CarBench v0.5: Benchmarks for Autonomous Driving

Michael Emanuel [1] **Hari Kothapalli** [2] **Bryan Lee** [2] **Tasha Schoenstein** [3]

Abstract

We used the Deepdrive framework provided by Voyage to benchmark autonomous driving agents on one task, making an unprotected left turn (ULT). Agents were benchmarked on task performance (score awarded according to contest rules) and average time to take an action. Using Deepdrive, we configured a benchmarking framework which successfully used the Deepdrive environment simulation to test multiple algorithms against custom benchmarking software we integrated with Deepdrive's existing simulator. This benchmarking framework may be extended to different tasks, learning algorithms, driving environments, and benchmarking metrics. We also attempted to get a high score in Deepdrive's ULT contest by training our own agents and successfully achieved second and third place on the Deepdrive ULT challenge leaderboard, beating Deepdrive's baseline. Based on our benchmarking metrics, the PID controller earns a higher average reward than the trained MNet-2 and PPO agents in the ULT task.

1 Introduction

While self driving car research has existed for decades (Kanade et al., 1986; Pomerleau, 1989), over the last few years, this research has exploded. Some recent research has focused on studying autonomous driving in specific environments, such as off-road environments (Pan et al., 2018). Other research has emphasized particular types of control algorithms, including imitation learning algorithms (Pan et al., 2018), end-to-end learning algorithms using neural networks (Bojarski et al., 2016), and algorithms that combine traditional approaches like model predictive control with learning (Drews et al., 2017). Other research has investigated how we might approach the safety problems resulting from uncertainty in autonomous driving (McAllister et al., 2017).

Despite the fact that this research has become widespread in both academia and industry, benchmarks for comparing different autonomous driving projects are limited. While industry regularly uses miles per disengagement to compare performance, this is far from sufficient. Additionally, while competitions such as the DARPA Urban Challenge might be thought of as real world benchmarks, software benchmarks that allow for comparison of different approaches in simulation are important for facilitating research.

[1]School of Engineering and Applied Sciences, Harvard University, Cambridge, MA, USA [2]Harvard College, Harvard University, Cambridge, MA, USA [3]Department of the History of Science, Harvard University, Cambridge, MA, USA. Correspondence to: Michael Emanuel <mse999@g.harvard.edu>, Hari Kothapalli <hkothapalli@college.harvard.edu>, Bryan Lee <bryanlee@college.harvard.edu>, Tasha Schoenstein <tschoenstein@g.harvard.edu>.

Preliminary work. Do not distribute.

1.1 Motivation

The proposed benchmarking framework seeks to address multiple key issues in designing algorithms for autonomous driving. First, many reinforcement learning algorithms require a large number of examples to be trained. Applied to autonomous vehicles, these algorithms are largely infeasible to train in physical environments because of the need for negative examples and crashes. This means that a benchmarking software requires integration with a software driving simulator in order to train and test these algorithms.

Second, the benchmarking framework assists users in selecting among the many available control algorithms (e.g. imitation learning, reinforcement learning, use of control theory) by providing key metrics which can be used to compare the given algorithms. Indeed, this is particularly important in applications to autonomous vehicles because the reward scheme for autonomous driving is unclear. Whereas Atari games, for example, have set point rewards upon goal completion, autonomous driving involves many factors (e.g. mileage, passenger comfort, passenger safety) which are ambiguous. To address this, the proposed benchmarking framework provides access to many low-level features (e.g. g-force, angular acceleration, velocity, throttle, camera image, etc.) that the vehicle observes at each time step to track the vehicle's performance with respect to many potential optimization factors.

To address these challenges, we propose an autonomous vehicle driving benchmarking framework based on Voyage's

Deepdrive simulator which allows users to compare multiple control algorithms in different test environments using a diverse set of low-level metrics.

1.2 Benchmarking Framework

An abstracted schematic of our proposed benchmarking framework is shown in Figure 1. The framework seeks to address the aforementioned challenges. The framework first incorporates a Deepdrive autonomous driving simulator, based off of an Unreal physics engine, to generate environments and train the autonomous driving algorithm. Additionally, Deepdrive engine is exposed as an OpenAI gym environment, which allows multiple algorithms from OpenAI baselines to be trained on the simulator. Finally, the benchmarking framework assesses the trained algorithms with multiple metrics to determine the driving quality.

1.3 Contributions

In developing this framework, we provided the following contributions.

Deepdrive Environment Configuration: We successfully configured the Deepdrive environment on both cloud based and local GPU servers. We tested algorithms both in local (i.e. one process for both the environment and the agents) and client/server modes (i.e. where the environment runs on one 'server' process and the agent runs on another 'client' process). Using the client/server mode, which is required for contest entries, we saved our agents as Docker images that we pushed to a private Docker repository to enter the Deepdrive contest leaderboard.

Algorithm Benchmarking: We developed a custom script which parses the Deepdrive extension of the OpenAI gym to collect low-level metrics for benchmarking at each time step (e.g. rewards, action-times, velocities, throttle).

