PENENTU MUTU LADA BERDASARKAN PERSENTASE BENDA ASING BERBASIS PENGOLAHAN CITRA

PROYEK AKHIR

Laporan akhir ini dibuat dan diajukan untuk memenuhi salah satu syarat kelulusan Sarjana Terapan Politeknik Manufaktur Negeri Bangka Belitung

Disusun Oleh:

Muhammad Erfani Ramadhani	NIM : 1051917
Siti Barokah	NIM : 1051926

POLITEKNIK MANUFAKTUR NEGERI

BANGKA BELITUNG

TAHUN 2023

LEMBAR PENGESAHAN

PENENTU MUTU LADA BERDASARKAN PERSENTASE BENDA ASING BERBASIS PENGOLAHAN CITRA

Oleh :

Muhammad Erfani Ramadhani /1051917

Siti Barokah /1051926

Laporan akhir ini telah disetujui dan disahkan sebagai salah satu syarat kelulusan Program Sarjana Terapan Politeknik Manufaktur Negeri Bangka Belitung

Menyetujui,

Pembimbing 1	Pembimbing 2
(Indra Dwisaputra, M.T.)	(Ocsirendi, M.T.)
Penguji 1	**Penguji 2**
(Irwan, M.Sc.,Ph.D.)	(Aan Febriansyah, M.T.)

PERNYATAAN BUKAN PLAGIAT

Yang bertanda tangan dibawah ini :

Nama Mahasiswa 1 : Muhammad Erfani Ramadhani NIM : 1051917

Nama Mahasiswa 2 : Siti Barokah NIM : 1051926

Dengan Judul : Penentu Mutu Lada Berdasarkan Persentase Benda Asing Berbasis Pengolahan Citra

Menyatakan bahwa laporan akhir ini adalah hasil kerja kami sendiri dan bukan merupakan plagiat. Pernyataan ini kami buat dengan sebenarnya dan bila ternyata dikemudian hari ternyata melanggar pernyataan ini, kami bersedia menerima sanksi yang berlaku.

Sungailiat, 15 Februari 2023

Nama Mahasiswa	Tanda Tangan
1. Muhammad Erfani Ramadhani	…….
2. Siti Barokah	…….

ABSTRAK

Salah satu rempah-rempah yang paling banyak digunakan adalah lada, baik dalam hal kontribusinya terhadap devisa negara maupun penggunaaannya dalam memasak. Bangka Belitung merupakan salah satu daerah penghasil lada putih terbaik di Indonesia yang dikenal dengan "Muntok white pepper". Para petani di Bangka menjual lada ke pengepul/pembeli lada. Para pengepul di daerah ini memerlukan pemeriksaan lada dan benda asing masih menggunakan cara manual. Cara ini sering kali tidak efesien karena presisi setiap orang berbeda-beda. Penentuan standar mutu lada telah diatur dalam standar mutu nasional (SNI). Dengan perkembangan ilmu pengetahuan dan teknologi muncul sebuah ide untuk membuat sistem penentu mutu lada berdasarkan persentase benda asing secara otomatis. Sistem ini menggunakan metode You Only Look Once (YOLO), lebih tepatnya YOLOv3-tiny yang merupakan versi lebih ringan dari YOLOv3. Dengan menggunakan model YOLOv3-tiny untuk membuat sistem pendeteksian ini, didapatkan hasil bahwa model YOLOv3-tiny memiliki nilai performa jaringan yang cukup tinggi yaitu nilai Precision sebesar 0,99, nilai Recall diatas 70%, dan F1 Score bernilai diatas 80%. Model YOLOv3-tiny ini memberikan hasil yang baik dan sistem juga dapat menentukan mutu lada berdasarkan persentase benda asing. Persentase didapatkan dari total benda asing yang berhasil terdeteksi dibagi total keseluruhan benda yang berhasil terdeteksi, kemudian kalikan 100. Menurut standart mutu lada (SNI) nilai yang didapatkan harus dibawah 2%. Kemudian dilakukan perbandingan antara sistem deteksi yang telah dibangun dengan perhitungan benda secara manual. Didapatkan bahwa sistem dapat mendeteksi 7 lada dengan waktu 0,27 detik lebih baik daripada cara manual dan 26 lada dengan waktu 8,97 detik lebih baik daripada cara manual.

Kata kunci : Lada, YOLOv3, TinyYOLOv3, image detection

ABSTRACT

One of the most widely used spices is pepper, both in terms of its contribution to the country's foreign exchange and its use in cooking. Bangka Belitung is one of the best white pepper producing regions in Indonesia known as "Muntok white pepper". Farmers in Bangka sell pepper to pepper collectors/buyers. Collectors in this area require manual inspection of pepper and foreign objects such as leaves and stalks. This method is often inefficient because everyone's precision is different. The determination of pepper quality standards has been regulated in the national quality standards (SNI). With the development of science and technology came an idea to create a system for determining the quality of pepper based on the percentage of foreign bodies automatically. This system uses the You Only Look Once (YOLO) method, more precisely YOLOv3-tiny which is a lighter version of YOLOv3. By using the YOLOv3-tiny model to create this detection system, it was found that the YOLOv3-tiny model has a fairly high network performance value, namely a Precision value of 0.99, a Recall value above 70%, and an F1 Score worth above 80%. This YOLOv3-tiny model gives good results and the system can also determine the quality of pepper based on the percentage of foreign objects. The percentage is obtained from the total foreign objects that were successfully detected divided by the total number of objects that were successfully detected, then multiply by 100. According to the pepper quality standard (SNI) the value obtained must be below 2%. Then a comparison is made between the detection system that has been built with the calculation of objects manually. It was found that the system could detect 7 peppers with a time of 0.27 seconds better than the manual way and 26 peppers with a time of 8.97 seconds better than the manual way.

***Keywords** : Pepper, YOLOv3, TinyYOLOv3, image detection*

KATA PENGANTAR

Puji syukur kita panjatkan atas kehadirat Allah SWT. Yang telah memberikan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Laporan Proyek Akhir yang berjudul "**Penentu Mutu Lada Berdasarkan Persentase Benda Asing Berbasis Pengolahan Citra**" tepat pada waktunya. Shalawat serta salam kepada Rasulullah Muhammad S.A.W, yang telah membawa manusia ke jalan yang damai, terang benderang dan penuh ilmu pengetahuan.

Penyusunan laporan akhir ini selain merupakan salah satu persyaratan kelulusan pendidikan Diploma IV di Politeknik Manufaktur Negeri Bangka Belitung juga dimaksudkan untuk menambah ilmu pengetahuan pada bidang pengolahan citra atau jaringan saraf tiruan.

Pada kesempatan ini, penulis ingin mengucapkan rasa terima kasih kepada berbagai pihak yang telah memberikan bantuan dan dukungan baik materil maupun nonmateril yang diberikan kepada penulis dalam menyelesaikan laporan akhir ini. Oleh karena itu, izinkan penulis mengucapkan rasa terima kasih kepada :

1. Allah SWT. yang telah memberikan rahmat dan hidayah-Nya hingga penulis dapat menyelesaikan laporan proyek akhir ini.
2. Kedua Orang Tua dan keluarga yang selalu mendukung serta mendoakan penulis sehingga dapat menyelesaikan laporan akhir ini.
3. Bapak I Made Andik Setiawan, M.Eng., Ph.D., selaku Direktur di Politeknik Manufaktur Negeri Bangka Belitung.
4. Bapak Indra Dwisaputra, M.T, selaku pembimbing 1 yang telah banyak meluangkan waktu, tenaga dan pikiran, serta memberikan saran dalam pengerjaan Proyek Akhir.
5. Bapak Ocsirendi, M.T, selaku pembimbing 2 yang telah memberikan saran dan solusi pada pengerjaan Proyek Akhir.

6. Seluruh Staff Komisi Proyek Akhir yang sudah membantu kegiatan Proyek Akhir.
7. kepada rekan-rekan mahasiswa yang telah membantu dalam pengerjaan Laporan akhir ini.
8. semua pihak yang telah memberikan bantuan yang tidak dapat penulis sebutkan satu persatu sehingga mengantarkan penulis untuk menyelesaikan laporan akhir ini.

Dalam penyusunan Laporan ini tentunya masih banyak terdapat kekurangan, kesalahan dan kekhilafan karena keterbatasan kemampuan penulis, untuk itu sebelumnya penulis mohon maaf yang sebesar-besarnya. Penulis juga mengharapkan kritik dan saran dari semua pihak demi perbaikan yang bersifat membangun atas laporan ini.

Akhir kata dengan segala kerendahan hati penulis mengucapkan rasa terima kasih dan semoga laporan ini dapat bermanfaat bagi penulis maupun kita bersama.

Sungailiat, 15 Februari 2023

Hormat kami,

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1 Latar Belakang

Salah satu rempah-rempah yang paling banyak digunakan adalah lada, baik dalam hal kontribusinya terhadap devisa negara maupun karena penggunaannya dalam memasak. Salah satu negara penghasil lada teratas di dunia adalah Indonesia. Lada hitam dari Indonesia disebut "lada hitam Lampung" sedangkan lada putih disebut "lada putih Muntok" di negara lain[1].Lada putih muntok atau "*Muntok white pepper*" berada di salah satu wilayah Indonesia yaitu provinsi Bangka Belitung, kabupaten Bangka Barat, kota Mentok. Pengepul di daerah ini memerlukan pemeriksaan lada dan benda asing seperti daun dan tangkai, dengan cara tradisional atau secara manual. Cara ini sering kali tidak efesien karena presisi setiap orang berbeda-beda. Penentuan standar mutu lada telah diatur dalam standar mutu nasional (SNI). Untuk standar komiditi buah lada pada SNI-01-0004-1995 dan SNI-01-0005-1995 merupakan dua kategori standar yang telah dibuat oleh Badan Standardisasi Nasional. Standar tersebut didasarkan pada kadar air, kadar lada kehitaman, kandungan kontaminasi jamur, kandungan benih ringan, kandungan kontaminasi hewan, warna, dan kandungan benda asing. Diantara standar mutu lada diatas, kami mengambil topik benda asing pada lada. Sehingga menciptakan sistem penentu mutu lada berdasarkan *persentase* benda asing bisa menjadi opsi untuk pengembangan pemeriksaan lada secara otomatis sekaligus menentukan mutu lada tersebut.

Ranah pengenalan objek dan pendeteksian dari zaman ke zaman semakin maju sesuai perkembangan ilmu pengetahuan dan teknologi. Pengembangan pengenalan dan deteksi objek dapat dibagi menjadi dua *fase*[2]: *fase* tradisional, di mana manusia terus memainkan peran penting dalam menasihati sistem tentang apa yang perlu dilakukan dalam proses deteksi yang dilakukan oleh sistem dan selama itu metode mengenali dan mendeteksi objek masih dilakukan

secara manual. Selain itu, *fase* pembelajaran mendalam teknik pembelajaran mesin, yang memungkinkan algoritma sistem untuk belajar dan berkembang secara mandiri dengan menggunakan informasi yang disajikan dan pengetahuan yang diperoleh sebelumnya tanpa keterlibatan manusia yang diperlukan[3],[4]. Pada akhir-akhir ini , *Deep Learning* adalah topik yang sedang *tren* yang lebih sering digunakan untuk mengembangkan deteksi objek, wajah, dan jenis lainnya. *Detektor* yang disukai untuk mendeteksi objek dan wajah termasuk *You Only Look Once* (YOLO), *Fast-RCNN*, dan *Faster-RCNN*. *Detektor* ini memiliki deteksi *presisi* namun ringan yang canggih di berbagai bidang [5].

Pada proyek akhir ini kami akan membuat sistem penentu mutu lada berdasarkan persentase benda asing berbasis *You Only Look Once* (YOLO). YOLO merupakan cabang *Convolutional Neural Network* (CNN). Salah satu metode deteksi objek yang sering digunakan. Sistem ini akan mendeteksi objek lada dan benda asing dan menentukan mutu lada tersebut dengan menggunakan metode *You Only Look Once* (YOLO). Terdapat kelemahan pada YOLO seperti memerlukan arsitektur yang kuat dan aktual, *Desain arsitektur* yang lebih kecil, yang dikenal sebagai YOLO-*tiny*, dibuat dengan mempertimbangkan ukuran *ekstraktor fitur* yang sangat besar karena detail pelatihan yang lebih panjang juga akan memakan waktu lama[5]. Kedalaman lapisan *konvolusional* berkurang dalam versi kompak arsitektur YOLO untuk meningkatkan kecepatan pelatihan. Meskipun YOLO-arsitektur kecil lebih kecil dari YOLO dan menggunakan lebih sedikit lapisan konvolusi, namun dapat menunjukkan deteksi yang memadai[6].

Penelitian penggunaan metode *You Only Look Once* (YOLO) telah banyak dilakukan, namun untuk penelitian pendeteksian lada masih sedikit sekali. pada tahun 2021 mahasiswa D4 telah membuat sistem pendeteksian buah lada, akan tetapi menggunakan metode *Convolutional Neural Network* (CNN) dan hanya bisa mendeteksi satu buah lada dan lada yang dideteksi berbentuk dalam satu tangkai lada, bukan berbentuk butiran biji lada. Oleh karena itu, pada proyek akhir ini kami mengembangkan sistem pendeteksian lada tersebut dengan judul proyek akhir yaitu Penentu Mutu Lada Berdasarkan Persentase Benda Asing

Berbasis Pengolahan Citra. Sistem ini menggunakan metode *You Only Look Once* (YOLO). Dengan membuat sistem ini, diharapkan sistem bisa mendeteksi lebih dari satu lada dan mendeteksi lada dalam bentuk butiran. Selain mendeteksi lada sistem ini juga bisa mendeteksi benda asing pada lada seperti daun dan tangkai lada sehingga dapat menentukan mutu lada berdasarkan persentase benda asing pada lada.

1.2 Rumusan Masalah

Permasalahan ada pada pengepul di daerah Bangka ini yang masih memerlukan pemeriksaan lada dengan cara manual sehingga didapatkan rumusan masalah sebagai berikut:

1. Bagaimana membuat sistem penentu mutu lada berdasarkan persentase benda asing berbasis pengolahan citra dengan metode *You Only Look Once* (YOLO).
2. Bagaimana cara sistem penentu mutu lada dapat mengenali objek buah lada dan benda asing pada buah lada.
3. Bagaimana cara sistem penentu mutu lada dapat membuat persentase benda asing sebagai penentu mutu lada.

1.3 Batasan Masalah

Batasan masalah pada sistem penentu mutu lada berdasarkan persentase benda asing ini adalah sebagai berikut:

1. Sistem ini mendeteksi buah lada dan dapat menentukan mutu buah lada berdasarkan persentase benda asing pada buah lada sesuai dengan SNI.
2. Sistem ini mendeteksi buah lada dan benda asing pada lada seperti daun dan batang serta dapat menentukan mutu buah lada berdasarkan persentase benda asing pada buah lada tersebut.
3. Biji lada yang diperlukan untuk pengujian adalah biji lada dalam bentuk butiran dan biji lada tidak berdekatan.

1.4 Tujuan Proyek Akhir

Adapun tujuan pembuatan sistem penentu mutu lada berdasarkan persentase benda asing berbasis pengolahan citra adalah:

1. Membuat sistem penentu mutu lada berdasarkan persentase benda asing berbasis pengolahan citra menggunakan metode *You Only Look Once* (YOLO).
2. Dapat membaca image buah lada dan benda asing pada buah lada.
3. Menampilkan hasil penentu mutu buah lada pada layar monitor.
4. Dapat menerapkan kecerdasan buatan dengan metode YOLOv3-*tiny*.

BAB II
DASAR TEORI

2.1 Standar Mutu Lada Indonesia

Penetapan baku mutu lada ditetapkan sesuai dengan baku mutu nasional yaitu SNI. Mengingat semakin besarnya peran penjaminan mutu atau standardisasi hasil mutu dalam komersialisasi produksi pertanian di masyarakat internasional, maka penerapan standarisasi mutu hasil, Petani kecil semakin dituntut untuk menerapkan standar mutu ISO 9000, ISO 14000, HACCP, dan SPS agar dapat bersaing di pasar global.

Untuk menghindari hal tersebut, sedang dilakukan upaya standarisasi kualitas produk lada, mulai dari penyediaan bahan baku atau bahan olahan hingga pengemasan dan pemasaran hasilnya, sehingga eksportir dapat memenuhi baku mutu produsen (petani) dan dipasarkan baik secara individu maupun kelompok/kemitraan. Penekanan pada model kemitraan dengan perusahaan mitra atau pihak lain, mulai dari pengolahan pasca panen hingga pemasaran, diperlukan untuk membantu pengembangan sumber daya yang ditujukan untuk pertumbuhan petani dan organisasi pertanian untuk mencapai tingkat standar kualitas lada yang tinggi. Dua jenis standar produk lada telah dirilis oleh Badan Standardisasi Nasional: standar mutu lada putih (SNI 01-0004-1995) dan standar mutu lada hitam (SNI 01-0005-1995).

Tabel 2. 1 Syarat Mutu Lada Putih

No	Syarat Mutu Lada Putih	Mutu I	Mutu II
1	Cemaran Binatang	Bebas dari serangga hidup/mati, bebas dari bagian yang berasal dari binatang.	Bebas dari serangga hidup/mati, bebas dari bagian yang berasal dari binatang.

2	Warna	Putih Kekuningan	Putih Kekuningan, putih keabu-abuan atau putih kecoklatan
3	Kadar Benda Asing	Maks 1,0 (%)	Maks 2,0 (%)
4	Kadar Biji Enteng	Maks 2,0 (%)	Maks 3,0 (%)
5	Kadar Cemaran Kapang	Maks 1,0 (%)	Maks 2,0 (%)
6	Kadar Lada Kehitam-hitaman	Maks 1,0 (%)	Maks 2,0 (%)
7	Kadar Air	Maks 13,0 (%)	Maks 14,0 (%)

Jenis lada terdapat dua macam yaitu lada putih dan lada hitam. Untuk mutu lada putih telah ditentukan dengan SNI 01-0004-19995 yang ditunjukan pada tabel 2.1 diatas. Mutu lada putih terdapat 2 mutu yang ditentukan yang pertama, mutu lada putih I dan mutu lada putih II. Syarat mutu lada putih yaitu dari cemaran binatang, warna, kadar benda asing, kadar biji enteng, kadar cemaran kapang, kadar lada kehitam-hitaman, dan kadar air. Setiap persyaratan terdapat mutu lada putih I dan mutu lada putih II yang telah ditentukan.

Tabel 2. 2 Syarat Mutu Lada Hitam

No	Syarat Mutu Lada Hitam	Mutu I	Mutu II
1	Cemaran Binatang	Bebas dari serangga hidup/mati, bebas dari bagian yang berasal dari binatang.	Bebas dari serangga hidup/mati, bebas dari bagian yang berasal dari binatang.
2	Kadar Benda Asing	Maks 1,0 (%)	Maks 2,0 (%)
3	Kadar Biji Enteng	Maks 2,0 (%)	Maks 3,0 (%)

4	Kadar Cemaran Kapang	Maks 1,0 (%)	Maks 2,0 (%)
5	Kadar Air	Maks 13,0 (%)	Maks 14,0 (%)

Begitu pula dengan standart mutu lada hitam yang telah diatur pada SNI 01-0005-1995. Mutu lada hitam yang telah diatur sama dengan mutu lada putih dimana terdapat mutu lada hitam I dan mutu lada hitam II. Syarat mutu lada hitam dapat dilihat pada tabel 2.2 diatas.

2.2 Pemrosesan Gambar Digital

Pemrosesan dan interpretasi foto digital dengan bantuan komputer dikenal sebagai "pemrosesan gambar digital." Gambar berfungsi sebagai *input* untuk pemrosesan gambar, dan gambar yang diproses berfungsi sebagai *output* [7]. Pemrosesan gambar dua dimensi adalah nama umum untuk pemrosesan gambar digital. Pemrosesan gambar digital juga mencakup semua data 2D. Intensitas atau tingkat kelabu bayangan pada posisi tertentu diwakili oleh fungsi F(x,y). Pemrosesan citra digital mencakup operasi yang *input* dan *output*-nya adalah gambar, serta operasi yang mengekstrak properti gambar dan mengidentifikasi objek tertentu[8].

2.3 *You Only Look Once* (YOLO)

You Look Only Once (YOLO) adalah sistem pemilihan objek real-time yang memperlakukan pengenalan objek sebagai masalah regresi tunggal, bergerak dari gambar bertitik ke berbagai kotak identifikasi spasial dan opsi kategori yang sesuai, dan berkonsentrasi pada pemrosesan waktu nyata[9]. Berikut beberapa parameter yang dapat diatur pada YOLO :

1) Batch Size

Jumlah gambar atau data pelatihan yang dimasukkan selama proses pelatihan dikendalikan oleh ukuran *batch* variabel. Prosedur pelatihan data akan berjalan lebih cepat semakin kecil nilai ukuran *batch* yang dipilih. Prosedur pelatihan akan

memakan waktu lama semakin besar ukuran *batch.* Ini terjadi karena lebih banyak ruang penyimpanan diperlukan selama prosedur pelatihan data. Akurasi sistem juga dipengaruhi olehnya. Akurasi akan meningkat karena nilai ukuran *batch* meningkat karena sistem akan mempelajari lebih banyak fitur [9].

2) Learning Rate

Kuantitas berat yang diisi ulang selama proses *backpropagation* ditentukan oleh tingkat pembelajaran. Untuk mencapai fungsi kehilangan terkecil, tingkat iterasi juga ditentukan oleh tingkat pembelajaran. Proses pelatihan bergerak lebih cepat semakin tinggi tingkat pembelajarannya. Namun, diperlukan beberapa upaya untuk mendapatkan nilai tingkat pembelajaran yang ideal, karena jika tingkat pembelajaran terlalu tinggi, nilai fungsi kehilangan dapat berfluktuasi secara tidak menentu [9].

