PENGATURAN LAMPU LALU LINTAS BERDASARKAN KEPADATAN KENDARAAN MENGGUNAKAN METODE YOLO

TRAFFIC LIGHT CONTROL BASED ON VEHICLE DENSITY USING THE YOLO METHOD

Muhammad Irfan Hermawan[1], Iwan Iwut Tritoasmoro[2], Nur Ibrahim[3]

[1,2,3]Prodi S1 Teknik Telekomunikasi, Fakultas Teknik Elektro, Universitas Telkom
[1]**mirfanhermawan@student.telkomuniversity.ac.id**, [2]**iwaniwut@telkomuniversity.ac.id**,
[3]**nuribrahim@telkomuniversity.ac.id**

Abstrak

Kemacetan adalah salah satu masalah utama di kota padat penduduk, terutama dipersimpangan jalan pada waktu sibuk. Lampu lalu lintas merupakan salah satu solusi untuk mengurangi angka kemacetan dipersimpangan jalan, namun pengaturan lalu lintas perlu diperhatikan untuk memaksimalkan fungsi dari lampu lalu lintas tersebut. Oleh karena itu, diperlukan sistem pengaturan lampu lalu lintas berdasarkan kepadatan kendaraan. Penelitian ini menggunakan algoritma *You Only Look Once* (YOLO) dan *Simple Online Realtime Tracking* (SORT) untuk menghitung jumlah kendaraan pada video untuk pengaturan lampu lalu lintas. YOLO merupakan pengembangan dari algoritma deteksi objek *Convolutional Neural Network* (CNN), sehingga YOLO mampu mendeteksi suatu objek dengan tingkat akurasi yang cukup tinggi dan *frame rate* yang lebih tinggi dibandingkan dengan algoritma deteksi objek *state-of-the-art* lainnya. Skema *dataset* yang digunakan sebanyak 624 data latih dan 156 data uji. Parameter perfomansi yang ditinjau yaitu *Average Precession* (AP) dan akurasi. Nilai AP tertinggi sebesar 0,89 dengan konfigurasi *hyperparameter learning rate* 0,0001, *epoch* 60, dan *batch size* 4. Hasil pengujian akurasi didapat nilai rata-rata akurasi tertinggi adalah 98,80% dengan garis virtual diletakan 30% dari ujung atas frame video.

Kata kunci : ***Object detection, CNN, You Only Look Once, Simple Online Realtime Tracking***

Abstract

Congestion is one of the main problems in congested city, mostly on complicated roads and in busy times. Traffic lights represent one solution to reduce traffic jams at junctions, but the traffic lights need to be approved to improve the function of the traffic lights. Therefore, a traffic light control system for vehicle density is needed. This research uses You Only Look Once (YOLO) algorithm and Simple Online Realtime Tracking (SORT) to count vehicles on video for traffic light settings. YOLO is one of the developments of Convolutional Neural Network (CNN) object detection algorithm, and YOLO is able to detect an object with a fairly high accuracy and higher frame rate compared to other state-of-the-art object detectional algorithms. The number of datasets used was 624 training data and 156 test data. The performance parameters reviewed were Average Precession (AP) and accuracy. The highest AP value is 89.01% with a hyperparameter learning rate configuration of 0.0001, epoch 60, and batch size 4. The highest average accuracy value is 98.80% with a virtual line placed 30% from the top of the frame video

Keywords: ***Object detection, CNN, You Only Look Once, Simple Online Realtime Tracking***

1. Pendahuluan

Proses Indonesia merupakan negara yang memiliki 267 juta penduduk, khususnya di Provinsi Jawa Barat memiliki 49 juta penduduk[1]. Sebagian masyarakat di pulau jawa menggunakan transportasi kendaraan bermotor, baik menggunakan kendaraan umum maupun kendaraan pribadi. Dengan jumlah penduduk yang dapat dikatakan cukup padat, masalah kemacetan dan kecelakaaan lalu lintas dalam berkendara merupakan suatu masalah yang sering terjadi di pulau jawa. Padatnya penduduk khususnya pulau jawa mengakibatkan tingginya penggunaan kendaraan pribadi dan kendaraan umum yang menimbulkan masalah baru yaitu kemacetan lalu lintas. Untuk menyelesaikan masalah kemacetan tersebut maka dibuatlah sistem lampu lintas untuk mengatasi kepadatan kendaraan dan kecelakaan lalu lintas di setiap persimpangan jalan, agar setiap persimpangan jalan menjadi tertib.

