#import

import tensorflow as tf

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.applications import Xception

from tensorflow.keras.optimizers import Adam, Adamax

from tensorflow.keras.models import Model

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dropout, Flatten, Dense

from sklearn.metrics import confusion\_matrix, classification\_report, accuracy\_score

import seaborn as sns

import os

import matplotlib.pyplot as plt

import numpy as np

# Path ke folder dataset

dataset\_path = 'dataset2/'

train\_dir = os.path.join(dataset\_path, 'train')

valid\_dir = os.path.join(dataset\_path, 'valid')

test\_dir = os.path.join(dataset\_path, 'test')

# Parameter

IMG\_SIZE = (224, 224, 3)

BATCH\_SIZE = 16

LEARNING\_RATE = 0.0001

EPOCHS = 30

# Normalisasi data dan augmentasi data menggunakan ImageDataGenerator

train\_datagen = ImageDataGenerator(

rescale=1.0/255.0,

rotation\_range=30,

width\_shift\_range=0.2,

height\_shift\_range=0.2,

shear\_range=0.2,

zoom\_range=0.2,

horizontal\_flip=True,

fill\_mode='nearest'

)

valid\_datagen = ImageDataGenerator(rescale=1.0/255.0)

test\_datagen = ImageDataGenerator(rescale=1.0/255.0)

# Generator data

train\_generator = train\_datagen.flow\_from\_directory(train\_dir,

target\_size=IMG\_SIZE[:2],

batch\_size=BATCH\_SIZE,

class\_mode='categorical')

valid\_generator = valid\_datagen.flow\_from\_directory(valid\_dir,

target\_size=IMG\_SIZE[:2],

batch\_size=BATCH\_SIZE,

class\_mode='categorical')

test\_generator = test\_datagen.flow\_from\_directory(test\_dir,

target\_size=IMG\_SIZE[:2],

batch\_size=BATCH\_SIZE,

class\_mode='categorical')

# Mendapatkan jumlah kelas dari data latih

class\_count = len(train\_generator.class\_indices)

# Load model Xception tanpa bagian fully connected (include\_top=False)

base\_model = Xception(weights='imagenet', include\_top=False, input\_shape=IMG\_SIZE)

# Freeze semua layer pretrained

for layer in base\_model.layers:

layer.trainable = False

# Menambahkan layer custom untuk klasifikasi

x = Conv2D(64, (3, 3), activation='relu', padding='same', strides=1)(base\_model.output)

x = MaxPooling2D(pool\_size=(2, 2))(x)

x = Dropout(0.5)(x)

x = Flatten()(x)

x = Dense(128, activation='relu')(x)

x = Dropout(0.5)(x)

output\_layer = Dense(class\_count, activation='softmax')(x)

# Membuat model akhir

model = Model(inputs=base\_model.input, outputs=output\_layer)

# Kompilasi model

model.compile(optimizer=tf.keras.optimizers.Adam(learning\_rate=LEARNING\_RATE),

loss='categorical\_crossentropy',

metrics=['accuracy'])

# Melatih model

history = model.fit(train\_generator,

validation\_data=valid\_generator,

epochs=EPOCHS,

steps\_per\_epoch=train\_generator.samples // BATCH\_SIZE,

validation\_steps=valid\_generator.samples // BATCH\_SIZE)

# Menemukan best epoch berdasarkan loss dan accuracy

best\_epoch\_loss = np.argmin(history.history['val\_loss']) + 1

best\_epoch\_accuracy = np.argmax(history.history['val\_accuracy']) + 1

# Membuat grafik Training & Validation Loss dan Accuracy

fig, axes = plt.subplots(1, 2, figsize=(14, 6))

# Loss

axes[0].plot(history.history['loss'], label='Training Loss', color='blue', linestyle='-')

axes[0].plot(history.history['val\_loss'], label='Validation Loss', color='red', linestyle='--')

axes[0].scatter(best\_epoch\_loss - 1, min(history.history['val\_loss']), color='purple', label=f'Best Epoch (Loss): {best\_epoch\_loss}')

axes[0].set\_title('Training and Validation Loss')

axes[0].set\_xlabel('Epochs')

axes[0].set\_ylabel('Loss')

axes[0].legend()

axes[0].grid(True)

# Accuracy

axes[1].plot(history.history['accuracy'], label='Training Accuracy', color='green', linestyle='-')

axes[1].plot(history.history['val\_accuracy'], label='Validation Accuracy', color='orange', linestyle='--')

axes[1].scatter(best\_epoch\_accuracy - 1, max(history.history['val\_accuracy']), color='purple', label=f'Best Epoch (Accuracy): {best\_epoch\_accuracy}')

axes[1].set\_title('Training and Validation Accuracy')

axes[1].set\_xlabel('Epochs')

axes[1].set\_ylabel('Accuracy')

axes[1].legend()

axes[1].grid(True)

plt.tight\_layout()

plt.show()

print(f"\nBest Epoch berdasarkan Validation Loss: Epoch {best\_epoch\_loss}")

print(f"Validation Loss Terendah: {min(history.history['val\_loss']):.4f}")

print(f"\nBest Epoch berdasarkan Validation Accuracy: Epoch {best\_epoch\_accuracy}")

print(f"Validation Accuracy Tertinggi: {max(history.history['val\_accuracy']):.4f}")

# Simpan model ke file .h5

model.save('model\_percobaanx.h5')

print("Model berhasil disimpan ke file 'model\_percobaanx.h5'")

# Evaluasi model

test\_loss, test\_accuracy = model.evaluate(test\_generator)

print(f"Test Loss: {test\_loss:.4f}")

print(f"Test Accuracy: {test\_accuracy \* 100:.2f}%")

# Simpan model ke file .h5

model.save('model\_percobaanx.h5')

print("Model berhasil disimpan ke file 'model\_percobaanx.h5'")

#convusion metrix

from tensorflow.keras.models import load\_model

from sklearn.metrics import confusion\_matrix, classification\_report

# 1. Muat model

model = load\_model('model\_percobaanx.h5')

# 2. Memproses data uji

test\_dir = 'dataset2/test' # Ganti dengan path folder test

IMG\_SIZE = (224, 224) # Ukuran input sesuai model

BATCH\_SIZE = 32

test\_datagen = ImageDataGenerator(rescale=1./255) # Rescale pixel value ke [0,1]

test\_generator = test\_datagen.flow\_from\_directory(

test\_dir,

target\_size=IMG\_SIZE,

batch\_size=BATCH\_SIZE,

class\_mode='categorical',

shuffle=False # Jangan acak karena akan dibandingkan dengan label asli

)

# 3. Prediksi kelas

predictions = model.predict(test\_generator)

y\_pred = np.argmax(predictions, axis=1) # Kelas prediksi

y\_true = test\_generator.classes # Kelas asli

# 4. Membuat confusion matrix

conf\_matrix = confusion\_matrix(y\_true, y\_pred)

# 5. Visualisasi confusion matrix

plt.figure(figsize=(8, 6))

sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='Blues',

xticklabels=test\_generator.class\_indices.keys(),

yticklabels=test\_generator.class\_indices.keys())

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.title('Confusion Matrix')

plt.show()

# 6. Classification report (opsional)

print("Classification Report:")

print(classification\_report(y\_true, y\_pred, target\_names=test\_generator.class\_indices.keys()))