SKRIPSI

IMPLEMENTASI ALGORITMA YOLOv5 UNTUK MENDETEKSI DAN MENGHITUNG JUMLAH KENDARAAN MENGGUNAKAN METODE DEEPSORT

Oleh:
Zaka Darmawan
065119091

PROGRAM STUDI ILMU KOMPUTER
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS PAKUAN
2024

HALAMAN PENGESAHAN

Judul : Implementasi Algoritma YOLOv5 Untuk Mendeteksi dan Menghitung Jumlah Kendaraan Menggunakan Metode *DeepSORT*
Nama : Zaka Darmawan
NPM : 065119091

Mengesahkan,

Pembimbing Pendamping
Program Studi Ilmu Komputer
FMIPA - UNPAK

Mulyati, S.Kom., M.Kom.

Pembimbing Utama
Program Studi Ilmu Komputer
FMIPA - UNPAK

Dr. Tjut Awaliyah Zuraiyah, M.Kom.

Mengetahui,

Ketua Program Studi Ilmu Komputer
FMIPA - UNPAK

Arie Qurania, M.Komp

Dekan
FMIPA - UNPAK

Asep Denih, S.Kom., M.Sc., Ph.D.

PERNYATAAN KEASLIAN KARYA TULIS SKRIPSI

Dengan ini, saya yang bertanda tangan dibawah ini:

Nama	:	Zaka Darmawan
NPM	:	065119091
Program Studi	:	Ilmu Komputer
Fakultas	:	Matematika dan Ilmu Pengetahuan Alam Universitas Pakuan Bogor

Menyatakan bahwa skripsi yang berjudul "Implementasi Algoritma YOLOv5 Untuk Mendeteksi dan Menghitung Jumlah Kendaraan Menggunakan Metode *DeepSORT*". Sejauh yang saya ketahui, karya tulis ini bukan merupakan karya tulis yang pernah dipublikasikan atau sudah pernah dipakai untuk mendapatkan gelar sarjana di Universitas lain, kecuali pada bagian-bagian dimana sumber informasinya dicantumkan dengan cara referensi yang semestinya.

Demikian pernyataan ini saya buat dengan sebenar-benarnya. Apabila kelak ditemukan hari terdapat gugatan, penulis bersedia dikenakan sanksi sesuai dengan peraturan yang berlaku.

Bogor, Juni 2024

Zaka Darmawan
065119091

PERNYATAAN PELIMPAHAN SKRIPSI DAN SUMBER INFORMASI SERTA PELIMPAHAN HAK CIPTA

Saya yang bertandatangan di bawah ini :

Nama	: Zaka Darmawan
NPM	: 065119091
Judul	: Implementasi Algoritma Yolov5 Untuk Mendeteksi Dan Menghitung Jumlah Kendaraan Menggunakan Metode Deepsort

Dengan ini saya menyatakan bahwa Paten dan Hak Cipta dari produk Skripsi dan Tugas Akhir di atas adalah benar karya saya dengan arahan dari komisi pembimbing dan belum diajukan dalam bentuk apapun kepada perguruan tinggi manapun.

Sumber informasi yang berasal atau dikutip dari karya yang diterbitkan maupun tidak diterbitkan dari penulis lain telah disebutkan dalam teks dan dicantumkan dalam Daftar Pustaka di bagian akhir skripsi ini.

Dengan ini saya melimpahkan Paten, hak cipta dari karya tulis saya kepada Universitas Pakuan.

Bogor, 19 September 2024

Zaka Darmawan
065119091

RIWAYAT HIDUP

Zaka Darmawan, dilahirkan di Kota Bogor pada tanggal 01 Januari 2001 dari pasangan Bapak Slamet Hadi dan Ibu Julaeha sebagai anak ketiga dari tiga bersaudara. Penulis memulai Pendidikan sekolah dasar pada tahun 2007 di SDN Kebon Pedes 1 Bogor dan lulus pada tahun 2013. Di tahun yang sama penulis melanjutkan Pendidikan ke SMPN 12 Bogor dan lulus pada tahun 2016, lalu pada tahun yang sama penulis melanjukan Pendidikan ke SMA Kosgoro Bogor lulus pada tahun 2019. Pada tahun 2019 penulis melanjutkan Pendidikan ke Universitas Pakuan Bogor dengan Program Studi Ilmu Komputer Fakultas Matematika dan Ilmu Pengetahuan Alam. Selama menjadi mahasiswa di Universitas Pakuan Bogor, pada bulan Juli 2022 penulis melaksanakan Praktik Lapang di Pengadilan Agama Bogor. Pada bulan April 2023 penulis memulai penelitian tugas akhir dan selesai pada bulan Juni 2024 dengan judul "Implementasi Algoritma YOLOv5 Untuk Mendeteksi dan Menghitung Jumlah Kendaraan Menggunakan Metode *DeepSORT*" untuk menuntaskan jenjang perkuliahan dan mendapatkan gelar Sarjana Ilmu Komputer.

RINGKASAN

Zaka Darmawan, 2024. Implementasi Algoritma YOLOv5 Untuk Mendeteksi dan Menghitung Jumlah Kendaraan Menggunakan Metode *DeepSORT*. Dibawah bimbingan **Dr.Tjut Awaliyah Zuraiyah, M.Kom. dan Mulyati, M.Kom.**

Kota Bogor yang merupakan salah satu kota penyangga ibu kota menjadi kawasan menarik untuk para pendatang, kenaikan jumlah penduduk di Kota Bogor dari tahun ke tahun terus terjadi seiring dengan pembangunan Kota Bogor. Pertumbuhan ini berdampak pada peningkatan jumlah kendaraan yang masuk ke kota setiap harinya. Informasi yang akurat mengenai jumlah kendaraan yang masuk sangat penting untuk perencanaan dan pengelolaan lalu lintas, termasuk dalam menentukan kebijakan transportasi, memperkirakan tingkat kepadatan lalu lintas, serta merencanakan pengembangan infrastruktur jalan yang sesuai dengan kebutuhan. Diperlukan sebuah data yang akurat mengenai kendaraan yang masuk ke kota Bogor agar data membuat keputusan yang tepat dalam tata kelola berkendara. Proses pengamatan dan perhitungan kendaraan secara manual oleh manusia dalam jangka waktu lama dapat menimbulkan kesalahan perhitungan (human error) dikarenakan fokus manusia dapat berkurang apabila melakukan suatu hal dalam waktu yang lama

Maka dibutuhkan penerapan *AI (Artificial Intelligence)* khususnya *Computer vision* dalam melakukan penghitungan kendaraan secara otomatis. sistem pengawasan lalu lintas cerdas diperlukan untuk membantu manusia mencapai manajemen lalu lintas yang cerdas. Fungsi dari sistem ini adalah untuk mendeteksi kendaraan dalam sebuah video. Kemudian kendaraan akan diklasifikasikan menurut jenisnya seperti sepeda motor, mobil, bus, dan truk. Salah satu algoritma yang digunakan untuk mendeteksi kendaraan adalah algoritma YOLO.

Pengujian ini dimulai dari tahap pengumpulan data berupa video yang direkam langsung di JPO Botani menggunakan smarftphone berjumlah 14 video di waktu pagi dan sore hari dengan jarak 50 meter. Total *dataset* yang didapatkan 2276 gambar yang terbagi menjadi 3 bagian yaitu *dataset* train 70% dengan data sebanyak 1614, *dataset* val 20% dengan data sebanyak 466 dan *dataset* test 10 % dengan data sebanyak 221

Penelitian ini bertujuan untuk mengimplementasi Algoritma YOLOv5 untuk mendeteksi dan menghitung jumlah kendaraan menggunakan metode DeepSORT. Hasil dari uji UAT model test menggunakan input sample video sebanyak 14 pada waktu pagi dan sore hari di atas JPO Botani dengan jarak 50 meter yang diproses melalui website dengan model yang sudah diproses menggunakan *dataset* mandiri mendeteksi jumlah kendaraan sebanyak 4640 dengan hasil true positive (TP) sebanyak 4633, true negative (TN) sebanyak 6, false positive (FP) sebanyak 1, false negative (FN) sebanyak dengan 1 dan menghasilkan akurasi sebesar 99%, precision dengan nilai 99%

Kata Kunci: *Computer Vision*, *Deepsort,* Kendaraan, *Machine Learning, Yolov5*.

KATA PENGANTAR

Puji syukur kehadirat Allah SWT, karena rahmat dan hidayah- Nya penulisa dapat menyelesaikan skripsi ini yang berjudul "Implementasi Algoritma YOLOv5 Untuk Mendeteksi dan Menghitung Jumlah Kendaraan Menggunakan Metode *DeepSORT* ". Penulisan tugas akhir ini merupakan salah satu syarat memperoleh gelar Sarjana Komputer di Program Studi Ilmu Komputer FMIPA UNPAK Bogor.

Dalam penulisan tugas akhir ini, penulis dengan senang hati ingin mengucapkan terima kasih yang sebesar-besarnya kepada:

1. Dr Tjut Awaliyah Zuraiyah, M.Kom., selaku Pembimbing Utama yang telah memberikan dorongan moril dan motivasi kepada penulis.
2. Mulyati, S.Kom. M.Kom., selaku pembimbing Pendamping yang telah memberikan bimbingan, semangat dan motivasi.
3. Arie Qur'nia, M.Kom., Selaku Ketua Program Studi Ilmu Komputer FMIPA Universitas Pakuan Bogor.
4. Kedua Orangtua serta keluarga yang selalu memberikan semangat, dukungan dan do'a.
5. Teman-teman Program Studi Ilmu Komputer angkatan 2019, Fakultas Matematika Dan Ilmu Pengetahuan Alam Universitas Pakuan yang telah memberikan semangat dan dukungan dalam penyusunan program dan laporan ini.

Saran dan kritik yang membangun dalam penulisan tugas akhir ini akan diterima dengan senang hati. Mudah-mudahan Allah SWT akan membalas semua kebaikan kepada semua pihak yang membantu. Akhir kata, semoga laporan ini dapat bermanfaat bagi kita semua. Aamiin.

Bogor, Juni 2024

Zaka Darmawan
065119091

DAFTAR ISI

Halaman

DAFTAR GAMBAR

Halaman

DAFTAR TABEL

Halaman

DAFTAR LAMPIRAN

Halaman

BAB I
PENDAHULUAN

1.1 Latar Belakang

Kota Bogor yang merupakan salah satu kota penyangga ibu kota menjadi kawasan menarik untuk para pendatang, kenaikan jumlah penduduk di Kota Bogor dari tahun ke tahun terus terjadi seiring dengan pembangunan Kota Bogor (Farizkhar, 2022). Pertumbuhan ini berdampak pada peningkatan jumlah kendaraan yang masuk ke kota setiap harinya. Informasi yang akurat mengenai jumlah kendaraan yang masuk sangat penting untuk perencanaan dan pengelolaan lalu lintas, termasuk dalam menentukan kebijakan transportasi, memperkirakan tingkat kepadatan lalu lintas, serta merencanakan pengembangan infrastruktur jalan yang sesuai dengan kebutuhan. kendaraan yang masuk ke Kota Bogor sebagai kota wisata cukup tinggi sehingga menimbulkan kemacetan di kota Bogor. Diperlukan sebuah data yang akurat mengenai kendaraan yang masuk ke kota Bogor agar data membuat keputusan yang tepat dalam tata kelola berkendara. Data kendaraan yang dibutuhkan meliputi jumlah sepeda motor, mobil, bus, truk yang masuk.

Seiring dengan berkembangnya penelitian tentang kecerdasan buatan salah satunya tentang pendeteksian objek, teknologi ini dapat membantu untuk mengenali objek pada sebuah gambar. Pendeteksian objek termasuk ke dalam bidang *computer vision* yang merupakan suatu ilmu yang mempelajari tentang seperti apa komputer dapat menganalisis dan melihat pada objek didalam gambar. Proses pengamatan dan perhitungan kendaraan secara manual oleh manusia dalam jangka waktu lama dapat menimbulkan kesalahan perhitungan (*human error*) dikarenakan fokus manusia dapat berkurang apabila melakukan suatu hal dalam waktu yang lama (Adi et al., 2020). Maka dibutuhkan penerapan AI (*Artificial Intelligence*) khususnya *Computer vision* dalam melakukan penghitungan kendaraan secara otomatis. sistem pengawasan lalu lintas cerdas diperlukan untuk membantu manusia mencapai manajemen lalu lintas yang cerdas. Fungsi dari sistem ini adalah untuk mendeteksi kendaraan dalam sebuah video. Kemudian kendaraan akan diklasifikasikan menurut jenisnya seperti sepeda motor, mobil, bus, dan truk. Salah satu algoritma yang digunakan untuk mendeteksi kendaraan adalah algoritma YOLO. Algoritma deteksi objek YOLO adalah yang terbaik untuk digunakan untuk mendeteksi objek dengan akurasi tinggi secara real-time daripada deteksi objek pembelajaran mendalam lainnya seperti DPM dan R-CNN menurut (Bin Zuraimi & Kamaru Zaman, 2021).

Sistem pendeteksi metode YOLOv5 terbukti lebih cepat dan akurat untuk mendeteksi objek pada gambar atau citra sehingga paling sesuai jika diterapkan untuk pendeteksian objek pada video. Algoritma *DeepSORT* telah terbukti menjadi salah satu yang tercepat dan algoritma paling kuat untuk pelacakan beberapa objek (Parico & Ahamed, 2021). Jenis dan jumlah kendaraan yang melintas akan terdeteksi otomatis berdasarkan nilai hasil tingkat akurasinya dan klasifikasi.

Beberapa penelitian terdahulu dilakukan oleh Bin Zuraimi & Kamaru Zaman (2021) yang berjudul "Deteksi dan Pelacakan Kendaraan Menggunakan YOLO dan *DeepSORT*". Hasil dari penelitian ini menunjukan bahwa model terbaik di antara model YOLO adalah Yolov4 yang telah mencapai hasil tercsanggih dengan 82,08% AP50 menggunakan *dataset* khusus pada kecepatan waktu nyata sekitar 14 FPS pada GTX 1660ti. Penelitian lainnya dilakukan oleh Santos et al., (2020) yang berjudul "Menghitung Kendaraan dengan Presisi Tinggi dalam bahasa Brasil Jalan

Menggunakan YOLOv3 dan *DeepSORT*". Hasil dari pengujian ini menemukan nilai optimal untuk Skor YOLO sama dengan 0,7, yaitu confidence 70%. Apalagi nilai dari a*K* sama dengan 7 dipilih untuk membuat trek. Selain itu, dilakukan validasi akurasi YOLOv3 dan *DeepSORT* dengan hiperparameter optimal terpilih, yang mencapai akurasi di atas 90% dalam skenario nyata jalan raya federal Brasil.Penelitian lainnya dilakukan oleh Novita et al., (2021) yang berjudul "Deteksi dan pelacakan berbagai jenis mobil dengan kombinasi model YOLO dan algoritma penyortiran mendalam berdasarkan visi lalu lintas komputer". Berdasarkan hasil pengujian YOLOv4 menghasilkan tingkat akurasi deteksi dengan mAP sebesar 87,98% dimana kombinasi YOLOv4 dengan algoritma *DeepSORT* dapat mendeteksi, melacak dan menghitung 13 jenis kendaraan.

Berdasarkan permasalahan dan penelitian terkait maka akan dilakukan deteksi jumlah kendaraan yang masuk kekota bogor menggunakan YOLOv5 dan *DeepSORT*. Sistem ini diharapkan mampu mendeteksi kendaraan dan menghitung jumlah kendaraan berdasarkan jenisnya.

1.2 Tujuan

Adapun tujuan dari penelitian ini adalah mengimplementasi Algoritma YOLOv5 untuk mendeteksi dan menghitung jumlah kendaraan menggunakan metode *DeepSORT*.

1.3 Ruang Lingkup

Kegiatan penelitian ini, penulisan laporan akan dibatasi pada ruang lingkup sebagai berikut :

1. Objek penelitian yang akan dideteksi yaitu kendaraan berupa sepeda motor, mobil, truk dan bus.
2. Menggunakan metode *You Only Look Once* (YOLOv5) untuk mendeteksi Objek dan *DeepSORT* untuk *tracking* objek.
3. Menggunakan bahasa pemrograman *Python*.
4. Data yang digunakan berupa video yang diambil secara langsung pada ruas jalan raya Kota Bogor JPO Botani berupa video dalam 7 hari, pengambilan video pada pagi dan sore hari.
5. Perkembangan difokuskan pada klasifikasi kendaraan motor, mobil, truk dan bus.
6. Hanya menghitung jumlah kendaraan yang bergerak dari arah Simpang Ekalokasari ke arah JPO Botani.

1.4 Manfaat Penelitian

Adapun manfaat dari penelitian ini adalah:

1. Memberikan alternatif bagi Dinas Perhubungan untuk melakukan survei dengan biaya lebih rendah dan meminimalkan risiko kesalahan saat mensurvei kendaraan yang masuk ke Kota Bogor secara manual.
2. Menjadi referensi dan kontribusi dalam pengembangan teknologi pengenalan objek pada bidang transportasi.
3. Menjadikan penelitian ini sebagai referensi untuk penelitian mendeteksi dan menghitung kendaraan yang menggunakan metode YOLOv5 dan *DeepSORT*.

BAB II
TINJAUAN PUSTAKA

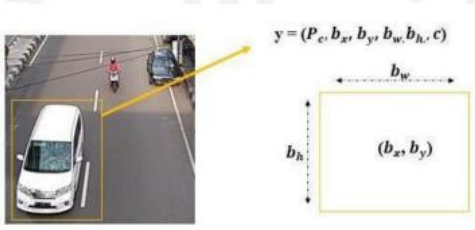
2.1 *You Only Look Once* (YOLO)

YOLO merupakan algoritma deteksi objek yang berasal dari pengembangan metode Convolutional Neural Network (CNN) (Leriansyah & Kurniawardhani, 2020). YOLO menggunakan satu lapisan jaringan syaraf (*Neural Network*) pada citra. Jaringan ini akan membagi citra menjadi beberapa regions (daerah) dan memprediksi *bounding boxes* dan probabilitas setiap *regions* secara bersamaan (Zou et al., 2023)..

Setiap sel grid dirancang untuk memprediksi sebanyak B *bounding boxes* dan *confidence scores* dari setiap *bounding boxes*. *confidence scores* menunjukkan nilai seberapa yakin (*confidence*) Model kemungkinan keberadaan suatu objek dalam *bounding boxes* dan Akurasi perkiraan *bounding boxes* itu sendiri. Nilai *confidence* sebuah *bounding box* dapat dihitung dengan Persamaan 1.

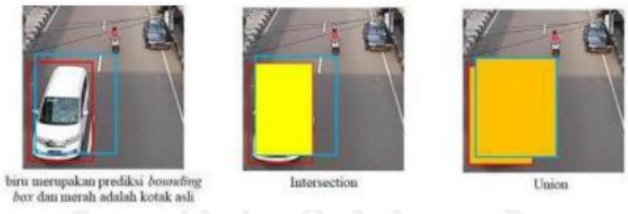
$$\boldsymbol{p}_{\boldsymbol{r}(\boldsymbol{Object})} \text{ X } IOU_{pred}^{truth} \tag{1}$$

Nilai *confidence* juga dapat ditentukan oleh nilai *Intersection Over Union* (IOU) antara *bounding boxes* yang diprediksi dengan *ground truth box* (kotak pembatas yang dibuat oleh manusia selama proses pelabelan dan berisi jenis dan posisi objek yang tepat secara manual). Sebaliknya jika tidak ada objek di sel grid, maka nilai *confidence* nol. Setiap sel grid memprediksi probabilitas kondisional kelas-C menurut (Sauqi, 2022). Setiap *bounding boxes* berisi lima nilai prediksi, yaitu x, y, w, h, dan *confidence*, di mana x dan y adalah koordinat yang mewakili pusat *bounding boxes*, w adalah lebarnya (lebar) *bounding boxes*, h adalah tinggi *bounding boxes*, dan *confidence* adalah Representasi IOU antara *bounding boxes* yang diprediksi dan bidang kebenaran tanah menurut (Sauqi, 2022). Gambar 1, menunjukkan ilustrasi *bounding boxes*.



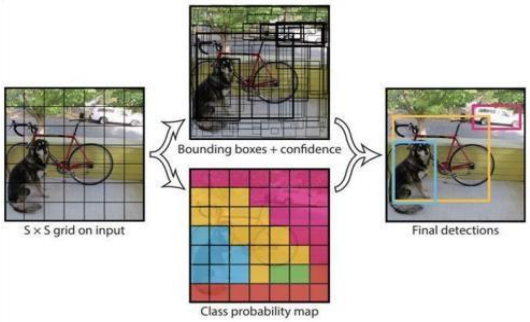
Gambar 1. Komponen *Bounding Box*

Nilai IOU adalah nilai komparatif area yang berpotongan antara area yang diprediksi *bounding box* dan area *ground truth box* dengan area gabungan prediksi *bounding box* dan area *ground truth box*, seperti yang ditunjukkan pada Gambar 2.



Gambar 2. Contoh *Intersection Over Union* (IOU)

Algoritma YOLO membagi citra masukan menjadi beberapa sel grid (kotak kecil) berukuran s X s. Jika titik pusat dari sebuah objek dalam citra masuk ke salah satu sel grid pada citra, maka sel grid tersebut bertanggung jawab untuk mendeteksi objek tersebut menurut (Sauqi, 2022). Proses YOLO ditunjukan seperti Gambar 3.

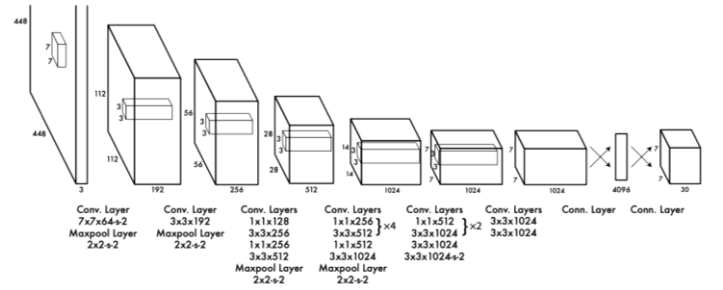


Gambar 3. Proses YOLO

Di berbagai macam versi YOLO, feature extraction pada YOLO v2 menggunakan Darknet-19, YOLO v3 menggunakan darknet-53, dan untuk versi YOLO v5 menggunakan pytorch framework dengan menggunakan CSP Darknet 53 sebagai backbone. YOLO menggunakan input grid S x S. setiap cell pada input grid disebut grid cell bertugas bertanggung jawab dalam memprediksi objek yang ada di dalam grid cell. Sistem pendeteksi metode YOLOv5 terbukti lebih cepat dan akurat untuk mendeteksi objek pada gambar atau citra sehingga paling sesuai jika diterapkan untuk pendeteksian objek pada video sehingga cocok digunakan pada penelitian ini.

2.2 Arsitektur *You Only Look Once* (YOLO)

Algoritma YOLO diimplementasikan sebagai jaringan syaraf *Convolutional* atau *Convolutional Neural Network* (CNN) dan dievaluasi menggunakan PASCAL VOC *Detection Dataset*. Lapisan *Convolutional* digunakan untuk mengekstrak fitur dari citra dan lapisan *Fully Connected* digunakan untuk memprediksi probabilitas dan koordinat keluaran. YOLO terdiri dari 24 lapisan *Convolutional* yang *Fully Connected* (Sauqi, 2022). Berikut Gambar 4 yang memaparkan arsitektur jaringan YOLO



Gambar 4. Arsitektur YOLO

Arsitektur YOLO sebenarnya cukup sederhana. Sistem menerima input gambar dengan bentuk (448, 448, 3), yaitu gambar berukuran 448 x 448 dengan 3 *channel* Kemudian melewati proses *convolutional network* untuk produksi *Output* dengan formulir (7, 7, 30), di mana 7 x 7 mewakili ukuran grid sel (S = 7) dan 30 adalah nilai jumlah *bounding box* B dikalikan dengan jumlah di antara jumlah Kelas dan jumlah komponen dalam kotak B (B×5+C, B = 2, C = 20). Untuk mencari perhitungan *matrix* ditujunkan pada Persamaan 2.

$$matrikyolo = \begin{Bmatrix} x \\ y \\ w \\ h \\ cf \\ c1 \\ c2 \end{Bmatrix} \quad (2)$$

Keterangan :
x = Coordinate bounding Box X
y = Coordinate bounding Box Y
w = Bounding box width
h = Bounding box height
cf = Confidence
c1 = Class 1
c2 = Class 2

apabila satu grid cell terdapat 2 objek dapat dalam satu matrix ditunjukan pada Persamaan 3.

$$\text{Bbox}7 = Bbox\,8 = \begin{Bmatrix} x \\ y \\ w \\ h \\ cf \\ c1 \\ c2 \\ x \\ y \\ w \\ h \\ cf \\ c1 \\ c2 \end{Bmatrix} \quad (3)$$

Nilai parameter koordinat grid cell *(x, y, w, h)* dinormalisasi menjadi ukuran 0 hingga 1. Nilai w (*width*) dan h (*height*) dinormalisasikan dengan cara membagi nilai ukuran *w*, *h* yang merujuk pada ukuran *bounding box* dengan ukuran input gambar. Setiap *grid cell* memiliki *bounding box* masing-masing. *Bounding box* yang memiliki nilai *confidence* 0 tidak diakui dan yang memiliki nilai akan terbentuk *bounding box.* Nilai *x*, *y* dinormalisasikan dengan cara mengurangi titik tengah *bounding box* dengan titik koordinat *grid cell* lalu dibagi dengan lebar *grid cell*. Normalisasi dari *(x,y,w,h)* ditunjukan pada Persamaan 4,5,6 dan 7.

$$normalisasi\ x = \frac{(midpoint\ x - gridcell\ x)}{width\ grid\ cell} \quad (4)$$

$$normalisasi\ y = \frac{(midpoint\ y - gridcell\ y)}{width\ grid\ cell} \quad (5)$$

$$normalisasi\ w = \frac{w}{image\ width} \tag{6}$$

$$normalisasi\ h = \frac{h}{image\ \ height} \tag{7}$$

nilai hasil setiap *grid cell* dapat diilustrasikan pada Tabel 1. Karena setiap *grid cell* mampu mengidentifikasi objek maksimal 2 maka antara *Bbox* 1 dan 2 berada pada 1 *grid cell*.

