

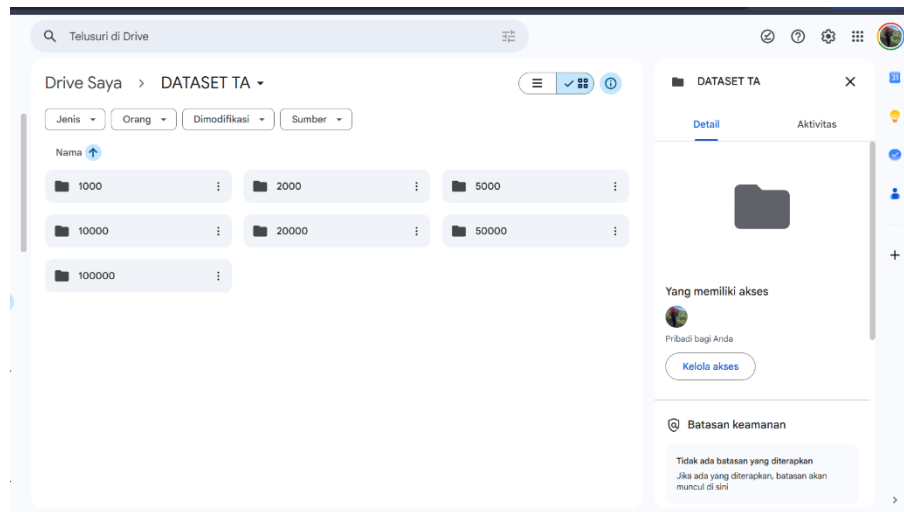
LAMPIRAN

Lampiran 1. Dataset Penelitian

a. Struktur Folder Dataset

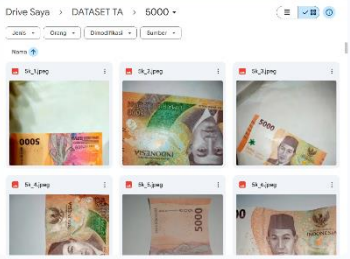
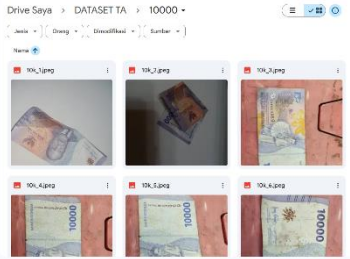
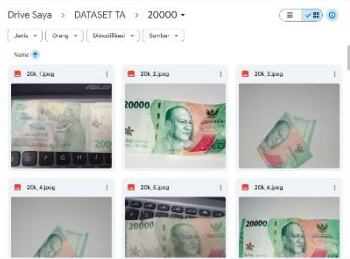
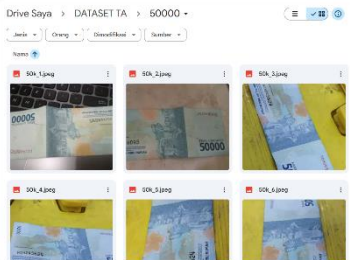
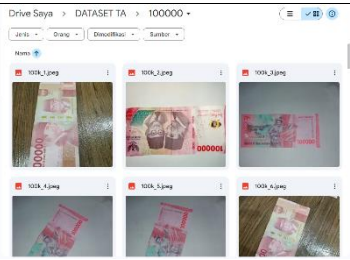
Dataset 1 : <https://www.kaggle.com/daniakhmadmaulana/citra-uang-rupiah-emisi-2022>

Dataset 2 : <https://www.kaggle.com/datasets/fannyzahrahmadhan/uang-emisi-2022-baru>



b. Contoh Citra Dataset

No	Nominal	Contoh Citra
(1)	(2)	(3)
1	Rp1.000	
2	Rp2.000	

3	Rp5.000	
4	Rp10.000	
5	Rp20.000	
6	Rp50.000	
7	Rp100.000	

Lampiran 2. Source Code CNN Custom

```
# Cell 1 - Mount Google Drive dan Instalasi Library

from google.colab import drive
drive.mount('/content/drive')
!pip install split-folders -q

# Cell 2 - Import Library

import os                                # untuk operasi path dan folder
import json                              # untuk menyimpan hasil
evaluasi ke Drive
```

```

import numpy as np                # untuk operasi array dan
matriks
import matplotlib.pyplot as plt  # untuk visualisasi grafik
training
import seaborn as sns            # untuk visualisasi confusion
matrix
import splitfolders              # untuk membagi dataset
train/val/test
import tensorflow as tf          # framework utama deep learning
from tensorflow.keras.preprocessing.image import
ImageDataGenerator # augmentasi dan preprocessing gambar
from tensorflow.keras.models import
Sequential                # untuk membangun arsitektur
CNN Custom
from tensorflow.keras.layers import Conv2D, MaxPooling2D,
Flatten, Dense, Dropout, Input # layer-layer CNN
from tensorflow.keras.callbacks import
EarlyStopping             # untuk menghentikan training
otomatis
from sklearn.metrics import classification_report,
confusion_matrix # untuk evaluasi metrik dan confusion matrix
import time              # untuk mengukur inference time

# Cell 3 - Split Dataset (70% Train, 15% Validasi, 15% Test)

dataset_path = '/content/drive/MyDrive/DATASET TA'
output_path = '/content/dataset_split'

splitfolders.ratio(
    dataset_path,
    output=output_path,
    seed=42,
    ratio=(0.7, 0.15, 0.15)
)

print("Split selesai.")
print("Isi folder hasil split:")
for folder in ['train', 'val', 'test']:
    total = 0
    for kelas in os.listdir(os.path.join(output_path, folder)):
        jumlah = len(os.listdir(os.path.join(output_path,
folder, kelas)))
        total += jumlah
    print(f" {folder}: {total} gambar")

# Cell 4 - Preprocessing dan Augmentasi Data

img_size = (224, 224)
batch_size = 32

train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    horizontal_flip=True,
    zoom_range=0.2,
    width_shift_range=0.2,
    height_shift_range=0.2
)

