Jurnal Rekayasa Mekatronika (JRM)
Vol. 1 No. 2, September 2024, hal. 22 – 37 ISSN xxxx-xxxx

SISTEM KLASIFIKASI CITRA UNTUK INSPEKSI KAIN BERBASIS RASPBERRY PI

[1] Alifa Rustiyanti, [2] Emmanuel Agung Nugroho*, [3] Nanang Roni Wibowo
[1,2,3] Teknologi Rekayasa Mekatronika, Politeknik Enjinering Indorama, Purwakarta, Indonesia
e-mail: [1]alifarustiyantii@gmail.com [2]emmanuel.agung@pei.ac.id [3]nanang.roni@pei.ac.id
Co-author: [2]emmanuel.agung@pei.ac.id

Abstrak

Perkembangan teknologi di era revolusi industri 4.0 memberikan dampak positif pada industri tekstil. Salah satu proses produksi tekstil yang memanfaatkan teknologi 4.0 adalah unit pengendalian kualitas produk kain. Salah satu masalah yang sering muncul dalam proses produksi tekstil adalah cacat kain. Penelitian ini mengembangkan Aplikasi Image Classification untuk Inspeksi cacat kain dengan pemodelan citra digital menggunakan Teachable Machine. Tujuan penelitian ini adalah mengurangi kesalahan pencatatan dalam inspeksi kain. Metode yang digunakan dalam penelitian ini adalah dengan menggunakan kamera pendeteksi cacat kain yang terintegrasi dengan Teachable Machine yang telah dilatih untuk mengenali jenis-jenis cacat. Pengujian dilakukan terhadap kecepatan waktu inferensi dan akurasi sistem menggunakan Confusion Matrix Analysis. Hasil pengujian menunjukkan rata-rata data inferensi untuk kategori kain bagus, kain cacat jarang dan kain cacat slap sebesar 143 mS dengan tingkat akurasi mencapai 98%. Performa ini mampu memberikan kontribusi positif pada proses pengawasan kualitas produk tekstil.

Kata kunci: *Image Classification, Pemodelan Citra, Teachable Machine, Inspeksi Kain, Confusion Matrix.*

Abstract

Technological developments in the era of the industrial revolution 4.0 have had a positive impact on the textile industry. One textile production process that utilizes 4.0 technology is the fabric product quality control unit. One problem that often arises in the textile production process is fabric defects. This research develops an Image Classification Application for fabric defect inspection with digital image modeling using a Teachable Machine. The purpose of this research is to reduce recording errors in fabric inspection. The method used in this research is to use a fabric defect detection camera integrated with a Teachable Machine trained to recognize the types of defects. Tests were conducted on the speed of inference time and system accuracy using Confusion Matrix Analysis. The test results show that the average inference data for the categories of good fabric, rare defective fabric and slap defective fabric is 143 mS with an accuracy rate of 98%. This performance can make a positive contribution to the process of monitoring the quality of textile products.

Keywords*: Image Classification, Image Modeling, Teachable Machine, Fabric Inspection, Confusion Matrix*

1. PENDAHULUAN

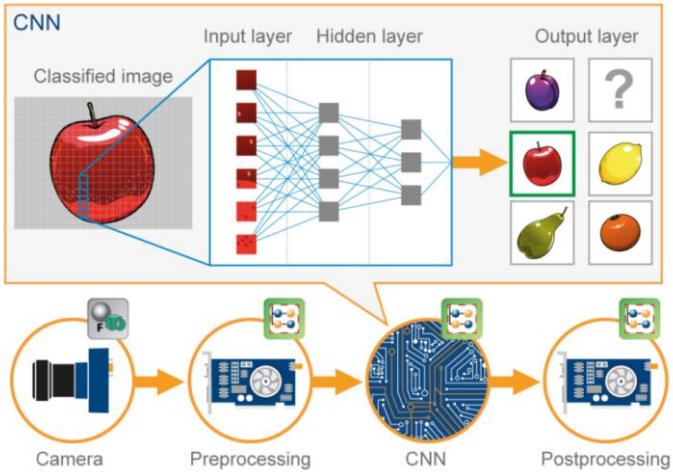
Dikutip dari website Kementerian perindustrian, industri manufaktur tekstil dan pakaian jadi mencatatkan pertumbuhan tertinggi sebesar 15,08 persen pada triwulan III tahun 2019, melampaui pertumbuhan ekonomi 5,02 persen pada periode yang sama. Dalam jurnal yang ditulis oleh Dina Firma Dewanti dan Darminto Pujotomo, dijelaskan bahwa PT. Iskandar Indah *Printing Textile* adalah salah satu perusahaan tekstil yang sedang berkembang di Indonesia. Untuk mencapai kepuasan pelanggan, perusahaan ini selalu berusaha untuk menjaga kualitas barang yang dibuat [1]. Cacat produk mempengaruhi efisiensi produksi perusahaan dan tingkat kepuasan pelanggan. Dalam proses produksi

yang dilakukan secara terus-menerus, seringkali tidak bisa dihindari terjadinya produk cacat. Produk cacat merupakan produk yang dihasilkan dalam proses produksi dimana produk yang dihasilkan tidak sesuai dengan standar mutu yang ditetapkan. Salah satu contoh dalam kecacatan suatu kain yaitu adanya Cacat Jarang (terdapat ruang pada anyaman suatu kain) dan *Slap* (terdapat kotoran pada kain yang salah satunya dapat disebabkan oleh debu, oli mesin, dll). Penyebab dari cacat Jarang, Slap, dan Corak meleset itu sendiri bisa mempengaruhi kualitas kain yang di produksi. Meminimalkan jumlah cacat sangat penting untuk memastikan kualitas produk. Memproduksi produk yang berkualitas adalah wajib untuk mempertahankan dalam pasar kompetitif. Banyak faktor yang sifatnya tidak terkendali yang menyebabkan produk cacat, sehingga perlu adanya perhatian khusus karena mempengaruhi kelancaran operasi dan biaya produksi [2]–[4].

2. TINJAUAN PUSTAKA

2.1 *Machine Learning* untuk *Image Classification*

Machine learning untuk *image classification* adalah penerapan teknik *machine learning* dalam mengenali dan mengklasifikasikan gambar berdasarkan karakteristik atau kategori tertentu. Ini melibatkan pelatihan model dengan dataset gambar yang telah diberi label, sehingga komputer dapat memahami dan mengasosiasikan pola atau fitur dalam gambar dengan kategori tertentu. Setelah pelatihan, model ini dapat digunakan untuk mengklasifikasikan gambar-gambar.



