Optimizing Deep-Neural-Network-Driven Autonomous Race Car Using Image Scaling

Yaqub Mahmoud^{1,*}, Yuichi Okuyama^{1,**}, Tomohide Fukuchi^{1,***}, Tanaka Kosuke^{1,****}, and Iori Ando^{1,†}

¹School of Computer Science and Engineering, The University of Aizu, Aizuwakamatsu, Japan

Abstract. In this work we propose scaling down the image resolution of an autonomous vehicle and measuring the performance difference using pre-determined metrics. We formulated a testing strategy and provided suitable testing metrics for RC driven autonomous vehicles. Our goal is to measure and prove that scaling down an image will result in faster response time and higher speeds. Our model shows an increase in response rate of the neural models, improving safety and results in the car having higher speeds.

1 Introduction

Autonomous driving algorithms give vehicles the capability of sensing their environment and moving safely with little or no human input. Autonomous Driving (AD) scenarios are considered very complex environments as they are highly dynamic containing multiple object classes that move at different speeds in diverse directions [1]. Since the DARPA Grand Challenge competition [2], autonomous driving has been actively researched [3]. Significant progress in Machine Learning (ML) techniques like Deep Neural Networks (DNNs) over the last decade has enabled the development of safety-critical ML systems like autonomous cars. Several major car manufacturers including Tesla, GM, Ford, BMW, and Waymo/Google are building and actively testing these cars [1].

Modern autonomous vehicles combine a variety of sensors and cameras to perceive their surroundings and utilize advanced control systems to interpret the sensory information to identify appropriate navigation paths. These various sensors each have their own advantages and disadvantages. For instance, ultrasonic sensors provide good performance of depth measurement for close obstacles but they lack semantic information and perform poorly for far objects. Camera sensors, on the other hand, provide rich color information from which scene semantics can be extracted. However, they lack information on depth and rely on scene illumination [2]. One goal of autonomous driving is to make a vehicle sense its environment and navigate without human input [4], and to maintain control similarly to humans so that the vehicle's motion is interpretable and comfortable [3].

Now, the major challenge becomes how to get the neural networks into series production cars, in a safety-

conform way [5]. There have been a lot of strides recently in the development of open source software for small portable autonomous vehicles. These portable vehicles are aimed at enthusiasts who are curious about machine learning and also at researchers who want to develop better and more enhanced autonomous driving algorithms. They are cheaper than actual autonomous cars and offer an environment to speedily develop and test the driving algorithms.

However, before a neural network finds its way into series production cars, it has to first undergo strict assessment concerning functional safety [5]. Although open-source and widely available, limited testing has been done on the true capabilities of these small portable vehicles and more work is needed to see if the results they provide us with can be consistent and reliable. This research down-scales the images from a portable autonomous racing car and tests both the modified and pre-modified system with the aim of achieving an increase in speed and response rate of the autonomous vehicle. The goal is to increase the speed and response rate of the autonomous vehicle while assessing its reliability for research.

There have been several research using RC cars, however, no one has asked the question of how varying degrees of image sizes affects the overall behaviour of a deep-neural-network-driven autonomous car. To our best knowledge, we are the first to look into and question whether or not RC vehicles are suitable for research activities. We also believe that we are the first to look at the influence of image resolution on vehicle speed. The contributions of our work are as follows:

- We demonstrate that downscaling the image size of an autonomous vehicle results in higher speeds.
- We also show downscaling images results in faster response rate and thus increases safety.
- We analyze the different neural network models available in the system.

*e-mail: m5222108@u-aizu.ac.jp

**e-mail: okuyama@u-aizu.ac.jp

***e-mail: m5222102@u-aizu.ac.jp

****e-mail: s1260138@u-aizu.ac.jp

†e-mail: s1260211@u-aizu.ac.jp

- We provide a testing framework for research using RC based autonomous vehicles.
- We perform testing on the capabilities of RC driven autonomous vehicles.
- We provide experimental evidence proving our hypothesis.

The rest of this paper is organized as follows: Chapter 2 provides more background information on deep learning and RC autonomous cars. In Chapter 3, we provide detailed information on the neural network models available for use in the system. Chapter 4 describes the system model in use. Chapter 5 describes our simulation and experiment setup. In Chapter 6, we present our results. Finally, Chapter 7 concludes the paper.

