MACHINE LEARNING OBJECT DETECTION TANAMAN OBAT SECARA REAL-TIME MENGGUNAKAN METODE YOLO (You Only Look Once)

TUGAS AKHIR

Disusun Oleh :

FIRMAN MAULANA
361601016

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TINGGI MANAJEMEN INFORMATIKA & KOMPUTER
INDONESIA MANDIRI
BANDUNG
2021

LEMBAR PENGESAHAN

MACHINE LEARNING OBJECT DETECTION TANAMAN OBAT SECARA REAL-TIME MENGGUNAKAN METODE YOLO (You Only Look Once)

MACHINE LEARNING OBJECT DETECTION PLANT REAL-TIME USING YOLO METHOD (You Only Look Once)

Oleh :
FIRMAN MAULANA
361601016

Tugas Akhir Ini Telah Diterima Dan Disahkan Untuk
Memenuhi Persyaratan Mencapai Gelar
Sarjana Teknik Informatika

Pada

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TINGGI MANAJEMEN INFORMATIKA & KOMPUTER
INDONESIA MANDIRI

Bandung, 15 Februari 2021
Disetujui oleh

Ketua Program Studi,	Dosen Pembimbing,
Chalifa Chazar, S.T., M.T.	**Chalifa Chazar, S.T., M.T.**
NIDN: 0421098704	NIDN: 0421098704

LEMBAR PERSETUJUAN REVISI TUGAS AKHIR

MACHINE LEARNING OBJECT DETECTION TANAMAN OBAT SECARA REAL-TIME MENGGUNAKAN METODE YOLO (You Only Look Once)

MACHINE LEARNING OBJECT DETECTION PLANT REAL-TIME USING YOLO METHOD (You Only Look Once)

Telah Melakukan Sidang Tugas Akhir pada Hari Selasa,23 Februari 2021
Dan Telah Melakukan Revisi Sesuai Dengan Masukan pada Saat
Sidang Tugas Akhir

Menyetujui,

No	Nama	Keterangan	Tanda Tangan
1	Chalifa Chazar, S.T., M.T.	Pembimbing	
2	Hendra Gunawan, S.T., M.Kom.	Penguji 1	
3	Novi Rukhviyanti, S.T., M.Si.	Penguji 2	

Bandung,23 Februari 2021

Mengetahui,

Ketua Program Studi,

Chalifa Chazar, S.T., M.T.

NIDN: 0421098704

SURAT PERNYATAAN

Dengan ini saya menyatakan bahwa:

1. Penelitian ini adalah asli dan belum pernah diajukan untuk mendapatkan gelar akademik, baik di Sekolah Tinggi Manajemen Informatika dan Komputer Indonesia Mandiri maupun perguruan tinggi lainnya,

2. Penelitian ini murni merupakan karya penelitian saya sendiri dan tidak menjiplak karya pihak lain. Dalam hal ada bantuan atau arahan dari pihak lain maka telah saya sebutkan identitas dan jenis bantuannya di dalam lembar ucapan terima kasih

3. Seandainya ada karya pihak lain yang ternyata memiliki kemiripan dengan karya saya ini, maka hal ini adalah di luar pengetahuan saya dan terjadi tanpa kesengajaan dari pihak saya

Pernyataan ini saya buat dengan sesungguhnya dan apabila di kemudian hari terbukti adanya kebohongan dalam pernyataan ini, maka saya bersedia menerima sanksi akademik sesuai norma yang berlaku di Sekolah Tinggi Manajemen Informatika dan Komputer Indonesia Mandiri.

Bandung, 23 Februari 2021

Yang Membuat Pernyataan,

Firman Maulana

361601016

ABSTRAK

Machine Learning atau pembelajaran mesin merupakan pendekatan dalam AI*(Artificial Intelligence)* yang banyak digunakan untuk menggantikan atau menirukan perilaku manusia untuk menyelesaikan masalah atau melakukan otomatisasi. Salah satu pemanfaatan *machine learning* adalah dalam *object detection* yang digunakan untuk mendeteksi objek, bagaimana membangun aplikasi yang dapat mendeteksi objek tanaman obat, dengan menggunakan metode YOLO(*You Only Look Once*) sistem pendeteksi objek *real-time*, tujuan dan manfaat penelitian ini adalah membangun aplikasi yang dapat mendeteksi objek tanaman obat menggunakan metode YOLO *(You Only Look Once),* dan Aplikasi ini dibangun menggunakan bahasa pemrograman *Python,* metode *Waterfall* dan pemodelan UML *(Unified Modeling Languange)*.

Kata kunci : *machine learning, object detection, yolo.*

ABSTRACT

Machine Learning or machine learning is an approach in AI (Artificial Intelligence) that is widely used to replace or mimic human behavior to solve problems or automate. One of the uses of machine learning is in object detection which is used to detect objects, how to build applications that can detect medicinal plant objects, using the YOLO (You Only Look Once) method of real-time object detection systems, the purpose and benefit of this research is to build applications. which can detect medicinal plant objects using the YOLO (You Only Look Once) method, and this application was built using the Python programming language, the Waterfall method and UML (Unified Modeling Language) modeling.

Keywords: machine learning, object detection, yolo.

UCAPAN TERIMAKASIH

Dengan mengucap syukur Alhamdulillah, penelitian ini dapat diselesaikan untuk memenuhi syarat tugas akhir. Laporan penelitian dalam tugas akhir ini di ajukan untuk memenuhi dan melengkapi salah satu akademik dalam kelulusan jenjang Strata Satu (S1) jurusan Teknik Informatika pada Sekolah Tinggi Manajemen Informatika dan komputer Indonesia Mandiri.

Penyusunan tugas akhir ini tidak lepas dari dukungan dan bimbingan dari berbagai pihak, maka pada kesempatan ini penulis ingin menyampaikan rasa terimakasih yang sebesar-besarnya kepada:

1. Ibu Chalifa Chazar, S.T., M.T. selaku Dosen pembimbing dan Ketua program studi Teknik Informatika STMIK-IM yang selalu melungkan waktu, fikiran dan tenaga dalam memberikan bimbingan, masukan dan saran-sarannya.
2. Bapak Dr. Chairuddin, M.T., M.M. selaku Ketua Sekolah Tinggi Manajemen Informatika dan Komputer Indonesia Mandiri (STMIK-IM).
3. Segenap Dosen, staf dan karyawan STMIK-IM yang telah mendidik dan membantu dalam proses studi berlangsung.
4. Teruntuk orang tua kandung penulis Bapak Endang Amar (alm) dan Ibu Tia Setiawati yang sangat penulis sayangi dan cintai, terima kasih selalu memberikan nasehat, dukungan, didikan, kasih sayang, serta Do'a yang penuh dan tulus.
5. Orang tua angkat penulis Bapa Dusa Sumartaya dan Atin Hafidiah serta keluarga besar nya Dikdik Maulana, terima kasih selalu mendukung, dan mendo'akan penulis.

6. Sahabat-sahabat penulis yaitu Sadnan Hafiz, Alvin, Dwi, Sanno, Agung ,Agri, Danu, Aswin yang sama-sama berjuang untuk menyelesaikan laporan penelitian tugas akhir ini.
7. Seluruh rekan-rekan di STMIK-IM angkatan 2016 khususnya pada program studi Teknik Informatika yang sama-sama berjuang untuk terus meraih impian,
8. Seluruh rekan, sahabat, dan pihak-pihak yang tidak dapat penulis sebutkan satu persatu yang telah membantu Penulis baik secara langsung maupun tidak langsung memberikan semangat kepada Penulis dalam menyelesaikan laporan penelitian tugas akhir ini.

Penulis menyadari bahwa masih banyak kekurangan yang mendasar pada laporan penelitian tugas akhir ini. Oleh karena itu penulis mengundang pembaca untuk memberikan saran serta kritik yang dapat membangun penulis. Penulis berharap adanya kritik konstruktif dan saran yang membangun dari semua pihak.

Akhir kata, saya berharap semoga dengan selesainya laporan penelitian Tugas Akhir ini dapat memberikan manfaat bagi semua pihak serta menambah wawasan bagi pemikiran kita semua. Terima kasih.

KATA PENGANTAR

Dengan mengucap Alhamdulillah, sebagai wujud syukur ke Hadirat Allah SWT, yang senantiasa memberikan rahmat dan karunia-NYA sehingga penulis dapat menyelesaikan laporan penelitian tugas akhir ini dengan baik dan tepat pada waktunya.

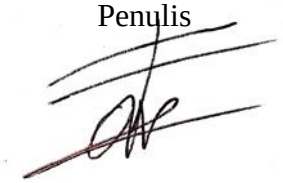
Penelitian ini, berjudul *MACHINE LEARNING OBJECT DETECTION TANAMAN OBAT SECARA REAL-TIME MENGGUNAKAN METODE YOLO (You Only Look Once)*, disusun untuk melengkapi tahapan akhir studi yang dijalani di Sekolah Tinggi Manajemen Informatika dan Komputer Indonesia Mandiri.

Penelitian ini berisi mengenai perancangan sebuah aplikasi object detection yang dapat menghasilkan nama tanaman obat dengan mendeteksi tanaman obat , yang meliputi analisis dari sistem yang diusulkan dengan harapan dapat mengatasi masalah yang di identifikasi.

Dengan segala keterbatasa tentunya diharapkan aplikasi ini dapat bermanfaat bagi berbagai pihak, khususnya bagi penulis sendiri.

Bandung, 10 Oktober 2020

Penulis



Firman Maulana

361601016

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I

PENDAHULUAN

1.1. Latar Belakang

Machine Learning atau pembelajaran mesin merupakan pendekatan dalam AI *(Artificial Intelligence)* yang banyak digunakan untuk menggantikan atau menirukan perilaku manusia untuk menyelesaikan masalah atau melakukan otomatisasi. Sesuai namanya, *machine learning* mencoba menirukan bagaimana proses manusia atau makhluk cerdas belajar dan mengeneralisasi. Setidaknya ada dua aplikasi utama dalam *machine learning* yaitu, klasifikasi dan prediksi. Ciri khas dari *machine learning* adalah adanya proses pelatihan, pembelajaran, atau training. Oleh karena itu, *machine learning* membutuhkan data untuk dipelajari yang disebut sebagai data training. Klasifikasi adalah metode dalam *machine learning* yang digunakan oleh mesin untuk memilah atau mengklasifikasikan obyek berdasarkan ciri tertentu sebagaimana manusia mencoba membedakan benda satu dengan yang lain(Ahmad, 2017).

Dua prosedur pembelajaran mesin telah diselidiki secara mendetail menggunakan permainan catur. Pekerjaan yang cukup telah dilakukan untuk memverifikasi fakta bahwa komputer dapat diprogram sehingga akan belajar memainkan permainan catur yang lebih baik daripada yang dapat dimainkan oleh orang yang menulis program tersebut. Lebih jauh lagi, ia dapat belajar melakukan ini dalam waktu yang sangat singkat (8 atau 10 jam waktu bermain mesin) ketika hanya diberi aturan permainan, rasa arah, dan daftar parameter yang berlebihan

dan tidak lengkap yang dianggap ada hubungannya dengan permainan, tetapi tanda dan bobot relatifnya yang benar tidak diketahui dan tidak ditentukan. Prinsip pembelajaran mesin yang diverifikasi oleh eksperimen ini, tentu saja, berlaku untuk banyak situasi lain(Samuel, 2000).

Object detection (pendeteksian objek) baru-baru ini menjadi salah satu bidang yang paling menarik dalam *computer vision* dan *artificial intelligence*. Pendeteksian objek merupakan teknologi komputer yang berkaitan dengan *computer vision* dan *image processing* yang berhubungan dengan mendeteksi suatu objek dalam citra digital yang dapat berupa warna dan bentuk objek(Dewi, 2018).

Banyak tanaman obat akan tetapi banyak masyarakat yang belum mengetahui jenis tanaman obat, dengan *object detection* dan *machine learning* bisa dimanfaatkan untuk mendeteksi tanaman obat, untuk mendeteksi permasalahanya butuh keakuratan yang baik ada beberapa metode yang bisa digunakan untuk mendeteksi YOLO dipilih karena memiliki keunggulan.

YOLO *(You Only Look Once)*, metode deteksi objek 2D, sangat cepat karena jaringan saraf tunggal memprediksi kotak dan probabilitas kelas yang terikat langsung dari gambar penuh dalam satu evaluasi. Namun, itu membuat lebih banyak kesalahan lokalisasi dan kecepatan pelatihannya relatif lambat. Mengambil manfaat dari pemikiran pusat cluster dalam segmentasi super-pixel dan kotak jangkar di *Faster R-CNN*, dalam penelitian ini, kami mengusulkan metode YOLO *(You Only Look Once)* yang dimodifikasi. Pertama, kami mengganti YOLO *(You Only Look Once)* yang sepenuhnya terhubung ke lapisan konvolusional, di mana kotak *cluster* (beberapa kotak jangkar berpusat pada pusat cluster) dapat sepenuhnya menutupi seluruh gambar pada awal pelatihan. Sebagai

hasilnya, struktur baru dapat mempercepat proses pelatihan. Hasil percobaan menunjukkan bahwa YOLO *(You Only Look Once),* meningkatkan akurasi lokalisasi sekitar 10%, kecepatan konvergensi proses pelatihan juga meningkat(Zhao, Xia Jia and Ni, 2018).

Untuk membangun aplikasi ini digunakan bahasa pemrograman *python* dengan Extensi *IDE PyCharm, Google Colab,* memanfaatkan *framework darknet* bahasa C dan *OpenCV*, untuk meningkatkan efisiensi dan akurasi *object detection*

Dengan menggunakan metode YOLO *(You Only Look Once)* pada *object detection* diharapkan dapat mengatasi kesulitan pengklasifikasian *object* Tanaman Obat yang dihasilkan nanti. Maka judul penelitian yang diambil adalah **"*MACHINE LEARNING OBJECT DETECTION TANAMAN OBAT SECARA REAL-TIME MENGGUNAKAN METODE YOLO (You Only Look Once)"***

1.2. Indentifikasi Masalah

Berdasarkan latar belakang di atas, maka indentifikasi masalah di penelitian ini bagaimana membangun aplikasi yang dapat mendeteksi objek Tanaman Obat menggunakan metode YOLO *(You Only Look Once),* serta bagaimana memberikan informasi kepada masyarakat dalam mengenali tanaman obat.

1.3. Tujuan dan Manfaat

Tujuan dan manfaat penelitian ini adalah membangun aplikasi yang dapat mendeteksi objek Tanaman Obat menggunakan metode YOLO *(You Only Look Once),* serta memberikan informasi kepada masyarakat dalam mengenali tanaman obat.

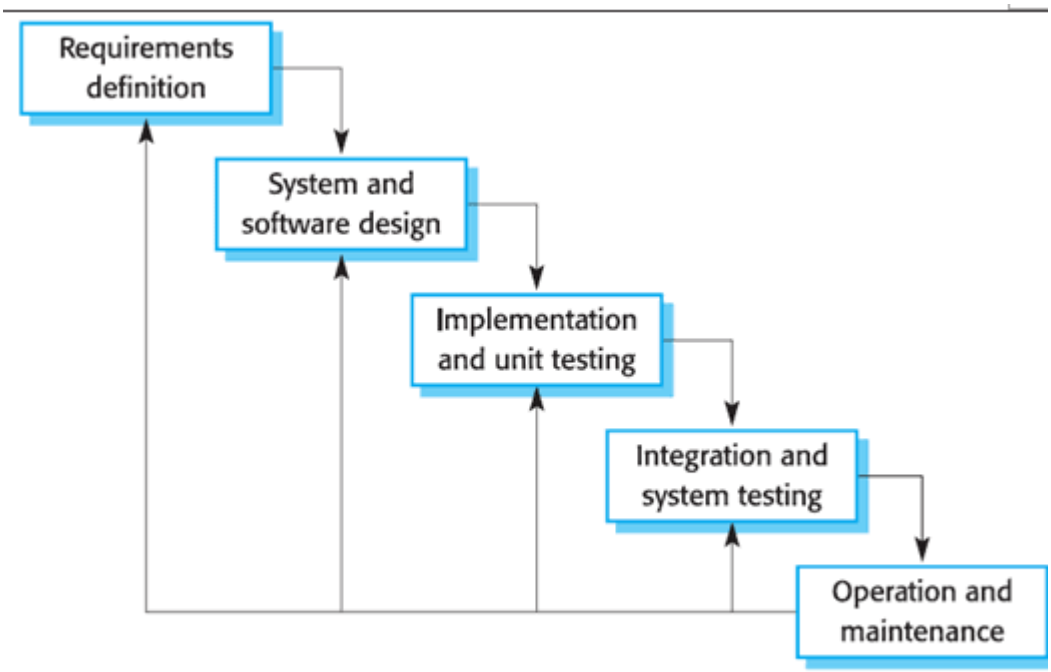
1.4. Batasan Masalah

Berdasarkan latar belakang masalah serta dengan memperhatikan keterbatasan waktu, tenaga, biaya dan kemampuan, maka permasalahan dibatasi pada hal-hal sebagai berikut :

1. Aplikasi ini dibuat menggunakan bahasa pemrograman *Python* 3.8 versi *Windows Operating System 64bit*.
2. *Training dataset* menggunakan *google colab.*
3. Aplikasi untuk *object detection* dijalankan melalui konsol *IDE PyCharm.*
4. Penelitian ini dilakukan hanya sampai proses *Penerapan / Pengujian Program (Integration & Testing)* (tidak melakukan proses Pemeliharaan *(Operation & Maintenance)*).
5. Tanaman Obat yang akan dideteksi menggunakan kamera yang terdapat pada laptop..
6. *Dataset* Tanaman Obat yang digunakan hanya mengambil 5 Tanaman Obat.
7. Aplikasi tidak menggunakan *database* melainkan menggunakan *data training.*

1.5. Metode Penelitian

Metode penelitian dalam penelitian ini mengacu pada SDLC (*Systems Development Life Cycle*), metode SDLC yang digunakan adalah Metode *Waterfall*.



Gambar 01. Metode Waterfall(Tristianto, 2018).

Tristianto menyatakan bahwa model *waterfall* memiliki tahapan-tahapan sebagai berikut:

a. *Requirement definition* (analisis kebutuhan)

Dalam langakah ini merupakan analisa terhadap kebutuhan sistem. Pengumpulan data dalam tahap ini bisa melakukan sebuah penelitian, wawancara atau study literatur. Seseorang system analisis akan menggali informasi sebanyak-banyaknya dari user sehingga akan tercipta sebuah sistem komputer yang bisa melakukan tugas-tugas yang diinginkan oleh user tersebut. Tahapan ini akan menghasilkan dokumen user requirement atau bisa dikatakan sebagai data yang berhubungan dengan keinginan user dalam pembuatan sistem. Dokumen inilah yang akan menjadi

acuan system analisis untuk menterjemahkan kedalam bahasa pemrograman(Tristianto, 2018)

b. *System and software design* (design sistem)

Proses design akan menterjemahkan syarat kebutuhan kesebuah perancangan perangkat lunak yang dapat diperkirakan sebelum dibuat koding. Proses ini berfokus pada : struktur data, arsitektur perangkat lunak, representasi interface, dan detail (algoritma) prosedural. Tahapan ini akan menghasilkan dokumen yang disebut software requirement. Dokumen inilah yang akan digunakan programmer untuk melakukan aktivitas pembuatan sistemnya(Tristianto, 2018).

c. *Implementation and unit testing* (penulisan sinkode program / implemention)

Coding merupakan penerjemahan design dalam bahasa yang bisa dikenali oleh komputer. Dilakukan oleh programmer yang akan meterjemahkan transaksi yang diminta oleh user. Tahapan inilah yang merupakan tahapan secara nyata dalam mengerjakan suatu sistem.Dalam artian penggunaan computer akan dimaksimalkan dalam tahapan ini. Setelah pengkodean selesai maka akan dilakukan testing terhadap sistem yang telah dibuat tadi. Tujuan testing adalah menemukan kesalahankesalahan terhadap system tersebut dan kemudian bisa diperbaiki(Tristianto, 2018).

d. *Intergration and system testing* (Penerapan / Pengujian Program)

Tahapan ini bisa dikatakan final dalam pembuatan sebuah sistem. Setelah melakukan analisa, design dan pengkodean maka sistem yang sudah jadikan digunakan oleh user(Tristianto, 2018).

e. *Operation and Maintenance* (Operasi dan Pemilaharaan)

Perangkat lunak yang susah disampaikan kepada pelanggan pasti akan mengalami perubahan. Perubahan tersebut bisa karena mengalami kesalahan karena perangkat lunak harus menyesuaikan dengan lingkungan (periperal atau system operasi baru) baru, atau karena pelanggan membutuhkan perkembangan fungsional (Tristianto, 2018).

