ANALISIS KINERJA PROTOTIPE *TRAFFIC SIGN RECOGNITION* UNTUK SISTEM *AUTONOMOUS CAR* MENGGUNAKAN *YOU ONLY LOOK ONCE*

PERFORMANCE ANALYSIS OF PROTOTYPING TRAFFIC SIGN RECOGNITION FOR AUTONOMOUS CAR SYSTEM BY USING YOU ONLY LOOK ONCE

Nur Cahyo Kuncoro[1], Suryo Adhi Wibowo[2], Koredianto Usman[3]
[1,2,3]Prodi S1 Teknik Telekomunikasi, Fakultas Teknik Elektro, Universitas Telkom
[1]**nurcahyokuncoro@student.telkomuniversity.ac.id**, [2]**suryoadhiwibowo@telkomuniversity.ac.id**, [3]**koredianto.usman@gmail.com**

Abstrak

Autonomous car adalah sistem mobil yang dikendalikan dengan komputer yang dapat memandu, beradaptasi dengan lingkungan dan beroperasi tanpa interaksi manusia. Ini membuat keadaan lalu lintas semakin rumit sehingga kecerdasan buatan seperti memberikan informasi rambu, dan membantu dalam kontrol kendaraan sangat dibutuhkan untuk memastikan keselamatan berkendara. Tugas Akhir ini akan merancang sebuah sistem yang dapat mengenali rambu lalu lintas dengan metode You Only Look Once (YOLO). YOLO merupakan pendeteksi objek dengan menggunakan convolutional network yang hanya akan dilewatkan satu kali saja. Berbeda dengan convolutional network pada umumnya yang melewatkan ribuan network untuk mendapatkan satu citra dengan komputasi yang cukup lama. Tugas akhir ini akan menggunakan arsitektur YOLO9000 dengan dataset sebanyak 3 class, yaitu rambu belok kanan, belok kiri, dan stop. Konfigurasi sistem yang digunakan adalah learning rate, batch size, dan step training. Dataset terdiri dari 384 citra data latih dan 1920 citra data uji. Dalam Tugas Akhir ini, konfigurasi sistem terbaik didapatkan pada konfigurasi learning rate 0.00002, batch size 8, dan step training 20K dengan hasil akurasi sebesar 93,41%.

Kata kunci : You Only Look Once, Traffic Sign Recognition, Akurasi.

Abstract

Autonomus car is a computer-controlled car system that can guide, adapt to the environtment, and operate without human interaction. This make traffic condition more complicated so that artificial intelligence such as providing sign information and assisting in vehicle control is needed to ensure driving safety. This final project, a system is designed to recognize traffic sign with You Only Look Once (YOLO) method. YOLO is object detection that is using only once convolutional network. It is different with convolutional network in general that spend thousand of networks to get an image with long computing time. This final project will use the YOLO9000 architecture with a dataset of 3 classes, such as turn right, turn left, and stop. The system configuration used is learning rate, batch size, and step training. The dataset consist of 384 training data and 1920 test data. In this Final Project, the best system configuration is obtained in the 0.00002 learning rate, 8 batch size, and 20K training step with an accuracy of 93.41%.

Keywords: You Only Look Once, Traffic Sign Recognition, Accuracy.

1. Pendahuluan

Pada masa ini kebutuhan akan sistem kecerdasan buatan telah meningkat dengan pesat. *Computer vision* merupakan salah satu bidang penelitian yang memiliki paling banyak perkembangan. Tujuannya adalah memberikan komputer untuk memiliki penglihatan seperti manusia seperti *visual tracking*, kamera keamanan, *object detection*, dan lain sebagainnya[1][2][3]. Ini merupakan salah satu latar belakang diciptakannya *autonomous* car[4]. *Autonomous car* adalah sistem mobil yang dikendalikan dengan komputer yang dapat memandu, beradaptasi dengan lingkungan dan beroperasi tanpa interaksi manusia [5]. Ini membuat keadaan lalu lintas semakin rumit sehingga kecerdasan buatan seperti memberikan informasi rambu, dan membantu dalam kontrol kendaraan sangat dibutuhkan untuk memastikan keselamatan berkendara[6].

