LAMPIRAN

LIBRARY YANG DIGUNAKAN

```
import os
import cv2
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report
import joblib
import tensorflow as tf
from tensorflow import keras
```

LOAD & PREPROCESS DATASET

```
_tomat"
labels = []
features = []

def extract_hsv_feature(image_path):
    img = cv2.imread(image_path)
    img = cv2.resize(img, (100, 100))
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    # Ambil rata-rata nilai H, S, dan V
    h_mean = np.mean(hsv[:,:,0])
    s_mean = np.mean(hsv[:,:,1])
    v_mean = np.mean(hsv[:,:,2])
    return [h_mean, s_mean, v_mean]

# Struktur folder: dataset_tomat/Matang/*.jpg, dataset_tomat/Sedang/*.jpg, dataset_tomat/Mentah/*.jpg
for class_name in os.listdir(DATASET_DIR):
    class_path = os.path.join(DATASET_DIR, class_name)
    if not os.path.isdir(class_path):
        continue
    for img_file in os.listdir(class_path):
        if img_file.endswith(('.jpg','.png','.jpeg')):
            fpath = os.path.join(class_path, img_file)

            features.append(extract_hsv_feature(fpath))
            labels.append(class_name)

features = np.array(features)
labels = np.array(labels)
```

SPLIT DATASET (80:20)

```
X_train, X_test, y_train, y_test = train_test_split(
    features, labels, test_size=0.2, random_state=42, stratify=labels
)
```

TRAIN MODEL KNN (K=3)

```
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
```

EVALUASI MODEL

```
y_pred = knn.predict(X_test)
acc = accuracy_score(y_test, y_pred)
print("Akurasi pada data uji:", acc)
print(classification_report(y_test, y_pred))
```

SIMPAN MODEL JOBLIB

```
joblib.dump(knn, "knn_tomat.joblib")
print("Model KNN disimpan ke knn_tomat.joblib")
```

KONVERSI KE TENSORFLOW LITE UNTUK ANDROID

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y_train_enc = le.fit_transform(y_train)
y_test_enc = le.transform(y_test)

# Simpan label encoder
joblib.dump(le, "label_encoder.joblib")

# Bangun model Dense sederhana untuk meniru KNN
tf_model = keras.Sequential([
    keras.layers.Input(shape=(3,)),  # HSV features
    keras.layers.Dense(16, activation='relu'),
    keras.layers.Dense(8, activation='relu'),

    keras.layers.Dense(len(le.classes_), activation='softmax')
])
tf_model.compile(optimizer='adam',
                 loss='sparse_categorical_crossentropy',
                 metrics=['accuracy'])

tf_model.fit(X_train, y_train_enc, epochs=100, batch_size=8, verbose=0)
loss, tflite_acc = tf_model.evaluate(X_test, y_test_enc, verbose=0)
print("Akurasi model TFLite clone:", tflite_acc)

# Simpan model .tflite
converter = tf.lite.TFLiteConverter.from_keras_model(tf_model)
tflite_model = converter.convert()
with open("knn_tomat.tflite", "wb") as f:
    f.write(tflite_model)
print("Model TFLite disimpan ke knn_tomat.tflite")
```

UJI MODEL JOBLIB

```
def predict_with_joblib(image_path):
    model = joblib.load("knn_tomat.joblib")
    feat = np.array(extract_hsv_feature(image_path)).reshape(1,-1)
    pred = model.predict(feat)[0]
    return pred
```

UJI MODEL TFLITE

```
predict_with_tflite(image_path):
    interpreter = tf.lite.Interpreter(model_path="knn_tomat.tflite")
    interpreter.allocate_tensors()
    input_details = interpreter.get_input_details()
    output_details = interpreter.get_output_details()
    feat = np.array(extract_hsv_feature(image_path), dtype=np.float32).reshape(1,3)

    interpreter.set_tensor(input_details[0]['index'], feat)
    interpreter.invoke()
    output = interpreter.get_tensor(output_details[0]['index'])
    pred_idx = np.argmax(output)
    le = joblib.load("label_encoder.joblib")
    return le.inverse_transform([pred_idx])[0]
```

CONTOH PENGUJIAN MANUAL