Dalam penelitian ini hanya akan dilakukan hingga tahap empat sehingga tahap penelitian ini tidak dilakukan dan menjadi batasan masalah.

1.6. Sistematika Penulisan

Sistematika penulisan yang digunakan dalam penelitian ini terbagi dalam beberapa pokok bahasan, yaitu:

BAB I : PENDAHULUAN

Bab ini berisi latar belakang, indentifikasi masalah, tujuan dan manfaat, metode penelitian dan sistematika penulisan penelitian mengenai Deteksi Objek Tanaman Obat Dengan YOLO *(You Only Look Once).*

BAB II : LANDASAN TEORI

Bab ini mengemukakan dasar-dasar teori, dan kumpulan studi pustaka yang berhubungan dengan topik penelitian yang

digunakan untuk perancangan dan pembangunan aplikasi Deteksi Objek Tanaman Obat Dengan YOLO *(You Only Look Once).*

BAB III : PEMBAHASAN

Bab ini membahas mengenai langkah-langkah yang dilakukan pada penelitian. Penyelesaian masalah tersebut diawali dengan pengumpulan *dataset* Tanaman Obat, Anotasi citra, Klasifikasi dengan YOLO, *Training* dan prediksi.

BAB IV : UJI COBA DAN IMPLEMENTASI

Dalam bab ini membahas tentang uji coba dan implementasi aplikasi dengan menggunakan *Black box testing*.

BAB V : KESIMPULAN DAN SARAN

Penutup berisi tentang kesimpulan dari penulisan penelitian dan saran – saran untuk pengembangan selanjutnya.

BAB II

LANDASAN TEORI

2.1. Tanaman Obat

Menurut Saudah,Vera Viena, dan Ernilasari dalam jurnalnya yang berjudul *Eksplorasi Spesies Tumbuhan Berkhasiat Obat Berbasis Pengetahuan Lokal Di Kabupaten Pidie* menjelaskan Tanaman Obat adalah Obat yang alami yang di dapatkan dari alam dengan berbagai jenis dan spesies, Tanaman Obat dapat di olah langsung dengan cara tradisional(Saudah, Viena and Ernilasari, 2019).

2.2. Kecerdasan Buatan (*Artificial Intelligence*)

Menurut Widodo Budiharto, dalam bukunya *Ai For Beginner* menjelaskan Kecerdasan buatan hadir sebagai cabang ilmu dari Computer Science yang menjanjikan banyak manfaat dalam menjawab kebutuhan manusia di masa depan. Literatur mengenai kecerdasan buatan menyebutkan bahwa ide mengenai kecerdasan buatan diawali pada awal abad 17 ketika Rene Descartes mengemukakan bahwa tubuh hewan bukanlah apa-apa melainkan hanya mesin-mesin yang rumit(Budiharto, 2018).

Kata "*intelligence*" sendiri berasal dari bahasa Latin "*intelligo*" yang berarti "saya paham". Berarti dasar dari intelligence ialah kemampuan untuk memahami dan melakukan aksi. *Intelligence* merupakan istilah yang komplek yang dapat didefinisikan dengan ungkapan yang berbeda seperti logika,

pemahaman, *self-awareness*, pembelajaran, perencanaan, dan problem solving. Sedangkan "*Artificial*" adalah sesuatu yang tidak nyata, seperti tipuan karena merupakan hasil simulasi(Budiharto, 2018).

2.2.1. Pengenalan Kecerdasan Buatan (*Artificial Intelligence*)

Bidang ilmu yang di pelajari dalam bidang AI (*Artificial Intelligence*) menurut Widodo Budiharto dan Derwin Suhartono diantaranya yaitu:

1. *Computer Vision*

   Cabang ilmu ini erat kaitannya dengan pembangunan arti/makna dari foto ke obyek secara fisik. Yang dibutuhkan didalamnya adalah metode-metode untuk memperoleh, melakukan proses, menganalisa dan memahami foto.

2. *Game Playing*

   Game yang membuat *non-player* memiliki strategi yang cerdas untuk mengalahkan *player*.

3. Sistem Pakar

   Bidang ilmu ini mempelajari bagaimana membangun sistem atau komputer yang memiliki keahlian untuk memecahkan masalah dan menggunakan penalaran dengan meniru atau mengadopsi keahlian yang dimiliki oleh pakar.

4. *Speech Recognition*

   Speech recognition merupakan mesin untuk memberikan instruksi pada beberapa komputer dengan menggunakan suara.

5. *Machine Learning*

   Merupakan mesin yang fokus pada pengembangan sebuah sistem yang mampu belajar sendiri tanpa harus berulang kali di program oleh manusia.

2.3. *Machine Learning*

Menurut Jan Wira Gotama Putra, dalam bukunya yang berjudul Pengenalan Konsep Pembelajaran Mesin dan Deep Learning, *Machine Learning* yaitu teknik untuk melakukan inferensi terhadap data dengan pendekatan matematis. Inti *machine learning* adalah untuk membuat model (matematis) yang mereflesikan pola-pola data(Jan and Gotama, 2019). Dalam *Machine Learning* ada beberapa cabang yang di kategorikan sebagai berikut:

1. *Supervised Learning*

   Supervised Learning adalah pembelajaran tearah/terawasi. Artinya, pada pembelajaran ini, ada guru yang mengajar (mengarahkan) dan siswa yang diajar. Kita disini berperan sebagai guru, ke- mudian mesin berperan sebagai siswa. Dalam *Supervised Learning* dibagi menjadi 2 kategori fungsi yaitu:

   a. Regresi dan Klasifikasi

      Pada persoalan regresi, kita ingin memprediksi output berupa bilangan kontinu. Misalnya pada regresi suatu fungsi polinomial, kita ingin mencari tahu fungsi f (x) diberikan data {(x1, y1), • • • , (xN , yN)}. Setelah itu, kita gunakan fungsi aproksimasi untuk mencari tahu nilai yN+1 dari data baru xN+1.

Perbedaan regresi dan klasifikasi adalah pada tipe output. Untuk regresi, tipe output adalah nilai kontinu, sementara tipe output pada persoalan klasifikasi adalah suatu objek pada himpunan (i.e., memilih opsi pada himpunan jawaban). Tetapi, kita dapat mengkonversi fungsi regresi menjadi fungsi klasifikasi

2. *Semi-supervised Learning*

Semi-supervised learning mirip dengan *supervised learning*, bedanya pada proses pelabelan data. Pada *supervised learning*, ada "guru" yang harus membuat "kunci jawaban" input-output. Sedangkan pada *semi-supervised learning* tidak ada "kunci jawaban" *eksplisit* yang harus dibuat guru. Kunci jawaban ini dapat diperoleh secara otomatis (misal dari hasil *clustering*). Pada kategori pembelajaran ini, umumnya kita hanya memiliki sedikit data. Kita kemudian menciptakan data tambahan baik menggunakan *supervise*d ataupun *unsupervised learning*, kemudian membuat model belajar dari data tambahan tersebut.

3. *Unsupervised Learning*

Jika pada supervised learning ada guru yang mengajar, maka pada *unsupervised learning* tidak ada guru yang mengajar. Contoh permasalahan *unsupervised learning* adalah clustering. Clustering adalah salah satu bentuk *unsupervised learning* ; yaitu salah satu hasil inferensi persamaan(Jan and Gotama, 2019).

2.4. YOLO (*You Only Look Once*)

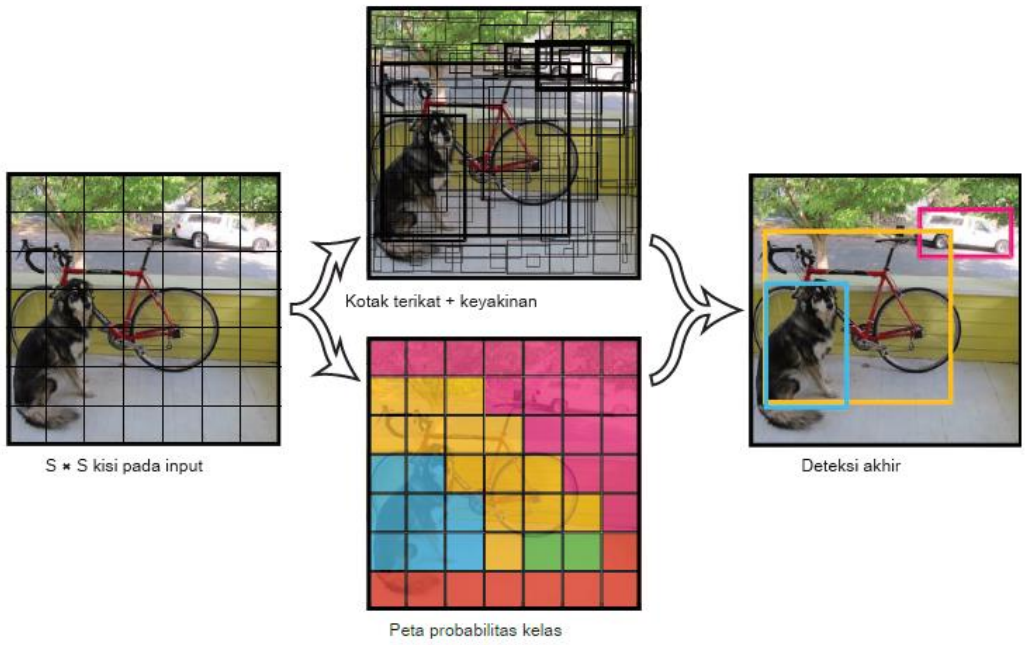
YOLO mempelajari representasi objek yang dapat digeneralisasikan. Ketika dilatih tentang gambar alami dan diuji pada karya seni, YOLO mengungguli metode deteksi top seperti DPM dan R-CNN dengan margin lebar. Karena YOLO

sangat dapat digeneralisasikan, kecil kemungkinannya untuk mogok ketika diterapkan pada domain baru atau input yang tidak terduga. YOLO masih ketinggalan dalam akurasi sistem deteksi canggih. Meskipun dapat dengan cepat mengidentifikasi objek dalam gambar, ia berjuang untuk secara tepat melokalisasi beberapa objek, terutama yang kecil. Kami memeriksa pengorbanan ini lebih lanjut dalam percobaan kami(Joseph Redmon , Santosh Divvala, 2016).

GAMBAR: 2.1. Sistem Deteksi YOLO(Redmon , Divvala, 2016).

Sistem Deteksi YOLO Memproses gambar dengan YOLO sederhana dan mudah. Sistem kami (1) mengubah ukuran gambar input menjadi 448 × 448, (2) menjalankan jaringan konvolusional tunggal pada gambar, dan (3) membatasi deteksi yang dihasilkan oleh kepercayaan model.

Untuk mengevaluasi YOLO pada PASCAL VOC, kami gunakan S = 7, B = 2. P ASCAL VOC memiliki 20 kelas berlabel demikian C = 20. Prediksi terakhir kami adalah a 7 × **7** × 30 *tensor***.**



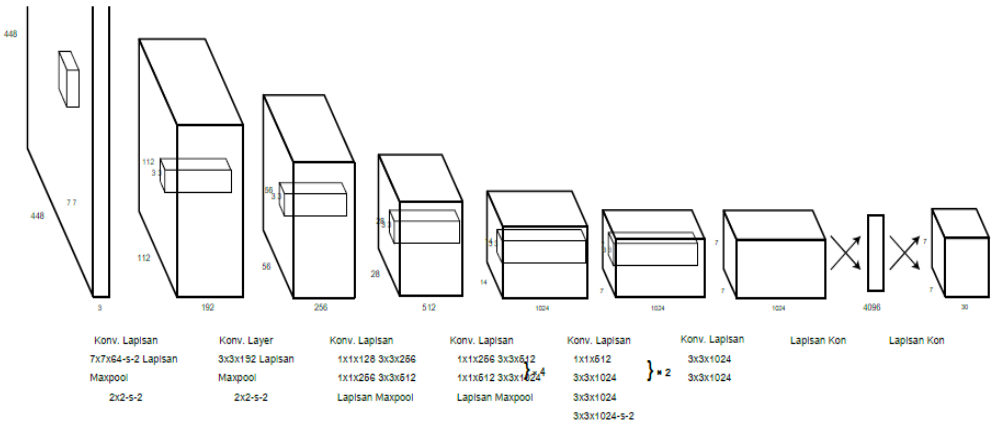
Gambar: 2.2. Model. Sistem(Redmon , Divvala, 2016).

Model Sistem ini mendeteksi model sebagai masalah regresi. Ini membagi gambar menjadi S × S kisi dan untuk setiap sel kisi memprediksi B kotak pembatas, kepercayaan diri untuk kotak-kotak itu, dan C probabilitas kelas. Prediksi ini dikodekan sebagai

S × S × (B ∗ 5 + C) tensor.

2.4.1. Jaringan

Sebagai jaringan saraf *convolutional* dan mengevaluasinya pada PASCAL Dataset pendeteksian VOC Lapisan konvolusional awal dari jaringan mengekstraksi fitur dari gambar sementara lapisan yang terhubung sepenuhnya memprediksi probabilitas dan koordinat keluaran. Arsitektur jaringan kami terinspirasi oleh model *GoogLeNet* untuk klasifikasi gambar Jaringan kami memiliki 24 lapisan konvolusional diikuti oleh 2 lapisan yang terhubung sepenuhnya. Alih-alih modul awal yang digunakan oleh *GoogLeNet*, kami hanya menggunakan 1 × 1 lapisan reduksi diikuti oleh 3 × 3 lapisan konvolusional, mirip dengan *Lin et al* Jaringan penuh ditunjukkan pada Gambar 2.3.



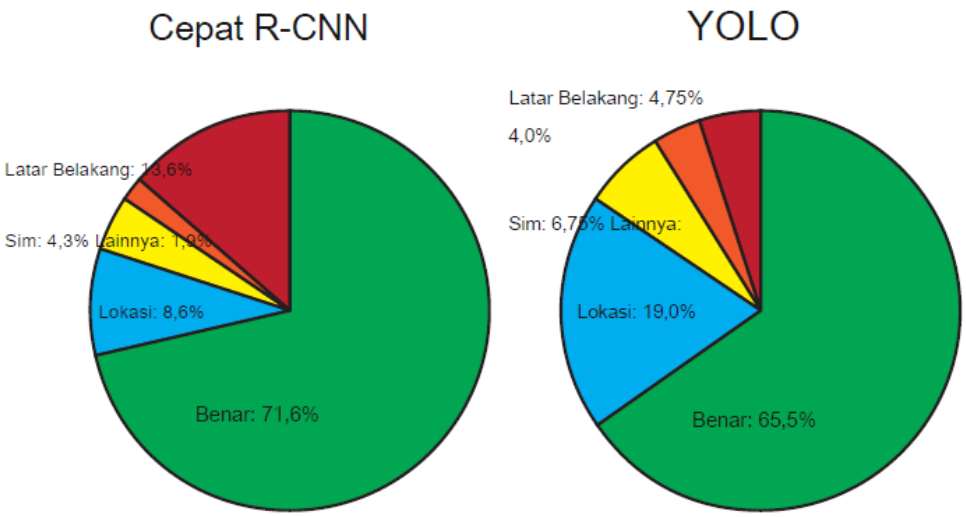
Gambar : 2.3. Arsitektur YOLO(Redmon , Divvala, 2016).

Jaringan deteksi memiliki 24 lapisan konvolusional diikuti oleh 2 lapisan yang terhubung sepenuhnya. Bergantian 1 × 1 lapisan konvolusional mengurangi ruang fitur dari lapisan sebelumnya. Kami mendahului lapisan konvolusional pada

tugas klasifikasi *ImageNet* di setengah resolusi (224 × 224 masukan gambar) lalu gandakan resolusi untuk deteksi(Redmon , Divvala, 2016).

2.4.2. Keterbatasan YOLO

YOLO memberikan batasan spasial yang kuat pada prediksi kotak pembatas karena setiap sel *grid* hanya memprediksi dua kotak dan hanya dapat memiliki satu kelas. Batasan spasial ini membatasi jumlah objek terdekat yang dapat diprediksi oleh model ini. Model ini berjuang dengan benda-benda kecil yang muncul dalam kelompok, seperti flok burung. Karena model kami belajar untuk memprediksi kotak pembatas dari data, ia berjuang untuk menggeneralisasi objek dalam rasio atau konfigurasi aspek baru atau tidak biasa. Model kami juga menggunakan fitur yang relatif kasar untuk memprediksi kotak pembatas karena arsitektur ini memiliki beberapa lapisan downsampling dari gambar input. Akhirnya, sementara melatih fungsi kerugian yang mendekati kinerja deteksi, fungsi kerugian kami memperlakukan kesalahan yang sama dalam kotak pembatas kecil versus kotak pembatas besar. Kesalahan kecil di kotak besar umumnya jinak tetapi kesalahan kecil di kotak kecil memiliki efek yang jauh lebih besar pada IOU(Redmon , Divvala, 2016).



Gambar : 2.4. Analisis Kesalahan(Redmon , Divvala, 2016).

Fast R-CNN vs YOLO Bagan ini menunjukkan persentase kesalahan lokalisasi dan latar belakang di deteksi N atas untuk berbagai kategori (N = objek # dalam kategori itu).

• Lainnya: kelas salah, IOU> . 1

• Latar belakang: IOU < . 1 untuk objek apa pun

Angka 4 menunjukkan rincian setiap jenis kesalahan yang dirata-rata disemua 20 kelas.

YOLO berjuang untuk melokalkan objek dengan benar. Akun kesalahan lokalisasi

untuk lebih banyak kesalahan *YOLO* daripada semua sumber lain digabungkan. *Fast R-CNN* membuat lebih sedikit kesalahan lokalisasi tetapi jauh lebih banyak

kesalahan latar belakang. 13,6% dari deteksi atasnya adalah positif palsu yang tidak mengandung benda apa pun. *Fast R-CNN* hampir 3x lebih mungkin untuk memprediksi deteksi latar belakang daripada *YOLO*(Redmon , Divvala, 2016).

2.5. *Python*

Menurut Dini Triasanti dalam bukunya, Konsep Dasar *Python* merupakan bahasa pemrograman yang freeware atau perangkat bebas dalam arti sebenarnya, tidak ada batasan dalam penyalinannya atau mendistribusikannya. Lengkap dengan *source code* nya, *debugger* dan *profiler*, antarmuka yang terkandung di dalamnya untuk pelayanan antarmuka, fungsi sistem, GUI (antarmuka pengguna grafis), dan basis datanya(Triasanti, 2000).

