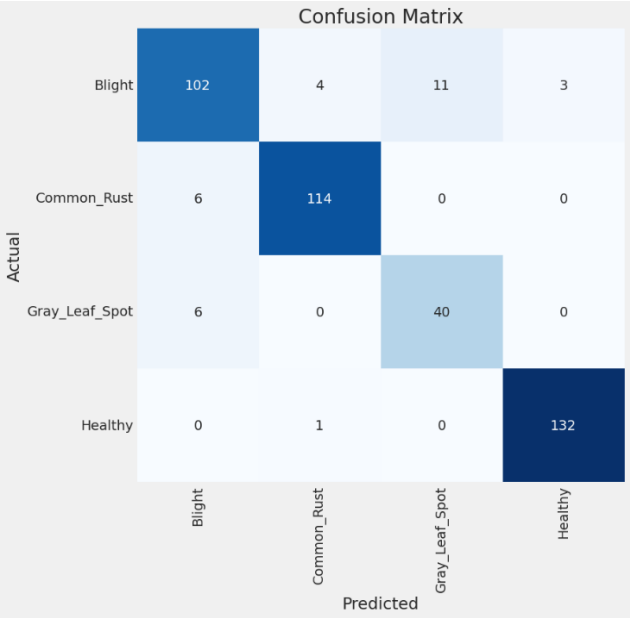
Tabel 5. merupakan Model yang digunakan terdiri dari dua bagian utama, yaitu VGG16 sebagai *feature extractor*, dan beberapa layer tambahan sebagai klasifikator. Arsitektur VGG16 menghasilkan output berupa fitur berukuran (7, 7, 512) dengan total parameter 14.714.688, yang tidak dilatih ulang (*frozen* atau *non-trainable*) untuk efisiensi dan mencegah overfitting. Output dari VGG16 diteruskan ke layer *Conv2D* dengan 32 filter yang bertugas menyesuaikan dan memperkaya fitur, diikuti oleh *MaxPooling2D* untuk mereduksi ukuran menjadi *(3, 3, 32)***.** Selanjutnya digunakan *Dropout* untuk mencegah overfitting dengan menonaktifkan sebagian neuron secara acak selama proses pelatihan.Hasilnya kemudian diratakan dengan *Flatten* menjadi vektor sepanjang 288, yang diteruskan ke layer *Dense* sebanyak 4 unit sebagai output layer. memakai aktivasi *softmax* untuk menghasilkan probabilitas terhadap 4 kelas target, yaitu kelas-kelas penyakit pada daun jagung. Jumlah parameter pada bagian klasifikasi ini relatif kecil dibandingkan VGG16, yaitu hanya sekitar 148.644 parameter**.** Berikut grafik dari *training*, *validation loss* dan *validation accuracy* seperti terlihat pada gambar 1.


Gambar 1. *Training*, validasi loss dan validasi akurasi

Berdasarkan grafik yang ditampilkan, model menunjukkan performa pelatihan yang sangat baik. Grafik sebelah kiri menggambarkan nilai *loss* pada data pelatihan dan validasi. Terlihat bahwa nilai *training loss* (garis merah) dan *validation loss* (garis hijau) turun dengan stabil seiring bertambahnya epoch, menandakan bahwa model belajar dengan stabil dan tidak mengalami *overfitting*. *Validation loss* mencapai titik terendah pada *epoch* ke-10, yang ditandai dengan titik biru sebagai *best epoch* berdasarkan kriteria *loss*. Sementara itu, grafik sebelah kanan menunjukkan akurasi pelatihan dan validasi. *Training accuracy* meningkat tajam di awal dan terus naik mendekati 95%, sedangkan *validation accuracy* juga meningkat dan stabil di atas 90%. Titik akurasi validasi terbaik tercapai pada *epoch* ke-8, yang juga ditandai dengan titik biru. Akurasi validasi cenderung stabil dengan sedikit perubahan, tanpa indikasi overfitting pada model.

**4. Tahap pengujian**

Tahap pengujian model. Hasil *confusin matrix* pada gambar 4.3 berikut.