Penelitian telah banyak dilakukan terutama pada pengenalan rambu lalu lintas. Saat ini banyak metode yang sedang dikembangkan menggunakan *deep learning*[8] seperti seperti R-CNN[9], Faster R-CNN[8]. Pada penelitian [10] dan [11] metode Faster R-CNN dinilai lebih cepat dan akurat dari R-CNN dalam pengenalan objek namun keduanya masih jauh dari performa *real-time* yang baik karena *frame per second* (fps) yang didapat adalah 7 fps dan 6 fps. Karena hal tersebut pada tugas akhir ini penulis mengusulkan menggunakan metode YOLO untuk pengenalan rambu lalu lintas yang dapat diimplementasikan pada prototipe sistem *autonomous car*.

YOLO dikenal sebagai metode yang cepat dan akurat untuk mendeteksi dan mengenali objek secara *real time*[11]. Pada tugas akhir ini akan dilakukan perancangan sistem yang dapat mengenali rambu lalu lintas.

Penulis akan membuat sistem untuk mengenal rambu lalu lintas pada prototipe sistem *autonomous car* dengan menggunakan metode YOLO. Konfigurasi parameter yang digunakan yaitu *batch size, learning rate* dan *step training*. Tugas Akhir ini melakukan uji coba terhadap setiap konfigurasi yang ditentukan. Setelah dilakukan percobaan, selanjutnya dianalisis untuk mendapatkan konfigurasi terbaik.

2. Dasar Teori dan Metodologi

2.1 *Autonomous car*

Sistem *Autonomous car* adalah sistem kendaraan yang mampu merasakan lingkungan sekitarnya dan dapat bergerak dengan aman tanpa campur tangan manusia[12]. Konsep dari *autonomous car* adalah menghadirkan layanan yang aman, dan mudah digunakan oleh pengguna mobil[5].

Ada beberapa teknologi yang digunakan pada *autonomous car*. Setiap teknologi memiliki perbedaan tujuan. Salah satu jenis teknologi yang digunakan pada *autonomous car* adalah *Computer Vision in Autonomous Cars*. *Computer vision* adalah fitur dalam *autonomous car* yang mampu melihat jalan dan mendeteksi hambatan di depan dan di sekitarnya, baik itu mobil lain, pejalan kaki, atau hambatan lainnya. Ini memungkinkan mobil untuk mengemudi di sepanjang jalan secara aman[5].*Sensors and Control* pada *autonomous car* digunakan untuk mendeteksi rambu lalu lintas dan mampu memberikan informasi pada pengendara[5].

2.2 *You Only Look Once*

YOLO merupakan metode deteksi objek yang cocok digunakan untuk realtime processing [10]. Metode deteksi objek bertugas untuk menentukan lokasi dimana objek berada pada suatu citra dan menentukan class objek[13]. Metode seperti R-CNN menggunakan beberapa langkah untuk melakukan tugas tersebut sehingga performansinya lambat karena setiap komponen perlu dilatih secara terpisah. Berbeda dengan YOLO yang menggunakan single neural network[10].

YOLO akan membagi citra input menjadi 7×7 grid cells untuk menyesuaikan dengan penggunaan PASCAL VOC dataset[10]. Setiap grid cells bertanggung jawab untuk memberikan predicted bounding boxes. Predicted bounding boxes akan memberikan memberikan score keberadaan objek dan score prediksi class dari objek yang terdeteksi.

