Vision based steering behavior imitation for self driving car

Chaoxing Huang(U6441859) **Zhaowen Xu(U6749372)** **Xinyu Gao(U6656678)**
Chensheng Zhang (U6615215) **Ruitao Leng(U6777647)**
College of Engineering & Computer Science, ANU
Group 27

Abstract

In this project, we look into the problem of utilizing RGB and temporal information in end to end steering behavior imitation for self driving car. We propose a two-stream CNN model which takes the RGB information and the optical-flow or the dynamic image as input to predict the steering angle. By comparing with the single frame based end to end steering prediction method, we find out that our two-stream model can significantly improve the steering angle prediction accuracy. We also deploy our model on the video being collected on the campus of the Australian National University and verify the validness of the model in an intuitive way.

1 Introduction

Steering angle prediction(or control) is an indispensable part in the self-driving system. Prior to the deep learning era, the steering angle is generally obtained from the image features produced by handcraft algorithms(lane segmentation, lane slope calculations,etc). With its ability of learning deep hierarchical representations and features with high non-linearity[6], deep convolutional neural network (CNN) enables the system to learn the steering behavior based on the raw pixels obtained by the vision system, which is also known as end to end driving. The method of establishing a mapping relations between steering angle and a single frame by using CNN has gained a remarkable success[3]. However, in reality, human driver does not always making steering decision by purely using information from one single frame. Rather, the steering behaviours contains some temporal dependencies. The problem is then defined as: How can the CNN learn the steering behavior by utilizing temporal information from the video stream. In this project, we propose a two-stream framework that takes both RGB information as well as temporal information as input to produce the steering output. We include the optical flow and the dynamic image in the temporal information. From our knowledge, the dynamic image has never been used in self-driving, and we verify its feasibility in this project. We also implement the single-frame end to end case for comparison and verify its ability in self-driving.The final demo video can be found here: `https://youtu.be/7juEYI-gGKw`
The rests of this report are as follows: Section 2 provides a short warp-up of the related work. Section 3 describe the methodology including model architectures. Section 4 and Section 5 introduces the dataset and the experiment. Section 6 is the discussion and Section 7 concludes the report.

2 Related work

Pomerleau et.al(1989) used Autonomous Land Vehicle in a Neural Network (ALVINN) to predict steering command from simulated road images' pixel inputs in 1989[4]. Although the structure was relatively naive according to current standard, the pioneering work demonstrated that neural networks could do the steering angle prediction from pixel values.In 2016, NVIDIA designed and developed DAVE2 system, this is the first time using a deep CNN to directly steer the car from

raw images[3].The training data is collected from single images sampled from video. The work demonstrates that CNN can directly learn the steering behaviour from images without road marking detection, semantic abstraction, path planning.

Recently, Joy Yue-Hei et.al used LSTM(Long short-term memory) to model the video as ordered sequence of frame and process video classification[7]. Eraqi et.al proposed and benchmarked a C-LSTM (Convolutional Long short-term memory) architecture that allows learning both visual and temporal dependencies of driving and achieved more stable steering by 87%[8]. Shuyang et.al developed steering angle prediction methodology from image recognition in 2017. Their approach uses 3D convolutional LSTM model to predict steering angles[9].

The concept of dynamic image has been introduced in the field of computer vision since 2016[10].It can provide the historical information via the format of a standard 2d image. By using CNN, it shows the state-of-art performance in action recognition[10], and also it is possible to make prediction on human action by only a few frames of a video containing an action by utilising dynamic image[11]. To our own knowledge, dynamic image for steering prediction in the field of autonomous driving has not been widely used in research. In the following methodology section, more detailed will be provided on how we use dynamic image in our model.

Optical flow was proposed in the area of motion detection in 1981[18]. Before the era of deep learning, it was already used in navigation and autopilot of unmanned drones and vehicles. In 2000, K. Nagatani et.al combined odometry and optical flow to perform position estimation for autonomous vehicle[19]. F. Kendoul et. al proposed a prediction based algorithm with adaptive patch for optical flow computation, addressing the problem of unmanned aerial vehicle's autonomous localization[20]. In recent years, optical flow is combined with CNN to perform robust target association[21]. Being computationally effective, optical flow can represent a video sequence as the input of CNN. In this case, it is considered as an alternative of dynamic image[22].

