STEERING, THROTTLE DAN BRAKE PREDICTION PADA SIMULATOR SELF-DRIVING CAR MEMANFAATKAN CNN

Andre Nyoto Raharjo, *Departemen Teknik Informatika, Institut Sains dan Teknologi Terpadu Surabaya,*
Andreas, *Fakultas Ekonomi dan Bisnis, Universitas Pelita Harapan Surabaya.*

***Abstrak*— Manusia dapat mengendarai mobil dengan cara belajar sendiri, belajar dari orang lain atau dengan menirukan cara mengemudi orang lain. Dalam mengemudikan mobil pengemudi akan melihat kondisi jalan dan kondisi mobil sebagai pertimbangan untuk menentukan kontrol kemudi yang akan dijalankan seperti nilai steering angle, throttle dan brake. Dengan pembelajaran atau menirukan kebiasaan manusia, mesin juga mampu untuk melakukan hal yang sama yang dapat dilakukan oleh manusia seperti mengemudikan mobil. Maka dari itu akan dibuat program yang dapat mengemudikan mobil dengan menentukan nilai steering angle, throttle dan brake menggunakan data gambar kamera depan mobil, nilai steering angle, throttle, brake dan speed terakhir mobil. Aplikasi akan dibuat menggunakan Keras untuk pembuatan model, training dan testing. Berbagai teknik augmentasi gambar digunakan untuk menghasilkan gambar baru dengan jumlah tidak terbatas dari data gambar asli yang dikumpulkan. Program menggunakan simulator mobil untuk melakukan pengumpulan data dan uji coba program. Simulator dipilih sebagai media karena mudah dalam hal pengumpulan data dan tidak ada risiko terjadi kecelakaan seperti pada dunia nyata. Dengan adanya program ini, mesin dapat mengemudikan mobil secara otonom. Dari hasil eksperimen disimpulkan bahwa performa terbaik model tercapai pada model yang menggunakan dataset Lake dan Mountain, menggunakan augmentasi data, menggunakan channel warna YUV, menggunakan gambar dan state dari mobil sebagai input, learning rate 0.0001, dan drop out 0.5.**

***Kata Kunci*—convolutional neural network, deep learning, self-driving car.**

I. PENDAHULUAN

Perkembangan dan penelitian di bidang informasi dan komputer menghasilkan banyak penemuan di berbagai bidang. Salah satu bidang yang mendapat efek besar adalah industri otomotif dan pengembangan kendaraan yang otonom sepenuhnya. Dengan adanya teknologi self-driving car dan sistem robot otomatis kompleks yang lain semakin banyak dijumpai pada kehidupan sehari-hari [1]. Banyak penelitian yang menyatakan bahwa teknologi tersebut dapat memudahkan kehidupan manusia sehari-hari. Sistem robot otomatis cepat digunakan pada kondisi yang berbahaya bagi manusia [2].

[1] Andre Nyoto Raharjo, Departemen Teknik Informatika, Institut Sains dan Teknologi Terpadu Surabaya, Jl. Ngagel Jaya Tengah 73-77, Surabaya, Jawa Timur, Indonesia 60284 (e-mail: andrenyoto22@gmail.com)

Andreas, Department of Management, Faculty of Economy and Business, Universitas Pelita Harapan, Surabaya, Jawa Timur, Indonesia (e-mail: andreas.jodhinata@gmail.com)

Data dalam jumlah besar dan komputasi yang besar dibutuhkan untuk menyelesaikan masalah pergerakan otomatis pada lingkungan. Solusi modern yang ada sekarang ini adalah dengan menggunakan sensor, perangkat keras dan program [3].

Salah satu solusi untuk menyelesaikan masalah tersebut adalah dengan menggunakan teknik algoritma Artificial Intelligence yang mampu mengontrol self-driving car dengan tidak lebih buruk dibandingkan dengan manusia berdasar data dalam jumlah besar yang disediakan[4].

