



UNIVERSITAS INDONESIA

**PENGEMBANGAN SISTEM DETEKSI MULTI OBJEK
UNTUK PENDETEKSIAN IKAN BERBASIS YOLOV7**

SKRIPSI

Muhammad Daffa Ajiputra

1906355781

**FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK KOMPUTER**

DEPOK

MEI 2023



UNIVERSITAS INDONESIA

**PENGEMBANGAN SISTEM DETEKSI MULTI OBJEK
UNTUK PENDETEKSIAN IKAN BERBASIS YOLOV7**

SKRIPSI

**Diajukan sebagai salah satu syarat untuk memperoleh gelar
Sarjana Teknik**

Muhammad Daffa Ajiputra

1906355781

**FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK KOMPUTER**

DEPOK


MEI 2023

HALAMAN PERNYATAAN ORISINALITAS

**Skripsi ini adalah hasil karya saya sendiri,
dan semua sumber baik yang dikutip maupun dirujuk
telah saya nyatakan dengan benar.**

Nama : Muhammad Daffa Ajiputra

NPM : 1906355781

Tanda Tangan :  _____

Tanggal : Mei 2023

LEMBAR PENGESAHAN

KATA PENGANTAR

Puji dan syukur dipanjatkan ke hadirat Allah SWT atas berkat, rahmat, dan nikmat-Nya sehingga buku Skripsi ini dapat diselesaikan tepat waktu dengan judul “Pengembangan Sistem Deteksi Multi Objek untuk Pendeteksian Ikan Berbasis YOLOv7”. Buku skripsi ini dibentuk sebagai salah satu syarat kelulusan Fakultas Teknik Universitas Indonesia, khususnya bagi Program Studi S1 Teknik Komputer dalam Departemen Teknik Elektro.

Buku skripsi tidak akan terselesaikan tanpa adanya bantuan dan dukungan dari pihak lain. Pada kesempatan ini, penulis ingin memberikan apresiasi setinggi-tingginya kepada pihak-pihak terkait yang terdiri dari namun tidak terbatas pada:

1. Orang Tua dan keluarga penulis yang selalu memberikan dukungan dan fasilitas selama menjalani masa perkuliahan.
2. Ibu Dr. Prima Dewi Purnamasari S.T., M.Sc. selaku dosen pembimbing yang selalu mengarahkan dan memberikan bantuan, kritik, serta saran dalam pengerjaan buku skripsi.
3. Bapak Dr. Ruki Harwahyu S.T., M.T., M.Sc. selaku pembimbing akademis yang sudah memberikan arahan dan bantuan selama perkuliahan di program studi Teknik Komputer FTUI
4. Tim peneliti proyek Fishmon baik pihak Aiseeyou, Ahmad Zufar Asshiddiqi, Karenina Kamila, dan Timothy Christian Panggabean yang memberikan bantuan dan wawasan dalam perancangan sistem.
5. Rekan-rekan seperbimbingan, grup “Tekkom Warrior”, asisten laboratorium digital serta rekan-rekan Teknik komputer 2019 yang selalu memberikan bantuan dan motivasi selama menjalani perkuliahan di program studi Teknik Komputer.

Penulis menyadari bahwa dalam menjalankan proses skripsi serta pembuatan buku terdapat beberapa kesalahan yang dilakukan. Oleh karena itu, penulis ingin memohon maaf kepada pihak yang merasakan kekurangan baik dari penulis, pelaksanaan skripsi ataupun dari buku skripsi yang dibuat. Segala kritik dan saran konstruktif dapat disampaikan kepada penulis agar penulis menjadi

pribadi yang lebih baik di waktu yang akan datang. Penulis juga berharap untuk mendapatkan kelancaran dalam menjalani kehidupan pasca-kampus serta menjadi pribadi yang berguna bagi agama, nusa dan bangsa.

Depok, Mei 2023

A handwritten signature in dark ink, appearing to read 'Daffa', with a horizontal line underneath.

Muhammad Daffa Ajiputra

**HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI KARYA
ILMIAH UNTUK KEPENTINGAN AKADEMIS**

Sebagai sivitas akademik Universitas Indonesia, saya yang bertandatangan di bawah ini:

Nama : Muhammad Daffa Ajiputra
NPM : 1906355781
Program Studi : Teknik Komputer
Fakultas : Fakultas Teknik
Jenis Karya : Skripsi

demikian pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Noneksklusif (Non-exclusive Royalty-Free Right)** atas karya ilmiah saya yang berjudul:

**PENGEMBANGAN SISTEM DETEKSI MULTI OBJEK UNTUK
PENDETEKSIAN IKAN BERBASIS YOLOV7**

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangkalan data (database), merawat, dan memublikasikan tugas akhir saya tanpa meminta izin dari saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok
Pada tanggal : Mei 2023
Yang menyatakan



Muhammad Daffa Ajiputra

ABSTRAK

Nama : Muhammad Daffa Ajiputra
Program Studi : Teknik Komputer
Judul : Pengembangan Sistem Deteksi Multi Objek untuk Pendeteksian Ikan Berbasis YOLOv7

Indonesia merupakan salah satu negara pengekspor ikan terbesar di dunia yang membuat sektor perikanan Indonesia memiliki banyak ancaman. *Illegal, unreported, unregulated* (IUU) *fishing* adalah salah satu permasalahan yang memiliki dampak yang cukup signifikan karena membuat kerugian yang cukup besar di sektor perikanan Indonesia. Untuk mencegah permasalahan tersebut, sudah banyak solusi yang diajukan, salah satunya adalah penerapan teknologi seperti kamera pengawas, tetapi solusi tersebut belum memiliki dampak yang signifikan dalam mengurangi dan menghilangkan IUU *fishing*. Oleh karena itu, penelitian ini dilakukan untuk mengembangkan sistem deteksi multi objek untuk pendeteksian ikan berbasis YOLOv7, sebuah model kecerdasan buatan yang dapat melakukan pendeteksian ikan untuk melakukan pengawasan terhadap jumlah ikan yang ditangkap oleh nelayan sehingga IUU *fishing* dapat berkurang secara signifikan. Dari pengujian yang dilakukan, model YOLOv7 menjadi varian model YOLOv7 terbaik yang dapat digunakan untuk melakukan pendeteksian ikan dengan nilai mAP yang dapat mencapai 86.1% dan jumlah FPS yang dapat mencapai 67.6. Nilai tersebut berhasil didapatkan dengan melakukan beberapa penyesuaian pada anotasi data, metode pelatihan model, ukuran citra, dan jumlah iterasi pelatihan. Namun, model YOLOv7 memiliki kecepatan pendeteksian yang sangat lambat ketika dipasang pada Jetson Nano meskipun tidak mengalami penurunan dari segi akurasi pendeteksian.

Kata Kunci: *deep learning, computer vision, object detection, YOLOv7*

ABSTRACT

Name : Muhammad Daffa Ajiputra
Study Program : Computer Engineering
Title : Development of a multi object detection system for fish detection based on YOLOv7

Indonesia is one of the world's largest exporters of fish, which exposes Indonesia's fishing sector to many threats. Illegal, unreported, unregulated (IUU) fishing is one of the problems that resulted in a significant impact in a form of a big loss that is created for the Indonesian fisheries sector. To prevent that problem, there are a lot of solutions that have been proposed, one of which is the application of technology such as surveillance cameras, but it still doesn't have a big impact to reduce and eliminate IUU fishing. Therefore, this research is conducted to develop a multi-object detection system for fish detection based on YOLOv7, an artificial intelligence model that can detect a fish to supervise the number of fish that is caught by the fisherman so IUU fishing can reduce significantly. From the testing, the YOLOv7 model becomes the best YOLOv7 model variant that can be used to detect a fish with the value of mAP that can reach up to 86.1% with an FPS total that can reach up to 67.6. The value can be achieved by doing some modifications in data annotation, the training model method, image size, and iteration on training. However, the YOLOv7 model has a very slow detection speed when it's installed in Jetson Nano even though there isn't any reduction in detection accuracy.

Key words: deep learning, computer vision, object detection, YOLOv7

DAFTAR ISI

HALAMAN PERNYATAAN ORISINALITAS	ii
LEMBAR PENGESAHAN	iii
KATA PENGANTAR.....	iv
HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI KARYA ILMIAH UNTUK KEPENTINGAN AKADEMIS	vi
ABSTRAK	vii
ABSTRACT	viii
DAFTAR ISI.....	ix
DAFTAR GAMBAR.....	xi
DAFTAR TABEL	xiv
BAB 1 PENDAHULUAN	1
1.1. Latar Belakang	1
1.2. Rumusan Masalah	2
1.3. Tujuan Penelitian	3
1.4. Batasan Masalah.....	3
1.5. Metodologi Penelitian	3
1.6. Sistematika Penulisan	5
BAB 2 SISTEM DETEKSI MULTI OBJEK BERBASIS YOLOV7	6
2.1. Machine Learning	6
2.1.1. Deep Learning	7
2.2. Computer Vision	11
2.2.1. Object Detection.....	13
2.3. Convolutional Neural Networks	19
2.4. YOLO.....	22
2.4.1. YOLOv7.....	24
2.5. Fishnet.....	29
2.6. <i>Polygon annotation</i>	30
2.7. Penelitian Terkait	31
BAB 3 PERANCANGAN SISTEM DETEKSI MULTI OBJEK BERBASIS YOLOV7	33
3.1. <i>System requirements</i>	33
3.2. Pengumpulan Data	34

3.3.	<i>Preprocessing Data</i>	34
3.4.	Rancangan Proses Pelatihan Model	36
3.5.	Rancangan Proses Pengujian Model	37
BAB 4 IMPLEMENTASI DAN ANALISIS SISTEM DETEKSI MULTI OBJEK BERBASIS YOLOV7		39
4.1.	Implementasi Sistem	39
4.2.	<i>Dataset</i> Pengujian	40
4.3.	Skenario Pengujian.....	41
4.4.	Pengujian Model	42
4.4.1.	Pengujian Pengaruh Data Berbentuk <i>Polygon annotation</i> Terhadap Akurasi dan Jumlah FPS	42
4.4.2.	Pengujian Perbandingan Metode <i>Object Detection</i> dengan <i>Instance segmentation</i> Terhadap Akurasi	46
4.4.3.	Pengujian Pengaruh Ukuran Gambar Terhadap Akurasi dan Jumlah FPS	49
4.4.4.	Pengujian Pengaruh Penambahan Epochs dalam Memicu <i>Early Stopping</i> Terhadap Akurasi dan Jumlah FPS.....	55
4.4.5.	Pengujian Pendeteksian Model dengan Jetson Nano.....	58
4.5.	Dampak Terhadap Lingkungan, Masyarakat dan/atau Bidang Lain.....	64
BAB 5 KESIMPULAN		66
5.1.	Kesimpulan	66
5.2.	Saran.....	67
DAFTAR PUSTAKA		68

DAFTAR GAMBAR

Gambar 2.1 Perbandingan alur machine learning tradisional dan <i>deep learning</i> [10]	7
Gambar 2.2 Contoh <i>backpropagation</i> pada 4-layer <i>neural networks</i> [10]	11
Gambar 2.3 Contoh <i>Region of Interest</i> pada citra sebuah anjing[10]	14
Gambar 2.4 Hasil prediksi sebelum dan sesudah diterapkan <i>non-maximum suppression</i> [10]	16
Gambar 2.5 Contoh <i>Precision-Recall Curve</i> dengan <i>IoU Threshold</i> = 0.5	17
Gambar 2.6 Proses ekstraksi fitur pada <i>convolutional layers</i> [10]	20
Gambar 2.7 Contoh <i>max pooling feature map</i> berukuran 4 X 4 dengan <i>window</i> berukuran 2 x 2[10]	20
Gambar 2.8 Contoh <i>global average pooling feature map</i> berukuran 4 X 4[10]...	21
Gambar 2.9 Arsitektur <i>fully connected layers</i> [10]	22
Gambar 2.10 Ilustrasi pendeteksian menggunakan algoritma YOLO[12]	23
Gambar 2.11 Arsitektur awal model YOLO[12]	24
Gambar 2.12 Arsitektur Model YOLOv7-Tiny	26
Gambar 2.13 <i>Layer Modul Convolutional</i> YOLOv7-Tiny	26
Gambar 2.14 Arsitektur ELAN Model YOLOv7-Tiny	26
Gambar 2.15 <i>Layer Modul SPPCSPC</i> YOLOv7-Tiny	26
Gambar 2.16 Arsitektur Model YOLOv7	27
Gambar 2.17 <i>Layer Modul Convolutional</i> YOLOv7	27
Gambar 2.18 Arsitektur ELAN pada <i>Backbone</i> (Atas) dan <i>Head</i> (Bawah) Model YOLOv7	28

Gambar 2.19 <i>Layer</i> Modul SPPCSPC YOLOv7	28
Gambar 2.20 <i>Layer</i> _Modul MP pada YOLOv7	28
Gambar 2.21 Arsitektur Model YOLOv7-X.....	29
Gambar 2.22 Arsitektur ELAN Model YOLOv7-X	29
Gambar 2.23 Contoh data <i>Polygon annotation</i> pada kerusakan sebuah mobil[16]	30
Gambar 3.1 Rancangan sistem pendeteksian ikan	33
Gambar 3.2 Tahapan <i>pre-processing</i> data	34
Gambar 3.3 Proses <i>labelling</i> dengan Label Studio [17]	35
Gambar 3.4 Skema proses pelatihan model YOLOv7	37
Gambar 4.1 Kode untuk Mengonversi <i>Dataset</i> dari <i>Polygon annotation</i> menjadi <i>Bounding Box</i>	43
Gambar 4.2 Hasil Pengujian Model YOLOv7 pada Data <i>Bounding Box</i>	45
Gambar 4.3 Hasil Pengujian Model YOLOv7 pada Data <i>Polygon annotation</i>	45
Gambar 4.4 Hasil Pengujian Model YOLOv7-X dengan Metode <i>Instance segmentation</i>	48
Gambar 4.5 Hasil Pengujian Model YOLOv7-X dengan Metode <i>Object Detection</i>	48
Gambar 4.6 Perbandingan Tren Peningkatan Nilai mAP pada YOLOv7-Tiny dengan Ukuran Citra 640 Piksel (Merah Muda) dan 800 Piksel (Jingga)	51
Gambar 4.7 Perbandingan Tren Peningkatan Nilai mAP pada YOLOv7 dengan Ukuran Citra 640 Piksel (Kuning) dan 800 Piksel (Biru Muda).....	51
Gambar 4.8 Perbandingan Tren Peningkatan Nilai mAP pada YOLOv7-X dengan Ukuran Citra 640 Piksel (Hijau Muda) dan 800 Piksel (Ungu)	52

Gambar 4.9 Hasil Pengujian Model YOLOv7-Tiny dengan Ukuran Citra 640 Piksel (Kiri) dan 800 Piksel (Kanan)	52
Gambar 4.10 Hasil Pengujian Model YOLOv7 dengan Ukuran Citra 640 Piksel (Kiri) dan 800 Piksel (Kanan)	53
Gambar 4.11 Hasil Pengujian Model YOLOv7 dengan Ukuran Citra 640 Piksel (Kiri) dan 800 Piksel (Kanan)	53
Gambar 4.12 Contoh Citra yang Digunakan dalam Pengujian	56
Gambar 4.13 Hasil Pengujian pada Model YOLOv7 dengan <i>Epochs</i> 300 (Kiri), <i>Epochs</i> 1000 dengan <i>Patience</i> 50 (Tengah), <i>Epochs</i> 1000 dengan <i>Patience</i> 100 (Kanan).....	56
Gambar 4.14 Perintah untuk Mencari <i>Swap Memory</i> [19]	59
Gambar 4.15 Perintah untuk Mencari Ruang Pada Memori Penyimpanan[19]....	59
Gambar 4.16 Perintah untuk Membuat <i>Swap File</i> [19]	60
Gambar 4.17 Perintah untuk Mengaktifkan <i>Swap File</i> menjadi <i>Swap Memory</i> ...	60
Gambar 4.18 Contoh Salah Satu <i>Frame</i> Hasil <i>Inference</i> pada Video Pengujian..	62
Gambar 4.19 Contoh Citra Hasil Ekstraksi <i>Frame</i> Video Pengujian	63
Gambar 4.20 Hasil Pengujian Akurasi Pendeteksian pada Jetson Nano (Kiri) dan PC Pelatihan (Kanan).....	63

DAFTAR TABEL

Tabel 2.1 Persebaran bagian <i>dataset</i> Fishnet	30
Tabel 2.2 Hasil <i>precision</i> , <i>recall</i> dan <i>F1-Score</i> model YOLOv3-Tiny, YOLOv4-Tiny dan YOLOv4-Tiny dengan <i>Spatial Pyramid Pooling</i> pada <i>dataset</i> Fish-Gres dalam pendeteksian kepala dan ekor ikan [14].	32
Tabel 3.1 Spesifikasi Komputer	33
Tabel 4.1 Spesifikasi Jetson Nano untuk Pengujian Model	39
Tabel 4.2 Persebaran citra pada <i>dataset</i>	40
Tabel 4.3 Jumlah data tiap kelas pada <i>dataset</i>	40
Tabel 4.4 Hasil Pengujian Model dengan <i>Dataset</i> Berbentuk <i>Polygon annotation</i> dan <i>Bounding Box</i>	44
Tabel 4.5 Hasil Pengujian Model dengan Metode <i>Object Detection</i> dan <i>Instance segmentation</i>	47
Tabel 4.6 Hasil Pengujian Model dengan Ukuran Citra 640 Piksel dan 800 Piksel	50
Tabel 4.7 Hasil Pengujian Model dengan Peningkatan <i>Epochs</i>	55
Tabel 4.8 Perbandingan <i>F1-Score</i> Model YOLOv7 dalam Mendeteksi Kelas yang Termasuk Ikan	57
Tabel 4.9 Hasil Pengujian Model Pada Jetson Nano dan Perangkat Pelatihan	61
Tabel 4.10 Hasil Pengujian Akurasi Model Pada Jetson Nano dan Perangkat Pelatihan	62

BAB 1

PENDAHULUAN

1.1. Latar Belakang

Indonesia merupakan salah satu negara pengekspor ikan terbesar di dunia. Tercatat pada tahun 2021, Indonesia berada di peringkat ke tujuh negara pengekspor ikan terbesar di dunia untuk ikan yang sudah difillet maupun ikan yang dibekukan dengan valuasi sebesar 558 juta US Dollar [1]. Salah satu komoditi ekspor ikan Indonesia yang memiliki nilai valuasi yang tinggi adalah ikan tuna. Nilai valuasi ekspor ikan tuna dari Indonesia pada tahun 2021 mencapai 325.4 juta US Dollar dengan volume ekspor mencapai 48.3 Mega Ton untuk ikan tuna yang difillet dan sudah dibekukan [2]. Nilai tersebut meningkat dari tahun 2020 sebesar 34.93% untuk valuasi ikan dan 43.31% untuk volume ekspor ikan tuna dan diproyeksikan akan meningkat Kembali pada tahun 2022 dengan meningkatnya pula nilai valuasi ekspor ikan tuna secara global.

Di sisi lain, dengan banyaknya komoditi ekspor laut Indonesia, Indonesia pun tidak terlepas dari kasus *illegal, unreported, unregulated (IUU) fishing*. Tercatat pada tahun 2021, Kementerian kelautan dan perikanan menangkap 167 Kapal Indonesia maupun Kapal Asing seperti kapal dari Vietnam dan Malaysia yang melakukan IUU *fishing* [3]. Diperkirakan IUU *fishing* yang terjadi selama 2021 membuat Indonesia mengalami kerugian sebesar 74 Juta US Dollar [4]. Kementerian Kelautan dan Perikanan sendiri sudah menerapkan beberapa solusi untuk mencegah terjadinya IUU *fishing* di wilayah Indonesia, sebagai contoh pada Maret 2021, terdapat 8 kapal Malaysia yang ditenggelamkan akibat *illegal fishing* [5]. Selain itu, negara-negara di ASEAN juga saling bekerja sama untuk mendorong keterbukaan informasi di sektor perikanan melalui sistem pemantauan kapal serta sistem pelaporan dan pengawasan elektronik.

Penerapan teknologi sendiri sudah menjadi salah satu metode yang digunakan untuk mencegah terjadinya tindak kejahatan di berbagai bidang, salah satu contohnya adalah *computer vision*. *Computer vision* merupakan bidang pada *artificial intelligence (AI)* yang memungkinkan komputer untuk mengobservasi

dan memahami informasi penting yang ada pada suatu citra, video dan masukan berbentuk visual lainnya [6]. *Computer vision* telah dikembangkan sejak tahun 1959 yang diawali dengan mengamati Bagaimana respon otak seekor kucing saat melihat deretan gambar dan disimpulkan bahwa respon pertamanya adalah suatu garis. Hingga saat ini, sudah terbentuk berbagai metode penerapan *computer vision* di berbagai aspek kehidupan, salah satunya adalah *object detection*.

Object detection merupakan metode *computer vision* yang berfungsi untuk melakukan pendeteksian sebuah objek pada suatu citra atau video [7]. Algoritma *object detection* pertama kali ditemukan pada tahun 2001 oleh pasangan Viola-Jones yang disebut sebagai *VJ-Detectors* yang saat itu masih terbuat secara manual. *Object detection* mengalami transisi pada 2012 menggunakan Teknik otomasi ketika *Alexnet* ditemukan. Sejak saat itu, algoritma *object detection* berbasis *deep learning* banyak ditemukan salah satunya adalah YOLO (*You Only Look Once*). YOLO pertama kali ditemukan oleh Joseph Redmon pada tahun 2016. YOLO bekerja dengan menerapkan satu *neural network* pada citra yang akan membagi citra tersebut menjadi beberapa wilayah. Algoritma YOLO terus mengalami perkembangan dan saat ini, sudah ada algoritma YOLO versi ketujuh (YOLOv7) yang merupakan pengembangan dari YOLO versi keenam.

