

LAMPIRAN

Lampiran 1 Surat Kesiediaan Membimbing TA Pembimbing 1

SURAT KESEDIAAN MEMBIMBING TA

Yang bertanda tangan di bawah ini:

NAMA : Wildani Eko Nugroho, M.Kom.
NIDN : 0617078204
NIPY : 12.013.169
Jabatan Struktural : Dosen Tetap
Jabatan Fungsional : Lektor

Dengan ini menyatakan bersedia untuk menjadi pembimbing I pada Tugas Akhir mahasiswa berikut:

Nama : Mohammad Maulana Ikhsan
NIM : 23040093
Program Studi : D-3 Teknik Komputer
Judul TA : SISTEM INFORMASI DETEKSI PENYAKIT
TUMBUHAN PISANG BERBASIS WEBSITE
MENGUNAKAN MERN STACK DAN
MACHINE LEARNING

Demikian pernyataan ini dibuat agar dapat dilaksanakan sebagaimana mestinya.

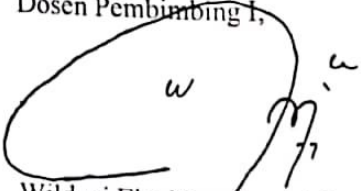
Tegal, 12 Juni 2026

Menyetujui

Ka Prodi DIII Teknik Komputer,


Arif Rakhman, SE, S.Pd, M.Kom.
NIPY. 05.016.291

Dosen Pembimbing I,


Wildani Eko Nugroho, M.Kom.
NIPY. 12.013.169

Lampiran 2 Surat Kesediaan Membimbing TA Pembimbing 2

SURAT KESEDIAAN MEMBIMBING TA

Yang bertanda tangan di bawah ini:

NAMA : Lukmanul Khakim, S.Kom., M.Tr.T., IPP
NIDN : 06040590004
NIPY : 08.017.343
Jabatan Struktural : Dosen Tetap
Jabatan Fungsional : Lektor

Dengan ini menyatakan bersedia untuk menjadi pembimbing II pada Tugas Akhir mahasiswa berikut:

Nama : Mohammad Maulana Ikhsan
NIM : 23040093
Program Studi : D-3 Teknik Komputer
Judul TA : SISTEM INFORMASI DETEKSI PENYAKIT
TUMBUHAN PISANG BERBASIS WEBSITE
MENGGUNAKAN MERN STACK DAN
MACHINE LEARNING

Demikian pernyataan ini dibuat agar dapat dilaksanakan sebagaimana mestinya.

Tegal, 12 Juni 2026

Menyetujui

Ka Prodi DIII Teknik Komputer,

Dosen Pembimbing II,


Arif Rakhman, SE, S.Pd, M.Kom.
NIPY. 05.016.291


Lukmanul Khakim, S.Kom., M.Tr.T., IPP
NIPY. 08.017.343

Lampiran 3 Surat Observasi di Kebun Pisang Ibnu

Nomor : 639.03/D-3 T.Komp.HN/VI/2026
Lampiran : -
Perihal : Permohonan Izin Observasi Tugas Akhir (TA)

Kepada Yth.

Kepala Kebun Pisang Ibnu

Di Jl. Cut Nyak Dien, Kalisapu, Kec. Slawi, Kabupaten Tegal

Dengan Hormat,

Sehubungan dengan tugas mata kuliah Tugas Akhir (TA) yang akan diselenggarakan di semester V Program Studi Diploma III Teknik Komputer Universitas Harkat Negeri, Maka dengan ini kami mengajukan izin observasi pengambilan data di Kebun Pisang Ibnu yang Bapak / Ibu Pimpin, untuk kepentingan dalam pembuatan produk Tugas Akhir, dengan Mahasiswa sebagai berikut:

No	NIM	NAMA	NO HP
1	23040093	MOHAMMAD MAULANA IKHSAN	085865839163

Demikian surat permohonan ini kami sampaikan atas izin dan kerjasamanya kami sampaikan terima kasih.

Tegal, 15 Juni 2026
Ka. Prodi DIII Teknik Komputer
Universitas Harkat Negeri


**Universitas
Harkat Negeri**
DIII Teknik Komputer
Arif Rakhman SE, S.Pd, M.Kom
NIPY. 05.016.291

Lampiran 4 Surat Observasi di Dinas Pertanian

Nomor : 449.03/KOM-PHB/XI/2025
Lampiran : -
Perihal : Permohonan Izin Observasi

Kepada Yth.