Gambar 2. *Confusion Matrix*

Berdasarkan *Confusion matrik* dapat diketahui bahwa model telah mengklasifikasikan empat kelas daun jagung yaitu Blight, Common Rust, Gray Leaf Spot, dan Healthy dengan performa yang baik. Untuk kelas Blight, sebanyak 102 gambar berhasil diklasifikasikan dengan benar, sementara 18 gambar diklasifikasikan secara keliru ke kelas lain. Untuk kelas *Common Rust*, model menunjukkan kinerja sangat baik dengan 114 prediksi benar dari total 120 data. Sedangkan pada kelas *Gray Leaf Spot*, terdapat 40 prediksi benar dari 46 data. Terakhir, untuk kelas *Healthy*, model menunjukkan performa tertinggi dengan 132 prediksi benar dari 133 data. Secara keseluruhan, model mampu mengenali citra daun jagung dengan tingkat akurasi tinggi dan hanya terdapat sedikit kesalahan klasifikasi, terutama pada kelas yang memiliki kemiripan visual seperti Blight dan Gray Leaf Spot.

1. *True Positive* (TP) Hasil tes yang benar-benar menunjukkan keberadaan suatu kondisi atau fakta.
2. *False Positive* (FP) Hasil tes yang secara keliru menunjukkan adanya suatu kondisi atau fakta, padahal sebenarnya tidak.
3. *True Negative* (TN) adalah hasil tes yang sebenarnya menunjukkan bahwa suatu kondisi atau fakta tidak ada.
4. *False Negative* (FN) . Hasil tes yang secara keliru menunjukkan bahwa suatu kondisi atau fakta tidak ada, padahal sebenarnya ada

Tabel 6 *Confusion Matrix*

| Nilai Prediksi | Nilai Input | | | | |
|---|---|---|---|---|---|
| | | Blight | Common Rust | Gray Leaf Spot | Healthy | **Total** |
| | Blight | 102 | 4 | 11 | 3 | **120** |
| | Common Rust | 6 | 114 | 0 | 0 | **120** |

| | | | | | | |
|---|---|---|---|---|---|---|
| | Grey Leaf Spot | 6 | 0 | 40 | 0 | **46** |
| | Healthy | 0 | 1 | 0 | 132 | **133** |

Berikut perhitungan manual perkelas dari tabel 6.

1. *Blight*

   TP = 102

   FP = 6 (*Common_Rust*) + 6 (*Gray_Leaf_Spot*) + 0 (*Healthy*) = 12

   FN = 4 + 11 + 3 = 18

   TN = Total sampel - TP - FP - FN = 419 - 102 - 12 - 18 = 287

   $$Precision = \frac{TP}{(TP + FP)} = \frac{102}{(102 + 12)} = 0.895$$

   $$Recall = \frac{TP}{(TP + FP)} = \frac{102}{(102 + 18)} = 0.85$$

   $$F1\ Score = 2\ x \frac{(Precision\ x\ Recall)}{(Precision\ x\ Recall)} = \frac{(0.895\ x\ 0.85)}{(0.895\ x\ 0.85)} = 0{,}872$$

2. *Common_Rust*

   TP = 114

   FP = 4 (*Blight*) + 0 (*Gray_Leaf_Spot*) + 1 (*Healthy*) = 5

   FN = 6 (*Blight*)

   TN = 419 - 114 - 5 - 6 = 294

   $$Precision = \frac{TP}{(TP + FP)} = \frac{114}{(114 + 5)} = 0.958$$

   $$Recall = \frac{TP}{(TP + FP)} = \frac{114}{(114 + 5)} = 0.95$$

   $$F1\ Score = 2\ x \frac{(Precision\ x\ Recall)}{(Precision\ x\ Recall)} = \frac{(0.958\ x\ 0.95)}{(0.958\ x\ 0.95)} = 0.954$$

   F1-Score = 0.954

3. *Gray_Leaf_Spot*

   TP = 40

   FP = 11 (*Blight*)

   FN = 6 (*Gray_Leaf_Spot*)

   TN = 419 - 40 - 11 - 6 = 362

   $$Precision = \frac{TP}{(TP + FP)} = \frac{40}{(40 + 11)} = 0.784$$

   $$Recall = \frac{TP}{(TP + FP)} = \frac{40}{(40 + 6)} = 0.82$$

$$F1\ Score = 2\ x \frac{(Precision\ x\ Recall)}{(Precision\ x\ Recall)} = \frac{(0.784\ x\ 0.869)}{(0.784\ x\ 0.869)} = 0.824$$

4. *Healthy*

TP = 132

FP = 3 (*Blight*)

FN = 1 (*Healthy*)