Dari latar belakang tersebut, penulis berinisiatif untuk membentuk suatu penelitian tentang sistem pendeteksian ikan dengan sistem deteksi secara multi objek menggunakan algoritma YOLOv7. Sistem pendeteksian ini akan memanfaatkan suatu kamera yang terpasang pada kapal dan terhubung dengan suatu Mini-PC yang didalamnya sudah terpasang model AI yang digunakan untuk melakukan pendeteksian ikan yang nantinya data pendeteksian tersebut akan dikirimkan menuju server *back-end* menggunakan *cloud* dan akan ditampilkan di suatu *dashboard* sederhana.

1.2. Rumusan Masalah

Rumusan masalah dari penelitian ini adalah sebagai berikut:

1. Bagaimana cara membuat sebuah sistem deteksi multi objek berbasis YOLOv7 sebagai sistem pendeteksian ikan?
2. Varian model YOLOv7 manakah yang memiliki hasil terbaik dan tingkat kesalahan yang kecil dalam melakukan pendeteksian ikan?

1.3. Tujuan Penelitian

Tujuan dari penelitian ini adalah sebagai berikut:

1. Merancang sistem pendeteksian multi objek dengan algoritma YOLOv7 yang dapat membedakan beberapa jenis ikan hasil tangkapan nelayan.
2. Membandingkan dan mengevaluasi model YOLOv7 sehingga diperoleh varian YOLOv7 terbaik yang akan digunakan dalam pembentukan sistem pendeteksian ikan secara multi objek.

1.4. Batasan Masalah

Batasan masalah dalam penelitian ini adalah sebagai berikut:

1. Sistem pendeteksian harus bisa mendeteksi enam jenis objek (*human, tuna, skipjack tuna, tongkol, squid, unknown*) berdasarkan himpunan data yang akan dikumpulkan dari video kamera cctv yang terpasang pada kapal.
2. Sistem pendeteksian akan mendeteksi melalui kamera cctv yang akan memberikan keluaran berupa sebuah kotak pembatas dengan label untuk objek yang terdeteksi.
3. Proses pelatihan dan pengujian model YOLOv7 dilakukan terhadap suatu citra dari video yang ditangkap melalui kamera cctv pada kapal dengan enam jenis objek (*human, tuna, skipjack tuna, tongkol, squid, unknown*) yang mungkin terdapat pada citra.

1.5. Metodologi Penelitian

Metode yang digunakan selama penelitian ini adalah sebagai berikut:

1. Studi literatur

Tahapan studi literatur mencakup pencarian dan pengumpulan referensi yang berkaitan dengan semua penerapan teknologi yang diterapkan pada penelitian ini.

2. Konsultasi dengan dosen pembimbing dan tim peneliti

Konsultasi dilakukan secara rutin setiap pekan dengan dosen pembimbing untuk mendapatkan umpan balik kemajuan penelitian, menerima saran dan membahas hambatan yang dialami mengenai topik-topik yang berkaitan dengan penelitian ini. Konsultasi juga dilakukan dengan tim peneliti untuk mendapatkan wawasan tambahan terkait model yang akan digunakan.

3. Pengumpulan dan pelabelan data

Pengumpulan data dilakukan dengan mengobservasi tiap *frame* video hasil tangkapan kamera pada kapal yang terdiri dari jenis ikan yang akan diklasifikasikan dan akan dilabeli sesuai dengan jenis ikan tersebut.

4. Desain Sistem

Desain sistem pendeteksian ikan dilakukan dengan membuat rancangan sistem yang akan diimplementasikan sesuai batasan masalah yang telah ditentukan.

5. Rancangan proses pelatihan model

Rancangan proses pelatihan model YOLOv7 dibentuk sesuai dengan studi literatur dan referensi-referensi lain sehingga dapat menghasilkan model yang maksimal

6. Rancangan proses pengujian dan perbandingan model

Rancangan proses pengujian dan perbandingan model YOLOv7, YOLOv7-Tiny dan YOLOv7-X dibentuk untuk membandingkan hasil pelatihan model dengan seimbang tanpa mengunggulkan varian mana pun.

7. Analisis

Analisis diberikan terhadap rancangan sistem pendeteksian ikan yang dibentuk dan ketiga varian model YOLOv7 yang telah melewati proses pelatihan model

8. Kesimpulan

Pemberian kesimpulan terhadap keseluruhan penelitian yang telah dilakukan berdasarkan hasil yang diperoleh dan analisis yang didapatkan

dari rancangan sistem pendeteksian ikan dan perbandingan varian model YOLOv7 agar dapat diterapkan pada sistem tersebut.

1.6. Sistematika Penulisan

Sistematika Penulisan pada penelitian ini dibagi ke dalam 5 bab, yaitu:

BAB I PENDAHULUAN

Bab ini menjelaskan latar belakang, tujuan, batasan masalah, dan metodologi yang digunakan sebagai acuan pelaksanaan penelitian

BAB II DASAR TEORI

Bab ini menjelaskan teori yang digunakan dalam pembuatan rancangan sistem pendeteksian ikan dan perbandingan model YOLOv7 yang digunakan

BAB III PERANCANGAN SISTEM

Bab ini menjelaskan rincian dari rancangan sistem pendeteksian ikan yang telah dibentuk, data dan pemrosesan data yang diterapkan dalam pelatihan dan pengujian data, serta rancangan proses pelatihan dan pengujian varian model YOLOv7

BAB IV IMPLEMENTASI DAN ANALISIS

Bab ini menjelaskan terkait implementasi dari sistem yang telah dirancang serta menampilkan hasil serta analisis dari pengujian sistem pada beberapa skenario yang diuji.

BAB V KESIMPULAN

Bab ini menjelaskan kesimpulan dari hasil penelitian yang telah dilakukan beserta beberapa saran yang dapat diterapkan untuk pengembangan sistem yang telah terbentuk.

BAB 2

SISTEM DETEKSI MULTI OBJEK BERBASIS YOLOV7

2.1. Machine Learning

Machine learning merupakan salah satu sub-bidang dari *Artificial Intelligence* (AI) yang bertujuan untuk membuat komputer bisa mereplikasi kemampuan yang dimiliki seorang manusia dengan menginstruksikannya untuk mempelajari sebuah data dalam jumlah yang banyak [8]. Menurut Mitchell (1997), kata *Learning* untuk komputer didefinisikan sebagai “Sebuah program belajar dari *experience* yang terdiri dari beberapa kelas *tasks* dan *performance measure*, jika *tasks* yang diukur dengan *performance measure* akan meningkat sesuai dengan *experience*” [9].

Machine learning memungkinkan kita untuk menyelesaikan *tasks* yang terlalu sulit untuk diselesaikan hanya dengan suatu program yang dibentuk oleh seorang manusia dan bersifat tetap. *Tasks* didefinisikan sebagai bagaimana suatu sistem *machine learning* harus mengolah sebuah *example* yang didefinisikan sebagai kumpulan dari *features* yang telah diukur secara kuantitatif dari beberapa objek atau kejadian yang akan diproses oleh sistem *machine learning*. Sebagai contoh, *example* yang digunakan merupakan suatu citra, maka *features* dari sebuah citra adalah piksel-piksel yang ada pada citra tersebut. Salah satu *tasks* yang paling umum diselesaikan menggunakan *machine learning* adalah klasifikasi. *Tasks* jenis ini meminta suatu komputer untuk coba mengklasifikasikan suatu masukan ke beberapa kategori yang ada.

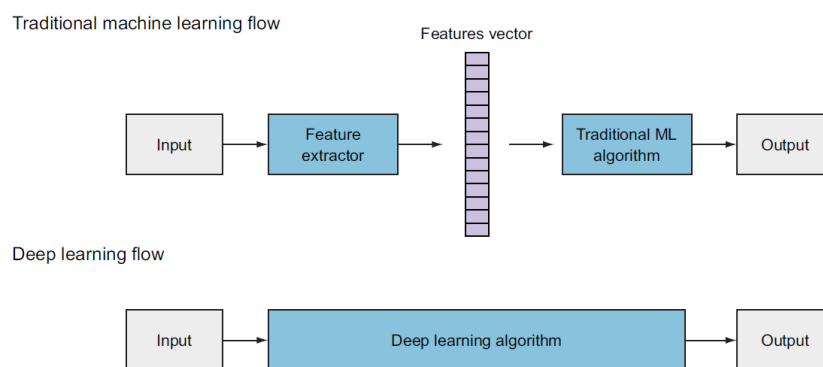
Untuk mengevaluasi kemampuan suatu algoritma *machine learning*, kita perlu membentuk *performance measure* yang sesuai dengan *tasks* yang terbentuk pada sistem. Sebagai contoh, pada *tasks* klasifikasi salah satu parameter pengukurannya adalah *accuracy*, yaitu proporsi dari *examples* yang keluarannya tepat dari model yang terbentuk. Selain itu, terdapat juga *error rate*, yaitu proporsi dari *examples* yang keluarannya salah dari model yang terbentuk. Lalu, untuk mengukur seberapa baik model yang terbentuk dalam penggunaannya di dunia

nyata, biasanya digunakan sebuah *test set* berupa data yang berbeda dari data yang digunakan untuk melatih sistem *machine learning*.

Algoritma *machine learning* sendiri dapat dibedakan sesuai dengan *experience* apa yang diberikan pada algoritma tersebut di mana sebuah *experience* akan berbentuk sebuah *dataset*, yaitu kumpulan dari banyak *examples*. Secara umum, *experience* yang dapat diterima oleh sebuah *machine learning* terbagi menjadi *unsupervised* dan *supervised*. Algoritma *unsupervised learning* adalah algoritma yang mendapatkan *experience* dari *dataset* yang memiliki banyak *features* dan mempelajari property dari *dataset* tersebut, sedangkan algoritma *supervised learning* adalah algoritma yang mendapatkan *experience* dari *dataset* yang memiliki sebuah *features*, tetapi juga terikat dengan sebuah label.

2.1.1. Deep Learning

Deep learning merupakan cabang dari *machine learning* yang menggunakan banyak layer dari *neural networks* [8]. *Deep learning* merupakan pengembangan dari alur *machine learning* tradisional yang masih menggunakan *feature extractor* untuk bisa mengambil *features* yang ada pada suatu masukan. Algoritma *deep learning* sendiri sudah memiliki *feature extractor* pada layer *neural networks* nya sehingga proses ekstraksi *features* dari masukan pada algoritma *deep learning* dilakukan secara otomatis.



Gambar 2.1 Perbandingan alur machine learning tradisional dan *deep learning* [10]

Deep learning terdiri dari beberapa layer *neural networks* di mana suatu *neural networks* terdiri dari beberapa *perceptron*. *Perceptron* merupakan *neural network* yang paling sederhana. *Perceptron* memiliki cara kerja yang mirip dengan neuron yang ada pada otak manusia di mana neuron manusia akan menerima sinyal elektrik dari dendrit, mengatur sinyal elektrik tersebut menjadi beberapa bagian dan mengirimkannya menuju sinapsis untuk dikirimkan ke neuron lain ketika sinyal masukan sudah melebihi batasannya. Dari cara kerja tersebut, *perceptron* bekerja dengan dua fungsi, yaitu mengkalkulasi *weighted sum* masukan yang merepresentasikan kekuatan sinyal dan menerapkan *step function* untuk menentukan apakah sinyal tersebut sudah melewati batas yang ditentukan atau belum. Oleh karena itu, suatu *perceptron* memiliki empat bagian utama, yaitu:

- A. *Input vector*, yaitu vektor dari *feature* yang berhasil diekstrak dari masukan. Umumnya, disimbolkan dengan X sebagai sebuah vektor dari masukan (x_1, x_2, \dots, x_n)
- B. *Weights vector*, yaitu representasi seberapa pentingnya *example* tersebut dibandingkan dengan *example* lain yang ada pada *dataset*. Direpresentasikan dengan sebuah vektor dari *weights* (w_1, w_2, \dots, w_n)
- C. *Neuron functions*, yaitu kalkulasi dari sinyal masukan berupa penjumlahan dari *weighted sum* dan *step activation function*.
- D. *Output*, yaitu hasil *neuron functions* yang diatur oleh *activation function*.

Sebuah *perceptron* merupakan sebuah fungsi linear yang sangat efektif digunakan untuk suatu *dataset* linear di mana *dataset* tersebut dapat dibagi oleh sebuah garis lurus. Akan tetapi, jika *dataset* yang digunakan memiliki bentuk yang lebih kompleks, contohnya adalah sebuah citra atau video, sebuah *perceptron* kurang efektif untuk menyelesaikan permasalahan tersebut. Oleh karena itu, untuk menyelesaikan permasalahan tersebut, dibutuhkan lebih dari satu *perceptron* untuk bisa menyesuaikan fungsi yang ada pada *dataset* atau bisa disebut sebagai *multilayer perceptron* [10].

Deep learning menggunakan *multilayer perceptron* dalam membentuk arsitekturnya. *Multilayer perceptron* bekerja dengan menumpuk neuron menjadi

beberapa layer yang biasa disebut sebagai *hidden layer*. Layer yang ada pada *hidden layer* saling terhubung satu sama lain dengan sebuah *weights connections*. Komponen utama yang ada pada suatu *multilayer perceptron* adalah:

- A. *Input layer*, yaitu layer yang terdiri dari *features* yang dimiliki oleh masukan data
- B. *Hidden layers*, yaitu layer yang terdiri dari neuron-neuron yang akan mengekstrak vektor dari *feature* yang akan diproses pada *input layer*.
- C. *Weight connections (edges)*, yaitu nilai yang terdapat pada tiap hubungan antar *nodes* pada *neural networks* untuk menggambarkan pentingnya *nodes* tersebut pada hasil prediksi.
- D. *Output layer*, yaitu hasil dari pemrosesan pada *hidden layers* yang dapat berbentuk suatu nilai riil atau kumpulan dari sebuah probabilitas.

Deep learning membutuhkan tiga tahap yang akan dilakukan secara terus menerus untuk menghasilkan model yang maksimal. Tahap pertama, melakukan kalkulasi dari *weights sum* tiap *nodes* dan *activation* untuk menghasilkan suatu prediksi atau disebut sebagai proses *feedforward* [11]. *Weights sum* adalah penjumlahan dari vektor *weights* yang ada pada suatu *perceptrons* untuk tiap masukan. Persamaan yang digunakan untuk menentukan *weights sum* adalah:

$$\mathbf{s}_i = \mathbf{w}_i^T \mathbf{x}_i + \mathbf{b}_i \quad (2.1)$$

Setelah mendapatkan *weights sum* tiap *nodes*, *weights sum* tersebut akan dikalikan dengan non-linear *activation function* yang direpresentasikan dengan h sehingga didapatkan fungsi untuk mendapatkan hasil prediksi tiap masukan adalah:

$$\mathbf{y}'_i = \mathbf{h}(\mathbf{s}_i) \quad (2.2)$$

Tahap kedua, mengukur tingkat kesalahan dari hasil prediksi *neural networks* dibandingkan dengan keluaran yang seharusnya. Caranya adalah dengan menerapkan suatu *error function* atau bisa juga disebut sebagai *loss functions*. Jika hasil prediksi memiliki nilai *loss* yang tinggi, model yang terbentuk kurang baik. Salah satu *loss function* yang umum digunakan adalah *cross-entropy*. *Cross-entropy* adalah *loss function* yang umum digunakan dalam permasalahan klasifikasi.

Cross-entropy akan menghitung perbedaan antara dua distribusi probabilitas, yaitu *predicted distribution* untuk distribusi hasil prediksi dan *true distribution* untuk distribusi hasil sebetulnya [10]. Persamaan untuk *cross-entropy* adalah:

$$E(W, b) = - \sum_{i=1}^m y'_i \log(p_i) \quad (2.3)$$

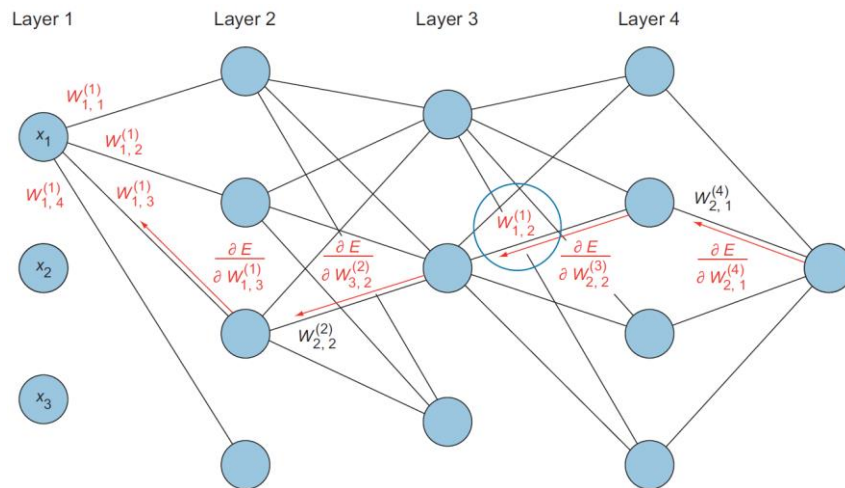
di mana y' adalah probabilitas sesungguhnya, p adalah probabilitas hasil prediksi dan m adalah jumlah kelas yang dilatih. Untuk mengetahui kesalahan *cross-entropy* untuk semua data (n) pada *dataset*, persamaan yang digunakan akan menjadi:

$$E(W, b) = - \sum_{i=1}^n \sum_{j=1}^m y'_{ij} \log(p_{ij}) \quad (2.4)$$

Tahap ketiga, menerapkan *optimization functions* seperti *gradient descent* atau jenis lainnya untuk memodifikasi *neural networks* secara berulang hingga di titik paling optimalnya yang dilakukan dengan mengkomputasi turunan *loss function* E_n untuk example n terhadap *weights* w menggunakan *chain rule* dari *output layer* menuju *input layer* yang bisa disebut sebagai *backpropagation*. Secara umum, *backpropagation* melakukan turunan dari *error* yang didapatkan terhadap suatu *weights*. Setelah mengalami *backpropagation*, *weights* yang dimiliki suatu node akan diperbaharui mengikuti persamaan

$$w_{next-step} = w_{current} + \Delta w \quad (2.5)$$

Ketika *backpropagation* diterapkan pada suatu *multilayer perceptron*, terdapat banyak *weights* yang perlu dikomputasi. Oleh karena itu, digunakan operasi *chain rule* untuk mencari nilai turunan dari error terhadap *weights* sesuai dengan node yang akan dilewati ketika kembali menuju *input layer*.



Gambar 2.2 Contoh *backpropagation* pada 4-layer *neural networks*[10]

Sebagai contoh, pada Gambar 2.2, untuk mencari tahu *weights* dari masukan setelah diterapkan *backpropagation*, turunan *error* terhadap *weights* dari *output layer* menuju masukan yang dituju adalah

$$\frac{dE}{dw_{1,3}^{(1)}} = \frac{dE}{dw_{2,1}^{(4)}} \times \frac{dw_{2,1}^{(4)}}{dw_{2,2}^{(3)}} \times \frac{dw_{2,2}^{(3)}}{dw_{3,2}^{(2)}} \times \frac{dw_{3,2}^{(2)}}{dw_{1,3}^{(1)}} \quad (2.7)$$

2.2. Computer Vision

Computer vision adalah salah satu sub-bidang dari *artificial intelligence* yang berfokus pada bagian visual untuk bisa menciptakan suatu teknologi yang bisa memahami suatu benda di dunia nyata melalui video atau citra. Teknologi *computer vision* dibentuk berdasarkan *visual perception*, yaitu bentuk aksi untuk melakukan observasi sebuah pola atau objek menggunakan suatu masukan visual. *Computer vision* dilatih tidak hanya untuk melihat lingkungan di sekitar kita, tetapi juga membentuk sistem yang bisa memahami lingkungan di sekitarnya melalui masukan visual [10].

Computer vision menjadi salah satu topik yang memiliki lingkup penelitian yang aktif hingga saat ini dikarenakan menjadi salah satu topik yang cukup menantang untuk sebuah komputer. Hal tersebut dikarenakan *computer vision*

merupakan *inverse problem*, di mana kita mencari beberapa informasi yang tidak cukup untuk diketahui dan mengembalikan informasi tersebut untuk menemukan solusi sepenuhnya dari permasalahan tersebut [11]. Dibutuhkan sebuah model yang didasari oleh konsep fisika dan probabilistic atau sebuah *machine learning* dengan *dataset* yang besar untuk bisa menghilangkan keambiguan dari solusi yang mungkin digunakan, tetapi memodelkan suatu data visualisasi dunia nyata jauh lebih kompleks dari yang dibayangkan.

Secara umum, *computer vision* memiliki prinsip yang sama dengan sistem penglihatan pada manusia. Sistem penglihatan seorang manusia terdiri dari mata yang berfungsi untuk menangkap visual yang ada di lingkungan sekitar makhluk tersebut serta otak yang berfungsi untuk memahami objek dari visual yang dilihat oleh mata. Dari konsep tersebut, *computer vision* memiliki dua komponen untuk diterapkan, yaitu *sensing devices* dan *interpreting device*.

Sensing devices adalah perangkat yang berfungsi seperti mata, yaitu untuk menangkap visual yang ada di sekitar perangkat tersebut. *Sensing devices* yang digunakan untuk menerapkan *computer vision* perlu disesuaikan dengan pengaplikasian yang diinginkan. Beberapa perangkat yang umum digunakan sebagai *sensing devices* seperti kamera, radar, CT scan, dan lain-lain.

