Nama: Isa Aulia Almadani
Dom: Sukoharjo

Proyek_Klasifikasi_Image_from_GDSC_by Isa Aulia Almadani

Project Overview: Klasifikasi Gambar Butir Padi Menggunakan Mobile Net V2

Proyek ini berfokus pada klasifikasi gambar butir padi menggunakan arsitektur model Mobile Net V2 yang telah dilatih sebelumnya. Mobile Net V2 dipilih karena efisiensi dan performanya yang baik dalam tugas-tugas klasifikasi gambar pada perangkat dengan keterbatasan sumber daya.

Tujuan Proyek

Tujuan dari proyek ini adalah untuk mengembangkan model yang mampu mengklasifikasikan gambar butir padi dengan akurasi tinggi. Model ini diharapkan dapat membantu dalam proses seleksi dan pengawasan kualitas butir padi secara otomatis.

Fitur Utama

- Import Library Mengimpor pustaka-pustaka yang diperlukan untuk pemrosesan data, pelatihan model, dan visualisasi hasil.
- Import Dataset from Kaggle Mengunduh dan memuat dataset gambar butir padi dari Kaggle untuk digunakan dalam pelatihan dan pengujian model.
- Preprocessing Melakukan praproses pada dataset, termasuk perubahan ukuran gambar, normalisasi, dan augmentasi data untuk meningkatkan kinerja model.
- Building Model with Pretrained Model Menggunakan Mobile Net V2 yang telah dilatih sebelumnya sebagai dasar untuk membangun model klasifikasi. Langkah ini melibatkan fine-tuning model untuk menyesuaikan dengan dataset butir padi.
- Plot Visualization Membuat visualisasi data dan hasil pelatihan model, termasuk grafik akurasi dan kehilangan (loss) selama proses pelatihan.
- Predict Image Menggunakan model yang telah dilatih untuk memprediksi kelas dari gambar butir padi baru, dan menampilkan hasil prediksi tersebut.

Teknologi yang Digunakan

- Python: Bahasa pemrograman utama yang digunakan dalam proyek ini.

- TensorFlow/Keras: Framework deep learning yang digunakan untuk membangun dan melatih model.
- Kaggle: Sumber dataset gambar butir padi.
- Matplotlib/Seaborn: Pustaka visualisasi yang digunakan untuk membuat plot dan grafik.

Rencana Pengembangan

- Mengoptimalkan model dengan teknik-teknik tambahan seperti fine-tuning dan hyperparameter tuning.
- Menerapkan teknik augmentasi data yang lebih lanjut untuk meningkatkan kinerja model.
- Mengintegrasikan model dengan aplikasi berbasis web untuk demo prediksi gambar secara real-time.

Hasil yang Diharapkan

- Menghasilkan model klasifikasi gambar butir padi dengan akurasi tinggi.
- Memudahkan proses seleksi dan pengawasan kualitas butir padi melalui otomatisasi.
- Memberikan kontribusi terhadap pengembangan teknologi pertanian berbasis AI di Indonesia.

