BAB 4
HASIL PENELITIAN

4.1 RINGKASAN HASIL PENELITIAN

Penelitian ini membandingkan performa dua algoritma klasifikasi citra digital, yaitu CNN dan SVM, dalam mendeteksi empat kondisi daun tanaman anggrek: *antraknosa*, *layu fusarium*, bercak daun, dan sehat. Dataset yang digunakan terdiri atas 200 citra, masing-masing kelas berjumlah 50 gambar, yang dilabeli oleh pakar anggrek berdasarkan pengamatan visual.

Model CNN dibangun menggunakan pendekatan *transfer learning* dengan arsitektur MobileNetV2. Bagian konvolusional dibekukan, sedangkan lapisan klasifikasi dimodifikasi untuk mengakomodasi empat kelas target. Model dilatih selama 10 *epoch* menggunakan *optimizer Adam* dengan *learning rate* 0,0001 dan proporsi data pelatihan 80:20. Hasil pelatihan menunjukkan tren peningkatan akurasi dan penurunan *loss*, namun belum mencapai performa optimal. Evaluasi akhir menunjukkan akurasi sebesar 67,50%, dengan presisi 70,45%, *recall* 67,50%, dan *F1-score* 68,21%. Kinerja yang kurang maksimal ini disebabkan oleh keterbatasan jumlah data, serta kemiripan gejala antar kelas yang menyulitkan proses klasifikasi.

Sebaliknya, model SVM menggunakan fitur berdimensi 1280 yang diekstraksi dari MobileNetV2 sebelum lapisan klasifikasi akhir. Dengan *kernel linier*, SVM menghasilkan akurasi sebesar 80,00%, presisi 81,25%, *recall* 80,00%, dan *F1-score* 79,80%. Selain lebih akurat, SVM juga menunjukkan efisiensi tinggi, dengan waktu pelatihan hanya 0,013 detik dan waktu pengujian sekitar 0,002 detik, jauh lebih cepat dibandingkan CNN.

Secara keseluruhan, pendekatan SVM dengan fitur dari *pretrained* model terbukti lebih unggul dalam hal akurasi dan efisiensi komputasi, terutama dalam konteks dataset kecil. Temuan ini memberikan rekomendasi bahwa integrasi *feature extractor* dari *deep learning* dengan algoritma klasifikasi konvensional

dapat menjadi solusi efektif untuk sistem klasifikasi penyakit tanaman berbasis citra digital.

4.2 TAHAP PENGUMPULAN DATA

Pengumpulan data dilakukan melalui observasi lapangan secara langsung di area budidaya tanaman anggrek. Data yang dikumpulkan berupa citra daun tanaman anggrek yang merepresentasikan empat kategori kondisi, yaitu *Antraknosa,* Bercak Daun*, Layu Fusarium*, dan Sehat. Seluruh citra diperoleh menggunakan kamera *smartphone* tanpa pencahayaan tambahan, agar kondisi visual citra tetap mencerminkan keadaan alami objek.

Pengambilan gambar dilakukan dengan mempertimbangkan variasi sudut pandang, jarak pengambilan, dan pencahayaan alami untuk memperoleh data yang representatif. Setiap gambar difokuskan pada permukaan daun yang menunjukkan ciri-ciri tertentu, seperti adanya bercak, perubahan warna, atau gejala kelayuan, guna memastikan bahwa citra dapat digunakan sebagai dasar klasifikasi visual.

Jumlah data yang diperoleh sebanyak 200 citra, yang masing-masing terbagi merata ke dalam empat kelas. Setiap kelas terdiri dari 50 citra yang telah melalui proses pelabelan. Proses pelabelan dilakukan secara manual oleh seorang pakar tanaman anggrek yang memiliki pengalaman lebih dari 25 tahun dalam perawatan dan budidaya anggrek. Klasifikasi dilakukan berdasarkan pengamatan visual terhadap karakteristik morfologis daun, seperti bentuk, warna, serta pola kerusakan yang muncul pada permukaan daun. Hasil dari proses ini berupa dataset citra digital yang telah dilabeli sesuai kelas masing-masing dan siap digunakan dalam tahap berikutnya.

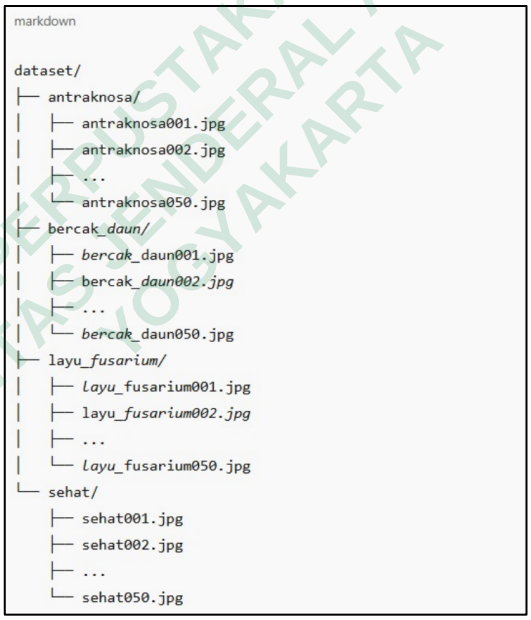
4.3 TAHAP PRA – PEMROSESAN CITRA

Pra-pemrosesan citra merupakan tahap awal yang bertujuan untuk mempersiapkan data agar memenuhi standar format dan struktur yang diperlukan oleh algoritma klasifikasi. Tahapan ini mencakup penataan direktori data, penyesuaian dimensi citra, normalisasi nilai piksel, serta penerapan *augmentasi* data untuk memperkaya variasi visual dalam pelatihan model CNN. Selain itu,

proses ekstraksi fitur dilakukan menggunakan arsitektur pralatih MobileNetV2 untuk menghasilkan representasi numerik citra sebagai masukan pada model SVM.

4.3.1 Struktur Dataset dan Pelabelan

Data citra yang telah dikumpulkan disusun ke dalam struktur direktori bertingkat untuk memfasilitasi proses *pra-pemrosesan* dan pelatihan model klasifikasi. Direktori utama terdiri atas empat subdirektori yang masing-masing mewakili satu kelas, yaitu *Antraknosa*, Bercak Daun, *Layu Fusarium*, dan Sehat. Setiap subdirektori memuat sebanyak 50 citra daun tanaman anggrek yang telah diberi label sesuai dengan kategori masing-masing, sehingga total keseluruhan data berjumlah 200 citra digital. Struktur direktori ini divisualisasikan pada Gambar 4.1. Dalam struktur tersebut, setiap folder kelas ditempatkan dalam direktori induk dan dinamai sesuai dengan nama kategori penyakit atau kondisi sehat.