Interpreting device adalah perangkat yang berfungsi seperti otak, yaitu untuk memahami dan menafsirkan objek yang berhasil ditangkap oleh *sensing devices*. Pada *computer vision*, algoritma yang digunakan merupakan *interpreting device* pada sistem *computer vision*. Karena prinsip kerja yang mirip layaknya otak seorang manusia, ilmuwan menerapkan konsep yang sama untuk membuat algoritma yang bertindak sebagai *interpreting device*. Dari konsep tersebut, terciptalah teknologi yang biasa dikenal sebagai *artificial neural networks* (ANNs). Ketika sebuah jaringan neuron memiliki jutaan hingga miliaran neuron didalamnya, jaringan tersebut akan menghasilkan suatu algoritma yang mampu melakukan pembelajaran yang biasa disebut *deep learning* [10].

2.2.1. Object Detection

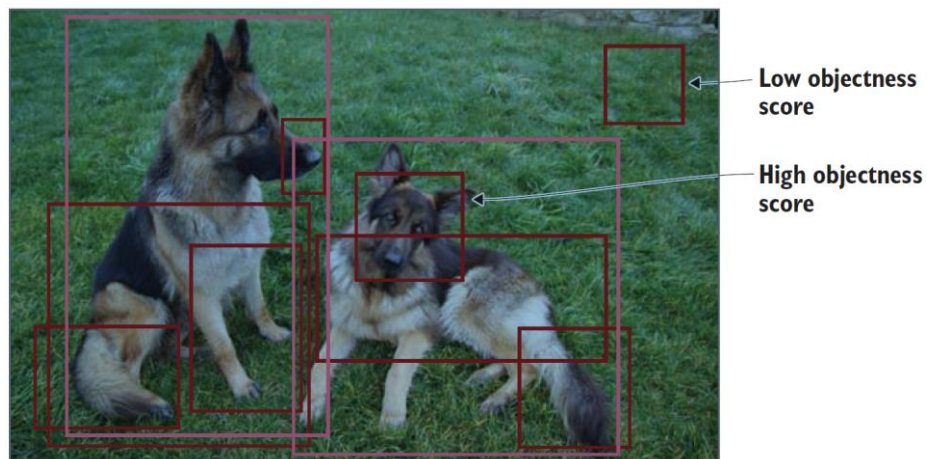
Object detection merupakan salah satu contoh pengaplikasian teknologi *computer vision* di kehidupan nyata. *Object detection* merupakan pengembangan dari *image classification* di mana *object detection* berfungsi untuk membuat *interpreting device* yang diterapkan pada sistem *computer vision* bisa mengetahui lokasi dan jumlah objek yang terdapat di suatu citra. Secara umum, *object detection* akan mengidentifikasi lokasi dan kelas objek dengan memecah citra tersebut menjadi bagian yang lebih kecil dan memberikan kelas untuk setiap bagian yang dipecah sehingga objek yang terdapat pada citra tersebut dapat terlabeli [10].

Framework object detection memiliki empat komponen utama, diantaranya:

A. Region Proposals

Region proposals adalah suatu bagian yang diberikan oleh model *deep learning* berbentuk suatu *region of interest* (RoIs) yang akan diproses lebih lanjut oleh sistem. *Region of interest* adalah bagian pada suatu citra yang sistem perkiraan memiliki sebuah objek yang dikalkulasi menggunakan *objectness score*.

Pada tahap ini, sistem akan membentuk ribuan *bounding boxes* untuk dianalisis dan diklasifikasikan oleh *neural networks*. *Objectness score* dari tiap *bounding boxes* adalah hasil analisis dari *neural networks* yang akan mengklasifikasikan *bounding boxes* tersebut sebagai *foreground* (Objek) atau *background* (Bukan Objek). Jika *bounding boxes* tersebut melewati *threshold* yang ditetapkan pada *neural networks*, *bounding boxes* tersebut akan diklasifikasikan sebagai *foreground* dan akan diteruskan ke tahap berikutnya oleh *neural networks* [10].



Gambar 2.3 Contoh *Region of Interest* pada citra sebuah anjing[10]

Terdapat beberapa parameter yang perlu diperhatikan, seperti *threshold* yang terkonfigurasi. Apabila terlalu kecil, akan terlalu banyak *bounding boxes* yang terbentuk oleh *neural networks* yang memungkinkan semua objek terdeteksi, tetapi akan membuat proses pendeteksian lebih lambat karena membutuhkan proses komputasional yang kompleks. Oleh karena itu, penetapan parameter perlu disesuaikan dengan permasalahan yang ingin diselesaikan untuk mengurangi jumlah RoIs yang terbentuk.

B. *Feature extraction and network predictions*

Komponen ini adalah lanjutan dari *region proposal* di mana *bounding boxes* yang termasuk sebagai *foreground region* akan diekstrak *feature*-nya dan akan digunakan untuk menentukan kelas dari objek yang dikenali pada citra tersebut. *Feature* pada *foreground region* akan diekstrak oleh sebuah *pretrained model image classification* agar hasil yang didapatkan akan tergeneralisasi secara merata. Setelah mengekstraksi *feature*, *foreground region* akan dianalisis oleh *neural networks* dan dibuat dua prediksi untuk tiap *foreground region*, yaitu

- *Bounding-box prediction*, yaitu prediksi untuk menentukan lokasi dari *foreground region* pada citra tersebut. Prediksi akan berbentuk sebuah tuple (x, y, w, h) di mana x dan y adalah koordinat titik tengah dari *foreground region*, sedangkan w dan h adalah panjang dan lebar dari region tersebut.

- *Class prediction*, yaitu prediksi yang umumnya terbentuk dengan *activation function softmax* yang akan memprediksi probabilitas tiap kelas yang dilatih untuk objek pada citra tersebut.

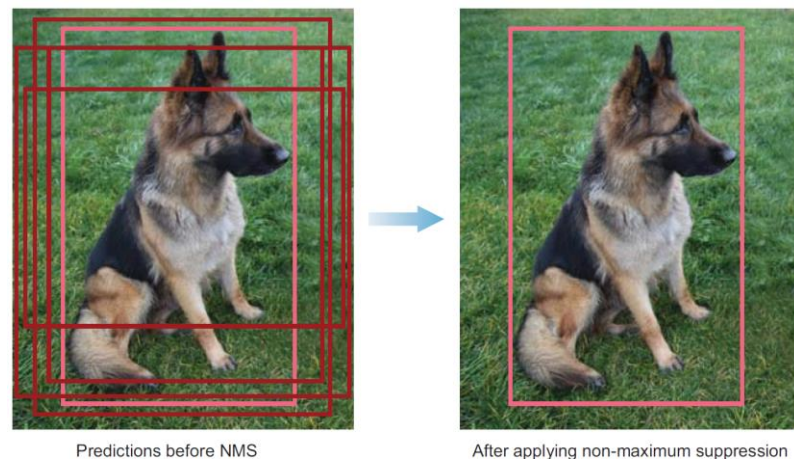
Sebuah objek yang dikenali pada suatu citra dapat memiliki beberapa *foreground region* yang secara tepat melakukan klasifikasi, tetapi tidak semua *region* akan digunakan untuk melakukan deteksi. Di beberapa permasalahan, hanya dibutuhkan satu *foreground region* saja dengan tingkat prediksi tertinggi untuk melakukan suatu pendeteksian karena *foreground region* yang banyak akan membuat objek tersebut terhitung dengan jumlah lebih dari satu. Oleh karena itu, *foreground region* yang bukan merupakan *region* dengan tingkat prediksi tertinggi perlu dihapuskan untuk mencegah hal tersebut.

C. *Non-maximum suppression (NMS)*

Non-maximum suppression (NMS) adalah komponen yang digunakan untuk memastikan bahwa hanya ada satu *bounding boxes* untuk tiap objek. Cara kerja dari komponen adalah NMS akan mencari semua *bounding boxes* yang mengarah ke sebuah objek dan mencari *bounding boxes* dengan probabilitas prediksi tertinggi dan akan menghilangkan semua *bounding boxes* yang memiliki probabilitas prediksi lebih rendah.

Cara kerja dari *non-maximum suppression* dalam mencari probabilitas prediksi tertinggi adalah:

- Menghilangkan semua *bounding boxes* yang probabilitas prediksinya tidak melewati *confidence threshold* yang sudah disesuaikan dengan kebutuhan.
- Mencari semua *bounding boxes* yang tersisa dan memilih *bounding boxes* dengan probabilitas tertinggi
- Mengkalkulasi area yang saling meniban antara *bounding boxes* hasil prediksi (*predicted*) dengan *bounding boxes* acuan (*ground truth*) dan mencari komparasi nilai area tersebut. Nilai komparasi itu disebut sebagai *intersection over union (IoU)*.
- Menghilangkan *bounding boxes* yang memiliki nilai IoU lebih rendah dibandingkan *threshold* yang diterapkan.



Gambar 2.4 Hasil prediksi sebelum dan sesudah diterapkan *non-maximum suppression*[10]

D. Evaluation Metrics

Komponen ini adalah komponen yang akan mengevaluasi hasil prediksi dari tiga komponen sebelumnya. Beberapa satuan yang umum digunakan dalam mengevaluasi hasil prediksi sebuah model *object detection* adalah:

- *Frames Per Second (FPS)*

Frames Per Second adalah satuan evaluasi yang mengukur kecepatan sebuah model *object detection* melakukan pendeteksian. Jumlah FPS dapat dikalkulasi menggunakan nilai *inference time* dengan persamaan

$$FPS = \frac{1000}{inference\ time\ (ms)} \quad (2.8)$$

- *Intersection over Union (IoU)*

Intersection over Union adalah satuan evaluasi yang akan menilai *overlap* yang terjadi antara *bounding boxes* hasil prediksi (*predicted*) dengan *bounding boxes* acuan (*ground truth*). IoU akan menentukan apakah hasil pendeteksian valid (*True Positive*) atau tidak (*False Positive*) di mana range nilainya antara 0-1 dan semakin besar nilainya, semakin baik hasil pendeteksian. Nilai IoU suatu *bounding boxes* dapat ditentukan menggunakan persamaan

$$IoU = \frac{B_{ground\ truth} \cap B_{predicted}}{B_{ground\ truth} \cup B_{predicted}} \quad (2.8)$$

Jika suatu *bounding boxes* memiliki nilai yang lebih rendah dibandingkan dengan *threshold* yang diterapkan, hasil pendeteksian dianggap tidak valid (*False Positive*), sedangkan jika melebihi *threshold*, hasil pendeteksian dianggap valid (*True Positive*).

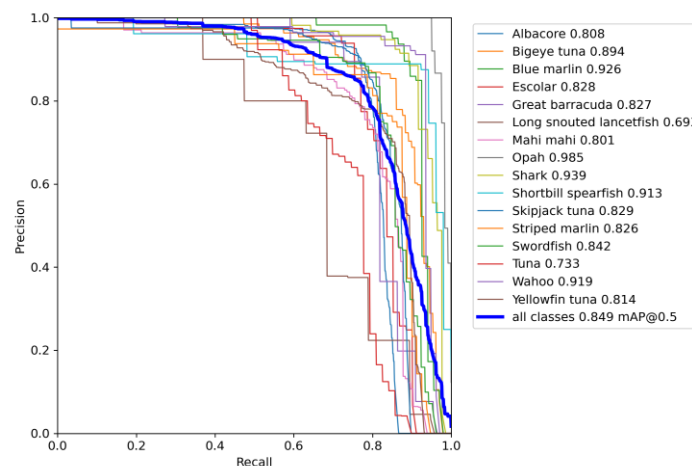
- *Precision-recall Curve (PR-Curve)*

Precision-recall Curve adalah kurva perbandingan antara *precision* dan *recall* dari suatu model *object detection* yang akan diproses untuk setiap *confidence threshold*. *Precision* adalah satuan evaluasi yang mengukur kemampuan model untuk mengidentifikasi sebuah objek yang relevan pada sebuah citra. Nilai *precision* sebuah model *object detection* dapat ditentukan menggunakan persamaan

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \quad (2.9)$$

Lalu, *recall* adalah satuan evaluasi yang mengukur kemampuan model untuk mencari semua objek yang relevan pada sebuah citra. Nilai *recall* dapat ditentukan menggunakan persamaan

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \quad (2.10)$$



Gambar 2.5 Contoh *Precision-Recall Curve* dengan *IoU Threshold* = 0.5

- *Mean Average Precision* (mAP)

Dari kurva *precision-recall*, kita dapat menentukan *average precision* (AP), yaitu nilai rata-rata presisi untuk setiap kelas pada pelatihan model sesuai dengan *threshold* yang ditentukan menggunakan nilai dari *interpolated precision* untuk setiap level *recall*. Nilai *interpolated precision* dapat dicari menggunakan persamaan

$$p_{interpolated} = \max_{r' \geq r} [p(r')] \quad (2.11)$$

Dari nilai tersebut, kita dapat menemukan nilai *average precision* dengan mencari rata-rata dari *interpolated precision* terhadap *recall* untuk setiap kelas yang dapat dicari menggunakan persamaan

$$AP = \frac{1}{n} \sum_{i=1}^{n-1} (r_{i+1} - r_i) p_{interpolated}(r_{i+1}) \quad (2.12)$$

Nilai *average precision* hanya merepresentasikan rata-rata presisi sebuah kelas pada model. Umumnya, sebuah model deteksi objek memiliki beberapa kelas objek sehingga perlu dicari rata-rata *average precision* seluruh kelas untuk mengetahui akurasi model secara keseluruhan yang umum disebut sebagai *mean Average Precision* (mAP). Nilai *mean Average Precision* (mAP) ditentukan dengan persamaan

$$mAP = \frac{1}{k} \sum_{l=1}^k AP_l \quad (2.13)$$

Pada persamaan tersebut, k merepresentasikan jumlah kelas yang dilatih pada model dan AP_l adalah *average precision* dari masing-masing kelas. Nilai mAP umumnya ditentukan sesuai dengan *IoU threshold* yang digunakan, seperti mAP_{0.5} dan mAP_{0.5:0.95}. Nilai mAP_{0.5} adalah nilai mAP pada *IoU threshold* = 0.5, sedangkan mAP_{0.5:0.95} adalah nilai mAP dalam rentang 10 *IoU threshold* antara 0.5 sampai 0.95[12]. Ketika diuji

pada *IoU threshold* yang lebih tinggi, nilai AP cenderung mengalami penurunan karena objek yang berhasil terdeteksi tidak banyak sehingga nilai mAP0.5:0.95 akan cenderung lebih kecil dibandingkan dengan mAP0.5.

- *F1 Score*

F1 Score adalah satuan evaluasi untuk mengetahui nilai *harmonic mean* dari *precision* dan *recall*. *F1 score* memiliki rentang nilai antara 0-1. Semakin besar nilai *F1 score*, model tersebut memiliki *precision* dan *recall* yang sempurna dan juga memiliki *accuracy* pendeteksian yang baik. *F1 score* dapat ditentukan dengan persamaan

$$F1 - Score = 2 * \frac{precision * recall}{precision + recall} \quad (2.14)$$

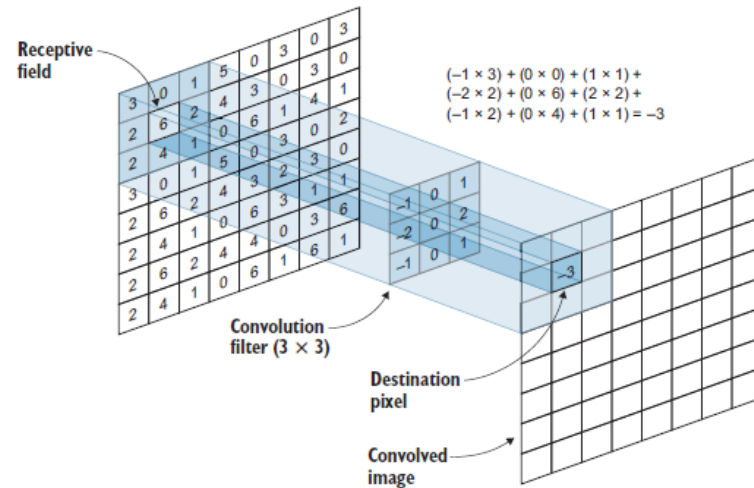
2.3. Convolutional Neural Networks

Convolutional neural networks merupakan sebuah *neural networks* yang menerapkan operasi *convolution* dalam membentuk sebuah arsitektur *neural networks*. *Convolution* merupakan sebuah operasi linear matematis dua buah fungsi untuk mencari fungsi baru. *Convolutional neural networks* umum digunakan untuk sebuah data yang bisa diterapkan sebuah *grid cell* didalamnya, seperti data *time series* yang didefinisikan dengan data dalam *grid* 1D atau data citra yang dapat didefinisikan sebagai data 2D dari sebuah piksel [9]. Secara umum, arsitektur *convolutional neural networks* terbagi menjadi 3 *layers* utama, yaitu:

A. Convolutional layers

Convolutional layers adalah *layers* yang berfungsi untuk mengekstraksi fitur-fitur pada sebuah citra untuk melakukan identifikasi objek yang ada pada citra tersebut. *Convolutional layers* menggunakan sebuah *window* yang disebut sebagai *convolutional filters* untuk mengekstrak fitur pada sebuah citra di mana filter ini akan memeriksa semua piksel yang ada semua citra. *Filter* tersebut memiliki *weights* tersendiri untuk setiap fitur pada citra dan akan melakukan operasi *dot product* antara *filter* dengan satu fitur tertentu yang akan dijumlahkan jika semua operasi *dot product* sudah dilakukan. *Layers* ini akan menghasilkan sebuah citra

baru yang disebut sebagai *convolved image* atau *feature map* yang fiturnya berasal dari operasi *dot product* antara *filter* dengan fitur citra masukan.



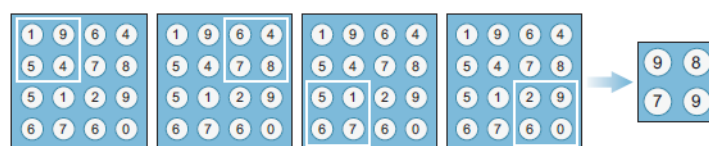
Gambar 2.6 Proses ekstraksi fitur pada *convolutional layers*[10]

B. Pooling layers

Pooling layers adalah *layers* yang berfungsi untuk mengurangi jumlah parameter agar proses komputasi yang dibutuhkan tidak terlalu kompleks. *Pooling layers* akan mengubah ukuran dari *feature map* menggunakan operasi statistika seperti *max* atau *average* untuk mengurangi jumlah parameter yang akan diteruskan menuju *layer* berikutnya di mana hanya parameter-parameter yang penting saja yang akan diteruskan menuju *layer* berikutnya. *Pooling layers* akan memodifikasi nilai *feature map* saja tanpa mengurangi jumlah *feature map* yang dihasilkan dari *convolutional layers*. *Pooling layers* terbagi menjadi dua jenis utama, yaitu:

- *Max Pooling*

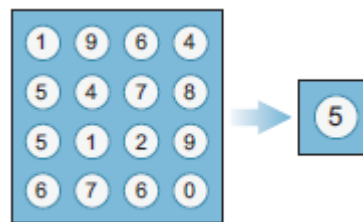
Max pooling adalah jenis *pooling layers* yang akan memeriksa *feature map* hasil *convolutional layers* dan mengambil nilai fitur yang tertinggi pada *window* sehingga terbentuk *feature map* baru yang hanya akan berisi nilai fitur hasil *pooling*.



Gambar 2.7 Contoh *max pooling feature map* berukuran 4 X 4 dengan *window* berukuran 2 x 2[10]

- *Average Pooling*

Average Pooling adalah jenis *pooling layers* yang akan memeriksa *feature map* hasil *convolutional layers* dan mengambil rata-rata nilai fitur pada *window*. Terdapat beberapa jenis *average pooling*, contohnya adalah *global average pooling* yang akan merata-rata semua nilai fitur pada *feature map* sehingga hanya ada satu nilai yang diteruskan ke *layer* berikutnya.

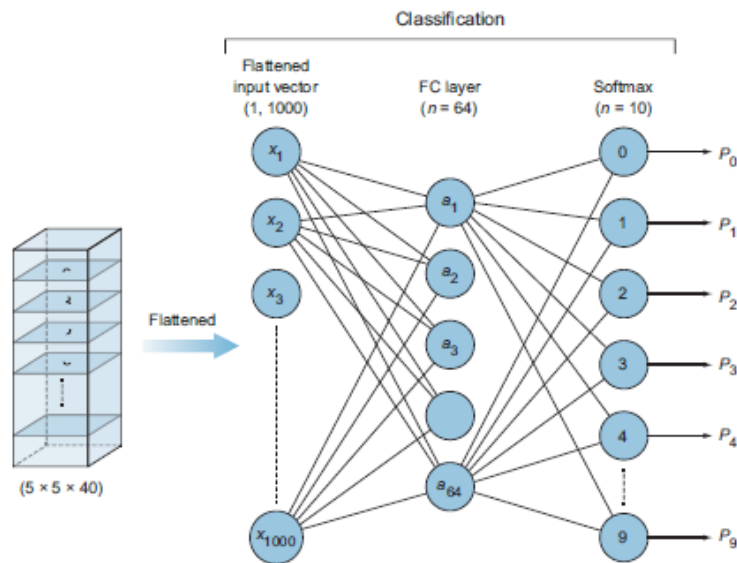


Gambar 2.8 Contoh *global average pooling feature map* berukuran 4 X 4[10]

C. *Fully connected layers*

Fully connected layers adalah *layer* yang berfungsi untuk melakukan klasifikasi. *Layer* ini akan menerima hasil *feature extraction* yang dilakukan pada *convolutional* dan *pooling layers* dan akan memprosesnya sehingga hasil ekstraksi dapat digunakan untuk melakukan klasifikasi. Arsitektur *fully connected layers* terdiri dari tiga komponen utama, yaitu:

- *Input flattened vector*, hasil ekstraksi fitur akan dimasukkan ke *flatten layer* agar nilainya dapat diubah menjadi sebuah vektor berukuran (1, n) dimana n adalah nilai sesuai dengan dimensi hasil ekstraksi.
- *Hidden layer*, vektor dari hasil ekstraksi akan dihubungkan satu sama lain yang akan menghasilkan kelas klasifikasi membentuk neuron-neuron.
- *Output layer*, neuron yang terbentuk akan dihubungkan dengan *activation function* yang sesuai dan menghasilkan node-node kelas objek.



