Fairness and Robustness of Mixed Autonomous Traffic Control with Reinforcement Learning

Ashay Athalye
ashay@mit.edu

Shannon Hwang
hwangys@mit.edu

Siddharth Nayak
sidnayak@mit.edu

Abstract

Mixed autonomy (MA) scenarios – where both autonomous vehicles (AVs) and human drivers share the same road – will become increasingly prevalent as autonomous vehicles are deployed into society. From a reinforcement learning perspective, this offers a variety of interesting research opportunities such as modeling problems with very large state spaces, multiple agents, and exploring reward design with fairness constraints. In this work we try to replicate an existing benchmark for the bottleneck environment and investigate the changes in learned agent policies and performance when explicitly considering fairness and human driver model variation during training. We find that adding a fairness term to the reward function significantly changes the learned behavior, allowing all vehicles to move through the bottleneck at approximately equal average speeds while decreasing the throughput through the bottleneck by small and at times insignificant amounts.

1 Introduction

In recent years, several sequential decision-making tasks have been tackled using reinforcement learning methods. Many works have tried using various methods like imitation learning[1, 2], reinforcement learning [3, 4], supervised and self-supervised learning [5, 6] for the autonomous driving task. With many companies like Google, Waymo, Tesla, etc. deploying autonomous vehicles in the real-world, mixed autonomous scenarios are inevitable where autonomous vehicles and human-driven vehicles share the same road networks. With the emergence of rich road network simulators like Flow [7] and SUMO [8] researchers have started investigating traffic control in these scenarios.

In this project, we consider the problem of traffic congestion in a previously-published bottleneck benchmark scenario [9]. Previous works have focused on optimizing throughput through the bottleneck, with less focus on ensuring fairness in the optimal solution and assessing robustness to possible environmental variation. Thus, our project explores the impacts of explicitly considering fairness and more human driver variation while training the AVs on their throughput-increasing effects in the bottleneck environment.

2 Related Works

In [9] the authors present benchmarks for the performance of reinforcement learning algorithms on four classic traffic scenarios: figure eight, merge, grid, and bottleneck. We base most of our work on this paper. Vinitsky et al. (2020) [10] study a fully decentralized mixed autonomy reinforcement learning (MARL) control scheme on a mixed autonomy two-stage bottleneck environment and report a significant improvement on the vehicular throughput as compared to hand-designed controllers. To the best of our knowledge, we did not find any related work which investigate fairness in the MA scenario explicitly or which add variation in human driver models parameters. Works we have seen use one human driver model and add some random noise to the actions they take, which is very

LaTeXTemplate borrowed from NeurIPS 2020

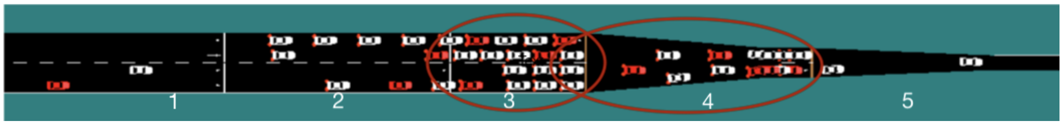


Figure 1: The bottleneck environment. Edges 1,2 and 3 have four lanes, edge 4 has two lanes and edge 5 has one lane. The red circles denote the edges where lanes merge to a smaller number of lanes.

different from having human drivers with different underlying driving behavior, such as different car following behavior.