Selain disebabkan oleh jumlah penduduk tersebut, pengaturan lampu lalu lintas merupakan parameter yang penting dalam mengatur dan menertibkan arus kendaraan khususnya di wilayah perkotaan. Saat ini, sistem pengaturan lampu lalu lintas menggunakan sistem waktu. Kekurangan yang terdapat dalam sistem waktu ini yaitu terjadinya waktu yang tidak sesuai dengan kepadatan kendaraan di persimpangan. Misalnya, arus kendaraan di salah satu persimpangan sudah kosong, namun lampu hijau masih menyala. Juga sebaliknya, arus kendaraan yang padat terkadang hanya diberi waktu lampu hijau yang sedikit[2].

Untuk mengatasi hal tersebut, sistem pengaturan lalu lintas berdasarkan jumlah kepadatan kendaraan di setiap persimpangan. Sistem ini dibuat menggunakan deteksi jumlah kendaraan berbasis *Object Detection*. Pada penelitian sebelumnya telah dilakukan pemantuan lalu lintas dengan metode YOLO, tetapi hanya mendeteksi kendaraan pada gambar dan tidak menghitung jumlah kendaraan[3]. YOLO dikenal sebagai metode yang cepat dan akurat untuk mendeteksi objek secara *real time*[4]. Penelitian pada Tugas Akhir ini menggunakan metode *You Only Look Once* (YOLO), YOLO yang mana menerapkan jaringan syaraf tunggal pada keseluruhan gambar. Jaringan ini akan membagi gambar menjadi wilayah-wilayah kemudian memprediksi kotak pembatas dan probabilitas[5]. Dengan hal tersebut diharapkan pengaturan sistem lalu lintas dapat lebih efisien dan efektif guna teciptanya ketertiban dalam berkendara.

2. Tinjauan Pustaka

2.1. *Object Detection*

Object detection adalah proses mencari dan menemukan objek dalam gambar atau video dengan *bounding box*, sekaligus tugas dari *object detection* yaitu dapat mengenali beberapa objek dan menentukan posisi objek-objek serta lokasinya. *Object detection* dapat memberikan prediksi objek yang terdeteksi yaitu kelas objek, kotak pembatas, dan koordinat untuk setiap objek[6].

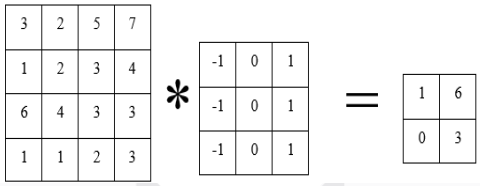
2.2. *Convolutional Neural Network* (CNN)

CNN adalah pengembangan dari *Multilayer Perceptron* (MLP) yang didesain untuk mengolah data dua dimensi. CNN termasuk dalam jenis *Deep Neural Network* karena kedalaman jaringan yang tinggi dan banyak diaplikasikan pada data citra. CNN bisa digunakan untuk mendeteksi dan mengenali objek pada sebuah image[7].

2.2.1. *Convolutional Layer*

Convolutional Layer merupakan *layer* utama pada CNN. Layer ini melakukan operasi konvolusi dengan matriks kernel atau matriks *filter* untuk mengekstraksi fitur dari citra *input*, salah contoh nilai fitur dari sebuah citra tersebut yaitu *edge detection*, *corner detection*, dan sebagainya[8].

