Jurnal Kecerdasan Buatan dan Teknologi Informasi
Vol. 2, No. 2, Mei 2023, hlm. 70-80
e-ISSN: 2964-2922, p-ISSN: 2963-6191

OBJECT DETECTION UNTUK MENDETEKSI CITRA BUAH BUAHAN MENGGUNAKAN METODE YOLO

Dede Haris Saputra[1], Bahtiar Imran*[2], Juhartini[3]

[1]Teknik Informatika, Fakultas Teknologi Komunikasi dan Informasi, Universitas Teknologi Mataram, Indonesia
[2]Rekayasa Sistem Komputer, Fakultas Teknologi Komunikasi dan Informasi, Universitas Teknologi Mataram, Indonesia
[3]Rekayasa Perangkat Lunak, Fakultas Teknologi Komunikasi dan Informasi, Universitas Teknologi Mataram, Indonesia

Email: [1]harisc.saputa@gmail.com, [2]bahtiarimranlombok@gmail.ac.id , [3]juhartini@gmail.com

(Naskah masuk : 14 Desember 2022, Revisi : 8 Mei 2023, Diterbitkan : 20 Mei 2023)

Abstrak

Perkembangan ilmu pengetahuan dan teknologi dalam *Artificial Intelligence* yang sangat pesat saat ini, telah membawa perubahan yang sangat pesat pula dalam berbagai aspek kehidupan. Terutama kecerdasan buatan merupakan sebuah teknologi komputer atau mesin yang memiliki kecerdasan layaknya manusia. Sederhananya sebuah instruksi pintar yang diberikan kepada program maupun mesin, salah satunya yaitu Object Detection untuk mendeteksi citra buah menggunakan *You Only Look Once* (YOLO). Metode yang dapat digunakan untuk pengenalan objek pada citra buah adalah Deep Learning. *You Only Look Once* (YOLO) merupakan salah satu model *Deep Learning* yang dapat digunakan untuk pengenalan objek. Penelitian ini bertujuan untuk pengenalan objek pada citra buah menggunakan YOLO. Pada penelitian menggunakan sebanyak 300 gambar data dengan tiga kelas yaitu apel, jeruk dan pisang. Hasil penelitian menunjukan algoritma (YOLO) dapat mengenali objek pada citra buah dengan menggunakan *pre-trained weights* yang telah dilatih dengan nilai akurasi mAP sebesar 91%.

Kata kunci: *kecerdasan buatan, YOLO, pengenalan objek.*

OBJECT DETECTION TO DETECT FRUITS IMAGES USING YOLO METHOD

Abstract

The rapid development of science and technology in Artificial Intelligence today has brought very rapid changes in various aspects of life. Especially artificial intelligence is a computer or machine technology that has human-like intelligence. It's simply a smart instruction given to a program or machine, one of which is Object Detection to detect fruit images using You Look Only Once (YOLO). The method that can be used for object recognition in fruit images is deep learning. You Look Only Once (YOLO) is a Deep Learning model that can be used for object recognition. This study aims to identify objects in fruit images using YOLO. The research uses a dataset of 300 images with three classes, namely apples, oranges and bananas. The results show that the You Only Look Once (YOLO) algorithm can recognize objects in fruit images using pre-trained weights that have been trained with a mean Average Precision (mAP) value 91%.

Keywords: *artificial intelligence, YOLO, object recognition.*

1. PENDAHULUAN

Deteksi objek adalah teknik visi komputer yang bertujuan untuk mendeteksi objek seperti buah, mobil, dan manusia, dan masih banyak lagi. Objek umumnya dapat diidentifikasi baik dari gambar atau video. Secara khusus, deteksi objek menarik kotak pembatas di sekitar objek yang terdeteksi ini, yang memungkinkan kita menemukan lokasi objek tersebut, Secara garis besar, deteksi objek dapat dipecah menjadi pendekatan berbasis Machine Learning dan pendekatan berbasis *Deep Learning* [1]. Dalam pendekatan berbasis ML yang lebih tradisional, teknik visi komputer digunakan untuk melihat berbagai fitur gambar, seperti histogram warna atau tepi, untuk mengidentifikasi

kelompok piksel yang mungkin dimiliki suatu objek. Fitur-fitur ini kemudian dimasukkan ke dalam model regresi yang memprediksi lokasi objek beserta labelnya [2].

Sampai sekarang masih sedikit teknologi pengolahan citra yang diterapkan untuk dapat mengenali objek khususnya objek buah-buahan. Berdasarkan Maarten Christenhusz pada publikasi yang berjudul *"The number of known plant species in the world and its annual increase*" jumlah tumbuhan yang ada didunia berkisar 374.000 tumbuhan dimana lebih dari 1000 tumbuhan tersebut merupakan tumbuhan yang menghasilkan buah. Karena begitu banyaknya jumlah buah yang ada maka kemungkinan terdapatnya buah-buahan yang memiliki kemiripan fisik baik berupa bentuk maupun warna yang dapat membuat keliru dalam mengenali buah-buahan yang ada. Oleh karena itu, pembuatan aplikasi ini dapat membantu untuk pengenalan buah-buahan yang ada [3].

