

**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**

G:\Gilang\Downloads\logo_ui_hitam.wmf

**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 Asshiddiqqi, 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



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

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universtas 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.

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

# 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.

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

# DAFTAR ISI

[HALAMAN PERNYATAAN ORISINALITAS ii](#_Toc134798797)

[LEMBAR PENGESAHAN iii](#_Toc134798798)

[KATA PENGANTAR iv](#_Toc134798799)

[HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI KARYA ILMIAH UNTUK KEPENTINGAN AKADEMIS vi](#_Toc134798800)

[ABSTRAK vii](#_Toc134798801)

[ABSTRACT viii](#_Toc134798802)

[DAFTAR ISI ix](#_Toc134798803)

[DAFTAR GAMBAR xi](#_Toc134798804)

[DAFTAR TABEL xii](#_Toc134798805)

[BAB 1 PENDAHULUAN 1](#_Toc134798806)

[1.1. Latar Belakang 1](#_Toc134798807)

[1.2. Rumusan Masalah 2](#_Toc134798808)

[1.3. Tujuan Penelitian 3](#_Toc134798809)

[1.4. Batasan Masalah 3](#_Toc134798810)

[1.5. Metodologi Penelitian 3](#_Toc134798811)

[1.6. Sistematika Penulisan 5](#_Toc134798812)

[BAB 2 SISTEM DETEKSI MULTI OBJEK BERBASIS YOLOV7 6](#_Toc134798813)

[2.1. Machine Learning 6](#_Toc134798814)

[2.1.1. Deep Learning 7](#_Toc134798815)

[2.2. Computer Vision 11](#_Toc134798816)

[2.2.1. Object Detection 13](#_Toc134798817)

[2.3. Convolutional Neural Networks 19](#_Toc134798818)

[2.4. YOLO 21](#_Toc134798819)

[2.4.1. YOLOv7 24](#_Toc134798820)

[2.5. Fishnet 25](#_Toc134798821)

[2.6. Penelitian Terkait 26](#_Toc134798822)

[BAB 3 PERANCANGAN SISTEM DETEKSI MULTI OBJEK BERBASIS YOLOV7 28](#_Toc134798823)

[3.1. *System requirements* 28](#_Toc134798824)

[3.2. Pengumpulan Data 29](#_Toc134798825)

[3.3. *Preprocessing* Data 29](#_Toc134798826)

[3.4. Rancangan Proses Pelatihan Model 30](#_Toc134798827)

[3.5. Rancangan Proses Pengujian Model 32](#_Toc134798828)

[BAB 4 IMPLEMENTASI DAN ANALISIS SISTEM DETEKSI MULTI OBJEK BERBASIS YOLOV7 34](#_Toc134798829)

[4.1. Implementasi Sistem 34](#_Toc134798830)

[4.2. Skenario Pengujian 34](#_Toc134798831)

[4.3. Pengujian Model 35](#_Toc134798832)

[4.3.1. Pengujian Pengaruh *Dataset* Berbentuk *Polynomial Segmentation* Terhadap Akurasi dan Jumlah FPS 36](#_Toc134798833)

[4.3.2. Pengujian Perbandingan Metode Object Detection dengan Instance Segmentation Terhadap Akurasi 36](#_Toc134798834)

[4.3.3. Pengujian Pengaruh Ukuran Gambar Terhadap Akurasi dan Jumlah FPS 36](#_Toc134798835)

[4.3.4. Pengujian Pengaruh Optimizers Terhadap Akurasi dan Jumlah FPS 36](#_Toc134798836)

[4.3.5. Pengujian Pengaruh Penambahan Epochs dalam Memicu *Early Stopping* Terhadap Akurasi dan Jumlah FPS 36](#_Toc134798837)

[4.3.6. Pengujian Pendeteksian Model dengan Jetson Nano 36](#_Toc134798838)

[4.4. Dampak Terhadap Lingkungan dan Masyarakat 36](#_Toc134798839)

[BAB 5 KESIMPULAN 37](#_Toc134798840)

[5.1. Kesimpulan 37](#_Toc134798841)

[5.2. Saran 37](#_Toc134798842)

[DAFTAR PUSTAKA 38](#_Toc134798843)

# DAFTAR GAMBAR

Gambar 2.1 Perbandingan alur machine learning tradisional dan *deep learning* [10] 7

Gambar 2.2 Contoh *backpropagation* pada 4-layer *neural networks* 11

Gambar 2.3 Contoh *Region of Interest* pada citra sebuah anjing 14

Gambar 2.4 Hasil prediksi sebelum dan sesudah diterapkan *non-maximum suppression* 16

Gambar 2.5 Contoh *Precision-Recall Curve* dengan IoU *Threshold* = 0.5 17

Gambar 2.6 Proses ekstraksi fitur pada *convolutional layers* 19

Gambar 2.7 Contoh *max pooling feature map* berukuran 4 X 4 dengan *window* berukuran 2 x 2 20

Gambar 2.8 Contoh *global average pooling* *feature map* berukuran 4 X 4 20

Gambar 2.9 Arsitektur *fully connected layers* 21

Gambar 2.10 Ilustrasi pendeteksian menggunakan algoritma YOLO 23

Gambar 2.11 Arsitektur awal model YOLO 24

Gambar 3.1 Rancangan sistem pendeteksian ikan 28

Gambar 3.2 Tahapan *pre-processing* data 29

Gambar 3.3 Proses *labelling* dengan Label Studio [15] 30

Gambar 3.4 Skema proses pelatihan model YOLOv7 32

# DAFTAR TABEL

Tabel 2.1 Persebaran bagian *dataset* Fishnet 25

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]. 26

Tabel 3.1 Spesifikasi Komputer 28

# PENDAHULUAN

## 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.

## 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?

## 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.

## 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.

## 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.

1. 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.

