BAB IV
PEMBAHASAN

4.1 Gambaran Umum Obyek Penelitian

Penelitian ini bertujuan untuk memberikan cara alternatif dalam mengetahui warna bunga mawar hanya dengan melihat daunnya. Obyek penelitian ditujukan untuk masyarakat yang terlibat langsung dalam budidaya dan jual beli bunga mawar, terutama petani dan calon pembeli di daerah Sidomulyo, Kota Batu, yang dikenal sebagai sentra penghasil bunga. Sistem yang dibuat diharapkan bisa membantu mereka mengenali warna bunga yang akan tumbuh sebelum bunganya benar-benar mekar.

Struktur organisasi yang terlibat dalam penelitian ini mencakup peneliti sebagai pelaksana utama seluruh proses, mulai dari pengumpulan data, pelatihan model, implementasi sistem, hingga evaluasi akhir. Dosen pembimbing dari UBHINUS berperan dalam pengawasan proses metodologi. Tidak terdapat kolaborator eksternal dalam pelaksanaan teknis sistem ini.

Dalam pelaksanaannya, pengambilan gambar daun dilakukan secara mandiri oleh peneliti di teras rumah, sebagai lokasi dengan pencahayaan alami yang stabil. Waktu pengambilan gambar dilakukan secara fleksibel sebanyak 1–2 kali per minggu, dengan masing-masing sesi menghasilkan 15–20 gambar per kelas warna (merah, putih, kuning). Seluruh gambar diambil menggunakan kamera smartphone dengan resolusi 2 megapiksel dan fitur makro, untuk memastikan detail tekstur daun dapat tertangkap dengan baik.

Beberapa aturan internal ditetapkan selama proses pengumpulan data, di antaranya adalah penggunaan satu jenis perangkat kamera, serta penerapan ambang batas minimal jumlah piksel hijau pada gambar (threshold ≥ 5000 piksel) agar dapat dianggap valid sebagai citra daun. Hal ini dilakukan guna menjamin kualitas dan konsistensi data yang akan digunakan dalam proses pelatihan model.

4.2 Implementasi

4.2.1. Spesifikasi Produk

Produk yang dihasilkan dari penelitian ini adalah sebuah sistem prediksi warna bunga mawar berbasis website. Sistem ini dirancang untuk memungkinkan pengguna mengunggah gambar daun mawar, yang kemudian akan diproses oleh model prediksi untuk mengidentifikasi warna bunga yang diperkirakan, yaitu merah, kuning, atau putih. Pengembangan sistem ini dilakukan menggunakan framework Django sebagai basis pengelolaan sisi backend dan antarmuka pengguna, sehingga menghasilkan aplikasi web yang interaktif dan mudah digunakan.

Spesifikasi perangkat keras dan lunak:

- Hardware:

  CPU: AMD Ryzen 5 4600H
  RAM: 16 GB
  Kamera: 2 MP (smartphone, mode makro)

- Software:

  Bahasa Pemrograman: Python
  Framework Web: Django
  Editor: Visual Studio Code
  OS: Windows 11
  Library: TensorFlow, Keras, NumPy, OpenCV, Matplotlib, PIL scikit-learn

4.2.2. Implementasi Program

a. Pra-pemrosesan

Langkah-langkah preprocessing meliputi:

- Resize gambar ke ukuran standar input CNN

```
target_size = (256, 256)
counter = 1
for filename in os.listdir(input_dir):
    if filename.lower().endswith(('.jpg', '.png', '.jpeg')):
        input_path = os.path.join(input_dir, filename)
        img = cv2.imread(input_path)
        if img is None:
            continue
        # ======== Padding agar gambar berbentuk 1:1 ========
        h_result, w_result = result.shape[:2]
        size = max(h_result, w_result)
        top = (size - h_result) // 2
        bottom = size - h_result - top
        left = (size - w_result) // 2
        right = size - w_result - left
        padded_img = cv2.copyMakeBorder(result, top, bottom, left, right,
                                        cv2.BORDER_CONSTANT, value=[255, 255, 255])
        # ======== Resize ke 256x256 ========
        resized_img = cv2.resize(padded_img, target_size,
interpolation=cv2.INTER_AREA)
        output_filename = f"putih_{counter}.jpg"
        output_path = os.path.join(output_dir, output_filename)
        cv2.imwrite(output_path, resized_img)
        counter += 1
```

