

**ANALISIS KINERJA *YOU ONLY LOOK ONCE* (YOLO) UNTUK  
DETEKSI OBJEK PADA PENYAKIT TANAMAN KENTANG**

**SKRIPSI**



**Disusun oleh:**

**DEVI LUTHFITANINGRUM  
19/440305/TK/48632**

**PROGRAM STUDI TEKNOLOGI INFORMASI  
DEPARTEMEN TEKNIK ELEKTRO DAN TEKNOLOGI INFORMASI  
FAKULTAS TEKNIK UNIVERSITAS GADJAH MADA  
YOGYAKARTA  
2023**

## HALAMAN PENGESAHAN

### ANALISIS KINERJA *YOU ONLY LOOK ONCE (YOLO)* UNTUK DETEKSI OBJEK PADA PENYAKIT TANAMAN KENTANG

### SKRIPSI

Diajukan sebagai Salah Satu Syarat untuk Memperoleh

Gelar Sarjana Teknik

pada Departemen Teknik Elektro dan Teknologi Informasi Fakultas Teknik

Universitas Gadjah Mada



Dosen Pembimbing I

Dr. Indah Soesanti, S.T., M.T.  
NIP. 197406152005012001

Dosen Pembimbing II

Dzuhri Radityo Utomo, S.T., M.E., Ph.D  
NIP. 111199010202001101



**SKRIPSI**

**ANALISIS KINERJA YOU ONLY LOOK ONCE (YOLO) UNTUK DETEKSI OBJEK  
PADA PENYAKIT TANAMAN KENTANG**

Dipersiapkan dan disusun oleh

**Devi Luthfitaningrum**  
19/440305/TK/48632

Telah dipertahankan di depan dewan penguji  
pada tanggal : **26 Juni 2023**

**Susunan Dewan Penguji**

Pembimbing Utama

**Dr. Indah Soesanti, S.T., M.T.**

Pembimbing Pendamping

**Dzuhri Radityo Utomo, S.T, M.E., Ph.D**

Anggota Dewan Penguji Lain

**Syukron Abu Ishaq Alfarizi, S.T., Ph.D.**

**Warsun Najib, S.T., M.Sc.**

Skripsi ini telah diterima sebagai salah satu persyaratan  
untuk memperoleh gelar Sarjana Teknik

Tanggal: 17 Juli 2023

Pengelola Program Studi: Sarjana Teknologi Informasi

**Ir. Agus Bejo, S.T., M.Eng., D.Eng., IPM.**

NIP 198001012015041002

Mengetahui,

Ketua Departemen Teknik Elektro dan Teknologi Informasi



**Prof. Ir. Hanung Adi Nugroho, S.T., M.Eng., Ph.D., IPM., SMIEEE.**

NIP 197802242002121001



## **PERNYATAAN BEBAS PLAGIASI**

Saya yang bertanda tangan di bawah ini :

Nama : Devi Luthfitaningrum  
NIM : 19/440305/TK/48632  
Tahun terdaftar : 2019  
Program Studi : Teknologi Informasi  
Fakultas : Teknik Universitas Gadjah Mada

Menyatakan bahwa dalam dokumen ilmiah Skripsi ini tidak terdapat bagian dari karya ilmiah lain yang telah diajukan untuk memperoleh gelar akademik di suatu lembaga Pendidikan Tinggi, dan juga tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang/lembaga lain, kecuali yang secara tertulis disitasi dalam dokumen ini dan disebutkan sumbernya secara lengkap dalam daftar pustaka.

Dengan demikian saya menyatakan bahwa dokumen ilmiah ini bebas dari unsur-unsur plagiasi dan apabila dokumen ilmiah Skripsi ini di kemudian hari terbukti merupakan plagiasi dari hasil karya penulis lain dan/atau dengan sengaja mengajukan karya atau pendapat yang merupakan hasil karya penulis lain, maka penulis bersedia menerima sanksi akademik dan/atau sanksi hukum yang berlaku.

Yogyakarta, 15 Juni 2023



Devi Luthfitaningrum  
NIM 19/440305/TK/48632

## **HALAMAN PERSEMPAHAN**

Tugas akhir ini kupersembahkan kepada kedua orang tuaku. Kupersembahkan pula kepada keluarga dan teman-teman semua, serta untuk bangsa, negara, dan agamaku.

## KATA PENGANTAR

Puji syukur ke hadirat Allah SWT atas limpahan rahmat, karunia, serta petunjuk-Nya sehingga tugas akhir berupa penyusunan skripsi berjudul "Analisis Kinerja *You Only Look Once* (YOLO) untuk Deteksi Objek pada Penyakit Tanaman Kentang" telah terselesaikan dengan baik. Sholawat serta salam juga disampaikan kepada junjungan dan teladan seluruh umat muslim, Nabi Muhammad SAW. Dalam hal penyusunan tugas akhir ini penulis telah banyak mendapatkan arahan, bantuan, serta dukungan dari berbagai pihak. Oleh karena itu pada kesempatan ini penulis mengucapkan terima kasih kepada:

1. Ir. Hanung Adi Nugroho, S.T., M.E., Ph.D., IPM. selaku Ketua Departemen Teknik Elektro dan Teknologi Informasi.
2. Ir. Lesnanto Multa Putranto, S.T., M.Eng., Ph.D., IPM. selaku Sekretaris Departemen Teknik Elektro dan Teknologi Informasi.
3. Dr. Indah Soesanti, S.T., M.T. selaku Dosen Pembimbing I dan Dzuhri Radityo Utomo, S.T., M.E., Ph. D. Selaku Dosen Pembimbing II yang telah memberikan ilmu dan bimbingan terbaik kepada penulis dengan penuh kesabaran dan ketulusan.
4. Kedua Orang Tua yaitu Bapak alm. Waluyo dan Ibu Titik Komsati, serta kakak bersama keluarga kecilnya yang senantiasa mendoakan, mendidik, dan mendukung penulis.
5. Seluruh teman saya, terutama teman-teman DTETI angkatan 2019 yang telah menjadi *support-system* saya.
6. Seluruh Dosen dan Tenaga Kependidikan Departemen Teknik Elektro dan Teknologi Informasi, Fakultas Teknik Universitas Gadjah Mada Yogyakarta yang telah membantu penulis dalam masa studi.
7. Pihak lain yang berjasa dalam hidup saya, tetapi tidak dapat saya sebut satu per satu.

Akhir kata penulis berharap semoga skripsi ini dapat memberikan manfaat bagi kita semua, Aamiin.

## DAFTAR ISI

HALAMAN PENGESAHAN .....	ii
PERNYATAAN BEBAS PLAGIASI .....	iii
HALAMAN PERSEMBAHAN .....	iv
KATA PENGANTAR .....	v
DAFTAR ISI.....	vi
DAFTAR TABEL .....	ix
DAFTAR GAMBAR .....	x
DAFTAR SINGKATAN.....	xiii
INTISARI.....	xiii
ABSTRACT .....	xiv
BAB I Pendahuluan .....	1
1.1 Latar Belakang .....	1
1.2 Rumusan Masalah .....	4
1.3 Tujuan Penelitian .....	4
1.4 Batasan Penelitian .....	4
1.5 Manfaat Penelitian .....	4
1.6 Sistematika Penulisan.....	5
BAB II Tinjauan Pustaka dan Dasar Teori .....	6
2.1 Tinjauan Pustaka .....	6
2.2 Dasar Teori .....	9
2.2.1 Penyakit Tanaman Kentang.....	9
2.2.1.1 Penyakit Tanaman Kentang <i>Early Blight</i> .....	10
2.2.1.2 Penyakit Tanaman Kentang <i>Late Blight</i> .....	10
2.2.2 Deteksi Objek .....	11
2.2.3 Akurasi Deteksi .....	13
2.2.3.1 <i>Confusion Matrix</i> .....	14
2.2.3.2 <i>Performance Evaluation Metrics</i> .....	15
2.2.4 Kompleksitas Model .....	17
2.2.5 Waktu Komputasi .....	17
2.2.6 <i>Deep Learning</i> .....	17
2.2.7 <i>Convolutional Neural Network (CNN)</i> .....	18
2.2.7.1 <i>Convolutional Layer</i> .....	19
2.2.7.2 <i>Pooling Layer</i> .....	19
2.2.7.3 <i>Fully Connected Layer</i> .....	19
2.2.7.4 <i>Activation Function</i> (Fungsi Aktivasi).....	20
2.2.7.5 <i>Loss Function</i> (Fungsi Kesalahan).....	21

2.2.8	<i>You Only Look Once</i> (YOLO) .....	21
2.2.8.1	YOLOv5 .....	25
2.2.8.2	YOLOv6 .....	27
2.2.8.3	YOLOv7 .....	29
2.2.8.4	YOLOv8 .....	31
2.3	Analisis Perbandingan Metode .....	32
2.4	Pertanyaan Tugas Akhir .....	34
BAB III	Metode Penelitian.....	35
3.1	Alat dan Bahan Tugas akhir .....	35
3.1.1	Alat Tugas akhir.....	35
3.1.2	Bahan Tugas akhir .....	35
3.2	Metode yang Digunakan.....	36
3.3	Alur Tugas Akhir .....	36
3.3.1	Perumusan Fokus Tugas Akhir .....	37
3.3.2	Perancangan Tugas Akhir.....	38
3.3.2.1	Pemilihan Dataset.....	38
3.3.2.2	Pemilihan Model.....	39
3.3.2.3	Perancangan Evaluasi Model.....	39
3.3.3	Implementasi Tugas Akhir.....	39
3.3.3.1	Anotasi Data .....	39
3.3.3.2	Pelatihan dan Validasi Model .....	40
3.3.3.3	Pengujian Model.....	42
3.3.3.4	Evaluasi Model .....	42
3.3.4	Penulisan Naskah Ilmiah .....	43
BAB IV	Hasil dan Pembahasan.....	44
4.1	Pengujian Model .....	44
4.1.1	Akurasi Deteksi .....	44
4.1.2	Kompleksitas Model .....	45
4.1.3	Waktu Komputasi .....	46
4.2	Evaluasi Model.....	48
4.3	Perbandingan Hasil Penelitian dengan Hasil Terdahulu .....	53
BAB V	Kesimpulan dan Saran.....	55
5.1	Kesimpulan.....	55
5.2	Saran.....	55
DAFTAR PUSTAKA.....		56
LAMPIRAN .....		L-1
L.1	Source Code dan Model .....	L-1
L.1.1	YOLOv5m.....	L-1
L.1.2	YOLOv6m.....	L-2

L.1.3	YOLOv7 .....	L-3
L.1.4	YOLOv8m .....	L-4
L.1.5	Hyperparameter .....	L-5
L.2	Dataset dan Sampel Hasil Deteksi .....	L-5

## **DAFTAR TABEL**

Tabel 2.1	Penelitian menggunakan metode CNN pada tanaman .....	8
Tabel 2.2	Perbandingan Model YOLOv5m, YOLOv6m, YOLOv7, dan YOLOv8m .....	33
Tabel 3.1	Pembagian Dataset .....	38
Tabel 3.2	Konfigurasi Hyperparameter Pelatihan Model .....	41
Tabel 4.1	Perbandingan AP@0.5 dan mAP@0.5 setiap model .....	44
Tabel 4.2	Waktu Deteksi Model .....	47
Tabel 4.3	Hasil Pengujian Model .....	48
Tabel 4.4	Perbandingan Hasil Penelitian .....	53
Tabel 4.5	Kelebihan dan Kekurangan Tugas Akhir .....	54

## DAFTAR GAMBAR

Gambar 2.1	Penyakit <i>Early Blight</i> [1] .....	10
Gambar 2.2	Penyakit <i>Late Blight</i> [2] .....	11
Gambar 2.3	Contoh Perbedaan Klasifikasi Objek dan Deteksi Objek [3] .....	12
Gambar 2.4	Perbedaan <i>two-stage detector</i> dan <i>one-stage detector</i> [4] .....	13
Gambar 2.5	Perbedaan <i>underfitting</i> , <i>balanced</i> , dan <i>overfitting</i> [5] .....	13
Gambar 2.6	<i>Confusion Matrix</i> [6] .....	14
Gambar 2.7	<i>Intersection over Union</i> [7] .....	14
Gambar 2.8	Perbedaan <i>Machine Learning</i> dengan <i>Deep Learning</i> [8] .....	18
Gambar 2.9	Ilustrasi Proses di <i>Convolutional Layer</i> .....	19
Gambar 2.10	Ilustrasi Proses di <i>Pooling Layer</i> .....	20
Gambar 2.11	Ilustrasi Proses di <i>Fully Connected Layer</i> .....	20
Gambar 2.12	Prinsip Kerja Algoritma YOLO [9] .....	22
Gambar 2.13	Visualisasi Perbedaan <i>Anchor-based</i> dan <i>Anchor-free</i> [10] .....	23
Gambar 2.14	Arsitektur Jaringan Model YOLO modern [11] .....	24
Gambar 2.15	Lini Masa YOLO [12] .....	25
Gambar 2.16	Arsitektur Jaringan Model YOLOv5 [13] .....	26
Gambar 2.17	Model-model YOLOv5 [14] .....	27
Gambar 2.18	Arsitektur Jaringan Model YOLOv6 [15] .....	28
Gambar 2.19	<i>Distribution Focal Loss</i> [16] .....	28
Gambar 2.20	Model-model YOLOv6 [17] .....	29
Gambar 2.21	Arsitektur Jaringan Model YOLOv7 [18] .....	30
Gambar 2.22	Model-model YOLOv7 [19] .....	30
Gambar 2.23	Arsitektur Jaringan Model YOLOv8 [20] .....	31
Gambar 2.24	Model-model YOLOv8 [21] .....	32
Gambar 3.1	Contoh Dataset <i>Healthy</i> , <i>Early Blight</i> , <i>Late Blight</i> .....	35
Gambar 3.2	Alur Tugas Akhir .....	37
Gambar 3.3	Proses Anotasi Data .....	40
Gambar 3.4	Augmentasi Mosaic [21] .....	42
Gambar 3.5	Augmentasi HSV [21] .....	42
Gambar 4.1	Kurva <i>Precision-Recall</i> .....	44
Gambar 4.2	Hasil Akurasi Deteksi Model .....	45
Gambar 4.3	Perbandingan Jumlah Parameter Model .....	46
Gambar 4.4	Waktu Pelatihan Model .....	46
Gambar 4.5	Waktu Deteksi Model .....	47
Gambar 4.6	Sampel Hasil Deteksi Citra pada Data Pengujian .....	48
Gambar 4.7	Sampel Hasil Deteksi Citra <i>Cluttered-background</i> .....	49
Gambar 4.8	Sampel Hasil Deteksi Citra <i>Mosaic</i> .....	49
Gambar 4.9	Sampel Hasil Deteksi Frame dari Video [22] .....	50
Gambar 4.10	Perbandingan Jumlah Parameter dengan Waktu Pelatihan Model .....	51
Gambar 4.11	Perbandingan mAP@0.5:0.95 terhadap Waktu Deteksi .....	52



## DAFTAR SINGKATAN

ha	= hekto are
ton/ha	= ton per hekto are
ResNet	= Residual Network
VGG	= Visual Geometry Group
KNN	= K-Nearest Neighbor
SVM	= Support Vector Machines
CNN	= Convolutional Neural Network
ANN	= Artificial Neural Network
DNN	= Deep Neural Network
RNN	= Recurrent Neural Network
YOLO	= You Only Look Once
SSD	= Single Shot Multibox Detector
R-CNN	= Region-based Convolutional Neural Networks
IoU	= Intersection over Union
TP	= True Positive
FP	= False Positive
FN	= False Negative
TN	= True Negative
AUC	= Area Under Curve
AP	= Average Precision
mAP	= mean Average Precision
NMS	= Non Maximum Suppression
FPS	= Frame Per Second
ms	= millisecond
ReLU	= Rectified Linear Unit
SiLU	= Sigmoid Linear Unit
CPU	= Central Processing Unit
GPU	= Graphics Processing Unit
CSPNet	= Cross Stage Partial Network
PANet	= Path Aggregation Network
SPP	= Spatial Pyramid Pooling
CmBN	= Cross mini-Batch Normalization
CIoU	= Complete Intersection over Union
GIoU	= Generalized Intersection over Union
SIoU	= Series Intersection over Union
TAL	= Task Aligned Learning
VFL	= Varifocal Loss
DFL	= Distribution Focal Loss
E-ELAN	= Extended-Efficient Layer Aggregation Networks

## INTISARI

Kentang adalah salah satu tanaman umbi yang berpotensi menjadi makanan pokok, serta memiliki tinggi manfaat bagi tubuh. Banyaknya peran kentang menyebabkan peningkatan permintaan dan nilai ekonomi kentang. Namun produktivitas kentang di Indonesia pada tahun 2022 masih sangat rendah yaitu 19,2 ton/ha. Selain itu, impor kentang Indonesia pada Januari 2023 sangat tinggi yaitu sebesar 7160 ton. Berdasarkan rendahnya produktivitas dan tingginya impor kentang di Indonesia, dapat diketahui bahwa produksi kentang di Indonesia belum memenuhi permintaan. Faktor utama yang menghambat produksi kentang adalah penyakit tanaman. Penyakit yang umumnya menyerang dan memiliki dampak besar pada tanaman kentang adalah *early blight* (bercak kering) dan *late blight* (hawar daun). Penyakit tersebut ditandai dengan pola pada daun yang dapat diklasifikasikan oleh model *machine learning* maupun *deep learning*. Namun pada penelitian sebelumnya, model belum dapat mendekripsi lokasi daun kentang yang terinfeksi. Pada penelitian ini, model *deep learning* diterapkan untuk mendekripsi kelas dan lokasi penyakit tanaman kentang pada daun. Model *deep learning* yang digunakan adalah model *Convolutional Neural Network* (CNN) dari keluarga *You Only Look Once* (YOLO) yaitu YOLOv5m, YOLOv6m, YOLOv7, dan YOLOv8m. Model dilatih dengan 2100 citra dan divalidasi dengan 600 citra, kemudian diuji dengan 300 citra. Pengujian dilakukan untuk mengetahui kinerja model berdasarkan akurasi deteksi, kompleksitas model, dan waktu komputasi. Hasil pengujian kemudian dievaluasi untuk menemukan model dengan kinerja terbaik. Model dengan kinerja terbaik untuk mendekripsi penyakit tanaman kentang adalah YOLOv6m dengan mAP@0.5 bernilai 0,995, mAP@0.5:0.95 bernilai 0,979, parameter berjumlah 34,8 juta, waktu pelatihan 1,55 jam, dan waktu deteksi 22,38 ms. YOLOv6m dipilih sebagai model terbaik karena YOLOv6m menghasilkan akurasi deteksi yang sebanding dengan kompleksitas model dan waktu komputasinya. Deteksi YOLOv6m mencapai hasil akurasi yang tinggi, baik pada citra terstruktur maupun citra tidak terstruktur.

Kata kunci : *Deep Learning*, *Convolutional Neural Network* (CNN), *You Only Look Once* (YOLO), *Early Blight*, *Late Blight*

## ABSTRACT

*Potato is one of the tuber crops that has the potential to become a staple food and has high benefits for the body. The many roles of potato leads an increase in demand and economic value of potato. However, productivity of potato in Indonesia still very low at 19,2 ton/ha in 2022. In addition, Indonesia's import of potato is very high in January 2023 that is 7160 tonnes. Based on low productivity and high import of potato in Indonesia, it can be seen that potato production has not fulfilled the demand. The main factor that inhibit potato production is plant disease. Disease that commonly attack and have a big impact on potato plant is early blight and late blight. The diseases characterized by pattern on potato leaves which can be classified by machine learning and deep learning models. However in several previous researchs, the models have not been able to detect the location of infected potato leaves. In this research, the deep learning models applied to detect the class and the location of potato diseases on leaves. The deep learning models used are Convolutional Neural Network (CNN) models from the You Only Look Once (YOLO) family, especially YOLOv5m, YOLOv6m, YOLOv7, and YOLOv8m. The models were trained with 2100 images and validated with 600 images, then tested with 300 images. Testing was conducted to find out model performance based on detection accuracy, model complexity, and computation time. The test results are evaluated to find the model with the best performance. The best performance model to detect potato diseases is YOLOv6m with mAP@0.5 is 0,995, mAP@0.5:0.95 is 0,979, there are 34,8 million parameters, the training time is 1,55 hours, and the detection time is 22,38 ms. YOLOv6m was chosen as the best model because YOLOv6m produces detection accuracy that is comparable with model complexity and computation time. YOLOv6m detection achieves high accuracy results, both on structured image and unstructured image.*

**Keywords :** Deep Learning, Convolutional Neural Network (CNN), You Only Look Once (YOLO), Early Blight, Late Blight

# BAB I

## PENDAHULUAN

### 1.1 Latar Belakang

Kentang adalah salah satu tanaman umbi yang memiliki kadar karbohidrat sebesar 19% [23]. Dengan kandungan karbohidrat yang cukup tinggi, kentang berpotensi menjadi sumber karbohidrat atau makanan pokok bagi masyarakat dunia. Selain dapat dijadikan makanan pokok, kentang juga memiliki manfaat bagi kesehatan tubuh seperti mengurangi risiko penyakit jantung dan kanker, mencegah diabetes, serta menjaga kesehatan pencernaan [24].

Banyaknya peran kentang tentunya menyebabkan peningkatan permintaan dan nilai ekonomi kentang. Dengan begitu produktivitas kentang di dunia, khususnya di Indonesia perlu dikaji dan ditingkatkan lagi. Produksi kentang di Indonesia pada tahun 2022 meningkat dibanding produksi tahun sebelumnya. Hal ini berdasarkan data yang dipublikasikan oleh Badan Pusat Statistik yaitu berjumlah 1,42 juta ton pada tahun 2022 dan berjumlah 1,36 juta ton pada tahun 2021 [25]. Namun hal tersebut belum meningkatkan produktivitas kentang di Indonesia secara signifikan. Berdasarkan data Badan Pusat Statistik, lahan panen tanaman kentang tahun 2021 adalah 71786 ha, sementara lahan panen tahun 2022 seluas 73714 ha [26]. Maka produktivitas kentang di Indonesia tahun 2021 adalah 19,0 ton/ha, sedangkan produktivitas kentang tahun 2022 adalah 19,2 ton/ha. Hal ini menunjukkan bahwa produktivitas kentang di Indonesia masih sangat rendah dibanding negara-negara Eropa yang berkisar antara 40-60 ton/ha [27]. Angka impor kentang Indonesia pada Januari 2023 juga tinggi, yaitu sebanyak 7160 ton [28]. Data-data tersebut menunjukkan bahwa kentang merupakan tanaman yang penting bagi penyediaan pangan dan pendukung perekonomian di Indonesia.

Dengan rendahnya produktivitas dan tingginya angka impor kentang di Indonesia, maka dapat diketahui bahwa produksi kentang di Indonesia belum memenuhi permintaan. Padahal menurut hasil penelitian Dinas Pertanian dan Ketahanan Pangan Provinsi Jawa Timur [29], Indonesia memiliki potensi produktivitas kentang hingga 30 ton/ha. Produktivitas kentang yang rendah disebabkan oleh beberapa faktor, diantaranya bibit yang kurang bermutu sehingga rentan terhadap patogen. Salah satu patogen yang umumnya menyerang tanaman kentang adalah jamur. Penyakit pada kentang yang disebabkan oleh jamur adalah penyakit *early blight* (bercak kering) dan penyakit *late blight* (hawar daun). Kedua penyakit tersebut dapat menyebabkan kematian tanaman kentang bahkan memunculkan risiko gagal panen.

Dalam penanganan penyakit tanaman kentang, diperlukan proses identifikasi penyakit yang menginfeksi tanaman. Hal ini menentukan langkah yang harus dilakukan

pada tanaman terinfeksi atau langkah untuk mencegah penyebaran penyakit tersebut. Proses identifikasi penyakit sebaiknya dilakukan dengan akurasi yang tinggi dan waktu yang cepat. Padahal tidak semua petani memiliki pengetahuan yang cukup tentang identifikasi penyakit tanaman kentang, maka identifikasi membutuhkan waktu yang lebih lama untuk menemukan seorang ahli agar hasil identifikasi akurat. Untuk mempermudah proses identifikasi penyakit kentang, beberapa penelitian di bidang visi komputer telah diterapkan. Penyakit pada tanaman dapat ditandai dengan adanya perubahan warna atau munculnya pola tertentu pada daun [30]. Perubahan warna atau munculnya pola tertentu dapat dikenali sehingga penyakit pada tanaman dapat diidentifikasi. Identifikasi penyakit tanaman kentang menggunakan pola daun berbasis *machine learning* telah diterapkan. Rakhmawati dkk, 2018 [31] menerapkan klasifikasi menggunakan model SVM berdasarkan fitur tekstur dan fitur warna daun. Penelitian ini menggunakan 300 citra data latih dan memperoleh akurasi 83,33% pada 90 data uji. Pada tahun 2020, Huda dkk [32] menerapkan model SVM dan KNN pada 900 citra latih berdasarkan fitur warna, tekstur, dan bentuk. Pengujian tersebut menghasilkan akurasi paling tinggi 88% pada 300 citra data uji yaitu ketika menggunakan algoritma SVM dengan fitur warna. Pada tahun yang sama, Kahfi, A. H. [33] melakukan klasifikasi penyakit tanaman kentang dengan KNN berdasarkan fitur warna dan tekstur daun. Penelitian ini melatih model dengan 1080 data pelatihan dan menghasilkan rata-rata akurasi 98,3% pada 120 citra data pengujian. Selanjutnya Amatullah dkk, 2021 [34] mencapai akurasi 80% dengan fitur dan model yang sama pada 90 citra data uji. Model dilatih dengan 90 data latih dan 90 data validasi. Penelitian-penelitian tersebut menunjukkan bahwa peningkatan jumlah citra yang digunakan dalam pelatihan dapat meningkatkan akurasi model. Hal ini disebabkan perubahan pola yang muncul akibat penyakit tidak beraturan dan sangat variatif. Maka diperlukan volume data yang lebih banyak agar model dapat mengenali pola-pola baru dari penyakit.

Volume data yang bertambah menyebabkan para peneliti beralih ke metode *deep learning*. *Deep learning* adalah bagian dari *machine learning* yang memiliki jaringan saraf tiruan untuk berlatih memahami data berskala besar. Salah satu metode *deep learning* yang banyak diterapkan pada identifikasi citra yaitu metode *Convolutional Neural Network* (CNN). Pada tahun 2020, Mohamed, S. I. [35] membandingkan akurasi klasifikasi penyakit tanaman kentang menggunakan beberapa metode *machine learning* dengan metode CNN. Pelatihan dilakukan dengan dataset PlantVillage berjumlah 1700 citra yang terdiri dari daun *Healthy*, *Early Blight*, dan *Late Blight*. Hasil akurasi yang diperoleh menunjukkan bahwa metode CNN memiliki akurasi klasifikasi yang lebih tinggi dibanding *machine learning* yaitu 98% pada 300 data uji. CNN merupakan metode pengenalan objek yang memiliki kinerja cukup tinggi karena meniru sistem pengenalan pada *visual cortex* manusia [36]. Keberhasilan CNN dalam identifikasi penyakit tanaman kentang mendorong penelitian lebih lanjut dengan berbagai modifikasi model CNN. Ningsih, 2021 [37] melakukan implementasi CNN model GoogleNet pada 1700 citra

latih, pengujian model menghasilkan akurasi 89,38% pada 452 citra uji. Rashid dkk, 2021 [38] melakukan implementasi CNN dengan model YOLOv5 pada 1725 citra latih, 214 citra validasi, dan 213 citra uji. Pengujian pada model klasifikasi ini memperoleh akurasi paling tinggi yaitu 99,75% dengan kompleksitas model yang rendah dibanding model VGG16, InceptionResNetV2, DenseNet169, dan Xception. Hal ini ditunjukkan dari parameter pada model yang hanya berjumlah sekitar 8,5 juta. Sementara model lain yang dibandingkan memiliki parameter yang berjumlah antara 12,6 hingga 20,8 juta. Lesmana dkk [39] menerapkan CNN pada 5400 citra daun kentang dengan berbagai skenario pembagian citra latih dan citra validasi. Akurasi tertinggi yang diperoleh pada citra latih 93% dan citra validasi 99%. Kemudian Kothari dkk [40] membandingkan CNN, GoogleNet, ResNet50, dan VGG16 pada 900 citra latih dan 300 citra validasi. Dari hasil analisis diperoleh akurasi tertinggi 98% menggunakan model ResNet50.