Pimpinan Dinas Ketahanan Pangan dan Pertanian Kabupaten Tegal
Di Tempat

Dengan Hormat,

Sehubungan dengan tugas mata kuliah Tugas Akhir yang diselenggarakan di semester 5 Program Studi DIII Teknik Komputer Universitas Harkat Negeri, maka bersama ini kami mengajukan izin observasi di Dinas Ketahanan Pangan dan Pertanian Kabupaten Tegal yang Bapak / Ibu pimpin untuk kepentingan pembuatan Tugas Mata Kuliah Tugas Akhir dengan mahasiswa sebagai berikut:

NO.	NIM	NAMA	NO HP
1	23040095	MUHAMMAD YAHYA AL HAKIM	088221899708
2	23040093	MOHAMMAD MAULANA IKHSAN	085865839163
3	23040037	AZRIL RESTU MAHINATA	082322486018

Demikian surat permohonan ini kami sampaikan atas izin dan kerjasamanya kami sampaikan terima kasih.

Tegal, 10 Juni 2026
Ka. Prodi DIII Teknik Komputer
Universitas Harkat Negeri



Arif Rakhman SE, S.Pd. M.Kom
NIPY. 05.016.291

Lampiran 5 Surat Balasan Observasi di Dinas Pertanian



PEMERINTAH KABUPATEN TEGAL
DINAS KETAHANAN PANGAN DAN PERTANIAN

Jalan Jend. Ahmad Yani No 17 A Slawi, Kode Pos 52412
Telepon (0283) 491513 Faks (0283) 491872
E-mail : tankp@tegalkab.go.id

Slawi, 2 Desember 2025

Nomor : 400.14.5.4/1843-2/20
Sifat : Biasa
Lampiran : -
Hal : Izin Observasi

Yth. Kepala Program Studi D-III Teknik Komputer
Politeknik Harapan Bersama Tegal
di
Tegal

Menanggapi surat Saudara Nomor : 449.03/KOM-PHB/XI/2025 tanggal 27 November 2025 perihal Permohonan Izin Observasi, bersama ini disampaikan bahwa pada prinsipnya kami tidak keberatan memberikan Izin Observasi atas nama :

No	Nama	NIM	Program Studi	No. Handphone
1	Muhammad Yahya Al Hakim	23040095	D-III Teknik Komputer	088221899708
2	Mohammad Maulana Ikhsan	23040093	D-III Teknik Komputer	085865839163
3	Azril Restu Mahinata	23040037	D-III Teknik Komputer	082138206792

Demikian untuk menjadikan maklum dan terima kasih.

Kepala Dinas Ketahanan Pangan dan
Pertanian,


Agus Sukoco, S.P. M.Si.
Pemimpin Utama Muda (IV/c)
NIP. 197103251996031003

Tembusan :

1. Yang bersangkutan;
2. Arsip.

Lampiran 6 Bimbingan Laporan Pembimbing TA

Lampiran 22
Bimbingan Proposal TA

IK P2M PHB d.5.1.e.1

NAMA MAHASISWA :
PEMBIMBING I:

BIMBINGAN PROPOSAL TA

No	HARI/ TANGGAL	URAIAN	TANDA TANGAN
1.	Pekan / 18-2-2026	Revisi : - format penulisan - latar belakang - Rumusan masalah - Batasan masalah - Tujuan / Manfaat penelitian - Teori terkait - Landasan Teori - Metodologi Penelitian - Jadwal Kegiatan - Rencana Kerja	
2.	Konfirmasi 26/2-2026	Revisi : - Hal. Judul. - Hal Pengantar - Daftar isi, Daftar gambar - Daftar tabel. - Latar belakang, Rumusan masalah, Batasan masalah - Manfaat, Teori terkait. - Metodologi Penelitian. - Format penulisan - Format tabel	

No	HARI/ TANGGAL	URAIAN	TANDA TANGAN
3.	Jenin. 2/5-2026	Revisi : - Hal Judul. - Daftar isi, Gambar, Tabel - Rumusan Masalah. - format penulisan - Metodologi penelitian (Perancangan System) - Study Literatur (Dataset yg dipakai) - Revisi proposal Langkah Bab 1, 2, 3 <i>[Signature]</i> 2/5-2026 <i>[Signature]</i>	<i>[Signature]</i>
4.	Babar 1/4-2026	Revisi layout Bab. 1, 2, 3 : - format tabel dan gambar - Sistematika penulisan	<i>[Signature]</i>

