Comparison of Machine Learning Algorithm's on Self-Driving Car Navigation using Nvidia Jetson Nano

Wuttichai Vijitkunsawat
Electronics and Telecommunication Engineering,
Rajamagala University of Technology Krungthep
Bangkok 10120, Thailand
wuttichai.v@mail.rmutk.ac.th

Peerasak Chantngarm
Electronics and Telecommunication Engineering,
Rajamagala University of Technology Krungthep
Bangkok 10120, Thailand
cpeerasak@gmail.com

***Abstract*— The vehicle is the most facility for business and living in the present. The vehicle is used in abundant activities, which lead to an increasing number of accidents on the roads. Many statistical reports pointed that more than 80% of accident causes came from direct human causes such as violating the speed limit, illegal overtaking, and suddenly cutting in. Therefore, the self-driving car was rapidly developed by starting a scaled RC-Car platform. This paper presents a self-driving car model to study behavior by using the three machine learning algorithms: SVM, ANN-MLP, and CNN-LSTM in 3-speed levels and 3 scenarios, both with obstacles and without obstacle. The results show that CNN-LSTM is the best accuracy in every scenario and speed levels.**

Keywords—*Self-driving car, machine learning, model prediction*

I. INTRODUCTION

Nowadays, transportation has played a critical role in people's daily lives. It can be seen that people can travel more convenient and save time than in the past, especially cars. Therefore, the number of vehicles has increased every year.

This cause leads to a dramatical rise in road traffic accident

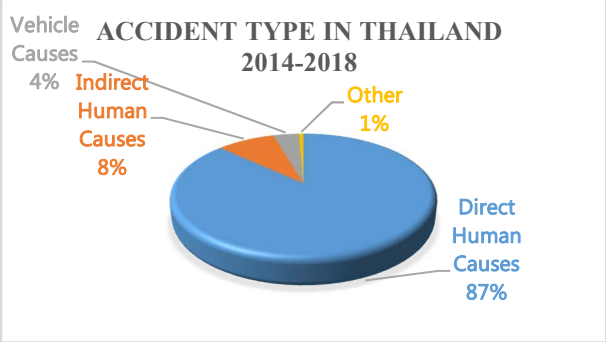

Fig.1 Traffic Accident Type in Thailand

From Fig.1, the given pie chart illustrates the proportion of traffic accident types in Thailand from 2014-2018. According to the pie chart, 87% of accident type is direct human causes such as violating the speed limit, suddenly cutting in, illegal overtaking, and failure to headlights, compared to 8% and 4% which is indirect human causes such as drunkenness and drowsiness and vehicle causes respectively [2]. Furthermore, Thai Road Safety Culture Data Center in 2019 found that there was almost 12,000 dead form traffic accident [3]. A few years ago, numerous researchers pointed out that self-driving car technology can decline the number of accidents from direct human causes and encourage more convenient for disabled and aging people. Besides, it also falls stress from driving as well. In 2005, Defense Advance Research Project Agent (DARPA) began autonomous robots' competition to find the best solution for reducing accidents, evading obstacles on the road, making conditions by using all sensors around them [4]. In 2007, a model car based on Fuzzy-logic control using Lab-View was proposed in [5]. The research proposed the learning and following method for this self-driving car via the computer vision. After that, copious techniques for self-driving cars were dramatically developed. Some studied were brought the FPGA technology and ROS system to progress self-driving cars [6]. In 2018, the researcher developed a self-driving car model by using Raspberry Pi3 b+ and Forest Tree (Machine learning algorithms) for processing data [7].

In this paper mainly presents the comparison of 3 crucial algorithms: Support Vector Machine (SVM), Artificial Neural Network Multilayer Perceptron (ANN-MLP) and Convolutional Neural Network - Long Short Term Memory (CNN-LSTM) to training and testing input images on different scenarios both speed levels and the number of obstacles on the road to find the best machine learning algorithm. Each algorithm is developed on the Nvidia Jetson Nano Developer Kit board and using the Proportional-Integral-Derivative Controller technique (PID) system for controlling the directions and speed movement of the self-driving car model (SDCM). Section II describes the all overview of the SDCM structure platform and the fundamental background of each algorithm. Section III explains all steps of the methodologies during the experiment environment, data set, and process of data evaluation. Section IV presents the experiment about all scenarios and parameters to compare the accuracy performance of each algorithm. Finally, the concluding remarks are given in Section V.

