PROTOTIPE APLIKASI PENGHITUNG JUMLAH KENDARAAN DI JALAN RAYA BERBASIS DEEP LEARNING DENGAN ARSITEKTUR YOLOV3

Abstract

Aplikasi ini merupakan sebuah prototipe aplikasi penghitung jumlah kendaraan pada jalan raya yang mengimplementasikan Algoritma Deep Learning dengan arsitektur YoloV3 yang merupakan salah satu model Deep Learning yang memiliki kemampuan yang baik dalam melakukan deteksi objek dari sebuah gambar. Tujuan dibuatnya aplikasi tersebut adalah untuk membantu proses penghitungan jumlah kendaraan yang melintas di suatu ruas jalan. Penghitungan kendaraan dilakukan berdasarkan tiga jenis kendaraan yang paling banyak digunakan, yakni mobil (minibus), bus, dan sepeda motor. Aplikasi ini dapat diintegrasikan dengan kamera CCTV yang terpasang di ruas-ruas jalan, terutama pada jembatan penyebarangan yang melintang di atas ruas jalan, namun pada prototipe ini kamera yang digunakan adalah kamera smartphone yang dipasang di atas jembatan penyebarangan.

Muhammad Fachrie, S.T., M.Cs.

muhammad.fachrie@staff.uty.ac.id

A. Penjelasan tentang Aplikasi

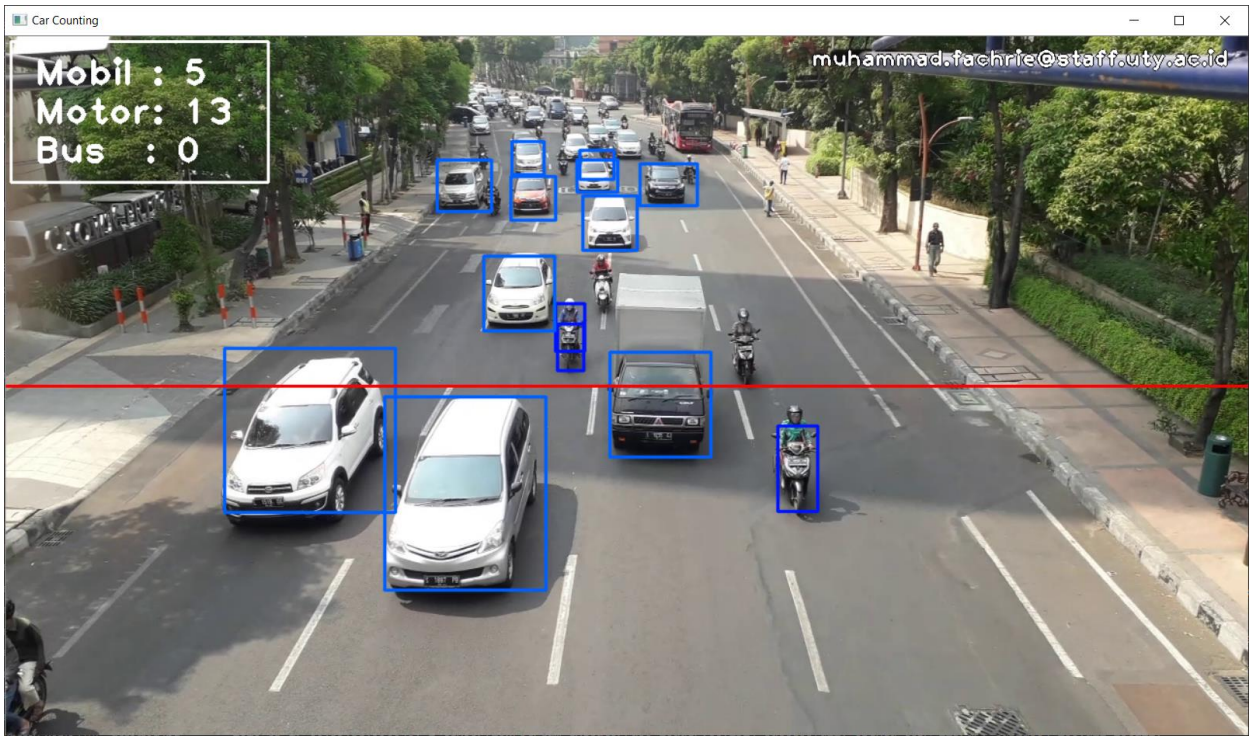
Aplikasi ini merupakan sebuah prototipe dari sistem penghitung jumlah kendaraan yang memanfaatkan teknologi Kecerdasan Buatan serta kamera-kamera CCTV yang terpasang di ruas-ruas jalan raya. Dengan adanya aplikasi ini, diharapkan mampu mendukung terciptanya model Smart Traffic Monitoring System yang dapat membantu manusia untuk memantau kepadatan lalu lintas di ruas-ruas jalan bermodalkan program komputer. Secara spesifik, aplikasi yang dibangun ini memiliki kemampuan untuk menghitung jumlah kendaraan yang melintas di suatu ruas jalan. Proses penghitungan jumlah kendaraan tersebut dibedakan berdasarkan tiga jenis kendaraan, yakni mobil (minibus), bus, dan sepeda motor.