2.5.1. Aplikasi bahasa *phyton*

a. Perangkat bantu shell. Tugas-tugas sistem administrator, program baris perintah.
b. Kerja bahasa ekstensi. Antarmuka untuk pustaka C/C++, kustomisasi.
c. Pembuatan *prototipe* secara cepat/pembuatan sistem aplikasi. *Prototipe* yang dapat dibuang atau sesuai dengan permintaan.
d. Modul berdasarkan bahasa pemrograman. Pengganti dari penulisan parser khusus.
e. Antarmuka pengguna grafis. Penggunaan *GUI API* sederhana dan canggih.
f. Pengaksesan basisdata. Penyimpanan objek tetap, antarmuka sistem *SQL*.
g. Pemrograman terdistribusi. Penggunaan *API* mekanisme *client/server* terintegrasi.

h. Skrip internet. Skrip *CGI*, antarmuka *HTTP*, *Aplet WWW*, dan lainnya.

2.5.2. Mengapa bahasa *phyton*

Sisi utama yang membedakan *Python* dengan bahasa lain adalah dalam hal aturan penulisan kode program. Bagi para *programmer* di luar *python* siap-siap dibingungkan dengan aturan indentasi, tipe data, tuple, dan *dictionary*. *Python* memiliki kelebihan tersendiri dibandingkan dengan bahasa lain terutama dalam hal penanganan modul, ini yang membuat beberapa *programmer* menyukai *python*. Selain itu *python* merupakan salah satu produk yang *open source*, *free*, dan *multiplatform*(Triasanti, 2000).

Beberapa fitur yang dimiliki *Python* adalah:

a. memiliki kepustakaan yang luas; dalam distribusi *Python* telah disediakan modul-modul siap pakai untuk berbagai keperluan.
b. memiliki tata bahasa yang jernih dan mudah dipelajari. memiliki aturan *layout* kode sumber yang memudahkan pengecekan, pembacaan kembali dan penulisan ulang kode sumber. berorientasi obyek.
c. memiliki sistem pengelolaan memori otomatis *(garbage collection*, seperti *java*) modular, mudah dikembangkan dengan menciptakan modul-modul baru; modul-modul tersebut dapat dibangun dengan bahasa *Python* maupun C/C++.
d. memiliki fasilitas pengumpulan sampah otomatis, seperti halnya pada bahasa pemrograman *Java*,

e. *python* memiliki fasilitas pengaturan penggunaan ingatan komputer sehingga para pemrogram tidak perlu melakukan pengaturan ingatan komputer secara langsung.

2.5. *Object Detection*

Object Detection adalah bagian besar dari kehidupan orang-orang. Kita, sebagai manusia, terus-menerus "mendeteksi" berbagai benda seperti orang, bangunan, dan mobil. Namun itu tetap menjadi misteri bagaimana kita mendeteksi objek secara akurat dan dengan sedikit usaha yang jelas. Penjelasan komperhensif telah menentang psikolog dan fisiolog selama lebih dari satu abad(Schneiderman, Henry, Kanade, 2000).

Object Detection memiliki banyak kegunaan potensial termasuk pengambilan gambar. Koleksi gambar digital telah tumbuh secara dramatis ditahun terakhir. Corbis memperkirakan memiliki lebih dari 67 juta gambar dalam koleksinya. *The Associated Press* mengumpulkan dan mengarsipkan sekitar 1.000 foto sehari. Saat ini, kegunaan dari koleksi ini dibatasi oleh kurangnya metode *retrieval* yang efektif. Untuk menemukan gambar tertentu dalam koleksi semacam itu, orang harus mencari menggunakan teks berbasis keterangan dan fitur gambar primitif seperti warna dan tekstur(Schneiderman, Henry, Kanade, 2000).

Object Detection dapat digunakan untuk mengekstraksi lebih banyak informasi dari gambar-gambar dan membantu memberi label dan mengkategorikannya. Metode pencarian yang ditingkatkan akan membuat database dapat diakses oleh kelompok pengguna yang lebih luas, seperti lembaga penegak hukum, praktisi medis, desainer grafis dan *multimedia*, dan seniman.

Object Detection juga bisa berguna dalam fotografi. Ketika teknologi kamera berubah dari film ke capture digital, kamera akan menjadi bagian optik dan bagian komputer(Schneiderman, Henry, Kanade, 2000).

2.6. *Website*

Website (web, site, situs) adalah tempat sentral dimana web pages (halaman) disimpan. Halaman tersebut mengandung konten atau isi dari website. Home page (laman) adalah halaman utama dimana seluruh konten saling terhubung. *URL* (*Uniform Resource Locator, www*) adalah alamat *website*.

WWW ini menggunakan satu berkas yang disebut *hypertext*. Berkas *hypertext* ini merupakan suatu berkas dengan menggunakan karakter ASCII dan berisi gabungan perintah dan dokumen. Pada berkas *hypertext* WWW (HTML) selain perintah untuk mengatur *style* (pola tulisan) juga terdapat perintah-perintah untuk menghubungkan satu HTML dengan HTML lainnya melalui kata kunci sebagai pointer (*link*) *Link* pada WWW ini dapat berupa lokasi berkas, berkas lain ataupun berkas pada server WWW lainnya. Link dalam satu berkas umumnya digunakan untuk mempercepat menampilkan keterangan yang bersesuaian dengan kata kunci tersebut(Ika Atman Satya, 1995).

2.7. UML *(Unified Modeling Language)*

Unified Modelling Language (UML) adalah bahasa pemodelan untuk sistem atau perangkat lunak yang berparadigma berorientasi objek. Abstraksi konsep dasar UML terdiri dari structural *classification*, *dynamic behavior*, dan model *management* dapat kita pahami main *concepts* sebagai *term* yang akan muncul pada saat membuat diagram dan *view* adalah kategori dari diagram

tersebut. UML mendefinisikan diagram-diagram sebagai *Use case diagram*, *Class diagram, Statechart diagram, Activity diagram,Sequence diagram, Collaboration diagram, Component diagram,* dan *Deployment diagram*(Adi Nugroho, 2010).

Unified Modeling Language (UML) adalah sebuah bahasa yang berdasarkan grafik atau gambar untuk memvisualisasi, menspesifikasikan, membangun, dan pendokumentasian dari sebuah sistem pengembangan *software* berbasis OO (*Object-Oriented*). UML sendiri juga memberikan standar penulisan sebuah sistem *blue print*, yang meliputi konsep bisnis proses, penulisan kelas-kelas dalam bahasa program yang spesifik, skema *database*, dan komponen-komponen yang diperlukan dalam sistem *software*.

Diagram *Unified Modelling Language (*UML*)* antara lain sebagai berikut:

1. *Use Case Diagram*

   Use case menggambarkan *external view* dari sistem yang akan kita buat modelnya . *Use case* harus mampu mengambarkan urutan aktor yang menghasilkan nilai terukur(Widodo, Prabowo.P, 2011).

   Model *use case* dapat dijabarkan dalam diagram *use case*, tetapi perlu diingat, diagram tidak indetik dengan model karena model lebih luas dari diagram(Poole, 2001).

2. *Class Diagram*

   Kelas sebagai suatu set objek yang memiliki atribut dan perilaku yang sama, kelas kadang disebut kelas objek(Jeffery L. dkk, 2006).

Class memiliki tiga area pokok yaitu : 1) Nama, kelas harus mempunyai sebuah nama. 2) *Atribut*, adalah kelengkapan yang melekat pada kelas. Nilai dari

suatu kelas hanya bisa diproses sebatas atribut yang dimiliki. 3) Operasi, adalah proses yang dapat dilakukan oleh sebuah kelas, baik pada kelas itu sendiri ataupun kepada kelas lainnya(Jeffery L. dkk, 2006).

3. *Activity Diagram*

   Diagram activity menunjukkan aktivitas sistem dalam bentuk kumpulan aksi-aksi, bagaimana masing-masing aksi tersebut dimulai, keputusan yang mungkin terjadi hingga berakhirnya aksi. *Activity diagram* juga dapat menggambarkan proses lebih dari satu aksi salam waktu bersamaan. *Diagram activity* adalah aktifitas-aktifitas, objek, state, transisi state dan event. Dengan kata lain kegiatan *diagram* alur kerja menggambarkan perilaku sistem untuk aktivitas(Haviluddin, 2011).

4. *Sequence Diagram*

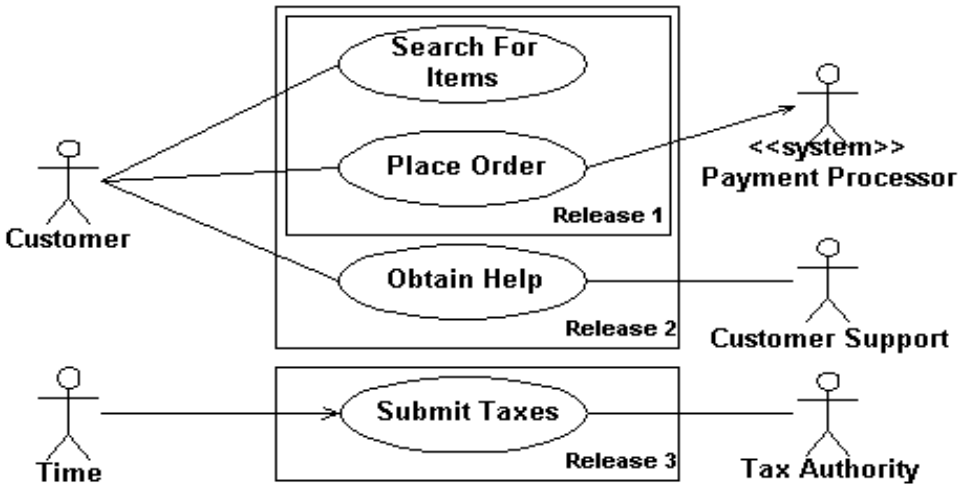
   Secara mudahnya *sequence diagram* adalah gambaran tahap demi tahap, termasuk kronologi (urutan) perubahan secara logis yang seharusnya dilakukan untuk menghasilkan sesuatu sesuai dengan us*e case diagram*(Haviluddin, 2011).

Use Case Diagram yang menggambarkan *actor, use case* dan relasinya sebagai suatu urutan tindakan yang memberikan nilai terukur untuk aktor. Sebuah *use case* digambarkan sebagai *elips horizontal* dalam suatu diagram *UML use case*.

Use Case memiliki dua istilah

a. *System use case*; interaksi dengan sistem.

b. *Business use case;* interaksi bisnis dengan konsumen atau kejadian nyata



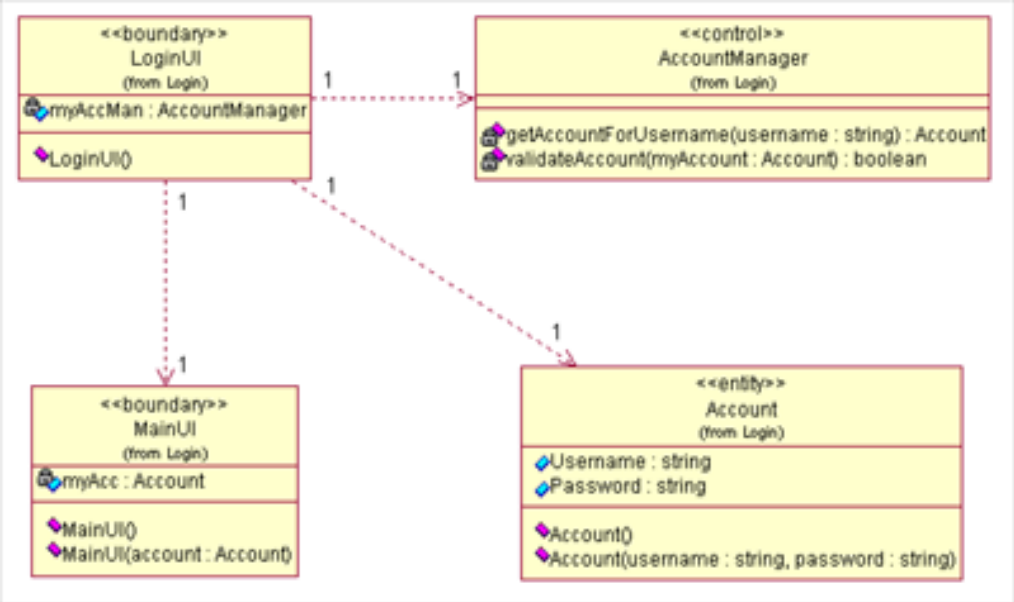
Gambar :2.5. Notasi *use case diagram*(Haviluddin, 2011).

Class diagram menggambarkan struktur statis dari kelas dalam sistem anda dan menggambarkan atribut, operasi dan hubungan antara kelas. *Class diagram* membantu dalam memvisualisasikan struktur kelas-kelas dari suatu sistem dan merupakan tipe diagram yang paling banyak dipakai. Selama tahap desain, *class diagram* berperan dalam menangkap struktur dari semua kelas yang membentuk arsitektur sistem yang dibuat.

Class memiliki tiga area pokok :

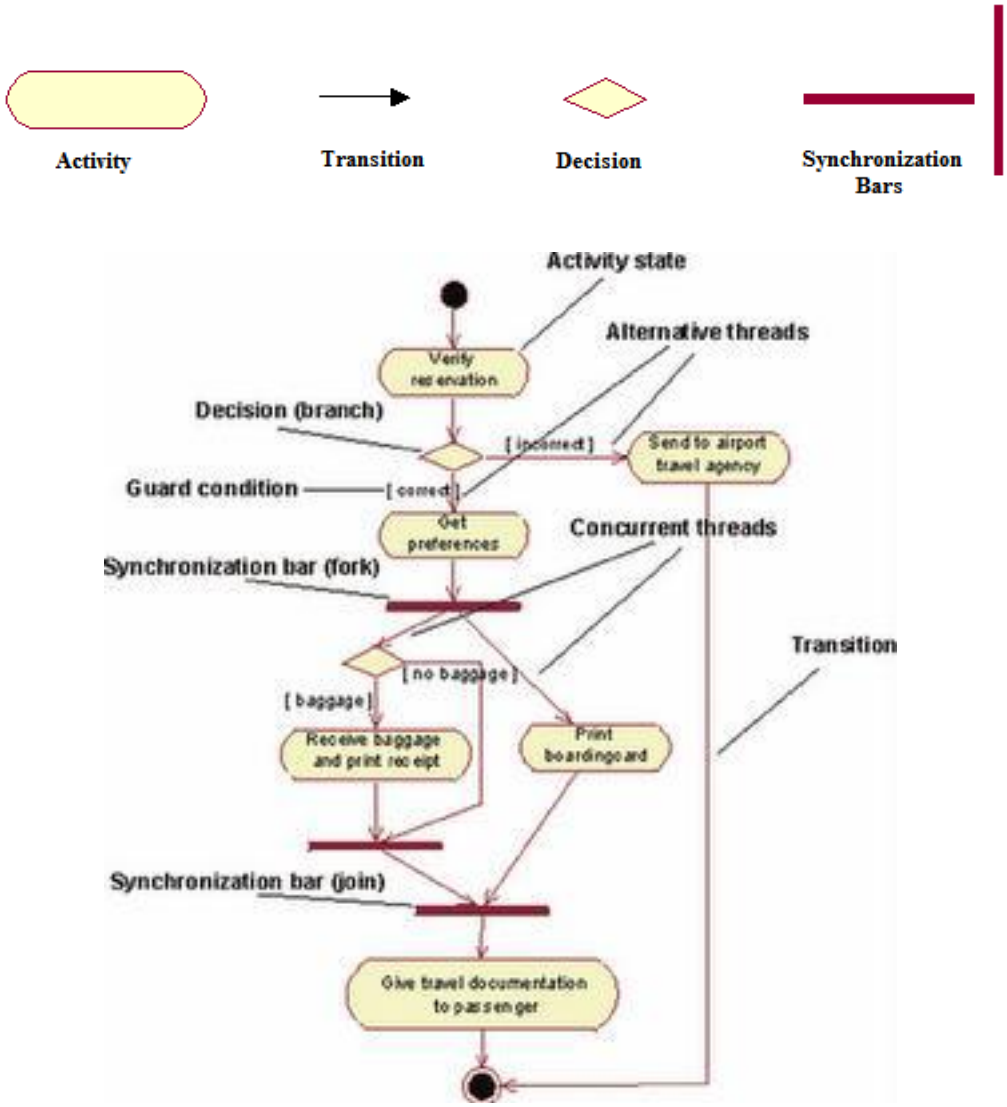
1. Nama (dan *stereotype*)
2. Atribut

3. Metoda



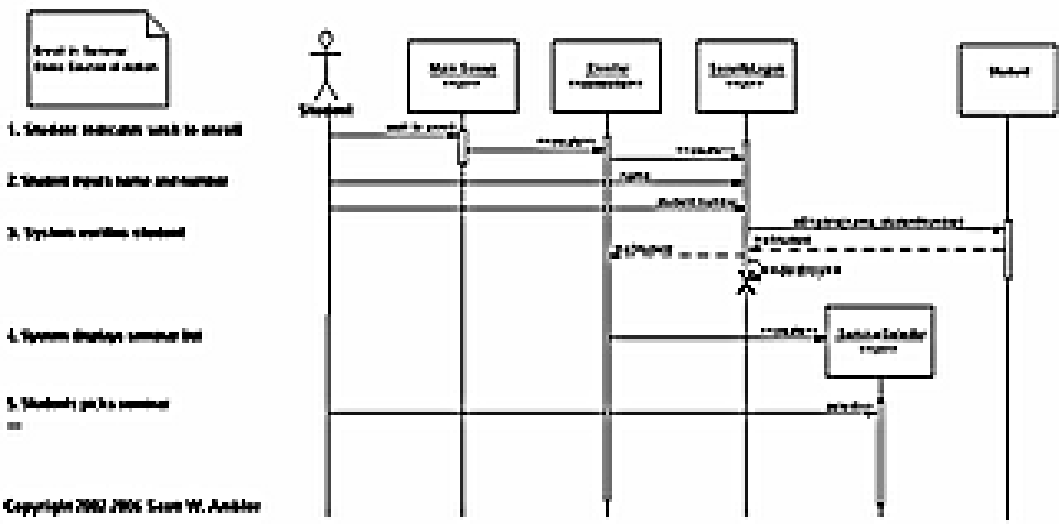
Gambar : 2.6. Notasi *class diagram*(Haviluddin, 2011).

Activity diagram Menggambarkan aktifitas-aktifitas, objek, *state*, transisi *state* dan *event*. Dengan kata lain kegiatan diagram alur kerja menggambarkan perilaku sistem untuk aktivitas



Gambar : 2.7. Notasi *activity* diagram(Haviluddin, 2011).

Sequence diagram menjelaskan interaksi objek yang disusun berdasarkan urutan waktu. Secara mudahnya *sequence* diagram adalah gambaran tahap demi tahap, termasuk kronologi (urutan) perubahan secara logis yang seharusnya dilakukan untuk menghasilkan sesuatu sesuai dengan *use case* diagram.

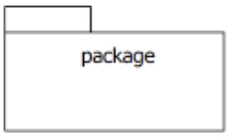
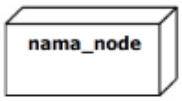


Gambar : 2.8. Notasi *sequence* diagram(Haviluddin, 2011).