Tabel 1. Ilustrasi nilai pada tiap *grid cell* (*1 grid cell 2 bbox*)

Bounding Box	*Coordinate (pixel)*				*Confidence*	*Class*	
	Bx	*By*	*Bw*	*Bh*		*Pc1*	*Pc2*
Bbox 1	…	…	…	…	…	…	…
Bbox 2	…	…	…	…	…	…	…
Bbox 3	…	…	…	…	…	…	…
Bbox 4	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…
Bbox 49	…	…	…	…	…	…	…

Selain nilai *(x, y, w, h)* nilai penting untuk menentukan tingkat kepercayaan dari objek deteksi adalah *confidence* (*cf*). nilai *confidence* dilakukan dengan perhitungan melalui Persamaan 8.

$$confidence\ score\ (cf) = P_r(object) * IoU \tag{8}$$

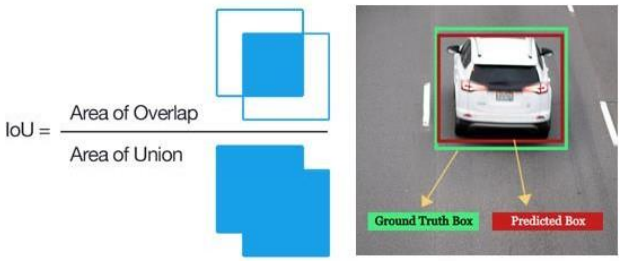
Namun nilai Pr (object) pada *confidence score* dapat dilewati karena nilai Persamaan 9.

$$P(class|object) * P(object) * IoU = P(class) * IoU^{truth} \tag{9}$$

sehingga nilai *confidence score = Iou.* persamaan untuk mencari nilai *Iou* dapat dilihat pada Persamaan 10. IoU sendiri berguna untuk mengukur tumpang tindih antara 2 batas *bounding box* (Jonathan Hui, 2018).

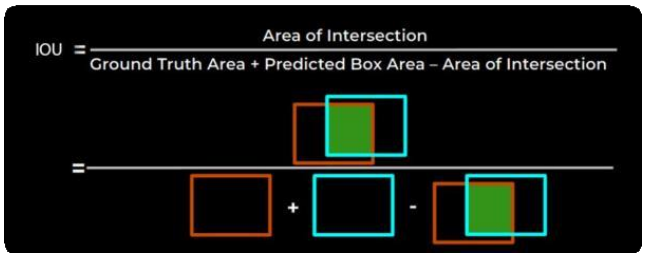
$$IoU_{pred}^{truth} = \frac{Area\ of\ Overlap}{Area\ of\ Union} \tag{10}$$

Pr (*object*) yaitu probabilitas objek yang terdeteksi dan termasuk dalam *class*. Nilai angka 1 mengindikasikan bahwa terdapat objek, sebaliknya nilai 0 mengindikasikan tidak terdapat objek. Nilai *IoU truth pred* yaitu *IoU* antara kotak prediksi dan kebenaran dasar. *IoU* (*intersection over union*) adalah perbandingan antara ukuran *bounding box* dengan *Ground Truth* yang didapat pada saat *training* data masing-masing kelas, ilustrasi *IoU* dapat dilihat pada Gambar 5.



Gambar 5. *ground truth & predicted box yolo*

Pada Gambar 5 *bounding box* berwarna hijau adalah *ground truth bounding box* yang terdapat pada mobil. Sedangkan *bounding box* berwarna merah adalah hasil dari prediksi YOLO. Tugas *Iou* adalah menghitung *confidence score* yang akan ditampilkan nanti pada label. Tentunya pada persamaan *confidence score* nilai $P_r(object)$ adalah 1 dikarenakan terdapat objek yaitu objek mobil. Perhitungan *iou* ditunjukan pada Gambar 6.



Gambar 6. *iou calculation*

YOLO memiliki probabilitas deteksi objek lebih dari 1 *bounding box* dengan kondisi *class* yang sama dengan posisi deteksi yang berbeda. Apabila terdapat lebih dari 1 class yang sama pada objek yang serupa maka akan melalui *non maximum suppression* NMS untuk menghilangkan nilai-nilai yang tidak maksimum. Contoh apabila terdapat 2 *Bbox* dengan *class* yang sama, maka hasil dimensi ukuran tensor dapat dihitung melalui Persamaan 11.

$$Tensor = (S\ x\ S\ x\ (nB\ x5 + nC)) \tag{11}$$

Keterangan :
S = *grid cell*
nB = jumlah Bbox objek
nC = jumlah *class*

2.3 Deteksi Objek

Deteksi objek merupakan teknologi yang dapat mendeteksi objek pada citra atau video. Tujuan deteksi objek adalah untuk mereplikasi kecerdasan yang dimiliki manusia dalam melihat benda menggunakan komputer. Cara kerja deteksi objek adalah deteksi objek menempatkan keberadaan objek dalam gambar dan menggambar kotak pembatas di sekitar objek itu. Ini biasanya melibatkan dua proses, yaitu mengklasifikasikan jenis objek, dan kemudian menggambar kotak di sekitar objek itu. Klasifikasi gambar dan skenario deteksi objek terlihat serupa. Secara umum, objek deteksi adalah mengidentikasi lokasi objek dalam gambar, dan misalnya menghitung jumlah instance suatu objek menurut (Aningtiyas *et al*., 2020).

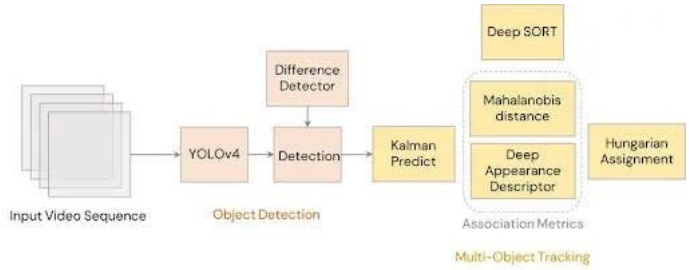
Gambar 7. Deteksi objek pada kendaraan *autonomous*

2.4 Kendaraan

Menurut Pasal 1 Angka 7 UU Nomor 22 Tahun 2009 Lalu Lintas dan Angkutan Jalan, kendaraan adalah suatu sarana angkut di jalan yang terdiri atas kendaraan bermotor dan kendaraan tidak bermotor. Kendaraan bermotor adalah setiap kendaraan yang digerakkan oleh peralatan mekanik berupa mesin selain kendaraan yang berjalan di atas rel, sedangkan kendaraan tidak bermotor setiap kendaraan yang digerakkan oleh tenaga manusia dan/atau hewan (Amwin, 2021). Dalam penelitian ini, akan dilakukan identifikasi terhadap jenis kendaran yang terdeteksi. Jenis kendaraan yang terdeteksi akan dikelompokkan menjadi 4 golongan yaitu sepeda motor, mobil, truk dan bus.

2.5 Deepsort

DeepSORT adalah pendekatan berbasis pembelajaran mendalam untuk melacak objek kustom dalam video menurut (Wojke et al., 2018). *DeepSORT* algoritma tracking-by-detection yang mempertimbangkan parameter kotak pembatas dari hasil deteksi, dan informasi tentang tampilan objek yang dilacak untuk mengaitkan deteksi dalam bingkai baru dengan objek yang dilacak sebelumnya dan mempertimbangkan informasi tentang frame saat ini dan sebelumnya untuk membuat prediksi tentang frame saat ini tanpa perlu memproses keseluruhan video sekaligus. Tahapan *deepsort* ditunjukan pada Gambar 8.



Gambar 8. Tahapan *DeepSORT*

Deepsort terbuat dari 4 komponen utama yaitu sebagai berikut:

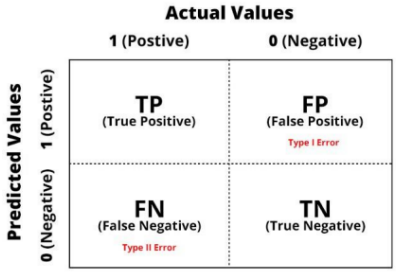
1. *Track handling* dan *estimation*, algoritma *DeepSORT* menggunakan filter *Kalman* untuk memperkirakan trek yang ada dalam bingkai saat ini. Algoritma ini menggunakan filter *Kalman* versi sederhana dan standar, yang menggunakan kecepatan konstan dan pengamatan linier. Dengan cara ini, posisi setiap track yang ada diperkirakan berdasarkan lokasi sebelumnya saat setiap frame baru datang. Estimasi lintasan hanya menggunakan informasi spasial.
2. *Assignment Problem,* dengan perkiraan posisi trek yang ada dan deskriptor tampilan, hasil deteksi baru dengan trek yang ada di setiap bingkai baru Ambang batas keyakinan deteksi digunakan untuk memfilter semua deteksi dengan keyakinan lebih rendah dari ambang batas. Di setiap frame baru, deteksi baru dikaitkan dengan trek yang ada menggunakan matriks biaya dan matriks gerbang ini.
3. *Deep Appearance Descriptor,* Dalam Masukan dari algoritma pelacakan multi-target adalah beberapa bingkai target. Pertama, beberapa target dikodekan dengan DNN untuk menghasilkan deskriptor fitur. Target dari frame berikutnya adalah dicocokkan dengan deskriptor fitur untuk menghasilkan hasil kotak pelacakan. Hasilnya meliputi informasi koordinat kotak pelacakan, tetapi informasi asli dari frame target dihapus, termasuk informasi kepercayaan dan *ID* target.

2.6 Pytorch

PyTorch merupakan pustaka sumber terbuka untuk pembelajaran mesin yang pertama kali dikembangkan oleh *Facebook Research*. *PyTorch* digunakan sebagai pengganti *numpy* untuk menggunakan kekuatan GPU dalam proses pembelajaran mesin. *PyTorch* menyediakan *Tensor* yang dapat hidup di CPU atau GPU dan mempercepat komputasi dalam jumlah besar (Benih *et al*., 2022)

2.7 Confusion Matrix

Confusion Matrix adalah suatu metode yang digunakan untuk melakukan perhitungan akurasi pada konsep *data mining* (Pratiwi et al., 2021). *Confusion Matrix* merupakan salah satu *tools analitik prediktif* yang menampilkan dan membandingkan nilai aktual dan nilai sebenarnya dengan nilai hasil *prediktif* model yang dapat digunakan untuk menghasilkan metrik evaluasi seperti *Accuracy* (akurasi), *Precision*, *Recall*, dan *F1-Score* atau *F-Measure* ditunjukan pada Gambar 9.



Gambar 9. *Confusion Matrix*

1. *True Positive* (*TP*)
   Merupakan kebenaran dari pendeteksian objek oleh sistem.
2. *False Positive* (*FP*)
   Merupakan Noise yang tidak diinginkan namun terdeteksi oleh sistem.

3. *False Negative* (*FN*)
   Merupakan objek yang seharusnya terdeteksi namun tidak terdeteksi oleh sistem.
4. *Accuracy*
   Merupakan perhitungan deteksi objek pada keseluruhan *frame* dengan Persamaan 15.

$$accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (15)$$

5. *Precision*

Merupakan jumlah prediksi dari hasil deteksi objek yang benar dibandingkan keseluruhan hasil yang terprediksi oleh sistem. Menghitung tingkat *precision* akan menjawab jumlah objek yang terdeteksi benar dari keseluruhan proses deteksi objek yang dilakukan oleh sistem. Persamaan untuk menghitung *precision* dapat dilakukan qseperti Persamaan 16.

$$precision = \frac{True\ Positive}{(\text{True Positive} + \text{False Positive})} \quad (16)$$

6. *Recall*

Proses *recall* akan menjawab jumlah hasil objek yang terdeteksi dengan benar dari keseluruhan jumlah objek yang sebenarnya terdeteksi oleh sistem. Persamaan untuk menghitung *recall* dapat dilakukan seperti Persamaan 17.

$$\text{Recall} = \frac{True\ Positive}{(\text{True Positive} + \text{False Positive})} \quad (17)$$

2.8 Penelitian Terdahulu

Terdapat beberapa penelitian terdahulu yang menjadi Referensi untuk Implementasi Algoritma YOLOv5 Untuk Mendeteksi dan Menghitung Jumlah Kendaraan Menggunakan Metode *DeepSORT*.

1. Nama : Muhammad Azhad dan Fadhlan Hafizhelmi Kamaru Zaman
   Judul : Deteksi dan Pelacakan Kendaraan menggunakan YOLO dan *DeepSORT*
   Tahun : 2021
   Isi : Penelitian ini memiliki tujuan untuk pendeteksian kendaraan di jalan yang akurat dan cepat dengan menggunakan volume kendaraan sebagai data berharga untuk mendeteksi kemacetan lalu lintas menggunakan YOLO dan *DeepSORT*. Hasil dari penelitian ini menunjukan bahwa model terbaik di antara model YOLO adalah Yolov4 yang telah mencapai hasil tercanggih dengan 82,08% AP50 menggunakan *dataset* khusus pada kecepatan waktu nyata sekitar 14 FPS pada GTX 1660ti.

2. Nama : Narinder Singh Punn, Sanjay Kumar Sonbhadra, Sonali Agarwal dan Gaurav Rai
   Judul : Memantau jarak sosial COVID-19 dengan deteksi dan pelacakan orang melalui YOLO yang disetel dengan baik v3 dan teknik *DeepSORT*
   Tahun : 2021

Isi : Penelitian ini memiliki tujuan untuk mengotomatiskan tugas pemantauan jarak sosial menggunakan video pengawasan menggunakan model deteksi objek YOLO v3 untuk memisahkan manusia dari latar belakang dan pendekatan *DeepSORT* untuk melacak orang yang teridentifikasi dengan bantuan kotak pembatas dan ID yang ditetapkan. Hasil dari penelitian ini menunjukan bahwa bahwa YOLO v3 dengan skema pelacakan *DeepSORT* menampilkan hasil terbaik dengan skor mAP dan FPS yang seimbang untuk memantau jarak sosial secara real-time.

3. Nama : Nisma Novita Hasibuan, Muhammad Zarlis dan Syahril Efendi
   Judul : Deteksi dan pelacakan berbagai jenis mobil dengan kombinasi model YOLO dan algoritme penyortiran mendalam berdasarkan visi lalu lintas komputer
   Tahun : 2021

   Isi : Penelitian ini memiliki tujuan untuk memperluas penelitian sebelumnya dengan memecah masalah menjadi sub-tugas yang berbeda menggunakan pendekatan YOLOv4 yang dikombinasikan dengan algoritma *DeepSORT* untuk mendeteksi dan melacak objek secara langsung pada rekaman CCTV aktivitas kendaraan di jalan raya tiga perhentian kota. Berdasarkan hasil pengujian YOLOv4 menghasilkan tingkat akurasi deteksi dengan mAP sebesar 87,98% dimana kombinasi YOLOv4 dengan algoritma *DeepSORT* dapat mendeteksi, melacak dan menghitung 13 jenis kendaraan.

4 Nama : Yao Zhang, Zhiyong Chen dan Bohan Wei
   Judul : Pelacakan Objek Atlet Olahraga Berdasarkan *DeepSORT* dan Yolo V4 dalam Kasus Gerakan Kamera
   Tahun : 2021
   Isi : Penelitian ini memiliki tujuan untuk membuat sistem pelacakan untuk semua pemain dalam permainan, menyelesaikan pelacakan waktu nyata dari setiap atlet, sehingga mendapatkan informasi trek yang relevan. Menggunakan YOLOv4 dan DeeepSORT. Berdasarkan hasil pengujian menggunakan INRIA Person *Dataset* untuk melatih Yolo V4, maks iou adalah 0,97673. Kemudian menggunakan *dataset* Market1501 & MARS untuk melatih *DeepSORT*, hasilnya peringkat-1 dengan skor 82,3%, peringkat-5 dengan skor 90,4% dan peringkat-10 dengan skor 93,7%. Dengan skor mAP 68,2%.

5 Nama : Adson M. Santos, Carmelo J. A. Bastos-Filho, Alexandre M. A. Maciel, Estanislau Lima
   Judul : Menghitung Kendaraan dengan Presisi Tinggi dalam bahasa Brasil Jalan Menggunakan YOLOv3 dan *DeepSORT*
   Tahun : 2020
   Isi : Penelitian ini memiliki tujuan untuk mengusulkan sistem yang menggunakan YOLOv3 untuk deteksi objek dan *DeepSORT* untuk

algoritma pelacakan beberapa objek. Hasil dari pengujian ini menemukan nilai optimal untuk Skor YOLO sama dengan 0,7, yaitu confidence 70%. Apalagi nilai dari K sama dengan 7 dipilih untuk membuat trek. Selain itu, kami memvalidasi akurasi YOLOv3 dan *DeepSORT* dengan hiperparameter optimal terpilih, yang mencapai akurasi di atas 90% dalam skenario nyata jalan raya federal Brasil.

2.9 Tabel Perbandingan Penelitian

Berdasarkan penjelasan pembahasan mengenai beberapa penelitian terdahulu, berikut tabel perbandingan penelitian yang dapat dilihat pada Tabel 2.

Tabel 2. Perbandingan Penelitian

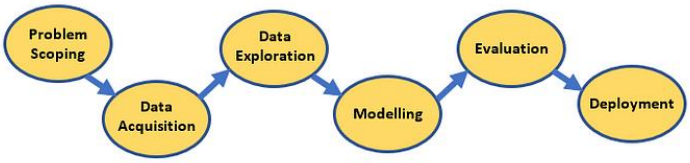
No	Nama Peneliti	Judul	Metode		Data Set	Kelas Keluaran	Hasil
			YOLO	*DeepSORT*			
1	Muhammad Azhad dan Fadhlan Hafizhelmi Kamaru Zaman (2021)	Deteksi dan Pelacakan Kendaraan menggunakan YOLO dan *DeepSORT*	✓	✓	*Dataset Custom*	Motor, mobil, truk, dan bus	Yolov4 yang telah mencapai hasil tercanggih dengan 82,08%
2	Narinder Singh Punn, Sanjay Kumar Sonbhadra, Sonali Agarwal dan Gaurav Rai (2021)	Memantau jarak sosial COVID-19 dengan deteksi dan pelacakan orang melalui YOLO yang disetel dengan baik v3 dan teknik *DeepSORT*	✓	✓	PASCAL-VOC and MS-COCO *dataset*	Manusia	Yolov3 yang telah mencapai hasil Nol 7560, mAP 0,846, TL 0,87dan FPS 23
3	Nisma Novita Hasibuan, Muhammad Zarlis, dan Syahril Efendi (2021)	Deteksi dan pelacakan berbagai jenis mobil dengan kombinasi model YOLO dan algoritme *DeepSORT* mendalam berdasarkan visi lalu lintas komputer	✓	✓	kitti *dataset* and LSVH *dataset*	City car, mini box, low mpv dan medium suv.	Akurasi deteksi dengan mAP sebesar 87,98% dimana kombinasi YOLOv4 dengan algoritma *DeepSORT* dapat mendeteksi, melacak dan menghitung 13 jenis kendaraan

4	Yao Zhang, Zhiyong Chen, dan Bohan Wei (2020)	Pelacakan Objek Atlet Olahraga Berdasarkan *DeepSORT* dan Yolo V4 dalam Kasus Gerakan Kamera	✓	✓	*Dataset* dengan Market1501 & MARS	Manusia	Menggunakan INRIA Person *Dataset* untuk melatih Yolo V4, maks iou adalah 0,97673., menggunakan dataset Market1501 & MARS untuk melatih DeepSort, hasilnya peringkat-1 dengan skor 82,3%, peringkat-5 dengan skor 90,4% dan peringkat-10 dengan skor 93,7%.
5	Adson M. Santos, Carmelo JA Bastos-Filho, Alexandre MA Maciel, dan Estanislau Lima (2020)	Menghitung Kendaraan dengan Presisi Tinggi dalam bahasa Brasil Jalan Menggunakan YOLOv3 dan *DeepSORT*	✓	✓	GRAM *dataset* dan the CD2014 *dataset*	truk, mobil, sepeda motor, dan bus.	Akurasi YOLOv3 dan *DeepSORT* dengan hiperparameter optimal terpilih, yang mencapai akurasi di atas 90% dalam skenario nyata jalan raya federal Brasil
6	Zaka Darmawan (2023)	Implementasi Algoritma YOLOv5 Untuk Mendeteksi dan Menghitung Jumlah Kendaraan Menggunakan Metode *DeepSORT*	✓	✓	*Dataset Custom*	motor, mobil, truk, dan bus	-

BAB III
METODE PENELITIAN

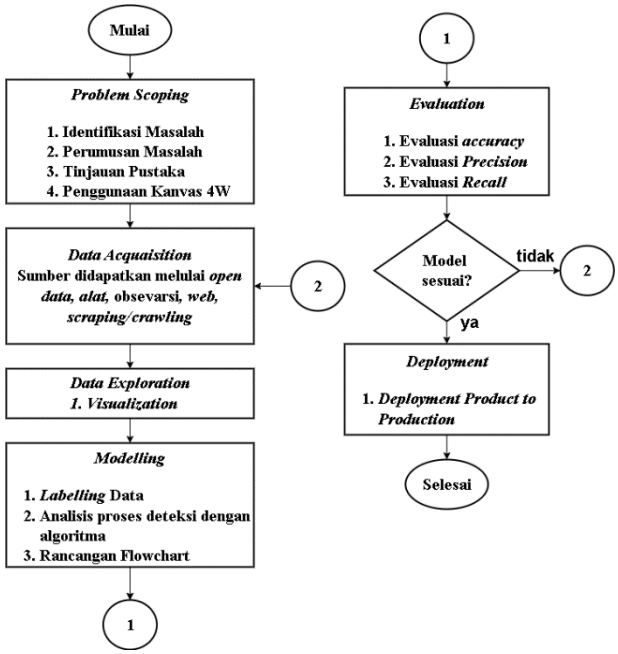
3.1 Metodologi Penelitian

Metodologi yang diterapkan pada penelitian ini menerapkan pendekatan *AI Project Cycle*. Di mana dalam melakukan pengembangan proyek AI secara utuh dilakukan melalui beberapa tahapan proses secara bertahap.*AI project cycle* bisa diartikan sebagai sebuah proses dalam membuat proyek AI secara utuh (Nafi'Ah et al., 2021). Proses secara bertahap tersebut dibagi menjadi 5 stage yaitu *problem scoping, data acquisition, data exploration, modeling, evaluation*, dan *deployment*.



Gambar 10. *AI Project Cycle*

Setelah melalui proses tersebut sampai ke proses *deployment*, model yang telah dikembangkan akan dilakukan proses implementasi AI dilakukan pada sistem dengan model yang telah dilatih yang dapat dilihat langsung dengan *output* deteksi dari model sesuai pada tujuan yang diinginkan. *Output* yang dihasilkan harus memiliki arti bahwa sesuai dengan tujuan pada proses *problem scoping*. Alur penelitian ini dapat dilihat pada *flowchart* Gambar 11.



Gambar 11. Diagram alur penelitian menggunakan metode *AI Project Cycle*

3.1.1 *Problem Scoping*

Problem scoping adalah proses mengidentifikasi batas-batas masalah yang diselesaikan untuk mencapai tujuan yang terarah dan jelas. Dalam penelitian ini: proses identifikasi masalah dilakukan dengan mengamati kendaraan yang masuk ke kota Bogor dengan dengan membedakannya menjadi 4 *class* yaitu, motor, mobil, bus

dan truk. Setelah mengidentifikasi masalah tahap selanjutnya, merumuskan masalah dengan tujuan menyelesaikan masalah yang selesai berfokus pada solusi yang sedang dibangun dan memiliki keterbatasan masalah dalam penelitian. Rumusan masalah dalam penelitian ini adalah bagaimana mengembangkan model yang dapat mengenali atau mengidentifikasi kendaraan yang masuk ke kota Bogor melalui tangkapan kamera.

