UBAYA
UNIVERSITAS SURABAYA

Modul Praktikum

Computer Vision

1604C39G

Program Studi Teknik Informatika
Fakultas Teknik
Universitas Surabaya
2022

Dibuat Oleh:
Mohammad Farid Naufal, S.Kom., M.Kom.
Dr. Joko Siswantoro, S.Si., M.Si.

Daftar Isi

Deskripsi Mata Kuliah

Program Studi : Teknik Informatika

Program : Data Science & Artificial Intelligence

Kode Mata Kuliah : 1604C339G

Nama Mata Kuliah : Computer Vision

Jumlah SKS : 3

Semester : Pilihan

Deskripsi:

Mata kuliah ini memberikan pembekalan kepada mahasiswa jurusan Teknik Informatika mengenai konsep dasar computer vision. Selain itu, mahasiswa juga diajarkan tentang aplikasi computer vision. Materi yang diajarkan pada mata kuliah pengolahan citra digital meliputi pengantar computer vision, formasi citra, pengolahan citra digital, pendeteksian ciri, segmentasi, motion tracking. kalibrasi kamera, 3D vision, recognition. Setelah menyelesaikan mata kuliah ini, mahasiswa diharapkan mampu memahami konsep dasar computer vision dan mampu mengimplementasikannya untuk menyelesaikan masalah riil. Metode pembelajaran yang digunakan adalah dengan mempraktekkan ilmu yang dipelajari dengan membuat program secara langsung di depan komputer. Mahasiswa juga akan mampu memanajemen diri sendiri, mempunyai keterampilan berpikir, dan integritas.

Capaian Pembelajaran Mata Kuliah:

KU	KU1: Mampu berpikir logis, kritis, sistematis dan inovatif dengan menerapkan keilmuan di bidang teknologi informasi dalam pengambilan keputusan dan mampu mendokumentasikan hasil pemikiran secara saintifik KU2: Mampu menunjukkan kinerja yang bermutu dan bertanggungjawab baik secara mandiri ataupun berkelompok, termasuk melakukan supervisi dan evaluasi, serta dapat berkomunikasi dan mengembangkan jaringan kerja dengan berbagai pihak.

KK	KK1: Mampu menganalisis masalah dan merumuskan solusinya melalui penggunaan teknologi informasi dan komunikasi
PP	PP1: Menguasai konsep matematika fundamental dan prinsip ilmu komputer serta bidang ilmu lain yang relevan PP2-DSAI: Menguasai konsep data science dan kecerdasan buatan, serta penerapannya untuk pengembangan perangkat lunak

Modul 1 : Prediksi komposisi alga pada perairan tambak udang menggunakan *computer vision system*

Tujuan Praktikum:

1. Melatih model *deep learning* untuk memprediksi komposisi alga pada perairan tambak menggunakan dataset citra permukaan perairan tambak
2. Mengembangkan *prototype computer vision system* untuk memprediksi komposisi alga pada perairan tambak berbasis model *deep learning* yang sudah dilatih

Alat dan bahan:

1. *Industrial camera*
2. AI *workstation*
3. Dataset citra permukaan perairan tambak udang

Teori dan Definisi:

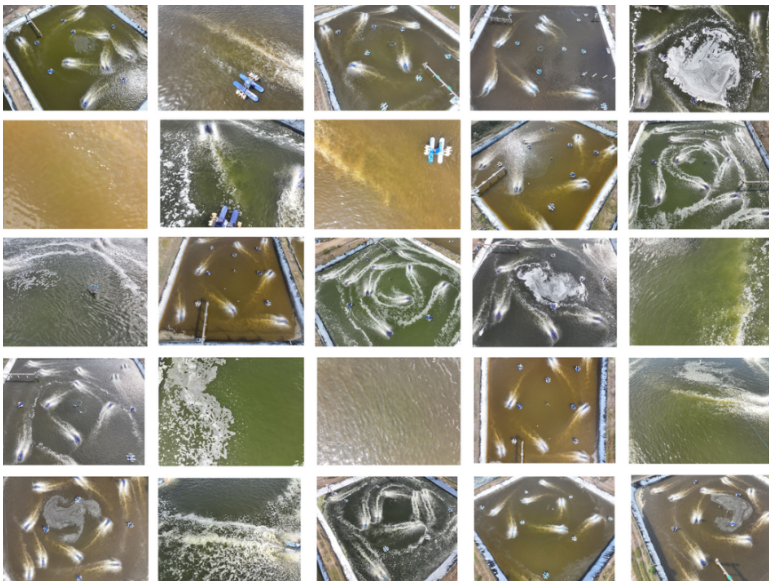
1. Alga pada perairan tambak udang
   Alga merupakan organisme di perairan tambak udang yang memiliki kepekaan tinggi terhadap perubahan parameter fisikokimia dan parameter lingkungan lain. Hal ini menjadikan alga sering digunakan sebagai indikator bagi keberhasilan tambak udang. Pengetahuan tentang jenis dan komposisi komunitas alga di perairan tambak dapat dimanfaatkan untuk modifikasi lingkungan perariran tambak untuk menghindari kerugian. Komposisi alga di perairan tambak juga berpengaruh terhadap warna air di perairan tambak.