Gambar 1. *Penerapan machine learning pada image classification*[5].

Gambar 1 menjelaskan cara kerja dari machine learning [6]–[11]. Sementara itu, *Image Classification* untuk Inspeksi Kain berbasis *Machine Learning* adalah penerapan serupa, tetapi dengan fokus khusus pada industri tekstil. Dalam hal ini, komputer diajarkan untuk mengenali dan membedakan kain baik dan rusak dengan menggunakan gambar kain yang telah diberi label [12]–[16]. Proses pelatihan melibatkan dataset gambar kain, sehingga model machine learning dapat secara otomatis menganalisis dan memberikan informasi tentang kualitas kain secara efisien. Hal ini berguna dalam memastikan kualitas produk tekstil tanpa perlu inspeksi manual yang mahal dan kurang efisien [17]–[20].

2.2 *OpenCV Python*

Open Source Computer Vision Library adalah sebuah perpustakaan perangkat lunak open-source yang dikembangkan oleh Intel. Perpustakaan ini digunakan untuk pengolahan citra dan pengenalan pola dalam bahasa pemrograman *Python*, serta untuk visi komputer secara real-time. *OpenCV* menyediakan berbagai alat dan fungsi yang memungkinkan komputer untuk melakukan berbagai tugas dalam pengolahan citra dan pemrosesan video. Dengan bantuan *OpenCV*, komputer dapat melakukan tugas seperti pengolahan visual yang mirip dengan kemampuan manusia[16], [21].

OpenCV Python adalah bahasa pemrograman tingkat tinggi yang sangat disukai karena sintaksisnya yang sederhana. Pengembangan perangkat lunak, analisis data, pembuatan situs web, dan kecerdasan buatan adalah beberapa bidang di mana ini digunakan[16].

2.3 *TensorFlow Lite*

TensorFlow Lite adalah versi ringan dari *framework machine learning* yang populer yaitu *TensorFlow*. Dikembangkan oleh *Google*, *TensorFlow Lite* dimaksudkan untuk menjalankan model *machine learning* di perangkat seluler, perangkat bergerak, dan perangkat *Internet of Things* (IoT) yang memiliki sumber daya terbatas.

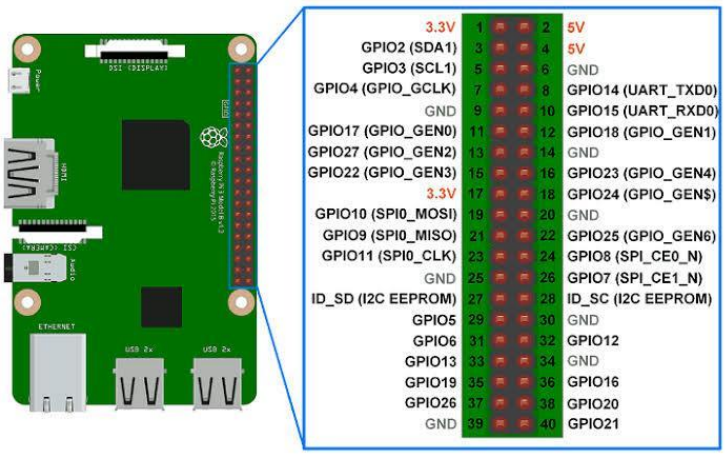
TensorFlow Lite menawarkan berbagai *library* dan alat untuk mengubah model menjadi berfungsi di perangkat seluler atau perangkat embedded. Selain itu, ini mendukung berbagai operasi pembelajaran mesin seperti image classification, deteksi objek, dan pengenalan suara [9], [22]–[24].

2.4 *Raspberry PI*

Raspberry Pi adalah rangkaian komputer mini berukuran kartu kredit yang dikembangkan oleh *Raspberry Pi Foundation*. *Raspberry Pi* awalnya dirancang sebagai alat untuk mendukung pendidikan komputer dan ilmu komputer yang terjangkau, tetapi seiring berjalannya waktu, perangkat ini telah mendapatkan popularitas dalam berbagai bidang termasuk di kalangan *hobbies*, penelitian, pengembangan projek elektronik, dan *Internet of Things*.

Gambar 2. *Raspberry PI*[16].

Gambar 2 menunjukan board fisik dari *Raspberry PI.* Dari segi fungsinya, mirip dengan komputer konvensional. *Raspberry Pi* memiliki input dan output seperti yang dimiliki oleh papan mikrokontroler. Model *Raspberry Pi* 4B adalah perangkat unggulan yang menonjol dengan prosesor yang lebih cepat, kemampuan multimedia, kinerja yang lebih baik, memori yang lebih besar, dan konektivitas yang lebih baik dibandingkan dengan generasi sebelumnya [25]–[27].



Gambar 3. *Pin GPIO raspberry PI* [25].

Gambar 3 menggambarkan *Pin General Purpose Input/Output* (GPIO) yang berjumlah 40 pin pada *Raspberry Pi*. *Pin-pin* ini digunakan untuk berkomunikasi dengan berbagai perangkat eksternal, seperti sensor, motor, LED, dan perangkat elektronik lainnya. *Pin* GPIO dapat dikonfigurasi sebagai input atau output, sehingga memungkinkan untuk membaca data dari perangkat eksternal atau mengendalikan perangkat tersebut [28]–[30].

3. METODE PENELITIAN

Dalam membangun penelitian ini memiliki beberapa tahapan. Setiap tahapan memiliki peran penting dalam memastikan bahwa perancangan dilakukan secara efektif dan efisien, serta dapat menghasilkan *Application Image Classification* yang akurat dan efektif dalam mengenali pola dan trend dalam data gambar. Beberapa tahapannya sebagai berikut:

3.1 Kebutuhan Libraries

Untuk membuat *Application Image Classification* dibutuhkan beberapa *libraries* untuk di-install terlebih dahulu pada Raspberry PI. Berikut adalah *libraries* yang digunakan, sebagaimana ditunjukan pada Tabel 1.

