

# Scaling Is All You Need: Autonomous Driving with JAX-Accelerated Reinforcement Learning

Moritz Harmel

zoox.com

Anubhav Paras

zoox.com

Andreas Pasternak

zoox.com

Nicholas Roy

zoox.com

Gary Linscott

zoox.com

## Abstract

*Reinforcement learning has been demonstrated to outperform even the best humans in complex domains like video games. However, running reinforcement learning experiments on the required scale for autonomous driving is extremely difficult. Building a large scale reinforcement learning system and distributing it across many GPUs is challenging. Gathering experience during training on real world vehicles is prohibitive from a safety and scalability perspective. Therefore, an efficient and realistic driving simulator is required that uses a large amount of data from real-world driving. We bring these capabilities together and conduct large-scale reinforcement learning experiments for autonomous driving. We demonstrate that our policy performance improves with increasing scale. Our best performing policy reduces the failure rate by 64% while improving the rate of driving progress by 25% compared to the policies produced by state-of-the-art machine learning for autonomous driving.*

## 1. Introduction

In this paper, we set up a scalable reinforcement learning framework and combine it with an efficient simulator for autonomous driving based on real-world data. We then conduct experiments on billions of agent steps with different model sizes in order to determine if we can overcome the constrained state space of the simulator by using increasingly more real-world data.

The main contributions of our work are:

1. We demonstrate how to use prerecorded real-world driving data in a hardware-accelerated simulator as part of distributed reinforcement learning to achieve improving policy performance with increasing experiment size.
2. We demonstrate that our largest scale experiments, using a 25M parameter model on 6000 h of human-expert driving from San Francisco training on 2.5 billion agent steps, reduced the failure rate compared to the current state of the art [9] by 64%.

## 2. Related work

In this paper we present a hardware-accelerated autonomous driving simulator for real-world driving scenarios, which is similar to Waymax [6]. Waymax also trains RL agents but only reported results from 30 million agent steps, approximately two orders of magnitude less than our experiments.

Training highly performant policies in complex and continuous real-world domains has mainly been achieved with distributed and scalable reinforcement learning using actor-critic methods. However, most existing results focus not on real-world problems but on video game environments [1, 11].

Of the techniques that have been used to train autonomous driving policies, the closest to our work is Imitation Is Not Enough [9] which also uses a combination of imitation learning (IL) and RL to train strong driving policies on a large dataset of real-world driving scenarios. That work established two fundamental driving metrics (failure rate and progress ratio) and provides a detailed description of the mining process of their evaluation dataset. The presented policies are state of the art (SOTA) and we will compare our best policy to theirs. Other work focuses on realistic traffic simulation [13], but also trains policies with a combined IL and RL approach. However, their datasets are approximately 3 orders of magnitude smaller than ours and focus on highway driving, which does not pose as complex challenges as the dense urban driving we focus on. Imitation learning approaches in open loop [3, 12, 14] and closed loop [2, 8] have also been proposed for autonomous driving. As [9] argue, IL approaches lack explicit knowledge of unsafe driving and can respond inappropriately in rare, long tail scenarios.

## 3. Large scale RL for autonomous driving

The challenges of using large scale reinforcement learning for autonomous driving are manifold. To achieve scale, the real-world problem must be modeled by a simulator, which requires creating realistic traffic scenes and modeling the interactions between different dynamic agents. The simu-

lator must be sufficiently efficient to generate environment interactions on the order of billions of steps in reasonable time and computational cost. Finally, a distributed learning architecture must be identified that can learn efficiently and scalably. Learners and simulation actors must be created across many machines, each leveraging parallel hardware, i.e. GPUs.

### 3.1. Scene generation and agent interactions

There are different ways of creating traffic scenes for simulation. Simulations scenarios can be generated entirely synthetically, including the placement of roads, agents, and traffic signals, for example as in the Carla simulator [4]. While entirely synthetic simulation gives very fine grained control over training scenario distribution, this approach raises the challenge of identifying what that scenario distribution should be, and how best to sample training instances from it. A different approach is to create scenes based on real-world data recorded from vehicles equipped with sensors, such as cameras and lidar sensors, driving on public roads. From the recorded sensor data, a 3D-scene can be created that contains, for example, the observed traffic light states and challenging obstacles, such as pedestrians, cyclists, and cars [6]. In this work, we use scenarios that have been created from real-world driving, for the fidelity of their representation of the real world.