```
test_image = "contoh_tomat.jpg" # ganti dengan path gambar uji
print("Prediksi JOBLIB :", predict_with_joblib(test_image))
print("Prediksi TFLITE :", predict_with_tflite(test_image))
```

TAMPILAN BERANDA GUI

```
<!DOCTYPE html>
<html lang="id">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>Aplikasi Deteksi Kematangan Buah Tomat</title>
 <link href="https://fonts.googleapis.com/css2?family=Poppins:wght@500;700&display=swap" rel="stylesheet">
 <style>
  body {
   font-family: 'Poppins', sans-serif;
   background: linear-gradient(135deg, #1f1f1f, #2c2c2c);
   color: #ffffff;
   margin: 0;
   padding: 20px;
   min-height: 100vh;
   display: flex;
   flex-direction: column;
   align-items: center;
   justify-content: center;
   text-align: center;
   box-sizing: border-box;
  }

  h1 {
   font-size: 2rem;
   margin-bottom: 6px;
   color: #00bcd4;
  }

  h4 {
   font-size: 1rem;
   font-weight: 400;
   color: #cccccc;
   margin-bottom: 30px;
  }

  .button-container {
   display: flex;
   flex-direction: column;
   gap: 20px;
   width: 100%;
   max-width: 320px;
  }

  button {
   padding: 14px 20px;
   font-size: 1rem;
   font-weight: 500;
   color: #ffffff;
   background: linear-gradient(145deg, #0077b6, #023e8a);
   border: none;
   border-radius: 8px;
   cursor: pointer;
   transition: all 0.3s ease;
   width: 100%;
   box-shadow: 0 6px 15px rgba(0, 0, 0, 0.3);
  }

  button:hover {
```

```
    transform: scale(1.05);
    background: linear-gradient(145deg, #0096c7, #0077b6);
  }

  @media (max-width: 768px) {
    h1 {
      font-size: 1.6rem;
    }

    h4 {
      font-size: 0.9rem;
    }

    button {
      font-size: 0.95rem;
    }
  }

  @media (max-width: 480px) {
    h1 {
      font-size: 1.4rem;
    }

    h4 {
      font-size: 0.85rem;
    }

    .button-container {
      max-width: 100%;
    }

    button {
      font-size: 0.9rem;
      padding: 12px;
    }
  }
 </style>
</head>
<body>
 <h1>Aplikasi Deteksi Kematangan Buah Tomat</h1>
 <h4>Muh. Suryanto Rustam</h4>

 <div class="button-container">
  <form action="realtime.php" method="post">
   <button type="submit">Test Realtime</button>
  </form>
  <form action="test_gambar.php" method="post">
   <button type="submit">Test Gambar</button>
  </form>
  <form action="logout.php" method="post">
   <button type="submit" onclick="return confirmLogout()">Keluar</button>
  </form>
 </div>

 <script>
  function confirmLogout() {
   return confirm("Anda yakin ingin keluar?");
  }
 </script>
</body>
</html>
```

TAMPILAN MENU PENGUJIAN REALTIME

```
<!DOCTYPE html>
<html lang="id">
<head>
 <meta charset="UTF-8" />
 <meta name="viewport" content="width=device-width, initial-scale=1.0" />
```

```
<title>Prediksi Real-Time</title>
<link
      href="https://fonts.googleapis.com/css2?family=Poppins:wght@500;700&display=swap" rel="stylesheet" />
<style>
 body {
  font-family: 'Poppins', sans-serif;
  background: linear-gradient(135deg, #1f1f1f, #2c2c2c);
  color: #ffffff;
  margin: 0;
  padding: 20px;
  min-height: 100vh;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: flex-start;
  text-align: center;
  box-sizing: border-box;
 }

 h1 {
  font-size: 2rem;
  margin-top: 30px;
  margin-bottom: 10px;
  color: #00bcd4;
 }

 #webcam-container {
  margin-top: 20px;
 }

 #prediction-container {
  margin-top: 25px;
  font-size: 1.1rem;
  font-weight: 500;
  background-color: #37474f;
  padding: 12px 20px;
  border-radius: 8px;
  box-shadow: 0 4px 12px rgba(0, 0, 0, 0.4);
  max-width: 90%;
 }