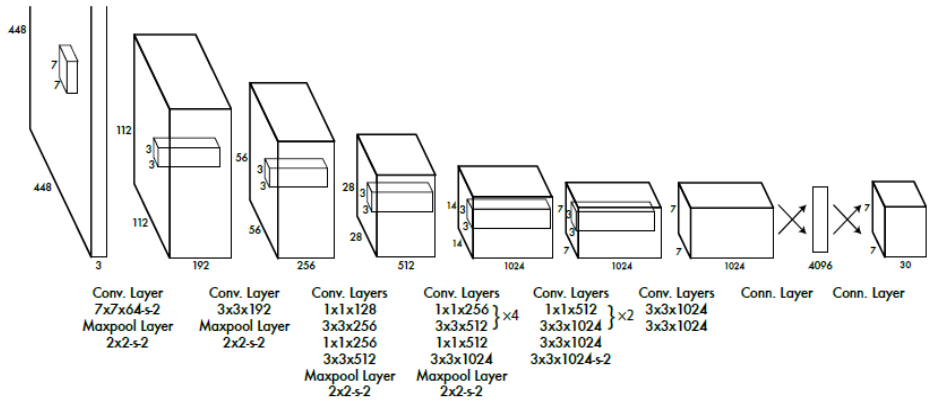
3) Max Batches

Proses mencari tahu berapa banyak iterasi yang akan ada dalam proses pelatihan data disebut *max batches*. Sistem akan memeriksa data pelatihan lebih banyak saat nilai iterasi meningkat. Data pelatihan tidak boleh lebih dari jumlah maksimum *batch*. Jumlah kelas item yang akan dideteksi harus diperhitungkan saat menetapkan nilai maksimum.

Max Batches = number of class x 2000

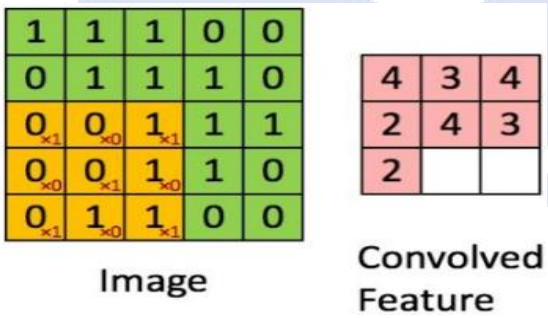
Saat ini, ada beberapa metode objek *detection* yang merupakan pengembangan dari arsitektur jaringan syaraf tiruan *Convolutional Neural Network* (CNN)[10], seperti *Faster R-CNN* [11], *Single Shot Detector* (SSD) [12], dan *You Only Look Once* (YOLO) [13]. Setiap metode tersebut terdapat kelebihan dan kekurangannya. *You Only Look Once* (YOLO) adalah salah satu metode pendeteksi objek tercepat dan paling akurat, mengungguli algoritma deteksi lainnya hingga 2x. Meski kecepatan deteksinya cepat, YOLO tidak memiliki *fase* deteksi terlebih dahulu, sehingga kesalahan dalam mengatur posisi objek besar. Selain itu, YOLO juga kesulitan mendeteksi benda-benda kecil yang berdekatan. Algoritma YOLO merupakan pengembangan dari jaringan syaraf *Convolutional* atau *Convolutional Neural Network* (CNN). Lapisan konvolusional membantu

mengekstraksi fitur dari gambar, dan lapisan yang terhubung sepenuhnya dengan koordinat *output*. YOLO ada 24 lapisan *convolutional* yaitu oleh 2 lapisan yang terhubung sepenuhnya. Berikut ini adalah arsitektur yang terdapat pada YOLO:



Gambar 2. 1 Arsitektur *YOLO*

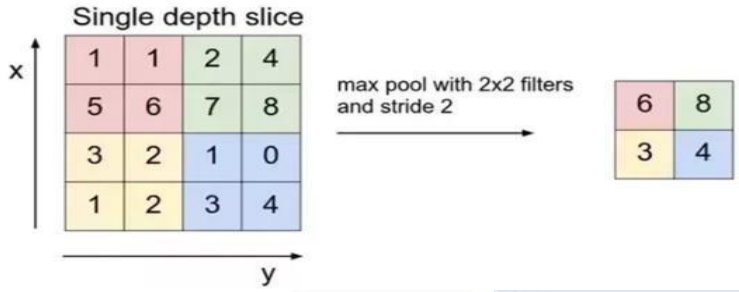
Lapisan konvolusiona*l* melakukan operasi konvolusi pada semua *input* jaringan (piksel gambar). Konvolusi diartikan sebagai proses mendapatkan piksel berdasarkan nilai piksel itu sendiri, yang meliputi matriks yang disebut *kernel* yang merepresentasikan pembobotan [14]. Berikut Operasi konvolusi yang dapat dilihat pada gambar 2.2.



Gambar 2. 2 Contoh Operasi Konvolusi Pada Lapisan *Convolutional*

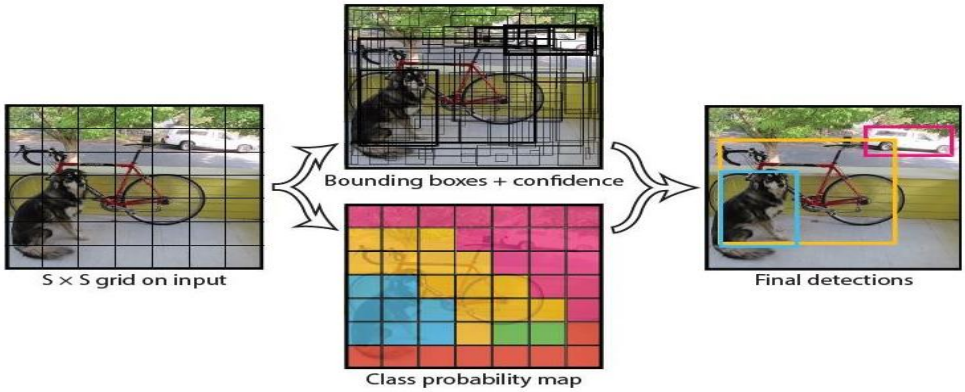
Operasi konvolusi dikerjakan dengan menggunakan perkalian titik pada 3 *matriks citra* awal, dimulai dari sudut kiri atas berukuran 3x3. Dengan operasi konvolusi ini, ada peta fitur yang disertakan dalam fungsi aktivasi *Leaky ReLu*. Fungsi aktivasi membantu jaringan memahami pola teratas dalam kumpulan data. Hasil

penggunaan fungsi aktivasi berlanjut ke level berikutnya. Pada fungsi aktivasi *ReLu Leaky*, ketika *input* kurang dari 0 (negatif), maka *output* mendekati 0, dan ketika *input* lebih besar dari 0, maka *output* setinggi *input*. Ada juga *pooling layer* yang membantu menambah atau mengurangi dimensi peta fitur. Operasi *Max Pooling* dapat dilihat pada gambar 2.3.



Gambar 2. 3 Contoh Operasi *Max Pooling*

Setelah proses *convolutional* dan *pooling layer* dilakukan oleh *feature map*, *feature map* digabungkan menjadi satu dan disisipkan pada *fully connected layer*. *Layer* yang terhubung sepenuhnya, kami menggunakan vektor yang berisi semua peta fitur dari *layer* sebelumnya untuk memasukkan fungsi aktivasi linier yang berisi hasil prediksi dari objek yang dikenali dengan *bounding box*. Alur kerja Algoritma YOLO dapat dilihat pada gambar 2.4.



Gambar 2. 4 Alur Kerja Algoritma

YOLO mengalami beberapa perkembangan pada setiap tahunnya. Pada tahun 2016 memiliki YOLO v2. dua tahun setelahnya terdapat versi YOLO v3. Setahun

kemudian, YOLO v4 dikembangkan lebih lanjut. Ada juga YOLO v5 yang memberi Anda kontrol lebih besar atas ukuran model, penggunaan Hardswish fitur aktivasi, dan augmentasi data.

Penelitian penggunaan metode *You Only Look Once* (YOLO) telah banyak dilakukan. Ada beberapa penelitian yang pernah dilakukan tentang *You Only Look Once* (YOLO). Penelitian yang telah dilakukan oleh Oktaviani Ella Karlina dan Dina Indarti membahas pengenalan objek makanan cepat saji pada video dan *realtime webcam* menggunakan metode *You Only Look Once* (YOLO). Hasilnya menunjukkan akurasi validasi mAP 100% dan kerugian rata-rata 4,6% untuk implementasi algoritma You Look Only Once (YOLO) dalam pengenalan objek makanan cepat saji. Dengan demikian, dapat dikatakan bahwa YOLO berhasil mengenali hal-hal dalam foto makanan cepat saji[15]. Mawaddah Harahap *dkk* membahas You Only Look Once v3. Enam objek diklasifikasikan dalam hasil: mobil, bus, kendaraan yang bergerak, seseorang, dan truk, yang semuanya dipengaruhi oleh sudut kamera dan ukuran objek saat memutar video. Dengan demikian, hasil deteksi objek seringkali tidak akurat. Mirip dengan apa yang dikatakan dalam deskripsi bus, bus memiliki dimensi interior besar yang sama dengan kendaraan mobil. Bentuk truk yang berskala besar, hampir sama dengan bus. CCTV Fix memiliki mAP tertinggi (*mean Average Precision*), yaitu 97%, sedangkan CCTV PTZ memiliki mAP tertinggi, yaitu 99% [16]. Adam Fahmi Fandisyah *dkk* membahas mengenai Deteksi Kapal di Laut Indonesia Menggunakan YOLOv3. Temuan menunjukkan bahwa metode Model 2, yang menggunakan model terlatih dan sekitar sembilan jangkar melalui prosedur *k-means*, memiliki kinerja klasifikasi terbaik untuk menentukan jenis tutup pada kendaraan satelit. Hasil penelitian menunjukkan bahwa model 2 memiliki rata-rata peningkatan profitabilitas dibandingkan model 1 dengan nilai nihil 0,1978. Selain itu, skor mAP Formulir 2 adalah 95,06% dan skor Model 1 masing-masing 94,85% lebih tinggi, daripada Formulir 2. Berdasarkan hasil akurasi rata-rata data pengujian, setiap kategori memiliki nilai lebih besar dari Formulir 1, yang berarti bahwa skor mAP Model 2, masing-masing, 50,41% dan 41,88%

lebih tinggi dari Model 1 [17]. Pembuatan modul untuk robot bergerak yang menggunakan pendekatan YOLO untuk mendeteksi objek manusia dicakup oleh Khairunnas *dkk*. Temuan menunjukkan bahwa modul deteksi objek manusia dalam penyelidikan ini berhasil mengidentifikasi barang-barang manusia. Menurut hasil uji kinerja YOLOv4, total 904 foto diproses dalam 116 detik dengan nilai mAP 87,03%. Modul ini dapat mendeteksi objek tunggal dan banyak objek dalam pengujian untuk menentukan sudut deteksi objek manusia. tes untuk melihat apakah suatu objek dapat dikenali yang dengan benar membedakan antara objek di dalam dan di luar rentang sudut tertentu. Ketika objek masih dalam bingkai, pengenalan objek dengan ID berhasil mengklasifikasikan objek berdasarkan ID mereka [18]. Calvin Geraldy dan Chairisni Lubis membahas tentang pendeteksian dan pengenalan jenis mobil menggunakan algoritma *You Only Look Once* dan *Convolutional Neural Network*. Temuan ini menunjukkan keefektifan *You Only Look Once* dan metode *Convolutional Neural Network* untuk deteksi dan identifikasi mobil. 88,1% dari tes untuk deteksi dan pengenalan mobil berhasil. Pengujian paling efektif untuk pengenalan mobil menggunakan zaman 100 dan tingkat pembelajaran 0,000005. Selama pengujian, aplikasi ini mampu mengidentifikasi beberapa jenis mobil pada satu gambar uji. Hasil deteksi dan pengenalan dapat bervariasi tergantung pada jumlah cahaya yang ada, jumlah mobil yang ada, dan bagaimana eksterior mobil telah diubah[19].

2.4 Darknet-53

YOLOv3 memiliki arsitektur *Darknet-53* yang terdapat 53 *convolutional layers*. *Darknet 53* terdiri dari 3x3 dan 1x1 *filters* dengan *shortcut connections*. Arsitektur ini lebih maju dari pada YOLOv2 dan juga mempunyai *shortcut connections*[20]. Jika dibandingkan dengan versi sebelumnya, yang menggunakan Darknet-19, jaringan dengan 19 lapisan ditambah 11 lapisan untuk mendeteksi objek, yang satu ini sangat sulit dalam mengenali benda kecil. Ini terjadi karena efek pengurangan masukan. Tidak ada lagi blok, melewatkan koneksi dan pengambilan

sampel adalah kelemahan lain dari YOLOv2. Kesalahan ini dicakup oleh Darknet-53[20]. Berikut ini adalah *Feature Extractory yang terdapat pada Darknet-53:*

Tabel 2. 3 *Feature Extractor Darknet-53*

Type	Filters	Size	Output	
Convolutional	32	3x3	256x256	
Convolutional	64	3x3/2	128x128	
Convolutional	32			
Convolutional	64	1x1		1x
Residual		3x3	128x128	----------
Convolutional	128	3x3/2	64x64	
Convolutional	64	1x1		
Convolutional	128	3x3		2x
Residual			64x64	----------
Convolutional	256	3x3/2	32x32	
Convolutional	128	1x1		
Convolutional	256	3x3		8x
Residual			32x32	----------
Convolutional	512	3x3/2	16x16	
Convolutional	256	1x1		
Convolutional	512	3x3		8x
Residual			16x16	----------
Convolutional	1024	3x3/2	8x8	
Convolutional	512	1x1		
Convolutional	1024	3x3		4x
Residual			8x8	----------
Avgpool		Global		
Connected		1000		
Softmax				

Model Darknet-53 ini menggabungkan *Residual Network* dengan *Darknet-19*, yang memiliki lapisan konvolusi dan *residual* dengan ukuran berturut-turut 1x1 dan 3x3, sebagai dasar untuk ekstraksi fitur dari YOLOv2. Pengertian 1x, 2x, 4x, dan 8x adalah bentuk perhitungan blok residu yang dilakukan secara berurutan dari 1 kali blok sisa 1x1 lapisan konvolusi dan lapisan konvolusi berukuran 3x3.

Perhitungan ini juga melewatkan koneksi dari lapisan konvolusi sebelum terjadinya blok residu ke lapisan konvolusi setelah blok residu. *Blok residual* dikalikan dengan 2x jika perhitungan residu blok berikutnya adalah 2x. *Avgpool* tidak aktif[20]. Skala pada input setelah proses augmentasi data adalah 416x416 jika skala disediakan sesuai dengan konfigurasi yang ditetapkan, dan detektor multi-skala dalam proses residu akhir akan menghasilkan 3 skala detektor untuk arsitektur *Darknet-53* dengan output langkah 32x32, 16x16, dan 8x8. *Darknet-53* seperti namanya, menggunakan 53 lapisan, yang masing-masing diisi dengan lapisan normalisasi *batch* dan aktivasi kebocoran ReLU [20].

2.5 *You Only Look Once* (YOLO) v3

YOLOv3 bisa mendeteksi objek dengan *frame rate* yang lebih tinggi daripada YOLO[20] dan YOLOv3 terdapat nilai parameter *Mean Average-Precision* (mAP) pada *metric Intersection over Union* (IoU) 0.5 yang lebih baik daripada SSD yang telah dilatih pada dataset COCO[20], yang berarti memiliki tingkat akurasi yang lebih baik dalam mengenali objek. YOLOv3 menggunakan arsitektur *Darknet-53* yang mempunyai 53 *convolutional layers*. Berikut adalah arsitektur *Darknet-53* yang dapat dilihat pada gambar 2.5.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Gambar 2. 5 Arsitektur YOLOv3

2.6 *You Only Look Once* (YOLO) v3-*tiny*

Pada proyek akhir kami menggunakan YOLOv3-*tiny*. Selain lebih akurat dari model sebelumnya, *Tiny*YOLO juga lebih ringan, lebih cepat, dan lebih efisien dari YOLO. Jaringan *tiny*YOLOv3 dapat digunakan untuk mengklasifikasikan dan mengenali objek. Dalam arsitektur YOLOv3-*tiny*, proses deteksi dimulai dengan *input* gambar yang akan diekstraksi oleh fitur tersebut, dan kemudian kita memodifikasinya menjadi 2 skala dengan dimensi 13x13 dan 26x26 dari hasil ekstraksi fitur[21]. Ada tiga skala dengan dimensi masing-masing 13x13, 26x26, dan 52x52, berbeda dengan YOLOv3 yang lebih besar[22]. Gambar dibagi menjadi sel kisi 13x13 dan 26x26 oleh *Tiny*YOLOv3. Tiga kotak jangkar hadir di setiap sel kisi, dan setiap kotak jangkar berisi empat koordinat kotak pembatas, 20 skor kelas, dan skor objek. Melalui label argumen, kode yolo-*tiny* v3.py membaca jumlah kelas[23]. Berikut ini adalah struktur *Feature Extractor pada* YOLOv3-*tiny* yang dapat dilihat pada tabel 2.4.

Tabel 2. 4 Struktur *Feature Extractor* YOLOv3-*tiny*

Layer	Type	Filters	Size/Stride	Input	Output
0	Convolutional	16	3x3/1	416x416x3	416x416x16
1	Maxpool		2x2/2	416x416x16	208x208x16
2	Convolutional	32	3x3/1	208x208x16	208x208x32
3	Maxpool		2x2/2	208x208x32	104x104x32
4	Convolutional	64	3x3/1	104x104x32	104x104x64
5	Maxpool		2x2/2	104x104x64	52x52x64
6	Convolutional	128	3x3/1	52x52x64	52x52x128
7	Maxpool		2x2/2	52x52x128	26x26x128
8	Convolutional	256	3x3/1	26x26x128	26x26x256
9	Maxpool		2x2/2	26x26x256	13x13x256
10	Convolutional	512	3x3/1	13x13x256	13x13x512
11	Maxpool		2x2/1	13x13x512	13x13x512
12	Convolutional	1024	3x3/1	13x13x512	13x13x1024
13	Convolutional	256	1x1/1	13x13x1024	13x13x256

14	Convolutional	512	3x3/1	13x13x256	13x13x512
15	Convolutional	255	1x1/1	13x13x512	13x13x255
16	YOLO				
17	Route 13				
18	Convolutional	128	1x1/1	13x13x256	13x13x128
19	Upsample		2x2/1	13x13x128	26x26x128
20	Route 19, 8				
21	Convolutional	256	3x3/1	26x26x384	26x26x256
22	Convolutional	255	1x1/1	26x26x256	26x26x255
23	YOLO				

Ada 23 level dengan merutekan lapisan dalam arsitektur YOLOv3-tiny untuk membangun detektor. Jika dibandingkan dengan ekstraktor fitur *Darknet-53*, struktur awal YOLOv3-*tiny* dimulai dengan 16 filter sementara Darknet-53 dimulai dengan 32 filter.

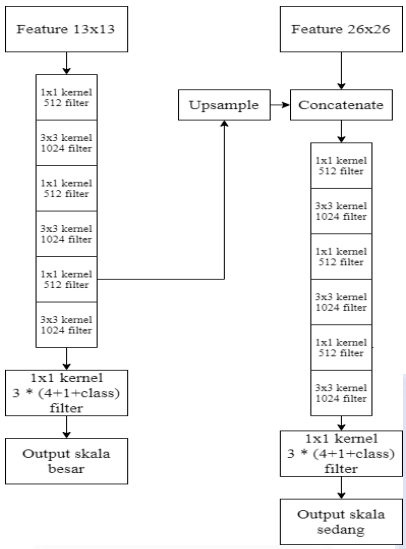
2.7 Multi Scale Detector

YOLOv3-*tiny* memprediksi objek dengan menggunakan dua skala deteksi alternatif. Yang diberikan adalah rasio 13x13 dan 26x26. diberikan rumus YOLOv3 untuk versi besar:

$$NxN *[B*(5+C)]$$

Dalam rumus, N adalah singkatan dari rasio fitur, B untuk jumlah kotak pembatas di dalam sel yang dapat diprediksi, dan C untuk jumlah total kelas. Jumlah jangkar yang diberikan dalam YOLOv3 yang telah disuntikkan pada COCO *Dataset* adalah 3, dan jumlah kelas yang diberikan adalah 80, maka ukuran *kernel* adalah Nx3x255[24]. dapat dihitung dari jumlah prediksi objektivitas dan *offse*t kotak pembatas angka 4, yang sama dengan 1. Peta fitur dari dua lapisan sebelumnya kemudian dipindahkan, dan *upsample* dua diterapkan. Dibandingkan dengan peta fitur sebelumnya, pendekatan ini dapat menghasilkan informasi yang lebih rinci dan informasi semantik yang bermakna dari fitur *upsampled*. Satu-satunya timbangan yang disediakan

untuk YOLOv3-*tiny* adalah 13x13 dan 26x26. Berikut ini adalah gambar tahap proses dari *feature* rasio 13x13 dan 26x26:



Gambar 2. 6 *Multiscale Detector*

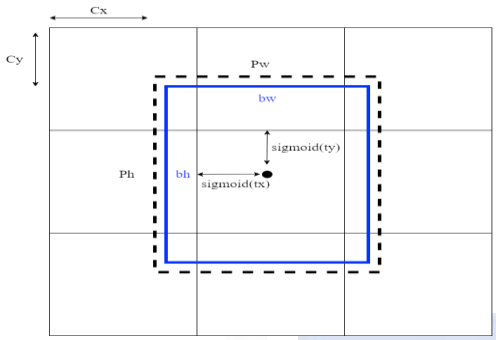
2.8 Bounding Box

Ketika gambar dikirimkan untuk prosedur Kotak Pembatas, target pada jaringan pertama kali dipilih, dan titik tengah gambar kemudian diidentifikasi dengan kotak jangkar langsung di kebenaran dasar[24]. Saat memutuskan kotak pembatas, kita akan memodifikasi rencana sedemikian rupa sehingga kita dapat menyelaraskannya dengan jangkar objek target. Setiap kotak pembatas di YOLOv3-*tiny* memiliki empat koordinat kotak pembatas yang dilambangkan (tx, ty, th, tw). Rumus untuk *bounding box* adalah:

$$b_x = \sigma(t_x) + c_x$$
$$b_y = \sigma(t_y) + c_y$$
$$b_w = P_w\, e^{t_w}$$
$$b_h = P_h\, e^{t_h}$$

Ukuran kotak pembatas adalah (tw,th), ukuran segmentasi kotak jangkar adalah (pw,ph), dan (tx,ty) adalah titik tengah kotak pembatas , dan koordinat *offset* adalah (cx,cy). Nilai koordinat berkisar dari 0 hingga 1 karena

normalisasi. Berdasarkan [B *(5 + C)], YOLOv3-tiny menetapkan nilai objek untuk setiap kotak pembatas dari tiga jangkar setiap skala. Berikut ini adalah contoh penjelasan *bounding box* yang dapat dilihat pada gambar 2.7.



