JURNAL RESPONSIF, Vol. 2 No.2 Agustus 2020, pp. 232~242
E-ISSN**:** 2685-6964

DETEKSI JENIS DAN KEMATANGAN PISANG MENGGUNAKAN METODE *EXTREME LEARNING MACHINE*

Ina Najiah[1], Ifani Hariyanti[2]

[1]Universitas Adhirajasa Reswara Sanjaya
e-mail: Inajiyah@ars.ac.id

[2]Universitas Adhirajasa Reswara Sanjaya
e-mail: ifani@ars.ac.id

Abstrak

Kebun Pisang Celak, yang berada di desa Celak Kec. Cililin adalah salah satu tempat yang khusus bercocok tanam buah pisang. Pisang yang terdapat di Kebun Pisang Celak ini beraneka ragam jenis, sebagai buah lokal yang memiliki nilai ekonomi tinggi dan memiliki potensi pasar yang masih terbuka luas, pisang menjadi salah satu komoditas buah-buahan yang dapat diandalkan. Permasalahan yang ditemukan adalah kurang tepatnya dan kurang pengetahuannya karyawan dalam membedakan jenis dan kematangan pisang terutama karyawan baru. Penelitian ini membuat aplikasi deteksi jenis pisang dan kematangan pisang menggunakan metode *Extreme learning machine.* Dataset pada penelitian ini merupakan gambar pisang dengan 9 jenis yaitu pisang ambon, pisang raja, pisang cavendish, pisang kirana, pisang barangan, pisang Nangka, pisang mas dan pisang kapok. Kematangan pisang pada penelitian ini yaitu tingkat mentah, matang dan terlalu matang. Program dibuat menggunakan tensorflow python. CNN diuji dan menghasilkan tingkat akurasi sebesar 89%. Hasil dari penelitian ini yaitu aplikasi berbasis android untuk mendeteksi jenis pisang.

Kata kunci: Deteksi Jenis Pisang, Deteksi Kematangan Pisang, *Extreme Learning Machine.*

Abstract

Celak Banana Plantation, which is in the village of Celak, Kec. Cililin is a place that specializes in planting bananas. The bananas found in the Celak Banana Garden are of various types, as a local fruit that has high economic value and has a market potential that is still wide open, banana is one fruit commodity that can be relied on. The problems found were inaccurate and inadequate knowledge of employees in distinguishing the types and ripeness of bananas, especially new employees. This study made an application to detect banana types and banana ripeness using the Extreme learning machine method. The dataset in this study is a picture of bananas with 9 types, namely Ambon banana, plantain, Cavendish banana, Kirana banana, barangan banana, jackfruit banana, gold banana and kapok banana. The ripeness of bananas in this study were the raw, ripe and overripe levels. The program is created using tensorflow python. CNN was tested and yielded an accuracy rate of 89%. The results of this study are an android-based application to detect types of bananas.

Keywords: *Detection Of Banana Types, Detection Of Ripeness Of Bananas, Extreme Learning Machine.*

Naskah diterima 26 Juli 2020; direvisi 15 Agustus 2020; diterbitkan 31 Agustus 2020

1. Pendahuluan

Meningkatnya kinerja ekspor sektor pertanian, salah satunya didorong oleh peningkatan ekspor subsektor hortikultura, khususnya buah-buahan tahunan. Pisang menjadi salah satu komoditas buah-buahan yang dapat diandalkan. Pisang memiliki nilai ekonomi tinggi dan potensi pasar yang masih terbuka luas. Pengelolaan pisang di Jawa Barat masih terbatas sebagai tanaman pekarangan atau perkebunan rakyat. Kebun pisang Celak yang berada di Celak Kec. Cililin adalah salah satu tempat yang digunakan untuk mengembangkan sektor perkebunan pisang.

Mendapatkan buah lokal seperti pisang yang berkualitas tidak mudah, dibutuhkan ketelitian dan pengetahuan dari produsen/petani, atas dasar tersebut pada penelitian ini penulis mencoba untuk mengklasifikasikan jenis pisang dan level kematangannya dari segi warna pisang, bentuk dan warna bibit, dan batang pisang untuk membantu memudahkan petani dan petugas dalam memanen hasil kebun.

Pisang pada Kebun Pisang Celak ini beraneka ragam jenis dan berbeda pula cara menanamnya. Beragam jenis pisang yang ada di kebun ini pun memiliki bibit yang berbeda. Penyeleksian bibit dilakukan secara manual, setiap bibit pisang mempunyai bentuk dan warna yang berbeda sehingga dalam penyeleksian bibit untuk menentukan jenis pisang membutuhkan waktu yang tidak sebentar (Ramdani, 2020). Guna mempercepat penyeleksian atau penentuan jenis pisang maka dibutuhkan sebuah sistem yang dapat mendeteksi dan mengklasifikasikan jenis pisang mulai dari bibit, buahnya dan bentuk batangnya.

