L
A
M
P
I
R
A
N

MobileNetV2.ipynb

```
from google.colab import drive
drive.mount('/content/drive')
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers,
       models #type: ignore
from
       tensorflow.keras.preprocessing
       .image import
       ImageDataGenerator #type:
       ignore
from tensorflow.keras.applications
       import MobileNetV2 #type:
       ignore
from
       tensorflow.keras.applications.
       mobilenet_v2 import
       preprocess_input as
       preprocessor_mobilenetv2
       #type: ignore
from tensorflow.keras.optimizers
       import Adam #type: ignore
from tensorflow.keras.callbacks
       import ReduceLROnPlateau,
       EarlyStopping #type: ignore
from tensorflow.keras.layers import
       Dense, GlobalAveragePooling2D,
       Dropout, Concatenate, Input,
       Average #type: ignore
from tensorflow.keras.models import
       Model #type: ignore
from tensorflow.keras.utils import
       plot_model #type: ignore
from sklearn.metrics import
       confusion_matrix,
       classification_report,
       precision_score, recall_score,
       f1_score, accuracy_score
from collections import Counter
from tensorflow.keras.losses import
       CategoricalCrossentropy #type:
       ignore
from tensorflow.keras.utils import
       to_categorical #type: ignore
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os
import json
data_dir =
       '/content/drive/MyDrive/image-
       dataset/kue/DATASET v3.2.1'
total_images = sum([len(files) for _,
       _, files in
       os.walk(data_dir)])
print(f"Jumlah data di dataset\t\t:
       {total_images} gambar")
print('------------------------------
       -------------------')

train_dir = data_dir + '/training'
val_dir = data_dir + '/validation'
test_dir = data_dir + '/testing'

def count_files(directory):
    total_files = 0
    for subdir, _, files in
       os.walk(directory):
        total_files += len(files)
    return total_files

# Menampilkan jumlah file dalam
       setiap direktori
print(f"Jumlah data di training
       set\t:
       {count_files(train_dir)}
       gambar ")
print(f"Jumlah data di validation
       set\t: {count_files(val_dir)}
       gambar")
print(f"Jumlah data di testing set
       \t: {count_files(test_dir)}
       gambar")


create_generators(batch_size,
seed=42):
    train_datagen =
ImageDataGenerator(
        preprocessing_function=prepro
cessor_mobilenetv2,
        rotation_range=15,
        width_shift_range=0.2,
        height_shift_range=0.2,
        shear_range=0.1,
        zoom_range=0.2,
        horizontal_flip=True,
        brightness_range=(0.9, 1.1),
        fill_mode='nearest',
    )

    val_datagen = ImageDataGenerator(
        preprocessing_function=prepro
cessor_mobilenetv2,
    )

    test_datagen =
ImageDataGenerator(
        preprocessing_function=prepro
cessor_mobilenetv2,
    )

    train_gen =
train_datagen.flow_from_directory(
        train_dir,
        target_size=(224, 224),
        batch_size=batch_size,
        class_mode="categorical",
        shuffle=True,
        seed=seed
    )

    val_gen =
val_datagen.flow_from_directory(
        val_dir,
        target_size=(224, 224),
        batch_size=batch_size,
        class_mode="categorical",
        shuffle=False
    )

    test_gen =
test_datagen.flow_from_directory(
        test_dir,
```

```
        target_size=(224, 224),
        batch_size=batch_size,
        class_mode="categorical",
        shuffle=False
    )

    return train_gen, val_gen,
test_gen

print("==== BATCH SIZE 16 ====")
train_gen16, val_gen16, test_gen16 =
create_generators(batch_size=16,
seed=42)
print("==== BATCH SIZE 32 ====")
train_gen32, val_gen32, test_gen32 =
create_generators(batch_size=32,
seed=42)
print("==== BATCH SIZE 64 ====")
train_gen64, val_gen64, test_gen64 =
create_generators(batch_size=64,
seed=42)
num_classes = train_gen16.num_classes
class_names =
list(train_gen16.class_indices.keys()
)
print(f"Jumlah kelas pada dataset:
{num_classes}")
print(f'Nama kelas: {class_names}')

print()
print(f"Jumlah data di training
set\t\t: {train_gen16.n} gambar")
print("Jumlah dataset per kelas di
Training\t:")
print(train_gen16.class_indices)
print(Counter(train_gen16.classes))

print()
print(f"Jumlah data di validation
set\t\t: {val_gen16.n} gambar")
print("Jumlah dataset per kelas di
validation\t:")
print(Counter(val_gen16.classes))

print()
print(f"Jumlah data di testing
set\t\t: {test_gen16.n} gambar")
print("Jumlah dataset per kelas di
testing\t:")
print(Counter(test_gen16.classes))
base_model =
MobileNetV2(input_shape=(224, 224,
3),
                                include
_top=False,
                                weights
='imagenet',
                                )

base_model.trainable = False

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(32, activation='relu')(x)
x = Dropout(0.2)(x)
outputs = layers.Dense(num_classes,
activation="softmax")(x)

MobileNetV2_model = keras.Model(
    inputs=base_model.input,
    outputs=outputs,
    name="MobileNetV2_epoch10_batch16
"
)
MobileNetV2_model.compile(
    optimizer=Adam(learning_rate=0.00
01),
    loss="categorical_crossentropy",
    metrics=["accuracy"]
    )

MobileNetV2_model.summary()
plot_model(
    MobileNetV2_model,
    to_file="/content/drive/MyDrive/h
asil-training/MobileNetV2
Training/MobileNetV2
Architecture.png",
    dpi=100,
    show_shapes=False,
    show_layer_names=False,
    show_trainable=True,
    expand_nested=True,
    show_layer_activations=True
    )
val_gen16.reset()

MobileNetV2_hist =
MobileNetV2_model.fit(
    train_gen16,
    epochs=10,
    validation_data=val_gen16,
    )
plt.figure(figsize=(14, 6))  #
Perbesar ukuran grafik
plt.suptitle("Grafik Pelatihan Model
MobileNetV2\n(Epoch=10, Batch
Size=16)", fontsize=16)

# Subplot 1: Accuracy
plt.subplot(1, 2, 1)
plt.plot(MobileNetV2_hist.history["ac
curacy"], label="Train Accuracy")
plt.plot(MobileNetV2_hist.history["va
l_accuracy"], label="Validation
Accuracy")
plt.xlabel("Epoch", fontsize=12)
plt.ylabel("Accuracy", fontsize=12)
plt.legend(loc="best", fontsize=10)

# Subplot 2: Loss
plt.subplot(1, 2, 2)
plt.plot(MobileNetV2_hist.history["lo
ss"], label="Train Loss")
plt.plot(MobileNetV2_hist.history["va
l_loss"], label="Validation Loss")
plt.xlabel("Epoch", fontsize=12)
plt.ylabel("Loss", fontsize=12)
plt.legend(loc="best", fontsize=10)

plt.savefig("/content/drive/MyDrive/h
asil-training/MobileNetV2
Training/MobileNetV2 (epoch=10,
batch_size=16).png")
plt.show()
print('============= TEST ACC & TEST
LOSS (PREDICT) ============')
test_gen16.reset()
predictions =
MobileNetV2_model.predict(test_gen16,
steps=len(test_gen16));
```

```
true_classes_indices =
test_gen16.classes
predicted_classes_indices =
np.argmax(predictions, axis=1)

class_labels =
list(test_gen16.class_indices.keys())

true_classes_one_hot =
to_categorical(true_classes_indices,
num_classes=num_classes)
test_acc_predict_manual =
accuracy_score(true_classes_indices,
predicted_classes_indices)
cce = CategoricalCrossentropy()
test_loss_predict_manual =
cce(true_classes_one_hot,
predictions).numpy()

print(f"Test Loss (predict_manual)\t:
{test_loss_predict_manual:.4f}")
print(f"Test Accuracy
(predict_manual)\t:
{test_acc_predict_manual:.4f}")

print("\n============= TEST ACC &
TEST LOSS (EVALUATE) ============")
test_gen16.reset()
test_loss, test_acc =
MobileNetV2_model.evaluate(test_gen16
);
print(f"Test Loss\t:
{test_loss:.4f}")
print(f"Test Accuracy\t:
{test_acc:.4f}")
cm =
confusion_matrix(true_classes_indices
, predicted_classes_indices)
print("\n============== HASIL
EVALUASI MODEL ==============\n")

report =
classification_report(true_classes_in
dices, predicted_classes_indices,
target_names=class_labels)
print(report)

accuracy =
accuracy_score(true_classes_indices,
predicted_classes_indices)
print(f"Akurasi Keseluruhan:
{accuracy:.4f} ({accuracy:.2%})")
print("\n============================
=====================")

plt.figure(figsize=(12, 10))
sns.heatmap(cm, annot=True, fmt='d',
cmap='Blues',
            xticklabels=class_labels,
yticklabels=class_labels)

plt.title('Confusion Matrix',
fontsize=16)
plt.ylabel('Kelas Sebenarnya (True
Class)', fontsize=12)
plt.xlabel('Kelas Prediksi (Predicted
Class)', fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.yticks(rotation=0)
plt.tight_layout()
plt.savefig("/content/drive/MyDrive/h
asil-training/MobileNetV2
Training/Confusion Matrix -
MobileNetV2 (epoch=10,
batch_size=16).png")
plt.show()

for i in range(num_classes):
    class_name = class_labels[i]

    TP = cm[i, i]
    FP = cm[:, i].sum() - TP
    FN = cm[i, :].sum() - TP
    TN = np.sum(cm) - (TP + FP + FN)

    print(f"\n--- Kelas: {class_name}
({i}) ---")
    print(f"  True Positives (TP):
{TP}")
    print(f"  False Positives (FP):
{FP}")
    print(f"  False Negatives (FN):
{FN}")
    print(f"  True Negatives (TN):
{TN}")
val_gen16.reset()

# Periksa accuracy dan loss
mobileNetV2 Model
test_loss, test_acc =
MobileNetV2_model.evaluate(val_gen16)
print("Test loss:", test_loss)
print("Test accuracy:", test_acc)

test_images, test_labels =
next(val_gen16)
pred_probs =
MobileNetV2_model.predict(val_gen16)
pred_labels = np.argmax(pred_probs,
axis=1)
true_labels = np.argmax(test_labels,
axis=1)
training_results = {
    "epoch": list(range(1,
len(MobileNetV2_hist.history["loss"])
+ 1)),
    "accuracy":
MobileNetV2_hist.history["accuracy"],
    "val_accuracy":
MobileNetV2_hist.history["val_accurac
y"],
    "loss":
MobileNetV2_hist.history["loss"],
    "val_loss":
MobileNetV2_hist.history["val_loss"],
}

testing_result = {
    "accuracy":
MobileNetV2_hist.history["accuracy"][
-1],
    "val_accuracy":
MobileNetV2_hist.history["val_accurac
y"][-1],
    "loss":
MobileNetV2_hist.history["loss"][-1],
    "val_loss":
MobileNetV2_hist.history["val_loss"][
-1],
    "test_accuracy": test_acc,
    "test_loss": test_loss
```

```
}

# Simpan ke file JSON
with
open("/content/drive/MyDrive/hasil-
training/MobileNetV2
Training/MobileNetV2 (epoch=10,
batch_size=16).json", "w") as f:
    json.dump(training_results, f,
indent=4)

with
open("/content/drive/MyDrive/hasil-
training/MobileNetV2 Training/testing
- MobileNetV2 (epoch=10,
batch_size=16).json", "w") as f:
    json.dump(testing_result, f,
indent=4)

print("Data training berhasil
disimpan ke MobileNetV2 (epoch=10,
batch_size=16).json")
print("Data testing berhasil disimpan
ke testing - MobileNetV2 (epoch=10,
batch_size=16).json")
MobileNetV2_model.save("/content/driv
e/MyDrive/hasil-training/MobileNetV2
Training/MobileNetV2 (epoch=10,
batch_size=16).keras");
MobileNetV2_model =
tf.keras.models.load_model("/content/
drive/MyDrive/hasil-
training/MobileNetV2
Training/MobileNetV2 (epoch=10,
batch_size=16).keras");

converter_keras =
tf.lite.TFLiteConverter.from_keras_mo
del(MobileNetV2_model)

tflite_model =
converter_keras.convert()
with
open('/content/drive/MyDrive/hasil-
training/MobileNetV2
Training/MobileNetV2 (epoch=10,
batch_size=16).tflite', 'wb') as f:
    f.write(tflite_model);
base_model =
MobileNetV2(input_shape=(224, 224,
3),
                                include
_top=False,
                                weights
='imagenet',
                                )

base_model.trainable = False

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(32, activation='relu')(x)
x = Dropout(0.2)(x)
outputs = layers.Dense(num_classes,
activation="softmax")(x)

MobileNetV2_model = keras.Model(
    inputs=base_model.input,
    outputs=outputs,
    name="MobileNetV2_epoch10_batch32
"
)
MobileNetV2_model.compile(
    optimizer=Adam(learning_rate=0.00
01),
    loss="categorical_crossentropy",
    metrics=["accuracy"]
    )

MobileNetV2_model.summary()
val_gen32.reset()

MobileNetV2_hist =
MobileNetV2_model.fit(
    train_gen32,
    epochs=10,
    validation_data=val_gen32,
    )
plt.figure(figsize=(14, 6))  #
Perbesar ukuran grafik
plt.suptitle("Grafik Pelatihan Model
MobileNetV2\n(Epoch=10, Batch
Size=32)", fontsize=16)

# Subplot 1: Accuracy
plt.subplot(1, 2, 1)
plt.plot(MobileNetV2_hist.history["ac
curacy"], label="Train Accuracy")
plt.plot(MobileNetV2_hist.history["va
l_accuracy"], label="Validation
Accuracy")
plt.xlabel("Epoch", fontsize=12)
plt.ylabel("Accuracy", fontsize=12)
plt.legend(loc="best", fontsize=10)

# Subplot 2: Loss
plt.subplot(1, 2, 2)
plt.plot(MobileNetV2_hist.history["lo
ss"], label="Train Loss")
plt.plot(MobileNetV2_hist.history["va
l_loss"], label="Validation Loss")
plt.xlabel("Epoch", fontsize=12)
plt.ylabel("Loss", fontsize=12)
plt.legend(loc="best", fontsize=10)

plt.savefig("/content/drive/MyDrive/h
asil-training/MobileNetV2
Training/MobileNetV2 (epoch=10,
batch_size=32).png")
plt.show()
print('============= TEST ACC & TEST
LOSS (PREDICT) ============')
test_gen32.reset()
predictions =
MobileNetV2_model.predict(test_gen32,
steps=len(test_gen32));

true_classes_indices =
test_gen32.classes
predicted_classes_indices =
np.argmax(predictions, axis=1)

class_labels =
list(test_gen32.class_indices.keys())

true_classes_one_hot =
to_categorical(true_classes_indices,
num_classes=num_classes)
test_acc_predict_manual =
accuracy_score(true_classes_indices,
predicted_classes_indices)
cce = CategoricalCrossentropy()
```

```
test_loss_predict_manual =
cce(true_classes_one_hot,
predictions).numpy()

print(f"Test Loss (predict_manual)\t:
{test_loss_predict_manual:.4f}")
print(f"Test Accuracy
(predict_manual)\t:
{test_acc_predict_manual:.4f}")

print("\n============ TEST ACC &
TEST LOSS (EVALUATE) ============")
test_gen32.reset()
test_loss, test_acc =
MobileNetV2_model.evaluate(test_gen32
);
print(f"Test Loss\t:
{test_loss:.4f}")
print(f"Test Accuracy\t:
{test_acc:.4f}")
cm =
confusion_matrix(true_classes_indices
, predicted_classes_indices)
print("\n============== HASIL
EVALUASI MODEL ==============\n")

report =
classification_report(true_classes_in
dices, predicted_classes_indices,
target_names=class_labels)
print(report)

accuracy =
accuracy_score(true_classes_indices,
predicted_classes_indices)
print(f"Akurasi Keseluruhan:
{accuracy:.4f} ({accuracy:.2%})")
print("\n============================
======================")

plt.figure(figsize=(12, 10))
sns.heatmap(cm, annot=True, fmt='d',
cmap='Blues',
            xticklabels=class_labels,
yticklabels=class_labels)

plt.title('Confusion Matrix',
fontsize=16)
plt.ylabel('Kelas Sebenarnya (True
Class)', fontsize=12)
plt.xlabel('Kelas Prediksi (Predicted
Class)', fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.yticks(rotation=0)
plt.tight_layout()
plt.savefig("/content/drive/MyDrive/h
asil-training/MobileNetV2
Training/Confusion Matrix -
MobileNetV2 (epoch=10,
batch_size=32).png")
plt.show()

for i in range(num_classes):
    class_name = class_labels[i]

    TP = cm[i, i]
    FP = cm[:, i].sum() - TP
    FN = cm[i, :].sum() - TP
    TN = np.sum(cm) - (TP + FP + FN)

    print(f"\n--- Kelas: {class_name}
({i}) ---")
    print(f"  True Positives (TP):
{TP}")
    print(f"  False Positives (FP):
{FP}")
    print(f"  False Negatives (FN):
{FN}")
    print(f"  True Negatives (TN):
{TN}")
val_gen32.reset()

# Periksa accuracy dan loss
mobileNetV2 Model
test_loss, test_acc =
MobileNetV2_model.evaluate(val_gen32)
print("Test loss:", test_loss)
print("Test accuracy:", test_acc)

test_images, test_labels =
next(val_gen32)
pred_probs =
MobileNetV2_model.predict(val_gen32)
pred_labels = np.argmax(pred_probs,
axis=1)
true_labels = np.argmax(test_labels,
axis=1)
training_results = {
    "epoch": list(range(1,
len(MobileNetV2_hist.history["loss"])
+ 1)),
    "accuracy":
MobileNetV2_hist.history["accuracy"],
    "val_accuracy":
MobileNetV2_hist.history["val_accurac
y"],
    "loss":
MobileNetV2_hist.history["loss"],
    "val_loss":
MobileNetV2_hist.history["val_loss"],
}

testing_result = {
    "accuracy":
MobileNetV2_hist.history["accuracy"][
-1],
    "val_accuracy":
MobileNetV2_hist.history["val_accurac
y"][-1],
    "loss":
MobileNetV2_hist.history["loss"][-1],
    "val_loss":
MobileNetV2_hist.history["val_loss"][
-1],
    "test_accuracy": test_acc,
    "test_loss": test_loss
}

# Simpan ke file JSON
with
open("/content/drive/MyDrive/hasil-
training/MobileNetV2
Training/MobileNetV2 (epoch=10,
batch_size=32).json", "w") as f:
    json.dump(training_results, f,
indent=4)

with
open("/content/drive/MyDrive/hasil-
training/MobileNetV2 Training/testing
```

```
- MobileNetV2 (epoch=10,
batch_size=32).json", "w") as f:
    json.dump(testing_result, f,
indent=4)

print("Data training berhasil
disimpan ke MobileNetV2 (epoch=10,
batch_size=32).json")
print("Data testing berhasil disimpan
ke testing - MobileNetV2 (epoch=10,
batch_size=32).json")
MobileNetV2_model.save("/content/driv
e/MyDrive/hasil-training/MobileNetV2
Training/MobileNetV2 (epoch=10,
batch_size=32).keras");
MobileNetV2_model =
tf.keras.models.load_model("/content/
drive/MyDrive/hasil-
training/MobileNetV2
Training/MobileNetV2 (epoch=10,
batch_size=32).keras");

converter_keras =
tf.lite.TFLiteConverter.from_keras_mo
del(MobileNetV2_model)

tflite_model =
converter_keras.convert()
with
open('/content/drive/MyDrive/hasil-
training/MobileNetV2
Training/MobileNetV2 (epoch=10,
batch_size=32).tflite', 'wb') as f:
    f.write(tflite_model);
base_model =
MobileNetV2(input_shape=(224, 224,
3),
                              include
_top=False,
                              weights
='imagenet',
                              )

base_model.trainable = False

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(32, activation='relu')(x)
x = Dropout(0.2)(x)
outputs = layers.Dense(num_classes,
activation="softmax")(x)

MobileNetV2_model = keras.Model(
    inputs=base_model.input,
    outputs=outputs,
    name="MobileNetV2_epoch10_batch64
"
)
MobileNetV2_model.compile(
    optimizer=Adam(learning_rate=0.00
01),
    loss="categorical_crossentropy",
    metrics=["accuracy"]
    )

MobileNetV2_model.summary()
val_gen64.reset()

MobileNetV2_hist =
MobileNetV2_model.fit(
    train_gen64,
    epochs=10,
    validation_data=val_gen64,
    )
plt.figure(figsize=(14, 6))  #
Perbesar ukuran grafik
plt.suptitle("Grafik Pelatihan Model
MobileNetV2\n(Epoch=10, Batch
Size=64)", fontsize=16)

# Subplot 1: Accuracy
plt.subplot(1, 2, 1)
plt.plot(MobileNetV2_hist.history["ac
curacy"], label="Train Accuracy")
plt.plot(MobileNetV2_hist.history["va
l_accuracy"], label="Validation
Accuracy")
plt.xlabel("Epoch", fontsize=12)
plt.ylabel("Accuracy", fontsize=12)
plt.legend(loc="best", fontsize=10)

# Subplot 2: Loss
plt.subplot(1, 2, 2)
plt.plot(MobileNetV2_hist.history["lo
ss"], label="Train Loss")
plt.plot(MobileNetV2_hist.history["va
l_loss"], label="Validation Loss")
plt.xlabel("Epoch", fontsize=12)
plt.ylabel("Loss", fontsize=12)
plt.legend(loc="best", fontsize=10)

plt.savefig("/content/drive/MyDrive/h
asil-training/MobileNetV2
Training/MobileNetV2 (epoch=10,
batch_size=64).png")
plt.show()
print('============= TEST ACC & TEST
LOSS (PREDICT) ============')
test_gen64.reset()
predictions =
MobileNetV2_model.predict(test_gen64,
steps=len(test_gen64));

true_classes_indices =
test_gen64.classes
predicted_classes_indices =
np.argmax(predictions, axis=1)

class_labels =
list(test_gen64.class_indices.keys())

true_classes_one_hot =
to_categorical(true_classes_indices,
num_classes=num_classes)
test_acc_predict_manual =
accuracy_score(true_classes_indices,
predicted_classes_indices)
cce = CategoricalCrossentropy()
test_loss_predict_manual =
cce(true_classes_one_hot,
predictions).numpy()

print(f"Test Loss (predict_manual)\t:
{test_loss_predict_manual:.4f}")
print(f"Test Accuracy
(predict_manual)\t:
{test_acc_predict_manual:.4f}")

print("\n============= TEST ACC &
TEST LOSS (EVALUATE) ============")
test_gen64.reset()
```

```
test_loss, test_acc =
MobileNetV2_model.evaluate(test_gen64
);
print(f"Test Loss\t:
{test_loss:.4f}")
print(f"Test Accuracy\t:
{test_acc:.4f}")
cm =
confusion_matrix(true_classes_indices
, predicted_classes_indices)
print("\n============== HASIL
EVALUASI MODEL ==============\n")

report =
classification_report(true_classes_in
dices, predicted_classes_indices,
target_names=class_labels)
print(report)

accuracy =
accuracy_score(true_classes_indices,
predicted_classes_indices)
print(f"Akurasi Keseluruhan:
{accuracy:.4f} ({accuracy:.2%})")
print("\n============================
======================")

plt.figure(figsize=(12, 10))
sns.heatmap(cm, annot=True, fmt='d',
cmap='Blues',
            xticklabels=class_labels,
yticklabels=class_labels)

plt.title('Confusion Matrix',
fontsize=16)
plt.ylabel('Kelas Sebenarnya (True
Class)', fontsize=12)
plt.xlabel('Kelas Prediksi (Predicted
Class)', fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.yticks(rotation=0)
plt.tight_layout()
plt.savefig("/content/drive/MyDrive/h
asil-training/MobileNetV2
Training/Confusion Matrix -
MobileNetV2 (epoch=10,
batch_size=64).png")
plt.show()

for i in range(num_classes):
    class_name = class_labels[i]

    TP = cm[i, i]
    FP = cm[:, i].sum() - TP
    FN = cm[i, :].sum() - TP
    TN = np.sum(cm) - (TP + FP + FN)

    print(f"\n--- Kelas: {class_name}
({i}) ---")
    print(f"  True Positives (TP):
{TP}")
    print(f"  False Positives (FP):
{FP}")
    print(f"  False Negatives (FN):
{FN}")
    print(f"  True Negatives (TN):
{TN}")
val_gen64.reset()

# Periksa accuracy dan loss
mobileNetV2 Model
test_loss, test_acc =
MobileNetV2_model.evaluate(val_gen64)
print("Test loss:", test_loss)
print("Test accuracy:", test_acc)

test_images, test_labels =
next(val_gen64)
pred_probs =
MobileNetV2_model.predict(val_gen64)
pred_labels = np.argmax(pred_probs,
axis=1)
true_labels = np.argmax(test_labels,
axis=1)
training_results = {
    "epoch": list(range(1,
len(MobileNetV2_hist.history["loss"])
+ 1)),
    "accuracy":
MobileNetV2_hist.history["accuracy"],
    "val_accuracy":
MobileNetV2_hist.history["val_accurac
y"],
    "loss":
MobileNetV2_hist.history["loss"],
    "val_loss":
MobileNetV2_hist.history["val_loss"],
}

testing_result = {
    "accuracy":
MobileNetV2_hist.history["accuracy"][
-1],
    "val_accuracy":
MobileNetV2_hist.history["val_accurac
y"][-1],
    "loss":
MobileNetV2_hist.history["loss"][-1],
    "val_loss":
MobileNetV2_hist.history["val_loss"][
-1],
    "test_accuracy": test_acc,
    "test_loss": test_loss
}

# Simpan ke file JSON
with
open("/content/drive/MyDrive/hasil-
training/MobileNetV2
Training/MobileNetV2 (epoch=10,
batch_size=64).json", "w") as f:
    json.dump(training_results, f,
indent=4)

with
open("/content/drive/MyDrive/hasil-
training/MobileNetV2 Training/testing
- MobileNetV2 (epoch=10,
batch_size=64).json", "w") as f:
    json.dump(testing_result, f,
indent=4)

print("Data training berhasil
disimpan ke MobileNetV2 (epoch=10,
batch_size=64).json")
print("Data testing berhasil disimpan
ke testing - MobileNetV2 (epoch=10,
batch_size=64).json")
MobileNetV2_model.save("/content/driv
e/MyDrive/hasil-training/MobileNetV2
Training/MobileNetV2 (epoch=10,
batch_size=64).keras");
```

```
MobileNetV2_model =
tf.keras.models.load_model("/content/
drive/MyDrive/hasil-
training/MobileNetV2
Training/MobileNetV2 (epoch=10,
batch_size=64).keras");

converter_keras =
tf.lite.TFLiteConverter.from_keras_mo
del(MobileNetV2_model)

tflite_model =
converter_keras.convert()
with
open('/content/drive/MyDrive/hasil-
training/MobileNetV2
Training/MobileNetV2 (epoch=10,
batch_size=64).tflite', 'wb') as f:
    f.write(tflite_model);
base_model =
MobileNetV2(input_shape=(224, 224,
3),
                              include
_top=False,
                              weights
='imagenet',
                              )

base_model.trainable = False

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(32, activation='relu')(x)
x = Dropout(0.2)(x)
outputs = layers.Dense(num_classes,
activation="softmax")(x)

MobileNetV2_model = keras.Model(
    inputs=base_model.input,
    outputs=outputs,
    name="MobileNetV2_epoch30_batch16
"
)
MobileNetV2_model.compile(
    optimizer=Adam(learning_rate=0.00
01),
    loss="categorical_crossentropy",
    metrics=["accuracy"]
    )

MobileNetV2_model.summary()
val_gen16.reset()

MobileNetV2_hist =
MobileNetV2_model.fit(
    train_gen16,
    epochs=30,
    validation_data=val_gen16,
    )
plt.figure(figsize=(14, 6))  #
Perbesar ukuran grafik
plt.suptitle("Grafik Pelatihan Model
MobileNetV2\n(Epoch=30, Batch
Size=16)", fontsize=16)

# Subplot 1: Accuracy
plt.subplot(1, 2, 1)
plt.plot(MobileNetV2_hist.history["ac
curacy"], label="Train Accuracy")
plt.plot(MobileNetV2_hist.history["va
l_accuracy"], label="Validation
Accuracy")
plt.xlabel("Epoch", fontsize=12)
plt.ylabel("Accuracy", fontsize=12)
plt.legend(loc="best", fontsize=10)

# Subplot 2: Loss
plt.subplot(1, 2, 2)
plt.plot(MobileNetV2_hist.history["lo
ss"], label="Train Loss")
plt.plot(MobileNetV2_hist.history["va
l_loss"], label="Validation Loss")
plt.xlabel("Epoch", fontsize=12)
plt.ylabel("Loss", fontsize=12)
plt.legend(loc="best", fontsize=10)

plt.savefig("/content/drive/MyDrive/h
asil-training/MobileNetV2
Training/MobileNetV2 (epoch=30,
batch_size=16).png")
plt.show()
print('============= TEST ACC & TEST
LOSS (PREDICT) ============')
test_gen16.reset()
predictions =
MobileNetV2_model.predict(test_gen16,
steps=len(test_gen16));

true_classes_indices =
test_gen16.classes
predicted_classes_indices =
np.argmax(predictions, axis=1)

class_labels =
list(test_gen16.class_indices.keys())

true_classes_one_hot =
to_categorical(true_classes_indices,
num_classes=num_classes)
test_acc_predict_manual =
accuracy_score(true_classes_indices,
predicted_classes_indices)
cce = CategoricalCrossentropy()
test_loss_predict_manual =
cce(true_classes_one_hot,
predictions).numpy()

print(f"Test Loss (predict_manual)\t:
{test_loss_predict_manual:.4f}")
print(f"Test Accuracy
(predict_manual)\t:
{test_acc_predict_manual:.4f}")

print("\n============= TEST ACC &
TEST LOSS (EVALUATE) ============")
test_gen16.reset()
test_loss, test_acc =
MobileNetV2_model.evaluate(test_gen16
);
print(f"Test Loss\t:
{test_loss:.4f}")
print(f"Test Accuracy\t:
{test_acc:.4f}")
cm =
confusion_matrix(true_classes_indices
, predicted_classes_indices)
print("\n============= HASIL
EVALUASI MODEL ==============\n")
```

```
report =
classification_report(true_classes_in
dices, predicted_classes_indices,
target_names=class_labels)
print(report)

accuracy =
accuracy_score(true_classes_indices,
predicted_classes_indices)
print(f"Akurasi Keseluruhan:
{accuracy:.4f} ({accuracy:.2%})")
print("\n==========================
======================")

plt.figure(figsize=(12, 10))
sns.heatmap(cm, annot=True, fmt='d',
cmap='Blues',
            xticklabels=class_labels,
yticklabels=class_labels)

plt.title('Confusion Matrix',
fontsize=16)
plt.ylabel('Kelas Sebenarnya (True
Class)', fontsize=12)
plt.xlabel('Kelas Prediksi (Predicted
Class)', fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.yticks(rotation=0)
plt.tight_layout()
plt.savefig("/content/drive/MyDrive/h
asil-training/MobileNetV2
Training/Confusion Matrix -
MobileNetV2 (epoch=30,
batch_size=16).png")
plt.show()

for i in range(num_classes):
    class_name = class_labels[i]

    TP = cm[i, i]
    FP = cm[:, i].sum() - TP
    FN = cm[i, :].sum() - TP
    TN = np.sum(cm) - (TP + FP + FN)

    print(f"\n--- Kelas: {class_name}
({i}) ---")
    print(f"  True Positives (TP):
{TP}")
    print(f"  False Positives (FP):
{FP}")
    print(f"  False Negatives (FN):
{FN}")
    print(f"  True Negatives (TN):
{TN}")
val_gen16.reset()

# Periksa accuracy dan loss
mobileNetV2 Model
test_loss, test_acc =
MobileNetV2_model.evaluate(val_gen16)
print("Test loss:", test_loss)
print("Test accuracy:", test_acc)

test_images, test_labels =
next(val_gen16)
pred_probs =
MobileNetV2_model.predict(val_gen16)
pred_labels = np.argmax(pred_probs,
axis=1)
true_labels = np.argmax(test_labels,
axis=1)

training_results = {
    "epoch": list(range(1,
len(MobileNetV2_hist.history["loss"])
+ 1)),
    "accuracy":
MobileNetV2_hist.history["accuracy"],
    "val_accuracy":
MobileNetV2_hist.history["val_accurac
y"],
    "loss":
MobileNetV2_hist.history["loss"],
    "val_loss":
MobileNetV2_hist.history["val_loss"],
}

testing_result = {
    "accuracy":
MobileNetV2_hist.history["accuracy"][
-1],
    "val_accuracy":
MobileNetV2_hist.history["val_accurac
y"][-1],
    "loss":
MobileNetV2_hist.history["loss"][-1],
    "val_loss":
MobileNetV2_hist.history["val_loss"][
-1],
    "test_accuracy": test_acc,
    "test_loss": test_loss
}

# Simpan ke file JSON
with
open("/content/drive/MyDrive/hasil-
training/MobileNetV2
Training/MobileNetV2 (epoch=30,
batch_size=16).json", "w") as f:
    json.dump(training_results, f,
indent=4)

with
open("/content/drive/MyDrive/hasil-
training/MobileNetV2 Training/testing
- MobileNetV2 (epoch=30,
batch_size=16).json", "w") as f:
    json.dump(testing_result, f,
indent=4)

print("Data training berhasil
disimpan ke MobileNetV2 (epoch=30,
batch_size=16).json")
print("Data testing berhasil disimpan
ke testing - MobileNetV2 (epoch=30,
batch_size=16).json")
MobileNetV2_model.save("/content/driv
e/MyDrive/hasil-training/MobileNetV2
Training/MobileNetV2 (epoch=30,
batch_size=16).keras");
MobileNetV2_model =
tf.keras.models.load_model("/content/
drive/MyDrive/hasil-
training/MobileNetV2
Training/MobileNetV2 (epoch=30,
batch_size=16).keras");

converter_keras =
tf.lite.TFLiteConverter.from_keras_mo
del(MobileNetV2_model)

tflite_model =
converter_keras.convert()
```

```
with
open('/content/drive/MyDrive/hasil-
training/MobileNetV2
Training/MobileNetV2 (epoch=30,
batch_size=16).tflite', 'wb') as f:
    f.write(tflite_model);
base_model =
MobileNetV2(input_shape=(224, 224,
3),
                                include
_top=False,
                                weights
='imagenet',
                                )

base_model.trainable = False

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(32, activation='relu')(x)
x = Dropout(0.2)(x)
outputs = layers.Dense(num_classes,
activation="softmax")(x)

MobileNetV2_model = keras.Model(
    inputs=base_model.input,
    outputs=outputs,
    name="MobileNetV2_epoch30_batch32
"
)
MobileNetV2_model.compile(
    optimizer=Adam(learning_rate=0.00
01),
    loss="categorical_crossentropy",
    metrics=["accuracy"]
    )

MobileNetV2_model.summary()
val_gen32.reset()

MobileNetV2_hist =
MobileNetV2_model.fit(
    train_gen32,
    epochs=30,
    validation_data=val_gen32,
    )
plt.figure(figsize=(14, 6))  #
Perbesar ukuran grafik
plt.suptitle("Grafik Pelatihan Model
MobileNetV2\n(Epoch=30, Batch
Size=32)", fontsize=16)

# Subplot 1: Accuracy
plt.subplot(1, 2, 1)
plt.plot(MobileNetV2_hist.history["ac
curacy"], label="Train Accuracy")
plt.plot(MobileNetV2_hist.history["va
l_accuracy"], label="Validation
Accuracy")
plt.xlabel("Epoch", fontsize=12)
plt.ylabel("Accuracy", fontsize=12)
plt.legend(loc="best", fontsize=10)

# Subplot 2: Loss
plt.subplot(1, 2, 2)
plt.plot(MobileNetV2_hist.history["lo
ss"], label="Train Loss")
plt.plot(MobileNetV2_hist.history["va
l_loss"], label="Validation Loss")
plt.xlabel("Epoch", fontsize=12)
plt.ylabel("Loss", fontsize=12)
plt.legend(loc="best", fontsize=10)

plt.savefig("/content/drive/MyDrive/h
asil-training/MobileNetV2
Training/MobileNetV2 (epoch=30,
batch_size=32).png")
plt.show()
print('============= TEST ACC & TEST
LOSS (PREDICT) ============')
test_gen32.reset()
predictions =
MobileNetV2_model.predict(test_gen32,
steps=len(test_gen32));

true_classes_indices =
test_gen32.classes
predicted_classes_indices =
np.argmax(predictions, axis=1)

class_labels =
list(test_gen32.class_indices.keys())

true_classes_one_hot =
to_categorical(true_classes_indices,
num_classes=num_classes)
test_acc_predict_manual =
accuracy_score(true_classes_indices,
predicted_classes_indices)
cce = CategoricalCrossentropy()
test_loss_predict_manual =
cce(true_classes_one_hot,
predictions).numpy()

print(f"Test Loss (predict_manual)\t:
{test_loss_predict_manual:.4f}")
print(f"Test Accuracy
(predict_manual)\t:
{test_acc_predict_manual:.4f}")

print("\n============= TEST ACC &
TEST LOSS (EVALUATE) ============")
test_gen32.reset()
test_loss, test_acc =
MobileNetV2_model.evaluate(test_gen32
);
print(f"Test Loss\t:
{test_loss:.4f}")
print(f"Test Accuracy\t:
{test_acc:.4f}")
cm =
confusion_matrix(true_classes_indices
, predicted_classes_indices)
print("\n============= HASIL
EVALUASI MODEL ==============\n")

report =
classification_report(true_classes_in
dices, predicted_classes_indices,
target_names=class_labels)
print(report)

accuracy =
accuracy_score(true_classes_indices,
predicted_classes_indices)
print(f"Akurasi Keseluruhan:
{accuracy:.4f} ({accuracy:.2%})")
print("\n==========================
======================")

plt.figure(figsize=(12, 10))
```

```
sns.heatmap(cm, annot=True, fmt='d',
cmap='Blues',
            xticklabels=class_labels,
yticklabels=class_labels)

plt.title('Confusion Matrix',
fontsize=16)
plt.ylabel('Kelas Sebenarnya (True
Class)', fontsize=12)
plt.xlabel('Kelas Prediksi (Predicted
Class)', fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.yticks(rotation=0)
plt.tight_layout()
plt.savefig("/content/drive/MyDrive/h
asil-training/MobileNetV2
Training/Confusion Matrix -
MobileNetV2 (epoch=30,
batch_size=32).png")
plt.show()

for i in range(num_classes):
    class_name = class_labels[i]

    TP = cm[i, i]
    FP = cm[:, i].sum() - TP
    FN = cm[i, :].sum() - TP
    TN = np.sum(cm) - (TP + FP + FN)

    print(f"\n--- Kelas: {class_name}
({i}) ---")
    print(f"  True Positives (TP):
{TP}")
    print(f"  False Positives (FP):
{FP}")
    print(f"  False Negatives (FN):
{FN}")
    print(f"  True Negatives (TN):
{TN}")
val_gen32.reset()

# Periksa accuracy dan loss
mobileNetV2 Model
test_loss, test_acc =
MobileNetV2_model.evaluate(val_gen32)
print("Test loss:", test_loss)
print("Test accuracy:", test_acc)

test_images, test_labels =
next(val_gen32)
pred_probs =
MobileNetV2_model.predict(val_gen32)
pred_labels = np.argmax(pred_probs,
axis=1)
true_labels = np.argmax(test_labels,
axis=1)
training_results = {
    "epoch": list(range(1,
len(MobileNetV2_hist.history["loss"])
+ 1)),
    "accuracy":
MobileNetV2_hist.history["accuracy"],
    "val_accuracy":
MobileNetV2_hist.history["val_accurac
y"],
    "loss":
MobileNetV2_hist.history["loss"],
    "val_loss":
MobileNetV2_hist.history["val_loss"],
}

testing_result = {
    "accuracy":
MobileNetV2_hist.history["accuracy"][
-1],
    "val_accuracy":
MobileNetV2_hist.history["val_accurac
y"][-1],
    "loss":
MobileNetV2_hist.history["loss"][-1],
    "val_loss":
MobileNetV2_hist.history["val_loss"][
-1],
    "test_accuracy": test_acc,
    "test_loss": test_loss
}

# Simpan ke file JSON
with
open("/content/drive/MyDrive/hasil-
training/MobileNetV2
Training/MobileNetV2 (epoch=30,
batch_size=32).json", "w") as f:
    json.dump(training_results, f,
indent=4)

with
open("/content/drive/MyDrive/hasil-
training/MobileNetV2 Training/testing
- MobileNetV2 (epoch=30,
batch_size=32).json", "w") as f:
    json.dump(testing_result, f,
indent=4)

print("Data training berhasil
disimpan ke MobileNetV2 (epoch=30,
batch_size=32).json")
print("Data testing berhasil disimpan
ke testing - MobileNetV2 (epoch=30,
batch_size=32).json")
MobileNetV2_model.save("/content/driv
e/MyDrive/hasil-training/MobileNetV2
Training/MobileNetV2 (epoch=30,
batch_size=32).keras");
MobileNetV2_model =
tf.keras.models.load_model("/content/
drive/MyDrive/hasil-
training/MobileNetV2
Training/MobileNetV2 (epoch=30,
batch_size=32).keras");

converter_keras =
tf.lite.TFLiteConverter.from_keras_mo
del(MobileNetV2_model)

tflite_model =
converter_keras.convert()
with
open('/content/drive/MyDrive/hasil-
training/MobileNetV2
Training/MobileNetV2 (epoch=30,
batch_size=32).tflite', 'wb') as f:
    f.write(tflite_model);
base_model =
MobileNetV2(input_shape=(224, 224,
3),
                                include
_top=False,
                                weights
='imagenet',
                                )
```

```
base_model.trainable = False

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(32, activation='relu')(x)
x = Dropout(0.2)(x)
outputs = layers.Dense(num_classes,
activation="softmax")(x)

MobileNetV2_model = keras.Model(
    inputs=base_model.input,
    outputs=outputs,
    name="MobileNetV2_epoch30_batch64
"
)
MobileNetV2_model.compile(
    optimizer=Adam(learning_rate=0.00
01),
    loss="categorical_crossentropy",
    metrics=["accuracy"]
    )

MobileNetV2_model.summary()
val_gen64.reset()

MobileNetV2_hist =
MobileNetV2_model.fit(
    train_gen64,
    epochs=30,
    validation_data=val_gen64,
    )
plt.figure(figsize=(14, 6))  #
Perbesar ukuran grafik
plt.suptitle("Grafik Pelatihan Model
MobileNetV2\n(Epoch=30, Batch
Size=64)", fontsize=16)

# Subplot 1: Accuracy
plt.subplot(1, 2, 1)
plt.plot(MobileNetV2_hist.history["ac
curacy"], label="Train Accuracy")
plt.plot(MobileNetV2_hist.history["va
l_accuracy"], label="Validation
Accuracy")
plt.xlabel("Epoch", fontsize=12)
plt.ylabel("Accuracy", fontsize=12)
plt.legend(loc="best", fontsize=10)

# Subplot 2: Loss
plt.subplot(1, 2, 2)
plt.plot(MobileNetV2_hist.history["lo
ss"], label="Train Loss")
plt.plot(MobileNetV2_hist.history["va
l_loss"], label="Validation Loss")
plt.xlabel("Epoch", fontsize=12)
plt.ylabel("Loss", fontsize=12)
plt.legend(loc="best", fontsize=10)

plt.savefig("/content/drive/MyDrive/h
asil-training/MobileNetV2
Training/MobileNetV2 (epoch=30,
batch_size=64).png")
plt.show()
print('============= TEST ACC & TEST
LOSS (PREDICT) ============')
test_gen64.reset()
predictions =
MobileNetV2_model.predict(test_gen64,
steps=len(test_gen64));

true_classes_indices =
test_gen64.classes
predicted_classes_indices =
np.argmax(predictions, axis=1)

class_labels =
list(test_gen64.class_indices.keys())

true_classes_one_hot =
to_categorical(true_classes_indices,
num_classes=num_classes)
test_acc_predict_manual =
accuracy_score(true_classes_indices,
predicted_classes_indices)
cce = CategoricalCrossentropy()
test_loss_predict_manual =
cce(true_classes_one_hot,
predictions).numpy()

print(f"Test Loss (predict_manual)\t:
{test_loss_predict_manual:.4f}")
print(f"Test Accuracy
(predict_manual)\t:
{test_acc_predict_manual:.4f}")

print("\n============= TEST ACC &
TEST LOSS (EVALUATE) ============")
test_gen64.reset()
test_loss, test_acc =
MobileNetV2_model.evaluate(test_gen64
);
print(f"Test Loss\t:
{test_loss:.4f}")
print(f"Test Accuracy\t:
{test_acc:.4f}")
cm =
confusion_matrix(true_classes_indices
, predicted_classes_indices)
print("\n============= HASIL
EVALUASI MODEL ==============\n")

report =
classification_report(true_classes_in
dices, predicted_classes_indices,
target_names=class_labels)
print(report)

accuracy =
accuracy_score(true_classes_indices,
predicted_classes_indices)
print(f"Akurasi Keseluruhan:
{accuracy:.4f} ({accuracy:.2%})")
print("\n============================
======================")

plt.figure(figsize=(12, 10))
sns.heatmap(cm, annot=True, fmt='d',
cmap='Blues',
            xticklabels=class_labels,
yticklabels=class_labels)

plt.title('Confusion Matrix',
fontsize=16)
plt.ylabel('Kelas Sebenarnya (True
Class)', fontsize=12)
plt.xlabel('Kelas Prediksi (Predicted
Class)', fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.yticks(rotation=0)
plt.tight_layout()
```

```
plt.savefig("/content/drive/MyDrive/hasil-training/MobileNetV2 Training/Confusion Matrix - MobileNetV2 (epoch=30, batch_size=64).png")
plt.show()

for i in range(num_classes):
    class_name = class_labels[i]

    TP = cm[i, i]
    FP = cm[:, i].sum() - TP
    FN = cm[i, :].sum() - TP
    TN = np.sum(cm) - (TP + FP + FN)

    print(f"\n--- Kelas: {class_name} ({i}) ---")
    print(f"  True Positives (TP): {TP}")
    print(f"  False Positives (FP): {FP}")
    print(f"  False Negatives (FN): {FN}")
    print(f"  True Negatives (TN): {TN}")
val_gen64.reset()

# Periksa accuracy dan loss mobileNetV2 Model
test_loss, test_acc = MobileNetV2_model.evaluate(val_gen64)
print("Test loss:", test_loss)
print("Test accuracy:", test_acc)

test_images, test_labels = next(val_gen64)
pred_probs = MobileNetV2_model.predict(val_gen64)
pred_labels = np.argmax(pred_probs, axis=1)
true_labels = np.argmax(test_labels, axis=1)
training_results = {
    "epoch": list(range(1, len(MobileNetV2_hist.history["loss"]) + 1)),
    "accuracy": MobileNetV2_hist.history["accuracy"],
    "val_accuracy": MobileNetV2_hist.history["val_accuracy"],
    "loss": MobileNetV2_hist.history["loss"],
    "val_loss": MobileNetV2_hist.history["val_loss"],
}

testing_result = {
    "accuracy": MobileNetV2_hist.history["accuracy"][-1],
    "val_accuracy": MobileNetV2_hist.history["val_accuracy"][-1],
    "loss": MobileNetV2_hist.history["loss"][-1],
    "val_loss": MobileNetV2_hist.history["val_loss"][-1],
    "test_accuracy": test_acc,
    "test_loss": test_loss
}

# Simpan ke file JSON
with open("/content/drive/MyDrive/hasil-training/MobileNetV2 Training/MobileNetV2 (epoch=30, batch_size=64).json", "w") as f:
    json.dump(training_results, f, indent=4)

with open("/content/drive/MyDrive/hasil-training/MobileNetV2 Training/testing - MobileNetV2 (epoch=30, batch_size=64).json", "w") as f:
    json.dump(testing_result, f, indent=4)

print("Data training berhasil disimpan ke MobileNetV2 (epoch=30, batch_size=64).json")
print("Data testing berhasil disimpan ke testing - MobileNetV2 (epoch=30, batch_size=64).json")
MobileNetV2_model.save("/content/drive/MyDrive/hasil-training/MobileNetV2 Training/MobileNetV2 (epoch=30, batch_size=64).keras");
MobileNetV2_model = tf.keras.models.load_model("/content/drive/MyDrive/hasil-training/MobileNetV2 Training/MobileNetV2 (epoch=30, batch_size=64).keras");

converter_keras = tf.lite.TFLiteConverter.from_keras_model(MobileNetV2_model)

tflite_model = converter_keras.convert()
with open('/content/drive/MyDrive/hasil-training/MobileNetV2 Training/MobileNetV2 (epoch=30, batch_size=64).tflite', 'wb') as f:
    f.write(tflite_model);
base_model = MobileNetV2(input_shape=(224, 224, 3),
                         include_top=False,
                         weights='imagenet',
                         )

base_model.trainable = False

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(32, activation='relu')(x)
x = Dropout(0.2)(x)
outputs = layers.Dense(num_classes, activation="softmax")(x)

MobileNetV2_model = keras.Model(
    inputs=base_model.input,
    outputs=outputs,
    name="MobileNetV2_epoch50_batch16"
```

```
)
MobileNetV2_model.compile(
    optimizer=Adam(learning_rate=0.00
01),
    loss="categorical_crossentropy",
    metrics=["accuracy"]
    )

MobileNetV2_model.summary()
val_gen16.reset()

MobileNetV2_hist =
MobileNetV2_model.fit(
    train_gen16,
    epochs=50,
    validation_data=val_gen16,
    )
plt.figure(figsize=(14, 6))  #
Perbesar ukuran grafik
plt.suptitle("Grafik Pelatihan Model
MobileNetV2\n(Epoch=50, Batch
Size=16)", fontsize=16)

# Subplot 1: Accuracy
plt.subplot(1, 2, 1)
plt.plot(MobileNetV2_hist.history["ac
curacy"], label="Train Accuracy")
plt.plot(MobileNetV2_hist.history["va
l_accuracy"], label="Validation
Accuracy")
plt.xlabel("Epoch", fontsize=12)
plt.ylabel("Accuracy", fontsize=12)
plt.legend(loc="best", fontsize=10)

# Subplot 2: Loss
plt.subplot(1, 2, 2)
plt.plot(MobileNetV2_hist.history["lo
ss"], label="Train Loss")
plt.plot(MobileNetV2_hist.history["va
l_loss"], label="Validation Loss")
plt.xlabel("Epoch", fontsize=12)
plt.ylabel("Loss", fontsize=12)
plt.legend(loc="best", fontsize=10)

plt.savefig("/content/drive/MyDrive/h
asil-training/MobileNetV2
Training/MobileNetV2 (epoch=50,
batch_size=16).png")
plt.show()
print('============= TEST ACC & TEST
LOSS (PREDICT) ============')
test_gen16.reset()
predictions =
MobileNetV2_model.predict(test_gen16,
steps=len(test_gen16));

true_classes_indices =
test_gen16.classes
predicted_classes_indices =
np.argmax(predictions, axis=1)

class_labels =
list(test_gen16.class_indices.keys())

true_classes_one_hot =
to_categorical(true_classes_indices,
num_classes=num_classes)
test_acc_predict_manual =
accuracy_score(true_classes_indices,
predicted_classes_indices)
cce = CategoricalCrossentropy()
test_loss_predict_manual =
cce(true_classes_one_hot,
predictions).numpy()

print(f"Test Loss (predict_manual)\t:
{test_loss_predict_manual:.4f}")
print(f"Test Accuracy
(predict_manual)\t:
{test_acc_predict_manual:.4f}")

print("\n============= TEST ACC &
TEST LOSS (EVALUATE) ============")
test_gen16.reset()
test_loss, test_acc =
MobileNetV2_model.evaluate(test_gen16
);
print(f"Test Loss\t:
{test_loss:.4f}")
print(f"Test Accuracy\t:
{test_acc:.4f}")
cm =
confusion_matrix(true_classes_indices
, predicted_classes_indices)
print("\n============== HASIL
EVALUASI MODEL ==============\n")

report =
classification_report(true_classes_in
dices, predicted_classes_indices,
target_names=class_labels)
print(report)

accuracy =
accuracy_score(true_classes_indices,
predicted_classes_indices)
print(f"Akurasi Keseluruhan:
{accuracy:.4f} ({accuracy:.2%})")
print("\n============================
======================")

plt.figure(figsize=(12, 10))
sns.heatmap(cm, annot=True, fmt='d',
cmap='Blues',
            xticklabels=class_labels,
yticklabels=class_labels)

plt.title('Confusion Matrix',
fontsize=16)
plt.ylabel('Kelas Sebenarnya (True
Class)', fontsize=12)
plt.xlabel('Kelas Prediksi (Predicted
Class)', fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.yticks(rotation=0)
plt.tight_layout()
plt.savefig("/content/drive/MyDrive/h
asil-training/MobileNetV2
Training/Confusion Matrix -
MobileNetV2 (epoch=50,
batch_size=16).png")
plt.show()

for i in range(num_classes):
    class_name = class_labels[i]

    TP = cm[i, i]
    FP = cm[:, i].sum() - TP
    FN = cm[i, :].sum() - TP
    TN = np.sum(cm) - (TP + FP + FN)
```

```
    print(f"\n--- Kelas: {class_name}
({i}) ---")
    print(f"  True Positives (TP):
{TP}")
    print(f"  False Positives (FP):
{FP}")
    print(f"  False Negatives (FN):
{FN}")
    print(f"  True Negatives (TN):
{TN}")
val_gen16.reset()

# Periksa accuracy dan loss
mobileNetV2 Model
test_loss, test_acc =
MobileNetV2_model.evaluate(val_gen16)
print("Test loss:", test_loss)
print("Test accuracy:", test_acc)

test_images, test_labels =
next(val_gen16)
pred_probs =
MobileNetV2_model.predict(val_gen16)
pred_labels = np.argmax(pred_probs,
axis=1)
true_labels = np.argmax(test_labels,
axis=1)
training_results = {
    "epoch": list(range(1,
len(MobileNetV2_hist.history["loss"])
+ 1)),
    "accuracy":
MobileNetV2_hist.history["accuracy"],
    "val_accuracy":
MobileNetV2_hist.history["val_accurac
y"],
    "loss":
MobileNetV2_hist.history["loss"],
    "val_loss":
MobileNetV2_hist.history["val_loss"],
}

testing_result = {
    "accuracy":
MobileNetV2_hist.history["accuracy"][
-1],
    "val_accuracy":
MobileNetV2_hist.history["val_accurac
y"][-1],
    "loss":
MobileNetV2_hist.history["loss"][-1],
    "val_loss":
MobileNetV2_hist.history["val_loss"][
-1],
    "test_accuracy": test_acc,
    "test_loss": test_loss
}

# Simpan ke file JSON
with
open("/content/drive/MyDrive/hasil-
training/MobileNetV2
Training/MobileNetV2 (epoch=50,
batch_size=16).json", "w") as f:
    json.dump(training_results, f,
indent=4)

with
open("/content/drive/MyDrive/hasil-
training/MobileNetV2 Training/testing
- MobileNetV2 (epoch=50,
batch_size=16).json", "w") as f:
    json.dump(testing_result, f,
indent=4)

print("Data training berhasil
disimpan ke MobileNetV2 (epoch=50,
batch_size=16).json")
print("Data testing berhasil disimpan
ke testing - MobileNetV2 (epoch=50,
batch_size=16).json")
MobileNetV2_model.save("/content/driv
e/MyDrive/hasil-training/MobileNetV2
Training/MobileNetV2 (epoch=50,
batch_size=16).keras");
MobileNetV2_model =
tf.keras.models.load_model("/content/
drive/MyDrive/hasil-
training/MobileNetV2
Training/MobileNetV2 (epoch=50,
batch_size=16).keras");

converter_keras =
tf.lite.TFLiteConverter.from_keras_mo
del(MobileNetV2_model)

tflite_model =
converter_keras.convert()
with
open('/content/drive/MyDrive/hasil-
training/MobileNetV2
Training/MobileNetV2 (epoch=50,
batch_size=16).tflite', 'wb') as f:
    f.write(tflite_model);
base_model =
MobileNetV2(input_shape=(224, 224,
3),
                                include
_top=False,
                                weights
='imagenet',
                                )

base_model.trainable = False

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(32, activation='relu')(x)
x = Dropout(0.2)(x)
outputs = layers.Dense(num_classes,
activation="softmax")(x)

MobileNetV2_model = keras.Model(
    inputs=base_model.input,
    outputs=outputs,
    name="MobileNetV2_epoch50_batch32
"
)
MobileNetV2_model.compile(
    optimizer=Adam(learning_rate=0.00
01),
    loss="categorical_crossentropy",
    metrics=["accuracy"]
    )

MobileNetV2_model.summary()
val_gen32.reset()

MobileNetV2_hist =
MobileNetV2_model.fit(
    train_gen32,
```

```
    epochs=50,
    validation_data=val_gen32,
    )
plt.figure(figsize=(14, 6))  # Perbesar ukuran grafik
plt.suptitle("Grafik Pelatihan Model MobileNetV2\n(Epoch=50, Batch Size=32)", fontsize=16)

# Subplot 1: Accuracy
plt.subplot(1, 2, 1)
plt.plot(MobileNetV2_hist.history["accuracy"], label="Train Accuracy")
plt.plot(MobileNetV2_hist.history["val_accuracy"], label="Validation Accuracy")
plt.xlabel("Epoch", fontsize=12)
plt.ylabel("Accuracy", fontsize=12)
plt.legend(loc="best", fontsize=10)

# Subplot 2: Loss
plt.subplot(1, 2, 2)
plt.plot(MobileNetV2_hist.history["loss"], label="Train Loss")
plt.plot(MobileNetV2_hist.history["val_loss"], label="Validation Loss")
plt.xlabel("Epoch", fontsize=12)
plt.ylabel("Loss", fontsize=12)
plt.legend(loc="best", fontsize=10)

plt.savefig("/content/drive/MyDrive/hasil-training/MobileNetV2 Training/MobileNetV2 (epoch=50, batch_size=32).png")
plt.show()
print('============= TEST ACC & TEST LOSS (PREDICT) ============')
test_gen32.reset()
predictions = MobileNetV2_model.predict(test_gen32, steps=len(test_gen32));

true_classes_indices = test_gen32.classes
predicted_classes_indices = np.argmax(predictions, axis=1)

class_labels = list(test_gen32.class_indices.keys())

true_classes_one_hot = to_categorical(true_classes_indices, num_classes=num_classes)
test_acc_predict_manual = accuracy_score(true_classes_indices, predicted_classes_indices)
cce = CategoricalCrossentropy()
test_loss_predict_manual = cce(true_classes_one_hot, predictions).numpy()

print(f"Test Loss (predict_manual)\t: {test_loss_predict_manual:.4f}")
print(f"Test Accuracy (predict_manual)\t: {test_acc_predict_manual:.4f}")

print("\n============= TEST ACC & TEST LOSS (EVALUATE) ============")
test_gen32.reset()
test_loss, test_acc = MobileNetV2_model.evaluate(test_gen32);
print(f"Test Loss\t: {test_loss:.4f}")
print(f"Test Accuracy\t: {test_acc:.4f}")
cm = confusion_matrix(true_classes_indices, predicted_classes_indices)
print("\n============== HASIL EVALUASI MODEL ==============\n")

report = classification_report(true_classes_indices, predicted_classes_indices, target_names=class_labels)
print(report)

accuracy = accuracy_score(true_classes_indices, predicted_classes_indices)
print(f"Akurasi Keseluruhan: {accuracy:.4f} ({accuracy:.2%})")
print("\n============================================================")

plt.figure(figsize=(12, 10))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=class_labels, yticklabels=class_labels)

plt.title('Confusion Matrix', fontsize=16)
plt.ylabel('Kelas Sebenarnya (True Class)', fontsize=12)
plt.xlabel('Kelas Prediksi (Predicted Class)', fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.yticks(rotation=0)
plt.tight_layout()
plt.savefig("/content/drive/MyDrive/hasil-training/MobileNetV2 Training/Confusion Matrix - MobileNetV2 (epoch=50, batch_size=32).png")
plt.show()
for i in range(num_classes):
    class_name = class_labels[i]

    TP = cm[i, i]
    FP = cm[:, i].sum() - TP
    FN = cm[i, :].sum() - TP
    TN = np.sum(cm) - (TP + FP + FN)

    print(f"\n--- Kelas: {class_name} ({i}) ---")
    print(f"  True Positives (TP): {TP}")
    print(f"  False Positives (FP): {FP}")
    print(f"  False Negatives (FN): {FN}")
    print(f"  True Negatives (TN): {TN}")
val_gen32.reset()

# Periksa accuracy dan loss mobileNetV2 Model
```

```
test_loss, test_acc =
MobileNetV2_model.evaluate(val_gen32)
print("Test loss:", test_loss)
print("Test accuracy:", test_acc)

test_images, test_labels =
next(val_gen32)
pred_probs =
MobileNetV2_model.predict(val_gen32)
pred_labels = np.argmax(pred_probs,
axis=1)
true_labels = np.argmax(test_labels,
axis=1)
training_results = {
    "epoch": list(range(1,
len(MobileNetV2_hist.history["loss"])
+ 1)),
    "accuracy":
MobileNetV2_hist.history["accuracy"],
    "val_accuracy":
MobileNetV2_hist.history["val_accurac
y"],
    "loss":
MobileNetV2_hist.history["loss"],
    "val_loss":
MobileNetV2_hist.history["val_loss"],
}

testing_result = {
    "accuracy":
MobileNetV2_hist.history["accuracy"][
-1],
    "val_accuracy":
MobileNetV2_hist.history["val_accurac
y"][-1],
    "loss":
MobileNetV2_hist.history["loss"][-1],
    "val_loss":
MobileNetV2_hist.history["val_loss"][
-1],
    "test_accuracy": test_acc,
    "test_loss": test_loss
}

# Simpan ke file JSON
with
open("/content/drive/MyDrive/hasil-
training/MobileNetV2
Training/MobileNetV2 (epoch=50,
batch_size=32).json", "w") as f:
    json.dump(training_results, f,
indent=4)

with
open("/content/drive/MyDrive/hasil-
training/MobileNetV2 Training/testing
- MobileNetV2 (epoch=50,
batch_size=32).json", "w") as f:
    json.dump(testing_result, f,
indent=4)

print("Data training berhasil
disimpan ke MobileNetV2 (epoch=50,
batch_size=32).json")
print("Data testing berhasil disimpan
ke testing - MobileNetV2 (epoch=50,
batch_size=32).json")
MobileNetV2_model.save("/content/driv
e/MyDrive/hasil-training/MobileNetV2
Training/MobileNetV2 (epoch=50,
batch_size=32).keras");

MobileNetV2_model =
tf.keras.models.load_model("/content/
drive/MyDrive/hasil-
training/MobileNetV2
Training/MobileNetV2 (epoch=50,
batch_size=32).keras");

converter_keras =
tf.lite.TFLiteConverter.from_keras_mo
del(MobileNetV2_model)

tflite_model =
converter_keras.convert()
with
open('/content/drive/MyDrive/hasil-
training/MobileNetV2
Training/MobileNetV2 (epoch=50,
batch_size=32).tflite', 'wb') as f:
    f.write(tflite_model);
base_model =
MobileNetV2(input_shape=(224, 224,
3),
                                include
_top=False,
                                weights
='imagenet',
                                )

base_model.trainable = False

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(32, activation='relu')(x)
x = Dropout(0.2)(x)
outputs = layers.Dense(num_classes,
activation="softmax")(x)

MobileNetV2_model = keras.Model(
    inputs=base_model.input,
    outputs=outputs,
    name="MobileNetV2_epoch50_batch64
"
)
MobileNetV2_model.compile(
    optimizer=Adam(learning_rate=0.00
01),
    loss="categorical_crossentropy",
    metrics=["accuracy"]
    )

MobileNetV2_model.summary()
val_gen64.reset()

MobileNetV2_hist =
MobileNetV2_model.fit(
    train_gen64,
    epochs=50,
    validation_data=val_gen64,
    )
plt.figure(figsize=(14, 6))  #
Perbesar ukuran grafik
plt.suptitle("Grafik Pelatihan Model
MobileNetV2\n(Epoch=50, Batch
Size=64)", fontsize=16)

# Subplot 1: Accuracy
plt.subplot(1, 2, 1)
plt.plot(MobileNetV2_hist.history["ac
curacy"], label="Train Accuracy")
```

```
plt.plot(MobileNetV2_hist.history["val_accuracy"], label="Validation Accuracy")
plt.xlabel("Epoch", fontsize=12)
plt.ylabel("Accuracy", fontsize=12)
plt.legend(loc="best", fontsize=10)

# Subplot 2: Loss
plt.subplot(1, 2, 2)
plt.plot(MobileNetV2_hist.history["loss"], label="Train Loss")
plt.plot(MobileNetV2_hist.history["val_loss"], label="Validation Loss")
plt.xlabel("Epoch", fontsize=12)
plt.ylabel("Loss", fontsize=12)
plt.legend(loc="best", fontsize=10)

plt.savefig("/content/drive/MyDrive/hasil-training/MobileNetV2 Training/MobileNetV2 (epoch=50, batch_size=64).png")
plt.show()
print('============ TEST ACC & TEST LOSS (PREDICT) ============')
test_gen64.reset()
predictions = MobileNetV2_model.predict(test_gen64, steps=len(test_gen64));

true_classes_indices = test_gen64.classes
predicted_classes_indices = np.argmax(predictions, axis=1)

class_labels = list(test_gen64.class_indices.keys())

true_classes_one_hot = to_categorical(true_classes_indices, num_classes=num_classes)
test_acc_predict_manual = accuracy_score(true_classes_indices, predicted_classes_indices)
cce = CategoricalCrossentropy()
test_loss_predict_manual = cce(true_classes_one_hot, predictions).numpy()

print(f"Test Loss (predict_manual)\t: {test_loss_predict_manual:.4f}")
print(f"Test Accuracy (predict_manual)\t: {test_acc_predict_manual:.4f}")

print("\n============ TEST ACC & TEST LOSS (EVALUATE) ============")
test_gen64.reset()
test_loss, test_acc = MobileNetV2_model.evaluate(test_gen64);
print(f"Test Loss\t: {test_loss:.4f}")
print(f"Test Accuracy\t: {test_acc:.4f}")
cm = confusion_matrix(true_classes_indices, predicted_classes_indices)
print("\n============== HASIL EVALUASI MODEL ==============\n")

report = classification_report(true_classes_indices, predicted_classes_indices, target_names=class_labels)
print(report)

accuracy = accuracy_score(true_classes_indices, predicted_classes_indices)
print(f"Akurasi Keseluruhan: {accuracy:.4f} ({accuracy:.2%})")
print("\n============================================================")

plt.figure(figsize=(12, 10))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=class_labels, yticklabels=class_labels)

plt.title('Confusion Matrix', fontsize=16)
plt.ylabel('Kelas Sebenarnya (True Class)', fontsize=12)
plt.xlabel('Kelas Prediksi (Predicted Class)', fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.yticks(rotation=0)
plt.tight_layout()
plt.savefig("/content/drive/MyDrive/hasil-training/MobileNetV2 Training/Confusion Matrix - MobileNetV2 (epoch=50, batch_size=64).png")
plt.show()

for i in range(num_classes):
    class_name = class_labels[i]

    TP = cm[i, i]
    FP = cm[:, i].sum() - TP
    FN = cm[i, :].sum() - TP
    TN = np.sum(cm) - (TP + FP + FN)

    print(f"\n--- Kelas: {class_name} ({i}) ---")
    print(f"  True Positives (TP): {TP}")
    print(f"  False Positives (FP): {FP}")
    print(f"  False Negatives (FN): {FN}")
    print(f"  True Negatives (TN): {TN}")
val_gen64.reset()

# Periksa accuracy dan loss mobileNetV2 Model
test_loss, test_acc = MobileNetV2_model.evaluate(val_gen64)
print("Test loss:", test_loss)
print("Test accuracy:", test_acc)

test_images, test_labels = next(val_gen64)
pred_probs = MobileNetV2_model.predict(val_gen64)
pred_labels = np.argmax(pred_probs, axis=1)
true_labels = np.argmax(test_labels, axis=1)
```

```
training_results = {
    "epoch": list(range(1,
len(MobileNetV2_hist.history["loss"])
+ 1)),
    "accuracy":
MobileNetV2_hist.history["accuracy"],
    "val_accuracy":
MobileNetV2_hist.history["val_accurac
y"],
    "loss":
MobileNetV2_hist.history["loss"],
    "val_loss":
MobileNetV2_hist.history["val_loss"],
}

testing_result = {
    "accuracy":
MobileNetV2_hist.history["accuracy"][
-1],
    "val_accuracy":
MobileNetV2_hist.history["val_accurac
y"][-1],
    "loss":
MobileNetV2_hist.history["loss"][-1],
    "val_loss":
MobileNetV2_hist.history["val_loss"][
-1],
    "test_accuracy": test_acc,
    "test_loss": test_loss
}

# Simpan ke file JSON
with
open("/content/drive/MyDrive/hasil-
training/MobileNetV2
Training/MobileNetV2 (epoch=50,
batch_size=64).json", "w") as f:
    json.dump(training_results, f,
indent=4)

with
open("/content/drive/MyDrive/hasil-
training/MobileNetV2 Training/testing
- MobileNetV2 (epoch=50,
batch_size=64).json", "w") as f:
    json.dump(testing_result, f,
indent=4)

print("Data training berhasil
disimpan ke MobileNetV2 (epoch=50,
batch_size=64).json")
print("Data testing berhasil disimpan
ke testing - MobileNetV2 (epoch=50,
batch_size=64).json")
MobileNetV2_model.save("/content/driv
e/MyDrive/hasil-training/MobileNetV2
Training/MobileNetV2 (epoch=50,
batch_size=64).keras");
MobileNetV2_model =
tf.keras.models.load_model("/content/
drive/MyDrive/hasil-
training/MobileNetV2
Training/MobileNetV2 (epoch=50,
batch_size=64).keras");

converter_keras =
tf.lite.TFLiteConverter.from_keras_mo
del(MobileNetV2_model)

tflite_model =
converter_keras.convert()

with
open('/content/drive/MyDrive/hasil-
training/MobileNetV2
Training/MobileNetV2 (epoch=50,
batch_size=64).tflite', 'wb') as f:
    f.write(tflite_model);
import os
import json
import pandas as pd
import glob
from tabulate import tabulate

# Path menuju direktori file JSON
json_dir =
"/content/drive/MyDrive/hasil-
training/MobileNetV2 Training/"  #
Tetap gunakan direktori yang sama

# Mencari semua file JSON dalam
direktori
json_files =
glob.glob(os.path.join(json_dir,
"*.json"))

# List untuk menyimpan hasil
results = []

# Memproses setiap file JSON
for json_file in json_files:
    # Mengambil nama file untuk
ekstrak parameter
    filename =
os.path.basename(json_file)

    # Ekstrak epoch dan batch_size
dari nama file
    # Asumsi format file:
"MobileNetV2 (epoch=X,
batch_size=Y).json"
    try:
        params =
filename.split("(")[1].split(")")[0]
        epoch_str =
params.split(",")[0].strip()
        batch_size_str =
params.split(",")[1].strip()

        epoch_val =
int(epoch_str.split("=")[1])
        batch_size_val =
int(batch_size_str.split("=")[1])

        # Baca file JSON
        with open(json_file, 'r') as
f:
            data = json.load(f)

        # Ambil nilai pada epoch
terakhir
        last_epoch_idx = -1  # Indeks
untuk epoch terakhir
        accuracy =
data['accuracy'][last_epoch_idx]
        val_accuracy =
data['val_accuracy'][last_epoch_idx]
        loss =
data['loss'][last_epoch_idx]
        val_loss =
data['val_loss'][last_epoch_idx]

        # Tambahkan hasil ke list
```

```
        results.append({
            'Epoch': epoch_val,
            'Batch Size':
batch_size_val,
            'Accuracy': accuracy,
            'Val Accuracy':
val_accuracy,
            'Loss': loss,
            'Val Loss': val_loss
        })
    except Exception as e:
        print(f"Error memproses file
{filename}: {e}")

# Buat DataFrame dan sort berdasarkan
epoch dan batch size
df = pd.DataFrame(results)
df = df.sort_values(by=['Epoch',
'Batch Size'])

# Tampilkan tabel hasil
print("\nHasil Pelatihan MobileNetV2
(Epoch Terakhir):")
print(tabulate(df, headers='keys',
tablefmt='grid', floatfmt='.4f'))

# Jika ingin menyimpan hasil ke CSV
df.to_csv(os.path.join(json_dir,
"mobilenetv2_results_summary.csv"),
index=False)
print(f"\nHasil juga disimpan dalam
file CSV: {os.path.join(json_dir,
'mobilenetv2_results_summary.csv')}")
```