2.3 *Convolutional Neural Network*

Convolutional Neural Network (CNN) merupakan pengembangan dari *Artificial Neural Network* (ANN). CNN dikenal sebagai salah satu algoritma terbaik pada *Deep Learning* untuk memecahkan masalah *object detection*[14]. Cara kerja CNN menggunakan operasi konvolusi. Secara umum CNN terbagi menjadi 3 *layer*, yaitu *convolutional layer, pooling layer*, dan *full-connected layer*[12]. YOLO9000 hanya menggunakan *convolutional layer* dan *pooling layer.*

2.4 *Learning rate*

Learning rate merupakan salah satu *hyperparameter* yang digunakan untuk melakukan *tuning.* Hal ini dikarenakan *learning rate* dapat meningkatkan efektivitas *training* dan menentukan kecepatan langkah pada setiap iterasi untuk menuju *loss function minimum.* Semakin besar nilai *learning rate* maka semakin cepat prosesi *training* yang dilakukan. Pada penelitian [16] pemilihan *learning rate* yang baik dilakukan dengan *trial and error* yaitu memulai dari nilai *learning rate* yang kecil sehingga didapatkan performansi yang baik pada sistem.

2.5 *Batch size*

Batch size merupakan *hyperparameter* yang menentukan seberapa banyak *sample* yang akan dimasukkan pada saat *training*. Berdasarkan hasil penelitian pada [17], semakin besar *batch size* maka sistem akan mempelajari fitur yang lebih banyak sehingga akan memiliki akurasi yang lebih tinggi.

2.6 *Training Step*

Training step merupakan besarnya iterasi yang dilakukan pada semua data *training. Training step* juga berperan penting unuk menentukan *checkpoint* sehingga apabila *training* berhenti karena adanya kendala, sistem dapat melanjutkan *training* berdasarkan *checkpoint* yang disimpan.

2.7 Perhitungan Parameter Performansi

Parameter performansi digunakan sebagai tolak ukur kinerja suatu sistem. Adapun parameter performansi yang digunakan sebagai acuan dalam tugas akhir ini adalah akurasi. Secara umum performansi dalam tugas akhir ini diukur berdasarkan hasil yang didapatkan dari proses *training* dan prediksi *dataset* uji.

a. Akurasi

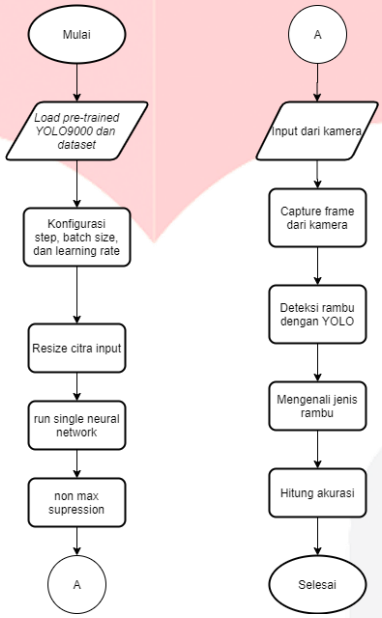
Akurasi didefinisikan sebagai presentase keberhasilan sistem dalam memprediksi suatu citra sesuai dengan kelasnya. Adapun persentase akurasi dapat dihitung dengan persamaan sebagai berikut,

$$\eta = \frac{B}{C} \times 100\% \quad (1)$$

sebagai keterangan, η adalah nilai akurasi, B adalah *rambu yang benar*, dan C adalah *total keseluruhan data*.

3. Perancangan dan Pembentukan Model Sistem

Tugas akhir ini merancang sistem *object recognition* dengan objek rambu lalu lintas berupa perintah belok kanan, belok kiri, dan *stop*. Skema dari tugas akhir ini menggunakan algoritma YOLO dengan arsitektur YOLO9000. Alur kerja sistem dapat dilihat pada Gambar 1.



Gambar 1. *Flowchart* rancangan sistem.