Selain menentukan jenis pisang, Kebun Pisang Celak menentukan tingkat kematangan Pisang masih secara manual, dimana kematangan dari segi warna dan ukuran dari pisang menjadi patokannya. Masalah yang ditemukan adalah jika terdapat pengunjung atau petugas yang belum mengetahui bagaimana menentukan tingkat kematangan pisang dan memilih pisang yang sudah matang saja untuk diambil. Menurut hasil wawancara dengan petani, hal tersebut menjadi permasalahan bagi petani jika petugas salah dalam mendeteksi jenis dan kematangan pisang dan akan menjadi sebuah kerugian. Solusi yang diperlukan adalah membuat sebuah aplikasi, sistem atau alat untuk menentukan kematangan pisang secara otomatis dari warna dan ukuran.

Image detection, Image recognition atau *Image processing* adalah salah satu bidang keilmuan *Artificial Intelligence* dimana *Image detection* dan *Image recognition* tersebut dapat membantu mengklasifikan jenis pisang dilihat dari bentuk bibit, warna bibit dan ukuran bibitnya. Karena bibit pisang berbeda-beda tergantung jenisnya. Bidang ini pula dapat membantu melakukan deteksi kematangan pisang dari warna buah pisang tersebut.

Penelitian sebelumnya yang pernah dilakukan untuk membantu petani dalam mengklasifikasikan jenis pisang berdasarkan warna adalah menggunakan *image detection* dan *image recognition.* Ada beberapa tahap yang dapat dilakukan dalam membuat klasifikasi berdasarkan warna, pertama melakukan segmentasi mengubah citra menjadi berwarna, kemudian ekstraksi ciri, dan yang terakhir adalah klasifikasi dengan menggunakan metode K-NN (K-Nearest neighbor), Support Vector Machine (SVM), dan DecisionTree (DT). Akurasi yang didapatkan sebesar 96,6% (Sabilla, et al., 2020).

Berdasarkan penelitian ini, maka KNN mendapatkan akurasi yang cukup bagus dalam mengklasifikasikan jenis pisang. Diusulkan metode CNN dimana metode ini dapat memberikan akurasi yang cukup pada penelitian lain, misalnya penelitan mengenai Klasifikasi Citra kelapa sawit dengan hasil rata- rata akurasi dari pengujian sistem sebesar 88,78%. Selain itu, hasil yang diperoleh dari penelitian ini menunjukkan bahwa metode *Extreme learning machine* mampu mengklasifikasikan gambar dengan baik meskipun resolusi gambar yang digunakan cukup kecil (Mahmud, Adiwijaya, Faraby, 2019). Penelitian lain yang dilakukan oleh Maulana & Rochmawati (2019) melakukan klasifikasi buah-buahan dan dataset diambil dari dataset Fruit-360. Kelas data yang digunakan yaitu sejumlah 15 kelas dari 111 kelas pada dataset fruit-360. Hasil dari proses learning didapatkan model CNN dengan akurasi 100% dan loss sebesar 0,012.

Proses pengujian model CNN yang mengguakan 45 sampel citra buah didapatkan akurasi sebesar 91,42%. Dapat disimpulkan bahwa metode CNN yang dirancang pada penelitian ini dapat mengklasifikasi citra dengan baik (Pujoseno, 2018). Pada penelitian ini diusulkan metode

CNN karena dari penelitian yang dipaparkan sebelumnya, metode CNN cukup baik dalam bidang *Image detection, Image recognition*, dan *Image processing*. Luaran atau aplikasi yang akan diusulkan dalam mengimplementasikan sistem klasifikasi jenis pisang ini yaitu aplikasi berbasis android. Aplikasi android atau aplikasi *mobile* saat ini banyak dipakai oleh banyak orang dikarenakan penggunaannya dianggap mudah dan *simple* (Sari, 2019), selain itu android dapat digunakan untuk masalah deteksi (Danukusumo, 2017). Bahasa pemrograman yang digunakan dalam pembuatan metode CNN ini adalah Bahasa pemrograman *python* dan *Kivy. Kivy* adalah Bahasa pemrograman *pyhton* untuk android.

2. Metode Penelitian

Deep Learning

Deep learning merupakan salah satu bidang dari *machine learning* yang memanfaatkan jaringan syaraf tiruan untuk implementasi permasalahan dengan dataset yang besar. Teknik *deep learning* memberikan arsitektur yang sangat kuat untuk *supervised learning*, dengan menambahkan lebih banyak lapisan maka model pembelajaran tersebut bisa mewakili data citra berlabel dengan lebih baik (O'Shea & Nash, 2015).