MobileNetV3Small.ipynb

```
from google.colab import drive
drive.mount('/content/drive')
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers,
models #type: ignore
from
tensorflow.keras.preprocessing.image
import ImageDataGenerator #type:
ignore
from tensorflow.keras.applications
import MobileNetV3Small #type: ignore
from
tensorflow.keras.applications.mobilen
et_v3 import preprocess_input as
preprocessor_mobilenetv3 #type:
ignore
from tensorflow.keras.optimizers
import Adam #type: ignore
from tensorflow.keras.callbacks
import ReduceLROnPlateau,
EarlyStopping #type: ignore
from tensorflow.keras.layers import
Dense, GlobalAveragePooling2D,
Dropout, Concatenate, Input, Average
#type: ignore
from tensorflow.keras.models import
Model #type: ignore
from tensorflow.keras.utils import
plot_model #type: ignore
from sklearn.metrics import
confusion_matrix,
classification_report,
precision_score, recall_score,
f1_score, accuracy_score
from tensorflow.keras.losses import
CategoricalCrossentropy
from tensorflow.keras.utils import
to_categorical
from collections import Counter
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os
import json
data_dir =
'/content/drive/MyDrive/image-
dataset/kue/DATASET v3.2.1'
total_images = sum([len(files) for _,
_, files in os.walk(data_dir)])
print(f"Jumlah data di dataset\t\t:
{total_images} gambar")
print('------------------------------
-------------------')

train_dir = data_dir + '/training'
val_dir = data_dir + '/validation'
test_dir = data_dir + '/testing'

def count_files(directory):
    total_files = 0
    for subdir, _, files in
os.walk(directory):
        total_files += len(files)
    return total_files

# Menampilkan jumlah file dalam
setiap direktori
print(f"Jumlah data di training
set\t: {count_files(train_dir)}
gambar ")
print(f"Jumlah data di validation
set\t: {count_files(val_dir)}
gambar")
print(f"Jumlah data di testing set
\t: {count_files(test_dir)} gambar")
def create_generators(batch_size,
seed=42):
    train_datagen =
ImageDataGenerator(
        preprocessing_function=prepro
cessor_mobilenetv3,
        rotation_range=15,
        width_shift_range=0.2,
        height_shift_range=0.2,
        shear_range=0.1,
        zoom_range=0.2,
        horizontal_flip=True,
        brightness_range=(0.9, 1.1),
        fill_mode='nearest',
    )

    val_datagen = ImageDataGenerator(
        preprocessing_function=prepro
cessor_mobilenetv3,
    )

    test_datagen =
ImageDataGenerator(
        preprocessing_function=prepro
cessor_mobilenetv3,
    )

    train_gen =
train_datagen.flow_from_directory(
        train_dir,
```

```
        target_size=(224, 224),
        batch_size=batch_size,
        class_mode="categorical",
        shuffle=True,
        seed=seed
    )

    val_gen =
val_datagen.flow_from_directory(
        val_dir,
        target_size=(224, 224),
        batch_size=batch_size,
        class_mode="categorical",
        shuffle=False
    )

    test_gen =
test_datagen.flow_from_directory(
        test_dir,
        target_size=(224, 224),
        batch_size=batch_size,
        class_mode="categorical",
        shuffle=False
    )

    return train_gen, val_gen,
test_gen

print("==== BATCH SIZE 16 ====")
train_gen16, val_gen16, test_gen16 =
create_generators(batch_size=16,
seed=42)
print("==== BATCH SIZE 32 ====")
train_gen32, val_gen32, test_gen32 =
create_generators(batch_size=32,
seed=42)
print("==== BATCH SIZE 64 ====")
train_gen64, val_gen64, test_gen64 =
create_generators(batch_size=64,
seed=42)
print(f"Train batch size:
{train_gen16.batch_size}")
print(f"Validation batch size:
{val_gen16.batch_size}")
print(f"Testing batch size:
{val_gen16.batch_size}")
num_classes = train_gen16.num_classes
class_names =
list(train_gen16.class_indices.keys()
)
print(f"Jumlah kelas pada dataset:
{num_classes}")
print(f'Nama kelas: {class_names}')

print()
print(f"Jumlah data di training
set\t\t: {train_gen16.n} gambar")
print("Jumlah dataset per kelas di
Training\t:")
print(train_gen16.class_indices)
print(Counter(train_gen16.classes))

print()
print(f"Jumlah data di validation
set\t\t: {val_gen16.n} gambar")
print("Jumlah dataset per kelas di
validation\t:")
print(Counter(val_gen16.classes))

print()
print(f"Jumlah data di testing
set\t\t: {test_gen16.n} gambar")
print("Jumlah dataset per kelas di
testing\t:")
print(Counter(test_gen16.classes))
base_model =
MobileNetV3Small(input_shape=(224,
224, 3),
                                include
_top=False,
                                weights
='imagenet',
                                )

base_model.Trainable = False

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(32, activation='relu')(x)
x = Dropout(0.2)(x)
outputs = layers.Dense(num_classes,
activation="softmax")(x)

MobileNetV3Small_model = keras.Model(
    inputs=base_model.input,
    outputs=outputs,
    name="MobileNetV3Small_epoch10_ba
tch16"
)
MobileNetV3Small_model.compile(
    optimizer=Adam(learning_rate=0.00
01),
    loss="categorical_crossentropy",
    metrics=["accuracy"]
    )

MobileNetV3Small_model.summary()
plot_model(
    MobileNetV3Small_model,
    to_file="/content/drive/MyDrive/h
asil-training/MobileNetV3
Training/MobileNetV3Small
Architecture.png",
    dpi=100,
    show_shapes=False,
    show_layer_names=False,
    show_trainable=True,
    expand_nested=True,
    show_layer_activations=True
    )
val_gen16.reset()

MobileNetV3Small_hist =
MobileNetV3Small_model.fit(
    train_gen16,
    epochs=10,
    validation_data=val_gen16,
    )
plt.figure(figsize=(14, 6))  #
Perbesar ukuran grafik
plt.suptitle("Grafik Pelatihan Model
MobileNetV3Small\n(Epoch=10, Batch
Size=16)", fontsize=16)

# Subplot 1: Accuracy
plt.subplot(1, 2, 1)
plt.plot(MobileNetV3Small_hist.histor
y["accuracy"], label="Train
Accuracy")
```

```
plt.plot(MobileNetV3Small_hist.histor
y["val_accuracy"], label="Validation
Accuracy")
plt.xlabel("Epoch", fontsize=12)
plt.ylabel("Accuracy", fontsize=12)
plt.legend(loc="best", fontsize=10)

# Subplot 2: Loss
plt.subplot(1, 2, 2)
plt.plot(MobileNetV3Small_hist.histor
y["loss"], label="Train Loss")
plt.plot(MobileNetV3Small_hist.histor
y["val_loss"], label="Validation
Loss")
plt.xlabel("Epoch", fontsize=12)
plt.ylabel("Loss", fontsize=12)
plt.legend(loc="best", fontsize=10)

plt.savefig("/content/drive/MyDrive/h
asil-training/MobileNetV3
Training/MobileNetV3Small (epoch=10,
batch_size=16).png")
plt.show()
print('============= TEST ACC & TEST
LOSS (PREDICT) ============')
test_gen16.reset()
predictions =
MobileNetV3Small_model.predict(test_g
en16, steps=len(test_gen16));

true_classes_indices =
test_gen16.classes
predicted_classes_indices =
np.argmax(predictions, axis=1)

class_labels =
list(test_gen16.class_indices.keys())

true_classes_one_hot =
to_categorical(true_classes_indices,
num_classes=num_classes)
test_acc_predict_manual =
accuracy_score(true_classes_indices,
predicted_classes_indices)
cce = CategoricalCrossentropy()
test_loss_predict_manual =
cce(true_classes_one_hot,
predictions).numpy()

print(f"Test Loss (predict_manual)\t:
{test_loss_predict_manual:.4f}")
print(f"Test Accuracy
(predict_manual)\t:
{test_acc_predict_manual:.4f}")

print("\n============= TEST ACC &
TEST LOSS (EVALUATE) ============")
test_gen16.reset()
test_loss, test_acc =
MobileNetV3Small_model.evaluate(test_
gen16);
print(f"Test Loss\t:
{test_loss:.4f}")
print(f"Test Accuracy\t:
{test_acc:.4f}")
cm =
confusion_matrix(true_classes_indices
, predicted_classes_indices)
print("\n============== HASIL
EVALUASI MODEL ==============\n")

report =
classification_report(true_classes_in
dices, predicted_classes_indices,
target_names=class_labels)
print(report)

accuracy =
accuracy_score(true_classes_indices,
predicted_classes_indices)
print(f"Akurasi Keseluruhan:
{accuracy:.4f} ({accuracy:.2%})")
print("\n============================
======================")

plt.figure(figsize=(12, 10))
sns.heatmap(cm, annot=True, fmt='d',
cmap='Blues',
            xticklabels=class_labels,
yticklabels=class_labels)

plt.title('Confusion Matrix',
fontsize=16)
plt.ylabel('Kelas Sebenarnya (True
Class)', fontsize=12)
plt.xlabel('Kelas Prediksi (Predicted
Class)', fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.yticks(rotation=0)
plt.tight_layout()
plt.savefig("/content/drive/MyDrive/h
asil-training/MobileNetV3
Training/Confusion Matrix -
MobileNetV3Small (epoch=10,
batch_size=16).png")
plt.show()

for i in range(num_classes):
    class_name = class_labels[i]

    TP = cm[i, i]
    FP = cm[:, i].sum() - TP
    FN = cm[i, :].sum() - TP
    TN = np.sum(cm) - (TP + FP + FN)

    print(f"\n--- Kelas: {class_name}
({i}) ---")
    print(f"  True Positives (TP):
{TP}")
    print(f"  False Positives (FP):
{FP}")
    print(f"  False Negatives (FN):
{FN}")
    print(f"  True Negatives (TN):
{TN}")
training_results = {
    "epoch": list(range(1,
len(MobileNetV3Small_hist.history["lo
ss"]) + 1)),
    "accuracy":
MobileNetV3Small_hist.history["accura
cy"],
    "val_accuracy":
MobileNetV3Small_hist.history["val_ac
curacy"],
    "loss":
MobileNetV3Small_hist.history["loss"]
,
    "val_loss":
MobileNetV3Small_hist.history["val_lo
ss"],
}
```

```
testing_result = {
    "accuracy":
MobileNetV3Small_hist.history["accura
cy"][-1],
    "val_accuracy":
MobileNetV3Small_hist.history["val_ac
curacy"][-1],
    "loss":
MobileNetV3Small_hist.history["loss"]
[-1],
    "val_loss":
MobileNetV3Small_hist.history["val_lo
ss"][-1],
    "test_accuracy": test_acc,
    "test_loss": test_loss
}

# Simpan ke file JSON
with
open("/content/drive/MyDrive/hasil-
training/MobileNetV3
Training/MobileNetV3Small (epoch=10,
batch_size=16).json", "w") as f:
    json.dump(training_results, f,
indent=4)

with
open("/content/drive/MyDrive/hasil-
training/MobileNetV3 Training/testing
- MobileNetV3Small (epoch=10,
batch_size=16).json", "w") as f:
    json.dump(testing_result, f,
indent=4)

print("Data training berhasil
disimpan ke MobileNetV3Small
(epoch=10, batch_size=16).json")
print("Data testing berhasil disimpan
ke testing - MobileNetV3Small
(epoch=10, batch_size=16).json")
val_gen16.reset()

# Periksa accuracy dan loss
mobileNetV2 Model
test_loss, test_acc =
MobileNetV3Small_model.evaluate(val_g
en16)
print("Test loss:", test_loss)
print("Test accuracy:", test_acc)

test_images, test_labels =
next(val_gen16)
pred_probs =
MobileNetV3Small_model.predict(val_ge
n16)
pred_labels = np.argmax(pred_probs,
axis=1)
true_labels = np.argmax(test_labels,
axis=1)
training_results = {
    "epoch": list(range(1,
len(MobileNetV3Small_hist.history["lo
ss"]) + 1)),
    "accuracy":
MobileNetV3Small_hist.history["accura
cy"],
    "val_accuracy":
MobileNetV3Small_hist.history["val_ac
curacy"],
    "loss":
MobileNetV3Small_hist.history["loss"]
,
    "val_loss":
MobileNetV3Small_hist.history["val_lo
ss"],
}

testing_result = {
    "accuracy":
MobileNetV3Small_hist.history["accura
cy"][-1],
    "val_accuracy":
MobileNetV3Small_hist.history["val_ac
curacy"][-1],
    "loss":
MobileNetV3Small_hist.history["loss"]
[-1],
    "val_loss":
MobileNetV3Small_hist.history["val_lo
ss"][-1],
    "test_accuracy": test_acc,
    "test_loss": test_loss
}

# Simpan ke file JSON
with
open("/content/drive/MyDrive/hasil-
training/MobileNetV3
Training/MobileNetV3Small (epoch=10,
batch_size=16).json", "w") as f:
    json.dump(training_results, f,
indent=4)

with
open("/content/drive/MyDrive/hasil-
training/MobileNetV3 Training/testing
- MobileNetV3Small (epoch=10,
batch_size=16).json", "w") as f:
    json.dump(testing_result, f,
indent=4)

print("Data training berhasil
disimpan ke MobileNetV3Small
(epoch=10, batch_size=16).json")
print("Data testing berhasil disimpan
ke testing - MobileNetV3Small
(epoch=10, batch_size=16).json")
MobileNetV3Small_model.save("/content
/drive/MyDrive/hasil-
training/MobileNetV3
Training/MobileNetV3Small (epoch=10,
batch_size=16).keras");
MobileNetV3Small_model =
tf.keras.models.load_model("/content/
drive/MyDrive/hasil-
training/MobileNetV3
Training/MobileNetV3Small (epoch=10,
batch_size=16).keras");

converter_keras =
tf.lite.TFLiteConverter.from_keras_mo
del(MobileNetV3Small_model)

tflite_model =
converter_keras.convert()
with
open('/content/drive/MyDrive/hasil-
training/MobileNetV3
Training/MobileNetV3Small (epoch=10,
batch_size=16).tflite', 'wb') as f:
```

```
    f.write(tflite_model);
base_model =
MobileNetV3Small(input_shape=(224,
224, 3),
                                include
_top=False,
                                weights
='imagenet',
                                )

base_model.trainable = False

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(32, activation='relu')(x)
x = Dropout(0.2)(x)
outputs = layers.Dense(num_classes,
activation="softmax")(x)

MobileNetV3Small_model = keras.Model(
    inputs=base_model.input,
    outputs=outputs,
    name="MobileNetV3Small_epoch10_ba
tch32"
)
MobileNetV3Small_model.compile(
    optimizer=Adam(learning_rate=0.00
01),
    loss="categorical_crossentropy",
    metrics=["accuracy"]
    )

MobileNetV3Small_model.summary()
val_gen32.reset()

MobileNetV3Small_hist =
MobileNetV3Small_model.fit(
    train_gen32,
    epochs=10,
    validation_data=val_gen32,
    )
plt.figure(figsize=(14, 6))  #
Perbesar ukuran grafik
plt.suptitle("Grafik Pelatihan Model
MobileNetV3Small\n(Epoch=10, Batch
Size=32)", fontsize=16)

# Subplot 1: Accuracy
plt.subplot(1, 2, 1)
plt.plot(MobileNetV3Small_hist.histor
y["accuracy"], label="Train
Accuracy")
plt.plot(MobileNetV3Small_hist.histor
y["val_accuracy"], label="Validation
Accuracy")
plt.xlabel("Epoch", fontsize=12)
plt.ylabel("Accuracy", fontsize=12)
plt.legend(loc="best", fontsize=10)

# Subplot 2: Loss
plt.subplot(1, 2, 2)
plt.plot(MobileNetV3Small_hist.histor
y["loss"], label="Train Loss")
plt.plot(MobileNetV3Small_hist.histor
y["val_loss"], label="Validation
Loss")
plt.xlabel("Epoch", fontsize=12)
plt.ylabel("Loss", fontsize=12)
plt.legend(loc="best", fontsize=10)
plt.savefig("/content/drive/MyDrive/h
asil-training/MobileNetV3
Training/MobileNetV3Small (epoch=10,
batch_size=32).png")
plt.show()
print('============= TEST ACC & TEST
LOSS (PREDICT) ============')
test_gen32.reset()
predictions =
MobileNetV3Small_model.predict(test_g
en32, steps=len(test_gen32));

true_classes_indices =
test_gen32.classes
predicted_classes_indices =
np.argmax(predictions, axis=1)

class_labels =
list(test_gen32.class_indices.keys())

true_classes_one_hot =
to_categorical(true_classes_indices,
num_classes=num_classes)
test_acc_predict_manual =
accuracy_score(true_classes_indices,
predicted_classes_indices)
cce = CategoricalCrossentropy()
test_loss_predict_manual =
cce(true_classes_one_hot,
predictions).numpy()

print(f"Test Loss (predict_manual)\t:
{test_loss_predict_manual:.4f}")
print(f"Test Accuracy
(predict_manual)\t:
{test_acc_predict_manual:.4f}")

print("\n============= TEST ACC &
TEST LOSS (EVALUATE) ============")
test_gen32.reset()
test_loss, test_acc =
MobileNetV3Small_model.evaluate(test_
gen32);
print(f"Test Loss\t:
{test_loss:.4f}")
print(f"Test Accuracy\t:
{test_acc:.4f}")
cm =
confusion_matrix(true_classes_indices
, predicted_classes_indices)
print("\n============== HASIL
EVALUASI MODEL ==============\n")

report =
classification_report(true_classes_in
dices, predicted_classes_indices,
target_names=class_labels)
print(report)

accuracy =
accuracy_score(true_classes_indices,
predicted_classes_indices)
print(f"Akurasi Keseluruhan:
{accuracy:.4f} ({accuracy:.2%})")
print("\n============================
======================")

plt.figure(figsize=(12, 10))
sns.heatmap(cm, annot=True, fmt='d',
cmap='Blues',
```

```
            xticklabels=class_labels,
yticklabels=class_labels)

plt.title('Confusion Matrix',
fontsize=16)
plt.ylabel('Kelas Sebenarnya (True
Class)', fontsize=12)
plt.xlabel('Kelas Prediksi (Predicted
Class)', fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.yticks(rotation=0)
plt.tight_layout()
plt.savefig("/content/drive/MyDrive/h
asil-training/MobileNetV3
Training/Confusion Matrix -
MobileNetV3Small (epoch=10,
batch_size=32).png")
plt.show()

for i in range(num_classes):
    class_name = class_labels[i]

    TP = cm[i, i]
    FP = cm[:, i].sum() - TP
    FN = cm[i, :].sum() - TP
    TN = np.sum(cm) - (TP + FP + FN)

    print(f"\n--- Kelas: {class_name}
({i}) ---")
    print(f"  True Positives (TP):
{TP}")
    print(f"  False Positives (FP):
{FP}")
    print(f"  False Negatives (FN):
{FN}")
    print(f"  True Negatives (TN):
{TN}")
training_results = {
    "epoch": list(range(1,
len(MobileNetV3Small_hist.history["lo
ss"]) + 1)),
    "accuracy":
MobileNetV3Small_hist.history["accura
cy"],
    "val_accuracy":
MobileNetV3Small_hist.history["val_ac
curacy"],
    "loss":
MobileNetV3Small_hist.history["loss"]
,
    "val_loss":
MobileNetV3Small_hist.history["val_lo
ss"],
}

testing_result = {
    "accuracy":
MobileNetV3Small_hist.history["accura
cy"][-1],
    "val_accuracy":
MobileNetV3Small_hist.history["val_ac
curacy"][-1],
    "loss":
MobileNetV3Small_hist.history["loss"]
[-1],
    "val_loss":
MobileNetV3Small_hist.history["val_lo
ss"][-1],
    "test_accuracy": test_acc,
    "test_loss": test_loss
}

# Simpan ke file JSON
with
open("/content/drive/MyDrive/hasil-
training/MobileNetV3
Training/MobileNetV3Small (epoch=10,
batch_size=32).json", "w") as f:
    json.dump(training_results, f,
indent=4)

with
open("/content/drive/MyDrive/hasil-
training/MobileNetV3 Training/testing
- MobileNetV3Small (epoch=10,
batch_size=32).json", "w") as f:
    json.dump(testing_result, f,
indent=4)

print("Data training berhasil
disimpan ke MobileNetV3Small
(epoch=10, batch_size=32).json")
print("Data testing berhasil disimpan
ke testing - MobileNetV3Small
(epoch=10, batch_size=32).json")
val_gen32.reset()

# Periksa accuracy dan loss
mobileNetV2 Model
test_loss, test_acc =
MobileNetV3Small_model.evaluate(val_g
en32)
print("Test loss:", test_loss)
print("Test accuracy:", test_acc)

test_images, test_labels =
next(val_gen32)
pred_probs =
MobileNetV3Small_model.predict(val_ge
n32)
pred_labels = np.argmax(pred_probs,
axis=1)
true_labels = np.argmax(test_labels,
axis=1)
training_results = {
    "epoch": list(range(1,
len(MobileNetV3Small_hist.history["lo
ss"]) + 1)),
    "accuracy":
MobileNetV3Small_hist.history["accura
cy"],
    "val_accuracy":
MobileNetV3Small_hist.history["val_ac
curacy"],
    "loss":
MobileNetV3Small_hist.history["loss"]
,
    "val_loss":
MobileNetV3Small_hist.history["val_lo
ss"],
}

testing_result = {
    "accuracy":
MobileNetV3Small_hist.history["accura
cy"][-1],
    "val_accuracy":
MobileNetV3Small_hist.history["val_ac
curacy"][-1],
    "loss":
MobileNetV3Small_hist.history["loss"]
[-1],
```

```
    "val_loss":
MobileNetV3Small_hist.history["val_lo
ss"][-1],
    "test_accuracy": test_acc,
    "test_loss": test_loss
}

# Simpan ke file JSON
with
open("/content/drive/MyDrive/hasil-
training/MobileNetV3
Training/MobileNetV3Small (epoch=10,
batch_size=32).json", "w") as f:
    json.dump(training_results, f,
indent=4)

with
open("/content/drive/MyDrive/hasil-
training/MobileNetV3 Training/testing
- MobileNetV3Small (epoch=10,
batch_size=32).json", "w") as f:
    json.dump(testing_result, f,
indent=4)

print("Data training berhasil
disimpan ke MobileNetV3Small
(epoch=10, batch_size=32).json")
print("Data testing berhasil disimpan
ke testing - MobileNetV3Small
(epoch=10, batch_size=32).json")
MobileNetV3Small_model.save("/content
/drive/MyDrive/hasil-
training/MobileNetV3
Training/MobileNetV3Small (epoch=10,
batch_size=32).keras");
MobileNetV3Small_model =
tf.keras.models.load_model("/content/
drive/MyDrive/hasil-
training/MobileNetV3
Training/MobileNetV3Small (epoch=10,
batch_size=32).keras");

converter_keras =
tf.lite.TFLiteConverter.from_keras_mo
del(MobileNetV3Small_model)

tflite_model =
converter_keras.convert()
with
open('/content/drive/MyDrive/hasil-
training/MobileNetV3
Training/MobileNetV3Small (epoch=10,
batch_size=32).tflite', 'wb') as f:
    f.write(tflite_model);
base_model =
MobileNetV3Small(input_shape=(224,
224, 3),
                                include
_top=False,
                                weights
='imagenet',
                                )

base_model.trainable = False

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(32, activation='relu')(x)
x = Dropout(0.2)(x)
outputs = layers.Dense(num_classes,
activation="softmax")(x)

MobileNetV3Small_model = keras.Model(
    inputs=base_model.input,
    outputs=outputs,
    name="MobileNetV3Small_epoch10_ba
tch64"
)
MobileNetV3Small_model.compile(
    optimizer=Adam(learning_rate=0.00
01),
    loss="categorical_crossentropy",
    metrics=["accuracy"]
    )

MobileNetV3Small_model.summary()
val_gen64.reset()

MobileNetV3Small_hist =
MobileNetV3Small_model.fit(
    train_gen64,
    batch_size=64,
    epochs=10,
    validation_data=val_gen64,
    )
plt.figure(figsize=(14, 6))  #
Perbesar ukuran grafik
plt.suptitle("Grafik Pelatihan Model
MobileNetV3Small\n(Epoch=10, Batch
Size=64)", fontsize=16)

# Subplot 1: Accuracy
plt.subplot(1, 2, 1)
plt.plot(MobileNetV3Small_hist.histor
y["accuracy"], label="Train
Accuracy")
plt.plot(MobileNetV3Small_hist.histor
y["val_accuracy"], label="Validation
Accuracy")
plt.xlabel("Epoch", fontsize=12)
plt.ylabel("Accuracy", fontsize=12)
plt.legend(loc="best", fontsize=10)

# Subplot 2: Loss
plt.subplot(1, 2, 2)
plt.plot(MobileNetV3Small_hist.histor
y["loss"], label="Train Loss")
plt.plot(MobileNetV3Small_hist.histor
y["val_loss"], label="Validation
Loss")
plt.xlabel("Epoch", fontsize=12)
plt.ylabel("Loss", fontsize=12)
plt.legend(loc="best", fontsize=10)

plt.savefig("/content/drive/MyDrive/h
asil-training/MobileNetV3
Training/MobileNetV3Small (epoch=10,
batch_size=64).png")
plt.show()
print('============== TEST ACC & TEST
LOSS (PREDICT) ============')
test_gen64.reset()
predictions =
MobileNetV3Small_model.predict(test_g
en64, steps=len(test_gen64));

true_classes_indices =
test_gen64.classes
predicted_classes_indices =
np.argmax(predictions, axis=1)
```

```
class_labels =
list(test_gen64.class_indices.keys())

true_classes_one_hot =
to_categorical(true_classes_indices,
num_classes=num_classes)
test_acc_predict_manual =
accuracy_score(true_classes_indices,
predicted_classes_indices)
cce = CategoricalCrossentropy()
test_loss_predict_manual =
cce(true_classes_one_hot,
predictions).numpy()

print(f"Test Loss (predict_manual)\t:
{test_loss_predict_manual:.4f}")
print(f"Test Accuracy
(predict_manual)\t:
{test_acc_predict_manual:.4f}")

print("\n============= TEST ACC &
TEST LOSS (EVALUATE) ============")
test_gen64.reset()
test_loss, test_acc =
MobileNetV3Small_model.evaluate(test_
gen64);
print(f"Test Loss\t:
{test_loss:.4f}")
print(f"Test Accuracy\t:
{test_acc:.4f}")
cm =
confusion_matrix(true_classes_indices
, predicted_classes_indices)
print("\n============== HASIL
EVALUASI MODEL ==============\n")

report =
classification_report(true_classes_in
dices, predicted_classes_indices,
target_names=class_labels)
print(report)

accuracy =
accuracy_score(true_classes_indices,
predicted_classes_indices)
print(f"Akurasi Keseluruhan:
{accuracy:.4f} ({accuracy:.2%})")
print("\n============================
======================")

plt.figure(figsize=(12, 10))
sns.heatmap(cm, annot=True, fmt='d',
cmap='Blues',
            xticklabels=class_labels,
yticklabels=class_labels)

plt.title('Confusion Matrix',
fontsize=16)
plt.ylabel('Kelas Sebenarnya (True
Class)', fontsize=12)
plt.xlabel('Kelas Prediksi (Predicted
Class)', fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.yticks(rotation=0)
plt.tight_layout()
plt.savefig("/content/drive/MyDrive/h
asil-training/MobileNetV3
Training/Confusion Matrix -
MobileNetV3Small (epoch=10,
batch_size=64).png")
plt.show()
```

```
for i in range(num_classes):
    class_name = class_labels[i]

    TP = cm[i, i]
    FP = cm[:, i].sum() - TP
    FN = cm[i, :].sum() - TP
    TN = np.sum(cm) - (TP + FP + FN)

    print(f"\n--- Kelas: {class_name}
({i}) ---")
    print(f"  True Positives (TP):
{TP}")
    print(f"  False Positives (FP):
{FP}")
    print(f"  False Negatives (FN):
{FN}")
    print(f"  True Negatives (TN):
{TN}")
training_results = {
    "epoch": list(range(1,
len(MobileNetV3Small_hist.history["lo
ss"]) + 1)),
    "accuracy":
MobileNetV3Small_hist.history["accura
cy"],
    "val_accuracy":
MobileNetV3Small_hist.history["val_ac
curacy"],
    "loss":
MobileNetV3Small_hist.history["loss"]
,
    "val_loss":
MobileNetV3Small_hist.history["val_lo
ss"],
}

testing_result = {
    "accuracy":
MobileNetV3Small_hist.history["accura
cy"][-1],
    "val_accuracy":
MobileNetV3Small_hist.history["val_ac
curacy"][-1],
    "loss":
MobileNetV3Small_hist.history["loss"]
[-1],
    "val_loss":
MobileNetV3Small_hist.history["val_lo
ss"][-1],
    "test_accuracy": test_acc,
    "test_loss": test_loss
}

# Simpan ke file JSON
with
open("/content/drive/MyDrive/hasil-
training/MobileNetV3
Training/MobileNetV3Small (epoch=10,
batch_size=64).json", "w") as f:
    json.dump(training_results, f,
indent=4)

with
open("/content/drive/MyDrive/hasil-
training/MobileNetV3 Training/testing
- MobileNetV3Small (epoch=10,
batch_size=64).json", "w") as f:
    json.dump(testing_result, f,
indent=4)
```

```
print("Data training berhasil
disimpan ke MobileNetV3Small
(epoch=10, batch_size=64).json")
print("Data testing berhasil disimpan
ke testing - MobileNetV3Small
(epoch=10, batch_size=64).json")
val_gen64.reset()

# Periksa accuracy dan loss
mobileNetV2 Model
test_loss, test_acc =
MobileNetV3Small_model.evaluate(val_g
en64)
print("Test loss:", test_loss)
print("Test accuracy:", test_acc)

test_images, test_labels =
next(val_gen64)
pred_probs =
MobileNetV3Small_model.predict(val_ge
n64)
pred_labels = np.argmax(pred_probs,
axis=1)
true_labels = np.argmax(test_labels,
axis=1)
training_results = {
    "epoch": list(range(1,
len(MobileNetV3Small_hist.history["lo
ss"]) + 1)),
    "accuracy":
MobileNetV3Small_hist.history["accura
cy"],
    "val_accuracy":
MobileNetV3Small_hist.history["val_ac
curacy"],
    "loss":
MobileNetV3Small_hist.history["loss"]
,
    "val_loss":
MobileNetV3Small_hist.history["val_lo
ss"],
}

testing_result = {
    "accuracy":
MobileNetV3Small_hist.history["accura
cy"][-1],
    "val_accuracy":
MobileNetV3Small_hist.history["val_ac
curacy"][-1],
    "loss":
MobileNetV3Small_hist.history["loss"]
[-1],
    "val_loss":
MobileNetV3Small_hist.history["val_lo
ss"][-1],
    "test_accuracy": test_acc,
    "test_loss": test_loss
}

# Simpan ke file JSON
with
open("/content/drive/MyDrive/hasil-
training/MobileNetV3
Training/MobileNetV3Small (epoch=10,
batch_size=64).json", "w") as f:
    json.dump(training_results, f,
indent=4)

with
open("/content/drive/MyDrive/hasil-
training/MobileNetV3 Training/testing
- MobileNetV3Small (epoch=10,
batch_size=64).json", "w") as f:
    json.dump(testing_result, f,
indent=4)

print("Data training berhasil
disimpan ke MobileNetV3Small
(epoch=10, batch_size=64).json")
print("Data testing berhasil disimpan
ke testing - MobileNetV3Small
(epoch=10, batch_size=64).json")
MobileNetV3Small_model.save("/content
/drive/MyDrive/hasil-
training/MobileNetV3
Training/MobileNetV3Small (epoch=10,
batch_size=64).keras");
MobileNetV3Small_model =
tf.keras.models.load_model("/content/
drive/MyDrive/hasil-
training/MobileNetV3
Training/MobileNetV3Small (epoch=10,
batch_size=64).keras");

converter_keras =
tf.lite.TFLiteConverter.from_keras_mo
del(MobileNetV3Small_model)

tflite_model =
converter_keras.convert()
with
open('/content/drive/MyDrive/hasil-
training/MobileNetV3
Training/MobileNetV3Small (epoch=10,
batch_size=64).tflite', 'wb') as f:
    f.write(tflite_model);
base_model =
MobileNetV3Small(input_shape=(224,
224, 3),
                                include
_top=False,
                                weights
='imagenet',
                                )

base_model.trainable = False

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(32, activation='relu')(x)
x = Dropout(0.2)(x)
outputs = layers.Dense(num_classes,
activation="softmax")(x)

MobileNetV3Small_model = keras.Model(
    inputs=base_model.input,
    outputs=outputs,
    name="MobileNetV3Small_epoch30_ba
tch16"
)
MobileNetV3Small_model.compile(
    optimizer=Adam(learning_rate=0.00
01),
    loss="categorical_crossentropy",
    metrics=["accuracy"]
    )

MobileNetV3Small_model.summary()
val_gen16.reset()
```

```
MobileNetV3Small_hist =
MobileNetV3Small_model.fit(
    train_gen16,
    epochs=30,
    validation_data=val_gen16,
    )
plt.figure(figsize=(14, 6))  #
Perbesar ukuran grafik
plt.suptitle("Grafik Pelatihan Model
MobileNetV3Small\n(Epoch=30, Batch
Size=16)", fontsize=16)

# Subplot 1: Accuracy
plt.subplot(1, 2, 1)
plt.plot(MobileNetV3Small_hist.histor
y["accuracy"], label="Train
Accuracy")
plt.plot(MobileNetV3Small_hist.histor
y["val_accuracy"], label="Validation
Accuracy")
plt.xlabel("Epoch", fontsize=12)
plt.ylabel("Accuracy", fontsize=12)
plt.legend(loc="best", fontsize=10)

# Subplot 2: Loss
plt.subplot(1, 2, 2)
plt.plot(MobileNetV3Small_hist.histor
y["loss"], label="Train Loss")
plt.plot(MobileNetV3Small_hist.histor
y["val_loss"], label="Validation
Loss")
plt.xlabel("Epoch", fontsize=12)
plt.ylabel("Loss", fontsize=12)
plt.legend(loc="best", fontsize=10)

plt.savefig("/content/drive/MyDrive/h
asil-training/MobileNetV3
Training/MobileNetV3Small (epoch=30,
batch_size=16).png")
plt.show()
print('============= TEST ACC & TEST
LOSS (PREDICT) ============')
test_gen16.reset()
predictions =
MobileNetV3Small_model.predict(test_g
en16, steps=len(test_gen16));

true_classes_indices =
test_gen16.classes
predicted_classes_indices =
np.argmax(predictions, axis=1)

class_labels =
list(test_gen16.class_indices.keys())

true_classes_one_hot =
to_categorical(true_classes_indices,
num_classes=num_classes)
test_acc_predict_manual =
accuracy_score(true_classes_indices,
predicted_classes_indices)
cce = CategoricalCrossentropy()
test_loss_predict_manual =
cce(true_classes_one_hot,
predictions).numpy()

print(f"Test Loss (predict_manual)\t:
{test_loss_predict_manual:.4f}")
print(f"Test Accuracy
(predict_manual)\t:
{test_acc_predict_manual:.4f}")

print("\n============= TEST ACC &
TEST LOSS (EVALUATE) ============")
test_gen16.reset()
test_loss, test_acc =
MobileNetV3Small_model.evaluate(test_
gen16);
print(f"Test Loss\t:
{test_loss:.4f}")
print(f"Test Accuracy\t:
{test_acc:.4f}")
cm =
confusion_matrix(true_classes_indices
, predicted_classes_indices)
print("\n============== HASIL
EVALUASI MODEL ==============\n")

report =
classification_report(true_classes_in
dices, predicted_classes_indices,
target_names=class_labels)
print(report)

accuracy =
accuracy_score(true_classes_indices,
predicted_classes_indices)
print(f"Akurasi Keseluruhan:
{accuracy:.4f} ({accuracy:.2%})")
print("\n============================
======================")

plt.figure(figsize=(12, 10))
sns.heatmap(cm, annot=True, fmt='d',
cmap='Blues',
            xticklabels=class_labels,
yticklabels=class_labels)

plt.title('Confusion Matrix',
fontsize=16)
plt.ylabel('Kelas Sebenarnya (True
Class)', fontsize=12)
plt.xlabel('Kelas Prediksi (Predicted
Class)', fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.yticks(rotation=0)
plt.tight_layout()
plt.savefig("/content/drive/MyDrive/h
asil-training/MobileNetV3
Training/Confusion Matrix -
MobileNetV3Small (epoch=30,
batch_size=16).png")
plt.show()

for i in range(num_classes):
    class_name = class_labels[i]

    TP = cm[i, i]
    FP = cm[:, i].sum() - TP
    FN = cm[i, :].sum() - TP
    TN = np.sum(cm) - (TP + FP + FN)

    print(f"\n--- Kelas: {class_name}
({i}) ---")
    print(f"  True Positives (TP):
{TP}")
    print(f"  False Positives (FP):
{FP}")
    print(f"  False Negatives (FN):
{FN}")
    print(f"  True Negatives (TN):
{TN}")
```

```
val_gen16.reset()

# Periksa accuracy dan loss
mobileNetV2 Model
test_loss, test_acc =
MobileNetV3Small_model.evaluate(val_g
en16)
print("Test loss:", test_loss)
print("Test accuracy:", test_acc)

test_images, test_labels =
next(val_gen16)
pred_probs =
MobileNetV3Small_model.predict(val_ge
n16)
pred_labels = np.argmax(pred_probs,
axis=1)
true_labels = np.argmax(test_labels,
axis=1)
training_results = {
    "epoch": list(range(1,
len(MobileNetV3Small_hist.history["lo
ss"]) + 1)),
    "accuracy":
MobileNetV3Small_hist.history["accura
cy"],
    "val_accuracy":
MobileNetV3Small_hist.history["val_ac
curacy"],
    "loss":
MobileNetV3Small_hist.history["loss"]
,
    "val_loss":
MobileNetV3Small_hist.history["val_lo
ss"],
}

testing_result = {
    "accuracy":
MobileNetV3Small_hist.history["accura
cy"][-1],
    "val_accuracy":
MobileNetV3Small_hist.history["val_ac
curacy"][-1],
    "loss":
MobileNetV3Small_hist.history["loss"]
[-1],
    "val_loss":
MobileNetV3Small_hist.history["val_lo
ss"][-1],
    "test_accuracy": test_acc,
    "test_loss": test_loss
}

# Simpan ke file JSON
with
open("/content/drive/MyDrive/hasil-
training/MobileNetV3
Training/MobileNetV3Small (epoch=30,
batch_size=16).json", "w") as f:
    json.dump(training_results, f,
indent=4)

with
open("/content/drive/MyDrive/hasil-
training/MobileNetV3 Training/testing
- MobileNetV3Small (epoch=30,
batch_size=16).json", "w") as f:
    json.dump(testing_result, f,
indent=4)

print("Data training berhasil
disimpan ke MobileNetV3Small
(epoch=30, batch_size=16).json")
print("Data testing berhasil disimpan
ke testing - MobileNetV3Small
(epoch=30, batch_size=16).json")
MobileNetV3Small_model.save("/content
/drive/MyDrive/hasil-
training/MobileNetV3
Training/MobileNetV3Small (epoch=30,
batch_size=16).keras");
MobileNetV3Small_model =
tf.keras.models.load_model("/content/
drive/MyDrive/hasil-
training/MobileNetV3
Training/MobileNetV3Small (epoch=30,
batch_size=16).keras");

converter_keras =
tf.lite.TFLiteConverter.from_keras_mo
del(MobileNetV3Small_model)

tflite_model =
converter_keras.convert()
with
open('/content/drive/MyDrive/hasil-
training/MobileNetV3
Training/MobileNetV3Small (epoch=30,
batch_size=16).tflite', 'wb') as f:
    f.write(tflite_model);
base_model =
MobileNetV3Small(input_shape=(224,
224, 3),
                                include
_top=False,
                                weights
='imagenet',
                                )

base_model.trainable = False

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(32, activation='relu')(x)
x = Dropout(0.2)(x)
outputs = layers.Dense(num_classes,
activation="softmax")(x)

MobileNetV3Small_model = keras.Model(
    inputs=base_model.input,
    outputs=outputs,
    name="MobileNetV3Small_epoch30_ba
tch32"
)
MobileNetV3Small_model.compile(
    optimizer=Adam(learning_rate=0.00
01),
    loss="categorical_crossentropy",
    metrics=["accuracy"]
    )

MobileNetV3Small_model.summary()
val_gen32.reset()

MobileNetV3Small_hist =
MobileNetV3Small_model.fit(
    train_gen32,
    epochs=30,
    validation_data=val_gen32,
    )
```

```
plt.figure(figsize=(14, 6))  # Perbesar ukuran grafik
plt.suptitle("Grafik Pelatihan Model MobileNetV3Small\n(Epoch=30, Batch Size=32)", fontsize=16)

# Subplot 1: Accuracy
plt.subplot(1, 2, 1)
plt.plot(MobileNetV3Small_hist.history["accuracy"], label="Train Accuracy")
plt.plot(MobileNetV3Small_hist.history["val_accuracy"], label="Validation Accuracy")
plt.xlabel("Epoch", fontsize=12)
plt.ylabel("Accuracy", fontsize=12)
plt.legend(loc="best", fontsize=10)

# Subplot 2: Loss
plt.subplot(1, 2, 2)
plt.plot(MobileNetV3Small_hist.history["loss"], label="Train Loss")
plt.plot(MobileNetV3Small_hist.history["val_loss"], label="Validation Loss")
plt.xlabel("Epoch", fontsize=12)
plt.ylabel("Loss", fontsize=12)
plt.legend(loc="best", fontsize=10)

plt.savefig("/content/drive/MyDrive/hasil-training/MobileNetV3 Training/MobileNetV3Small (epoch=30, batch_size=32).png")
plt.show()
print('============= TEST ACC & TEST LOSS (PREDICT) ============')
test_gen32.reset()
predictions = MobileNetV3Small_model.predict(test_gen32, steps=len(test_gen32));

true_classes_indices = test_gen32.classes
predicted_classes_indices = np.argmax(predictions, axis=1)

class_labels = list(test_gen32.class_indices.keys())

true_classes_one_hot = to_categorical(true_classes_indices, num_classes=num_classes)
test_acc_predict_manual = accuracy_score(true_classes_indices, predicted_classes_indices)
cce = CategoricalCrossentropy()
test_loss_predict_manual = cce(true_classes_one_hot, predictions).numpy()

print(f"Test Loss (predict_manual)\t: {test_loss_predict_manual:.4f}")
print(f"Test Accuracy (predict_manual)\t: {test_acc_predict_manual:.4f}")

print("\n============= TEST ACC & TEST LOSS (EVALUATE) ============")
test_gen32.reset()

test_loss, test_acc = MobileNetV3Small_model.evaluate(test_gen32);
print(f"Test Loss\t: {test_loss:.4f}")
print(f"Test Accuracy\t: {test_acc:.4f}")
cm = confusion_matrix(true_classes_indices, predicted_classes_indices)
print("\n============== HASIL EVALUASI MODEL ==============\n")

report = classification_report(true_classes_indices, predicted_classes_indices, target_names=class_labels)
print(report)

accuracy = accuracy_score(true_classes_indices, predicted_classes_indices)
print(f"Akurasi Keseluruhan: {accuracy:.4f} ({accuracy:.2%})")
print("\n============================================================")

plt.figure(figsize=(12, 10))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=class_labels, yticklabels=class_labels)

plt.title('Confusion Matrix', fontsize=16)
plt.ylabel('Kelas Sebenarnya (True Class)', fontsize=12)
plt.xlabel('Kelas Prediksi (Predicted Class)', fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.yticks(rotation=0)
plt.tight_layout()
plt.savefig("/content/drive/MyDrive/hasil-training/MobileNetV3 Training/Confusion Matrix - MobileNetV3Small (epoch=30, batch_size=32).png")
plt.show()

for i in range(num_classes):
    class_name = class_labels[i]

    TP = cm[i, i]
    FP = cm[:, i].sum() - TP
    FN = cm[i, :].sum() - TP
    TN = np.sum(cm) - (TP + FP + FN)

    print(f"\n--- Kelas: {class_name} ({i}) ---")
    print(f"  True Positives (TP): {TP}")
    print(f"  False Positives (FP): {FP}")
    print(f"  False Negatives (FN): {FN}")
    print(f"  True Negatives (TN): {TN}")
val_gen32.reset()

# Periksa accuracy dan loss mobileNetV2 Model
```

```
test_loss, test_acc =
MobileNetV3Small_model.evaluate(val_g
en32)
print("Test loss:", test_loss)
print("Test accuracy:", test_acc)

test_images, test_labels =
next(val_gen32)
pred_probs =
MobileNetV3Small_model.predict(val_ge
n32)
pred_labels = np.argmax(pred_probs,
axis=1)
true_labels = np.argmax(test_labels,
axis=1)
training_results = {
    "epoch": list(range(1,
len(MobileNetV3Small_hist.history["lo
ss"]) + 1)),
    "accuracy":
MobileNetV3Small_hist.history["accura
cy"],
    "val_accuracy":
MobileNetV3Small_hist.history["val_ac
curacy"],
    "loss":
MobileNetV3Small_hist.history["loss"]
,
    "val_loss":
MobileNetV3Small_hist.history["val_lo
ss"],
}

testing_result = {
    "accuracy":
MobileNetV3Small_hist.history["accura
cy"][-1],
    "val_accuracy":
MobileNetV3Small_hist.history["val_ac
curacy"][-1],
    "loss":
MobileNetV3Small_hist.history["loss"]
[-1],
    "val_loss":
MobileNetV3Small_hist.history["val_lo
ss"][-1],
    "test_accuracy": test_acc,
    "test_loss": test_loss
}

# Simpan ke file JSON
with
open("/content/drive/MyDrive/hasil-
training/MobileNetV3
Training/MobileNetV3Small (epoch=30,
batch_size=32).json", "w") as f:
    json.dump(training_results, f,
indent=4)

with
open("/content/drive/MyDrive/hasil-
training/MobileNetV3 Training/testing
- MobileNetV3Small (epoch=30,
batch_size=32).json", "w") as f:
    json.dump(testing_result, f,
indent=4)

print("Data training berhasil
disimpan ke MobileNetV3Small
(epoch=30, batch_size=32).json")
print("Data testing berhasil disimpan
ke testing - MobileNetV3Small
(epoch=30, batch_size=32).json")
MobileNetV3Small_model.save("/content
/drive/MyDrive/hasil-
training/MobileNetV3
Training/MobileNetV3Small (epoch=30,
batch_size=32).keras");
MobileNetV3Small_model =
tf.keras.models.load_model("/content/
drive/MyDrive/hasil-
training/MobileNetV3
Training/MobileNetV3Small (epoch=30,
batch_size=32).keras");

converter_keras =
tf.lite.TFLiteConverter.from_keras_mo
del(MobileNetV3Small_model)

tflite_model =
converter_keras.convert()
with
open('/content/drive/MyDrive/hasil-
training/MobileNetV3
Training/MobileNetV3Small (epoch=30,
batch_size=32).tflite', 'wb') as f:
    f.write(tflite_model);
base_model =
MobileNetV3Small(input_shape=(224,
224, 3),
                                include
_top=False,
                                weights
='imagenet',
                                )

base_model.trainable = False

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(32, activation='relu')(x)
x = Dropout(0.2)(x)
outputs = layers.Dense(num_classes,
activation="softmax")(x)

MobileNetV3Small_model = keras.Model(
    inputs=base_model.input,
    outputs=outputs,
    name="MobileNetV3Small_epoch30_ba
tch64"
)
MobileNetV3Small_model.compile(
    optimizer=Adam(learning_rate=0.00
01),
    loss="categorical_crossentropy",
    metrics=["accuracy"]
    )

MobileNetV3Small_model.summary()
val_gen64.reset()

MobileNetV3Small_hist =
MobileNetV3Small_model.fit(
    train_gen64,
    epochs=30,
    validation_data=val_gen64,
    )
plt.figure(figsize=(14, 6))  #
Perbesar ukuran grafik
```

```
plt.suptitle("Grafik Pelatihan Model
MobileNetV3Small\n(Epoch=30, Batch
Size=64)", fontsize=16)

# Subplot 1: Accuracy
plt.subplot(1, 2, 1)
plt.plot(MobileNetV3Small_hist.histor
y["accuracy"], label="Train
Accuracy")
plt.plot(MobileNetV3Small_hist.histor
y["val_accuracy"], label="Validation
Accuracy")
plt.xlabel("Epoch", fontsize=12)
plt.ylabel("Accuracy", fontsize=12)
plt.legend(loc="best", fontsize=10)

# Subplot 2: Loss
plt.subplot(1, 2, 2)
plt.plot(MobileNetV3Small_hist.histor
y["loss"], label="Train Loss")
plt.plot(MobileNetV3Small_hist.histor
y["val_loss"], label="Validation
Loss")
plt.xlabel("Epoch", fontsize=12)
plt.ylabel("Loss", fontsize=12)
plt.legend(loc="best", fontsize=10)

plt.savefig("/content/drive/MyDrive/h
asil-training/MobileNetV3
Training/MobileNetV3Small (epoch=30,
batch_size=64).png")
plt.show()
print('============= TEST ACC & TEST
LOSS (PREDICT) ============')
test_gen64.reset()
predictions =
MobileNetV3Small_model.predict(test_g
en64, steps=len(test_gen64));

true_classes_indices =
test_gen64.classes
predicted_classes_indices =
np.argmax(predictions, axis=1)

class_labels =
list(test_gen64.class_indices.keys())

true_classes_one_hot =
to_categorical(true_classes_indices,
num_classes=num_classes)
test_acc_predict_manual =
accuracy_score(true_classes_indices,
predicted_classes_indices)
cce = CategoricalCrossentropy()
test_loss_predict_manual =
cce(true_classes_one_hot,
predictions).numpy()

print(f"Test Loss (predict_manual)\t:
{test_loss_predict_manual:.4f}")
print(f"Test Accuracy
(predict_manual)\t:
{test_acc_predict_manual:.4f}")

print("\n============= TEST ACC &
TEST LOSS (EVALUATE) ============")
test_gen64.reset()
test_loss, test_acc =
MobileNetV3Small_model.evaluate(test_
gen64);

print(f"Test Loss\t:
{test_loss:.4f}")
print(f"Test Accuracy\t:
{test_acc:.4f}")
cm =
confusion_matrix(true_classes_indices
, predicted_classes_indices)
print("\n============== HASIL
EVALUASI MODEL ==============\n")

report =
classification_report(true_classes_in
dices, predicted_classes_indices,
target_names=class_labels)
print(report)

accuracy =
accuracy_score(true_classes_indices,
predicted_classes_indices)
print(f"Akurasi Keseluruhan:
{accuracy:.4f} ({accuracy:.2%})")
print("\n============================
======================")

plt.figure(figsize=(12, 10))
sns.heatmap(cm, annot=True, fmt='d',
cmap='Blues',
            xticklabels=class_labels,
yticklabels=class_labels)

plt.title('Confusion Matrix',
fontsize=16)
plt.ylabel('Kelas Sebenarnya (True
Class)', fontsize=12)
plt.xlabel('Kelas Prediksi (Predicted
Class)', fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.yticks(rotation=0)
plt.tight_layout()
plt.savefig("/content/drive/MyDrive/h
asil-training/MobileNetV3
Training/Confusion Matrix -
MobileNetV3Small (epoch=30,
batch_size=64).png")
plt.show()

for i in range(num_classes):
    class_name = class_labels[i]

    TP = cm[i, i]
    FP = cm[:, i].sum() - TP
    FN = cm[i, :].sum() - TP
    TN = np.sum(cm) - (TP + FP + FN)

    print(f"\n--- Kelas: {class_name}
({i}) ---")
    print(f"  True Positives (TP):
{TP}")
    print(f"  False Positives (FP):
{FP}")
    print(f"  False Negatives (FN):
{FN}")
    print(f"  True Negatives (TN):
{TN}")
val_gen64.reset()

# Periksa accuracy dan loss
mobileNetV2 Model
test_loss, test_acc =
MobileNetV3Small_model.evaluate(val_g
en64)
```

```
print("Test loss:", test_loss)
print("Test accuracy:", test_acc)

test_images, test_labels =
next(val_gen64)
pred_probs =
MobileNetV3Small_model.predict(val_ge
n64)
pred_labels = np.argmax(pred_probs,
axis=1)
true_labels = np.argmax(test_labels,
axis=1)
training_results = {
    "epoch": list(range(1,
len(MobileNetV3Small_hist.history["lo
ss"]) + 1)),
    "accuracy":
MobileNetV3Small_hist.history["accura
cy"],
    "val_accuracy":
MobileNetV3Small_hist.history["val_ac
curacy"],
    "loss":
MobileNetV3Small_hist.history["loss"]
,
    "val_loss":
MobileNetV3Small_hist.history["val_lo
ss"],
}

testing_result = {
    "accuracy":
MobileNetV3Small_hist.history["accura
cy"][-1],
    "val_accuracy":
MobileNetV3Small_hist.history["val_ac
curacy"][-1],
    "loss":
MobileNetV3Small_hist.history["loss"]
[-1],
    "val_loss":
MobileNetV3Small_hist.history["val_lo
ss"][-1],
    "test_accuracy": test_acc,
    "test_loss": test_loss
}

# Simpan ke file JSON
with
open("/content/drive/MyDrive/hasil-
training/MobileNetV3
Training/MobileNetV3Small (epoch=30,
batch_size=64).json", "w") as f:
    json.dump(training_results, f,
indent=4)

with
open("/content/drive/MyDrive/hasil-
training/MobileNetV3 Training/testing
- MobileNetV3Small (epoch=30,
batch_size=64).json", "w") as f:
    json.dump(testing_result, f,
indent=4)

print("Data training berhasil
disimpan ke MobileNetV3Small
(epoch=30, batch_size=64).json")
print("Data testing berhasil disimpan
ke testing - MobileNetV3Small
(epoch=30, batch_size=64).json")

MobileNetV3Small_model.save("/content
/drive/MyDrive/hasil-
training/MobileNetV3
Training/MobileNetV3Small (epoch=30,
batch_size=64).keras");
MobileNetV3Small_model =
tf.keras.models.load_model("/content/
drive/MyDrive/hasil-
training/MobileNetV3
Training/MobileNetV3Small (epoch=30,
batch_size=64).keras");

converter_keras =
tf.lite.TFLiteConverter.from_keras_mo
del(MobileNetV3Small_model)

tflite_model =
converter_keras.convert()
with
open('/content/drive/MyDrive/hasil-
training/MobileNetV3
Training/MobileNetV3Small (epoch=30,
batch_size=64).tflite', 'wb') as f:
    f.write(tflite_model);
base_model =
MobileNetV3Small(input_shape=(224,
224, 3),
                                include
_top=False,
                                weights
='imagenet',
                                )

base_model.trainable = False

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(32, activation='relu')(x)
x = Dropout(0.2)(x)
outputs = layers.Dense(num_classes,
activation="softmax")(x)

MobileNetV3Small_model = keras.Model(
    inputs=base_model.input,
    outputs=outputs,
    name="MobileNetV3Small_epoch50_ba
tch16"
)
MobileNetV3Small_model.compile(
    optimizer=Adam(learning_rate=0.00
01),
    loss="categorical_crossentropy",
    metrics=["accuracy"]
    )

MobileNetV3Small_model.summary()
val_gen16.reset()

MobileNetV3Small_hist =
MobileNetV3Small_model.fit(
    train_gen16,
    epochs=50,
    validation_data=val_gen16,
    )
plt.figure(figsize=(14, 6))  #
Perbesar ukuran grafik
plt.suptitle("Grafik Pelatihan Model
MobileNetV3Small\n(Epoch=50, Batch
Size=16)", fontsize=16)

# Subplot 1: Accuracy
```

```
plt.subplot(1, 2, 1)
plt.plot(MobileNetV3Small_hist.histor
y["accuracy"], label="Train
Accuracy")
plt.plot(MobileNetV3Small_hist.histor
y["val_accuracy"], label="Validation
Accuracy")
plt.xlabel("Epoch", fontsize=12)
plt.ylabel("Accuracy", fontsize=12)
plt.legend(loc="best", fontsize=10)

# Subplot 2: Loss
plt.subplot(1, 2, 2)
plt.plot(MobileNetV3Small_hist.histor
y["loss"], label="Train Loss")
plt.plot(MobileNetV3Small_hist.histor
y["val_loss"], label="Validation
Loss")
plt.xlabel("Epoch", fontsize=12)
plt.ylabel("Loss", fontsize=12)
plt.legend(loc="best", fontsize=10)

plt.savefig("/content/drive/MyDrive/h
asil-training/MobileNetV3
Training/MobileNetV3Small (epoch=50,
batch_size=16).png")
plt.show()
print('============= TEST ACC & TEST
LOSS (PREDICT) ============')
test_gen16.reset()
predictions =
MobileNetV3Small_model.predict(test_g
en16, steps=len(test_gen16));

true_classes_indices =
test_gen16.classes
predicted_classes_indices =
np.argmax(predictions, axis=1)

class_labels =
list(test_gen16.class_indices.keys())

true_classes_one_hot =
to_categorical(true_classes_indices,
num_classes=num_classes)
test_acc_predict_manual =
accuracy_score(true_classes_indices,
predicted_classes_indices)
cce = CategoricalCrossentropy()
test_loss_predict_manual =
cce(true_classes_one_hot,
predictions).numpy()

print(f"Test Loss (predict_manual)\t:
{test_loss_predict_manual:.4f}")
print(f"Test Accuracy
(predict_manual)\t:
{test_acc_predict_manual:.4f}")

print("\n============= TEST ACC &
TEST LOSS (EVALUATE) ============")
test_gen16.reset()
test_loss, test_acc =
MobileNetV3Small_model.evaluate(test_
gen16);
print(f"Test Loss\t:
{test_loss:.4f}")
print(f"Test Accuracy\t:
{test_acc:.4f}")

cm =
confusion_matrix(true_classes_indices
, predicted_classes_indices)
print("\n============== HASIL
EVALUASI MODEL ==============\n")

report =
classification_report(true_classes_in
dices, predicted_classes_indices,
target_names=class_labels)
print(report)

accuracy =
accuracy_score(true_classes_indices,
predicted_classes_indices)
print(f"Akurasi Keseluruhan:
{accuracy:.4f} ({accuracy:.2%})")
print("\n============================
======================")

plt.figure(figsize=(12, 10))
sns.heatmap(cm, annot=True, fmt='d',
cmap='Blues',
            xticklabels=class_labels,
yticklabels=class_labels)

plt.title('Confusion Matrix',
fontsize=16)
plt.ylabel('Kelas Sebenarnya (True
Class)', fontsize=12)
plt.xlabel('Kelas Prediksi (Predicted
Class)', fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.yticks(rotation=0)
plt.tight_layout()
plt.savefig("/content/drive/MyDrive/h
asil-training/MobileNetV3
Training/Confusion Matrix -
MobileNetV3Small (epoch=50,
batch_size=16).png")
plt.show()

for i in range(num_classes):
    class_name = class_labels[i]

    TP = cm[i, i]
    FP = cm[:, i].sum() - TP
    FN = cm[i, :].sum() - TP
    TN = np.sum(cm) - (TP + FP + FN)

    print(f"\n--- Kelas: {class_name}
({i}) ---")
    print(f"  True Positives (TP):
{TP}")
    print(f"  False Positives (FP):
{FP}")
    print(f"  False Negatives (FN):
{FN}")
    print(f"  True Negatives (TN):
{TN}")
val_gen16.reset()

# Periksa accuracy dan loss
mobileNetV2 Model
test_loss, test_acc =
MobileNetV3Small_model.evaluate(val_g
en16)
print("Test loss:", test_loss)
print("Test accuracy:", test_acc)
```

```
test_images, test_labels =
next(val_gen16)
pred_probs =
MobileNetV3Small_model.predict(val_ge
n16)
pred_labels = np.argmax(pred_probs,
axis=1)
true_labels = np.argmax(test_labels,
axis=1)
training_results = {
    "epoch": list(range(1,
len(MobileNetV3Small_hist.history["lo
ss"]) + 1)),
    "accuracy":
MobileNetV3Small_hist.history["accura
cy"],
    "val_accuracy":
MobileNetV3Small_hist.history["val_ac
curacy"],
    "loss":
MobileNetV3Small_hist.history["loss"]
,
    "val_loss":
MobileNetV3Small_hist.history["val_lo
ss"],
}

testing_result = {
    "accuracy":
MobileNetV3Small_hist.history["accura
cy"][-1],
    "val_accuracy":
MobileNetV3Small_hist.history["val_ac
curacy"][-1],
    "loss":
MobileNetV3Small_hist.history["loss"]
[-1],
    "val_loss":
MobileNetV3Small_hist.history["val_lo
ss"][-1],
    "test_accuracy": test_acc,
    "test_loss": test_loss
}

# Simpan ke file JSON
with
open("/content/drive/MyDrive/hasil-
training/MobileNetV3
Training/MobileNetV3Small (epoch=50,
batch_size=16).json", "w") as f:
    json.dump(training_results, f,
indent=4)

with
open("/content/drive/MyDrive/hasil-
training/MobileNetV3 Training/testing
- MobileNetV3Small (epoch=50,
batch_size=16).json", "w") as f:
    json.dump(testing_result, f,
indent=4)

print("Data training berhasil
disimpan ke MobileNetV3Small
(epoch=50, batch_size=16).json")
print("Data testing berhasil disimpan
ke testing - MobileNetV3Small
(epoch=50, batch_size=16).json")
MobileNetV3Small_model.save("/content
/drive/MyDrive/hasil-
training/MobileNetV3
Training/MobileNetV3Small (epoch=50,
batch_size=16).keras");
MobileNetV3Small_model =
tf.keras.models.load_model("/content/
drive/MyDrive/hasil-
training/MobileNetV3
Training/MobileNetV3Small (epoch=50,
batch_size=16).keras");

converter_keras =
tf.lite.TFLiteConverter.from_keras_mo
del(MobileNetV3Small_model)

tflite_model =
converter_keras.convert()
with
open('/content/drive/MyDrive/hasil-
training/MobileNetV3
Training/MobileNetV3Small (epoch=50,
batch_size=16).tflite', 'wb') as f:
    f.write(tflite_model);
base_model =
MobileNetV3Small(input_shape=(224,
224, 3),
                                include
_top=False,
                                weights
='imagenet',
                                )

base_model.trainable = False

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(32, activation='relu')(x)
x = Dropout(0.2)(x)
outputs = layers.Dense(num_classes,
activation="softmax")(x)

MobileNetV3Small_model = keras.Model(
    inputs=base_model.input,
    outputs=outputs,
    name="MobileNetV3Small_epoch50_ba
tch32"
)
MobileNetV3Small_model.compile(
    optimizer=Adam(learning_rate=0.00
01),
    loss="categorical_crossentropy",
    metrics=["accuracy"]
    )

MobileNetV3Small_model.summary()
val_gen32.reset()

MobileNetV3Small_hist =
MobileNetV3Small_model.fit(
    train_gen32,
    epochs=50,
    validation_data=val_gen32,
    )
plt.figure(figsize=(14, 6))  #
Perbesar ukuran grafik
plt.suptitle("Grafik Pelatihan Model
MobileNetV3Small\n(Epoch=50, Batch
Size=32)", fontsize=16)

# Subplot 1: Accuracy
plt.subplot(1, 2, 1)
```

```
plt.plot(MobileNetV3Small_hist.history["accuracy"], label="Train Accuracy")
plt.plot(MobileNetV3Small_hist.history["val_accuracy"], label="Validation Accuracy")
plt.xlabel("Epoch", fontsize=12)
plt.ylabel("Accuracy", fontsize=12)
plt.legend(loc="best", fontsize=10)

# Subplot 2: Loss
plt.subplot(1, 2, 2)
plt.plot(MobileNetV3Small_hist.history["loss"], label="Train Loss")
plt.plot(MobileNetV3Small_hist.history["val_loss"], label="Validation Loss")
plt.xlabel("Epoch", fontsize=12)
plt.ylabel("Loss", fontsize=12)
plt.legend(loc="best", fontsize=10)

plt.savefig("/content/drive/MyDrive/hasil-training/MobileNetV3 Training/MobileNetV3Small (epoch=50, batch_size=32).png")
plt.show()
print('============= TEST ACC & TEST LOSS (PREDICT) ============')
test_gen32.reset()
predictions = MobileNetV3Small_model.predict(test_gen32, steps=len(test_gen32));

true_classes_indices = test_gen32.classes
predicted_classes_indices = np.argmax(predictions, axis=1)

class_labels = list(test_gen32.class_indices.keys())

true_classes_one_hot = to_categorical(true_classes_indices, num_classes=num_classes)
test_acc_predict_manual = accuracy_score(true_classes_indices, predicted_classes_indices)
cce = CategoricalCrossentropy()
test_loss_predict_manual = cce(true_classes_one_hot, predictions).numpy()

print(f"Test Loss (predict_manual)\t: {test_loss_predict_manual:.4f}")
print(f"Test Accuracy (predict_manual)\t: {test_acc_predict_manual:.4f}")

print("\n============= TEST ACC & TEST LOSS (EVALUATE) ============")
test_gen32.reset()
test_loss, test_acc = MobileNetV3Small_model.evaluate(test_gen32);
print(f"Test Loss\t: {test_loss:.4f}")
print(f"Test Accuracy\t: {test_acc:.4f}")
cm = confusion_matrix(true_classes_indices, predicted_classes_indices)

print("\n============= HASIL EVALUASI MODEL ==============\n")

report = classification_report(true_classes_indices, predicted_classes_indices, target_names=class_labels)
print(report)

accuracy = accuracy_score(true_classes_indices, predicted_classes_indices)
print(f"Akurasi Keseluruhan: {accuracy:.4f} ({accuracy:.2%})")
print("\n============================================================")

plt.figure(figsize=(12, 10))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=class_labels, yticklabels=class_labels)

plt.title('Confusion Matrix', fontsize=16)
plt.ylabel('Kelas Sebenarnya (True Class)', fontsize=12)
plt.xlabel('Kelas Prediksi (Predicted Class)', fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.yticks(rotation=0)
plt.tight_layout()
plt.savefig("/content/drive/MyDrive/hasil-training/MobileNetV3 Training/Confusion Matrix - MobileNetV3Small (epoch=50, batch_size=32).png")
plt.show()

for i in range(num_classes):
    class_name = class_labels[i]

    TP = cm[i, i]
    FP = cm[:, i].sum() - TP
    FN = cm[i, :].sum() - TP
    TN = np.sum(cm) - (TP + FP + FN)

    print(f"\n--- Kelas: {class_name} ({i}) ---")
    print(f"  True Positives (TP): {TP}")
    print(f"  False Positives (FP): {FP}")
    print(f"  False Negatives (FN): {FN}")
    print(f"  True Negatives (TN): {TN}")
val_gen32.reset()

# Periksa accuracy dan loss mobileNetV2 Model
test_loss, test_acc = MobileNetV3Small_model.evaluate(val_gen32)
print("Test loss:", test_loss)
print("Test accuracy:", test_acc)

test_images, test_labels = next(val_gen32)
```

```
pred_probs =
MobileNetV3Small_model.predict(val_ge
n32)
pred_labels = np.argmax(pred_probs,
axis=1)
true_labels = np.argmax(test_labels,
axis=1)
training_results = {
    "epoch": list(range(1,
len(MobileNetV3Small_hist.history["lo
ss"]) + 1)),
    "accuracy":
MobileNetV3Small_hist.history["accura
cy"],
    "val_accuracy":
MobileNetV3Small_hist.history["val_ac
curacy"],
    "loss":
MobileNetV3Small_hist.history["loss"]
,
    "val_loss":
MobileNetV3Small_hist.history["val_lo
ss"],
}

testing_result = {
    "accuracy":
MobileNetV3Small_hist.history["accura
cy"][-1],
    "val_accuracy":
MobileNetV3Small_hist.history["val_ac
curacy"][-1],
    "loss":
MobileNetV3Small_hist.history["loss"]
[-1],
    "val_loss":
MobileNetV3Small_hist.history["val_lo
ss"][-1],
    "test_accuracy": test_acc,
    "test_loss": test_loss
}

# Simpan ke file JSON
with
open("/content/drive/MyDrive/hasil-
training/MobileNetV3
Training/MobileNetV3Small (epoch=50,
batch_size=32).json", "w") as f:
    json.dump(training_results, f,
indent=4)

with
open("/content/drive/MyDrive/hasil-
training/MobileNetV3 Training/testing
- MobileNetV3Small (epoch=50,
batch_size=32).json", "w") as f:
    json.dump(testing_result, f,
indent=4)

print("Data training berhasil
disimpan ke MobileNetV3Small
(epoch=50, batch_size=32).json")
print("Data testing berhasil disimpan
ke testing - MobileNetV3Small
(epoch=50, batch_size=32).json")
MobileNetV3Small_model.save("/content
/drive/MyDrive/hasil-
training/MobileNetV3
Training/MobileNetV3Small (epoch=50,
batch_size=32).keras");

MobileNetV3Small_model =
tf.keras.models.load_model("/content/
drive/MyDrive/hasil-
training/MobileNetV3
Training/MobileNetV3Small (epoch=50,
batch_size=32).keras");

converter_keras =
tf.lite.TFLiteConverter.from_keras_mo
del(MobileNetV3Small_model)

tflite_model =
converter_keras.convert()
with
open('/content/drive/MyDrive/hasil-
training/MobileNetV3
Training/MobileNetV3Small (epoch=50,
batch_size=32).tflite', 'wb') as f:
    f.write(tflite_model);
base_model =
MobileNetV3Small(input_shape=(224,
224, 3),
                                include
_top=False,
                                weights
='imagenet',
                                )

base_model.trainable = False

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(32, activation='relu')(x)
x = Dropout(0.2)(x)
outputs = layers.Dense(num_classes,
activation="softmax")(x)

MobileNetV3Small_model = keras.Model(
    inputs=base_model.input,
    outputs=outputs,
    name="MobileNetV3Small_epoch50_ba
tch64"
)
MobileNetV3Small_model.compile(
    optimizer=Adam(learning_rate=0.00
01),
    loss="categorical_crossentropy",
    metrics=["accuracy"]
    )

MobileNetV3Small_model.summary()
val_gen64.reset()

MobileNetV3Small_hist =
MobileNetV3Small_model.fit(
    train_gen64,
    epochs=50,
    validation_data=val_gen64,
    )
plt.figure(figsize=(14, 6))  #
Perbesar ukuran grafik
plt.suptitle("Grafik Pelatihan Model
MobileNetV3Small\n(Epoch=50, Batch
Size=64)", fontsize=16)

# Subplot 1: Accuracy
plt.subplot(1, 2, 1)
plt.plot(MobileNetV3Small_hist.histor
y["accuracy"], label="Train
Accuracy")
```

```
plt.plot(MobileNetV3Small_hist.histor
y["val_accuracy"], label="Validation
Accuracy")
plt.xlabel("Epoch", fontsize=12)
plt.ylabel("Accuracy", fontsize=12)
plt.legend(loc="best", fontsize=10)

# Subplot 2: Loss
plt.subplot(1, 2, 2)
plt.plot(MobileNetV3Small_hist.histor
y["loss"], label="Train Loss")
plt.plot(MobileNetV3Small_hist.histor
y["val_loss"], label="Validation
Loss")
plt.xlabel("Epoch", fontsize=12)
plt.ylabel("Loss", fontsize=12)
plt.legend(loc="best", fontsize=10)

plt.savefig("/content/drive/MyDrive/h
asil-training/MobileNetV3
Training/MobileNetV3Small (epoch=50,
batch_size=64).png")
plt.show()
print('============= TEST ACC & TEST
LOSS (PREDICT) ============')
test_gen64.reset()
predictions =
MobileNetV3Small_model.predict(test_g
en64, steps=len(test_gen64));

true_classes_indices =
test_gen64.classes
predicted_classes_indices =
np.argmax(predictions, axis=1)

class_labels =
list(test_gen64.class_indices.keys())

true_classes_one_hot =
to_categorical(true_classes_indices,
num_classes=num_classes)
test_acc_predict_manual =
accuracy_score(true_classes_indices,
predicted_classes_indices)
cce = CategoricalCrossentropy()
test_loss_predict_manual =
cce(true_classes_one_hot,
predictions).numpy()

print(f"Test Loss (predict_manual)\t:
{test_loss_predict_manual:.4f}")
print(f"Test Accuracy
(predict_manual)\t:
{test_acc_predict_manual:.4f}")

print("\n============= TEST ACC &
TEST LOSS (EVALUATE) ============")
test_gen64.reset()
test_loss, test_acc =
MobileNetV3Small_model.evaluate(test_
gen64);
print(f"Test Loss\t:
{test_loss:.4f}")
print(f"Test Accuracy\t:
{test_acc:.4f}")
cm =
confusion_matrix(true_classes_indices
, predicted_classes_indices)
print("\n============== HASIL
EVALUASI MODEL ==============\n")

report =
classification_report(true_classes_in
dices, predicted_classes_indices,
target_names=class_labels)
print(report)

accuracy =
accuracy_score(true_classes_indices,
predicted_classes_indices)
print(f"Akurasi Keseluruhan:
{accuracy:.4f} ({accuracy:.2%})")
print("\n============================
======================")

plt.figure(figsize=(12, 10))
sns.heatmap(cm, annot=True, fmt='d',
cmap='Blues',
            xticklabels=class_labels,
yticklabels=class_labels)

plt.title('Confusion Matrix',
fontsize=16)
plt.ylabel('Kelas Sebenarnya (True
Class)', fontsize=12)
plt.xlabel('Kelas Prediksi (Predicted
Class)', fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.yticks(rotation=0)
plt.tight_layout()
plt.savefig("/content/drive/MyDrive/h
asil-training/MobileNetV3
Training/Confusion Matrix -
MobileNetV3Small (epoch=50,
batch_size=64).png")
plt.show()

for i in range(num_classes):
    class_name = class_labels[i]

    TP = cm[i, i]
    FP = cm[:, i].sum() - TP
    FN = cm[i, :].sum() - TP
    TN = np.sum(cm) - (TP + FP + FN)

    print(f"\n--- Kelas: {class_name}
({i}) ---")
    print(f"  True Positives (TP):
{TP}")
    print(f"  False Positives (FP):
{FP}")
    print(f"  False Negatives (FN):
{FN}")
    print(f"  True Negatives (TN):
{TN}")
val_gen64.reset()

# Periksa accuracy dan loss
mobileNetV2 Model
test_loss, test_acc =
MobileNetV3Small_model.evaluate(val_g
en64)
print("Test loss:", test_loss)
print("Test accuracy:", test_acc)

test_images, test_labels =
next(val_gen64)
pred_probs =
MobileNetV3Small_model.predict(val_ge
n64)
pred_labels = np.argmax(pred_probs,
axis=1)
```

```
true_labels = np.argmax(test_labels,
axis=1)
training_results = {
    "epoch": list(range(1,
len(MobileNetV3Small_hist.history["lo
ss"]) + 1)),
    "accuracy":
MobileNetV3Small_hist.history["accura
cy"],
    "val_accuracy":
MobileNetV3Small_hist.history["val_ac
curacy"],
    "loss":
MobileNetV3Small_hist.history["loss"]
,
    "val_loss":
MobileNetV3Small_hist.history["val_lo
ss"],
}

testing_result = {
    "accuracy":
MobileNetV3Small_hist.history["accura
cy"][-1],
    "val_accuracy":
MobileNetV3Small_hist.history["val_ac
curacy"][-1],
    "loss":
MobileNetV3Small_hist.history["loss"]
[-1],
    "val_loss":
MobileNetV3Small_hist.history["val_lo
ss"][-1],
    "test_accuracy": test_acc,
    "test_loss": test_loss
}

# Simpan ke file JSON
with
open("/content/drive/MyDrive/hasil-
training/MobileNetV3
Training/MobileNetV3Small (epoch=50,
batch_size=64).json", "w") as f:
    json.dump(training_results, f,
indent=4)

with
open("/content/drive/MyDrive/hasil-
training/MobileNetV3 Training/testing
- MobileNetV3Small (epoch=50,
batch_size=64).json", "w") as f:
    json.dump(testing_result, f,
indent=4)

print("Data training berhasil
disimpan ke MobileNetV3Small
(epoch=50, batch_size=64).json")
print("Data testing berhasil disimpan
ke testing - MobileNetV3Small
(epoch=50, batch_size=64).json")
MobileNetV3Small_model.save("/content
/drive/MyDrive/hasil-
training/MobileNetV3
Training/MobileNetV3Small (epoch=50,
batch_size=64).keras");
MobileNetV3Small_model =
tf.keras.models.load_model("/content/
drive/MyDrive/hasil-
training/MobileNetV3
Training/MobileNetV3Small (epoch=50,
batch_size=64).keras");

converter_keras =
tf.lite.TFLiteConverter.from_keras_mo
del(MobileNetV3Small_model)

tflite_model =
converter_keras.convert()
with
open('/content/drive/MyDrive/hasil-
training/MobileNetV3
Training/MobileNetV3Small (epoch=50,
batch_size=64).tflite', 'wb') as f:
    f.write(tflite_model);
import os
import json
import pandas as pd
import glob
from tabulate import tabulate

# Path menuju direktori file JSON
json_dir =
"/content/drive/MyDrive/hasil-
training/MobileNetV3 Training/"  #
Tetap gunakan direktori yang sama

# Mencari semua file JSON dalam
direktori
json_files =
glob.glob(os.path.join(json_dir,
"*.json"))

# List untuk menyimpan hasil
results = []

# Memproses setiap file JSON
for json_file in json_files:
    # Mengambil nama file untuk
ekstrak parameter
    filename =
os.path.basename(json_file)

    # Ekstrak epoch dan batch_size
dari nama file
    # Asumsi format file:
"MobileNetV3Small (epoch=X,
batch_size=Y).json"
    try:
        params =
filename.split("(")[1].split(")")[0]
        epoch_str =
params.split(",")[0].strip()
        batch_size_str =
params.split(",")[1].strip()

        epoch_val =
int(epoch_str.split("=")[1])
        batch_size_val =
int(batch_size_str.split("=")[1])

        # Baca file JSON
        with open(json_file, 'r') as
f:
            data = json.load(f)

        # Ambil nilai pada epoch
terakhir
        last_epoch_idx = -1  # Indeks
untuk epoch terakhir
        accuracy =
data['accuracy'][last_epoch_idx]
```

```
        val_accuracy =
data['val_accuracy'][last_epoch_idx]
        loss =
data['loss'][last_epoch_idx]
        val_loss =
data['val_loss'][last_epoch_idx]

        # Tambahkan hasil ke list
        results.append({
            'Epoch': epoch_val,
            'Batch Size':
batch_size_val,
            'Accuracy': accuracy,
            'Val Accuracy':
val_accuracy,
            'Loss': loss,
            'Val Loss': val_loss
        })
    except Exception as e:
        print(f"Error memproses file
{filename}: {e}")

# Buat DataFrame dan sort berdasarkan
epoch dan batch size
df = pd.DataFrame(results)
df = df.sort_values(by=['Epoch',
'Batch Size'])

# Tampilkan tabel hasil
print("\nHasil Pelatihan
MobileNetV3Small (Epoch Terakhir):")
print(tabulate(df, headers='keys',
tablefmt='grid', floatfmt='.4f'))

# Jika ingin menyimpan hasil ke CSV
df.to_csv(os.path.join(json_dir,
"mobilenetv3small_results_summary.csv
"), index=False)
print(f"\nHasil juga disimpan dalam
file CSV: {os.path.join(json_dir,
'mobilenetv3small_results_summary.csv
')}")
```