2.8. *Deployment Diagram*

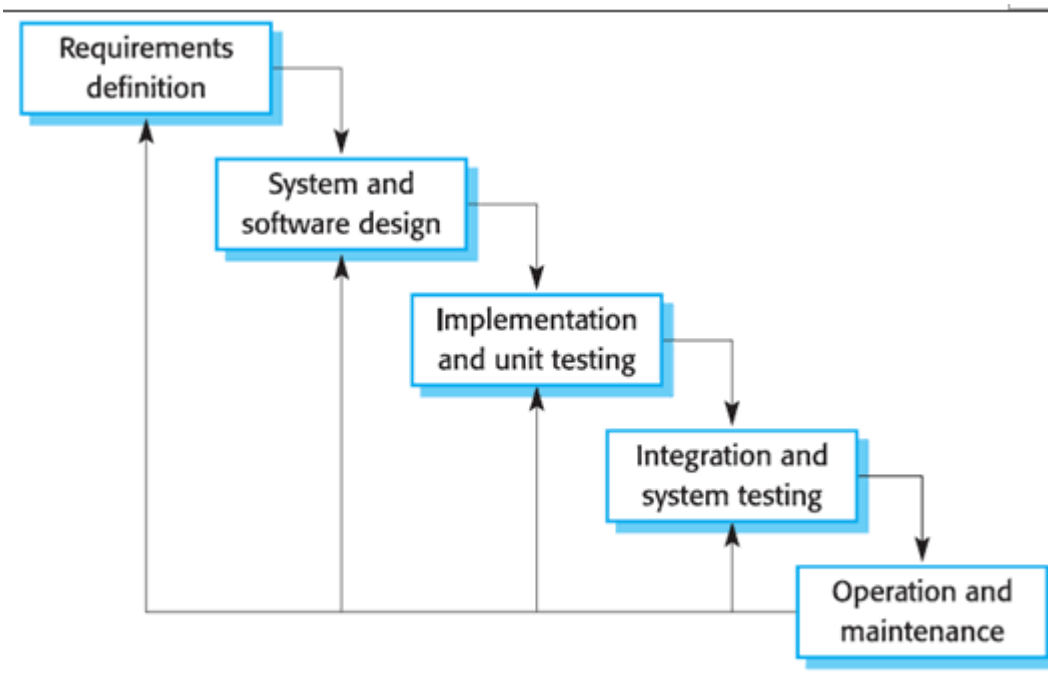
Deployment Diagram adalah salah satu model diagram dalam UML untuk mengerahkan artefak dalam node. Deployment diagram digunakan untuk memvisualisasikan hubungan antara *software* dan *hardware*. Secara spesifik deployment diagram dapat membuat *physical model* tentang bagaimana komponen perangkat lunak (artefak) digunakan pada komponen perangkat keras, yang dikenal sebagai node. Ini adalah salah satu diagram paling penting dalam tingkat implementasi perangkat lunak dan ditulis sebelum melakukan *coding* (Anisa,2020).

Tabel : 2.1. Notasi – notasi pada *Deployment Diagram*(Anisa, 2020).

Simbol	Deskripsi
Package 	package merupakan sebuah bungkusan dari satu atau lebih *node*
Node 	biasanya mengacu pada perangkat keras (*hardware*), perangkat lunak yang tidak dibuat sendiri (*software*), jika di dalam *node* disertakan komponen untuk mengkonsistenkan rancangan maka komponen yang diikutsertakan harus sesuai dengan komponen yang telah didefinisikan sebelumnya pada diagram komponen
Kebergantungan / *dependency* 	Kebergantungan antar *node*, arah panah mengarah pada *node* yang dipakai
Link	relasi antar *node*

2.9. Metode *Waterfall*

Metode *Waterfall* adalah suatu proses pengembangan perangkat lunak berurutan, di mana kemajuan dipandang sebagai terus mengalir ke bawah (seperti air terjun) melewati fase-fase perencanaan, pemodelan, implementasi (konstruksi), dan pengujian. Dalam pengembangannya metode *waterfall* memiliki beberapa tahapan yang runtut: *requirement* (analisis kebutuhan), desain sistem (*system design), Coding & Testing*, Penerapan Program, pemeliharaan(Tristianto, 2018).



Gambar : 2.9. Metode Waterfall(Tristianto, 2018).

Chrisantus Tristianto menyatakan bahwa model *waterfall* memiliki tahapan-tahapan sebagai berikut:

a. *Requirement* (analisis kebutuhan).

Dalam langakah ini merupakan analisa terhadap kebutuhan sistem. Pengumpulan data dalam tahap ini bisa melakukan sebuah penelitian, wawancara atau *study* literatur. Seseorang *system* analisis akan menggali informasi sebanyak-banyaknya dari user sehingga akan tercipta sebuah sistem komputer yang bisa melakukan tugas-tugas yang diinginkan oleh *user* tersebut. Tahapan ini akan menghasilkan dokumen user requirement atau bisa dikatakan sebagai data yang berhubungan dengan keinginan *user* dalam pembuatan sistem. Dokumen inilah yang akan menjadi acuan

system analisis untuk menterjemahkan kedalam bahasa pemrograman (Tristianto, 2018).

b. *Design System* (desain sistem)

Proses design akan menterjemahkan syarat kebutuhan kesebuah perancangan perangkat lunak yang dapat diperkirakan sebelum dibuat *coding*. Proses ini berfokus pada : struktur data, arsitektur perangkat lunak, representasi *interface*, dan detail *(algoritma) prosedural*. Tahapan ini akan menghasilkan dokumen yang disebut *software requirement*. Dokumen inilah yang akan digunakan *programmer* untuk melakukan aktivitas pembuatan sistemnya(Tristianto, 2018).

c. *Coding & Testing* (penulisan sinkode program / implemention)

Coding merupakan penerjemahan *design* dalam bahasa yang bisa dikenali oleh komputer. Dilakukan oleh *programmer* yang akan meterjemahkan transaksi yang diminta oleh *user*. Tahapan inilah yang merupakan tahapan secara nyata dalam mengerjakan suatu sistem. Dalam artian penggunaan *computer* akan dimaksimalkan dalam tahapan ini. Setelah pengkodean selesai maka akan dilakukan *testing* terhadap sistem yang telah dibuat tadi. Tujuan *testing* adalah menemukan kesalahan-kesalahan terhadap *system* tersebut dan kemudian bisa diperbaiki (Tristianto, 2018).

d. *Integration & Testing* (Penerapan / Pengujian Program)

Tahapan ini bisa dikatakan final dalam pembuatan sebuah sistem. Setelah melakukan analisa, *design* dan pengkodean maka sistem yang sudah jadikan digunakan oleh *user*(Tristianto, 2018).

e. *Operation & Maintenance* (Operasi dan Pemeliharaan)

Perangkat lunak yang susah disampaikan kepada pelanggan pasti akan mengalami perubahan. Perubahan tersebut bisa karena mengalami kesalahan karena perangkat lunak harus menyesuaikan dengan lingkungan (periperal atau *system* operasi baru) baru, atau karena pelanggan membutuhkan perkembangan *fungsional*(Tristianto, 2018).

2.10. *Black Box Testing*

Black Box Testing adalah pengujian yang mengabaikan mekanisme *internal* dari sistem atau komponen dan hanya berfokus pada output yang dihasilkan sebagai respon terhadap input yang dipilih dan kondisi eksekusi. Ada 7 (tujuh) jenis level pengujian yang terlibat dalam tes. Ada dua hal yang perlu dipertimbangkan dalam jenis pengujian, yang pertama adalah *opacity* yaitu pandangan kode pengujian (*Black Box* atau *White Box Testing*). Ketujuah pengujian dirangkap dalam *table* di bawah ini(Williams, 2006).

Tabel: 2.2. Bagian-bagian *Blackbox Testing*(Williams, 2006).

Jenis Pengujian	Spesifikasi	Ruang Lingkup Umum	Tipe Pengujian	Yang menguji sistem
Unit	Desain-Level-Rendah Struktur kode actual	*Unit* kecil dari kode dan tidak lebih besar dari kelas	*White box*	*Programer* yang menulis kode
Intergration	Desain-Level-Rendah Desain-Level-Tinggi	Kelas yang banyak	*White box* *Black box*	*Programer* yang menulis kode
Functional	Desain-Level-Tinggi	Semua Produk	*Black box*	Penguji *Independen*
System	Analisis kebutuhan	Semua produk di lingkungan *representive*	*Black box*	Penguji Independen
Acceptance	Analisis kebutuhan	Semua produk di lingkungan pelanggan	*Black box*	Pelanggan
Beta	Ad hoc	Semua produk di lingkungan pelanggan	*Black box*	Pelanggan
Regression	Dokumentasi yang berubah Desain-Level-Tinggi	Salah satu yang di atas	*Black box* *White box*	Programer atau Penguji Independen

BAB III

PEMBAHASAN

3.1. Metode Penelitian

Dalam penelitian ini digunakan metode pengembangan SDLC (*System Development Life Cycle*) dengan model *Waterfall*. Ada lima tahapan untuk pengembangan sistem ini, namun untuk penelitian ini dibatasi sampai dengan tahap keempat(Tristianto, 2018).

Berikut adalah tahapan metode *Waterfall* yang digunakan untuk penelitian ini:

1. *Requirement* (analisis kebutuhan)
2. *Design System* (desain sistem)
3. *Coding & Testing* (penulisan sinkode program / implementasi)
4. *Integration & Testing* (Penerapan / Pengujian Program)
5. *Operation &* Maintenance (Operasi dan Pemilaharaan)

Untuk tahapan dalam penelitian ini dibatasi sampai dengan tahapan Penerapan / Pengujian Program *(Integration & Testing).*

3.1.1. Requirement (analisis kebutuhan)

Untuk tahapan ini merupakan analisa terhadap kebutuhan sistem. proses pengumpulan informasi berupa data-data yang berkaitan dengan penelitian.

3.1.2. Metode Pengumpulan Data

Metode pengumpulan data untuk penelitian ini saya mengambil gambar objek tanaman obat di internet dan sebagian dengan pengambilan kamera.

3.1.3. Metode Wawancara

Metode wawancara yang dilakukan untuk penelitian ini adalah dengan mewawancari masyarakat sekitar yang berada di lingkungan tanaman, narasumber tentunya tahu dengan beberapa macam tanaman obat yang terdapat disekitaran rumah, semak-semak atau lapang yang terdapat tanaman obat apa saja yang biasanya untuk digunakan kebutuhan.

3.1.4. Metode Studi Literatur

Metode Studi Literatur yang didapatkan bersumber dari buku-buku atau jurnal yang berkaitan dengan tanaman obat, metode YOLO, *object detection*, *machine learning*, dan informasi lain yang berkaitan dengan penelitian ini, dibawah ini terdapat tabel yang merupakan referensi penelitian yang digunakan.

TABEL: 3.1. Referensi Penelitian

No	Literatur	Pembahasan
1	Joseph Redmon,Santosh Divvala,Ross Girshick dan Ali Farhadi, *"YOLO(You Only See One Time): Object Detection Integrated Real-Time"*	Penelitian mengenai YOLO *object detection* secara real time, yang menjelaskan algoritma klasifikasi yang cepat ,perbandingan dengan *Faster R-CNN*
2	Abu Ahmad, *"Mengenal Artificial Intelligence, Machine Learning, Neural Network, dan Deep Learning"*	Penelitian mengenai kecerdasan buatan atau *Artificial Intelligence (AI), Machine Learning, Neural Network, dan Deep Learning,* dalam pengenalan setiap metode.
3	Jan Wira Gotama Putra. *"Pengenalan Konsep Pembelajaran Mesin Dan Deep Learning."*	Penelitian mengenai algoritma machine learning, pengenalan kecerdasan buatan, dan logika fuzzy secara matematis.
4	Arthur L. Samuel, *"Some Studies in Machine Learning Using the Game of Checkers"*	Menjelaskan Metode mesin *learning* untuk pembuatan sebuah game yang dapat di adaptasi dengan mudah.
5	Widodo Budiharto, *"AI for Beginner"*	Penelitian yang membahas bidang kecerdasan buatan *(AI)*, yang dapat menghadirkan berbagai piranti dan sistem komersial dalam kehidupan.

Kesimpulan : memperkenalkan YOLO, model terpadu untuk deteksi objek. Model sederhana untuk dibangun dan dapat dilatih langsung pada gambar penuh melakukan pembuatan object detection menggunakan YOLO dengan melakukan riset pada source code yang terbuka. melihat perbedaan antara AI, ML, NN dan DL. Pada dasarnya, semua istilah tersebut saling berkaitan.

3.2. Analisis Permasalahan

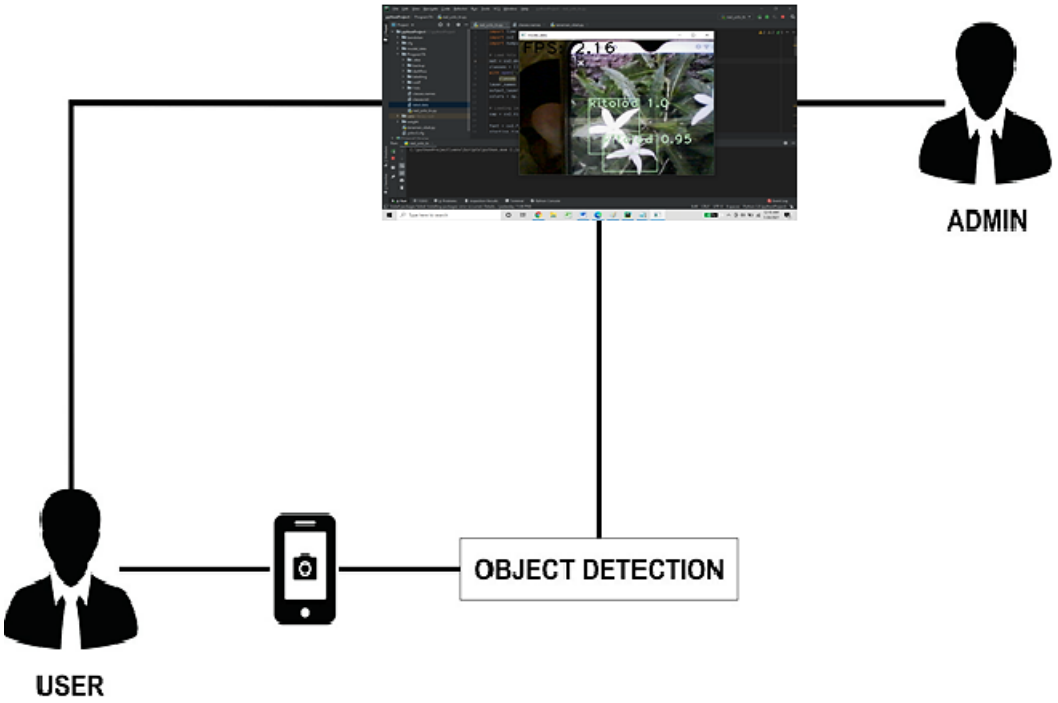
Machine learning adalah merupakan cabang dari ilmu *artificial Intelligence* yang memiliki manfaat di berbagai bidang, salah satunya adalah dalam mendeteksi *objek (object detection).* Dalam penerapannya *machine learning* dapat mempelajari beragam bentuk objek visual dari warna, bentuk, tekstur, dan gambar. Masalah yang sering dihadapi pada *object detection* adalah sulitnya mendeteksi objek atau non objek pada sebuah citra, banyaknya variabilitas objek yang tinggi seperti halnya dengan objek tanaman obat yang memiliki berbagai macam bentuk, warna, dan ukuran yang bervariasi. Permasalahan tersebut dapat mempersulit pengklasifikasian objek untuk menentukan objek yang di deteksi tersebut adalah tanaman obat apa.

3.2.1. Gambaran Umum Sistem Yang Diusulkan

Untuk mendapatkan informasi tanaman obat, maka akan dibangun suatu pengembangan aplikasi dengan menggunakan *object detection* dimana *object detection* ini akan mengenali nama tanaman obat yang diambil dari kamera atau webcam, kemudian hasilnya akan terdeteksi nama dan akurasi dari tanaman obat tersebut.

Aplikasi *object detection* akan dibangun menggunakan bahasa *python* dimana *user* harus menjalankan melalui IDE *PyCharm,* setelah itu ketika aplikasi memunculkan layar kamera *user* harus melakukan *scan object* agar kamera dapat menangkap objek dan menghasilkan nama objek tanaman obat tersebut.

Gambar 3.1 menjelaskan gambaran arsitektur teknologi dari aplikasi object detection yang diusulkan.



GAMBAR: 3.1. Arsitektur Teknologi Dari Aplikasi *Object Detection*

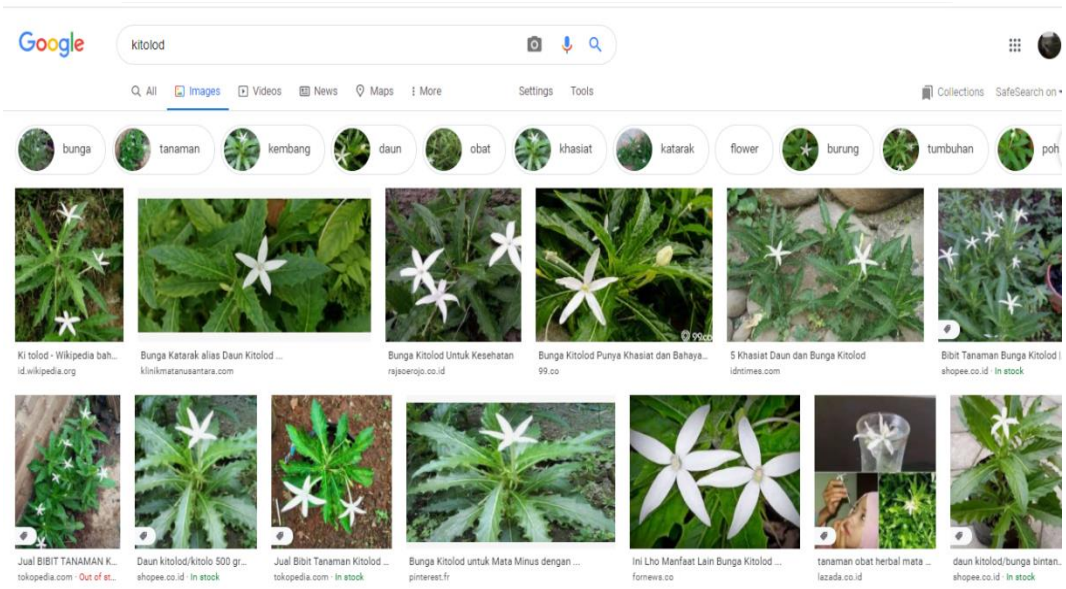
Dalam proses *object detection*, sebelumnya aplikasi *machine learning* ini membutuhkan dataset tanaman obat untuk dijadikan data *train*, sedangkan untuk mendapatkan hasil *object detection* maka *user* harus melakukan *scan object* baik itu dengan mengarahkan gambar tanaman obat ke layar kamera ataupun mengarahkan tanaman obat yang asli. Berikut adalah langkah-langkah *object detection:*

1. Mengumpulkan *Dataset*

Pada penelitian ini proses *object detection* tanaman obat memerlukan *dataset* berupa *weight* yang berisikan koordinat letak objek, *class* tanaman obat tertentu. Proses dari mengumpullkan *dataset* adalah:

a. Data Image

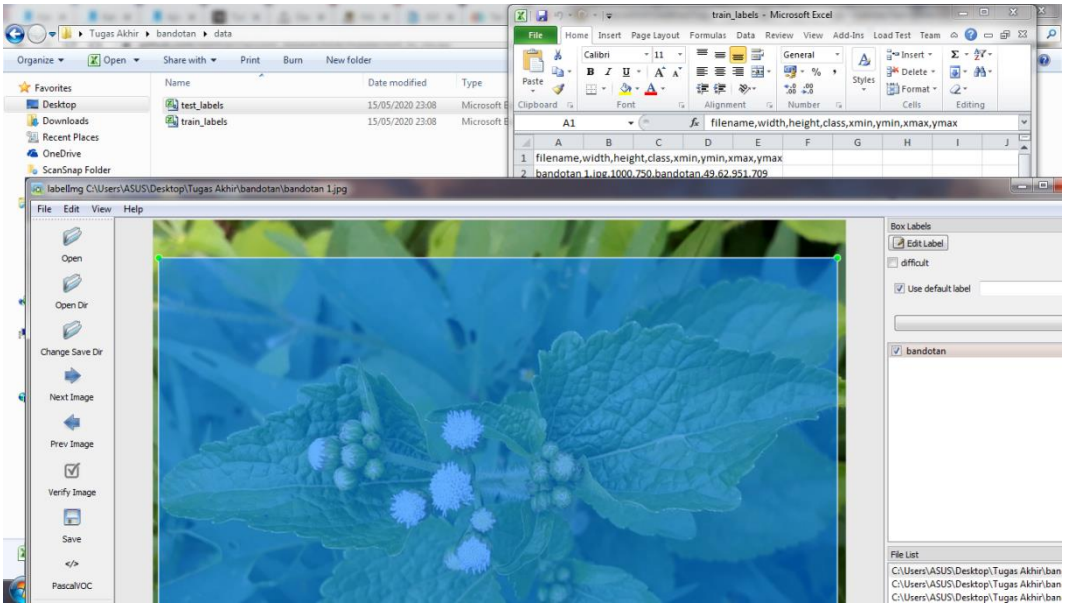
Tahapan pertama yaitu dengan mengumpulkan *dataset image* yang di perlukan dalam penelitian, untuk pengambilan data *image* melalui *google image*. Gambar 3.2 menjelaskan proses pengambilan *image*