Gambar 4.1 Struktur Direktori Dataset Citra Tanaman Anggrek

Berkas citra di dalam setiap subdirektori diberi nama secara berurutan, misalnya antraknosa001.jpg, bercakdaun001.jpg, layufusarium001.jpg, dan sehat001.jpg. Penamaan ini dilakukan secara sistematis dan konsisten untuk memastikan keteraturan saat proses pemanggilan data oleh pustaka pemrosesan citra digital seperti `ImageDataGenerator` pada TensorFlow Keras.

Pelabelan dilakukan secara manual oleh seorang pakar tanaman anggrek yang memiliki pengalaman lebih dari 25 tahun dalam bidang budidaya, perawatan, serta identifikasi penyakit tanaman anggrek. Penentuan label didasarkan pada pengamatan visual terhadap karakteristik morfologis daun, yang mencakup bentuk, warna, pola bercak, dan gejala kelayuan. Citra yang telah diberi label secara akurat kemudian digunakan sebagai data masukan pada tahap *pra-pemrosesan* dan pelatihan model klasifikasi.

4.3.2 Resize Citra

Seluruh citra daun tanaman anggrek yang dikumpulkan memiliki dimensi dan resolusi bervariasi akibat perbedaan perangkat dan kondisi pengambilan. Untuk memastikan kompatibilitas dengan MobileNetV2, seluruh citra diubah ukurannya menjadi 224 × 224 piksel yaitu dimensi standar yang digunakan pada saat pelatihan model tersebut dengan dataset `ImageNet`.

Pada implementasi CNN, *resize* citra dilakukan melalui metode `flow_from_directory` yang juga berfungsi untuk memuat *dataset* dan menghasilkan batch data secara otomatis. Kode program yang digunakan sebagai berikut:

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

img_size = (224, 224)
datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2,
)
train_gen = datagen.flow_from_directory(
    'dataset',
    target_size=img_size,
    batch_size=16,
    class_mode='categorical',
    subset='training',
    shuffle=True
)
```

Penjelasan kode:

1. `img_size` menentukan ukuran target citra sebesar 224 × 224 piksel.
2. `ImageDataGenerator` memuat data dan menerapkan normalisasi nilai piksel (`rescale=1./255`) serta membagi dataset menjadi 80% data pelatihan dan 20% data validasi (`validation_split=0.2`)

3. `flow_from_directory` membaca citra dari direktori, menyesuaikan ukuran citra ke dimensi target, serta menghasilkan *batch data* dengan jumlah data masing - masing *batch* adalah 16 data
4. `class_mode='categorical'` menunjukkan bahwa label dikodekan dalam format *one-hot encoding*

Pada SVM, proses *resize* dilakukan pada saat setiap citra diproses secara individual untuk diekstraksi fitur-fiturnya menggunakan arsitektur MobileNetV2. Fungsi `load_img` dari Keras digunakan untuk memuat gambar sekaligus mengubah ukurannya ke dimensi target. Selanjutnya, fungsi `img_to_array` mengonversi citra ke dalam format array agar dapat diproses lebih lanjut. Kode program yang digunakan untuk *resize* citra pada SVM adalah sebagai berikut:

```
from tensorflow.keras.preprocessing.image import load_img, img_to_array
import numpy as np

img = load_img(img_path, target_size=(224, 224))
img_array = img_to_array(img)
img_array = np.expand_dims(img_array, axis=0)
```

Penjelasan kode program:

1. Fungsi `load_img` untuk memuat gambar dari jalur file yang ditentukan (*img_path*) dan mengubah ukurannya ke 224 × 224 piksel.
2. Fungsi `img_to_array` kemudian mengonversi citra menjadi array, dan `expand_dims` digunakan untuk menambahkan dimensi *batch* pada array agar kompatibel dengan input model *pretrained* MobileNetV2.

Tahap penyesuaian ukuran ini menghasilkan dataset dengan dimensi citra yang seragam sehingga siap digunakan pada tahap normalisasi dan pelatihan model klasifikasi.

4.3.3 Normalisasi Citra

Setelah penyesuaian ukuran dilakukan, setiap citra daun tanaman anggrek melalui tahap normalisasi nilai piksel. Normalisasi ini bertujuan untuk memastikan data masukan memiliki skala yang seragam, sehingga sesuai dengan persyaratan arsitektur model yang digunakan. Nilai piksel citra yang semula berada pada rentang [0, 255] diubah menjadi skala yang lebih kecil sesuai kebutuhan model.

Pada implementasi CNN, normalisasi dilakukan menggunakan parameter `rescale` pada objek `ImageDataGenerator`. Parameter ini membagi seluruh nilai piksel dengan angka 255 sehingga rentang nilai piksel berada pada [0, 1]. Potongan kode program yang digunakan sebagai berikut:

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2,
)
```

Fungsi `rescale` memastikan setiap nilai piksel yang dimuat melalui `flow_from_directory` telah dinormalisasi sebelum digunakan dalam pelatihan jaringan saraf.

Pada implementasi SVM, normalisasi dilakukan dengan fungsi `preprocess_input` yang disediakan oleh modul MobileNetV2. Fungsi ini menyesuaikan nilai piksel dengan rentang dan distribusi yang digunakan saat pelatihan awal MobileNetV2. Kode program yang digunakan sebagai berikut:

```
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input

img_array = preprocess_input(img_array)
```

Fungsi `preprocess_input` mengubah nilai piksel menjadi rentang [-1, 1] dan melakukan *zero-centering* sesuai dengan *preprocessing* pada *dataset* `ImageNet`. Hal ini penting agar fitur yang diekstraksi dari MobileNetV2 konsisten dengan bobot *pretrained model*.

4.3.4 Augmentasi Data pada CNN

Pada tahap pelatihan model CNN, jumlah data pelatihan yang terbatas dapat mengurangi kemampuan generalisasi model terhadap data baru. Untuk mengatasi keterbatasan tersebut, diterapkan teknik *augmentasi* data guna memperluas keragaman citra pelatihan secara buatan. *Augmentasi* data merupakan proses transformasi acak pada citra asli untuk menghasilkan variasi visual tambahan, sehingga model dapat mempelajari pola dari representasi data yang lebih beragam.

Pendekatan ini untuk mengurangi risiko *overfitting* dan meningkatkan ketahanan model terhadap variasi sudut pandang, orientasi, serta kondisi pencahayaan.