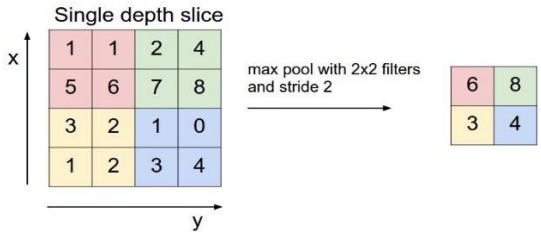
$$h(x) = f(x) * g(x) \tag{1}$$



Gambar 1. Operasi konvolusi matriks tanpa *padding* dan stride 1

2.2.2. *Pooling Layer*

Pooling Layer merupakan tahap setelah *Convolutional Layer*. *Pooling Layer* terdiri dari sebuah *filter* dengan ukuran dan *stride* tertentu. Setiap pergeseran akan ditentukan oleh jumlah *stride* yang akan digeser pada seluruh area *feature map* atau *activation map*. Dalam penerapannya, *pooling layer* yang biasa digunakan adalah *Max Pooling* dan *Average Pooling*. Sebagai contoh, apabila kita menggunakan *Max Pooling* 2×2 dengan *Stride* 2, maka pada setiap pergeseran *filter*, nilai yang diambil adalah nilai yang terbesar pada area 2×2 tersebut, Sedangkan *Average Pooling* akan mengambil nilai rata-rata[9].

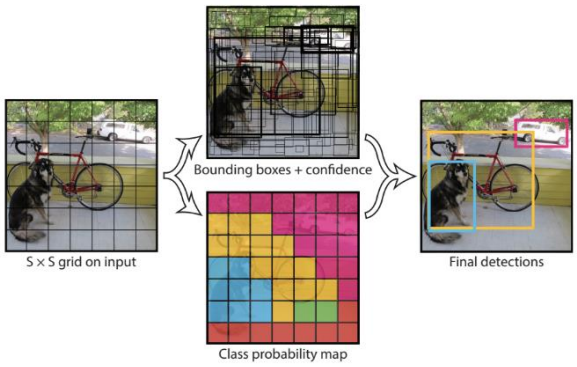


Gambar 2. *Max Polling*[10]

2.3. *You Only Look Once (YOLO)*

You Only Look Once atau YOLO adalah jaringan syaraf yang pintar untuk melakukan deteksi secara *real-time*. YOLO menerapkan jaringan syaraf tunggal pada keseluruhan gambar. Jaringan ini akan membagi gambar

menjadi wilayah-wilayah kemudian memprediksi kotak pembatas dan probabilitas, untuk setiap kotak wilayah pembatas ditimbang probabilitasnya untuk mengklasifikasian sebagai objek atau bukan[5]
.



Gambar 3. Ilustrasi algoritma You Only Look Once[5]

YOLO akan membagi *input* gambar menjadi *grid* berukuran $S \times S$ [11], dimana nilai S adalah 7 dengan *input* gambar berukuran 448×448. Untuk mendapatkan *bounding box*, akan dilakukan konvolusi dari inputan gambar. Sebuah *bounding box* memiliki 5 nilai yang perlu disimpan, koordinat x, koordinat y, lebar (*width*), tinggi (*height*), dan *confidence score* (nilai probabilitas *bounding box* yang bersangkutan memiliki sebuah objek).

2.4. *Simple Online and Realtime Tracking*

Simple Online and Realtime Tracking (SORT) adalah algoritma yang efektif dan sederhana untuk melakukan *tracking* pada *multiple objects* secara *real time.* Pada saat melakukan *tracking*, SORT akan menampilkan nomor (*id object*) setiap *bounding box* yang terdeteksi pada *real time tracking. Python* membutuhkan 3 *library* untuk algoritma SORT yaitu *scikit-learn, scikit-image,* dan *FilterPY*[12].

2.5. *Epoch*

Epoch adalah *hyperparameter* yang menentukan berapa kali algoritma *machine learning* akan bekerja mengolah seluruh *dataset training*[13]. Dalam proses *training*, untuk mendapatkan nilai *weight* yang akurat dibutuhkan lebih dari 1 *epoch* dan setiap 1 *epoch* yang sudah selesai akan memberikan informasi seperti *training loss, validation loss* dan waktu komputasi.

