DETEKSI KENDARAAN TRUK PADA VIDEO MENGGUNAKAN METODE *TINY-YOLO V4*

Primasdika Yunia Putra[1], Aji Seto Arifianto[2], Zilvanhisna Emka Fitri[3], Trismayanti Dwi Puspitasari[4]

[1,2,3,4]Teknik Informatika, Jurusan Teknologi Informasi,
[1,2,3,4]Politeknik Negeri Jember
[1]dickayunia1@gmail.com, [2]ajiseto@polije.ac.id, [3]zilvanhisnaef@polije.ac.id, [4]trismayanti@polije.ac.id

Abstrak

Computer vision mampu meniru kemampuan manusia dalam hal mengenali sesuatu dengan ciri visual, dikarenakan bersifat konsisten. Deteksi kendaraan secara *real-time* dalam video pengamatan di jalan raya menjadi salah satu topik yang menarik diangkat dalam riset. Mendeteksi truk memiliki tantangan tersendiri karena memiliki kesamaan dimensi dengan bus dan struktur mekanis yang menyerupai pikap. Mendeteksi objek dengan metode *YOLO (You Only Look Once)* sangat populer. Langkah awal dilakukan percobaan dengan bobot dan model asli YOLO, namun hanya mampu mendeteksi bagian kepala dan bak truk saja, jika ada truk bermuatan maka *bounding box* tidak berhasil menandainya. Sehingga dilakukan *custom dataset* dengan cara membuat *bounding box* untuk seluruh bagian kepala, bak serta truk yang bermuatan. *Tiny-YOLO* sebagai varian dari YOLO dipilih karena strukturnya lebih sederhana dan kompatibel dengan perangkat *low-end* hingga *high-end*. *Tiny-YOLO v4* diimplementasikan pada 3 perangkat dengan spesifikasi berbeda, menghasilkan akurasi pengujian 98,2% (13FPS) pada perangkat A, 98% (28FPS) pada perangakt B dan 97,5% (38FPS) pada perangkat C. Pengaruh spesifikasi perangkat keras terhadap perolehan FPS yang didapat dan akurasi pendeteksian ditandai dengan perbedaan hasil yang signifikan pada saat pengujian. Semakin tinggi FPS maka semakin menurun akurasi pendeteksian. Untuk perangkat B dan C dengan spesifikasi CPU diatas 3.0 GHz, RAM minimum 16 GB, GPU lebih dari 4GB, dapat menjalankan pendeteksian truk menggunakan *Tiny-Yolo v4* secara *real-time* dengan baik.

Kata kunci : *YOLO, Tiny-YOLO,* Deteksi Truk.

1. Pendahuluan

Implementasi *computer vision* bertujuan meniru kemampuan visualisasi manusia ke dalam mesin/sistem komputer sehingga dapat mengenali atau mendeteksi citra baik yang statik maupun bergerak. *Computer vision* bisa menjadi sebuah solusi karena mampu bekerja lebih baik dan konsisten dari manusia (Prabowo et al., 2018). Deteksi objek menjadi topik yang selalu menarik untuk diangkat dalam sebuah riset tentang *computer visio*n, khususnya deteksi objek bergerak seperti kendaraan secara *realtime* pada data video.

Penelitian terdahulu seputar deteksi kendaraan diantaranya mengenali sepeda motor, mobil, bus dengan background subtraction Mixture of Gaussians 2 (Kurniawan et al., 2021). Karya ilmiah lain untuk mengenali jenis kendaraan untuk deteksi tingkat kepadatan lalu lintas(Fadhlan et al., 2021). Kedua penelitian tersebut fokus untuk mengklasifiksikan jenis kendaraan roda dua dan roda empat dengan *background subtraction* tetapi jenis kendaraan yang diklasifikasikan memiliki perbedaan ukuran dan bentuk yang menonjol. Kelemahan lainnya adalah jika semakin banyak objek yang terdeteksi, maka kinerja sistem dalam semakin menurun. Metode *haar cascade classifier* juga digunakan untuk deteksi kecepatan kendaraan roda empat: mobil, pikap, minibus, sedan (Zulfikri et al., 2021). Metode lain seperti *convolutional neural network* juga pernah digunakan untuk klasfikasi jenis kendaraan seperti mobil MVP, mini bus, bus dan pikap dengan akurasi hasil 81,9%(Tri Nurolan, 2019). Metode didua penelitian yang disebut terakhir belum diterapkan untuk mendeteksi kendaraan angkut barang berdimensi besar seperti truk.

Penelitian kali ini dilakukan untuk mendeteksi objek kendaraan dijalan raya yang belum banyak dilakukan penelitian lain yaitu untuk mendeteksi truk. Bedasarkan data BPS jumlah truk di Indonesia mencapai 5.438.475 sampai tahun 2021. Peningkatan 6,98% dibandingkan jumlah pada tahun sebelumnya. Jawa Timur merupakan provinsi dengan jumlah truk terbanyak, yaitu 762.410 unit (BPS, 2021). Fakta tersebut tentu menimbulkan berbagai kemungkinan resiko kecelakaan di jalan raya, seperti rilis Badan Kebijakan Transportasi Kementerian Perhubungan bahwa dari tahun 2007 hingga 2019 terdapat 92 laporan investigasi dimana 33 merupakan kecelakaan yang melibatkan kendaraan angkutan barang/truk (Tazkiyah, 2021).