Implementasi *augmentasi* data dilakukan dengan memanfaatkan fungsi `ImageDataGenerator` dari pustaka TensorFlow Keras. Potongan kode program yang digunakan untuk konfigurasi augmentasi dijelaskan sebagai berikut:

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)
```

Penjelasan kode program:

1. `rescale=1./255` melakukan normalisasi nilai piksel dari rentang [0, 255] ke [0, 1] untuk memastikan kestabilan komputasi selama pelatihan.
2. `validation_split=0.2` menetapkan proporsi 20% dari dataset sebagai data validasi dan 80% sisanya sebagai data pelatihan.
3. `rotation_range=20` memberikan rotasi acak pada citra hingga batas maksimum 20°, yang mensimulasikan variasi sudut pengambilan gambar.
4. `width_shift_range=0.2` dan `height_shift_range=0.2` menggeser citra secara horizontal dan vertikal hingga 20% dari dimensi gambar, untuk meniru variasi posisi objek dalam bingkai.
5. `shear_range=0.2` menerapkan transformasi shearing dengan sudut maksimum 20% untuk memperkaya variasi bentuk geometri objek.
6. `zoom_range=0.2` melakukan pembesaran acak hingga 20% guna meniru variasi jarak antara kamera dan objek.
7. `horizontal_flip=True` membalik citra secara horizontal, sehingga memperluas orientasi arah objek dalam dataset.

8. `fill_mode='nearest'` mengisi piksel kosong yang muncul akibat transformasi dengan nilai piksel tetangga terdekat.

Penerapan augmentasi ini menghasilkan *dataset* pelatihan yang lebih bervariasi secara visual, sehingga model CNN memiliki kemampuan generalisasi yang lebih baik terhadap citra daun anggrek dengan variasi bentuk, posisi, dan pencahayaan.

4.3.5 Ekstraksi Fitur untuk Support Vector Machine (SVM)

SVM tidak dapat memproses citra secara langsung karena memerlukan data dalam bentuk vektor fitur berdimensi tetap. Oleh karena itu, tahap ekstraksi fitur diperlukan untuk mengubah citra daun anggrek menjadi representasi numerik yang dapat digunakan sebagai masukan pada algoritma SVM.

Pada penelitian ini, ekstraksi fitur dilakukan dengan memanfaatkan arsitektur *pretrained* MobileNetV2 dari pustaka TensorFlow Keras. MobileNetV2 dipilih karena merupakan model yang efisien dan telah dilatih pada *dataset* *ImageNet* dengan lebih dari 1 juta citra, sehingga mampu menghasilkan fitur representatif untuk berbagai objek visual, termasuk tanaman. Lapisan klasifikasi akhir pada MobileNetV2 dihilangkan (`include_top=False`), sedangkan lapisan *pooling global* (*global average pooling*) digunakan untuk mereduksi dimensi output menjadi vektor fitur berdimensi 1280. Potongan kode program yang digunakan untuk proses ekstraksi fitur dijelaskan sebagai berikut:

```
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import load_img, img_to_array
import numpy as np

# Inisialisasi model MobileNetV2 tanpa lapisan klasifikasi akhir
feature_extractor = MobileNetV2(weights='imagenet', include_top=False,
pooling='avg', input_shape=(224, 224, 3))

# Memuat dan memproses citra
img = load_img(img_path, target_size=(224, 224))
img_array = img_to_array(img)
img_array = np.expand_dims(img_array, axis=0)
img_array = preprocess_input(img_array)

# Ekstraksi fitur
features = feature_extractor.predict(img_array)
```

Penjelasan kode program:

1. Inisialisasi model dilakukan dengan menggunakan MobileNetV2 yang sudah memiliki bobot hasil pelatihan dari dataset `ImageNet`. Parameter `include_top=False` digunakan untuk menghapus lapisan klasifikasi terakhir dari model, sedangkan `pooling='avg'` digunakan agar output dari model berupa vektor fitur berdimensi 1280, yang merepresentasikan ciri-ciri penting dari gambar.
2. Proses pemuatan dan praproses citra dimulai dengan fungsi `load_img()` yang digunakan untuk membaca gambar dari jalur file dan mengubah ukurannya menjadi 224 × 224 piksel agar sesuai dengan ukuran input yang diterima oleh MobileNetV2.
3. Fungsi `img_to_array()` digunakan untuk mengubah gambar yang telah dimuat menjadi array NumPy agar dapat diproses lebih lanjut oleh model.
4. Fungsi `np.expand_dims()` digunakan untuk menambahkan satu dimensi baru pada array gambar, yaitu dimensi batch, sehingga format input sesuai dengan yang dibutuhkan oleh model deep learning.
5. Fungsi `preprocess_input()` digunakan untuk melakukan normalisasi piksel gambar sesuai dengan standar yang digunakan oleh MobileNetV2, agar hasil ekstraksi fitur lebih akurat dan stabil.
6. Setelah citra diproses, fungsi `predict()` dari MobileNetV2 digunakan untuk mengekstraksi fitur dari gambar. Hasilnya adalah vektor berdimensi 1280 yang berisi representasi numerik dari fitur visual penting pada gambar tersebut. Vektor ini kemudian digunakan sebagai input untuk model SVM.

Hasil dari tahap ini adalah sekumpulan vektor fitur berdimensi 1280 yang merepresentasikan masing-masing citra daun anggrek dalam *dataset*. Vektor-vektor fitur ini kemudian digunakan sebagai masukan pada algoritma SVM untuk proses pelatihan dan evaluasi model klasifikasi.

4.3.6 Pembagian Data untuk CNN dan SVM

Pembagian *dataset* menjadi data pelatihan dan data validasi merupakan tahap penting untuk mengevaluasi kinerja model secara objektif. Pada penelitian

ini, kedua algoritma CNN dan SVM menggunakan proporsi pembagian data yang sama, yaitu 80% data untuk pelatihan dan 20% data untuk validasi. Pada implementasi CNN, pembagian dataset dilakukan secara otomatis menggunakan parameter `validation_split` pada fungsi `ImageDataGenerator`. Parameter ini memisahkan 20% dari data yang dimuat sebagai data validasi, sementara 80% sisanya digunakan untuk pelatihan model. Potongan kode untuk pembagian data pada CNN dijelaskan sebagai berikut:

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Konfigurasi ImageDataGenerator dengan pembagian data
datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2
)
train_gen = datagen.flow_from_directory(
    'dataset',
    target_size=(224, 224),
    batch_size=16,
    class_mode='categorical',
    subset='training',
    shuffle=True
)
val_gen = datagen.flow_from_directory(
    'dataset',
    target_size=(224, 224),
    batch_size=16,
    class_mode='categorical',
    subset='validation',
    shuffle=False
)
```


Penjelasan kode program:

1. `validation_split=0.2` memisahkan 20% data sebagai validasi.
2. `subset='training'` dan `subset='validation'` digunakan untuk mengarahkan generator agar memuat data sesuai dengan kategori (pelatihan atau validasi).
3. `shuffle=True` diterapkan pada data pelatihan untuk mengacak urutan data setiap epoch. Data validasi tidak diacak (`shuffle=False`) agar evaluasi tetap konsisten.