Tabel 1. *Kebutuhan libraries.*

Nama Libraries	Fungsi
os	Digunakan untuk berinteraksi dengan sistem operasi. Ini memungkinkan untuk melakukan operasi seperti mengakses file, direktori, mengatur variabel lingkungan, dan sebagainya.
argparse	Memungkinkan penguraian argumen yang disediakan melalui baris perintah saat menjalankan program.
sys	*Libraries* sys memberikan akses ke variabel dan fungsi yang terkait dengan *interpreter Python.*
time	*Libraries* ini menyediakan berbagai fungsi yang terkait dengan waktu. Dapat menggunakannya untuk mengukur waktu, membuat jeda, atau melakukan operasi yang melibatkan pengaturan waktu.
cv2	*Cv2* atau *OpenCV* adalah pustaka untuk pengolahan *image* dan komputer visi. Digunakan untuk mengambil *image* dari kamera yang terhubung ke perangkat dan menjalankan berbagai operasi pengolahan *image*.
Tensorflow lite	Ini berkaitan dengan pustaka *TFLite (TensorFlow Lite),* yang digunakan untuk menjalankan model *machine learning* yang sudah dilatih pada Raspberry Pi
GPIO	Raspberry Pi *GPIO (General Purpose Input/Output)* memungkinkan pengendalian pin *GPIO* pada Raspberry Pi

Berdasar dari Table 1 maka perlu adanya *instalasi libraries* yang dibutuhkan. Berikut tahapan dalam *instalasi libraries*.

1. *Install zip example tensorflow* yang ada di github

```
pi@raspberrypi:~ $ wget
    https://github.com/tensorflow/examples/archive/refs/heads/master.zip
```

2. Lalu *unzip* file tersebut

```
pi@raspberrypi:~ $ unzip master.zip
```

3. File tersebut memiliki folder Bernama examples-master, masuk ke folder image classification dengan cara berikut

```
pi@raspberrypi:~ $ cd examples-master/lite/examples/image_classification/raspberry_pi
```

4. *Install* semua *libraries* yang dibutuhkan dengan cara menjalankan *file bash* yang ada di folder ini yaitu dengan cara berikut.

```
pi@raspberrypi:~ $ sh setup.sh
```

3.2 Kebutuhan File

Untuk menjalankan sistem *Application Image Classification* juga membutuhkan dua komponen utama lainnya yaitu file model dan label yang sudah kita buat pada tahap *modelling*. File tersebut dimuat atau *import* pada program *Python Application Image Classification* nantinya. Berikut adalah kebutuhan file pada Tabel 2.

Tabel 2. *Kebutuhan file application image classification.*

File	Fungsi
model.tflite	Berkas .tflite adalah format ringkas yang digunakan oleh *TensorFlow Lite* untuk menyimpan model mesin pembelajaran
labels.txt	File labels.txt adalah berkas teks yang berisi daftar label nama kelas yang digunakan
ApplicationImageClassification.py	Ini merupakan file *Application Image Classification*

Berdasar dari Tabel 2 ketiga file tersebut merupakan kebutuhan file untuk menjalankan sistem *Application Image Classification*. Didalam *Application Image Classification* ini ada beberapa sistem yang diintegrasikan, beberapa sistem tersebut yaitu :

1. Meng-*handle* untuk menerima data gambar dari webcam melalui libraries opencv(CV2) sebagai input gambar
2. Setiap *frame* yang dibaca dari *input* gambar tersebut diubah menjadi format RGB melalui proses konversi dari BGR ke RGB agar dapat digunakan oleh model TFLite untuk klasifikasi.
3. Model menghasilkan *output* berupa nilai klasifikasi.
4. *Output* nilai klasifikasi disimpan lalu ditampilkan melalui *interface*

3.3 *Application Image Classification*

Dalam *Application Image Classification* ini ada beberapa sistem yang diintegrasikan, beberapa sistem tersebut yaitu:

1. Menghandle untuk menerima data *image* dari kamera melalui *libraries opencv* sebagai *input*.
2. Setiap *frame image* yang dibaca tersebut diubah menjadi format *RGB* melalui proses konversi dari *BGR* ke *RGB* agar dapat digunakan oleh model *TFLite* untuk klasifikasi.
3. *Image* digunakan sebagai input untuk model *tensorflow lite*
4. Model menghasilkan *output* berupa label dan nilai klasifikasi.
5. Label dan nilai klasifikasi dari model ditampilkan melalui interface *Application Image Classification*
6. Kendali terhadap GPIO berdasarkan nilai klasifikasi.

Berikut adalah penjelasan mengenai program ApplicationImageClassification.py atau program *Application Image Classification*.

1. Baris-baris ini meng-*import* berbagai pustaka yang akan digunakan dalam program. Ini termasuk pustaka os, argparse, sys, time, cv2 (OpenCV), dan berbagai modul dari pustaka tflite_support untuk mendukung klasifikasi *image* menggunakan *TensorFlow Lite*. Selain itu, pustaka RPi.GPIO digunakan untuk mengendalikan pin GPIO pada Raspberry Pi.

```
import os

import argparse

import sys

import time

import cv2

from tflite_support.task import core
```

```
from tflite_support.task import processor

from tflite_support.task import vision


import RPi.GPIO as GPIO
```

2. Baris-baris ini mengatur mode GPIO ke "GPIO.BOARD", yang berarti menggunakan penomoran pin berdasarkan nomor pin fisik pada Raspberry Pi. Kemudian, pin GPIO untuk berbagai fungsi (mulai, berhenti, LED hijau, LED merah, darurat, pengemudi, cacat 1, dan cacat 2) didefinisikan dengan nomornya masing-masing.

```
GPIO.setmode(GPIO.BOARD)


button_start = 16

button_stop = 18

green_led = 37

red_led = 35

emergency = 33

driver = 15

cacat1 = 13

cacat2 = 11
```

3. Baris-baris ini menginisialisasi pin GPIO yang telah didefinisikan sebelumnya. Pin "button_start" dan "button_stop" diatur sebagai input dengan pull-up resistor yang diaktifkan, sementara pin lainnya diatur sebagai *output*.

```
#inisialisasi GPIO

GPIO.setup(button_start,GPIO.IN,pull_up_down=GPIO.PUD_UP)

GPIO.setup(button_stop,GPIO.IN,pull_up_down=GPIO.PUD_UP)

GPIO.setup(green_led, GPIO.OUT)

GPIO.setup(red_led, GPIO.OUT)

GPIO.setup(driver, GPIO.OUT)

GPIO.setup(emergency, GPIO.OUT)

GPIO.setup(cacat1, GPIO.OUT)

GPIO.setup(cacat2, GPIO.OUT)
```