   Secara tradisional analisis jenis dan komposisi alga dilakukan dengan mengambil sampel air pada kedalaman tertentu. Sampel kemudian dibawa ke laboratorium untuk diamati menggunakan mikroskop. Jumlah dan jenis alga dihitung dan diklasifikasikan secara manual oleh seorang ahli dari pengamatan sampel di bawah mikroskop. Proses ini sangat memakan waktu dan membosankan. Akurasi klasifikasi jenis plankton secara manual juga masih rendah. Masalah lain yang mungkin terjadi pada perhitungan dan klasifikasi jenis alga secara manual adalah sering terjadi ketidaksepakatan antar ahli dan perbedaan kriteria seorang ahli yang sama pada waktu dan spesies yang berbeda.

2. Dataset Citra Permukaan Tambak

   Dataset citra permukaan perairan tambak udang merupakan dataset citra digital yang terdiri dari citra-citra permukaan air tambak. Citra disimpan dalam format Setiap citra juga dilengkapi dengan hasil analisis komposisi alga yang terkandung di dalamnya dan

disimpan dalam file CSV. Contoh citra permukaan perairan tambak udang dan komposisi alga dapat dilihat pada Gambar 1 dan Gambar 2.



Gambar 1. Contoh citra permukaan perairan tambak udang

Gambar 1. Contoh komposisi alga di perairan tambak udang

3. Industrial Camera

Industrial Camera (Kamera industri) adalah jenis kamera khusus yang disesuaikan untuk bekerja dalam kondisi yang keras (suhu tinggi, tekanan, dan getaran). Mereka digunakan untuk mengontrol siklus produksi, melacak unit pada konveyor, mendeteksi bagian yang sangat kecil, dll. Oleh karena itu, secara umum, cakupannya hampir tidak terbatas. Gambar 2 menunjukkan contoh industrial camera.

Gambar 2.. Industrial Camera

Pada kasus ini Industrial Camera dapat digunakan untuk mengambil citra digital permukaan perairan tambak dan citra mikroskopis alga di dalam perairan tambak udang. Citra digital yang diambil oleh Industrial Camera selanjutnya digunakan sebagai dataset untuk melatih model *deep learning* untuk memprediksi komposisi alga.

4. AI workstation

AI Workstation adalah komputer atau server khusus yang mendukung AI dengan komputasi yang intensif. AI workstation ini menawarkan kinerja yang jauh lebih tinggi dibandingkan dengan workstation tradisional, dengan memanfaatkan beberapa unit Graphical Processing Unit (GPU) spek tinggi. GPU dengan spek tinggi ini bermanfaat saat digunakan untuk melatih model deep learning yang membutuhkan komputasi secara parallel. Gambar 3 menunjukkan AI Workstation buatan Lenovo dengan series P350

Gambar 4. Lenovo P350 AI Workstation

Spesifikasi AI Workstation Lenovo adalah sebagai berikut.

Processor : Intel® Xeon W-1390 (8C / 16T, 2.8 / 5.2GHz, 16MB) Processor Sockets : 1x FCLGA1200 Discrete Graphics : NVIDIA® RTX™ A4000 16GB GDDR6 (4x DP 1.4a) Chipset : Intel W580 Memory : 2 x 16 GB DDR4-3200MHz UDIMM M.2 SSD RAID Controller : Onboard Intel RSTe PCIe M.2 SSD Drive : 512GB SSD M.2 2280 PCIe 4.0 NVMe Opal HDD Controller : Onboard Intel RST SATA RAID SATA/SAS Drive : 1x 1TB HDD 7200rpm 3.5" SATA6Gb/s Optical : 9.0mm DVD±RW Audio Chip : High Definition (HD) Audio, Realtek® ALC623-CG codec Power Supply : 750W Platinum Fixed Onboard Ethernet : Intel I219-LM, 1x GbE RJ-45 WLAN + Bluetooth : Intel AX201 11ax, 2x2 + BT5.2

5. Model *deep learning*

Deep learning dapat diartikan sebagai salah satu teknik dalam machine learning yang mengarahkan sebuah sistem komputer maupun mesin untuk bekerja layaknya manusia secara natural, yakni dengan mempelajari situasi dengan pembelajaran atau pemrograman tertentu.