Using a subset of these metrics (i.e. reward and action time), we benchmarked six agents on the left-hand turn task:

- **forward-agent:** the minimal forward-agent, which always steps on the accelerator
- **lazy-driver:** the hand tuned left turn that is at the top of the leaderboard
- **PID:** a hand built PID controller written in C++ and supplied by Voyage
- **MNet-2-baseline:** a pre-trained baseline model using an MNet-2 agent, supplied by Voyage
- **ppo-baseline:** a pre-trained baseline model using a PPO agent, supplied by Voyage
- **MNet-2-trained:** a second MNet-2 agent that we trained from scratch for 3 GPU hours

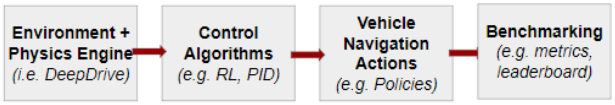

Figure 1. Components of the benchmarking stack. Building upon the Deepdrive framework, we deploy multiple agents (e.g. RL, PID), and compare the agents using our custom benchmarking code. We submitted our best performing agent and placed third on the Deepdrive challenge leaderboard, outperforming Deepdrive's agent.

As a proof-of-concept, we trained neural network agents built on the Mobilenet-v2 (MNet-2) architecture both locally and on the cloud.

Top Deepdrive Leaderboard Performance: We trained a MNet-2 agent that has some success on the left turn task, and submitted an entry for this agent that took third place on the leaderboard, outperforming Deepdrive's baseline algorithm. This trained algorithm was only outperformed by ungeneralizable, brute-force left turn algorithms. While this agent does not consistently navigate the turn safely, it demonstrates meaningful progress in learning to drive. We also submitted an entry that achieves second place on the leaderboard by imitating a technique that takes advantage of low domain randomization to memorize a correctly timed turn.

1.4 Software and Data

All code for this project is available at our public GitHub repository: memanuel/Deepdrive. This is a fork of the original Deepdrive project. Our modifications primarily include code for benchmarking. Data necessary to train the agents was supplied by Voyage and is available by AWS. Voyage includes detailed instructions to obtain the dataset in their online tutorial.

2 Related Work

There are a few main categories of related research in benchmarking, including benchmarking for UAVs and robots, perception/mapping/planning, and autonomous vehicles.

2.1 Benchmarks: UAVs and Robots

A significant body of work has aimed to develop benchmarks for unmanned aerial vehicles (UAVs) and other robots. In one related project, AirLearning, the researchers developed a benchmarking system that looks at performance and energy use in a variety of simulated environments for reinforcement learning-based algorithm-hardware systems for UAVs (Krishnan et al., 2019). The AirLearning project

added both useful functionalities like the ability to experiment with hardware and new methods for evaluating the algorithms using ideas about energy.

The AirLearning project used intuitions about energy simulation developed in the MAVBench project (Boroujerdian et al., 2019). As with AirLearning, for the MAVBench project, researchers developed benchmarks for UAVs by focusing on micro aerial vehicles. Additionally, the MAVBench project developed a simulation system that, along with the benchmarks, facilitated the analysis and design of micro aerial vehicles in ways that address bottlenecks related to power, performance, or computation and in ways that encourage software-hardware co-design.

In another work, the RoboBench project built a major benchmark suite and simulator for a variety of robotics applications, including both UAVs and vaguely humanoid robots (Weisz et al., 2016). While RoboBench approached the problem using a more end-to-end methodology than earlier benchmarking projects, it did not include benchmarks for self-driving cars.

2.2 Benchmarks: Perception, Mapping, and/or Planning

A significant amount of benchmarking research has been developed for subsets of the autonomous vehicle algorithm stack, such as mapping, planning, and/or perception. The SLAMBench project, for example, developed performance, accuracy, and energy efficiency benchmarks for SLAM as it is used on a variety of hardware for a variety of applications (Nardi et al., 2015). Older benchmarks for SLAM had focused on performance and accuracy without attention to energy efficiency (Sturm et al., 2012).

Other self-driving car benchmarks have focused on the problem of benchmarking motion planning for automated driving. The CommonRoad project, for example, focused on providing identifiers for use in benchmarking for motion planning in self-driving cars rather than performance metrics, such as those provided by SLAMBench (Althoff et al., 2017). The earlier MoVeMA project focused on benchmarking algorithms for motion for both self-driving vehicles and robots (Calisi et al., 2008). Other research developed benchmarks for traffic control where the traffic includes some vehicles controlled via reinforcement learning and some human-controlled vehicles (Vinitsky et al., 2018).

Another group of benchmarks have focused on problems of perception for autonomous vehicles. In particular, there is a robust group of datasets for perception for autonomous driving that connect high-quality images to ground truth data (Geiger et al., 2012; Kondermann et al., 2016; Richter et al., 2017). Other researchers have also developed a UAV-specific benchmark suite and simulator that focused on visual tracking for UAVs (Mueller et al., 2016).

While benchmarking segments of the autonomous vehicle pipeline has some uses, there is now an increasing need for end-to-end benchmarking for autonomous driving. This project aims to contribute to the development of end-to-end benchmarks for autonomous cars.

2.3 Benchmarks: Self-Driving Cars

Relatively little research appears to have been done on benchmarking of self-driving cars. Perhaps the closest effort appears to be the DARPA Urban Challenge, which fulfilled some of the goals of benchmarking by defining specific tasks and testing a variety of autonomous vehicles against those tasks (Leonard et al., 2007; Bacha et al., 2008; Urmson et al., 2009). Real-world "benchmarking" of the sort carried out in the DARPA Urban Challenge is an incredibly useful form of benchmarking; however, building benchmarking software that allows for the comparison of algorithms for autonomous driving will facilitate comparison earlier on in research and development pipelines. Benchmarking software that operates in simulation also allows for the far less expensive testing of variations in the technical details of the hardware and sensors for self-driving cars.