1. 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.

1. Desain Sistem

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

1. Rancangan proses pelatihan model

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

1. 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.

1. Analisis

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

1. 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.

## 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.

# SISTEM DETEKSI MULTI OBJEK BERBASIS YOLOV7

## Machine Learning

*Machine learning* merupakan salah satu sub-bidang dari *Artificial Intelligence* (AI) yang bertujuan untuk membuat komputer bisa mereplika 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].

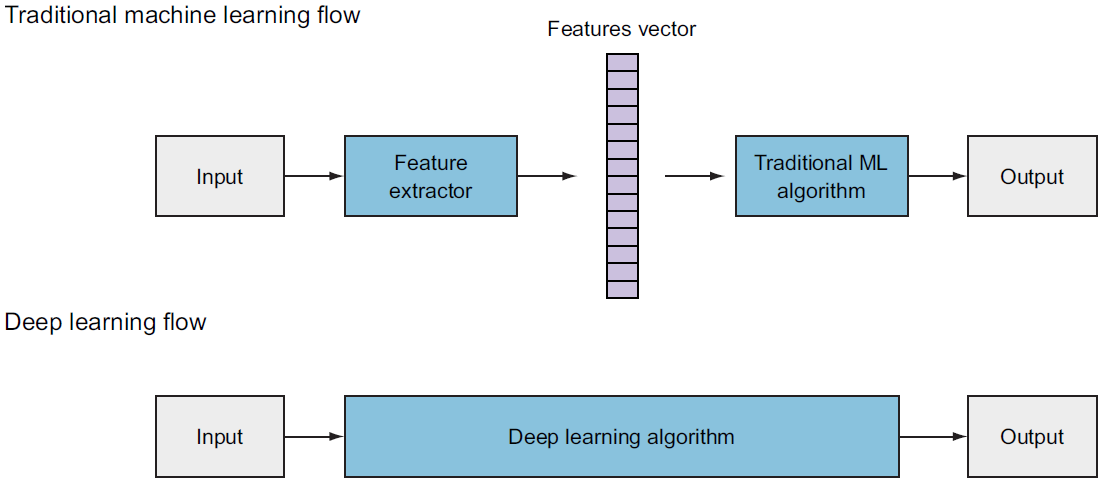
*Machine learning* memungkinkan kita untuk menyelesaikan *tasks* yang terlalu sulit unuk diselesaikan hanya dengan suatu program yang dibentuk oleh seorang manusia dan bersifat tetap. *Tasks* didefinisikan sebagai bagaimana suatu sistem *machine learning* harus mengolah sebuah *example* yang didefinisikan 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 pixel-pixel 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.

### 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:

1. *Input vector,* yaitu vektor dari *feature* yang berhasil diekstrak dari masukan. Umumnya, disimbolkan dengan X sebagai sebuah vektor dari masukan ()
2. *Weights vector,* yaitu representasi seberapa pentingnya *example* tersebut dibandingkan dengan *example* lain yang ada pada *dataset*. Direpresentasikan dengan sebuah vektor dari *weights* ()
3. *Neuron functions*, yaitu kalkulasi dari sinyal masukan berupa penjumlahan dari *weighted sum* dan *step activation function*.
4. *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:

1. *Input layer,* yaitu layer yang terdiri dari *features* yang dimiliki oleh masukan data
2. *Hidden layers,* yaitu layer yang terdiri darineuron-neuron yang akan mengekstrak vektor dari *feature* yang akan diproses pada *input layer*.
3. *Weight connections (edges),* yaitu nilai yang terdapat pada tiap hubungan antar *nodes* pada *neural networks* untuk menggambarkan pentingnya *nodes* tersebut pada hasil prediksi.
4. *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:

|  |  |
| --- | --- |
|  | (2.1) |

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

|  |  |
| --- | --- |
|  | (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:

|  |  |
| --- | --- |
|  | (2.3) |

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

|  |  |
| --- | --- |
|  | (2.4) |

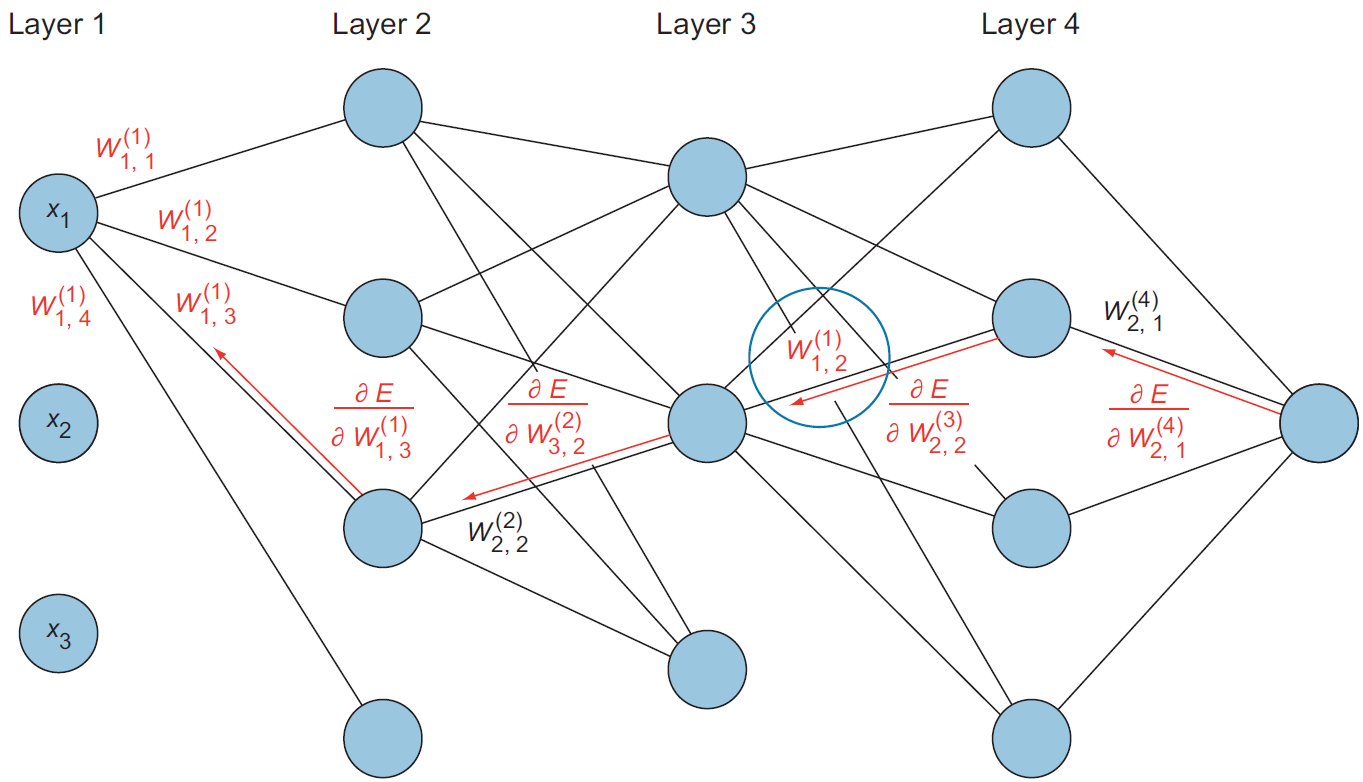
Tahap ketiga, menerapkan *optimization functions* seperti *gradient descent* atau jenis lainnya untuk memodifikasi *neural networks* secara berulang hingga di titik paling optimalnya yang dilakukan dengan mengkomputasi turunan *loss function* untuk example terhadap *weights* menggunakan *chain rule* dari *output layer* Kembali ke *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

|  |  |
| --- | --- |
|  | (2.5) |

Untuk mengomputasi turunan dari *loss* terhadap *weights*, kita dapat menggunakan *chain rule* sehingga menghasilkan persamaan

|  |  |
| --- | --- |
|  | (2.6) |

Ketika *backpropagation* diterapkan pada suatu *multilayer perceptron*, terdapat banyak *weights* yang perlu dikomputasi. Oleh karena itu, digunakan *chain rule* untuk mencari nilai turunan dari error sesuai dengan node yang akan mengarah 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

|  |  |
| --- | --- |
|  | (2.7) |

## 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].

### Object Detection

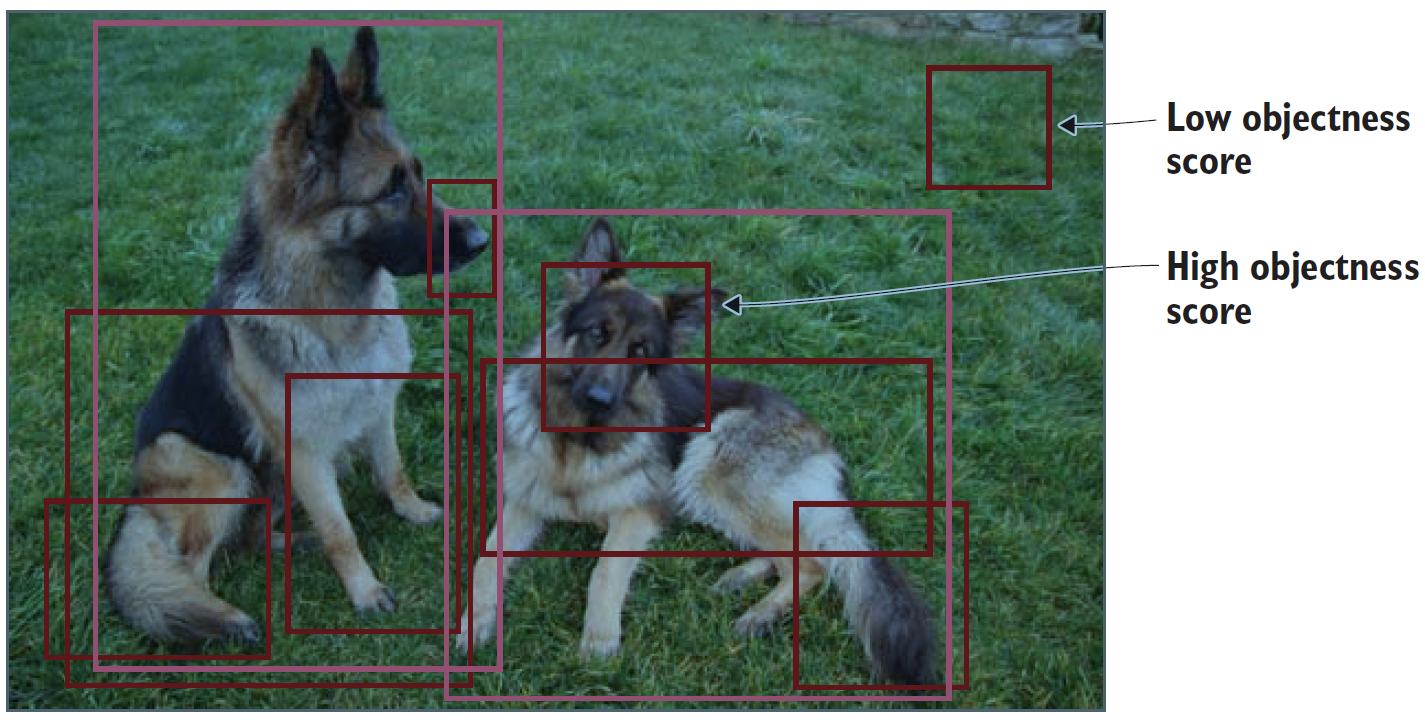
*Object detection* merupakan salah satu contoh pengaplikasian teknologi *computer vision* di kehidupan nyata. *Object detection* merupakan pengembangan dari *image classification* di mana *object detection* berfungsi untuk membuat *interpreting device* yang diterapkan pada sistem *computer vision* bisa 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:

1. *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 perkirakan 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

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 lambah karena membutuhkan proses komputasional yang kompleks. Oleh karena iu, penetapan parameter perlu disesuaikan dengan permasalahan yang ingin diselesaikan untuk mengurangi jumlah RoIs yang terbentuk.

1. *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.

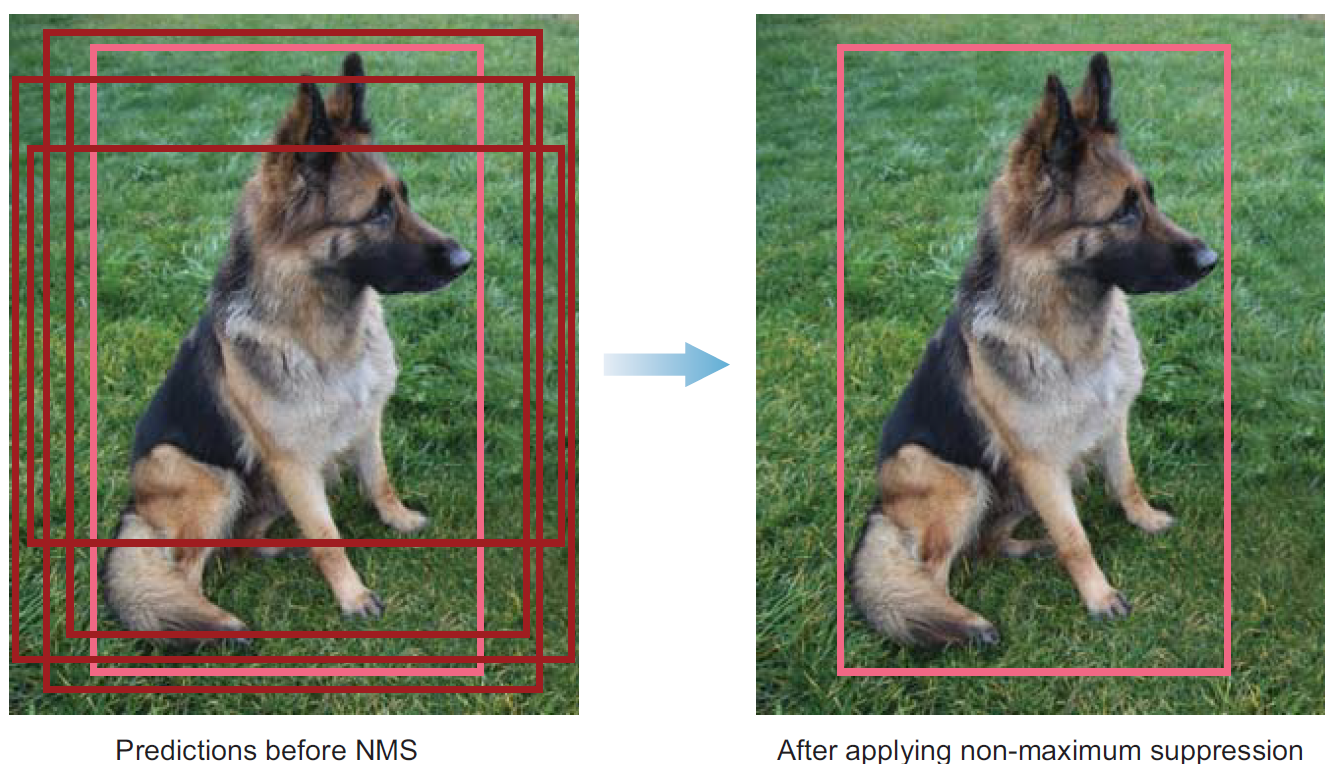
Sebuah objek yang dikenali pada suatu citra dapat memiliki beberapa *foreground region* yang secara tepat melakukan klasifikasi, tetapi tidak semua *region* akan digunakan untuk melakukan deteksi. Di beberapa permasalahan, hanya dibutuhkan satu *foreground region* saja dengan tingkat prediksi tertinggi 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.

1. *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*

1. *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.

* *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

|  |  |
| --- | --- |
|  | (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*).