No	HARI/ TANGGAL	URAIAN	TANDA TANGAN
5	Rahm. 1/4-2026	Di Bab 1, 2, 3 lanjut ke pembahasan 2 Bab 4, 5, 6 dan Proses TA	Cj
6.	Selwa. 2/6-2026.	Peneri Bab 4.: - Identifikasi aktor dan Usecase. - Use case diagram - Sequence diagram - Activity diagram Bab 5 dan 6	Cj

PEMBIMBING II:

BIMBINGAN LAPORAN TA

No	HARI/ TANGGAL	URAIAN	TANDA TANGAN
1.	19/05-2026	- perbaiki keterangan gambar - penambahan Analisa permasalahan	
2.	25/05-2026	ACC BAB 4-5 lanjutkan BAB 6 !!	
3.	26/05-2026	- Acc BAB 6 - Acc proyek TA	

Lampiran 7 Dokumentasi Hasil Observasi



Lampiran 8 Dokumentasi Hasil Pengujian



Lampiran 9 Source Code

```
import warnings
warnings.filterwarnings('ignore')

import os
import json
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.applications import ResNet50, MobileNetV2
from tensorflow.keras.applications.resnet50 import
preprocess_input as resnet_preprocess
from tensorflow.keras.applications.mobilenet_v2 import
preprocess_input as mobilenet_preprocess
from tensorflow.keras.preprocessing.image import
ImageDataGenerator
from tensorflow.keras.callbacks import (EarlyStopping,
ModelCheckpoint, ReduceLROnPlateau, CSVLogger)
from sklearn.metrics import (classification_report,
confusion_matrix, precision_recall_fscore_support)
from sklearn.utils.class_weight import compute_class_weight

print(f"TensorFlow : {tf.__version__}")
print(f"GPU      :
{len(tf.config.list_physical_devices('GPU'))} device(s)")

SEED          = 42
IMG_SIZE      = (224, 224)
BATCH_SIZE    = 32
EPOCHS_HEAD   = 30
EPOCHS_FT     = 15
VAL_SPLIT     = 0.2

np.random.seed(SEED)
tf.random.set_seed(SEED)

DATA_DIR = '/kaggle/working/banana-disease-recognition-
dataset/Augmented images/Augmented images'

FOLDER_TO_CLEAN = {
    'Augmented Banana Black Sigatoka Disease'      : 'Black
Sigatoka',
    'Augmented Banana Bract Mosaic Virus Disease' : 'Bract
Mosaic Virus',
    'Augmented Banana Healthy Leaf'                  : 'Healthy
Leaf',
    'Augmented Banana Insect Pest Disease'          : 'Insect
Pest',
```

```

        'Augmented Banana Moko Disease'           : 'Moko
Disease',
        'Augmented Banana Panama Disease'         : 'Panama
Disease',
        'Augmented Banana Yellow Sigatoka Disease' : 'Yellow
Sigatoka',
    }

NUM_CLASSES = len(FOLDER_TO_CLEAN)

!git clone https://github.com/ikhsan-kun/banana-disease-
recognition-dataset.git
!rm -rf /kaggle/working/banana-disease-recognition-dataset/.git

print("\n" + "="*65)
print("ANALISIS DISTRIBUSI DATASET")
print("="*65)

class_counts = {}
for cls in sorted(os.listdir(DATA_DIR)):
    path = os.path.join(DATA_DIR, cls)
    if os.path.isdir(path):
        n = len([f for f in os.listdir(path)
                  if f.lower().endswith(('.jpg', '.jpeg',
'.png'))])
        class_counts[cls] = n

total_images = sum(class_counts.values())
print(f"\nTotal gambar : {total_images}")
print(f"Jumlah kelas : {len(class_counts)}\n")
print(f"{'Kelas':<48} {'Gambar':>6} {'%':>5}")
print("-"*65)
for cls, cnt in sorted(class_counts.items(), key=lambda x: -
x[1]):
    bar = "#" * int(cnt / total_images * 40)
    print(f"  {cls:<46} {cnt:>6} {cnt/total_images*100:>4.1f}%
{bar}")
print(f"\nImbalance ratio : "

f"{max(class_counts.values())/min(class_counts.values()):.1f}x")

def make_generators(preprocess_fn, val_split=VAL_SPLIT,
                    img_size=IMG_SIZE, batch=BATCH_SIZE):
    train_datagen = ImageDataGenerator(
        preprocessing_function=preprocess_fn,
        rotation_range=10,
        width_shift_range=0.1,
        height_shift_range=0.1,
        shear_range=0.1,
        zoom_range=0.1,
        horizontal_flip=True,
        vertical_flip=False,
        brightness_range=[0.8, 1.2],
        channel_shift_range=20,