Two-stream method was first introduced in action recognition and prediction[5].With its ability of synthesizing spatial and temporal information, it demonstrates outstanding performance in many temporal based tasks like video classification and sound recognition. Two-stream model was first utilized in steering angle prediction in [1] due to its robustness and low training cost compared with RNN, LSTM and 3D CNN. In this work, the streams only takes the current frame of RGB and optical flow as input.

3 Methodology

3.1 Mathematical notations

We first introduce the notations that will be used in the following context.
Frame at time t : I_t.
Optical-flow at time t : $F_t = f(\{I_{t-1}, I_t\})$.
Dynamic image at time t that includes the past n frames : $D_t = d(\{I_{t-n}, ..., I_t\})$.
steering angle: θ.

3.2 Loss function and performance metrics

We use a supervised learning method to train the CNN to imitate human's steering behaviour by using data with steering commands(labels).The object is to enforce the predicted angle as close as possible to the ground truth steering angle. Thus, we use the mean square error (MSE) as the loss function:

$$L = \frac{1}{N}\sum_{i=1}^{N}(\theta_i - \bar{\theta_i})^2 \tag{1}$$

where N is the batch size, θ and $\bar{\theta_i}$ are the predicted angle and ground-truth respectively.We use the root mean square error to measure the performance of the model over the whole test dataset.

3.3 Dynamic image and Optical-flow

3.3.1 Dynamic image

The dynamic image is an RGB image that summarizes the appearance and dynamical information of a given video sequence[10] The dynamic image is obtained by optimizing a rank function of a series of frames.The frames which are closer to the last frame are supposed to have higher rank. The dynamic image at time t that contains the information of the last n frames can be obtained by the following equations:

$$D^* = d(I_{t-n}, ..., I_t) = argmin_D E(D) \tag{2}$$

$$E(D) = Reg + \frac{2}{n(n-1)} \times \sum_{q>i} max(0, 1 - S(q|D) + S(i|D)) \tag{3}$$

where Reg is the regularization term. i and q are the frame id and S is the ranking score. Details can be referred to this work[10].

3.3.2 Optical flow

Optical flow is a typical temporal representation in video streams, which captures the motion changes between two frames. The optical flow being used in our work is based on the implementation of the Farneback's method in OpenCV[12]. For the optical flow at time t, we can compute the optical flow value in both vertical and horizontal direction, which can be denoted as $F_t^x = f^x(I_{t-1}, I_t)$ and $F_t^y = f^y(I_{t-1}, I_t)$. From the magnitudes in both direction, the overall magnitude and the direction can then be obtained. The optical flow will then be encoded into HSV space as an RGB image. The direction is represented by Hue and the magnitude is represented by V[1].

3.4 Network architectures

We adapt two baseline model in our work, the single stream model and the two-stream model. To avoid over-fitting and for the purpose of simplicity and demonstration of the effectiveness of temporal information, the models are relatively shallow.

3.4.1 Single stream model

Inspired by Nvidia's work[3], we implement the CNN architecture shown in Figure 1. The CNN has two convolutional blocks and one max-pooling layer. Dropout is applied to the model to avoid over-fitting. This model has two variance sub-model using different input. In the single frame-single stream case, the input is purely a 3-channel RGB frame obtained by the car camera. We also implement a single stream model that takes a short sequence as input. In this case, for the frame at time t that the steering angle needs to be predicted, the past two frames I_{t-1} and I_{t-2} are included to and they are stacked with the current frame as a 9-channel tensor, which is denoted as $\{I_{t-2}, I_{t-1}, I_t\}$.