```

```

val_test_datagen = ImageDataGenerator(rescale=1./255)

train_data = train_datagen.flow_from_directory(
    os.path.join(output_path, 'train'),
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical'
)

val_data = val_test_datagen.flow_from_directory(
    os.path.join(output_path, 'val'),
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical'
)

test_data = val_test_datagen.flow_from_directory(
    os.path.join(output_path, 'test'),
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False
)

print("Label kelas:", train_data.class_indices)

# Cell 4b - Visualisasi Teknik Augmentasi Data

import random
import matplotlib.pyplot as plt
import numpy as np
import os
from tensorflow.keras.preprocessing.image import load_img,
img_to_array, ImageDataGenerator

# Ambil 1 gambar secara acak dari kelas tertentu
test_folder = os.path.join(output_path, 'test')
kelas_list = sorted(os.listdir(test_folder))
img_path = os.path.join(test_folder, kelas_list[5],
    random.choice(os.listdir(os.path.join(test_folder,
    kelas_list[5]))))
img_asli = load_img(img_path, target_size=(224, 224))
img_array = img_to_array(img_asli) / 255.0
img_expanded = np.expand_dims(img_array, axis=0)

augmentasi = {
    'Asli': ImageDataGenerator(),
    'Rotasi': ImageDataGenerator(rotation_range=20),
    'Flip Horizontal': ImageDataGenerator(horizontal_flip=True),
    'Zoom': ImageDataGenerator(zoom_range=0.2),
    'Geser Horizontal':
    ImageDataGenerator(width_shift_range=0.2),
    'Geser Vertikal': ImageDataGenerator(height_shift_range=0.2)
}

fig, axes = plt.subplots(1, 6, figsize=(20, 4))
fig.suptitle('Teknik Augmentasi Data', fontsize=14)

for idx, (nama, aug) in enumerate(augmentasi.items()):

```

```

        aug_iter = aug.flow(img_expanded, batch_size=1)
        aug_img = next(aug_iter)[0]
        axes[idx].imshow(aug_img)
        axes[idx].set_title(nama, fontsize=10)
        axes[idx].axis('off')

plt.tight_layout()
plt.savefig('/content/drive/MyDrive/visualisasi_teknik_augmentas
i.png', dpi=150, bbox_inches='tight')
plt.show()
print("Gambar visualisasi teknik augmentasi tersimpan ke
Drive.")

# Cell 5 - Arsitektur CNN Custom

model_cnn = Sequential([
    Input(shape=(224, 224, 3)),

    # Blok Konvolusi 1
    Conv2D(32, (3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),

    # Blok Konvolusi 2
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),

    # Blok Konvolusi 3
    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),

    # Flatten dan Fully Connected Layer
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),

    # Output Layer
    Dense(7, activation='softmax')
])

model_cnn.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)

model_cnn.summary()

# Cell 6 - Training CNN Custom

early_stopping = EarlyStopping(
    monitor='val_loss',
    patience=10,
    restore_best_weights=True
)

history_cnn = model_cnn.fit(
    train_data,
    epochs=50,
    validation_data=val_data,

```

```

        callbacks=[early_stopping]
    )

    model_cnn.save('/content/drive/MyDrive/model_cnn_run1.keras')
    print("Model CNN Custom Run 1 tersimpan ke Drive.")

    history_dict = {key: [float(v) for v in val] for key, val in
                    history_cnn.history.items()}
    with open('/content/drive/MyDrive/history_cnn_run1.json', 'w')
    as f:
        json.dump(history_dict, f)
    print("History CNN Custom Run 1 tersimpan ke Drive.")

    # Load model CNN Custom Run 1 dari Drive (jalankan jika sesi
    reset)

    from tensorflow.keras.models import load_model

    model_cnn =
    load_model('/content/drive/MyDrive/model_cnn_run1.keras')
    print("Model CNN Custom Run 1 berhasil di-load.")
    # Load history CNN Custom Run 1 dari Drive (jalankan jika sesi
    reset)

    with open('/content/drive/MyDrive/history_cnn_run1.json', 'r')
    as f:
        history_dict = json.load(f)

    class HistoryObj:
        def __init__(self, d):
            self.history = d

    history_cnn = HistoryObj(history_dict)

    print("History CNN Custom Run 1 berhasil di-load.")

    # Cell 7 - Evaluasi CNN Custom Run 1

    test_loss_cnn, test_acc_cnn = model_cnn.evaluate(test_data,
    verbose=0)
    print(f"Test Accuracy CNN Custom : {test_acc_cnn:.4f}")
    print(f"Test Loss CNN Custom      : {test_loss_cnn:.4f}")

    y_pred_cnn = model_cnn.predict(test_data)
    y_pred_cnn_classes = np.argmax(y_pred_cnn, axis=1)
    y_true = test_data.classes

    class_names = list(test_data.class_indices.keys())
    print("\nClassification Report CNN Custom Run 1:")
    print(classification_report(y_true, y_pred_cnn_classes,
    target_names=class_names))

    sample_batch = next(iter(test_data))[0][:1]
    start = time.time()
    for _ in range(100):
        model_cnn.predict(sample_batch, verbose=0)
    end = time.time()
    inference_time_cnn = (end - start) / 100 * 1000

```

```

print(f"Inference Time CNN Custom: {inference_time_cnn:.2f} ms
per gambar")

model_size_cnn =
os.path.getsize('/content/drive/MyDrive/model_cnn_run1.keras') /
(1024 * 1024)
print(f"Ukuran Model CNN Custom : {model_size_cnn:.2f} MB")

# Simpan hasil evaluasi lengkap ke Drive
from sklearn.metrics import precision_score, recall_score,
f1_score
hasil_run1 = {
    'accuracy': float(test_acc_cnn),
    'loss': float(test_loss_cnn),
    'precision': float(precision_score(y_true,
y_pred_cnn_classes, average='macro')),
    'recall': float(recall_score(y_true, y_pred_cnn_classes,
average='macro')),
    'f1_score': float(f1_score(y_true, y_pred_cnn_classes,
average='macro')),
    'inference_time_ms': float(inference_time_cnn),
    'model_size_mb': float(model_size_cnn)
}
with open('/content/drive/MyDrive/hasil_cnn_run1.json', 'w') as
f:
    json.dump(hasil_run1, f)
print("\nHasil evaluasi CNN Custom Run 1 tersimpan ke Drive.")

# Cell 8 - Tabel Evaluasi Per Kelas CNN Custom Run 1
# Accuracy dihitung sesuai rumus Bab II

from sklearn.metrics import classification_report,
multilabel_confusion_matrix
import pandas as pd

# Classification report dalam bentuk dictionary
report = classification_report(
    y_true,
    y_pred_cnn_classes,
    target_names=class_names,
    output_dict=True,
    zero_division=0
)

# Multilabel Confusion Matrix
mcm = multilabel_confusion_matrix(y_true, y_pred_cnn_classes)

hasil = []

for i, class_name in enumerate(class_names):

    tn, fp, fn, tp = mcm[i].ravel()

    # Accuracy sesuai rumus Bab II
    accuracy = (tp + tn) / (tp + tn + fp + fn)

    hasil.append({
        "Nominal": class_name,
        "Accuracy (%)": round(accuracy * 100, 2),