Pada algoritma SVM, pembagian data dilakukan menggunakan fungsi `train_test_split` dari pustaka Scikit-learn. *Dataset* vektor fitur yang telah

diekstraksi sebelumnya dipecah menjadi dua subset dengan proporsi yang sama seperti CNN, yaitu 80% untuk pelatihan dan 20% untuk pengujian. Potongan kode untuk pembagian data pada SVM dijelaskan sebagai berikut:

```
from sklearn.model_selection import train_test_split

# Pembagian dataset vektor fitur
X_train, X_test, y_train, y_test = train_test_split(
    X, y_encoded,
    test_size=0.2,
    random_state=42,
    stratify=y_encoded
)
```

Penjelasan kode program:

1. `test_size=0.2` menentukan 20% data sebagai data uji, sedangkan 80% sisanya sebagai data pelatihan.
2. `random_state=42` digunakan untuk memastikan hasil pembagian data konsisten setiap eksekusi.
3. `stratify=y_encoded` menjaga distribusi kelas agar tetap proporsional antara data pelatihan dan data pengujian.

Dengan pembagian data yang seragam ini, kedua algoritma memperoleh kondisi evaluasi yang setara, sehingga memungkinkan perbandingan kinerja model yang lebih objektif.

4.4 PELATIHAN DAN EVALUASI MODEL

4.4.1 Pelatihan Model

Tahap pelatihan model merupakan langkah penting dalam pengoptimalan parameter jaringan untuk menghasilkan prediksi yang akurat. Model CNN dibangun dengan menggunakan arsitektur MobileNetV2 yang diterapkan dengan pendekatan *transfer learning*. Pada model ini, lapisan *feature extractor* pada MobileNetV2 dibekukan (*frozen*) untuk mempertahankan bobot yang diperoleh dari pelatihan sebelumnya pada *dataset* `ImageNet`, sementara lapisan klasifikasi akhir ditambahkan secara khusus untuk menyesuaikan dengan jumlah kelas pada *dataset* yang digunakan.

Model dilatih selama 10 *epoch* menggunakan algoritma optimisasi `Adam` dengan *learning rate* sebesar 0,0001. Data pelatihan dan validasi dihasilkan secara otomatis melalui *object generator* yang telah dikonfigurasi sebelumnya. Berikut adalah potongan kode yang digunakan untuk pelatihan model CNN:

```
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.layers import GlobalAveragePooling2D,
     DenseDropout

# Inisialisasi model dasar
base_model = MobileNetV2(
     weights='imagenet',
     include_top=False,
     input_shape=(224, 224, 3)
)

base_model.trainable = False

# Menambah lapisan klasifikasi
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dropout(0.3)(x)
x = Dense(128, activation='relu')(x)
x = Dropout(0.2)(x)
predictions = Dense(train_gen.num_classes, activation='softmax')(x)

# Membentuk model akhir
model = Model(inputs=base_model.input, outputs=predictions)


# Kompilasi model
model.compile(
     optimizer=Adam(learning_rate=0.0001),
     loss='categorical_crossentropy',
     metrics=['accuracy']
)

# Pelatihan model
history = model.fit(train_gen, validation_data=val_gen, epochs=10)
```

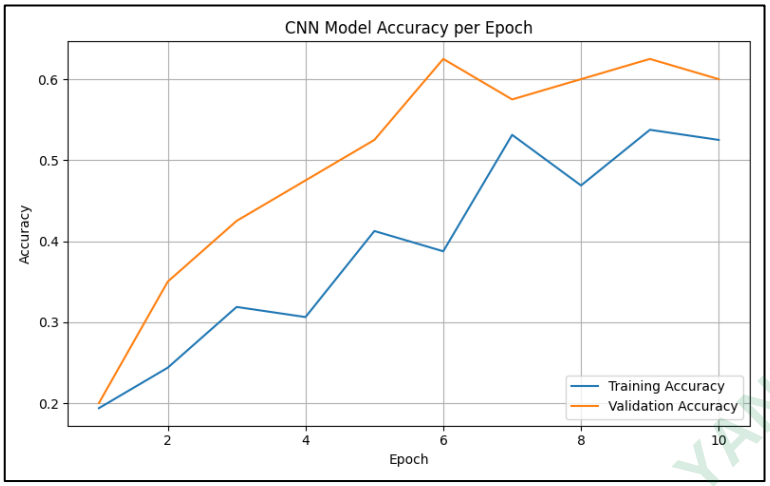

Penjelasan kode program:

1. Model MobileNetV2 digunakan sebagai *feature extractor* dengan bobot yang telah dipelajari pada *dataset* `ImageNet`, dengan menghilangkan lapisan klasifikasi akhir *(parameter include_top=False)* agar model hanya

digunakan untuk mengekstraksi fitur dari citra input, tanpa melakukan klasifikasi langsung.