Deep learning juga merupakan kunci dari pengembangan teknologi yang mengandalkan kecerdasan buatan atau Artificial Intelligence (AI). Dalam deep learning, sebuah komputer akan mempelajari berbagai model dan mengklasifikasikan tugas-tugasnya melalui data yang dikumpulkan. Data tersebut bisa berupa gambar, teks, hingga suara. Bahkan, tingkat akurasinya pun lebih tinggi dalam mengolah data-data berjumlah besar.

6. Transfer learning

Meskipun deep learning dapat membuat komputer bekerja selayaknya manusia,, penggunaan deep learning membutuhkan sejumlah data yang besar. Untuk mengatasi hal tersebut maka dapat digunakan konsep yang disebut transfer learning.

Transfer Learning merupakan teknik yang memanfaatkan model yang sudah ditraining sebelumnya (pretrained model) untuk digunakan mengklasifikasikan dataset yang baru sehingga tidak perlu untuk melakukan training data dari awal dan dilakukan penyesuaian pada bagian akhir dari model.

Beberapa jenis pretrained model *deep learning* yang sering digunakan dalam transfer learning adalah

1. Inception
2. Xception
3. VGG Family
4. ResNet
5. MobileNetV2

Langkah Kerja:

A. Melatih model *deep learning*

1. *Import library* yang diperlukan untuk melatih model *deep learning*, seperti berikut:

```
import tensorflow as tf
from keras.callbacks import EarlyStopping
import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras.utils import to_categorical, load_img, img_to_array
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from tqdm import tqdm
import pickle
from scipy.special import softmax
%matplotlib inline
```

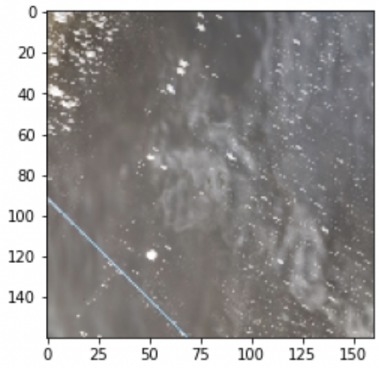
2. *Load* semua citra yang ada di dalam dataset dan lakukan data preprocessing untuk mengubah ukuran citra dan nilai intensitas piksel

```
image = []
for i in tqdm(range(dataset.shape[0])):
    img = load_img('/content/drive/MyDrive/DATASET/' + dataset['Id'][i],
                   target_size=(160,160,3))
    img = img_to_array(img)
    img = img/255
    image.append(img)
X = np.array(image)
```

3. Tampilkan salah satu contoh citra dari dataset.

```
plt.imshow(X[2390])
plt.show()
```

```
<matplotlib.image.AxesImage at 0x7f5700283110>
```



4. Load output target dari setiap citra yang ada di dataset dari file CSV yang disediakan. Kemudian tampilkan lima contoh data teratas.

```
dataset = pd.read_csv('/content/drive/MyDrive/DATASET/LABEL_DATASET.csv')
dataset.head()      # printing first five rows of the file
```

	Id	Green Algae	Blue Green Algae	Chrysophyta	Euglenophyta	Dinoflagellata	Protozoa
0	1.JPG	98.4	0.4	0.8	0	0.4	0
1	2.JPG	98.4	0.4	0.8	0	0.4	0
2	3.JPG	98.4	0.4	0.8	0	0.4	0
3	4.JPG	98.4	0.4	0.8	0	0.4	0
4	5.JPG	98.4	0.4	0.8	0	0.4	0

5. Bagi citra di dalam dataset menjadi data *training* (80%) dan data *testing* (20%)

```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42, test_size=0.2)
```

6. *Load pretrained deep learning model.* Pada kasus ini menggunakan arsitektur MobileNetV2.

```
MobileNetV2 = tf.keras.applications.MobileNetV2(input_shape=(160,160,3), include_top=False, weights='imagenet')
MobileNetV2.trainable = True

# Set untrainable layers (layer 1-60 from 154)
for layer in MobileNetV2.layers[:10]:
  layer.trainable = False
```

