9 772775 412005
JURNAL Sintaks Logika (JSilog)
E-ISSN : 2775-412X
https://jurnal.umpar.ac.id/index.php/sylog

DETEKSI OBJEK PADA GAMBAR MENGGUNAKAN ALGORITMA YOLO (YOU ONLY LOOK ONCE)

Muhammad Zainal[1*], Ahmad Selao[2], Muhammad Ilham Alamsyah[3]

[1,2,3]Program Studi Teknik Infotmatika, Universitas Muhammadiyah Parepare, Indonesia
zainalmuh045@gmail.com, ahmadselao23@gmail.com, masterviland10@gmail.com

Informasi Artikel

Riwayat Artikel:

Dikirim *Author* : -
Diterima Redaksi : -
Revisi *Reviewer*: -
Diterbitkan *online*: -

Keywords:

Object Detection; Computer Vision; Darknet; YOLO (You Only Look Once) Algorithm.

ABSTRACT

Object detection is a phenomenon in computer vision that involves detecting various objects in an image. Some of the objects detected include humans, animals, plants, buildings, vehicles, home appliances, etc. Object detection aims to identify all target objects in the target image and determine the category of position information to achieve machine vision understanding. Object detection has benefits including being able to analyze road traffic, being able to be smart parking management, being able to improve the security system on CCTV cameras, and many other benefits. Many approaches have been proposed to solve this problem, mainly inspired by computer vision and deep learning methods. However, existing approaches always underperform to detect small, dense objects, and even fail to detect objects. From these problems, the YOLO (You Only Look Once) method is present as one of the most successful approaches to solving common problems in object detection. Introduced by Joseph Redmon et al., the YOLO method became phenomenal because it was able to detect objects in real time. The results of this study have succeeded in building a test application to detect objects using the YOLO algorithm. Completion of this process using models and trained weights. Based on the test results of 10 test datasets on the Pink Yarn object, it has succeeded in detecting objects with an average accuracy value of 59.72%.

Kata kunci:

Deteksi Objek; Computer Vision; Darknet; Algoritma YOLO (You Only Look Once).

ABSTRAK

Deteksi objek adalah fenomena dalam *computer vision* yang melibatkan pendeteksian berbagai objek dalam gambar. Beberapa objek yang terdeteksi antara lain manusia, hewan, tumbuhan, bangunan, kendaraan, peralatan rumah dll. Deteksi objek bertujuan untuk mengidentifikasi semua objek target dalam gambar target dan menentukan kategori informasi posisi untuk mencapai pemahaman *machine vision*. Deteksi objek memiliki manfaat diantaranya mampu menganalisis lalu lintas jalan, mampu menjadi manajemen parkir cerdas, mampu meningkatkan sistem keamanan pada kamera CCTV, dan masih banyak lagi manfaat lainnya. Banyak pendekatan telah diusulkan untuk memecahkan masalah ini, terutama terinspirasi oleh metode *computer vision* dan *deep learning*. Namun, pendekatan yang ada selalu berkinerja buruk untuk mendeteksi objek kecil, padat, dan bahkan gagal dalam mendeteksi objek. Dari permasalahan tersebut, Metode YOLO (You Only Look Once) hadir sebagai salah satu pendekatan yang paling berhasil memecahkan masalah umum dalam melakukan pendeteksian objek. Diperkenalkan oleh Joseph Redmon dkk., Metode YOLO menjadi sangat fenomenal karena mampu mendeteksi objek secara real time. Adapun hasil dari penelitian ini telah berhasil membangun aplikasi pengujian untuk mendeteksi objek menggunakan algoritma YOLO. Penyelesaian proses ini menggunakan model dan *trained weight*. Berdasarkan hasil uji coba 10 dataset pengujian pada objek Benang berwarna Pink telah berhasil melakukan pendeteksian objek dengan nilai rata-rata akurasi 59.72%.

Penulis Korespondensi:

Muhammad Zainal,
Program Studi Teknik Informatika,
Universitas Muhammadiyah Parepare,
Jl. Jendral Ahmad Yani KM.6 Kota Parepare, Indonesia
Email: zainalmuh045@gmail.com

This is an open access article under the CC BY-SA license.