Berdasarkan penelitian terdahulu yang dipaparkan, model CNN dapat mengklasifikasikan penyakit tanaman dengan akurasi tinggi terutama implementasi CNN dengan model YOLOv5. Namun model belum dapat mendeteksi letak tanaman kentang yang terinfeksi penyakit, hal ini kurang efisien diterapkan pada lahan kentang berskala besar. Pada lahan kentang berskala besar diperlukan sistem monitoring dan automasi kendali penyakit tanaman berbasis *real-time* dan *internet of things* agar penanganan penyakit lebih cepat. Padahal sistem monitoring dan automasi kendali penyakit tanaman berbasis *real-time* dan *internet of things* sangat memerlukan estimasi lokasi infeksi. Oleh karena itu, dalam penelitian ini akan diimplementasikan model deteksi objek. Deteksi objek merupakan lingkup visi komputer yang meliputi klasifikasi dan lokalisasi dari suatu objek [41]. Deteksi objek sudah banyak diteliti menggunakan berbagai model. Deteksi objek dapat dilakukan dengan 2 metode, yaitu metode tradisional dan *deep learning* [42]. Deteksi objek berbasis *deep learning* memiliki kinerja lebih baik dibanding metode tradisional [43]. Salah satu model deteksi objek berbasis *deep learning* adalah *You Only Look Once* (YOLO). YOLOv1 memiliki waktu komputasi yang cepat dan efisien, tetapi memiliki akurasi yang masih rendah karena berbasis *one-stage detector*. Namun model YOLO selalu dikembangkan hingga terdapat lebih dari 30 detektor model YOLO mulai dari YOLOv1 hingga YOLOv8. Pada penelitian terdahulu Rashid dkk [38] menunjukkan bahwa YOLOv5 memperoleh akurasi tertinggi sebagai model klasifikasi objek dengan kompleksitas model yang rendah. Hal ini menunjukkan bahwa setiap model YOLO mempunyai kinerja yang berbeda. Hal tersebut perlu dianalisis agar diperoleh model terbaik untuk diterapkan dalam deteksi objek pada citra. Analisis perlu dilakukan terhadap kelebihan dan kekurangan setiap model. Namun belum banyak penelitian yang membandingkan kinerja berbagai model deteksi objek YOLO. Dengan demikian, penelitian ini bertujuan untuk menganalisis kinerja model-model YOLO dalam deteksi objek pada penyakit tanaman kentang.

## **1.2 Rumusan Masalah**

Berdasarkan latar belakang yang telah diuraikan sebelumnya, maka permasalahan yang dihadapi dalam penelitian ini adalah sebagai berikut:

1. Belum diketahui kinerja model YOLO dalam mendekripsi penyakit tanaman kentang *early blight* (bercak kering) dan *late blight* (hawar daun).
2. Belum diketahui model YOLO dengan kinerja terbaik dalam mendekripsi penyakit tanaman kentang *early blight* (bercak kering) dan *late blight* (hawar daun).

## **1.3 Tujuan Penelitian**

Berdasarkan permasalahan yang harus dihadapi, penelitian ini bertujuan untuk:

1. Menguji model untuk mengetahui kinerja model YOLO dalam mendekripsi penyakit tanaman kentang *early blight* (bercak kering) dan *late blight* (hawar daun).
2. Mengevaluasi model untuk menemukan model YOLO dengan kinerja terbaik dalam mendekripsi penyakit tanaman kentang *early blight* (bercak kering) dan *late blight* (hawar daun).

## **1.4 Batasan Penelitian**

Batasan penelitian ini adalah:

1. Pengujian model hanya menggunakan sampel berjumlah 3000 citra daun kentang. Citra tersebut diambil dari PlantVillage Dataset, Potato Leaf Dataset, dan Potato Disease Leaf Dataset yang terdiri dari 1000 citra tiap kelas objek.
2. Dataset yang digunakan berupa citra dengan rasio ukuran teratur dan fokus pada daun kentang sehingga citra terstruktur dan tidak terdapat *obstacle*.
3. Objek yang dideteksi model adalah tanaman kentang *healthy* (sehat), *early blight* (bercak kering), dan *late blight* (hawar daun).
4. Penelitian ini berfokus untuk menemukan model YOLO yang memiliki kinerja terbaik untuk pendekripsi objek penyakit tanaman kentang, sehingga tidak membahas mengenai implementasi pada perangkat keras tertentu.
5. Model YOLO yang dianalisis kinerjanya hanya model YOLOv5, YOLOv6, YOLOv7, dan YOLOv8.

## **1.5 Manfaat Penelitian**

1. Menjadi referensi penelitian lebih lanjut terkait analisis kinerja model deteksi objek, khususnya untuk mendekripsi penyakit tanaman kentang.

- Mempermudah pengambilan keputusan dalam menentukan model deteksi objek yang dapat diimplementasikan pada sistem monitoring dan automasi kendali penyakit tanaman berbasis *real-time* dan *internet of things*.

## 1.6 Sistematika Penulisan

Sistematika penulisan berdasarkan bahasan setiap Bab pada penulisan penelitian ini adalah sebagai berikut:

Bab I membahas pendahuluan penelitian yang memaparkan latar belakang masalah, rumusan masalah yang dihadapi, tujuan yang akan dicapai dengan penelitian ini, batasan penelitian, manfaat penelitian, dan sistematika penulisan.

Bab II membahas mengenai tinjauan pustaka yang berisi penelitian-penelitian terdahulu sehingga ditemukan hal yang dapat dikembangkan melalui penelitian ini. Pada bab ini juga dijelaskan teori-teori relevan yang mendasari penelitian ini. Tinjauan pustaka dan dasar teori diperoleh dari berbagai referensi yang kredibel.

Bab III menguraikan alat dan bahan yang digunakan dalam penelitian. Pada bab ini juga dijelaskan metode penelitian meliputi langkah-langkah yang dilakukan oleh peneliti.

Bab IV menjelaskan hasil dan pembahasan dari penelitian yang sudah dijalankan. Pada bab ini juga dibahas perbandingan penelitian yang dilakukan oleh penulis dengan penelitian terdahulu.

Bab V menjelaskan apakah hasil penelitian dapat mencapai tujuan dari penelitian yang dilakukan. Pada bab ini juga dipaparkan temuan dan kesimpulan berdasarkan hasil penelitian, serta saran bagi akademisi terkait penelitian lebih lanjut di masa mendatang.

## **BAB II**

### **TINJAUAN PUSTAKA DAN DASAR TEORI**

#### **2.1 Tinjauan Pustaka**

Metode CNN telah diterapkan di beberapa penelitian dalam identifikasi objek tanaman. Dalam penerapannya, metode CNN dikembangkan menjadi berbagai model. Setiap model CNN yang diimplementasikan dalam identifikasi objek memiliki kemampuan dan prinsip kerja yang berbeda. Hal inilah yang mendorong beberapa peneliti untuk menganalisis kinerja berbagai model CNN berdasarkan akurasi, kompleksitas, maupun waktu komputasinya.

KC, Kamal dkk (2019) [44] membandingkan model AlexNet, VGG, MobileNet, Modified MobileNet, dan Reduced MobileNet dalam mengklasifikasi penyakit pada 38 tanaman berdasarkan citra daun. Penelitian menggunakan PlantVillage Dataset yang berjumlah 82161 citra terdiri dari 55 kelas, hasil akurasi klasifikasi tertinggi adalah model VGG yaitu 99,53% dengan 14,7 juta parameter. Kemudian menyusul model AlexNet dengan akurasi 99,44%, MobileNet dengan akurasi model 98,65%, Reduced MobileNet dengan akurasi 98,34%, dan Modified MobileNet dengan akurasi 97,65%. Namun penelitian ini menyimpulkan bahwa MobileNet merupakan model paling berhasil karena dapat mencapai akurasi yang tinggi dengan parameter 6 kali lebih sedikit dibanding VGG. Meskipun VGG mencapai akurasi tertinggi, modelnya terlalu kompleks dalam klasifikasi objek. Dengan demikian, MobileNet merupakan model yang ideal untuk digunakan di perangkat dengan waktu dan komputasi terbatas.

Pada tahun 2020, Rasywir dkk [45] mengimplementasikan model CNN pada 2490 citra kelapa sawit yang berlabel 11 penyakit yaitu Bercak Daun, Busuk Daun, Karat Daun, Tajuk Daun, Busuk Kuncup, Busuk Pangkal, Ulat Api, Tungau Merah, Kumbang Tanduk, Pengerek Tandan Buah, dan Nematoda Rhadinaphelenchus Cocophilus. Pada penelitian tersebut, rata-rata akurasi dari klasifikasi 11 kelas penyakit adalah 87%.

Pada tahun berikutnya, Ningsih [37] melakukan klasifikasi penyakit *early blight* dan *late blight* pada daun tanaman kentang menggunakan model arsitektur GoogleNet. Data yang digunakan pada penelitian berjumlah 2152 citra daun kentang terdiri dari citra *early blight* dan *late blight* masing-masing berjumlah 1076. Hasil akurasi tertinggi sebesar 89,38% diperoleh dengan pembagian data pelatihan 80% dan data pengujian 20%. Pelatihan menggunakan hyperparameter ukuran batch size 32, epoch 10, dan nilai dropout 0,6. Pada tahun yang sama, Khalifa dkk [46] melakukan penelitian menggunakan Potato Leaf Blight Dataset yang terdiri dari 122 daun sehat, 800 daun *early blight*, dan 800 daun *late blight*. Penelitian tersebut menggunakan model CNN dengan 14 lapisan dan teknik augmentasi. Dataset dibagi menjadi data pelatihan sejumlah 80% dan data

pengujian 20% menghasilkan akurasi klasifikasi 98%.

Selanjutnya Rashid dkk, 2021 [38] melakukan modifikasi model CNN dengan teknik augmentasi YOLOv5s. Penelitian menggunakan PLD Dataset yang terdiri dari 1628 citra early blight, 1414 citra late blight, dan 1020 citra healthy. Citra dibagi dengan perbandingan 80%: 10%: 10% untuk data pelatihan, validasi, dan pengujian. Model yang diajukan disebut PDDCNN (Potato Leaf Disease Detection Convolutional Neural Network). Model ini merupakan multi-level *training* dan *validation* menggunakan model modifikasi CNN dengan YOLOv5s. Akurasi tertinggi yang diperoleh dari model tersebut adalah 99,75% dengan 7 teknik augmentasi, disebutkan dalam penelitian bahwa akurasi tersebut lebih tinggi dibanding model VGG16, InceptionResNetV2, DenseNet\_121, DenseNet169, dan Xception. Parameter pada model PDDCNN juga lebih sedikit dibanding VGG16, InceptionResNetV2, DenseNet169, dan Xception yaitu berjumlah 8,5 juta. Parameter yang lebih sedikit akan menurunkan waktu pelatihan dan harga komputasi. Model PDDCNN memiliki parameter lebih banyak hanya dibanding DenseNet\_121, dengan parameter DenseNet\_121 berjumlah 7 juta dan mencapai akurasi 99,26%. Dengan demikian, model PDDCNN merupakan model yang tidak terlalu kompleks tetapi berhasil mencapai akurasi yang sangat tinggi.

Pada tahun 2022, Lesmana dkk [39] mengimplementasikan model CNN pada 5400 citra daun kentang. Citra terbagi menjadi 3 kelas yaitu 1824 citra *healthy*, 1939 citra *early blight*, dan 1939 citra *late blight*. Akurasi data validasi tertinggi adalah 99% dengan perbandingan data pelatihan: data validasi adalah 80:20. Pelatihan dilakukan 20 epoch dengan 25 steps per epoch, 4 validation steps, dan ukuran citra 150x150 piksel. Total parameter dari model adalah 9,5 juta. Pada tahun yang sama, Kothari dkk [40] membandingkan berbagai model CNN yaitu CNN, GoogleNet, ResNet50, dan VGG16. Penelitian menggunakan 1200 citra daun kentang *early blight*, *late blight*, dan *healthy*. Pada 20 epoch akurasi tertinggi diperoleh CNN yaitu 95%, 30 epoch akurasi klasifikasi tertinggi diperoleh ResNet50 yaitu 98% sama halnya dengan implementasi 40 epoch. Rata-rata waktu pelatihan per epochnya dari keempat model adalah 2 detik untuk CNN, 4 detik untuk GoogleNet, 13 detik untuk ResNet50, dan 17 detik untuk VGG16. Oleh karena itu, model terbaik adalah ResNet50 dengan akurasi tertinggi, serta memiliki waktu pelatihan lebih cepat dibanding VGG.

Pada tahun 2023, Dang dkk [47] menggunakan dataset CottonWeedDet12 berjumlah 5648 citra terdiri dari 12 kelas gulma dengan 9370 bounding box. Model yang digunakan adalah 25 detektor dari 7 model YOLO mulai dari YOLOv3 hingga YOLOv7 untuk dibandingkan kinerjanya. Penelitian ini menghasilkan mAP@0.5 tertinggi 95,22% diperoleh dari model YOLOv4 dan mAP@0.5:0.95 tertinggi 89,72% diperoleh model Scaled-YOLOv4-P6. Secara keseluruhan, semua model YOLOv4 memiliki akurasi deteksi yang tinggi. Namun semua model YOLOv4 memiliki parameter banyak berjumlah

antara 50 - 100 juta yang tentunya meningkatkan waktu komputasi yang dibutuhkan. Sementara model YOLOv5s dan YOLOv5m memiliki jumlah parameter dan waktu inferensi yang lebih sedikit dibanding semua model YOLOv4. Jumlah parameter YOLOv5s sekitar 6 juta dengan mAP@0.5 93,68 dan mAP@0.5:0.95 86,60. Sementara parameter YOLOv5m berjumlah sekitar 20 juta dengan mAP@0.5 94,02% dan mAP@0.5:0.95 87,10%. Waktu inferensi yang dibutuhkan model YOLOv5s dan YOLOv5m adalah 1 – 3 ms, sementara waktu inferensi model YOLOv4 adalah 4 – 7 ms. Dengan demikian, model YOLOv5s dan YOLOv5m memiliki keunggulan yaitu model yang tidak kompleks dan waktu inferensi yang cepat. Hal ini seimbang dengan akurasi deteksi yang komparatif terhadap YOLOv4. Analisis metode yang digunakan dan hasil penelitian tinjauan pustaka diuraikan pada Tabel 2.1

Tabel 2.1. Penelitian menggunakan metode CNN pada tanaman

Peneliti	Model	Akurasi Klasifikasi
KC dkk, (2019) [44]	MobileNet	98,65%
Rasywir dkk, (2020) [45]	CNN	87%
Ningsih, (2021) [37]	GoogleNet	89,38%
Khalifa dkk, (2021) [46]	CNN	98%
Rashid dkk, (2021) [38]	YOLOv5s	99,75%
Lesmana dkk, (2022) [39]	CNN	99%
Kothari dkk, (2022) [40]	ResNet50	98%

Berdasarkan penelitian-penelitian tersebut dapat diketahui bahwa implementasi YOLO dapat meningkatkan akurasi model klasifikasi. Hal ini ditunjukkan di penelitian Rashid dkk [38] bahwa akurasi tertinggi didapatkan dengan model implementasi YOLO. Penelitian tersebut juga menunjukkan bahwa implementasi YOLO juga memiliki parameter jauh lebih sedikit dibanding VGG16, InceptionResNetV2, DenseNet169, dan Xception. Jumlah parameter mempengaruhi harga komputasi dan waktu inferensi [47], maka dapat diketahui bahwa implementasi YOLO dapat meningkatkan akurasi klasifikasi tanpa meningkatkan komputasi karena jumlah parameternya yang sedikit. Pada deteksi objek, YOLO juga memiliki keunggulan dalam hal komputasi karena menggabungkan proses klasifikasi dan lokalisasi dalam satu tahap, sehingga YOLO termasuk ke dalam *one-stage detector*. Menurut Dang dkk [47] *one-stage detector* lebih cepat dalam inferensi dibanding *two-stage detector*.

Kelebihan YOLO dalam harga komputasi membuat YOLO populer digunakan untuk deteksi objek, dengan selalu ditingkatkan akurasi deteksinya. Beberapa penelitian deteksi objek menggunakan YOLO diantaranya: [48] Deteksi wajah manusia, [49] Deteksi objek makanan cepat saji, [50] Deteksi rambu lalu lintas, [51] Deteksi kesegaran buah mangga, [52] Deteksi jenis mobil, [53] Deteksi alat pelindung diri, [42] Deteksi

kendaraan, [54] Deteksi kerusakan aspal, dan [55] Deteksi fasilitas rumah sakit. Penelitian Hu dkk [55] membandingkan kinerja *one-stage detector* YOLOv5-L, YOLOX-L, YOLOv6-L, dan YOLOv7, serta *two-stage detector* Faster R-CNN, Deformable DETR, VFNet, dan DyHead. Berdasarkan penelitian tersebut, penulis merekomendasikan *one-stage detector* untuk penelitian visi komputer karena kinerjanya seimbang antara akurasi dan komputasinya. Dari beberapa implementasi YOLO yang disebutkan, hanya Hu dkk [55] dan Dang dkk [47] yang membandingkan kinerja beberapa model YOLO. Hal ini menunjukkan bahwa belum banyak penelitian yang membandingkan kinerja model YOLO. Selain itu, implementasi YOLO untuk deteksi penyakit tanaman masih jarang, khususnya pada tanaman kentang. Oleh karena itu, dalam penelitian ini akan diimplementasikan dan dianalisis kinerja model YOLO.

## 2.2 Dasar Teori

Berisi dasar teori yang digunakan dalam penulisan skripsi. Dasar teori diperoleh dari berbagai referensi yang dapat dipertanggungjawabkan.

### 2.2.1 Penyakit Tanaman Kentang

Kentang adalah salah satu tanaman umbi dengan nama ilmiah *Solanum tuberosum L.* Tanaman kentang tumbuh optimal pada suhu di antara 15-30 °C dengan kelembaban 80 – 90 % [56]. Kondisi lingkungan yang tidak optimal akan menyebabkan terhambatnya pembentukan umbi kentang. Umbi kentang memiliki kandungan kalori, protein, vitamin, mineral dan antioksidan. Tingginya kandungan gizi pada kentang dapat meningkatkan permintaan terhadap kentang.

Tingginya angka permintaan kentang di Indonesia dibuktikan oleh tingginya angka impor kentang Indonesia yaitu sebanyak 7160 ton [28]. Hal ini menunjukkan bahwa produksi kentang di Indonesia belum dapat memenuhi permintaan. Produktivitas kentang di Indonesia pada tahun 2022 juga masih tergolong rendah yaitu sebesar 19,2 ton/ha. Terdapat beberapa faktor yang mengakibatkan rendahnya produktivitas kentang, salah satunya adalah terserangnya tanaman kentang oleh patogen. Patogen adalah organisme hidup yang dapat menyebabkan penyakit pada tanaman, seperti jamur, bakteri, dan virus [57]. Namun, patogen tidak akan menimbulkan gejala penyakit apabila kondisi tanaman dan lingkungan tidak mendukung patogen. Patogen akan berkembang pada tanaman yang rentan dan lingkungan yang sesuai bagi pertumbuhan patogen. Penyakit yang umumnya menyerang kentang akibat adanya patogen adalah penyakit *early blight* (bercak kering) dan penyakit *late blight* (hawar daun).

### **2.2.1.1 Penyakit Tanaman Kentang *Early Blight***

*Early blight* atau penyakit bercak kering disebabkan oleh patogen jamur *Alternaria solani*, patogen ini menyukai suhu di antara 24-29 °C dan kelembaban tinggi [58]. Patogen jamur dapat menginfeksi tanaman kentang melalui bibit atau tanah yang terkontaminasi. Produksi spora jamur *Alternaria solani* meningkat apabila terjadi cuaca basah dan kering secara bergantian. Hal ini biasanya terjadi ketika periode curah hujan tinggi di daerah tropis dan subtropis.

Penyakit *early blight* terjadi pada daun yang lebih tua dengan gejala bercak cincin melingkar berwarna abu-abu hingga coklat kekuningan [34]. Penyakit ini mengakibatkan daun akan gugur sehingga kentang rentan terhadap luka bakar matahari. Bercak ini juga menjalar ke batang dan umbi kentang. Menurut Xue dkk [59], penyakit *early blight* merupakan penyakit daun kentang yang sangat penting bagi perekonomian dunia. Penyakit tersebut bisa menyebabkan kematian hingga gagal panen. Gambar 2.1 menunjukkan contoh daun dari tanaman kentang yang terinfeksi *early blight*.



Gambar 2.1. Penyakit *Early Blight* [1]

Penanganan tanaman kentang terinfeksi *early blight* dapat dilakukan dengan dua pengendalian, yaitu pengendalian hayati dan kimiawi. Pengendalian hayati yang dapat dilakukan salah satunya adalah pemberian campuran batu kapur alga, susu bebas lemak, dan air [60].

### **2.2.1.2 Penyakit Tanaman Kentang *Late Blight***

*Late blight* atau penyakit hawar daun disebabkan oleh patogen *oomycete* (jamur air) *Phytophthora infestans*, patogen ini menyukai suhu dingin di antara 12-18 °C [58]. Jamur *Phytophthora infestans* termasuk parasit obligat yang menumpang pada inangnya untuk bertahan hidup melewati musim dingin. Jamur ini menginfeksi inang melalui luka dan robekan di kulit pada kondisi basah yang panjang dengan kelembaban relatif 90%. Penyakit ini umumnya menyerang tanaman kentang berumur 5-6 minggu.

Gejala penyakit *late blight* adalah membusuknya daun dengan munculnya bercak coklat mulai dari tepi daun dan tumbuhnya jamur menyerupai kapas di bagian bawah daun. Gejala ini akan menyebabkan daun layu dan menyebar ke tangkai, batang, serta umbi [61]. Umbi kentang yang terinfeksi akan memiliki kulit berbintik biru dan daging berwarna coklat sehingga tidak bisa dikonsumsi. Oleh karena itu, penyakit ini membahayakan tanaman dan menyebabkan petani gagal panen hingga menimbulkan aroma membusuk pada lahan. Penyakit ini dapat dikendalikan dengan pengaplikasian fungisida yang melapisi daun tanaman kentang [62]. Selain itu, penyakit ini dapat dicegah dengan pembersihan gulma yang berpotensi menularkan penyakit. Gambar 2.2 menunjukkan contoh daun dari tanaman kentang yang terinfeksi *late blight*.



Gambar 2.2. Penyakit *Late Blight* [2]

### 2.2.2 Deteksi Objek

Deteksi objek adalah pengembangan dari klasifikasi objek. Klasifikasi objek dan deteksi objek merupakan bagian dari bidang ilmu visi komputer. Visi komputer (*computer vision*) adalah cabang ilmu yang mengembangkan algoritma untuk membantu komputer mengenal, mempelajari, dan memahami sebuah citra sebagaimana penglihatan manusia [63]. Komputer dilatih menggunakan citra yang sudah melalui proses anotasi atau pelabelan objek, sehingga komputer dapat mengenali objek yang sama pada citra yang baru. Visi komputer merupakan pengetahuan dan kecerdasan komputer untuk mengetahui dan menganalisis citra sehingga menghasilkan suatu interpretasi atau persepsi bagi komputer. Interpretasi yang dihasilkan pada proses klasifikasi objek adalah kelas objek. Sementara interpretasi yang dihasilkan pada proses deteksi objek adalah kelas dan letak suatu objek pada citra. Dengan demikian, proses deteksi objek pada bidang visi komputer adalah proses klasifikasi dan lokalisasi objek. Deteksi objek menjawab permasalahan ketika harus mengenali beberapa objek yang sangat mirip, tetapi masuk ke kelas yang

berbeda. Gambar 2.3 adalah contoh perbedaan hasil klasifikasi dan deteksi terhadap anjing dan serigala yang sangat mirip apabila dilihat secara sekilas.

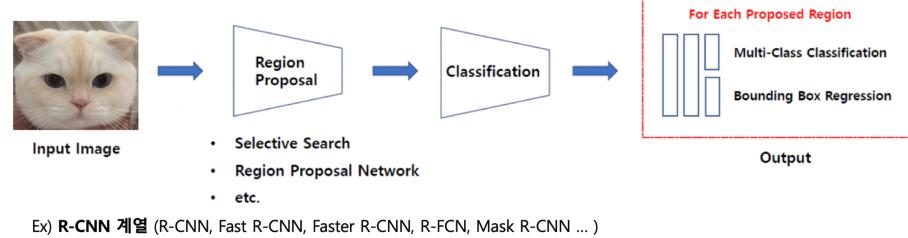


Gambar 2.3. Contoh Perbedaan Klasifikasi Objek dan Deteksi Objek [3]

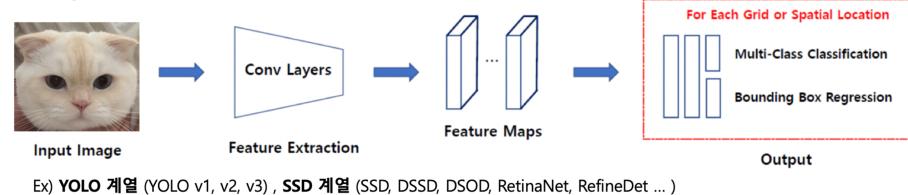
Gambar 2.3 menunjukkan bahwa hasil klasifikasi objek masih bersifat ambigu. Bagi orang yang belum mengenal perbedaan anjing dan serigala, orang tersebut mungkin ragu menentukan objek yang mana yang merupakan anjing atau serigala. Hal ini dapat diatasi dengan pemisahan citra menjadi hanya memiliki satu objek. Namun pada lahan kentang berskala besar, kurang efisien apabila klasifikasi penyakit dilakukan satu per satu daun. Teknologi deteksi objek mempermudah pengenalan beberapa objek dalam satu citra dengan memperjelas posisi objek. Deteksi objek dapat dilakukan dengan 2 metode, yaitu metode tradisional dan *deep learning* [42]. Deteksi objek yang tergolong tradisional adalah *Viola Jones Detector*, *Histogram of Oriented Gradients Detector*, dan *Deformable Part-based Model*. Deteksi objek berbasis *deep learning* dikembangkan sejak tahun 2014 dan dinyatakan memiliki kinerja lebih baik dibanding metode tradisional [43]. Deteksi objek berbasis *deep learning* tergolong menjadi 2 detektor yaitu *two-stage detector* dan *one-stage detector*. Detektor yang tergolong *two-stage detector* adalah Region-based Convolutional Neural Networks (R-CNN), Fast R-CNN, Faster R-CNN, dan SPPNet. Detektor yang tergolong *one-stage detector* adalah *You Only Look Once* (YOLO), *Single Shot Multibox Detector* (SSD), dan RetinaNet. *Two-stage detector* memiliki 2 tahap dalam mendekripsi objek. Tahap pertama adalah tahap pre-proses yaitu menghasilkan *region proposal* dari objek, kemudian tahap kedua mengklasifikasi objek dan melakukan regresi *bounding box*. Kedua tahap tersebut dapat meningkatkan akurasi klasifikasi dan lokalisasi objek, tetapi juga meningkatkan waktu dan komputasi. Sedangkan *one-stage detector* tidak melakukan proses *region proposal*, melainkan langsung melakukan deteksi objek dalam satu tahap. Oleh karena itu, *one-stage detector* lebih efisien dalam komputasi dan lebih cepat dalam inferensi. Perbedaan proses deteksi menggunakan *one-stage detector* dan *two-stage detector* diilustrasikan pada Gambar 2.4.

