LAMPIRAN

Lampiran 1 Coding

Lampiran 1. 1 Script Install library

```
!pip install pyyaml h5py
!pip install matplotlib-venn
!pip install kaggle
```

Lampiran 1. 2 Script penggunaan library

```
import numpy as np
import pandas as pd
import os
import cv2
import matplotlib.pyplot as plt
import tensorflow as tf
import zipfile
from zipfile import *
from tqdm import tqdm
from random import shuffle
from tensorflow.keras.utils  import to_categorical
print("Tensorflow"+" "+ tf.__version__)
!python --version
```

Lampiran 1. 3 Script upload data Json

```
from google.colab import files

uploaded = files.upload()

for fn in uploaded.keys():
  print('User uploaded file "{name}" with length {length} bytes
'.format(
      name=fn, length=len(uploaded[fn])))

# Then move kaggle.json into the folder where the API expects t
o find it.
!mkdir -
p ~/.kaggle/ && mv kaggle.json ~/.kaggle/ && chmod 600 ~/.kaggl
e/kaggle.json
```

Lampiran 1. 4 Script download dataset from kaggle

```
!kaggle datasets download -d syifanuraini/fruitoranges
```

```
local_zip = 'fruitoranges.zip'
zip_ref = zipfile.ZipFile(local_zip, 'r')
zip_ref.extractall('/content')
zip_ref.close()
```

Lampiran 1. 5 Script import library

```
from tensorflow import keras
from tensorflow.keras.preprocessing.image import ImageDataGener
ator
from tensorflow.keras.preprocessing import image
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import GlobalAveragePooling2D, Den
se, BatchNormalization, Dropout, Flatten, Conv2D, MaxPooling2D
from tensorflow.keras.optimizers import Adam
import numpy as np
import random
import matplotlib.pyplot as plt
%matplotlib inline
```

Lampiran 1. 6 Script reshape image dan menempatkan direktori file yang sesuai

```
im_shape = (100,100)

TRAINING_DIR = '/content/fruitoranges/dataset/train'
TEST_DIR = '/content/fruitoranges/dataset/test'

seed = 10

BATCH_SIZE = 16
```

Lampiran 1. 7 Script untuk preprocessing

```
data_generator = ImageDataGenerator(rescale=1./255, validation_
split=0.2)
val_data_generator = ImageDataGenerator(rescale=1./255, validat
ion_split=0.2)
data_generator = ImageDataGenerator(
        validation_split=0.2,
        rotation_range=20,
        width_shift_range=0.2,
        height_shift_range=0.2,
        rescale=1./255,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True,
        fill_mode='nearest')
```

```
val_data_generator = ImageDataGenerator(rescale=1./255, validat
ion_split=0.2)
# Generator for train
train_generator = data_generator.flow_from_directory(TRAINING_D
IR, target_size=im_shape, shuffle=False, seed=seed,
                                                     class_mode
='categorical', batch_size=BATCH_SIZE, subset="training")
# Generator for validation
validation_generator = val_data_generator.flow_from_directory(T
RAINING_DIR, target_size=im_shape, shuffle=False, seed=seed,
                                                     class_mode
='categorical', batch_size=BATCH_SIZE, subset="validation")

# Generator for dataset
test_generator = ImageDataGenerator(rescale=1./255)
test_generator = test_generator.flow_from_directory(TEST_DIR, t
arget_size=im_shape, shuffle=False, seed=seed,
                                                     class_mode
='categorical', batch_size=BATCH_SIZE)

nb_train_samples = train_generator.samples
nb_validation_samples = validation_generator.samples
nb_test_samples = test_generator.samples
classes = list(train_generator.class_indices.keys())
print('Classes: '+str(classes))
plt.figure(figsize=(15,15))
for i in range(9):
    plt.subplot(330 + 1 + i)
    batch = train_generator.next()[0]*255
    image = batch[0].astype('uint8')
    plt.imshow(image)
plt.show()
```

