End to End Learning for Self-driving Plane in GTA V

Chaoxing Huang
Australian National University
u6441859@anu.edu.au

Yinjiya Bai
Australian National University
u6007355@anu.edu.au

Abstract

The aim of this project is to train a convolutional neural network to automatically drive a plane in the game Grand Theft Auto V to avoid crashing under the condition of low altitude flight. A dataset is collected and processed and the supervised learning method is adapted to train the CNN to pilot the plane. We tune an architecture of a CNN in this project and it is verified that the CNN is able to drive a plane in GTA V .

1. Introduction

Low altitude flight plays an important role in humans' daily life in different aspects, such as agricultural seeding, aerial photographing, rescue and military mission. However, low altitude flying requires a pilot with an outstanding technique, since the accident caused by a high-speed low height flying vehicle is fatal**[8]**. Therefore, it is worth looking into the autonomous driving of a plane system at a low altitude.

The first end-to-end based self-driving technique is proposed in 1989 by Pomerleau**[1]**. End-to-end learning has been widely utilised in the problem of self-driving since the booming of deep convolutional network. Nvidia proposed an end-to-end CNN architecture to predict the steering angle of a car and the road test turned out to be successful**[2]**. A vehicle self-tracking system is proposed in **[3]** that a DNN is trained to control the vehicle to autonomously track the vehicle in the front.In **[4]**, an end-to-end self-driving model is implemented by using large scale video dataset and FCN+LSTM architecture. Recently, **[6]** proposed a method of using neural networks to figure out the driving manoeuvre by integrating the information obtained by the 360 degree camera. Apart from end-to-end self-driving, deep reinforcement learning method is also looked into by the self -driving community**[5]**. Inspired by the idea of end-to-end learning in self-driving car, this project implements an end-to-end learning model on driving a plane to avoid crashing in the GTA V.The rest of the report is organized by the following parts: Part 2 introduces the system overview and the dataset as well as the methodology of training an end-to-end driving CNN. Part 3 displays and analysis the results. Part 4 is the further discussion and Part 5 is the conclusion.

2. System overview and methodology

2.1. Overview of the system

The game Grand Theft Auto Five (GTA V) is an action-adventured game where players can drive varies of vehicles(car,plane,boat,etc) to explore different landscapes from city to the wild. GTA V has been used as simulation environment in some of the works of self-driving due to its high similarity to the actual world. For the plane driving problem in GTA V, it is actually somehow similar to the actual operation of an aircraft with 3 degrees of freedom(pitch, yaw and roll) operated by eight keys on the keyboard or joystick, namely throttle plus, throttle minus, left turn, right turn, left roll, right roll, pitch dive and pitch-pull up. Since the throttle value of the plane can be fixed in the game and the turning can be roughly replaced by a *"roll + pull up "* manipulation, the manoeuvre of the plane in this project is limited to 2 degrees of freedom, with two keys controlling the roll and two keys controlling the pitch. Thus, there are five possible manoeuvres for the plane, which are left roll,right roll,pitch dive, pitch-pull up and release all the keys.In this project, we do not consider the flight route or the dynamic control of the plane and the final object is that the plane is able to maintain itself in the sky without crashing onto the ground or other obstacles. There are different view angles during the driving of vehicle in GTA V, we choose the first-person perspective and feed it into the CNN,since the first-person perspective is similar to the viewing angle in reality. The scene is shown in **Figure 1**.

2.2. Methodology

As the possible control manoeuvres are five discrete key-presses, the problem is then defined as a classification problem. Firstly, we manually drive the plane in GTA V and collect the data images as well as label them with the corresponding key press.Then we feed the data into a CNN to train the model. After the tunning of the model, we then freeze the network and deploy it to drive the plane autonomously by forward pass the images from the stream of the game to the network.

