

**IMPLEMENTASI ALGORITMA YOLO V5 UNTUK
DETEKSI DAN KLASIFIKASI TINGKAT
KUALITAS TANAMAN PADI**

TUGAS AKHIR

Sebagai salah satu syarat untuk menyelesaikan
Program Diploma III Teknik Informatika
Politeknik Negeri Indramayu



Oleh:

RENOL NINDI KARA NATASASMITA

NIM 1903027

**PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
POLITEKNIK NEGERI INDRAMAYU
AGUSTUS 2021**

HALAMAN PENGESAHAN

Tugas Akhir ini diajukan oleh:

Nama : RENOL NINDI KARA NATASASMITA
NIM : 1903027
Program Studi : Diploma III Teknik Informatika
Judul : Implementasi Algoritma YOLO V5 Untuk Deteksi dan Klasifikasi Tingkat Kualitas Tanaman Padi
Pembimbing : 1. A. Sumarudin, S.Pd., M.T., M.Sc.
NIP 19861010 201903 1 014
2. Adi Suheryadi, S.ST., M.Kom.
NIP 19900322 201903 1 007

Telah berhasil dipertahankan dihadapan dewan penguji pada tanggal Agustus 2022 dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Ahli Madya Program Studi Diploma III Teknik Informatika, Jurusan Teknik Informatika, Politeknik Negeri Indramayu.

DEWAN PENGUJI

Ketua Penguji :
NIP
Anggota :
Penguji I NIP
Anggota :
Penguji II NIP

Indramayu, September 2022

Ketua Jurusan Teknik Informatika

Iryanto, S.Si., M.Si.

NIP 19900801 201903 1 014

HALAMAN PERNYATAAN KEASLIAN

Saya menyatakan dengan sesungguhnya bahwa Tugas Akhir ini adalah asli hasil karya saya sendiri, dan sepanjang pengetahuan saya tidak terdapat karya atau pendapat yang pernah atau ditulis atau dipublikasikan oleh orang lain, kecuali yang secara tertulis disebutkan sumbernya dalam naskah dan dalam daftar pustaka.

Indramayu, Agustus 2022

Yang Menyatakan,

RENOL NINDI KARA N.

NIM 1903027

ABSTRAK

Indonesia merupakan salah satu negara agraria terbesar di dunia, karena sebagian penduduk Indonesia memiliki mata pencaharian sebagai petani atau bercocok tanam. Tanaman Padi merupakan komoditas tanaman pangan yang penting di Indonesia, selain itu Tanaman Padi juga termasuk tanaman pertanian yang berasal dari benua Asia termasuk Indonesia. Tanaman Padi biasanya diolah menjadi beras sebagai makanan pokok bagi sebagian besar penduduk di Indonesia, maka dari itu Tanaman Padi sangat perlu diperhatikan kualitasnya. BPS mencatat, pada tahun 2019 struktur lapangan pekerjaan utama masyarakat Indonesia paling besar di bidang pertanian dengan persentase 27,33%. Antusiasme masyarakat untuk bergelut di bidang pertanian harusnya membawa kesejahteraan bagi para petani, namun hal tersebut justru menimbulkan kesenjangan sebab angka kemiskinan di sektor pertanian paling tinggi. Pengelolaan lahan yang kurang ideal serta kurangnya sumber daya manusia untuk mengelola lahan menjadi penyebab berkurangnya pendapatan para petani. Perkembangan teknologi *Artificial Intelligence* khususnya di bidang *Deep Learning* dan juga teknologi UAV atau *drone* dapat menjadi solusi untuk menyelesaikan permasalahan yang ada. Pendeteksian tingkat kualitas tanaman padi dapat dilakukan menggunakan metode *object detection*, salah satunya yaitu menggunakan algoritma YOLO. YOLO merupakan salah satu algoritma untuk pendeteksian objek, dengan menggunakan algoritma ini, petani dapat mengetahui tingkat kualitas tanaman padi dari masing-masing lahan. Agar algoritma YOLO ini dapat mendeteksi tingkat kualitas tanaman padi dari masing-masing lahan, dibutuhkan *datasets* yang diambil melalui citra udara atau teknologi *drone*, *datasets* tersebut akan digunakan untuk melatih model agar dapat mendeteksi tingkat kualitas tanaman padi. Algoritma YOLO dapat mendeteksi tingkat kualitas tanaman padi berdasarkan *point of interest* pada gambar yang diunggah oleh petani ke *server*. Gambar yang telah diolah menggunakan algoritma YOLO akan menghasilkan *output* berupa *bounding boxes* dan *confidence score* untuk masing-masing objek yang terdeteksi. Hasil dari mAP@0,5 (*mean average precision*) adalah 97,98%.

Kata kunci : *Artificial Intelligence, Deep Learning, YOLO Algorithm, Object Detection, Computer Vision.*

ABSTRACT

Indonesia is one of the largest agrarian countries in the world, because most of Indonesia's population has a livelihood as farmers or farming. Rice is an important food crop commodity in Indonesia, besides that, rice plants are also agricultural crops originating from the Asian continent, including Indonesia. Rice plants are usually processed into rice as a staple food for most of the population in Indonesia, therefore the quality of rice plants needs to be considered. BPS noted that in 2019 the largest employment structure for the Indonesian people was in agriculture with a percentage of 27.33%. The enthusiasm of the people to work in the agricultural sector should bring prosperity to the farmers, but this actually creates a gap because the poverty rate in the agricultural sector is the highest. Land management that is less than ideal and the lack of human resources to manage land are the causes of reduced income for farmers. The development of Artificial Intelligence technology, especially in the field of Deep Learning and also UAV or drone technology can be a solution to solve existing problems. Detection of the quality level of rice plants can be done using object detection methods, one of which is using the YOLO algorithm. YOLO is one of the algorithms for object detection, by using this algorithm, farmers can find out the quality level of rice plants from each land. In order for the YOLO algorithm to detect the quality level of rice plants from each field, datasets are needed that are taken through aerial imagery or drone technology, these datasets will be used to train the model in order to detect the quality level of rice plants. The YOLO algorithm can detect the quality level of rice plants based on the point of interest in the image uploaded by the farmer to the server. Images that have been processed using the YOLO algorithm will produce output in the form of bounding boxes and confidence scores for each detected object. The result of mAP@0.5 (mean average precision) is 97.98%.

Keywords : Artificial Intelligence, Deep Learning, YOLO Algorithm, Object Detection, Computer Vision

MOTTO

“Berjuanglah untuk hidup, karena One Piece belum tamat”

- Renol

KATA PENGANTAR

Alhamdulillah Rabbil'alamin, Puji dan syukur kehadiran Allah Subhanahu Wata'ala, yang telah melimpahkan rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan tugas akhir ini dengan baik. Adapun judul tugas akhir ini yaitu “Implementasi Algoritma YOLO V5 Untuk Deteksi dan Klasifikasi Tingkat Kualitas Tanaman Padi”.

Tugas akhir ini merupakan salah satu tugas yang wajib ditempuh oleh mahasiswa tingkat akhir sebagai persyaratan utama untuk dapat dinyatakan lulus sebagai Ahli Madya Diploma 3.

Selanjutnya, pengerjaan tugas akhir ini dapat terlaksana dengan lancar berkat kerjasama, bantuan, arahan dan dukungan dari berbagai pihak baik secara langsung maupun tidak langsung. Oleh karena itu, penulis dengan segenap hati mengucapkan banyak terima kasih kepada:

1. Orang tua penulis dan keluarga yang selalu mendoakan dan memotivasi penulis selama pengerjaan tugas akhir ini.
2. Bapak Casiman Sukardi, S.T., M.T. selaku Direktur Politeknik Negeri Indramayu.
3. Bapak Iryanto, S.Si., M.Si. selaku Kepala Jurusan Teknik Informatika.
4. Bapak A. Sumarudin, S.Pd., M.T., M.Sc. selaku Dosen Pembimbing I.
5. Bapak Adi Suheryadi, S.ST., M.Kom. selaku Dosen Pembimbing II.
6. Seluruh teman-teman Teknik Informatika tahun angkatan 2019 khususnya yang telah banyak membantu penulis selama penyelesaian tugas akhir ini dan berjuang bersama dalam menuntut ilmu di Politeknik Negeri Indramayu.
7. Semua pihak yang tidak dapat penulis sebutkan satu persatu yang telah terlibat dalam penyelesaian pembuatan tugas akhir ini.

Penulis menyadari masih terdapat banyak sekali kekurangan serta kesalahan dalam penulisan laporan tugas akhir ini baik dari segi pembahasan, metode, analisa maupun implementasi dalam aplikasi. Penulis menerima dengan senang hati apabila pembaca memberikan kritik dan saran yang membangun sehingga akan menjadi bekal penulis guna menyempurnakan penulisan di kemudian hari.

Indramayu, ... Agustus 2022

Penulis

DAFTAR ISI

Halaman

HALAMAN PENGESAHAN.....	iii
HALAMAN PERNYATAAN KEASLIAN.....	iv
ABSTRAK	v
ABSTRACT	vi
MOTTO	vii
KATA PENGANTAR.....	viii
DAFTAR ISI.....	x
DAFTAR TABEL	xiii
DAFTAR GAMBAR.....	xiv
DAFTAR LAMPIRAN.....	xvi
BAB 1 PENDAHULUAN.....	1
1.1. Latar Belakang	1
1.2. Rumusan Masalah	2
1.3. Batasan Masalah.....	2
1.4. Tujuan Penelitian	3
1.5. Manfaat Penelitian	3
1.6. Sistematika Penulisan Laporan	3
BAB 2 LANDASAN TEORI.....	5
2.1. <i>Object Detection</i>	5
2.2. Klasifikasi Citra	5
2.3. <i>Deep Learning</i>	6
2.4. <i>Neural Network</i>	7
2.5. <i>Convolutional Neural Network</i>	8
2.5.1. <i>Extraction Feature Layer (Feature Learning)</i>	9
2.5.2. <i>Classification Layer</i>	10
2.6. <i>Convolutional Layer</i>	10
2.7. <i>Pooling Layer</i>	11

2.8.	<i>Fully-Connected Layer</i>	12
2.9.	<i>Dropout Regulation</i>	13
2.10.	<i>Softmax Classifier</i>	14
2.11.	Algoritma YOLO	15
2.12.	Algoritma SSDMobileNetV2.....	17
2.13.	Python	18
2.14.	Golang	19
2.15.	Jupyter <i>Notebook</i>	19
2.16.	Google Colab	19
2.17.	Augmentasi Data.....	20
2.17.1.	Rotasi	20
2.17.2.	Flipping	20
2.17.3.	Cropping.....	21
2.18.	<i>Image Labeling</i>	22
BAB 3 METODOLOGI PELAKSANAAN.....		23
3.1.	Metodologi Pelaksanaan	23
3.2.	Pengumpulan Data	24
3.3.	Analisis Kebutuhan Sistem	26
3.3.1.	Kebutuhan <i>Hardware</i>	26
3.3.2.	Kebutuhan <i>Software</i>	26
3.4.	Perancangan Sistem	27
3.4.1.	Arsitektur Sistem.....	28
3.4.2.	Diagram Proses Pembuatan Sistem.....	28
3.4.3.	<i>Flowchart</i> Sistem	29
3.5.	Skenario Pengujian.....	30
BAB 4 HASIL DAN PEMBAHASAN		32
4.1.	Hasil	32
4.2.	Pembahasan.....	32
4.2.1.	Struktur Direktori	32
4.2.2.	Implementasi Persiapan Data	33
4.2.3.	<i>Training Model</i>	37

4.2.4.	Pengujian Model	41
4.2.5.	Hasil Integrasi Sistem	44
BAB V	PENUTUP.....	46
5.1	Kesimpulan	46
5.2	Saran dikembangkan	46
DAFTAR PUSTAKA		47
LAMPIRAN		49

DAFTAR TABEL

	Halaman
Tabel 3.3.1. Tabel Tingkat Kualitas.....	26
Tabel 3.3.2. Tabel Kebutuhan <i>Hardware</i>	26
Tabel 3.3.3. Tabel Kebutuhan <i>Software</i>	26
Tabel 3.3.4. Tabel Spesifikasi Google Colab (Hidayatullah, P. 2021).	27
Tabel 3.3.5. <i>Confusion Matrix</i> (Hidayatullah, P. 2021).....	31
Tabel 4.4.1. Tabel Tingkat Kualitas.....	33
Tabel 4.2 Tabel Data <i>Splitting</i>	37
Tabel 4.3 Kinerja Hasil <i>Training</i> Model YOLOv5s	39
Tabel 4.4 Kinerja Hasil <i>Training</i> Model SSDMobileNetV2	40
Tabel 4.5 Tabel Perbandingan Hasil Uji Coba Antara Model YOLOv5s dan YOLOv5m.	41

DAFTAR GAMBAR

	Halaman
Gambar 2.1. <i>Machine Learning vs Deep Learning</i> (D. K. P. Wira, 2020)	6
Gambar 2.2. Arsitektur <i>CNN</i> (Krizhevsky et al, 2012).....	9
Gambar 2.3. Proses <i>Convolution</i>	10
Gambar 2.4. Cara menghitung nilai <i>Convolution</i>	11
Gambar 2.5. Operasi <i>Max-pooling</i> (Medium Samuel Sena, 2017).....	12
Gambar 2.6. <i>Processing of a Fully-Connected Layer</i>	13
Gambar 2.7. <i>Dropout Regulation</i>	14
Gambar 2.8 Ilustrasi proses deteksi YOLO	15
Gambar 2.9 Arsitektur YOLOv5	16
Gambar 2.10 Arsitektur SSDMobileNetV2	18
Gambar 2.11. Ukuran Normal.....	21
Gambar 2.12. <i>Pixel Cropping</i>	21
Gambar 2.13 LabelImg	22
Gambar 3.1. Metode Pelaksanaan	23
Gambar 3.2 Contoh data <i>sample</i>	25
Gambar 3.3 Arsitektur Sistem.....	28
Gambar 3.4 Diagram Proses	29
Gambar 3.5 <i>Flowchart</i> Sistem	30
Gambar 4.1 Struktur direktori <i>root Project</i> Sistem Deteksi	32
Gambar 4.2 Kaggle <i>Datasets</i>	34
Gambar 4.3 <i>Directory File List</i> LabelImg	35
Gambar 4.4 Proses <i>Labelling</i> Menggunakan LabelImg.....	35
Gambar 4.5 <i>File</i> Hasil Anotasi.....	36
Gambar 4.6 Arsitektur YOLOv5s.....	37
Gambar 4.7 Proses <i>Training</i> Model YOLOv5s	38
Gambar 4.8 Arsitektur SSDMobileNetV2	39

Gambar 4.9. Proses <i>Training</i> Model SSDMobileNetV2	40
Gambar 4.10 YOLOv5s vs SSDMobileNetV2 Metrics	42
Gambar 4.11 Deteksi Tingkat Kualitas Tanaman Padi menggunakan YOLOv5s	43
Gambar 4.12 Arsitektur Sistem.....	44
Gambar 4.13 Format API.....	45
Gambar 4.14 Halaman <i>Form</i> Unggah Gambar	45
Gambar 4.15 Halaman Hasil Deteksi Tingkat Kualitas Tanaman Padi	45

DAFTAR LAMPIRAN

LAMPIRAN 1 KODE PROGRAM

LAMPIRAN 2 BIODATA PENULIS

BAB 1

PENDAHULUAN

1.1. Latar Belakang

Indonesia merupakan salah satu negara agraris terbesar di dunia, karena sebagian penduduk Indonesia memiliki mata pencaharian sebagai petani atau bercocok tanam. Sektor pertanian juga berperan penting untuk meningkatkan perekonomian dan memenuhi kebutuhan pangan.

Tanaman Padi merupakan komoditas tanaman pangan yang penting di Indonesia, selain itu Tanaman Padi juga termasuk tanaman pertanian yang berasal dari benua Asia termasuk Indonesia. Tanaman Padi biasanya diolah menjadi beras sebagai makanan pokok bagi sebagian besar penduduk di Indonesia, maka dari itu Tanaman Padi sangat perlu diperhatikan kualitasnya.

Peningkatan kebutuhan hasil pertanian yang lebih tinggi dengan perlindungan kualitas lingkungan yang lebih baik, telah mendorong lahirnya Gerakan “peningkatan hasil dengan dampak lebih rendah”. Gerakan tersebut di Eropa dikenal sebagai “*Smart Farming*” atau pertanian pintar dengan istilah Pertanian 4.0. Gagasan Pertanian 4.0 menarik perhatian pelaku pertanian dalam mendukung pengembangan pertanian modern. Pertanian 4.0 adalah pertanian presisi yang dikombinasikan dengan teknologi informasi digital yang utamanya didukung oleh *big data*, *mobile internet* dan *cloud computing* (Santosa, Edi dan Winarno, Tohir, 2019).

BPS mencatat, pada tahun 2019 struktur lapangan pekerjaan utama masyarakat Indonesia paling besar di bidang pertanian dengan persentase 27,33%. Antusiasme masyarakat untuk bergelut di bidang pertanian harusnya membawa kesejahteraan bagi para petani, namun hal tersebut justru menimbulkan kesenjangan sebab angka kemiskinan di sektor pertanian paling tinggi yaitu sebesar 26,14 juta jiwa di tahun 2013 yang meningkat sebanyak 49,41% pada tahun 2019. Tingginya kemiskinan di bidang pertanian terjadi karena jumlah pemasukan rendah untuk petani atau keuntungan yang sedikit.

Pengelolaan lahan yang kurang ideal serta kurangnya sumber daya manusia untuk mengelola lahan menjadi penyebab berkurangnya pendapatan para petani, oleh karena itu dibutuhkan sebuah sistem cerdas yang digunakan untuk mendeteksi tingkat kualitas tanaman padi dengan cepat dan cakupan deteksinya luas. Teknologi pemrosesan gambar yang dilengkapi dengan kecerdasan buatan dan teknologi *drone* dapat membantu petani dalam mendeteksi tingkat kualitas tanaman padi tanpa perlu membutuhkan SDM yang banyak.

Oleh karena itu pada tugas akhir ini penulis akan membangun sebuah sistem yang dapat mendeteksi tingkat kualitas yang terdapat pada tanaman padi dengan metode *Object Detection* menggunakan algoritma YOLO v5.

1.2. Rumusan Masalah

Berdasarkan latar belakang yang dijabarkan, maka diperoleh suatu rumusan permasalahan yang menjadi dasar pembuatan sistem tersebut, yakni sebagai berikut :

1. Bagaimana membuat sebuah sistem yang dapat mendeteksi dan mengklasifikasi tingkat kualitas tanaman padi?
2. Bagaimana algoritma pendeteksian objek YOLO v5 (*You Only Look Once*) dapat digunakan atau dapat diimplementasikan dalam proses pendeteksian tingkat kualitas tanaman padi?
3. Berapa tinggi tingkat keberhasilan atau akurasi dari hasil deteksi dan klasifikasi yang dilakukan dengan menggunakan algoritma YOLO v5?

1.3. Batasan Masalah

Untuk memfokuskan pembahasan, dapat diperoleh beberapa batasan masalah, di antaranya:

1. Data pengujian diperoleh dari pengambilan sampel langsung menggunakan citra udara atau teknologi *drone* dari pertanian tanaman padi di Kabupaten Majalengka dan Kabupaten Indramayu.
2. Deteksi yang dilakukan hanya pada tanaman padi yang diambil melalui citra udara atau teknologi *drone*.
3. Tanaman Padi yang diambil sebagai *datasets* berumur 45 sampai 60 hari atau dalam fase pertumbuhan vegetatif dan berumur 60 sampai 75 hari

atau dalam fase pertumbuhan generatif.