Segmen Program 4. 1 Resize gambar

- Normalisasi nilai piksel

```
train_datagen = ImageDataGenerator(
    rescale=1./255, rotation_range=30,
    width_shift_range=0.2, height_shift_range=0.2,
    shear_range=0.2, zoom_range=0.2,
    horizontal_flip=True, brightness_range=[0.8, 1.2],
    validation_split=0.2
)
test_datagen = ImageDataGenerator(rescale=1./255)
```

Segmen Program 4. 2 Normalisasi

- Augmentasi data (rotasi, flipping, zoom, brightness)

```
img = random.choice(selected_images)
    # ===== Augmentasi Gambar =====
    aug_img = img.copy()
    choice = random.randint(0, 4)
    if choice == 0:
        aug_img = cv2.flip(aug_img, 1)  # Flip horizontal
    elif choice == 1:
        aug_img = cv2.rotate(aug_img, cv2.ROTATE_90_CLOCKWISE)
    elif choice == 2:
        aug_img = cv2.rotate(aug_img,
cv2.ROTATE_90_COUNTERCLOCKWISE)  # Rotasi -90 derajat
    elif choice == 3:
        aug_img = cv2.convertScaleAbs(aug_img, alpha=1.2, beta=10)
    elif choice == 4:
        aug_img = cv2.GaussianBlur(aug_img, (5, 5), 0)  # Blur
    new_filename = f"aug_{len(selected_images)}.jpg"
    cv2.imwrite(os.path.join(selected_dir, new_filename), aug_img)
    selected_images.append(aug_img)
```

Segmen Program 4. 3 Augmentasi

- Pembersihan background untuk menjadikan latar belakang netral

```
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
lower_green = np.array([25, 40, 40])
upper_green = np.array([85, 255, 255])
mask = cv2.inRange(hsv, lower_green, upper_green)
mask_inv = cv2.bitwise_not(mask)
img_fg = cv2.bitwise_and(img, img, mask=mask)
white_bg = np.full(img.shape, 255, dtype=np.uint8)
result = np.where(mask[:, :, np.newaxis] == 0, white_bg, img_fg)
```

Segmen Program 4. 4 Menghapus background dan mengubah mendaji warna netral menggunakan menyeleksi yang warna hijau

```
import rembg                import cv2
import numpy as np          import os
import io                   from PIL import Image
def remove_background(image_path):
    with open(image_path, 'rb') as f:
        input_image = f.read()
    output_image = rembg.remove(input_image)
    image =
Image.open(io.BytesIO(output_image)).convert("RGBA")
    white_background = Image.new("RGBA", image.size, (255,
255, 255, 255))
    image = Image.alpha_composite(white_background,
image).convert("RGB")
    image.save(image_path, format="PNG")
    print(f"Background dihapus dan diubah menjadi putih:
{image_path}")
```

Segmen Program 4. 5 Mengahapus background menggunakan library remove background

- Threshold piksel hijau: hanya gambar dengan ≥5.000 piksel hijau yang diproses

```
threshold_leaf = 5000
def has_leaf(img):
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    lower_green = np.array([25, 40, 40])
    upper_green = np.array([85, 255, 255])
    mask = cv2.inRange(hsv, lower_green, upper_green)
    green_pixels = np.sum(mask > 0)
    return green_pixels > threshold_leaf
selected_images = []
for filename in tqdm(os.listdir(input_dir)):
    if filename.lower().endswith(('.jpg', '.png', '.jpeg')):
        img_path = os.path.join(input_dir, filename)
        img = cv2.imread(img_path)
        if img is None:
            continue
        if has_leaf(img):
            selected_images.append(img)
            cv2.imwrite(os.path.join(selected_dir, filename), img)
```

Segmen Program 4. 6 Menghapus gambar yang tidak sesuai

b. Model CNN dan Training

Model CNN dibuat dengan *Keras,* dilatih selama 50 *epoch* dan menggunakan *ModelCheckpoint* untuk menyimpan model dengan akurasi validasi terbaik. *Cross-validation* dilakukan dengan 5 *fold*.

Tabel 4. 1 Hasil Training model Cross-Validation

Fold	Akurasi
1	0.5500
2	0.6000
3	0.5900
4	0.6267
5	0.6200

```
import os
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
import seaborn as sns
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense,
Dropout
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.models import Model
from tensorflow.keras.layers import GlobalAveragePooling2D, Input
from tensorflow.keras.regularizers import l2
from sklearn.metrics import confusion_matrix, classification_report
```