```

```

        "Precision (%)": round(report[class_name]["precision"] *
100, 2),
        "Recall (%)": round(report[class_name]["recall"] * 100,
2),
        "F1-Score (%)": round(report[class_name]["f1-score"] *
100, 2),
        "Support": int(report[class_name]["support"]))
    })

df_hasil = pd.DataFrame(hasil)

print("Tabel Evaluasi Per Kelas CNN Custom Run 1")
display(df_hasil)

# Cell 8b - Visualisasi Grafik Training CNN Custom Run 1

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 5))

# Grafik Accuracy
ax1.plot(history_cnn.history['accuracy'], label='Train
Accuracy')
ax1.plot(history_cnn.history['val_accuracy'], label='Val
Accuracy')
ax1.set_title('CNN Custom Run 1 - Accuracy')
ax1.set_xlabel('Epoch')
ax1.set_ylabel('Accuracy')
ax1.legend()
ax1.grid(True)

# Grafik Loss
ax2.plot(history_cnn.history['loss'], label='Train Loss')
ax2.plot(history_cnn.history['val_loss'], label='Val Loss')
ax2.set_title('CNN Custom Run 1 - Loss')
ax2.set_xlabel('Epoch')
ax2.set_ylabel('Loss')
ax2.legend()
ax2.grid(True)

plt.tight_layout()
plt.savefig('/content/drive/MyDrive/grafik_cnn_run1.png',
dpi=150, bbox_inches='tight')
plt.show()

# Cell 9 - Confusion Matrix CNN Custom Run 1

cm_cnn = confusion_matrix(y_true, y_pred_cnn_classes)
class_names = list(test_data.class_indices.keys())

plt.figure(figsize=(10, 8))
sns.heatmap(cm_cnn, annot=True, fmt='d', cmap='Blues',
            xticklabels=class_names,
            yticklabels=class_names)
plt.title('Confusion Matrix - CNN Custom Run 1')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.tight_layout()
plt.savefig('/content/drive/MyDrive/cm_cnn_run1.png', dpi=150,
bbox_inches='tight')
plt.show()

```

```

# Cell - TP, TN, FP, FN per Kelas CNN Custom Run 1
# Menghitung nilai TP, TN, FP, dan FN untuk setiap kelas
berdasarkan confusion matrix

from sklearn.metrics import multilabel_confusion_matrix
import pandas as pd

mcm = multilabel_confusion_matrix(y_true, y_pred_cnn_classes)
class_names = list(test_data.class_indices.keys())

hasil_tp = []
print("TP, TN, FP, FN per Kelas - CNN Custom Run 1")
print("=" * 60)
for i, class_name in enumerate(class_names):
    tn, fp, fn, tp = mcm[i].ravel()
    hasil_tp.append({
        "Kelas": class_name,
        "TP": tp,
        "FP": fp,
        "FN": fn,
        "TN": tn
    })

df_tp = pd.DataFrame(hasil_tp)
df_tp = df_tp[["Kelas", "TP", "FP", "FN", "TN"]]
df_tp.index = df_tp.index + 1
display(df_tp)

# Cell - TP, TN, FP, FN per Kelas CNN Custom Run 1
# Menghitung nilai TP, TN, FP, dan FN untuk setiap kelas
berdasarkan confusion matrix

from sklearn.metrics import multilabel_confusion_matrix
import pandas as pd

mcm = multilabel_confusion_matrix(y_true, y_pred_cnn_classes)
class_names = list(test_data.class_indices.keys())

hasil_tp = []
print("TP, TN, FP, FN per Kelas - CNN Custom Run 1")
print("=" * 60)
for i, class_name in enumerate(class_names):
    tn, fp, fn, tp = mcm[i].ravel()
    hasil_tp.append({
        "Kelas": class_name,
        "TP": tp,
        "FP": fp,
        "FN": fn,
        "TN": tn
    })

df_tp = pd.DataFrame(hasil_tp)
df_tp = df_tp[["Kelas", "TP", "FP", "FN", "TN"]]
df_tp.index = df_tp.index + 1
display(df_tp)

```

Lampiran 3. Source Code MobileNetV2

```
# Cell 1 - Mount Google Drive dan Instalasi Library

from google.colab import drive
drive.mount('/content/drive')

!pip install split-folders -q

# Cell 2 - Import Library

import os                    # operasi path dan folder
import json                  # simpan hasil ke file JSON
import numpy as np           # operasi array
import matplotlib.pyplot as plt # visualisasi grafik
import seaborn as sns        # visualisasi confusion matrix
import splitfolders          # split dataset
import tensorflow as tf      # framework deep learning
from tensorflow.keras.preprocessing.image import
ImageDataGenerator # augmentasi dan preprocessing
from tensorflow.keras.models import
Model              # Functional API untuk
MobileNetV2
from tensorflow.keras.layers import Dense, Dropout,
GlobalAveragePooling2D, Input # layer klasifikasi baru
from tensorflow.keras.applications import
MobileNetV2             # pretrained MobileNetV2
from tensorflow.keras.callbacks import
EarlyStopping           # early stopping
from sklearn.metrics import classification_report,
confusion_matrix, precision_score, recall_score, f1_score #
metrik evaluasi
import time              # ukur inference time

# Cell 3 - Split Dataset (70% Train, 15% Validasi, 15% Test)

dataset_path = '/content/drive/MyDrive/DATASET TA'
output_path = '/content/dataset_split'

splitfolders.ratio(
    dataset_path,
    output=output_path,
    seed=42,
    ratio=(0.7, 0.15, 0.15)
)

print("Split selesai.")
print("Isi folder hasil split:")
for folder in ['train', 'val', 'test']:
```

```

        total = 0
        for kelas in os.listdir(os.path.join(output_path, folder)):
            jumlah = len(os.listdir(os.path.join(output_path,
folder, kelas)))
            total += jumlah
print(f"  {folder}: {total} gambar")

# Cell 4 - Preprocessing dan Augmentasi Data

img_size = (224, 224)
batch_size = 32

# Augmentasi hanya untuk data train
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    horizontal_flip=True,
    zoom_range=0.2,
    width_shift_range=0.2,
    height_shift_range=0.2
)

# Val dan test hanya dinormalisasi
val_test_datagen = ImageDataGenerator(rescale=1./255)

train_data = train_datagen.flow_from_directory(
    os.path.join(output_path, 'train'),
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical'
)

val_data = val_test_datagen.flow_from_directory(
    os.path.join(output_path, 'val'),
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical'
)

test_data = val_test_datagen.flow_from_directory(
    os.path.join(output_path, 'test'),
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False
)

print("Label kelas:", train_data.class_indices)

# Cell 4b - Visualisasi Teknik Augmentasi Data

import random
import matplotlib.pyplot as plt
import numpy as np
import os
from tensorflow.keras.preprocessing.image import load_img,
img_to_array, ImageDataGenerator