Beberapa penelitian dengan menggunakan metode *Deep Learning* juga diusulkan oleh beberapa peneliti untuk pengenalan objek makanan cepat saji pada video dan *real time* webcam menggunakan Metode *You Only Look Once* (YOLO), Algoritma CNN dan YOLO [4]. *Convolutional Neural Network* (CNN) [4] Metode (YOLO) menjadi salah satu metode yang cepat dan akurat dalam melakukan pendeteksian objek. Metode ini mampu melakukan deteksi objek hingga 2 kali lebih cepat daripada algoritma yang lain. Pada penelitian ini, penulis menggunakan metode YOLOv3 karena memiliki beberapa peningkatan dalam mendeteksi objek dan nilai akurasi yang lebih tinggi daripada versi sebelumnya. Dengan menggunakan metode YOLO, Dari latar belakang masalah tersebut peneliti tertarik untuk mengkaji lebih dalam tentang Object Detection untuk mendeteksi citra pada buah-buahan menggunakan metode *You Look Only Once*(YOLO). Sehingga pada penelitian ini bermaksud untuk melakukan penelitian dengan Judul "*Object Detection* untuk mendeteksi citra pada buah-buahan menggunakan metode *You Look Only Once* (YOLO)".

2. METODE PENELITIAN

2.1. Data Yang digunakan

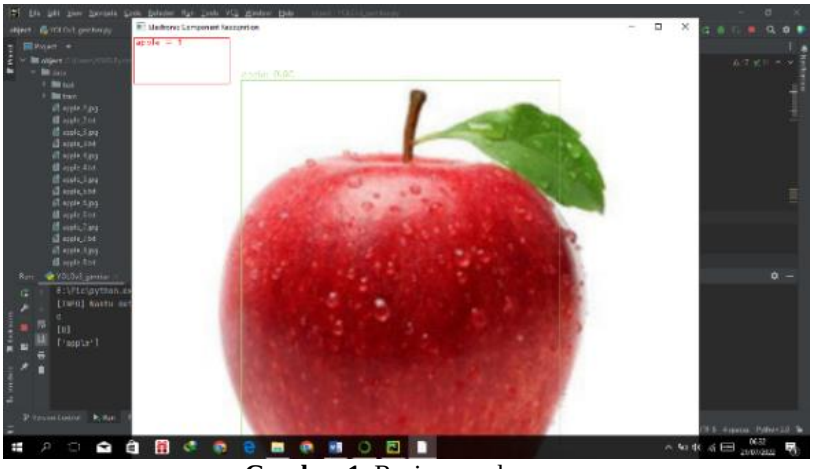
Data yang digunakan pada penelitian ini yaitu data *image* dengan ekstensi jpg yang diperoleh dari sumber berbagi dataset yaitu kaggle, data dapat di download dari link https://www.kaggle.com/datasets/mbkinaci/fruits-images-for-object-detection. Data image yang diperoleh dibagi menjadi dua dataset yaitu data training dan data testing dimana data training berjumlah 240 image dan data testing sebanyak 60 image.

a) Analisis Sistem

Pada tahap ini merupakan proses pengolahan citra buah-buahan yang telah diperoleh. Proses yang dilakukan terdiri dari resize, pelabelan objek, dan training data.

a) *Resize*

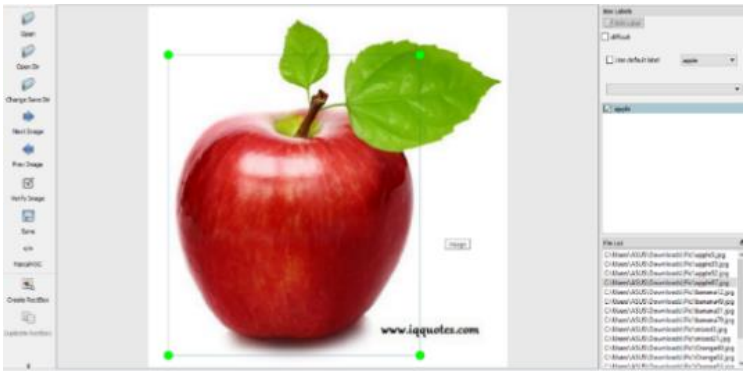
Pada tahap awal proses *preprocessing* yaitu dilakukan *resizing* untuk mengubah ukuran citra dengan memperkecil ukuran citra secara horizontal dan/atau *vertical*, dengan ukuran *image* yaitu 490 x 360. Hal ini dilakukan untuk membuat proses *training* lebih ringan dan cepat.



