DAFTAR PUSTAKA

Adriana. 2016. “Analisi Kualitas Air Kolam Renang Indoor dan Outdoor Depok Sport Center dan Tirta Sari Di Kabupaten Sleman Berdasarkan Ketentuan-ketentuan Peraturan Menteri Kesehatan RI No 416/MENKES/PER/IX/1990,” Desember.

Arko, Terry. 2005. *The Book on Water Clarity*. Vanson HaloSOurce, Inc.

Bovik, Alan C. 2000. *Handbook of Image and Video Processing*.

Darmawan, Aan, dan Angki Dwi Saptani. 2010. “Perbandingan Metode Pergeseran Rata-Rata, Pergeseran Logaritma, Dan Alpha Blending Dalam Proses Metamorfosis Dari Dua Gambar Dijital.” *Maranatha Electrical Engineering Journal*.

Gerintya, Scholastica. 2018. “Lari, Indonesia, Lari!” tirto.id. 4 Januari 2018. https://tirto.id/lari-indonesia-lari-cCHS.

Harariet, Fadila, Darmiah Darmiah, dan Imam Santoso. 2017. “Hubungan Jumlah Perenang dengan Sisa Klor di Kolam Renang Antasari Banjarbaru Tahun 2016.” *Jurnal Kesehatan Lingkungan* 14 (1). https://doi.org/10.31964/jkl.v14i1.

Heller, Martin. 2019. “Machine Learning Algorithms Explained.” InfoWorld. 9 Mei 2019. https://www.infoworld.com/article/3394399/machine-learning-algorithms-explained.html.

Manning, Colin E. 2008. “Digital Video Compression.” http://www.newmediarepublic.com/dvideo/compression/adv02.html.

Mediatama, Grahanusa. 2019. “Ini bahaya yang mengintai di kolam renang umum bagi kesehatan.” PT. Kontan Grahanusa Mediatama. 20 Februari 2019. https://kesehatan.kontan.co.id/news/ini-bahaya-yang-mengintai-di-kolam-renang-umum-bagi-kesehatan.

Merchant, Armaan. 2019. “Neural Networks Explained.” Medium. 30 Januari 2019. https://medium.com/datadriveninvestor/neural-networks-explained-6e21c70d7818.

Munir, Rinaldi. 2004. *Pengolahan Citra Digital dengan Pendekatan Algoritmik*.

Narkhede, Sarang. 2019. “Understanding Confusion Matrix.” Medium. 29 Agustus 2019. https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62.

Nugraha, Yogi Adi. 2014. “Implementasi Sistem Otomatis Pada Robot Kapal Berbasis Komputer Vision Untuk Kontes Kapal Cepat Tak Berawak Nasional (KKCTBN).”

PT. Kuhanda Semesta Group. 2019. “7 Syarat Air Kolam Renang Sehat I Anda harus Tahu !” *Kontraktor Kolam Renang Terbaik Jakarta Murah JABODETABEK* (blog). 15 April 2019. https://kuhandagroup.com/syarat-air-kolam-renang-sehat/.

Redmon, Joseph Chet. 2018. “YOLO: Real-Time Object Detection.” 2018. https://pjreddie.com/darknet/yolo/.

Riadi, Muchlisin. 2017. “Pengertian, Manfaat, Prinsip dan Gaya Renang.” *KajianPustaka.com* (blog). 18 Agustus 2017.

https://www.kajianpustaka.com/2017/08/pengertian-manfaat-prinsip-gaya-renang.html.

Rosebrock, Adrian. 2016. “Intersection over Union (IoU) for Object Detection.” *PyImageSearch* (blog). 7 November 2016. https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/.

Shalev-Shwartz, Shai, dan Shai Ben-David. 2014. *Understanding Machine Learning: From Theory to Algorithm*.