EfficientNetB0.ipynb

```
from google.colab import drive
drive.mount('/content/drive')
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers,
models #type: ignore
from
tensorflow.keras.preprocessing.image
import ImageDataGenerator #type:
ignore
from tensorflow.keras.applications
import EfficientNetB0 #type: ignore
from
tensorflow.keras.applications.efficie
ntnet import preprocess_input as
preprocessor_efficientnet #type:
ignore
from tensorflow.keras.optimizers
import Adam #type: ignore
from tensorflow.keras.callbacks
import ReduceLROnPlateau,
EarlyStopping #type: ignore
from tensorflow.keras.layers import
Dense, GlobalAveragePooling2D,
Dropout, Concatenate, Input, Average
#type: ignore
from tensorflow.keras.models import
Model #type: ignore
from tensorflow.keras.utils import
plot_model #type: ignore
from sklearn.metrics import
confusion_matrix,
classification_report,
precision_score, recall_score,
f1_score, accuracy_score
from collections import Counter
from tensorflow.keras.losses import
CategoricalCrossentropy #type: ignore
from tensorflow.keras.utils import
to_categorical #type: ignore
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os
import json
data_dir =
'/content/drive/MyDrive/image-
dataset/kue/DATASET v3.2.1'
total_images = sum([len(files) for _,
_, files in os.walk(data_dir)])
print(f"Jumlah data di dataset\t\t:
{total_images} gambar")
print('------------------------------
-------------------')

train_dir = data_dir + '/training'
val_dir = data_dir + '/validation'
test_dir = data_dir + '/testing'

def count_files(directory):
    total_files = 0
    for subdir, _, files in
os.walk(directory):
        total_files += len(files)
    return total_files

# Menampilkan jumlah file dalam
setiap direktori
print(f"Jumlah data di training
set\t: {count_files(train_dir)}
gambar ")
print(f"Jumlah data di validation
set\t: {count_files(val_dir)}
gambar")
print(f"Jumlah data di testing set
\t: {count_files(test_dir)} gambar")
def create_generators(batch_size,
seed=42):
    train_datagen =
ImageDataGenerator(
        preprocessing_function=prepro
cessor_efficientnet,
        rotation_range=15,
        width_shift_range=0.2,
        height_shift_range=0.2,
        shear_range=0.1,
        zoom_range=0.2,
        horizontal_flip=True,
        brightness_range=(0.9, 1.1),
        fill_mode='nearest',
    )

    val_datagen = ImageDataGenerator(
        preprocessing_function=prepro
cessor_efficientnet,
    )
```

```
    test_datagen =
ImageDataGenerator(
        preprocessing_function=prepro
cessor_efficientnet,
    )

    train_gen =
train_datagen.flow_from_directory(
        train_dir,
        target_size=(224, 224),
        batch_size=batch_size,
        class_mode="categorical",
        shuffle=True,
        seed=seed
    )

    val_gen =
val_datagen.flow_from_directory(
        val_dir,
        target_size=(224, 224),
        batch_size=batch_size,
        class_mode="categorical",
        shuffle=False
    )

    test_gen =
test_datagen.flow_from_directory(
        test_dir,
        target_size=(224, 224),
        batch_size=batch_size,
        class_mode="categorical",
        shuffle=False
    )

    return train_gen, val_gen,
test_gen

print("==== BATCH SIZE 16 ====")
train_gen16, val_gen16, test_gen16 =
create_generators(batch_size=16,
seed=42)
print("==== BATCH SIZE 32 ====")
train_gen32, val_gen32, test_gen32 =
create_generators(batch_size=32,
seed=42)
print("==== BATCH SIZE 64 ====")
train_gen64, val_gen64, test_gen64 =
create_generators(batch_size=64,
seed=42)
num_classes = train_gen16.num_classes
class_names =
list(train_gen16.class_indices.keys()
)
print(f"Jumlah kelas pada dataset:
{num_classes}")
print(f'Nama kelas: {class_names}')

print()
print(f"Jumlah data di training
set\t\t: {train_gen16.n} gambar")
print("Jumlah dataset per kelas di
Training\t:")
print(train_gen16.class_indices)
print(Counter(train_gen16.classes))

print()
print(f"Jumlah data di validation
set\t\t: {val_gen16.n} gambar")
print("Jumlah dataset per kelas di
validation\t:")
print(Counter(val_gen16.classes))

print()
print(f"Jumlah data di testing
set\t\t: {test_gen16.n} gambar")
print("Jumlah dataset per kelas di
testing\t:")
print(Counter(test_gen16.classes))
base_model =
EfficientNetB0(input_shape=(224, 224,
3),
                                include
_top=False,
                                weights
='imagenet',
                                )

base_model.trainable = False

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(32, activation='relu')(x)
x = Dropout(0.2)(x)
outputs = layers.Dense(num_classes,
activation="softmax")(x)

EfficientNetB0_model = keras.Model(
    inputs=base_model.input,
    outputs=outputs,
    name="EfficientNetB0_epoch10_batc
h16"
)
EfficientNetB0_model.compile(
    optimizer=Adam(learning_rate=0.00
01),
    loss="categorical_crossentropy",
    metrics=["accuracy"]
    )

EfficientNetB0_model.summary()
val_gen16.reset()

EfficientNetB0_hist =
EfficientNetB0_model.fit(
    train_gen16,
    epochs=10,
    validation_data=val_gen16,
    )
plt.figure(figsize=(14, 6))  #
Perbesar ukuran grafik
plt.suptitle("Grafik Pelatihan Model
EfficientNetB0\n(Epoch=10, Batch
Size=16)", fontsize=16)

# Subplot 1: Accuracy
plt.subplot(1, 2, 1)
plt.plot(EfficientNetB0_hist.history[
"accuracy"], label="Train Accuracy")
plt.plot(EfficientNetB0_hist.history[
"val_accuracy"], label="Validation
Accuracy")
plt.xlabel("Epoch", fontsize=12)
plt.ylabel("Accuracy", fontsize=12)
plt.legend(loc="best", fontsize=10)

# Subplot 2: Loss
plt.subplot(1, 2, 2)
plt.plot(EfficientNetB0_hist.history[
"loss"], label="Train Loss")
plt.plot(EfficientNetB0_hist.history[
"val_loss"], label="Validation Loss")
plt.xlabel("Epoch", fontsize=12)
```

```
plt.ylabel("Loss", fontsize=12)
plt.legend(loc="best", fontsize=10)

plt.savefig("/content/drive/MyDrive/hasil-training/EfficientNetB0 Training/EfficientNetB0 (epoch=10, batch_size=16).png")
plt.show()
print('============ TEST ACC & TEST LOSS (PREDICT) ============')
test_gen16.reset()
predictions = EfficientNetB0_model.predict(test_gen16, steps=len(test_gen16));

true_classes_indices = test_gen16.classes
predicted_classes_indices = np.argmax(predictions, axis=1)

class_labels = list(test_gen16.class_indices.keys())

true_classes_one_hot = to_categorical(true_classes_indices, num_classes=num_classes)
test_acc_predict_manual = accuracy_score(true_classes_indices, predicted_classes_indices)
cce = CategoricalCrossentropy()
test_loss_predict_manual = cce(true_classes_one_hot, predictions).numpy()

print(f"Test Loss (predict_manual)\t: {test_loss_predict_manual:.4f}")
print(f"Test Accuracy (predict_manual)\t: {test_acc_predict_manual:.4f}")

print("\n============ TEST ACC & TEST LOSS (EVALUATE) ============")
test_gen16.reset()
test_loss, test_acc = EfficientNetB0_model.evaluate(test_gen16);
print(f"Test Loss\t: {test_loss:.4f}")
print(f"Test Accuracy\t: {test_acc:.4f}")
cm = confusion_matrix(true_classes_indices, predicted_classes_indices)
print("\n============== HASIL EVALUASI MODEL ==============\n")

report = classification_report(true_classes_indices, predicted_classes_indices, target_names=class_labels)
print(report)

accuracy = accuracy_score(true_classes_indices, predicted_classes_indices)
print(f"Akurasi Keseluruhan: {accuracy:.4f} ({accuracy:.2%})")
print("\n============================================================")

plt.figure(figsize=(12, 10))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=class_labels, yticklabels=class_labels)

plt.title('Confusion Matrix', fontsize=16)
plt.ylabel('Kelas Sebenarnya (True Class)', fontsize=12)
plt.xlabel('Kelas Prediksi (Predicted Class)', fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.yticks(rotation=0)
plt.tight_layout()
plt.savefig("/content/drive/MyDrive/hasil-training/EfficientNetB0 Training/Confusion Matrix - MobileNetV2 (epoch=10, batch_size=16).png")
plt.show()

for i in range(num_classes):
    class_name = class_labels[i]

    TP = cm[i, i]
    FP = cm[:, i].sum() - TP
    FN = cm[i, :].sum() - TP
    TN = np.sum(cm) - (TP + FP + FN)

    print(f"\n--- Kelas: {class_name} ({i}) ---")
    print(f"  True Positives (TP): {TP}")
    print(f"  False Positives (FP): {FP}")
    print(f"  False Negatives (FN): {FN}")
    print(f"  True Negatives (TN): {TN}")
val_gen16.reset()

test_loss, test_acc = EfficientNetB0_model.evaluate(val_gen16)
print("Test loss:", test_loss)
print("Test accuracy:", test_acc)

test_images, test_labels = next(val_gen16)
pred_probs = EfficientNetB0_model.predict(val_gen16)
pred_labels = np.argmax(pred_probs, axis=1)
true_labels = np.argmax(test_labels, axis=1)
training_results = {
    "epoch": list(range(1, len(EfficientNetB0_hist.history["loss"]) + 1)),
    "accuracy": EfficientNetB0_hist.history["accuracy"],
    "val_accuracy": EfficientNetB0_hist.history["val_accuracy"],
    "loss": EfficientNetB0_hist.history["loss"],
    "val_loss": EfficientNetB0_hist.history["val_loss"],
```

```
}

testing_result = {
    "accuracy":
EfficientNetB0_hist.history["accuracy
"][-1],
    "val_accuracy":
EfficientNetB0_hist.history["val_accu
racy"][-1],
    "loss":
EfficientNetB0_hist.history["loss"][-
1],
    "val_loss":
EfficientNetB0_hist.history["val_loss
"][-1],
    "test_accuracy": test_acc,
    "test_loss": test_loss
}

# Simpan ke file JSON
with
open("/content/drive/MyDrive/hasil-
training/EfficientNetB0
Training/EfficientNetB0 (epoch=10,
batch_size=16).json", "w") as f:
    json.dump(training_results, f,
indent=4)

with
open("/content/drive/MyDrive/hasil-
training/EfficientNetB0
Training/testing - EfficientNetB0
(epoch=10, batch_size=16).json", "w")
as f:
    json.dump(testing_result, f,
indent=4)

print("Data training berhasil
disimpan ke EfficientNetB0 (epoch=10,
batch_size=16).json")
print("Data testing berhasil disimpan
ke testing - EfficientNetB0
(epoch=10, batch_size=16).json")
EfficientNetB0_model.save("/content/d
rive/MyDrive/hasil-
training/EfficientNetB0
Training/EfficientNetB0 (epoch=10,
batch_size=16).keras");
EfficientNetB0_model =
tf.keras.models.load_model("/content/
drive/MyDrive/hasil-
training/EfficientNetB0
Training/EfficientNetB0 (epoch=10,
batch_size=16).keras");

converter_keras =
tf.lite.TFLiteConverter.from_keras_mo
del(EfficientNetB0_model)

tflite_model =
converter_keras.convert()
with
open('/content/drive/MyDrive/hasil-
training/EfficientNetB0
Training/EfficientNetB0 (epoch=10,
batch_size=16).tflite', 'wb') as f:
    f.write(tflite_model);
base_model =
EfficientNetB0(input_shape=(224, 224,
3),
                                    include
_top=False,
                                    weights
='imagenet',
                                    )

base_model.trainable = False

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(32, activation='relu')(x)
x = Dropout(0.2)(x)
outputs = layers.Dense(num_classes,
activation="softmax")(x)

EfficientNetB0_model = keras.Model(
    inputs=base_model.input,
    outputs=outputs,
    name="EfficientNetB0_epoch10_batc
h32"
)
EfficientNetB0_model.compile(
    optimizer=Adam(learning_rate=0.00
01),
    loss="categorical_crossentropy",
    metrics=["accuracy"]
    )

EfficientNetB0_model.summary()
val_gen32.reset()

EfficientNetB0_hist =
EfficientNetB0_model.fit(
    train_gen32,
    epochs=10,
    validation_data=val_gen32,
    )
plt.figure(figsize=(14, 6))  #
Perbesar ukuran grafik
plt.suptitle("Grafik Pelatihan Model
EfficientNetB0\n(Epoch=10, Batch
Size=32)", fontsize=16)

# Subplot 1: Accuracy
plt.subplot(1, 2, 1)
plt.plot(EfficientNetB0_hist.history[
"accuracy"], label="Train Accuracy")
plt.plot(EfficientNetB0_hist.history[
"val_accuracy"], label="Validation
Accuracy")
plt.xlabel("Epoch", fontsize=12)
plt.ylabel("Accuracy", fontsize=12)
plt.legend(loc="best", fontsize=10)

# Subplot 2: Loss
plt.subplot(1, 2, 2)
plt.plot(EfficientNetB0_hist.history[
"loss"], label="Train Loss")
plt.plot(EfficientNetB0_hist.history[
"val_loss"], label="Validation Loss")
plt.xlabel("Epoch", fontsize=12)
plt.ylabel("Loss", fontsize=12)
plt.legend(loc="best", fontsize=10)

plt.savefig("/content/drive/MyDrive/h
asil-training/EfficientNetB0
Training/EfficientNetB0 (epoch=10,
batch_size=32).png")
plt.show()
print('============= TEST ACC & TEST
LOSS (PREDICT) ============')
```

```
test_gen32.reset()
predictions =
EfficientNetB0_model.predict(test_gen
32, steps=len(test_gen32));

true_classes_indices =
test_gen32.classes
predicted_classes_indices =
np.argmax(predictions, axis=1)

class_labels =
list(test_gen32.class_indices.keys())

true_classes_one_hot =
to_categorical(true_classes_indices,
num_classes=num_classes)
test_acc_predict_manual =
accuracy_score(true_classes_indices,
predicted_classes_indices)
cce = CategoricalCrossentropy()
test_loss_predict_manual =
cce(true_classes_one_hot,
predictions).numpy()

print(f"Test Loss (predict_manual)\t:
{test_loss_predict_manual:.4f}")
print(f"Test Accuracy
(predict_manual)\t:
{test_acc_predict_manual:.4f}")

print("\n============= TEST ACC &
TEST LOSS (EVALUATE) ============")
test_gen32.reset()
test_loss, test_acc =
EfficientNetB0_model.evaluate(test_ge
n32);
print(f"Test Loss\t:
{test_loss:.4f}")
print(f"Test Accuracy\t:
{test_acc:.4f}")
cm =
confusion_matrix(true_classes_indices
, predicted_classes_indices)
print("\n============== HASIL
EVALUASI MODEL ==============\n")

report =
classification_report(true_classes_in
dices, predicted_classes_indices,
target_names=class_labels)
print(report)

accuracy =
accuracy_score(true_classes_indices,
predicted_classes_indices)
print(f"Akurasi Keseluruhan:
{accuracy:.4f} ({accuracy:.2%})")
print("\n============================
======================")

plt.figure(figsize=(12, 10))
sns.heatmap(cm, annot=True, fmt='d',
cmap='Blues',
            xticklabels=class_labels,
yticklabels=class_labels)

plt.title('Confusion Matrix',
fontsize=16)
plt.ylabel('Kelas Sebenarnya (True
Class)', fontsize=12)
plt.xlabel('Kelas Prediksi (Predicted
Class)', fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.yticks(rotation=0)
plt.tight_layout()
plt.savefig("/content/drive/MyDrive/h
asil-training/EfficientNetB0
Training/Confusion Matrix -
MobileNetV2 (epoch=10,
batch_size=32).png")
plt.show()

for i in range(num_classes):
    class_name = class_labels[i]

    TP = cm[i, i]
    FP = cm[:, i].sum() - TP
    FN = cm[i, :].sum() - TP
    TN = np.sum(cm) - (TP + FP + FN)

    print(f"\n--- Kelas: {class_name}
({i}) ---")
    print(f"  True Positives (TP):
{TP}")
    print(f"  False Positives (FP):
{FP}")
    print(f"  False Negatives (FN):
{FN}")
    print(f"  True Negatives (TN):
{TN}")
val_gen32.reset()

test_loss, test_acc =
EfficientNetB0_model.evaluate(val_gen
32)
print("Test loss:", test_loss)
print("Test accuracy:", test_acc)

test_images, test_labels =
next(val_gen32)
pred_probs =
EfficientNetB0_model.predict(val_gen3
2)
pred_labels = np.argmax(pred_probs,
axis=1)
true_labels = np.argmax(test_labels,
axis=1)
training_results = {
    "epoch": list(range(1,
len(EfficientNetB0_hist.history["loss
"]) + 1)),
    "accuracy":
EfficientNetB0_hist.history["accuracy
"],
    "val_accuracy":
EfficientNetB0_hist.history["val_accu
racy"],
    "loss":
EfficientNetB0_hist.history["loss"],
    "val_loss":
EfficientNetB0_hist.history["val_loss
"],
}

testing_result = {
    "accuracy":
EfficientNetB0_hist.history["accuracy
"][-1],
    "val_accuracy":
EfficientNetB0_hist.history["val_accu
racy"][-1],
```

```
    "loss":
EfficientNetB0_hist.history["loss"][-
1],
    "val_loss":
EfficientNetB0_hist.history["val_loss
"][-1],
    "test_accuracy": test_acc,
    "test_loss": test_loss
}

# Simpan ke file JSON
with
open("/content/drive/MyDrive/hasil-
training/EfficientNetB0
Training/EfficientNetB0 (epoch=10,
batch_size=32).json", "w") as f:
    json.dump(training_results, f,
indent=4)

with
open("/content/drive/MyDrive/hasil-
training/EfficientNetB0
Training/testing - EfficientNetB0
(epoch=10, batch_size=32).json", "w")
as f:
    json.dump(testing_result, f,
indent=4)

print("Data training berhasil
disimpan ke EfficientNetB0 (epoch=10,
batch_size=32).json")
print("Data testing berhasil disimpan
ke testing - EfficientNetB0
(epoch=10, batch_size=32).json")
EfficientNetB0_model.save("/content/d
rive/MyDrive/hasil-
training/EfficientNetB0
Training/EfficientNetB0 (epoch=10,
batch_size=16).keras");
EfficientNetB0_model =
tf.keras.models.load_model("/content/
drive/MyDrive/hasil-
training/EfficientNetB0
Training/EfficientNetB0 (epoch=10,
batch_size=16).keras");

converter_keras =
tf.lite.TFLiteConverter.from_keras_mo
del(EfficientNetB0_model)

tflite_model =
converter_keras.convert()
with
open('/content/drive/MyDrive/hasil-
training/EfficientNetB0
Training/EfficientNetB0 (epoch=10,
batch_size=16).tflite', 'wb') as f:
    f.write(tflite_model);
base_model =
EfficientNetB0(input_shape=(224, 224,
3),
                               include
_top=False,
                               weights
='imagenet',
                               )

base_model.trainable = False

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(32, activation='relu')(x)
x = Dropout(0.2)(x)
outputs = layers.Dense(num_classes,
activation="softmax")(x)

EfficientNetB0_model = keras.Model(
    inputs=base_model.input,
    outputs=outputs,
    name="EfficientNetB0_epoch10_batc
h64"
)
EfficientNetB0_model.compile(
    optimizer=Adam(learning_rate=0.00
01),
    loss="categorical_crossentropy",
    metrics=["accuracy"]
    )

EfficientNetB0_model.summary()
val_gen64.reset()

EfficientNetB0_hist =
EfficientNetB0_model.fit(
    train_gen64,
    epochs=10,
    validation_data=val_gen64,
    )
plt.figure(figsize=(14, 6))  #
Perbesar ukuran grafik
plt.suptitle("Grafik Pelatihan Model
EfficientNetB0\n(Epoch=10, Batch
Size=64)", fontsize=16)

# Subplot 1: Accuracy
plt.subplot(1, 2, 1)
plt.plot(EfficientNetB0_hist.history[
"accuracy"], label="Train Accuracy")
plt.plot(EfficientNetB0_hist.history[
"val_accuracy"], label="Validation
Accuracy")
plt.xlabel("Epoch", fontsize=12)
plt.ylabel("Accuracy", fontsize=12)
plt.legend(loc="best", fontsize=10)

# Subplot 2: Loss
plt.subplot(1, 2, 2)
plt.plot(EfficientNetB0_hist.history[
"loss"], label="Train Loss")
plt.plot(EfficientNetB0_hist.history[
"val_loss"], label="Validation Loss")
plt.xlabel("Epoch", fontsize=12)
plt.ylabel("Loss", fontsize=12)
plt.legend(loc="best", fontsize=10)

plt.savefig("/content/drive/MyDrive/h
asil-training/EfficientNetB0
Training/EfficientNetB0 (epoch=10,
batch_size=64).png")
plt.show()
print('============== TEST ACC & TEST
LOSS (PREDICT) ============')
test_gen64.reset()
predictions =
EfficientNetB0_model.predict(test_gen
64, steps=len(test_gen64));

true_classes_indices =
test_gen64.classes
predicted_classes_indices =
np.argmax(predictions, axis=1)
```

```
class_labels =
list(test_gen64.class_indices.keys())

true_classes_one_hot =
to_categorical(true_classes_indices,
num_classes=num_classes)
test_acc_predict_manual =
accuracy_score(true_classes_indices,
predicted_classes_indices)
cce = CategoricalCrossentropy()
test_loss_predict_manual =
cce(true_classes_one_hot,
predictions).numpy()

print(f"Test Loss (predict_manual)\t:
{test_loss_predict_manual:.4f}")
print(f"Test Accuracy
(predict_manual)\t:
{test_acc_predict_manual:.4f}")

print("\n============= TEST ACC &
TEST LOSS (EVALUATE) ============")
test_gen64.reset()
test_loss, test_acc =
EfficientNetB0_model.evaluate(test_ge
n64);
print(f"Test Loss\t:
{test_loss:.4f}")
print(f"Test Accuracy\t:
{test_acc:.4f}")
cm =
confusion_matrix(true_classes_indices
, predicted_classes_indices)
print("\n============== HASIL
EVALUASI MODEL ==============\n")

report =
classification_report(true_classes_in
dices, predicted_classes_indices,
target_names=class_labels)
print(report)

accuracy =
accuracy_score(true_classes_indices,
predicted_classes_indices)
print(f"Akurasi Keseluruhan:
{accuracy:.4f} ({accuracy:.2%})")
print("\n============================
======================")

plt.figure(figsize=(12, 10))
sns.heatmap(cm, annot=True, fmt='d',
cmap='Blues',
            xticklabels=class_labels,
yticklabels=class_labels)

plt.title('Confusion Matrix',
fontsize=16)
plt.ylabel('Kelas Sebenarnya (True
Class)', fontsize=12)
plt.xlabel('Kelas Prediksi (Predicted
Class)', fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.yticks(rotation=0)
plt.tight_layout()
plt.savefig("/content/drive/MyDrive/h
asil-training/EfficientNetB0
Training/Confusion Matrix -
MobileNetV2 (epoch=10,
batch_size=64).png")
plt.show()
```

```
for i in range(num_classes):
    class_name = class_labels[i]

    TP = cm[i, i]
    FP = cm[:, i].sum() - TP
    FN = cm[i, :].sum() - TP
    TN = np.sum(cm) - (TP + FP + FN)

    print(f"\n--- Kelas: {class_name}
({i}) ---")
    print(f"  True Positives (TP):
{TP}")
    print(f"  False Positives (FP):
{FP}")
    print(f"  False Negatives (FN):
{FN}")
    print(f"  True Negatives (TN):
{TN}")
val_gen64.reset()

test_loss, test_acc =
EfficientNetB0_model.evaluate(val_gen
64)
print("Test loss:", test_loss)
print("Test accuracy:", test_acc)

test_images, test_labels =
next(val_gen64)
pred_probs =
EfficientNetB0_model.predict(val_gen6
4)
pred_labels = np.argmax(pred_probs,
axis=1)
true_labels = np.argmax(test_labels,
axis=1)
training_results = {
    "epoch": list(range(1,
len(EfficientNetB0_hist.history["loss
"]) + 1)),
    "accuracy":
EfficientNetB0_hist.history["accuracy
"],
    "val_accuracy":
EfficientNetB0_hist.history["val_accu
racy"],
    "loss":
EfficientNetB0_hist.history["loss"],
    "val_loss":
EfficientNetB0_hist.history["val_loss
"],
}

testing_result = {
    "accuracy":
EfficientNetB0_hist.history["accuracy
"][-1],
    "val_accuracy":
EfficientNetB0_hist.history["val_accu
racy"][-1],
    "loss":
EfficientNetB0_hist.history["loss"][-
1],
    "val_loss":
EfficientNetB0_hist.history["val_loss
"][-1],
    "test_accuracy": test_acc,
    "test_loss": test_loss
}

# Simpan ke file JSON
```

```
with
open("/content/drive/MyDrive/hasil-
training/EfficientNetB0
Training/EfficientNetB0 (epoch=10,
batch_size=64).json", "w") as f:
    json.dump(training_results, f,
indent=4)

with
open("/content/drive/MyDrive/hasil-
training/EfficientNetB0
Training/testing - EfficientNetB0
(epoch=10, batch_size=64).json", "w")
as f:
    json.dump(testing_result, f,
indent=4)

print("Data training berhasil
disimpan ke EfficientNetB0 (epoch=10,
batch_size=64).json")
print("Data testing berhasil disimpan
ke testing - EfficientNetB0
(epoch=10, batch_size=64).json")
EfficientNetB0_model.save("/content/d
rive/MyDrive/hasil-
training/EfficientNetB0
Training/EfficientNetB0 (epoch=10,
batch_size=64).keras");
EfficientNetB0_model =
tf.keras.models.load_model("/content/
drive/MyDrive/hasil-
training/EfficientNetB0
Training/EfficientNetB0 (epoch=10,
batch_size=64).keras");

converter_keras =
tf.lite.TFLiteConverter.from_keras_mo
del(EfficientNetB0_model)

tflite_model =
converter_keras.convert()
with
open('/content/drive/MyDrive/hasil-
training/EfficientNetB0
Training/EfficientNetB0 (epoch=10,
batch_size=64).tflite', 'wb') as f:
    f.write(tflite_model);
base_model =
EfficientNetB0(input_shape=(224, 224,
3),
                                include
_top=False,
                                weights
='imagenet',
                                )

base_model.trainable = False

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(32, activation='relu')(x)
x = Dropout(0.2)(x)
outputs = layers.Dense(num_classes,
activation="softmax")(x)

EfficientNetB0_model = keras.Model(
    inputs=base_model.input,
    outputs=outputs,
    name="EfficientNetB0_epoch30_batc
h16"
)

EfficientNetB0_model.compile(
    optimizer=Adam(learning_rate=0.00
01),
    loss="categorical_crossentropy",
    metrics=["accuracy"]
    )

EfficientNetB0_model.summary()
val_gen16.reset()

EfficientNetB0_hist =
EfficientNetB0_model.fit(
    train_gen16,
    epochs=30,
    validation_data=val_gen16,
    )
plt.figure(figsize=(14, 6))  #
Perbesar ukuran grafik
plt.suptitle("Grafik Pelatihan Model
EfficientNetB0\n(Epoch=30, Batch
Size=16)", fontsize=16)

# Subplot 1: Accuracy
plt.subplot(1, 2, 1)
plt.plot(EfficientNetB0_hist.history[
"accuracy"], label="Train Accuracy")
plt.plot(EfficientNetB0_hist.history[
"val_accuracy"], label="Validation
Accuracy")
plt.xlabel("Epoch", fontsize=12)
plt.ylabel("Accuracy", fontsize=12)
plt.legend(loc="best", fontsize=10)

# Subplot 2: Loss
plt.subplot(1, 2, 2)
plt.plot(EfficientNetB0_hist.history[
"loss"], label="Train Loss")
plt.plot(EfficientNetB0_hist.history[
"val_loss"], label="Validation Loss")
plt.xlabel("Epoch", fontsize=12)
plt.ylabel("Loss", fontsize=12)
plt.legend(loc="best", fontsize=10)

plt.savefig("/content/drive/MyDrive/h
asil-training/EfficientNetB0
Training/EfficientNetB0 (epoch=30,
batch_size=16).png")
plt.show()
print('============= TEST ACC & TEST
LOSS (PREDICT) ============')
test_gen16.reset()
predictions =
EfficientNetB0_model.predict(test_gen
16, steps=len(test_gen16));

true_classes_indices =
test_gen16.classes
predicted_classes_indices =
np.argmax(predictions, axis=1)

class_labels =
list(test_gen16.class_indices.keys())

true_classes_one_hot =
to_categorical(true_classes_indices,
num_classes=num_classes)
test_acc_predict_manual =
accuracy_score(true_classes_indices,
predicted_classes_indices)
cce = CategoricalCrossentropy()
```

```
test_loss_predict_manual =
cce(true_classes_one_hot,
predictions).numpy()

print(f"Test Loss (predict_manual)\t:
{test_loss_predict_manual:.4f}")
print(f"Test Accuracy
(predict_manual)\t:
{test_acc_predict_manual:.4f}")

print("\n============ TEST ACC &
TEST LOSS (EVALUATE) ============")
test_gen16.reset()
test_loss, test_acc =
EfficientNetB0_model.evaluate(test_ge
n16);
print(f"Test Loss\t:
{test_loss:.4f}")
print(f"Test Accuracy\t:
{test_acc:.4f}")
cm =
confusion_matrix(true_classes_indices
, predicted_classes_indices)
print("\n============== HASIL
EVALUASI MODEL ==============\n")

report =
classification_report(true_classes_in
dices, predicted_classes_indices,
target_names=class_labels)
print(report)

accuracy =
accuracy_score(true_classes_indices,
predicted_classes_indices)
print(f"Akurasi Keseluruhan:
{accuracy:.4f} ({accuracy:.2%})")
print("\n============================
======================")

plt.figure(figsize=(12, 10))
sns.heatmap(cm, annot=True, fmt='d',
cmap='Blues',
            xticklabels=class_labels,
yticklabels=class_labels)

plt.title('Confusion Matrix',
fontsize=16)
plt.ylabel('Kelas Sebenarnya (True
Class)', fontsize=12)
plt.xlabel('Kelas Prediksi (Predicted
Class)', fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.yticks(rotation=0)
plt.tight_layout()
plt.savefig("/content/drive/MyDrive/h
asil-training/EfficientNetB0
Training/Confusion Matrix -
MobileNetV2 (epoch=30,
batch_size=16).png")
plt.show()

cm =
confusion_matrix(true_classes_indices
, predicted_classes_indices)
print("\n============== HASIL
EVALUASI MODEL ==============\n")

report =
classification_report(true_classes_in
dices, predicted_classes_indices,
target_names=class_labels)
print(report)

accuracy =
accuracy_score(true_classes_indices,
predicted_classes_indices)
print(f"Akurasi Keseluruhan:
{accuracy:.4f} ({accuracy:.2%})")
print("\n============================
======================")

plt.figure(figsize=(12, 10))
sns.heatmap(cm, annot=True, fmt='d',
cmap='Blues',
            xticklabels=class_labels,
yticklabels=class_labels)

plt.title('Confusion Matrix',
fontsize=16)
plt.ylabel('Kelas Sebenarnya (True
Class)', fontsize=12)
plt.xlabel('Kelas Prediksi (Predicted
Class)', fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.yticks(rotation=0)
plt.tight_layout()
plt.savefig("/content/drive/MyDrive/h
asil-training/EfficientNetB0
Training/Confusion Matrix -
MobileNetV2 (epoch=30,
batch_size=16).png")
plt.show()

val_gen16.reset()

test_loss, test_acc =
EfficientNetB0_model.evaluate(val_gen
16)
print("Test loss:", test_loss)
print("Test accuracy:", test_acc)

test_images, test_labels =
next(val_gen16)
pred_probs =
EfficientNetB0_model.predict(val_gen1
6)
pred_labels = np.argmax(pred_probs,
axis=1)
true_labels = np.argmax(test_labels,
axis=1)
training_results = {
    "epoch": list(range(1,
len(EfficientNetB0_hist.history["loss
"]) + 1)),
    "accuracy":
EfficientNetB0_hist.history["accuracy
"],
    "val_accuracy":
EfficientNetB0_hist.history["val_accu
racy"],
    "loss":
EfficientNetB0_hist.history["loss"],
    "val_loss":
EfficientNetB0_hist.history["val_loss
"],
}

testing_result = {
```

```
    "accuracy":
EfficientNetB0_hist.history["accuracy
"][-1],
    "val_accuracy":
EfficientNetB0_hist.history["val_accu
racy"][-1],
    "loss":
EfficientNetB0_hist.history["loss"][-
1],
    "val_loss":
EfficientNetB0_hist.history["val_loss
"][-1],
    "test_accuracy": test_acc,
    "test_loss": test_loss
}

# Simpan ke file JSON
with
open("/content/drive/MyDrive/hasil-
training/EfficientNetB0
Training/EfficientNetB0 (epoch=30,
batch_size=16).json", "w") as f:
    json.dump(training_results, f,
indent=4)

with
open("/content/drive/MyDrive/hasil-
training/EfficientNetB0
Training/testing - EfficientNetB0
(epoch=30, batch_size=16).json", "w")
as f:
    json.dump(testing_result, f,
indent=4)

print("Data training berhasil
disimpan ke EfficientNetB0 (epoch=30,
batch_size=16).json")
print("Data testing berhasil disimpan
ke testing - EfficientNetB0
(epoch=30, batch_size=16).json")
EfficientNetB0_model.save("/content/d
rive/MyDrive/hasil-
training/EfficientNetB0
Training/EfficientNetB0 (epoch=30,
batch_size=16).keras");
EfficientNetB0_model =
tf.keras.models.load_model("/content/
drive/MyDrive/hasil-
training/EfficientNetB0
Training/EfficientNetB0 (epoch=30,
batch_size=16).keras");

converter_keras =
tf.lite.TFLiteConverter.from_keras_mo
del(EfficientNetB0_model)

tflite_model =
converter_keras.convert()
with
open('/content/drive/MyDrive/hasil-
training/EfficientNetB0
Training/EfficientNetB0 (epoch=30,
batch_size=16).tflite', 'wb') as f:
    f.write(tflite_model);
base_model =
EfficientNetB0(input_shape=(224, 224,
3),
                                include
_top=False,
                                weights
='imagenet',
                                )

base_model.trainable = False

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(32, activation='relu')(x)
x = Dropout(0.2)(x)
outputs = layers.Dense(num_classes,
activation="softmax")(x)

EfficientNetB0_model = keras.Model(
    inputs=base_model.input,
    outputs=outputs,
    name="EfficientNetB0_epoch30_batc
h32"
)
EfficientNetB0_model.compile(
    optimizer=Adam(learning_rate=0.00
01),
    loss="categorical_crossentropy",
    metrics=["accuracy"]
    )

EfficientNetB0_model.summary()
val_gen32.reset()

EfficientNetB0_hist =
EfficientNetB0_model.fit(
    train_gen32,
    epochs=30,
    validation_data=val_gen32,
    )
plt.figure(figsize=(14, 6))  #
Perbesar ukuran grafik
plt.suptitle("Grafik Pelatihan Model
EfficientNetB0\n(Epoch=30, Batch
Size=32)", fontsize=16)

# Subplot 1: Accuracy
plt.subplot(1, 2, 1)
plt.plot(EfficientNetB0_hist.history[
"accuracy"], label="Train Accuracy")
plt.plot(EfficientNetB0_hist.history[
"val_accuracy"], label="Validation
Accuracy")
plt.xlabel("Epoch", fontsize=12)
plt.ylabel("Accuracy", fontsize=12)
plt.legend(loc="best", fontsize=10)

# Subplot 2: Loss
plt.subplot(1, 2, 2)
plt.plot(EfficientNetB0_hist.history[
"loss"], label="Train Loss")
plt.plot(EfficientNetB0_hist.history[
"val_loss"], label="Validation Loss")
plt.xlabel("Epoch", fontsize=12)
plt.ylabel("Loss", fontsize=12)
plt.legend(loc="best", fontsize=10)

plt.savefig("/content/drive/MyDrive/h
asil-training/EfficientNetB0
Training/EfficientNetB0 (epoch=30,
batch_size=32).png")
plt.show()
print('============= TEST ACC & TEST
LOSS (PREDICT) ============')
test_gen32.reset()
predictions =
EfficientNetB0_model.predict(test_gen
32, steps=len(test_gen32));
```

```
true_classes_indices =
test_gen32.classes
predicted_classes_indices =
np.argmax(predictions, axis=1)

class_labels =
list(test_gen32.class_indices.keys())

true_classes_one_hot =
to_categorical(true_classes_indices,
num_classes=num_classes)
test_acc_predict_manual =
accuracy_score(true_classes_indices,
predicted_classes_indices)
cce = CategoricalCrossentropy()
test_loss_predict_manual =
cce(true_classes_one_hot,
predictions).numpy()

print(f"Test Loss (predict_manual)\t:
{test_loss_predict_manual:.4f}")
print(f"Test Accuracy
(predict_manual)\t:
{test_acc_predict_manual:.4f}")

print("\n============= TEST ACC &
TEST LOSS (EVALUATE) ============")
test_gen32.reset()
test_loss, test_acc =
EfficientNetB0_model.evaluate(test_ge
n32);
print(f"Test Loss\t:
{test_loss:.4f}")
print(f"Test Accuracy\t:
{test_acc:.4f}")
cm =
confusion_matrix(true_classes_indices
, predicted_classes_indices)
print("\n============== HASIL
EVALUASI MODEL ==============\n")

report =
classification_report(true_classes_in
dices, predicted_classes_indices,
target_names=class_labels)
print(report)

accuracy =
accuracy_score(true_classes_indices,
predicted_classes_indices)
print(f"Akurasi Keseluruhan:
{accuracy:.4f} ({accuracy:.2%})")
print("\n============================
======================")

plt.figure(figsize=(12, 10))
sns.heatmap(cm, annot=True, fmt='d',
cmap='Blues',
            xticklabels=class_labels,
yticklabels=class_labels)

plt.title('Confusion Matrix',
fontsize=16)
plt.ylabel('Kelas Sebenarnya (True
Class)', fontsize=12)
plt.xlabel('Kelas Prediksi (Predicted
Class)', fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.yticks(rotation=0)
plt.tight_layout()
plt.savefig("/content/drive/MyDrive/h
asil-training/EfficientNetB0
Training/Confusion Matrix -
MobileNetV2 (epoch=30,
batch_size=32).png")
plt.show()

for i in range(num_classes):
    class_name = class_labels[i]

    TP = cm[i, i]
    FP = cm[:, i].sum() - TP
    FN = cm[i, :].sum() - TP
    TN = np.sum(cm) - (TP + FP + FN)

    print(f"\n--- Kelas: {class_name}
({i}) ---")
    print(f"  True Positives (TP):
{TP}")
    print(f"  False Positives (FP):
{FP}")
    print(f"  False Negatives (FN):
{FN}")
    print(f"  True Negatives (TN):
{TN}")
val_gen32.reset()

test_loss, test_acc =
EfficientNetB0_model.evaluate(val_gen
32)
print("Test loss:", test_loss)
print("Test accuracy:", test_acc)

test_images, test_labels =
next(val_gen32)
pred_probs =
EfficientNetB0_model.predict(val_gen3
2)
pred_labels = np.argmax(pred_probs,
axis=1)
true_labels = np.argmax(test_labels,
axis=1)
training_results = {
    "epoch": list(range(1,
len(EfficientNetB0_hist.history["loss
"]) + 1)),
    "accuracy":
EfficientNetB0_hist.history["accuracy
"],
    "val_accuracy":
EfficientNetB0_hist.history["val_accu
racy"],
    "loss":
EfficientNetB0_hist.history["loss"],
    "val_loss":
EfficientNetB0_hist.history["val_loss
"],
}

testing_result = {
    "accuracy":
EfficientNetB0_hist.history["accuracy
"][-1],
    "val_accuracy":
EfficientNetB0_hist.history["val_accu
racy"][-1],
    "loss":
EfficientNetB0_hist.history["loss"][-
1],
```

```
    "val_loss":
EfficientNetB0_hist.history["val_loss
"][-1],
    "test_accuracy": test_acc,
    "test_loss": test_loss
}

# Simpan ke file JSON
with
open("/content/drive/MyDrive/hasil-
training/EfficientNetB0
Training/EfficientNetB0 (epoch=30,
batch_size=32).json", "w") as f:
    json.dump(training_results, f,
indent=4)

with
open("/content/drive/MyDrive/hasil-
training/EfficientNetB0
Training/testing - EfficientNetB0
(epoch=30, batch_size=32).json", "w")
as f:
    json.dump(testing_result, f,
indent=4)

print("Data training berhasil
disimpan ke EfficientNetB0 (epoch=30,
batch_size=32).json")
print("Data testing berhasil disimpan
ke testing - EfficientNetB0
(epoch=30, batch_size=32).json")
EfficientNetB0_model.save("/content/d
rive/MyDrive/hasil-
training/EfficientNetB0
Training/EfficientNetB0 (epoch=30,
batch_size=32).keras");
EfficientNetB0_model =
tf.keras.models.load_model("/content/
drive/MyDrive/hasil-
training/EfficientNetB0
Training/EfficientNetB0 (epoch=30,
batch_size=32).keras");

converter_keras =
tf.lite.TFLiteConverter.from_keras_mo
del(EfficientNetB0_model)

tflite_model =
converter_keras.convert()
with
open('/content/drive/MyDrive/hasil-
training/EfficientNetB0
Training/EfficientNetB0 (epoch=30,
batch_size=32).tflite', 'wb') as f:
    f.write(tflite_model);
base_model =
EfficientNetB0(input_shape=(224, 224,
3),
                               include
_top=False,
                               weights
='imagenet',
                               )

base_model.trainable = False

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(32, activation='relu')(x)
x = Dropout(0.2)(x)
outputs = layers.Dense(num_classes,
activation="softmax")(x)

EfficientNetB0_model = keras.Model(
    inputs=base_model.input,
    outputs=outputs,
    name="EfficientNetB0_epoch30_batc
h64"
)
EfficientNetB0_model.compile(
    optimizer=Adam(learning_rate=0.00
01),
    loss="categorical_crossentropy",
    metrics=["accuracy"]
    )

EfficientNetB0_model.summary()
val_gen64.reset()

EfficientNetB0_hist =
EfficientNetB0_model.fit(
    train_gen64,
    epochs=30,
    validation_data=val_gen64,
    )
plt.figure(figsize=(14, 6))  #
Perbesar ukuran grafik
plt.suptitle("Grafik Pelatihan Model
EfficientNetB0\n(Epoch=30, Batch
Size=64)", fontsize=16)

# Subplot 1: Accuracy
plt.subplot(1, 2, 1)
plt.plot(EfficientNetB0_hist.history[
"accuracy"], label="Train Accuracy")
plt.plot(EfficientNetB0_hist.history[
"val_accuracy"], label="Validation
Accuracy")
plt.xlabel("Epoch", fontsize=12)
plt.ylabel("Accuracy", fontsize=12)
plt.legend(loc="best", fontsize=10)

# Subplot 2: Loss
plt.subplot(1, 2, 2)
plt.plot(EfficientNetB0_hist.history[
"loss"], label="Train Loss")
plt.plot(EfficientNetB0_hist.history[
"val_loss"], label="Validation Loss")
plt.xlabel("Epoch", fontsize=12)
plt.ylabel("Loss", fontsize=12)
plt.legend(loc="best", fontsize=10)

plt.savefig("/content/drive/MyDrive/h
asil-training/EfficientNetB0
Training/EfficientNetB0 (epoch=30,
batch_size=64).png")
plt.show()
print('============= TEST ACC & TEST
LOSS (PREDICT) ============')
test_gen64.reset()
predictions =
EfficientNetB0_model.predict(test_gen
64, steps=len(test_gen64));

true_classes_indices =
test_gen64.classes
predicted_classes_indices =
np.argmax(predictions, axis=1)

class_labels =
list(test_gen64.class_indices.keys())
```

```
true_classes_one_hot =
to_categorical(true_classes_indices,
num_classes=num_classes)
test_acc_predict_manual =
accuracy_score(true_classes_indices,
predicted_classes_indices)
cce = CategoricalCrossentropy()
test_loss_predict_manual =
cce(true_classes_one_hot,
predictions).numpy()

print(f"Test Loss (predict_manual)\t:
{test_loss_predict_manual:.4f}")
print(f"Test Accuracy
(predict_manual)\t:
{test_acc_predict_manual:.4f}")

print("\n============= TEST ACC &
TEST LOSS (EVALUATE) ============")
test_gen64.reset()
test_loss, test_acc =
EfficientNetB0_model.evaluate(test_ge
n64);
print(f"Test Loss\t:
{test_loss:.4f}")
print(f"Test Accuracy\t:
{test_acc:.4f}")
cm =
confusion_matrix(true_classes_indices
, predicted_classes_indices)
print("\n============== HASIL
EVALUASI MODEL ==============\n")

report =
classification_report(true_classes_in
dices, predicted_classes_indices,
target_names=class_labels)
print(report)

accuracy =
accuracy_score(true_classes_indices,
predicted_classes_indices)
print(f"Akurasi Keseluruhan:
{accuracy:.4f} ({accuracy:.2%})")
print("\n============================
======================")

plt.figure(figsize=(12, 10))
sns.heatmap(cm, annot=True, fmt='d',
cmap='Blues',
            xticklabels=class_labels,
yticklabels=class_labels)

plt.title('Confusion Matrix',
fontsize=16)
plt.ylabel('Kelas Sebenarnya (True
Class)', fontsize=12)
plt.xlabel('Kelas Prediksi (Predicted
Class)', fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.yticks(rotation=0)
plt.tight_layout()
plt.savefig("/content/drive/MyDrive/h
asil-training/EfficientNetB0
Training/Confusion Matrix -
MobileNetV2 (epoch=30,
batch_size=64).png")
plt.show()

for i in range(num_classes):
    class_name = class_labels[i]

    TP = cm[i, i]
    FP = cm[:, i].sum() - TP
    FN = cm[i, :].sum() - TP
    TN = np.sum(cm) - (TP + FP + FN)

    print(f"\n--- Kelas: {class_name}
({i}) ---")
    print(f"  True Positives (TP):
{TP}")
    print(f"  False Positives (FP):
{FP}")
    print(f"  False Negatives (FN):
{FN}")
    print(f"  True Negatives (TN):
{TN}")
val_gen64.reset()

test_loss, test_acc =
EfficientNetB0_model.evaluate(val_gen
64)
print("Test loss:", test_loss)
print("Test accuracy:", test_acc)

test_images, test_labels =
next(val_gen64)
pred_probs =
EfficientNetB0_model.predict(val_gen6
4)
pred_labels = np.argmax(pred_probs,
axis=1)
true_labels = np.argmax(test_labels,
axis=1)
training_results = {
    "epoch": list(range(1,
len(EfficientNetB0_hist.history["loss
"]) + 1)),
    "accuracy":
EfficientNetB0_hist.history["accuracy
"],
    "val_accuracy":
EfficientNetB0_hist.history["val_accu
racy"],
    "loss":
EfficientNetB0_hist.history["loss"],
    "val_loss":
EfficientNetB0_hist.history["val_loss
"],
}

testing_result = {
    "accuracy":
EfficientNetB0_hist.history["accuracy
"][-1],
    "val_accuracy":
EfficientNetB0_hist.history["val_accu
racy"][-1],
    "loss":
EfficientNetB0_hist.history["loss"][-
1],
    "val_loss":
EfficientNetB0_hist.history["val_loss
"][-1],
    "test_accuracy": test_acc,
    "test_loss": test_loss
}

# Simpan ke file JSON
with
open("/content/drive/MyDrive/hasil-
```

```
training/EfficientNetB0
Training/EfficientNetB0 (epoch=30,
batch_size=64).json", "w") as f:
    json.dump(training_results, f,
indent=4)

with
open("/content/drive/MyDrive/hasil-
training/EfficientNetB0
Training/testing - EfficientNetB0
(epoch=30, batch_size=64).json", "w")
as f:
    json.dump(testing_result, f,
indent=4)

print("Data training berhasil
disimpan ke EfficientNetB0 (epoch=30,
batch_size=64).json")
print("Data testing berhasil disimpan
ke testing - EfficientNetB0
(epoch=30, batch_size=64).json")
EfficientNetB0_model.save("/content/d
rive/MyDrive/hasil-
training/EfficientNetB0
Training/EfficientNetB0 (epoch=30,
batch_size=64).keras");
EfficientNetB0_model =
tf.keras.models.load_model("/content/
drive/MyDrive/hasil-
training/EfficientNetB0
Training/EfficientNetB0 (epoch=30,
batch_size=64).keras");

converter_keras =
tf.lite.TFLiteConverter.from_keras_mo
del(EfficientNetB0_model)

tflite_model =
converter_keras.convert()
with
open('/content/drive/MyDrive/hasil-
training/EfficientNetB0
Training/EfficientNetB0 (epoch=30,
batch_size=64).tflite', 'wb') as f:
    f.write(tflite_model);
base_model =
EfficientNetB0(input_shape=(224, 224,
3),
                               include
_top=False,
                               weights
='imagenet',
                               )

base_model.trainable = False

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(32, activation='relu')(x)
x = Dropout(0.2)(x)
outputs = layers.Dense(num_classes,
activation="softmax")(x)

EfficientNetB0_model = keras.Model(
    inputs=base_model.input,
    outputs=outputs,
    name="EfficientNetB0_epoch50_batc
h16"
)
EfficientNetB0_model.compile(
    optimizer=Adam(learning_rate=0.00
01),
    loss="categorical_crossentropy",
    metrics=["accuracy"]
    )

EfficientNetB0_model.summary()
val_gen16.reset()

EfficientNetB0_hist =
EfficientNetB0_model.fit(
    train_gen16,
    epochs=50,
    validation_data=val_gen16,
    )
plt.figure(figsize=(14, 6))  #
Perbesar ukuran grafik
plt.suptitle("Grafik Pelatihan Model
EfficientNetB0\n(Epoch=50, Batch
Size=16)", fontsize=16)

# Subplot 1: Accuracy
plt.subplot(1, 2, 1)
plt.plot(EfficientNetB0_hist.history[
"accuracy"], label="Train Accuracy")
plt.plot(EfficientNetB0_hist.history[
"val_accuracy"], label="Validation
Accuracy")
plt.xlabel("Epoch", fontsize=12)
plt.ylabel("Accuracy", fontsize=12)
plt.legend(loc="best", fontsize=10)

# Subplot 2: Loss
plt.subplot(1, 2, 2)
plt.plot(EfficientNetB0_hist.history[
"loss"], label="Train Loss")
plt.plot(EfficientNetB0_hist.history[
"val_loss"], label="Validation Loss")
plt.xlabel("Epoch", fontsize=12)
plt.ylabel("Loss", fontsize=12)
plt.legend(loc="best", fontsize=10)

plt.savefig("/content/drive/MyDrive/h
asil-training/EfficientNetB0
Training/EfficientNetB0 (epoch=50,
batch_size=16).png")
plt.show()
print('============= TEST ACC & TEST
LOSS (PREDICT) ============')
test_gen16.reset()
predictions =
EfficientNetB0_model.predict(test_gen
16, steps=len(test_gen16));

true_classes_indices =
test_gen16.classes
predicted_classes_indices =
np.argmax(predictions, axis=1)

class_labels =
list(test_gen16.class_indices.keys())

true_classes_one_hot =
to_categorical(true_classes_indices,
num_classes=num_classes)
test_acc_predict_manual =
accuracy_score(true_classes_indices,
predicted_classes_indices)
cce = CategoricalCrossentropy()
```

```
test_loss_predict_manual =
cce(true_classes_one_hot,
predictions).numpy()

print(f"Test Loss (predict_manual)\t:
{test_loss_predict_manual:.4f}")
print(f"Test Accuracy
(predict_manual)\t:
{test_acc_predict_manual:.4f}")

print("\n============ TEST ACC &
TEST LOSS (EVALUATE) ============")
test_gen16.reset()
test_loss, test_acc =
EfficientNetB0_model.evaluate(test_ge
n16);
print(f"Test Loss\t:
{test_loss:.4f}")
print(f"Test Accuracy\t:
{test_acc:.4f}")
cm =
confusion_matrix(true_classes_indices
, predicted_classes_indices)
print("\n============== HASIL
EVALUASI MODEL ==============\n")

report =
classification_report(true_classes_in
dices, predicted_classes_indices,
target_names=class_labels)
print(report)

accuracy =
accuracy_score(true_classes_indices,
predicted_classes_indices)
print(f"Akurasi Keseluruhan:
{accuracy:.4f} ({accuracy:.2%})")
print("\n============================
======================")

plt.figure(figsize=(12, 10))
sns.heatmap(cm, annot=True, fmt='d',
cmap='Blues',
            xticklabels=class_labels,
yticklabels=class_labels)

plt.title('Confusion Matrix',
fontsize=16)
plt.ylabel('Kelas Sebenarnya (True
Class)', fontsize=12)
plt.xlabel('Kelas Prediksi (Predicted
Class)', fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.yticks(rotation=0)
plt.tight_layout()
plt.savefig("/content/drive/MyDrive/h
asil-training/EfficientNetB0
Training/Confusion Matrix -
MobileNetV2 (epoch=50,
batch_size=16).png")
plt.show()

for i in range(num_classes):
    class_name = class_labels[i]

    TP = cm[i, i]
    FP = cm[:, i].sum() - TP
    FN = cm[i, :].sum() - TP
    TN = np.sum(cm) - (TP + FP + FN)

    print(f"\n--- Kelas: {class_name}
({i}) ---")
    print(f"  True Positives (TP):
{TP}")
    print(f"  False Positives (FP):
{FP}")
    print(f"  False Negatives (FN):
{FN}")
    print(f"  True Negatives (TN):
{TN}")
val_gen16.reset()

test_loss, test_acc =
EfficientNetB0_model.evaluate(val_gen
16)
print("Test loss:", test_loss)
print("Test accuracy:", test_acc)

test_images, test_labels =
next(val_gen16)
pred_probs =
EfficientNetB0_model.predict(val_gen1
6)
pred_labels = np.argmax(pred_probs,
axis=1)
true_labels = np.argmax(test_labels,
axis=1)
training_results = {
    "epoch": list(range(1,
len(EfficientNetB0_hist.history["loss
"]) + 1)),
    "accuracy":
EfficientNetB0_hist.history["accuracy
"],
    "val_accuracy":
EfficientNetB0_hist.history["val_accu
racy"],
    "loss":
EfficientNetB0_hist.history["loss"],
    "val_loss":
EfficientNetB0_hist.history["val_loss
"],
}

testing_result = {
    "accuracy":
EfficientNetB0_hist.history["accuracy
"][-1],
    "val_accuracy":
EfficientNetB0_hist.history["val_accu
racy"][-1],
    "loss":
EfficientNetB0_hist.history["loss"][-
1],
    "val_loss":
EfficientNetB0_hist.history["val_loss
"][-1],
    "test_accuracy": test_acc,
    "test_loss": test_loss
}

# Simpan ke file JSON
with
open("/content/drive/MyDrive/hasil-
training/EfficientNetB0
Training/EfficientNetB0 (epoch=50,
batch_size=16).json", "w") as f:
    json.dump(training_results, f,
indent=4)
```

```
with
open("/content/drive/MyDrive/hasil-
training/EfficientNetB0
Training/testing - EfficientNetB0
(epoch=50, batch_size=16).json", "w")
as f:
    json.dump(testing_result, f,
indent=4)

print("Data training berhasil
disimpan ke EfficientNetB0 (epoch=50,
batch_size=16).json")
print("Data testing berhasil disimpan
ke testing - EfficientNetB0
(epoch=50, batch_size=16).json")
EfficientNetB0_model.save("/content/d
rive/MyDrive/hasil-
training/EfficientNetB0
Training/EfficientNetB0 (epoch=50,
batch_size=16).keras");
EfficientNetB0_model =
tf.keras.models.load_model("/content/
drive/MyDrive/hasil-
training/EfficientNetB0
Training/EfficientNetB0 (epoch=50,
batch_size=16).keras");

converter_keras =
tf.lite.TFLiteConverter.from_keras_mo
del(EfficientNetB0_model)

tflite_model =
converter_keras.convert()
with
open('/content/drive/MyDrive/hasil-
training/EfficientNetB0
Training/EfficientNetB0 (epoch=50,
batch_size=16).tflite', 'wb') as f:
    f.write(tflite_model);
base_model =
EfficientNetB0(input_shape=(224, 224,
3),
                               include
_top=False,
                               weights
='imagenet',
                               )

base_model.trainable = False

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(32, activation='relu')(x)
x = Dropout(0.2)(x)
outputs = layers.Dense(num_classes,
activation="softmax")(x)

EfficientNetB0_model = keras.Model(
    inputs=base_model.input,
    outputs=outputs,
    name="EfficientNetB0_epoch50_batc
h32"
)
EfficientNetB0_model.compile(
    optimizer=Adam(learning_rate=0.00
01),
    loss="categorical_crossentropy",
    metrics=["accuracy"]
    )

EfficientNetB0_model.summary()

val_gen32.reset()

EfficientNetB0_hist =
EfficientNetB0_model.fit(
    train_gen32,
    epochs=50,
    validation_data=val_gen32,
    )
plt.figure(figsize=(14, 6))  #
Perbesar ukuran grafik
plt.suptitle("Grafik Pelatihan Model
EfficientNetB0\n(Epoch=50, Batch
Size=32)", fontsize=16)

# Subplot 1: Accuracy
plt.subplot(1, 2, 1)
plt.plot(EfficientNetB0_hist.history[
"accuracy"], label="Train Accuracy")
plt.plot(EfficientNetB0_hist.history[
"val_accuracy"], label="Validation
Accuracy")
plt.xlabel("Epoch", fontsize=12)
plt.ylabel("Accuracy", fontsize=12)
plt.legend(loc="best", fontsize=10)

# Subplot 2: Loss
plt.subplot(1, 2, 2)
plt.plot(EfficientNetB0_hist.history[
"loss"], label="Train Loss")
plt.plot(EfficientNetB0_hist.history[
"val_loss"], label="Validation Loss")
plt.xlabel("Epoch", fontsize=12)
plt.ylabel("Loss", fontsize=12)
plt.legend(loc="best", fontsize=10)

plt.savefig("/content/drive/MyDrive/h
asil-training/EfficientNetB0
Training/EfficientNetB0 (epoch=50,
batch_size=32).png")
plt.show()
print('============= TEST ACC & TEST
LOSS (PREDICT) ============')
test_gen32.reset()
predictions =
EfficientNetB0_model.predict(test_gen
32, steps=len(test_gen32));

true_classes_indices =
test_gen32.classes
predicted_classes_indices =
np.argmax(predictions, axis=1)

class_labels =
list(test_gen32.class_indices.keys())

true_classes_one_hot =
to_categorical(true_classes_indices,
num_classes=num_classes)
test_acc_predict_manual =
accuracy_score(true_classes_indices,
predicted_classes_indices)
cce = CategoricalCrossentropy()
test_loss_predict_manual =
cce(true_classes_one_hot,
predictions).numpy()

print(f"Test Loss (predict_manual)\t:
{test_loss_predict_manual:.4f}")
print(f"Test Accuracy
(predict_manual)\t:
{test_acc_predict_manual:.4f}")
```

```
print("\n============= TEST ACC &
TEST LOSS (EVALUATE) ============")
test_gen32.reset()
test_loss, test_acc =
EfficientNetB0_model.evaluate(test_ge
n32);
print(f"Test Loss\t:
{test_loss:.4f}")
print(f"Test Accuracy\t:
{test_acc:.4f}")
cm =
confusion_matrix(true_classes_indices
, predicted_classes_indices)
print("\n============== HASIL
EVALUASI MODEL ==============\n")

report =
classification_report(true_classes_in
dices, predicted_classes_indices,
target_names=class_labels)
print(report)

accuracy =
accuracy_score(true_classes_indices,
predicted_classes_indices)
print(f"Akurasi Keseluruhan:
{accuracy:.4f} ({accuracy:.2%})")
print("\n============================
======================")

plt.figure(figsize=(12, 10))
sns.heatmap(cm, annot=True, fmt='d',
cmap='Blues',
            xticklabels=class_labels,
yticklabels=class_labels)

plt.title('Confusion Matrix',
fontsize=16)
plt.ylabel('Kelas Sebenarnya (True
Class)', fontsize=12)
plt.xlabel('Kelas Prediksi (Predicted
Class)', fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.yticks(rotation=0)
plt.tight_layout()
plt.savefig("/content/drive/MyDrive/h
asil-training/EfficientNetB0
Training/Confusion Matrix -
MobileNetV2 (epoch=50,
batch_size=32).png")
plt.show()

for i in range(num_classes):
    class_name = class_labels[i]

    TP = cm[i, i]
    FP = cm[:, i].sum() - TP
    FN = cm[i, :].sum() - TP
    TN = np.sum(cm) - (TP + FP + FN)

    print(f"\n--- Kelas: {class_name}
({i}) ---")
    print(f"  True Positives (TP):
{TP}")
    print(f"  False Positives (FP):
{FP}")
    print(f"  False Negatives (FN):
{FN}")
    print(f"  True Negatives (TN):
{TN}")
```

```
val_gen32.reset()

test_loss, test_acc =
EfficientNetB0_model.evaluate(val_gen
32)
print("Test loss:", test_loss)
print("Test accuracy:", test_acc)

test_images, test_labels =
next(val_gen32)
pred_probs =
EfficientNetB0_model.predict(val_gen3
2)
pred_labels = np.argmax(pred_probs,
axis=1)
true_labels = np.argmax(test_labels,
axis=1)
training_results = {
    "epoch": list(range(1,
len(EfficientNetB0_hist.history["loss
"]) + 1)),
    "accuracy":
EfficientNetB0_hist.history["accuracy
"],
    "val_accuracy":
EfficientNetB0_hist.history["val_accu
racy"],
    "loss":
EfficientNetB0_hist.history["loss"],
    "val_loss":
EfficientNetB0_hist.history["val_loss
"],
}

testing_result = {
    "accuracy":
EfficientNetB0_hist.history["accuracy
"][-1],
    "val_accuracy":
EfficientNetB0_hist.history["val_accu
racy"][-1],
    "loss":
EfficientNetB0_hist.history["loss"][-
1],
    "val_loss":
EfficientNetB0_hist.history["val_loss
"][-1],
    "test_accuracy": test_acc,
    "test_loss": test_loss
}

# Simpan ke file JSON
with
open("/content/drive/MyDrive/hasil-
training/EfficientNetB0
Training/EfficientNetB0 (epoch=50,
batch_size=32).json", "w") as f:
    json.dump(training_results, f,
indent=4)

with
open("/content/drive/MyDrive/hasil-
training/EfficientNetB0
Training/testing - EfficientNetB0
(epoch=50, batch_size=32).json", "w")
as f:
    json.dump(testing_result, f,
indent=4)
```

```
print("Data training berhasil
disimpan ke EfficientNetB0 (epoch=50,
batch_size=32).json")
print("Data testing berhasil disimpan
ke testing - EfficientNetB0
(epoch=50, batch_size=32).json")
EfficientNetB0_model.save("/content/d
rive/MyDrive/hasil-
training/EfficientNetB0
Training/EfficientNetB0 (epoch=50,
batch_size=32).keras");
EfficientNetB0_model =
tf.keras.models.load_model("/content/
drive/MyDrive/hasil-
training/EfficientNetB0
Training/EfficientNetB0 (epoch=50,
batch_size=32).keras");

converter_keras =
tf.lite.TFLiteConverter.from_keras_mo
del(EfficientNetB0_model)

tflite_model =
converter_keras.convert()
with
open('/content/drive/MyDrive/hasil-
training/EfficientNetB0
Training/EfficientNetB0 (epoch=50,
batch_size=32).tflite', 'wb') as f:
    f.write(tflite_model);
base_model =
EfficientNetB0(input_shape=(224, 224,
3),
                               include
_top=False,
                               weights
='imagenet',
                               )

base_model.trainable = False

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(32, activation='relu')(x)
x = Dropout(0.2)(x)
outputs = layers.Dense(num_classes,
activation="softmax")(x)

EfficientNetB0_model = keras.Model(
    inputs=base_model.input,
    outputs=outputs,
    name="EfficientNetB0_epoch50_batc
h64"
)
EfficientNetB0_model.compile(
    optimizer=Adam(learning_rate=0.00
01),
    loss="categorical_crossentropy",
    metrics=["accuracy"]
    )

EfficientNetB0_model.summary()
val_gen64.reset()

EfficientNetB0_hist =
EfficientNetB0_model.fit(
    train_gen64,
    epochs=50,
    validation_data=val_gen64,
    )

plt.figure(figsize=(14, 6))  #
Perbesar ukuran grafik
plt.suptitle("Grafik Pelatihan Model
EfficientNetB0\n(Epoch=50, Batch
Size=64)", fontsize=16)

# Subplot 1: Accuracy
plt.subplot(1, 2, 1)
plt.plot(EfficientNetB0_hist.history[
"accuracy"], label="Train Accuracy")
plt.plot(EfficientNetB0_hist.history[
"val_accuracy"], label="Validation
Accuracy")
plt.xlabel("Epoch", fontsize=12)
plt.ylabel("Accuracy", fontsize=12)
plt.legend(loc="best", fontsize=10)

# Subplot 2: Loss
plt.subplot(1, 2, 2)
plt.plot(EfficientNetB0_hist.history[
"loss"], label="Train Loss")
plt.plot(EfficientNetB0_hist.history[
"val_loss"], label="Validation Loss")
plt.xlabel("Epoch", fontsize=12)
plt.ylabel("Loss", fontsize=12)
plt.legend(loc="best", fontsize=10)

plt.savefig("/content/drive/MyDrive/h
asil-training/EfficientNetB0
Training/EfficientNetB0 (epoch=50,
batch_size=64).png")
plt.show()
print('============= TEST ACC & TEST
LOSS (PREDICT) ============')
test_gen64.reset()
predictions =
EfficientNetB0_model.predict(test_gen
64, steps=len(test_gen64));

true_classes_indices =
test_gen64.classes
predicted_classes_indices =
np.argmax(predictions, axis=1)

class_labels =
list(test_gen64.class_indices.keys())

true_classes_one_hot =
to_categorical(true_classes_indices,
num_classes=num_classes)
test_acc_predict_manual =
accuracy_score(true_classes_indices,
predicted_classes_indices)
cce = CategoricalCrossentropy()
test_loss_predict_manual =
cce(true_classes_one_hot,
predictions).numpy()

print(f"Test Loss (predict_manual)\t:
{test_loss_predict_manual:.4f}")
print(f"Test Accuracy
(predict_manual)\t:
{test_acc_predict_manual:.4f}")

print("\n============= TEST ACC &
TEST LOSS (EVALUATE) ============")
test_gen64.reset()
test_loss, test_acc =
EfficientNetB0_model.evaluate(test_ge
n64);
```

```
print(f"Test Loss\t:
{test_loss:.4f}")
print(f"Test Accuracy\t:
{test_acc:.4f}")
cm =
confusion_matrix(true_classes_indices
, predicted_classes_indices)
print("\n============== HASIL
EVALUASI MODEL ==============\n")

report =
classification_report(true_classes_in
dices, predicted_classes_indices,
target_names=class_labels)
print(report)

accuracy =
accuracy_score(true_classes_indices,
predicted_classes_indices)
print(f"Akurasi Keseluruhan:
{accuracy:.4f} ({accuracy:.2%})")
print("\n============================
======================")

plt.figure(figsize=(12, 10))
sns.heatmap(cm, annot=True, fmt='d',
cmap='Blues',
            xticklabels=class_labels,
yticklabels=class_labels)

plt.title('Confusion Matrix',
fontsize=16)
plt.ylabel('Kelas Sebenarnya (True
Class)', fontsize=12)
plt.xlabel('Kelas Prediksi (Predicted
Class)', fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.yticks(rotation=0)
plt.tight_layout()
plt.savefig("/content/drive/MyDrive/h
asil-training/EfficientNetB0
Training/Confusion Matrix -
MobileNetV2 (epoch=50,
batch_size=64).png")
plt.show()

for i in range(num_classes):
    class_name = class_labels[i]

    TP = cm[i, i]
    FP = cm[:, i].sum() - TP
    FN = cm[i, :].sum() - TP
    TN = np.sum(cm) - (TP + FP + FN)

    print(f"\n--- Kelas: {class_name}
({i}) ---")
    print(f"  True Positives (TP):
{TP}")
    print(f"  False Positives (FP):
{FP}")
    print(f"  False Negatives (FN):
{FN}")
    print(f"  True Negatives (TN):
{TN}")
val_gen64.reset()

test_loss, test_acc =
EfficientNetB0_model.evaluate(val_gen
64)
print("Test loss:", test_loss)
print("Test accuracy:", test_acc)

test_images, test_labels =
next(val_gen64)
pred_probs =
EfficientNetB0_model.predict(val_gen6
4)
pred_labels = np.argmax(pred_probs,
axis=1)
true_labels = np.argmax(test_labels,
axis=1)
training_results = {
    "epoch": list(range(1,
len(EfficientNetB0_hist.history["loss
"]) + 1)),
    "accuracy":
EfficientNetB0_hist.history["accuracy
"],
    "val_accuracy":
EfficientNetB0_hist.history["val_accu
racy"],
    "loss":
EfficientNetB0_hist.history["loss"],
    "val_loss":
EfficientNetB0_hist.history["val_loss
"],
}

testing_result = {
    "accuracy":
EfficientNetB0_hist.history["accuracy
"][-1],
    "val_accuracy":
EfficientNetB0_hist.history["val_accu
racy"][-1],
    "loss":
EfficientNetB0_hist.history["loss"][-
1],
    "val_loss":
EfficientNetB0_hist.history["val_loss
"][-1],
    "test_accuracy": test_acc,
    "test_loss": test_loss
}

# Simpan ke file JSON
with
open("/content/drive/MyDrive/hasil-
training/EfficientNetB0
Training/EfficientNetB0 (epoch=50,
batch_size=64).json", "w") as f:
    json.dump(training_results, f,
indent=4)

with
open("/content/drive/MyDrive/hasil-
training/EfficientNetB0
Training/testing - EfficientNetB0
(epoch=50, batch_size=64).json", "w")
as f:
    json.dump(testing_result, f,
indent=4)

print("Data training berhasil
disimpan ke EfficientNetB0 (epoch=50,
batch_size=64).json")
print("Data testing berhasil disimpan
ke testing - EfficientNetB0
(epoch=50, batch_size=64).json")
EfficientNetB0_model.save("/content/d
rive/MyDrive/hasil-
training/EfficientNetB0
```

```
Training/EfficientNetB0 (epoch=50,
batch_size=64).keras");
EfficientNetB0_model =
tf.keras.models.load_model("/content/
drive/MyDrive/hasil-
training/EfficientNetB0
Training/EfficientNetB0 (epoch=50,
batch_size=64).keras");

converter_keras =
tf.lite.TFLiteConverter.from_keras_mo
del(EfficientNetB0_model)

tflite_model =
converter_keras.convert()
with
open('/content/drive/MyDrive/hasil-
training/EfficientNetB0
Training/EfficientNetB0 (epoch=50,
batch_size=64).tflite', 'wb') as f:
    f.write(tflite_model);
import os
import json
import pandas as pd
import glob
from tabulate import tabulate

# Path menuju direktori file JSON
json_dir =
"/content/drive/MyDrive/hasil-
training/EfficientNetB0 Training/"

# Mencari semua file JSON dalam
direktori
json_files =
glob.glob(os.path.join(json_dir,
"*.json"))

# List untuk menyimpan hasil
results = []

# Memproses setiap file JSON
for json_file in json_files:
    # Mengambil nama file untuk
ekstrak parameter
    filename =
os.path.basename(json_file)

    # Ekstrak epoch dan batch_size
dari nama file
    # Asumsi format file:
"EfficientNetB0 (epoch=X,
batch_size=Y).json"
    try:
        params =
filename.split("(")[1].split(")")[0]
        epoch_str =
params.split(",")[0].strip()
        batch_size_str =
params.split(",")[1].strip()

        epoch_val =
int(epoch_str.split("=")[1])
        batch_size_val =
int(batch_size_str.split("=")[1])

        # Baca file JSON
        with open(json_file, 'r') as
f:
            data = json.load(f)

        # Ambil nilai pada epoch
terakhir
        last_epoch_idx = -1  # Indeks
untuk epoch terakhir
        accuracy =
data['accuracy'][last_epoch_idx]
        val_accuracy =
data['val_accuracy'][last_epoch_idx]
        loss =
data['loss'][last_epoch_idx]
        val_loss =
data['val_loss'][last_epoch_idx]

        # Tambahkan hasil ke list
        results.append({
            'Epoch': epoch_val,
            'Batch Size':
batch_size_val,
            'Accuracy': accuracy,
            'Val Accuracy':
val_accuracy,
            'Loss': loss,
            'Val Loss': val_loss
        })
    except Exception as e:
        print(f"Error memproses file
{filename}: {e}")

# Buat DataFrame dan sort berdasarkan
epoch dan batch size
df = pd.DataFrame(results)
df = df.sort_values(by=['Epoch',
'Batch Size'])

# Tampilkan tabel hasil
print("\nHasil Pelatihan
EfficientNetB0 (Epoch Terakhir):")
print(tabulate(df, headers='keys',
tablefmt='grid', floatfmt='.4f'))

# Jika ingin menyimpan hasil ke CSV
df.to_csv(os.path.join(json_dir,
"efficientnet_results_summary.csv"),
index=False)
print(f"\nHasil juga disimpan dalam
file CSV: {os.path.join(json_dir,
'efficientnet_results_summary.csv')}"
)
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-
8"?>
<manifest
xmlns:android="http://schemas.android
.com/apk/res/android"

xmlns:tools="http://schemas.android.c
om/tools"

package="com.rifaldi.myapplication">

    <uses-sdk />

    <uses-feature
android:name="android.hardware.camera
.autofocus" />
    <uses-feature
android:name="android.hardware.camera
" />
    <uses-feature
```

```
android:name="android.hardware.camera
.ar"
        android:required="false" />

    <uses-permission
android:name="android.permission.CAME
RA" />
    <uses-permission
android:name="android.permission.INTE
RNET" />
    <uses-permission
android:name="android.permission.READ
_EXTERNAL_STORAGE" />
    <uses-permission
android:name="android.permission.WRIT
E_EXTERNAL_STORAGE" />

    <application
        android:allowBackup="true"

android:dataExtractionRules="@xml/dat
a_extraction_rules"

android:fullBackupContent="@xml/backu
p_rules"

android:icon="@mipmap/my_aplication_i
con"

android:label="@string/app_name"

android:requestLegacyExternalStorage=
"true"

android:roundIcon="@mipmap/ic_launche
r_round"
        android:supportsRtl="true"

android:theme="@style/Theme.Design.No
ActionBar"
        tools:targetApi="31">
        <activity

android:name=".ui.AveragingDetail"
            android:exported="false"
/>
        <activity

android:name=".ui.SplashActivity"
            android:exported="true">
            <intent-filter>
                <action
android:name="android.intent.action.M
AIN" />

                <category
android:name="android.intent.category
.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity

android:name=".ui.DetectionPage"
            android:exported="false"
/>
        <activity

android:name=".ui.DetectionResult"
            android:exported="false"
/>
        <activity

android:name=".ui.CakeDescription"
            android:exported="false"
/>
        <activity

android:name=".ui.About_dev"
            android:exported="false"
/>
        <activity

android:name=".ui.DevProfile"
            android:exported="false"
/>
        <activity

android:name=".ui.Catalog"
            android:exported="false"
/>
        <activity
            android:name=".ui.About"
            android:exported="false"

android:theme="@style/Theme.MyApplica
tion" />
        <activity

android:name=".ui.AppLayout"
            android:exported="false"
/>

        <provider

android:name="androidx.core.content.F
ileProvider"

android:authorities="${applicationId}
.fileprovider"
            android:exported="false"

android:grantUriPermissions="true">
            <meta-data

android:name="android.support.FILE_PR
OVIDER_PATHS"

android:resource="@xml/file_paths" />
        </provider>

        <meta-data

android:name="com.google.ar.core"
            android:value="required"
/>
    </application>

</manifest>
```

