Learning to Drive in a Day

Alex Kendall **Jeffrey Hawke** **David Janz** **Przemyslaw Mazur** **Daniele Reda**
John-Mark Allen **Vinh-Dieu Lam** **Alex Bewley** **Amar Shah**
Wayve
research@wayve.ai

Abstract

We demonstrate the first application of deep reinforcement learning to autonomous driving. From randomly initialised parameters, our model is able to learn a policy for lane following in a handful of training episodes using a single monocular image as input. We provide a general and easy to obtain reward: the distance travelled by the vehicle without the safety driver taking control. We use a continuous, model-free deep reinforcement learning algorithm, with all exploration and optimisation performed on-vehicle. This demonstrates a new framework for autonomous driving which moves away from reliance on defined logical rules, mapping, and direct supervision. We discuss the challenges and opportunities to scale this approach to a broader range of autonomous driving tasks.

1 Introduction

Autonomous driving is a topic that has gathered a great deal of attention from both the research community and companies, due to its potential to radically change mobility and transport. Broadly, most approaches to date focus on formal logic which defines driving behaviour in annotated 3D maps. This can be difficult to scale, as it relies heavily on mapping infrastructure rather than primarily using an understanding of the local scene.

In order to make autonomous driving a truly ubiquitous technology, we advocate for robotic systems which address the ability to drive and navigate in absence of maps and explicit rules, relying - like humans - on a comprehensive understanding of the immediate environment [1] while following higher level directions (e.g., turn-by-turn route commands). Recent work has demonstrated that this is possible on rural roads, using GPS for coarse localisation and LIDAR to perceive the scene [2].

In recent years, reinforcement learning (RL) – a machine learning subfield focused on solving Markov Decision Problems (MDP) [3] where an agent learns to select actions in an environment in an attempt to maximise some reward function – has shown an ability to achieve super-human results at games such as Go [4] or chess [5], a great deal of potential in simulated environments like computer games [6], and on simple tasks with robotic manipulators [7]. We argue that the generality of reinforcement learning makes it a useful framework to apply to autonomous driving. Most importantly, it provides a corrective mechanism to improve learned autonomous driving behaviour.

To this end, in this paper we:

1. pose autonomous driving as an MDP, explain how to design the various elements of this problem to make it simpler to solve, whilst keeping it general and extensible,
2. show that a canonical RL algorithm (deep deterministic policy gradients [8]) can rapidly learn a simple autonomous driving task in a simulation environment,
3. discuss the system required to make learning efficient on a real-world vehicle,
4. learn to drive a real-world autonomous vehicle in a few episodes with a continuous deep reinforcement learning algorithm, using only on-board computation.

We present the first demonstration of a deep reinforcement learning agent driving a real car.

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

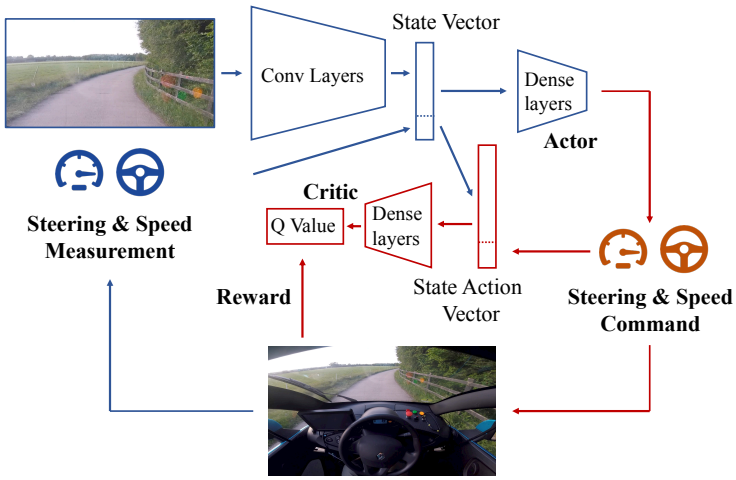


Figure 1: We design a deep reinforcement learning algorithm for autonomous driving. This figure illustrates the actor-critic algorithm which we use to learn a policy and value function for driving. Our agent maximises the reward of distance travelled before intervention by a safety driver. A video of our vehicle learning to drive is available at `https://wayve.ai/blog/l2diad`

2 Related Work

We believe this is the first work to show that deep reinforcement learning is a viable approach to autonomous driving. We are motivated by its potential to scale beyond that of imitation learning, and hope the research community examines autonomous driving from a reinforcement learning perspective more closely. The closest work in the current literature can predominantly be categorised as either imitation learning or classical approaches relying on mapping.

