Deteksi Kesegaran Daging Ikan Yang Bersifat Non-Destructive Pada Aplikasi Mobile Menggunakan YOLOv4 dan YOLOv4-Tiny

Malik Abdul Aziz
19917009

Tesis diajukan sebagai syarat untuk meraih gelar Magister Komputer
Konsentrasi Sains Data
Program Studi Informatika Program Magister
Fakultas Teknologi Industri
Universitas Islam Indonesia
2023

Lembar Pengesahan Pembimbing

Deteksi Kesegaran Daging Ikan Yang Bersifat Non-Destructive Pada Aplikasi Mobile Menggunakan YOLOv4 dan YOLOv4-Tiny

Malik Abdul Aziz
19917009

Yogyakarta, 23 Agustus 2023

Pembimbing I	Pembimbing II
Dhomas Hatta Fudholi, S.T., M. Eng., Ph.D.	Arrie Kurniawardhani, S.Si., M.Kom.

Lembar Pengesahan Penguji

Deteksi Kesegaran Daging Ikan Yang Bersifat Non-Destructive Pada Aplikasi Mobile Menggunakan YOLOv4 dan YOLOv4-Tiny

Malik Abdul Aziz

19917009

Yogyakarta, 29 Agustus 2023

Tim Penguji,

Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D.
Ketua

Dr. Syarif Hidayat, S.Kom., M.I.T.
Anggota I

Chandra Kusuma Dewa, M.Kom., M.Cs., Ph.D.s
Anggota II

Mengetahui,
Ketua Program Studi Teknik Informatika Program Magister
Fakultas Teknologi Industri
Universitas Islam Indonesia

Irving Vitra Paputungan, S.T., M.Sc., Ph.D.

Abstrak

Deteksi Kesegaran Daging Ikan Yang Bersifat Non-Destructive Pada Aplikasi Mobile Menggunakan YOLOv4 dan YOLOv4-Tiny

Ikan merupakan salah satu bahan konsumsi yang menyediakan protein berkualitas tinggi dan dapat membantu dalam membentuk pola hidup sehat. Mutu daging ikan yang mudah rusak, mengharuskan konsumen cerdas dalam memilah ikan yang akan dikonsumsi. Oleh sebab itu, pengetahuan terhadap kondisi kesegaran daging ikan menjadi hal penting bagi para konsumen. Pada penelitian terdahulu, beberapa cara yang digunakan untuk menilai tingkat kesegaran daging ikan cukup menyulitkan user ataupun konsumen. Beberapa diantaranya memerlukan alat seperti *spectometer*, sensor gas, hingga sebuah komputer dalam memperoleh parameter dan memproses data untuk menilai tingkat kesegaran daging ikan. Penelitian ini mencoba membangun sebuah aplikasi berbasis *Mobile* yang menerapkan *deep learning* model, menggunakan arsitektur *You Look Only Once Version* I (YOLOv4) dan YOLOv4-Tiny untuk menilai tingkat kesegaran daging ikan berdasarkan mata dan kulit ikan. Tingkat kesegaran yang digunakan adalah segar, sedang, dan busuk. Data latih yang digunakan oleh model berupa gambar ikan Tongkol Deho (*Euthynnus Affinis*), ikan Manglah (*Priacanthus Tayenus*), ikan Solok (*Rastrelliger Brachysoma*), ikan Mackerel (*Scomber Australasicus*), ikan Kuwe Lilin (*Caranx Melanophygus*), ikan Teribang (*Nemipterus Virgatus*), ikan Banyar (*Restrelliger Kanagurta*), dan ikan Kolong (*Atule mate*). mAP yang dicapai oleh YOLOv4 adalah sebesar 99.17% dan YOLOv4-Tiny sebesar 97.25%. Dengan waktu proses tercepat menilai tingkat kesegaran daging ikan pada aplikasi mencapai 2.5 detik per Image untuk YOLOv4 dan 0.15 detik untuk YOLOv4-Tiny.

Kata kunci

Aplikasi Mobile, Deep Learning, Kesegaran Daging Ikan, YOLOv4, YOLOv4-Tiny

Abstract

Non-Destructive Fish Freshness Detection System for Mobile Application Using YOLOv4 And YOLOv4-Tiny

Fish is one of the ingredients for consumption that provides high-quality protein and can help form a healthy lifestyle. The perishable quality of fish meat demands that consumers be smart in sorting out the fish to be consumed. Therefore, knowledge about the freshness condition of fish meat is important for consumers. In previous research, several methods used to assess the freshness of fish meat were quite difficult for users or consumers. Some of them require tools such as spectrometers, gas sensors, and computers to obtain parameters and process data to assess the degree of freshness of fish meat. This study tries to build a mobile-based application that applies the Deep Learning model, using architecture You Look Only Once (YOLOv4) and YOLOv4-Tiny to detect the freshness level of fish based on the eyes and skin of the fish. The level of freshness used is fresh, medium, and spoil. The dataset used by the model are images of Deho tuna fish (Euthynnus Affinis), Manglah fish (Priacanthus Tayenus), Solok fish (Rastrelliger Brachysoma), Mackerel fish (Scomber Australasicus), Kuwe Lilin fish (Caranx Melanophygus), Teribang fish (Nemipterus virgatus), Banyar fish (Restrelliger Kanagurta), and Kolong fish (Atule mate). The mAP achieved by YOLOv4 is 99.17% and YOLOv4-Tiny is 97.25%. The fastest processing time, the freshness level of fish meat in the application reaches 2.5 seconds per image for YOLOv4 and 0.15 seconds for YOLOv4-Tiny.

Keywords

Fish Freshness Detection, Deep Learning, Mobile Application, YOLOv4, YOLOv4-Tiny

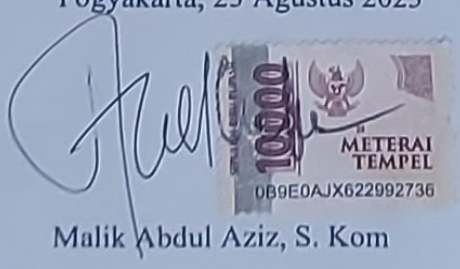
Pernyataan Keaslian Tulisan

Dengan ini saya menyatakan bahwa tesis ini merupakan tulisan asli dari penulis, dan tidak berisi material yang telah diterbitkan sebelumnya atau tulisan dari penulis lain terkecuali referensi atas material tersebut telah disebutkan dalam tesis. Apabila ada kontribusi dari penulis lain dalam tesis ini, maka penulis lain tersebut secara eksplisit telah disebutkan dalam tesis ini.

Dengan ini saya juga menyatakan bahwa segala kontribusi dari pihak lain terhadap tesis ini, termasuk bantuan analisis statistik, desain survei, analisis data, prosedur teknis yang bersifat signifikan, dan segala bentuk aktivitas penelitian yang dipergunakan atau dilaporkan dalam tesis ini telah secara eksplisit disebutkan dalam tesis ini.

Segala bentuk hak cipta yang terdapat dalam material dokumen tesis ini berada dalam kepemilikan pemilik hak cipta masing-masing. Apabila dibutuhkan, penulis juga telah mendapatkan izin dari pemilik hak cipta untuk menggunakan ulang materialnya dalam tesis ini.

Yogyakarta, 23 Agustus 2023



Malik Abdul Aziz, S. Kom

Daftar Publikasi

Publikasi yang menjadi bagian dari tesis

Aziz, A. A., Fudholi, D. H., Kurniawardhani, A. (2023). Deteksi Kesegaran Daging Ikan Bersifat Non-Destructive Pada Aplikasi Mobile Menggunakan YOLOv4 dan YOLOv4-Tiny. Jurnal Teknik Informatika dan Sistem Informasi. Vol 10, No. 4, (2023)

Sitasi publikasi 1

Kontributor	Jenis Kontribusi
Malik Abdul Aziz	Melakukan komputasi, analisis, *development* aplikasi, dan menulis *paper*
Dhomas Hatta Fudholi	Melakukan analisis model dan mengedit *paper*
Arrie Kurniawardhani	Melakukan analisis model dan mengedit *paper*

Halaman Kontribusi

Dalam penulisan tesis ini pembimbing I dan II memberikan beberapa masukkan sebagai perbaikan dari cara penulisan tesis serta memberikan saran tentang data yang akan diolah, dan dianalisis.

Halaman Persembahan

بسْم اللَّهِ الرَّحْمن الرَّحِيْم

Kupersembahkan Tesis ku ini dengan setulus hati & sepenuh jiwa untuk : Orang tuaku tercinta Ibuku Supaiyah dan Ayahku Parto. Yang selalu mengiringi langkah kakiku dengan do'a, yang tak pernah putus asa untuk mengurus, mendidik, menasehati, menyemangati, membimbing serta memberikan motivasi kepadaku.

Istriku, Annisa Fitri Nur Laili. Terimakasih telah memberikan *support* terbaikmu, agar aku dapat menyelesaikan pendidikan program Magisterku.

Kata Pengantar

بسْم اللَّهِ الرَّحْمن الرَّحِيْم

Alhamdulillahirabbil'alamin, segala puji syukur kehadirat Allah SWT senantiasa penulis panjatkan karena atas limpahan rahmat dan hidayah-Nya, tesis yang berjudul "Deteksi Kesegaran Daging Ikan Yang Bersifat Non-Destructive Pada Aplikasi Mobile Menggunakan YOLOv4 dan YOLOv4-Tiny" dapat berjalan dengan lancar dalam penyelesaiannya. Tesis ini diajukan sebagai bagian dalam menyelesaikan studi dan sebagai salah satu syarat untuk memperoleh gelar Magister Komputer pada Program Studi Magister Teknik Informatika Fakultas Teknologi Industri Universitas Islam Indonesia.

Dalam penyelesaian Tesis ini, penulis banyak mendapatkan bantuan dari berbagai pihak, untuk itu penulis menyampaikan ucapan terima kasih setulusnya kepada:

1. Bapak Prof. Fathul Wahid, S.T., M.Sc., Ph.D., selaku Rektor Universitas Islam Indonesia.
2. Bapak Irving Vitra Paputungan, S.T., M.Sc., Ph.D., selaku Ketua Program Studi Teknik Informatika Program Studi Magister Universitas Islam Indonesia.
3. Bapak Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D., selaku dosen pembimbing satu, yang telah banyak membantu penulis dalam memberikan semangat, ide, saran dan kritiknya.
4. Ibu Arrie Kurniawardhani, S.Si., M.Kom., selaku dosen pembimbing dua yang banyak memberikan kemudahan baik pengarahan maupun bimbingan selama pengajuan dan pengerjaan Tesis.
5. Bapak Dr. Syarif Hidayat, S.Kom., M.I.T., selaku dewan penguji anggota I tesis yang telah memberikan banyak pengarahan dan masukan dalam penyusunan danpenyempurnaan Tesis ini
6. Bapak Chandra Kusuma Dewa, M.Kom., M.Cs., Ph.D., selaku dewan penguji anggota II tesis yang telah memberikan banyak pengarahan dan masukan dalam penyusunan dan penyempurnaan Tesis ini.
7. Dosen Program Studi Magister Teknik Informatika yang telah memberikan bekal ilmu pengetahuan kepada penulis, semoga ilmunya menjadi amal jariyah di dunia maupun akhirat.

8. Staff Akademik Program Pascasarjana Fakultas Teknologi Universitas Islam Indonesia, yang telah membantu dalam segala urusan administrasi di kampus.
9. Sahabat seperjuangan Sains Data (Yurio, Malik, Fahmi, Yopi, Satya, Windi, Suryatin, dan Rifai).
10. Teman-teman Magister Teknik Informatika Universitas Islam Indonesia Yogyakarta se- angkatan.

Penulis menyadari bahwa dalam penulisan tesis ini masih banyak kelemahan dan kekurangan. Oleh karena itu, kritik dan saran yang sifatnya membangun sangat penulis harapkan agar tesis ini dapat menjadi lebih baik.

Daftar Isi

Daftar Tabel

Daftar Gambar

BAB 1
Pendahuluan

1.1 Pendahuluan

Ikan merupakan bahan pangan yang terdiri dari air, protein, lemak, karbohidrat serta beberapa vitamin, dan mineral seperti kalium, kalsium, magnesium, dan klorida. Selain itu daging ikan juga memiliki kandungan asam amino dan berprotein tinggi (Suhartini & Hidayat, 2005). Secara ketahanan mutunya, daging ikan merupakan komoditi yang cepat busuk. Oleh karenanya penanganan daging ini merupakan salah satu bagian penting pada mata rantai industri (Afrianto & Liviawaty, 1989). Memperhatikan faktor pembusukan daging ikan yang relatif cepat, penilaian terhadap kelayakan konsumsi daging ikan merupakan pengetahuan yang sangat diperlukan oleh para konsumen dari berbagai kalangan (perorangan ataupun industri). Pengetahuan ini jarang dimiliki oleh masyarakat pada umumnya, sehingga penelitian ini mencoba untuk mengusulkan sebuah sistem cerdas yang dapat memberikan informasi tingkat kesegaran dari daging ikan berdasarkan tampilan dari mata dan kulit ikan.

Beberapa pemantauan penilaian kesegaran daging ikan terhadap tampilan dari citra ikan, sebelumnya telah dilakukan pada ikan Mas (*Cyprinus Carpio*) yang mencangkup kulit dan mata dengan mengaplikasikan *Deep CNN* (*Convolutional Neural Network*) dimana akurasi yang diperoleh sebesar 98.2%. Dalam pengaplikasiannya, arsitektur VGG-16 digunakan untuk melakukan ekstraksi fitur dari ikan pada gambar. Selanjutnya, dikembangkan blok pengklasifikasi yang terdiri dari *dropout* dan *fully connected layer* membedakan gambar ikan berdasarkan kesegarannya (Taheri-Garavand et al., 2020). Penelitian serupa juga diterapkan pada ikan *Threadfin* (*Scomberoides Commersonnianus*) dan *Queenfish* (*Scomberoides Commersonnianus*). Penelitian tersebut memanfaatkan *Quality Index Method* (QIM) dan nilai RGB dari citra ikan untuk dapat menilai kesegaran dari ikan. Relasi dari hasil QIM dengan perubahan nilai RGB selama empat belas hari, dilihat lebih lanjut menggunakan *Statistical Package for the Social Science* (SPSS) (Abd Aziz et al., 2021).

Atribut lain yang dapat menilai tingkat kesegaran daging ikan adalah bau, tekstur, dan rasa. Dengan memanfaatkan sensor MQ2, MQ3, MQ6, MQ9, dan MQ135 yang merupakan bagian dari sensor *metal oxide*, perubahan bau yang dihasilkan oleh ikan segar

kebusuk dapat terdeteksi dengan baik. Akurasi dari penelitian yang menghasilkan sistem tersebut sebesar 97% (Leghrib et al., 2020).

Deep learning pada beberapa penelitian lain, juga dilakukan untuk menilai data spectral pada daging ikan *fillet* menggunakan *spectrometer*. Penelitian pertama memanfaatkan CNN untuk menganalisis data *spectral* yang dihasilkan, yang kemudian memverifikasinya berdasarkan kandungan *Power of Hidrogen* (pH) pada daging (Moon et al., 2020). Pada penelitian yang lain, data *spectral* dikombinasikan dengan citra dari *fillet* daging ikan dengan memanfaatkan CNN untuk melakukan ekstraksi fitur pada gambar daging *fillet*. Akurasi yang diperoleh dari kombinasi data *spectral* dengan citra daging *fillet* sebesar 92.3% (L. Wu et al., 2019).

Pada penelitian ini, penulis akan melakukan pendekatan yang berbeda dari penelitian sebelumnya dalam menilai kesegaran daging ikan, yaitu dengan memanfaatkan deteksi objek untuk mendapatkan posisi dari ikan. Selanjutnya, ikan tersebut dinilai tingkat kesegarannya berdasarkan tampilan dari mata dan kulit ikan (Riley, 2005; Saputra et al., 2022) pada sebuah citra digital dengan mengimplementasikannya pada aplikasi berbasis *Android*. Adapun daging ikan yang menjadi fokus penelitian meliputi ikan Tongkol Deho (*Euthynnus Affinis*), ikan Manglah (*Priacanthus Tayenus*), ikan Solok (*Rastrelliger Brachysoma*), ikan *Mackerel* (*Scomber Australasicus*), ikan Kuwe Lilin (*Caranx Melanophygus*), ikan Teribang (*Nemipterus Virgatus*), ikan Banyar (*Restrelliger Kanagurta*), dan ikan Kolong (*Atule mate*). Terdapat tiga level kesegeraan yang digunakan untuk penilaian, yaitu busuk, sedang, segar (Moon et al., 2020). Model *deep learning* yang akan digunakan untuk mendeteksi objek adalah YOLO dengan arsitektur YOLOv4 dan YOLOv4-Tiny.

1.2 Rumusan Masalah

Berdasarkan pembahasan pada latar belakang, maka rumusan masalah pada penelitian ini adalah bagaimana membangun model deteksi kesegaran daging ikan berdasarkan mata dan kulit ikan menggunakan YOLOv4 dan YOLOv4-Tiny.

1.3 Tujuan Penelitian

1. Mengimplementasikan Model YOLOv4 dan YOLOv4-Tiny untuk menilai tingkat kesegaran mata dan kulit ikan.
2. Membuat *rule* untuk menilai tingkat kesegaran daging ikan berdasarkan *level* kesegaran mata dan kulit ikan yang dinilai oleh model YOLOv4 dan YOLOv4-Tiny.

3. Mengimplementasikan Model YOLOv4 dan YOLOv4-Tiny yang telah dilatih ke dalam aplikasi *smartphone* berbasis *android*.

1.4 Batasan Masalah

Adapun batasan masalah pada penelitian ini mencangkup:

1. Objek penelitian untuk ikan laut yaitu ikan Tongkol Deho (*Euthynnus Affinis*), ikan Manglah (*Priacanthus Tayenus*), ikan Solok (Rastrelliger Brachysoma), ikan Mackerel (*Scomber Australasicus*), ikan Kuwe Lilin (*Caranx Melanophygus*), ikan Teribang (*Nemipterus Virgatus*), ikan Banyar (*Restrelliger Kanagurta*), dan ikan Kolong (*Atule mate*).
2. Pengkategorian tingkat kesegaran daging menggunakan busuk, sedang, segar.
3. Parameter pengukur kesegaran daging ikan menggunakan mata dan kulit ikan.
4. Tempat pengambilan ikan berlokasi di Pelabuhan Muncar, kecamatan Muncar, kabupaten Banyuwangi.

1.5 Manfaat Penelitian

1. Manfaat Akademis
   a. Dapat memberikan informasi dan pengetahuan tingkat kesegaran daging ikan bagi khalayak umum yang akan mengkonsumsi daging ikan.
   b. Menambah wawasan tentang implementasi *deep learning* pada proses klasifikasi.
   c. Memberikan informasi yang berguna, khususnya pada ruang lingkup ketahanan mutu daging ikan.
   d. Sebagai acuan atau rujukan untuk penelitian selanjutnya yang ingin melakukan penelitian tentang mendeteksi tingkat kesegaran daging ikan.
2. Manfaat Praktis

   Dapat menjadi referensi ilmiah yang berguna untuk menentukan daging ikan dalam skala kecil seperti masakan olahan rumah hingga skala besar (industri).

1.6 Sistematika Penulisan

1. Abstrak berisi rangkuman laporan isi laporan secara umum.
2. Pernyataan Keaslian Tulisan.
3. Daftar Publikasi
4. Halaman Kontribusi berisi dosen yang terlibat dalam penelitian.