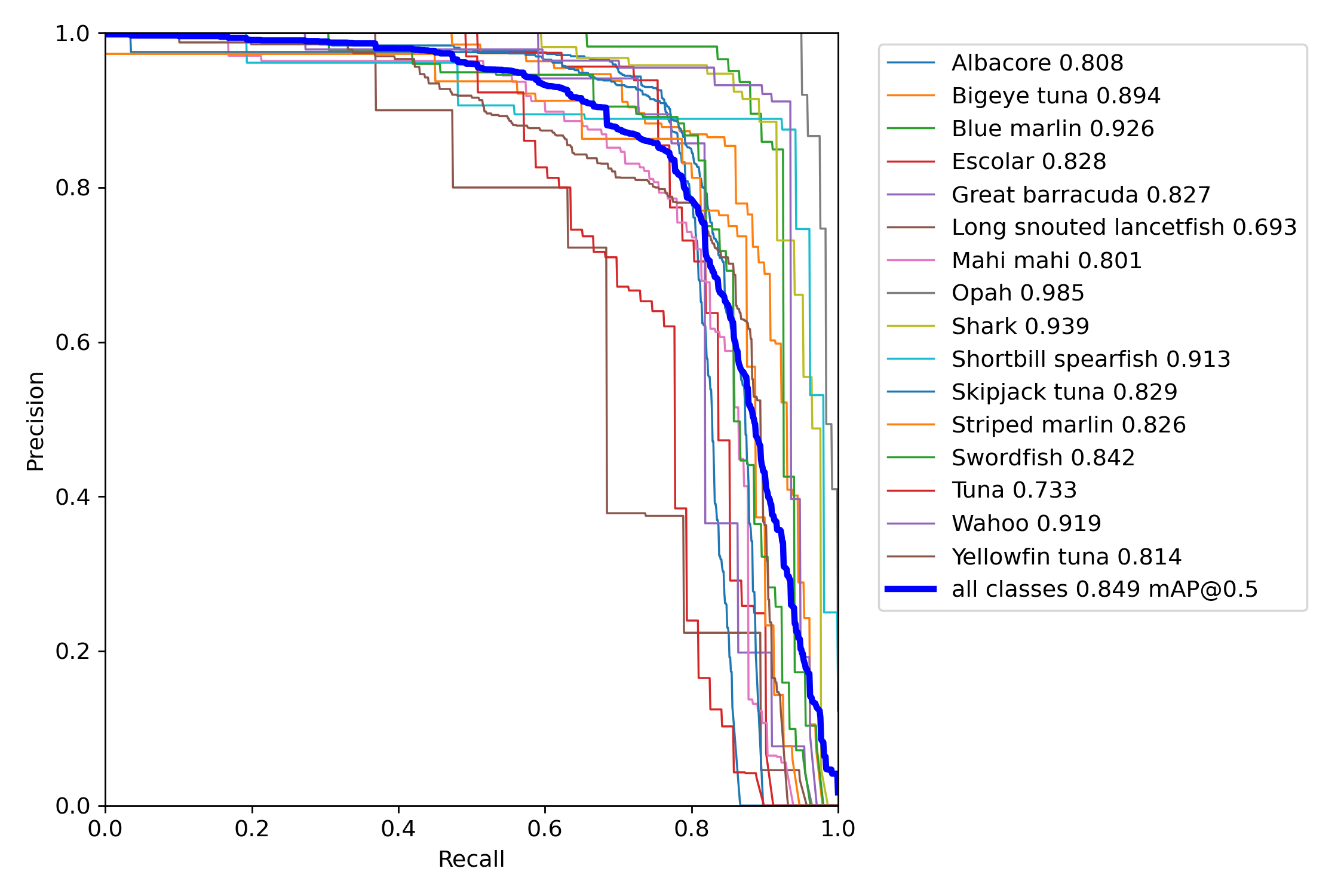
* *Precision-recall Curve* (*PR-Curve*)

*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

|  |  |
| --- | --- |
|  | (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

|  |  |
| --- | --- |
|  | (2.10) |



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

Dari kurva tersebut, dapat dicari tahu apakah sebuah model *object detection* sudah baik atau belum. Jika *precision* selalu bernilai tinggi seiring bertambahnya nilai *recall*, maka model tersebut dapat dikatakan sebagai model *object detection* yang baik. Kita juga dapat menentukan *average precision* (AP) menggunakan nilai dari *interpolated precision* untuk setiap level *recall*. Nilai *interpolated precision* dapat dicari menggunakan persamaan

|  |  |
| --- | --- |
|  | (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

|  |  |
| --- | --- |
|  | (2.12) |

Jika terdapat beberapa kelas yang dilatih, kita perlu mencari rata-rata dari *average precision* untuk tiap kelas yang disebut sebagai *mean Average Precision* (mAP). Nilai *mean Average Precision* (mAP) ditentukan dengan persamaan [8]

|  |  |
| --- | --- |
|  | (2.13) |

* *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

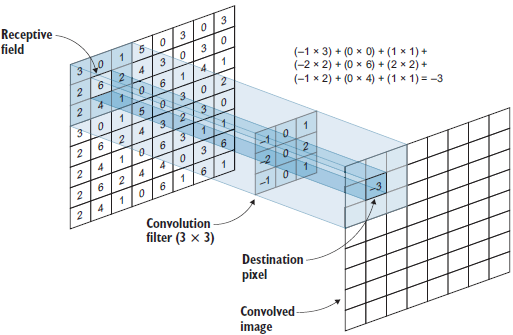
|  |  |
| --- | --- |
|  | (2.14) |

## 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:

1. *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.



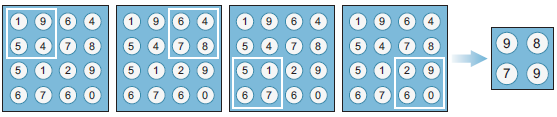
Gambar 2.6 Proses ekstraksi fitur pada *convolutional layers*[10]

1. *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*

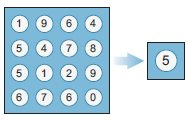
*Max pooling* adalah jenis *pooling layers* yang akan memeriksa *feature map* hasil *convolutional layers* dan mengambil nilai fitur yang tertinggi pada *window* sehingga terbentuk *feature map* baru yang hanya akan berisi nilai fitur hasil *pooling*.



