Jurnal Asiimetrik: Jurnal Ilmiah Rekayasa Dan Inovasi
Terakreditasi **"Peringkat 4"** oleh Kemenristek/BRIN, Nomor SK: 200/M/KPT/2020
Volume 5 Nomor 1 Tahun 2023, Januari 2023, Halaman 19-26
Website : http://journal.univpancasila.ac.id/index.php/asiimetrik/

p-ISSN 2655-1861
e-ISSN 2716-2923

Implementasi Algoritma *Convolutional Neural Network* Pada Kendaraan Tanpa Awak Skala Kecil

Implementation of Convolutional Neural Network Algorithm on Small-Scale Unmanned Vehicles

Muhammad Zacky Asy'ari*, Anthony Williams Gouw dan Desliong Arjuna Limanjaya
Automotive and Robotics Engineering Program, Computer Engineering Department, Bina Nusantara University, Banten, Indonesia

Informasi artikel:

Diterima:
21/10/2022
Direvisi:
11/11/2022
Disetujui:
12/11/2022

Abstract

Autonomous Vehicle is a vehicle capable of navigating the car independently without requiring input from the driver. This research aims to design and manufacture a prototype of an unmanned vehicle that can maneuver across a simple artificial road. This study also aims to analyze the performance of the NVIDIA Jetson Nano in processing deep learning models and driving actuators according to the predictions given by the model. The research stages include designing a prototype, creating an artificial path, taking image data, conducting training, and then implementing the training model on the car prototype. After testing the prototype, the training model made the correct steering angle prediction using epoch 50 with RMSE train and validation, 0.1792 and 0.1896, respectively. NVIDIA Jetson Nano also performs well in computing steering angle predictions with live input from the camera.

***Keywords**: autonomous vehicles, convolutional neural network, computer vision.*

***SDGs*:**

Abstrak

Autonomous Vehicle adalah sebuah kendaraan yang mampu menavigasikan mobil secara mandiri tanpa memerlukan *input* dari pengendara. Tujuan penelitian ini adalah merancang dan membuat sebuah prototipe kendaraan tanpa awak yang dapat bermanuver melintasi jalan buatan sederhana. Penelitian ini juga bertujuan untuk menganalisa performa dari NVIDIA Jetson Nano dalam memproses model *deep learning* dan menggerakkan aktuator sesuai dengan prediksi yang diberikan model. Tahapan penelitian termasuk merancang prototipe, membuat jalur buatan, mengambil data gambar, melakukan pelatihan menggunakan *Convolutional Neural Network*, lalu mengimplementasikan model pelatihan pada purwarupa mobil. Setelah dicoba pada purwarupa, model hasil pelatihan berhasil membuat prediksi sudut steering terbaik menggunakan epoch 50 dengan RMSE pelatihan dan validasi, 0,1792 dan 0,1896 berturut-turut. NVIDIA Jetson Nano juga memiliki performa yang baik dalam melakukan komputasi prediksi sudut steering dengan *live input* dari kamera.

Kata Kunci: kendaraan tanpa awak, *convolutional neural network, computer vision.*

*Penulis Korespondensi. Tel: +6221 5369 6969; Hp: +62 821 1343 4649
email : muhammad.zacky@binus.ac.id

1. PENDAHULUAN

Seiring berkembangnya peradaban, transportasi telah menjadi salah satu pilar yang sangat penting dalam menjaga keberlangsungan dari kegiatan masyarakat (Sutandi, 2015). Akan tetapi, hal tersebut terkadang dapat membawa musibah kepada para penggunanya. Dilansir dari Komite Nasional Keselamatan Transportasi (KNKT),selama kurun waktu 2007-2022 paling sering terjadi di Indonesia sebesar 70% adalah bertabrakan dan disebabkan oleh faktor manusia berupa *human error* (Polri, 2022). *Human error* yang dimaksud berupa kelalaian dalam berkendara, tidak fokus, dan faktor kelelahan.