7. Modifikasi lapisan *fully connected* dari *pretrained deep learning model* untuk disesuaikan dengan output prediksi komposisi alga. Jumlah label alga sebanyak 6, sehingga layer

output diubah menjadi 6 buah neuron

```
inputs = tf.keras.Input(shape=(160, 160, 3))
x = tf.keras.applications.mobilenet_v2.preprocess_input(inputs)
x = MobileNetV2(x, training=True)
x = tf.keras.layers.GlobalAveragePooling2D()(x)
x = tf.keras.layers.Dropout(0.4)(x)
outputs = tf.keras.layers.Dense(6, activation='relu')(x)

model = tf.keras.Model(inputs, outputs)
```

8. Definisikan *loss function, optimizer,* dan informasi mengenai output dari proses *training*, kemudian tampilan *summary* model.

```
opt = tf.keras.optimizers.Adam(learning_rate=0.001)
model.compile(optimizer=opt,
                loss="mean_absolute_error",metrics=['mean_absolute_error'])

model.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applica
9406464/9406464 [==============================] - 0s 0us/step
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 160, 160, 3)]     0

 tf.math.truediv (TFOpLambda  (None, 160, 160, 3)      0
 )

 tf.math.subtract (TFOpLambd  (None, 160, 160, 3)      0
 a)

 mobilenetv2_1.00_160 (Funct  (None, 5, 5, 1280)       2257984
 ional)

 global_average_pooling2d (G  (None, 1280)             0
 lobalAveragePooling2D)

 dropout (Dropout)           (None, 1280)              0

 dense (Dense)               (None, 6)                 7686

=================================================================
Total params: 2,265,670
Trainable params: 2,228,198
Non-trainable params: 37,472
```

9. Latih model *deep learning* yang telah dirancang menggunakan data *training*

```
epoch = 10
es = EarlyStopping(monitor='val_loss', mode='min', restore_best_weights=True, patience=8)
history = model.fit(
      X_train, y_train,
      epochs=epoch,
      validation_data=(X_test, y_test),
      callbacks=es)
```

```
Epoch 1/10
60/60 [==============================] - 20s 119ms/step - loss: 10.9424 - mean_absolute_error: 10.9424 - val_loss: 5.9198 - val_mean_absolute_error: 5.9198
Epoch 2/10
60/60 [==============================] - 5s 85ms/step - loss: 3.7903 - mean_absolute_error: 3.7903 - val_loss: 2.6869 - val_mean_absolute_error: 2.6869
Epoch 3/10
60/60 [==============================] - 5s 86ms/step - loss: 2.7341 - mean_absolute_error: 2.7341 - val_loss: 2.4995 - val_mean_absolute_error: 2.4995
Epoch 4/10
60/60 [==============================] - 5s 86ms/step - loss: 2.5005 - mean_absolute_error: 2.5005 - val_loss: 2.4970 - val_mean_absolute_error: 2.4970
Epoch 5/10
60/60 [==============================] - 5s 86ms/step - loss: 2.4399 - mean_absolute_error: 2.4399 - val_loss: 2.3555 - val_mean_absolute_error: 2.3555
Epoch 6/10
60/60 [==============================] - 5s 86ms/step - loss: 2.3309 - mean_absolute_error: 2.3309 - val_loss: 2.3987 - val_mean_absolute_error: 2.3987
Epoch 7/10
60/60 [==============================] - 5s 87ms/step - loss: 2.2144 - mean_absolute_error: 2.2144 - val_loss: 2.1686 - val_mean_absolute_error: 2.1686
Epoch 8/10
60/60 [==============================] - 5s 85ms/step - loss: 2.1309 - mean_absolute_error: 2.1309 - val_loss: 2.1995 - val_mean_absolute_error: 2.1995
Epoch 9/10
60/60 [==============================] - 5s 88ms/step - loss: 2.0957 - mean_absolute_error: 2.0957 - val_loss: 1.9475 - val_mean_absolute_error: 1.9475
Epoch 10/10
60/60 [==============================] - 5s 87ms/step - loss: 1.9081 - mean_absolute_error: 1.9081 - val_loss: 1.9469 - val_mean_absolute_error: 1.9469
```

10. Evaluasi model *deep learning* yang telah dilatih menggunakan data *testing*

```
results = model.evaluate(x_test, y_test)
print("mae", results)
```