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.

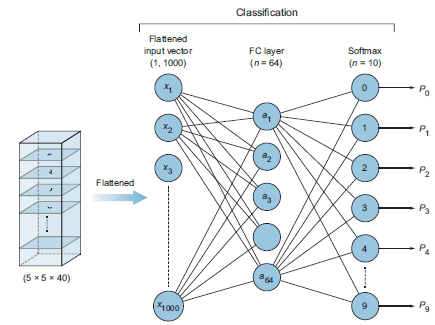


Gambar 2.8 Contoh *global average pooling* *feature map* berukuran 4 X 4[10]

1. *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]

## 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 pertama kali dipublikasikan pada tahun 2016 oleh Joseph Redmon di konferensi IEEE untuk *computer vision* dan *pattern recognition*. Beberapa algoritma sebelum YOLO menggunakan metode untuk mengubah fungsi dari sistem pengklasifikasi menjadi sistem pendeteksian. Contohnya adalah algoritma *deformable parts models* (DPM) yang menggunakan metode *sliding window* untuk membagi sama rata semua bagian pada sebuah citra. Selain itu, ada algoritma R-CNN yang menggunakan metode *region proposal* untuk membentuk beberapa *bounding boxes* yang mungkin dan melakukan prediksi awal yang kemudian akan diterapkan *post-processing* untuk menyaring *bounding boxes,* menghilangkan duplikasi *bounding boxes* dan menghitung *bounding boxes* tersebut yang membutuhkan banyak *resource* karena membutuhkan *training* secara terpisah [12].

YOLO menggunakan pendekatan yang berbeda dalam melakukan pendeteksian di mana YOLO akan membagi citra menjadi beberapa bagian menggunakan sebuah *grid cell* berukuran *S X 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* umunya didefinisikan sebagai

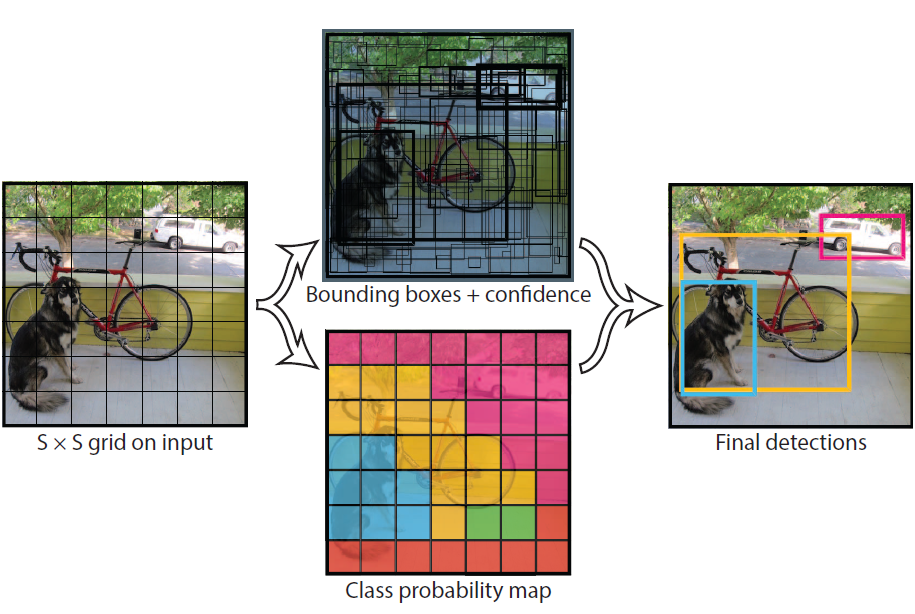
|  |  |
| --- | --- |
|  | (2.15) |

*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.

G*rid 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(*Class*i|*Object*) yang akan dikalikan nilai *confidence* tiap *box* pada saat pengujian sehingga akan dihasilkan nilai *confidence* untuk tiap kelas objek dengan persamaan

|  |  |
| --- | --- |
|  | (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].

### 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*. Tipe model YOLOv7 terbagi menjadi YOLOv7 P5 dan P6 di mana YOLOv7 P5 terdiri dari YOLOv7-Tiny, YOLOv7 dan YOLOv7-X, sedangkan YOLOv7 P6 terdiri dari YOLOv7-W6, YOLOv7-E6, YOLOv7-D6, YOLOv7-E6E. Perbedaan dari kedua tipe ini adalah resolusi masukan citra yang digunakan, *hyperparameter* untuk proses *training* dan penerapannya dimana YOLOv7 P5 berfokus pada GPU untuk *edge computing* dan umum, sedangkan YOLOv7 P6 berfokus pada GPU untuk *cloud computing* [13].

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. Selain itu, YOLOv7 memiliki nilai *average precision* yang sama dengan PPYOLOE-L, tetapi memiliki FPS hasil *inference* yang lebih cepat 83 FPS dengan pengurangan parameter yang akan digunakan sebesar 41%. Kemudian, YOLOv7-X memiliki nilai *average precision* yang lebih baik dibandingkan YOLOv5-X dengan peningkatan sebesar 2.2% dengan pengurangan jumlah parameter sebesar 22%. Hasil *inference* YOLOv7-X juga lebih cepat 31 FPS dibandingkan YOLOv5-X.

