Simulation-based reinforcement learning for real-world autonomous driving

Błażej Osiński
deepsense.ai, University of Warsaw
`b.osinski@mimuw.edu.pl`

Adam Jakubowski
deepsense.ai

Piotr Miłoś
deepsense.ai,
Institute of Mathematics of the Polish Academy of Sciences

Paweł Zięcina
deepsense.ai

Christopher Galias
deepsense.ai, Jagiellonian University

Silviu Homoceanu
Volkswagen AG, Wolfsburg, Germany

Henryk Michalewski
University of Warsaw

Abstract

We use synthetic data and a reinforcement learning algorithm to train a system controlling a full-size real-world vehicle in a number of restricted driving scenarios. The driving policy uses RGB images as input.

We analyze how design decisions about perception, control and training impact the real-world performance.

1   Introduction

This work focuses on verification whether using synthetic data from a simulator it is possible to obtain a driving system which can be deployed in a real car. Our policies were trained end-to-end using a reinforcement learning (RL) algorithm and confirmed to be useful. We run tests on a full-sized passenger vehicle with state-of-the-art equipment required for Level 4 autonomy.

A number of design decisions were made, taking into consideration the restriction of business environments. In particular we use mostly synthetic data, with labelled real-world data appearing only in the training of the segmentation network. We also decided to use only the RGB input provided by a single camera.

The driving policy is evaluated only concerning its real-world performance on multiple scenarios outlined in Section 3. To complete a scenario, the driving agent needs to execute from 250 to 700 actions at 10 Hz at speed varying from 15 to 30 km/h. In some of our experiments, the learned controller outputs the steering command directly. In other, the controller outputs waypoints which are transformed to steering commands using a proprietary control system. In this work, we decided to limit intermediate human-designed or learned representations of the real world only to semantic segmentation. The semantic segmentator used in our system is the only component trained using the real-world data – its training process mixes real-world and synthetic images. Our driving policies are trained only in simulation and directly on visual inputs, understood as RGB images along with

their segmentation masks. The input contains also selected car metrics and a high-level navigation command inspired by [9].

Using reinforcement learning and RGB inputs was a conscious decision. The goal behind this choice was to answer the following research question: Is a system backed by the state-of-the-art RL methods able to learn driving in the end-to-end fashion?

In particular, is it able to acquire an intermediate representation of a scene, which is transferable from simulation to the real world? We note that being able to train end-to-end is desirable as it reduces human engineering effort. Even more importantly, it also eliminates errors arising when gluing a heterogeneous system consisting in of separate perception and control modules.

A major difficulty related to learning in simulation was stated in Section 5 of [14]: "when using a realistic simulator, an extensive hyperparameters search becomes infeasible". We use the same realistic simulator as in [14] — CARLA, based on Unreal Engine 4. In order to alleviate the difficulty related to the time consuming training we implemented a parallelized training architecture inspired by IMPALA [15], Ape-X [19], OpenAI Five [30], Horovod [42], DBA3C [1], see also recent work [49]. With our current infrastructure and parallelization methods, in all our experiments we generated as much as 100 years of simulated driving experience. To verify whether synthetic data from a simulator helps in improving driving skills we conducted the following experiments which constitute the main contribution of this work:

1. In simulation: we verify the influence of visual randomizations on transfer between different scenarios in simulation; results are summarized in Section 4.1.

2. In real-world: we test 10 models listed in Table 1 on 9 driving scenarios. In total we report results gathered over more than 400 test drives. See Section 4.2 for a detailed description.

model	description
CONTINUOUS-PLAIN	base experiment
CONTINUOUS-LOW-RAND	experiment using smaller number of randomizations
DISCRETE-PLAIN	model using discrete actions
CONTINUOUS-REG	experiment with additional l_2 regularization
DISCRETE-REG	analog of DISCRETE-PLAIN with additional l_2 regularization
SEMSEG-ONLY	model with semantic segmentation as only visual input
WAYPOINTS-DISCRETE	model driving with waypoints
AUXILIARY-DEPTH	model predicting depth as auxiliary task
DYNAMICS-RAND-FFW	feed-forward model trained with dynamics randomizations
DYNAMICS-RAND-RNN	model with memory trained with dynamics randomizations

Table 1

In Section 4.3 we describe two failure cases and in Section 4.4 we assess a proxy metric potentially useful for offline evaluation of models. We provide recordings from 9 autonomous test drives on `https://bit.ly/2k8syvh`. The test drives are in correspondence with the scenarios listed in Figures 1.

2 Related work

Synthetic data and real-world robotics   Synthetic images were used in the ALVINN experiment [33]. In [38] was proposed a training procedure for drones and in [21, 32, 31, 47, 30] were proposed experiments with robotic manipulators where training was performed using only synthetic data. Progressive nets and data generated using the MuJoCo engine [48] were used in [36] to learn policies in the domain of robot manipulation. A driving policy for a one-person vehicle was trained in [3]. The policy in [3] is reported to show good performance on a rural road and the training used mostly synthetic data generated by Unreal Engine 4. Our inclusion of segmentation as described in Section 3 is inspired by sim2real experiments presented in [22]. Visual steering systems inspired by [38] and trained using synthetic data were presented in [39, 37].

Synthetic data and simulated robotics　Emergence of high-quality general purpose physics engines such as MuJoCo [48], along with game engines such as Unreal Engine 4 and Unity, and their specialized extensions such as CARLA [14] or AirSim [43], allowed for creation of sophisticated photo-realistic environments which can be used for training and testing of autonomous vehicles. A deep RL framework for autonomous driving was proposed in [40] and tested using the racing car simulator TORCS. Reinforcement learning methods led to very good performance in simulated robotics, see for example solutions to complicated walking tasks in [18, 26]. In the context of CARLA, impressive driving policies were trained using imitation learning [9, 34], affordance learning [41], reinforcement learning [5], and a combination of model-based and imitation learning methods proposed in [34]. However, as stated in [3]: "training and evaluating methods purely in simulation is often 'doomed to succeed' at the desired task in a simulated environment" and indeed, in our suite of experiments described in Section 3 most of the simulated tasks can be relatively easily solved, in particular when a given environment is deterministic and simulated observations are not perturbed.

Reinforcement learning and real-world robotics　An survey of various applications of RL in robotics can be found in [10, Section 2.5]. The role of simulators and RL in robotics is discussed in [44] in Section IV. In [38, 32, 31, 47, 30, 22, 3, 36] policies are deployed on real-world robots and training is performed mostly using data generated by simulators. [24] proposes a system with dynamics trained using real-world data and perception trained using synthetic data. Training of an RL policy in the TORCS engine with a real-world deployment is presented in [45].

3　Environment and learning algorithm

We use CARLA [14], an open-source simulator for autonomous driving research, based on Unreal Engine 4. CARLA features open assets, including seven built-in maps, 14 predefined weather settings, semantic segmentation, as well as camera and LIDAR sensors (in our experiments we only use RGB information). Camera position, orientation, and settings are customizable. CARLA also features multiple vehicles with different physical parameters. Two visual quality levels (LOW and EPIC) are supported; the latter implements visual features including shadows, water reflections, sun flare effect, and antialiasing.

Below we describe our experimental setup as used in the basic CONTINUOUS-PLAIN experiment. We varied its various elements in the other experiments. Details are provided in Section 4.2.