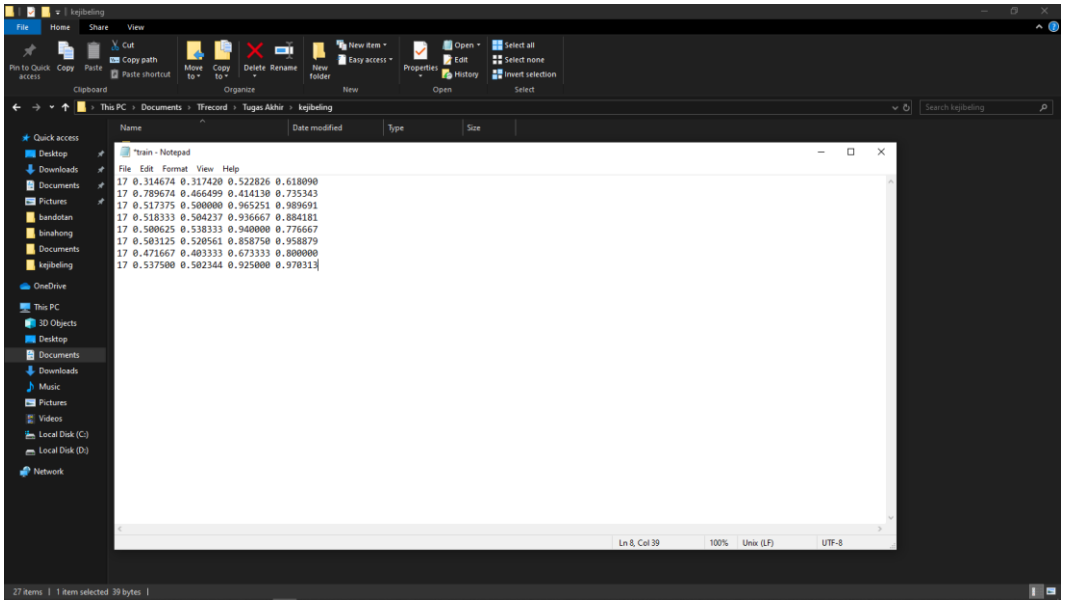
GAMBAR : 3.2. Proses Pengambilan *Image*

b. *Labelling Image*

Dalam tahap ini diambil 5 contoh tanaman obat, dimana setiap tanaman obat memiliki jumlah 100 gambar. Gambar tersebut akan diberikan label dengan cara memberikan kotak pada objek tanaman obat tersebut lalu diberikan nama sebagai identitas kelas objek tersebut, dan hasil *labelling image* ini akan disimpan lalu akan terdapat sebuah *file* dengan format *.txt* yang memiliki data titik koordinat kotak *label* yang dibuat dalam gambar. Gambar 3.3 menjelaskan gambaran *labelling image*, Gambar 3.4 menggambarkan isi dari hasil *labelling image* (*file .txt*).



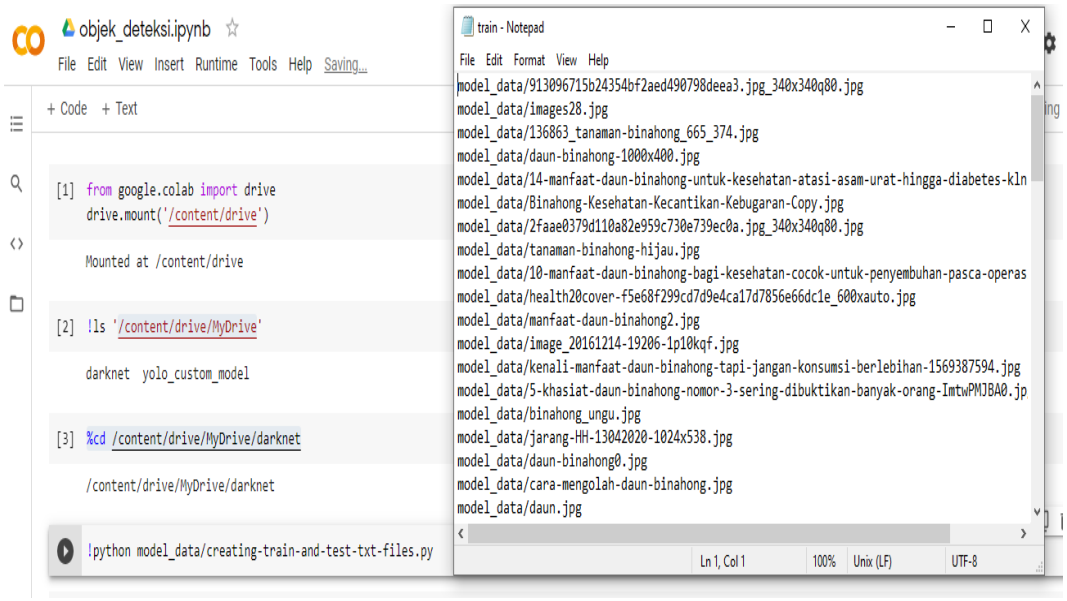
GAMBAR: 3.3. Membuat *Labelling Image*



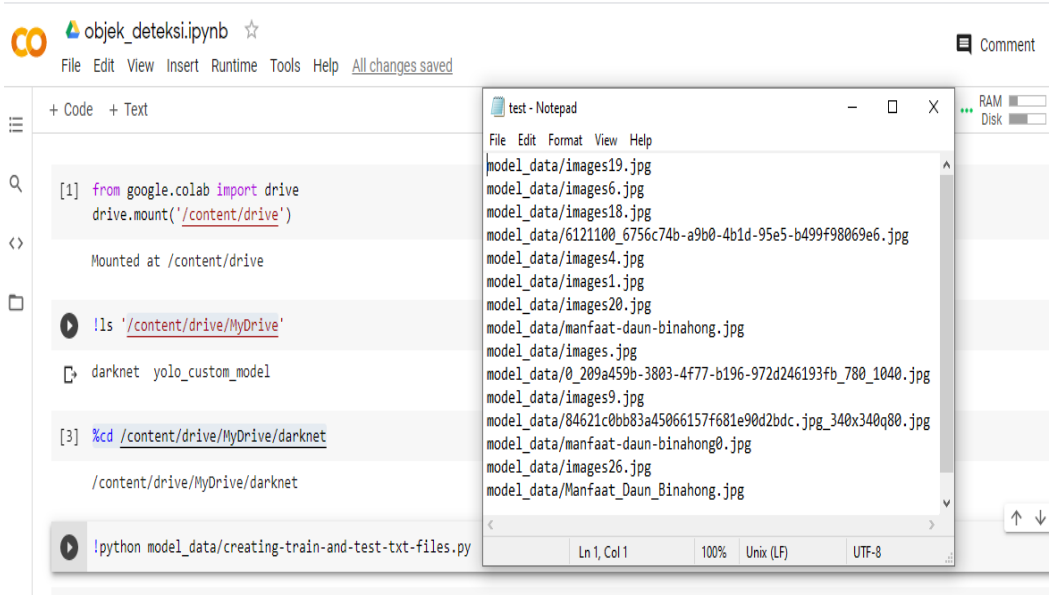
GAMBAR: 3.4. Isi File Format *.txt* Hasil *Labelling Image*

2. Membuat file *Training* dan *Testing*

Selanjutnya gambar akan di bagi dari 100 images ini menjadi 85% data *training* dan 15% data *testing*. Caranya dengan menggunakan *code python*, nantinya kita akan membentuk *weights* baru yang berisi model hasil *training* objek tanaman obat tersebut. Gambar 3.4 menjelaskan proses membuat file *train*. Gambar 3.5 menjelaskan proses membuat file *test*.



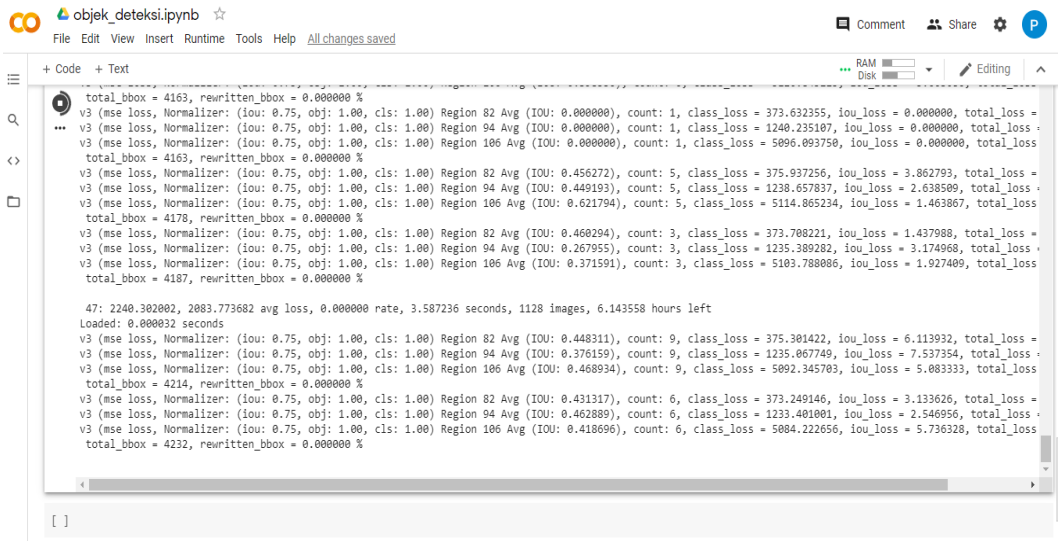
GAMBAR: 3.4. Proses Membuat File *Train*



GAMBAR: 3.5. Proses Membuat File *Test*

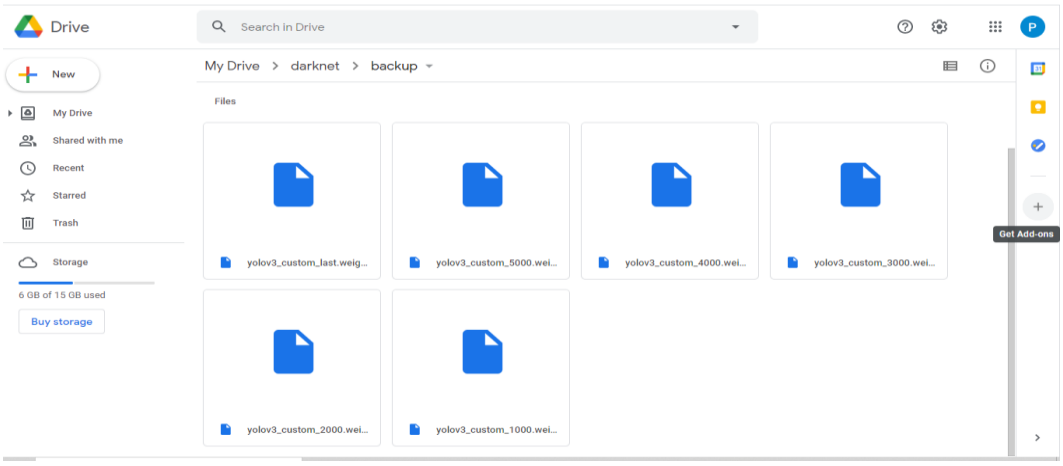
a. *Training*

Dalam tahap ini seluruh *dataset* yang dijadikan *data training* akan di *train* (dilatih) dengan menggunakan model YOLO *(You Look Only Once)*. Seluruh data dari kelas tanaman obat akan dikenali agar aplikasi dapat mendeteksi objek dengan akurat dan sesuai dengan kelas yang mereka miliki. Gambar 3.6 menjelaskan proses untuk *training data*.



GAMBAR: 3.6. Proses *Training* Sedang Berjalan

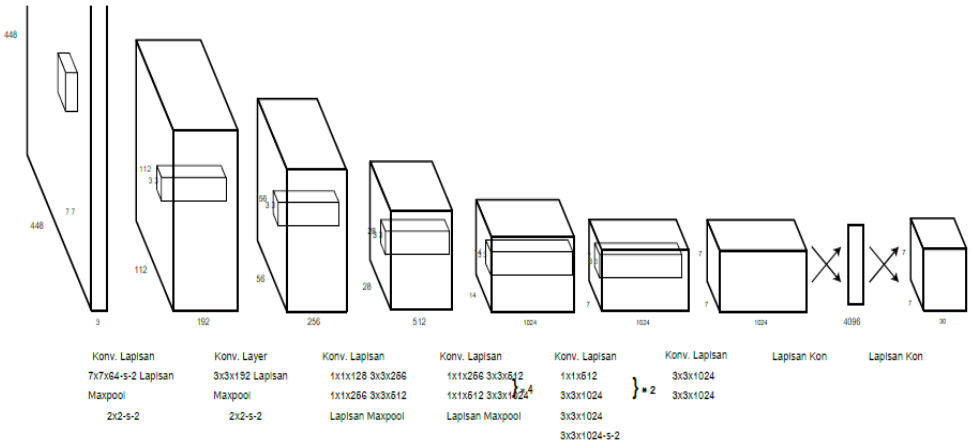
dalam aplikasi membutuhkan *data traning* , *data traning* tersebut berupa *dataset* yang sudah di latih dengan yang nantinya dapat di baca oleh *system object detection* YOLO. Gambar 37 menjelaskan gambaran data traning yang di hasilkan.



GAMBAR: 3.7. Hasil Data Traning yang berhasil.

3. Object Detection menggunakan metode YOLO (*You Only Look Once*)

Dalam melakukan *object detection* diperlukan sebuah metode, yaitu metode YOLO *(You Only Look Once)* mempelajari representasi objek yang dapat digeneralisasikan. Ketika dilatih tentang gambar alami dan diuji pada karya seni, YOLO mengungguli metode deteksi top seperti DPM dan R-CNN dengan margin lebar. Karena YOLO sangat dapat digeneralisasikan, kecil kemungkinannya untuk mogok ketika diterapkan pada domain baru atau input yang tidak terduga. Gambar 3.8 menjelaskan mengenai gambaran arsitektur jaringan YOLO *(You Only Look Once)*



GAMBAR: 3.8. Arsitektur YOLO(Redmon , Divvala, 2016).

Berikut ini cara kerja YOLO *(You Only Look Once)*:

1. Hanya sebuah *Convolutional Neural Network* saja yang akan memprediksi beberapa kotak pembatas dan probabilitas kelas tiap kotak.
2. YOLO menerima sebuah gambar masukan yang dibagi menjadi *grid* sebesar S x S yang dikirimkan ke sebuah jaringan saraf untuk membuat kotak pembatas dan prediksi kelas.
3. Setiap *grid cell* memprediksi B *bounding box* dan *confidenceskor* dari tiap kotak. Skor kepercayaan inilah yang merefleksikan apapun model kepercayaan tingkat bahwa objek di dalam kotak berupa objek yang diprediksikan. YOLO menilai keyakinan sebagai Pr (*Object*) * *IOUtruth* (*Intersection of Union*).
4. Tiap kotak terdiri dari 5 prediksi: x, y, w, h. Koordinat (x, y) merepresentasikan pusat dari kotak relatif dengan batas dari *grid* sel.

Lebar (lebar) dan Tinggi (tinggi) adalah prediksi relatif dari seluruh gambar.

5. Sel *grid* Tiap memprediksi probabilitas kelas kondisional C,Pr (*Classi* |

3.2. *Design System (desain sistem)*

Untuk tahapan ini penelitian akan memfokuskan pada penjadwalan pengerjaan penelitian. Pada Penelitian ini terdapat beberapa proses yang harus dilakukan dari tahap *requirement* hingga *integration & testing*, maka dari itu diperlukan penjadwalan yang tepat agar penelitian ini dapat selesai tepat pada waktunya, berikut penjadwalan penelitian berdasarkan aktifitas yang dilakukan dengan skala waktu per minggu. Tabel 3.1 Akan melihatkan jadwal pengerjaan dalam penelitian ini.

TABEL: 3.1. Penjadwalan Penelitian

Tahapan	No	Aktifitas	Oktober				November				Desember				Januari				Februari	
			1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2
Requirement	1	Pengumpulan Data																		
	2	Analisis																		
	3	Pendefinisian Fungsi																		
Design System	1	Penjadwalan																		
	2	Analisis Kebutuhan Perangkat Keras dan Perangkat Lunak																		
Coding & Testing	1	Perancangan Sistem menggunakan (UML)																		
	2	*User Interface Webcame*																		
	3	Pengkodean																		
Integration & Testing	2	Testing *(Blackbox)*																		

3.2.1. Analisis Kebutuhan Perangkat Keras dan Perangkat Lunak

Untuk membangun dan menjalankan aplikasi ini dibutuhkan persyaratan minimal perangkat keras agar aplikasi dapat berjalan dengan semestinya dan tanpa kendala, Tabel 3.2 menjelaskan persyaratan minimal perangkat keras yang digunakan untuk membangun dan menjalankan aplikasi.

TABEL: 3.2. Spesifikasi Kebutuhan Perangkat Keras

	Developer	***User***
Perangkat Keras	*Processor Intel Core i5 10.1GHz*	*Processor Intel Core i3 1.80GHz*
	RAM 8 Gb	*RAM 2 Gb*
	Free Storage Harddisk 5 Gb	*Free Storage Harddisk 2 Gb*
	Grafis NVIDIA Tesla K4 59C 2 Gb	*Grafis NVIDIA GeForce GT 920MX 2 Gb*
	GPU HD Graphics 620	*GPU HD Graphics 505*
	Monitor dengan resolusi layar 1024x768	Monitor dengan resolusi layar 1024x768
	Keyboard	*Keyboard*
	Mouse	*Mouse*

Selain perangkat keras untuk membangun sebuah aplikasi dan menjalankan aplikasi, membutuhkan spesifikasi perangkat lunak agar aplikasi dapat berjalan dengan lancar dan tanpa kendala, Tabel 3.3 menjelaskan spesifikasi minimal perangkat lunak yang digunakan untuk membangun dan menjalankan aplikasi.

TABEL: 3.3. Spesifikasi Kebutuhan Perangkat Lunak

	Developer	User
Perangkat Lunak	*Sistem Operasi Windows 10*	*Sistem Operasi Windows 7*
	Python 3.8 versi Windows Operating System	*IDE PyCharm*
	LabelImg.exe	
	Sublime text 3/Notepad	
	Google Colab free GPU	
	IDE PyCharm	

3.4. *Coding & Testing* (penulisan sinkode program / implementasi)

Dalam tahapan ini akan dilakukan perancangan dengan menggunakan *Diagram Unified Modeling Language* (*UML*) diantaranya *use case diagram*, *activity diagram, sequence diagram,* dan *class diagram.*

3.4.1. *Use Case Diagram*

Use case diagram menggambarkan proses dari sebuah *system,* hubungan antara *use case* dan *actor* berdasarkan kebutuhan *system* dan menggambarkan fungsionalitas yang diharapkan dari sebuah *system*.

