SELF-DRIVING ROBOT CARS USING IMAGE PROCESSING AND DEEP LEARNING

Duong Dinh Tu, Phan Xuan Hieu, Hoang Tuan Hiep
Institute of Engineering and Technology, Vinh University, Vietnam
Received 19/10/2022, accepted for publication on 23/11/2022
DOI: https://doi.org/10.56824/vujs.2022nt27

Abstract: In the field of image processing, image recognition is one of the major challenges for researchers in recent years. The goal of recognition is to detect and extract features in the images to classify samples into different layers. One problem of great interest in this field is self-driving robot cars using image processing. The aim of this research is to present a deep learning method to solve the path recognition matter for robots. A self-driving robot model will be built based on the convolutional neural network with the use of different layers to automatically extract the best features in the image. Experimentation procedure has been carried out and results with low error rates had been obtained.

Keywords: Self-driving robot model; image processing; Convolutional Neural Network; deep learning.

1. Introduction

In recent years, many outstanding achievements in the field of image processing have been witnessed. Great image processing systems such as Google, Facebook or Amazon have developed a lot of smart features, such as facial recognition, self-driving cars or automated delivery robots, etc. Convolutional Neural Network (CNN) is one of the most advanced deep learning models, which will help build today's inttelligent systems with high accuracy [1].

Self-driving robots are an issue of great interest because they can be widely applied in many fields [3]. Self-driving robots have been widely applied for transportation in factories, hospitals, working in special environments that are difficult to access by humans, and has evolved to an even higher level of self-driving cars.

From the success of neural networks in image processing, a CNN model had been built to solve the issue of self-driving robots. Self-driving robots are built through three procedures, including data collection, training and experimentation [3], [5], [7]. For this recognition system, Raspberry Pi 4 embedded computer has been used to store information about the path image and steering angle of the self-driving robots, then perform the training procedure. By the time of experimentation, self-driving robots will recognize the path to perform the appropriate steering angle and speed accordingly.

2. Research Methods

2.1. Self-driving robot model

Self-driving robot model includes the following components: an ordinary robot model, a Raspberry Pi 4 embedded computer and a camera (Figure 1). The camera is connected to the Raspberry Pi 4 to capture pictures, which will be sent to the CNN as the input images. The control signals are generated in the python coding, derived from the

Email: duongdinhtu@vinhuni.edu.vn (D. D. Tu)

execution of the Raspberry Pi 4 board, used to control the self-driving robot. The CNN model predicts one of possible outcomes: left, right or forward. When a predicted outcome is obtained, the corresponding signal will be activated, thereby activating the corresponding signal with the support of the controller, assisting the robot to move in a specific direction.

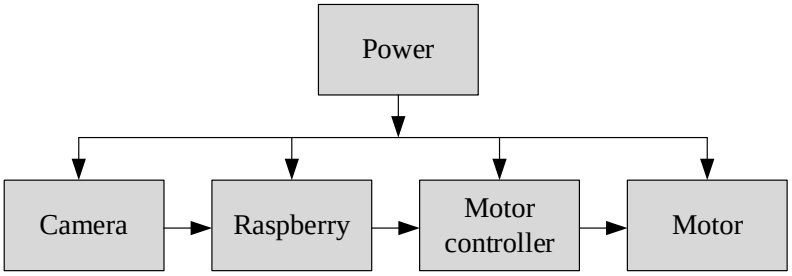


Fig. 1: *System Architecture*

2.2. CNN model

In this part, a CNN model will be constructed to solve the issue of path recognition for self-driving robot. Figure 2 shows the general architecture of the model [3], [5, [7], including two main parts:

- A convolution engine that separates and identifies different features of the image for analysis in a procedure called feature extraction. The feature extraction network consists of multiple pairs of convolutional or pooling layers;

- A fully connected layer uses the output from the convolution procedure and predicts the layer of the image based on the features extracted in previous stages.