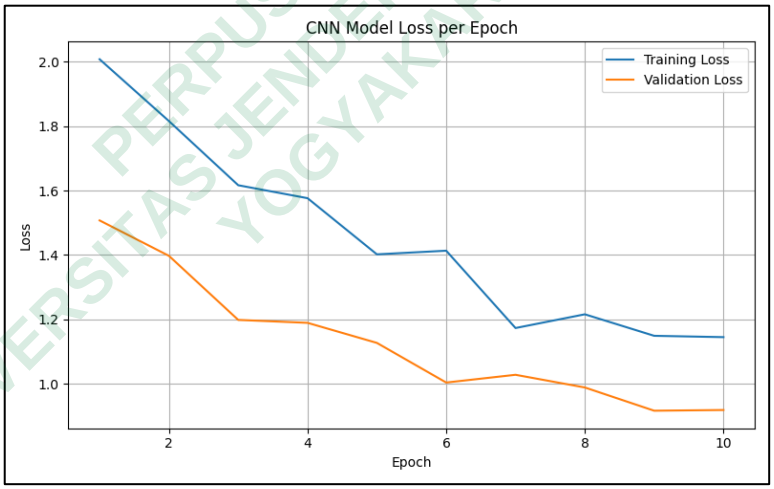
2. `Global Average Pooling` diterapkan pada *output* dari *feature extractor* untuk mereduksi dimensi fitur dengan merata-ratakan nilai-nilai fitur pada tiap-tiap *channel* dalam *feature map*, menghasilkan vektor fitur berdimensi lebih rendah yang tetap mempertahankan informasi penting dari citra.
3. `Dropout` diterapkan pada dua titik dalam model untuk mencegah *overfitting* yang bekerja dengan mematikan (menonaktifkan) sejumlah unit secara acak pada setiap *epoch* pelatihan.
4. `Dense layer` dengan aktivasi ReLU digunakan untuk memproses *output* dari lapisan *pooling*. Lapisan akhir berupa `dense layer` dengan aktivasi Softmax digunakan untuk menghasilkan *output* berupa probabilitas setiap kelas, yang jumlahnya selalu 1. Aktivasi softmax diterapkan untuk mengoptimalkan model dalam menangani tugas klasifikasi multi-kelas.
5. `Adam optimizer` digunakan untuk menyesuaikan *learning rate* selama pelatihan. *Learning rate* diatur pada nilai 0,0001. Fungsi `loss` yang digunakan adalah `categorical_crossentropy`.
6. Fungsi `fit()` digunakan untuk melatih model dengan data pelatihan, sementara kinerja model dipantau menggunakan data validasi. Proses pelatihan dilakukan selama 10 *epoch*, yang memungkinkan pemantauan akurasi dan loss pada setiap *epoch* untuk mengevaluasi kinerja model serta mendeteksi potensi *overfitting*.

Setelah pelatihan selama 10 *epoch*, model CNN menghasilkan rekaman akurasi dan *loss* pada setiap *epoch* untuk data pelatihan dan validasi. Grafik ini digunakan untuk mengevaluasi konsistensi peningkatan kinerja model serta mendeteksi kemungkinan terjadinya *overfitting*.



Gambar 4.2 Grafik Akurasi per Epoch Model CNN

Gambar 4.2 menunjukkan grafik akurasi model pada data pelatihan dan validasi, yang menunjukkan peningkatan akurasi seiring bertambahnya jumlah *epoch*. Hal ini mengindikasikan bahwa model berhasil mempelajari fitur-fitur penting dari citra pelatihan.



Gambar 4.3 Grafik Loss per Epoch Model CNN

Gambar 4.3 memperlihatkan grafik *loss* model, yang menunjukkan penurunan nilai *loss* seiring berjalannya waktu, yang menandakan bahwa kesalahan prediksi model menurun secara bertahap. Tidak terdapat indikasi *overfitting*, karena nilai akurasi dan *loss* pada data validasi tidak menyimpang jauh dari data pelatihan.

Berbeda dengan CNN, algoritma SVM tidak memerlukan arsitektur jaringan saraf. Sebagai gantinya, model dilatih menggunakan vektor fitur

berdimensi 1280 yang dihasilkan dari ekstraksi fitur MobileNetV2. SVM menggunakan *kernel linier* untuk memisahkan kelas secara optimal pada ruang fitur berdimensi tinggi. Potongan kode program untuk pelatihan SVM adalah sebagai berikut:

```
# Inisialisasi dan pelatihan model SVM
clf = svm.SVC(kernel='linear')
clf.fit(X_train, y_train)
```

Penjelasan kode program:

1. Inisialisasi Model: Objek SVM dibuat dengan `kernel linier` untuk memproyeksikan data ke dalam ruang fitur dengan margin maksimal.
2. Objek *SVM* dibuat dengan `kernel linier`, untuk memproyeksikan data ke dalam ruang fitur dengan margin maksimal antara kelas-kelas yang ada.
3. Fungsi `fit()` digunakan untuk melatih model SVM dengan dataset pelatihan `X_train` dan label `y_train`.

4.4.2 Evaluasi Model

Setelah tahap pelatihan selesai, evaluasi dilakukan untuk mengukur kinerja dua model klasifikasi, yaitu CNN dan SVM, dalam mengklasifikasikan citra daun anggrek ke dalam empat kelas yakni *antraknosa, layu fusarium, bercak daun,* dan *sehat.* Evaluasi ini bertujuan untuk menilai efektivitas masing-masing model dalam memprediksi kelas-kelas tersebut berdasarkan metrik akurasi, presisi, *recall*, dan *F1-score*.

Model CNN dievaluasi menggunakan data validasi yang terdiri dari 20% dari total dataset, yang dipisahkan secara otomatis menggunakan parameter `validation_split` pada `ImageDataGenerator`. Proses evaluasi meliputi dua langkah utama, yaitu prediksi label menggunakan data validasi dan perhitungan metrik evaluasi dengan menggunakan fungsi `classification_report` dari pustaka `scikit-learn`. Potongan kode program untuk evaluasi model CNN sebagai berikut:

```
from sklearn.metrics import classification_report

val_gen.reset()
y_pred_probs = model.predict(val_gen)
```

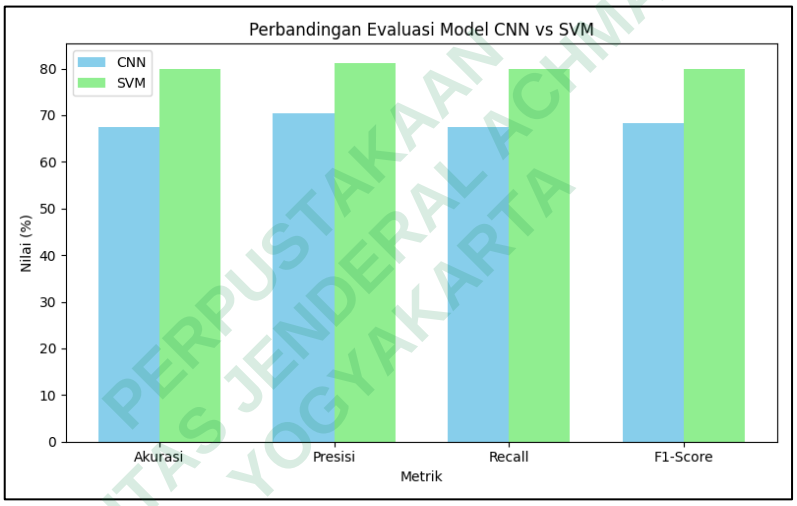
```
y_pred = np.argmax(y_pred_probs, axis=1)
y_true = val_gen.classes

report_cnn = classification_report(
    y_true, y_pred,
    target_names=val_gen.class_indices.keys(),
    output_dict=True
)
cnn_metrics = {
    "accuracy": report_cnn["accuracy"],
    "precision": np.mean([report_cnn[k]["precision"] for k in
val_gen.class_indices.keys()]),
    "recall": np.mean([report_cnn[k]["recall"] for k in
val_gen.class_indices.keys()]),
    "f1_score": np.mean([report_cnn[k]["f1-score"] for k in
val_gen.class_indices.keys()])
}
```