#back-button {
  margin-top: 40px;
  padding: 14px 24px;
  font-size: 1rem;
  font-weight: 500;
  color: #ffffff;
  background: linear-gradient(145deg, #0077b6, #023e8a);
  border: none;
  border-radius: 8px;
  cursor: pointer;
  transition: all 0.3s ease;
  box-shadow: 0 6px 15px rgba(0, 0, 0, 0.3);
}

#back-button:hover {
  transform: scale(1.05);
  background: linear-gradient(145deg, #0096c7, #0077b6);
}

@media (max-width: 768px) {
  h1 {
   font-size: 1.6rem;
  }

  #prediction-container {
   font-size: 1rem;
  }

  #back-button {
   width: 100%;
  }
}
```

```
  @media (max-width: 480px) {
   h1 {
    font-size: 1.4rem;
   }

   #prediction-container {
    font-size: 0.95rem;
    padding: 10px 16px;
   }

   #back-button {
    font-size: 0.95rem;
    padding: 12px;
   }
  }
 </style>

 <script
      src="https://cdn.jsdelivr.net/
      npm/@tensorflow/tfjs@lates
      t/dist/tf.min.js"></script>
 <script
      src="https://cdn.jsdelivr.net/
      npm/@teachablemachine/im
      age@latest/dist/teachablema
      chine-
      image.min.js"></script>
</head>
<body>
 <h1>Prediksi Real-Time</h1>
 <div id="webcam-
      container"></div>
 <div id="prediction-
      container">Prediksi: -</div>
 <button id="back-button"
      onclick="goBack()">Kemba
      li ke Menu Utama</button>

 <script>
 const URL = "./model/";
 let model, webcam,
      maxPredictions;

async function init() {
 const modelURL = URL + "model.json";
 const metadataURL = URL +
      "metadata.json";

 model = await
      tmImage.load(modelURL,
      metadataURL);
 maxPredictions =
      model.getTotalClasses();

 const flip = false;
 webcam = new tmImage.Webcam(400,
      300, flip);

 try {
  await webcam.setup({ facingMode:
      "environment" });
  await webcam.play();
  document.getElementById("webcam-
      container").appendChild(webcam.c
      anvas);
  loop();
 } catch (error) {
  alert("Gagal mengakses kamera.
      Pastikan izin telah diberikan dan
      gunakan HTTPS.");
  console.error("Kesalahan webcam:",
      error);
 }
}

async function loop() {
 webcam.update();
 await predict();
 window.requestAnimationFrame(loop);
}

async function predict() {
 try {
  const predictions = await
      model.predict(webcam.canvas);
```

```
    const threshold = 0.5;

    // Ambil hanya yang di atas threshold
    const validPredictions = predictions.filter(pred => pred.probability > threshold);
    validPredictions.sort((a, b) => b.probability - a.probability);
    const topPredictions = validPredictions.slice(0, 2);

    let predictionsText = "Prediksi: ";

    if (topPredictions.length > 0) {
      predictionsText += topPredictions
        .map(pred => `${pred.className} (${(pred.probability * 100).toFixed(2)}%)`)
        .join(" dan ");
    } else {
      predictionsText = "Objek Bukan Bagian dari Dataset!";
    }

      document.getElementById("prediction-container").innerText = predictionsText;
  } catch (error) {
    console.warn("Gagal melakukan prediksi:", error);
  }
}

function goBack() {
  window.location.href = "index.php";
}

window.onload = init;
</script>
</body>
</html>
```

TAMPILAN MENU PENGUJIAN GAMBAR

```
<!DOCTYPE html>
<html lang="id">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <title>Test Gambar</title>
  <link href="https://fonts.googleapis.com/css2?family=Poppins:wght@500;700&display=swap" rel="stylesheet" />
  <script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@latest/dist/tf.min.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/@teachablemachine/image@latest/dist/teachablemachine-image.min.js"></script>
  <style>
    body {
      font-family: 'Poppins', sans-serif;
      background: linear-gradient(135deg, #1f1f1f, #2c2c2c);
      color: #ffffff;
      margin: 0;
      padding: 20px;
      min-height: 100vh;
      display: flex;
      flex-direction: column;
      align-items: center;
      box-sizing: border-box;
    }
```

