PERANCANGAN APLIKASI KLASIFIKASI IKAN KOI BERBASIS ANDROID MENGGUNAKAN CNN DENGAN MODEL DEEP LEARNING YOLO

TUGAS AKHIR

BUDIANTO HALIM
NIM : 311710006

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI DAN DESAIN
UNIVERSITAS MA CHUNG
MALANG
2024

LEMBAR PENGESAHAN

TUGAS AKHIR

PERANCANGAN APLIKASI KLASIFIKASI IKAN KOI BERBASIS ANDROID MENGGUNAKAN CNN DENGAN MODEL DEEP LEARNING YOLO

Oleh:

BUDIANTO HALIM

NIM. 311710006

dari:

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI dan DESAIN

UNIVERSITAS MA CHUNG

Telah dinyatakan lulus dalam melaksanakan Tugas Akhir sebagai syarat kelulusan dan berhak mendapatkan gelar Sarjana Teknik Informatika

Dosen Pembimbing I,

Hendry Setiawan, ST., M.Kom.

NIP. 20100006

Dosen Pembimbing II,

Dr. Kestrilia Rega Prilianti, M.Si.

NIP. 20120035

Dekan Fakultas Teknologi dan Desain,

Prof. Dr. Eng. Romy Budhi Widodo

NIP. 20070035

PERNYATAAN KEASLIAN TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi Tugas Akhir ini dengan judul PERANCANGAN APLIKASI KLASIFIKASI IKAN KOI BERBASIS ANDROID MENGGUNAKAN CNN DENGAN MODEL DEEP LEARNING YOLO adalah benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diizinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka. Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Malang, 20 Agustus 2024

Budianto Halim

NIM. 311710006

PERANCANGAN APLIKASI KLASIFIKASI IKAN KOI BERBASIS ANDROID MENGGUNAKAN CNN DENGAN MODEL DEEP LEARNING YOLO

Budianto Halim, Kestrilia Rega Prilianti, Hendry Setiawan

Universitas Ma Chung

Abstrak

Ikan koi menjadi sorotan ikan hias air tawar yang banyak diminati dan digemari oleh orang-orang. Ikan koi menawarkan banyak jenis dan variasi yang dapat diidentifikasi dari bentuk lekukan warna tubuh, sisik, bentuk tubuh, dan lain-lain. Hal ini menimbulkan permasalahan terutama dalam menentukan jenis dari sekian banyak variasi jenis ikan koi yang ada di dunia. Maka dari itu, tujuan dari penelitian ini yaitu membuat aplikasi android yang mampu mengidentifikasi jenis ikan koi dengan sebagai informasi kepada pengguna.

Penelitian ini menggunakan model CNN YOLO yang digunakan sebagai model pengenalan objek. Salah satu kelebihan model CNN YOLO yaitu proses identifikasi gambar yang cepat, dan dapat dilakukan dengan *real-time*. Kemudian model ini akan diimplementasikan pada aplikasi android yang dikembangkan menggunakan *framework* Flutter.

Dari penelitian yang dilakukan, didapat akurasi sebesar 89% pada model dalam mengidentifikasi jenis ikan koi selama proses pelatihan dengan jumlah data digunakan sebanyak 640 gambar yang diambil secara acak pada dataset. Dapat disimpulkan bahwa pengaplikasian model ini dijalankan dengan baik walau memiliki keterbatasan data *train* dan *test*.

Kata Kunci : Convolutional Neural Network, YOLO, Flutter, Android

DEVELOPMENT OF ANDROID-BASED KOI FISH CLASSIFICATION APPLICATION USING CNN WITH YOLO DEEP LEARNING MODEL

Budianto Halim, Kestrilia Rega Prilianti, Hendry Setiawan

Ma Chung University

Abstract

Koi fish are a popular and highly sought-after freshwater ornamental fish. They offer a wide range of types and variations that can be identified by the color patterns on their bodies, scales, body shapes, and more. This diversity creates challenges, especially in determining the species among the many existing koi fish varieties. Therefore, the aim of this research is to develop an Android application capable of identifying koi fish species and providing this information to users.

This research utilizes the YOLO CNN model for object recognition. One of the advantages of the YOLO CNN model is its fast image identification process, which can be performed in real-time. The model is then implemented into an Android application developed using the Flutter framework.

The research results show an accuracy of 89% for the model in identifying koi fish species during the training process using 640 images which randomly picked from the dataset. It can be concluded that the application of this model works well despite the limitations in training and testing data.

Keywords: Convolutional Neural Network, YOLO, Flutter, Android

Kata Pengantar

Segala puji dan syukur penulis panjatkan kepada Tuhan Yang Maha Esa atas berkat dan karunia-Nya sehingga penulis dapat menyelesaikan laporan Tugas Akhir dengan sebaik-baiknya. Laporan dengan Judul "Perancangan Aplikasi Klasifikasi Ikan Koi Berbasis Android Menggunakan CNN Dengan Model Deep Learning YOLO" ini disusun sebagai syarat kelulusan perkuliahan pada Universitas Ma Chung.

Penyusunan laporan Tugas Akhir ini tidak terlepas dari dukungan, semangat, serta bimbingan dari berbagai pihak, baik bersifat moral maupun materi. Oleh karena itu penulis ingin menyampaikan ucapan terima kasih kepada:

1. Orang tua dan keluarga yang selalu mendukung dan memberi semangat selama menjalani kegiatan pengerjaan skripsi ini,
2. Ibu Dr. Kestrilia Rega Prilianti, M.Si., selaku Dekan dari Fakultas Sains dan Teknologi Universitas Ma Chung dan Dosen Penguji,
3. Bapak Windra Swastika, Ph.D, selaku Ketua Penguji.
4. Bapak Hendry Setiawan, ST., M.Kom., selaku Kepala Program Studi Teknik Informatika dan Dosen Penguji,
5. Bapak Mochamad Subianto, S.Kom., M.Cs., selaku Dosen Pembimbing Akademik,
6. Teman-teman yang selalu mendukung penulis secara emosional.

Tidak menutup kemungkinan bila dalam laporan ini terdapat kesalahan. Oleh karena itu, kritik dan saran sangat membantu bagi penulis untuk memperbaiki kekurangan yang ada. Demikian semoga laporan Tugas Akhir ini bisa bermanfaat bagi semua pihak.

Daftar Isi

Daftar Gambar

Daftar Tabel

BAB I
Pendahuluan

1.1 Latar Belakang

Ikan koi, juga dikenal sebagai *Cyprinus carpio koi*, adalah salah satu jenis ikan air tawar yang sangat populer di seluruh dunia. Ikan koi merupakan salah satu ikan hias yang diminati karena memiliki pola tubuh berwarna indah. Ikan koi unggulan dapat dilihat dari segi kualitas dan kuantitas. Nilai dari harga ikan koi ini dinilai dari ikan yang kulitnya sehat, bentuk dan ukurannya proporsional, gerakannya tampak anggun dalam air, kemudian warna koi yang paling indah dan menawan. Koi juara memiliki pewarnaan terbaik. Pola dan warna tubuhnya begitu seimbang, sesuai dengan varietasnya.

Terkait dengan hal menentukan jenis dari ikan koi sendiri, hal ini dapat menjadi suatu permasalahan dikarenakan banyaknya jenis ikan koi yang ada di dunia. Contoh dari jenis-jenis ikan koi ini antara lain yaitu Asagi, Bekko, Doitsu, Ginrin, Goshiki, Hirenaga, Kawarimono, Kohaku, Koromo, Ogon, Platinum, Showa, Shusui, Thaiso Sanke, Tancho, Utsurimono, dan lain-lain. Masing-masing jenis ikan ini memiliki corak yang berbeda, dan dipercayai oleh pada penghobi bahwa corak-corak ini memiliki makna tersendiri. Corak emas pada melambangkan kesuksesan, corak merah dan putih melambangkan simbol cinta dan kesuksesan dalam hubungan, corak platinum melambangkan kekayaan, dan lain-lain (Lengkap, Ini 15 Jenis Ikan Koi Beserta Arti Coraknya - Hewania, n.d.).

Perkembangan teknologi membawa banyak dampak pada kehidupan masyarakat sekitar. Perkembangan teknologi ini dapat menjadi dampak yang baik maupun dampak yang buruk, namun pembawa dampak ini dipegang oleh pembuat, dan pengguna dari produk teknologi itu sendiri. Contoh dari dampak positif perkembangan teknologi yang sering dijumpai saat ini antara lain *chat* yang membuat pertukaran informasi menjadi cepat dan mudah diakses dimanapun pengguna berada menggunakan fitur, dan *Artificial Intelligence* (AI) mampu melakukan berbagai macam kegiatan seperti mengidentifikasi objek, *Stable Diffusion* yang mampu membuat gambar sesuai dengan *prompt* yang dimasukkan oleh pengguna, dan lain-lain. Secara garis besar, perkembangan teknologi dapat

membawa dampak positif bila digunakan dengan baik dan sewajarnya. Namun, terdapat dampak negatif dari perkembangan teknologi ini, yaitu dengan semakin berkembangnya teknologi, terdapat perspektif buruk dengan adanya hal tersebut. Akan tetapi hal ini sering disalahgunakan dengan kebebasan yang dimiliki oleh pengguna teknologi itu sendiri, dengan melatih model dari AI menggunakan gambar-gambar yang telah dibuat oleh orang lain tanpa izin tersebut untuk membuat gambar yang digenerasi oleh AI.

Perkembangan teknologi menyediakan berbagai macam produk yang dapat digunakan sehari-hari. Produk tersebut dapat berupa produk digital maupun produk fisik yang telah dirancang untuk menjalankan tugas tertentu, baik untuk kebutuhan hiburan atau kebutuhan yang dapat membantu pekerjaan seseorang. Salah satu bentuk dari produk digital tersebut adalah aplikasi *mobile*. Menurut data milik Badan Pusat Statistik (BPS), tercatat 67,88% penduduk Indonesia yang berusia di atas 5 tahun sudah memiliki ponsel atau *handphone* pada 2022 (Adhiat, n.d.).

Dalam membuat aplikasi *mobile*, *framework* dapat membantu pengembangan aplikasi dengan mudah dan cepat. *Framework* adalah suatu kerangka kerja yang dirancang untuk mempercepat dan memudahkan *developer* dalam mengembangkan sebuah aplikasi dengan menyediakan sejumlah fitur yang sudah siap dipakai seperti komponen-komponen yang layaknya dibangun terlebih dahulu. Dengan bantuan *framework*, pengalaman dalam membangun sebuah aplikasi dapat didukung juga dengan optimasi aplikasi yang optimal, berjalan dengan lancar, terstruktur dan rapi.

Flutter adalah salah satu dari sekian banyak *framework*. Flutter dibangun oleh Google dan menggunakan bahasa pemrograman Dart, yang juga dibuat oleh Google sendiri. Menurut survei yang dikeluarkan oleh JetBrains (Vailshery, n.d.), Flutter merupakan *mobile framework cross-platform* terpopuler, tercatat pada tahun 2022. Berdasarkan hasil survei, 46% *developer*s memilih untuk menggunakan Flutter untuk membangun sebuah aplikasi. Secara keseluruhan, hampir sepertiga jumlah dari *mobile developer* memilih menggunakan teknologi *cross-platform framework*, dan dua pertiga lainnya masih memilih menggunakan *Native tools*.

Salah satu contoh layanan aplikasi *mobile* adalah aplikasi yang mampu mengidentifikasi dan mengklasifikasi objek. Kemampuan yang dimiliki oleh AI ini

didasari oleh Pengolahan Citra Digital, yaitu ilmu yang mempelajari bagaimana suatu citra yang diberikan dapat dianalisis, diidentifikasi, dan diklasifikasi sesuai dengan data yang telah diberikan untuk melatih model yang dibuat.

Convolutional Neural Network (CNN) merupakan salah satu dari banyak bagian dari *deep learning* yang merupakan jenis dari jaringan saraf tiruan yang kerap digunakan pada pengenalan objek dan piksel pada suatu citra yang diberikan. CNN terdiri menjadi tiga bagian utama, yaitu *convolutional layer*, *pooling layer*, dan *fully-connected* (FC) *layer* yang memiliki tugas masing-masing.

Penerapan metode CNN pada pengklasifikasian jenis ikan koi dapat membantu beberapa orang yang berminat untuk terjun ke hobi ini. Penerapan ini akan dibuat menggunakan *Framework* Flutter agar dapat berjalan pada sistem operasi Android, dan model akan dilatih menggunakan CNN untuk mengenali objek berupa ikan koi, dan secara eksplisit mengetahui jenis ikan koi pada gambar yang digunakan oleh peminat.

Berdasarkan paparan latar belakang di atas, maka penulis melakukan penelitiannya dengan judul "Perancangan Aplikasi Klasifikasi Ikan Koi Menggunakan Deep Learning".

1.2 Identifikasi Masalah

Berdasarkan latar belakang masalah yang dikemukakan, dapat diindentifikasikan masalah pada penelitian ini, adalah kurangnya wawasan baik oleh penggemar ikan koi maupun orang awam mengenai jenis ikan koi, dikarenakan ikan koi dengan jenis yang berbeda memiliki nilai jual yang berbeda.

1.3 Batasan Masalah

Batasan masalah dari pengembangan aplikasi berikut ini adalah:

1. Pengembangan aplikasi menggunakan *framework* Flutter dengan *package-package* yang tersedia,
2. Pengembangan dan pelatihan model pengenalan objek berupa Ikan Koi beserta klasifikasi jenis ikan koi menggunakan model CNN YOLO,

3. Terdapat 18 jenis ikan koi meliputi Asagi, Bekko, Doitsu koi, Ghosiki, Goromo, Hikarimoyo, Hikarimuji mono, Hikariutsuri, Kanoko koi, Kawarimono, Kin Ginrin, Kohaku, Sanke, Showa, Shusui, Tancho, Utsuri, dan Yamato Nishiki
4. Dataset yang diambil dari Kaggle memiliki jumlah data train dan test yang relatif sedikit dalam melatih model,
5. Menerapkan model pada aplikasi Flutter yang akan dibuat.

1.4 Rumusan Masalah

Berdasarkan latar belakang permasalahan di atas, maka rumusan masalah yang didapatkan yaitu bagaimana cara membuat aplikasi identifikasi jenis ikan koi yang dapat dengan mudah dimengerti oleh pengguna menggunakan algoritma CNN.

1.5 Tujuan

Tujuan dari pengembangan aplikasi ini adalah untuk membuat aplikasi Android yang dapat memfasilitasi pengguna dalam mengidentifikasi ikan koi secara *real-time* menggunakan algoritma CNN dengan Model Deep Learning YOLO.

1.6 Luaran

Luaran dari penelitian ini berupa aplikasi android yang dapat mengidentifikasi jenis ikan koi menggunakan algoritma CNN dengan model Deep Learning YOLO, beserta publikasi ilmiah.

1.7 Sistematika Penulisan

Sistematika penulisan skripsi ini terdiri dari lima BAB yang tersusun agar pembahasan dapat terfokus pada setiap pokok permasalahannya. Sistematika penulisan skripsi ini terdiri dari:

1. BAB I Pendahuluan

   BAB pertama, yaitu pendahuluan, berisi pembicaraan mengenai pokok permasalahan yang mendukung pengerjaan skripsi ini. Pendahuluan ini terdiri dari latar belakang, identifikasi masalah, batasan masalah, rumusan masalah, tujuan, luaran, dan sistematika penulisan.

