Name: Nguyễn Tấn Phát
ID: 20146511

Self-Driving Car: Predicting Steering Angle

Abstract: The paper describes a project on Udacity to predict the steering angle of a self-driving car using deep learning. The project involves training a convolutional neural network (CNN) on a dataset of images and corresponding steering angles. The trained network is used to predict the steering angle from a live video stream captured by the car's camera. The aim of the project is to create a safe and accurate system to control the steering of a self-driving car. The project also includes techniques for data augmentation, model visualization, and fine-tuning of the network. The results demonstrate the effectiveness of the trained model in predicting the steering angle of the car in real-time with high accuracy and stability. This project showcases the potential of deep learning to advance the development of self-driving cars and other autonomous systems.

I. Introduction

Artificial Intelligence (AI) is currently changing the way we live and work. In this era, autonomous vehicles are one of the most advanced applications of artificial intelligence. Autonomous vehicles have the ability to automatically sense their surroundings and navigate through traffic and other obstacles safely and efficiently. AI technology is an important part of self-driving cars, helping to predict the steering angle of the car in the road ahead that the car receives through the camera.

In this paper, we focus on the application of artificial intelligence in self-driving cars, specifically the prediction of the vehicle's steering angle. Using an advanced CNN network model improved by increasing the number of convolutional layers, using ELU activation function instead of ReLU, and optimizing with Adam's algorithm, the new model achieves good results on data obtained from Udacity. This research can help improve the performance of automated vehicles and apply in real applications. In addition, artificial intelligence technology is also used to improve the navigation and navigation systems of autonomous vehicles. In this study, we used the self-driving car simulation software on Udacity to explain and illustrate the applications of artificial intelligence in autonomous vehicles.

II. CNN model

Convolutional Neural Network (CNN) is one of the important tools in the field of artificial intelligence and image processing. Developed in the 1990s, CNN has become an integral part of many applications such as image recognition, face recognition, signal classification and many more. CNN is developed based on the traditional artificial neural network, but with the important difference is the convolutional layers. These classes are designed to automatically find the features of the image, making the model capable of more accurate image classification. CNN is developed from previous studies on digital image processing, in which kernels (filters) are applied to extract features of an image. This idea has been applied to CNN, with kernels applied directly to the input image to create feature maps.

The first CNN network was introduced in 1998 by Yann LeCun, Léon Bottou and Yoshua Bengio. However, it was not until 2012 that CNN really became popular and widely used in image recognition applications when Alex Krizhevsky, Ilya Sutskever and Geoffrey Hinton used CNN to win the ImageNet Challenge with a high degree of accuracy. Accuracy up to 85%.

III. Data set

For the data I used the Training mode in the Udacity simulator, I manually steered the vehicle conveniently on the first track for three laps in the same direction and three more laps in reverse. The data collected during the training was saved in a csv file and contained the path to the image, in addition to the included data as well as steering wheel angle, accelerator pedal, forward gear and speed. The steering angles have been pre-scaled by a factor of 1/25 so that they range from -1 to 1. The data I have collected includes 4163 for the center, left and right shots because of the overall. all the photos i collected are 12489 photos. The image is 320 x 160, an example illustrated below is a left/middle/right image taken at a time when the car was running during training from a car.

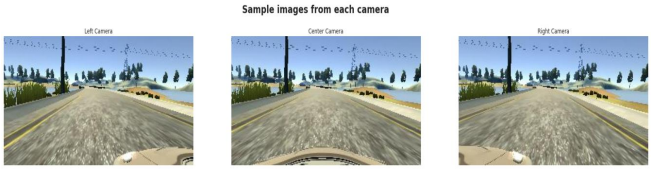

Figure 3.1 Left, right and center images from Car Camera

Data Augmentation

Enhancement helps to represent as much information from the image data as possible. I used four enhancement techniques to enhance the image that the model will see during training, with this tendency the Overfitting effect is reduced. The image enhancement technique used is the same as described below:

• Dimming: change the brightness to simulate day and night conditions.

• Left and right camera images: use right and left camera images to simulate the effect of steering angle when taking images from one side of the Camera and fine-tuning.

• Horizontal and vertical shift: shift the camera image horizontally to simulate the effect of different car positions on the road and change it vertically to simulate the image taken when driving up or downhill.

• Inversion in y-axis: Image from left to right is obtained as training data unevenly, inversion is very important for model contention.

Data Preprocessing

After image enhancement, preprocessing was applied to format the images before using them to train the model. The purpose of preprocessing is to improve the quality of the image so that it can be better analyzed by the computer.

The image preprocessing techniques used are described as follows:

• Crop: cropped the y-axis data and kept the pixel 40 to pixel 135 portion of each image to remove some of the front view of the vehicle and the sky above.

• RGB to YUV: the image is converted from RGB to YUV form because it has more advantages in light change and shape detection.

• Resize: to fit the model, all images have been resized to 66 x 200.