4. Baris-baris ini mendefinisikan parameter-parameters yang digunakan untuk menampilkan hasil klasifikasi dan FPS pada *image* dan juga parameter untuk *font* yang akan menampilkan hasil klasifikasi.

```
# Parameter Font pada tampilan aplikasi
_ROW_SIZE = 30  # pixels
_LEFT_MARGIN = 25  # pixels
_TEXT_COLOR = (255, 255, 255)  # white
_FONT_SIZE = 0.6
_FONT_THICKNESS = 1
_FPS_AVERAGE_FRAME_COUNT = 10
```

5. Mendefinisikan fungsi time_elapsed yang digunakan untuk menghitung dan mencetak durasi waktu yang telah berlalu. Ini akan digunakan untuk mengukur berapa lama waktu yang diperlukan untuk berbagai tugas, seperti inferensi.

```
def time_elapsed(start_times,event):
    time_now=time.time()
    duration = (time_now - start_times)*1000
    duration = round(duration,2)
    print (">> ", duration, " ms (",event, ")" )
```

6. Fungsi *run* adalah fungsi utama program yang akan menjalankan klasifikasi image berkelanjutan dari *image* yang diperoleh dari kamera. Fungsi ini menerima beberapa argumen, termasuk nama model TFLite, label, jumlah hasil maksimal, ambang batas skor, jumlah *thread* CPU, penggunaan EdgeTPU, ID kamera, lebar frame, dan tinggi *frame*. Argumen-argumen ini digunakan untuk mengkonfigurasi dan menjalankan model klasifikasi image pada image yang diambil dari kamera.

```
def run(model: str, label: str,max_results: int, score_threshold: float, num_threads: int,
        enable_edgetpu: bool, camera_id: int, width: int, height: int) -> None:
"""
Menjalankan inferensi secara kontinu pada gambar yang diperoleh dari kamera.
Argumen:
model: Nama model klasifikasi gambar TFLite.
max_results: Maksimum kategori klasifikasi yang ditampilkan.
score_threshold: Ambang batas skor hasil klasifikasi yang hanya ditampilkan.
num_threads: Jumlah core CPU untuk menjalankan model.
enable_edgetpu: Opsi jika akan menjalankan model dengan integrasi EdgeTPU.
camera_id: ID kamera yang akan dikirimkan ke OpenCV.
width: Lebar bingkai yang diambil dari kamera.
height: Tinggi bingkai yang diambil dari kamera.
```

```
"""
```

7. Baris program menginisialisasi model klasifikasi *image* dari file TFLite dengan konfigurasi yang sesuai. Ini termasuk penggunaan EdgeTPU jika diaktifkan.

```
# Inisialisasi model dan opsi edgetpu
base_options = core.BaseOptions(
file_name=model, use_coral=enable_edgetpu,
num_threads=num_threads)
```

8. Program mengatur opsi untuk klasifikasi *image*, termasuk jumlah hasil maksimal dan ambang batas skor. Kemudian, opsi ini digunakan untuk membuat pengklasifikasi dari modul *vision*.

```
#pengaturan jika opsi edgetpu dipilih
classification_options = processor.ClassificationOptions(
    max_results=max_results, score_threshold=score_threshold)
 options = vision.ImageClassifierOptions(
    base_options=base_options,
classification_options=classification_options)
  classifier = vision.ImageClassifier.create_from_options(options)
```

9. Variabel-variabel ini digunakan untuk menghitung dan mencatat FPS selama program berjalan.

```
# Variable untuk menghitung FPS
  counter, fps = 0, 0
  start_time = time.time()
```

10. Program menginisialisasi perangkat kamera dan mengatur beberapa properti, seperti lebar dan tinggi *frame*. Variabel run digunakan untuk mengendalikan status jalannya inspeksi.

```
# memulai mengambil data gambar dari kamera
  cap = cv2.VideoCapture(camera_id)
  cap.set(cv2.CAP_PROP_FRAME_WIDTH, width)
  cap.set(cv2.CAP_PROP_FRAME_HEIGHT, height)
  run = 0
```

11. Loop utama program dimulai di sini. Program secara berulang-ulang mengambil *image* dari kamera, memutar image secara *horizontal* (flipping), dan memeriksa apakah operasi pembacaan berhasil.

```
# Secara kontinu mengambil gambar dari kamera dan mengklasifikasi
  while cap.isOpened():
```

```
success, image = cap.read()

if not success:

  sys.exit(

      'ERROR: Unable to read from webcam. Please verify your webcam settings.'

  )


counter += 1

image = cv2.flip(image, 1)
```

12. Program mengambil waktu awal dan kemudian mengonversi *image* dari format BGR (*Blue-Green-Red*) ke RGB (*Red-Green-Blue*) karena model TFLite memerlukan image dalam format RGB. Selanjutnya, gambar RGB dikonversi ke objek *TensorImage* yang akan digunakan untuk menjalankan inferensi gambar.

```
start_times = time.time()

# konversi dari BGR to RGB untuk kebutuhan model TFLite

rgb_image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)


# Membuat TensorImage

tensor_image = vision.TensorImage.create_from_array(rgb_image)
```

13. Program menjalankan inferensi menggunakan model yang telah diinisialisasi sebelumnya. Hasil klasifikasi, termasuk kategori dan skor, disimpan dalam *variabel categories*. Selain itu, program membaca label nama yang sesuai dengan hasil klasifikasi dari file teks yang telah didefinisikan.

```
# Menyimpan hasil klasifikasi dan label kategori klasifikasi

categories = classifier.classify(tensor_image)


label_file = os.path.join(os.path.dirname(model), label)

with open(label_file, 'r') as f :

   label_names = [line.strip() for line in f.readlines()]
```

14. Program mengekstraksi hasil klasifikasi dan mengiterasi melalui kategori-kategori yang ditemukan. Itu memperoleh indeks kategori, nama kategori, dan skor kategori. Selain itu, program juga memeriksa status tombol "*start*" dan "*stop*" yang terhubung ke pin GPIO untuk mengendalikan jalannya inspeksi. Selanjutnya, program akan mengubah status inspeksi (*run*) berdasarkan hasil klasifikasi.

```
# Menampilkan hasil klasifikasi

for idx, category in enumerate(categories.classifications[0].categories):
```

```
    index = category.index
    category_name = label_names[index]
    score = round(category.score, 2)
    score_percentage = score * 100
    result_text = ' Classification Result >> ' + str(category_name) + ' ' + str(score_percentage)
+ '%'
    start_button_state = GPIO.input(button_start)
    stop_button_state = not GPIO.input(button_stop)


    if start_button_state == 0:
       run = 1
    elif stop_button_state == False:
       run = 0
```

