DeepDrive an end to end autonomous vehicle

Tizayi Zirereza[1]

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

DeepDrive is a convolutional neural network created for the purpose of end to end vehicle automation. The model was implemented on a Raspberry Pi-Car with a single front-facing 120° camera and trained using training data derived from a range of toy tracks designed to simulate a number of different road conditions. The model was then tested on test data. As well as tasked to navigate the toy tracks in question during a live session. The model was able to achieve a mean square error loss on unseen test data of 0.02886. The model performed well in maintaining the correct lane on the straight and oval tracks as well as stopping for obstructions in the road. The model struggled with advancing at a green traffic light, directional road signs, and crossing a 4-way intersection.

I. INTRODUCTION

An autonomous vehicle is a vehicle able to sense its environment and control its movement accordingly. It is able to react to the shape of the road, obstruction in the road, other vehicles and pedestrians accordingly without input from a human diver. An autonomous vehicle is also able to appropriately deal with the dynamic environments such as cities by picking out the important information from its sensors to act upon. The information it uses to do this can be derived from a range of sensors such as front and rear mounted cameras, GPS, and radar. There are different levels of autonomy for vehicles which give different levels of control over to automated systems. The society of automotive engineers (SAE) have defined a classification system for describing the transition from fully manual to fully autonomous vehicle systems [4]. The classification is split into 5 levels:

- **Level 0:** The automated system issues warning and may intervene but is not responsible for sustained vehicle control.
- **Level 1:** The driver and the automated system share control of the vehicle.
- **Level 2:** The automated system takes control of the vehicle accelerating, braking and steering. However the driver must be prepared to take control if the automated system fails and thus must be alert at all times.
- **Level 3:** The driver can safely take their attention away from driving tasks. The driver may need to intervene if the automated system requests intervention.
- **Level 4:** No driver intervention is ever needed for the purpose of safety. The automated system can pull over by itself if a driver does not respond to an intervention request.
- **Level 5:** Full automation, the automated system is able to drive in all condition and situations a human driver would be able to. No human intervention required at all.

Level 0 and Level 1 are examples of automated systems in the vehicle but the vehicle is not autonomous. These levels have been implemented in car systems such as cruise control [4]. DeepDrive is a level 2 autonomous vehicle.

There are a number of different approaches to creating autonomous vehicles. One approach uses modular systems which operate as a pipeline of separate components mapping the sensory inputs to the actuator outputs. These separate components may have tasks like localisation , object detection , assessment, planning and vehicle control. The approach taken in this paper is an end to end model where one component is responsible for directly mapping sensory inputs to actuator outputs. In this paper this is done using supervised learning with a convolutional neural network but can also be done using deep reinforcement learning and other techniques [5].

The Nvidia model is an end to end self driving car model developed by the Nvidia corporation and described in their paper "End to End Learning for Self-Driving Cars" [2]. Training data for the model was collected from real world driving scenarios taken over many hours of human driving. The car had 3 cameras for which to draw information from these images would be fed into a convolutional neural network which would output a steering angle for each image. The notable parts of their model's architecture were the 5 convolutional layers, 3 fully connected layers, and the use of the ELU activation function. They also defined an autonomy score to assess the performance of their model on a test track and a real car. This score was based on the number of times a human had to intervene. With this autonomy score the model achieved a score of approximately 98% on their test track. The Nvidia model is an example of a level 2 autonomous vehicle.

In their paper "A Convolutional Neural Network Approach Towards Self-Driving Cars" [1] Agnihotri et al implemented a convolutional neural network to simulate the steering of an autonomous vehicle. The model was implemented on a Raspberry Pi-car and used ultra sonic

sensors to track the distance between objects in front of the car. The model used a deep convolutional neural network with 11 million parameters and was trained with 2 different data sets: Udacity containing 33,478 images and Nvidia containing 7,064 . It achieved validation mean squared square error loss of 0.0003798 and 1.013 respectively.

II. THE DEEP DRIVE MODEL

DeepDrive is a convolutional neural network designed as an end to end solution for the problem of autonomous vehicles. The model was implemented on a Raspberry Pi-car which uses a Raspberry Pi 4. Images from a front mounted wide angle 120°video camera were fed directly into DeepDrive. The network would then output a speed and steering angle within milliseconds for the car to execute. The model was trained and tested on a range of tracks and a range of tasks.