3 Algorithm/System Description

3.1 Rationale in Selecting Deepdrive

This project uses the Deepdrive simulator from Voyage (Dee). A screenshot from the Kevindale map in the Deepdrive simulator is shown in Figure 2. Deepdrive includes a high-quality driving simulator that renders scenes in Unreal Engine 4 (UE4) with both simulated camera and LIDAR input. Deepdrive is exposed as an OpenAI environment, which allows machine learning agents to be trained using OpenAI gym and allows agents to run as clients against an environment server. Notably, Deepdrive's infrastructure was well documented, unlike some of the other simulators available. Due to our limited access to GPUs, we utilized cloud computing infrastructure offered by Paperspace which is pre-configured for machine learning development (pap). Uniquely, Deepdrive offers a competition which crowdsources algorithms to solve a difficult autonomous driving task. We competed in the unprotected-left-turn challenge.

In selecting simulation software, we considered alternatives to Deepdrive. Uber has developed open-source visualization software for autonomous vehicles (Chen et al., 2019). Unlike Deepdrive, Uber's system does not include autonomous agents or infrastructure to rapidly train autonomous agents, which is important for researcher adoption.

We also examined Microsoft's AirSim (Shah et al., 2017) as used by the MAVBench and AirLearning projects (Boroujer-

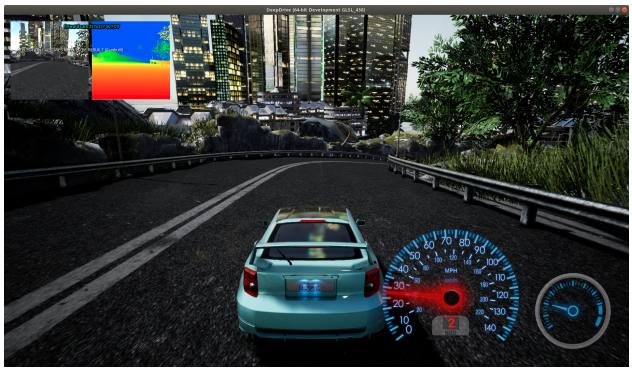

Figure 2. Example of Deepdrive driving simulator. The top left displays the camera and LIDAR sensor readings while the bottom right displays the car's velocity and throttle. The car's goal is to drive laps around the given track.

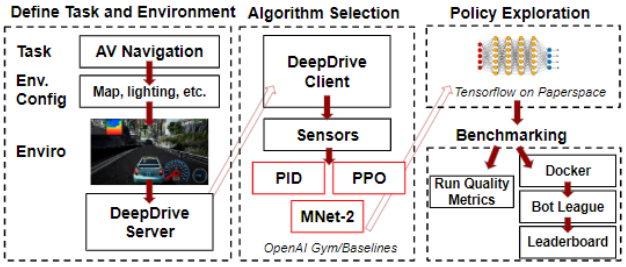

Figure 3. Example of Deepdrive driving simulator.

dian et al., 2019; Krishnan et al., 2019). While AirSim does support simulation of cars, our attempts to use AirSim by extending MAVBench or AirLearning had intractable technical problems. These problems included lack of access to appropriate hardware and code that relied on legacy versions of ROS and Unreal Engine. In particular, we attempted to install and run the AirLearning package on a modern GPU server running Ubuntu 18.04. This was an exercise in futility that consumed many hours with no tangible results.

Lastly, we rejected the possibility of using the infrastructure developed by Baidu's Apollo project (apo). The Apollo project requires significantly more hardware and is significantly more complicated to extend than in Deepdrive.[1]

3.2 Architecture and System Overview

Figure 3 illustrates the detailed workflow of our benchmarking infrastructure. The workflow can be segmented into four primary parts: defining the task and environment, algorithm selection, policy exploration, and benchmarking.

For the task of autonomous vehicle navigation, we configured used the default environment in the Deepdrive simulator. Notably, the Deepdrive simulator offers three different training maps and domain randomization (via lighting, camera resolution, and other factors) which provides a diverse set of environments.

Using the server and client training mode, the Deepdrive environment (i.e. server) connects to a client which implements one of multiple algorithms which were tested (e.g. PID, PPO, MNet-2). These algorithms were the among the baselines Deepdrive provides, allowing them to serve as a natural proof-of-concept for algorithm selection. Deepdrive's integration with OpenAI gym allows for extensibility of the set of algorithms to others in the OpenAI baseline. The Deepdrive server also provides sensor inputs (e.g. camera, LIDAR, velocity, etc.) to these algorithms for training. Subsequently, we trained a subset of these algorithms (i.e. MNet-2) on Voyage's publicly available autonomous driving data set using tensorflow on Paperspace.

Last, we utilized Deepdrive's existing sensor data to perform new benchmarking through driving quality metrics and submission to the Deepdrive leaderboard. To extract and display run-quality metrics, we modified Deepdrive's Open AI environment extension and extracted existing sensor readings the vehicle records at each time step and plotted them for the user's analysis. This provides a suite of metrics which users can track for each algorithm across all time steps. Notably, a subset of the metrics can be selected based on the user's individual use-case. An example of the displayed metrics for a single run of one of Deepdrive's baseline algorithm is shown in Figure 4.

In addition to establishing quality of driving metrics, we developed two agents to submit to Deepdrive's leaderboard for the unprotected-left-turn competition: a brute-force left turn agent, and a trained MNet-2 agent. In order to submit agents to the Deepdrive leaderboard (which is developed using the Bot League framework), the agents must be placed in Docker containers. To address this, we created a Docker repository, stored our trained agents as Docker images, pushed the agents to the remote Docker repository, and made a pull request to the central Deepdrive Bot League GitHub repository to make submissions. Figure 5 illustrates the performance of these submissions. The brute-force left turn agent is currently second on the leader board, out performing more generalizable solutions, as expected. Additionally, our trained MNet-2 agent placed third on the leaderboard, outperforming the Deepdrive baseline agent.