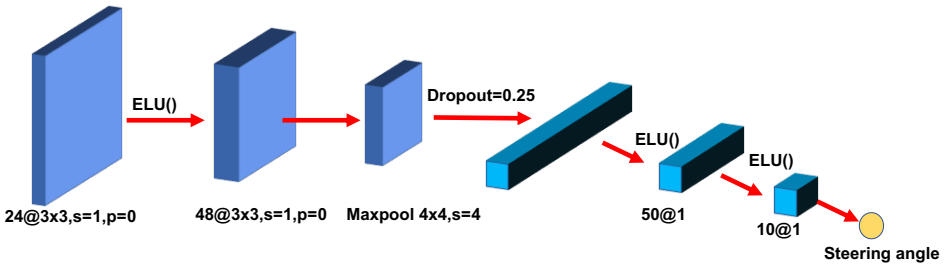

Figure 1: The CNN architecture of the single stream model.

3.4.2 Two-stream model

Two-stream CNN models are generated from the hypothesis that human brain uses two separate streams to perform object and motion recognition [17]. The two-stream model that we used was

inspired by [5]. The original work built a deep convolutional network architecture for video action recognition by combining two different streams, spatial stream and temporal stream. Our two-stream model architecture is shown in Figure 2. Each stream has two convolutional blocks and a max-pooling layer. After dropout, the two streams are then concatenated to go through fully-connected (FC) layers.

For the spatial stream, our model uses 3 frames of RGB images as input, which can be denoted as $\{I_{t-2}, I_{t-1}, I_t\}$. For the temporal stream, we use two different variations, optical flow and dynamic image. The optical flow is either single frame $F_t = f(\{I_{t-1}, I_t\})$, or two consecutive frames $F_{t-1} = f(\{I_{t-2}, I_{t-1}\})$ and $F_t = f(\{I_{t-1}, I_t\})$. The dynamic image is either generated from 3 frames $D_t = d(\{I_{t-2}, I_{t-1}, I_t\})$, or 5 frames $D_t = d(\{I_{t-4}, I_{t-3}, I_{t-2}, I_{t-1}, I_t\})$.

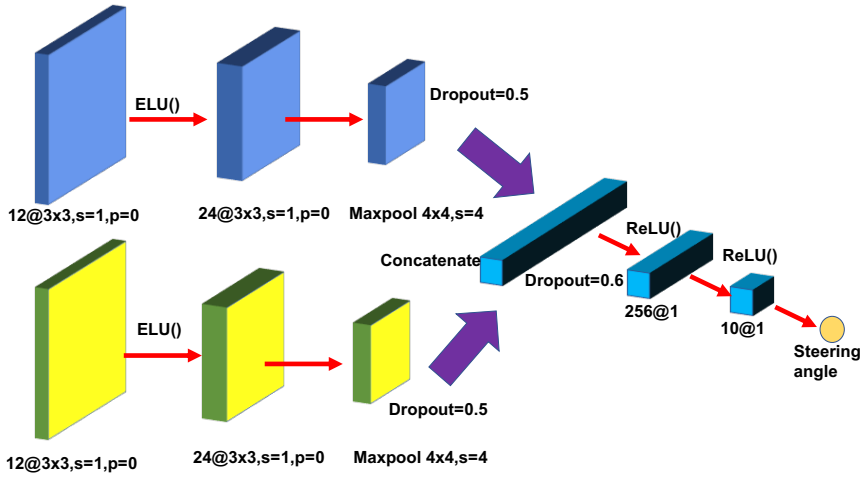

Figure 2: The CNN architecture of the two streams model.

4 Dataset

In terms of the datasets, we use a public shared dataset created by USC researcher Sully Chen[13] and a self made dataset from the Udacity simulator[14].

4.1 Sully Chen's Dataset

There are two subsets in this dataset: 2016 dataset and 2018 dataset. Both of them are made up with a number of image files and time-stamped car steering angle logs. The image data was recorded by a camera mounted on the 2012 Honda Civic front windshield at 20 frames per second. The steering angle data is accessed by a Arduino shield connected to the OBD-II port on car which can access and decode the CAN(Controller Area Network)-BUS data.