A. Data

The model was trained on 13793 training images of different simulated road conditions. This included straight roads, bends as well as obstructions in the road or by the side of the road. These images were designed to mimic a subset of the challenges a real life autonomous vehicle would have to encounter. A sample of the images used are shown in figure 1.

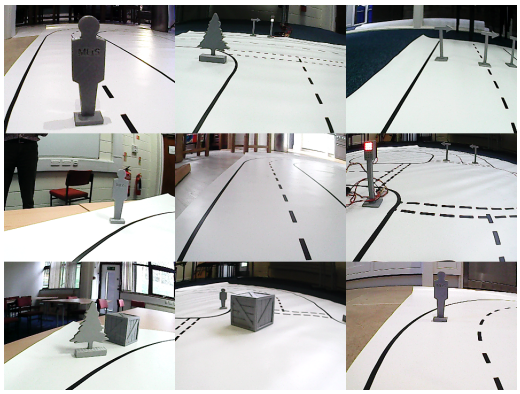

FIG. 1. A sample of 9 examples of training images

There were also 1020 test images to help evaluate the models performance. All images were 3 channel RGB images of size (240×320). Each training example had a label for its speed and steering angle. The speed was normalised to take a binary value of either 1 or 0 corresponding to the car stopping and moving the real world speed of the car is approximately $0.2ms^{-1}$. The steering angle was normalised to take values between 0 and 1:

$$angle_{nom} = \frac{(angle - 50)}{80} \tag{1}$$

.

B. Architecture

Finding the speed and angle are treated as a single regression problem with 2 output dimensions corresponding 2 neural network output units. The architecture can broadly be split up into 2 parts: the convolutional section and the fully connected section. The architecture is shown in figure 2.

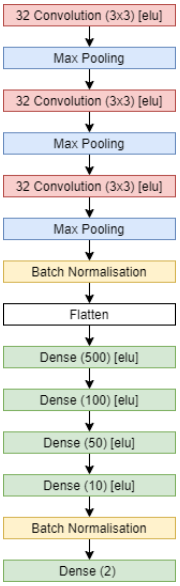

FIG. 2. The architecture of DeepDive

The convolutional section of the architecture is able to extract features from the image at different scales ,the use of the max pooling layers in between successive convolutional layers allows the successive convolutional layers to extract features at larger and larger scales in the image. The fully connected layers take these extracted features and maps then onto an output speed and an output steering angle. Batch normalisation layers were used as a regularisation method. The model had 1,186,494 trainable parameters and was compiled with the mean square error loss function as it is a regression problem. This model was implemented in pythons tensorflow API.

Exponential linear unit activation

The exponential linear unit (ELU) activation function was used for all layers apart from the output layer. The

activation function (ELU) can be written as:

$$ELU(z) = \begin{cases} z & z > 0 \\ \alpha(e^z - 1) & z \leq 0 \end{cases} \quad (2)$$

where z is an input and α is a positive constant. This activation function is linear for positive values but is exponential for negative values with the range of negative outputs determined by a positive constant α. The ELU function is shown in figure 3.

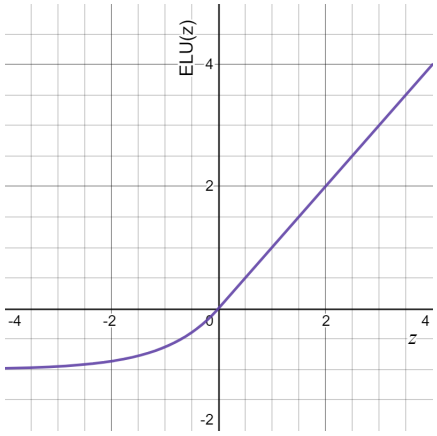

FIG. 3. A plot of the Exponential linear unit activation.

ELU is able to converge much faster and produce more accurate results [3] than ReLU activation function making it a strong alternative for ReLU . Another benefit of ELU is that it can output negative values unlike ReLU this means that ELU is not susceptible to dead neurons. ELU is also used in [2] for its training speed and efficiency.