### 3.2. Accelerated autonomous driving simulator

Because our ultimate goal is to run reinforcement learning experiments with billions of agent steps, the simulator must be very efficient. This can be achieved by running the simulation on accelerated hardware, such as GPUs. The parallel computing power of accelerated hardware can lead to massive speed ups compared to CPUs. However, challenges regarding the simulation data structures and control flow need to be addressed to enable large scale parallelism. In particular, all data need to be of the same size and no logical branches depending on the values of the data can be introduced.

#### 3.2.1 Preparing data for parallel execution

A traffic scene can be described by data that changes over time (dynamic data) and data that is constant throughout the scenario (static data). An example of dynamic data are the agents in the scene — the number of agents can change during a scenario as well as between scenarios. An example for static data are the road segments, which do not change within a scenario but change across scenarios as different locations require different roads. Furthermore, the number of time steps per scene can vary.

Data segments of different sizes are not suitable for parallel execution on hardware accelerators. To overcome this

issue, a common maximum size for each type of data is defined. All data elements in the dynamic data are then padded to the defined maximum size for each time step.

#### 3.2.2 The accelerated simulation utilizing JAX

Conceptually, the simulation can be divided into a phase of action generation and a phase of advancing the environment state by applying the selected actions to the active agent and updating all other agent positions based on the logged trajectories. From the updated environment state including the recorded data, the required observations can be retrieved and used in the next step of action generation from the learning system as described in Section 3.4. The action generation is well-suited for batched inference, so the primary challenge in simulating on parallel hardware is to implement the environment update function to process batched data in parallel. We used the JAX library to rewrite the update function, so it can be jit-compiled and executed on batches on the GPU. Finally, to maximize the simulation speed, we combine the batched environment call with the batched model call and scan along the time axis of the dynamic data via the *jax.lax.scan* primitive. The entire simulation is then jit-compiled into a single graph and run in XLA.

#### 3.2.3 Simulator performance benchmark

To demonstrate the performance of our simulator, we compare the environment step time with Waymax [6], which is closest to our work. Table 1 reports the step time for different batch sizes on Nvidia v100 GPUs, showing that our simulator runs slightly faster.

### 3.3. RL problem formulation

For our reinforcement learning approach we need to specify how we retrieve the model inputs (observations) from the state  $s$  and how we generate the physical actions from the model outputs. The state  $s$  is the state of the simulation described previously, i.e., agents, roads, traffic lights etc. We also need to define the rewards, so the simulated data can be used to calculate the parameter update from an RL method.

Table 1. Runtime comparison for different batch sizes (BS) between Waymax [6] and our simulator for one controlled agent.

Simulator	Device	Step time [ms]	
		BS 1	BS 16
Waymax	v100	0.75	2.48
Ours	v100	0.52	0.82

### 3.3.1 Observation space

The observation space retrieves a subset of the information of the environment state and transforms the fields of the observation vector into an agent-centric coordinate frame.

### 3.3.2 Action space

Our model directly controls the longitudinal acceleration as well as the steering angle rate. Using these controls guarantees that the associated dynamic constraints are not violated. We are using discrete actions, which we found to be more stable than continuous actions during reinforcement learning training.

### 3.3.3 Rewards

The goal of the policy is to navigate safely through traffic. In particular, the agent should make progress along the desired route while not colliding with other agents and adhering to basic traffic rules. In order to achieve this, we introduce dense rewards as well as done signals that are associated with sparse rewards.

Done signals have been introduced for collisions, off-route driving, as well as running red lights and stop lines and are associated with high negative rewards. Dense rewards are introduced for the progress along the planned route (positive), velocity above the speed limit, as well as on the squared lateral and longitudinal acceleration (all negative).

## 3.4. Distributed learning system

We can now establish the reinforcement learning approach. Our base reinforcement learner uses actor-critic Proximal Policy Optimization (PPO) [10]. However, to achieve scale, we set up an asynchronous reinforcement learning system similar to Dota2 [1]. The asynchronous setting allows to run the learners and actors independently avoiding any slow down. This in turn causes the data to be off-policy. We address this challenge for actor-critic methods by using the V-trace off-policy correction algorithm [5]. Furthermore, we pre-train a policy via behavioral cloning, similar to AlphaStar [11] because a good initial policy can speed up the RL training. However, only pre-training the policy and not the value network poses challenges to the stability at the start of RL training. Therefore, we use the discounted return of the expert trajectories as the value target. We calculate the discounted return by replaying the expert trajectories in our simulator and assigning the defined RL rewards.