Simulated and real-world scenarios　The real-world deployments were tested on 9 scenarios, see Figure 1. The scenarios include turns and an overpass. In training, we assume that the simulated environment is static, without any moving cars or pedestrians, hence a number of a human driver interventions during test deployments in real traffic is unavoidable. We developed new CARLA-compatible maps which cover approximately 50% of the testing grounds used in real-world deployments. We use these maps along with maps provided in CARLA for training, with some scenarios reserved for validation only. In all scenarios agent's goal is to follow a route from start to finish. These routes are lists of checkpoints on the map: they are generated procedurally in CARLA maps and predefined in maps developed by us. The number of timesteps to complete a given scenario ranges from 250 to 700. Agents are expected to drive in their own lanes, but other traffic rules are ignored.

Rewards in simulation and metrics of the real-world performance　In simulation, the agent is rewarded for following a reference trajectory, which provides a dense training signal. The episode fails if the agent diverges from the trajectory more than 5 meters or collides with an obstacle. In the real world, for each scenario, we measure the percentage of distance driven autonomously (i.e. without human intervention); results are presented in Figures 2 and Figure 11 in Appendix. Since tests were made in an uncontrolled environment with other vehicles and pedestrians, the human driver was instructed to take over in all situations which were potentially risky. We measure also divergence from expert trajectories (Figure 3).

Actions　Vehicles are controlled by two values: throttle and steering. The throttle is controlled by a PID controller with speed set to a constant, and thus our neural network policies only command the steering. We explore various possibilities for actions spaces. Unless stated otherwise, in training the policy is modeled as a Gaussian distribution over the angle of the steering wheel. In evaluations we use the mean of the distribution.
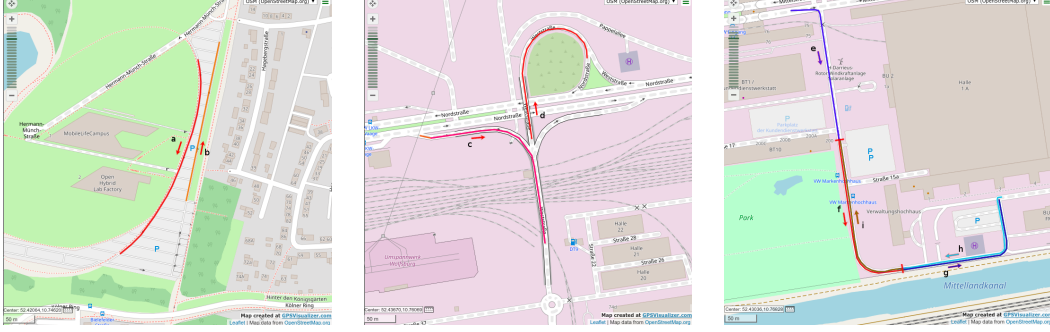
Figure 1: All real-world scenarios used in our experiments. Left map: (a) `autouni-arc`, (b) `autouni-straight`. Center map: (c) `factory_city-overpass*`, (d) `factory_city-overpass_exit`. Right map: (e) `factory_city-tunnel-bt10*`, (f) `factory_city-bt10-u_turn`, (g) `factory_city-u_turn-sud_strasse`, (h) `factory_city-sud_strasse_u_turn*`, (i) `factory_city-u_turn-bt10*`. Scenarios marked with * were used for training in simulation.

Observations The observation of the agent consists of an RGB image from a single front camera which is downscaled to the resolution 134×84 pixels. The RGB observation is concatenated with its semantic segmentation and two car metrics: speed and acceleration. The camera position and orientation in simulation was configured to reflect the real-world setup. The agent is also provided with a high-level navigation command: lane follow, turn right/left or go straight.

Semantic segmentation The semantic segmentation model is trained in a supervised way separately from the reinforcement learning loop. We used the U-Net [35] architecture and synthetic data from CARLA, the Mapillary dataset [29] as well as real-world labeled data from an environment similar to the one used in test drives.

Domain randomizations Randomizations are considered to be an important element to achieve sim-to-real transfer (and robust polices in general), see eg. [31]. In our experiments we used the following list of visual randomizations: 10 weather settings (we used CARLA weather presets, which affect only the visual features of the environment), the simulation quality (we used LOW and EPIC from CARLA settings), camera input randomizations (we used a set of visual augmentations as adding gaussian noise, varying brightness, applying blur or cutout [11]). We recall that our policies are trained on multiple scenarios and different maps, which is also aimed to increase robustness. These randomizations, unless stated otherwise, are used in all experiments. We also evaluated using randomizations of dynamics in `DYNAMICS-RAND-RNN`, `DYNAMICS-RAND-FFW`; these experiments are detailed in the next section.

Network architecture RL policy is implemented using a neural network, see the overview of its architecture in Figure 7 in Appendix. As the feature extractor for the visual input (RGB and semantic segmentation) we use the network from [15]. Our choice was influenced by [7], where this network was shown to generalize well between different RL environments. Note that policy transfer between simulation and reality can be seen as a generalization challenge.

Learning algorithm We used OpenAI Baselines [12] `ppo2` (see Appendix C for training hyperparameters). Thanks to dense rewards the training in simulation was quite stable across models and hyperparameters. For deployments we have decided to use 1-4 models per experiment type, using roughly 100M synthetic frames per training (equivalent to about 115 days of simulated time).

System identification Inspired by the importance of system identification for sim-to-real transfer demonstrated in [46], we configured the CARLA simulator to mimic some values measured in the car used for deployment. These were the maximal steering angle and the time for a steering command to take effect.

4 Experiments

Our models have been evaluated both in simulation and in the reality, with much bigger focus on the latter. Below we provide detailed description of experiments conducted in both domains.
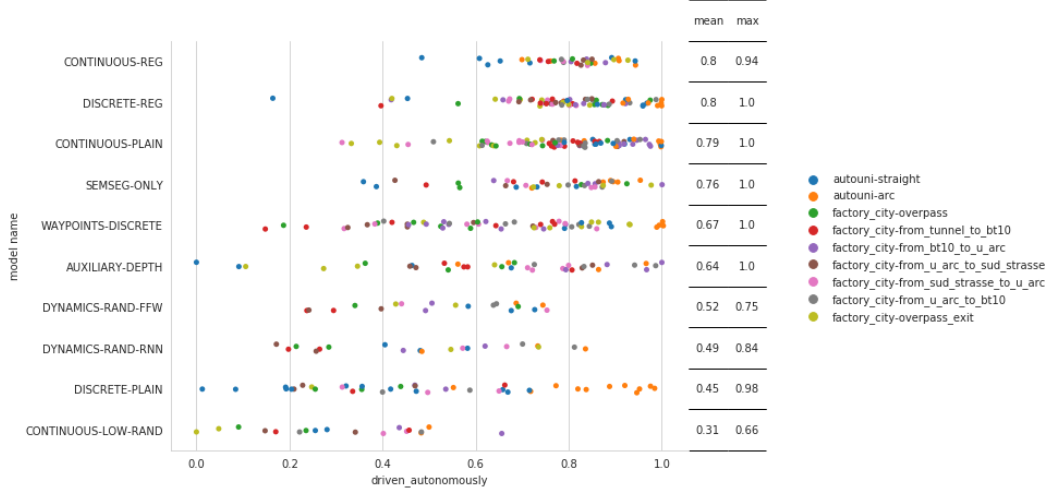

