Deep learning for autonomous driving

MinLong Chen[1, *]
[1] School of Physics and Astronomy, University of Nottingham, Nottingham, NG7 2RD, UK

With the development of artificial intelligence ,Autonomous driving has become the most exciting hot spots in computer vision.Many research works have proved that CNN are the suitable technology to boost autonomous driving.In this work,InceptionResnetV2 and Elizabeth CNN were trained and evaluated to boost the car to find the optimal angle and speed.We use both the exist kaggle dataset and self-captured dataset to train the model,use data augmentation and deploy the model to the Pi-car.Finally,the transfer model was selected as the model for Kaggle competition,While Elizabeth model proved to be the best model for pi-car live test due to hardware limitations and less inference time gap.

I. INTRODUCTION

Autonuomous cars are self-driving machines which combine AI algorithms and advanced feature engineer to help to operate with human driving.Based on the sensor data and 2d image captured by cameras, the AI-based model can perceive their environment and make appropriate choices.

Before the wide use of CNNs, most AI tasks were performed with human hand-crafted feature extraction and ML classfier.Due to the ability to learn automatically from 2D images,CNN has been widely used in end-to-end image reconition task.[1]

In autonuomous driving task,people usually collect vehicle's steering angle and speed via sensor and camera images,then train a end-to-end modified cnn model to map image features to angle and speed.[2]

The progress of autonomous car research has been driven by the combination of cutting-edge technologies and the need for more efficient transportation solutions. Google[3] has made significant contributions to autonomous driving research through their published papers on perception, planning, and machine learning algorithms. NVIDIA[2] has focused on deep learning for perceptual and simulation-based assessment, while the Tesla Autopilot system has been utilized into our reality life.

II. METHODS

A. Dataset Information

The dataset was provided on the kaggle competition website.It had 13794 images which were 3 channel RGB with size 240x320. These images include t-junction and straight tracks,pedestrians,obstructions and green red light.There is also a csv file mapping the image file to their target speed and steering angle.

The speed label is binary,it means 0 as stop and 1 as start.The speed is caculated by this formula.

$$speed_{norm} = \frac{speed - 0}{35}$$

The speed distribution is shown in fig. 1,it is unbalance.

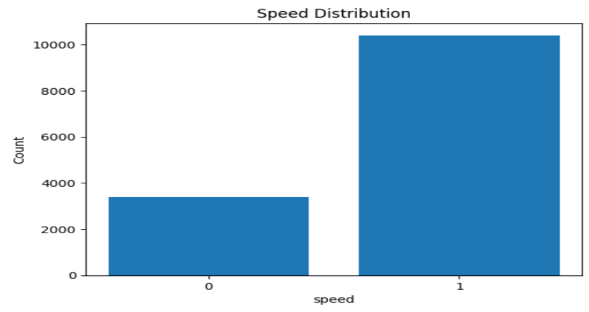

FIG. 1. dataset speed

The angle label is normalized decimal between 0 and 1,using the formula.

$$angle_{norm} = \frac{speed - 50}{80}$$

The speed distribution is also unbalance,according to fig. 2.It seems it lacks of enough large scale steering angle data.

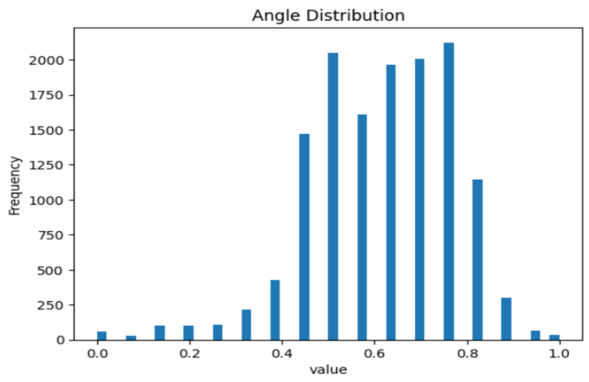

FIG. 2. dataset angle

* ppxcm4@nottingham.ac.uk

In addition,1020 images were given to predict and evaluate model performance.Moreover, to solve the unbalanced data problem, we record more data using pi-car camera including oval track,red green light,large scale steering angle and right side steering image.