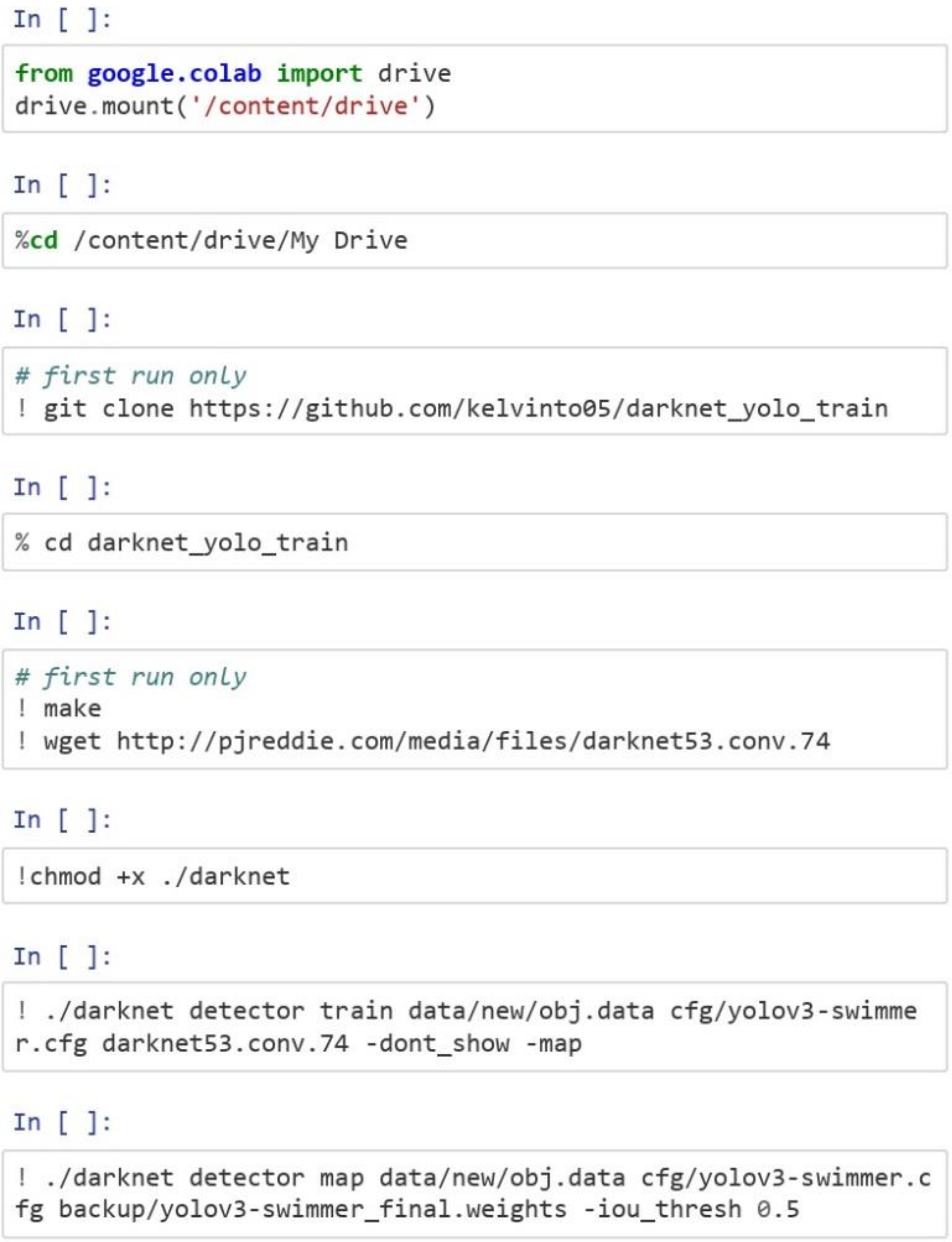
Szeliski, Richard. 2010. *Computer VIsion - Algorithms and Applications*.

Tashandra, Nabilla. 2018. “Trik Agar Renang Datangkan Manfaat Maksimal bagi Tubuh Halaman all.” KOMPAS.com. 28 Agustus 2018. https://lifestyle.kompas.com/read/2018/08/28/090645020/trik-agar-renang-datangkan-manfaat-maksimal-bagi-tubuh.