Figure 2: Summary of experiments with baselines across nine real-world scenarios. The columns to the right show the mean and max of autonomy (the percentage of distance driven autonomously). Models are sorted according to their mean performance. Print in color for better readability.
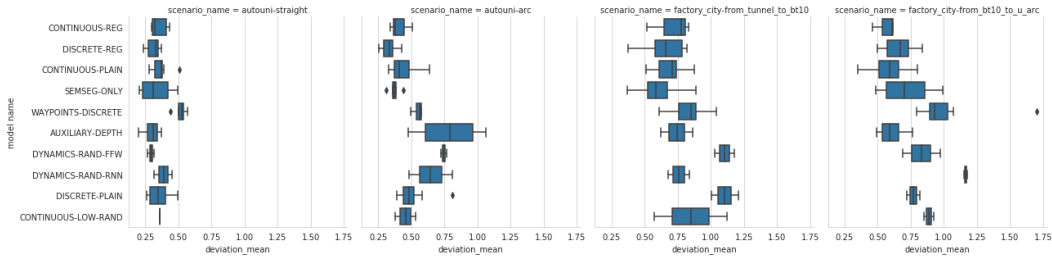

Figure 3: Average deviation of models from expert trajectories. Measurements based on GPS. The graphs for all scenarios can be found in Figure 12 in Appendix.
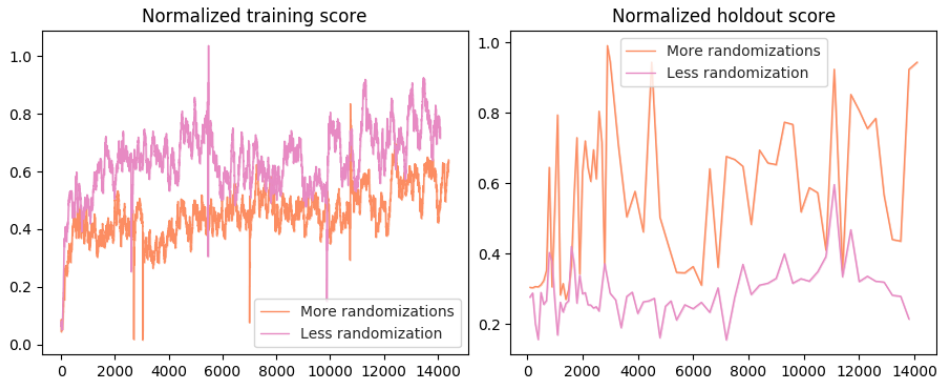
4.1 Experiment in simulation

Figure 4: Left: Episode scores obtained during training. Less randomization variant is easier and faster to train. Right: On a holdout town with holdout weather better results hold for a model trained with more randomization.

In this experiment, we measure in simulation how randomizations affect performance. To this end, we apply fewer randomizations then the set used throughout all other experiments. Precisely, we used only one weather setting, trained only on the LOW quality settings and did not augment the camera inputs. We conclude that the model trained with this standard set generalizes better to drive in a holdout town and with a holdout weather setting, see Figure 4.

4.2 Experiments in the real world

Figure 2 summarizes the performance of our models in terms of the percentage of distance *driven autonomously* in all scenarios. See Figure 11 in Appendix for a more fine-grained presentation. Below we describe in more details our all real-world experiments.
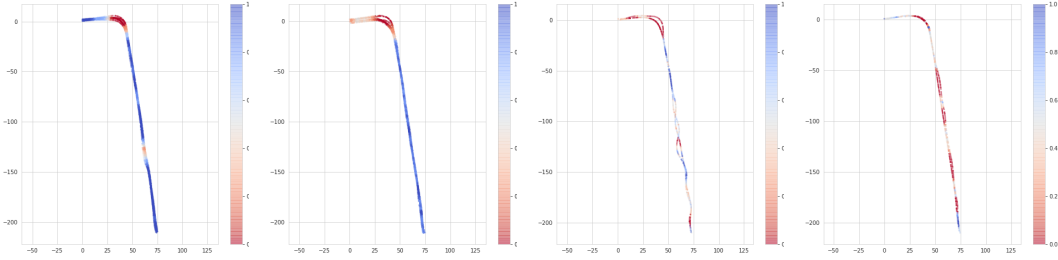

Figure 5: Qualitative comparison of autonomy with two good models `CONTINUOUS-PLAIN` and `DISCRETE-REG` along with two lagging models `DISCRETE-PLAIN` and `CONTINUOUS-LOW-RAND`. The graphs show the scale between blue (full autonomy in all experiments) and red (human assistance in all experiments). Graphs for other models and other routes are available in the supplementary materials as well as via the project webpage `http://bit.ly/34xh7z4`.

Base experiments The model `CONTINUOUS-PLAIN` exhibited a very good performance and serves as a strong baseline for comparisons with other variants discussed below. We aimed at creating a relatively simple model and training procedure. In other experiments we show that further simplifications deteriorate performance.

Analogously to the experiment in simulation described above, we have verified the impact of training with fewer randomizations on the real-world performance. As expected the resulting model `CONTINUOUS-LOW-RAND` performs significantly worse, being in fact the worse model tested.

Discrete action space This experiment aimed at measuring the impact of using discrete distribution for the action space. The steering angles were discretized into unevenly distributed atoms. More of them were placed around 0 to improve smoothness of driving without increasing the action space too much (viz. $[0., \pm0.01, \pm0.02, \pm0.03, \pm0.05, \pm0.08, \pm0.12, \pm0.15, \pm0.2, \pm0.25, \pm0.3, \pm0.4]$, values are in radians). During training the action was sampled from the distribution, while during evaluation a deterministic policy was outputing expected action (i.e. sum of the atoms values multiplied by their probabilities).

The resulting model `DISCRETE-PLAIN` performed disappointingly in the real world evaluations, mostly due to severe wobbling. We performed some more experiments with discrete actions, which results were mostly weak and are presented in Appendix E.

Regularization Improved performance in RL generalization when using regularization was reported in [7]. We evaluated using regularization in two experiments. In the first experiment, `DISCRETE-REG`, we fine-tuned `DISCRETE-PLAIN` by further training: reducing policy's entropy and applying l_2 regularization. The resulting model behaved significantly better (for example the wobbling observed before almost disappeared). In the second experiment, the performance of the continuous model trained with regularization - `CONTINUOUS-REG` - is only slightly improved over `CONTINUOUS-PLAIN`.

Control via waypoints Following the approach presented in [28], we train model `WAYPOINTS-DISCRETE` to predict the next waypoint instead of steering. Given a waypoint, low-level

steering of the driving wheel is executed in order to reach this point. In simulation, it is realized by a PID controller while in the case of the real car, we use a proprietary control system. To ensure similar performance in simulation and reality, we limit the action space of the RL agent to waypoints reachable by both of the controllers. It consists of points within a radius of 5 meters of the car. The action space is discrete - potential waypoints are located every 5 degrees between -30 and 30, where 0 is the current orientation of the vehicle.

Following [28] we also used branched neural network architecture with separate heads for each of the high-level command (such as turn left or lane follow). In some scenarios we have observed better behaviour in multi-modal situations; we plan to investigate this in future work.

In contrast to the experiments with direct steering, the continuous version of this experiment - `WAYPOINTS-CONTINUOUS` - was weaker and exhibited strong wobbling, even in simulation.

Dynamics randomizations   In [31, 30] dynamics randomization is pointed as an important ingredient for a successful sim-to-real transfer. In order to verify this in our context we introduced randomization to the following aspects of the environment: target speed, steering response (including random multiplicative factor and bias), latency (the delay between observation and applying policy's response to it), and noise in car metrics observation (speed, acceleration, wheel angle). Dynamic randomization parameters are sampled once at the beginning of each training episode.

