BAB II

KAJIAN PUSTAKA

A. Kajian Teoritis

1. Penyakit Tanaman Kedelai

Tanaman kedelai (*Glycine max* (L.) *Merr*) adalah tanaman legum yang termasuk dalam keluarga *Fabaceae*. Tanaman kedelai merupakan tanaman pertanian yang cukup banyak dikonsumsi sebagai bahan pangan yang dapat memenuhi gizi masyarakat. Selain itu, kedelai memiliki kandungan protein yang tinggi, sebagai sumber lemak, mineral, dan vitamin (Suryandari, 2023:2-4). Biji merupakan salah satu hasil dari produksi kedelai yang kemudian dapat dikonsumsi secara langsung dan dapat dimanfaatkan sebagai bahan baku industri seperti tahu, tempe, taoco, dan masih banyak lagi (Suartha & Wedastra, 2023).

Menurut Soesanto (2022:5) selama pertumbuhannya, tanaman kedelai biasa diganggu oleh organisme pengganggu, khususnya penyakit yang dapat menyebabkan kematian pada tanaman, baik pada stadium vegetatif maupun reproduktif. Tidak sedikit masyarakat atau petani yang gagal panen karena serangan penyakit pada daun yang mengakibatkan terganggunya proses fotosintesis. Berbagai penyakit yang disebabkan oleh jamur, bakteri, virus, dan faktor lingkungan dapat merusak daun tanaman kedelai. Ada beberapa jenis penyakit yang tampak pada daun kedelai yakni

sindrom kematian mendadak, hawar bakteri, target spot, hama *diabrotica sp.*, hama ulat daun, virus *mossaic* kuning, dan masih banyak lagi.

2. Pengolahan Citra Digital

Menurut Kirana (2021:2) citra adalah representasi dua dimensi dari benda tiga dimensi melalui integritas titik, garis, bentuk, dan warna. Dalam teknologi digital, citra adalah gabungan piksel yang terdiri dari warna untuk meniru objek serta menyimpan fitur tertentu. Dalam konteks komputer, citra biasanya merujuk pada gambar digital yang dapat dilihat di layar, dicetak, atau disimpan sebagai file. Citra digital terdiri dari kumpulan piksel yang masing-masing mewakili nilai warna tertentu.

Pengolahan citra digital atau *digital image processing* merupakan proses rekayasa dan analisis gambar menggunakan teknologi komputer yang melibatkan berbagai teknik dan algoritma untuk mengolah data guna mencapai tujuan tertentu. Pengolahan citra digital adalah pengolahan dan analisis gambar yang menggunakan persepsi visual (Aditya et al., 2020). Pengolahan citra merupakan jenis pengolahan yang *input*-nya berupa gambar maupun video dengan *output* pengolahannya dapat berupa gambar atau sejumlah parameter atau karakteristik yang terkait dengan gambar (Andrian et al., 2022).

3. *Pre-Processing*

Sebelum dilakukannya identifikasi, citra terlebih dahulu dilakukan *pre*-processing. Dataset penyakit pada daun tanaman kedelai yang telah dikumpulkan memiliki ukuran yang berbeda. Maka, *pre*-

processing dilakukan untuk mengubah ukuran citra asli sehingga setiap citra memiliki ukuran yang sama.

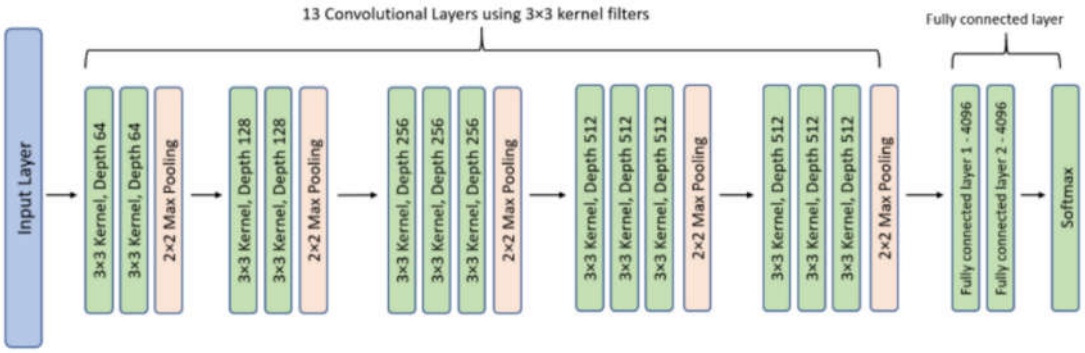
Pre-processing menurut Prasetyo et al. (2023) merupakan langkah awal dalam mempersiapkan data mentah agar model pembelajaran mesin dapat menggunakannya dengan baik. Langkah-langkah ini berfokus pada pembersihan dan transformasi data agar lebih mudah diolah oleh model. Dalam penelitian ini, *pre-processing* yang dilakukan pada citra diantaranya:

a. *Resizing, resizing* pada citra dilakukan dengan mengubah ukuran menjadi 224x224 *pixels*.
b. Normalisasi, normalisasi pada citra dilakukan ke dalam rentang 0 hingga 1 dengan membagi nilai *pixels* dengan 225.
c. Augmentasi data, augmentasi dilakukan dengan menambah variasi pada data pelatihan dengan membuat modifikasi rotasi dan pemotongan.