11. Simpan model *deep learning* ke dalam file

```
model.save('predictor.h5')
```

12. Buat function untuk menangkap citra dari kamera dengan cara:
    - Klik Code snippest panel (simbol < >) pada Google Colab
    - Klik Camera Capture
    - Klik Insert
13. Tangkap citra alga yang diletakkan di bawah *industrial camera*, kemudian klasifikasi menggunakan model yang telah dilatih

```
from IPython.display import Image
try:
  filename = takePhoto()
  print('Saved to {}'.format(filename))
  display(Image(filename))
  img=cv2.imread(filename)
  algaClass=classify(img)
  print('The captured fruit is', fruitClass)
except Exception as err:
  print(str(err))
```

Modul 2 : Klasifikasi buah-buahan Indonesia menggunakan *computer vision system*

Tujuan Praktikum:

1. Melakukan proses segmentasi citra untuk memisahakan objek buah-buahan dari latar belakang.
2. Melakukan ekstraksi fitur warna statistik dari citra buah-buahan
3. Melatih model *machine learning* untuk mengklasifikasikan citra buah-buahan berdasarkan fitur warna statistik.
4. Mengembangkan *prototype computer vision system* untuk mengklasifikasikan buah-buahan menggunakan model *machine learning* yang sudah dilatih

Alat dan bahan:

1. *Industrial camera*
2. AI *workstation*
3. Dataset citra buah-buahan

Teori dan Definisi:

1. Klasifikasi buah-buahan di industri

   Klasifikasi buah-buahan bertujuan untuk mengelompokkan buah-buahan ke dalam salah satu kelas. Di industri agrikultur, klasifikasi buah-buahan dapat diaplikasikan untuk penyortiran, grading, pengukuran, dan penetapan harga. Biasanya klasifikasi bauah-buahan dilakukan secara manual oleh seorang ahli. Namun, klasifikasi manual kadang kala tidak akurat dan hasilnya sulit untuk distandarisasi. Saat ini, banyak metode telah dikembangkan menggunakan computer vision system untuk menggantikan manusia dalam klasifikasi buah-buahan. Langkah umum untuk klasifikasi buah-buahan menggunakan computer vision system meliputi penangkapan citra buah-buahan menggunakan kamera, citra kemudian diproses untuk mengekstrak fitur-fiturnya seperti warna, bentuk, dan tekstur. Buah-buahan kemudian diklasifikasikan menggunakan pengklasifikasi berdasarkan fitur yang telah diekstraksi.

2. Industrial Camera

   Industrial Camera (Kamera industri) adalah jenis kamera khusus yang disesuaikan untuk bekerja dalam kondisi yang keras (suhu tinggi, tekanan, dan getaran). Mereka digunakan untuk mengontrol siklus produksi, melacak unit pada konveyor, mendeteksi bagian yang

sangat kecil, dll. Oleh karena itu, secara umum, cakupannya hampir tidak terbatas. Gambar 2 menunjukkan contoh industrial camera.

Gambar 2.. Industrial Camera

Pada kasus ini Industrial Camera dapat digunakan untuk mengambil citra digital permukaan perairan tambak dan citra mikroskopis alga di dalam perairan tambak udang. Citra digital yang diambil oleh Industrial Camera selanjutnya digunakan sebagai dataset untuk melatih model *deep learning* untuk memprediksi komposisi alga.

3. AI workstation

AI Workstation adalah komputer atau server khusus yang mendukung AI dengan komputasi yang intensif. AI workstation ini menawarkan kinerja yang jauh lebih tinggi dibandingkan dengan workstation tradisional, dengan memanfaatkan beberapa unit Graphical Processing Unit (GPU) spek tinggi. GPU dengan spek tinggi ini bermanfaat saat digunakan untuk melatih model deep learning yang membutuhkan komputasi secara parallel. Gambar 3 menunjukkan AI Workstation buatan Lenovo dengan series P350

Gambar 4. Lenovo P350 AI Workstation

Spesifikasi AI Workstation Lenovo adalah sebagai berikut.

Processor : Intel® Xeon W-1390 (8C / 16T, 2.8 / 5.2GHz, 16MB) Processor Sockets : 1x FCLGA1200 Discrete Graphics : NVIDIA® RTX™ A4000 16GB GDDR6 (4x DP 1.4a) Chipset : Intel W580 Memory : 2 x 16 GB DDR4-3200MHz UDIMM M.2 SSD RAID Controller : Onboard Intel RSTe PCIe M.2 SSD Drive : 512GB SSD M.2 2280 PCIe 4.0 NVMe Opal HDD Controller : Onboard Intel RST SATA RAID SATA/SAS Drive : 1x 1TB HDD 7200rpm 3.5" SATA6Gb/s Optical : 9.0mm DVD±RW Audio Chip : High Definition (HD) Audio, Realtek® ALC623-CG codec Power Supply : 750W Platinum Fixed Onboard Ethernet : Intel I219-LM, 1x GbE RJ-45 WLAN + Bluetooth : Intel AX201 11ax, 2x2 + BT5.2