3.1 *Load Pre-Trained YOLO9000 dan Dataset*

Pre-trained YOLO9000 telah dilatih dengan PASCAL VOC yang memiliki 20 class. Sedangkan tugas akhir ini menggunakan dataset yang memiliki 3 class, yaitu rambu perintah belok kanan, belok kiri, dan *stop*. Proses *load pre-trained* YOLO9000 dan *dataset* hanya dilakukan satu kali saat mulainya sistem. Tugas akhir ini menggunakan *dataset* berupa citra rambu lalu lintas berupa perintah belok kanan, belok kiri, dan *stop*. Citra dataset diambil dari berbagai jarak. Dataset dibagi menjadi tiga yaitu data latih, data validasi, dan data uji. Dengan rincian seperti Tabel 1.

Tabel 1. Sistematika Pembagian Data.

Kelas	Data latih (*sample*)	Data Validasi (*sample*)	Data Uji (*sample*)
Belok Kanan	128	64	640
Belok Kiri	128	64	640
Stop	128	64	640

3.2 Konfigurasi *Hyperparameter*

Konfigurasi *hyperparameter* yang diterapkan meliputi pengaturan pada jumlah *learning rate*, yaitu 0.00001, 0.000015, dan 0.00002. *Batch size* yang digunakan pada masing – masing *learning rate* adalah 2, 4, dan 8. Selanjutnya akan diuji pada *step* 10K, 15K, dan 20K.

3.3 *Input* dari Kamera

Input dalam sistem ini menggunakan video dengan ruang warna RGB. Data *input* dilakukan dengan meletakkan satu rambu lalu lintas dan kamera bergerak perlahan mendekati rambu sehingga kamera dapat menangkap rambu dari jarak yang berbeda.

3.4 *Capture Frame dari Kamera*

Pada tahap ini dilakukan pengubahan *input* yang sebelumnya video menjadi citra. Tahap ini dilakukan karena diperlukan pengambilan *sampling* dari video.

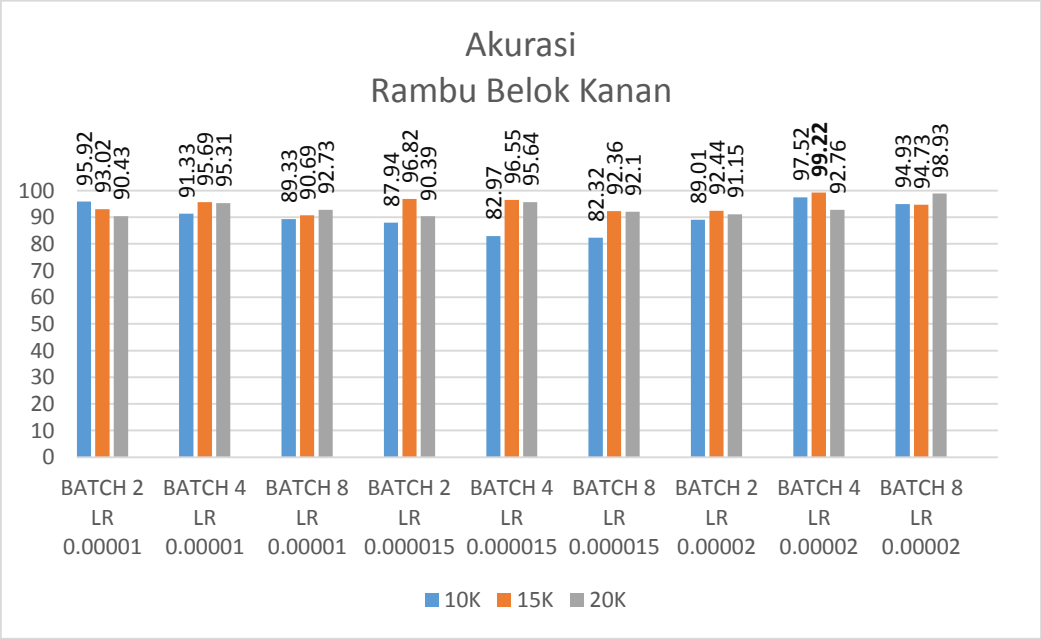
3.5 *Capture Frame dari Kamera*

Proses pengenalan rambu merupakan bagian penting dalam sistem ini. Keluaran sistem ini yaitu menentukan jenis rambu lalu lintas yang dideteksi. Hasil dari rambu yang benar kemudian akan digunakan untuk mendapatkan akurasi.