4.1.1 2016 dataset

2016 dataset contains 45567 frames and corresponding steering angle labels. The dataset records a trajectory of approximately 4km around the Rolling Hills in LA, USA. The data distribution is shown in Figure 3.

4.1.2 2018 dataset

2018 dataset contains 63825 frames and corresponding steering angle label as well as time stamps. This dataset records a trajectory of approximately 6km along the Palos Verdes Dr in LA, USA. The data distribution is shown in Figure 3.

4.2 Udacity simulator

The Udacity simulator is an open-source self-driving simulator based on Unity. It provides simple 3D model environments for self-driving, including water-side lane,jungles and mountains. The lane in

jungle and mountain environment is the hardest one in the simulator, which has sharp turnings,steep slopes and cliff overpass. The simulation is divided into training mode and autonomous mode in the simulator. The training mode is for data collection and the autonomous mode is for model deployment(test). The car model in the simulator is equipped with 3 cameras(left,middle and right) and it will label this image with steering command in real time. We first manually drive the car in the jungle-mountain lane in both clockwise and counter-clockwise direction and the total image amount is 9030. The data distribution is shown in Figure 3, and the steering commands are all normalized by the simulator.

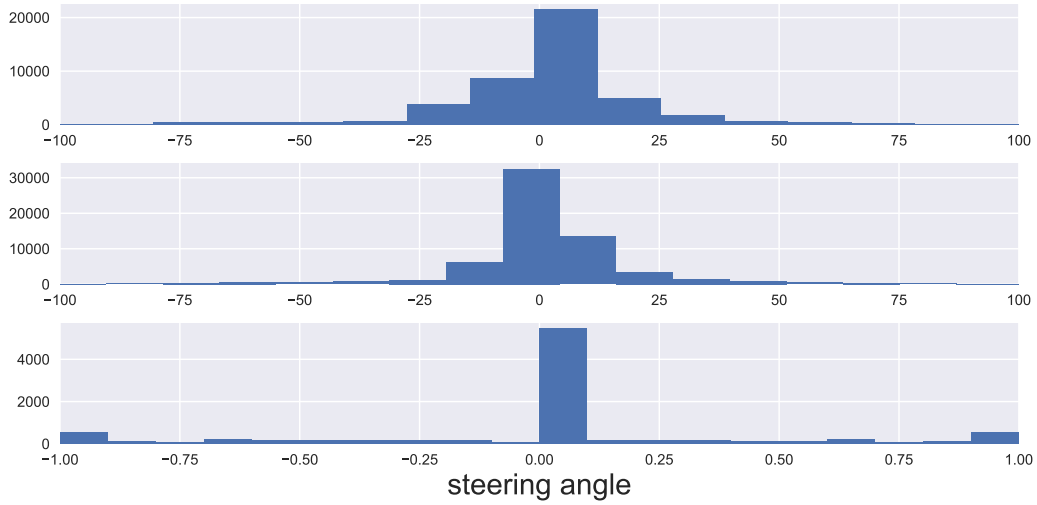

Figure 3: Top: Data distribution of 2016 dataset. Middle: Data distribution of 2018 dataset. Down: Data distribution of Udacity simulator

5 Experiments and results

5.1 Data pre-processing

We choose Sully Chen's 2016 dataset as training set. In the 2016 dataset, 80 percent of the dataset is splited out as training data and the rest 20 percent are the validation data. For the test set, We choose two groups of data from the 2018 dataset as two test sub-dataset: the 0 to 20000$^{\text{th}}$ frame as the "city test" and the 27500$^{\text{th}}$ to 40000$^{\text{th}}$ as the "hill test".The city subset mainly contains driving scene in normal city road while the hill subset conatins conditions of driving in mountainous area.The reason why we do so is because the 2018 dataset contains a considerable amount of video segment of driving in parking lot and turning around with obvious pre-planning(e.g., 180 degree turing)

For the dynamic images, the dynamic image corresponds to the current frame is computed using the current frame and the consecutive frames before it. For example, a three-frame dynamic image of the frame I_{100} is computed by a serious of frames, which is $\{I_{98}, I_{99}, I_{100}\}$.

