Penerapan Convolutional Neural Network untuk Deteksi Pedestrian pada Sistem Autonomous Vehicle

Renjira Naufhal Dhiaegana
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Bandung, Indonesia
naufhalrenjira@gmail.com

Dr. Ir. Rinaldi Munir, M.T.
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Bandung, Indonesia
rinaldi@informatika.org

***Abstrak*—Deteksi *pedestrian* adalah salah satu dari banyak tugas yang harus diselesaikan oleh sistem autonomous vehicle. Saat ini, penggunaan *Convolutional Neural Network* (CNN) sangat mempengaruhi kinerja pendeteksian objek. Pengorbanan akurasi dan kecepatan adalah masalah utama dalam deteksi objek *real-time* karena berkurangnya resolusi input ke model menghilangkan fitur objek. Untuk mengatasi masalah tersebut, dilakukan deteksi ulang area yang mengandung sebagian besar objek pedestrian, terutama di bagian tengah citra. Hasil bagian gambar yang dideteksi ulang digabungkan dengan deteksi aslinya. Deteksi duplikat dihilangkan dengan melakukan *Non Maximum Suppression* (NMS). Model yang digunakan adalah YOLOv4 dan YOLOv4-*tiny*, dilatih terhadap *dataset CrowdHuman* yang diuji untuk menghasilkan hasil yang *state-of-the-art* (mAP tinggi dan kecepatan *real-time*) untuk kasus deteksi objek manusia. Pertama, model diuji dan dilatih menggunakan dataset *Crowdhuman* untuk menemukan model terbaik dari pelatihan. Setelah itu, model terbaik digunakan untuk menguji kinerja sistem deteksi ganda, dan juga dibandingkan tanpa menggunakan sistem tersebut. Penggunaan model tunggal dibandingkan dengan beberapa kombinasi model dengan konfigurasi berbeda untuk melihat efek deteksi ulang atau deteksi *double*. Hasil pengujian menunjukkan bahwa deteksi *double* memberikan dampak yang cukup baik untuk pendeteksian dengan model *tiny*. Model YOLOv4-*tiny*-416-double mencapai 69,02 mAP dengan kecepatan 41 FPS, sedikit peningkatan pada mAP dan peningkatan kecepatan sebanyak 13,13 persen dibandingkan dengan deteksi tunggal YOLOv4-320 yang mencapai 68,86 mAP dengan kecepatan 36,24 FPS yang berjalan di GPU Nvidia Tesla T4.**

Kata Kunci—Deteksi Objek Real-time, Deteksi Double, CNN, YOLO, mAP, FPS

I. PENDAHULUAN

Computer Vision sebagai salah satu bagian dari inteligensi buatan memiliki popularitas yang tidak kalah dengan bidang lainnya. Pengembangan yang merupakan fokus utama saat ini adalah fungsionalitas dari vehicle guidance system, yakni sebuah kendaraan dikendalikan dengan sistem yang dapat memilih aksi dari kondisi lingkungan saat ini [1]. Salah satu kegunaan *computer vision* dalam *autonomous vehicle* adalah mendeteksi *pedestrian* yang menjadi entitas pengguna jalan raya. Kenakalan pejalan kaki di seluruh dunia adalah masalah utama yang tak kunjung henti. Banyak sekali pengguna jalan yang mementingkan diri sendiri, yang menyebabkan meningkatnya risiko kecelakaan di jalan raya. Menurut WHO (*World Health Organization*) pada tahun 2013, kematian *pedestrian* dari seluruh dunia mencapai 22 persen dari total kecelakaan lalu lintas yang ada. Salah satu kasus kecelakaan terkait autonomous vehicle adalah kasus mobil otonom *Uber* yang menabrak pejalan kaki yang tengah menyebrang pada malam hari. Kecelakaan ini membuat *Uber* harus memberhentikan percobaan mobil otonom di jalan umum [2].

Hasil analisis [3] menunjukkan bahwa sensor pendeteksi memiliki kemampuan untuk mengurangi tumbukan bukan fatal dari <30% hingga >90%. Solusi naif adalah menggunakan seluruh teknologi untuk mengurangi resiko, tetapi akan mengeluarkan biaya yang besar hanya untuk satu sistem saja. Oleh karena itu, dilakukan pengembangan lebih lanjut terkait deteksi *pedestrian* hanya dengan menggunakan komponen penting saja (kamera). Idealnya, sistem *autonomous vehicle* harus dapat mendeteksi *pedestrian* secepat manusia mendeteksi *pedestrian* saat mengedarai mobil. Umumnya manusia dapat mempersepsi gerak objek secara baik pada video yang berkecepatan 15 citra per detik. Kecepatan lebih akan menghasilkan efek kontinuitas visual yang menyebabkan penglihatan terhadap objek menjadi buram atau *blurry* [4].