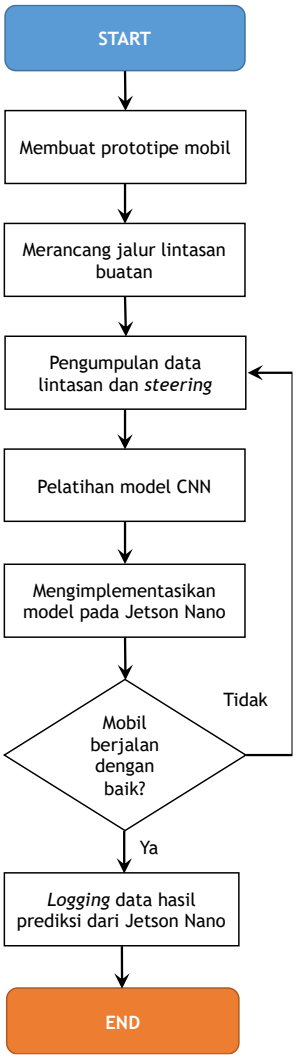
Demi mengurangi tingkat kecelakaan yang disebabkan oleh kesalahan manusia, pengembangan teknologi *Autonomous Vehicle* (AV) mulai dilakukan. Teknologi pada AV memiliki fitur-fitur seperti mampu menavigasikan mobil secara sendiri tanpa memerlukan input dari pengendara. AV juga dapat mempertahankan pola berkendara yang aman dan mampu melakukan berbagai jenis manuver, semuanya hanya dengan menggunakan data lingkungan yang diperolehnya dari sensor-sensor (Hussain dan Zeadally, 2018).

Untuk membuat *Autonomous Vehicle*, tentu saja membutuhkan deep learning, salah satunya termasuk *Convolutional Neural Network* (CNN). CNN adalah salah satu arsitektur deep learning ternama yang terinspirasi mekanisme persepsi visual dari makhluk hidup (Gu dkk., 2018). CNN mahir dalam mendeteksi pola pada gambar atau video dan memahaminya, yang membuatnya sangat cocok digunakan untuk melakukan analisis gambar (Jogin dkk., 2018).

Atas dasar permasalahan di atas, penelitian ini ditujukan untuk merancang, membuat, serta mengembangkan kendaraan tanpa awak berskala kecil untuk dapat bermanuver dalam jalur buatan menyerupai trek lomba adu kecepatan kendaraan dengan metode deep learning yaitu CNN menggunakan NVIDIA Jetson Nano sebagai basis komputernya (Cass, 2020; Vijitkunsawat dan Chantngarm, 2020). Analisa performa model ini di ukur dengan Root Mean Square Error (RMSE) pada proses pelatihan dan validasi (Eraqi, Moustafa dan Honer, 2017).

2. METODOLOGI

Pelaksanaan penelitian ini dilakukan dengan metode penelitian kualitatif, yang dilaksanakan dalam beberapa tahapan seperti yang akan ditunjukkan pada Gambar 1.



Gambar 1. Diagram alir tahap penelitian

Pertama, dirancang dan dibuat prototipe mobil sebagai kerangka dari kendaraan tanpa awak (lihat Gambar 2), lalu dibuat jalur buatan yang melingkar dengan sudut tumpul sebesar kira-kira 135° pada setiap tikungannya (lihat Gambar 3). Untuk memulai proses pengumpulan data, kendaraan diletakkan pada jalur, lalu sudut pandang kamera disesuaikan sehingga kamera hanya menangkap gambar garis hitam dari jalur seperti pada Gambar 4.

(a). Desain (b). Purwarupa

Gambar 2. Model mobil

Gambar 3. Jalur buatan

Gambar 4. Sudut pandang kamera

Proses pengambilan data dilakukan dengan mengontrol *steering* dari mobil secara manual menggunakan *joystick*. Mobil akan dikendalikan memutari jalur buatan, lalu gambar dari kamera serta respon *steering* dari *joystick* selama pengendalian tersebut akan direkam dalam bentuk data. Gambar yang telah direkam akan disimpan dalam satu berkas, dan setiap berkasnya akan didampingi oleh file *comma-separated values* (*CSV)* yang berisikan jalur *file* gambar pada komputer beserta dengan nilai *steering* yang sesuai. Data-data ini yang nantinya akan dimasukkan sebagai input dari *deep learning*.

Setelah pengumpulan data sudah lengkap, tahap selanjutnya yaitu pelatihan model. Model dilatih menggunakan *library* TensorFlow-Keras dengan *optimizer Adam* (Bock and Weiß, 2019) dan *learning rate* 0,0001. Dilakukan beberapa pelatihan model dengan mengubah parameter *training* yaitu *epoch*, lalu dibandingkan performa dari model (Brownlee, 2022). Model dengan performa terbaik akan diambil dan diimplementasikan pada *prototype*, lalu dilakukan data *logging* dan analisa performa model.

