Safe Deep Neural Network-driven Autonomous Vehicles Using Software Safety Cages

Sampo Kuutti$^{1[0000-0002-7020-4370]}$, Richard Bowden$^{2[0000-0003-3285-8020]}$, Harita Joshi^{3}, Robert de Temple^{3}, and Saber Fallah$^{1[0000-0002-1298-1040]}$

^{1} Connected Autonomous Vehicles Lab, University of Surrey, UK
{s.j.kuutti,s.fallah}@surrey.ac.uk
^{2} Centre for Vision, Speech and Signal Processing, University of Surrey, UK
r.bowden@surrey.ac.uk
^{3} Jaguar Land Rover
{hjoshi3,rdetempl}@jaguarlandrover.com

Abstract. Deep learning is a promising class of techniques for controlling an autonomous vehicle. However, functional safety validation is seen as a critical issue for these systems due to the lack of transparency in deep neural networks and the safety-critical nature of autonomous vehicles. The black box nature of deep neural networks limits the effectiveness of traditional verification and validation methods. In this paper, we propose two software safety cages, which aim to limit the control action of the neural network to a safe operational envelope. The safety cages impose limits on the control action during critical scenarios, which if breached, change the control action to a more conservative value. This has the benefit that the behaviour of the safety cages is interpretable, and therefore traditional functional safety validation techniques can be applied. The work here presents a deep neural network trained for longitudinal vehicle control, with safety cages designed to prevent forward collisions. Simulated testing in critical scenarios shows the effectiveness of the safety cages in preventing forward collisions whilst under normal highway driving unnecessary interruptions are eliminated, and the deep learning control policy is able to perform unhindered. Interventions by the safety cages are also used to re-train the network, resulting in a more robust control policy.

Keywords: Automatic Control · Autonomous Vehicles · Cyber-physical Systems · Deep Learning · Safety.

1 Introduction

Autonomous vehicles are proposed as the future of intelligent transportation systems to address problems such as traffic congestion, pollution, and road safety [4, 14, 16, 19]. Deep learning has emerged as a popular artificial intelligence technique for autonomous vehicles, and has been proposed for many uses in autonomous vehicles, including vehicle control [13]. The downside of deep learning techniques is the opaqueness of the learned systems. In a safety-critical system, such as an autonomous vehicle, the safety of all sub-components as well as the overall system must be validated to a high level of safety assurance. The lack of interpretability in deep neural networks currently prevents effective safety validation. Moreover, the complex nature of the driving task and the operational environment mean that targeted testing methods are less useful due to the inability to test the autonomous vehicle in all possible use cases [3,10]. In order to introduce deep learning solutions to the next generation of intelligent vehicles, new safety validation techniques must be found to address them specifically [13].

Safety cages have been proposed for black box systems, where the safety of the system must be ensured without necessarily having full understanding of how the system works. Given the opaqueness of deep neural networks, such safety systems show great promise to improve the safety of the overall system. Safety cages eliminate unsafe actions by imposing limits on possible control actions. Utilising run-time monitoring, the safety cages can change dynamically based on the state of the system and its environment [7]. For instance, the safety cages in an autonomous vehicle can change the limits on acceleration based on the relative distance to nearby vehicles or the current speed of the vehicle. However, given that the safety cages intervene on the control output, a well designed safety cage must minimise unnecessary interventions. Safety cages have been used in cyber-physical systems where full offline safety validation is not possible, such as in robotics [6,12] or aerospace [17] applications, to intervene on the controller outputs in the presence of faults or dangerous control outputs. For autonomous vehicles, Heckemann et al. [7] suggested that these techniques could be useful for ensuring the safety of complex and adaptive machine learning systems in autonomous vehicles. Adler et al. [1] proposed safety cages based on five constraints such as "accelerating if a slower vehicle is closely in front" to meet the five Automotive Safety Integrity Levels (ASIL) defined in ISO26262 [8].