Both models trained with dynamics randomization - `DYNAMICS-RAND-RNN` and `DYNAMICS-RAND-FFW` - performed weakly during evaluation on the real car. Somewhat surprisingly, the feed-forward model `DYNAMICS-RAND-FFW` performed slightly better than `DYNAMICS-RAND-RNN` using GRU memory cell [6], even though intuitively, an agent with memory could infer the dynamics parameters at the beginning of the episode and utilize it for better driving. We intend to further evaluate the possibility of using the dynamics randomization for sim-to-real autonomous driving in future work. As a first step we will probably look for explanation of the above mediocre performance. We speculate that this might be due to poor alignment of our randomizations with the real-world requirements or overfitting when using high-capacity models with memory.

Auxiliary depth prediction   Auxiliary tasks are an established method of improving RL training, see e.g. [20]. Following that, the model `AUXILIARY-DEPTH` apart for the policy predicts depth. The depth prediction is learned in a supervised way, along with the RL training. This auxiliary task slightly speeds the training in simulation. However in real-world evaluations, it does not improve over the baseline experiments.

Segmentation only   Similarly to [28] we test hypothesis that semantic segmentation is a useful common representation space for the simulation to real transfer. Our `SEMSEG-ONLY` model takes only segmentation as input, and performs only slightly weaker than baseline.

4.3   Selected failure cases

Besides major design decisions as described in the previous sections there is a number of small tweaks and potential pitfalls. Below we present two examples, which we find illustrative and hopefully useful for other researches and practitioners.

4.3.1   Single-line versus double-line road markings

In initial experiments we have used CARLA's TOWN1 and TOWN2 maps which feature only double-line road markings. When evaluated on real-world footage, such a policy was not sensitive to single-line road markings, whereas it was sensitive to double-line road markings in simulation.

This problem was fixed after introducing our custom CARLA maps which feature a single-line road markings.

4.3.2   Bug in reward function resulting in driving over the curb

Our reward function includes a term which incentives the agent to follow waypoints on the scenarios routes. The waypoints are connected by straight lines. In one of our scenarios we applied a waypoints pruning procedure, which turned out to be too aggressive. The line in which connected two consecutive points went over a curb on a bending of the road. We observed that only after doing the real-world test, where a similar behaviour occurred. This brings a few valuable, in our opinion,

lessons. Firstly, it calls for excessive debugging in simulation (which in fact we applied afterwards). Secondly, it illustrates well-known tendency of RL methods to overfit to idiosyncrasies of the reward signal design. Thirdly, it calls for developing methods with stronger generalization properties. Even though our model was able to correctly follow many other bends both in simulation and reality, when faced with the bend incorrectly modeled in simulation it decided to drive over the curb. Somewhat funny in this case our system overcame the sim-to-real gap and transferred the buggy behaviour.

4.4 Offline models evaluation

A fundamental issue in sim-to-real experiments is that good performance in simulation does not necessarily transfer to real-world. It is aggravated by the fact that real-world testing is costly both in time and resources. Inspired by [8] we introduced a proxy metric, which can be calculated offline and correlates with real-world evaluations. Namely, for seven scenarios with prefix `factory_city` we obtained a human reference drive. Frame by frame, we compared the reference steering with the one given by our models calculating the mean square error, mae. We observe a clear trend, see Figure 6.

Figure 6: Dependence of mean $driven\_autonomously$ metric on their mae with the reference drives. Models using waypoints are not included due to the different action space.

While this result is still statistically rather weak, we consider it to be a promising future research direction. We present an additional metric (F_1) in the Appendix D.
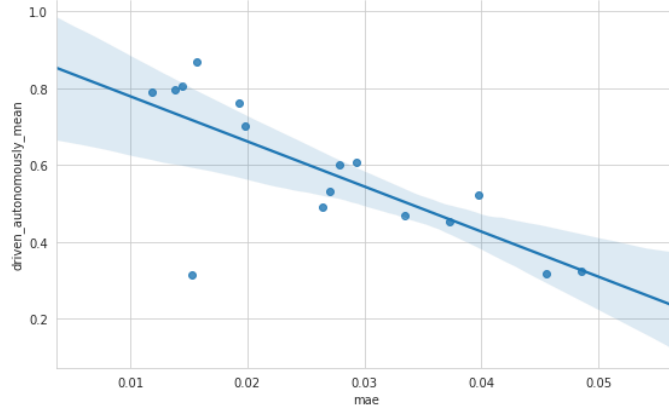
5 Conclusions and future work

We presented an overview of a series of experiments intended to train an end-to-end driving policy using the CARLA simulator. Our policies were deployed and tested on a full-size car exhibiting substantial level of autonomy in a number of restricted driving scenarios.

The current results let us to speculate about the following promising directions: using more regularization, control via waypoints and using off-line proxy metric. While we obtained poor results with memory-augmented architectures, we plan to investigate the topic further.

We also consider other training algorithms which use a replay buffer such as V-trace [15] and SAC [17]. The asymmetric actor-critic architecture presented in [32] and a generator-discriminator pair similar to the one in [4] can be also beneficial for training of driving polices. Another interesting and challenging direction is integration of an intermediate representation layer — for example a 2D-map or a bird's-eye view, as proposed in [5, 34, 13, 2]. Focusing RL training on fragments of scenarios with the highest uncertainty, see, e.g., [25] might improve driving stability. Integration of model-based methods similar to [27, 23] would be a desirable step towards better sample efficiency.

6 Acknowledgements

References

[1] Igor Adamski, Robert Adamski, Tomasz Grel, Adam Jedrych, Kamil Kaczmarek, and Henryk Michalewski. Distributed deep reinforcement learning: Learn how to play atari games in 21 minutes. In *High Performance Computing - 33rd International Conference, ISC High Performance 2018, Frankfurt, Germany, June 24-28, 2018, Proceedings*, pages 370–388, 2018.

[2] Mayank Bansal, Alex Krizhevsky, and Abhijit S. Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. In *Robotics: Science and Systems XV, University of Freiburg, Freiburg im Breisgau, Germany, June 22-26, 2019.*, 2019.

[3] Alex Bewley, Jessica Rigley, Yuxuan Liu, Jeffrey Hawke, Richard Shen, Vinh-Dieu Lam, and Alex Kendall. Learning to drive from simulation without real world labels. In *International Conference on Robotics and Automation, ICRA 2019, Montreal, QC, Canada, May 20-24, 2019*, pages 4818–4824, 2019.

[4] Konstantinos Bousmalis, Alex Irpan, Paul Wohlhart, Yunfei Bai, Matthew Kelcey, Mrinal Kalakrishnan, Laura Downs, Julian Ibarz, Peter Pastor, Kurt Konolige, Sergey Levine, and Vincent Vanhoucke. Using simulation and domain adaptation to improve efficiency of deep robotic grasping. In *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018*, pages 4243–4250, 2018.

[5] Jianyu Chen, Bodi Yuan, and Masayoshi Tomizuka. Model-free deep reinforcement learning for urban autonomous driving. *CoRR*, abs/1904.09503, 2019.

[6] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734, 2014.

[7] Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In *International Conference on Machine Learning*, pages 1282–1289, 2019.

[8] Felipe Codevilla, Antonio López, Vladlen Koltun, and Alexey Dosovitskiy. On offline evaluation of vision-based driving models. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XV*, pages 246–262, 2018.