Langkah pertama saat memasuki proses *training* adalah melakukan augmentasi gambar. Proses ini penting untuk menambah variasi data yang akan dilatih. Dengan memberi mesin input gambar yang telah diaugmentasi, maka mesin akan tetap dapat mengenali gambar tersebut walaupun gambarnya buram, terlalu kecil, bahkan terbalik sekalipun.

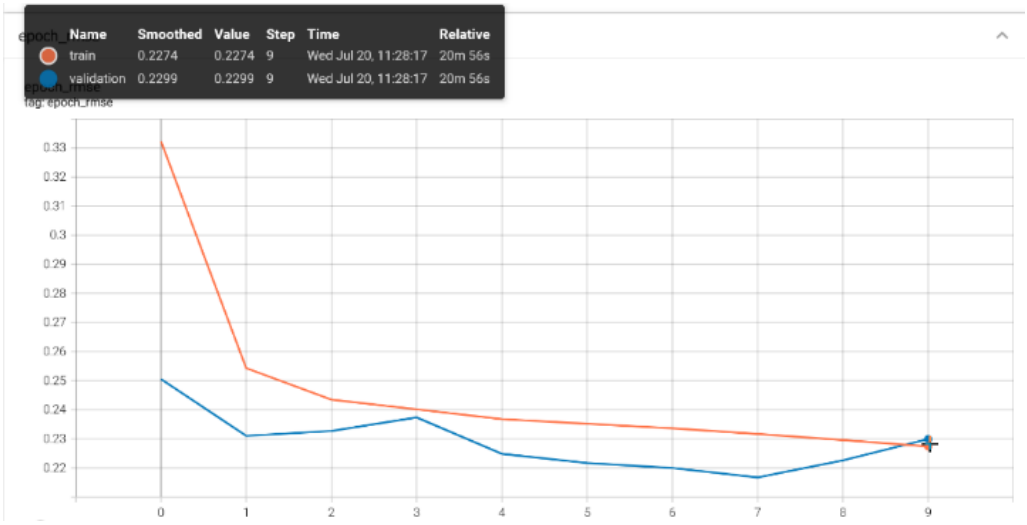
Data gambar yang diperoleh pada tahap koleksi data juga perlu dipisah menjadi data pelatihan dan validasi. Digunakan ukuran pengetesan sebesar 0,4 dari *dataset*, yang berarti 60% gambar digunakan sebagai data pelatihan dan sisanya akan digunakan sebagai data validasi . Konfigurasi pelatihan juga dapat diatur pada fungsi *"model.fit()"*, di mana jumlah *epoch*, langkah per epoch, Langkah validasi maupun ukuran *batch* dapat diubah sesuai dengan kebutuhan pelatihan.

Pelatihan ini akan menghasilkan *file* model *".h5"* yang nantinya akan diimplementasikan pada NVIDIA Jetson Nano. Model akan dimuat, lalu akan dijalankan untuk membuat prediksi *steering* berdasarkan gambar yang ditangkap oleh kamera pada NVIDIA Jetson Nano.

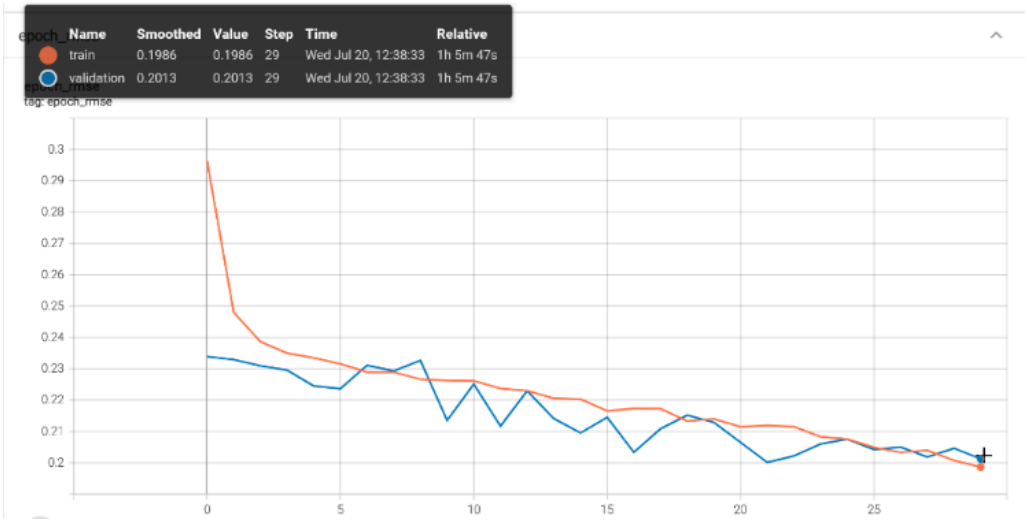
Kendaraan akan dibiarkan untuk terus berjalan secara mandiri dengan mengandalkan prediksi *steering* dari model hasil pelatihan. Jika kendaraan sudah dapat mendeteksi jalur dengan baik dan berhasil berjalan memutari jalur buatan beberapa putaran tanpa keluar garis, maka dapat dinyatakan bahwa performa model pelatihan sudah baik dan NVIDA Jetson Nano sudah berhasil melakukan komputasi prediksi *steering* secara *real-time*. Lalu, langkah terakhir adalah melakukan *logging* data hasil prediksi dari Jetson Nano untuk digunakan sebagai data analisis.