Dalam perkembangannya, metode deteksi objek selalu dievaluasi kinerjanya oleh para peneliti di bidang visi komputer. Hal ini bertujuan agar metode yang digunakan sesuai dengan kebutuhan dalam proses deteksi. Kinerja model deteksi objek yang dianalisis

**2-Stage Detector** - Regional Proposal와 Classification이 순차적으로 이루어짐.



**1-Stage Detector** - Regional Proposal와 Classification이 동시에 이루어짐.

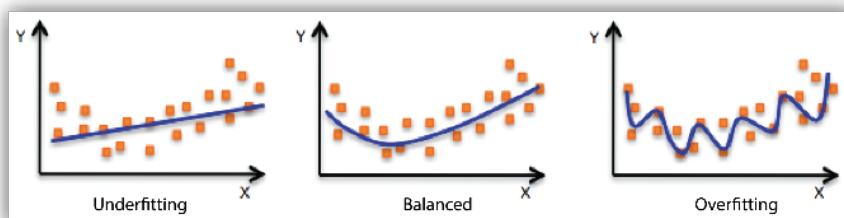


Gambar 2.4. Perbedaan *two-stage detector* dan *one-stage detector* [4]

meliputi akurasi deteksi, kompleksitas model, dan waktu komputasi.

### 2.2.3 Akurasi Deteksi

Model dengan akurasi yang baik adalah model yang tidak mengalami *underfitting* dan *overfitting*. *Underfitting* terjadi ketika model belum mempelajari tren pola data dengan optimal, sedangkan *overfitting* terjadi ketika model hanya mempelajari data yang ada [64]. Model mempelajari data latih terlalu baik sehingga tidak dapat memprediksi data baru. Model yang baik adalah model yang berhasil mempelajari tren pola data sehingga mampu memprediksi data baru, hal ini disebut *balanced*. Ilustrasi perbedaan *underfitting*, *balanced*, dan *overfitting* ditunjukkan pada Gambar 2.5.



Gambar 2.5. Perbedaan *underfitting*, *balanced*, dan *overfitting* [5]

Untuk mengevaluasi apakah model sudah *balanced*, maka perlu dilakukan pengujian terhadap model yang sudah dilatih. Model yang sudah dilatih dapat diujikan pada data validasi atau data pengujian. Berdasarkan pengujian model, kita dapat memvisualisasikan kinerja model dengan *confusion matrix* dan mengukur akurasi model dengan *performance evaluation metrics*.

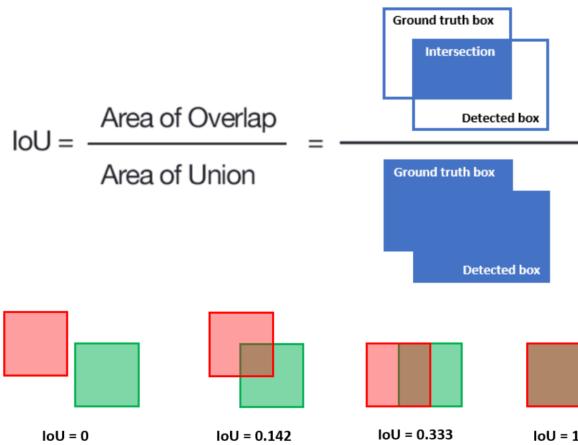
### 2.2.3.1 Confusion Matrix

*Confusion matrix* dapat ditentukan dengan membandingkan kelas aktual dengan kelas prediksi dari suatu objek. *Confusion matrix* berupa matriks yang berisi nilai kejadian *True Positive*, *True Negative*, *False Positive*, dan *False Negative*. Visualisasi dari *confusion matrix* dapat ditunjukkan pada Gambar 2.6.

		Real	
		Positive	Negative
Predicted	Positive	True Positive (tp)	False Positive (fp)
	Negative	False Negative (fn)	True Negative (tn)

Gambar 2.6. *Confusion Matrix* [6]

Untuk menghitung nilai-nilai kejadian pada *confusion matrix*, dalam deteksi objek dikenal *Jaccard Index* atau disebut juga *Intersection over Union* (IoU). IoU membandingkan area irisan terhadap area gabungan antara kotak label pada objek ( $B_{aktual}$ ) dengan kotak prediksi ( $B_{prediksi}$ ). IoU dapat diilustrasikan dengan Gambar 2.7



Gambar 2.7. *Intersection over Union* [7]

Perhitungan IoU ditunjukkan pada Persamaan 2-1 [9].

$$IoU = \left\{ \frac{B_{aktual} \cap B_{prediksi}}{B_{aktual} \cup B_{prediksi}} \right\} \quad (2-1)$$

Semakin tinggi nilai IoU maka akurasi model semakin bagus. Dengan mendapat nilai IoU, model akan memprediksi kejadian positif apabila nilai IoU bernilai lebih dari

ambang batas ( $\alpha$ ) yang ditentukan. Ambang batas yang terlalu rendah akan menyebabkan kemampuan model dalam mendekripsi lokasi menjadi rendah. Namun ambang batas yang terlalu tinggi sulit dilakukan model karena hasil deteksi lokasi objek tidak dapat sangat tepat dengan lokasi sebenarnya. Selanjutnya menggunakan nilai IoU dan ambang batas dapat ditentukan nilai-nilai *confusion matrix* sebagai berikut:

1. ***True Positive***: Model memprediksi objek secara tepat dengan nilai  $IoU \geq \alpha$ . Contohnya ketika objek berlabel X, diprediksi oleh model ke kelas X dengan lokasi kotak prediksi sebagian besar tepat sesuai dengan kotak aktual. Artinya jumlah nilai *true positive* dipengaruhi oleh ambang batas IoU ( $\alpha$ ). Pada model deteksi *multi-class*, maka nilai *true positive* dihitung terhadap setiap kelas.
2. ***True Negative***: Kejadian ini terjadi apabila model tepat memprediksi suatu kelas ke kelas negatif. Dalam deteksi objek, hal ini tidak berlaku karena terlalu banyak kemungkinan model dalam memprediksi *background* [65].
3. ***False Positive***: Model memprediksi objek ke kelas X, tetapi objek sebenarnya bukan berlabel X. Pada deteksi objek, kejadian ini juga terjadi apabila kotak prediksi memiliki nilai  $IoU < \alpha$  terhadap kotak aktual. Artinya jumlah nilai *false positive* dipengaruhi oleh ambang batas IoU ( $\alpha$ ). Pada model deteksi *multi-class*, maka nilai *false positive* dihitung terhadap setiap kelas.
4. ***False Negative***: Model tidak dapat memprediksi objek ke kelas objek yang sebenarnya. Misalnya objek berlabel X tidak dapat diprediksi oleh model sebagai kelas X. Pada model deteksi *multi-class*, maka nilai *false negative* dihitung terhadap setiap kelas.

### 2.2.3.2 Performance Evaluation Metrics

Setelah menentukan nilai *true positive*, *true negative*, *false positive*, dan *false negative*, kita dapat mengukur akurasi menggunakan metrik evaluasi kinerja. Beberapa metrik yang digunakan untuk mengukur akurasi model deteksi objek adalah *precision* (presisi), *recall* atau *sensitivity* (sensitivitas), dan *mean Average Precision* (mAP).

1. ***Precision***: *Precision* merupakan perbandingan antara citra terdeteksi positif benar dengan seluruh citra terdeteksi positif. *Precision* yang dihasilkan model dapat direpresentasikan dengan Persamaan 2-2 [65].

$$Precision = \left\{ \frac{TP}{TP + FP} \right\} \quad (2-2)$$

*Precision* akan bernilai rendah jika jumlah *false positive* tinggi, hal ini akan terjadi jika ambang batas IoU rendah. Sebaliknya *precision* akan bernilai tinggi jika jumlah *false positive* rendah, hal ini akan terjadi jika ambang batas IoU tinggi. Tetapi

ambang batas IoU terlalu tinggi menyebabkan banyak objek tidak terdeteksi sehingga *false negative* menjadi tinggi, sehingga dapat menurunkan akurasi model. Oleh karena itu, metrik *precision* saja belum cukup dijadikan tolok ukur dalam mengevaluasi akurasi model.

2. ***Recall***: *Recall* merupakan perbandingan antara citra terdeteksi positif benar dengan seluruh citra positif. *Recall* yang dihasilkan model dapat direpresentasikan dengan Persamaan 2-3 [65].

$$Recall = \left\{ \frac{TP}{TP + FN} \right\} \quad (2-3)$$

*Recall* akan bernilai rendah jika jumlah *false negative* tinggi, hal ini akan terjadi jika ambang batas IoU tinggi. Sebaliknya *recall* akan bernilai tinggi jika jumlah *false negative* rendah, hal ini akan terjadi jika ambang batas IoU rendah. Tetapi ambang batas IoU terlalu rendah menyebabkan banyak objek terdeteksi tidak tepat sehingga *false positive* menjadi tinggi, sehingga dapat menurunkan akurasi model. Oleh karena itu, metrik *recall* saja belum cukup dijadikan tolok ukur dalam mengevaluasi akurasi model.

3. ***mean Average Precision***: Adanya *trade-off* antara metrik *precision* dan *recall*, menunjukkan bahwa model yang baik adalah model yang memiliki *precision* dan *recall* tetap tinggi pada berbagai variasi ambang batas ( $\alpha$ ) IoU. Pada deteksi objek dikenal suatu kurva yang mem-*plotting* nilai *precision* terhadap *recall*. Kurva *Precision-Recall* menunjukkan akurasi model dengan menghitung luas area di bawah kurva. Luas *Area Under Curve* (AUC) mempresentasikan rata-rata *precision* di semua nilai *recall*. Luas AUC bernilai antara 0 hingga 1 yang dapat dijadikan metrik untuk mengukur akurasi deteksi objek, metrik ini disebut *Average Precision* (AP). Cara menghitung AP dapat ditunjukkan dengan Persamaan 2-4 [66].

$$AP@\alpha = \left\{ \int_0^1 p(r)dr \right\} \quad (2-4)$$

Namun metrik ini hanya dapat menghitung akurasi deteksi satu kelas objek. Pada model yang dapat mendeteksi objek *multi-class* atau lebih dari satu kelas, maka perlu dihitung terlebih dahulu nilai AP setiap kelas. Kemudian untuk mengukur akurasi model secara keseluruhan, dicari rata-rata dari nilai AP setiap kelas. Oleh karena itu, terdapat metrik yang disebut *mean Average Precision* (mAP) untuk mengukur akurasi deteksi objek *multi-class*. Cara menghitung mAP ditunjukkan pada Persamaan 2-5 [66].

$$mAP@\alpha = \left\{ \frac{1}{n} \sum_{i=1}^n AP_i \right\} \quad (2-5)$$

Nilai mAP dapat dihitung dengan berbagai ambang batas ( $\alpha$ ) IoU, metrik yang sering digunakan pada deteksi objek adalah mAP@0.5 dan mAP@0.5:0.95. mAP@0.5 artinya menggunakan ambang batas IoU 0,5, sedangkan mAP@0.5:0.95 artinya menghitung metrik dengan variasi nilai ambang batas IoU dari 0,5 hingga 0,95.

#### 2.2.4 Kompleksitas Model

Kompleksitas model dapat diukur dari jumlah parameter pada model [47]. Semakin banyak jumlah parameter model menyebabkan kebutuhan memori yang lebih tinggi dalam proses pelatihan maupun penerapannya. Hal ini akan mempengaruhi harga dan waktu komputasi. Parameter dari model deteksi objek berisi bobot dan bias. Setiap model memiliki lapisan yang memproses data *input* dari lapisan sebelumnya menjadi suatu *output* yang akan diteruskan ke lapisan berikutnya. Proses inilah yang memanfaatkan nilai bobot dan bias untuk meningkatkan akurasi model. Nilai bobot dan bias pada proses pelatihan akan terus diperbarui. Semakin besar nilai bobot dari sebuah input, maka input tersebut memiliki kontribusi semakin tinggi bagi lapisan berikutnya. Nilai bias juga menentukan apakah informasi dari input lebih baik diteruskan ke lapisan berikutnya atau tidak.

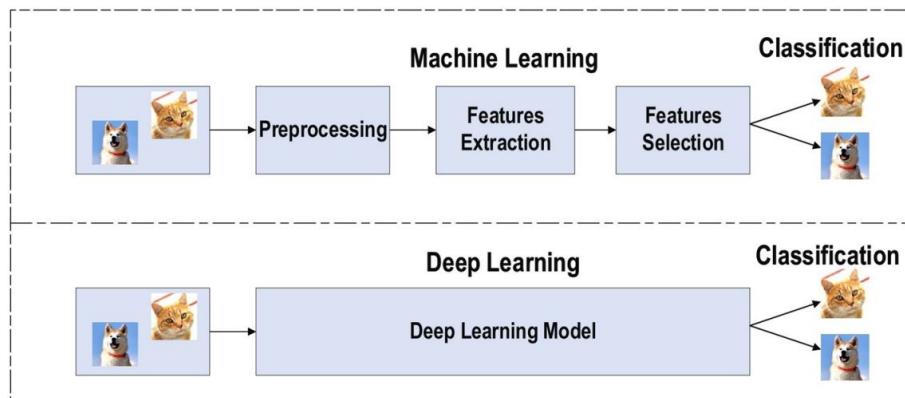
#### 2.2.5 Waktu Komputasi

Waktu komputasi dipengaruhi oleh kompleksitas model dan perangkat keras yang digunakan dalam komputasi deteksi objek. Waktu komputasi perlu dipertimbangkan dalam penerapan model karena hal ini akan mempengaruhi harga komputasi. Pada deteksi objek, waktu yang dibutuhkan dalam proses komputasi terdiri dari waktu pelatihan dan waktu deteksi. Waktu pelatihan adalah waktu yang dibutuhkan dalam melatih model dan memvalidasi model menggunakan dataset pelatihan dan validasi sehingga model akan mengenal dan memahami objek. Waktu deteksi adalah waktu yang dibutuhkan model dalam mendekripsi objek pada suatu citra. Waktu deteksi terdiri dari waktu pra-inferensi, waktu inferensi, dan waktu pasca-inferensi. Waktu pra-inferensi adalah waktu yang dibutuhkan sebelum model melakukan prediksi terhadap citra. Waktu inferensi adalah waktu yang diperlukan model untuk memprediksi objek pada citra. Dalam visi komputer, model akan memprediksi beberapa *bounding box* yang *overlap*. Waktu pasca-inferensi adalah waktu yang dibutuhkan untuk memproses citra yang sudah diprediksi, termasuk proses *Non Maximum Suppression* (NMS). NMS adalah proses menyeleksi *bounding box* yang *overlap* menjadi satu *bounding box* berdasarkan nilai IoU [67].

#### 2.2.6 Deep Learning

*Deep learning* adalah bagian dari bidang keilmuan *Artificial Intelligence* yang merupakan turunan dari *machine learning*. *Artificial Intelligence* adalah sistem komputasi yang meniru kecerdasan manusia dalam berpikir, bernalar, dan beradaptasi untuk

membantu penyelesaian tugas manusia. *Machine learning* adalah bagian dari *artificial intelligence* yang memungkinkan komputer memiliki kemampuan mempelajari pola data yang diberikan. Model *machine learning* dilatih menggunakan ekstraksi dan seleksi fitur untuk menyelesaikan tugas yang diberikan manusia. *Deep learning* merupakan bagian dari *machine learning* yang dikembangkan sehingga memiliki kinerja lebih unggul dibanding *machine learning*. *Deep learning* memiliki lapisan jaringan yang lebih dalam dibanding *machine learning*. Hal inilah yang membuatnya disebut “*deep learning*”. *Deep learning* mampu mempelajari data tanpa diberikan fitur oleh manusia, model akan membuat fitur secara mandiri berdasarkan pola dari data yang diberikan. Perbedaan antara *deep learning* dan *machine learning* dapat diilustrasikan pada Gambar 2.8.



Gambar 2.8. Perbedaan *Machine Learning* dengan *Deep Learning* [8]

*Deep learning* memiliki jaringan saraf untuk belajar dan berlatih memahami data yang besar. Jaringan saraf tersebut memiliki komputasi sangat kompleks dan berlapis-lapis menyerupai jaringan saraf manusia. Komputasi *deep learning* akan membuat komputer semakin cerdas ketika data yang dipelajari semakin bertambah. Dengan demikian, *deep learning* membutuhkan data yang besar, perangkat komputasi yang memadai, dan waktu pelatihan yang panjang. *Deep learning* terbagi menjadi *deep supervised learning*, *deep semi-supervised learning*, *deep unsupervised learning*, dan *deep reinforcement learning* [8]. Pada penelitian ini digunakan *deep supervised learning*. Beberapa teknik *deep supervised learning* adalah RNN, CNN, dan DNN.

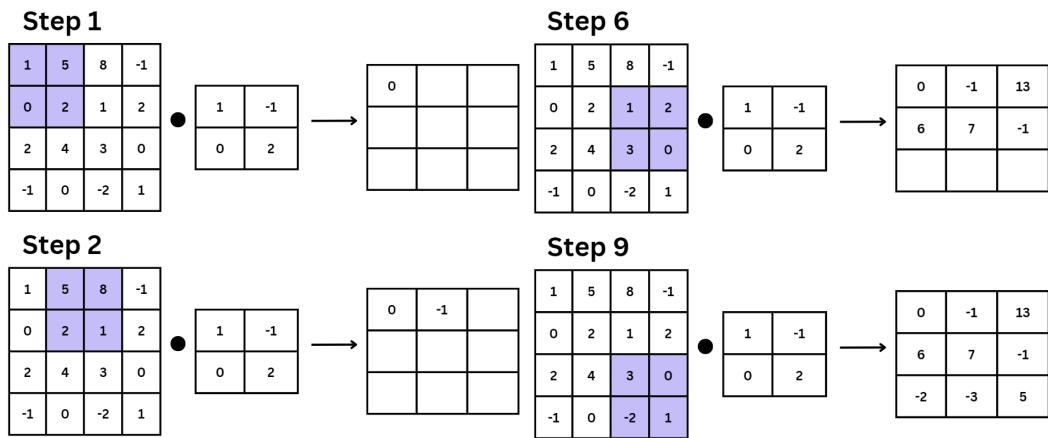
### 2.2.7 Convolutional Neural Network (CNN)

*Convolutional Neural Network* adalah salah satu teknik *deep supervised learning*. CNN lebih unggul dibanding teknik pendahulunya seperti RNN, ANN, dan DNN karena fiturnya lebih kompatibel dan komputasinya lebih hemat. CNN merupakan teknik *deep learning* yang jaringannya memiliki bobot bersama dan koneksi lokal. Bobot bersama dan koneksi lokal pada CNN menyederhanakan jaringan dan mempercepat proses pelatihan [8]. Fungsi tersebut sangat berguna untuk memproses *input* 2 dimensi seperti citra.

Hal ini menyerupai sistem pengenalan pada *visual cortex* manusia [36]. Arsitektur CNN memiliki beberapa komponen yaitu *convolutional layer*, *pooling layer*, *fully connected layer*, *activation function*, dan *loss function*.

### 2.2.7.1 Convolutional Layer

Convolutional layer adalah lapisan yang melakukan operasi konvolusional suatu citra *input* dengan *kernel* yang berisi bobot tertentu. Citra *input* diilustrasikan sebagai matriks piksel dimana *kernel* akan bergeser sesuai ukurannya, kemudian dilakukan operasi dot antara piksel citra *input* dengan matriks bobot *kernel*. Operasi konvolusional menghasilkan *output feature map* yang diilustrasikan pada Gambar 2.9



Gambar 2.9. Ilustrasi Proses di *Convolutional Layer*

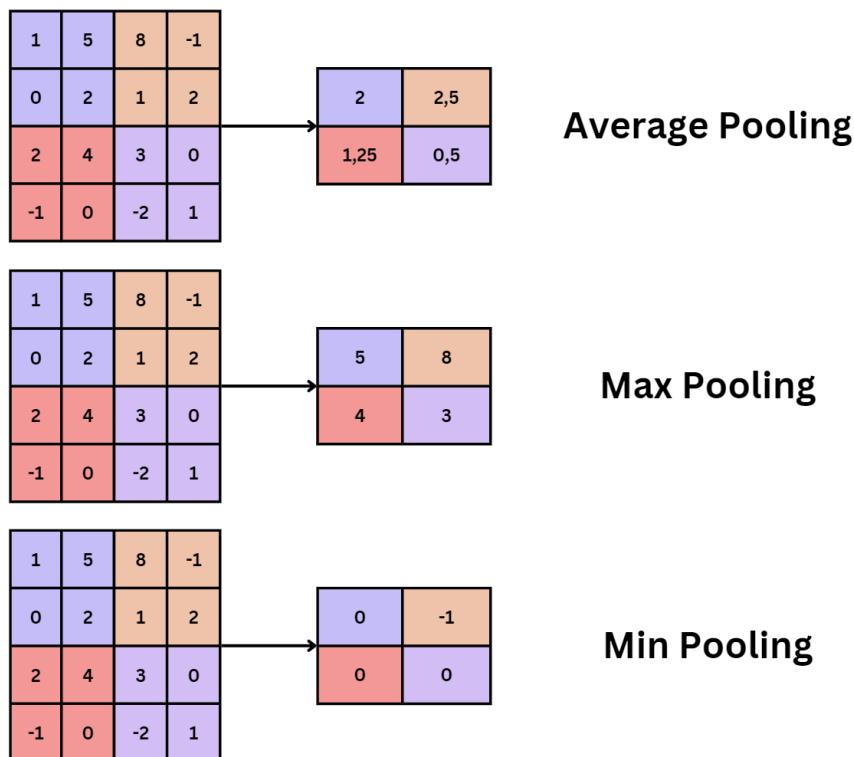
### 2.2.7.2 Pooling Layer

*Pooling layer* adalah lapisan yang menyaring *feature map* yang dihasilkan dari *convolutional layer*. Proses ini akan mengurangi ukuran *feature map* yang akan berdampak pada jumlah parameter sehingga mempercepat komputasi. Metode *pooling* antara lain *tree pooling*, *gated pooling*, *average pooling*, *min pooling*, *max pooling*, *global average pooling*, dan *global max pooling*. *Pooling* yang familiar digunakan pada CNN adalah *average pooling*, *max pooling*, dan *min pooling*. *Average pooling* akan memperhalus citra, *max pooling* akan memilih piksel paling terang, sedangkan *min pooling* akan memilih piksel paling gelap [68]. Gambar 2.10 mengilustrasikan proses *pooling*.

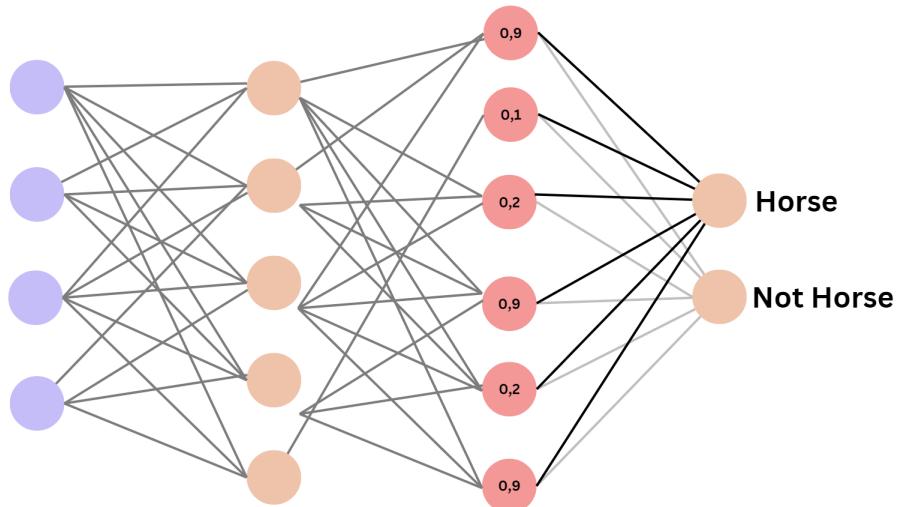
### 2.2.7.3 Fully Connected Layer

*Fully connected layer* adalah lapisan dimana setiap saraf terkoneksi dengan saraf lainnya, terletak di akhir jaringan CNN.

*Input* di jaringan ini berupa *feature map* yang sudah dilakukan *flattening*. *Flattening* mengubah *feature map* yang berupa array multidimensi menjadi vektor. *Output* jaringan ini berupa hasil prediksi akhir diilustrasikan pada Gambar 2.11.



Gambar 2.10. Ilustrasi Proses di *Pooling Layer*



Gambar 2.11. Ilustrasi Proses di *Fully Connected Layer*

#### 2.2.7.4 Activation Function (Fungsi Aktivasi)

*Activation function* adalah fungsi yang memetakan *input* ke *output* dengan membedakan fitur yang signifikan. Fungsi aktivasi digunakan untuk memetakan hasil akumulasi *input* dengan nilai bobot dan bias, proses ini disebut *forward pass*. Fungsi aktivasi dibagi menjadi 2 yaitu fungsi linear dan non-linear. Fungsi linear adalah fungsi aktivasi yang hanya meneruskan *input* menjadi *output* tanpa perubahan nilai. Sedangkan fungsi

non-linear memiliki nilai tertentu untuk mengubah nilai input. Fungsi aktivasi yang banyak digunakan pada CNN adalah fungsi non-linear, terutama pada *hidden layer*. Fungsi non-linear diantaranya fungsi sigmoid, tanh, ReLU (*Rectified Linear Unit*), leaky ReLU, mish, SiLU (*Sigmoid Linear Unit*). Fungsi sigmoid memetakan input menjadi output bilangan di antara 0 dan 1. Fungsi tanh memetakan input menjadi output bilangan di antara -1 dan 1. Fungsi ReLU memetakan input ke bilangan positif. Fungsi sigmoid, tanh, dan ReLU memiliki kekurangan jika digunakan dalam model berlapis banyak, karena nilai turunan fungsinya akan sangat kecil. Hal ini akan memperlambat proses pelatihan model. Fungsi aktivasi Leaky ReLU memiliki kelebihan dalam mengatasi masalah turunan fungsi. Fungsi aktivasi ini digunakan pada YOLOv1 hingga YOLOv3. Kemudian terdapat fungsi aktivasi mish yang memiliki kinerja lebih bagus dibanding ReLU tetapi memiliki komputasi lebih banyak [69]. Fungsi mish mulai digunakan pada model YOLOv4. Fungsi aktivasi selalu dikembangkan oleh peneliti di bidang *deep learning*, sehingga masih banyak fungsi aktivasi lainnya. Fungsi aktivasi yang digunakan pada model disesuaikan dengan tujuan model *deep learning* yang diimplementasikan.