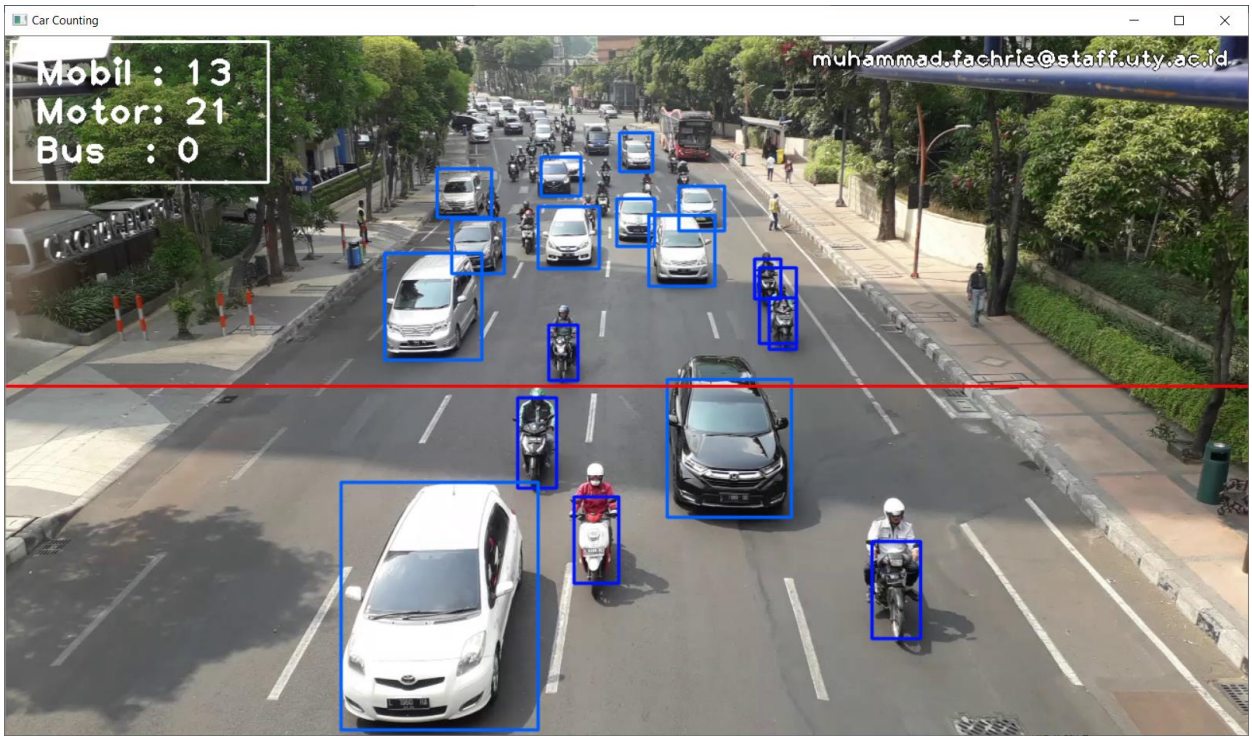
Kemampuan deteksi dan klasifikasi jenis kendaraan yang dimiliki aplikasi ini diperoleh dengan memanfaatkan Algoritma Deep Learning yang ditanamkan di dalamnya. Arsitektur Deep Learning yang digunakan adalah model YoloV3 yang dikembangkan oleh Joseph Redmon yang memiliki kemampuan deteksi dan klasifikasi objek yang akurat, serta memiliki kompleksitas komputasi yang lebih ringan dari arsitektur Deep Learning lainnya. Arsitektur YoloV3 sendiri menerapkan konsep Convolutional Neural Network (CNN), yakni model fundamental Deep Learning yang digunakan untuk mengenali objek pada sebuah citra atau gambar. Meskipun demikian, Algoritma Deep Learning membutuhkan perangkat keras yang mumpuni -yang tentunya memiliki harga yang mahal- untuk mengeksekusi kode program yang dibuat.