Segmen Program 4. 7 Library yang digunakan dalam pembuatan model

```
test_fold = 5
base_dir = "D:/Kuliah/Semester_6/Metpen/Dataset_KFold"
all_folds = [f"Fold_{i}" for i in range(1, 6)]
train_folds = [f for f in all_folds if f != f"Fold_{test_fold}"]
train_dir = os.path.join(base_dir, "temp_train")
os.makedirs(train_dir, exist_ok=True)
import shutil
for fold in train_folds:
    for label in ["kuning", "merah", "putih"]:
        src = os.path.join(base_dir, fold, label)
        dst = os.path.join(train_dir, label)
        os.makedirs(dst, exist_ok=True)
        for file in os.listdir(src):
```

Segmen Program 4. 8 Segmen program untuk menentuan fold mana yang digunakan sebagai training dan testing

```
# Direktori simpan model
model_save_dir = "D:/Kuliah/Semester_6/Metpen/Models"
os.makedirs(model_save_dir, exist_ok=True)
```

Segmen Program 4. 9 Direktori untuk menyimpan hasil atau model

```
# Hyperparameter
img_size = (256, 256)
batch_size = 32
num_classes = 3
max_epochs = 50
```

Segmen Program 4. 10 Parameter yang akan digunakan untuk pembuatan model

```
# Fungsi untuk membuat model CNN dengan pretrained MobileNetV2
def create_cnn():
    base_model = MobileNetV2(include_top=False,
input_tensor=Input(shape=(256, 256, 3)), weights='imagenet')
    base_model.trainable = False
    x = base_model.output
    x = GlobalAveragePooling2D()(x)
    x = Dense(64, activation='relu', kernel_regularizer=l2(0.001))(x)
    x = Dropout(0.4)(x)
    predictions = Dense(num_classes, activation='softmax')(x)
    model = Model(inputs=base_model.input, outputs=predictions)
    model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),
loss='categorical_crossentropy', metrics=['accuracy'])
    return model
```

Segmen Program 4. 11 Fungsi untuk model CNN

```
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=30,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    brightness_range=[0.8, 1.2],
    validation_split=0.2
)
test_datagen = ImageDataGenerator(rescale=1./255)
train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
    subset='training'
)
val_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation'
)
```

Segmen Program 4. 12 Augmentasi dan noemalisasi dalam pembuatan model

```
# Callback Model Checkpoint
checkpoint_path = os.path.join(model_save_dir,
f"best_model_fold_{test_fold}.h5")
model_checkpoint = ModelCheckpoint(checkpoint_path,
monitor='val_accuracy', save_best_only=True, mode='max', verbose=1)
```

Segmen Program 4. 13 Code untuk menyimpan hasil terbaik dalam training 50 Epoch

```
# Buat model CNN
model = create_cnn()
# Train model
print(f"\n Training, Validasi menggunakan fold selain Fold {test_fold}...\n")
history = model.fit(train_generator, validation_data=val_generator,
epochs=max_epochs,
                    callbacks=[model_checkpoint], verbose=1)
```

Segmen Program 4. 14 Code membuat model CNN menggunakan fungsi yang ada

```
final_epoch = len(history.history['loss'])
best_epoch = np.argmax(history.history['val_accuracy']) + 1
print(f"\n Training selesai di epoch: {final_epoch}")
print(f" Epoch terbaik yang dipilih: {best_epoch}")
# Load model terbaik
model.load_weights(checkpoint_path)
print(f" Model terbaik digunakan dari: {checkpoint_path}")
# Simpan model akhir
model_keras_path = os.path.join(model_save_dir,
f"model_best_fold_{test_fold}.keras")
model.save(model_keras_path)
model_h5_path = os.path.join(model_save_dir,
f"model_best_fold_{test_fold}.h5")
```

Segmen Program 4. 15 Code untuk mencetak hasil training

```
test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False
)
test_loss, test_acc = model.evaluate(test_generator)
print(f"\n Akurasi Testing (Fold {test_fold}): {test_acc:.4f}")
y_pred = np.argmax(model.predict(test_generator), axis=1)
y_true = test_generator.classes
conf_matrix = confusion_matrix(y_true, y_pred)
plt.figure(figsize=(6,6))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues",
xticklabels=test_generator.class_indices.keys(),
yticklabels=test_generator.class_indices.keys())
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title(f"Confusion Matrix Fold {test_fold}")
plt.show()
class_report = classification_report(y_true, y_pred,
target_names=test_generator.class_indices.keys())
print(f"\n Classification Report:\n{class_report}")
```

Segmen Program 4. 16 Code untuk mengevaluasi pada Fold yang digunakan untuk testing