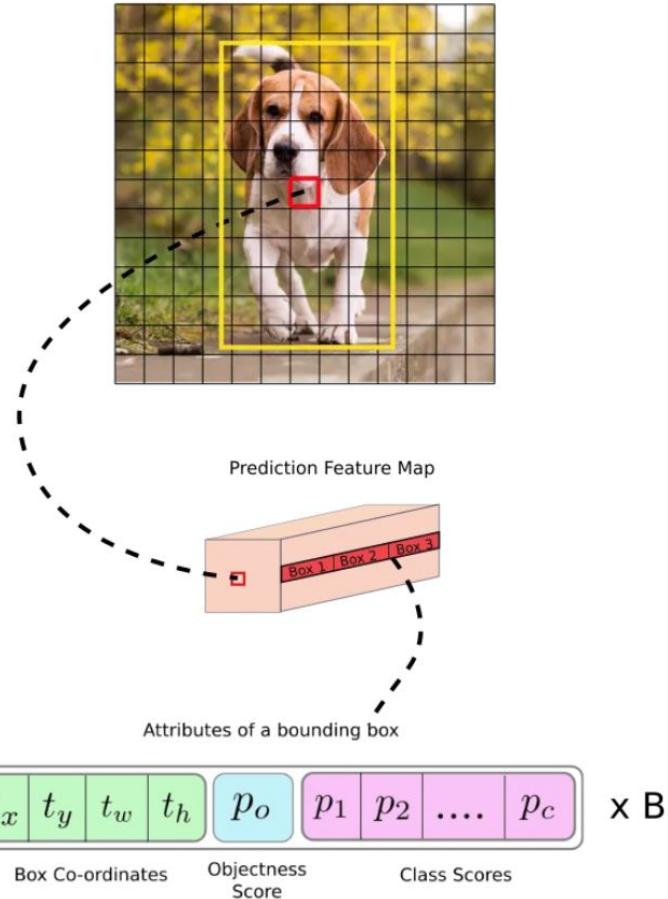
#### 2.2.7.5 *Loss Function* (Fungsi Kesalahan)

*Loss function* adalah fungsi yang menghitung prediksi kesalahan dari pelatihan dalam jaringan model. Kesalahan mempresentasikan perbedaan data aktual dengan hasil prediksi. Hasil prediksi sebagai parameter pertama dan data aktual sebagai parameter kedua dalam perhitungan *loss*. Jaringan saraf bekerja dengan optimisasi berbasis turunan *loss function* terhadap bobot. Optimisasi ini disebut *Gradient Descent* yang *step*-nya ditentukan oleh tiap *learning rate*. Optimisasi ini akan memperbarui nilai parameter untuk menurunkan nilai *loss* sehingga meningkatkan akurasi model. Proses ini disebut *back-propagation error*. Beberapa *loss function* yang digunakan pada model YOLO adalah *cross-entropy* atau *softmax loss function*, *focal loss*, *IoU loss*, *distribution focal loss*, dan *varifocal loss*.

#### 2.2.8 *You Only Look Once* (YOLO)

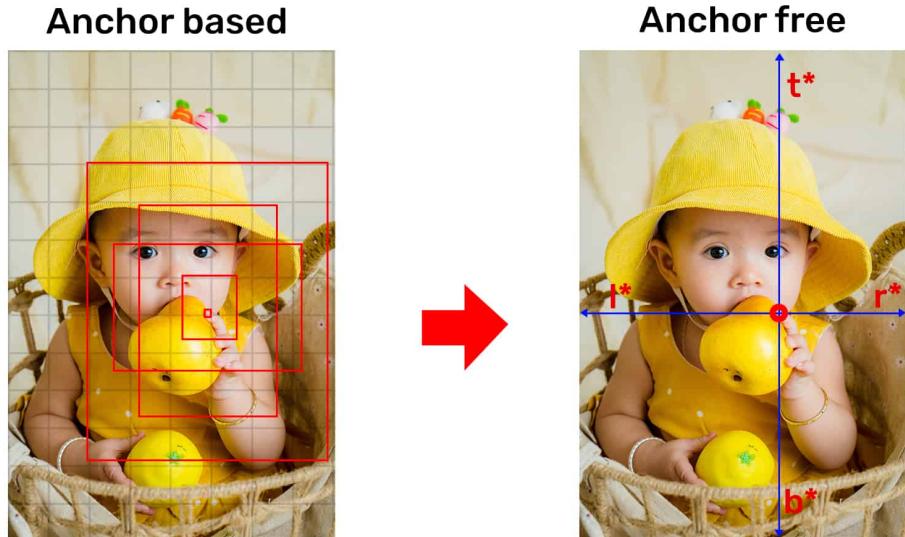
*You Only Look Once* (YOLO) adalah algoritma deteksi objek pertama yang berbasis *one-stage detector*. YOLO melakukan pendekatan regresi dan probabilitas dengan jaringan CNN tunggal dalam mendekripsi objek. YOLO bekerja dengan membagi citra ke beberapa *grid*, kemudian setiap *grid* memprediksi *bounding box* dan probabilitas kelas [48]. *Bounding box* adalah kotak pembatas yang memprediksi 5 nilai terkait objek pada setiap *grid* berukuran s x s. *Bounding box* memprediksi *object confidence score*, x, y, *width*, dan *height* [9]. *Object confidence score* menunjukkan tingkat kepercayaan model terhadap keberadaan objek yang mirip dengan objek pada data pelatihan. Nilai x dan y adalah koordinat titik pusat dari objek yang dideteksi pada *grid*. Nilai *width* dan *height* adalah lebar dan tinggi *bounding box* dari objek. Kemudian *grid* akan memprediksi nilai

probabilitas kelas jika berdasarkan *confidence score* diprediksi terdapat objek di dalam *grid* tersebut. Prinsip kerja algoritma YOLO dipaparkan pada Gambar 2.12.



Gambar 2.12. Prinsip Kerja Algoritma YOLO [9]

Model YOLO terbagi menjadi dua berdasarkan cara kerjanya, yaitu *anchor-free* dan *anchor-based*. *Anchor-free* merepresentasikan objek sebagai *keypoint* yang merupakan pusat dari objek yang dideteksi pada *grid*. Model juga mendeteksi ukuran objek yaitu *width* dan *height* dari objek. Pada setiap *grid* juga diprediksi *object confidence score* dan probabilitas kelas. Nilai-nilai yang dideteksi di setiap *grid* kemudian dibandingkan dengan nilai-nilai *ground-truth* atau kotak label untuk dicari nilai *loss*-nya sehingga dapat ditemukan pola untuk memprediksi kelas objek dan *bounding box* sebagai lokasi objek. Sementara *anchor-based* memerlukan pemrogram untuk mendefinisikan *anchor box* sebelum pelatihan model. Pemrogram perlu menentukan ambang batas, ukuran, dan jumlah *anchor*. Ambang batas berupa ambang batas IoU yang didefinisikan oleh pemrogram. Ukuran akan menentukan bentuk *anchor* yang dapat berupa persegi, persegi panjang horizontal, atau persegi panjang vertikal. Jumlah *anchor* menentukan jumlah *anchor box* di setiap *grid*, baik *grid* yang memiliki objek maupun *grid* yang hanya menyimpan latar belakang. *Anchor box* bekerja pada *feature map*, kemudian model akan membandingkan *anchor box* dengan *ground-truth* berdasarkan ambang batas IoU yang



Gambar 2.13. Visualisasi Perbedaan *Anchor-based* dan *Anchor-free* [10]

digunakan. Selanjutnya *anchor box* diperbaiki berdasarkan titik pusat dan ukuran kotaknya sehingga dapat memprediksi lokasi objek lebih tepat, kotak ini disebut *bounding box*. Perbedaan *anchor-based* dengan *anchor-free* dapat divisualisasikan pada Gambar 2.13.

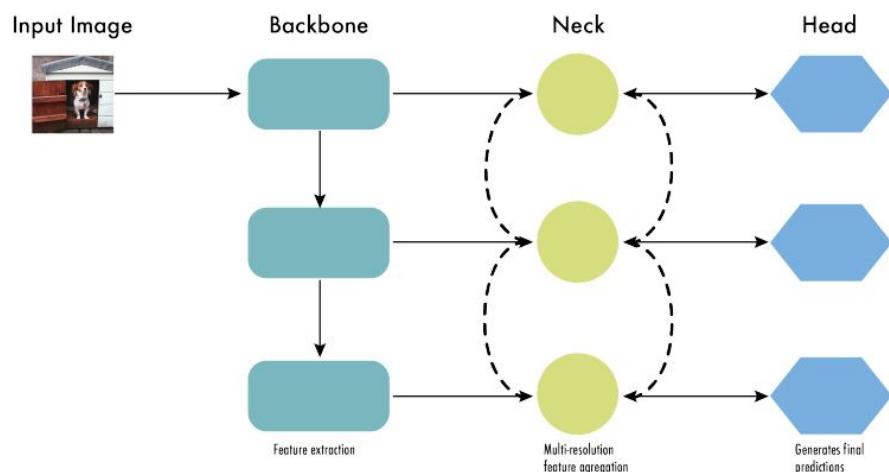
Algoritma YOLO adalah pengembangan dari metode CNN yang dikembangkan pertama kali oleh Redmon dkk [70], algoritma ini disebut YOLOv1. Model ini diperkenalkan pada Juni 2016 dengan arsitektur yang mengurangi komputasi jaringan dan waktu deteksi, tetapi memiliki mAP dua kali lipat dari detektor *real-time* lainnya. Arsitektur YOLOv1 terdiri dari 24 *convolutional layer* yang terhubung dengan 2 *fully connected layer* yang dapat memprediksi koordinat dan probabilitas *bounding box*. *Convolutional layer* tersebut menggunakan *activation function* leaky ReLU, sedangkan *fully connected layer* menggunakan fungsi aktivasi linear. *Loss function* yang digunakan adalah *sum-squared error* untuk *classification loss*, *localization loss*, dan *confidence loss*. YOLOv1 masih memiliki keterbatasan yaitu hanya bisa mendekripsi maksimal dua objek dari kelas yang sama di setiap sel *grid*. YOLOv1 juga sulit mendekripsi objek dengan rasio yang tidak terlihat di pelatihan data.

YOLOv1 dikembangkan kembali oleh Redmon dkk [71] menjadi YOLOv2 dan YOLO 9000 yang dinyatakan lebih akurat dan cepat dibanding Faster R-CNN dan SSD dengan mAP 78,6 dalam 40 FPS. YOLOv2 menggunakan *anchor box* sehingga mengatasi keterbatasan YOLOv1 dalam mendekripsi beberapa objek dalam satu sel *grid*, tapi dapat meningkatkan jumlah *bounding box* yang *overlap*. YOLOv2 dapat meningkatkan konvergensi dan mengurangi *overfitting* dengan adanya normalisasi *batch*. Normalisasi dilakukan terhadap nilai bobot input semua *convolutional layer*. YOLOv2 menggunakan *backbone* untuk ekstraksi fitur yang relevan bernama Darknet-19. Darknet-19 terdiri dari

19 *convolutional layer* dan 5 *max-pooling layer*. Model ini masih menggunakan fungsi aktivasi leaky ReLU.

Pada tahun 2018, Redmon dkk [72] memperkenalkan YOLOv3 yang memiliki akurasi mirip dengan SSD dan RetinaNet tetapi dengan kecepatan 3-4 kali lebih cepat. *Backbone* dari YOLOv3 disebut Darknet-53 yang terdiri dari 53 *convolutional layer* dan *residual connections*. *Residual connections* menambahkan lapisan konvolusi yang menghubungkan konvolusi 1 x 1 dengan konvolusi 3 x 3. Model ini menggunakan fungsi aktivasi leaky ReLU dan normalisasi *batch* sama seperti model YOLOv2. YOLOv3 juga mengaplikasikan *anchor box* dengan 3 ukuran yang berbeda yaitu 52x52x3, 26x26x3, dan 13x13x3 berjumlah 3 kotak pada setiap grid. Prediksi multi-skala ini dapat meningkatkan prediksi objek kecil secara signifikan dibanding versi-versi sebelumnya. YOLOv3 juga menambahkan blok *Spatial Pyramid Pooling* (SPP) pada *backbone* yang menggabungkan beberapa *max pooling output*. Model ini menggunakan *binary cross entropy* yang mengizinkan klasifikasi *multi-class* atau adanya beberapa label kelas dalam satu kotak.

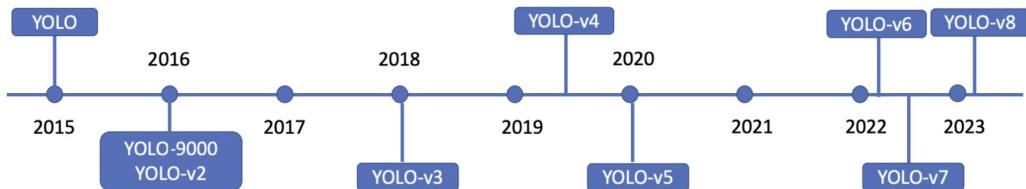
Pada tahun 2020, Bochkovskiy dkk [73] memodifikasi jaringan YOLO yang dikenal dengan nama YOLOv4 sehingga meningkatkan akurasi dan kecepatannya. Model ini memperkenalkan fungsi aktivasi mish yang dapat mengurangi *underfitting* dan menghindari perlambatan pelatihan. Arsitektur YOLOv4 terdiri dari *backbone*, *neck*, dan *head*. *Backbone* bertanggungjawab untuk ekstraksi fitur dari gambar input. *Neck* bertugas menggabungkan dan menyempurnakan fitur yang telah diekstraksi oleh *backbone*, hal ini dapat meningkatkan informasi spasial dan semantik. *Head* bertanggungjawab untuk membuat prediksi berdasarkan fitur yang diekstrak oleh *backbone* dan *neck*. Arsitektur tersebut merupakan arsitektur YOLO modern yang dapat ditunjukkan pada Gambar 2.14.



Gambar 2.14. Arsitektur Jaringan Model YOLO modern [11]

*Backbone* YOLOv4 adalah modifikasi Darknet-53 dengan *Cross-Stage Partial connections* (CSPNet). CSP dapat mengurangi perhitungan model sekaligus menjaga akurasi. *Neck* YOLOv4 adalah versi modifikasi dari SPP pada YOLOv3 dengan *Path Aggregation Network* (PANet). YOLOv4 menggunakan SPP pada YOLOv3 sebagai *head*, sehingga arsitektur YOLOv4 disebut CSPDarknet53-PANet-SPP. YOLOv4 memperkenalkan teknik augmentasi mosaic sehingga meningkatkan kemampuan deteksi citra yang bervariasi dan sulit, serta mengurangi kebutuhan untuk normalisasi *batch*. YOLOv4 juga menambahkan CIoU loss dan *Cross mini-Batch Normalization* (CmBN). CIoU loss (*Complete Intersection over Union*) mempertimbangkan jarak pusat antara kotak label dan kotak prediksi. CmBN mengakumulasikan hasil normalisasi seluruh batch, tidak seperti versi sebelumnya yang melakukan normalisasi di setiap batch. YOLOv4 menggunakan optimasi hyperparameter dengan algoritma genetik.

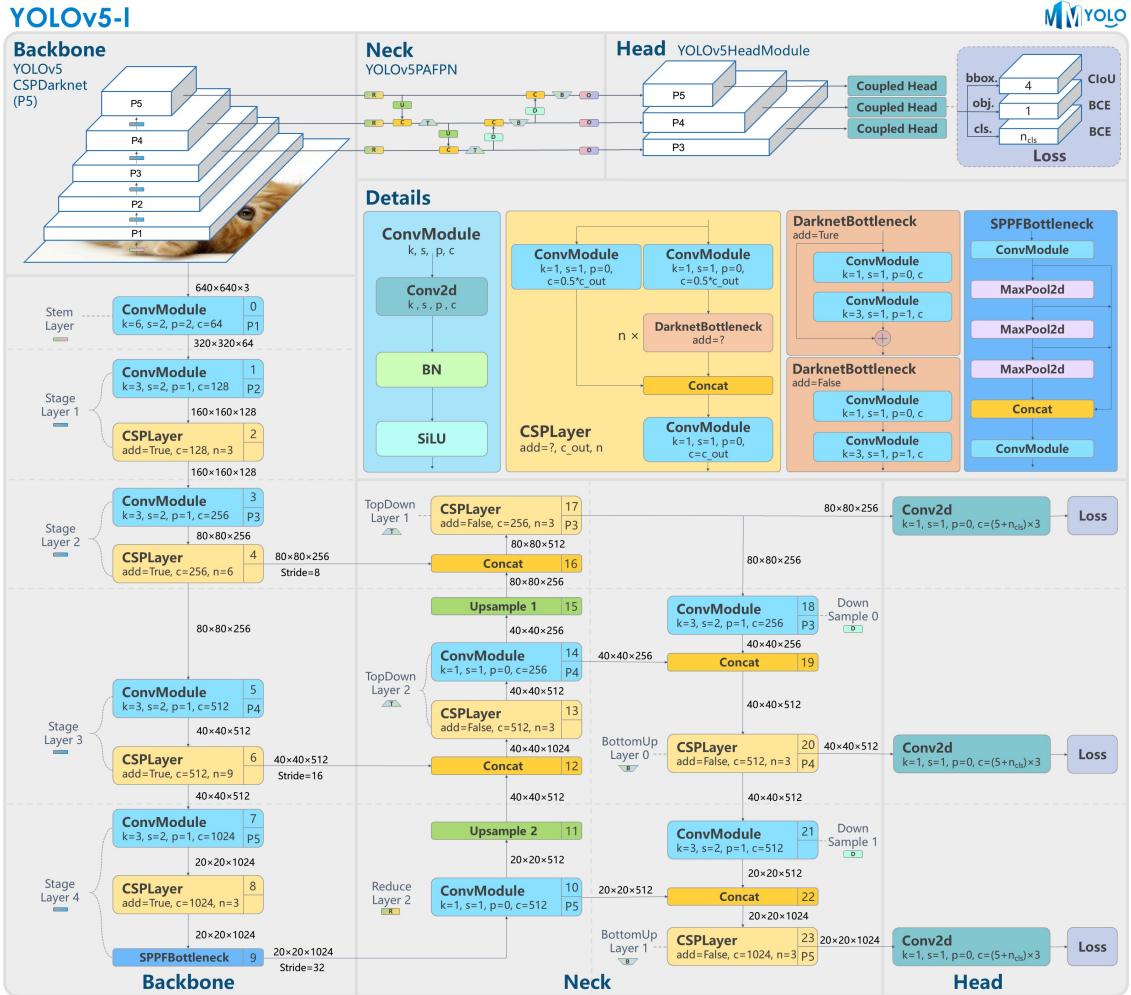
Arsitektur jaringan model modern yang dirancang di YOLOv4 selalu dikembangkan oleh para peneliti di bidang deteksi objek. YOLOv5 dirilis pada tahun 2020, tidak lama dari rilisnya YOLOv4. Selanjutnya perkembangan YOLO masih berlanjut hingga tahun 2023 dengan adanya model YOLOv6, YOLOv7, dan YOLOv8. Lini masa model YOLO secara rinci dapat ditunjukkan pada Gambar 2.15.



Gambar 2.15. Lini Masa YOLO [12]

### 2.2.8.1 YOLOv5

YOLOv5 diperkenalkan oleh Jocher, G dkk [74] sebagai *founder* dan CEO Ultralytics LLC pada tahun 2020. YOLOv5 tidak diperkenalkan melalui konferensi formal seperti model-model YOLO sebelumnya, melainkan hanya melalui *repository* github. Meskipun kontroversial, YOLOv5 memiliki akurasi dan kecepatan lebih tinggi dibanding YOLOv4 [75]. YOLOv5 memiliki arsitektur yang mirip dengan YOLOv4, tetapi terdapat perbedaan signifikan terkait fitur *autoanchor* dan *framework* yang digunakan. YOLOv5 memiliki fitur *autoanchor* untuk membangun *anchor box* sesuai dengan data [76]. Fitur ini bekerja dengan menghitung suatu metrik, perhitungan tersebut dilakukan dalam beberapa iterasi sampai menemukan *anchor box* yang baik. YOLOv5 ditulis menggunakan Python dan *framework* PyTorch, ini merupakan hal baru karena model versi sebelumnya selalu menggunakan *framework* Darknet. YOLOv5 memiliki fitur konversi format model PyTorch (.pt) ke format TorchScript, ONNX, OpenVINO, TensorRT, Core ML, dan lain-lain.



Gambar 2.16. Arsitektur Jaringan Model YOLOv5 [13]

*Backbone* pada YOLOv5 adalah CSPDarknet, *neck* jaringan YOLOv5 disebut SPPF, CSP-PAN, dan *head*-nya adalah YOLOv3 head [14]. *Backbone* CSPDarknet adalah penggabungan antara Cross Stage Partial Network (CSPNet) dengan Darknet. *Neck* YOLOv5 adalah implementasi CSP pada Path Aggregation Network dengan mengantikan SPP menjadi SPPF (SPP – Fast). Darknet adalah *backbone* pada YOLOv3, adanya CSPNet pada YOLOv5 mengurangi lapisan, parameter, harga komputasi, dan waktu pelatihan. Namun hasil deteksi YOLOv5 lebih akurat dibanding YOLOv3 karena adanya CSPNet dan fitur augmentasinya. SPPF dan CSP-PAN melakukan fusi fitur untuk dilanjutkan ke *head*, SPPF memproses data dua kali lipat lebih cepat dibanding SPP pada YOLOv3. YOLOv3 *head* adalah akhir dari proses deteksi yang menghasilkan kelas, skor, lokasi, dan ukuran berupa *coupled head*. Fungsi aktivasi yang digunakan adalah SiLU. *Loss function* yang digunakan YOLOv5 sama seperti pada model YOLOv4. *Binary Cross Entropy* untuk *classification loss* dan *objectness loss*. Sedangkan untuk *bounding box loss* menggunakan CIoU. Arsitektur jaringan YOLOv5 secara rinci dipaparkan pada Gambar 2.16.

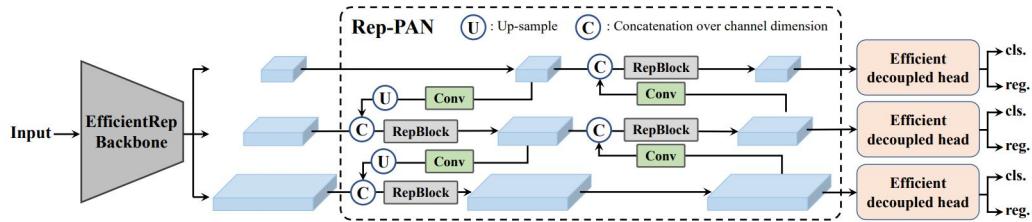
YOLOv5 memiliki 5 model, yaitu YOLOv5n, YOLOv5s, YOLOv5m, YOLOv5l, dan YOLOv5x. YOLOv5n (*nano model*) tepat digunakan untuk perangkat IoT dengan jaringan terkecil dan komputasi tercepat, tapi akurasinya paling rendah. YOLOv5s (*small model*) ideal pada mobile atau CPU, sementara YOLOv5m (*medium model*) ideal pada GPU. YOLOv5m memiliki keseimbangan yang baik antara kecepatan dan akurasi. YOLOv5l (*large model*) dan YOLOv5x (*x-large model*) ideal untuk mendekripsi objek yang kecil, kedua model tersebut ideal untuk implementasi *cloud computing*. Perbandingan kinerja model-model YOLOv5 berdasarkan kecepatan dan akurasi deteksi dapat dilihat pada Gambar 2.17.

Model	size (pixels)	mAp <sup>val</sup> 50-95	mAp <sup>val</sup> 50	Speed CPU b1 (ms)	Speed V100 b1 (ms)	Speed V100 b32 (ms)	params (M)	FLOPs @640 (B)
YOLOv5n	640	28.0	45.7	45	6.3	0.6	1.9	4.5
YOLOv5s	640	37.4	56.8	98	6.4	0.9	7.2	16.5
YOLOv5m	640	45.4	64.1	224	8.2	1.7	21.2	49.0
YOLOv5l	640	49.0	67.3	430	10.1	2.7	46.5	109.1
YOLOv5x	640	50.7	68.9	766	12.1	4.8	86.7	205.7
YOLOv5n6	1280	36.0	54.4	153	8.1	2.1	3.2	4.6
YOLOv5s6	1280	44.8	63.7	385	8.2	3.6	12.6	16.8
YOLOv5m6	1280	51.3	69.3	887	11.1	6.8	35.7	50.0
YOLOv5l6	1280	53.7	71.3	1784	15.8	10.5	76.8	111.4
YOLOv5x6 + TTA	1280 1536	55.0 55.8	72.7 72.7	3136 -	26.2 -	19.4 -	140.7 -	209.8 -

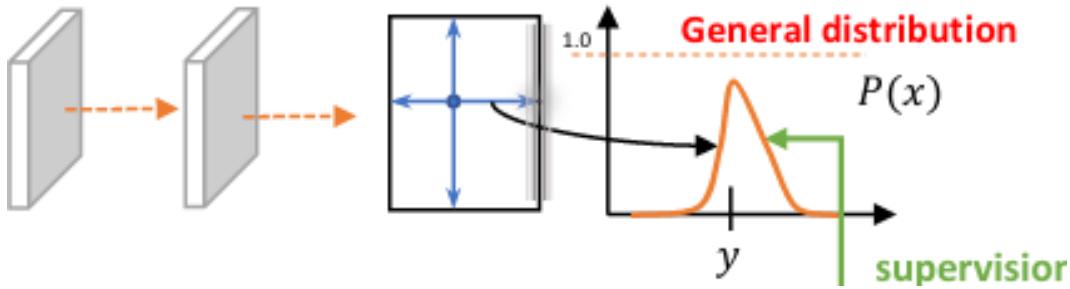
Gambar 2.17. Model-model YOLOv5 [14]

### 2.2.8.2 YOLOv6

YOLOv6 diperkenalkan oleh Li dkk [15] pada bulan Juni 2022, model tersebut dapat diakses melalui *repository* github Meituan Vision AI Department [17]. Pembaruan pada YOLOv6 terdapat pada *network design*, *label assignment*, *loss function*, dan beberapa fitur baru yang dapat meningkatkan kinerja model. *Backbone* pada YOLOv6 adalah EfficientRep berupa *backbone* VGG yang sudah dilakukan reparameterisasi untuk mengurangi latensi inferensi. *Neck* pada YOLOv6 adalah reparameterisasi dari PANet sehingga disebut Rep-PAN yang terdiri dari RepBlock dan RepConv. Untuk model YOLOv6m dan YOLOv6l, terdapat sedikit perbedaan pada arsitektur yaitu dengan reparameterisasi jaringan CSP pada Repblock sehingga disebut CSPStackRep. Kemudian *head* dari YOLOv6 adalah *efficient decoupled head*. Arsitektur pada YOLOv6n dan YOLOv6s diilustrasikan pada Gambar 2.18.