Lampiran 1. 8 Script Training model

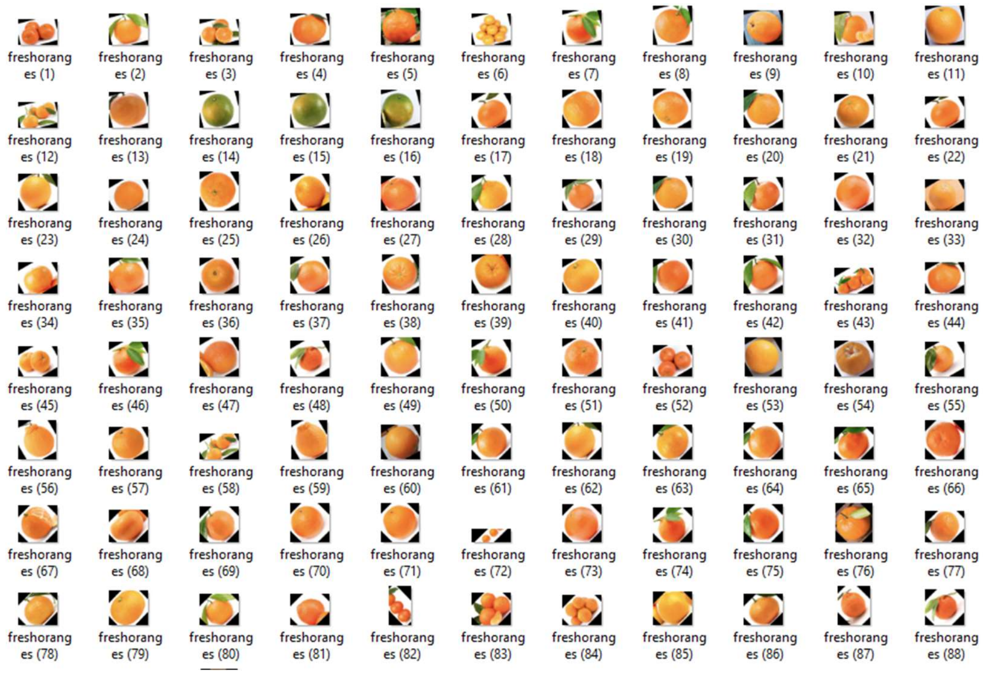
```
model = Sequential()
model.add(Conv2D(20, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=(im_shape[0],im_shape[1],3)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(40, kernel_size=(3,3), activation='relu'))
model.add(Flatten())
model.add(Dense(100, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
model.summary()

# Compile model
model.compile(loss='categorical_crossentropy',
              optimizer=Adam(),
              metrics=['accuracy'])


epochs = 50

#Callback to save the best model
callbacks_list = [
    keras.callbacks.ModelCheckpoint(
        filepath='model.h5',
        monitor='val_loss', save_best_only=True, verbose=1),
   keras.callbacks.EarlyStopping(monitor='val_loss', patience=1
0,verbose=1)
]

#Training
history = model.fit(
        train_generator,
        steps_per_epoch=nb_train_samples // BATCH_SIZE,
        epochs=epochs,
        callbacks = callbacks_list,
        validation_data=validation_generator,
        verbose = 1,
        validation_steps=nb_validation_samples // BATCH_SIZE)
```