4. Segmentasi Citra

Dalam pengolahan citra digital dan computer vision, segmentasi citra adalah proses pembagian citra digital ke dalam beberapa bagian (objek). Segmentasi citra bertujuan untuk menyederhanakan penggambaran citra ke dalam bentuk yang lebih bermakna dan lebih mudah dianalisis. Segmentasi citra biasa dipakai untuk menentukan letak objek dan batasannya (garis, kurva, dan lain-lain) dalam citra. Secara khusus, segmentasi citra adalah proses penentuan label tiap piksel sedemikian hingga piksel-piksel berlabel sama memiliki ciri yang sama. Hasil segmentasi citra adalah himpunan segmen yang secara kolektif menutupi seluruh citra. Dalam modul ini, segmentasi digunakan untuk memisahkan objek

buah-buahan dari latar belakangnya. Hasil dari segmentasi adalah citra biner, dengan piksel putih menjatakan objek dan piksel hitam menyatakan latar belakang. Metode yang akan digunakan dalam modul ini adalah *automatic thresholding*. Nilai *threshold T* akan ditentukan secara otomatis menggunakan metode Otsu. Piksel dengan nilai intesitas lebih dari *T* akan dianggap sebagai piksel buah-buahan dan diubah menjadi piksel berwarna putih. Sebaliknya, akan dianggap sebagai piksel latar belakang dan diubah menjadi piksel hitam.

5. Ekstraksi Fitur Warna

   Fitur adalah nilai-nilai numerik yang diekstrak dari citra untuk diinputkan ke pengklasifikasi. Beberapa fitur yang dapat diekstrak dari citra adalah warna, tekstur, bentuk, dan ukuran. Fitur warna merepresentasikan distribusi warna pada citra di ruang warna tertentu. Pada modul ini fitur warna yang digunakan akan diekstrak pada ruang warna HSV (Hue, Saturation, Value). Fitur warna yang akan diekstrak adalah fitur warna statistik yang terdiri dari mean, standard deviation, skewness, dan kurtosis.

6. Klasifikasi menggunakan model *machine learning*

   Machine learning adalah bagian dari kecerdasan buatan yang bertujuan untuk mengajari komputer agar dapat berpikir seperti manusia tanpa diprogram secara eksplisit. Klasifikasi adalah merupakan salah satu algoritma *machine learning* yang diguakan untuk mengelompokkan data ke dalam satu kelas Untuk melatih sebuah model *machine learning* untuk klasifikasi diperlukan data *training* beserta label kelasnya. Dalam modul ini pengklasifikasi yang akan digunakan adalah model *k*-nearest neighbors (k-NN). k-NN adalah model klasifikasi paling sederhana yang menggunakan kemiripan dengan tetangga terdekat untuk mengelompokkan sebuah data baru. k-NN memprediksi kelas dari data yang belum diketahui dengan kelas yang mempunyai proporsi terbesar dari k data tetangga terdekat. k-NN menggunakan jarak untuk mengukur kemiripan antar data. Beberapa jarak yang umum digunakan dalam k-NN adalah jarak Euclidean, jarak Manhattan, dan jarak Minkowski. Nilai k pada k-NN akan ditentukan secara heuristik sehingga diperoleh akurasi terbaik.

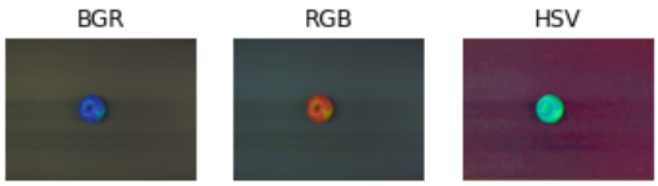
Langkah Kerja:

1. *Import library* yang diperlukan untuk melatih model *machine learning*

```
import cv2
import os
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
import joblib
from IPython.display import display, Javascript
from google.colab.output import eval_js
from base64 import b64decode
```