2. BAB II Tinjauan Pustaka

   BAB kedua, yaitu tinjauan pustaka, berisi tentang komponen-komponen yang menunjang pengerjaan skripsi ini.

3. BAB III Analisis Kebutuhan Dan Perancangan Sistem

   BAB ketiga, yaitu analisis kebutuhan dan perancangan sistem, membahas tentang alur pengerjaan proyek, analisis kebutuhan, studi pustaka, serta perancangan dan persiapan mengenai alat yang dibutuhkan dalam pengembangan proyek.

4. BAB IV Hasil dan Pembahasan

   BAB keempat, yaitu hasil dan pembahasan, akan membahas tentang paparan hasil dari proyek yang sudah dikerjakan. Pembicaraan ini membahas terkait kinerja model, fungsionalitas aplikasi, dan lain-lain.

5. BAB V Simpulan dan Saran

   BAB kelima, simpulan dan saran, Menjelaskan mengenai kesimpulan dari penelitian yang telah dikerjakan, dan saran-saran yang diharapkan mampu meningkatkan kinerja dari aplikasi secara keseluruhan.

BAB II
Tinjauan Pustaka

2.1 Ikan Koi

Ikan koi merupakan salah satu dari sekian banyak jenis ikan air tawar dan juga jenis ikan hias yang sering diminati oleh penghobi karena aspek-aspek estetik yang dimiliki oleh masing-masing jenis ikan tersebut, tergantung dari motif yang dimiliki. Ikan koi merupakan salah satu jenis ikan carp amur (*Cyprinus rubrofuscus)* yang berasal dari Niigata, Jepang. Ikan koi memiliki banyak penggemar karena penampilannya yang menarik. Karena warna dan jenisnya yang indah, ikan koi dianggap sebagai ikan hias, dan para pecinta ikan koi tidak hanya menyukainya sebagai ikan hias, tetapi mereka juga dapat menawarkan peluang bisnis yang menjanjikan. Ikan-ikan ini tidak hanya memiliki warna yang indah, tetapi mereka juga dihormati karena kemampuan mereka untuk menyembul dan melompat ke atas air.

2.1.1 Jenis Ikan Koi

Dibalik keindahan yang dimiliki dari ikan koi, terdapat berbagai macam jenis dari ikan koi. Jenis dari ikan koi ini dapat dilihat dari motif yang terdapat pada bagian badan dari ikan koi tersebut. Masing-masing motif menunjukkan jenis yang berbeda dan dipercayai memiliki makna yang berbeda dari masing-masing ikan tersebut. Jenis dari ikan koi tersebut antara lain adalah sebagai berikut.

1. Asagi

Gambar 2. 1 Ikan Koi Asagi

(Sumber: https://www.orami.co.id/magazine/jenis-ikan-koi)

Ikan koi Asagi adalah ikan koi dengan ciri khas corak tubuh yang memiliki warna biru serta merah pada pangkal sirip dada. Warna merah di dasar sirip dada pada ikan koi jenis ini biasanya disebut Motoaka.

2. Bekko

Gambar 2. 2 Ikan Koi Bekko

(Sumber: https://www.orami.co.id/magazine/jenis-ikan-koi)

Ikan koi Bekko memiliki dua warna, yaitu hitam dan putih. Ciri khas dari jenis ikan koi Bekko ini adalah bercak pada tubuh ikan koi yang saling berdekatan dan dianggap sebagai ‘batu loncatan’.

3. Doitsu

Gambar 2. 3 Ikan Koi Doitsu

(Sumber: https://www.orami.co.id/magazine/jenis-ikan-koi)

Ikan koi Doitsu, atau kerap disebut sebagai ikan mas Jerman, adalah Ikan koi yang memiliki ciri khas yaitu tubuhnya yang terlihat seperti tidak memiliki sisik.

Terdapat karakteristik unik ikan koi Doitsu, yaitu ikan ini dapat lahir dengan berbagai macam kombinasi warna.

4. Goshiki

Gambar 2. 4 Ikan Koi Goshiki

(Sumber: https://www.orami.co.id/magazine/jenis-ikan-koi)

Ikan koi jenis ini berasal dari indukan Asagi dengan Kohaku. Ikan koi ini memiliki kombinasi warna yang cantik, terdiri dari warna hitam, biru, dan abu-abu pada sepanjang tubuhnya yang biasanya didominasi oleh warna merah dan putih. Alasan ikan koi ini diberi nama Goshiki karena ikan koi ini mengandung lima macam warna yang berbeda pada corak tubuhnya.

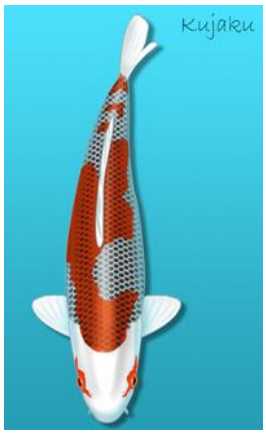
5. Goromo

Gambar 2. 5 Ikan Koi Goromo

(Sumber: https://www.kerutokoi.com/post/koi-varieties-goromo)

Ikan koi jenis ini berasal dari indukan Kohaku dan Asagi. Ikan koi ini biasanya memiliki warna bercak merah pada tubuhnya yang diikuti dengan sedikit tambahan warna biru pada bercak merahnya.

6. Hikarimoyo



Gambar 2. 6 Ikan Koi Hikarimoyo

(Sumber: https://www.kloubeckoi.com/hikari-moyo-koi/)

Ikan koi Hikarimoyo adalah jenis ikan koi yang memiliki 2 corak warna *metallic* pada tubuhnya, seperti platinum dan merah.

7. Hikarimuji mono

Gambar 2. 7 Ikan Koi Hikarimuji Mono

(Sumber: https://www.kaggle.com/code/dangtantai/koi-fish/input)

Ikan Koi Hikarimuji Mono adalah jenis ikan koi yang memiliki warna tunggal, dengan kulit yang berkilau.

8. Hikariutsuri

Gambar 2. 8 Ikan Koi Hikariutsuri

(Sumber: https://www.kaggle.com/code/dangtantai/koi-fish/input)

Ikan Koi Hikariutsuri adalah jenis ikan koi yang memiliki ciri-ciri warna dasar *metallic* dengan corak warna hitam pada tubuhnya.

9. Kanoko koi

Gambar 2. 9 Ikan Koi Kanoko Koi

(Sumber: https://www.kaggle.com/code/dangtantai/koi-fish/input)

Ikan Koi Kanoko Koi adalah jenis ikan koi yang memiliki ciri-ciri warna dasar putih dengan pola acak berwarna merah dan hitam.

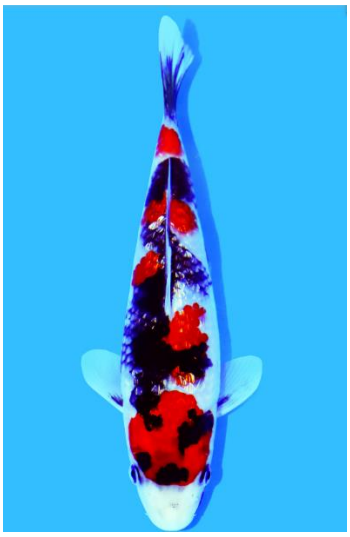
10. Kawarimono

Gambar 2. 10 Ikan Koi Kawarimono

(Sumber: https://www.kaggle.com/code/dangtantai/koi-fish/input)

Ikan koi Kawarimono adalah ikan koi yang memiliki ciri-ciri tubuh yang didominasi oleh satu warna saja.

11. Kin-Ginrin

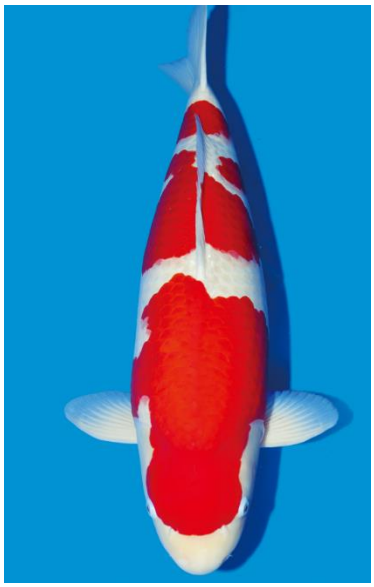


Gambar 2. 11 Ikan Koi Kin-Ginrin

(Sumber: https://www.kaggle.com/code/dangtantai/koi-fish/input)

Kin-Ginrin dapat diartikan sebagai “sisik emas dan perak”, yang juga merupakan ciri khas dari ikan koi jenis ini yaitu ikan koi yang memiliki ciri-ciri sisik berkilau berwarna emas atau perak.

12. Kohaku



Gambar 2. 12 Ikan Koi Kohaku

(Sumber: https://www.kaggle.com/code/dangtantai/koi-fish/input)

Ikan koi Kohaku adalah ikan koi yang memiliki ciri-ciri badan yang dominan berwarna merah dengan bercak merah.

13. Sanke

Gambar 2. 13 Ikan Koi Sanke

(Sumber: https://www.kaggle.com/code/dangtantai/koi-fish/input)

Ikan koi Sanke adalah ikan koi yang memiliki ciri-ciri badan berwarna putih dan bercak pola merah dan hitam pada tubuhnya.

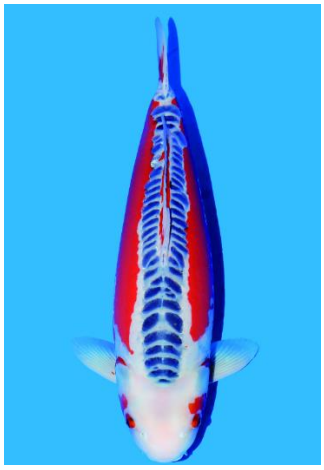
14. Showa

Gambar 2. 14 Ikan Koi Showa

(Sumber: https://www.kaggle.com/code/dangtantai/koi-fish/input)

Ikan koi Showa ini memiliki karakteristik warna yang sama seperti ikan koi Sanke, namun perbedaan yang terdapat pada ikan koi Showa yaitu terdapat pada warna badan yang dominan berwarna hitam, diikuti dengan bercak merah dan putih.

15. Shusui

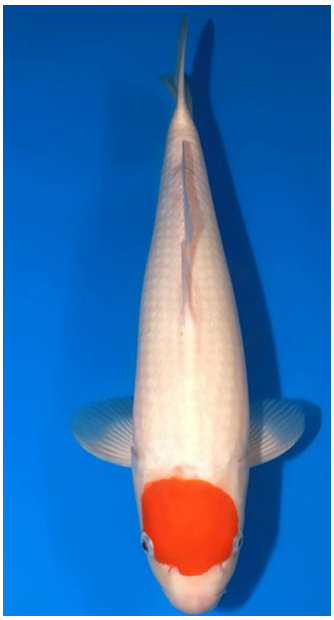


Gambar 2. 15 Ikan Koi Shusui

(Sumber: https://www.kaggle.com/code/dangtantai/koi-fish/input)

Ikan koi Shusui ini memiliki ciri-ciri unik yaitu sisik hitam di sepanjang punggungnya yang menyerupai tulang rusuk, diikuti dengan warna putih dan merah pada bagian samping tubuhnya.

16. Tancho



Gambar 2. 16 Ikan Koi Tancho

(Sumber: https://www.kaggle.com/code/dangtantai/koi-fish/input)

Ikan koi Tancho adalah ikan koi yang memiliki ciri-ciri badan berwarna putih diikuti dengan pola unik yaitu pola titik merah besar di kepala yang menyerupai mahkota.

17. Utsuri

Gambar 2. 17 Ikan Koi Utsuri

(Sumber: https://www.kaggle.com/code/dangtantai/koi-fish/input)

Ikan koi Utsuri adalah ikan koi yang memiliki ciri-ciri badan berwarna kontras seperti merah dan putih, diikuti dengan bercak garis-garis hitam tebal dan bercabang di sekitar tubuhnya.

18. Yamato

Gambar 2. 18 Ikan Koi Yamato

(Sumber: https://www.kaggle.com/code/dangtantai/koi-fish/input)

Ikan koi Yamato adalah ikan koi yang memiliki ciri-ciri sama seperti ikan koi Sanke, namun letak perbedaannya terdapat pada warna sisik putih yang memiliki karakteristik metalik, serta corak yang sama pada tubuhnya yang berwarna merah dan hitam.

2.2 Machine Learning

Machine Learning adalah salah satu cabang dari *Artificial Intelligence*. Teknologi ini banyak digunakan untuk menggantikan tugas manusia dalam menyelesaikan berbagai masalah melalui otomatisasi. Sesuai dengan namanya, *machine learning* membutuhkan proses pembelajaran terlebih dahulu untuk menyelesaikan masalah, yang dilakukan dengan memanfaatkan data. Data tersebut kemudian diproses menggunakan algoritma-algoritma pembelajaran. *Machine Learning* memiliki ciri khas yaitu adanya proses *training*. Oleh karena itu, data memiliki peran yang sangat penting karena tanpa data, *machine learning* tidak akan bisa berfungsi.

Ciri pertama dari *machine learning* adalah proses *training*, di mana sebagian data digunakan untuk melatih algoritma pembelajaran sehingga menghasilkan sebuah model. Ciri kedua adalah penggunaan data testing, yang berfungsi sebagai data uji coba untuk mengevaluasi kinerja model yang telah dilatih. Dari hasil testing ini, akan diperoleh laporan akurasi model berdasarkan data dan algoritma pembelajaran yang telah ditentukan. *Machine Learning* terbagi menjadi tiga jenis berdasarkan metode pembelajarannya, antara lain *Supervised Learning*, yaitu metode pembelajaran yang menggunakan data *training* dengan label output yang sudah ditentukan, lalu *Unsupervised Learning*, yaitu metode pembelajaran yang menggunakan data *training* tanpa label *output* yang ditentukan, dan *Reinforcement Learning*, yaitu metode pembelajaran dengan algoritma yang mendapatkan umpan balik berupa *reward* atau penalti setiap kali melakukan tindakan.

Penerapan *machine learning* di bidang teknologi meliputi pengenalan wajah pada beberapa perangkat seperti ponsel, *fingerprint recognition*, pengenalan objek tertentu seperti yang ada di *smart car*, dan masih banyak lagi.

2.3 Deep Learning

Deep Learning adalah bagian dari *machine learning* yang terinspirasi dari struktur otak manusia. Struktur tersebut dinamakan *Artificial Neural Networks* (ANN). Dalam *deep learning*, terdapat tiga lapisan yang terdiri dari input layer, *hidden* layer, dan *output* layer. *Input* layer berisi *node-node* yang setiap *nodenya* menyimpan nilai *input* tetap pada tahap *training* dan akan berubah jika diberikan nilai baru. *Hidden* layer biasanya terdiri dari beberapa lapis yang berguna untuk menemukan komposisi algoritma yang tepat guna meminimalisir *error* pada *output*. Sedangkan *output* layer berfungsi untuk menampilkan hasil perhitungan sistem dengan fungsi *aktivasi* di bagian *hidden* layer berdasarkan *input*. Gambaran dari arsitektur *deep learning* dengan tiga lapisan tersebut contohnya adalah arsitektur *Multi Layer Perceptron* (MLP).