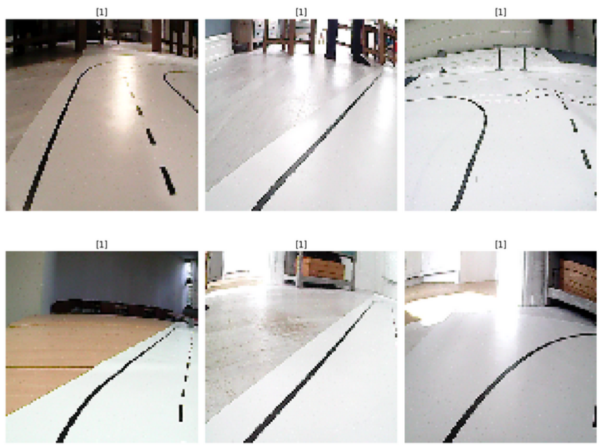

FIG. 3. picture in dataset

B. Data Pre-processing

For training dataset,we build a pandas dataframe, creating a column FilePath to store their image path.For external dataset,we build a script to interval the folder with DFS algorithm to mapping the file and their correspond label to one csv file.

Then an ImageDataGenerator was created which accepts these dataframes as inputs and augments their picture.The augmentation process contains normalizing each pixel to the range 0-1 by dividing 255,resizing the input image to 100x100 and using the batching size of 16.the train dataset has also been divided into train and validation set in the ration of 85:15.

To augment data,we also use random brightness in the range 0.8-1 and random pixel shift to simulate the shaky camera and dynamic light in real driving condition.For angle prediction task,we transfer the image from 3 channel rgb to 1 channel gray scale to reduce data dimensionality.For speed prediction,we still use RGB channel image for red green light prediction.

C. Loss Function

For the speed prediction task,this is a binary classification task,we employ LogLoss on it.

$$L(y,\hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}[y_i\log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)$$

For the angle prediction,this is a regression task,we employ Mean Square Error(MSE) on it.

$$MSE(y,\hat{y}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

D. CNN

Convolution Neural Network is a deep learning model inspired by the visual cortex of mammals,designed to process the grid-like architecture such as image.It can extract local and global features from the image,then followed by a fully connected layer for pattern recolonization task.[4]

1. Convolutional Layer

The convolution layer applies a convolution operation to the input image, which involves sliding a small filter over the image.By computing the matrix dot product between the filter and the image at each location,The result are used to create a feature map which highlights the fundamental patterns, such as edges, corners, or textures.

By stacking multiple convolution layers and using filters of different sizes and shapes,The model can detect more complex features.[4]

2. Pooling Layer

Pooling layer is a transformation which operate on each local region of an image and perform a down sampling operation.This helps to decrease the number of parameters in the network, which in turn reduces overfitting and improves the model's generalization ability.[4]

In max pooling, the pixel with the maximum value in the local region will be extracted and whereas the average pooling computes the average of each pixel in the region.

3. Activation function

Activation function are usually applied to the output of each neuron,which can introduce non-linearity and allow the network to learn complex patterns in data.[4] ReLU (Rectified Linear Unit) is a widely used activation function that outputs the input value if it is positive, and zero otherwise.

$$Relu(x) = \begin{cases} 0 & \text{if } x < 0, \\ x & \text{if } x \geq 0. \end{cases}$$

To adress "dying ReLU"problem,Leaky Relu can output a small negative instead of zero.

$$LeakyReLU(x) = \begin{cases} -0.01x & \text{if } x < 0, \\ x & \text{if } x \geq 0. \end{cases}$$

Sigmoid is used to finally output a value in the range 0-1,it is used here for speed and angle predict.

$$sigmoid(x) = \frac{1}{1+e^{-x}}$$

4. Fully connected layer

FCP layer is responsible for processing the features extracted by the CNN layers and making the final prediction of classification and regression output.In an FCP layer,each neuron is fully connected to every neuron in previous layer.The input feature from CNN will be flattened into a 1D vector,the output is then passed through a sigmoid function for prediction output.[4]

5. Elizabeth model

Nvdia has invited a 5 layer CNN model for self-driving problem[2],but this model can be overfitting on our limited dataset,so we define a 3 layer CNN Elizabeth model with 5x5 filter including one 32 feature map outputs and two 64 feature map outputs with MaxPool layer.Then,flatten these outputs to 1-D array and pass through FCP layer.