Gambar 1. Resize gambar

b) Pelabelan Objek

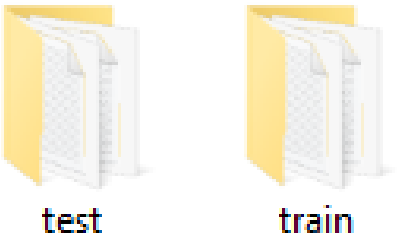
Pada tahap ini dilakukan pelabelan dan pemberian *bounding box* terhadap objek pada setiap *image* yang telah diambil hal ini bertujuan untuk memberikan ciri dari masing-masing objek dan melatih sistem untuk mengenali dan memberikan *bounding box* kepada setiap objek. Hasil dari pelabelan objek ini adalah file .txt yang berisi informasi tentang kelas dan *bounding box* objek dimana setiap *image* akan memiliki satu file .txt. Seperti gambar 3.2 dan 3.3.



Gambar 2. Proses pelabelan menggunakan Labeling

2.2. *Training data*

Tahap *training* adalah tahap melatihkan data yang telah digenerate untuk dipelajari oleh sistem. Algoritma yang digunakan adalah *You Only Look Once* (YOLO). Pada penelitian ini penulis menggunakan 300 *images*, dimana 240 *images* sebagai *training images* dan 60 *images* sebagai *data test.*

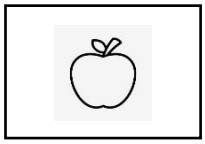
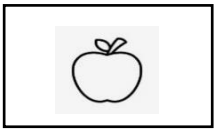


Gambar 3. gambar *dataset training* dan testing

2.3. Perancangan Antarmuka

Pada bagian ini akan dibahas rancangan antarmuka aplikasi ini. Pada penelitian ini terdapat 2 *scene* perancangan antarmuka yaitu *Object Detection* dan *Show Process* [5]. Adapun rancangan antarmuka pada penelitian ini yaitu dapat dilihat pada Tabel 1.

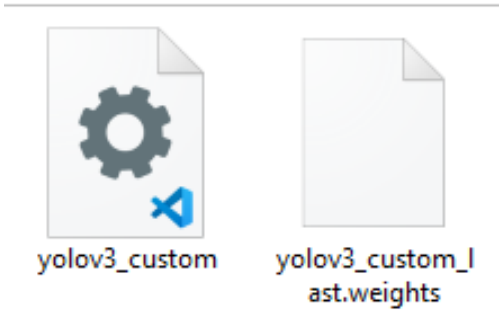
Tabel 1. Rancangan Antarmuka

Rancangan Antarmuka	Keterangan
	Tampilan ini merupakan tampilan dari proses object detection, menu yang tersedia, yaitu : a. *Show Process*, dapat menunjukkan process yang sedang dilakukan oleh sistem.
	Tampilan ini merupakan dimana user dapat melihat process yang berjalan di background dengan cara mengklik *run.*

2.4. Instalasi Software

a) *Download file weight* dan *file configuration* YOLO

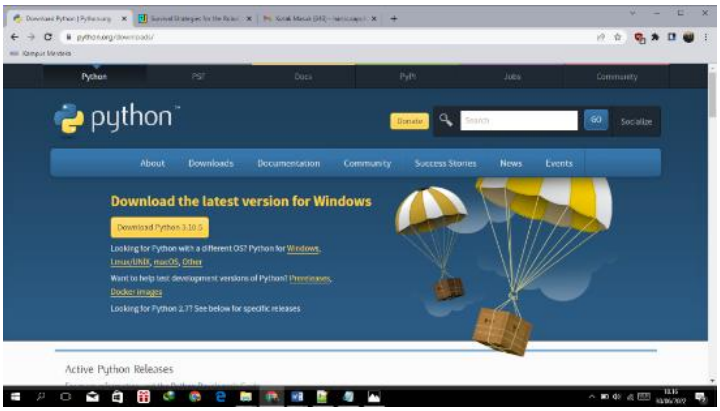
Langkah selanjutnya adalah melakukan *download file weight* dan *file configuration* YOLOv3 yang sudah selesai di-*training* seperti pada Gambar 3.13. File ini adalah inti dari algoritma YOLO untuk mendeteksi objek dan *file configuration* adalah pengaturan dari algoritma YOLO.