CNN sudah banyak digunakan pada pendeteksi objek. Sejumlah diantaranya adalah SSD, YOLO, dan Faster R-CNN. SSD [5] menggunakan pendekatan berbasis CNN yang mendapatkan mAP sebesar 74.3% dengan kecepatan 59 *frame per second* (FPS) pada Nvidia Titan X untuk data *test* VOC2007. Di sisi lain, pendekatan *Faster R-CNN* sudah digunakan dengan tambahan informasi semantik menggunakan *semantic segmentation network* [6]. Pendekatan ini mencapai *miss rate* sebesar 5.7% dengan kecepatan 0.32 detik per citra. Pendekatan menggunakan *Faster R-CNN* merupakan *two-stage object detector* sedangkan SSD merupakan *single-stage*. Dilihat dari kecepatan pemrosesannya, untuk selanjutnya akan dipilih *single-stage object detector*.

Untuk meningkatkan akurasi model pendeteksi, digunakan deteksi ganda atau *double detection* untuk mendeteksi objek jarak dekat dan jarak jauh. Dengan menggunakan metode ini,

diharapkan model yang menggunakan potongan citra jarak jauh dapat melihat objek-objek yang akan dideteksi, yakni *pedestrian*. Penglihatan lebih jelas diharapkan karena model, konfigurasi, beserta potongan citra dapat menyimpan fitur objek yang berukuran kecil pada citra yang diproses.

II. PENELITIAN TERKAIT

A. Pembangunan Dataset Baru Crowdhuman

CrowdHuman [7] adalah dataset yang mengutamakan pendeteksian pedestrian yang ada di kerumunan. Dataset *CrowdHuman* mengandung sebanyak 470.000 sampel citra orang dari bagian data latihan dan validasi, dengan rata-rata 22,6 orang per citra dan sejumlah oklusi pada dataset. Perbedaan dataset *CrowdHuman* dengan dataset pedestrian lain seperti Caltech, KITTI, CityPersons, dan COCOPersons terdapat pada anotasi *bounding box* yang dilakukan.

Pada *CrowdHuman* dilakukan *bounding box* terhadap tiga hal, yakni Visible Bbox, Full Bbox dan Head Bbox. Visible Bbox digunakan untuk membatasi daerah dimana seseorang terlihat tanpa tertutup objek, Full Bbox untuk membatasi daerah yang merupakan tinggi orang sebenarnya dan tidak memperhatikan objek yang menutupnya, dan Head Bbox untuk membatasi bagian kepala seseorang.

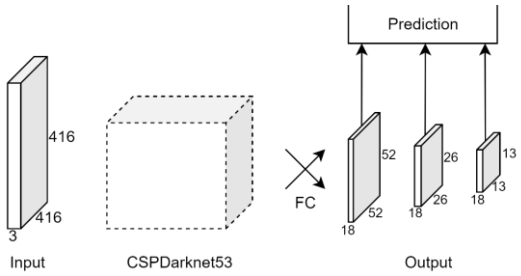
TABEL I. TABEL PERBANDINGAN DATASET *CROWDHUMAN* DENGAN DATASET LAINNYA

	Caltech	KITTI	City Persons	COCO Persons	*CrowdHuman*
# citra	42,782	3,712	2,975	**64,115**	15,000
# orang	13,674	2,322	19,238	257,252	**339,565**
# *ignore*	50,363	45	6,768	5,206	**99,227**
# orang/citra	0.32	0.63	6.47	4.01	**22.64**
# orang unik	1,273	< 2,322	19,238	257,252	**339,565**

B. You Only Look Once

YOLO [8] atau *You Only Look Once* adalah salah satu pendeteksi objek *single-stage* selain SSD. Dengan YOLO, deteksi objek dilakukan dengan melihat masalah tersebut sebagai masalah regresi untuk memisahkan secara spasial *bounding box* dan kelas probabilitas yang terkait terhadap *bounding box* tersebut. Sebuah neural network memprediksi *bounding box* dan kelas prediksi langsung dari keseluruhan citra dari satu evaluasi. YOLO berjalan pada *framework darknet* [9]. Arsitektur bawaan YOLO juga bernama *darknet*.

YOLOv4 [10] merupakan versi orisinil terbaru yang menambahkan sejumlah implementasi tambahan yang meningkatkan akurasi dan efisiensi pendeteksian seperti *Weighted Residual Connection* [11], *Cross Stage Partial Connections* (CSP), *Cross Mini-Batch Normalization* [12], *Self Adversarial Training* [13], *Mosaic Data Augmentation*, *DropBlock regularization* [14], dan *CioU loss* [15].