The contributions of this paper are three-fold. First, we present an imitation learning method for longitudinal control of an autonomous vehicle. The model is trained on data collected in IPG CarMaker [9], where the default driver demonstrates the desired driving policy. Second, we present two software safety cages designed to prevent forward collisions in an autonomous vehicle during highway driving. The safety cages are used to intervene on the control output of the neural network in safety critical scenarios, only when the network output is not reacting adequately to the current danger. Unnecessary interventions on the control output, which would degrade the performance of the controller and the comfort of the passengers, are minimised. Extensive testing under normal driving scenarios and in critical scenarios on two different neural networks demonstrate the effectiveness of the approach. Third, we demonstrate that the interventions by the safety cages can be used for re-training the network, improving the robustness of the learned policy to mistakes in novel states. The remainder of the paper is structured as follows. Section II describes the neural network algorithm developed for longitudinal control of an autonomous vehicle. Section III describes the developed safety cages to prevent forward collisions. Testing of the safety cages in a simulated car following scenario are presented in Section IV. Finally, concluding remarks and potential future work is given in Section V.

2 Longitudinal Control Algorithm

2.1 Data Collection

The collection of training data was carried out by defining various highway driving scenarios in IPG CarMaker [9] where the lead vehicle varies its velocity over time. In these scenarios, CarMaker's pre-defined driver, IPG Driver, was used to control the host vehicle and was set to maintain a 2s time headway from the lead vehicle. The velocity of the lead vehicle was in the interval [17, 30]m/s, whilst the acceleration was limited to the interval [-2, 2]m/s^2, since higher acceleration magnitudes would be uncomfortable for passengers [20]. All data was collected under dry road conditions with a road friction coefficient $\mu = 1.0$. The combined scenarios amount a total of 2 hours of driving data. Sampling the simulation at 50Hz, this amounts to 375,000 data points. The collected data set was then split into the training and validation data sets, with 80% of data in the training set and 20% in the validation set. This data was solely used for training and validating the deep learning control policy. A further 10 hours of simulation, for each control policy, was used in live testing of these policies in conjunction with the safety cages in a variety of scenarios and road conditions that were not seen during training, and hence would be expected to prove challenging for the control policy to generalise to.

2.2 Learning Algorithm

The inputs to the network were defined as time headway t_{hw}, relative velocity v_{rel}, host vehicle velocity v, and host vehicle acceleration a. The output of the network y was defined as a single parameter based on the gas pedal and brake pedal values such that $y \in$ [-1, 1], where positive values signal the use of the gas pedal whereas negative values correspond to the use of the brake pedal. The activation function at each hidden layer neuron is the Rectified Linear Unit (ReLU) function, whilst the output layer uses a tanh activation. The network output is then compared to the output of IPG Driver, $\hat{y}_k$, in the training data. The model is then trained through imitation learning, using the mean square error between predicted output and ground truth as the loss function. After performing a grid search for the network hyperparameters, the final network architecture has 3 hidden layers with 50 neurons each, trained with a learning rate of 1x10^{-2} and batch size of 100. The final model was trained for 1x10^6 training steps, resulting in a final validation loss of 0.0150463. For the purposes of testing the effectiveness of the safety cages on an unsafe controller, a second smaller neural network was also trained for the same task. The second suboptimal network has the same parameters as the above-mentioned network, except it only has one hidden layer with 10 neurons. Note, this network's parameters were not optimised for performance, as a suboptimal model gives a better opportunity to investigate the safety benefit offered by the proposed safety cages. To avoid confusion between the two trained networks, the deeper network with optimised hyperparameters will henceforth be referred to as the *deep network* and the smaller suboptimal network will be referred to as the *shallow network*. The losses during training for both networks can be seen in Fig. 1.

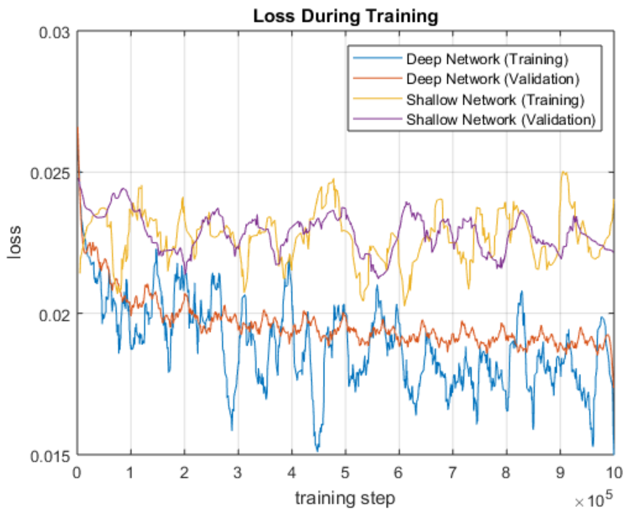


Fig. 1: Smoothed loss curves for training and validation sets.