4. Pengujian Model Sistem

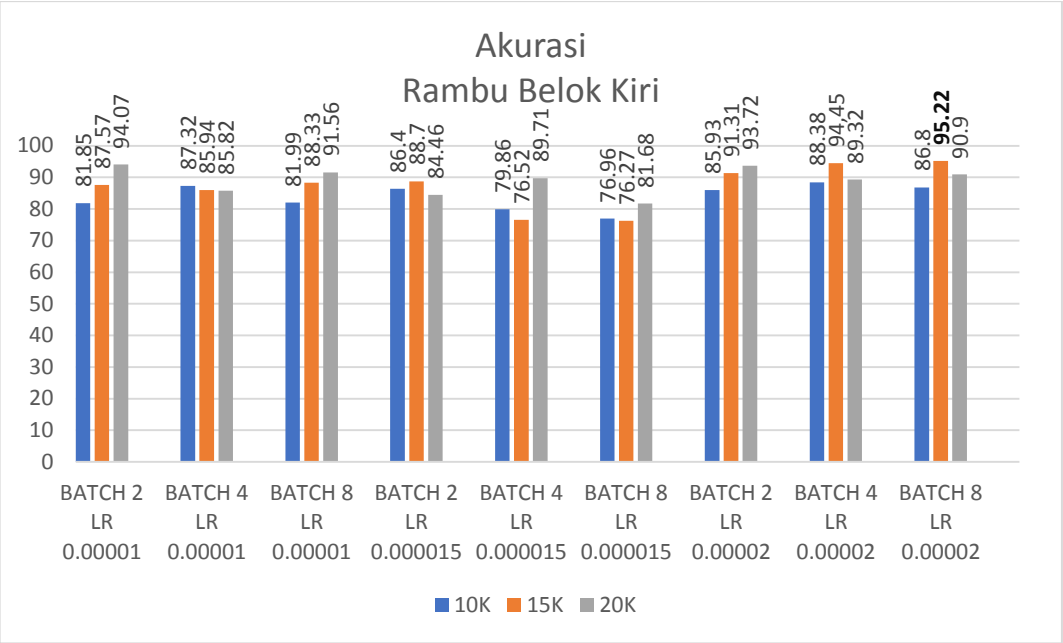
Pengujian dilakukan terhadap hasil prediksi dari setiap yang telah dilatih pada tahap sebelumnya untuk mengetahui pengaruh pengaturan konfigurasi yang dilakukan terhadap parameter akurasi.

