LAMPIRAN

Lampiran 1 Kode Program

1. Import Drive

```
from google.colab import drive
import os

# Mount Drive
drive.mount('/content/drive')

# Tentukan Path Dataset Utama
dataset_path = '/content/drive/MyDrive/Dataset_Rerajahan_Ulap_Ulap'

# Fungsi Validasi Struktur Folder
def cek_struktur_dataset(subfolder_name):
    target_path = os.path.join(dataset_path, subfolder_name)

    if os.path.exists(target_path):
        sub_folders = [f for f in os.listdir(target_path)
                       if os.path.isdir(os.path.join(target_path, f))
        ]
        print(f"\n--- Memeriksa Folder {subfolder_name.upper()} ---")
        print(f"Total kelas ditemukan: {len(sub_folders)} (9 Class)")
        print(f"Daftar kelas: {sorted(sub_folders)}")
    else:
        print(f"\nPeringatan: Folder {subfolder_name} tidak ditemukan di
{target_path}")

# Eksekusi pengecekan
cek_struktur_dataset('train')
cek_struktur_dataset('test')
```

2. Resize Citra

```
import cv2
import os
import numpy as np

# Path target size untuk arsitektur CNN (ResNet50 / MobileNetV2)
target_size = (224, 224)

def resize_images_in_folder(main_path):
    if not os.path.exists(main_path):
        print(f"Peringatan: Folder {main_path} tidak ditemukan!")
        return

    counter = 0
    for root, dirs, files in os.walk(main_path):
        for file in files:
            if file.lower().endswith(('.png', '.jpg', '.jpeg')):
                file_path = os.path.join(root, file)

                # Baca gambar
                img = cv2.imread(file_path)

                if img is not None:
                    # Resize menggunakan INTER_AREA
                    resized_img = cv2.resize(img, target_size,
interpolation=cv2.INTER_AREA)

                    # Simpan kembali (Menimpa file asli)
                    cv2.imwrite(file_path, resized_img)
                    counter += 1
                else:
                    print(f"Gagal membaca file (kemungkinan rusak):
{file_path}")

    print(f"Selesai! Berhasil me-resize {counter} gambar di: {main_path}")

# Eksekusi resize secara terpisah untuk Train dan Test
print("Memulai proses resize citra ke 224x224...")
```

```
resize_images_in_folder(os.path.join(dataset_path, 'train'))
resize_images_in_folder(os.path.join(dataset_path, 'test'))
print("Semua tahapan resize selesai.")
```

3. Augmentasi Citra

```
import os
import cv2
import numpy as np
import random
import matplotlib.pyplot as plt

# 1. SETTING PATH
path_asal = '/content/drive/Dataset_Rerajahan_Ulap_Ulap/train'
path_tujuan = '/content/drive/MyDrive/Dataset_Rerajahan_Ulap_Ulap/train_aug'

os.makedirs(path_tujuan, exist_ok=True)

print("Sedang memproses augmentasi... Mohon tunggu.")