Menurut O'Shea & Nash (2015) Teknik yang terdapat pada *machine learning* untuk menggunakan ekstraksi fitur dari data pelatihan dan algoritma pembelajaran khusus untuk mengklasifikasi citra maupun untuk mengenali suara, masih memiliki beberapa kekurangan baik dalam hal kecepatan dan akurasi. Aplikasi konsep jaringan syaraf tiruan yang dalam (banyak lapisan) dapat ditangguhkan pada algoritma Machine learning yang sudah ada sehingga komputer sekarang bisa belajar dengan kecepatan, akurasi, dan skala yang besar (O'Shea & Nash, 2015)

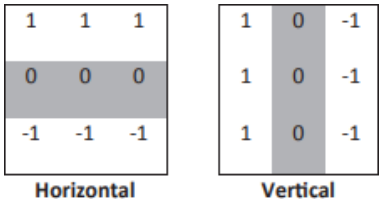
Menurut Pujoseno (2018) *Feature Engineering* adalah salah satu fitur utama dari *Deep learning* untuk mengekstrak pola yang berguna dari data yang akan memudahkan model untuk membedakan kelas*. Feature Engineering* juga merupakan teknik yang paling penting untuk mencapai hasil yang baik pada tugas prediksi. Namun, sulit untuk dipelajari dan dikuasai karena kumpulan data dan jenis data yang berbeda memerlukan pendekatan teknik yang berbeda juga (Pujoseno, 2018).

Algoritma yang digunakan pada *Feature Engineering* dapat menemukan pola umum yang penting untuk membedakan antara kelas *deep learning*, metode CNN atau *extreme learning machine* sangatlah bagus dalam menemukan fitur yang baik pada citra ke lapisan berikutnya untuk membentuk hipotesis nonlinier yang dapat meningkatkan kekompleksitasan sebuah model. 21 Model yang kompleks tentunya akan membutuhkan waktu pelatihan yang lama sehingga di dunia *Deep learning* pengunaan GPU sudah sangatlah umum (Danukusumo, 2017).

Convolution Extreme learning machine

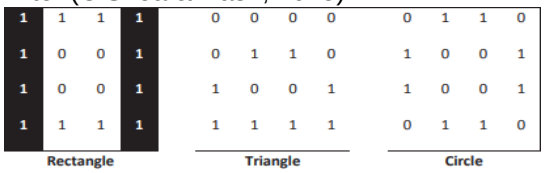
Jaringan saraf konvolusi adalah salah satu struktur jaringan yang representatif dalam pembelajaran mendalam dan telah menjadi hotspot di bidang analisis dan pengenalan gambar (Wicaksono, 2020). Menurut O'Shea & Nash (2015) kerangka ide penting dari jaringan saraf konvolusi adalah perceptron area lokal; berbagi berat; pengambilan sampel spasial. Tiga karakteristik CNN adalah bahwa distorsi data input di ruang sangat kuat. CNN umumnya menggunakan lapisan berbelit-belit dan lapisan sampel secara berurutan mengatur, yaitu, lapisan lapisan berbelit-belit yang terhubung ke lapisan pengambilan sampel, lapisan pengambilan sampel diikuti oleh konvolusi.

Lapisan pertama akan memiliki filter yang mencari tepi horisontal dan filter lain untuk tepi vertikal. Filter ini ditunjukkan pada Gambar 1 sebagai matriks 3×3. Jadi, kita tahu berapa banyak filter untuk digunakan di lapisan conv pertamadan juga apa filter ini. Ukuran 3 × 3 dipilih untuk filter karena ini adalah ukuran yang baik di mana struktur tepi horisontal dan vertikal jelas (O'Shea & Nash, 2015).

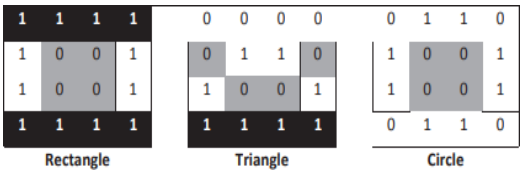


Gambar 1. Filter untuk mengenali tepi horisontal dan vertikal ukuran 3 × 3

Setelah menerapkan filter ini di atas matriks pada Gambar 1, lapisan konvolusi akan dapat mengenali tepi vertikal pada Gambar 2 dan tepi horisontal pada Gambar 3 (Wicaksono, 2020). Llapisan ini mampu mengenali tepi horisontal dan vertikal dalam persegi panjang. Itu juga mengenali tepi horisontal di dasar segitiga, tetapi tidak ada tepi dalam lingkaran. Pada saat ini, CNN memiliki dua kandidat untuk menjadi persegi panjang yang merupakan bentuk yang memiliki setidaknya satu sisi. Meskipun yakin bahwa bentuk ketiga tidak bisa persegi panjang CNN harus menyebarkannya ke lapisan lain sampai membuat keputusan di lapisan terakhir, karena menggunakan dua filter dalam lapisan konv pertama, ini menghasilkan dua output, satu untuk setiap filter (O'Shea & Nash, 2015).