4.1 Hasil Pengujian Data



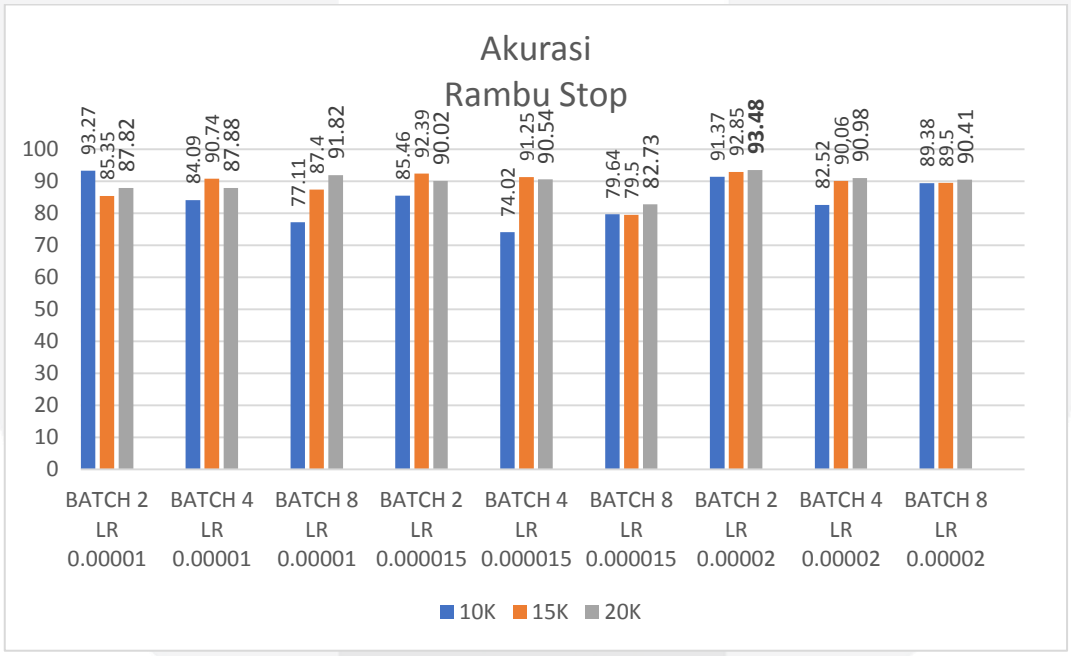
Gambar 2. Akurasi seluruh model rambu belok kanan.

Grafik hasil pengujian sistem pada rambu belok kanan dapat dilihat pada Gambar 2. Dari grafik tersebut dapat dilihat bahwa akurasi tertinggi adalah 99.22%. Akurasi tertinggi pada rambu belok kanan didapat dengan konfigurasi *batch size* 4, learning rate 0.00002 pada *step training* 15K. Pada Gambar 2 dapat dilihat bahwa akurasi dari *learning rate* 0.00001 akurasi pada setiap *step training* cenderung stabil. Sedangkan ketika *learning rate* 0.000015 akurasi pada step 10K turun menjadi dibawah 90%, dan pada learning rate 0.00002 akurasi pada keseluruhan *step training* kembali stabil.



Gambar 3. Akurasi seluruh model rambu belok kiri.

Grafik hasil pengujian sistem pada rambu belok kiri dapat dilihat pada Gambar 3. Dari grafik tersebut dapat dilihat bahwa didapatkan akurasi diatas 80%. Akurasi tertinggi pada rambu belok kiri adalah 95,22% didapat dengan konfigurasi *batch size* 8, learning rate 0.00002 pada *step training* 15K. Pada Gambar 3 dapat dilihat bahwa akurasi dari *learning rate* 0.00001 akurasi pada setiap *step training* cenderung stabil. Sedangkan ketika *learning rate* 0.000015 akurasi pada batch size 2 meningkat sedangkan pada batch size 4 dan 8 turun, dan pada learning rate 0.00002 akurasi pada keseluruhan *step training* meningkat dan kembali stabil.



Gambar 4. Akurasi seluruh model rambu *stop*.

Grafik hasil pengujian sistem pada rambu *stop* dapat dilihat pada Gambar 4. Dari grafik tersebut dapat dilihat bahwa akurasi tertinggi adalah 93,48%. Akurasi tertinggi pada rambu *stop* didapat dengan konfigurasi *batch size* 2, learning rate 0.00002 pada *step training* 20K. Pada Gambar 4 dapat dilihat bawah pada setiap *learning rate*, akurasi yang didapatkan ketika batch size 4 dan 8 cenderung berkurang. Pada learning rate 0.00002 akurasi menurun dan akurasi tiap *step training* cenderung stabil dari model dengan *learning rate* yang lain yang cenderung tidak stabil.

4.2 Analisis Hasil Pengujian

Berikut merupakan analisis hasil pengujian terhadap sistem untuk mendapatkan konfigurasi *learning rate*, *batch size*, dan *step training* terbaik untuk semua rambu lalu lintas. Selanjutnya akan dianalisis untuk mendapatkan parameter performansi terbaik untuk sistem yaitu akurasi.