```

```

# Ambil 1 gambar secara acak dari kelas tertentu
test_folder = os.path.join(output_path, 'test')
kelas_list = sorted(os.listdir(test_folder))
img_path = os.path.join(test_folder, kelas_list[2],
random.choice(os.listdir(os.path.join(test_folder,
kelas_list[2]))))
img_asli = load_img(img_path, target_size=(224, 224))
img_array = img_to_array(img_asli) / 255.0
img_expanded = np.expand_dims(img_array, axis=0)

augmentasi = {
    'Asli': ImageDataGenerator(),
    'Rotasi': ImageDataGenerator(rotation_range=20),
    'Flip Horizontal': ImageDataGenerator(horizontal_flip=True),
    'Zoom': ImageDataGenerator(zoom_range=0.2),
    'Geser Horizontal':
ImageDataGenerator(width_shift_range=0.2),
    'Geser Vertikal': ImageDataGenerator(height_shift_range=0.2)
}

fig, axes = plt.subplots(1, 6, figsize=(20, 4))
fig.suptitle('Teknik Augmentasi Data', fontsize=14)

for idx, (nama, aug) in enumerate(augmentasi.items()):
    aug_iter = aug.flow(img_expanded, batch_size=1)
    aug_img = next(aug_iter)[0]
    axes[idx].imshow(aug_img)
    axes[idx].set_title(nama, fontsize=10)
    axes[idx].axis('off')

plt.tight_layout()
plt.savefig('/content/drive/MyDrive/visualisasi_teknik_augmentas
i.png', dpi=150, bbox_inches='tight')
plt.show()
print("Gambar visualisasi teknik augmentasi tersimpan ke
Drive.")

# Cell 5 - Load Pre-trained MobileNetV2

# Memuat MobileNetV2 dengan bobot ImageNet, tanpa lapisan
klasifikasi bawaan
base_model = MobileNetV2(
    input_shape=(224, 224, 3),
    include_top=False,
    weights='imagenet'
)

# Bekukan semua layer base model untuk training tahap 1
base_model.trainable = False

print(f"Jumlah layer base model: {len(base_model.layers)}")
print(f"Trainable: {base_model.trainable}")

# Cell 6 - Bangun Arsitektur MobileNetV2

inputs = Input(shape=(224, 224, 3)) # titik masuk data, gambar
224x224 RGB
x = base_model(inputs, training=False) # alirkan input ke base
model, training=False agar BatchNorm pakai statistik ImageNet

```

```

x = GlobalAveragePooling2D()(x) # rangkum feature map 7x7x1280
jadi vektor 1280
x = Dense(128, activation='relu')(x) # fully connected layer
untuk klasifikasi
x = Dropout(0.5)(x) # nonaktifkan 50% neuron untuk cegah
overfitting
outputs = Dense(7, activation='softmax')(x) # output 7 kelas,
softmax mengubah ke probabilitas

model_mobilenet = Model(inputs, outputs) # gabungkan semua
layer jadi satu model

model_mobilenet.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001), #
    LR lebih kecil dari CNN Custom agar tidak merusak bobot
    pretrained
    loss='categorical_crossentropy', # loss standar untuk
    klasifikasi multi-kelas
    metrics=['accuracy']
)

model_mobilenet.summary()

# Cell 7 - Training MobileNetV2 Tahap 1 (Base Model Freeze)

early_stopping = EarlyStopping(
    monitor='val_loss',
    patience=10,
    restore_best_weights=True
)

history_mobilenet_1 = model_mobilenet.fit(
    train_data,
    epochs=50,
    validation_data=val_data,
    callbacks=[early_stopping]
)

# Simpan history tahap 1 ke Drive
history_dict_1 = {key: [float(v) for v in val] for key, val in
history_mobilenet_1.history.items()}
with open('/content/drive/MyDrive/history_mobilenet1_run4.json',
'w') as f:
    json.dump(history_dict_1, f)
print("History MobileNetV2 tahap 1 Run 4 tersimpan ke Drive.")

# Cell 8 - Fine-Tuning MobileNetV2 Tahap 2

# Buka layer 100-153 base model untuk dilatih ulang
base_model.trainable = True
for layer in base_model.layers[:100]:
    layer.trainable = False # layer awal tetap dibekukan

# Compile ulang dengan learning rate lebih kecil
model_mobilenet.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=0.00001),
    # LR sangat kecil agar tidak merusak fitur pretrained
    loss='categorical_crossentropy',

