Reinforced Damage Minimization in Critical Events for Self-Driving Vehicles

Francesco Merola, Fabrizio Falchi, Claudio Gennaro, Marco Di Benedetto
Institute of Information Science and Technologies "Alessandro Faedo" (ISTI)
National Research Council (CNR), Pisa (PI), Italy
{*name.surname*}*@isti.cnr.it*

Keywords: Autonomous Driving, Reinforcement Learning, Critical Scenarios, Deep Learning, Double Deep Q-learning, Vision Based

Abstract: Self-driving systems have recently received massive attention in both academic and industrial contexts, leading to major improvements in standard navigation scenarios typically identified as well-maintained urban routes. Critical events like road accidents or unexpected obstacles, however, require the execution of specific emergency actions that deviate from the ordinary driving behavior and are therefore harder to incorporate in the system. In this context, we propose a system that is specifically built to take control of the vehicle and perform an emergency maneuver in case of a dangerous scenario. The presented architecture is based on a deep reinforcement learning algorithm, trained in a simulated environment and using raw sensory data as input. We evaluate the system's performance on several typical pre-accident scenario and show promising results, with the vehicle being able to consistently perform an avoidance maneuver to nullify or minimize the incoming damage.

1 INTRODUCTION

In the last few years, research towards fully autonomous driving techniques has received massive attention and investments from both academic communities and private companies. Promises of life-changing safety and ease have been hung on these techniques, as self-driving vehicles have the potential to drastically change mobility and transport.

Significant progress has already been made and several studies (Urmson et al., 2008) (Levinson et al., 2011) (Broggi et al., 2013) (Kendall et al., 2019) have demonstrated that it is possible to successfully automate key driving tasks such as following the road and maintaining distance from other vehicles in normal circumstances. Critical situations, however, have shown to be particularly hard to handle and therefore currently constitute one of the main obstacles towards the realization of a fully autonomous driving system. These would include adverse weather scenarios with low visibility, intricate road topologies and traffic, sudden and unexpected obstacles and so on. The issue derives from the fundamental differences between the optimal behavior in normal operations and in critical events. In the latter case, the system is asked to perform an emergency action that often breaks the rules of ordinary driving. Steering into an empty sidewalk, for example, may be considered acceptable if it is needed to avoid a crash. Despite being rare, these events are crucial for the safeness on the road and potentially represent the point where an automated system could bring the most improvements with respect to an human driver, which makes them very worth of interest.

Given these premises, this work aims to investigate and propose a solution to address critical pre-accident scenarios. Starting from the assumption that the vehicle is already able to handle ordinary operations, the focus is put on learning to correctly navigate sudden dangerous situations. The proposed system is based on an end-to-end design trained with reinforcement learning (see Figure 1). Briefly, by using a *simulated environment*, the system learns to map sensory data coming from an RGB camera to vehicle actions directly by interacting with the virtual world in a trial-and-error fashion. The model updates are driven by a user defined *reward* that is designed to encourage correct decisions while punishing the wrong ones.

The remaining parts of this paper are organized as follows: Section 2 discusses some of the most important works related to the autonomous driving field; in Section 3 we present the architecture of the proposed system and the learning algorithm; Section 4 contains the experimental methodology used to test the system

as well as the achieved results; finally, the paper draws the conclusions in Section 5 with some future directives.