4. Sistem dibangun menggunakan bahasa pemrograman Python.

1.4. Tujuan Penelitian

Tujuan dari pembuatan sistem yang dibuat meliputi:

1. Merancang *Automatic Object Detector System*, untuk mempermudah pendeteksian tingkat kualitas pada tanaman padi.
2. Mengimplementasikan algoritma YOLO v5 untuk mendeteksi dan mengklasifikasikan tingkat kualitas tanaman padi berdasarkan pola atau *matrixs* gambar yang diambil.

1.5. Manfaat Penelitian

Manfaat yang dapat diperoleh dengan dibuatnya sistem ini adalah:

1. Memberikan kemudahan bagi para petani dalam mendeteksi tingkat kualitas tanaman padi.
2. Diharapkan dapat meningkatkan kualitas dan produktifitas hasil pertanian.
3. Tugas akhir (TA) ini diharapkan dapat memberikan kontribusi untuk pengembangan literatur dalam penelitian yang berhubungan dengan *computer vision* atau *deep learning*.

1.6. Sistematika Penulisan Laporan

Dalam sistematika penulisan laporan tugas akhir ini dibagi menjadi lima bab, dimana setiap bagian memiliki pembahasan yang berbeda-beda tetapi saling terkait antara satu dengan lainnya. Untuk memudahkan penulisan laporan tugas akhir ini, maka akan diurutkan dan dijabarkan setiap bagian secara sistematis. Adapun sistematika penulisan laporan ini sebagai berikut:

BAB I PENDAHULUAN

Pada bab pendahuluan dipaparkan mengenai latar belakang masalah, rumusan masalah, batasan masalah, tujuan serta manfaat penelitian Implementasi Algoritma YOLO V5 Untuk Deteksi dan Klasifikasi Tingkat Kualitas Tanaman Padi.

BAB II LANDASAN TEORI

Landasan teori merupakan seperangkat definisi maupun konsep yang telah disusun berdasarkan sumber referensi yang valid tentang teori-teori terkait penyusunan laporan tugas akhir serta beberapa literatur review yang berhubungan dengan penelitian. Sumber yang dijadikan referensi berasal dari buku maupun dari internet.

BAB III METODOLOGI PENELITIAN

Metodologi penelitian menjelaskan tentang tahapan serta metode penelitian yang digunakan dalam mengimplementasikan Algoritma YOLO V5 Untuk Deteksi dan Klasifikasi Tingkat Kualitas Tanaman Padi. Pada bagian ini terdapat perancangan sistem berupa flowchart, arsitektur sistem, diagram proses, dan skenario pengujian.

BAB IV HASIL DAN PEMBAHASAN

Pada bab ini terdapat hasil penelitian dari Implementasi Algoritma YOLO V5 Untuk Deteksi dan Klasifikasi Tingkat Kualitas Tanaman Padi berdasarkan perancangan dan pembangunan yang telah dipaparkan. Pada bagian ini dibahas mengenai *data preprocessing*, pembuatan model, dan pengujian akurasi pendeteksian model yang sudah dibuat.

BAB V PENUTUP

Bab ini berisi kesimpulan dan saran yang berkaitan dengan analisa dan optimalisasi sistem berdasarkan yang telah diuraikan pada bab-bab sebelumnya. Kesimpulan dihasilkan dari rangkuman berdasarkan poin-poin penting pada penulisan laporan tugas akhir Implementasi Algoritma YOLO V5 Untuk Deteksi dan Klasifikasi Tingkat Kualitas Tanaman Padi. Sedangkan saran berisi tentang pendapat atau masukan dari penulis kepada pembaca.

BAB 2

LANDASAN TEORI

2.1. Object Detection

Object Detection adalah sebuah teknik yang digunakan komputer untuk mendeteksi *instance object* dari kelas tertentu dalam citra digital, seperti wajah, mobil, rambu lalu lintas dan lain-lain. Berbeda dengan klasifikasi, deteksi objek hanya mendeteksi objek serta lokasinya saja di gambar / koordinat *bounding box* nya.

Terdapat beberapa model yang dapat digunakan untuk mendeteksi objek pada citra digital, diantaranya adalah *YOLO (you only look once)*, *SSD (single shot multibox detector)*, *Faster R-CNN*, *Mask R-CNN* dan sebagainya. Sebelum menggunakan model *object detection* kita perlu membuat setiap gambar memiliki anotasi objek yang berisi *bounding box* dan *class* nya.

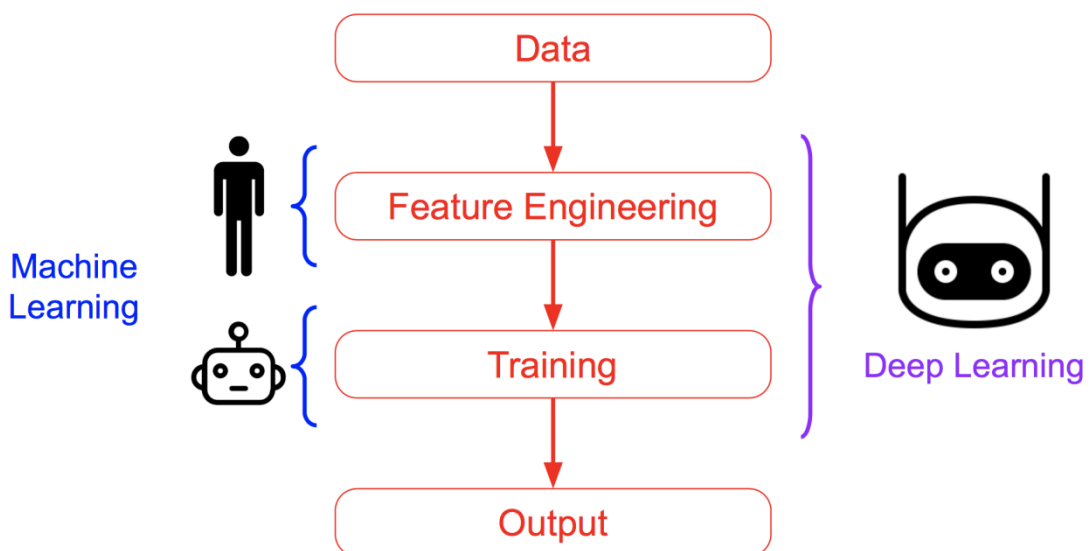
2.2. Klasifikasi Citra

Klasifikasi citra adalah sebuah pekerjaan untuk memasukkan citra dan menempatkan ke dalam suatu kategori. Ini merupakan salah satu dari permasalahan yang ada pada *Computer Vision* yang dapat disederhanakan dan memiliki berbagai macam aplikasinya. Salah satu aplikasi dalam klasifikasi citra adalah pengklasifikasian penyakit paru-paru berdasarkan citra *x-ray* paru-paru.

Setiap citra yang di input pada *training set* data diberikan label atau penamaan. Saat klasifikasi, label atau penamaan tersebut akan menjadi perbandingan dengan hasil hipotesis yang diberikan oleh model pembelajaran dan akan menghasilkan nilai error. Klasifikasi yang terawasi ini bisa sangat efektif dan akurat dalam mengklasifikasikan citra tempat maupun objek lainnya. Banyak metode dan algoritma yang dapat mendukung proses klasifikasi yang terawasi terutama dengan teknik Deep Learning, diantaranya *CNN*, *InceptionV3*, *VGG19* dan sebagainya. Pada klasifikasi citra ini, model hanya dapat mendeteksi satu kelas dalam satu kali proses deteksi.

2.3. Deep Learning

Deep learning merupakan area baru dalam bidang ilmu *machine learning*. *Deep learning* pertama sekali dikemukakan oleh Bapak *neural network* yaitu Prof. Dr. Geoffrey Hinton dari Toronto University, yang dipublikasikan di dalam karya tulisnya yang berjudul *A fast learning algorithm for deep belief nets*. Ketenaran *Deep Learning* baru bermula pada tahun 2012 ketika *deep learning* berhasil menyelesaikan klasifikasi lebih dari 1,2 juta gambar dari kompetisi ImageNet. Keberhasilan ini menjadikan industri IT skala besar seperti Google, Facebook, Amazon, dll kembali tertarik dengan bidang Kecerdasan Buatan. Google Brain di tahun yang sama juga berhasil melaksanakan “*Cat Experiment*” menggunakan dataset yang sangat besar dan dalam jenis *unsupervised learning*. Google Brain berhasil mengidentifikasi wajah kucing dengan pembelajaran yang mendalam. *Deep Learning* merupakan teknik dalam *machine learning* yang memiliki arsitektur yang lebih “*deep*” dibanding dengan teknik *machine learning* lainnya dalam menyelesaikan masalah prediksi, maupun klasifikasi. Perbedaan *machine learning* dengan *deep learning* dapat dilihat secara visual pada Gambar 2.1 (S. Faza, 2018).



Gambar 2.1. *Machine Learning vs Deep Learning* (D. K. P. Wira, 2020)

Deep Learning merupakan cabang ilmu dari *Machine Learning* yang berbasis Jaringan Syaraf Tiruan (JST) atau dapat dikatakan perkembangan dari JST yang mengajarkan komputer untuk dapat melakukan tindakan yang dianggap alami

oleh manusia. Misalnya yaitu belajar dari contoh. Dalam *Deep Learning*, sebuah komputer dapat belajar mengklasifikasi secara langsung dari gambar, suara, teks, atau video sekalipun. Sebuah komputer seperti dilatih dengan menggunakan data set berlabel dan jumlahnya sangat besar yang kemudian dapat mengubah nilai piksel dari sebuah gambar menjadi suatu representasi *internal* atau *feature vector* yang dimana pengklasifikasiannya dapat digunakan untuk mendeteksi atau mengklasifikasi pola pada masukan *input* (LeCun, Bengio, & Hinton, 2015).

Metode *deep learning* adalah metode pembelajaran dengan beberapa tingkat representasi, dimana representasi dapat membentuk arsitektur jaringan syaraf yang mempunyai banyak layer (lapisan). Lapisan pada *deep learning* terbagi menjadi tiga bagian yaitu, *input layer*, *hidden layer*, dan *output layer*. Pada *hidden layer* dapat dibuat banyak lapis atau berlapis-lapis untuk menemukan komposisi algoritma yang tepat agar dapat meminimalisir *error* pada *output*. Semakin banyak layer yang dihasilkan, maka akan semakin kecil *error* yang dihasilkan, sehingga dapat menghasilkan akurasi yang lebih bagus atau lebih tinggi.

Bobot adalah koneksi antar lapisan yang berupa nilai yang menentukan fungsi *input-output* dari mesin. Bobot adalah parameter penyesuaian yang diatur oleh mesin untuk mengukur kesalahan antara nilai *output* dan pola nilai yang diinginkan pada pembelajaran. Sehingga, nilai bobot inilah yang diatur mesin untuk mengurangi kesalahan yang mungkin terjadi. Dalam sistem *deep learning*, kemungkinan ada ratusan juta bobot yang dapat diatur. Untuk menyesuaikan vektor bobot dengan benar, algoritma menghitung nilai gradien vektor untuk setiap bobot berdasarkan jumlah kesalahan yang meningkat atau menurun jika bobot meningkat dalam jumlah kecil (R. K. S. C. Putri, 2018).

2.4. Neural Network

Jaringan Syaraf Tiruan atau Artificial Neural Network adalah teknik dalam Machine Learning yang menirukan syaraf manusia yang merupakan bagian fundamental dari otak. Neural network terdiri atas lapis masukan (*input layer*) dan lapis keluaran (*output layer*). Setiap lapis terdiri atas satu atau beberapa unit neuron yang mempunyai sebuah fungsi aktivasi yang menentukan keluaran dari

unit tersebut. Dapat dilakukan penambahan lapis tersembunyi (*hidden layer*) untuk menambah kemampuan dari *neural network* tersebut. *neural network* bisa dilatih dengan menggunakan data *training*. Semakin banyak data *training* maka akan semakin bagus unjuk kerja dari *neural network* tersebut. Namun, kemampuan *neural network* juga terbatas pada jumlah lapisan, semakin banyak jumlah lapisan semakin tinggi kapasitas *neural network* tersebut. Semakin banyak lapisan juga membawa kekurangan yaitu semakin banyaknya jumlah iterasi atau *training* yang dibutuhkan. Untuk mengatasi hal ini, dikembangkanlah teknik *Deep Learning*. Beberapa aplikasi *neural network* antara lain untuk *Principal Component Analysis*, 2 regresi, klasifikasi citra, dan lain-lain (A. Ahmad, 2017).

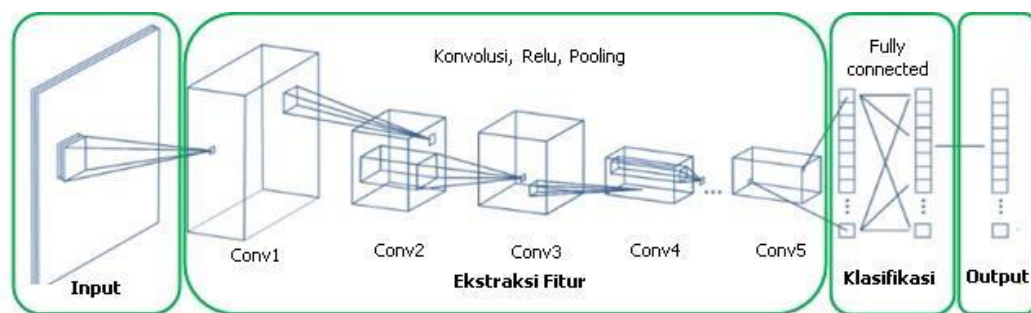
2.5. Convolutional Neural Network

Convolutional Neural Network merupakan salah satu algoritma dari deep learning yang merupakan hasil pengembangan dari Multilayer Perceptron (MLP) yang dirancang untuk melakukan olah data menjadi bentuk dua dimensi, misalnya yaitu: gambar atau suara. Convolution neural network digunakan untuk melakukan klasifikasi data yang berlabel dengan menggunakan metode supervised learning yang cara kerjanya dari supervised learning adalah terdapat data yang dilatih dan terdapat variabel yang ditargetkan sehingga tujuan dari metode ini yaitu mengelompokkan suatu data ke data yang sudah ada.

Convolution neural network sering digunakan untuk mengenali benda berdasarkan penampakan dan melakukan deteksi dan melakukan segmentasi objek. *Convolution neural network* belajar langsung melalui data citra, sehingga dapat menghilangkan ekstraksi ciri dengan cara manual. Penelitian awal yang menjadi dasar penemuan ini yaitu pertama kali dilakukan oleh Hubel dan Wiesel yang melakukan penelitian visual *cortex* pada indera penglihatan kucing. Visual *cortex* pada hewan sangat *powerful* kemampuannya dalam sistem pemrosesan visual yang pernah ada. Sehingga, banyak penelitian yang terinspirasi oleh cara kerjanya dan menghasilkan banyak model-model baru yang beberapa diantaranya yaitu, Neocognitron, HMAX, LeNet-5, dan AlexNet.

CNN juga merupakan saraf yang dikhususkan untuk memproses data yang memiliki struktur kotak (grid). Sebagai contoh yaitu berupa citra dua dimensi.

Nama konvolusi merupakan operasi dari aljabar linear yang mengalikan matriks dari filter pada citra yang akan diproses. Proses ini disebut dengan lapisan konvolusi dan merupakan salah satu jenis dari banyak lapisan yang bisa dimiliki dalam suatu jaringan. Meskipun begitu, lapisan konvolusi ini merupakan lapisan utama yang paling penting digunakan. Jenis lapisan yang lain yang biasa digunakan adalah *Pooling Layer*, yakni lapisan yang digunakan untuk mengambil suatu nilai maksimum atau nilai rata-rata dari bagian-bagian lapisan piksel pada citra. Berikut merupakan gambaran umum arsitektur *Convolution Neural Network*:



Gambar 2.2. Arsitektur *CNN* (Krizhevsky et al, 2012)

Secara umum, lapisan / arsitektur *Convolution Neural Network* dibagi menjadi dua bagian, yaitu:

2.5.1. *Extraction Feature Layer (Feature Learning)*

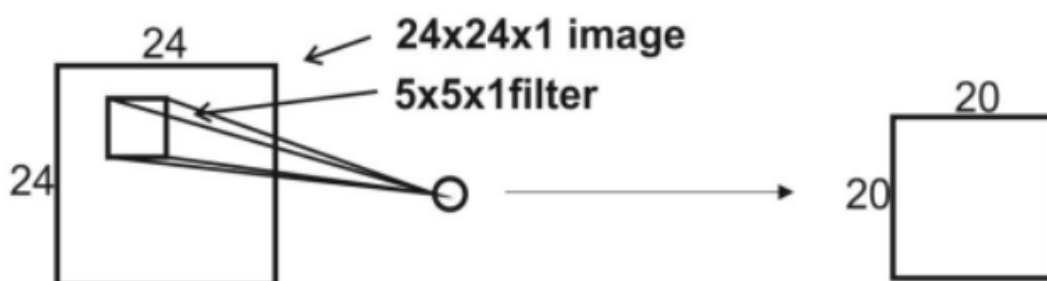
Gambar yang letaknya ada di awal arsitektur yang tersusun atas beberapa lapisan dan di setiap susunan lapisannya atas *neuron* yang terkoneksi pada daerah lokal (*local region*) dari lapisan sebelumnya. Lapisan pada jenis pertama yaitu adalah *convolutional layer* dan lapisan kedua adalah *pooling layer*. Pada setiap lapisan diberlakukan fungsi aktivasi dengan posisinya yang berselang-seling antara jenis pertama dan jenis kedua. Lapisan ini menerima *input* gambar secara langsung dan memprosesnya sampai menghasilkan *output* berupa vektor untuk diolah di lapisan berikutnya.

2.5.2. Classification Layer

Layer ini tersusun atas beberapa lapisan yang di setiap lapisan tersusun atas *neuron* yang terkoneksi secara penuh (*fully connected*) dengan lapisan yang lainnya. Layer ini menerima *input* dari hasil *output* layer ekstraksi fitur gambar berupa vektor yang kemudian di transformasikan seperti pada *Multi Neural Network* dengan tambahan beberapa *hidden layer*. Hasil *output* berupa akurasi kelas untuk klasifikasi. Dengan ini, CNN merupakan metode untuk melakukan transformasi gambar asli lapisan per lapisan dari nilai piksel gambar ke dalam nilai skoring kelas untuk klasifikasi. Setiap lapisan ada yang memiliki *hyperparameter* dan ada yang tidak memiliki parameter (bobot dan bias pada *neuron*).