```

```

        metrics=['accuracy']
    )

    early_stopping = EarlyStopping(
        monitor='val_loss',
        patience=10,
        restore_best_weights=True
    )

    history_mobilenet_2 = model_mobilenet.fit(
        train_data,
        epochs=50,
        validation_data=val_data,
        callbacks=[early_stopping]
    )

    # Simpan model dan history tahap 2 ke Drive
    model_mobilenet.save('/content/drive/MyDrive/model_mobilenet_run
4.keras')
    print("Model MobileNetV2 Run 4 tersimpan ke Drive.")

    history_dict_2 = {key: [float(v) for v in val] for key, val in
history_mobilenet_2.history.items()}
    with open('/content/drive/MyDrive/history_mobilenet2_run4.json',
'w') as f:
        json.dump(history_dict_2, f)
    print("History MobileNetV2 tahap 2 Run 4 tersimpan ke Drive.")

    # Load model MobileNetV2 Run 4 dari Drive (jalankan jika sesi
reset)

    from tensorflow.keras.models import load_model

    model_mobilenet =
load_model('/content/drive/MyDrive/model_mobilenet_run4.keras')
    print("Model MobileNetV2 Run 4 berhasil di-load.")

    # Load history MobileNetV2 Run 4 dari Drive (jalankan jika sesi
reset)

    with open('/content/drive/MyDrive/history_mobilenet1_run4.json',
'r') as f:
        h1 = json.load(f)

    with open('/content/drive/MyDrive/history_mobilenet2_run4.json',
'r') as f:
        h2 = json.load(f)

    class HistoryObj:
        def __init__(self, d):
            self.history = d

    history_mobilenet_1 = HistoryObj(h1)
    history_mobilenet_2 = HistoryObj(h2)

    print("History MobileNetV2 Run 4 berhasil di-load.")

    # Cell 9 - Evaluasi MobileNetV2 Run 4

```

```

test_loss_mobilenet, test_acc_mobilenet =
model_mobilenet.evaluate(test_data, verbose=0)
print(f"Test Accuracy MobileNetV2 : {test_acc_mobilenet:.4f}")
print(f"Test Loss MobileNetV2      : {test_loss_mobilenet:.4f}")

y_pred_mobilenet = model_mobilenet.predict(test_data)
y_pred_mobilenet_classes = np.argmax(y_pred_mobilenet, axis=1)
y_true = test_data.classes

class_names = list(test_data.class_indices.keys())
print("\nClassification Report MobileNetV2 Run 4:")
print(classification_report(y_true, y_pred_mobilenet_classes,
target_names=class_names))

# Inference time diukur 100x untuk hasil yang lebih stabil
sample_batch = next(iter(test_data))[0][:1]
start = time.time()
for _ in range(100):
    model_mobilenet.predict(sample_batch, verbose=0)
end = time.time()
inference_time_mobilenet = (end - start) / 100 * 1000
print(f"Inference Time MobileNetV2 :
{inference_time_mobilenet:.2f} ms per gambar")

model_size_mobilenet =
os.path.getsize('/content/drive/MyDrive/model_mobilenet_run4.ker
as') / (1024 * 1024)
print(f"Ukuran Model MobileNetV2      : {model_size_mobilenet:.2f}
MB")

# Simpan hasil evaluasi ke Drive
hasil_run4 = {
    'accuracy': float(test_acc_mobilenet),
    'loss': float(test_loss_mobilenet),
    'precision': float(precision_score(y_true,
y_pred_mobilenet_classes, average='macro')),
    'recall': float(recall_score(y_true,
y_pred_mobilenet_classes, average='macro')),
    'f1_score': float(f1_score(y_true, y_pred_mobilenet_classes,
average='macro')),
    'inference_time_ms': float(inference_time_mobilenet),
    'model_size_mb': float(model_size_mobilenet)
}
with open('/content/drive/MyDrive/hasil_mobilenet_run4.json',
'w') as f:
    json.dump(hasil_run4, f)
print("\nHasil evaluasi MobileNetV2 Run 4 tersimpan ke Drive.")

# Cell - Tabel Evaluasi Per Kelas MobileNetV2 Run 4
# Accuracy dihitung sesuai rumus Bab II

from sklearn.metrics import classification_report,
multilabel_confusion_matrix
import pandas as pd

# Classification Report dalam bentuk dictionary
report = classification_report(
    y_true,
    y_pred_mobilenet_classes,

```

```

        target_names=class_names,
        output_dict=True,
        zero_division=0
    )

    # Multilabel Confusion Matrix
    mcm = multilabel_confusion_matrix(y_true,
    y_pred_mobilenet_classes)

    hasil = []

    for i, class_name in enumerate(class_names):

        tn, fp, fn, tp = mcm[i].ravel()

        accuracy = (tp + tn) / (tp + tn + fp + fn)

        hasil.append({
            "Nominal": class_name,
            "Accuracy (%)": round(accuracy * 100, 2),
            "Precision (%)": round(report[class_name]["precision"] *
100, 2),
            "Recall (%)": round(report[class_name]["recall"] * 100,
2),
            "F1-Score (%)": round(report[class_name]["f1-score"] *
100, 2),
            "Support": int(report[class_name]["support"])
        })

    df_hasil = pd.DataFrame(hasil)

    print("Tabel Evaluasi Per Kelas MobileNetV2 Run 4")
    display(df_hasil)

# Cell 10 - Visualisasi Grafik Training MobileNetV2 Run 4

# Gabungkan history tahap 1 dan tahap 2
history_combined = {
    'accuracy': history_mobilenet_1.history['accuracy'] +
history_mobilenet_2.history['accuracy'],
    'val_accuracy': history_mobilenet_1.history['val_accuracy']
+ history_mobilenet_2.history['val_accuracy'],
    'loss': history_mobilenet_1.history['loss'] +
history_mobilenet_2.history['loss'],
    'val_loss': history_mobilenet_1.history['val_loss'] +
history_mobilenet_2.history['val_loss'],
}

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 5))

# Grafik Accuracy
ax1.plot(history_combined['accuracy'], label='Train Accuracy')
ax1.plot(history_combined['val_accuracy'], label='Val Accuracy')
ax1.axvline(x=50, color='red', linestyle='--', label='Fine-
tuning dimulai')
ax1.set_title('MobileNetV2 Run 4 - Accuracy')
ax1.set_xlabel('Epoch')
ax1.set_ylabel('Accuracy')