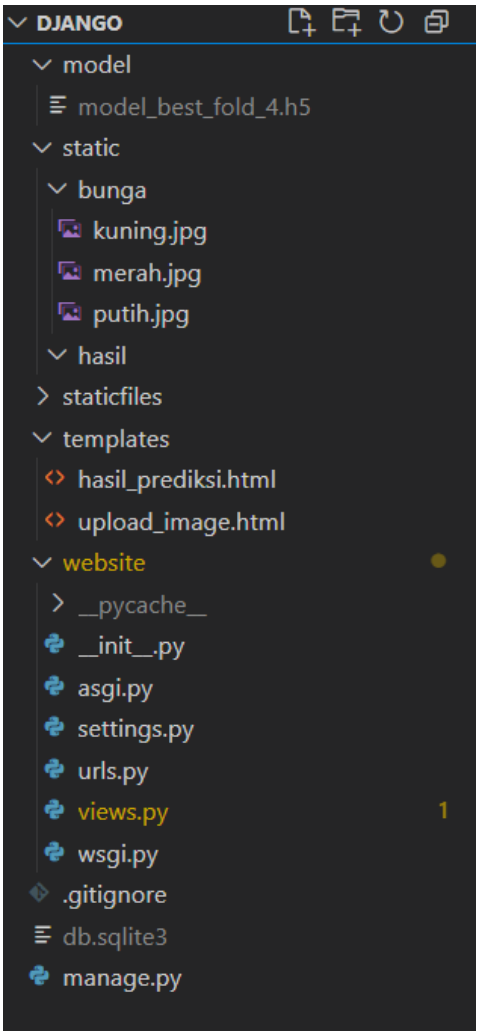
```
# ======== BERSIHKAN DATA SEMENTARA TRAINING ========
shutil.rmtree(train_dir)
```

Segmen Program 4. 17 Code untuk membersihkan data training

c. Implementasi Website Django

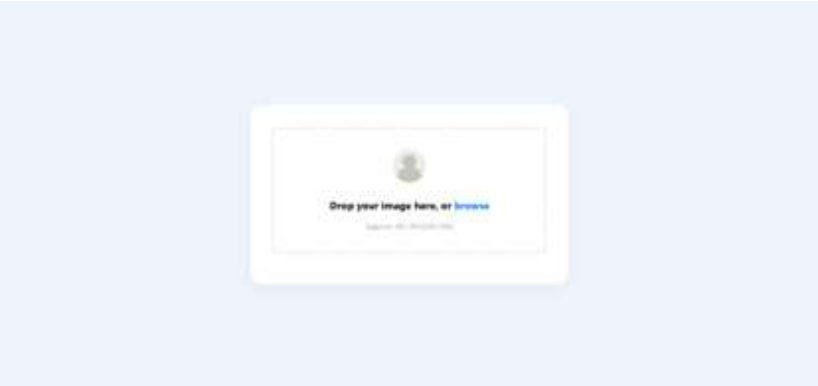
Website dibuat menggunakan Django dan menyediakan fitur unggah gambar daun, kemudian menampilkan hasil prediksi.

Struktur direktori Django:

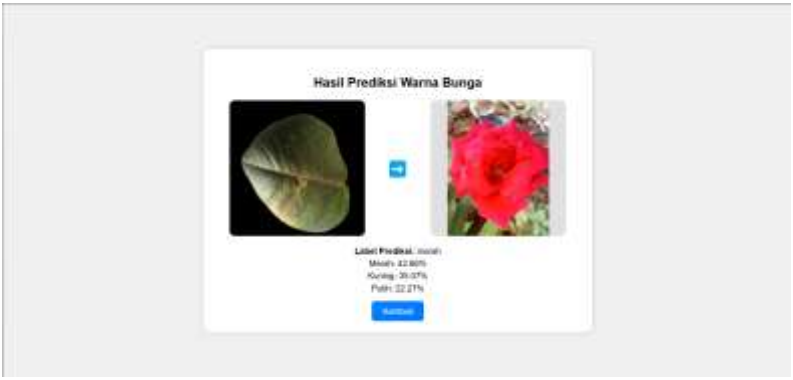


Gambar 4. 1 Struktur direktori website dengan framework DJANGO

Berikut tampilan hasil implementasi Website menggunakan framework Django



Gambar 4. 2 Tampilan upload image



Gambar 4. 3 Tampilan Hasil prediksi

```
# project_website/views.py
import tensorflow as tf
from django.shortcuts import render
from tensorflow.keras.models import load_model
import numpy as np
from PIL import Image
import os
import io
from rembg import remove
from django.conf import settings
from django.utils.crypto import get_random_string
# Path model
model_path = os.path.join(os.path.dirname(os.path.dirname(__file__)),
'model', 'model_best_fold_4.h5')
model = load_model(model_path)
class_names = ['kuning', 'merah', 'putih']
def predict(request):
    if request.method == "POST" and request.FILES.get("image"):
        image = request.FILES["image"]
        img = Image.open(image).convert("RGBA")

        # Convert ke bytes untuk rembg
        buffered = io.BytesIO()
        img.save(buffered, format="PNG")
        img_bytes = buffered.getvalue()

        # Remove background
        output_bytes = remove(img_bytes)
        img_no_bg =
Image.open(io.BytesIO(output_bytes)).convert("RGB")
```