Gambar 2. Tepi vertikal yang diakui berwarna hitam



Gambar 3. Tepi horizontal yang dikenali berwarna hitam

Tensorflow

Nama *Tensorflow* terdiri dari dua kata. Yang pertama adalah tensor yang merupakan unit data yang digunakan TF dalam komputasinya. Kata kedua adalah flow yang mencerminkan bahwa ia menggunakan paradigma aliran data. Akibatnya, TF membangun grafik komputasi yang terdiri dari data yang direpresentasikan sebagai tensor dan operasi yang diterapkan padanya. Untuk membuat hal-hal lebih mudah dipahami, ingatlah bahwa daripada menggunakan variabel dan metode, TF menggunakan tensor dan operasi (Setiawan & Herdianto, 2018). Berikut ini beberapa keuntungan menggunakan dataflow dengan TF:

a. Paralelisme: Lebih mudah untuk mengidentifikasi operasi yang dapat di eksekusi secara pararel
b. Eksekusi Terdistribusi: Program TF dapat dipartisi di beberapa perangkat (CPU, GPU, dan Unit Pemrosesan TF [TPU]). TF sendiri menangani pekerjaan yang diperlukan untuk komunikasi dan kerja sama antar perangkat.
c. Portabilitas: Grafik aliran data adalah representasi kode model yang tidak tergantung bahasa. Grafik aliran data bias dibuat menggunakan Python, disimpan, dan kemudian dipulihkan dalam program C ++ (Setiawan & Herdianto, 2018).

Tahapan Penelitian

Adapun tahapan yang dilakukan pada penelitian ini, seperti yang terlihat pada Gambar 4.

Gambar 4. Tahapan Penelitian

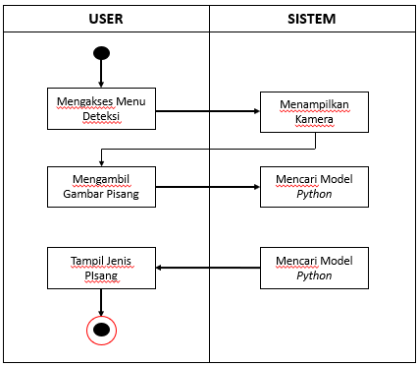
3. Hasil dan Pembahasan

Hasil dan pembahasan dijelaskan dalam tahap-tahap berikut:

3.1. Perancangan dan Desain

a. *Use Case Diagram*

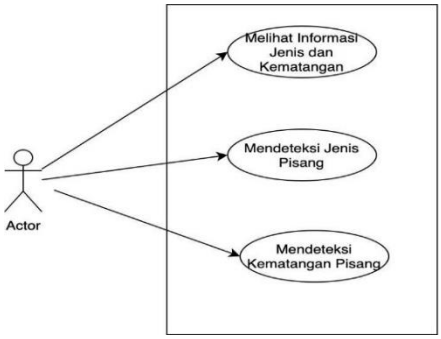
Sebuah usecase diagram menyatakan visualisasi interaksi yang terjadi antara pengguna (aktor) dengan sistem (Kurniawan, 2018). *Use case diagram* yang dibangun dapat dilihat pada gambar berikut.



Gambar 5. Use Case Diagram

b. *Activity Diagram User* melakukan Deteksi Jenis Pisang

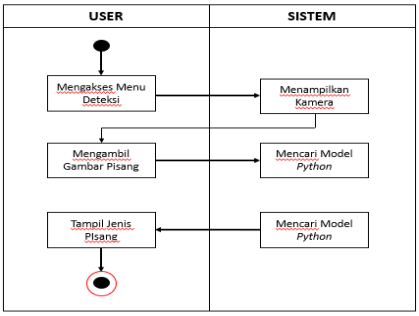
Activity diagram user melakukan deteksi jenis pisang digambarkan pada Gambar 6.



Gambar 6. *Activity Diagram User* melakukan Deteksi Jenis Pisang

c. *Activity Diagram User* melakukan Deteksi Kematangan Pisang

Activity diagram user melakukan deteksi level kematangan pisang digambarkan pada Gambar 7.



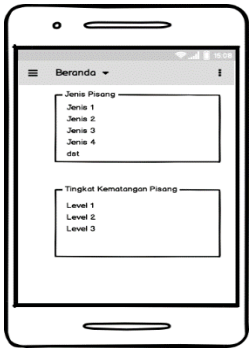
Gambar 7. Activity Diagram User melakukan Deteksi Kematangan Pisang

d. Perancangan Antarmuka

Perancangan antarmuka bertujuan untuk menggambarkan tampilan antarmuka pada sistem yang dibangun. Perancangan antarmuka ini menggambarkan bagaimana interaksi setiap komponen di dalam antarmuka.

1) Antarmuka *Home*

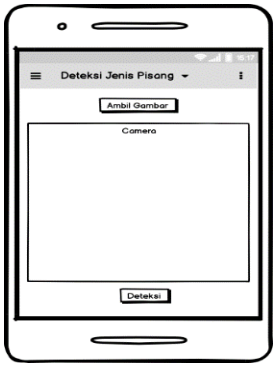
Berikut akan digambarkan bagaimana tampilan dari halaman deteksi jenis pisang yang akan digunakan oleh pengguna. Adapun antarmuka deteksi jenis pisang dapat dillihat pada Gambar 8.