Gambar 2. 7 *Bounding Box*

Hanya hasil prediksi yang di atas ambang batas yang akan digunakan untuk mendapatkan kotak pembatas dan kelas objek dengan probabilitas tinggi. Ketika kotak pembatas digandakan, *Non-max Suppression* (NMS) membantu mencegah salinan ditempatkan di lokasi yang tidak diinginkan.

2.9 Pelatihan Jaringan

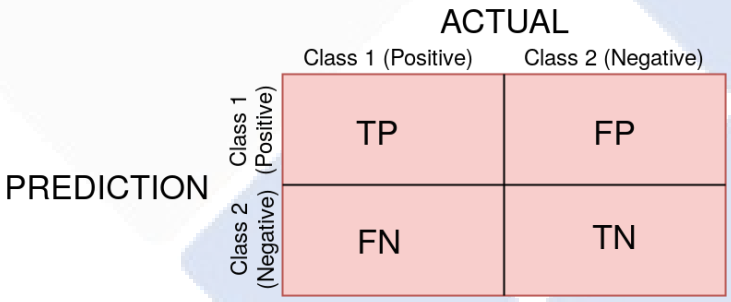
Pembelajaran jaringan berlangsung selama pelatihan jaringan, yang mempersiapkan jaringan untuk melaksanakan perintah. Jaringan akan mencari bobot ideal selama prosedur pelatihan yang dapat memberikan kinerja pekerjaan yang memuaskan. Bobot terbaik akan ditentukan melalui perhitungan menggunakan *input* himpunan data dan sejumlah parameter tambahan, juga dikenal sebagai *hyperparameter*. Ukuran dan kualitas himpunan data dan pemilihan pengaturan *hyperparameter* menentukan berapa lama waktu yang dibutuhkan jaringan untuk menyelesaikan proses pelatihan.

Dalam YOLO v3, beberapa *hyperparameter*, termasuk ukuran gambar input, ukuran *batch*, subdivisi, *batch* maksimum, dan tingkat pembelajaran, harus diklarifikasi lebih lanjut. Seberapa cepat jaringan dapat mengubah bobotnya tergantung pada tingkat pembelajaran. Jumlah tingkat pembelajaran yang besar

akan mempercepat pelatihan tetapi mengurangi peluang untuk menghasilkan nilai bobot yang paling baik. Sebaliknya, nilai tingkat pembelajaran yang kecil akan memperlambat pelatihan tetapi meningkatkan peluang untuk mendapatkan nilai bobot yang lebih baik.

2.10 Pengujian Peforma Jaringan

Confusion Matrix adalah salah satu dari banyak tes kinerja yang dapat digunakan untuk menentukan apakah jaringan berhasil. *Confusion Matrix* adalah gambaran umum hasil prediksi masalah klasifikasi. *Confusion Matrix* membandingkan klasifikasi keseluruhan peristiwa yang benar dengan peristiwa yang benar-benar terjadi (*True Positive*) dan yang tidak (*True Negative*), dan klasifikasi palsu keseluruhan dari suatu peristiwa dengan peristiwa yang benar-benar terjadi (*False Positive*) dan yang tidak *False Positive* (False Negative) [25]. Berikut ini adalah contoh *Confusion Matrix* yang dapat dilihat pada gambar 2.8.



Gambar 2. 8 *Confusion Matrix*

Karena nilai ini ditentukan dari jumlah total peristiwa yang diklasifikasikan dengan benar tetapi tidak terjadi, ada banyak kemungkinan peristiwa/klasifikasi peristiwa yang seharusnya tidak teridentifikasi, sehingga tidak dapat menemukan nilai negatif yang sebenarnya. Dari matriks konfusi, nilai presisi, *recall*, dan skor F1 dapat dihitung dengan menggunakan rumus berikut:

$$Precision = \frac{TP}{TP+FP} \times 100\%$$

$$Recall = \frac{TP}{TP+FN} \times 100\%$$

$$F1\ Score = 2\ x\ \frac{(Precision\ x\ Recall)}{(Precision+Recall)}$$

Akurasi adalah keahlian model untuk mendeteksi objek yang relevan sebagai

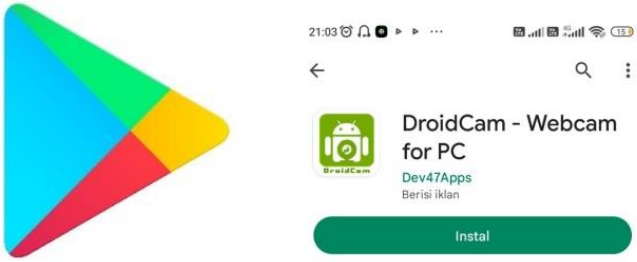
prediksi persentase yang benar, dan *recall* adalah keahlian model untuk mendapatkan semua objek yang hadir sebagai nilai persentase positif yang benar dapat dikenali di semua kebenaran dasar. Skor F1 adalah rata-rata harmonik presisi dan daya ingat.

2.11 DroidCam

DroidCam memiliki antarmuka yang mudah digunakan yang ditujukan untuk pengguna. Pengguna PC atau laptop yang kekurangan *webcam* atau kamera dapat menggunakan aplikasi ini sebagai penggantinya [26]. Karena mereka telah dibuat sesederhana dan sesederhana mungkin, penerapan, tata letak, dan operasi fitur semuanya mudah dipahami . Alih-alih menggunakan kamera atau webcam pada PC *desktop* atau laptop yang menjalankan *Windows*, aplikasi ini menggunakan *smartphone*. Ada dua metode untuk mengintegrasikan *smartphone* dengan komputer *desktop* atau laptop menggunakan *Droidcam Wireless Webcam*. Yang pertama terhubung ke PC dan *smartphone* menggunakan kabel USB, sedangkan yang kedua terhubung melalui jaringan Wireless / Wi-Fi. *DroidCam* akan mengkomunikasikan *IP address* dan *port* yang digunakan pada *smartphone Android* untuk terhubung pada *DroidCam* yang terpasang pada PC atau laptop pengguna, terutama untuk pilihan yang menggunakan koneksi Wi-Fi. Penggunaan *DroidCam* adalah sebagai berikut :

1. Langkah Pertama (di HP)

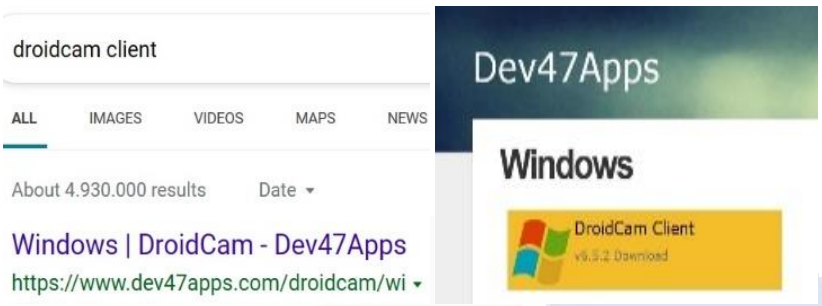
Cari *DroidCam* buatan Dev47Apps di *playstore* pada *android*/HP. Kemudian *install* aplikasi seperti biasanya. Berikut merupakan Logo *playstore* dan *DroidCam* yang dapat dilihat pada gambar 2.9.



Gambar 2. 9 *PlayStore* dan *DroidCam*

2. Langkah Kedua (di PC/Laptop)

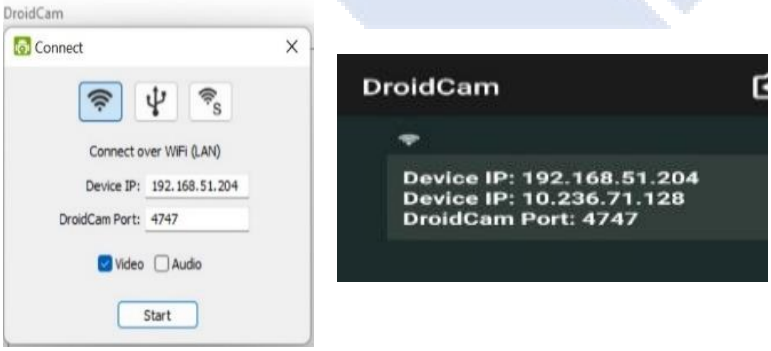
Cari *DroidCam-Client* pada *Google Search*. Kemudian klik *windows | DroidCam-Dev47App*s maka akan muncul ikon untuk mendownload *DroidCam-Client*. Klik ikon tersebut untuk memulai instalasi. Berikut ini adalah cara mengunduh *DroidCam-Client* yang apat dilihat pada gambar 2.10.



Gambar 2. 10 *DroidCam-Client*

3. Langkah Ketiga

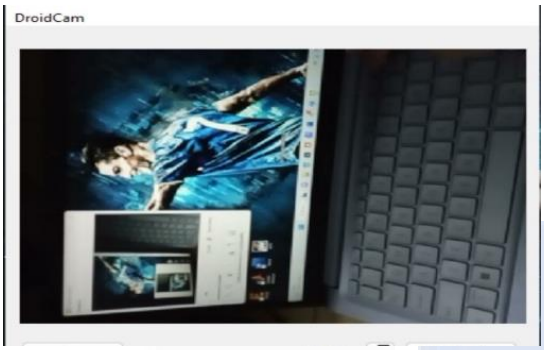
Buka *DroidCam* di *smartphone* dan buka *DroidCam-Client* di PC atau laptop. Pastikan laptop dan *smartphone* memiliki koneksi WIFI yang sama. Aktifkan ikon WIFI Kemudian ubahlah *Device* IP dan *DroidCam Port* pada *DroidCam-Client*. Untuk menghubungkan keduanya maka alamat IP harus sama dengan alamat IP pada *smartphone*, jika telah sama maka *Droidcam Client* pada laptop memiliki koneksi internet yang sama dengan *Droidcam* pada *android*/HP. Berikut ini adalah tampilan dari alamat IP *DroidCam* dan *DroidCam-Client* :



Gambar 2. 11 Alamat IP dan *Port*

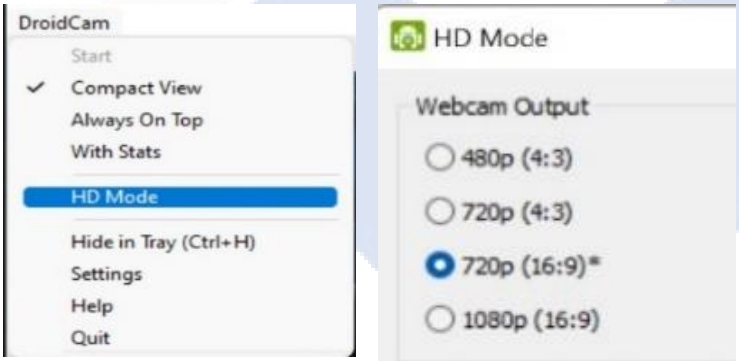
4. Langkah Keempat

Setelah mengisi *Device* IP dan *Port*, selanjutnya klik *start*. Jika berhasil maka kamera HP kamu sudah berfungsi sebagai pengganti *webcam* dan bisa mulai melakukan deteksi ke benda-benda sekitar dengan leluasa. Berikut tampilan hasil dari kamera HP yang telah terhubung dengan *DroidCam* :



Gambar 2. 12 Tampilan Kamera HP

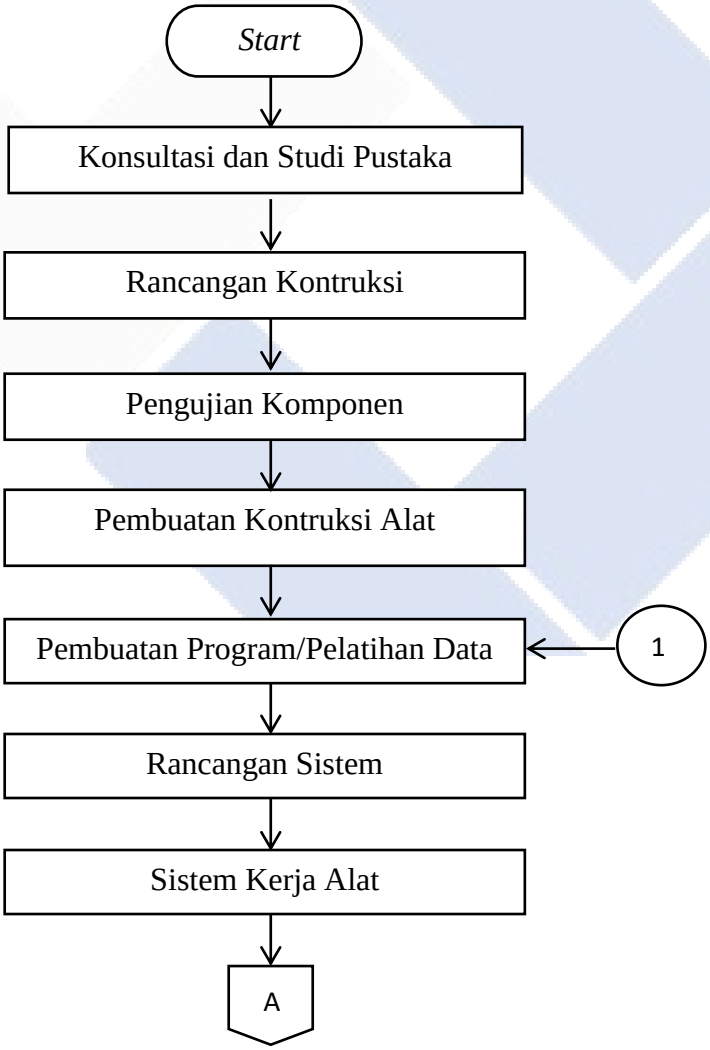
Droidcam juga mempunyai fitur untuk pengaturan resolusi/kualitas video yang lebih baik, yaitu dengan cara mengatur resolusi pada *DroidCam-Client*. Berikut ini adalah cara mengubah resolusi yang dapat dilihat pada gambar 2.13.

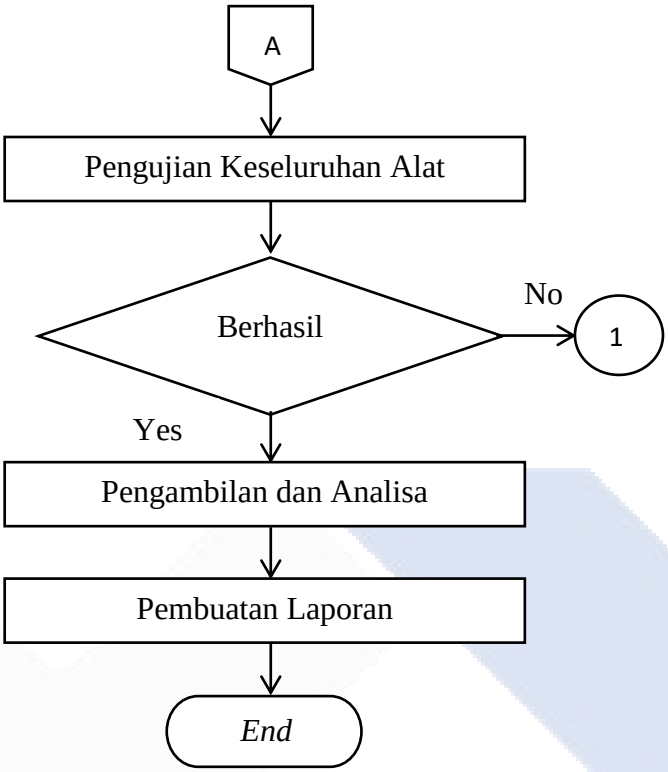


Gambar 2. 13 Resolusi Video

BAB III
METODE PELAKSANAAN

Untuk mempermudah dalam proses pembuatan proyek akhir yang berjudul "Penentu Mutu lada Berdasarkan Persentase Benda Asing Berbasis Pengolahan Citra", maka dibuatlah beberapa langkah-langkah ataupun tahapan proses dalam pengerjaannya. Metode Pelaksanaan dari proyek akhir ini dapat dijelaskan sebagai berikut pada *flowchart* dibawah ini :





Gambar 3. 1 *Flowchart* Metode Pelaksanaan

3.1 Konsultasi dan Studi Pustaka

Pertama, melakukan diskusi segala sesuatu yang berhubungan dengan tugas akhir dengan pembimbing. Setelah konsultasi, dilakukan studi pustaka untuk mempelajari sistem pendeteksi berbasis pengolahan citra yang telah dihasilkan oleh pihak lain untuk mendukung tugas akhir ini. Proses pada tahap ini adalah dengan mencari di internet dan berkonsultasi dengan dosen pembimbing untuk mencari berbagai referensi, dan kami langsung melakukan survei ke seorang petani di Air Limau, Muntok, Bangka Barat. Kami memiliki beberapa pertanyaan, yaitu:

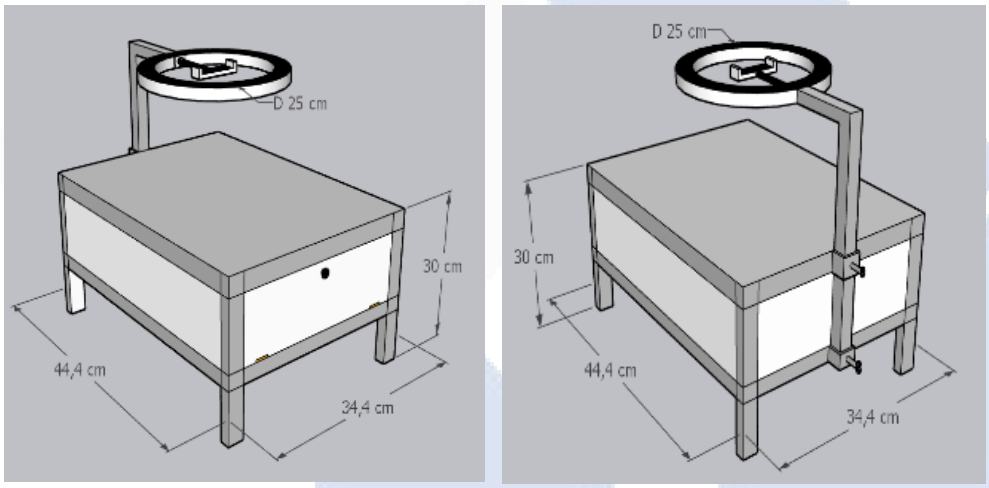
1. Bentuk Biji Lada.
2. Benda asing yang terdapat dalam lada kering.

Kumpulkan data dari konsultasi dengan *supervisor*, *studi literatur*, dan survei. Data yang terkumpul digunakan sebagai acuan untuk proses selanjutnya dalam

produksi tugas akhir “Penentu Mutu Lada Berdasarkan Persentase Benda Asing Berbasis Pengolahan Citra” dan juga sebagai sumber untuk pembuatan makalah tugas akhir.

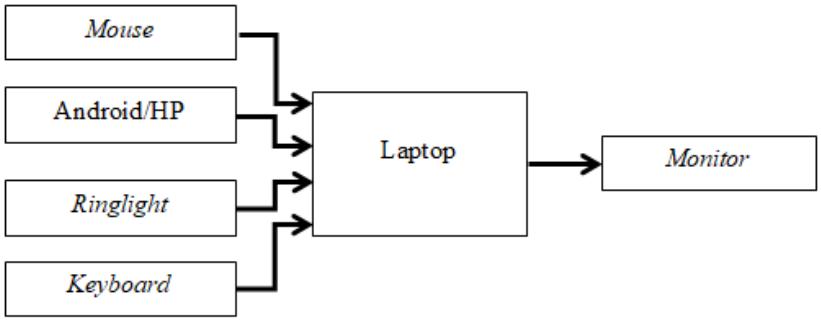
3.2 Rancangan Kontruksi

Perancangan kontruksi dapat diawali dengan membuat desain kontruksi, menentukan komponen-komponen dan membuat diagram blok sistem yang akan digunakan dalam pembuatan Proyek Akhir. Berikut merupakan kontruksi Alat yang telah kami buat yang dapat dilihat pada gambar 3.2.



Gambar 3. 2 Kontruksi Alat

Setelah merancang desain kontruksi alat dan menentukan komponen yang digunakan, proses selanjutnya adalah merancang blok diagram. Dimana setiap blok saling terhubung satu sama lain. Diagram blok memiliki beberapa fungsi antara lain : menjelaskan cara kerja suatu alat secera sederhana, menganalisa cara kerja rangkaian, mempermudah memeriksa kesalahan suatu alat yang dibuat. Berikut ini adalah blok diagram yang telah kami dibuat dan dapat dilihat pada gambar 3.3.



Gambar 3. 3 Blok Diagram Alat

Adapun fungsi dari masing-masing blok diagram adalah sebagai berikut :

1. Blok masukan : Blok masukan terdiri dari kamera *Android*/HP, *ringlight*, *mouse*, dan *keyboard*. Disini kamera berjenis *DroidCam* yang akan mendeteksi lada, dan ringlight berfungsi sebagai penerangan pada saat pendeteksian. Kemudian *mouse* dan *keyboard* sebagai pengatur penginputan gambar ke blok proses.
2. Blok proses : Blok proses terdiri dari Laptop. Laptop sebagai tempat Pembuatan program. Pembuatan program dilakukan menggunakan *Google Colab* dan *Jupyter* yang dimana semuanya akan diproses menggunakan laptop.
3. Blok Keluaran : Blok keluaran terdiri dari *monitor* laptop yang dimana DroidCam pada *android*/HP yang terhubung dengan *DroidCam-Client* pada laptop *monitor*. Dimana *monitor* berfungsi untuk menampilkan hasil deteksi, kemudian *output*-nya berupa objek buah lada atau objek benda asing.