Untuk proses pembelajaran, para peneliti mencari berbagai sumber sebagai referensi dalam pembuatan model sebagai bahan penelitian atau tinjauan pustaka, meliputi dari Jurnal, skripsi, *dataset* dari Internet, artikel yang terkait dengan topik penelitian. Salah satu yang melakukan proses identifikasi masalah cara menerapkan kerangka kerja 4W (*Who, What, Where, Why*) untuk mempermudah proses *problem scoping*.

3.1.2 *Data Acquisition*

Pada tahap ini dilakukan pengumpulan data-data yang dibutuhkan untuk mendukung proyek AI yang akan dibangun. Proses ini wajib dalam pengembangan proyek AI, karena data dianalisis dan diproses menjadi model data latih. Data yang dapat digunakan dapat berupa informasi dan statistik. Beberapa Sumber data yang dapat digunakan dalam pengumpulan data dalam proses data acquisition, contohnya yaitu,

1. *Tools*/Alat *: Camera, Computer.*
2. Observasi *:* Survey, Penelitian.
3. *Open Data :* Data diambil dari situs *BPS, Kaggle, Google Images*, menggunakan data sendiri.

Data yang dikumpulkan diperlukan untuk tujuan referensi atau analisis, diperlukan untuk menentukan tipe data yang sesuai dengan permasalahan yang sudah ditentukan. Data ini akan diproses *labelling* yang dilakukan pada tahap *modeling*.

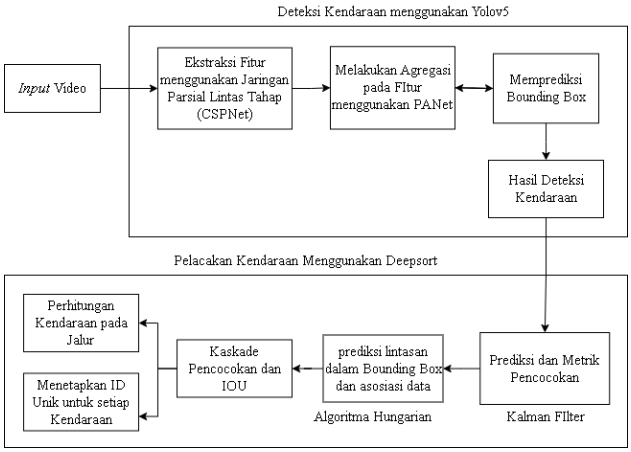
3.1.3 *Data Exploration*

Pada tahap ini bertujuan untuk melakukan visualisasi dengan melihat pola data tersebut dari data yang sudah terkumpul di tahap sebelumnya. Metode yang dapat dilakukan dalam proses data *exploration* yaitu *Visualization*, merupakan penyajian data dalam bentuk grafik (Bar Chart, Histogram, Box Plot, Scatter Plot, Star Plot, Chernoff Plot, Maps). Penelitian kali ini dalam melakukan visualisasi data hanya menggunakan 4 *class* yaitu sepeda motor, mobil, bus, dan truk. Berikut penjelasan mengenai data *class* yang digunakan:

1. Sepeda Motor, kendaraan beroda dua yang digerakkan oleh sebuah mesin.
2. Mobil, kendaraan beroda empat yang menggunakan bahan bakar untuk menghidupkan mesinnya
3. Bus, kendaraan bermotor yang dilengkapi dengan tempat duduk untuk lebih dari delapan orang, tidak termasuk tempat duduk untuk pengemudi, baik dilengkapi atau tidak dilengkapi bagasi.
4. Truk, kendaraan bermotor yang digunakan untuk angkutan barang, selain mobil penumpang, mobil bis, dan kendaraan bermotor roda dua.

3.1.4 *Modelling*

Dalam penelitian ini algoritma *deep learning* YOLO (*You Only Look Once*) dan *DeepSORT* dalam pembuatan model kecerdasan buatan dan menggunakan pendekatan *learning based*. *Learning based* didasarkan pada pengalaman pembelajaran *machine learning* dengan data yang sudah dilatih. *Modelling flowchart* diagram ditunjukan pada Gambar 12.



Gambar 12. *Modelling Flowvchart diagram*

Pada gambar 13 menunjukan proses sistem yang akan berjalan, setelah yolov5 mampu mendeteksi object deepsort melakukan prediksi lintasan dalam *boundingbox*dan pencocokan *IOU* untuk menetapkan ID dan perhitungan umlah kendaraan. Dalam model penelitian ini, yang menggunakan jenis *supervised learning*, yaitu data yang telah dikumpulkan dan diberi label dengan identifikasi objek, proses *labelling* data pada dilakukan pada *platform Roboflow* yang memiliki dukungan format YOLO v5 yang berbasis *supervised learning.* Hasil *output* file setelah melakukan *labelling* dengan ekstensi XML yang memuat data informasi dari citra gambar yang sebelumnya sudah diberikan label.

3.1.5 *Evaluation*

Tahap selanjutnya yaitu *evaluation*, yaitu proses evaluasi dari hasil model yang telah dieksekusi dengan data latih. Untuk bentuk klasifikasi *multi label* pada dasarnya sama dengan *multi class* dimana data dikelompokkan menjadi beberapa kelas (Rasywir et al., 2020). Evaluasi yang dilakukan yaitu perhitungan akurasi menggunakan *confusion metric* seperti yang ditunjunkan pada Gambar 9.

3.1.6 *Deployment*

Tahap terakhir yaitu *deployment*. Proses implementasi AI pada sistem dengan model yang telah dilatih dapat langsung dilihat dengan *output* deteksi dari model yang sudah dilatih sesuai tujuan yang dikehendaki. Pada penelitian ini *output* berupa model dan berbasis web ditunjukan pada Tabel 3.

Tabel 3. *Model Output*

No	*Output*	Keterangan
1	Model	implementasi model dilakukan dengan proses running program secara langsung menggunakan *default python* yang tersedia proses informasi disajikan pada *command line*.
2	*Website*	Implementasi model *website* menggunakan *user interface* untuk memudahkan dalam navigasi, *user experience* yang lebih baik dan informasi tambahan dapat ditampilkan

3.2 Alat dan Bahan

3.2.1 Alat

Alat yang dibutuhkan pada penelitian ini berupa perangkat lunak (software) dan perangkat keras (hardware) yaitu :

a. Perangkat Lunak (*Software*)

Perangkat lunak yang digunakan adalah:

1. Sistem Operasi Windows 10 64-Bit
2. Google Chrome
3. Visual Studio Code
4. Microsoft Office

b. Perangkat Keras (*Hardware*)

Perangkat keras yang digunakan adalah sebuah personal komputer dengan spesifikasi:

1. Laptop Acer Aspire
2. Processor: AMD Ryzen 3 2200U
3. RAM: 16.00 GB

3.2.2 Bahan

Bahan yang diperlukan pada penelitian ini adalah:

1. Jurnal, media cetak, dan internet sebagai penunjang referensi dalam pelaksanaan penelitian dan pembuatan laporan proposal penelitian.
2. Buku panduan penulisan skripsi dan tugas akhir Universitas Pakuan Program Studi Ilmu Komputer, juga data lain yang mendukung dalam penulisan laporan ini.

BAB IV
PERANCANGAN DAN IMPLEMENTASI

4.1 *Problem Scoping*

Pada tahap problem scoping, peneliti melakukan observasi di jalan raya perkotaan Bogor dengan mengamati jumlah kendaraan yang masuk ke kota Bogor. Selain itu, untuk identifikasi masalah, kerangka kerja 4*W* digunakan untuk melakukan *problem scoping* dengan melingkupi sesuai dengan batasan-batasan masalah yang akan diselesaikan.

1. *Who* : kendaraan yang bergerak dari Simpang Ekalokasari ke arah JPO Botani.
2. *What* : Objek yang akan diteliti adalah menghitung kendaraan yang teridentifikasi dibagi menjadi 4 *class*, yaitu sepeda motor, mobil, bus, dan truk.
3. *Where* : peneliti akan melakukan pengamatan arus lalu lintas Jalan Raya Kota Bogor.
4. *Why* : dengan menggunakan sistem berbasis pengenalan objek, informasi jumlah kendaraan yang masuk ke kota Bogor dapat diperoleh. Secara khusus, penggunaan sistem pengenalan objek dapat meningkatkan fungsi pengawasan video yang umumnya hanya memantau arus lalu lintas.

4.2 *Data Acquisition*

pengumpulan data-data yang dibutuhkan dalam proses pembuatan model dilakukan dengan beberapa cara diantaranya:

1. Alat

Alat yang digunakan yaitu kamera Canon 1200D dan *smartphone* yang berfungsi memgambil gambar dan video untuk pemrosesan input gambar, dan pengumpulan *dataset*.

2. Observasi

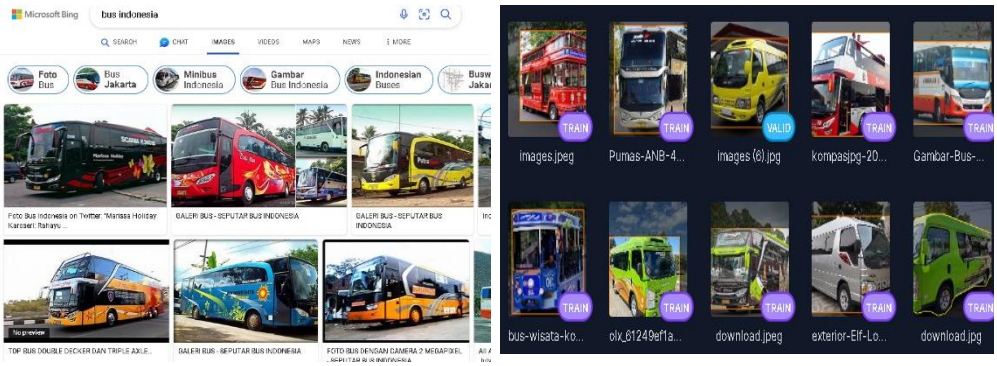
Pengamatan dilakukan di ruas jalan raya perkotaan Bogor di JPO Botani dengan jarak 50 meter untuk mengumpulkan data kendaraan dengan cara merekam kondisi lalu lintas menggunakan *smartphone*.

Gambar 13. Pengumpulan data melalui observasi

Pada Gambar 13 pengambilan data diatas JPO dengan jarak 50 meter, pengambilan data berupa video yang direkam pada waktu pagi hari dan sore hari.

3. *Open Data*

Pada penelitian ini, data tambahan diperlukan agar model dapat mencapai hasil yang lebih baik dalam proses *training*. Untuk proses pengumpulan data tambahan, digunakan fungsi pencarian gambar pada Google beserta kumpulan gambar pada situs *Roboflow Universe,* ditunjukan pada Gambar 14.

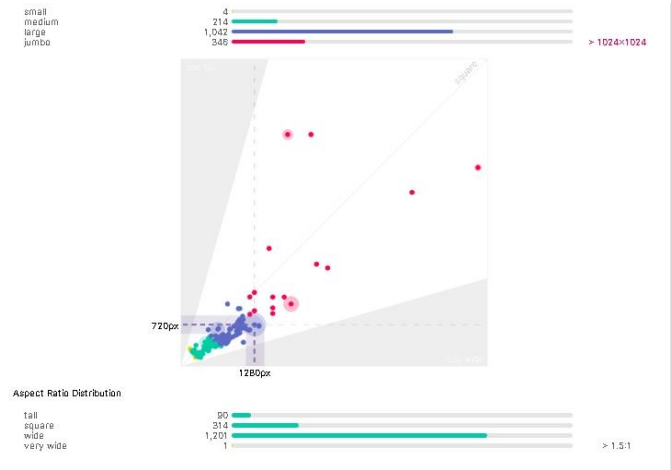


Gambar 14. Pengumpulan data melalui open data

Selama pengumpulan data, total ada 1606 gambar yang dikumpulkan dengan 4 *class*, yaitu sepeda motor, mobil, bus, dan truk. Dari data yang terkumpul, ukuran rata-rata *dataset* adalah 1280x720 piksel dengan ukuran 100 MB.

4.3 *Data exploration*

Setelah mengumpulkan gambar melalui proses data *acquisition*, kumpulan gambar akan divisualisasikan dengan melihat pola datanya. Data yang dikumpulkan merupakan gambar yang termasuk ke dalam total 4 *class*. Berikut ini adalah hasil distribusi ukuran *dataset* yang akan diolah menjadi model pada Gambar 15.



Gambar 15. Sebaran informasi data citra

Setelah melalui proses pengumpulan gambar, *dataset* akan dibagi menjadi 3 bagian yaitu *data training*, data *validation* dan *data testing*. Pembagian jumlah gambar yang akan digunakan pada kumpulan *dataset* ini adalah 70% untuk *data training*, 20% untuk *data validation* dan 10% untuk *data testing*.

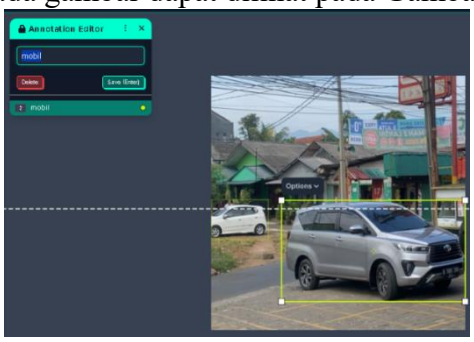
Dari jumlah citra sebanyak 1606 gambar, dilakukan proses *augmentasi* data dimana gambar tersebut mengalami proses *flip horizontal* untuk membuat variasi data baru

untuk *training*, sehingga total gambar *dataset* yang digunakan adalah sebanyak 2447 citra, dengan jumlah *class* pembagian data:

1. *data training* sebanyak 1614 gambar
2. *data validation* sebanyak 466 gambar
3. *data testing* sebanyak 221 gambar

4.4 *Modeling*

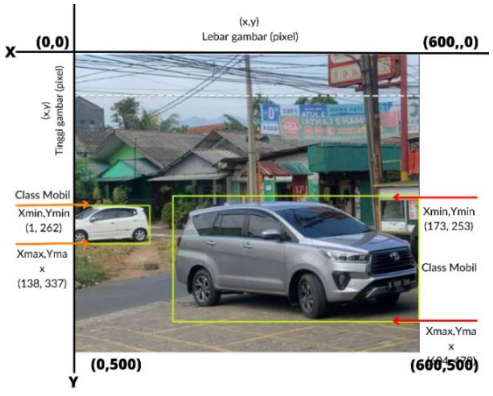
Sebelum melakukan proses pemodelan, *dataset* gambar harus diberi label sesuai dengan *class* yang telah ditentukan. Proses pelabelan data dilakukan karena dalam CNN YOLO termasuk dalam kategori *supervised learning*, di mana data dipelajari berdasarkan label dan algoritma kemudian mempelajari pola dari pasangan data di antara label tersebut. Peneliti melabeli data menggunakan *Roboflow*. Proses melakukan labelling pada gambar dapat dilihat pada Gambar 16.



Gambar 16. Proses labeling gambar

Setelah gambar diberi label, sebuah *output* dihasilkan yang berisi koordinat *bounding box* dan *class*. Output yang dihasilkan untuk setiap gambar adalah dalam format VOC XML. Hal ini karena YOLOv5 menggunakan data dalam format .txt. Data XML dikonversi dengan *LabelImg*, yang menghasilkan *output* pada Lampiran 18.

Setelah mengonversi data pelabelan ke dalam format YOLO v5 *Pytorch*, nilainya dinormalisasi dalam kisaran 0 hingga 1. Untuk gambaran umum lokasi koordinat pada data yang diperoleh setelah pelabelan gambar, lihat Gambar 17.



Gambar 17. Koordinat citra gambar

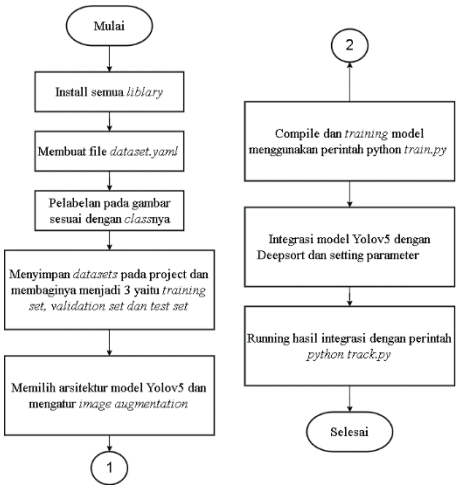
Untuk gambar yang diberi label setelah *export dataset, data anotasi* sesuai dengan *class* pasangan gambar sehingga *dataset* dapat digunakan untuk proses *modeling*.

Pengembangan model dalam penelitian ini untuk *class dataset*, label dan definisi *class* ditunjukkan pada Tabel 4.

Tabel 4. *Label* dan *Class Dataset*

Number Class	*Label*	Definisi *Class*
0	Bus	Gambar berupa kendaraan bus
1	Mobil	Gambar berupa kendaraan mobil
2	Motor	Gambar berupa kendaraan motor
3	Truk	Gambar berupa kendaraan truk

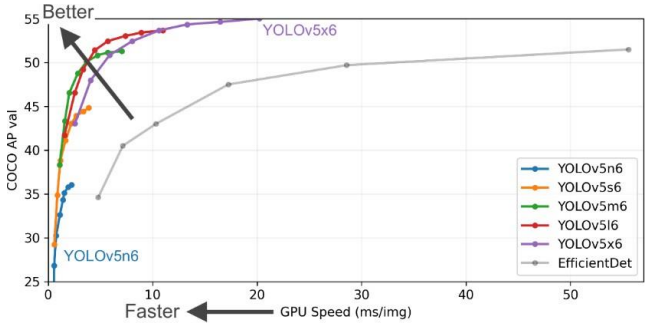
Proses *Modelling* ditampilkan menggunakan diagram air yang ditunjukan pada Gambar 18.



Gambar 18. Flowchart proses *Modelling*

Model yang digunakan pada penelitian ini yaitu *base on* YOLO v5s, pemilihan model mempengaruhi kecepatan dalam proses pembuatan model dan saat model dijalankan. Berikut merupakan tabel perbandingan jenis *pretained model* yang telah disediakan oleh YOLO v5 dapat dilihat pada Lampiran Tabel 5.

Nilai dari data pengujian ditentukan oleh pengujian yang dilakukan oleh ultralytics menggunakan dataset COCO val2017. Dalam penelitian ini, *base model* YOLO v5s digunakan karena diasumsikan bahwa model yang akan dikembangkan cukup untuk mengenali objek yang dituju dengan menerima jumlah *frame per second* yang tinggi untuk menguji hasil *realtime* yang baik.Hasil *benchmark* untuk setiap model ditunjukkan pada Gambar 19.



Gambar 19. Yolo model comparasion

Pada saat proses *training* model YOLO CNN, ada beberapa parameter yang diperlukan untuk menentukan keberlangsungan proses *training*, antara lain penentuan *epoch, batch size, optimizer, class config file, img-size*, dijelaskan pada sub-bab dibawah ini :

4.4.1 Epoch

Epoch adalah nilai dari proses *training* yang menentukan berapa kali algoritma *deep learning* akan berjalan melalui seluruh *dataset* dengan ketentuan ketika seluruh training data selesai, maka akan dilanjutkan dengan nilai *epoch* berikutnya. Nilai *epoch* biasanya lebih besar dari 1 untuk mencapai akurasi model yang optimal. Pada penelitian ini, digunakan nilai *epoch* sebesar 150.

4.4.2 Batch Size

Batch size adalah proses *training* jumlah sampel data pada setiap *epoch* yang untuk pembaruan *weight* yang dilakukan oleh jaringan saraf tiruan (JST) pada keseluruhan sampel *dataset* tertentu. Pada penelitian ini, digunakan parameter *batch size* sebesar 32.

4.4.3 Optimizer

Optimizer merupakan proses optimasi nilai untuk menemukan *weight* yang memberikan output lebih baik. *Gradient Descent* merupakan metode optimasi yang paling sering digunakan untuk pelatihan jaringan saraf tiruan (JST). Pada Implementasi pengembangan YOLOv5 terdapat 2 metode *optimizer* yaitu SGD (*Stochastic Gradient Descent*), dan *ADAM*. Proses *optimizer* dilakukan untuk mengembangkan model parameter yang digunakan pada algoritma *optimizer* yaitu *Learning Rate* dan *Batch size*.

4.4.4 Learning Rate

Learning rate digunakan untuk mengatur besarnya langkah pada *Gradient Descent*. Nilai *learning rate* menentukan cepat atau lambatnya proses *training* model, jika angka *learning rate* yang diberikan besar maka proses *training* model akan lebih cepat namun hasil model akan kurang optimal dan jika angka *learning rate* yang diberikan terlalu kecil maka proses *training* model berlangsung lebih lama akan menyebabkan *training error* yang tinggi, nilai *Learning rate* memiliki rentang nilai 0 hingga 1.

4.4.5 Metrik Instance

Merupakan ambang batas untuk menentukan kesamaan objek dengan ReID, semakin tinggi nilainya semakin mudah untuk menganggap itu adalah objek yang sama, nilai *MAP* memiliki nilai 0,25.

4.4.6 *Torchreid*

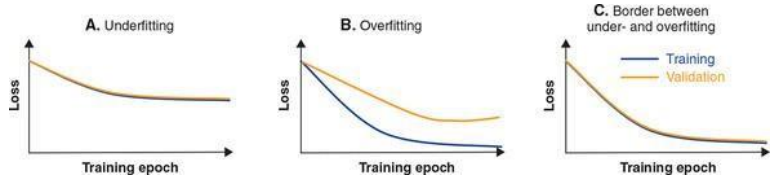
Merupakan pustaka perangkat lunak yang dibangun diatas *PyTorch* untuk menyediakan *dataset* ID ulang gambar dan video, yang disederhanakan untuk pengembangan cepat dan pelatihan evaluasi model ID ulang yang mendalam.

4.5 *Evaluation*

Pada tahap ini dilakukan proses mengukur seberapa baik model melakukan pengenalan sesuai dengan yang diharapkan.

Pada tahap evaluasi data yang disajikan meliputi data *epoch, train/box_loss, train/obj_loss, precision, recall, mAP, val/box_loss, val/obj_loss* yang terdapat pada history training pada *file* result.csv

Gambar 20. Evaluasi plot model



Terjadinya *Overfitting* dan *Underfitting* menyebabkan model tidak bekerja dengan baik. Untuk lebih jelasnya berikut penjelasan mengenai *Overfitting, Underfitting* dan YOLO v5 *hyperparams* untuk menambah variasi data baru.

4.5.1 Overfitting

Overfitting adalah kondisi dimana hampir semua data yang telah melalui proses training mencapai persentase yang baik, tetapi terjadi ketidaksesuaian pada proses prediksi (Nugroho et al., 2020). Penyebab terjadinya *overfitting* pada model yaitu:

1. Data sampel berisi banyak informasi yang tidak relevan, yang disebut data tidak berarti.
2. Model melatih terlalu lama dalam satu set data sampel.
3. Model sangat kompleks sehingga model mempelajari data tidak berarti dalam data pelatihan.

Untuk mencegah terjadinya *Overfitting* pada model maka dilakukan:

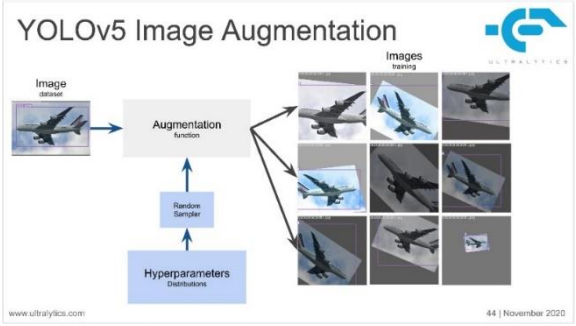
1. Membagi *dataset* kendaraan menjadi tiga yaitu 70 :20 : 10 , 70% untuk *data training,* 20% *data valid* dan 10% *data test.*
2. Menggunakan data hasil observasi secara langsung berupa video di JPO Botani Kota Bogor.

4.5.2 Uderfitting

Underfitting dapat terjadi akibat dua hal, model yang dibuat terlalu sederhana, ataupun model yang dibuat memiliki regulasi yang terlalu banyak. Pada penelitian ini dilakukan proses *augmentasi* menggunakan *flip horizontal* pada *dataset* yang telah dikumpulkan.

4.5.3 Yolo Hyperparams

Yolo *Hyperparams* dilakukan bertujuan untuk mengoptimalkan metrik performa model *deeplearning* seperti akurasi, *precision* dan *recall.*



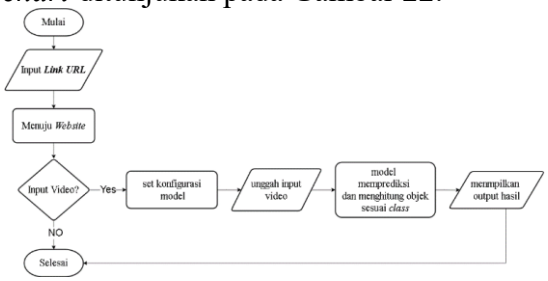
Gambar 21. *Image augmentation* yolov5

Sebagai upaya lebih lanjut untuk memperbaiki model dalam kasus *overfitting*, penelitian ini akan menggunakan *hyperparameter* YOLO v5 untuk mendapatkan variasi data baru untuk mendapatkan variasi data baru dalam melakukan optimasi model termasuk optimasi nilai :

1. *Leaning rate* adalah nilai koreksi bobot pada waktu proses training
2. *Batch size* adalah jumlah sampel dari dataset yang dimasukkan ke dalam model.
3. *Flip* Menggunakan konfigurasi *flip* pada gambar, terdapat 2 *flip* pada YOLOv5 yaitu *flipud* melakukan pembalikan secara vertikal sedangkan *fliplr* melakukan pembalikan secara horizontal.