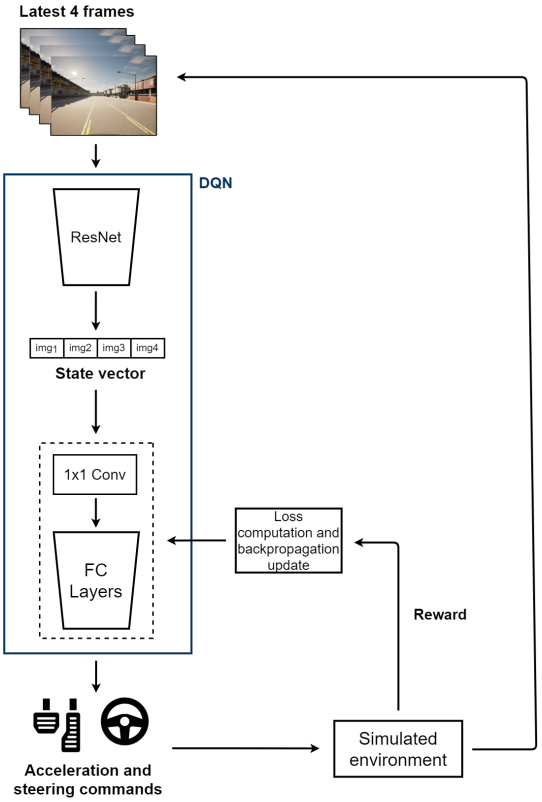

Figure 1: Learning algorithm used to optimize the critical event driving policy. At each step, the latest 4 frames gathered by the camera are subjected to feature extraction and then fed into the RL algorithm. The DQN produces the values associated with each action, then the best one is selected and applied. The resulting reward is used to compute the loss and update the weights of our custom feed-forward NN (represented by the dotted line).

2 RELATED WORK

Existing approaches usually tackle the autonomous driving problem from a general prospective and very few take into account critical events, which thus remains an open problem. That being said, most of the works in the current literature can be categorised as either *modular pipelines* or *end-to-end*.

Modular Pipelines The modular design represents the standard for autonomous driving, and is based on the idea of breaking down the complex mapping function from high dimensional inputs to vehicle control variables into independent modules that can be worked on separately. Some examples of this approach would include Boss from CMU (Urmson et al., 2008), Junior from Stanford (Levinson et al., 2011) and BRAiVE from University of Parma (Broggi et al., 2013). Their designs are all based on the typical modular pipeline, composed by the perception module, followed by scene prediction, planning and, finally, by the control module, responsible for generating motor commands. The environment sensing is generally carried out by some combination of cameras, radars, LiDARs and ultrasonic sensors, while computer vision tasks such as semantic segmentation (Badrinarayanan et al., 2017) and object detection (González et al., 2016) enable the vehicle to understand the scene. A recent work that is particularly relevant to this paper is the one by Vitelli et al. (Vitelli et al., 2021) that combines a machine learning planner with a rule-based fallback layer to safely tackle critical driving scenarios.

Supported by benchmark datasets (Geiger et al., 2012) (Maddern et al., 2017), this kind of approach has achieved good results and shown some clear advantages, such as the possibility of exploiting past knowledge for the construction of each module and the interpretability offered by the separate output of each of them. On the other hand, auxiliary loss functions are required to optimize each module separately, which poses a considerable complexity challenge and introduces the problem of error propagation (McAllister et al., 2017).

End-to-end Designs An alternative approach, commonly referred to as *end-to-end*, has gained popularity in recent time: aimed at combining the tasks of perception, planning and control into a single model, a deep neural network is trained end-to-end on the whole process. This way, all the parameters can be directly optimized with respect to a joint end goal, significantly reducing complexity. A solution leads to use *imitation learning* (Bojarski et al., 2016) (Xu et al., 2017), which is a supervised approach where the model learns to replicate the behaviour of an expert. Imitation learning, however, suffers from overfitting (i.e., a modelling error that aligns the function being modelled too close to the training exemplars) and is difficult to scale, since collecting expert data that covers all the possible scenarios is practically impossible. Because of this, at test time, the vehicle is likely to encounter new situations it has not been trained for and therefore failing to act in a proper way.

An alternative to the above solutions is *reinforcement learning* (RL), a technique that lets the vehicle learn directly by interacting with the environment

with the goal of maximizing a specific user defined reward (Sutton and Barto, 2020) (see Section 3.1).