Deep Learning, yang juga dikenal sebagai *deep structural learning*, *hierarchical learning*, atau *deep neural networks*, adalah metode pembelajaran yang menggunakan *multiple linear transformations* (Nugroho et al., 2020). Beberapa algoritma yang termasuk dalam kategori *deep learning* antara lain:

- *Convolutional Neural Network* (CNN)
- *Long Short Term Memory Network* (LSTM)
- *Recurrent Neural Network* (RNN)
- *Self Organizing Maps* (SOM)
- *Deep Neural Network* (DNN)
- *Restricted Boltzmann Machine* (RBM)
- *Deep Belief Network* (DBN)
- *Stacked Autoencoders*

Deep learning memungkinkan pemodelan data yang kompleks dan non-linear, membuatnya sangat efektif dalam berbagai aplikasi seperti pengenalan gambar, pemrosesan bahasa alami, dan pengenalan suara.

2.4 Convolutional Neural Network

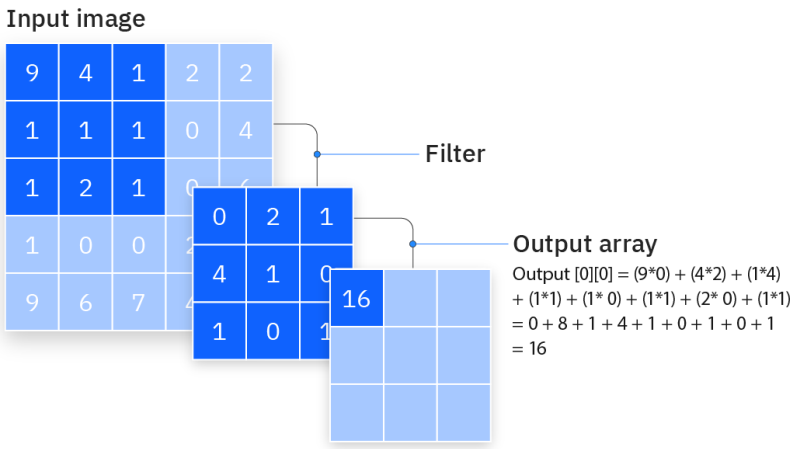
Convolutional Neural Network (CNN) adalah salah satu algoritma dari *deep learning* yang merupakan pengembangan dari *Multi Layer Perceptron* (MLP) dan digunakan untuk mengolah data dua dimensi seperti gambar atau suara. CNN juga termasuk dalam jenis *Deep Neural Network* (DNN) karena memiliki kedalaman jaringan yang tinggi dan biasanya diaplikasikan pada data berjenis citra.

CNN memiliki tiga lapisan utama, yaitu *Convolutional layer*, *Pooling layer*, dan *Fully Connected* (FC) layer. *Convolutional layer* merupakan lapisan pertama pada CNN, diikuti dengan beberapa *Convolutional layer* tambahan atau *Pooling layer*, dan diakhiri dengan *Fully Connected* layer sebagai akhirnya. Dengan bertambahnya jumlah lapisan, tingkat kompleksitas sebuah model CNN juga akan ikut bertambah. Lapisan-lapisan pertama bertanggung jawab dalam menangani fitur sederhana, seperti tepian dan warna. Seiring gambar diproses melalui lapisan-lapisan selanjutnya, model CNN akan mempelajari dan mengenali elemen-elemen rumit sampai pada akhirnya CNN mampu mengidentifikasi sebuah objek.

Convolutional layer adalah *building-block* utama pada CNN, pada layer inilah dimana komputasi paling sering terjadi. Layer ini membutuhkan beberapa komponen, yaitu *input* data, filter, dan sebuah map *feature*. Sebuah gambar berwarna yang dimasukkan akan menjadi sebuah *input* data yang memiliki 3 dimensi. 3 dimensi ini meliputi *dimension* – tinggi, *width*, dan *depth* – yang

merepresentasikan RGB pada sebuah gambar. Pada lapisan ini juga terdapat *feature detector*, yang dikenal juga sebagai filter, yang akan bergerak pada perlahan pada seluruh bagian gambar untuk mengecek apakah fitur terdapat pada gambar yang dijadikan *input*. Proses ini juga dikenali sebagai *convolution*.

Filter adalah *array* 2 dimensi yang merepresentasikan sebuah gambar. Walaupun ukurannya sering berbeda-beda, filter biasanya berukuran *matrix* 3x3 yang juga menentukan ukuran dari *receptive field*. Filter ini akan diaplikasikan kepada area gambar, lalu produk berupa dot yang dihasilkan adalah kalkulasi dari *pixel* pada gambar dengan filternya. Dot ini lalu akan dimasukkan pada *output array*. Selanjutnya, filter akan bergerak sesuai dengan *stride* yang sudah ditentukan, mengulangi proses kalkulasi dot sampai semua gambar dikalkulasi dan menghasilkan *output array* penuh. *Output array* ini biasa disebut sebagai *feature map*, *activation map*, atau *convolved feature*.



Gambar 2. 19 Ilustrasi Cara Kerja Filter Terhadap Citra

CNN sangat efektif dalam mengolah dan mengenali pola pada data citra, sehingga banyak digunakan dalam berbagai aplikasi seperti pengenalan wajah, klasifikasi objek, deteksi benda, dan lain-lain.

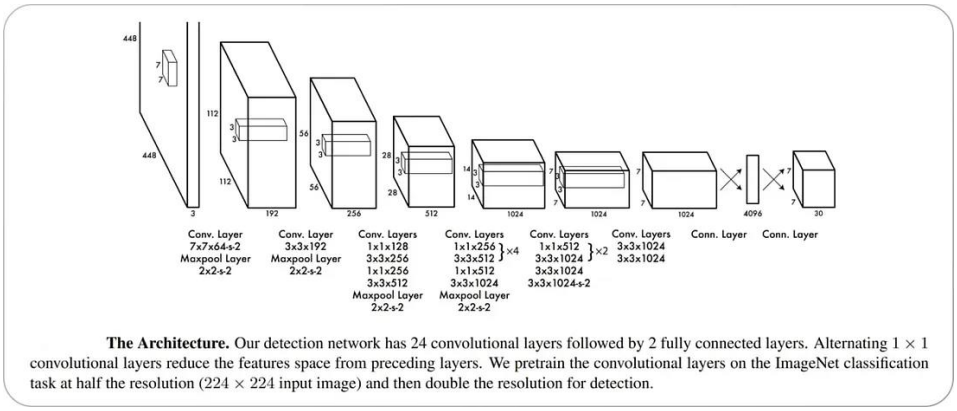
Pooling layer, dikenal sebagai *downsampling*, meliputi proses pengurangan resolusi dan mengurangi jumlah parameter pada *input*. Sama seperti *convolutional layer*, operasi *pooling* akan melakukan proses filter pada seluruh bagian pada *input*, letak perbedaannya yaitu filter ini tidak memiliki bobot apapun, namun kernel akan mengaplikasikan *aggregation function* pada *receptive field*. Ada 2 tipe utama dalam melakukan pooling, yaitu *Max Pooling* dan *Average Pooling*. *Max Pooling* akan

berjalan menyusuri gambar seperti yang dilakukan filter, lalu mengambil nilai tertinggi pada *receptive field*, lalu dikirim ke *output array*. *Average Pooling* akan berjalan menyusuri gambar seperti yang dilakukan filter, namun nilai yang dikirim ke *output* layer adalah nilai rata-rata pada jumlah *receptive field*.

Fully Connected Layer akan melakukan proses klasifikasi berdasarkan fitur yang telah diekstrak melalui proses yang dilalui oleh layer-layer sebelumnya. Dikala *Convolutional layer* dan *Pooling layer* cenderung menggunakan ReLU *function*, *Fully Connected* layer biasanya menggunakan *softmax activation function* untuk mengklasifikasi *input*, dengan probabilitas *output* dari 0 sampai 1.

2.5 YOLO Object Detection

You Only Look Once (YOLO) adalah salah satu dari sekian banyak model klasifikasi objek yang banyak digunakan *developer* untuk pengembangan aplikasi pengenalan dan klasifikasi objek. Model YOLO ini pertama kali dikembangkan oleh Joseph Redmon dan Ali Farhadi pada tahun 2015. Salah satu kelebihan model YOLO ini adalah kemampuannya dalam mendeteksi objek secara *real-time*.



The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

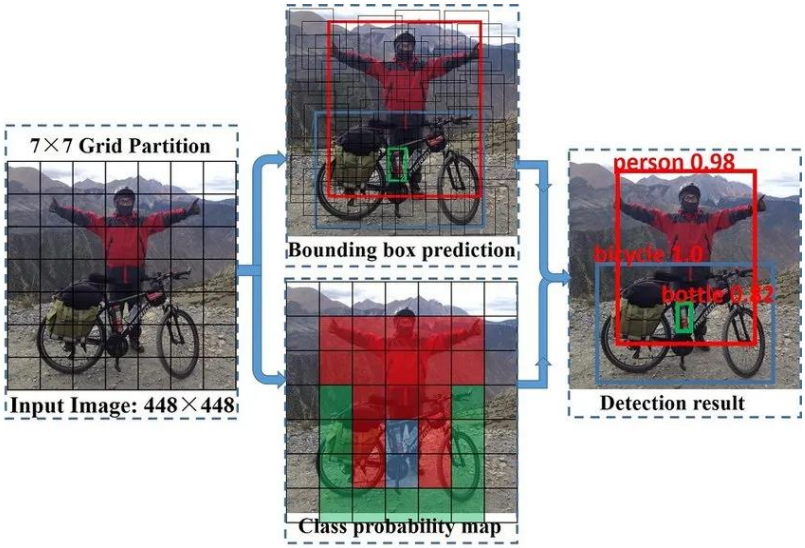
Gambar 2. 20 Struktur Model YOLO

Algoritma dari model YOLO sendiri bekerja dengan membagi input yang dimasukkan menjadi bagian-bagian berupa *cell* yang berukuran sama pada setiap *cell*-nya, lalu memprediksi probabilitas keberadaan objek dan memprediksi lokasi koordinat *bounding box* pada objek yang ditemukan. Model YOLO juga mampu memprediksi *class* pada objek yang ditargetkan. Berbeda dengan *two-stage object detector* seperti R-CNN dan jenis lainnya, YOLO merupakan *single-stage object detector* model, yang berarti YOLO memproses input yang dimasukkan hanya melewati satu proses memprediksi objek yang ada, *bounding box*, dan juga

probabilitas *class*-nya, menjadikan proses model identifikasi semakin cepat dan lebih efisien.

Hingga saat ini, YOLO sudah memiliki banyak versi, terdiri dari YOLOv1, YOLOv2, YOLOv3, YOLOv4, YOLOv5, YOLOv6, YOLOv7, sampai dengan YOLOv8. Setiap model YOLO yang baru selalu menggunakan struktur versi sebelumnya, dengan tambahan beberapa fitur baru demi meningkatkan akurasi, waktu pemrosesan yang lebih cepat, dan meningkatkan kemampuan dalam mengatasi objek-objek berukuran kecil.

Cara kerja model YOLO dimulai dengan memasukkan input yang akan diteruskan kepada CNN untuk mengekstrak fitur yang ada didalam input tersebut. Setelah itu, fitur yang sudah terpindai tadi akan diteruskan lagi pada sekumpulan *Fully Connected Layer*, yang akan memprediksi objek dan lokasi *bounding box* pada input tersebut. Selanjutnya, input yang dimasukkan tadi akan dipisah menjadi sekumpulan cells yang bertanggung jawab dalam memprediksi objek yang ada pada setiap cell, serta menentukan *bounding box* yang tepat. Luaran yang didapat setelah proses sebelumnya yaitu prediksi sementara terhadap objek yang sudah dideteksi, termasuk *class* dan juga *bounding box*-nya. *Bounding box* yang sudah didapat tadi kemudian difilter kembali menggunakan *post-process algorithm* yang disebut sebagai *non-max supression*, yang digunakan untuk menghilangkan sebagian ukuran *bounding box* pada objek dan memilih objek dengan probabilitas tertinggi. Pada proses terakhir, output didapat berupa hasil prediksi *bounding box* beserta label *class* yang diprediksi pada input yang dimasukkan.

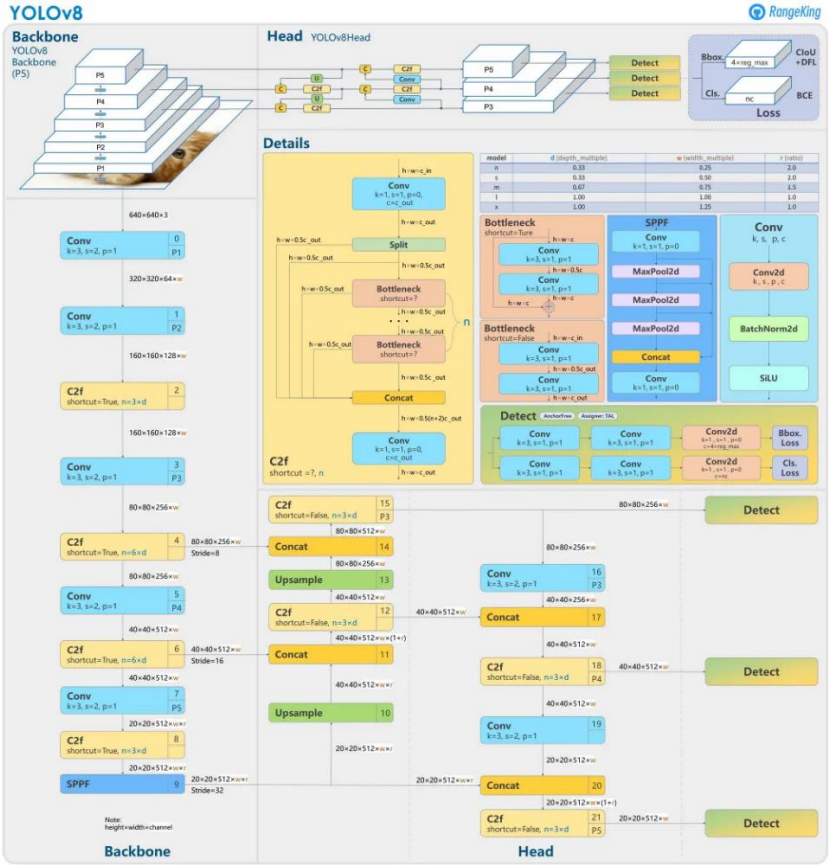


Gambar 2. 21 Alur Pengenalan Objek Menggunakan YOLO
(Sumber: https://medium.com/@nurulnyiqoniah/apa-itu-yolo-b2a5c1eb53d4)

YOLOv8 pertama kali dikembangkan oleh Ultralytics. Struktur model YOLOv8 dibagi menjadi 3 bagian, yaitu *Backbone*, *Neck*, dan *Head*. *Backbone* pada YOLOv8 menggunakan CSPDarknet53, yang merupakan modifikasi arsitektur dari versi Darknet untuk meningkatkan kinerja model pada kapasitas dan *learning rate*-nya. *Backbone* bertanggung jawab dalam mengekstrak fitur dasar pada input yang dimasukkan. Kemudian *Neck* pada YOLOv8 menggunakan PANet, yaitu sebuah *feature pyramid network* yang memfasilitasi alur informasi dari berbagai macam ukuran dari input, dan juga meningkatkan kemampuan model dalam memprediksi objek dengan ukuran-ukuran yang berbeda. Lalu terakhir, *head* pada YOLOv8 bertugas untuk menentukan *bounding box*, skor *confidence* dan juga *class* yang terdapat pada input yang dimasukkan. Struktur dari YOLOv8 dapat dilihat pada gambar 2. 22.