3. HASIL DAN PEMBAHASAN

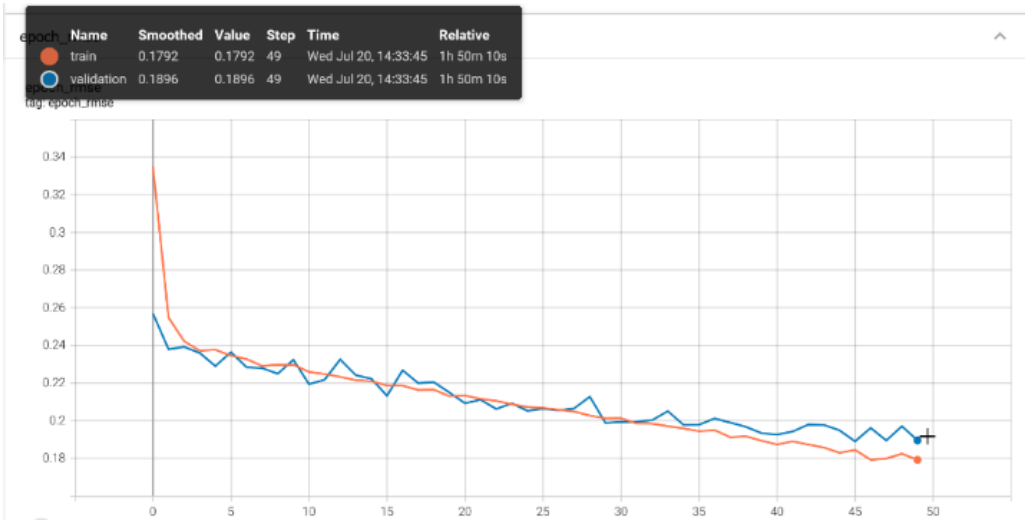
Dalam proses pelatihan model, beberapa parameter diubah untuk meneliti pengaruh dan korelasi dari masing-masing parameter terhadap nilai akhir akurasi dari model. Parameter yang diubah adalah nilai *epoch*, yaitu jumlah total



Gambar 5. Grafik pelatihan *epoch* = 10



Gambar 6. Grafik pelatihan *epoch* = 30



Gambar 7. Grafik pelatihan *epoch* = 50

iterasi pelatihan melewati satu siklus dataset, grafik hasil pelatihan divisualisasikan menggunakan *Tensorboard*.

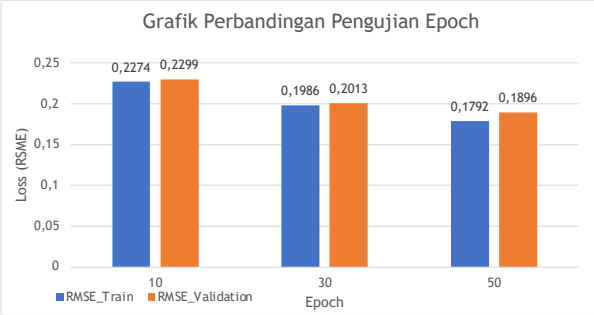
Untuk menguji pengaruh *epoch* terhadap tingkat akurasi model, digunakan tiga nilai *epoch* berbeda yaitu 10, 30, dan 50. Hasil pelatihan yang berupa nilai *loss* dapat diamati pada Gambar 5, Gambar 6, dan Gambar 7.

Penggunaan grafik *TensorBoard* untuk memvisualisasikan metrik pelatihan, seperti *loss* dan akurasi pelatihan dan validasi, dapat membantu dalam mengenali *overfitting*, *underfitting*, atau tepat. *Overfitting* adalah titik dalam pelatihan ketika kita beralih dari proses belajar ke menghafal. Pada titik ini, model tidak akan lagi menggeneralisasi dengan baik. Kondisi ini adalah titik di mana *training loss* menuju nol (meningkat), namum *validation loss* validasi mulai meningkat secara dramatis (menjadi lebih buruk) (Vogelsang dan Erickson, 2020).