Reinforcement learning has shown the ability to reach super-human performances in board games such as Gammon (Tesauro, 1995) and Go (Silver et al., 2017), as well as computer games (Mnih et al., 2015). The paradigm has also seen its first successful applications in the autonomous driving field, such as Sallab et al. (Sallab et al., 2017) and Kendall et al. (Kendall et al., 2019) works that are the closest to this paper and represented a major source of inspiration.

RL's biggest flaw is the need for an online training, meaning that the agent has to operate in the environment during the learning process, when the behavioral policy is not yet optimized. This implies that, in most real world applications, including this work, the training must be carried out with the aid of a simulator, which in turn introduces the need for an additional fine-tuning step to transfer the system from virtual to real environment. Despite these issues, RL is an extremely general and flexible framework, and we believe that these characteristics make for a useful tool to apply to critical driving events.

3 SYSTEM ARCHITECTURE

The system is built with the specific aim of handling pre-accident scenarios and assumes the existence of a base model that is able to drive in ordinary circumstances. Moreover, our proposal would ideally work in conjunction with a classificator module, in charge of discerning those scenarios that require emergency actions from those that don't (see Section 5). That said, by using RL as our methodology, the attention is placed on the optimization of avoidance maneuvers in order to minimize damage.

3.1 Driving as an MDP

In a RL approach, the problem is shaped as a Markov Decision Process (MDP) (Sutton and Barto, 2020) consisting of:

- a set of states $\mathcal{S}$
- a set of actions $\mathcal{A}$
- a transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0,1]$ that defines the probability of landing in a particular state s' at time t, given the state-action pair (s,a) at time *t-1*, for each $s' \in \mathcal{S}$
- a reward function $\mathcal{R} : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ that defines the reward $R(s,a,s')$ for each possible triple state s, action a, next-state s'
- a discount factor $\gamma \in [0,1]$ that regulates the weight of future rewards based on their distance in terms of time

The learner (and executor) is called *agent*, and the solution is represented by a policy $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0,1]$ that associates to every state-action pair (s,a) the probability of action a being selected from state s in order to maximize the obtained reward. RL is a broad paradigm that encompasses many algorithms, but this work focuses on Q-learning (Watkins and Dayan, 1992), which defines the Q-function as the objective to maximize:

$$Q_\pi(s,a) \; = \; \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a \right] \quad (1)$$

In other words, the aim is to learn a policy π that maximizes the expected cumulative reward over an arbitrarily long time horizon, that can either be finite or infinite. In this paper we structure the task in episodes, which corresponds to having a finite time horizon.

For our self-driving scenario, we are then required to define a state space $\mathcal{S}$, an action space $\mathcal{A}$ and a reward function $\mathcal{R}$. Given those three components, the transition function is automatically fixed by the mechanics of the used simulator.

State space The state space defines the type of information that the reinforcement learning agent receives at each time step about the environment. The first consideration to make concerns how to sense the environment. This work opts for the most straightforward route by solely relying on vision, just like humans do, to keep the approach simple. Moreover, cameras are relatively cheap and most vehicles have at least one on board, which makes the proposed system's setup cost very low. The data is hence provided by a single RGB camera, placed just above the windscreen. Because of the Markov property, future states should only be dependent on the present and not on the past. This means that, theoretically, every state should be a Markov representation of all previous observations. With this in mind, using a single image to describe the current environment state may be too much of an oversimplification: even a human would have trouble basing his decisions on a single RGB frame, as that would not provide enough information to determine direction, speed and steering angles of the agent and every other moving object in the scene. For this reason, the sequence of the most recent 4 frames is included in each environment observation.

The second consideration regards how to process the input image. Compressed representations are usually preferable (Müller et al., 2018), (Kendall et al.,

2019), therefore we use a residual neural network (ResNet) (He et al., 2016), pre-trained on the ImageNet dataset (Deng et al., 2009), as a feature extractor for each of the 4 frames. Directly extracting the highest level features, however, results in poor performance due to the aggressive pooling operated by the ResNet, which eliminates a lot of spatial information in favour of a classification oriented latent space. We therefore choose to extract features just before the pooling layer and then perform dimensionality reduction by means of a 1x1 convolutional layer trained for our driving task.