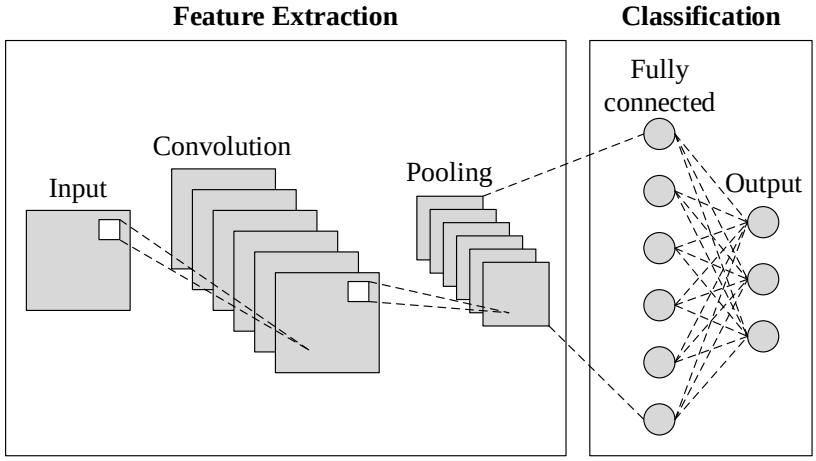


Fig. 2: *Convolutional neural network model*

The layers of CNN model include:

- Convolutional layer: This layer is used to extract various features from the input images, in which the convolution operation is performed between the input image and a filter of specific size $m \times m$. By sliding the filter over the input image, the dot product is taken between the filter and the parts of the input image with respect to the size of the filter. The output is termed as the feature map that provides information about the image such as the corners and edges. The convolution layer in CNN passes the result to the next layer after applying the convolution operation in the input;

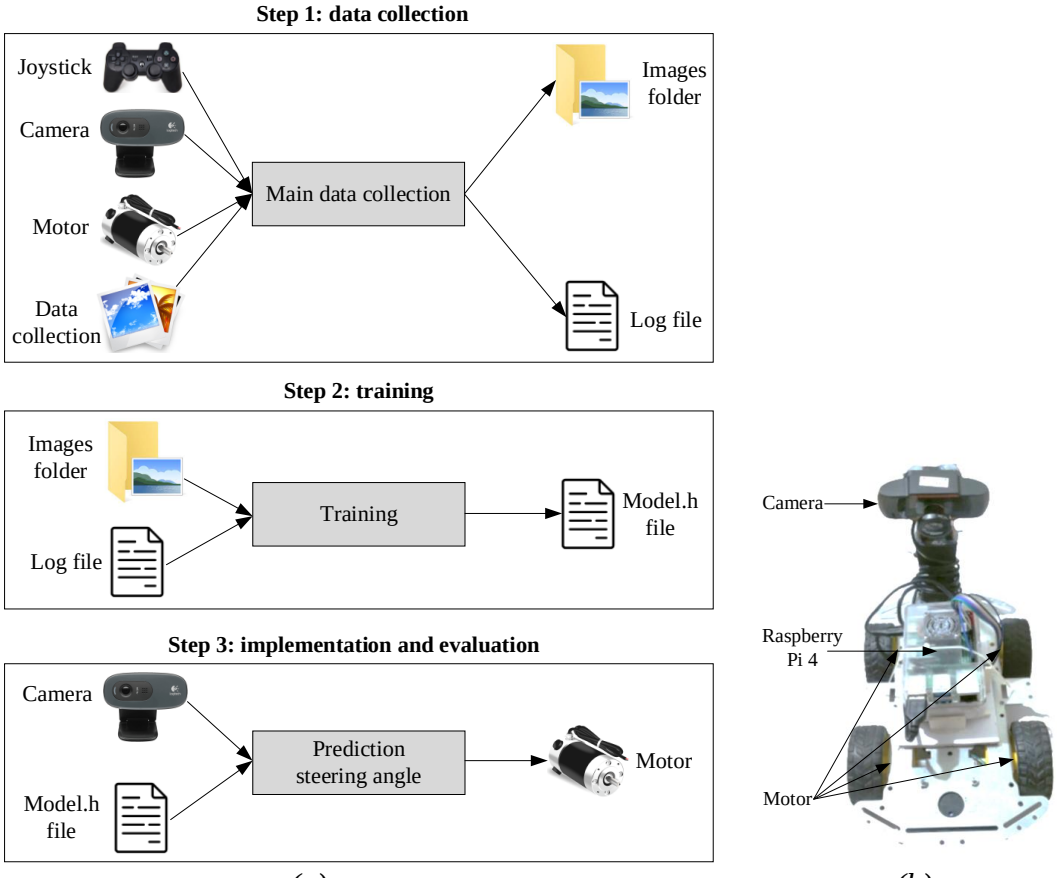
- Pooling layer: The next layer of CNN is the pooling layer. The primary aim of this layer is to decrease the size of the convolved feature map to reduce the computational costs. This is done by reducing the connections between layers and independently operating on each feature map. Depending upon the method used, there are several types of pooling operations. It basically summarises the features generated by a convolution layer. The pooling layer usually serves as a bridge between the convolutional layer and the fully connected layer;