Oleh karena itu mengatur penghentian awal dalam menentukan kondisi di mana pelatihan akan berhenti sebelum mengeksekusi jumlah epoch yang ditentukan adalah kritikal. Misalnya, jika tidak terlihat peningkatan dalam *loss* set validasi untuk beberapa iterasi, mungkin pelatihan telah maksimal, dan lebih banyak *epoch* akan menghasilkan *overfitting*. Karakteristik tanda *overfitting* terlihat ialah ketika kondisi validasi *loss* mulai naik, walaupun saat *training loss* terus menurun.

Berdasarkan dari ketiga grafik tersebut, pada Gambar 5 tren *loss* pelatihan masih terlihat menurun, artinya masih ada kemungkinan nilai *loss* terus berkurang. Sedangkan membandingkan Gambar 6 dan Gambar 7 terlihat bahwa *epoch* yang tepat di sekitar angka 30. Namun, tren loss masih menunjukkan penurunan. Sehingga, masih perlu di analisa dengan penambahan jumlah *epoch*.

Dari hasil pengujian nilai parameter *epoch*, dapat diamati bahwa semakin besar nilai *epoch*, semakin kecil nilai *loss* yang dimiliki oleh model, yang berarti akurasi dari model semakin meningkat (Rawat, Patel dan Manry, 2013). Nilai *RMSE loss* terendah dicapai saat *epoch* bernilai 50, yaitu sebesar 0,1792 untuk *training loss* dan 0,1896 untuk *validation loss* (lihat Gambar 8).



Gambar 8. Perbandingan pengujian *epoch*

Untuk membuat perbandingan antara ketiga grafik di atas, nilai *loss RMSE* akhir pada setiap *epoch* disajikan pada Tabel 1.

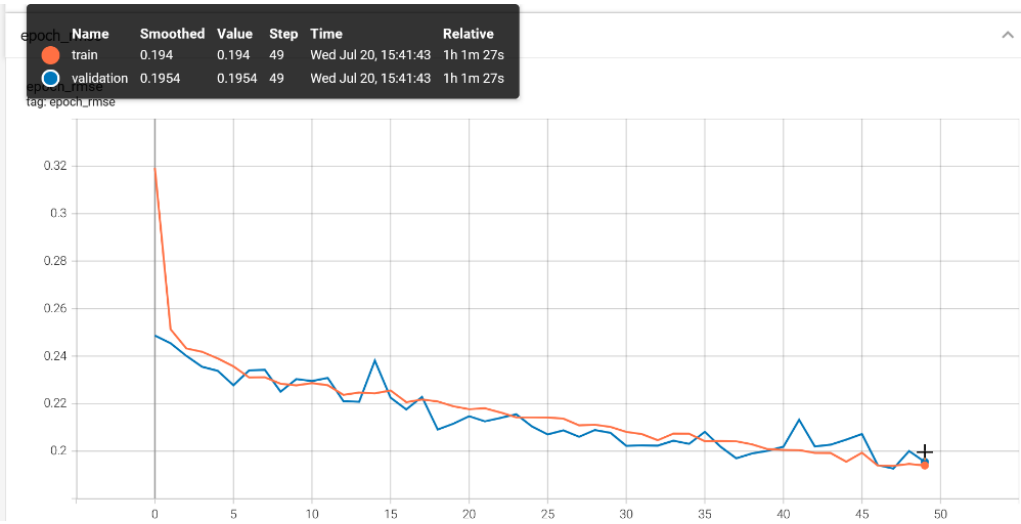
Tabel 1. Pengujian parameter epoch

Epoch	RMSE_Train	RMSE_Validation	Waktu
10	0,2274	0,2299	20 menit 56 dtk
30	0,1986	0,2013	1 jam 5 menit 47 dtk
50	0,1792	0,1896	1 jam 50 menit 10 dtk