CakeAdapter.java

```
package
com.rifaldi.myapplication.adapter;

import
android.annotation.SuppressLint;
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import
android.graphics.drawable.ColorDrawab
le;
import
```

```
android.graphics.drawable.Drawable;
import android.util.Log;
import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import
android.view.animation.Animation;
import
android.view.animation.AnimationUtils
;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.PopupWindow;
import android.widget.TextView;
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import
androidx.recyclerview.widget.Recycler
View;

import com.bumptech.glide.Glide;
import
com.bumptech.glide.load.DataSource;
import
com.bumptech.glide.load.engine.GlideE
xception;
import
com.bumptech.glide.request.RequestLis
tener;
import
com.bumptech.glide.request.target.Tar
get;
import com.google.gson.Gson;
import com.rifaldi.myapplication.R;
import
com.rifaldi.myapplication.model.Cake;
import
com.rifaldi.myapplication.ui.CakeDesc
ription;
import java.util.List;

public class CakeAdapter extends
RecyclerView.Adapter<CakeAdapter.Cake
ViewHolder> {
    private final Context context;
    private List<Cake> cakeList;

    public CakeAdapter(Context
context, List<Cake> cakeList) {
        this.context = context;
        this.cakeList = cakeList;
    }


@SuppressLint("NotifyDataSetChanged")
    public void setData(List<Cake>
newList){
        this.cakeList = newList;
        notifyDataSetChanged();
    }

    @NonNull
    @Override
    public CakeViewHolder
onCreateViewHolder(@NonNull ViewGroup
parent, int viewType) {
        View view =
LayoutInflater.from(context).inflate(
R.layout.item_cake, parent, false);
        return new
CakeViewHolder(view);
    }

    @Override
    public void
onBindViewHolder(@NonNull
CakeViewHolder holder, int position)
{
        Cake cake =
cakeList.get(position);
        if (cake != null) {

holder.tvCakeName.setText(cake.getCak
eName());

holder.tvShortDescription.setText(cak
e.getCakeShortDescription());
        }


holder.itemView.setOnLongClickListene
r(v -> {
            assert cake != null;

showToolTips(holder.itemView,
cake.getCakeName(),
cake.getCakeDescription(),
cake.getCakeImageUrl());
            return true; //
Menandakan event long-click telah
ditangani
        });


holder.itemView.setOnClickListener(v
-> {
            Intent intent = new
Intent(holder.itemView.getContext(),
CakeDescription.class);

            // Pass some data to next
activity
            assert cake != null;

intent.putExtra("cakeName",cake.getCa
keName());

intent.putExtra("cakeDescription",cak
e.getCakeDescription());

intent.putExtra("cakeImageUrl",
cake.getCakeImageUrl());

            // Using GSON for pass
data
            Gson gson = new Gson();
            String recipesJson =
gson.toJson(cake.getRecipes());
            String tipsJson =
gson.toJson(cake.getTips());
            String referencesJson =
gson.toJson(cake.getReferences());

            Log.d("CakeAdapter",
recipesJson);


intent.putExtra("cakeRecipes",
recipesJson);

intent.putExtra("cakeTips",
```

```
tipsJson);

intent.putExtra("cakeReferences",
referencesJson);


holder.itemView.getContext().startAct
ivity(intent);
        });
    }

    @Override
    public int getItemCount() {
        return cakeList != null ?
cakeList.size() : 0;
    }

    public static class
CakeViewHolder extends
RecyclerView.ViewHolder {
        TextView tvCakeName,
tvShortDescription;

        public
CakeViewHolder(@NonNull View
itemView) {
            super(itemView);
            tvCakeName =
itemView.findViewById(R.id.tv_cake_na
me);
            tvShortDescription =
itemView.findViewById(R.id.tv_cake_sh
ort_description);
        }
    }


@SuppressLint("ClickableViewAccessibi
lity")
    private void showToolTips(View
view, String toolTipHeader, String
toolTipText, String toolTipImageUrl)
{
        LayoutInflater inflater =
LayoutInflater.from(context);

@SuppressLint("InflateParams") View
popupView =
inflater.inflate(R.layout.popup_toolt
ip_catalog, null);
        popupView.setElevation(12);

        TextView tooltipHeader =
popupView.findViewById(R.id.tooltipCa
keHeader);
        TextView tvTooltipText =
popupView.findViewById(R.id.tooltipCa
ke_txt);
        ImageView tooltipImage =
popupView.findViewById(R.id.tooltipCa
keImage);

        Log.d("CakeAdapter",
toolTipImageUrl);


tvTooltipText.setText(toolTipText);

tooltipHeader.setText(toolTipHeader);

        Glide.with(view.getContext())
.load(toolTipImageUrl)

.error(R.drawable.button_black_about_
icon)
                .listener(new
RequestListener<Drawable>() {
                    @Override
                    public boolean
onLoadFailed(@Nullable GlideException
e, Object model, Target<Drawable>
target, boolean isFirstResource) {
                        assert e !=
null;

Log.e("CakeAdapter", "Error loading
image: " + e.getMessage());
                        return false;
                    }

                    @Override
                    public boolean
onResourceReady(Drawable resource,
Object model, Target<Drawable>
target, DataSource dataSource,
boolean isFirstResource) {

Log.d("CakeAdapter", "Image loaded
successfully");
                        return false;
                    }
                })
                .into(tooltipImage);

        final PopupWindow popupWindow
= new PopupWindow(popupView,

LinearLayout.LayoutParams.WRAP_CONTEN
T,

LinearLayout.LayoutParams.WRAP_CONTEN
T);


popupWindow.setFocusable(true);

popupWindow.setOutsideTouchable(true)
;

popupWindow.setBackgroundDrawable(new
ColorDrawable());

        // Pastikan vOverlay sudah
ada dalam layout activity atau
fragment Anda
        View vOverlay = ((Activity)
context).findViewById(R.id.overlayCat
alog);

vOverlay.setVisibility(View.VISIBLE);


popupWindow.setOnDismissListener(() -
> {
            Animation fadeOut =
AnimationUtils.loadAnimation(context,
R.anim.fade_out);

fadeOut.setAnimationListener(new
Animation.AnimationListener() {
                @Override
```

```
                public void
onAnimationStart(Animation animation)
{}

                @Override
                public void
onAnimationEnd(Animation animation) {

vOverlay.setVisibility(View.GONE);
                }

                @Override
                public void
onAnimationRepeat(Animation
animation) {}
            });

vOverlay.startAnimation(fadeOut);
        });

        Animation slideUp =
AnimationUtils.loadAnimation(context,
R.anim.fade_in);

popupView.startAnimation(slideUp);


popupWindow.showAtLocation(view.getRo
otView(), Gravity.CENTER, 0, 0);

        vOverlay.setOnClickListener(v
-> popupWindow.dismiss());

popupView.setOnTouchListener((v1,
event) -> {
            popupWindow.dismiss();
            return true;
        });
    }
}
```