9 772775 412005
JURNAL Sintaks Logika (JSilog)
E-ISSN : 2775-412X
https://jurnal.umpar.ac.id/index.php/jsilog

I. PENDAHULUAN

Manusia melihat sebuah gambar dan langsung mengetahui objek apa saja yang ada di dalam gambar tersebut, dimana mereka berada, dan bagaimana mereka berinteraksi. Manusia melihat cepat dan akurat, yang memungkinkan untuk melakukan tugas-tugas kompleks seperti mengemudi dengan sedikit pemikiran sadar. Algoritma yang cepat dan akurat untuk pendeteksian objek akan memungkinkan komputer mengemudikan mobil tanpa sensor khusus, memungkinkan perangkat bantu untuk menyampaikan secara real-time dll[7].

Banyak pendekatan yang telah dilakukan seperti *fast R-CNN*, *Retina-Net*, dan *Single-Shot Multibox Detector* (SSD), namun algoritma YOLO (You Only Look Once) merupakan pendekatan yang paling berhasil memecahkan masalah umum dalam melakukan pendeteksian objek[3].

Banyak penelitian terdahulu yang berkaitan dengan deteksi objek menggunakan algoritma YOLO sebagai penerapan dalam mendeteksi objek tertentu seperti penelitian yang dilakukan oleh Mada Lazuard Nazilly, Basuki Rahmat dan Eva Yulia Puspaningrum Dengan judul "Implementasi Algoritma Yolo (You Only Look Once) Untuk Deteksi Api" yang memanfaatkan algoritma YOLO dalam mendeteksi api pada citra. Penelitian ini dilakukan dengan mengembangkan sistem deteksi api berbasis drone dengan menerapkan algoritma YOLO[5]. Penelitian lainnya yang dilakukan oleh Sisco Jupiyandi, Fadil Rizqullah Saniputra, Yoga Pratama, Muhammad Robby Dharmawan dan Imam Chollissodin dengan judul "Pengembangan Deteksi Citra Mobil Untuk Mengetahui Jumlah Tempat Parkir Menggunakan Cuda Dan Modified Yolo" yang memanfaatkan algoritma YOLO dalam mendeteksi mobil pada citra. Penelitian ini dilakukan dengan menggunakan penerapan pemograman GPU yang dikombinasikan dengan algoritma YOLO untuk mendeteksi citra mobil dan jumlah mobil secara paralel[2]. Dan masih banyak lagi penelitian serupa lainnya.

Adapun tujuan dari penelitian ini adalah untuk mengetahui seberapa akurat algoritma YOLO dalam melakukan pendeteksian objek pada citra.

II. METODOLOGI PENELITIAN

A. Jenis Penelitian

Dalam penelitian ini menggunakan metode deskripsif yang menggambarkan fakta-fakta dan informasi secara sistematis, faktual dan akurat. Penelitian ini dilakukan melalui internet yang dapat memberikan sumber data dan pengetahuan mengenai sistem yang diteliti, kemudian mencocokkan dengan kemungkinan yang terjadi dalam usaha penyelesaian masalah.

B. Tempat dan Waktu Penelitian

Lokasi penelitian di Kota Parepare, Kecamatan Ujung, Kelurahan Lapadde dan waktu penelitian akan dilaksanakan kurang lebih +4 bulan pada tahun 2021.

C. Alat dan Bahan Penelitian

Untuk melakukan proses penelitian dalam mendeteksi objek pada gambar, maka diperlukan perangkat keras dan perangkat lunak guna untuk mendukung kegiatan penelitian. Berikut ini merupakan perangkat keras dan perangkat lunak yang digunakan dalam mendeteksi objek pada gambar.