Hasil evaluasi model CNN menunjukkan bahwa model mencapai akurasi 67.50%, presisi 70.45%, *recall* 67.50%, dan *F1-score* 68.21%. Meskipun *dataset* yang digunakan terbatas, model CNN berhasil menunjukkan kinerja yang cukup baik dalam klasifikasi citra tanaman anggrek. Penggunaan MobileNetV2 dengan pendekatan *transfer learning* terbukti memberikan hasil yang baik dalam mengekstrak fitur dari citra tanaman anggrek, meskipun dataset yang digunakan terbatas.

Model SVM dievaluasi menggunakan data pengujian yang terpisah dari data pelatihan dengan menggunakan metrik yang sama seperti pada model CNN. Potongan kode untuk evaluasi model SVM dapat dilihat sebagai berikut.

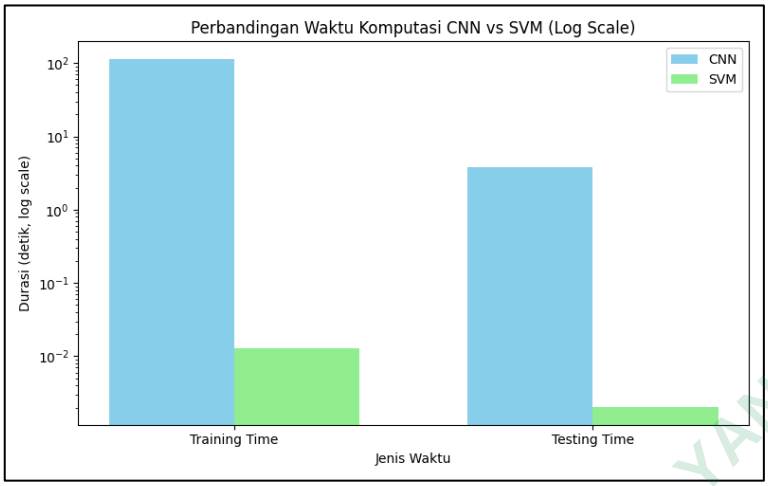
```
y_pred = clf.predict(X_test)
report_svm = classification_report(
    y_test, y_pred,
    target_names=le.classes_,
    output_dict=True
)
svm_metrics = {
    "accuracy": report_svm["accuracy"],
    "precision": np.mean([report_svm[k]["precision"] for k in
le.classes_]),
    "recall": np.mean([report_svm[k]["recall"] for k in le.classes_]),
    "f1_score": np.mean([report_svm[k]["f1-score"] for k in
le.classes_])
}
```

Hasil evaluasi model SVM menunjukkan bahwa model ini mencapai akurasi sebesar 80.00%, presisi 81.25%, *recall* 80.00%, dan *F1-score* 79.80%. Hasil evaluasi menunjukkan bahwa meskipun model SVM tidak menggunakan arsitektur jaringan saraf, model ini mampu mencapai akurasi yang lebih tinggi dibandingkan dengan CNN. Penggunaan MobileNetV2 untuk ekstraksi fitur memungkinkan SVM menghasilkan performa yang baik dalam klasifikasi citra tanaman anggrek, meskipun SVM merupakan model yang lebih sederhana dan tidak melibatkan pelatihan jaringan saraf dalam prosesnya. Untuk memvisualisasikan perbedaan performa antara CNN dan SVM, dilakukan perbandingan metrik evaluasi yang dapat dilihat pada Gambar 4.4.



Gambar 4.4 Perbandingan Metrik Evaluasi Model CNN dan SVM

Gambar 4.4 menunjukkan bahwa model SVM memiliki nilai yang lebih tinggi dibandingkan dengan CNN pada metrik akurasi, presisi, *recall*, dan *F1-score*. Dalam hal efisiensi komputasi, waktu pelatihan dan pengujian untuk masing-masing model dapat dilihat pada Gambar 4.5.



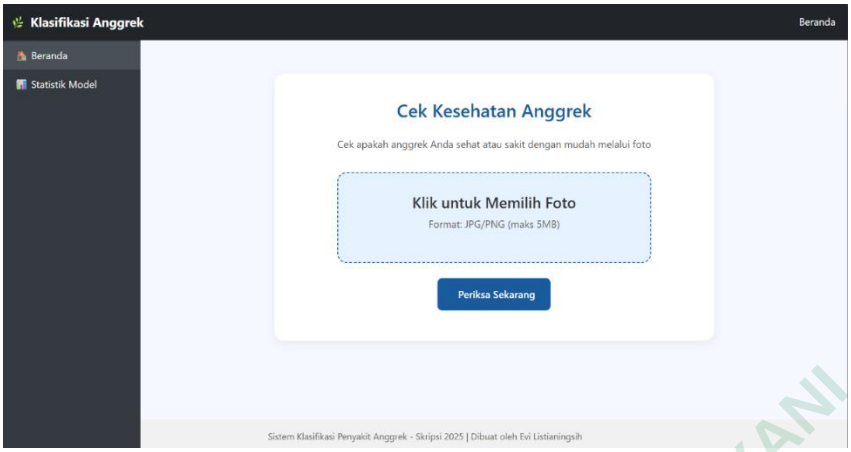
Gambar 4.5 Perbandingan Efisiensi Komputasi pada Model CNN dan SVM

Model CNN membutuhkan waktu pelatihan sekitar ±114 detik dan waktu pengujian ±3,8 detik. Sementara itu, model SVM menunjukkan waktu pelatihan yang sangat singkat, yaitu sekitar ±0,013 detik, dan waktu pengujian hanya ±0,002 detik. Perbedaan ini mengindikasikan bahwa SVM lebih efisien dalam hal komputasi dibandingkan dengan CNN.