TechnologyAdapter.java

```
package
com.rifaldi.myapplication.adapter;

import
android.annotation.SuppressLint;
import android.app.Activity;
import android.content.Context;
import
android.graphics.drawable.ColorDrawab
le;
import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import
android.view.animation.Animation;
import
android.view.animation.AnimationUtils
;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.PopupWindow;
import android.widget.TextView;
import androidx.annotation.NonNull;
import
androidx.recyclerview.widget.Recycler
View;

import com.bumptech.glide.Glide;
import
com.bumptech.glide.load.engine.DiskCa
cheStrategy;
import com.rifaldi.myapplication.R;
import
com.rifaldi.myapplication.model.Techn
ology;

import java.util.List;

public class TechnologyAdapter
extends
RecyclerView.Adapter<TechnologyAdapte
r.ViewHolder> {
    private final List<Technology>
technologies;
    private final Context context;

    public TechnologyAdapter(Context
context, List<Technology>
technologies) {
        this.technologies =
technologies;
        this.context = context;
    }

    @NonNull
    @Override
    public ViewHolder
onCreateViewHolder(ViewGroup parent,
int viewType) {
        View view =
LayoutInflater.from(parent.getContext
()).inflate(R.layout.item_technology,
parent, false);
        return new ViewHolder(view);
    }

    @Override
    public void
onBindViewHolder(ViewHolder holder,
int position) {
        Technology technology =
technologies.get(position);

Glide.with(holder.itemView.getContext
())

.load(technology.getIconUrl())

.placeholder(R.drawable.button_black_
about_icon)

.error(R.drawable.button_black_back_i
con)

.diskCacheStrategy(DiskCacheStrategy.
ALL)

.into(holder.imageView);


holder.imageView.setContentDescriptio
n(technology.getName());


holder.itemView.setOnClickListener(v
-> showToolTips(holder.itemView,
technology.getShortDescription(),
technology.getName()));
```

```
    }

    @Override
    public int getItemCount() {
        return technologies.size();
    }

    public static class ViewHolder
extends RecyclerView.ViewHolder {
        ImageView imageView;

        public ViewHolder(View
itemView) {
            super(itemView);
            imageView =
itemView.findViewById(R.id.technology
_icon);
        }
    }


@SuppressLint("ClickableViewAccessibi
lity")
    private void showToolTips(View
view, String toolTipText, String
toolTipHeader) {
        LayoutInflater inflater =
LayoutInflater.from(context);
        ViewGroup parent =
view.findViewById(android.R.id.conten
t);
        View popupView =
inflater.inflate(R.layout.popup_toolt
ip_technologies, parent, false);
        popupView.setElevation(12);

        TextView tooltipView =
popupView.findViewById(R.id.tooltipTe
chnology_text);
        TextView tooltipHeader =
popupView.findViewById(R.id.tooltipTe
chnology_header);

tooltipView.setText(toolTipText);

tooltipHeader.setText(toolTipHeader);

        final PopupWindow popupWindow
= new PopupWindow(popupView,

LinearLayout.LayoutParams.WRAP_CONTEN
T,

LinearLayout.LayoutParams.WRAP_CONTEN
T);


popupWindow.setFocusable(true);

popupWindow.setOutsideTouchable(true)
;

popupWindow.setBackgroundDrawable(new
ColorDrawable());


        View vOverlay = ((Activity)
context).findViewById(R.id.vOverlay);

vOverlay.setVisibility(View.VISIBLE);

popupWindow.setOnDismissListener(() -
> {
            Animation fadeOut =
AnimationUtils.loadAnimation(context,
R.anim.fade_out);

fadeOut.setAnimationListener(new
Animation.AnimationListener() {
                @Override
                public void
onAnimationStart(Animation animation)
{}

                @Override
                public void
onAnimationEnd(Animation animation) {

vOverlay.setVisibility(View.GONE);
                }

                @Override
                public void
onAnimationRepeat(Animation
animation) {}
            });

vOverlay.startAnimation(fadeOut);
        });

        Animation slideUp =
AnimationUtils.loadAnimation(context,
R.anim.fade_in);

popupView.startAnimation(slideUp);


popupWindow.showAtLocation(view.getRo
otView(), Gravity.CENTER, 0, 0);

        vOverlay.setOnClickListener(v
-> popupWindow.dismiss());

popupView.setOnTouchListener((v1,
event) -> {
            popupWindow.dismiss();
            return true;
        });
    }


}
```