Truk merupakan kendaraan bak terbuka yang digunakan untuk mengangkut barang, memiliki dua bagian yaitu bagian kepala dan bak. Kendaraan lain yang memiliki faktor kesamaan dengan truk adalah

bus dari sudut pandang dimensi dan mobil pikap dari struktur mekanis bak terbukanya. Data video yang digunakan dalam artikel ini merupakan hasil rekaman di jalan raya perbatasan Kabupaten Jember dan Banyuwangi provinsi Jawa Timur.

Salah satu pendekatan untuk pendeteksian objek secara *real-time* menggunakan metode *YOLO (You Only Look Once)*(Hutauruk et al., 2020). YOLO banyak digunakan untuk deteksi kendaraan, seperti untuk mendeteksi mobil, sepeda motor, truk, bus, dan becak dengan video CCTV di Medan, hasil mAP diatas 99% (Amwin, 2021). Penelitian dengan menggabungkan metode YOLO dan Faster R-CNN juga telah dilakukan untuk deteksi sedan, SUV, MPV, dan bus dengan tujuan meningkatkan akurasi, walaupun diakhir hanya memperoleh akurasi dibawah 80%(Shianto et al., 2019). Riset lain untuk deteksi jenis kendaraan mobil, motor, truk dan bus di lahan parkir dengan akurasi diatas 98%, hanya saja tidak dilakukan di jalan raya (Asni et al., 2021).

Merujuk pada penelitian-penelitian sebelumnya. Peneliti melakukan percobaan awal deteksi truk menggunakan YOLO dengan bobot dan model asli (*default*), model ini hanya mampu mendeteksi bagian kepala dan bak truk saja, jika ada truk bermuatan maka *bounding box* tidak berhasil menandainya. Kekurangan lain terjadi karena kendaraan pikap terdeteksi sebagai truk karena sama-sama memiliki bagian kepala dan bak. Berikutnya peneliti melakukan *custom dataset* dengan cara membuat *bounding box* untuk seluruh bagian kepala, bak truk serta truk yang bermuatan.

Metode YOLO dalam perkembangannya telah diturunkan menjadi beberapa versi seperti YOLO v1, *v1 tiny*, *v2*, *v2 tiny*, *v3, v3 tiny*(Adarsh et al., 2020), bahkan sampai pada *v4* dan *v4 tiny* (Wang et al., 2022). *Tiny-YOLO* dikembangkan untuk membuat struktur YOLO lebih sederhana dan mengurangi parameter sehingga layak untuk dikembangkan pada perangkat seluler dan perangkat tertanam. Metode ini dapat bekerja pada perangkat *low-end* hingga *high-end*. *Tiny-YOLO* telah mengalami kompresi proses (Techzizou, 2021). Hal yang juga penting untuk dipertimbangkan yaitu tingkat pemrosesan data video atau *frame-rate* agar aplikasi dapat berjalan secara *real-time*. Pemrosesan deteksi objek pada video *real-time* harus mampu mengolah data minimal 24 *frame per second* (FPS) (Iskandar Mulyana and Rofik, 2022), hasil lain menyebutkan YOLO v1 bekerja baik di frame-rate 45 FPS(Yuan, 2020).

Keterbatasan perangkat komputer merupakan hal lazim sehingga perlu juga dilakukan studi terhadap spesifikasi perangkat keras komputer yang sesuai untuk mengoperasikan aplikasi computer vision untuk deteksi kendaraan truk. Tujuan utama artikel ini untuk mengetahui hubungan antara FPS dan akurasi deteksi objek truk secara *real-time*. Sebagai pembanding dan memastikan kemampuan *computer vision* yang dikembangkan maka digunakan 3 kelas kendaraan yaitu bus, truk, dan pikap, serta tiga perangkat komputer yang berbeda spesifikasinya.

2. Dasar Teori

2.1 *Computer Vision*

Disampaikan oleh (Prabowo et al., 2018) *Computer vision* berkaitan dengan teori di balik sistem buatan terhadap ekstrak informasi dari citra atau gambar. Masukan gambar atau citra dapat diambil dari banyak bentuk, misalnya urutan video, pandangan dari beberapa kamera, atau citra multi-dimensi dari scanner medis(Prayitna et al., 2022). Suatu sistem dapat dikatakan *computer* vision ketika sistem tersebut menerapkan Pengolahan Citra Digital dan Sistem Cerdas, hal tersebut yang membuat sistem dapat menduplikasi penglihatan manusia. Contoh sitem/aplikasi *computer vision* antara lain:

a. Pengendalian proses (contohnya, sebuah robot industri atau kendaraan).
b. Mendeteksi peristiwa (contohnya, untuk pengawasan visual).
c. Mengorganisir informasi (contohnya, untuk pengindeksan *database* foto).
d. Modeling benda atau lingkungan (contohnya, inspeksi industri, analisis citra).
e. Interaksi (contohnya, sebagai input untuk interaksi komputer-manusia).

Penelitian ini termasuk pada aplikasi *computer vision* yang dapat mendeteksi objek, sesuai fungsi pada *output* penelitian dimana sistem dapat mendeteksi truk beserta muatannya dalam citra video secara otomatis.