Salah satu fitur terbaru pada YOLOv8 yang membedakan versi ini dengan versi sebelumnya adalah *Anchor-free detection*. *Anchor-free detection* adalah pendekatan di mana model deteksi objek memprediksi langsung pusat objek tanpa mengukur *offset* dari *anchor box* yang telah ditentukan sebelumnya. *Anchor box* adalah sekumpulan kotak dengan *height* dan *width* tertentu yang digunakan untuk mendeteksi kelas objek dengan skala dan rasio aspek yang diinginkan, dipilih berdasarkan ukuran objek dalam *dataset* pelatihan, dan ditempatkan di seluruh

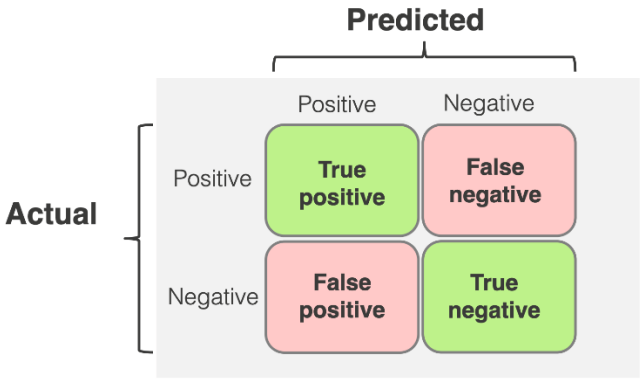
gambar selama proses deteksi. Model kemudian menghasilkan probabilitas dan atribut seperti latar belakang, IoU, dan offset untuk setiap kotak yang ditempatkan, yang digunakan untuk menyesuaikan anchor box. Beberapa *anchor box* dapat didefinisikan untuk berbagai ukuran objek, berfungsi sebagai titik awal tetap untuk penentuan batas kotak. Keunggulan dari deteksi anchor-free adalah fleksibilitas dan efisiensi yang lebih tinggi, karena tidak memerlukan spesifikasi manual dari anchor box, yang sering kali sulit dipilih dan dapat menghasilkan hasil yang kurang optimal pada model YOLO sebelumnya seperti v1 dan v2 (YOLOv8 for Object Detection Explained [Practical Example] | by Encord | Encord | Medium, n.d.).



Gambar 2. 22 Struktur YOLOv8

2.6 Confusion Matrix

Confusion matrix adalah suatu metode yang digunakan untuk menunjukkan peforma model dalam melakukan klasifikasi pada data uji yang sudah diketahui hasil sebenarnya.



Gambar 2. 23 Contoh Confusion Matrix

Ilustrasi pada gambar 2. 23 menjelaskan bagaimana cara kerja *confusion matrix*. *Confusion matrix* bekerja sebagai catatan poin benar suatu model dalam melakukan klasifikasi pada proses pengujian. *Confusion matrix* akan memberikan luaran berupa 4 bagian penting. Seluruh bagian tersebut digabung menjadi tabel sederhana seperti ilustrasi di atas. Visualisasi dari tabel ini adalah menganalisis hasil prediksi yang dibuat oleh model, sehingga tingkat keakuratan sebuah model dapat diketahui, dan dapat mudah dipahami. Bagian-bagian pada *confusion matrix* terdiri dari:

1. True Positive (TP) : Total prediksi oleh model dimana citra kelas positif diprediksi positif.

2. True Negative (TN) : Total prediksi oleh model dimana citra kelas negatif diprediksi negatif.

3. False Positive (FP) : Total prediksi oleh model dimana citra kelas negatif diprediksi positif.

4. False Negative (FN) : Total prediksi oleh model dengan citra kelas negatif diprediksi negatif.

Untuk menghitung hasil akurasi dari suatu model pada dataset pengujian, digunakan sebuah rumus seperti berikut.

$$Akurasi = \frac{TP + TN}{TP + TN + FP + FN} \times 100\% \qquad (2-1)$$

2.7 Optimizers

Pada *Deep Learning*, *optimizer* adalah elemen penting dalam menyempurnakan parameter neural *network* selama proses *training* model. Tujuan utama dari *optimizers* ini adalah untuk mengurangi *error* atau *loss function* pada model, sehingga performa model akan meningkat. *Optimizers* menggunakan berbagai strategi untuk mencapai nilai parameter optimal secara efisien, yang akan meningkatkan tingkat keakuratan prediksi pada model yang dilatih.

Beberapa algoritma *optimizers* ternama seperti Stochastic Gradient Descent (SGD), Adam dan RMSprop, dilengkapi dengan *update rules*, *learning rates*, dan momentum strategi yang berbeda-beda, bertujuan untuk mencapai parameter model yang optimal, sehingga dapat meningkatkan performa secara keseluruhan.

Dalam melatih model *deep learning*, *optimizers* akan memodifikasi bobot pada setiap epoch untuk meminimalkan *loss function* dan meningkatkan akurasi. Memilih bobot yang tepat untuk model adalah tugas yang menantang, karena model *deep learning* umumnya terdiri dari jutaan parameter. Ini menimbulkan kebutuhuan untuk memilih algoritma optimasi yang sesuai untuk model yang dibuat.

Terdapat beberapa hal penting yang perlu diketahui terlebih dahulu sebelum mendalami *deep learning*, terutama mengenai *optimizers* ini. Hal-hal tersebut antara lain

1. Epoch : Jumlah pengulangan yang dijalankan pada algoritma untuk seluruh proses pelatihan dataset.
2. Sampel : Dataset yang digunakan untuk melatih model.
3. Batch : Menunjukkan jumlah sampel yang diperlukan untuk memperbarui parameter model.
4. Learning Rate : Parameter yang menentukan sejauh mana bobot model harus diperbarui.
5. Cost Function/Loss Function : Digunakan untuk menghitung selisih antara nilai prediksi dan nilai aktual.
6. Bobot : Parameter yang bisa diatur dalam sebuah model untuk mengatur sinyal antara dua neuron.

2.7.1 AdamW Optimizers

AdamW adalah varian dari optimizer Adam (Adaptive Moment Estimation) yang dirancang untuk mengatasi masalah overfitting yang disebabkan oleh penurunan bobot (weight decay) yang tidak konsisten. Berbeda dengan Adam, AdamW memisahkan proses penurunan bobot dari pembaruan gradien, sehingga penurunan bobot dilakukan secara eksplisit setelah gradien diperbarui (Loshchilov & Hutter, 2017). Pendekatan ini memastikan bahwa gradien tetap akurat dan tidak terpengaruh oleh penurunan bobot, sehingga regularisasi lebih efektif dan membantu mencegah overfitting pada model machine learning.

2.8 Python

Python adalah bahasa pemrograman *open source* yang dibuat dan dikembangkan oleh Guido Van Rossum pada tahun 1990. Nama "Python" diambil dari acara televisi terkenal bernama "Monty Python's Flying Circus," yang merupakan acara favorit Guido Van Rossum. Lima tahun kemudian, Python dikembangkan lebih lanjut oleh Van Rossum untuk lebih kompatibel dengan teknologi masa itu. Hingga tahun 2000, Python terus diperbarui hingga versi 3 dan seterusnya.

Python adalah salah satu bahasa pemrograman yang dapat mengeksekusi beberapa instruksi multi guna secara langsung (*interpreted*) dengan basis orientasi objek (*Object-Oriented Programming*). Python dikenal karena kemampuannya menggabungkan kapabilitas dan sintaksis kode yang jelas, serta dilengkapi dengan berbagai *library* yang memiliki fungsionalitas masing-masing. Sebagai bahasa pemrograman tingkat tinggi, Python dirancang agar sintaksis kodenya mudah dipahami dan dipelajari.

Beberapa fitur menarik dari Python antara lain:

- Sistem Pengelolaan Data dan Memori Otomatis: Python mengelola data dan memori secara otomatis, sehingga memudahkan pengembang.
- Modul yang Selalu Diperbarui: Python memiliki modul-modul yang secara rutin diperbarui.
- Banyak *Library* Fungsional: Python memiliki banyak *library* dengan fungsionalitas khusus yang membantu pengembang dalam membuat program.

Python banyak diaplikasikan di berbagai sistem operasi terkenal seperti Linux, Windows, Mac OS, Android, Symbian OS, Palm, Amiga, dan lain-lain. Dalam pengembangan di ranah *machine learning* dan *computer vision*, Python menyediakan berbagai *library* yang sangat membantu, antara lain:

- OpenCV: Digunakan untuk *computer vision*.
- Keras: *Library* untuk neural *networks*.
- Matplotlib: Untuk visualisasi data.
- TensorFlow: Platform untuk *machine learning*.
- PyTorch: *Library* untuk *deep learning*.
- Scikit-learn: Digunakan untuk *data mining* dan *data analysis*.
- NumPy: Untuk komputasi ilmiah.
- Pandas: Untuk manipulasi dan analisis data.

Dengan fitur-fitur ini, Python menjadi pilihan utama bagi banyak pengembang dan ilmuwan data dalam berbagai aplikasi pemrograman dan analisis data.

2.9 Google Colab

Google Colaboratory, atau sering disebut juga Google Colab, merupakan Notebook Jupyter berbasis *online cloud* yang dapat digunakan pengguna untuk mengeksekusi perintah kode Python pada browser, yang sering digunakan pada pengembangan bidang *Machine Learning*, dan *Data Science*. Kelebihan Google Colab ini yaitu kemudahan dalam mengakses dengan menggunakan browser apapun. Proses eksekusi kode dan komputasi pada Colab dijalankan menggunakan GPU dan TPU oleh penyedia jasa, sehingga proses dapat dijalankan dengan cepat tanpa terikat pada spesifikasi komputer. Akses pada *library-library* yang sering populer pun tersedia dan dapat langsung di-*import* pada proyek yang sedang dikerjakan. Google Colab menyediakan 2 model paket berlangganan, yaitu gratis dan berbayar. Paket berlangganan berbayar pada Google Colab menyediakan GPU yang lebih cepat dan memori yang lebih luas sehingga pengguna dapat merasakan kelebihan langganan paket ini.

2.10 Android

Android adalah sistem operasi *mobile* yang populer pada perangkat *smartphone* dan tablet. Android dikembangkan oleh Google sejak 2003, mulanya dirancang sebagai sistem operasi pada kamera digital, hingga pada tahun 2004 projek dialihkan menjadi sistem operasi untuk *smartphone* dan merupakan versi modifikasi dari sistem kernel Linux dan *open-source software* lainnya, yang biasanya didesain untuk ponsel layar sentuh. Android pertama kali diluncurkan pada November tahun 2007, dengan perangkat komersial pertama mereka, HTC Dream yang diluncurkan pada September 2008.

Salah satu kelebihan dari Android sendiri yaitu sistem operasi yang bersifat *open-source*, yang berarti semua aplikasi yang dapat digunakan pada *smartphone* dengan Android dapat dikembangkan oleh banyak programmer, dibantu dengan ketersediaan referensi yang banyak dan dapat diakses dengan mudah.

2.11 Flutter

Flutter adalah sebuah *framework cross-platform* dan juga *UI toolkit* yang dirancang agar *developer* mampu membangun aplikasi yang dapat dijalankan pada sistem operasi yang berbeda seperti iOS, Android, Desktop, maupun Web hanya dengan satu *codebase*. Flutter merupakan sebuah *framework* bersifat *open-source* yang dikembangkan oleh Google, dan didukung oleh bahasa pemrograman Dart.

Gambar 2. 24 Logo Flutter

(Sumber: https://flutter.dev/brand)

Sejarah dibuatnya Flutter dimulai pada tahun 2015, dengan dibentuknya sebuah tim kecil di Google, dipimpin oleh para *developer* yang bekerja pada proyek Chrome, mulai mengerjakan pada suatu proyek baru yang dinamai "Sky". Sky diharapkan mampu menyediakan sarana pengembangan *user interface* yang cepat dan fleksibel. Namun, Sky justru berhadapan dengan beberapa permasalahan teknis, sehingga tim tersebut sepakat untuk mengulang dan memulai dari awal.

Pada awal 2017, nama proyek tersebut diubah menjadi "Flutter" dan mengubah acuan agar memfokuskan tujuan pada pembuatan *framework cross-platform* menggunakan bahasa pemrograman Dart. Dart adalah salah satu dari proyek *open-source* yang dikembangkan oleh Google yang secara resmi dirilis pada tahun 2011. Flutter memanfaatkan fitur-fitur pada Dart dalam menyediakan *framework* yang berperforma tinggi dan reaktif dalam membangun *user interface*.

Pada 4 Desember 2018, Flutter akhirnya berhasil mengembangkan framework versi pertamanya, Flutter 1.0, yang merupakan suatu pencapaian yang penting pada *framework* itu sendiri. Ini juga menunjukkan kesiapan dan kematangan oleh *framework* Flutter dalam pembuatan produk siap pakai. Sejak saat itu, Google selalu merawat dan memperbarui Flutter dengan merilis *update* dan fitur-fitur baru(Flutter (Software) - Wikipedia, n.d.).

Framework ini dijalankan pada sebuah *Virtual Machine* yang memiliki fitur *Hot Reload*, yang mampu menampilkan setiap perubahan pada kode dengan cepat. Fitur ini menjadi suatu nilai penting dalam pembangunan aplikasi dengan Flutter karena proses *recompiling* pada perubahan biasanya memakan waktu cukup lama dalam pemrograman bahasa lainnya, sehingga proses pembuatan aplikasi dapat dilaksanakan dengan lebih efisien dalam aspek waktu.

Flutter menggunakan widget dalam membangun sebuah tampilan pada aplikasi. Widget adalah sebuah komponen siap pakai pada Flutter yang digunakan untuk menampilkan komponen yang ingin dimunculkan pada suatu *user interface*. Setiap widget yang ada pada Flutter adalah sebuah class yang dapat dipanggil. Widget pada Flutter dapat berada di dalam widget yang lain, oleh karena itu ada istilah yang disebut dengan *tree of widgets* yang berisi dengan objek-objek widget di dalamnya.

Terdapat berbagai macam widget yang dapat digunakan pada Flutter, dan widget-widget tersebut dapat dikombinasikan satu sama lain dengan tujuan tertentu. Sebagai contoh, widget row, column dan stack dapat digunakan oleh *developer* aplikasi Flutter dalam menampung banyak widget di dalamnya, sekaligus digunakan untuk mengatur *layout* dan ukuran dari bagian yang ingin digunakan, sehingga tatanan *layout* dapat diatur dengan rapi. Contoh yang baru saja disebut memiliki properti children yang mampu menampung banyak widget di dalamnya,

dan ada pula beberapa widget yang hanya memiliki properti child, yang mana widget tersebut hanya bisa menampung satu widget.

Selain menggunakan widget yang tersedia, Flutter memungkinkan *developer* dalam membuat widget secara mandiri dengan menggabungkan widget-widget yang ada. Terdapat dua subclass utama dalam Flutter, yaitu stateful dan stateless widget. Bila widget yang ingin dibuat tidak memiliki properti yang berubah-ubah secara dinamis tergantung dari suatu kondisi, maka widget tersebut dapat dibuat dengan stateless widget. Sebaliknya, bila widget memiliki kondisi khusus dan kondisi tersebut mengubah suatu properti yang ada di dalamnya secara dinamis, maka widget tersebut dapat dibuat dengan stateful widget. Contoh dari kondisi khusus tersebut yaitu, bila terdapat teks yang tampilannya berubah bila suatu button diklik, maka *value* dari teks tersebut akan berubah.