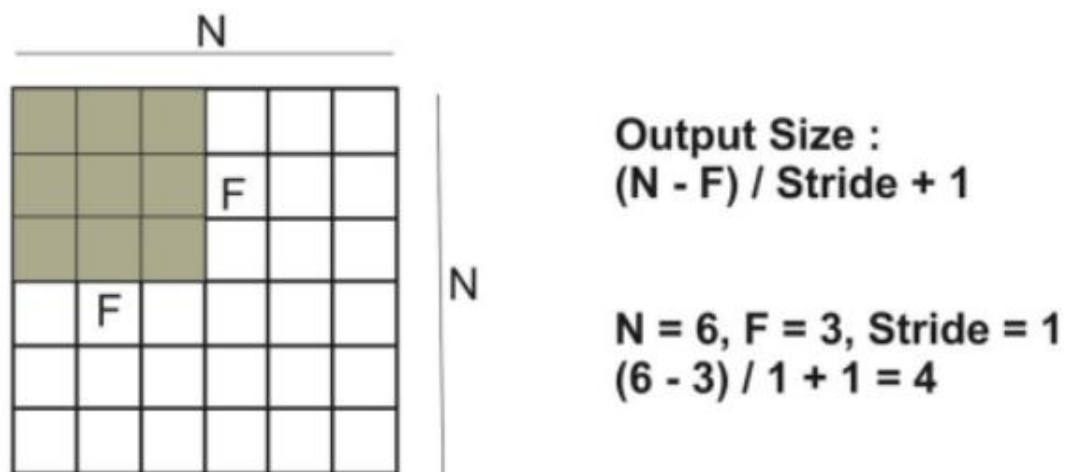
2.6. Convolutional Layer

Convolutional Layer bagian yang melakukan operasi konvolusi yaitu mengkombinasikan linier filter terhadap daerah lokal. Layer ini yang pertama kali menerima gambar yang diinputkan pada arsitektur. Bentuk layer ini adalah sebuah filter dengan panjang (pixel), tinggi (pixel), dan tebal sesuai dengan channel image data yang diinputkan. Ketiga filter ini akan bergeser ke seluruh bagian gambar. Pergeseran tersebut akan melakukan operasi “dot” antara input dan nilai dari filter tersebut sehingga akan menghasilkan output yang disebut sebagai activation map atau feature map. **Gambar 2.3.** menampilkan proses konvolusi yang ada di dalam convolutional layer dan **Gambar 2.4.** adalah cara menghitung nilai konvolusinya (R. K. S. C. Putri, 2018).



Gambar 2.3. Proses *Convolution*

Convolution Layer adalah inti dari CNN. *Convolution Layer* menghasilkan citra baru yang menunjukkan fitur dari citra *input*. Dalam proses tersebut, *Convolution Layer* menggunakan *filter* pada setiap citra yang menjadi masukan. *Filter* pada *layer* ini berupa *array* 2 dimensi bisa berukuran 5×5 , 3×3 atau 1×1 . Proses *convolution* dengan menggunakan *filter* pada *layer* ini akan menghasilkan *feature map* yang akan digunakan pada *activation layer*.

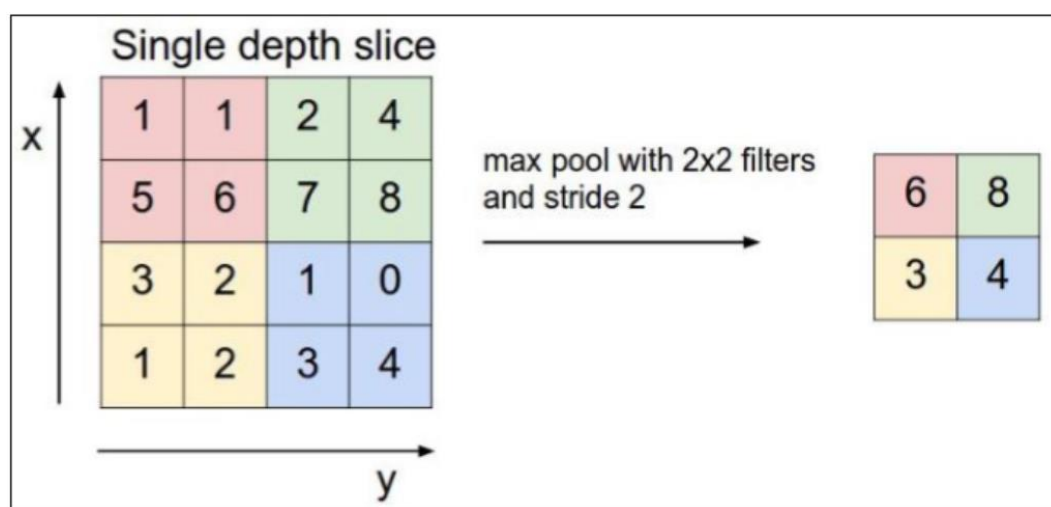


Gambar 2.4. Cara menghitung nilai *Convolution*

2.7. Pooling Layer

Pooling merupakan pengurangan ukuran matriks dengan menggunakan operasi *pooling*. *Pooling Layer* biasanya berada setelah conv. Pada dasarnya *pooling layer* terdiri dari sebuah filter dengan ukuran dan *stride* tertentu yang akan secara bergantian bergeser pada seluruh area *feature map*. Dalam *pooling layer* terdapat dua macam *pooling* yang biasa digunakan yaitu *average pooling* dan *max-pooling*. Nilai yang diambil pada *average pooling* adalah nilai rata-rata, sedangkan pada *max-pooling* adalah nilai maksimal. Lapisan *Pooling* yang dimasukkan di antara lapisan konvolusi secara berturut-turut dalam arsitektur model CNN dapat secara progresif mengurangi ukuran *volume output* pada *Feature Map*, sehingga mengurangi jumlah parameter dan perhitungan di jaringan, untuk mengendalikan *Overfitting*. Lapisan *pooling* bekerja di setiap tumpukan *feature map* dan melakukan pengurangan pada ukurannya. Bentuk

lapisan *pooling* umumnya dengan menggunakan *filter* dengan ukuran 2×2 yang diaplikasikan dengan langkah sebanyak dua dan beroperasi pada setiap irisan dari inputnya. *Pooling layer* menerima input dari *activation layer* kemudian mengurangi jumlah paramaternya. Poling juga biasa disebut *subsampling* atau *downsampling* yang mengurangi dimensi dari *feature map* tanpa menghilangkan informasi penting di dalamnya. Proses dalam *pooling layer* cukup sederhana. pertama-tama kita menentukan ukuran *downsampling* yang akan digunakan pada feature map, misalnya 2×2 . Contoh operasi *max-pooling* dapat dilihat pada **Gambar 2.5**. (Julian Turlan, 2020).



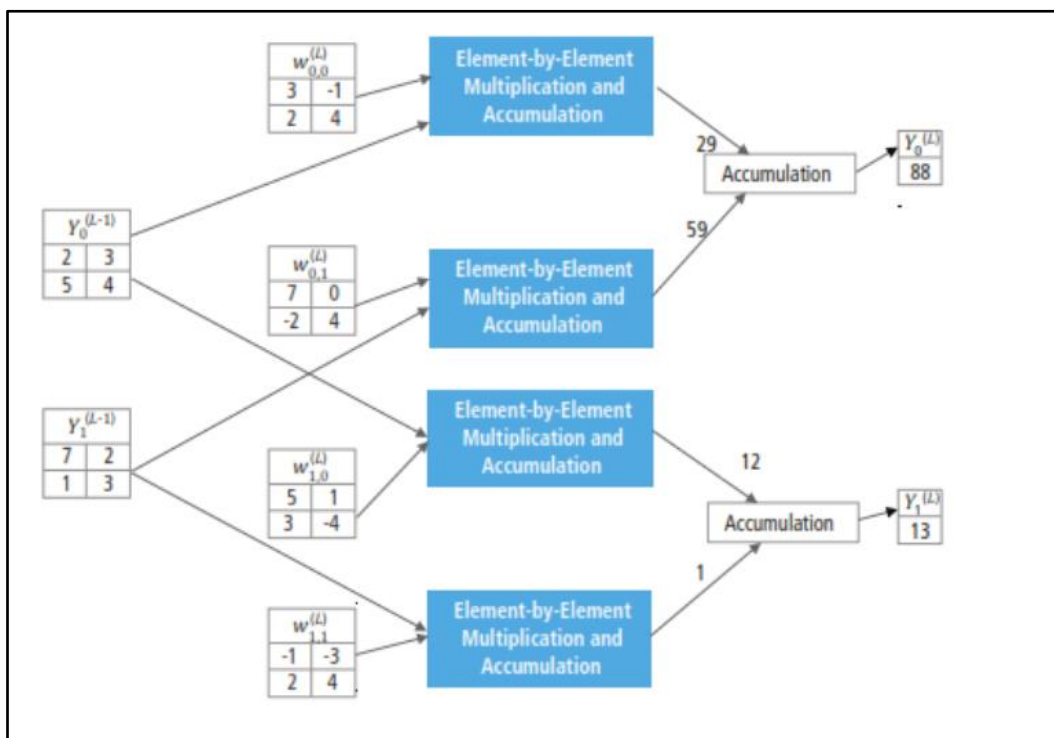
Gambar 2.5. Operasi *Max-pooling* (Medium Samuel Sena, 2017)

Berdasarkan **Gambar 2.5**. menunjukan proses dari *max-pooling*, *output* dari proses *pooling* adalah sebuah matriks dengan dimensi yang lebih kecil dibandingkan dengan citra awal. Lapisan *pooling* akan beroperasi pada setiap irisan kedalaman volume *input* secara bergantian. Jika dilihat dari **Gambar 2.5**. operasi *max-pooling* dengan menggunakan ukuran filter 2×2 . Masukan pada proses tersebut berukuran 4×4 , dari masing-masing 4 angka pada *input* operasi tersebut diambil nilai maksimalnya kemudian dilanjutkan membuat ukuran *output* baru menjadi ukuran 2×2 (T. Nurhikmat, 2018).

2.8. Fully-Connected Layer

Fully-Connected Layer adalah sebuah lapisan dimana sesuai *neuron* aktivasi dari lapisan sebelumnya terhubung semua dengan *neuron* di lapisan selanjutnya

sama seperti halnya dengan *neural network* biasa. Pada dasarnya lapisan ini biasanya digunakan pada MLP (*Multi Layer Perceptron*) yang mempunyai tujuan untuk melakukan transformasi pada dimensi data, agar data dapat diklasifikasikan secara linear. Perbedaan antara lapisan *Fully-Connected* dan lapisan konvolusi biasa adalah *neuron* di lapisan konvolusi terhubung hanya ke daerah tertentu pada *input*, sementara lapisan *Fully-Connected* memiliki *neuron* yang secara keseluruhan terhubung. Namun, kedua lapisan tersebut masih mengoperasikan produk *dot*, sehingga fungsinya tidak begitu berbeda. Pada Gambar 2.6. diperlihatkan proses *fully-connected layer* (T. Nurhikmat, 2018).

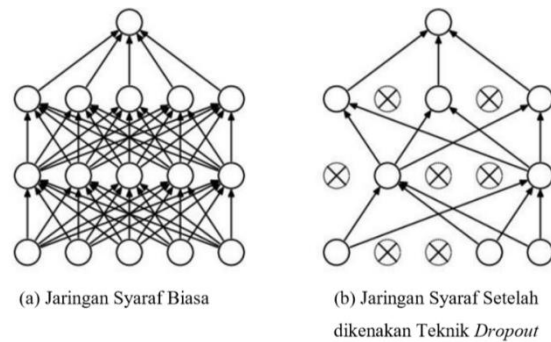


Gambar 2.6. *Processing of a Fully-Connected Layer*

2.9. Dropout Regulation

Dropout merupakan sebuah teknik regulasi jaringan syaraf dengan tujuan memilih beberapa neuron secara acak dan tidak akan dipakai selama proses pelatihan, dengan kata lain neuron-neuron tersebut dibuang secara acak. Hal ini berarti bahwa kontribusi neuron yang dibuang akan diberhentikan sementara

jaringan dan bobot baru juga tidak diterapkan pada neuron pada saat melakukan backpropagation. Berikut adalah gambar proses dropout:



Gambar 2.7. *Dropout Regulation*

2.10. *Softmax Classifier*

Softmax classifier adalah bentuk lain dari algoritma regresi logistik yang dapat digunakan untuk melakukan klasifikasi lebih dari dua kelas. Standar dari klasifikasi yang biasanya dilakukan oleh algoritma dari regresi logistik adalah tugas untuk klasifikasi kelas biner. Pada softmax mempunyai bentuk persamaan sebagai berikut:

$$f_j(z) = \frac{e^{z_j}}{\sum_k e^{z_k}} \quad (2.1)$$

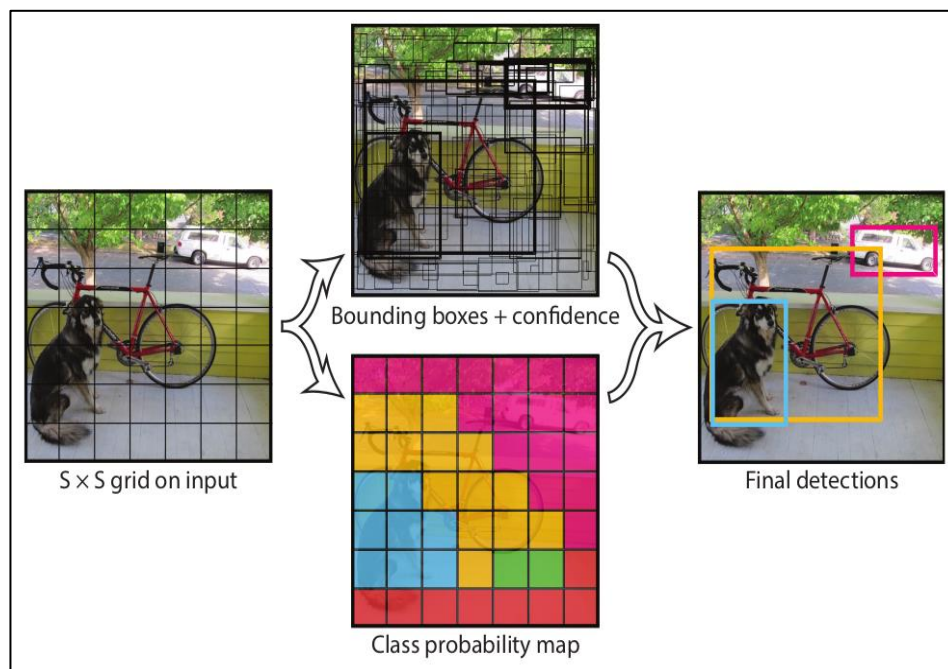
Pada persamaan diatas untuk notasi f_j menunjukkan hasil fungsi untuk setiap elemen ke- j pada vektor output kelas. Argumen z merupakan hipotesis yang diberikan oleh model pelatihan supaya dapat diklasifikasikan oleh fungsi softmax. Softmax juga memberikan hasil yang lebih intuitif dan memiliki hasil interpretasi probabilistik yang lebih baik dibandingkan dengan algoritma klasifikasi lainnya. Softmax memungkinkan peneliti untuk menghitung nilai probabilitas untuk semua label. Hasil dari label yang ada, akan diambil sebuah vektor nilai yang mempunyai nilai riil dan merubahnya menjadi vektor dengan nilai antara nol dan satu. Jika semua hasil dijumlah maka akan bernilai satu.

2.11. Algoritma YOLO

Pengembangan YOLO (*You Only Look Once*) ini telah sampai di versi ke-5 yang memiliki akurasi lebih baik dari versi – versi sebelumnya. Memiliki sembilan *pre-trained model*, YOLO V5 memberikan pilihan untuk menyesuaikan dengan *hardware* yang akan digunakan sehingga dapat berjalan dengan baik sesuai *hardware* yang telah tersedia. *Pre-trained model* YOLO versi 5 dibagi menjadi empat versi yaitu: YOLOv5s, YOLOv5m, YOLOv5l, dan YOLOv5x. Pembagian ini berdasarkan besar ukuran model, ukuran terbesar memiliki akurasi yang lebih tinggi, dan waktu deteksi untuk satu gambar akan meningkat pula.

Teknologi yang digunakan dalam input dari YOLO V5 menyertakan *Mosaic data*, *enhancement*, *adaptive anchor*, *calculation*, dan *adaptive image scaling*. Teknologi yang digunakan pada ‘*Backbone*’ tersemat *Focus structure* dan CSP (*Common Spatial Pattern*) structure. Teknik yang digunakan pada bagian ‘*Neck*’ tersemat FPN+PAN structure; *In Prediction*, *GIoU_Loss* yang digunakan untuk menggantikan metode penghitungan IoU biasanya.

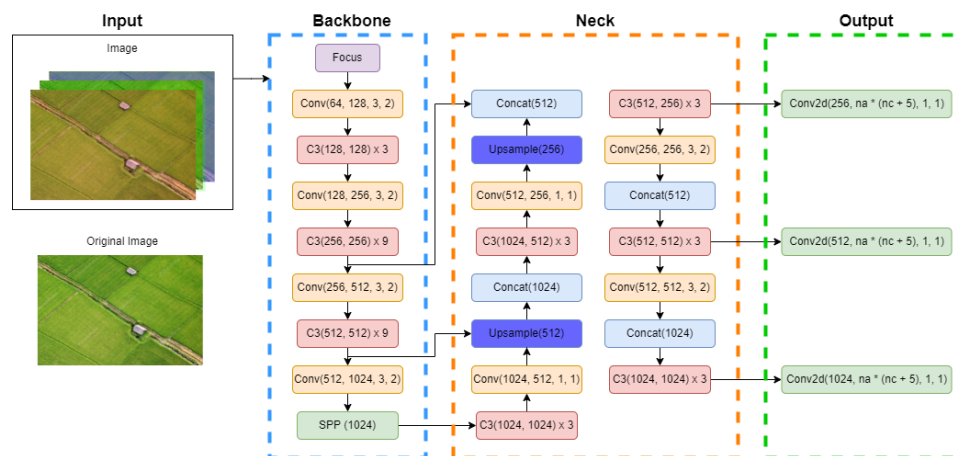
YOLO v5 kurang mampu dibandingkan pendahulunya YOLOv4 dalam hal kinerja, tetapi jauh lebih fleksibel dan lebih cepat daripada YOLOv4, sehingga memiliki keuntungan dalam penggunaan model nya. (Yang, 2021).



Gambar 2.8 Ilustrasi proses deteksi YOLO

YOLO menerima sebuah input image yang dibagi menjadi *grid* sebesar $S \times S$ yang dikirimkan ke sebuah *neural network* untuk membuat *bounding box* dan *class prediction*. Setiap *grid cell* memprediksi *bounding box* dan *confidence score* dari tiap kotak. *Confidence score* inilah yang merefleksikan seberapa tingkat kepercayaan model bahwa objek di dalam kotak berupa objek yang diprediksikan. YOLO menilai *confidence* sebagai $Pr(\text{Object}) * \text{IOU truth}$ (*Intersection of Union*). Kebanyakan sistem deteksi sebelumnya menggunakan pengklasifikasian atau *localizer* untuk melakukan deteksi dengan menerapkan model ke gambar di beberapa lokasi dan memberi nilai *confidence* pada gambar sebagai bahan untuk pendeteksian. YOLO menggunakan pendekatan yang sangat berbeda dengan algoritma sebelumnya, yakni menerapkan jaringan syaraf tunggal pada keseluruhan gambar. Jaringan ini akan membagi gambar menjadi wilayah-wilayah kemudian memprediksi kotak pembatas dan probabilitas, untuk setiap kotak wilayah pembatas ditimbang probabilitas nya untuk mengklasifikasikan sebagai objek atau bukan (R, Aditya. 2018).