- Fully connected layer: This layer consists of the weights and biases along with the neurons and is used to connect every input neuron to every output neuron between two different layers. These layers are usually placed before the output layer and form the last few layers of a CNN architecture. The classification process begins during this stage.

One of the most important parameters of the CNN model is the activation function. It adds non-linearity to the network. There are several commonly used activation functions such as the ReLU, softmax, tanh and sigmoid functions. Each of these functions has a specific usage. For our CNN model, softmax function is used.

3. Experimental results

The experimental procedure was conducted in three steps: data collection, training, implementation and evaluation as shown in Figure 3a. Self-driving robot is designed as shown in Figure 3b.



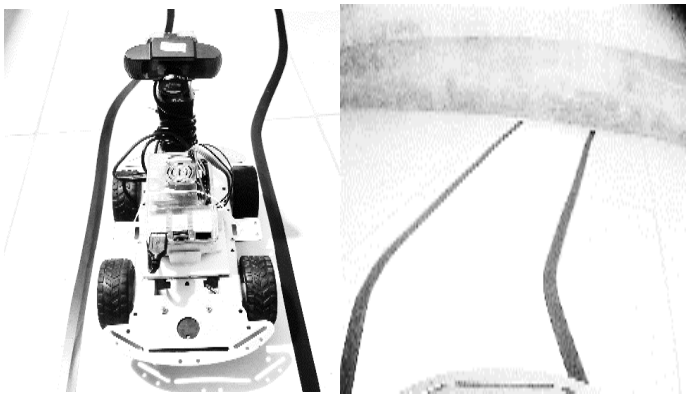
(a) *(b)*

Fig. 3: *a) Experimental procedure; b) Self-driving robot*

3.1. Dataset

In this step, training data is collected using a camera that records the robot's path and control angle of the robot with corresponding images. The joystick plays the role of controlling the robot to follow a predetermined route, the camera records the picture frames, while Raspberry Pi 4 embedded system plays the role of image and the steering angle processing center (Figure 4).

The robot was controlled for 10 minutes, resulting in 3000 images. This image stream is stored in the file format driving_log.csv (Figure 5), where column 1 contains the path to the image obtained from the camera, column 2 contains the corresponding steering angle. In column 2, value 0 means go straight, negative value means turn left, positive value means turn right. This data is then tested in self-driving mode, where the robot moves on its own over complex terrain without a controller, to see the performance of the CNN model.