Action space The most important actions in the driving context are throttle, brake and steer. The reinforcement learning algorithm we use (see Section 3.2) requires the action space to be finite, therefore we opt for a two-dimensional discrete action space, dedicating one dimension to throttle/brake and one dimension to steer. This way, at each time step, the action can be represented as $a_t = (acc_t, steer_t)$, where $acc_t \in \{Brake, DoNothing, Throttle\}$ and $steer_t \in \{SteerLeft, GoStraight, SteerRight\}$. This ultimately results in 9 possible actions to choose from.

Reward function The reward function is critical as it drives the agent behavior evolution and should be carefully designed to encourage correct decisions while punishing the wrong ones. The first step is to clarify the behavior that the vehicle should learn which, in our case, can be summarized as "perform a quick emergency maneuver to minimize damage in the current critical event". With this in mind, we propose a reward structure based on several different factors.

The main factor is collision damage: the agent is heavily punished for colliding with an object. More precisely, every impact produces a negative reward with magnitude proportional to its intensity. This encourages the agent to avoid collisions as much as possible while also providing a feedback in those cases where complete damage nullification is not achievable, favouring mild impacts over strong ones.

Despite crash damage being undoubtedly the most important element in the evaluation of a road accident, a reward function solely based on it would be sparse, even more so as the system's performance improves and collisions get rarer. For this reason, we include other minor factors in the reward structure, with the aim of providing a stable signal to the agent and reinforcing a more comprehensive behavior. Covered distance, speed control and road following all fill this role. We therefore reward the agent for each meter traveled and punish it for braking when the speed is below a certain threshold v_{min}, throttling when speed is above a second threshold v_{max}, and steering in a way that widens the angle ϕ between its traveling direction and the road direction, as long as $|\phi|$ is greater than a certain steering margin ϕ_{max} (see Figure 2).

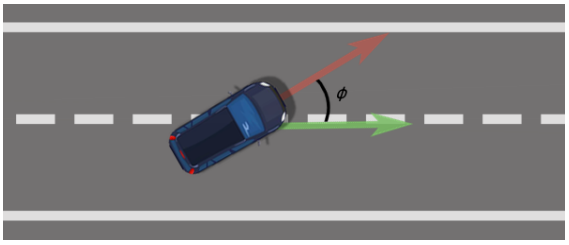

Figure 2: Angle ϕ between agent vehicle moving direction and road direction. While $|\phi|$ is greater than ϕ_{max}, steering towards the direction that further increases the angle results in a penalty.

3.2 Algorithm Choice: Deep Q-learning

For training we used our implementation of Deep Q-Learning (Mnih et al., 2013), one of the most popular reinforcement learning algorithms for discrete state spaces. Deep Q-learning is based on standard Q-learning (Watkins and Dayan, 1992), a model-free algorithm that aims at estimating the action-value function $Q(a,s)$ through a series of iterative updates based on the Bellman equation:

$$Q_*(s,a) = \mathbb{E}[R_{t+1} + \gamma \max_{a'} Q_*(s',a')] \quad (2)$$

Where $Q_*(s,a)$ is the value that the Q-function assumes under the optimal policy. Thanks to its recursive form, such equation provides a simple way to define a loss function with respect to a set of parameters θ, that can be approximated using a neural network (Deep Q-Network, DQN):

$$L(\theta) = [R + \gamma \max_{a'} Q_*(s',a';\theta) - Q(s,a;\theta)]^2 \quad (3)$$

Minimizing $L(\theta)$ is the objective of Deep Q-Learning.

