JCENIM

ISSN 2986-1764
Volume 1, Nomor 1, Maret 2023

Deteksi Lampu Lalu Lintas Menggunakan YOLO untuk Autonomous Car

Dedy Hidayat Kusuma[1] ,Mauizah[2]

[1]Prodi Bisnis Teknik Informatika Digital, Politeknik Negeri Banyuwangi, Indonesia
[2]Dept. Teknik Komputer, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia

Email: dedy@poliwangi.ac.id , mauizah230997@gmail.com

Abstrak

Autonomous car adalah pengembangan dari teknologi mobil dengan berbagai fitur pintar yang dilengkapi pada mobil tersebut. Salah satu fitur yang perlu dikembangkan adalah deteksi lampu lalu lintas.Dalam fitur ini Autonomous Car harus dapat mengenali letak dan warna lampu lalu lintas. Pada tugas akhir ini akan dikembangkan sebuah sistem pendeteksian letak dan warna lampu lalu lintas untuk keamanan Autonomous Car, sehingga Autonomous Car dapat berhenti saat lampu berwarna merah, mengurangi kecepatan saat lampu berwarna kuning dan jalan saat lampu berwarna hijau. Dengan menggunakan sensor kamera dan metode YOLO akan dapat mengenali lampu lalu lintas dan diklasifikasikan menurut fungsinya, Dari hasil klasifikasi ini menghasilkan respon yang sesuai dengan perintah lampu lalu lintas yang sudah ada pada Autonomous Car.

Kata kunci : Marka Jalan, Convolutional Neural Network (CNN), TensorFlow, You Only Look Once (YOLO)

Diterima Redaksi: 20-10-2022, Selesai Revisi: 09-12-2022, Diterbitkan Online: 27-02-2023
DOI: https://doi.org/10.59378/jcenim.v1i1.6

I. PENDAHULUAN

Perkembangan teknologi mobil dengan berbagai fitur pintar saat ini telah sampai pada Autonomous Car. Kendaraan ini memiliki beberapa tantangan seperti mengenali Lampu lalu lintas, rambu-rambu, pejalan kaki, tanda jalur tidak jelas dan lain-lain[1].Akan tetapi, sampai saat ini belum ada sistem pada Autonomous Car untuk mengenali lampu lalu lintas berdasarkan letak dan warnanya. Lampu lalu lintas menurut Undang-Undang Republik Indonesia Tahun 2009 Pasal 1 ayat 19 adalah perangkat elektronik yang menggunakan isyarat lampu yang juga dapat menggunakan isyarat bunyi untuk mengatur lalu lintas orang / kendaraan dipersimpangan atau pada ruas jalan[2]. Di Indonesia jarang ditemukan sistem pendeteksi lampu lalu lintas pada mobil sehingga masih banyak hal yang dapat dikembangkan pada Autonomous Car.

II. Kajian Pustaka

Pada pertengahan 2000-an, Badan Proyek Penelitian Lanjutan Pertahanan (DARPA) memilih Tantangan Besar di mana kelompok-kelompok berkumpul untuk bersaing dengan kendaraan yang mengemudi sendiri. Pada tahun 2009, Google memulai usaha mobil tanpa pengemudi, termasuk kolega yang telah mengabdikan tahun secara efektif untuk inovasinya. Pada tahun 2012, mobil Google mulai diuji. Selama bertahun-tahun, mobil ini dikembangkan dan dilengkapi dengan banyak sensor, radar, laser, Global Positioning System (GPS), menggunakan peta yang sangat rinci, dan banyak hal lain untuk mengemudi dan menavigasi dirinya sendiri dengan aman tanpa interaksi manusia.