Figure 1. The first person view from the cockpit in GTA V

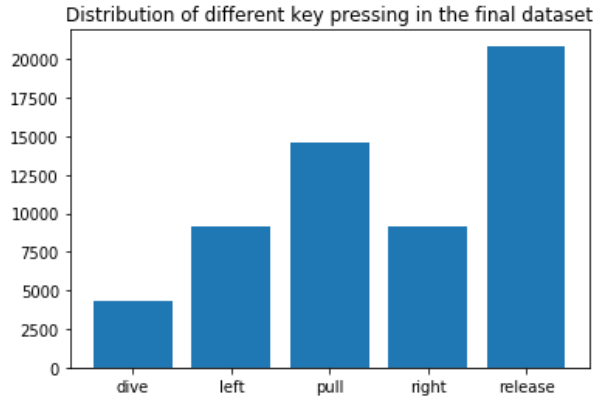

Figure 2. Data distribution of the final dataset

2.2.1 Dataset

The dataset consists of labelled images which are collected from the game when we manually play the game. When collecting the data, we arranged the window size of the game as 800×600 and first took off the plane from the ground, which is the stage not being included in the dataset. Then we manually adjusted the plane at an altitude range from 0 to 1000 meters and use OpenCV to grab the screen-shot of the game window.Every time an image is grabbed, the keypress which is corresponding to this frame is recorded by using Python's win32api package. The data was mainly collected in the sky above the city and some periphery areas with mountains and seashores which are nearby the urban area. The dataset includes the situation of day time and night time flight with different weather conditions(sunny, cloudy and foggy,tiny rain,etc).Originally, we got 70275 labelled images after 3 times of collections. In every collection, we manually piloted the plane without interruption for more than 30 minutes to ensure continuity and the data was collected by two different persons to avoid subjective tendentious.

Data pre-processing One problems of the initial dataset is the data unbalance.For each of the keypress set, there are 42152 for release,10999 for pull up, 6881 for left,6680 for right and 3563 for dive.

To address the problem, we add another 10000 new collected images and deleted some of the dominant data to balance the dataset. The distribution of the balanced data is shown in **Figure 2**.We also used random flipping and rotation to do the data augmentation.

It should be pointed out that we still maintain the unbalance of the data to a certain extent, because this is not a pure static classification problem and the network is required to somehow imitate the piloting behaviour of human, and the manipulations of human itself is unbalanced.

2.2.2 Network architecture and transfer learning

Inspired by Nvidia's network architecture, we use a network architecture(GTA-Net) with five convolutional blocks as feature extractor and 2 fully connected layers as the output "controller".By empirical tunning, we adjust the number and size of the filters and add maxpooling as well as BN to the network, which is different from the Nvidia's self-driving architecture. The network architecture is shown in **Table1**.The activation function in the network is ReLu.

Besides using our GTA-Net, considering the amount of some of the keypress data in the dataset is comparatively small and collecting for more data is time consuming, we also apply transfer learning method to train an AlexNet, in which we frozen the CNN as feature extractors and purely trained the FC layers as the classifier.

2.2.3 Loss function and Performance Metrics

For a classification problem, the loss function we adapt is the cross entropy loss. When we deploy the network to the test driving, the performance metrics is no longer the classification accuracy, since the environment that the plane encounter is non-static and the ground truths are also unknown to human. We use a metric called autonomy**[2]**, which is the percentage of the manipulations that the network could autonomously pilot the plane in the sky without human intervention within a range of given time. In a fixed flight time, the human would intervent the plane when the plane fails to pilot itself and about to crash.The mathematics expression of autonomy is given below:

$$autonomy = 1 - \frac{number\ \ of\ \ human\ \ interventions}{total\ \ manipulations}$$

3. Experiment and Result

3.1. Setup

We separated out 80% of the dataset as training data and the rest 20% as validation data. The implementation environment was Pytorch 0.4.0 and the network was trained on a Nvidia GTX 1070 GPU. For the GTA-Net,The images were resized to 100×100 and fed into the network as RGB images.The learning rate was 1e-4 and a mini-batch method

Table 1. Network architecure

conv in=3, $32 \times 5 \times 5$,stride=1,padding=2
ReLu
BN
conv $64 \times 5 \times 5$ stride=1,padding=2
ReLu
BN
maxpooling 2×2, stride=2
conv $128 \times 3 \times 3$ stride=1,padding=1
ReLu
BN
maxpooling 2×2, stride=2
conv $256 \times 3 \times 3$ stride=1,padding=1
ReLu
BN
maxpooling 2×2, stride=1
conv $256 \times 3 \times 3$ stride=1,padding=1
ReLu
BN
maxpooling 2×2, stride=1
FC Layer
Flatten
ReLu
512
ReLu
256
ReLu
5

with batch size of 16 was used. We used Adam**[7]** as the optimizer, with $\beta_1 = 0.9$,$\beta_2 = 0.99$ and $\epsilon = 1e-8$.The network was trained for totally 20 epochs. The training curves are shown in **Figure 3 and 4**. For training the AlexNet, the images were resized to 224×224 and the learning rate was set to be 1e-4 with a decay rate of 0.99 for every two epochs.We eliminated the dropout in the FC layers by empirical tunning and add a global weight decay as 1e-4.