Gambar 4. *File weight* dan cfg YOLO

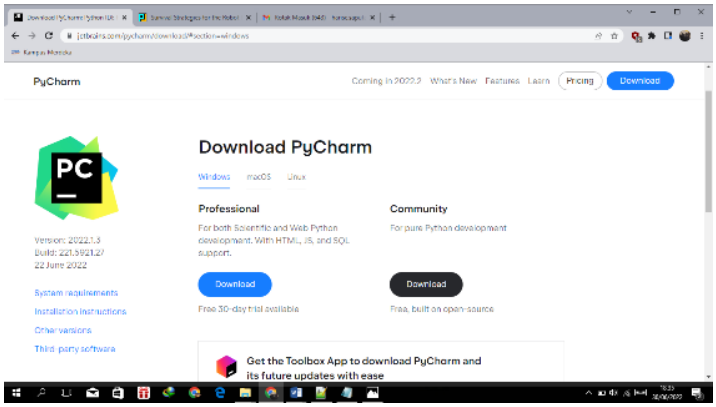
c) Instalasi Python

Pada penelitian ini, digunakan bahasa python untuk melakukan pendeteksian.



Gambar 5. Tampilan *website* python

d) Instalasi Pycharm Digunakan pycharm untuk deployment aplikasi object deteksi.



Gambar 6. Tampilan website *Pycharm*

e) Instalasi OpenCV

Melakukan instalasi openCV disini digunakan openCV dengan kode seperti di gambar 8.

```
pip install opencv-python
```

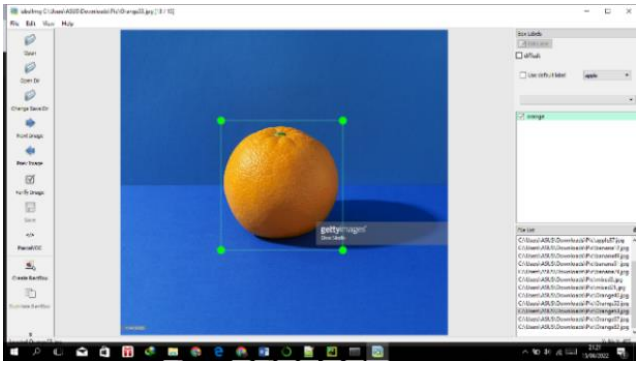
Gambar 7. Kode program instalasi *OpenCV*

3. HASIL DAN PEMBAHASAN

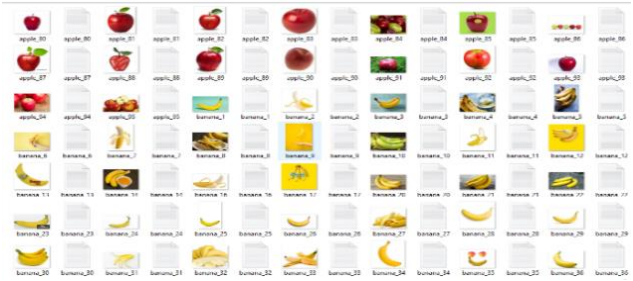
Data yang digunakan dalam proses *training* yang berjumlah 300 data dimana 80 % data ini digunakan untuk *training* data dan 20 % untuk testing data.

3.1 Anotasi Objek

Gambar 8. Contoh hasil pelabelan



Gambar 9. Proses anotasi *dataset*



Gambar 10. *Dataset* sudah diberikan label

```
1  0 0.422009 0.316667 0.352564 0.326667
2  0 0.599359 0.568889 0.275641 0.288889
3  0 0.191239 0.585556 0.369658 0.366667
4  0 0.554487 0.835556 0.301282 0.275556
5  0 0.887821 0.641111 0.224359 0.175556
```

Gambar 11. Hasil dari .txt file

3.2 Training Data

Setelah anotasi selesai dilakukan, langkah selanjutnya melakukan proses training. Proses ini bertujuan untuk melatih komputer dengan cara mengolah gambar dan anotasi yang sudah dibuat sehingga terbentuk pola atau karakteristik dari setiap kelas yang akan menjadi bahan pertimbangan komputer dalam mencapai sebuah keputusan atau prediksi [1], [6]. Pada bagian ini menggunakan pre-trained weights YOLOv3. Dengan menggunakan teknik

transfer learning. Transfer learning adalah suatu teknik yang menggunakan model yang telah di-training sebelumnya (pre-trained model) yang dapat digunakan untuk klasifikasi dataset baru tanpa harus melakukan training data dari awal. Proses transfer learning pada Darknet menggunakan data file, cfg file, dan pre-trained weigths. Data file berisi lokasi gambar yang digunakan untuk train dan test. Cfg file berisi bentuk jaringan yang digunakan untuk training, dan pre-trained weights berisi model weight yang telah dilatih sebelumnya pada jaringan YOLO.