Cake.java

```
package
com.rifaldi.myapplication.model;

import java.util.List;
import java.util.Map;

public class Cake {
    private String cakeName,
cakeShortDescription,
cakeDescription, cakeImageUrl;
    private List<String> references;
    private List<String> tips;
    private Map<String, Object>
recipes;

    // Constructor
    public Cake() {
```

```
        // Needed for Firebase
    }

    public Cake(String cakeName,
String cakeShortDescription, String
cakeDescription, String cakeImageUrl,
List<String> references, List<String>
tips, Map<String, Object> recipes) {
        this.cakeName = cakeName;
        this.cakeShortDescription =
cakeShortDescription;
        this.cakeDescription =
cakeDescription;
        this.cakeImageUrl =
cakeImageUrl;
        this.references = references;
        this.tips = tips;
        this.recipes = recipes;
    }

    // Getters and Setters
    public String getCakeName() {
        return cakeName;
    }

    public void setCakeName(String
cakeName) {
        this.cakeName = cakeName;
    }

    public String
getCakeShortDescription() {
        return cakeShortDescription;
    }

    public void
setCakeShortDescription(String
cakeShortDescription) {
        this.cakeShortDescription =
cakeShortDescription;
    }

    public String
getCakeDescription() {
        return cakeDescription;
    }

    public void
setCakeDescription(String
cakeDescription) {
        this.cakeDescription =
cakeDescription;
    }

    public String getCakeImageUrl(){
        return cakeImageUrl;
    }

    public void
setCakeImageUrl(String cakeImageUrl){
        this.cakeImageUrl =
cakeImageUrl;
    }

    public List<String>
getReferences(){
        return references;
    }

    public void
setReferences(List<String>
references){
        this.references = references;
    }

    public List<String> getTips(){
        return tips;
    }

    public void setTips(List<String>
tips){
        this.tips = tips;
    }

    public Map<String, Object>
getRecipes(){
        return recipes;
    }

    public void
setRecipes(Map<String, Object>
recipes){
        this.recipes = recipes;
    }
}
```

CookingStep.java

```
package
com.rifaldi.myapplication.model;

import
com.google.gson.annotations.Serialize
dName;

import java.util.List;

public class CookingStep {
    @SerializedName("nama")
    private String nama;

    @SerializedName("langkah")
    private List<String> langkah;

    public CookingStep() {}

    public String getNama() {
        return nama;
    }

    public void setNama(String nama)
{
        this.nama = nama;
    }

    public List<String> getLangkah()
{
        return langkah;
    }

    public void
setLangkah(List<String> langkah) {
        this.langkah = langkah;
    }
}
```

Model.java

```
package
com.rifaldi.myapplication.model;

public class Model {
    private String name;
    private String val_accuracy;
```

```
    private String val_loss;

    // Constructor kosong diperlukan
oleh Firebase
    public Model() {
    }

    public Model(String name, String
val_accuracy, String val_loss) {
        this.name = name;
        this.val_accuracy =
val_accuracy;
        this.val_loss = val_loss;
    }

    // Getter dan Setter
    public String getName() {
        return name;
    }

    public void setName(String name)
{
        this.name = name;
    }

    public String getVal_accuracy() {
        return val_accuracy;
    }

    public void
setVal_accuracy(String val_accuracy)
{
        this.val_accuracy =
val_accuracy;
    }

    public String getVal_loss() {
        return val_loss;
    }

    public void setVal_loss(String
val_loss) {
        this.val_loss = val_loss;
    }
}
```

Recipe.java

```
package
com.rifaldi.myapplication.model;

import
com.google.gson.annotations.Serialize
dName;

import java.util.List;

public class Recipe {
    @SerializedName("bahan_bahan")
    private List<String> bahanBahan;


@SerializedName("bahan_saus_santan")
    private List<String>
bahanSausSantan;

    @SerializedName("cara_membuat")
    private List<CookingStep>
caraMembuat;

    public Recipe() {}

    public List<String>
getBahanBahan() {
        return bahanBahan;
    }

    public void
setBahanBahan(List<String>
bahanBahan) {
        this.bahanBahan = bahanBahan;
    }

    public List<String>
getBahanSausSantan() {
        return bahanSausSantan;
    }

    public void
setBahanSausSantan(List<String>
bahanSausSantan) {
        this.bahanSausSantan =
bahanSausSantan;
    }

    public List<CookingStep>
getCaraMembuat() {
        return caraMembuat;
    }

    public void
setCaraMembuat(List<CookingStep>
caraMembuat) {
        this.caraMembuat =
caraMembuat;
    }
}
```

Technology.java

```
package
com.rifaldi.myapplication.model;

public class Technology {
    private String name;
    private String iconUrl;
    private String shortDescription;

    public Technology(){

    }

    public Technology(String name,
String iconUrl, String
shortDescription) {
        this.name = name;
        this.iconUrl = iconUrl;
        this.shortDescription =
shortDescription;
    }

    public String getName() {
        return name;
    }

    public String getIconUrl() {
        return iconUrl;
    }

    public String
getShortDescription(){
        return shortDescription;
    }
}
```

About.java

```
package com.rifaldi.myapplication.ui;

import androidx.annotation.NonNull;
import
androidx.appcompat.app.AppCompatActiv
ity;
import
androidx.core.content.ContextCompat;
import
androidx.recyclerview.widget.LinearLa
youtManager;
import
androidx.recyclerview.widget.Recycler
View;

import
android.annotation.SuppressLint;
import android.os.Bundle;
import android.util.Log;
import android.view.Window;
import android.view.WindowManager;
import android.widget.Toast;

import
com.google.firebase.database.DataSnap
shot;
import
com.google.firebase.database.Database
Error;
import
com.google.firebase.database.ValueEve
ntListener;
import com.rifaldi.myapplication.R;

import
com.google.firebase.database.Database
Reference;
import
com.google.firebase.database.Firebase
Database;
import
com.rifaldi.myapplication.adapter.Tec
hnologyAdapter;
import
com.rifaldi.myapplication.databinding
.ActivityAboutBinding;
import
com.rifaldi.myapplication.model.Techn
ology;
import
com.rifaldi.myapplication.util.Networ
kUtil;

import java.util.ArrayList;
import java.util.List;

public class About extends
AppCompatActivity {

    private RecyclerView
recyclerView;
    private TechnologyAdapter
adapter;
    private List<Technology>
technologyList;

    private ActivityAboutBinding
binding;

    private final DatabaseReference
mDatabaseReference = FirebaseDatabase

.getInstance("https://kuecnn-e1f7f-
default-rtdb.asia-
southeast1.firebasedatabase.app/")
            .getReference();

    @Override
    protected void onCreate(Bundle
savedInstanceState) {

super.onCreate(savedInstanceState);
        binding =
ActivityAboutBinding.inflate(getLayou
tInflater());

setContentView(binding.getRoot());

        Window window = getWindow();

window.addFlags(WindowManager.LayoutP
arams.FLAG_DRAWS_SYSTEM_BAR_BACKGROUN
DS);

window.setStatusBarColor(ContextCompa
t.getColor(this, R.color.green));

        initializeViews();


binding.ivBack.setOnClickListener(v -
> onBackPressed());

binding.sRLAbout.setOnRefreshListener
(this::retrieveData);

        initializeRecyclerView();

        if
(!NetworkUtil.isInternetAvailable(thi
s)){

FirebaseDatabase.getInstance().goOffl
ine();
            Toast.makeText(this, "NO
INTERNET CONNECTION!\nData might not
be updated!", Toast

.LENGTH_LONG)
                    .show();

            retrieveData();
        }
        retrieveData();

    }

    public void initializeViews(){
        recyclerView =
findViewById(R.id.recycler_view);

recyclerView.setLayoutManager(new
LinearLayoutManager(this,
LinearLayoutManager.HORIZONTAL,
false));
    }

    public void
initializeRecyclerView(){
        technologyList = new
```

```
ArrayList<>();
        adapter = new
TechnologyAdapter(this,
technologyList);

recyclerView.setAdapter(adapter);
    }

    private void retrieveData() {

binding.sRLAbout.setRefreshing(true);


mDatabaseReference.keepSynced(true);


mDatabaseReference.child("about").chi
ld("app_description").addListenerForS
ingleValueEvent(new
ValueEventListener() {
            @Override
            public void
onDataChange(@NonNull DataSnapshot
snapshot) {
                if
(snapshot.exists()) {

binding.tvAppDescription.setText(snap
shot.getValue(String.class));
                }else {
                    Log.e("Firebase",
"DATA (app_description) NOT FOUND!");

Toast.makeText(About.this, "DATA IS
NOT FOUND!",
Toast.LENGTH_LONG).show();
                }


binding.sRLAbout.setRefreshing(false)
;
            }

            @Override
            public void
onCancelled(@NonNull DatabaseError
error) {
                Log.e("Firebase",
"DATABASE ERROR!",
error.toException());

Toast.makeText(About.this, "ERROR TO
LOAD DATA FROM DATABASE! \n MAKE SURE
YOUR INTERNET CONNECTION IS
AVAILABLE",
Toast.LENGTH_SHORT).show();

binding.sRLAbout.setRefreshing(false)
;
            }
        });


mDatabaseReference.child("about").chi
ld("technology").addListenerForSingle
ValueEvent(new ValueEventListener() {

@SuppressLint("NotifyDataSetChanged")
            @Override
            public void
onDataChange(@NonNull DataSnapshot
snapshot) {

technologyList.clear();

                if
(snapshot.exists()) {
                    for (DataSnapshot
technologySnapshot :
snapshot.getChildren()) {
                        String name =
technologySnapshot.child("name").getV
alue(String.class);
                        String
iconUrl =
technologySnapshot.child("image_url")
.getValue(String.class);
                        String
shortDescription =
technologySnapshot.child("short_descr
iption").getValue(String.class);

                        if (name !=
null && iconUrl != null) {

Technology technology = new
Technology(name, iconUrl,
shortDescription);

technologyList.add(technology);
                        } else {

Log.e("Firebase", "DATA (technology)
NOT FOUND!");

Toast.makeText(About.this, "DATA IS
NOT FOUND!",
Toast.LENGTH_LONG).show();
                        }
                    }


adapter.notifyDataSetChanged();


binding.sRLAbout.setRefreshing(false)
;
                } else {
                    Log.e("Firebase",
"DATA (technology) NOT FOUND!");

Toast.makeText(About.this, "DATA IS
NOT FOUND!",
Toast.LENGTH_LONG).show();
                }
            }

            @Override
            public void
onCancelled(@NonNull DatabaseError
error) {
                Log.e("Firebase",
"DATABASE ERROR!",
error.toException());

Toast.makeText(About.this, "ERROR TO
LOAD DATA FROM DATABASE! \n MAKE SURE
YOUR INTERNET CONNECTION IS
AVAILABLE",
Toast.LENGTH_SHORT).show();
            }
        });
```

```
    }
}
```

About_dev.java

```
package com.rifaldi.myapplication.ui;

import android.os.Bundle;
import android.util.Log;
import android.view.Window;
import android.view.WindowManager;
import android.widget.Toast;

import androidx.annotation.NonNull;
import
androidx.appcompat.app.AppCompatActiv
ity;
import
androidx.core.content.ContextCompat;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import
androidx.core.view.WindowInsetsCompat
;

import
com.bumptech.glide.load.engine.DiskCa
cheStrategy;
import
com.google.firebase.database.DataSnap
shot;
import
com.google.firebase.database.Database
Error;
import
com.google.firebase.database.ValueEve
ntListener;
import
com.rifaldi.myapplication.util.Networ
kUtil;
import com.rifaldi.myapplication.R;
import com.bumptech.glide.Glide;
import
com.google.firebase.database.Database
Reference;
import
com.google.firebase.database.Firebase
Database;
import
com.rifaldi.myapplication.databinding
.ActivityAboutDevBinding;

public class About_dev extends
AppCompatActivity {

    private ActivityAboutDevBinding
binding;

    @Override
    protected void onCreate(Bundle
savedInstanceState) {

super.onCreate(savedInstanceState);
        binding =
ActivityAboutDevBinding.inflate(getLa
youtInflater());

setContentView(binding.getRoot());

ViewCompat.setOnApplyWindowInsetsList
ener(findViewById(R.id.aboutDevLayout
), (v, insets) -> {
            Insets systemBars =
insets.getInsets(WindowInsetsCompat.T
ype.systemBars());

v.setPadding(systemBars.left,
systemBars.top, systemBars.right,
systemBars.bottom);
            return insets;
        });

        Window window = getWindow();

window.addFlags(WindowManager.LayoutP
arams.FLAG_DRAWS_SYSTEM_BAR_BACKGROUN
DS);

window.setStatusBarColor(ContextCompa
t.getColor(this, R.color.green));

        if
(!NetworkUtil.isInternetAvailable(thi
s)){

FirebaseDatabase.getInstance().goOffl
ine();
            Toast.makeText(this, "NO
INTERNET CONNECTION!\nData might not
be updated!", Toast
                    .LENGTH_LONG)
                    .show();

            retrieveData();
        }

        retrieveData();
    }

    private void retrieveData(){
        DatabaseReference
mDatabaseReference = FirebaseDatabase

.getInstance("https://kuecnn-e1f7f-
default-rtdb.asia-
southeast1.firebasedatabase.app/")
                .getReference()
                .child("about_dev");


mDatabaseReference.keepSynced(true);


mDatabaseReference.addListenerForSing
leValueEvent(new ValueEventListener()
{
            @Override
            public void
onDataChange(@NonNull DataSnapshot
snapshot) {
                if
(snapshot.exists()){

binding.tvDevName.setText(snapshot.ch
ild("name").getValue(String.class));

binding.tvDevClass.setText(snapshot.c
hild("NIM").getValue(String.class));

binding.tvDevNIM.setText(snapshot.chi
ld("class").getValue(String.class));
```

```
Glide.with(About_dev.this)

.load(snapshot.child("image_URL").get
Value(String.class))

.placeholder(R.drawable.dummy_image_c
aptured)

.error(R.drawable.image_error)

.diskCacheStrategy(DiskCacheStrategy.
ALL)

.into(binding.ivProfilePicture);
                } else {
                    Log.e("Firebase",
"DATA (about_dev) IS NOT FOUND!");

Toast.makeText(About_dev.this, "DATA
IS NOT FOUND!",
Toast.LENGTH_LONG).show();
                }
            }

            @Override
            public void
onCancelled(@NonNull DatabaseError
error) {
                Log.e("Firebase",
"DATABASE ERROR!",
error.toException());

Toast.makeText(About_dev.this, "ERROR
TO LOAD DATA FROM DATABASE! \n MAKE
SURE YOUR INTERNET CONNECTION IS
AVAILABLE",
Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```

AppLayout.java

```
package com.rifaldi.myapplication.ui;

import
androidx.appcompat.app.AppCompatActiv
ity;
import
android.annotation.SuppressLint;
import android.content.Intent;
import
android.graphics.drawable.ColorDrawab
le;
import android.os.Bundle;
import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.MotionEvent;
import android.view.View;
import
android.view.animation.Animation;
import
android.view.animation.AnimationUtils
;
import android.widget.Button;
import android.widget.LinearLayout;
import android.widget.PopupWindow;
import android.widget.TextView;

import com.rifaldi.myapplication.R;
import
com.rifaldi.myapplication.databinding
.ActivityAppLayoutBinding;

import java.lang.ref.WeakReference;
import java.util.HashMap;
import java.util.Map;

public class AppLayout extends
AppCompatActivity {

    private ActivityAppLayoutBinding
binding;
    private static Animation
scaleAnimation, fadeInAnimation,
fadeOutAnimation;

    private static final Map<Integer,
String> toolTipTexts = new
HashMap<Integer, String>() {{
        put(R.id.btnIdentify, "Menu
yang digunakan untuk deteksi kue
menggunakan metode CNN (Convolutional
Neural Network)");
        put(R.id.btnCatalog, "Menu
untuk menampilkan daftar katalog kue
tradisional");
        put(R.id.btnAbout, "Menu
untuk menampilkan hal-hal yang
terkait dengan aplikasi");
        put(R.id.btnDevProfile, "Menu
untuk menampilkan profile pengembang
aplikasi");
        put(R.id.btnOut, "Keluar");
    }};

    @Override
    protected void onCreate(Bundle
savedInstanceState) {

super.onCreate(savedInstanceState);
        binding =
ActivityAppLayoutBinding.inflate(getL
ayoutInflater());

setContentView(binding.getRoot());

        initAnimations();
        startButtonAnimations();
        setButtonListeners();
    }

    // Inisialisasi animasi sebagai
static untuk menghindari alokasi
berulang
    private void initAnimations() {
        if (scaleAnimation == null) {
            scaleAnimation =
AnimationUtils.loadAnimation(this,
R.anim.button_press);
        }
        if (fadeInAnimation == null)
{
            fadeInAnimation =
AnimationUtils.loadAnimation(this,
R.anim.fade_in);
        }
        if (fadeOutAnimation == null)
{
            fadeOutAnimation =
AnimationUtils.loadAnimation(this,
R.anim.fade_out);
```

```
        }
    }

    private void setButtonListeners()
{
        // Menghubungkan tombol
dengan aktivitas

binding.btnAbout.setOnClickListener(v
-> openActivity(About.class));

binding.btnIdentify.setOnClickListene
r(v ->
openActivity(DetectionPage.class));

binding.btnCatalog.setOnClickListener
(v -> openActivity(Catalog.class));

binding.btnDevProfile.setOnClickListe
ner(v ->
openActivity(About_dev.class));

binding.btnOut.setOnClickListener(vie
w -> {
            finish();
            System.exit(0);
        });

        // Set animasi tekan dan
tooltip untuk semua tombol
        for (Button button : new
Button[]{binding.btnIdentify,
binding.btnCatalog, binding.btnAbout,
binding.btnDevProfile,
binding.btnOut}) {

setButtonTouchListener(button);

setToolTipListener(button);
        }
    }

    private void
openActivity(Class<?> activityClass)
{
        Intent intent = new
Intent(AppLayout.this,
activityClass);
        startActivity(intent);
    }

    private void
startButtonAnimations() {
        int delay = 100;
        for (Button button : new
Button[]{binding.btnIdentify,
binding.btnCatalog, binding.btnAbout,
binding.btnDevProfile,
binding.btnOut}) {

startAnimationWithDelay(button,
delay);
            delay += 50;
        }
    }

    private void
startAnimationWithDelay(View button,
int delay) {
        Animation animation =
AnimationUtils.loadAnimation(this,
R.anim.fade_in);

animation.setStartOffset(delay);

button.startAnimation(animation);
    }

    private void
setToolTipListener(Button button) {

button.setOnLongClickListener(view ->
{
            showToolTips(button,
toolTipTexts.get(button.getId()));
            return true;
        });
    }


@SuppressLint("ClickableViewAccessibi
lity")
    private void showToolTips(Button
button, String toolTipText) {
        WeakReference<PopupWindow>
popupWindowRef = new
WeakReference<>(createToolTipPopup(to
olTipText));
        PopupWindow popupWindow =
popupWindowRef.get();

        if (popupWindow == null)
return;


binding.overlay.setVisibility(View.VI
SIBLE);


popupWindow.setOnDismissListener(() -
> {

binding.overlay.startAnimation(fadeOu
tAnimation);

binding.overlay.setVisibility(View.GO
NE);
        });


popupWindow.showAtLocation(button.get
RootView(), Gravity.CENTER, 0, 0);


binding.overlay.setOnClickListener(v
-> {
            if
(popupWindow.isShowing())
popupWindow.dismiss();
        });
    }

    private PopupWindow
createToolTipPopup(String
toolTipText) {
        LayoutInflater inflater =
(LayoutInflater)
getSystemService(LAYOUT_INFLATER_SERV
ICE);

@SuppressLint("InflateParams") View
popupView =
```

```
inflater.inflate(R.layout.popup_tooltip, null);

        TextView tooltipView = popupView.findViewById(R.id.txtTooltip);

tooltipView.setText(toolTipText);

        PopupWindow popupWindow = new PopupWindow(popupView,

LinearLayout.LayoutParams.WRAP_CONTENT,

LinearLayout.LayoutParams.WRAP_CONTENT);


popupWindow.setFocusable(true);

popupWindow.setOutsideTouchable(true);

popupWindow.setBackgroundDrawable(new ColorDrawable());

popupView.startAnimation(fadeInAnimation);

        return popupWindow;
    }


@SuppressLint("ClickableViewAccessibility")
    private void setButtonTouchListener(Button button) {
        button.setOnTouchListener((v, event) -> {
            if (event.getAction() == MotionEvent.ACTION_DOWN) {

v.startAnimation(scaleAnimation);
            } else if (event.getAction() == MotionEvent.ACTION_UP || event.getAction() == MotionEvent.ACTION_CANCEL) {
                v.clearAnimation();
            }
            return false;
        });
    }
}
```

CakeDescription.java

```
package com.rifaldi.myapplication.ui;

import android.annotation.SuppressLint;
import android.app.Dialog;
import android.content.ClipData;
import android.content.ClipboardManager;
import android.content.Context;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.drawable.ColorDrawable;
import android.net.Uri;
import android.os.Bundle;
import android.text.Spannable;
import android.text.SpannableString;
import android.text.SpannableStringBuilder;
import android.text.style.LeadingMarginSpan;
import android.text.style.RelativeSizeSpan;
import android.text.style.StyleSpan;
import android.util.Log;
import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.PopupWindow;
import android.widget.TextView;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;
import androidx.constraintlayout.widget.ConstraintLayout;
import androidx.core.content.ContextCompat;
import androidx.core.content.FileProvider;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;
import com.google.gson.reflect.TypeToken;
import com.rifaldi.myapplication.R;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.lang.reflect.Type;
import java.util.List;
import java.util.Map;
import com.google.gson.Gson;
import com.rifaldi.myapplication.model.CookingStep;
import com.rifaldi.myapplication.model.Recipe;

import com.bumptech.glide.Glide;

import java.util.logging.Level;
import java.util.logging.Logger;


public class CakeDescription extends AppCompatActivity {
    private static final Logger
```

```
logger =
Logger.getLogger(CakeDescription.clas
s.getName());

    private TextView tvCakeName,
tvCakeDescription;
    private ImageView ivCakeImage;
    private ConstraintLayout
clCakeDescriptionPage;

    @Override
    protected void onCreate(Bundle
savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity_cake
_description);

ViewCompat.setOnApplyWindowInsetsList
ener(findViewById(R.id.cakeDescriptio
nPage), (v, insets) -> {
            Insets systemBars =
insets.getInsets(WindowInsetsCompat.T
ype.systemBars());

v.setPadding(systemBars.left,
systemBars.top, systemBars.right,
systemBars.bottom);
            return insets;
        });

        Window window = getWindow();

window.addFlags(WindowManager.LayoutP
arams.FLAG_DRAWS_SYSTEM_BAR_BACKGROUN
DS);

window.setStatusBarColor(ContextCompa
t.getColor(this, R.color.green));

        tvCakeName =
findViewById(R.id.tvCakeName);
        tvCakeDescription =
findViewById(R.id.tv_description);
        ivCakeImage =
findViewById(R.id.ivCake);

        retrieveData();
        initializedViewCopies();


findViewById(R.id.ivBackCakeDescripti
on).setOnClickListener(v ->
onBackPressed());

findViewById(R.id.ivCake).setOnClickL
istener(v -> showFullScreenImage());

findViewById(R.id.ivShare).setOnClick
Listener(v -> showShareOptions());
    }

    public void
showFullScreenImage(){
        Dialog dialog = new
Dialog(this,
android.R.style.Theme_Black_NoTitleBa
r_Fullscreen);

dialog.requestWindowFeature(Window.FE
ATURE_NO_TITLE);

dialog.setContentView(R.layout.fullsc
reen_image);

        ImageView ivImage =
findViewById(R.id.ivCake);
        ImageView fullscreenImageView
=
dialog.findViewById(R.id.fullscreenIm
ageView);

        Glide.with(this)

.load(ivImage.getDrawable())

.into(fullscreenImageView);

        dialog.show();
    }


@SuppressLint("ClickableViewAccessibi
lity")
    public void showShareOptions(){
        ImageView ivShare =
findViewById(R.id.ivShare);
        View vOverlayCakeDescription
=
findViewById(R.id.vOverlayCakeDescrip
tion);

        LayoutInflater inflater =
(LayoutInflater)
getSystemService(LAYOUT_INFLATER_SERV
ICE);

@SuppressLint("InflateParams") View
popupView =
inflater.inflate(R.layout.popup_layou
t_share_options, null);
        popupView.setElevation(12);

        final PopupWindow popupWindow
= new PopupWindow(
                popupView,

LinearLayout.LayoutParams.WRAP_CONTEN
T,

LinearLayout.LayoutParams.WRAP_CONTEN
T
        );


popupWindow.setFocusable(true);

popupWindow.setOutsideTouchable(true)
;

popupWindow.setBackgroundDrawable(new
ColorDrawable());

popupWindow.showAtLocation(findViewBy
Id(android.R.id.content),
Gravity.CENTER, 0, 0);


vOverlayCakeDescription.setVisibility
(View.VISIBLE);
```

```
        Animation fadeIn =
AnimationUtils.loadAnimation(ivShare.
getContext(), R.anim.fade_in);

popupView.startAnimation(fadeIn);


vOverlayCakeDescription.setOnClickLis
tener(v -> {
            popupWindow.dismiss();

v.setVisibility(View.GONE);
        });


popupWindow.setOnDismissListener(() -
> {
            Animation fadeOut =
AnimationUtils.loadAnimation(ivShare.
getContext(), R.anim.fade_out);


fadeOut.setAnimationListener(new
Animation.AnimationListener() {
                @Override
                public void
onAnimationStart(Animation animation)
{

                }

                @Override
                public void
onAnimationEnd(Animation animation) {

vOverlayCakeDescription.setVisibility
(View.GONE);
                }

                @Override
                public void
onAnimationRepeat(Animation
animation) {

                }
            });


vOverlayCakeDescription.startAnimatio
n(fadeOut);
        });


popupView.findViewById(R.id.llShareAs
AnImageOption).setOnClickListener(v -
> {
            clCakeDescriptionPage =
findViewById(R.id.cakeDescriptionPage
);

captureAndShareScreenshot(clCakeDescr
iptionPage, CakeDescription.this);
        });


popupView.findViewById(R.id.llShareAs
ATextOption).setOnClickListener(v ->
{
            shareTextDescription();
        });
    }

    public void
captureAndShareScreenshot(View view,
Context context) {
        try {
            Bitmap bitmap =
getScreenshot(view);
            File file =
saveBitmap(bitmap, context);
            shareScreenshot(file,
context);
        } catch (IOException e) {
            Log.e("FAILED TO SHARE
IMAGE", "Error occurred while sharing
image", e);
        }
    }

    public Bitmap getScreenshot(View
view) {
        // Ukur tinggi total dari
ConstraintLayout
        int width = view.getWidth();
        int height =
measureViewHeight(view);

        Bitmap bitmap =
Bitmap.createBitmap(width, height,
Bitmap.Config.ARGB_8888);
        Canvas canvas = new
Canvas(bitmap);

        view.layout(0, 0, width,
height);
        view.draw(canvas);
        return bitmap;
    }

    private int
measureViewHeight(View view) {
        view.measure(

View.MeasureSpec.makeMeasureSpec(view
.getWidth(),
View.MeasureSpec.EXACTLY),

View.MeasureSpec.makeMeasureSpec(0,
View.MeasureSpec.UNSPECIFIED)
        );
        return
view.getMeasuredHeight();
    }

    public File saveBitmap(Bitmap
bitmap, Context context) throws
IOException {
        File cacheDir =
context.getCacheDir();
        File file = new
File(cacheDir, "screenshot.png");
        FileOutputStream fos = new
FileOutputStream(file);

bitmap.compress(Bitmap.CompressFormat
.PNG, 100, fos);
        fos.flush();
        fos.close();
        return file;
    }
```

```
    public void shareScreenshot(File
file, Context context) {
        Uri uri =
FileProvider.getUriForFile(context,
context.getPackageName() +
".fileprovider", file);

        Intent intent = new
Intent(Intent.ACTION_SEND);
        intent.setType("image/png");

intent.putExtra(Intent.EXTRA_STREAM,
uri);

intent.addFlags(Intent.FLAG_GRANT_REA
D_URI_PERMISSION);

context.startActivity(Intent.createCh
ooser(intent, "Share Screenshot"));
    }


    public void
shareTextDescription(){
        TextView tvCakeName =
findViewById(R.id.tvCakeName);
        TextView tvCakeDescription =
findViewById(R.id.tv_description);
        TextView tvCakeIngredients =
findViewById(R.id.tv_ingredients);
        TextView
tvCakeAdditionalIngredients =
findViewById(R.id.tv_sauce_ingredient
s);
        TextView tvCakeSteps =
findViewById(R.id.tv_steps);
        TextView tvCakeTips =
findViewById(R.id.tv_tips);
        TextView tvCakeReferences =
findViewById(R.id.tv_references);

        String cakeName =
tvCakeName.getText().toString();
        String cakeDescription =
tvCakeDescription.getText().toString(
);
        String cakeIngredients =
tvCakeIngredients.getText().toString(
);
        String
cakeAdditionalIngredients =
tvCakeAdditionalIngredients.getText()
.toString();
        String cakeSteps =
tvCakeSteps.getText().toString();
        String cakeTips =
tvCakeTips.getText().toString();
        String cakeReferences =
tvCakeReferences.getText().toString()
;

        if
(cakeAdditionalIngredients.isEmpty())
{
            cakeAdditionalIngredients
= "";
        } else {
            cakeAdditionalIngredients
= "Bahan-bahan (tambahan)\n" +
cakeAdditionalIngredients;
        }

        String fullText = "=== "+
cakeName + " ====\n\n" + "Penjelasan:
\n" + cakeDescription + "\n"
+"\nResep: \nBahan-bahan\n" +
cakeIngredients + "\n" +
cakeAdditionalIngredients + "\n" +
"Cara Membuat:\n" + cakeSteps + "\n"
+ "Tips: \n" + cakeTips +
"\nReferensi: \n" + cakeReferences;

        Intent shareIntent = new
Intent (Intent.ACTION_SEND);

shareIntent.setType("text/plain");

shareIntent.putExtra(Intent.EXTRA_TEX
T, fullText);

startActivity(Intent.createChooser(sh
areIntent, "Bagikan teks dengan: "));
    }

    public void
initializedViewCopies(){
        ImageView ivCopyTvDescription
=
findViewById(R.id.ivCopyTvDescription
);
        ImageView ivCopyCakeTips =
findViewById(R.id.ivCopyCakeTips);
        ImageView
ivCopyCakeReferences =
findViewById(R.id.ivCopyCakeReference
s);
        ImageView ivCopyCakeRecipes =
findViewById(R.id.ivCopyCakeRecipes);

        TextView tvCakeDescription =
findViewById(R.id.tv_description);
        TextView tvCakeTips =
findViewById(R.id.tv_tips);
        TextView tvCakeReferences =
findViewById(R.id.tv_references);
        TextView tvCakeIngredients =
findViewById(R.id.tv_ingredients);
        TextView
tvCakeSauceIngredients =
findViewById(R.id.tv_sauce_ingredient
s);
        TextView tvCakeSteps =
findViewById(R.id.tv_steps);

        String cakeIngredients =
"Bahan-bahan: \n\n" +
tvCakeIngredients.getText().toString(
) + "\n";
        String cakeSteps = "Cara
Membuat: \n\n"
+tvCakeSteps.getText().toString() +
"\n";
        String cakeSauceIngredients;

        if
(tvCakeSauceIngredients.getText().toS
tring().isEmpty()){
            cakeSauceIngredients =
"";
        } else {
            cakeSauceIngredients =
"Bahan Tambahan : \n\n" +
```

```
tvCakeIngredients.getText().toString(
);
        }

        String cakeRecipes = "Resep
(" + tvCakeName.getText().toString()
+ "): \n\n" + cakeIngredients +
cakeSauceIngredients + cakeSteps;


ivCopyTvDescription.setOnClickListene
r(v -> copyTextWithMessage("Deskripsi
(" + tvCakeName.getText().toString()
+ "): \n\n"+
tvCakeDescription.getText().toString(
), "Deskripsi disalin ke
clipboard"));

ivCopyCakeTips.setOnClickListener(v -
> copyTextWithMessage("Tips (" +
tvCakeName.getText().toString() + "):
\n\n"+
tvCakeTips.getText().toString(),
"Tips disalin ke clipboard"));

ivCopyCakeReferences.setOnClickListen
er(v ->
copyTextWithMessage("Referensi (" +
tvCakeName.getText().toString() + "):
\n\n" +
tvCakeReferences.getText().toString()
, "Referensi disalin ke clipboard"));

ivCopyCakeRecipes.setOnClickListener(
v -> copyTextWithMessage(cakeRecipes,
"Resep disalin ke cliboard"));
    }

    private void
copyTextWithMessage(String
textToCopy, String message) {
        ClipboardManager clipboard =
(ClipboardManager)
getSystemService(Context.CLIPBOARD_SE
RVICE);
        ClipData clip =
ClipData.newPlainText("Copied Text",
textToCopy);

clipboard.setPrimaryClip(clip);

Toast.makeText(CakeDescription.this,
message, Toast.LENGTH_SHORT).show();
    }

    public void retrieveData(){
        String cakeName =
getIntent().getStringExtra("cakeName"
);
        String cakeDescription =
getIntent().getStringExtra("cakeDescr
iption");
        String cakeImageUrl =
getIntent().getStringExtra("cakeImage
Url");

        tvCakeName.setText(cakeName);

tvCakeDescription.setText(cakeDescrip
tion);

        Glide.with(this)
                .load(cakeImageUrl)
                .into(ivCakeImage);

        // Konversi kembali ke object
        String recipesJson =
getIntent().getStringExtra("cakeRecip
es");
        String tipsJson =
getIntent().getStringExtra("cakeTips"
);
        String referencesJson =
getIntent().getStringExtra("cakeRefer
ences");

        Log.d("JSON_DATA", "Received
JSON: " + recipesJson);

        try {
            Gson gson = new Gson();
            Recipe recipe =
gson.fromJson(recipesJson,
Recipe.class);

            if (recipe == null) {
                Log.e("JSON_ERROR",
"Recipe object is null");
                return;
            }

            // lanjutkan kode Anda...

        } catch (Exception e) {
            logger.log(Level.SEVERE,
"An exception occurred", e);
        }

        Gson gson = new Gson();

        // Parse recipe
        Recipe recipe =
gson.fromJson(recipesJson,
Recipe.class);
        List<CookingStep> caraMembuat
= recipe.getCaraMembuat();

        Log.d("CakeDescription
resep", recipesJson);

        // Tampilkan bahan-bahan
        StringBuilder bahanBahanText
= new StringBuilder();
        List<String> bahanBahan =
recipe.getBahanBahan();
        if (bahanBahan != null) {
            for (String bahan :
bahanBahan) {

bahanBahanText.append("•
").append(bahan).append("\n");
            }
        }
        Log.d("Bahan_bahan",
bahanBahanText.toString());

        TextView tvIngredients =
findViewById(R.id.tv_ingredients);
        TextView tvSauceTitle =
findViewById(R.id.tv_sauce_title);

tvIngredients.setText(bahanBahanText.
```

```
toString());

        List<String> bahanSausSantan
= recipe.getBahanSausSantan(); //
Metode untuk mendapatkan daftar bahan

        if (bahanSausSantan != null
&& !bahanSausSantan.isEmpty()) {
            StringBuilder
bahanSausText = new StringBuilder();
            for (String bahan :
bahanSausSantan) {

bahanSausText.append("•
").append(bahan).append("\n");
            }

            if (tvSauceTitle != null)
{

tvSauceTitle.setVisibility(View.VISIB
LE);
            }

            TextView
tvSauceIngredients =
findViewById(R.id.tv_sauce_ingredient
s);
            if (tvSauceIngredients !=
null) {

tvSauceIngredients.setVisibility(View
.VISIBLE);

tvSauceIngredients.setText(bahanSausT
ext.toString());
            }
        } else {

Log.d("Bahan_Bahan_sauce",
"bahanSausSantan is null or empty!");
        }

        if (caraMembuat != null) {
            SpannableStringBuilder
stepsText = new
SpannableStringBuilder();
            int categoryNumber = 1;

            // Iterate through each
cooking step
            for (CookingStep step :
caraMembuat) {
                String kategori =
step.getNama();
                List<String> steps =
step.getLangkah();

                // Create category
with bold text and larger size
                SpannableString
kategoriSpannable = new
SpannableString(categoryNumber + ". "
+ kategori + ":\n");

kategoriSpannable.setSpan(new
StyleSpan(android.graphics.Typeface.B
OLD), 0, kategori.length() + 4,
Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);

kategoriSpannable.setSpan(new
RelativeSizeSpan(1.05f), 0,
kategori.length() + 4,
Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);


stepsText.append(kategoriSpannable);

                // Iterate through
each step in the category
                for (String
stepDetail : steps) {
                    SpannableString
stepSpannable = new
SpannableString("• " + stepDetail +
"\n");

stepSpannable.setSpan(new
LeadingMarginSpan.Standard(40), 0,
stepSpannable.length(),
Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);

stepsText.append(stepSpannable);
                }


stepsText.append("\n"); // Add space
between categories
                categoryNumber++;
            }

            // Set text to TextView
            TextView tvSteps =
findViewById(R.id.tv_steps);
            if (tvSteps != null) {

tvSteps.setText(stepsText);
            } else {
                Log.d("Cara_membuat",
"TextView not found!");
            }
        } else {
            Log.d("Cara_membuat",
"caraMembuat is null!");
        }


        // Parse dan tampilkan tips
        Type tipsType = new
TypeToken<List<String>>(){}.getType()
;
        List<String> tips =
gson.fromJson(tipsJson, tipsType);
        if (tips != null) {
            StringBuilder tipsText =
new StringBuilder();
            for (String tip : tips) {
                tipsText.append("•
").append(tip).append("\n");
            }
            TextView tvTips =
findViewById(R.id.tv_tips);

tvTips.setText(tipsText.toString());
        }

        // Parse dan tampilkan
references
        Type referencesType = new
TypeToken<List<String>>(){}.getType()
;
        List<String> references =
```

```
gson.fromJson(referencesJson,
referencesType);
        if (references != null) {
            StringBuilder
referencesText = new StringBuilder();
            for (String reference :
references) {

referencesText.append("•
").append(reference).append("\n");
            }
            TextView tvReferences =
findViewById(R.id.tv_references);

tvReferences.setText(referencesText.t
oString());
        }
    }

    private String
capitalizeWords(String str) {
        String[] words = str.split("
");
        StringBuilder capitalized =
new StringBuilder();
        for (String word : words) {
            if (!word.isEmpty()) {

capitalized.append(Character.toUpperC
ase(word.charAt(0)))

.append(word.substring(1).toLowerCase
()).append(" ");
            }
        }
        return
capitalized.toString().trim();
    }

    @Override
    public void onBackPressed() {
        super.onBackPressed();
        finish();
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
    }

}
```

Catalog.java

```
package com.rifaldi.myapplication.ui;

import android.os.Bundle;
import android.util.Log;
import android.view.Window;
import android.view.WindowManager;
import android.widget.EditText;

import
androidx.appcompat.widget.SearchView;

import androidx.annotation.NonNull;
import
androidx.appcompat.app.AppCompatActiv
ity;
import
androidx.core.content.ContextCompat;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import
androidx.core.view.WindowInsetsCompat
;
import
androidx.recyclerview.widget.LinearLa
youtManager;
import
androidx.recyclerview.widget.Recycler
View;

import
com.google.firebase.database.DataSnap
shot;
import
com.google.firebase.database.Database
Error;
import
com.google.firebase.database.ValueEve
ntListener;
import
com.rifaldi.myapplication.adapter.Cak
eAdapter;
import com.rifaldi.myapplication.R;

// firebase
import
com.google.firebase.database.Database
Reference;
import
com.google.firebase.database.Firebase
Database;
import
com.rifaldi.myapplication.model.Cake;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class Catalog extends
AppCompatActivity {

    private DatabaseReference
mDatabaseReference;
    private RecyclerView
recyclerView;
    private CakeAdapter adapter;
    private List<Cake> cakeList;
    private List<Cake>
filteredCakeList;

    @Override
    protected void onCreate(Bundle
savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity_cata
log);

ViewCompat.setOnApplyWindowInsetsList
ener(findViewById(R.id.catalogLayout)
, (v, insets) -> {
            Insets systemBars =
insets.getInsets(WindowInsetsCompat.T
ype.systemBars());

v.setPadding(systemBars.left,
systemBars.top, systemBars.right,
systemBars.bottom);
```

```
            return insets;
        });

        Window window = getWindow();

window.addFlags(WindowManager.LayoutParams.FLAG_DRAWS_SYSTEM_BAR_BACKGROUNDS);

window.setStatusBarColor(ContextCompat.getColor(this, R.color.green));

        try {
            mDatabaseReference = FirebaseDatabase.getInstance("https://kuecnn-e1f7f-default-rtdb.asia-southeast1.firebasedatabase.app/").getReference();
            Log.d("FIREBASE", "Firebase initialized successfully");
        } catch (Exception e) {
            Log.e("FIREBASE", "Failed to initialize Firebase", e);
        }

        cakeList = new ArrayList<>();
        filteredCakeList = new ArrayList<>();
        Log.d("INITIALIZED LIST", "Lists initialized");

        // Setup RecyclerView dengan log
        try {
            recyclerView = findViewById(R.id.rvCakeItem);

recyclerView.setLayoutManager(new LinearLayoutManager(this));
            adapter = new CakeAdapter(this, filteredCakeList);

recyclerView.setAdapter(adapter);
            Log.d("INITIALIZED RECYCLER VIEW", "RecyclerView setup complete");
        } catch (Exception e) {
            Log.e("INITIALIZED RECYCLER VIEW", "Error setting up RecyclerView", e);
        }

        // Setup SearchView dengan log
        try {
            SearchView searchView = findViewById(R.id.svCatalog);

searchView.setIconified(false);

            EditText searchEditText = searchView.findViewById(androidx.appcompat.R.id.search_src_text);

searchEditText.setTextColor(ContextCompat.getColor(this, R.color.soft_black));

searchEditText.setHint("Cari Kue...");

searchEditText.setHintTextColor(ContextCompat.getColor(this, R.color.soft_black));


searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
                @Override
                public boolean onQueryTextSubmit(String query) {

Log.d("INITIALIZED SEARCH VIEW", "Search submitted: " + query);
                    filter(query);
                    return true;
                }

                @Override
                public boolean onQueryTextChange(String newText) {

Log.d("INITIALIZED SEARCH VIEW", "Search text changed: " + newText);
                    filter(newText);
                    return true;
                }
            });
            Log.d("INITIALIZED SEARCH VIEW", "SearchView setup complete");
        } catch (Exception e) {
            Log.e("INITIALIZED SEARCH VIEW", "Error setting up SearchView", e);
        }

        readAllData();

findViewById(R.id.ivBack).setOnClickListener(v -> onBackPressed());
    }

    @Override
    public void onBackPressed() {
        super.onBackPressed();
        finish();
    }

    private void filter(String text) {
        final String TAG = "Catalog Activity";
        Log.d(TAG, "Filtering with text: " + text);
        filteredCakeList.clear();

        if (text.isEmpty()) {

filteredCakeList.addAll(cakeList);
            Log.d(TAG, "Showing all items: " + filteredCakeList.size());
        } else {
            String searchText = text.toLowerCase().trim();
            for (Cake cake : cakeList) {
                if (cake != null && cake.getCakeName() != null) {
                    String cakeName = cake.getCakeName().toLowerCase();

                    if
```

```
(cakeName.contains(searchText)) {

filteredCakeList.add(cake);
                        Log.d(TAG,
"Added to filtered list: " +
cake.getCakeName());
                    }
                }
            }
            Log.d(TAG, "Filtered
items count: " +
filteredCakeList.size());
        }


adapter.setData(filteredCakeList);
// Memperbarui data di adapter

adapter.notifyDataSetChanged();
    }

    private void readAllData() {

mDatabaseReference.child("katalog_kue
").addValueEventListener(new
ValueEventListener() {
            @Override
            public void
onDataChange(@NonNull DataSnapshot
snapshot) {
                if
(snapshot.exists()) {
                    cakeList.clear();


Log.d("Katalog_snapshot",
snapshot.toString());

                    for (DataSnapshot
cakeSnapshot: snapshot.getChildren())
{
                        try {

Log.d("FIREBASE GET", "Processing
cake: " + cakeSnapshot.getKey());
                            String
cakeName =
cakeSnapshot.child("nama_kue").getVal
ue(String.class);
                            String
cakeShortDescription =
cakeSnapshot.child("penjelasan_singka
t").getValue(String.class);
                            String
cakeDescription =
cakeSnapshot.child("penjelasan").getV
alue(String.class);
                            String
cakeImageUrl =
cakeSnapshot.child("image_url").getVa
lue(String.class);
                            // Log
the cake data we're trying to parse



List<String> cakeReferences = new
ArrayList<>();

List<String> cakeTips = new
ArrayList<>();

Map<String, Object> cakeRecipes = new
HashMap<>();

                            // Safely
get lists and map
                            if
(cakeSnapshot.child("referensi").exis
ts()) {
                                for
(DataSnapshot ref :
cakeSnapshot.child("referensi").getCh
ildren()) {

cakeReferences.add(ref.getValue(Strin
g.class));
                                }
                            }

                            if
(cakeSnapshot.child("tips").exists())
{
                                for
(DataSnapshot tip :
cakeSnapshot.child("tips").getChildre
n()) {

cakeTips.add(tip.getValue(String.clas
s));
                                }
                            }

                            if
(cakeSnapshot.child("resep").exists()
) {

cakeRecipes = (Map<String, Object>)
cakeSnapshot.child("resep").getValue(
);
                            }

                            Cake cake
= new Cake(cakeName,
cakeShortDescription,
cakeDescription, cakeImageUrl,

cakeReferences, cakeTips,
cakeRecipes);


Log.d("TEST", "Successfully added
cake to list: " + cakeName);

cakeList.add(cake);
                        } catch
(Exception e) {

Log.e("Firebase", "Error parsing cake
data", e);
                        }
                    }
                    // Reset filter
setelah data baru dimuat
                    filter("");  //
Ini akan memperbarui filteredCakeList
dengan semua data
                }
            }

            @Override
            public void
```

```
onCancelled(@NonNull DatabaseError
error) {
                Log.e("Firebase",
"Gagal membaca data",
error.toException());
            }
        });
    }
}
```

DetectionPage.java

```
package com.rifaldi.myapplication.ui;

import android.Manifest;
import android.content.Intent;
import
android.content.pm.PackageManager;
import android.graphics.Bitmap;
import
android.graphics.BitmapFactory;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;
import android.os.Environment;
import android.provider.MediaStore;
import android.util.Log;
import android.view.Window;
import android.view.WindowManager;
import android.widget.Toast;

import
androidx.appcompat.app.AppCompatActiv
ity;
import
androidx.core.app.ActivityCompat;
import
androidx.core.content.ContextCompat;
import
androidx.core.content.FileProvider;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import
androidx.core.view.WindowInsetsCompat
;

import com.rifaldi.myapplication.R;
import
com.rifaldi.myapplication.ml.MobileNe
tV2;
import
com.rifaldi.myapplication.ml.BestEffi
cientNetB0;
import
com.rifaldi.myapplication.ml.BestMobi
leNetV3Small;

import org.tensorflow.lite.DataType;
import
org.tensorflow.lite.support.common.op
s.DequantizeOp;
import
org.tensorflow.lite.support.common.op
s.NormalizeOp;
import
org.tensorflow.lite.support.image.Ima
geProcessor;
import
org.tensorflow.lite.support.image.Ten
sorImage;
import
org.tensorflow.lite.support.image.ops
.ResizeOp;
import
org.tensorflow.lite.support.tensorbuf
fer.TensorBuffer;

import java.io.BufferedReader;
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.List;
import java.util.Objects;

public class DetectionPage extends
AppCompatActivity {

    private static final int
REQUEST_IMAGE_CAPTURE = 1;
    private static final int
REQUEST_CODE_PERMISSION = 101;
    private static final int
IMAGE_SIZE = 224;
    private Uri photoUri;
    private static final String TAG =
"LabelLoader";

    @Override
    protected void onCreate(Bundle
savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity_dete
ction_page);