3 Safety Cages

Software safety cages have been used in many applications as a means to improve safety by limiting outputs to a safe operational envelope. In their simplest forms, the safety cages can be hard upper/lower limits on the output. By using run-time monitoring to observe the state of the system and its environment, context-aware safety cages can use the observed states to dynamically limit the control output as the situation requires. For instance, in the problem of autonomous driving, a control output calling for full acceleration may be safe when there are no vehicles ahead but would be unsafe if the host vehicle is already close to the lead vehicle. Using such situational awareness, the potentially dangerous outputs of the neural network can be prevented by limiting their outputs during critical scenarios. Therefore, the safety validation requirements on the neural network can be relaxed, given that the software safety cages can be validated with high assurance [11].

The safety cages developed here focus on the longitudinal control of an autonomous vehicle described in previous sections, attempting to prevent forward collisions in highway driving scenarios. The safety cages observe the time headway (t_{hw}) and time-to-collision (TTC) to the lead vehicle as given by:

$$t_{hw} = \frac{x_{rel}}{v} \tag{1}$$

$$TTC = \frac{x_{rel}}{v_{rel}} \tag{2}$$

where v is the host vehicle velocity, v_{rel} is the relative velocity of the host and lead vehicles, and x_{rel} is the distance to the lead vehicle.

The TTC and $\mathrm{t_{hw}}$ were chosen as the metrics for the safety cages as they represent the risk of a forward collision. The TTC value represents the time required for two vehicles to collide if they continue at their current velocities and trajectories. Therefore, for a car following scenario in a single lane, such as the scenarios considered in this paper, a low TTC value means a forward collision is likely. However, TTC alone does not provide the full information regarding the risk of a forward collision. For instance, two vehicles driving at high speeds on the highway may be very close to each other, but if their relative speed is low or zero the TTC metric would not indicate the full risk of the situation. If the lead vehicle in this situation had to suddenly brake, the vehicle behind would be too close to react in time and prevent a collision from occurring. Therefore, a metric such as time headway is useful. Time headway represents the intervehicular distance in time, based on the host vehicle's velocity. Since t_{hw} does not make assumptions about the lead vehicle's actions as TTC does, it acts as a good safety metric in a car following scenario. These observed states are then used to identify a risk level for a possible forward collision. The risk levels were based on the TTC and $\mathrm{t_{hw}}$ based risk threshold presented in [2, 5, 15] with the final risk thresholds tuned to avoid collisions whilst minimising unnecessary interventions by the safety cages, as shown in Fig. 2. The safety cages impose a minimum brake pedal value when the risk of forward collision exceeds the given threshold. The safety cages intervene on the control action, if the neural network outputs a control action with less than the minimum required braking. Therefore, the safety cages do not decide the correct action for each scenario, but only intervene on the control action as a fall-back safety mechanism when the neural network is not responding to the level of risk adequately. Using the notation from the previous section, where a negative output y represents braking, the final control action is given by

$$y = min(y_{nn}, y_{sc}) \tag{3}$$

where y_{nn} is the output of the neural network and y_{sc} is the minimum required braking imposed by the safety cages.

4 Simulation Results

In order to investigate the effectiveness of the proposed safety cages, the two trained neural networks were tested under various car following scenarios. All