Karena proses komputasi yang berat saat melakukan proses training, maka penulis menggunakan Google Colaboratory yang merupakan sebuah platform yang telah disediakan oleh Google. Ketika proses training sedang berjalan, dibutuhkan koneksi internet yang stabil agar runtime saat proses training tidak terputus. Proses training ini dilakukan menggunakan framework neural network Darknet-53 dengan konfigurasi seperti pada Tabel 4.1

Tabel 2 Konfigurasi pada Darknet-53

Jenis Konfigurasi	**Keterangan**
Load model	Darknet-53
Load Weight	Yolov3
OPENCN	1
GPU	1
CUDNN	1

Proses training menggunakan Darknet-53 sebagai load model dengan susunan layer seperti Gambar 2.8. dan YOLOv3 sebagai load weight dengan konfigurasi seperti pada Tabel 4.3. Jumlah batch menentukan jumlah gambar yang akan diproses sebelum network weight mengalami pembaharuan. Subdivision bertujuan untuk memproses sebagian kecil batch size sekaligus pada GPU. Max_batch merupakan batas iterasi dalam melakukan training yang didapatkan dengan persamaan. Ketika iterasi sudah mencapai 6000, maka proses training akan otomatis berhenti. Nilai max_batches ditentukan pada persamaan 4.1.

max $\_batches = jumlahclass$. 1078 (4.1)

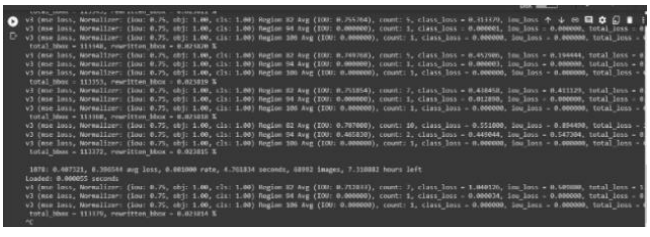
Height dan Weight merupakan dimensi gambar masukan yang akan dilatih. Classes merupakan jumlah kelas yang akan dideteksi. Nilai filter didapatkan pada persamaan 4.2.

$filter = (jumlahclass + 5) \times 3$ (4.2)

Nilai step didapatkan pada persamaan (4.3)

$steps$ =(40%max$_batches$),(45%max $\_batches$)

Proses *training* dan *avg loss* dapat dilihat pada Gambar 4.5



Gambar 12 Iterasi Training Data dan Avg loss

Gambar 13 menunjukkan proses *training* data telah mencapai seribu tujuh puluh delapan iterasi dengan waktu *training* sekitar satu jam tiga puluh menit hal ini dilakukan agar tingkat *Avg loss* pada model semakin kecil, proses ini menunjukkan tingkat *loss* pada pelatihan kecil 0.39.

Tabel 3. Konfigurasi pada weights YOLOv3

Jenis Konfigurasi	Keterangan
batch	64
Subdivisions	16
Width	416
Height	416
Max_batches	6000
Avg Loss	0.39
Steps	4000, 4500
Classes	3
Filter	24

3.3 Deteksi Buah dengan Algoritma You Only Look Once (YOLO)

Pada proses deteksi, setelah semua software yang dibutuhkan telah ter-instalasi dengan baik. Langkah selanjutnya adalah menggabungkan *file weight*, configuration, dan *dataset* pada satu folder.

a) Import Library

Menggunakan *library* cv2, numpy, glob, random dan time. *Library* cv2 berfungsi untuk melakukan *computer vision* task. *Library numpy* berfungsi untuk pengolahan data *numerical*. *Glob* berfungsi untuk mengetahui gambar apa yang akan kedeteksi.

```
import cv2
import numpy as np
import glob
import random
import time
```

Gambar 13. Kode program *Import Library*

b) Memuat *network* YOLOv3

Menggunakan *file weight*, cfg, dan folder data. *Weight file* adalah model yang sudah di- *training*, inti dari algoritma YOLO untuk mendeteksi objek. Cfg file adalah file konfigurasi, dimana semua pengaturan algoritma YOLO terdapat pada file tersebut. folder data adalah file yang berisi nama-nama objek yang dapat dideteksi menggunakan algoritma YOLO. Disini digunakan folder data yang berisikan tiga kelas yang akan dideteksi yaitu *apple, banana* dan *orange.* Setelah itu ada *images_path* untuk mengetahui gambar apa yang akan kita deteksi dan hasilnya itu akan mengeluarkan *bounding box* prediksinya [7].

```
net = cv2.dnn.readNet("yolov3_custom_last.weights", "yolov3_custom.cfg")

classes = ["apple", "banana", "orange"]


images_path = glob.glob("data/mixed_12.jpg") #ganti dengan direktori gambar anda

layer_names = net.getLayerNames()
output_layers = [layer_names[i - 1] for i in net.getUnconnectedOutLayers()]
colors = np.random.uniform(35, 255, size=(len(classes), 3))
random.shuffle(images_path)
for img_path in images_path:
    img = cv2.imread(img_path)
    img = cv2.resize(img, (490,360))
    height, width, channels = img.shape
```