Gambar 2.18. Arsitektur Jaringan Model YOLOv6 [15]



Gambar 2.19. *Distribution Focal Loss* [16]

Beberapa fitur baru dari YOLOv6 adalah penggunaan *label assignment Task Aligned Learning* (TAL), *varifocal loss* (VFL), *Generalized/Series Intersection over Union loss* (GIoU/SIoU), *distribution focal loss* (DFL), kuantisasi, dan *self-distillation*. Strategi TAL diterapkan agar model berbasis *anchor-free*, hal ini merupakan salah satu aspek yang membedakan YOLOv6 dengan versi YOLO lainnya. Strategi TAL menggabungkan skor deteksi objek dan kualitas kotak prediksi sehingga meningkatkan akurasi model. VFL digunakan sebagai *classification loss function* sementara GIoU/SIoU sebagai *box regression loss function (box loss)*. Berdasarkan eksperimen, penulis YOLOv6 mengklaim bahwa implementasi VFL meningkatkan akurasi dibanding implementasi *focal loss* pada YOLOv7. DFL digunakan untuk fungsi kerugian deteksi objek (*dfl loss*). DFL merepresentasikan *objectness confidence score* dengan *general distribution* dimana *grid* yang memiliki titik pusat objek bernilai 1, kemudian *grid* yang semakin jauh dari titik pusat objek bernilai turun mendekati 0 secara eksponensial. Fungsi ini dapat divisualisasikan pada Gambar 2.19. DFL hanya diperkenalkan di YOLOv6-m dan YOLOv6-l karena implementasi DFL sangat mempengaruhi waktu inferensi pada model yang kecil. *Quantized model* dengan RepOptimizer memiliki akurasi terbaik dan komputasi tercepat. Strategi kuantisasi yang diterapkan pada YOLOv6 adalah PTQ dan QAT. YOLOv6 menggunakan pengetahuan distilasi berbasis *self-distillation* untuk meningkatkan akurasi. Pengetahuan distilasi adalah strategi ketika model guru digunakan untuk melatih model murid, dimana prediksi model guru dijadikan *soft label* ketika proses pelatihan model murid. *Self-distillation* artinya model murid juga bertindak sebagai model guru.

YOLOv6 memiliki 5 model yaitu, YOLOv6-n (*nano model*), YOLOv6-t (*tiny model*), YOLOv6-s (*small model*), YOLOv6-m (*medium model*), dan YOLOv6-l (*large model*).

*ge model*). YOLOv6-n, YOLOv6-t, dan YOLOv6-s ideal untuk pelatihan pada CPU, YOLOv6-m ideal untuk pelatihan pada GPU, serta YOLOv6-l ideal untuk *cloud*. Perbandingan kinerja model-model YOLOv6 berdasarkan kecepatan dan akurasi deteksi dapat dilihat pada Gambar 2.20.

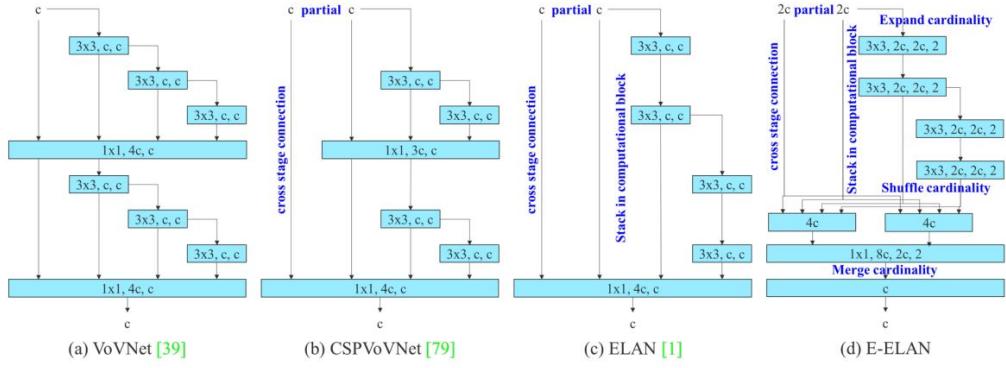
Model	Size	mAP <sub>val</sub> 0.5:0.95	Speed <sup>T4</sup> trt fp16 b1 (fps)	Speed <sup>T4</sup> trt fp16 b32 (fps)	Params (M)	FLOPs (G)
YOLOv6-N	640	37.5	779	1187	4.7	11.4
YOLOv6-S	640	45.0	339	484	18.5	45.3
YOLOv6-M	640	50.0	175	226	34.9	85.8
YOLOv6-L	640	52.8	98	116	59.6	150.7
YOLOv6-N6	1280	44.9	228	281	10.4	49.8
YOLOv6-S6	1280	50.3	98	108	41.4	198.0
YOLOv6-M6	1280	55.2	47	55	79.6	379.5
YOLOv6-L6	1280	57.2	26	29	140.4	673.4

Gambar 2.20. Model-model YOLOv6 [17]

### 2.2.8.3 YOLOv7

YOLOv7 dipublikasikan pada bulan Juli 2022 oleh penulis model YOLOv4, [18]. Model tersebut dapat diakses melalui *repository* github WongKinYiu [19]. YOLOv7 mengimplementasikan fitur *autoanchor* yang terdapat pada model YOLOv5 dan perluasan lapisan jaringan yang efisien. Pembaruan pada YOLOv7 terletak pada arsitektur, disertai dengan fitur *trainable Bag-of-Freebies*. *Backbone* pada YOLOv7 disebut *Extended-Efficient Layer Aggregation Networks* (E-ELAN). *Backbone* E-ELAN dapat meningkatkan kemampuan pembelajaran jaringan dengan memperluas, mengacak, dan menggabungkan kardinalitas tanpa merusak *gradient*. E-ELAN merupakan perluasan dari jaringan ELAN yang dirancang dapat mengendalikan *gradient* agar jaringan lebih dalam, sehingga dapat mempelajari dan menyatu secara efektif. Jaringan ELAN berawal dari pengembangan VoVNet ke CSPVoVNet yang dirancang dengan analisis *gradient* untuk mengaktifkan bobot pada lapisan yang berbeda, sehingga dapat mempelajari fitur yang lebih beragam. *Neck* dari YOLOv7 adalah PANet serta *head*-nya menggunakan *lead head* dan *auxiliary head*. Arsitektur YOLOv7 secara rinci dipaparkan pada Gambar 2.21.

YOLOv7 juga merancang *compound model scaling* yaitu penskalaan model berbasis concatenate dengan penskalaan lebar dan kedalaman. Fitur ini diajukan agar model dapat menyesuaikan penerapan pada perangkat yang berbeda, tetapi dengan *trade-off*



Gambar 2.21. Arsitektur Jaringan Model YOLOv7 [18]

yang baik antara akurasi dan kecepatan. YOLOv7-tiny dengan parameter berjumlah sekitar 6 juta ideal untuk edge GPU, YOLOv7 ideal untuk GPU normal dengan parameter berjumlah sekitar 37 juta, sedangkan model lainnya memiliki parameter yang lebih banyak. Perbandingan kinerja model-model YOLOv7 berdasarkan *trade-off* antara akurasi dan kecepatan dapat dilihat pada Gambar 2.22.

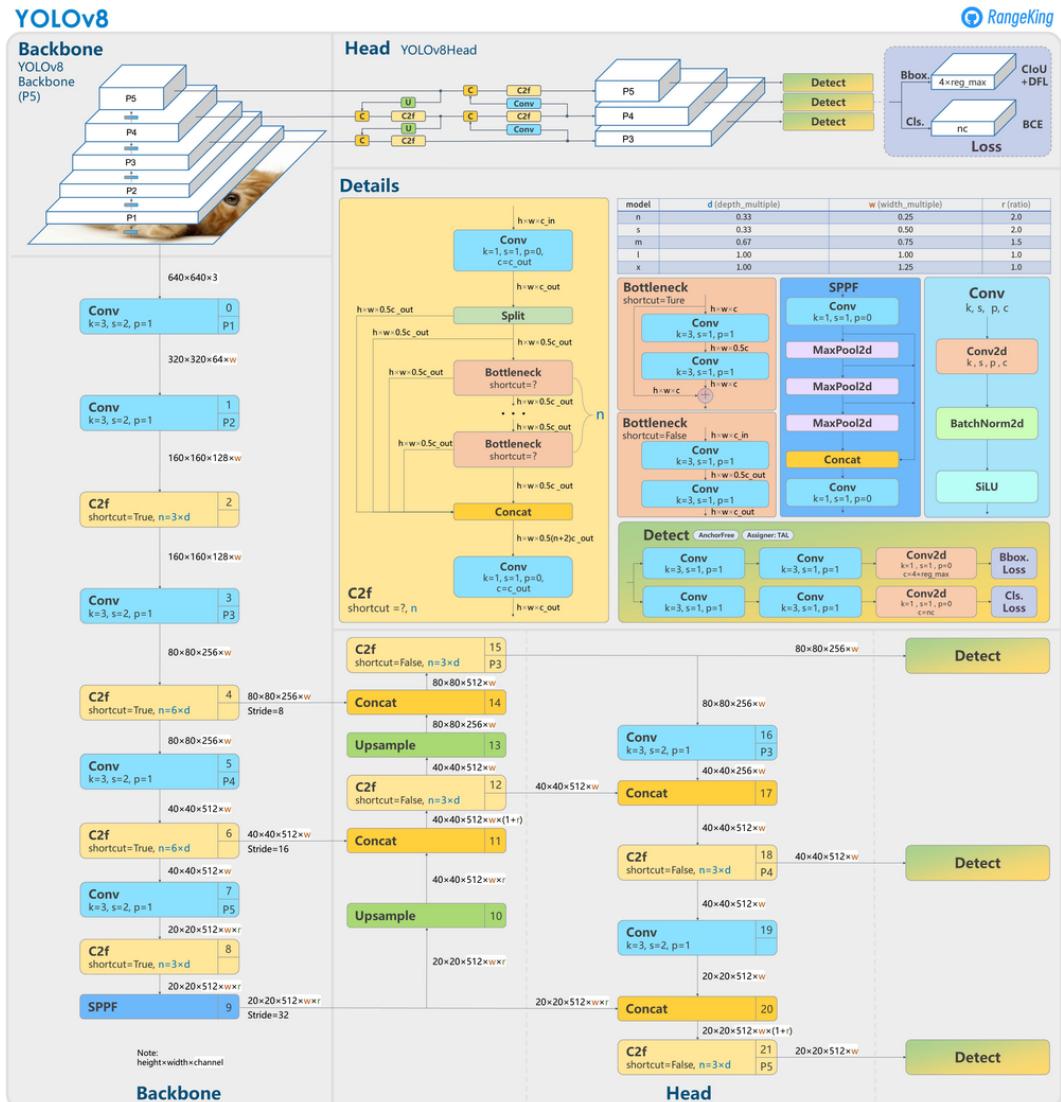
Model	Test Size	$AP^{test}$	$AP_{50}^{test}$	$AP_{75}^{test}$	batch 1 fps	batch 32 average time
YOLOv7	640	51.4%	69.7%	55.9%	161 fps	2.8 ms
YOLOv7-X	640	53.1%	71.2%	57.8%	114 fps	4.3 ms
YOLOv7-W6	1280	54.9%	72.6%	60.1%	84 fps	7.6 ms
YOLOv7-E6	1280	56.0%	73.5%	61.2%	56 fps	12.3 ms
YOLOv7-D6	1280	56.6%	74.0%	61.8%	44 fps	15.0 ms
YOLOv7-E6E	1280	56.8%	74.4%	62.1%	36 fps	18.7 ms

Gambar 2.22. Model-model YOLOv7 [19]

Metode baru yang disediakan pada YOLOv7 yaitu *Bag-of-Freebies* (BoF) bertujuan untuk meningkatkan kinerja tanpa meningkatkan harga komputasi. Metode BoF fokus pada teknik reparameterisasi dan *double head* pada YOLOv7. Teknik reparameterisasi adalah teknik peningkatan kinerja model setelah pelatihan. YOLOv7 menggunakan teknik reparameterisasi *module level* yaitu RepConv tanpa koneksi identitas yang disebut RepConvN. RepConv adalah teknik reparameterisasi yang menggabungkan konvolu-

si 3x3, konvolusi 1x1, dan koneksi identitas. Namun koneksi identitas pada RepConv di beberapa arsitektur menghancurkan *residual* dan *concatenation* sehingga menyebabkan peningkatan *gradient*, maka YOLOv7 merancang RepConvN dalam menggantikan lapisan konvolusi dengan *residual* atau *concatenate*. *Double head* pada YOLOv7 adalah *lead head* dan *auxiliary head*. *Lead head* bertanggungjawab terhadap hasil akhir deteksi, sedangkan *auxiliary head* bertanggungjawab untuk membantu proses pelatihan di lapisan tengah. Hasil akhir dari *lead head* dijadikan sebagai *soft label*, label tersebut digunakan untuk melatih model dan menghitung *loss* pada *lead head* maupun *auxiliary head*. Bobot pada *auxiliary head* selalu diperbarui selama membantu proses pelatihan.

## 2.2.8.4 YOLOv8



Gambar 2.23. Arsitektur Jaringan Model YOLOv8 [20]

YOLOv8 dirilis pada tanggal 10 Januari 2023 oleh pengembang model YOLOv5, Ultralytics [21]. Model YOLOv8 memiliki beberapa perbedaan dibanding YOLOv5,

diantaranya model berbasis *anchor-free*. Cara kerja *anchor-free* pada model YOLOv8 sama dengan YOLOv6 yang menggunakan TAL, *decoupled head*, dan menghapus *objectness*. Deteksi *anchor-free* mengurangi jumlah kotak prediksi sehingga mempercepat waktu NMS setelah proses inferensi [77]. Pada model YOLOv8 juga menggunakan fungsi CIoU loss, DFL loss, dan *Binary cross entropy*. Namun YOLOv8 memiliki *head* sendiri yang disebut Split Ultralytics Head. Arsitektur dari YOLOv8 adalah modifikasi dari arsitektur YOLOv5 yaitu perubahan modul CSPLayer menjadi modul c2f, mengganti konvolusi 6x6 menjadi 3x3, menghapus beberapa lapisan konvolusi, dan mengganti konvolusi 1x1 dengan konvolusi 3x3 pada Bottleneck. Modul c2f berisi lapisan konvolusi, lapisan *split*, *concatenation*, dan Bottleneck. Pada *backbone* YOLOv8 juga menggunakan SPPF seperti pada YOLOv5 dengan beberapa perubahan. Arsitektur YOLOv8 dapat diilustrasikan pada Gambar 2.23.

YOLOv8 memiliki keunggulan dalam penggunaannya, YOLOv8 bisa diakses melalui *repository* github [78] atau menggunakan PIP *package* sehingga lebih *user-friendly* dibanding model YOLO sebelumnya. YOLOv8 juga menawarkan kemudahan ekspor ke format TorchScript, ONNX, OpenVINO, TensorRT, CoreML, dan lain-lain. Kemampuan ini sama seperti yang ditawarkan YOLOv5, hal ini merupakan keunggulan versi YOLO keluaran Ultralytics. YOLOv8 memiliki 5 model sama seperti YOLOv5, yaitu YOLOv8n, YOLOv8s, YOLOv8m, YOLOv8l, dan YOLOv8x. Perbandingan kinerja dari kelima model tersebut ditunjukkan pada Gambar 2.24.

Model	size (pixels)	mAP <sub>val</sub> 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

Gambar 2.24. Model-model YOLOv8 [21]

### 2.3 Analisis Perbandingan Metode

Berdasarkan tinjauan pustaka pada subbab 2.1 dapat diketahui bahwa belum ada identifikasi penyakit tanaman kentang yang disertai lokalisasi. Penelitian yang sudah ada hanya mampu melakukan klasifikasi, padahal proses lokalisasi dapat meningkatkan akurasi identifikasi objek seperti yang sudah dijelaskan pada subbab 2.2.2. Estimasi lokasi juga dapat mempercepat penanganan penyakit kentang karena mempermudah petani menemukan lokasi infeksi penyakit. Permasalahan ini dapat diselesaikan dengan adanya penelitian di bidang deteksi objek. Deteksi objek berbasis *deep learning* dinyatakan memiliki kinerja lebih baik dibanding metode tradisional, tetapi membutuhkan data,

memori, perangkat, dan waktu komputasi yang tinggi. Berdasarkan *trade-off* tersebut, peneliti di bidang deteksi objek selalu mengembangkan model deteksi objek berbasis *deep learning* agar lebih optimal. Model deteksi objek berbasis *deep learning* digolongkan menjadi dua, yaitu *two-stage detector* dan *one-stage detector*. *Two-stage detector* dikembangkan sejak tahun 2014. Model dengan teknik ini memiliki akurasi yang baik, tetapi memiliki waktu komputasi yang lambat dan kompleksitas model yang tinggi. Masalah tersebut menarik para peneliti bidang deteksi objek untuk menciptakan model yang lebih cepat dengan mengurangi kompleksitas model. Model ini berbasis *one-stage detector* yang mulai diperkenalkan pada tahun 2016.

Model *one-stage detector* pertama adalah model *You Only Look Once* (YOLO) artinya model deteksi hanya melihat citra yang dideteksi dalam satu kali jaringan tunggal. Model YOLO pertama yang disebut YOLOv1 membutuhkan waktu komputasi lebih cepat dan memori lebih kecil dibanding model *two-stage detector*. Namun di awal perkembangannya, akurasi YOLO masih rendah dibanding model *two-stage detector*. Kemudian model YOLO selalu diteliti dan dikembangkan hingga adanya model-model versi modern dari YOLO. Model YOLO versi modern ini berhasil meningkatkan akurasinya, seperti YOLOv5, YOLOv6, YOLOv7, dan YOLOv8. Model-model tersebut memiliki arsitektur dan prinsip kerja yang berbeda. Hal ini menyebabkan setiap model memiliki akurasi deteksi, kompleksitas model, dan waktu komputasi yang berbeda pada implementasinya. Dengan demikian, pada penelitian ini akan diuji model-model YOLO modern untuk mendeteksi penyakit yang menginfeksi tanaman kentang melalui citra daunnya.

Tabel 2.2. Perbandingan Model YOLOv5m, YOLOv6m, YOLOv7, dan YOLOv8m

Pembeda	YOLOv5m	YOLOv6m	YOLOv7	YOLOv8m
<i>Backbone</i>	CSPDarknet	EfficientRep	E-ELAN	Darknet
<i>Neck</i>	SPPF, CSP-PAN	CSPStackRep	PAN	SPPF, PAN
<i>Head</i>	YOLOv3	<i>decoupled head</i>	<i>double head</i>	<i>split head</i>
<i>Anchor</i>	Ya	Tidak	Ya	Tidak
<i>Loss</i>	<i>box, obj, cls</i>	<i>iou, dfl, cls</i>	<i>box, obj, cls</i>	<i>box, dfl, cls</i>
mAP@0.5	64,1	-	69,7	-
mAP@0.5:0.95	45,4	50,0	-	50,2
Parameter	21,2 juta	34,9 juta	36,9 juta	25,9 juta
Waktu Deteksi	1,7 ms	226 FPS	161 FPS	1,83 ms

Daun kentang bukan merupakan objek yang kecil, sehingga dapat digunakan ukuran citra pelatihan *default* model yaitu ukuran 640 x 640 (dalam piksel). Perangkat yang digunakan pada penelitian adalah GPU T4, maka analisis kinerja dilakukan terhadap model yang ideal diimplementasikan pada GPU. Model yang ideal diimplementasikan pada GPU adalah model YOLOv5m, YOLOv6m, YOLOv7, dan YOLOv8m.

Perbandingan arsitektur, prinsip kerja, dan kinerja keempat model berdasarkan dokumentasi pengujian pengembang dapat dirangkum pada Tabel 2.2. Kinerja model YOLO pada penelitian ini dianalisis berdasarkan akurasi deteksi, kompleksitas model, dan waktu komputasi karena mengadaptasi beberapa tinjauan pustaka yang melakukan analisis kinerja model. Analisis kinerja model dilakukan pada penelitian KC dkk [44] dan Rashid dkk [38] berdasarkan akurasi klasifikasi dan kompleksitas model, Kothari dkk [40] berdasarkan akurasi klasifikasi dan waktu pelatihan, serta Dang dkk [47] berdasarkan akurasi deteksi, kompleksitas model, dan waktu inferensi. Selain itu, indikator akurasi deteksi, kompleksitas model, dan waktu komputasi adalah hal yang sangat penting dalam identifikasi penyakit tanaman kentang. Akurasi deteksi dianalisis berdasarkan metrik mAP@0.5 dan mAP@0.5:0.95. Kompleksitas model dibandingkan berdasarkan jumlah parameter. Waktu komputasi dianalisis berdasarkan waktu pelatihan dan waktu deteksi. Dengan adanya penelitian ini diharapkan dapat menemukan model dengan komputasi yang hemat tetapi memiliki akurasi yang tinggi dalam mendeteksi penyakit tanaman kentang.

## 2.4 Pertanyaan Tugas Akhir

1. Bagaimana akurasi deteksi model YOLOv5m, YOLOv6m, YOLOv7, dan YOLOv8m dalam mendeteksi penyakit *Early Blight* dan *Late Blight* pada tanaman kentang?
2. Bagaimana kompleksitas model YOLOv5m, YOLOv6m, YOLOv7, dan YOLOv8m dalam mendeteksi penyakit *Early Blight* dan *Late Blight* pada tanaman kentang?
3. Bagaimana waktu komputasi model YOLOv5m, YOLOv6m, YOLOv7, dan YOLOv8m dalam mendeteksi penyakit *Early Blight* dan *Late Blight* pada tanaman kentang?
4. Model manakah di antara model YOLOv5m, YOLOv6m, YOLOv7, dan YOLOv8m yang memiliki kinerja terbaik dalam mendeteksi penyakit *Early Blight* dan *Late Blight* pada tanaman kentang?

## BAB III

### METODE PENELITIAN

#### 3.1 Alat dan Bahan Tugas akhir

##### 3.1.1 Alat Tugas akhir

Alat - alat yang digunakan pada tugas akhir ini berupa perangkat keras maupun perangkat lunak sebagai sarana pendukung antara lain:

1. Pada tugas akhir ini digunakan laptop dengan spesifikasi sistem operasi Windows 10 64 bit, prosesor Intel Core i5 1035G1 CPU @1.00 GHz, da RAM 8 GB. Spesifikasi minimum perangkat dipengaruhi oleh model detektor yang diimplementasikan. Pada penelitian ini menggunakan model detektor yang membutuhkan prosesor dengan spesifikasi minimum GPU.
2. Adanya keterbatasan prosesor perangkat keras yang digunakan, sehingga tugas akhir ini menggunakan perangkat lunak Google Colaboratory dengan *hardware accelerator* GPU T4. Perangkat tersebut digunakan untuk menjalankan program model deteksi objek dengan *runtime* Python 3. GPU pada *runtime* yang digunakan memiliki limitasi kapasitas RAM yaitu 15 gigabyte.
3. Tugas akhir ini menggunakan perangkat lunak MakeSense sebagai *annotation tool*. MakeSense diakses melalui makesense.ai menggunakan *web browser* Google Chrome. Anotasi data diekspor dengan format YOLO.
4. Penyimpanan dataset dan model deteksi objek menggunakan Google Drive.

##### 3.1.2 Bahan Tugas akhir

Penelitian ini menggunakan dataset dari 3 sumber yaitu PlantVillage Dataset [79], Potato Leaf Dataset [80], dan Potato Disease Leaf Dataset [81]. Ketiga dataset dipublikasikan di situs Kaggle masing-masing pada tahun 2018, 2020, dan 2021. Dari ketiga



Gambar 3.1. Contoh Dataset *Healthy*, *Early Blight*, *Late Blight*

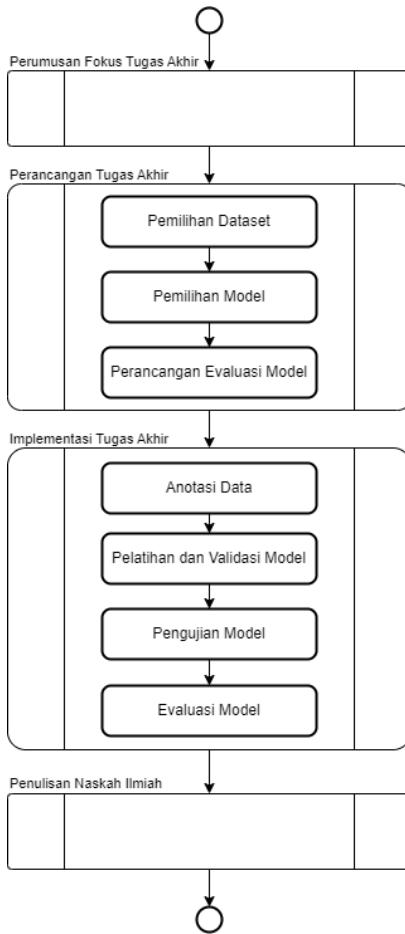
dataset diambil 3000 citra daun kentang yang terdiri dari 1000 citra *healthy* (sehat), 1000 citra *early blight* (bercak kering), dan 1000 citra *late blight* (hawar daun). Contoh citra *healthy*, citra *early blight*, dan citra *late blight* ditunjukkan pada Gambar 3.1. Pengambilan dataset dari ketiga sumber bertujuan menghindari *imbalanced class* karena terdapat perbedaan jumlah data antar kelas yang signifikan pada PlantVillage Dataset dan Potato Disease Leaf Dataset. Sementara Potato Leaf Dataset memiliki jumlah data yang seimbang tiap kelasnya, tetapi hanya memiliki 500 citra pada setiap kelas. Jumlah ini masih sedikit jika digunakan pada pelatihan. Selain itu, ketiga sumber dataset memiliki perbedaan *background* dan diambil pada tahun yang berbeda. Hal ini akan meningkatkan kemampuan model apabila dilatih dan diuji dengan data yang variatif. Sampel dataset dibagi secara acak dengan persentase 70% data pelatihan, 20% data validasi, dan 10% data pengujian.

### 3.2 Metode yang Digunakan

Penelitian ini termasuk penelitian kuantitatif yang dilakukan dengan eksperimen. Eksperimen diawali dengan melakukan pelatihan terhadap 4 model deteksi objek YOLO, yaitu YOLOv5m, YOLOv6m, YOLOv7, dan YOLOv8m. Pelatihan dilakukan secara iterasi dengan menambah jumlah epoch hingga mencapai akurasi deteksi yang baik. Eksperimen dilanjutkan dengan melakukan pengujian terhadap kinerja model dalam mendekripsi penyakit tanaman kentang menggunakan citra daun. Kinerja model yang dibandingkan adalah akurasi deteksi, kompleksitas model, dan waktu komputasi. Akurasi deteksi dianalisis berdasarkan metrik mAP@0.5 dan mAP@0.5:0.95. Kompleksitas model dibandingkan berdasarkan jumlah parameter. Waktu komputasi dianalisis berdasarkan waktu pelatihan dan waktu deteksi. Berdasarkan pengujian tersebut dapat dianalisis dan dievaluasi kinerja model sehingga diketahui kelebihan dan kekurangan model. Dengan mengetahui kelebihan dan kekurangan model dapat diketahui model terbaik untuk mendekripsi penyakit tanaman kentang. Dalam melakukan eksperimen, peneliti menggunakan Python sebagai bahasa pemrograman yang dijalankan pada *runtime* Google Colaboratory.

### 3.3 Alur Tugas Akhir

Tugas akhir ini dimulai dengan merumuskan fokus dari tugas akhir. Berdasarkan fokus yang sudah dirumuskan, dilanjutkan perancangan jalannya penelitian pada tugas akhir ini. Tahap selanjutnya adalah pelaksanaan rancangan tugas akhir, tahap ini disebut implementasi tugas akhir. Kemudian tugas akhir ini diakhiri dengan tahap penulisan naskah ilmiah. Alur penelitian pada tugas akhir ini dapat dilihat pada Gambar 3.2 dan dijelaskan secara rinci pada subbab selanjutnya.