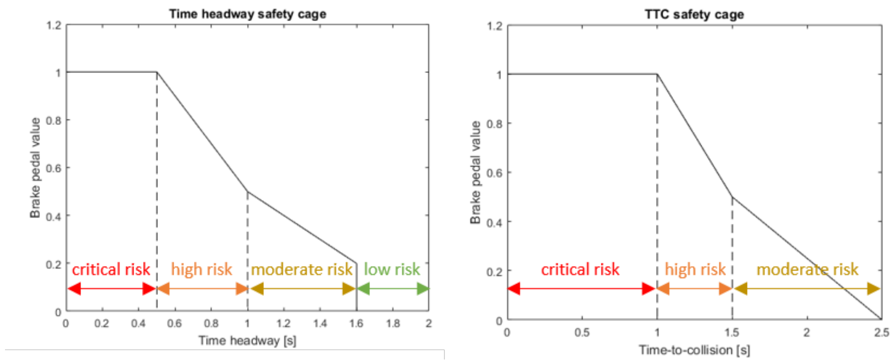
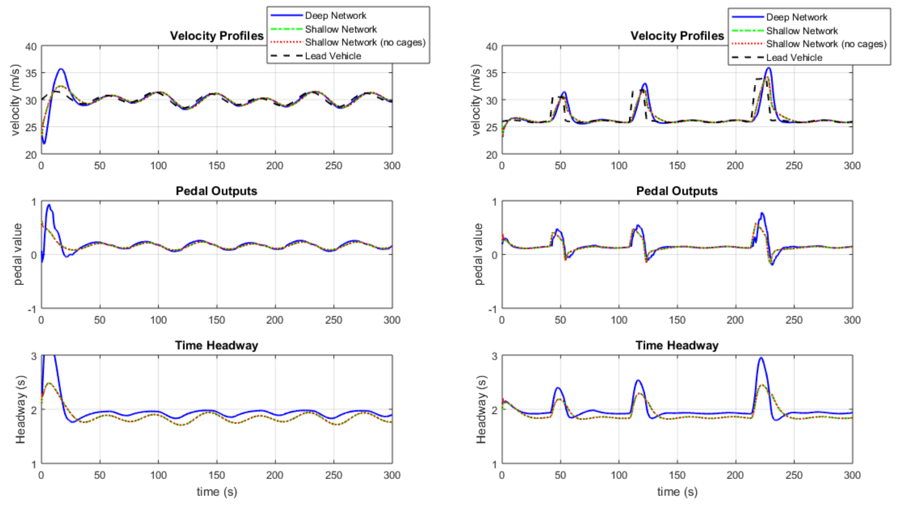


Fig. 2: Safety Cages with Time headway and Time-to-Collision based safety envelopes.

testing was done in the IPG CarMaker simulation platform. The scenarios were broken down to 5-minute episodes, where the episode ends after the 5 minutes or if a crash occurred. At the start of each episode a road friction coefficient value between 0.4 and 1.0 was chosen. The lead vehicle performed randomly chosen manoeuvrers, with the velocity limited to $v_{lead} \in$ [17, 40]m/s, and the acceleration limited to $a_{lead} \in$ [-2, 2]m/s^2. The exception to this was emergency braking manoeuvrers, which the lead vehicle performed, on average, once an hour. During emergency braking the deceleration was chosen between $a_{lead} \in$ [-6, -3]m/s^2. The combined testing includes 40 hours of driving overall, with various road conditions and different manoeuvrers performed by the lead vehicle. This includes testing the deep network with safety cages, shallow network with safety cages, shallow network without safety cages, and re-trained shallow network, for 10 hours each. We start the section by presenting results of the key types of scenarios (e.g. normal highway driving, emergency braking, wet road conditions) to investigate what the networks have learned and the effectiveness of the safety cages. Finally, we present and discuss the overall results of each 10 hour test run.

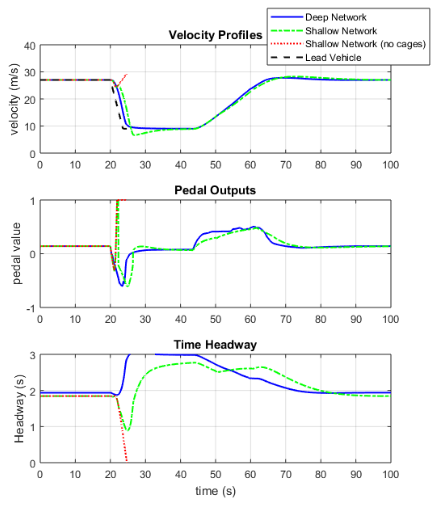
The first tests validated the performance of the networks under typical highway driving scenarios similar to those seen in the training data (Fig. 3a). Both networks show that they have learned a reasonable driving policy, keeping a safe headway close to the target headway of 2s. Moreover, no safety cage interventions are required in these scenarios for either network. This is not surprising as we are asking the networks to predict vehicle control actions for scenarios similar to those they were trained to operate in. Following this, the generalisation capability to completely new scenarios was tested. Firstly, the networks were tested under different road conditions. Since all training data was from dry road conditions (road friction coefficient μ = 1.0), the performance was tested under various μ values ranging from 0.4 to 1.0. Vehicle following scenario at μ = 0.55 can be seen in Fig. 3b. Both networks have learned to keep a safe time headway and adjust well to different road conditions, even if they were trained only on dry road condition. Again, no interventions by the safety cages occur in these scenarios. This in itself is impressive and demonstrates the power of deep learning.