Mapping approaches. Since early examples [9, 10], autonomous vehicle systems have been designed to navigate safely through complex environments using advanced sensing and control algorithms [11, 12, 13]. These systems are traditionally composed of many specific independently engineered components, such as perception [1], localisation [15], state estimation [14], mapping, planning and control [16]. However, because each component needs to be individually specified and tuned, this can be difficult to scale to more difficult driving scenarios due to complex interdependencies. These modular mapping approaches are largely the focus of commercial efforts to date.

Imitation learning. A more recent approach to some driving tasks is imitation learning [19, 20], which aims to learn a control policy by observing expert demonstrations. One important advantage of this approach is that it can use end-to-end deep learning, optimising all parameters of a model jointly with respect to an end goal thus reducing the effort of tuning of each component. However, imitation learning is also challenging to scale. It is impossible to obtain expert examples to imitate for every potential scenario an agent may encounter, and it is challenging to deal with distributions of demonstrated policies (e.g., driving in each lane).

Reinforcement learning. Reinforcement learning is a broad class of algorithms for solving Markov Decision Problems (MDPs) [21]. The solution of an MDP is a policy $\pi\colon \mathcal{S} \rightarrow \mathcal{A}$ that for every $s_0 \in \mathcal{S}$ maximises the future discounted reward, given by the two Bellman equations.

In other words, reinforcement learning algorithms aim to learn a policy π that obtains a high cumulative reward. They are generally split into two categories: model-based [22] and model-free reinforcement learning. In the former approach, explicit models for the transition and reward functions are learnt, and then used to find a policy that maximises cumulative reward under those estimated functions. In the latter, we directly estimate the value $Q(s, a)$ of taking action a in state s, and then follow a policy that selects the action with the highest estimated value in each state.

In autonomous driving, deep learning has been used to learn dynamics models for model-based reinforcement learning using off-line data [23]. Reinforcement learning has also been used to learn autonomous driving agents in video games. However, this can simply the problem, with access

to ground truth reward signals which are not available in the real-world, such as the angle of the car to the the lane [8]. The closest work to this paper is from Riedmiller et al. [24] who train a reinforcement learning agent which drives a vehicle to follow a GPS trajectory in an obstacle-free environment. They demonstrate learning on-board the vehicle using a dense reward function based on GPS thresholded tracking error. We build on this work in a number of ways; we demonstrate learning to drive with deep learning, from an image-based input, using a sparse reward function to lane follow.

3 System Architecture

3.1 Driving as a Markov Decision Process

A key focus of this paper is the set-up of driving as an MDP. Our goal is that of autonomous driving, and the exact definition of the state space $\mathcal{S}$, action space $\mathcal{A}$ and reward function R are free for us to be defined. The transition model is implicitly fixed once a state and action representation is fixed, with the remaining degrees of freedom – the transitions themselves – dictated by the mechanics of the simulator/vehicle used.

State space. Key to defining the state space is the definition of the observations O_t that the algorithm receives at each time step. Many sensors have been developed in order to provide sophisticated observations for driving algorithms, not limited to LIDAR, IMUs, GPS units and IR depth sensors; an endless budget could be spent on advanced sensing technology. In this paper, we show that for simple driving tasks it is sufficient to use a monocular camera image, together with the observed vehicle speed and steering angle.

A second consideration is how to treat the image itself: the raw image could be fed directly into the reinforcement learning algorithm through a series of convolutions [25]; alternatively, a small compressed representation of the image, using, for example, a Variational Autoencoder (VAE) [26] [27], could be used. We compare the performance of reinforcement learning using these two approaches in Section 4. In our experiments, we train the VAE online from five purely random exploration episodes, using a KL loss and a L2 reconstruction loss [27].