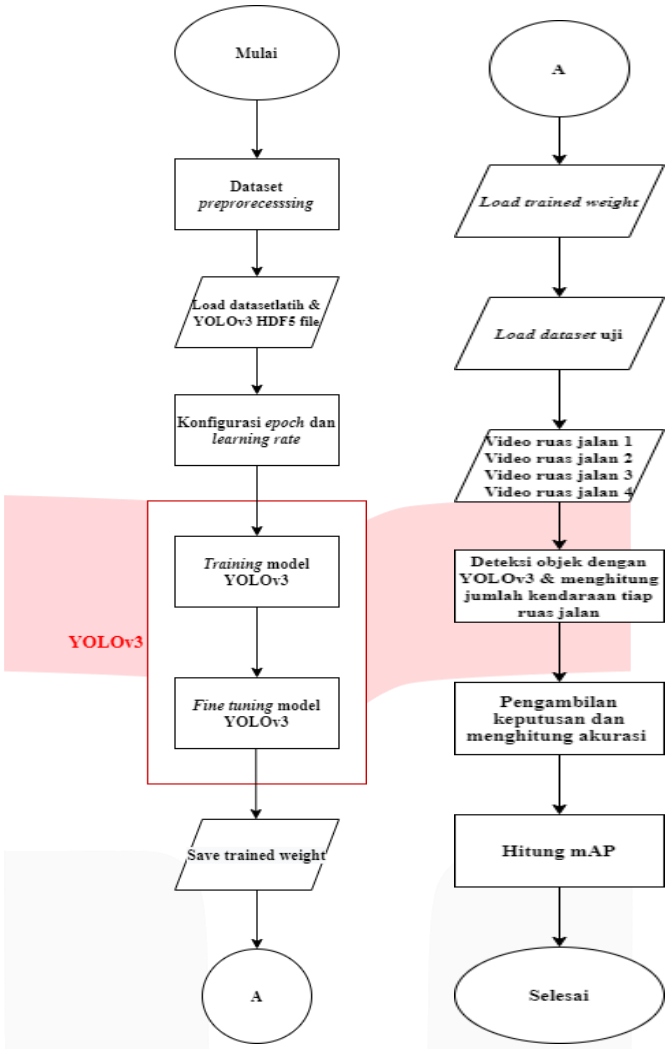
2.6. *Learning Rate*

Learning rate merupakan salah satu *hyperparameter* yang digunakan untuk mengatur *weight* dari *neural network* untuk mengurasi *loss gradient*. Hal ini menentukan seberapa lama waktu proses *training* untuk mendapatkan *weight* yang optimal[14]. *Learning rate* semakin besar mengakibatkan proses *training* akan lebih cepat, namun akan membuat hasil yang kurang optimal. Sedangkan untuk *learning rate* semakin kecil proses *training* akan lebih lama dan memungkinkan model sistem untuk melawatkan banyak nilai-nilai bobot yang penting selama proses *training*.

2.7. *Batch Size*

Batch size adalah banyaknya sample *dataset* yang akan di masukan dalam sekali *training*. Meningkatkan ukuran *batch size* semakin besar, maka semakin besar memori yang dibutuhkan dan waktu yang diperlukan lebih lama untuk sekali *training* tergantung pada perangkat yang digunakan.

3. Desain Model Sistem



Gambar 2. Diagram Alir Sistem.

3.1. *Dataset Preprocessing*

Tahap *dataset preprocessing* merupakan tahap pengumpulan *dataset* untuk dijadikan data latih dan data uji. Skema pembagian *dataset* ditunjukkan pada Tabel 1.

Tabel 1. Skema *Dataset*

Objek	Data Latih	Data Validasi	Data Uji
Mobil	312	312	156
Jumlah	624		156

3.2. *Load Dataset* Latih dan *Weight* HDF5 YOLOv3

Dataset latih berupa citra dengan label *ground truth* yang sudah dibuat, selanjutnya dikumpulkan satu *file* anotasi berformat txt yang menunjukan lokasi masing-masing citra latih yang sudah diberi label *ground truth*, lokasi 4 titik (*x-min, xmax, y-min,* dan *y-max*). *File pre-trained weight* YOLOv3 yang telah diunduh kemudian dikonversi ke dalam file keras HDF5. Pengkonversian *format file* dilakukan karena file *weight* yang didukung pada arsitektur YOLOv3 adalah HDF5. File HDF5 akan digunakan saat proses pendeteksian objek dan pengujian model yang telah dilatih.