1). Perangkat Keras

Perangkat keras yang digunakan dalam melakukan proses penelitian deteksi objek adalah sebagai berikut:

Tabel 1. Spesifikasi Perangkat Keras

Spesifikasi	
Merek laptop	Asus A456U
Processor laptop	Intel(R) Core(TM) i7-7500U, CPU @ 3.5 GHz
Ram laptop	8 (8192) GB

2). Perangkat Lunak

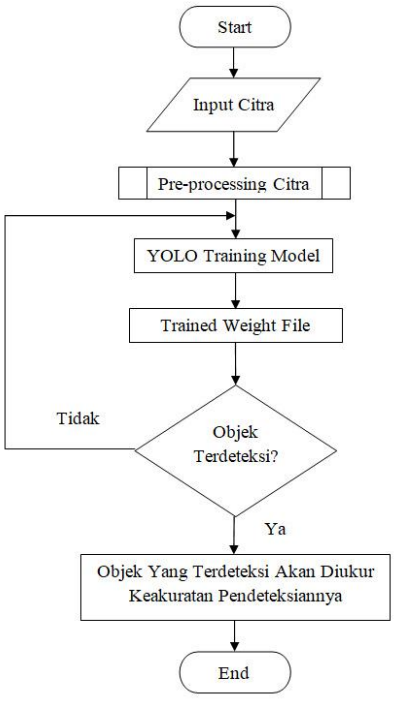
Perangkat lunak yang digunakan dalam melakukan proses penelitian deteksi objek adalah sebagai berikut:

Tabel 2. Spesifikasi Perangkat Lunak

Spesifikasi	
Sistem operasi	Windows 10
Tool pemrograman	Google Colab, Anaconda
Bahasa pemrograman	Python

D. Diagram Perancangan Proses

Diagram tahapan perancangan proses ditunjukan pada Gambar Berikut :



Gambar 1. Diagram Alur Perancangan Proses

Ringkasan perancangan proses adalah sebagai berikut:
1). Input citra yang telah diperoleh secara manual menggunakan smartphone
2). Pra-processing citra yang terdiri dari kelas dan pelabelan citra.
3). YOLO (You Only Look Once) training model merupakan proses konfigurasi dan pelatihan pada citra.
4). Trained weight file yaitu hasil output dari proses pelatihan yang berisi bobot berformat .weight.
5). Jika objek terdeteksi maka objek yang terdeteksi akan diukur keakuratan pendeteksiannya sedangkan jika objek tidak terdeteksi maka citra perlu dilakukan proses pelatihan ulang.

E. Metode Pengumpulan Data

1.) Studi Pustaka

Mencari referensi yang berhubungan dengan topik yang akan dibuat. Pencarian referensi akan dicari di buku-buku, internet dan hasil penelitian maupun karya tulis ilmiah.

2). Pengamatan (Observasi)

Pengamatan adalah metode pengumpulan data dengan cara mengadakan tinjauan secara langsung ke objek yang diteliti. Untuk mendapatkan data yang bersifat nyata dan meyakinkan maka penulis melakukan pengamatan.

III. HASIL DAN PEMBAHASAN

A. *Perancangan Data*

1). Data Awal

Sebelum melakukan proses pelatihan dan proses pengujian, langkah pertama yang perlu dilakukan adalah mempersiapkan data citra yang dibutuhkan pada penelitian ini. Berdasarkan batasan masalah, pengumpulan data citra diperoleh secara manual menggunakan *smartphone*, sehingga keterangan objek untuk setiap citra perlu dibangun terlebih dahulu. Keterangan yang dimaksud merupakan informasi tentang letak objek dan kelas objek yang akan dideteksi dan diklasifikasi. Keterangan ini berfungsi sebagai target atau acuan untuk memperoleh bobot pada saat proses pelatihan data dan menjadi pembanding nilai output pada saat proses pengujian[5].