2.2 Konvolusi Citra

Konvolusi citra merupakan konvolusi dikrit pada dua dimensi(Affifah et al., 2022). Definisi secara matematis ditunjukkan pada persamaan 2.1 berikut:

$$I'(u,v) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} I(u+i, v+j) \cdot H(i,j) \quad (1)$$

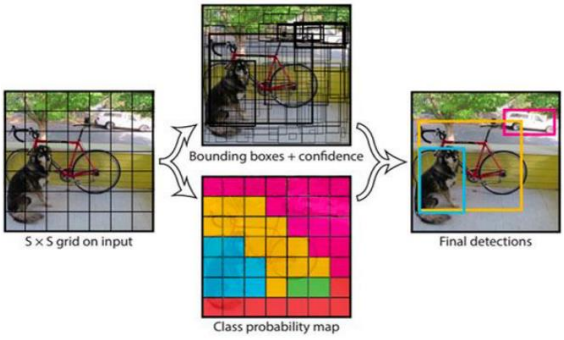
Dimana:
I'(u,v): output piksel di koordinat *(u,v)*
I(u,v): input piksel di koordinat *(u,v)*
H(i,j): filter kernel berukuran $i \times j$

Citra masukan *(input)* dan citra keluaran *(output)* terdiri dari *width*(lebar) × *height*(tinggi) piksel. Konvolusi citra menggunakan *filter kernel* yang dilakukan dengan mengalikan citra masukan dengan *filter kernel*, proses tersebut dilakukan secara *elementwise*. Piksel yang sedang dihitung, *I'(u,v),* merupakan jumlah semua piksel *I(u,v)* pada citra masukan yang telah dikalikan dengan koefisien pada *filter kernel H*.

2.3 *You Only Look Once*

Menurut (Hutauruk et al., 2020) *YOLO (You Only Look Once)* adalah sebuah pendekatan baru untuk sistem pendeteksian objek, yang ditargetkan untuk pemrosesan secara real-time. *YOLO*

membingkai pendeteksian objek sebagai masalah regresi tunggal, dimana dari piksel citra langsung ke *bounding box* spasial yang terpisah dan probabilitas kelas yang terkait. *YOLO* menjalankan pendeteksian dan pengenalan objek dengan sebuah jaringan syaraf tunggal, yang memprediksi *bounding box* dan probabilitas kelas (*class*) secara langsung dalam satu evaluasi. Dalam karya tulis (Adiwibowo et al., 2020) disampaikan bahwa YOLO memiliki tugas untuk membagi gambar menjadi *grid* berukuran S x S. Jika suatu objek berada dalam sel *grid*, sel tersebut bertanggung jawab untuk memprediksi objek tersebut, setiap *grid* sel memprediksi B *(Bounding Boxes)* dan nilai daripada *Bounding Boxes* tersebut. Diagram Algoritma YOLO disampaikan pada Gambar 2.1



Gambar 2.1 Diagram Algoritma YOLO

Jika pusat suatu objek jatuh ke dalam sel grid, maka sel grid bertanggung jawab untuk mendeteksi objek tersebut. Setiap sel grid memprediksi kotak pembatas B dan nilai keyakinan (Box confidence scores) untuk kotak tersebut, serta probabilitas kelas kondisional C. Prediksi kotak pembatas B memiliki 5 komponen x, y, w, h, dan Box confidence score. Koordinat (x, y) mewakili pusat kotak, relatif terhadap batas-batas kotak grid(Amwin, 2021). Normalisasi diterapkan pada koordinat (x, y) ini agar titik tersebut jatuh di antara 0 dan 1. Dimensi kotak (w, h) relatif terhadap ukuran gambar, dan juga dinormalisasikan(Rahma et al., 2021). *Box confidence score* (nilai keyakinan) mencerminkan seberapa yakinnya bobot, bahwa kotak pembatas B berisi suatu objek dan seberapa akurat menurutnya bahwa kotak itu yang ia prediksi. Oleh karena itu, prediksi YOLO memiliki bentuk vektor *output [S, S, B×5+C]*.

2.4 *Bounding Box*

Menurut (Putra et al., 2021) *Bounding Box* merupakan kotak imajiner tidak nyata yang mengelilingi objek yang teridentifikasi. *Bounding Box* sendiri berbentuk kotak yang mana besarnya sama seperti besar objek yang teridentifikasi tersebut. Untuk membuat *Bounding Box* dibutuhkan koordinat piksel objek *upper-left* (UL), *upper-right* (UR), *lower-left* (LL), dan *lower-right* (LR).

2.5 *Confidence Score*

Dijelaskan pada karya tulis oleh (Hutauruk et al., 2020b) YOLO mendefinisikan *confidence score* untuk setiap kotak B sebagai nilai probabilitas kotak berisi objek yang dikalikan dengan IoU *(intersection over union)* antara kotak prediksi dan kebenaran dasar (*ground truth*), yang mana kebenaran dasar didapat selama masa pelatihan (*training*). IoU adalah pengukuran evaluasi yang digunakan untuk mengukur seberapa akuratnya pendeteksi objek pada suatu *dataset*. Perhitungan *Box confidence score* atau *nilai confidence* untuk sebuah kotak pembatas dipaparkan pada persamaan 2 dan persamaan 3.

$$C = P_r(object) \,.\, IoU_{pred}^{truth} \quad (2)$$

$$IoU_{pred}^{truth} = \frac{Area\ of\ Overlap}{Area\ of\ Union} \quad (3)$$

Dimana:
C : Box Confidence Score
$P_r(object)$: probabilitas kotak berisi objek, jika ada bernilai 1,jika tidak bernilai 0.
IoU_{pred}^{truth} : *IoU* Antara kotak prediksi dan kebenaran dasar.

Faktor penentu untuk mendapatkan prediksi final adalah *Class confidence score* yang didapat, berdasarkan probabilitas kondisional *class* dan *Box confidence score.* Nilai kepercayaan pada klasifikasi dan lokalisasi objek diukur oleh *Class confidence score. Class confidence score* memberi nilai kepercayaan kelas secara spesifik untuk setiap kotak, hal tersebut mengkodekan kemungkinan kelas yang muncul di kotak dan seberapa sesuai kotak yang diprediksi dengan objek. Persamaan pada *Class confidence score* untuk setiap kotak prediksi ditunjukkan pada persamaan 4.