Mobil tidak hanya dapat mengemudi sendiri tetapi dapat diparkir sendiri, dapat melaju di jalan raya, Kamera digunakan untuk mencari dan mendeteksi objek yang kemudian diproses oleh komputer di dalam mobil [3]. Pada Mei 2014, Google mempresentasikan konsep baru untuk mobil tanpa pengemudi mereka yang tidak memiliki roda kemudi atau pedal dan meluncurkan prototipe yang berfungsi penuh pada bulan Desember tahun itu yang mereka rencanakan untuk diuji pada 2015 [4].Para penulis [5],[6],

(a) Rambu lalu lintas warna merah

(b) Rambu lalu lintas warna hijau

Gambar 1: Data set rambu lalulintas

dan [7] telah mengembangkan prototipe kendaraan tak berawak di mana mereka telah bekerja pada penghindaran rintangan dan perencanaan jalur [8].

Dalam makalah ini , bertujuan untuk membuat sistem deteksi lampu lalu lintas beserta dengan pengenalan lampu untuk direspon oleh Autonomous Car dan dapat merespon sesuai dengan makna lampu lalu lintas.

III. Perancangan Sistem

Dataset Gambar

Dataset adalah data yang akan digunakan atau diolah. Pada tahap dataset gambar ini dilakukan pengambilan dari internet berupa gambar yang dibuat oleh Morten Borno Jensen dan satu kolaborasi dengan Mark Philip dan dataset yang diambil di lokasi Surabaya .

You Only Look Once (YOLO)

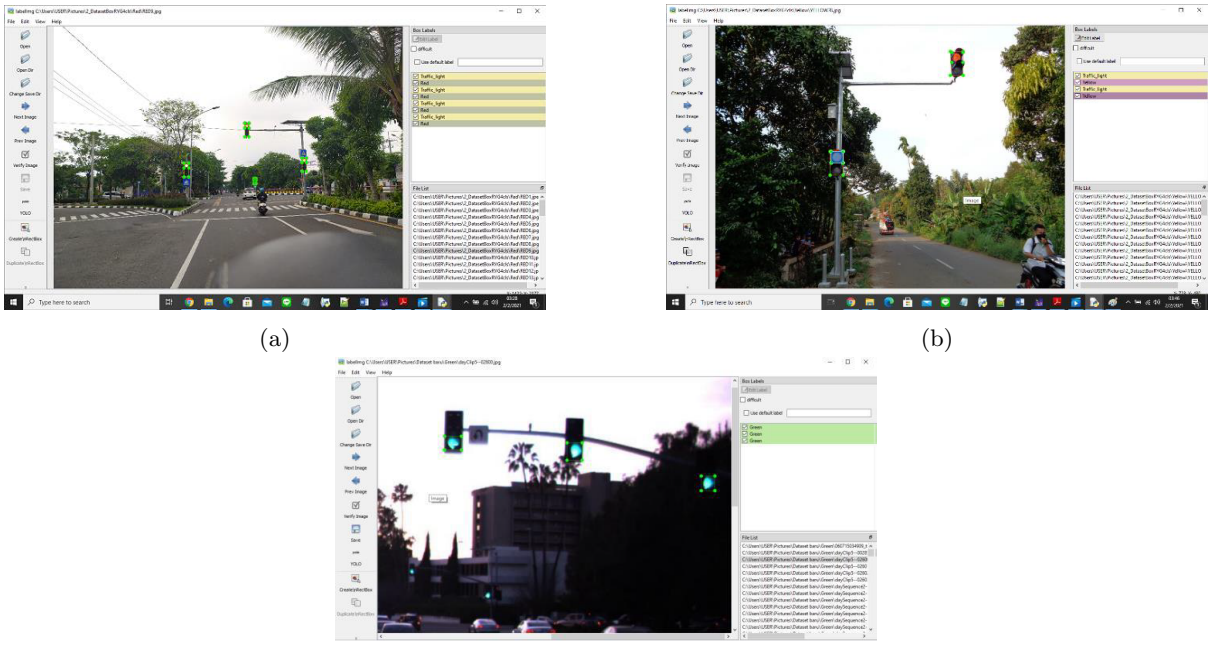
Metode pendeteksian lampu lalu lintas pada makalah ini menggunakan You Only Look Once (YOLO). YOLO memproses gambar secara real-time pada 45 frames per second. Dibandingkan dengan sistem deteksi yang lainnya, YOLO membuat lebih banyak kesalahan pada pelokalannya tetapi dapat memprediksi kesalahan pada backround objek. YOLO dapat merepresentasi objek yang sangat umum.