Tiap *bounding box* terdiri dari 5 prediksi: class, x, y, w, h. Koordinat (x, y) merepresentasikan pusat dari kotak relatif dengan batas dari *grid cell*. Lebar (*width*) dan tinggi (*height*) adalah prediksi relatif dari seluruh gambar. Tiap *grid cell* memprediksi *conditional class probabilities* C, $Pr(\text{Class} / \text{Object})$. Probabilitas ini dikoneksikan ke *grid cell* yang ada objeknya.



Gambar 2.9 Arsitektur YOLOv5

Gambar diatas merupakan arsitektur YOLOv5, YOLOv5 merupakan *single-stage object detector*, *single-stage object detector* memiliki tiga bagian penting yaitu *Backbone*, *Neck* dan *Head*. Model *Backbone* pada YOLOv5 di dalamnya memiliki CSP (*Cross Stage Partial Networks*) yang digunakan untuk mengekstrak fitur yang informatif dari gambar input. CSP menunjukkan peningkatan signifikan dalam waktu pemrosesan dengan jaringan yang lebih dalam. Pada proses *Backbone* gambar akan diekstrak menggunakan CSPNet yang akan menyimpan informasi penting dari gambar sekaligus mengurangi model yang kompleks.

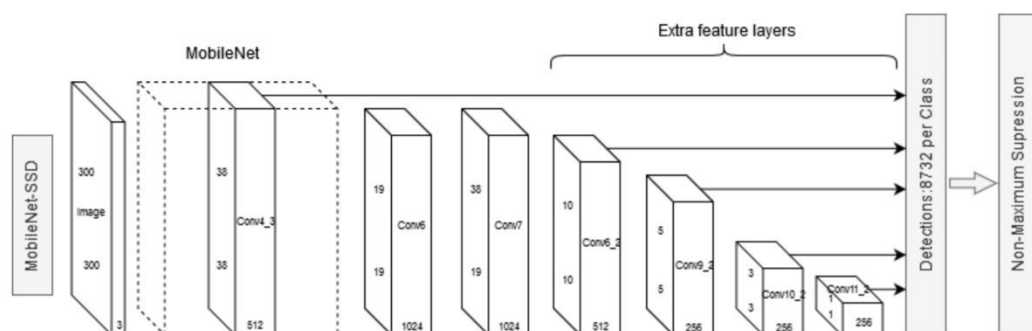
Model *Neck* digunakan untuk menghasilkan piramida fitur. Piramida fitur membantu model untuk mengidentifikasi objek yang sama dengan ukuran dan skala yang berbeda dengan cara digeneralisasi dengan baik pada penskalaan objeknya. Piramida fitur sangat berguna dan membantu model untuk bekerja baik pada data yang tidak terlihat, YOLO v5 menggunakan PANet untuk mendapatkan atau menghasilkan piramida fitur. Pada proses *Neck* informasi yang telah dihasilkan oleh *backbone*, *Neck* digunakan untuk membuat informasi yang berguna di setiap level fitur yang akan menyebar langsung ke sub-jaringan.

Model *Head* atau *Output* digunakan untuk melakukan bagian pada pendeteksian akhir, model ini menerapkan *box*, akurasi dan kelas pada objek yang terdeteksi pada gambar. Model *Head* akan menghasilkan 3 ukuran *layer* yang berbeda untuk mencapai deteksi *multi-scale*, deteksi *multi-scale* akan memastikan bahwa model dapat mengikuti perubahan ukuran. Selanjutnya layer akan menghasilkan prediksi seperti kelas, akurasi dan kotak pembatas.

2.12. Algoritma SSDMobileNetV2

SSDMobileNetV2 merupakan sebuah algoritma deteksi objek. Algoritma ini terdiri dari SSD yang berperan sebagai base model dan mobile net v2 yang berperan sebagai network model. SSD akan mengatur pendeteksian objek dengan membuat bounding box dan mobile net akan bekerja untuk mengekstrak fitur yang nantinya akan diklasifikasi. SSD merupakan singkatan dari single shot detector, ssd disebut sebagai single shot detector karena semua proses deteksi yang dilakukan di algoritma ini merangkum semua perhitungan dalam satu

jaringan. Model SSD biasanya digabungkan dengan arsitektur CNN seperti mobile net, VGG16, dan lain sebagainya untuk membuat algoritma pendeteksian objek. MobileNetV2 merupakan sebuah algoritma CNN, peran mobile net pada algoritma SSDMobileNetV2 adalah untuk membantu mengklasifikasi objek yang terdapat pada suatu gambar. MobileNetV2 merupakan pengembangan dari model terdahulu nya yaitu MobileNetV1, untuk perbedaannya mobile net versi pertama hanya terdapat satu layer sedangkan mobile net versi kedua terdapat dua layer yang membuat waktu proses semakin cepat dan memiliki akurasi yang lebih tinggi. Berikut adalah arsitektur SSDMobileNetV2.



Gambar 2.10 Arsitektur SSDMobileNetV2

2.13. Python

Bahasa pemrograman Python merupakan bahasa pemrograman yang dapat dikembangkan oleh siapa saja karena bersifat open source atau dengan kata lain bahasa pemrograman ini gratis, dapat digunakan tanpa lisensi, dan dapat dikembangkan semampu yang dapat dilakukan. Sebenarnya bahasa pemrograman Python ini mudah dipelajari karena penulisan sintaks yang lebih fleksibel. Selain itu, bahasa pemrograman Python ini memiliki efisiensi tinggi untuk struktur data level tinggi, pemrograman berorientasi objek lebih sederhana tetapi efektif, dapat bekerja pada multi-platform, dan dapat digabungkan dengan bahasa pemrograman lain untuk menghasilkan aplikasi yang diinginkan.

Python dikenal sebagai bahasa pemrograman interpreter, karena Python dieksekusi dengan sebuah interpreter. Terdapat dua cara untuk menggunakan interpreter, yaitu dengan mode baris perintah dan modus script. Pada mode baris,

perintah diketikkan pada shell atau command line dan Python langsung menampilkan hasilnya.

2.14. Golang

Go atau biasa disebut Golang merupakan sebuah bahasa pemrograman open-source yang dikembangkan oleh Google. Hingga saat ini golang digunakan oleh banyak perusahaan-perusahaan besar maupun startup yang bergerak di bidang teknologi. Golang merupakan salah satu back-end tercepat golang menurut perhitungan *benchmarks* yang dikerjakan oleh Tech Empower.

Menurut tech empower dalam perhitungan *benchmark* terhadap 122 total *web frameworks*, *fiber* merupakan salah satu *web framework* tercepat yang dibangun menggunakan bahasa pemrograman golang dengan weighted score sebesar 5.837, disusul *atreugo* dan *gearbox*. (Tech Empower. 2021).

2.15. Jupyter Notebook

Jupyter *Notebook* (file yang berekstensi .ipynb) adalah dokumen yang dihasilkan oleh Jupyter *Notebook App* yang berisikan kode komputer dan *rich text element* seperti paragraf, persamaan matematik, gambar dan tautan (*links*). Jupyter *Notebook* dikenal sebelumnya sebagai IPython *Notebook* berevolusi menjadi Jupyter Lab (Setiabudidaya, 2018).

2.16. Google Colab

Google Colaboratory atau yang biasa disingkat “Google Colab” atau “Colab” adalah produk dari Google Research. Colab adalah layanan Jupyter *Notebook* yang dihosting. Selain itu Colab juga menyediakan akses gratis ke sumber daya komputasi termasuk GPU. Bahasa Pemrograman yang *support* pada Colab adalah Python, Versi Python yang *support* dengan Colab adalah Python 3. Colab sangat cocok digunakan untuk *machine learning*, *data analysis*, dan pendidikan (Hidayatullah, P. 2021).

Google Colab menjadi solusi terutama untuk para mahasiswa karena memiliki kelebihan-kelebihan yaitu memiliki spesifikasi yang relatif tinggi, *ready to use* karena sudah terinstall Python, OpenCV, CUDA, dan cuDNN yang satu sama lain *compatible* (Hidayatullah, P. 2021).

2.17. Augmentasi Data

Augmentasi data merupakan proses dalam pengolahan data gambar. Pada proses ini data gambar akan diubah atau dimodifikasi sedemikian rupa dengan banyak bentuk sehingga dapat menghasilkan data baru. Walaupun dengan sumber gambar yang sama, namun mesin akan menganggap gambar tersebut adalah gambar yang berbeda.

2.17.1. Rotasi

Rotasi yang digunakan yaitu 90 derajat searah berlawanan jarum jam (*counter clockwise*). Rotasi dilakukan dengan persamaan:

$$x' = x \cos[\theta] - y \sin[\theta] \quad (2.2)$$

$$y' = x \sin[\theta] + y \cos[\theta] \quad (2.3)$$

Dalam hal ini, θ adalah sudut rotasi berlawanan dengan arah jarum jam. Jika citra semula adalah A, dan citra hasil rotasi adalah B, maka rotasi citra dari A ke B adalah sebagai berikut:

$$A[x, y] = B[x \cos[\theta] - y \sin[\theta], x \sin[\theta] + y \cos[\theta]] \quad (2.4)$$

2.17.2. Flipping

Flip merupakan teknik membalikkan gambar berlawanan arah seperti pada cermin sehingga teknik ini juga disebut sebagai teknik pencerminan. *Flip* dibagi menjadi dua, yaitu horizontal dan vertikal. *Flip* horizontal adalah pencerminan terhadap sumbu Y. Berikut ini persamaan untuk *Flip* horizontal.

$$B[x, y] = A[N - x][y] \quad (2.5)$$

Sedangkan untuk *flip* vertikal adalah pencerminan terhadap sumbu X. Berikut adalah persamaan *flip* vertikal.

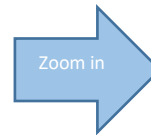
$$B[x, y] = A[x][M - y] \quad (2.6)$$

2.17.3. Cropping

Cropping pada pengolahan citra digital dilakukan dengan cara memperbesar ukuran piksel sebuah gambar.

1	1	2	2
1	1	2	2
3	3	4	4
3	3	4	4

Gambar 2.11. Ukuran Normal



1	2
3	4

Gambar 2.12. *Pixel Cropping*

Rumus untuk *cropping zoom in* ataupun *zoom out*, adalah sebagai berikut:

$$x' = s_x \cdot x \quad (2.7)$$

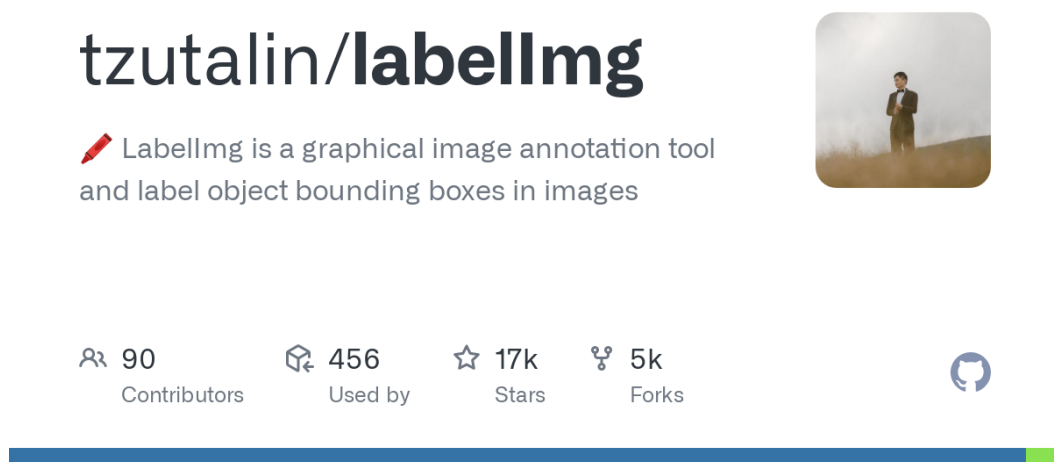
$$y' = s_y \cdot y \quad (2.8)$$

s_x pada persamaan (7) merupakan faktor penyekalaan untuk x dalam arah x, dan s_y pada persamaan (8) merupakan faktor penyekalaan untuk y dalam arah y. Jika citra semula adalah Gambar 3.10. dan citra hasil adalah Gambar 3.11, maka *cropping* citra dinyatakan sebagai berikut.

$$A[x,y] = A[s_x \cdot x, s_y \cdot y] = B[x',y'] \quad (2.9)$$

2.18. Image Labeling

Image labeling adalah sebuah proses pemberian anotasi label terhadap objek yang menjadi target pada gambar berdasarkan *class* nya. Pada sistem yang penulis buat, pendeteksian gejala dapat dilakukan lebih dari satu objek dalam satu gambar (*Multilabel*). Proses *labeling* dilakukan menggunakan *tools/aplikasi* LabelImg. LabelImg adalah tools gratis dan *open-source* yang dikembangkan Tzutalin, labeling merupakan program *executable* yang dibangun menggunakan bahasa pemrograman python. LabelImg tidak hanya menyediakan format labeling untuk YOLO, ada juga format lainnya seperti PascalVOC dan CreatML. LabelImg dapat diunduh melalui *repository* github <https://github.com/tzutalin/labelImg>



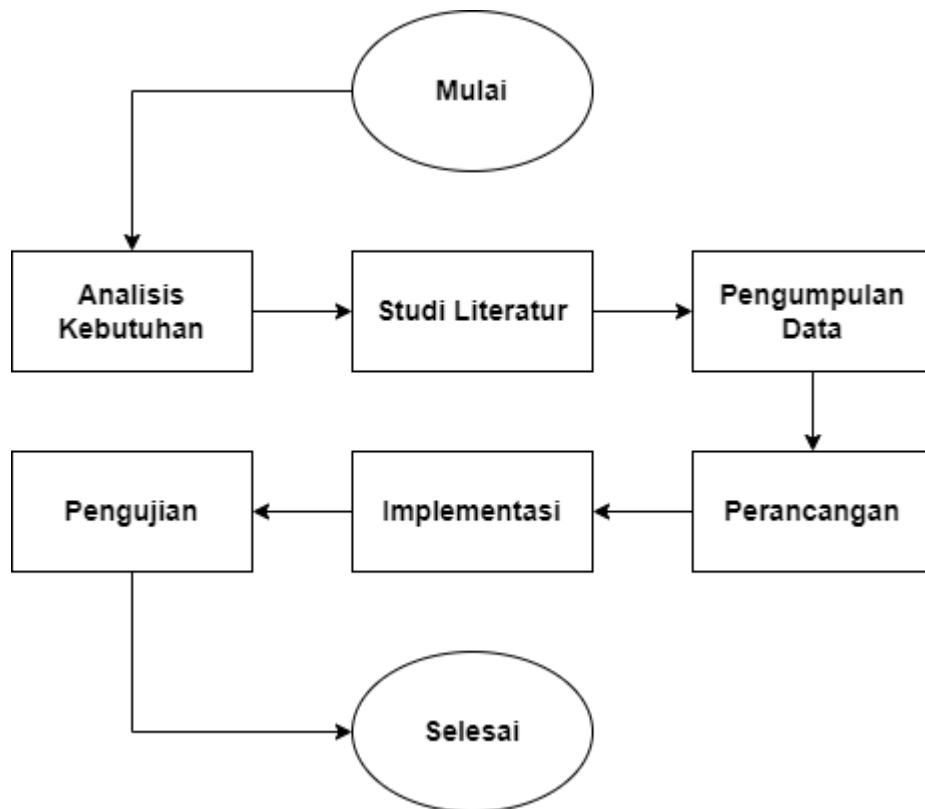
Gambar 2.13 LabelImg

BAB 3

METODOLOGI PELAKSANAAN

3.1. Metodologi Pelaksanaan

Berdasarkan rumusan masalah dan tujuan penelitian yang ada pada Bab I, metode pelaksanaan ini berisikan langkah-langkah yang digunakan dalam mengembangkan sistem untuk mendeteksi dan mengklasifikasikan tingkat kualitas tanaman padi menggunakan algoritma YOLOv5 sebagai tugas akhir. Tahapan dalam pelaksanaannya dapat dilihat pada Gambar 3.1 berikut.



Gambar 3.1. Metode Pelaksanaan

Metode Pelaksanaan pada pengembangan sistem ini dimulai dari melakukan analisis kebutuhan mengenai sistem yang akan dibangun, seperti menentukan teknologi, *framework*, kebutuhan *hardware* dan *software*. Kemudian pada tahap selanjutnya penulis melakukan studi literatur seputar *computer vision*, *deep learning*, *image labeling*, algoritma YOLO v5 dan pengolahan citra. Selanjutnya

adalah tahap pengumpulan data, data yang dikumpulkan berupa gambar atau foto yang diambil menggunakan citra udara atau *drone*, lalu dilakukan *labeling* pada setiap data yang telah diambil. Tahap berikutnya adalah perancangan, perancangan ini meliputi pembuatan model *deep learning*, *image preprocessing* dan pembuatan sistem *back-end* yang nantinya akan digunakan untuk menerima dan mendeteksi gambar. Pada tahap ini akan dijelaskan mengenai alur pembuatan model, mulai dari menyiapkan *library* yang dibutuhkan, membersihkan data yang sudah diberi label dan membagi data ke dalam *train* dan *validation*. Setelah data sudah siap, tahap selanjutnya adalah pembuatan model *deep learning*, pada tahap ini dilakukan proses *training and validation* untuk mendapatkan bobot (*weights*) terbaik yang akan dijadikan model untuk pendeteksian. Setelah didapat model terbaik akan dilakukan pengujian atau *testing* untuk mengetahui performa dari model. Pengujian tersebut dilakukan untuk mengetahui seberapa besar *confidence score*, *precision*, *recall* dan akurasi dari pendeteksian tiap class.

3.2. Pengumpulan Data

Datasets merupakan hal yang paling penting dalam *Deep Learning*, karena tanpa data sistem tidak akan bisa belajar apapun. Data berperan penting terhadap performa model, semakin banyak data yang kualitasnya baik dan juga bervariasi maka akan menghasilkan model dengan performa yang sangat baik, begitupun sebaliknya.

Data *sample* yang digunakan untuk pelatihan model diambil dari forum ternama penyedia *datasets* yaitu Kaggle, data tersebut berisi gambar tanaman padi yang diambil menggunakan citra udara atau *drone*. Data tersebut berjumlah 7820 data yang terdiri dari 3610 data tanaman padi kualitas baik dan 4210 data tanaman padi berkualitas buruk. Data yang berisi gambar tersebut berukuran 640 x 640 *pixels*. Contoh dari data yang digunakan dapat dilihat pada Gambar 3.2 sebagai berikut.



Gambar 3.2 Contoh data *sample*

Pada Gambar 3.2. merupakan data mentah yang masih belum bisa dijadikan sebagai dataset, dikarenakan butuh beberapa proses yang harus dilakukan sebelum data tersebut masuk ke proses *training*. Dari data tersebut, penulis menemukan 2 tingkat kualitas pada tanaman padi. Tingkat kualitas tersebut terdapat pada Tabel 3.1.

Tabel 3.3.1. Tabel Tingkat Kualitas

No	Tingkat Kualitas	Label
1	Baik / Sehat	good
2	Buruk / Tidak Sehat	bad

3.3. Analisis Kebutuhan Sistem

Dalam pembuatan sistem deteksi objek ini terdapat kebutuhan *hardware* dan kebutuhan *software*. Berikut adalah kebutuhan-kebutuhan yang diperlukan:

3.3.1. Kebutuhan *Hardware*

Kebutuhan *hardware* yang diperlukan pada pembuatan sistem yang penulis buat ini merupakan spesifikasi minimal untuk dapat membuat dan menjalankan program dengan baik. Sesuai dengan rekomendasi yang dipaparkan pada dokumentasi resmi YOLOv5, kebutuhan *hardware* yang diperlukan penulis uraikan pada Tabel 3.2 dibawah ini.