3.3. Skema Konfigurasi *Hyperparameter* Model

Konfigurasi *hyperparameter* yang diimplementasikan pada tugas akhir ini meliputi pada *epoch, batch size dan learning rate*. Ukuran *batch size* yang digunakan yaitu *batch size* 32 saat proses *training* dan 4 saat *fine-tuning* pada 4 model. Skema pembagian konfigurasi model yang penulis rancang dapat dilihat pada Tabel 2.

Tabel 2. Skema Konfigurasi *Hyperparameter*

Skema	*Training initial epoch*	*Training epoch*	*Train Learning rate*	*Fine tuning initial epoch*	*Fine tuning epoch*	*Fine tuning Learning rate*
A	0	30	0.001	30	40	0.0001
B	0	35	0.0001	35	50	0.00001
C	0	40	0.001	40	60	0.0001
D	0	50	0.0001	50	100	0.00001

3.4. *Training Model*

Dalam mendeteksi objek, YOLO bekerja dengan awalan mengubah citra masukan menjadi 416x416. Pada sistem ini menggunakan Darknet-53 sebagai ekstrasi fitur dan deteksi objek pada citra. Citra masukan akan di ekstrak fiturnya menggunakan Darknet53 pada layer 75, kemudian setelah fitur didapatkan akan dijadikan sebagai input ke *detector* yang berfungsi untuk memprediksi *bounding box* dan *class* objek yang terdekteksi. Pada proses *training* model, hasil *output* model *training* nanti akan berbentuk file H5.

Selanjutnya masuk ke proses *fine tuning* yang berfungsi untuk optimasi model yang telah dilatih dengan mengurangi kemungkinan *overfitting* sekecil mungkin. Pada proses ini *weight* dari proses pelatihan secara terus-menerus diupdate sebanyak nilai *fine tuning epoch* yang telah ditentukan. Oleh karena itu jika model mengalami *overfitting* maka proses *fine tuning* akan otomatis berhenti dan tidak sesuai dengan nilai *fine tuning epoch* yang diberikan di awal.

3.5. Pengujian Model

Setelah proses *training* model, model yang berupa file *weight* HDF5 tersebut akan digunakan untuk mendeteksi objek yang berupa mobil dan menghitung jumlah mobil yang melewati garis virtual pada video. Model ini juga nanti akan digunakan untuk menghitung nilai AP dengan *dataset* uji berupa citra yang telah disediakan.

3.6. Pengimplementasian Model

Setelah mendapatkan hasil nilai AP keempat model yang telah dilatih, lalu memilih model yang memiliki hasil yang AP terbaik dari keempat model tersebut. Mempersiapkan 4 video ruas jalan yang masing-masing ruas jalan akan diambil data untuk mendeteksi objek dan menghitung jumlah kendaraan. Pengambilan video berjarak 70 meter dari titik pengambilan video. Pada ruas jalan terdapat mobil dengan jumlah yang berbeda, sehingga dapat menghitung jumlah kendaran. Kamera digunakan untuk merekam ruas jalan sebagai input sistem. Objek berupa mobil ditangkap oleh kamera dan selanjutnya akan dilakukan proses deteksi objek dan menghitung jumlah kendaraan. Setelah jumlah kendaraan dihitung, dipilih ruas jalan yang memiliki jumlah kendaraan paling banyak untuk pengambilan keputusan urutan lampu lalu lintas. Pengambilan keputusan merupakan proses untuk mengatur lampu lalu lintas. Sistem akan membandingkan antara 4 ruas jalan, ruas jalan yang memiliki jumlah kendaraan yang lebih banyak akan didahulukan (lampu hijau). Objek yang termasuk ke dalam perhitungan merupakan objek yang melewati garis virtual pada video. Pengujian menghitung jumlah kendaran dilakukan pada siang hari agar objek mudah terdeteksi.