Action space. Driving itself has what one might think are a natural set of actions: throttle, brake, signals etc. But what domain should the output of the reinforcement learning algorithm be? Overall, experiments on a simple simulator (Section 4.1) showed that continuous actions, whilst somewhat harder to learn, provide for a smoother controller. We use a two-dimensional action space; steering angle in the range [-1, 1] and speed setpoint in km/h.

Reward function. Design of reward functions can approach supervised learning – given a lane classification system, a reward to learn lane-following can be set up in terms of minimising the predicted distance from centre of lane, the approach taken in [8]. This approach is limited in scale: the system can only be as good as the human intuition behind the hand-crafted reward. We do not take this approach. Instead, we define the reward as forward speed and terminate an episode upon an infraction of traffic rules – thus the value of a given state $V(s_t)$ corresponds to the average distance travelled before an infraction.

3.2 Reinforcement Learning Algorithm – Deep Deterministic Policy Gradients

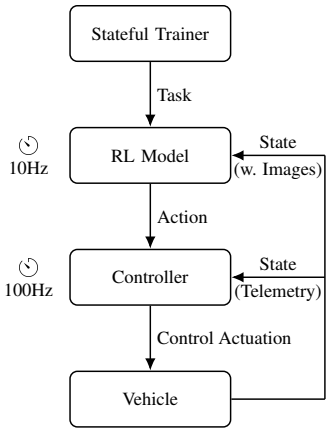
We selected a simple continuous action domain model-free reinforcement learning algorithm: deep deterministic policy gradients (DDPG) [8], to show that an off-the-shelf reinforcement learning algorithm with no task-specific adaptation is capable of solving the MDP posed in Section 3.1.

DDPG consists of two function approximators: a critic $Q\colon \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, which estimates the value $Q(s, a)$ of the expected cumulative discounted reward upon using action a in state s, trained to satisfy the Bellman equation under a policy given by the actor $\pi\colon \mathcal{S} \to \mathcal{A}$, which attempts to estimate a Q-optimal policy $\pi(s) = \text{argmax}_a Q(s, a)$. The error in the Bellman equality, which the critic attempts to minimise, is termed the temporal difference (TD) error. Many variants of actor-critic methods exist, see e.g. [28, 29].

DDPG training is done online and is an off-policy learning algorithm, meaning that actions performed during training come from a policy distinct from the learn optimal policy by the actor. Our exploration policy is formed by adding discrete Ornstein-Uhlenbeck process noise [31] to the optimal policy.

```
1: while True do
2:     Request task
3:     Waiting for environment reset
4:     if task is train then
5:         Run episode with noisy policy
6:         if exploration time is over then
7:             Optimise model
8:         end if
9:     else if task is test then
10:        Run episode with optimal policy
11:    else if task is undo then
12:        Revert previous train/test task
13:    else if task is done then
14:        Exit experiment
15:    end if
16: end while
```

(a) Task-based workflow for on-vehicle training



(b) Policy execution architecture, used to run episodes during model training or testing.

Figure 2: Outline of the workflow and the architecture for efficiently training the algorithm from a safety driver's feedback.

3.3 Task-based Training Architecture

Deployment of a reinforcement learning algorithm on a full-sized robotic vehicle running in a real world environment requires adjustment of common training procedures, to account for both driver intervention and external variables affecting the training.

We structure the architecture of the algorithm as a simple state machine, outlined in Figure 2a, in which the safety driver is in control of the different tasks. This provides an on-demand execution of episodes instead of an a priori fixed schedule. The train and test tasks allow us to interact with the vehicle in autonomous mode, executing the current policy. The difference between the two tasks consists in noise being added to the model output and the model being optimised in training tasks, whereas test tasks run directly the model output actions. During early episodes, we skip optimisation to favour exploration of the state space. We continue the experiment until the test reward stops increasing. Each episode is executed until the system detects that automation is lost (i.e. the driver intervened). In a real world environment, the system can not reset automatically between episodes, this is done by a human safety driver.