Many implementations improve upon this basic version of the algorithm by incorporating some extensions. We use two of the most famous, namely *experience replay* and *target networks*, that are known to bring substantial performance gains (Mnih et al., 2015). Experience replay is a technique that lets the agent store its experiences in a buffer, called *replay memory*. The experience at a certain time step t is a tuple, $e_t = (s_t, a_t, r_{t+1}, s_{t+1})$, containing the state at time t, the action taken by the agent, the reward obtained at the following time step and the following state. The network is then trained by sampling batches of data from the buffer at each time step in order to increase

data efficiency and break the correlation between consecutive samples (Lin, 1992). The target network, instead, is a second network, with weights θ', that gets updated with a certain delay with respect to the first one and is used to compute the value of $Q_*(s',a')$ inside the loss function:

$$L(\theta) = [R + \gamma Q_*(s', \max_{a'} Q_*(s',a';\theta);\theta') - Q(s,a;\theta)]^2 \quad (4)$$

This effectively decouples target selection and evaluation, reducing the maximization bias typical of Deep Q-learning while also granting increased stability thanks to the delayed weights updates (He et al., 2016).

3.3 Training Architecture

The training process is illustrated in Figure 1. First, the vision data, composed by the 4 more recent frames provided by an RGB camera, is subjected to feature extraction by means of a pre-trained ResNet (in our case a ResNet18), gaining thus efficiency on a well-investigated architecture. The data flow is then intercepted before entering the average pooling layer (which would remove feature positional information), gathering then a latent space representation with dimension 7x7x512 for each input image, which is then concatenated with the previous other three on a final state vector of dimension 14x14x512. Such vector constitutes the input part of our custom feed-forward neural network (represented by the dotted rectangle in the figure), which is composed by a 1x1 convolutional layer used for dimensionality reduction, followed by 4 fully connected layers. The 1x1 convolution has 512 input and 16 output channels, resulting in a 14x14x16 representation. The encoded state is flattened before entering the linear layers, with hidden size equal to 256. This process outputs an estimate of the value related to each one of the possible actions. The action with the maximum output value is then selected and applied to the agent vehicle in the simulated environment. Finally, the simulation advances by one step producing a reward and a new RGB frame. The latter replaces the oldest of the previous frames and is then used to craft the following state vector. The reward, instead, is fed into the computation of the loss, which gets backpropagated to update the DQN's weights by means of gradient descent. Note that, despite being partly pre-trained, the DQN is a single model in charge of performing direct mapping from raw sensory data to vehicle actions. It is in this sense that we refer to our system as an end-to-end architecture.

Although the process is relatively simple, it is worth making some clarifications. First, the algorithm makes use of a replay buffer (see Section 3.2). This implies that the loss computation and weight update steps are performed by sampling mini batches of transitions from said buffer. Secondly, the training is done *off-policy*, meaning that actions performed during the learning phase come from a policy different from the one being optimized. This happens in order to widen the explored state-action distribution as much as possible, thus increasing robustness. There are several *exploration strategies* that can be used to achieve this (Tijsma et al., 2016). We opted for ε-*greedy*, which consists introducing a probability ε of selecting a random action instead of the best one.

4 EXPERIMENTS

We conduct our experiments in CARLA, an open urban-driving simulator specifically built to support development and validation of self-driving models (Dosovitskiy et al., 2017).

The experiments were structured into episodes of 60 time steps length, with 1 step being 0.1 simulated seconds long. This has two important implications. First, it means that the agent can act 10 times per second. Reducing the time step length would increase the responsiveness but also affect the computational load. We found the value 0.1 to be low enough for the agent to comfortably handle the critical scenarios, simultaneously keeping the load under control. The second implication concerns the total episode length in terms of simulated time, which is 6 seconds. We chose the smallest value that allows to fully solve all of the proposed scenarios when acting properly, in order to favour training efficiency in the system's target use case, i.e. critical driving events.