2.12 Dart

Dart adalah bahasa pemrograman yang didesain oleh Lars Bak dan Kasper Lund, dan dikembangkan oleh Google pada 2011 (Wikipedia, 2023). Dart merupakan bahasa pemrograman yang bersifat *open-source* dan *object-oriented* dengan model *syntax* yang menyerupai bahasa pemrograman C. Tujuan utama bahasa pemrograman ini adalah untuk membuat *frontend user interface* pada *platform* Web, dan aplikasi *mobile*. Bahasa pemrograman ini juga mendukung banyak konsep dasar dalam bahasa pemrograman lain seperti class, interface, dan function.

Gambar 2. 25 Logo Dart

(Sumber: https://dart.dev/brand)

Sejarah dibangunnya Dart dimulai pada 11 Oktober 2011. Dart pertama kali dipublikasikan di GOTO conference Aarhus, Denmark, dengan Lars Bak dan Kasper Lund sebagai pengemuka proyek tersebut. Setelah itu, Dart 1.0 secara resmi

dirilis pada 14 November 2013. Namun Dart menemui beberapa masalah saat beberapa penggagas mengkritik rencana Dart dalam menyatukan Dart *Virtual Machine* dengan Chrome. Oleh karena itu, rencana tersebut dibatalkan pada Dart versi 1.9, dan mengganti acuan pengembangan pada compiling Dart ke JavaScript. Pada Agustus 2018, Dart 2.0 dirilis dan berhasil mengimplementasi permasalahan sebelumnya. Sejak saat itu, Dart terus berkembang pesat hingga Dart 2.6 memperkenalkan ekstensi baru yaitu dart2native, yang mampu memperluas jangkauan compile hingga platform seperti Linux, MacOS, dan Windows desktop. Hingga pada saat ini, Dart 3.0 memperkenalkan sistem baru yang dinamai sound null safety system, yang memungkinkan *developer* dalam membuat aplikasi dengan variable yang dapat mendeteksi value dari variable tersebut agar tidak bernilai null, sehingga peluang sebuah aplikasi untuk mendapatkan bug semakin sedikit (Dart (Programming Language) - Wikipedia, n.d.).

Dart memiliki banyak fitur yang menarik untuk digunakan. Contoh dari beberapa fitur ini yaitu Just-In-Time (JIT) compilation dan Ahead-Of-Time (AOT) compilation. Fitur ini memberikan keuntungan pada *developer*. Just-In-Time compilation memungkinkan *developer* untuk proses edit-refresh aplikasi dengan cepat dengan fitur hot reload, dan *developer* mampu memonitor debugging dan performance dari aplikasi Flutter secara langsung dengan fitur live metrics collections yang ada pada Dart DevTools. Sedangkan Ahead-Of-Time compilation ditujukan pada aplikasi pada tahap produksi atau aplikasi yang siap dipakai. Ahead-Of-Time compilation akan meng-compile aplikasi pada machine code platformnya masing-masing, menghasilkan aplikasi yang mampu dibuka dengan kecepatan yang konstan dan cepat pada startup (Dart Overview | Dart, n.d.).

2.13 Visual Studio Code

Visual Studio Code (VS Code) adalah sebuah *text editor open-source* yang dikembangkan oleh Microsoft. Teks editor ini ringan dan dapat digunakan untuk membuat banyak aplikasi dengan bahasa pemrograman yang berbeda. Teks editor ini dilengkapi dengan banyak fitur yang dapat membantu *developer* dalam melakukan *coding*, *debugging*, dan *version control*.

Gambar 2. 26 Logo Visual Studio Code

(Sumber: https://code.visualstudio.com/brand)

Visual Studio Code menyediakan *user interface* dan *environment* yang dapat dikustomisasi, dan ekstensi-ekstensi yang dapat membantu fungsionalitas dari teks editor tersebut. Selain itu, Visual Studio Code juga menyediakan fitur seperti integrasi dengan Git, *syntax highlighting*, *code snippet* untuk *code completion*, *debugging support*, dan *intelligent code suggestions*. Teks editor ini juga memiliki *built-in terminal* yang dapat membantu *developer* dalam mengeksekusi *command* secara langsung pada aplikasi.

Salah satu keunggulan pada Visual Studio Code adalah *extension marketplace*. *Extension marketplace* ini menyediakan *add-on* yang luas dalam menunjang produktivitas saat membangun sebuah aplikasi. *Add-on* tersebut dibuat oleh komunitas besar dan aktif dalam memperbarui extension tersebut.

Visual Studio Code tersedia pada platform Windows, macOS, dan Linux dan digunakan oleh banyak *developer* pada bahasa pemrograman yang berbeda-beda. Ini membuat Visual Studio Code menjadi *text editor* yang populer, berkat fleksibilitas, kemudahan, performa, serta *extension* yang tersedia, menjadikan Visual Studio Code pilihan utama dari banyak *developer*.

2.14 Penelitian Terdahulu

Penelitian serupa pernah dilakukan oleh sebelumnya dengan rentang waktu lima tahun terakhir. Penelitian pertama ini menggunakan CNN dengan model VGG16 untuk mengidentifikasi jenis 17 jenis ikan koi yang dilakukan oleh (Asmoro et al., 2024). Hasil dari penelitian ini berupa aplikasi yang mampu mengidentifikasi ikan koi dengan tingkat keakuratan 94,38% untuk pengujian

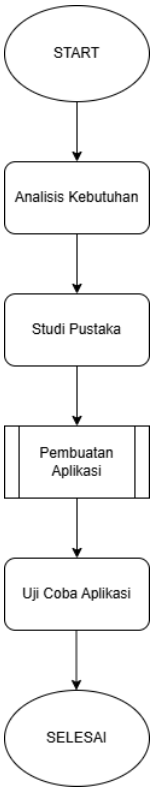
training dan 97,70% untuk pengujian testing dari total data 1700 data gambar yang digunakan. Penelitian kedua menggunakan CNN dengan model VGG16 namun ditambah dengan segmentasi pada setiap input data gambar dalam mengidentifikasi jenis ikan koi. Penelitian ini dilakukan oleh (Hermawan et al., 2021). Penelitian ini membandingkan hasil dari prediksi ikan koi dengan menggunakan segmentasi dan tidak menggunakan segmentasi. Hasil yang didapat yaitu tingkat akurasi pengenalan jenis ikan koi yang lebih akurat dengan segmentasi, mendapatkan nilai tingkat akurasi sebesar 0.901, dibandingkan dengan tanpa segmentasi dengan nilai tingkat akurasi sebesar 0.870.

BAB III

Analisis Kebutuhan Dan Perancangan Sistem

3.1 Alur Penelitian

Penelitian ini dilakukan untuk mengidentifikasi berbagai jenis ikan koi menggunakan *Deep Learning*. Alur dari penelitian ini dapat dilihat pada bagan berikut.



Gambar 3. 1 Diagram Alur Penelitian

3.2 Analisis Kebutuhan

Penelitian ini didukung oleh perangkat laptop ASUS Vivobook 14X dengan spesifikasi prosesor AMD Ryzen 5 5600H dan RAM 16GB. Penelitian ini difokuskan pada kinerja aplikasi yaitu untuk mengidentifikasi jenis ikan koi. Berikut adalah poin-poin yang diutamakan:

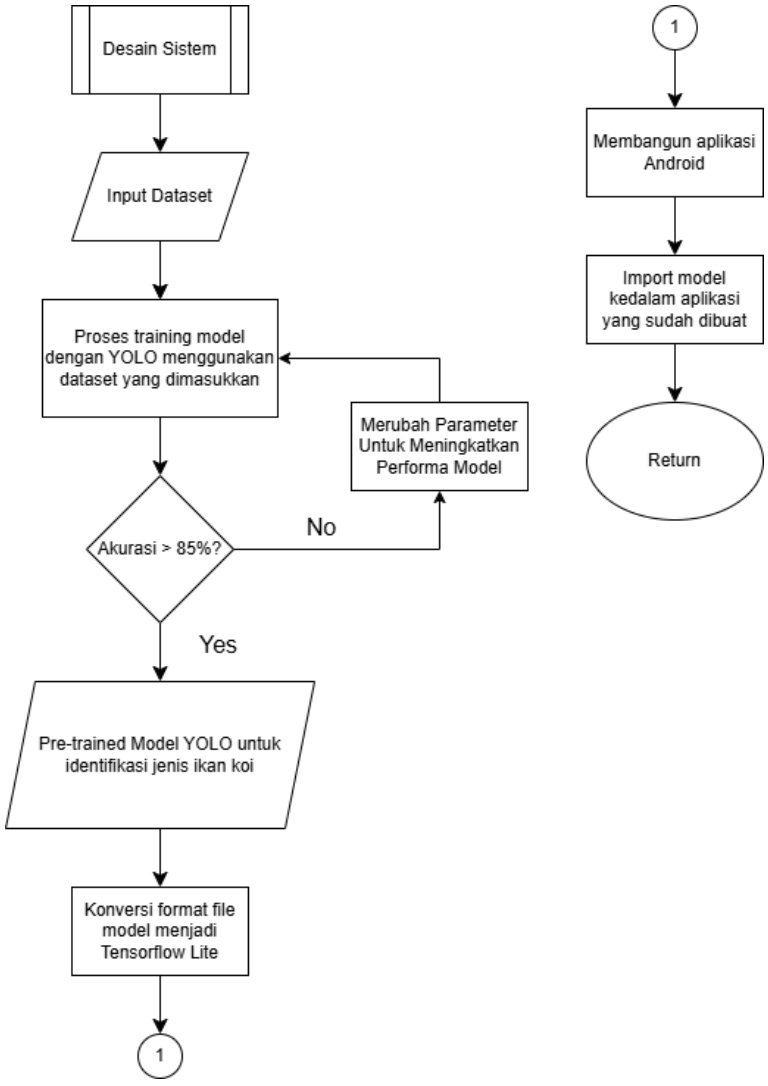
1. Proses identifikasi objek yang akan memberikan luaran berupa jenis ikan koi, dan tingkat keakuratan identifikasi, deskripsi singkat mengenai ikan koi yang ditujukan melalui kamera.
2. Terdapat halaman petunjuk cara pemakaian aplikasi ditujukan pada pengguna awam.

3.3 Studi Pustaka

Studi pustaka penelitian ini difokuskan pada metode *deep learning*, *convolutional neural network,* dan *platform* Flutter. Topik-topik tersebut merupakan hal penting dalam pengembangan aplikasi ini. Materi yang digunakan sebagai pembelajaran topik di atas didapat melalui berbagai sumber, antara lain jurnal-jurnal terkait, *website*, Youtube, buku, dan lain-lain.

3.4 Pembuatan Aplikasi

Proses pembuatan aplikasi ini meliputi alur perancangan sistem, sampai dengan perancangan desain aplikasi. Alur desain sistem pada penilitian ini dapat dilihat pada diagram berikut.



Gambar 3. 2 Diagram Predefine Process Untuk Desain Sistem

3.4.1 Pengumpulan Dataset

Dataset yang digunakan pada penelitian ini diambil dari internet, melalui *website* Kaggle dengan jumlah total 1062 gambar. Dataset yang digunakan terdiri dari *data test* dan *data train* yang nantinya akan digunakan untuk media pelatihan model pada aplikasi identifikasi objek Ikan Koi. Pada masing-masing *data test* dan *data train* terdapat 18 kategori jenis ikan koi. Terdapat kekurangan pada dataset ini, yaitu tidak meratanya jumlah gambar yang pada setiap jenis ikan koi dan beragamnya resolusi gambar yang tersedia, sehingga muncul kekhawatiran mengenai hasil training yang kurang baik pada beberapa jenis ikan koi. Jenis-jenis ikan koi dan jumlah ikan koi tersebut antara lain adalah:

1. Asagi, 70 gambar
2. Bekko, 22 gambar
3. Doitsu Koi, 22 gambar
4. Ghosiki, 28 gambar
5. Goromo, 39 gambar
6. Hikarimoyo, 62 gambar
7. Hikarimuji mono, 110 gambar
8. Hikariutsuri koi, 15 gambar
9. Kanako koi, 35 gambar
10. Kawarimono, 47 gambar
11. Kin-Ginrin, 57 gambar
12. Kohaku, 113 gambar
13. Sanke, 92 gambar
14. Showa, 110 gambar
15. Shusui, 65 gambar
16. Tancho, 84 gambar
17. Utsuri, 65 gambar
18. Yamato Nishiki, 26 gambar

Resolusi ukuran gambar terkecil untuk panjang yaitu berukuran 168 pixel dan tinggi berukuran 183 pixel.

3.4.2 Training Model

Pada tahap ini akan dilakukan proses pelatihan model menggunakan YOLO untuk mengenali objek berupa ikan koi. Versi model YOLO yang digunakan adalah YOLOv8m dengan epoch *training* sebanyak 100 dan akan diambil 640 gambar secara acak dari keseluruhan gambar pada dataset. Bila nilai akurasi dari model yang sudah dilatih cukup memuaskan, model ini akan diformat terlebih dahulu menjadi .tflite sebelum bisa masuk kedalam aplikasi Android.

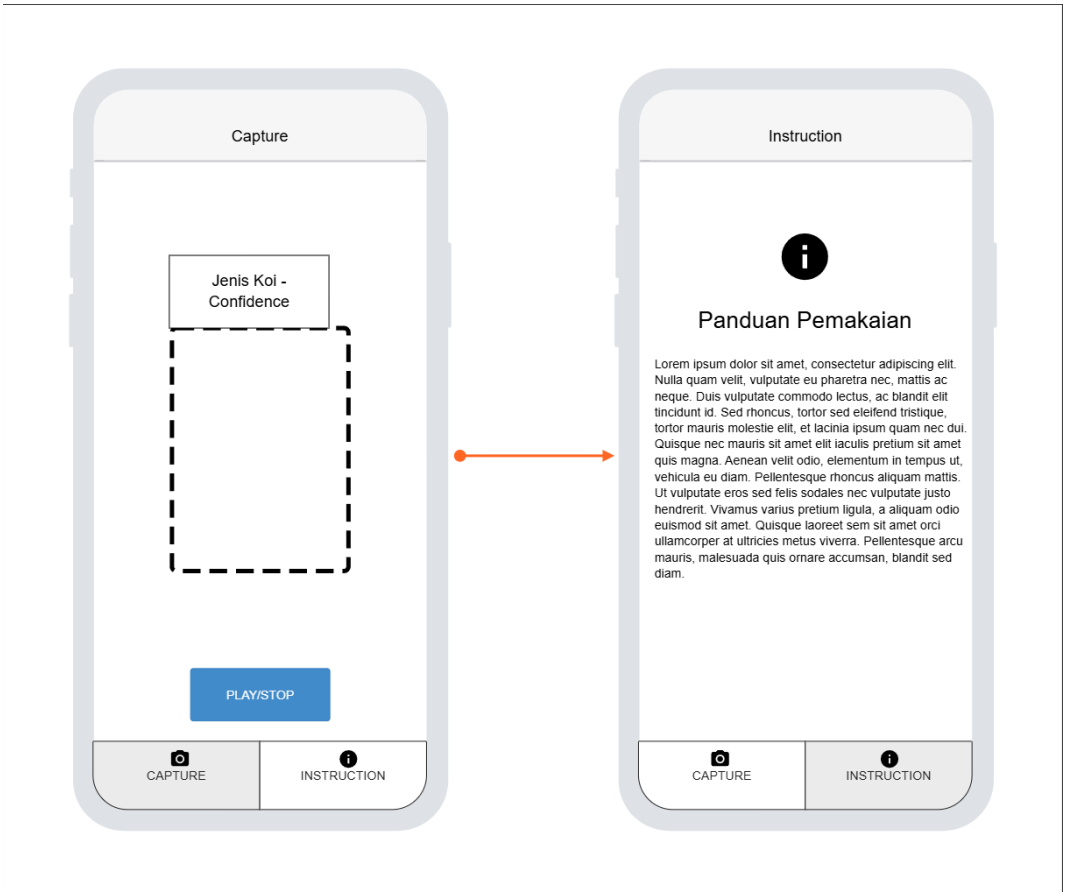
Untuk mengukur kemampuan model dalam mendeteksi objek berupa ikan koi, beberapa metriks evaluasi akan digunakan untuk melakukan pengukuran performanya. Pada proses *training* yang dilakukan menggunakan *library* dari Ultralytics, seperti yang dilakukan pada proses *training* ini, metriks evaluasi berupa akan diberikan pada folder hasil *training* setelah proses *training* berhasil dilakukan. Metriks evaluasi ini berupa grafik yang dapat diinterpretasi hasilnya nantinya untuk menggambarkan kemampuan model itu sendiri. Metriks-metriks evaluasi yang didapat setelah proses *training* ini meliputi *confusion matrix*, *box loss graph, class loss graph, distribution focal loss, precision-confidence curve, recall-confidence curve,* dan *F1-score curve*.