15. Program mencetak pesan berdasarkan hasil klasifikasi. Jika kain dinyatakan tidak bermasalah (indeks 0), itu mencetak bahwa "Kain tidak bermasalah". Jika inspeksi sedang berjalan (run == 1) dan hasilnya adalah "Cacat Jarang" (indeks 1), maka program mengubah status inspeksi menjadi 2. Jika hasilnya adalah "Cacat Slap" (indeks 2), program mengubah status inspeksi menjadi 3.

```
    if index == 0:
       print("Kain tidak bermasalah")
    elif  run == 1 and index == 1 :
       print("Cacat Jarang")
       run = 2
    elif  run == 1 and index == 2 :
       run = 3
       print("Cacat Slap")
```

16. Program mengendalikan perangkat keras eksternal, seperti lampu LED dan *relay*, berdasarkan hasil klasifikasi. Ini adalah bagian yang mengendalikan tindakan fisik saat kain dinyatakan cacat. Status ada empat kategori yaitu status run 1 ketika kain berjalan normal atau sesuai standar, status run 0 ketika keadaan tombol stop ditekan, status run 2 dan run 3 ketika inspeksi mendeteksi adanya kain cacat.

```
if run == 1: #kondisi berjalan normal
       GPIO.output(green_led, GPIO.HIGH)
       GPIO.output(red_led, GPIO.HIGH)
```

```
        GPIO.output(driver, GPIO.HIGH)
        GPIO.output(cacat1, GPIO.LOW)
        GPIO.output(cacat2, GPIO.LOW)
        GPIO.output(emergency, GPIO.LOW)
    elif run == 0: #kondisi stop
        GPIO.output(green_led, GPIO.LOW)
        GPIO.output(red_led, GPIO.LOW)
        GPIO.output(driver, GPIO.LOW)
        GPIO.output(cacat1, GPIO.LOW)
        GPIO.output(cacat2, GPIO.LOW)
        GPIO.output(emergency, GPIO.LOW)
    elif run == 2:
        GPIO.output(green_led, GPIO.LOW)
        GPIO.output(red_led, GPIO.LOW)
        GPIO.output(driver, GPIO.LOW)
        GPIO.output(cacat1, GPIO.HIGH)
        GPIO.output(cacat2, GPIO.LOW)
        GPIO.output(emergency, GPIO.HIGH)
    elif run == 3:
        GPIO.output(green_led, GPIO.LOW)
        GPIO.output(red_led, GPIO.LOW)
        GPIO.output(driver, GPIO.LOW)
        GPIO.output(cacat1, GPIO.LOW)
        GPIO.output(cacat2, GPIO.HIGH)
        GPIO.output(emergency, GPIO.HIGH)
```

17. Bagian ini digunakan untuk menampilkan hasil klasifikasi (misalnya, "Cacat Jarang" atau "Cacat Slap") yang sedang diambil. Fungsi cv2.putText dari OpenCV digunakan untuk menambahkan teks pada image. Lokasi teks ditentukan oleh *text_location*, dan teksnya adalah *result_text* yang telah dibuat sebelumnya.

```
text_location = (50,450)
cv2.putText(image, result_text, text_location, cv2.FONT_HERSHEY_TRIPLEX,
            _FONT_SIZE, _TEXT_COLOR, _FONT_THICKNESS)
```

18. Program memanggil fungsi time_elapsed yang telah didefinisikan sebelumnya untuk mengukur waktu yang diperlukan untuk melakukan inferensi. Hasil klasifikasi dan waktu yang diukur dicetak ke konsol.

```
time_elapsed(start_t1, "Inference")

  print(result_text)
```

19. Program menghitung FPS (frame per detik) dengan menghitung jumlah *frame* yang telah diproses dalam periode waktu tertentu. FPS dicetak dan digunakan untuk memantau kecepatan pemrosesan *image*.

```
# Mengkalkulasi FPS

if counter % _FPS_AVERAGE_FRAME_COUNT == 0:

  end_time = time.time()

  fps = _FPS_AVERAGE_FRAME_COUNT / (end_time - start_time)

  start_time = time.time()
```

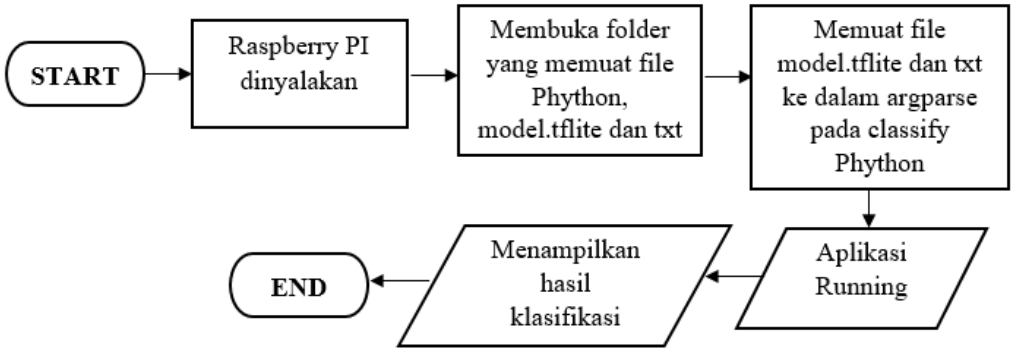
20. Program menghitung FPS dan menyimpannya dalam variabel fps. Kemudian, program mengkonversi FPS menjadi milidetik per frame dan mencetaknya di *image*.

```
if fps > 0:

    fps_ms_raw = 1 / fps * 1000

  else:

    fps_ms_raw = 0

  fps_ms = round(fps_ms_raw, 2)

  fps_text = str(int(fps)) +  ' fps / ' + str(fps_ms) + ' ms'

  text_location = (_LEFT_MARGIN, _ROW_SIZE)

  cv2.putText(image, fps_text, text_location, cv2.FONT_HERSHEY_TRIPLEX,

          0.5, _TEXT_COLOR, 1)
```