```

```

        fill_mode='nearest',
        validation_split=val_split
    )
    val_datagen = ImageDataGenerator(
        preprocessing_function=preprocess_fn,
        validation_split=val_split
    )
    train_gen = train_datagen.flow_from_directory(
        DATA_DIR, target_size=img_size, batch_size=batch,
        class_mode='categorical', subset='training',
        shuffle=True, seed=SEED
    )
    val_gen = val_datagen.flow_from_directory(
        DATA_DIR, target_size=img_size, batch_size=batch,
        class_mode='categorical', subset='validation',
        shuffle=False, seed=SEED
    )
    return train_gen, val_gen

print("\n" + "="*65)
print("SETUP DATA GENERATORS")
print("="*65)

resnet_train,    resnet_val    =
make_generators(resnet_preprocess)
mobilenet_train, mobilenet_val =
make_generators(mobilenet_preprocess)

print(f"\nTrain samples      : {resnet_train.samples}")
print(f"Validation samples : {resnet_val.samples}")
print(f"Batch size          : {BATCH_SIZE}")
print(f"Input size           : {IMG_SIZE}")

idx_to_clean = {
    idx: FOLDER_TO_CLEAN[folder]
    for folder, idx in resnet_train.class_indices.items()
}
CLEAN_NAMES = [idx_to_clean[i] for i in range(NUM_CLASSES)]

print("\nUrutan kelas (sinkron dengan generator):")
print("-"*55)
for i, name in enumerate(CLEAN_NAMES):
    folder = [f for f, c in FOLDER_TO_CLEAN.items() if c ==
name][0]
    print(f"  [{i}] {name:<25} <- {folder}")

print("\n" + "="*65)
print("CLASS WEIGHTS (smoothed, power 0.5)")
print("="*65)

cw_raw    = compute_class_weight('balanced',
classes=np.unique(resnet_train.classes),
                                y=resnet_train.classes)
cw_smooth = cw_raw ** 0.5
class_weights = {i: float(w) for i, w in enumerate(cw_smooth)}

```

```

print(f"\n  {'Kelas':<25}  {'Raw':>6}  {'Smooth':>8}")
print("  " + "-"*42)
for i, name in enumerate(CLEAN_NAMES):
    print(f"  [{i}] {name:<23}  {cw_raw[i]:>6.3f}
{cw_smooth[i]:>8.3f}")
print(f"\nGap raw      : {cw_raw.max()/cw_raw.min():.1f}x")
print(f"Gap smooth : {cw_smooth.max()/cw_smooth.min():.1f}x  <-
lebih seimbang")

viz_datagen = ImageDataGenerator(rescale=1./255,
validation_split=VAL_SPLIT)
viz_gen = viz_datagen.flow_from_directory(
    DATA_DIR, target_size=IMG_SIZE, batch_size=32,
    class_mode='categorical', subset='training',
    shuffle=True, seed=SEED
)

images_viz, labels_viz = next(viz_gen)

fig, axes = plt.subplots(3, 8, figsize=(22, 9))
for img, lbl, ax in zip(images_viz[:24], labels_viz[:24],
axes.flatten()):
    ax.imshow(img)
    ax.set_title(CLEAN_NAMES[np.argmax(lbl)], fontsize=7)
    ax.axis('off')

plt.suptitle('Sample Training Images (setelah augmentasi)',
            fontsize=13, fontweight='bold')
plt.tight_layout()
plt.savefig('sample_images.png', dpi=150, bbox_inches='tight')
plt.show()
print("Saved: sample_images.png")

def build_head(base_output):
    x = layers.GlobalAveragePooling2D()(base_output)
    x = layers.Dense(256, activation='relu',

kernel_regularizer=keras.regularizers.l2(0.01))(x)
    x = layers.Dropout(0.5)(x)
    x = layers.Dense(128, activation='relu',

kernel_regularizer=keras.regularizers.l2(0.01))(x)
    x = layers.Dropout(0.4)(x)
    x = layers.Dense(NUM_CLASSES, activation='softmax')(x)
    return x

def make_callbacks(model_name, phase):
    return [
        EarlyStopping(
            monitor='val_loss',
            patience=7 if phase == 1 else 5,
            restore_best_weights=True,
            mode='min',
            verbose=1
        ),