5. Halaman Persembahan berisi ucapan terhadap keluarga.
6. Kata Pengantar berisi ucapan terimakasih kepada berbagai pihak yang terlibat dalam penelitian.
7. Daftar Isi berisi daftar judul bab, dan sub bab laporan.
8. Daftar Tabel berisi daftar nama tabel.
9. Daftar Gambar berisi daftar nama Gambar.
10. BAB 1 Pendahuluan
11. BAB 2 Tinjauan Pustaka
12. BAB 3 Metodologi
13. BAB 4 Hasil Dan Pembahasan
14. BAB 5 Kesimpulan Dan Saran
15. Daftar Pustaka berisi rujukan yang dikutip dalam laporan.
16. Lampiran berisi alamat github yang berisi hasil penelitian yang telah dilakukan.

BAB 2
Tinjauan Pustaka

2.1 Penelitian Terdahulu

Mengkonsumsi secara teratur daging ikan segar, dapat menyediakan protein berkualitas tinggi dan nutrisi untuk kesehatan. Selain itu, mengkonsumsi ikan berkaitan dengan pengurangan risiko penyakit jantung koroner dan meningkatkan perkembangan saraf. Bahkan, mengkonsumsi ikan dapat menurunkan faktor risiko sindrom metabolik, untuk ikan yang kaya akan asam lemak tak jenuh ganda (Maciel et al., 2019). Secara ketahanan mutu, ikan segar merupakan produk yang sangat mudah rusak karena komposisi kimia dari daging ikan dan jumlah mikroba yang tinggi pada kulit ikan (Tsironi et al., 2020). Memperhatikan faktor pembusukan daging ikan yang relatif cepat, informasi terhadap tingkat kesegaran daging ikan, merupakan hal penting yang diperlukan bagi para konsumen. Baik konsumen untuk skala kecil maupun skala besar. Atribut yang dapat digunakan untuk menilai kesegaran daging ikan diantaranya adalah reaksi kimiawi, fisik ikan, mikroba pada ikan, dan atribut sensorik (tampilan, warna, tekstur, dan lain sebagainya) (L. Wu et al., 2019). Tabel 2. memperlihatkan perbedaan tampilan dari ikan segar dan busuk yang disebutkan oleh Lougovois dan Kyrana (Riley, 2005).

Tabel 2.1 Perbandingan Tampilan Kulit Serta Mata Ikan Segar Dan Busuk

Parameter	Ikan Segar	Ikan Busuk
Kulit	Berkilau, berwarna-warni yang dilapisi dengan lapisan tipis yang tersebar merata dan lendir yang hampir transparan	Kulit kehilangan bentuknya yang mengembang, mulai kusam, memutih dan kasar saat disentuh
Mata	Berwarna cerah, berbentuk kembung dengan pupil yang hitam pekat dan kornea yang masih transparan	Berangsur menyusut dan secara bertahap beralih dari datar menjadi cekung. Pupil mengeruh seperti susu. Dan kornea menjadi buram.

Deep Learning merupakan *subfield* dari *machine learning*. *Machine learning* sendiri merupakan bagian *artificial intelligence*. Pendekatan pada *deep learning*, dapat dikategorikan ke dalam *supervised learning*, *semi- supervised searning*, *unsupervised*

learning, *reinforcement learning* (Alom et al., 2019). Penelitian ini akan berfokus pada kategori pertama yaitu *supervised learni*ng. Untuk dapat mengaplikasikannya, penulis akan memanfaatkan teknik *object detection* dan klasifikasi. Model *deep learning* yang akan digunakan merupakan model yang digolongkan ke dalam *single stage*. Dimana model ini memprioritaskan kecepatan dengan akurasi yang stabil. Contoh dari model *single stage* adalah YOLO (Soviany & Ionescu, 2018).

Sebelumnya Taheri-Garavand, A. et al., (2020) telah melakukan penilaian dan *monitoring* kesegaran pada ikan Mas (*Cyprinus Carpio*) dengan akurasi sebesar 98.21% yang memanfaatkan *Deep CNN*. Pada penelitian tersebut, ikan disimpan pada *thermocol* dengan es dan disimpan selama waktu dua minggu. Kemudian akan diperiksa setiap hari untuk mengambil sampel gambarnya. VGG-16 dipilih pada penelitian ini dalam hal ekstraksi fitur secara otomatis dan kemudian dikombinasikan dengan blok pengklasifikasi yang dikembangkan untuk membedakan fitur dalam pada ikan yang masih segar dan tidak. Pada penelitian, fokus utamanya adalah mengklasifikasikan kesegaran daging ikan

Moon, E. J. et al., (2020) melakukan penilain kesegaran pada daging Salmon, Tuna, dan sapi dengan menggunakan alat *portable Visible* / *Near-Infrared* (VIS/NIR) s*pectrometer* yang tersedia secara komersial dijual kepada publik. Alat tersebut diperlukan untuk memperoleh data *spectral* yang selanjutnya dianalisis model CNN untuk mengevaluasi kesegaran dari objek. Untuk proses verifikasi, pengukuran pH (*Power of Hydrogen*) digunakan untuk mengkategorikan kesegaran dari ikan ke dalam segar, kemungkinan busuk, dan busuk. penelitian tersebut mencapai akurasi sebesar 85% untuk Salmon, 88% untuk ikan Tuna, dan 92% untuk daging sapi. Kekurangan pada penelitian ini terletak pada daging yang diukur oleh VIS/NIR, data tersebut merupakan potongan daging bukan ikan secara utuh. Hal tersebut mengharuskan daging ikan untuk dipotong terlebih dahulu agar dapat dilihat pada bagian daging supaya dipastikan kesegarannya. Cara ini bersifat merusak apabila konsumen ingin menilai dagingnya secara utuh.

Pada penelitian yang lain, Wu, T. et al., (2019) juga melakukan penilaian kesegaran daging ikan salmon pada data spektral yang dipadukan dengan gambar *fillet* pada daging tersebut. Penelitian tersebut, menggunakan model *neural network CNN* untuk mengekstraksi gambar fillet ikan dengan data spektral untuk dapat memprediksi nilai TVC dan TVB-N dari daging ikan. Untuk mengurangi dampak negatif dari gambar fillet ikan yang memiliki dimensi yang besar, metode PCA (*Principal Component Analysis*)dipilih untuk melakukan reduksi ukuran dari setiap gambar ikan salmon. Keberhasilan dari kombinasi data spectral dan gambar *fillet* ikan salmon ini mencapai 92.3%. Pada penelitian ini, kelemahannya tidak

jauh berbeda dengan yang dilakukan oleh Moon, E. J. et al., (2020). Daging yang akan dinilai harus dipotong terlebih dahulu agar dapat melihat daging bagian dalamnya.

Penelitian yang memanfaatkan tampilan dari kulit ikan sebelumnya juga dilakukan oleh Aziz, Naim K. et al., (2021). Dengan tujuan menghasilkan detektor tingkat kesegaran ikan, penelitian tersebut memadukan QIM dengan nilai *Red*, *Green*, *Blue* (RGB) pada sebuah gambar ikan jenis *Threadfin* dan *Queenfish*. Pada sisi gambar, nilai yang diekstrak hanya pada RGB-nya. Sedangkan jika memperhatikan setiap perubahan tampilan pada datanya, ada perubahan detail seperti memudarnya bintik ikan dan munculnya lendir masih dapat diproses. Selain itu, gambar yang dianalisis dalam kurun waktu per hari, satu gambar untuk satu hari.

Penelitian kesegaran daging ikan dengan pendekatan yang berbeda juga telah dilakukan oleh Leghrib. et al (2020). Dalam eksperimennya, Leghrib mengusulkan sebuah sistem *portable* yang berdasarkan sensor gas. Sistem ini memanfaatkan lima sensor gas *Metal Oxide* yang meliputi MQ2 (Sensitif terhadap Metana, Butana, dan asap), MQ3 (Alkohol, Etanol, dan asap), MQ6 (Liquified Petroleum, Butana, dan asap) MQ9 (Karbon Monoksida dan gas yang mudah terbakar), dan MQ135 (untuk kualitas udara). Pemeriksaan lebih lanjut terhadap kinerja sistem, penelitian tersebut memanfaatkan PCA untuk dapat melakukan *plotting*. Akurasi yang diperoleh dari penelitian tersebut sebesar 97%. Penelitian ini cukup kompleks bila dilihat dari alatnya. Selain itu, alat ini belum diuji coba di lapangan seperti pasar atau minimarket. Ada kemungkinan, penggunaan alat ini dalam mendeteksi gas dari ikan bisa bercampur dengan gas-gas lain disekitarnya.

Pada kesempatan yang lain, Miao. et al, (2017) melakukan penelitian terhadap pengaruh kesegaran ikan tuna yang berbeda pada kualitas steak tuna. Dalam menilai kesegaran daging ikan, penelitian tersebut memanfaatkan zat yang berkaitan dengan ATP (*Adenosine Triphosphate*), nilai *K value* (rasio jumlah Inosin (HxR) dan hipoksantin (Hx) terhadap jumlah ATP dan degradasi produk) yang ditentukan berdasarkan HPLC (LC-20A, Shimadzu, Jepang), Histamin (Histamin adalah amina biogenik yang berkadar tinggi dalam jaringan otot ikan Scombroid), kualitas sensorik, *E-Nose* (menilai perbedaan aroma dari variasi sampel ikan), dan pemanfaatan *E-Tongue* (untuk menilai *supernatant*). *K value*, Histamin, dan ATP diukur berdasarkan HPLC (LC-20A, Shimadzu, Jepang), evaluasi kualitas sensorik berdasarkan QIM, analisis *E-Nose* menggunakan FOXα 4000 Sensor Array System (Alpha M.O.S France), dan *E-Tongue* menggunakan ASTREE (Alpha M.O.S France). Keseluruhan nilai yang diperoleh, selanjutnya dianalisis menggunakan SPSS dan Microsoft Excel. Hasilnya menunjukkan, kesegaran ikan tuna mentah yang menurun dari

Sashimi grade ke *cooking* grade mempengaruhi kualitas setelah dimasak. Proses yang komplek menjadi kekurangan pada penelitian ini apabila diterapkan pada kalangan masyarakat umum.

YOLOv4 telah menunjukkan keberhasilannya dalam melakukan deteksi objek pada berbagai bidang. Contohnya, Li dkk (2022) memanfaatkan YOLOv4 dalam mendeteksi kuncup bunga kiwi secara *realtime* sebagai sarana penyerbukan oleh robot agar menghemat biaya operasinal. Pada penelitian yang dilakukan oleh Li dkk, YOLOv3 dan YOLOv4 dilatih dan dikomparasi hasilnya untuk melihat model yang lebih baik dalam mendeteksi kuncup bunga kiwi. Hasilnya menunjukkan, *Mean Average Precision* (mAP) milik YOLOv4 lebih unggul dari YOLOv3. YOLOv4 memperoleh mAP sebesar 97.61% dan YOLOv3 memperoleh mAP sebesar 95,24% (Li et al., 2022).

Pada penelitian yang lain, Kim dkk (2020) menguji kemampuan YOLOv4, dengan Faster R-CNN (*Region-based Convolutional Network*) dan SSD (*Single Shot Detection*) dalam mengenali tipe kendaaran secara *realtime* pada sebuah video. Dari hasil kajian tersebut, mAP milik YOLOv4 memiliki *score* paling tinggi, yaitu 98.19%. Sedangkan untuk Faster R-CNN memiliki *score* mAP 93.40% dan SSD 90.56%. Meskipun model SSD memiliki tingkat akurasi paling rendah, model tersebut memiliki kecepatan proses paling baik jika dibandingkan dengan dua model yang lain (J. A. Kim et al., 2020).

State of the art dari referensi kajian sebelumnya yang digunakan dalam penelitian ini dijelaskan secara singkat pada Tabel 2.2.

Tabel 2.2 State of the art penelitian

Peneliti	**Bidang / Tema**	**Keterangan / Alat Analisis**
(Taheri-Garavand et al., 2020)	*Deep learning* untuk menilai tingkat kesegara daging ikan	Penggunaan arsitek *deep learning* VGG-16 untuk ekstraksi fitur dan mengklasifikasikan tingkat kesegaran daging ikan.
(Moon et al., 2020)	Penilaian kesegaran daging ikan dengan nilai *spectral*	Analisis CNN pada nilai VIS/NIR yang dikombinasikan dengan nilai pH untuk mengukur kesegaran daging ikan

Tabel 2.2 Lanjutan.

Peneliti	**Bidang / Tema**	**Keterangan / Alat Analisis**
(T. Wu et al., 2019)	Penilaian kesegaran daging ikan dengan nilai *spectral*	Penilaian kesegaran daging ikan menggunakan nilai RGB pada gambar dan nilai *spectral* dari ikan, kemudian diprediksi kesegarannnya berdasarkan TVC dan TVB-N
(Abd Aziz et al., 2021)	Penilaian kesegaran daging ikan menggunakan QIM	QIM dan *software* SPSS untuk menganalisis relasi nilai RGB dari perubahan level kesegaran daging ikan
(Miao et al., 2017)	Penilaian kesegaran daging ikan memanfaatkan reaksi zat kimia pada ikan	Analisis tingkat kesegaran daging ikan berdasarkan zat kimia menggunakan SPSS dan Excel
(Leghrib et al., 2020)	Sensor gas untuk menilai kesegaran daging ikan	Perancangan sensor gas *metal oxyde* untuk menilai gas yang dihasilkan dari kesegaran daging ikan
(Saputra et al., 2022)	*Machine learning* untuk menilai kesegaran daging ikan	Ekstrasi dan analisis nilai sebaran RGB pada mata menggunakan Naïve Bayes
(Li et al., 2022)	*Deep learning* Model untuk mendeteksi objek	YOLOv3 dan YOLOv4 digunakan untuk mendeteksi kuncup bunga kiwi.
(J. A. Kim et al., 2020)	*Deep learning* Model untuk mendeteksi objek	Perbandingan mAP dan kecepatan proses Faster R-CNN, SSD dan YOLOv4 dalam mengenali tipe kendaraan.

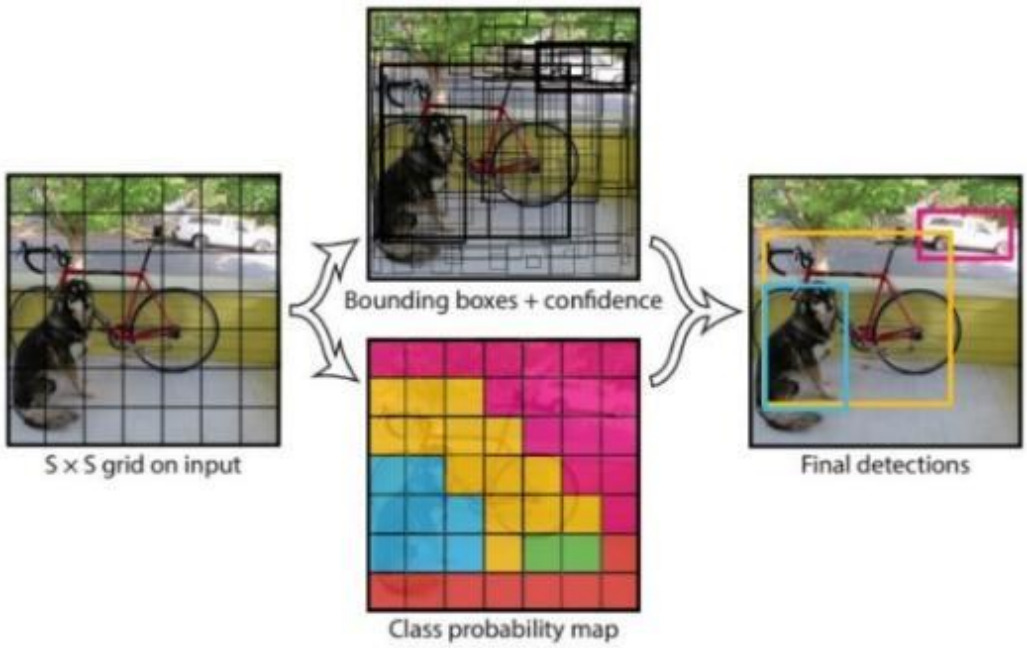
2.2 Hipotesis Untuk Penelitian Terdahulu

Hipotesis berikut didasarkan pada penelitian-penelitian sebelumnya terkait penliaian tingkat kesegaran daging ikan. Baik pendekatannya melalui *deep learning* ataupun tidak. maka hipotesis yang ada pada penelitian ini adalah sebagai berikut:

1. Model *deep learning* untuk memprediksi tingkat kesegaran daging ikan dapat memberikan *insight* ataupun rekomendasi mutu terhadap ikan yang hendak dikonsumsi.
2. Penggunaan YOLOv4 dan YOLOv4-Tiny, menunjukkan kemampuan yang baik dalam mendeteksi Objek. Didasarkan pada hasil mAP yang tinggi.
3. Pemanfaatan *smartphone* sebagai alat mendeteksi tingkat kesegaran daging ikan yang mudah digunakan dan dapat dioperasikan secara *realtime*, menjadi solusi yang efektif dan efisien.
4. Model YOLOv4-Tiny, memberikan opsi pada pengguna yang memiliki keterbatasan kapasitas *memory internal* untuk tetap dapat melakukan *assesment* kesegaran daging ikan.

2.3 Konsep Pengetahuan Model Yang Digunakan

Guna menilai tingkat kesegaran dari daging ikan, penelitian ini menggunakan model *deep learning* sebagai algoritma mendeteksi objek. Model yang digunakan pada penelitian ini adalah YOLO dengan arsitektur YOLOv4 dan YOLOv4-Tiny. YOLO merupakan metode pada *object detection* yang mendifinisikan *object detection* sebagai masalah regresi dan memiliki keseimbangan antara kecepatan dan akurasi. Pada prosesnya, YOLO membagi sebuah gambar ke dalam beberapa *Region* yang kemudian memprediksi kotak pembatas dari object yang dideteksi (*bounding box*), *probabilities*, dan *conditional class probabilities* (Redmon et al., 2016). Gambaran dari Detection *Pipelines* YOLO ditampilkan pada Gambar 2.1 (Redmon et al., 2016).



Gambar 2.1 *Detection Pipelines* pada model YOLO

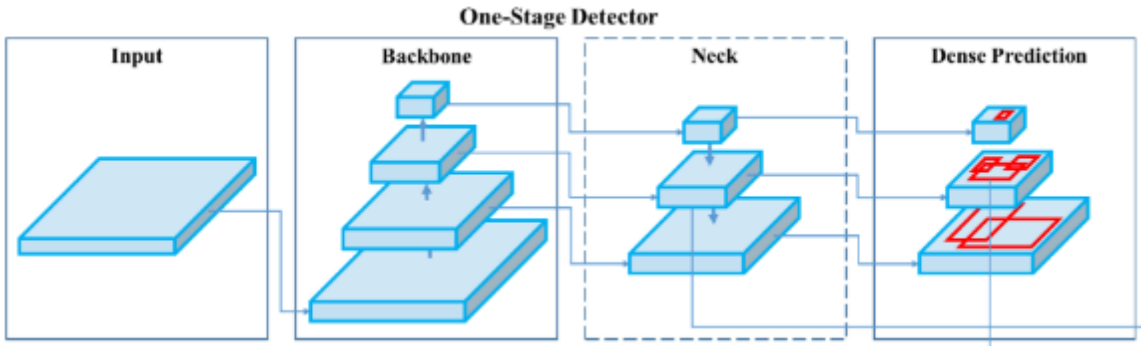
Dari sisi performa, YOLO memiliki keunggulan dibandingkan *Single Shot Detection* (SSD) dan Faster R-CNN (*Region-based Convolutional Network*) dalam menganilisis gambar pada dataset Microsoft COCO (*Common Object in Context*) (Srivastava et al., 2021) dan dalam mengenali tipe dari kendaraan (J. A. Kim et al., 2020). Dalam konteks sebagai model yang dapat dioperasikan pada *device* dengan *resource* yang minim seperti *smartphone*, YOLO merupakan model tercepat dan relatif akurat dalam melakukan deteksi objek dibandingkan SSD (VGG-300), Faster R-CNN dan R-FCN (*Region-based Fully Convolutional Network*) (C. E. Kim et al., 2019).