Pre-accident scenarios The system's performance is evaluated on three different typical pre-accident scenarios, shown in Figure 3. Situation (a) represents one of the most common pre-crash scenario involving two vehicles: the leading car suddenly brakes and the following one (the agent, in our case) must quickly react by either braking itself or performing a lane change to avoid collisions. Scenario (b) is also modeled after a relatively common occurrence. In this case the agent has to adjust its behavior to avoid crashing with a vehicle that suddenly crosses an unsignalized intersection at high speed. Finally, scenario (c) is more of a limit case, where the agent must react to a second vehicle, coming from the opposite direction, that suddenly loses control and invades the lane.

Note that in all of the above, the damage minimization's task difficulty changes depending on the

distance between the two vehicles, their speed, and the suddenness of the second vehicle's actions.

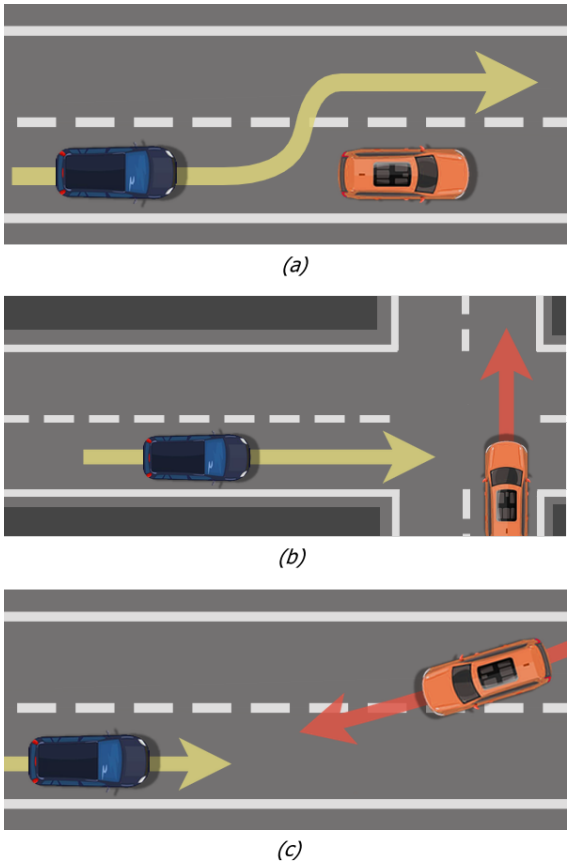

Figure 3: Scenarios used to evaluate the system, the agent is represented by the blue car. In scenario (a) the agent must avoid the preceding vehicle that suddenly brakes. Scenario (b) involves prevention of a crash at an intersection. In scenario (c) a vehicle loses control and invades the agent's lane, requiring a quick avoidance maneuver.

Training methodology For our experiments we chose to use one of the pre-built urban worlds (named *town03*) offered by CARLA. This is relevant, despite the episodes being short, as urban scenes are generally more cluttered with buildings and environmental objects than highway or rural ones.

We set the same probability for each scenario to appear and vary the agent's spawn location by randomly selecting among 5 different points in the simulated world, 3 of them being a rectilinear and 2 an intersection. Both vehicle's starting speed is randomly selected in the range 30-60 Km/h.

In terms of hyperparameters, we found the following configuration to be the most effective through preliminary testing carried out on a simplified scenario: slowly decaying learning rate in the range $[10^{-4}, 10^{-6}]$, discount factor of 0.99, target network update interval of 5000 steps and batch size 64.

The reward structure is described in Section 3.1, but it is important to note that the weight of each category is not the same. We found collision damage and speed control to be the most relevant elements in the agent's behavior shaping. Therefore, we set the magnitude to be 1 for steering actions (road following) and distance covered, and 5 for actions related to speed. The collision damage penalty, on the other hand, fluctuates between 1 and 100 depending on the impact intensity. Given the way we set up the reward signal, with high emphasis on discouraging wrong decisions, a negative sign is to be expected in most episodes and values approaching 0 can be considered good.