Pada Tabel 2 ditampilkan rata-rata hasil akurasi pada semua rambu. Akurasi akan semakin baik apabila mendekati 100%. Akurasi tertinggi didapatkan pada learning rate 0.00002, yaitu 91,74%.

Tabel 2. Nilai akurasi pada keseluruhan pengujian.

Step	*Learning rate* 0,00001			*Learning rate* 0,000015			*Learning rate* 0,00002		
	Batch 2	*Batch* 4	*Batch* 8	*Batch* 2	*Batch* 4	*Batch* 8	*Batch* 2	*Batch* 4	*Batch* 8
10K	90,34	87,58	82,81	86,6	78,95	79,64	88,77	89,47	90,37
15K	88,64	90,79	88,8	92,63	94,08	82,71	92,2	94,57	93,15
20K	90,77	89,67	92,03	88,29	91,96	85,5	92,78	91,02	93,41
Average		**89,04**			**86,70**			**91,74**	

Berdasarkan Tabel 2, dapat disimpulkan bahwa pada sistem ini *learning rate* terbaik adalah 0.00002. Selanjutnya pada Tabel 3 ditampilkan hasil konfigurasi *batch size* pada *learning rate* 0.00002. Pada Tabel 3 dapat disimpulkan bahwa rata-rata nilai akurasi pada sistem ini semakin meningkat ketika nilai *batch size* bertambah. Hal ini karena semakin besar *batch size* maka semakin banyak fitur yang dipelajari oleh sistem.

Tabel 3. Nilai akurasi *learning rate* 0.00002.

Step	Learning rate *0,00002*		
	Batch 2	*Batch 4*	*Batch 8*
10K	88,77	89,47	90,37
15K	92,2	94,57	93,15
20K	92,78	91,02	93,41
Average	91,25	91,68667	**92,31**

Pada Tabel 4 ditampilkan nilai akurasi terbaik berdasarkan *learning rate* dan *batch size* yang diujikan. Dapat dilihat bahwa rata-rata akurasi terbaik terdapat pada *learning rate* 0.00002, dan *batch size* 8 berdasarkan analisis sebelumnya.

Tabel 4. Hasil konfigurasi *step training* pada *learning rate 0.00002* dan *batch size 8*.

Step	*Learning rate 0,00002*
	Batch 8
10K	90,37
15K	93,15
20K	**93,41**

Akurasi terbaik yang didapatkan berdasarkan Tabel 4 yaitu, 93,41% pada *step training* 20K. Sehingga konfigurasi dengan *learning rate* 0.00002, *batch size* 8, pada *step training* 20K mendapatkan nilai akurasi yang terbaik pada sistem yang dibuat.

5. Kesimpulan

Pengenalan objek berupa rambu-rambu lalu lintas menggunakan YOLO pada 3 *class* telah berhasil diimplementasikan dan mendapatkan rata-rata nilai akurasi 93,41%. Konfigurasi terbaik didapat saat *learning rate* 0.00002, *batch size* 8 dan *step training* 20K dengan nilai akurasi pada rambu belok kanan, belok kiri, dan *stop* adalah 98,93%, 90,9%, dan 90,41%. Hasil pengujian pada sistem yang dibuat menunjukkan akurasi semakin baik ketika *learning rate* ditingkatkan dari 0.00001 ke 0.00002. Selain itu, semakin besar *batch size*, menunjukkan nilai akurasi yang semakin baik.

6. Daftar Pustaka:

S. A. Wibowo, H. Lee, E. K. Kim, and S. Kim, "Collaborative Learning based on Convolutional Features and Correlation Filter for Visual Tracking," *Int. J. Control. Autom. Syst.*, vol. 16, no. 1, pp. 335–349, 2018, doi: 10.1007/s12555-017-0062-x.