Pengumpulan data dari jalan yang ada di dunia nyata tidaklah mudah dan membutuhkan waktu dan biaya yang banyak. Oleh sebab itu penggunaan media simulator adalah solusi terbaik untuk mengatasi masalah pengumpulan data. Dengan adanya simulator juga untuk menguji kemampuan self-driving car tidak perlu memikirkan risiko kecelakaan seperti pada dunia nyata

Makalah ini bertujuan untuk melakukan pengumpulan artificial data dan mensimulasikan proses mengendarai mobil menggunakan simulator. Diperlukan pembuatan arsitektur network yang mampu memprediksi nilai steering angle, throttle dan brake dari data gambar dan kondisi terakhir mobil yang terdiri dari data steering angle, throttle, brake dan speed terakhir dari mobil.

II. CONVOLUTIONAL NEURAL NETWORK

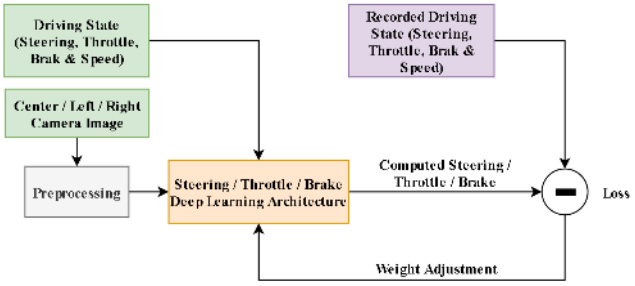
Convolutional Neural Network (CNN) sangat berguna dalam hal pemrosesan gambar dan jenis neural network ini dapat diaplikasikan pada berbagai bidang. Contoh penggunaan CNN adalah computer vision, pengenalan wajah, pelabelan scene untuk pengenalan Tindakan, dan lain sebagainya. CNN merupakan arsitektur neural network yang khusus dibuat dengan tujuan untuk pengenalan gambar yang efektif [5]. CNN terdiri dari beberapa layer yang berbeda, seperti convolutional layer, subsampling layer dan fully connected layer [6].

Convolutional Neural Network berhasil digunakan pada masalah pengenalan pola pada video, dimana sistem tersebut sangat penting pada self-driving car [7]. Pada makalah ini penggunaan CNN adalah untuk melakukan ekstraksi fitur pada gambar kamera depan dari mobil [8]. Ekstraksi fitur tersebut ditujukan untuk sistem dapat mengenali bagian gambar yang merupakan jalan dan yang bukan jalan. Hasil dari ekstraksi tersebut kemudian akan digabungkan dengan data kondisi mobil untuk digunakan sebagai pertimbangan sistem untuk menentukan kontrol mobil.

III. METODOLOGI

Penelitian ini menggunakan bahasa pemrograman Python

yang dijalankan pada Anconda Virual Environtment. Proses pelatihan dan testing akan menggunakan library Keras [9] yang berjalan pada Tensorflow [10] backend.



Gambar. 1. Diagram proses training

Metode training yang digunakan dapat dilihat pada Gambar. 1. Detail proses dari metode tersebut adalah sebagai berikut:

1. Data gambar melalui tahap preprocessing
2. Data gambar hasil preprocessing dan data driving state menjadi input network
3. Network akan memprediksi nilai steering angle / throttle / brake
4. Hasil nilai prediksi model akan dibandingkan dengan nilai dari recorded driving state untuk mendapat nilai loss
5. Nilai loss yang didapat digunakan untuk mengubah weight pada network

Proses Training yang dilakukan memiliki jumlah iterasi maksimal 200, ukuran batch size 40, ukuran sampel setiap iterasi untuk data training adalah 400 dan sampel untuk data validasi training adalah 40. Training akan berakhir ketika mencapai iterasi maksimal atau selama 3 iterasi berurutan tidak terjadi perbaikan nilai validation loss.

Layer (type)	Output Shape	Param #	Connected to
input_7 (InputLayer)	(None, 66, 200, 3)	0	
lambda_4 (Lambda)	(None, 66, 200, 3)	0	input_7[0][0]
convolution0 (Conv2D)	(None, 33, 100, 24)	1824	lambda_4[0][0]
convolution1 (Conv2D)	(None, 15, 48, 36)	21636	convolution0[0][0]
convolution2 (Conv2D)	(None, 6, 22, 48)	43248	convolution1[0][0]
convolution3 (Conv2D)	(None, 4, 20, 64)	27712	convolution2[0][0]
convolution4 (Conv2D)	(None, 2, 18, 64)	36928	convolution3[0][0]
dropout_10 (Dropout)	(None, 2, 18, 64)	0	convolution4[0][0]
flatten_4 (Flatten)	(None, 2304)	0	dropout_10[0][0]
input_8 (InputLayer)	(None, 4)	0	
concatenate_4 (Concatenate)	(None, 2308)	0	flatten_4[0][0] input_8[0][0]
dense1 (Dense)	(None, 100)	230900	concatenate_4[0][0]
dropout_11 (Dropout)	(None, 100)	0	dense1[0][0]
dense2 (Dense)	(None, 50)	5050	dropout_11[0][0]
dropout_12 (Dropout)	(None, 50)	0	dense2[0][0]
dense3 (Dense)	(None, 10)	510	dropout_12[0][0]
output (Dense)	(None, 1)	11	dense3[0][0]