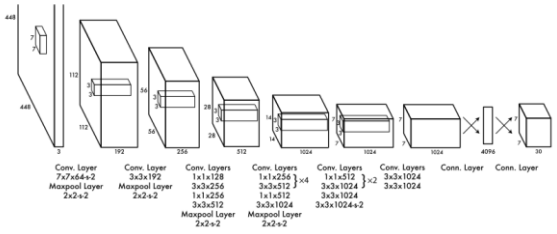
$$P_r(Class_i|object) \cdot box\ confidence\ score = P_r(Class_i) \cdot IoU_{pred}^{truth} \quad (4)$$

Dimana:
$P_r(Class_i|object)$ = probabilitas kondisional kelas *i*
$P_r(Class_i)$ = probabilitas kelas *i*

2.6 Arsitektur *YOLO*

Arsitektur *YOLO* seperti yang ditampilkan pada Gambar 2.2 terdiri atas 24 lapisan konvolusional (*convolutional layer*) dengan 4 lapisan *max pooling*, yang diikuti oleh 2 lapisan yang terhubung penuh *(fully connected layer).* Beberapa lapisan konvolusi menggunakan lapisan reduksi 1×1 sebagai alternatif untuk mengurangi kedalaman *feature maps*. Arsitektur yang diperkenalkan oleh Joseph Redmon dalam (Hutauruk et al., 2020). Arsitektur *YOLO* sesungguhnya cukup sederhana. Sistem akan menerima input citra dengan bentuk (448, 448, 3)

yaitu citra berukuran 448 x 448 dengan 3 *channel*, yang kemudian akan melewati satu kali proses *convolutional network* hingga menghasilkan *output* dengan bentuk (7, 7, 30), dimana 7 x 7 merupakan ukuran *grid* sel (S=7) dan 30 merupakan nilai dari jumlah kotak pembatas B yang dikali dengan penjumlahan antara jumlah kelas dan jumlah komponen dalam satu kotak B (B×5+C, B = 2, C=20).



Gambar 2.2 Arsitektur *YOLO*

Setiap proses konvolusi, selain memiliki ukuran *kernel filter* dan jumlah *filter*, juga terdapat parameter lain yang mempengaruhi bentuk dari *output* hasil operasi konvolusi, yaitu *padding* dan *stride*. *Padding* merupakan parameter yang menyatakan jumlah penambahan *border* di seluruh tepian dari input, yang berfungsi meminimalisir kehilangan informasi pada tepian citra. Hal tersebut terjadi dikarenakan proses konvolusi itu sendiri, dimana bagian tepi dari citra akan terlewati oleh *kernel filter*, kecuali pada *kernel* berukuran 1x1. Metode populer dan paling sederhana untuk mengatasi masalah ini adalah penggunaan *zero- padding*, *zero-padding* memberi nilai 0 pada setiap tepian citra (*input*).

Sedangkan *stride* (Affifah et al., 2022) adalah parameter yang menyatakan jumlah pergeseran (langkah) yang dilakukan oleh *kernel filter*. Parameter ini biasa digunakan untuk mereduksi ukuran keluaran (*output*). Ukuran keluaran hasil dari operasi konvolusi dapat dihitung dengan menggunakan persamaan 5

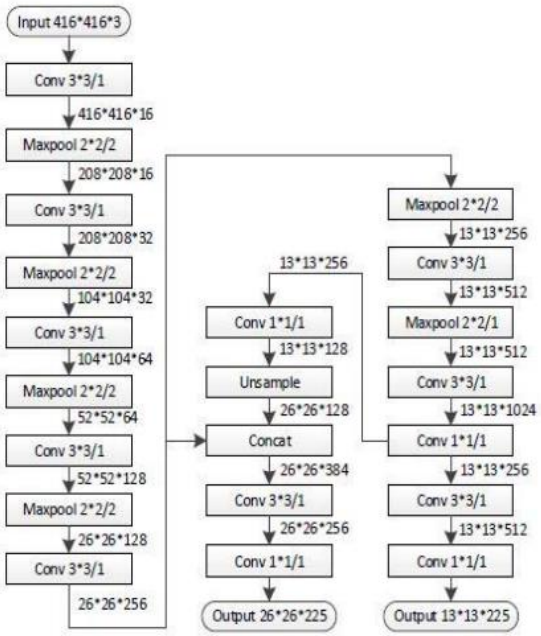
$$O = 1 + \frac{N+2P-F}{S} \quad (5)$$

Dimana:
O : ukuran output yang didapat setelah konvolusi.
N : ukuran input.
P : jumlah padding.
F : ukuran kernel filter.
S : jumlah stride.

2.7 *Tiny-YOLO*

Sejarah penemuan *Tiny-YOLO* berawal dari penyampaian *Fast YOLO* oleh Joseph Redmon pada tahun 2016. Dikutip dari karya tulis milik (Shinde et al., 2019) *Fast YOLO* menggunakan sebuah jaringan syaraf dengan lapisan konvolusional yang lebih sedikit (9 alih – alih 24) dan menggunakan lebih sedikit *filter* pada setiap lapisan konvolusional tersebut. Selain ukuran jaringan, semua parameter pelatihan dan pengujian adalah sama antara *YOLO* dan *Fast YOLO*. Pada pengembangan *Yolo v3* maka ditemukanlah varian terkompresi dari metode *YOLO* yakni *Tiny-YOLO*.