## 4. Evaluation

This section describes the different metrics and the dataset used for policy evaluation and comparison.

## 4.1. Metrics

The goal of the metrics are to measure the quality of the trained policy. This is already a complex problem for autonomous driving as the quality of driving comprises many different aspects. For the scope of this paper, we follow the work of [9] who introduced the failure rate and progress ratio as relevant metrics for autonomous driving. The failure rate measures the fundamental safety of the policy. If the agent collides or drives off-road in the simulation the scenario is considered as failed. The progress ratio is the distance traveled by the agent in the simulation divided by the distance traveled of the vehicle in the original log. When the agent travels the same distance as the vehicle in the log, this metric becomes 100 %.

In addition to collisions and off-route failures, we also implemented metrics for stop line and traffic light violations. We did not include these violations in the failure rate to maintain consistency with previously reported results. However, we do report these metrics in Table 2.

## 4.2. Dataset

As the current state-of-the-art policies [9] are evaluated on proprietary datasets, our goal was to achieve the fairest comparison by following the same dataset mining procedure. A dataset of 10k randomly sampled 10 s segments was created using data collected from human-expert driving in San Francisco. This is comparable to the "All" evaluation dataset in [9].

## 5. Experiments

Combining the real-world driving simulator, the scalable reinforcement learning framework and the described evaluation metrics and dataset, we conduct experiments with different training dataset and model sizes. For all these experiments we keep the hyperparameters the same. In particular we run our experiments for larger models across more GPUs to achieve the same batch size.

We mined three different training datasets of 600 h, 2000 h and 6000 h from human-expert driving in San Francisco. We also created three different model sizes of 0.75M, 2.5M and 25M parameters by increasing the attention dimensions of the network. Each model is trained first by behavior cloning for 20 epochs on the given dataset. The pre-trained policy is then refined by reinforcement learning on 2.5B agent steps. We evaluate the policy during reinforcement learning every 20M agent steps and after training select the checkpoint with the lowest failure rate on the evaluation dataset.

We conduct experiments on all combinations of model size and dataset size, with the exception of the small 600 h dataset in combination with the large 25M parameter model. Figure 1a shows that increasing the dataset size improves

	600 h	2000 h	6000 h
0.75M	3.12 %	2.27 %	1.38 %
2.5M	1.92 %	1.19 %	1.14 %
25M		1.35 %	0.88 %

(a) Minimum failure rate

	600 h	2000 h	6000 h
0.75M	345 h	333 h	342 h
2.5M	822 h	975 h	805 h
25M		16118 h	18476 h

(b) GPU time

Figure 1. Results for experiments with different model sizes (rows) and dataset sizes (columns). Colors represent the numerical results on color scales. (a) The performance of the policy improves with increasing model and dataset size. (b) The model size is the major driver of the required GPU time and therefore cost of training. Dataset size has no effect on the training time, but it can affect one time costs during data preprocessing which is not considered here.

the performance of the trained policy in terms of failure rate. Increasing the model size in general also improves the policy performance. The 2.5M model is strictly better than the 0.75M model and the best policy is trained on the 25M model. However, we observe that increasing the model size only helps when sufficient real-world driving data is available. On the 2000 h dataset the 25M performs worse than the 2.5M model and only on the 6000 h dataset it performs better. The largest experiment achieves a failure rate of 0.88 %.

In Table 2 we compare the policy performance of our largest setting after behavioral cloning and after reinforcement learning training with the current SOTA [9]. Our behavioral cloning policy performs quite poorly, achieving a failure rate of 19.85 %. This is much higher than the pure BC failure rate of 3.64 % reported in the current SOTA [9].

The reinforcement learning training improves the policy and achieves a failure rate of 0.88 % and a progress ratio of 120.8 %. Compared to the best policy of the current SOTA on a similar dataset, the failure rate is reduced by 64% and the progress ratio improved by 25%.

## 6. Conclusions

In this paper we combined an efficient and realistic autonomous driving simulator with a scalable reinforcement learning framework. This allowed us to run large scale reinforcement learning experiments training on billions of agents steps with increasing model size on different dataset sizes of real-world driving.

Our data shows that we can obtain similar scaling behavior as in other reinforcement learning settings [7] when using increasingly large datasets of real-world driving. In particular, we were able to obtain better policies with larger models when using sufficiently large datasets. Our best policy reduces the failure rate compared to the current SOTA [9] by 64% while improving progress by 25%. These results are very encouraging, and motivate further experiments with increasing size. However, to ultimately answer whether the presented approach can be scaled beyond human performance a validation framework that can reliably compare the safety of the policy to human drivers is also required.