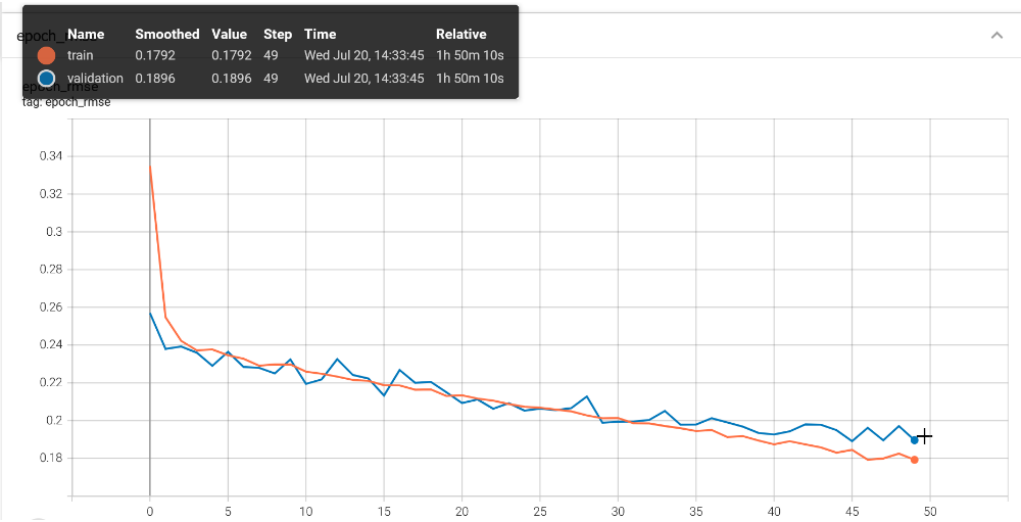
Jika mengamati tren grafik nilai *loss* dari model masih dapat diminimalkan dengan cara menambah nilai *epoch*. Walaupun demikian, efisiensi dari pelatihan pun perlu diperhatikan. Nilai *epoch* yang lebih besar membutuhkan waktu pelatihan yang lebih lama. Nilai *loss* pada *epoch* 50 sudah cukup baik dan sudah dapat berjalan sesuai dengan ekspektasi. Menambah nilai *epoch* hanya akan menurunkan efisiensi dari pelatihan, memakan waktu lebih lama, dan dapat menyebabkan risiko *overtraining* (Bilbao dan Bilbao, 2017). Cara terbaik untuk mengatur konfigurasi pelatihan adalah mencari nilai di mana nilai *loss* dari model sudah cukup rendah untuk berjalan dengan baik, dan pada saat yang bersamaan waktu yang diperlukan untuk pelatihan pun tidak terlalu lama, serta nilai *validation* tidak meningkat.

3.1. Pengujian Parameter Ukuran *Batch*

Untuk menguji pengaruh ukuran batch terhadap tingkat akurasi dari model (Radiuk, 2017), digunakan dua nilai ukuran *batch* yaitu 50 dan 100. Nilai epoch yang akan digunakan adalah 50, karena nilai epoch tersebut mempunyai nilai



Gambar 9. Pelatihan ukuran *batch* = 50



Gambar 10. Pelatihan ukuran *batch* = 100

loss terendah pada pengujian sebelumnya. Hasil pelatihan pengaruh ukuran *batch* dapat diamati pada Gambar 9 dan Gambar 10.

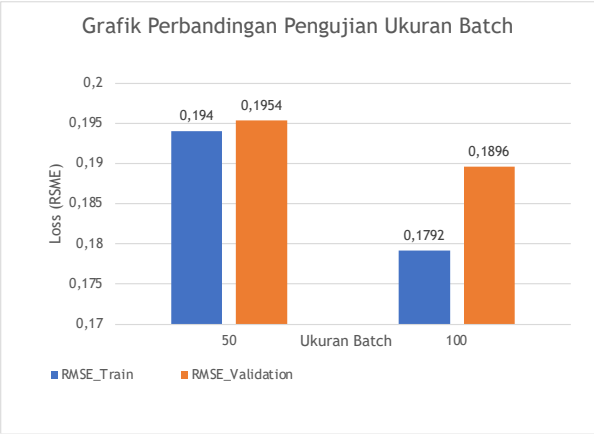
Pada pelatihan *batch* 50 terlihat bahwa tren penurunan pelatihan *loss* masih menurun dan nilai validasi masih berfluktuasi (lihat Gambar 9), sehingga perlu di ketahui lebih lanjut dengan jumlah batch yang lebih tinggi.

Pada Gambar 10, terlihat bahwa meskipun *loss* pelatihan masih terus menurun, namun *loss* validasi mulai merenggang yang merupakan indikasi awal pelatihan *overfitting*, sehingga tidak ada manfaat untuk melakukan pelatihan dengan jumlah *batch* yang lebih besar.