Menurut(Adarsh et al., 2020) *Tiny-YOLO* merupakan versi *YOLO* yang memiliki ketebalan lapisan konvolusional lebih sedikit. Maka dari itu, kecepatan pendeteksian meningkat secara signifikan, sekitar 442% lebih cepat daripada varian sebelumnya pada *YOLO* namun akurasi yang didapatkan lebih kecil. *Tiny-YOLO* menggunakan *pooling layer* dan mengurangi gambar untuk lapisan konvolusional, Memprediksi sebuah *three-dimensional tensor* yang berisi *objectness score*, *bounding box*, dan *class predictions* pada dua skala yang berbeda. Hal tersebut membagi gambar menjadi sel *grid* S x S. Untuk deteksi akhir, *Tiny-YOLO* mengabaikan *bounding box* yang memiliki *objectness score* bukan yang terbaik. Untuk mengekstrak fitur, lapisan konvolusi dan lapisan *max-pooling* digunakan dalam pengaturan *feed forward Tiny-YOLO*. Prediksi *bounding box* terjadi pada dua skala peta fitur yang berbeda, yaitu 13x13 dan 26x26. Arsitektur Tiny-YOLO dipaparkan pada Gambar 2.3



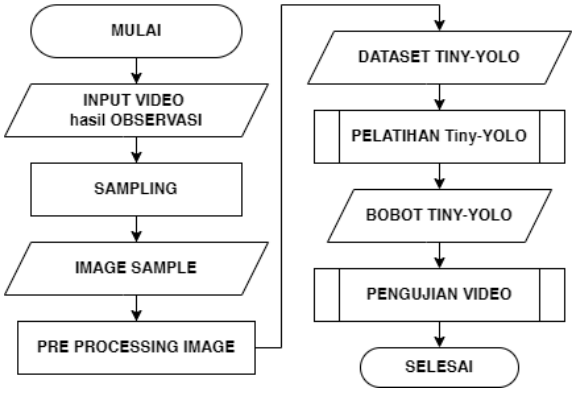
Gambar 2.3 Arsitektur *Tiny-YOLO*

3. Desain Sistem

3.1 Analisis Sistem

Flowchart pada Gambar 3.1 menjelaskan alur sistem secara garis besar akan menerima masukan berupa video observasi, lalu akan dilakukan sampling, *pre-processing image*, *training* dan terakhir *testing*. Video observasi merupakan data primer, diambil pada 4 tempat berbeda dengan total 132 video. Pada tahap sampling, penulis menerapkan teknik *Simple Random Sampling* (Khanifudin et al., 2020) untuk memilih 1 *frame* gambar pada video

setiap detik video yang diputar, dan *Stratified Random Sampling* dengan cara menyeleksi gambar yang mengandung citra truk/pikap/bus. Pemberian label dan anotasi diberikan pada *image sample* dalam tahap *pre-processing image*.



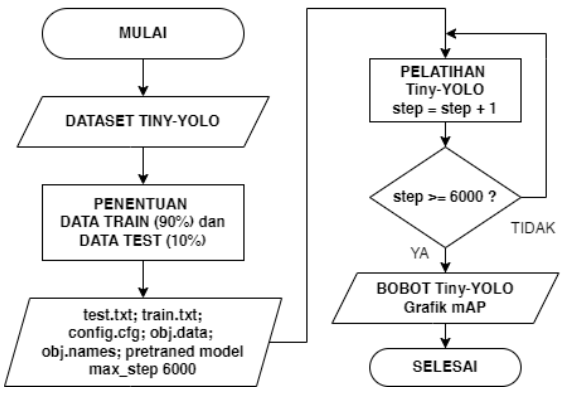
Gambar 3.1 Alur Sistem

3.2 Alur Pelatihan *Tiny-Yolo*

Dataset menjadi masukan utama pada proses pelatihan bobot *Tiny-Yolo*, dari keseluruhan *image sample* pada *dataset*, akan dibagi menjadi data *train* dan data *test*. Lalu atribut yang menjadi konfigurasi utama pelatihan akan diinisialisasi terlebih dahulu. Atribut yang diperlukan dijelaskan pada Tabel 3.1.

Tabel 3.1 Tabel Atribut Konfigurasi Pelatihan

Jml gambar	654
Jml train	589 (90%)
Jml test	65 (10%)
Jml class	3 (truk, bus, pikap)
Jml iterasi	6000 => berasal dari (jml class x 2000)
Filters	24 => berasal dari (jml class + 5) x 3
Network size	416 x 416
Batch	64 (proses latih 64 gambar dalam 1 iterasi)
Subdivision	8 (proses latih 8 gambar per 1 proses GPU)

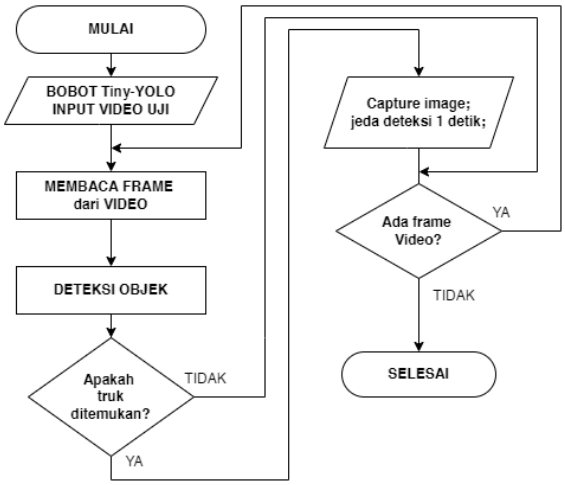


Gambar 3.2 Alur Pelatihan Tiny-Yolo

Perulangan iterasi terus dijalankan hingga *step* mencapai 6000. Bobot terbaik *Tiny-Yolo* didapatkan setelah pelatihan selesai. Seluruh pelatihan dilakukan menggunakan bantuan Google Colab. *Flowchart* alur pelatihan *Tiny-Yolo* dijelaskan pada Gambar 3.2