Pembuatan model YOLOv4 dan YOLOv4-Tiny menggunakan konsep serta rancangan yang telah didefinisikan pada akun github resmi milik pengembang dari arsitektur YOLOv4 dan YOLOv4-Tiny. proses *setup*, *training*, *testing*, *serta converting* model ke dalam TFLite dijabarkan pada akun github: https://github.com/AlexeyAB/darknet.

2.3.1 YOLOv4

Arsitektur YOLOv4, terdiri dari tiga bagian, yaitu *backbone*, *neck*, dan *head* (Bochkovskiy et al., 2020). Gambaran dari arsitektur YOLOv4 dapat dilihat pada Gambar 2.2. Bagian *backbone* adalah bagian yang berlaku sebagai detektor. Pada bagian ini jaringan yang digunakan adalah *Cross Stage Partial Darknet-53* (CSPDarknet53). Bagian selanjutnya adalah *neck*. Pada bagian *neck*, terdapat modul *Spatial Pyramid Aggregation Network*

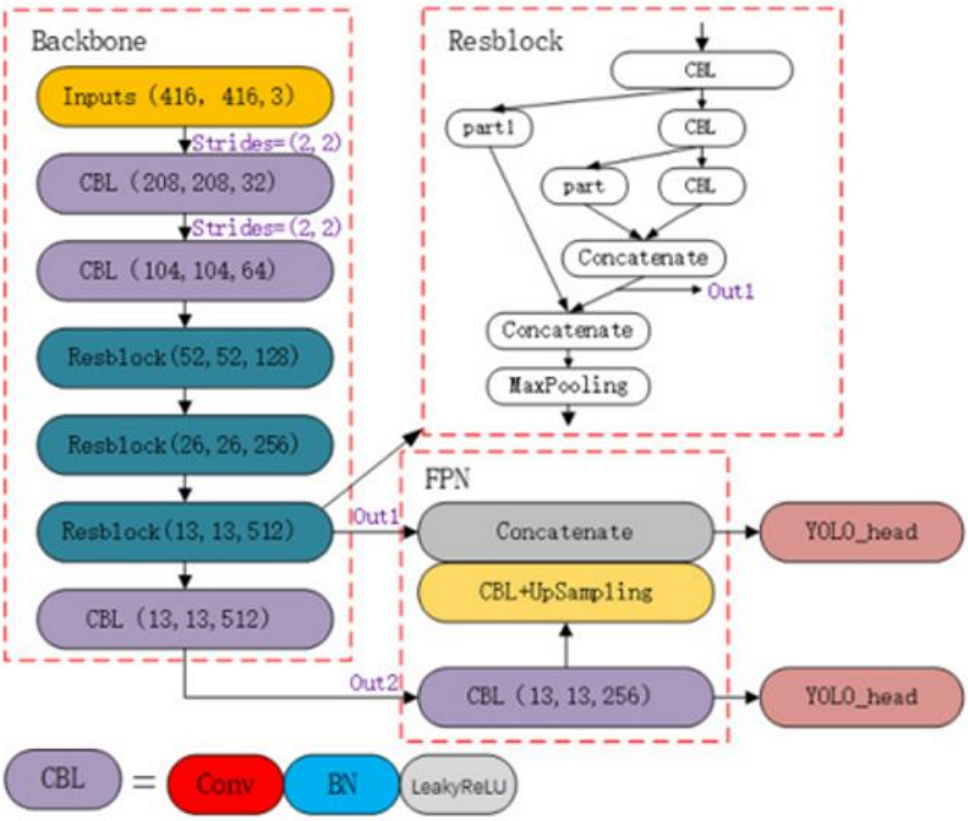
(PANet) yang ditugaskan dalam melakukan agregasi parameter (*parameter aggregation*) dari tingkatan *backbone* yang berbeda terhadap tingkatan *detector* yang lain. Bagian terakhir adalah *head*. Bagian ini menggunakan head dari YOLOv3 (*anchor based*) sebagai *dense prediction*-nya (Bochkovskiy et al., 2020).



Gambar 2.2 Arsitektur YOLOv4

2.3.2 YOLOv4-Tiny

YOLOv4-Tiny merupakan versi sederhana dari YOLOv4, memiliki kecepatan deteksi lebih cepat sembari menjamin akurasi jaringan dan dapat melakukan deteksi secara *realtime* pada perangkat dengan daya komputasi yang rendah (Jiang et al., n.d.). Arsitektur YOLOv4-Tiny, menggunakan CSPDarknet53-Tiny sebagai *backbone* dan FPN (*Feature Pyramid Network*) untuk meningkatkan fitur ekstraksi. Selain itu, terdapat juga *feature layers* dengan skala 13x13 dan 26x26 untuk memprediksi regresi dari suatu gambar. Untuk fungsi aktivasi yang digunakan adalah LeakyReLU. Ilustrasi dari YOLOv4-Tiny (Jiang et al., n.d.) ditunjukkan pada Gambar 2.3.



Gambar 2.3 Arsitektur YOLOv4-Tiny

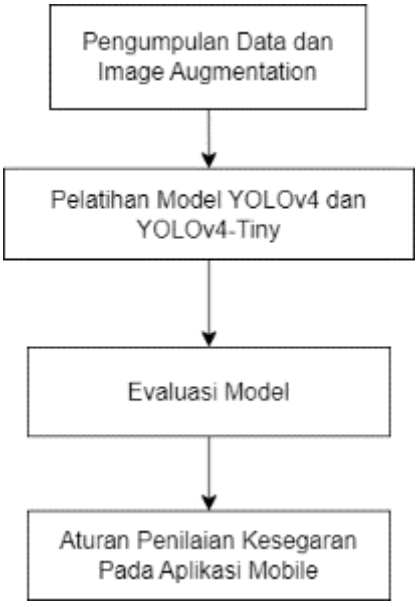
2.3.3 Implementasi Model pada Aplikasi

Pada tahap ini, penulis akan mengimplementasikan model YOLOv4 dan YOLOv4-Tiny yang telah dibuat ke dalam perangkat smartphone berbasis *android*. Pertama-tama Model yang dihasilkan dari proses training akan dikonversi ke dalam bentuk TFLite dengan menggunakan *Quantization Mode Floating Point 16* (FP16) dan *treshold* sebesar 0.7. Setelah model selesai dikonversi ke dalam bentuk TFLite, model akan diimplementasikan pada aplikasi *mobile* untuk dapat menilai tingkat kesegaran daging ikan secara *realtime* ataupun melalui sebuah gambar.

BAB 3
Metodologi

3.1 Tahapan Penelitian

Tahapan penelitian ini dimulai dengan melakukan studi literatur, mengumpulkan ikan, mengambil gambar ikan, dan melakukan *image augmetation* pada gambar ikan. Seluruh gambar selanjutnya dilabeli dengan keterangan ikan, kondisi kulit dan mata ikan. Total label yang digunakan sebanyak tujuh, yaitu ikan, mata segar, mata sedang, mata busuk, kulit segar, kulit sedang, dan kulit busuk. Pelatihan model menggunakan YOLOv4 dan YOLOv4-Tiny untuk melakukan deteksi objek (Bochkovskiy et al., 2020) yang berupa ikan, kondisi dari mata, dan kulit ikan. Bobot dari hasil pelatihan model akan dikonversi ke dalam bentuk TFLite, dan diimplementasikan bersama dengan *rule* yang dikonsultasikan dengan pakar untuk menilai tingkat kesegaran daging ikan. Hasil akhir penelitian ini adalah sebuah aplikasi untuk mendeteksi tingkat kesegaran daging ikan yang berjalan pada *platform mobile* berbasis *android*. Bagan alir penelitan ini ditampilkan pada Gambar 3.1



Gambar 3.1 Bagan Alir Penelitian

3.1.1 Pengumpulan Data dan Image Augmentation

Proses pengumpulan dan pelabelan foto ikan dilakukan dengan seorang pakar yang pernah memiliki pengalaman sebagai seorang nelayan dan pemasok ikan pada PT. Sari Laut Jaya Food Products di kabupaten Banyuwangi selama lebih dari 25 tahun. Berdasarkan saran dari pakar, delapan jenis ikan dengan kondisi segar dikumpulkan dari saudagar ikan yang

menjadi penadah pertama dari para nelayan di pelabuhan Muncar, kecamatan Muncar, kabupaten Banyuwangi. Ikan yang digunakan pada penelitian ini mencangkup ikan Tongkol Deho (*Euthynnus Affinis*), ikan Manglah (*Priacanthus Tayenus*), ikan Solok (*Rastrelliger Brachysoma*), ikan *Mackerel* (*Scomber Australasicus*), ikan Kuwe Lilin (*Caranx Melanophygus*), ikan Teribang (*Nemipterus Virgatus*), ikan Banyar (*Restrelliger Kanagurta*), dan ikan Kolong (*Atule mate*). Jumlah seluruh ikan yang digunakan pada penelitian ini sebanyak 93 ekor dengan pembagian dua belas sampel untuk ikan Tongkol, Teribang, Solok, Kuwe Lilin, dan *Mackerel*. Sepuluh Sampel untuk ikan Banyar dan Kolong. Terakhir, tiga belas sampel untuk ikan Teribang. Tampilan dari ikan Manglah diperlihatkan pada Gambar 3.2 (a), ikan Tongkol Deho pada Gambar 3.2 (b), ikan Kuwe Lilin 3.2 (c), ikan Banyar pada Gambar 3.2 (d), ikan Kolong Gambar 3.2 (e), ikan Mackerel pada Gambar 3.2 (f), ikan Teribang pada Gambar 3.2 (g), dan ikan Solok pada Gambar 3.2 (h).

Gambar 3.2 *Dataset*

Selama proses pengambilan gambar, ikan-ikan tersebut diletakkan pada kotak *styrofoam* berdasarkan jenisnya. Setiap kotak *styrofoam*, kemudian diisi es yang sudah dihancurkan dan setiap ikan suhunya dijaga pada kisaran lima derajat selsius. Penggantian es dilakukan setiap pagi (setelah pengambilan gambar) dan malam hari. Tempat

pengambilan gambar ikan yang digunakan, terinspirasi dari kotak yang dibuat oleh Taheri dkk (2020), supaya jarak antara kamera dengan objek selalu sama. Kotak yang digunakan pada penelitian ini memiliki beberapa penyesuaian pada ukuran dan bahannya. Tampilan depan dari bentuk kotak yang digunakan dalam proses pengambilan gambar dapat dilihat pada Gambar 3.3 (A). Kami meletakkan kamera pada lapisan atas kotak. Jarak antara kamera dan ikan sebesar tiga puluh sentimeter dengan lapisan untuk *background* ikan berwarna hijau. Untuk pencahayaan, diberikan secara terpisah dari luar kotak dengan meletakkan lampu senter pada bagian depan kotak. Pencahayaan ini diperlukan agar kilat warna dari ikan nampak. Tampilan dari tempat device kamera yang ada pada lapisan atas serta penempatan senter untuk pencahayaan, ditampilkan pada Gambar 3.3 (B). Spesifikasi dari *device* dan konfigurasi kamera yang digunakan dalam melakukan pengambilan gambar dapat dilihat pada Tabel 3.1

Tabel 3.1 Spesifikasi *Device* Untuk Pemotretan

Nama Variable	**Nilai**
Device	Samsung S10 Lite
Image Size	1800 x 4000 *pixels*
Zoom	*No Zooms*
Flash Mode	*Off*
Sensitivity	*Auto*
Image Type	JPEG

Gambar 3.3 Tampilan Kotak Pengambilan Gambar Ikan pada Bagian Depan (A) dan Bagian Atas (B)

Lama waktu menyimpan ikan pada penelitian ini mengikuti kajian yang dilakukan oleh Taheri dkk (2020) dan Aziz (2021) yang dilakukan selama empat belas hari. Masing-masing ikan yang disimpan, akan diambil gambarnya pada setiap sisi. Penelitian ini mengunakan tiga tingkatan untuk menilai kesegaran dari daging, mata, dan kulit ikan. Tiga tingkatan tersebut adalah segar (*Fresh*), sedang (*Medium*), dan busuk (*Spoiled*) (Moon et al., 2020; Saputra et al., 2022). Label yang akan dimodelkan sebanyak tujuh label yang mencangkup *Fish*, *Fresh* Eye, *Medium Eyes*, *Spoiled Eyes*, *Fresh Skin*, *Medium Skin*, Dan *Spoiled Skin*. Pelabelan *dataset* dilakukan bersama pakar setelah semua ikan selesai diambil gambarnya. Foto-foto ikan tersebut diperlihatkan pada pakar melalui tampilan layar pada laptop. Tujuan proses pelabelan *dataset* tidak dilakukan saat pengambilan gambar adalah, agar sipakar hanya menggunakan proyeksi visual saja tanpa menggunakan indera pengrasa yang lain (seperti penciuman dan indera pengrasa pada sentuhan). Gambar 3.4 memperlihatkan contoh data dari tampilan ikan Mackerel dalam kondisi segar (a), sedang (b), dan busuk (c). Gambar 3.5 menampilkan total sebaran data dari setiap *level* kesegaran mata dan kulit ikan pada *dataset*.

Gambar 3.4 Tampilan Ikan Segar (a), Ikan Kondisi Sedang (b), dan Busuk (c)

Class Object	Ikan Banyar	Ikan Kolong	Ikan Kuwe Lilin	Ikan Mackarel	Ikan Solok	Ikan Tongkol	Ikan Teribang	Ikan Manglah	Total
Fresh Eyes	56	88	150	119	83	131	59	129	815 Mata Ikan
Fresh Skin	143	197	237	183	153	144	124	176	1357 Kulit Ikan
Normal Eyes	139	83	83	115	76	100	88	110	794 Mata Ikan
Normal Skin	63	30	99	49	94	130	89	149	703 Kulit Ikan
Spoil Eyes	58	102	133	109	188	109	191	125	1015 Mata Ikan
Spoil Skin	47	46	30	111	100	66	125	39	564 Kulit Ikan

Gambar 3.5 Total Data Tingkat Kesegaran Mata dan Kulit Ikan Pada *Dataset*.

Jumlah gambar ikan yang diperoleh dari hasil pemotretan adalah sebanyak 2604 gambar. Setelah proses pengambilan gambar selesai, 1302 gambar dipilih untuk diterapkan *image augmentation* agar menghasilkan gambar baru dengan posisi objek yang berbeda. 1302 gambar yang dipilih, berdasarkan sebaran data dari kondisi mata dan kulit ikan pada setiap gambar dari masing-masing jenis ikan. Gambar 3.6 menampilkan total gambar ikan dengan kondisi dari mata dan kulit ikan, untuk setiap jenis ikan yang diperoleh selama proses pengambilan gambar. Masing-masing total gambar dari setiap jenis ikan, dibagi dua dan hasilnya kami gunakan untuk menentukan seberapa banyak *image* yang diaugmentasi untuk

jenis ikan tersebut. Misalnya, ikan Banyar dengan kondisi mata segar dan kulit segar, memiliki total gambar sebanyak 56. Maka gambar baru yang harus dihasilkan dari *image augmentation* adalah sebanyak 28. Total gambar yang akan dihasilkan selama proses *image augmentation* dari setiap jenis ikan dengan setiap kondisi mata dan kulitnya, dtaampilkan pada Gambar 3.7.

Kondisi Mata - Kulit ikan	Ikan Banyar	Ikan Kolong	Ikan Kuwe Lilin	Ikan Mackarel	Ikan Solok	Ikan Tongkol	Ikan Teribang	Ikan Manglah	Total
Fresh - Fresh	56	88	150	118	83	118	55	125	793
Fresh - Normal	0	0	0	0	0	7	4	0	11
Fresh - Spoil	0	0	0	0	0	0	0	0	0
Normal - Fresh	87	79	72	64	69	20	62	46	499
Normal - Normal	42	1	9	38	7	72	26	62	257
Normal - Spoil	10	0	0	13	0	7	0	2	32
Spoil - Fresh	0	27	13	0	1	0	7	1	49
Spoil - Normal	21	29	90	11	87	50	59	87	434
Spoil - Spoil	37	46	30	98	100	56	125	37	529
Total	253	270	364	342	347	330	338	360	**2604**

Gambar 3.6 Total Gambar Setiap Jenis Ikan Dengan Setiap Kondisi Mata dan Kulit Ikan.

Total Data Tambahan Setiap Ikan Dari Hasil Image Augmentation									
Kondisi Mata - Kulit ikan	Ikan Banyar	Ikan Kolong	Ikan Kuwe Lilin	Ikan Mackarel	Ikan Solok	Ikan Tongkol	Ikan Teribang	Ikan Manglah	Total
Fresh - Fresh	28	44	75	59	42	59	28	63	397
Fresh - Normal	0	0	0	0	0	4	2	0	6
Fresh - Spoil	0	0	0	0	0	0	0	0	0
Normal - Fresh	44	40	36	32	35	10	31	23	250
Normal - Normal	21	1	5	19	4	36	13	31	129
Normal - Spoil	5	0	0	7	0	4	0	1	16
Spoil - Fresh	0	14	7	0	1	0	4	1	25
Spoil - Normal	11	15	45	6	44	25	30	44	217
Spoil - Spoil	19	23	15	49	50	28	63	19	265
Total	127	135	182	171	174	165	169	180	**1302**

Gambar 3.7 Total Gambar Yang Akan Diaugmentasi Dari Setiap Jenis Ikan Untuk Setiap Kondisi Mata dan Kulit Ikan.

Adapun manipulasi citra digital yang diterapkan adalah yang tergolong ke dalam *geometric augmentation*, yaitu *rotation*, *vertical flip*, *horizontal flip*, *height shift*, dan *width shift* (Shorten & Khoshgoftaar, 2019). Selain kelima manipulasi tersebut, juga diterapkan *zoom in* dan *zoom out* agar citra lebih bervariasi. Setelah menerapkan *image augmentation*, jumlah gambar bertambah menjadi 4138 gambar. Contoh beberapa tampilan hasil dari *image augmentation* ditampilkan pada gambar 3.8.

Gambar 3.8 *Image* Hasil *Image Augmentation*.

Proses terakhir yang dilakukan pada *dataset*, adalah melabeli bagian dari ikan, mata dan kulit ikan berdasarkan kelasnya sesuai dengan arahan dari pakar dan referensi Lougovois dan Kyrana (2005). Guna mempermudah proses pelabelan, penelitian ini menggunakan *tools* yang bernama LabelImg (https://github.com/byhqsr/tzutalin-labelImg).