[9] Felipe Codevilla, Matthias Müller, Alexey Dosovitskiy, Antonio López, and Vladlen Koltun. End-to-end driving via conditional imitation learning. *CoRR*, abs/1710.02410, 2017.

[10] Marc Peter Deisenroth, Gerhard Neumann, and Jan Peters. A survey on policy search for robotics. *Foundations and Trends in Robotics*, 2(1-2):1–142, 2013.

[11] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

[12] Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov. OpenAI Baselines. https://github.com/openai/baselines, 2017.

[13] Nemanja Djuric, Vladan Radosavljevic, Henggang Cui, Thi Nguyen, Fang-Chieh Chou, Tsung-Han Lin, and Jeff Schneider. Motion prediction of traffic actors for autonomous driving using deep convolutional networks. *CoRR*, abs/1808.05819, 2018.

[14] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. *arXiv oloreprint arXiv:1711.03938*, 2017.

[15] Lasse Espeholt, Hubert Soyer, Rémi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: scalable distributed deep-rl with importance weighted actor-learner architectures. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 1406–1415, 2018.

[16] Samuel Greydanus, Anurag Koul, Jonathan Dodge, and Alan Fern. Visualizing and understanding atari agents. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 1787–1796, 2018.

[17] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 1856–1865, 2018.

[18] Nicolas Heess, Dhruva TB, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, S. M. Ali Eslami, Martin A. Riedmiller, and David Silver. Emergence of locomotion behaviours in rich environments. *CoRR*, abs/1707.02286, 2017.

[19] Dan Horgan, John Quan, David Budden, Gabriel Barth-Maron, Matteo Hessel, Hado van Hasselt, and David Silver. Distributed prioritized experience replay. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[20] Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z. Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[21] Stephen James, Andrew J. Davison, and Edward Johns. Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task. *CoRR*, abs/1707.02267, 2017.

[22] Stephen James, Paul Wohlhart, Mrinal Kalakrishnan, Dmitry Kalashnikov, Alex Irpan, Julian Ibarz, Sergey Levine, Raia Hadsell, and Konstantinos Bousmalis. Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 12627–12637, 2019.

[23] Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H. Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, Ryan Sepassi, George Tucker, and Henryk Michalewski. Model-based reinforcement learning for atari. *CoRR*, abs/1903.00374, 2019.

[24] Katie Kang, Suneel Belkhale, Gregory Kahn, Pieter Abbeel, and Sergey Levine. Generalization through simulation: Integrating simulated and real data into deep reinforcement learning for vision-based autonomous flight. In *International Conference on Robotics and Automation, ICRA 2019, Montreal, QC, Canada, May 20-24, 2019*, pages 6008–6014, 2019.

[25] Alex Kendall, Vijay Badrinarayanan, and Roberto Cipolla. Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. In *British Machine Vision Conference 2017, BMVC 2017, London, UK, September 4-7, 2017*, 2017.

[26] Łukasz Kidziński, Sharada Prasanna Mohanty, Carmichael F. Ong, Zhewei Huang, Shuchang Zhou, Anton Pechenko, Adam Stelmaszczyk, Piotr Jarosik, Mikhail Pavlov, Sergey Kolesnikov, Sergey Plis, Zhibo Chen, Zhizheng Zhang, Jiale Chen, Jun Shi, Zhuobin Zheng, Chun Yuan, Zhihui Lin, Henryk Michalewski, Piotr Milos, Blazej Osinski, Andrew Melnik, Malte Schilling, Helge Ritter, Sean F. Carroll, Jennifer Hicks, Sergey Levine, Marcel Salathé, and Scott Delp. Learning to run challenge solutions: Adapting reinforcement learning methods for neuromusculoskeletal environments. In Sergio Escalera and Markus Weimer, editors, *The NIPS '17 Competition: Building Intelligent Systems*, pages 121–153, Cham, 2018. Springer International Publishing.

[27] Kendall Lowrey, Aravind Rajeswaran, Sham M. Kakade, Emanuel Todorov, and Igor Mordatch. Plan online, learn offline: Efficient learning and exploration via model-based control. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

[28] Matthias Mueller, Alexey Dosovitskiy, Bernard Ghanem, and Vladlen Koltun. Driving policy transfer via modularity and abstraction. In *2nd Annual Conference on Robot Learning, CoRL 2018, Zürich, Switzerland, 29-31 October 2018, Proceedings*, pages 1–15, 2018.

[29] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bulò, and Peter Kontschieder. The mapillary vistas dataset for semantic understanding of street scenes. In *International Conference on Computer Vision (ICCV)*, 2017.

[30] OpenAI, Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *arXiv preprint arXiv:1808.00177*, 2018.

[31] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018*, pages 1–8, 2018.

[32] Lerrel Pinto, Marcin Andrychowicz, Peter Welinder, Wojciech Zaremba, and Pieter Abbeel. Asymmetric actor critic for image-based robot learning. In *Robotics: Science and Systems XIV, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, June 26-30, 2018*, 2018.

[33] Dean Pomerleau. ALVINN: an autonomous land vehicle in a neural network. In *Advances in Neural Information Processing Systems 1, [NIPS Conference, Denver, Colorado, USA, 1988]*, pages 305–313, 1988.

[34] Nicholas Rhinehart, Rowan McAllister, and Sergey Levine. Deep imitative models for flexible inference, planning, and control. *CoRR*, abs/1810.06544, 2018.

[35] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2015 - 18th International Conference Munich, Germany, October 5 - 9, 2015, Proceedings, Part III*, pages 234–241, 2015.

[36] Andrei A. Rusu, Matej Vecerík, Thomas Rothörl, Nicolas Heess, Razvan Pascanu, and Raia Hadsell. Sim-to-real robot learning from pixels with progressive nets. In *1st Annual Conference on Robot Learning, CoRL 2017, Mountain View, California, USA, November 13-15, 2017, Proceedings*, pages 262–270, 2017.

[37] Fereshteh Sadeghi. Divis: Domain invariant visual servoing for collision-free goal reaching. In *Robotics: Science and Systems XV, University of Freiburg, Freiburg im Breisgau, Germany, June 22-26, 2019.*, 2019.

[38] Fereshteh Sadeghi and Sergey Levine. CAD2RL: real single-image flight without a single real image. In *Robotics: Science and Systems XIII, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, July 12-16, 2017*, 2017.

[39] Fereshteh Sadeghi, Alexander Toshev, Eric Jang, and Sergey Levine. Sim2real viewpoint invariant visual servoing by recurrent control. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 4691–4699, 2018.

[40] Ahmad El Sallab, Mohammed Abdou, Etienne Perot, and Senthil Yogamani. Deep reinforcement learning framework for autonomous driving. *Electronic Imaging*, 2017(19):70–76, Jan 2017.

[41] Axel Sauer, Nikolay Savinov, and Andreas Geiger. Conditional affordance learning for driving in urban environments. In *2nd Annual Conference on Robot Learning, CoRL 2018, Zürich, Switzerland, 29-31 October 2018, Proceedings*, pages 237–252, 2018.

[42] Alexander Sergeev and Mike Del Balso. Horovod: fast and easy distributed deep learning in tensorflow. *arXiv preprint arXiv:1802.05799*, 2018.

[43] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics, Results of the 11th International Conference, FSR 2017, Zurich, Switzerland, 12-15 September 2017*, pages 621–635, 2017.