• Scale the image before processing: divide the pixel value of the image by 255, the pixel value after Scale is only between 0 and 1.

Figure 3.2 Initial image and image after processing

Build the CNN model

As mentioned earlier, a CNN model was used. The model structure used is inspired by the NVIDIA model used in its DAVE-2 system.

This model has the following characteristics:

• The input image is 66 x 200.

• It has a series of three kernel 5x5 convolutional layers, and the depth of each layer is 24, 36 and 48 respectively when entering the convolutional network.
• Then, there exist two consecutive kernels of size 3x3 for convolutional layers, with a depth of 64.
• The result is flattened as input for the Fully Connected phase.
• Finally, there is a series of Fully Connected connections, which are descending in size: 1164, 200, 50 and 10.
• There is also an additional dropout in 2 layers to avoid overfitting.
• It has 252, 219 trainable parameters.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 31, 98, 24)        1824

conv2d_1 (Conv2D)            (None, 14, 47, 36)        21636

conv2d_2 (Conv2D)            (None, 5, 22, 48)         43248

conv2d_3 (Conv2D)            (None, 3, 20, 64)         27712

conv2d_4 (Conv2D)            (None, 1, 18, 64)         36928

dropout (Dropout)            (None, 1, 18, 64)         0

flatten (Flatten)            (None, 1152)              0

dense (Dense)                (None, 100)               115300

dropout_1 (Dropout)          (None, 100)               0

dense_1 (Dense)              (None, 50)                5050

dense_2 (Dense)              (None, 10)                510

dense_3 (Dense)              (None, 1)                 11

=================================================================
Total params: 252,219
Trainable params: 252,219
Non-trainable params: 0
```

Figure 3.3 Shows the full summary of model implemented

The model uses a non-linear exponential linear unit (ELU) as the activation function. Also Adam optimizer was used, it's an extension for stochastic gradient descent, it uses squared gradients to scale learning, iterative update of network weights based on data train. The learning rate parameter used learning rate = 0.0001. A summary of the parameters used for the model is shown in the table below

Epoch	25
Steps per epoch	1000
Batch size	64
Optimizer	Adam(0.0001)
Activation Function	Elu

IV. Results

The initial data set is divided into training and testing sets, which are 80% for training and 20% for validation. Figure 10 shows the loss function
compare train and val.

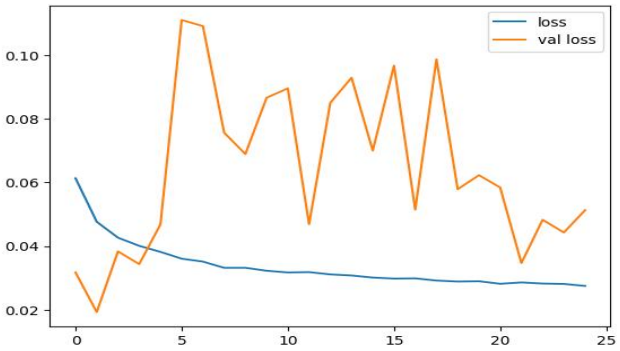

Figure 4.1. Training Loss vs Validation Loss.

As we can see in Figure 10, as the number of epochs increases, the loss value decreases. Although at the beginning there is a big gap between loss and val loss. However, at the last epochs, they were close to touching each other. The graph of the val loss function is relatively unstable because it shows many fluctuations. Video shows the journey of the car here: https://youtu.be/uOoJeFQ6JeM

V. Conclusions

Through the implementation it has been demonstrated that it is possible to generate a reliable predictive model of a vehicle's steering angle prediction system using a CNN network and a variety of data enhancement techniques.

After observing the results, I believe that the obtained model is a very good one, as it shows well on both maps. Although I think a better training can be done

parameter tuning and or enhancing the number of training images. That can be a model with the same or even better performance that can be achieved with less training time by adjusting the model parameters.

However, the model created with a relatively good accuracy for the car to steer, but in terms of speed, the car only runs correctly on the lane when the speed is still low. In the case of high speed, I think the return signal is too slow for the car to predict the steering angle, leading to the case of the car running astray and hitting the curb.

I plan to continue developing the application of steering angle prediction for autonomous vehicles in high-speed simulation, by using controllers to predict steering angle such as PID controller. In addition, in the future, I will build a real model for self-driving cars to predict the steering angle in the upcoming mechatronics project, in addition to lane recognition, the car will recognize signs and weather to control. steering angle and speed.

VI. References

Nguyễn Thanh Tuấn, Deep Learning cơ bản, access at the link on 10/5.

Link:

https://drive.google.com/file/u/1/d/1lNjzISABdoc7SRq8tg-xkCRRZRABPCKi/view

Theo udacity/self-driving-car-sim, access at the link on 30/4.

Link

https://github.com/udacity/self-driving-car-sim

Self Driving Simulation using NVIDIA's Model, access at the link on 10/5.

Link:

https://www.computervision.zone/courses/self-driving-simulation-using-nvidias-model/