Gambar 8. Antarmuka *Home*

2) Antarmuka Deteksi Jenis Pisang

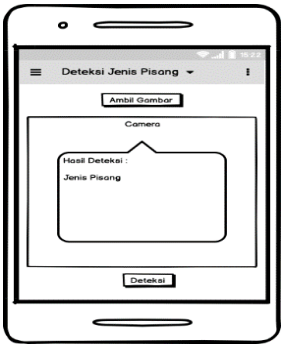
Pada antarmuka ini akan digambarkan bagaimana tampilan dari halaman deteksi jenis pisang yang akan digunakan oleh pengguna. Adapun antarmuka deteksi jenis pisang dapat dilihat pada Gambar 9.



Gambar 9. Antarmuka Deteksi Jenis Pisang

3) Antarmuka Hasil Deteksi Jenis Pisang

Berikut ini akan digambarkan tampilan dari halaman hasil deteksi jenis pisang yang akan digunakan oleh pengguna. Adapun antarmuka hasil deteksi jenis pisang dapat dillihat pada Gambar 10.

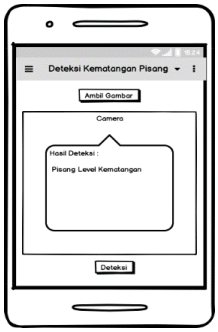


Gambar 10. Antarmuka Hasil Deteksi Jenis Pisang

4) Antarmuka Deteksi Kematangan Pisang
Berikut ini digambarkan bagaimana dari halaman deteksi kematangan pisang yang akan digunakan oleh pengguna, dapat dilihat pada gambar 11.

Gambar 11. Antarmuka Deteksi Kematangan Pisang

5) Antarmuka Hasil Deteksi Kematangan Pisang
Berikut ini akan digambarkan tampilan dari halaman hasil deteksi kematangan pisang yang akan digunakan oleh pengguna. Adapun antarmuka hasil deteksi kematangan pisang dapat dillihat pada Gambar 15.



Gambar 12. Antarmuka Hasil Deteksi Kematangan Pisang

3.2. Implementasi Sistem

Proses implementasi yaitu implementasi model. Model yang dibuat pada penelitian ini yaitu model atau metode *Extreme learning machine* (Salawazo, et al., 2019). Model CNN dirancang dengan Bahasa pemrograman *python* menggunakan *back end* yang bernama *tensorflow*. Berikut merupakan proses implementasi *Extreme learning machine* pada aplikasi deteksi pisang dan deteksi kematangan pisang.

a. Implementasi Input Dataset

Proses input dataset adalah proses pertama pada tahap training (Ferdiana, et al., 2019). Dataset digunakan agar *machine* dapat melakukan pelatihan sebelum dapat mendeteksi jenis dan kematangan pisang secara akurat. Dataset pada penelitian ini dibuat per-folder guna proses training mengenali jenis pisang berdasarkan foldernya. Berikut merupakan susunan folder pada tahap training.