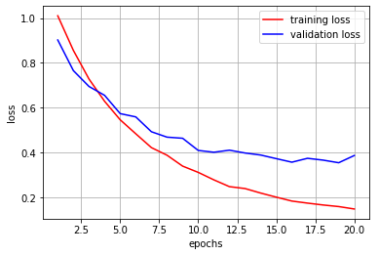

Figure 3. loss curve vs epochs

3.2. Model test

In the test stage, the trained network was frozen and the data images were grabbed from the streamed game window. The outputs were transferred into virtual keyboard

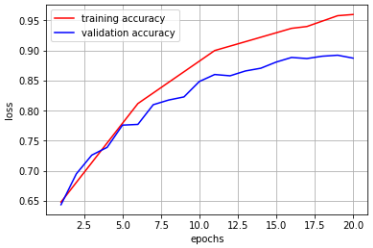

Figure 4. accuracy curve vs epochs

command(I,J,K,L) to control the aircraft. Another four keys(W,A,S,D) were set for human intervention.In every time of the test flight, the human player was required to sit in front the screen and only save the plane when the auto pilot was about to fail. The plane was firstly manually taken off and the player randomly operated the plane to a certain altitude within the range of 0 to 1000 meters, where the player boots the test program. The model was tested in five different conditions, namely *city sunny*,*city rainy*,*city foggy and cloudy*,*city clear night*,*city thunder storm* and *mountain sunny*. Every condition was tested sequentially for fives time and we calculated the mean of the autonomy. The given test duration of each time was set to be 120s.The autonomy results are shown in **Table 2**.In the condition of *city thunder storm*,even human found it difficult to handle the plane and prevent it from crashing and it turned out that the networks failed to pilot the aircraft. A short demo video of how the GTA-Net drive the plane is here:
`http:/https://www.youtube.com/watch?v=_yy678iEGRs&t=10s`
It can be seen that the GTA-Net is able to pilot the plane with high autonomy in landscape of the city, especially in good weather conditions.When the landscape is changed to the mountainous area, the driving performance become worse even with good weather condition. The GTA-Net also outperforms the AlexNet in most of the flights in the city landscape, especially in the bad weather conditions. Surprisingly, AlexNet outperforms the GTA-Net in the mountainous area with sunny weather.

Table 2. Autonomy results(%)

condition	GTA-Net	AlexNet
city sunny	99.96	97.426
city clear night	99.04	93.49
city rainy	98.43	75.23
city foggy and cloudy	98.86	82.36
mountain sunny	94.98	97.06
city thunder storm	NA	NA

3.3. Features visualization

3.3.1 Features from GTA-Net

To have a intuitive understanding of what the network has learned in the automatic piloting task. We open the GTA-Net and see some of the features being learned by the filters and compare it with our intuition.Here, we mainly present some of the filters' result from the third and fifth layer.The features of two example scenes are shown below.

Figure 5. The scene that the plane comes across a mountain

Figure 6. The night CBD scene

In the third conv-layer,the network mainly extract the overall background of the scene that the plane is facing with(the sky, the city landscape with bright lights, the main body part of the mountain).In the fifth conv-layer, the features that the GTA-Net learned are mainly some significant object regions, such as the highlight skyscrapers in the CBD and the large rock areas stretch from the mountain, which are also essential elements for human pilots to avoid crashing.

3.3.2 What the AlexNet extracted ?

Since the transfer learning model performs worse than the GTA-Net in most of the cases, we have a glimpse of what the pre-trained CNN can actually extract.The second conv block's results of the night CBD scene is shown here.