Gambar. 1. Arsitektur Sederhana Pendeteksi Objek YOLOv4

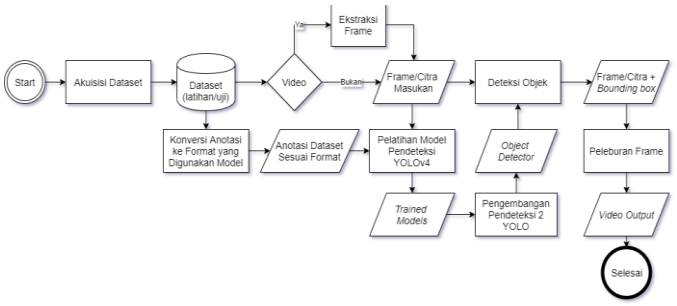
Pada Gambar 1 , dapat dilihat arsitektur yang sangat sederhana dari YOLOv4. Arsitektur ini menggunakan arsitektur *backbone* CNN *CSPDarknet53* [10]. Arsitektur ini berisi 162 lapisan. Masukan atau input merupakan citra yang sudah di-*resize* sesuai ukuran yang diatur pada konfigurasi. Selanjutnya, lapisan input dimasukan kedalam arsitektur *backbone CSPDarknet53*. Pada *CSPDarknet53* seperti yang sudah dijelaskan sebelumnya terdapat implementasi seperti CSP, FPN(*Feature Pyramid Network*), SPP (*Spatial Pyramid Pooling*), dan lainnya yang meningkatkan performa deteksi. Pada bagian luaran, output dihasilkan dari proses konvolusi dari luaran arsitektur *backbone CSPDarknet53*. Selain itu juga terdapat konkatenasi dari hasil konvolusi lapisan yang sebelumnya sudah dilakukan terhadap lapisan output. Hal ini ditujukan untuk menambahkan performa dengan melihat fitur pada lapisan sebelumnya.

C. Pendeteksi Objek Tangguh Melalui Pembelajaran Simbiotik dalam Robot Seluler

Pada penelitian [16], dikembangkan metode perhitungan *mean average precision*(mAP) yang dapat menilai tingkat keakuratan sebuah pendeteksi objek. Hal ini dilakukan dengan cara membandingkan hasil deteksi dengan *ground truth* dan menghitung *intersection over union* (IoU) untuk mendapatkan hasil deteksi berupa *true positive* dan *false positive* untuk menghitung *precision* dan *recall*. Setelah itu dihitung mAP dengna melakukan interpolasi pada grafik *precision-recall*.

III. RANCANGAN SOLUSI

Dikembangkan alur proses yang menggambarkan sistem dari awal akuisisi dataset, pelatihan model, pengembangan program, hingga menggunakan model pendeteksi untuk menghasilkan *bounding-box*. Alur proses dapat dilihat pada gambar 2. Model pendeteksi objek yang digunakan adalah YOLOv4 dan YOLOv4-*tiny* dengan arsitektur CNN *CSPDarknet53* sebagai arsitektur YOLOv4 dan arsitektur *custom* untuk YOLOv4-*tiny*. Secara ringkas solusi yang diajukan adalah menggunakan dua model pendeteksi yang masing-masing YOLOv4 (*original* atau *tiny*) dengan besaran *network-size* yang sama atau dapat berbeda. Penggunaan dua model atau dua kali deteksi bertujuan untuk meningkatkan akurasi (mAP) pendeteksian YOLOv4 dengan melakukan deteksi pada bagian ramai pejalan kaki di tengah citra.



Gambar. 2. Alur Proses Pengembangan Pendeteksi Pedestrian

A. *Akuisisi Dataset*

Untuk pelatihan model, digunakan *dataset* yang bersumber dari *CrowdHuman* [7], dan data pengujian bersumber dari Youtube, yang nantinya akan diubah menjadi citra agar dapat digunakan untuk perhitungan metrik. *Dataset CrowdHuman* diunduh melalui situs web resmi. *Dataset CrowdHuman* memiliki format yang berbeda dari format yang digunakan oleh *framework darknet* yang digunakan YOLO, oleh karena itu, perlu dilakukan konversi format label dataset ke format yang digunakan yolo menggunakan script *python* [17]. Isi *dataset CrowdHuman* merupakan sekumpulan citra dengan format JPG (*Joint Photographics Experts Group*)

B. *Pengembangan Model Pendeteksi Objek*

Model pendeteksksi objek dikembangkan menggunakan backbone CNN berupa CSPDarknet-53 dan pendeteksi objek YOLOv4 serta YOLOv4-*tiny*. Pemilihan menggunakan YOLOv4 didasarkan dari nilai akurasi yang lebih tinggi dibandingkan dengan SSD (*Single Shot Detector*) yang merupakan salah satu pendeteksi objek lain yang berjenis *single-stage* dan lebih cepat dari model pendeteksi *two-staged*. Dengan dipilihnya model pendeteksi yang cukup baik dari segi akurasi dan kecepatan inferensi, diharapkan model dapat mendeteksi *pedestrian* dengan akurasi tinggi dan dalam kecepatan yang *real-time*. Kedua model YOLOv4 dan YOLOv4-*tiny* dilatih terhadap dataset *CrowdHuman* diharapkan meningkatkan performa deteksi *pedestrian* yang berada pada konidsi yang ramai atau tertutup oleh objek lain.