(a) *(b)*

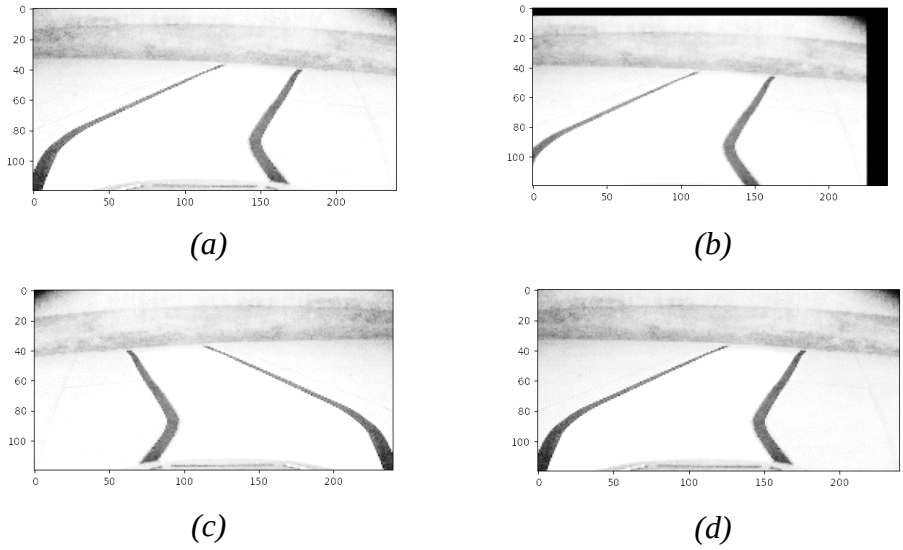
Fig. 4: *a) Robot performs data collection; b) Input image*

/home/pi/project/Code/cnn/Step1/DataCollected/IMG0/Image_1666182781518167.jpg	0
/home/pi/project/Code/cnn/Step1/DataCollected/IMG0/Image_1666182781558383.jpg	0
/home/pi/project/Code/cnn/Step1/DataCollected/IMG0/Image_1666182781598562.jpg	0
/home/pi/project/Code/cnn/Step1/DataCollected/IMG0/Image_1666182781642777.jpg	0
/home/pi/project/Code/cnn/Step1/DataCollected/IMG0/Image_1666182781682428.jpg	-0.44
/home/pi/project/Code/cnn/Step1/DataCollected/IMG0/Image_1666182781722145.jpg	-0.55
/home/pi/project/Code/cnn/Step1/DataCollected/IMG0/Image_1666182781762151.jpg	-0.55
/home/pi/project/Code/cnn/Step1/DataCollected/IMG0/Image_1666182781802257.jpg	-0.55
/home/pi/project/Code/cnn/Step1/DataCollected/IMG0/Image_166618278184216.jpg	-0.55

Fig. 5: *File driving_log.csv*

3.2. Building and training the model

- *Augment Data:* A CNN model can handle up to millions of parameters, adjusting the parameters requires millions of instances of training data. If the training data is too small, it can lead to overfitting. Therefore, image enhancement techniques were used to avoid this phenomenon. For more general data, the robot images were captured moving in different weather, light, road and traffic conditions. Since then, thousands of new versions of the image in real time have been created. Some techniques are shown in Figure 6.

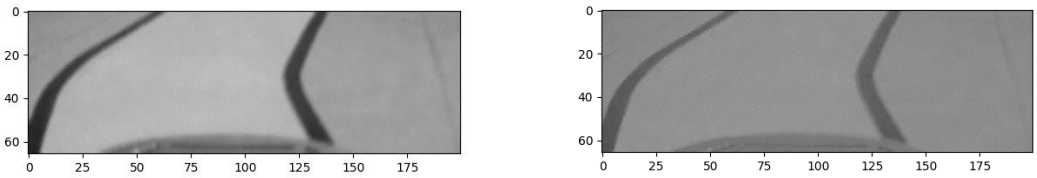


(a) *(b)*

(c) *(d)*

Fig. 6: *Augmenting variations of images:(a) original image; (b) pan image; (c) horizontally flipped image; (d) adjusted for brightness*

- *Preprocess Data:* The images in the dataset are all RGB images. So, for the ease of model training, these images are converted to YUV format (Figure 7).