3.1.2 Pelatihan Model YOLOv4 dan YOLOv4-Tiny

Proses *training* penelitian ini menggunakan Darknet sebagai *deep learning framework* untuk menjalankan YOLOv4 dan Google Colab. Untuk mendapatkan spesifikasi dari GPU (*Graphics Processing Unit*) yang lebih efisien dalam melakukan *training*, paket Google Colab yang digunakan adalah Google Colab Pro. Kapasitas GPU yang tersedia pada peket tersebut adalah NVIDIA K80, P100, dan T4 (diatur otomatis). Kapasitas RAM sebesar 24 GB. Limit Sessions yang diberikan untuk menjalankan proses *training* selama 24 jam. Dan untuk CPUs sebanyak 2 vCPU. Selama proses Training, delapan *hyperparameter* diatur untuk mendapatkan model yang optimal. Tabel 3.2 menampilkan konfigurasi dari setiap Hyperparemeter.

Tabel 3.2. Konfigurasi Hyper Parameter

Properties	YOLOv4	YOLOv4-Tiny
Pretrained Model	yolov4.conv.137	yolov4-tiny.conv.29
Dimension	416x416	416x416
Data Augmentation	*Yes*	*Yes*
Iteration	14000	14000
Learning Rate	0.001	0.00261
Momentum	0.949	0.9
Decays	0.0005	0.0005
Batch Size	64	64

Model pada penelitian ini akan menggunakan metode *transfer learning*. *Transfer learning* merupakan proses *transfer* informasi dari suatu model yang sebelumnya telah dilatih pada kumpulan data dalam jumlah yang besar. Hasilnya berupa bobot-bobot dari model yang telah dilatih (*pre-trained model*) . Pre-Trained model yang digunakan pada penelitian ini merupakan *Prep-trained Model* yang dilatihkan pada dataset MSCOCO dan telah dilatih hingga 137 lapisan jaringan konvolusinya.

3.1.3 Evaluasi Model

Beberapa indikator yang digunakan untuk mengukur kinerja dari model meliputi pengukuran relevansi hasil deteksi (*Precision*), pengukuran atas banyaknya hasil deteksi yang benar-benar relevan yang dikembalikan (*Recall*), *F_1 Score* (mengukur perpaduan yang optimal antara *Precision* (P) dan *Recall* (R)), *Average Precision* (AP), dan *Mean Average Precision* (mAP) (Li et al., 2022). *Overlapping* diukur dari suatu *region* pada dua gambar diukur menggunakan *Intersection Over Union* (IoU) dengan nilai 0,5. Bila skor IoU dari hasil deteksi lebih besar dari 0.5, maka model sudah mampu mendeteksi dengan baik. Begitupun sebaliknya. Persamaan 1 menampilkan rumus dari IoU. Dimana A adalah prediksi dari *bounding box*, dan B merepresentasikan *bounding box* yang sebenarnya.

$$IoU = \left|\frac{A \cap B}{A \cup B}\right| \quad (1)$$

Rumus yang digunakan oleh P, ditampilkan pada Persamaan 2. Untuk R, ditampilkan pada Persamaan 3. Untuk *F_1 Score*, pada Persamaan 4. Dimisalkan, mata ikan dilabeli sebagai A dan dideteksi sebagai kelas A pada suatu *image*, maka hasilnya akan dinilai sebagai *True Positive* (TP). Tapi jika A, dideteksi sebagai kelas yang lain, maka dianggap

False Negative (FN). Dan jika pada suatu *image* kelas A tidak ada tetapi terdeteksi ada, maka dianggap *False Positive* (FP). TP, FP, dan FN adalah variabel-variabel yang digunakan untuk menghitung P dan R.

$$P = \frac{TP}{TP + FP} \tag{2}$$

$$\mathrm{R} = \frac{TP}{TP + FN} \tag{3}$$

$$F_1 \text{ Score} = \frac{2P \times R}{P + R} \tag{4}$$

Fungsi dari AP adalah mengevaluasi perfoma model dalam mendeteksi setiap *class* atau kategori objek. Persamaan yang gunakan oleh AP ditampilkan pada Persamaan 5 dan mAP pada Persamaan 6. Semakin tinggi nilai yang dihasilkan oleh AP dan mAP, maka semakin baik hasil deteksi dari model terhadap suatu objek. Nilai k pada AP dan mAP mewakili kelas-kelas dari objek yang dideteksi.

$$AP_k = \int_0^1 P_k(R_k) dR_k \tag{5}$$

$$mAP = \frac{1}{k} \sum_{i=1}^{i} AP_i \tag{6}$$

3.1.4 Aturan Penilaian Kesegaran pada Aplikasi Mobile

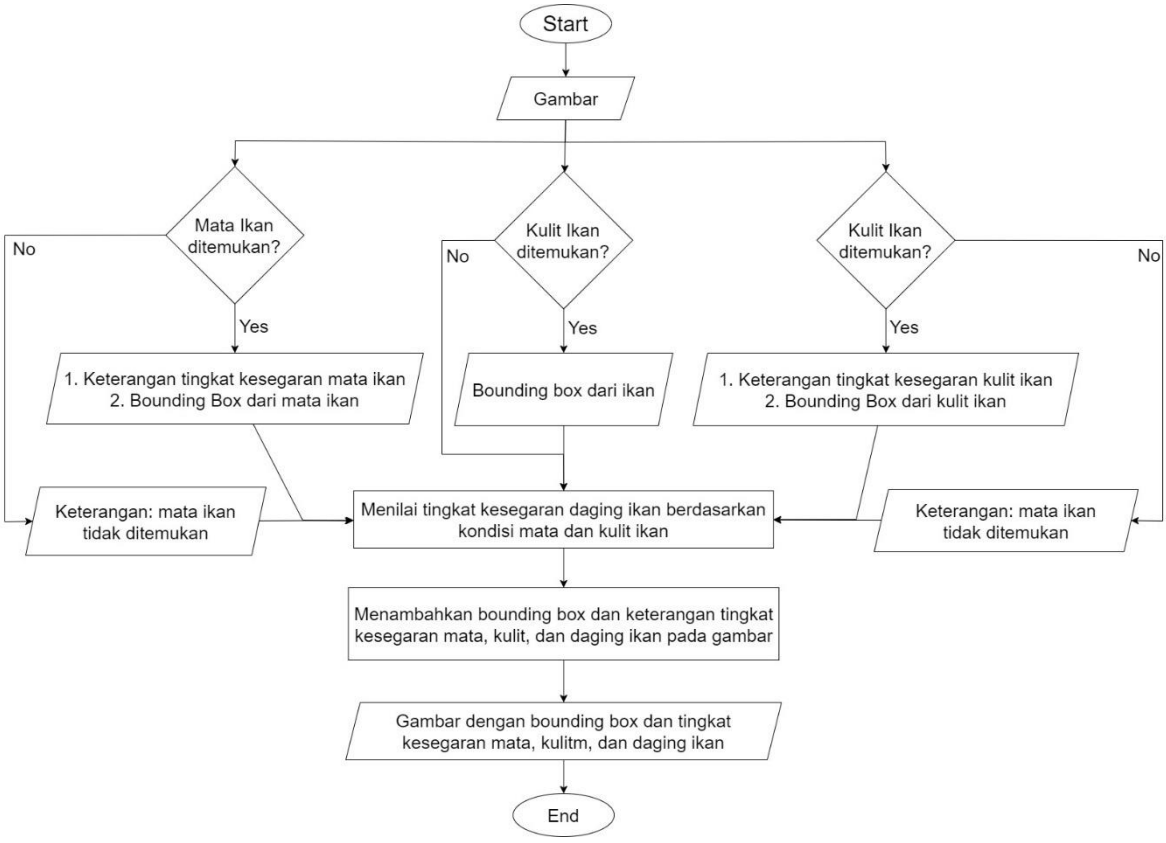
Model yang dihasilkan, adalah model yang dapat mendeteksi tingkat kesegaran dari kulit dan mata ikan. Sehingga, Guna memberikan kesimpulan akhir dari tingkat kesegaran daging ikan, sebuah *rule* dibuat bersama pakar berdasarkan nilai yang dihasilkan dari model terhadap mata dan kulit ikan. *Flowchart* aturan kesegaran daging ikan yang diterapkan ditampilkan pada Gambar 3.9. *Rule* yang ditetapkan oleh pakar dan referensi dari penelitian Lougovis (2005) untuk menilai kondisi daging ikan dapat dilihat pada Tabel 3.3.

Tabel 3.3 *Rule* Untuk Menentukan Tingkat Kesegaran Daging Ikan

Kondisi Mata	Kondisi Kulit	Tingkat Kesegaran
Segar	Segar	Segar
Segar	Sedang	Sedang
Segar	Busuk	Sedang
Sedang	Segar	Sedang
Sedang	Sedang	Sedang

Tabel 3.3 Lanjutan

Sedang	Busuk	Busuk
Busuk	Segar	Sedang
Busuk	Sedang	Sedang
Busuk	Busuk	Busuk



Gambar 3.9 *Flowchart Rule* Untuk Menentukan Tingkat Kesegaran Daging Ikan.

3.2 Analisis Kebutuhan

Analisis kebutuhan adalah metode untuk memperoleh detail dan spesifikasi peralatan yang berfokus pada perangkat lunak atau *software* yang akan dikembangkan oleh penulis. Tujuan analisis kebutuhan adalah untuk menentukan kebutuhan sistem dalam mendeteksi tingkat kesegaran daging ikan. Yang mencangkup kebutuhan *software, input,* dan *output*.

3.2.1 Analisis Kebutuhan Masukan (*Input*)

Kebutuhan input untuk proses pelatihan model mendeteksi tingkat kesegaran daging ikan adalah *file* gambar, yang dikumpulkan secara langsung dengan melakukan pemotretan pada tampilan ikan segar dan busuk.

3.2.2 Analisis Kebutuhan Perangkat Lunak (*Software*)

Kebutuhan *software* untuk melakukan pelabelan pada setiap image menggunakan *tools* LableImg. Untuk proses pelatihan, pengujian dan konversi model ke dalam TFLite adalah *software* yang digunakan adalah Google Colab. Sedangkan untuk mengembangkan aplikasi *mobile* berbasis *android*, penulis menggunakan *Software Android Studio*.

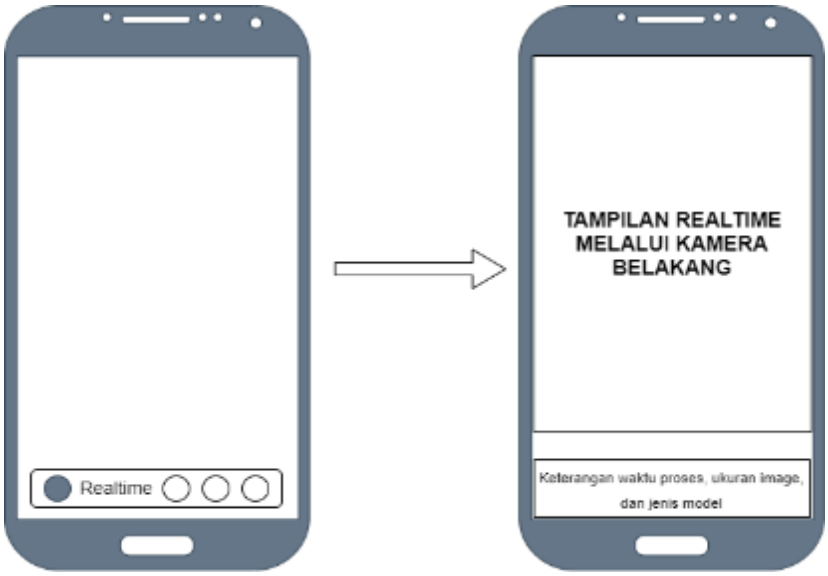
3.2.3 Analisis Kebutuhan Keluaran (*Output*)

Kebutuhan *output* adalah berupa *image* dengan *bounding box* tingkat kesegaran dari mata ikan, kulit ikan, dan daging ikan.

3.3 Perancangan *User Interface* Aplikasi

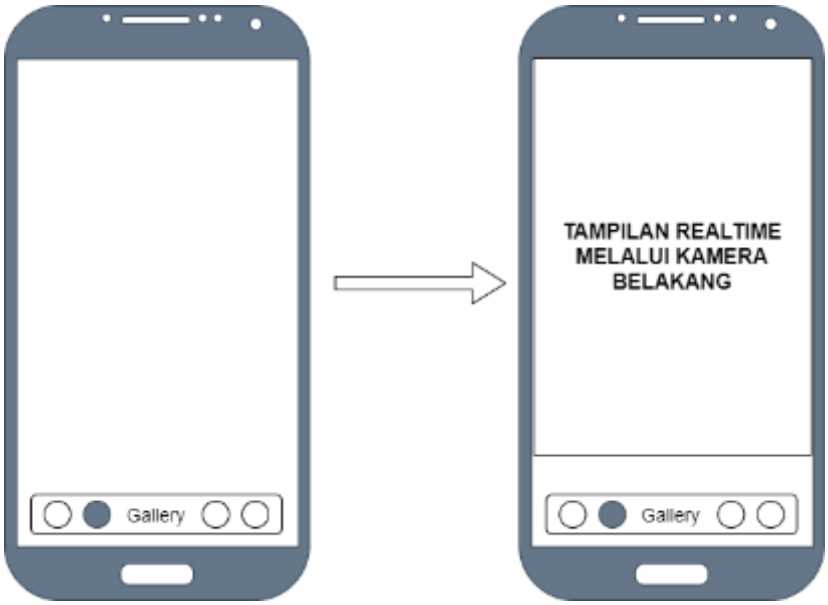
Design halaman aplikasi deteksi kesegaran daging ikan, terdiri atas empat menu utama. Menu pertama, berfungsi untuk melakukan proses deteksi kesegaran daging ikan secara *Realtime* melalui kamera *smartphone*. Menu kedua berfungsi untuk melakukan deteksi kesegaran daging ikan dengan melakukan *import* gambar dari galeri. Menu ketiga adalah informasi dari pengembang dan jenis model yang digunakan. Menu terkahir, menampilkan keterangan singkat yang menunjukkan fungsi dari setiap menu.

Menu *Realtime*, akan membuka fungsi kamera pada bagian belakang dan menampilkan proses penilaian tingkat kesegaran daging ikan secara *realtime*. Jika kamera mendeteksi adanya ikan, kulit ikan, dan mata ikan, maka pada tampilan dalam menu *Realtime* ini akan menampilkan secara langsung *bounding box* serta tingkat kesegaran dari masing-masing objek yang terdeteksi. Selain itu, pada tampilan menu *Realtime*, juga akan ditampilkan berapa lama waktu proses, dimensi gambar yang diproses, serta versi model yang dipakai. Tampilan menu *Realtime* dan tampilan halaman yang dimunculkan dapat dilihat pada Gambar 3.10.



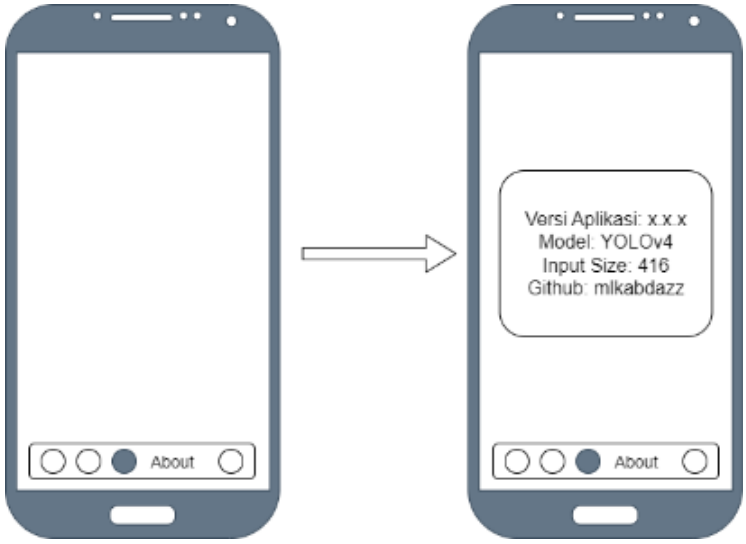
Gambar 3.10 Desain Menu *Realtime*

Menu kedua adalah menu yang memungkinkan pengguna melakukan deteksi dan penilaian tingkat kesegaran daging ikan melalui gambar yang dipilih dari *gallery*. Menu ini berfungsi, jika *user* akan melakukan proses penilaian daging ikan dari gambar-gambar lama yang dimiliki atau yang sebelumnya telah disimpan. Gambar yang dipilih, nantinya akan menampilkan masing-masing *bounding box* dari setiap objek yang terdeteksi dan disimpan secara temporer. *Design* dari tampilan menu *import* atau pilih gambar dari *gallery*, ditampilkan pada Gambar 3.11.



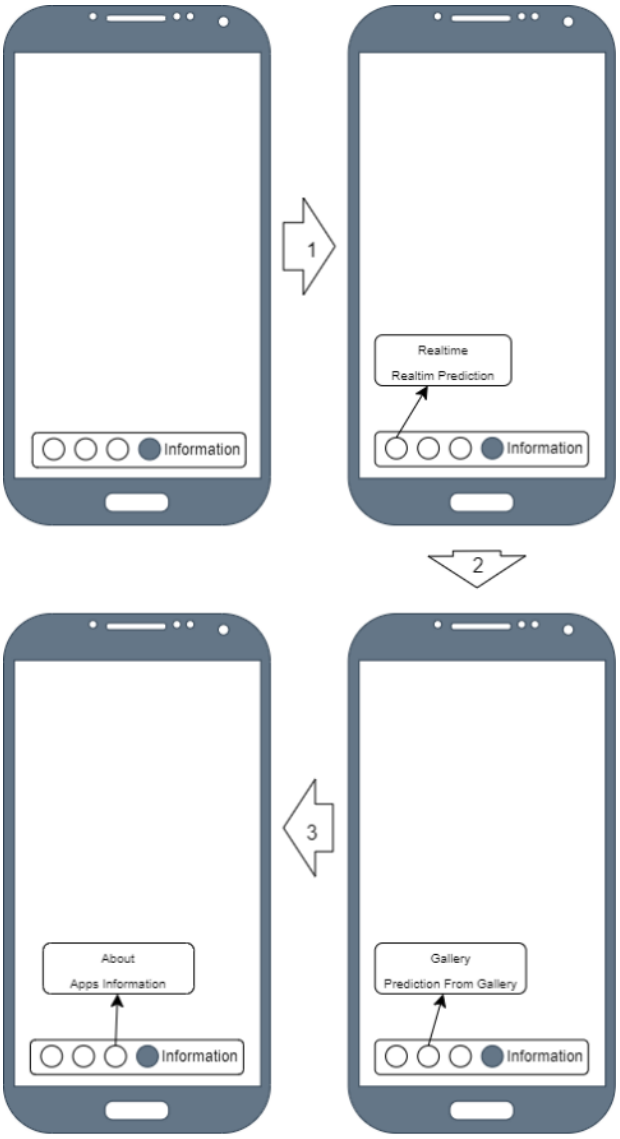
Gambar 3.11 Tampilan Menu *Import* Gambar Dari Galeri

Menu ketiga adalah menu yang memberikan informasi aplikasi dan informasi pengembang yang ditampilkan dalam bentuk *popup*. Informasi yang ditampilkan mencangkup versi aplikasi, model yang digunakan, ukuran dimensi dari *image* yang diproses, serta akun github dari pengembang. Gambar dari tampilan menu ketiga atau menu *about*, ditampilkan pada Gambar 3.12.