4.1 Results and Discussion

The training results are shown (Figure 4). The graph shows a constant improvement roughly until episode 6000, where the reward stabilizes with slight fluctuations at around -10. These results correspond to the agent being able to avoid collisions in almost every situation, with rare mild impacts in some of the most challenging ones. The system was capable of adopting the optimal behavior in each of the three proposed scenarios, correctly handling their internal variations as well (performing a more or less abrupt maneuver depending on the danger level, for example). This is particularly important as the experimental scenarios are fundamentally different from one another and therefore require non-trivial adaptation capabilities to be solved. Figure 5 shows an example of the agent vehicle's performance in all three critical scenarios.

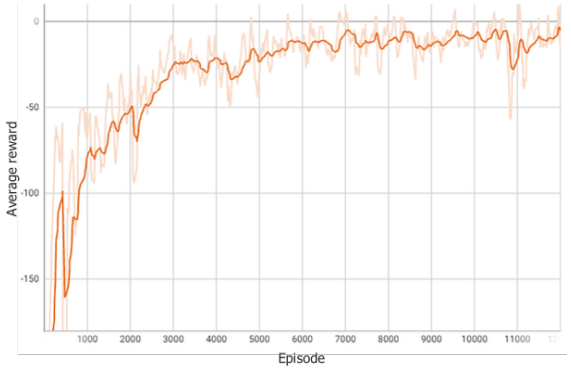

Figure 4: Experiments results - training graph reporting the average reward over the past 100 episodes. The average reward stabilizes at a value of around -10, corresponding to a good performance level in terms of collision avoidance.

On the negative side, despite its correctness, the

Figure 5: Three frame sequences illustrating the agent performance in the different critical scenarios. Each sequence shows two points of view: bird's eye view (above) and agent's camera (below), with time increasing left to right. In the first sequence the agent avoids the leading car that suddenly brakes. The second shows the agent braking at an intersection to let a second vehicle pass, then driving forward. In the last sequence, the agent performs a quick steering maneuver to avoid a vehicle that is invading the lane.

learned policy was a bit jerky at times, probably due to the discrete and rather small action space. This did not affect the system's ability to correctly read and adapt to different critical events, but it is worth noting as something to potentially improve upon in the future. Although the agent's performance is still not suitable for actual real world applications, the conducted experiments provided some interesting insights about the potential of the followed approach. The major finding is that reinforcement learning is able to solve the critical driving tasks in a relatively short time, as 6000 episodes took around 5 hours to complete on our

system, equipped with a very affordable GPU (GTX 1060 6GB). The fact that the model was able to learn how to manage all the different training scenarios is encouraging and confirms the generalization potential that RL offers as a framework, even when using a relatively simple off-the-shelf algorithm. We believe these factors suggest that reinforcement learning approaches to critical driving events deserve further investigations.

5 CONCLUSIONS AND FUTURE WORK

This paper proposes a deep reinforcement learning approach for self-driving in pre-accident scenarios, with the aim of investigating its effectiveness and safeness specifically in critical circumstances, when damage minimization should be the only priority.

The control system is based on an end-to-end design that directly maps raw sensory data, coming from a single RGB camera, to vehicle commands. The images gathered by the camera are subjected to feature extraction by means of a pre-trained ResNet and then fed into the RL algorithm for the learning process that takes place in a virtual environment. Model updates are driven by an hand-crafted reward function, specifically designed to encourage emergency maneuvers in critical situations, taking into account collision damage and other minor factors, namely speed control, road following and covered distance. Experiments were carried out on several typical pre-accident scenario recreated in the CARLA simulated world, where the autonomous vehicle showed promising performance, managing, in the vast majority of cases, to avoid collisions with the other vehicle in the scene.

Despite these encouraging results, the work leaves room for future improvements. First, as previously stated, our model is designed to be embedded in a broader system involving a module that takes care of driving in ordinary circumstances. To this aim, a neural network could be trained to identify critical scenarios and act as a switch between the two driving systems, passing the control to the emergency one when danger is detected.