Gambar 2.9 Arsitektur *fully connected layers*[10]

2.4. YOLO

You Only Look Once (YOLO) adalah salah satu keluarga model *deep learning* yang umum digunakan untuk melakukan *object detection*. YOLO bukan model *object detection* dengan tingkat akurasi tertinggi dibandingkan model lain seperti R-CNN, tetapi YOLO merupakan salah satu model dengan waktu pendeteksian tercepat sehingga banyak diterapkan pada sistem *real-time* seperti kamera pengawas [8].

YOLO menggunakan pendekatan yang berbeda dalam melakukan pendeteksian di mana YOLO akan membagi citra menjadi beberapa bagian menggunakan sebuah *grid cell* berukuran $S \times S$. Tiap *grid cell* tersebut akan melakukan pendeteksian terhadap objek dengan membentuk beberapa *bounding boxes* dan menentukan nilai *confidence* dari *bounding boxes* tersebut. Nilai *confidence* akan menggambarkan kepastian model untuk mendeteksi objek dan keakuratan model dalam memprediksi objek pada *box* tersebut. Nilai *confidence* umumnya didefinisikan sebagai

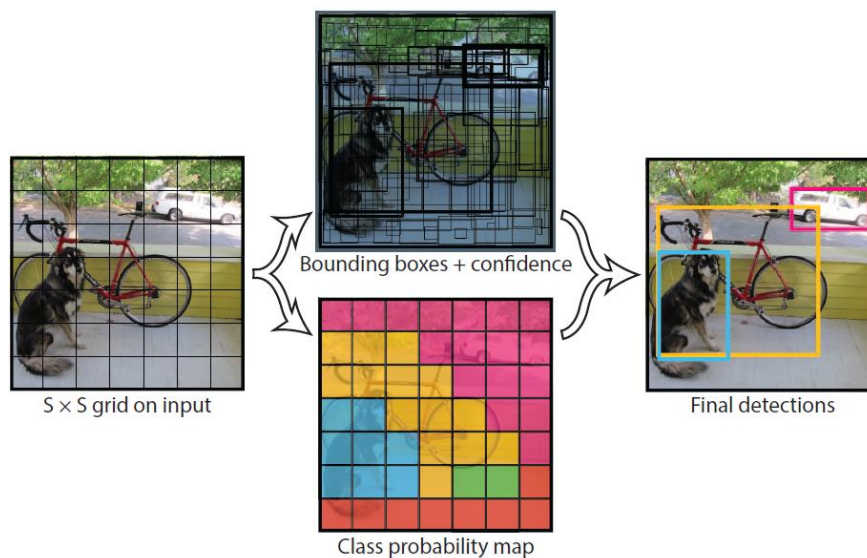
$$Conf = Pr(Object) * IoU_{pred}^{truth} \quad (2.15)$$

Bounding boxes yang terbentuk memiliki lima unsur utama, yaitu x , y , w , h , dan *confidence* di mana x dan y merepresentasikan titik tengah *bounding boxes* terhadap *grid cell*, w dan h merepresentasikan tinggi dan lebar hasil prediksi terhadap keseluruhan citra serta *confidence* merepresentasikan IoU *bounding boxes* hasil prediksi dengan *bounding boxes* acuan.

Grid cell yang memiliki objek didalamnya akan melakukan prediksi untuk menentukan kelas objek tersebut dengan hanya satu kemungkinan kelas yang akan coba diprediksi untuk tiap *grid cell* berapapun jumlah *bounding boxes* yang terbentuk. Prediksi tersebut dapat direpresentasikan dengan $\Pr(\text{Class}_i|\text{Object})$ yang akan dikalikan nilai *confidence* tiap *box* pada saat pengujian sehingga akan dihasilkan nilai *confidence* untuk tiap kelas objek dengan persamaan

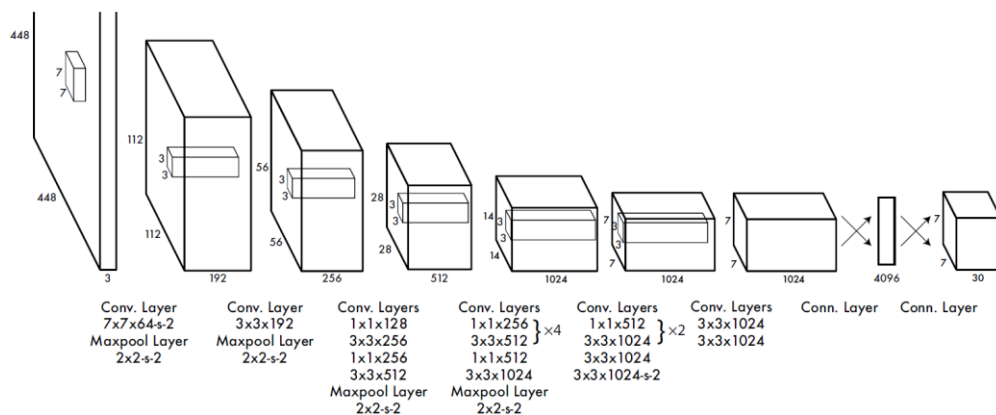
$$\text{Conf}(\text{Class}_i) = \Pr(\text{Class}_i) * \text{IoU}_{\text{pred}}^{\text{truth}} \quad (2.16)$$

Nilai prediksi tersebut akan merepresentasikan probabilitas dari sebuah kelas objek tampil pada *box* dan Bagaimana kualitas dari *box* hasil prediksi sesuai dengan objek yang ada.



Gambar 2.10 Ilustrasi pendeteksian menggunakan algoritma YOLO[12]

YOLO menggunakan *convolutional neural networks* sebagai arsitektur utama yang akan mengekstraksi fitur citra dan menampilkan hasil prediksi. *Convolutional neural networks* yang digunakan mengambil referensi dari arsitektur model *GoogLeNet* untuk melakukan *image classification* dengan 24 *convolution layers* dan dua *fully connected layers*. YOLO juga mengganti salah satu modul yang digunakan *GoogLeNet* berupa modul *Inception* menjadi 1 X 1 *reduction layers* ditambah 3 X 3 *convolutional layers*. Arsitektur tersebut akan menghasilkan sebuah kumpulan prediksi berukuran 7 X 7 X 30 dalam bentuk sebuah *tensor*.



Gambar 2.11 Arsitektur awal model YOLO[12]

Walaupun YOLO merupakan salah satu algoritma dengan kecepatan pendeteksian yang tercepat, versi awal YOLO memiliki beberapa kelemahan. Pertama, satu *grid cell* hanya bisa melakukan pendeteksian untuk satu kelas objek saja sehingga jika terdapat lebih dari satu kelas objek, YOLO tidak bekerja dengan optimal. Kedua, jika *grid cell* mendeteksi kumpulan dari objek-objek kecil seperti sebuah burung, YOLO juga tidak bekerja dengan optimal dikarenakan *bounding boxes* yang terbentuk membutuhkan *spatial constraints* yang kuat sehingga terdapat limitasi dari objek yang dapat dideteksi pada *grid cell* yang sama [8].

2.4.1. YOLOv7

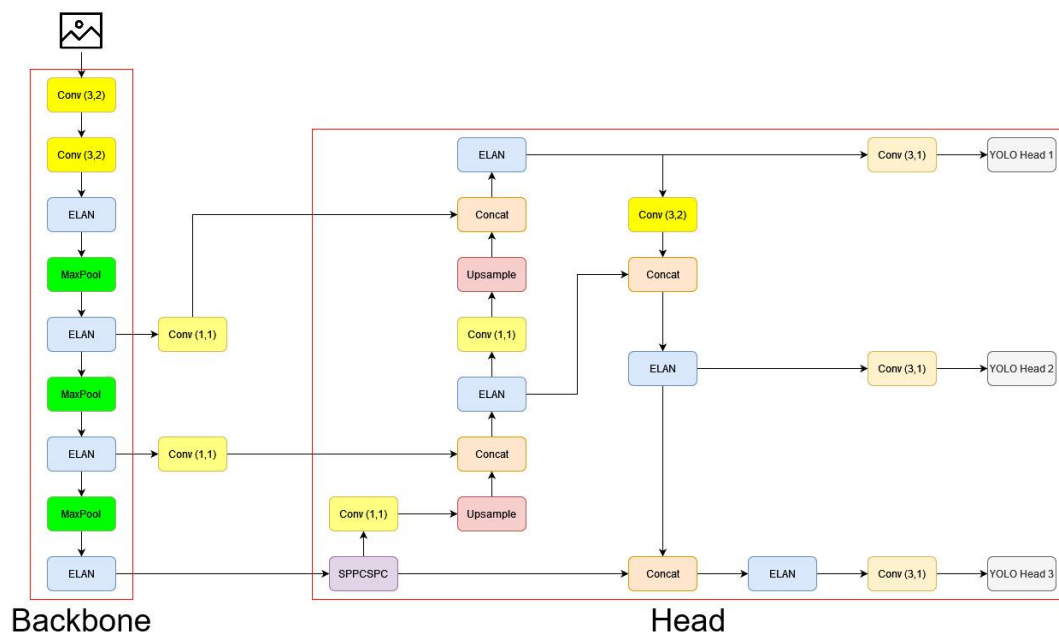
YOLOv7 merupakan salah satu pengembangan dari keluarga model YOLO yang dipublikasikan pada tahun 2022. Pengembangan YOLOv7 berfokus pada bagaimana mengoptimalkan proses training yang dilakukan sebuah model dengan

memperkuat *training cost* untuk meningkatkan akurasi tanpa meningkatkan *inference cost* menggunakan *optimized modules* dan *optimized methods*. Pengujian yang dilakukan menggunakan YOLOv7 menunjukkan bahwa model YOLOv7 memiliki performa yang lebih baik dibandingkan dengan beberapa model lain seperti YOLOv5, YOLOR, YOLOX, PPYOLOE, dan beberapa model lain. Sebagai contoh, YOLOv7-Tiny-SiLU memiliki nilai *average precision* yang lebih baik dibandingkan dengan YOLOv5-N dengan peningkatan sebesar 10.7% yang diikuti dengan hasil *inference* yang lebih cepat 127 FPS[13].

Secara umum, arsitektur YOLOv7 terbagi menjadi dua bagian, yaitu bagian *backbone* dan *head*. Bagian *backbone* adalah bagian pada arsitektur yang akan melakukan ekstraksi fitur untuk mendapatkan objek yang akan dideteksi, sedangkan *head* adalah bagian arsitektur yang akan melakukan pendeteksian pada model deteksi objek yang dihubungkan oleh bagian *neck*[14]. YOLOv7 terbagi menjadi dua tipe utama, yaitu YOLOv7-P5 dan YOLOv7-P6. Perbedaan utama dari kedua tipe itu adalah perangkat yang umum digunakan dalam mengeksekusi model dimana YOLOv7-P5 lebih digunakan pada perangkat GPU *consumer* atau *edge*, sedangkan YOLOv7-P6 lebih digunakan pada GPU untuk *Cloud Server*. Oleh karena itu, penelitian ini akan lebih berfokus pada tipe YOLOv7-P5. YOLOv7-P5 memiliki beberapa varian, diantaranya:

A. YOLOv7-Tiny

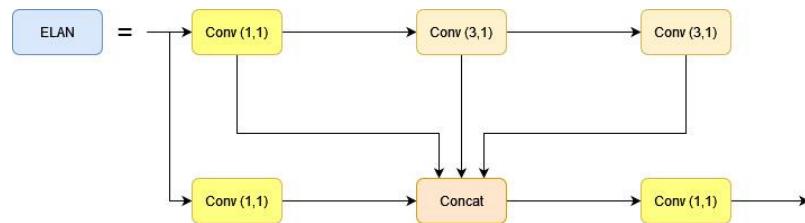
YOLOv7-Tiny merupakan varian model YOLOv7 dengan arsitektur paling sederhana. Karena memiliki arsitektur yang sederhana, YOLOv7-Tiny umum digunakan sebagai model kecerdasan buatan pada perangkat-perangkat *edge devices* seperti mikrokontroler atau mini-PC. Arsitektur YOLOv7-Tiny terdiri dari bagian *backbone* dan *head* dimana *backbone* YOLOv7-Tiny terdiri dari 28 *layer*, sedangkan *head* YOLOv7-Tiny terdiri dari 49 *layer*. Berikut merupakan visualisasi arsitektur YOLOv7-Tiny.



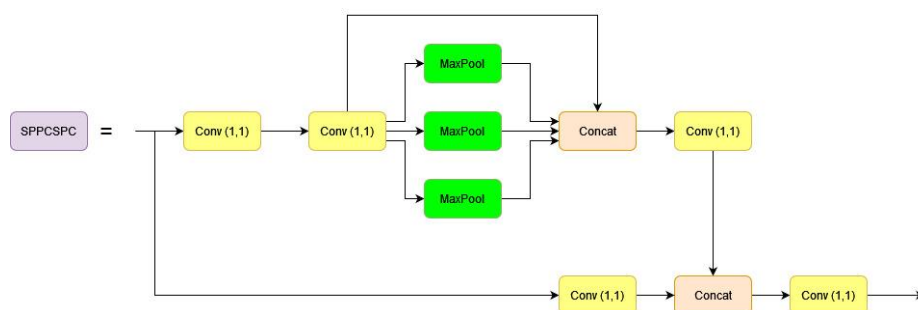
Gambar 2.12 Arsitektur Model YOLOv7-Tiny



Gambar 2.13 Layer Modul Convolutional YOLOv7-Tiny



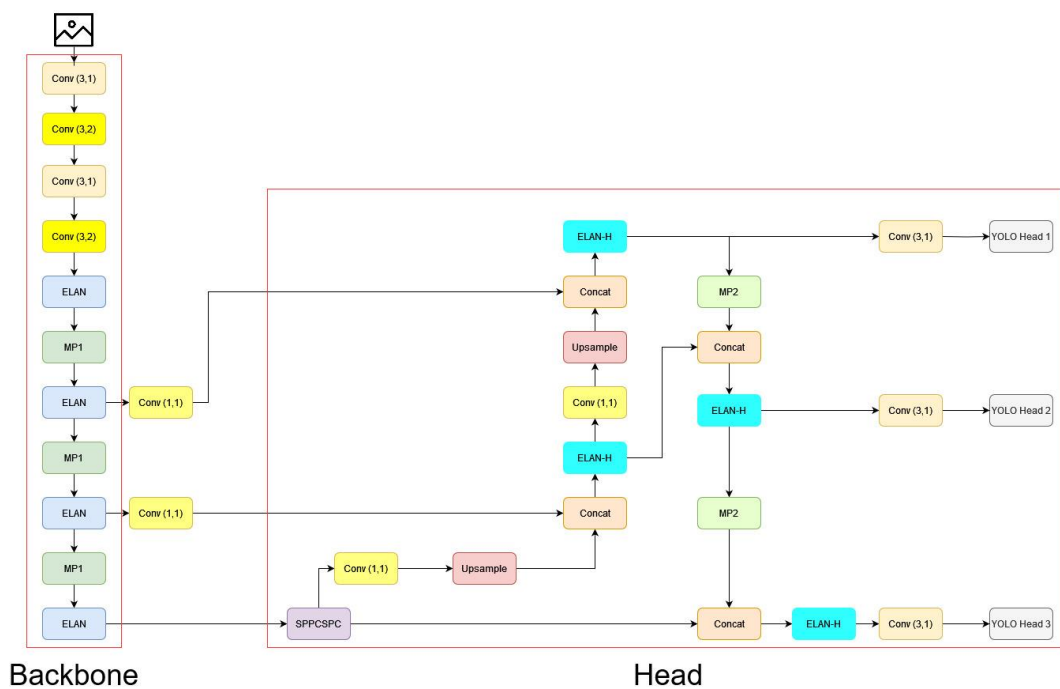
Gambar 2.14 Arsitektur ELAN Model YOLOv7-Tiny



Gambar 2.15 Layer Modul SPPCSPC YOLOv7-Tiny

B. YOLOv7

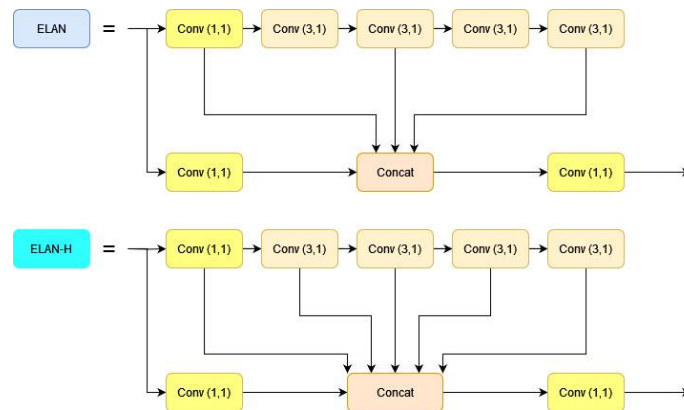
YOLOv7 merupakan standar dari seluruh varian model YOLOv7. YOLOv7 memiliki arsitektur yang lebih kompleks dibandingkan dengan YOLOv7-Tiny, tetapi bukan varian dengan arsitektur yang paling kompleks. Perbedaan antara arsitektur YOLOv7-Tiny dengan YOLOv7 adalah tidak adanya modul MP pada YOLOv7-Tiny, yaitu modul untuk menggabungkan hasil *pooling* dengan modul *convolutional*. Selain itu, YOLOv7 memiliki dua arsitektur ELAN pada bagian *backbone* dan *head*. Bagian *backbone* YOLOv7 terdiri dari 50 layer, sedangkan bagian *head* YOLOv7 terdiri dari 55 layer. Berikut merupakan visualisasi arsitektur YOLOv7.



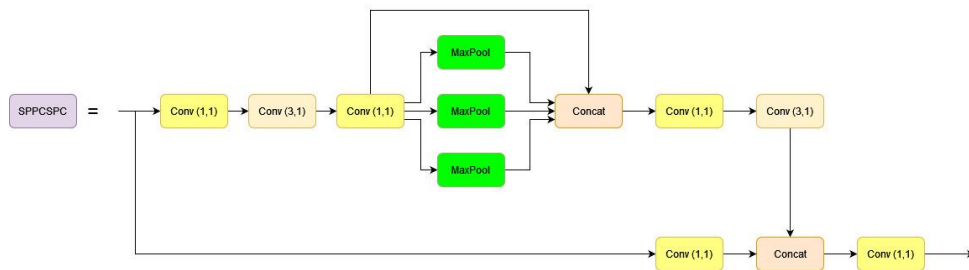
Gambar 2.16 Arsitektur Model YOLOv7



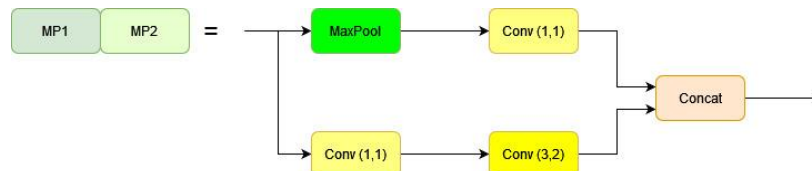
Gambar 2.17 Layer Modul Convolutional YOLOv7



Gambar 2.18 Arsitektur ELAN pada *Backbone* (Atas) dan *Head* (Bawah) Model YOLOv7



Gambar 2.19 *Layer* Modul SPPCSPC YOLOv7

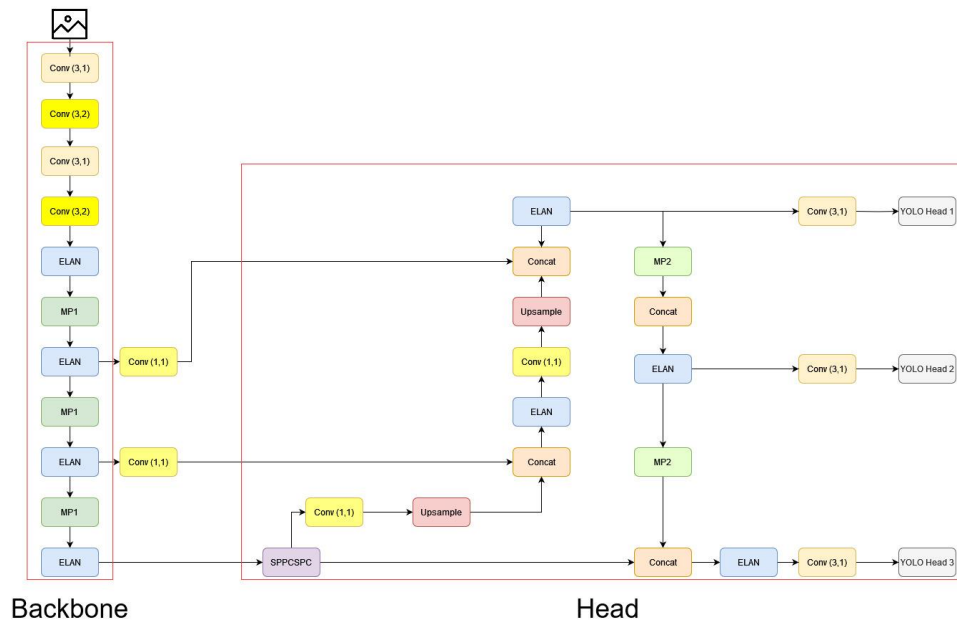


Gambar 2.20 *Layer*_Modul MP pada YOLOv7

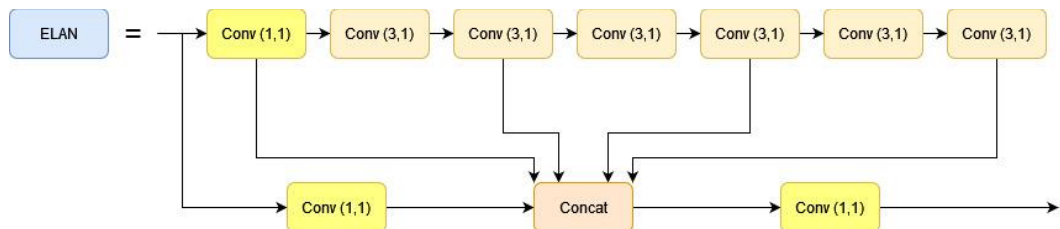
C. YOLOv7-X

YOLOv7-X merupakan varian model YOLOv7 yang paling kompleks pada tipe YOLOv7-P5. YOLOv7-X terbentuk dengan melakukan modifikasi varian YOLOv7 pada bagian *neck* dan menambahkan metode *compound scaling* pada model. Secara umum, arsitektur YOLOv7 dan YOLOv7-X tidak memiliki perbedaan yang signifikan. Perbedaan kedua model tersebut hanya pada arsitektur ELAN yang digunakan dimana YOLOv7-X memiliki lebih banyak layer pada arsitektur ELAN-nya dan tidak terbagi menjadi ELAN untuk bagian *backbone* dan

head. Modul lain seperti SPPCSPC dan MP tidak memiliki perbedaan dengan model YOLOv7. Berikut merupakan visualisasi arsitektur YOLOv7-X



Gambar 2.21 Arsitektur Model YOLOv7-X



Gambar 2.22 Arsitektur ELAN Model YOLOv7-X

2.5. Fishnet

Fishnet merupakan *dataset* yang terdiri dari berkas citra kumpulan ikan dari kamera pengawasan pada kapal yang sudah dianotasi dengan *bounding boxes* untuk tiap objeknya. *Dataset* ini terdiri dari 34 kelas objek yang mencakup spesies-spesies ikan serta manusia yang didapatkan dari beberapa pihak perikanan dan kelautan. Versi terbaru *dataset* fishnet memiliki 143.818 citra dengan 549.209 *bounding boxes* di mana terdapat 4 *bounding boxes* di setiap citra pada *dataset*. Berkas *dataset* fishnet sendiri memiliki ukuran total sebesar 30 GB [15].