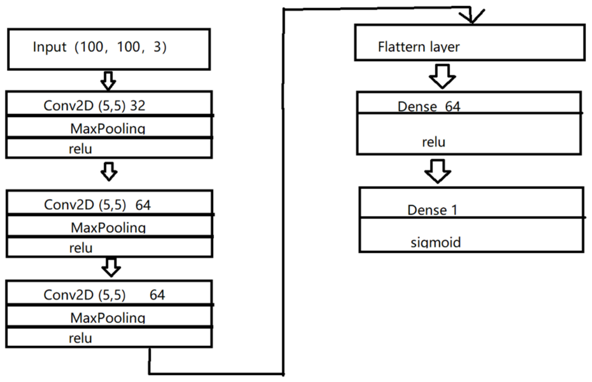

FIG. 4. Elizabeth model

E. Transfer Learning

Transfer Learing is a technique which can leverage the knowledge acquired from one task to improve performance on a different,but related task.[5]

The pre-trained model isusually trained on a large-scale dataset,such as ImageNet,we usually replace the top model with FCP dense model.By reusing the features,the model can achieve better performance with less training datasets.

1. ResNet model

As NN become deeper, the gradients of the loss function can become increasingly small leading to vanishing gradients problem.Microsoft researcher introduce the ResNet(Residual Neural Network) architecture in 2015 to solve this problem.[6]

with the invention of "skip connections",Microsoft connect a layer's original input to its output to produce a residual connection.These connections bypass one or more layers and allow the gradient to flow directly through the shortcut. This helps to alleviate the vanishing gradient problem.

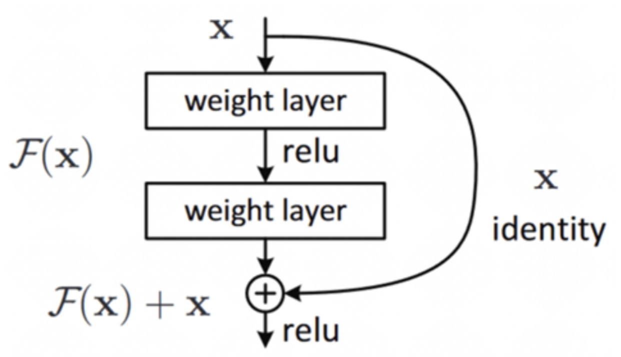
x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

FIG. 5. Skip connections[6]

2. Inception network

Inception network is deisigned with multiple inception modules which perform parallel convolutions of various filter(1x1,3x3,5x5) on the input image.The results of these inception models are then concatenated together to capture both local and global features at different levels of detail.[7]

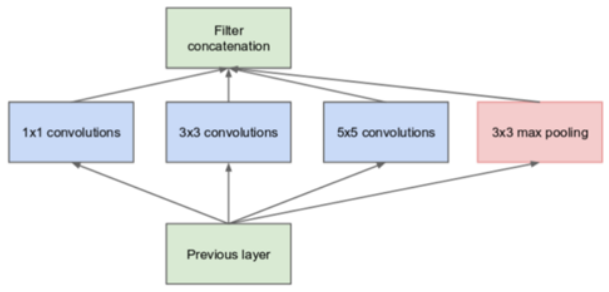

FIG. 6. Skip connections[7]

3. *InceptionResNet model*

InceptionResNet combine the advantages of ResNet and Inception networks to computer multiple parallel convolutions of different sizes and solve the vanishing gradient problem.It has been shown to perform well on a variety of computer vision tasks, including image classification and object detection.[8]

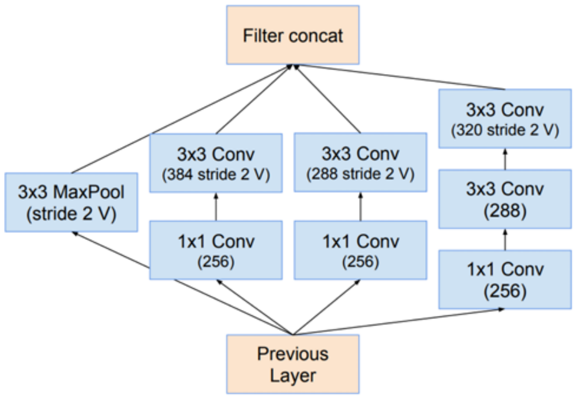

FIG. 7. InceptionResNet model[8]

F. Exponential Learning rate

Exponential learning rate is a method used to adjust the learning rate during training,the learning rate is decreased exponentially over time, according to a defined decay rate.It can help improve the stability of the model during training process.[9]

$$\alpha_t = \alpha_0 \left(\frac{\alpha_{\text{final}}}{\alpha_0} \right)^{\frac{t}{T}}$$

α_0 is initial learning rate,α_{final} is final learning rate.This expression will decay the learning rate from α_0 to α_{final} over T epochs, with an exponential decay curve. In this training process,α_0=0.001,α_{final}=0.0001,T=200.