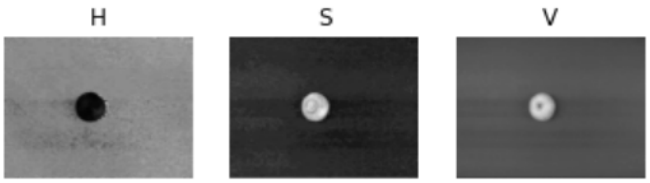
2. Tampilkan salah satu contoh citra di dalam dataset, dalam rauang warna BGR, RGB, dan HSV

```
img=cv2.imread('/content/drive/MyDrive/FruitImages3/apple/ap0001.jpg')
imrgb=cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
imhsv=cv2.cvtColor(img,cv2.COLOR_BGR2HSV)
plt.subplot(1,3,1)
plt.imshow(img)
plt.title('BGR')
plt.axis(False)
plt.subplot(1,3,2)
plt.imshow(imrgb)
plt.title('RGB')
plt.axis(False)
plt.subplot(1,3,3)
plt.imshow(imhsv)
plt.title('HSV')
plt.axis(False)
plt.show()
```



3. Tampilkan juga citra pada setiap komponen H, S, dan V.

```
h,s,v=cv2.split(imhsv)
plt.figure()
plt.subplot(1,3,1)
plt.imshow(h,cmap='gray')
plt.title('H')
plt.axis(False)
plt.subplot(1,3,2)
plt.imshow(s,cmap='gray')
plt.title('S')
plt.axis(False)
plt.subplot(1,3,3)
plt.imshow(v,cmap='gray')
plt.title('V')
plt.axis(False)
plt.show()
```

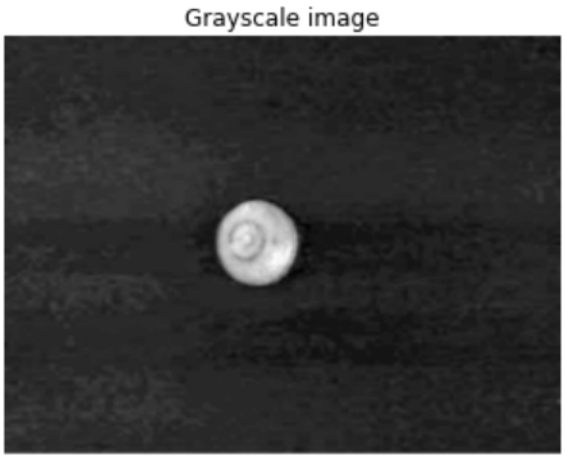


4. Buat function untuk melakukan prapemrosesan citra

```
def preprocessing(img):
  imhsv=cv2.cvtColor(img,cv2.COLOR_BGR2HSV)
  h,s,v=cv2.split(imhsv)
  imgray=s
  imgray=cv2.normalize(imgray,None,0,255,cv2.NORM_MINMAX)
  imgray=cv2.GaussianBlur(imgray,(5,5),0).astype('uint8')
  return imgray
```

5. Tampilkan contoh hasil prapemrosesan citra

```
imgray=preprocessing(img)
plt.figure()
plt.imshow(imgray,cmap='gray')
plt.title('Grayscale image')
plt.axis(False)
plt.show()
```

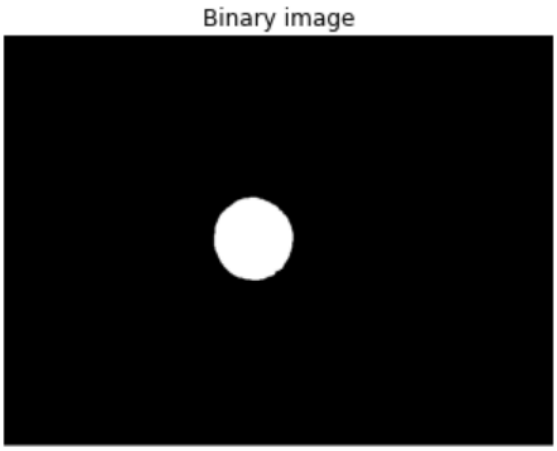


6. Buat function untuk melakukan segmentasi citra

```
def segmentation(img):
  imgray=preprocessing(img)
  ret, imbin = cv2.threshold(imgray,0,255,
                             cv2.THRESH_BINARY+cv2.THRESH_OTSU)
  return imbin
```

7. Tampilkan contoh hasil segmentasi citra

```
imbin=segmentation(img)
plt.figure()
plt.imshow(imbin,cmap='gray')
plt.title('Binary image')
plt.axis(False)
plt.show()
```