3.3 Alur Pengujian *Tiny-Yolo*

Pengujian video dilakukan pada semua video, baik video pelatihan dan video uji. Dimulai dengan memuat bobot *Tiny-Yolo* dan video, lalu sistem membaca setiap *frame* dari video selama *frame* masih didapatkan. Jika *frame* didapatkan, sistem akan mendeteksi objek berdasarkan *class* (truk/bus/ *pikap*). Jika ditemukan truk, maka sistem akan menangkap dan menyimpan *frame* truk tersebut. Alur pengujian dijelaskan pada Gambar 3.3.



Gambar 3.3 Alur Pengujian Video

4. Implementasi

Implementasi Sistem dilakukan pada tiga perangkat komputer dengan spesifikasi perangkat yang disampaikan pada Tabel 4.1.

Tabel 4.1 Spesifikasi Perangkat

	Perangkat A	Perangkat B	Perangkat C
Nama	Laptop Acer E5	Lenovo Legion 5	PC Dell Precision 3640 Tower
CPU	Intel® Core™ i5-7200U CPU @ 2.50GHz (4 CPUs)	AMD Ryzen5 4600H @ 3.0 GHz (12 CPUs)	Intel® Core™ i7-9700K @ 3.6 GHz (8 CPUs)
RAM	8GB	16GB	16GB
GPU	Nvidia GeForce 940MX 2GB VRAM	Nvidia GeForce GTX 1650 4GB VRAM	Nvidia Quadro P1000 4GB VRAM

Implementasi pengkodean sistem menggunakan bahasa pemrograman *Python* dengan versi 3.95 64 bit yang dijalankan pada *PyCharm Community*. Adapun beberapa *library* yang mendukung sistem ini, antara lain 1) *Darknet*; 2) *OpenCV*; 3) *Tkinter*; 4) *Numpy*.

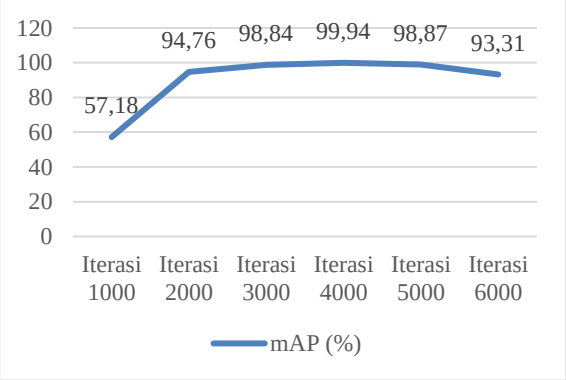
5. Analisa dan Pengujian

5.1 Hasil Pelatihan *Tiny-Yolo*

Berdasarkan pelatihan menggunakan konfigurasi dan *dataset* yang telah dibuat sebelumnya, menghasilkan bobot terbaik dengan detail hasil pelatihan dijelaskan pada Tabel 5.1 dan mendapatkan grafik performa pelatihan yang dipaparkan pada Gambar 5.1

Tabel 5.1 Hasil Pelatihan Tiny-Yolo

Jml *Interrupt*	2 kali (dikarenakan sesi habis dan koneksi internet *off*)
mAP	mAP@0.50 = 93.31%. dengan AP pada setiap class: AP class truk = 99,94 % AP class bus = 80,00 % AP class pikap = 100%
Kecepatan deteksi	Deteksi video memiliki kecepatan deteksi rata rata yakni 37.6 FPS dengan *resource RAM* dan GPU *Google Colab*



Gambar 5.1 Grafik Performa Pelatihan

5.2 Hasil Pengujian *Tiny-Yolo*

Berdasrkan bobot Tiny-Yolo yang telah diperoleh, bobot tersebut dimuat untuk digunakan pada pengujian video, serta didapatkan kumpulan *frame*/gambar truk yang ditangkap oleh sistem yang dijelaskan pada Tabel 5.2.

Tabel 5.2 Tabel *capture frame* sistem

	Jml Video	Jml *Capture*	*True*	*False*	(%)
Seg1	24	195	183	12	**93,84**
Seg2	22	68	67	1	**98,52**
Seg3	24	182	178	4	**97,80**
Seg4	33	96	93	3	**96,87**
Uji	13	113	111	2	**98,23**
Total	**116**	**654**	**633**	**21**	**96,78**

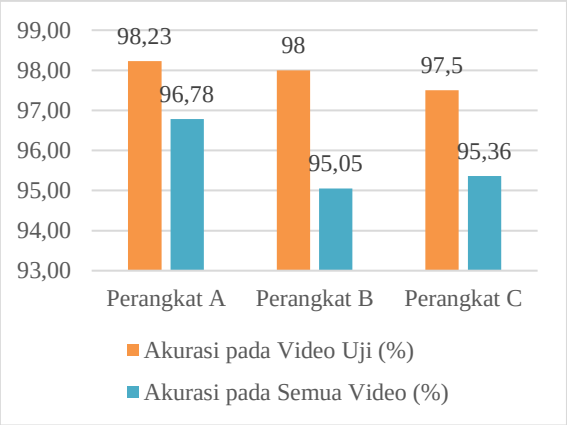
Penjelasan Tabel:

a. **Seg1 :** Kelompok video latih pada 2021/08/16
b. **Seg2 :** Kelompok video latih pada 2022/02/08
c. **Seg3 :** Kelompok video latih pada 2022/02/09
d. **Seg4 :** Kelompok video latih pada 2022/02/12
e. **Uji :** Kelompok video uji
f. ***True* :** Gambar yang dihasilkan mengandung truk
g. ***False* :** Gambar yang dihasilkan bukan truk / bagian belakang truk
h. **Jml *Capture*:** jumlah gambar yang diambil oleh sistem