Gambar 3.12 Tampilan Menu Tentang *Developer*

Menu terakhir merupakan menu Help. Menu yang akan menunjukkan informasi dari setiap menu. Jika pengguna memilih menu ini, maka akan muncul keterangan singkat dari setiap menu yang tersedia. Tujuan dari menu ini adalah memberikan informasi fungsi dari setiap menu yang disediakan. Harapan penulis, pengguna dapat memahami fungsi dari setiap menu yang ada dengan baik melalui menu keempat atau menu Help. Tampilan dari menu tersebut ditunjukkan pada Gambar 3.13.



Gambar 3.13 Tampilan Menu Help

BAB 4
Hasil dan Pembahasan

Guna menilai tingkat kesegaran daging ikan, penelitian ini memanfaat dua parameter dari tampilan daging ikan. Yaitu dari tampilan mata dan tampilan kulit. kedua parameter tersebut dikombinasikan untuk menentukan tingkat kesegaran daging ikan. Dengan memanfaatkan model *deep learning*, penulis melakukan deteksi dan klasifikasi pada tingkat kesegaran masing-masing parameter. Selain itu, penulis juga memanfaatkan model *deep learning* untuk mendeteksi ikan. Deteksi bentuk dari objek ikan bertujuan untuk dapat mengimplementasikan *rule* yang telah dibuat bersama pakar, sebagaimana yang ditampilkan pada Gambar 3.9.

Delapan jenis ikan yang telah dikumpulkan, masing-masing foto dari sisinya setiap hari selama empat belas hari, dan menghasilkan total foto di awal sebanyak 2604. Kemudian 1302 diambil secara acak dari setiap jenis ikan untuk meningkatkan jumlah data menggunakan *image augmentation*. Total *dataset* keseluruhan pada penelitian ini sebesar 4138. Persentase pembagian *dataset training* dan *testing* diatur secara acak selama proses *setup* dan *preparation*. Presentase yang digunakan adal 75% untuk data *training*, dan 25% untuk data *testing* dan dibagi secara acak (Abd Aziz et al., 2021).

Model *deep learning* memiliki keunggulan untuk mempelajari dan mengenali *object* dalam jumlah besar jika dibandingkan pada model *machine learning*. Untuk mencapai tujuan dari penelitian ini, penulis menggunakan model YOLOv4 dan YOLOv4-Tiny. Dimana secara performa dan implementasinya terhadap *device* dengan spesifikasi yang lebih rendah, mampu berjalan dengan baik. Penulis juga memanfaatkan pendekatan *transfer learning* saat melakukan pelatihan pada model. Pendekatan ini memungkinkan model untuk memanfaatkan pengetahuan yang sudah ada dalam model yang telah dilatih sebelumnya, sehingga dapat meningkatkan performa dan efisiensi dalam proses pelatihan terhadap *dataset* yang baru. *Pre-trained* model yang digunakan adalah model yang telah dilatih pada dataset MSCOCO (Bochkovskiy et al., 2020).

Terakhir, dua model yang dihasilkan akan dikonversi ke dalam bentuk TFLite agar dapat dioperasikan pada aplikasi berbasis *mobile* yang dikembangkan oleh penulis. Nilai *quantization mode* yang digunakan saat melakukan konversi adalah FP16. Nilai *threshold* yang digunakan sebagai batas ambang *bounding box* ditampilkan pada aplikasi adalah sebesar 0.7. Apabila tidak terpenuhi, maka *bounding box* tidak akan ditampilkan. Meski ada

kemungkinan terdapat *Bounding box* yang bisa ditampilkan di bawah nilai *threshold* tersebut.

4.1 *Source Image Augmentation*

Sebelum melakukan proses *training*, penulis melakukan proses *image augmentation* pada gambar awal yang diperoleh setelah sesi pemotretan pada ikan selama empat belas hari. Sebagaimana dijelaskan sebelumnya, tujuannya adalah memperkaya variasi citra digital dari ikan dengan posisi berbeda-beda dari posisi awal. Gambar 4.1 menampilkan *source code* dari *importing library* yang diperlukan, menentukan *folder* target dari *image* yang akan diproses, dan tempat *image* yang dihasilkan setelah proses selesai. Guna mempersingkat waktu proses dan menghasilkan *output* yang maksimal, penelitian ini menggunakan *function* ImageDataGenerator milik keras.

```
from keras.preprocessing.image import ImageDataGenerator
saving_dir = 'TARGET_FOLDER_FOR_SAVE'
data_dir = 'DIRECTORY_DATA_FOR_IMPL_IMAGE_AUGMENTATION'
```

Gambar 4.1 *Source Code Import Library*.

Setelah melakukan import *function* dari *library* keras dan menentukan *input output directory* untuk *image augmentation*, selanjutnya kami mendefinisikan *function* untuk memproses masing-masing Image. Gambar 4.2 memperlihatkan penggunaan *function* ImageDataGenerator, dengan parameter sebagai berikut:

1. rotation_range. Parameter ini digunakan untuk menentukan nilai rentang derajat untuk merotasi *image* secara acak. Nilai *rotation range* yang kami gunakan adalah 45, sehingga akan mengacak jarak rotasi gambar dari 0 hingga 45 derajat.
2. zoom_range. Digunakan untuk menentukan *range* atau jarak saat melakukan *zoom in* dan *zoom out* secara acak. Nilai *zoom range* yang kami gunakan adalah 0.7, sehingga akan mengacak jarak *zoom in* dan *zoom out* dari 0 hingga 0.7 dari posisi awal citra.
3. horizontal_flip. Parameter yang digunakan untuk melakukan *horizontal flip* selama proses *image augmentation*.
4. vertical_flip. Parameter yang digunakan untukan melakukan *vertical flip* selama proses *image augmentation*.
5. height_shift_range. Parameter ini digunakan untuk menentukan nilai *range* pergeseran *image* secara *vertical* (*image* akan digeser ke atas atau ke bawah). Nilai yang kami gunakan adalah 0.6, sehingga akan mengacak jarak pergeseran gambar secara *vertical* dari 0 hingga 0.6 dari posisi awal gambar.

6. width_shift_range. Parameter ini digunakan untuk menentukan nilai *range* pergeseran *image* secara *horizontal* (*image* akan digeser ke samping kanan atau kiri). nilai yang kami gunakan adalah 0.6, sehingga akan mengacak jarak pergeseran gambar secara *horizontal* dari 0 hingga 0.6 dari posisi awal gambar.

```
datagen = ImageDataGenerator(
    rotation_range=45,
    zoom_range=0.7,
    horizontal_flip=True,
    vertical_flip=True,
    height_shift_range=0.6,
    width_shift_range=0.6
)
```

Gambar 4.2 *Source Code* untuk mendfinisikan *Function* ImageDataGenerator

Proses terakhir pada *image augmentation*, adalah menjalankan *function* yang sebelumnya didefinisikan terhadap ImageDataGenerator. *Source code* untuk menjalan *function* tersebut ditampilkan pada Gambar 4.3. Adapun parameter yang diperlukan saat menjalankan *function* ImageDataGenerator adalah:

1. data_dir. folder *Image* yang akan diproses.
2. batch_size. Parameter yang menentukan seberapa banyak *image* yang akan diolah dalam satu kali proses. agar menghemat *memory* dan menghindari *crash* saat proses, kami hanya memproses enam belas *image* dalam satu kali *loop* proses *image augmentation*.
3. target_size. Parameter yang memberikan instruksi *output size* dari setiap *image* yang akan dihasilkan dari proses *image augmentation*. Target *size* yang kami gunakan adalah ukuran awal *image* yang kami miliki, yaitu ukuran 1800 x 4000 *pixel*.
4. save_to_dir. Alamat dari *folder* tempat penyimpanan hasil proses *image augmentation.*
5. save_prefix adalah parameter yang meberikan perintah untuk menamai *output* dari *image augmentation* dengan *prefix* yang diinginkan. dalam *source code* kami, kami mengisi *Prefix*-nya dengan nama “aug”.
6. save_format adalah parameter yang betujuan untuk menyimpan *image* dengan format yang diinginkan.

```
i = 0
for batch in datagen.flow_from_directory(
    data_dir,
    batch_size=16,
    target_size=(1800, 4000),
    save_to_dir=saving_dir,
    save_prefix='aug',
    save_format='jpg'):
    i+=1
    if i > 13:
        break
```

Gambar 4.3 *Source Code* Menjalankan ImageDataGenerator.

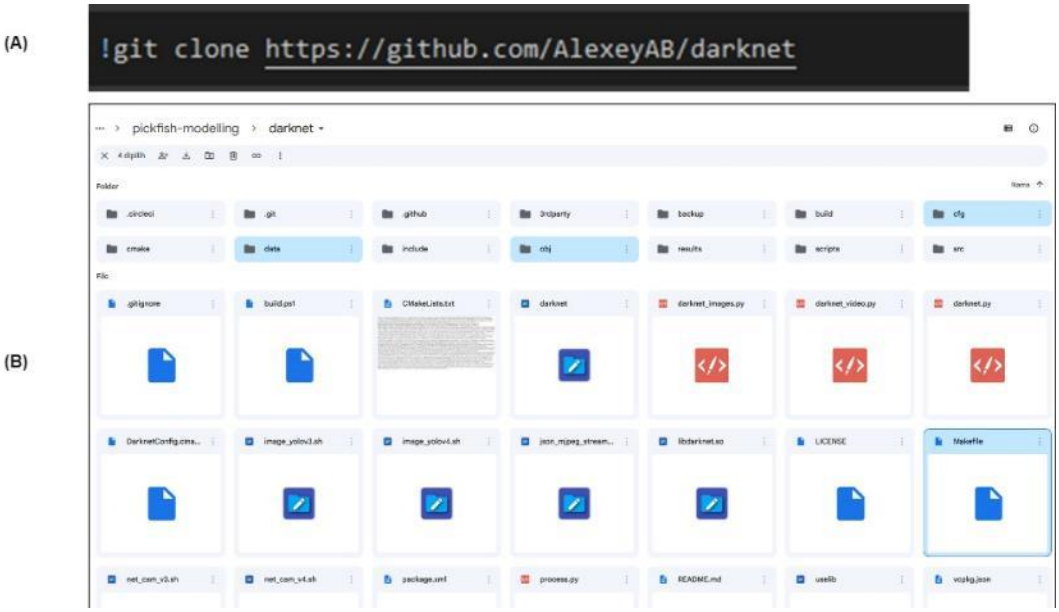
Pada *source code* Gambar 4.3, penulis melakukan *looping* sebanyak tiga belas kali dalam satu kali proses augmentasi. Tujuannya agar mempermudah dalam memilah image yang akan digunakan. Variabel i pada baris pertama dan kondisi (i > 13), penulis gunakan untuk memastikan program berjalan tiga belas kali dalam satu kali proses augmentasi gambar. Contoh dari *output* dari proses *Image Augmentation* ditampilkan pada Gambar 4.4 berikut.

Gambar 4.4 Contoh Hasil *Image Augmentation*

4.2 Source Code Training

Proses pertama untuk melakukan *training* pada dataset ikan yang kami miliki adalah melakukan *clonning repository* github milik Alexey Bochkovskiy (Bochkovskiy et al., 2020). *Repository* tersebut berisikan *code* dari model YOLO dengan *framework* YOLOv4 dan YOLOv4-Tiny. Dengan menggunakan *repository* dari Alexey Bochkovskiy, kita dapat dengan mudah menjalankan proses *training* tanpa perlu melakukan *coding* dari *scratch*.

Script untuk melakukan proses *clonning* ditampilkan pada Gambar 4.5 (A) dan isi repositori hasil *clonning* ditampilkan pada Gambar 4.5 (B).



Gambar 4.5 *Source Code Clonning* (A) dan Isi Repositori Hasil *Clonning* Github Alexey Bochokovskiy (B).

Setelah meng-*clone* repositori milik github milik Alexey Bochkovskiy, terdapat beberapa folder dan file yang harus diperhatikan di dalam repositori tersebut.

1. Folder data. folder ini digunakan untuk meletakkan data *training*, data *testing*, label, serta segala konfigurasi terkait penggunaan data.
2. File obj.data pada folder data. digunakan untuk menentukan berapa banyak *class* yang digunakan, letak data *training* dan data *testing*, letak bobot hasil *training* disimpan.
3. File obj.names pada folder data. digunakan untuk memberikan informasi label yang digunakan.
4. Folder cfg. folder ini digunakan sebagai tempat untuk meletakkan segala *pre-trained* model yang diperlukan saat menjalankan proses *training* dan *testing* YOLOv4.
5. *File* Makefile. *File* yang bertujuan untuk konfigurasi GPU (*Graphics Processing Unit*), *library* OpenCV (*Open Computer Vision*), dan lain-lain (akan dijelaskan pada paragraf selanjutnya).

Setelah proses *clonning* selesai, tahap selanjutnya adalah melakukan konfigurasi pada beberapa *file* dalam *repository* tersebut. *Script* untuk melakukan *update* konfigurasi *environment* seperti OPENCV, GPU, dan lain-lain ditampilkan pada Gambar 4.6.

```
%cd darknet_basic_data

# if not work, update manually
!sed -i 's/OPENCV=0/OPENCV=1/' Makefile
!sed -i 's/GPU=0/GPU=1/' Makefile
!sed -i 's/CUDNN=0/CUDNN=1/' Makefile
!sed -i 's/CUDNN_HALF=0/CUDNN_HALF=1/' Makefile
!sed -i 's/LIBSO=0/LIBSO=1/' Makefile
```

Gambar 4.6 Konfigurasi OpenCV, GPU, CUDNN, dan LIBSO.

Penjelasan dari *source code* yang ditampilkan pada Gambar 4.6 adalah sebagai berikut:

1. Baris pertama untuk memasuki *folder* yang sebelumnya telah di *clone*. Kami mengganti nama dari folder yang sebelumnya Darknet menjadi darknet_basic_data.
2. Baris kedua *command* atau catatan penulis.
3. Baris ketiga, untuk mengaktivasi *library* OpenCV (nilai 1 untuk mengaktifkan fitur. nilai 0 adalah untuk menonaktifkan fitur).
4. Baris keempat, untuk mengaktivasi konfigurasi GPU (nilai 1 untuk mengaktifkan fitur. nilai 0 adalah untuk menonaktifkan fitur).
5. Baris kelima, untuk mengaktivasi CUDNN (*CUDA Deep Neural Network*) (nilai 1 untuk mengaktifkan fitur. nilai 0 adalah untuk menonaktifkan fitur).
6. Baris keenam, untuk aktivasi CUDNN Half yang digunakan untuk mempercepat waktu proses 3x lebih baik (nilai 1 untuk mengaktifkan fitur. nilai 0 adalah untuk menonaktifkan fitur).
7. Baris ketujuh, untuk membuat *interface* penghubung Darknet dengan Python (nilai 1 untuk mengaktifkan fitur. nilai 0 adalah untuk menonaktifkan fitur).

Selanjutnya adalah proses untuk melakukan *build* Darknet atau mempersiapkan dan menjalankan segala *environment* yang dibutuhkan, termasuk konfigurasi yang dikonfigurasi di awal (Gambar 4.6). Perintah untuk melakukannya ditampilkan pada Gambar 4.7. Proses terakhir sebelum menjalankan perintah untuk menjalankan modelnya adalah melakukan proses *transfer learning*. kami menggunakan bobot yang sebelumnya telah dilatih terhadap data MSCOCO (Bochkovskiy et al., 2020). Perintah untuk mengambil bobot-bobot tersebut dari github milik Alexey Bochkovskiy ditampilkan pada Gambar 4.8 untuk YOLOv4 dan

Gambar 4.9 untuk YOLOv4-Tiny. Terakhir, jika semua preparation dan *environment* telah siap, maka *source code* untuk me-*running* YOLOv4 atau YOLOv4-Tiny dapat dijalankan. Perintah instruksi untuk menjalankan YOLOv4 ditampilkan pada Gambar 4.10 dan Gambar 4.11 untuk YOLOv4-Tiny

```
# Build darknet
!make
```

Gambar 4.7 *Source Code Build* Darknet.

```
# Pretrained yolo conv 137
!wget https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v3_optimal/yolov4.conv.137
```

Gambar 4.8 *Source Code* Mengambil *Pre-Trained* Model YOLOv4.

```
!wget https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v4_pre/yolov4-tiny.conv.29
```

Gambar 4.9 *Source Code* Mengambil *Pre-Trained* Model YOLOv4 Tiny.

```
!./darknet detector train \
  data/obj.data \
  cfg/yolov4-custom.cfg \
  yolov4.conv.137 -dont_show -map
```

Gambar 4.10 *Source Code* Menjalankan *Training* YOLOv4.

```
!./darknet detector train \
  data/obj.data \
  cfg/yolov4-tiny-custom-512.cfg \
  yolov4-tiny.conv.29 -dont_show -map
```

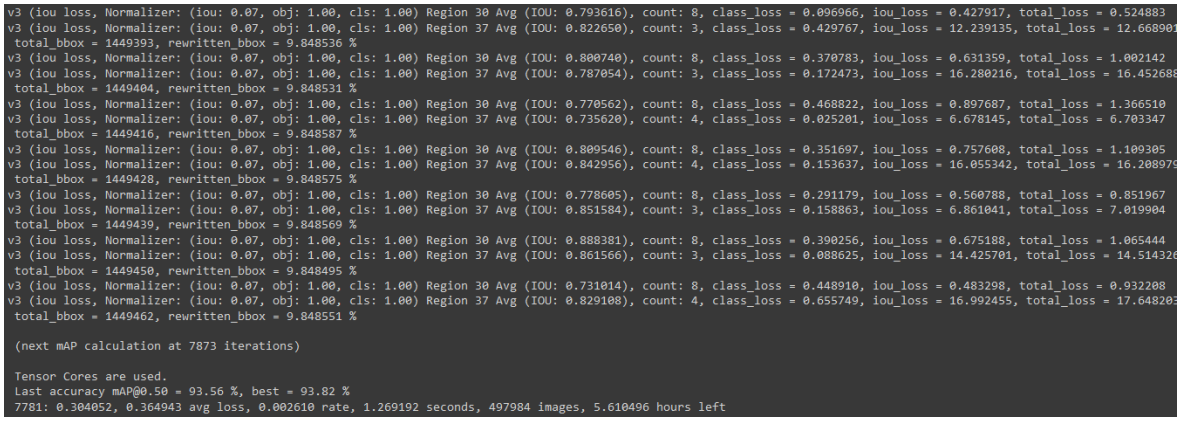
Gambar 4.11 *Source Code* Menjalankan *Training* YOLOv4-Tiny.

Penjelasan dari *source code* yang ditampilkan pada Gambar 4.10 dan Gambar 4.11 adalah sebagai berikut:

1. Baris pertama adalah perintah untuk menjalankan proses *training.*
2. Baris kedua menjelaskan letak data beserta konfigurasi data *testing* dan *training*.

3. Baris ketiga menjelaskan letak konfigurasi yang digunakan untuk YOLOv4 (Gambar 4.10) atau YOLOv4-Tiny (Gambar 4.11).
4. Baris keempat menjelaskan *pre-trained* model yang digunakan.
5. Perintah -don't_show digunakan agar program tidak menampilkan popup grafik pelatihan. Karena *source code* tersebut dijalankan pada web browser, sehingga untuk menghindari *crash* selama pelatihan hal tersebut kami hindari.
6. Perintah -map digunakan untuk menampilkan nilai mAP pada *log* selama proses *training*.