[1] Additional concerns arose from reports of malware/spyware embedded in some software produced by major Chinese software companies, including Baidu.

3.3 Challenges Faced

Popular culture holds that completing a marathon is hard (it is!), but it is arguably harder to complete an 18 week training program and reach the starting line in fit condition than it is to give it your all on race day. Machine learning problems can also exhibit this pattern where reaching the starting line is the hardest part of the task. In our case, reaching the starting line covered a number of unglamorous but critical steps. While the following information would normally be included in the discussion section of a formal paper, considering this project was conducted over a very limited time frame, we believe it is important to acknowledge the efforts that otherwise would not make it into this report.

First we had to identify a computing framework for simulation. After trying and failing to follow AirLearning, with a framework of Ubuntu 16.04, ROS Indigo, and AirSim, we followed a suggestion from Professor Reddi and investigated Deepdrive. This step consumed multiple work days with different team members trying to install packages on existing Ubuntu computers or trying to set up new virtual machines on laptops without the necessary hardware resources.

The next step to reach the starting line was to configure the necessary computing infrastructure. Two team members had access to computers with the necessary hardware including high end GPUs. Both of us needed to invest significant effort run Deepdrive on these machines. One of us reinstalled Ubuntu 18.04 from scratch on what had been an old Windows computer. The other spent 10 hours reconfiguring a headless GPU server so it could run remotely with native graphics provided by the NVIDIA GPU via OpenGL 4.6. Specifically, the previous configuration ran remotely with Remote Desktop; in that mode, graphics were provided by software emulation and the Deepdrive simulator would not run. While these methods were unsuccessful, we encountered Paperspace which offers pre-configured machine learning infrastructure on Ubuntu. Indeed, configuring the cloud service on Paperspace was relatively straightforward, but not cheap. While the pricing is reasonable, estimated costs for this project amount to about $200.

Once we had the Deepdrive environment up and running on our local and cloud servers, the starting line was in sight. By going through the tutorials, we were able to figure out the difference between the basic usage pattern, where one process hosts both the simulation and the agent, and the more sophisticated client / server mode of operation. While the documentation is excellent overall, this was one point that was not spelled out explicitly, and which we only figured out after a fair amount of trial and error. The next major challenge to reach the starting line was learning how to encapsulate both the server and client processes in Docker containers. Finally, we needed to follow a somewhat byzantine process to submit an entry to the leaderboard by forking a repository called Bot League and including a magic json file in it pointing to the image of our Docker image with an agent.

At this point, we might say we had reached the starting line; we were ready to run agents, train agents, and submit to the leaderboard. Our first major task was to extract the information we wanted for benchmarking. We modified the sim class (which implements the OpenAI gym environment API) to extract the relevant information we wanted (e.g. step-wise rewards, action times, velocity, acceleration) and prepared Python scripts to conduct analysis on this data for benchmarking across algorithms.

4 EXPERIMENTS

Our first round of experiments was to run the agents supplied by Voyage in different scenarios. Before we extracted quantitative benchmarking metrics, we ran the agents and observed their behavior. Driving is a complex task, and it is not at all straightforward to write a function that assigns a sensible score to a driving outing. This is in contrast to some other areas where RL has achieved notable success, including Go, Chess, and video games. While we can't readily write a scoring function, we as humans are very good at rating the skills of a driving agent by sitting in a car with them or watching a video of them drive.

We started with the minimal forward-agent, which always floors the accelerator, because it is the simplest agent and therefore easiest to get working. Needless to say, this is not a safe driving technique... Experiments with this agent reveal that the simulator does not handle collisions between the car and building elements realistically. While the simulator correctly recognizes that a collision has taken place, it also produces some comical output frames with the car driving right through translucent buildings before it shuts down.

Experiments with the hand tuned PID controller revealed that it is a respectable driving agent that can handle a number of the simple driving scenarios provided with the Deepdrive environment. The contest gave this agent poor scores because it heavily penalized it for having too much acceleration, but it was able to drive laps around the Kevindale map and make left turns with fewer crashes and less erratic behavior than either of the baseline neural network agents.

Experiments with the pre-trained baseline agents for both neural networks (MNet-2 and PPO) were disappointing to us. We've all seen videos of Waymo and Tesla vehicles successfully navigating simple driving tasks. These baseline agents were appallingly bad. We could only laugh as they swerved all over the lane before crashing into guard rails or parking meters in the absence of any traffic.