```
h1 {
  font-size: 2rem;
  margin-top: 30px;
  margin-bottom: 20px;
  color: #00bcd4;
  text-align: center;
}

input[type="file"] {
  margin-top: 15px;
  padding: 12px;
  border: 2px dashed #00bcd4;
  border-radius: 10px;
  background-color: #263238;
  color: #ffffff;
  font-size: 1rem;
  width: 90%;
  max-width: 340px;
  transition: all 0.3s ease;
}

input[type="file"]:hover {
  background-color: #37474f;
  cursor: pointer;
}

button {
  margin-top: 25px;
  padding: 14px 28px;
  font-size: 1rem;
  font-weight: 500;
  color: #ffffff;
  background: linear-gradient(145deg, #0077b6, #023e8a);
  border: none;
  border-radius: 8px;
  cursor: pointer;
  transition: all 0.3s ease;
  box-shadow: 0 6px 15px rgba(0, 0, 0, 0.3);
}

button:hover {
  background: linear-gradient(145deg, #0096c7, #0077b6);
  transform: scale(1.05);
}

#selectedImage {
  max-width: 300px;
  height: auto;
  margin: 25px auto 10px;
  display: none;
  border-radius: 10px;
  box-shadow: 0 4px 12px rgba(0, 0, 0, 0.4);
}

#result {
  margin-top: 20px;
  font-size: 1.1rem;
  font-weight: 500;
  background-color: #37474f;
  color: #ffffff;
  padding: 14px 20px;
  border-radius: 10px;
  box-shadow: 0 4px 12px rgba(0, 0, 0, 0.4);
  max-width: 90%;
  text-align: center;
}

@media (max-width: 480px) {
  h1 {
    font-size: 1.5rem;
  }

  input[type="file"] {
    font-size: 0.95rem;
    padding: 10px;
  }
```

```
    button {
      width: 100%;
      font-size: 0.95rem;
      padding: 12px;
    }

    #result {
      font-size: 1rem;
      padding: 12px 16px;
    }
  }
 </style>
</head>
<body>
 <h1>Prediksi Gambar Statis</h1>

 <input type="file" id="imageUpload" accept="image/*" />
 <button type="button" onclick="predictImage()">Prediksi Gambar</button>

 <img id="selectedImage" src="" alt="" />

 <div id="result">Hasil prediksi akan muncul di sini.</div>

 <button onclick="goBack()">Kembali ke Menu Utama</button>

 <script>
  const URL = "./model/";
  let model;

  window.onload = async () => {
   const modelURL = URL + "model.json";
   const metadataURL = URL + "metadata.json";
   model = await tmImage.load(modelURL, metadataURL);
  };

  async function predictImage() {
   const imageUpload = document.getElementById('imageUpload');
   const image = imageUpload.files[0];
   const imgElement = document.getElementById('selectedImage');

   if (image) {
    const reader = new FileReader();
    reader.onload = async (event) => {
     imgElement.src = event.target.result;
     imgElement.style.display = 'block';

     const processedImage = new Image();
     processedImage.onload = async () => {
      const predictions = await model.predict(processedImage);
      displayPrediction(predictions);
     };
     processedImage.src = event.target.result;
    };
    reader.readAsDataURL(image);
   } else {
    alert("Silakan pilih gambar terlebih dahulu.");
   }
  }

  function displayPrediction(predictions) {
   const resultContainer = document.getElementById('result');
   resultContainer.innerHTML = '';
```

```
    const threshold = 0.5;
    const validPredictions = predictions.filter(pred => pred.probability > threshold);

    if (validPredictions.length > 0) {
      let predictionsText = "<strong>Prediksi:</strong><br>";
      validPredictions.forEach(pred => {
        predictionsText += ${pred.className} - ${(pred.probability * 100).toFixed(2)}%<br>;
      });
      resultContainer.innerHTML = predictionsText;
    } else {
      resultContainer.innerHTML = "Tidak ada prediksi yang valid.";
    }
  }

  function goBack() {
    window.location.href = "index.php";
  }
 </script>
</body>
</html>
```