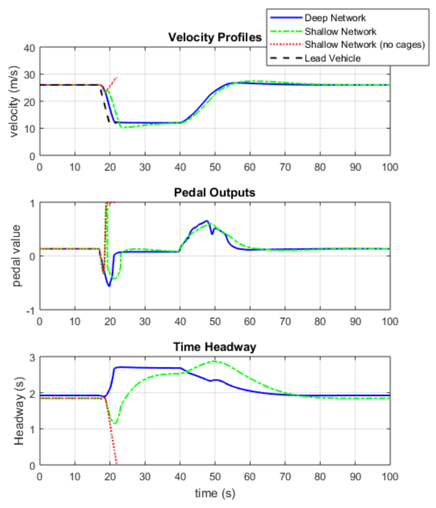
(a) (b)

Fig. 3: Neural network controllers following a lead vehicle at (a) $\mu = 1.0$ and (b) $\mu = 0.55$.

The second generalisation tests involved emergency braking performed by the lead vehicle. Since no emergency manoeuvrers were included in the training data, the networks' response to emergency braking by the lead vehicle is more interesting. Fig. 4a and 4b show emergency braking manoeuvrers performed by the lead vehicle with a deceleration of 5m/s^2 at μ values of 0.9 and 0.5, respectively. Here, the deep network generalises its previously learned rules to perform well at the emergency manoeuvrer, whilst the shallow network fails to generalise. Both emergency braking scenarios lead to the shallow network causing a forward collision with the lead vehicle when no safety cages are used. The shallow network initially starts to decelerate as the lead vehicle decelerates, but when seeing inputs to the network not seen during training, it cannot generalise to the new situation due to insufficient amount of parameters compared to the deep network and begins to accelerate until it crashes. However, when safety cages intervene on the shallow network's control action, the braking by the safety cages brings the vehicle back from the low time headway to a safe one, where the shallow network resumes control of the vehicle. Moreover, these scenarios



(a)



(b)

Fig. 4: Neural network controllers with the lead vehicle performing an emergency braking manoeuvrer (a) at $\mu = 0.9$, where the shallow network without safety cages crashes at t = 24.72 s and (b) at $\mu = 0.5$, where the shallow network without safety cages crashes at t = 21.96 s.

show that the safety cages can respond adequately to an emergency scenario by keeping the vehicle at a safe distance without excessive braking.

The networks were each tested in 10 hours of overall driving, for which the results can be seen in Table 1. The results from the IPG Driver demonstrator are also shown as a baseline for comparison. It can be seen from the results that the deep network has learned a safe driving policy, which keeps the vehicle at a safe distance from the lead vehicle without requiring the safety cages to intervene. The results show that the deep network has learned to outperform the IPG Driver, showing it generalises better to new scenarios (e.g. emergency braking) compared to the rule-based system. Also, given that the deep network can safely operate the vehicle, it can be seen that the safety cages never unnecessarily intervened on the control output, which could degrade the performance of the controller and lead to discomfort for the passengers. In comparison, the shallow network showed adequate performance in situations where the network inputs were in the same region as those seen during training. However, under emergency braking scenarios the shallow network fails to generalise and unexpectedly begins to accelerate until it crashes. This type of unexpected behaviour in new scenarios shows the efficacy of the proposed safety cages. The safety cages intervene on the network outputs, by decelerating the vehicle to a low risk region before handing the control back to the neural network again. Without the safety cages, the shallow network crashed 6 times in 10 hours, whilst using the safety cages caused the cages to intervene on 360 control actions (equal to only 14.40 s total duration) which prevented all collisions from occurring.