```

```

        ReduceLROnPlateau(
            monitor='val_loss',
            factor=0.5,
            patience=3,
            min_lr=1e-8,
            verbose=1
        ),
        ModelCheckpoint(
            f'best_{model_name}_phase{phase}.keras',
            monitor='val_loss',
            save_best_only=True,
            mode='min',
            verbose=1
        ),
        CSVLogger(f'log_{model_name}_phase{phase}.csv')
    ]

def plot_history(hist_dict, model_name, phase_epochs=None):
    fig, axes = plt.subplots(1, 2, figsize=(14, 5))
    fig.suptitle(f'Training History - {model_name}',
                 fontsize=14, fontweight='bold')

    for ax, (train_k, val_k, label) in zip(
        axes,
        [('accuracy', 'val_accuracy', 'Accuracy'),
         ('loss', 'val_loss', 'Loss')]):
        ax.plot(hist_dict[train_k], label='Train',
                linewidth=2, color='#2196F3')
        ax.plot(hist_dict[val_k], label='Validation',
                linewidth=2, color='#FF9800')
        if phase_epochs:
            ax.axvline(phase_epochs - 1, color='red',
                      linestyle='--', alpha=0.7, label='Fine-
tune start')
        ax.set_title(label, fontweight='bold')
        ax.set_xlabel('Epoch')
        ax.legend()
        ax.grid(True, alpha=0.3)

    plt.tight_layout()
    fname = f'curves_{model_name.lower().replace(" ", "_")}.png'
    plt.savefig(fname, dpi=150, bbox_inches='tight')
    plt.show()
    print(f"Saved: {fname}")

def evaluate_model(model, val_generator, model_name):
    val_generator.reset()
    y_pred_proba = model.predict(val_generator, verbose=0)
    y_pred = np.argmax(y_pred_proba, axis=1)
    y_true = val_generator.classes

    precision, recall, f1, support =
precision_recall_fscore_support(
    y_true, y_pred, average=None, zero_division=0)
    acc = np.mean(y_pred == y_true)
    macro_f1 = np.mean(f1)

```

```

cm          = confusion_matrix(y_true, y_pred)

# Confusion Matrix + F1 bar chart
fig, axes = plt.subplots(1, 2, figsize=(18, 7))

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=CLEAN_NAMES,
            yticklabels=CLEAN_NAMES,
            ax=axes[0], cbar_kws={'label': 'Count'})
axes[0].set_title(f'{model_name} - Confusion Matrix',
                 fontweight='bold', fontsize=13)
axes[0].set_xlabel('Predicted', fontweight='bold')
axes[0].set_ylabel('True Label', fontweight='bold')
axes[0].tick_params(axis='x', rotation=45)

bar_colors = ['#4CAF50' if s >= 0.80 else
              '#FF9800' if s >= 0.60 else '#F44336'
              for s in f1]
bars = axes[1].barh(CLEAN_NAMES, f1, color=bar_colors,
                   edgecolor='black', linewidth=1.2)
for bar, score in zip(bars, f1):
    axes[1].text(bar.get_width() + 0.01,
                 bar.get_y() + bar.get_height() / 2,
                 f'{score:.3f}', va='center',
                 fontsize=10, fontweight='bold')
axes[1].set_xlim(0, 1.15)
axes[1].set_xlabel('F1-Score', fontweight='bold')
axes[1].set_title(f'{model_name} - F1-Score per Class',
                 fontweight='bold', fontsize=13)
axes[1].axvline(0.80, color='green', linestyle='--',
               alpha=0.6, label='Target >= 0.80')
axes[1].legend()
axes[1].grid(axis='x', alpha=0.3)

plt.tight_layout()
fname = f'eval_{model_name.lower().replace(" ", "_")}.png'
plt.savefig(fname, dpi=150, bbox_inches='tight')
plt.show()
print(f"Saved: {fname}")

print(f"\n{'='*65}")
print(f"CLASSIFICATION REPORT - {model_name}")
print(f"{'='*65}")
print(classification_report(y_true, y_pred,
                           target_names=CLEAN_NAMES,
                           digits=4, zero_division=0))
print(f"Overall Accuracy : {acc:.4f} ({acc*100:.2f}%)")
print(f"Macro F1-Score    : {macro_f1:.4f}")
print(f"Macro Precision    : {np.mean(precision):.4f}")
print(f"Macro Recall       : {np.mean(recall):.4f}")

return {
    'accuracy': acc,
    'macro_f1': macro_f1,
    'precision': precision,
    'recall': recall,