```

```

ax1.legend()
ax1.grid(True)

# Grafik Loss
ax2.plot(history_combined['loss'], label='Train Loss')
ax2.plot(history_combined['val_loss'], label='Val Loss')
ax2.axvline(x=50, color='red', linestyle='--', label='Fine-
tuning dimulai')
ax2.set_title('MobileNetV2 Run 4 - Loss')
ax2.set_xlabel('Epoch')
ax2.set_ylabel('Loss')
ax2.legend()
ax2.grid(True)

plt.tight_layout()
plt.savefig('/content/drive/MyDrive/grafik_mobilenet_run4.png',
dpi=150, bbox_inches='tight')
plt.show()

# Cell 11 - Confusion Matrix MobileNetV2 Run 4

cm_mobilenet = confusion_matrix(y_true,
y_pred_mobilenet_classes)
class_names = list(test_data.class_indices.keys())

plt.figure(figsize=(10, 8))
sns.heatmap(cm_mobilenet, annot=True, fmt='d', cmap='Blues',
            xticklabels=class_names,
            yticklabels=class_names)
plt.title('Confusion Matrix - MobileNetV2 Run 4')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.tight_layout()
plt.savefig('/content/drive/MyDrive/cm_mobilenet_run4.png',
dpi=150, bbox_inches='tight')
plt.show()
# Cell - TP, TN, FP, FN per Kelas MobilenetV2 Run 4
# Menghitung nilai TP, TN, FP, dan FN untuk setiap kelas
berdasarkan confusion matrix

from sklearn.metrics import multilabel_confusion_matrix
import pandas as pd

mcm = multilabel_confusion_matrix(y_true,
y_pred_mobilenet_classes)
class_names = list(test_data.class_indices.keys())

hasil_tp = []
print("TP, TN, FP, FN per Kelas - MobileNetV2 Run 4")
print("=" * 60)
for i, class_name in enumerate(class_names):
    tn, fp, fn, tp = mcm[i].ravel()
    hasil_tp.append({
        "Kelas": class_name,
        "TP": tp,
        "FP": fp,
        "FN": fn,
        "TN": tn
    })

```

```
df_tp = pd.DataFrame(hasil_tp)
df_tp = df_tp[["Kelas", "TP", "FP", "FN", "TN"]]
df_tp.index = df_tp.index + 1
display(df_tp)
```

Lampiran 4. Source Code Analisis Hasil Penelitian

```
# Cell 1 - Mount Google Drive dan Import Library
# Mengimpor library yang diperlukan dan menghubungkan ke Google
Drive
from google.colab import drive
drive.mount('/content/drive')

import json
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Cell 2 - Load Hasil Evaluasi Semua Run dari Drive
# Memuat file hasil evaluasi dari setiap percobaan yang tersimpan
di Google Drive
# Load hasil CNN Custom (5 run)
with open('/content/drive/MyDrive/hasil_cnn_run1.json', 'r') as
f:
    cnn_run1 = json.load(f)
with open('/content/drive/MyDrive/hasil_cnn_run2.json', 'r') as
f:
    cnn_run2 = json.load(f)
with open('/content/drive/MyDrive/hasil_cnn_run3.json', 'r') as
f:
    cnn_run3 = json.load(f)
with open('/content/drive/MyDrive/hasil_cnn_run4.json', 'r') as
f:
    cnn_run4 = json.load(f)
with open('/content/drive/MyDrive/hasil_cnn_run5.json', 'r') as
f:
    cnn_run5 = json.load(f)

# Load hasil MobileNetV2 (5 run)
with open('/content/drive/MyDrive/hasil_mobilenet_run1.json',
'r') as f:
    mobilenet_run1 = json.load(f)
with open('/content/drive/MyDrive/hasil_mobilenet_run2.json',
'r') as f:
    mobilenet_run2 = json.load(f)
with open('/content/drive/MyDrive/hasil_mobilenet_run3.json',
'r') as f:
    mobilenet_run3 = json.load(f)
with open('/content/drive/MyDrive/hasil_mobilenet_run4.json',
'r') as f:
    mobilenet_run4 = json.load(f)
with open('/content/drive/MyDrive/hasil_mobilenet_run5.json',
'r') as f:
    mobilenet_run5 = json.load(f)
```

```

print('Semua data berhasil di-load.')

# Cell 3 - Tabel Hasil Per Run
# Menampilkan hasil evaluasi seluruh percobaan dalam bentuk tabel
cnn_runs = [cnn_run1, cnn_run2, cnn_run3, cnn_run4, cnn_run5]
mobilenet_runs = [mobilenet_run1, mobilenet_run2, mobilenet_run3,
mobilenet_run4, mobilenet_run5]

data_per_run = []
for i, run in enumerate(cnn_runs, 1):
    data_per_run.append({
        'Model': 'CNN Custom',
        'Run': f'Run {i}',
        'Accuracy (%)': round(run['accuracy'] * 100, 2),
        'Precision (%)': round(run['precision'] * 100, 2),
        'Recall (%)': round(run['recall'] * 100, 2),
        'F1-Score (%)': round(run['f1_score'] * 100, 2),
        'Inference Time (ms)': round(run['inference_time_ms'],
2),
        'Ukuran Model (MB)': round(run['model_size_mb'], 2)
    })

for i, run in enumerate(mobilenet_runs, 1):
    data_per_run.append({
        'Model': 'MobileNetV2',
        'Run': f'Run {i}',
        'Accuracy (%)': round(run['accuracy'] * 100, 2),
        'Precision (%)': round(run['precision'] * 100, 2),
        'Recall (%)': round(run['recall'] * 100, 2),
        'F1-Score (%)': round(run['f1_score'] * 100, 2),
        'Inference Time (ms)': round(run['inference_time_ms'],
2),
        'Ukuran Model (MB)': round(run['model_size_mb'], 2)
    })

df_per_run = pd.DataFrame(data_per_run)
df_per_run.index = df_per_run.index + 1

print('Tabel Hasil Per Run')
print('=' * 80)
display(df_per_run)

# Cell 4 - Hitung Mean dan Standar Deviasi
# Menghitung nilai rata-rata dan standar deviasi dari seluruh
percobaan

metrics = ['accuracy', 'precision', 'recall', 'f1_score',
'inference_time_ms', 'model_size_mb']

cnn_mean = {m: np.mean([r[m] for r in cnn_runs]) for m in metrics}
cnn_std = {m: np.std([r[m] for r in cnn_runs], ddof=1) for m in
metrics}

mobilenet_mean = {m: np.mean([r[m] for r in mobilenet_runs]) for
m in metrics}
mobilenet_std = {m: np.std([r[m] for r in mobilenet_runs],
ddof=1) for m in metrics}