Selama proses *training*, *console log* akan secara terus-menerus menampilkan hasil mAP dari setiap gambar yang diproses dari setiap iterasi. Selain itu, *console log* juga akan menampilkan seberapa besar nilai akurasi terakhir yang diperoleh, serta nilai akurasi terbaik yang telah diperoleh selama proses *training*. Bobot dengan akurasi terbaik akan disimpan pada file yang ada pada folder backup dengan nama yolov4-custom_best.weights untuk YOLOv4, dan yolov4-tiny-custom_best.weights untuk YOLOv4-Tiny pada *respository* github milik Alexey yang sebelumnya telah di *download*. Proses ini secara terus menerus akan berlangsung selama proses *training* yang dapat memakan waktu berjam-jam. Tergantung seberapa besar GPU yang digunakan. Semakin besar GPU yang digunakan, maka semakin cepat pula waktu prosesnya. Contoh tampilan *console log* milik penulis, diperlihatkan pada Gambar 4.12.

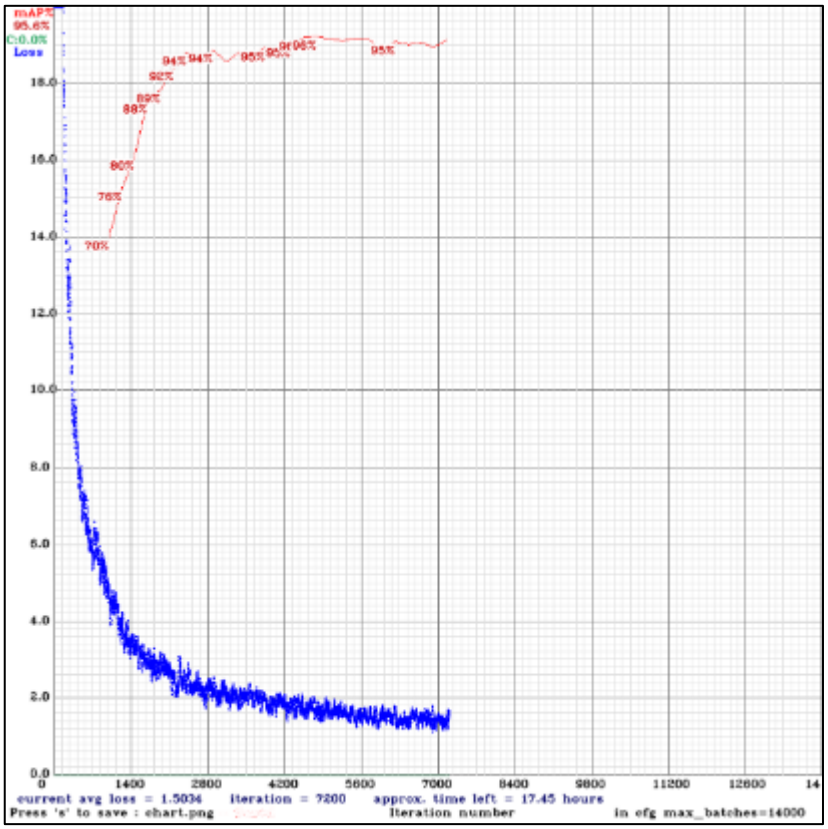


Gambar 4.12 Contoh *Console Log* Selama Proses *Training*.

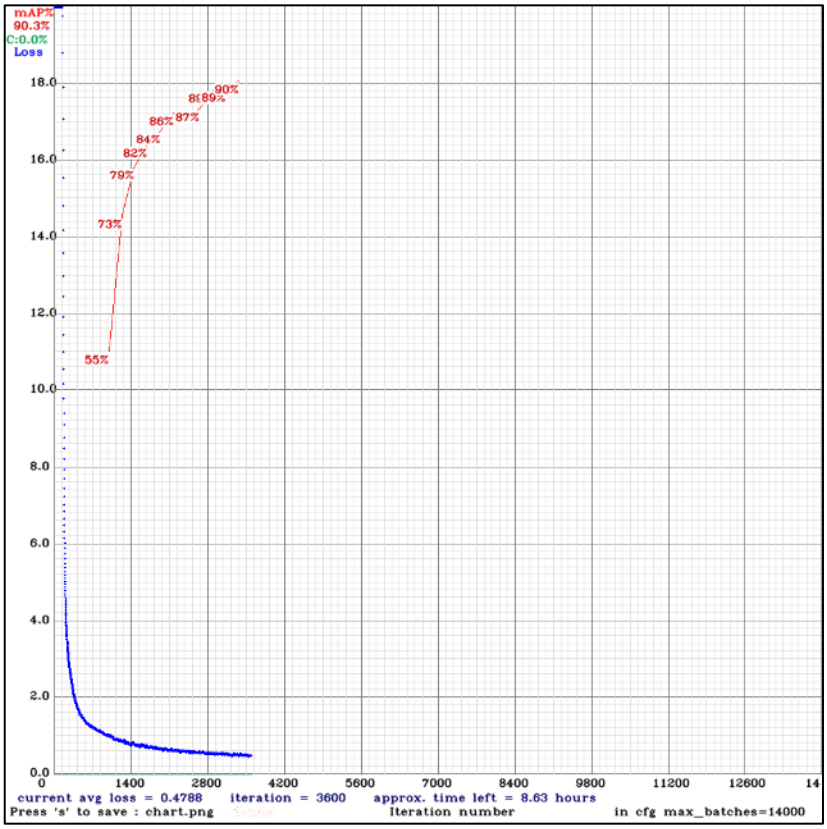
Hasil proses *training* pada penelitian ini untuk YOLOv4 berhenti pada iterasi ke 7200 dengan mAP terbaik yang diperoleh sebesar 95.6%. Untuk YOLOv4-Tiny, kami menghentikan proses pelatihan pada iterasi ke 6000 dengan perolehan mAP terbaiknya sebesar 92.5%. Alasan kami menghentikan proses pelatihan sebelum mencapai batas

maksimal dari iterasi yang kami tentukan, yakni 14000 adalah, karena selama beberapa iterasi terakhir, peningkatan mAP sudah tidak cukup signifikan atau curam. Sebagaimana dikutip dalam Github milik Alexey, apabila selama proses *Training* mAP atau nilai *loss function* tidak terdapat penurunan yang signifikan, sebaiknya proses dihentikan meski iterasi belum mencapai angka maksimal yang ditentukan (Bochkovskiy et al., 2020). Tujuannya adalah untuk menghindari *overfitting*. *Overfitting* adalah kondisi ketika model lebih bagus dalam mengenali data latih, dari pada data baru. Tampilan grafik proses pelatihan YOLOv4, ditampilkan pada Gambar 4.13. Sedangkan untuk YOLOv4-Tiny, ditampilkan pada Gambar 4.14 dan Gambar 4.15.

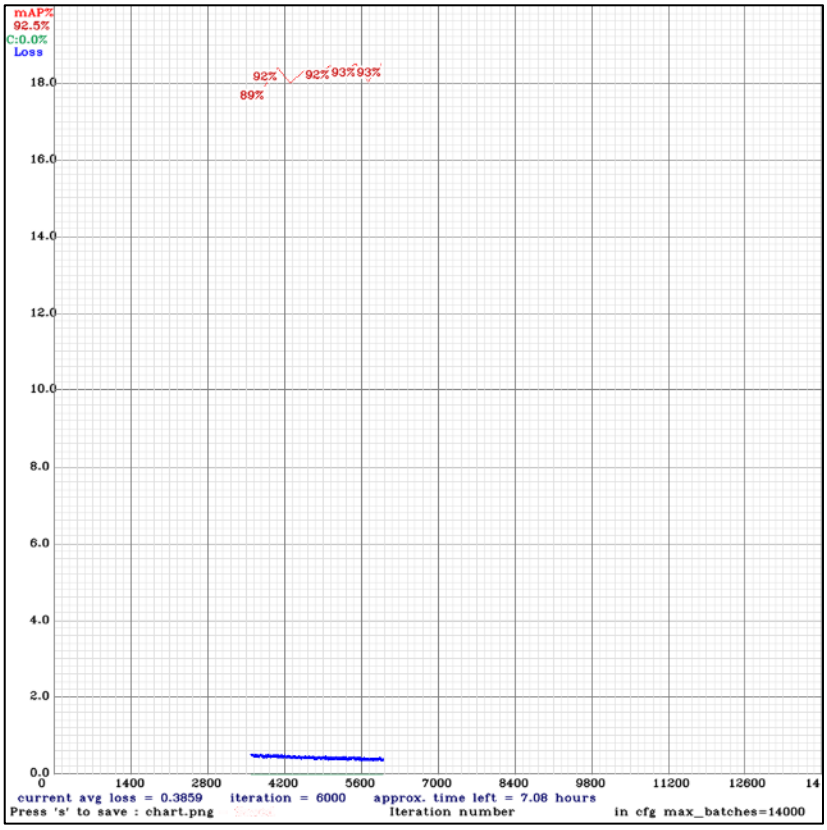
Jika diperhatikan pada Gambar 4.14, grafik YOLOv4-Tiny sempat terputus. Hal tersebut disebabkan karena terjadi kendala teknis selama proses *training* yang berakibat *sessions* dari Google Colab habis atau melewati batas waktu tunggu. Sehingga saat proses pelatihan dilanjutkan berdasarkan proses pelatihan sebelumnya, grafik yang sebelumnya telah dibuat akan ditimpa dengan grafik yang baru (Gambar 4.15). *Script* untuk dapat melanjutkan proses berdasarkan *checkpoint* terakhir apabila ada kendala teknis, ditampilkan pada Gambar 4.16 untuk YOLOv4, dan Gambar 4.17 untuk YOLOv4-Tiny.



Gambar 4.13 Grafik Proses *Training* YOLOv4



Gambar 4.14 Grafik Proses *Training* YOLOv4-Tiny Bagian Pertama



Gambar 4.15 Grafik Proses *Training* YOLOv4-Tiny Bagian Kedua

```
!./darknet detector train \
  data/obj.data \ # TARGET DATASET
  cfg/yolov4-custom.cfg \ # CONFIGURATION FOR TRAINING PROCESS
  backup/yolov4-custom_last.weights \ # CHECKPOINT FROM LAST TRAINING
  -dont_show -map
```

Gambar 4.16 *Source Code* Melanjutkan Proses *Training* YOLOv4

```
!./darknet detector train \
  data/obj.data \ # TARGET DATASET
  cfg/yolov4-custom.cfg \ # CONFIGURATION FOR TRAINING PROCESS
  backup/yolov4-custom_last.weights \ # CHECKPOINT FROM LAST TRAINING
  -dont_show -map
```

Gambar 4.17 *Source Code* Melanjutkan Proses *Training* YOLOv4-Tiny

Penjelasan dari *source code* yang ditampilkan pada Gambar 4.16 dan Gambar 4.17 adalah sebagai berikut:

1. Baris pertama adalah perintah untuk menjalankan proses *training.*
2. Baris kedua menjelaskan letak data beserta konfigurasi data *testing* dan *training*.
3. Baris ketiga menjelaskan letak konfigurasi yang akan digunakan selama proses *training*.
4. Baris keempat menjelaskan letak bobot terakhir (*checkpoint*) yang akan dilanjutkan proses *training*.

4.3 *Source Code* Evaluasi Model

Pada tahap ini, penulis akan melakukan proses evaluasi model untuk melihat hasil *precision*, *recall*, *F1 score*, mAP dan AP dari masing-masing model. *Code* untuk melakukan proses tersebut ditampilkan pada Gambar 4.18. Hasil dari evaluasi model dari YOLOv4 dan YOLOv4-Tiny ditampilkan pada Tabel 4.1.

```
!./darknet detector map \
  data/obj.data \
  /content/gdrive/MyDrive/workspace/pickfish-modelling/existing-model/Basic/yolov4-custom.cfg \
  /content/gdrive/MyDrive/workspace/pickfish-modelling/existing-model/Basic/yolov4-custom_best.weights \
  -points 0
```

Gambar 4.18 *Source Code* Menjalankan Proses *Testing*.

Penjelasan dari *source code* yang ditampilkan pada Gambar 4.18 adalah sebagai berikut:

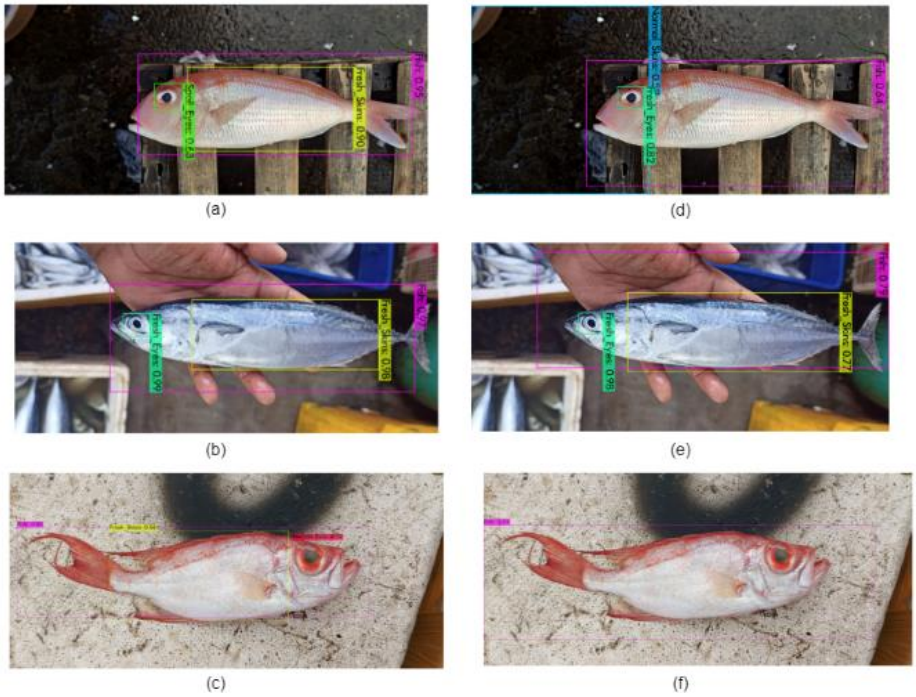
1. Baris pertama adalah perintah untuk menjalankan proses evaluasi model.
2. Baris kedua menjelaskan letak data beserta konfigurasi data.
3. Baris ketiga menjelaskan letak konfigurasi dari model (pada Gambar 4.18, contoh yang digunakan adalah YOLOv4).
4. Baris keempat menjelaskan letak bobot yang akan digunakan untuk melakukan evaluasi model.
5. Perintah -points 0 digunakan untuk melakukan evaluasi model terhadap data *custom*. Karena data yang kami gunakan adalah data pribadi, maka data tersebut tergolong data *custom*.

Tabel 4.1 Hasil Evaluasi Model

Model	**P(%)**	**R(%)**	**F_1 Score (%)**	**mAP (%)**	**AP(%)**						
					F	**FE**	**FS**	**ME**	**MS**	**SE**	**SS**
YOLOv4	89	98	93	99.17	99.1	99.0	99.7	99.8	99.3	99.6	99.7
YOLOv4 Tiny	84	97	90	97.25	98.7	98.6	97.6	94.8	95.5	98.4	96.7

Pada Tabel 4.1, singkatan masing-masing kolom di bawah *cell Average Precision* (AP) adalah nilai AP untuk kategori *Fish* (F), *Fresh Eyes* (FE), *Fresh Skins* (FS), *Medium* atau *Normal Eyes* (ME), *Medium* atau *Normal Skins* (MS), *Spoiled Eyes* (SE), *Spoiled Skins* (SS).

Untuk memastikan model dapat mendeteksi objek dengan baik, kami melakukan pengujian pada data baru untuk melihat performa model dalam mendeteksi objek mata ikan, kulit ikan, dan ikan dengan *background* yang berbeda dan data yang berbeda dari data *training* dan *testing*. Gambar 4.19 (a), Gambar 4.19 (b), dan Gambar 4.19 (c) memperlihatkan model dalam mendeteksi data baru dengan *background* yang berbeda untuk model YOLOv4. Sedangkan untuk YOLOv4-Tiny pada Gambar 4.19 (d), Gambar 4.19 (e), dan Gambar 4.19 (f)



Gambar 4.19 Hasil Deteksi Data Baru Dari Proses Model YOLOv4 ((a), (b), (c) Dan YOLOv4-Tiny ((d), (e), (f)).

Pada Gambar 4.19 (c) dan Gambar 4.19 (f), terdapat perbedaan yang signifikan saat YOLOv4 dan YOLOv4-Tiny mengenali mata dan kulit ikan Manglah. YOLOv4-Tiny, tampak gagal dalam mendeteksi mata dan kulit ikan. Hal tersebut wajar, karena secara performa mendeteksi kedetailan dari suatu objek, YOLOv4 lebih unggul jika dibandingkan dengan YOLOv4-Tiny. Pada Gambar 4.19 (d), terdapat kelas "normal" yang digunakan untuk mengkategorikan *level* kesegaran dari mata ikan. Nilai pada kelas tersebut setara dengan kelas *Medium*. Hal itu terjadi karena selama proses *training*, penulis salah menggunakan label yang digunakan. Tetapi, pada saat diimplementasikan pada aplikasi *smartphone*, kelas "Normal" masih dapat direvisi menjadi "*Medium*" agar tidak membingungkan pengguna aplikasi.

Setelah melakukan evaluasi mode, tahapan selanjutnya adalah melakukan proses konversi model kedalam bentuk TFLite sebelum model diimplementasikan ke dalam aplikasi yang dikembangkan oleh penulis.

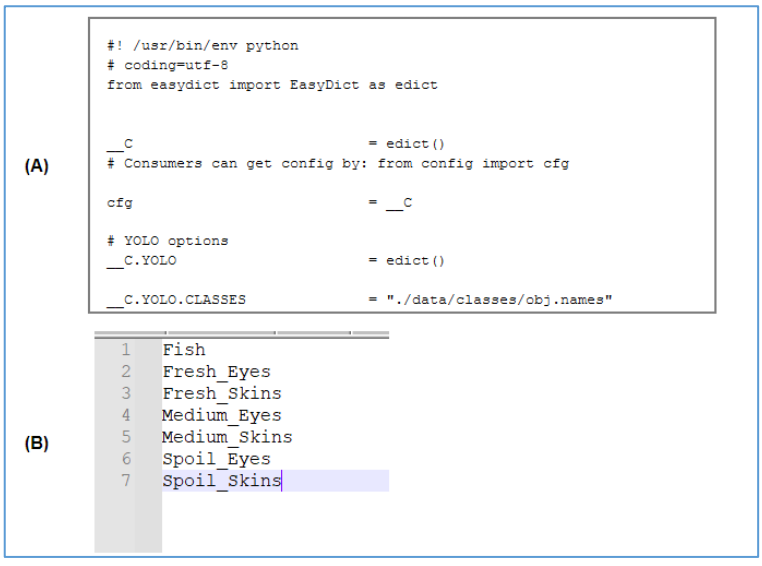
4.4 Source Code Conversion Model ke TFLite

Pada proses ini, penulis akan melakukan konversi model YOLOv4 dan YOLOv4-Tiny dari hasil *training* ke dalam bentuk TFLite. Tujuannya adalah, agar model tersebut dapat diaplikasikan ke dalam *device smartphone* berbasis *android*. Tahap pertama, melakukan

cloning repository milik Việt Hùng (Bochkovskiy et al., 2020). *Repository* tersebut terdapat *source code* untuk melakukan konversi dari bobot-bobot pelatihan yang dihasilkan selama proses *training* YOLOv4 dan YOLOv4-Tiny. Di dalamnya terdapat:

1. Folder Checkpoint untuk menyimpan model YOLOv4 dan YOLOv4-Tiny yang telah dirubah kebentuk Tensorflow dan TFLite.
2. Folder Core untuk berisikan *file* konfigurasi yang digunakan untuk mengkonversi model YOLOv4 kebentuk Tensorflow.
3. Folder data untuk meletakkan model YOLOv4 dan YOLOv4-tiny yang akan dikonversi kebentuk tensorflow.
4. Folder dan *file* yang lain adalah folder dan *file* default miliki Việt Hùng untuk melakukan proses konversi.