Convolutional Neural Network (CNN)

Convolutional Neural Network atau CNN adalah variasi dari Multilayer Perceptron yang terinspirasi dari jaringan syaraf manusia. CNN merupakan suatu layer yang memiliki susunan neuron 3D (lebar, tinggi, kedalaman). Lebar dan tinggi merupakan ukuran layer sedangkan kedalaman mengacu pada jumlah layer. Secara umum jenis layer pada CNN dibedakan menjadi dua yaitu:

1. Layer ekstraksi fitur gambar, letaknya berada pada awal arsitektur tersusun atas beberapa layer dan setiap layer tersusun atas neuron yang terkoneksi pada daerah lokal (local region) layer sebelumnya. Layer jenis pertama adalah layer konvolusi dan layer kedua adalah layer pooling. Setiap layer diberlakukan fungsi aktivasi. Posisinya berselang-seling antara jenis pertama dengan jenis kedua. Layer ini menerima input gambar secara langsung dan memprosesnya.

Gambar 2: Deteksi objek menggunakan YOLO



(a) (b)

(c)

Gambar 3: Pemberian label pada rambu lalulintas (a) warna mera,(b) Kuning dan (c) Hijau

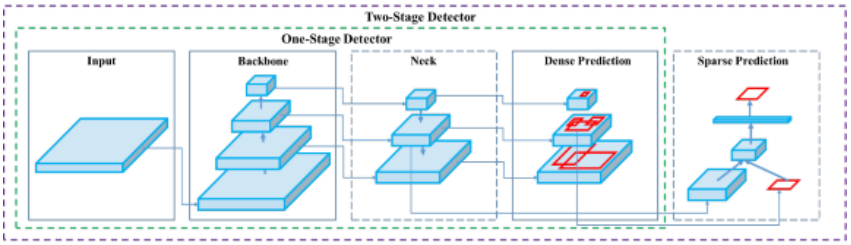
2. Layer klasifikasi, tersusun atas beberapa layer dan setiap layer tersusun atas neuron yang terkoneksi secara penuh (fully connected) dengan layer lainnya. Layer ini menerima input dari hasil keluaran layer ekstrasi fitur gambar berupa vektor kemudian ditransformasikan seperti Multi Neural Networks dengan tambahan beberapa hidden layer. Hasil keluaran berupa skoring kelas untuk klasifikasi.

Dengan demikian CNN merupakan metode untuk mentransformasikan gambar original layer per layer dari nilai piksel gambar kedalam nilai skoring kelas untuk klasifikasi [9].

Training pada Dataset

Training dataset atau training set adalah bagian dataset yang dilatih untuk membuat prediksi atau menjalankan fungsi dari sebuah algoritma ML. Kita memberikan petunjuk melalui algoritma agar mesin yang dilatih bisa mencari korelasinya sendiri atau belajar pola dari data yang digunakan.

1. Labeling.
   Training data diawali dengan tahap labelling dataset yang telah didapatkan. Labelling adalah proses memberikan label atau anotasi pada dataset gambar yang nantinya akan digunakan untuk training data. Labelling berisikan data informasi Bounding Box beserta labelnya. Dalam proses ini aplikasi yang digunakan adalah labelling. Gambar yang sebelumnya didapatkan dari proses pengambilan Dataset, akan dimuat di dalam program Labelling. Setelah dimuat, tiap-tiap gambar