4. *Feature Extraction*

Feature extraction menurut merupakan proses pengurangan di mana sekumpulan data mentah diubah menjadi sekumpulan fitur yang telah direduksi dengan tujuan untuk mengurangi jumlah sumber daya yang dibutuhkan untuk menggambarkan sekumpulan data yang besar secara akurat dengan tetap mempertahankan informasi penting di dalamnya. Pada penelitian ini, ekstraksi fitur menggunakan arsitektur VGG16.

VGG16 menurut Yang et al. (2021) merupakan arsitektur CNN yang dikembangkan oleh Visual Geometry Group (VGG) dari University of Oxford, digunakan untuk menunjukkan bahwa memperluas kedalaman jaringan dapat meningkatkan kinerja jaringan dalam situasi tertentu. Arsitektur VGG16 memiliki jumlah lapisan layer sebanyak 16 yang terdiri dari 13 lapisan *convolution*, 2 lapisan *fully connected*, dan 1 lapisan *classifier*.



Gambar 2. 1 Arsitektur VGG16

Sumber: Rismiyati & Luthfiarta (2021)

Pada gambar 2.1, lapisan *convolution* mempunyai ukuran 3x3 dengan perbedaan pada jumlah filter di setiap lapisan. Dua lapisan *convolution* pertama memiliki filter sebanyak 64, pada lapisan 3 dan 4 memiliki jumlah karnel sebanyak 128, begitu juga dengan lapisan 5, 6, 7 dengan jumlah filter sebanyak 256 dan lapisan 8 sampai 13 memiliki jumlah filter sebanyak 512. *Max pooling* 2x2 dilakukan setelah lapisan *convolution* ke 2, 4, 7, 10, dan 13. Keluaran dari *max pooling* terakhir akan dikirim ke *fully connected* layer dan akan terhubung ke *classifier* untuk menentukan kelas dari citra.

5. Metode Citra Digital

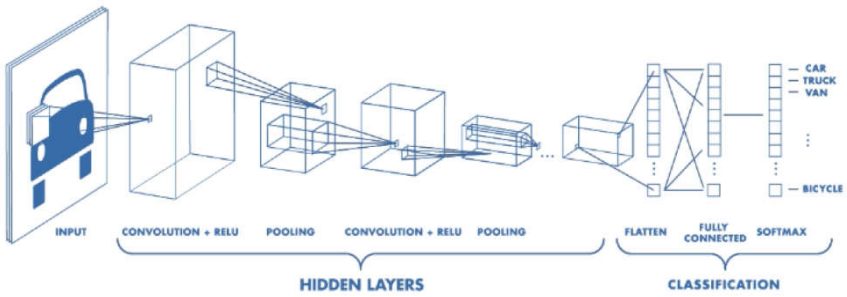
a. *Convolutional Neural Network*

1) Pengertian *Convolutional Neural Network*

Menurut Putro et al. (2020:84) *Convolutional Neural Network* (CNN) adalah salah satu algoritma *deep learning* yang dikerjakan melalui *input image*, ini memungkinkan CNN untuk membedakan satu gambar dengan yang lainnya. CNN menggunakan *feed forward neural network* yang penerapannya menggunakan cara kerja *visual cortex* manusia dengan menggunakan beberapa komponen yang bekerja sama untuk mengolah data berpola *grid* (Sa'idah et al., 2022).

Convolutional Neural Network (CNN) adalah modifikasi dari *Multi Layer Perceptron* (MLP) yang mempunyai sedikit parameter bebas (Irfansyah et al., 2021). CNN memiliki arsitektur *network* yang terdiri dari puluhan hingga ratusan layer untuk memproses citra dan menghasilkan *output* pada kelas tertentu. *Output* dari setiap layer, digunakan sebagai dasar *input* untuk layer selanjutnya. Pada awal *network*, layer akan menghasilkan fitur seperti warna, kecerahan, dan tepi. Kemudian *network* akan menghasilkan fitur yang lebih kompleks (Setiawan, 2023:8-9).

Menurut Peryanto et al. (2019) *convolutional neural network* (CNN) terdiri dari 4 jenis komponen utama diantaranya *convolutional layer, pooling layer, fully connected layer* dan *dropout*. Lapisan-lapisan tersebutlah yang membangun metode CNN, sehingga dapat digunakan untuk pemrosesan gambar karena setiap *neuron*-nya dipresentasikan dalam bentuk 3D. Arsitektur *convolutional neural network* dapat dilihat pada gambar 2.2 berikut:



Gambar 2. 2 Arsitektur *Convolutional Neural Network*

Sumber: Peryanto et al. (2019)

Langkah-langkah yang digunakan dalam metode CNN untuk mengidentifikasi citra dapat dijelaskan sebagai berikut:

a) Pengumpulan dan pra-pemrosesan data

(1) Mengumpulkan dataset dalam bentuk gambar yang relevan dengan tujuan identifikasi.

(2) Melakukan pra-pemrosesan pada dataset seperti normalisasi, *resizing, cropping,* augmentasi data, serta pembagian dataset menjadi data latih dan data uji.

b) Membuat arsitektur CNN

(1) Menentukan arsitektur CNN yang akan digunakan untuk identifikasi gambar, termasuk jumlah lapisan, ukuran karnel, *pooling, dropout,* dan fungsi aktivasi.