Sebelum melakukan konversi model YOLOv4 dan YOLOv4-Tiny kemodel TFLite, terdapat *file* pada repositori yang harus ditambahkan dan dikonfigurasi terlebih dahulu. *File* yang harus dikonfigurasi adalah *file* config.py pada folder core, dan yang ditambahkan adalah *file* baru berisikan *class* digunakan saat melatih model YOLOv4 dan YOLOv4-Tiny pada folder classes dalam folder data. Gambar 4.20 (A) menampilkan konfigurasi pada *file* config.py untuk menginformasikan letak file *class* yang digunakan. Gambar 4.20 (B) menampilkan *class* atau label yang kami gunakan (nama *file* untuk menyimpan *class* adalah obj.names)



Gambar 4.20 Isi Konfigurasi *File* config.py (A) dan Isi *File* obj.names (B).

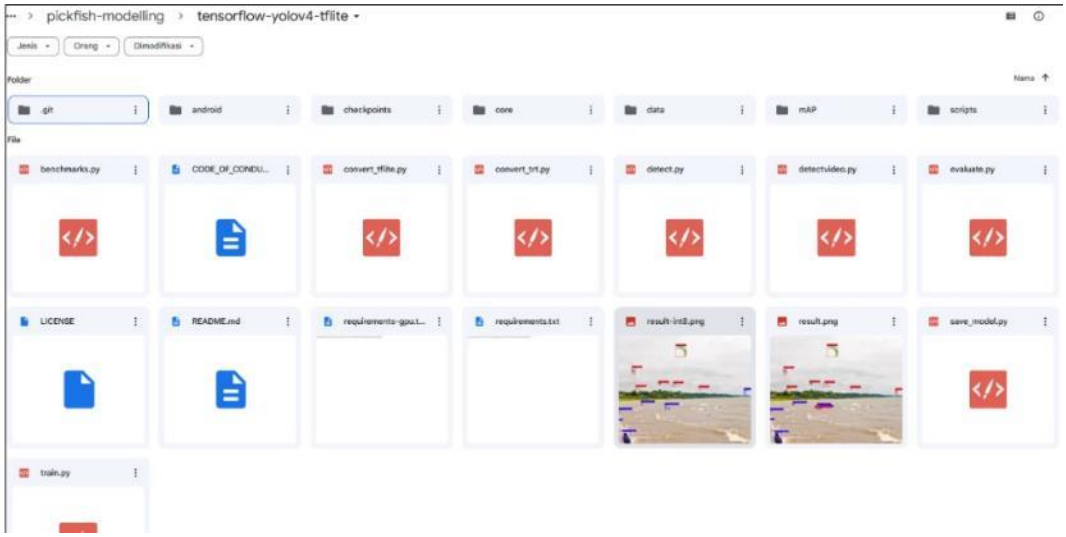
Konfigurasi letak model YOLO yang akan digunakan, letak hasil model Tensorflow akan disimpan, dimensi citra, dan lain sebagainya, akan dilakukan melalui perintah saat

menjalankan *source code*. *Source code* untuk melakukan *clonning repository*, ditampilkan pada Gambar 4.21 (A) dan isi dari repositori yang telah di *clone* ditampilkan pada Gambar 4.21 (B).

(A)

```
!git clone https://github.com/hunglc007/tensorflow-yolov4-tflite.git
```

(B)



Gambar 4.21 *Source Code* Untuk Melakukan *Clonning* (A) dan Isi Repositori Hasil *Clonning* dari Github Việt Hùng (B).

Tahapan selanjutnya adalah menyimpan model YOLO ke dalam bentuk Tensorflow. model yang telah dirubah dalam bentuk Tensorflow ini, nantinya akan dikonversi ke dalam bentuk TFLite. Gambar 4.22 adalah *source code* untuk melakukan konversi model dari YOLOv4 ke dalam bentuk Tesorflow. Untuk melakuakan konversi pada YOLOv4-Tiny kedalam bentuk Tensorflow, perintah konversinya diperlihatkan pada Gambar 4.23.

```
!python save_model.py \
  --weights ./data/yolov4-custom_best.weights \
  --output ./checkpoints/yolov4 \
  --input_size 416 \
  --model yolov4 \
  --framework tflite
```

Gambar 4.22 *Source Code* Konversi Model YOLOv4 Kebentuk Tensorflow.

```
!python save_model.py \
  --weights ./data/yolov4-tiny-custom_best.weights \
  --output ./checkpoints/yolov4-tiny \
  --input_size 416 \
  --model yolov4 --tiny \
  --framework tflite
```

Gambar 4.23 *Source Code* Konversi Model YOLOv4-Tiny Kebentuk Tensorflow.

Keterangan dari setiap baris masing-masing *code* dari Gambar 4.22 atau Gambar 4.23 adalah sebagai berikut:

1. Baris pertama, adalah perintah untuk mejalankan *script* milik Việt Hùng agar model dikonversi ke dalam bentuk Tensorflow.
2. Baris kedua adalah untuk menentukan lokasi bobot hasil *training* YOLOv4 atau YOLOv4-Tiny yang akan dikonversi.
3. Baris ketiga digunakan untuk mementukan letak *output* atau hasil dari model yang telah dikonversi.
4. Baris keempat untuk menentukan *input size image* untuk Model (pada Gambar 4.22 dan 4.23, dimensi *input size* yang digunakan kami gunakan berukuran 416x416 px. Dimensi yang digunakan harus sama seperti dimensi diatur selama proses pelatihan model YOLOv4 dan YOLOv4-Tiny).
5. Baris kalimat adalah perintah untuk memberikan informasi pada *code* point kesatu terkait model yang akan digunakan.
6. Baris terakhir untuk memberikan informasi *target framework* yang selanjutnya akan digunakan setelah model dikonversi ke dalam bentuk Tensorflow. Pada Gambar 4.22 dan 4.23 *target framework* yang selanjutnya akan digunakan setelah model di konversi kebentuk Tensorflow adalah TFLite.
7. Baris keenam pada Gambar 4.22 dan Gambar 4.23, apabila tidak diberikan keterangan *target framework* yang selanjutnya digunakan, maka konversi model yang dihasilkan bukan dalam bentuk Tensorflow, melainkan TensorRT.

Tahapan terakhir dalam proses konversi model ini adalah melakukan konversi model yang YOLOv4 atau YOLOv4-Tiny yang disimpan dalam bentuk *Tensorflow* ke dalam bentuk TFLite. Gambar 4.24 adalah *source code* untuk melakukan konversi YOLOv4 ke dalam bentuk TFLite. Sedangkan untuk *source code* YOLOv4-Tiny, ditampilkan pada Gambar 4.25.

```
!python convert_tflite.py \
  --weights ./checkpoints/yolov4/ \
  --output ./checkpoints/yolov4-fp16.tflite \
  --quantize_mode float16
```

Gambar 4.24 *Source Code* Konversi Model Tensorflow YOLOv4 Kebentuk TFLite.

```
!python convert_tflite.py \
  --weights ./checkpoints/yolov4-tiny/ \
  --output ./checkpoints/yolov4-tiny-fp16.tflite \
  --quantize_mode float16
```

Gambar 4.25 *Source Code* Konversi Model Tensorflow YOLOv4-Tiny Kebentuk TFLite.

Keterangan dari setiap baris masing-masing *code* dari Gambar 4.24 atau Gambar 4.25 adalah sebagai berikut:

1. Baris pertama, adalah perintah untuk mejalankan *script* milik Việt Hùng agar model dikonversi ke dalam bentuk TFLite.
2. Baris kedua digunakan untuk menginformasikan *script* pada baris pertama, letak dari bobot YOLOv4 atau YOLOv4-Tiny yang telah di *convert* ke dalam bentuk Tensorflow.
3. Baris ketiga digunakan untuk menentukan letak *file* TFLite yang dihasilkan dan akan disimpan saat proses konversi model selesai.
4. Baris keempat merupakan perintah penggunaan *quantization mode* yang akan digunakan oleh model TFLite.

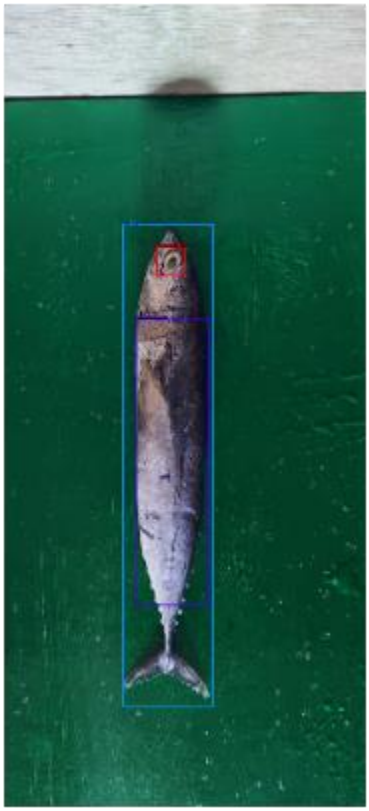
Kami menggunakan *script* pada Gambar 4.26 untuk memastikan konversi model Tensorflow sudah benar dan proses konversinya telah berhasil. Setelah script dijalankan, maka akan dihasilkan gambar baru dengan berupa ikan dengan *bounding box* hasil dari model Tensorflow. Contoh gambar ikan dengan *bounding box* dari Hasil Tensorflow ditampilkan pada Gambar 4.27.

```
!python detect.py \
  --weights yolov4-custom_best-416/ \
  --size 416 \
  --model yolov4 \
  --image ../darknet/data/Test2-SN.jpg
```

Gambar 4.26 Test *Script* Model YOLOv4 TFLite Pada Sebuah Gambar Ikan.

Keterangan dari setiap baris masing-masing *code* dari Gambar 4.26 adalah sebagai berikut:

1. Baris pertama, adalah perintah untuk menjalankan test *script* untuk mendeteksi gambar ikan milik Việt Hùng.
2. Baris kedua digunakan untuk menentukan bobot yang sudah dikonversi ke dalam Tensorflow.
3. Baris ketiga menjelaskan dimensi masukan yang akan digunakan pada model (pada Gambar 4.26, dimensi yang digunakan adalah 416 *pixel*).
4. Baris keempat untuk menentukan tipe model YOLOv4 yang akan digunakan (YOLOv4 dan YOLOv4-Tiny).
5. Baris kelima menjelaskan lokasi image yang diproses.



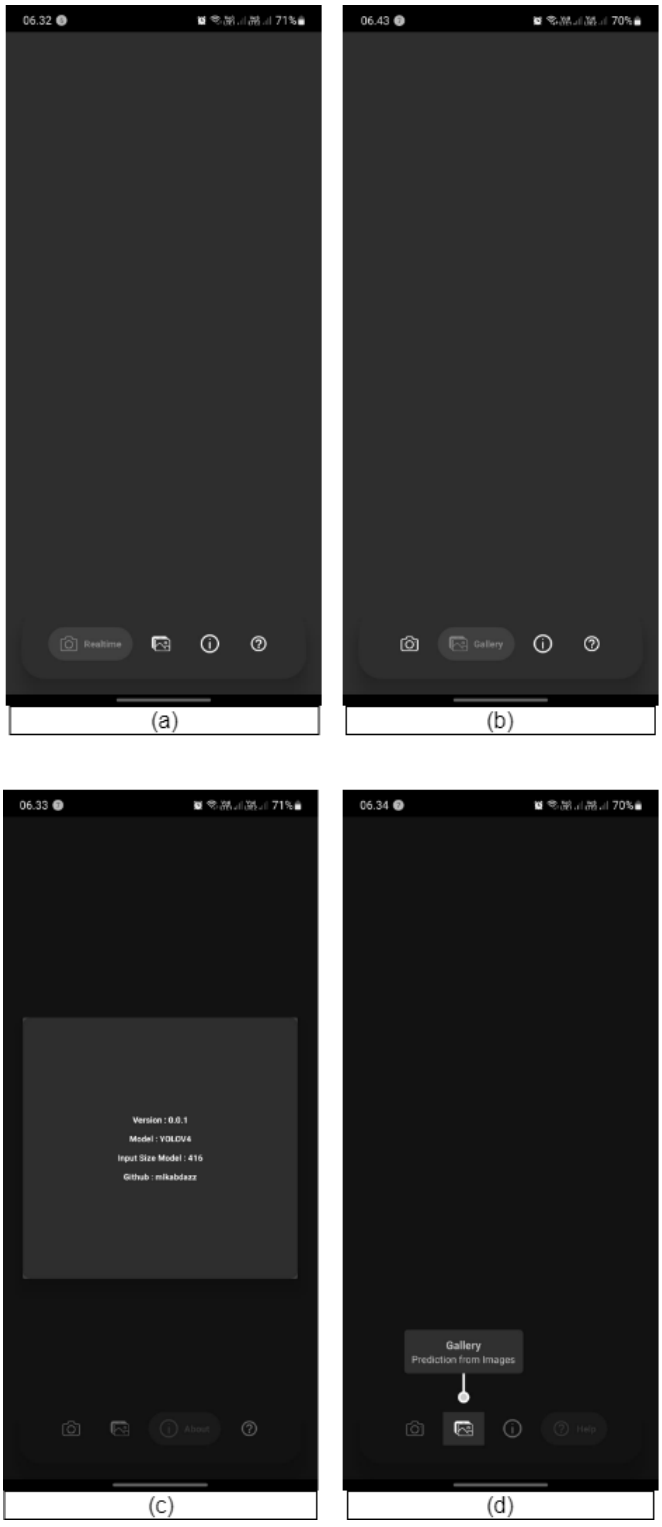
Gambar 4.27 Hasil *Testing* YOLOv4 Dalam Bentuk TFLite.

4.5 Implementasi Model pada Aplikasi

Saat pertama kali membuka aplikasi, *user* akan ditampilkan *Splash Screen* berupa logo dari aplikasi. Aplikasi pendeteksi kesegaran daging ikan, oleh penulis dinamakan PickFish. Tampilan dari Splash Screen tersebut ditampilkan pada Gambar. 4.28.

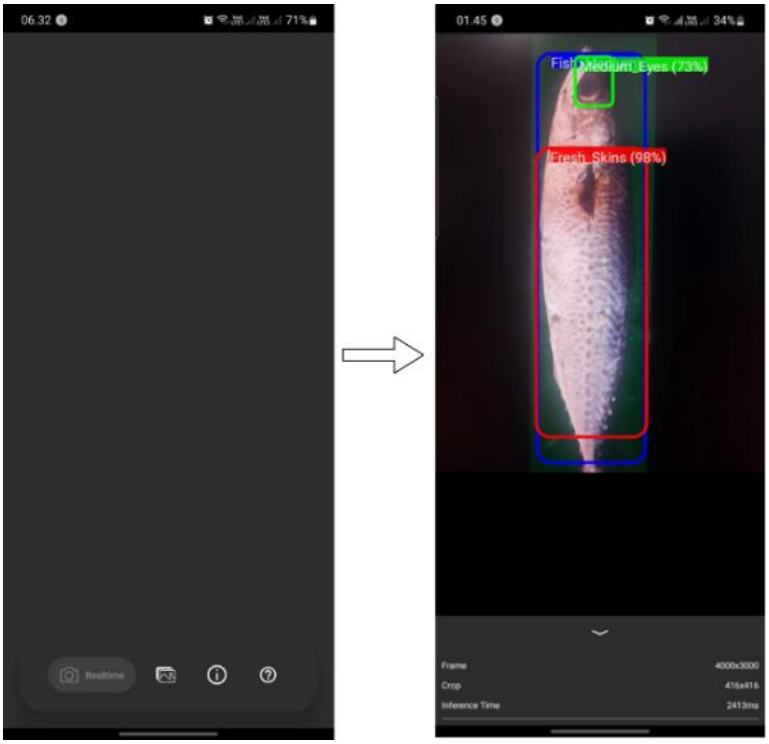
Gambar 4.28 *Splash Screen* Aplikasi.

Setelah melalui Splash Screen, pengguna selanjutnya diarahkan pada tampilan utama dari aplikasi. Pada tampilan ini, user akan diberikan 4 item menu. dimana masing-masing menu memiliki fungsi yang berbeda. Menu pertama adalah menu Realtime, yang digunakan untuk melakukan deteksi kesegaran dari daging ikan secara *realtime*. Menu kedua untuk melakukan deteksi kesegaran daging ikan pada gambar yang dipilih dari galeri. menu ketiga adalah menu informasi pengembang. Dan menu terakhir adalah menu Help, yang ditujukan sebagai menu untuk meberikan informasi singkat dari masing-masin menug pada tampilan utama. Tampilan dari masing-masing menu ditunjukkan pada Gambar 4.29.



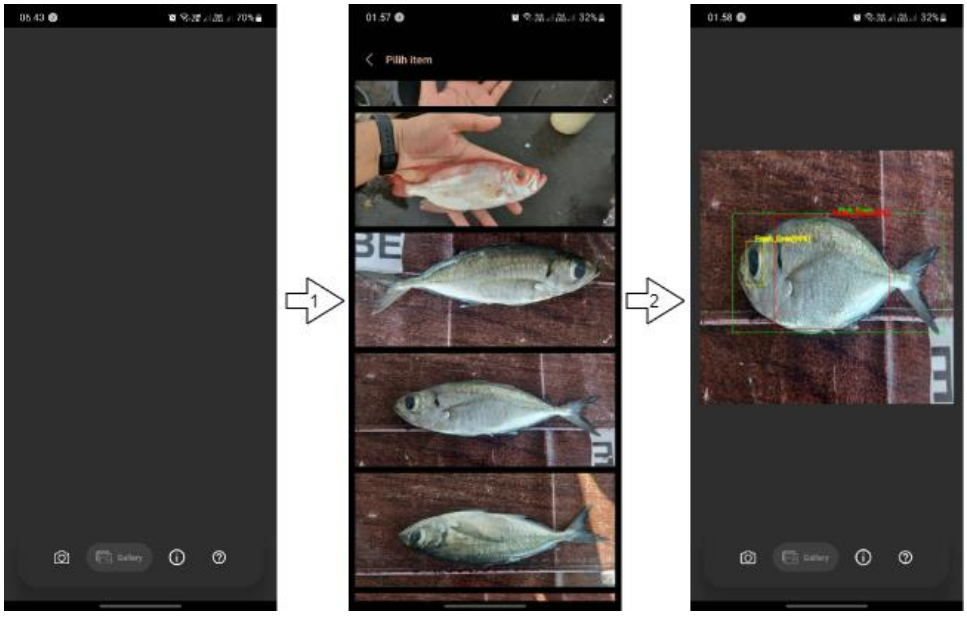
Gambar 4.29 Tampilan Utama.