3.3 Pengujian Komponen

Tahap ini merupakan tahapan pengujian komponen yang bertujuan untuk memeriksa komponen yang akan digunakan. Berikut pengujian komponen yang telah dilakukan antara lain:

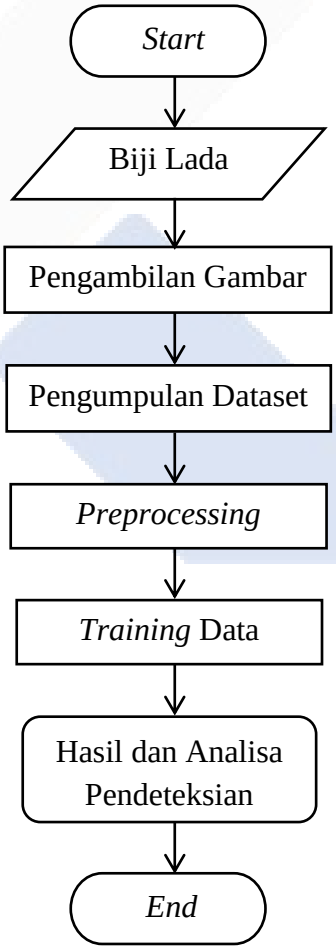
1. Uji coba program dengan metode YOLO menggunakan *Google Colab* dan *Jupyter*.
2. Uji coba DroidCam

3.4 Pembuatan Kontruksi Alat

Tahap ini merupakan proses pembuatan kontruksi alat dengan acuan desain kontruksi yang sudah dibuat. Pembuatan kontruksi dimulai dengan tahap membuat kerangka. Pada kerangka terdapat tempat wadah peletakan biji lada kemudian di atas wadah terdapat kamera *android*/HP sebagai pendeteksi biji lada. Tahap selanjutnya *DroidCam* pada *android*/HP akan dihubungkan dengan *DroidCam-Client* di laptop untuk menjalankan program pada laptop.

3.5 Pembuatan Program/Pelatihan Data

Proses pembuatan program atau pelatihan data ditunjukkan pada *flowchart* pada gambar 3.4 sebagai berikut :

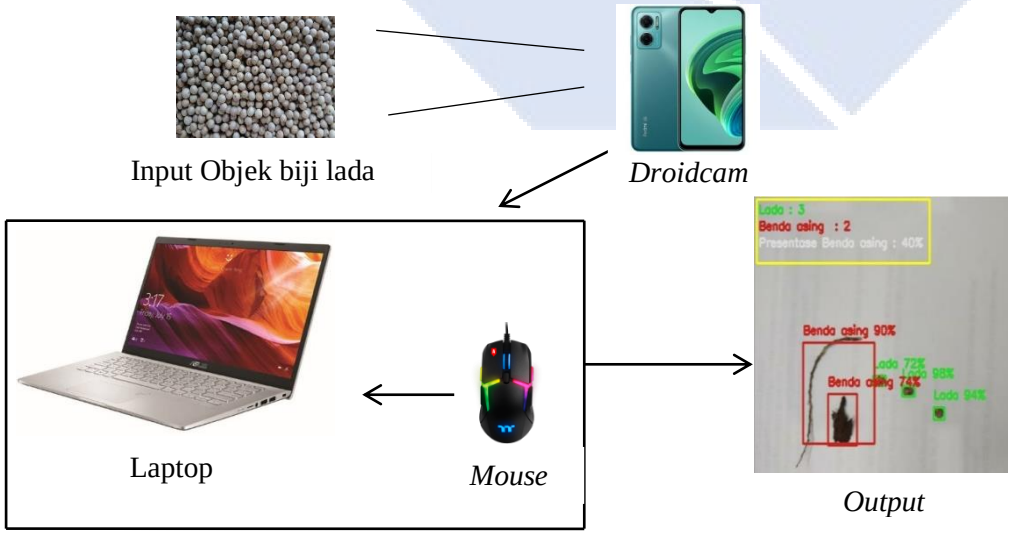


Gambar 3. 4 Skema Pembuatan Program

Sebelum pembuatan program, dilakukan tahapan pengumpulan data citra terlebih dahulu. Citra lada dan benda asing pada proyek akhir ini diambil secara manual menggunakan kamera *android*. Citra yang diambil akan dikumpulkan dalam satu *folder* untuk *dataset*. Setelah pengumpulan data, tahap selanjutnya dilakukan Pra-proses data citra yang terdiri dari perubahan ukuran citra dan pelabelan. Citra yang diambil menggunakan kamera android akan dilakukan *resize* (perubahan ukuran ke ukuran yang lebih kecil). Ukuran citra pada dataset sangat mempengaruhi proses training nanti. Selanjutnya akan dilakukan pelabelan, pelabelan citra adalah tahap awal dimana setiap citra pada *dataset* diberikan label dengan tujuan menyampaikan informasi citra. Proses label dilakukan dengan cara memberikan *bounding box* beserta nama kelas pada setiap objek citra. Setelah pelabelan selesai, tahap selanjutnya adalah pembuatan program atau pelatihan data dengan menggunakan *Google Colab* dan *notebook jupyter* pada aplikasi *anaconda3*, selanjutnya program tersebut akan di uji coba, bisa atau tidaknya mendeteksi biji lada dan benda asing pada lada, serta persentase mutu lada.

3.6 Rancangan sistem

Pada pengerjaan proyek akhir ini memiliki rancangan sistem yang digunakan. Berikut rancangan sistem akan ditunjukan pada gambar 3.5 sebagai berikut :

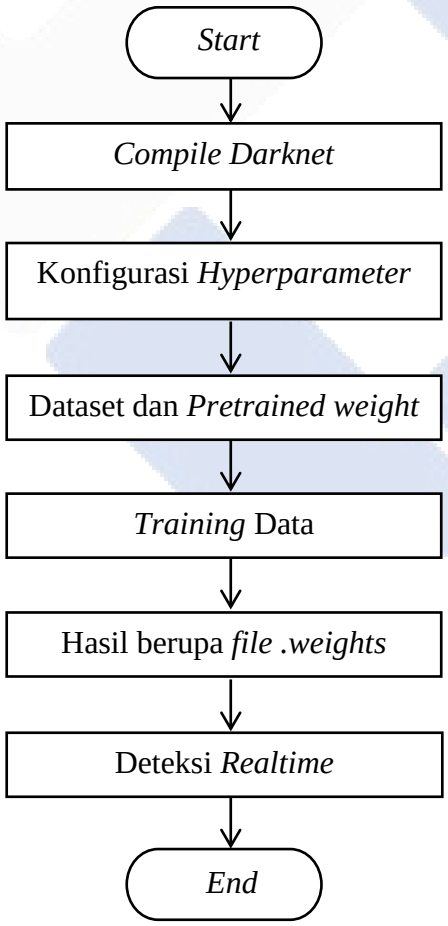


Gambar 3. 5 Rancangan Sistem

Pada proyek akhir ini, kami menggunakan *DroidCam* sebagai pendeteksi objek. Disini kami berfokus pada *image* biji lada dan benda asing pada biji lada yaitu daun dan tangkai lada. *Image* lada dan benda asing akan diambil menggunakan kamera android. Setelah pengambilan *image*, *image* akan dikumpulkan dalam satu *folder* untuk *dataset* pada laptop. *Image* akan di *resize* dan diberi label terlebih dahulu kemudian mulai melakukan pelatihan data. Setelah pelatihan selesai, hasil pelatihan akan diuji coba yang akan menghasilkan *output* seperti gambar diatas.

3.7 Sistem Kerja Alat

Pada pengerjaan proyek akhir ini terdapat cara kerja alat, Berikut ini adalah *flowchart* cara kerja alat yang ditunjukkan pada gambar 3.6.



Gambar 3. 6 Sistem Kerja Alat

Sistem kerja alat pada rancangan sistem ini yaitu menggunakan *DroidCam* sebagai pengganti *webcam* untuk pendeteksian objek dan menggunakan *Google Colab* dan *Jupyter* untuk membuat program deteksi. Tahap pertama yang dilakukan adalah pengambilan gambar atau pengumpulan data citra. Disini kami berfokus pada *image* biji lada dan benda asing pada biji lada yaitu yang terdiri dari daun dan tangkai lada. Setelah pengumpulan data citra maka *image* tersebut akan disimpan dalam satu *folder* yang akan menjadi *dataset* untuk diolah pada tahap selanjutnya yaitu pra-proses data citra. Pada pra-proses data citra dilakukan *re-size* dan pelabelan data citra. Setelah pra-proses data citra selesai, tahap selanjutnya pembuatan program atau pelatihan data di laptop menggunakan *Google Colab* dan Jupyter, *dataset* akan diolah pada pelatihan data dengan menggunakan metode yaitu YOLOv3-*tiny*. Tahapan pertama yaitu *compile Darknet53* ke Google Colab. Tahap selanjutnya, Ubah konfigurasi *Hyperparameter* pada *file* konfigurasi ".cfg" sesuai kelas, dimana terdapat dua kelas yaitu "Lada" dan "Benda_asing". Kemudian masukan *dataset* yang telah diberi label ke Google Colab yang telah dipindahkan ke Google Drive menggunakan fungsi *mount* dan masukan *pretrained weight "darknet53.conv.74*" yang sudah dilatih sebelumnya dengan *dataset ImageNet*. Jalankan proses pelatihan data menggunakan metode YOLOv3-*tiny*. *Training* data dilakukan menggunakan Google Colab dan hasil dari *training* tersebut berupa *weight* yang tersimpan pada *folder backup*. Jika *training* gagal maka tidak ada *file weight* yang tersimpan pada *file backup*. *File weights* yang tersimpan akan digunakan untuk *test detection* pada gambar statis dan *real-time detection*. Untuk deteksi secara *realtime* akan dilakukan di *Jupyter* pada *anaconda3*. sistem di implementasikan untuk mendeteksi dan menentukan mutu biji lada berdasarkan persentase benda asing. Hasil *output* secara *realtime* akan ditampilkan pada layar monitor berupa deteksi objek biji lada dan benda asing. Selain itu, terdapat *Bounding Box* pada kiri atas yang terdiri dari keterangan jumlah lada yang terdeteksi dan jumlah benda asing yang terdeteksi, serta mutu lada berdasarkan persentase benda asing yang didapatkan dari jumlah benda asing dibagi total keseluruhan. Pada tabel 3.1, menampilkan arsitektur jaringan deteksi objek lada dan benda asing YOLOv3-*tiny*

yang akan dibangun pada proyek akhir ini. Berikut merupakan struktur *Feature Extractor* pada *YOLOv3-tiny* :

Tabel 3. 1 Struktur *Feature Extractor* YOLOv3-*tiny*

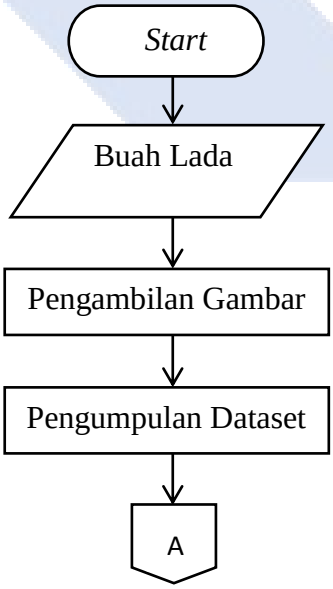
Layer	Type	Filters	Size/Stride	Input	Output
0	Convolutional	16	3x3/1	416x416x3	416x416x16
1	Maxpool		2x2/2	416x416x16	208x208x16
2	Convolutional	32	3x3/1	208x208x16	208x208x32
3	Maxpool		2x2/2	208x208x32	104x104x32
4	Convolutional	64	3x3/1	104x104x32	104x104x64
5	Maxpool		2x2/2	104x104x64	52x52x64
6	Convolutional	128	3x3/1	52x52x64	52x52x128
7	Maxpool		2x2/2	52x52x128	26x26x128
8	Convolutional	256	3x3/1	26x26x128	26x26x256
9	Maxpool		2x2/2	26x26x256	13x13x256
10	Convolutional	512	3x3/1	13x13x256	13x13x512
11	Maxpool		2x2/1	13x13x512	13x13x512
12	Convolutional	1024	3x3/1	13x13x512	13x13x1024
13	Convolutional	256	1x1/1	13x13x1024	13x13x256
14	Convolutional	512	3x3/1	13x13x256	13x13x512
15	Convolutional	255	1x1/1	13x13x512	13x13x255
16	YOLO				
17	Route 13				
18	Convolutional	128	1x1/1	13x13x256	13x13x128
19	Upsample		2x2/1	13x13x128	26x26x128
20	Route 19, 8				
21	Convolutional	256	3x3/1	26x26x384	26x26x256
22	Convolutional	255	1x1/1	26x26x256	26x26x255
23	YOLO				

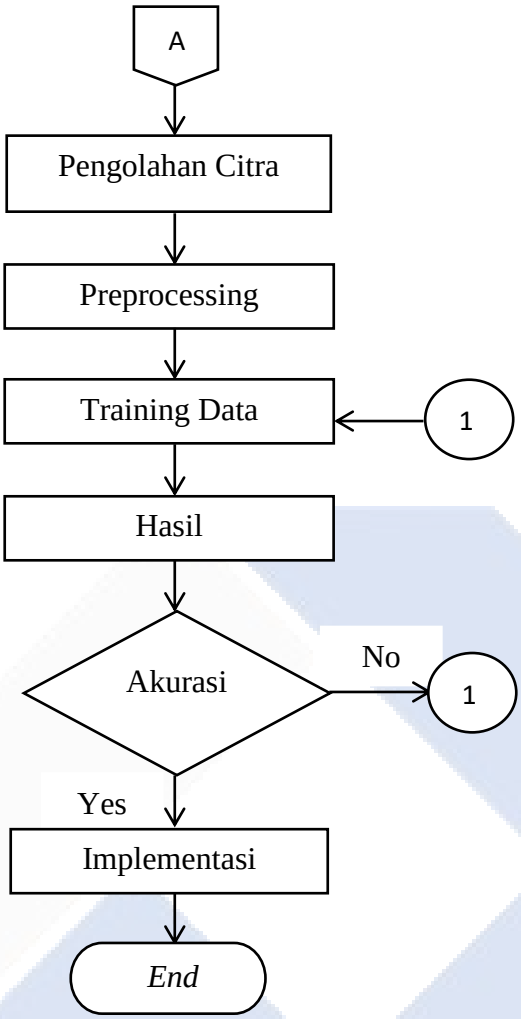
Dalam arsitektur YOLOv3-tiny, proses deteksi dimulai dengan input gambar yang akan diekstraksi oleh fitur, dan kemudian kita memodifikasinya menjadi dua skala

dengan dimensi 13×13 dan 26×26 dari hasil ekstraksi fitur[21]. Gambar dibagi oleh TinyYOLOv3 menjadi sel kisi 13×13, 26×26. Ada tiga kotak jangkar di setiap sel kisi, dan setiap sel tautan berisi empat koordinat batas, 20 klasifikasi kelas, dan satu derajat objek[22]. Menggunakan dua lapisan panduan untuk membangun detektor, hingga 23 lapisan dapat ada dalam desain YOLOv3-*tiny*. Untuk mempercepat proses pengemasan lapisan, pada kecil YOLOv3 tidak menjalankan blok yang tersisa jika YOLOv3 terjadi. Level berikut lapisan 16 adalah di mana YOLOv2 berbeda dari algoritma lain. Kemudian, 26x26 dari 19 layer, Yolov3-*tiny* menggunakan skala kedua untuk membuat *Convolution layer* secara berurutan[23]. Dibandingkan dengan ekstraktor fitur *Darknet-53*, struktur awal YOLOv3-*tiny* dimulai dengan 16 kandidat, sedangkan *Darknet-53* dimulai dengan 32 *filters*.

3.8 Pengujian Keseluruhan Alat

Pada tahap ini dilakukan pengujian keseluruhan alat untuk mengetahui bekerja atau tidaknya sistem kerja alat yang telah dibuat dan apakah berfungsi sesuai yang diinginkan. Berikut pengujian keseluruhan alat yang ditunjukkan pada *flowchart* pada gambar 3.7.





Gambar 3. 7 *Flowchart* Sistem Kerja Alat

Jika pengumpulan data, pra-proses data, dan pengerjaan program/pelatihan data berhasil, maka data-data hasil tersebut akan dikumpulkan dan digunakan untuk menarik kesimpulan dan memberikan informasi untuk pengembangan lebih lanjut.

3.9 Pengambilan dan Analisa Data

Tahapan ini merupakan tahapan dimana data yang berupa gambar yang ambil saat pengujian alat baik deteksi melalui gambar dan deteksi secara realtime secara akan di uji coba dan hasil dari uji coba alat akan dikumpulkan dan dianalisa. Tahap ini bertujuan untuk melihat kekurangan dan kelebihan dari alat yang telah dibuat, baik dari segi kontruksi, sistem deteksi dan pengimplementasian.

3.10 Pembuatan Laporan Proyek Akhir

Tahapan ini merupakan tahapan akhir dari pengerjaan proyek akhir. Tahapan ini bertujuan untuk menyimpulkan keseluruhan dari proses pengerjaan proyek akhir yang dimana tahap ini akan memberikan informasi yang telah didapatkan dari keseluruhan proses pembuatan alat yang dikerjakan.

BAB IV
PEMBAHASAN

Pada bab ini, akan diuraikan proses pengerjaan proyek akhir ini berdasarkan metode yang telah disebutkan pada bab sebelumnya. Proyek akhir ini menggunakan Bahasa pemprograman *python* dan menggunakan aplikasi *Google Colab* dan juga *Jupyter* pada aplikasi *anaconda3*. Berikut tahapan-tahapan yang dilakukan pada pembahasan kali ini antara lain:

1. Pengumpulan Data Citra
2. Pra-proses Data Citra
3. Training/Pelatihan Data
4. Proses Deteksi YOLO v3-tiny
5. Pengujian Performa Jaringan

4.1 Pengumpulan Data Citra

Langkah awal yang akan dikerjakan adalah proses pengambilan citra menggunakan kamera *android*/HP. Didapatkan 212 citra, yang terdiri dari 104 citra lada, 31 citra tangkai lada, 57 citra daun, dan 20 citra variasi (lada, daun, tangkai). Citra akan dibagi menjadi dua kelas “Lada” dan “Benda_asing”. Kelas pertama yaitu biji lada dan kelas kedua benda asing yang terdiri dari daun dan tangkai lada. Semua citra berformat “jpg”. Citra ini akan dikumpulkan dalam satu *folder* untuk dijadikan data atau *dataset* pelatihan jaringan YOLOv3-*tiny*. Pengambilan citra *dataset* dilakukan pada siang dan malam hari. Hasil pengujian yang didapatkan cukup bagus. Namun, untuk mendeteksi objek lada dan benda asing, semakin banyak lada dan benda asing yang diletakkan pada saat pengujian secara *realtime,* maka semakin banyak pula lada atau benda asing yang terdeteksi. Ada pula objek yang tidak terdeteksi saat pengujian, namun tidak sebanyak dari hasil yang berhasil dideteksi. Jika *dataset* yang dilatih banyak, maka hasil pendeteksian pun semakin baik. Selain itu, jumlah *dataset* dan ukuran *input*

gambar yang dimasukan untuk dataset sangat mempengaruhi lamanya saat pelatihan data. Untuk pelatihan dengan dataset 212 dengan ukuran *input* gambar sebesar 1000 x 750, waktu pelatihan yang telah kami lakukan yaitu sekitar 5 jam. Semakin banyak *dataset* maka semakin lama pula waktu pelatihan yang dilakukan. Hal ini sama dengan ukuran *input* gambar *dataset*. Semakin besar ukuran dataset maka semakin lama pula pelatihan data yang dilakukan. Berikut *dataset* yang telah kami ambil menggunakan kamera *android*/HP yang terdapat pada gambar dibawah ini.

Gambar 4. 1 *Dataset* Lada

Pengambilan data biji lada mulai dari 1 butir lada hingga 104 butir biji lada dalam satu *frame* citra.

Gambar 4. 2 *Dataset* Daun

Sedangkan untuk benda asing terdiri dari daun dan tangkai lada dimana citra daun yang diambil mulai dari 1 citra daun hingga 57 citra daun.

Gambar 4. 3 *Dataset* Tangkai Lada

Dan pengambilan gambar lada mulai dari 1 citra tangkai lada hingga 31 citra tangkai lada.

Gambar 4. 4 *Dataset* Campuran Lada, Daun, dan Tangkai

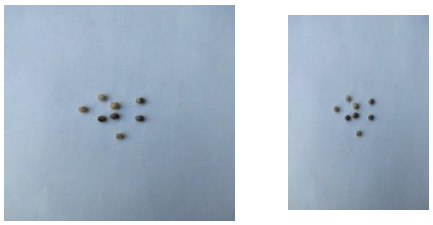
Dan pengambilan gambar campuran yaitu terdiri dari biji lada, daun, dan tangkai mulai dari 1 citra hingga 20 citra.

4.2 Pra-proses Data Citra

Setelah melakukan pengambilan gambar selesai dan dikumpulkan dalam satu *folder*, langkah selanjutnya yaitu pra-proses data citra. Ada dua tahapan pada pra-proses data citra antara lain sebagai berikut :

1. *Resize* Citra

Dataset yang telah dikumpulkan dalam satu *folder* akan di *resize* (perubahan ukuran ke ukuran yang lebih kecil) karena citra yang diambil menggunakan kamera *android*/HP akan mengikuti format ukuran dari *android* itu sendiri. Oleh karena itu, perlu dilakukannya *resize* citra karena ukuran citra pada *dataset* akan sangat memnpengaruhi pada proses *training*/pelatihan data nanti. Semakin besar ukuran gambar, semakin panjang waktu pada proses *training* akan dilakukan. jika spesifikasi laptop yang kurang mendukung juga akan membuat proses *training* sangat lama. Pada proyek akhir ini ukuran dataset yang sesuai format ukuran gambar pada *android* yaitu 4000 x 3000. Ukuran ini terbilang sangat besar yang menyebabkan proses *training* sangat lama. Oleh sebab itu, *resize* citra menjadi ukuran yang disesuaikan yaitu 1000 x 750. Sebelum itu kami mencoba ukuran yang lebih kecil dari 1000 x 750, namun karena objek biji lada pada citra kecil, citra objek tidak terlihat jelas atau buram. Dikhawatirkan sistem akan kebingungan mendeteksi objek lada tersebut. Maka sesuai anjuran dan bimbingan kami memilih ukuran 1000 x 750. Ukuran ini termasuk sudah kecil dan biji lada terlihat jelas pada citra objek. Berikut ini adalah contoh dari hasil Risize yang dapat dilihat pada gambar 4.5



(a) (b)

Gambar 4. 5 Sebelum dan Sesudah Re-Size

2. Pelabelan citra

Setelah *resize* telah dilakukan, langkah selanjutnya yaitu melakukan labeling pada 212 citra. pelabelan citra adalah tahapan dimana setiap citra pada dataset diberikan label dengan tujuan menyampaikan informasi citra. Proses label dilakukan dengan cara memberikan *bounding box* beserta nama kelas pada setiap objek citra. Pada proses labeling ini kami menggunakan aplikasi *makesense.ai*. berikut langkah-langkah melakukan labeling menggunakan *makesense.ai* sebagai berikut :

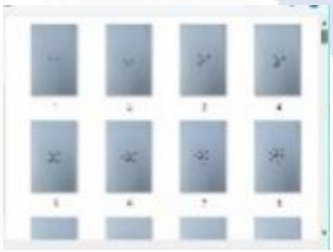
1. Cari aplikasi *makesense.ai* pada *google search*, dimana tampilan awal *makesense.ai* seperti gambar dibawah ini.