4 Experiments

The main task we use to showcase the vehicle is that of lane-following; this is the same task as addressed in [8], however done on a real vehicle as well as on simulation, and done from image input, without knowledge of lane position. It is a task core to driving, and was the cornerstone of the seminal ALVINN [19]. We first accomplish this task in simulation in Section 4.1, and then use these results and knowledge of appropriate hyperparameters to demonstrate a solution on a real vehicle in Section 4.2.

For both simulation and real-world experiments we use a small convolutional neural network. Our model has four convolutional layers, with 3×3 kernels, stride of 2 and 16 feature dimensions, shared between the actor and critic models. We then flatten the encoded state and concatenate the vector the scalar state for the actor, additionally concatenating the actions for the critic network. For both networks we then apply one fully-connected layer with feature size 8 before regressing to the output. For the VAE experiments, a decoder of the same size as the encoder is used, replacing strided convolution with transposed convolution to upsample the features. A graphical depiction is shown in Figure 1.

4.1 Simulation

To test reinforcement learning algorithms in the context of lane following from image inputs we developed a 3D driving simulator, using Unreal Engine 4. It contains a generative model for country

Figure 3: Examples of different road environments randomly generated for each episode in our lane following simulator. We use procedural generation to randomly vary road texture, lane markings and road topology each episode. We train using a forward facing driver-view image as input.

	Training			Test	
Model	Episodes	Distance	Time	Meters per Disengagement	# Disengagements
Random Policy	-	-	-	7.35	34
Zero Policy	-	-	-	22.7	11
Deep RL from Pixels	35	298.8 m	37 min	143.2	1
Deep RL from VAE	11	195.5 m	15 min	-	0

Table 1: Deep RL results on an autonomous vehicle over a 250m road. We report the best performance for each model. We observe the baseline agent can learn to lane follow from scratch, while the VAE variant is more efficient, learning to successfully drive the route after 11 training episodes.

roads, supports varied weather conditions and road textures, and will in the future support more complex environments (see Figure 3 for game screenshots).

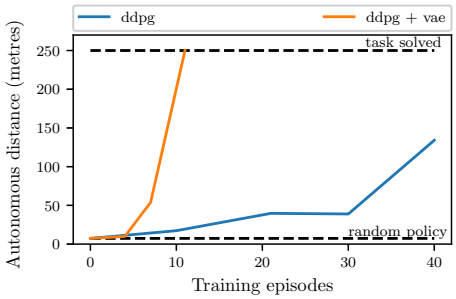
The simulator proved essential for tuning reinforcement learning parameters including: learning rates, number of gradient steps to take following each training episode and the correct termination procedure – conservative termination leads to a better policy. It confirmed a continuous action space is preferable – discrete led to a jerky policy – and that DDPG is a suitable reinforcement learning algorithm. As described in the environment setup in Section 3.1, reward granted in the simulator corresponded to the distance travelled before exiting lane, with new episodes resetting the car to the centre of the lane.

We found that we could reliably learn to learn follow in simulation from raw images within 10 training episodes. Furthermore, we found little advantage to using a compressed state representation (provided by a Variational Autoencoder). We found the following hyperparameters to be most effective, which we use for our real world experiments: future discount factor of 0.9, noise half-life of 250 episodes, noise parameters of θ of 0.6 and σ of 0.4, 250 optimisation steps between episodes with batch size 64 and gradient clipping of 0.005.

4.2 Real-world driving

Our real world driving experiments mimic in many ways those conducted in simulation. However, executing this experiment in the real world is significantly more challenging. Many environmental factors cannot be controlled, and real-time safety and control systems must be implemented. For these experiments, we use a 250 meter section of road. The car begins at the start of the road to commence training episodes. When the car deviates from the lane and enters an unrecoverable position, the safety driver takes control of the vehicle ending the episode. The vehicle is then returned to the center of the lane to begin the next episode. We use the same hyperparameters that we found to be effective in simulation, with the noise model adjusted to give vehicle behaviour similar to that in simulation under the dynamics of the vehicle itself.