(2) Menggunakan arsitektur standar seperti VGG-*Net*, *ResNet,* atau *AlexNet*, atau membuat model dari awal.

c) Pelatihan model

(1) Melatih model CNN menggunakan data latih yang telah diproses dengan menggunakan teknik *backpropagation* dan optimasi *gradien* untuk meng-*update* bobot model.

(2) Menentukan parameter pelatihan seperti jumlah *epoch, batch size, learning rate,* dan *optimizer.*

d) Evaluasi model

(1) Evaluasi performa model dengan menggunakan data uji yang belum pernah dibuat sebelumnya, dan hitung *accuracy, precision, recall,* dan *f1-score* untuk mengevaluasi performa model.

(2) Jika model yang dilatih tidak memuaskan, coba modifikasi parameter pelatihan atau ubah arsitektur model dan lakukan pelatihan ulang.

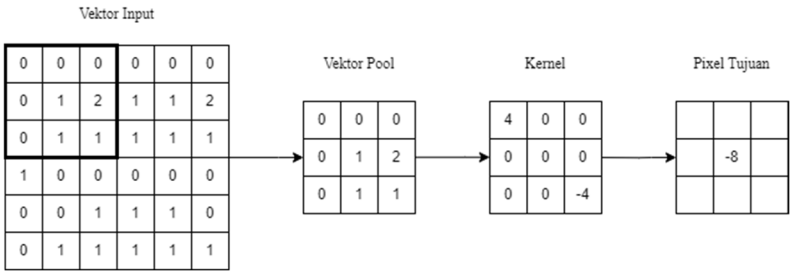
e) Penggunaan model

Setelah model dipelajari dan dievaluasi performanya sesuai, gunakan model tersebut untuk melakukan prediksi pada gambar baru dan dapatkan hasil identifikasi yang diinginkan.

Langkah-langkah di atas dapat diulang dan ditingkatkan dengan mengubah parameter model, augmentasi data, atau menggunakan teknik *transfer learning* untuk meningkatkan performa model.

2) *Convolution Layer*

Convolution layer adalah lapisan utama metode *convolutional neural network* (CNN) karena proses *convolution* yang memungkinkan proses untuk mengidentifikasi pola seperti tepi, bentuk, warna, dan tekstur pada gambar. Hasil filter yang menghasilkan pola bergantung pada matriks filter, yang memiliki nilai awal acak. Kemudian, ukuran filter ditentukan dalam nilai baris dan kolomnya (Ersyad et al., 2020). Tujuannya adalah untuk mempelajari representasi fitur *input* melalui mengekstraksi fitur dari gambar, yang *output*-nya berupa transformasi *linear* dari data *input* (Alwanda et al., 2020). Ilustrasi *convolutional layer* dapat dilihat pada gambar 2.3 berikut:



Gambar 2. 3 Representasi Visual *Convolutional Layer*

Sumber: Magdalena et al. (2021)

Convolution layer memiliki beberapa kumpulan filter yang dapat menggabungkan semua *input* gambar dan menghasilkan berbagai jenis *feature map*. Proses konvolusi mengubah semua data ke seluruh bagian proses filter dengan hasil setiap *binary* (Sukma & Mukhaiyar, 2022). *Convolution layer* melakukan operasi konvolusi pada output lapisan sebelumnya. Tujuan dari melakukan konvolusi pada data gambar adalah untuk mengekstrak fitur dari gambar *input*. Operasi konvolusi dapat ditulis seperti yang ditunjukkan dalam persamaan 1:

$$FM[i]_{j,k} = \left(\sum_m \sum_n N_{[j-m,k-n]} F_{[m,n]}\right) + bF \qquad (1)$$

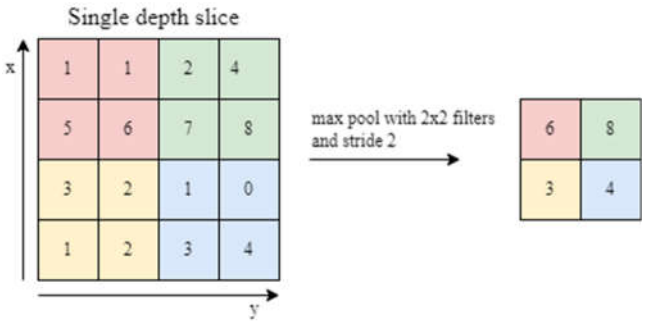
Keterangan

FM[i] : Matriks *feature map* ke-i
j, k : Posisi *pixel* pada matriks citra *input*
m, n : Posisi *pixel* pada matriks *filter* konvolusi
N : Matriks citra masukan
F : Matriks *filter* masukan
bF : Nilai bias pada *filter*