ViewCompat.setOnApplyWindowInsetsList
ener(findViewById(R.id.detectionPage)
, (v, insets) -> {
            Insets systemBars =
insets.getInsets(WindowInsetsCompat.T
ype.systemBars());

v.setPadding(systemBars.left,
systemBars.top, systemBars.right,
systemBars.bottom);
            return insets;
        });

        Window window = getWindow();

window.addFlags(WindowManager.LayoutP
arams.FLAG_DRAWS_SYSTEM_BAR_BACKGROUN
DS);

window.setStatusBarColor(ContextCompa
t.getColor(this, R.color.green));

        initializeButtons();
    }

    public void initializeButtons(){

findViewById(R.id.btnGetFromCamera).s
etOnClickListener(view ->
dispatchTakePictureIntent());


findViewById(R.id.btnGetFromGallery).
```

```
setOnClickListener(view -> {
            Intent galleryIntent =
new Intent(Intent.ACTION_PICK,
MediaStore.Images.Media.EXTERNAL_CONT
ENT_URI);
            if
(checkAndRequestPermission()){

startActivityForResult(galleryIntent,
102);
            } else {
                Toast.makeText(this,
"Dibutuhkan akses ke galeri untuk
menjalankan aplikasi ini",
Toast.LENGTH_LONG).show();
            }

        });


findViewById(R.id.ivBackDetectionPage
).setOnClickListener(v ->
super.onBackPressed());
    }

    private void
dispatchTakePictureIntent() {
        Intent takePictureIntent =
new
Intent(MediaStore.ACTION_IMAGE_CAPTUR
E);

        File photoFile = new
File(getExternalFilesDir(Environment.
DIRECTORY_PICTURES),
                "photo_" +
System.currentTimeMillis() + ".jpg");

        photoUri =
FileProvider.getUriForFile(this,

"com.rifaldi.myapplication.fileprovid
er",  // Ganti dengan authority
aplikasi Anda
                photoFile);

takePictureIntent.putExtra(MediaStore
.EXTRA_OUTPUT, photoUri);

        try {
            if
(takePictureIntent.resolveActivity(ge
tPackageManager()) != null) {

startActivityForResult(takePictureInt
ent, REQUEST_IMAGE_CAPTURE);
            }
        } catch (Exception e) {
            Toast.makeText(this,
"Gagal membuka kamera: " +
e.getMessage(),

Toast.LENGTH_LONG).show();
            finish();
        }
    }

    private boolean
checkAndRequestPermission() {
        if (Build.VERSION.SDK_INT >=
Build.VERSION_CODES.TIRAMISU) {
            if
(ContextCompat.checkSelfPermission(th
is,
Manifest.permission.READ_MEDIA_IMAGES
) !=
PackageManager.PERMISSION_GRANTED) {

ActivityCompat.requestPermissions(thi
s, new
String[]{Manifest.permission.READ_MED
IA_IMAGES}, REQUEST_CODE_PERMISSION);
                return false;
            }
        } else {
            if
(ContextCompat.checkSelfPermission(th
is,
Manifest.permission.READ_EXTERNAL_STO
RAGE) !=
PackageManager.PERMISSION_GRANTED) {

ActivityCompat.requestPermissions(thi
s, new
String[]{Manifest.permission.READ_EXT
ERNAL_STORAGE},
REQUEST_CODE_PERMISSION);
            }
        }
        return true;
    }


    @Override
    protected void
onActivityResult(int requestCode, int
resultCode, Intent data) {

super.onActivityResult(requestCode,
resultCode, data);
        try {
            if (requestCode ==
REQUEST_IMAGE_CAPTURE && resultCode
== RESULT_OK) {
                try {
                    Bitmap
fullResImage =
MediaStore.Images.Media.getBitmap(thi
s.getContentResolver(), photoUri);

                    if (fullResImage
!= null) {
                        // Reduced
Image Resolution
                        int
originalWidth =
fullResImage.getWidth();
                        int
originalHeight =
fullResImage.getHeight();

                        // Calculate
the new dimensions for 50% scaling
                        int
reducedWidth, reducedHeight;

                        if
(originalWidth >= 3000){

reducedWidth = originalWidth / 8;

reducedHeight = originalHeight / 8;
```

```
                        } else if
(originalWidth >= 2000){

reducedWidth = originalWidth / 3;

reducedHeight = originalHeight / 3;
                        } else if
(originalWidth >= 1000){

reducedWidth = originalWidth / 2;

reducedHeight = originalHeight / 2;
                        } else {

reducedWidth = originalWidth;

reducedHeight = originalHeight;
                        }
                        Bitmap
reducedImage =
Bitmap.createScaledBitmap(

fullResImage,

reducedWidth,

reducedHeight,
                                    true
                        );

                        // Membuat
salinan gambar untuk klasifikasi
                        Bitmap
classificationImage =
Bitmap.createScaledBitmap(

fullResImage,

IMAGE_SIZE,

IMAGE_SIZE,
                                    true
                        );


classifyImage(reducedImage);

                        // CLEAN UP
IMAGES

classificationImage.recycle();

reducedImage.recycle();

                    } else {
                        throw new
Exception("Gagal mengambil gambar
dari kamera");
                    }

                } catch (IOException
e) {

Toast.makeText(this, "Error
processing image: " + e.getMessage(),
Toast.LENGTH_LONG).show();
                    Log.e("Detection
Page","Error processing image: ", e);
                } catch (Exception e)
{
                    Log.e("Detection
Page","Error: ", e);
                        throw new
RuntimeException(e);
                }

            } else if (requestCode ==
102 && resultCode == RESULT_OK &&
data != null) {
                    Uri selectedImageUri
= data.getData();
                    if (selectedImageUri
!= null) {
                        try {
                            // Set up
options for safe bitmap loading

BitmapFactory.Options options = new
BitmapFactory.Options();

options.inJustDecodeBounds = true;

                            // First
pass: get dimensions only
                            InputStream
imageStream =
getContentResolver().openInputStream(
selectedImageUri);

BitmapFactory.decodeStream(imageStrea
m, null, options);
                            assert
imageStream != null;

imageStream.close();

                            // Get the
original size
                            int
originalWidth = options.outWidth;
                            int
originalHeight = options.outHeight;

                            // Calculate
the new dimentions for 50% scaling
                            int
reducedWidth, reducedHeight;

                            if
(originalWidth >= 3000){

reducedWidth = originalWidth / 9;

reducedHeight = originalHeight / 9;
                            } else if
(originalWidth >= 2000){

reducedWidth = originalWidth / 3;

reducedHeight = originalHeight / 3;
                            } else if
(originalWidth >= 1000){

reducedWidth = originalWidth / 2;

reducedHeight = originalHeight / 2;
                            } else {

reducedWidth = originalWidth;

reducedHeight = originalHeight;
                            }
```

```
                        // Calculate
sample size

options.inSampleSize =
calculateInSampleSize(options);

                        // Second
pass: decode with calculated sample
size

options.inJustDecodeBounds = false;
                        imageStream =
getContentResolver().openInputStream(
selectedImageUri);
                        Bitmap
originalImage =
BitmapFactory.decodeStream(imageStrea
m, null, options);
                        assert
imageStream != null;

imageStream.close();

                        if
(originalImage == null) {
                            throw new
IOException("Failed to load image");
                        }

                        // Resize the
original image to exactly 224x224
                        Bitmap
fullResImage =
Bitmap.createScaledBitmap(

originalImage,

IMAGE_SIZE,

IMAGE_SIZE,
                                true
                        );

                        // Resize the
original image to exactly 224x224
                        Bitmap
reducedImage =
Bitmap.createScaledBitmap(

originalImage,

reducedWidth,

reducedHeight,
                                true
                        );


classifyImage(reducedImage);

                        // CLEAN UP
IMAGES

fullResImage.recycle();

originalImage.recycle();

reducedImage.recycle();

//
startActivity(Thisintent);
                    } catch
(IOException e) {

Toast.makeText(this, "Error
processing gallery image: " +
e.getMessage(),

Toast.LENGTH_LONG).show();

Log.e("Detection_Page", "Error
processing gallery image", e);
                    }
                }
            } else {
                Toast.makeText(this,
"Pengambilan gambar dibatalkan",
Toast.LENGTH_SHORT).show();
                finish();
            }
        } catch (Exception e) {
            Toast.makeText(this,
"Error: " + e.getMessage(),
Toast.LENGTH_LONG).show();
        }
    }

    private void classifyImage(Bitmap
bitmap){
        float[]
firstModelProbabilities,
secondModelProbabilities;
        try {
            firstModelProbabilities =
getModelProbabilities(bitmap,
"EfficientNetB0");
            secondModelProbabilities
= getModelProbabilities(bitmap,
"EfficientNetB0");

            String firstModelResult =
getMaxProbabilityLabel(firstModelProb
abilities);
            String secondModelResult
=
getMaxProbabilityLabel(secondModelPro
babilities);
            String averagingResult =
averagingScore(firstModelProbabilitie
s, secondModelProbabilities);
            String
detailedAveragingOutput =
detailedAveragingOutput(firstModelPro
babilities,
secondModelProbabilities);


prepareResultAndIntent(bitmap,
firstModelResult, secondModelResult,
averagingResult,
detailedAveragingOutput);

        } catch (IOException e) {
            throw new
RuntimeException(e);
        }
    }

    private void
prepareResultAndIntent(Bitmap bitmap,
String firstResult, String
```

```
secondResult, String averagingResult,
String detailedAveraging){
        String firstLabel =
firstResult.substring(firstResult.ind
exOf(":") + 2, firstResult.indexOf("
(")).trim();
        String firstConfScore =
firstResult.substring(firstResult.ind
exOf("(") + 1,
firstResult.indexOf("%)")).trim();

        String secondLabel =
secondResult.substring(secondResult.i
ndexOf(":") + 2,
secondResult.indexOf(" (")).trim();
        String secondConfScore =
secondResult.substring(secondResult.i
ndexOf("(") + 1,
secondResult.indexOf("%)")).trim();

        String averagingScore =
averagingResult.substring(averagingRe
sult.lastIndexOf("(") + 1,
averagingResult.lastIndexOf("%)"));
        String averagingLabel =
(averagingResult.contains(" (")) ?
averagingResult.substring(0,
averagingResult.indexOf(" (")) :
averagingResult;

        // Mengirim hasil ke
HasilDeteksiActivity
        Intent intent = new
Intent(DetectionPage.this,
DetectionResult.class);

intent.putExtra("HASIL_DETEKSI",
averagingLabel);

intent.putExtra("HASIL_DETEKSI_CONFID
ENCE_SCORE", averagingScore);

intent.putExtra("FIRST_MODEL_RESULT",
firstLabel);

intent.putExtra("FIRST_MODEL_CONF_SCO
RE", firstConfScore);

intent.putExtra("SECOND_MODEL_RESULT"
, secondLabel);

intent.putExtra("SECOND_MODEL_CONF_SC
ORE", secondConfScore);

intent.putExtra("DETAILED_AVERAGING_S
CORE", detailedAveraging);

        // Log dalam satu baris
        Log.d("IntentData",
"HASIL_DETEKSI: " + averagingLabel +
                " |
HASIL_DETEKSI_CONFIDENCE_SCORE: " +
averagingScore +
                " |
FIRST_MODEL_RESULT: " + firstLabel +
                " |
FIRST_MODEL_CONF_SCORE: " +
firstConfScore +
                " |
SECOND_MODEL_RESULT: " + secondLabel
+
                " |
SECOND_MODEL_CONF_SCORE: " +
secondConfScore +
                " |
DETAILED_AVERAGING_SCORE: " +
detailedAveraging);


        // Menyimpan gambar ke byte
array
        ByteArrayOutputStream stream
= new ByteArrayOutputStream();

bitmap.compress(Bitmap.CompressFormat
.PNG, 100, stream);
        byte[] byteArray =
stream.toByteArray();
        intent.putExtra("GAMBAR_KUE",
byteArray);

        startActivity(intent);
    }

    // Add this helper method to
calculate appropriate sample size
    private int
calculateInSampleSize(BitmapFactory.O
ptions options) {
        final int height =
options.outHeight;
        final int width =
options.outWidth;
        int inSampleSize = 1;

        if (height > 224 || width >
224) {
            final int halfHeight =
height / 2;
            final int halfWidth =
width / 2;

            while ((halfHeight /
inSampleSize) >= 224 && (halfWidth /
inSampleSize) >= 224) {
                inSampleSize *= 2;
            }
        }

        return inSampleSize;
    }

    private TensorImage
preprocessImage(Bitmap bitmap, String
modelName) {
        try {
            Bitmap scaledBitmap =
Bitmap.createScaledBitmap(bitmap,
IMAGE_SIZE, IMAGE_SIZE, true);
            Bitmap convertedBitmap;
            if
(scaledBitmap.getConfig() !=
Bitmap.Config.ARGB_8888) {
                convertedBitmap =
scaledBitmap.copy(Bitmap.Config.ARGB_
8888, true);

scaledBitmap.recycle();
            } else {
                convertedBitmap =
scaledBitmap;
            }
```

```
            TensorImage tensorImage =
new TensorImage(DataType.FLOAT32);

tensorImage.load(convertedBitmap);

            ImageProcessor
imageProcessor = null;
            if
(Objects.equals(modelName,
"MobileNetV2")){
                imageProcessor = new
ImageProcessor.Builder()
                        .add(new
ResizeOp(IMAGE_SIZE, IMAGE_SIZE,
ResizeOp.ResizeMethod.BILINEAR))
//                      .add(new
NormalizeOp(0, 255f))   // DEFAULT
                        .add(new
NormalizeOp(127.5f, 127.5f))   //
DEFAULT
                        .build();
            } else if
(Objects.equals(modelName,
"MobileNetV3")){
                // MOBILENETV3
                imageProcessor = new
ImageProcessor.Builder()
                        .add(new
ResizeOp(IMAGE_SIZE, IMAGE_SIZE,
ResizeOp.ResizeMethod.BILINEAR))
                        .add(new
NormalizeOp(0f, 1/255.0f)) // Ubah ke
[0,1]
                        .add(new
DequantizeOp(0f, 1/255.0f)) //
Memastikan konversi ke float
                        .add(new
NormalizeOp(1.0f, 2.0f)) // Ubah
[0,1] ke [-1,1]
                        .build();
            } else if
(Objects.equals(modelName,
"EfficientNetB0")){
                // EFFICIENTNETB0
                imageProcessor = new
ImageProcessor.Builder()
                        .add(new
ResizeOp(IMAGE_SIZE, IMAGE_SIZE,
ResizeOp.ResizeMethod.BILINEAR))
                        .add(new
NormalizeOp(0f, 1/255.0f)) // Ubah ke
[0,1]
                        .add(new
DequantizeOp(0f, 1/255.0f)) //
Memastikan konversi ke float
                        .add(new
NormalizeOp(1.0f, 2.0f)) // Ubah
[0,1] ke [-1,1]
                        .build();
            }


            assert imageProcessor !=
null;
            TensorImage
processedImage =
imageProcessor.process(tensorImage);

            if (convertedBitmap !=
null && convertedBitmap != bitmap) {
convertedBitmap.recycle();
            }

            return processedImage;
        } catch (Exception e) {
            throw new
RuntimeException("Gagal memproses
gambar: " + e.getMessage());
        }
    }

    private String
getMaxProbabilityLabel(float[]
probabilities) {
        String[] labels =
loadLabels(); // Memuat label dari
file
        int maxIndex = 0;
        float maxProb =
probabilities[0];

        for (int i = 1; i <
probabilities.length; i++) {
            if (probabilities[i] >
maxProb) {
                maxProb =
probabilities[i];
                maxIndex = i;
            }
        }

        return "Prediksi: " +
labels[maxIndex] + " (" + maxProb *
100 + "%)";
    }


    private String[] loadLabels() {
        List<String> labelList = new
ArrayList<>();

        try (InputStream is =
getAssets().open("cake_labels.txt");
             BufferedReader reader =
new BufferedReader(new
InputStreamReader(is))) {

            String line;
            while ((line =
reader.readLine()) != null) {

labelList.add(line.trim());
            }

        } catch (IOException e) {
            Log.e(TAG, "Gagal memuat
label dari file cake_labels.txt", e);
            return new String[0];
        }

        return labelList.toArray(new
String[0]);
    }

    private String
averagingScore(float[]
probabilities1, float[]
probabilities2) {
        if (probabilities1.length !=
probabilities2.length) {
```

```
            return "Error: Jumlah
kelas berbeda antara dua model";
        }
        String[] labels =
loadLabels();

        // Log probabilitas dari
masing-masing model untuk setiap
kelas
        Log.d("ModelFusion",
"Probabilitas dari kedua model:");
        for (int i = 0; i <
probabilities1.length; i++) {
            Log.d("ModelFusion",
"Kelas " + labels[i] +
                    " - Model 1: " +
probabilities1[i] +
                    ", Model 2: " +
probabilities2[i]);
        }

        // Hitung probabilitas dengan
simple averaging (tanpa pembobotan)
        float[] combinedProbabilities
= new float[probabilities1.length];
        for (int i = 0; i <
probabilities1.length; i++) {
            // Rata-rata sederhana
(0.5 untuk setiap model)
            combinedProbabilities[i]
= (probabilities1[i] +
probabilities2[i]) / 2.0f;

            // Log hasil averaging
            Log.d("ModelFusion",
"Kelas " + labels[i] +
                    " - Setelah
averaging: (" + probabilities1[i] +
                    " + " +
probabilities2[i] +
                    ")/2 = " +
combinedProbabilities[i]);
        }

        // Cari kelas dengan
probabilitas tertinggi
        int maxIndex = 0;
        float maxProb =
combinedProbabilities[0];
        for (int i = 1; i <
combinedProbabilities.length; i++) {
            if
(combinedProbabilities[i] > maxProb)
{
                maxProb =
combinedProbabilities[i];
                maxIndex = i;
            }
        }

        Log.d("ModelFusion", "Hasil
averaging - Prediksi: " +
                labels[maxIndex] + "
dengan probabilitas " + maxProb);

        float confidenceScore =
maxProb * 100;
        DecimalFormat df = new
DecimalFormat("#.##");
        String formattedScore =
df.format(confidenceScore);

        return labels[maxIndex] + "
(" + formattedScore + "%)";
    }

    private String
detailedAveragingOutput(float[]
probabilities1, float[]
probabilities2) {
        if (probabilities1.length !=
probabilities2.length) {
            return "Error: Jumlah
kelas berbeda antara dua model";
        }

        String[] labels =
loadLabels();
        StringBuilder output = new
StringBuilder();
        DecimalFormat df = new
DecimalFormat("#.##");

        // Bagian 1: Tampilkan
probabilitas dari kedua model untuk
setiap kelas
        for (int i = 0; i <
probabilities1.length; i++) {
            float confidence1 =
probabilities1[i] * 100;
            float confidence2 =
probabilities2[i] * 100;

            output.append(labels[i])
                    .append(" -
Model1 (")

.append(df.format(confidence1))
                    .append("%),
Model2 (")

.append(df.format(confidence2))
                    .append("%)\n");
        }

        output.append("\n"); //
Tambahkan baris kosong untuk
memisahkan bagian

        // Bagian 2: Tampilkan hasil
averaging untuk setiap kelas
        float[] combinedProbabilities
= new float[probabilities1.length];
        for (int i = 0; i <
probabilities1.length; i++) {
            float confidence1 =
probabilities1[i] * 100;
            float confidence2 =
probabilities2[i] * 100;
            combinedProbabilities[i]
= (probabilities1[i] +
probabilities2[i]) / 2.0f;
            float averageConfidence =
combinedProbabilities[i] * 100;

            output.append(labels[i])
                    .append(" -
Setelah Averaging (")

.append(df.format(confidence1))
                    .append("% + ")
```

```
.append(df.format(confidence2))
                    .append("%) / 2 =
")

.append(df.format(averageConfidence))
                    .append("%\n");
        }

        // Cari kelas dengan
probabilitas tertinggi
        int maxIndex = 0;
        float maxProb =
combinedProbabilities[0];
        for (int i = 1; i <
combinedProbabilities.length; i++) {
            if
(combinedProbabilities[i] > maxProb)
{
                maxProb =
combinedProbabilities[i];
                maxIndex = i;
            }
        }

        // Tambahkan hasil prediksi
akhir
        output.append("\nHasil
Prediksi: ")

.append(labels[maxIndex])
                .append(" (")

.append(df.format(maxProb * 100))
                .append("%)");

        return output.toString();
    }


    private float[]
getModelProbabilities(Bitmap bitmap,
String modelType) throws IOException
{
        try {
            TensorImage tensorImage =
preprocessImage(bitmap, modelType);

            switch (modelType) {
                case "MobileNetV3": {

BestMobileNetV3Small cake_model =
BestMobileNetV3Small.newInstance(getA
pplicationContext());

BestMobileNetV3Small.Outputs outputs
=
cake_model.process(tensorImage.getTen
sorBuffer());
                    TensorBuffer
probabilityBuffer =
outputs.getOutputFeature0AsTensorBuff
er();
                    float[]
probabilities =
probabilityBuffer.getFloatArray();

cake_model.close();
                    return
probabilities;

                }
                case "MobileNetV2": {
                    MobileNetV2
cake_model =
MobileNetV2.newInstance(getApplicatio
nContext());

MobileNetV2.Outputs outputs =
cake_model.process(tensorImage.getTen
sorBuffer());
                    TensorBuffer
probabilityBuffer =
outputs.getOutputFeature0AsTensorBuff
er();
                    float[]
probabilities =
probabilityBuffer.getFloatArray();

cake_model.close();
                    return
probabilities;

                }
                case
"EfficientNetB0": {

BestEfficientNetB0 cake_model =
BestEfficientNetB0.newInstance(getApp
licationContext());

BestEfficientNetB0.Outputs outputs =
cake_model.process(tensorImage.getTen
sorBuffer());
                    TensorBuffer
probabilityBuffer =
outputs.getOutputFeature0AsTensorBuff
er();
                    float[]
probabilities =
probabilityBuffer.getFloatArray();

cake_model.close();
                    return
probabilities;
                }
                default:

Log.e("getModelProbabilites() ",
"Model type unknown");
                    throw new
IOException("Model type tidak
dikenal: " + modelType);
            }
        } catch (IOException e) {

Log.e("getModelProbabilites() ",
"Failed to load model", e);
            throw new
IOException("Gagal mendapatkan
probabilitas model: " +
e.getMessage());
        }
    }
}
```

DetectionResult.java

```
package com.rifaldi.myapplication.ui;

import
android.annotation.SuppressLint;
import android.app.ProgressDialog;
import android.content.Context;
```

```
import android.content.Intent;
import android.graphics.Bitmap;
import
android.graphics.BitmapFactory;
import android.graphics.Canvas;
import
android.graphics.drawable.ColorDrawab
le;
import
android.graphics.drawable.Drawable;
import android.net.Uri;
import android.os.Bundle;
import android.util.Log;
import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.Window;
import android.view.WindowManager;
import
android.view.animation.Animation;
import
android.view.animation.AnimationUtils
;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.PopupWindow;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import
androidx.appcompat.app.AppCompatActiv
ity;
import
androidx.constraintlayout.widget.Cons
traintLayout;
import
androidx.core.content.ContextCompat;
import
androidx.core.content.FileProvider;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import
androidx.core.view.WindowInsetsCompat
;

import
com.google.firebase.database.DataSnap
shot;
import
com.google.firebase.database.Database
Error;
import
com.google.firebase.database.Database
Reference;
import
com.google.firebase.database.Firebase
Database;
import
com.google.firebase.database.ValueEve
ntListener;
import com.google.gson.Gson;
import com.rifaldi.myapplication.R;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.text.DecimalFormat;
import java.util.Objects;

public class DetectionResult extends
AppCompatActivity {
    private TextView tvResult,
tvConfidenceScore;
    private String hasilDeteksi,
confidenceScore;
    private String firstModelResult,
secondModelResult,
firstModelConfidenceScore,
secondModelConfidenceScore;
    private String
detailedAveragingScore;
    private Button btnGoToResep;
    private byte [] byteArrayImage;
    private String cakeName,
cakeDescription,
cakeShortDescription, cakeImageUrl,
recipesJson, tipsJson,
referencesJson;
    private static final float
MIN_GREAT_CONFIDENCE_SCORE = 10.0f;
    private static final float
MIN_GOOD_CONFIDENCE_SCORE = 10.0f;
    private static final float
MIN_BAD_CONFIDENCE_SCORE = 10.0f;
    private ConstraintLayout
clDetectionPageResult;
    private TextView
tvDetectionResult;
    private View vOverlay;

    @Override
    protected void onCreate(Bundle
savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity_dete
ction_result);

ViewCompat.setOnApplyWindowInsetsList
ener(findViewById(R.id.detectionResul
tPage), (v, insets) -> {
            Insets systemBars =
insets.getInsets(WindowInsetsCompat.T
ype.systemBars());

v.setPadding(systemBars.left,
systemBars.top, systemBars.right,
systemBars.bottom);
            return insets;
        });

        Window window = getWindow();

window.addFlags(WindowManager.LayoutP
arams.FLAG_DRAWS_SYSTEM_BAR_BACKGROUN
DS);

window.setStatusBarColor(ContextCompa
t.getColor(this, R.color.green));

        tvDetectionResult =
findViewById(R.id.tvDetectionResult);

        initializeOverlay();
        retrieveDataFromIntent();
        initializeListeners();
    }
```

```
    private void initializeOverlay(){
        vOverlay =
findViewById(R.id.overlay);
    }

    public void
initializeListeners(){

findViewById(R.id.ivBackDetectionResu
ltPage).setOnClickListener(v ->
onBackPressed() );

//
findViewById(R.id.tvDetectionResult).
setOnClickListener( v -> {
//
showSpecification(tvDetectionResult);
//        });


findViewById(R.id.ivShareDetectionRes
ult).setOnClickListener( v -> {
            Log.d("Detection_Result",
"Button share clicked!");

Toast.makeText(DetectionResult.this,
"Tunggu....\nSedang Memuat Gambar",
Toast.LENGTH_LONG).show();
            clDetectionPageResult =
findViewById(R.id.detectionResultPage
);

captureAndShareScreenshot(clDetection
PageResult, DetectionResult.this);
        });


findViewById(R.id.btnGoToResep).setOn
ClickListener(v -> {
            String tvCakeRecognition
=
capitalizeWords(capitalizeWords(tvRes
ult.getText().toString())

.replaceAll("\\s?Mungkin\\s?", "") //
Menghapus "Mungkin" dan spasi di
sekitarnya
                    .replace(")","")

.replace("(","");


            String
confidenceScoreString =
tvConfidenceScore.getText().toString(
);
            float confidenceScore;

            try {
                confidenceScore =
Float.parseFloat(confidenceScoreStrin
g.replaceAll("[^0-9.,]",
"").replace(',', '.').replaceAll("%",
""));
            } catch
(NumberFormatException e) {
                Log.e("Error",
"Invalid confidence score format: " +
confidenceScoreString, e);

Toast.makeText(DetectionResult.this,
"Error: Invalid confidence score
format", Toast.LENGTH_SHORT).show();
                return;
            }

            if (confidenceScore <
MIN_GOOD_CONFIDENCE_SCORE) {

Toast.makeText(getApplicationContext(
), "PASTIKAN AGAR HASIL DARI DETEKSI
SELALU DI ATAS "+
MIN_GREAT_CONFIDENCE_SCORE,
Toast.LENGTH_SHORT).show();
                Log.d("Value",
"Confidence Score: " +
confidenceScoreString);
            } else {
                ProgressDialog
progressDialog = new
ProgressDialog(DetectionResult.this);

progressDialog.setMessage("Memuat
data...");

progressDialog.show();


fetchKueByName(tvCakeRecognition, new
DetectionResult.OnDataFetchedListener
() {
                    @Override
                    public void
onDataFetched() {

progressDialog.dismiss();

                        Intent intent
= new Intent(DetectionResult.this,
CakeDescription.class);

intent.putExtra("cakeName",
cakeName);

intent.putExtra("cakeDescription",
cakeDescription);

intent.putExtra("cakeImageUrl",
cakeImageUrl);

intent.putExtra("cakeRecipes",
recipesJson);

intent.putExtra("cakeTips",
tipsJson);

intent.putExtra("cakeReferences",
referencesJson);

startActivity(intent);
                    }
                });
            }
        });
    }


@SuppressLint({"ClickableViewAccessib
ility", "DefaultLocale"})
    private void
showSpecification(TextView tv){
        LayoutInflater inflater =
```

```
(LayoutInflater)
getSystemService(LAYOUT_INFLATER_SERV
ICE);
        ViewGroup parent =
findViewById(android.R.id.content);
        View popupView =
inflater.inflate(R.layout.popup_model
s, parent, false);
        popupView.setElevation(12);

        TextView tvFirstModelResult =
popupView.findViewById(R.id.tvFirstMo
delResult);
        TextView
tvFirstModelConfidenceScore =
popupView.findViewById(R.id.tvFirstMo
delConfidenceScore);
        TextView tvSecondModelResult
=
popupView.findViewById(R.id.tvSecondM
odelResult);
        TextView
tvSecondModelConfidenceScore =
popupView.findViewById(R.id.tvSecondM
odelConfidenceScore);
//        Button
btnShowDetailedAveragingScore =
popupView.findViewById(R.id.btnShowDe
tailedAveraging);
//
//
btnShowDetailedAveragingScore.setOnCl
ickListener(new
View.OnClickListener() {
//            @Override
//            public void
onClick(View v) {
//                Intent intent = new
Intent(DetectionResult.this,
AveragingDetail.class);
//
intent.putExtra("DETAILED_AVERAGING_S
CORE", detailedAveragingScore);
//
startActivity(intent);
//            }
//        });

        // MODEL SCORE
        DecimalFormat df = new
DecimalFormat("#.##");
        String
firstModelResultFormatted =
df.format(Double.parseDouble(firstMod
elConfidenceScore));
        String
secondModelResultFormatted =
df.format(Double.parseDouble(secondMo
delConfidenceScore));


tvFirstModelResult.setText(firstModel
Result);

tvSecondModelResult.setText(secondMod
elResult);

tvFirstModelConfidenceScore.setText(S
tring.format("%s %%",
firstModelResultFormatted));

tvSecondModelConfidenceScore.setText(
String.format("%s %%",
secondModelResultFormatted));

        // MODEL WEIGHT
        TextView tvAverangingFormula2
=
popupView.findViewById(R.id.tvAverang
ingFormula2);

        String
tvWeightedAverangingFormula2Str = "("
+ firstModelResultFormatted + " + " +
secondModelResultFormatted + " ) /
2";

tvAverangingFormula2.setText(tvWeight
edAverangingFormula2Str);

        // AVERAGING SCORE
        TextView tvAveragingScore =
popupView.findViewById(R.id.tvAveragi
ngScore);

tvAveragingScore.setText(confidenceSc
ore);

        final PopupWindow popupWindow
= new PopupWindow(
                popupView,

LinearLayout.LayoutParams.WRAP_CONTEN
T,

LinearLayout.LayoutParams.WRAP_CONTEN
T
        );


popupWindow.setFocusable(true);

popupWindow.setOutsideTouchable(true)
;

popupWindow.setBackgroundDrawable(new
ColorDrawable());


vOverlay.setVisibility(View.VISIBLE);

        // Tambahkan
OnDismissListener

popupWindow.setOnDismissListener(() -
> {
            Animation fadeOut =
AnimationUtils.loadAnimation(tv.getCo
ntext(), R.anim.fade_out);

fadeOut.setAnimationListener(new
Animation.AnimationListener() {
                @Override
                public void
onAnimationStart(Animation animation)
{}

                @Override
                public void
onAnimationEnd(Animation animation) {

vOverlay.setVisibility(View.GONE);
```

```
                }

                @Override
                public void
onAnimationRepeat(Animation
animation) {}
            });

vOverlay.startAnimation(fadeOut);
        });

        Animation slideUp =
AnimationUtils.loadAnimation(tv.getCo
ntext(), R.anim.fade_in);

popupView.startAnimation(slideUp);


popupWindow.showAtLocation(tv.getRoot
View(), Gravity.CENTER, 0, 0);

        // Tambahkan click listener
untuk overlay
        vOverlay.setOnClickListener(v
-> {
            popupWindow.dismiss();
        });


popupView.setOnTouchListener((v1,
event) -> {
            popupWindow.dismiss();
            return true;
        });
    }

    public void
retrieveDataFromIntent(){
        hasilDeteksi =
Objects.requireNonNull(getIntent().ge
tStringExtra("HASIL_DETEKSI")).toUppe
rCase();
        confidenceScore =
Objects.requireNonNull(getIntent().ge
tStringExtra("HASIL_DETEKSI_CONFIDENC
E_SCORE")) + " %";
        byteArrayImage =
Objects.requireNonNull(getIntent().ge
tByteArrayExtra("GAMBAR_KUE"));

        firstModelResult =
Objects.requireNonNull(getIntent().ge
tStringExtra("FIRST_MODEL_RESULT"));
        firstModelConfidenceScore =
Objects.requireNonNull(getIntent().ge
tStringExtra("FIRST_MODEL_CONF_SCORE"
));
        secondModelResult =
Objects.requireNonNull(getIntent().ge
tStringExtra("SECOND_MODEL_RESULT"));
        secondModelConfidenceScore =
Objects.requireNonNull(getIntent().ge
tStringExtra("SECOND_MODEL_CONF_SCORE
"));
        detailedAveragingScore =
Objects.requireNonNull(getIntent().ge
tStringExtra("DETAILED_AVERAGING_SCOR
E"));
        setViews();
    }

    public void setViews(){
        tvResult =
findViewById(R.id.tvDetectionResult);
        tvConfidenceScore =
findViewById(R.id.tvConfidenceScore);

        Log.d("confidenceScore",
confidenceScore);

        ImageView imageView =
findViewById(R.id.ivCapturedImage);


tvConfidenceScore.setText(confidenceS
core);
        if (byteArrayImage != null) {
            Bitmap bitmap =
BitmapFactory.decodeByteArray(byteArr
ayImage, 0, byteArrayImage.length);

imageView.setImageBitmap(bitmap);
        }

        btnGoToResep =
findViewById(R.id.btnGoToResep);

        String pureConfidenceScore =
confidenceScore.replace("%","").repla
ce(",",".");
        Log.d("DETECTION_RESULT",
pureConfidenceScore);

        try{
            String detectionText;
            double confidenceScore =
Double.parseDouble(pureConfidenceScor
e);
            if (confidenceScore >
MIN_GREAT_CONFIDENCE_SCORE){
                detectionText =
hasilDeteksi;

btnGoToResep.setBackgroundColor(Conte
xtCompat.getColor(this,
R.color.dark_green));

btnGoToResep.setTextColor(ContextComp
at.getColor(this,
R.color.soft_black));

findViewById(R.id.tvDescriptionResult
).setVisibility(TextView.GONE);
            } else if
(confidenceScore >
MIN_GOOD_CONFIDENCE_SCORE){
                detectionText =
"Mungkin (" + hasilDeteksi + ")";

btnGoToResep.setTextColor(ContextComp
at.getColor(this,
R.color.soft_black));


findViewById(R.id.tvDescriptionResult
).setVisibility(TextView.GONE);
            } else if
(confidenceScore >
MIN_BAD_CONFIDENCE_SCORE){

btnGoToResep.setBackgroundResource(0)
;
```

```
btnGoToResep.setBackgroundResource(R.
drawable.bg_disable_button);

btnGoToResep.setTextColor(ContextComp
at.getColor(this, R.color.white));


tvConfidenceScore.setTextColor(Contex
tCompat.getColor(this,
R.color.warning));

tvResult.setTextColor(ContextCompat.g
etColor(this, R.color.warning));

                Drawable
disabledDrawableLeft =
ContextCompat.getDrawable(this,
R.drawable.button_white_recipe_icon);

btnGoToResep.setCompoundDrawablesWith
IntrinsicBounds(disabledDrawableLeft,
null, null, null);

                detectionText =
"Bukan Kue";

findViewById(R.id.tvDescriptionResult
).setVisibility(TextView.VISIBLE);
            } else {
                detectionText =
"ERROR";
            }


tvResult.setText(detectionText);

        } catch
(NumberFormatException e){

Log.e("ERROR_ON_DETECTION_RESULT",
"Invalid confidence score format: " +
pureConfidenceScore, e);

btnGoToResep.setEnabled(false);
        }

    }

    @Override
    public void onBackPressed(){
        super.onBackPressed();
    }


    private interface
OnDataFetchedListener {
        void onDataFetched();
    }


    public void fetchKueByName(String
inputNamaKue,
DetectionResult.OnDataFetchedListener
listener) {
        DatabaseReference
databaseReference =
FirebaseDatabase.getInstance(
                "https://kuecnn-
e1f7f-default-rtdb.asia-
southeast1.firebasedatabase.app/").ge
tReference();


databaseReference.child("katalog_kue"
).orderByChild("nama_kue").equalTo(in
putNamaKue)

.addListenerForSingleValueEvent(new
ValueEventListener() {
                    @Override
                    public void
onDataChange(@NonNull DataSnapshot
dataSnapshot) {

Log.d("SNAPSHOT",
dataSnapshot.toString());
                        if
(dataSnapshot.exists()) {
                            for
(DataSnapshot cakeSnapshot:
dataSnapshot.getChildren()) {
                                try {

Log.d("FIREBASE GET", "Processing
cake: " + cakeSnapshot.getKey());

cakeName =
cakeSnapshot.child("nama_kue").getVal
ue(String.class);

cakeShortDescription =
cakeSnapshot.child("penjelasan_singka
t").getValue(String.class);

cakeDescription =
cakeSnapshot.child("penjelasan").getV
alue(String.class);

cakeImageUrl =
cakeSnapshot.child("image_url").getVa
lue(String.class);


// Using GSON for pass data

Gson gson = new Gson();

recipesJson =
gson.toJson(cakeSnapshot.child("resep
").getValue());

tipsJson =
gson.toJson(cakeSnapshot.child("tips"
).getValue());

referencesJson =
gson.toJson(cakeSnapshot.child("refer
ensi").getValue());


Log.d("DetectionResult", cakeName);

Log.d("DetectionResult",
cakeDescription);

Log.d("DetectionResult",
cakeShortDescription);

Log.d("DetectionResult",
cakeImageUrl);
```

```
Log.d("DetectionResult",
referencesJson);

Log.d("DetectionResult", tipsJson);

Log.d("DetectionResult",
recipesJson);


// Call the listener after data is
successfully fetched

listener.onDataFetched();
                                }
catch (Exception e) {

Log.e("Firebase", "Error parsing cake
data", e);
                                }
                            }
                        } else {

System.out.println("Kue dengan nama "
+ inputNamaKue + " tidak
ditemukan.");

Toast.makeText(DetectionResult.this,

"Kue dengan nama " + inputNamaKue + "
tidak ditemukan.",

Toast.LENGTH_SHORT).show();
                        }
                    }

                    @Override
                    public void
onCancelled(@NonNull DatabaseError
databaseError) {

System.err.println("Database error: "
+ databaseError.getMessage());

Toast.makeText(DetectionResult.this,

"Error: " +
databaseError.getMessage(),

Toast.LENGTH_SHORT).show();
                    }
                });
    }

    private String
capitalizeWords(String str) {
        String[] words = str.split("
");
        StringBuilder capitalized =
new StringBuilder();
        for (String word : words) {
            if (!word.isEmpty()) {

capitalized.append(Character.toUpperC
ase(word.charAt(0)))

.append(word.substring(1).toLowerCase
()).append(" ");
            }
        }
        return
capitalized.toString().trim();
    }

    public void
captureAndShareScreenshot(View view,
Context context) {
        try {
            Bitmap bitmap =
getScreenshot(view);
            File file =
saveBitmap(bitmap, context);
            shareScreenshot(file,
context);
        } catch (IOException e) {
            Log.e("FAILED TO SHARE
IMAGE", "Error occurred while sharing
image", e);
        }
    }

    public Bitmap getScreenshot(View
view) {
        int width = view.getWidth();
        int height =
view.getHeight();

        Bitmap bitmap =
Bitmap.createBitmap(width, height,
Bitmap.Config.ARGB_8888);
        Canvas canvas = new
Canvas(bitmap);

        view.layout(0, 0, width,
height);
        view.draw(canvas);

        return bitmap;
    }

    public File saveBitmap(Bitmap
bitmap, Context context) throws
IOException {
        File cacheDir =
context.getCacheDir();
        File file = new
File(cacheDir, "screenshot.png");
        FileOutputStream fos = new
FileOutputStream(file);

bitmap.compress(Bitmap.CompressFormat
.PNG, 100, fos);
        fos.flush();
        fos.close();
        return file;
    }

    public void shareScreenshot(File
file, Context context) {
        Uri uri =
FileProvider.getUriForFile(context,
context.getPackageName() +
".fileprovider", file);

        Intent intent = new
Intent(Intent.ACTION_SEND);
        intent.setType("image/png");

intent.putExtra(Intent.EXTRA_STREAM,
uri);

intent.addFlags(Intent.FLAG_GRANT_REA
D_URI_PERMISSION);
```

```
context.startActivity(Intent.createChooser(intent, "Share Screenshot"));
    }
}
```

DevProfile.java

```
package com.rifaldi.myapplication.ui;

import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.Window;
import android.view.WindowManager;

import
androidx.appcompat.app.AppCompatActivity;
import
androidx.core.content.ContextCompat;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import
androidx.core.view.WindowInsetsCompat;

import com.rifaldi.myapplication.R;

public class DevProfile extends
AppCompatActivity {

    @Override
    protected void onCreate(Bundle
savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity_credits);

ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.creditsLayout), (v, insets) -> {
            Insets systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars());

v.setPadding(systemBars.left,
systemBars.top, systemBars.right,
systemBars.bottom);
            return insets;
        });

        Window window = getWindow();

window.addFlags(WindowManager.LayoutParams.FLAG_DRAWS_SYSTEM_BAR_BACKGROUNDS);

window.setStatusBarColor(ContextCompat.getColor(this, R.color.green));

    }

}
```

SplashActivity.java

```
package com.rifaldi.myapplication.ui;

import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.os.Looper;
import android.view.Window;
import android.view.WindowManager;

import
androidx.appcompat.app.AppCompatActivity;
import
androidx.core.content.ContextCompat;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import
androidx.core.view.WindowInsetsCompat;
import com.rifaldi.myapplication.R;

public class SplashActivity extends
AppCompatActivity {

    @Override
    protected void onCreate(Bundle
savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity_splash);

ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v,
insets) -> {
            Insets systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars());

v.setPadding(systemBars.left, 0,
systemBars.right, systemBars.bottom);
            return insets;
        });

        Window window = getWindow();

window.getDecorView().setSystemUiVisibility(WindowManager.LayoutParams.FLAG_DRAWS_SYSTEM_BAR_BACKGROUNDS);

window.setStatusBarColor(ContextCompat.getColor(this, R.color.white));

        Intent intent = new
Intent(this, AppLayout.class);

        new
Handler(Looper.getMainLooper()).postDelayed(() -> {
            startActivity(intent);
            finish();
        }, 1500);
    }
}
```

MyApp.java

```
package
com.rifaldi.myapplication.util;

import android.app.Application;
import
com.google.firebase.database.FirebaseDatabase;
```

```
public class MyApp extends
Application {

    @Override
    public void onCreate(){
        super.onCreate();

        FirebaseDatabase

.getInstance("https://kuecnn-e1f7f-
default-rtdb.asia-
southeast1.firebasedatabase.app/")

.setPersistenceEnabled(true);
    }
}
```

NetworkUtil.java

```
package
com.rifaldi.myapplication.util;

import android.content.Context;
import
android.net.ConnectivityManager;
import
android.net.NetworkCapabilities;

public class NetworkUtil{
    public static boolean
isInternetAvailable(Context context)
{
        ConnectivityManager
connectivityManager =
(ConnectivityManager)
context.getSystemService(Context.CONN
ECTIVITY_SERVICE);

        if (connectivityManager !=
null) {
            NetworkCapabilities
capabilities =
connectivityManager.getNetworkCapabil
ities(connectivityManager.getActiveNe
twork());
            return capabilities !=
null &&
(capabilities.hasTransport(NetworkCap
abilities.TRANSPORT_WIFI) ||
capabilities.hasTransport(NetworkCapa
bilities.TRANSPORT_CELLULAR));
        }
        return false;
    }
}
```

Activity_about.xml

```
<?xml version="1.0" encoding="utf-
8"?>
<androidx.constraintlayout.widget.Con
straintLayout
xmlns:android="http://schemas.android
.com/apk/res/android"
    android:id="@+id/aboutLayout"

xmlns:app="http://schemas.android.com
/apk/res-auto"

xmlns:tools="http://schemas.android.c
om/tools"

android:layout_width="match_parent"

android:layout_height="match_parent"
    android:background="@color/white"
    android:clickable="true"
    android:focusable="true"
    tools:context=".ui.About">

    <LinearLayout

android:id="@+id/llTopBarAbout"

android:layout_width="match_parent"

android:layout_height="wrap_content"
        android:paddingTop="12dp"
        android:paddingBottom="12dp"

app:layout_constraintLeft_toLeftOf="p
arent"

android:background="@drawable/bg_menu
_rectangle"

app:layout_constraintRight_toRightOf=
"parent"

app:layout_constraintTop_toTopOf="par
ent">

        <ImageView
            android:id="@+id/ivBack"

android:layout_width="26dp"

android:layout_height="26dp"

android:src="@drawable/button_black_b
ack_icon"

android:contentDescription="@string/b
utton_back_content_description"

android:layout_marginStart="12dp"

android:layout_marginEnd="24dp"
            android:layout_weight="1"

android:layout_gravity="center_vertic
al"
            />

        <TextView
            android:id="@+id/tvAbout"

android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:text="@string/tv_tentang_apli
kasi"
            android:textSize="20sp"

android:fontFamily="@font/overpass_bo
ld"

android:textColor="@color/soft_black"

android:gravity="center|center_vertic
al"

android:letterSpacing=".05"
```

```
            android:layout_weight="3"
android:layout_gravity="center_vertic
al" />

        <View
android:layout_width="32dp"
android:layout_height="32dp"
            android:layout_weight="1"
android:paddingStart="24dp"
            android:paddingEnd="12dp"
android:layout_gravity="center_vertic
al"/>

    </LinearLayout>


<androidx.swiperefreshlayout.widget.S
wipeRefreshLayout
        android:id="@+id/sRLAbout"
android:layout_width="match_parent"
        android:layout_height="400dp"
app:layout_constraintTop_toBottomOf="
@id/llTopBarAbout"
app:layout_constraintStart_toStartOf=
"parent"
app:layout_constraintEnd_toEndOf="par
ent">


<androidx.constraintlayout.widget.Con
straintLayout
android:layout_width="match_parent"
android:layout_height="wrap_content">

            <TextView
android:id="@+id/tvHeaderAbout"
android:layout_width="match_parent"
android:layout_height="wrap_content"
app:layout_constraintStart_toStartOf=
"parent"
app:layout_constraintEnd_toEndOf="par
ent"
app:layout_constraintTop_toTopOf="par
ent"
android:text="Penjelasan Singkat
Aplikasi"
android:textColor="@color/soft_black"
android:fontFamily="@font/overpass_bo
ld"
android:textSize="16sp"
android:letterSpacing="0.03"
android:layout_marginTop="24dp"
android:layout_marginStart="16dp"
android:layout_marginEnd="16dp"
                />

            <View
android:id="@+id/dividerHeaderAbout"
android:layout_width="match_parent"
android:layout_height="1dp"
android:background="#aaa"
app:layout_constraintTop_toBottomOf="
@id/tvHeaderAbout"
app:layout_constraintStart_toStartOf=
"parent"
app:layout_constraintEnd_toEndOf="par
ent"
android:layout_marginTop="4dp"
android:layout_marginStart="16dp"
android:layout_marginEnd="16dp"
                />

            <TextView
android:id="@+id/tvAppDescription"
android:layout_width="match_parent"
android:layout_height="wrap_content"
app:layout_constraintTop_toBottomOf="
@id/dividerHeaderAbout"
app:layout_constraintStart_toStartOf=
"parent"
app:layout_constraintEnd_toEndOf="par
ent"
android:text="@string/tv_app_descript
ion_text"
android:fontFamily="@font/overpass_re
gular"
android:textSize="15sp"
android:textColor="#555"
android:justificationMode="inter_word
"
android:lineHeight="28dp"
android:letterSpacing="0.03"
android:layout_marginTop="12dp"
```