Tabel 3.3.2. Tabel Kebutuhan *Hardware*

No	Jenis <i>Hardware</i>	Kebutuhan <i>Hardware</i>	Keterangan
1	<i>Processor</i>	4 Core	Minimal
2	<i>RAM</i>	32 GB	Minimal
3	<i>GPU</i>	Tesla M60	Minimal

3.3.2. Kebutuhan *Software*

Adapun kebutuhan *software* dalam pembuatan sistem pendeteksian gejala penyakit tanaman bawang merah ini terdapat pada Tabel 3.3.

Tabel 3.3.3. Tabel Kebutuhan *Software*

No	Jenis <i>Software</i>	Kebutuhan <i>Software</i>
1	Bahasa Pemrograman	Python, Go

2	<i>Software Pengolah</i>	Visual Studio Code, Jupyter Notebook, LabelImg, Google Colab
3	<i>Library</i>	Pytorch, YOLOv5, OpenCV

Untuk proses *training* model *deep learning* sistem pendeteksian dan klasifikasi tingkat kualitas pada tanaman padi ini menggunakan Google Colab. Google Colab mempunyai spesifikasi yang relatif tinggi. Berikut ini adalah spesifikasi dari Google Colab.

Tabel 3.3.4. Tabel Spesifikasi Google Colab (Hidayatullah, P. 2021).

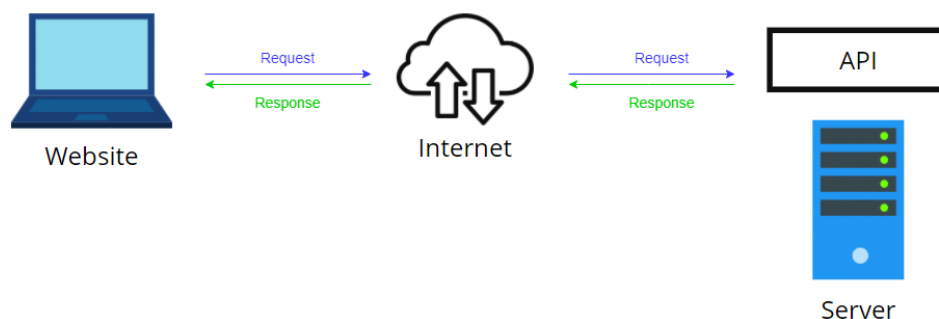
No	Parameter	Colab
1	CPU	Intel(R) Xeon(R)
2	CPU <i>Frequency</i>	2.30GHz
3	CPU <i>Cores</i>	2
4	<i>Available</i> RAM	12 GB (<i>Upgradable to 25GB</i>)
5	<i>Disk Space</i>	68GB
6	GPU	NVIDIA K80/T4
7	GPU Memory	12GB / 16GB
8	Compute Capability	3,7 / 7,5

3.4. Perancangan Sistem

Untuk mendapatkan gambaran mengenai sistem yang dibuat, maka dibuat rancangan terkait bagaimana sistem ini bekerja. Ada 2 rancangan untuk menjabarkan bagaimana sistem ini bekerja yaitu arsitektur sistem dan diagram proses. Arsitektur sistem menjelaskan mengenai bagaimana sistem dapat digunakan oleh sistem lain agar tujuan penelitian dapat tercapai. Sedangkan diagram proses menjelaskan proses dari awal hingga pengujian model dilakukan.

3.4.1. Arsitektur Sistem

Pada pembuatan sistem deteksi dan klasifikasi tingkat kualitas tanaman padi, penulis akan membangun sebuah arsitektur sistem. Sistem yang penulis bangun merupakan sebuah sistem yang terintegrasi dengan sistem lain. Oleh karena itu penulis menyediakan API yang dapat diakses oleh sistem yang telah ditentukan. Penulis membangun API menggunakan bahasa pemrograman go atau lebih sering disebut sebagai golang. Pada Gambar 3.3. merupakan rancangan arsitektur yang akan penulis bangun, dimana *user* dapat mengakses *service* dengan cara mengaksesnya melalui jaringan internet.

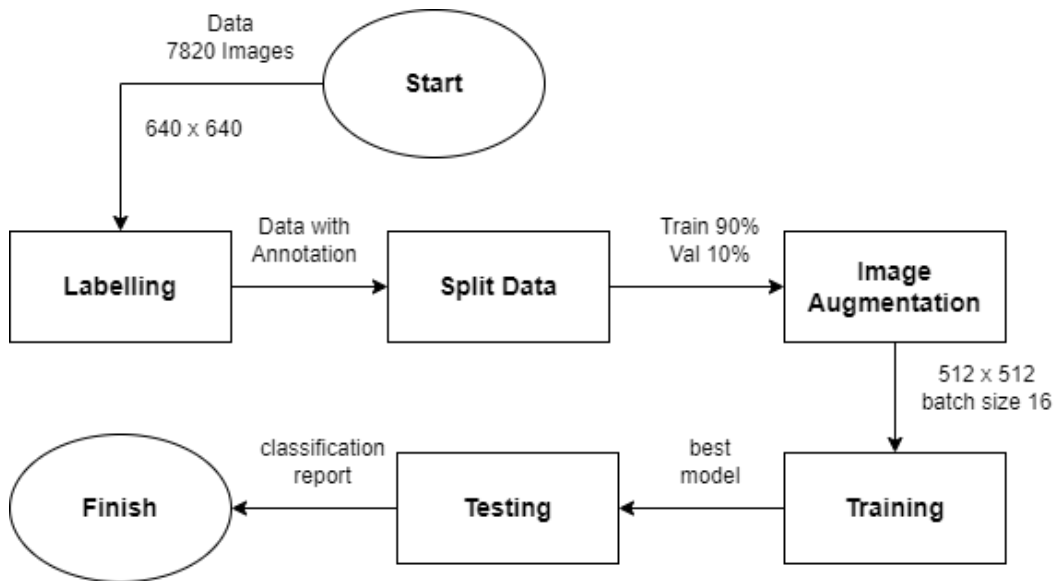


Gambar 3.3 Arsitektur Sistem

3.4.2. Diagram Proses Pembuatan Sistem

Diagram proses menggambarkan alur proses pembuatan model *deep learning* dari awal data di persiapan, sampai ke tahap pengujian model. Dimulai dengan *labelling*, yang mana pada tahap ini merupakan sebuah proses memberi *bounding box* dan anotasi atau label pada setiap *datasets*. Proses selanjutnya adalah *splitting data* atau membagi data ke dalam folder *train* dan *validation* dengan rasio 9 banding 1 atau 90% banding 10%. Proses selanjutnya adalah *Image Augmentation* atau dikenal sebagai augmentasi gambar, augmentasi gambar merupakan sebuah proses untuk memodifikasi data agar menjadi bentuk lain seperti ukuran yang awalnya 640 x 640 menjadi 512 x 512. Proses selanjutnya adalah *training*, yaitu melatih model agar dapat digunakan untuk mendeteksi, proses *training* menghasilkan 2 model yaitu *last* dan *best weights*. Tahap terakhir adalah pengujian atau *testing*, pengujian dilakukan untuk melihat seberapa tinggi

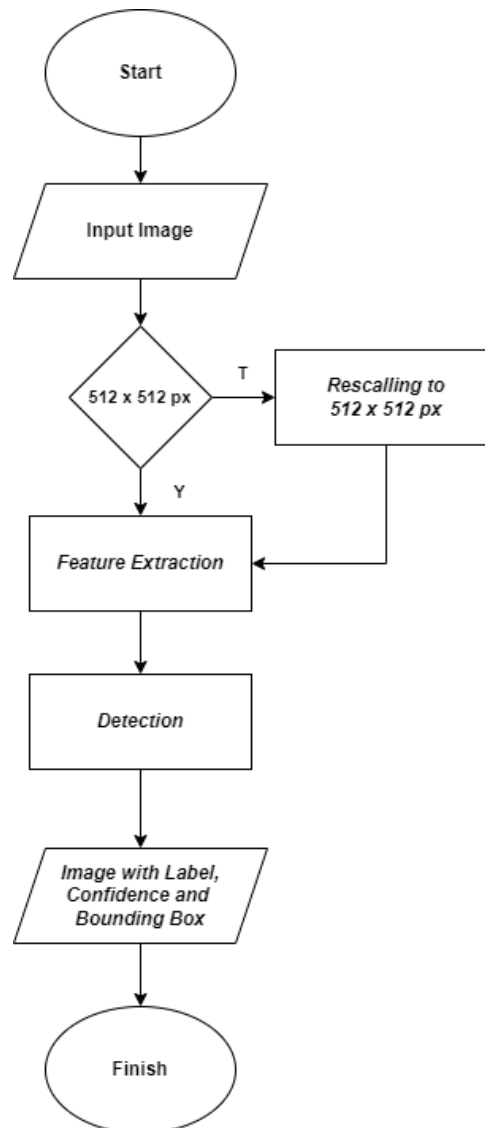
akurasi, model yang telah dilatih terhadap data baru. Untuk lebih jelasnya dapat dilihat pada Gambar 3.4 dibawah ini.



Gambar 3.4 Diagram Proses

3.4.3. Flowchart Sistem

Flowchart sistem pendeteksian diawali dengan menginputkan gambar yang dilakukan oleh petani. Gambar yang dimasukkan oleh petani harus berukuran 512x512 piksel, apabila tidak sesuai maka akan dilakukan proses *rescaling*. Selanjutnya gambar tersebut masuk ke proses ekstraksi fitur (*feature extraction*), lalu masuk ke tahap atau proses deteksi menggunakan algoritma YOLOv5 sebelum akhirnya didapatkan label, *confidence* atau kemiripan dan *bounding boxes*.

Gambar 3.5 *Flowchart Sistem*

3.5. Skenario Pengujian

Skenario pengujian model pada sistem deteksi dan klasifikasi tingkat kualitas tanaman padi ini yaitu melakukan pengujian terhadap datasets yang diambil secara langsung dari pertanian yang berada di Kabupaten Majalengka, Cirebon dan Indramayu. Pengujian ini bertujuan untuk memilih model terbaik yang akan digunakan atau diimplementasikan, model yang digunakan untuk pengujian yaitu YOLOv5s dan SSDMobileNetV2. *Output* yang dihasilkan adalah *Precision*, *Recall*, *mAP@0.5*, *mAP@.5:.95*, waktu deteksi dan ukuran file dari masing-masing model. Confusion Matrix adalah parameter yang dapat mengukur

kinerja model klasifikasi, baik itu klasifikasi biner (yes / no) maupun klasifikasi multi class.

Tabel 3.3.5. *Confusion Matrix* (Hidayatullah, P. 2021)

	Hasil Prediksi 0	Hasil Prediksi 1
Realita 0	TN	FP
Realita 1	FN	TP

Dari *confusion matrix* tersebut dapat digunakan untuk menentukan *precision* dan *recall*. Untuk menentukan *precision* bisa menggunakan persamaan dibawah ini (Hidayatullah, P. 2021).

$$\text{precision} = \frac{TP}{TP+FP} = \frac{TP}{DB} \quad (3.1)$$

Untuk menentukan *recall* bisa menggunakan persamaan dibawah ini (Hidayatullah, P. 2021).

$$\text{recall} = \frac{TP}{TP + FN} = \frac{TP}{GT} \quad (3.2)$$

Sedangkan untuk menentukan mAP (*mean Average Precision*) dapat menggunakan persamaan dibawah ini (Hidayatullah, P. 2021).

$$AP = \frac{1}{11} \times (AP_r(0) + AP_r(0,1) + \dots + AP_r(1,0)) \quad (3.3)$$

$$\text{mAP} = \frac{\sum AP}{\text{Jumlah Class}} \quad (3.4)$$

BAB 4

HASIL DAN PEMBAHASAN

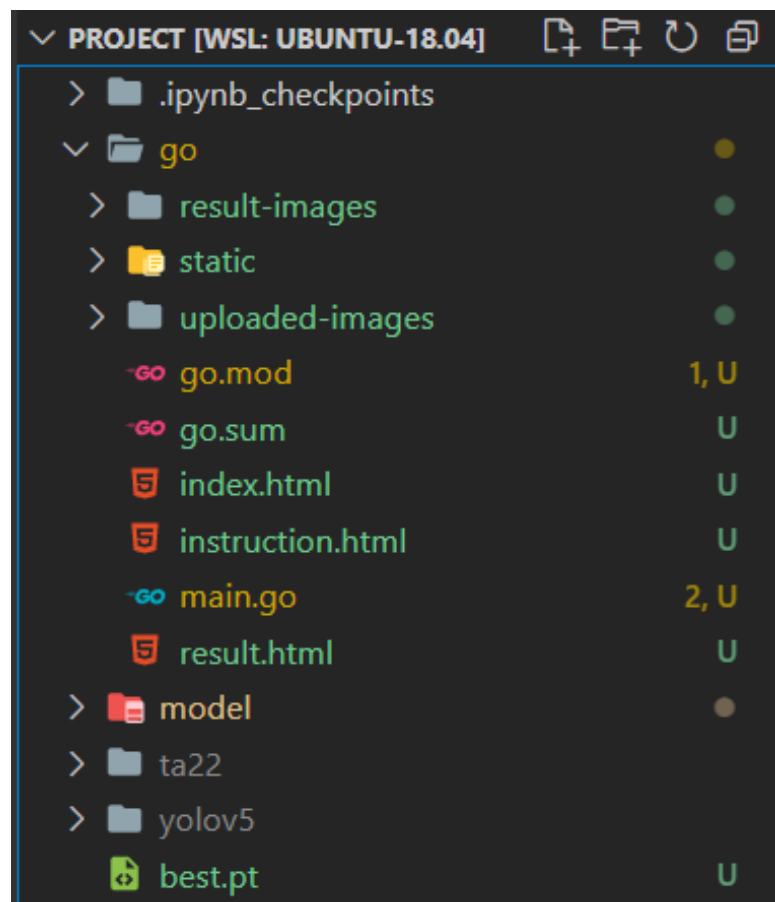
4.1. Hasil

Pada bagian ini merupakan tahap pembahasan dari hasil implementasi yang penulis kumpulkan, analisa dan rancangan sehingga tercipta sistem pendeteksi dan klasifikasi tingkat kualitas tanaman padi menggunakan algoritma YOLOv5.

4.2. Pembahasan

4.2.1. Struktur Direktori

Struktur direktori menjelaskan urutan atau *file-file* apa saja yang terdapat pada tiap-tiap *project* dalam membangun sistem pendeteksi dan klasifikasi tingkat kualitas tanaman padi.



Gambar 4.1 Struktur direktori *root Project* Sistem Deteksi

Pada Gambar 4.1 terdapat struktur direktori *root* dari sistem yang penulis rancang. Pada *folder root* ini terdapat *folder go* yang berisi sebuah sistem *back-end* yang dibangun menggunakan bahasa pemrograman go, pada *folder go* terdapat *folder result-images* yang berisi semua gambar yang telah berhasil dideteksi oleh Algoritma YOLOv5s, selanjutnya ada *folder static* yang berisi *file* untuk *styling* tampilan *dashboard* seperti *css* dan *js*, lalu yang terakhir adalah *folder uploaded-images* yang berisi semua gambar yang telah diunggah oleh para petani yang nantinya akan dideteksi oleh Algoritma YOLOv5s. Selanjutnya pada *folder root* ini terdapat *folder model*, yang berisi *datasets* dan semua hasil *experiment* Algoritma Deep Learning seperti *Notebook*, *Classification Report*, *Confusion Matrix* dan *Model Weights*. *Folder ta22* merupakan *virtual environment* yang dibuat untuk keperluan *experiment* Algoritma Deep Learning. Selanjutnya *folder yolov5* yang berisi file-file hasil *cloning* dari *repository* asli, *folder* ini berfungsi untuk pelatihan model menggunakan Algoritma YOLOv5. Selanjutnya adalah *file best.pt*, *file* tersebut adalah model terbaik yang telah dipilih berdasarkan *score precision*, *recall*, *f1-score*, *mAP:0.5*, waktu deteksi dan ukuran *file*.

4.2.2. Implementasi Persiapan Data

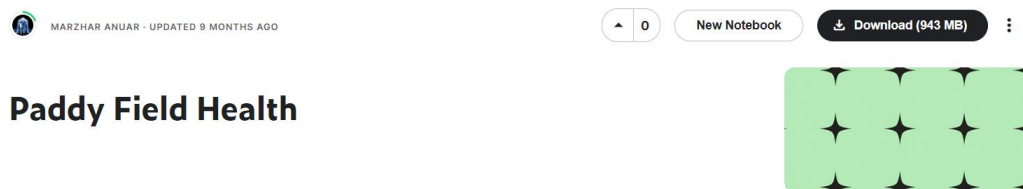
Pendeteksian dan klasifikasi tingkat kualitas tanaman padi dilakukan dengan cara menentukan terlebih dahulu kelas atau kategori tingkat kualitas yang diperoleh selama pencarian data. Tingkat Kualitas yang ditemukan selama pencarian hanya 2 tingkat kualitas. Tingkat Kualitas tersebut penulis paparkan pada Tabel 4.1 dibawah ini.

Tabel 4.4.1. Tabel Tingkat Kualitas

No	Tingkat Kualitas	Label
1	Baik / Sehat	good
2	Buruk / Tidak Sehat	bad

4.2.2.1. Pengumpulan *Dataset*

Data tanaman padi yang penulis gunakan sebagai *dataset* didapatkan dari *platform* penyedia *datasets* ternama yaitu Kaggle. *Datasets* tersebut di *upload* oleh *user* bernama Marzhar Anuar pada tanggal 18 Juli 2021, *datasets* tersebut berukuran 942.14 Mb (*Megabyte*) yang berisi 2 *folder* utama yaitu Healthy dan Unhealthy. Pada *folder* Healthy berisi 3610 Gambar dengan ukuran 640 x 640 *pixels*, dan *folder* UnHealthy berisi 4210 Gambar dengan ukuran 640 x 640 *pixels*. Data tersebut merupakan gambar tanaman padi yang diambil melalui citra udara atau *drone*.