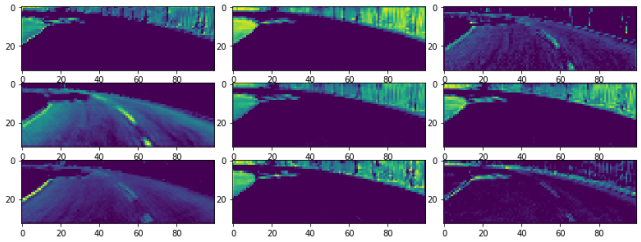
Total params: 367,819
Trainable params: 367,819
Non-trainable params: 0

Gambar. 2. Arsitektur Network

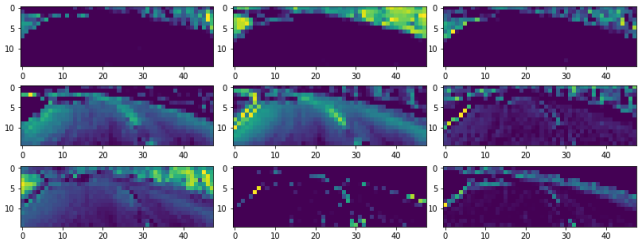
A. *Arsitektur Deep Learning*

Arsitektur deep learning terbaik yang digunakan memiliki 17 layer. Detail susunan setiap layer dapat dilihat pada gambar summary model pada gambar. 2.

Pada gambar 2, Arsitektur network akan memiliki 2 input layer, 1 lambda layer, 5 convolutional layer, 1 flatten layer, 1 concatenate layer, 3 dropout layer, 3 fully connected layer dan 1 output layer. Convolutional0 sampai convolutional 4 dan dense1 sampai dense3 akan menggunakan ReLU sebagai activation function. Arsitektur tersebut akan menggunakan Adam sebagai optimizer. Input layer1 berupa gambar berukuran 66x200x3, 66 adalah tinggi gambar, 200 lebar gambar dan 3 adalah color channel. Contoh output layer Convolutional0 dan Convolutional1 dapat dilihat pada gambar 3 dan gambar 4



Gambar. 3. Output Convolutional0



Gambar. 4. Output Convolutional1

B. *Dataset*

Dataset yang digunakan ada 2 macam, yaitu data gambar dan data numerik. Dataset didapatkan menggunakan simulator pada lintasan yang digunakan. Dataset yang gambar yang didapatkan berupa gambar kamera depan bagian kiri, tengah dan kanan. Kemudian untuk data numerik yang digunakan adalah data steering angle, throttle brake, dan speed. Dataset yang digunakan akan berasal dari 3 buah lintasan yang berbeda, yaitu lintasan lake, mountain dan castle. Contoh dataset lintasan lake dapat dilihat pada gambar .5 .

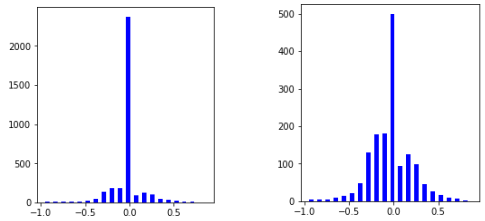
Gambar. 5. Diagram proses training

TABEL I
PERBANDINGAN DATASET

No	Data set	Ukuran Image	Total
1	Lake	320x160	3392
2	Mountain	320x160	6493
3	Castle	320x160	6479

Pada tabel I dapat dilihat perbandingan jumlah dataset

yang digunakan. Dataset yang digunakan pada saat training akan melalui tahap perataan persebaran data. Hal ini dilakukan untuk mengurangi data dengan frekuensi yang sangat besar dibanding data yang lain. Setelah proses tersebut data akan dibagi menjadi 2, yaitu data training dan validation dengan perbandingan 80% data testing dan 20% data validation. Contoh perbandingan data asli dan hasil perataan dapat dilihat pada gambar.6 .