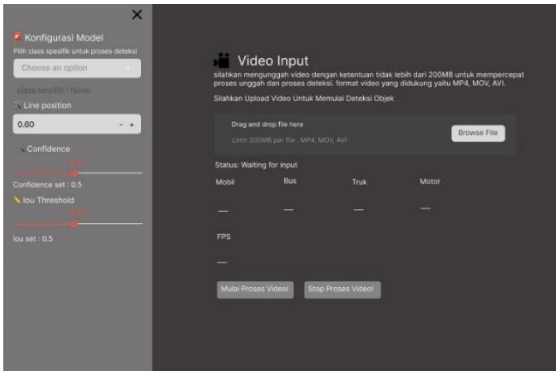
4.6 Implementasi Model *Website*

Untuk pembuatan *website* langkah awal yang harus dilakukan yaitu membuat *flowchart*. Ketetapan yang diinginkan pada *website* yaitu model dapat menerima *input* melalui video. *Flowchart* ditunjukan pada Gambar 22.



Gambar 22. *Flowchart* model *website*

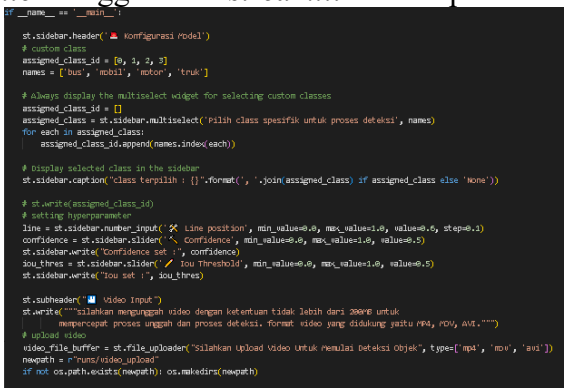
Selanjutnya untuk tahap *deployment website* pada server hosting menggunakan *github* dan *streamlit cloud.* Dalam membangun sebuah *website* hal pertama yang harus dilakukan yaitu pembuatan desain *wirreframe website*. Menggunakan software *figma.* Rancangan halaman *website* yang akan dibuat yaitu 1 halaman yang memuat halaman video *input.* Hasil pembuatan rancangan *user interface* dapat dilihat pada Gambar 23.



Gambar 23. Rancangan *user interface*

Untuk mengimplementasi rancangan *website* yang sudah dibuat, maka perlu dibuatkan perangkat lunak dengan melalui *coding.* Pengembangan model dilakukan menggunakan *software Visual Studio Code* dan *Jupyter Notebook* yang bisa digunakan secara gratis. Proses pengembangan model dan anotasi gambar akan melalui proses *training* menggunakan *jupyter notebook* yang tersedia ketika sudah melakukan instalisasi *software* Anaconda. Selanjutnya proses pemngembangan *website*

menggunakan library *streamlit,* penggunaan *streamlit* akan memudahkan dalam membangun aplikasi menggunakan bahasa pemrograman *python*. Berikut merupakan pengembangan *website* menggunakan *streamlit* terlihat pada Gambar 24.



```
if __name__ == '__main__':

    st.sidebar.header('🚘 Konfigurasi Model')
    # custom class
    assigned_class_id = [0, 1, 2, 3]
    names = ['bus', 'mobil', 'motor', 'truk']

    # Always display the multiselect widget for selecting custom classes
    assigned_class_id = []
    assigned_class = st.sidebar.multiselect('Pilih class spesifik untuk proses deteksi', names)
    for each in assigned_class:
        assigned_class_id.append(names.index(each))

    # Display selected class in the sidebar
    st.sidebar.caption("class terpilih : {}".format(', '.join(assigned_class) if assigned_class else 'None'))

    # st.write(assigned_class_id)
    # setting hyperparameter
    line = st.sidebar.number_input('🛠️ Line position', min_value=0.0, max_value=1.0, value=0.6, step=0.1)
    confidence = st.sidebar.slider('🔧 Confidence', min_value=0.0, max_value=1.0, value=0.5)
    st.sidebar.write("Confidence set :", confidence)
    iou_thres = st.sidebar.slider('🔧 Iou Threshold', min_value=0.0, max_value=1.0, value=0.5)
    st.sidebar.write("Iou set :", iou_thres)

    st.subheader("📊 Video Input")
    st.write("""silahkan mengunggah video dengan ketentuan tidak lebih dari 200MB untuk
             mempercepat proses unggah dan proses deteksi. format video yang didukung yaitu MP4, MOV, AVI.""")
    # upload video
    video_file_buffer = st.file_uploader("Silahkan Upload Video Untuk Memulai Deteksi Objek", type=['mp4', 'mov', 'avi'])
    newpath = r"runs/video_upload"
    if not os.path.exists(newpath): os.makedirs(newpath)
```

Gambar 24. *Streamlit code*

Berbeda dengan HTML membuat sebuah elemen untuk membuat *select box* menggunakan *tag <select> dan <options>* sebagai pilihan item-nya, sedangkan pada *streamlit* untuk membuat suatu elemen seperti *select box* dapat menggunakan perintah *st.selectbox()*.

4.7 Implementasi Model *Python*

Pada tahap implementasi model *python* setiap proses dan pengembangan modelnya peneliti menggunakan *jupyter notebook* untuk menyusun tiap langkah-langkah yang akan dijalankan satu persatu setiap prosesnya. Dan untuk menjalankan YOLO perlu melakukan instalasi *requirements library file* yang diperlukan saat menjalankan program.

BAB V
HASIL DAN PEMBAHASAN

5.1 Hasil

5..1.1 Import Library

Library yang digunakan pada *python* berupa *package* dan *module* yang akan memudahkan dalam pembuatan aplikasi dengan meminimalkan kode yang akan ditulis. Dan kelebihan dari menggunakan *library* dapat menghemat waktu karena dapat digunakan berulang kali atau disebut *reusable.*

Berikut model *library* yang akan digunakan pada penelitian ini:

1. *pytorch* yaitu *library* yang dioptimalkan untuk aplikasi yang membutuhkan akses GPU dan CPU untuk mengembangkan dan melatih *neural network*. *Pytorch* dikembangkan oleh *facebook*.
2. *Matplotlib*, *library* untuk visualisasi data berupa *plot* dan diagram.
3. *Numpy*, *library* yang digunakan untuk memproses *array* terutama *numpy* dapat memproses matriks berukuran besar.
4. *Pandas*, *library* yang digunakan untuk Analisa data, manipulasi data, dan pembersihan data. Pada penelitian ini *library pandas* digunakan untuk proses manipulasi data pada tabel hasil model.
5. *Opencv*, *library* yang digunakan untuk mengolah gambar dan video memiliki kemampuan mengekstrak data untuk digunakan dalam bidang *computer vision*.
6. *Time* dan *datetime*, menyediakan fungsionalitas berkaitan dengan waktu dengan mengakses berbagai fungsi.
7. OS, menyediakan fungsi untuk berinteraksi dengan sistem operasi. *Python* memiliki interaksi pada sistem operasi *windows, linux*, dan *mac*.
8. *Streamlit*, *library* yang digunakan untuk memudahkan pengguna dengan mengubah data *script* program *python* menjadi aplikasi berbasis web yang interaktif.

```
import torch
from matplotlib import pyplot as plt
import numpy as np
import cv2
import os
os.environ['KMP_DUPLICATE_LIB_OK'] = 'True'
import time
import pandas as pd
✓ 49.1s
```

Gambar 25. *Install pytorch* dan *import library*

Setelah melakukan *install pytorch* dan *import library* dibutuhkan juga *requirements library* yang harus dipasang secara mandiri. Untuk itu lebih dulu dilakukan *clone repository* dari Github dengan cara *"git clone https://github.com/ultralytics/yolov5"* pada *command line* tau bisa dengan cara mengunduh secara langsung pada halaman *github.* Pada Gambar 26 menunjukan setelah *cloning* dan proses *install library* YOLO v5.

```
1 %cd yolov5
2 %pip install -r requirements.txt
```

Gambar 26. Proses *install library yolo*

5..1.2 *Dataset Information File*

Dataset information file dibuat untuk mengetahui direktori dan jumlah *class* pada proses *training* sesuai dengan nama *class* dengan proses *labelling*. Untuk membuat *dataset file* dibutuhkan *file* baru dengan nama *dataset.yaml.*

```yaml
path: Kendaraan-21
train: train/images
val: valid/images
test: test/images

names:
  0: bus
  1: mobil
  2: motor
  3: truk
```

Gambar 27. Konfigurasi *custom dataset*

Pada Gambar 29 merupakan direktori *library* yolov5 yang berisi alamat direktori *dataset, train, val,* dan *class name* yang digunakan.

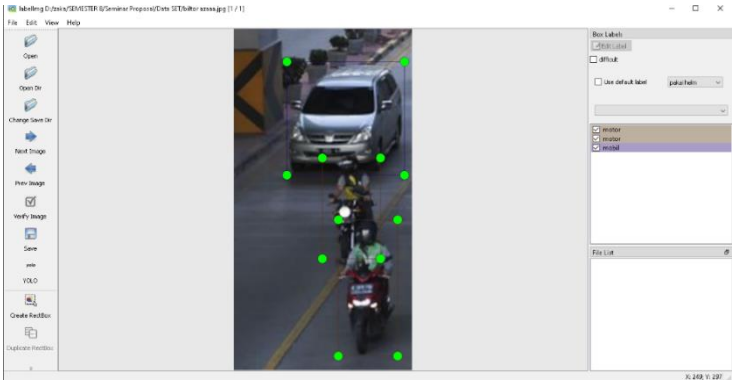
5..1.3 *Labelling Image*

Pada proses *labelling* menggunakan platform *roboflow,* satu per satu gambar akan diberi label dengan cara *draw box* pada objek yang dituju sesuai *class*. *Class* akan dibagi menjadi 4 yaitu, *class* bus, *class* mobil, *class* motor dan *class* truk. Proses *labelling* ditunjukan pada Gambar 28.

```
PS C:\Users\ZAKA DARMAWAN\Documents\KFlearning\yolov5deepsort_skripsi> git clone https://github.com/heartexlabs/labelImg.git
Cloning into 'labelImg'...
remote: Enumerating objects: 2097, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 2097 (delta 0), reused 4 (delta 0), pack-reused 2090
Receiving objects: 100% (2097/2097), 237.14 MiB | 4.07 MiB/s, done.
Resolving deltas: 100% (1245/1245), done.
```

Gambar 28. Proses *install labelling*

Untuk menjalankan program labelimg bisa dilakukan dengan cara menjalankan *python labelimg.py*, untuk proses pelabelan ditunjukan pada Gambar 29.



Gambar 29. Proses *labelling* pada gambar

Proses dari pelabelan objek pada gambar akan disesuaikan dengan *class*nya yang dibagi menjadi 4 yaitu motor, mobil, bus dan truk. Pengambilan gambar di atas JPO Botani dengan jarak 50 meter.

5..1.4 *Preprocessing Data*

Tujuan *preprocessing* untuk membuat citra digital agar sesuai dengan kebutuhan ekstraksi fiturnya (Maftukhah et al., 2023). Pada penelitian ini proses *preprocessing data* dilakukan *data cleaning, data integration.*

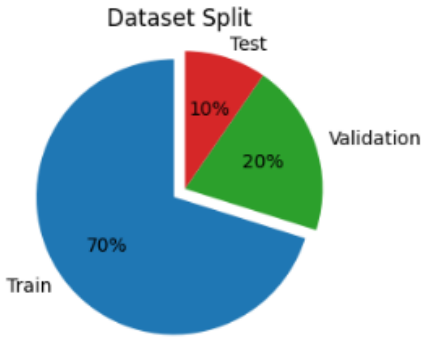
5.1.4.1 Data Cleansing

Pada tahap proses *data cleansing* data akan diseleksi ulang jika tidak memiliki label atau data yang kurang relevan, data yang digunakan yaitu data *training* sebesar 70%, data *validation* 20% dan data *test* 10%. Pada Gambar 30 merupakan pengecekan pembagian *dataset.*

```
%matplotlib inline
label= ['Train','Validation','Test']
colors=['tab:blue', 'tab:green', 'tab:red']
myexplode = [0.1, 0, 0]
fig1, ax1 = plt.subplots(figsize=(3, 3))
data = [int(len(file_train)),int(len(file_valid)),int(len(file_test))]
# data = [int(len(label_train)),int(len(label_valid))]
plt.pie(data, labels=label, autopct='%1.0f%%', startangle=90, colors=colors,explode = myexplode)
plt.title('Dataset Split')
ax1.axis('equal')
plt.show()
```

Gambar 30. Hasil pembagian *dataset*

Pada gambar 32 merupakan kode untuk menampilkan pembagian *dataset* menggunakan chart *pie,* yang ditunjukan pada Gambar 31.



Gambar 31. Hasil *output* pembagian *dataset*

Hasil dari *output* pembagian *dataset* 70% *dataset training, dataset 20% validation* dan *dataset 10% test*

5.1.4.2 Data Integration

Proses data *integration* merupakan pengecekan kembali data yang akan digunakan pada saat proses *training,* menggunakan gambar serta label anotasi gambar untuk proses *training.* Pengecekan kembali gambar dan label akan memiliki ekstensi *file* yang tepat yang dapat dilihat pada Gambar 32.

```
lenTrainImg = os.listdir('/content/yolov5/Kendaraan-21/train/images')
lenValidImg = os.listdir('/content/yolov5/Kendaraan-21/valid/images')
lenTestImg = os.listdir('/content/yolov5/Kendaraan-21/test/images')
lenTrainLabel = os.listdir('/content/yolov5/Kendaraan-21/train/labels')
lenValidLabel = os.listdir('/content/yolov5/Kendaraan-21/valid/labels')
lenTestLabel = os.listdir('/content/yolov5/Kendaraan-21/test/labels')
```

Gambar 32. Proses data integration

Pada Gambar 33 output yang ingin dikeluarkan yaitu jumlah gambar dan jumlah label gambar sesuai dengan total gambar dan label file anotasi.

```
jumlah train sample gambar : 1614
jumlah validation sample gambar : 466
jumlah test sample gambar : 221
total gambar : 2301

jumlah train label  : 1614
jumlah val label : 466
jumlah test label : 221
total label gambar : 2301
```

Gambar 33. Hasil proses *integration*

Pada Gambar 35 menghasilkan *output* jumlah gambar dan jumlah label pada direktori *train* 70% dengan data sebanyak 1614, pada direktori *val* 20% dengan data sebanyak 466 dan pada direktori *test* 10% dengan data sebanyak 221, dan jumlah label atau file anotasi dan jumlah label pada direktori *train* 70% dengan data sebanyak 1614, pada direktori *val* 20% dengan data sebanyak 466 dan pada direktori *test* 10% dengan data sebanyak 221

5..1.5 Proses Training

Pada tahap proses *training* yang akan dilakukan adalah eksekusi program terkait dengan jumlah *class,* keberadaan *path dataset* pada *file dataset yaml* serta konfigurasi *epoch¸batch,* dan *optimizer.* Berikut merupakan *proses training* dilihat pada Gambar 34.

```
#train command object detection using dataset regular yolov5s
!cd yolov5 && python train.py --img 640 --batch-size 32 --epochs 150 --data datasett.yaml --weights yolov5s.pt
--hyp hyp.scratch-low.yaml --workers 2 --optimizer SGD
```

Gambar 34. Proses *training model*

Proses *training* pada Gambar 36 merupakan eksekusi program di *train.py* papa *repository library* yolo v5, dengan ukuran 640 *pixel, epoch* 150, 32 *batch size* dan lokasi *dataset* ada pada *file dataset yaml*. *Optimizer* yang digunakan merupakan SGD. Untuk proses *training* bisa dilihat pada *training result* dan setiap sesi *training* terbentuk pada folder *exp* dengan isi hasil proses *training* dengan menunjukan plot dan *preview data training* dan *validasi.* Untuk mencantumkan hasil proses *training model* dengan format *.csv,* untuk visualisasi *tensorboard* dengan format dan untuk hasil *weight* model menggunakan format *.pt.*

5..1.6 Proses Evaluation

Pada tahap proses *Evaluatin* proses pengenbangan model dengan melihat hasil proses model lagi yang telah dilakukan *training.* Pada hasil *output* data *training* berbentuk sekumpulan angka secara sistematis pada proses dengan sesuai jumlah *epoch* yang disusun pada hasil *training result.* Pada penelitian ini jumlah *epoch* berjumlah 150 sesuai pada program yang akan dijalankan pada proses *training.* Pada tahapan ini dikhususkan untuk evaluasi untuk mengontrol kualitas dari suatu model. Proses evaluasi model menggunakan data grafik untuk melakukan pengecekan ulang pada *output* yang suatu kolomnnya ada datanya ada atau tidak pada *training result* Berikut merupakan proses visualisasi *training result* pada Gambar 35.

```
train_result = pd.read_csv('./yolov5/runs/train/exp/results.csv')
pd.set_option('display.max_rows', None)
train_result.head(10)
# train_result
```

Gambar 35. Proses hasil *training model*

Kode berikut untuk mengecek *output training resu*lt dan untuk memeriksa kolom terkait datanya ada atau tidak pada Gambar 36.

```
#check data
train_result.isnull().sum()
```

Gambar 36. Proses pengecekan nilai kosong

Untuk melakukan visualisasi *training result plot* proses akan menggunakan *library matlotlib* yang telah di *install*. *Plot* akan membandingkan hasil *confidence* data *training* dengan data validasi yang ada untuk mengamati model layak atau tidak jika digunakan pada data yang belum pernah ada. Pada Gambar 37 menunjukan kode *visualisasi plot box loss* dan *obj loss*.

```
%matplotlib inline
figure, axis3 = plt.subplots(2, 1)
figure.tight_layout(pad=3.0)

# plot 1
epochs= train_result['               epoch'].values
train_boxloss = train_result['     train/box_loss'].values
val_boxloss = train_result['       val/box_loss'].values
test_boxloss = train_result['     test/box_loss'].values
axis3[0].set_title('box loss', fontsize=12)
axis3[0].plot(epochs, train_boxloss, val_boxloss)

# plot 2
epochs= train_result['               epoch'].values
train_objloss = train_result['     train/obj_loss'].values
val_objloss = train_result['       val/obj_loss'].values
test_objloss = train_result['     test/box_loss'].values
axis3[1].set_title('obj loss', fontsize=12)
axis3[1].plot(epochs, train_objloss, val_objloss)
```

Gambar 37. Kode visualisasi *box loss* dan *obj loss*

Kemudian kode dibawah ini untuk visualisasi *metrics precision* dan *plot recall* pada Gambar 38.

```
%matplotlib inline
figure, axis4 = plt.subplots(2, 1)
figure.tight_layout(pad=3.0)

# plot 1
epochs= train_result['               epoch'].values
recall = train_result['      metrics/recall'].values
axis4[0].set_title('Recall', fontsize=12)
axis4[0].plot(epochs, recall)

# plot 2
epochs= train_result['               epoch'].values
precision = train_result['   metrics/precision'].values
axis4[1].set_title('Precision', fontsize=12)
axis4[1].plot(epochs, precision)
```

Gambar 38. Kode visualisasi *metrics* dan *precision*

5..1.7 Implementasi dan pengujian model

Pada tahap implementasi model dilakukan 2 cara yaitu, memanfaatkan *framework streamlit* untuk *web* dan menggunakan *python* untuk model *output.* Implementasi *framework streamlit* dilakukan dengan cara *pip install streamlit,* sedangkan untuk pengujian model secara langsung menggunakan *python* akan membutuhkan *library* yolo v5 yang telah di*install.*

5.1.7.1 Model *python* Yolo v5

Yolov5 terintegrasi ke dalam pustaka pytorch dan memuat model yang sudah ada yang telah melalui proses pelatihan Model yang telah melalui proses training dapat langsung digunakan digunakan tanpa harus memuat model ke dalam Torch hub. Model yang dimuat telah dilokalisasi Model yang dimuat terlokalisasi dan dapat langsung digunakan tanpa harus menggunakan internet untuk mengunduh model yang telah dilatih. Klik untuk mengunduh model yang telah dilatih.

Model yang ada sudah dan telah diproses dapat langsung digunakan tanpa harus membuat model pada *torch hub,* karena yolo v5 sudah terintegrasi dengan *library pytorch.* Model dapat dimuat secara langsung tanpa menggunakan internet karena bersifat *local*. Berikut proses memuat model *local* dapat dilihat pada Gambar 39.

```
model = torch.hub.load('yolov5/', 'custom', path='yolov5/runs/train/exp/weights/last.pt',
                       force_reload=True, source='local')
model
```

Gambar 39. Memuat *custom* model

Hal yang dibutuhkan untuk memuat model *local* yaitu *source path* yang menunjukan bahwa *torch* memuat model secara *local.* Kemudian *output* susunan arsitektur model akan ditampilkan dengan cara *print.* Pada Gambar 40 menunjukan o*utput* arsitektur gambaran model.

```
AutoShape(
  (model): DetectMultiBackend(
    (model): DetectionModel(
      (model): Sequential(
        (0): Conv(
          (conv): Conv2d(3, 32, kernel_size=(6, 6), stride=(2, 2), padding=(2, 2))
          (act): SiLU(inplace=True)
        )
        (1): Conv(
          (conv): Conv2d(32, 64, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
          (act): SiLU(inplace=True)
        )
        (2): C3(
          (cv1): Conv(
            (conv): Conv2d(64, 32, kernel_size=(1, 1), stride=(1, 1))
            (act): SiLU(inplace=True)
          )
          (cv2): Conv(
            (conv): Conv2d(64, 32, kernel_size=(1, 1), stride=(1, 1))
            (act): SiLU(inplace=True)
          )
          (cv3): Conv(
            (conv): Conv2d(64, 64, kernel_size=(1, 1), stride=(1, 1))
            (act): SiLU(inplace=True)
          )
```

Gambar 40. Arsitektur model

Untuk selanjutnya menguji model objek deteksi menggunakan data baru yang telah disiapkan berupa gambar untuk diuji coba, Gambar yang sudah dikumpulkan pada folder *assets.* Untuk format gambar menggunakan *jpg* dan *jpeg* dengan ukuran tidak lebih dari 1mb. Kemudian gambar di*import* menggunakan fungsi dari model *python os,* fungsi ini menyatukan dua atau lebih jalur ke dalam satu integrasi. Atribut pada penelitian ini yang pertama yaitu *assets* untuk jalur *direktori,* yang kedua *path* yaitu direktori *image ,* dan yang ketiga nama file yang dituju yaitu file gambar. Untuk proses deteksi menggunakan fungsi model dengan memasukan parameter data gambar yang dapat dilihat langsung pada Gambar 41.

```
1 #uji coba model menggunakan gambar
2 img = os.path.join('/','content','assets', '7 Dec - Sore 4_00_mp4-0.jpg')
3 print(img)
```

Gambar 41. Uji coba model menggunakan gambar

Pada *output model result* menampilkan ukuran gambar yang digunakan dan jumlah hasil deteksi objek. *Output result* berupa teks berupa informasi deteksi gambar tidak dalam *visual output* gambar dan dalam proses penyajian *output* gambar menggunakan *library matplotlib* dan *numpy* yang visualisasi keluarannya berupa koordinat *bounding box*. Berikut merupakan proses visualisasi deteksi gambar pada Gambar 42.

Gambar 42. Hasil *output* deteksi gambar

Dari hasil *output* gambar dengan hasil 19 objek deteksi dengan *class* mobil dengan jumlah 17, *class* motor jumlahnya 2 dan *class* bus berjumlah 1, dengan hasil yang sesuai dengan *output* teks yang sudah dilakukan dengan *result print().* Untuk mengetahui koordinat posisi *bounding box* sesuai dengan gambar yaitu nilai *xmin, ymin, xmax, ymax, confidence,* dan *class* pada yolov5 ditunjukan pada Gambar 43.

```
#nilai xyxy menuju pada : xmin, ymin, xmax, ymax, confidence, class
results.xyxy
```

Gambar 43. *Plot bounding box* dan *class hasil deteksi*

Bentuk dari hasil *output* deteksi yolo akan berbentuk *array,* yang disajikan seperti tabel agar memudahkan untuk dibaca. Untuk perhitungan jumlah *class* tiap objek ditunjukan pada Gambar 44.

```
if 'motor' in table_results["name"].values:
    count_motor = table_results["name"].value_counts()['motor']
    print('motor: {}'.format(count_motor))
else:
    print('motor: {}'.format('-'))

if 'mobil' in table_results["name"].values:
    count_mobil = table_results["name"].value_counts()['mobil']
    print('mobil: {}'.format(count_mobil))
else:
    print('mobil: {}'.format('-'))

if 'bus' in table_results["name"].values:
    count_bus = table_results["name"].value_counts()['bus']
    print('bus: {}'.format(count_bus))
else:
    print('bus: {}'.format('-'))

if 'truk' in table_results["name"].values:
    count_truk = table_results["name"].value_counts()['truk']
    print('truk: {}'.format(count_truk))
else:
    print('truk: {}'.format('-'))
```

Gambar 44. Perhitungan jumlah *class*

Pada Gambar 46 hasil *output* yang dihasilkan merupakan jumlah kendaraan yang terdeteksi sesuai dengan *class*nya.