Gambar 3.2. Alur Tugas Akhir

### 3.3.1 Perumusan Fokus Tugas Akhir

Pada tahapan ini, penulis merumuskan fokus penelitian untuk mengarahkan jalannya penelitian. Perumusan fokus penelitian dilandasi dengan bidang studi peneliti disertai studi literatur. Sebelum menetapkan fokus penelitian, ditentukan terlebih dahulu topik penelitian. Topik dari penelitian ini adalah analisis kinerja model YOLO untuk deteksi objek. Analisis dilakukan dengan memperhatikan prinsip kerja, kelebihan, dan kekurangan model YOLO. Topik ini diambil karena terdapat banyak model YOLO yang diimplementasikan untuk deteksi objek. Namun belum banyak penelitian yang membandingkan dan menganalisis kinerja model-model tersebut. Padahal kinerja model deteksi objek YOLO perlu dipertimbangkan sebelum melakukan implementasi.

Penelitian terkait kinerja YOLO ini diharapkan dapat membantu sektor pertanian yang berperan penting bagi kehidupan masyarakat. Setelah menggali informasi dari sudut pandang pertanian maupun ekonomi, diketahui bahwa tanaman kentang memiliki kontribusi tinggi dalam penyediaan pangan dan perekonomian masyarakat. Selanjutnya dilakukan identifikasi faktor-faktor yang menghambat produktivitas tanaman kentang. Salah satu faktor yang menghambat produktivitas tanaman kentang adalah penyakit ta-

naman. Penyakit tanaman menjadi permasalahan yang dipelajari dalam penelitian ini. Penyakit yang sering menyerang tanaman kentang dan memiliki dampak besar adalah penyakit *early blight* (bercak kering) dan *late blight* (hawar daun). Gejala penyakit tersebut dapat dideteksi dengan munculnya pola tertentu pada daun tanaman kentang. Maka daun tanaman kentang digunakan sebagai objek dalam penelitian untuk mendeteksi penyakit tanaman kentang. Dengan demikian, fokus penelitian ini adalah analisis kinerja YOLO untuk deteksi objek pada penyakit tanaman kentang.

### 3.3.2 Perancangan Tugas Akhir

Setelah menetapkan fokus penelitian, peneliti melakukan tinjauan pustaka untuk mempelajari dan memahami penelitian terdahulu. Tinjauan pustaka dilakukan terhadap penelitian terdahulu yang fokus penelitiannya hampir sama dengan fokus penelitian ini. Penelitian terdahulu dapat dijadikan referensi dalam perancangan penelitian meliputi dataset, model, serta indikator evaluasi kinerja model yang digunakan dan dianalisis.

#### 3.3.2.1 Pemilihan Dataset

Berdasarkan tinjauan pustaka, beberapa penelitian melakukan implementasi model klasifikasi penyakit tanaman kentang. Penyakit tanaman kentang dapat diklasifikasikan berdasarkan pola daun kentang, maka penelitian ini menggunakan dataset daun kentang yang dipublikasikan di Kaggle. Dataset tersebut terdiri dari daun *healthy* (sehat), *early blight* (bercak kering), dan *late blight* (hawar daun). Proporsi pembagian data dalam implementasi mempertimbangkan beberapa referensi dari tinjauan pustaka, seperti yang sudah dijelaskan pada subbab 3.1.2. Pembagian data dipilih secara acak menjadi data pelatihan, data validasi, dan data pengujian. Data pelatihan digunakan untuk melatih model, jika semakin banyak jumlahnya maka model akan semakin baik. Data validasi digunakan untuk memvalidasi model agar tidak terjadi model yang *overfitting* terhadap data pelatihan. Data pengujian digunakan untuk menguji model, sebagai simulasi penggunaan model di dunia nyata. Jumlah hasil pembagian dataset dapat dirinci pada Tabel 3.1.

Tabel 3.1. Pembagian Dataset

Kelas	Data Pelatihan	Data Validasi	Data Pengujian
<i>Healthy</i>	700	200	100
<i>Early Blight</i>	700	200	100
<i>Late Blight</i>	700	200	100

### **3.3.2.2 Pemilihan Model**

Hasil dari tinjauan pustaka menemukan bahwa beberapa penelitian menerapkan model klasifikasi objek untuk klasifikasi penyakit tanaman kentang. Model klasifikasi yang baik adalah model klasifikasi berbasis *deep learning*. Sedangkan implementasi model deteksi objek untuk deteksi penyakit tanaman kentang belum ditemukan dalam tinjauan pustaka, maka penelitian terkait deteksi objek pada penyakit tanaman kentang masih dibutuhkan. Hal ini bertujuan untuk meningkatkan akurasi model. Kemudian dilanjutkan tinjauan pustaka terkait model-model deteksi objek. Tinjauan pustaka menunjukkan model deteksi objek berbasis *deep learning* yang banyak digunakan dan unggul adalah model YOLO, tetapi model ini belum diimplementasikan untuk deteksi penyakit tanaman kentang. Selanjutnya diketahui bahwa YOLO memiliki banyak model detektor seperti YOLOv5, YOLOv6, YOLOv7, dan YOLOv8. Model yang dipilih dalam penelitian ini adalah YOLOv5m, YOLOv6m, YOLOv7, dan YOLOv8m karena pengujian dilakukan dengan *runtime* GPU. Dari beberapa model tersebut, perlu dianalisis kinerjanya untuk menemukan model YOLO terbaik. Analisis dilakukan dengan memperhatikan prinsip kerja, kelebihan, dan kekurangan.

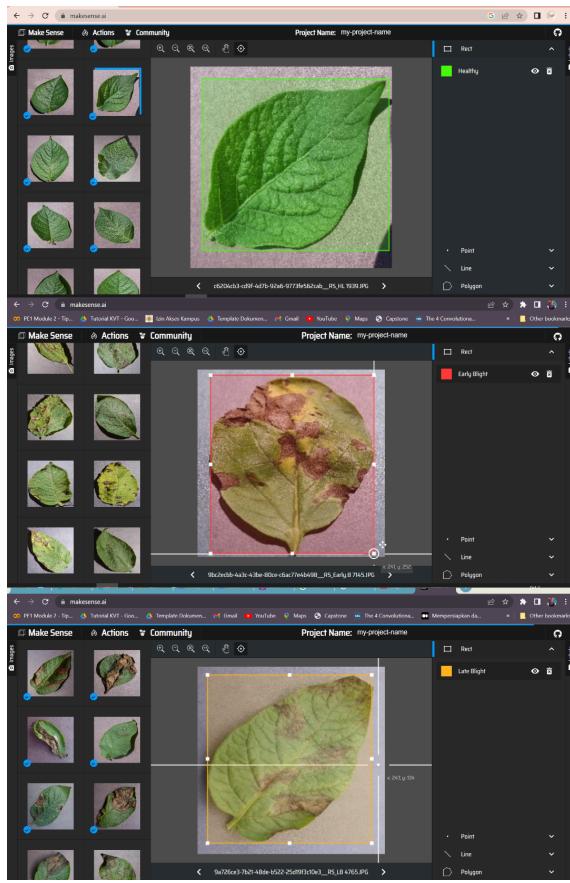
### **3.3.2.3 Perancangan Evaluasi Model**

Berdasarkan tinjauan pustaka dan dasar teori yang ada, diketahui bahwa model deteksi objek yang baik adalah model dengan akurasi deteksi, kompleksitas model, dan waktu komputasi yang sesuai untuk kebutuhan implementasinya. Akurasi dapat diukur berdasarkan metrik mAP@0.5 dan mAP@0.5:0.95. Kemudian jumlah parameter adalah salah satu indikator yang dapat mengukur kompleksitas model. Waktu komputasi dapat dibagi menjadi waktu pelatihan dan waktu deteksi. Waktu pelatihan diukur dalam satuan waktu jam, sedangkan waktu deteksi diukur dalam satuan waktu ms. Waktu pelatihan adalah waktu yang diperlukan untuk melatih dan memvalidasi model. Waktu deteksi adalah rata-rata waktu yang diperlukan model dalam keseluruhan proses deteksi objek pada tiap citra. Proses deteksi objek terdiri dari pra-inferensi, inferensi, dan pasca-inferensi, maka waktu deteksi diukur berdasarkan total dari waktu pra-inferensi, waktu inferensi, dan waktu pasca-inferensi. Beberapa indikator tersebut dapat diukur ketika model diterapkan. Oleh karena itu, penelitian ini akan melakukan uji coba model dalam mendeteksi penyakit tanaman kentang.

## **3.3.3 Implementasi Tugas Akhir**

### **3.3.3.1 Anotasi Data**

Anotasi data adalah proses pemberian label untuk setiap objek pada citra. Label terdiri dari kelas dan *bounding box* objek. Dalam penelitian ini, objek diklasifikasikan menjadi 3 kelas yaitu kelas daun tanaman *healthy* (sehat), *early blight* (bercak kering),



Gambar 3.3. Proses Anotasi Data

dan *late blight* (hawar daun). *Bounding box* menandai label koordinat x dan y sebagai titik pusat objek, serta lebar dan tinggi *bounding box*. *Bounding box* pada label disebut *ground truth box*. Proses pelabelan pada penelitian ini menggunakan *tool* aplikasi website makesense.ai, kemudian diekspor ke format YOLO. Proses anotasi dapat digambarkan pada Gambar 3.3.

### 3.3.3.2 Pelatihan dan Validasi Model

Model dilatih dengan 2100 citra tanpa *image enhancement*. Data pelatihan berisi citra dan label langsung digunakan untuk melatih model. Pelatihan model menggunakan Google Colaboratory dengan bahasa pemrograman python. Pemrosesannya menggunakan *runtime* dengan *hardware accelerator* GPU T4. Ukuran citra yang digunakan dalam proses pelatihan adalah 640x640 piksel. Pelatihan model menggunakan optimisasi *Stochastic Gradient Descent* (SGD) dengan hyperparameter batch size 16. Batch size menentukan jumlah sampel data yang melewati jaringan dalam satu waktu sebelum memperbarui parameter, umumnya bernilai perpangkatan dari angka 2. Semakin besar batch size menghasilkan normalisasi batch yang semakin baik, tetapi perlu mempertimbangkan kemampuan perangkat penelitian dan waktu komputasi. Penelitian ini pernah menerapkan pelatihan model menggunakan batch size 32, tetapi proses pelatihan terkendala pada

beberapa model yang memiliki jumlah parameter tinggi seperti model YOLOv7 dan YOLOv6m. Pelatihan menggunakan batch size 32 pada kedua model melebihi kapasitas RAM dari perangkat penelitian, sehingga dipilih batch size dengan nilai *default* 16.

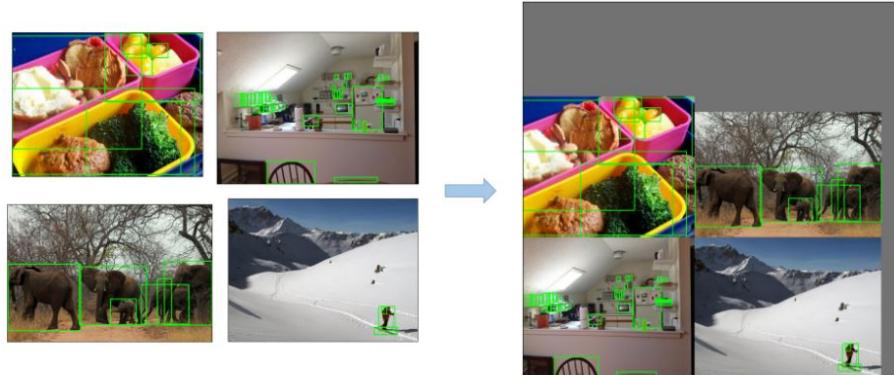
Pelatihan model dikonfigurasi dengan nilai *default* hyperparameter dari model, kecuali nilai epoch. Epoch yang digunakan untuk pelatihan model adalah 50. 1 epoch dihitung ketika seluruh dataset sudah melalui proses pelatihan pada jaringan. Jumlah epoch yang terlalu rendah akan menimbulkan *underfitting*, sementara jumlah epoch yang terlalu tinggi akan menimbulkan *overfitting*. Untuk menghindari hal tersebut maka digunakan data validasi yang berjumlah 600 citra tanpa *image enhancement*. Konfigurasi hyperparameter pada pelatihan model secara rinci dapat ditunjukkan pada Tabel 3.2.

Tabel 3.2. Konfigurasi Hyperparameter Pelatihan Model

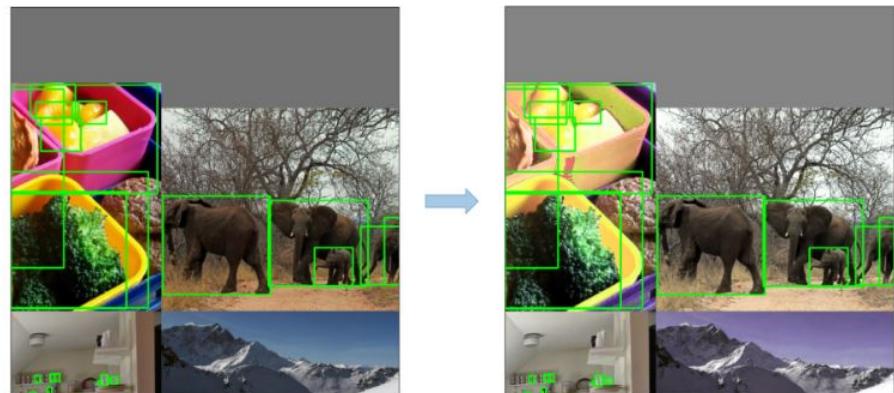
Hyperparameter	Nilai
epoch	50
batch size	16
image size	640 x 640
optimizer	SGD
learning rate	0.01
scheduler	cosine
momentum	0.937
weight decay	0.0005
hsv	0.015, 0.7, 0.4
translate	0.1
scale	0.5
fliplr	0.5
flipud	0.0
mosaic	1.0

Pada pelatihan model YOLO dikonfigurasi *default* menggunakan beberapa teknik augmentasi seperti mosaic, translate, scale, flip, hsv. Mosaic adalah teknik mengkombinasikan beberapa gambar seperti pada Gambar 3.4. Translate adalah teknik augmentasi dengan pergeseran gambar. Scale adalah teknik augmentasi dengan perubahan ukuran gambar pada skala tertentu, dapat berupa pengecilan atau pembesaran gambar. Flip adalah pembalikan gambar yang dapat dilakukan secara vertikal dan horizontal. HSV adalah teknik augmentasi menggunakan pengaturan nilai hue, saturation, dan value. Hue adalah gradasi dari ketiga warna dasar yaitu merah, kuning, dan biru. Saturation adalah intensitas kemurnian warna, dimana semakin tinggi nilai saturasi maka warnanya semakin murni. Value adalah nilai yang menunjukkan kecerahan gambar. Warna paling tinggi nilai value-nya adalah warna putih, sebaliknya warna paling rendah nilai value-nya adalah

warna hitam. Teknik augmentasi HSV akan mengubah warna gambar asli seperti pada Gambar 3.5.



Gambar 3.4. Augmentasi Mosaic [21]



Gambar 3.5. Augmentasi HSV [21]

### 3.3.3.3 Pengujian Model

Model yang telah melalui proses pelatihan dan validasi akan diuji menggunakan data pengujian. Tujuan proses pengujian adalah untuk menguji dan mengetahui kinerja model dalam mendekripsi data yang belum digunakan untuk pelatihan maupun validasi. Penelitian ini menggunakan 300 citra untuk menguji model. Pengujian model dilakukan dengan membandingkan mAP@0.5, mAP@0.5:0.95, waktu pelatihan dan waktu deteksi.

### 3.3.3.4 Evaluasi Model

Berdasarkan hasil pengujian model, kinerja model akan dianalisis dan dievaluasi lebih lanjut mengenai kelebihan dan kekurangan model. Evaluasi dilakukan berdasarkan hasil uji coba model sebagai simulasi dunia nyata. Uji coba model juga dilakukan terhadap citra *cluttered-background* dan *mosaic*. Hasil evaluasi digunakan sebagai pertimbangan dalam menemukan model terbaik untuk mendekripsi penyakit tanaman kentang.

### **3.3.4 Penulisan Naskah Ilmiah**

Setelah metode penelitian selesai dilakukan, maka dilanjutkan penulisan hasil penelitian. Berdasarkan hasil penelitian akan diperoleh kesimpulan mengenai model YOLO yang terbaik dalam deteksi penyakit tanaman kentang. Penulisan naskah juga bertujuan memberi saran dan pertimbangan dari hasil penelitian ini agar dikembangkan pada penelitian selanjutnya.

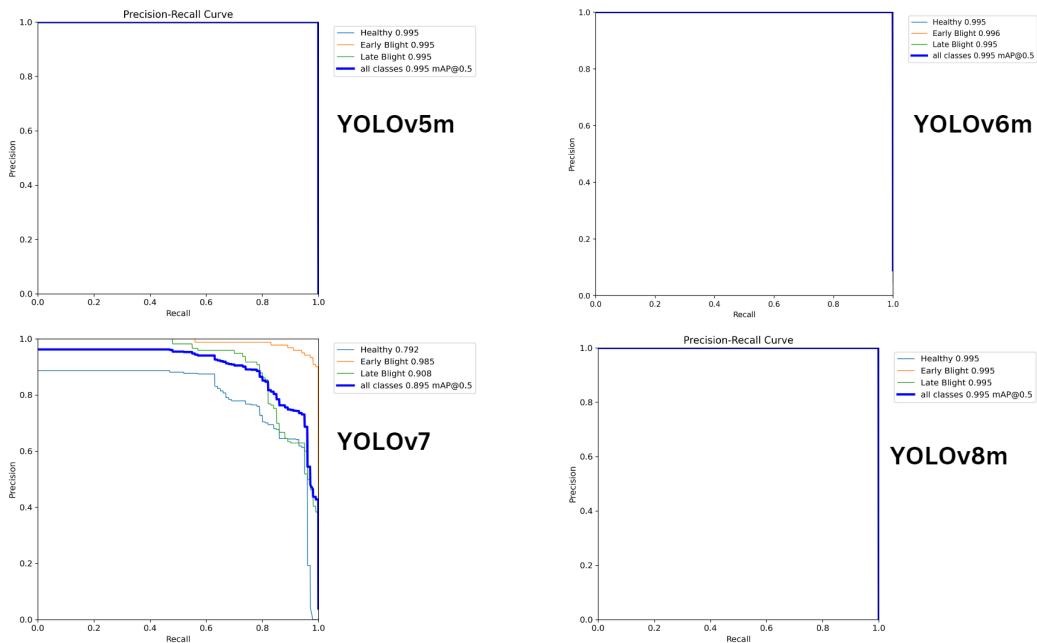
## BAB IV

# HASIL DAN PEMBAHASAN

### 4.1 Pengujian Model

#### 4.1.1 Akurasi Deteksi

Pada subbab ini akan dipaparkan mengenai perbandingan akurasi deteksi di antara empat model yang telah diuji berdasarkan nilai mAP. Perhitungan mAP@0.5 pada 300 data pengujian dapat divisualisasikan dengan kurva *precision-recall* pada Gambar 4.1.



Gambar 4.1. Kurva *Precision-Recall*

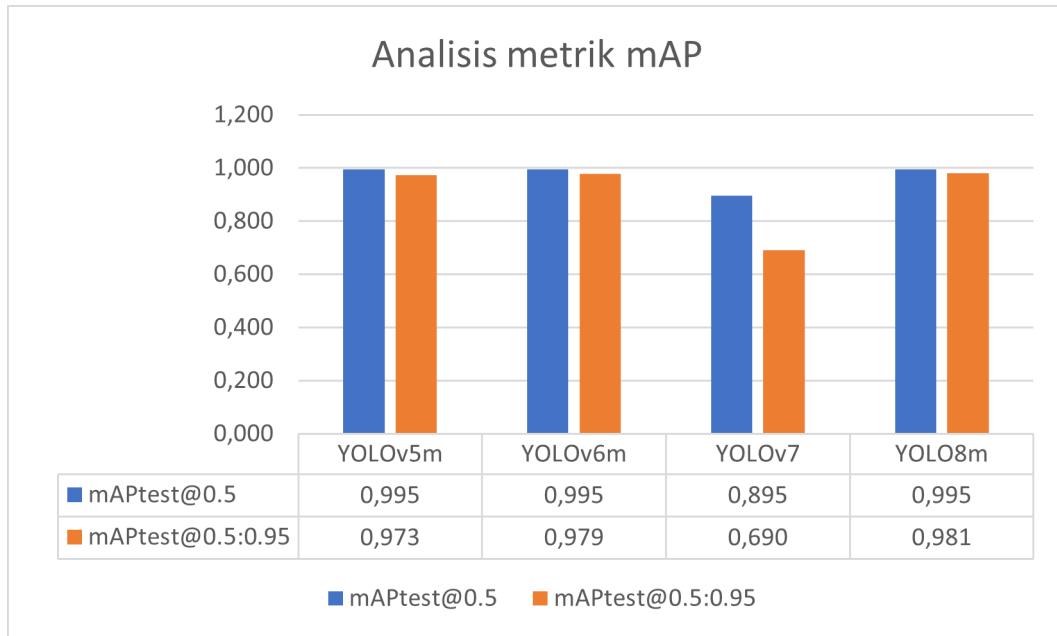
Berdasarkan kurva *precision-recall* pada Gambar 4.1 dapat dirinci nilai AP@0.5 setiap kelas yang dapat ditunjukkan pada Tabel 4.1.

Tabel 4.1. Perbandingan AP@0.5 dan mAP@0.5 setiap model

Model	healthy	early blight	late blight	mAP@0.5
YOLOv5m	0.995	0.995	0.995	0.995
YOLOv6m	0.995	0.996	0.995	0.995
YOLOv7	0.792	0.985	0.908	0.895
YOLOv8m	0.995	0.995	0.995	0.995

Berdasarkan nilai AP@0.5 dan mAP@0.5 yang diperoleh dari pengujian model, model YOLOv7 memiliki akurasi deteksi paling rendah di antara keempat model. Metrik mAP@0.5 model YOLOv7 bernilai 0,895, dengan kelas yang dapat diklasifikasikan

paling baik adalah kelas “*early blight*”. Sementara ketiga model lainnya mencapai nilai 0,995, artinya model YOLOv5m, YOLOv6m, dan YOLOv8m tidak memiliki perbedaan akurasi deteksi apabila menggunakan ambang batas IoU 0,5. Untuk menganalisis akurasi deteksi lebih jauh, ambang batas IoU ditingkatkan dengan menghitung metrik mAP@0.5:0.95. Perbandingan mAP@0.5 dan mAP@0.5:0.95 keempat model dapat divisualisasikan dengan Gambar 4.2.

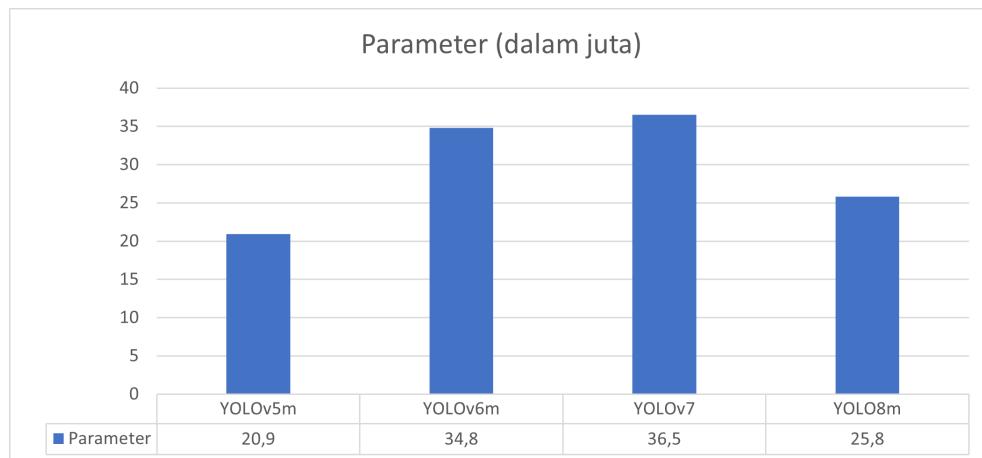


Gambar 4.2. Hasil Akurasi Deteksi Model

Nilai mAP@0.5:0.95 tertinggi dicapai oleh YOLOv8m dengan nilai 0,981. Kemudian diikuti model YOLOv6m dengan nilai 0,979, YOLOv5m dengan nilai 0,973, dan YOLOv7 dengan nilai 0,690. Nilai mAP@0.5:0.95 antara ketiga model yaitu YOLOv5m, YOLOv6m, dan YOLOv8m tidak berbeda signifikan. Sementara model YOLOv7 memiliki akurasi yang rendah secara signifikan dibanding ketiga model lainnya.

#### 4.1.2 Kompleksitas Model

Kompleksitas model dapat diukur berdasarkan jumlah parameter model ketika diimplementasikan pada dataset. Jumlah parameter model YOLOv5m, YOLOv6m, YOLOv7, dan YOLOv8m dipaparkan pada Gambar 4.3. Model yang memiliki jumlah parameter terbanyak di antara keempat model adalah YOLOv7. Dengan banyaknya jumlah parameter, kompleksitas model YOLOv7 juga tinggi. Kompleksitas model paling rendah adalah YOLOv5m dengan jumlah parameter 20,9 juta, kemudian diikuti YOLOv8m dengan jumlah parameter 25,8 juta. Kedua model tersebut memiliki parameter yang jauh lebih sedikit dibanding YOLOv6m yang berjumlah 34,8 juta. Jumlah parameter YOLOv6m tidak memiliki perbedaan signifikan dengan model YOLOv7. Dengan demikian,

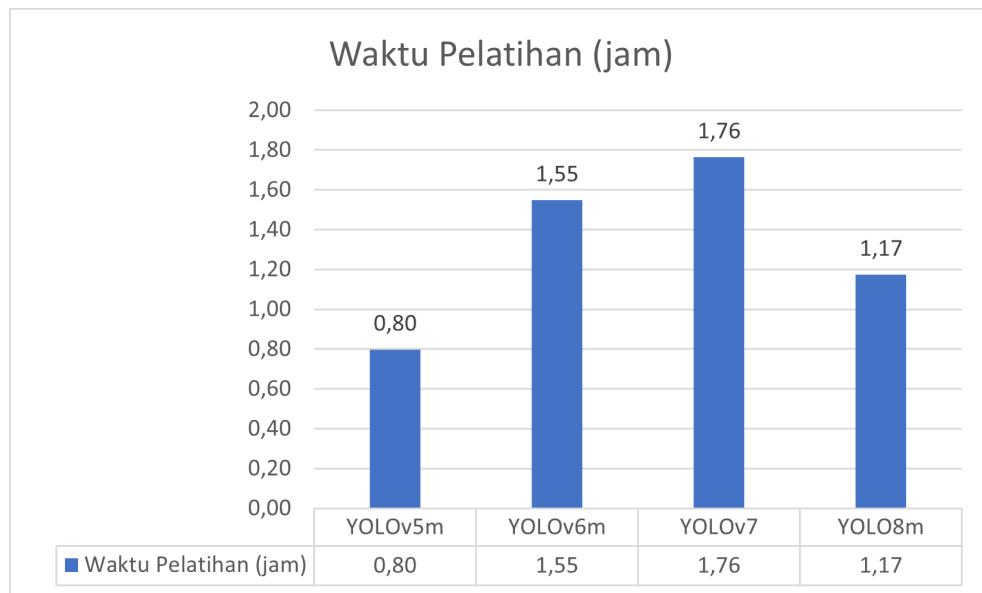


Gambar 4.3. Perbandingan Jumlah Parameter Model

YOLOv7 dan YOLOv6m kurang efisien karena membutuhkan memori dan harga komputasi yang tinggi.