Gambar 14. Kode program memuat network YOLOv3

Hasil keluaran *labels* yang akan muncul setelah menjalakan *code network* YOLOv3 diatas.

```
[1 3 2]
['apple', 'banana', 'orange']
```

Gambar 15. Hasil keluaran *labels*

c) Mengambil fungsi *blob* dari *image*

Data akan diolah menggunakan *library OpenCV* untuk diubah menjadi bentuk *blob* (*Binary Large Object*) dari *image*. Pada fungsi 'cv2.dnn.*blob*FromImage' berisi parameter *image* yang akan diproses melalui jaringan saraf dalam bentuk klasifikasi. *Blob* digunakan untuk mengekstrak fitur gambar dan mengubah ukurannya. Pada YOLO terdapat 3 ukuran, yaitu 320x320, 416x416, dan 609x609. Disini digunakan ukuran 416x416 agar proses deteksi memiliki kecepatan dan akurasi yang seimbang.

```
blob = cv2.dnn.blobFromImage(img, 0.00392, (416, 416), (0, 0, 0), True, crop=False)
```

Gambar 16. Kode program mengambil fungsi *blob* dari *image*

d) Menerapkan *Forward Pass*

Mengimplementasi *forward pass* dengan '*blob*' melalui lapisan keluaran. Lalu, menghitung waktu deteksi yang dibutuhkan.

```
    print("[INFO] YOLO {:.6f} seconds".format(end - start))
    end = time.time()
    outs = net.forward(output_layers)
    start = time.time()
net.setInput(blob)
```

Gambar 17. Kode program menerapkan *Forward Pass*

e) Mendapatkan *bounding boxes*

Pada langkah ini dilakukan ekstraksi seluruh informasi objek yang dideteksi dan menampilkannya pada layar. ClassIDs untuk memberikan label kelas objek yang terdeteksi. *Confidences* memberikan nilai keyakinan pada YOLO untuk suatu objek, nilai keyakinan (*confidence*) dari objek yang terdeteksi bernilai 0 sampai 1. *Bounding_boxes* untuk kotak pembatas yang mengelilingi objek. Kemudian if *confidence* > 0.3: maksudnya yaitu ketika nilai *confidence* atau nilai keyakinan lebih dari 0.3 atau lebih dari 30 persen maka akan dinyatakan terdeteksi.

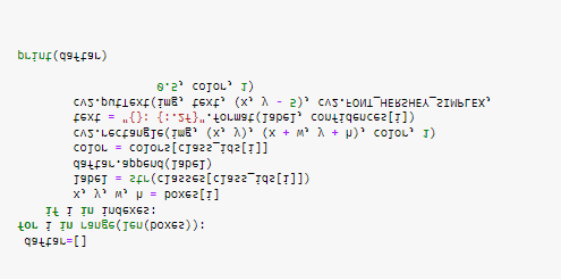
```
class_ids = []
confidences = []
boxes = []
for out in outs:
    for detection in out:
        scores = detection[5:]
        class_id = np.argmax(scores)
        confidence = scores[class_id]
        if confidence > 0.3:
            print(class_id)

            center_x = int(detection[0] * width)
            center_y = int(detection[1] * height)
            w = int(detection[2] * width)
            h = int(detection[3] * height)
            x = int(center_x - w / 2)
            y = int(center_y - h / 2)
            boxes.append([x, y, w, h])
            confidences.append(float(confidence))
            class_ids.append(class_id)
font = cv2.FONT_HERSHEY_PLAIN
indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)
unique, counts = np.unique(class_ids, return_counts=True)
```

Gambar 18. Kode program mendapatkan *bounding boxes*

f) Mendapatkan *bounding boxes* dengan *labels*

Menggunakan perintah if untuk memeriksa jika setidaknya ada satu objek yang terdeteksi setelah NMS. Dengan mendapatkan *bounding box* saat ini, lebar, dan tingginya. Kemudian mempersiapkan warna untuk *bounding box* saat ini dan mengkonversi dari *array numpy*. Setelah itu, mempersiapkan teks dengan label dan nilai *confidence* untuk *bounding box* saat ini dan menampilkan hasilnya.