Manage file import from kaggle

```
!pip install -q kaggle
from google.colab import files
files.upload()
```

```
Pilih File Tidak ada file yang dipilih  Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving kaggle.json to kaggle.json
{'kaggle.json': b'{"username":"almadzzz","key":"ba1252fd1a739eae3dac7ba3e4a53952"}'}
```

```

!mkdir ~/.kaggle
!mv kaggle.json ~/.kaggle
!chmod 600 ~/.kaggle/kaggle.json
```

```
!kaggle datasets list -s ' rice dataset'
```

```
ref                                                  title                          
---------------------------------------------------  -----------------------------
muratkokludataset/rice-image-dataset                 Rice Image Dataset
muratkokludataset/rice-dataset-commeo-and-osmancik   Rice Dataset Commeo and Osmanc
muratkokludataset/rice-msc-dataset                   Rice MSC Dataset
vbookshelf/rice-leaf-diseases                        Rice Leaf Diseases Dataset
minhhuy2810/rice-diseases-image-dataset              Rice Diseases Image Dataset
rajkumar898/rice-plant-dataset                       Rice Plant Dataset
aman2000jaiswal/agriculture-crop-images              Agriculture crop images
mssmartypants/rice-type-classification               Rice type classification
```

```
maimunulkjisan/rice-leaf-dataset-from-mendeley-data    Rice Leafs Disease Dataset
shayanriyaz/riceleafs                                  Rice Leafs
mdwaquarazam/agricultural-crops-image-classification   Agricultural crops image class
abhijitdahatonde/crop-production-1996-to-2021          🌽 Crop Production 1996 to 20
seymasa/rice-dataset-gonenjasmine                      Rice Seed Dataset (Gonen&Jasmi
nafishamoin/new-bangladeshi-crop-disease               New Bangladeshi Crop Disease
zsinghrahulk/rice-pest-and-diseases                    Rice - Pest and Diseases
nischallal/rice-disease-dataset                        Rice Disease Dataset
fhabibimoghaddam/five-different-rice-image-dataset     Rice Image Dataset
trolukovich/food11-image-dataset                       Food-11 image dataset
timmofeyy/-cerial-prices-changes-within-last-30-years  🌽 Cerial Prices Changes With
asheniranga/leaf-disease-dataset-combination           Leaf Disease Dataset (combinat
```

```
!kaggle datasets download -d 'muratkokludataset/rice-image-dataset    '
```

```
Downloading rice-image-dataset    .zip to /content
 98% 216M/219M [00:01<00:00, 167MB/s]
100% 219M/219M [00:01<00:00, 134MB/s]
```

Prepare Dataset

Import Library

```
import os
import json
import zipfile


# from keras.layers import Input
from keras.applications import EfficientNetB7, MobileNetV2, EfficientNetV2M

import tensorflow as tf
from keras.layers import Flatten
from keras.models import Sequential
from keras.layers import GlobalAveragePooling2D, Dense, BatchNormalization, Conv2D, Dr
from keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.callbacks import ReduceLROnPlateau, ModelCheckpoint
import matplotlib.pyplot as plt
from warnings import filterwarnings
from sklearn.metrics import classification_report, confusion_matrix

%matplotlib inline
import matplotlib.pyplot as plt
```

```
filezip = "rice-image-dataset    .zip"
extractZip = zipfile.ZipFile(filezip, 'r')
extractZip.extractall("datasets")
```

```
os.listdir("/content/datasets/Rice_Image_Dataset")
```

```
['Rice_Citation_Request.txt',
 'Jasmine',
 'Karacadag',
 'Ipsala',
 'Basmati',
 'Arborio']
```

```

dataset_path = '/content/datasets/Rice_Image_Dataset'
class_samples = {}

# Loop melalui setiap kelas di dalam folder dataset
for class_folder in os.listdir(dataset_path):
    class_path = os.path.join(dataset_path, class_folder)

    if os.path.isdir(class_path):
        num_class = len([file for file in os.listdir(class_path) if file.endswith('.jp

        class_samples[class_folder] = num_class

        # Loop melalui setiap file di dalam kelas
        for dirname, _, filenames in os.walk(class_path):
            for filename in filenames:
                file_path = os.path.join(dirname, filename)
                print(file_path)

for class_name, num_class in class_samples.items():
    print(f"Kelas '{class_name}': {num_class} images")
```

```
Streaming output truncated to the last 5000 lines.
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (4643).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (9692).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (3741).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (1168).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (1812).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (4867).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (3758).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (13207).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (8483).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (7838).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (8613).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (12614).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (9600).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (38).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (4597).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (13913).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (12845).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (14637).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (191).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (8782).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (3409).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (3449).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (14868).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (14427).jpg
```

```
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (13720).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (7418).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (9572).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (10467).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (9844).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (4011).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (6933).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (9931).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (14412).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (7107).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (7538).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (9853).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (1453).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (2620).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (13660).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (5285).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (3099).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (13654).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (6213).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (1276).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (4571).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (1128).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (1624).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (12760).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (10817).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (4002).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (6505).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (8935).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (7210).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (10021).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (10142).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (52).jpg
/content/datasets/Rice_Image_Dataset/Ipsala/Ipsala (9607).jpg
```