(a) *(b)*

Fig. 7: *Converting the image: (a) RGB image; (b) YUV image*

The images are also blurred using openCV's Gaussian blur and resized so that unimportant parts like the background scene were cut out. Each pixel is then divided by 255 for equal priority. These pixels will receive a value of 0 or 1. Part of the Python code for this task is shown as follows:

```
def preProcess(img):
img = img[54:120,:,:]
img = cv2.cvtColor(img, cv2.COLOR_RGB2YUV)
img = cv2.GaussianBlur(img,  (3, 3), 0)
img = cv2.resize(img, (200, 66))
img = img/255
return img
```

- *Training model:* A CNN model will be trained on the dataset obtained in the previous step. In the experiment, the model is built with the application of several optimization algorithms and the setting of different values of learning rate. The dimensions for the filters are set to 5×5 and 3×3 respectively. The example below is

a training model with the use of adam optimization algorithm and the learning rate value is set to 0,0001:

```
def createModel():
  model = Sequential()
  model.add(Convolution2D(24, (5, 5), (2, 2),
          input_shape=(66, 200, 3), activation='elu'))
  model.add(Convolution2D(36, (5, 5), (2, 2), activation='elu'))
  model.add(Convolution2D(48, (5, 5), (2, 2), activation='elu'))
  model.add(Convolution2D(64, (3, 3), activation='elu'))
  model.add(Convolution2D(64, (3, 3), activation='elu'))
  model.add(Flatten())
  model.add(Dense(100, activation = 'elu'))
  model.add(Dense(50, activation = 'elu'))
  model.add(Dense(10, activation = 'elu'))
  model.add(Dense(1)
  model.compile(Adam(lr=0.0001),loss='mse')
  return model
```

3.3. Model evaluation

Figure 8 shows the error rate at each epoch when evaluating the built model. From the experimental results, the built model achieved the lowest error rate of 7% when using filters with the corresponding amounts of 64 and 32, the optimization function adam, learning rate is 0,0001. This low error rate indicates the efficiency of the CNN model. When testing self-driving robot, high working efficiency, very good path following, low error rate, has been observed.

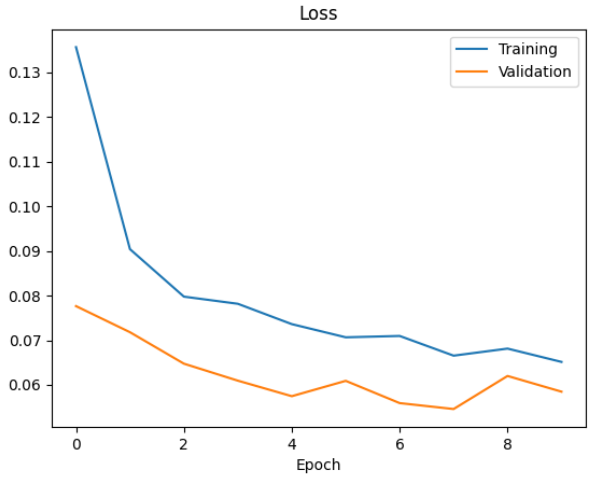


Fig. 8: *Loss comparison between validation and training data*

However, is has also been found that the time to train the model increased significantly as the number of filters in each convolutional layer increases. Specifically, when using filters with amounts of 32 and 16 in convolutional layers, the model will take about 51 seconds per epoch, while if the amount of filters is increased to 64 and 32, the

time for each epoch is about 68 seconds. Thus, to achieve high results, it requires a lot of time to train the model.

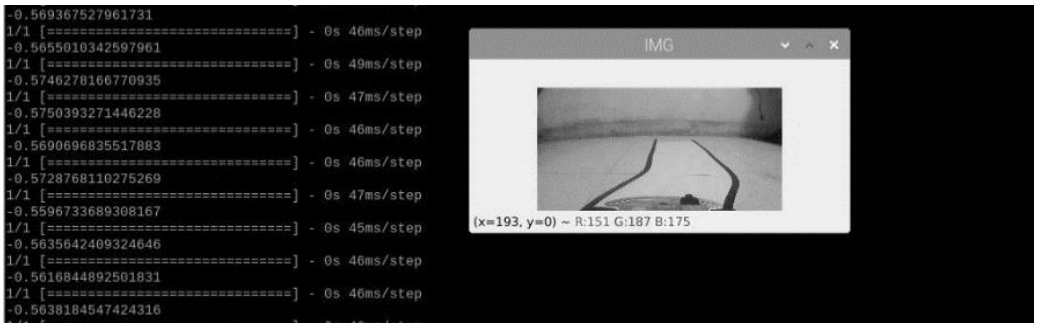


Fig. 9: *Testing self-dring robots*

4. Conclusions

In this paper, the problem of self-driving robots applying image processing and convolutional neural network has been solved. The obtained results show that self-driving robot with proposed CNN model performs with high accuracy and low error rate. With the initial success in solving the path recognition of self-driving robots, the issue can be expanded in a number of directions such as: improving the algorithm to achieve a more efficient recognition procedure, detecting, tracking and following objects.