## 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 [14].

*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** |
| *train* | 100.673 |
| *validation* | 30.202 |
| *Test* | 12.943 |
| **Total** | 143.818 |

## 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].

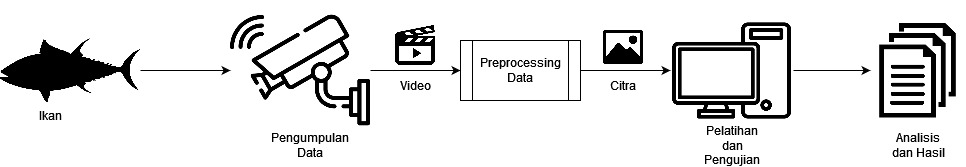
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Objek | Sesi | Metrics | YOLOv3-Tiny | YOLOv4-Tiny | YOLOv4-Tiny-SPP |
| *Head* | *Training* | *Precision* | 88.16 | 91.99 | 92.31 |
| *Recall* | 79.36 | 87.48 | 91.37 |
| *F1-score* | 83.53 | 89.68 | 91.84 |
| *Validation* | *Precision* | 85.19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

# PERANCANGAN SISTEM DETEKSI MULTI OBJEK BERBASIS YOLOV7

## *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.

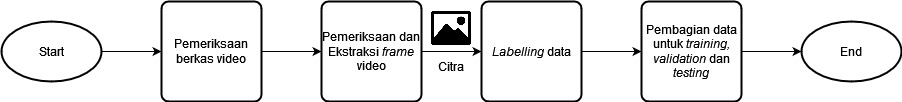
Tabel 3.1 Spesifikasi Komputer

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

## 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.

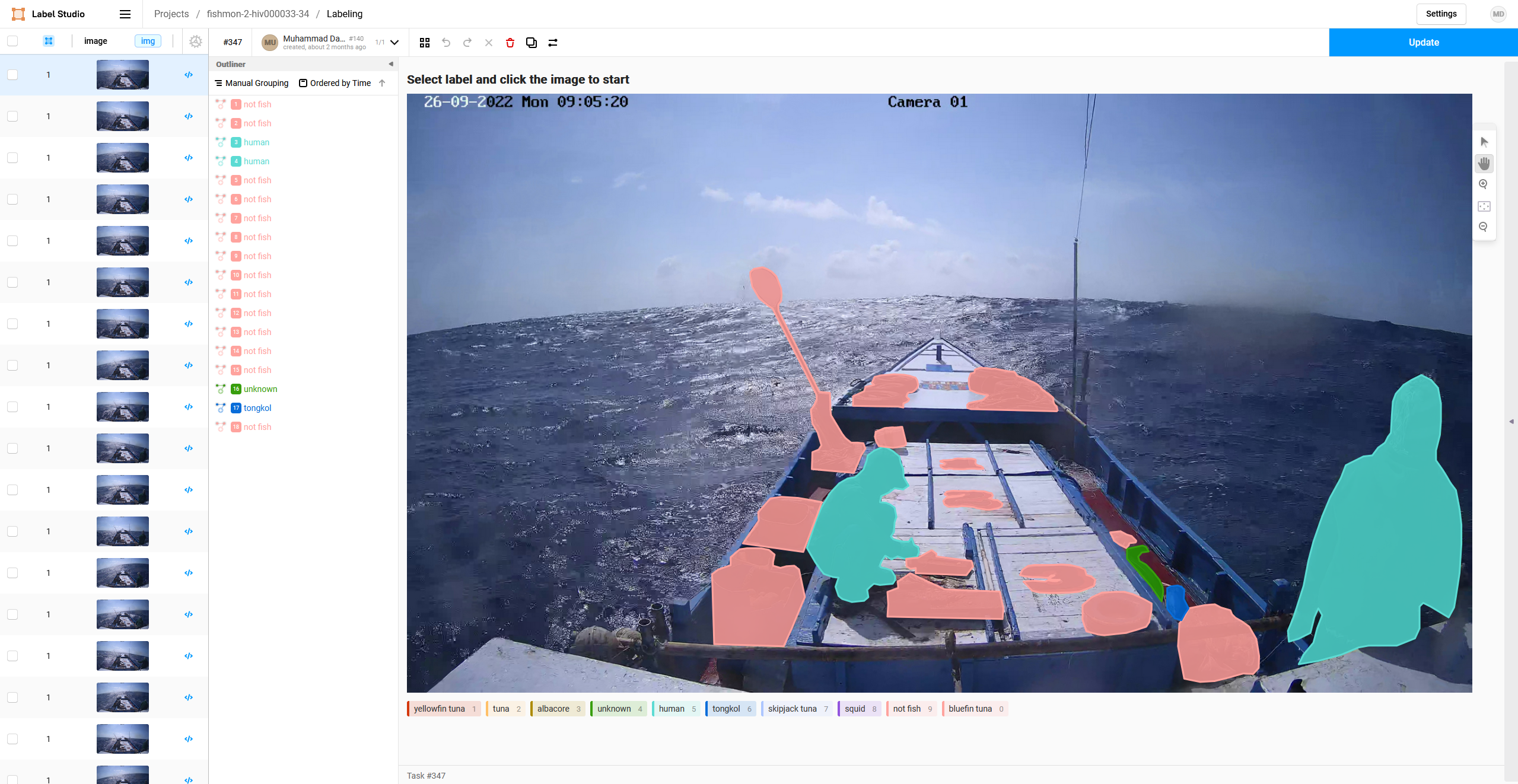
## *Preprocessing* Data



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 dilabeli 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.