II. SYSTEM OVERVIEW

A. Hardware Platform

In this research, we built a self-driving car for collecting data. Our car comprised of three main subsystems: (1) Nvidia Jetson Nano is a small microprocessor board for developing and training models by using GPU 128-core Maxwell to rapidly processing AI frameworks and models for applications such as image classification, object detection and segmentation [8]. (2) The STM32 microcontroller board is used for controlling the direction and movement of a self-

driving car model [9]. (3) Mini camera IMX-219 is a small module for the Jetson Nano board to detect road marks and object detections. It is capable of 3280x2464 pixels, 8 Megapixels, and 160º diagonal fields of view. Overall, size of self- driving car is 15 cm x 25cm x 15 cm (width x length x height).

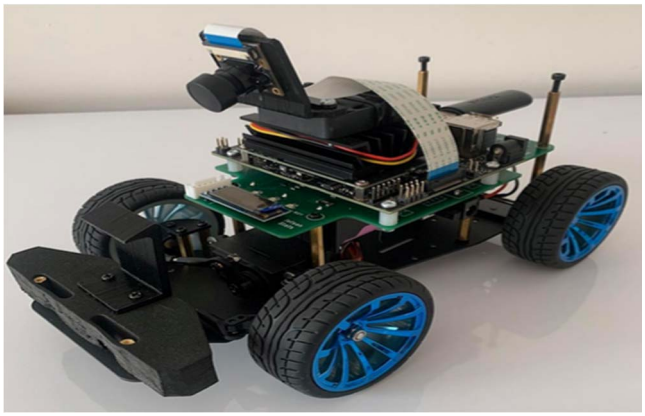

Fig. 2 Self-Driving Car Model

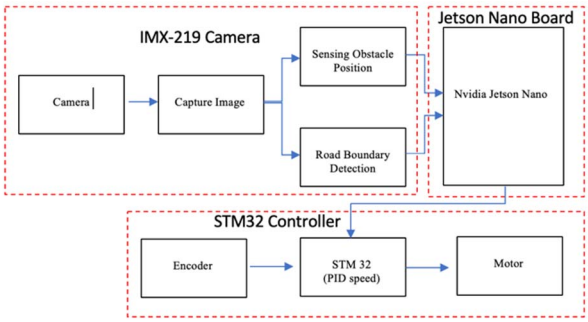

Fig. 3 Block Diagram of Self-Driving Car Model

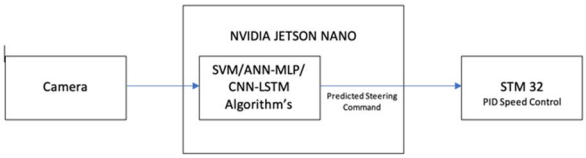

Fig. 4 Block Diagram within Jetson Nano

B. Relative Algorithm

In the algorithm's parts, we focused on supervised learning algorithms to apply in this work. For the training model, we used the three models, which are Support Vector Machine (SVM), Artificial Neural Network Multilayer Perceptron (ANN-MLP), and Convolution Neural Network - Long Short Term Memory (CNN-LSTM) for comparison to finding the best accuracy for our SDCM.

a) The SVM can encourage both classification and regression problems, including linear and non-linear hyper-plane by using a kernel function to reduce complicated feature spaces that use the decision function is below [10]:

$$f(x) = sgn(\sum_{i=1}^{n} y_i \alpha_i K(x_i, x) + b) \qquad (1)$$

where $K(x_i, x)$ is the kernel

b.) The ANN-MLP is an artificial neural network and it is a nonparametric estimator to use for classifying and detecting objects. The fundamental characteristic is the feed-forward method; however, the parameter can be changed to the back-propagation algorithm that comprises three layers: an input layer, hidden layer, and output layer. By the activate function, we generally use a sigmoid function [11].

$$o(x_i) = \sum_{i=1}^{n} w_i x_i + w_o \qquad (2)$$

where x_i is input data.
w_i is weight.
w_o is the bias value.

c.) Convolution Neural Network Long Short-Term Memory (CNN-LSTM) is one of the models which is suitable for fixing classification problems, which consist of five main layers: Convolution stage, Detector stage, Pooling stage, LSTM stage, and Fully connected stage. The primary function of the convolution state is the image separation parts, which are filtered elements such as edges, colors, sharps of pictures, and sends it to the detector stage. The detector state receives data form the convolution state and converts it to the non-linear platform and adds activate function, ReLU, to separate characteristics of each data. The pooling state is a filter that finds the highest value in the area where the filter is placed on the data and then selects the highest value on that filter as a new result. It shifts the filter to the stride specified by the filter size of max pooling. Next, the LSTM stage is designed for solving memory cell problems in which it can control forgetting, writing, and reading of stages.