2). Pre-trained Weights

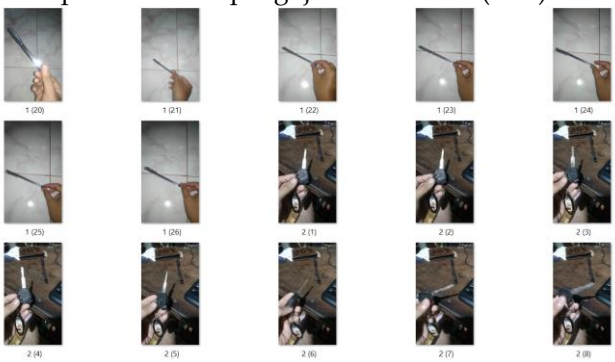
Dalam proses pelatihan menggunakan model dari *darknet open source*, diperlukan bobot awal dari sumber yang sama. Dalam pelatihan ini bobot awal yang digunakan bukan data random atau acak, tetapi data yang dilatih sebelumnya pada jaringan YOLO (You Only Look Once) untuk data citra. Data *pre-trained Weights* pada penelitian ini menggunakan model *yolov4-tiny* yang di unduh dari https://github.com/AlexeiAB/darknet/ dengan format file .weights.

B. Rancangan Proses

Secara umum proses deteksi objek pada gambar menggunakan algoritma YOLO dapat dilihat pada diagram alur dalam Gambar 1. Rincian tentang perancangan proses deteksi objek pada gambar menggunakan algoritma YOLO ini yaitu sebagai berikut.

1). Input Citra

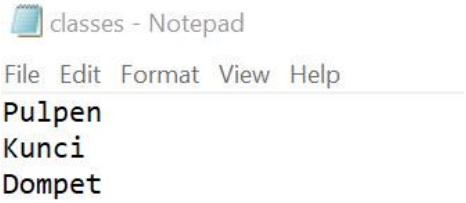
Data citra yang diinput merupakan citra yang diperoleh secara manual menggunakan *smartphone*. Jumlah data citra yang perlu dikumpulkan untuk proses pelatihan dan pengujian minimal 1 (satu) citra.



Gambar 2. Citra yang diinput dari *Smartphone*

2). Pre-processing Citra

Pada tahap ini dilakukan *pre-processing* citra yang terdiri dari kelas dan pelabelan pada citra, Kelas berjumlah sesuai dengan jenis citra yang akan dideteksi. Setelah menentukan kelas, citra yang sudah diinput akan masuk ke proses pelabelan citra[4].



Gambar 3. Kelas untuk pelabelan

Pelabelan citra adalah tahap awal yang dimana setiap data citra yang akan diberikan label mempunyai tujuan untuk menyimpan informasi pada citra. Proses pelabelan citra yaitu dengan cara memberikan *bounding box* serta *classes* pada objek citra yang tersorot[10]. Pelabelan citra menggunakan *LabelImg* yang merupakan salah satu tools atau modul *Python*.



Gambar 4. Pelabelan citra

Hasil dari pelabelan citra akan menghasilkan *output file* yang berisikan urutan objek dan koordinat (x, y, width, weight) pada citra[10].



Gambar 5. Hasil pelabelan citra

3). YOLO training model

Setelah dilakukan *pre-processing* citra, Selanjutnya akan dilakukan proses konfigurasi jaringan YOLO. Unduh Darknet terlebih dahulu dan aktifkan GPU dan depency lainnya pada perangkat, seperti OpenCV dan *library* lainnya untuk melakukan *build* pada Darknet yang akan digunakan untuk melakukan proses *training*[6].

```
[ ]  # unduh darknet
     !git clone https://github.com/AlexeyAB/darknet
```

Gambar 6. Unduh Darknet

```
[ ]  # mengubah makefile untuk mengaktifkan GPU,CUDNN,CUDNN_HALF,LIBSO dan OPENCV
     %cd /content/darknet/
     !sed -i 's/OPENCV=0/OPENCV=1/' Makefile
     !sed -i 's/GPU=0/GPU=1/' Makefile
     !sed -i 's/CUDNN=0/CUDNN=1/' Makefile
     !sed -i 's/CUDNN_HALF=0/CUDNN_HALF=1/' Makefile
     !sed -i 's/LIBSO=0/LIBSO=1/' Makefile
```

Gambar 7. Mengaktifkan GPU,cuDNN dan openCV

```
[ ]  # build darknet
     !make
```