```

```

        'f1':          f1,
        'cm':          cm
    }

print("\n" + "="*65)
print("MODEL 1 - ResNet50 (Transfer Learning)")
print("="*65)

base_resnet = ResNet50(
    weights='imagenet',
    include_top=False,
    input_shape=(*IMG_SIZE, 3)
)
base_resnet.trainable = False

inputs_rn    = keras.Input(shape=(*IMG_SIZE, 3))
x_rn         = base_resnet(inputs_rn, training=False)
out_rn        = build_head(x_rn)
model_resnet = keras.Model(inputs_rn, out_rn,
    name='ResNet50_BananaDisease')

model_resnet.compile(
    optimizer=keras.optimizers.Adam(learning_rate=1e-4),
    loss='categorical_crossentropy',
    metrics=['accuracy',
        keras.metrics.Precision(name='precision'),
        keras.metrics.Recall(name='recall'),
        keras.metrics.AUC(name='auc')]
)

trainable_p1 = sum(keras.backend.count_params(w)
    for w in model_resnet.trainable_weights)
print(f"\nTrainable params (Phase 1) : {trainable_p1:,}")
model_resnet.summary(line_length=100)

print("\nPhase 1 - Head Training (frozen base, max 30
epoch)...")
history_rn_p1 = model_resnet.fit(
    resnet_train,
    validation_data=resnet_val,
    epochs=EPOCHS_HEAD,
    callbacks=make_callbacks('resnet50', 1),
    class_weight=class_weights,
    verbose=1
)
model_resnet =
keras.models.load_model('best_resnet50_phase1.keras')
p1_rn_epochs = len(history_rn_p1.history['loss'])
print(f"Phase 1 selesai di epoch {p1_rn_epochs}")

print("\nPhase 2 - Fine-Tuning (unfreeze 20 layer terakhir)...")

base_resnet.trainable = True
freeze_until_rn = len(base_resnet.layers) - 20
for i, layer in enumerate(base_resnet.layers):
    layer.trainable = i >= freeze_until_rn

```

```

        if isinstance(layer, layers.BatchNormalization):
            layer.trainable = False

trainable_p2 = sum(keras.backend.count_params(w)
                    for w in model_resnet.trainable_weights)
print(f"Trainable params (Phase 2) : {trainable_p2:,}")

model_resnet.compile(
    optimizer=keras.optimizers.Adam(learning_rate=1e-5),
    loss='categorical_crossentropy',
    metrics=['accuracy',
             keras.metrics.Precision(name='precision'),
             keras.metrics.Recall(name='recall'),
             keras.metrics.AUC(name='auc')]
)

history_rn_p2 = model_resnet.fit(
    resnet_train,
    validation_data=resnet_val,
    epochs=EPOCHS_FT,
    callbacks=make_callbacks('resnet50', 2),
    class_weight=class_weights,
    verbose=1
)

model_resnet =
keras.models.load_model('best_resnet50_phase2.keras')
print(f"Phase 2 selesai di epoch "
      f"{p1_rn_epochs + len(history_rn_p2.history['loss'])}")

history_resnet = {
    k: history_rn_p1.history[k] + history_rn_p2.history[k]
    for k in history_rn_p1.history
}

print("\n" + "="*65)
print("EVALUASI RESNET50")
print("="*65)
plot_history(history_resnet, 'ResNet50',
             phase1_epochs=p1_rn_epochs)
result_resnet = evaluate_model(model_resnet, resnet_val,
                               'ResNet50')

print("\n" + "="*65)
print("MODEL 2 - MobileNetV2 (Transfer Learning)")
print("="*65)

base_mobilenet = MobileNetV2(
    weights='imagenet',
    include_top=False,
    input_shape=(*IMG_SIZE, 3)
)
base_mobilenet.trainable = False

inputs_mb      = keras.Input(shape=(*IMG_SIZE, 3))
x_mb           = base_mobilenet(inputs_mb, training=False)
out_mb         = build_head(x_mb)