After our qualitative assessment of the agents, we turned

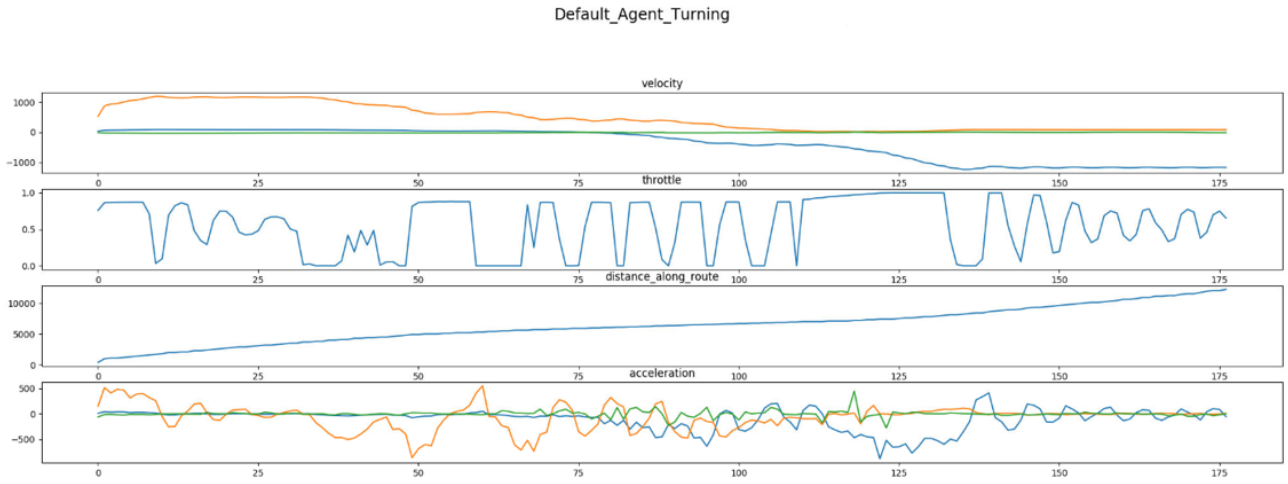

Figure 4. A sample of the metrics displayed via our benchmarking analysis. The image displays velocity, throttle, distance along route, and acceleration at each time step for the trained baseline agent from Deepdrive for a unique right turn task.

our attention to training the neural network based agents to learn basic driving. We trained both the MNet-2 and PPO agents. Both of these agents are based on the dagger approach, where a training episode is started from a random point in a real episode, and the agent is rewarded for imitating the expert demonstrator. Voyage provided 100 GB of training data, covering 8.2 hours of driving performed by what they described as an "Oracle" agent. Getting the training to run was straightforward; Voyage provided example commands in the documentation, and they worked immediately. Getting the agents to learn anything useful, on the other hand, was far more challenging.

Between the two agents, training the PPO agent proved to be almost hopeless. The PPO agent uses continuous control, while the MNet-2 agent uses a simpler discretized control scheme. After multiple attempts training for 2+ hours on an RTX 2080 Ti GPU, including one overnight session, the PPO agent was still laughably bad, so we gave up on it. In particular, the PPO agent repeatedly got stuck in low information training regimes for extended periods of time. For instance, it would crash against the guard rail, and remain stuck for the remaining duration of the episode. These results align with the in-class presentation by Alexandra Faust from Google Brain, where she discussed some of their difficulties in training PPO agents in continuous control spaces.

We had slightly better results with the MNet-2 agent. After training it for on the order of 2 hours, we had an agent that performed comparably to the baseline agent. This agent, however, was still far from being able to drive satisfactorily. Not only would a sane person refuse to ride in a car driven by this agent; even a generous driving instructor would terminate the session early and tell the student to practice on a simulator until they could stay in lane.

One of the nice features of Deepdrive and the TensorFlow back end it uses for training is that it makes it convenient to monitor training progress with Tensorboard. Figure 6 shows the MNet-2 agent's progress in matching the various controls and metrics during 2:45 of imitation learning.

Our next experiment was to attempt score well on the contest leaderboard. Early efforts were dominated by mastering the submission mechanics. We eventually figured out how to build a Docker image, push it to a repo on Docker Hub, and point our submission to the image of our driving agent. At this stage we noticed that the entry at the top of the leaderboard, from "lazy driver", used a very sneaky trick, essentially memorizing the speed with which to turn. While it doesn't generalize at all, its best run gets a high score. We made one entry with the goal of only getting a high place on the leaderboard using this same technique. More significantly, we trained an MNet-2 agent long enough that it was able to successfully complete the task some of the time. While this agent was unable to beat the scores of the two "sneaky" hand tuned agents, it achieved third place on the leaderboard, the highest spot of any agent that isn't grossly overfitting the small sample size. This agent also outperforms the submission generated by Deepdrive.

5 Results

It has been said that a picture is worth a thousand words. When it comes to understanding the performance of driving agents (human or autonomous), a movie may well be worth a thousand charts. We therefore begin with links to five

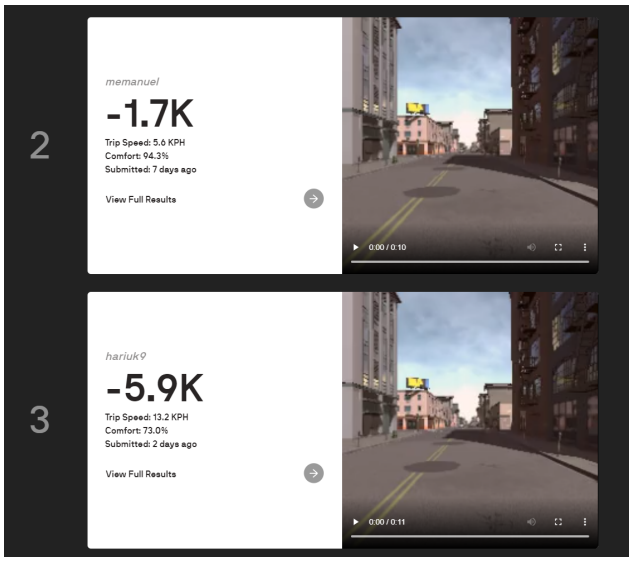

Figure 5. Current position on the Deepdrive leaderboard.