#### 4.1.3 Waktu Komputasi

Waktu komputasi dianalisis berdasarkan waktu pelatihan dan waktu deteksi. Waktu pelatihan adalah waktu yang diperlukan untuk melatih dan memvalidasi model. Pada penelitian ini waktu pelatihan diukur dalam satuan jam. Waktu pelatihan untuk setiap model dapat dilihat pada Gambar 4.4.



Gambar 4.4. Waktu Pelatihan Model

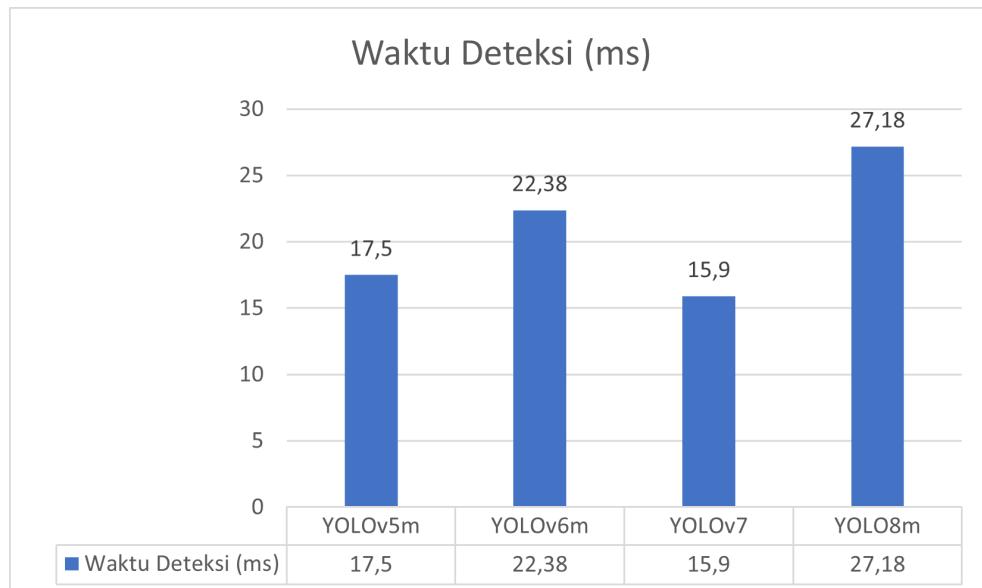
YOLOv7 memerlukan waktu pelatihan paling lama dibanding model lainnya dengan waktu 1,76 jam selama pelatihan 50 epoch. Berikutnya adalah model YOLOv6m, model YOLOv8m, kemudian terakhir model YOLOv5m yang memerlukan waktu pelatihan paling cepat.

Waktu deteksi adalah waktu yang dibutuhkan model dalam mendeteksi objek pada satu citra. Pada penelitian ini pengujian dilakukan pada 300 citra, sehingga waktu deteksi yang dianalisis adalah rata-rata waktu deteksi dari 300 citra. Waktu deteksi diukur dalam satuan ms. Waktu deteksi adalah total dari waktu pra-inferensi, waktu inferensi, dan waktu pasca-inferensi yang dirinci pada Tabel 4.2.

Tabel 4.2. Waktu Deteksi Model

Model	pra-inferensi	inferensi	pasca-inferensi	Waktu Deteksi
YOLOv5m	0,30	14,70	2,50	17,50
YOLOv6m	0,22	20,88	1,28	22,38
YOLOv7	-	13,90	2,00	15,90
YOLOv8m	2,88	21,42	2,88	27,18

Perbandingan waktu deteksi pada pengujian keempat model dapat divisualisasikan pada Gambar 4.5. YOLOv8m membutuhkan waktu paling lama untuk mendeteksi objek pada citra yaitu selama 27,18 ms. Waktu deteksi paling cepat dilakukan oleh model YOLOv7, kemudian diikuti model YOLOv5m yang tidak berbeda signifikan dengan YOLOv7. Sementara model YOLOv6m memiliki waktu deteksi lebih lama dibanding YOLOv7 dan YOLOv5m. Dengan demikian, model YOLOv5m unggul dalam waktu komputasi. YOLOv5m memiliki waktu pelatihan tercepat yang perbedaannya signifikan terhadap model lainnya. Sementara YOLOv7 memiliki waktu deteksi tercepat tetapi perbedaannya tidak signifikan terhadap model YOLOv5m.



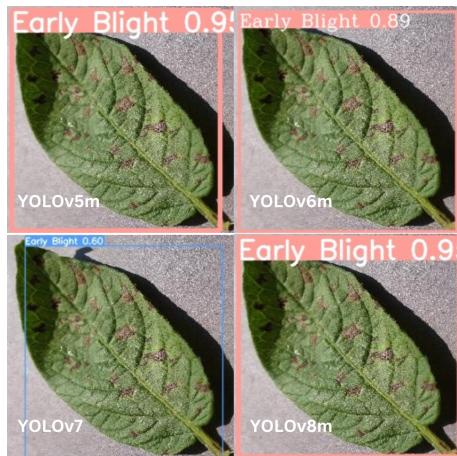
Gambar 4.5. Waktu Deteksi Model

## 4.2 Evaluasi Model

Evaluasi model dilakukan dengan menganalisis kelebihan dan kekurangan model dari berbagai sudut pandang. Analisis kelebihan dan kekurangan model dipertimbangkan terhadap hasil pengujian model yaitu akurasi deteksi, kompleksitas model, dan waktu komputasi. Pengujian model berdasarkan akurasi deteksi, kompleksitas model, dan waktu komputasi dapat ditunjukkan pada Tabel 4.3.

Tabel 4.3. Hasil Pengujian Model

Model	mAP@0.5	mAP@0.5:0.95	Parameter	Pelatihan	Deteksi
YOLOv5m	0,995	0,973	20,9 juta	0,80 jam	17,50 ms
YOLOv6m	0,995	0,979	34,8 juta	1,55 jam	22,38 ms
YOLOv7	0,895	0,690	36,5 juta	1,76 jam	15,90 ms
YOLOv8m	0,995	0,981	25,8 juta	1,17 jam	27,18 ms

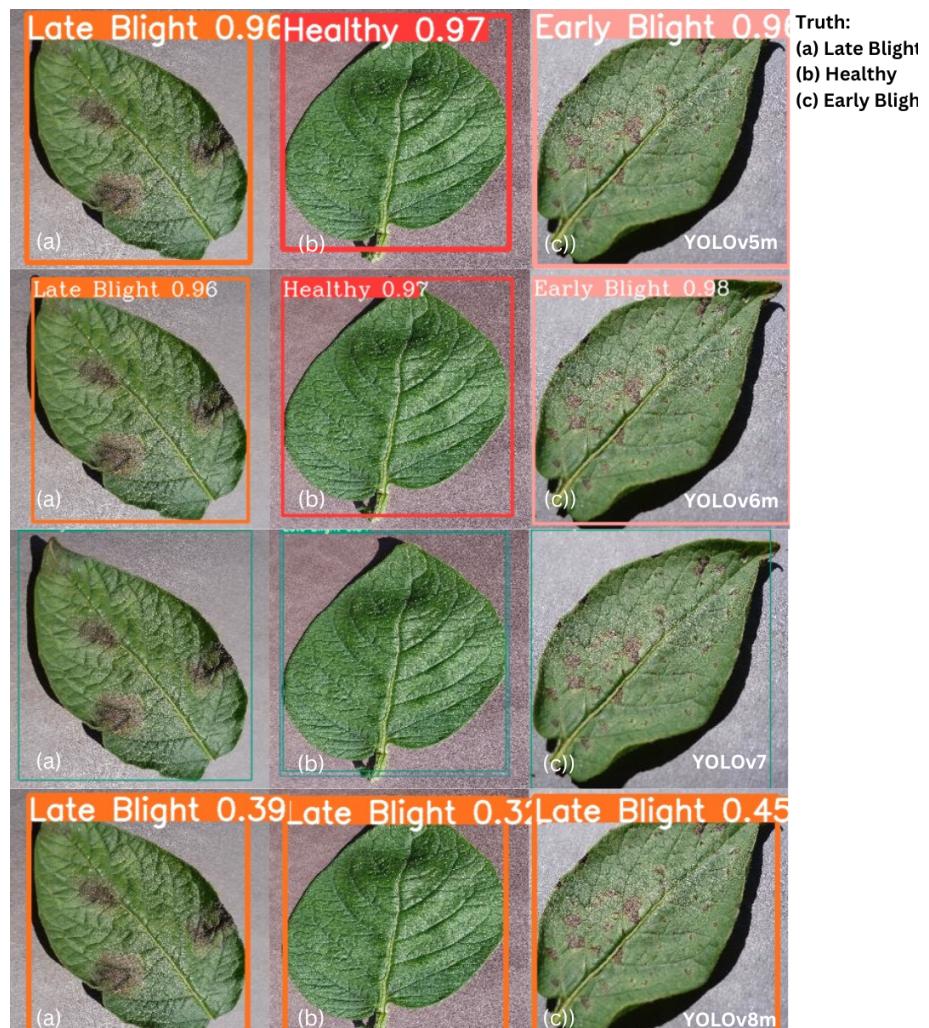


Gambar 4.6. Sampel Hasil Deteksi Citra pada Data Pengujian

Berdasarkan hasil pengujian model, model YOLOv5m, YOLOv6m, dan YOLOv8m memiliki nilai mAP@0.5 yang sama. Sementara nilai mAP@0.5:0.95 dari ketiga model terdapat perbedaan, dengan urutan dari nilai tertinggi diperoleh model YOLOv8m, YOLOv6m, kemudian YOLOv5m. Perbedaan nilai mAP@0.5:0.95 ketiga model tidak berbeda secara signifikan. Perbedaan signifikan terlihat pada nilai mAP model YOLOv7 dibanding ketiga model lainnya, baik nilai mAP@0.5 maupun mAP@0.5:0.95. Gambar 4.6 menunjukkan sampel hasil deteksi pada data pengujian memiliki nilai *confidence* yang tinggi, kecuali hasil deteksi model YOLOv7. Meskipun demikian, hasil klasifikasi objek dengan keempat model benar, yaitu kelas “*Early Blight*”. Namun nilai AP@0.5 untuk kelas *early blight* adalah AP@0.5 paling tinggi dibanding kelas lainnya pada model YOLOv7 seperti yang sudah dipaparkan pada subbab 4.1.1. Sementara nilai AP@0.5 pada ketiga model lainnya bernilai seimbang. Pengujian ini menunjukkan bahwa model YOLOv7 memiliki kekurangan dalam hal akurasi deteksi.



Gambar 4.7. Sampel Hasil Deteksi Citra *Cluttered-background*



Gambar 4.8. Sampel Hasil Deteksi Citra *Mosaic*

Deteksi objek pada data pengujian tergolong mudah karena variasi citranya serupa dengan data pelatihan dan data validasi. Namun pada penerapan di dunia nyata, citra daun yang dideteksi dapat lebih bervariasi. Oleh karena itu, pada penelitian ini dilakukan uji coba model untuk mendeteksi citra dengan variasi yang lebih sulit dideteksi. Model dicoba untuk mendeteksi citra *cluttered-background* dan citra *mosaic*.

Hasil deteksi keempat model pada citra *cluttered-background* ditunjukkan pada Gambar 4.7. Berdasarkan percobaan pada citra *cluttered-background* dapat diketahui bahwa hasil deteksi YOLOv7 dan YOLOv8m kurang akurat. Hal ini ditunjukkan dengan nilai *confidence box* yang rendah. Sementara hasil deteksi terbaik dilakukan oleh model YOLOv6m, diikuti model YOLOv5m dengan perbedaan yang tidak signifikan.

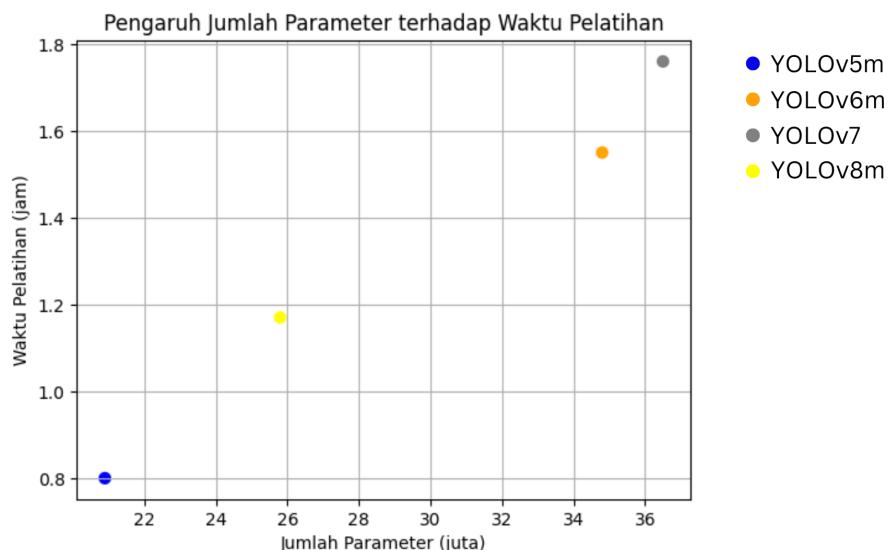
Kemudian hasil deteksi keempat model pada citra *mosaic* ditunjukkan pada Gambar 4.8. Pada percobaan citra *mosaic* dapat diketahui bahwa model YOLOv7 dan YOLOv8m tidak berhasil mendeteksi penyakit tanaman dengan tepat. Sementara deteksi penyakit tanaman pada citra *mosaic* berhasil dideteksi dengan benar oleh model YOLOv5m dan YOLOv6m. Maka meskipun YOLOv8m memiliki nilai mAP@0.5:0.95 tertinggi, YOLOv8m memiliki kelemahan dalam mendeteksi objek pada variasi citra *cluttered-background* dan *mosaic*.



Gambar 4.9. Sampel Hasil Deteksi Frame dari Video [22]

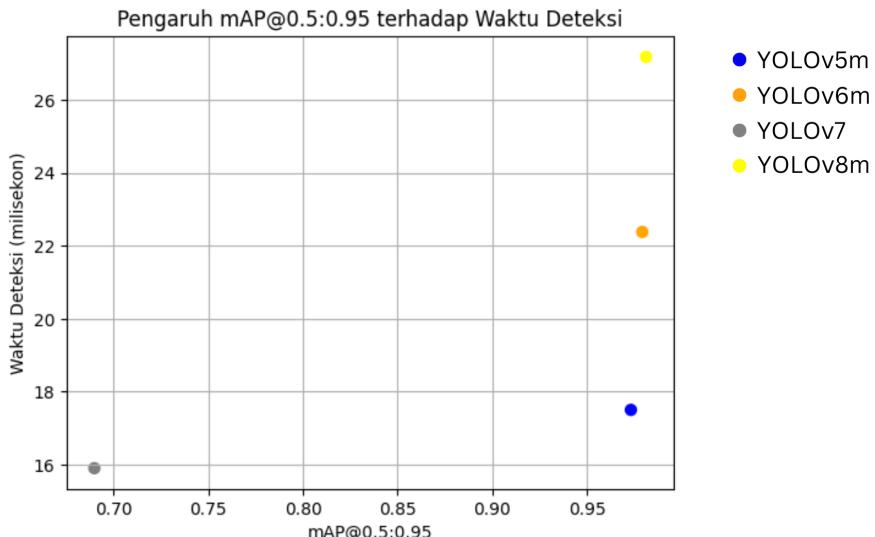
Citra yang dideteksi sebelumnya merupakan data citra terstruktur yang fokus pada satu daun kentang. Untuk mengevaluasi model lebih jauh, model diuji coba dengan frame dari video tanaman kentang yang terinfeksi penyakit *early blight*. Hasil deteksi model dapat dilihat pada Gambar 4.9. Keempat model tidak berhasil mendeteksi penyakit tanaman kentang dengan tepat pada frame sebelah kiri. Sementara frame sebelah kanan berhasil dideteksi dengan akurasi cukup baik oleh model YOLOv6m. Sementara model YOLOv5m, YOLOv7 dan YOLOv8m tidak dapat mendeteksi daun sebagai *early blight*. Dengan demikian, model yang unggul dalam akurasi deteksi adalah YOLOv6m.

Akurasi model dalam mendeteksi objek dapat ditingkatkan dengan penambahan epoch dan dataset dalam proses pelatihan. Proses pelatihan pada penelitian ini masih menggunakan data citra terstruktur dengan rasio ukuran teratur dan fokus pada daun kentang. Oleh karena itu, akurasi deteksi juga dapat ditingkatkan dengan melibatkan tipe data citra yang tidak terstruktur atau memiliki ukuran objek daun kentang yang berbeda-beda pada proses pelatihan model. Namun teknik-teknik peningkatan akurasi deteksi tersebut kurang efisien apabila diterapkan pada model yang kompleks. Kompleksitas model dapat diukur dari jumlah parameter seperti yang sudah dibahas pada subbab 4.1.2. Pada bagian tersebut dijelaskan bahwa YOLOv7 memiliki jumlah parameter paling banyak. Pengaruh jumlah parameter terhadap waktu pelatihan dapat ditunjukkan pada Gambar 4.10 yang membuktikan bahwa semakin banyak jumlah parameter, model akan memerlukan waktu pelatihan semakin lama. Model yang memiliki jumlah parameter paling banyak dan waktu pelatihan paling lama adalah YOLOv7. Dengan demikian, penambahan epoch dan data untuk meningkatkan akurasi deteksi pada pelatihan model YOLOv7 akan meningkatkan waktu pelatihan secara signifikan.



Gambar 4.10. Perbandingan Jumlah Parameter dengan Waktu Pelatihan Model

Selain waktu pelatihan, waktu deteksi adalah hal yang perlu dipertimbangkan dalam evaluasi model seperti yang dijelaskan pada subbab 4.1.3. Model yang lama dalam mendeteksi objek, kurang tepat jika diimplementasikan pada sistem berbasis *real-time*. Penelitian ini bertujuan menemukan model yang berbasis *real-time* karena deteksi penyakit tanaman memerlukan waktu yang cepat dalam penanganannya. Berdasarkan tinjauan pustaka, dinyatakan bahwa mAP yang tinggi cenderung berhubungan dengan lamanya waktu deteksi. Perbandingan nilai mAP@0.5:0.95 terhadap waktu deteksi model dapat dilihat pada Gambar 4.11 yang menunjukkan bahwa semakin tinggi nilai mAP@0.5:0.95 menyebabkan waktu yang diperlukan model untuk mendeteksi objek semakin lama.



Gambar 4.11. Perbandingan mAP@0.5:0.95 terhadap Waktu Deteksi

Berdasarkan hasil pengujian model, nilai mAP@0.5:0.95 tertinggi dicapai oleh YOLOv8m karena data pengujian merupakan citra terstruktur, tetapi YOLOv8m memiliki kelemahan dalam mendekripsi objek pada citra *cluttered-background* dan *mosaic*, serta frame dari video yang tidak terstruktur. Selain itu, YOLOv8m memiliki waktu deteksi paling lama dibanding model lainnya.

YOLOv7 memiliki akurasi deteksi yang sangat rendah, jumlah parameter yang sangat banyak, dan waktu pelatihan yang sangat lama. YOLOv7 memiliki waktu deteksi yang cepat tetapi hasil deteksinya memiliki akurasi yang sangat rendah dibanding model lainnya. Penambahan epoch dan dataset untuk meningkatkan akurasi YOLOv7 tidak efisien karena jumlah parameter dan waktu pelatihannya tidak sebanding dengan kerendahan akurasinya.

Hasil terbaik dalam mendekripsi objek pada data pengujian maupun pada citra *cluttered-background* dan *mosaic*, serta frame dari video dicapai oleh model YOLOv6m. Dengan demikian, arsitektur dan cara kerja model YOLOv6m memiliki keunggulan akurasi deteksi dalam mendekripsi penyakit tanaman menggunakan dataset daun kentang. Sementara itu, YOLOv6m memiliki *trade-off* terhadap kompleksitas model dan waktu pelatihan yang cukup tinggi. Namun dalam hal tersebut, YOLOv6m masih lebih unggul dibanding YOLOv7. Selain itu, waktu deteksi yang diperlukan YOLOv6m masih lebih unggul dibanding YOLOv8m, serta hanya selisih 4,88 ms dari model YOLOv5m dan 6,48 ms dari YOLOv7 sehingga masih tergolong *real-time*.

Di antara model YOLOv5m, YOLOv6m, dan YOLOv8m, model YOLOv5m memiliki nilai mAP@0.5:0.95 paling rendah tetapi tidak signifikan. YOLOv5m unggul dengan rendahnya kompleksitas model dan waktu komputasi. YOLOv5m juga masih unggul dalam percobaan deteksi objek pada variasi citra *cluttered-background* dan *mo-*

*saic*, tetapi lemah dalam mendeteksi frame dari video. Dengan demikian, arsitektur dan cara kerja model YOLOv6m memiliki keunggulan dalam mendeteksi penyakit tanaman menggunakan dataset daun kentang.

### 4.3 Perbandingan Hasil Penelitian dengan Hasil Terdahulu

Pada tinjauan pustaka subbab 2.1 terdapat beberapa penelitian terdahulu yang melakukan analisis kinerja model CNN dalam visi komputer. Tugas akhir ini mengajukan beberapa kelebihan yang diharapkan mampu memperbaiki kekurangan penelitian terdahulu. Namun adanya keterbatasan dalam tugas akhir ini tentunya tidak dapat dihindari, sehingga tugas akhir ini masih memiliki kekurangan dibanding penelitian terdahulu. Perbandingan hasil penelitian tugas akhir ini dengan penelitian terdahulu dapat dirinci pada Tabel 4.4.

Tabel 4.4. Perbandingan Hasil Penelitian

Peneliti	Analisis	Model terbaik
Tugas Akhir KC dkk, (2019) [44]	Akurasi, Kompleksitas, Waktu Komputasi	YOLOv6m
Rashid dkk, (2021) [38]	Akurasi, Kompleksitas	MobileNet
Kothari dkk, (2022) [40]	Akurasi, Kompleksitas	YOLOv5s
Dang dkk, (2023) [47]	Akurasi, Waktu Pelatihan	ResNet50
	Akurasi, Kompleksitas, Waktu Inferensi	YOLOv5m

KC, Kamal dkk (2019) [44] membandingkan model AlexNet, VGG, MobileNet, Modified MobileNet, dan Reduced MobileNet. Penelitian ini menyimpulkan bahwa MobileNet merupakan model paling berhasil karena dapat mencapai akurasi yang tinggi yaitu 98,65% dengan parameter berjumlah 3,2 juta. Kelebihan penelitian KC, Kamal dkk dibanding penelitian tugas akhir ini adalah kemampuan modelnya dalam mengklasifikasi 55 kelas penyakit pada 38 tanaman berdasarkan citra daun. Sementara penelitian tugas akhir ini hanya dapat mendeteksi tanaman kentang yang sehat dan terinfeksi 2 kelas penyakit. Namun penelitian KC, Kamal dkk tidak dapat melakukan lokalisasi objek, serta tidak ada analisis kinerja model terhadap waktu komputasi.

Selanjutnya Rashid dkk, 2021 [38] melakukan modifikasi model CNN dengan teknik augmentasi YOLOv5s. Model yang digunakan adalah model modifikasi CNN dengan YOLOv5s. Akurasi tertinggi yang diperoleh dari model tersebut adalah 99,75% dengan parameter berjumlah 8,5 juta. Oleh karena itu, disimpulkan bahwa model ini lebih baik dibanding model VGG16, InceptionResNetV2, DenseNet\_121, DenseNet169, dan Xception. Penelitian Rashid dkk belum ada analisis kinerja model terhadap waktu komputasi. Model penelitian Rashid dkk juga hanya mampu melakukan klasifikasi objek. Sementara tugas akhir ini outputnya berupa klasifikasi dan lokalisasi objek sehingga memiliki jumlah parameter lebih banyak.

Pada tahun 2022, Kothari dkk [40] membandingkan berbagai model CNN yaitu CNN, GoogleNet, ResNet50, dan VGG16. Model terbaik adalah ResNet50 dengan akurasi tertinggi, serta memiliki waktu pelatihan lebih cepat dibanding VGG. Penelitian Kothari dkk belum menganalisis kinerja model berdasarkan kompleksitas model dan waktu deteksi. Selain itu, belum ada lokalisasi objek, sedangkan tugas akhir ini outputnya berupa klasifikasi dan lokalisasi objek sehingga membutuhkan waktu pelatihan lebih lama.

Pada tahun 2023, Dang dkk [47] menggunakan model YOLO untuk mendekripsi 12 kelas gulma. Model yang digunakan adalah 25 detektor model YOLO mulai dari YOLOv3 hingga YOLOv7 untuk dibandingkan kinerjanya. Penelitian Dang dkk menghasilkan model terbaik YOLOv5m dengan parameter 20 juta dan waktu inferensi 3 ms. Namun model ini berhasil mencapai mAP@0.5 94,02% dan mAP@0.5:0.95 87,10%. Kelebihan penelitian Dang dkk dibanding penelitian tugas akhir ini adalah kemampuan modelnya dalam mendekripsi 12 kelas gulma pada tanaman kapas. Sementara penelitian tugas akhir ini hanya dapat mendekripsi tanaman kentang yang sehat dan terinfeksi 2 kelas penyakit berdasarkan citra daun. Selain itu, waktu inferensi yang dibutuhkan pada tugas akhir lebih lama, tetapi memiliki nilai mAP lebih tinggi. Penelitian Dang dkk belum menganalisis kinerja model terhadap lamanya waktu pelatihan.

Berdasarkan beberapa penelitian terdahulu, ternyata model YOLOv5 memiliki keunggulan dalam tingginya akurasi klasifikasi maupun deteksi, rendahnya kompleksitas model, dan rendahnya waktu inferensi yang diperlukan. Oleh karena itu, model terbaik yang diajukan dalam penelitian ini yaitu YOLOv6m dapat dijadikan pertimbangan dalam penelitian selanjutnya agar dapat meningkatkan kemampuan model deteksi objek di bidang pertanian, khususnya dalam mendekripsi penyakit tanaman. Agar lebih mudah dipahami, kelebihan dan kekurangan metode penelitian tugas akhir dibanding penelitian pada tinjauan pustaka dapat dipaparkan pada Tabel 4.5.

Tabel 4.5. Kelebihan dan Kekurangan Tugas Akhir

Peneliti	Kemampuan	Kelebihan	Kekurangan
KC dkk, (2019) [44]	Klasifikasi	Analisis waktu komputasi	Kelas sedikit
Rashid dkk, (2021) [38]	Klasifikasi	Analisis waktu komputasi	-
Kothari dkk, (2022) [40]	Klasifikasi	Analisis waktu deteksi	-
Dang dkk, (2023) [47]	Deteksi	Analisis waktu pelatihan	Kelas sedikit

## **BAB V**

### **KESIMPULAN DAN SARAN**

#### **5.1 Kesimpulan**

Penelitian tugas akhir ini menganalisis kinerja 4 model deteksi objek YOLO yaitu model YOLOv5m, YOLOv6m, YOLOv7, dan YOLOv8m. Analisis kinerja dilakukan agar diketahui kinerja model dan ditemukan model terbaik untuk mendeteksi penyakit tanaman kentang. Analisis kinerja menghasilkan poin-poin berikut:

1. YOLOv5m memiliki akurasi deteksi tinggi pada citra terstruktur, tetapi rendah pada citra tidak terstruktur. YOLOv5m memiliki kompleksitas model paling rendah dan waktu komputasi paling cepat dibanding model lainnya.
2. YOLOv6m memiliki akurasi deteksi tinggi pada citra terstruktur maupun tidak terstruktur. YOLOv6m memiliki kompleksitas model dan waktu pelatihan yang lebih rendah dibanding YOLOv7, serta waktu deteksi lebih cepat dibanding YOLOv8m.
3. YOLOv7 memiliki hasil paling buruk karena dengan kompleksitas model yang tinggi dan waktu komputasi yang lama, akurasi deteksi yang dicapai sangat rendah.
4. YOLOv8m memiliki akurasi deteksi tinggi pada citra terstruktur, tetapi rendah pada citra tidak terstruktur. Kompleksitas dan waktu pelatihan YOLOv8m lebih tinggi dibanding YOLOv5m, serta waktu deteksinya paling lama.