```

```

model_mobilenet = keras.Model(inputs_mb, out_mb,
name='MobileNetV2_BananaDisease')

model_mobilenet.compile(
    optimizer=keras.optimizers.Adam(learning_rate=1e-4),
    loss='categorical_crossentropy',
    metrics=['accuracy',
             keras.metrics.Precision(name='precision'),
             keras.metrics.Recall(name='recall'),
             keras.metrics.AUC(name='auc')]
)

trainable_mb_p1 = sum(keras.backend.count_params(w)
                      for w in
model_mobilenet.trainable_weights)
print(f"\nTrainable params (Phase 1) : {trainable_mb_p1:,}")
model_mobilenet.summary(line_length=100)

print("\nPhase 1 - Head Training (frozen base, max 30
epoch)...")
history_mb_p1 = model_mobilenet.fit(
    mobilenet_train,
    validation_data=mobilenet_val,
    epochs=EPOCHS_HEAD,
    callbacks=make_callbacks('mobilenetv2', 1),
    class_weight=class_weights,
    verbose=1
)
model_mobilenet =
keras.models.load_model('best_mobilenetv2_phase1.keras')
p1_mb_epochs = len(history_mb_p1.history['loss'])
print(f"Phase 1 selesai di epoch {p1_mb_epochs}")

print("\nPhase 2 - Fine-Tuning (unfreeze 20 layer terakhir)...")

base_mobilenet.trainable = True
freeze_until_mb = len(base_mobilenet.layers) - 20
for i, layer in enumerate(base_mobilenet.layers):
    layer.trainable = i >= freeze_until_mb
    if isinstance(layer, layers.BatchNormalization):
        layer.trainable = False

trainable_mb_p2 = sum(keras.backend.count_params(w)
                      for w in
model_mobilenet.trainable_weights)
print(f"Trainable params (Phase 2) : {trainable_mb_p2:,}")

model_mobilenet.compile(
    optimizer=keras.optimizers.Adam(learning_rate=1e-5),
    loss='categorical_crossentropy',
    metrics=['accuracy',
             keras.metrics.Precision(name='precision'),
             keras.metrics.Recall(name='recall'),
             keras.metrics.AUC(name='auc')]
)

```

```

history_mb_p2 = model_mobilenet.fit(
    mobilenet_train,
    validation_data=mobilenet_val,
    epochs=EPOCHS_FT,
    callbacks=make_callbacks('mobilenetv2', 2),
    class_weight=class_weights,
    verbose=1
)
model_mobilenet =
keras.models.load_model('best_mobilenetv2_phase2.keras')
print(f"Phase 2 selesai di epoch "
      f"{p1_mb_epochs + len(history_mb_p2.history['loss'])}")

history_mobilenet = {
    k: history_mb_p1.history[k] + history_mb_p2.history[k]
    for k in history_mb_p1.history
}

print("\n" + "="*65)
print("EVALUASI MOBILENETV2")
print("="*65)
plot_history(history_mobilenet, 'MobileNetV2',
             phase1_epochs=p1_mb_epochs)
result_mobilenet = evaluate_model(model_mobilenet,
mobilenet_val, 'MobileNetV2')

print("\n" + "="*65)
print("PERBANDINGAN ResNet50 vs MobileNetV2")
print("="*65)

results = {
    'ResNet50': result_resnet,
    'MobileNetV2': result_mobilenet
}

comp = pd.DataFrame([
    {
        'Model'      : name,
        'Accuracy'   : r['accuracy'],
        'Macro F1'   : r['macro_f1'],
        'Precision'  : np.mean(r['precision']),
        'Recall'     : np.mean(r['recall'])
    }
    for name, r in results.items()
]).sort_values('Macro F1',
ascending=False).reset_index(drop=True)

print("\n" + comp.to_string(index=False,
float_format='{:.4f}'.format))
best_model_name = comp.iloc[0]['Model']
print(f"\nModel terbaik : {best_model_name}")

# Bar chart perbandingan semua metrik
metrics_plot = ['Accuracy', 'Macro F1', 'Precision', 'Recall']
colors        = ['#2196F3', '#4CAF50', '#FF9800', '#9C27B0']
x             = np.arange(len(comp))