Gambar 4

```
 layer   filters  size/strd(dil)      input                output
 0 conv     32       3 x 3/ 1    416 x 416 x   3 ->  416 x 416 x  32 0.299 BF
 1 conv     64       3 x 3/ 2    416 x 416 x  32 ->  208 x 208 x  64 1.595 BF
 2 conv     64       1 x 1/ 1    208 x 208 x  64 ->  208 x 208 x  64 0.354 BF
 3 route  1                                     ->  208 x 208 x  64
 4 conv     64       1 x 1/ 1    208 x 208 x  64 ->  208 x 208 x  64 0.354 BF
 5 conv     32       1 x 1/ 1    208 x 208 x  64 ->  208 x 208 x  32 0.177 BF
 6 conv     64       3 x 3/ 1    208 x 208 x  32 ->  208 x 208 x  64 1.595 BF
 7 Shortcut Layer: 4,  wt = 0, wn = 0, outputs: 208 x 208 x  64 0.003 BF
 8 conv     64       1 x 1/ 1    208 x 208 x  64 ->  208 x 208 x  64 0.354 BF
 9 route  8 2                                   ->  208 x 208 x 128
10 conv     64       1 x 1/ 1    208 x 208 x 128 ->  208 x 208 x  64 0.709 BF
11 conv    128       3 x 3/ 2    208 x 208 x  64 ->  104 x 104 x 128 1.595 BF
12 conv     64       1 x 1/ 1    104 x 104 x 128 ->  104 x 104 x  64 0.177 BF
13 route  11                                    ->  104 x 104 x 128
14 conv     64       1 x 1/ 1    104 x 104 x 128 ->  104 x 104 x  64 0.177 BF
15 conv     64       1 x 1/ 1    104 x 104 x  64 ->  104 x 104 x  64 0.089 BF
16 conv     64       3 x 3/ 1    104 x 104 x  64 ->  104 x 104 x  64 0.797 BF
17 Shortcut Layer: 14,  wt = 0, wn = 0, outputs: 104 x 104 x  64 0.001 BF
18 conv     64       1 x 1/ 1    104 x 104 x  64 ->  104 x 104 x  64 0.089 BF
19 conv     64       3 x 3/ 1    104 x 104 x  64 ->  104 x 104 x  64 0.797 BF
20 Shortcut Layer: 17,  wt = 0, wn = 0, outputs: 104 x 104 x  64 0.001 BF
21 conv     64       1 x 1/ 1    104 x 104 x  64 ->  104 x 104 x  64 0.089 BF
22 route  21 12                                ->  104 x 104 x 128
23 conv    128       1 x 1/ 1    104 x 104 x 128 ->  104 x 104 x 128 0.354 BF
24 conv    256       3 x 3/ 2    104 x 104 x 128 ->   52 x  52 x 256 1.595 BF
25 conv    128       1 x 1/ 1     52 x  52 x 256 ->   52 x  52 x 128 0.177 BF
26 route  24                                    ->   52 x  52 x 256
27 conv    128       1 x 1/ 1     52 x  52 x 256 ->   52 x  52 x 128 0.177 BF
28 conv    128       1 x 1/ 1     52 x  52 x 128 ->   52 x  52 x 128 0.089 BF
29 conv    128       3 x 3/ 1     52 x  52 x 128 ->   52 x  52 x 128 0.797 BF
30 Shortcut Layer: 27,  wt = 0, wn = 0, outputs:  52 x  52 x 128 0.000 BF
```

Gambar 5: Hidden Layer

dapat diproses, yang pertama adalah proses membuat Bounding Box pada obyek yang dideteksi dengan cara menggunakan Create RectBox. Setelah Bounding Box dibuat, maka inputkan label dari obyek yang akan dideteksi. Pastikan format penyimpanan adalah YOLO, lalu simpan pada direktori yang sama dengan gambar dataset. Ulangi semua proses hingga semua gambar yang diambil telah diberi Bounding Box. Hasil dari Subproses Labeling adalah sebuah data yang berisikan informasi Bounding Box beserta labelnya dalam bentuk .txt. File .txt memiliki beberapa informasi penting yang digunakan dalam training data.