We use the OpenCV's inbuild Farneback method to compute the optical flow. The 2 channel resulst is then encoded into HSV space to form a single 3 channels image. Direction of the optical flow is encoded by Hue and the magnitute is encoded by the V value. The sample of the optical flow and dynamic image of the dataset are shown in Figure 4.

Before feeding the images into the model, we resize the images as 80×320 to expand the road and shrink the background. Random brightness jitter is applied as data augmentation strategy to mimic the sunlight and shadow changing on the road during driving. The brightness jitter ratio range is set to be $\pm 5\%$ .All the image are normalized to $[-1, 1]$ and the angles are normalized to $[-\pi, \pi]$ for a better convergence during training.

For the simulation in Udacity simulator, we use 90% of the data for training and the rest 10% for testing. Since the winding roads snakes between cliffs, the car may "see" information that are very

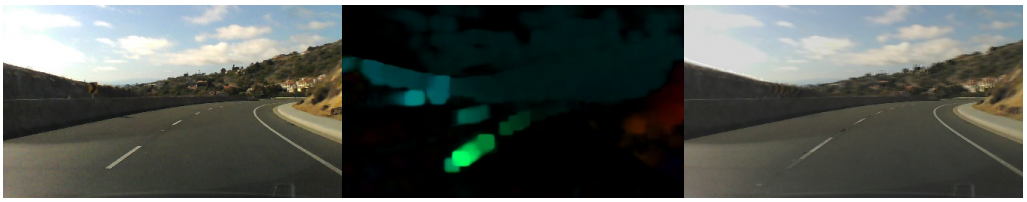

Figure 4: Left:Original frame. Middle: Optical flow. Right: Dynamic image using 3 frames.

far away, which is not relevant to the current steering control and it may confuse the model.Thus, we crop the image to eliminate those "far information". An example is shown in Figure 5 . Random image flipping is used as data augmentation technique and normalisation is applied. For the image from left and right camera, the angle value is empirically reduced or added by 0.2.

Figure 5: Left:Original frame. Right: Cropped frame.

5.2 Experiments

5.2.1 Implementation details and results

Mainly, 6 variance models that based on single stream and two streams framework are compared in the experiment. They are: Single stream with single frame input, Single stream with 3 consecutive frames input, Two streams(3 frames + one frame of optical flow), Two streams(3 frames + two frames of optical flow), Two streams(3 frames+dynamic image of 3 frames),Two streams(3 frames + dynamic image of 5 frames). Note that in the one frame of optical flow case, it is the optical flow of the current RGB frame and the previous one RGB frame. The experiment is implemented with Pytorch 1.0.0, on a machine with an NVIDIA RTX-2070 GPU and i5 9600 CPU. The batch size is 16 and the learning rate is 1e-4. The optimizer here is Adam[15], with $\beta_1 = 0.6$,$\beta_2 = 0.99$,and $\epsilon = 1e - 8$.Regularization is utilised to suppress overfitting with a weight decay factor of 1e-4. Manual learning rate decay is used when it is observed from the Tensorboard that the loss becomes plateus . Every training lasts for about 12 hours with approximately 100 epochs. The models are tested on the city testset and the hill testset, the RMSE results are shown in Table 1 and two steering angle trajectories are shown in Figure 6&7 to provide an intuition .

Table 1: RMSE of different models(in degree)

Model	City testset	Hill testset
single stream single RGB	19.04	23.89
single stream 3× RGB	7.23	13.91
3×RGB +1× optical flow	7.10	10.92
3×RGB +2× optical flow	6.09	9.73
3×RGB +dynamic image of 3 frames	6.93	10.15
3×RGB +dynamic image of 5 frames	5.99	11.51

From the results above, it can be seen that when models taking the temporal information into account(consecutive frame, dynamic image, optical flow), the performance can be significantly improved. In the city testset, it can be seen that the model with 2 optical flow input outperform the model with 1 optical flow input, and the model with dynamic image of 5 frames outperform that of 3 frames. The more temporal information is acquired, the higher the performance.Also, from the result in both testset, model with 2 optical flows input outperform the model with dynamic images of 3