C. Training

The training data was split into 2 sets: 80% of the data was used for training the model while 20% of the data was used for validation. The model was trained for 30 epochs with a learning rate of 0.001. A callback in tensorflow was used to save each iteration of the model in which the validation loss decreased and thus the final output of the training process would be the iteration of the model with the lowest validation loss. This method was implemented to prevent over flitting. The call back motivated the choice in the number of epochs as it allowed the model to be trained for an arbitrarily high number of epochs without worrying about over fitting. The choice in learning rate was motivated by spikes in the validation loss, a higher learning rate leads to larger gradient update steps which could cause the loss minimisation overshoot minima in the loss space thus causing peaks in the validation loss while a lower learning rate would increase the training time. 0.001 was found through trial and error to smooth out the validation loss while also not significantly increasing the time needed to train the model.

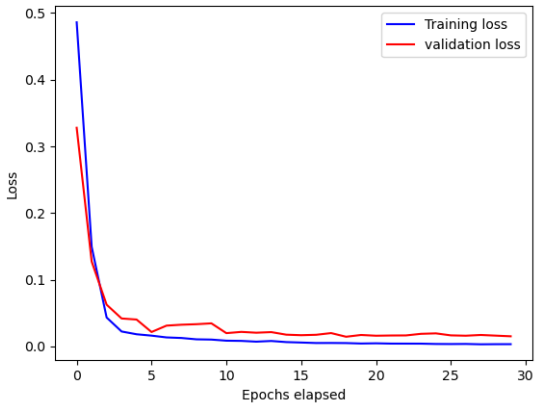

FIG. 4. The training and test loss. Note this graph is derived from a reconstruction of the training process with the same model and hyper parameters and is not the exact training process that produced DeepDrive

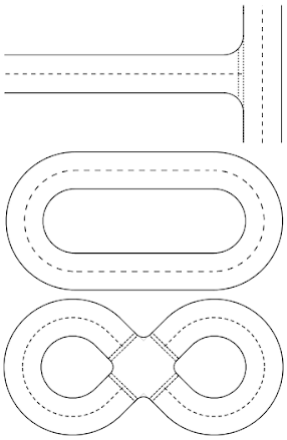

FIG. 5. The 3 tracks the automated vehicle was tested on: the T junction track , the oval track, and the figure 8 track

III. RESULTS

The model achieved a training loss of 0.0036 , a validation loss of 0.0152 and a test loss 0.02886 . The training and validation loss of the model are shown in figure 4.

A. Live Testing

The model was tested on a live track over a number of different situations. There were 3 tracks with a number of tasks for each track. The 3 tracks are shown in figure 5.

T junction track

The T junction track had 4 tasks associated with it:

- **Drive down a straight road:** The vehicle was required to drive along the straight part of the track while staying in the correct lane and not veering off the track.
- **Ignore Roadside tree:** The vehicle was required to drive down the track and ignore a tree figure on the side of the road.
- **Turn left:** The vehicle was required to turn left at the junction, with the directional arrows pointing left.
- **Turn right:** The vehicle was required to turn right at the junction, with the directional arrows pointing right.

The car was able to drive down the straight road and stay in the correct lane. The model was also able to ignore the roadside tree. The model was able to turn right correctly but was not able to turn left correctly, when the model came to the junction with the arrows pointing left the car chose to turn right instead.

Oval track

The Oval track had 3 tasks associated with it:

- **Drive around the oval:** The vehicle was required to complete one loop of the oval track while staying in the right lane and remaining on the track.
- **Ignore a roadside pedestrian:** The vehicle was required to drive around the oval track while ignoring a pedestrian figure at the side of the road.
- **Stop for pedestrian in the road:** The vehicle was required to stop for a pedestrian figure in the middle of the road.

The vehicle performed very well on this track and its associated tasks. The vehicle was able to drive around the track while remaining in the correct lane. It was also able to ignore the roadside pedestrian and stop for the pedestrian in the road. The vehicles steering on this track was very good.