```

```

rekap = {
    'Metrik': ['Accuracy (%)', 'Precision (%)', 'Recall (%)', 'F1-
Score (%)', 'Inference Time (ms)', 'Ukuran Model (MB)'],
    'CNN Mean': [
        round(cnn_mean['accuracy']*100, 2),
        round(cnn_mean['precision']*100, 2),
        round(cnn_mean['recall']*100, 2),
        round(cnn_mean['f1_score']*100, 2),
        round(cnn_mean['inference_time_ms'], 2),
        round(cnn_mean['model_size_mb'], 2)
    ],
    'CNN SD': [
        round(cnn_std['accuracy']*100, 2),
        round(cnn_std['precision']*100, 2),
        round(cnn_std['recall']*100, 2),
        round(cnn_std['f1_score']*100, 2),
        round(cnn_std['inference_time_ms'], 2),
        round(cnn_std['model_size_mb'], 2)
    ],
    'MobileNetV2 Mean': [
        round(mobilenet_mean['accuracy']*100, 2),
        round(mobilenet_mean['precision']*100, 2),
        round(mobilenet_mean['recall']*100, 2),
        round(mobilenet_mean['f1_score']*100, 2),
        round(mobilenet_mean['inference_time_ms'], 2),
        round(mobilenet_mean['model_size_mb'], 2)
    ],
    'MobileNetV2 SD': [
        round(mobilenet_std['accuracy']*100, 2),
        round(mobilenet_std['precision']*100, 2),
        round(mobilenet_std['recall']*100, 2),
        round(mobilenet_std['f1_score']*100, 2),
        round(mobilenet_std['inference_time_ms'], 2),
        round(mobilenet_std['model_size_mb'], 2)
    ]
}

df_rekap = pd.DataFrame(rekap)
df_rekap.index = df_rekap.index + 1

print('Tabel Rekapitulasi Mean dan Standar Deviasi')
print('=' * 80)
display(df_rekap)
# Cell 5 - Grafik Perbandingan Accuracy dan F1-Score per Percobaan
# Membandingkan nilai Accuracy dan F1-Score CNN Custom dan
MobileNetV2 pada setiap percobaan

runs = ['Run 1', 'Run 2', 'Run 3', 'Run 4', 'Run 5']
cnn_acc = [round(r['accuracy']*100, 2) for r in cnn_runs]
mobilenet_acc = [round(r['accuracy']*100, 2) for r in
mobilenet_runs]
cnn_f1 = [round(r['f1_score']*100, 2) for r in cnn_runs]
mobilenet_f1 = [round(r['f1_score']*100, 2) for r in
mobilenet_runs]

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 5))
x = range(len(runs))
width = 0.35

```

```

ax1.bar([i - width/2 for i in x], cnn_acc, width, label='CNN
Custom', color='steelblue')
ax1.bar([i + width/2 for i in x], mobilenet_acc, width,
label='MobileNetV2', color='darkorange')
ax1.set_xlabel('Percobaan')
ax1.set_ylabel('Accuracy (%)')
ax1.set_title('Perbandingan Accuracy per Percobaan')
ax1.set_xticks(list(x))
ax1.set_xticklabels(runs)
ax1.set_ylim(min(cnn_acc + mobilenet_acc) - 5, 100)
ax1.legend()
ax1.grid(axis='y', alpha=0.3)

ax2.bar([i - width/2 for i in x], cnn_f1, width, label='CNN
Custom', color='steelblue')
ax2.bar([i + width/2 for i in x], mobilenet_f1, width,
label='MobileNetV2', color='darkorange')
ax2.set_xlabel('Percobaan')
ax2.set_ylabel('F1-Score (%)')
ax2.set_title('Perbandingan F1-Score per Percobaan')
ax2.set_xticks(list(x))
ax2.set_xticklabels(runs)
ax2.set_ylim(min(cnn_f1 + mobilenet_f1) - 5, 100)
ax2.legend()
ax2.grid(axis='y', alpha=0.3)

plt.tight_layout()
plt.savefig('/content/drive/MyDrive/grafik_perbandingan_per_run.
png', dpi=150, bbox_inches='tight')
plt.show()
print('Grafik tersimpan ke Drive.')

# Cell 6a - Grafik Perbandingan Rata-rata Metrik Klasifikasi
# Membandingkan rata-rata Accuracy, Precision, Recall, dan F1-
Score kedua model

klasifikasi = ['Accuracy', 'Precision', 'Recall', 'F1-Score']
cnn_scores = [
    round(cnn_mean['accuracy']*100, 2),
    round(cnn_mean['precision']*100, 2),
    round(cnn_mean['recall']*100, 2),
    round(cnn_mean['f1_score']*100, 2)
]
mobilenet_scores = [
    round(mobilenet_mean['accuracy']*100, 2),
    round(mobilenet_mean['precision']*100, 2),
    round(mobilenet_mean['recall']*100, 2),
    round(mobilenet_mean['f1_score']*100, 2)
]

x = range(len(klasifikasi))
width = 0.35

fig, ax = plt.subplots(figsize=(10, 5))
ax.bar([i - width/2 for i in x], cnn_scores, width, label='CNN
Custom', color='steelblue')
ax.bar([i + width/2 for i in x], mobilenet_scores, width,
label='MobileNetV2', color='darkorange')
ax.set_ylabel('Nilai (%)')

```

```

ax.set_title('Perbandingan Rata-rata Metrik Klasifikasi')
ax.set_xticks(list(x))
ax.set_xticklabels(klasifikasi)
ax.set_ylim(min(cnn_scores + mobilenet_scores) - 5, 100)
ax.legend()
ax.grid(axis='y', alpha=0.3)

plt.tight_layout()
plt.savefig('/content/drive/MyDrive/grafik_metrik_ratarata.png',
            dpi=150, bbox_inches='tight')
plt.show()
print('Grafik Metrik Klasifikasi tersimpan ke Drive.')
# Cell 6b - Grafik Perbandingan Rata-rata Efisiensi Model
# Membandingkan rata-rata Inference Time dan Ukuran Model kedua
model

efisiensi = ['Inference Time (ms)', 'Ukuran Model (MB)']
cnn_eff = [
    round(cnn_mean['inference_time_ms'], 2),
    round(cnn_mean['model_size_mb'], 2)
]
mobilenet_eff = [
    round(mobilenet_mean['inference_time_ms'], 2),
    round(mobilenet_mean['model_size_mb'], 2)
]

x2 = range(len(efisiensi))
width = 0.35

fig, ax = plt.subplots(figsize=(10, 5))
ax.bar([i - width/2 for i in x2], cnn_eff, width, label='CNN
Custom', color='steelblue')
ax.bar([i + width/2 for i in x2], mobilenet_eff, width,
label='MobileNetV2', color='darkorange')
ax.set_ylabel('Nilai')
ax.set_title('Perbandingan Rata-rata Efisiensi Model')
ax.set_xticks(list(x2))
ax.set_xticklabels(efisiensi)
ax.legend()
ax.grid(axis='y', alpha=0.3)

plt.tight_layout()
plt.savefig('/content/drive/MyDrive/grafik_efisiensi_ratarata.png',
            dpi=150, bbox_inches='tight')
plt.show()
print('Grafik Efisiensi Model tersimpan ke Drive.')
# Cell 7 - Pemenang Tiap Metrik dan Selisih
# Menentukan model dengan performa terbaik pada setiap metrik
evaluasi
print('PEMENANG TIAP METRIK')
print('=' * 60)

metrik_label = {
    'accuracy': 'Accuracy',
    'precision': 'Precision',
    'recall': 'Recall',
    'f1_score': 'F1-Score'
}