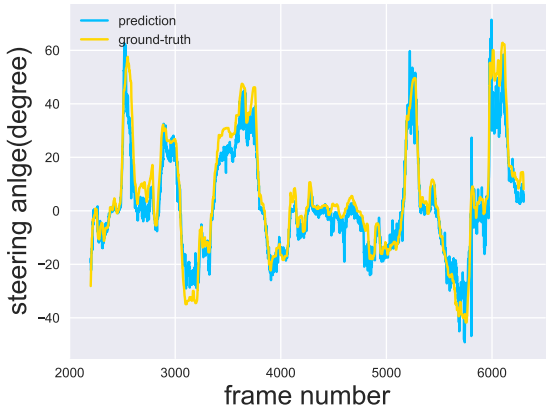

Figure 6: Part of the Steering angle trajectory of the Hill testset using 3×RGB +2× optical flow

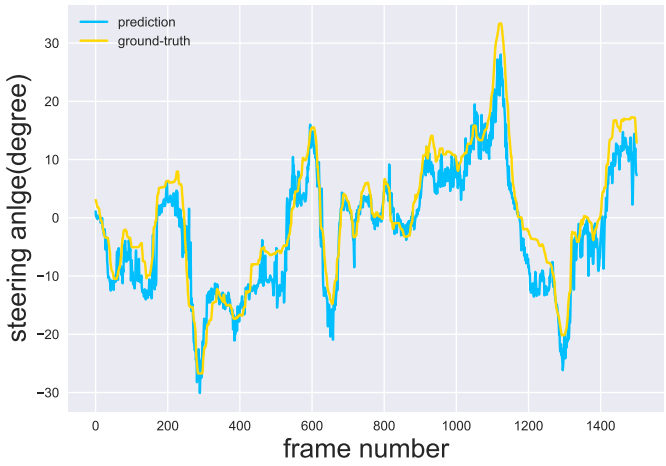

Figure 7: Part of the Steering angle trajectory of the City testset using 3×RGB +dynamic image of 5 frames

frames. This reveals under the condition that the amount of temporal information are the same(both 3 frames of information), optical flows can achieve a better result. In the hill testset, most of the experiment phenomenon still holds, but the results of the model with dynamic image of 5 frames does not achieve the best result among all. This will be discussed in the latter context.The steering ratio of cars ranges from 1:12 to 1:20. For the worst case(1:12),our best two-stream model can achieve a mean wheel angle error of less than 0.67 degree.

5.2.2 ANU campus test

Once we finish training and testing our model on the dataset, we deploy our model on the video that we collect from the campus of the Australian National University, Canberra. The video was taken on the Daley Road of the campus, and it contains the "hook turn" which are not included the training dataset. The route and the sample from the video are shown in Figure 8 . We choose the two streams(3×RGB +2× optical flow) model to test on the video and it turns out that the model being trained on the dataset in the U.S. is able to handle the steering behavior on the Australian road. The steering results is shown in the demo video link at the start of this report.

5.2.3 Simulation

We train the single stream single frame model on the collected data from the Udacity simulator. Since the physical environment in the simulator is fairly simple, it only takes 2 hours to train the model. We then switch on the autonomous driving mode of the simulator and deploy our network. It turns out at