3.7. Analisis Parameter Perfomansi

Desain sistem yang akan dirancang dengan menggunakan metode YOLO terdapat 2 parameter perfomansi yang akan dianalisis dan diperhitungkan untuk menilai kinerja sistem. Parameter perfomansinya yaitu :

3.7.1 Akurasi

Akurasi merupakan salah satu parameter perfomansi pada video *input*. Parameter ini akan mengukur tingkat kedekatan dengan data objek mobil pada video yang dideteksi.

$$A = \frac{T}{N} \text{ X } 100\% \quad (1)$$

dimana A merupakan tingkat akurasi, T merupakan bentuk objek yang akurat, dan N merupakan total keseluruhan data.

3.7.2 Average Precision (AP)

Average Precision (mAP) merupakan nilai rata-rata dari *Average Precision* (AP) yang digunakan sebagai parameter model yang telah dilatih menggunakan *dataset* tertentu. Untuk mengetahui AP perlu diketahui nilai *presisi* dan *recall* yang dapat dihitung menggunakan persamaan:

$$\text{Presisi} = \frac{TP}{\text{TP+FP}} \quad \text{Recall} = \frac{TP}{\text{TP+FN}} \tag{2}$$

Kemudian nilai *presisi* diplot terhadap nilai *recall* untuk mendapatkan pola yang berbentuk zigzag. Untuk menghitung nilai AP, hasil plot berbentuk zig-zag tersebut akan dihaluskan, dimana nilai *presisi* (*p*) pada suatu *recall* (*r*) akan disamakan dengan nilai *presisi* maksimum pada *recall* selanjutnya (*r'*) atau dituliskan menggunakan persamaan berikut:

$$P_{inter}(r_{n+1}) = \max p(r'); r' \geq r_{n+1} \tag{3}$$

Nilai AP didapatkan dengan menghitung area kurva pada nilai *recall*, ketika nilai *presisi* maksimum jatuh. Sehingga area yang terhitung merupakan area dimana zigzag dihaluskan. Selanjutnya, nilai AP didapatkan menggunakan persamaan berikut:

$$AP = \frac{1}{11} \times (APr(0) + APr(0.1) \ldots + APr(1.0)) \tag{4}$$

Kemudian untuk mendapatkan nilai mAP dapat dihitung menggunakan persamaan:

$$\text{mAP} = \sum_{i=1}^{N} \frac{AP_i}{N} \, x \, 100\% \tag{5}$$

Dimana N merupakan banyak kelas yang terdapat pada suatu model yang telah dilatih

4. Hasil dan Analisis

4.1 Hasil dan Analisis Pengujian Model

Pengujian model dan analisis hasil pengujian menggunakan parameter performansi *Average Precission* (AP).

4.1.1 Pengujian Terhadap AP

Pengujian dilakukan menggunakan 4 model dari hasil pelatihan model dengan menggunakan *batch size* 4 dan 32. Hasil pengujian sistem dipilih hasil yang terbaik, yaitu memilih AP terbaik.

Tabel 3. Hasil Pengujian mAP

Skema	Average Precission
A	0.79
B	0.57
C	0.89
D	0.81

Pada tabel diatas nilai AP tertinggi yang dihasilkan sebesar 0.89 pada model dengan menggunakan skema C, sedangkan nilai mAP terendah yang dihasilkan sebesar 0.57 pada model dengan menggunakan skema B. Pada saat proses proses *fine tuning* dengan menggunakan *batch size* 4, terjadi penurunan nilai *learning rate* secara otomatis oleh *Tensorflow* untuk mendapat nilai *training loss* dan *validation loss* yang lebih baik dari proses *epoch* yang sebelumnya. Saat nilai *training loss* dan *validation loss* tidak dapat diperkecil lagi bahkan dengan nilai *learning rate* yang lebih kecil dari sebelumnya, maka *TensorFlow* otomatis akan menghentikan proses *fine tuning* walaupun nilai *epoch* yang sebelumnya telah diberikan tidak mencapai dengan nilai yang telah diberikan sebelumnya. Berikut tabel yang menunjukan *epoch* dan penyesuaian *learning rate*, sehingga didapatkan model dengan *weight* paling optimal**.**