Proses pembuatan aplikasi ini menggunakan Tensorflow yang merupakan salah satu *framework* Deep Learning yang paling populer saat ini. Penggunaan *framework* tersebut memudahkan kita untuk membangun program computer berbasis Deep Learning yang memanfaatkan proses komputasi dengan GPU (*Graphic Processing Unit*). Tensorflow yang digunakan pada saat pembuatan aplikasi ini adalah versi 1.4.0 yang merupakan salah satu versi Tensorflow yang stabil. Selain itu, pengembangan aplikasi ini juga menggunakan *library* ImageAI versi 2.1.6 karya Moses Olafenwa yang memudahkan kita dalam membangun aplikasi berbasis *Object Detection* berbasis Deep Learning. *Library* ImageAI mampu melakukan deteksi objek dengan beberapa arsitektur Deep Learning, yakni RetinaNet, YoloV3, dan TinyYolo, sehingga menjadikan proses pengembangan bersifat fleksibel dalam memilih arsitektur yang sesuai dengan kebutuhan. Kecepatan pendeteksian objek juga dapat disesuaikan dalam empat level, yakni “Normal”, “Fast”, “Faster”, dan “Flash”, namun semakin cepat proses pendeteksian yang digunakan, maka akurasi pendeteksian objek semakin menurun.

B. Tampilan Prototipe Aplikasi

Berikut ini adalah beberapa tampilan dari aplikasi penghitung jumlah kendaraan pada jalan raya. Video-video yang digunakan pada aplikasi ini direkam menggunakan kamera yang ada pada *smartphone* dengan resolusi 4K.