[2] S. A. Wibowo, H. Lee, E. K. Kim, and S. Kim, "Convolutional Shallow Features for Performance Improvement of Histogram of Oriented Gradients in Visual Object Tracking," *Math. Probl. Eng.*, vol. 2017, 2017, doi: 10.1155/2017/6329864.

[3] S. A. Wibowo, H. Lee, E. K. Kim, and S. Kim, "Visual tracking based on complementary learners with distractor handling," *Math. Probl. Eng.*, vol. 2017, 2017, doi: 10.1155/2017/5295601.

[4] K. Jo, J. Kim, D. Kim, C. Jang, and M. Sunwoo, "Development of autonomous car-part i: Distributed system architecture and development process," *IEEE Trans. Ind. Electron.*, vol. 61, no. 12, pp. 7131–7140, 2014, doi: 10.1109/TIE.2014.2321342.

[5] R. Hussain and S. Zeadally, "Autonomous Cars: Research Results, Issues, and Future Challenges," *IEEE Commun. Surv. Tutorials*, vol. 21, no. 2, pp. 1275–1313, 2019, doi: 10.1109/COMST.2018.2869360.

[6] C. Wang, "Research and application of traffic sign detection and recognition based on deep learning," *Proc. - 2018 Int. Conf. Robot. Intell. Syst. ICRIS 2018*, pp. 150–152, 2018, doi: 10.1109/ICRIS.2018.00047.

[7] Y. Han, K. Virupakshappa, and E. Oruklu, "Robust traffic sign recognition with feature extraction and k-NN classification methods," *IEEE Int. Conf. Electro Inf. Technol.*, vol. 2015-June, pp. 484–488, 2015, doi: 10.1109/EIT.2015.7293386.

[8] R. Gavrilescu, C. Fo, C. Zet, and D. Cotovanu, "Faster R-CNN : an Approach to Real-Time Object Detection," pp. 165–168, 2018.

[9] S. Jung, U. Lee, J. Jung, and D. H. Shim, "Real-time Traffic Sign Recognition system with deep convolutional neural network," in *2016 13th International Conference on Ubiquitous Robots and Ambient Intelligence, URAI 2016*, 2016, pp. 31–34, doi: 10.1109/URAI.2016.7734014.

[10] 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.

[11] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 6517–6525, 2017, doi: 10.1109/CVPR.2017.690.

[12] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8689 LNCS, no. PART 1, pp. 818–833, 2014, doi: 10.1007/978-3-319-10590-1_53.

[13] R. Phadnis, J. Mishra, and S. Bendale, "Objects Talk - Object Detection and Pattern Tracking Using TensorFlow," *Proc. Int. Conf. Inven. Commun. Comput. Technol. ICICCT 2018*, pp. 1216–1219, 2018, doi: 10.1109/ICICCT.2018.8473331.

[14] G. Hu *et al.*, "When Face Recognition Meets with Deep Learning: An Evaluation of Convolutional Neural Networks for Face Recognition," *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2016-Febru, pp. 384–392, 2016, doi: 10.1109/ICCVW.2015.58.

[15] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, 2010, doi: 10.1109/TKDE.2009.191.

[16] L. N. Smith, "Cyclical learning rates for training neural networks," *Proc. - 2017 IEEE Winter Conf. Appl. Comput. Vision, WACV 2017*, no. April 2015, pp. 464–472, 2017, doi: 10.1109/WACV.2017.58.

[17] P. M. Radiuk, "Impact of Training Set Batch Size on the Performance of Convolutional Neural Networks for Diverse Datasets," *Inf. Technol. Manag. Sci.*, vol. 20, no. 1, pp. 20–24, 2018, doi: 10.1515/itms-2017-0003.

[18] R. L. Halterman, *Learning to program with Python*. 2011.

[19] Y. Zhang, Y. Chen, C. Huang, and M. Gao, "Object detection network based on feature fusion and attention mechanism," *Futur. Internet*, vol. 11, no. 1, pp. 1–14, 2019, doi: 10.3390/fi11010009.