Furthermore, the use of a more advanced reinforcement learning algorithm is also worth considering. Algorithms like *Deep Deterministic Policy Gradient (DDPG)* (Lillicrap et al., 2015) or *Soft Actor-Critic (SAC)* (Haarnoja et al., 2018) have recently shown good performances and their ability to deal with continuous action spaces could favour a more accurate and smooth driving style.

ACKNOWLEDGEMENTS

This work was partially funded by the H2020 project AI4Media "A European Excellence Centre for Media, Society and Democracy" under GA 951911.

REFERENCES

Badrinarayanan, V., Kendall, A., and Cipolla, R. (2017). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495.

Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., et al. (2016). End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*.

Broggi, A., Buzzoni, M., Debattisti, S., Grisleri, P., Laghi, M. C., Medici, P., and Versari, P. (2013). Extensive tests of autonomous driving technologies. *IEEE Transactions on Intelligent Transportation Systems*, 14(3):1403–1415.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V. (2017). Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR.

Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3354–3361. IEEE.

González, A., Vázquez, D., López, A. M., and Amores, J. (2016). On-board object detection: Multicue, multimodal, and multiview random forest of local experts. *IEEE transactions on cybernetics*, 47(11):3980–3990.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Kendall, A., Hawke, J., Janz, D., Mazur, P., Reda, D., Allen, J.-M., Lam, V.-D., Bewley, A., and Shah, A. (2019). Learning to drive in a day. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8248–8254. IEEE.

Levinson, J., Askeland, J., Becker, J., Dolson, J., Held, D., Kammel, S., Kolter, J. Z., Langer, D., Pink, O., Pratt, V., et al. (2011). Towards fully autonomous driving:

Systems and algorithms. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 163–168. IEEE.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Lin, L.-J. (1992). *Reinforcement learning for robots using neural networks*. Carnegie Mellon University.

Maddern, W., Pascoe, G., Linegar, C., and Newman, P. (2017). 1 year, 1000 km: The oxford robotcar dataset. *The International Journal of Robotics Research*, 36(1):3–15.

McAllister, R., Gal, Y., Kendall, A., Van Der Wilk, M., Shah, A., Cipolla, R., and Weller, A. (2017). Concrete problems for autonomous vehicle safety: Advantages of bayesian deep learning. International Joint Conferences on Artificial Intelligence, Inc.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533.

Müller, M., Dosovitskiy, A., Ghanem, B., and Koltun, V. (2018). Driving policy transfer via modularity and abstraction. *arXiv preprint arXiv:1804.09364*.

Sallab, A. E., Abdou, M., Perot, E., and Yogamani, S. (2017). Deep reinforcement learning framework for autonomous driving. *Electronic Imaging*, 2017(19):70–76.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017). Mastering the game of go without human knowledge. *nature*, 550(7676):354–359.

Sutton and Barto (2020). *Reinforcement Learning, An Introduction*. The MIT Press.

Tesauro, G. (1995). Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3):58–68.

Tijsma, A. D., Drugan, M. M., and Wiering, M. A. (2016). Comparing exploration strategies for q-learning in random stochastic mazes. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8. IEEE.

Urmson, C., Anhalt, J., Bagnell, D., Baker, C., Bittner, R., Clark, M., Dolan, J., Duggins, D., Galatali, T., Geyer, C., et al. (2008). Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(8):425–466.

Vitelli, M., Chang, Y., Ye, Y., Wołczyk, M., Osiński, B., Niendorf, M., Grimmett, H., Huang, Q., Jain, A., and Ondruska, P. (2021). Safetynet: Safe planning for real-world self-driving vehicles using machine-learned policies. *arXiv preprint arXiv:2109.13602*.

Watkins, C. J. and Dayan, P. (1992). Q-learning. *Machine learning*, 8(3-4):279–292.

Xu, H., Gao, Y., Yu, F., and Darrell, T. (2017). End-to-end learning of driving models from large-scale video datasets. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2174–2182.