As an additional experiment for improving the robustness of the neural network, the interventions by the safety cages were used to augment the original training set. The shallow network was then re-trained for a further 100,000 training steps using the new augmented training set and previously stated training parameters. This approach is inspired by similar multi-stage training methods, such as DAgger [18], where the agent is initially trained using imitation learning, then tested in its intended operational domain, and the states seen during testing are re-labelled by the expert (which was used to generate the initial dataset) to create an augmented dataset, which is then used for re-training. However, for autonomous vehicles, the expert (i.e. human driver) would be costly to use for re-labelling all the states seen during testing. Instead, in our approach the safety cages provide a more accessible signal for ground truth, albeit only in scenarios where the neural network is outputting dangerous actions. Using this framework, the re-trained shallow network was then tested again through 10 hours of simulated driving, with the new results shown in the last row of Table 1. Although some interventions are still required, the results show that the re-training has improved the overall performance and safety of the shallow network. Thus, in our training framework, the agent is allowed to make mistakes and the knowledge from the safety cages is used to teach the network how to correct these mistakes, thereby boosting the robustness of the network without requiring costly queries to a human expert for ground truth labels.

Table 1: Neural network performance with 10 hours of testing.

Network	$\mathbf{x_{rel}}$ min./mean (m)	$\mathbf{v_{rel}}$ max./mean (m/s)	$\mathbf{t_{hw}}$ min./mean (s)	**collisions**	**safety cage violations**
IPG Driver	10.7372 / 75.1552	13.8896 / 0.1866	1.0459 / 2.5471	0	-
Deep	23.8440 / 57.3687	8.8781 / 0.0197	1.7383 / 1.9895	0	0
Shallow	7.2504 / 54.4619	13.4619 / 0.0096	0.7765 / 1.8836	0	360
Shallow (no safety cages)	0 / 54.4530	20.8884 / 0.0005	0 / 1.8789	6	-
Shallow (re-trained)	13.1289 / 57.5260	11.2086 / 0.0222	1.1527 / 1.9881	0	198

5 Concluding Remarks

In this paper, two safety cages were proposed for prevention of forward collisions in an autonomous vehicle. The safety cages aim to improve the safety and reliability of the system, without requiring any white-box knowledge of the machine learning system. This is achieved by limiting the control output of the system to a safe envelope, which is defined by the time headway and time-to-collision to the vehicle in front. Therefore, by using run-time monitoring to observe the state of the host vehicle and the lead vehicle, the control action can be limited dynamically by a context-aware software safety cage. The results presented in Section IV demonstrated the efficacy of the proposed safety cages. The safety cages were shown to correctly identify unsafe scenarios where control interventions were required and bring the vehicle back to a low risk region. Moreover, the results under normal highway driving indicate that the safety cages do not unnecessarily intervene on the controllers actions and degrade the overall performance of the system. Instead, the safety cages step in when the neural network shows unexpected behaviour (e.g. by failing to generalise to a completely new scenario not included in the training data set) and ensure safe control actions are used. Therefore, the safety cages increase the confidence in the safety of the autonomous vehicle, by ensuring that incorrect outputs can be eliminated and giving an idea of what the worst case performance of the vehicle would be in safety-critical scenarios. Furthermore, it was shown that interventions by the safety cages could be used for re-training the network, which improved the performance and safety of the learned policy.

This work opens multiple potential avenues for future work. The safety cages presented here mitigate forward collisions under highway driving. The presented techniques could be used to further develop safety cages to account for lateral control, urban driving, etc. To improve the safety offered by the safety cages, fault identification and mitigation could be used to identify the effect of faulty measurements on the effectiveness of the safety cages. Furthermore, the interventions by the safety cages could be leveraged to better understand what the network has learned (or has not learned) by identifying the edge cases where the network fails. Further improvements to the re-training framework could also be investigated, for example, by using iterative testing and re-training, or by addressing the imbalance between the scarce interventions of the safety cages relative to the significantly larger original dataset.