5.1.7.2 Model Deepsort reid

Dataset yang digunakan pada penelitian ini yaitu *Market1502*, untuk memuat data *training* dan data *testing* dikemas ke dalam *DataManager* yang bertanggung jawab atas konstruksi pengambilan sampel, metode penambahan data, dan memuat data. *Datamanager* berfungsi sebagai input ke pipline *training* dan *Image DataManager* berguna untuk *dataset* gambar dan *dataset* video. Seperti yang ditunjukan pada Gambar 45.

```
import torchreid
datamanager = torchreid.data.ImageDataManager(
    root="reid-data",
    sources="market1501",
    targets="market1501",
    height=256,
    width=128,
    batch_size_train=32,
    batch_size_test=100,
    transforms=["random_flip", "random_crop"]
)
```

Gambar 45. Input *dataset*

Pada gambar 47 *ImageDataManager* merupakan proses *training* dan *evaluasi* model pada Market1501. *Torchreid* akan mengimplementasikan dua fungsi visualisasi yaitu:

1. Fungsi *visrank*, dapat memvisualisasikan hasil yang dapat memvisualisasikan hasil peringkat dari re-ID CNN dengan menyimpan untuk setiap gambar query ke dalam gambar galeri yang mirip untuk setiap gambar query.
2. Fungsi *visactmap*, yang merupakan singkatan dari visualisasi peta aktivasi. Diberikan sebuah gambar input, peta aktivasi dapat digunakan untuk menganalisis dimana CNN berfokus untuk mengekstraksi fitur pelatihan model yang akan dilakukan yaitu berupa model OSNet (Zhou & Xiang, 2019)

Model pengenalan digunakan secara luas sebagai tumpuan re-ID CNN. Model yang digunakan pada penelitian ini yaitu *OSNet* merupakan model yang sudah ada dan dikembangkan oleh *KaiyangZhou*. Proses training model ditunjukan pada Gambar 46.

```
!python /content/deep-person-reid/scripts/main.py \
--config-file configs/im_osnet_x1_0_softmax_256x128_amsgrad_cosine.yaml \
--transforms random_flip random_erase \
--root "$PATH_TO_DATA"
```

Gambar 46. *Training* model *OSNet*

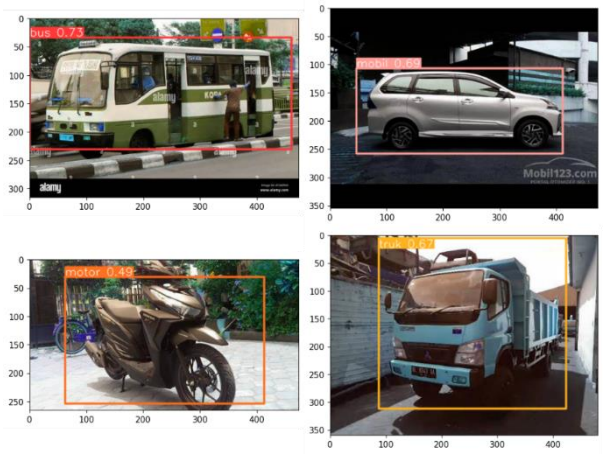
5.1.7.3 Model *Website* Dengan Streamlit

Penggunaan streamlit pada penelitian ini untuk memudahkan proses *running* program dan memudahkan dalam proses pengembangan web untuk *machine learning.* Pengembangan *website* menggunakan *framework streamlit* dengan bahasa pemrograman *python.* Proses instalasi s*treamlit* dengan melakukan *pip install streamlit* fitur yang disajikan yaitu konfigurasi model dengan deteksi video, *custom class, confidence, line position,* penyajian jumlah hasil deteksi sesuai *class,* dan FPS berdasarkan dengan UI yang telah dibuat dengan input video. Pengembangan *sidebar* pada *website* dapat dilihat pada lampiran 8.

5.1.7.4 Pengujian Model Python Yolo v5

Untuk pengujian model program deteksi menggunakan gambar pada *assets* yang telah dikumpulkan sebanyak 4 gambar, dengan hasil *output* dapat dilihat

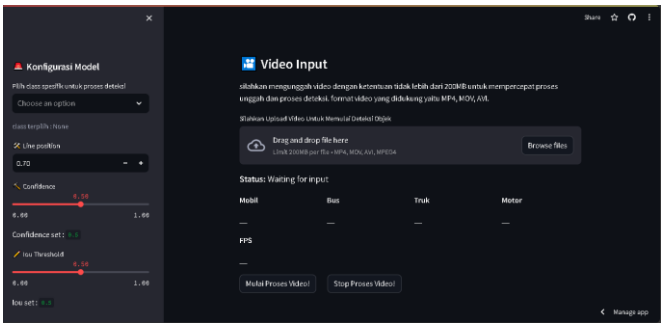
langsung pada Gambar 47. Hasil yang didapat *output* 4 gambar, pada *class* bus memiliki nilai *confidence* diatas 70%, mobil memiliki nilai *confidence* diatas 65%, motor memiliki nilai *confidence* diatas 45%, dan truk memiliki nilai *confidence* diatas 65%. Dan untuk percobaan lainya menggunakan video *local* berdurasi 19 detik dengan hasil yang didapat model mampu mendeteksi objek dengan akurat.



Gambar 47. pengujian model menggunakan input gambar

5.1.7.5 Pengujian Model *Website* Dengan *Streamlit*

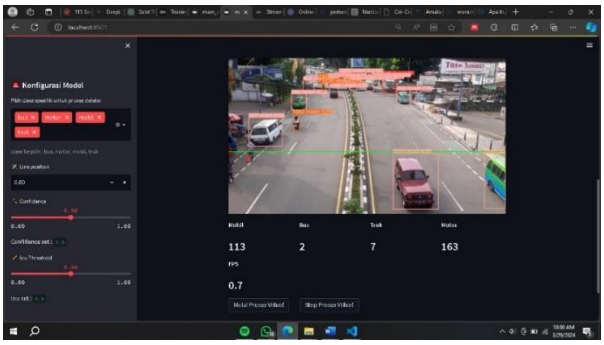
Pada pengujian model *website* dilakukan menggunakan *local server* dengan mengeksekusi program melalui terminal, dan diarahkan ke *link localhost* yang mengarah ke *project* yang sedang dijalankan secara *local* ditunjukan pada Gambar 48.



Gambar 48. *Interface* halaman *input* video pada *website*

Untuk ukuran video yang dapat diunggah tidak melebihi 200MB dengan jenis file MP4, MOV,– dan AVI dan video yang diunggah akan tersimpan pada *local server*. Untuk memulai menjalankan proses deteksi dengan menekan tombol start, dengan keluaran

akan menampilkan *class* dan nilai *confidence*. Pada Gambar 41 menunjukan hasil *output* deteksi.



Gambar 49. *Output* hasil input video

5.1.8 Deployment

Pada proses *deployment website*, menggunakan *Github* dan *Streamlit Cloud* untuk melakukan *hosting.* Langkah pertama proses *deployment* membuat *repository* di *Github* sebagai tempat *file project* yang akan digunakan. Pembuatan *repository* dilakukan dengan memasukan nama *repository,* deskripsi, dan menyeting *repository* untuk akses *public* atau *private* setelah itu *upload file repository* menggunakan *Git* melalui *CLI.* Kemudian sebelum meng*upload* harus membuat *file requirements* yang berisi nama *library* yang digunakan, *file requirements* berfungsi untuk *streamlit cloud* membaca kebutuhan *library project.* Berikut isi *file requirements* ditunjukan pada lampiran 9 &10.

Langkah selanjutnya melakukan konfigurasi *streamlit cloud* untuk *deploy website* sebagai akses ke internet dengan melakukan *login* pada *streamlit.io* kemudian pilih *new app.* Pada halaman *deploy an app* ada beberapa *input* yang berhubungan dengan lokasi *repository, branch* dan lokasi *file* pada *streamlit.* Apabila sudah menginput semua dengan benar langkah selanjutnya tekan *deploy*, yang akan membutuhkan waktu selama kurang 2 menit. Ditunjukan pada lampiran 11.

5.2 Pembahasan

5.2.1 Ringkasan Model

Dari hasil proses *training* data yang telah dikumpulkan akan mengeluarkan *output* model dengan *format*.pt. yang bisa dilihat pada *library torch.*hasil berupa teks parameter yang digunakan, informasi *hardware* dan *software* yang digunakan pada proses *training* dan informasi lapisan konvolusi. Pada model ini banyak menggunakan susunan fungsi *sequential*() yang memiliki susunan konfigurasi fungsi *Conv*() yang mengarah pada konvolusi dan penggunaan *activation function* yang menggunakan model , *Maxpool2d*(), *updample*(), dan *Detect*(). Dibawah ini merupakan informasi model *summary 1* yang ditunjukan pada Tabel 5.

Tabel 5. *Model Summary 1*

	Model Sequential 1							
No	Type	Layer	*Input (W, H, C)*	N	S	P	Act	*params*
1	Conv	Conv2d	630,640, 3	6,6	2,2	2,2	Silu	3488
2	Conv	Conv2d	320, 320, 32	3,3	2,2	1	Silu	18496

3	C3	Conv2d	160, 160, 64	1,1	1,1	-	Silu	2080
		Conv2d	160, 160, 64	1,1	1,1	-	Silu	2080
		Conv2d	160, 160, 64	1,1	1,1	-	Silu	36928
Total *Sequential* 1*Function Params*								63072

Perhitungan *params* dapat dilakukan menggunakan persamaan 18 seperti berikut:

$$params = output_{channel} * input_{channel} * size_{channel} \quad (18)$$

Dengan menggunakan persamaan 18 nilai jumlah parameter *model sequential 1* menghasilkan

$$Parameter\ Conv\ 1 = 32 * (3 * (6 * 6) + 1) = 3488$$
$$Parameter\ Conv\ 2 = 64 * (32 * (3 * 3) + 1) = 18496$$
$$Parameter Conv C3 1 = 32 * (64 * (1 * 1) + 1) = 2080$$
$$Parameter Conv C3 2 = 32 * (64 * (1 * 1) + 1) = 2080$$
$$Parameter\ Conv\ C3\ 3 = 64 * (64 * (3 * 3) + 1) = 36928$$

Sampai ke lapisan selanjutnya perhitungan persamaan dilakukan ke lapisan terakhir dan akan dihitung jumlah total keseluruhannya. Pada proses perhitungan *output* dimensi konvolutional pada Tabel 6 jika yolov5 menggunakan *input* 640 x 640 x 3 yang dilakukan pada perhitungan ini ditunjukan pada lampiran 11.

Nilai persamaan yang ada pada jumlah parameter didapatkan dari *output* hasil program pada bagian *output* conv2d *(input_channel, output channel, kernel_size, stride,* dan *padding).* Perhitungan ini sama saja dengan fungsi *sequential* lainnya saat dijalankan secara bersamaan. Kemudian terdapat perhitungan lainya yaitu *max pooling* yang ada pada fungsi MaxPool2d() berguna untuk mengurangi resolusi gambar dan mempertahankan informasi pada gambar dan lapisan pada *max pooling* berguna untuk menangkap satu *pixel* yang nilainya paling tinggi pada setiap area *pixel* digambar yang akan menghasilkan gambar yang baru. Dan pada akhir *output* model ada fungsi *module list* yaitu *output* daftar lapisan yang dilakukan berulang dan disimpan informasinya. Pada fungsi *module list* ada tiga lapisan yang dapat dilihat pada Tabel 6.

Tabel 6. *Model Summary 2*

Module list							
No	*Type*	*Layer*	*Input (W, H, C)*	*N*	*S*	*P*	*Params*
1	Conv	Conv2d	640, 640, 3	6,6	2,2	-	1.175.295
2	Conv	Conv2d	318, 318, 128	3,3	2,2	-	587.775
3	Conv	Conv2d	158, 158, 256	1,1	1,1	-	130.815
Total Params							1.893.885

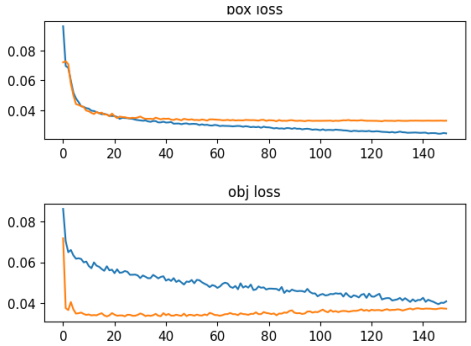
Dari Tabel 7 dapat disimpulkan jumlah layer dan layer yang dihasilkan akan besar jika ukuran inputnya juga besar. Pada Persamaan 18 fungsi *module list* menghasilkan nilai jumlah parameter sebagai berikut:

$$Parameter\ Conv\ 1 = 255 * \{128 * \{6 * 6\} + 1\} = 1.175.295$$
$$Parameter\ Conv\ 2 = 255 * \{256 * \{3 * 3\} + 1\} = 587.775$$
$$Parameter\ Conv\ 2 = 255 * \{512 * \{1 * 1\} + 1\} = 130.815$$

Untuk proses perhitungan *output* dimensi konvolusi menggunakan *input* 640 x 640 x 3 pada tabel 7 dapat ditunjukan pada lampiran 12.

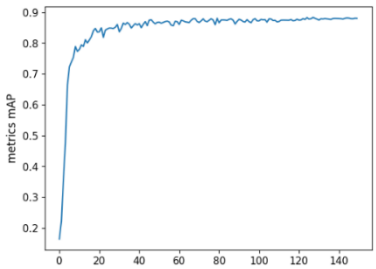
5.2.2 Evaluasi *Training Model*

Proses *training* model menghasilkan nilai *confusion matrix* selama proses *training* berlangsung sebagai *output* dari hasil deteksi data *training* dan data *validasi.* Dari hasil proses *training* juga menghasilkan akumulasi dan akurasi *box loss* dan *obj loss* bisa dilihat pada Gambar 50.



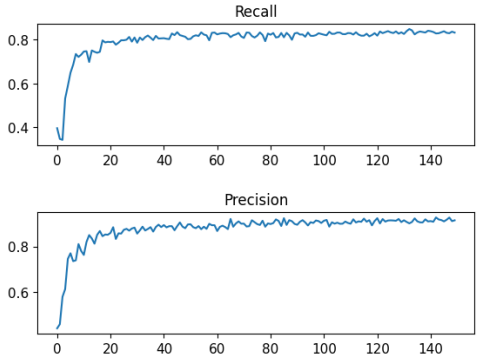
Gambar 50. *box loss & obj loss training model dataset*

Pada Gambar 52 plot 1 *box loss* garis berwarna biru menunjukan hasil proses *training* pada *training model dataset* dan garis *orange* pada plot 1 *box loss* menunjukan hasil *test* menggunakan data validasi. Sedangkan pada plot 2 yaitu *obj loss*, sama dengan plot 1 garis biru untuk hasil proses *training* menggunakan data *training* dan garis *orange* hasil *test* menggunakan data validasi. Plot 1 *box loss* merupakan seberapa baik algoritma dapat menemukan pusat objek dan seberapa baik *bounding box* bisa memprediksi dan menutupi objek, sedangkan plot 2 *obj loss* merupakan pengukuran *error* dengan memperhatikan ukuran kemungkinan suatu objek ada pada wilayah yang dikehendaki. Pada penelitian proses *training* menggunakan *epoch* sebesar 150 dengan hasil *training box loss* dan *validation box loss* sebesar 0.09 dan 0.07, sedangkan obj loss *training dan validation* sebesar 0.08 dan 0.055. Hasil dari proses *training* juga menghasilkan nilai *mAP (mean average precision)* merupakan besaran tingkat akurasi model yang digunakan untuk mendeteksi objek dan posisinya objek. Nilai *mAP* juga digunakan sebagai alat ukur akurasi model serta performa pada model, untuk nilai *mAP* yang didapatkan sebesar 0.85 atau 85% bisa dilihat pada Gambar 51.



Gambar 51. *mAP training model*

Plot lainnya yang dihasilkan yaitu *metrics recall* dan *precision, metrics* berfungsi untuk menghitung *harmonic man F1* dengan range 0 sampai 1. Untuk *F1 score* ini berdampak pada klasifikasi menguji model seberapa akurat dan seberapa kuat dalam mendeteksi objek. Jika nilai hasil presisi tinggi yang berpotensi menimbulkan hasil *false positive,* apabila nilai *recall* yang didapatkan jauh lebih rendah maka akan memberikan hasil yang lebih akurat. Jika nilai *recall* yang dihasilkan tinggi dan presisinya rendah maka model dapat melewatkan momentum objek yang seharusnya dapat terdeteksi namun mengurangi hasil *false positive.* Hasil *metrics precision* sebesar 0.8 seperti yang ditunjukan pada Gambar 52.



Gambar 52. *recall & precision training model dataset*

Nilai *F1 score* merupakan analogi dari rata-rata *precision* dan *recall* yang dibandingkan. Berikut ini hasil *metrics precision* dan *metrics recall* dari proses *training* model dengan *epoch* sebesar 150 yang ditunjukan pada Tabel 7.

Tabel 7. *Precision, Recall Training Output Model*

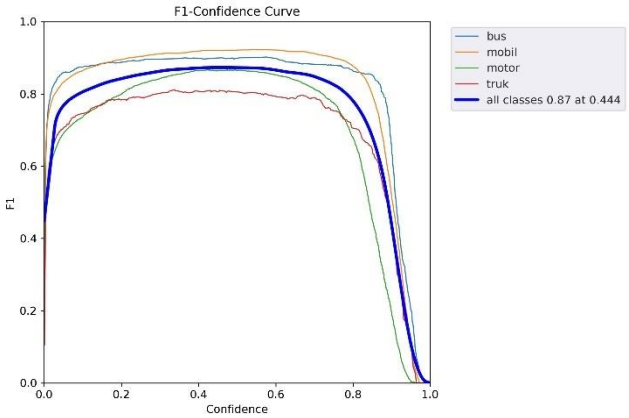
epoch	*metrics/precision*	*metrics/recall*
0	0.37054	0.36352
1	0.39289	0.4385
2	0.49019	0.58539
3	0.74993	0.64659
4	0.72398	0.69874
5	0.75729	0.74831
. . .	. . .	. . .
. . .	. . .	. . .
145	0.92566	0.85979
146	0.91843	0.86041
147	0.91633	0.8662
148	0.91187	0.8705
149	0.92051	0.85876

Keterangan:
Hasil proses *metrics precision* 0.92051 * 100 = 92,051% dibulatkan 92%

Hasil proses *metric recall* 0.85876 * 100 = 85,876% dibulatkan 86%

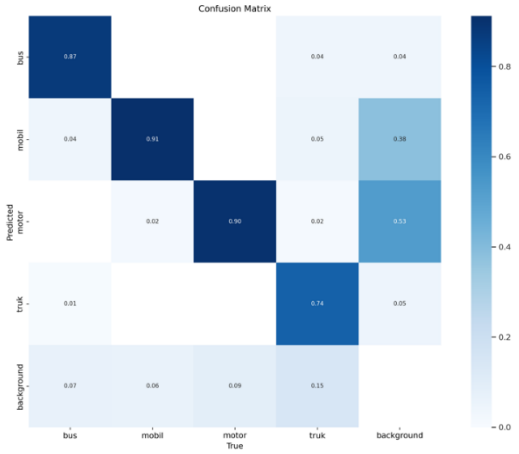
$$F1\ score = 2 * \frac{recall * precision}{recall + precision} = * 2\ \frac{92\% * 86\%}{92\% + 86\%} = 89\%$$

Hasil dari *F1 confidence* pada *training model* ditunjukan pada Gambar 53.



Gambar 53. *F1 confidence score training model*

Untuk plot lainnya yaitu *confusion matrix* yang memberikan informasi hasil dari proses klasifikasi objek yang dibuat oleh model dengan membandingkan hasil dari klasifikasi yang seharusnya. Hasil dari proses *training confusion metrics* pada model mendapatkan nilai *True Positive (TP)* 0,90 pada *class* bus, 0,92 pada *class* mobil, 0,90 pada *class* motor dan 0,84 pada *class* truk. Seperti yang ditunjukan pada Gambar 54.



Gambar 54. *confusion matrix training model dataset*

Hasil *confusion matrix* Gambar 60, didapatkan model mampu meraih 0,90 TP dengan *class* bus, 0,92 TN dengan *class* mobil, 0,90 TN dengan *class* motor dan 0,84 TN dengan *class* truk.

5.2.3 *User Acceptance Test Model (UAT)*

Untuk mengetahui model sudah berjalan dengan baik dan sudah sesuai dengan yang diinginkan dilakukan proses UAT yaitu pengujian penerimaan pengguna. Proses UAT dilakukan untuk meninjau fungsi model pada *website* yang sudah dikembangkan, dan untuk model yang digunakan yaitu hasil dari *training dataset* mandiri yang

memiliki format.*pt.* proses UAT pada model dilakukan menggunakan video sebanyak 14 dengan output yang akan memberikan nilai *accuracy, recall,* dan *precision*. Dan untuk pengujian UAT pada *website* dilakukan dengan test pada fungsi-fungsi pada fitur yang ada pada *website.* Pengujian UAT model menghasilkan jumlah kendaraan yang terdeteksi sebanyak 4640 dengan hasil gambar *true positive (TP)* sebanyak 4633, *true negative (TN)* sebanyak 6, *false positive (FP)* sebanyak 1*, dan false negative (FN)* sebanyak 0. Dari hasil *UAT* kendaraan dapat terdeteksi pada jarak 100 meter lebih dari itu objek kendaraan tidak dapat terdeteksi.

$$accuracy = \frac{TP+TN}{TP+TN+FP+FN}\ \frac{4636}{4640} = 0{,}99$$

$$recall = \frac{TP}{TP+FN}\ \frac{4633}{4633+0} = 1$$

$$precision = \frac{TP}{TP+FP}\ \frac{4633}{4633+1} = 0{,}99$$

Hasil dari uji *UAT* model test menggunakan *input* sample video sebanyak 14 pada waktu pagi dan sore hari di atas JPO Botani dengan jarak 50 meter yang diproses melalui website dengan model yang sudah diproses menggunakan *dataset* mandiri mendapatkan akurasi sebesar 99%, *precision* dengan nilai 99%.

BAB VI
KESIMPULAN DAN SARAN

6.1 Kesimpulan

Hasil dari penelitian yang telah dilakukan mengenai pengembangam model objek deketsi dan *tracking* kendaraan sudah berhasil dibuat dengan metode *You Only Look Once* untuk mendeteksi kendaraan dan *Deepsort* untuk men*tracking* kendaraan yang masuk ke kota Bogor dengan *class* terbagi menjadi 4. Pengembangan yang dilakukan mulai dari model sampai pengembangan *website* yang digunakan sebagai media informasi yang didapat dari pengembangan model menggunakan *dataset* yang dikumpulkan secara mandiri dengan pengambilan gambar dan video di JPO Botani Bogor. Bahasa pemrograman *python* digunakan untuk mengintegrasikan model ke dalam web dengan *framework* pendukung yang digunakan yaitu *streamlit.*

Untuk menguji coba *website* sudah sesuai dengan fungsinya, peneliti melakukan *User acceptance testing (UAT)* dengan 7 *test case* yang dilakukan untuk membuktikan *website* yang dikembangkan sudah berfungsi dan berjalan dengan baik. Pengujian dilakukan setelah *website* berhasil di*hosting* dengan *framewework streamlit* dan *github* dengan *url* https://yolov5deepsortapp.streamlit.app. Hasil *running* menggunakan *CPU* mendapatkan *fps* sebesar 1 dengan resolusi video *output* 480. Pada proses *training* dengan menggunakan *dataset* mandiri yang berjumlah total 2276 gambar yang terbagi menjadi 3 bagian yaitu *dataset train* 70% dengan data sebanyak 1614, *dataset val* 20% dengan data sebanyak 466 dan *dataset test* 10 % dengan data sebanyak 221. Dari hasil *UAT* kendaraan dapat terdeteksi pada jarak 100 meter lebih dari itu objek kendaraan tidak dapat terdeteksi.

Hasil *output* mendapatkan nilai *mean average precision (mAP)* mendapat nilai hingga 85% dengan tingkat *error box loss* dan *object loss* yang minim yaitu 0.09% dan 0.08%. Selain itu model mendapatkan *F1 score* sebesar 89%.