3) *Pooling Layer*

Pooling layer adalah lapisan ekstraksi setelah lapisan *convolutional*. Prosesnya didasarkan pada pengambilan sampel

bawah, yang digunakan untuk mengurangi kompleksitas pada lapisan berikutnya. Proses ini menggunakan prinsip filter, yang memiliki ukuran dan langkah tertentu, dan bergerak di seluruh arep *feature map* (Magdalena et al., 2021). *Pooling layer* digunakan untuk mengekstrak fitur representatif dari tensor *input* dan mengurangi *overfitting* pada citra, dan mengurangi jumlah parameter dari *tensor input* dan mengurangi perhitungan yang membantu efisiensi (Kholik, 2021). *Max pooling* dan *average pooling* biasanya digunakan untuk perhitungan untuk lapisan *pooling*. Layer pooling menghasilkan nilai spasial yang dikurangi berdasarkan ukuran baris dan kolomnya. Proses pada lapisan *pooling* hampir sama dengan lapisan *convolution*, tetapi nilai yang dihasilkan adalah nilai terbesar atau rata-rata untuk setiap *neuron*. Ilustrasi *pooling layer* dapat dilihat pada gambar 2.4 sebagai berikut:



Gambar 2. 4 Ilustrasi *Pooling Layer*

Sumber: Peryanto et al. (2019)

Pada gambar 2.4 menunjukkan bahwa metode CNN dilakukan dengan operasi *max pooling* pada citra dengan ukuran

4x4 menggunakan *pooling mask* 2x2. Hasil dari matriks asal pada proses di atas adalah matriks dengan ukuran yang lebih kecil. Proses konvolusi dan *pooling* ini dilakukan berulang kali hingga menghasilkan fitur yang diinginkan. Selanjutnya, fitur tersebut akan dijadikan *input* pada *fully connected layer*.

4) *Fully Connected Layer*

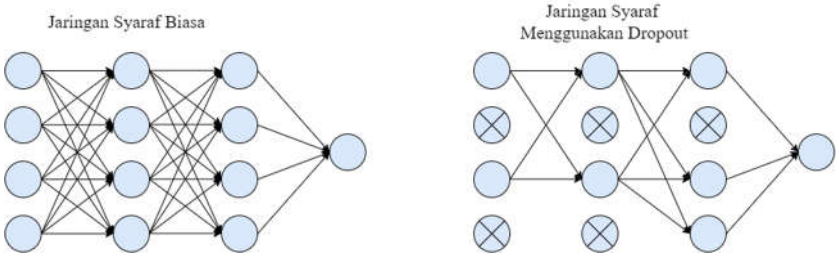
Feature map yang dihasilkan pada tahap sebelumnya masih berbentuk *multidimensional array*. Sehingga, *feature map* akan melalui proses *flatten* sebelum masuk pada tahap *fully connected layer*. Proses *flatten* menghasilkan sebuah vektor untuk digunakan sebagai *input* pada *fully connected layer*. *Fully connected layer* merupakan lapisan yang digunakan untuk transformasi pada dimensi agar data dapat diklasifikasikan secara *linear*. Untuk mendapatkan hasil *output* tidak dibutuhkan operasi konvolusi, melainkan menggunakan perkalian matriks yang diikuti dengan *bias offset*. Maka pada lapisan sebelumnya, setiap *neuron* memiliki koneksi penuh ke semua aktivasi (Alwanda et al., 2020).

5) *Dropout*

Dropout menurut Peryanto et al. (2019) merupakan metode untuk regularisasi jaringan *neuron* dengan memilih beberapa *neuron* secara acak dan melepaskan atau tidak digunakan selama pelatihan. Ini berarti bahwa kontribusi *neuron*

yang dibuang akan dihentikan sementara jaringan dan bobot baru tidak diterapkan pada *neuron* saat *backpropagation. Dropout* adalah metode untuk meminimalisir *overfitting neural network* dengan solusi melakukan penyesuaian pada proses pelatihan data. *Dropout* juga dapat mempercepat proses pelatihan selama proses pelatihan (Magdalena et al., 2021).

Overfitting adalah kondisi di mana model sesuai dengan data pelatihan namun mengalami kegagalan pada saat generalisasi ke kumpulan data lainnya. Dalam hal ini, fungsi kerugian adalah pedang bermata dua (Dawis et al., 2022:142). Suatu *neuron* dalam jaringan baik lapisan tersembunyi maupun lapisan terlihat akan dihilangkan sementara oleh *dropout* dalam sistem kerjanya. Penggunaan *dropout* dapat dilihat pada gambar 2.5 sebagai berikut:



Gambar 2. 5 Perbedaan Ketika Menggunakan *Dropout*

Sumber: Kholik (2021)

6) *Activation Function*

Fungsi aktivasi dihitung setelah operasi konvolusi selesai. Pada *convolutional neural network* (CNN), fungsi aktivasi yang sering digunakan adalah *ReLu* dan *Softmax.*

a. *Rectified Linear Units* (*ReLu*)