```
android:layout_marginStart="16dp"

android:layout_marginEnd="16dp"
                />



</androidx.constraintlayout.widget.Co
nstraintLayout>


</androidx.swiperefreshlayout.widget.
SwipeRefreshLayout>


    <TextView

android:id="@+id/tvHeader2About"

android:layout_width="match_parent"

android:layout_height="wrap_content"

app:layout_constraintStart_toStartOf=
"parent"

app:layout_constraintEnd_toEndOf="par
ent"

app:layout_constraintTop_toBottomOf="
@id/sRLAbout"
        android:text="Teknologi yang
digunakan"

android:textColor="@color/soft_black"

android:fontFamily="@font/overpass_bo
ld"
        android:textSize="16sp"
        android:letterSpacing="0.03"

android:layout_marginTop="24dp"

android:layout_marginStart="16dp"

android:layout_marginEnd="16dp"
        />

    <View

android:id="@+id/dividerHeader2About"

android:layout_width="match_parent"
        android:layout_height="1dp"
        android:background="#aaa"

app:layout_constraintTop_toBottomOf="
@id/tvHeader2About"

app:layout_constraintStart_toStartOf=
"parent"

app:layout_constraintEnd_toEndOf="par
ent"

android:layout_marginTop="4dp"

android:layout_marginStart="16dp"
android:layout_marginEnd="16dp"
        />


<androidx.recyclerview.widget.Recycle
rView

android:id="@+id/recycler_view"

android:layout_width="match_parent"

android:layout_height="wrap_content"

android:layout_marginTop="12dp"

android:layout_marginStart="16dp"

android:layout_marginEnd="16dp"

app:layout_constraintTop_toBottomOf="
@id/dividerHeader2About"

app:layout_constraintStart_toStartOf=
"parent"

app:layout_constraintEnd_toEndOf="par
ent"

android:orientation="horizontal"
        />

    <View
        android:id="@+id/vOverlay"

android:layout_width="match_parent"

android:layout_height="match_parent"

android:background="#80000000"
        android:elevation="10dp"

app:layout_constraintStart_toStartOf=
"parent"
        android:visibility="gone" />



</androidx.constraintlayout.widget.Co
nstraintLayout>
```

Activity_about_dev.xml

```
<?xml version="1.0" encoding="utf-
8"?>
<androidx.constraintlayout.widget.Con
straintLayout
xmlns:android="http://schemas.android
.com/apk/res/android"
    android:id="@+id/aboutDevLayout"

xmlns:app="http://schemas.android.com
/apk/res-auto"

xmlns:tools="http://schemas.android.c
om/tools"

android:layout_width="match_parent"

android:layout_height="match_parent"
    android:background="@color/white"
    android:clickable="true"
    android:focusable="true"
```

```
    tools:context=".ui.About_dev">

    <LinearLayout

android:id="@+id/llTopBarAboutDev"

android:layout_width="match_parent"

android:layout_height="wrap_content"
        android:paddingTop="12dp"
        android:paddingBottom="12dp"

app:layout_constraintLeft_toLeftOf="p
arent"

android:background="@drawable/bg_menu
_rectangle"

app:layout_constraintRight_toRightOf=
"parent"

app:layout_constraintTop_toTopOf="par
ent">

        <ImageView
            android:id="@+id/ivBack"

android:layout_width="26dp"

android:layout_height="26dp"

android:src="@drawable/button_black_b
ack_icon"

android:contentDescription="@string/b
utton_back_content_description"

android:layout_marginStart="12dp"

android:layout_marginEnd="24dp"
            android:layout_weight="1"

android:layout_gravity="center_vertic
al"
            />

        <TextView

android:id="@+id/tvAboutDev"

android:layout_width="wrap_content"

android:layout_height="wrap_content"
            android:text="Profile
Pengembang"
            android:textSize="18sp"

android:fontFamily="@font/overpass_bo
ld"

android:textColor="@color/soft_black"

android:gravity="center|center_vertic
al"

android:letterSpacing=".05"
            android:layout_weight="3"

android:layout_gravity="center_vertic
al" />

        <View

android:layout_width="32dp"

android:layout_height="32dp"
            android:layout_weight="1"

android:paddingStart="24dp"
            android:paddingEnd="12dp"

android:layout_gravity="center_vertic
al"/>

    </LinearLayout>

    <LinearLayout
        android:id="@+id/llProfile"

android:layout_width="match_parent"

android:layout_height="wrap_content"

app:layout_constraintTop_toBottomOf="
@+id/llTopBarAboutDev"

app:layout_constraintStart_toStartOf=
"parent"

app:layout_constraintEnd_toEndOf="par
ent"

android:layout_marginStart="24dp"

android:layout_marginEnd="24dp"

android:layout_marginTop="24dp"

android:background="@color/white"

android:gravity="center_horizontal"

android:orientation="horizontal">

        <ImageView

android:id="@+id/ivProfilePicture"

android:layout_width="200dp"

android:layout_height="200dp"

android:layout_gravity="center"

android:contentDescription="Image Dev
Profile"

android:src="@drawable/dummy_image_ca
ptured" />

    </LinearLayout>

    <LinearLayout

android:id="@+id/llProfileDesc"

android:layout_width="match_parent"

android:layout_height="wrap_content"

app:layout_constraintTop_toBottomOf="
@id/llProfile"
```

```
android:layout_marginStart="24dp"

android:layout_marginEnd="20dp"

android:layout_marginTop="20dp"

android:orientation="vertical"

app:layout_constraintStart_toStartOf=
"parent"

app:layout_constraintEnd_toEndOf="par
ent"
        >

        <LinearLayout

android:id="@+id/llDevName"

android:layout_width="match_parent"

android:layout_height="wrap_content"

android:orientation="horizontal"

android:layout_marginTop="20dp">

            <TextView

android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:layout_weight="1"

android:fontFamily="@font/overpass_re
gular"

android:textSize="16sp"

android:textColor="#555"
                android:text="Dev
Name: "/>

            <TextView

android:id="@+id/tvDevName"

android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:layout_weight="1"

android:fontFamily="@font/overpass_bo
ld"

android:textSize="16sp"

android:textColor="@color/soft_black"
                android:text=""
                android:gravity="end"
                />

        </LinearLayout>

        <View

android:layout_width="match_parent"

android:layout_height="1dp"
            android:background="#555"

android:layout_marginTop="12dp"/>

        <LinearLayout
            android:id="@+id/llNIM"

android:layout_width="match_parent"

android:layout_height="wrap_content"

android:orientation="horizontal"

android:layout_marginTop="20dp">

            <TextView

android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:layout_weight="1"

android:fontFamily="@font/overpass_re
gular"

android:textSize="16sp"

android:textColor="#555"
                android:text="NIM:
"/>

            <TextView

android:id="@+id/tvDevNIM"

android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:layout_weight="1"

android:fontFamily="@font/overpass_bo
ld"

android:textSize="16sp"

android:textColor="@color/soft_black"
                android:text=""
                android:gravity="end"
                />

        </LinearLayout>

        <View

android:layout_width="match_parent"

android:layout_height="1dp"

android:background="@color/soft_black
"

android:layout_marginTop="12dp"/>

        <LinearLayout
            android:id="@+id/llClass"

android:layout_width="match_parent"
```

```
android:layout_height="wrap_content"

android:orientation="horizontal"

android:layout_marginTop="20dp">

            <TextView

android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:layout_weight="1"

android:fontFamily="@font/overpass_re
gular"

android:textSize="16sp"

android:textColor="#555"
                android:text="Class:
"/>

            <TextView

android:id="@+id/tvDevClass"

android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:layout_weight="1"

android:fontFamily="@font/overpass_bo
ld"

android:textSize="16sp"

android:textColor="@color/soft_black"
                android:text=""
                android:gravity="end"
                />

        </LinearLayout>

        <View

android:layout_width="match_parent"

android:layout_height="1dp"

android:background="@color/soft_black
"

android:layout_marginTop="12dp"/>

    </LinearLayout>

</androidx.constraintlayout.widget.Co
nstraintLayout>
```

activity_app_layout.xml

```
<?xml version="1.0" encoding="utf-
8"?>
<androidx.constraintlayout.widget.Con
straintLayout
xmlns:android="http://schemas.android
.com/apk/res/android"
    android:id="@+id/app_layout"

xmlns:app="http://schemas.android.com
/apk/res-auto"

xmlns:tools="http://schemas.android.c
om/tools"

android:layout_width="match_parent"

android:layout_height="match_parent"

android:background="@drawable/main_ac
tivity_background"
    tools:context=".ui.AppLayout">

    <TextView
        android:id="@+id/tvHeader"

android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:layout_marginStart="32dp"

android:layout_marginTop="48dp"

android:fontFamily="@font/overpass_ex
trabold"

android:text="@string/header_1"

android:textColor="@color/soft_black"
        android:textSize="26sp"
        android:textStyle="bold"
        android:letterSpacing=".05"

android:background="@drawable/bg_head
er"
        android:padding="10dp"

app:layout_constraintLeft_toLeftOf="p
arent"

app:layout_constraintBottom_toTopOf="
@id/tvHeader2"

app:layout_constraintTop_toTopOf="par
ent" />

    <TextView
        android:id="@+id/tvHeader2"

android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:text="@string/tvHeader2_txt"

android:fontFamily="@font/overpass_ex
trabold"
        android:textSize="26sp"
        android:letterSpacing=".05"

android:textColor="@color/soft_black"

android:background="@drawable/bg_head
er"

app:layout_constraintTop_toBottomOf="
@id/tvHeader"

app:layout_constraintLeft_toLeftOf="p
arent"
```

```
android:layout_marginStart="32dp"

app:layout_constraintBottom_toTopOf="
@id/tvHeader3"

android:layout_marginTop="4dp"
        android:padding="10dp"/>

    <TextView
        android:id="@+id/tvHeader3"

android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:text="@string/tvHeader3_txt"

android:fontFamily="@font/overpass_ex
trabold"
        android:textSize="26sp"

android:textColor="@color/soft_black"
        android:letterSpacing=".05"

android:background="@drawable/bg_head
er"

app:layout_constraintTop_toBottomOf="
@id/tvHeader2"

app:layout_constraintLeft_toLeftOf="p
arent"

android:layout_marginStart="32dp"

android:layout_marginTop="4dp"
        android:padding="10dp"/>

    <LinearLayout
        android:id="@+id/llMenu"

android:layout_width="match_parent"

android:layout_height="wrap_content"

android:layout_marginTop="56dp"
        android:elevation="10dp"

android:orientation="vertical"

app:layout_constraintRight_toRightOf=
"parent"

app:layout_constraintStart_toStartOf=
"parent"

app:layout_constraintTop_toBottomOf="
@id/tvHeader3">

        <Button

android:id="@+id/btnIdentify"

android:layout_width="280dp"

android:layout_height="wrap_content"

android:layout_gravity="end"

android:layout_marginBottom="12dp"
            android:layout_weight="1"

android:background="@drawable/bg_menu
_rectangle"

android:drawableStart="@drawable/butt
on_black_identify_icon"

android:drawablePadding="32dp"

android:fontFamily="@font/overpass_se
mibold"

android:textColor="@color/soft_black"

android:letterSpacing=".05"

android:gravity="start|center_vertica
l"

android:paddingStart="32dp"
            android:paddingTop="12dp"
            android:paddingEnd="0dp"

android:paddingBottom="12dp"

android:text="@string/identify"

android:textAllCaps="false"
            android:textSize="15sp"
/>


        <Button

android:id="@+id/btnCatalog"

android:layout_width="280dp"

android:layout_height="wrap_content"

android:layout_gravity="end"

android:layout_marginBottom="12dp"
            android:layout_weight="1"

android:background="@drawable/bg_menu
_rectangle"

android:drawableStart="@drawable/butt
on_black_catalog_icon"

android:drawablePadding="32dp"

android:fontFamily="@font/overpass_se
mibold"

android:textColor="@color/soft_black"

android:letterSpacing=".05"

android:gravity="start|center_vertica
l"

android:paddingStart="32dp"
            android:paddingTop="12dp"
            android:paddingEnd="0dp"

android:paddingBottom="12dp"

android:text="@string/katalog"
```

```
android:textAllCaps="false"
            android:textSize="15sp"
/>


        <Button

android:id="@+id/btnAbout"

android:layout_width="280dp"

android:layout_height="wrap_content"

android:layout_gravity="end"

android:layout_marginBottom="12dp"
            android:layout_weight="1"

android:background="@drawable/bg_menu
_rectangle"

android:drawableStart="@drawable/butt
on_black_about_icon"

android:drawablePadding="32dp"

android:fontFamily="@font/overpass_se
mibold"

android:textColor="@color/soft_black"

android:letterSpacing=".05"

android:gravity="start|center_vertica
l"

android:paddingStart="32dp"
            android:paddingTop="12dp"
            android:paddingEnd="0dp"

android:paddingBottom="12dp"

android:text="@string/about"

android:textAllCaps="false"
            android:textSize="15sp"
/>

        <Button

android:id="@+id/btnDevProfile"

android:layout_width="280dp"

android:layout_height="wrap_content"

android:layout_gravity="end"

android:layout_marginBottom="12dp"
            android:layout_weight="1"

android:background="@drawable/bg_menu
_rectangle"

android:drawableStart="@drawable/butt
on_black_credit_icon"

android:drawablePadding="32dp"

android:fontFamily="@font/overpass_se
mibold"

android:textColor="@color/soft_black"

android:letterSpacing=".05"

android:gravity="start|center_vertica
l"

android:paddingStart="32dp"
            android:paddingTop="12dp"
            android:paddingEnd="0dp"

android:paddingBottom="12dp"

android:text="@string/dev_profile"

android:textAllCaps="false"
            android:textSize="15sp"
/>


        <Button
            android:id="@+id/btnOut"

android:layout_width="280dp"

android:layout_height="wrap_content"

android:layout_gravity="end"

android:layout_marginBottom="12dp"
            android:layout_weight="1"

android:background="@drawable/bg_menu
_rectangle"

android:drawableStart="@drawable/butt
on_black_exit_icon"

android:drawablePadding="32dp"

android:fontFamily="@font/overpass_se
mibold"

android:textColor="@color/soft_black"

android:letterSpacing=".05"

android:gravity="start|center_vertica
l"

android:paddingStart="32dp"
            android:paddingTop="12dp"
            android:paddingEnd="0dp"

android:paddingBottom="12dp"

android:text="@string/out"

android:textAllCaps="false"
            android:textSize="15sp"
/>


    </LinearLayout>

<!--    <TextView-->
<!--
android:layout_width="match_parent"--
>
```

```
<!--
android:layout_height="wrap_content"-
->
<!--        android:text="© 2023
Rifaldi. All rights reserved."-->
<!--
android:textColor="@color/white"-->
<!--
android:textAlignment="center"-->
<!--        android:textSize="16sp"--
>
<!--
android:fontFamily="@font/overpass_re
gular"-->
<!--
app:layout_constraintBottom_toBottomO
f="parent"-->
<!--
app:layout_constraintStart_toStartOf=
"parent"-->
<!--
app:layout_constraintEnd_toEndOf="par
ent"-->
<!--
android:layout_marginBottom="48dp"/>-
->


    <View
        android:id="@+id/overlay"

android:layout_width="match_parent"

android:layout_height="match_parent"

android:background="#80000000"
        android:elevation="10dp"

app:layout_constraintStart_toStartOf=
"parent"
        android:visibility="gone" />



</androidx.constraintlayout.widget.Co
nstraintLayout>
```

activity_cake_description.xml

```
<?xml version="1.0" encoding="utf-
8"?>
<androidx.constraintlayout.widget.Con
straintLayout
xmlns:android="http://schemas.android
.com/apk/res/android"

xmlns:app="http://schemas.android.com
/apk/res-auto"

xmlns:tools="http://schemas.android.c
om/tools"

android:id="@+id/cakeDescriptionPage"

android:layout_width="match_parent"

android:layout_height="match_parent"

tools:context=".ui.CakeDescription"

android:background="@color/white">

    <LinearLayout

android:id="@+id/llTopBarCakeDescript
ionAbout"

android:layout_width="match_parent"

android:layout_height="wrap_content"
        android:paddingTop="12dp"
        android:paddingBottom="12dp"

app:layout_constraintLeft_toLeftOf="p
arent"

android:background="@drawable/bg_menu
_rectangle"

app:layout_constraintRight_toRightOf=
"parent"

app:layout_constraintTop_toTopOf="par
ent">

        <ImageView

android:id="@+id/ivBackCakeDescriptio
n"

android:layout_width="26dp"

android:layout_height="26dp"

android:src="@drawable/button_black_b
ack_icon"

android:contentDescription="@string/b
utton_back_content_description"

android:layout_marginStart="12dp"
            android:layout_weight="1"

android:layout_gravity="center_vertic
al"
            />

        <TextView

android:id="@+id/tvCakeName"

android:layout_width="wrap_content"

android:layout_height="wrap_content"
            android:text="Nama Kue"
            android:textSize="20sp"

android:fontFamily="@font/overpass_bo
ld"

android:textColor="@color/soft_black"

android:gravity="center|center_vertic
al"

android:letterSpacing=".05"
            android:layout_weight="3"

android:layout_gravity="center_vertic
al" />

        <ImageView
            android:id="@+id/ivShare"
```

```
android:layout_width="26dp"

android:layout_height="26dp"

android:src="@drawable/button_black_s
hare_icon"

android:contentDescription="button_sh
are"

android:layout_marginEnd="12dp"
            android:layout_weight="1"

android:layout_gravity="center_vertic
al"
            />

    </LinearLayout>

    <ScrollView

android:id="@+id/svCakeDescription"

android:layout_width="match_parent"

android:layout_height="wrap_content"

app:layout_constraintStart_toStartOf=
"parent"

app:layout_constraintEnd_toEndOf="par
ent"

app:layout_constraintTop_toBottomOf="
@id/llTopBarCakeDescriptionAbout"
        android:scrollbars="none"
        android:paddingBottom="24dp"
        >

        <LinearLayout

android:layout_width="match_parent"

android:layout_height="wrap_content"

android:orientation="vertical"
            android:padding="12dp">

            <ImageView

android:id="@+id/ivCake"

android:layout_width="match_parent"

android:layout_height="200dp"

android:layout_marginTop="16dp"

android:layout_marginBottom="16dp"

android:scaleType="centerCrop" />

            <LinearLayout

android:layout_width="match_parent"

android:layout_height="wrap_content"

android:orientation="horizontal">

                <TextView

android:id="@+id/tv_title_description
"

android:layout_width="0dp"

android:layout_height="match_parent"

android:letterSpacing="0.05"

android:layout_gravity="start"

android:gravity="center_vertical"

android:layout_weight="1"

android:text="Penjelasan"

android:textStyle="bold"

android:textSize="22sp"

android:textColor="@color/soft_black"

android:fontFamily="@font/overpass_bo
ld"

android:layout_marginBottom="8dp"/>

                <ImageView

android:id="@+id/ivCopyTvDescription"

android:layout_width="32dp"

android:layout_height="32dp"

android:src="@drawable/button_black_c
opy_icon"

android:contentDescription="ivButtonC
opyDescription"

android:padding="4dp"/>

            </LinearLayout>


            <View

android:layout_width="match_parent"

android:layout_height="1dp"

android:background="@color/soft_black
"/>

            <TextView

android:id="@+id/tv_description"

android:layout_width="match_parent"

android:layout_height="wrap_content"

android:layout_marginTop="6dp"

android:layout_marginBottom="18dp"
```

```
android:justificationMode="inter_word
"

android:textColor="#444"

android:lineHeight="26dp"

android:letterSpacing="0.03"

android:fontFamily="@font/overpass_re
gular"

android:textSize="15sp"/>

                <LinearLayout

android:layout_width="match_parent"

android:layout_height="wrap_content"

android:orientation="horizontal">

                    <TextView

android:layout_width="match_parent"

android:layout_height="wrap_content"

android:letterSpacing="0.05"

android:layout_gravity="start"

android:gravity="center_vertical"

android:layout_weight="1"

android:text="Resep"

android:textStyle="bold"

android:textSize="22sp"

android:textColor="@color/soft_black"

android:fontFamily="@font/overpass_bo
ld"

android:layout_marginBottom="8dp"
                        />

                    <ImageView

android:id="@+id/ivCopyCakeRecipes"

android:layout_width="32dp"

android:layout_height="32dp"

android:src="@drawable/button_black_c
opy_icon"

android:contentDescription="ivButtonC
opyDescription"

android:padding="4dp"/>

                </LinearLayout>

                <View

android:layout_width="match_parent"

android:layout_height="1dp"

android:background="@color/soft_black
"/>

                <!-- Bahan-bahan -->
                <TextView

android:layout_width="match_parent"

android:layout_height="wrap_content"

android:letterSpacing="0.05"

android:text="Bahan-bahan:"

android:textStyle="bold"

android:textSize="18sp"

android:fontFamily="@font/overpass_bo
ld"

android:textColor="@color/soft_black"

android:layout_marginTop="6dp"

android:layout_marginBottom="8dp"
                    />

                <TextView

android:id="@+id/tv_ingredients"

android:layout_width="match_parent"

android:layout_height="wrap_content"

android:layout_marginBottom="8dp"

android:layout_marginStart="6dp"

android:textColor="#444"

android:lineHeight="26dp"

android:letterSpacing="0.03"

android:fontFamily="@font/overpass_re
gular"

android:textSize="15sp"
                    />

                <!-- Bahan (Optional)
-->
            <TextView

android:id="@+id/tv_sauce_title"

android:layout_width="match_parent"

android:layout_height="wrap_content"

android:lineHeight="26dp"

android:letterSpacing="0.05"
                android:text="Bahan
tambahan:"
```

```
android:textStyle="bold"

android:textSize="18sp"

android:fontFamily="@font/overpass_bo
ld"

android:textColor="@color/soft_black"

android:layout_marginTop="6dp"

android:layout_marginBottom="8dp"

android:visibility="gone"
                />

            <TextView

android:id="@+id/tv_sauce_ingredients
"

android:layout_width="match_parent"

android:layout_height="wrap_content"

android:layout_marginBottom="8dp"

android:layout_marginStart="6dp"

android:textColor="#444"

android:lineHeight="26dp"

android:letterSpacing="0.03"

android:fontFamily="@font/overpass_re
gular"

android:textSize="15sp"

android:visibility="gone"
                />

                <!-- Cara Membuat -->
                <TextView

android:layout_width="match_parent"

android:layout_height="wrap_content"

android:letterSpacing="0.05"

android:text="Cara Membuat:"

android:textStyle="bold"

android:textSize="18sp"

android:fontFamily="@font/overpass_bo
ld"

android:textColor="@color/soft_black"

android:layout_marginBottom="8dp"/>

                <TextView

android:id="@+id/tv_steps"

android:layout_width="match_parent"

android:layout_height="wrap_content"

android:textColor="#444"

android:lineHeight="28dp"

android:letterSpacing="0.03"

android:textSize="15sp"

android:fontFamily="@font/overpass_re
gular"/>



            <LinearLayout

android:id="@+id/llCakeTips"

android:layout_width="match_parent"

android:layout_height="wrap_content"

android:orientation="horizontal">

                <TextView

android:layout_width="match_parent"

android:layout_height="wrap_content"

android:letterSpacing="0.05"

android:layout_gravity="start"

android:gravity="center_vertical"

android:layout_weight="1"

android:text="Tips"

android:textStyle="bold"

android:textSize="22sp"

android:fontFamily="@font/overpass_bo
ld"

android:textColor="@color/soft_black"

android:layout_marginBottom="8dp"/>

                <ImageView

android:id="@+id/ivCopyCakeTips"

android:layout_width="32dp"

android:layout_height="32dp"

android:src="@drawable/button_black_c
opy_icon"

android:contentDescription="ivButtonC
opyCakeTips"

android:padding="4dp"/>

            </LinearLayout>

            <View
```

```
android:layout_width="match_parent"

android:layout_height="1dp"

android:background="@color/soft_black
"/>

            <TextView

android:id="@+id/tv_tips"

android:layout_width="match_parent"

android:layout_height="wrap_content"

android:layout_marginTop="6dp"

android:layout_marginBottom="18dp"

android:justificationMode="inter_word
"

android:textColor="#444"

android:lineHeight="26dp"

android:letterSpacing="0.03"

android:fontFamily="@font/overpass_re
gular"

android:textSize="15sp"
                />

        <LinearLayout

android:id="@+id/llCakeReferences"

android:layout_width="match_parent"

android:layout_height="wrap_content"

android:orientation="horizontal">

            <TextView

android:layout_width="match_parent"

android:layout_height="wrap_content"

android:letterSpacing="0.05"

android:layout_gravity="start"

android:gravity="center_vertical"

android:layout_weight="1"

android:text="Referensi"

android:textStyle="bold"

android:textSize="22sp"

android:fontFamily="@font/overpass_bo
ld"

android:textColor="@color/soft_black"

android:layout_marginBottom="8dp"/>

            <ImageView

android:id="@+id/ivCopyCakeReferences
"

android:layout_width="32dp"

android:layout_height="32dp"

android:src="@drawable/button_black_c
opy_icon"

android:contentDescription="ivButtonC
opyCakeTips"

android:padding="4dp"/>

        </LinearLayout>

            <View

android:layout_width="match_parent"

android:layout_height="1dp"

android:background="@color/soft_black
"/>

            <TextView

android:id="@+id/tv_references"

android:layout_width="match_parent"

android:layout_height="wrap_content"

android:clickable="true"

android:layout_marginTop="6dp"

android:layout_marginBottom="18dp"

android:textColor="#444"

android:lineHeight="26dp"

android:letterSpacing="0.03"

android:fontFamily="@font/overpass_re
gular"

android:textSize="15sp"
                />

        </LinearLayout>
    </ScrollView>

    <View

android:id="@+id/vOverlayCakeDescript
ion"

android:layout_width="match_parent"

android:layout_height="match_parent"

android:background="#80000000"
        android:elevation="10dp"

app:layout_constraintStart_toStartOf=
```

```
"parent"

app:layout_constraintEnd_toEndOf="par
ent"

app:layout_constraintTop_toTopOf="par
ent"

app:layout_constraintBottom_toBottomO
f="parent"
        android:visibility="gone" />

    <ProgressBar

android:id="@+id/progressBarCakeDescr
iption"
        android:layout_width="200dp"
        android:layout_height="200dp"

android:layout_centerInParent="true"

app:layout_constraintStart_toStartOf=
"parent"

app:layout_constraintEnd_toEndOf="par
ent"

app:layout_constraintTop_toTopOf="par
ent"

app:layout_constraintBottom_toBottomO
f="parent"
        android:elevation="15dp"
        android:visibility="gone"
        android:indeterminate="true"

android:indeterminateTint="@color/sof
t_black" />

</androidx.constraintlayout.widget.Co
nstraintLayout>
```

activity_catalog.xml

```
<?xml version="1.0" encoding="utf-
8"?>
<androidx.constraintlayout.widget.Con
straintLayout
xmlns:android="http://schemas.android
.com/apk/res/android"

xmlns:app="http://schemas.android.com
/apk/res-auto"

xmlns:tools="http://schemas.android.c
om/tools"
    android:id="@+id/catalogLayout"

android:layout_width="match_parent"

android:layout_height="match_parent"
    android:background="@color/white"
    tools:context=".ui.Catalog">

    <LinearLayout

android:id="@+id/llTopBarCatalog"

android:layout_width="match_parent"

android:layout_height="wrap_content"
        android:paddingTop="12dp"
        android:paddingBottom="12dp"

app:layout_constraintLeft_toLeftOf="p
arent"

android:background="@drawable/bg_menu
_rectangle"

app:layout_constraintRight_toRightOf=
"parent"

app:layout_constraintTop_toTopOf="par
ent">

        <ImageView
            android:id="@+id/ivBack"

android:layout_width="26dp"

android:layout_height="26dp"

android:src="@drawable/button_black_b
ack_icon"

android:contentDescription="@string/b
utton_back_content_description"

android:layout_marginStart="12dp"

android:layout_marginEnd="24dp"
            android:layout_weight="1"

android:layout_gravity="center_vertic
al"
            />

        <TextView

android:id="@+id/tvAboutDev"

android:layout_width="wrap_content"

android:layout_height="wrap_content"
            android:text="Katalog
Kue"
            android:textSize="20sp"

android:fontFamily="@font/overpass_bo
ld"

android:textColor="@color/soft_black"

android:gravity="center|center_vertic
al"

android:letterSpacing=".05"
            android:layout_weight="3"

android:layout_gravity="center_vertic
al" />

        <View

android:layout_width="32dp"

android:layout_height="32dp"
            android:layout_weight="1"

android:paddingStart="24dp"
            android:paddingEnd="12dp"
```

```
android:layout_gravity="center_vertic
al"/>

    </LinearLayout>


<androidx.appcompat.widget.SearchView
        android:id="@+id/svCatalog"

android:layout_width="match_parent"

android:layout_height="wrap_content"
        android:minHeight="56dp"
        android:maxHeight="56dp"

android:background="@drawable/bg_sear
ch_view"

app:layout_constraintTop_toBottomOf="
@id/llTopBarCatalog"

app:layout_constraintStart_toStartOf=
"parent"

app:layout_constraintEnd_toEndOf="par
ent"
        android:paddingStart="12dp"
        android:paddingEnd="12dp"
        android:paddingTop="4dp"
        android:paddingBottom="4dp"

android:layout_marginStart="12dp"

android:layout_marginEnd="12dp"

android:layout_marginTop="12dp"

android:layout_marginBottom="12dp"

app:queryBackground="@color/white"
        app:queryHint="Cari Kue..."
        />


<androidx.recyclerview.widget.Recycle
rView
        android:id="@+id/rvCakeItem"

android:layout_width="match_parent"
        android:layout_height="0dp"

app:layout_constraintTop_toBottomOf="
@id/svCatalog"

app:layout_constraintStart_toStartOf=
"parent"

app:layout_constraintEnd_toEndOf="par
ent"

app:layout_constraintBottom_toBottomO
f="parent"
        android:paddingStart="12dp"
        android:paddingEnd="12dp"

android:layout_marginTop="20dp"
        />

    <View

android:id="@+id/overlayCatalog"

android:layout_width="match_parent"

android:layout_height="match_parent"

android:background="#80000000"
        android:elevation="10dp"

app:layout_constraintStart_toStartOf=
"parent"
        android:visibility="gone" />

</androidx.constraintlayout.widget.Co
nstraintLayout>
```

Activity_credits.xml

```
<?xml version="1.0" encoding="utf-
8"?>
<androidx.constraintlayout.widget.Con
straintLayout
xmlns:android="http://schemas.android
.com/apk/res/android"

xmlns:app="http://schemas.android.com
/apk/res-auto"

xmlns:tools="http://schemas.android.c
om/tools"
    android:id="@+id/creditsLayout"

android:layout_width="match_parent"

android:layout_height="match_parent"
    android:background="@color/white"
    tools:context=".ui.DevProfile">

    <LinearLayout

android:id="@+id/llTopBarCredits"

android:layout_width="match_parent"

android:layout_height="wrap_content"
        android:paddingTop="12dp"
        android:paddingBottom="12dp"

app:layout_constraintLeft_toLeftOf="p
arent"

android:background="@drawable/bg_menu
_rectangle"

app:layout_constraintRight_toRightOf=
"parent"

app:layout_constraintTop_toTopOf="par
ent">

        <ImageView
            android:id="@+id/ivBack"

android:layout_width="26dp"

android:layout_height="26dp"

android:src="@drawable/button_black_b
ack_icon"

android:contentDescription="@string/b
utton_back_content_description"
```

```
android:layout_marginStart="12dp"

android:layout_marginEnd="24dp"
            android:layout_weight="1"

android:layout_gravity="center_vertic
al"
            />

        <TextView

android:id="@+id/tvAboutDev"

android:layout_width="wrap_content"

android:layout_height="wrap_content"
            android:text="Terima
Kasih"
            android:textSize="20sp"

android:layout_marginStart="8dp"

android:layout_marginEnd="0dp"

android:fontFamily="@font/overpass_bo
ld"

android:textColor="@color/soft_black"

android:gravity="center|center_vertic
al"

android:letterSpacing=".05"
            android:layout_weight="3"

android:layout_gravity="center_vertic
al" />

        <View

android:layout_width="32dp"

android:layout_height="32dp"
            android:layout_weight="1"

android:paddingStart="24dp"
            android:paddingEnd="12dp"

android:layout_gravity="center_vertic
al"/>

    </LinearLayout>

<!--
<io.github.sceneview.SceneView-->
<!--
android:id="@+id/sceneView"-->
<!--
android:layout_width="match_parent"--
>
<!--
android:layout_height="match_parent"
/>-->

<!--    <Button-->
<!--
android:id="@+id/placeObjectButton"--
>
<!--
android:layout_width="wrap_content"--
>
<!--
android:layout_height="wrap_content"-
->
<!--        android:text="Place
Object"-->
<!--
android:layout_margin="16dp"-->
<!--
app:layout_constraintBottom_toBottomO
f="parent"-->
<!--
app:layout_constraintEnd_toEndOf="par
ent" />-->



</androidx.constraintlayout.widget.Co
nstraintLayout>
```

activity_detection_page.xml

```
<?xml version="1.0" encoding="utf-
8"?>
<androidx.constraintlayout.widget.Con
straintLayout
xmlns:android="http://schemas.android
.com/apk/res/android"

xmlns:app="http://schemas.android.com
/apk/res-auto"

xmlns:tools="http://schemas.android.c
om/tools"
    android:id="@+id/detectionPage"

android:layout_width="match_parent"

android:layout_height="match_parent"
    tools:context=".ui.DetectionPage"

android:background="@color/white">

    <LinearLayout

android:id="@+id/llTopBarDetectionPag
e"

android:layout_width="match_parent"

android:layout_height="wrap_content"
        android:paddingTop="12dp"
        android:paddingBottom="12dp"

app:layout_constraintLeft_toLeftOf="p
arent"

android:background="@drawable/bg_menu
_rectangle"

app:layout_constraintRight_toRightOf=
"parent"

app:layout_constraintTop_toTopOf="par
ent">

        <ImageView

android:id="@+id/ivBackDetectionPage"

android:layout_width="26dp"

android:layout_height="26dp"
```

```
android:src="@drawable/button_black_back_icon"

android:contentDescription="@string/button_back_content_description"

android:layout_marginStart="12dp"
            android:layout_weight="1"

android:layout_gravity="center_vertical"
            />

        <TextView

android:id="@+id/tvDetectionPage"

android:layout_width="wrap_content"

android:layout_height="wrap_content"
            android:text="Halaman Deteksi"
            android:textSize="20sp"

android:fontFamily="@font/overpass_bold"

android:textColor="@color/soft_black"

android:gravity="center|center_vertical"

android:letterSpacing=".05"
            android:layout_weight="3"

android:layout_gravity="center_vertical" />

        <ImageView

android:id="@+id/ivNothing"

android:layout_width="26dp"

android:layout_height="26dp"

android:contentDescription="button_share"

android:layout_marginEnd="12dp"
            android:layout_weight="1"

android:layout_gravity="center_vertical"
            />

    </LinearLayout>

    <ImageView

android:id="@+id/ivImageCaptured"

android:layout_width="match_parent"
        android:layout_height="300dp"

app:layout_constraintStart_toStartOf="parent"

app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintTop_toBottomOf="@id/llTopBarDetectionPage"

android:scaleType="centerCrop"

android:contentDescription="Image Captured"

android:src="@drawable/dummy_image_captured"
        />

    <LinearLayout

android:id="@+id/llCapturedOptions"

android:layout_width="match_parent"

android:layout_height="wrap_content"

app:layout_constraintStart_toStartOf="parent"

app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintTop_toBottomOf="@id/ivImageCaptured"

app:layout_constraintBottom_toBottomOf="parent"

android:orientation="vertical"
        android:gravity="center"
        >

        <Button

android:id="@+id/btnGetFromCamera"

android:layout_width="240dp"

android:layout_height="wrap_content"
            android:paddingTop="18dp"

android:paddingBottom="18dp"

android:paddingStart="18dp"
            android:paddingEnd="8dp"

android:drawableStart="@drawable/button_black_camera_icon"

android:fontFamily="@font/overpass_bold"
            android:text="Ambil Dari Kamera"
            android:textSize="14sp"

android:textColor="@color/soft_black"

android:backgroundTint="@color/green"

android:layout_marginBottom="20dp"/>

        <Button

android:id="@+id/btnGetFromGallery"

android:layout_width="240dp"
```

```
android:layout_height="wrap_content"
            android:paddingTop="18dp"

android:paddingBottom="18dp"

android:paddingStart="18dp"
            android:paddingEnd="8dp"

android:drawableStart="@drawable/butt
on_black_gallery_icon"

android:fontFamily="@font/overpass_bo
ld"
            android:text="Ambil Dari
Gallery"
            android:textSize="14sp"

android:textColor="@color/soft_black"

android:backgroundTint="@color/green"
/>

    </LinearLayout>

</androidx.constraintlayout.widget.Co
nstraintLayout>
```

activity_detection_page.xml

```
<?xml version="1.0" encoding="utf-
8"?>
<androidx.constraintlayout.widget.Con
straintLayout
xmlns:android="http://schemas.android
.com/apk/res/android"

xmlns:app="http://schemas.android.com
/apk/res-auto"

xmlns:tools="http://schemas.android.c
om/tools"

android:id="@+id/detectionResultPage"

android:layout_width="match_parent"

android:layout_height="match_parent"
    android:background="@color/white"

app:layout_constraintTop_toTopOf="par
ent"

app:layout_constraintEnd_toEndOf="par
ent"

app:layout_constraintBottom_toBottomO
f="parent"

app:layout_constraintStart_toStartOf=
"parent"

tools:context=".ui.DetectionResult">

    <LinearLayout

android:id="@+id/llTopBarDetectionRes
ult"

android:layout_width="match_parent"

android:layout_height="wrap_content"
        android:paddingTop="12dp"
        android:paddingBottom="12dp"

app:layout_constraintLeft_toLeftOf="p
arent"

android:background="@drawable/bg_menu
_rectangle"

app:layout_constraintRight_toRightOf=
"parent"

app:layout_constraintTop_toTopOf="par
ent">

        <ImageView

android:id="@+id/ivBackDetectionResul
tPage"

android:layout_width="26dp"

android:layout_height="26dp"

android:src="@drawable/button_black_b
ack_icon"

android:contentDescription="@string/b
utton_back_content_description"

android:layout_marginStart="12dp"
            android:layout_weight="1"

android:layout_gravity="center_vertic
al"
            />

        <TextView

android:id="@+id/tvDetectionResultTop
Bar"

android:layout_width="wrap_content"

android:layout_height="wrap_content"
            android:text="Hasil
Deteksi"
            android:textSize="20sp"

android:fontFamily="@font/overpass_bo
ld"

android:textColor="@color/soft_black"

android:gravity="center|center_vertic
al"

android:letterSpacing=".05"
            android:layout_weight="3"

android:layout_gravity="center_vertic
al" />

        <ImageView

android:id="@+id/ivShareDetectionResu
lt"

android:layout_width="28dp"

android:layout_height="28dp"
```

```
android:contentDescription="button_sh
are"

android:src="@drawable/button_black_s
hare_icon"

android:layout_marginEnd="12dp"
            android:layout_weight="1"

android:layout_gravity="center_vertic
al"
            />

    </LinearLayout>

    <ImageView

android:id="@+id/ivCapturedImage"

android:layout_width="match_parent"
        android:layout_height="280dp"

app:layout_constraintStart_toStartOf=
"parent"

app:layout_constraintEnd_toEndOf="par
ent"

app:layout_constraintTop_toBottomOf="
@id/llTopBarDetectionResult"

android:scaleType="centerCrop"

android:contentDescription="Image
Captured"

android:src="@drawable/dummy_image_ca
ptured"

android:layout_marginBottom="16dp"
        />

    <TextView

android:id="@+id/tvHeaderClassifiedIm
age"

android:layout_width="match_parent"

android:layout_height="wrap_content"

app:layout_constraintStart_toStartOf=
"parent"

app:layout_constraintEnd_toEndOf="par
ent"

app:layout_constraintTop_toBottomOf="
@id/ivCapturedImage"

app:layout_constraintBottom_toTopOf="
@+id/tvDetectionResult"
        android:letterSpacing="0.05"

android:text="Diklasifikasikan
sebagai: "

android:textColor="@color/soft_black"
        android:textSize="20sp"

android:textAlignment="center"

android:fontFamily="@font/overpass_se
mibold"

android:layout_marginTop="24dp"
        />

    <TextView

android:id="@+id/tvDetectionResult"

android:layout_width="match_parent"

android:layout_height="wrap_content"

app:layout_constraintTop_toBottomOf="
@id/tvHeaderClassifiedImage"

app:layout_constraintEnd_toEndOf="par
ent"

app:layout_constraintStart_toStartOf=
"parent"

android:fontFamily="@font/overpass_ex
trabold"

android:layout_marginTop="6dp"
        android:letterSpacing="0.02"

android:textColor="@color/dark_green"
        android:text="HASIL DETEKSI"
        android:textSize="24sp"

android:textAlignment="center"/>

    <TextView

android:id="@+id/tvDescriptionResult"

android:layout_width="match_parent"

android:layout_height="wrap_content"
        android:textSize="14sp"
        android:text="INFO:\nGambar
yang ditangkap mungkin bukan
kue\natau model belum mengenali kue
tersebut\n(Sebaiknya lakukan
pengambilan ulang gambar)"

android:textColor="@color/warning"

android:fontFamily="@font/overpass_re
gular"

app:layout_constraintStart_toStartOf=
"parent"

app:layout_constraintEnd_toEndOf="par
ent"

app:layout_constraintTop_toBottomOf="
@id/tvConfidenceScore"

android:layout_marginTop="8dp"

android:textAlignment="center"
        android:visibility="gone"
        />
```

```
    <TextView

android:id="@+id/tvHeaderConfidenceSc
ore"

android:layout_width="match_parent"

android:layout_height="wrap_content"
        android:text="Tingkat
Keyakinan Model: "

app:layout_constraintTop_toBottomOf="
@id/tvDetectionResult"

app:layout_constraintStart_toStartOf=
"parent"

app:layout_constraintEnd_toEndOf="par
ent"

app:layout_constraintBottom_toTopOf="
@id/tvConfidenceScore"
        android:letterSpacing="0.05"

android:textColor="@color/soft_black"
        android:textSize="20sp"

android:textAlignment="center"

android:fontFamily="@font/overpass_se
mibold"

android:layout_marginTop="24dp"/>

    <TextView

android:id="@+id/tvConfidenceScore"

android:layout_width="match_parent"

android:layout_height="wrap_content"

app:layout_constraintTop_toBottomOf="
@id/tvHeaderConfidenceScore"

app:layout_constraintEnd_toEndOf="par
ent"

app:layout_constraintStart_toStartOf=
"parent"

android:fontFamily="@font/overpass_ex
trabold"

android:layout_marginTop="6dp"
        android:letterSpacing="0.02"

android:textColor="@color/dark_green"
        android:text="CONFIDENCE
SCORE"
        android:textSize="24sp"

android:textAlignment="center"/>

    <Button

android:id="@+id/btnGoToResep"

android:layout_width="wrap_content"

android:layout_height="wrap_content"

app:layout_constraintStart_toStartOf=
"parent"

app:layout_constraintEnd_toEndOf="par
ent"

app:layout_constraintTop_toBottomOf="
@id/tvConfidenceScore"

app:layout_constraintBottom_toBottomO
f="parent"

android:drawableStart="@drawable/butt
on_black_recipe_icon"

android:fontFamily="@font/overpass_bo
ld"
        android:text="Lihat Resep
Kue"
        android:textSize="14sp"

android:textColor="@color/soft_black"

android:layout_marginStart="32dp"

android:layout_marginEnd="32dp"

android:layout_marginTop="54dp"

android:drawablePadding="12dp"

android:background="@drawable/bg_menu
_rectangle"
        android:paddingTop="18dp"
        android:paddingBottom="18dp"
        android:paddingStart="18dp"
        android:paddingEnd="18dp"
        android:letterSpacing=".1"
        android:padding="10dp" />

    <View
        android:id="@+id/overlay"

android:layout_width="match_parent"

android:layout_height="match_parent"

android:background="#80000000"
        android:elevation="10dp"

app:layout_constraintStart_toStartOf=
"parent"
        android:visibility="gone" />


</androidx.constraintlayout.widget.Co
nstraintLayout>
```

KARTU MONITORING BIMBINGAN

MAHASISWA PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS MUHAMMADIYAH PAREPARE

PROPOSAL

Mahasiswa : Rifaldi	**Pembimbing I : Ade Hastuty, ST., S.Kom.,MT.**
NIM : 218280017	**Pembimbing II : Ahmad Selao, S.TP., M.Sc.**
Judul Skripsi : APLIKASI PENGENALAN KUE TRADISIONAL BUGIS BERBASIS ANDROID	

ARAHAN PEMBIMBING I	HARI/TGL & PARAF PEMBIMBING	ARAHAN PEMBIMBING II	HARI/TGL & PARAF PEMBIMBING
Konsultasi 1 Perbaikan Bab I - Latar Belakang - Rumusan Masalah		Konsultasi 1 - jurnal & tujuan penelitis disikron kan	
Konsultasi 2 Perbaikan Bab II - Kajian Teori - Kerangka Pikir		Konsultasi 2 urutkan penelitis yg menjadi dasar ide & perbedaanya	
Konsultasi 3 Perbaikan Bab III - Use case		Konsultasi 3 Cantumkan dasar/ perhitungan CNN di algoritmakan & pseudocode nya	
Konsultasi 4 Acc Seminar Proposal		Konsultasi 4 Tambahkan CNN tahapannya dalam metode deteksi nya.	
Konsultasi 5		Konsultasi 5 Acc sekali di perbaiki	

Lanjut ke halaman sebelah...

Perhatian :

1. Mahasiswa wajib konsultasi minimal 5 kali
2. Kartu ini wajib dibawa oleh mahasiswa disetiap konsultasi dan diisi oleh Pembimbing

KARTU MONITORING BIMBINGAN

MAHASISWA PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS MUHAMMADIYAH PAREPARE

SKRIPSI

Mahasiswa : RIFALDI	Pembimbing I : ADE HASTUTY, ST., S.KOM., MT.
NIM : 218280017	Pembimbing II : AHMAD SELAO, S.TP., M.SC.
Judul Skripsi : APLIKASI PENGENALAN KUE TRADISIONAL BUGIS MENGGUNAKAN METODE CONVOLUTIONAL NEURAL NETWORK (CNN) BERBASIS ANDROID	

ARAHAN PEMBIMBING I	HARI/TGL & PARAF PEMBIMBING	ARAHAN PEMBIMBING II	HARI/TGL & PARAF PEMBIMBING
Konsultasi 1 - Perbaikan beberapa penulisan. -		Konsultasi 1 Pertanyaan analisis. Data piksel sebagai input pada CNN	
Konsultasi 2 - Dokumentasi pengujian aplikasi - Penambahan sumber/referensi pada Aplikasi		Konsultasi 2 Bahas hasil yang di peroleh berdasarkan code program	
Konsultasi 3 Teori wajib ① Dasar pemrograman 11, 10, 7, 8		Konsultasi 3 Hasil validasi dengan tools CNN & output recognition kue tradisional	
Konsultasi 4 Teori → Bab 2 ② UML ③ Pengujian program		Konsultasi 4 Analisa berdasarkan piksel CNN terhadap Kitra kue tradisional	
Konsultasi 5		Konsultasi 5 Susun secara sistematis abstraksi, LB - kesimpulan	

Lanjut ke halaman sebelah...

Perhatian :

1. Mahasiswa wajib konsultasi minimal 5 kali
2. Kartu ini wajib dibawa oleh mahasiswa disetiap konsultasi dan diisi oleh Pembimbing
3. Kartu ini wajib dilampirkan pada laporan skripsi dan menjadi salah satu persyaratan untuk ikut seminar proposal/ujian skripsi
4. Kartu ini dicetak di atas kertas karton A4 berwarna Hijau muda dan dicetak timbal balik