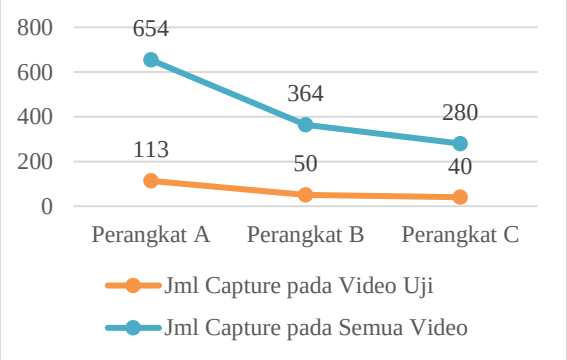
Hasil deteksi truk pada saat pengujian dapat dilihat pada gambar 5.2. Truk dapat dideteksi dengan baik secara keseluruhan yakni kepala truk, bak truk dan muatan truk.

Gambar 5.2 Hasil Deteksi Truk.

Pengujian bobot terbaik *Tiny-Yolo* dilakukan pada 3 spesifikasi perangkat berbeda dan didapatkan hasil uji coba dalam bentuk grafik yang dapat dilihat pada Gambar 5.3. Uji Bobot *Tiny-Yolo* menghasilkan bagan jumlah *capture* dengan garis menurun dari perangkat A ke perangkat C pada Gambar 5.4. Hal ini menandakan Perangkat A menghasilkan lebih banyak *capture* dikarenakan *load* dan deteksi *Tiny-Yolo* pada Perangkat A membutuhkan waktu lebih lama dibandingkan Perangkat B dan C.

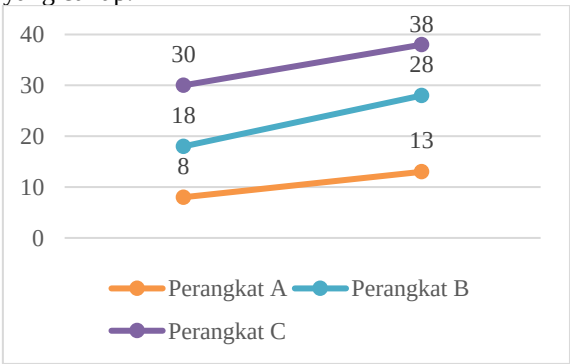


Gambar 5.3 Hasil akurasi uji Tiny-Yolo



Gambar 5. 4 Jumlah *capture* sistem

Nilai FPS pada setiap percobaab dan di masing-masing perangkat memiliki variasi yang dipengaruhi oleh jumlah objek kendaraan dalam video. Perangkat A memiliki rentang nilai FPS yang rendah dibandingkan Perangkat B dan C. Gambar 5.5 menunjukkan nilai manimum dan maksimum FPS dimasing-masing perangkat. Spesifikasi perangkat sangat mempengaruhi perolehan FPS dikarenakan untuk melakukan *render* sekaligus deteksi objek pada setiap *frame* dalam video memerlukan komputasi CPU yang besar dan dukungan kartu grafis *(GPU)* yang cukup.



Gambar 5.5 Rentang Nilai FPS

Analisa berdasarkan hal tersebut, performa CPU, RAM dan GPU Perangkat A kurang baik untuk untuk menjalankan aplikasi deteksi truk secara *real-time*, sedangkan Perangkat B (spesifikasi CPU 3,0 GHz, RAM 16GB dan 12 GPU) dan Perangkat C (spesifikasi CPU 3,6 GHz, RAM 16GB dan 8 GPU) sangat baik untuk menjalankan pendeteksian truk menggunakan *Tiny-Yolo* secara *real-time*.

6. Kesimpulan

Kesimpulan yang dapat diambil dari penelitian ini adalah pelatihan *Tiny-Yolo* menggunakan *custom dataset* dengan tiga *class* yakni (truk, bus, pikap) menghasilkan mAP 93,31%. Pengujian video menghasilkan akurasi tertinggi 98,23% pada perangkat A namun hanya dengan kecepatan deteksi 13 fps kurang cepat untuk aplikasi *real-time*. Pengaruh spesifikasi perangkat keras terhadap perolehan fps yang didapat dan akurasi pendeteksian ditandai dengan perbedaan hasil yang signifikan pada saat pengujian. Semakin tinggi fps maka semakin menurun akurasi pendeteksian, hal tersebut sesuai dengan grafik performa *mAP* yang disampaikan oleh (Techzizou, 2021). Sedangkan Perangkat B dan C dengan akurasi data uji 97,5%-98% lebih sesuai untuk deteksi *real-time* (28-38 FPS)

Untuk pengembangan selanjutnya (*future work*) akan dilakukan deteksi truk *over dimension*. Dapat ditambah *dataset* yang lebih bervariasi khususnya untuk *angle* pengambilan gambar truk.

Daftar Pustaka:

Adarsh, P., Rathi, P., Kumar, M., 2020. YOLO v3-Tiny: Object Detection and Recognition using one stage improved model. In: 2020 6th International Conference on Advanced Computing and Communication Systems, ICACCS 2020.