$$f_t = \sigma(W_{x^f} x_t + W_{h^f} h_{t-1} + b_f) \qquad (3)$$

$$i_t = \sigma(W_{x^i} x_t + W_{h^i} h_{t-1} + b_i) \qquad (4)$$

$$g_t = tanh(W_{x^c} x_t + W_{h^c} h_{t-1} + b_c) \qquad (5)$$

$$c_t = f_t \Theta c_{t-1} + i_t \Theta g_t \qquad (6)$$

$$o_t = \sigma(W_{x^o} x_t + W_{h^o} h_{t-1} + b_o) \qquad (7)$$

$$h_t = o_t \Theta \tanh(c_t) \qquad (8)$$

where f is forgot gate
i is input gate
c is cell memory state
o is output gate
Θ is Hadamard product
g is input modulation gate

The fully connected (FC) stage, which is the final stage of CNN-LSTM, has been proven very successful in recognizing and classifying images for computer vision. All processed CNN-LSTM are shown in Fig. 6 [12]

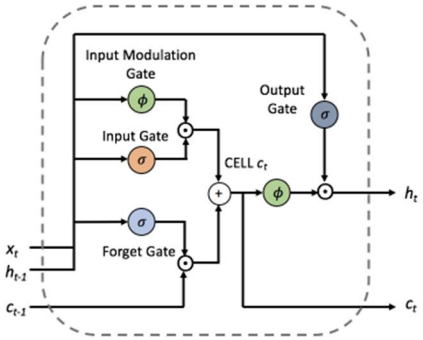

Fig 5. Long Short-Term Memory Network (LSTMs)

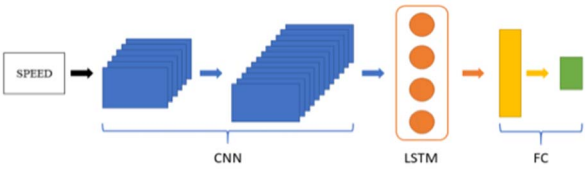

Fig. 6 The CNN-LSTM model architecture

III. METHODOLOGY

In this section, we separate three main parts for stepping solutions. (1) Experimental environment (2) data set to use various models and (3) process of data evaluation.

A. The Experimental Environment

It consists of 2 lanes by each lane wide 35 centimeters, and the size of the overall track is 3m x 5m (length x width). Moreover, the experimental track still has two obstacles, which are a garbage truck and a fire truck on the road. All obstacles and experimental route are shown in Fig. 7 and 8

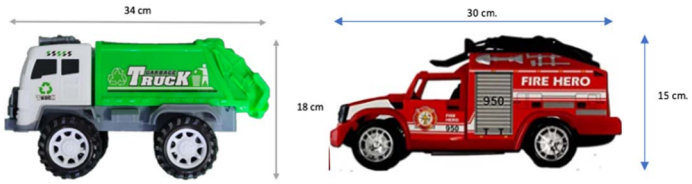

Fig 7. Obstacle object (Garbage truck and Fire truck)

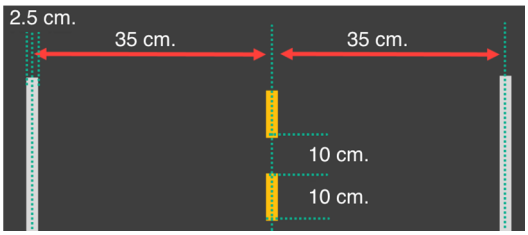

Fig 8. Road width and lane width

B. Data Set

Data Set is input image information by using the snapshot camera at a rate of 5 frames per second on the car model to use the training algorithm for each model. Jetson Nano record the images and driving information by the user manually driving the car around the route with speed about 1 km/h, and the original resolution of images is 320 x 240 pixels which car model collected data images contained over 3600 images. Sample images of the data set are shown in Fig. 9