1. Deskripsi *Actor*

Tabel 3.4 menjelaskan deskripsi aktor yang terlibat dalam aplikasi ini.

TABEL: 3.4. Deskripsi Aktor

No	Aktor	Keterangan
1	*User*	*User* yang dimaksud adalah orang yang akan menjadi pengguna aplikasi
2	*Admin*	*Admin* merupakan orang yang mengelola informasi tanaman obat.

2. Deskripsi *Use Case*

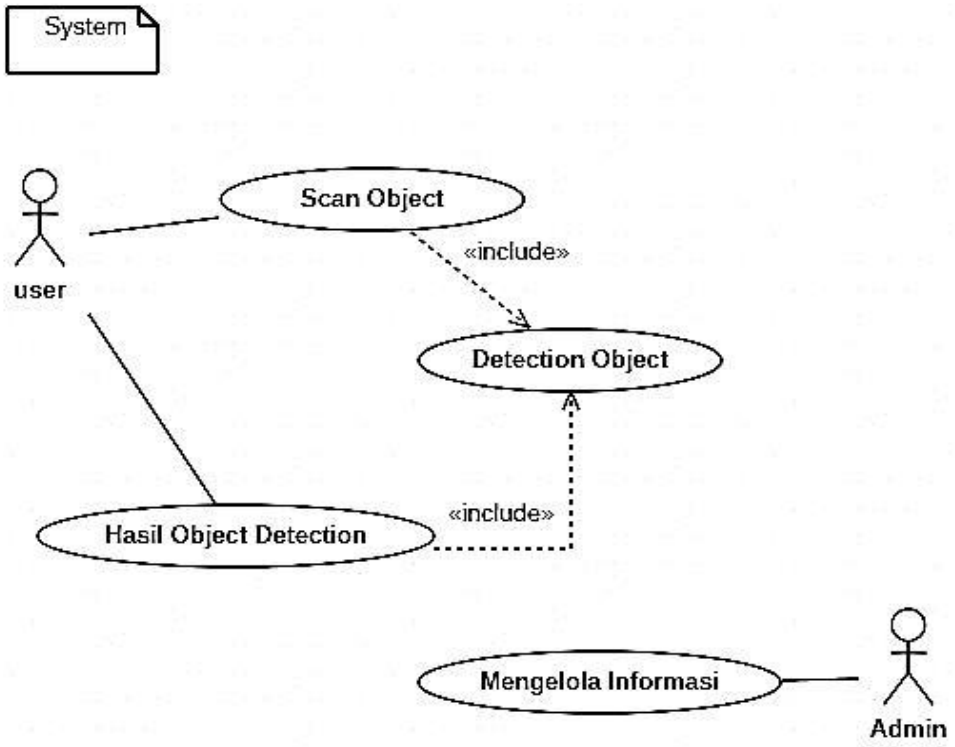
Tabel 3.5. menjelaskan lebih lanjut deskripsi *use case* pada aplikasi ini.

TABEL: 3.5. Deskripsi *Use Case*

No	***Use Case***	**Deskripsi**
1	*Scan Object*	Merupakan fitur yang dapat dilakukan oleh *user* sebelum mendapatkan hasil deteksi objek tanaman obat, *user* harus mengarahkan objek tanaman obat pada kamera yang selanjutnya akan di deteksi. Fitur ini dapat dijalankan dengan menggunakan aplikasi *object detection* yang dijalankan melalui IDE *PyCharm.*
2	*Detecting Object*	Merupakan proses dimana objek yang ditangkap oleh kamera akan dikenali, di deteksi dan diproses oleh *machine learning* untuk dapat menghasilkan nama tanaman obat beserta akurasinya. Fitur ini dapat dijalankan dengan menggunakan aplikasi *object detection* yang dijalankan melalui IDE *PyCharm.*

No	*Use Case*	Deskripsi
3	Melihat Hasil *Object Detection*	Merupakan fitur dimana *user* dapat melihat langsung hasil deteksi objek yang muncul, lengkap dengan nama tanaman obat objek tersebut dan presentase akurasinya. Fitur ini dapat dijalankan dengan menggunakan aplikasi *object detection* yang dijalankan melalui IDE *PyCharm.*
4	Mengelola informasi	Merupakan fitur dimana admin dapat mengubah atau menghapus informasi nama tanaman obat.

Gambar 3.9 menjelaskan *use case* diagram aplikasi**.**



GAMBAR: 3.9. *Use Case Diagram* Aplikasi

Tabel 3.6 menggambarkan *flow of event* (skenario) dari tiap *use case diagram* untuk menjelaskan proses apa saja yang terjadi dan bagaimana respon yang dikeluarkan oleh *system*.

1. *Scan Object*

TABEL: 3.6. Flow of Event Scan Object

Identifikasi			
Nomor *Use Case*	1		
Nama *Use Case*	*Scan Object*		
Deskripsi	Proses untuk *scan object*		
Aktor	*User*		
Skenario			
Kondisi Awal	IDE *PyCharm*		
No	**Aksi Aktor**	**No**	**Respon Sistem**
1	Jalankan aplikasi *object detection* melalui IDE *PyCharm*	2	Menampilkan aplikasi *object detection* dan membuka layar kamera
3	Mengarahkan objek ke Kamera	4	*Scan object*
Kondisi Akhir	Objek di *scan* oleh kamera		

2. *Detecting Object*

TABEL: 3.7. Flow of Event Detecting Object

Identifikasi			
Nomor *Use Case*	2		
Nama *Use Case*	*Detecting Object*		
Deskripsi	Untuk mengenali objek yang terdeteksi oleh kamera dan memproses hasilnya		
Aktor	*User*		
Skenario			
Kondisi Awal	Aplikasi *object detection*		
No	**Aksi Aktor**	**No**	**Respon Sistem**
1	Mengarahkan objek ke kamera	2	Mendeteksi apakah ada objek yang tertangkap kamera dan menampilkan objek yang terdeteksi kamera
3	Menunggu hasil deteksi	4	Memproses hasil
Kondisi Akhir	Hasil deteksi objek diproses		

3. Melihat Hasil *Object Detection*

TABEL: 3.8. Flow of Event Melihat Hasil Object Detection

Identifikasi			
Nomor *Use Case*	3		
Nama *Use Case*	Melihat hasil *object detection*		
Deskripsi	Melihat hasil objek yang di deteksi		
Aktor	*User*		
Skenario			
Kondisi Awal	Aplikasi *object detection*		
No	**Aksi Aktor**	**No**	**Respon Sistem**
1	Menunggu hasil deteksi	2	Menampilkan hasil *object detection* di layar kamera (nama dan akurasi)
3	Melihat hasil objek yang di deteksi	4	Hasil *object detection* ditampilkan
Kondisi Akhir	Hasil deteksi objek dilihat oleh *user*		

4. Mengelola Informasi Tanaman Obat

TABEL: 3.9. *Flow of Event* Mengelola Informasi Tanaman Obat

Identifikasi			
Nomor Use Case	4		
Nama Use Case	Mengelola Informasi		
Deskripsi	Mengelola informasi untuk tanaman obat		
Aktor	*Admin*		
Skenario			
Kondisi Awal	Melatih data traning		
No	**Aksi Aktor**	**No**	**Respon Sistem**
1	Membuka *Google drive*	2	Menampilkan pilihan menu
3	Klik menu Google Colab	4	Menampilkan lembar kerja
5	Konfigurasi data, data di traning	6	Data di traning
Kondisi Akhir	Hasil data traning		

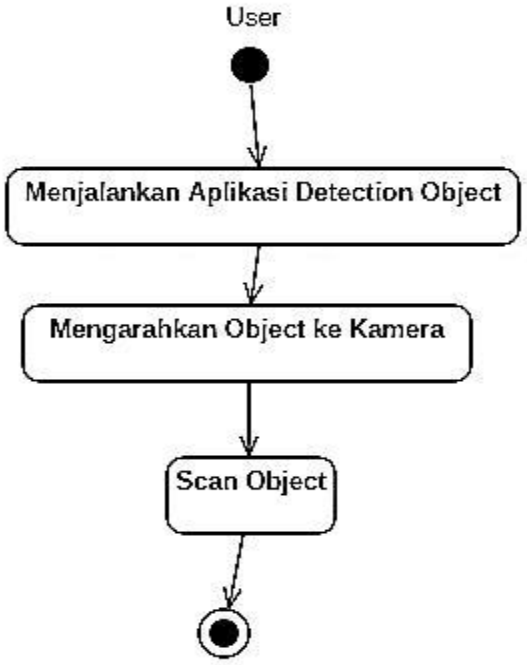
3.4.2. *Activity Diagram*

Activity diagram adalah diagram yang menunjukan aktifitas dari setiap fungsi, yang menggambarkan *workflow* (aliran kerja) dari sebuah sistem atau proses bisnis, dapat juga menggambarkan mengenai aktifitas menu yang ada pada

aplikasi. Berikut adalah *activity diagram* pada aplikasi yang dirancang:

1. *Scan Object*

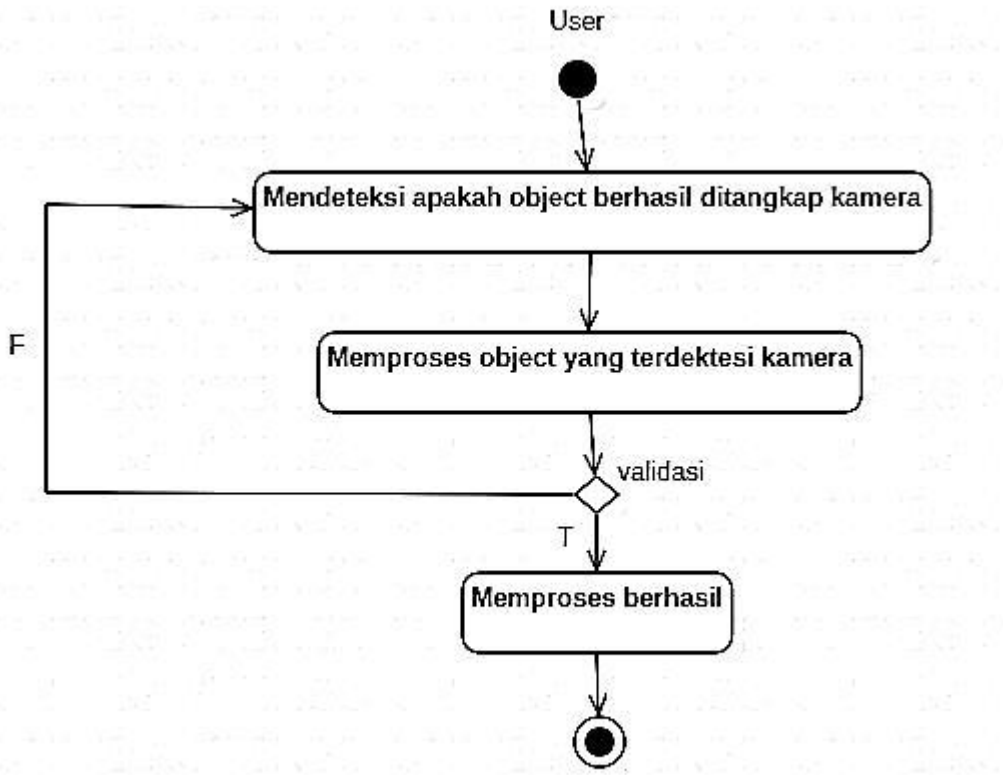
Pada Gambar 3.10 Menunjukan aktifitas pada saat *user* menjalankan aplikasi *object detection*. Ketika *user* ingin menjalankan aplikasi *object detection* maka *user* harus membuka IDE *PyCharm* lalu menjalankan perintah aplikasi *object detection* tersebut, setelah itu maka sistem akan merespon dengan membuka aplikasi *object detection* dan menampilkan layar kamera untuk dapat melakukan *scan object.*



GAMBAR: 3.10. Activity Diagram Scan Object

2. *Detecting Object*

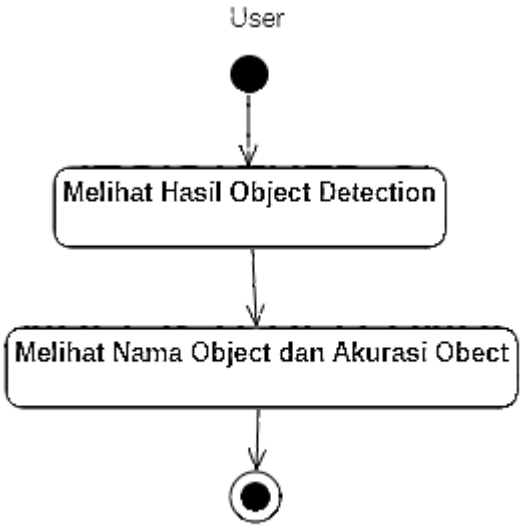
Pada Gambar 3.11 menampilkan aktifitas ketika proses *scan object* sudah dilakukan dan selanjutnya *object* akan di deteksi oleh sistem melalui kamera, jika ada *object* yang terdeteksi maka *object* tersebut akan diproses untuk dikenali dan diproses hasilnya.



GAMBAR: 3.11. Activity Diagram Detecting Object

3. Melihat Hasil *Object Detection*

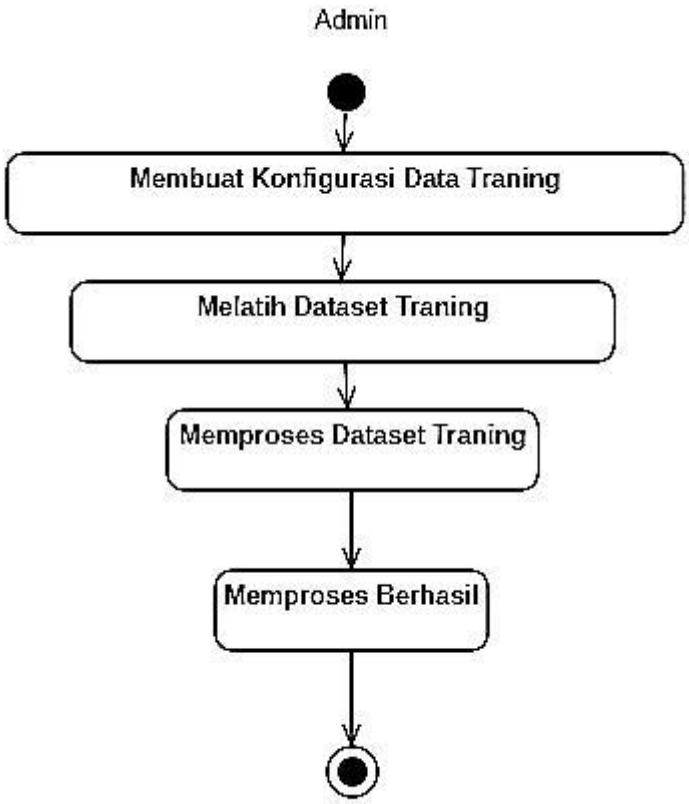
Pada Gambar 3.12 menampilkan aktifitas disaat proses *object detection* sudah dilakukan dan hasilnya sudah diproses, maka hasil *object detection* berupa nama *object* tanaman obat dan akurasinya akan ditampilkan oleh sistem melalui layar kamera yang ditampilkan.



GAMBAR: 3.12. Activity Diagram Melihat Hasil Object Detection

4. Mengelola Informasi Tanaman Obat

Pada Gambar 3.13 menampilkan aktifitas disaat proses mengelola informasi tanaman obat dengan mengkonfigurasi data, melatih data sampai proses berhasil.



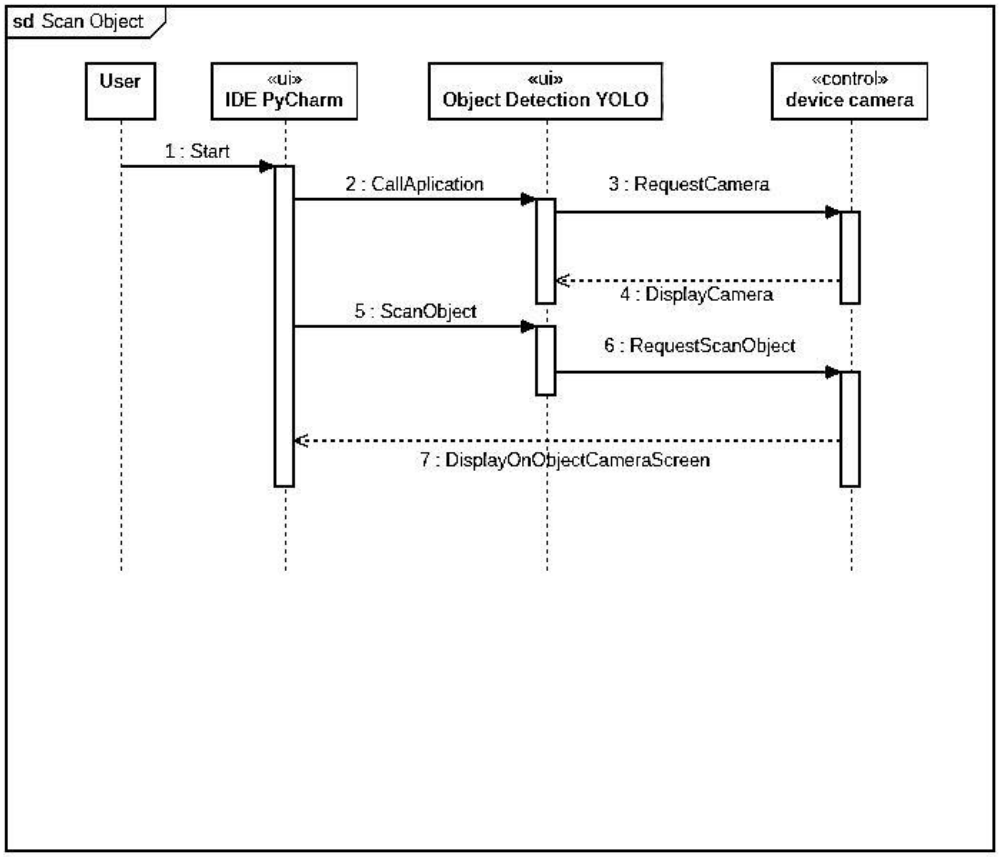
GAMBAR: 3.13. *Activity Diagram* Mengelola Informasi Tanaman Obat

3.4.2. Sequence Diagram

Sequence Diagram digunakan untuk menggambarkan perilaku pada setiap objek pada sebuah skenario. Diagram ini menunjukkan sejumlah contoh objek dan pesan yang diletakkan diantara objek-objek ini didalam *use case*. Komponen utama *sequence diagram* terdiri atas objek yang digambarkan dengan kotak bernama. Pesan diwakili oleh garis dengan tanda panah dan waktu yang ditunjukkan dengan progress vertical.