C. *Pengembangan Pendeteksi Ganda*

Model pendeteksi objek yang telah dikembangkan dari tahapan sebelumnya digunakan dalam pembuatan program program pendeteksi *pedestrian* yang berjalan dua kali atau pendeteksi ganda. Program tersebut menerima masukan jenis model dan konfigurasi yang beragam untuk meningkatkan fleksibilitas penggunaan program. Untuk pendeteksian jarak jauh, bagian pada citra akan dipotong. Dalam kasus deteksi *pedestrian*, bagian citra yang diambil adalah bagian tengah pada cuplikan keadaan jalan raya. Dengan konfigurasi besar input citra yang besar, maka citra akan disesuaikan dengan ukuran konfigurasi sehingga diharapkan model dapat melihat lebih jelas *pedestrian* yang berada di kejauhan. Hasil deteksi dari satu *frame* merupakan *bounding-box* yang digambarkan ke *frame* tersebut. Hasil akhirnya merupakan masing-masing frame dilebur menjadi satu video utuh.

IV. IMPLEMENTASI

Implementasi dilakukan pada lingkungan cloud dengan menggunakan *google colab*. Keseluruhan program dan dataset disimpan didalam *google drive* yang dapat diakses oleh *google colab*. Program berjalan dengan bahasa python dan menggunakan *library OpenCV*, *Darknet*, *NMS*, *Numpy*, *Glob*, dan *Time*. Pelatihan dan pengujian program juga dijalankan di *google colab*, yang menggunakan GPU *Nvidia Tesla T4*. Secara garis besar, implementasi terdiri dari persiapan dataset, pelatihan model serta pemilihan model terbaik, dan pengembangan program pendeteksi dua kali.

A. *Dataset*

Model dilatih dengan *dataset* yang mengandung citra serta anotasi objek yang dideteksi, yakni *pedestrian*. Seperti yang sudah disebutkan pada bagian sebelumnya, akan digunakan *dataset CrowdHuman* [7] sebagai data pelatihan. Anotasi *dataset* yang berbeda dengan format *darknet* perlu dikonversi ke format yang sesuai dengan menggunakan *script* [17] yang mengambil data anotasi dengan format json dan diubah ke format txt yang digunakan *darknet*. Anotasi yang diambil hanya anotasi *Full Bbox*, yang merupakan anotasi keseluruhan satu manusia yang dapat tertutup oleh suatu objek. Hal ini ditujukan agar pendeteksi objek yang dilatih dapat mendeteksi *pedestrian* yang tertutup oleh objek lainnya.

Data pengujian diambil dari sumber *YouTube* yang berisi rekaman dasbor mobil maupun rekaman pejalan kaki saat *car free day* (CFD). Video tersebut diambil karena mengandung kondisi sebenarnya jalan raya di Indonesia (khususnya di Jakarta). Selain itu, video juga mengandung kasus-kasus seperti ramai *pedestrian* (CFD, rekaman di daerah pasar), atau kondisi sepi (di jalan raya). Dataset yang dipilih memiliki kondisi pencahayaan terang, walaupun ada idataset yang siang hari menjelang sore hari.

B. *Konfigurasi Model*

Pada bagian sebelumnya, disebutkan bahwa digunakan sistem pendeteksi objek YOLOv4 yang menggunakan arsitektur *backbone CSPDarknet53*. Arsitektur ini mengandung sejumlah implementasi tambahan [10]. Bagian arsitektur tidak akan diubah, hanya akan dilakukan penyesuaian terhadap sejumlah parameter pada konfigurasi yang dapat dilihat pada Tabel II.

TABEL II. KONFIGURASI YOLOV4 DAN YOLOV4-*TINY* UNTUK PELATIHAN

No.	Parameter	YOLOv4	YOLOv4-tiny
1	Batch size	64	
2	*Network* size	416	608
3	Learning rate	0.001	0.0026
4	Momentum	0.9	0.949
5	Decay	0.0005	
6	Burn in	1000	
7	Max batch	30000	
8	Activation function	Leaky ReLU, Linear, Mish	

Selain parameter tersebut, jumlah filter pada setiap lapisan sebelum konvolusi juga diubah sesuai dengan jumlah kelas yang dilatih. Jumlah filter pada setiap lapisan konvolusi sebelum lapisan yolo diubah dengan menggunakan persamaan (1).

$$F = (C + Co + P) * M \quad (1)$$

dengan F sebagai jumlah filter, C sebagai jumlah kelas, Co sebagai jumlah koordinat yang digunakan (x, y, w, h), P sebagai nilai ke-objek-an suatu daerah yang diajukan, serta M sebagai jumlah *anchor box* yang digunakan. Jumlah filter diubah menjadi (1+4+1)*3 = 18.