Hasil dari uji *UAT* model test menggunakan *input* sample video sebanyak 14 pada waktu pagi dan sore hari di atas JPO Botani dengan jarak 50 meter yang diproses melalui website dengan model yang sudah diproses menggunakan *dataset* mandiri mendeteksi jumlah kendaraan sebanyak 4640 dengan hasil *true positive (TP)* sebanyak 4633, *true negative (TN)* sebanyak 6, *false positive (FP)* sebanyak 1*, false negative (FN)* sebanyak dengan 1 dan menghasilkan akurasi sebesar 99%, *precision* dengan nilai 99%.

6.2 Saran

Saran yang perlu dilakukan pada penelitian ini adalah:

1. Penggunaan YOLO dengan versi yang terbaru.
   Pada penelitan ini versi YOLO yang digunakan yolov5, pada versi terbaru dari yolo adanya pembaharuan dalam pelabelan id pada objek.
2. Pengumpulan *dataset* yang kurang bervariasi
   Pada penelitian ini *dataset* yang dikumpulkan masi kurang bervariasi dalam hal cuaca dan pencahayaan. Karena faktor cuaca dapat mempengaruhi pendeteksian pada detail objek dan menyebabkan kesalahan dalam proses perhitungan objek sesuai dengan *class.*
3. Perhitungan secara 2 arah
   Perhitungan kendaraan bisa 2 arah tidak hanya yang masuk ke kota Bogor saja.

DAFTAR PUSTAKA

Adi, L., Agung, I. P., & Suar, K. (2020). *Sistem Hitung Kendaraan Berdasarkan Jenis Menggunakan Metode Background Subtraction*. *1*(2).

Alfarizi, M. R. sirfatullah, Al-farish, M. Z., Taufiqurrahman, M., Ardiansah, G., & Elgar, M. (2023). Penggunaan Python Sebagai Bahasa Pemrograman Untuk Machine Learning dan Deep Learning. *Karimah Tauhid*, *2*(1), 1–6. https://ojs.unida.ac.id/karimahtauhid/article/view/7518/3480

Amwin, A. (2021). Deteksi Dan Klasifikasi Kendaraan Berbasis Algoritma You Only Look Once (YOLO). *Universitas Islam Indonesia*. https://dspace.uii.ac.id/handle/123456789/34154

Aningtiyas, P., Sumin, A., & wirawan, S. (2020). Pembuatan Aplikasi Deteksi Objek Menggunakan TensorFlow Object Detection API dengan Memanfaatkan SSD MobileNet V2 Sebagai Model Pra - Terlatih. *Jurnal Ilmiah Komputasi*, *19*(3), 421–430. https://doi.org/10.32409/jikstik.19.3.68

Benih, P., Sawit, K., & Utama, P. (2022). *IMPLEMENTASI ALGORITMA YOLOv7 PADA SISTEM TIME SKRIPSI NASRULLAH M . HARIS MAKASSAR DESEMBER 2022*.

Bin Zuraimi, M. A., & Kamaru Zaman, F. H. (2021). Vehicle detection and tracking using YOLO and DeepSORT. *ISCAIE 2021 - IEEE 11th Symposium on Computer Applications and Industrial Electronics*, 23–29. https://doi.org/10.1109/ISCAIE51753.2021.9431784

Dio, M., Pratama, R., Priyatna, B., Shofiah, S., & Lia, A. (2022). *Deteksi Objek Kecelakaan Pada Kendaraan Roda Empat Menggunakan Algoritma YOLOv5 Car Vehicle Accident Object Detection Using YOLOv5 Algorithm*. *12*(2), 15–24.

Farizkhar. (2022). *Farizkhar, 2022 ANALISIS SPASIAL KETERSEDIAAN RUANG TERBUKA HIJAU PUBLIK DI KOTA BOGOR DENGAN METODE OBJECT-BASED IMAGE ANALYSIS (OBIA) PADA CITRA SPOT-6 Universitas Pendidikan Indonesia | repository.upi.edu | perpustakaan.upi.edu*. 1–15.

Handayanto, R. T., & Herlawati, H. (2020). Prediksi Kelas Jamak dengan Deep Learning Berbasis Graphics Processing Units. *Jurnal Kajian Ilmiah*, *20*(1), 67–76. https://doi.org/10.31599/jki.v20i1.71

Hasibuan, N. N., Zarlis, M., & Efendi, S. (2021). Detection and tracking different type of cars with YOLO model combination and deep sort algorithm based on computer vision of traffic controlling. *Jurnal Dan Penelitian Teknik Informatika*, *6*(1), 210–220. https://doi.org/10.33395/sinkron.v6i1.11231

Leriansyah, M., & Kurniawardhani, A. (2020). Klasifikasi dan Perhitungan Kendaraan untuk Mengetahui Arus Kepadatan Lalu Lintas Menggunakan Metode YOLO. *AUTOMATA*, *1*(1). https://journal.uii.ac.id/AUTOMATA/article/view/13970

Maftukhah, A., Fadlil, A., & Sunardi, S. (2023). Segmentasi Citra Kupu-Kupu Menggunakan Metode Multilevel Thresholding. *J-SAKTI (Jurnal Sains Komputer Dan Informatika)*, *7*(2), 545–554. https://ejurnal.tunasbangsa.ac.id/index.php/jsakti/article/view/665

Nafi'Ah, H., Rahmawati, P., Hikmaturokhman, A., & Larasati, S. (2021). Design of LoRaWAN for Smart Factories in Industrial Estates. *10th IEEE International Conference on Communication, Networks and Satellite, Comnetsat 2021 - Proceedings*, 116–122. https://doi.org/10.1109/COMNETSAT53002.2021.9530791

Nugroho, P. A., Fenriana, I., & Arijanto, R. (2020). Implementasi Deep Learning Menggunakan Convolutional Neural Network (Cnn) Pada Ekspresi Manusia. *Algor*, *2*(1), 12–21.

Parico, A. I. B., & Ahamed, T. (2021). Real time pear fruit detection and counting using yolov4 models and deep sort. *Sensors*, *21*(14), 4803. https://doi.org/10.3390/s21144803

Pratiwi, B. P., Handayani, A. S., & Sarjana, S. (2021). PENGUKURAN KINERJA SISTEM KUALITAS UDARA DENGAN TEKNOLOGI WSN MENGGUNAKAN CONFUSION MATRIX. *Jurnal Informatika Upgris*, *6*(2). https://doi.org/10.26877/jiu.v6i2.6552

Rasywir, E., Sinaga, R., & Pratama, Y. (2020). Analisis dan Implementasi Diagnosis Penyakit Sawit dengan Metode Convolutional Neural Network (CNN). *Paradigma - Jurnal Komputer Dan Informatika*, *22*(2), 117–123. https://doi.org/10.31294/p.v22i2.8907

Santos, A. M., Bastos-Filho, C. J. A., MacIel, A. M. A., & Lima, E. (2020). Counting Vehicle with High-Precision in Brazilian Roads Using YOLOv3 and Deep SORT. *Proceedings - 2020 33rd SIBGRAPI Conference on Graphics, Patterns and Images, SIBGRAPI 2020*, 69–76. https://doi.org/10.1109/SIBGRAPI51738.2020.00018

Sauqi, M. (2022). Deteksi Kendaraan Menggunakan Algoritma You Only Look Once (YOLO) V3. *Universitas Islam Indonesia*, 5–8.

Widi Hastomo, Nur Aini, Adhitio Satyo Bayangkari Karno, & L.M. Rasdi Rere. (2022). Metode Pembelajaran Mesin untuk Memprediksi Emisi Manure Management. *Jurnal Nasional Teknik Elektro Dan Teknologi Informasi*, *11*(2), 131–139. https://doi.org/10.22146/jnteti.v11i2.2586

Wojke, N., Bewley, A., & Paulus, D. (2018). Simple online and realtime tracking with a deep association metric. *Proceedings - International Conference on Image Processing, ICIP*, *2017-Septe*, 3645–3649. https://doi.org/10.1109/ICIP.2017.8296962

Zhou, K., & Xiang, T. (2019). *Torchreid: A Library for Deep Learning Person Re-Identification in Pytorch*. http://arxiv.org/abs/1910.10093

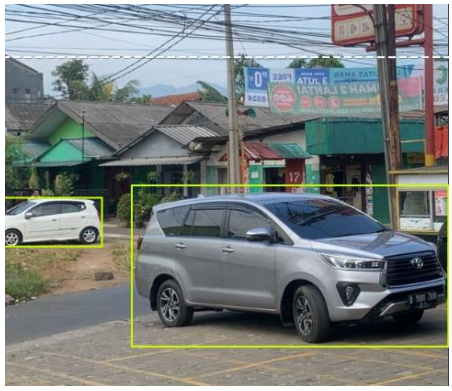
Zou, Z., Chen, K., Shi, Z., Guo, Y., & Ye, J. (2023). Object Detection in 20 Years: A Survey. *Proceedings of the IEEE*. https://doi.org/10.1109/JPROC.2023.3238524

LAMPIRAN

Lampiran 1. Proses image *labeling* menggunakan *roboflow*

Tabel 8. *Pretrained Checkpoints*

model	size (pixels)	map^{val} 50-95	map^{val} 50	speed cpu b1 (ms)	speed v100 b1 (ms)	Speed v100 b32 (ms)	params (m)	flops @640 (b)
YOLOv5n	640	28.0	45.7	45	6.3	0.6	1.9	4.5
YOLOv5s	640	37.4	56.8	98	6.4	0.9	7.2	16.5
YOLOv5m	640	45.4	64.1	224	8.2	1.7	21.2	49.0
YOLOv5l	640	49.0	67.3	430	10.1	2.7	46.5	109.1
YOLOv5x	640	50.7	68.9	766	12.1	4.8	86.7	205.7
YOLOv5n6	1280	36.0	54.4	153	8.1	2.1	3.2	4.6
YOLOv5s6	1280	44.8	63.7	385	8.2	3.6	12.6	16.8
YOLOv5m6	1280	51.3	69.3	887	11.1	6.8	35.7	50.0
YOLOv5l6	1280	53.7	71.3	1784	15.8	10.5	76.8	111.4



Gambar 55. Proses *Image Labelling* Menggunakan *Roboflow*

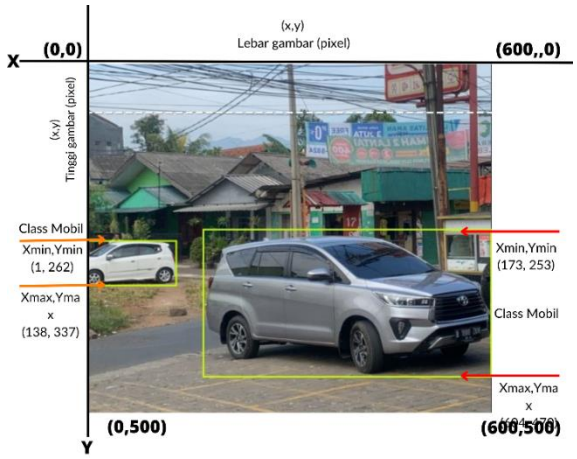
Proses *labelling* data pada dilakukan pada *platform Roboflow* yang memiliki dukungan format YOLO v5 yang berbasis *supervised learning*. Hasil *output file* setelah melakukan *labelling* dengan ekstensi XML yang memuat data informasi dari citra gambar yang sebelumnya sudah diberikan label. Proses pemberian label harus dilakukan setiap citra satu persatu. Dalam satu citra dapat memiliki lebih dari satu objek kelas. Untuk memproses pendeteksian objek, YOLO membutuhkan *dataset* yang telah memiliki label untuk referensi pembelajaran proses pendeteksian objek saat melakukan testing. Berikut contoh isi salah satu dari hasil pelabelan gambar pada *file* memiliki format ekstensi XML dapat dilihat pada Gambar 56

```xml
<annotation>
    <folder>Data SET</folder>
    <filename>kendaraan.jpeg</filename>
    <size>
        <width>604</width>
        <height>526</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>mobil</name>
        <pose>Unspecified</pose>
        <truncated>1</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>173</xmin>
            <ymin>253</ymin>
            <xmax>604</xmax>
            <ymax>470</ymax>
        </bndbox>
    </object>
    <object>
        <name>mobil</name>
        <pose>Unspecified</pose>
        <truncated>1</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>1</xmin>
            <ymin>262</ymin>
            <xmax>138</xmax>
            <ymax>337</ymax>
        </bndbox>
    </object>
</annotation>
```

.

Gambar 56. File anotasi koordinat gambar *roboflow*

Dari isi *file* XML tersebut dapat diketahui bahwa *file* yang diberi anotasi dengan nama *file* kendaraan.jpg yang dapat dilihat pada Gambar 11**,** memiliki ukuran Panjang & lebar 200 x 300 *pixel* dengan 1 *object class* yang dilabeli mobil 1 dengan koordinat *xmin, ymin* = 173, 253 dan *xmax, ymax* = 604, 470. Lalu dengan *class* mobil 2 dengan koordinat *xmin, ymin* = 1, 162 dan *xmax, ymax* = 138, untuk lebih detail lihat pada Gambar 57.



Gambar 57. Ilustrasi koordinat gambar *input*

Tahap-tahap yang dilalui dalam proses deteksi dari *input* hingga *output* berupa hasil deteksi objek yang dilakukan YOLO sebagai berikut :

Lampiran 2. Pengunggahan Citra

Citra dapat berupa video atau foto. Untuk video dapat pengunggahan deteksi citra dapat dilakukan secara *realtime* menggunakan *internet protocol camera.* Contoh citra berupa gambar ditunjukan pada Gambar 58.

Gambar 58. Pengunggahan ctra

Lampiran 3. Proses Komputasi Yolo

Dalam melakukan proses objek deteksi menggunakan YOLO beberapa proses penting dilakukan hingga input dapat dilakukan klasifikasi objek merujuk class yang dimaksud, proses YOLO di antaranya:

A. *Input* citra dilakukan *resize* dalam ukuran 640 x 640 pada *default* YOLO v5 dengan 3 *channel* apabila *input* berjenis RGB. *Input* citra dapat di *resize* pada *library* YOLO v5 dengan nama *file detect.py* dengan mengubah parameter nilai *default argument –img-size*. citra akan di *resize* sebesar 448 x 448 untuk mempercepat dalam proses deteksi. Contoh gambar setelah di *resize* menjadi 448 x 448 dapat dilihat pada Gambar 59.

Gambar 59. Ilustrasi gambar telah di *resize* 448 x 448

B. tiap *frame* citra diproses melalui *convolution network* secara syaraf tunggal (*Single neural network*) hingga menghasilkan *output* dengan bentuk SxS *grid cell*. *Grid cell* pada setiap kotak bertanggung jawab dalam memprediksi objek. Pada penelitian ini jumlah *grid cell* 7x7. Ilustrasi gambar saat dilakukan *grid cell* 7x7 dapat dilihat pada Gambar 19. Terhitung jumlah *grid cell* yang terdapat pada tiap gambar berjumlah 60.

Gambar 60. Gambar yang memiliki 49 *grid cell*

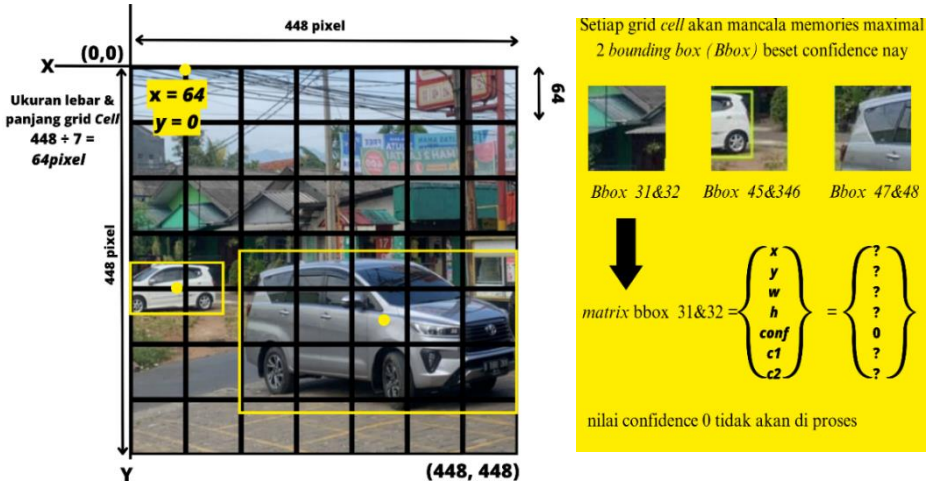
$$Jumlah\ grid\ cell = 7 * 7 = 49$$

jumlah *class* objek adalah 2 (*nC*=2) yaitu mobil *(c1)* & mobil *(c2).* Setiap satu *grid* maksimal dapat mendeteksi 2 objek. Apabila setiap *grid* dapat mendeteksi 2 objek, total prediksi *bounding box* dapat dilakukan dengan Persamaan berikut:

$$total\ prediksi\ bounding\ box = S * S * 2 = 7 * 7 * 2 = 98$$

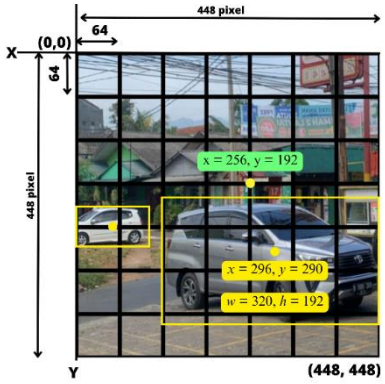
Gambar 61. Ilustrasi 98 jumlah maksimal *bounding box* deteksi *yolo*

Pada Gambar 61 merepresentasikan 98 *bounding box* apabila 49 *grid cell* dengan masing-masing *grid cell* mendeteksi 2 *bounding box.* Yolo membaca setiap *grid cell* dengan membaca lokasi untuk probabilitas *class*. merepresentasikan bahwa *bounding box* berwarna ungu adalah objek "mobil*",* panah putih adalah *bounding box* yang tidak mendeteksi *class*, artinya nilai *confidence* dari pemeriksaan *bounding box* oleh Yolo bernilai 0 yang tidak akan diproses selanjutnya. Ilustrasi m matrik apabila satu grid cell terdapat 2 objek dapat dalam satu matrix pada Gambar 62.



Gambar 62. Ilustrasi matriks pada tiap *grid cell*

Nilai parameter koordinat *grid cell (x, y, w, h)* dinormalisasi menjadi ukuran 0 hingga 1. Nilai *w* (*width*) dan *h* (*height*) dinormalisasikan dengan cara membagi nilai ukuran *w, h* yang merujuk pada ukuran *bounding box* dengan ukuran input gambar. Setiap *grid cell* memiliki bounding box masing-masing. *Bounding box* yang memiliki nilai *confidence* 0 tidak diakui dan yang memiliki nilai akan terbentuk *bounding box*. Nilai *x, y* dinormalisasikan dengan cara mengurangi titik tengah bounding box dengan titik koordinat *grid cell* lalu dibagi dengan lebar *grid cell*.



Gambar 63. Gambaran letak titik *grid cell*

Contoh pada Gambar 63 diilustrasikan memiliki nilai width & height gambar sebesar 448 x 448 pixel dengan jumlah grid cell 7x7 berukuran width & height grid cell sebesar 64x64 pixel. Input gambar tersebut memiliki bounding box berwarna ungu dengan width & height sebesar 70 x 74 pixel. Setiap bounding box memiliki titik tengah yang berada pada grid cell, untuk melakukan normalisasi nilai x, y pada bounding box ditentukan dengan grid cell dengan keberadaan titik tengah bounding box yang bersangkutan. Berikut contoh hasil dari normalisasi pada nilai (x, y, w, h) sesuai dengan Persamaan 4,5,6,7.

$$normalisasi\ x = \frac{(296 - 256)}{64} = 0{,}62$$

nilai 296 didapat dari titik *gridecell x* yang dihitung dari kiri dikali dengan satu nilai *bounding box* yaitu 64, nilai 256 didapat dari titik *gridcell x* dan 64 merupakan nilai satu *bounding box.*

$$normalisasi\ y = \frac{(290 - 192)}{64} = \ 1{,}53$$

nilai 290 didapat dari titik *gridecell y* yang dihitung dari atas dikali dengan satu nilai *bounding box* yaitu 64, nilai 192 didapat dari titik *gridecell y* dan 64 merupakan nilai satu *bounding box.*

$$normalisasi\ w = \frac{320}{448} = \ 0{,}71$$

nilai *w* merupakan *width bounding box* dengan nilai 320, nilai 448 merupakan nilai image *width.*

$$normalisasi\ h = \frac{192}{448} = 0{,}42$$

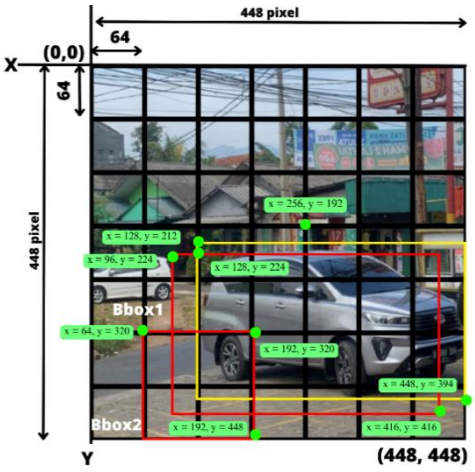
nilai *h* merupakan *height bounding box* dengan nilai 196, nilai 448 merupakan nilai image *height.* Contoh apabila terdapat 2 Bbox dengan class yang sama, maka hasil dimensi ukuran tensor dapat dihitung sebagai berikut. Sesuai dengan Persamaan 11.

Tensor = (7 x 7 x (2 x 5 + 2)) = (7 x 7 x 12)

Ukuran matriks 2 bbox, Sesuai dengan Persamaan 9.

Tensor = (nC x (S x S x nB)) = 2 x (7 x 7 x 2) = (2 x 98)

Contoh perhitungan *Iou* apabila kondisi 2 bounding box pada class yang sama memiliki koordinat *bounding box* yang dapat dilihat pada Gambar 64.



Gambar 64. *Plot bounding box*

Proses perhitungan *Iou Bbox* 1

$$Iou_{pred}^{truth} \frac{(448-128)\ X\ (394-224)}{((448-128)\ X\ (394-212)+(416-96)X\ (416-224)-area\ of\ overlap}$$

$$Bbox1ConfidenceScore = \frac{54400}{58240 + 61440 - 54400} = 0{,}47$$

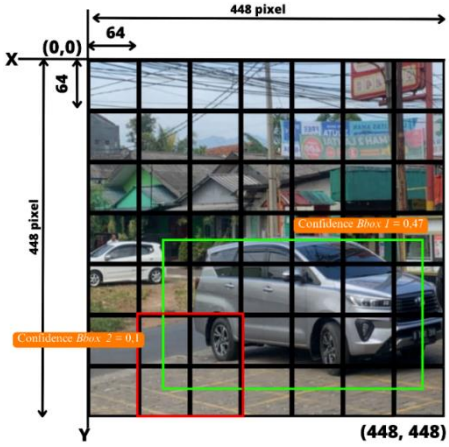
Proses perhitungan *Iou Bbox* 2

$$Iou_{pred}^{truth} \frac{(394-320\ )\ X\ (192-96)}{((448-128)\ X\ (394-212)+(416-64)X\ (416-320)-area\ of\ overlap}$$

$$Bbox1ConfidenceScore = \frac{7104}{58240 + 33792 - 7104} = 0{,}1$$

C. arsitektur YOLO menggunakan *reduction factor* 32 untuk melakukan *downsampling* input citra. Apabila input citra berukuran 448 x 448 x 3 (3 adalah parameter RGB) saat proses *convolutional network* dilakukan penurunan *resolusi output* dengan cara melakukan operasi pembagian 448/32 = 14. *Output* akan menghasilkan *vector* dengan bentuk (14, 14, 38), 13x13 adalah *grid* akhir dan 38 didapatkan dari operasi *Bx(5+C).* nilai *Bx* adalah parameter *matrix yolo* (*x, y, w, h, cf*) berjumlah 5 dan nilai *C* banyaknya *class* yang dilakukan penelitian. Penelitian ini menggunakan 2 *class.* Proses selanjutnya melalui proses *Convolution layer* dengan *ReLu* (*Rectified Linear Unit*) / *Silu (Sigmoid- weighted Linear Unit)* yaitu fungsi aktivasi *element wise* ke *output* dari aktivasi yang dihasilkan lapisan sebelumnya.

D. pada *fully connected layers*, citra gambar akan menghasilkan *bounding box* yang akan digunakan untuk proses klasifikasi objek, namun untuk hasil prediksi yang banyak akan dilakukan proses *threshold* atau *non-maximum suppression*, yaitu *bounding box* dengan hasil *confidence* yang tinggi yang akan ditampilkan. Proses ini akan menyortir *bounding box* dengan memiliki tingkat nilai *confidence* tinggi yang akan diperhitungkan. NMS (*non-maximum suppression*) melakukan proses apabila pada banyak *bounding box* mengalami *overlap* satu sama lain maka *confidence* tertinggi yang akan diperhitungkan dan sisanya akan di suppression atau disingkirkan. Proses NMS ditunjukan pada Gambar 65.