Segmen Program 4. 18 Code views.py_1

```
        # Resize ke 256x256 terlebih dahulu
        img_resized = img_no_bg.resize((256, 256))

        # Jika gambar tidak 1:1 (persegi), tambahkan latar belakang putih
        width, height = img_resized.size
        if width != height:
            # Tentukan ukuran sisi terpanjang
            new_size = max(width, height)
            # Buat gambar baru dengan background putih dan ukuran baru
            new_img = Image.new("RGB", (new_size, new_size), (255, 255,
255))
            # Menempatkan gambar asli yang sudah diresize di tengah
            new_img.paste(img_resized, ((new_size - width) // 2, (new_size
- height) // 2))
            img_resized = new_img

        # Konversi ke array untuk prediksi
        img_array = np.array(img_resized) / 255.0
        img_array = np.expand_dims(img_array, axis=0)

        # Prediksi
        predictions = model.predict(img_array)[0]  # ambil prediksi baris
pertama
        predicted_class = np.argmax(predictions)
        predicted_label = class_names[predicted_class]

        prediction_percentages = {
            class_names[i]: round(float(prob * 100), 2)
            for i, prob in enumerate(predictions)
        }
```

Segmen Program 4. 19 Code views.py_2

```
    # Membuat nama file gambar acak untuk gambar yang diupload
    random_filename = f"{get_random_string(8)}.png"
    hasil_dir = os.path.join(settings.BASE_DIR, 'static', 'hasil')
    os.makedirs(hasil_dir, exist_ok=True)

    full_upload_path = os.path.join(hasil_dir, random_filename)
    img_resized.save(full_upload_path)

    # URL untuk gambar yang diupload
    uploaded_image_url = f"/static/hasil/{random_filename}"

    # Path ke gambar hasil prediksi (contoh referensi)
    predicted_image_url = f"/static/bunga/{predicted_label}.jpg"

    return render(request, "hasil_prediksi.html", {
        "uploaded_image": uploaded_image_url,
        "predicted_image": predicted_image_url,
        "prediction_percentages": prediction_percentages,
        "label": predicted_label
    })

return render(request, "upload_image.html")
```

Segmen Program 4. 20 Code views.py_3

```
<!DOCTYPE html>
<html lang="id">
<head>
    <meta charset="UTF-8">
    <title>Hasil Prediksi Warna Bunga</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            display: flex;
            justify-content: center;
            align-items: center;
            min-height: 100vh;
            margin: 0;
            background: #f0f0f0;
        }

        .container {
            background: white;
            padding: 30px;
            border-radius: 12px;
            box-shadow: 0 0 10px rgba(0,0,0,0.15);
            text-align: center;
            max-width: 700px;
        }

        .image-row {
            display: flex;
            justify-content: center;
            align-items: center;
            margin-bottom: 20px;
        }
```

Segmen Program 4. 21 Code hasil_prediksi.html_1

```
.image-box {
    width: 256px;
    height: 256px;
    background-color: #e0e0e0;
    margin: 0 20px;
    display: flex;
    justify-content: center;
    align-items: center;
    border-radius: 10px;
    overflow: hidden;
}

.image-box img {
    max-width: 100%;
    max-height: 100%;
}

.arrow {
    font-size: 32px;
    margin: 0 20px;
}

.result {
    margin-top: 20px;
}
.result p {
    margin: 5px 0;
}
.button-container {
    margin-top: 25px;
}
```

Segmen Program 4. 22 Code hasil_prediksi.html_2

```
        .button-container a {
            background-color: #007bff;
            color: white;
            padding: 10px 20px;
            border-radius: 6px;
            text-decoration: none;
        }
        .button-container a:hover {
            background-color: #0056b3;
        }
    </style>
</head>
<body>
    <div class="container">
        <h2>Hasil Prediksi Warna Bunga</h2>

        <div class="image-row">
            <div class="image-box">
                <img src="{{ uploaded_image }}" alt="Gambar Daun">
            </div>
            <div class="arrow">-></div>
            <div class="image-box">
                <img src="{{ predicted_image }}" alt="Hasil Prediksi">
            </div>
        </div>
        <div class="result">
            <p><strong>Label Prediksi:</strong> {{ label }}</p>
            <p>Merah: {{ prediction_percentages.merah }}%</p>
            <p>Kuning: {{ prediction_percentages.kuning }}%</p>
            <p>Putih: {{ prediction_percentages.putih }}%</p>
        </div>
```