[44] Niko Sünderhauf, Oliver Brock, Walter J. Scheirer, Raia Hadsell, Dieter Fox, Jürgen Leitner, Ben Upcroft, Pieter Abbeel, Wolfram Burgard, Michael Milford, and Peter Corke. The limits and potentials of deep learning for robotics. *I. J. Robotics Res.*, 37(4-5):405–420, 2018.

[45] Bowen Tan, Nayun Xu, and Bingyu Kong. Autonomous driving in reality with reinforcement learning and image translation. *CoRR*, abs/1801.05299, 2018.

[46] Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. In Tom Howard Hadas Kress-Gazit, Siddhartha Srinivasa and Nikolay Atanasov, editors, *Robotics: Science and System XIV*, 2018.

[47] Josh Tobin, Lukas Biewald, Rocky Duan, Marcin Andrychowicz, Ankur Handa, Vikash Kumar, Bob McGrew, Alex Ray, Jonas Schneider, Peter Welinder, Wojciech Zaremba, and Pieter Abbeel. Domain randomization and generative models for robotic grasping. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2018, Madrid, Spain, October 1-5, 2018*, pages 3482–3489, 2018.

[48] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2012, Vilamoura, Algarve, Portugal, October 7-12, 2012*, pages 5026–5033, 2012.

[49] Marin Toromanoff, Émilie Wirbel, and Fabien Moutarde. Is deep reinforcement learning really superhuman on atari? *CoRR*, abs/1908.04683, 2019.

A   Main algorithm and the network architecture

We used OpenAI Baselines [12] ppo2[1] with a custom policy that operates on multiple input tensors (see Figure 7).
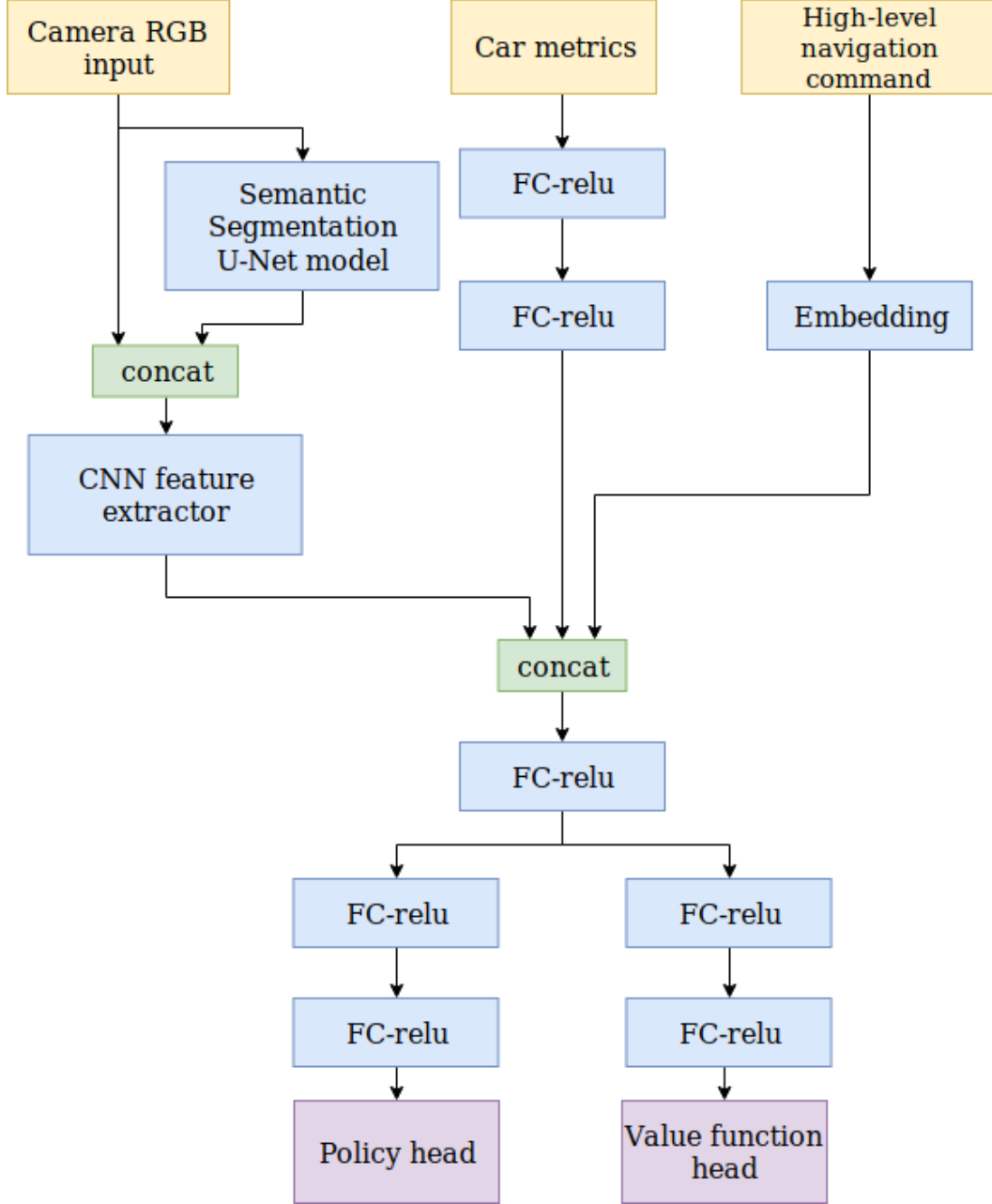

Figure 7: Network architecture.

B   Saliency maps

In order to gain some insight into the inner workings of the policy we generate saliency maps that visualize policy output sensitivity to different parts of camera input. Saliency maps were

---

introduced early in the project as a way to understand some results when deploying policies in real life (example videos from training with saliency maps can be seen in at the project webpage `http://bit.ly/34xh7z4`).

The first and simplest way to generate saliency map is to utilize the calculation graph framework and calculate the gradient of the output with respect to policy input. As said in [16]: "The simplest approach is to take the Jacobian with respect to the output of interest. Unfortunately, the Jacobian does not usually produce human-interpretable saliency maps". Such saliency maps are very noisy on a single-pixel level. Results can be seen on the left image in Figure 8.

In [16] is proposed an alternative way to calculate saliency maps. Their technique consists of blurring different patches of the input image (effectively removing some information from that patch) and measuring the output difference. This second technique generated more easily interpretable maps, but at the cost of generation time. The policy network needs to be applied hundreds of times to different perturbations of one image to generate the saliency maps. In our case, generating a video with saliency maps for a 17-minute piece of footage took about $\sim$2 hours on a machine with a GPU.