2 Background

2.1 Deep Learning for Autonomous Driving

Classical methods provide of general self-driving poor performance compared to deep learning methods in addition to high complexity due to complicated pipelines used. Deep learning algorithms are becoming successful beyond object detection [4, 2]. Deep learning can be applied to a self-driving car to recognize objects in front of the vehicle or to judge for itself when any situation arises. But to make these judgements, we need a lot of data as training data [5, 6]. In this paper, we collect learning data with an end-to-end method [1] and an autonomous driving technique announced by Nvidia using car games [7].

Figure 1: Deep-Neural-Network-Driven Autonomous Car

The key component of an autonomous vehicle is the perception module controlled by the underlying Deep Neural Network (DNN). Fig. 1 shows an autonomous car with a deep-neural-network. The DNN takes input from different sensors like camera, light detection and ranging sensor (LiDAR), and IR (infrared) sensor that measure the environment and outputs the steering angle, braking, etc. necessary to maneuver the car safely under current conditions. The first few layers of an autonomous car DNN extract low-level features such as edges and directions, while the deeper layers identify objects like stop signs and other

cars, and the final layer outputs the steering decision (e.g., turning left or right). Each layer of a DNN consists of a sequence of individual computing units called neurons. The neurons in different layers are connected with each other through edges and each edge has a corresponding weight [1].

2.2 Donkey Car

Donkey car is an open source Do-It-Yourself (DIY) self-driving platform for small vehicles. Donkey is a high-level self-driving library written in Python, developed with a focus on enabling fast experimentation and easy contribution [8]. It uses Raspberry Pi 4, or similar boards, together with a camera to control an RC car to drive through tracks autonomously. It is one of the most popular RC cars used by both enthusiasts and researchers who are striving to further develop the field.

3 Neural Network Architecture

In this section, we detail the different neural network architectures available in donkey car. Neural networks have proven to be a very good way of training images that have been collected [9]. It is crucial to perform testing on neural networks to make sure they are consistent in their results. Keras is a high-level neural network API written in Python. It is good for fast implementation and capable of running on TensorFlow. Donkey car uses keras to reproduce the steering and throttle based-on the image the camera sees [8]. Table 1 provides a summary of the different neural network models used in donkey car.

4 System Model

In this section we have provided a detailed list of the various components used in the system and also highlighted their importance.

4.1 Autonomous Driving System

Creating a system specification is one of the major challenges in testing a complex neural-based system like autonomous vehicle [1]. Fig. 2. shows an overview of our autonomous self-driving system that we used in this paper. Autonomous driving is achieved by recording images and your inputs while you're driving it. The data is then retrieved and fed into a neural network to train the data and output a file. Finally, the output is then placed into the car and it will use this for inference in order to achieve self-driving.

In summary, our car drives in the following steps:

- Use a controller to move the car and simultaneously collect data using the camera.
- Train the data collected on a google colaboratory server.
- Transfer the deep learning output from the server to the car.
- Input commands for self-driving.

Table 1: Description of Neural Networks of the System

Neural Network	Network Structure	Output
Keras Linear	5 conv. layers 2 dense layers	2 dense layers with 1 scalar output each for steering & throttle
Keras Categorical	5 conv. layers 2 dense layers	2 dense layers 16 & 20 w categorical output
Keras IMU	5 conv. layers 2 dense layers 3 dense layers for vector data	2 dense layers with 1 scalar output each for steering & throttle
Keras Latent	5 conv. layers to a 10x1x1 vector 6 Conv2dTranspose layers (b4 outputting images)	2 dense layers with 1 scalar output each for steering & throttle Outputs Images
Keras RNN	4 time distr. conv. layers 2 LSTM layers 3 dense layers Driving controls	1 dense layer with 2 scalar outputs for steering & throttle
Keras 3D	4 3D conv. layers (each with max pooling) 2 dense layers Driving controls	1 dense layer with 2 scalar outputs for steering & throttle
Keras Behavior	5 conv. layers 2 dense layers Driving controls	Categorical steering Throttle is similar to keras linear
Keras Localizer	5 conv. layers 2 dense layers	2 dense layers with 1 scalar each output for steering & throttle 1 categorical output for location

Figure 2: Overview of Autonomous Driving System

4.2 Data Acquisition

Cameras are a very common way of getting input data for autonomous vehicles. They have proven to excel in providing rich colors and a very comprehensive and detailed “world view”. A. Ghofrani et. al. [10] used an end-to-end RGB-based Indoor Camera Positioning System using deep convolutional neural networks. R. Blin et. al. [11] has demonstrated road scene analysis by using polarization-encoded images. We gathered over 10,000 images and each image has its own JSON file associated with it as shown in Fig. 3(a). Inside each JSON file, the steering angle, throttle speed and time is recorded for each image frame. Fig. 3(b). shows the distribution of the image data with number of images on the y-axis and steering angle on the x-axis. A +ve x-axis value indicates that the car was moving to the right while a -ve value indicates that the car was moving to the left side.