Rectified linear units (*ReLu*) menurut Budi et al. (2021) merupakan fungsi aktivasi dengan kelebihan dalam fondasi matematis karena dapat memproses data ukuran besar dengan cepat di antara lapisan *convolutional* dan *pooling*. *ReLu* menjaga hasil citra konvolusi pada domain yang didefinisikan positif, sehingga nilai negatif dari proses konvolusi akan melalui proses *ReLu* dan menjadi sama dengan 0. Persamaan *ReLu* dapat ditulis seperti pada persamaan 2 berikut:

$$f = \max(0, x) \qquad (2)$$

b. *Softmax*

Softmax adalah fungsi aktivasi yang umum digunakan pada *neural network* dengan banyak kategori *output* (*multi-class*). Fungsinya adalah mengubah nilai perhitungan menjadi nilai probabilitas, sehingga nilai perhitungan dapat dibandingkan satu sama lain. Melalui *softmax*, diperoleh kelas yang menghasilkan nilai kemungkinan terbesar, yang akan menjadi kelas terpilih, dan seluruh masukan berikutnya akan digunakan dalam kelas tersebut (Budi et al., 2021). Persamaan *softmax* dapat ditulis seperti pada persamaan 3 berikut:

$$f_i(x) = \frac{e^{xi}}{\sum_{j=1}^{k} e^{xj}} \qquad (3)$$

$$w \ \ ere\ i = 1,2,3, \dots, k$$

$$0 \leq f_i(x) \leq 1$$

Dimana k adalah jumlah *input* dan x adalah vektor *input.*

7) *Epoch*

Menurut Kurnia & Wibowo (2021) *epoch* merupakan salah satu *hyperparameter* yang dihasilkan dari satu putaran dataset yang direpresentasikan dalam bentuk angka. Dalam proses identifikasi gambar, *epoch* biasanya dikombinasikan dengan *batch size* untuk meningkatkan akurasi. *Epoch* digunakan untuk menentukan berapa kali algoritma dilatih menggunakan dataset yang diberikan. Semakin tinggi nilai *epoch*, maka semakin tinggi pula nilai akurasi dan semakin rendah nilai *loss*.

Satu siklus algoritma belajar dari dataset pelatihan yang ingin diproses ditunjukkan oleh *epoch*. Jika menggunakan *min batch*, mesin tidak belajar dari keseluruhan data, namun hanya dari *batch* yang telah ditentukan. Dalam satu *epoch*, sebuah algoritma telah belajar data pelatihan secara keseluruhan. Dalam *artificial neural network*, pembelajaran yang berulang bertujuan untuk mencapai konvergensi nilai bobot (Wasil et al., 2022).

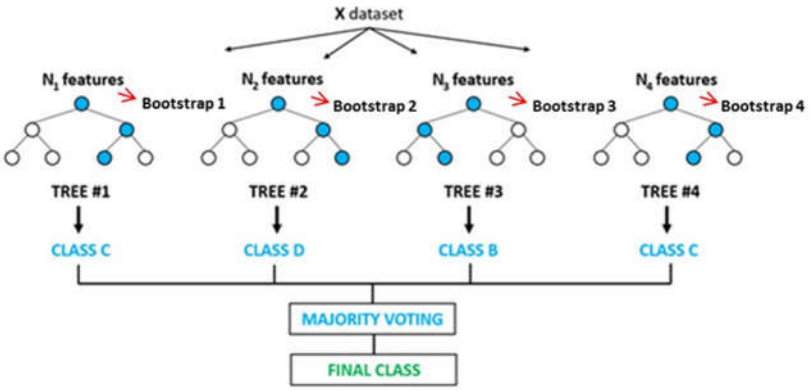
b. *Random Forest*

1) Pengertian *Random Forest*

Menurut Rahman et al. (2023) *Random Forest* (RF) merupakan teknik *bagging* yang menghasilkan banyak pohon dari

data sampel. Dalam pelatihan data, pembentukan satu pohon tidak bergantung pada pohon sebelumnya. Dalam metode *ensemble learning*, *random forest* banyak menerapkan pohon keputusan dan luaran berupa kelas modus pada pohon individu (Akbar, 2020). *Random forest* merupakan metode klasifikasi *supervised learning* yang memiliki pengelompokan pada variabel dependennya. *Random forest* terdiri dari beberapa model klasifikasi pohon dengan mengambil sampel ulang data. Setiap pohon menampilkan hasil klasifikasi yang kemudian dipilih oleh *random forest* dengan sebagian besar muncul berdasarkan agregasi dari hasil pohon yang terbentuk (Khairani et al., 2022).

Keputusan akhir dari klasifikasi ini didasarkan pada jumlah voting dari pohon dan ditentukan suara yang paling banyak (modus) (Adi & Wintarti, 2022). Metode ini memiliki keunggulan diantaranya, hasil identifikasi yang baik, hasil error yang lebih rendah, dan secara efisien dapat mengatasi data training dalam jumlah besar. Ilustrasi proses pada metode *random forest* dapat dilihat pada gambar 2.6 berikut:



Gambar 2. 6 Ilustrasi *Random Forest*

Sumber: Khairani et al. (2022)

Berdasarkan pada gambar di atas, konsep tahapan persiapan dan estimasi menggunakan metode *random forest* adalah sebagai berikut:

a) Langkah awal melakukan *bootstrap* hingga mencapai jumlah 1, yaitu dengan mengambil sampel secara acak dari data latih variabel independen yang dimiliki dengan ukuran kembali n.

b) Menggunakan contoh pada setiap *bootstrap* untuk membangun pohon hingga mencapai ukuran maksimum. Susun pohon berdasarkan *bootstrap*. Setiap proses pemisahan memilih m < p sebagai variabel independen dan pemisahan terbaik dilakukan pada tahap *sub-setting* secara acak.

c) Ulangi langkah 1-2 sebanyak 1 kali untuk membentuk sebuah hutan yang terdiri dari pohon-pohon.

d) Melakukan prediksi berdasarkan hasil prediksi pada setiap dataset hasil *bootstrap* dengan menggunakan suara untuk kasus klasifikasi.