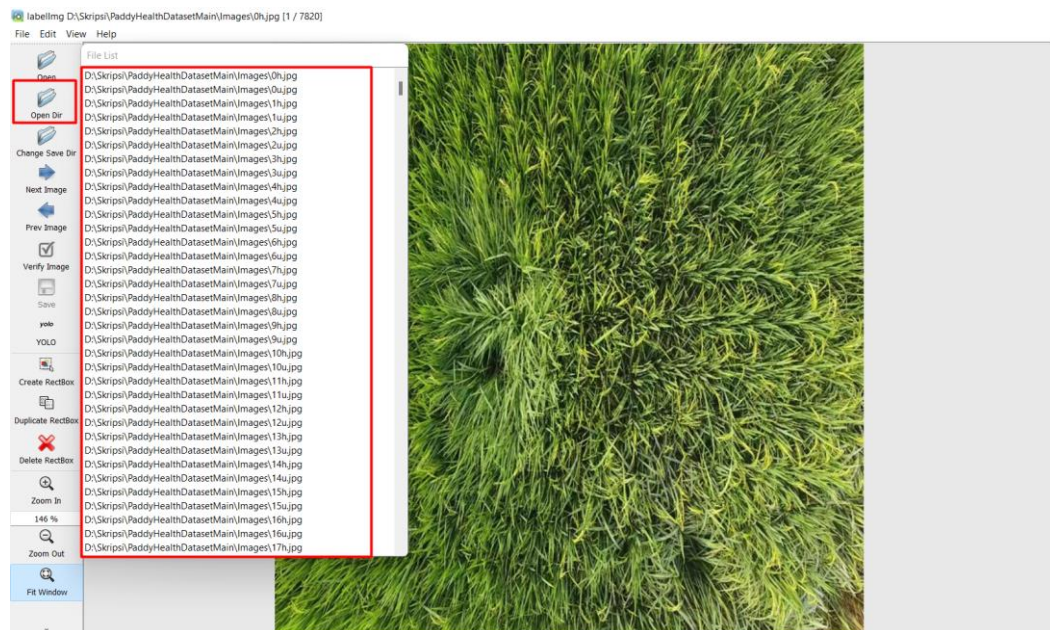
Gambar 4.2 Kaggle *Datasets*

4.2.2.2. Persiapan Data

Setelah data terkumpul, tahap selanjutnya adalah tahap persiapan data. Persiapan data yang harus dilakukan sebelum masuk ke tahap proses *training* model diantaranya yaitu *Labeling*, *Splitting* dan *Preprocessing*.

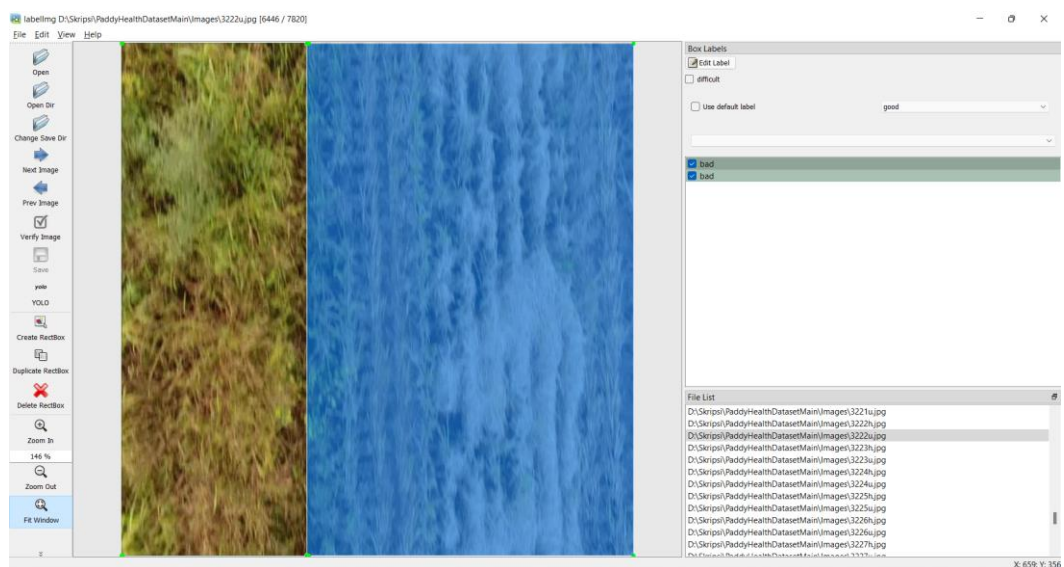
a. *Labeling*

Labeling adalah sebuah proses pemberian anotasi label terhadap objek yang menjadi target pada gambar berdasarkan *class* nya. Pada sistem yang penulis buat, pendeteksian gejala dapat dilakukan lebih dari satu objek dalam satu gambar (*Multilabel*). Proses *labeling* dilakukan menggunakan *tools/aplikasi* LabelImg. LabelImg adalah *tools* yang gratis dan *open source* berbasis *desktop*, tidak hanya membuat anotasi untuk format YOLO, *labelimg* juga bisa membuat anotasi untuk format lainnya seperti *Pascal / VOC* yang akan menghasilkan anotasi dalam bentuk file XML dan format CreateML yang akan menghasilkan anotasi dalam bentuk file JSON. Sebelum melakukan proses anotasi, terlebih membuka *folder* yang berisi *datasets* yang telah dimasukan ke dalam 1 *folder*. Untuk lebih jelasnya dapat dilihat pada gambar dibawah ini.



Gambar 4.3 *Directory File List* LabellImg

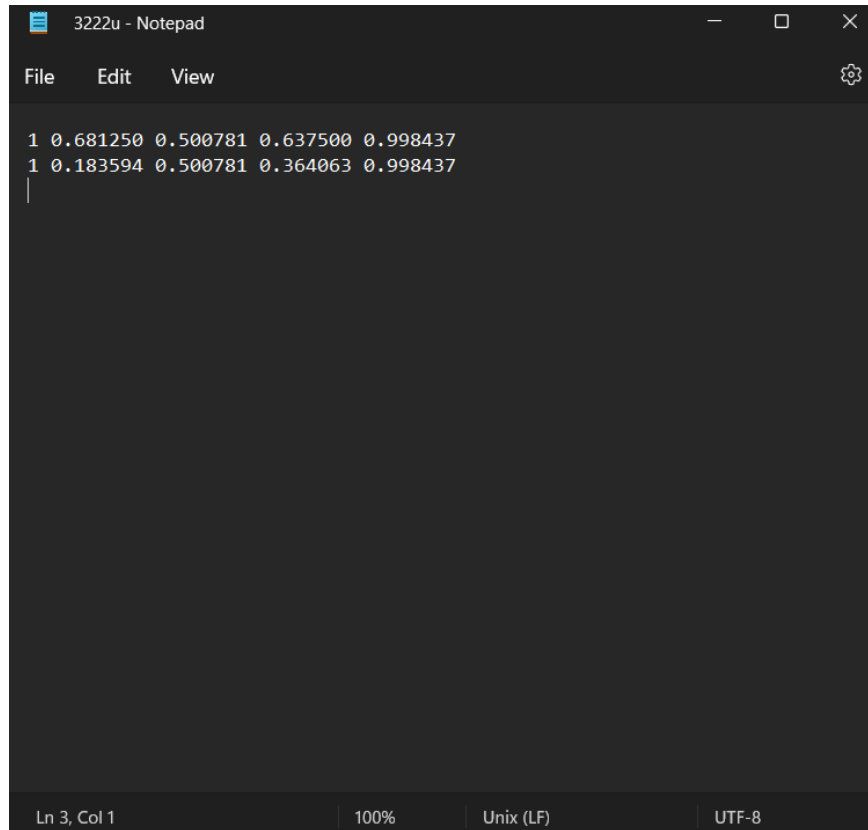
Setelah *task* dibuat, proses selanjutnya adalah pembuatan anotasi pada data yang sudah penulis masukan ke dalam *Directory File List*. Untuk lebih jelasnya dapat dilihat pada gambar dibawah ini.



Gambar 4.4 Proses *Labelling* Menggunakan LabellImg

Pada gambar diatas bisa dilihat bahwa terdapat 1 tingkat kualitas yaitu buruk / tidak sehat (bad). Maka pada objek tersebut penulis tandai menggunakan *rectangle* atau *bouding box*, lalu pilih sesuai *class* nya. Setelah membuat *bouding box* beserta pilihan *class* nya LabellImg akan otomatis membuat hasil

anotasi nya dengan format YOLO. Hasil anotasi tersebut akan menghasilkan sebuah file dengan ekstensi txt sesuai dengan nama *file* gambar yang dibuat anotasinya, untuk lebih jelasnya dapat dilihat pada gambar dibawah ini.



Gambar 4.5 *File* Hasil Anotasi

b. *Splitting*

Setelah data selesai melalui proses *labelling*, tahap selanjutnya sebelum masuk ke proses *training* dataset tersebut harus dibagi menjadi *training set* dan *validation set*. Pembagian data ini bertujuan agar model yang dibuat tidak *overfitting*. *Overfitting* terjadi saat model mendapatkan akurasi atau performa yang baik pada data *training*, namun mendapatkan akurasi atau performa yang buruk ketika menggunakan data yang belum pernah dilihat sebelumnya atau data *testing*.

Untuk pembagian dataset dalam pembuatan sistem deteksi dan klasifikasi tingkat kualitas tanaman padi ini penulis membagi dataset menjadi dua bagian. Pembagian terbanyak terdapat pada data *training* karena proses

training sangat penting untuk mendapatkan performa terbaik. Untuk pembagian *dataset* akan penulis jelaskan pada tabel dibawah ini.

Tabel 4.2 Tabel Data *Splitting*

<i>Data Training</i>	90%	7038
<i>Data Validation</i>	10%	782
Total Data	100%	7820

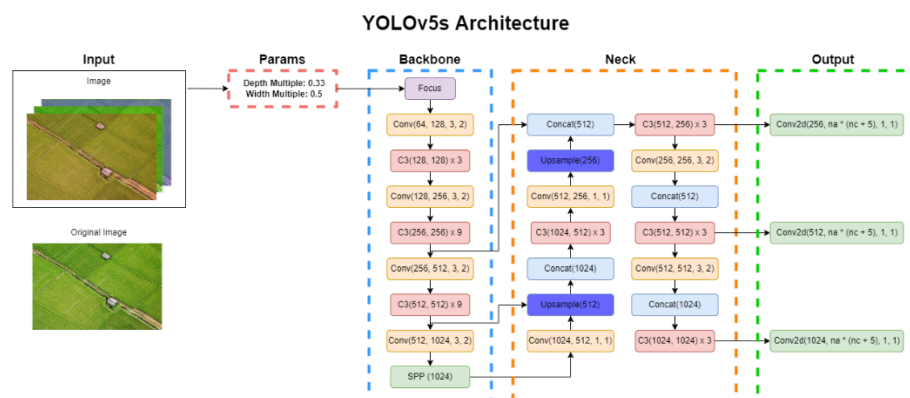
c. *Preprocessing*

Pada tahap *preprocessing* ini hanya mengubah ukuran gambar yang sebelumnya berukuran 640x640 *pixel* menjadi 512x512 *pixel*. Nilai tersebut adalah ukuran yang digunakan penulis untuk konfigurasi Model YOLOv5 dan SSDMobileNetV2. Hal ini bertujuan agar mempercepat proses *training*.

4.2.3. Training Model

4.2.3.1. Hasil Training YOLOv5s

Proses *training*, merupakan sebuah proses untuk melatih model menggunakan *neural network* / jaringan saraf tiruan. Sebelum melatih model kita perlu membuat arsitektur model terlebih dahulu, karena YOLO merupakan *pre-trained model* / model yang sudah dilatih pada dataset besar seperti COCO atau ImageNet maka kita hanya akan menggunakan arsitektur nya saja yang berisi *layer neural network*. Berikut adalah gambar dari arsitektur YOLOv5s.



Gambar 4.6 Arsitektur YOLOv5s

Arsitektur YOLOv5s terbagi menjadi 3 bagian utama yaitu *Backbone*, *Neck* dan *Output Layer*. *Backbone Layer* berfungsi mengekstrak informasi penting dari gambar sekaligus mengurangi model yang kompleks. Selanjutnya adalah *Neck Layer*, *layer* ini berfungsi membuat piramida fitur, piramida fitur membantu model untuk mengidentifikasi objek yang sama dengan ukuran dan skala yang berbeda dengan cara digeneralisasi dengan baik pada penskalaan objeknya. Informasi yang dihasilkan oleh backbone layer, informasi tersebut akan di proses oleh neck layer sehingga menghasilkan informasi yang berguna di setiap level fitur yang akan menyebar langsung ke sub-jaringan. *Output Layer* berfungsi untuk melakukan pendeteksian pada bagian akhir, *layer* ini menghasilkan *box*, *confidence* / akurasi dan kelas pada objek yang terdeteksi pada gambar. Pada Bagian ini model akan menghasilkan 3 ukuran *layer* yang berbeda untuk mencapai deteksi *multi-scale*, deteksi *multi-scale* akan memastikan bahwa model dapat mengikuti perubahan ukuran.

Untuk menjalankan proses *training* dengan cara menuliskan perintah “!<python> <train.py> <batch> <epoch> <data> <weights> <cache> <nosave>”. Berikut ini adalah gambar proses *training* model YOLOv5s.

TRAIN

```
In [5]: # Train YOLOv5s
!python train.py --img 500 --batch 16 --epochs 5 --data cstm.yaml --weights yolov5s.pt --cache --nosave
```

	all	782	798	0.992	0.973	0.976	0.934
Epoch	gpu_mem	box	obj	cls	labels	img_size	
2/4	2.39G	0.01766	0.009152	0.002696	41	512: 100% 440/440 [05:15<00:00, 1.39it/s]	
Class	Images	Labels	P	R	mAP@.5	mAP@.5:.95: 100% 25/25 [00:12<00:00, 2.06it/s]	
all	782	798	0.983	0.966	0.973	0.934	
Epoch	gpu_mem	box	obj	cls	labels	img_size	
3/4	2.39G	0.01335	0.007838	0.001506	54	512: 100% 440/440 [05:14<00:00, 1.40it/s]	
Class	Images	Labels	P	R	mAP@.5	mAP@.5:.95: 100% 25/25 [00:11<00:00, 2.09it/s]	
all	782	798	0.992	0.973	0.975	0.941	
Epoch	gpu_mem	box	obj	cls	labels	img_size	
4/4	2.39G	0.01205	0.007279	0.002134	42	512: 100% 440/440 [05:14<00:00, 1.40it/s]	
Class	Images	Labels	P	R	mAP@.5	mAP@.5:.95: 100% 25/25 [00:12<00:00, 2.08it/s]	
all	782	798	0.991	0.973	0.98	0.944	

5 epochs completed in 0.457 hours.
Optimizer stripped from runs/train/exp/weights/last.pt, 14.4MB
Optimizer stripped from runs/train/exp/weights/best.pt, 14.4MB

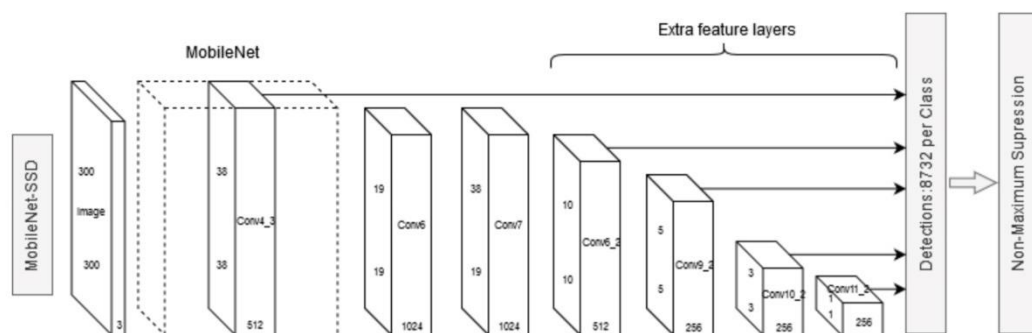
Gambar 4.7 Proses *Training* Model YOLOv5s

Proses *training* pada model pertama selesai dalam waktu 0,457 jam dengan hasil yang bisa dikatakan sudah baik. Untuk lebih jelasnya, penulis paparkan hasil *training* tersebut pada Tabel 4.3 di bawah ini.

Tabel 4.3 Kinerja Hasil *Training* Model YOLOv5s

No	Variabel Kinerja	Score
1	<i>Box_loss</i>	0,012
2	<i>Obj_loss</i>	0,007
3	<i>Cls_loss</i>	0,002
4	<i>Precision</i>	0,991
5	<i>Recall</i>	0,973
6	<i>mAP@0,5</i>	0,979
7	<i>mAP@0,5:0.95</i>	0,944

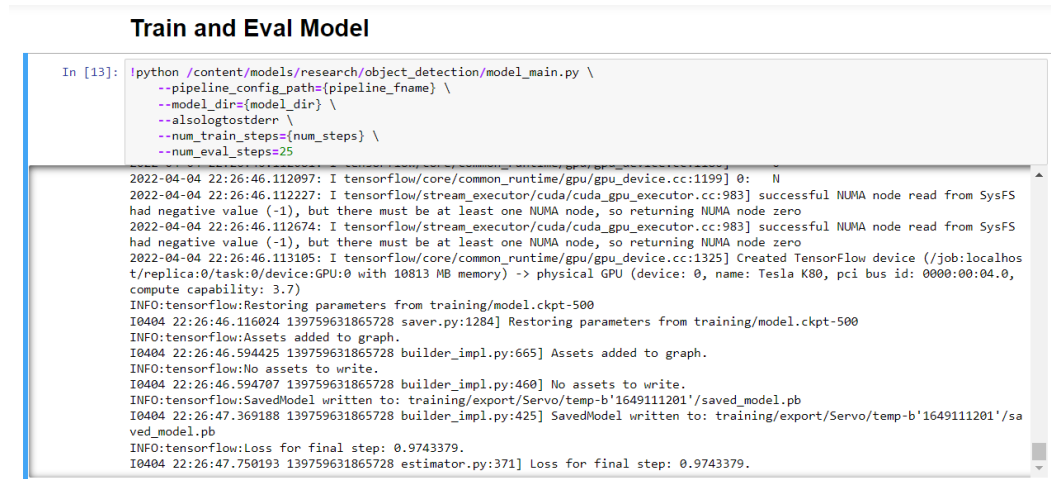
4.2.3.2. Hasil Training SSDMobileNetV2



Gambar 4.8 Arsitektur SSDMobileNetV2

Arsitektur SSDMobileNetV2 terdiri dari SSD yang berperan sebagai base model dan MobileNet sebagai *Network Model*. SSD akan mengatur pendeteksian objek dengan membuat *bounding box*. MobileNet akan bekerja untuk mengekstrak fitur yang nantinya akan diklasifikasi. Penggabungan SSD dan Mobilenet akan membantu dalam proses pembuatan aplikasi deteksi objek. Dalam aplikasi deteksi objek dibutuhkan SSD untuk membuat lokalisasi gambar untuk menentukan posisi objek. Sedangkan Mobilenet akan dibutuhkan untuk membantu mengklasifikasi objek yang terdapat dalam suatu gambar. Klasifikasi akan menghasilkan kategori untuk masing-masing objek yaitu Good dan Bad.

Untuk menjalankan proses *training* dengan cara menuliskan perintah “!`<python>` `<model_main.py>` `<pipeline_config_path>` `<model_dir>` `<alsologtostderr>` `<num_train_steps>` `<num_eval_steps>`”. Berikut ini adalah gambar proses *training* model SSDMobileNetV2.



```

Train and Eval Model

In [13]: !python /content/models/research/object_detection/model_main.py \
--pipeline_config_path={pipeline_fname} \
--model_dir={model_dir} \
--alsologtostderr \
--num_train_steps={num_steps} \
--num_eval_steps=25

2022-04-04 22:26:46.112097: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1199] 0:  N
2022-04-04 22:26:46.112227: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:983] successful NUMA node read from SysFS
had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2022-04-04 22:26:46.112674: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:983] successful NUMA node read from SysFS
had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2022-04-04 22:26:46.113105: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1325] Created TensorFlow device (/job:localhos
t/replica:0/task:0/device:GPU:0 with 10813 MB memory) -> physical GPU (device: 0, name: Tesla K80, pci bus id: 0000:00:04.0,
compute capability: 3.7)
INFO:tensorflow:Restoring parameters from training/model.ckpt-500
I0404 22:26:46.116024 139759631865728 saver.py:1284] Restoring parameters from training/model.ckpt-500
INFO:tensorflow:Assets added to graph.
I0404 22:26:46.594425 139759631865728 builder_impl.py:665] Assets added to graph.
INFO:tensorflow:No assets to write.
I0404 22:26:46.594707 139759631865728 builder_impl.py:460] No assets to write.
INFO:tensorflow:SavedModel written to: training/export/Servo/temp-b'1649111201'/saved_model.pb
I0404 22:26:47.369188 139759631865728 builder_impl.py:425] SavedModel written to: training/export/Servo/temp-b'1649111201'/sa
ved_model.pb
INFO:tensorflow:Loss for final step: 0.9743379.
I0404 22:26:47.750193 139759631865728 estimator.py:371] Loss for final step: 0.9743379.