sampel_kelas = None
sampel_file_base = None

for kelas in sorted(os.listdir(path_asal)):
    folder_kelas_asal = os.path.join(path_asal, kelas)
    folder_kelas_tujuan = os.path.join(path_tujuan, kelas)

    if not os.path.isdir(folder_kelas_asal):
        continue
    os.makedirs(folder_kelas_tujuan, exist_ok=True)

    daftar_file = [f for f in os.listdir(folder_kelas_asal) if
f.lower().endswith(('.png', '.jpg', '.jpeg'))]

    for nama_file in daftar_file:
        img = cv2.imread(os.path.join(folder_kelas_asal, nama_file))
        if img is None:
            continue

        base = os.path.splitext(nama_file)[0]

        if sampel_file_base is None:
            sampel_kelas = kelas
            sampel_file_base = base

        # --- 0. Simpan Gambar Asli ---
        cv2.imwrite(os.path.join(folder_kelas_tujuan, f"{base}_orig.jpg"), img)

        # --- 1. Brightness ---
        bright = cv2.convertScaleAbs(img, beta=45)
        cv2.imwrite(os.path.join(folder_kelas_tujuan, f"{base}_brt.jpg"),
bright)

        # --- 2. Contrast ---
        contrast = cv2.convertScaleAbs(img, alpha=1.4)
        cv2.imwrite(os.path.join(folder_kelas_tujuan, f"{base}_cnt.jpg"),
contrast)

        # --- 3. Noise ---
        mean = 0
        sigma = 10
        gaussian_noise = np.random.normal(mean, sigma,
img.shape).astype(np.float32)
```

```
        noised = np.clip(img.astype(np.float32) + gaussian_noise, 0,
255).astype(np.uint8)
        cv2.imwrite(os.path.join(folder_kelas_tujuan, f"{base}_nos.jpg"),
noised)

        # --- 4. Shear ---
        M = np.float32([[1, 0.2, 0], [0, 1, 0]])
        sheared = cv2.warpAffine(img, M, (int(img.shape[1]*1.2), img.shape[0]),
borderMode=cv2.BORDER_REFLECT)
        sheared = cv2.resize(sheared, (224, 224))
        cv2.imwrite(os.path.join(folder_kelas_tujuan, f"{base}_shr.jpg"),
sheared)

        # --- 5. Blur ---
        blurred = cv2.GaussianBlur(img, (5, 5), 0)
        cv2.imwrite(os.path.join(folder_kelas_tujuan, f"{base}_blr.jpg"),
blurred)

print(f"Selesai! Data augmentasi tersimpan di: {path_tujuan}\n")
```

4. Train Model MobileNetV2 + Nadam

```
import os, time, psutil, gc, pickle
import numpy as np
import tensorflow as tf
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D, Dropout
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Nadam
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint, Callback
from sklearn.model_selection import StratifiedKFold

path_train_aug = '/content/drive/MyDrive//Dataset_Rerajahan_Ulap_Ulap/train'
save_dir = '/content/drive/MyDrive/Models/MobileNetV2/Nadam_Scenario'
os.makedirs(save_dir, exist_ok=True)

IMG_SIZE = (224, 224)
BATCH_SIZE = 32
EPOCHS = 100
PATIENCE = 10
DROPOUT_RATE = 0.6
LEARNING_RATE = 0.0001
K_FOLDS = 5

def load_data(path):
    images, labels = [], []
    class_names = sorted([d for d in os.listdir(path) if
os.path.isdir(os.path.join(path, d))])
    print(f" Sedang memuat data dari {len(class_names)} kelas...")

    for i, cls in enumerate(class_names):
        cls_path = os.path.join(path, cls)
        files = [f for f in os.listdir(cls_path) if f.lower().endswith(('.jpg',
'.jpeg', '.png'))]
        for img_name in files:
            img_path = os.path.join(cls_path, img_name)
            img = tf.keras.preprocessing.image.load_img(img_path,
target_size=IMG_SIZE)
            img = tf.keras.preprocessing.image.img_to_array(img) / 255.0
            images.append(img)
            labels.append(i)
```

```
    return np.array(images, dtype=np.float32), np.array(labels, dtype=np.int32),
class_names

X, y, class_names = load_data(path_train_aug)
num_classes = len(class_names)
print(f" Total gambar berhasil dimuat ke array: {len(X)}")