Gambar 8. *Build* Darknet

Setelah melakukan *build* pada Darknet, selanjutnya terlebih dahulu melakukan konfigurasi dan pembuatan model pada modul. Untuk melakukan proses *training* pada data, melakukan pengaturan pada *batch* sebanyak 64 *batch* dan *subdivisions* sesuai pada perangkat yang cocok dan mengubah semua ukuran pada citra menjadi 416x416 sesuai dengan ketentuan pada YOLO [7].

```
batch=64
subdivisions=16
width=416
height=416
```

Gambar 9. Mengatur *batch*, *subdivisions*, *width*, dan *height*

Banyaknya *max batches* yang diperlukan harus ditentukan. Jumlah *max batches* dalam ketentuan YOLO adalah 2000*jumlah kelas yang akan di proses. Kemudian melakukan penentuan dalam akurasi pada pelatihan, yaitu 80%—90% dari total *max batches* sesuai ketentuan YOLO [7].

```
max_batches = 2000
policy=steps
steps=1600,1800
scales=.1,.1
```

Gambar 10. Mengatur *max batches* dan *steps*

Selanjutnya melakukan konfigurasi jumlah pada kelas (*classes*) dan *filtering* ((Jumlah Kelas + width + height + x + y + confidence) * num)) pada YOLO yang dapat dilihat pada Gambar 11. Cara menentukan jumlah kelas (*classes*) dan *filters* adalah sebagai berikut:

filters = (jumlah kelas + 5) * 3
= (jumlah kelas + width + height + x + y + confidence) * num
= (jumlah kelas + 1 + 1 + 1 + 1 + 1) * num
= (jumlah kelas + 5) * num
num = jumlah mask dalam layer YOLO
= (jumlah kelas + 5) * 3 [7].

```
[convolutional]
size=1
stride=1
pad=1
filters=18
activation=linear

[yolo]
mask = 1,2,3
anchors = 10,14,  23,27,  37,58,  81,82,  135,169,  344,319
classes=1
```

Gambar 11. Mendefinisikan jumlah kelas dan *filtering*

Setelah melakukan konfigurasi pada *filtering* dan jumlah kelas, selanjutnya masuk ke dalam tahap memasukkan *backup path* dan jumlah kelas yang akan digunakan untuk proses *training*. Gambar 12 adalah jumlah kelas yang akan dilakukan proses pelatihan dan file *backup* yang digunakan untuk menyimpan hasil output dari proses *training*.

```
classes = 1
train  = data/train.txt
valid  = data/test.txt
names = data/obj.names
backup = /mydrive/yolov4-tiny/training
```

Gambar 12. Mendefinisikan *backup path* dan jumlah kelas

Gambar 13 adalah file konfigurasi model terakhir yang diperlukan sebelum memulai proses *training*. File konfigurasi model ini berfungsi untuk menahan *path* relatif pada semua citra dalam proses *training*. Hal ini dilakukan agar iterasi pada setiap citra dengan ekstensi jpg memproses setiap citra menjadi data *train*. Yang kemudian melakukan proses *generate* pada file untuk memuat setiap *path* pada citra yang diiterasikan.

```
import glob, os

# Current directory
current_dir = os.path.dirname(os.path.abspath(__file__))

print(current_dir)

current_dir = 'data/obj'

# Percentage of images to be used for the test set
percentage_test = 10;

# Create and/or truncate train.txt and test.txt
file_train = open('data/train.txt', 'w')
file_test = open('data/test.txt', 'w')

# Populate train.txt and test.txt
counter = 1
index_test = round(100 / percentage_test)
for pathAndFilename in glob.iglob(os.path.join(current_dir, "*.jpg")):
    title, ext = os.path.splitext(os.path.basename(pathAndFilename))

    if counter == index_test:
        counter = 1
        file_test.write("data/obj" + "/" + title + '.jpg' + "\n")
    else:
        file_train.write("data/obj" + "/" + title + '.jpg' + "\n")
        counter = counter + 1
```