TN = 419 - 132 - 3 - 1 = 283

$$Precision = \frac{TP}{(TP + FP)} = \frac{132}{(132 + 3)} = 0.978$$

$$Recall = \frac{TP}{(TP + FP)} = \frac{132}{(132 + 1)} = 0.992$$

$$F1\ Score = 2\ x \frac{(Precision\ x\ Recall)}{(Precision\ x\ Recall)} = \frac{(0.978\ x\ 0.992)}{(0.978\ x\ 0.992)} = 0.0.985$$

Hasil penhujian pada Tabel 6. Nilai akurasi mencapai 93%, yang menunjukkan bahwa model memiliki tingkat prediksi yang sangat akurat. Semua metrik lainnya *precision*, *recall*, dan *f1-score* berada pada kisaran 0.80 - 0.90, memperkuat bukti bahwa model bekerja sangat baik untuk semua kelas, tanpa menunjukkan bias terhadap salah satu kelas. Kesalahan klasifikasi yang terjadi sangat minimal dan tergolong wajar. Kesalahan-kesalahan kecil ini kemungkinan besar disebabkan oleh kemiripan visual antar kelas atau kualitas gambar tertentu yang tidak representatif. Secara keseluruhan, hasil pengujian membuktikan bahwa model *deep learning* yang dibangun sangat efektif dan andal dalam mendeteksi penyakit daun jagung. Tingkat akurasi yang tinggi, disertai dengan kestabilan matrik lainnya, menunjukkan bahwa model ini siap digunakan untuk implementasi nyata dalam sistem deteksi penyakit tanaman secara otomatis.

Tabel 7. Tabel Matrix Evaluasi

| Kelas | *Precision* | *Recall* | *F1-Score* | *Report* |
|---|---|---|---|---|
| Blight | 0,89 | 0.85 | 0.87 | 120 |
| Common_Rust | 0.96 | 0.95 | 0.95 | 120 |
| Gray_Leaf_Spot | 0.78 | 0,87 | 0.82 | 46 |
| Healthy | 0,98 | 0.99 | 0,99 | 133 |
| Akurasi | | | 0.93 | 419 |
| Macro Avg | 0.90 | 0.92 | 0.91 | 419 |
| Weighted Avg | 0.93 | 0.93 | 0.93 | 419 |

Berdasarkan hasil evaluasi performa model dari tabel 7, diperoleh nilai akurasi keseluruhan sebesar 93%, yang menunjukkan bahwa model mampu mengklasifikasikan citra daun jagung dengan sangat baik. Selain akurasi, metrik evaluasi lain yang digunakan adalah *precision*, *recall*, dan F1-*score* untuk setiap kelas. Nilai *precision* menggambarkan seberapa tepat prediksi model untuk setiap kelas, sedangkan *recall* menunjukkan seberapa baik model mengenali semua data aktual dari setiap kelas. M*acro average* sebesar 0,90 (*precision*), 0,92 (*recall*), dan 0,91 (*F1-score*), serta *weighted average* sebesar 0,93 untuk ketiganya, menunjukkan bahwa model ini memiliki keseimbangan

performa yang baik di semua kelas, bahkan pada kelas dengan jumlah data yang lebih sedikit. Oleh karena itu, dapat disimpulkan bahwa model klasifikasi yang dikembangkan telah menunjukkan kinerja yang baik, dengan *precision* dan *recall* yang tinggi, sehingga layak digunakan untuk klasifikasi penyakit daun jagung.

## V. Kesimpulan dan Saran

### 1. Kesimpulan

Berdasarkan evaluasi model, dapat diketahui bahwa model telah mengklasifikasikan empat kelas daun jagung yaitu *Blight*, *Common Rust*, *Gray Leaf Spot*, dan *Healthy* dengan performa yang baik. Untuk kelas *Blight*, sebanyak 102 gambar berhasil diklasifikasikan dengan benar, sementara 18 gambar diklasifikasikan secara keliru ke kelas lain. *Precision* kelas ini adalah sekitar 89,5%, dan *recall*-nya sekitar 85%, menghasilkan *F1-score* sekitar 87,2%. Untuk kelas *Common Rust*, model menunjukkan kinerja sangat baik dengan 114 prediksi benar dari total 120 data, memberikan *precision* sekitar 95,8% dan *recall* 95%. Sedangkan pada kelas *Gray Leaf Spot*, terdapat 40 prediksi benar dari 46 data, dengan *precision* sekitar 78,4% dan *recall* 86,9%. Terakhir, untuk kelas *Healthy*, model menunjukkan performa tertinggi dengan 132 prediksi benar dari 133 data, menghasilkan *precision* sekitar 97,7% dan *recall* 99,2%. Tingkat akurasi dari algoritma *Convolutional Neural Network* dengan arsitektur *VGG16* dalam mengklasifikasi hama pada daun tanaman jagung menghasilkan akurasi keseluruhan yang tinggi yaitu 93%.