Segmen Program 4. 23 Code hasil_prediksi.html_3

```
from django.contrib import admin
from django.urls import path
from . import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', views.predict, name='predict'),
]
```

Segmen Program 4. 24 Code urls.py

```
</html>
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0"
/>
    <title>Image Upload</title>
    <style>
        body {
            background: #eef5fc;
            font-family: 'Segoe UI', sans-serif;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
            margin: 0;
        }
```

Segmen Program 4. 25 Code upload_ image.html_1

```
.upload-container {
    background: white;
    border-radius: 20px;
    box-shadow: 0 10px 30px rgba(0, 0, 0, 0.05);
    padding: 40px;
    width: 500px;
    text-align: center;
}

.upload-box {
    border: 2px dashed #ccc;
    border-radius: 12px;
    padding: 40px;
    cursor: pointer;
    transition: 0.3s;
}

.upload-box.dragover {
    background-color: #f0f8ff;
    border-color: #3e98ff;
}

.upload-box img {
    width: 60px;
    margin-bottom: 10px;
}

.upload-box span {
    color: #007BFF;
    cursor: pointer;
}
```

Segmen Program 4. 26 Code upload_ image.html_2

```
.formats {
    font-size: 12px;
    color: #999;
    margin-top: 5px;
}

.progress-section {
    display: none;
    margin-top: 20px;
}
.upload-status {
    font-size: 14px;
    color: #777;
    margin-bottom: 10px;
}
.progress-bar {
    background: #f0f0f0;
    border-radius: 10px;
    overflow: hidden;
    height: 8px;
    position: relative;
}

.progress {
    width: 0%;
    background-color: #3e98ff;
    height: 100%;
    transition: width 0.3s ease;
}
```

Segmen Program 4. 27 Code upload_ image.html_3

```
        .submit-btn {
            margin-top: 20px;
            padding: 10px 20px;
            background-color: #007bff;
            color: white;
            border: none;
            border-radius: 8px;
            font-size: 14px;
            cursor: pointer;
            display: none;
        }
        .submit-btn:hover {
            background-color: #0056b3;
        }
        input[type="file"] {
            display: none;
        }
    </style>
</head>
<body>
    <div class="upload-container">
        <form method="POST" enctype="multipart/form-data">
            {% csrf_token %}
            <div class="upload-box" id="drop-area">
                <input type="file" id="fileInput" name="image" accept="image/*"
required/>
                <img src="https://cdn-icons-
png.flaticon.com/512/847/847969.png" alt="upload">
                <h3>Drop your image here, or <span>browse</span></h3>
                <div class="formats">Supports: JPG, JPEG2000, PNG</div>
            </div>\\\
```

Segmen Program 4. 28 Code upload_ image.html_4

```
        <div class="progress-section" id="progressSection">
            <div class="upload-status" id="uploadStatus">Uploading...
0%</div>
            <div class="progress-bar">
                <div class="progress" id="progressBar"></div>
            </div>
            <button type="submit" class="submit-btn"
id="submitBtn">Submit</button>
        </div>
    </form>
  </div>

  <script>
    const dropArea = document.getElementById("drop-area");
    const fileInput = document.getElementById("fileInput");
    const progressSection =
document.getElementById("progressSection");
    const progressBar = document.getElementById("progressBar");
    const uploadStatus = document.getElementById("uploadStatus");
    const submitBtn = document.getElementById("submitBtn");
    // Click to upload
    dropArea.addEventListener("click", () => fileInput.click());
    // File selected via browse
    fileInput.addEventListener("change", handleFiles);
    // Drag events
    dropArea.addEventListener("dragover", (e) => {
      e.preventDefault();
      dropArea.classList.add("dragover");
    });
```

Segmen Program 4. 29 Code upload_ image.html_5

```
    dropArea.addEventListener("dragleave", () => {
        dropArea.classList.remove("dragover");
    });
    dropArea.addEventListener("drop", (e) => {
        e.preventDefault();
        dropArea.classList.remove("dragover");
        const files = e.dataTransfer.files;
        handleFiles({
            target: {
                files
            }
        });
    });
    function handleFiles(e) {
        const file = e.target.files[0];
        if (!file) return;
        progressSection.style.display = "block";
        let progress = 0;
        const interval = setInterval(() => {
            progress += 5;
            progressBar.style.width = progress + "%";
            uploadStatus.textContent = progress < 100 ?
                `Uploading... ${progress}%` :
                "Completed!";
            if (progress >= 100) {
                clearInterval(interval);
                submitBtn.style.display = "inline-block";
            }
        }, 200); // Simulate 5% every 200ms (100% = 4s)
    }
</script></body></html>
```