Preprocessing

```
1 dataset_dir = "/content/datasets/Rice_Image_Dataset"
2
```

```
data_gen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    vertical_flip=True,
    fill_mode="nearest",
    validation_split=0.2
)
train_generator = data_gen.flow_from_directory(
    dataset_dir,
    target_size=(224, 224),
    batch_size=64,
    shuffle = True,
    color_mode = 'rgb',
    class_mode="categorical",
    subset="training"
)

validation_generator = data_gen.flow_from_directory(
    dataset_dir,
    target_size=(224, 224),
    batch_size=64,
    shuffle = True,
    color_mode = 'rgb',
    class_mode="categorical",
    subset="validation"
)


```

```
Found 60000 images belonging to 5 classes.
Found 15000 images belonging to 5 classes.
```

Building model

```
pre_trained_model = MobileNetV2(weights="imagenet", include_top=False,
                                input_shape=(224, 224, 3))

pre_trained_model.trainable = False

model = Sequential([
    pre_trained_model,
    Conv2D(32, (3, 3), activation='relu', padding='same'),
    MaxPooling2D(pool_size=(2, 2)),
    Flatten(),
    Dropout(0.5),
    # Conv2D(64, (3, 3), activation='relu', padding='same'),
    # Dense(128, activation='relu'),
    Dense(128, activation='relu'),
    BatchNormalization(),
    Dropout(0,1),
    Dense(64, activation='relu'),
    Dropout(0.1),
    BatchNormalization(),
    Dense(5, activation='softmax')
])

learning_rate = 0.001
optimizer = 'adam'
model.compile(
    loss='categorical_crossentropy',
    optimizer=optimizer,
    metrics=['accuracy'],
)
model.summary()

```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mc
9406464/9406464 [==============================] - 0s 0us/step
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 mobilenetv2_1.00_224 (Func  (None, 7, 7, 1280)        2257984
 tional)

 conv2d (Conv2D)             (None, 7, 7, 32)          368672

 max_pooling2d (MaxPooling2  (None, 3, 3, 32)          0
 D)

 flatten (Flatten)           (None, 288)               0

 dropout (Dropout)           (None, 288)               0

 dense (Dense)               (None, 128)               36992

 batch_normalization (Batch  (None, 128)               512
 Normalization)

 dropout_1 (Dropout)         (None, 128)               0
```

```
 dense_1 (Dense)             (None, 64)                8256

 dropout_2 (Dropout)         (None, 64)                0

 batch_normalization_1 (Bat  (None, 64)                256
 chNormalization)

 dense_2 (Dense)             (None, 5)                 325

=================================================================
Total params: 2672997 (10.20 MB)
Trainable params: 414629 (1.58 MB)
Non-trainable params: 2258368 (8.61 MB)
_________________________________________________________________
```

```
# checkpoint = ModelCheckpoint(
#     "best_model.h5",
#     monitor="val_accuracy",
#     save_best_only=True,
#     mode="max",
#     verbose=2
# )
reduce_lr = ReduceLROnPlateau(monitor= 'val_accuracy', factor=0.3, patience=2, min_del

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

```

class stopCallBack(tf.keras.callbacks.Callback):
    def on_epoch_end(self, epoch, logs={}):
        if self.has_reached_accuracy(logs):
            print(' Stop training model, acc & val_acc > 92% ')
            self.model.stop_training = True

    def has_reached_accuracy(self, logs):
        return (logs.get('accuracy') > 0.92  and  logs.get('val_accuracy') > 0.92)

callbacks = stopCallBack()

history = model.fit(train_generator,
              epochs=32,
              steps_per_epoch = 19,
              validation_data=validation_generator,
              callbacks = [callbacks, reduce_lr],
              )