4.3 Raspberry Pi Controller

Several projects including [12, 7] have used raspberry pi in their implementations. And while raspberry pi can be used for various computations, it falls short when trying to run deep learning models on it [7]. For this reason, we only use the raspberry pi for the least computationally intensive tasks such as inference and we reserved running deep

learning models on google colaboratory. We also keep in mind the limited processing power of raspberry pi when making changes to the system. To achieve inference, we first installed raspian as a fundamental linux-based OS. We then installed python because it is the de-facto programming language for AI development. We then installed Tensor-flow as a library for deep learning. Finally, we installed the donkey car software to provide the controls necessary for donkey car hardware.

4.4 Google Colaboratory

Rendering a collection of images sufficiently large for machine learning algorithm training and evaluation is prohibitively slow with only one machine. Using cloud resources, in our case the Google Cloud Platform, we created many images in parallel to create a significant data set in a practical amount of time. Donkey car has both inference and training capabilities. But due to limited computation, we only use it for inference. For training, we use Google Colaboratory because it provides free powerful computational resources. Images were modified using image manipulation techniques borrowed from [13, 14]. We then modified the system to be able to operate on all varying degree of input image resolutions.

5 Simulation and Experiment

5.1 Simulation Setup

We set up a simulation experiment of donkey car on the Linux Ubuntu operating system as shown in Fig. 4. We obtained over 10,000 frames with the accompanying data entries as shown in Fig. 3(a). After that, the images were then down-scaled from the default resolution of 160x120 to the resolutions 120x90, 80x60 and 40x30 pixels. A

Figure 3: Image Data Visualization

Figure 4: Donkey Car Simulation in Linux Environment

thing to note is that all new resolutions have the same aspect ratio of 4:3 as the default resolution. This was done so as to maintain a level of consistency in our tests and results.

After rescaling the image data as shown in Fig. 5(a), we uploaded them separately to google colab for training. The training results in a H5 file. The H5 file is then uploaded to the simulation vehicle to be used for inference that will result in self-driving. Testing was done only for the Keras Linear and Keras Categorical neural network models.

5.2 Experiment Setup

We also set up an experiment by creating a simple real-world to obtain 10,000 frames and then test the autonomous car driving around the course multiple times. We explored the influence of reducing the resolution of an

image on the speed of the self-driving vehicle. An assessment of network resilience in terms of accuracy has been done by Blasinski et al.[9]. But our system seeks to see an increase in speed given lower resolution images. Our proposed method downscales the image size from 160x120 to 120x90 pixels as shown in Fig. 5(b). Both of which have an aspect ratio of 4:3. Having tried various image resolutions, we found 120x90 to be the perfect mix between increasing the driving response and retaining as much relevant information as possible for self-driving.

We then performed evaluation on both the default system and also the modified system. To evaluate the system, we first got training data from the autonomous vehicle using the Keras IMU model. Since Keras IMU has all the inputs required by other models, and since colab automatically parses the data needed for each model, the other models can use the same training data. In other words, there is no need of obtaining training data from each and every model. To evaluate the system we chose to measure the frames-per-second (FPS) and lap time of the vehicle. Testing for the above criteria was done for the Keras Linear and Keras Categorical neural network models. Donkey car currently only supports the old IMU MPU6050. We found a way to implement a newer version of IMU MPU9250 which provides more accurate relay signals and we made donkey car support it.

6 Results

We tested each neural network 5 times for each evaluation criteria and showed the average in the results section. We then separated the results of testing into 2 main categories: when the system is default and when the system is changed.

6.1 Simulation Result

Fig. 6 shows results from testing of inference of the autonomous car in the simulation. We obtained four different

Figure 5: Downsampling Images to Different Resolutions

Figure 6: Simulation results showing a gradual increase of speed as the scale of the image decreases

graphs, one for each image size. The bottom half shows the speed at which the car self-drove itself. The results show an increase in speed the lower the resolution gets. Thus, with regards to our goal of increasing the speed of self-driving, we have been successful in the simulation.