21. Program memeriksa apakah tombol "ESC" pada keyboard ditekan. Jika ditekan, program berhenti dan melepaskan sumber daya yang digunakan (kamera dan jendela tampilan). Itu adalah penjelasan untuk bagian akhir dari program Python. Program ini berfungsi untuk melakukan inspeksi dengan mengklasifikasikan kain dan mengendalikan perangkat keras eksternal berdasarkan hasil klasifikasi.

```
# memberhentikan program Ketika tombol ESC ditekan

if cv2.waitKey(1) == 27:

  break

cv2.imshow('image_classification', image)
```

```
cap.release()

cv2.destroyAllWindows()
```



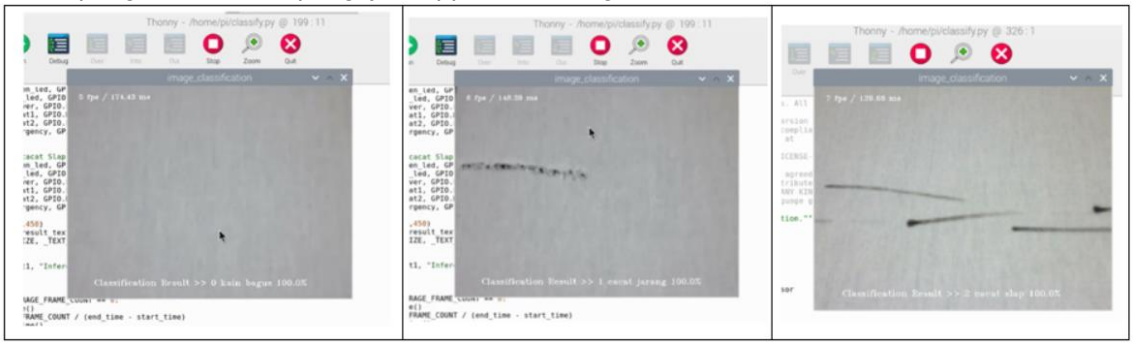
Gambar 4. *Flowchart aplikasi.*

Berdasarkan Gambar 4 proses menjalankan aplikasi ini dimulai dari menyalakan *Raspberry*, Lalu memuat file model.tflite dan txt yang sudah dijadikan satu *file* dengan file *Python* ke dalam *argparse* pada *classify Python*. Lalu aplikasi yang sudah dibuat bisa di *running*. Aplikasi akan menampilkan hasil dari klasifikasi yang terhubung dengan kamera pada alat.

4. Hasil Penelitian dan Pengujian

4.1 Hasil Application Image Classification

Dalam pengujian hasil klasifikasi model, variabel-variabel utama yang diamati meliputi data FPS (*Frame Per Second*), waktu *inference* dan hasil klasifikasi. Data-data ini merupakan komponen penting dalam pengambilan data pengujian *Application Image Classification*.



Gambar 5. *Hasil application image classification.*

Gambar 5 adalah tampilan dari *Application Image Classification*. Dalam tampilan ini, terdapat beberapa informasi penting yang disajikan:

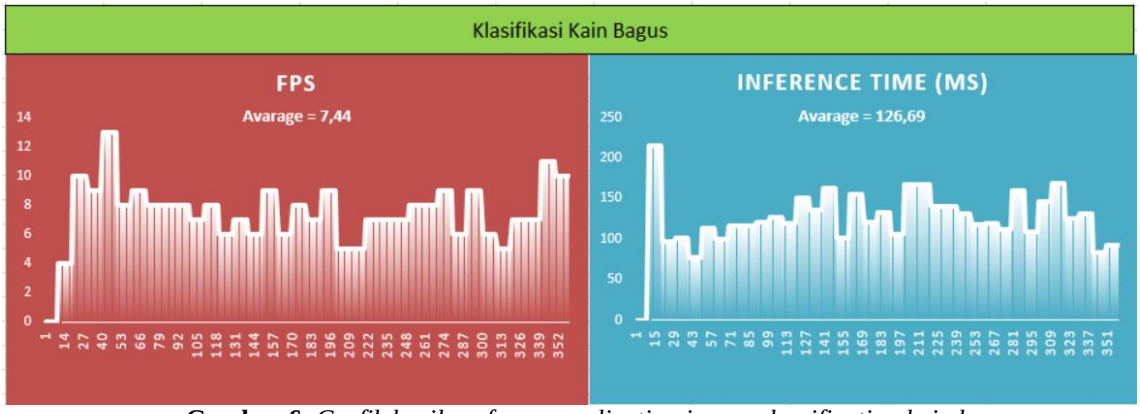
1. *Image* yang sedang di capture oleh kamera: Ini adalah image atau citra yang saat itu sedang diambil oleh kamera.
2. Data FPS: Data FPS mengacu pada *Frames Per Second*, yaitu seberapa cepat perangkat mampu menghasilkan bingkai image dalam satu detik.
3. *Data inference*: *Data inference* mengacu pada waktu yang diperlukan untuk melakukan proses inferensi terkait dengan image yang ditangkap.
4. Hasil klasifikasi dengan nama labelnya: Ini adalah informasi terkait dengan hasil klasifikasi image yang sedang di capture. Hasil klasifikasi ini dapat mencakup kategori atau label yang menjelaskan apa yang terdapat pada image tersebut.

5. Hasil nilai klasifikasi: Nilai klasifikasi ini adalah indikasi tingkat keyakinan sistem terhadap hasil klasifikasi yang diberikan. Ketika nilai klasifikasi mencapai 100%, itu menandakan bahwa sistem sangat yakin bahwa hasil klasifikasinya benar dan sesuai. Ini mencerminkan tingkat keyakinan yang sangat tinggi dalam keakuratan klasifikasi tersebut.