Acknowledgments. This work was supported by the UK-EPSRC grant EP/R512217/1 and Jaguar Land Rover.

References

1. Adler, R., Feth, P., Schneider, D.: Safety engineering for autonomous vehicles. In: 2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop (DSN-W). pp. 200–205. IEEE (2016)
2. Archer, J.: Indicators for traffic safety assessment and prediction and their application in micro-simulation modelling: A study of urban and suburban intersections. Ph.D. thesis, KTH (2005)
3. Burton, S., Gauerhof, L., Heinzemann, C.: Making the case for safety of machine learning in highly automated driving. In: International Conference on Computer Safety, Reliability, and Security. pp. 5–16. Springer (2017)

4. Department for Transport: Research on the Impacts of Connected and Autonomous Vehicles (CAVs) on Traffic Flow: Summary Report (2017)
5. Glaser, S., Vanholme, B., Mammar, S., Gruyer, D., Nouveliere, L.: Maneuver-based trajectory planning for highly autonomous vehicles on real road with traffic and driver interaction. IEEE Transactions on Intelligent Transportation Systems **11**(3), 589–606 (2010)
6. Haddadin, S., Haddadin, S., Khoury, A., Rokahr, T., Parusel, S., Burgkart, R., Bicchi, A., Albu-Schäffer, A.: On making robots understand safety: Embedding injury knowledge into control. The International Journal of Robotics Research **31**(13), 1578–1602 (2012)
7. Heckemann, K., Gesell, M., Pfister, T., Berns, K., Schneider, K., Trapp, M.: Safe automotive software. In: International Conference on Knowledge-Based and Intelligent Information and Engineering Systems. pp. 167–176. Springer (2011)
8. International Organization for Standardization: Iso 26262: Road vehicles-functional safety. International Standard ISO/FDIS (2011)
9. IPG Automotive GmbH: Carmaker: Virtual testing of automobiles and light-duty vehicles (2017), `https://ipg-automotive.com/products-services/simulation-software/carmaker/`
10. Kalra, N., Paddock, S.M.: Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? Transportation Research Part A: Policy and Practice **94**, 182–193 (2016)
11. Koopman, P., Wagner, M.: Challenges in autonomous vehicle testing and validation. SAE International Journal of Transportation Safety **4**(1), 15–24 (2016)
12. Kuffner Jr, J.J., Anderson-Sprecher, P.E.: Virtual safety cages for robotic devices (Dec 20 2016), uS Patent 9,522,471
13. Kuutti, S., Fallah, S., Bowden, R., Barber, P.: Deep Learning for Autonomous Vehicle Control: Algorithms, State-of-the-Art, and Future Prospects. Morgan & Claypool Publishers (2019)
14. Kuutti, S., Fallah, S., Katsaros, K., Dianati, M., Mccullough, F., Mouzakitis, A.: A survey of the state-of-the-art localization techniques and their potentials for autonomous vehicle applications. IEEE Internet of Things Journal **5**(2), 829–846 (2018)
15. Lu, J., Dissanayake, S., Xu, L., Williams, K.: Safety evaluation of right-turns followed by u-turns as an alternative to direct left turns: Crash data analysis. Florida Department of Transportation (2001)
16. Montanaro, U., Dixit, S., Fallah, S., Dianati, M., Stevens, A., Oxtoby, D., Mouzakitis, A.: Towards connected autonomous driving: review of use-cases. Vehicle System Dynamics pp. 1–36 (2018)
17. Polycarpou, M., Zhang, X., Xu, R., Yang, Y., Kwan, C.: A neural network based approach to adaptive fault tolerant flight control. In: Proceedings of the 2004 IEEE International Symposium on Intelligent Control, 2004. pp. 61–66. IEEE (2004)
18. Ross, S., Gordon, G., Bagnell, D.: A reduction of imitation learning and structured prediction to no-regret online learning. In: Proceedings of the fourteenth international conference on artificial intelligence and statistics. pp. 627–635 (2011)
19. Thrun, S.: Toward robotic cars. Communications of the ACM **53**(4), 99 (2010)
20. Treiber, M., Kesting, A.: Car-following models based on driving strategies. In: Traffic Flow Dynamics, pp. 181–204. Springer (2013)