2) *Bagging* (*Bootstrap Aggregating*)

Dalam upaya meningkatkan stabilitas dan akurasi pada *random forest*, *bagging* atau *bootstrap aggregating* adalah teknik *ensemble learning* dimana sejumlah besar pohon keputusan dibuat pada subset data latih yang acak dengan menggunakan subset yang berbeda dari atribut. Subset fitur dipilih secara acak dalam setiap pohon keputusan untuk membuat pemilihan fitur lebih acak dan mengurangi ketergantungan pada fitur tertentu. *Random forest* dapat membuat prediksi yang lebih akurat dan konsisten dengan menggunakan teknik *bagging* daripada hanya menggunakan satu pohon keputusan. Hal ini disebabkan dengan membangun model dengan banyak pohon keputusan yang berbeda, *bagging* dapat mengurangi kemungkinan *overfitting* dan meningkatkan generalisasi.

3) Regresi *Random Forest*

Regresi *random forest* menurut Fachid & Triayudi (2022) merupakan salah satu algoritma *machine learning* yang melibatkan metode *ensemble learning* untuk proses regresi. Metode ini menggunakan metode *random forest* dengan menggabungkan prediksi dari prosedur pemecahan pada *machine*

learning dengan algoritma regresi untuk mendapatkan hasil prediksi yang stabil dan lebih akurat berdasarkan variabel predikator.

Dalam regresi *random forest,* beberapa *decision tree* dibangun menggunakan sampel data yang diambil secara acak dari dataset *training*. Setiap *decision tree* memprediksi nilai target berdasarkan variabel predikator dan hasilnya kemudian digabungkan untuk membuat prediksi akhir. Regresi *random forest* sangat efektif dalam menangani data yang memiliki banyak fitur dan kompleksitas tinggi, dan dapat mengurangi terjadinya *overfitting*. Oleh karena itu, metode ini banyak digunakan dalam berbagai aplikasi seperti pengolahan citra, pemodelan ekonomi, dan ilmu lingkungan.

4) *Ensemble Learning*

Ensemble learning menurut Sepbriant & Utomo (2024) merupakan teknik *machine learning* dengan penggabungan *learner* atau *classifier* yang tidak begitu kuat dengan sebuah *learner* yang mempunyai hasil prediksi yang baik untuk mendapatkan performasi yang lebih baik dan meningkatkan akurasi, kesalahan, atau bias model. Ketika model individu memiliki kekuatan dan kelemahan yang berbeda, metode ini sangat bermanfaat karena dapat membantu mengimbangi kelemahan dan menghasilkan sistem prediksi yang lebih akurat.

Teknik *ensemble learning* dapat diterapkan pada berbagai jenis metode seperti *decision tree*, *neural network,* dan *regression* model. Beberapa teknik *ensemble learning* yang populer diantaranya adalah *bagging, boosting,* dan *stacking.*

5) *Variable Importance*

Dalam *machine learning*, istilah *variable importance* mengacu pada ukuran kontribusi relatif dari setiap variabel atau fitur dalam dataset terhadap kinerja model secara keseluruhan. Ini merupakan komponen penting dari evaluasi dan pemilihan model karena membantu dalam menentukan karakteristik yang paling relevan yang mendorong prediksi. Metode *ensemble learning* seperti *random forest* biasanya menggunakan *variable importance* untuk menemukan fitur terpenting yang berkontribusi pada akurasi model. Dengan begitu, *variable importance* adalah alat yang ampuh pada metode *random forest* yang membantu mengidentifikasi fitur yang paling relevan dalam kumpulan data.

6. *Confusion Matrix*

Pada penelitian ini, pengukuran performa identifikasi yang dilakukan menggunakan bantuan *Confusion Matrix*. Menurut Delfariyadi et al. (2022) *confusion matrix* merupakan *matrix* yang menunjukkan hasil evaluasi klasifikasi dan menghitung jumlah prediksi yang benar yang dibuat oleh mesin. *Confusion matrix* merupakan tabel yang sering

digunakan untuk menunjukkan performa model pada dataset uji yang nilainya sudah diketahui (Prasojo & Haryatmi, 2021).

Dalam *confusion matrix*, terdapat empat istilah untuk mempresentasikan hasil dari identifikasi yaitu *True Positive* (TP), *True Negative* (TF), *False Positive* (FP), dan *False Negative* (FN). Berikut adalah penjelasan dari keempat istilah tersebut:

True Positive (TP) : Kasus dimana model memprediksi hasil positif dengan tepat (hasil sebenarnya positif).

True Negative (TN) : Kasus dimana model memprediksi hasil negatif dengan tepat (hasil sebenarnya negatif).

False Positive (FP) : Kasus dimana model salah memprediksi hasil positif (hasil sebenarnya negatif).

False Negative (FN) : Kasus dimana model salah memprediksi hasil negatif (hasil sebenarnya positif).