Untuk membuat perbandingan antara kedua grafik tersebut, nilai *loss* RMSE akhir pada setiap nilai ukuran batch disajikan pada Gambar 11. Dari hasil pegujian nilai ukuran *batch*, perubahan signifikan yang dapat diamati adalah waktu yang diperlukan untuk menjalankan pelatihan. Menurunkan nilai ukuran *batch* dari 100 ke 50 menghasilkan waktu pelatihan turun sebesar hampir setengah dari waktu awal (lihat Tabel 2). Hal ini menandakan bahwa nilai dari ukuran batch akan meningkatkan total waktu yang diperlukan untuk menyelesaikan pelatihan (Wang dkk., 2022).

Tabel 2. Hasil pengujian ukuran *batch*

Batch Size	RMSE_Train	RMSE_Validation	Waktu
50	0,194	0,1954	1 jam 1 menit 27 dtk
100	0,1792	0,1896	1 jam 50 menit 10 dtk



Gambar 11. Perbandingan pengujian ukuran *batch*

3.2. Analisa Pengaruh Parameter Pelatihan

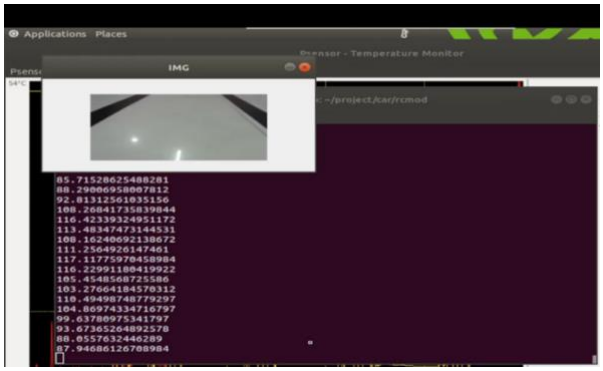
Berdasarkan kedua hasil percobaan yangtelah dilakukan, nilai *epoch* dan ukuran *batch* sangat berpengaruh terhadap hasil akhir pelatihan. Nilai keduanya dapat dikatakan berbanding lurus dengan akurasi dari model, sekaligus dengan waktu yang dibutuhkan untuk menyelesaikan proses pelatihan.

Pengamatan tersebut mengindikasikan bahwa tingkat akurasi dari sebuah model dapat ditingkatkan semakin banyak dilakukan pelatihan, akan tetapi adapun biaya peluang berupa waktu yang dibutuhkan untuk pelatihan. Cara terbaik untuk menyesuikan konfigurasi pelatihan adalah mencari nilai di mana nilai *loss* dari model sudah cukup rendah untuk berjalan dengan baik (Wang dkk., 2022), dan saat yang bersamaan waktu yang diperlukan untuk pelatihan pun tidak terlalu lama.

3.3. Analisa Performa Model Pelatihan

Model hasil pelatihan dijalankan pada NVIDIA Jetson Nano untuk dinilai performanya dalam mendeteksi tikungan dan mengatur derajat *steering* untuk menyesuaikan dengan tikungan.

Pertama-tama, perlu diketahui bahwa rentang sudut dari servo penggerak roda mobil adalah 0° – 180°, di mana belokan kanan berada pada rentang $0° \leq \theta < 90°$ dan belokan kiri berada pada rentang $90 < \theta \leq 180°$. Sehingga, pada sudut 90° mobil akan berjalan lurus. Respon komputasi dari Jerson Nano dapat dilihat pada Gambar 12.



Gambar 12. Simulasi model *steering*

Berdasarkan hasil percobaan, dapat disimpulkan bahwa Jetson Nano berhasil membuat prediksi dengan baik sesuai dengan kondisi jalur. Hal ini mengindikasikan bahwa model hasil pelatihan sudah baik, dan Jetson Nano juga mampu melakukan komputasi prediksi dengan cepat, sehingga dapat menyesuaikan prediksi dengan gambar *live* yang tertangkap oleh kamera.

4. SIMPULAN

Berdasarkan hasil analisis dari penelitian ini, dapat disimpulkan bahwa hasil purwarupa kendaraan tanpa awak sudah berhasil bermanuver secara mandiri melintas jalur buatan. Model hasil pelatihan CNN juga sudah berhasil membuat prediksi sudut *steering* yang tepat untuk setiap tikungan pada jalur buatan. Dengan parameter *epoch* 50, telah dibuktikan bahwa NVIDIA Jetson Nano memiliki performa yang baik (Prasetyo dkk., 2020) dalam melakukan komputasi prediksi sudut *steering* dengan *live input* dari kamera karena berhasil memutari *trek* buatan. Dari segi pelatihan, semakin lama durasi pelatihan pun akan meningkatkan akurasi dari model CNN dalam memprediksi sudut *steering*. Namun, perlu diperhatikan kondisi *overfitting* dalam proses pelatihan.