Segmen Program 4. 30 Code upload_ image.html_6

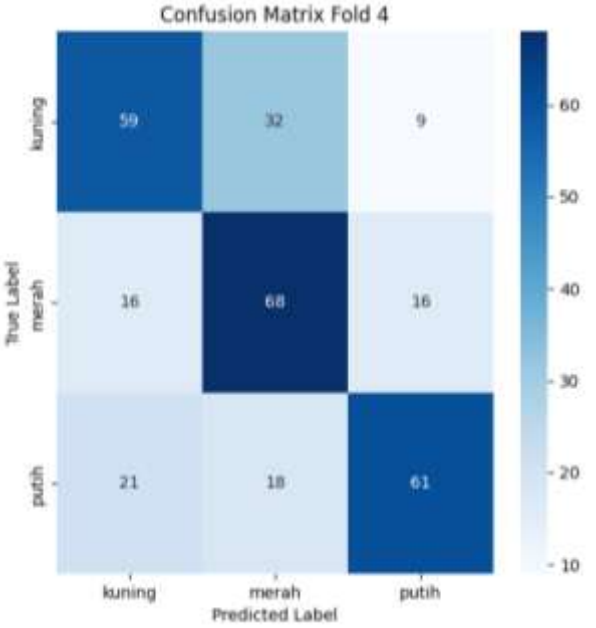
4.3 Uji Coba

Uji coba dilakukan untuk mengevaluasi kinerja model Convolutional Neural Network (CNN) serta menguji sistem prediksi warna bunga mawar berbasis website dalam lingkungan nyata. Proses uji coba ini dibagi menjadi dua bagian, yaitu uji performa model dan uji sistem aplikasi.

4.3.1 Uji Performa Model (Cross-Validation dan Evaluasi)

Model dilatih dan diuji menggunakan metode 5-Fold Cross Validation, di mana pada setiap fold, data dibagi menjadi 80% untuk pelatihan dan 20% untuk pengujian. Hasil evaluasi menunjukkan bahwa model dengan performa terbaik diperoleh pada Fold 4 dengan tingkat akurasi sebesar 62,67%.

Untuk mengevaluasi performa model secara lebih menyeluruh, digunakan metrik evaluasi klasifikasi, yaitu akurasi, presisi, recall, dan F1-score, serta confusion matrix sebagai alat bantu visualisasi performa prediksi pada masing-masing kelas warna bunga.



Gambar 4. 4 Confusion Matrix Fold 4

Kelas Kuning

- TP = 59
- FP = 16 (dari merah) + 21 (dari putih) = 37
- FN = 32 + 9 = 41
- Precision = 59 / (59 + 37) = 59 / 96 ≈ 0.6146
- Recall = 59 / (59 + 41) = 59 / 100 = 0.59
- F1 = 2 × (0.6146 × 0.59) / (0.6146 + 0.59) ≈ 0.6021

Kelas Merah

- TP = 68
- FP = 32 (kuning) + 18 (putih) = 50
- FN = 16 + 16 = 32
- Precision = 68 / (68 + 50) = 68 / 118 ≈ 0.5763
- Recall = 68 / (68 + 32) = 68 / 100 = 0.68
- F1 = 2 × (0.5763 × 0.68) / (0.5763 + 0.68) ≈ 0.6236

Kelas Putih

- TP = 61
- FP = 9 (kuning) + 16 (merah) = 25
- FN = 21 + 18 = 39
- Precision = 61 / (61 + 25) = 61 / 86 ≈ 0.7093
- Recall = 61 / (61 + 39) = 61 / 100 = 0.61
- F1 = 2 × (0.7093 × 0.61) / (0.7093 + 0.61) ≈ 0.6555

Akurasi Keseluruhan

Jumlah data = 300

Total benar (TP seluruh kelas) = 59 + 68 + 61 = 188

Hasil evaluasi pada Fold ke-4 menunjukkan bahwa model CNN yang dikembangkan memiliki kemampuan dalam membedakan warna bunga mawar berdasarkan citra daun, dengan performa terbaik diperoleh pada kelas putih. Namun, prediksi pada kelas merah dan kuning masih menunjukkan tingkat kesalahan yang cukup tinggi, baik berupa false positive maupun false negative.