Dataset fishnet terbagi menjadi tiga bagian, yaitu *train*, *validation*, *test*. Ketiga bagian tersebut disesuaikan untuk proses pelatihan dan pengujian model *deep learning* yang ingin dibentuk. Perbandingan jumlah citra untuk *training* dan *validation-testing* adalah 7:3 di mana citra pada bagian *validation* dan *testing* dibagi kembali dengan perbandingan 7:3.

Tabel 2.1 Persebaran bagian *dataset* Fishnet

Nama Bagian	Jumlah Citra
<i>train</i>	100.673
<i>validation</i>	30.202
<i>Test</i>	12.943
Total	143.818

2.6. *Polygon annotation*

Polygon annotation adalah cara dalam pemberian label pada sebuah data untuk pembentukan model *machine learning* dengan menentukan koordinat x dan y dari tiap ujung objek yang ingin dideteksi[16]. Dengan menentukan koordinat x dan y dari tiap ujung objek, bentuk label yang dihasilkan akan memiliki tingkat presisi yang sama objek yang sesungguhnya. Karena hal tersebut, metode *polygon annotation* umum digunakan sebagai metode pemberian anotasi pada *dataset* model *object detection* atau *object recognition* di kehidupan nyata seperti pada CT scan, memperhatikan pertumbuhan sebuah tanaman, perhitungan biaya reparasi kerusakan kendaraan, dan lain-lain.



Gambar 2.23 Contoh data *Polygon annotation* pada kerusakan sebuah mobil[16]

Polygon annotation memiliki beberapa keunggulan seperti fleksibel ketika diterapkan pada sebuah bentuk yang tidak biasa dan membuat piksel yang bukan

bagian dari objek tidak diikutkan pada label sehingga hasil pendeteksian bisa lebih baik. Namun, waktu yang dibutuhkan untuk melakukan anotasi lebih lama sesuai dengan kompleksitas dari objek yang dianotasi dan tidak semua *annotator* memiliki fitur untuk melakukan *polygon annotation* pada objek yang memiliki lubang seperti ban, donat dan lain-lain.

2.7. Penelitian Terkait

Belum ada penelitian yang membahas terkait model YOLOv7 dalam melakukan pendeteksian ikan, tetapi terdapat beberapa jurnal yang menggunakan model YOLO untuk melakukan pendeteksian ikan. Salah satu penelitian adalah dengan model YOLOv3-tiny dan YOLOv4-Tiny melakukan uji coba untuk *dataset* Fish-Gres yang merupakan *dataset* citra ikan yang diambil dari sebuah pasar di kabupaten Gresik untuk mendeteksi kepala dan ekor ikan pada tahun 2021. Penelitian tersebut membandingkan hasil *precision*, *recall* dan *F1-Score* dari tiga varian model YOLOv3-Tiny, YOLOv4-Tiny dan YOLOv4-Tiny dengan *Spatial Pyramid Pooling* di mana model dilatih menggunakan 200 citra dengan 160 citra untuk *training* dan 40 citra untuk *testing*.

Tabel 2.2 Hasil *precision*, *recall* dan *F1-Score* model YOLOv3-Tiny, YOLOv4-Tiny dan YOLOv4-Tiny dengan *Spatial Pyramid Pooling* pada *dataset* Fish-Gres dalam pendeteksian kepala dan ekor ikan [14].

Objek	Sesi	Metrics	YOLOv3-Tiny	YOLOv4-Tiny	YOLOv4-Tiny-SPP
Head	Training	<i>Precision</i>	88.16	91.99	92.31
		<i>Recall</i>	79.36	87.48	91.37
		<i>F1-Score</i>	83.53	89.68	91.84
	Validation	<i>Precision</i>	85.19	91.23	86.89
		<i>Recall</i>	69.70	78.79	80.30
		<i>F1-Score</i>	76.67	84.55	83.47
Tail	Training	<i>Precision</i>	85.64	89.85	91.73
		<i>Recall</i>	66.80	73.44	78.22
		<i>F1-Score</i>	75.06	80.82	84.44
	Validation	<i>Precision</i>	75.47	81.54	77.46
		<i>Recall</i>	38.83	51.46	53.40
		<i>F1-Score</i>	51.28	63.10	63.22
All	Training	<i>Precision</i>	87.11	91.11	92.07
		<i>Recall</i>	59.30	65.29	68.74
		<i>F1-Score</i>	70.56	76.07	78.71
	Validation	<i>Precision</i>	81.99	87.71	83.42
		<i>Recall</i>	56.17	66.81	68.51
		<i>F1-Score</i>	66.67	75.85	75.23

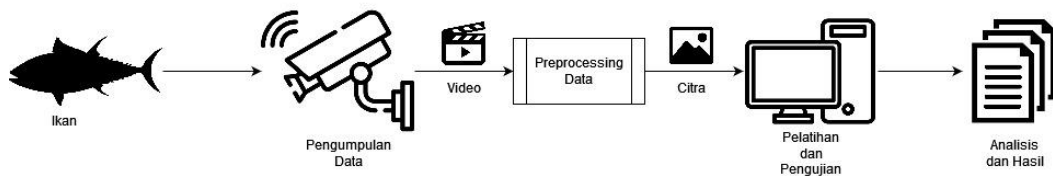
Setelah dilakukan observasi lebih lanjut, penulis menemukan dua kelemahan dari jurnal ini. Pertama, *dataset* yang digunakan kurang representatif dikarenakan hanya menggunakan 200 citra yang tidak bisa menjadi tolak ukur performa dari model yang digunakan. Kedua, nilai *precision*, *recall* dan *F1-Score* saat *validation* mengalami penurunan yang cukup tinggi sehingga terdapat indikasi bahwa model yang terbentuk belum sepenuhnya berhasil membedakan data selain data yang digunakan saat *training*.

BAB 3

PERANCANGAN SISTEM DETEKSI MULTI OBJEK BERBASIS YOLOV7

3.1. *System requirements*

Sistem pendeteksian ikan yang akan digunakan terdiri dari dua komponen utama, yaitu sebuah kamera dan komputer. Sistem ini akan menangkap objek dari kamera yang terpasang pada kapal dan akan menjadi masukan dari proses pendeteksian ikan. Objek akan ditangkap secara *real-time* dan akan diproses menggunakan sebuah komputer yang didalamnya sudah terpasang sebuah model kecerdasan buatan berbasis bahasa pemrograman Python yang akan mengolah objek yang tertangkap. Model ini kemudian akan membagi citra yang tertangkap menjadi beberapa bagian, mengekstrak fitur dari tiap bagian dan membentuk prediksi dari objek di tiap bagian yang terbentuk menggunakan *bounding boxes*. Setelah itu, akan dicari *bounding boxes* dengan nilai prediksi tertinggi dengan kelas dari objek tersebut yang menjadi representasi keluaran dari model. Rancangan sistem diilustrasikan secara visual pada Gambar 3.1.



Gambar 3.1 Rancangan sistem pendeteksian ikan

Model kecerdasan buatan yang akan dilatih, diuji dan dijalankan menggunakan sebuah komputer. Spesifikasi dari komputer yang digunakan untuk membangun model tercantum pada Tabel 3.1.

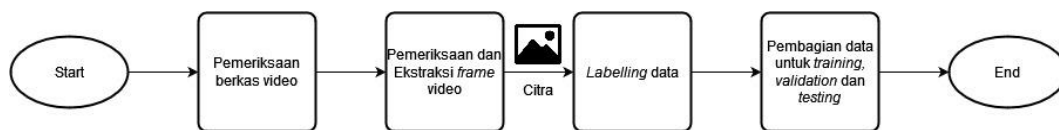
Tabel 3.1 Spesifikasi Komputer

<i>Central Processor Unit</i>	Intel i5-13600K @ 3.50 GHz (20 CPUs)
<i>Graphics Processing Unit</i>	NVIDIA GeForce RTX 3070
<i>Memory</i>	32 GB DDR5
<i>Operating System</i>	EndeavourOS Arch Linux
<i>Python Version</i>	Python 3.8.16

3.2. Pengumpulan Data

Data yang akan digunakan dalam proses pelatihan dan pengujian varian model YOLOv7 diambil dari hasil tangkapan kamera yang terpasang pada kapal. Hasil tangkapan akan berbentuk sebuah berkas video yang disimpan pada memori penyimpanan berupa *SD Card* yang nantinya akan diteruskan menuju tahap *pre-processing* untuk diverifikasi. Data yang akan digunakan harus mencakup enam jenis objek yang akan dideteksi oleh sistem. Jika data tiap jenis ikan belum mencukupi, kekurangan data akan diambil dari *dataset* Fishnet. Data yang diambil berupa sebuah citra yang akan disesuaikan dengan jenis ikan yang belum memiliki data yang cukup. Nantinya, data tersebut akan digabungkan dengan data yang berasal dari berkas video hasil rekaman.

3.3. Preprocessing Data

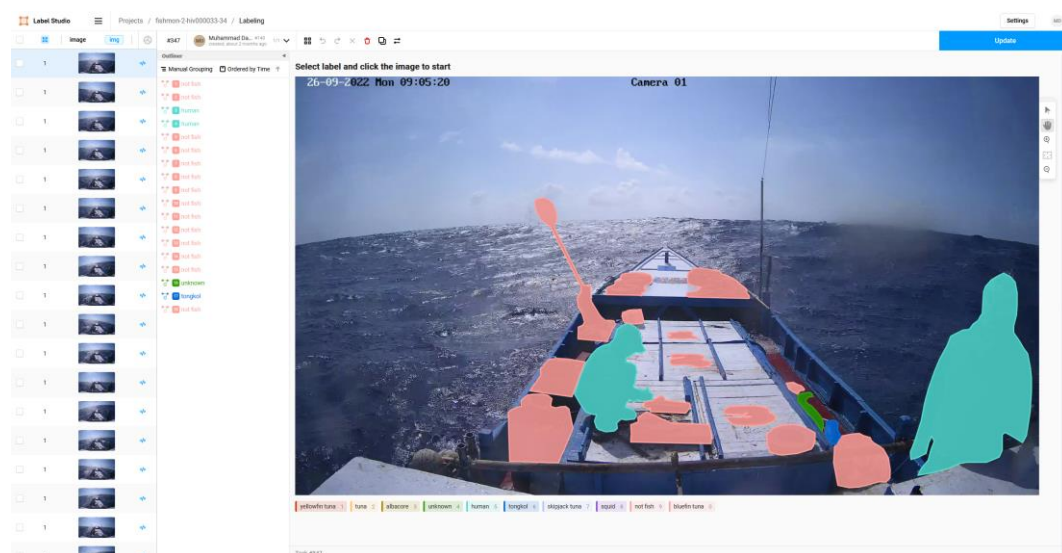


Gambar 3.2 Tahapan *pre-processing* data

Data yang telah dikumpulkan perlu disesuaikan terlebih dahulu agar dapat digunakan sebagai data masukan dalam proses pelatihan maupun pengujian melalui tahap *preprocessing*. Terdapat beberapa proses yang dilakukan pada tahap *preprocessing*. Pertama, berkas video dari hasil tangkapan kamera akan diperiksa terlebih dahulu untuk memastikan apakah berkas tersebut mengalami *corrupt* atau tidak. Selain itu, pemeriksaan juga dilakukan untuk memastikan apakah di berkas tersebut terdapat salah satu dari enam jenis objek yang ingin dideteksi atau tidak. Jika berkas video tersebut mengalami *corrupt* atau tidak terdapat enam jenis objek yang ingin dideteksi, berkas video tersebut akan dipisahkan dan tidak digunakan sebagai data pelatihan dan pengujian.

Setelah semua berkas video telah diperiksa, berkas yang tidak mengalami *corrupt* dan terdapat salah satu dari enam jenis objek di dalamnya akan kembali diperiksa. Pemeriksaan dilakukan dengan memeriksa tiap *frame* yang ada berkas

video memiliki objek berupa ikan atau tidak. *Frame* video yang memiliki objek ikan akan diekstrak menjadi sebuah citra dalam format .jpg. Kemudian, citra tersebut akan dianotasi agar tiap objek yang terdapat pada citra memiliki identitasnya. Proses *labelling* dilakukan pada *framework* Label Studio yang merupakan salah satu *framework labelling* bersifat *open source*. Nantinya tiap citra akan dibagi menjadi beberapa proyek untuk mempermudah identifikasi asal berkas video dari tiap citra. Data akan dianotasi menggunakan metode *polygon annotation*. Metode tersebut digunakan untuk mendapatkan tingkat presisi kontur yang sama dengan objek yang ingin dideteksi sehingga tidak ada piksel yang tidak berkaitan dengan objek masuk dalam hasil anotasi. Selain itu, data *polygon annotation* juga lebih mudah untuk dikonversi menjadi data *bounding box* karena nilai parameter yang ada dapat digunakan untuk membentuk label *bounding box*.



Gambar 3.3 Proses *labelling* dengan Label Studio [17]

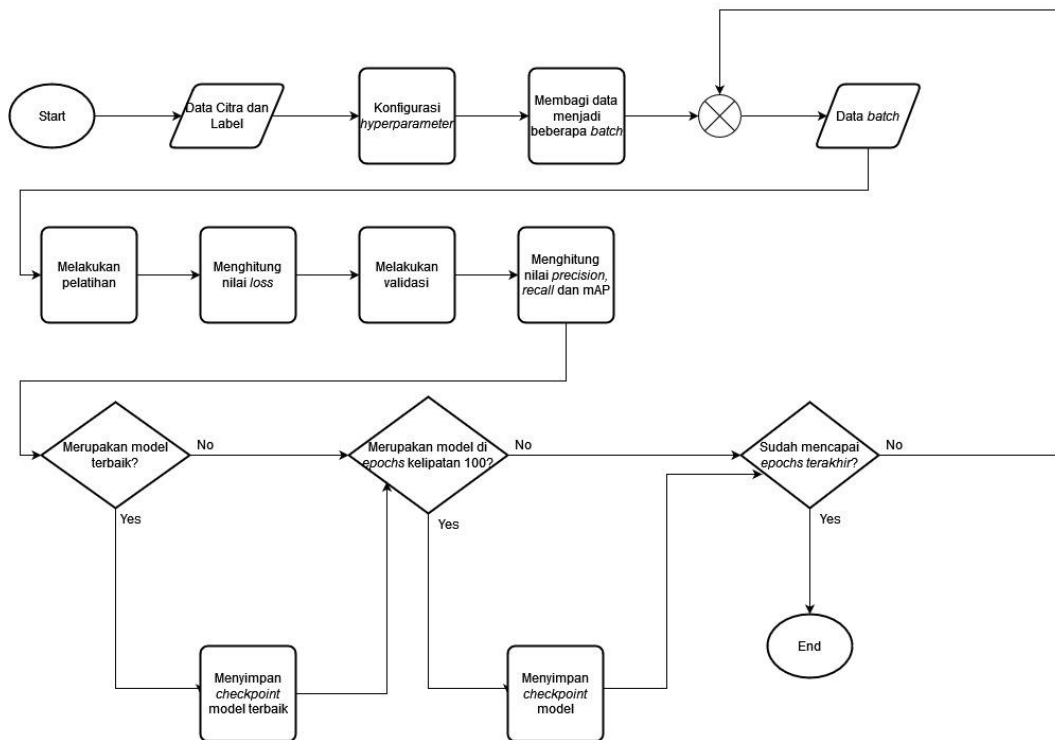
Setelah semua citra telah dilabeli, citra yang ada di Label Studio akan diekspor menjadi format *dataset* YOLO. *Dataset* tersebut kemudian akan dibagi menjadi untuk proses *training*, *validation* dan *testing*. Dalam proses ini juga, *dataset* akan diperiksa untuk memastikan apakah berkas citra untuk tiap proses telah mencukupi atau belum. Jika belum mencukupi, berkas citra akan ditambahkan dari *dataset* Fishnet dengan label yang disesuaikan.

3.4. Rancangan Proses Pelatihan Model

Pelatihan model dilakukan secara berulang terhadap data citra dan label yang digunakan. Setelah citra disesuaikan pada tahap *preprocessing*, pelatihan dimulai dengan melakukan konfigurasi *hyperparameter* yang akan digunakan saat proses *training*. Beberapa *hyperparameter* yang dapat dikonfigurasi seperti *learning rate*, *IoU threshold*, *optimizers momentum*, dan lain-lain. Setelah selesai konfigurasi, data akan dibagi menjadi beberapa *batch* sesuai dengan *batch size* yang diatur pada parameter pelatihan. Semakin besar *batch size* yang digunakan, semakin banyak memori yang dibutuhkan oleh GPU untuk melakukan pelatihan.

Proses pelatihan model akan dilakukan melalui dua tahap. Tahap pertama, model akan dilatih menggunakan *dataset* pelatihan. Tahapan ini dilakukan untuk mencari nilai bobot yang sesuai untuk melakukan pendeteksian ikan. Hasil dari tahapan ini adalah nilai *loss* yang terbagi menjadi tiga, yaitu *box_loss* untuk *loss* dari *bounding boxes* pada citra, *cls_loss* untuk *loss* dari kelas yang dilatih serta *obj_loss* untuk *loss* dari objek yang ada pada citra. Kemudian, model akan memasuki tahapan validasi dimana model akan menggunakan *dataset* pada bagian ‘*validation*’ untuk melihat apakah model yang telah terbentuk pada proses pelatihan dapat melakukan pendeteksian pada data yang belum pernah dilihat. Hasil dari tahapan ini adalah beberapa *evaluation metrics* seperti *precision*, *recall* dan mAP.

Setelah melalui kedua tahapan tersebut, program akan melakukan pemeriksaan terhadap model yang terbentuk. Jika model tersebut merupakan model dengan nilai bobot yang terbaik, model tersebut akan tersimpan sebagai *checkpoint* model dengan bobot terbaik. Kemudian, akan dilakukan pemeriksaan kepada model yang terbentuk dari *epochs* kelipatan 100. Jika model tersebut terbentuk pada *epochs* kelipatan 100, model tersebut akan tersimpan sebagai *checkpoint* model setiap 100 *epochs*. Proses pelatihan akan berhenti jika sudah mencapai *epochs* terakhir sesuai dengan parameter yang terpasang sebelum proses pelatihan. Jika kondisi tersebut belum terpenuhi, proses pelatihan akan berlanjut hingga mencapai *epochs* terakhir. Proses pelatihan model yang dilakukan divisualisasikan pada Gambar 3.4.



Gambar 3.4 Skema proses pelatihan model YOLOv7

3.5. Rancangan Proses Pengujian Model

Skenario pengujian model YOLOv7 akan dilakukan dengan memuat model pada GPU dan akan melakukan prediksi terhadap *dataset* pengujian yang berbeda dengan *dataset* pelatihan dan validasi. Terdapat beberapa parameter yang dapat disesuaikan untuk membentuk beberapa skenario pengujian seperti *confidence threshold*, *IoU threshold* dan *batch size*. Hasil prediksi akan berbentuk sebuah grafik yang menggambarkan performa dari model tersebut seperti grafik *confusion matrix*, *F1-Score*, *precision-recall Curve* dan nilai IoU. Setelah diuji pada berkas citra, pengujian akan dilakukan pada berkas video yang didalamnya terdapat salah satu dari jenis ikan yang ingin dideteksi. Hasil prediksi akan berbentuk sebuah *bounding boxes* dengan label dari kelas objek yang dilatih.