Untuk dapat memudahkan pemahaman mengenai *confusion matrix*, dapat dilihat pada gambar 2.7 seperti berikut:

Gambar 2. 7 *Confusion Matrix*

Sumber: Afifah (2024)

Salah satu fungsi utamanya adalah untuk menentukan *accuracy, precision, recall*, dan *f1-score* untuk merepresentasikan prediksi dan keadaan aktual dari data yang sudah diproses oleh algoritma (Mardiyyah et al., 2024). Nilai *accuracy* adalah ukuran yang menunjukkan seberapa akurat sistem dalam mengidentifikasi data dengan benar. Persamaan *accuracy* dapat dilihat pada persamaan 4 seperti berikut:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} x100\% \quad (4)$$

Perbandingan jumlah data positif yang diidentifikasi secara akurat dengan jumlah total data positif yang diidentifikasi dikenal sebagai *precision*. Persamaan *precision* dapat dilihat pada persamaan 5 seperti berikut:

$$Precision = \frac{TP}{TP+FP} x100\% \quad (5)$$

Recall penting untuk menentukan jumlah data kategori positif yang diidentifikasi dengan benar oleh sistem. Persamaan *recall* dapat dilihat pada persamaan 6 seperti berikut:

$$Recall = \frac{TP}{TP+FN} x100\% \quad (6)$$

Nilai *f1-score* adalah nilai *harmonic mean* untuk *precision* dan *recall*, dengan nilai tertinggi 1 dan nilai terendah 0. Persamaan *f1-score* dapat dilihat pada persamaan 7 seperti berikut:

$$F1 \quad Score = 2x \frac{precision\ x\ recall}{precision+recall} x100\% \quad (7)$$

7. *Python*

Menurut Suharto (2023:1) *Python* adalah bahasa pemrograman dinamis tingkat tinggi yang berfungsi sebagai interpreter, yang berarti

bahwa itu dapat mengubah *source code* menjadi *machine code* secara langsung saat program dijalankan. Bahasa ini juga dianggap sebagai bahasa gabungan yang banyak melibatkan kapabilitas dengan sintaks kode yang sangat jelas, serta memiliki banyak fitur pustaka standar luas dan komprehensif (Akbar et al., 2023).

Python termasuk bahasa *general purpose* atau bahasa yang dapat dimanfaatkan untuk berbagai bidang, misalnya digunakan untuk *artificial intelligence, data analysis, machine learning, data science*, dan membangun *website* (Surya & Efitra, 2023:1). Keunggulan *Python* memiliki banyak modul pustaka yang dapat digunakan untuk meringankan proses pengembangan perangkat lunak dan pengolahan citra, serta memiliki *source code* yang mudah dipelajari serta sederhana (Mas'ud et al., 2023).

8. *Keras*

Keras adalah platform yang menyederhanakan kompleksitas yang terkait dengan *deep neural network* yang berdasarkan prinsip-prinsip keramahan pengguna, kompatibilitas dengan *python*, dan kemampuan untuk digunakan di berbagai perangkat dan platform serta unggul dalam pembuatan model yang lebih cepat dan dukungan yang kuat untuk penyebaran dan adopsi (Coursera, 2024).

Menurut Kapoor et al. (2022:3) *Keras* adalah API yang indah untuk menyusun blok bangunan untuk membuat dan melatih model *deep learning* yang dapat diintegrasikan dengan berbagai jenis mesin *deep*

learning, termasuk *Google Tensorflow, Microsoft CNTK, Amazon MXNet*, dan *Theano*. Salah satu manfaat besar menggunakan *keras* sebagai pengantar *deep learning* adalah bahwa ia sangat ramah pengguna, fungsi-fungsi canggih seperti pengoptimal dan lapisan sudah dibangun ke dalam pustaka dan tidak harus ditulis dari awal. *Keras* digunakan untuk membuat dan melatih jaringan saraf dan tidak menawarkan banyak hal dalam hal algoritma pembelajaran mesin lainnya (Moocarme et al., 2020:34).

9. *TensorFlow*

Menurut Kapoor et al. (2022:1) *tensorflow* merupakan *library* perangkat lunak *open source* yang kuat yang dikembangkan oleh *Google Brain Team* untuk jaringan syaraf tiruan. *Tensorflow* merupakan kerangka kerja pembelajaran mesin *end-to-end* yang dirancang untuk berjalan lebih cepat pada perangkat keras yang dioptimalkan (misalnya, GPU dan TPU). *Tensorflow* adalah kerangka kerja *machine learning* yang menyeluruh dan dapat mendukung berbagai kemampuan serta tahapan proyek *machine learning* (Ganegedara, 2022:4).

Tensorflow merupakan *library open-source* yang digunakan untuk membangun dan melatih model *machine learning*, *deep learning*, dan pekerjaan yang berkaitan dengan analisis statistik lainnya. Tujuan utamanya adalah untuk membuat proses pengembangan analitik tingkat lanjut lebih mudah diakses bagi banyak pihak. Salah satu kelebihan dari *tensorflow* adalah memanfaatkan sistem GPU dan CPU serta memiliki arsitektur TPU yang dapat memungkinkan komputasi yang lebih cepat.

10. *Flask*

Flask merupakan modul bahasa *python* yang menghasilkan aplikasi halaman *web* yang memanfaatkan HTML, CSS, dan *JavaScript* (Bonney et al., 2022). *Flask* adalah *framework python* yang digunakan untuk membuat aplikasi berbasis *web* dan memungkinkan pengembang untuk menggunakan berbagai jenis ekstensi sesuai kebutuhan. *Flask* menyediakan sejumlah fungsi standar bagi pengembang untuk menambahkan beberapa *library* atau *plugin* ke dalam sebuah ekstensi (Suraya & Sholeh, 2022).