Finally, we decided to use a hybrid approach — calculating the gradient analytically, like in first approach, but using patches instead of single pixels, like in second approach. This results in a saliency map of acceptable quality with a fast generation time (a few minutes for a 17-minute piece of footage). An example can be seen on Figure 8.
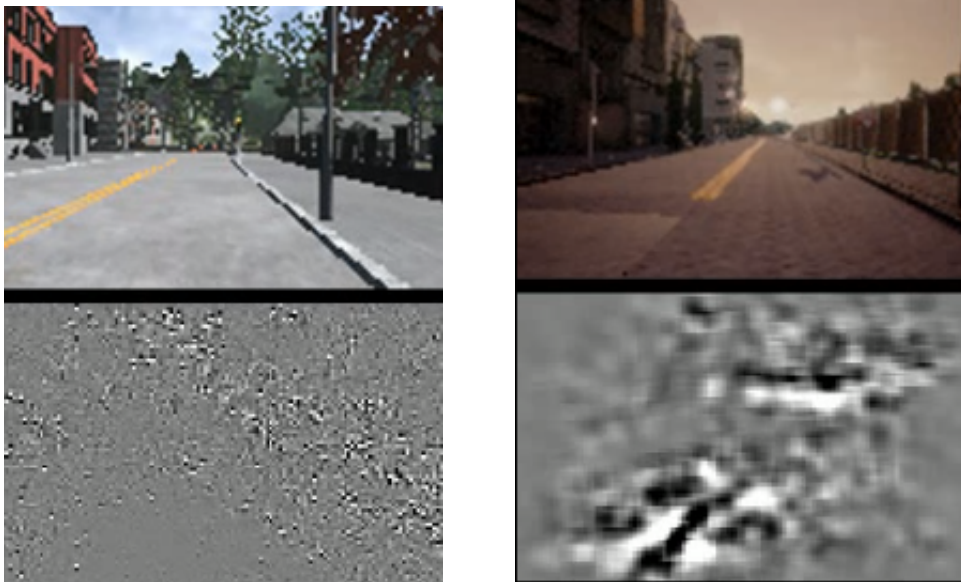

Figure 8: Left: Saliency map generated w.r.t. single pixels. Right: Saliency map generated w.r.t. 5×5 pixel patches. Each pixel encodes output sensitivity with regard to some input patch around it. White denotes positive gradient and black denotes negative gradient. Gray denotes gradient values close to zero. The saliency maps show that the network is most sensitive to road curbs and lane markings.

C PPO Hyperparameters

HYPERPARAMETER	VALUE	DESCRIPTION
LEARNING RATE	0.0003	
N-STEP	256	NUMBER OF STEPS USED FOR CALCULATING RETURNS
GAMMA	0.99	DISCOUNTING FACTOR
LAMBDA	0.95	LAMBDA IN GENERALIZED ADVANTAGE ESTIMATOR
ENTROPY COEFFICIENT	0.01	
CLIP RANGE	0.1	

D Offline models evaluation

On top of mae metric presented in Section 4.4 we have also introduced and evaluated another one: *average F1 score*. To compute the metric we again process frame by frame human reference drive and compare human action and output of the evaluated model. We classify requested steering wheel angle into one of three buckets: left, straight or right, if it is respectively less than -0.02 radian, between -0.02 and 0.02 radian or greater than 0.02 radian. For each of the buckets, we compute a F1 score between human reference action and the model output. The average of these three values is the final *average F1 score*. As you can see in Figure 9, this metric also seems to correlate with the model's real-world performance.
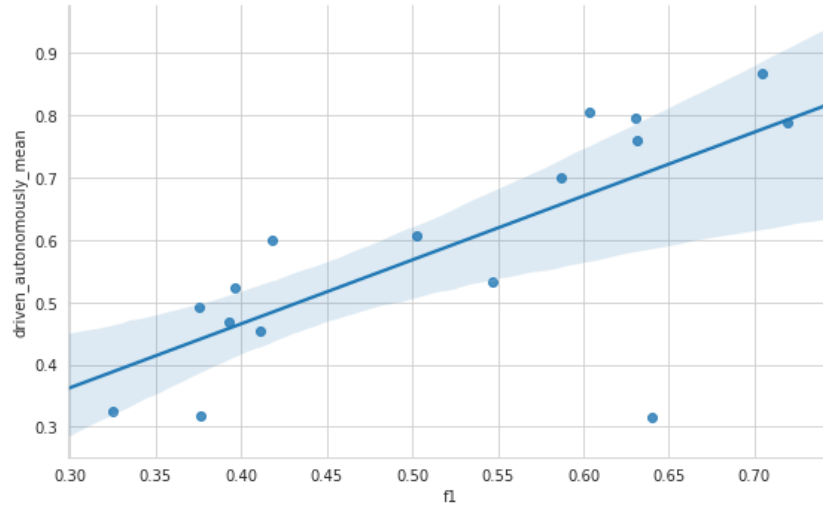

Figure 9: Dependence of mean $driven\_autonomously$ metric on their *average F1 score*. Models using waypoitns are not included due to the different action space.

E All models

In the table below we present all models evaluated in the real world. Their respective performance can be found in Figures 10 and 11.

| model	action space	comment
CONTINUOUS-PLAIN	continuous	base experiment
DISCRETE-PLAIN	discrete	
CONTINUOUS-LOW-RAND	continuous	fewer randomizations
DISCRETE-LOW-RAND	discrete	fewer randomizations
CONTINUOUS-REG	continuous	l_2 regularization
DISCRETE-REG	discrete	l_2 regularization
SEMSEG-ONLY	continuous	semantic segmentation as only visual input
DISCRETE-SEMSEG-ONLY	discrete	semantic segmentation as only visual input
WAYPOINTS-DISCRETE	discrete waypoints	
WAYPOINTS-CONTINUOUS	continuous waypoints	
AUXILIARY-DEPTH	continuous	auxiliary task: depth prediction
DISCRETE-AUX-DEPTH	discrete	auxiliary task: depth prediction
DYNAMICS-RAND-FFW	continuous	feed-forward model trained with dynamics randomizations
DYNAMICS-RAND-RNN	continuous	model with memory trained with dynamics randomizations
DISCRETE-DYNAMICS-RAND-FFW	discrete	feed-forward model trained with dynamics randomizations
DISCRETE-DYNAMICS-RAND-RNN	discrete	model with memory trained with dynamics randomizations