C. Implementasi Model Pendeteksi Ganda

Setelah model terbaik dari hasil pelatihan dipilih, model tersebut digunakan dalam pengembangan sistem pendeteksi *pedestrian*. Program dibuat dengan bahasa *python*. Program penggunaan dari *darknet* sudah tersedia pada *source code* untuk selanjutnya dikembangkan menjadi pendeteksi ganda. Sistem pendeteksi dapat menerima masukan berupa dua model yang berbeda ataupun satu model dengan konfigurasi masing-masing.

Program mendeteksi dengan dua iterasi deteksi untuk masing-masing model. Dikarenakan keterbatasan komputasional dalam pengembangan program ini, tidak diimplementasikan pemrosesan paralel untuk masing-masing model. Pemrosesan paralel tidak dapat dilakukan pada satu GPU saja dikarenakan *framework darknet* sudah mengimplementasikan pemrosesan paralel pada *source code*-nya.

Pendeteksi ganda dimulai dengan mendeteksi objek dari citra utuh terlebih dahulu, seperti deteksi dengan YOLO umumnya. Citra input masuk ke model dengan ukuran citra yang di-*resize* sesuai dengan konfigurasi besar ukuran citra masukan. Semakin besar ukuran citra masukan, maka semakin akurat deteksi objek yang dilakukan dan semakin lambat pemrosesan yang dilakukan dan sebaliknya. Pendeteksian kedua dilakukan dengan input citra yang merupakan bagian potongan dari citra utuh yang umumnya mengandung *pedestrian* dengan jarak yang lebih jauh dari kamera dan berada di jalur mobil yang berjalan lurus. Citra potongan ini juga dapat berisi *pedestrian* yang Citra potongan ini juga dapat berisi *pedestrian* yang berada dalam jarak dekat sehingga menutupi penglihatan objek yang berada pada jarak jauh. Proses berjalannya sistem deteksi ganda dapat dilihat pada gambar 3.

Gambar. 3. Proses Pendeteksian Ganda dengan YOLO

Pada gambar 3, dapat dilihat setelah melakukan deteksi dua kali dihasilkan *bounding box* deteksi *pedestrian* untuk masing-masing citra masukan. Hasil deteksi dari kedua citra masukan akan digabungkan menjadi satu. Hasil deteksi dapat mengandung deteksi duplikat untuk satu *pedestrian*. Oleh karena itu, deteksi duplikat diatasi dengan melakukan *non-maximum suppression* untuk melakukan reduksi *bounding box* yang beririsan untuk satu objek agar menghasilkan hanya satu *bounding box* untuk satu objek saja.

V. PENGUJIAN

Sistem pendeteksi *pedestrian* yang dikembangkan diuji dengan melakukan perhitungan terhadap metrik kecepatan FPS serta metrik mAP. Pengujian dilakukan untuk melakukan evaluasi serta melihat hasil dari sistem pendeteksi yang telah dikembangkan dan membandingkannya dengan pendeteksi *state-of-the-art*. Kriteria yang ingin dicapai adalah kecepatan model yang *real-time* untuk melakukan deteksi objek dan nilai mAP yang paling tinggi dipilih sebagai model terbaik. Metrik mAP diperuntukkan untuk membandingkan model yang menggunakan sistem deteksi ganda dan model yang tidak menggunakan deteksi ganda (*state-of-the-art*).

A. Persiapan Pengujian

Dataset pengujian diambil dari sumber *YouTube*. Video pengujian dipotong menjadi sekumpulan citra dengan menggunakan perkakas *Yolo_mark* [18]. Setelahnya, citra tersebut disaring kembali untuk menghilangkan skenario yang kurang relevan dengan skenario *autonomous vehicle*. Selanjutnya, masing-masing citra akan dilabeli untuk tiap objek *pedestrian*. Pelabelan dilakukan dengan perkakas *LabelImg* [19] karena lebih mudah penggunaannya. Data uji yang dihasilkan berisi 338 citra dengan resolusi 1280x720 (720p) dengan format anotasi YOLO. Cara pelabelan sama dengan dataset *CrowdHuman*, tetapi hanya melakukan pelabelan terhadap *bounding box* keseluruhan *pedestrian* yang memungkinkan *pedestrian* yang tertutup oleh objek lain. Dari jumlah tersebut, kerumunan atau yang tidak terlihat seperti *pedestrian* tidak dilabelkan. Pengambilan dataset ini dipilih untuk menguji dengan data yang kondisi asli di Indonesia.