Lampiran 1. 9 Script menampilkan hasil training

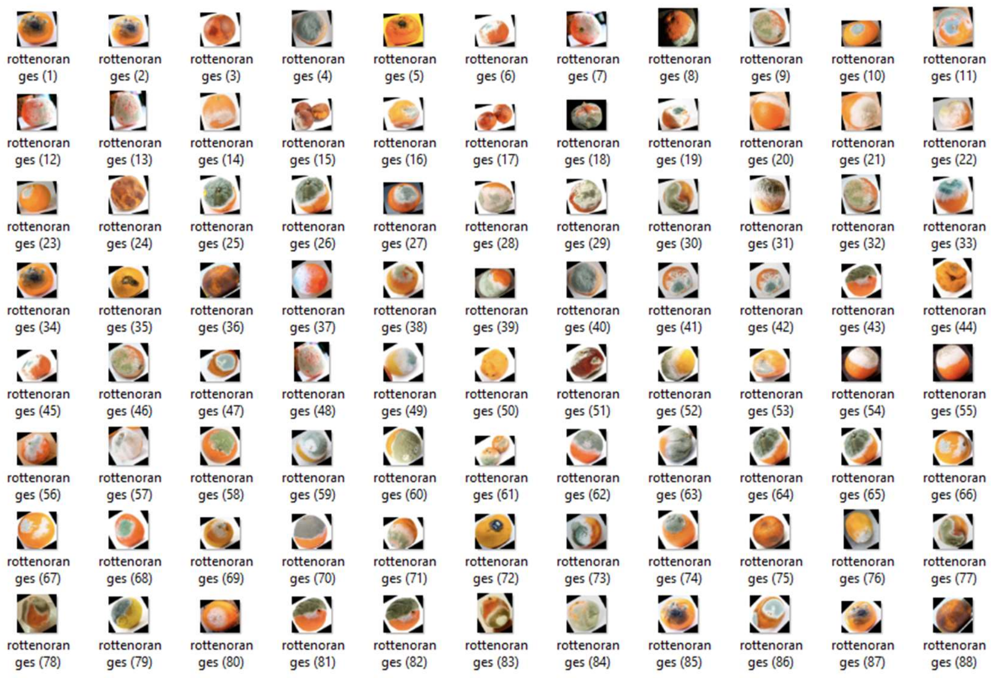
```
import matplotlib.pyplot as plt

history_dict = history.history
loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']

epochs_x = range(1, len(loss_values) + 1)
plt.figure(figsize=(10,10))
plt.subplot(2,1,1)
plt.plot(epochs_x, loss_values, 'bo', label='Training loss')
plt.plot(epochs_x, val_loss_values, 'b', label='Validation loss
')
plt.title('Training and validation Loss and Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.subplot(2,1,2)
acc_values = history_dict['accuracy']
val_acc_values = history_dict['val_accuracy']
plt.plot(epochs_x, acc_values, 'bo', label='Training acc')
plt.plot(epochs_x, val_acc_values, 'b', label='Validation acc')
plt.xlabel('Epochs')
plt.ylabel('Acc')
plt.legend()
plt.show()
```