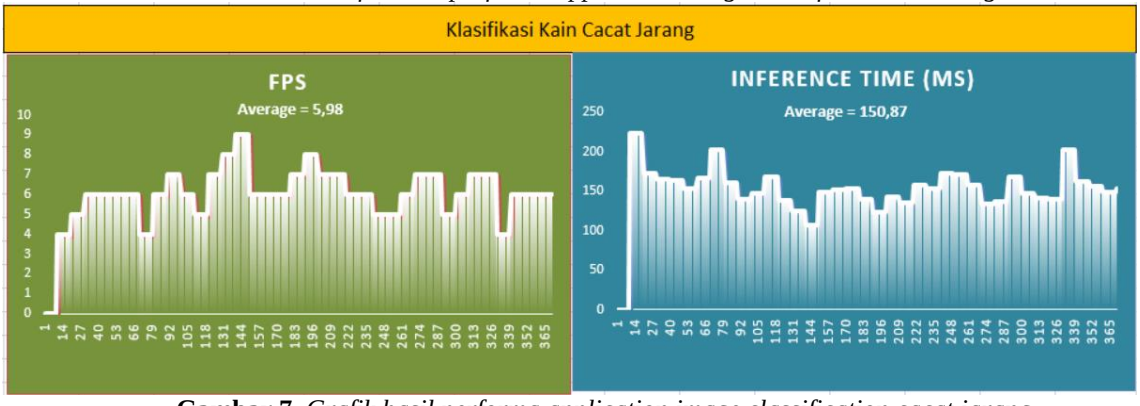
4.2 Pengujian Performa *Application Image Classification*

Pengujian Performa *Application Image Classification* mencakup pengukuran kecepatan proses inferensi dalam parameter (*Inference Times*), yaitu seberapa cepat model dapat menghasilkan output klasifikasi, dan tingkat kecepatan dalam satuan *frame per second* (FPS), yang menunjukkan seberapa banyak frame gambar yang dapat diproses dalam satu waktu. Data yang diambil adalah hasil dari pengambilan data sebelumnya, yang berupa file csv. Nilai parameter inference times dan fps untuk setiap kategori ditunjukkan pada Gambar 6-8 berikut.

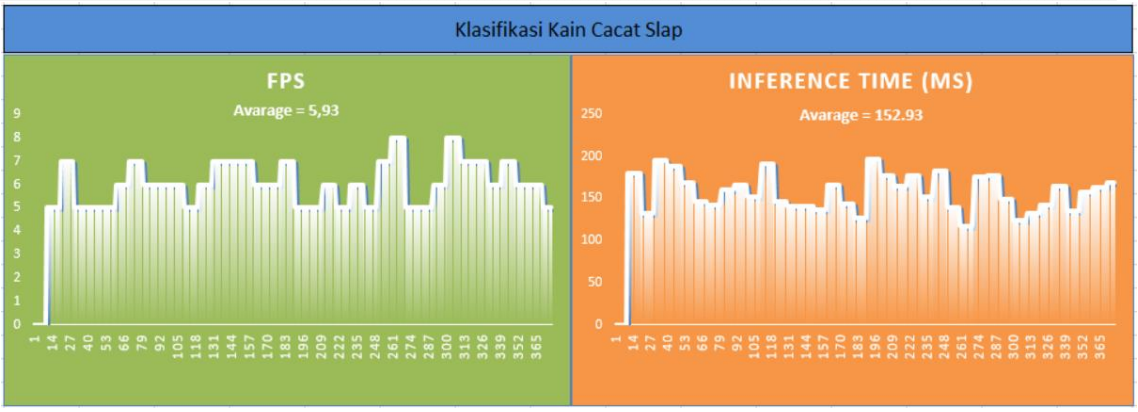
Kategori	Rata-rata FPS	Rata-rata Inference (ms)
Kain Bagus	7,44	126,69
Cacat Jarang	5,98	150,87
Cacat Slap	5,93	152,93
Semua Kategori	6,45	143,43



Gambar 6. *Grafik hasil performa application image classification kain bagus.*



Gambar 7. *Grafik hasil performa application image classification cacat jarang.*



Gambar 8. *Grafik hasil performa application image classification cacat slap.*

Dari hasil tersebut, dapat diambil kesimpulan bahwa *Application Image Classification* ini dapat menghasilkan klasifikasi dengan waktu inferensi yaitu dengan rata rata pada semua kategori sebesar 143,43 ms dan rata rata FPS pada semua kategori sebesar 6,45 FPS. Meskipun ada perbedaan kecil dalam waktu inferensi dan FPS antara klasifikasi kain cacat slap dengan yang lain, secara umum aplikasi ini dapat mengklasifikasikan gambar dengan respons yang baik. Kecepatan FPS yang konsisten di antara klasifikasi menunjukkan konsistensi dalam performa aplikasi.

5. KESIMPULAN

Berdasarkan hasil dan pembahasan pada penelitian ini maka dapat diperoleh kesimpulan sebagai berikut :

1. Telah mampu membangun sebuah *Application Image Classification* menggunakan bahasa pemrograman *Python*, yang mampu mendeteksi kain cacat dengan hasil tangkapan kamera secara real-time.
2. Pengujian performa *Application Image Classification* dapat menghasilkan klasifikasi dengan waktu inferensi yaitu dengan rata rata pada semua kategori sebesar 143,43 ms dan rata rata FPS pada semua kategori sebesar 6,45 FPS.
3. Secara keseluruhan, pencapaian-pencapaian ini merupakan langkah penting dalam usaha meningkatkan efisiensi dan kualitas produksi dalam industri tekstil. Dengan tingkat akurasi klasifikasi yang tinggi dalam mendeteksi cacat kain, ini merupakan solusi inovatif dalam mencapai tujuan utama yaitu menghasilkan produk kain berkualitas dan meningkatkan efisiensi produksi.

Referensi

[1] Kementrian Perindustrian, “Mendorong Kinerja Industri Tekstil dan Produk Tekstil di Tengah Pandemi,” *Buku Anal. Pembang. Ind.*, pp. 1–37, 2021.

[2] A. Y. Kristanto, R. Rumita, and Sriyanto, “Analisis Penyebab Cacat Kain dengan Menggunakan Metode Failure Mode and Effect Analysis (FMEA) dan Fault Tree Analysis (FTA),” *Ind. Eng. Online J.*, vol. 5, no. 1, pp. 1–8, 2016.

[3] A. Y. Kristanto and R. Rumita, “Analisis Penyebab Cacat Kain dengan Menggunakan Metode Failure Mode And Effect Analysis (FMEA) dan Fault Tree Analysis (FTA),” pp. 1–8.

[4] S. Kasus, P. T. Iskandar, I. Printing, D. F. Dewanti, and D. Pujotomo, “Analisis Penyebab Cacat Produk Kain dengan Menggunakan Metode Failure Mode And Effect Analysis (FMEA)”.