4.1.1.1 Konfigurasi *Learning Rate*

Tabel 4. Hasil Konfigurasi Learning Rate

Epoch	Learning Rate	mAP
40	0.0001	0.79
50	0.00001	0.57
60	0.0001	0.89
100	0.00001	0.81

Pada tabel diatas ditampilkan hasil konfigurasi *learning rate* dengan menggunakan *batch size* 4, model dengan nilai AP tertinggi yaitu 0.89 pada *learning rate* 0.0001. Pada saat *fine tuning*, sistem akan otomatis mengubah *learning rate* apabila nilai *validation loss* dan *loss* tidak mengalami penurunan setelah proses 3 kali *epoch*. Perubahan otomatis mengubah *learning rate* dilakukan untuk menghindari terjadinya *overfitting*, menyebabkan keberagaman nilai *learning rate* pada setiap model. Apabila *learning rate* terlalu tinggi menyebabkan sistem akan lebih mengalami *loss* yang tidak stabil, namun apabila terlalu rendah, sistem perlu waktu yang jauh lebih banyak agar mendapatkan hasil yang optimal dan besar kemungkinan memori perangkat akan *overload* sebelum hasil optimal didapatkan. *Learning rate* 0.0001 menghasilkan nilai AP yang lebih tinggi dibandingkan dengan learning rate 0.00001.

4.1.1.2 Konfigurasi Epoch

Tabel 5. Hasil Konfigurasi *Epoch*

Skema	Initial Epoch	Used Epoch	mAP
A	40	40	0.79
B	50	50	0.57
C	60	60	0.89
D	100	95	0.81

Pada tabel 4.4 ditampilkan hasil konfigurasi *epoch* dengan menggunakan *batch size* 4, model dengan nilai AP tertinggi yaitu 0.89 dengan menggunakan skema C dengan *epoch* sebesar 60. Sedangkan untuk nilai AP terendah yaitu dengan menggunakan skema B dengan *epoch* sebesar 50. Pada proses ini dapat di lihat pada skema D dengan nilai *initial epoch* 100 dan hanya nilai 95 *epoch* yang digunakan. Nilai *epoch* yang digunakan tidak sama dengan yang diberikan pada diawal, dikarenakan sistem melakukan *early stopping* ketika nilai *validation loss* tidak dapat turun lagi setelah 10 kali *epoch* untuk menghindari *overfitting* pada model, sehingga akan berhenti di *epoch* tersebut. Jumlah nilai *epoch* yang terlalu besar tidak bisa menjamin nilai AP lebih tinggi, dikarenakan sistem akan berhenti jika nilai *validation loss* tidak bisa turun lagi.

4.2 Hasil Pengujian dan Analisis Akurasi

Setelah mendapatkan hasil nilai AP yang terbaik, lalu memilih model yang memiliki hasil yang AP terbaik. Pengujian akurasi dilakukan menggunakan 4 video ruas jalan, untuk menghitung jumlah kendaraaan. Objek yang termasuk pada perhitungan jumlah kendaraan untuk pengujian akurasi, objek yang terdeteksi harus melewati garis virtual pada video.

Garis Virtual		20%	30%	40%	50%	60%	70%	80%
Ruas Jalan 1	Akurasi	89,47%	100%	95%	100%	72,22%	77,77%	44,44%
Ruas Jalan 2		75%	100%	87,5%	81,25%	60%	46,66%	46,66%
Ruas Jalan 3		80,95%	95,23%	100%	85,71%	66,66%	95,45%	36.36%

Ruas Jalan 4		80%	100%	88,23%	62,5%	75%	43,75%	37,5%
Hasil Rata-Rata		81,35%	98,80%	92,60%	82,36%	68,47%	65,90%	41,24%

5. Kesimpulan

YOLOv3 mampu mendeteksi objek dan menghitung jumlah kendaraan berupa mobil, dengan hasil pengujian model terbaik dengan menggunakan parameter *average precission* sebesar 0,89 pada model skema C. Nilai *learning rate* pada model tersebut yaitu 0,0001, nilai *epoch* sebesar 60 dan *batch* size sebesar 4. Saat proses *training* model terdapat model dengan skema D, *epoch* yang diberikan saat awal tidak sesuai dengan yang digunakan, karena untuk menghindari *overfitting*. Hasil pengujian akurasi didapat nilai rata-rata akurasi tertinggi adalah 98,80% dengan menggunakan garis virtual 30% dari ujung atas frame video.