Gambar 4. 6 Tampilan Awal *Makesense.Ai*

2. Pada tampilan tersebut ada tulisan *get started*, klik *get started* maka akan mengarah ke tampilan kedua pada gambar dibawah ini.

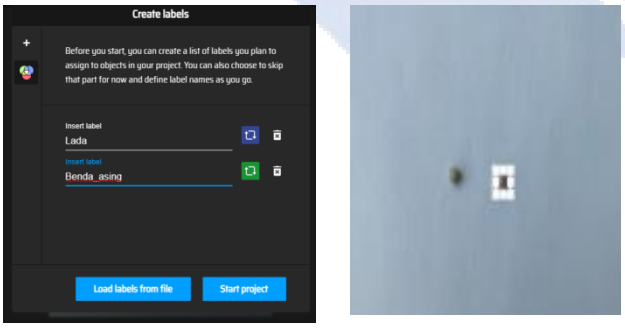
Gambar 4. 7 Tampilan Kedua *Makesense.Ai*

3. Pada halaman diatas, klik *drop image* or klik *here to select from*. Kemudian akan mengarah ke halaman untuk pemilihan *image*. pilih *image* yang ingin dilabeling dengan cara memblok gambar. Kemudian pilih *open* untuk *drop image* di *makesense.ai* seperti gambar dibawah ini.



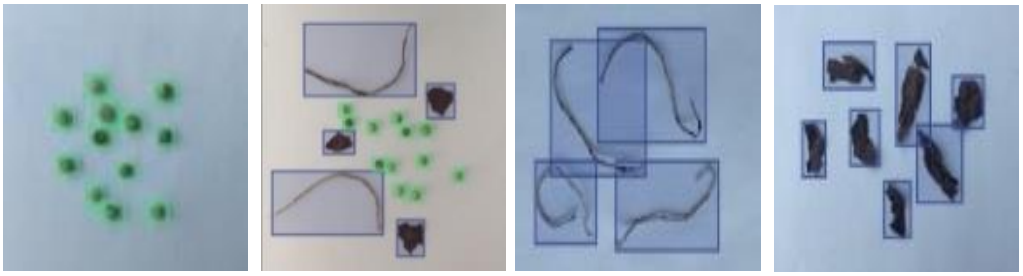
Gambar 4. 8 Tampilan Pemilihan *Image*

4. Setelah *image* dipilih, klik *object detection* dan akan mengarah ke halaman lain yaitu halaman untuk menentukan kelas objek. setelah membuat kelas, klik *start project* dan pelabelan siap dilakukan. Berikut merupakan contoh dari pemilihan kelas dan cara pelebelan :



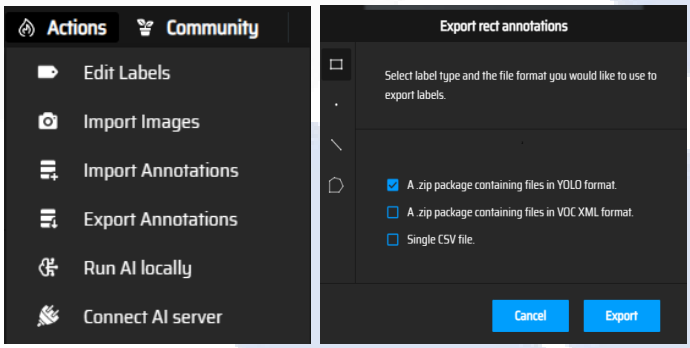
Gambar 4. 9 Pemilihan Kelas dan Cara Label

5. Setelah dilabel, pilih label *image* sesuai kelas yang telah ditentukan. Maka tampilannya akan seperti gambar dibawah ini



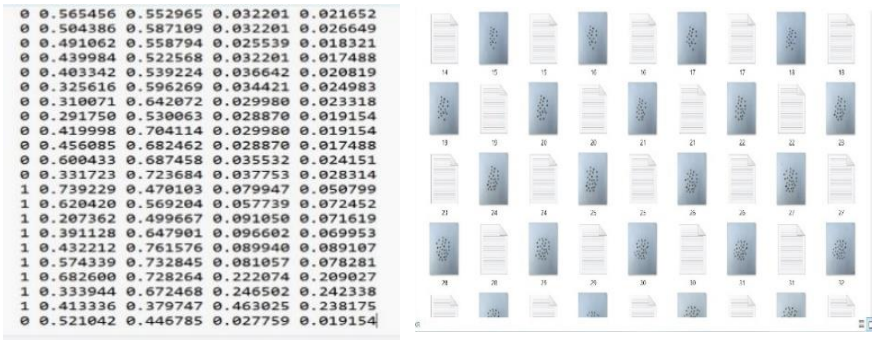
Gambar 4. 10 Tampilan Setelah di-*Labeling*

6. Setelah semua image di-*labeling*, simpan image dengan cara klik ikon *actions* dan pilih *export annotations*. Kemudian pilih format yang diinginkan. Karena kita menggunakan metode YOLOv3-*tiny* maka kita memilih format YOLO. Berikut merupakan contoh dari proses *Export* hasil dengan format YOLO :



Gambar 4. 11 *Export* dan Format YOLO

7. Hasil dari pelabelan menggunakan *makesense.ai* akan berbentuk sebuah *file zip* ".txt". *unzip file* kemudian isi *file* txt ke dalam gambar yang telah dilabeling. Jangan lupa membuat *file* txt untuk kelas. Pada *file* txt tersebut akan menunjukan nomor kelas, koordinat x dan y, serta lebar dan tinggi benda. Nomor kelas 0 untuk kelas Lada dan 1 untuk kelas Benda_asing. Nomor 2 dan nomor 3 untuk menunjukan koordinat x dan y, sedangkan nomor 4 dan 5 untuk menunjukan lebar dan tinggi objek. Berikut ini adalah contoh dari anotasi citra dan isi file txt :



Gambar 4. 12 Isi *File* "txt" dan 212 Citra Beserta Anotasi

4.3 Training/Pelatihan Data

Setelah tahap pra-proses data citra pada *dataset*, tahap selanjutnya yaitu melakukan *training* atau pelatihan data menggunakan metode YOLOv3-*tiny*. Pelatihan data akan dilakukan menggunakan *Google Colab*. Pada pelatihan menggunakan *platform Google Colab* ada beberapa tahap yang harus dilakukan antara lain :

1. Proses *Mounting* dan *cloning*, serta membangun *Darknet*

Dalam proses pelatihan data, pastikan *file* untuk pelatihan data telah disimpan pada sebuah *folder* yang berisi *file* "txt" kelas dan *file dataset* "*zip*" yang nantinya *file dataset* akan di-*unzip* untuk pelatihan data. *Folder* akan diisi pada *Google Drive*. *Google Drive* bisa sebagai tempat penyimpanan data tersebut dan menggunakan *Google Colaboratory* sebagai pelaksanaan untuk menjalankan program *Python*. Untuk menghubungkan *Drive* tempat penyimpanan dengan *Google Colab*, kita menggunakan fungsi *mount* pada program. Setelah proses *mount* berhasil, langkah selanjutnya melakukan *cloning Darknet* ke dalam *Google Colab*. Berikut merupakan contoh kode pemrograman dari proses *mounting* :

```
#clone
!git clone https://github.com/AlexeyAB/darknet

#configure and compile
%cd darknet
!sed -i 's/OPENCV=0/OPENCV=1/' Makefile
!sed -i 's/GPU=0/GPU=1/' Makefile
!sed -i 's/CUDNN=0/CUDNN=1/' Makefile
!make
```

Gambar 4. 13 Kode Program Proses *Mounting* dan Menunjukan *File* Pada *Drive*

```
#Mount
from google.colab import  drive
drive.mount('/content/ gdrive')

# menunjukan file pada drive
!ln -s /content/gdrive/My\ Drive/ /mydrive
!ls /mydrive
```

Gambar 4. 14 Kode *Clone, Configure* dan *Compile*

Pada kode program *configure*, ubah OPENCV,GPU, dan CUDNN menjadi nilai 1 yang dimana berarti kita menggunakan OPEN CV, GPU, dan CUDNN. Adanya penggunaan komponen-komponen tersebut kecepatan *training* akan lebih bagus. Ketiga komponen tersebut merupakan *hyperparameter Darknet*.

2. Konfigurasi *Hyperparameter*

Selain *hyperparameter Darknet*, tentukan juga Konfigurasi *hyperparameter* YOLOv3-*tiny* yang merupakan proses memilih parameter yang dipakai untuk melakukan *training* atau pelatihan data. Ada beberapa parameter yang harus ditentukan antara lain :

a. *Classes* yang merupakan nama kelas yang akan dilatih. Pada pelatihan ini ada dua kelas yang akan dilatih yaitu nomor 0 adalah kelas “Lada” dan nomor 1 adalah kelas “Benda_asing”.

b. *Max_batches* merupakan jumlah literasi maksimal yang digunakan ketika melakukan *training*. *Max_batches* didapatkan dari keseluruhan jumlah kelas dikalikan dengan 2000. 2000 didapatkan dari jumlah iterasi dalam satu kelas. Pada pelatihan ini kita menggunakan dua kelas maka jumlah maksimal iterasi adalah 4000.

c. *Filters* merupakan jumlah *kernel* filter yang diperlukan pada pendeteksian. *Filters* didapatkan dari (jumlah total kelas + 5) * 3. Jumlah total kelas adalah 2 ditambah 5 dikalikan 3 sama dengan 21. Maka jumah *kernel* pada pelatihan ini adalah 21 filter untuk 2 kelas. Didapat kan 5 dari 5 hal pada *Bounding Box* yaitu jumlah kelas, koordinat x, koordinat y, dan lebar dan tinggi objek. sedangkan 3 diambil dari banyaknya prediksi yang dilakukan pada jaringan

YOLOv3. Berikut ini adalah hasil dari konfigurasi Parameter YOLO pada sistem yang telah kami :

Tabel 4. 1 Konfigurasi Parameter YOLO

Nama Parameter	Nilai
Class	2
Max_Batch	4000
Filters YOLO	21

Konfigurasi akan dilakukan pada *file* yolov3-*tiny*.cfg yang telah di*copy* dari Darknet kemudian akan diubah ke *file* yolov3_*training*.cfg yang akan diperlukan saat *training*. Ada beberapa hal yang diubah yaitu *hyperparameter batch, subdivisions, max_batch, classes*, dan *filters*. !cp merupakan perintah kode program untuk menyalin yaitu disini untuk menyalin file yolov3_*training*.cfg. sedangkan !sed merupakan perintah kode program untuk mengubah beberapa garis pada file yolov3_training.cfg. yaitu mengubah *hyperparameter* YOLOv3-*tiny* yang dimana disesuaikan menurut parameter pelatihan yang telah ditentukan. Berikut ini merupakan contoh kode pemrograman untuk konfigurasi *hyperparameter* :

```
# menyalin yolov3_training.cfg
!cp cfg/yolov3-tiny.cfg cfg/yolov3_training.cfg
!sed -i 's/batch=1/batch=64/' cfg/yolov3_training.cfg
!sed -
i 's/subdivisions=1/subdivisions=16/' cfg/yolov3_training.cf
g
!sed -
i 's/max_batches = 500200/max_batches = 4000/' cfg/yolov3_tr
aining.cfg
!sed -
i '610 s@classes=80@classes=2@' cfg/yolov3_training.cfg
!sed -
i '696 s@classes=80@classes=2@' cfg/yolov3_training.cfg
```

```
! sed -
i '783 s@classes=80@classes=2@' cfg/yolov3_training.cfg
!sed -
i '603 s@filters=255@filters=21@' cfg/yolov3_training.cfg
! sed -
i '689 s@filters=255@filters=21@' cfg/yolov3_training.cfg
!sed -
i '776 s@filters=255@filters=21@' cfg/yolov3_training.cfg
```

Gambar 4. 15 konfigurasi *hyperparameter*

3. Menentukan f*ile obj.names* dan *obj.data*

Setelah melakukan konfigurasi *hyperparameter*, tahap selanjutnya yang akan dilakukan adalah menentukan *file* obj.names dan *obj.data*. file obj.names dan obj.data diambil dari data pada *Darknet.* Setelah itu, kita akan membuat *file* obj.names dan obj.data sesuai pelatihan data yang dilakukan yang dimana obj.names berisikan nama kelas yaitu "Lada" dan "Benda_asing". Sedangkan obj.data berisikan jumlah kelas, *path file train*.txt, *test*.txt dan tempat menyimpan *file weight* yang telah telah dilatih. Perintah !echo akan mengubah tulisan sesuai dengan pelatihan data. Kemudian salin obj.names kedalam *folder* yang terdapat *dataset*. Berikut ini adalah kode programnya yang ditunjukan pada gambar 4.16.

```
# membuat obj.names dan obj.data
! echo -e 'Lada\nBenda_asing' > data/obj.names
!echo -
e 'class = 2\ntrain= data/train.txt\nvalid= data/test.txt\nn
ames= data/obj.names\nbackup= /mydrive/Yolov3tiny' > data/ob
j.data

#menyalin obj.names kedalam drive menjadi classes.txt
!cp data/obj.names /mydrive/yolov3-tiny_v3/classes.txt
```

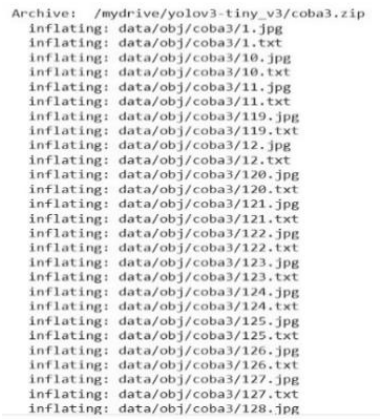
Gambar 4. 16 Kode Program *Obj.Names* dan *Obj.Data*

4. Membuat data/obj

Setelah file obj.names dan obj.data, tahap selanjutnya yaitu membuat *file* untuk menyimpan *dataset* yaitu *file* "obj". setelah *file* obj telah ada didirektori *Darknet*, selanjutnya adalah *unzip file dataset* pada Google drive kedalam directori *Darknet* "data/obj" agar bisa melakukan pengolahan data di direktori *Darknet*. Berikut membuat direktori dan *unzip dataset* yang ditunjukan pada gambar 4.17.

```
# membuat direktori pada Darknet
!mkdir data/obj

#unzip dataset
! unzip /mydrive / yolov3-tiny_v3/coba5.zip -d  data/obj
```

Gambar 4. 17 Kode Program Membuat Direktori Dan *Unzip Dataset*



Gambar 4. 18 *dataset setelah* di-*unzip*

5. Membuat *file train*.txt

Selanjutnya membuat *file train*.txt yang dimana *file* tersebut berisikan nama *path file dataset* beserta anotasinya. *File train*.txt akan menunjukan bahwa dataset citra inilah yanag akan dilatih pada *Darknet*. Pada kode program ini kita menggunakan perintah *glob* untuk menunjukan *path file dataset* citra. Berikut ini merupakan kode dari perintah *Glob* :

```
import glob
images_list = glob.glob("data/obj/coba5/*.jpg")
with open("data / train.txt", "w") as f :
    f.write ("\n".join(images_list))
```

Gambar 4. 19 Kode *Glob*

6. Memuat *Pretrained Weights*

Setelah membuat *file train*.txt, selanjutnya memuat *file trained weight* yaitu dengan mengunduh *pretrained weight* "*darknet.conv.74*" untuk digunakan pada pelatihan data. Pada program ini menggunakan perintah kode !wget untuk mengunduh bobot *pretrained weight*. Berikut ini adalah kode program perintah *pretrained weight* :

```
! wget https://pjreddie.com/media/files/darknet53.conv.74
```

Gambar 4. 20 Kode Program *Pretrained Weight*

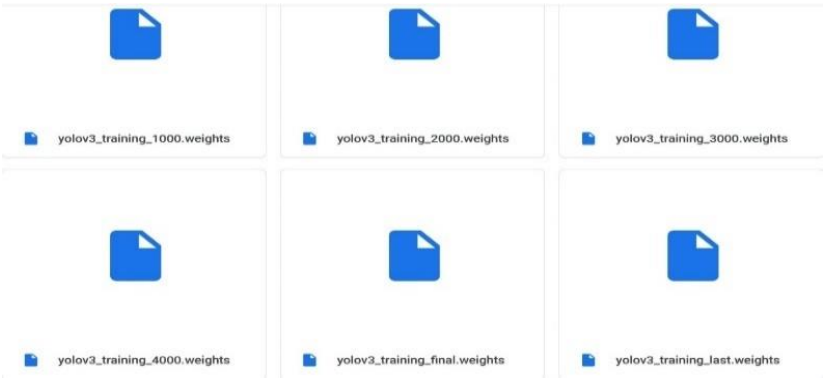
7. Proses Pelatihan Data

Proses terakhir yang dilakukan pada *platform Google Colab* yaitu proses pelatihan data menggunakan *pretrained weight* yang telah diunduh. Berikut ini adalah kode untuk melakukan proses *training* :

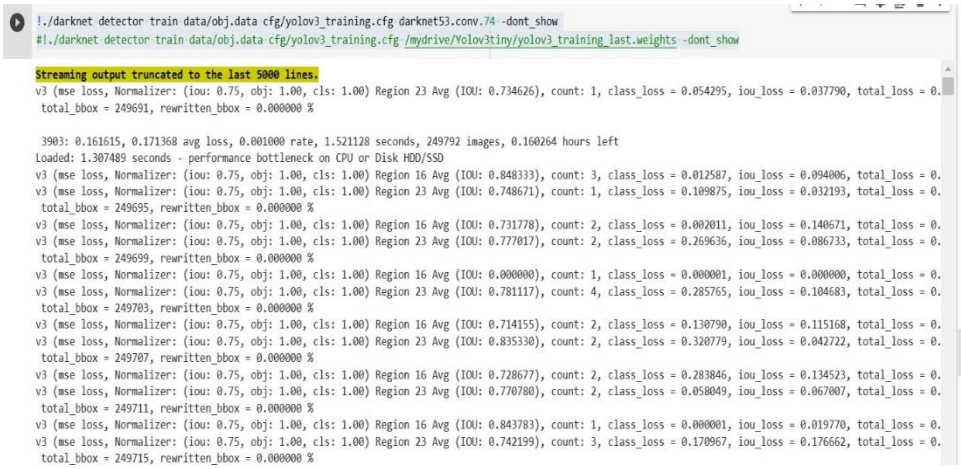
```
!./darknet detector train data /obj.data cfg/ yolov3_trainin
g.cfg darknet53.conv.74  -dont_show
#!./darknet detector train data/obj.data cfg/yolov3_training
.cfg  / mydrive/ Yolov3tiny/yolov3_training_last.weights -
dont_show
```

Gambar 4. 21 Kode Proses *Training*

!./darknet merupakan perintah untuk membuka *file Darknet* yang berisi berbagai fungsi untuk melakukan pelatihan data YOLOv3-*tiny*. Sedangkan *detector train* merupakan fungsi untuk melakukan pelatihan pada *train set* pada *file* "obj.data" menggunakan *hyperparameter* yang telah ditentukan pada *file* yolov3_*training*.cfg dan memerlukan *pretrained weight* "*darknet.conv.74*" yang merupakan proses *Transfer Learning*. Setelah proses pelatihan data atau *training* berhasil maka hasil *training* akan tersimpan dalam bentuk *file weights*. Proses menyimpan *weights* akan dimulai pada iterasi 1000 iterasi pertama. *File weights* akan tersimpan pada *folder backup* pada Google Drive. *File weight* yang tersimpan mulai dari *file* "yolov3_*training_1000.weight*" hingga "yolov3_*training_last.weight*". *File weight* "yolov3_*training_final.weight*" yang tersimpan akan digunakan untuk *test detection* pada gambar statis dan *real-time detection*. Berikut merupakan contoh dari *file waight* yang tersimpan pada *drive* :



Gambar 4. 22 *File Weight* Yang Tersimpan



Gambar 4. 23 Proses Training

4.4 Proses Deteksi YOLOv3-*tiny*

Setelah melakukan proses pelatihan pada *platform Google Colab* dan mendapatkan *file weight* untuk digunakan pada pendeteksian, langkah selanjutnya yaitu melakukan proses deteksi YOLOv3-*tiny*. Sebelum melakukan pendeteksian pastikan terlebih dahulu sudah mengunduh *file* "*yolov3_training_final.weight*" yang merupakan *weight final* untuk pelatihan data yolov3-*tiny* yang telah dilatih pada *Google Colab* dan *file* "yolov3-*tiny*" yang berisi konfigurasi *hyperparameter* pada jaringan YOLOv3-*tiny*, serta *file* "classes.txt" yang terdapat nama kelas yang telah dilatih. Selanjutnya memulai proses deteksi secara *realtime* menggunakan bobot YOLOv3-*tiny* yang telah dilatih dimana proses deteksi akan menggunakan *jupyter notebook* pada *anaconda3*.