8. Buat function untuk mengekstrak fitur warna statistik

```
def extractFeatures(img):
  features=[]
  imgray=preprocessing(img)
  imbin=segmentation(img)
  mean,stdev=cv2.meanStdDev(imgray,imbin)
  s=cv2.subtract(imgray,mean,imbin)
  skew=cv2.pow(s,3)
  skew=cv2.sumElems(skew)[0]/cv2.countNonZero(imbin)/stdev**3
  kurt=cv2.pow(s,4)
  kurt=cv2.sumElems(kurt)[0]/cv2.countNonZero(imbin)/stdev**4
  features.append(mean[0,0])
  features.append(stdev[0,0])
  features.append(skew[0,0])
  features.append(kurt[0,0])
  return features
```

9. Tampilkan contoh hasil ekstraksi fitur

```
extractFeatures(img)
```

```
[46.53034830729167,
 23.467859854868408,
 0.013240144274437735,
 0.0006555486917292528]
```

10. *Load* semua citra beserta label kelasnya yang ada di dalam dataset kemudian ekstrak semua fitur dari setiap citra dan simpan dalam array.

```
drive='/content/drive/MyDrive/FruitImages3'
features=[]
target=[]
label=0
for folder in os.listdir(drive):
  for file in os.listdir(drive+'/'+folder):
    path=drive+'/'+folder+'/'+file
    img=cv2.imread(path)
    features.append(extractFeatures(img))
    target.append(label)
  print(label,folder)
  label=label+1
```

```
features=np.array(features)
target=np.array(target)
```

```
0 tomato
1 mango
2 apple
```

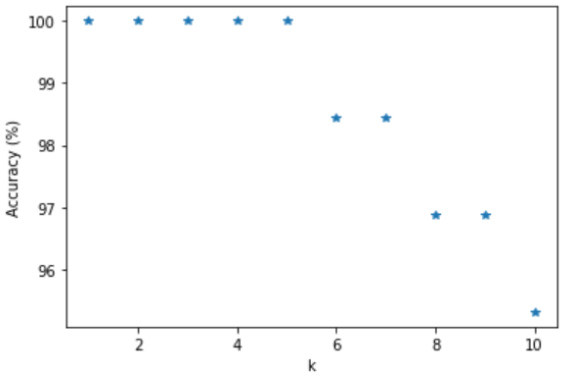
11. Bagi semua fitur dan label kelas menjadi data *training* (60%) dan data *testing* (40%), kemudian tampilkan ukuran data *training* dan data *testing*.

```
xtrain,xtest,ytrain,ytest=train_test_split(features,target,test_size=.4,random_state=1)
print(xtrain.shape)
print(ytrain.shape)
print(xtest.shape)
print(ytest.shape)
```

```
(96, 4)
(96,)
(64, 4)
(64,)
```

12. Latih model k-NN menggunakan data *training* dengan beberapa nilai k, kemudian evaluasi menggunakan data *testing* dengan menghitung akurasinya

```
accuracy=[]
for k in range(1,11):
  knn=KNeighborsClassifier(n_neighbors=k)
  knn.fit(xtrain,ytrain)
  ypred=knn.predict(xtest)
  acc=100*accuracy_score(ytest,ypred)
  accuracy.append(acc)
plt.figure()
plt.plot(range(1,11),accuracy,'*')
plt.xlabel('k')
plt.ylabel('Accuracy (%)')
plt.show()
```



13. Simpan model k-NN telah dilatih ke dalam file

```
knn=KNeighborsClassifier(n_neighbors=3)
knn.fit(xtrain,ytrain)
joblib.dump(knn,'/content/drive/MyDrive/Data Set/knn_model.pkl')
```

```
['/content/drive/MyDrive/Data Set/knn_model.pkl']
```

14. Buat function untuk mengklasifikasikan citra buah-buahan menggunakan model k-NN yang sudah dilatih

```
def classify(img):
  features=np.array(extractFeatures(img))
  knn=joblib.load('/content/drive/MyDrive/Data Set/knn_model.pkl')
  output=knn.predict(features.reshape(1,-1))
  labels=['tomato','mango','apple']
  return labels[output[0]]
```

15. Buat function untuk menangkap citra dari kamera dengan cara:
    - Klik Code snippest panel (simbol < >) pada Google Colab
    - Klik Camera Capture
    - Klik Insert
16. Tangkap citra buah yang diletakkan di bawah *industrial camera* dengan latar belakang hitam, kemudian klasifikasi menggunakan k-NN

```
from IPython.display import Image
try:
  filename = takePhoto()
  print('Saved to {}'.format(filename))
  display(Image(filename))
  img=cv2.imread(filename)
  fruitClass=classify(img)
  print('The captured fruit is', fruitClass)
except Exception as err:
  print(str(err))
```