class PerformanceMonitor(Callback):
    def on_train_begin(self, logs={}):
        self.start_time, self.epoch_times, self.ram_usage = time.time(), [], []
        self.best_val_acc, self.best_epoch = -1.0, 1

    def on_epoch_begin(self, epoch, logs={}):
        self.ep_start = time.time()

    def on_epoch_end(self, epoch, logs={}):
        duration = time.time() - self.ep_start
        self.epoch_times.append(duration)
        ram = psutil.virtual_memory().used / (1024 ** 2)
        self.ram_usage.append(ram)

        val_acc = logs.get('val_accuracy', 0)
        if val_acc > self.best_val_acc:
            self.best_val_acc, self.best_epoch = val_acc, epoch + 1
        print(f" - Train Time: {duration:.2f}s | RAM Usage: {ram:.1f}MB | Best
Ep: {self.best_epoch}")

    def on_train_end(self, logs={}):
        self.total_time = time.time() - self.start_time

def build_model_mobilenet_nadam(num_classes):
    tf.keras.backend.clear_session()
    base_model = MobileNetV2(weights='imagenet', include_top=False,
input_shape=(224, 224, 3))
    base_model.trainable = False

    x = base_model.output
    x = GlobalAveragePooling2D()(x)
    x = Dense(256, activation='relu')(x)
    x = Dropout(DROPOUT_RATE)(x)
    predictions = Dense(num_classes, activation='softmax')(x)

    model = Model(inputs=base_model.input, outputs=predictions)

    optimizer = Nadam(learning_rate=LEARNING_RATE)

    model.compile(optimizer=optimizer, loss='categorical_crossentropy',
metrics=['accuracy'])
    return model

skf = StratifiedKFold(n_splits=K_FOLDS, shuffle=True, random_state=42)
fold_no = 1
all_fold_histories = []

for train_idx, val_idx in skf.split(X, y):
    print(f"\n" + "="*60)
    print(f" MEMULAI TRAINING FOLD KE-{fold_no} DARI {K_FOLDS} (SKENARIO:
NADAM)")
    print("="*60)

    X_train, X_val = X[train_idx], X[val_idx]
    y_train = tf.keras.utils.to_categorical(y[train_idx], num_classes)
    y_val = tf.keras.utils.to_categorical(y[val_idx], num_classes)
```

```
    model = build_model_mobilenet_nadam(num_classes)
    monitor_cb = PerformanceMonitor()

    ckpt_path = os.path.join(save_dir, f'best_nadam_fold_{fold_no}.h5')

    callbacks = [
        EarlyStopping(monitor='val_loss', patience=PATIENCE,
restore_best_weights=True, verbose=1),
        ModelCheckpoint(ckpt_path, monitor='val_accuracy', save_best_only=True,
mode='max', verbose=1),
        monitor_cb
    ]

    history = model.fit(
        X_train, y_train,
        validation_data=(X_val, y_val),
        epochs=EPOCHS,
        batch_size=BATCH_SIZE,
        callbacks=callbacks,
        verbose=1
    )

    b_idx = monitor_cb.best_epoch - 1
    h_dict = history.history

    h_dict.update({
        'train_time_per_epoch': monitor_cb.epoch_times,
        'ram_usage': monitor_cb.ram_usage,
        'fold_index': fold_no,
        'best_epoch_saved': monitor_cb.best_epoch,
        'total_fold_time_seconds': round(monitor_cb.total_time, 2),
        'model_size_kb': round(os.path.getsize(ckpt_path) / 1024, 3) if
os.path.exists(ckpt_path) else 0,
        'best_loss': h_dict['loss'][b_idx],
        'best_accuracy': h_dict['accuracy'][b_idx],
        'best_val_loss': h_dict['val_loss'][b_idx],
        'best_val_accuracy': h_dict['val_accuracy'][b_idx]
    })

    all_fold_histories.append(h_dict)

    with open(os.path.join(save_dir, f'history_fold_{fold_no}.pkl'), 'wb') as f:
        pickle.dump(h_dict, f)

    with open(os.path.join(save_dir, 'all_fold_histories_mobilenet_nadam.pkl'),
'wb') as f:
        pickle.dump(all_fold_histories, f)

    print(f"DATA FOLD {fold_no} SELESAI. Berkas .h5 dan .pkl skenario Nadam
berhasil diamankan.")