Pada pendeteksian secara *real-time* kami menggunakan *DroidCam* sebagai pengganti *webcam* yang dihubungkan ke laptop. Alasan kami menggunakan *DroidCam* yaitu dalam pendeteksian kualitas gambar yang dihasilkan lebih bagus daripada *webcam* yang telah kami coba. Saat menggunakan *webcam* gambar objek yang akan dideteksi buram atau tidak jelas karena ukuran objek yang kecil, hal ini sangat mempengaruhi pendeteksian objek. oleh karena itu, kami memilih opsi *DroidCam* dimana ini menggunakan kamera *android*/HP. Pada *DroidCam* ada fitur fokus dan pengaturan resolusi kamera. Saat melakukan pendeteksian

secara *realtime*, fokus *android* bisa diatur sehingga kamera tidak buram atau tidak jelas dan resolusi bisa diatur.

Untuk pendeteksian secara *realtime* menggunakan library *OPEN CV*. *Computer Vision* merupakan salah satu cabang dari *Artificial Intelegence* (AI), di mana sebuah komputer dilatih agar dapat melihat konten digital berupa citra ataupun video layaknya manusia, yang bertujuan untuk mengekstrak informasi yang terdapat di dalamnya[4]. Dalam banyak aplikasi pemrosesan gambar dan visi komputer, *Application Peripheral Interface* (API) *OpenCV* (*Open Source Computer Vision Library*) *Intel* dapat digunakan. *OpenCV* awalnya merupakan proyek *Intel Research* yang merupakan bagian dari serangkaian inisiatif, termasuk *real-time ray tracing* dan dinding layar 3D, untuk mengembangkan aplikasi intensif CPU. Secara resmi diluncurkan pada tahun 1999. Pustaka *OpenCV* terdiri dari beberapa kelas, metode C/C++, dan algoritma untuk pemrosesan gambar dan visi komputer. Antarmuka untuk *Phyton, Ruby, Matlab*, dan bahasa lain sedang dikembangkan secara aktif. OpenCV dibuat dengan penekanan signifikan pada aplikasi *real-time* dan kinerja pemrosesan. *OpenCV* dibangun dalam C yang telah dioptimalkan sehingga dapat menggunakan CPU *multicor*e. Kami memerlukan pustaka *Integrated Primitives* (IPP) *Intel*, yang berisi rutinitas yang dioptimalkan tingkat rendah di berbagai area algoritmik, jika pengoptimalan otomatis lebih lanjut pada arsitektur *Intel* diperlukan. Jika pustaka diinstal, *OpenCV* akan menggunakan pustaka IPP yang relevan saat *runtime*. Lebih dari 500 fungsi di *OpenCV* mencakup berbagai topik terkait penglihatan, termasuk keamanan, antarmuka pengguna, kalibrasi kamera, visi stereo, robotika, dan inspeksi produk industri. *Library OpenCV* dikembangkan atas dasar gagasan bahwa dengan menawarkan infrastruktur gratis dan terbuka di mana upaya yang berbeda dari komunitas visi dapat dikonsolidasikan dan kinerja dioptimalkan, itu akan memfasilitasi aplikasi komersial visi komputer di bidang-bidang seperti antarmuka manusia-komputer, robotika, pemantauan, biometrik, dan keamanan. Dukungan rutin untuk input, tampilan, dan penyimpanan film dan gambar tunggal disertakan dalam dukungan *OpenCV* yang luas untuk penglihatan. *Microsoft*

membuat satu set API yang disebut *DirectX* yang digunakan oleh *OpenCV* untuk membuat program dan game multimedia. Salah satu tujuan *OpenCV* adalah untuk menawarkan infrastruktur visi komputer yang mudah digunakan yang memungkinkan orang untuk dengan cepat membuat aplikasi visi yang agak kompleks. Berikut ini adalah beberapa tujuan awal *OpenCV:*

a. Dengan menawarkan tidak hanya kode terbuka tetapi juga dioptimalkan untuk infrastruktur visi fundamental, penelitian visi lebih lanjut.
b. Untuk membuat kode lebih mudah dibaca dan ditransfer, infrastruktur standar dibuat yang dapat digunakan pengembang untuk menyebarkan informasi visi.
c. Dengan membuat kode portabel yang dioptimalkan kinerjanya tersedia secara bebas di bawah lisensi yang tidak mengharuskan aplikasi komersial terbuka atau bebas, aplikasi komersial berbasis visi lanjutan.

Kode program deteksi secara *realtime* sebagai berikut :

```
import cv2

import numpy as np
```

Gambar 4. 24 Kode Program *Library*

*Librar*y cv merupakan *library* dalam memprograman saat terjadinya pengoperasian yang menyangkut gambar dan *library numpy as np* merupakan *library* untuk proses perhitungan matematis.

Selanjutnya setelah meng-*import library*, kita melakukan *load* jaringan deteksi objek YOLOv3-*tiny* yang telah dilatih menggunakan fungsi *readNet*(). Ditunjukan pada kode program berikut:

```
net  = cv2.dnn.readNet ("yolov3_training_final.weights ",
"yolov3-tiny.cfg")
```

Gambar 4. 25 Kode Program Readnet

Pada fungsi ini akan dimasukan *file weight* yang telah dilatih pada kode program di *Google Colab* sebelumnya yaitu “yolov3_training_final.weights” dan juga *file*

“yolov3-tiny.cfg”, dimana merupakan *file* yang berisi konfigurasi *hyperparameter* yang telah diubah sebelumnya.

Kemudian akan di-*load* nama kelas objek ke dalam *list* “*classes*” dengan fungsi *read*(). Ditunjukan pada kode program berikut :

```
classes = []

with open("coba5 /classes.txt", "r") as f:

    classes  = f.read().splitlines()
```

Gambar 4. 26 Kode Program *List Classes*

Pada kode program ini terdapat nama *file* penyimpanan *dataset* atau data yang telah kita olah (coba5). Kemudian terdapat *file* classes.txt, dimana merupakan file yang nama kelas yang telah dilatih. Selanjutnya kode program menentukan nilai *confidence_threshold* dan NMS_*threshold*. Ditunjukan pada kode program berikut:

```
net.setPreferableBackend(cv2.dnn.DNN_BACKEND_CUDA)

net.setPreferableTarget(cv2.dnn.DNN_TARGET_CUDA_FP16)



model = cv2.dnn_DetectionModel(net)

model.setInputParams(size=(416, 416), scale=1 / 255,
swapRB=True)

CONFIDENCE_THRESHOLD = 0.3

NMS_THRESHOLD = 0.5
```

Gambar 4. 27 Kode Program *Confidence Threshold* Dan NMS_*Threshold*

Pada program ini nilai *confidence_threshold* yaitu 0.3 yang berarti jika nilai *confidence_threshold* objek lada dan benda asing kurang dari 0.3 maka hasil deteksi akan dihilangkan. Kemudian akan dilakukan NMS (*Non Maximum Supression*) yang bernilai 0.5 yang dimana NMS berguna untuk menghilangkan penempatan *bounding box* yang kurang tepat. Selanjutnya adalah kode untuk deteksi secara *realtime*. Untuk pendeteksian secara *realtime* akan menggunakan fungsi cv2.*videocapture*(). Kode program seperti gambar dibawah ini.

```
cap  = cv2.VideoCapture(1)
if cap.read()[0] is False:
    cap = cv2.VideoCapture(1)
    if cap.read()[0] is False:
        cap = cv2.VideoCapture(2)
while True :
    (grabbed, img ) = cap.read()
    classes, scores, boxes = model.detect (img,
CONFIDENCE_THRESHOLD, NMS_THRESHOLD)
```

Gambar 4. 28 Kode Program Fungsi *Cv2.Videocapture()*

Pada pendeteksian ini kami menggunakan *DroidCam* sebagai pengganti *webcam*. *Capture camera* menggunakan *DroidCam* dapat menggunakan fungsi cv2.*videocapture*. sesuai dengan nama fungsinya, fungsi ini bisa digunakan untuk membaca deteksi video juga. Untuk membedakan pembacaan fungsi ini dibedakan pada pengisian alamat pada *DroidCam.* Biasanya jika alamat diberikan 0, maka akan terhubungkan ke kamera internal pada laptop. Sedangkan alamat yang diberikan 1 maka akan terhubung pada kamera eksternal pada laptop. Oleh karena itu kita beri alamat 1 untuk menghubungkan *DroidCam*. Dan pada kode program ini kami menggunakan *while* Ketika deteksi secara *realtime* sedang dibaca atau dibuka. Fungsi ini digunakan agar program berjalan terus menerus. Kemudian untuk membaca input dari *DroidCam,* kita menggunakan fungsi read().Selanjutnya untuk menghitung jumlah biji lada dan benda asing pada lada, serta untuk menentukan persentase berdasarkan benda asing digunakan fungsi *count.*

```
count_Lada = 0
    count_Benda_asing = 0
    persentase = 0

    for (classid, score, box) in zip(classes, scores, boxes):
        x, y, w, h = box[0], box[1], box[2], box[3]
```

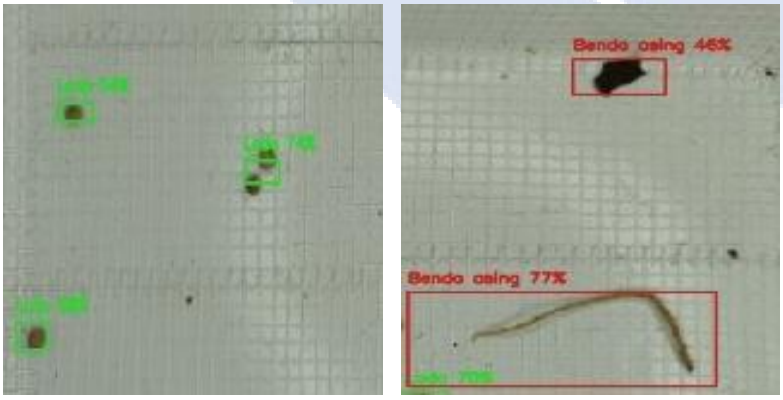
Gambar 4. 29 Kode Program *Count*

Fungsi ini untuk pengembalian jumlah total elemen yang diberikan pada sebuah string atau menghitung jumlah kemunculan suatu nilai. perhitungan biji lada, benda asing, dan persentase benda asing dimulai dengan nol. *Classes* atau *classid* untuk menyimpan id kelas keluaran objek yang terdeteksi. *Score* untuk keluaran banyaknya objek yang terdeteksi, dan *boxes* untuk menyimpan nilai koordinat x, koordinat y, dan lebar (*width*) serta tinggi (*height*) *bounding box* objek yang dideteksi. Selanjutnya membuat *bounding box* objek. pada kode program, kami akan menggunakan logika *if*. Berikut ini adalah contoh dari kode pemrograman untuk menampilkan kotak pembatas :

```
 # Membuat Bounding Box

        if classid == 0:

            count_Lada += 1

            cv2.rectangle(img, (x, y), (x + w, y + h), (42,
217, 39), 2)

            cv2.putText(img, "Lada " + str(int(score * 100)) +
"%", (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (42, 217, 39),
2)
```

Gambar 4. 30 Kode Program Bounding Box

jika *bounding box* berwarna hijau maka terdeteksi biji lada dan jika *bounding box* berwarna merah maka terdeteksi benda asing. Berikut tampilan *bounding box* pada gambar 4. 31.



Gambar 4. 31 Tampilan *Bounding Box*

Untuk mendapatkan hasil *bounding box* maka diperlukan program seperti pada gambar 4.31.

```
 elif classid == 1:

            count_Benda_asing += 1

            cv2.rectangle(img, (x, y), (x + w, y + h), (28, 34,
200), 2)



            cv2.putText(img, "Benda asing " + str(int(score *
100)) + "%", (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (28,
34, 200), 2)



    if count_Lada != 0 or count_Benda_asing != 0:

        persentase = 100 * (count_Benda_asing / (count_Lada +
count_Benda_asing))
```

Gambar 4. 32 Kode Program *Bounding Box*

Jika sebuah lada terdeteksi, maka terhitung 1. Jika 2 atau lebih lada terdeteksi, maka akan dihitung. Begitu pula jika terdeteksi benda asing baik daun maupun tangkai pada lada, maka benda asing tersebut akan dihitung. Kemudian persentase didapatkan dari total keseluruhan yaitu jumlah biji lada yang terdeteksi ditambah jumlah benda asing yang terdeteksi kemudian dibagi dengan jumlah benda asing yang terdeteksi. Dan hasil penjumlahan tersebut dikalikan dengan 100, maka didaptkan lah persentase benda asing (jumlah lada + jumlah benda asing / jumlah benda asing) * 100. Setelah perhitungan dilakukan maka akan diatur *bounding box* yaitu warna, ukuran dan jenis tulisan pada *bounding box*. Berikut ini merupakan kode pemrogram untuk *bounding box* kiri atas :

```
 cv2.rectangle(img, (3, 3), (260, 80), (0,255,255), 2)

    cv2.putText(img, "Lada : " + str(count_Lada), (6, 20),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (42, 217, 39), 2)

    cv2.putText(img, "Benda asing  : " +
str(count_Benda_asing), (6, 40), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
(28, 34, 200), 2)

    cv2.putText(img, "Persentase Benda asing : " +
str(int(persentase)) + "%", (6, 60), cv2.FONT_HERSHEY_SIMPLEX,
0.5, (217, 218, 219),2)
```

Gambar 4. 33 Kode Program *Bounding Box* Kiri Atas

Untuk menggambarkan objek yang terdeteksi, maka kami menggunakan kotak pembatas berbentuk persegi Panjang yang menggunakan fungsi *rectangle*(). Fungsi ini akan menunjukan letak objek yang terdeteksi. Kemudian kita menggunakan fungsi putText() untuk memberikan *bounding box* pada objek agar dapat memberikan informasi bahwa objek tersebut terdeteksi. Variabel "Lada" dan "Benda_asing" untuk menyimpan total objek yang berhasil terdeteksi. Hasil deteksi akan ditampilkan pada *bounding box* pada kiri atas. Variable "Lada" untuk menunjukan total lada yang terdeteksi, "Benda asing" untuk menyimpan total benda asing baik daun maupun tangkai lada yang berhasil terdeteksi, dan "persentase Benda asing" untuk menunjukan berapa persentase benda asing, semakin sedikit benda asing yang terdeteksi maka semakin rendah hasil persentase benda asing yang didapatkan. Berikut ini adalah tampilan dari *bounding box kiri atas* :

Gambar 4. 34 Tampilan *Bounding Box* Kiri Atas

Selanjutnya kode program untuk menampilkan hasil deteksi citra yang telah dipanggil dan dirubah menjadi sebuah variabel.

```
 cv2.imshow('Deteksi Lada', img)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cv2.waitKey(1)
cv2.destroyAllWindows()
```

Gambar 4. 35 Kode Program Variabel

Fungsi cv2.*imshow*() digunakan untuk menampilkan variabel hasil citra yang telah dipanggil. Pada fungsi ini akan menampilkan sebuah jendala untuk menampilkan citra yang ukuran jendelanya akan secara otomatis sesuai dengan ukuran citra objek deteksi. Pada cv2.*imshow* ada beberapa penambahan yaitu nama *window*nya dan diikuti dengan nama variabel yang berisi citra yang akan ditampilkan. Fungsi

cv2.waitKey() merupakan fungsi untuk menahan citra yang sedang kita gunakan. Pada deteksi secara *realtime,* cv2.*waitKey*() tidak boleh bernilai 0 atau diam. Kemudian kode program terakhir yaitu cv2.*destroyAllWindows*() merupakan fungsi untuk menutup semua jendela yang aktif pada sebuah program yang sedang berjalan. Untuk menutup semua jendela kita dapat menekan ‘q’, maka otomatis sistem yang sedang berjalan akan berhenti bekerja.

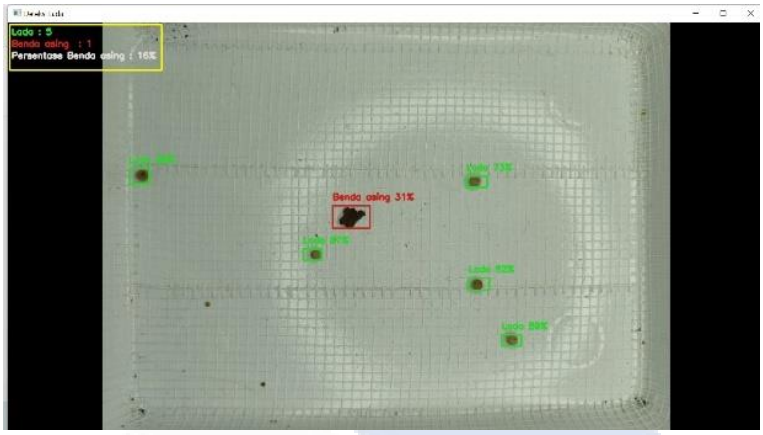
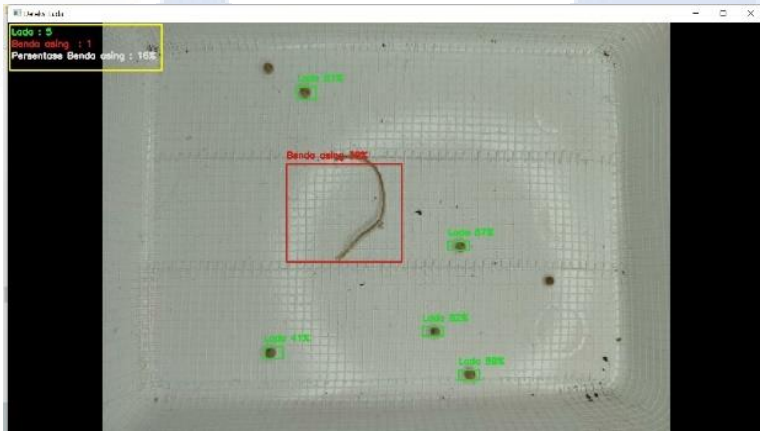
4.5 Pengujian Performa Jaringan

Pengujian performa jaringan YOLOv3-*tiny*, nilai akan diambil dengan menghitung nilai *confusion matrix, recall, precision,* serta *F1 score* pada pengujian deteksi objek lada dan benda asing yang telah dilatih. Pengujian performa akan dilakukan menggunakan 16 gambar dari data hasil uji coba secara *realtime* menggunakan *DroidCam* yang dimana deteksi dilakukan pada penampang berlatar putih dengan ukuran 17,6 x 22,8 cm yang merupakan tempat meletakkan sampel lada yang akan dideteksi. Pada awalnya kami melakukan pengujian menggunakan *webcam* namun hasil deteksi yang didapatkan sedikit dan resolusi kamera *webcam* tidak terlalu bagus sehingga kualitas gambar yang dihasilkan buram. Dan ini menjadi salah satu alasan hasil objek yang berhasil terdeteksi sedikit. Kemudian dengan bimbingan dan saran didapatkan solusi dari permasalahan ini, yaitu menggunakan *DroidCam* sebagai pengganti *webcam*. Pada pemakaian *DroidCam* ini kita memanfaatkan kamera android/HP. Pada saat pengujian menggunakan DroidCam ada fitur bawaan pada kamera *android*/HP yaitu fitur fokus dimana saat mendeteksi jika gambar pendeteksian buram kita dapat mengatur fokus pada kamera sehingga gambar terlihat jelas. Kemudian selain fitur fokus, *DroidCam* juga memiliki pengaturan resolusi kamera sehingga kualitas dapat diatur menjadi lebih baik lagi. Dan dari hasil uji coba yang telah dilakukan menggunakan *DroidCam*, hasil deteksi yang didapatkan berbeda cukup signifikan daripada menggunakan *webcam*.

Pada pengujian ini kami menggunakan jarak yang telah ditentukan yaitu 20 cm dari penampang dimana sesuai dengan ukuran luas penampang. pada jarak ini lah

pendeteksian mendapatkan hasil deteksi yang paling bagus. Jarak deteksi yang paling jauh yaitu 36 cm dimana tidak ada objek yang terdeteksi. Pada jarak 25 cm masih bisa mendeteksi tetapi hasil deteksi hanya satu atau tidak sama sekali. Sedangkan jarak paling dekat yaitu 19 cm, namun jarak tidak sesuai dengan luas penampang. Oleh karena itu, jarak sangat mempengaruhi hasil pendeteksian. Dengan menggunakan lampu *ringlight* yang berukuran 26 cm yang dipasang sejajar dengan peletakan kamera *android*/HP dalam penggunaan *DroidCam*. Lampu *ringlight* digunakan sebagai bantuan pencahayaan pada saat pengujian dilakukan, dan pengujian dilakukan pada siang dan malam hari. Kami melakukan pengujian menggunakan pencahayaan dan tanpa pencahayaan. Dan didapatkan hasil menggunakan bantuan pencahayaan akan membantu dalam hasil pendeteksian. Jika menggunakan pencahayaan, gambar akan lebih jelas dan hasil deteksi yang didapatkan lebih baik dari pada tanpa pencahayaan. Jika tanpa pencahayaan akan terdapat bayangan pada *frame* pendeteksian dan hal itu akan menyebabkan bayangan tersebut akan terdeteksi benda asing padahal tidak ada objek benda asing pada frame pendeteksian. Selain itu, pada saat percobaan pengujian, kami menguji pada siang dan malam hari. Karena menggunakan bantuan pencahayaan hasil pendeteksian tidak berbeda antara siang dan malam, namun jika tanpa pencahayaan pendeteksian tidak akan dapat bekerja dengan baik. Berikut ini adalah beberapa hasil dari pengujian secara *Realtime* yang akan dipaparkan pada table 4.2

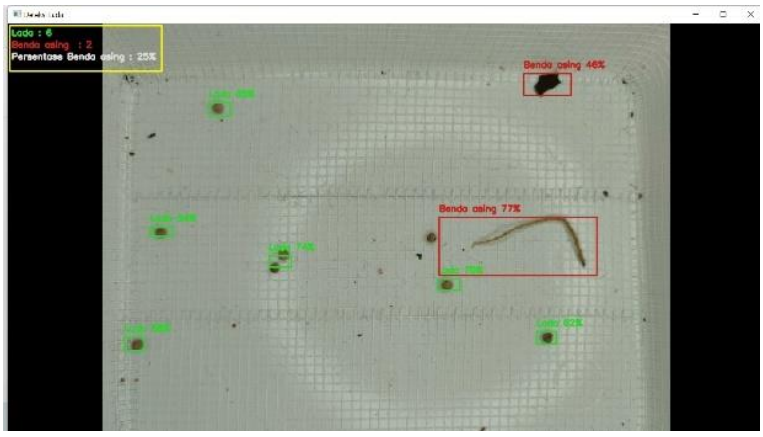
Tabel 4. 2 Hasil Pengujian Secara *Realtime*

Aktual	Hasil
Gambar 3 (siang) Lada : 5 Benda Asing : 1	 Lada : 5 Benda Asing : 1 Pada pendeteksian ini, semua benda berhasil dideteksi.
Aktual	Hasil
Gambar 9 (siang) Lada : 7 Benda Asing : 1	 Lada : 5 Benda Asing : 1 Pada pendeteksian ini, terdapat dua lada yang tidak berhasil dideteksi, hal ini dikarenakan kurangnya kualitas dari kamera atau atau kurangnya *dataset* yang di*training*.
Aktual	Hasil

Gambar 46 (siang)

Lada : 8

Benda Asing : 2



Lada : 6

Benda Asing : 2

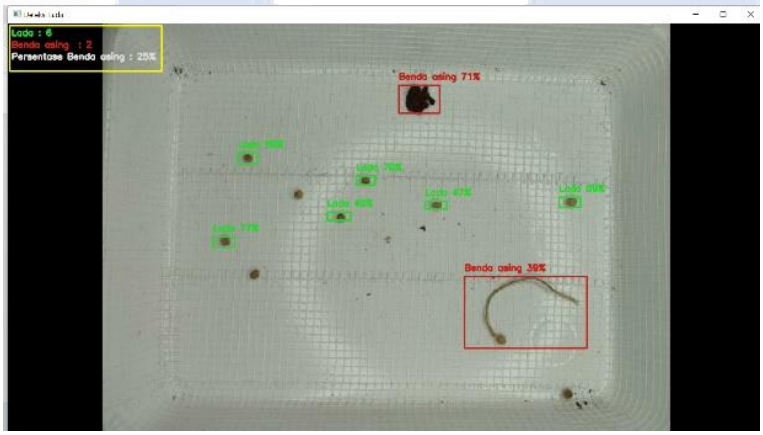
Pada pendeteksian ini, terdapat dua lada yang tidak berhasil dideteksi, hal ini dikarenakan lada berdempetan, kurangnya kualitas dari kamera atau atau kurangnya *dataset* yang di*training*.