[5] K. Azmi, S. Defit, and S. Sumijan, “Implementasi Convolutional Neural Network (CNN) untuk Klasifikasi Batik Tanah Liat Sumatera Barat,” *J. Unitek*, vol. 16, no. 1, pp. 28–40, 2023, doi: 10.52072/unitek.v16i1.504.

[6] A. Ali, R. Pinciroli, F. Yan, and E. Smirni, “Batch: Machine learning inference serving on serverless platforms with adaptive batching,” *Int. Conf. High Perform. Comput. Networking, Storage Anal. SC*, vol. 2020-Novem, 2020, doi: 10.1109/SC41405.2020.00073.

[7] I. D. Id, “Machine Learning : Teori , Studi Kasus dan Implementasi Menggunakan Python,” no. July, 2021, doi: 10.5281/zenodo.5113507.

[8] C. Chazar and M. H. Rafsanjani, “LPPM STMIK ROSMA / Prosiding Seminar Nasional : Inovasi

& Adopsi Teknologi Penerapan Teachable Machine Pada Klasifikasi Machine Learning Untuk Identifikasi Bibit Tanaman," 2022.
[9] C. Chazar and M. H. Rafsanjani, "Penerapan Teachable Machine pada Klasifikasi Machine Learning untuk Identifikasi Bibit Tanaman Pendahuluan," 2022.
[10] A. Roihan, P. A. Sunarya, and A. S. Rafika, "Pemanfaatan Machine Learning dalam Berbagai Bidang: Review paper," *IJCIT (Indonesian J. Comput. Inf. Technol.*, vol. 5, no. 1, pp. 75–82, 2020, doi: 10.31294/ijcit.v5i1.7951.
[11] A. Lindholm, N. Wahlström, F. Lindsten, and T. B. Schön, "Supervised Machine Learning: Statistical Machine Learning course," p. 112, 2019, [Online]. Available: http://www.it.uu.se/edu/course/homepage/sml/literature/lecture_notes.pdf
[12] T. A. Dompeipen and S. R. U. . Sompie, "Penerapan computer vision untuk pendeteksian dan penghitung jumlah manusia," *J. Tek. Inform.*, vol. 15, no. 4, pp. 1–12, 2020.
[13] A. Purno and W. Wibowo, "Implementasi Teknik Computer Vision Dengan Metode Colored Markers Trajectory Secara Real Time," *J. Tek. Inform.*, vol. 8, no. 1, pp. 38–42, 2016.
[14] B. G. Batchelor, "Machine Vision for Industrial Applications BT - Machine Vision Handbook," B. G. Batchelor, Ed., London: Springer London, 2012, pp. 1–59. doi: 10.1007/978-1-84996-169-1_1.
[15] H. Bandyopadhyay, "What Is Computer Vision? [Basic Tasks & Techniques]," *V7*, 2022.
[16] A. Purno and W. Wibowo, "Implementasi Teknik Computer Vision Dengan Metode Colored Markers Trajectory Secara Real Time • realtime • image," vol. 8, no. 1, pp. 38–42, 2016.
[17] *Sustainability and Circularity in the Textile Value Chain - A Global Roadmap Textile Value Chain*. 2023. doi: 10.59117/20.500.11822/42580.
[18] S. Sha *et al.*, "Image Classification and Restoration of Ancient Textiles Based on Convolutional Neural Network," *Int. J. Comput. Intell. Syst.*, vol. 17, no. 1, 2024, doi: 10.1007/s44196-023-00381-9.
[19] A. Shanu, "Processes of Manufacturing in the Textile Industry," *J. Emerg. Technol. Innov. Res.*, vol. 6, no. 1, pp. 115–119, 2019.
[20] J. Xu, Y. Wei, A. Wang, H. Zhao, and D. Lefloch, "Analysis of Clothing Image Classification Models: A Comparison Study between Traditional Machine Learning and Deep Learning Models," *Fibres Text. East. Eur.*, vol. 30, no. 5, pp. 66–78, 2022, doi: 10.2478/ftee-2022-0046.
[21] R. Sathya, A. Asha, K. Muralikrishna, and V. Bakyalakshmi, "Bespoke Measurement based on Convolutional Neural Network using OpenCV Bespoke Measurement based on Convolutional Neural Network using OpenCV," no. May, 2021, doi: 10.30534/ijatcse/2020/39932020.
[22] M. A. Abu, N. H. Indra, A. H. A. Rahman, N. A. Sapiee, and I. Ahmad, "A study on image classification based on deep learning and tensorflow," *Int. J. Eng. Res. Technol.*, vol. 12, no. 4, pp. 563–569, 2019.
[23] E. Ariesto, U. Malahina, R. P. Hadjon, and F. Y. Bisilisin, "Teachable Machine : Real-Time Attendance of Students Based on Open Source System," vol. 6, no. 3, pp. 140–146, 2022, doi: 10.30865/ijics.v6i3.4928.
[24] H. Taherdoost, *Machine Learning Algorithms*. 2023. doi: 10.4018/978-1-7998-9220-5.ch054.
[25] K. Wisnudhanti and F. Candra, "Image Classification of Pandawa Figures Using Convolutional Neural Network on Raspberry Pi 4," *J. Phys. Conf. Ser.*, vol. 1655, no. 1, 2020, doi: 10.1088/1742-6596/1655/1/012103.
[26] M. Penelitian and M. Pengembangan, "Implementasi Raspberry Pi 4 Sebagai Server E-Learning," vol. 13, 2021.
[27] I. D. Wijaya, U. Nurhasan, and M. A. Barata, "Implementasi Raspberry Pi Untuk Rancang Bangun Sistem Keamanan Pintu Ruang Server Dengan Pengenalan Wajah Menggunakan Metode Triangle Face," *J. Inform. Polinema*, vol. 4, no. 1, p. 9, 2017, doi: 10.33795/jip.v4i1.138.
[28] F. B. Setiawan, H. W. Kusuma, S. Riyadi, and leonardo H. Pratomo, "Penerapan PI Cam Menggunakan Program Berbasis Raspberry PI 4," *Cyclotr. J. Tek. Elektro*, vol. 5, no. 2, pp. 51–56, 2022.
[29] B. R. Pi, "Penerapan PI Cam Menggunakan Program," vol. 5, no. 02, 2022.
[30] P. Corke, *Vision and ALGORITHMS*.