3.4.3 Desain UI Aplikasi Android

Aplikasi ini akan dijalankan pada perangkat Android, dan didukung oleh *framework* Flutter dalam pengembangannya. Sebelum membangun aplikasi ini, dibutuhkan desain rancangan aplikasi yang akan dibuat. Racangan aplikasi yang akan dibuat ini mempunyai 2 halaman utama, yaitu halaman pengenalan objek sebagai fitur utama, dan halaman petunjuk pemakaian aplikasi.

Pada Gambar 3.3, dapat dilihat rancangan aplikasi yang akan dibuat. Pada halaman *capture* terdapat opsi untuk menangkap gambar yang akan diidentifikasi dengan menekan tombol *play*. Pada saat pertama kali dibuka, aplikasi tidak akan secara langsung mengidentifikasi objek, maka pengguna dapat menggunakan tombol *play* ini untuk menangkap objek berupa ikan koi. Setelah pengguna menekan tombol *play* ini, logo pada tombol *play* ini akan berubah menjadi *stop*, dan akan mulai menangkap objek berupa ikan koi secara *real-time* melalui tampilan tangkapan kamera pada layar perangkat pengguna. Bila model yang tertanam pada

aplikasi mengidentifikasi objek berupa ikan koi, maka ikan koi tersebut nantinya akan dikelilingi oleh kotak *bounding box*, serta akan muncul prediksi jenis ikan koi dan nilai *confidence*-nya, terletak pada kiri atas dimana *bounding box* berada. Selanjutnya terdapat rancangan tampilan petunjuk penggunaan aplikasi, dilengkapi teks yang dapat dibaca pengguna. Pada rancangan aplikasi tersebut juga terdapat tombol navigasi pada pojok bawah layar yang dapat digunakan pengguna untuk mengganti tampilan dengan leluasa.



Gambar 3. 3 Rancangan Utama Aplikasi

BAB IV
Hasil dan Pembahasan

4.1 Hasil Training Model

Proses *training* model ini dilakukan pada Google Colab sebagai Jupyter *Notebook*. Proses ini dilakukan untuk menciptakan model yang mampu mengenali objek berupa ikan koi beserta jenis-jenisnya. Proses *training* ini dimulai dengan mengunggah dataset koi pada Google Drive, yang nantinya akan diintegrasi dengan Google Colab agar dapat diakses sebagai *local storage.* Potongan kode untuk menyambungkan Google Colab dengan Google Drive dapat dilihat pada gambar dibawah.

```
1  from google.colab import drive
2  drive.mount("/content/gdrive")
```

Gambar 4. 1 Potongan Kode Integrasi Google Colab dan Google Drive

Setelah *dataset* dimasukkan dan Google Colab sudah dihubungkan dengan Google Drive, setiap gambar pada setiap folder diberi harus diberi label sebagai tanda pada setiap jenis ikan koi agar model mampu mengenali setiap jenis ikan koi yang akan dilatih pada model. Potongan kode pembuatan label dapat dilihat pada gambar berikut.

```
1 import os
2 import glob
3 import random
4 import shutil
5
6 # Define paths
7 base_dir = '/path/to/data_images'
8 train_image_dir = os.path.join(base_dir, 'images/train')
9 val_image_dir = os.path.join(base_dir, 'images/val')
10 train_label_dir = os.path.join(base_dir, 'labels/train')
11 val_label_dir = os.path.join(base_dir, 'labels/val')
12
13 # Create directories if they don't exist
14 os.makedirs(train_image_dir, exist_ok=True)
15 os.makedirs(val_image_dir, exist_ok=True)
16 os.makedirs(train_label_dir, exist_ok=True)
17 os.makedirs(val_label_dir, exist_ok=True)
18
19 # Define class names and create a class-to-index mapping
20 class_names = ['Asagi', 'Bekko', 'Doitsu_koi', 'Ghosiki', 'Goromo',
'Hikarimoyo', 'Hikarimuji_mono',
```

```
21                'Hikariutsuri', 'Kanoko_koi', 'Kawarimono', 'Kin_Ginrin',
'Kohaku', 'Sanke', 'Showa',
22                'Shusui', 'Tancho', 'Utsuri', 'Yamato_Nishiki']
23 class_to_idx = {class_name: idx for idx, class_name in enumerate(class_names)}
24
25 # Function to move images and create labels
26 def process_images(split_ratio=0.8):
27     for class_name in class_names:
28         class_dir = os.path.join(base_dir, class_name)
29         if os.path.isdir(class_dir):
30             images = glob.glob(f"{class_dir}/*.*")
31             random.shuffle(images)
32             split_point = int(split_ratio * len(images))
33
34             train_images = images[:split_point]
35             val_images = images[split_point:]
36
37             for img_file in train_images:
38                 process_image(img_file, train_image_dir, train_label_dir,
class_name)
39
40             for img_file in val_images:
41                 process_image(img_file, val_image_dir, val_label_dir,
class_name)
42
43 def process_image(img_file, img_dir, label_dir, class_name):
44     img_name = os.path.basename(img_file)
45     label_file = os.path.join(label_dir, img_name.replace('.jpg',
'.txt').replace('.png', '.txt'))
46     new_img_file = os.path.join(img_dir, img_name)
47
48     # Move image to the target folder
49     shutil.copy(img_file, new_img_file)
50
51     # Write label file (assuming one bounding box covering the entire image)
52     with open(label_file, 'w') as f:
53         class_idx = class_to_idx[class_name]
54         f.write(f"{class_idx} 0.5 0.5 1.0 1.0\n")  # This is a placeholder,
adjust bounding box as needed
55
56 # Process the images and create labels
57 process_images()
58
59 print("Image processing and label generation complete.")
```

Gambar 4. 2 Potongan kode pembuatan label pada setiap gambar ikan koi

Setelah label terbuat untuk setiap gambar yang ada, proses pelatihan model dapat dimulai dengan meng-*install library* dari Ultralytics. Potongan kode untuk meng-*install library* Ultralytics dapat dilihat pada gambar dibawah.

```
1 pip install opencv-python ultralytics
```

Gambar 4. 3 Potongan kode untuk meng-*install* library Ultralytics.

Proses *training* model dilakukan dengan meng-*import* library YOLO, lalu memilih versi model YOLO yang ingin digunakan. Setelah itu, proses training dapat dijalankan dengan arahan dari file koi_fish.yaml yang berisi arahan berupa *class* yang digunakan, dan lokasi data input yang digunakan. Potongan kode proses tersebut dapat dilihat pada gambar berikut.

```
1 from ultralytics import YOLO
2
3 # Load a model
4 model = YOLO("yolov8n.yaml")
5
6 # Train the model
7 results = model.train(data="koi_fish.yaml", epochs=100,
imgsz=640)
```

Gambar 4. 4 Potongan kode proses training model

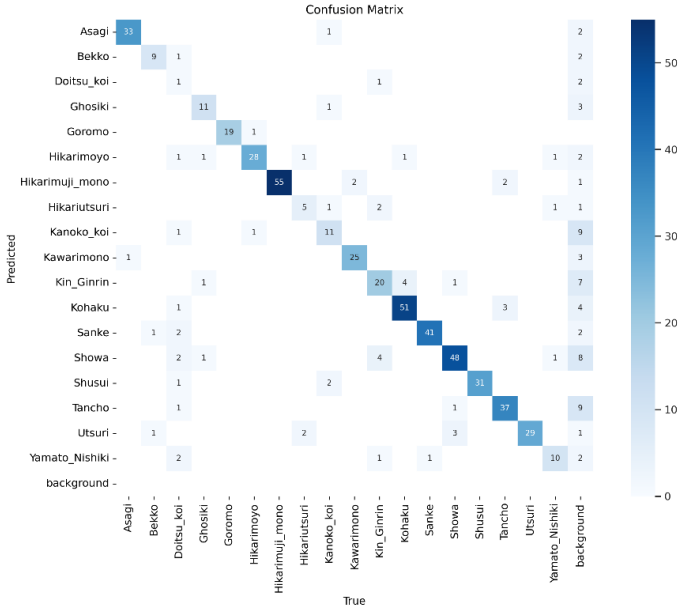
Proses *training* ini dilakukan menggunakan *optimizer* AdamW, *epoch* sebanyak 100 kali dan juga jumlah gambar sebanyak 640 secara acak, yang nantinya akan di-*split* nantinya dengan rasio 8:2 untuk *train* dan *test*. Proses *training* ini berlangsung selama 1 jam 28 menit untuk proses pelatihan, dan setelah selesai, didapat hasil berupa grafik-grafik metriks yang dapat diinterpretasi. Pertama-tama, hasil dari *confusion matrix* yang didapat pada proses training dapat dilihat pada gambar 4. 5.

Gambar 4.5 menunjukkan hasil *confusion matrix* yang didapat selama pelatihan. Hasil prediksi yang dijawab benar dapat dilihat pada *cell* diagonal dari atas kiri ke *cell* bawah kanan. Dari gambar yang tersedia, dapat ditarik kesimpulan nilai akurasi total dengan menghitung nilai prediksi yang benar dibagi dengan jumlah prediksi yang dilakukan, lalu didapatlah nilai sebagai berikut.

$$Akurasi = \frac{TP + TN}{TP + TN + FP + FN} \times 100\% \qquad (4-1)$$

$$Akurasi = \frac{464}{521} \times 100\%$$

$$Akurasi = 89{,}92\%$$



Gambar 4. 5 Hasil *Confusion Matrix* Pelatihan Model

Setelah dilakukan pengecekkan pada proses validasi model pada beberapa gambar yang sudah di-*split*, terdapat beberapa *error* pada pendeteksian. Ini menunjukkan bahwa model juga dapat menunjukkan hasil yang tidak selalu menampilkan hasil prediksi yang akurat. Hasil pelabelan pada validasi tersebut dapat dilihat pada gambar berikut.

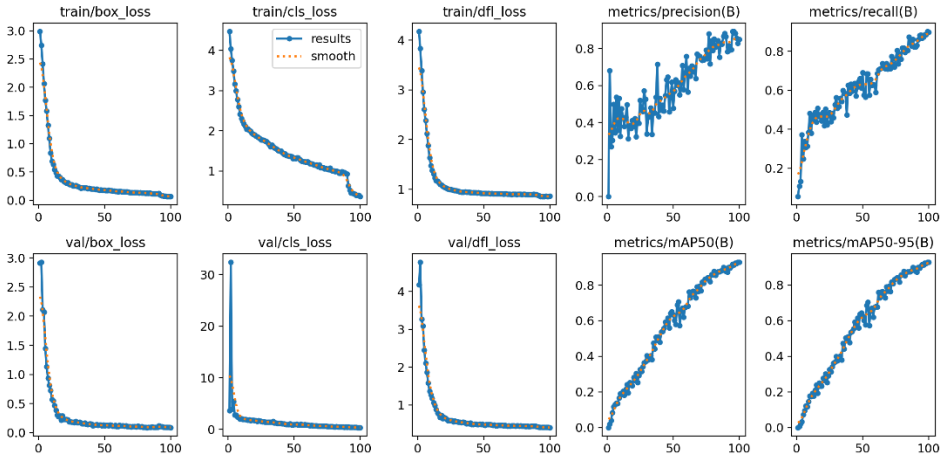
Gambar 4. 6 Proses Validasi dan Pelabelan Gambar 1

Gambar 4. 7 Proses Validasi dan Pelabelan Gambar 2

Gambar 4. 8 Proses Validasi dan Pelabelan Gambar 3

Adapun didapat metriks grafik yang menunjukkan nilai keakuratan model dalam mendeteksi objek, grafir tersebut dapat dilihat pada gambar 4. 9.

Dapat dilihat pada gambar 4.9, terdapat banyak metriks yang dapat diinterpretasi hasilnya, seperti *box loss*, *cls loss*, *dfl loss*, *precision*, *recall*, mAP50 dan mAP50-95.



Gambar 4. 9 Hasil Pelatihan Model Dalam Mendeteksi Objek

Box loss menunjukkan seberapa akurat sebuah model memprediksi *bounding box* pada data input, semakin kecil nilainya, maka semakin baik model dalam memprediksi objek pada gambar. Dari hasil grafik pada gambar 4. 9, *box loss* menunjukkan arah ke bawah seiring berjalannya proses pelatihan dari *epoch* tertentu, menunjukkan model mampu meletakkan *bounding box* dengan baik pada setiap objek, baik dari proses *train* maupun *test*.

Cls loss menunjukkan tingkat keakuratan suatu model dalam menentukan *class* pada sebuah input berupa gambar, semakin kecil nilainya, maka semakin baik model dalam memprediksi *class* yang benar. Dari hasil yang dapat dilihat pada gambar 4.9, *cls loss* menunjukkan arah ke bawah, menunjukkan performa model yang mampu memberikan *class* yang tepat pada input yang diberikan pada proses *train* dan *test*.