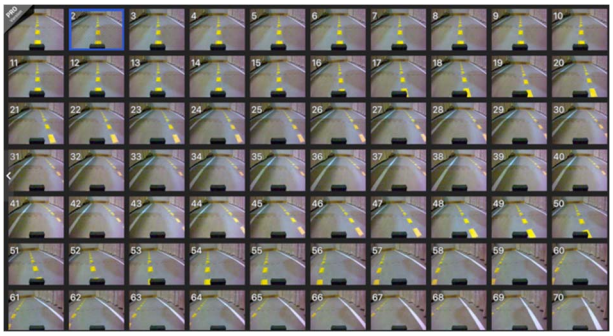

Fig 9. Data set between collected images

C. Process of Data Evalution

In this part, we received data set from previous processes and brought all data sets to use training on three models. In the first step, the data set was resized from 320 x240 pixels to smaller and an attached label to be raw data. Next, raw data was converted to CSV files separated into two groups, which were 80% of the data set for training models and 20% of another group for predicting each model. In the CNN-LSTM model, the Network architecture consisted of 9 layers, which be separated Convolution Layer 5 layers and Fully connected four layers. Moreover, Convolutional layers were designed to perform characteristic extraction and were selected practically via a sequence of experiments. For the convolution in the first three layers, comprise of 5x5 kernel, 2x2 strides, and two last layers of convolution are 3x3 kernel. The depth of each layer is 24, 36,48, and 64, respectively. In the part of fully connected layers are decreased sizes: 64, 16, 8, and 1. Furthermore, all hidden layers provide with Rectified Linear Unit (ReLU) [14] (eq. 9) to improve convergence. The whole network got about 1,213,941 parameters. Additionally, we used 0.3 dropout rates to prevent over-fitting. After that, we brought data from after training and testing to compare with each other by using the Mean Square Error (MSE) (eq.10) technique to reduce loss. Furthermore, we still used Adaptive Moment Estimation (Adam) optimizer [15] technique to decrease the learning rate of some parameters to find the best accuracy model for SDCM.

$$f(x) = \max(0, x) = \begin{cases} 0 & for\ x \leq 0 \\ x & for\ x > 0 \end{cases} \quad (9)$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \bar{Y})^2 \quad (10)$$

where n is sample of data
Y_i is data after training algorithm
$\bar{Y}$ is data from testing model

All processes are shown in Fig 10.

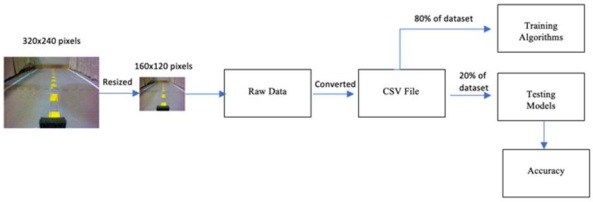

Fig 10. Block diagram of data evaluation models

IV. EXPERIMENTAL RESULT

In this section, we propose three-speed levels and three

scenarios for comparing the accuracy of each algorithm: SVM, ANN-MLP, and CNN-LSTM. For the first experiment, we set up the three-speed levels, which are 1, 2, and 3 km/h, respectively, by without obstacle on the road.

In the first experiment, SDCM has to run on the road without obstacle and test on three different speed levels to compare with the accuracy of each algorithm.

Fig 11. Testing SDCM on the road without obstacle

Table I. Accuracy of each algorithm (without obstacle)

Algorithm	Accuracy (%)		
	Speed		
	1 km/h	2 km/h	3 km/h
SVM	82.5	74.4	68.2
ANN-MLP	78.1	72.6	67.3
CNN-LSTM	88.7	81.3	75.9

According to table I, it can be seen that the percentage of the accuracy rate of the CNN-LSTM algorithm is the highest performance of all models at every speed level without obstacle on the road.

For the second experiment, we add an obstacle on the road (garbage truck) and test in same 3 speed levels to compare each algorithm performance.

Fig 12. Testing SDCM on the road with an obstacle

Table II. Accuracy of each algorithm (with an obstacle)

Algorithm	Accuracy (%)		
	Speed		
	1 km/h	2 km/h	3 km/h
SVM	79.5	72.7	67.3
ANN-MLP	73.4	70.1	66.1
CNN-LSTM	81.2	80.7	77.9

According to table II, although we add one condition to a scenario by adding an obstacle, CNN-LSTM is still the best accuracy algorithm.

In the final experiment, we add two obstacles on the road, and the position of obstacles are different lanes.