Table 2. Comparison of our policies with the current SOTA [9].

	BC [9]	MGAIL [9]	SAC [9]	BC+SAC [9]	Our BC	Our BC+PPO
Failure Rate [%] [9]	3.64	2.45	5.60	2.81	19.85	<b>0.88</b>
Progress Ratio [%] [9]	98.1	96.6	71.1	87.6	96.91	<b>120.8</b>
Collisions [%]	-	-	-	-	10.32	<b>0.46</b>
Off-Route Events [%]	-	-	-	-	10.35	<b>0.49</b>
Stop Line Violations [%]	-	-	-	-	2.47	<b>0.02</b>
Traffic Light Violations [%]	-	-	-	-	2.08	<b>0.28</b>

## References

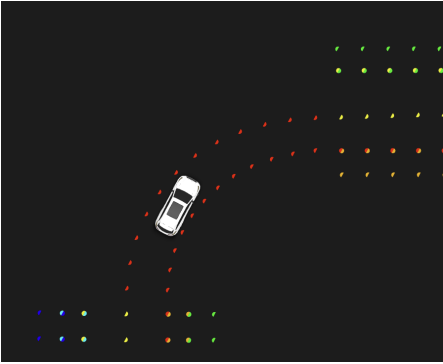
- [1] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique P. d. O. Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with large scale deep reinforcement learning, 2019. [1](#), [3](#)
- [2] Eli Bronstein, Mark Palatucci, Dominik Notz, Brandyn White, Alex Kuefler, Yiren Lu, Supratik Paul, Payam Nikdel, Paul Mougin, Hongge Chen, Justin Fu, Austin Abrams, Punit Shah, Evan Racah, Benjamin Frenkel, Shimon Whiteson, and Dragomir Anguelov. Hierarchical model-based imitation learning for planning in autonomous driving, 2022. [1](#)
- [3] Eli Bronstein, Sirish Srinivasan, Supratik Paul, Aman Sinha, Matthew O’Kelly, Payam Nikdel, and Shimon Whiteson. Embedding synthetic off-policy experience for autonomous driving via zero-shot curricula, 2022. [1](#)
- [4] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator, 2017. [2](#)
- [5] Lasse Espeholt, Hubert Soyer, Remi 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, 2018. [3](#)
- [6] Cole Gulino, Justin Fu, Wenjie Luo, George Tucker, Eli Bronstein, Yiren Lu, Jean Harb, Xinlei Pan, Yan Wang, Xiangyu Chen, John D. Co-Reyes, Rishabh Agarwal, Rebecca Roelofs, Yao Lu, Nico Montali, Paul Mougin, Zoey Yang, Brandyn White, Aleksandra Faust, Rowan McAllister, Dragomir Anguelov, and Benjamin Sapp. Waymax: An accelerated, data-driven simulator for large-scale autonomous driving research, 2023. [1](#), [2](#)
- [7] Jacob Hilton, Jie Tang, and John Schulman. Scaling laws for single-agent reinforcement learning, 2023. [4](#)
- [8] Maximilian Igl, Daewoo Kim, Alex Kuefler, Paul Mougin, Punit Shah, Kyriacos Shiarlis, Dragomir Anguelov, Mark Palatucci, Brandyn White, and Shimon Whiteson. Symphony: Learning realistic and diverse agents for autonomous driving simulation, 2022. [1](#)
- [9] Yiren Lu, Justin Fu, George Tucker, Xinlei Pan, Eli Bronstein, Rebecca Roelofs, Benjamin Sapp, Brandyn White, Aleksandra Faust, Shimon Whiteson, Dragomir Anguelov, and Sergey Levine. Imitation is not enough: Robustifying imitation with reinforcement learning for challenging driving scenarios, 2023. [1](#), [3](#), [4](#)
- [10] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. [3](#)
- [11] Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander S. Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom L. Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019. [1](#), [3](#)
- [12] Matt Vitelli, Yan Chang, Yawei Ye, Maciej Wołczyk, Błażej Osiński, Moritz Niendorf, Hugo Grimmett, Qiangui Huang, Ashesh Jain, and Peter Ondruska. SafetyNet: Safe planning for real-world self-driving vehicles using machine-learned policies, 2021. [1](#)
- [13] Chris Zhang, James Tu, Lunjun Zhang, Kelvin Wong, Simon Suo, and Raquel Urtasun. Learning realistic traffic agents in closed-loop, 2023. [1](#)
- [14] Zhejun Zhang, Alexander Liniger, Dengxin Dai, Fisher Yu, and Luc Van Gool. End-to-end urban driving by imitating a reinforcement learning coach, 2021. [1](#)