Gambar 13. Susunan Folder Luar

Gambar 14. Susunan Folder Jenis Pisang

Gambar 15. Susunan Folder Kematangan

b. Implementasi Program

1) *File Training*

```
from
  future  import division

import os
import cv2
import numpy as np
import sys
import  pickle
from  optparse  import  OptionParser
import  time
from  keras_frcnn  import  config
from  keras  import  backend  as  K
from  keras.layers  import  Input
from  keras.models  import  Model
from  keras_frcnn  import  roi_helpers
sys.setrecursionlimit(40000)
parser  =  OptionParser()
parser.add_option("-p",                "--path",
dest="test_path",  help="Path  to test  data.")
parser.add_option("-n",  "--num_rois",
        type="int",
dest="num_rois",
help="Number  of  ROIs  per  iteration.
Higher  means  more  memory  use.",
```

```
default=32)
parser.add_option("--config_filename",
        dest="config_filename",
help=
"Location to read the
        metadata
related to the training (generated when
training).",
default="config.pickle")
parser.add_option("--network",
        dest="network", help="Base network
to use. Supports vgg or resnet50.",
default='resnet50')
(options, args) = parser.parse_args()
if not options.test_path: # if
filename is not given
parser.error('Error: path to test data must
be specified.
Pass --path to command line')
config_output_filename =
options.config_filename
with open(config_output_filename, 'rb') as
f_in:
C = pickle.load(f_in)
if C.network == 'resnet50':
import keras_frcnn.resnet as nn
elif C.network == 'vgg':
import keras_frcnn.vgg as nn
# turn off any data augmentation at test
time
C.use_horizontal_flips = False
C.use_vertical_flips = False
C.rot_90 = False
img_path = options.test_path
def format_img_size(img, C):
""" formats the image size based on
config """
img_min_side = float(C.im_size)
(height,width,_) = img.shape
if width <= height:
ratio = img_min_side/width
new_height = int(ratio * height)
new_width = int(img_min_side)
else:
ratio = img_min_side/height
new_width = int(ratio * width)
new_height = int(img_min_side)
img = cv2.resize(img, (new_width,
        new_height),
interpolation=cv2.INTER_CUBIC)
return img, ratio
def format_img_channels(img, C):
""" formats the image channels based on
config """
img = img[:, :, (2, 1, 0)]
img = img.astype(np.float32)
img[:, :, 0] -= C.img_channel_mean[0]
img[:, :, 1] -= C.img_channel_mean[1]
img[:, :, 2] -= C.img_channel_mean[2]
img /= C.img_scaling_factor
img = np.transpose(img, (2, 0, 1))
img = np.expand_dims(img, axis=0)
return img

def format_img(img, C):
""" formats an image for model prediction
based on config
"""
img, ratio = format_img_size(img, C)
img = format_img_channels(img, C)
return img, ratio
# Method to transform the coordinates of
the bounding box to its original size
def get_real_coordinates(ratio, x1, y1, x2,
y2):
real_x1 = int(round(x1 // ratio))
real_y1 = int(round(y1 // ratio))
real_x2 = int(round(x2 // ratio))
real_y2 = int(round(y2 // ratio))
return (real_x1, real_y1, real_x2 ,real_y2)
class_mapping = C.class_mapping
if 'bg' not in class_mapping:
class_mapping['bg'] = len(class_mapping)
class_mapping = {v: k for k, v in
class_mapping.items()}
print(class_mapping)
class_to_color = {class_mapping[v]:
np.random.randint(0, 255, 3)
for v in class_mapping}
C.num_rois = int(options.num_rois)
if C.network == 'resnet50':
num_features = 1024
elif C.network == 'vgg':
num_features = 512
if K.image_dim_ordering() == 'th':
input_shape_img = (3, None, None)
input_shape_features = (num_features,
None, None)
else:
input_shape_img = (None, None, 3)
input_shape_features = (None, None,
num_features)
img_input = Input(shape=input_shape_img)
roi_input = Input(shape=(C.num_rois, 4))
feature_map_input =
Input(shape=input_shape_features)
# define the base network (resnet here,
can be VGG, Inception, etc)
shared_layers = nn.nn_base(img_input,
trainable=True)
# define the RPN, built on the base
layers
num_anchors = len(C.anchor_box_scales)
* len(C.anchor_box_ratios)
```

```
rpn_layers = nn.rpn(shared_layers, num_anchors)
classifier = nn.classifier(feature_map_input, roi_input, C.num_rois, nb_classes=len(class_mapping), trainable=True)
model_rpn = Model(img_input, rpn_layers)
model_classifier_only = Model([feature_map_input, roi_input], classifier)
model_classifier= Model([feature_map_input, roi_input], classifier)
print('Loading weights from {}'.format(C.model_path))
model_rpn.load_weights(C.model_path, by_name=True)
model_classifier.load_weights(C.model_path, by_name=True)
model_rpn.compile(optimizer='sgd', loss='mse')
model_classifier.compile(optimizer='sgd', loss='mse')
all_imgs = []
classes = {}
bbox_threshold = 0.8
visualise = True
for idx, img_name in enumerate(sorted(os.listdir(img_path))):
if not img_name.lower().endswith(('.bmp', '.jpeg', '.jpg', '.png', '.tif', '.tiff')):
continue
print(img_name)
st = time.time()
filepath = os.path.join(img_path,img_name)
img = cv2.imread(filepath)
X, ratio = format_img(img, C)
if K.image_dim_ordering() == 'tf':
X = np.transpose(X, (0, 2, 3, 1))
# get the feature maps and output from the RPN
[Y1, Y2, F] = model_rpn.predict(X)
R = roi_helpers.rpn_to_roi(Y1, Y2, C, K.image_dim_ordering(), overlap_thresh=0.7)
# convert from (x1,y1,x2,y2) to (x,y,w,h)
R[:, 2] -= R[:, 0]
R[:, 3] -= R[:, 1]
# apply the spatial pyramid pooling to the proposed regions
bboxes = {}
probs = {}
for jk in range(R.shape[0]//C.num_rois + 1):
ROIs = np.expand_dims(R[C.num_rois*jk:C.num_rois*(jk+1), :], axis=0)
if ROIs.shape[1] == 0:
break
if jk == R.shape[0]//C.num_rois:
#pad R
curr_shape = ROIs.shape
target_shape = (curr_shape[0],C.num_rois,curr_shape[2])
ROIs_padded = np.zeros(target_shape).astype(ROIs.dtype)
ROIs_padded[:, :curr_shape[1], :] = ROIs
ROIs_padded[0, curr_shape[1]:, :] = ROIs[0, 0, :]
ROIs = ROIs_padded
[P_cls, P_regr] = model_classifier_only.predict([F, ROIs])
for ii in range(P_cls.shape[1]):

if np.max(P_cls[0, ii, :]) < bbox_threshold or np.argmax(P_cls[0, ii, :]) == (P_cls.shape[2] - 1):
continue
cls_name = class_mapping[np.argmax(P_cls[0, ii, :])]
if cls_name not in bboxes:
bboxes[cls_name] = []
probs[cls_name] = []
(x, y, w, h) = ROIs[0, ii, :]
cls_num = np.argmax(P_cls[0, ii, :])
try:
(tx, ty, tw, th) = P_regr[0, ii, 4*cls_num:4*(cls_num+1)]
tx /= C.classifier_regr_std[0]
ty /= C.classifier_regr_std[1]
tw /= C.classifier_regr_std[2]
th /= C.classifier_regr_std[3]
x, y, w, h = roi_helpers.apply_regr(x, y, w, h, tx, ty, tw, th)
except:
pass
bboxes[cls_name].append([C.rpn_stride*x, C.rpn_stride*y, C.rpn_stride*(x+w), C.rpn_stride*(y+h)])
probs[cls_name].append(np.max(P_cls[0, ii, :]))
all_dets = []
for key in bboxes:
bbox = np.array(bboxes[key])
new_boxes, new_probs = roi_helpers.non_max_suppression_fast(bbox, np.array(probs[key]), overlap_thresh=0.5)
for jk in range(new_boxes.shape[0]):
(x1, y1, x2, y2) = new_boxes[jk,:]
(real_x1, real_y1,real_x2,real_y2)
```

```
        = get_real_coordinates(ratio, x1, y1, x2, y2)
cv2.rectangle(img,(real_x1, real_y1), (real_x2, real_y2),
        (int(class_to_color[key][0]), int(class_to_color[key][1]), int(class_to_color[key][2])),2)
textLabel = '{}: {}'.format(key,int(100*new_probs[jk]))
all_dets.append((key,100*new_probs[jk]))
(retval,baseLine) = cv2.getTextSize(textLabel,cv2.FONT_HERSHEY_COMPLEX,1,1)
textOrg = (real_x1, real_y1-0)
cv2.rectangle(img, (textOrg[0] - 5, textOrg[1]+baseLine - 5), (textOrg[0]+retval[0] + 5, textOrg[1]-retval[1] - 5), (0, 0, 0), 2)
cv2.rectangle(img, (textOrg[0] - 5,textOrg[1]+baseLine - 5), (textOrg[0]+retval[0] + 5, textOrg[1]-retval[1] - 5), (255, 255, 255), -1)
cv2.putText(img, textLabel, textOrg, cv2.FONT_HERSHEY_DUPLEX, 1, (0, 0, 0), 1)

print('Elapsed time = {}'.format(time.time() - st))
print(all_dets)
cv2.imshow('img', img)
cv2.waitKey(0)
```