Pada menu Realtime, user akan dipindahkan pada tampilan baru. Tampilan tersebut adalah tampilan untuk *user* dapat melakukan proses deteksi kesegaran daging ikan secara *realtime* menggunakan kamera pada bagian belakang. Gambar 4.30 adalah tampilan dari penggunaan deteksi kesegaran daging ikan secara *realtime*.



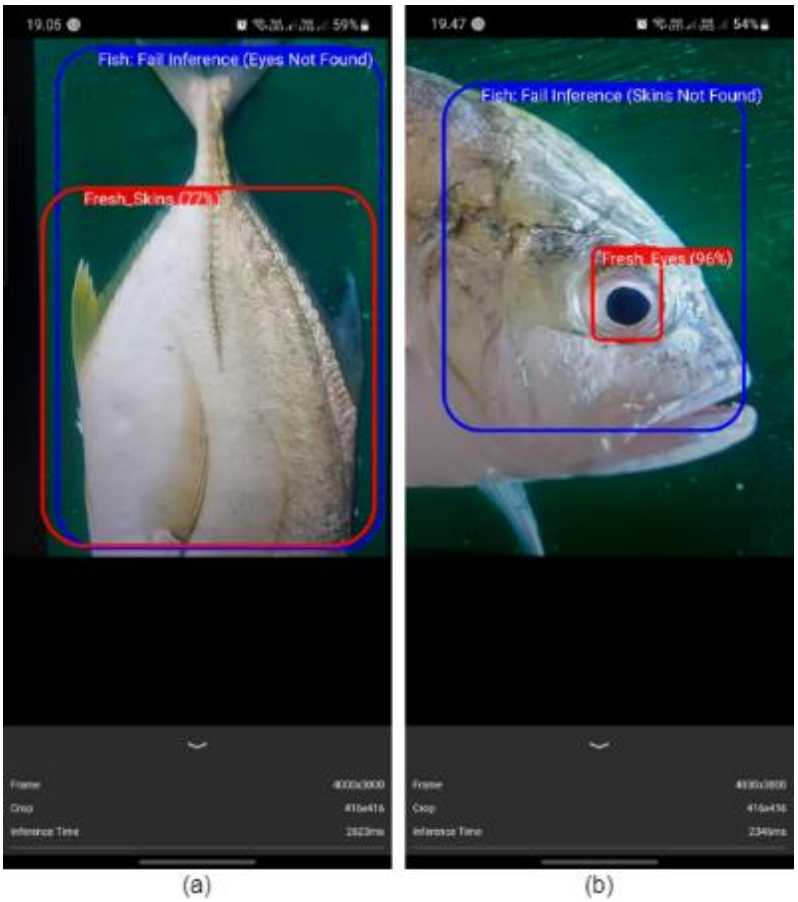
Gambar 4.30 Tampilan Menu *Realtime*.

Menu kedua adalah menu untuk melakukan deteksi kesegaran daging ikan pada sebuah gambar melalui dengan *import* gambar tersebut dari galeri *smartphone*. Gambar 4.31 menampilkan proses deteksi kesegaran daging ikan yang dilakukan dengan memilih gambar dari galeri.



Gambar 4.31 Tampilan Menu *Import* dari Galeri

Dijelaskan pada point 3.1.4, terdapat *rule* untuk menilai tingkat kesegaran daging ikan yang dirancang berdasarkan penelitian Lougovis (2005) dan pakar. Sehingga pada saat mendeteksi tingkat kesegaran daging ikan, besar kemungkinan akan terjadi *case* dimana aplikasi gagal dalam mendeteksi kesegaran daging ikan karena kurang satu parameter. Misalnya, objek dari ikan dan mata ikan ditemukan, tetapi kulitnya tidak ditemukan. *case* tersebut akan berakibat penilaian akhir dari deteksi daging ikan gagal. Untuk menghindari kebingungan dari *user* karena kegagalan aplikasi dalam menilai tingkat kesegaran daging ikan, penulis memberikan keterangan kegagalan menilai tingkat kesegaran yang diletakkan pada *bounding box* dari objek ikan. Gambar 4.30 menampilkan contoh aplikasi PickFish dalam menangani *case* atau kondisi jika ikan ditemukan, tetapi salah satu parameter dari mata ikan (Gambar 4.32 (a)) atau kulit ikan (Gambar 4.32 (b)) tidak terdeteksi oleh model saat menjalankan *rule* atau aturan (Gambar 3.9) untuk menilai tingkat kesegaran daging ikan pada aplikasi.



Gambar 4.32 Hasil Deteksi Tingkat Kesegaran Jika Aplikasi Gagal Mendeteksi Mata Ikan (a) atau kulit Ikan (b).

4.6 Perbandingan Aplikasi Model YOLOv4 dan YOLOv4-Tiny

Dari implementasi model terhadap aplikasi, kami menghasilkan dua jenis aplikasi dengan implementasi model yang berbeda. Aplikasi pertama adalah aplikasi yang mengimplementasikan model YOLOv4. Aplikasi kedua adalah aplikasi yang mengimplementasikan model YOLOv4-Tiny. Aplikasi pertama, memiliki *size* sebesar 163.6 MB. Besar *size* aplikasi yang menggunalan model YOLOv4 dikarenakan *file* TFLite dari YOLOv4 sebesar 122.3 MB. Sedangkan untuk *size* dari aplikasi kedua hanya berukuran 35.3 MB. Kami mengujikan aplikasi pada dua *device smartphone* yang bertipe Samsung S10 Lite dan Samsung A54 dengan spesifikasi yang ditampilkan pada Tabel 4.2. Hasil pengujian yang kami peroleh pada dua *device* tersebut, ditampilkan pada Tabel 4.3 untuk aplikasi yang mengimplementasikan model YOLOv4 dan Tabel 4.4 untuk aplikasi yang mengimplementasikan model YOLOv4-Tiny. Kemudian pada Tabel 4.5, kami menampilkan pengujian aplikasi dengan intensitas cahaya yang berbeda dengan menggunakan Samsung S10 Lite. Jarak yang kami atur untuk pengujian yang kami tampilkan pada Tabel 4.5 adalah sekitar 15cm.

Tabel 4.2 Spesifikasi *Device* Samsung S10 Lite dan Samsung A54

Keterangan	**Samsung S10 Lite**	**Samsung A54**
Chipset	Snapdragon 855	Exynos 1380
GPU	Adreno 640	Mali-G68 MP5
Kamera	48 Mega Pixel	50 Mega Pixel
Bulan Rilis	Februari 2020	Maret 2023

Tabel 4.3 Hasil Pengujian Model YOLOv4 Pada Kedua *Device*

Keterangan	**Samsung S10 Lite**	**Samsung A54**
Time processing (per gambar)	1.9 detik hingga 2 detik	1.9 detik hingga 2.1 detik
Jarak terjauh *device* terhadap objek hingga mampu menilai tingkat kesegaran daging ikan	40 cm	50 cm

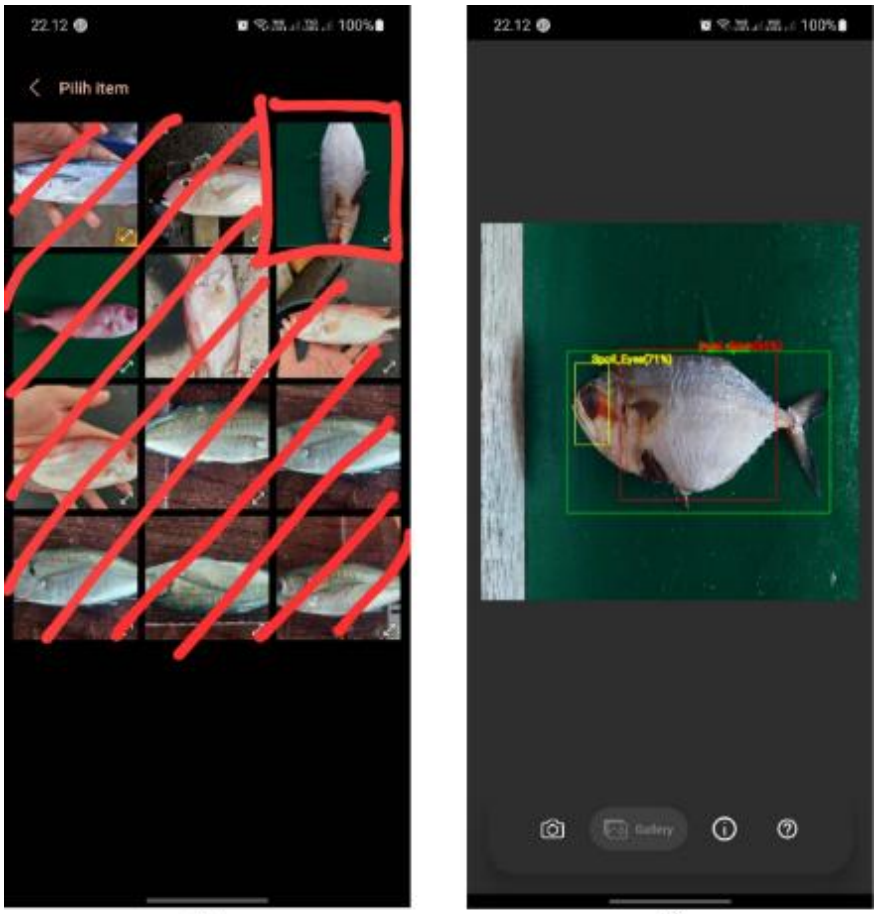
Tabel 4.4 Hasil Pengujian YOLOv4-Tiny Pada Kedua *Device.*

Keterangan	Samsung S10 Lite	Samsung A54
Time processing (per gambar)	0.13 detik hingga 0.17 detik	0.18 detik hingga 0.23 detik
Jarak terjauh *device* terhadap objek hingga mampu menilai tingkat kesegaran daging ikan	70 cm	80 cm

Tabel 4.5 Hasil Pengujian Aplikasi Dengan Intensitas Cahaya Yang Berbeda.

Model	Pencahayaan Kurang	Pencahayaan Cukup
YOLOv4	Model dapat mengenali objek mata dan kulit ikan, hingga menghasilkan tingkat kesegaran dari daging ikan. Tetapi cukup lama dan beberapa kali gagal mendeteksi bagian mata ikan.	Model dapat bekerja dengan baik untuk mendeteksi tingkat kesegaran mata, kulit, dan tingkat kesegaran daging ikan.
YOLOv4-Tiny	Model hanya mampu mengenali objek dari ikan dan cukup kesulitan dalam mengenali objek kulit dan mata ikan.	Model dapat bekerja dengan baik untuk mendeteksi tingkat kesegaran mata, kulit, dan tingkat kesegaran daging ikan.

Pada paragraf sebelumnya, kami telah menjelaskan perihal perbedaan size, hasil pengujian secara *realtime* terhadap aplikasi pada dua device yang berbeda, serta hasil pengujian aplikasi terhadap tingkat pencahayaan yang berbeda-beda. Dalam implementasi model pada menu Galeri, aplikasi memiliki kelemahan saat melakukan deteksi kesegaran daging ikan pada gambar dengan posisi *vertical*. Gambar yang dipilih dengan posisi *vertical,* nantinya posisi gambar akan berubah bentuk menjadi *horizontal.* Tetapi kelemahan tersebut tidak mempengaruhi performa model dalam mendeteksi tingkat kesegaran daging ikan. Gambar 4.33 menampilkan perubahan objek ikan yang posisi awalnya *vertical* (Gambar 4.33 (A)) saat dipilih dari galeri, kemudian menjadi *horizontal* ketika ditampilkan pada aplikasi (Gambar 4.33 (B)).



Gambar 4.33 Posisi Ikan Berubah Bentuk Dari *Vertical* (A) Menjadi *Horizontal* (B).

BAB 5
Kesimpulan dan Saran

5.1 Kesimpulan

Dari hasil dua model yang telah dilatih untuk mendeteksi tingkat kesegaran daging ikan, terdapat dua alternatif yang dihasilkan berdasarkan kondisi dari *device* yang digunakan. Penelitian ini menghasilkan dua aplikasi dengan implementasi model yang berbeda. Aplikasi pertama, adalah aplikasi yang mengimplementasikan model YOLOv4 dengan mAP yang mencapai 95.6%. Aplikasi kedua mengimplementasikan YOLOv4-Tiny dengan mAP sebesar 92.5%. Kedua aplikasi tersebut memliliki keunggulan dan kelemahan sebagaimana dijelaskan pada subbab 4.6. Mempertimbangkan ikan sebagai bahan konsumsi dan dapat membantu dalam membentuk pola hidup sehat (Tsironi et al., 2020), penelitian ini merekomendasikan untuk menggunakan aplikasi pertama. Melihat kemampuannya yang lebih baik dalam mendeteksi kesegaran dari kulit dan mata ikan meski jarak antara ikan dan kamera tidak cukup dekat. Akan tetapi, jika media penyimpanan dari *Smartphone* pengguna tidak cukup besar, maka aplikasi kedua dapat menjadi pilihan.

5.2 Saran

Pengembangan dan penelitian selanjutnya terbagi menjadi dua bagian, yaitu pada bagian *dataset* dan aplikasi. Pada bagian *dataset*, diperlukan pembaharuan atau penambahan data dengan *Background* dari ikan yang bervariasi. Untuk pengembangan pada bagian aplikasi, Rule untuk melakukan *inference* akhir dari kondisi ikan memungkinkan untuk lebih dari satu objek ikan. Selain itu, penilaian tingkat kesegaran daging ikan pada menu *Realtime* sebaiknya dapat diambil nilainya dengan menghitung rata-rata tingkat kesegaran daging ikan yang muncul pad interval waktu tertentu. Kemudian hasilnya ditampilkan pada tampilan berbeda, tidak digabung dengan *bounding box*.

Daftar Pustaka

Abd Aziz, K. N., Zainol, Z. E., Roslani, M. A., Kamaruddin, S. A., & Mohd Hanif, A. R. (2021). Fish Skin Freshness Level By Integrated Rgb Colour Image and Quality Index Method (Qim) Assessment. *Malaysian Journal of Computing*, *6*(1), 667. https://doi.org/10.24191/mjoc.v6i1.10436

Alom, M. Z., Taha, T. M., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M. S., Hasan, M., Van Essen, B. C., Awwal, A. A. S., & Asari, V. K. (2019). A state-of-the-art survey on deep learning theory and architectures. In *Electronics (Switzerland)* (Vol. 8, Issue 3). MDPI AG. https://doi.org/10.3390/electronics8030292

Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). *YOLOv4: Optimal Speed and Accuracy of Object Detection*. http://arxiv.org/abs/2004.10934

Jiang, Z., Zhao, L., Li, S., Jia, Y., & Liquan, Z. (n.d.). *Real-time object detection method for embedded devices*.

Kim, C. E., Dar Oghaz, M. M., Fajtl, J., Argyriou, V., & Remagnino, P. (2019). A comparison of embedded deep learning methods for person detection. *VISIGRAPP 2019 - Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, *5*, 459–465. https://doi.org/10.5220/0007386304590465

Kim, J. A., Sung, J. Y., & Park, S. H. (2020, November 1). Comparison of Faster-RCNN, YOLO, and SSD for Real-Time Vehicle Type Recognition. *2020 IEEE International Conference on Consumer Electronics - Asia, ICCE-Asia 2020*. https://doi.org/10.1109/ICCE-Asia49877.2020.9277040

Leghrib, R., Aantri, Y., Sanchez, J. B., Berger, F., & Kaaya, A. (2020). Assessing the freshness of Agadir blue fish using a metal oxide gas sensing array. *Materials Today: Proceedings*, *22*(xxxx), 1–5. https://doi.org/10.1016/j.matpr.2019.08.054

Li, G., Suo, R., Zhao, G., Gao, C., Fu, L., Shi, F., Dhupia, J., Li, R., & Cui, Y. (2022). Real-time detection of kiwifruit flower and bud simultaneously in orchard using YOLOv4 for robotic pollination. *Computers and Electronics in Agriculture*, *193*. https://doi.org/10.1016/j.compag.2021.106641

Maciel, E. da S., Sonati, J. G., Galvão, J. A., & Oetterer, M. (2019). Fish consumption and lifestyle: A cross-sectional study. *Food Science and Technology (Brazil)*, *39*, 141–145. https://doi.org/10.1590/fst.40617

Miao, H., Liu, Q., Bao, H., Wang, X., & Miao, S. (2017). Effects of different freshness on

the quality of cooked tuna steak. *Innovative Food Science and Emerging Technologies*, *44*, 67–73. https://doi.org/10.1016/j.ifset.2017.07.017

Moon, E. J., Kim, Y., Xu, Y., Na, Y., Giaccia, A. J., & Lee, J. H. (2020). Evaluation of Salmon, Tuna, and Beef Freshness Using a Portable Spectrometer. *Sensors (Switzerland)*, 1–12.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2016-Decem*, 779–788. https://doi.org/10.1109/CVPR.2016.91

Riley, A. P. (2005). *Food policy, control, and research*. Nova Biomedical Books.

Saputra, S., Yudhana, A., & Umar, R. (2022). Implementation of Naïve Bayes for Fish Freshness Identification Based on Image Processing. *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi)*, *6*(3), 412–420. https://doi.org/10.29207/resti.v6i3.4062

Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, *6*(1). https://doi.org/10.1186/s40537-019-0197-0

Soviany, P., & Ionescu, R. T. (2018). Optimizing the trade-off between single-stage and two-stage deep object detectors using image difficulty prediction. *Proceedings - 2018 20th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC 2018*, 209–214. https://doi.org/10.1109/SYNASC.2018.00041

Srivastava, S., Divekar, A. V., Anilkumar, C., Naik, I., Kulkarni, V., & Pattabiraman, V. (2021). Comparative analysis of deep learning image detection algorithms. *Journal of Big Data*, *8*(1). https://doi.org/10.1186/s40537-021-00434-w

Taheri-Garavand, A., Nasiri, A., Banan, A., & Zhang, Y. D. (2020). Smart deep learning-based approach for non-destructive freshness diagnosis of common carp fish. *Journal of Food Engineering*, *278*(January), 109930. https://doi.org/10.1016/j.jfoodeng.2020.109930

Tsironi, T., Houhoula, D., & Taoukis, P. (2020). Hurdle technology for fish preservation. In *Aquaculture and Fisheries* (Vol. 5, Issue 2, pp. 65–71). KeAi Communications Co. https://doi.org/10.1016/j.aaf.2020.02.001

Wu, L., Pu, H., & Sun, D. W. (2019). Novel techniques for evaluating freshness quality attributes of fish: A review of recent developments. In *Trends in Food Science and Technology* (Vol. 83, pp. 259–273). Elsevier Ltd. https://doi.org/10.1016/j.tifs.2018.12.002

Wu, T., Yang, L., Zhou, J., Lai, D. C., & Zhong, N. (2019). An improved nondestructive

measurement method for salmon freshness based on spectral and image information fusion. *Computers and Electronics in Agriculture*, *158*(January), 11–19. https://doi.org/10.1016/j.compag.2019.01.039

LAMPIRAN

Tabel lampiran berisi proses *Training*, *Testing*, beserta hasil pengujian model.

Komponen Penelitian	Keterangan / link penyimpanan
Dataset	Silahkan request melalui email themalekmalek@gmail.com
Model Hasil *Training*	https://github.com/mlkabdazz/pickfish-modelling
Source Code Aplikasi	https://github.com/mlkabdazz/pickfish
Aplikasi	https://github.com/mlkabdazz/pickfish/releases