```

Gambar 4.9. Proses *Training* Model SSDMobileNetV2

Proses *training* pada model pertama selesai dalam waktu 0,25 jam dengan hasil yang bisa dikatakan cukup baik. Untuk lebih jelasnya, penulis paparkan hasil *training* tersebut pada Tabel 4.4 di bawah ini

Tabel 4.4 Kinerja Hasil *Training* Model SSDMobileNetV2

No	Variabel Kinerja	Score
1	<i>Classification_loss</i>	6,579
2	<i>Localization_loss</i>	2,467
3	<i>Regularization_loss</i>	0,307
4	<i>Precision</i>	0,850
5	<i>Recall</i>	0,907
6	<i>mAP@0,5</i>	0,946
7	<i>mAP@0,5:0.95</i>	0,851

Ada beberapa hal yang perlu diperhatikan setelah melakukan proses *training model*, yaitu membuat *classification report* untuk memilih model terbaik yang nantinya akan digunakan oleh sistem. Ada beberapa parameter yang menjadi tolak ukur untuk memilih model terbaik seperti *score precision*, *recall*, *f1-score*, *mAP:0,5*, waktu deteksi dan ukuran *file*.

4.2.4. Pengujian Model

Datasets untuk pengujian diambil langsung dari pertanian yang berada di daerah Kabupaten Majalengka, Cirebon dan Indramayu, ada sekitar 100 datasets dengan 111 *bounding box* pada gambar, diantaranya 40 gambar sawah kualitas bagus dengan 47 *bounding box* dan 60 gambar sawah kualitas rusak / jelek dengan 64 *bounding box* yang diambil menggunakan teknologi *drone*. Pengujian ini bertujuan untuk memilih model terbaik yang akan digunakan atau diimplementasikan, model yang digunakan untuk pengujian yaitu YOLOv5s dan SSDMobileNetV2. *Output* yang dihasilkan adalah *Precision*, *Recall*, *mAP@0.5*, *mAP@.5:.95*, waktu deteksi dan ukuran file dari masing-masing model.

4.2.4.1. Perbandingan Hasil Pengujian

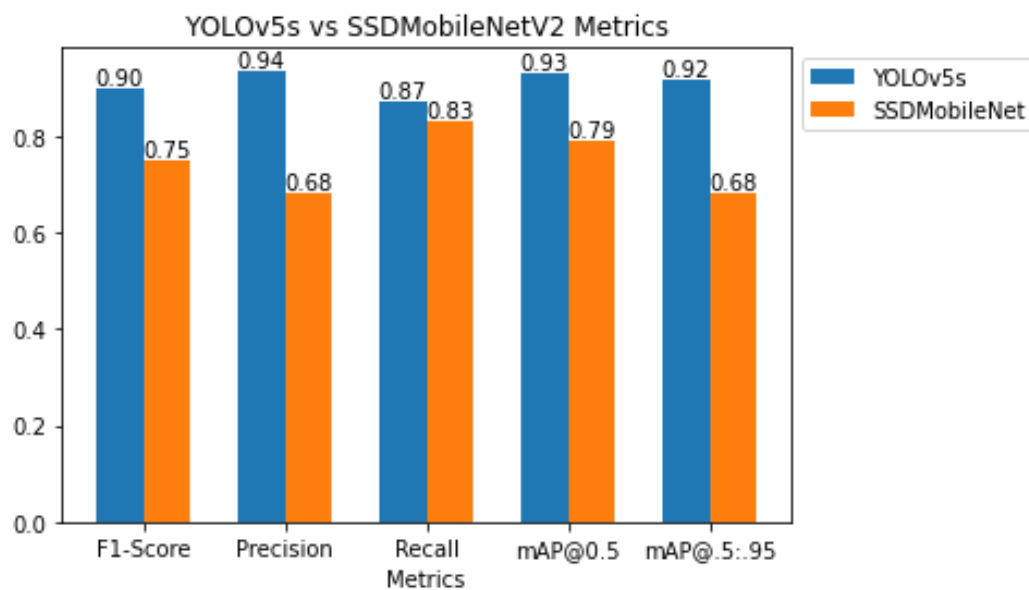
Berikut merupakan tabel hasil perbandingan antara model YOLOv5s dan SSDMobileNetV2.

Tabel 4.5 Tabel Perbandingan Hasil Uji Coba Antara Model YOLOv5s dan YOLOv5m.

No	Variabel Kinerja	YOLOv5s	SSDMobileNetV2
2	F1-Score	0,90	0,750
3	Precision	0,936	0,683
4	Recall	0,87	0,832
5	mAP@0.5	0,93	0,790
6	mAP@.5:.95	0,916	0,683
7	Waktu Deteksi	0,063s	0,144s

8	Ukuran File	14,4Mb	18,4Mb
---	-------------	--------	--------

Penulis memutuskan untuk menggunakan model YOLOv5s untuk penelitian ini, YOLOv5s dapat bekerja lebih cepat dibanding SSDMobileNetV2 serta waktu deteksi dan ukuran file dari model nya pun lebih kecil. YOLOv5s memiliki keunggulan pada nilai *F1-Score*, *Precision*, *Recall* dan mAP(*mean average precision*) dibuktikan dengan gambar grafik perbandingan berikut.

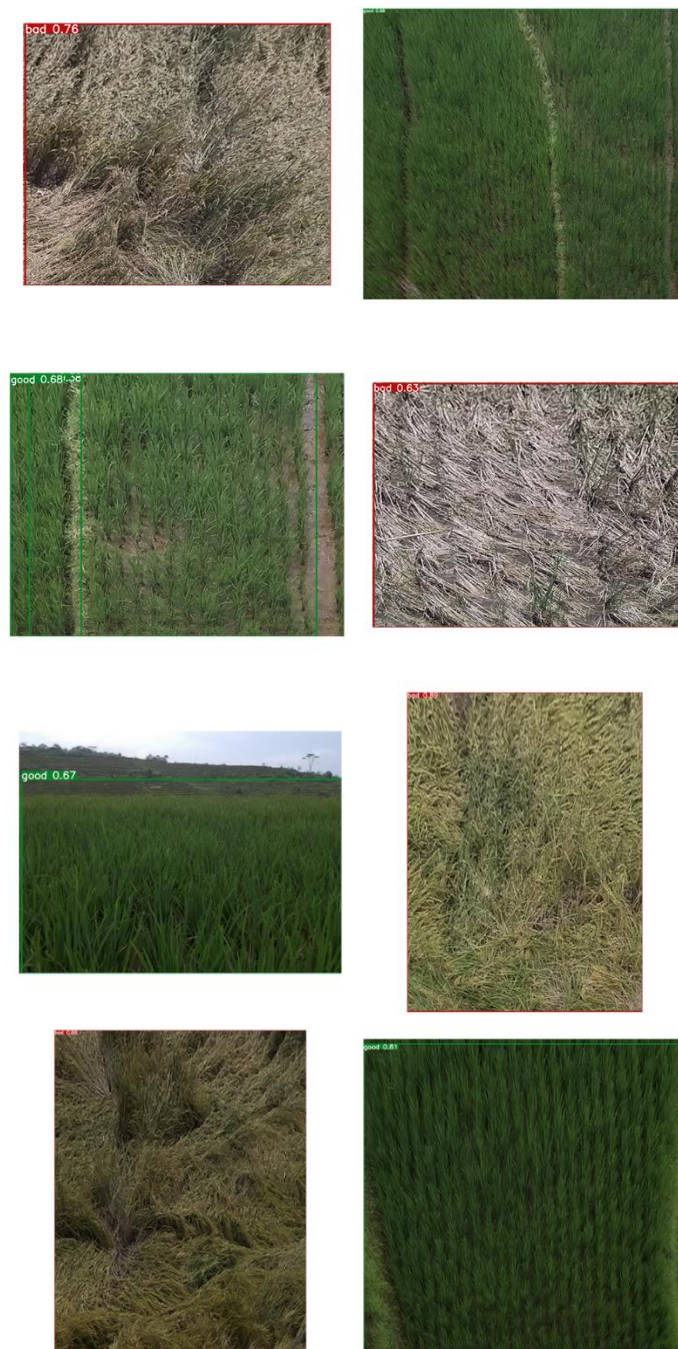


Gambar 4.10 YOLOv5s vs SSDMobileNetV2 Metrics

Gambar diatas merupakan grafik perbandingan dari masing-masing model terhadap *metrics* yang digunakan oleh kedua model yaitu COCO. Dapat dilihat pada gambar diatas, YOLOv5s selalu unggul di semua *metrics* yang digunakan, ini membuktikan bahwa YOLOv5s terbukti lebih baik dalam memprediksi datasets yang diambil langsung dari lapangan. Waktu deteksi YOLOv5s pun lebih cepat dibandingkan waktu deteksi SSDMobileNetV2, ini berarti FPS yang dihasilkan akan jauh lebih tinggi dan waktu komputasi pun tidak akan terlalu lama. Maka dari itu model YOLOv5s merupakan pilihan terbaik untuk mendeteksi tingkat kualitas tanaman padi.

4.2.4.2. Pengujian Deteksi Tingkat Kualitas Tanaman Padi

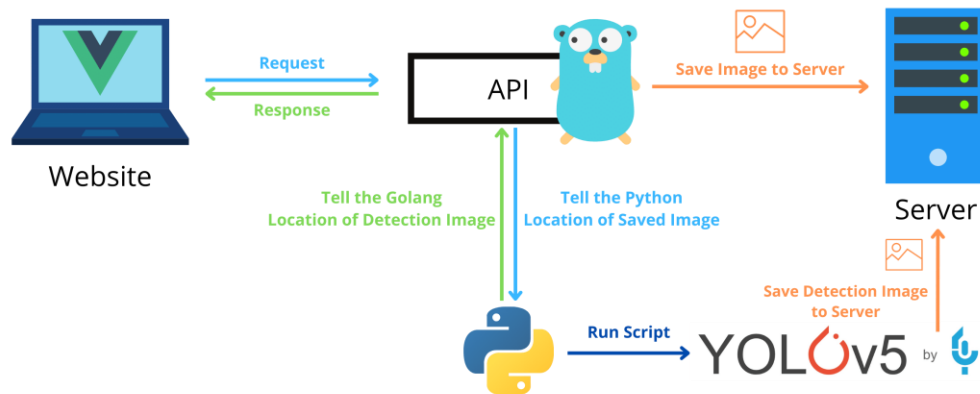
Setelah dibandingkan dan mendapat kesimpulan, bahwa model yang menggunakan arsitektur YOLOv5s adalah yang terbaik untuk mendeteksi tingkat kualitas tanaman padi. Model YOLOv5s menjadi acuan untuk sistem deteksi ini, berikut beberapa hasil deteksi yang dilakukan:



Gambar 4.11 Deteksi Tingkat Kualitas Tanaman Padi menggunakan YOLOv5s

4.2.5. Hasil Integrasi Sistem

Pada bagian ini akan dibahas hasil dari integrasi antara sistem yang penulis buat dengan Aplikasi SiPetaniCerdas (Sistem Informasi Petani Cerdas). Berikut adalah Gambar Arsitektur Sistem yang digunakan.



Gambar 4.12 Arsitektur Sistem

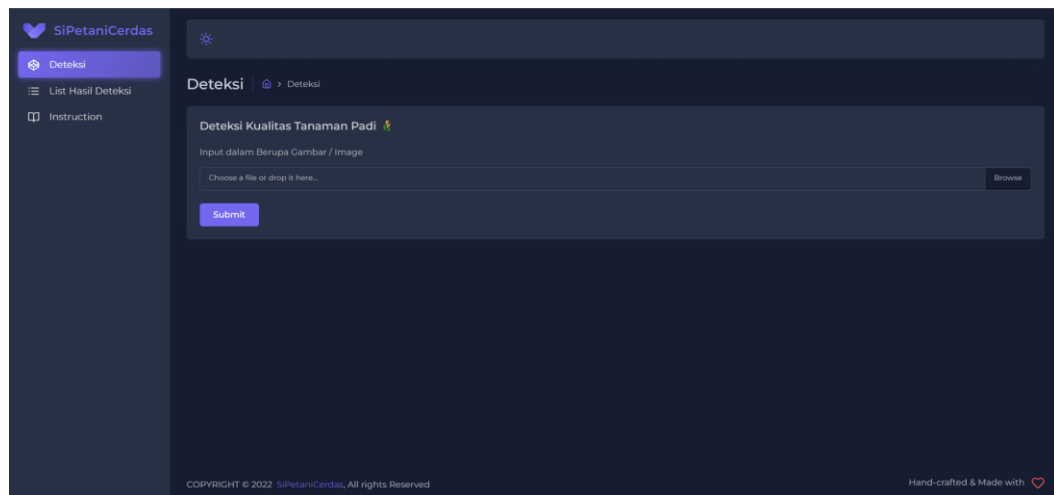
Seperti yang digambarkan diatas bahwa untuk menjembatani hubungan antara *server* dengan Aplikasi SiPetaniCerdas yaitu dengan menggunakan API sebagai perantara. *Website* mengirimkan gambar menggunakan format *form-data* ke API yang dibuat menggunakan golang, gambar tersebut akan disimpan ke dalam *server*, dan lokasi gambar yang disimpan akan dimasukkan ke dalam format eksekusi *script* python. Selanjutnya setelah format *script* python terbentuk, golang akan membuat perintah untuk mengeksekusi nya melalui *command-line*. Setelah *script* python dijalankan, *script* tersebut akan mengeksekusi perintah untuk mendeteksi tingkat kualitas tanaman padi menggunakan model YOLOv5s, selanjutnya *script* akan menyimpan hasil deteksi ke dalam bentuk gambar yang akan disimpan di *server*, *script* akan mengeluarkan *output* berupa lokasi gambar pada *server* yang telah dideteksi. *Output* berisi lokasi gambar yang telah dideteksi tersebut akan digunakan golang untuk mengembalikan *response* kepada *website*, dan nantinya akan ditampilkan oleh *website* menggunakan *framework* vue-js. Pada Aplikasi SiPetaniCerdas, pengecekan tingkat kualitas tanaman padi dilakukan secara otomatis dengan hanya mengunggah gambar. Metode tersebut

sangat cocok diterapkan pada fitur yang membutuhkan input manual dari petani. Adapapun fitur dan format sebagai berikut.

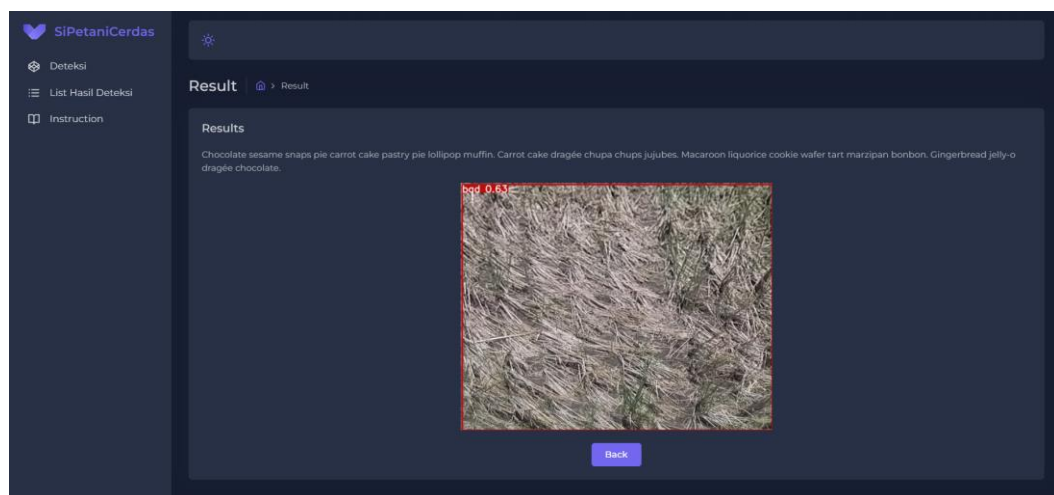
API Format



Gambar 4.13 Format API



Gambar 4.14 Halaman *Form* Unggah Gambar



Gambar 4.15 Halaman Hasil Deteksi Tingkat Kualitas Tanaman Padi

BAB V

PENUTUP

5.1 Kesimpulan

Kesimpulan yang didapat setelah mengumpulkan, menganalisa, mengimplementasikan dan melakukan pengujian sistem deteksi dan klasifikasi tingkat kualitas tanaman padi menggunakan Algoritma YOLOv5 adalah sebagai berikut:

1. Sistem pendeteksian tingkat kualitas tanaman padi dibangun menggunakan arsitektur atau Algoritma YOLOv5. Sistem dapat mendeteksi 2 tingkat kualitas berdasarkan hasil training menggunakan dataset yang telah dikumpulkan
2. Sistem yang dibangun menggunakan algoritma YOLOv5 diimplementasikan pada sistem Petani Cerdas berbasis Web
3. Akurasi (mAP@0,5) yang didapatkan dengan menggunakan model YOLOv5 yaitu sebesar 0,979 atau 97,9% dan hasil ini sudah cukup baik

5.2 Saran dikembangkan

Pada sistem deteksi dan klasifikasi tingkat kualitas tanaman padi ini masih terdapat beberapa kekurangan yang sudah penulis sebutkan pada bagian kesimpulan. Oleh karena itu penulis menyarankan untuk pengembangan selanjutnya agar dapat mempertimbangkan saran yang diberikan agar sistem ini lebih baik lagi dari segi akurasi maupun performa. Saran-saran tersebut antara lain sebagai berikut:

1. Sistem deteksi Perlu di integrasikan menggunakan Programming Drone (Mapping).
2. Drone harus bisa mengirim datanya langsung ke dalam Sistem.