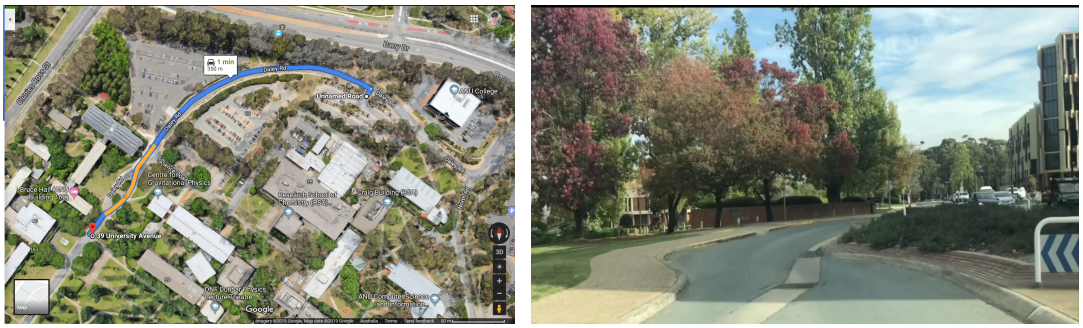

Figure 8: Left:The route in the video Right: The "hook turn" in the video

most of the time, the CNN is able to safely drive the car on the road.The results can be seen in this video: `https://www.youtube.com/watch?v=nHDanjkQyhQ&t=6s`.

5.2.4 Visualisation

Self-driving problem is highly linked with the safety. Thus, how the CNN "see" the world should be explainable to human beings.To tackle this problem, we use the VisualBackProp(VBP) method proposed by NVIDIA[16]. The procedures are:(1) The activation maps of each conv-layers are averaged and results in an average-map for every layers. (2) The deepest average map is up-scaled to the size of the average map of the previous layer by de-convolutions.(3) Pointwise timing the up-scale mask to the average map to get a new map.(4) Use the new map and repeat step (2) and (3) until a mask with a size of the input image is got. Finally, lay the final mask to the input image to see the result. The result of the Udacity simulator and the real world are shown in Figure 9 and Figure 10 respectively.

Figure 9: Left column:The scene in the real world video Right column: The masked scene in the real world video using VBP

The masked scenes show that the CNN is particularly interested in the road lanes and the edge of the side walk, which are the key reference objects that reveals the changing curve trend of the road. Also, some fast moving objects on both sides(the road bars in Figure 9 , pedestrians and truck in Figure 10 are also highlighted. All these are informative evidence for human during daily driving. What the CNN has learned are intuitively explainable. Besides, it should be noted that some of the background information in the real world scene are marginally activated with yellowish color.

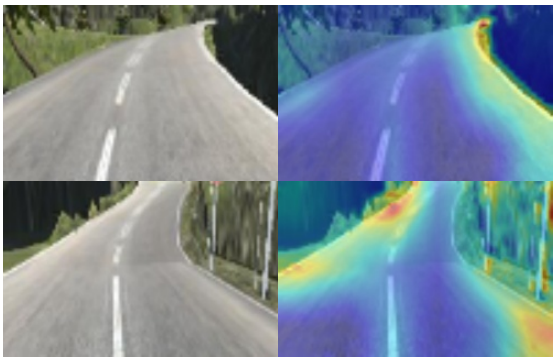

Figure 10: Results of scene from Udacity simulator.Left column:The original scene Right column: The masked scene of VBP

6 Discussion

6.1 Temporal information, a double edge sword

The results have shown that model with temporal input can significantly improve the steering performance. This is natural since the steering behaviour of human is not a discrete behaviour. Rather it is a continuous action that based on current and previous information. However, the model with input of dynamic image of 5 frames has a slightly worse accuracy in the hill testset. In [2], it mentioned although more frames can lead to smoother behaviour and more accurate prediction, it may cause the algorithm relies on wrong long-term dependencies.There are many sharp turns in the mountainous area and it is reasonable to say that relying on redundant past information may confuse the model. Figure 11 shows that dynamic image of 5 frames tends to generate more noisy information, like the blurring of trees,which may be regarded as the redundant temporal information. For an extreme case, if someone is driving from the Sydney Opera house to the Parliament House in Canberra, there is no clues that the driver shall utilize the vision information in Sydney when the car has arrived in Canberra.

Figure 11: Left: Dynamic image of 5 frames in hill testset. Right: Dynamic image of 5 frames in city testset.