Gambar. 6. Distribusi data asli(kiri) dan hasil perataan(kanan)

C. *Preprocessing*

Preprocessing adalah tahap yang harus dilakukan sebelum gambar digunakan sebagai input. Ada tiga tahap preprocessing yang digunakan. Pertama adalah crop untuk membuang bagian depan mobil dan langit pada gambar, pemotongan dilakukan sebesar 25px dari bawah dan 60px dari atas. Kedua adalah resize untuk mengubah ukuran gambar menjadi 200px lebar dan 66px tinggi. Proses terakhir adalah pengubahan color channel.

D. *Augmentation*

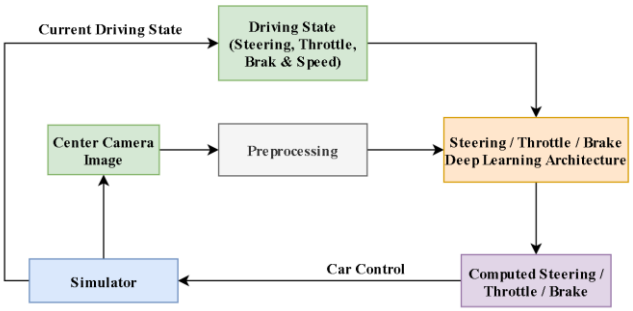
Augmentasi adalah teknik untuk membuat gambar baru dari gambar yang sudah ada. Teknik augmentasi yang dilakukan ada 2 macam, yaitu augmentasi dengan pengubahan data numerik dan tanpa pengubahan data numerik. Augmentasi dengan pengubahan data numerik adalah random choose image, flip, dan translate.

Untuk random choose gambar akan dipilih random dari gambar kiri, tengah dan kanan dan dilakukan pengubahan nilai steering sebesar +0.15 jika gambar kiri yang terpilih dan -0.15 jika gambar kanan yang terpilih. Pada augmentasi flip akan dilakukan pembalikan nilai steering angle, jika nilai awal positif akan dijadikan negatif dan sebaliknya.

Pada augmentasi translate akan dilakukan penambahan nilai steering sebesar -0.0015 x jumlah pixel pergeseran ke kiri dan + 0.0015 x jumlah pixel pergeseran ke kanan. Augmentasi tanpa ada pengubahan data numerik meliputi shadow, brightness, fog, flare, rain, snow dan gravel.

E. *Ilustrasi Testing*

Proses testing akan menggunakan model steering, throttle dan brake yang sudah dilatih untuk mengemudikan mobil pada simulator.



Gambar. 7. Diagram proses testing

Metode testing yang digunakan pada makalah ini dapat dilihat pada Gambar. 7. Detail proses dari metode tersebut adalah sebagai berikut:

1. Program akan mengambil data gambar kamera depan, nilai steering angle, throttle, brake, dan speed terakhir mobil
2. Data gambar kamera depan melalui tahap preprocessing
3. Data gambar hasil preprocessing dan data driving state menjadi input network
4. Network akan memprediksi nilai steering angle / throttle / brake
5. Hasil nilai prediksi akan dikirimkan ke simulator sebagai car kontrol
6. Proses berlangsung selamnya sampai diberhentikan secara manual

IV. EKSPERIMEN

Terdapat 6 macam eksperimen yang dilakukan dan dari 5 macam tersebut terdapat 15 skenario eksperimen. Eksperimen yang dilakukan meliputi perbedaan input yang digunakan, perbedaan dataset yang digunakan, perbedaan color channel yang digunakan, perbedaan augmentasi yang digunakan, perbedaan parameter learning rate, dan perbedaan parameter dropout. Detail skenario eksperimen yang dilakukan dapat dilihat pada tabel II.

Uji coba dilakukan pada lintasan lake dan mountain. Setiap skenario uji coba akan diamati performa program untuk mengendarai mobil. Performa diukur dengan kemampuan program mengenali jalan untuk menentukan kendali yang diperlukan. Jumlah kejadian mobil keluar jalur atau menabrak dan jumlah kendali manual yang digunakan juga akan diamati. Semakin sedikit kontrol manual yang diperlukan dan semakin sedikit mobil berjalan keluar jalur atau menabrak dianggap memiliki performa yang baik.