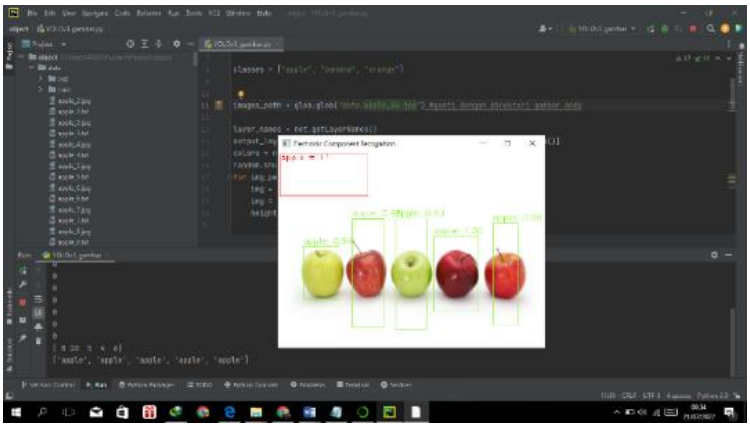


Gambar 19. Kode program mendapatkan *bounding boxes* dengan *labels*

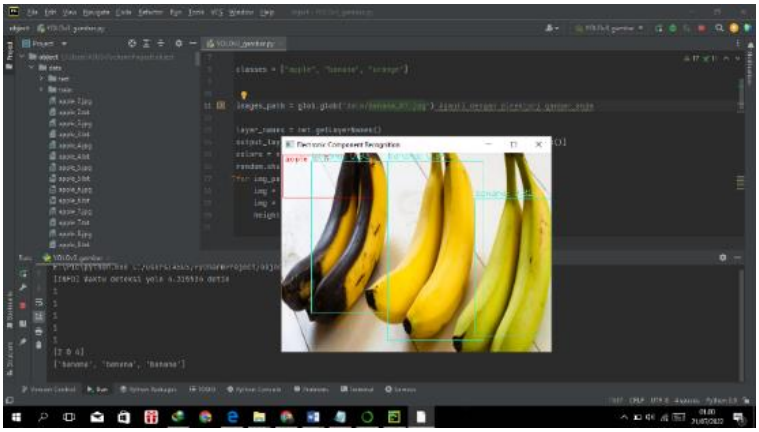
Tabel 4. Pengujian hasil *data test*

Load model		Yolov3 custom
Apple_86	AP	99%
	TP	83
	TN	8
	FP	2
Banana_87	AP	94%
	TP	57
	TN	21
	FP	10
	FN	1
Orange_89	AP	98%
	TP	71
	TN	7
	FP	11
Rata-rata waktu deteksi		4 detik
Recall		0,99
Precision		0,90
mAP		0,91%

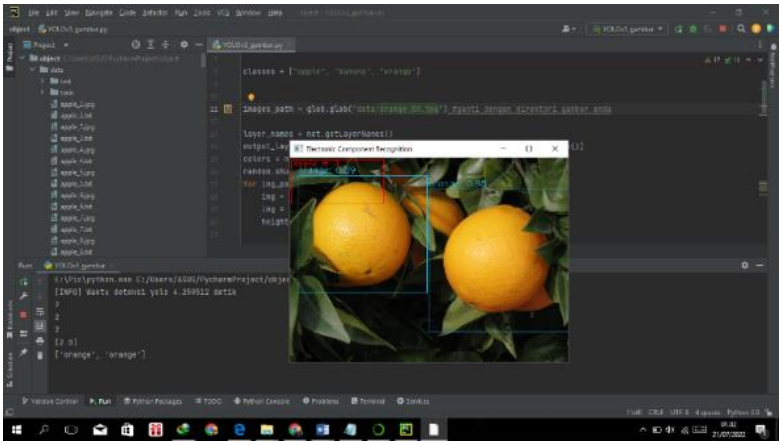
Pada Tabel 4 menunjukkan perbandingan akurasi waktu beberapa metode untuk pendeteksian objek. Metode YOLOv3 dengan masukan 608 menghasilkan nilai mAP dua persen lebih rendah dari metode FPN FRCN [8], yaitu 57,9% untuk YOLOv3-608,dan 59,1% untuk FPN FRCN, tetapi metode YOLOv3-608 dapat memproses masukan lebih cepat selama 51 ms dibandingkan FPN FRCN selama 172 ms. mAP adalah suatu metrik untuk mengukur akurasi objek detektor dan semakin tinggi mAP berarti pendeteksi objek semakin akurat. Beberapa metode yang lain menghasilkan nilai mAP yang baik, namun waktu pemrosesan lebih lama dibandingkan metode YOLOv3 dengan masukan 608x608, 416x416, dan 320x320 (Redmon & Farhadi, 2018). Sedangkan pada penelitian ini menggunakan *pre- trained weights* model YOLOv3 dengan teknik *transfer learning* menghasilkan nilai mAP sebesar 91% dengan waktu pemrosesan selama 4 detik. Hal ini menunjukkan metode YOLOv3 bekerja dengan baik untuk melakukan deteksi objek.