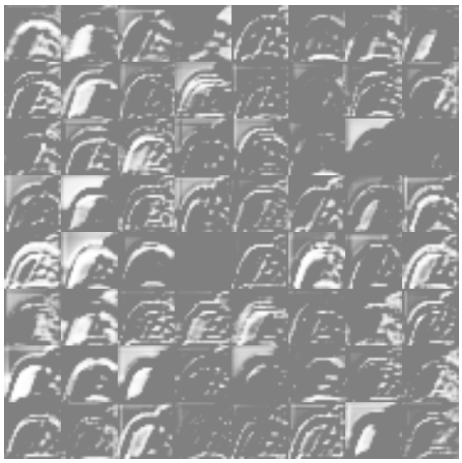

Figure 7. Features being learned by the filters of conv3 from the mountain scene

Figure 8. 3 selected filters of conv3 from the CBD scene

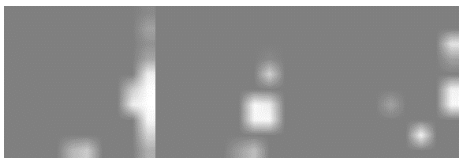

Figure 9. 3 selected filters of conv5 from the mountain scene

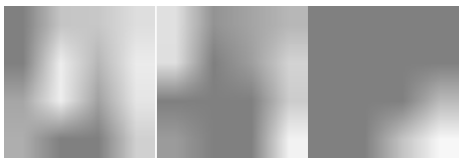

Figure 10. 3 selected filters of AlexNet's second conv-layer of the night CBD scene

We disappointedly find out that the features extracted by the second conv-layer are intuitively unclear,which may have a negative effect on the output keypress decision from the network.

4. Discussion

The GTA-Net outperforms AlexNet in most of the cases and it may be resulted by the poor feature extraction in the AlexNet model. Since the pre-trained dataset is Imagenet, and it might be different from our own dataset, the pre-trained feature extractor should be fine-tuned or even the whole model requires being trained from scratch.
The project also reveals some drawbacks and limitations.After a painful data collection, the dataset we col-

lected in the game still cannot cover all the possible situations of terrain and weathers,which downgrade the performance of the driving in complex terrains and weathers in the test. Using a static CNN to classify the keypress is also sensitive to unbalanced data, while collecting more data and maintaining the data to a comparatively balanced level is either time consuming or suffered from the unsimilarity to the human's actual behaviour. Thus, using LSTM network to help the model infer the action based on time series or applying reinforcement learning method to learn the driving skill interactively with the environment may be the future works.

5. Conclusion

In this project, we explored the problem of using a CNN to autonomously drive a plane in GTA-V based on end-to-end supervised learning.We collected our own dataset by manually piloting the plane,which contains the scene in the flight as well as the corresponding manoeuvres.We proposed and tunned our own GTA-Net, and it is verified that the network is able to drive the plane with high autonomy. A transfer learning model's performance is also compared. We visualize some of the features extracted by the networks to have a intuitive understanding of the networks' performance. Finally, we discussed some of the limitation of our supervised learning method and future works on collecting more comprehensive dataset, fine-tunning and even reinforcement learning are required.

Appendix

Authors' contributions:
Chaoxing's work: 60%
1 Implementing the program for collecting data from the game and the program for CNN to play the game
2 Implementing the network and tune the architecture.
3 Feature visualisation.
4 Drafting the whole report

Baiyinjiya's work: 40%
1 Collecting most of the training data
2 Implementing the data balance
3 Tune the parameters and train the network.Give Chaoxing suggestion on what he observed from the outcome.
4 Modifying the report.

References

[1] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," in Advances in neural information processing systems, 1989, pp. 305-313.

[2] M. Bojarski et al., "End to end learning for self-driving cars," arXiv preprint arXiv:1604.07316, 2016.

[3] J. M. Pierre, "End-to-End Deep Learning for Robotic Following," in Proceedings of the 2018 2nd International Conference on Mechatronics Systems and Control Engineering, 2018, pp. 77-85: ACM.

[4] H. Xu, Y. Gao, F. Yu, and T. Darrell, "End-to-end learning of driving models from large-scale video datasets," arXiv preprint, 2017.

[5] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani, "Deep reinforcement learning framework for autonomous driving," Electronic Imaging, vol. 2017, no. 19, pp. 70-76, 2017.

[6] S. Hecker, D. Dai, and L. Van Gool, "End-to-end learning of driving models with surround-view cameras and route planners," in European Conference on Computer Vision (ECCV), 2018.

[7] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.

[8] S. Scherer, S. Singh, L. Chamberlain, and S. Saripalli, "Flying fast and low among obstacles," in Robotics and Automation, 2007 IEEE International Conference on, 2007, pp. 2023-2029: IEEE.