Gambar 65. Proses NMS dalam mengurangi *overlap* pada *bounding box*

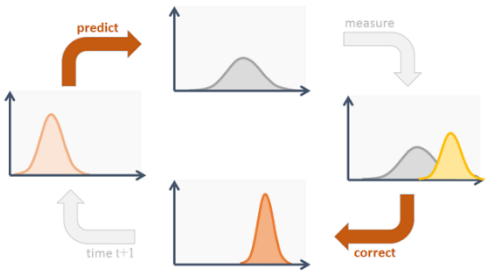
Setelah berhasil melalui proses YOLO *bounding Box* akan diberi label lalu hasil deteksi objek dengan *bounding box* dan label pada citra gambar atau video dengan aturan nilai yang lebih tinggi dari *threshold* yang ditentukan akan ditampilkan pada *output.* Contoh *output* pada YOLO seperti pada Gambar 66.

Gambar 66. *Output* deteksi YOLO

Lampiran 4. *DeepSORT*

Dalam melakukan proses tracking objek menggunakan *DeepSORT* beberapa proses penting dilakukan hingga dapat dilakukan tracking objek merujuk class yang dimaksud, proses *DeepSORT* di antaranya:

A. *Track handling* dan *estimation*, yaitu penggunaan *filter kalman* digunakan untuk memperhitungkan *noise* dalam pendeteksian dan menggunakan keadaan sebelumnya dalam memprediksi kesesuaian untuk *bounding box*. Seperti yang ditunjukan pada Gambar 67.



Gambar 67. ilustrasi dengan *filter kalman*

Berdasarkan gambar 26 filter kalman bekerja secara *rekursif,* pada pengambilan pembacaan saat ini untuk memprediksi saat ini dan menggunakan pengukuran tersebut dan memperbaharui hasil prediksi sehingga dapat membuat perkiraan yang tepat dimana objek akan berada pada frame berikutnya

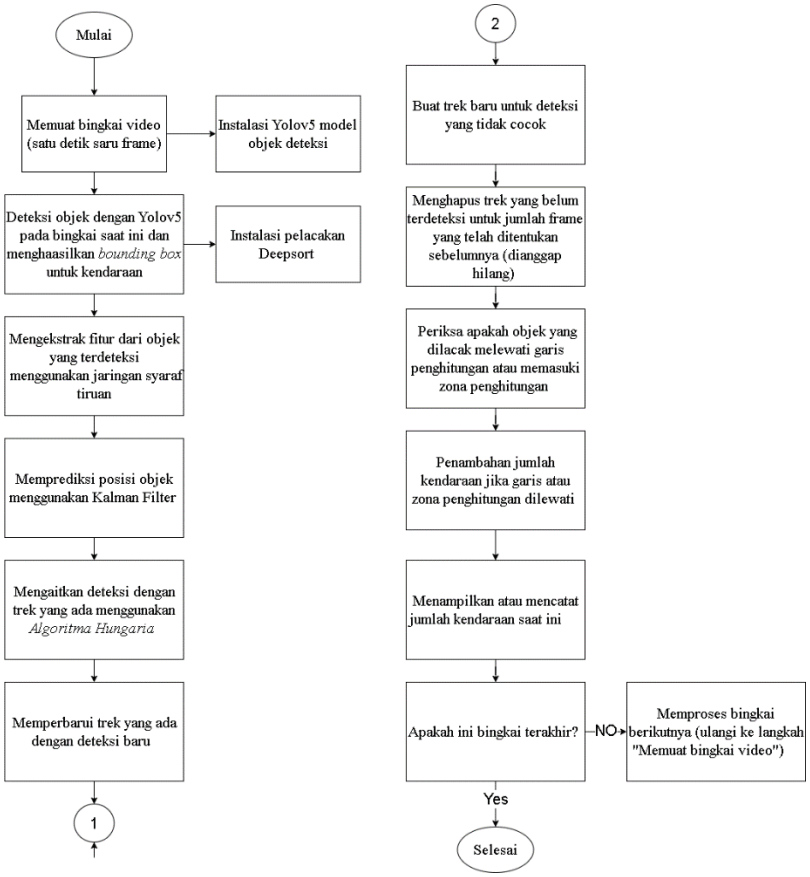
Untuk setiap deteksi perlu dibuatnya *trek,* yang berisi semua informasi status yang diperlukan seperti parameter untuk melacak dan menghapus *trek* yang terakhir berhasil dideteksi, karena objek tersebut akan meninggalkan tempat kejadian. Untuk menghilangkan duplikat *trek,* ada jumlah minimum ambang batas deteksi untuk beberapa *frame* pertama.

B. *Assignment problem* merupakan penyelesaian hubungan antara keadaan Kalman yang diprediksi dan pengukuran yang baru dengan cara membangun masalah penugasan yang dapat diselesaikan dengan menggunakan Algoritma Hungaria.

Saat kita memiliki kotak pembatas baru yang dilacak dari filter Kalman, masalah berikutnya adalah mengaitkan deteksi baru dengan prediksi terbaru.

C. *Deep Appearance Descriptor,* CNN dibuat untuk mengekstrak informasi fitur dari target, dan fitur tersebut diproyeksikan ke *hypersphere* terpadu menggunakan normalisasiL_2. Setelah dilatih perlu meneruskan semua potongan *bounding box* yang terdeteksi dari gambar ke jaringan dan mendapatkan vektor fitur dimensi "128x1".

Berikut merupakan *flowchart* arsitektur diagram deepsort ditunjukan pada Gambar 68.



Gambar 68. Arsitektur diagram deepsort

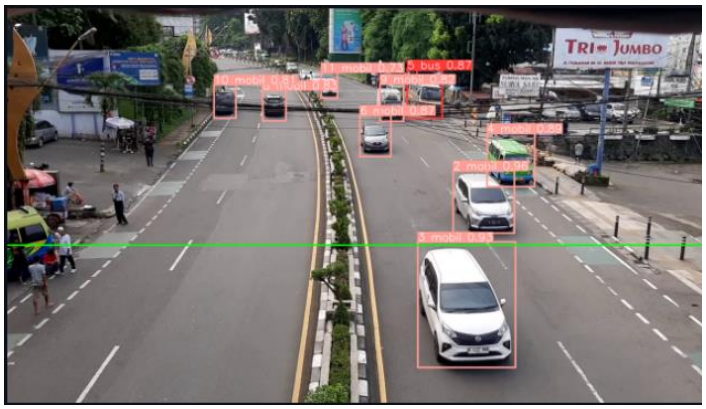
Dari Gambar 70 diketahui deepsort memiliki kekurangan dan kelebihan, berikut merupakan kekurangan dari algoritma deepsort:

1. Ketergantungan pada Detektor Objek
   Kualitas pelacakan sangat bergantung pada akurasi dan keandalan detektor objek yang digunakan. Jika deteksi objek gagal atau tidak akurat, pelacakan juga akan terpengaruh.
2. Kompleksitas Komputasi
   DeepSORT masih memerlukan sumber daya komputasi yang cukup besar, terutama jika digunakan dengan model deteksi objek yang kompleks dan berat seperti YOLOv5 atau Faster R-CNN.
3. Pemeliharaan Identitas Objek

identitas objek yang dilacak dapat berubah jika objek-objek tersebut sangat mirip satu sama lain atau jika ada kesalahan deteksi yang menyebabkan asosiasi yang salah

adapun kelebihan dari algotima deepsort seperti berikut:

1. Akurasi Tinggi dalam Pelacakan Multi-Objek
   DeepSORT menggunakan metode deteksi fitur yang kuat yang memungkinkan pelacakan multi-objek dengan akurasi tinggi. Algoritma ini mampu mempertahankan identitas objek bahkan dalam skenario yang padat.
2. Real-Time Processing
   DeepSORT dioptimalkan untuk memproses video secara real-time, yang sangat penting untuk aplikasi seperti pengawasan lalu lintas, keamanan, dan analisis video langsung.
3. Integrasi dengan Detektor Objek Modern
   DeepSORT dapat diintegrasikan dengan berbagai detektor objek modern seperti YOLO, SSD, dan Faster R-CNN, sehingga fleksibel dan dapat digunakan dalam berbagai kasus penggunaan. Setelah berhasil melalui beberapa tahapan proses contoh output dari yolov5 dan DeepSORT seperti pada Gambar 69.

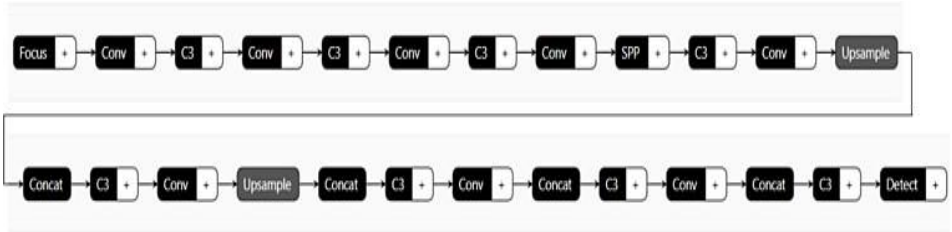


Gambar 69. *Output tracking DeepSORT*

Pada Gambar 71 merupakan hasil *output* dari gabungan algoritma *yolov5* dengan *deepsort*, *line* berwarna hijau berfungsi untuk menghitung setiap kendaraan yang lewat sesuai dengan *class*nya.

Lampiran 5. Konfigurasi Model

Pada penelitian ini menggunakan model arsitektur yolo v5s yang dapat dilihat pada Gambar 70.



Gambar 70. *Yolov5 model*

Pada jaringan Yolo v5s memiliki nilai *depth multiple* dan *width multiple* bernilai 0,33 dan 0,50 berbeda dengan seri model yolo v5 yang memiliki konfigurasi nilai tertinggi yaitu 1.0 dan 1.0. susunan pembentukan model dapat dilihat pada Gambar 71.

yolov5 > models > yolov5s.yaml

```yaml
# YOLOv5 🚀 by Ultralytics, GPL-3.0 license

# Parameters
nc: 80  # number of classes
depth_multiple: 0.33  # model depth multiple
width_multiple: 0.50  # layer channel multiple
anchors:
  - [10,13, 16,30, 33,23]  # P3/8
  - [30,61, 62,45, 59,119]  # P4/16
  - [116,90, 156,198, 373,326]  # P5/32

# YOLOv5 v6.0 backbone
backbone:
  # [from, number, module, args]
  [[-1, 1, Conv, [64, 6, 2, 2]],  # 0-P1/2
   [-1, 1, Conv, [128, 3, 2]],  # 1-P2/4
   [-1, 3, C3, [128]],
   [-1, 1, Conv, [256, 3, 2]],  # 3-P3/8
   [-1, 6, C3, [256]],
   [-1, 1, Conv, [512, 3, 2]],  # 5-P4/16---------
   [-1, 9, C3, [512]],
   [-1, 1, Conv, [1024, 3, 2]],  # 7-P5/32
   [-1, 3, C3, [1024]],
   [-1, 1, SPPF, [1024, 5]],  # 9
  ]*

# YOLOv5 v6.0 head
head:
  [[-1, 1, Conv, [512, 1, 1]],
   [-1, 1, nn.Upsample, [None, 2, 'nearest']],
   [[-1, 6], 1, Concat, [1]],  # cat backbone P4
   [-1, 3, C3, [512, False]],  # 13

   [-1, 1, Conv, [256, 1, 1]],
   [-1, 1, nn.Upsample, [None, 2, 'nearest']],
   [[-1, 4], 1, Concat, [1]],  # cat backbone P3
   [-1, 3, C3, [256, False]],  # 17 (P3/8-small)

   [-1, 1, Conv, [256, 3, 2]],
   [[-1, 14], 1, Concat, [1]],  # cat head P4
   [-1, 3, C3, [512, False]],  # 20 (P4/16-medium)

   [-1, 1, Conv, [512, 3, 2]],
   [[-1, 10], 1, Concat, [1]],  # cat head P5
   [-1, 3, C3, [1024, False]],  # 23 (P5/32-large)

   [[17, 20, 23], 1, Detect, [nc, anchors]],  # Detect(P3, P4, P5)
  ]
```

Gambar 71. Konfigurasi model

Anchors merupakan kotak pembatas yang sudah ditentukan nilai tinggi dan lebarnya, yang digunakan untuk menangkap skala dan rasio terhadap *object class* yang diinginkan.

Lampiran 6. Yolo *hyperparams*

Yolo *hyperparams* atau *image augmentation* digunakan untuk mendapatkan data yang baru dalam proses mengoptimalkan model yang dibuat. Pada proses *training model* nilai *learning rate* (lr0) merupakan nilai terpenting sebesar 0,01. *Learning rate* ini memiliki nilai di 0 sampai dengan 1, jika semakin besar nilai *learning rate* proses *training* akan semakin cepat namun akan memiliki akurasi yang rendah dan kemungkinan error semakin tinggi. Berikut proses terjadinya *image augmentation* pada Gambar 72.

Gambar 72. Jenis *image augmentation*

Lampiran 7. *Source Code* Fungsi *Video_input()* Pada *Model Website*

```
# upload video
video_file_buffer = st.file_uploader("Silahkan Upload Video Untuk Memulai Deteksi Objek", type=['mp4', 'mov', 'avi'])
newpath = r"runs/video_upload"

if video_file_buffer:
    st.text('"Detail video"')
    st.video(video_file_buffer)
    # save video from streamlit into "videos" folder for future detect
    with open(os.path.join('videos', video_file_buffer.name), 'wb') as f:
        f.write(video_file_buffer.getbuffer())
    st.success("File Uploaded")

status = st.empty()
stframe = st.empty()
if video_file_buffer is None:
    status.markdown('<font size= "4"> **Status:** Waiting for input </font>', unsafe_allow_html=True)
else:
    status.markdown('<font size= "4"> **Status:** Ready </font>', unsafe_allow_html=True)

mobil, bus, truk, motor = st.columns(4)

with mobil:
    st.markdown('**Mobil**')
    mobil_text = st.markdown('__')
with bus:
    st.markdown('**Bus**')
    bus_text = st.markdown('__')
with truk:
    st.markdown('**Truk**')
    truk_text = st.markdown('__')
with motor:
    st.markdown('**Motor**')
    motor_text = st.markdown('__')
```

Gambar 73. Tampilan input *function streamlit website*

Lampiran 8. Pengembangan sidebar pada website

```python
if __name__ == '__main__':        You, 3 weeks ago • initial commit
    st.sidebar.header('🚗 Konfigurasi Model')
    # custom class
    assigned_class_id = [0, 1, 2, 3]
    names = ['bus', 'mobil', 'motor', 'truk']

    # Always display the multiselect widget for selecting custom classes
    assigned_class_id = []
    assigned_class = st.sidebar.multiselect('Pilih class spesifik untuk proses deteksi', names)
    for each in assigned_class:
        assigned_class_id.append(names.index(each))

    # Display selected class in the sidebar
    st.sidebar.caption("class terpilih : {}".format(', '.join(assigned_class) if assigned_class else 'None'))

    # st.write(assigned_class_id)
    # setting hyperparameter
    line = st.sidebar.number_input('🛠️ Line position', min_value=0.0, max_value=1.0, value=0.6, step=0.1)
    confidence = st.sidebar.slider('🔨 Confidence', min_value=0.0, max_value=1.0, value=0.5)
    st.sidebar.write("Confidence set :", confidence)
    iou_thres = st.sidebar.slider('🖊️ Iou Threshold', min_value=0.0, max_value=1.0, value=0.5)
    st.sidebar.write("Iou set :", iou_thres)

    st.subheader("🎞️ Video Input")
    st.write("""silahkan mengunggah video dengan ketentuan tidak lebih dari 200MB untuk
            mempercepat proses unggah dan proses deteksi. format video yang didukung yaitu MP4, MOV, AVI.""")
    # upload video
    video_file_buffer = st.file_uploader("Silahkan Upload Video Untuk Memulai Deteksi Objek", type=['mp4', 'mov', 'avi'])
    newpath = r"runs/video_upload"
    if not os.path.exists(newpath): os.makedirs(newpath)

    if video_file_buffer:
        st.success("File Uploaded")
        st.text('"Detail video"')
        st.video(video_file_buffer)
        # save video from streamlit into "videos" folder for future detect
        with open(os.path.join("runs/video_upload", video_file_buffer.name), 'wb') as f:
            f.write(video_file_buffer.getbuffer())

    status = st.empty()
    stframe = st.empty()
    if video_file_buffer is None:
        status.markdown('<font size= "4"> **Status:** Waiting for input </font>', unsafe_allow_html=True)
    else:
        status.markdown('<font size= "4"> **Status:** Ready </font>', unsafe_allow_html=True)
```

Gambar 74. *Source Code* menampilkan sidebar

Pengembangan proses video di*input* menggunakan fungsi *video_file_buffer* yang berguna untuk memuat video *input* dengan mendukung format mp4, mov, avi. Setelah berhasil melakukan *upload file* ke dalam *website* maka akan dicek ketersediaan videonya, dan akan disimpan ke dalam *runs/video_upload.*

Lampiran 9. Proses unggah dan file *requirements*

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS  COMMENTS

 create mode 100644 yolov5/utils/google_app_engine/additional_requirements.txt
 create mode 100644 yolov5/utils/google_app_engine/app.yaml
 create mode 100644 yolov5/utils/loggers/__init__.py
 create mode 100644 yolov5/utils/loggers/clearml/README.md
 create mode 100644 yolov5/utils/loggers/clearml/__init__.py
 create mode 100644 yolov5/utils/loggers/clearml/clearml_utils.py
 create mode 100644 yolov5/utils/loggers/clearml/hpo.py
 create mode 100644 yolov5/utils/loggers/comet/README.md
 create mode 100644 yolov5/utils/loggers/comet/comet_utils.py
 create mode 100644 yolov5/utils/loggers/comet/hpo.py
 create mode 100644 yolov5/utils/loggers/wandb/README.md
 create mode 100644 yolov5/utils/loggers/wandb/__init__.py
 create mode 100644 yolov5/utils/loggers/wandb/log_dataset.py
 create mode 100644 yolov5/utils/loggers/wandb/sweep.py
 create mode 100644 yolov5/utils/loggers/wandb/sweep.yaml
 create mode 100644 yolov5/utils/loggers/wandb/wandb_utils.py
 create mode 100644 yolov5/utils/loss.py
 create mode 100644 yolov5/utils/metrics.py
 create mode 100644 yolov5/utils/plots.py
 create mode 100644 yolov5/utils/segment/__init__.py
 create mode 100644 yolov5/utils/segment/augmentations.py
 create mode 100644 yolov5/utils/segment/dataloaders.py
 create mode 100644 yolov5/utils/segment/general.py
 create mode 100644 yolov5/utils/segment/loss.py
 create mode 100644 yolov5/utils/segment/metrics.py
 create mode 100644 yolov5/utils/segment/plots.py
 create mode 100644 yolov5/utils/torch_utils.py
 create mode 100644 yolov5/utils/triton.py
 create mode 100644 yolov5/val.py
 create mode 100644 yolov5deepsort_skripsi.ipynb
 create mode 100644 yolov5s.pt
PS C:\Users\ZAKA DARMAWAN\Documents\KFlearning\yolov5deepsort_skripsi> git push origin master
Enumerating objects: 366, done.
Counting objects: 100% (366/366), done.
Delta compression using up to 4 threads
Compressing objects: 100% (360/360), done.
Writing objects: 100% (366/366), 14.26 MiB | 6.69 MiB/s, done.
Total 366 (delta 66), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (66/66), done.
To https://github.com/ZakaDarmawan/Yolov5Deepsort_Streamlit.git
 * [new branch]      master -> master
```

Gambar 75. proses unggah proyek ke *repository github*

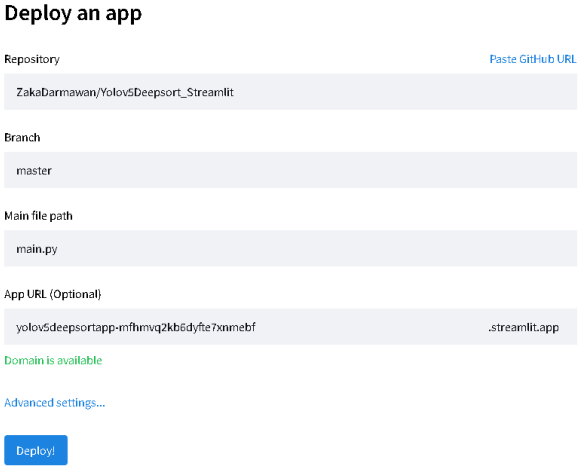
untuk *requirements* yang digunakan pada penelitian ini ditunjukan pada Gambar 76.

```
# YOLOv5 requirements
# Usage: pip install -r requirements.txt

# Base ----------------------------------------
matplotlib>=3.2.2
numpy>=1.18.5
opencv-python>=4.1.1        You, 2 days ago • initial commit
Pillow>=7.1.2
PyYAML>=5.3.1
requests>=2.23.0
scipy>=1.4.1
torch>=1.7.0  # see https://pytorch.org/get-started/locally/ (recommended)
torchvision>=0.8.1
tqdm>=4.64.0
# protobuf<=3.20.1  # https://github.com/ultralytics/yolov5/issues/8012

# Logging -------------------------------------
tensorboard>=2.4.1
# clearml>=1.2.0
# comet

# Plotting ------------------------------------
pandas>=1.1.4
seaborn>=0.11.0

# Export --------------------------------------
# coremltools>=6.0  # CoreML export
# onnx>=1.9.0  # ONNX export
# onnx-simplifier>=0.4.1  # ONNX simplifier
# nvidia-pyindex  # TensorRT export
# nvidia-tensorrt  # TensorRT export
# scikit-learn<=1.1.2  # CoreML quantization
# tensorflow>=2.4.1  # TF exports (-cpu, -aarch64, -macos)
# tensorflowjs>=3.9.0  # TF.js export
# openvino-dev  # OpenVINO export