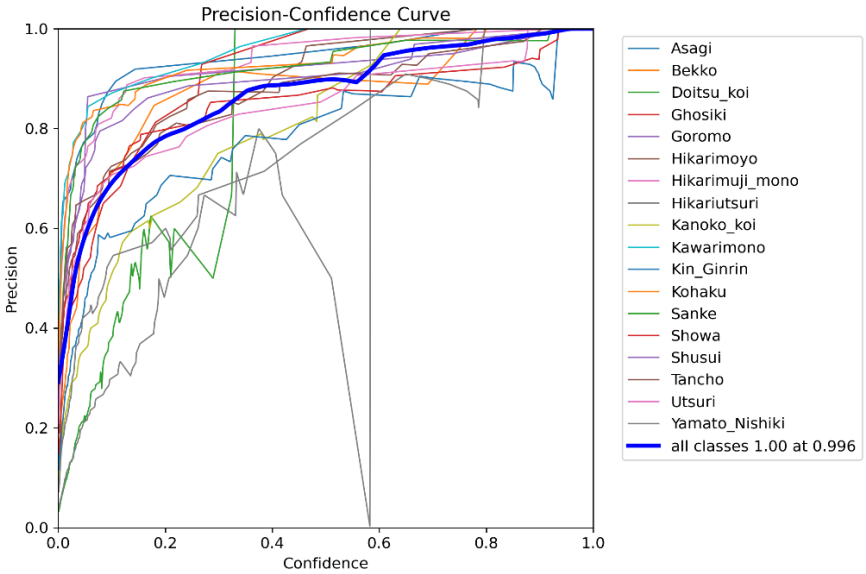
Dfl Loss atau "*distribution focal loss*"menunjukkan seberapa baik *micro-adjustment bounding box* agar posisi dapat terletak lebih akurat, semakin kecil nilainya, maka semakin baik peletakan *bounding box*-nya. *Dfl loss* adalah varian dari focal loss yang dirancang untuk meningkatkan performa model ketika data pelatihan tidak seimbang. Distribution focal loss digunakan khususnya untuk mengatasi ketidakseimbangan *class* yang muncul ketika melatih model pada dataset yang mengandung objek yang sangat tidak seimbang. Ketika terdapat sangat sedikit contoh dari suatu kelas objek dalam set pelatihan, jaringan model seringkali kesulitan untuk mempelajari cara mendeteksi objek tersebut dengan benar. Distribution focal loss bertujuan untuk mengatasi masalah ini dan memastikan bahwa model mampu mendeteksi objek-objek langka tersebut dengan baik (Losses and Their Weights in Yolov8, n.d.). Dari hasil yang ditunjukkan pada gambar 4. 9, didapat hasil bahwa model mampu mengatasi permasalahan peletakan *bounding box* pada permasalahan *focal loss* baik pada saat proses *training* maupun *test*.

Selain itu, terdapat grafik yang menunjukkan nilai *Precision* dan *Recall*. *Precision* akan mengukur proporsi dari nilai prediksi positif yang benar dijawab oleh model pada keseluruhan data yang diprediksi positif oleh model. Secara umum, *precision* adalah cara untuk menghitung akurasi dari seluruh prediksi positif yang dilakukan dari model. Rumus untuk menghitung *Precision* adalah sebagai berikut.

$$Precision = \frac{TP}{TP + FP} \qquad (4-2)$$

Nilai *precision* ini juga menggambarkan kemampuan model dalam memprediksi input yang dimasukkan dengan benar, bagaimana kinerja model yakin dalam memberikan label positif pada sebuah input. Dengan demikian, maka metriks ini dapat menyimpulkan seberapa yakin sebuah model mampu memberikan label yang tepat pada sebuah input. Grafik dari *precision* dapat dilihat pada gambar 4. 10.

Gambar 4. 10 menunjukkan interpretasi grafis mengenai nilai *precision* pada nilai *confidence* yang berbeda. Grafik menunjukkan arah perkembangan *precision* menuju ke atas, menunjukkan bahwa model mampu memprediksi, melabeli gambar dengan baik dan mampu mengatasi permasalahan prediksi *False Positive* (FP). Saat nilai *confidence* meningkat, model akan melabel positif suatu input bila model itu sendiri yakin bahwa gambar dan kelas yang diberikan tepat sasaran. Sehingga, dapat diartikan model mampu memprediksi jenis ikan koi dengan cukup tepat dari keseluruhan hasil prediksi. Terlepas dari itu, terdapat beberapa permasalahan pada pendeteksian, pada gambar 4. 10 terlihat grafik *precision* pada jenis ikan koi Tancho menurut drastis pada kisaran nilai *confidence* 50% sampai dengan 60%. Ini menggambarkan bahwa model belum dapat mengenali objek berupa ikan koi Tancho, didukung dengan nilai *confidence* yang rendah dalam mendeteksi ikan koi Tancho.



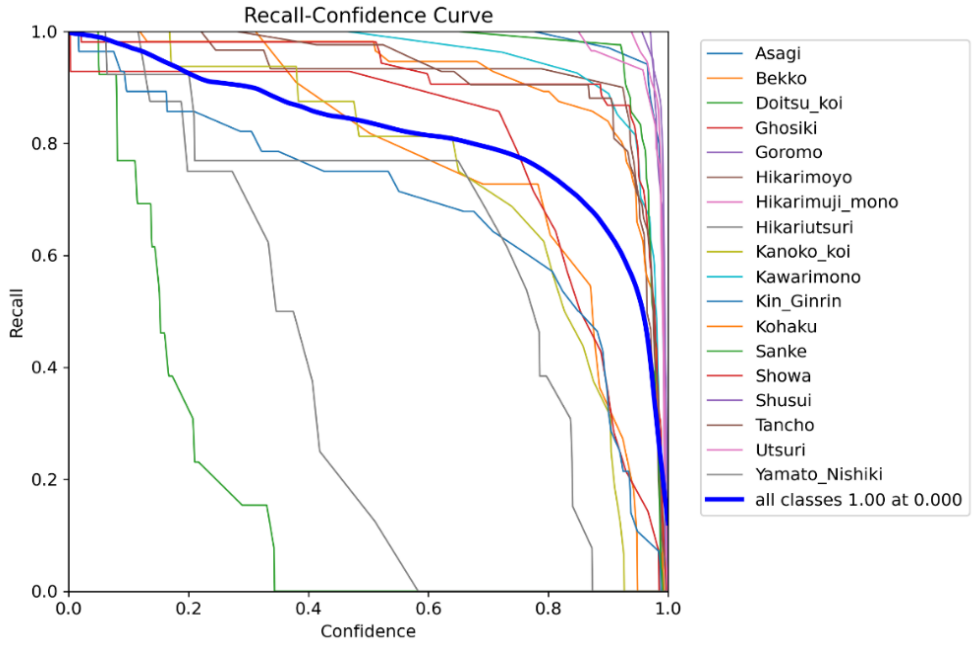
Gambar 4. 10 Grafik *Precision-Confidence Curve*

Recall menunjukkan proporsi hasil prediksi positif yang benar dijawab oleh model dengan keseluruhan data yang benar-benar positif. Adapun rumus untuk menghitung *Recall* adalah sebagai berikut.

$$Recall = \frac{TP}{TP + FN} \qquad (4-3)$$

Nilai *recall* ini juga menggambarkan kemampuan model dalam memberikan label kepada input dengan lebih berhati-hati, dan juga mengukur kemampuan model dalam mendeteksi seluruh *class* yang terlibat pada sebuah proyek (YOLO Performance Metrics - Ultralytics YOLO Docs, n.d.). Grafik dari *recall* dapat dilihat pada gambar 4. 10.

Gambar 4. 10 menunjukkan interpretasi grafis mengenai nilai *recall* pada nilai *confidence* yang berbeda. Saat nilai confidence meningkat, model akan semakin berhati-hati dalam memberi label positif pada input hanya bila input memiliki *confidence* yang tinggi. Sehingga, dapat diartikan model menjadi lebih selektif dalam mempertimbangkan prediksinya. Pada dasarnya, nilai *recall* akan menurun seiring bertambahnya nilai *confidence* karena *threshold* yang tinggi akan menyebabkan lebih sedikit prediksi positif, sehingga prediksi positif yang sebenarnya bisa terlewat (F1 Confidence, Precision-Recall Curve, Precision-Confidence Curve, Recall Confidence Curve and Confusion Matrix · Issue #7307 · Ultralytics/Ultralytics · GitHub, n.d.). Grafik pada gambar 4. 11 menunjukkan arah perkembangan menuju bawah, menunjukkan bahwa model mampu memprediksi input gambar dengan baik dan mampu mengatasi permasalahan prediksi *False Negative* (FN) dengan lebih berhati-hati dalam memberikan label, dan juga menunjukkan model mampu mengenali berbagai macam *class* yang sudah dipelajari pada proses *training*.

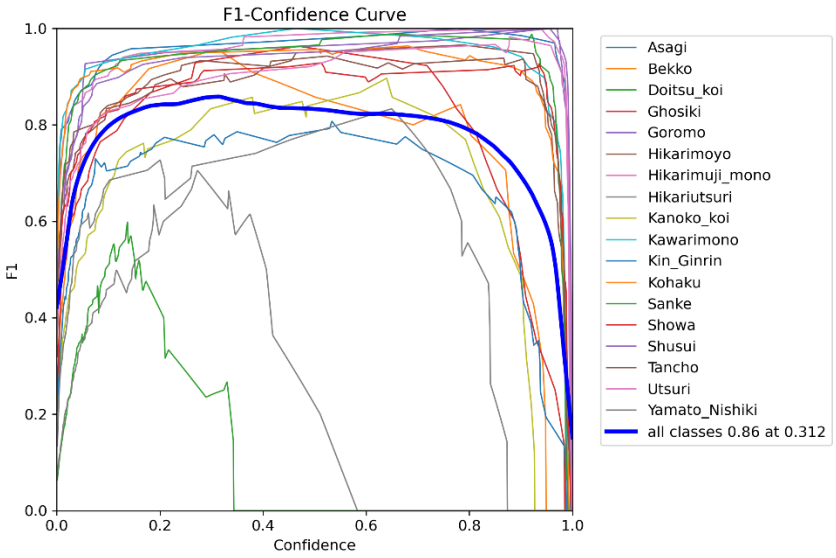


Gambar 4. 11 Grafik *Recall-Confidence Curve*

Kedua nilai dari *Recall* dan *Precision* ini memiliki nilai evaluasi gabungan yang memiliki nama F1-*score*. F1-*score* ini memiliki rentang nilai dari 0 sampai dengan 1 yang menandakan kinerja pada sebuah model dan nilai keselarasan oleh *Recall* dan *Precision*. Hasil F1 pada model ini dapat dilihat pada gambar 4. 12.

F1-*score* adalah nilai *mean harmonic* dari *precision* dan *recall*. Tujuan dari F1-*score* adalah untuk menunjukkan *confidence threshold* optimal untuk memaksimalkan hasil prediksi pada model (F1 Confidence, Precision-Recall Curve, Precision-Confidence Curve, Recall Confidence Curve and Confusion Matrix · Issue #7307 · Ultralytics/Ultralytics · GitHub, n.d.). Adapun rumus untuk menghitung F1-*score* sebagai berikut.

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \qquad (4-4)$$



Gambar 4. 12 Grafik F1-*score.*

Kembali pada gambar 4. 9, terdapat metriks *Mean Average Precision at IoU 0.50* (mAP50) dan *Mean Average Precision at IoU 0.50 to 0.95*(mAP50-95). Kedua metriks ini adalah metode evaluasi yang sering digunakan juga pada model CNN lainnya. mAP50 adalah rata-rata *precision* di semua kelas objek, dengan Intersection over Union (IoU) minimal 0,50 dianggap sebagai *true positive*, yang artinya sebuah prediksi yang dilakukan model dianggap benar jika nilai IoU nya melebihi dari 0.50 atau lebih dari *ground truth box*-nya. mAP50-95 di sisi lain sama seperti mAP50, namun nilai rata-rata ini diambil dari nilai IoU pada rentang 0.50 sampai dengan 0.95, dan dengan interval 0.05. Kedua metriks ini penting untuk digunakan dengan fungsi yang berbeda, dimana mAP50 digunakan untuk mengukur kemampuan model dalam mendeteksi objek dan memberi *bounding box* dengan akurasi yang longgar, sedangkan mAP50-95 digunakan untuk menghitung kemampuan model untuk mendeteksi objek pada *threshold-threshold* tertentu dari 0.50 sampai dengan 0.95 untuk memberikan evaluasi yang lebih kompleks (*What Is the Difference between MAP50 and MAP50-95? | 4 Answers from Research Papers*, n.d.). Berdasarkan hasil dari grafik pada gambar 4. 9, dapat disimpulkan bahwa model mampu mengenali obkek dengan baik dan memberikan label yang cukup tepat pada setiap inputnya.

4.2 Ekspor Model Menjadi TFLite

Setelah model berhasil di-*train*, model dapat diekspor menjadi format tflite agar dapat diimplementasikan ke dalam aplikasi Android yang dibuat. Lampiran *source code* pada proses konversi model menjadi tflite dapat dilihat pada gambar 4. 13 dan 4. 14.

Proses konversi ini dimulai dengan instalasi *package* sesuai dengan pada gambar 4. 13. Pada package tersebut, pengguna dapat langsung memasukkan model yang sudah dilatih, kemudian pengguna dapat menggunakan function bawaan dari *package* tersebut untuk diubah menjadi format tflite, sehingga didapat output berupa model tflite seperti pada gambar 4. 15.

```
1 pip install ultralytics
```

Gambar 4. 13 *Command Line* instalasi *package* dari Ultralytics

```
1 from ultralytics import YOLO
2 from google.colab import drive
3
4 drive.mount("/content/gdrive")
5
6 model = YOLO("/path/to/best.pt")
7
8 # Export the model to TFLite format
9 model.export(format="tflite")
```

Gambar 4. 14 *Source Code* konversi model menjadi format tflite

Gambar 4. 15 *Output* konversi model menjadi tflite

Setelah proses konversi selesai dilakukan, didapat file format tflite dengan ukuran 101MB. Dengan demikian, model ini dapat diproses untuk diimplementasi kedalam aplikasi Android.

4.3 Komponen Pendukung Pembuatan Aplikasi Android

Aplikasi Android ini dibangun dengan *Framework Flutter*. Flutter merupakan sebuah kerangka kerja yang mampu mendukung *developer* dalam mengembangkan sebuah aplikasi. Dengan dukungan pub.dev, yaitu situs kumpulan package yang dapat digunakan oleh pengembang aplikasi Flutter, *developer* dapat mengembangkan aplikasi yang diinginkan dengan menggunakan *package-package* yang tersedia untuk menunjang fungsionalitas aplikasi. *Package-package* yang digunakan antara lain adalah sebagai berikut.

4.3.1 Flutter Vision

Flutter Vision merupakan package Framework Flutter yang mampu mendukung developer dalam mengintegrasi model CNN yang sudah dikonversi menjadi tflite kepada proyek Flutter yang sedang dikerjakan. Flutter Vision mendukung beberapa versi model CNN antara lain YOLOv5, YOLOv8, dan Tesseract v5. Developer cukup meng-install package ini pada proyek dan meng-import package yang sudah di-install untuk memakai package ini.

4.3.2 Camera

Camera merupakan package framework Flutter yang dapat digunakan untuk mengakses fitur kamera pada smartphone yang ter-install dengan aplikasi berkaitan. Package ini berguna sebagai penengah aplikasi dengan perangkat yang ditujukan dalam mengakses kamera.