Table 2
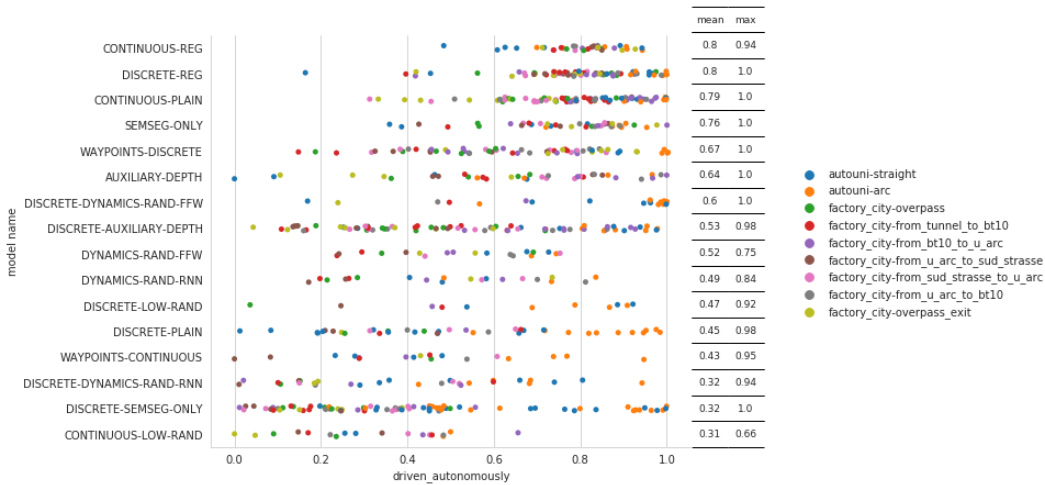
Figure 10: Summary of experiments with baselines across nine scenarios. The columns to the right show the mean and max of autonomy (the percentage of distance driven autonomously). Models are sorted according to their mean performance. Print in color for better readability.
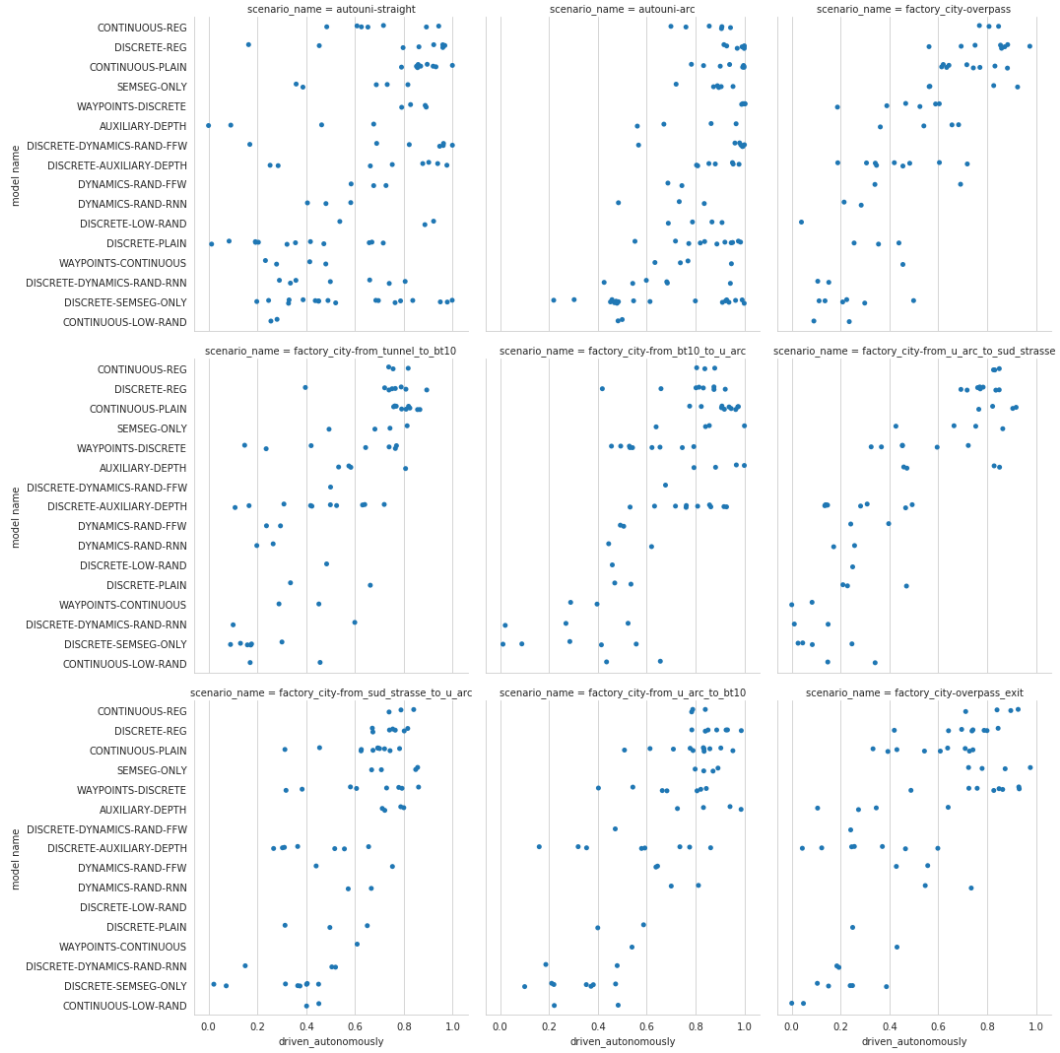
Figure 11: Summary of experiments across 9 scenarios with baselines. Each subfigure represents performance for a given deployment scenario.
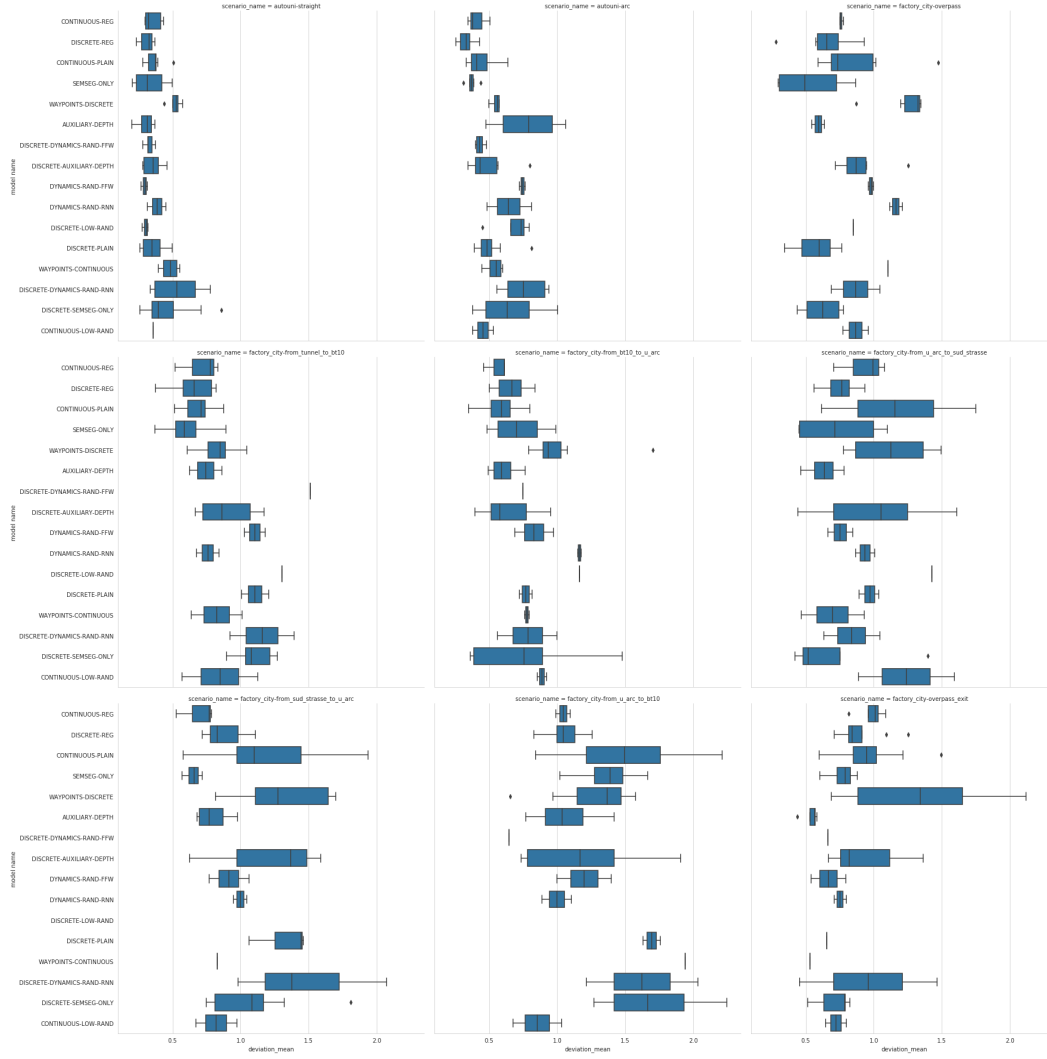
Figure 12: Average deviation of all models from expert trajectories. Measurements based on GPS.