```

```

for key, label in metrik_label.items():
    cnn_val = round(cnn_mean[key]*100, 2)
    mob_val = round(mobilenet_mean[key]*100, 2)
    selisih = round(abs(cnn_val - mob_val), 2)
    pemenang = 'MobileNetV2' if mob_val >= cnn_val else 'CNN Custom'
    print(f'{label:15} → {pemenang} (selisih {selisih}%)')

# Inference time (lebih kecil lebih baik)
cnn_inf = round(cnn_mean['inference_time_ms'], 2)
mob_inf = round(mobilenet_mean['inference_time_ms'], 2)
selisih_inf = round(abs(cnn_inf - mob_inf), 2)
pemenang_inf = 'CNN Custom' if cnn_inf <= mob_inf else 'MobileNetV2'
print(f'{"Inference Time":15} → {pemenang_inf} (selisih {selisih_inf} ms)')

# Ukuran model (lebih kecil lebih baik)
cnn_size = round(cnn_mean['model_size_mb'], 2)
mob_size = round(mobilenet_mean['model_size_mb'], 2)
selisih_size = round(abs(cnn_size - mob_size), 2)
pemenang_size = 'CNN Custom' if cnn_size <= mob_size else 'MobileNetV2'
print(f'{"Ukuran Model":15} → {pemenang_size} (selisih {selisih_size} MB)')

# Cell 8 - Kesimpulan Analisis
# Merangkum hasil analisis perbandingan kedua model secara keseluruhan
print('KESIMPULAN ANALISIS')
print('=' * 60)
print(f'CNN Custom - Accuracy rata-rata : {round(cnn_mean["accuracy"]*100,2)}% (SD: {round(cnn_std["accuracy"]*100,2)}%)')
print(f'MobileNetV2 - Accuracy rata-rata : {round(mobilenet_mean["accuracy"]*100,2)}% (SD: {round(mobilenet_std["accuracy"]*100,2)}%)')
print()
print(f'CNN Custom - F1-Score rata-rata : {round(cnn_mean["f1_score"]*100,2)}% (SD: {round(cnn_std["f1_score"]*100,2)}%)')
print(f'MobileNetV2 - F1-Score rata-rata : {round(mobilenet_mean["f1_score"]*100,2)}% (SD: {round(mobilenet_std["f1_score"]*100,2)}%)')
print()
print(f'CNN Custom - Inference Time rata-rata : {round(cnn_mean["inference_time_ms"],2)} ms (SD: {round(cnn_std["inference_time_ms"],2)} ms)')
print(f'MobileNetV2 - Inference Time rata-rata : {round(mobilenet_mean["inference_time_ms"],2)} ms (SD: {round(mobilenet_std["inference_time_ms"],2)} ms)')
print()
print(f'CNN Custom - Ukuran Model : {round(cnn_mean["model_size_mb"],2)} MB')
print(f'MobileNetV2 - Ukuran Model : {round(mobilenet_mean["model_size_mb"],2)} MB')
print()
print('KESIMPULAN:')

```

```

print('MobileNetV2 unggul dalam accuracy, precision, recall, F1-
score, dan ukuran model.')
print('CNN Custom unggul dalam inference time.')

# Cell 9 - Menyusun tabel rekapitulasi akhir sebagai dasar
penulisan Bab IV

rekap_final = {
    'Model': ['CNN Custom', 'MobileNetV2'],
    'Accuracy (%)': [round(cnn_mean['accuracy']*100, 2),
round(mobilenet_mean['accuracy']*100, 2)],
    'SD Accuracy (%)': [round(cnn_std['accuracy']*100, 2),
round(mobilenet_std['accuracy']*100, 2)],
    'Precision (%)': [round(cnn_mean['precision']*100, 2),
round(mobilenet_mean['precision']*100, 2)],
    'SD Precision (%)': [round(cnn_std['precision']*100, 2),
round(mobilenet_std['precision']*100, 2)],
    'Recall (%)': [round(cnn_mean['recall']*100, 2),
round(mobilenet_mean['recall']*100, 2)],
    'SD Recall (%)': [round(cnn_std['recall']*100, 2),
round(mobilenet_std['recall']*100, 2)],
    'F1-Score (%)': [round(cnn_mean['f1_score']*100, 2),
round(mobilenet_mean['f1_score']*100, 2)],
    'SD F1-Score (%)': [round(cnn_std['f1_score']*100, 2),
round(mobilenet_std['f1_score']*100, 2)],
    'Inference Time (ms/gambar)':
[round(cnn_mean['inference_time_ms'], 2),
round(mobilenet_mean['inference_time_ms'], 2)],
    'Ukuran Model (MB)': [round(cnn_mean['model_size_mb'], 2),
round(mobilenet_mean['model_size_mb'], 2)]
}

df_rekap_final = pd.DataFrame(rekap_final)
df_rekap_final.index = df_rekap_final.index + 1

print('Tabel Rekapitulasi Final')
print('=' * 80)
display(df_rekap_final)

# Simpan ke Drive
df_rekap_final.to_csv('/content/drive/MyDrive/tabel_rekap_final.
csv', index=True)
df_per_run.to_csv('/content/drive/MyDrive/tabel_per_run.csv',
index=True)
print('Tabel tersimpan ke Drive.')

```