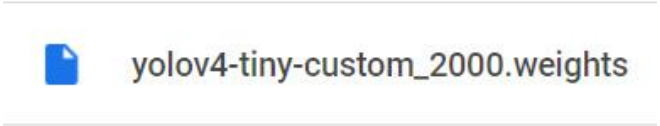
Gambar 13. Generate_train.py

Setelah semua konfigurasi model telah dikonfigurasi dan dibuat, Selanjutnya melakukan proses *training* data citra pada model yang telah disiapkan.

```
# menjalankan proses kostum training detektor
!./darknet detector train data/obj.data cfg/yolov4-tiny-custom.cfg yolov4-tiny.conv.29 -dont_show -map
```

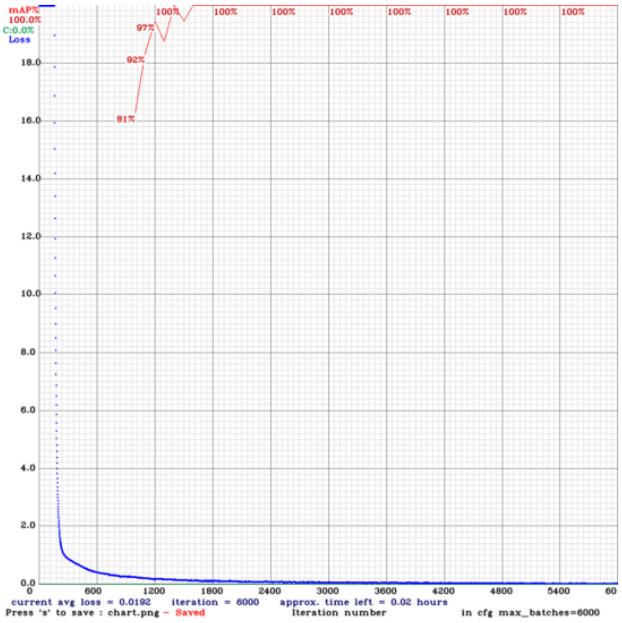
Gambar 14. Perintah untuk melakukan *training*

Hasil dari *training* akan menghasilkan keluaran berupa *trained weight* yang masuk pada folder *backup* seperti pada Gambar 15.



Gambar 15. Hasil *training*

Setelah menghasilkan *trained weight*, Selanjutnya memilih batch terbaik yang memiliki keakuratan *mean Average Precision (mAP)* tertinggi dengan melihat proses *training* data melalui *chart* seperti pada Gambar 16.



Gambar 16. *Chart* proses *training*

4). Rumus Deviasi Standar dan Deviasi Error

Deviasi Standar dan Deviasi error dibutuhkan untuk mengetahui deviasi dari waktu deteksi pengujian pada dataset.

$$\sigma = \sqrt{\frac{\sum(x_i - \mu)^2}{N}}$$

Gambar 17. Rumus Standar Deviasi[1]

$$\mathrm{SE} = \frac{\sigma}{\sqrt{n}}$$

Gambar 18. Rumus Standar Deviasi[1]

C. Pengujian

1). Dataset Pengujian

Sebelum melewati proses pengujian, Dataset pengujian perlu disiapkan terlebih dahulu yang digunakan untuk proses pengujian, banyaknya citra yang digunakan untuk mengetahui sejauh mana keakuratan model dalam mendeteksi suatu objek[8].

2). Implementasi Aplikasi Pengujian

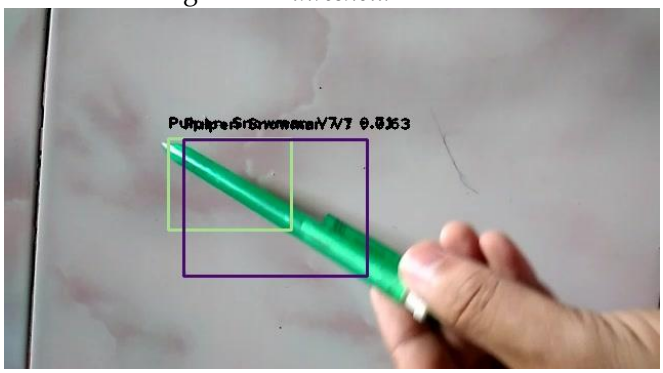
Proses pengujian diimplementasikan pada perangkat lunak untuk mengetahui hasil kinerja metode. Pada Gambar 19 merupakan tampilan dari aplikasi pengujian deteksi objek.