B. Skenario Pengujian

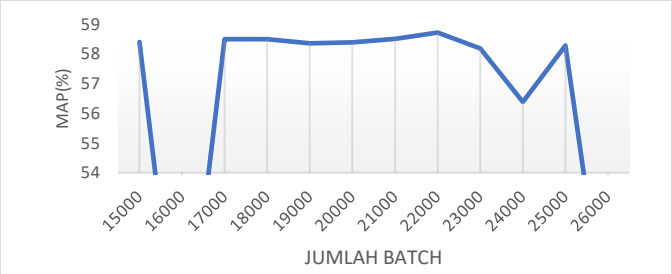
Pengujian dilakukan dengan mengkombinasikan dua model atau satu model dengan konfigurasi yang bermacam. Hal ini ditujukan untuk mencari kombinasi terbaik dari bermacam kombinasi. Setelahnya dilakukan perbandingan dengan model *state-of-the-art* yang sudah dilatih pada *dataset* yang sama. Kombinasi yang diuji dapat dilihat pada Tabel III.

TABEL III. KOMBINASI PENGUJIAN DETEKSI GANDA

No.	Kombinasi	Keterangan	Jumlah Kombinasi
1.	Model Pendeteksi Objek	Kombinasi antar kedua model: • YOLOv4 + YOLOv4 • YOLOv4 + YOLOv4-*tiny* • YOLOv4-*tiny* + YOLOv4 • YOLOv4-*tiny* + YOLOv4-*tiny*	4
2.	Besar *Network*	Ukuran 1:1 untuk Panjang dan Lebar pada 256, 320, dan 416: • 256 + 256, 320, 416 • 320 + 256, 320, 416 • 416 + 256, 320, 416	9
3.	Metode NMS	*Fast* dan Malisiewicz	2
		Total Kombinasi:	72

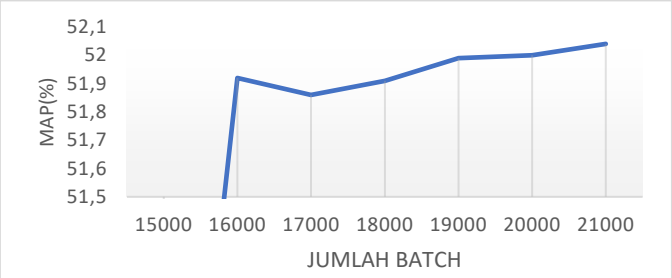
C. *Hasil Pengujian*

Pada gambar 4 dan gambar 5, diperlihatkan hasil dari pelatihan model YOLOv4 dan YOLOv4-*tiny* sesuai dengan konfigurasi yang ada pada tabel II. Dari hasil tersebut, dipilih model terbaik yang digunakan dalam pengujian lainnya. Model yang dipilih adalah model yang memiliki nilai mAP terbaik. Untuk model YOLOv4, model terbaik diraih oleh model dengan *batch* pelatihan sejumlah 22000 *batch* dengan nilai mAP 58,71 persen. Khusus untuk model YOLOv4 hanya dilatih sebanyak 26000 *batch*, tidak hingga 30000 *batch*. Hal ini dikarenakan nilai mAP yang sudah mulai menurun pada *batch* setelah 22000.



Gambar. 4.Hasil Pelatihan YOLOv4

Selain itu, untuk model YOLOv4-*tiny*, model dengan nilai tertinggi diraih oleh model dengan jumlah *batch* pelatihan sebanyak 30000 *batch* atau *batch* terakhir. Kedua model terpilih tersebut dipilih sebagai model *state-of-the-art* dan juga digunakan dalam sistem deteksi ganda.



Gambar. 5. Hasil Pelatihan YOLOv4-*tiny*

VI. EVALUASI HASIL

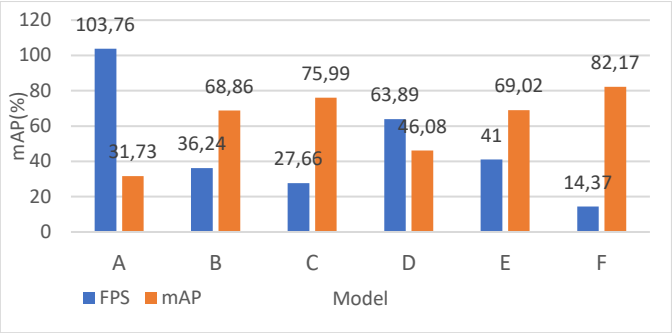
Pada Tabel IV, Model YOLOv4 menghasilkan kecepatan yang jauh lebih rendah dibandingkan dengan model YOLOv4-*tiny* karena model memiliki arsitektur yang lebih tebal (lebih banyak lapisan) dibandingkan dengan model *tiny*. Selain itu, penggunaan algoritme NMS Malisiewicz juga menurunkan mAP, walau terlihat peningkatan pada kecepatan deteksi. Model-model terbaik dicapai dengan algoritme NMS *fast* yang merupakan implementasi dari OpenCV. Model dengan kecepatan terbaik diraih oleh model A. Model yang paling presisi dicapai oleh model F.