1. *Scan Object*

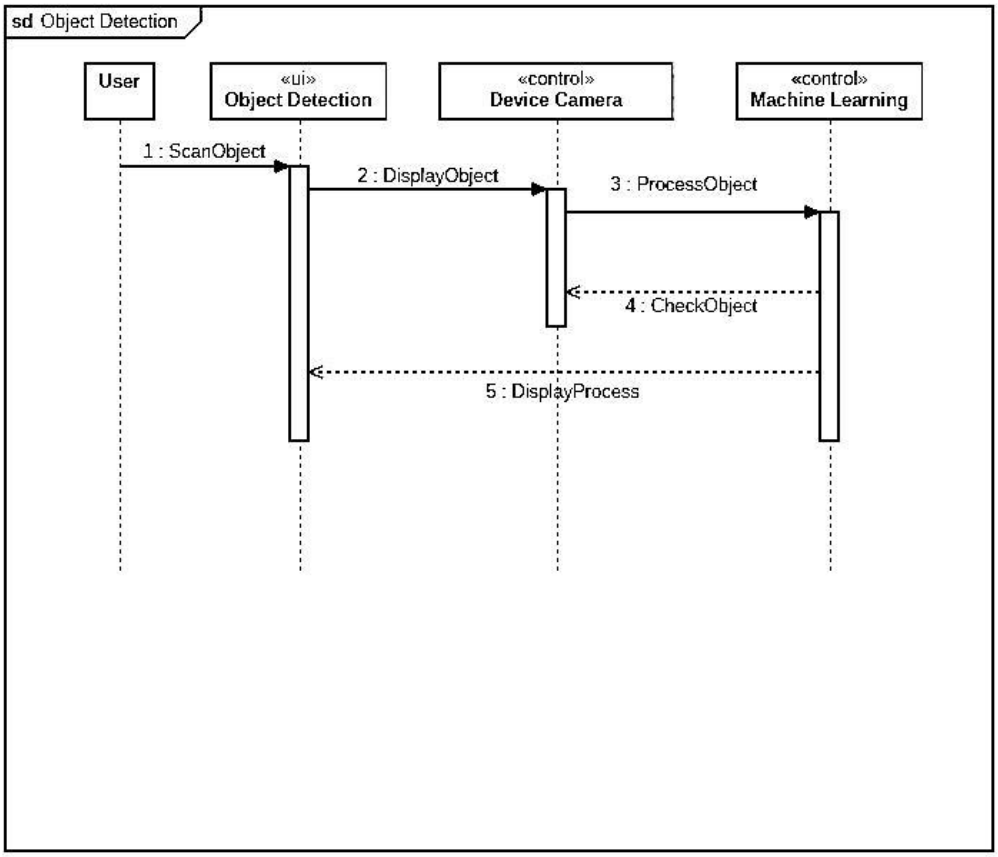
Pada Gambar 3.14 menunjukan *sequence diagram* dari proses *scan object* yang dilakukan oleh user.



GAMBAR: 3.14. *Sequence Diagram Scan Object*

2. *Detecting Object*

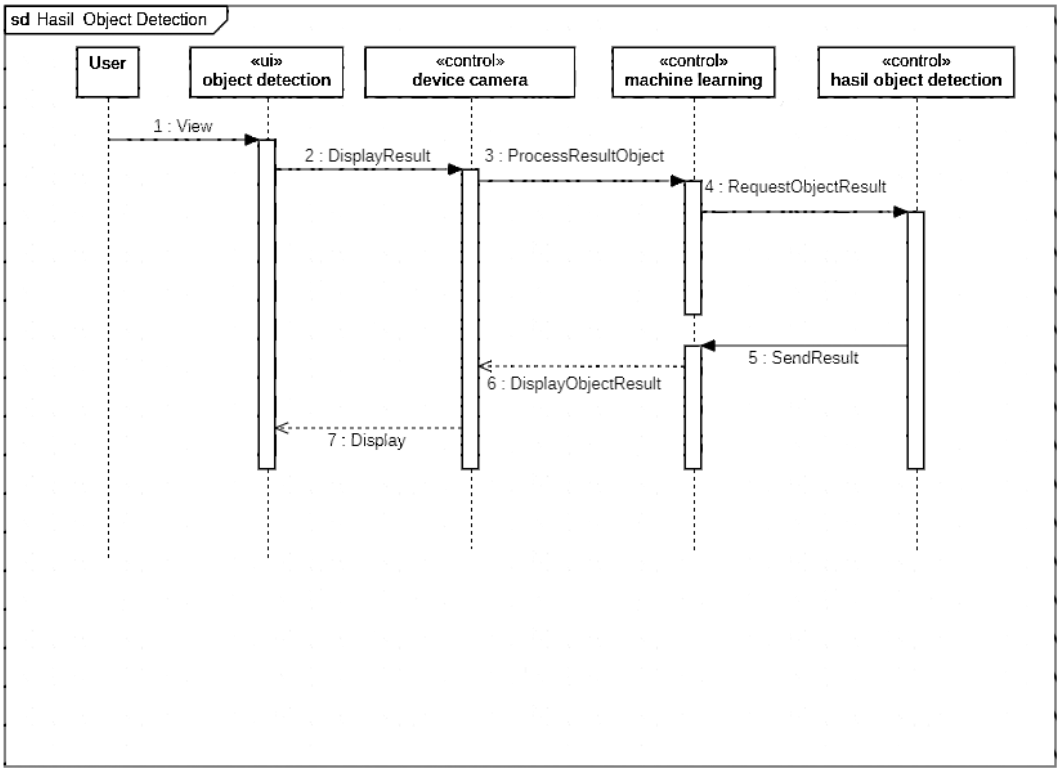
Pada Gambar 3.15 menampilkan *sequence diagram* dari proses *detecting object*.



GAMBAR: 3.15. Sequence Diagram Detecting Object

3. Melihat Hasil *Object Detection*

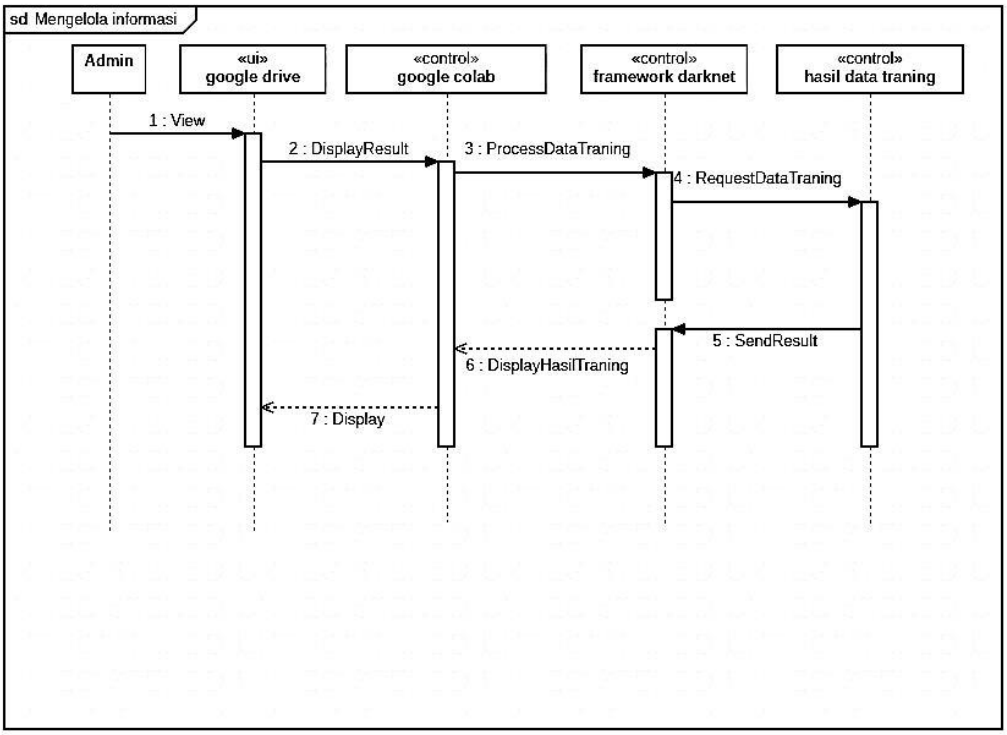
Pada Gambar 3.16 menampilkan *sequence diagram* dari proses hasil *object detection* yang dapat dilihat oleh *user*.



GAMBAR: 3.16. Sequence Diagram Melihat Hasil Object Detection

4. Mengelola Informasi Tanaman Obat

Pada Gambar 3.17 menampilkan *sequence diagram* dari proses Mengelola Informasi Tanaman Obat yang dilakukan oleh *Admin.*



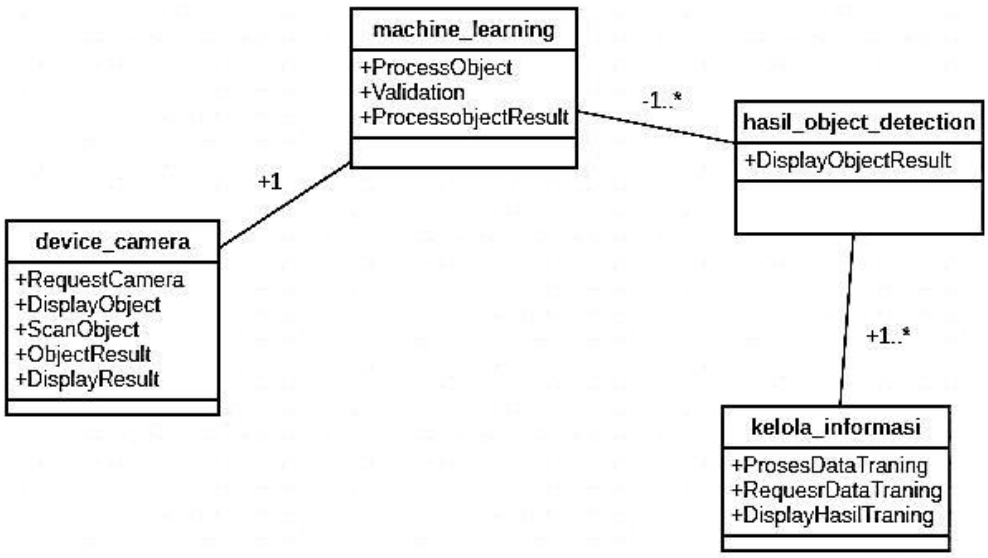
GAMBAR: 3.17. *Sequence Diagram* Mengelola Informasi Tanaman Obat

3.4.3. *Class Diagram*

Class Diagram menggambarkan struktur statis dari kelas dalam sistem dan menggambarkan atribut, operasi dan hubungan antara kelas. *Class diagram* membantu dalam memvisualisasikan struktur kelas-kelas dari suatu sistem dan merupakan tipe diagram yang paling banyak dipakai. Selama tahap desain, *class*

diagram berperan dalam menangkap struktur dari semua kelas yang membentuk arsitektur sistem yang dibuat.

Gambar 3.18 Menggambarkan *class diagram*.

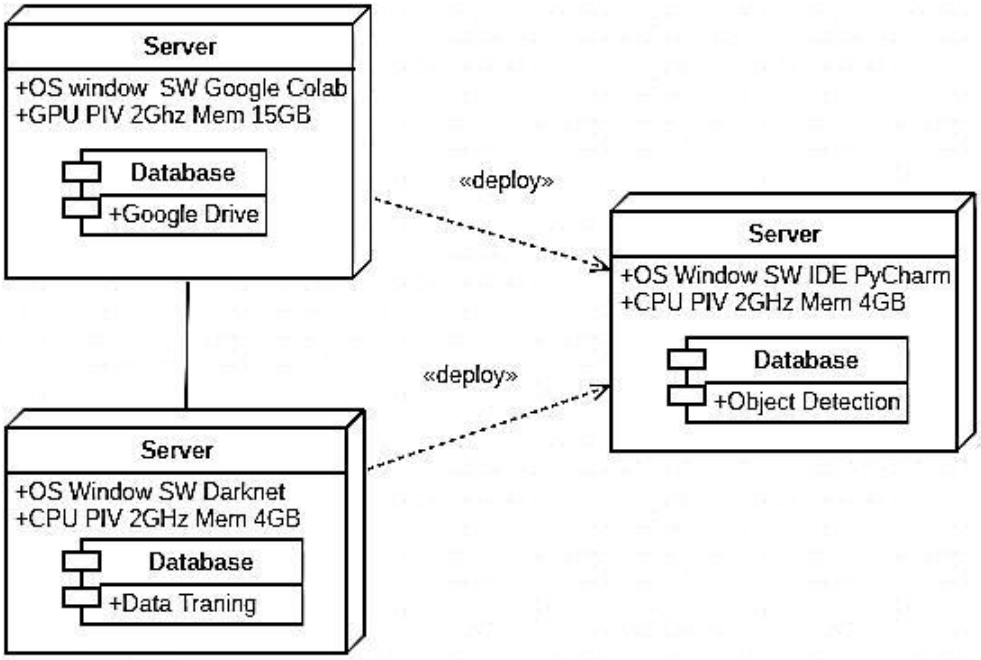


GAMBAR: 3.18. *Class Diagram* Sistem Aplikasi

3.4.4. *Deployment Diagram*

Deployment diagram digunakan untuk memvisualisasikan hubungan antara software dan hardware.

Gambar 3.19 Menggambarkan *deployment diagram*

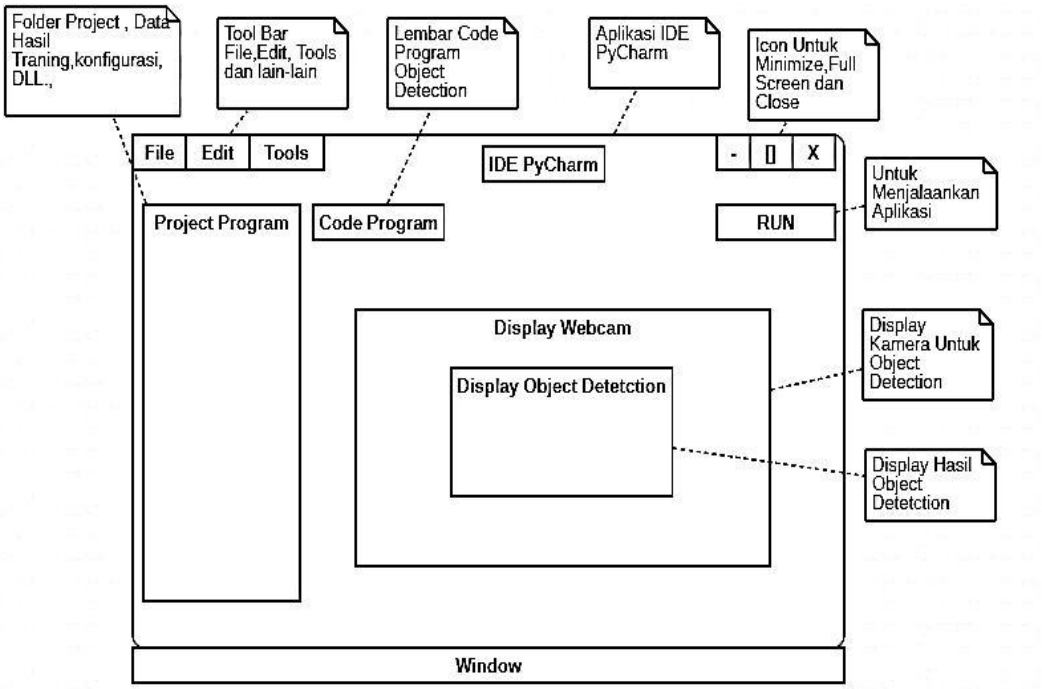


GAMBAR: 3.19. *Deployment Diagram* Sistem Aplikasi

3.4.5. *User Interface*

1. Aplikasi *Object Detection*

Pada Gambar 3.20 menampilkan *User Interface* Aplikasi dari hasil *object detetction* pada *IDE PyCharm*



GAMBAR: 3.20. *User Interface* Display Objek deteksi Tanaman Obat

BAB IV

UJI COBA DAN IMPLEMENTASI

4.1. *Integration & Testing* (Penerapan / Pengujian Program)

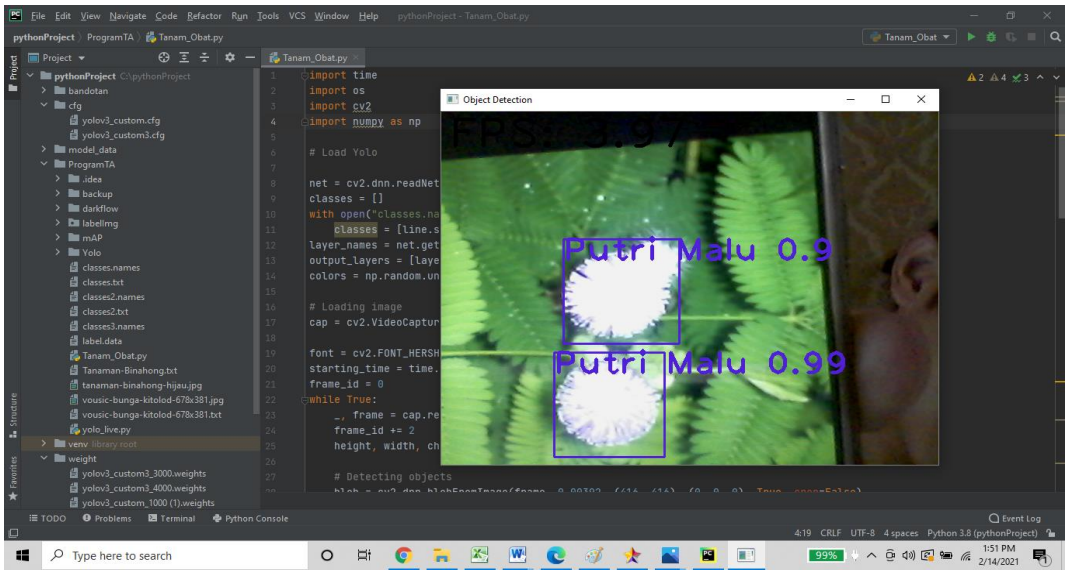
Dalam tahapan ini penelitian berfokus pada penerapan, pengujian program menggunakan bahasa *Python*, setelah itu dilakukan pengujian hasil menggunakan metode *blackbox testing*. *Blackbox testing* ini untuk menguji spesifikasi suatu fungsi atau modul, apakah berjalan sesuai yang diharapkan atau tidak.

4.1.1. *Integration*

Gambar 4.1 menjelaskan penerapan *sincode program.*

GAMBAR: 4.1. Penerapan *Sincode Program Object Detection.*

Gambar 4.2 menjelaskan gambaran *display object detection program.*



GAMBAR: 4.2. Hasil Tampilan *Object Detection* Dengan Data Gambar

4.1.2. *Testing*

TABEL 4.1. menampilkan rencana pengujian.

TABEL: 4.1. Tabel Rencana Pengujian

Setelah melakukan pengujian dengan menggunakan metode *black box*

No	Kelas Uji	Tahap Uji	Jenis Pengujian
1	Pengujian Scan Object	Validasi Layar Kamera dan *Object*	*Black box*
2	Pengujian *Object Detection*	Validasi Data Object	*Black box*
3	Pengujian Melihat Hasil *Object Detection*	Validasi *Object*	*Black box*

maka didapatkan hasil pengujian pada Tabel 4.2.

TABEL: 4.2. Tabel Hasil Pengujian *Black Box*

No	Data Masukan	Hasil yang Diharapkan	Pengamatan	Kesimpulan
1	Objek tanaman obat	Objek muncul di layar kamera	Aplikasi menampilkan objek di layar kamera	*Valid*
2	Objek tanaman obat	Objek di deteksi oleh kamera	Kamera mendeteksi objek	*Valid*
3	Objek tanaman obat	Hasil deteksi objek ditampilkan di layar kamera	Aplikasi menampilkan hasil deteksi objek berupa nama dan akurasi di layar kamera	*Valid*
4	Objek Lain	Objek tidak akan terdeteksi hasilnya	Aplikasi tidak menampilkan hasil nama atau akurasi di objek tersebut	*Valid*

BAB V

KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berdasarkan hasil pembahasan beserta penelitian yang telah dilakukan, maka dapat diambil beberapa kesimpulan, diantaranya:

1. *Object Detection* Metode YOLO *(You Only Look Once)* sangat cepat karena jaringan saraf tunggal memprediksi kotak dan probabilitas kelas yang terikat langsung dari gambar penuh dalam satu evaluasi. Namun, itu membuat lebih banyak kesalahan lokalisasi dan kecepatan pelatihannya relatif lambat.
2. Hasil *object detection* yang ditampilkan menampilkan nama dan kotak pembatas (bounding box) dengan akurasi nya yang muncul pada layar.