Gambar 19. Aplikasi Pengujian Deteksi Objek

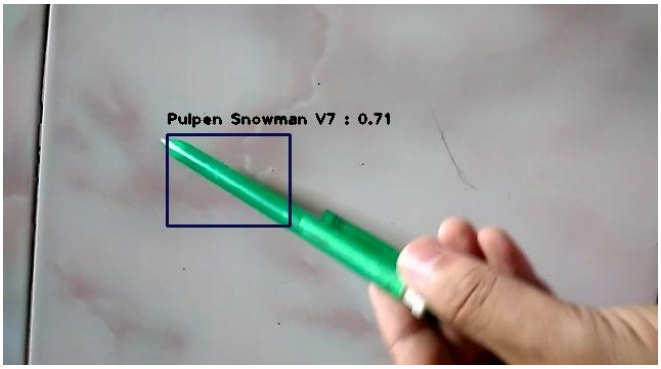
3). Threshold

Beberapa dataset yang dijelaskan sebelumnya digunakan untuk proses pengujian. Hasil dalam proses pengujian yaitu dengan munculnya prediksi kotak pada pembatas, kelas, serta nilai *threshold* pada objek dalam citra. Sehingga dalam proses pengujian perlu diperhatikan suatu nilai *threshold* sebagai batas nilai kepercayaan kotak pembatas yang memuat suatu objek pada kotak pembatas. Jika nilai *threshold* pengujian terlalu rendah atau bahkan tidak terdeteksi, bisa jadi satu objek yang dideteksi memiliki beberapa kotak pembatas. Contoh kasus tersebut bisa dilihat pada Gambar 20 dengan nilai *threshold* 0.7.



Gambar 20. Kotak pembatas yang ganda

Pada Gambar 20, hasil pengujian menunjukan bahwa terdapat 2 objek yang terdeteksi. Namun pada kenyataanya hanya terdapat 1 objek citra. Hal tersebut bisa terjadi dikarenakan nilai keyakinan pada bebrapa kotak pembatas lebih tinggi dari nilai *threshold* yang ditentukan. Sehingga kotak pembatas mendapatkan hasil tersebut. Contoh citra dengan pendeteksian yang tepat dan sesuai dapat dilihat pada Gambar 21 dengan nilai *threshold* 0.5.



Gambar 21. Kotak pembatas yang sesuai

Berdasarkan 2 contoh diatas, perlunya penyesuaian nilai threshold dalam suatu dataset pada citra agar objek bisa dikenali. Untuk mengatasi masalah pada penentuan nilai threshold, penelitian ini menggunakan nilai *threshold* yang rendah dan akan mengambil nilai kepercayaan kotak pembatas yang paling tinggi sebagai hasil dari deteksi.

1. *Output Pengujian*

Setelah mengatur nilai *threshold*, proses pengujian dilakukan dengan menguji dataset pengujian. Nilai *confident* tertinggi yaitu 1.0 atau 100% dan nilai *confident* terendah yaitu 0.0 atau 0%, dapat dilihat bahwa objek pada gambar 22 ini berhasil dikenali dengan tepat dan memperoleh nilai confident tertinggi dengan deteksi kotak pembatas yang sesuai.



Gambar 22. Hasil pengujian citra pada Benang Pink

4). Kinerja Metode YOLO

Kinerja Metode YOLO pada penelitian ini dipengaruhi oleh nilai threshold, Nilai tersebut diberikan kepada seluruh dataset pengujian. Kinerja metode ditentukan oleh Waktu deteksi, Nilai confidence[9]. Pada Tabel 3 merupakan rincian hasil uji coba dataset.