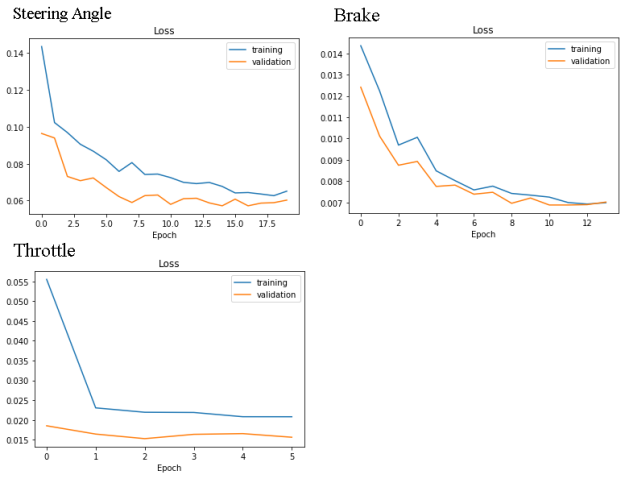
V. HASIL EKSPERIMEN

Subbab ini akan membahas hasil eksperimen dari setiap skenario yang ada pada tabel. Dari semua skenario didapatkan bahwa skenario no 2 memiliki performa terbaik dari semua skenario.

TABEL II
SKENARIO EKSPERIMEN

No	Data set	Aug-ment-ation	Color Channel	Input	Learn-ing Rate	Drop Out
1	LK+MT	Some	YUV	***Im***	0.0001	0.5
2	***LK+MT***	***Some***	***YUV***	***Im+St***	***0.0001***	***0.5***
3	***LK***	Some	YUV	Im+St	0.0001	0.5
4	***MT***	Some	YUV	Im+St	0.0001	0.5
5	***CS***	Some	YUV	Im+St	0.0001	0.5
6	***LK+MT +CR***	Some	YUV	Im+St	0.0001	0.5
7	LK+MT	***None***	YUV	Im+St	0.0001	0.5
8	LK+MT	***Full***	YUV	Im+St	0.0001	0.5
9	LK+MT	Some	***RGB***	Im+St	0.0001	0.5
10	LK+MT	Some	***YCbCr***	Im+St	0.0001	0.5
11	LK+MT	Some	***CIELAB***	Im+St	0.0001	0.5
12	LK+MT	Some	YUV	Im+St	***0.001***	0.5
13	LK+MT	Some	YUV	Im+St	***0.0005***	0.5
14	LK+MT	Some	YUV	Im+St	0.0001	***0.1***
15	LK+MT	Some	YUV	Im+St	0.0001	***1***

LK adalah Lake, MT adalah mountain, CS adalah Castle, CR adalah crash recovery, Im adalah Image, dan St adalah State(Steering, Throttle, Brake dan Speed).



Gambar. 8. Grafik loss dan val_loss training model terbaik

Pada gambar.8 . adalah gambar grafik hasil training untuk steering angle, throttle, dan brake. Pada pelatihan steering angle proses traing berakhir pada epoch 20 dan hasil terbaik pada epoch 17 dengan nilai validation loss 0.0571828. Throttle selesai pada epoch 6 dan hasil terbaik pada epoch 3 dengan nilai validation loss 0.00152912. Brake selesai pada epoch 14 dengan hasil terbaik pada epoch 11 dengan nilai validation loss 0.0068767.

Hasil dari ketiga model tersebut dapat mengemudikan mobil pada kedua lintasan tanpa ada bantuan kendali manual. Pada lintasan lake mobil dapat menyelesaikan 1 lap dengan waktu 1 menit 19 detik dan pada lintasan mountain dengan waktu 2 menit 24 detik.

Eksperimen perbedaan dataset program tidak dapat mengemudikan mobil dengan baik pada lintasan yang belum diketahui sebelumnya. Penggunaan dataset castle memiliki performa terburuk dari dataset yang lain. Penambahan teknik crash recovery tidak memberikan kenaikan performa dibandingkan tanpa teknik crash recovery.