4.5 VISUALISASI SISTEM

4.5.1 Halaman Beranda

Halaman beranda merupakan antarmuka awal yang digunakan untuk mengunggah citra tanaman anggrek yang akan diklasifikasikan. Sistem menyediakan fitur unggah gambar yang mendukung format JPG dan PNG dengan batas ukuran maksimum sebesar 5 MB. Setelah citra dipilih melalui tombol unggah yang tersedia, pengguna dapat memulai proses klasifikasi dengan menekan tombol "Periksa Sekarang", sebagaimana ditampilkan pada Gambar 4.6.



Gambar 4.6 Halaman Beranda Sebelum Upload Foto

Berikut penjelasan kode untuk tombol "Periksa Sekarang":

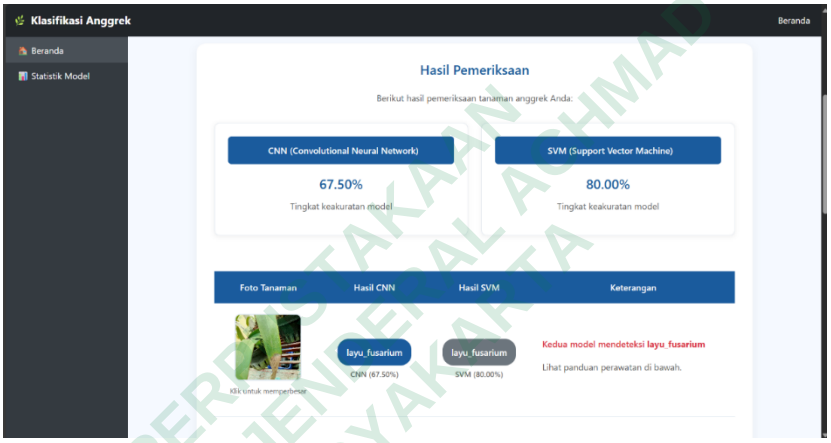
```
<div style="text-align: center; margin-top: 1rem;">
  <button type="submit" class="btn btn-primary action-btn"
style="background-color: #1a5b9d;">
    <i class="fas fa-search mr-2"></i> Periksa Sekarang
  </button>
</div>Penjelasan kode program:
```

Penjelasan kode program:

1. `<div style="text-align: center; margin-top: 1rem;">` Elemen `<div>` ini digunakan untuk membungkus tombol agar tampil rata tengah. Properti `margin-top: 1rem` memberikan jarak vertikal dari elemen atasnya.
2. `<button type="submit" class="btn btn-primary action-btn" style="background-color: #1a5b9d;">` Elemen `<button>` bertipe `submit`, yang berfungsi mengirimkan form ke server saat ditekan. Kelas `btn` dan `btn-primary` berasal dari Bootstrap untuk tampilan modern, sedangkan `action-btn` digunakan untuk gaya kustom tambahan. Warna tombol diatur secara manual menjadi biru tua `#1a5b9d`.
3. `<i class="fas fa-search mr-2"></i>` Menampilkan ikon kaca pembesar (search) menggunakan Font Awesome. Kelas `mr-2` memberikan jarak ke kanan agar tidak terlalu rapat dengan teks tombol.

4. `Periksa Sekarang` Merupakan teks yang muncul pada tombol, memberi tahu pengguna bahwa tombol ini digunakan untuk memulai proses pemeriksaan gambar yang telah diunggah.
5. `</button>` Menutup elemen tombol.
6. `</div>` Menutup elemen pembungkus tombol.

Setelah proses unggah dan klasifikasi selesai, sistem menampilkan hasil klasifikasi secara langsung pada halaman yang sama. Informasi yang disajikan meliputi nilai akurasi dari dua model klasifikasi yang digunakan, yaitu CNN dan SVM.



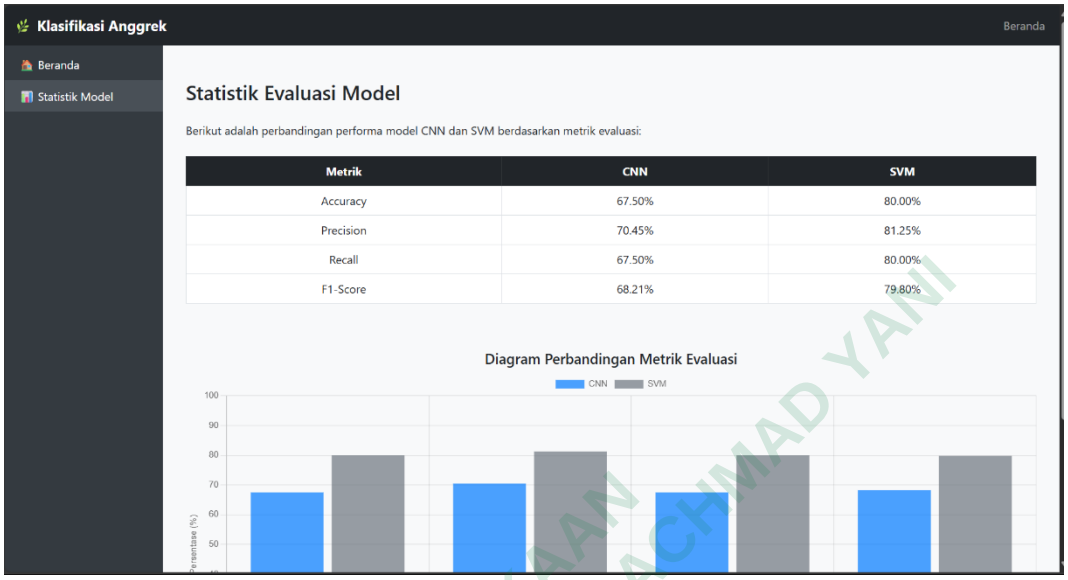
Gambar 4.7 Halaman Beranda Setelah Upload Foto

Berdasarkan ilustrasi pada Gambar 4.7, model CNN menghasilkan tingkat akurasi sebesar 67.50%, sedangkan model SVM mencapai tingkat akurasi sebesar 80.00%. Selain akurasi model, sistem juga menampilkan hasil diagnosis kondisi tanaman berdasarkan klasifikasi yang dilakukan. Sebagai contoh, ditampilkan bahwa tanaman terdeteksi mengalami gejala *layu fusarium*. Bersamaan dengan hasil tersebut, sistem menyediakan informasi tambahan berupa rekomendasi tindakan perawatan sesuai kondisi yang terdeteksi.

4.5.2 Halaman Statistik Model

Halaman ini menampilkan perbandingan performa dua model klasifikasi, yaitu CNN dan SVM, berdasarkan metrik evaluasi berupa akurasi, presisi, *recall*, dan *F1-score*. Nilai evaluasi disajikan dalam tabel yang merangkum capaian masing-masing model secara kuantitatif. Selain itu, disediakan diagram batang

yang memperjelas. Sebagai pelengkap, diagram batang turut ditampilkan guna memvisualisasikan selisih performa antar model.