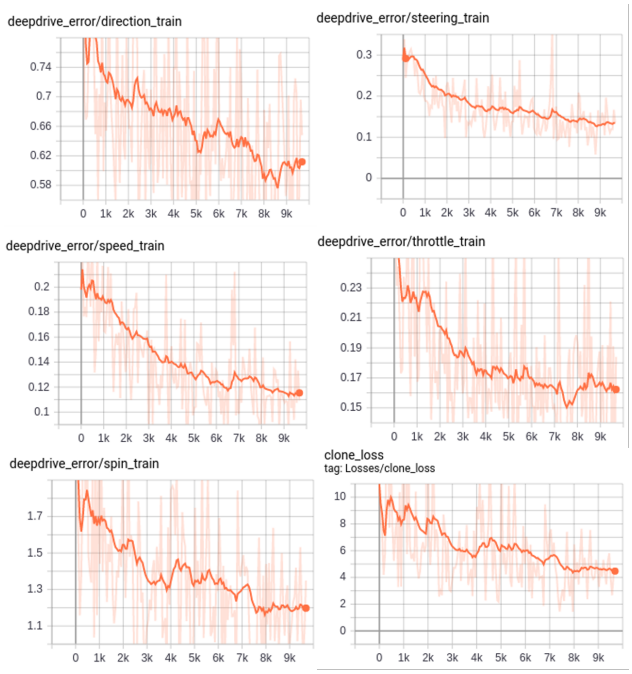

Figure 6. Progress training MNet-2 DAgger agent with imitation learning for 2:45

YouTube videos. These show five agents attempting the unprotected left turn task: forward agent (always on the accelerator), MNet-2 baseline agent, MNet-2 trained agnet, PPO baseline agent, and PID agent. These simulations were run on a local server with an NVIDIA GTX 2080 Ti GPU. Even this powerful server is unable to render the frames in real time, though it comes close. Most of the computation is being used to simulate the environment; the agents are able to keep up easily, though they don't drive very well...

- forward-agent
- MNet-2-baseline
- MNet-2-trained
- ppo-baseline
- PID

As mentioned earlier, of course the forward-agent can't drive safely. This video demonstrates how the simulator handles the scenario and the strange way it renders collisions with buildings. The MNet-2-baseline video shows the car immediately crashing into a parking meter without the slightest provocation. The MNet-2-trained video shows an agent that has been trained for an additional 3 GPU hours. This agent does a bit better than the baseline. Instead of crashing immediately into a parking meter, it makes it to the end of the block before slamming into a traffic light. The ppo-baseline agent might be generously described as a "deer in headlights." It just sits there, twitching between tiny taps on the accelerator and brakes. You can see the brake-lights flickering on and off. We stopped the video after a minute, but after two minutes this agent still has not made it up to the intersection. The PID video shows the agent successfully navigating the left turn in traffic. We are a bit perplexed that this agent gets a score of -15,000 on the leaderboard due to low driver comfort while agents that routinely crash get higher scores.

6 Conclusions

We've learned more about how to do future research on benchmarking autonomous driving agents than we've learned about the performance of different agents. Our first conclusion is that obtaining the necessary computing infrastructure is a nontrivial challenge, especially for a team working on a "pop-up" project without long term access to the same dedicated hardware resources. If a computing framework has been selected that requires a specific environment, e.g. the AirLearning environment with an older version of ROS, a cloud-based virtual machine can be a convenient alternative to a dedicated machine. If we were to do future research on a platform similar to AirLearning,

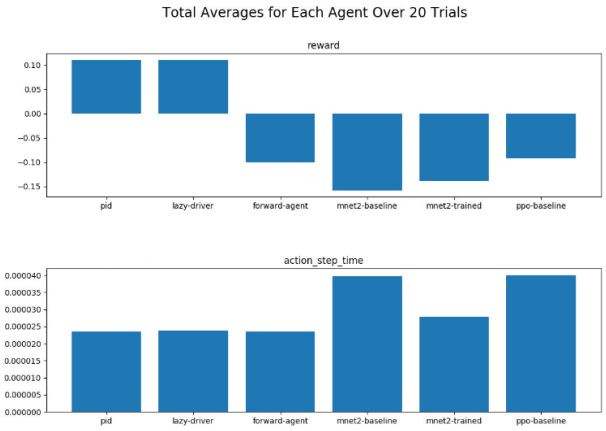

Figure 7. Average performance (reward and action time) of all tested agents across 20 iterations

we would invest the time to first port AirLearning to the latest versions of ROS and Ubuntu. This would make it significantly more feasible to use a dedicated machine.

If the goal of a future research project is to carefully measure the end to end performance of a particular autonomous driving system, including both its hardware and software, we conclude that the Deepdrive environment might be suitable, pending further testing. For an accurate simulation that accounts for specific hardware differences in different systems, either hardware in the loop or a very granular hardware simulator would be required. If you are willing to assume that your sensor data is consistent with the simulated camera and LIDAR output of the Deepdrive simulator, you might be able to get a HIL system working. The idea would be to run the Deepdrive environment on a powerful server, and run the proposed driving agent with its own native hardware. We have not tested that configuration here; instead, we've only run the environment and agents on the same high end computers. But in principle, there is no reason that the agent could not run on a separate computer. The only requirement is that the server and client communicate over an IP network. Based on the challenges we faced getting ROS working, our suggestion for the future CarBench project would be to first investigate whether a HIL system using Deepdrive could be made to work. If so, we would recommend it as an alternative to ROS / AirSim that is more accessible and more amenable to cloud computing if needed.