Berdasarkan hasil analisis kinerja, YOLOv6m memiliki kompleksitas model dan waktu komputasi yang cukup tinggi tetapi menghasilkan akurasi deteksi yang sebanding. Deteksi YOLOv6m mencapai hasil akurasi yang tinggi, baik pada citra terstruktur maupun citra tidak terstruktur. Dengan demikian, model YOLO terbaik yang dapat digunakan untuk mendeteksi penyakit tanaman kentang adalah model YOLOv6m.

#### **5.2 Saran**

Penelitian lanjutan dapat dilakukan untuk meningkatkan manfaat deteksi objek di bidang pertanian, khususnya dalam mendeteksi penyakit tanaman. Dalam implementasinya, model deteksi objek dapat ditingkatkan kinerjanya. Peningkatan dapat dilakukan berdasarkan perspektif pertanian maupun teknologi. Dalam perspektif pertanian dapat dilakukan penambahan objek yang dideteksi, seperti jenis tanaman, penyakit, gulma, dan hama. Berdasarkan perspektif teknologi dapat dilakukan peningkatan hyperparameter, jumlah dan jenis data, augmentasi, serta optimisasi dalam pelatihan model. Selain itu dapat diuji model deteksi objek lainnya, seperti model YOLO-NAS yang baru dirilis tanggal 3 Mei 2023.

## DAFTAR PUSTAKA

- [1] A. Irawan, “Ciri ciri fisik daun tanaman yang terkena penyakit bercak kering (alternaria),” Agrokompleks Kita, 2019. [Online]. Available: <https://agrokompleskita.com/penyakit-bercak-kering-alternaria-pada-tanaman-kentang/ciri-ciri-fisik-daun-tanaman-yang-terkena-penyakit-bercak-kering-alternaria/>
- [2] “Perlu di waspadai, penyakit busuk daun yang menyerang tanaman kentang,” PT. Hextar Fertilizer Indonesia. [Online]. Available: <https://www.hextarfertilizerindonesia.com/perlu-di-waspadai-penyakit-busuk-daun-yang-menyerang-tanaman-kentang/>
- [3] Aaradhya, “11 dogs similar to wolf,” The Arena Group, March 2023. [Online]. Available: <https://pethelpful.com/dogs/11-dogs-similar-to-wolf>
- [4] G. M. Su, “Deep learning bible,” May 2023. [Online]. Available: <https://wikidocs.net/163640>
- [5] “Amazon machine learning (developer guide version latest) model fit: Underfitting vs. overfitting,” Amazon Web Services, Inc, 2023. [Online]. Available: <https://docs.aws.amazon.com/machine-learning/latest/dg/model-fit-underfitting-vs-overfitting.html>
- [6] A. Lekhtman, “Data science in medicine — precision recall or specificity sensitivity?” Towards Data Science, August 2019. [Online]. Available: <https://towardsdatascience.com/should-i-look-at-precision-recall-or-specificity-sensitivity-3946158aace1>
- [7] “Object detection intersection vs union,” Baeldung, May 2023. [Online]. Available: <https://www.baeldung.com/cs/object-detection-intersection-vs-union>
- [8] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, S. J, M. A. Fadhel, M. Al-Amidie, and L. Farhan, “Review of deep learning: concepts, cnn architectures, challenges, applications, future directions,” *Journal of Big Data*, vol. 8, no. 53, pp. 1–74, 2021.
- [9] S. Oni, “Understanding anchors(backbone of object detection) using yolo,” Becoming Human: Artificial Intelligence Magazine, April 2020. [Online]. Available: <https://becominghuman.ai/understanding-anchors-backbone-of-object-detection-using-yolo-54962f00fbdb>
- [10] Kukil, “Yolox object detector paper explanation and custom training,” LearnOpenCV, October 2022. [Online]. Available: <https://learnopencv.com/yolox-object-detector-paper-explanation-and-custom-training/>
- [11] J. R. Terven and D. M. Cordova-Esparaza, “A comprehensive review of yolo: From yolov1 to yolov8 and beyond,” 2023. [Online]. Available: <https://arxiv.org/abs/2304.00501>
- [12] S. R. Hussain, “Detecting cow teat stall numbers with cowstallnumbers dataset,” 2023. [Online]. Available: [https://www.researchgate.net/publication/370153499\\_Detecting\\_Cow\\_Teat\\_Stall\\_Numbers\\_with\\_CowStallNumbers\\_Dataset](https://www.researchgate.net/publication/370153499_Detecting_Cow_Teat_Stall_Numbers_with_CowStallNumbers_Dataset)

- [13] hhaAndroid, March 2023. [Online]. Available: <https://github.com/open-mmlab/mmyolo/tree/main/configs/yolov5>
- [14] G. Jocher, Ultralytics LLC, 2023. [Online]. Available: <https://docs.ultralytics.com/yolov5/>
- [15] C. Li, L. Li, and H. Jiang, “Yolov6: A single-stage object detection framework for industrial applications,” September 2022. [Online]. Available: <https://arxiv.org/pdf/2209.02976.pdf>
- [16] X. Li, W. Wang, L. Wu, S. Chen, X. Hu, J. Li, J. Tang, and J. Yang, “Paper review [review] generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection,” CDM Machine Learning Deep Learning, June 2020. [Online]. Available: <https://cdm98.tistory.com/52>
- [17] C. Li, Meituan Vision AI Department, June 2022. [Online]. Available: <https://github.com/meituan/YOLOv6>
- [18] C. Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, “Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors,” July 2022. [Online]. Available: <https://arxiv.org/abs/2207.02696>
- [19] WongKinYiu, July 2022. [Online]. Available: <https://github.com/WongKinYiu/yolov7>
- [20] RangeKing, January 2023. [Online]. Available: <https://github.com/ultralytics/ultralytics/issues/189>
- [21] G. Jocher, A. Chaurasia, and J. Qiu, “Yolo by ultralytics,” 2023. [Online]. Available: <https://docs.ultralytics.com/>
- [22] “Kenali alternaria/ bercak kering dan cara penanggulangan nya,” Rumah Petani Pintar, July 2021. [Online]. Available: <https://youtu.be/P0GPhKz7jTo>
- [23] L. Nindyawati, P. T. Ina, and A. A. I. S. Wiadnyani, “Pengaruh perbandingan kentang kukus dan tepung kacang hijau (*phaseolus radiatus* l.) terhadap karakteristik flakes,” *Jurnal Ilmu dan Teknologi Pangan*, vol. 8, no. 1, pp. 66–74, 2019.
- [24] UMA, “Benefits of potatoes for body health,” November 2022. [Online]. Available: <https://bamai.uma.ac.id/2022/11/02/manfaat-kentang-untuk-kesehatan-tubuh/>
- [25] BPS, “Produksi tanaman sayuran menurut provinsi dan jenis tanaman,” Badan Pusat Statistik, 2022. [Online]. Available: [https://www.bps.go.id/indikator/indikator/view\\_data\\_pub/0000/api\\_pub/eHEwRmg2VUZjY2lWNWNYaVhQK1h4QT09/da\\_05/1](https://www.bps.go.id/indikator/indikator/view_data_pub/0000/api_pub/eHEwRmg2VUZjY2lWNWNYaVhQK1h4QT09/da_05/1)
- [26] —, “Luas panen tanaman sayuran menurut provinsi dan jenis tanaman,” Badan Pusat Statistik, 2022. [Online]. Available: [https://www.bps.go.id/indikator/indikator/view\\_data\\_pub/0000/api\\_pub/bXNVb1pmZndqUDhKWEIUSjhZRitidz09/da\\_05/2](https://www.bps.go.id/indikator/indikator/view_data_pub/0000/api_pub/bXNVb1pmZndqUDhKWEIUSjhZRitidz09/da_05/2)
- [27] hortiindonesia, “Peluang kentang industri terbuka, kenali permasalahannya,” Desember 2021. [Online]. Available: <https://hortiindonesia.com/berita/peluang-kentang-industri-terbuka-kenali-permasalahannya>

- [28] A. Rachman, “Ri borong kentang jerman as, impor naik gila-gilaan,” February 2023. [Online]. Available: <https://www.cnbcindonesia.com/news/20230216072813-4-414181/ri-borong-kentang-jerman-as-impor-naik-gila-gilaan>
- [29] I. Fachmawati, “Budidaya kentang, tanaman bernilai ekonomi tinggi,” November 2020. [Online]. Available: <http://cybex.pertanian.go.id/mobile/artikel/95283/Budidaya-Kentang-Tanaman-bernilai-Ekonomi-Tinggi/>
- [30] H. Rahmat, “Tesis: Analisis perbandingan kinerja algoritme object detection berbasis deep learning pada perangkat komputasi terbatas (studi kasus: Deteksi kelainan daun tanaman melon),” Ilmu Komputer Sekolah Pascasarjana Institut Pertanian Bogor, 2021.
- [31] P. U. Rakhmawati, Y. M. Pranoto, and E. Setyati, “Klasifikasi penyakit daun kentang berdasarkan fitur tekstur dan fitur warna menggunakan support vector machine,” in *Seminar Nasional Teknologi dan Rekayasa (SENTRA)*, 2018.
- [32] M. R. K. Huda and N. Nafi’iyah, “Identifikasi penyakit daun kentang berdasarkan fitur warna, tekstur, dan bentuk dengan svm dan knn,” in *Seminar Nasional Multimedia Artificial Intelligence SMAI 2020*, 2020.
- [33] A. H. Kahfi, “Tesis: Identifikasi penyakit pada tanaman kentang dengan k-nearest neighbor berdasarkan fitur warna dan tekstur daun,” Program Studi Ilmu Komputer (S2) STMIK Nusa Mandiri, 2020.
- [34] L. Amatullah, I. Ein, and M. M. Santoni, “Identifikasi penyakit daun kentang berdasarkan fitur tekstur dan warna dengan menggunakan metode k-nearest neighbor,” in *Seminar Nasional Mahasiswa Ilmu Komputer dan Aplikasinya (SENAMIKA)*, 2021.
- [35] S. I. Mohamed, “Potato leaf disease diagnosis and detection system based on convolution neural network,” *International Journal of Recent Technology and Engineering (IJRTE)*, vol. 9, no. 4, pp. 254–259, 2020.
- [36] M. Ahmad, J. Joe, and D. Han, “Cortexnet: Convolutional neural network with visual cortex in human brain,” in *2018 IEEE International Conference on Consumer Electronics - Asia (ICCE-Asia)*. IEEE, 2018.
- [37] D. F. NINGSIH, “Skripsi: Klasifikasi jenis penyakit daun kentang menggunakan convolutional neural network (cnn) model arsitektur googlenet,” Program Studi Teknik Informatika Universitas Yudharta Pasuruan, 2021.
- [38] J. Rashid, I. Khan, G. Ali, S. H. Almotiri, M. A. AlGhamdi, and K. Masood, “Multi-level deep learning model for potato leaf disease recognition,” *MDPI Open Access Journal*, vol. 10, no. 17, pp. 1–27, 2021.
- [39] A. M. Lesmana, R. P. Fadhillah, and C. Rozikin, “Identifikasi penyakit pada citra daun kentang menggunakan convolutional neural network (cnn),” *Jurnal Sains dan Informatika*, vol. 8, no. 1, pp. 21–30, 2022.
- [40] D. Kothari, H. Mishra, M. Gharat, V. Pandey, and R. Thakur, “Potato leaf disease detection using deep learning,” *International Journal of Engineering Research Technology (IJERT)*, vol. 11, no. 11, pp. 8–12, 2022.

- [41] F. R. Haikal, “Skripsi: Analisis perbandingan akurasi dan performa algoritma deteksi objek pada yolo v3 dengan single shot multibox detector (ssd),” Program Studi Teknik Informatika Politeknik Negeri Jakarta, 2021.
- [42] M. S. Khatami, “Skripsi: Deteksi kendaraan menggunakan algoritma you only look once (yolo) v3,” Program Studi Informatika Universitas Islam Indonesia, 2022.
- [43] D. S. Trigueros, L. Meng, and M. Hartnett, “Face recognition: From traditional to deep learning methods,” 2018. [Online]. Available: <https://arxiv.org/abs/1811.00116>
- [44] K. KC, Z. Yin, M. Wu, and W. Zhilu, “Depthwise separable convolution architectures for plant disease,” *Computers and Electronics in Agriculture*, vol. 165, p. 104948, 2019. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0168169918318696>
- [45] E. Rasywir, R. Sinaga, and P. Yovi, “Analisis dan implementasi diagnosis penyakit sawit dengan metode convolutional neural network (cnn),” *Paradigma – Jurnal Informatika dan Komputer*, vol. 2, no. 2, pp. 117–123, 2020.
- [46] N. E. M. Khalifa, M. H. N. Taha, L. M. El-Maged, and A. E. Hassanien, *Artificial Intelligence in Potato Leaf Disease Classification: A Deep Learning Approach*. Springer, 2021, pp. 63–79.
- [47] F. Dang, D. Chen, Y. Lu, and Z. Li, “Yoloweeds: A novel benchmark of yolo object detectors for multi-class weed detection in cotton production systems,” *Computers and Electronics in Agriculture*, vol. 205, p. 107655, 2023.
- [48] M. Y. A. Thoriq, K. E. Permana, and I. A. Siradjuddin, “Deteksi wajah manusia berbasis one stage detector menggunakan metode you only look once (yolo),” *JURNAL TEKNOINFO*, vol. 17, no. 1, pp. 66–73, 2023.
- [49] O. E. Karlina and D. Indarti, “Pengenalan objek makanan cepat saji pada video dan real time webcam menggunakan metode you look only once (yolo),” *Jurnal Ilmiah Informatika Komputer*, vol. 24, no. 3, pp. 199–208, 2019.
- [50] G. N. Goenawan, A. N. Tjondrowiguno, and Liliana, “Skripsi: Pengenalan rambu lalu lintas di indonesia secara real-time menggunakan yolov4-tiny,” Program Studi Informatika Universitas Kristen Petra, 2022.
- [51] M. S. Nuha and R. A. H., “Pemanfaatan yolo untuk pengenalan kesegaran buah mangga,” *Joutica : Journal of Informatic Unisla*, vol. 7, no. 1, pp. 513–518, 2022.
- [52] C. Geraldyn and C. Lubis, “Pendeteksian dan pengenalan jenis mobil menggunakan algoritma you only look once dan convolutional neural network,” *Jurnal Ilmu Komputer dan Sistem Informasi*, vol. 8, no. 2, pp. 197–199, 2020.
- [53] J. Adiwibowo, K. Gunadi, and E. Setyadi, “Deteksi alat pelindung diri menggunakan metode yolo dan faster r-cnn,” *JURNAL INFRA*, vol. 8, no. 2, pp. 106–112, 2020.

- [54] R. H. Pramesty, “Tesis: Deteksi dan klasifikasi kerusakan jalan aspal menggunakan metode yolo berbasis citra digital,” Program Magister Departemen Matematika Institut Teknologi Sepuluh Nopember, 2018.
- [55] D. Hu, S. Li, and M. Wang, “Object detection in hospital facilities: A comprehensive dataset and performance evaluation,” *Engineering Applications of Artificial Intelligence*, vol. 123, p. 106223, 2020.
- [56] I. W. Suastawan, “Budidaya kentang,” BPP Kintamani Timur, August 2019. [Online]. Available: <http://cybex.pertanian.go.id/mobile/artikel/65496/Budidaya-Kentang/>
- [57] “Patogen dan hubungannya dengan perkembangan penyakit tanaman,” Jurusan Manajemen Pertanian Lahan Kering Politeknik Pertanian Negeri Kupang, 2023. [Online]. Available: <https://mplk.politanikoe.ac.id/index.php/program-studi/28-manajemen-pertanian-lahan-kering/informasi-materi-kuliah-praktek1/155-patogen-dan-hubungannya-dengan-perkembangan-penyakit-tanaman>
- [58] “Difference between early blight and late blight of potato,” BYJU’S, 2023. [Online]. Available: <https://byjus.com/biology/difference-between-early-blight-and-late-blight-of-potato/>
- [59] W. Xue, K. G. Haynes, and X. Qu, “Characterization of early blight resistance in potato cultivars,” *Plant Disease*, vol. 103, no. 4, pp. 629–637, 2019, pMID: 30742554. [Online]. Available: <https://doi.org/10.1094/PDIS-05-18-0794-RE>
- [60] “Bercak kering,” PEAT GmbH: Plantix. [Online]. Available: <https://plantix.net/id/library/plant-diseases/100321/early-blight>
- [61] R. J. Harahap, “Tim dosen usu dampingi petani tingkatkan produksi kentang,” Tim Dosen Universitas Sumatera Utara, 2022. [Online]. Available: <https://www.usu.ac.id/id/pengabdian-masyarakat/tim-dosen-usu-dampingi-petani-tingkatkan-produksi-kentang>
- [62] “Hawar daun kentang,” PEAT GmbH: Plantix. [Online]. Available: <https://plantix.net/id/library/plant-diseases/100040/potato-late-blight>
- [63] V. Willey and T. Lucas, “Computer vision and image processing: A paper review,” *International Journal Of Artificial Intelligence Research*, vol. 2, no. 1, pp. 28–36, 2018.
- [64] F. A. Rahmah, PT. Algoritma Data Indonesia, January 2022. [Online]. Available: <https://algorit.ma/blog/data-science/overfitting-underfitting/>
- [65] N. Wulandari, “Tesis: Perbandingan implementasi metode deep learning pada deteksi objek di bawah air,” Program Studi Teknologi Informasi Universitas Gadjah Mada, 2022.
- [66] K. E. Koech, “Object detection metrics with worked example,” Towards Data Science, August 2020. [Online]. Available: <https://towardsdatascience.com/should-i-look-at-precision-recall-or-specificity-sensitivity-3946158aace1>

- [67] J. Prakash, “Non maximum suppression: Theory and implementation in pytorch,” LearnOpenCV, June 2021. [Online]. Available: <https://learnopencv.com/non-maximum-suppression-theory-and-implementation-in-pytorch/>
- [68] M. Basavarajaiah, “Maxpooling vs minpooling vs average pooling,” Medium, February 2019. [Online]. Available: <https://medium.com/@bdhumu/which-pooling-method-is-better-maxpooling-vs-minpooling-vs-average-pooling-95fb03f45a9>
- [69] M. A. Zulkiflie, “Skripsi: Implementasi algoritma object detection yolov4 dan euclidean distance dalam mendekripsi pelanggaran social distancing,” Program Studi Sistem Informasi Universitas Hasanuddin, 2021.
- [70] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016.
- [71] J. Redmon and A. Farhadi, “Yolo9000: Better, faster, stronger,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016.
- [72] ——, “Yolov3: An incremental improvement,” 2018.
- [73] A. Bochkovskiy, C. Y. Wang, and H.-Y. M. Liao, “Yolov4: Optimal speed and accuracy of object detection,” 2020. [Online]. Available: <https://arxiv.org/abs/2004.10934>
- [74] G. Jocher, Ultralytics LLC, 2020. [Online]. Available: <https://github.com/ultralytics/yolov5>
- [75] D. Thuan, “Skripsi: Evolution of yolo algorithm and yolov5: The state-of-the-art object detection algorithm,” Information Technology Oulu University of Applied Sciences, 2021.
- [76] J. Nelson and J. Solawetz, “Responding to the controversy about yolov5,” Roboflow, Inc, June 2020. [Online]. Available: <https://github.com/ultralytics/ultralytics>
- [77] J. Solawetz and Francesco, “What is yolov8? the ultimate guide.” Roboflow, Inc, January 2023. [Online]. Available: <https://blog.roboflow.com/whats-new-in-yolov8/>
- [78] G. Jocher, Ultralytics LLC, 2023. [Online]. Available: <https://github.com/ultralytics/ultralytics>
- [79] T. O. Emmanuel, Kaggle, Google LLC, 2018. [Online]. Available: <https://www.kaggle.com/datasets/emmarex/plantdisease>
- [80] A. Ataiemontazer, Kaggle, Google LLC, 2020. [Online]. Available: <https://www.kaggle.com/datasets/abbasataiemontazer/potato-leaf>
- [81] R. Saeed, Kaggle, Google LLC, 2021. [Online]. Available: <https://www.kaggle.com/datasets/rizwan123456789/potato-disease-leaf-datasetpld>

## LAMPIRAN

### L.1 Source Code dan Model

#### L.1.1 YOLOv5m

```
1 #clone dan install YOLOv5
2 !git clone https://github.com/ultralytics/yolov5 # clone
3 cd yolov5
4 pip install -r requirements.txt # install
5 #melatih data dengan epoch 50
6 !python train.py --batch 16 --img 640 --epochs 50 --data
    custom_data.yaml --weights yolov5m.pt --nosave --cache
7 #menguji model dengan dataset pengujian
8 !python val.py --weights /content/drive/Shareddrives/Skripsi/
    Yolov5/result/epoch50/train/weights/best.pt --data
    custom_data.yaml --img 640 --task test
9 #memprediksi data baru
10 !python detect.py --weights runs/train/exp5/weights/best.pt --
    img 640 --conf 0.25 --source /content/drive/Shareddrives/
        Skripsi/Yolov5/testing
```

Tautan source code: <https://s.id/SourceCodeYOLOv5m440305>

Tautan folder model YOLOv5: <https://s.id/ModelYOLOv5m440305>

## L.1.2 YOLOv6m

```
1 #clone YOLOv6
2 !git clone https://github.com/meituan/YOLOv6.git
3 cd YOLOv6
4 #install YOLOv6
5 %pip install -r requirements.txt
6 #clone YOLOv6m
7 !wget https://github.com/meituan/YOLOv6/releases/download/0.2.0/
     yolov6m.pt
8 #install library addict
9 %pip install addict
10 #melatih data dengan epoch 50
11 !python tools/train.py --batch 16 --img 640 --epochs 50 --conf
    configs/yolov6m_finetune.py --data-path data/dataset.yaml --
    device 0
12 #menguji model dengan dataset pengujian
13 !python tools/eval.py --data data/dataset.yaml --batch 32 --
    weights /content/drive/Shareddrives/Skripsi/Yolov6/result/
    epoch50/train/weights/best_ckpt.pt --task test --do_pr_metric
    =True --do_coco_metric=False
14 #memprediksi dataset pengujian
15 !python tools/infer.py --weights /content/drive/Shareddrives/
    Skripsi/Yolov6/result/train50/weights/best_ckpt.pt --source /
    content/drive/Shareddrives/Skripsi/Yolov6/images/test --yaml
    /content/drive/Shareddrives/Skripsi/Yolov6/YOLOv6/data/
    dataset.yaml --save-dir /content/drive/Shareddrives/Skripsi/
    Yolov6/result --device 0
```

Tautan source code: <https://s.id/SourceCodeYOLOv6m440305>

Tautan folder model YOLOv6: <https://s.id/ModelYOLOv6m440305>

### L.1.3 YOLOv7

```
1 #clone YOLOv7
2 !git clone https://github.com/WongKinYiu/yolov7
3 #clone model yolov7
4 !wget https://github.com/WongKinYiu/yolov7/releases/download/v0
     .1/yolov7.pt
5 #melatih data dengan epoch 50
6 !python train.py --batch-size 16 --img 640 640 --hyp data/hyp.
     scratch.custom.yaml --cfg cfg/training/yolov7-custom.yaml --
     epochs 50 --data data/custom_data.yaml --weights yolov7.pt --
     name yolov7-custom
7 #menguji model dengan dataset pengujian
8 !python test.py --data data/custom_data.yaml --img 640 --device
     0 --weights runs/train/yolov7-custom5/weights/best.pt --name
     yolov7_640_test --task test
9 #memprediksi data baru
10 !python detect.py --weights runs/train/yolov7-custom5/weights/
     best.pt --img-size 640 --source /content/drive/Shareddrives/
      Skripsi/Yolov7/testing
```

Tautan source code: <https://s.id/SourceCodeYOLOv7440305>

Tautan folder model YOLOv7: <https://s.id/ModelYOLOv7440305>

#### L.1.4 YOLOv8m

```
1 #install package ultralytics
2 !pip install ultralytics -q
3 #import libraries YOLO
4 from ultralytics import YOLO
5 #install model YOLOv8m
6 model = YOLO("yolov8m.pt")
7 #melatih data dengan epoch 50
8 model.train(data = "/content/data.yaml", epochs = 50, batch =
16, imgsz = 640, dfl = 0.2)
9 #menguji model dengan dataset pengujian
10 model = YOLO("/content/drive/Shareddrives/Skripsi/Yolov8/result/
epoch50/train/best.pt")
11 model.val(data='/content/drive/Shareddrives/Skripsi/Yolov8/data.
yaml', split = 'test')
12 #memprediksi data dengan model
13 infer = YOLO("/content/drive/Shareddrives/Skripsi/Yolov8/result/
epoch50/train/best.pt")
14 #memprediksi data baru
15 infer.predict("/content/drive/Shareddrives/Skripsi/Yolov8/
testing", save = True, save_txt = True)
```

Tautan source code: <https://s.id/SourceCodeYOLOv8m440305>

### L.1.5 Hyperparameter

```
1 epoch: 50 #number of epoch
2 batch: 16 # size of batch
3 img_sz: 640 x 640 # size of image
4 opt: SGD #optimizer SGD
5 lr0: 0.01 # initial learning rate (SGD=1E-2, Adam=1E-3)
6 lrf: 0.01 # final OneCycleLR learning rate (lr0 * lrf)
7 momentum: 0.937 # SGD momentum/Adam beta1
8 weight_decay: 0.0005 # optimizer weight decay 5e-4
9 warmup_epochs: 3.0 # warmup epochs (fractions ok)
10 warmup_momentum: 0.8 # warmup initial momentum
11 warmup_bias_lr: 0.1 # warmup initial bias lr
12 hsv_h: 0.015 # image HSV-Hue augmentation (fraction)
13 hsv_s: 0.7 # image HSV-Saturation augmentation (fraction)
14 hsv_v: 0.4 # image HSV-Value augmentation (fraction)
15 degrees: 0.0 # image rotation (+/- deg)
16 translate: 0.1 # image translation (+/- fraction)
17 scale: 0.5 # image scale (+/- gain)
18 shear: 0.0 # image shear (+/- deg)
19 perspective: 0.0 # image perspective (+/- fraction), range
0–0.001
20 flipud: 0.0 # image flip up-down (probability)
21 filplr: 0.5 # image flip left-right (probability)
22 mosaic: 1.0 # image mosaic (probability)
23 mixup: 0.0 # image mixup (probability)
24 copy_paste: 0.0 # segment copy-paste (probability)
```

## L.2 Dataset dan Sampel Hasil Deteksi

Dataset: <https://s.id/Dataset440305>

Sampel Hasil Deteksi: <https://s.id/SampelHasilDeteksi440305>