5.2. Saran

Dengan adanya kesimpulan diatas, ada beberapa saran yang dapat pertimbangan lebih lanjut untuk meningkatkan produktifitas kerja dari aplikasi *object detection* ini.

1. Karena aplikasi ini hanya untuk mendeteksi objek tanaman obat, jadi objek selain tanaman obat tidak dapat di deteksi harap menambahkan jenis lainnya.
2. Karena *dataset* awal hanya diambil beberapa objek tanaman obat saja, jadi tidak seluruh tanaman obat dapat terdeteksi. Kendala yang didapat adalah dari proses *training* yang sangat lama. maka memerlukan spesifikasi laptop serta GPU yang lebih besar.

DAFTAR PUSTAKA

Adi Nugroho (2010) 'Rekayasa Perangkat Lunak UML dan Java', *Yogyakarta: Andi Offset*.

Ahmad, A. (2017) 'Mengenal Artificial Intelligence, Machine Learning, Neural Network, dan Deep Learning', *Yayasan Cahaya Islam, Jurnal Teknologi Indonesia*, pp. 1–5.

Anisa, N. (2020) *Erbedaan Deployment Diagram Dan Component Diagram*, *Binus University*. Available at: https://sis.binus.ac.id/2020/04/20/perbedaan-deployment-diagram-dan-component-diagram/.

Budiharto, W. (2018) 'AI for Beginner', *AI for Beginner*, pp. 1–11.

Dewi, S. R. (2018) 'Deep Learning Object Detection Pada Video', *Deep Learning Object Detection Pada Video Menggunakan Tensorflow Dan Convolutional Neural Network*, pp. 1–60. Available at: https://dspace.uii.ac.id/bitstream/handle/123456789/7762/14611242_Syarifah Rosita Dewi_Statistika.pdf?sequence=1.

Haviluddin (2011) 'Memahami Penggunaan UML (Unified Modelling Language)', *Memahami Penggunaan UML (Unified Modelling Language)*, 6(1), pp. 1–15. Available at: https://informatikamulawarman.files.wordpress.com/2011/10/01-jurnal-informatika-mulawarman-feb-2011.pdf.

Ika Atman Satya (1995) 'Penyebaran Informasi Menggunakan Www (World Wide Web)', *Baca: Jurnal Dokumentasi Dan Informasi*, 20(5). doi: http://dx.doi.org/10.14203/j.baca.v20i5.35.

Jan, P. and Gotama, W. (2019) 'Pengenalan Pembelajaran Mesin dan Deep Learning Pengenalan Konsep Pembelajaran Mesin dan Deep Learning', 1(July), pp. 1–199. Available at: https://wiragotama.github.io/.

Jeffery L. Whitten, Lonnie D. Bentley, K. C. D. (2006) 'Metode desain dan analisa sistem', *yogyakarta Andi*, 3(xx), p. 726.

Joseph Redmon , Santosh Divvala, R. G. dan A. F. (2016) 'Anda Hanya Melihat Sekali : Deteksi Objek', *Konferensi IEEE*, pp. 779–788.

Poole, J. D. (2001) 'Model-Driven Architecture: Vision, Standards And Emerging Technologies', *Workshop on Metamodeling and Adaptive Object Models, ECOOP01*, (April), pp. 1–15.

Samuel, A. L. (2000) 'Some studies in machine learning using the game of checkers', *IBM Journal of Research and Development*, 44(1–2), pp. 207–219. doi: 10.1147/rd.441.0206.

Saudah, S., Viena, V. and Ernilasari, E. (2019) ‘Eksplorasi Spesies Tumbuhan Berkhasiat Obat Berbasis Pengetahuan Lokal Di Kabupaten Pidie’, *Jurnal Tumbuhan Obat Indonesia,* 12(2), pp. 56–67. doi: 10.22435/jtoi.v12i2.952.

Schneiderman, Henry, Kanade, T. (2000) ‘A statistical method for 3D object detection applied to faces andcars’, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, (1), pp. 746–751. doi: 10.1109/CVPR.2000.855895.

Triasanti, D. (2000) ‘Konsep dasar python’, *AP2B*, pp. 1–6.

Tristianto, C. (2018) ‘Penggunaan Metode Waterfall Untuk Pengembangan Sistem Monitoring Dan Evaluasi Pembangunan Pedesaan’, *Jurnal Teknologi informasi*, 12(1).

Widodo, Prabowo.P, D. (2011) ‘Pemodelan Sistem Berorientasi Obyek Dengan UML’, *Graha ilmu*.

Williams, L. (2006) ‘Testing Overview and Black-Box Testing Techniques’, *International Conference on Software Engineering (ISCE) 2007*, pp. 35–59.

Zhao, Xia Jia, H. and Ni, Y. (2018) ‘A novel three-dimensional object detection with the modified You Only Look Once method’, *International Journal of Advanced Robotic Systems*, 15.

LAMPIRAN-LAMPIRAN

Proses Traning Dataset


```
{
 "nbformat": 4,
 "nbformat_minor": 0,
 "metadata": {
  "colab": {
   "name": "objek_deteksi.ipynb",
   "provenance": []
  },
  "kernelspec": {
   "name": "python3",
   "display_name": "Python 3"
  },
  "accelerator": "GPU"
 },
 "cells": [
  {
   "cell_type": "code",
   "metadata": {
    "colab": {
     "base_uri": "https://localhost:8080/"
    },
    "id": "uXLGBNaWxFxy",
    "outputId": "3b06e38d-5f04-4f1d-af34-ce303a4ff01d"
   },
   "source": [
    "from google.colab import drive\r\n",
    "drive.mount('/content/drive')"
   ],
   "execution_count": null,
   "outputs": [
    {
     "output_type": "stream",
     "text": [
      "Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount(\"/content/drive\", force_remount=True).\n"
     ],
```

```
      "name": "stdout"
    }
   ]
  },
  {
   "cell_type": "code",
   "metadata": {
    "colab": {
     "base_uri": "https://localhost:8080/"
    },
    "id": "VQUpF0ub_eHo",
    "outputId": "324b1f68-caa1-4594-fba4-57d563dbf654"
   },
   "source": [
    "!ls '/content/drive/MyDrive'"
   ],
   "execution_count": null,
   "outputs": [
    {
     "output_type": "stream",
     "text": [
      "darknet  yolo_custom_model\n"
     ],
     "name": "stdout"
    }
   ]
  },
  {
   "cell_type": "code",
   "metadata": {
    "colab": {
     "base_uri": "https://localhost:8080/"
    },
    "id": "DPNGpj2EATiX",
    "outputId": "40cd1861-f063-461e-c3f6-2e790bfb9da1"
   },
   "source": [
    "%cd /content/drive/MyDrive/darknet"
   ],
   "execution_count": null,
   "outputs": [
    {
```

```
      "output_type": "stream",
      "text": [
        "/content/drive/MyDrive/darknet\n"
      ],
      "name": "stdout"
    }
  ]
},
{
  "cell_type": "code",
  "metadata": {
    "id": "IY8AtTwo-8ub"
  },
  "source": [
    "#!python ./modeltanam/creating-train-and-test-txt-files.py"
  ],
  "execution_count": null,
  "outputs": []
},
{
  "cell_type": "code",
  "metadata": {
    "colab": {
      "base_uri": "https://localhost:8080/"
    },
    "id": "RVf4Gwu1BOPr",
    "outputId": "a92313fa-760b-4d1d-9cf0-501717a6ebd6"
  },
  "source": [
    "%cd /content/drive/MyDrive/darknet"
  ],
  "execution_count": null,
  "outputs": [
    {
      "output_type": "stream",
      "text": [
        "/content/drive/MyDrive/darknet\n"
      ],
      "name": "stdout"
    }
  ]
},
```

```
{
 "cell_type": "code",
 "metadata": {
  "colab": {
   "base_uri": "https://localhost:8080/"
  },
  "id": "OYP0nZPEAcjW",
  "outputId": "11e9e2d6-89b9-47bc-b6bf-0c10eb1a0b98"
 },
 "source": [
  "!make"
 ],
 "execution_count": null,
 "outputs": [
  {
   "output_type": "stream",
   "text": [
    "chmod +x *.sh\n"
   ],
   "name": "stdout"
  }
 ]
},
{
 "cell_type": "code",
 "metadata": {
  "id": "g34qszj7AjuV"
 },
 "source": [
  "!chmod +x ./darknet"
 ],
 "execution_count": null,
 "outputs": []
},
{
 "cell_type": "code",
 "metadata": {
  "colab": {
   "base_uri": "https://localhost:8080/"
  },
  "id": "U_pwmvs1AmGu",
  "outputId": "338dbc06-e84a-4585-c3f2-62066c7d4e8c"
```

```
  },
  "source": [
    "!rm /content/darknet/backup -r"
  ],
  "execution_count": null,
  "outputs": [
    {
      "output_type": "stream",
      "text": [
        "rm: cannot remove '/content/darknet/backup': No such file or directory\n"
      ],
      "name": "stdout"
    }
  ]
},
{
  "cell_type": "code",
  "metadata": {
    "colab": {
      "base_uri": "https://localhost:8080/"
    },
    "id": "qzpE-YOnA1an",
    "outputId": "bfa314a7-2663-4363-c850-a5c496936145"
  },
  "source": [
    "!ln -s /content/drive/'My Drive'/backup/darknet"
  ],
  "execution_count": null,
  "outputs": [
    {
      "output_type": "stream",
      "text": [
        "ln: failed to create symbolic link './darknet': File exists\n"
      ],
      "name": "stdout"
    }
  ]
},
{
  "cell_type": "code",
  "metadata": {
    "colab": {
```

```
        "base_uri": "https://localhost:8080/"
       },
       "id": "IDIJUHlOA4c-",
       "outputId": "5eb6be49-b973-45e6-b456-28579d4f4556"
      },
      "source": [
       "!sudo apt install dos2unix"
      ],
      "execution_count": null,
      "outputs": [
       {
        "output_type": "stream",
        "text": [
         "Reading package lists... Done\n",
         "Building dependency tree       \n",
         "Reading state information... Done\n",
         "The following NEW packages will be installed:\n",
         "  dos2unix\n",
         "0 upgraded, 1 newly installed, 0 to remove and 15 not upgraded.\n",
         "Need to get 351 kB of archives.\n",
         "After this operation, 1,267 kB of additional disk space will be used.\n",
         "Get:1 http://archive.ubuntu.com/ubuntu bionic/universe amd64 dos2unix amd64 7.3.4-3 [351 kB]\n",
         "Fetched 351 kB in 1s (347 kB/s)\n",
         "debconf: unable to initialize frontend: Dialog\n",
         "debconf: (No usable dialog-like program is installed, so the dialog based frontend cannot be used. at /usr/share/perl5/Debconf/FrontEnd/Dialog.pm line 76, <> line 1.)\n",
         "debconf: falling back to frontend: Readline\n",
         "debconf: unable to initialize frontend: Readline\n",
         "debconf: (This frontend requires a controlling tty.)\n",
         "debconf: falling back to frontend: Teletype\n",
         "dpkg-preconfigure: unable to re-open stdin: \n",
         "Selecting previously unselected package dos2unix.\n",
         "(Reading database ... 146442 files and directories currently installed.)\n",
         "Preparing to unpack .../dos2unix_7.3.4-3_amd64.deb ...\n",
         "Unpacking dos2unix (7.3.4-3) ...\n",
         "Setting up dos2unix (7.3.4-3) ...\n",
         "Processing triggers for man-db (2.8.3-2ubuntu0.1) ...\n"
        ],
        "name": "stdout"
       }
```

```
 ]
},
{
 "cell_type": "code",
 "metadata": {
  "colab": {
   "base_uri": "https://localhost:8080/"
  },
  "id": "cv43vzWlBCOd",
  "outputId": "63b4b8d9-92d4-492e-963d-87e69388fcbd"
 },
 "source": [
  "!dos2unix ./modeltanam/train.txt"
 ],
 "execution_count": null,
 "outputs": [
  {
   "output_type": "stream",
   "text": [
    "dos2unix: converting file ./modeltanam/train.txt to Unix format...\n"
   ],
   "name": "stdout"
  }
 ]
},
{
 "cell_type": "code",
 "metadata": {
  "colab": {
   "base_uri": "https://localhost:8080/"
  },
  "id": "guljgAHuBI9t",
  "outputId": "f5a240d6-54a3-479f-9ddb-44639b8b2f45"
 },
 "source": [
  "!dos2unix ./modeltanam/test.txt"
 ],
 "execution_count": null,
 "outputs": [
  {
   "output_type": "stream",
   "text": [
```

```
        "dos2unix: converting file ./modeltanam/test.txt to Unix format...\n"
       ],
       "name": "stdout"
     }
    ]
   },
   {
    "cell_type": "code",
    "metadata": {
     "colab": {
      "base_uri": "https://localhost:8080/"
     },
     "id": "rRYzxTekBQLe",
     "outputId": "5de78e0d-3768-4b55-8e8c-e59a4ff560ad"
    },
    "source": [
     "!dos2unix ./modeltanam/label.data"
    ],
    "execution_count": null,
    "outputs": [
     {
      "output_type": "stream",
      "text": [
       "dos2unix: converting file ./modeltanam/label.data to Unix format...\n"
      ],
      "name": "stdout"
     }
    ]
   },
   {
    "cell_type": "code",
    "metadata": {
     "colab": {
      "base_uri": "https://localhost:8080/"
     },
     "id": "abUlwlxHBXhG",
     "outputId": "7c329e77-0f8a-4ff8-9fad-f1de0d40929c"
    },
    "source": [
     "!dos2unix ./modeltanam/classes.names"
    ],
    "execution_count": null,
```

```
  "outputs": [
   {
    "output_type": "stream",
    "text": [
     "dos2unix: converting file ./modeltanam/classes.names to Unix format...\n"
    ],
    "name": "stdout"
   }
  ]
 },
 {
  "cell_type": "code",
  "metadata": {
   "colab": {
    "base_uri": "https://localhost:8080/"
   },
   "id": "AAGmx8AlBfmd",
   "outputId": "7da39f13-b973-494b-ff95-ce8f1fff8cd2"
  },
  "source": [
   "!dos2unix ./cfg/yolov3_custom.cfg"
  ],
  "execution_count": null,
  "outputs": [
   {
    "output_type": "stream",
    "text": [
     "dos2unix: converting file ./cfg/yolov3_custom.cfg to Unix format...\n"
    ],
    "name": "stdout"
   }
  ]
 },
 {
  "cell_type": "code",
  "metadata": {
   "colab": {
    "base_uri": "https://localhost:8080/"
   },
   "id": "YXSxUaA6B7JO",
   "outputId": "873b898f-6cb3-4753-d58e-15babee60b8a"
  },
```

```
    "source": [
      "%cd /content/drive/MyDrive/darknet"
    ],
    "execution_count": null,
    "outputs": [
      {
        "output_type": "stream",
        "text": [
          "/content/drive/MyDrive/darknet\n"
        ],
        "name": "stdout"
      }
    ]
  },
  {
    "cell_type": "code",
    "metadata": {
      "colab": {
        "base_uri": "https://localhost:8080/"
      },
      "id": "F3-qD0vKCB4c",
      "outputId": "b2e9b50e-9cec-4f3d-829b-e0e10e950c77"
    },
    "source": [
      "!./darknet detector train modeltanam/label.data cfg/yolov3_custom.cfg custom_weight/darknet53.conv.74 -dont_show"
    ],
    "execution_count": null,
    "outputs": [
      {
        "output_type": "stream",
        "text": [
"v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 82 Avg (IOU: 0.894220), count: 4, class_loss = 0.004885, iou_loss = 0.053846, total_loss = 0.058731 \n",
          "v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 94 Avg (IOU: 0.000000), count: 1, class_loss = 0.278106, iou_loss = 0.000000, total_loss = 0.278106 \n",
          "v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 106 Avg (IOU: 0.223465), count: 1, class_loss = 0.530107, iou_loss = 1.253662, total_loss = 1.783769 \n",
          " total_bbox = 624192, rewritten_bbox = 0.131690 % \n",
```

"v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 82 Avg (IOU: 0.893650), count: 2, class_loss = 0.008363, iou_loss = 0.070553, total_loss = 0.078916 \n",
"v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 94 Avg (IOU: 0.000000), count: 1, class_loss = 0.000000, iou_loss = 0.000000, total_loss = 0.000000 \n",
"v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 106 Avg (IOU: 0.000000), count: 1, class_loss = 0.000000, iou_loss = 0.000000, total_loss = 0.000000 \n",
" total_bbox = 624194, rewritten_bbox = 0.131690 % \n",
"v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 82 Avg (IOU: 0.000000), count: 1, class_loss = 0.000000, iou_loss = 0.000000, total_loss = 0.000000 \n",
"v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 94 Avg (IOU: 0.908295), count: 4, class_loss = 0.000365, iou_loss = 0.067227, total_loss = 0.067593 \n",
"v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 106 Avg (IOU: 0.000000), count: 1, class_loss = 0.000000, iou_loss = 0.000000, total_loss = 0.000000 \n",
" total_bbox = 624198, rewritten_bbox = 0.131689 % \n"
],
"name": "stdout"
}
]
}
]
}

Proses Object Detection

```
import time
import os
import cv2
import numpy as np

# Load Yolo

net = cv2.dnn.readNet("../weight/yolov3_custom_3000.weights", "../cfg/yolov3_custom.cfg")
classes = []
with open("classes.names", "r") as f:
    classes = [line.strip() for line in f.readlines()]
```

```
layer_names = net.getLayerNames()
output_layers = [layer_names[i[0] - 1] for i in net.getUnconnectedOutLayers()]
colors = np.random.uniform(0, 255, size=(len(classes), 3))

# Loading image
cap = cv2.VideoCapture(0)

font = cv2.FONT_HERSHEY_PLAIN
starting_time = time.time()
frame_id = 0
while True:
    _, frame = cap.read()
    frame_id += 2
    height, width, channels = frame.shape

    # Detecting objects
    blob = cv2.dnn.blobFromImage(frame, 0.00392, (416, 416), (0, 0, 0), True, crop=False)

    net.setInput(blob)
    outs = net.forward(output_layers)

    # Showing informations on the screen
    class_ids = []
    confidences = []
    boxes = []
    for out in outs:
        for detection in out:
            scores = detection[5:]
            class_id = np.argmax(scores)
            confidence = scores[class_id]
            if confidence > 0.2:
                # Object detected
                center_x = int(detection[0] * width)
                center_y = int(detection[1] * height)
                w = int(detection[2] * width)
                h = int(detection[3] * height)

                # Rectangle coordinates
                x = int(center_x - w / 2)
                y = int(center_y - h / 2)
```

```
            boxes.append([x, y, w, h])
            confidences.append(float(confidence))
            class_ids.append(class_id)

    indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.8, 0.3)

    for i in range(len(boxes)):
        if i in indexes:
            x, y, w, h = boxes[i]
            label = str(classes[class_ids[i]])
            confidence = confidences[i]
            color = colors[class_ids[i]]
            cv2.rectangle(frame, (x, y), (x + w, y + h), color, 2)
            cv2.putText(frame, label + " " + str(round(confidence, 2)), (x, y + 30), font, 3,
color, 3)

    elapsed_time = time.time() - starting_time
    fps = frame_id / elapsed_time
    cv2.putText(frame, "FPS: " + str(round(fps, 2)), (10, 50), font, 4, (0, 0, 0), 3)
    cv2.imshow("Object Detection", frame)
    key = cv2.waitKey(2)
    if key == 27:
        break

cap.release()
cv2.destroyAllWindows()
```