Pada tahap selanjutnya, dikombinasikan model, *network size*, serta algoritme NMS sesuai tabel III. Sama seperti hasil pengujian sebelumnya, algoritme NMS *fast* menghasilkan performa yang lebih baik dibandingkan dengan algoritme NMS Malisiewicz. Kecepatan paling tinggi dicapai model A. Untuk kecepatan *real-time* dengan mAP tertinggi diraih oleh model E dengan *network size* 416 untuk masing-masing model yang mencapai mAP 69,02 persen pada kecepatan 41 FPS. Untuk model dengan akurasi paling tinggi dicapai oleh kombinasi model F dengan *network size* 416 untuk masing-masing model yang mencapai angka mAP 82,17 persen dengan kecepatan 14,37 FPS. Selain performa, terdapat pola yang unik pada hasil pengujian tersebut. Model yang menggunakan konfigurasi *network size* yang sama cenderung melakukan performa deteksi yang lebih baik dan sedikit lebih cepat dibandingkan konfigurasi *network size* yang berbeda.

Pada Gambar 6, diperlihatkan perbandingan antara model-model terbaik yang didapatkan dari pengujian yang nama modelnya disamakan dengan Tabel IV. Pada grafik tersebut, dapat dilihat bahwa penggunaan deteksi dua kali tidak mengalami perubahan yang signifikan pada model YOLOv4 atau kombinasinya. Di sisi lain, untuk model YOLOv4-*tiny*, dapat dilihat bahwa terdapat peningkatan angka mAP untuk model dengan konfigurasi *network size* kecil. Perbandingan paling jelas dapat dilihat pada model B dengan model E dimana model E memiliki nilai mAP lebih tinggi 0,16 persen dan kecepatan lebih tinggi 4,76 FPS dibandingkan dengan model B. Walaupun menggunakan model dengan citra masukan kecil, model dapat menyandingi, bahkan menghasilkan performa yang lebih baik dibandingkan hanya menggunakan satu kali deteksi saja. Deteksi pejalan kaki untuk kasus autonomous vehicle umumnya berada di bagian tengah citra. Pada perbandingan antara kedua model paling akurat, yakni model C dan model F memperlihatkan hasil yang berbanding jauh, dimana model C masih mendekati kecepatan *real-time* pada 27,66 FPS dibandingkan dengan model C yang hanya mencapai 14,37 FPS. Tetapi, pengorbanan pada sisi kecepatan memang membuahkan hasil yang baik, dimana model F memiliki mAP yang lebih tinggi di angka 82,17 persen dibandingkan model C yang mencapai 75,99 persen.

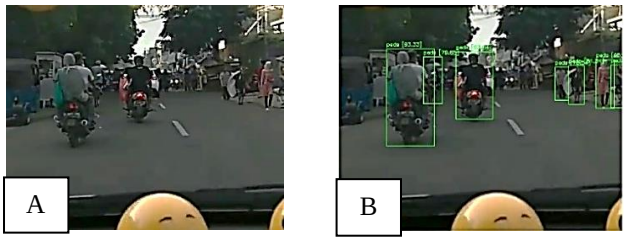
TABEL IV. MODEL TERBAIK DARI HASIL PENGUJIAN

No.	Nama	Model 1	Model 2	Besar Input *Network 1*	Besar Input *Network* 2	NMS	FPS	mAP/AP (%)
1	A	YOLOv4-*tiny*	-	256	-	Fast	103,76	31,73
2	B	YOLOv4	-	320	-		36,24	68,86
3	C	YOLOv4	-	416	-		27,66	75,99
4	D	YOLOv4-*tiny*	YOLOv4-*tiny*	256	256		63,89	46,08
5	E	YOLOv4-*tiny*	YOLOv4-*tiny*	416	416		41,00	69,02
6	F	YOLOv4	YOLOv4	416	416		14,37	82,17



Gambar. 6. Grafik Perbandingan Model dari Tabel IV

Pada Gambar 7, A adalah hasil deteksi model A dan B adalah hasil deteksi model D. Dapat dilihat penggunaan deteksi ganda berpengaruh terhadap jumlah deteksi Selain itu, dapat dilihat model juga mendeteksi pengendara motor karena kemiripan fitur antara pengedara dengan *pedestrian*. Hal ini menunjukkan bahwa dataset *CrowdHuman* tidak hanya terbatas dengan mendeteksi *pedestrian*, tetapi manusia secara keseluruhan. Oleh karena itu, penggunaan modul deteksi ini memiliki tradeoff yang cukup kontras, dimana apabila digunakan akan mendapatkan akurasi tambahan dengan pengorbanan performa kecepatan deteksi.