Gambar 4.8 Halaman Statistik Evaluasi Model

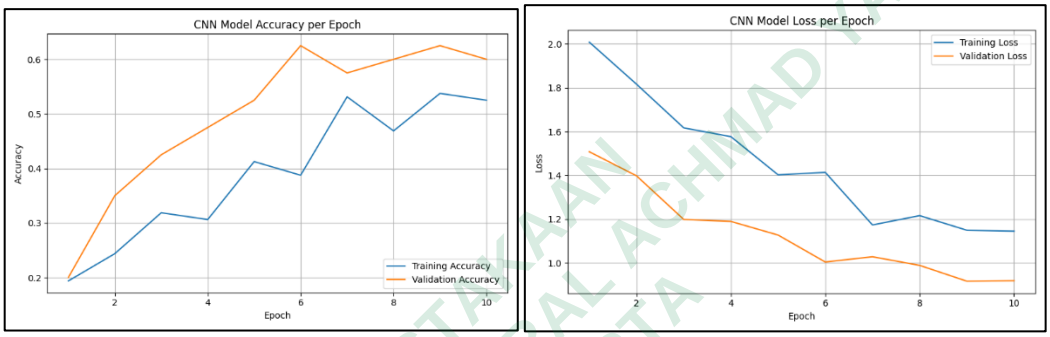
Berdasarkan hasil yang ditampilkan pada Gambar 4.8, algoritma SVM menunjukkan keunggulan pada seluruh metrik dengan akurasi sebesar 80.00%, presisi 81.25%, *recall* 80.00%, dan *F1-score* 79.80%, melampaui capaian model CNN dalam setiap kategori evaluasi.

4.6 PEMBAHASAN

Penelitian ini bertujuan membandingkan performa dua algoritma klasifikasi citra, yaitu CNN dan SVM, dalam mendeteksi kondisi tanaman anggrek berdasarkan citra visual. *Dataset* yang digunakan terdiri atas 200 citra tanaman anggrek, yang terbagi ke dalam empat kelas yakni *antraknosa, layu fusarium, bercak daun,* dan *sehat*, masing-masing sebanyak 50 citra. Setiap gambar telah diberi label secara manual oleh seorang pakar tanaman anggrek berpengalaman lebih dari 25 tahun dalam perawatan dan budidaya tanaman tersebut.

Model CNN dibangun dengan pendekatan *transfer learning* menggunakan arsitektur MobileNetV2. Lapisan awal (*konvolusional*) dibekukan (*frozen*) untuk mempertahankan bobot hasil pelatihan dari *dataset* `ImageNet`. Sementara itu, struktur lapisan akhir dimodifikasi agar sesuai dengan kebutuhan klasifikasi empat

kelas target, yaitu dengan menambahkan *Global Average Pooling*, *Dropout*, dan *Dense layer* sebagai lapisan klasifikasi. Model ini dilatih selama 10 epoch dengan algoritma optimisasi `Adam` dan *learning rate* sebesar 0,0001. Data pelatihan dan validasi dibagi secara otomatis oleh `ImageDataGenerator`, dengan 20% data digunakan untuk validasi dan 80% untuk pelatihan. Hasil pelatihan memperlihatkan peningkatan akurasi dan penurunan *loss* secara bertahap, tanpa tanda-tanda *overfitting*, meskipun jumlah data relatif terbatas. Hasil pelatihan CNN dapat dilihat pada Gambar 4.9.



Gambar 4.9 Grafik Akurasi dan Loss per Epoch pada CNN

Berdasarkan hasil evaluasi, model CNN memperoleh akurasi sebesar 67.50%, dengan presisi 70.45%, recall 67.50%, dan F1-score 68.21%. Nilai-nilai tersebut menunjukkan bahwa model sudah cukup mampu mengenali pola visual dari citra tanaman dan mengklasifikasikannya ke dalam kelas yang sesuai. Namun, akurasi tersebut belum optimal, terutama dalam membedakan gejala penyakit yang memiliki kemiripan visual, seperti bercak daun dan antraknosa. Keterbatasan utama terletak pada ukuran dataset yang kecil untuk standar model *deep learning*. Walaupun teknik *augmentasi* gambar telah diterapkan, variasi data yang dihasilkan belum sepenuhnya mampu mewakili keanekaragaman gejala di kondisi nyata.

Sebagai pembanding, model SVM dilatih menggunakan fitur visual berdimensi 1280 yang diekstraksi dari MobileNetV2 (tanpa lapisan klasifikasi akhir). Model ini menggunakan *kernel linier* dan dilatih dengan data yang sama seperti pada model CNN. Hasilnya, SVM menunjukkan performa yang lebih baik, dengan akurasi mencapai 80.00%, presisi 81.25%, recall 80.00%, dan F1-score 79.80%. Hal ini menunjukkan bahwa kombinasi fitur pralatih dari MobileNetV2

dengan klasifikasi konvensional cukup efektif, terutama ketika jumlah data pelatihan terbatas.

Dari segi waktu komputasi, model SVM jauh lebih efisien. Waktu pelatihannya hanya sekitar 0,013 detik, dan pengujian sekitar 0,002 detik. Sebaliknya, CNN membutuhkan waktu pelatihan hingga 114 detik dan waktu pengujian sekitar 3,8 detik. Perbedaan ini menunjukkan bahwa SVM lebih ringan dan cepat, sehingga lebih cocok diterapkan pada sistem klasifikasi berbasis citra yang ingin dijalankan di perangkat dengan spesifikasi rendah.

Secara keseluruhan, pendekatan menggunakan MobileNetV2 sebagai *feature extractor* yang digabungkan dengan SVM memberikan hasil yang lebih baik dari sisi akurasi maupun efisiensi. Sementara CNN tetap memiliki potensi tinggi, penggunaannya lebih tepat untuk kasus dengan *dataset* yang besar dan beragam. Ringkasan perbandingan kinerja kedua algoritma tersebut dapat dilihat pada Tabel 4.1.

Tabel 4.1 Tabel Perbandingan Hasil Metrik Evaluasi pada CNN dan SVM

No	Metrik Evaluasi	CNN	SVM
1	Akurasi	67.50%	80.00%
2	Presisi	70.45%	81.25%
4	Recall	67.50%	80.00%
5	F1-score	68.21%	79.80%
6	Waktu Pelatihan	114 detik	0.013 detik