# Scaling Is All You Need: Autonomous Driving with JAX-Accelerated Reinforcement Learning

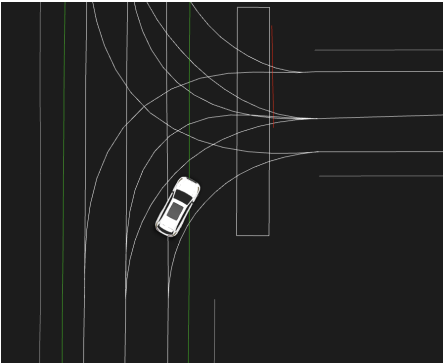
## Supplementary Material

### 7. Roads observations

The implemented roads library in our simulator delivers important information for our rewards and observations. For example, it can calculate the distance to the next stop line. We also use it to create observations of the route and the nearby road segments. This is illustrated in [Figure 2](#).



(a) Route border points of all drivable lanes. Only one lane is a valid connection for the right turn the agent is performing. Before and after the turn, other lanes are also valid.



(b) Road network points visualized by poly-lines. The color depends on the properties of the annotation, for example the stop line being visualized in red and bike lane boundaries in green.

Figure 2. Route and road network observations obtained from the roads library.

### 8. Metric validation

We validated the collision and off-route detection by taking 25 positive and 25 negative samples for each metric from the validation dataset. These have been inspected by human triagers to confirm whether the metric is correct. For the collision and the off-route detection this validation found that all 50 scenarios for each metric were labeled correctly

according to our definition. However, overall both metrics were found to be conservative, leading potentially to higher failure rates.

For the collision-free metric a bounding box approximating the vehicle is checked against the bounding box of other vehicles. As the bounding box includes all parts of the vehicle, for example the mirrors and sensors, checking collisions against the bounding box is conservative. [Figure 3a](#) illustrates this conservative check.

In many situations the permissible lane along the route is overly restrictive leading to off-route events. This occurs particularly in junctions ([Figure 3b](#)) and lane merges or branches ([Figure 3c](#)). Human expert drivers were found to not exactly follow these lane geometries either. Future work to relax the conditions in these cases to the entire drivable surface is required. On top of that, the check is also on the conservative side due to the increased bounding box size.

### 9. Framework scalability

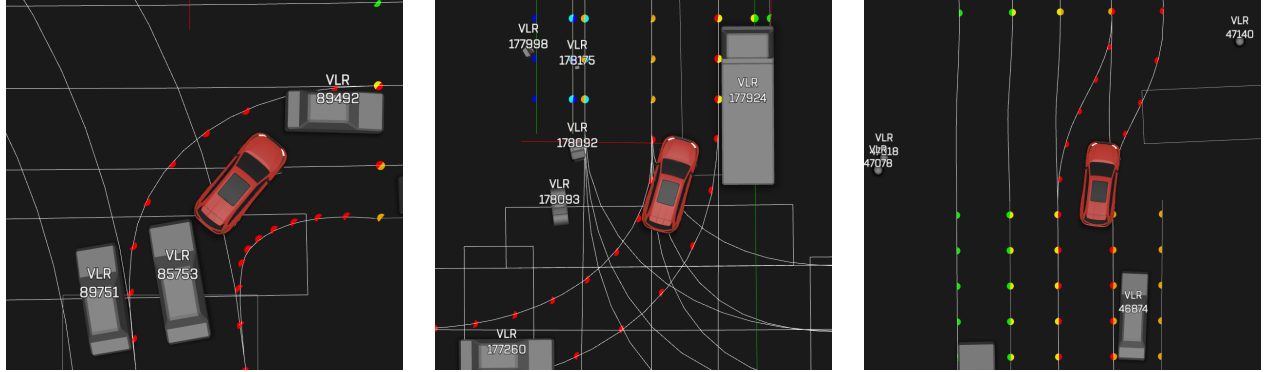
When scaling experiments to more GPUs it is important that the policy performance is not affected and that the overall runtime is reduced. Ideally, the reduction in runtime is inversely proportional to the number of GPUs used, so the overall GPU time and, therefore, the cost to run the experiment stays the same. [Table 3](#) shows the results for experiments running on 8, 16, and 32 machines. The overall policy performance is very similar and the differences can be considered noise. The overall GPU time sees a marginal uptick. We calculated the normalized GPU time by dividing the total GPU time by the total GPU time of the 8-GPU experiment. As the numbers are close to 1, the framework scales almost perfectly.