Terdapat beberapa kriteria utama yang digunakan untuk melakukan perbandingan model YOLOv7-Tiny, YOLOv7 dan YOLOv7-X. Kriteria pertama adalah mAP yang menggambarkan bagaimana model tersebut melakukan pendeteksian pada berkas citra atau video. Nilai mAP perlu dibandingkan karena

sebuah sistem pendeteksian yang baik harus bisa membedakan sebuah objek di berbagai skenario dengan kondisi lingkungan pendeteksian yang tidak menentu yang dapat digambarkan oleh model dengan nilai mAP yang tinggi. Kriteria kedua adalah FPS yang menggambarkan seberapa cepat model tersebut melakukan pendeteksian pada sebuah objek. FPS perlu dibandingkan karena model akan diterapkan pada sebuah sistem *real-time* yang akan melakukan pendeteksian dalam kurun waktu milisekon. Jika sebuah model memiliki mAP yang baik, tetapi tidak memiliki FPS yang tinggi, model tersebut kurang cocok untuk diterapkan pada sebuah sistem *real-time* sehingga dibutuhkan model yang memiliki mAP yang baik dengan FPS yang tinggi untuk bisa diterapkan pada sistem *real-time* di kehidupan nyata. Kriteria ketiga adalah *F1-Score* yang menggambarkan harmonisasi antara nilai *precision* dan *recall* dari sebuah kelas. *F1-Score* perlu dibandingkan karena jika sebuah model memiliki *F1-Score* yang rendah, nilai *precision* dan *recall* dari model tersebut tidak memiliki harmonisasi yang baik sehingga memiliki kemungkinan untuk model tersebut melakukan kesalahan dalam proses pendeteksian sehingga sebuah model juga perlu memiliki nilai *F1-Score* yang baik.

BAB 4

IMPLEMENTASI DAN ANALISIS SISTEM DETEKSI MULTI OBJEK BERBASIS YOLOV7

4.1. Implementasi Sistem

Untuk membentuk sistem pendeteksian ikan, diperlukan proses pengembangan model *machine learning*. Model *machine learning* dikembangkan menggunakan algoritma YOLOv7 dengan menggunakan PC pribadi penulis dengan spesifikasi yang tertera pada Tabel 3.1. Perangkat yang digunakan sudah terpasang CUDA dan CUDNN, yaitu *library* dari Nvidia yang berfungsi untuk mengaktifasi GPU dalam pembentukan model *machine learning* sehingga proses pembentukan model dapat berjalan lebih cepat dengan memanfaatkan *core* pada GPU serta CPU secara bersamaan.

Proses pembentukan model *machine learning* akan dilakukan menggunakan *framework* PyTorch. Terdapat 3 varian model YOLOv7 yang akan dilatih untuk dibandingkan sebelum dipasang pada sistem, yaitu YOLOv7-Tiny, YOLOv7 dan YOLOv7-X. Model yang akan dipasang pada sistem akan ditentukan melalui beberapa skenario pengujian.

Pengujian model *machine learning* akan dilakukan pada dua perangkat, yaitu PC pribadi penulis dan Jetson Nano. Pengujian pada Jetson Nano dilakukan untuk mendapatkan gambaran performa dari model ketika diaplikasikan pada perangkat yang akan terpasang pada kapal untuk melakukan pendeteksian berupa mini-pc. Rincian spesifikasi Jetson Nano untuk pengujian tertera pada Tabel 4.1.

Tabel 4.1 Spesifikasi Jetson Nano untuk Pengujian Model

<i>Central Processor Unit</i>	Quad-Core ARM [®] Cortex [®] -A57 MPCore processor (4 CPUs)
<i>Graphics Processing Unit</i>	NVIDIA Maxwell [™] architecture
<i>Memory</i>	4 GB 64-bit LPDDR4
<i>Operating System</i>	Ubuntu 18.04
<i>Python Version</i>	Python 3.6.9

4.2. Dataset Pengujian

Pengujian model dilakukan menggunakan *dataset* berupa citra yang telah melewati proses pengambilan data dan *labelling*. Pengambilan data dilakukan menggunakan dua metode, yaitu pengambilan data dengan mengekstrak *frame* video hasil tangkapan kamera dengan objek yang ingin dideteksi serta mengambil dari *dataset* fishnet. Data tersebut akan diberikan label menggunakan *label studio* dengan memprioritaskan citra yang terdapat objek ikan didalamnya sebagai objek utama pendeteksian.

Setelah melalui proses *labelling*, citra tersebut akan dibagi menjadi tiga bagian untuk proses pelatihan dan pengujian model, yaitu bagian *training*, *validation* dan *testing*. Hal ini bertujuan untuk memastikan model yang sudah terbentuk tidak mengalami *overfitting* pada *dataset training* sehingga ketika terdapat data baru yang digunakan, model tersebut masih dapat mendeteksi dan mengklasifikasikan data tersebut. Secara keseluruhan, jumlah citra yang digunakan sebagai *dataset* pelatihan dan pengujian model adalah 4280 citra yang terdiri dari 6 jenis objek. Rincian dari persebaran citra serta jumlah data untuk tiap kelas terdapat pada tabel 4.2 dan tabel 4.3.

Tabel 4.2 Persebaran citra pada *dataset*

Nama Bagian	Jumlah Citra
<i>train</i>	2.999
<i>validation</i>	641
<i>Test</i>	640
Total	4.280

Tabel 4.3 Jumlah data tiap kelas pada *dataset*

Kelas	Jumlah Data		
	Training	Validation	Testing
<i>human</i>	7423	1566	1553
<i>tuna</i>	4734	1071	967
<i>Skipjack tuna</i>	262	58	42
Tongkol	447	98	94
<i>Squid</i>	564	93	121
<i>unknown</i>	2631	560	586
Total	16061	3446	3363

4.3. Skenario Pengujian

Dalam penelitian ini, ketiga varian model YOLOv7 yang dibandingkan akan diuji dengan beberapa skenario pengujian untuk mendapatkan model dengan kualitas pendeteksian dan performa yang terbaik. Pengujian dilakukan menggunakan *dataset* yang tertera pada subbab 4.2 dengan jumlah data yang sama sehingga ketiga varian tersebut dapat dibandingkan dengan skenario pengujian yang sama. *Dataset* yang akan digunakan dalam pengujian adalah *dataset* yang termasuk pada bagian ‘*test*’.

Pada penelitian ini, pengujian dilakukan pada tahap pelatihan model dan implementasi pada mini-pc sebagai perangkat yang digunakan di dunia nyata. Tahap pelatihan model akan dilakukan pada tiap varian model YOLOv7 yang dibandingkan untuk mendapatkan kualitas dan performa tiap varian dalam melakukan pendeteksian. Pelatihan yang dilakukan pada tiap varian akan menggunakan *batch size* yang berbeda dikarenakan kompleksitas arsitektur yang berbeda sehingga pelatihan model dapat dilakukan tanpa melebihi kapabilitas perangkat yang digunakan.

Pengujian pada saat pelatihan model dilakukan dengan beberapa komponen pengujian diantaranya adalah bentuk *dataset*, teknik pelatihan model serta parameter yang digunakan dalam pelatihan seperti ukuran gambar, *optimizers* dan jumlah *epochs*. Kualitas dari hasil pelatihan model yang terbentuk akan ditentukan menggunakan mAP dan *F1-Score*. Kedua kriteria tersebut umum digunakan dalam menentukan kualitas model deteksi objek karena perhitungannya yang dapat merepresentasikan akurasi model secara menyeluruh dengan tidak hanya mencakup nilai *true positive* saja, tetapi juga mencakup nilai *false positive* dan *false negative* sebagai nilai yang merepresentasikan kesalahan model dalam melakukan pendeteksian.

Kemudian, pengujian pada performa model dilakukan menggunakan *inference time*. *Inference time* adalah satuan untuk menghitung waktu yang dibutuhkan model deteksi objek dalam melakukan pendeteksian. *Inference time* berkaitan dengan jumlah FPS yang model tersebut dapat hasilkan sehingga semakin

kecil *inference time* dari model, semakin banyak jumlah FPS yang dapat dihasilkan dan diproses.

Setelah pengujian pada saat pelatihan model, model yang terbaik akan diuji pada mini-pc sebagai perangkat yang akan digunakan ketika diimplementasikan di dunia nyata. Hal ini bertujuan untuk membandingkan apakah terdapat penurunan baik secara kualitas maupun performa model dalam melakukan pendeteksian ketika perangkat yang digunakan memiliki proses komputasi yang lebih rendah dibandingkan perangkat untuk pelatihan model.

4.4. Pengujian Model

Berdasarkan skenario pengujian yang telah dijelaskan pada subbab 4.3, berikut adalah hasil dari tiap pengujian yang dilakukan:

4.4.1. Pengujian Pengaruh Data Berbentuk *Polygon annotation* Terhadap Akurasi dan Jumlah FPS

Pada pengujian ini, tiap varian model YOLOv7 akan memasuki tahap pelatihan model dengan bentuk data yang berbeda. Hal ini bertujuan untuk mengetahui apakah bentuk data yang terdapat pada *dataset* dapat memiliki pengaruh yang signifikan terhadap nilai akurasi yang dimiliki oleh model. Bentuk data yang akan dibandingkan adalah data yang berbentuk *polygon annotation* dengan data yang berbentuk *bounding box*. Sebelum memulai pelatihan dan pengujian model, data pada *dataset* pengujian yang tercantum pada tabel 4.3 perlu dikonversi terlebih dahulu untuk bisa mendapatkan *dataset* pengujian dengan data berbentuk *bounding box* dikarenakan proses *labelling* dilakukan dengan metode *polygon annotation*. Data berbentuk *bounding box* terdiri dari 4 komponen, yaitu titik tengah objek pada sumbu x dan y, tinggi objek dan lebar objek. Oleh karena itu, label dari tiap citra akan diperiksa untuk mencari nilai maksimum serta minimum objek pada sumbu x dan y. Untuk mengonversi *dataset* pengujian gunakan kode berikut yang berfungsi untuk mencari nilai maksimum dan minimum objek yang nanti akan disimpan untuk mencari komponen data berbentuk *bounding box*.

```

data_dir = ""
for img_file in os.listdir(data_dir):
    img_file_id = '.'.join(img_file.split('.')[:-1])
    # Load instance segmentation labels
    seg_path = os.path.join(data_dir, img_file_id + ".txt")
    if not os.path.exists(seg_path):
        continue
    else:
        with open(seg_path, "r") as f:
            labels = f.readlines()

    new_labels = []
    for label in labels:
        label = label.strip().split()
        object_class = label[0]
        vertices = label[1:]

        x_values = [float(vertices[i]) for i in range(0, len(vertices), 2)]
        y_values = [float(vertices[i+1]) for i in range(0, len(vertices), 2)]

        # Calculate the minimum bounding box that encompasses the polygon vertices
        xmin = min(x_values)
        xmax = max(x_values)
        ymin = min(y_values)
        ymax = max(y_values)

        x, y, w, h = (xmin + xmax) / 2.0, (ymin + ymax) / 2.0, xmax - xmin,
ymax - ymin

        new_labels.append(f"{object_class} {x} {y} {w} {h}")

    save_dir = os.path.join(data_dir, "result")
    with open(os.path.join(save_dir, img_file_id + ".txt"), 'w') as f:
        f.write('\n'.join(new_labels))

```

Gambar 4.1 Kode untuk Mengonversi *Dataset* dari *Polygon annotation* menjadi *Bounding Box*

Konversi dilakukan dengan menggabungkan data citra dan label dari tiap bagian *dataset*. Kemudian, program akan memastikan bahwa terdapat pasangan antara citra dengan labelnya sehingga jika tidak memiliki label, citra tersebut akan dilewati. Lalu, isi dari label tersebut akan dibaca tiap barisnya untuk mengidentifikasi objek apa saja yang terdapat pada citra beserta dengan koordinat sumbu x dan y tiap ujung objek. Data tersebut kemudian akan diproses untuk dicari nilai maksimum dan minimumnya yang akan digunakan untuk membentuk data *bounding box*. Kemudian, data *bounding box* akan disimpan pada berkas baru sehingga tidak akan menghilangkan data label yang sebelumnya.

Setelah semua data terkonversi, proses pelatihan model akan dilakukan. Ketiga varian model YOLOv7 (YOLOv7-Tiny, YOLOv7, YOLOv7-X) akan dilatih menggunakan kedua bentuk data. Kemudian, model tersebut akan diuji dan dibandingkan untuk menentukan data yang akan digunakan untuk pengujian berikutnya. Pelatihan model akan menggunakan citra berukuran 640-piksel dari *dataset* bagian *train* dengan *learning rate* sebesar 10^{-2} , diiterasi sebanyak 300 kali dengan *optimizers* SGD untuk mengoptimalkan hasil dari pelatihan model. *Batch size* yang digunakan akan menyesuaikan dengan kapabilitas dari perangkat pelatihan untuk setiap varian model. Kemudian, model yang terbentuk akan diuji menggunakan *dataset* bagian *test* dengan *batch size* 1. Pengujian akan menggunakan ukuran citra yang sama dengan pelatihan model dengan *IoU Threshold* sebesar 0.6 dan *Confidence Threshold* sebesar 0.001.

Tabel 4.4 Hasil Pengujian Model dengan *Dataset* Berbentuk *Polygon annotation* dan *Bounding Box*

No	Varian Model	Bentuk Data	mAP@0.5	mAP@0.5:0.95	F1-Score	Jumlah FPS	Inference Time (ms)
1	YOLOv7-Tiny	<i>Polygon annotation</i>	0.763	0.449	0.76	416.7	2.4
2		<i>Bounding Box</i>	0.77	0.455	0.77	400	2.5
3	YOLOv7	<i>Polygon annotation</i>	0.823	0.53	0.81	106.4	9.4
4		<i>Bounding Box</i>	0.809	0.525	0.81	100	10
5	YOLOv7-X	<i>Polygon annotation</i>	0.807	0.528	0.8	64.1	15.6
6		<i>Bounding Box</i>	0.818	0.533	0.8	60.2	16.6

Dari hasil pada tabel 4.4, dapat terlihat bahwasanya baik data berbentuk *polygon annotation* maupun *bounding box* memiliki hasil yang kurang lebih mirip. Dari segi akurasi, tidak ada perbedaan yang signifikan antara model yang dilatih dengan data berbentuk *polygon annotation* dengan *bounding box*. Selisih dari nilai mAP dan *F1-Score* untuk setiap varian model yang dilatih juga sangat kecil yang tidak mencapai 2% sehingga secara hasil pendeteksian, kedua model ini akan menghasilkan pendeteksian yang mirip. Akan tetapi, jika diteliti lebih lanjut pada

hasil pengujian, model yang dilatih dengan data *polygon annotation* memiliki tingkat pendeteksian yang sedikit lebih baik dibandingkan dengan data *bounding box*.



Gambar 4.2 Hasil Pengujian Model YOLOv7 pada Data *Bounding Box*



Gambar 4.3 Hasil Pengujian Model YOLOv7 pada Data *Polygon annotation*

Sebagai contoh, pada gambar 4.2 dan 4.3, terlihat bahwasanya model YOLOv7 yang dilatih dengan data *bounding box* mengidentifikasi kelas ‘tuna’ sebagai kelas ‘human’ yang termasuk dalam *false negative*, sedangkan model yang dilatih dengan data *polygon annotation* dapat memprediksi dengan tepat ‘tuna’ tersebut, tetapi memiliki satu nilai *false positive* pada kelas ‘human’. Dari gambar tersebut, terlihat

bahwa kedua model tersebut sama-sama melakukan kesalahan dalam pendeteksian objek, tetapi model yang dilatih dengan data *polygon annotation* masih dapat mengidentifikasi seluruh objek yang seharusnya terdeteksi pada citra, sedangkan model yang dilatih dengan data *bounding box* tidak bisa mengidentifikasi seluruh objek yang seharusnya terdeteksi. Oleh karena itu, model dengan data *polygon annotation* memiliki tingkat pendeteksian yang sedikit lebih baik dibandingkan model dengan data *bounding box*.

Kemudian, dari segi kecepatan pendeteksian, model yang dilatih dengan data *polygon annotation* mendeteksi lebih cepat dibandingkan dengan model yang dilatih dengan data *bounding box*. Walaupun secara peningkatan tidak mencapai 10%, dengan jumlah frame yang dihasilkan lebih banyak, model tersebut secara kapabilitas mampu untuk melakukan pendeteksian yang lebih banyak tiap detiknya sehingga tidak akan ada objek yang tidak terdeteksi oleh perangkat masukan. Berdasarkan hasil pengujian, dapat disimpulkan bahwa data berbentuk *polygon annotation* memiliki lebih banyak keunggulan dibandingkan data berbentuk *bounding box* seperti hasil prediksi yang sedikit lebih baik dan jumlah FPS yang lebih tinggi sehingga data *polygon annotation* akan digunakan pada pelatihan model untuk pengujian berikutnya.

4.4.2. Pengujian Perbandingan Metode *Object Detection* dengan *Instance segmentation* Terhadap Akurasi

Pengujian berikutnya dilakukan dengan membandingkan metode *object detection* dan *instance segmentation* yang digunakan dalam pelatihan model. Berdasarkan hasil pengujian sebelumnya, data berbentuk *polygon annotation* memiliki beberapa keunggulan dibandingkan dengan data berbentuk *bounding box*. Oleh karena itu, pengujian ini bertujuan untuk membandingkan performa pendeteksian model yang dilatih dengan dua metode yang berbeda menggunakan data berbentuk *polygon annotation*. *Instance segmentation* merupakan pengembangan dari metode *object detection* dimana model akan melakukan pendeteksian terhadap objek secara presisi dengan mencari ujung-ujung dari objek sehingga hasil pendeteksiannya akan berbentuk seperti objek tersebut. Oleh karena

itu, *instance segmentation* umum digunakan pada model yang membutuhkan tingkat presisi pendeteksian yang tinggi seperti pada sebuah kendaraan dengan fitur *autonomous driving*.

Proses pengujian dilakukan dengan melatih kembali ketiga varian model YOLOv7 menggunakan *dataset polygon annotation* bagian *train* dengan masing-masing metode pembentukan model. Pelatihan model juga akan menggunakan parameter yang sama dengan pengujian pertama kecuali pada bagian *batch size* yang menyesuaikan dengan varian model dan metode pembentukan agar tidak melebihi kapabilitas perangkat pelatihan. Kemudian, model yang terbentuk akan diuji dengan *dataset* bagian *test* untuk mendapatkan performa pendeteksian masing-masing model menggunakan parameter pengujian yang sama dengan pengujian pertama.

Tabel 4.5 Hasil Pengujian Model dengan Metode *Object Detection* dan *Instance segmentation*

No	Varian Model	Metode Pembentukan	Batch Size pelatihan	mAP@0.5	mAP@0.5 :0.95	F1-Score	Jumlah FPS	Inference Time (ms)
1	YOLOv7-Tiny	<i>Object Detection</i>	32	0.763	0.449	0.76	416.7	2.4
2		<i>Instance segmentation</i>	20	0.654	0.297	0.68	250.0	4
3	YOLOv7	<i>Object Detection</i>	7	0.823	0.53	0.81	106.4	9.4
4		<i>Instance segmentation</i>	5	0.741	0.353	0.76	83.3	12
5	YOLOv7-X	<i>Object Detection</i>	5	0.807	0.528	0.8	64.1	15.6
6		<i>Instance segmentation</i>	4	0.75	0.365	0.77	56.8	17.6

Dari hasil pada tabel 4.5, terlihat bahwa metode *object detection* memiliki hasil yang lebih baik dibandingkan *instance segmentation*. Dari segi performa, dapat terlihat bahwa *object detection* mengalami peningkatan nilai mAP@0.5 sebesar 7 – 15% dan nilai mAP@0.5:0.95 sebesar 45 – 55%. Selain itu, model

dengan metode *object detection* juga memiliki nilai *F1-Score* yang lebih baik sebesar 3 – 11%. Faktor yang paling berpengaruh dalam perbedaan tersebut adalah bagaimana *dataset* yang digunakan diproses oleh kedua metode tersebut. *Object detection* akan memroses *dataset* yang digunakan dengan memberikan label *bounding box* pada objek yang terdeteksi meskipun *dataset* tersebut berbentuk *polygon segmenation*, sedangkan *instance segmentation* memroses *dataset* tersebut dengan membentuk objek yang terdeteksi sepresisi mungkin.



Gambar 4.4 Hasil Pengujian Model YOLOv7-X dengan Metode *Instance segmentation*



Gambar 4.5 Hasil Pengujian Model YOLOv7-X dengan Metode *Object Detection*

Seperti pada gambar 4.4 dan 4.5, terlihat bahwasanya model YOLOv7-X dengan metode *object detection* tidak melakukan kesalahan dalam melakukan pendeteksian,

sedangkan model *instance segmentation* mengalami kesalahan *false positive* dengan mendeteksi kelas ‘unknown’. Hal tersebut mungkin terjadi dikarenakan bentuk pada citra yang terdeteksi sebagai ‘unknown’ memiliki pola yang sama dengan salah satu pola ‘unknown’ pada *dataset* bagian *train* sehingga pola tersebut dapat terdeteksi sebagai unknown. Oleh karena itu, model *instance segmentation* lebih mungkin untuk melakukan kesalahan sehingga memiliki nilai mAP dan *F1-Score* yang lebih rendah dikarenakan pendeteksian yang membutuhkan tingkat presisi yang tinggi untuk menyesuaikan pola dari objek yang akan dideteksi.