Eksperimen color channel, YCbCr memiliki performa yang sedikit di bawah color channel lain pada lintasan lake. RGB, CIELAB dan YUV memiliki performa yang cukup serupa pada lintasan lake. Pada lintasan mountain penggunaan color channel YUV memiliki performa terbaik dibanding yang lain.

Eksperimen perbedaan augmentasi, penggunaan augmentasi data dapat meningkatkan performa program untuk mengendarai mobil. Akan tetapi penggunaan augmentasi rain, snow, fog, flare dan gravel tidak memberikan peningkatan performa. Augmentasi tersebut tidak membantu banyak karena tidak sesuai dengan lintasan dan data gambar yang dipakai.

Eksperimen perbedaan nilai learning rate, semakin tinggi learning rate maka semakin cepat proses training dengan risiko penurunan performa. Eksperimen nilai perbedaan drop rate, nilai drop rate yang terlalu tinggi dan terlalu rendah dapat menurunkan performa. Hasil eksperimen pada setiap skenario dapat dilihat pada tabel III. Cuplikan hasil eksperimen dapat dilihat pada gambar 9 dan 10.

TABEL III
PERFORMA HASIL EKSPERIMEN

No	Lake Track		Mountain Track	
	No Crash	No Manual	No Crash	No Manual
1	✗	✗	✗	✗
2	✓	✓	✓	✓
3	✗	✗	✗	✗
4	✗	✗	✗	✗
5	✗	✗	✗	✗
6	✗	✗	✗	✗
7	✗	✗	✗	✗
8	✗	✗	✗	✗
9	✓	✓	✗	✗
10	✗	✗	✗	✗
11	✓	✓	✗	✗
12	✗	✗	✗	✗
13	✗	✗	✗	✗
14	✗	✗	✗	✗
15	✗	✗	✗	✗

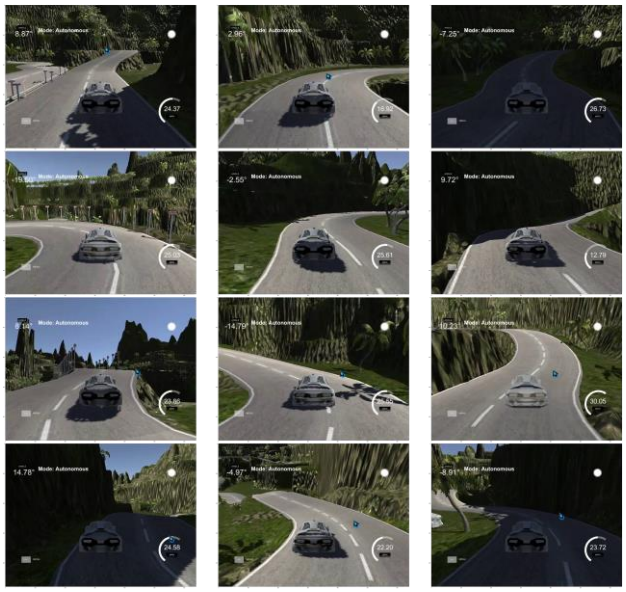
VI. PENUTUP

Bab ini akan berisi kesimpulan dan saran. Kesimpulan didapatkan berdasarkan hasil pengamatan proses uji coba yang telah dilakukan. Dari beberapa kesimpulan yang diambil, dapat dilihat keunggulan atau kekurangan dari program yang dibuat. Maka dari itu, bab ini akan memberikan saran untuk mengembangkan aplikasi serupa atau aplikasi lain dengan metode yang sama. Saran tersebut juga akan diberikan per poin.

A. Kesimpulan

Pada subbab ini akan di bahas mengani kesimpulan yang di dapat setelah pembuatan penelitian ini. Kesimpulan merupakan hasil yang didapatkan dari pengamatan performa program selama proses uji coba. Kesimpulan tersebut akan dijelaskan dalam beberapa poin sebagai berikut:

Gambar. 9. Gambar cuplikan self-driving dengan model scenario 2 pada lintasan lake



Gambar. 10. Gambar cuplikan self-driving dengan model scenario 2 pada lintasan mountain