    del model, X_train, X_val, y_train, y_val, history
    gc.collect()
    fold_no += 1

print(f"\n PROSES SELESAI FULL. Seluruh data skenario Nadam tersimpan di:
{save_dir}")
```

5. Evaluasi Model MobileNetV2 + Nadam

```
import os
import numpy as np
import pandas as pd
import tensorflow as tf
```

```
import pickle
import gc
from sklearn.metrics import accuracy_score, precision_recall_fscore_support
from tensorflow.keras.models import load_model

MODEL_DIR = '/content/drive/MyDrive/Models/MobileNetV2/Nadam_Scenario'
TEST_DIR = '/content/drive/MyDrive/Dataset_Rerajahan_Ulap_Ulap/test'
IMG_SIZE = (224, 224)

def load_test_data(path):
    images, labels = [], []
    class_names = sorted([d for d in os.listdir(path) if
os.path.isdir(os.path.join(path, d))])
    print(f" Memuat data uji murni dari {len(class_names)} kelas...")
    for i, cls in enumerate(class_names):
        cls_path = os.path.join(path, cls)
        files = [f for f in os.listdir(cls_path) if f.lower().endswith(('.jpg',
'.jpeg', '.png'))]
        for img_name in files:
            img_path = os.path.join(cls_path, img_name)
            img = tf.keras.preprocessing.image.load_img(img_path,
target_size=IMG_SIZE)
            images.append(tf.keras.preprocessing.image.img_to_array(img) /
255.0)
            labels.append(i)
    return np.array(images, dtype=np.float32), np.array(labels, dtype=np.int32),
class_names

X_test, y_test, class_names = load_test_data(TEST_DIR)

results = []
for i in range(1, 6):
    model_path = os.path.join(MODEL_DIR, f'best_nadam_fold_{i}.h5')
    history_file = os.path.join(MODEL_DIR, f'history_fold_{i}.pkl')

    if os.path.exists(model_path):
        model = load_model(model_path)
        y_pred = np.argmax(model.predict(X_test, batch_size=32, verbose=0),
axis=1)

        acc = accuracy_score(y_test, y_pred)
        p, r, f1, _ = precision_recall_fscore_support(y_test, y_pred,
average='macro')

        try:
            with open(history_file, 'rb') as f:
                hist = pickle.load(f)
                avg_time_epoch = np.mean(hist.get('train_time_per_epoch', 0))
                total_fold_time = hist.get('total_fold_time_seconds', 0)
                max_ram = np.max(hist.get('ram_usage', 0))
                model_size_kb = hist.get('model_size_kb', 0)
                best_epoch_saved = hist.get('best_epoch_saved', 0)
                total_epochs = len(hist.get('loss', 0))
        except FileNotFoundError:
            print(f" Berkas histori .pkl untuk Fold {i} tidak ditemukan di
folder Nadam. Menggunakan nilai default.")