Table 3. Runtime and policy performance metrics for experiments running on different numbers of GPUs. The policy performance is very similar, but the runtime goes down as expected.

Number of GPUs	8	16	32
Runtime [h]	41.58	21.94	11.99
Total GPU time [h]	332.6	351.0	383.7
Normalized GPU time	1	1.05	1.15
Failure Rate	1.28	1.25	1.45

### 10. Ablation of SL pre-training

For this ablation we removed the SL pre-training in order to understand the effectiveness of this step. We used the large



(a) Detected collision with agent 85753 because of the conservative bounding box check.

(b) Overly restrictive lane permissibility in a junction. Off-route event detected due to increased bounding box size.

(c) Overly restrictive lane permissibility on a road segment with branching lanes.

Figure 3. Examples of conservative detection of collisions and off-route events.

6000 h dataset, the 2.5M model and again trained on 2.5B agent steps.

Even without the SL pre-training the agent can learn to make progress and reduce the failure rate over the course of training as depicted in Figure 4. However, after 2.5B agent steps the policy without pre-training is still at about 2 % failure rate, which the pre-trained policy reached already after 0.5B steps. Also the failure rate at 2.5B agent steps is lower for the policy that has been pre-trained. The increased training speed is also observed for the progress ratio. The pre-trained policy reaches values around 120 % after 0.5B steps and the policy without pre-training catches up to that value at around 2.5B steps. These results overall confirm the effectiveness of the pre-training step in terms of speed. Pre-training also helps to reach better final performance in terms of safety.

## 11. Hyperparameters

Table 4 shows the most important hyperparameters for both the behavioral cloning stage and the reinforcement learning stage.

Table 4. BC and RL Hyperparameters

	BC	RL
Batch size	32768	512
Sequence length	-	32
Learning rate	$2 * 10^{-3}$	$5.6 * 10^{-5}$
PPO clip param	-	0.3
Value loss scaling	$10^{-4}$	$10^{-2}$
PPO entropy coefficient	-	$3 * 10^{-2}$

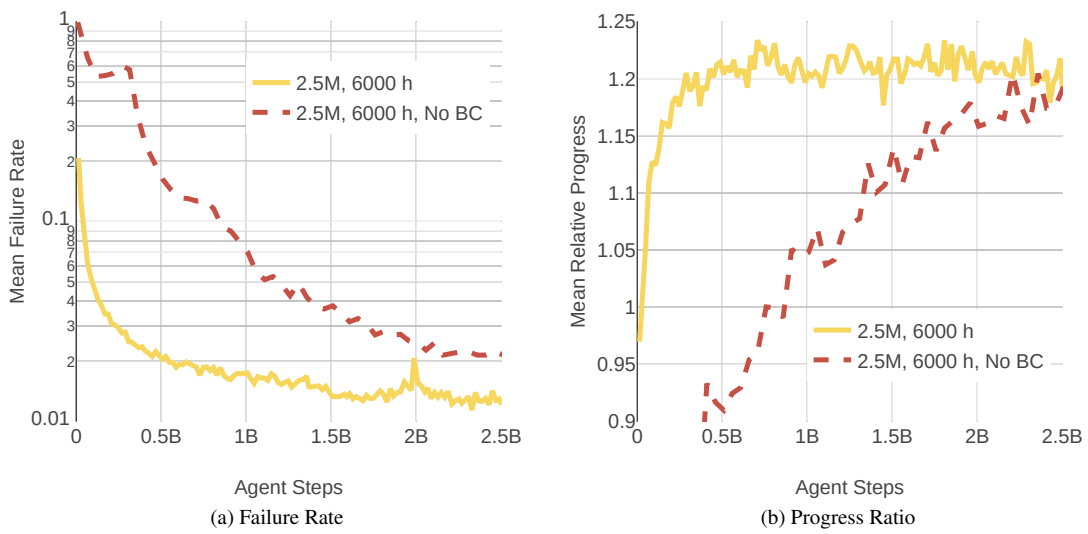


Figure 4. Training curves for experiments with and without the SL pre-training on the 2.5M parameter model and the 6000 h dataset.