Tabel 3. Hasil Uji Coba Dataset Pengujian Pada Benang Pink

No Pengujian	Threshold	Waktu Deteksi	Confidence
1	0.5	0.1	95.55%
2	0.5	0.2	68.11%
3	0.5	0.2	56.17%
4	0.5	0.1	79.90%
5	0.5	0.1	93.38%
6	0.5	0.1	57.84%
7	0.5	0.1	36.29%
8	0.5	0.09	0%
9	0.5	0.07	41.07%
10	0.5	0.1	68.92%
Nilai Rata-rata		0.186	59.72%
Standar Deviasi		0.006744	-
Standar Error		0.0006744	-

I. KESIMPULAN

Pada penelitian ini, penulis telah berhasil membangun aplikasi pengujian untuk mendeteksi objek menggunakan algoritma YOLO. Penyelesaian proses ini menggunakan model dan *trained weight*. Berdasarkan hasil uji coba 10 dataset pengujian pada objek Benang berwarna Pink telah berhasil melakukan pendeteksian objek dengan waktu rata-rata deteksi 0.186, Standar deviasi waktu 0.006744, Standar error waktu 0.0006744 dan Nilai rata-rata akurasi 59.72%.

REFERENSI

[1] Furey, E. (2006). Standar Deviation Calculator. Tersedia dari: https://www.calculatorsoup.com/calculators/statistics/standard-deviation-calculator.php. [27 Maret 2022].

[2] Jupiyandi, S., Saniputra, F. R., Pratama, Y., Dharmawan, M. R., dan Cholissodin, I. "Pengembangan Deteksi Citra Mobil Untuk Mengetahui Jumlah Tempat Parkir Menggunakan Cuda Dan Modified Yolo," *JTIIK.*, vol. 6(4), hlm. 413-419, 2019. [https://doi.org/10.25126/jtiik.2019641275].

[3] Karimi, G. (2021). Introduction to YOLO Algorithm for Object Detection. Tersedia dari: https://www.section.io/engineering-education/introduction-to-yolo-algorithm-for-object-detection. [5 Mei 2021].

[4] Karlina, O. E., Indarti, D. "Pengenalan Objek Makanan Cepat Saji Pada Video dan Real Time Webcam Menggunakan Metode You Look Only Once (YOLO)," *SINTA.*, vol. 24(3), hlm. 199-208, Desember 2019. [https://doi.org/10.35760/ik.2019.v24i3.2362].

[5] Nazilly, M. L., Rahmat, B., dan Puspaningrum, E. Y. "Implementasi Algoritma Yolo (You Only Look Once) Untuk Deteksi Api," JIFoSI., vol. 1(1), hlm. 81–91, Maret 2020.

[6] Pamungkas, B. P. G., Nugroho, B., dan Anggraeny, F. "Deteksi dan Menghitung Manusia Menggunakan YOLO-CNN," *JIFoSI.*, vol. 02(01), hlm. 67-76, Maret 2021.

[7] Redmon, J., Divvala, S., Girshick, R., dan Farhadi, A. "(YOLO) You Only Look Once," *Cvpr.*, 2016. [https://doi.org/10.1109/CVPR.2016.91].

[8] Santoso, A., dan Ariyanto, G. "Implementasi Deep Learning Berbasis Keras Untuk Pengenalan Wajah," *Emitor.*, Vol. 18(1). hlm. 15-21, 2018. [https://doi.org/10.23917/emitor.v18i01.6235].

[9] Sutjiadi, R., dan Pattiasina, T. J. "Deteksi Objek menggunakan Dashboard Camera untuk Sistem Peringatan Pencegah Kecelakaan pada Mobil," *JTIIK.*, vol. 7(2), hlm. 427-434, 2020. [https://doi.org/10.25126/jtiik.2020712520].

[10] Tanuwijaya, Evan., dan Fatichah, Chastine. "Penandaan Otomatis Tempat Parkir Menggunakan YOLO untuk Mendeteksi Ketersediaan Tempat Parkir Mobil pada Video CCTV," *Brilliant.*, vol. 5(1), hlm. 189-198, 2020. [https://doi.org/10.28926/briliant.v5i1.434].