Kemudian, jika dilihat dari segi kecepatan pendeteksian, model *object detection* juga memiliki hasil yang lebih baik dibandingkan dengan model *instance segmentation*. Model *object detection* bisa memiliki nilai *inference time* dan jumlah FPS yang lebih baik hingga 66,7% dibandingkan model *instance segmentation*. Faktor yang menyebabkan hal tersebut juga berkaitan dengan pemrosesan *dataset* yang digunakan. Dikarenakan *instance segmentation* membutuhkan tingkat presisi yang tinggi dalam melakukan pendeteksian, dibutuhkan waktu yang lebih lama untuk mencari pola yang mirip dengan objek pada *dataset* dan memastikan bahwa pola tersebut merupakan objek yang ingin dideteksi dibandingkan dengan *object detection* yang hanya perlu menyesuaikan *bounding box* dari objek yang ingin dideteksi. Oleh karena itu, model *instance segmentation* lebih mungkin untuk memiliki *inference time* dan jumlah FPS yang lebih kecil dibandingkan dengan model *object detection*. Berdasarkan hasil pengujian, diketahui bahwa metode *object detection* lebih baik dalam melakukan pendeteksian dibandingkan dengan metode *instance segmentation* sehingga metode *object detection* akan digunakan pada pelatihan model untuk pengujian berikutnya

4.4.3. Pengujian Pengaruh Ukuran Gambar Terhadap Akurasi dan Jumlah FPS

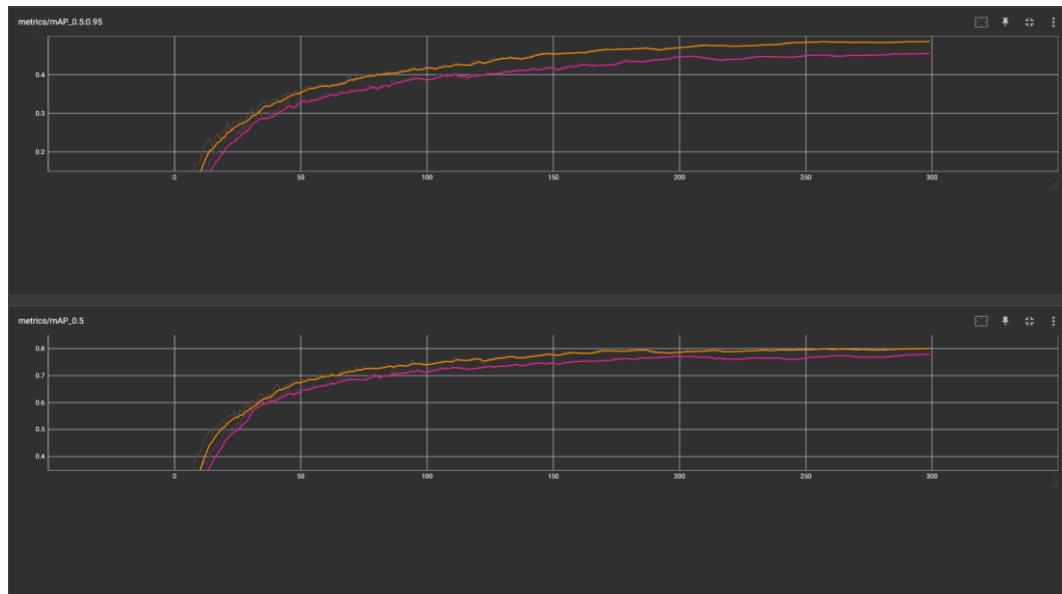
Pengujian berikutnya dilakukan dengan membandingkan ukuran citra yang digunakan untuk pelatihan model. Pengujian ini bertujuan untuk mengetahui apakah ukuran citra yang digunakan pada pelatihan model dapat berpengaruh terhadap model yang terbentuk. Pengujian dilakukan dengan melakukan pelatihan

model menggunakan ukuran citra yang berbeda, yaitu ukuran 640 piksel dan 800 piksel untuk setiap varian model dengan bagian *dataset* dan parameter yang sama seperti pengujian sebelumnya terkecuali pada *batch size* yang menyesuaikan dengan kapabilitas perangkat. Kemudian, pengujian akan dilakukan menggunakan parameter yang sama dengan pengujian sebelumnya terkecuali pada ukuran citra yang menyesuaikan dengan ukuran yang digunakan saat pelatihan model. Model dengan hasil terbaik akan digunakan pada pengujian berikutnya.

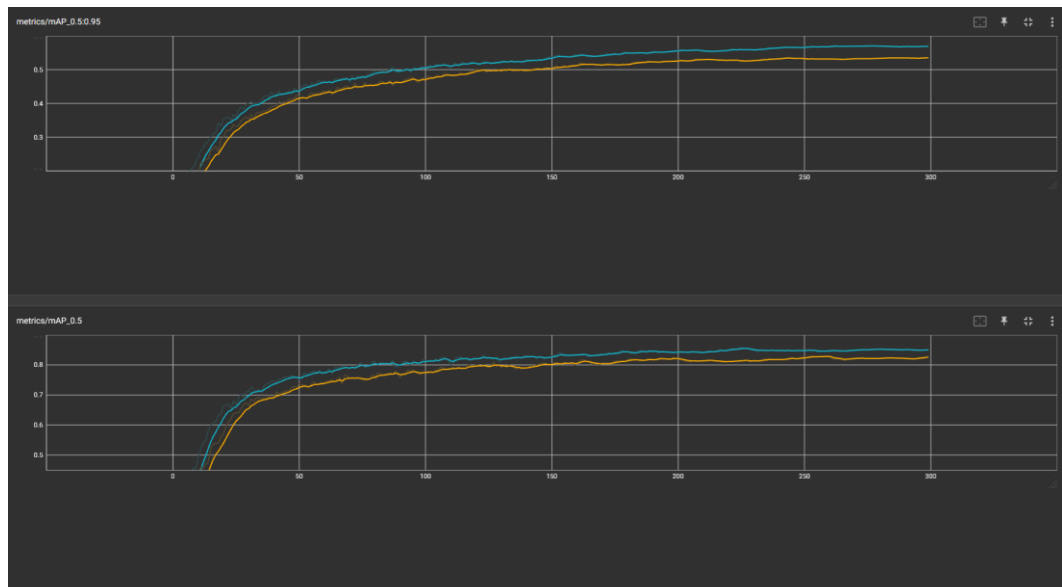
Tabel 4.6 Hasil Pengujian Model dengan Ukuran Citra 640 Piksel dan 800 Piksel

No	Varian Model	Ukuran Citra	Batch Size pelatihan	mAP@0.5	mAP@0.5: 0.95	F1-Score	Jumlah FPS	Inference Time (ms)	Ukuran berkas (MB)
1	YOLOv7-Tiny	640	32	0.763	0.449	0.76	416.7	2.4	12
2		800	19	0.789	0.48	0.79	312.5	3.2	
3	YOLOv7	640	7	0.823	0.53	0.81	106.4	9.4	72
4		800	4	0.843	0.557	0.82	69.0	14.5	
5	YOLOv7-X	640	5	0.807	0.528	0.8	64.1	15.6	139
6		800	3	0.852	0.571	0.83	38.8	25.8	

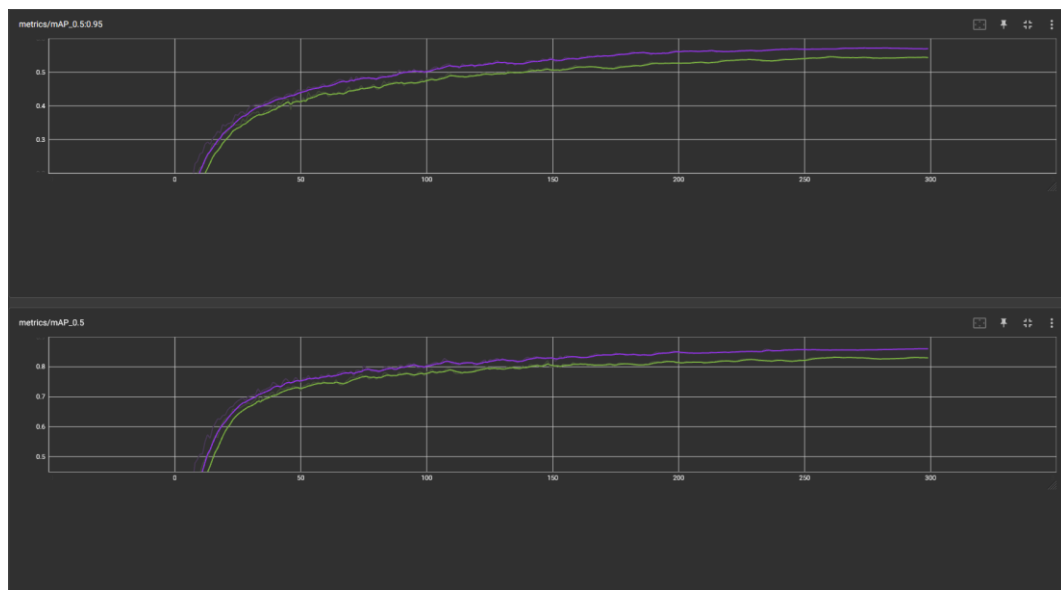
Dari hasil pada tabel 4.6, terlihat bahwasanya masing-masing ukuran citra memiliki keunggulannya masing-masing. Dari segi performa, terlihat bahwa dengan menaikkan ukuran citra menjadi 800 piksel pada pelatihan model, nilai mAP dan *F1-Score* dari model dapat meningkat walaupun tidak secara signifikan. Hal tersebut terjadi pada ketiga varian model yang mengalami peningkatan mAP dan *F1-Score* dalam rentang 2 – 8%. Jika dilihat pada tren peningkatan mAP, model dengan ukuran citra 800 piksel selalu memiliki nilai mAP yang lebih baik dibandingkan model dengan ukuran citra 640 piksel pada setiap iterasinya.



Gambar 4.6 Perbandingan Tren Peningkatan Nilai mAP pada YOLOv7-Tiny dengan Ukuran Citra 640 Piksel (Merah Muda) dan 800 Piksel (Jingga)



Gambar 4.7 Perbandingan Tren Peningkatan Nilai mAP pada YOLOv7 dengan Ukuran Citra 640 Piksel (Kuning) dan 800 Piksel (Biru Muda)



Gambar 4.8 Perbandingan Tren Peningkatan Nilai mAP pada YOLOv7-X dengan Ukuran Citra 640 Piksel (Hijau Muda) dan 800 Piksel (Ungu)

Terlihat pada gambar 4.6 – 4.8 yang merupakan grafik mAP untuk setiap iterasi, ketiga varian model yang dilatih menggunakan ukuran citra berukuran 800 piksel selalu memiliki nilai mAP yang lebih baik dibandingkan dengan menggunakan ukuran citra 640 piksel baik pada mAP@0.5 dan mAP@0.5:0.95. Peningkatan yang terlihat dari gambar 4.6 – 4.8 juga dapat dibuktikan dengan hasil pendeteksian pada data pengujian.



Gambar 4.9 Hasil Pengujian Model YOLOv7-Tiny dengan Ukuran Citra 640 Piksel (Kiri) dan 800 Piksel (Kanan)



Gambar 4.10 Hasil Pengujian Model YOLOv7 dengan Ukuran Citra 640 Piksel (Kiri) dan 800 Piksel (Kanan)



Gambar 4.11 Hasil Pengujian Model YOLOv7 dengan Ukuran Citra 640 Piksel (Kiri) dan 800 Piksel (Kanan)

Berdasarkan hasil pengujian pada gambar 4.9 – 4.11, dapat terlihat bahwa model yang dilatih menggunakan citra berukuran 800 piksel lebih mungkin untuk tidak salah mendeteksi dibandingkan dengan citra berukuran 640 piksel. Sebagai contoh, pada model YOLOv7-Tiny, terdapat 3 pendeteksian yang termasuk dalam *false positive* pada model dengan ukuran citra 640 piksel, sedangkan hanya terdapat 1 pendeteksian yang termasuk dalam *false positive* pada model dengan ukuran citra 800 piksel. Hal tersebut juga terjadi pada model YOLOv7 dimana model dengan ukuran citra 800 piksel tidak melakukan kesalahan dalam pendeteksian, sedangkan

model dengan ukuran citra 640 piksel memiliki satu pendeteksian yang termasuk sebagai *false positive*. Hanya pada model YOLOv7-X hasil yang didapatkan memiliki perbedaan.

Jika dianalisis, ukuran citra yang lebih besar membuat lebih banyak objek yang terlihat dan objek tersebut juga lebih mudah untuk dideteksi dikarenakan jumlah piksel yang lebih banyak. Hal tersebut berkaitan dengan nilai mAP yang lebih tinggi dimana nilai *precision* dan *recall* dari model dengan ukuran citra yang lebih besar akan lebih tinggi karena objek yang dideteksi akan lebih mudah untuk terdeteksi. Selain itu, model dengan ukuran citra yang lebih besar juga memiliki nilai *loss* yang lebih rendah dikarenakan lebih banyak fitur dari data yang berhasil diekstrak dan digunakan dalam pendeteksian sehingga kemungkinan model salah mendeteksi akan lebih kecil. Oleh karena itu, dengan menggunakan ukuran citra yang lebih besar, model yang terbentuk dapat memiliki akurasi yang lebih baik.

Dari segi kecepatan pendeteksian, model dengan ukuran citra 640 piksel memiliki kecepatan pendeteksian yang lebih baik dibandingkan dengan model dengan ukuran citra 800 piksel. Terlihat bahwa model yang terbentuk dengan ukuran citra 640 piksel memiliki peningkatan nilai *inference time* sebesar 25 – 40% dan jumlah FPS hingga 65% dibandingkan dengan 800 piksel. Hal tersebut dapat terjadi dikarenakan piksel yang diproses pada ukuran citra 640 piksel lebih sedikit dibandingkan ukuran citra 800 piksel sehingga kompleksitas model dalam melakukan pendeteksian akan lebih rendah. Oleh karena itu, ukuran citra yang lebih kecil dapat meningkatkan kecepatan pendeteksian dan jumlah FPS.

Berdasarkan hasil pengujian, dapat disimpulkan bahwa model dengan ukuran citra yang lebih besar memiliki akurasi pendeteksian yang lebih baik dan kecepatan pendeteksian yang lebih rendah, sedangkan model dengan ukuran citra yang lebih kecil memiliki akurasi pendeteksian yang lebih rendah dan kecepatan pendeteksian yang lebih baik. Dikarenakan model yang dibutuhkan adalah model dengan tingkat akurasi yang tinggi, akurasi dari model lebih diprioritaskan dibandingkan kecepatan pendeteksian. Oleh karena itu, varian model YOLOv7 dengan ukuran citra 800 piksel akan dipilih sebagai model yang digunakan pada

pengujian berikutnya dikarenakan memiliki akurasi yang baik dengan kecepatan pendeteksian yang cukup baik dan ukuran berkas yang tidak terlalu besar. Selain itu, varian model YOLOv7 dengan ukuran citra 800 piksel memiliki kemungkinan yang lebih kecil untuk salah mendeteksi dibandingkan dengan beberapa varian model lainnya jika dilihat berdasarkan hasil pengujian.

4.4.4. Pengujian Pengaruh Penambahan Epochs dalam Memicu *Early Stopping* Terhadap Akurasi dan Jumlah FPS

Pengujian berikutnya dilakukan dengan menambah jumlah *epochs* sebagai jumlah iterasi yang dilakukan dalam pelatihan model. Dengan menambah jumlah *epochs*, terdapat kemungkinan model tersebut memicu *early stopping*, yaitu sebuah algoritma yang digunakan untuk mencari jumlah *epochs* paling optimal untuk mencegah terjadinya *overfitting* pada model [10]. Oleh karena itu, pengujian ini bertujuan untuk mengetahui apakah dengan terpicunya *early stopping*, model yang terbentuk memiliki akurasi yang lebih baik dan tidak mengalami *overfitting*. Pengujian akan dilakukan dengan model YOLOv7 menggunakan ukuran citra 800 yang akan dilatih dengan parameter yang sama seperti pengujian lain terkecuali pada jumlah *epochs* yang ditingkatkan menjadi 1000 dengan *patience* yang divariasikan sebagai parameter untuk jumlah *epochs* yang dibutuhkan untuk menghentikan pelatihan model ketika tidak terjadi peningkatan. *Batch size* yang digunakan juga menyesuaikan dengan *batch size* yang digunakan model pada pengujian sebelumnya, yaitu 4. Berikut adalah hasil yang ditemukan:

Tabel 4.7 Hasil Pengujian Model dengan Peningkatan *Epochs*

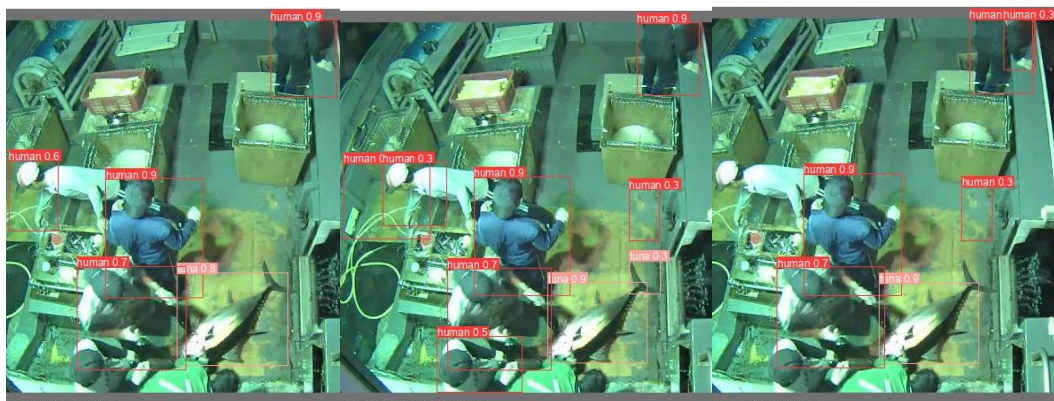
No	Jumlah <i>Epochs</i>	<i>Patience</i>	mAP@0.5	mAP@0.5: 0.95	<i>F1-Score</i>	Jumlah FPS	<i>Inference Time</i> (ms)	<i>Epochs</i> Terbaik
1	300	100	0.843	0.557	0.83	69	14.5	-
2	1000	50	0.861	0.573	0.86	67.6	14.8	644/1000
3	1000	100	0.859	0.569	0.86	63.7	15.7	641/1000

Berdasarkan hasil pengujian pada tabel 4.7, terlihat bahwasanya peningkatan jumlah *epochs* memiliki pengaruh yang kurang lebih mirip dengan

peningkatan ukuran citra dimana akurasi model akan menjadi lebih baik dan kecepatan pendeteksian akan lebih rendah. Akurasi dari model tidak mengalami peningkatan yang signifikan dimana hanya terjadi peningkatan sekitar 3% dibandingkan model yang dilatih dengan 300 *epochs*. Hal tersebut bisa saja terjadi dikarenakan model yang terbentuk sudah bisa mengidentifikasi kelas yang ingin dideteksi sehingga peningkatan akurasi hanya akan menambah node pada *neural networks* dan menambah kemungkinan untuk teridentifikasi sebagai kelas tertentu. Peningkatan akurasi tersebut juga diikuti dengan bertambahnya pendeteksian yang termasuk dalam *false positive* dan *false negative*.



Gambar 4.12 Contoh Citra yang Digunakan dalam Pengujian



Gambar 4.13 Hasil Pengujian pada Model YOLOv7 dengan *Epochs* 300 (Kiri), *Epochs* 1000 dengan *Patience* 50 (Tengah), *Epochs* 1000 dengan *Patience* 100 (Kanan)

Sebagai contoh, pada Gambar 4.12 – 4.13, jika dianalisis, model YOLOv7 dengan *epochs* 300 memiliki 2 kesalahan pendeteksian yang termasuk dalam *false negative*.

Kemudian, model YOLOv7 dengan *epochs* 1000 dan *patience* 50 memiliki 4 kesalahan pendeteksian yang terbagi menjadi 3 *false positive* dan 1 *false negative*. Lalu, model YOLOv7 dengan *epochs* 1000 dan *patience* 100 memiliki 5 kesalahan pendeteksian yang terbagi menjadi 3 *false negative* dan 2 *false positive*. Dari ketiga hasil tersebut, terlihat bahwasanya model dengan *epochs* yang lebih besar rentan untuk melakukan kesalahan. Hal tersebut bisa saja terjadi dikarenakan model tersebut sudah hampir memasuki fase *overfitting* yang membuat model tidak tergeneralisir dalam melakukan pendeteksian.

Namun, jika ditelaah lebih lanjut, ketiga model tersebut berhasil mendeteksi kelas ‘tuna’ dengan tepat meskipun terdapat *false positive* pada model YOLOv7 dengan *epochs* sebesar 1000 dan *patience* sebesar 50. Oleh karena itu, ketiga model ini masih dapat mengidentifikasi kelas ‘tuna’ dengan baik. Untuk mencari tahu bagaimana akurasi model dalam melakukan pendeteksian pada kelas yang termasuk sebagai ikan, perlu membandingkan bagaimana *F1-Score* dari tiap model ketika melakukan pendeteksian pada kelas yang termasuk dalam ikan sebagai tujuan utama model dibentuk.