G. Early Stop

Early stop technology is a effective way to conquer over-fitting problem by monitoring the loss on validation dataset. When the model's performance on the validation dataset stops improving or begins to degrade, the training process will be interrupted by early stop.In our training process,we set the shuffle to 5,it means after 5 epochs,if the validation loss do not perform better,the process will be interrupted.[10]

H. Dropout

Neural networks with a large number of parameters are very easy to be over-fitting,Dropout technology can effectively handle this problem. Dropout can randomly drop out a proportion of neurons in one layer during train by setting their weight to zero.This can reduce the over-fitting and the dependency between feature to improve the model's adaptability.[11]

I. L2 regularization

L2 regularization adds a penalty term to the loss fuction,which is related to the squre of the model's weight.[12]

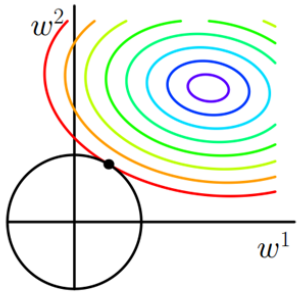

FIG. 8. L2 regularization[12]

the formula is $L(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|_2^2$,and it encourages the model to have smaller weights,more close to 0.

III. RESULTS

In this project,we built two separate neural networks, one for predicting the speed and the other for predicting the steering angle. Since the Pi-car's camera can only capture images with a resolution of 100x100, we have to resize the images. For the speed model, we used three-channel RGB images to enable red-green-light recognition, while for the angle model, we used single-channel gray scale images to avoid interference from color information.But for tranfer model,we use rgb chanel for both task.

A. Model Performance

In this project,we implement two types of model,Elizabeth CNN and InceptionResNet based transfer model and compare their performance.

For Elizabeth model,we use the same architecture with different loss function to deal speed and steering angle separately. The angle process has been trained for 25 epochs,with 0.0063 on training set and 0.0062 on validation set.

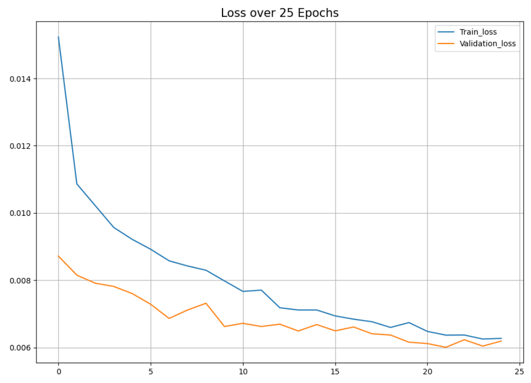

FIG. 9. CNN angle loss

The speed process has been trained for 21 epochs,with 0.0395 on training set and 0.095 on validation set,achieving 97.47 % on validation set.

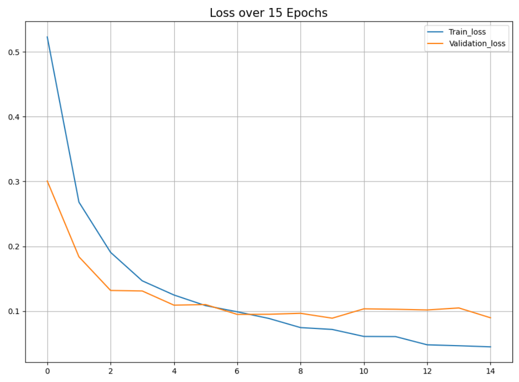

FIG. 10. CNN speed loss

For InceptionResNet transfer model,we use the same architecture with different loss function to deal speed and steering angle separately. The angle process has been trained for 25 epochs,with 0.0053 on training set and 0.0091 on validation set.

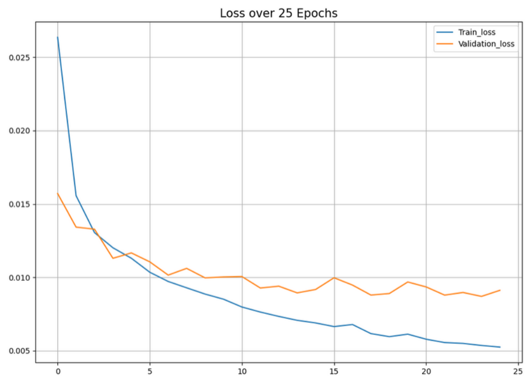

FIG. 11. InceptionResnet angle loss

The speed process has been trained for 21 epochs,with 0.0455 on training set and 0.1151 on validation set,achieving 96.46% accuracy on validation set.