Aktual Hasil

Gambar 1 (siang)

Lada : 10

Benda Asing : 2

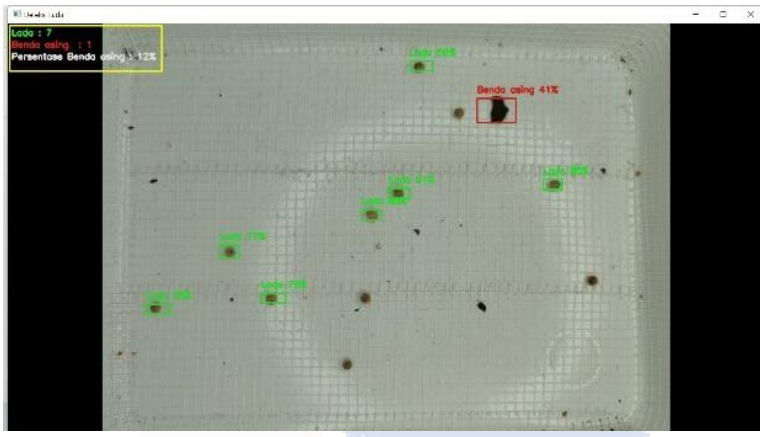
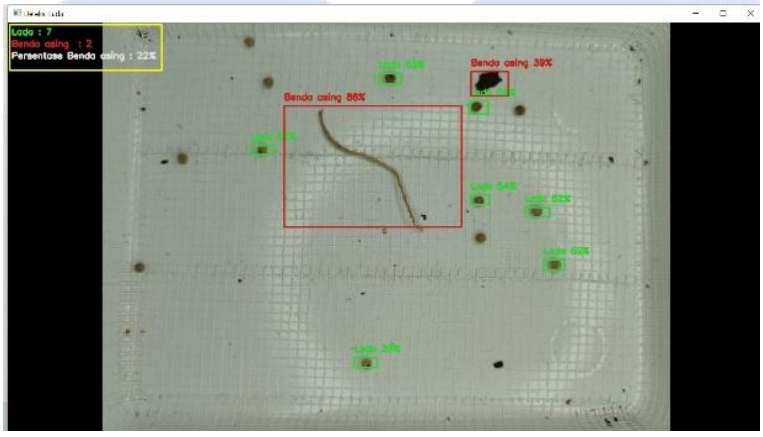


Lada : 6

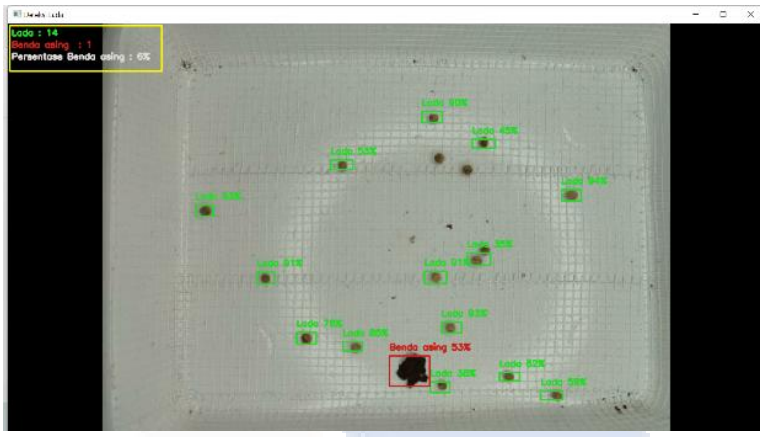
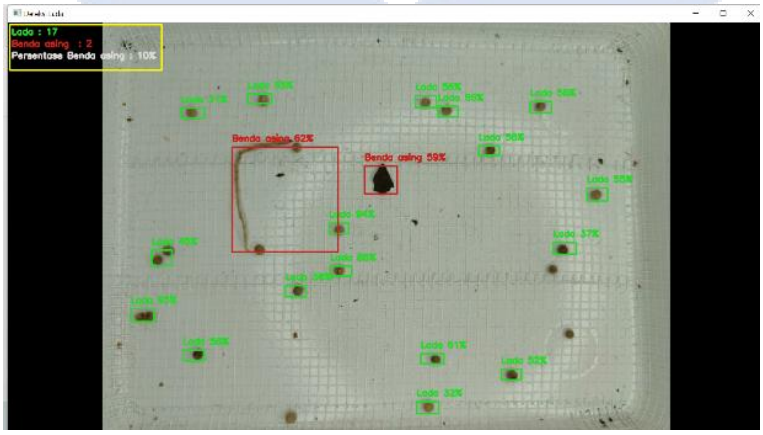
Benda Asing : 2

Pada pendeteksian ini, terdapat empat lada dan satu benda asing yang tidak berhasil dideteksi, hal ini dikarenakan lada berdempetan dengan tangkai, kurangnya kualitas dari

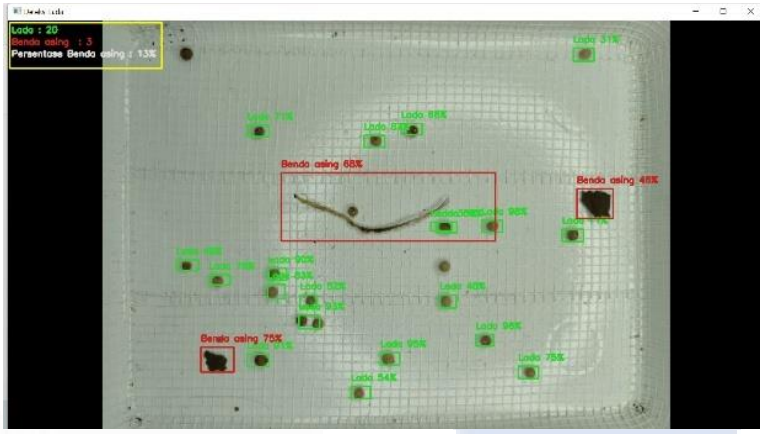
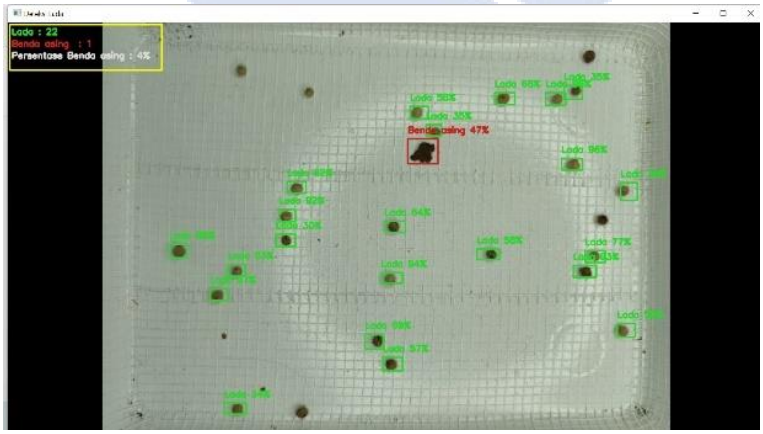
kamera atau kurangnya *dataset* yang di*training*.

Aktual	Hasil
Gambar 37 (siang) Lada : 11 Benda Asing : 1	 Lada : 7 Benda Asing : 1 Pada pendeteksian ini, terdapat empat lada yang tidak berhasil dideteksi, hal ini dikarenakan kurangnya kualitas dari kamera atau atau kurangnya *dataset* yang di*training*.
Aktual	Hasil
Gambar 48 (siang) Lada : 13 Benda Asing : 2	 Lada : 7 Benda Asing : 2 Pada pendeteksian ini, terdapat enam lada yang tidak berhasil dideteksi, hal ini dikarenakan kurangnya kualitas

dari kamera atau atau kurangnya *dataset* yang di*training*.

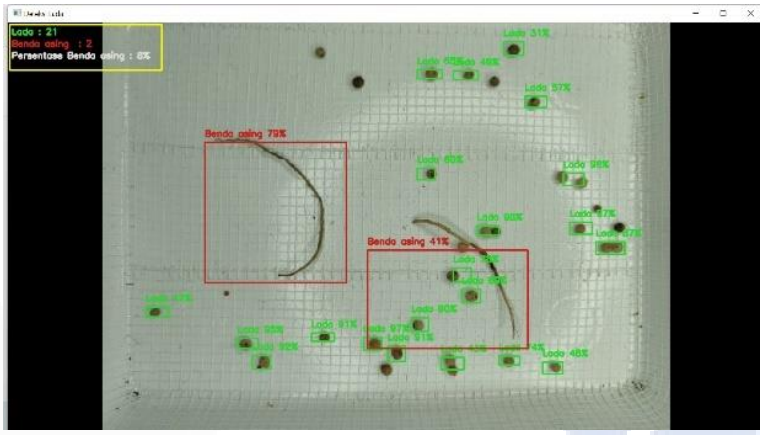
Aktual	Hasil
Gambar 4 (siang) Lada : 17 Benda Asing : 1	 Lada : 14 Benda Asing : 1 Pada pendeteksian ini, terdapat tiga lada yang tidak berhasil dideteksi, hal ini dikarenakan dikarenakan lada berdempetan, kurangnya kualitas dari kamera atau atau kurangnya *dataset* yang di*training*.
Aktual	Hasil
Gambar 37 (siang) Lada : 23 Benda Asing : 2	 Lada : 17 Benda Asing : 2 Pada pendeteksian ini, terdapat enam lada yang tidak

berhasil dideteksi, hal ini dikarenakan dikarenakan lada berdempetan, kurangnya kualitas dari kamera atau atau kurangnya *dataset* yang di*training*.

Aktual	Hasil
Gambar 18 (malam) Lada : 24 Benda Asing : 3	 Lada : 20 Benda Asing : 3 Pada pendeteksian ini, terdapat empat lada yang tidak berhasil dideteksi, hal ini dikarenakan dikarenakan lada berdempetan, kurangnya kualitas dari kamera atau atau kurangnya *dataset* yang di*training*.
Aktual	Hasil
Gambar 30 (malam) Lada : 27 Benda Asing : 1	 Lada : 22

Benda Asing : 1

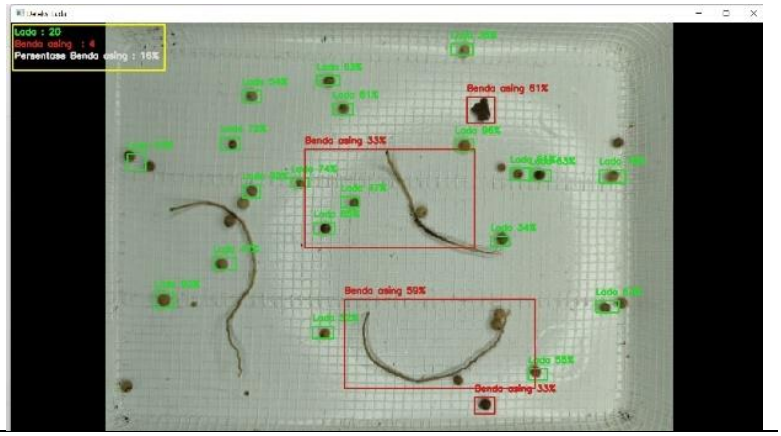
Pada pendeteksian ini, terdapat lima lada yang tidak berhasil dideteksi, hal ini dikarenakan dikarenakan kurangnya kualitas dari kamera atau atau kurangnya *dataset* yang di*training*.

Aktual	Hasil
Gambar 34 (malam) Lada : 31 Benda Asing : 2	

Lada : 21

Benda Asing : 2

Pada pendeteksian ini, terdapat sepuluh lada yang tidak berhasil dideteksi, hal ini dikarenakan dikarenakan lada berdempetan, kurangnya kualitas dari kamera atau atau kurangnya *dataset* yang di*training*.

Aktual	Hasil
Gambar 42 (malam) Lada : 32 Benda Asing : 4	

Lada : 20

Benda Asing : 3

Pada pendeteksian ini, terdapat sebelas lada dan satu lada terdeteksi benda asing, serta satu tangkai yang tidak terdeteksi, hal ini dikarenakan dikarenakan lada berdempetan, kurangnya kualitas dari kamera atau atau kurangnya *dataset* yang di*training*.

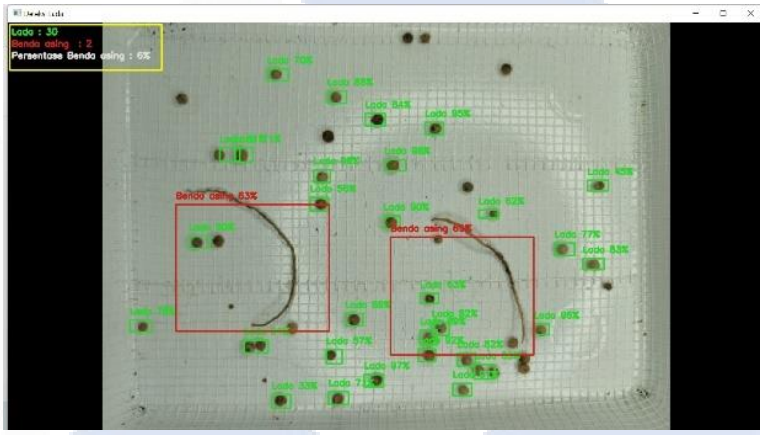
Aktual

Gambar 35 (malam)

Lada : 44

Benda Asing : 2

Hasil



Lada : 30

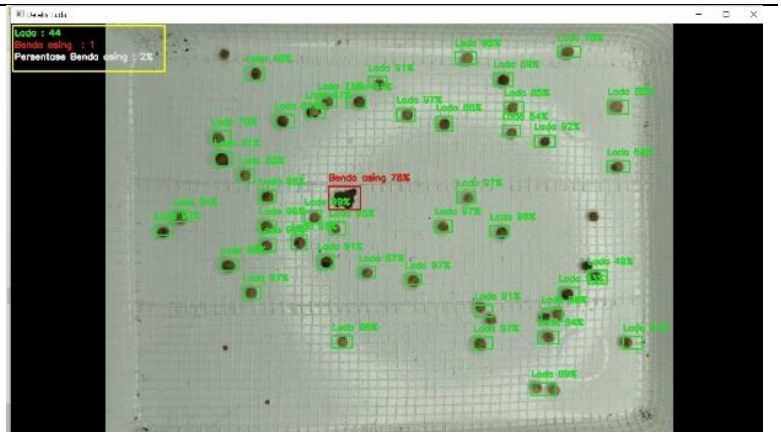
Benda Asing : 2

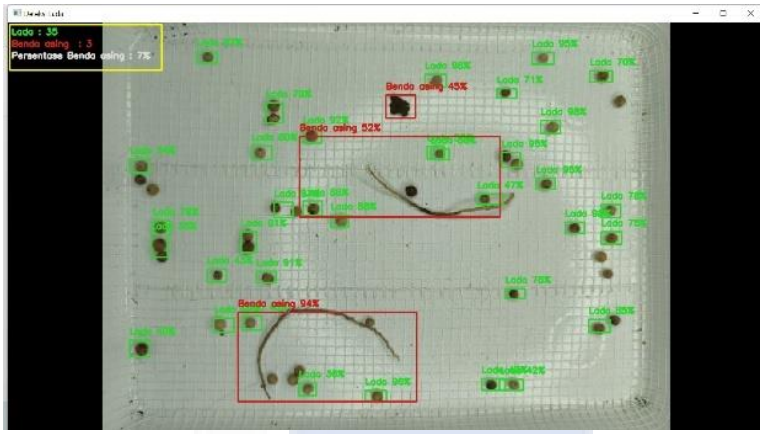
Pada pendeteksian ini, terdapat empatbelas lada yang tidak berhasil dideteksi, hal ini dikarenakan dikarenakan lada berdempetan, kurangnya kualitas dari kamera atau atau kurangnya *dataset* yang di*training*.

Aktual

Gambar 23 (malam)

Hasil

Lada : 49 Benda Asing : 1	 Lada : 44 Benda Asing : 1 Pada pendeteksian ini, terdapat lima lada yang tidak berhasil dideteksi, hal ini dikarenakan dikarenakan lada berdempetan, kurangnya kualitas dari kamera atau atau kurangnya *dataset* yang di*training*.
Aktual Gambar 24 (malam) Lada : 50 Benda Asing : 1	Hasil  Lada : 33 Benda Asing : 1 Pada pendeteksian ini, terdapat tujuhbelas lada yang tidak berhasil dideteksi, hal ini dikarenakan dikarenakan lada berdempetan, kurangnya kualitas dari kamera atau atau kurangnya *dataset* yang di*training*.

Aktual	Hasil
Gambar 27 (malam) Lada : 51 Benda Asing : 3	 Lada : 35 Benda Asing : 3 Pada pendeteksian ini, terdapat enambelas lada yang tidak berhasil dideteksi, hal ini dikarenakan dikarenakan lada berdempetan, kurangnya kualitas dari kamera atau atau kurangnya *dataset* yang di*training*.
Total Lada : 402 Benda asing : 29	Lada : 292 Benda asing : 27

Pada hasil pengujian diatas didapatkan hasil *output* pada layar *monitor* dimana objek yang terdeteksi jika berwarna hijau maka terdeteksi lada dan jika objek yang terdeteksi berwarna merah maka mendapatkan hasil deteksi berupa benda asing yaitu daun atau tangkai lada. tulisan lada berwarna hijau pada pojok kiri atas akan menampilkan *output* banyaknya lada yang terdeteksi dan tulisan benda asing berwarna hijau menampilkan *output* banyaknya benda asing yang berhasil terdeteksi, serta tulisan persentase benda asing berwarna putih akan menampilkan persentase mutu lada yang didapatkan. Berdasarkan pada table pengujian diatas, maka akan dibuat *Confusion Matrix* hasil deteksi objek lada dan benda asing seperti pada Tabel 4.3.

Tabel 4. 3 *Confusion Matrix*

Confusion Matrix		**Aktual**	
		Lada	**Benda asing**
Prediksi	Lada	292	-
	Benda asing	1	27
	Tidak Terdeteksi	109	1

Berdasarkan Tabel diatas dapat dicari nilai *Confusion Matrix,* yaitu nilai *True Positive, False Positive, False Negative, Precision, Recall,* dan *F1 Score.* Berikut persamaan untuk mencari nilai-nilai tersebut.

True Positive = 292 + 27 = 319

False Positive = 1

False Negative = 109 +1 = 110

$$Precision = \frac{TP}{TP+FP} = \frac{319}{319+1} = 0{,}99$$

$$Recall = \frac{TP}{TP+FN} = \frac{319}{319+110} = 0{,}74$$

$$F1\ Score = 2\ x\ \frac{(Precision\ x\ Recall)}{(Precision+Recall)} = 2\ x\ \frac{(0{,}99\ x\ 0{,}74)}{(0{,}99+0{,}74)} = 0{,}84$$

Didapatkan nilai *True Positive* sebanyak 319 objek lada dan benda asing yang berhasil dideteksi dengan benar, *False Positive* sebanyak 1 yang berarti sebanyak 1 objek yang tidak dapat dikenali, *False Negative* sebanyak 110 yang berarti terdapat 110 objek yang tidak berhasil dideteksi, *Precision* sebesar 0,99 yang artinya memiliki nilai hampir sempurna (sebesar 1), *Recall* sebesar 0,74 yang artinya tingkat deteksi objek lada dan benda asing secara keseluruhan secara benar sebesar 0,74 dan *F1 Score* yang bernilai 0,84. Oleh karena itu, nilai pengujian performa jaringan secara *realtime* mempunyai nilai yang tinggi.

Setelah melakukan pengujian performa jaringan yang dimana didapatkan nilai *Precision, Recall,* dan *F1 Score,* selanjutnya dilakukan perhitungan untuk

menentukan mutu lada berdasarkan persentase benda asing. Untuk melakukan perhitungan tersebut, biji lada dan benda asing pada satu kali pendeteksian akan dihitung, misal contoh pada gambar 24 diatas. Dari 50 lada dan 1 benda asing, terdapat 33 biji lada yang berhasil terdeteksi dan 1 benda asing yang berhasil terdeteksi. Untuk mendapatkan mutu lada berdasarkan persentase benda asing , total benda asing dibagi dengan total biji lada yang terdeteksi ditambahkan dengan total benda asing yang terdeteksi, kemudian dikalikan 100 (total benda asing/(total biji lada + total benda asing) * 100. Seperti pada contoh gambar 24, biji lada yang terdeteksi 33 dan 1 benda asing, maka 1/(33 +1) * 100 = 2%. Mutu lada menurut standart SNI ada 2 yaitu mutu lada 1 = 1% dan mutu lada 2 = 2%. Untuk mendapatkan *output* persentase, hasil pendeteksian secara *realtime* harus bagus. Jika pendeteksian tidak bisa dilakukan maka persentase yang didapatkan tidak akan ada keluaran.