We conduct our experiments using a modified Renault Twizy vehicle, which is a two seater electric vehicle, shown in Figure 1. The vehicle weighs 500kg, has a top speed of 80 km/h and has a range of 100km on a single battery charge. We use a single monocular forward-facing video camera mounted in the centre of the roof at the front of the vehicle. We use retrofitted electric motors to actuate the brake and steering, and electronically emulate the throttle position to regulate torque to the wheels. All computation is done on-board using a single NVIDIA Drive PX2 computer. The vehicle's drive-by-wire automation automatically disengages if the safety driver intervenes, either by using vehicle controls (brake, throttle, or steering), toggling the automation mode, or pressing the emergency stop. An episode would terminate when either speed exceeded 10km/h, or drive-by-wire automation disengaged, indicating the safety driver has intervened. The safety driver would then reset the car to the centre of the road and continue with the next episode.



(a) Algorithm results

(b) Route

Figure 4: Using a VAE with DDPG greatly improves data efficiency in training over DDPG from raw pixels, suggesting that state representation is an important consideration for applying reinforcement learning on real systems. The 250m driving route used for our experiments is shown on the right.

Table 1 shows the results of these experiments. Here, the major finding is that reinforcement learning can solve this problem in a handful of trials. Using 250 optimisation steps with batch size 64 took approximately 25 seconds, which made the experiment extremely manageable, considering manoeuvring the car to the centre of the lane to commence the next episode takes approximately 10 seconds anyway. We also observe in the real world, where the visual complexity is much more difficult than simulation, a compressed state representation provided by a Variational Autoencoder trained online together with the policy greatly improved reliability of the algorithm. We compare our method to a zero policy (driving straight with constant speed) and random exploration noise, in order to confirm that the trial indeed required a non-trivial policy. [1]

5 Discussion

This work presents the first application of deep reinforcement learning to a full sized autonomous vehicle. The experiments demonstrate we are able to learn to lane follow with under thirty minutes of training – all done on on-board computers.

In order to tune hyperparameters, we built a simple simulated driving environment where we experimented with reinforcement learning algorithms, maximising distance before a traffic infraction using DDPG as a canonical algorithm. The parameters found transferred amicably to the real-world.

In this work, we present a general reward function which asks the agent to maximise the distance travelled without intervention from a safety driver. While this reward function is general, it has a number of limitations. It does not consider conditioning on a given navigation goal. Furthermore, it is incredibly sparse. As our agent improves, interventions will become significantly less frequent, resulting in weaker training signal. It is likely that further work is required to design a more effective reward function to learn a super-human driving agent. This will involve the careful consideration of many safety [32] and ethical issues [33].

Another area for development suggested by the results here is a better state representation. Our experiments have shown that a simple Variational Autoencoder greatly improves the performance of DDPG in the context of driving a real vehicle. Beyond pixel-space autoencoders is a wealth of computer vision research addressing effective compression of images: here existing work in areas such as semantic segmentation, depth, egomotion and pixel-flow provide an excellent prior for what is important in driving scenes [34, 1, 35]. This research needs to be integrated with reinforcement learning approaches for real tasks, both model-free and model-based.

The algorithm used here is intentionally a canonical approach, chosen to demonstrate how reinforcement learning may be applied to driving. There is no doubt that more advanced reinforcment learning algorithms would improve performance [37, 38, 39, 40, 41]. We hope this paper inspires more research into applying reinforcement learning research to autonomous driving, perhaps combining it with elements from other machine learning techniques such as imitation learning and control theory. The method here solved a simple driving task in half an hour – what more could be done in a day?

[1] A video of the training process for our vehicle learning to drive the 250m length of private road with the stateful RL training architecture (Section 3.3) is available at `https://wayve.ai/blog/l2diad`

References

[1] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for scene segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[2] Teddy Ort, Liam Paull, and Daniela Rus. Autonomous vehicle navigation in rural environments without detailed prior maps. In *International Conference on Robotics and Automation (ICRA)*, 2018.

[3] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[4] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[5] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy P. Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *CoRR*, abs/1712.01815, 2017.