6.2 Redundant activation by CNN

As mentioned, the activation map of CNN shows that the edges of road are significantly important for the prediction. However, the CNN also marginally activate some background areas, like the edges near the trees and sky,which may not be helpful in steering prediction. One interpretation is that redundant edge areas may interfere the prediction. This may cause problems when the image has a relatively large amount of edge areas. It also suggests that cropping the background information may be helpful in the model training.

7 Conclusion and Future Work

By doing comparative study on single-stream model and two stream model, we conclude that a) video sequence as well as temporal information are able to enhance the steering angle prediction performance. b) It is feasible to take the dynamic image as temporal information when doing the steering angle prediction. Redundant temporal information may degrade the performance.

Since the 2D CNN has limitations in mining more complicated features in time domain, network architectures like RNN, LSTM and 3D CNN are worth looking into in the future. Also, supervised learning method for autonomous steering is somehow unrealistic in large scale applications, deep reinforcement learning or generative adversarial network are the tools to tackle the problem.

8 What We Learned

We learned how to implement a CNN-based algorithm from scratch using Pytorch. We also learned knowledge in video processing(dynamic image,optical flow) and experiment technique in deep learning as well as the application of CNN in self-driving car . The skills in paper research and team collaboration are also what we learned.

References

[1] N. Fernandez, "Two-stream convolutional networks for end-to-end learning of self-driving cars," arXiv preprint arXiv:1811.05785, 2018.

[2] End-to-End Deep Learning for Steering Autonomous Vehicles Considering Temporal Dependencies

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

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

[5] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in Advances in neural information processing systems, 2014, pp. 568-576.

[6] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," nature, vol. 521, no. 7553, p. 436, 2015.

[7] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond short snippets: Deep networks for video classification," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 4694-4702.

[8] H. M. Eraqi, M. N. Moustafa, and J. Honer, "End-to-end deep learning for steering autonomous vehicles considering temporal dependencies," arXiv preprint arXiv:1710.03804, 2017.

[9] S. Du, H. Guo, and A. Simpson, "Self-driving car steering angle prediction based on image recognition," Department of Computer Science, Stanford University, Tech. Rep. CS231-626, 2017.

[10] . Bilen, B. Fernando, E. Gavves, A. Vedaldi, and S. Gould, "Dynamic image networks for action recognition," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 3034-3042.

[11] C. Rodriguez, B. Fernando, and H. Li, "Action anticipation by predicting future dynamic images," in Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 0-0.

[12] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," in Scandinavian conference on Image analysis, 2003, pp. 363-370: Springer.

[13] Sully Chen's dataset: `https://github.com/SullyChen/driving-datasets`

[14] Udacity simulator `https://github.com/udacity/self-driving-car-sim`

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

[16] M. Bojarski et al., "Visualbackprop: visualizing cnns for autonomous driving," arXiv preprint arXiv:1611.05418, vol. 2, 2016.

[17] M. A. Goodale and A. D. Milner. Se visual pathways for perception and action. Trends in Neuro-sciences, 1992, pp.20-25.

[18] B. Horn and B. Schunck. "Determining optical flow." Artificial intelligence 1981, pp. 185-203.

[19] K. Nagatani, et al. "Improvement of odometry for omnidirectional vehicle using optical flow information." Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2000.

[20] F. Kendoul, I. Fantoni, and K. Nonami. "Optic flow-based vision system for autonomous 3D localization and control of small aerial vehicles." Robotics and Autonomous Systems, 2009, pp. 591-602.

[21] L. Leal-Taixé, C. Canton-Ferrer, and K. Schindler. "Learning by tracking: Siamese CNN for robust target association." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. 2016.

[22] J. Wang, A. Cherian, and F. Porikli. "Ordered pooling of optical flow sequences for action recognition." 2017 IEEE Winter Conference on Applications of Computer Vision (WACV). IEEE, 2017.

Peer review:

Group Member	Work load distribution	Contribution ratio
Chaoxing Huang	Network implementation Method design	24%
Ruitao Leng	Data processing Test code implementation	18%
Zhaowen Xu	Parameter tunning Paper research	22%
Chensheng Zhang	Data analysis Model testing	18%
Xinyu Gao	Data processing Paper research	18%