V, Avinash Sharma. 2017. “Understanding Activation Functions in Neural Networks.” Medium. 30 Maret 2017. https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0.

LAMPIRAN

Lampiran 1. *Source code* training YOLO V3 melalui Google Colab



```
In [ ]:
from google.colab import drive
drive.mount('/content/drive')

In [ ]:
%cd /content/drive/My Drive

In [ ]:
# first run only
! git clone https://github.com/kelvinto05/darknet_yolo_train

In [ ]:
% cd darknet_yolo_train

In [ ]:
# first run only
! make
! wget http://pjreddie.com/media/files/darknet53.conv.74

In [ ]:
!chmod +x ./darknet

In [ ]:
! ./darknet detector train data/new/obj.data cfg/yolov3-swimme
r.cfg darknet53.conv.74 -dont_show -map

In [ ]:
! ./darknet detector map data/new/obj.data cfg/yolov3-swimmer.c
fg backup/yolov3-swimmer_final.weights -iou_thresh 0.5
```

Lampiran 2. *Source code* deteksi dan perhitungan jumlah perenang dengan YOLO V3

```
import argparse
import time
import os
import sys
import configparser
import csv
import datetime
from PIL import Image, ImageDraw, ImageFont
import numpy as np
import pandas as pd
import requests
import matplotlib.pyplot as plt
import cv2

def define_args():
    ap = argparse.ArgumentParser()
    ap.add_argument("-c", "--
config", required=True, help="Configuration file")
    return vars(ap.parse_args())

def read_config(filename):
    print("[INFO] Reading config: {}".format(filename))
    if not os.path.isfile(filename):
        print("[ERROR] Config file \"{}\" not found.".format(filename))
        exit()
    cfg = configparser.ConfigParser()
    cfg.read(filename)
    return cfg

def execute_network(image, network, layernames):
    blob = cv2.dnn.blobFromImage(image, 1 / 255.0, (416, 416), swapRB=True,
    start2 = time.time()
    network.setInput(blob)
    outputs = network.forward(layernames)
    end2 = time.time()
    print("[INFO] YOLO  took      : %2.1f sec" % (end2-start2))
    return outputs

def load_network(network_folder):
    # Derive file paths and check existance
    labelspath = os.path.sep.join([network_folder, "obj.names"])
    if not os.path.isfile(labelspath):
        print("[ERROR] Network: Labels file \"
{}\" not found.".format(labelspath))
        exit()

    weightspath = os.path.sep.join([network_folder, "yolov3.weights"])
    if not os.path.isfile(weightspath):
        print("[ERROR] Network: Weights file \"
{}\" not found.".format(weightspath))
        exit()

    configpath = os.path.sep.join([network_folder, "yolov3.cfg"])
    if not os.path.isfile(configpath):
        print("[ERROR] Network: Configuration file \"
{}\" not found.".format(configpath))
        exit()

    # Network storend in Darknet format
    print("[INFO] loading YOLO from disk...")
    labels = open(labelspath).read().strip().split("\n")
    network = cv2.dnn.readNetFromDarknet(configpath, weightspath)
    names = network.getLayerNames()
    names = [names[i[0] - 1] for i in network.getUnconnectedOutLayers()]
    return network, names, labels

def get_detected_items(layeroutputs, confidence_level, threshold, img_width
    # initialize our lists of detected bounding boxes, confidences, and cla
    detected_boxes = []
    detection_confidences = []
    detected_classes = []

    for output in layeroutputs:
        # loop over each of the detections
        for detection in output:
            # extract the class ID and confidence (i.e., probability) of th
            scores = detection[5:]
            classid = np.argmax(scores)
            confidence = scores[classid]

            # filter out weak predictions by ensuring the detected probabil
            if confidence > confidence_level:
                # scale the bounding box coordinates back relative to the s
                box = detection[0:4] * np.array([img_width, img_height, img
                (center_x, center_y, width, height) = box.astype("int")

                # use the center (x, y)-