DAFTAR PUSTAKA

- Aningtiyas, P. R., dkk. (2020). Pembuatan Aplikasi Deteksi Objek Menggunakan Tensorflow Object Detection API dengan Memanfaatkan SSD MobileNet V2 Sebagai Model Pra-Terlatih. *Jurnal Ilmiah Komputasi*. 19(1).
- Ahmad, A. (2017). *Mengenal artificial intelligence, machine learning, neural network, dan deep learning*. J. Teknol. Indonesia., no. Oktober, 3.
- Badan Pusat Statistik. (2019). Tingkat Pengangguran Terbuka (TPT) sebesar 5,28 Persen. Jakarta: Berita Resmi Statistik. Diakses pada 25 Januari 2021, dari <https://www.bps.go.id/pressrelease/2019/11/05/1565/agustus-2019--tingkat-pengangguran-terbuka--tpt--sebesar-5-28-persen.html>.
- Farhadi, A., & Redmon, J. (2018). *Yolov3: An incremental improvement*. *Computer Vision and Pattern Recognition*, cite as.
- Faza, S. (2018). *Peningkatan Kinerja dalam Pengklasifikasian Menggunakan Deep Learning*.
- Hidayatullah, P (2021). Buku Sakti *Deep Learning Computer Vision Menggunakan YOLO untuk Pemula*.
- Kusumanto, R. D., & Tompunu, A. N. (2011). *Pengolahan citra digital untuk mendeteksi obyek menggunakan pengolahan warna model normalisasi RGB*. Semantik, 1(1).
- LeCun, Y., Bengio, Y., & Hinton, G. (2015). *Deep learning*. nature, 521(7553), 436-444.
- Nurhikmat, T. (2018). *Implementasi Deep Learning Untuk Image Classification Menggunakan Algoritma Convolutional Neural Network (CNN) Pada Citra Wayang Golek*.
- Putri, R. K. S. C. (2018). *Implementasi Deep Learning Menggunakan Metode Convolutional Neural Network Untuk Klasifikasi Gambar (Studi Kasus: Klasifikasi Gambar Pada Tanaman Anggrek Bulan Putih, Anggrek Dendrobium, dan Anggrek Ekor Tupai)*
- R, Aditya. (2018). *You Only Look Once*. Diakses pada 9 Maret 2021, dari <https://machinelearning.mipa.ugm.ac.id/2018/08/05/yolo-you-only-look-once/>.
- Sanotsa, Edi dan Winarno Tohir. (2019). *Pertanian Presisi, Menuju Pertanian 4.0 Berkesejahteraan*. Diakses pada 12 Maret 2021, dari <http://himapasca.fp.ub.ac.id/pertanian-presisi-menuju-pertanian-4-0-berkesjahteraan/>.

- Setiabudidaya. (2018). *Penggunaan Piranti Lunak Jupyter Notebook dalam Upaya Mensosialisasikan Open Science*. Palembang: Jurusan Fisika Universitas Sriwijaya.
- Tech Empower. (2021). *Web Framework Benchmarks*. Tersedia di <https://www.techempower.com/benchmarks/> [Diakses pada tanggal 26 Maret 2022].
- Tuluran, J. (2020). *Sistem Deteksi Pakan Udang Dengan Metode Deep Learning* (Doctoral dissertation, Universitas Hasanuddin).
- Yang, Dingming, dkk. (2021). *Deep Learning Based Steel Pipe Weld Defect Detection*.
- Zain, F. H., (2021). *Sistem Deteksi Kerusakan Gedung Menggunakan Algoritma YOU ONLY LOOK ONCE dengan Unamed Aero Vehicle*. Jakarta: Jurusan Teknik Elektro Politeknik Negeri Jakarta.

LAMPIRAN

LAMPIRAN 1. KODE PROGRAM

Predict Object

```
# YOLOv5 🚀 by Ultralytics, GPL-3.0 license
"""
Run inference on images, videos, directories, streams, etc.

Usage - sources:
    $ python path/to/detect.py --weights yolov5s.pt --source 0
# webcam

# image
img.jpg

# video
vid.mp4

# directory
path/

# glob
path/*.jpg
"""

import argparse
import os
import sys
from pathlib import Path

import cv2
import torch
import torch.backends.cudnn as cudnn

FILE = Path(__file__).resolve()
ROOT = FILE.parents[0] # YOLOv5 root directory
if str(ROOT) not in sys.path:
    sys.path.append(str(ROOT)) # add ROOT to PATH
ROOT = Path(os.path.relpath(ROOT, Path.cwd())) # relative
```

```

from models.common import DetectMultiBackend

from utils.datasets import IMG_FORMATS, VID_FORMATS, LoadImages,
LoadStreams

from utils.general import (LOGGER, check_file, check_img_size,
check_imshow, check_requirements, colorstr,
                           increment_path, non_max_suppression,
print_args, scale_coords, strip_optimizer, xyxy2xywh)
from utils.plots import Annotator, colors, save_one_box
from utils.torch_utils import select_device, time_sync

@torch.no_grad()
def run(weights=ROOT / 'yolov5s.pt', # model.pt path(s)
        source=ROOT / 'data/images', # file/dir/URL/glob, 0 for
webcam
        data=ROOT / 'data/coco128.yaml', # dataset.yaml path
        imgsz=(640, 640), # inference size (height, width)
        conf_thres=0.25, # confidence threshold
        iou_thres=0.45, # NMS IOU threshold
        max_det=1000, # maximum detections per image
        device='', # cuda device, i.e. 0 or 0,1,2,3 or cpu
        view_img=False, # show results
        save_txt=False, # save results to *.txt
        save_conf=False, # save confidences in --save-txt labels
        save_crop=False, # save cropped prediction boxes
        nosave=False, # do not save images/videos
        classes=None, # filter by class: --class 0, or --class 0 2
3
        agnostic_nms=False, # class-agnostic NMS
        augment=False, # augmented inference
        visualize=False, # visualize features
        update=False, # update all models
        project=ROOT / 'runs/detect', # save results to project/name
        name='exp', # save results to project/name
        exist_ok=False, # existing project/name ok, do not increment

```

```

        line_thickness=3, # bounding box thickness (pixels)
        hide_labels=False, # hide labels
        hide_conf=False, # hide confidences
        half=False, # use FP16 half-precision inference
        dnn=False, # use OpenCV DNN for ONNX inference
    ):
        source = str(source)

        save_img = not nosave and not source.endswith('.txt') # save
inference images

        is_file = Path(source).suffix[1:] in (IMG_FORMATS + VID_FORMATS)
        is_url = source.lower().startswith(('rtsp://', 'rtmp://',
'http://', 'https://'))
        webcam = source.isnumeric() or source.endswith('.txt') or
(is_url and not is_file)
        if is_url and is_file:
            source = check_file(source) # download

    # Directories
    save_dir = increment_path(Path(project) / name,
exist_ok=exist_ok) # increment run
    (save_dir / 'labels' if save_txt else
save_dir).mkdir(parents=True, exist_ok=True) # make dir

    # Load model
    device = select_device(device)

    model = DetectMultiBackend(weights, device=device, dnn=dnn,
data=data, fp16=half)

    stride, names, pt = model.stride, model.names, model.pt
    imgsz = check_img_size(imgsz, s=stride) # check image size

    # Dataloader
    if webcam:
        view_img = check_imshow()

        cudnn.benchmark = True # set True to speed up constant image
size inference

        dataset = LoadStreams(source, img_size=imgsz,

```

```

stride=stride, auto=pt)

    bs = len(dataset) # batch_size
else:
    dataset = LoadImages(source, img_size=imgsz, stride=stride,
auto=pt)

    bs = 1 # batch_size
vid_path, vid_writer = [None] * bs, [None] * bs

# Run inference
model.warmup(imgsz=(1 if pt else bs, 3, *imgsz)) # warmup
dt, seen = [0.0, 0.0, 0.0], 0
for path, im, im0s, vid_cap, s in dataset:
    t1 = time_sync()
    im = torch.from_numpy(im).to(device)
    im = im.half() if model.fp16 else im.float() # uint8 to
fp16/32

    im /= 255 # 0 - 255 to 0.0 - 1.0
    if len(im.shape) == 3:
        im = im[None] # expand for batch dim

    t2 = time_sync()
    dt[0] += t2 - t1

    # Inference
    visualize = increment_path(save_dir / Path(path).stem,
mkdir=True) if visualize else False

    pred = model(im, augment=augment, visualize=visualize)
    t3 = time_sync()
    dt[1] += t3 - t2

    # NMS
    pred = non_max_suppression(pred, conf_thres, iou_thres,
classes, agnostic_nms, max_det=max_det)

    dt[2] += time_sync() - t3

```

```

        # Second-stage classifier (optional)

        #      pred      =      utils.general.apply_classifier(pred,
classifier_model, im, im0s)

    # Process predictions

    for i, det in enumerate(pred): # per image
        seen += 1

        if webcam: # batch_size >= 1
            p, im0, frame = path[i], im0s[i].copy(), dataset.count
            s += f'{i}: '
        else:
            p, im0, frame = path, im0s.copy(), getattr(dataset,
'frame', 0)

        p = Path(p) # to Path
        save_path = str(save_dir / p.name) # im.jpg
        txt_path = str(save_dir / 'labels' / p.stem) + ('' if
dataset.mode == 'image' else f'_{frame}') # im.txt
        s += '%gx%g ' % im.shape[2:] # print string
        gn = torch.tensor(im0.shape)[[1, 0, 1, 0]] #
normalization gain whwh
        imc = im0.copy() if save_crop else im0 # for save_crop
        annotator = Annotator(im0, line_width=line_thickness,
example=str(names))

        if len(det):
            # Rescale boxes from img_size to im0 size
            det[:, :4] = scale_coords(im.shape[2:], det[:, :4],
im0.shape).round()

            # Print results
            for c in det[:, -1].unique():
                n = (det[:, -1] == c).sum() # detections per class
                s += f"{n} {names[int(c)]}{'s' * (n > 1)}, " # add
to string

```

```

        # Write results
        for *xyxy, conf, cls in reversed(det):
            if save_txt: # Write to file
                xywh = (xyxy2xywh(torch.tensor(xyxy).view(1,
4)) / gn).view(-1).tolist() # normalized xywh
                line = (cls, *xywh, conf) if save_conf else
(cls, *xywh) # label format
                with open(txt_path + '.txt', 'a') as f:
                    f.write((' %g ' * len(line)).rstrip() % line
+ '\n')

            if save_img or save_crop or view_img: # Add bbox
to image
                c = int(cls) # integer class
                label = None if hide_labels else (names[c] if
hide_conf else f'{names[c]} {conf:.2f}')
                annotator.box_label(xyxy, label,
color=colors(c, True))

                if save_crop:
                    save_one_box(xyxy, imc, file=save_dir /
'crops' / names[c] / f'{p.stem}.jpg', BGR=True)

        # Stream results
        im0 = annotator.result()
        if view_img:
            cv2.imshow(str(p), im0)
            cv2.waitKey(1) # 1 millisecond

        # Save results (image with detections)
        if save_img:
            if dataset.mode == 'image':
                cv2.imwrite(save_path, im0)
            else: # 'video' or 'stream'
                if vid_path[i] != save_path: # new video
                    vid_path[i] = save_path
                if
                    isinstance(vid_writer[i],

```

```

cv2.VideoWriter):
                                vid_writer[i].release()          # release
previous video writer
                                if vid_cap: # video
                                    fps = vid_cap.get(cv2.CAP_PROP_FPS)
                                    w =
                                =
int(vid_cap.get(cv2.CAP_PROP_FRAME_WIDTH))
                                    h =
                                =
int(vid_cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
                                else: # stream
                                    fps, w, h = 30, im0.shape[1], im0.shape[0]
                                    save_path =
                                =
str(Path(save_path).with_suffix('.mp4')) # force *.mp4 suffix on
results videos
                                vid_writer[i] = cv2.VideoWriter(save_path,
cv2.VideoWriter_fourcc(*'mp4v'), fps, (w, h))
                                vid_writer[i].write(im0)

                                # Print time (inference-only)
                                LOGGER.info(f'{s}Done. ({t3 - t2:.3f}s)')

                                # Print results
                                t = tuple(x / seen * 1E3 for x in dt) # speeds per image
                                LOGGER.info(f'Speed: %.1fms pre-process, %.1fms inference,
%.1fms NMS per image at shape {(1, 3, *imgsz)}' % t)
                                if save_txt or save_img:
                                    s = f"\n{len(list(save_dir.glob('labels/*.txt')))} labels
saved to {save_dir / 'labels'}" if save_txt else ''
                                    LOGGER.info(f"Results saved to {colorstr('bold',
save_dir)}{s}")
                                if update:
                                    strip_optimizer(weights) # update model (to fix
SourceChangeWarning)

def parse_opt():
    parser = argparse.ArgumentParser()
    parser.add_argument('--weights', nargs='+', type=str,

```

```

default=ROOT / 'yolov5s.pt', help='model path(s)')

    parser.add_argument('--source', type=str, default=ROOT /
'data/images', help='file/dir/URL/glob, 0 for webcam')

    parser.add_argument('--data', type=str, default=ROOT /
'data/coco128.yaml', help='(optional) dataset.yaml path')

    parser.add_argument('--imgsz', '--img', '--img-size',
nargs='+', type=int, default=[640], help='inference size h,w')

    parser.add_argument('--conf-thres', type=float, default=0.25,
help='confidence threshold')

    parser.add_argument('--iou-thres', type=float, default=0.45,
help='NMS IoU threshold')

    parser.add_argument('--max-det', type=int, default=1000,
help='maximum detections per image')

    parser.add_argument('--device', default='', help='cuda device,
i.e. 0 or 0,1,2,3 or cpu')

    parser.add_argument('--view-img', action='store_true',
help='show results')

    parser.add_argument('--save-txt', action='store_true',
help='save results to *.txt')

    parser.add_argument('--save-conf', action='store_true',
help='save confidences in --save-txt labels')

    parser.add_argument('--save-crop', action='store_true',
help='save cropped prediction boxes')

    parser.add_argument('--nosave', action='store_true', help='do
not save images/videos')

    parser.add_argument('--classes', nargs='+', type=int,
help='filter by class: --classes 0, or --classes 0 2 3')

    parser.add_argument('--agnostic-nms', action='store_true',
help='class-agnostic NMS')

    parser.add_argument('--augment', action='store_true',
help='augmented inference')

    parser.add_argument('--visualize', action='store_true',
help='visualize features')

    parser.add_argument('--update', action='store_true',
help='update all models')

    parser.add_argument('--project', default=ROOT / 'runs/detect',
help='save results to project/name')

    parser.add_argument('--name', default='exp', help='save
results to project/name')

    parser.add_argument('--exist-ok', action='store_true',
help='existing project/name ok, do not increment')

```



```

    parser.add_argument('--line-thickness', default=3, type=int,
help='bounding box thickness (pixels)')

    parser.add_argument('--hide-labels',            default=False,
action='store_true', help='hide labels')

    parser.add_argument('--hide-conf',            default=False,
action='store_true', help='hide confidences')

    parser.add_argument('--half', action='store_true', help='use
FP16 half-precision inference')

    parser.add_argument('--dnn', action='store_true', help='use
OpenCV DNN for ONNX inference')

    opt = parser.parse_args()

    opt.imgsz *= 2 if len(opt.imgsz) == 1 else 1 # expand

    print_args(FILE.stem, opt)

    return opt

def main(opt):

    check_requirements(exclude=('tensorboard', 'thop'))

    run(**vars(opt))

if __name__ == "__main__":
    opt = parse_opt()
    main(opt)

```

API Golang

```

package main

import (
    "context"
    "fmt"
    "log"
    "os"
    "os/exec"
    "time"

```

```

    "github.com/gofiber/fiber/v2"
    "github.com/gofiber/fiber/v2/middleware/cors"
    "go.mongodb.org/mongo-driver/bson"
    "go.mongodb.org/mongo-driver/mongo"
    "go.mongodb.org/mongo-driver/mongo/options"
)

var collection *mongo.Collection

func showData(c *fiber.Ctx) error {
    cursor, err := collection.Find(c.Context(), bson.M{})
    if err != nil {
        fmt.Println(err)
    }
    var imgd []bson.M

    if err := cursor.All(c.Context(), &imgd); err != nil {
        return c.JSON(fiber.Map{"status": 500, "message":
"Server error", "data": nil})
    }
    return c.JSON(fiber.Map{"status": 200, "message":
"Success Get Data", "data": imgd})
}

// SERVICES
func handleImageServices(c *fiber.Ctx) error {
    file, err := c.FormFile("image")
    if err != nil {

```

```

        log.Println("Image Upload Error", err)

        return c.JSON(fiber.Map{"status": 500, "message":
"Server error", "data": nil})

    }

    filename := file.Filename

    image := fmt.Sprintf("%s", filename)

    err = c.SaveFile(file,
fmt.Sprintf("./uploaded-images/%s", image))

    if err != nil {

        log.Println("image save error --> ", err)

        return c.JSON(fiber.Map{"status": 500, "message":
"Server error", "data": nil})

    }

    // Deteksi Gambar Menggunakan YOLOv5

    cmm := "python ../yolov5/detect.py --source
uploaded-images/" + file.Filename + " --weights
../best.pt --img 500 --project ../go/result-images
--name hasildeteksi --exist-ok"

    // python ../yolov5/detect.py --source
uploaded-images/107u.jpg --weights ../best.pt --img 500
--project ../go/result-images --name hasildeteksi
--exist-ok

    cmd := exec.Command("bash", "-c", cmm)

    cmd.Stdout = os.Stdout

    err = cmd.Run()

    if err != nil {

        fmt.Println(err)

    }

    fmt.Println(cmm)

```

```

    imgUrl                                                    :=
fmt.Sprintf("http://localhost:9000/result/%s", image)

    dataToStore := bson.D{
        {"imgUrlData", imgUrl},
        {"timestamp", time.Now().Format(time.RFC3339)},
    }

    ins, errsa :=
collection.InsertOne(context.Background(),
dataToStore)

    if errsa != nil {
        fmt.Println(errsa)
    }

    fmt.Println("Success Insert Data", ins.InsertedID)

// Create meta data for response
data := map[string]interface{}{
    "imageUrl": imgUrl,
    "header":   file.Header,
    "size":     file.Size,
}

// Return JSON
return c.JSON(fiber.Map{"status": 200, "message":
"Image Upload Success", "data": data})
}

var port = ":9000"

func main() {

```

```

// MongoDB

clientOptions                                     :=
options.Client().ApplyURI("mongodb://localhost:27017")

client, err := mongo.Connect(context.TODO(),
clientOptions)

if err != nil {
    fmt.Println("Mongo Connect ERROR : ", err)
    os.Exit(1)
}

collection                                     =
client.Database("paddy").Collection("imageData")

// Main Function
fmt.Println("Berjalan di localhost", port)

// FIBER
app := fiber.New()

// Communication using HTTP and Allow Cors
app.Use(cors.New())

// Akses Folder
app.Static("/result", "result-images/hasildeteksi")

// Get List Deteksi
app.Get("/imageDetectionData", showData)

```

```
// Upload Image Services Route
app.Post("/image-services", handleImageServices)

log.Fatal(app.Listen(port))

}
```

LAMPIRAN 2. BIODATA PENULIS

BIODATA PENULIS TUGAS AKHIR

Nama :
Nomor Induk Mahasiswa :
Jenis Kelamin :
Tempat & Tanggal Lahir :
Status :
Agama :
Asal Ijazah : Nama Sekolah Kota Sekolah Tahun Ijazah

SD :
SLTP :
SLTA :
Program Studi / Jurusan : D3 Teknik Informatika / Teknik Informatika
Alamat Rumah Asal :
No. Telpn / HP :
Email :
Nama Orang Tua :
Pekerjaan Orang Tua :
Judul Tugas Akhir :
Dosen Pembimbing : 1. A. Sumarudin, S.Pd., M.T., M.Sc.
2. Adi Suheryadi, S.ST., M.Kom.

Pas foto 3 x 4

Berwarna

Terbaru

Indramayu, 18 Agustus 2021

Penulis,

RENOL NINDI KARA N

NIM 1903027