# Deploy --------------------------------------
# tritonclient[all]~=2.24.0

# Extras --------------------------------------
ipython  # interactive notebook
psutil  # system utilization
thop>=0.1.1  # FLOPs computation
```

Gambar 76. *Library streamlit cloud requirements*

Pada Gambar 78 merupakan library yang digunakan untuk diberi hastag merupakan komentar pada *file requirements* tidak ikut masuk ke *requirements project*. Proses *upload* menggunakan terminal dengan langkah pertama meng*install Git* menggunakan perintah *git init,* langkah berikutnya menambahkan perubahan pada *staging* area dengan melakukan perintah *git add.* Untuk menghubungkan *local project* dan *repository github* ke *github* dengan perintah *git remote add origin link_repository.* Selanjutnya mengunggah dan mengirim perubahan ke *repository github* dengan melakukan perintah *git push origin main-force.*

Lampiran 10. Proses *deployment*



Gambar 77. proses *deploy app* melalui *streamlit cloud*

Setelah proses *deploy* selesai *website* dapat langsung digunakan menggunakan URL https://yolov5deepsortapp.streamlit.app. Hasil dari uji coba *website* berhasil menggunakan *streamkit cloud,* dan berjalan dengan baik. Untuk deteksi video akan berjalan dengan lambat dikarenakan *resource* menggunakan *server cloud.*

Lampiran 11. Perhitungan output dimensi konvolutional pada tabel 6 jika yolov5 menggunakan input 640 x 640 x 3.

1. Image input yolo 640 x 640 x 3 (Conv 1)

$$Output\ W = \frac{W-N+P}{S} + 1 = \frac{640-2+2*2}{2} + 1 = 320$$

$$Output\ H = \frac{W-N+P}{S} + 1 = \frac{640-2+2*2}{2} + 1 = 320$$

2. Image input 320 x 320 x 32 (Conv 2)

$$Output\ W = \frac{W-N+2P}{S} + 1 = \frac{320-3+2*1}{2} + 1 = 160$$

$$Output\ H = \frac{W-N+2P}{S} + 1 = \frac{320-3+2*1}{2} + 1 = 160$$

3. Image input 160 x 160 x 32 (C3)

$$Output\ W = \frac{W-N+P}{S} = \frac{160-1+2*0}{2} = = 1 = 160$$

$$Output\ h = \frac{W-N+2P}{S} = \frac{160-1+2*0}{2} = = 1 = 160$$

Lampiran 12. Perhitungan *output* dimensi konvolusi menggunakan *input* 640 x 640 x 3 pada tabel 7

1. Image input yolo 640 x 640 x 3 (Conv 1)

$$Output\ W = \frac{W-N+P}{S} + 1 = \frac{640-6+2*0}{2} + 1 = 318$$

$$Output\ H = \frac{W-N+P}{S} + 1 = \frac{640-6+2*0}{2} + 1 = 318$$

2. Image input yolo 318 x 318 x 128 (Conv 2)

$$Output\ W = \frac{W-N+P}{S} + 1 = \frac{318-3+2*0}{2} + 1 = 158$$

$$Output\ H = \frac{W-N+P}{S} + 1 = \frac{318-3+2*0}{2} + 1 = 158$$

Lampiran 13. *User Acceptance Test Website*

Hasil UAT *Website*					
No	*Test Case*	Berhasil / Gagal	Penguji	Tanggal Uji	Catatan
1	Nama Uji : Tampilan Halaman Deskripsi : Halaman *Website* memiliki 1 halaman Kasus Uji : Halaman *input video* Hasil yang diharapkan : halaman sesuai dengan menu *input*	Berhasil	Zaka	3/02/2024	Tampilan halaman meliputi halaman primer dan *sidebar*
2	Nama Uji : *Website* dapat melakukan unggah *file* Deskripsi : *Website* dapat melakukan unggah video Kasus Uji : unggah 1 video Hasil yang diharapkan : berhasil melakukan proses *upload* dan file tersimpan pada direktori	Berhasil	Zaka	3/01/2024	*File* yang diunggah memiliki batas ukuran tidak melebihi 200mb
3	Nama Uji : *preview* video Deskripsi : *Website* dapat menampilkan *preview* video Kasus Uji : *Upload* 1 video pada *input* video Hasil yang diharapkan : *Website* berhasil menampilkan *preview* video	Berhasil	Zaka	03/02/2024	*Website* menampilkan *preview* video setelah proses unggah
4	Nama Uji : *Website* dapat deteksi model video Deskripsi : *Website* berhasil melakukan deteksi video *real time* dengan model Kasus Uji : menggunakan 1 video	Berhasil	Zaka	03/02/2024	Model akan mendeteksi objek pada video dan *hardware* sangat berpengaruh pada

	Hasil yang diharapkan : model mendeteksi objek pada video sampai selesai				kecepatan proses deteksi
5	Nama Uji : simpan *output* deteksi pada video	Berhasil	Zaka	03/02/2024	Hasil *output* tersimpan pada direktori *runs* dengan *format* .MP4
	Deskripsi : menyimpan hasil deteksi objek *output* simpan pada direktori runs				
	Kasus Uji : menyimpan hasil deteksi				
	Hasil yang diharapkan : *Website* dapat menyimpan hasil *output* deteksi				
6	Nama Uji : Uji *custom class*	Berhasil	Zaka	03/02/2024	Hasil video deteksi sesuai *class* yang dipilih akan tersimpan pada direktori *runs*
	Deskripsi : *model* dapat mendeteksi sesuai dengan *class* yang dipilih				
	Kasus Uji : *input video*				
	Hasil yang diharapkan : hasil potongan objek gambar dapat disimpan pada menu *runs*				
7	Nama Uji : *tracking class* pada *website*	Berhasil	Zaka	03/02/2024	*Website* menampilkan hasil jumlah deteksi sesuai dengan *classnya*
	Deskripsi : *Website* dapat menghitung jumlah deteksi				
	Kasus Uji : *upload* 1 video pada *input* video				
	Hasil yang diharapkan : *Website* dapat menghitung jumlah hasil deteksi sesuai dengan *class*nya				

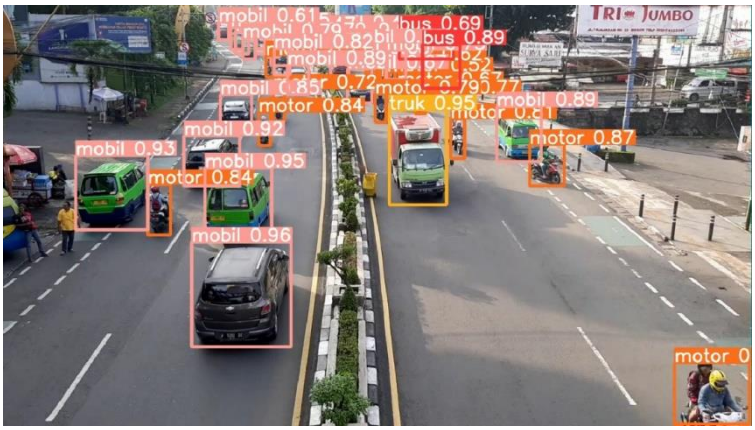
Lampiran 14. *User Acceptance Test Model*

Hasil *UAT Model Dataset Regular*							
No	video	*Class* seharusnya (motor, mobil, bus, dan truk)	Hasil Deteksi	TP	TN	FP	FN
1	Pagi, 5 December (2.32 menit)	137, 92, 0, 9	137, 92, 0, 12	243	3	0	0
2	Sore, 5 December (2.38 menit)	116, 122, 4, 4	116, 122, 4, 4	246	0	0	0
3	Pagi, 6 December (3.00 menit)	233, 107, 2, 15	233, 107, 2, 15	357	0	0	0
4	Sore, 6 December (3.03 menit)	151, 149, 10, 12	151, 149, 10, 12	322	3	1	0
5	Pagi, 7 December (2.35 menit)	125, 121, 3, 15	125, 121, 3, 15	264	0	0	0
6	Sore, 7 December (2.40 menit)	177, 131, 9, 8	177, 131, 9, 8	325	0	0	0
7	Pagi, 8 December (3.02 menit)	193, 128, 5, 17	193, 128, 5, 17	343	0	0	0
8	Sore, 8 December (3.03 menit)	242, 173, 0, 10	242, 173, 0, 10	425	0	0	0
9	Pagi, 9 December (2.31 menit)	163, 113, 2, 7	163, 113, 2, 7	285	0	0	0
10	Sore, 9 December (2.55 menit)	160, 126, 7, 3	160, 126, 7, 3	296	0	0	0
11	Pagi, 10 December (2.42 menit)	161, 102, 1, 14	161, 102, 1, 14	278	0	0	0
12	Sore, 10 December (2.56 menit)	193, 128, 5, 17	193, 128, 5, 17	343	0	0	0
13	Pagi, 11 December (3.24 menit)	348, 155, 3, 21	348, 155, 3, 21	522	0	0	0
14	Sore, 11 December (3.01 menit)	182, 185, 6, 14	182, 185, 6, 14	387	0	0	0

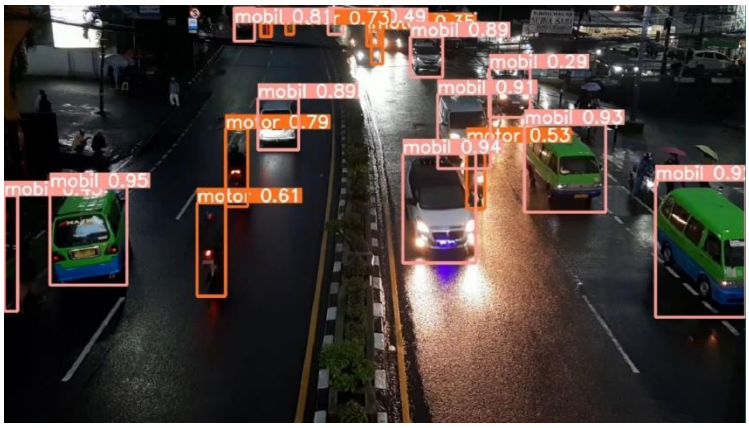
Lampiran 15. Source code *track.py*

```
import os
os.environ["OMP_NUM_THREADS"] = "1"
os.environ["OPENBLAS_NUM_THREADS"] = "1"
os.environ["MKL_NUM_THREADS"] = "1"
os.environ["VECLIB_MAXIMUM_THREADS"] = "1"
os.environ["NUMEXPR_NUM_THREADS"] = "1"
      You, 4 weeks ago • initial commit
import sys
sys.path.insert(0, './yolov5')

import streamlit as st
import time
import IPython
import argparse
import os
import platform
import shutil
import time
import pathlib
# temp = pathlib.PosixPath
pathlib.PosixPath = pathlib.WindowsPath
from pathlib import Path
import cv2
import torch
import torch.backends.cudnn as cudnn
import posixpath

from yolov5.models.experimental import attempt_load
from yolov5.utils.downloads import attempt_download
from yolov5.models.common import DetectMultiBackend
from yolov5.utils.dataloaders import LoadImages, LoadStreams
from yolov5.utils.general import (LOGGER, check_img_size, non_max_suppression, scale_boxes,
                                  check_imshow, xyxy2xywh, increment_path)
from yolov5.utils.torch_utils import select_device, time_sync
from yolov5.utils.plots import Annotator, colors
from deep_sort.utils.parser import get_config
from deep_sort.deep_sort import DeepSort

FILE = Path(__file__).resolve()
ROOT = FILE.parents[0]  # yolov5 deepsort root directory
if str(ROOT) not in sys.path:
    sys.path.append(str(ROOT))  # add ROOT to PATH
ROOT = Path(os.path.relpath(ROOT, Path.cwd()))  # relative

# count_mobil, count_bus, count_truk = 0, 0, 0
data_mobil = []
data_bus = []
data_truk = []
data_motor = []
already = []
line_pos = 0.6

def detect(opt, stframe, mobil, bus, truk, motor, line, fps_rate, class_id):
    out, source, yolo_model, deep_sort_model, show_vid, save_vid, save_txt, imgsz, evaluate, half, project, name, exist_ok= \
        opt.output, opt.source, opt.yolo_model, opt.deep_sort_model, opt.show_vid, opt.save_vid, \
        opt.save_txt, opt.imgsz, opt.evaluate, opt.half, opt.project, opt.name, opt.exist_ok       You, 4 weeks ago • initial c
    # choose custom class from streamlit
    opt.classes = class_id
    webcam = source == '0' or source.startswith(
        'rtsp') or source.startswith('http') or source.endswith('.txt')
    sum_fps = 0
    line_pos = line
    save_vid = True
    # initialize deepsort
    cfg = get_config()
    cfg.merge_from_file(opt.config_deepsort)
    deepsort = DeepSort(deep_sort_model,
                        max_dist=cfg.DEEPSORT.MAX_DIST,
                        max_iou_distance=cfg.DEEPSORT.MAX_IOU_DISTANCE,
                        max_age=cfg.DEEPSORT.MAX_AGE, n_init=cfg.DEEPSORT.N_INIT, nn_budget=cfg.DEEPSORT.NN_BUDGET,
                        use_cuda=True)
```

Lampiran 16. Hasil Pengujian Model *Yolov5*



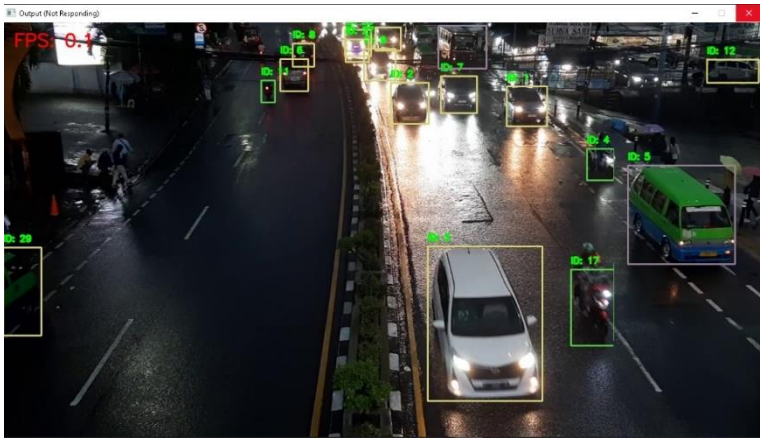
Gambar 78. Pengujian input video pagi hari



Gambar 79. Pengujian input video sore hari

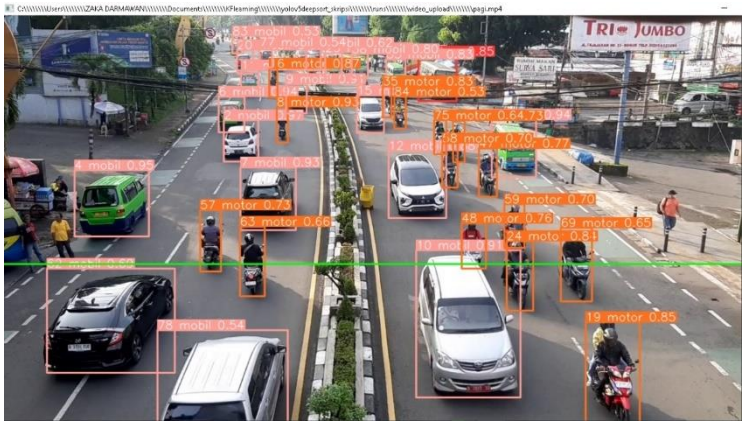
Lampiran 17. Hasil Pengujian Model *Deepsort*

Gambar 80. Pengujian input video pagi hari

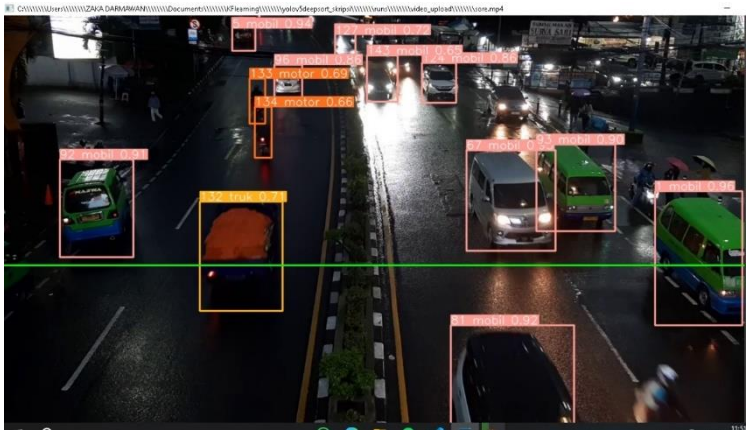


Gambar 81. Pengujian input video sore hari

Lampiran 18. Hasil pengujian Model gabung *Yolov5 + Deepsort*



Gambar 82. Pengujian input video pagi hari



Gambar 83. Pengujian input sore hari

KEPUTUSAN DEKAN
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS PAKUAN No. : 3833/KEP/D/FMIPA/XI/2023

T E N T A N G

PENGANGKATAN PEMBIMBING TUGAS AKHIR
PADA PROGRAM STUDI ILMU KOMPUTER
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS PAKUAN

DEKAN FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS PAKUAN,

Menimbang : a. bahwa setiap mahasiswa tingkat akhir Program Strata Satu (S1) harus melaksanakan Tugas Akhir sebagaimana tercantum di dalam kurikulum setiap Program Studi di lingkungan Fakultas MIPA Universitas Pakuan.
b. bahwa untuk pelaksanaan Tugas Akhir diperlukan pengawasan dari pembimbing.
c. bahwa sehubungan dengan point a dan b di atas perlu dituangkan dalam suatu Keputusan Dekan.

Mengingat : 1. Undang-undang RI No.: 20 Tahun 2003 tentang Sistem Pendidikan Nasional.
2. Peraturan Pemerintah No.: 60 Tahun 1999 tentang Pendidikan Tinggi.
3. Statuta Universitas Pakuan Tahun 2019.
4. Surat Keputusan Rektor Nomor: 35/KEP/REK/VIII/2020 tanggal 03 Agustus 2020 tentang Pemberhentian Dekan dan Wakil Dekan Masa Bakti 2015-2020 serta Pengangkatan Dekan dan Wakil Dekan Masa Bakti 2020-2025 di lingkungan Universitas Pakuan.
5. Ketentuan Akademik yang tercantum dalam Buku Panduan Studi Fakultas MIPA, Universitas Pakuan Tahun 2023.

Memperhatikan : Usulan dari Ketua Program Studi Ilmu Komputer FMIPA UNPAK.

M E M U T U S K A N

Menetapkan :

Pertama : Mengangkat pembimbing yang namanya tersebut di bawah ini :
1.Pembimbing Utama : Dr. Tjut Awaliyah Zuraiyah, S.Kom., M.Kom.
2.Pembimbing Pendamping : Mulyati, M.Kom.

Untuk membimbing dalam rangka melaksanakan tugas akhir bagi mahasiswa :

Nama	: Zaka Darmawan
NPM	: 065119091
Program Studi	: Ilmu Komputer
Judul Skripsi	: Implementasi Algoritma Yolov5 Untuk Mendeteksi Dan
Menghitung	Jumlah Kendaraan Menggunakan Metode DeepSORT

Kedua : Kepada para pembimbing diharapkan dapat menjalankan tugasnya sebagai pembimbing dengan sebaik-baiknya.

Ketiga : Dalam waktu 1 (satu) bulan setelah diterbitkannya SK ini, mahasiswa wajib melaksanakan Seminar Rencana Penelitian yang diselenggarakan oleh Program Studi Ilmu Komputer dengan dihadiri oleh Pembimbing dan Penguji.

Keempat : Dana untuk honorarium pembimbing dibebankan kepada mahasiswa yang ketentuannya diatur oleh Fakultas MIPA.

Kelima : Surat Keputusan ini berlaku untuk jangka waktu 1 (satu) tahun sejak tanggal ditetapkan sampai dengan mahasiswa tersebut Lulus Sidang/Ujian Skripsi, dengan ketentuan akan diadakan perubahan/ perbaikan sebagaimana mestinya bila dikemudian hari terdapat kekeliruan dalam penetapannya.

Ditetapkan di : Bogor
Pada tanggal : 06 November 2023

Dekan,

Asep Denih, S,Kom., M.Sc., Ph.D.

Tembusan :

1. Yth. Ketua Program Studi Komputer;
2. Yth. Dr. Tjut Awaliyah ZuraiyahS.Kom., M.Kom.; 3. Yth. Mulyati, M.Kom.;
4. Arsip.

Jl. Pakuan P.O. Box 452, Bogor 16143, Telp./Fax. (0251) 8375547
Website : https://fmipa.unpak.ac.id

Kartu Bimbingan Mahasiswa

Program Studi Ilmu komputer FMIPA – UNPAK

Nama Mahasiswa : Zaka Darmawan

NPM : 065119091

Judul : Implementasi Yolov5 Untuk Mendeteksi Dan Menghitung Jumlah Kendaraan Menggunakan Deepsort

Pembimbing I : Tjut Awaliyah Zuraiyah,M.kom

Pembimbing II : Mulyati,S.kom

No	Hari, Tanggal	Catatan	Tanda Tangan	
			Pemb.I	Pemb.II
1.				
2.				
3.				
4.				

Bogor, ……

Program Studi Ilmu Komputer
Fakultas MIPA –
UNPAK Ketua,

Arie Qur'ania, M.Kom.

ORIGINALITY REPORT

20%	19%	7%	8%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	dspace.uii.ac.id Internet Source	2%
2	eprints.unpak.ac.id Internet Source	2%
3	github.com Internet Source	1%
4	www.jptam.org Internet Source	1%
5	repository.teknokrat.ac.id Internet Source	1%
6	digilib.unila.ac.id Internet Source	1%
7	repository.uin-suska.ac.id Internet Source	1%
8	media.neliti.com Internet Source	1%
9	perpustakaan.fmipa.unpak.ac.id Internet Source	<1%

10	docplayer.info Internet Source	<1%
11	repository.upi.edu Internet Source	<1%
12	Submitted to Syiah Kuala University Student Paper	<1%
13	ojs.umrah.ac.id Internet Source	<1%
14	repository.unhas.ac.id Internet Source	<1%
15	download.garuda.kemdikbud.go.id Internet Source	<1%
16	Submitted to Defense University Student Paper	<1%
17	Submitted to Universitas Bengkulu Student Paper	<1%
18	semnas.univbinainsan.ac.id Internet Source	<1%
19	www.coursehero.com Internet Source	<1%
20	kc.umn.ac.id Internet Source	<1%
21	eprints.uad.ac.id Internet Source	<1%

22	ojs.unida.ac.id Internet Source	<1%
23	Submitted to Universitas Pendidikan Indonesia Student Paper	<1%
24	jurnal.ugm.ac.id Internet Source	<1%
25	www.researchgate.net Internet Source	<1%
26	repository.poliupg.ac.id Internet Source	<1%
27	www.scribd.com Internet Source	<1%
28	conference.binadarma.ac.id Internet Source	<1%
29	id.scribd.com Internet Source	<1%
30	dblp.dagstuhl.de Internet Source	<1%
31	text-id.123dok.com Internet Source	<1%
32	Submitted to Rochester Institute of Technology Student Paper	<1%

33	jurusan.tik.pnj.ac.id Internet Source	<1%
34	repository.unair.ac.id Internet Source	<1%
35	edoc.pub Internet Source	<1%
36	jurnal.unimed.ac.id Internet Source	<1%
37	ojs.unikom.ac.id Internet Source	<1%
38	Alif Musthofa, Majid Rahardi. "Perbandingan Algoritma Support Vector Machine dan K-Nearest Neighbors Pada Sinyal Tubuh Perokok", Indonesian Journal of Computer Science, 2023 Publication	<1%
39	Dadang Iskandar Mulyana, M Ainur Rofik. "Implementasi Deteksi Real Time Klasifikasi Jenis Kendaraan Di Indonesia Menggunakan Metode YOLOV5", Jurnal Pendidikan Tambusai, 2022 Publication	<1%
40	www.bagiteknologi.com Internet Source	<1%
41	Submitted to Universitas Budi Luhur Student Paper	<1%

42	repository.unim.ac.id Internet Source	<1%
43	jim.unisma.ac.id Internet Source	<1%
44	journal.widyatama.ac.id Internet Source	<1%
45	repositori.usu.ac.id:8080 Internet Source	<1%
46	repository.upnjatim.ac.id Internet Source	<1%
47	Submitted to Universitas Muria Kudus Student Paper	<1%
48	ecampus.pelitabangsa.ac.id Internet Source	<1%
49	Submitted to University of New Haven Student Paper	<1%
50	etheses.uin-malang.ac.id Internet Source	<1%
51	repository.uncp.ac.id Internet Source	<1%
52	Submitted to Tarumanagara University Student Paper	<1%
53	if.binadarma.ac.id Internet Source	<1%

54	Submitted to University of Paisley Student Paper	<1%
55	Submitted to University of Sheffield Student Paper	<1%
56	repository.unika.ac.id Internet Source	<1%
57	ebin.pub Internet Source	<1%
58	Submitted to Hialeah Gardens Senior High School Student Paper	<1%
59	Nizar Masmoudi, Wael Jaafar, Safa Cherif, Jihene Ben Abderrazak, Halim Yanikomeroglu. "UAV-based Crowd Surveillance in Post COVID-19 Era", IEEE Access, 2021 Publication	<1%
60	medium.com Internet Source	<1%
61	perpustakaan.jak-stik.ac.id Internet Source	<1%
62	repository.uinsu.ac.id Internet Source	<1%
63	root.cern Internet Source	<1%

64	Ahmad Kurniadi S.Kom. "Implementasi Convolutional Neural Network Untuk Klasifikasi Varietas Pada Citra Daun Sawi Menggunakan Keras", DoubleClick: Journal of Computer and Information Technology, 2020 Publication	<1%
65	dpmptsp.badungkab.go.id Internet Source	<1%
66	ejournal.ft-undar.ac.id Internet Source	<1%
67	ejournal3.undip.ac.id Internet Source	<1%
68	journal.unipdu.ac.id Internet Source	<1%
69	orizahasfi.blogspot.com Internet Source	<1%
70	repository.unpas.ac.id Internet Source	<1%
71	123dok.com Internet Source	<1%
72	Kadafi Eka Sakti, Mardiana Mardiana, Rio Ariestia Pradipta. "RANCANG BANGUN APLIKASI WEB PENDETEKSI WARNA PADA PIXEL GAMBAR DENGAN KNN CLASSIFIER",	<1%

	Jurnal Informatika dan Teknik Elektro Terapan, 2023 Publication	
73	core.ac.uk Internet Source	<1%
74	eprints.uny.ac.id Internet Source	<1%
75	idoc.pub Internet Source	<1%
76	jutif.if.unsoed.ac.id Internet Source	<1%
77	repository.ar-raniry.ac.id Internet Source	<1%
78	repository.uksw.edu Internet Source	<1%
79	repository.universitasbumigora.ac.id Internet Source	<1%
80	wahyurev07.blogspot.com Internet Source	<1%
81	www.csdn.net Internet Source	<1%
82	www.kompas.com Internet Source	<1%
83	www.mdpi.com Internet Source	<1%

84	M. Najamudin Ridha, Endang Setyati, Yosi Kristian. "Identifikasi Foto Wanita Berhijab dari Majalah Untuk Pembuatan Katalog Busana Muslim Otomatis Memanfaatkan Convolutional Neural Network", Journal of Intelligent System and Computation, 2019 Publication	<1%
85	Submitted to Universitas Nasional Student Paper	<1%
86	anzdoc.com Internet Source	<1%
87	bmadi.wordpress.com Internet Source	<1%
88	edoc.site Internet Source	<1%
89	ejournal.jak-stik.ac.id Internet Source	<1%
90	intheredwinebar.com Internet Source	<1%
91	jurnal.polgan.ac.id Internet Source	<1%
92	nurainiteti.blogspot.com Internet Source	<1%
93	pontianakpost.co.id Internet Source	<1%

94	pt.slideshare.net Internet Source	<1%
95	repository.usd.ac.id Internet Source	<1%
96	tytomulyono.com Internet Source	<1%
97	ejournal.antarbangsa.ac.id Internet Source	<1%
98	repository.its.ac.id Internet Source	<1%
99	journal.thamrin.ac.id Internet Source	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	On		