[6] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[7] Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 3389–3396. IEEE, 2017.

[8] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2016.

[9] Takeo Kanade, Chuck Thorpe, and William Whittaker. Autonomous land vehicle project at cmu. In *Proceedings of the 1986 ACM fourteenth annual conference on Computer science*, pages 71–80. ACM, 1986.

[10] Richard S Wallace, Anthony Stentz, Charles E Thorpe, Hans P Moravec, William Whittaker, and Takeo Kanade. First results in robot road-following. In *IJCAI*, pages 1089–1095. Citeseer, 1985.

[11] Michael Montemerlo, Jan Becker, Suhrid Bhat, Hendrik Dahlkamp, Dmitri Dolgov, Scott Ettinger, Dirk Haehnel, Tim Hilden, Gabe Hoffmann, Burkhard Huhnke, et al. Junior: The stanford entry in the urban challenge. *Journal of field Robotics*, 25(9):569–597, 2008.

[12] Jesse Levinson, Jake Askeland, Jan Becker, Jennifer Dolson, David Held, Soeren Kammel, J Zico Kolter, Dirk Langer, Oliver Pink, Vaughan Pratt, et al. Towards fully autonomous driving: Systems and algorithms. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 163–168. IEEE, 2011.

[13] Uwe Franke, Dariu Gavrila, Steffen Gorzig, Frank Lindner, F Puetzold, and Christian Wohler. Autonomous driving goes downtown. *IEEE Intelligent Systems and Their Applications*, 13(6):40–48, 1998.

[14] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT press, 2005.

[15] Chris Linegar, Winston Churchill, and Paul Newman. Made to measure: Bespoke landmarks for 24-hour, all-weather localisation with a camera. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Stockholm, Sweden, May 2016.

[16] Urs Muller, Jan Ben, Eric Cosatto, Beat Flepp, and Yann L Cun. Off-road obstacle avoidance through end-to-end learning. In *Advances in neural information processing systems*, pages 739–746, 2006.

[17] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[18] Will Maddern, Geoffrey Pascoe, Chris Linegar, and Paul Newman. 1 year, 1000 km: The oxford robotcar dataset. *The International Journal of Robotics Research*, 36(1):3–15, 2017.

[19] Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *Advances in neural information processing systems*, pages 305–313, 1989.

[20] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[21] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*. MIT press, 1998.

[22] Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML)*, pages 465–472, 2011.

[23] Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M Rehg, Byron Boots, and Evangelos A Theodorou. Information theoretic mpc for model-based reinforcement learning. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 1714–1721. IEEE, 2017.

[24] Martin Riedmiller, Mike Montemerlo, and Hendrik Dahlkamp. Learning to drive a real car in 20 minutes. In *Frontiers in the Convergence of Bioscience and Information Technologies, 2007. FBIT 2007*, pages 645–650. IEEE, 2007.

[25] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[26] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *The International Conference on Learning Representations (ICLR)*, 2014.

[27] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on machine learning (ICML)*, 2014.

[28] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.

[29] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2016.

[30] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *International Conference on Learning Representations (ICLR)*, 2015.

[31] G. E. Uhlenbeck and L. S. Ornstein. On the theory of the brownian motion. *Phys. Rev.*, 36:823–841, Sep 1930.

[32] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

[33] Judith Jarvis Thomson. The trolley problem. *The Yale Law Journal*, 94(6):1395–1415, 1985.

[34] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[35] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[36] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[37] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897, 2015.

[38] Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. *CoRR, abs/1507.06527*, 2015.

[39] Maximilian Igl, Luisa Zintgraf, Tuan Anh Le, Frank Wood, and Shimon Whiteson. Deep variational reinforcement learning for pomdps. In *Proceedings of the 28th International Conference on machine learning (ICML)*, 2018.

[40] Gregory Farquhar, Tim Rocktäschel, Maximilian Igl, and Shimon Whiteson. Treeqn and atreec: Differentiable tree planning for deep reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2018.

[41] David Ha and Jürgen Schmidhuber. World models. *CoRR*, abs/1803.10122, 2018.