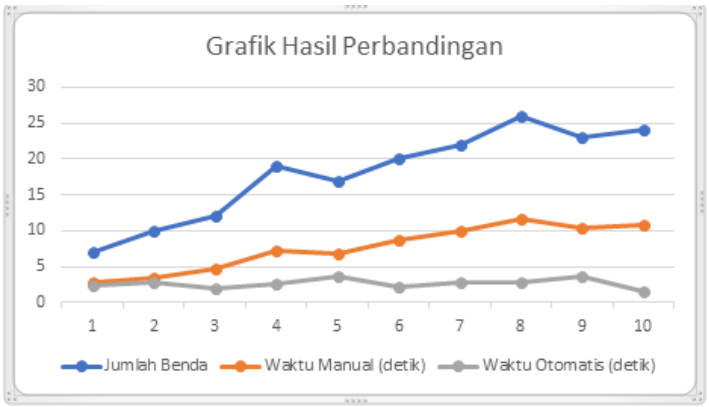
Selain melakukan pengujian performa jaringan menggunakan 16 gambar dari hasil uji coba secara *realtime,* kami juga menghitung lamanya waktu pendeteksian sistem. Untuk perbandingan lamanya waktu sistem mendeteksi objek, kami melakukan pengujian dengan membandingkan antara pengujian menggunakan sistem deteksi dan pengujian dengan cara menghitung sendiri lada yang akan di uji coba (dilakukan oleh manusia). Untuk melakukan perbandingan, kami mengambil 10 sampel percobaan. Berikut ini adalah Data sampel perbandingan waktu dari pendeteksian secara otomatis dan perhitungan manual :

Tabel 4. 4 Data Sampel Dan Waktu Deteksi

No.	Jumlah Lada (per biji)	Waktu Otomatis (detik)	Waktu Manual (detik)
1	7	2,5	2,77
2	10	2,91	3,42
3	12	1,89	4,77
4	19	2,66	7,31
5	17	3,73	6,82
6	20	2,15	8,8
7	22	2,82	9,98
8	26	2,79	11,76

9	23	3,74	10,31
10	24	1,52	10,73

pada tabel diatas, kami membandingkan apakah lebih cepat menggunakan sistem deteksi atau dihitung satu per satu secara manual. Dan hasil yang didapatkan dapat dilihat pada gambar grafik berikut ini :



Gambar 4. 36 Grafik Hasil Perbandingan

Pada gambar grafik diatas didapatkan bahwa waktu pendeteksian menggunakan sistem lebih cepat dibandingkan dengan dihitung secara manual. Saat menggunakan sistem deteksi objek lada dan benda asing yang muncul pada frame akan langsung terdeteksi tetapi membutuhkan waktu untuk penyesuaian letak objek deteksi. Sedangkan jika kita menghitung lada secara manual, waktu yang dibutuhkan lebih lama karena objek lada kecil sehingga kita butuh waktu menghitung Kembali apakah benar biji lada dan benda asing yang ada sesuai perhitungan yang telah kita lakukan. Pada saat perbandingan yang dilakukan oleh sistem deteksi dan yang dilakukan secara manual (oleh manusia), didapatkan hasil bahwa jumlah lada sangat mempengaruhi perbedaan waktu perbandingan pendeteksian. Semakin banyak lada maka semakin susah pula untuk menghitung objek biji lada dan benda asing secara manual. Dari data yang didapatkan pada tabel 4.4, sistem dapat mendeteksi 7 lada dengan waktu 0,27 detik lebih baik daripada cara manual dan 26 lada dengan waktu 8,97 detik lebih baik daripada cara manual. Sedangkan sistem bisa mendeteksi secara maksimal apabila tidak ada kendala jaringan pada waktu pendeteksian.

Pada saat pengujian lamanya waktu deteksi dengan sistem (waktu otomatis) menggunakan *DroidCam*, terdapat hasil data waktu otomatis mendeteksi lebih cepat daripada waktu pendeteksian otomatis yang lain seperti pada sampel ke 3 dan sampel ke 10 waktu pendeteksian lebih cepat padahal jumlah lada yang terdeteksi banyak, hal ini dikarenakan saat pendeteksian, WIFI yang dihubungkan dengan *DroidCam* mengalami kendala sinyal sehingga pendeteksian yang dilakukan terhambat. Jika jaringan internet lancar maka akan semakin baik pula hasil pendeteksian.

BAB V
PENUTUP

5.1 Kesimpulan

Dari pengujian sistem yang sudah dilakukan, ini berhasil mengimplementasikan sistem deteksi lada dan benda asing. Kemudian untuk lebih detail ada beberapa kesimpulan sebagai berikut :

a. Pada penelitian ini, sistem pendeteksi biji lada dan benda asing berbasis pengolahan citra menggunakan metode *You Only Look One* (YOLO) berhasil mendeteksi objek lada dan benda asing.
b. Pada pengujian ini, sistem ini sudah bisa mendeteksi lebih dari satu objek dan menentukan persentase benda asing.
c. Pengujian deteksi dengan 16 gambar dari hasil uji coba secara *realtime* dengan luas penampang 17,6 x 22,8 cm sebagai tempat peletakan sampel lada dan jarak pendeteksian 20 cm, serta dengan bantuan pencahayaan, didapatkan hasil pengujian memiliki nilai kinerja jaringan yang cukup tinggi yaitu nilai *Precision* sebesar 0,99 atau nilai *Precision* yang hampir sempurna, nilai *Recall* diatas 70%, dan *F1 Score* bernilai diatas 80%. Untuk nilai pengujian performa jaringan ini bisa dipengaruhi oleh jumlah dataset dan resolusi *dataset*.
d. Pada pengujian perbandingan kecepatan waktu pendeteksian oleh sistem dan perhitungan secara manual (oleh manusia), waktu pendeteksian menggunakan sistem lebih cepat dibandingkan dilakukan secara manual. Semakin banyak lada maka semakin lama pula waktu pendeteksian.
e. Jumlah gambar pada proses *training* sangat mempengaruhi hasil pengujian dalam mendeteksi objek yang diinginkan. Semakin banyak jumlah gambar pada data training maka semakin tinggi tingkat akurasi, sebaliknya jika gambar pada data training sedikit maka tingkat akurasi juga akan menurun atau kurang tepat dalam mendeteksi objek yang diinginkan.

f. Ukuran gambar sangat mempengaruhi proses data *training*, semakin besar ukuran *pixel* gambar yang digunakan maka semakin lama dalam melakukan proses data *training*.

g. *Input* gambar yang di training diatas sebanyak 212 gambar, 104 gambar lada, 31 gambar tangkai lada, 57 gambar daun, dan 20 gambar campuran. Jumlah ini terbilang sedikit untuk proses pelatihan menggunakan metode YOLOv3-*Tiny*, untuk mendapat hasil deteksi dengan tingkat akurasi tinggi maka diperlukan *input / dataset* yang banyak.

5.2 Saran

Dari hasil proyek akhir ini penulis memiliki beberapa saran untuk pengembangan yang lebih lanjut sebagai berikut :

a. Memperbanyak *input* citra atau dataset, resolusi gambar dan waktu pengambilan gambar (siang atau malam hari). Hal ini untuk menambah fitur pengenalan *object detection* dalam berbagai situasi.

b. Menggunakan algoritma deteksi objek yang lainnya, seperti YOLO v4, YOLO v5, *Faster R-CNN*, *Single Shot Detector* (SSD) dan algoritma lainnya. Diharapkan dengan menggunakan algoritma deteksi objek lainnya didapatkan perbedaan kinerja pada pendeteksian.

c. Menggunakan kamera *android* yang spesifikasinya tinggi sehingga kualitas *input* citra dan hasil pendeteksian yang didapatkan lebih baik

DAFTAR PUSTAKA

[1] N. Nurdjannah, "Perbaikan Mutu Lada dalam Rangka Meningkatkan Daya Saing di Pasar Dunia," Balai Besar Penelitian dan Pengembangan Pasca Panen Pertanian, *Perspektif*, vol. 5, no. 1, pp. 13-25, 2006.

[2] N. O' Mahony *dkk.*, "Deep Learning vs. Traditional Computer Vision," IMaR Technology Gateway, Institute of Technology Tralee, Tralee, Ireland, 2019. doi**:** 10.1007/978-3-030-17795-9.

[3] D. Gasong, "Belajar dan Pembelajaran*,"* 20 Juni 2018. [Online], Available: https://books.google.co.id/books?hl=en&lr=&id=3rljDwAAQBAJ&oi=fnd&pg=PA162&dq=mampu+untuk+belajar+dan+berkembang+dengan+sendirinya+machine+learning&ots=tlA9 Bzkeu1&sig=tf-rWiJjmXlLW_iA_-PRGVWjGP8&redir_esc=y#v=onepage&q&f=false.2021. [accessed 14 November 2022].

[4] Y. Suzana dan I. Jayanto, "Teori Belajar & Pembelajaran," April 2021. [Online],Available: https://books.google.co.id/books?hl=en&lr=&id=cyYvEAAAQBAJ&oi=fnd&pg=PP1&dq=mampu+untuk+belajar+dan+berkembang+dengan+sendirinya+machine+learning&ots=zh9CHaMCX5&sig=aiIX2c_uDjJFjhUSkXfcSWrCsp4&redir_esc=y#v=onepage&q&f=false. 2021. [accessed 14 November 2022].

[5] W. Fang, L. Wang, dan P. Ren, "Tinier-YOLO: A Real-Time Object Detection Method for Constrained Environments," *IEEE Access*, vol. 8, pp. 1935–1944, 2020, doi: 10.1109/ACCESS.2019.2961959.

[6] T. Li, Y. Ma, dan T. Endoh, "A Systematic Study of Tiny YOLO3 Inference: Toward Compact Brainware Processor with Less Memory and Logic Gate," *IEEE Access*, vol. 8, pp. 142931–142955, 2020, doi: 10.1109/ACCESS.2020.3013934.

[7] D. A. Prabowo, D. Abdullah, dan A. Manik, "DETEKSI DAN PERHITUNGAN OBJEK BERDASARKAN WARNA MENGGUNAKAN COLOR OBJECT TRACKING," *Jurnal Pseudocode*, vol. 5, no. 2, pp.85-91, 2018.

[8] J. Pedoeem dan R. Huang, "YOLO-LITE: A Real-Time Object Detection Algorithm Optimized for Non-GPU Computers," November 2018, doi: 10.48550/arxiv.1811.05588.

[9] T. A. A. H. Kusuma, K. Usman, dan S. Saidah, "PEOPLE COUNTING FOR PUBLIC TRANSPORTATIONS USING YOU ONLY LOOK ONCE METHOD," *Jurnal Teknik Informatika (Jutif)*, vol. 2, no. 1, pp. 57–66, Feb 2021, doi: 10.20884/1.jutif.2021.2.2.77.

[10] B. Liu, W. Zhao, dan Q. Sun, "Study Of Object Detection Based On Faster R-CNN." College of Automation & Electronic Engineering Qingdao University of Science and Technology Qingdao, China, 2017.

[11] W. Liu *dkk.*, "SSD: Single Shot MultiBox Detector," 29 Desember 2016, doi: 10.1007/978-3-319-46448-0_2.

[12] J. Redmon, S. Divvala, R. Girshick, dan A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," Juni 2015, [Online]. Available: http://arxiv.org/abs/1506.02640. [accessed 25 Desember 2022].

[13] T.-Y. Lin *dkk.*, "Microsoft COCO: Common Objects in Context," Mei 2014, [Online]. Available: http://arxiv.org/abs/1405.0312. [accessed 25 Desember 2022].

[14] S. Ilahiyah dan A. Nilogiri, "Implementasi Deep Learning Pada Identifikasi Jenis Tumbuhan Berdasarkan Citra Daun Menggunakan Convolutional Neural Network," *JUSTINDO* (Jurnal Sistem & Teknologi Informasi Indonesia), vol. 3, no. 2, pp. 49-56, 2018.

[15] O. E. Karlina dan 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, doi: 10.35760/ik.2019.v24i3.2362.

[16] M. Harahap *dkk.*, "Sistem Cerdas Pemantauan Arus Lalu Lintas Dengan YOLO (You Only Look Once v3)," *Seminar Nasional APTIKOM (SEMNASTIK)*, pp. 367-376, 2019.

[17] "A. F. Fandisyah, N. Iriawan end W. S. Winahju, "Deteksi Kapal di Laut Indonesia Menggunakan YOLOv3," *Jurnal Sains dan Seni ITS* vol. 10 no. 1, pp. D25-D32, 2021.

[18] Khairunnas, E. M. Yuniarno dan A. Zaini, "Pembuatan Modul Deteksi Objek Manusia Menggunakan Metode YOLO untuk Mobile Robot," *Jurnal Teknik ITS,* vol. 10, no. 1, pp. A50-A55, 2021.

[19] C. Geraldy dan C. Lubis, "PENDETEKSIAN DAN PENGENALAN JENIS MOBIL MENGGUNAKAN ALGORITMA YOU ONLY LOOK ONCE DAN CONVOLUTIONAL NEURAL NETWORK," *Jurnal Ilmu Komputer dan Sistem Informasi*, vol. 08, no. 2, pp. 197-199, 2020.

[20] J. Redmon dan A. Farhadi, "YOLOv3: An Incremental Improvement," April 2018, [Online]. Available : http://arxiv.org/abs/1804.02767. [accessed 14 November 2022].

[21] M. S. Chauhan, A. Singh, M. Khemka, A. Prateek, dan R. Sen, "Embedded CNN based vehicle classification and counting in non-laned road traffic," dalam *ACM International Conference Proceeding Series*, 2019. doi: 10.1145/3287098.3287118.

[22] Y. Liu *dkk.*, "Research on automatic location and recognition of insulators in substation based on YOLOv3," *High Voltage*, vol. 5, no. 1, pp. 62–68, 2020, doi: 10.1049/hve.2019.0091.

[23] H. Zhang *dkk.*, "Real-Time Detection Method for Small Traffic Signs Based on Yolov3," *IEEE Access*, vol. 8, pp. 64145–64156, 2020, doi: 10.1109/ACCESS.2020.2984554.

[24] D. Giancini, E. Yulia Puspaningrum, Y. Vita Via, "Identifikasi Penggunaan Masker Menggunakan Algoritma CNN YOLOv3-Tiny," *Seminar Nasional Informatika Bela Negara (SANTIKA)*, vol. 1, pp. 153-159, 2020.

[25] A. S. Riyadi, I. P. Wardhani, M. S. Wulandari, dan S. Widayati, "Perbandingan Metode ResNet, YoloV3, dan TinyYoloV3 pada Deteksi Citra dengan Pemrograman Python," *PETIR*, vol. 15, no. 1, pp. 135–144, 2022, doi: 10.33322/petir.v15i1.1302.

[26] A. Widiyono, M. Nofiyanti, H. Minnah, L. Zahro, L. Inayah, dan M. U. Absor, "Training on Making Learning Videos through Bandicam and Droidcam Applications for Teachers Elementary Schools," *MITRA: Jurnal Pemberdayaan Masyarakat*, vol. 6, no. 2, pp. 169–179, 2022, doi: 10.25170/mitra.v6i2.3252.

LAMPIRAN

LAMPIRAN 1

DAFTAR RIWAYAT HIDUP

1. Data Pribadi

Nama Lengkap	: Muhammad Erfani Ramadhani
Tempat & Tanggal lahir	: Muntok, 21 November 2001
Alamat	: Kampung Menjelang, RT 01 RW 01, Muntok
Jenis Kelamin	: Laki-laki
Agama	: Islam
Telp	: -
Hp	: 085783874738
E-mail	: fanie4843@gmail.com
Hobi	: Memancing

2. Riwayat Pendidikan

   MIS Al-ISHLAH Lulus Tahun 2013

   SMP N 1 Muntok Lulus Tahun 2016

   SMK Bina Karya 1 Muntok Lulus Tahun 2019

3. Riwayat Pendidikan Non Formal

   -

Sungailiat, 15 Februari 2023

Muhammad Erfani Ramadhani

DAFTAR RIWAYAT HIDUP

1. Data Pribadi

Nama Lengkap	: Siti Barokah
Tempat & Tanggal lahir	: Dalil, 21 Januari 2001
Alamat	: Jl. Olahraga, RT 07 RW 02, Desa Dalil
Jenis Kelamin	: Perempuan
Agama	: Islam
Telp	: -
Hp	: 085896526294
E-mail	: barokahsiti001@gmail.com
Hobi	: Membaca novel dan cerpen

2. Riwayat Pendidikan

   SDN 5 Bakam Lulus Tahun 2013

   MTS Nurul Hidayah Dalil Lulus Tahun 2016

   SMA N 1 Bakam Lulus Tahun 2019

3. Riwayat Pendidikan Non Formal

   -

Sungailiat, 15 Februari 2023

Siti Barokah

LAMPIRAN 2

Program Keseluruhan

```
!nvidia-smi
from google.colab import drive
drive.mount('/content/gdrive')
!ln -s /content/gdrive/My\ Drive/ /mydrive
!ls /mydrive
!git clone https://github.com/AlexeyAB/darknet
%cd darknet
!sed -i 's/OPENCV=0/OPENCV=1/' Makefile
!sed -i 's/GPU=0/GPU=1/' Makefile
!sed -i 's/CUDNN=0/CUDNN=1/' Makefile
!make
!cp cfg/yolov3-tiny.cfg cfg/yolov3_training.cfg
!sed -i 's/batch=1/batch=64/' cfg/yolov3_training.cfg
!sed -i 's/subdivisions=1/subdivisions=16/' cfg/yolov3_training.cfg
!sed -i 's/max_batches = 500200/max_batches = 4000/' cfg/yolov3_training.cfg
!sed -i '610 s@classes=80@classes=2@' cfg/yolov3_training.cfg
!sed -i '696 s@classes=80@classes=2@' cfg/yolov3_training.cfg
!sed -i '783 s@classes=80@classes=2@' cfg/yolov3_training.cfg
!sed -i '603 s@filters=255@filters=21@' cfg/yolov3_training.cfg
!sed -i '689 s@filters=255@filters=21@' cfg/yolov3 training.cfg
!sed -i '776 s@filters=255@filters=21@' cfg/yolov3_training.cfg
!echo -e 'Lada\nBenda_asing' > data/obj.names
!echo -
e 'class= 2\ntrain= data/train.txt\nvalid= data/test.txt\nnames= data/obj.names\nb
ackup= /mydrive/Yolov3tiny' > data/obj.data
!cp data/obj.names /mydrive/yolov3-tiny_v3/classes.txt
!mkdir data/obj
!unzip /mydrive/yolov3-tiny v3/coba5.zip -d data/obj
import glob
images list = glob.glob("data/obj/coba5/*.jpg")
with open("data/train.txt", "w") as f:
    f.write("\n".join(images list))
!wget https://pjreddie.com/media/files/darknet53.conv.74
!./darknet detector train data/obj.data cfg/yolov3 training.cfg darknet53.conv.74
-dont_show
#!./darknet detector train data/obj.data cfg/yolov3 training.cfg /mydrive/Yolov3ti
ny/yolov3 training last.weights -dont show
       import cv2
import numpy as np
net = cv2.dnn.readNet("yolov3_training_final.weights", "yolov3-tiny.cfg")
```

```
    classes = []
    with open("coba5/classes.txt", "r") as f:
classes = f.read().splitlines()
    net.setPreferableBackend(cv2.dnn.DNN BACKEND CUDA)
    net.setPreferableTarget(cv2.dnn.DNN_TARGET_CUDA_FP16)
    model = cv2.dnn DetectionModel(net)
    model.setInputParams(size=(416, 416), scale=1 / 255, swapRB=True)
    CONFIDENCE THRESHOLD = 0.3
    NMS_THRESHOLD = 0.5
    cap = cv2.VideoCapture(0)
    if cap.read()[0] is False:
        cap = cv2.VideoCapture(1)
        if cap.read()[0] is False:
            cap = cv2.VideoCapture(2)
    while True:
        (grabbed, img) = cap.read()
        classes, scores, boxes = model.detect(img, CONFIDENCE_THRESHOLD,
    NMS THRESHOLD)
        count Lada = 0
        count Benda asing = 0
        persentase = 0
        for (classid, score, box) in zip(classes, scores, boxes):
            x, y, w, h = box[0], box[1], box[2], box[3]
            # Make Bounding Box
            if classid == 0:
                count Lada += 1
                cv2.rectangle(img, (x, y), (x + w, y + h), (42, 217, 39), 2)
                cv2.putText(img, "Lada " + str(int(score * 100)) + "%", (x, y -
    10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (42, 217, 39), 2)
            elif classid == 1:
                count_Benda_asing += 1
                cv2.rectangle(img, (x, y), (x + w, y + h), (28, 34, 200), 2)
                cv2.putText(img, "Benda asing " + str(int(score * 100)) + "%",
    (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (28, 34, 200), 2)
        if count_Lada != 0 or count_Benda_asing != 0:
            persentase = 100 * (count_Benda_asing / (count_Lada +
    count_Benda_asing))
        cv2.rectangle(img, (3, 3), (260, 80), (0,255,255), 2)
        cv2.putText(img, "Lada : " + str(count_Lada), (6, 20),
    cv2.FONT_HERSHEY_SIMPLEX, 0.5, (42, 217, 39), 2)
        cv2.putText(img, "Benda asing  : " + str(count_Benda_asing), (6, 40),
    cv2.FONT_HERSHEY_SIMPLEX, 0.5, (28, 34, 200), 2)
        cv2.putText(img, "Persentase Benda asing : " + str(int(persentase)) +
    "%", (6, 60), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (217, 218, 219),
                    2)
```

```
            cv2.imshow('Deteksi Lada', img)
            if cv2.waitKey(1) & 0xFF == ord('q'):
                break
        cv2.waitKey(1)
cv2.destroyAllWindows()
```