2. Training dengan Darknet
   Setelah dilakukan Labelling, selanjutnya adalah subproses utama yaitu training data. Framework yang digunakan adalah Darknet.Darknet adalah framework neural network open source yang ditulis dalam C dan CUDA. Darknet juga dapat melakukan komputasional baik secara CPU maupun GPU. Model yang digunakan dalam I adalah Yolov4. Arsitektur dari Yolov4 dapat dilihat pada Gambar 5.

   Detector Modern biasanya terdiri dari 2 bagian,yaitu Backbone yang sudah ditraining sebelumnya di ImageNet dan Head (Neck) yang digunakan untuk memprediksi class dan bounding box setiap objek. Untuk detector yang berjalan dengan performa GPU, Jenis Backbone yang dapat digunakan adalah VGG, ResNet, ResNeXt sedangkan jika hanya menggunakan CPU, jenis Backbone yang bisa 21 digunakan adalah SqueezeNet, MobileNet atau ShuffleNet. Adapun pada bagian Head, dapat dikategorikan menjadi 2 bagian, yaitu One Stage Detector dan Two Stage Detector. Pada One Stage Detector bisa juga disebut dengan Dense Prediction, contohnya adalah YOLO, SSD dan RetinaNet sedangkan Two Stage Detector atau bisa disebut juga Sparse Prediction contohnya adalah Fast R-CNN, Faster R-CNN dan series lainnya. Pada dasarnya bagian Neck merupakan subset dari Backbone, dimana berfungsi untuk meningkatkan ketangguhan dari feature yang digunakan.


(a) adf


(b) adf


(c) adf

Gambar 6

Tabel 1: Hasil Pengujian mAP

Iterasi tertinggi	8000			
Class	Traffic Light	Red	Yellow	Green
TP	68	20	21	24
FP	19	4	8	5
Waktu Pemrosesan (s)	11	11	11	11
Precission	0.79	0.79	0.79	0.79
Recall	0.88	0.88	0.88	0.88
Average IoU	61.13%	61.13%	61.13%	61.13%
mAP@0.5	0.82713%	0.82713%	0.82713%	0.82713%

Contoh dari Neck ini adalah Feature Pyramid Network (FPN), Path Aggregation Network (PAN), BiFPN dan NAS-FPN.
Dalam proses training, perangkat yang digunakan adalah Google Colab. Google Colab memiliki spesifikasi yaitu CPU Intel(R) Xeon(R) CPU dengan clock speed sebesar 2.20GHz. GPU yang digunakan adalah NVIDIA Tesla 4 dengan memori sebesar 15079 MB.

Model Pendeteksian Lampu Lalu Lintas dan Warnanya

Setelah proses training data dilakukan selama 8000 iterasi, didapatkan file yolov4-last.weights dengan average loss sebesar 0.126990 File .weights tersimpan sesuai dengan direktori backup pada file obj.data ini yang nantinya akan digunakan dalam proses pendeteksian bersama dengan file .cfg. Sebelum melakukan deteksi,terlebih dahulu ubah mode pada file .cfg, menjadi mode untuk testing.

IV. Hasil Pengujian

Pada bab ini menguraikan tentang pengujian dan hasil analisa dari sistem yang telah dibuat. Pengujian ini dilakukan untuk menguji kemampuan sistem yang telah dibuat dalam menjawab rumusan masalah pada bab pendahuluan.

Pengujian Sistem Deteksi Lampu Lalu Lintas dan Warnanya.

Model yang digunakan dalam sistem pendeteksian lampu lalu lintas dan warnanya adalah yolov4-obj.cfg. yolov4 adalah perkembangan dari YOLOv3 Pada YOLOv4 ini mengalami peningkatan yang sangat signifikan dalam hal kecepatan dan performance.Tujuan dari YOLOv4 ini adalah merancang operasi pendeteksian objek yang cepat dan optimisasi untuk komputasi paralel. YOLOv4 berharap objek yang didesain dapat digunakan dengan mudah untuk training dan testing.Berikut hidden layer yang digunakan pada training penelitian ini dapat dilihat pada Gambar 6.