Flask adalah *micro-framework* yang dirancang untuk membuat aplikasi *web* dalam waktu singkat dengan mengimplementasikan fungsionalitas dan memberikan fleksibilitas kepada pengembang untuk menambahkan fitur yang dibutuhkan (Ghimire, 2020). *Flask* dapat diklasifikasikan ke dalam *micro-framework* karena tidak memerlukan alat atau pustaka tertentu dan memiliki basis data bawaan (Singh et al., 2019).

B. Kajian Empiris

Sebelum adanya penelitian ini, ada beberapa penelitian sebelumnya yang serupa menggunakan metode *convolutional neural network* dan *random forest* untuk sistem identifikasi tertentu. Penelitian yang dilakukan menggunakan berbagai jenis data dan model identifikasi yang sesuai dengan kebutuhan.

Penelitian serupa mengenai metode *convolutional neural network* dapat digunakan untuk mendeteksi penyakit pada daun alpukat. Penelitian

dengan judul “Implementasi Pendeteksi Penyakit pada Daun Alpukat Menggunakan Metode CNN” ini mendapatkan hasil bahwa pengujian yang dilakukan menunjukkan bahwa metode CNN dapat medeteksi penyakit pada daun alpukat dengan tingkat akurasi sebesar 80%. Pada penelitian ini, peneliti menggunakan dataset uji sebanyak 265 data yang terbagi menjadi 3 validasi, hasil yang didapatkan yaitu akurasi dengan rata-rata adalah 0,9444 pada *epoch* sebanyak 10 *epoch* (Vicky et al., 2023).

Penelitian lain yang serupa mengenai metode *convolutional neural network* dapat digunakan untuk klasifikasi penyakit citra daun anggur. Penelitian dengan judul “Klasifikasi Penyakit Citra Daun Anggur Menggunakan Model CNN-VGG16” dilakukan dengan mengubah citra RGB menjadi citra LAB, metode CNN dengan model VGG16 dapat diterapkan pada klasifikasi penyakit citra daun anggur dengan akurasi sebesar 97,25%. Pada penelitian ini, peneliti menggunakan 400 citra uji dalam dataset dan 100 citra uji di luar dataset dengan *epoch* 20, 50, dan 60 *epoch* (Hasan et al., 2021).

Penelitian lain yang juga serupa mengenai metode *convolutional neural network* dapat digunakan untuk klasifikasi penyakit tanaman jagung. Penelitian dengan judul “Klasifikasi Penyakit Tanaman Jagung Menggunakan Metode *Convolutional Neural Network* (CNN)” mendapati bahwa metode CNN sudah cukup baik dalam melakukan klasifikasi penyakit tanaman jagung dengan tingkat akurasi sebesar 97,5%. Pada penelitian ini, peneliti menggunakan 2000 dataset dengan 1600 sebagai data *training* dan 400 sebagai

data *validation* pada 100 *epoch* dengan waktu 10 menit (Iswantoro & Handayani, 2022).

Penelitian serupa mengenai metode *random forest* dapat digunakan untuk klasifikasi citra bunga menggunakan proses segmentasi. Penelitian dengan judul "Klasifikasi Citra Menggunakan Metode *Random Forest* dan *Sequential Minimal Optimization*" dengan menggunakan perhitungan *weka tools*, klasifikasi menggunakan metode *random forest* mendapat nilai akurasi sebesar 100% untuk kedua skenario (k=10 dan split 66%) sedangkan metode SMO mendapat nilai sebesar 92,68% untuk skenario split 66%. Pada penelitian ini, peneliti menggunakan 120 gambar yang sudah diolah dengan ekstraksi fitur (Saprudin et al., 2021).

Penelitian lain yang serupa mengenai metode *random forest* yaitu klasifikasi kanker. Penelitian dengan judul "Klasifikasi Kanker Menggunakan Algoritma NNGE, *Random Forest*, dan *Random Committee*" menyatakan bahwa penanganan kasus klasifikasi kanker dengan menggunakan metode NNGE, *random forest*, dan *random committee* mendapat nilai akurasi yang sangat tinggi pada beberapa alternatif penanganan. Pada penelitian ini, peneliti menggunakan 861 baris data *training* dan 100 baris data uji (Akbar, 2020).

Penelitian lain yang juga serupa mengenai metode *random forest* yaitu prediksi kemungkinan diabetes tahap awal. Penelitian dengan judul "Prediksi Kemungkinan Diabetes pada Tahap Awal Menggunakan Algoritma Klasifikasi *Random Forest*" menyatakan bahwa algoritma *random forest* dapat memprediksi kemungkinan diabetes dengan lebih akurat dibandingkan dengan

algoritma lainnya dengan nilai akurasi sebesar 97,88% dengan nilai k=10. Pada penelitian ini, peneliti menggunakan data sebanyak 520 data, 17 atribut, dan 1 kelas (Apriliah et al., 2021).

C. Kerangka Berpikir

Gambar 2. 8 Kerangka Berpikir