Adiwibowo, J., Gunadi, K., Setyati, E., 2020. Deteksi Alat Pelindung Diri Menggunakan Metode YOLO dan Faster R-CNN. J. Infra 18, 106–112.

Affifah, D.D., Permanasari, Y., Matematika, P., Matematika, F., Alam, P., Bandung, U.I., 2022. Teknik Konvolusi pada Deep Learning untuk Image Processing. In: Bandung Conference Series: Mathematics. UPT Publikasi Ilmiah, Universitas Islam Bandung, Bandung, pp. 103–112.

Amwin, A., 2021. DETEKSI DAN KLASIFIKASI KENDARAAN BERBASIS ALGORITMA YOU ONLY LOOK ONCE (YOLO) DETEKSI DAN KLASIFIKASI KENDARAAN BERBASIS ALGORITMA YOU ONLY LOOK ONCE (YOLO). UNIVERSITAS ISLAM INDONESIA.

Asni, A.B., Amin, K, M.W., 2021. Penerapan Metode Yolo Object Detection V1 Terhadap Proses Pendeteksian Jenis Kendaraan Di Parkiran. J. Tek. elektro UNIBA 6, 194–199.

BPS, 2021. Jumlah Kendaraan Bermotor Menurut Provinsi dan Jenis Kendaraan (unit), 2021 [WWW Document]. Badan Pus. Stat. URL

https://www.bps.go.id/indikator/indikator/view_data_pub/0000/api_pub/V2w4dFkwdFNLNU5mSE95Und2UDRMQT09/da_10/1

Fadhlan, M.Y., B. Hanafi, U., Aulia, M.R., 2021. Implementasi algoritma pendeteksi tingkat kepadatan lalu lintas menggunakan metode background subtraction. JITEL (Jurnal Ilm. Telekomun. Elektron. dan List. Tenaga) 1.

Hutauruk, J.S.W., Matulatan, T., Hayaty, N., 2020. Deteksi Kendaraan secara Real Time menggunakan Metode YOLO Berbasis Android. J. Sustain. J. Has. Penelit. dan Ind. Terap. 9, 8–14.

Iskandar Mulyana, D., Rofik, M.A., 2022. Implementasi Deteksi Real Time Klasifikasi Jenis Kendaraan Di Indonesia Menggunakan Metode YOLOV5. J. Pendidik. Tambusai 6, 13971–13982.

Khanifudin, K., Jajang, J., Guswanto, B.H., 2020. APLIKASI BAHASA C++ DAN PHP UNTUK MENENTUKAN UKURAN SAMPEL PADA METODE STRATIFIED RANDOM SAMPLING. J. Ilm. Mat. dan Pendidik. Mat. 11.

Kurniawan, L.A., Bayupati, I.P.A., Suar Wibawa, K., 2021. Sistem Hitung Kendaraan Berdasarkan Jenis Menggunakan Metode Background Subtraction. JITTER J. Ilm. Teknol. dan Komput. 1, 265–273.

Prabowo, D.A., Abdullah, D., Manik, A., 2018. Deteksi dan Perhitungan Objek Berdasarkan Warna Menggunakan Color Object Tracking. Pseudocode 5, 85–91.

Prayitna, D.H., Djajadi, A., Teknologi, M., Universitas, I., Alamat, P., Business, S., Sangereng, C., Dua, K., Sekolah, A., 2022. PERANCANGAN PROTOTYPE DETEKSI KELENGKAPAN. J. Inov. Inform. 7, 57–69.

Putra, B., Pamungkas, G., Nugroho, B., Anggraeny, F., 2021. Deteksi Dan Menghitung Manusia Menggunakan 02, 67–76.

Rahma, L., Syaputra, H., Mirza, A.H., Purnamasari, S.D., 2021. Objek Deteksi Makanan Khas Palembang Menggunakan Algoritma YOLO (You Only Look Once). J. Nas. Ilmu Komput. 2.

Shianto, K.A., Gunadi, K., Setyati, E., 2019. Deteksi Jenis Mobil Menggunakan Metode YOLO Dan Faster R-CNN. J. Infra 7.

Shinde, P., Yadav, S., Rudrake, S., Kumbhar, P., 2019. Smart Traffic Control System Using YOLO. Int. Res. J. Eng. Technol. 6, 169–172.

Tazkiyah, 2021. PERILAKU PENGEMUDI TRUK DAN KONTRIBUSINYA TERHADAP KECELAKAAN [WWW Document]. Badan Kebijak. Transp. Kementeri. Perhub. URL https://baketrans.dephub.go.id/berita/perilaku-pengemudi-truk-dan-kontribusinya-terhadap-kecelakaan#!

Techzizou, 2021. YOLOv4 vs YOLOv4-tiny [WWW Document]. medium.com.

Tri Nurolan, A., 2019. Deteksi Dan Klasifikasi Jenis Kendaraan Berbasis Pengolahan Citra Dengan Metode Convolutional Neural Network (Cnn). Deteksi Jenis Kendaraan.

Wang, J., Gao, Z., Zhang, Y., Zhou, J., Wu, J., Li, P., 2022. Real-time detection and location of potted flowers based on a ZED camera and a YOLO V4-tiny deep learning algorithm. Horticulturae 8.

Yuan, Y., 2020. YOLO creator Joseph Redmon stopped CV research due to ethical concerns. Medium.

Zulfikri, M., Latif, K.A., Hairani, H., Ahmad, A., Hammad, R., Syahrir, M., 2021. Deteksi dan Estimasi Kecepatan Kendaraan dalam Sistem Pengawasan Lalu Lintas Menggunakan Pengolahan Citra. Techno.Com 20, 455–467.