Gambar 7: Hasil Deteksi saat hujan

Gambar 8: Hasil Deteksi pada sore hari

Evaluasi Model Pendeteksian Lampu Lalu Lintas dan Warnanya.

Evaluasi model pendeteksian dilakukan untuk mengetahui performa secara keseluruhan pada model yolov4-obj.cfg yang digunakan.Dari evaluasi ini dapat kita tentukan apakah model bekerja dengan baik atau tidak. Dari hasil training data yang dilakukan dengan menggunakan Google Colab didapatkan model pendeteksian dengan mAP sebesar 82.71% dapat dilihat pada tabel 1.

Deteksi pada Lampu Lalu Lintas dan warnanya

Percobaan untuk Deteksi lampu lalu lintas dilakukan pada kondisi cahaya terang atau seperti pada waktu siang hari.Percobaan dilakukan dengan mengambil video dan menjalankan di Google Colab Contoh deteksi dapat dilihat pada Gambar 7,8 dan 9.

Deteksi pada Lampu Lalu Lintas dan warnanya pada saat hujan

Percobaan dilakukan dengan mengambil video saat kondisi hujan pada waktu siang menjelang sore.Kemudian video dilakukan pengujian dengan menggunakan Google Colab Contoh deteksi dapat dilihat pada gambar 10 dan 11.

Deteksi pada Lampu Lalu Lintas dan warnanya pada sore hari

Percobaan dilakukan dengan mengambil video pada waktu sore hari.Kemudian video dilakukan pengujian dengan menggunakan Google Colab Berikut adalah Hasil deteksi yang telah didapatkan dan dapat dilihat pada Gambar 12 ,13 dan 14.

V. Kesimpulan

Dari pelaksanaan pengujian dan analisa yang dilakukan, maka dapat ditarik kesimpulan, yaitu :

- Model memiliki mean average precision (mAP@0.50) = 0.827132 atau 82.71% , dimana model cukup baik dalam melakukan pendeteksian.
- Model dapat mendeteksi pada saat hujan seperti yang sudah dipaparkan.
- Model sudah dapat mendeteksi lampu lalu lintas dan juga warna merah , kuning dan hijau
- Model dapat mendeteksi pada kondisi hujan.

Daftar Pustaka

[1] Raturaj Kulkarni, Shruti Dhavalikar, Sona Bangar. "Traffic Light Detection and Recognition for Self Driving Cars using Deep Lerning. 2019

[2] Republik Indonesia. 2009. Undang-Undang Nomor 22 Tahun 2009 Tentang Lalu Lintas dan Angkutan Jalan.

[3] Details-google-car, http://googlecarjames.weebly.com/details.html

[4] A. Fisher, "Inside Google's Quest To Popularize SelfDriving Cars," Popular Science, 2013. [Online]. Available:http://www.popsci.com/cars/article/2013-09/google-self-driving-car

[5] K. R. Memon, S. Memon, B. Memon, A. R. Memon, and M. Z. A. S. Syed, "Real time Implementation of Path planning Algorithm with Obstacle Avoidance for Autonomous Vehicle," in 3rd 2016 International Conference on Computing for Sustainable Global Development", New Delhi, India, 2016

[6] E. Frink, D. Flippo, and A. Sharda, "Invisible Leash: Object-Following Robot," Journal of Automation, Mobile Robotics & Intelligent Systems, vol. 10, no. 1, pp. 3–7, feb 2016

[7] P. Svec, A. Thakur, E. Raboin, B. C. Shah, and S. K. Gupta, "Target˘ following with motion prediction for unmanned surface vehicle operating in cluttered environments," Autonomous Robots, vol. 36, no. 4, pp. 383–405, apr 2013