Gambar. 7. Perbandingan Hasil Deteksi pada Bagian *Cropping* Model A dan Model D

VII. KESIMPULAN

Arsitektur CNN *CSPDarknet53* pada YOLOv4 sudah sangat baik dalam melakukan proses ekstraksi fitur dan deteksi pedestrian yang dapat dilihat dari hasil pengujiannya terhadap data uji pedestrian untuk kasus Indonesia. Dikarenakan lapisan arsitektur yang lebih banyak dari arsitektur *tiny*, menghasilkan akurasi yang lebih tinggi. Arsitektur CNN yang digunakan pada YOLOv4-*tiny* juga memiliki kelebihan di sisi kecepatan deteksi. Deteksi dua kali sangat berpengaruh untuk model yang memiliki network size yang kecil. Sedangkan untuk model YOLOv4, tidak ada pengaruh yang signifikan dalam pemrosesan citra dan sudah dapat melakukan pendeteksian objek kecil dengan sendirinya.

Akhir kata, pendeteksi objek YOLOv4 merupakan salah satu pendeteksi yang dapat melakukan pemrosesan secara *real-time*. Kemampuan generalisasi objek yang baik menghasilkan deteksi yang konsisten di beragam dataset. Metrik yang dihasilkan pada pengujian sudah sangat baik mengingat perbedaan dataset yang digunakan. Meskipun berhasil untuk sebagian kasus, sistem ini masih memiliki kekurangan yang dapat dikembangkan kedepannya. Saran pengembangan untuk kedepannya adalah melakukan implementasi pemrosesan paralel dengan menggunakan dua GPU serta melakukan implementasi daerah potongan yang dapat berpindah sesuai kondisi secara otomatis.

REFERENSI

[1] M. Maurer, J. C. Gerdes, B. Lenz dan H. Winner, Autonomous Driving, Ladenburg: Springer Open, 2015.

[2] L. Greenemeier, "Uber Self Driving Car Fatality Reveals the Technology's Blind Spots," 2018. [Online]. Dikutip dari: https://www.scientificamerican.com/article/uber-self-driving-car-fatality-reveals-the-technologys-blind-spots1/. [Diakses 30 September 2019].

[3] T. S. Combs, L. S. Sandt, M. P. Clamann dan N. C. McDonald, "Automated Vehicles and Pedestrian Safety: Exploring the Promise and Limits of Pedestrian Detection," *American Journal of Preventive Medicine*, 2019.

[4] P. Read dan M. Mark-Paul, Restoration of Motion Picture Film, Jordan Hill, Oxford: Butterworth-Heinemann, 2000.

[5] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu dan A. C. Berg, "SSD: Single Shot Multibox Detector," *arXiv:1512.02325vv5[cs.CV]*, 2015.

[6] T. Liu dan T. Stathaki, "Faster R-CNN for Robust Pedestrian Detection Using Semantic Segmentation Network," *frontiers in Neurorobotics*, vol. 12, no. 64, 2018.

[7] S. Shao, Z. Zhao, B. Li, T. Xiao, G. Yu, X. Zhang dan J. Sun, "CrowdHuman: A Benchmark for Detecting Human in a Crowd," *arXiv:1805.00123v1[cs.CV]*, 2018.

[8] J. Redmon, S. Divvala, R. Girshick dan A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2015.

[9] J. Redmon, "Darknet: Open Source Neural Networks in C," 2013. [Online]. Available: https://pjreddie.com/darknet/.

[10] A. Bochkovskiy, C.-Y. Wang dan H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," arXiv preprint arXiv:2004.10934, 2020.

[11] K. He, X. Zhang, S. Ren dan J. Sun, "Deep Residual Learning for Image Recognition," dalam *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[12] Z. Yao, Y. Cao, S. Zheng, G. Huang dan S. Lin, "Cross-Iteration Batch Normalization," arXiv preprint arxiv:2002.05712, 2020.

[13] D. Misra, "Mish: A self regularized non-monotonic neural activation function," arXiv preprint arXiv: 1908.08681, 2019.

[14] G. Ghiasi, T.-Y. Lin dan Q. V. Le, "DropBlock: A regularization method for convolutional networks," dalam *Advances in Neural Information Processing Systems (NIPS)*, 2018.

[15] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye dan D. Ren, "Distance-IoU Loss: Faster and better learning for bounding box regression.," dalam *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

[16] J. Cartucho, R. Ventura dan M. Veloso, "Robust Object Recognition Through Symbiotic Deep Learning In Mobile Robots," dalam *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 2336-2341.

[17] A. Laksana. [Online]. Dikutip dari: https://github.com/alaksana96/darknet-crowdhuman.

[18] A. Bochkovskiy, "Yolo_mark," [Online]. Dikutip dari: https://github.com/AlexeyAB/Yolo_mark.

[19] Tzutalin, "LabelImg," 2015. [Online]. Dikutip dari: https://github.com/tzutalin/labelImg.