coordinates to derive the top left corner of the bounding box
                top_x = int(center_x - (width / 2))
                top_y = int(center_y - (height / 2))
                if top_y <= 122 or top_y >= 875:
                    continue

                # update our list of bounding box coordinates, confidences,
                detected_boxes.append([top_x, top_y, int(width), int(height
                detection_confidences.append(float(confidence))
                detected_classes.append(classid)

    # apply non-
maxima suppression to suppress weak, overlapping bounding boxes
    indexes = cv2.dnn.NMSBoxes(detected_boxes, detection_confidences, confi

    return indexes, detected_classes, detected_boxes, detection_confidences


def update_frame(image, people_indxs, class_ids, detected_boxes, conf_level
    count_swimmer = 0

    if len(people_indxs) >= 1:
        # loop over the indexes we are keeping
        for i in people_indxs.flatten():
            # extract the bounding box coordinates
            (x, y, w, h) = (detected_boxes[i][0], detected_boxes[i]
[1], detected_boxes[i][2], detected_boxes[i][3])

            if classIDs[i] == 0:
                count_swimmer += 1
                # Blur, if required, people in the image
                if blur:
                    image = blur_area(image, max(x, 0), max(y, 0), w, h)
            # draw a bounding box rectangle and label on the frame
            if (show_boxes and classIDs[i] == 0) or box_all_objects:
                color = colors
                cv2.rectangle(image, (x, y), (x + w, y + h), color, 2)
                text = "
{}: {:.2f}".format(labels[classIDs[i]], conf_levels[i])
                cv2.putText(image, text, (x, y - 5), cv2.FONT_HERSHEY_SIMPL

    # write number of people in bottom corner
    text = "Swimmer: {}".format(count_swimmer)
    cv2.putText(image, text, (10, image.shape[0] - 20), cv2.FONT_HERSHEY_SI
    return image, count_swimmer


if __name__ == '__main__':
    # construct the argument parse and parse the arguments
    args = define_args()
    config = read_config(args["config"])

    # Read config
    network_path = config['NETWORK']['Path']
    showpeopleboxes = (config['OUTPUT']['ShowPeopleBoxes'] == "yes")
    showallboxes = (config['OUTPUT']['ShowAllBoxes'] == "yes")
    blurpeople = (config['OUTPUT']['BlurPeople'] == "yes")
    nw_confidence = float(config['NETWORK']['Confidence'])
    nw_threshold = float(config['NETWORK']['Threshold'])
    path_to_images_folder = config['READER']['Inputpath']
    path_to_output_folder = config['OUTPUT']['Outputpath']
    i=1

    # Load the trained network
    (net, ln, LABELS) = load_network(network_path)

    start0 = time.time()
    for image in os.listdir(path_to_images_folder):
        img = cv2.imread(os.path.join(path_to_images_folder, image))
        img_height, img_width, channels = img.shape
        print("
[INFO] Frame W x H: {} x {}".format(img_width, img_height))

        # Initialize a list of colors to represent each possible class labe
        np.random.seed(42)
        COLORS = (128,0,128)

        # Start counting process time
        start1 = time.time()

        # Feed frame to network
        layerOutputs = execute_network(img, net, ln)
        # Obtain detected objects, including cof levels and bounding boxes
        (idxs, classIDs, boxes, confidences) = get_detected_items(layerOutp

        # Update frame with recognised objects
        img, npeople = update_frame(img, idxs, classIDs, boxes, confidences

        end1 = time.time()

        print("[INFO] Total handling  : %2.1f sec" % (end1 - start1))
        print("[INFO] Swimmer in frame : {}".format(npeople))
        print("[INFO] cleaning up...")
        filename = path_to_output_folder + "/swimmer-
output_" + str(i) + ".png"
        # Write output
        cv2.imwrite(filename ,img)
        i+=1
    end0 = time.time()
    print("[INFO] Total process  : %2.1f sec" % (end0 - start0))
```