Lampiran 2 Dataset

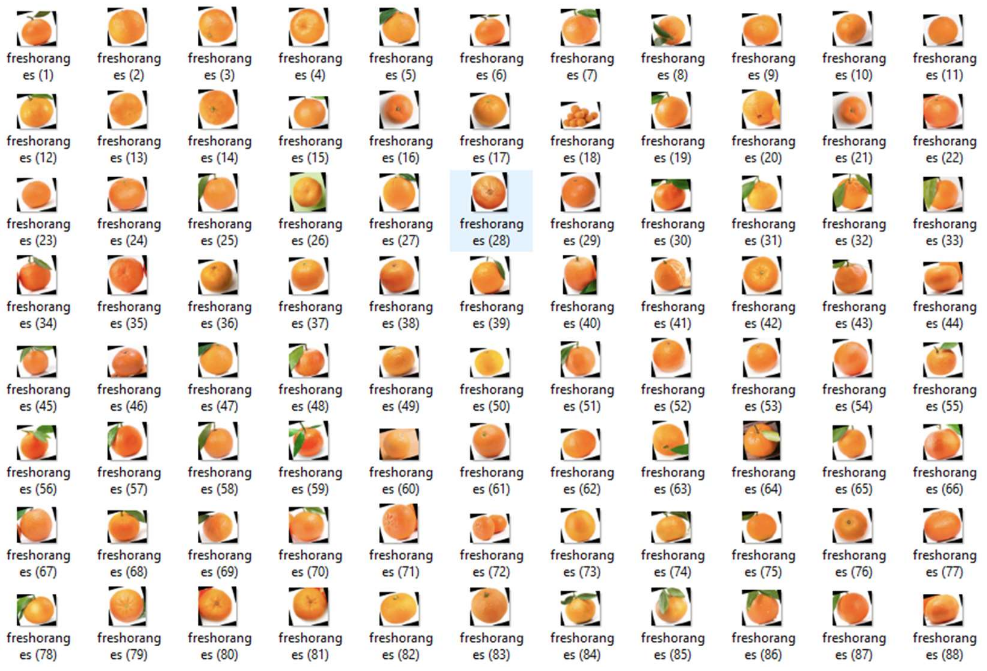
Lampiran 2. 1 Dataset test fruitoranges



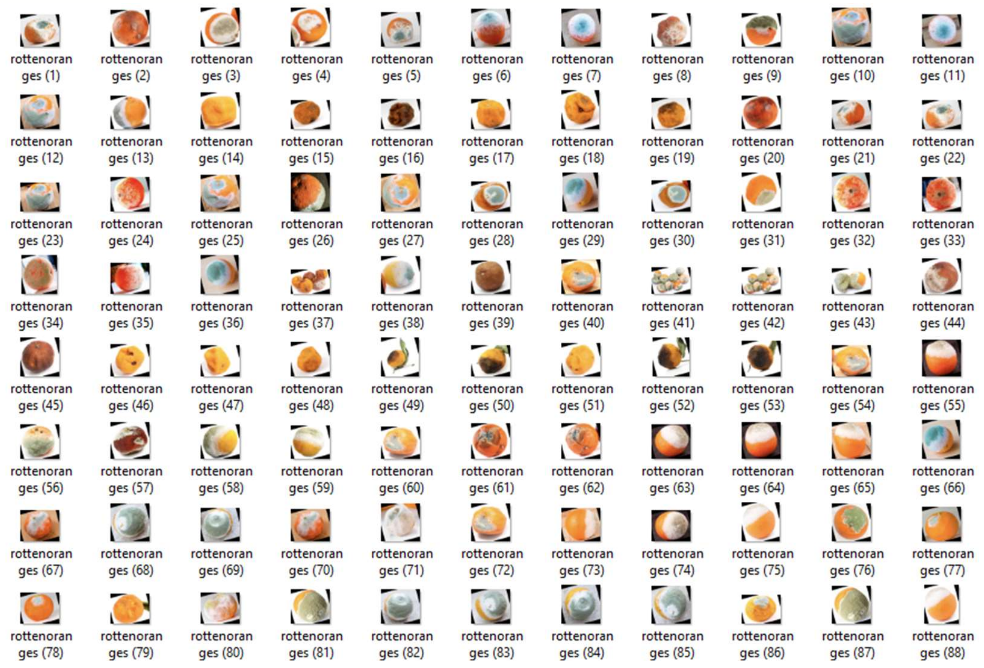
Lampiran 2. 2Dataset test rottenanges



Lampiran 2. 3 Dataset train fruitoranges



Lampiran 2. 4Dataset train rottenoranges


rottenoran ges (1)
rottenoran ges (2)
rottenoran ges (3)
rottenoran ges (4)
rottenoran ges (5)
rottenoran ges (6)
rottenoran ges (7)
rottenoran ges (8)
rottenoran ges (9)
rottenoran ges (10)
rottenoran ges (11)
rottenoran ges (12)
rottenoran ges (13)
rottenoran ges (14)
rottenoran ges (15)
rottenoran ges (16)
rottenoran ges (17)
rottenoran ges (18)
rottenoran ges (19)
rottenoran ges (20)
rottenoran ges (21)
rottenoran ges (22)
rottenoran ges (23)
rottenoran ges (24)
rottenoran ges (25)
rottenoran ges (26)
rottenoran ges (27)
rottenoran ges (28)
rottenoran ges (29)
rottenoran ges (30)
rottenoran ges (31)
rottenoran ges (32)
rottenoran ges (33)
rottenoran ges (34)
rottenoran ges (35)
rottenoran ges (36)
rottenoran ges (37)
rottenoran ges (38)
rottenoran ges (39)
rottenoran ges (40)
rottenoran ges (41)
rottenoran ges (42)
rottenoran ges (43)
rottenoran ges (44)
rottenoran ges (45)
rottenoran ges (46)
rottenoran ges (47)
rottenoran ges (48)
rottenoran ges (49)
rottenoran ges (50)
rottenoran ges (51)
rottenoran ges (52)
rottenoran ges (53)
rottenoran ges (54)
rottenoran ges (55)
rottenoran ges (56)
rottenoran ges (57)
rottenoran ges (58)
rottenoran ges (59)
rottenoran ges (60)
rottenoran ges (61)
rottenoran ges (62)
rottenoran ges (63)
rottenoran ges (64)
rottenoran ges (65)
rottenoran ges (66)
rottenoran ges (67)
rottenoran ges (68)
rottenoran ges (69)
rottenoran ges (70)
rottenoran ges (71)
rottenoran ges (72)
rottenoran ges (73)
rottenoran ges (74)
rottenoran ges (75)
rottenoran ges (76)
rottenoran ges (77)
rottenoran ges (78)
rottenoran ges (79)
rottenoran ges (80)
rottenoran ges (81)
rottenoran ges (82)
rottenoran ges (83)
rottenoran ges (84)
rottenoran ges (85)
rottenoran ges (86)
rottenoran ges (87)
rottenoran ges (88)