            avg_time_epoch, total_fold_time, max_ram, model_size_kb,
best_epoch_saved, total_epochs = 0, 0, 0, 0, 0, 0

        results.append([
            acc, p, r, f1,
            avg_time_epoch, total_fold_time, max_ram, model_size_kb,
            best_epoch_saved, total_epochs
        ])
```

```
        print(f" Fold {i} Berhasil Dievaluasi | Acc: {acc*100:.2f}% | F1:
{f1*100:.2f}%")

        del model
        tf.keras.backend.clear_session()
        gc.collect()
    else:
        print(f" Berkas model fold {i} tidak ditemukan di: {model_path}")

columns = [
    'Accuracy', 'Precision', 'Recall', 'F1-Score',
    'Avg Train Time/Epoch (s)', 'Total Fold Time (s)', 'Max RAM Usage (MB)',
    'Model Size (KB)', 'Best Epoch Saved', 'Total Epochs'
]
df = pd.DataFrame(results, columns=columns, index=[f'Fold {i}' for i in range(1,
len(results)+1)])

df.loc['RATA-RATA'] = df.mean()
df.loc['STD DEV'] = df.std()

df.at['STD DEV', 'Best Epoch Saved'] = np.nan
df.at['STD DEV', 'Total Epochs'] = np.nan

print("\n--- TABEL EVALUASI INTEGRASI METRIK LENGKAP MOBILENETV2 (NADAM) ---")
display(df.round(4))

csv_output = os.path.join(MODEL_DIR, 'tabel_evaluasi_nadam.csv')
df.to_csv(csv_output)
print(f"\n Tabel komprehensif skenario Nadam berhasil disimpan di:
{csv_output}")
```

6. Evaluasi *Confusion Matrix* MobileNetV2 + Nadam

```
import os
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import tensorflow as tf
import gc
from sklearn.metrics import confusion_matrix
from tensorflow.keras.models import load_model

MODEL_DIR = '/content/drive/MyDrive/Models/MobileNetV2/Nadam_Scenario'
TEST_DIR = '/content/drive/MyDrive/Dataset_Rerajahan_Ulap_Ulap/test'
IMG_SIZE = (224, 224)

def load_test_data(path):
    images, labels = [], []
    class_names = sorted([d for d in os.listdir(path) if
os.path.isdir(os.path.join(path, d))])
    print(f" Memuat data uji dari {len(class_names)} kelas...")
    for i, cls in enumerate(class_names):
        cls_path = os.path.join(path, cls)
        files = [f for f in os.listdir(cls_path) if f.lower().endswith(('.jpg',
'.jpeg', '.png'))]
        for img_name in files:
            img_path = os.path.join(cls_path, img_name)
            img = tf.keras.preprocessing.image.load_img(img_path,
target_size=IMG_SIZE)
            images.append(tf.keras.preprocessing.image.img_to_array(img) /
255.0)
            labels.append(i)
    return np.array(images, dtype=np.float32), np.array(labels, dtype=np.int32),
class_names

X_test, y_test, class_names = load_test_data(TEST_DIR)
num_classes = len(class_names)
```

```
combined_cm_nadam = np.zeros((num_classes, num_classes))

print("\n Menyusun Confusion Matrix Akumulasi Nadam (.h5 format)...")
for i in range(1, 6):
    model_path = os.path.join(MODEL_DIR, f'best_nadam_fold_{i}.h5')

    if os.path.exists(model_path):
        model = load_model(model_path)

        y_pred = np.argmax(model.predict(X_test, batch_size=32, verbose=0),
axis=1)

        combined_cm_nadam += confusion_matrix(y_test, y_pred,
labels=range(num_classes))
        print(f" Fold {i} Berhasil Diakumulasikan")

        del model
        tf.keras.backend.clear_session()
        gc.collect()
    else:
        print(f" Berkas model fold {i} tidak ditemukan di: {model_path}")

plt.figure(figsize=(11, 9))

row_sums = combined_cm_nadam.sum(axis=1)[:, np.newaxis]
cm_normalized = np.divide(combined_cm_nadam, row_sums,
out=np.zeros_like(combined_cm_nadam), where=row_sums!=0)

sns.heatmap(cm_normalized,
            annot=True, fmt='.2%', cmap='Blues',
            xticklabels=class_names, yticklabels=class_names,
            annot_kws={"size": 10})

plt.title('Normalized Confusion Matrix\nMobileNetV2 + 5-Fold Cross Validation +
Nadam', fontsize=14, pad=15)
plt.xlabel('Predicted Label', fontsize=12, labelpad=10)
plt.ylabel('Actual Label', fontsize=12, labelpad=10)
plt.xticks(rotation=45, ha='right')
plt.yticks(rotation=0)
plt.tight_layout()

output_image_path = os.path.join(MODEL_DIR, 'confusion_matrix_nadam.png')
plt.savefig(output_image_path, dpi=300)
print(f"\n Grafik Confusion Matrix Nadam berhasil disimpan di:
{output_image_path}")

plt.show()
```

7. Grafik Pembelajaran MobileNetV2 + Nadam

```
import os
import pickle
import numpy as np
import matplotlib.pyplot as plt

MODEL_DIR = '/content/drive/MyDrive/Models/MobileNetV2/Nadam_Scenario'
history_path = os.path.join(MODEL_DIR, 'all_fold_histories_mobilenet_nadam.pkl')

if os.path.exists(history_path):
    with open(history_path, 'rb') as f:
        all_fold_histories = pickle.load(f)
    print(f" Berhasil memuat {len(all_fold_histories)} histori fold dari folder
Nadam_Scenario.")