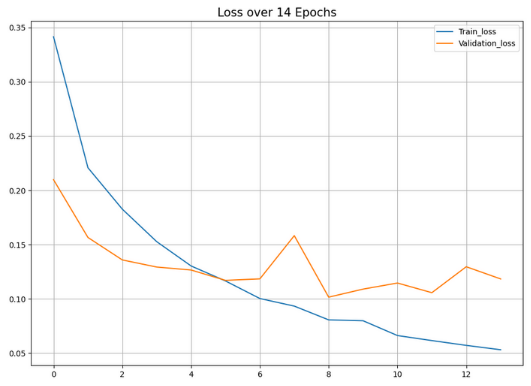

FIG. 12. InceptionResnet speed loss

And this table shows the Score with two of these models.The Elizabeth CNN model has better performance than the InceptionResNet transfer model.

Model	private score	public score
InceptionResNet	0.03895	0.04232
Elizabeth CNN	0.0305	0.03066

TABLE I. Performance with CNN and transfer model

B. Data Collection

The original dataset provided in Kaggle has a very serious imbalance problem,leading to perform very weak in live demo while showing a good score on Kaggle competition.We witness there is less steering picture in dataset so we collect more on the junction and oval tracks.with 2053 external images,the pi-car can corner well.

C. Model deployment

Due to the hardware limitation,the transfer model's inference time is nearly 140 ms,which will result in a large delay, making it problematic for the trolley to pass the oval,So we choose the Elizabeth model to deploy within 110ms.

To further reduce latency, we chose to use the tflite format instead of the previous PB format,reducing the delay time from 110 ms to 80ms,leading the speed and angle model can be well matched and the steering process is very smooth.

We also modify the model python file,this python file contains a function predict() which is used to transform angle model's predict images to the real pi-car angle,We make many optimization on it.For the range 88-92,we fix the output to 90,making the pi-car can run straightly on the road.For the range greater than 92,we use ceil() for upward rounding.For the range less than 88,we use floor() for downward rounding.By dong this,We can make the pi-car steering angle more bigger.

D. Live Demo score board

Track Event	Score
Straight Drive	3
Straight Drive; object at side	3
Straight Drive; pedestrian on track	3
Oval Drive; both directions	2
Oval Drive; object at side	3
Oval Drive; pedestrian on track	3
Figure of 8 drive; no stop	2
Figure of 8 drive; object in junction	3
Figure of 8 drive; traffic light	0
T-junction drive; left turn	0
T-junction drive; right turn	3
Oval Drive; max speed (50)	3

TABLE II. Live demo score

We get the 27 as our total score.But we fail to pass the traffic light and left turn in T-junction.

IV. CONCLUSIONS

In our project,we build 2 models for speed and steering angle prediction task,and we compare the performance between the Elizabeth CNN model and InceptionResNet transfer model.Also we do suitable data augmentation and collect external data.To handle with the live demo task,we use tflite format,although this may loss precision but can achieve low delay and then make the pi-car run steadily.

For limited amount of dataset,it is not a good way to use complex model,We first use NVIDIA's CNN model [2],but it has 5 convolution layers leading to over-fit in our dataset,so we build a 3 convolution layers.Although the inceptionResNet based transfer learning has some knowledge acquired from "image net" task,it perform not as good as the Elizabeth model.

Finally,DL model performs well in academic test such kaggle doesn't mean it will perform well in industry.Althoug our model has a very low loss on the kaggle competition,it can not go around corner at oval track because of the imbalence dataset and limited dataset,we have to collect more real data to fix this problem even this will lead decrease kaggle scores.

Appendix A: References

[1] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel, Backpropagation applied to handwritten zip code recognition (1989).

[2] M. Bojarski, D. W. del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, End to end learning for self-driving cars (2016).

[3] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, Y. Zhang, J. Shlens, Z. Chen, and D. Anguelov, Scalability in perception for autonomous driving: Waymo open dataset (2020).

[4] J. Murphy, An overview of convolutional neural network architectures for deep learning (2016).

[5] S. Bozinovski, Reminder of the first paper on transfer learning in neural networks, 1976 (2020).

[6] K. He, X. Zhang, S. Ren, and J. Sun, Deep residual learning for image recognition (2016).

[7] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, Going deeper with convolutions (2015).

[8] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, Inception-v4, inception-resnet and the impact of residual connections on learning (2017).

[9] Z. Li and S. Arora, An exponential learning rate schedule for deep learning (2019).

[10] R. Caruana, S. Lawrence, and C. Giles, Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping (2000).

[11] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting (2014).

[12] C. Cortes, M. Mohri, and A. Rostamizadeh, L2 regularization for learning kernels (2012).