```

```
Epoch 1/32
19/19 [==============================] - 231s 12s/step - loss: 1.4520 - accuracy: 0.4
```

```
Epoch 2/32
19/19 [==============================] - 215s 12s/step - loss: 0.6620 - accuracy: 0.7
Epoch 3/32
19/19 [==============================] - 218s 12s/step - loss: 0.4000 - accuracy: 0.8
Epoch 4/32
19/19 [==============================] - 212s 12s/step - loss: 0.2802 - accuracy: 0.9
Epoch 5/32
19/19 [==============================] - ETA: 0s - loss: 0.2391 - accuracy: 0.9202 St
19/19 [==============================] - 213s 12s/step - loss: 0.2391 - accuracy: 0.9
```

Plot Visualization

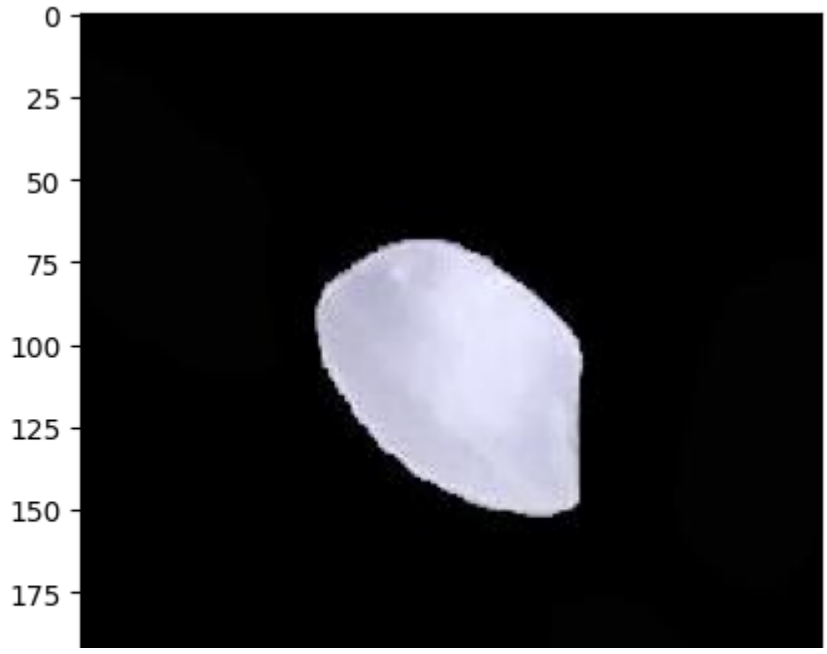
```
1
2 plt.figure(figsize=(12,4))
3
4 plt.subplot(121)
5 plt.plot(history.history['loss'], label='Train Loss')
6 plt.plot(history.history['val_loss'], label='Validation Loss')
7 plt.legend()
8
9 plt.subplot(122)
10 plt.plot(history.history['accuracy'], label='Train Accuracy')
11 plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
12 plt.legend()
13
14 plt.tight_layout()
15 plt.show()
```

Predict Image

```
import numpy as np
from tensorflow.keras.preprocessing import image
import matplotlib.image as mpimg

train_generator.class_indices
uploaded = files.upload()

for fn in uploaded.keys():
    path = fn
    img = image.load_img(path, target_size=(224, 224 ))

    imgplot = plt.imshow(img)
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0)
    images = np.vstack([x])

    classes = model.predict(images, batch_size=10)
    output_class = np.argmax(classes)
    print(fn)
    print(classes[0])
    print(output_class)

```

Pilih File Tidak ada file yang dipilih Upload widget is only available when the cell has been executed

```
Saving Karacadag (10000).jpg to Karacadag (10000).jpg
1/1 [==============================] - 0s 24ms/step
Karacadag (10000).jpg
[0.02432952 0.01136964 0.05485887 0.01845919 0.89098275]
4
```



```
train_generator.class_indices
print(train_generator.class_indices)

```

```
{'Arborio': 0, 'Basmati': 1, 'Ipsala': 2, 'Jasmine': 3, 'Karacadag': 4}
```

```
if output_class == 0:
    print('Arborio')
elif output_class == 1:
    print('Basmati')
```