Tabel 4.8 Perbandingan *F1-Score* Model YOLOv7 dalam Mendeteksi Kelas yang Termasuk Ikan

No	Kelas	Metrics	YOLOv7 <i>epochs</i> =300	YOLOv7 <i>epochs</i> =1000, <i>patience</i> =50	YOLOv7 <i>epochs</i> =1000, <i>patience</i> =100
1	Tuna	Precision	0.887	0.903	0.904
		Recall	0.893	0.915	0.908
		<i>F1-Score</i>	0.890	0.909	0.906
2	Skipjack Tuna	Precision	0.772	0.785	0.768
		Recall	0.646	0.738	0.738
		<i>F1-Score</i>	0.703	0.761	0.753
3	Tongkol	Precision	0.935	0.951	0.956
		Recall	0.936	0.936	0.936
		<i>F1-Score</i>	0.935	0.943	0.946
4	Squid	Precision	0.848	0.825	0.830
		Recall	0.529	0.661	0.628
		<i>F1-Score</i>	0.652	0.734	0.715
5	Unknown	Precision	0.799	0.779	0.788
		Recall	0.751	0.777	0.771
		<i>F1-Score</i>	0.774	0.778	0.779

Dari hasil pada tabel 4.8, terlihat bahwa model dengan *epochs* yang lebih besar memiliki nilai *F1-Score* yang lebih tinggi dalam mendeteksi kelas yang termasuk sebagai ikan. Hal tersebut disebabkan oleh nilai *recall* dari model dengan *epochs* yang lebih besar cenderung lebih besar dibandingkan dengan jumlah *epochs* yang lebih kecil sehingga model tersebut lebih kecil kemungkinannya melakukan pendeteksian yang termasuk sebagai *false negative*. Oleh karena itu, hasil pendeteksian model dengan *epochs* yang lebih tinggi akan lebih baik secara keseluruhan dikarenakan seluruh objek yang terdapat pada citra akan lebih mungkin untuk terdeteksi. Secara keseluruhan, model dengan *epochs* sebesar 1000 dan *patience* sebesar 50 adalah model yang terbaik dibandingkan model lain karena terdapat 3 dari 5 kelas yang memiliki nilai *F1-Score* tertinggi pada model tersebut.

Dari segi kecepatan pendeteksian, penambahan jumlah *epochs* membuat model memiliki kecepatan pendeteksian yang sedikit lebih rendah. Hal tersebut dapat terlihat dimana model dengan *epochs* sebesar 1000 mengalami penurunan yang tidak terlalu signifikan pada nilai *inference time* sekitar 2 -7%. Penurunan nilai *inference time* dapat terjadi dikarenakan kompleksitas dari *neural networks* yang terbentuk pada model dengan makin banyak *node* yang membuat probabilitas dari setiap kelas juga bertambah sehingga dapat menambah waktu yang dibutuhkan untuk mengklasifikasikan objek yang terdeteksi. Model dengan *epochs* sebesar 1000 dan *patience* sebesar 50 sebagai model yang terbaik dari segi akurasi tidak mengalami penurunan yang cukup signifikan saat melakukan pendeteksian. Oleh karena itu, model ini menjadi model yang terbaik dari pengujian dan akan digunakan pada pengujian berikutnya.

4.4.5. Pengujian Pendeteksian Model dengan Jetson Nano

Pengujian terakhir dilakukan dengan menguji model yang terbaik berdasarkan hasil pengujian-pengujian sebelumnya di Jetson Nano. Pengujian ini bertujuan untuk mendapatkan perbandingan hasil pendeteksian antara perangkat yang digunakan untuk pelatihan model dengan perangkat yang dapat digunakan pada dunia nyata. Sebelum melakukan pengujian, Jetson Nano perlu dipersiapkan terlebih dahulu dengan melakukan instalasi pada *library* yang digunakan untuk

mengeksekusi model YOLOv7. Jetson Nano sendiri memiliki RAM berukuran 4 GB dimana ukuran tersebut bisa dikatakan tidak cukup untuk mengeksekusi model YOLOv7 sehingga dibutuhkan RAM tambahan agar model tersebut dapat dieksekusi. Karena Jetson Nano menggunakan sistem operasi ubuntu, hal tersebut dapat dilakukan menggunakan *Swap memory*.

Swap memory adalah fitur pada sistem operasi linux yang digunakan ketika perangkat membutuhkan sumber daya untuk mengeksekusi *process* pada perangkat, tetapi RAM pada perangkat tidak memiliki ruang yang tersisa. Linux akan mengambil ruang yang tidak digunakan pada memori penyimpanan untuk digunakan sebagai RAM tambahan sehingga *process* baru dapat dieksekusi[18]. *Swap memory* dapat dibentuk menggunakan berbagai cara seperti *Swap partition* dan *Swap file*. Karena tidak akan digunakan secara permanen, *Swap memory* akan dibentuk menggunakan *Swap file*. Terdapat beberapa langkah yang diperlukan untuk membuat sebuah *Swap file*. Sebelum membuat *Swap file*, perlu dipastikan bahwa di perangkat tersebut masih belum memiliki ruang khusus untuk *Swap memory*. Berikut adalah perintah yang dapat digunakan untuk mengetahui perangkat belum memiliki ruang khusus untuk *Swap memory*.

```
sudo swapon --show
free -h
```

Gambar 4.14 Perintah untuk Mencari *Swap Memory*[19]

Perintah pertama berfungsi untuk mencari tahu apakah sudah terdapat *Swap memory* yang terpasang pada perangkat, sedangkan perintah kedua berfungsi untuk memastikan tidak ada *Swap memory* yang sedang aktif pada perangkat. Kemudian, setelah memastikan tidak ada *swap memory* yang terpasang ataupun aktif pada perangkat, kita bisa memeriksa apakah terdapat ruang yang dapat digunakan sebagai *swap memory* pada memori penyimpanan. Berikut adalah perintah yang dapat digunakan untuk mengetahui ruang yang dapat digunakan pada memori penyimpanan.

```
df -h
```

Gambar 4.15 Perintah untuk Mencari Ruang Pada Memori Penyimpanan[19]

Setelah memastikan terdapat ruang pada memori penyimpanan yang dapat digunakan sebagai *Swap memory*, kita dapat membuat *Swap file*. Berikut adalah perintah yang dapat digunakan untuk membuat *Swap file* pada ubuntu.

```
sudo fallocate -l size directory  
ls -lh directory
```

Gambar 4.16 Perintah untuk Membuat *Swap File*[19]

Perintah pertama berfungsi untuk membentuk *swap file* pada perangkat. Kita perlu memasukkan ukuran dari *swap file* yang akan digunakan dan direktori *swap file* tersimpan pada perangkat. Kemudian, perintah kedua berfungsi untuk memastikan apakah *swap file* memiliki ukuran yang sesuai dengan perintah pertama. Kemudian, *swap file* perlu kita aktifkan sebagai sebuah *swap space* agar dapat digunakan sebagai *swap memory*. Berikut adalah perintah untuk mengaktifkan *swap file* menjadi sebuah *swap memory*.

```
sudo chmod 600 directory  
ls -lh directory  
sudo mkswap directory  
sudo swapon directory
```

Gambar 4.17 Perintah untuk Mengaktifkan *Swap File* menjadi *Swap Memory*

Perintah pertama digunakan untuk mengubah *permission* dari direktori *swap file* yang dibuat agar hanya bisa diakses user dengan *permission* administrator atau *root* pada perangkat sehingga *swap file* tersebut tidak dapat dieksploitasi oleh user tanpa *permission* administrator. Perintah kedua digunakan untuk memastikan apakah direktori sudah dalam *permission* administrator. Kemudian, perintah ketiga berfungsi untuk membuat direktori *swap file* menjadi *swap space* sehingga dapat digunakan sebagai *swap memory*. Lalu, perintah keempat berfungsi untuk mengaktifkan *swap file* menjadi *swap memory* pada perangkat.

Pengujian akan dilakukan menggunakan sebuah video yang tidak termasuk dalam *dataset* pengujian dengan *frame rate* 24 FPS. Video tersebut terdiri dari beberapa objek yang termasuk dalam kelas pada pelatihan model. Model akan dimuat pada Jetson Nano dan akan melakukan inferensi pada video tersebut untuk mendeteksi objek yang terdapat pada video dengan parameter *IoU threshold*

sebesar 0.45 dan *confidence threshold* sebesar 0.25. Kemudian, video tersebut akan diekstrak tiap *frame*-nya untuk nantinya beberapa *frame* akan digunakan sebagai *dataset* baru dan digunakan untuk menguji apakah akurasi dari model pada Jetson Nano dan perangkat pelatihan model memiliki hasil yang sama. Pengujian akurasi model pada Jetson Nano menggunakan parameter yang sama dengan parameter pada pengujian-pengujian sebelumnya.

Tabel 4.9 Hasil Pengujian Model Pada Jetson Nano dan Perangkat Pelatihan

No	Berkas Video	Perangkat	Jumlah FPS	<i>Inference Time</i> (ms)	Ukuran Video
1	hiv00074. mp4	Jetson Nano	1.3	750.3	384 x 640
2		PC Pelatihan	98.0	10.2	
3	hiv00167. mp4	Jetson Nano	1.3	797.6	
4		PC Pelatihan	98.0	10.2	
5	hiv00171. mp4	Jetson Nano	1.4	697.2	
6		PC Pelatihan	98.0	10.2	

Berdasarkan hasil pengujian pada Tabel 4.9, dapat terlihat bahwasanya nilai *inference time* model dalam mendeteksi pada Jetson Nano mengalami penurunan yang sangat signifikan dibandingkan dengan *inference time* pada PC pelatihan. Kecepatan pendeteksian pada Jetson Nano dapat mengalami penurunan hingga 700% dibandingkan kecepatan pendeteksian pada PC pelatihan. Hal tersebut dapat terjadi karena beberapa faktor, salah satunya adalah perbedaan proses komputasi yang dapat dilakukan oleh Jetson Nano dan PC pelatihan. Jetson Nano memiliki spesifikasi yang jauh lebih rendah dibandingkan dengan PC pelatihan seperti CPU, GPU dan RAM. Menurunnya spesifikasi perangkat membuat proses komputasi yang dapat dilakukan Jetson Nano tidak bisa menyamai proses komputasi yang dapat dilakukan oleh PC pelatihan. Selain itu, varian model YOLOv7 memiliki arsitektur yang cukup kompleks jika dibandingkan dengan varian YOLOv7 yang lebih rendah seperti YOLOv7-Tiny sehingga *trainable paramters* model memiliki jumlah yang lebih banyak. Oleh karena itu, dengan spesifikasi yang tertera pada

Tabel 4.1, Jetson Nano tidak memiliki spesifikasi yang dapat memroses seluruh parameter dengan cepat sehingga kecepatan pendeteksian pada Jetson Nano akan menurun. Akan tetapi, jika ditelaah lebih lanjut pada video hasil *inference*, tidak ada *frame* video yang hilang meskipun kecepatan pendeteksian mengalami penurunan.



Gambar 4.18 Contoh Salah Satu *Frame* Hasil *Inference* pada Video Pengujian

Sebagai contoh, Gambar 4.17 merupakan salah satu *frame* dari berkas video hiv00167.mp4. Dari gambar tersebut, terlihat bahwa hasil *inference* model pada video masih menampilkan *frame* dengan objek deteksi yang sama meskipun kecepatan pendeteksian mengalami penurunan. Hal tersebut dapat terjadi dikarenakan *inference* akan dilakukan pada semua *frame* yang terdapat pada video sehingga hasil akhirnya akan memiliki durasi dan *frame rate* yang sama meskipun kecepatan pendeteksian mengalami penurunan.

Tabel 4.10 Hasil Pengujian Akurasi Model Pada Jetson Nano dan Perangkat Pelatihan

No	Perangkat	Ukuran Citra	mAP@0.5	mAP@0.5:0.95	F1-Score
1	Jetson Nano	640	0.669	0.47	0.68
2	PC Pelatihan		0.669	0.47	0.68

Kemudian, berdasarkan akurasi pendeteksian, model yang dipasang pada Jetson Nano dan PC pelatihan tidak mengalami perbedaan. Dari hasil pada Tabel

4.10, dapat terlihat bahwa model tersebut memiliki nilai mAP dan *F1-Score* yang sama meskipun menggunakan perangkat dengan spesifikasi yang sangat berbeda.



Gambar 4.19 Contoh Citra Hasil Ekstraksi *Frame* Video Pengujian



Gambar 4.20 Hasil Pengujian Akurasi Pendeteksian pada Jetson Nano (Kiri) dan PC Pelatihan (Kanan)

Sebagai contoh, pada Gambar 4.18, terlihat bahwa citra memiliki dua label, yaitu kelas ‘tuna’ dan ‘human’. Pada Gambar 4.19, hasil pengujian model memiliki hasil yang sama dimana model pada Jetson Nano dan PC pelatihan berhasil mendeteksi kedua objek yang seharusnya terdeteksi dengan label yang tepat ditambah dengan dua kesalahan *false positive* pada kelas ‘unknown’ dan ‘human’. Hal tersebut dapat terjadi karena model yang digunakan memiliki jumlah *trainable parameters* yang

sama sehingga model tersebut akan memiliki kapabilitas pendeteksian yang sama meskipun dipasang pada perangkat dengan spesifikasi yang sangat berbeda. Oleh karena itu, dapat disimpulkan bahwa penggunaan perangkat Jetson Nano hanya akan menurunkan kecepatan pendeteksian tanpa mengurangi akurasi pendeteksian.

4.5. Dampak Terhadap Lingkungan, Masyarakat dan/atau Bidang Lain

Jika ditinjau dari aspek lingkungan, masyarakat atau bidang lain, pengembangan sistem deteksi multi objek untuk pendeteksian ikan berbasis YOLOv7 memberikan beberapa dampak pada beberapa aspek. Dampak utama dari pengembangan sistem deteksi ini adalah dapat mencegah terjadinya tindak *illegal, unreported and unregulated (IUU) fishing* sesuai dengan latar belakang pengembangan sistem. Dengan sistem ini, hasil tangkapan dari nelayan dapat dipantau sehingga jumlah ikan dapat diketahui dan diverifikasi ketika tiba di pelabuhan. Jika jumlah ikan yang dilaporkan tidak sesuai dengan hasil pemantauan, dapat terindikasi bahwa nelayan melakukan tindak IUU *fishing* dengan memeriksa kembali hasil pemantauan.

Kemudian, dengan berkurangnya tindak IUU *fishing*, pendapatan devisa negara Indonesia dapat bertambah dari sektor perikanan. Hal tersebut berkaitan dengan bertambahnya jumlah ikan yang dapat diekspor ke berbagai negara jika penjualan tidak dilakukan secara ilegal sehingga sektor perikanan dapat dikembangkan oleh pemerintah Indonesia. Selain itu, ekosistem laut Indonesia dapat terjaga dari kerusakan karena IUU *fishing*. Tindak IUU *fishing* sangat umum dilakukan dengan menggunakan peralatan yang tidak diperbolehkan untuk melakukan penangkapan ikan seperti jaring pukat harimau atau bom ikan agar mendapatkan banyak hasil tangkapan. Oleh karena itu, dengan berkurangnya penggunaan peralatan yang tidak diperbolehkan, ekosistem kelautan Indonesia akan tetap terjaga dari kerusakan atau kepunahan spesies.

Namun, pengembangan sistem pendeteksian ikan ini masih belum sempurna. Terdapat beberapa kelemahan dari hasil pengembangan sistem yang telah dilakukan seperti model yang cukup berat untuk dieksekusi pada perangkat *edge devices* seperti mini-PC atau mikrokontroler karena membutuhkan proses

komputasi yang cukup besar untuk mendapatkan kecepatan pendeteksian yang cepat. Selain itu, *dataset* yang digunakan juga masih terbatas pada 6 kelas sehingga dibutuhkan kelas tambahan untuk bisa mengidentifikasi objek yang mungkin terdeteksi. Rasio data antar kelas juga masih belum seimbang sehingga model lebih cenderung berhasil melakukan pendeteksian secara tepat pada kelas-kelas tertentu.

BAB 5

KESIMPULAN

5.1. Kesimpulan

Berdasarkan penelitian yang telah dilakukan, terdapat beberapa hal yang dapat disimpulkan terkait pengembangan sistem pendeteksian ikan secara multi objek menggunakan algoritma deteksi objek YOLOv7, yaitu:

1. Pengembangan sistem deteksi multi objek untuk pendeteksian ikan dengan model YOLOv7 berhasil dilakukan dengan *dataset* yang dirancang secara personal dengan jumlah gambar sebanyak 4280 mencakup 22.870 label dan 6 kelas.
2. Bentuk data *polygon annotation* tidak memiliki peningkatan yang signifikan pada akurasi pendeteksian, tetapi dapat meningkatkan kecepatan pendeteksian model YOLOv7 dibandingkan bentuk data *bounding box*.
3. Metode *object detection* memiliki hasil akurasi dan kecepatan pendeteksian yang lebih baik dibandingkan metode *instance segmentation* dalam sistem pendeteksian ikan secara multi objek dengan model YOLOv7.
4. Menggunakan ukuran citra yang lebih besar pada pelatihan model dapat meningkatkan akurasi pendeteksian pada model YOLOv7 dengan menurunkan kecepatan pendeteksian.
5. Meningkatkan jumlah *epochs* yang digunakan tidak meningkatkan akurasi pendeteksian secara signifikan, tetapi mengurangi jumlah pendeteksian *false negative* dari model YOLOv7.
6. Varian YOLOv7 menjadi varian model YOLOv7 terbaik yang dapat digunakan untuk sistem pendeteksian ikan secara multi objek dengan melakukan perubahan pada parameter saat pelatihan model.
7. Pengimplementasian sistem pendeteksian ikan secara multi objek berhasil dilakukan pada Jetson Nano dengan kecepatan pendeteksian yang lebih lambat dibandingkan dengan perangkat pelatihan tanpa mengurangi akurasi pendeteksian.

5.2. Saran

Setelah menyelesaikan seluruh rangkaian penelitian, terdapat beberapa saran yang dapat diberikan untuk penelitian lebih lanjut, yakni:

1. Menambah jumlah kelas dari *dataset* yang digunakan sehingga lebih banyak jenis objek yang dapat terdeteksi oleh model dan menyeimbangkan rasio dari data untuk setiap kelas pada *dataset* agar pendeteksian tidak cenderung baik hanya pada kelas tertentu.
2. Menggunakan varian model YOLOv7 yang lebih rendah seperti YOLOv7-Tiny jika akan dikembangkan pada sebuah Jetson nano atau menggunakan perangkat *edge devices* yang lebih kencang dibandingkan Jetson Nano.
3. Mengembangkan model pendeteksian menggunakan metode *instance segmentation* agar perhitungan jumlah ikan yang terdeteksi bisa lebih presisi dan menambah penggunaan dari model pendeteksian.

DAFTAR PUSTAKA

- [1] “Fish; fillets, frozen exports by country |2021.”
<https://wits.worldbank.org/trade/comtrade/en/country/ALL/year/2021/trade-flow/Exports/partner/WLD/product/030420> (accessed Nov. 20, 2022).
- [2] “Tuna export company and exporters in Indonesia - Tridge.”
<https://www.tridge.com/intelligences/atlantic-bluefin-tuna/ID/export>
 (accessed Nov. 21, 2022).
- [3] “KKP Tangkap 167 Kapal Pelaku Illegal Fishing Selama 2021 Halaman all
 - Kompas.com.”
<https://money.kompas.com/read/2021/12/13/173905726/kkp-tangkap-167-kapal-pelaku-illegal-fishing-selama-2021?page=all> (accessed Nov. 21, 2022).
- [4] “IUU Fishing as an Evolving Threat to Southeast Asia’s Maritime Security | Asia Maritime Transparency Initiative.” <https://amti.csis.org/iuu-fishing-as-an-evolving-threat-to-southeast-asias-maritime-security/> (accessed Nov. 21, 2022).
- [5] “Indonesia sinks 8 Malaysian vessels for illegal fishing.”
<https://www.aa.com.tr/en/asia-pacific/indonesia-sinks-8-malaysian-vessels-for-illegal-fishing/2181821> (accessed Nov. 21, 2022).
- [6] “What is Computer Vision? | IBM.” <https://www.ibm.com/id-en/topics/computer-vision> (accessed Dec. 05, 2022).
- [7] Z. Zou, Z. Shi, Y. Guo, J. Ye, and S. Member, “Object Detection in 20 Years: A Survey,” May 2019, doi: 10.48550/arxiv.1905.05055.
- [8] V. Lakshmanan, M. Görner, and R. Gillard, *Practical Machine Learning for Computer Vision End-to-End Machine Learning for Images*.
- [9] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*.

- [10] M. Elgendy, O'Reilly for Higher Education (Firm), and an O. M. Company. Safari, *Deep Learning for Vision Systems*.
- [11] R. Szeliski, *Computer Vision: Algorithms and Applications 2nd Edition*. 2021. [Online]. Available: <https://szeliski.org/Book>,
- [12] "COCO - Common Objects in Context." <https://cocodataset.org/#detection-eval> (accessed May 25, 2023).
- [13] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors".
- [14] "Getting Started with YOLO v4 - MATLAB & Simulink." <https://www.mathworks.com/help/vision/ug/getting-started-with-yolo-v4.html> (accessed May 25, 2023).
- [15] "Description." <https://www.fishnet.ai/description> (accessed Dec. 16, 2022).
- [16] "Polygon Annotation | Humans in the Loop." <https://humansintheloop.org/services/polygon-annotation/> (accessed May 18, 2023).
- [17] "Label Studio." <https://fishmon-tagging.aiseeyou.tech/projects/?page=1> (accessed Dec. 17, 2022).
- [18] "Chapter 15. Swap Space Red Hat Enterprise Linux 7 | Red Hat Customer Portal." https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/storage_administration_guide/ch-swapspace (accessed May 22, 2023).
- [19] "How To Add Swap Space on Ubuntu 18.04 | DigitalOcean." <https://www.digitalocean.com/community/tutorials/how-to-add-swap-space-on-ubuntu-18-04> (accessed May 22, 2023).