If a future research project is focused less on hardware-algorithm co-design, and more on the performance of different algorithms using a shared hardware environment, then we recommend the Deepdrive environment without qualification. From what we've seen of it, the back end driving simulator for Deepdrive is high quality (accurate), flexible enough to accommodate different scenarios, and performs reasonably fast. Deepdrive comes integrated with the popular OpenAI gym reinforcement learning environment. This makes it much easier to get off the ground running if you are interested in studying reinforcement learning agents.

While we've learned more about how to benchmark autonomous driving agents than about the agents themselves, we've learned at least a few things. Engineering rewards for autonomous driving agents is not obvious. The reward function in Deepdrive, which on the whole is excellent software, did not make sense to us compared to our subjective interpretation of agent performance. Improving the reward function is an interesting open research problem.

All of our driving agents were tested on high end desktop computers or servers. The computing resources of these machines very comfortably handled the compute demands for all of the tested agents. While training the neural network agents was highly compute intensive, running inference on them is easy. The MNet-2 network, as the name suggests, is built on a scale that is intended to run quickly on the limited hardware available on mobile devices. Running a neural network on this scale should not present any challenge to a vehicle platform.

Our results also suggest to us that neural networks on the scale of MNet-2 are unlikely to be able to handle the driving task in realistic conditions. We found that performance of these agents plateaued relatively quickly during training at levels that were far below what would be required to drive safely. Our estimate is that any future autonomous driving agents based on neural networks will need much larger network sizes with a corresponding increase in computing requirements.

We also reached some conclusions about the Unprotected Left Turn contest run by Deepdrive. We think this competition is an excellent idea, and it facilitates the development of benchmarks on characteristics not examined by the competition. However, the competition evaluation has significant limits. In particular, entries that fail to complete the task of making an unprotected left but end quickly and have low g-forces are ranked much higher than entries that complete the task without crashing or exiting early but do so with higher g-forces. An even larger limitation in the contest framework is the combination of inadequate domain randomization and the small sample size used to assess competition entry. The top ranked entry, "lazy-driver", is a hand tuned left turn that does not generalize at all. It merely memorizes the steering inputs to make one smooth left turn after the two oncoming cars drive past it. There is some domain randomization, and depending on the random seed, this agent will sometimes crash. But contest standing is based on the single trial with the highest score. This admits a strategy of hand tuning an agent that can achieve a high score on one lucky outing. An obvious improvement to the contest design would be to

assess entries on their scores over a larger number of trials with randomized conditions.

ACKNOWLEDGEMENTS

This material benefited from suggestions from and discussions with Vijay Janapa Reddi as well as the other course staff of Harvard University's fall 2019 course CS 249R.

REFERENCES

Deepdrive from Voyage. URL https://deepdrive.voyage.auto/.

Apollo. URL http://apollo.auto/.

Paperspace. URL https://www.paperspace.com/.

Althoff, M., Koschi, M., and Manzinger, S. CommonRoad: Composable benchmarks for motion planning on roads. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 719–726, June 2017. doi: 10.1109/IVS.2017.7995802. ISSN: null.

Bacha, A., Bauman, C., Faruque, R., Fleming, M., Terwelp, C., Reinholtz, C., Hong, D., Wicks, A., Alberi, T., Anderson, D., Cacciola, S., Currier, P., Dalton, A., Farmer, J., Hurdus, J., Kimmel, S., King, P., Taylor, A., Covern, D. V., and Webster, M. Odin: Team VictorTango's entry in the DARPA Urban Challenge. *Journal of Field Robotics*, 25(8):467–492, August 2008. ISSN 1556-4967. doi: 10.1002/rob.20248. URL https://onlinelibrary.wiley.com/doi/10.1002/rob.20248.

Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., Zhang, X., Zhao, J., and Zieba, K. End to end learning for self-driving cars. *arXiv:1604.07316 [cs]*, April 2016. URL http://arxiv.org/abs/1604.07316. arXiv: 1604.07316.

Boroujerdian, B., Genc, H., Krishnan, S., Cui, W., Faust, A., and Reddi, V. J. MAVBench: Micro Aerial Vehicle Benchmarking. *arXiv:1905.06388 [cs]*, May 2019. URL http://arxiv.org/abs/1905.06388. arXiv: 1905.06388.

Calisi, D., Iocchi, L., and Nardi, D. A unified benchmark framework for autonomous Mobile robots and Vehicles Motion Algorithms (MoVeMA benchmarks). In *Workshop on experimental methodology and benchmarking in robotics research (RSS 2008)*, pp. 4, 2008.

Chen, X., Lisee, J., Wojtaszek, T., and Gupta, A. Introducing AVS, an open standard for autonomous vehicle visualization from Uber, February 2019. URL https://eng.uber.com/avs-autonomous-vehicle-visualization/.

Drews, P., Williams, G., Goldfain, B., Theodorou, E. A., and Rehg, J. M. Aggressive deep driving: Model Predictive Control with a CNN cost model. *arXiv:1707.05303 [cs]*, July 2017. URL http://arxiv.org/abs/1707.05303. arXiv: 1707.05303.

Geiger, A., Lenz, P., and Urtasun, R. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3354–3361, June 2012. doi: 10.1109/CVPR.2012.6248074. ISSN: 1063-6919.

Kanade, T., Thorpe, C., and Whittaker, W. Autonomous land vehicle project at cmu. In *Proceedings of the 1986 ACM Fourteenth Annual Conference on Computer Science*, CSC '86, pp. 71–80, New York, NY, USA, 1986. ACM. ISBN 0-89791-177-6. doi: 10.1145/324634.325197. URL http://doi.acm.org.ezp-prod1.hul.harvard.edu/10.1145/324634.325197.