Mencoba melatih model menggunakan metode *deep learning* lainnya (Shrestha dan Mahmood, 2019) untuk dibandingkan performanya dengan model CNN. Selain itu, mendesain jalur buatan yang lebih kompleks untuk menguji performa dari model seolah-olah berada di jalan raya.

DAFTAR PUSTAKA

Bilbao, I. dan Bilbao, J. (2017) 'Overfitting problem and the over-training in the era of data: Particularly for Artificial Neural Networks', in *2017 eighth international conference on intelligent computing and information systems (ICICIS)*. IEEE, hal. 173–177.

Bock, S. dan Weiß, M. (2019) 'A proof of local convergence for the Adam optimizer', in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, hal. 1–8.

Brownlee, J. (2022) *Difference Between a Batch and an Epoch in a Neural Network*, *MachineLearningMastery.com*. Available at: https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/ [Online] (Diakses: 8 September 2022).

Cass, S. (2020) 'Nvidia makes it easy to embed AI: The Jetson nano packs a lot of machine-learning power into DIY projects-[Hands on]', *IEEE Spectrum*, 57(7), hal. 14–16.

Eraqi, H.M., Moustafa, M.N. dan Honer, J. (2017) 'End-to-end deep learning for steering autonomous vehicles considering temporal dependencies', in *31st Conference on Neural Information Processing Systems (NIPS 2017). 31st Conference on Neural Information Processing Systems (NIPS 2017)*, Long Beach, CA, USA: arXiv, hal. 1–8.

Gu, J. dkk. (2018) 'Recent advances in convolutional neural networks', *Pattern recognition*, 77, hal. 354–377.

Hussain, R. dan Zeadally, S. (2018) 'Autonomous cars: Research results, issues, and future challenges', *IEEE Communications Surveys & Tutorials*, 21(2), hal. 1275–1313.

Jogin, M. dkk. (2018) 'Feature extraction using convolution neural networks (CNN) and deep learning', in *2018 3rd IEEE international conference on recent trends in electronics, information & communication technology (RTEICT). 2018 3rd IEEE international conference on recent trends in electronics, information & communication technology (RTEICT)*, IEEE, hal. 2319–2323.

Polri, K. (2022) *Statistik Laka, Korlantas Polri*. Available at: https://korlantas.polri.go.id/statistik-laka/ [Online] (Diakses: 7 June 2022).

Prasetyo, E.W. dkk. (2020) 'Spatial Based Deep Learning Autonomous Wheel Robot Using CNN', *Lontar Komputer: Jurnal Ilmiah Teknologi Informasi*, 11(3), hal. 167–177.

Radiuk, P.M. (2017) 'Impact of training set batch size on the performance of convolutional neural networks for diverse datasets', *Information Technology and Management Science*, 20(1), hal. 20–24.

Rawat, R., Patel, J.K. dan Manry, M.T. (2013) 'Minimizing validation error with respect to network size and number of training epochs', in *The 2013 international joint conference on neural networks (IJCNN). The 2013 international joint conference on neural networks (IJCNN)*, Dallas, TX, USA: IEEE, hal. 1–7.

Shrestha, A. dan Mahmood, A. (2019) 'Review of deep learning algorithms and architectures', *IEEE access*, 7, hal. 53040–53065.

Sutandi, A.C. (2015) 'Pentingnya Transportasi Umum untuk Kepentingan Publik', *Jurnal Administrasi Publik*, 12(1), hal. 19–34.

Vijitkunsawat, W. dan Chantngarm, P. (2020) 'Comparison of Machine Learning Algorithm's on Self-Driving Car Navigation using Nvidia Jetson Nano', in *2020 17th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON). 2020 17th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, Phuket, Thailand: IEEE, hal. 201–204.

Vogelsang, D.C. dan Erickson, B.J. (2020) 'Magician's corner: 6. TensorFlow and TensorBoard', *Radiology: Artificial Intelligence*, 2(3), hal 1-3.

Wang, Q. dkk. (2022) 'A comprehensive survey of loss functions in machine learning', *Annals of Data Science*, 9(2), hal. 187–212.