Figure 8 track

The figure 8 track included 2 loops with an 4- way intersection which the vehicle would need to cross by going straight. The track had 4 tasks associated with it:

- **Drive around the figure 8:** The vehicle was tasked with making 1 complete loop of the track including crossing the intersection, and remaining in the correct lane.
- **Stop for Box in the road:** The vehicle was required to stop for a box obstructing the intersection.
- **Stop at read traffic light:** The vehicle was required to stop at a red traffic light at the intersection.
- **Continue for green light:** The vehicle was required to continue when the traffic light at the intersection turned green.

The vehicle did not perform well on this track while it was able to make one minor loop of the figure 8 it struggled once it came to the intersection. The vehicle would either just stop at the intersection or veer off the track completely. The model however was able to stop for the box obstructing the road as well as stop for the red traffic light but was not able to continue once the light had turned green.

IV. DISCUSSION

The model achieved a training loss of 0.0036 , a validation loss of 0.0152, and a test loss 0.02886. These are relatively low loss values. From figure 4 we can see that the training and validation loss seem to have plateaued. This indicates that the model has reached its limit in terms of minimising the loss. To further decrease the loss the model would need more a deeper convolutional section. Another possible improvement would be to split the model up into 2 different models for the steering angle and the speed. While this could increase the computational cost of running the model, it could mean each network could become optimised for extracting only the features relevant to its task as the high level features needed for deciding the vehicles speed and the vehicles angle are different.

The model seemed to generally perform well in driving around the straight and oval tracks it is likely the model was able to extract the information from the lanes on the road well. The model was also able to ignore roadside objects but act on obstructions in the road. This means it was able to detect these objects very well despite lack of specific object detection. The model struggled with the intersection, the traffic light, and making the left turn at the T junction. This may be due to the model not recognising the colour of the traffic light very well. It is possible that the model learned to stop at the sight of a traffic light rather than consider the colour of the light. To improve the models performance on these tasks object

detection methods could be employed as well as a deeper convolutional section.

The models poor performance at the figure 8 tracks intersection may also be improved by a deeper convolutional section to help recognise the features needed for the intersection. The models choice to turn right no matter which way the arrows were pointing may indicate an inability to see and consider the arrows all together. More testing is needed to find the exact cause of this behaviour but more robust object detection could also have helped with this issue.

DeepDrive is limited in the sense it only had access to one sensor the front facing camera to draw information from and was trained and tested in a very sanitised environment. In a real world environment an autonomous vehicle would need an army sensors to draw information from and would need to be able to deal with dynamic obstructions as well as generalise to roads and places it had never seen before. For this reason future work would also look into training the model with moving pedestrian and a number of other vehicles, obstructions, and road features.

V. CONCLUSION

DeepDrive performed well in maneuvering on straight and curved roads. DeepDrive also performed well in stopping for objects in the road and ignoring objects on the side of the road. The model did however struggle with the 4 way intersection, recognising greens lights, and left facing directional arrows. Future work would look into adding more object detection methods creating a deeper convolutional section, explore making 2 separate networks for each output value, and add more situations to train and test on.

REFERENCES

[1] Akhil Agnihotri, Prathamesh Saraf, and Kriti Rajesh Bapnad. "A Convolutional Neural Network Approach Towards Self-Driving Cars". In: *2019 IEEE 16th India Council International Conference (INDICON)*. 2019 IEEE 16th India Council International Conference (INDICON). Rajkot, India: IEEE, Dec. 2019, pp. 1–4. ISBN: 978-1-72812-327-1. DOI: 10.1109/INDICON47234.2019.9030307. URL: https://ieeexplore.ieee.org/document/9030307/ (visited on 05/19/2021).

[2] Mariusz Bojarski et al. "End to End Learning for Self-Driving Cars". In: (Apr. 25, 2016). arXiv: 1604.07316 [cs.CV].

[3] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)". In: *arXiv:1511.07289 [cs]* (Feb. 22, 2016). arXiv: 1511.07289. URL: http://arxiv.org/abs/1511.07289 (visited on 05/20/2021).

[4] *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles, J3016_201806*. 2014. URL: https://www.sae.org/standards/content/j3016_201806/.

[5] Ekim Yurtsever et al. "A Survey of Autonomous Driving: Common Practices and Emerging Technologies". In: *IEEE Access* 8 (2020), pp. 58443–58469. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.2983149. arXiv: 1906.05113. URL: http://arxiv.org/abs/1906.05113 (visited on 05/20/2021).