Gambar 3.3 Proses *labelling* dengan Label Studio [15]

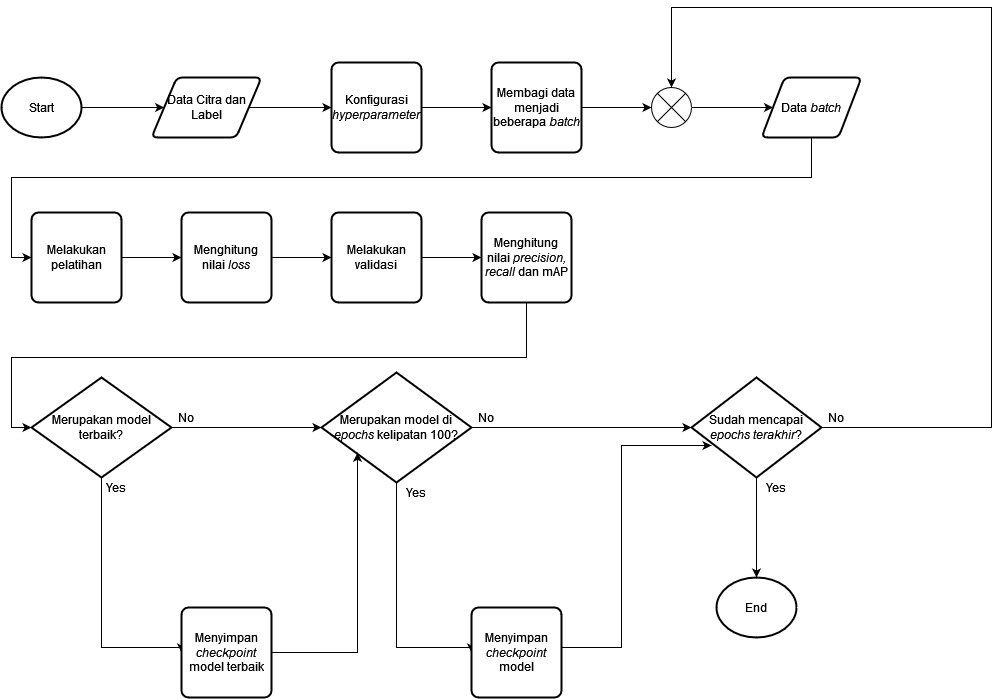
Setelah semua citra telah dilabeli, citra yang ada di Label Studio akan diekspor menjadi format *dataset* yang sesuai dengan model yang akan digunakan. *Dataset* tersebut kemudian akan dibagi menjadi untuk proses *training, validation* dan *testing*. Dalam proses ini juga, *dataset* akan diperiksa untuk 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.

## 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* validasi 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 25. Jika model tersebut terbentuk pada *epochs* kelipatan 25, model tersebut akan tersimpan sebagai *checkpoint* model setiap 25 *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

## 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.

# IMPLEMENTASI DAN ANALISIS SISTEM DETEKSI MULTI OBJEK BERBASIS YOLOV7

## 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 mengaktivasi 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

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

## 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 4.1 dan tabel 4.2

Tabel 4.2 Persebaran citra pada dataset

|  |  |
| --- | --- |
| **Nama Bagian** | **Jumlah Citra** |
| *train* | 2.999 |
| *validation* | 641 |
| *Test* | 640 |
| **Total** | 4.280 |

Tabel 4.3 Jumlah data tiap objek pada dataset

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

## 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.

## Pengujian Model

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

### Pengujian Pengaruh *Dataset* Berbentuk *Polynomial Segmentation* Terhadap Akurasi

Outline:

* Jelasin tujuan pengujian
* Jelasin segmentasi tuh apa dan biasanya penggunaan datasetnya buat apa
* Proses pengujiannya kaya gmn
* Hipotesis awal
* Hasilnya kaya apa
* Analisis

### Pengujian Perbandingan Metode Object Detection dengan Instance Segmentation Terhadap Akurasi

* Tujuan pengujian
* Hipotesis awal
* Proses pengujiannya
* Hasilnya kaya apa
* Analisis

### Pengujian Pengaruh Ukuran Gambar Terhadap Akurasi dan Jumlah FPS

* Jelasin tujuan
* Hipotesis awal
* Proses pengujian
* Hasilnya kaya apa
* analisis

### Pengujian Pengaruh Optimizers Terhadap Akurasi dan Jumlah FPS

* Tujuan pengujian
* Jelasin tiap optimizersnya singkat
* Hipotesis awal
* Proses pengujian
* Hasilnya kaya apa
* Analisis

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

* Tujuan pengujian
* Early stopping teh naon jelasin?
* Hipotesis awal
* Pengujian
* Hasilnya kaya apa
* Analisis

### Pengujian Pendeteksian Model dengan Jetson Nano

* Tujuan
* Hipotesis awal
* Pengujian
* Hasilnya kaya apa
* Analisis

## Dampak Terhadap Lingkungan dan Masyarakat

* Dapat mencegah terjadinya illegal fishing
* Dapat menambah devisa negara di sektor perikanan
* Menjaga ekosistem kelautan Indonesia tetap baik

# KESIMPULAN

## 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:

## Saran

# DAFTAR PUSTAKA

[1] “Fish; fillets, frozen exports by country |2021.” https://wits.worldbank.org/trade/comtrade/en/country/ALL/year/2021/tradeflow/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] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Dec. 2016, vol. 2016-December, pp. 779–788. doi: 10.1109/CVPR.2016.91.

[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] “Description.” https://www.fishnet.ai/description (accessed Dec. 16, 2022).

[15] “Label Studio.” https://fishmon-tagging.aiseeyou.tech/projects/?page=1 (accessed Dec. 17, 2022).