REFERENCES

[1] Ha Thi Kim Duyen, Le Manh Long, Nguyen Duc Duy, Phan Sy Thuan, Nguyen Ngoc Hai, Nguyen Thi Tu Uyen, Ngo Manh Tien, "Research and develop self-driving car system applying artificial intelligence," *Vietnam Journal of Science and Technology*, Vol. 57 (5), 38-43, 2021.

[2] Đoan Hong Quang, Le Hong Minh, Thai Doan Nguyen, "Face recognition in video using convolutional neural network," *Vietnam Journal of Science and Technology*, Vol. 62 (1), 8-12, 2020.

[3] Aditya Kumar Jain, "Working model of Self-driving car using Convolutional Neural Network, Raspberry Pi and Arduino," *Proceedings of the 2nd International conference on Electronics, Communication and Aerospace Technology*, pp. 1630-1635, 2018.

[4] Henggang Cui, Vladan Radosavljevic, Fang-Chieh Chou, Tsung-Han Lin, Thi Nguyen, Tzu-Kuo Huang, Jeff Schneider and Nemanja Djuric, "Multimodal Trajectory Predictions for Autonomous Driving using Deep Convolutional Networks," *International Conference on Robotics and Automation (ICRA)*, pp. 2090-2096, 2019.

[5] Nitika Garg, Kanakagiri Sujay Ashrith, Gulab Sana Parveen, Kotha Greshwanth Sai, Anish Chintamaneni, Fatima Hasan, "Self-Driving Car to Drive Autonomously using Image Processing and Deep Learning," *International Journal of Research in Engineering, Science and Management*, Vol. 5, Issue 1, pp. 125-132, 2022.

[6] Sofiane Lagraa, Maxime Cailac, Sean Rivera, Fred eric Beck, Radu State, "Real-time attack detection on robot cameras: A self-driving car application," *Third IEEE International Conference on Robotic Computing*, pp.102-109, 2019.

[7] Shilpa Satre, Vinayak Bhat, Pranay Gadhave, Nikhil Jadhav, "Self-Driving Car based on Image Processing with Machine Learning," *International Research Journal of Engineering and Technology*, Vol. 8, Issue 5, pp. 1253-1255, 2021.

TÓM TẮT

ROBOT TỰ HÀNH ỨNG DỤNG XỬ LÝ ẢNH VÀ HỌC SÂU

Dương Đình Tú, Phan Xuân Hiếu, Hoàng Tuấn Hiệp

Trường Đại học Vinh

Ngày nhận bài 19/10/2022, ngày nhận đăng 23/11/2022

Trong lĩnh vực xử lý ảnh, nhận dạng là một trong những thách thức lớn đối với các nhà nghiên cứu trong những năm qua. Mục tiêu của nhận dạng là phát hiện, trích chọn các đặc trưng trong ảnh để phân loại các mẫu vào các lớp khác nhau. Một bài toán được quan tâm nhiều trong lĩnh vực này là robot tự hành ứng dụng xử lý hình ảnh. Trong bài báo này, phương pháp xử lý ảnh dựa trên một thuật toán học sâu để nhận dạng đường đi cho robot đã được trình bày. Mô hình robot tự hành được xây dựng dựa trên mạng nơ-rơn tích chập với việc sử dụng nhiều lớp khác nhau để trích chọn các đặc trưng tốt nhất trong ảnh. Các thực nghiệm đã được thực hiện, thu được các kết quả có tỷ lệ lỗi thấp.

Từ khóa: Robot tự hành; xử lý ảnh; mạng tích chập; học sâu.