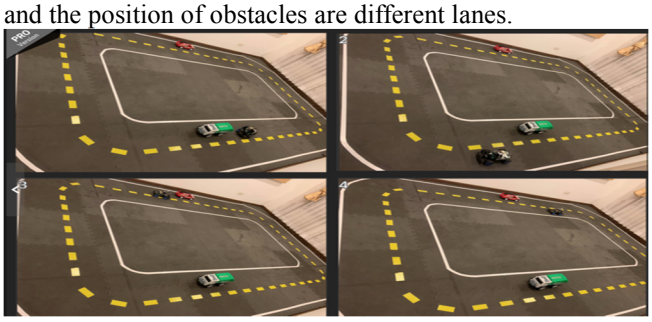

Fig 13. Road with two obstacles and different lanes

Table III. Accuracy of each algorithm (with 2 obstacles)

Algorithm	Accuracy (%)		
	Speed		
	1 km/h	2 km/h	3 km/h
SVM	78.8	69.7	67.0
ANN-MLP	72.8	68.2	66.1
CNN-LSTM	80.5	78.1	75.6

According to table III, it is apparent that even though we add more the obstacles on the road, the percentage of accuracy rate of the CNN-LSTM algorithm is higher than any other algorithm in the experiment.

V. CONCLUSION

This paper presents a comparison of 3 famous algorithms of machine learning: SVM, ANN-MLP, and CNN-LSTM on different scenarios and different speed levels. From the experiment, it can be seen that the percentage of the accuracy rate of the CNN-LSTM algorithm is the highest efficiency, not only with obstacles but also without obstacle. However, the percentage of the accuracy rate of all algorithms gradually drops when adding more obstacles and more high-speed levels.

REFERENCES

[1] Bureau of Highway Safety, Department of Highways. Retrieved from http://bhs.doh.go.th/files/accident/61/report_accident2561.pdf

[2] National Statistical Office Ministry of Information and Communication Technology. Retrieved from http://web.nso.go.th/index.htm

[3] Thai RSC Statistical Report. Retrieved from www.thairsc.com

[4] Buehler, K. lagnemma, and S. Singh, "The 2005 DARPA Grand Challenge", The great robot race, Vol. 36. Springer Science & Business Media, 2007.

[5] Juan Manuel Ramirez, Pilar Gomez-Gil and Filberto Lopez Larios, "A Robot-Vision System for Autonomous Vehicle Navigation with Fuzzy-logic Control using Lab-View", Electronics, Robotics and Automotive Mechanics Conference, 2007

[6] Musashi Aoto, Yousuke Numata and Yasutaka Wada, " Development of an FPGA controlled Mini-Car toward Autonomous Driving", Internation Conference on Field-Programable Technology, pp.403-405, 2018

[7] Uvais Karni, S. Shreyas Ramachadran, K. Sivaraman and A.K. Veeraraghavan, "Development of Autonomous Downscaled Model Car using Neural Network and Machine Learning", Proceeding of the Third International Conference on Computing Methodologies and Communication, pp. 1089-1094, 2019.

[8] NVIDIA Autonomous Machines (Jetson Nano Developer Kit). Retrieved from https://developer.nvidia.com/embedded/jetson-nano-developer-kit

[9] STM32 32-bits Arm Cortex MCUs. Retrieved from https://www.st.com/en/microcontrollersmicroprocessors/stm32-32-bit-arm-cortex-mcus.html

[10] Madan Somvanshi, Shital Tambade, Pranjali Chavan and S.V. Shinde, "A Review of Machine Learning Techniques using Decision Tree and Support Vector Machine", International Conference on Computing Communication Control and Automation, 2016.

[11] Susmita Ray, "A quick Review of Machine Learning Algorithm", International Conference on Machine learning, Big Data, Cloud and Parallel Computing, pp. 35-39, 2019

[12] Wanida Liyong and Peerapon Vateekul, "Traffic Prediction Using Attentional Spatial-Temporal Deep Learning with Accident Embedding", International Conference on Computational Intelligence and Applications, pp. 98-103, 2019

[13] Jie Meng, Anbang Liu Yang, Zhe Wu and Qingyang Xu, "Two-wheeled robot platform based on PID control ", International Conference on Information Science and Control Engineering, pp. 1011-1014, 2018.

[14] Yao Ying, Jianlin Su and Peng Shan,"Rectified Exponential Units for Convolution Neural Network", International Journal of IEEE Access, Vol. 7, pp. 101633 – 101640, 2019

[15] Zijun Zhang, "Improved Adam Optimizer for Deep Neural Networks", IEEE/ACM 26th International Symposium on Quality of Service, 201