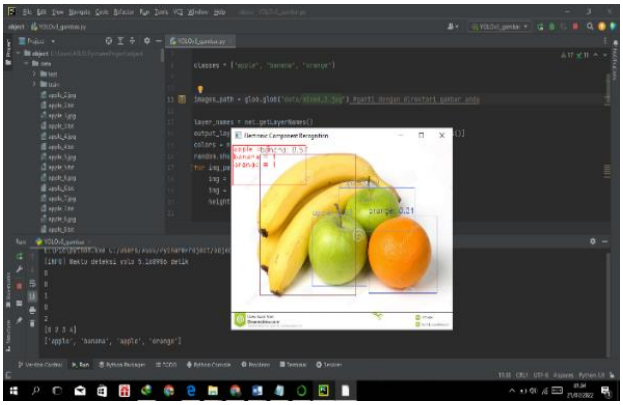
Gambar 20. Hasil deteksi apple_86



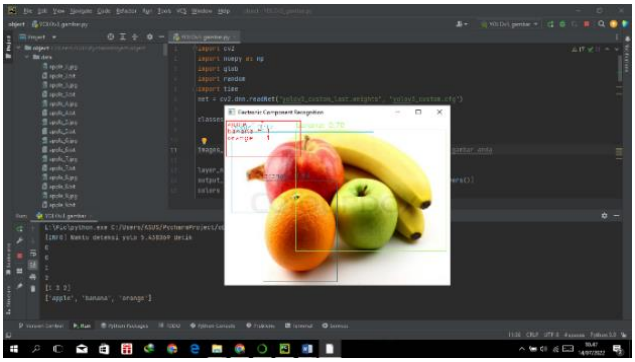
Gambar 21. Hasil deteksi Banana_87



Gambar 22. Hasil deteksi Orange_89



Gambar 23. Hasil deteksi mixed_2



Gambar 24. Hasil deteksi mixed_13

4. KESIMPULAN

Dari hasil penelitian yang sudah dilakukan, dapat ditarik kesimpulan sebagai berikut : Hasil dari pendeteksian buah buahan pada kelas *apple, banana*, dan *orange* menggunakan algoritma YOLO dapat dinilai bekerja dengan baik. Hasil dari deteksi menggunakan masukan berupa gambar menghasilkan nilai *confidence* yang berbeda-beda. Hal ini disebabkan oleh kualitas gambar yang berbeda beda. Berdasarkan tabel perbandingan kecepatan akurasi dan waktu deteksi dari beberapa model deteksi objek, YOLOv3 mampu mendeteksi dengan akurasi yang baik dan waktu deteksi yang cepat. Hasil *training* data pada setiap kelas menunjukan sistem telah dapat mendeteksi objek dengan baik. Nilai *confidence* pada kelas *banana*_87 sebesar 94%, pada *apple*_86 sebesar 99%, dan *orange*_89 sebesar 98%. Sedangkan nilai akurasi mAP sebesar 91% dengan waktu deteksi selama 4 detik.

DAFTAR PUSTAKA

[1] Y. I. Kurniawan, E. Soviana, and I. Yuliana, "Merging Pearson Correlation and TAN-ELR algorithm in recommender system," *AIP Conf. Proc.*, vol. 1977, no. August 2019, 2018, doi: 10.1063/1.5042998.

[2] E. A. Shams and A. Rizaner, "A novel support vector machine based intrusion detection system for mobile ad hoc networks," *Wirel. Networks*, vol. 24, no. 5, 2018.

[3] P. R. Kannari, N. S. Chowdary, and R. L. Biradar, "An anomaly-based intrusion detection system using recursive feature elimination technique for improved attack detection," *Theor. Comput. Sci.*, vol. 931, 2022.

[4] D. Hananta Firdaus, B. Imran, L. Darmawan Bakti, and E. Suryadi, "Klasifikasi Penyakit Katarak Pada Mata Menggunakan Metode Convolutional Neural Network (Cnn) Berbasis Web," *J. Kecerdasan Buatan dan Teknol. Inf.*, vol. 1, no. 3, pp. 18–26, 2022.

[5] Y. Guo, S. Han, Y. Li, C. Zhang, and Y. Bai, "K-Nearest Neighbor combined with guided filter for hyperspectral image classification," *Procedia Comput. Sci.*, vol. 129, pp. 159–165, 2018, doi: 10.1016/j.procs.2018.03.066.

[6] A. Kumar and H. Singh, "A REVIEW ON SOFTWARE TESTING AND ITS METHODOLOGY By," *i-manager's J. Softw. Eng.*, vol. 3, no. 1, p. 2013, 2013.

[7] J. Jabez, G. Shanmugam, V. S, J. A. Mayan, and S. Srinivasulu, "Anomaly Detection by Using CFS Subset and Neural Network with WEKA Tools," *Inf. Commun. Technol. Intell.*, 2019.

[8] Y. I. Kurniawan, A. Rahmawati, N. Chasanah, and A. Hanifa, "Application for determining the modality preference of student learning," *J. Phys. Conf. Ser.*, vol. 1367, no. 1, 2019, doi: 10.1088/1742-6596/1367/1/012011.