2) *File Testing*

```
from
  future import division
import os
import cv2
import numpy as np
import sys
import pickle
from optparse import OptionParser
import time
from keras_frcnn import config
from keras import backend as K
from keras.layers import Input
from keras.models import Model
from keras_frcnn import roi_helpers
sys.setrecursionlimit(40000)
parser = OptionParser()
parser.add_option("-p", "--path", dest="test_path", help="Path to test data.")
parser.add_option("-n", "--num_rois", type="int", dest="num_rois",
help="Number of ROIs per iteration. Higher means more memory use.", default=32)
parser.add_option("--config_filename", dest="config_filename", help=
"Location to read the metadata related to the training (generated when training).", default="config.pickle")
parser.add_option("--network", dest="network", help="Base network to use. Supports vgg or resnet50.", default='resnet50')
(options, args) = parser.parse_args()
if not options.test_path: # if filename is not given
parser.error('Error: path to test data must be specified.
Pass --path to command line')
config_output_filename = options.config_filename
with open(config_output_filename, 'rb') as f_in:
C = pickle.load(f_in)
if C.network == 'resnet50':
import keras_frcnn.resnet as nn
elif C.network == 'vgg':
import keras_frcnn.vgg as nn
# turn off any data augmentation at test time
C.use_horizontal_flips = False
C.use_vertical_flips = False
C.rot_90 = False
img_path = options.test_path
def format_img_size(img, C):
""" formats the image size based on config """
img_min_side = float(C.im_size)
(height,width,_) = img.shape
if width <= height:
ratio = img_min_side/width
new_height = int(ratio * height)
new_width = int(img_min_side)
else:
ratio = img_min_side/height
new_width = int(ratio * width)
new_height = int(img_min_side)
img = cv2.resize(img , (new_width, new_height), interpolation=cv2.INTER_CUBIC)
return img, ratio
def format_img_channels(img, C):
""" formats the image channels based on config """
```

c. Implementasi Perangkat Lunak

Perangkat lunak yang digunakan untuk menjalankan aplikasi deteksi pisang adalah sebagai berikut:

Tabel 1. Spesifikasi Perangkat Lunak

No	Perangkat	Spesifikasi
1	Sistem Operasi	Windows 10
2	Browser	Mozilla Firefox, Google Chrome
3	Database Server	-
4	Aplikasi	Anaconda Python 3.7 Android Studio Gradle JDK Jupyter Notebook

d. Implementasi Perangkat Keras

Perangkat keras yang dipakai untuk menjalankan sistem informasi manajemen berbasis web. Spesifikasi perangkat keras yang digunakan untuk menjalankan aplikasi deteksi pisang adalah sebagai berikut:

Tabel 2. Spesifikasi Perangkat Keras

No	Perangkat	Spesifikasi
1	Processor	Speed 2 Ghz
2	Memory	2 GB
3	VGA Card	512 MB
4	Harddisk	1 GB
5	Monitor	14”

3.3. Pengujian Sistem

3.3.1. Metode Pengujian

Metode yang dipakai untuk pengujian pada penelitian ini yaitu metode *blackbox testing.*

3.3.2. Kasus dan Hasil Pengujian

Tabel 3. Kasus dan Hasil Pengujian

No	Skenario	Harapan	Hasil
1	Mengarahkan camera ke arah pisang, klik OK	Menangkap gambar pisang	Sesuai Harapan
2	Memilih button deteksi jenis	Mendeteksi jenis pisang	Sesuai Harapan
3	Memilih button deteksi kematangan	Mendeteksi kematangan pisang	Sesuai Harapan

Berdasarkan pada hasil pengujian aplikasi dengan metode *Blackbox* yang menyatakan bahwa sistem yang dibangun telah mencapai harapan yang diinginkan. Begitu juga dengan pengujian kuesioner yang telah dijabarkan diatas menyatakan bahwa responden memberikan respon positif terhadap sistem yang dibangun.

4. Kesimpulan

Kesimpulan dari penelitian ini sebagai berikut:

a. Penelitian ini berhasil membuat sistem untuk mengklasifikasikan jenis pisang dan level kematangannya dari segi warna pisang, bentuk dan warna yang berdasarkan questioner dapat membantu memudahkan petani pisang dan karyawannya di Kebun Pisang Kabupaten Cililin.
b. Metode CNN untuk mendeteksi jenis Pisang dan level kematangannya pada penelitian ini msenghasilkan akurasi baik sebesar 86% tingkat keakurasiannya.

Referensi

Danukusumo, K.P. (2017). Implementasi *Deep Learning Menggunakan Extreme learning Machine* Untuk Klasifikasi Citra Candi Berbasis GPU, Yogyakarta: Universitas Atma Jaya.

Ferdiana, R., Jatmiko, F., Purwanti, D.D., Ayu, A. S. T., Dicka, W. F. (2019). Dataset Indonesia Untuk Analisis Sentimen. JNTETI*,* vol. 8, no. 4, 334-339.

Kurniawan, T. A. (2018). Pemodelan Use Case (UML) : Evaluasi Terhadap Beberapa Kesalahan Dalam Praktik. *Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK),* vol. 5, pp. 77-86.

Mahmud, K. H., Adiwijaya & Al Faraby, S. (2019). Klasifikasi Citra Multi-Kelas Menggunakan Extreme learning machine. E-*Proceeding of Engineering : Vol.6 ISSN : 2355-9365*.

Maulana, F. F., & Rochmawati, N. (2020). Klasifikasi Citra Buah Menggunakan Convolutional Neural Network. *Journal of Informatics and Computer Science (JINACS)*, *1*(02).

O'Shea, K., & Nash, R. (2015). *An Introduction to Extreme learning machines, Neural and Evolutionary Computing*: Cornel University.

Pujoseno, J. (2018). Implementasi Deep Learning Menggunakan Convolutional Neural Network Untuk Klasifikasi Alat Tulis. Program Studi Statistika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Islam Indonesia

Ramdani, S. (2020). BUDIDAYA PISANG [Online]. Available: https://dinpertan.purbalinggakab.go.id/budidaya-pisang/.

Sabilla, I. A., Wahyuni, C. S., Fatichah, C., & Herumurti, D. (2019). *Determining Banana Types and Ripeness from Image using Machine Learning Methods. International Conference of Artificial Intelligence and Information Technology* (ICAIIT).

Salawazo, V. M. P., Gea, D. P. J., & Azmi, Fadillah. (2019). Implementasi Metode Extreme learning machine (CNN) Pada Pengenalan Objek Video CCTV. *Jurnal Mantik Penusa,* vol. 3, no. 11.

Sari, D. E. (2016). QUIZLET: APLIKASI PEMBELAJARAN BERBASIS SMARTPHONE ERA GENERASI MILENIAL. *Jurnal Pendidikan Ilmu Sosial, Vol 29, No.1, Juni 2019,* pp. 9-15.

Setiawan, E. B., & Herdianto, R. (2018). Penggunaan Smartphone Android sebagai Alat Analisis Kebutuhan Kandungan Nitrogen pada Tanaman Padi. *JNTETI, Vol. 7, No. 3,* ISSN 2301 - 4156*,* pp. 273-280, 2018.

Wicaksono, A. F. (2020). Tutorial Dasar Tensorflow. [Online]. Available: https://ir.cs.ui.ac.id/alfan/tutorial/tf_intro.html.