Temuan ini mengindikasikan bahwa terdapat kemiripan atau tumpang tindih fitur visual antar kelas yang menyulitkan model dalam melakukan klasifikasi secara akurat. Oleh karena itu, diperlukan upaya

perbaikan, seperti optimalisasi proses preprocessing (misalnya peningkatan kualitas gambar, segmentasi yang lebih presisi), serta penambahan teknik ekstraksi fitur atau penggunaan augmentasi data yang lebih beragam, guna meningkatkan performa model secara keseluruhan.

4.3.2 Uji Sistem Prediksi Berbasis Website

Pengujian sistem website dilakukan dengan mengunggah gambar daun secara langsung melalui halaman antarmuka web. Untuk menguji kestabilan dan keakuratan sistem, dilakukan pengujian menggunakan 30 gambar baru yang tidak termasuk dalam data pelatihan maupun validasi, terdiri dari:

- 10 gambar daun bunga mawar merah
- 10 gambar daun bunga mawar kuning
- 10 gambar daun bunga mawar putih

Hasil prediksi ditampilkan langsung oleh sistem dalam bentuk label warna bunga yang diprediksi.

Tabel 4. 2 Hasil pengujian website dengan 10 warna Kuning

No	Gambar Daun	Kelas Sebenarnya	Hasil Prediksi
1		Kuning	Kuning
2		Kuning	Kuning

3		Kuning	Kuning
4		Kuning	Kuning
5		Kuning	Kuning
6		Kuning	Putih
7		Kuning	Kuning

8		Kuning	Kuning
9		Kuning	Kuning
10		Kuning	Kuning

Tabel 4. 3 Hasil pengujian website dengan 10 warna Merah

No	Gambar Daun	Kelas Sebenarnya	Hasil Prediksi
1		Merah	Merah
2		Merah	Merah

3		Merah	Merah
4		Merah	Merah
5		Merah	Kuning
6		Merah	Kuning
7		Merah	Kuning

8		Merah	Putih
9		Merah	Putih
10		Merah	Putih

Tabel 4. 4 Hasil pengujian website dengan 10 warna Putih

No	Gambar Daun	Kelas Sebenarnya	Hasil Prediksi
1		Putih	Kuning
2		Putih	Kuning

3		Putih	Putih
4		Putih	Putih
5		Putih	Putih
6		Putih	Kuning
7		Putih	Putih

8		Putih	Kuning
9		Putih	Putih
10		Putih	Putih

Tabel 4. 5 Hasil Uji Coba 30 gambar

True Label	Predicted Kuning	Predicted Merah	Predicted Putih
Kuning	9	0	1
Merah	3	4	3
Putih	4	0	6

Berdasarkan hasil tersebut, model berhasil memprediksi 19 dari 30 gambar secara tepat, dengan tingkat akurasi sebesar 63,33%.

Selain itu, dilakukan perhitungan metrik evaluasi lain seperti Precision, Recall, dan F1-Score untuk masing-masing kelas. Metrik-metrik ini memberikan gambaran yang lebih detail mengenai performa klasifikasi model, terutama dalam situasi ketidakseimbangan kelas :

- Precision = TP / (TP + FP)
- Recall = TP / (TP + FN)
- F1-Score = 2 × (Precision × Recall) / (Precision + Recall)

a. Evaluasi Kelas Kuning
   - True Positive (TP) = 9
   - False Positive (FP) = 7
   - False Negative (FN) = 1
   - Precision = 56,25%
   - Recall = 90,00%
   - F1-Score = 69,23%

b. Evaluasi Kelas Merah
   - TP = 4
   - FP = 0
   - FN = 6
   - Precision = 100,00%
   - Recall = 40,00%
   - F1-Score = 57,14%

c. Evaluasi Kelas Putih
   - TP = 6
   - FP = 4
   - FN = 4
   - Precision = 60,00%
   - Recall = 60,00%
   - F1-Score = 60,00%

Secara umum, sistem menunjukkan kinerja yang cukup baik dalam mengklasifikasikan gambar daun ke dalam warna bunga mawar yang sesuai. Kinerja tertinggi dicapai pada kelas kuning dengan Recall sebesar 90%, sedangkan kelas merah memiliki Precision sempurna (100%) namun dengan Recall rendah. Hal ini menunjukkan bahwa sistem lebih baik dalam mengidentifikasi daun berwarna kuning, namun masih perlu perbaikan pada klasifikasi kelas merah agar tidak banyak terjadi kesalahan pengenalan.