1. Penggunaan teknik augmentasi gambar sangat membantu dalam hal mengatasi masalah terbatasnya jumlah training data. Augmentasi gambar mampu menghasilkan gambar-gambar baru dengan jumlah tak terbatas yang dapat memperkaya dataset yang digunakan untuk melatih model.
2. Teknik augmentasi flip, translate, shadow dan brightness dan penyesuaian nilai steering angle dapat meningkatkan performa program untuk mengendalikan mobil.
3. Teknik augmentasi snow, fog, rain, gravel, dan flare tidak memberikan perubahan performa yang signifikan pada lintasan danau dan penurunan performa pada lintasan gunung.
4. Penggunaan color channel YUV memiliki performa yang paling baik pada kedua lintasan yang digunakan. RGB, YCbCr dan CIELAB memiliki performa yang kurang lebih sama baik pada lintasan danau, dan pada lintasan gunung YCbCr dan CIELAB memiliki performa yang mirip dan RGB memiliki performa yang paling rendah.
5. Penggunaan input gambar dan input kondisi mobil meningkatkan kemampuan program untuk menentukan nilai steering angle, throttle, dan brake dibandingkan dengan hanya menggunakan input gambar saja.
6. Program belum berjalan baik pada lintasan yang baru/berbeda dimana data tersebut tidak terdapat pada saat training dan masih memerlukan pengembangan lebih lanjut
7. Penggunaan nilai drop out terlalu tinggi atau terlalu rendah dapat menurunkan performa model yang dilatih
8. Nilai learning rate semakin tinggi akan mempercepat proses training, tetapi pada saat yang bersamaan juga menurunkan performa model yang dilatih

B. Saran

Selain menarik kesimpulan dari pembuatan program ini, berdasarkan proses pembuatan dan uji coba terdapat saran yang mungkin berguna bagi pembaca dalam pembuatan aplikasi serupa. Beberapa saran yang ada antara lain :

1. Menggunakan gabungan arsitektur Convolutional Neural Network dengan arsitektur lain seperti Long-Short Term Neural Network dan Recurent Neural Network.
2. Menggunakan gambar dengan resolusi dan ukuran yang lebih besar.
3. Untuk pengembangan selanjutnya dapat menambahkan jenis-jenis lintasan lain sebagai data set, supaya program dapat mengenali berbagai macam lintasan.
4. Menggunakan ukuran batch yang lebih besar supaya program dapat mengenali fitur-fitur penting dengan lebih baik.

DAFTAR PUSTAKA

[1] M. V Smolyakov, A. I. Frolov, V. N. Volkov, and I. V Stelmashchuk, “Self-driving car steering angle prediction based on deep neural network an example of CarND udacity simulator,” in *2018 IEEE 12th international conference on application of information and communication technologies (AICT)*, 2018, pp. 1–5.

[2] J. Kocić, N. Jovičić, and V. Drndarević, “An end-to-end deep neural network for autonomous driving designed for embedded automotive platforms,” *Sensors*, vol. 19, no. 9, p. 2064, 2019.

[3] M. Bojarski *et al.*, “End to end learning for self-driving cars,” *arXiv Prepr. arXiv1604.07316*, 2016.

[4] Y. Tian, K. Pei, S. Jana, and B. Ray, “Deeptest: Automated testing of deep-neural-network-driven autonomous cars,” in *Proceedings of the 40th international conference on software engineering*, 2018, pp. 303–314.

[5] T.-D. Do, M.-T. Duong, Q.-V. Dang, and M.-H. Le, “Real-time self-driving car navigation using deep neural network,” in *2018 4th International Conference on Green Technology and Sustainable Development (GTSD)*, 2018, pp. 7–12.

[6] A. Navarro, J. Joerdening, R. Khalil, A. Brown, and Z. Asher, “Development of an autonomous vehicle control strategy using a single camera and deep neural networks,” 2018.

[7] M. A. Nielsen, *Neural networks and deep learning*, vol. 25. Determination press San Francisco, CA, 2015.

[8] M. Spryn, “Autonomous Driving using End-to-End Deep Learning: an AirSim tutorial,” 2017. https://github.com/Microsoft/AutonomousDrivingCookbook/tree/master/AirSimE2EDeepLearning.

[9] F. Chollet, “Keras: The Python Deep Learning library,” *Keras.Io*, 2015.

[10] M. Abadi *et al.*, “TensorFlow: A system for large-scale machine learning,” 2016.