### 2. Saran

Beberapa rekomendasi untuk penelitian lebih lanjut dalam konteks yang sama berdasarkan temuan yang dievaluasi dalam studi ini, khususnya yang berikut ini:

1. Menggunakan dataset yang baik tidak buram dan perhatikan jumlah dataset setiap kelasnya jangan ada ketimpangan data sehingga proses pelatihan lebih maksimal dengan hasil yang bisa lebih baik.
2. Selain menggunakan VGG16, peneliti selanjutnya dapat mencoba arsitektur CNN lain yang lebih modern seperti *ResNet*, *EfficientNet*, atau *MobileNet* untuk melihat perbandingan performa dan efisiensi komputasi.

## Referensi


[1] D. Iswantoro And D. Handayani Un, "Klasifikasi Penyakit Tanaman Jagung Menggunakan Metode Convolutional Neural Network (Cnn)," *J. Ilm. Univ. Batanghari Jambi*, Vol. 22, No. 2, P. 900, 2022, Doi: 10.33087/Jiubj.V22i2.2065.

[2] M. I. Fathur Rozi, N. O. Adiwijaya, And D. I. Swasono, "Identifikasi Kinerja Arsitektur Transfer Learning Vgg16, Resnet-50, Dan Inception-V3 Dalam Pengklasifikasian Citra Penyakit Daun Tomat," *J. Ris. Rekayasa Elektro*, Vol. 5, No. 2, P. 145, 2023, Doi: 10.30595/Jrre.V5i2.18050.

[3] B. W. Kurniadi, H. Prasetyo, G. L. Ahmad, B. Aditya Wibisono, And D. Sandya Prasvita, "Analisis Perbandingan Algoritma Svm Dan Cnn Untuk Klasifikasi Buah," *Semin. Nas. Mhs. Ilmu Komput. Dan Apl. Jakarta-Indonesia*, No. September, Pp. 1–11, 2021.

[4] A. Purnamawati, W. Nugroho, D. Putri, And W. F. Hidayat, "Deteksi Penyakit Daun Pada Tanaman Padi Menggunakan Algoritma Decision Tree, Random Forest, Naïve Bayes, Svmdan Knn," *Infotekjar J. Nas. Inform. Dan Teknol. Jar.*, Vol. 5, No. 1, Pp. 212–215, 2020, [Online]. Available: Https://Doi.Org/10.30743/Infotekjar.V5i1.2934.

[5] U. Khultsum And A. Subekti, "Penerapan Algoritma Random Forest Dengan Kombinasi Ekstraksi Fitur Untuk Klasifikasi Penyakit Daun Tomat," *J. Media Inform. Budidarma*, Vol. 5, No. 1, P. 186, 2021, Doi: 10.30865/Mib.V5i1.2624.

[6] F. Syahputra, "Analisis Arsitektur Deep Learning Mobilenet Dalam Mengklasifikasi Hama Daun Jambu Madu," 2023, [Online]. Available: Https://Repositori.Uma.Ac.Id/Handle/123456789/22686.

[7] J. Nurhakiki *Et Al.*, "Studi Kepustakaan: Pengenalan 4 Algoritma Pada Pembelajaran Deep Learning Beserta Implikasinya," *J. Pendidik. Berkarakter*, No. 1, Pp. 270–281, 2024, [Online]. Available: Https://Doi.Org/10.51903/Pendekar.V2i1.598.

[8] S. N. Nugraha, R. Pebrianto, And E. Fitri, "Penerapan Deep Learning Pada Klasifikasi Tanaman Paprika Berdasarkan Citra Daun Menggunakan Metode Cnn," *Inf. Syst. Educ. Prof. J. Inf. Syst.*, Vol. 8, No. 2, P. 15, 2023, Doi: 10.51211/Isbi.V8i2.2671.

[9] M. R. Alamadani And D. Indriyana, "Klasifikasi Kesehatan Tanaman Padi Menggunakan Algoritma Convolutional Neural Network," Vol. 8, No. 5, Pp. 10177–10182, 2024.

[10] U. Shodiq *Et Al.*, "Klasifikasi Jalan Rusak Menggunakan Transfer Learning Arsitektur Vgg16," *Joisie (Journal Inf. Syst. Informatics Eng.*, Vol. 8, No. 1, 2024.