C. Kode Sumber

```
#import libraries
import os
import cv2
from imageai.Detection import ObjectDetection
import threading
import pytesseract
import time
import numpy as np

def detect_object(frame, iteration):
    global output_array
    global all_centroid, car_centroid, motor_centroid, bus_centroid
    all_centroid = []
    car_centroid = []
    motor_centroid = []
    bus_centroid = []

    if (frame is not None):
        output_array, detections = detector.detectCustomObjectsFromImage(custom_objects = custom_objects, input_image = frame, input_type = "array",
                                            output_type = "array", display_object_name = False,
                                            display_percentage_probability = False, minimum_percentage_probability = 50,
                                            thread_safe = True)

        for vehicle in detections:
            #set object centroid
            vehicle_area = vehicle["box_points"]
            x1 = vehicle_area[0]
            x2 = vehicle_area[2]
            y1 = vehicle_area[1]
            y2 = vehicle_area[3]
            centroid = [int((x1+x2)/2), int((y1+y2)/2)]
            all_centroid.append(centroid)

            if (vehicle["name"] == "car"):
                car_centroid.append(centroid)
            elif (vehicle["name"] == "motorcycle"):
                motor_centroid.append(centroid)
            elif (vehicle["name"] == "bus"):
                bus_centroid.append(centroid)

def count_vehicle(centroid_list):
    #min_distance = []
    temp_num_vehicle = 0
    decrease1 = False
    decrease2 = False
    for i in centroid_list[0]:
        #check distance of i to the border line (threshold = 10)
        if (abs(i[1] - line) <= 10):
            temp_num_vehicle += 1
            for j in centroid_list[1]:
                if (abs(j[1] - line) <= 10):
                    distance = ((i[0] - j[0])**2 + (i[1] - j[1])**2)**0.5

                    #check distance between i and j (threshold = 30)
                    if (distance < 15):
                        if (temp_num_vehicle > 0):
                            decrease1 = True
            for j in centroid_list[2]:
                if (abs(j[1] - line) <= 10):
                    distance = ((i[0] - j[0])**2 + (i[1] - j[1])**2)**0.5

                    #check distance between i and j (threshold = 30)
                    if (distance < 15):
                        if (temp_num_vehicle > 0):
                            decrease2 = True

            if (decrease1 or decrease2):
                temp_num_vehicle -= 1

    num_vehicle = temp_num_vehicle
    return num_vehicle

#def tracking(current_centroid, object_centroid):
#read the execution path
execution_path = os.getcwd()

#create object 'detector'
detector = ObjectDetection()

#set and load pretrained CNN model = yolov3
detector.setModelTypeAsYOLOv3()
detector.setModelPath(os.path.join(execution_path, "yolov3.h5"))
detector.loadModel(detection_speed="normal")

#set and load pretrained CNN model = retinanet
#detector.setModelTypeAsRetinaNet()
#detector.setModelPath(os.path.join(execution_path, "resnet50_coco_best_v2.0.1.h5"))
#detector.loadModel(detection_speed="normal")

#define custom objects
custom_objects = detector.CustomObjects(car = True, motorcycle = True, bus = True)

#================================================================================
# VIDEO STREAMING
#================================================================================
#stream the video
#cap = cv2.VideoCapture('D:/Documents/Jobs/IT/VARx/Koleksi Video utk Demo/Video dari Surabaya/20191120_090326.mp4')
cap = cv2.VideoCapture('D:/Documents/Jobs/IT/VARx/Koleksi Video utk Demo/Video dari Surabaya/20191120_150841.mp4')
#cap = cv2.VideoCapture('D:/Documents/Jobs/IT/VARx/Koleksi Video utk Demo/video1.mp4')
#cap = cv2.VideoCapture('D:/Documents/Jobs/IT/VARx/Koleksi Video utk Demo/Video dari Surabaya/20191120_090207.mp4')
#cap = cv2.VideoCapture('D:/Documents/Jobs/IT/VARx/Koleksi Video utk Demo/Video dari Surabaya/20191120_152120.mp4')
#cap = cv2.VideoCapture('D:/Documents/Jobs/IT/VARx/Koleksi Video utk Demo/Video dari Surabaya/20191120_151651.mp4')