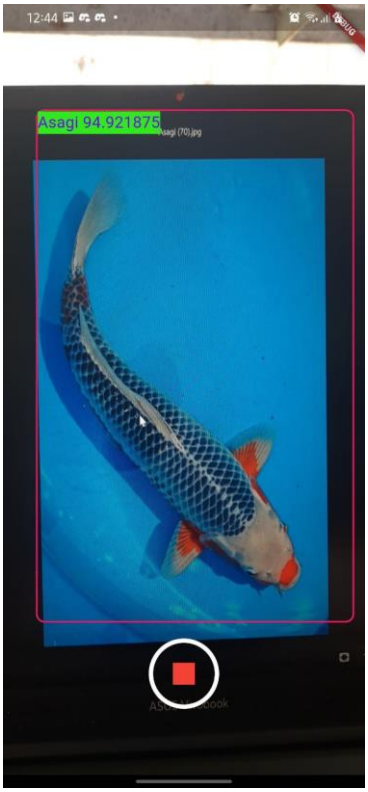
4.4 Pembuatan Aplikasi Android

Aplikasi Android yang dibuat ini cukup sederhana, didesain dengan 2 tampilan utama yang akan dimunculkan kepada pengguna. Tampilan ini dibikin serupa dengan *mock up* aplikasi yang sudah dibuat seperti pada gambar 3.3. Bahasa pemrogramman yang digunakan untuk membangun aplikasi ini adalah Dart, didukung dengan *framework* Flutter. Berikut tampilan yang sudah dibuat.

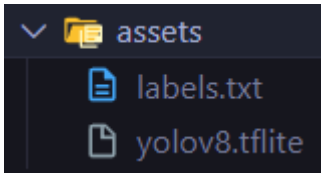
4.4.1 Halaman Capture

Halaman Pertama yaitu tampilan *Capture* yang dapat pengguna gunakan untuk memulai identifikasi objek, yaitu ikan koi itu sendiri. Halaman ini terdiri tombol tampilan kamera, dan juga tombol *Play/Stop* yang dapat digunakan pengguna untuk memulai dan berhenti mengidentifikasi objek pada kamera. Tampilan pada halaman *capture* dapat dilihat pada gambar 4. 16.

Pada halaman ini, kedua *package* yaitu Flutter Vision dan Camera digunakan untuk menunjang fungsionalitas aplikasi. *Package* Flutter Vision ini digunakan untuk mengintegrasi aplikasi dengan model yang dimasukkan ke dalam aplikasi. Penggunaan *package* ini dimulai dengan memasukkan model ke dalam folder *assets* proyek dan melalukan konfigurasi proyek agar aplikasi dapat mengenali *assets* yang dimasukkan. Proses memasukkan model dapat dilihat pada gambar dan potongan kode dibawah berikut.



Gambar 4. 16 Tampilan Halaman Capture Pada Aplikasi



Gambar 4. 17 Memasukkan model dan label pada aplikasi

```
flutter:
  assets:
    - assets/labels.txt
    - assets/yolov8.tflite
```

Gambar 4. 18 Konfigurasi proyek terhadap assets

Setelah konfigurasi proyek selesai, model dapat dimuat dengan *package* Flutter Vision dengan potongan kode berikut.

```
Future<void> loadYoloModel() async {
    await vision.loadYoloModel(
        labels: 'assets/labels.txt',
        modelPath: 'assets/yolov8.tflite',
        modelVersion: "yolov8",
        numThreads: 1,
        useGpu: true);
    setState(() {
      isLoaded = true;
    });
}
```

Gambar 4. 19 Memuat model pada Flutter Vision

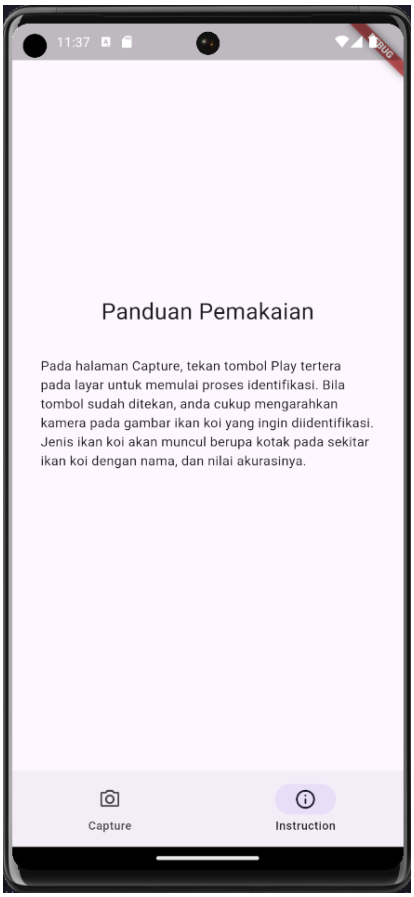
Setelah model dimuat, model ini dapat diintegrasi dengan *package camera* agar kamera pada perangkat Android dapat menangkap gambar dan melakukan identifikasi pada objek secara *real-time* bila tombol *play* ditekan, dan berhenti bila tombol *play* ditekan sekali lagi untuk stop. Pada langkah ini, aplikasi sudah dapat digunakan untuk mengidentifikasi objek berupa ikan koi. Potongan kode untuk implementasi ini dapat dilihat pada gambar 4. 21. Pada potongan kode ini juga, seluruh proses memuat model didahulukan saat aplikasi diluncurkan agar saat kamera siap dipakai, fitur untuk pengenalan objek juga dapat langsung digunakan.

4.4.2 Halaman Instruction

Lalu halaman *intstruction* berisi tentang cara pemakaian aplikasi. Pada halaman ini hanya terdapat panduan singkat tentang cara penggunaan aplikasi pada pengguna. Tampilan pada halaman ini dapat pada gambar 4. 19.

4.5 Pengujian Black Box

Tahap pengujian *Black Box* ini adalah tahap yang bertujuan untuk menguji fitur yang terdapat pada aplikasi. Pengujian *Black Box* pada aplikasi ini dapat dilihat pada tabel 4. 1.



Gambar 4. 20 Tampilan Halaman Instruction Pada Aplikasi.

```
1 @override
2   void initState() {
3     super.initState();
4     init();
5   }
```

```
6   init() async {
7     camerass = await availableCameras();
8     vision = FlutterVision();
9     controller = CameraController(camerass[0],
ResolutionPreset.high);
10     controller.initialize().then((value) {
11       loadYoloModel().then((value) {
12         setState(() {
13           isLoaded = true;
14           isDetecting = false;
15           yoloResults = [];
16         });
17       });
18     });
19   }
20   @override
21   void dispose() async {
22     super.dispose();
23     controller.dispose();
24     await vision.closeYoloModel();
25   }
```

Gambar 4. 21 Integrasi kamera dengan model untuk pengenalan objek.

Tabel 4. 1 Tabel Uji Coba Black Box

No.	Komponen Uji	Hasil Yang Diharapkan	Keterangan
1	*Navigasi Aplikasi*	Tombol navigasi aplikasi yang digunakan berpindah ke tampilan *capture* dan *instruction* dapat	Sukses

		digunakan dengan baik.	
2	Kamera pada halaman *capture*	Aplikasi mampu menampilkan fitur utama yaitu kamera	Sukses
3	Integrasi kamera dengan model .tflite	Kamera yang diimplementasikan pada aplikasi dapat diintegrasi dan memuat model .tflite dan mampu mengidentifikasi objek berupa koi.	Sukses
4	Tampilan hasil prediksi dari model	Model yang tertanam melalui kamera pada aplikasi mampu mengenali objek, dengan menampilkan jenis ikan koi yang diidentifikasi dengan nilai *confidence*-nya.	Sukses
5	Tombol *play/stop* pada halaman *capture*	Tombol *play/stop* yang berfungsi dengan baik dalam memulai proses identifikasi objek bila tombol *play* disentuh, dan akan berhenting mengidentifikasi	Sukses

		objek bila tombol *stop* disentuh.	
3	Halaman *Instruction*	Aplikasi menampilkan penjelasan singkat tentang cara memakai aplikasi.	Sukses

24.6 Pengujian Aplikasi

Pada tahap ini, aplikasi yang sudah berhasil dibuat akan diuji hasil dari pendeteksiannya, untuk mengukur kemampuan model yang sudah diimplementasi pada aplikasi dapat berjalan dengan baik atau tidak. Proses ini akan dilakukan dengan melakukan tes aplikasi dan diarahkan pada objek berupa ikan koi pada setiap jenis yang ada. Melalui tes ini, dapat disimpulkan kemampuan dari aplikasi dalam mengenali setiap *class* yang sudah dipelajari melalui proses training. Hasil dari pengujian aplikasi dapat dilihat pada tabel 4. 2.

Tabel 4. 2 Tabel Uji Coba Aplikasi Pada Setiap Jenis Ikan Koi

No.	Jenis Ikan Koi	Jumlah Gambar yang Diuji	Jumlah Prediksi yang Tepat
1	Asagi	5	3
2	Bekko	5	2
3	Doitsu Koi	5	2
4	Ghosiki	5	3
5	Goromo	5	3
6	Hikarimoyo	5	4
7	Hikarimuji Mono	5	4
8	Hikariutsuri	5	1
9	Kanoko Koi	5	4
10	Kawarimono	5	2
11	Kin Ginrin	5	4

12	Kohaku	5	4
13	Sanke	5	2
14	Showa	5	3
15	Shusui	5	3
16	Tancho	5	2
17	Utsuri	5	4
18	Yamato Nishiki	5	2

Dapat dilihat pada pada tabel diatas, hasil terlihat pada Hikariutsuri dengan perolehan nilai predisi salah terbanyak. Hal ini diasumsikan karena jumlah dataset pada jenis ikan koi Hikariutsuri yang sedikit, dan juga dikarenakan jumlah data yang digunakan pada proses *training* tidak memakai keseluruhan data yang ada pada dataset, sehingga model memiliki permasalahan dalam memprediksi jenis ikan koi tersebut.

BAB V
Simpulan dan Saran

5.1 Simpulan

Berdasarkan hasil pengembangan aplikasi ini, yang dimulai dari proses pelatihan model sampai dengan implementasi pada aplikasi Android, didapat simpulan sebagai berikut.

1. Proses pelatihan model menggunakan YOLOv8 dengan 100 epoch dan Optimizer AdamW, dan menggunakan 640 gambar yang diambil secara acak dari *dataset* ini berhasil berjalan dengan baik dengan nilai akurasi sebesar 89.92% pada proses training.
2. Proses validasi menampilkan hasil pelabelan pada gambar yang tidak seluruhnya tepat, menunjukkan terdapat kekurangan pada model yang dibuat.
3. Proses konversi model menjadi tflite agar dapat diimplementasi kepada aplikasi berhasil dijalankan, dan model tflite ini berfungsi dengan baik setelah diimplementasikan kepada aplikasi Android.
4. Setelah aplikasi berhasil dibuat, sering terjadi *miss-classification* yang menunjukkan bahwa aplikasi masih memiliki banyak kekurangan dalam mendeteksi ikan koi, walau nilai akurasinya yang terhitung tinggi pada saat proses *training*.

5.2 Saran

Berdasarkan simpulan di atas, didapat pula beberapa saran dan perbaikan dalam pelatihan ini sebagai berikut.

1. Diperlukan jumlah dataset yang lebih besar yang diharapkan mampu meningkatkan kinerja model dalam memprediksi jenis ikan koi agar semakin akurat.
2. Melakukan proses training dengan dataset yang tersedia, tanpa perlu dipotong jumlah yang digunakan untuk melakukan proses training sehingga model dapat mengenali objek dengan maksimal.

3. Melakukan modifikasi parameter pada saat pelatihan model yang diharapkan mampu meningkatkan kinerja model dalam mengidentifikasi objek.
4. Pembenahan UI yang lebih agar terlihat lebih menarik dan dapat mengikuti perkembangan aplikasi jaman sekarang.

Daftar Pustaka

Adhiat, A. (n.d.). *67% Penduduk Indonesia Punya Handphone pada 2022, Ini Sebarannya*. Retrieved June 24, 2023, from https://databoks.katadata.co.id/datapublish/2023/03/08/67-penduduk-indonesia-punya-handphone-pada-2022-ini-sebarannya

Asmoro, S. S., Amrulloh, M. F., Toybah, Moch. A., & Saputra, M. A. (2024). Rancang Bangun Aplikasi Mobile Untuk Klasifikasi Jenis Ikan Koi Menggunakan Algoritma Convolutional Neural Network. *Seminar Nasional Teknologi & Sains*, *3*(1), 270–277. https://doi.org/10.29407/STAINS.V3I1.4312

Dart overview | Dart. (n.d.). Retrieved July 20, 2023, from https://dart.dev/overview

Dart (programming language) - Wikipedia. (n.d.). Retrieved July 20, 2023, from https://en.wikipedia.org/wiki/Dart_(programming_language)

F1 confidence, Precision-Recall curve, Precision-Confidence curve, Recall Confidence curve and Confusion matrix · Issue #7307 · ultralytics/ultralytics · GitHub. (n.d.). Retrieved August 18, 2024, from https://github.com/ultralytics/ultralytics/issues/7307

Flutter (software) - Wikipedia. (n.d.). Retrieved July 4, 2023, from https://en.wikipedia.org/wiki/Flutter_(software)

Hermawan, A., Zaeni, I., Wibawa, A., Gunawan, G., Kristian, Y., & Darmawan, S. (2021). Pengenalan Varietas Ikan Koi Berdasarkan Foto Menggunakan Simple Linear Iterative Clustering Superpixel Segmentation dan Convolutional Neural. *Jurnal Inovasi Teknologi Dan Edukasi Teknik*, *1*(11), 806–814. https://doi.org/10.17977/UM068V1I112021P806-814

Lengkap, Ini 15 Jenis Ikan Koi Beserta Arti Coraknya - Hewania. (n.d.). Retrieved August 17, 2024, from https://hewania.com/15-jenis-ikan-koi-beserta-arti-corak-ikan-koi/4237/?srsltid=AfmBOoo1qmm7RXhyxMs9CDNK3Rq06ccsz2A-FPts8RYzhVtwEBEL2hM3

Loshchilov, I., & Hutter, F. (2017). Decoupled Weight Decay Regularization. *7th International Conference on Learning Representations, ICLR 2019*. https://arxiv.org/abs/1711.05101v3

Losses and Their Weights in Yolov8. (n.d.). Retrieved August 17, 2024, from https://www.linkedin.com/pulse/losses-weights-yolov8-dsaisolutions-x1ggf

Nugroho, P. A., Fenriana, I., & Arijanto, R. (2020). IMPLEMENTASI DEEP LEARNING MENGGUNAKAN CONVOLUTIONAL NEURAL NETWORK (CNN) PADA EKSPRESI MANUSIA. *ALGOR*, *2*(1), 12–20. https://jurnal.ubd.ac.id/index.php/algor/article/view/441

Vailshery, L. S. (n.d.). *Cross-platform mobile frameworks used by global developers 2022 | Statista*. Retrieved June 24, 2023, from https://www.statista.com/statistics/869224/worldwide-software-developer-working-hours/#statisticContainer

What is the difference between mAP50 and mAP50-95? | 4 Answers from Research papers. (n.d.). Retrieved August 18, 2024, from https://typeset.io/questions/what-is-the-difference-between-map50-and-map50-95-5asxtut1cj

YOLO Performance Metrics - Ultralytics YOLO Docs. (n.d.). Retrieved August 18, 2024, from https://docs.ultralytics.com/guides/yolo-performance-metrics/#object-detection-metrics

YOLOv8 for Object Detection Explained [Practical Example] | by Encord | Encord | Medium. (n.d.). Retrieved August 17, 2024, from https://medium.com/cord-tech/yolov8-for-object-detection-explained-practical-example-23920f77f66a