Kondermann, D., Nair, R., Honauer, K., Krispin, K., Andrulis, J., Brock, A., Gussefeld, B., Rahimimoghaddam, M., Hofmann, S., Brenner, C., and Jahne, B. The HCI Benchmark Suite: Stereo and flow ground truth with uncertainties for urban autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 19–28, 2016. URL https://www.cv-foundation.org/openaccess/content_cvpr_2016_workshops/w3/html/Kondermann_The_HCI_Benchmark_CVPR_2016_paper.html.

Krishnan, S., Borojerdian, B., Fu, W., Faust, A., and Reddi, V. J. Air Learning: An AI research platform for algorithm-hardware benchmarking of autonomous aerial robots. *arXiv:1906.00421 [cs]*, June 2019. URL http://arxiv.org/abs/1906.00421. arXiv: 1906.00421.

Leonard, J. J., Barrett, D., How, J. P., Teller, S., Antone, M., Campbell, S., Epstein, A., Fiore, G. A., Fletcher, L., Frazzoli, E., Huang, A. S., Jones, T., Koch, O., Kuwata, Y., Mahelona, K., Moore, D., Moyer, K., Olson, E., Peters, S. C., Sanders, C., Teo, J., and Walter, M. R. Team MIT Urban Challenge technical report. Technical report, Massachusetts Institute of Technology, 2007.

McAllister, R., Gal, Y., Kendall, A., Van Der Wilk, M., Shah, A., Cipolla, R., and Weller, A. Concrete problems for autonomous vehicle safety: Advantages of Bayesian deep learning. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, pp. 4745–4753. AAAI Press, 2017. ISBN 978-0-9992411-0-3. URL http://dl.acm.org/citation.cfm?

id=3171837.3171951. event-place: Melbourne, Australia.

Mueller, M., Smith, N., and Ghanem, B. A benchmark and simulator for UAV tracking. In Leibe, B., Matas, J., Sebe, N., and Welling, M. (eds.), *Computer Vision – ECCV 2016*, volume 9905, pp. 445–461. Springer International Publishing, Cham, 2016. ISBN 978-3-319-46447-3 978-3-319-46448-0. doi: 10.1007/978-3-319-46448-0_27. URL http://link.springer.com/10.1007/978-3-319-46448-0_27.

Nardi, L., Bodin, B., Zia, M. Z., Mawer, J., Nisbet, A., Kelly, P. H. J., Davison, A. J., Luján, M., O'Boyle, M. F. P., Riley, G., Topham, N., and Furber, S. Introducing SLAMBench, a performance and accuracy benchmarking methodology for SLAM. *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5783–5790, May 2015. doi: 10.1109/ICRA.2015.7140009. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7140009IEEEXplore2015. arXiv: 1410.2167.

Pan, Y., Cheng, C.-A., Saigol, K., Lee, K., Yan, X., Theodorou, E., and Boots, B. Agile autonomous driving using end-to-end deep imitation learning. *Robotics: Science and Systems*, 2018. URL http://arxiv.org/abs/1709.07174. arXiv: 1709.07174.

Pomerleau, D. A. Alvinn: An autonomous land vehicle in a neural network. In *Advances in neural information processing systems*, pp. 305–313, 1989.

Richter, S. R., Hayder, Z., and Koltun, V. Playing for Benchmarks. *arXiv:1709.07322 [cs]*, September 2017. URL http://arxiv.org/abs/1709.07322. arXiv: 1709.07322.

Shah, S., Dey, D., Lovett, C., and Kapoor, A. AirSim: High-fidelity visual and physical simulation for autonomous vehicles. *arXiv:1705.05065 [cs]*, July 2017. URL http://arxiv.org/abs/1705.05065. arXiv: 1705.05065.

Sturm, J., Engelhard, N., Endres, F., Burgard, W., and Cremers, D. A benchmark for the evaluation of RGB-D SLAM systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 573–580, Vilamoura-Algarve, Portugal, October 2012. IEEE. ISBN 978-1-4673-1736-8 978-1-4673-1737-5 978-1-4673-1735-1. doi: 10.1109/IROS.2012.6385773. URL http://ieeexplore.ieee.org/document/6385773/.

Urmson, C., Baker, C., Dolan, J., Rybski, P., Salesky, B., Whittaker, W., Ferguson, D., and Darms, M. Autonomous driving in traffic: Boss and the Urban Challenge. *AI Magazine*, 30(2):17, February 2009. ISSN 0738-4602, 0738-4602. doi: 10.1609/aimag.v30i2.2238. URL https://aaai.org/ojs/index.php/aimagazine/article/view/2238.

Vinitsky, E., Kreidieh, A., Flem, L. L., Kheterpal, N., Jang, K., Wu, C., Wu, F., Liaw, R., Liang, E., and Bayen, A. M. Benchmarks for reinforcement learning in mixed-autonomy traffic. In Billard, A., Dragan, A., Peters, J., and Morimoto, J. (eds.), *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pp. 399–409. PMLR, 29–31 Oct 2018. URL http://proceedings.mlr.press/v87/vinitsky18a.html.

Weisz, J., Huang, Y., Lier, F., Sethumadhavan, S., and Allen, P. RoboBench: Towards sustainable robotics system benchmarking. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3383–3389, May 2016. doi: 10.1109/ICRA.2016.7487514. ISSN: null.