```

```python
#initializing variables
step = 1
iteration = 1
frame = []
car_centroid_list = []
motor_centroid_list = []
bus_centroid_list = []
num_vehicle = 0
create_video = False

#loop forever until I press 'q' key on keyboard
while True:
    #read the image/ frame from video
    ret, frame = cap.read()

    if (ret):
        #resize the frames
        height = len(frame)
        width = len(frame[0])
        if (height >= 2000):
            height_resized = int(0.3 * height)
            width_resized = int(0.3 * width)
        elif (height >= 1080):
            height_resized = int(0.7 * height)
            width_resized = int(0.7 * width)
        else:
            height_resized = int(1 * height)
            width_resized = int(1 * width)

        #height_resized = int(1 * height)
        #width_resized = int(1 * width)

        frame_resized = cv2.resize(frame, (width_resized, height_resized))

        if (iteration == 1 and create_video):
            #output_video = cv2.VideoWriter('car_counting_1.avi', cv2.VideoWriter_fourcc('M','J','P','G'), 10, (width_resized, height_resized))
            output_video = cv2.VideoWriter('car_counting_3a.mp4', cv2.VideoWriter_fourcc(*'FMP4'), 10, (width_resized, height_resized))

        #process the image each 2 frames = 30 fps
        if (step % 2 == 0):
            list_vehicle_id = []

            #initialized the thread to detect object
            t = threading.Thread(target=detect_object, args=(frame_resized, iteration, ))

            #start the thread
            t.start()
            t.join()

            #set the line on video
            line = int(0.50*height_resized)   #vertical axis of line border
            cv2.line(output_array, (0, line), (width, line), (0, 0, 255), thickness = 2)    #color: (B,G,R)

            #create list centroid dari 3 frame yang berurutan
            if (iteration <= 3):
                car_centroid_list.append(car_centroid)
                motor_centroid_list.append(motor_centroid)
                bus_centroid_list.append(bus_centroid)
            else:
                #pop/ erase the oldest centroid (centroid in index 0)
                car_centroid_list.pop(0)
                motor_centroid_list.pop(0)
                bus_centroid_list.pop(0)

                #add the current_centroid to centroid_list
                car_centroid_list.append(car_centroid)
                motor_centroid_list.append(motor_centroid)
                bus_centroid_list.append(bus_centroid)

            #count the vehicles
            if (iteration >= 3):
                num_car = num_car + count_vehicle(car_centroid_list)
                num_motor = num_motor + count_vehicle(motor_centroid_list)
                num_bus = num_bus + count_vehicle(bus_centroid_list)
            else:
                num_car = 0
                num_motor = 0
                num_bus = 0

            #show the object centroid
            for i in range(len(all_centroid)):
                x = all_centroid[i][0] - 2
                y = all_centroid[i][1]

                #cv2.putText(output_array, '.', (x, y), cv2.FONT_HERSHEY_PLAIN, 1, (255, 0, 0), 2)

            #Show the video
            cv2.putText(output_array, 'Mobil : ' + str(num_car), (30, 50), cv2.FONT_HERSHEY_PLAIN, 2.5, (255, 255, 255), 3)
            cv2.putText(output_array, 'Motor: ' + str(num_motor), (30, 90), cv2.FONT_HERSHEY_PLAIN, 2.5, (255, 255, 255), 3)
            cv2.putText(output_array, 'Bus  : ' + str(num_bus), (30, 130), cv2.FONT_HERSHEY_PLAIN, 2.5, (255, 255, 255), 3)
            cv2.rectangle(output_array, (5,5), (270, 150), (255, 255, 255), thickness=2)
            author = 'muhammad.fachrie@staff.uty.ac.id'
            x_author = width_resized - 450
            cv2.putText(output_array, author, (x_author, 30), cv2.FONT_HERSHEY_PLAIN, 1.5, (0, 0, 0), 3)
            cv2.putText(output_array, author, (x_author, 30), cv2.FONT_HERSHEY_PLAIN, 1.5, (255, 255, 255), 2)
            cv2.imshow('Car Counting', output_array)

            #write the image to video
            if (create_video):
                output_video.write(output_array)

```

```
            #increment the iteration
            iteration += 1

        step += 1

        if (cv2.waitKey(1) & 0xFF == ord('q')):
            break
    else:
        cap.release()
        cv2.waitKey()
        cv2.destroyAllWindows()
        break

#stop the capturing process
cap.release()
cv2.destroyAllWindows()
```