6.2 Frames-Per-Second (FPS)

We developed a frame rate recording application into the system to record the fps during the inference stage when the car is driving itself. This give us more insight into the performance of the autonomous algorithms in play. The code below was implemented in the vehicle.py file of don-key car.

```
start_time = time();
counter = 0;
while(1){
    counter++;
    if(counter == 10){
        fps = 10 / (time() - start_time);
        counter = 0;
        start_time = time();
    }
}
```

Fig. 7 shows an improvement in frames-per-second after changing the image size. This shows that a lower image size increases the response rate of the vehicle. Compared to [2] our system is at least 80% faster in frames-per-second and thus more responsive.

Figure 7: Average Frames-Per-Second

6.3 Lap Time

We set up a track and Donkey car was allowed to run 5 laps around the track for each neural network model for both the 160x120 and 120x90 image resolutions. The total time taken to complete the 5 laps was then recorded. Fig. 8 shows a decrease in total time taken to go round the track once the image has been downsampled from 160x120 to 120x90 pixels. This means that we achieved an increase in speed after downsampling the image size.

Figure 8: Total Lap Time

7 Conclusion

In this work, we succeeded in making an RC car run faster and have a higher response rate by downscaling the image size. We have shown that we were able to obtain and measure an increase in speed and response time for both the simulation and on the real-world race track. By increasing response time, we have made the system faster. From our results we deduce that the categorical model is the best in terms of consistency and reliability for research. However, better standards and processes are required in order to allow the creation of algorithms with better predictability and safety in mind. And although open-source autonomous vehicles have come a long way, more work is needed if they are to be used for serious research work requiring very high degrees of accuracy. We hope that this study encourages further investigation into making RC vehicles more adaptable for use in research activities.

8 Acknowledgements

To the Fabo Company, Japan for assisting with their expertise. The first author had received financial support from the Otsuka-Toshimi Scholarship Foundation.

References

- [1] Yuchi Tian et al. "Deeptest: Automated testing of deep-neural-network-driven autonomous cars". In: *Proceedings of the 40th international conference on software engineering*. 2018, pp. 303–314.
- [2] Hazem Rashed et al. "Fusmodnet: Real-time camera and lidar based moving object detection for robust low-light autonomous driving". In: *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 2019, pp. 0–0.
- [3] Justin Zheng, Kazuhide Okamoto, and Panagiotis Tsiotras. "Vision-Based Autonomous Vehicle Control using the Two-Point Visual Driver Control Model". In: *arXiv preprint arXiv:1910.04862* (2019).
- [4] Xinlei Pan et al. "Virtual to real reinforcement learning for autonomous driving". In: *arXiv preprint arXiv:1704.03952* (2017).
- [5] Qing Rao and Jelena Frtunikj. "Deep learning for self-driving cars: chances and challenges". In: *Proceedings of the 1st International Workshop on Software Engineering for AI in Autonomous Systems*. 2018, pp. 35–38.
- [6] Qi Zhang and Tao Du. "Self-driving scale car trained by Deep reinforcement Learning". In: *arXiv preprint arXiv:1909.03467* (2019).
- [7] Juntae Kim et al. "Deep learning algorithm using virtual environment data for self-driving car". In: *2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIIC)*. IEEE. 2019, pp. 444–448.
- [8] *Donkey Car Documentation*. URL: docs.donkeycar.com.
- [9] Henryk Blasinski et al. "Optimizing image acquisition systems for autonomous driving". In: *Electronic Imaging 2018.5* (2018), pp. 161–1.
- [10] Ali Ghofrani, Rahil Mahdian Toroghi, and Seyed Mojtaba Tabatabaie. "ICPS-net: an end-to-end RGB-based indoor camera positioning system using deep convolutional neural networks". In: *Twelfth International Conference on Machine Vision (ICMV 2019)*. Vol. 11433. International Society for Optics and Photonics. 2020, p. 1143323.
- [11] Rachel Blin et al. "Road scenes analysis in adverse weather conditions by polarization-encoded images and adapted deep learning". In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE. 2019, pp. 27–32.
- [12] Raj Shirolkar et al. "Self-Driving Autonomous Car using Raspberry Pi". In: *International Journal of Engineering Research Technology (IJERT) Volume 08, Issue 05* (2019).
- [13] Anju Sudhakaran and Manju Manuel. "High-Quality Image Scaling Using V-Model". In: *2018 International Conference on Circuits and Systems in Digital Enterprise Technology (ICCSDET)*. IEEE. 2018, pp. 1–4.
- [14] Yunfeng Zhang et al. "Single-image super-resolution based on rational fractal interpolation". In: *IEEE Transactions on Image Processing* 27.8 (2018), pp. 3782–3797.