else:
    raise FileNotFoundError(f" Berkas histori tidak ditemukan di path:
{history_path}")

min_epochs = min([len(h['accuracy']) for h in all_fold_histories])
print(f" Batas epoch minimum yang diambil untuk grafik rata-rata Nadam:
{min_epochs} epoch.")
```

```
train_acc_all = np.array([h['accuracy'][:min_epochs] for h in
all_fold_histories])
val_acc_all = np.array([h['val_accuracy'][:min_epochs] for h in
all_fold_histories])
train_loss_all = np.array([h['loss'][:min_epochs] for h in all_fold_histories])
val_loss_all = np.array([h['val_loss'][:min_epochs] for h in
all_fold_histories])

avg_train_acc = np.mean(train_acc_all, axis=0)
avg_val_acc = np.mean(val_acc_all, axis=0)
avg_train_loss = np.mean(train_loss_all, axis=0)
avg_val_loss = np.mean(val_loss_all, axis=0)

epochs = range(1, min_epochs + 1)

plt.figure(figsize=(15, 6))

plt.subplot(1, 2, 1)
plt.plot(epochs, avg_train_acc, 'b-', label='Avg Training Accuracy',
linewidth=2)
plt.plot(epochs, avg_val_acc, 'r-', label='Avg Validation Accuracy',
linewidth=2)

plt.fill_between(epochs, avg_train_acc - np.std(train_acc_all, axis=0),
                 avg_train_acc + np.std(train_acc_all, axis=0), color='blue',
alpha=0.1)
plt.fill_between(epochs, avg_val_acc - np.std(val_acc_all, axis=0),
                 avg_val_acc + np.std(val_acc_all, axis=0), color='red',
alpha=0.1)

plt.title('Average Accuracy\nMobileNetV2 + 5-Fold Cross Validation + Nadam',
fontsize=12, pad=10)
plt.xlabel('Epochs', fontsize=11)
plt.ylabel('Accuracy', fontsize=11)
plt.legend(loc='lower right')
plt.grid(True, alpha=0.3)

plt.subplot(1, 2, 2)
plt.plot(epochs, avg_train_loss, 'b-', label='Avg Training Loss', linewidth=2)
plt.plot(epochs, avg_val_loss, 'r-', label='Avg Validation Loss', linewidth=2)

plt.fill_between(epochs, avg_train_loss - np.std(train_loss_all, axis=0),
                 avg_train_loss + np.std(train_loss_all, axis=0), color='blue',
alpha=0.1)
plt.fill_between(epochs, avg_val_loss - np.std(val_loss_all, axis=0),
                 avg_val_loss + np.std(val_loss_all, axis=0), color='red',
alpha=0.1)

plt.title('Average Loss\nMobileNetV2 + 5-Fold Cross Validation + Nadam',
fontsize=12, pad=10)
plt.xlabel('Epochs', fontsize=11)
plt.ylabel('Loss', fontsize=11)
plt.legend(loc='upper right')
plt.grid(True, alpha=0.3)

plt.tight_layout()

output_chart_path = os.path.join(MODEL_DIR, 'learning_curve_average_nadam.png')
plt.savefig(output_chart_path, dpi=300)
print(f"\n Grafik Kurva Pembelajaran rata-rata Nadam berhasil disimpan di:
{output_chart_path}")

plt.show()
```