Daftar Pustaka:

[1] B. P. Statistik, "Jumlah Penduduk Indonesia Diproyeksikan Mencapai 270 Juta pada 2020," 2019. https://databoks.katadata.co.id/datapublish/2019/09/13/jumlah-penduduk-indonesia-diproyeksikan-mencapai-270-juta-pada-2020 (accessed Sep. 26, 2019).

[2] Q. Hidayati, "Kendali Lampu Lalu Lintas dengan Deteksi Kendaraan Menggunakan Metode Blob Detection," *J. Nas. Tek. Elektro dan Teknol. Inf.*, vol. 6, no. 2, 2017, doi: 10.22146/jnteti.v6i2.318.

[3] M. Harahap *et al.*, "Sistem Cerdas Pemantauan Arus Lalu Lintas Dengan YOLO (You Only Look Once v3)," *Semin. Nas. APTIKOM*, p. 2019, 2019.

[4] J. Redmon and A. Farhadi, "Yolo V2.0," *Cvpr2017*, no. April, pp. 187–213, 2017, [Online]. Available: http://www.worldscientific.com/doi/abs/10.1142/9789812771728_0012.

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 779–788, 2016, doi: 10.1109/CVPR.2016.91.

[6] M. Simon, S. Milz, K. Amende, and H. M. Gross, "Complex-YOLO: An euler-region-proposal for real-time 3D object detection on point clouds," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 11129 LNCS, pp. 197–209, 2019, doi: 10.1007/978-3-030-11009-3_11.

[7] W. Sugiarto, Y. Kristian, and E. R. Setyaningsih, "Estimasi Arah Tatapan Mata Menggunakan Ensemble Convolutional Neural Network," *Teknika*, vol. 7, no. 2, pp. 94–101, 2018, doi: 10.34148/teknika.v7i2.126.

[8] M. Liang and X. Hu, "Recurrent convolutional neural network for object recognition," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 07-12-June, no. Figure 1, pp. 3367–3375, 2015, doi: 10.1109/CVPR.2015.7298958.

[9] A. Santoso and G. Ariyanto, "Implementasi Deep Learning Berbasis Keras Untuk Pengenalan Wajah," *Emit. J. Tek. Elektro*, vol. 18, no. 01, pp. 15–21, 2018, [Online]. Available: http://journals.ums.ac.id/index.php/emitor/article/view/6235.

[10] S. Ilahiyah and A. Nilogiri, "Implementasi Deep Learning Pada Identifikasi Jenis Tumbuhan Berdasarkan Citra Daun Menggunakan Convolutional Neural Network," vol. 3, no. 2, pp. 49–56, 2018.

[11] R. Huang, J. Pedoeem, and C. Chen, "YOLO-LITE: A Real-Time Object Detection Algorithm Optimized for Non-GPU Computers," *Proc. - 2018 IEEE Int. Conf. Big Data, Big Data 2018*, pp. 2503–2510, 2019, doi: 10.1109/BigData.2018.8621865.

[12] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," *Proc. - Int. Conf. Image Process. ICIP*, vol. 2016-Augus, pp. 3464–3468, 2016, doi: 10.1109/ICIP.2016.7533003.

[13] M. S. Wibawa, "Pengaruh Fungsi Aktivasi , Optimisasi dan Jumlah Epoch Terhadap Performa Jaringan Saraf Tiruan," *J. Sist. dan Inform.*, vol. 11, no. 2, pp. 167–174, 2017, doi: 10.13140/RG.2.2.21139.94241.

[14] L. Luo, Y. Xiong, Y. Liu, and X. Sun, "Adaptive gradient methods with dynamic bound of learning rate," *7th Int. Conf. Learn. Represent. ICLR 2019*, no. 2018, pp. 1–19, 2019.