```

```

width = 0.18

fig, ax = plt.subplots(figsize=(11, 5))
for i, (metric, color) in enumerate(zip(metrics_plot, colors)):
    bars = ax.bar(x + i * width, comp[metric], width,
                  label=metric, color=color, alpha=0.85,
                  edgecolor='black')
    for bar, val in zip(bars, comp[metric]):
        ax.text(bar.get_x() + bar.get_width() / 2,
                bar.get_height() + 0.008,
                f'{val:.3f}', ha='center', va='bottom',
                fontsize=9, fontweight='bold')

ax.set_xticks(x + width * 1.5)
ax.set_xticklabels(comp['Model'], fontsize=12,
                  fontweight='bold')
ax.set_ylim(0, 1.15)
ax.set_title('ResNet50 vs MobileNetV2 - All Metrics',
             fontsize=14, fontweight='bold')
ax.set_ylabel('Score')
ax.legend(loc='upper right')
ax.axhline(0.80, color='red', linestyle='--', alpha=0.5)
ax.grid(axis='y', alpha=0.3)
plt.tight_layout()
plt.savefig('model_comparison.png', dpi=150,
          bbox_inches='tight')
plt.show()
print("Saved: model_comparison.png")

# F1 per kelas side-by-side
fig, axes = plt.subplots(1, 2, figsize=(16, 6), sharey=True)
fig.suptitle('F1-Score per Kelas - ResNet50 vs MobileNetV2',
             fontsize=13, fontweight='bold')

for ax, (name, res) in zip(axes, results.items()):
    fl_scores = res['f1']
    bar_colors = ['#4CAF50' if s >= 0.80 else
                  '#FF9800' if s >= 0.60 else '#F44336'
                  for s in fl_scores]
    bars = ax.barh(CLEAN_NAMES, fl_scores,
                  color=bar_colors, edgecolor='black',
                  linewidth=1.1)
    for bar, score in zip(bars, fl_scores):
        ax.text(bar.get_width() + 0.01,
                bar.get_y() + bar.get_height() / 2,
                f'{score:.3f}', va='center', fontsize=9.5,
                fontweight='bold')
    ax.set_xlim(0, 1.18)
    ax.set_title(f'{name} (F1 avg: {np.mean(fl_scores):.3f})',
                 fontweight='bold', fontsize=12)
    ax.axvline(0.80, color='red', linestyle='--', alpha=0.5)
    ax.grid(axis='x', alpha=0.3)

plt.tight_layout()
plt.savefig('f1_comparison.png', dpi=150, bbox_inches='tight')
plt.show()

```

```

print("Saved: f1_comparison.png")

print("\n" + "="*65)
print("MENYIMPAN MODEL DAN ARTEFAK")
print("="*65)

model_resnet.save('model_resnet50_final.keras')
model_mobilenet.save('model_mobilenetv2_final.keras')

comp.to_csv('model_comparison.csv', index=False)

class_mapping = {
    'class_indices' : resnet_train.class_indices,
    'clean_names'   : CLEAN_NAMES,
    'num_classes'   : NUM_CLASSES,
    'img_size'      : list(IMG_SIZE)
}
with open('class_mapping.json', 'w') as f:
    json.dump(class_mapping, f, indent=2)

print("\nFile tersimpan:")
print("  model_resnet50_final.keras")
print("  model_mobilenetv2_final.keras")
print("  model_comparison.csv")
print("  class_mapping.json")
print("  best_resnet50_phase1.keras")
print("  best_resnet50_phase2.keras")
print("  best_mobilenetv2_phase1.keras")
print("  best_mobilenetv2_phase2.keras")
print("  curves_resnet50.png")
print("  curves_mobilenetv2.png")
print("  eval_resnet50.png")
print("  eval_mobilenetv2.png")
print("  model_comparison.png")
print("  f1_comparison.png")
print("  sample_images.png")

```