3 Method

3.1 Environment

We use Flow [7] along with SUMO (Simulation of Urban MObility) [8] which is a framework for deep reinforcement learning (RL) and control experiments for traffic simulation. In particular we use the `bottleneck0` and the `bottleneck1` environment for our experiments. In the bottleneck environment as shown in Figure 1, the lanes reduce from four to two to one. The goal of this problem is to maximise the total outflow of the vehicles in a mixed-autonomy setting. In the environment, each segment of the road is termed as an "edge" (numbered from 1-5 in Figure 1). The Markov Decision Process (MDP) [11] for the bottleneck environment is as follows:

- **States**: The mean positions and velocities of human drivers for each lane for each edge segment. The mean positions and velocities of the connected autonomous vehicles (CAVs) on each segment. The outflow of the system in vehicles per/hour over the last 5 seconds.
- **Actions**: For a given edge-segment and a given lane, the RL action shifts the maximum speed of all the CAVs in the segment from their current value. By shifting the max-speed to higher or lower values, the system indirectly controls the velocity of the RL vehicles.
- **Rewards**: $r_t = \sum_{i=t-\frac{5}{\Delta t}}^{i=t} \frac{n_{exit}(i)}{\frac{5}{\Delta t * n_{lanes} * 500}}$ where $n_{exit}(i)$ is the number of vehicles that exited the system at time-step i. Basically, this is the outflow of the vehicles over the last 5 seconds normalised by the number of lanes, n_{lanes} and a factor of 500.

In our experiments we use two different variants of the bottleneck environment:

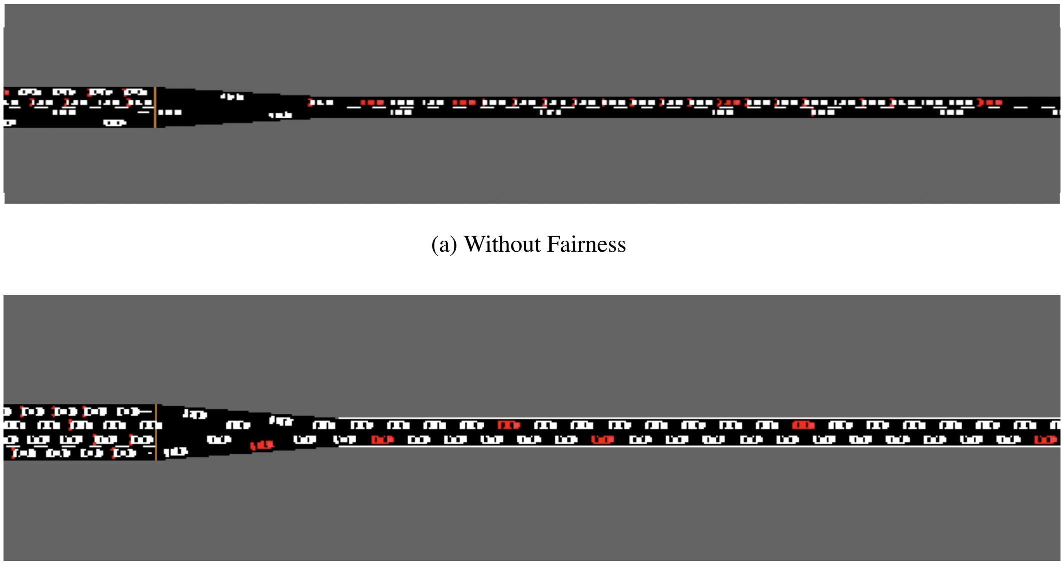
- `bottleneck0`: inflow = 1900 veh/hour, 10% CAV penetration. No vehicles are allowed to lane change. $(\mathcal{S} \in \mathbb{R}^{141}, \mathcal{A} \in \mathbb{R}^{20}, T = 1000)$
- `bottleneck1`: inflow = 1900 veh/hour, 10% CAV penetration. The human drivers follow the standard lane changing model in the simulator. $(\mathcal{S} \in \mathbb{R}^{141}, \mathcal{A} \in \mathbb{R}^{20}, T = 1000)$

3.2 Training

We use the RLlib [12] benchmark for the Proximal Policy Optimization (PPO) with Generalized Advantage Estimation (GAE) [13] algorithm for training the RL agents. We use the default hyperparameters as used in [9] for all of our experiments. We were unable to spend more time on hyperparameter tuning due to time constraints (run takes an enormous amount of time) and compute power limitations. We train all of our agents for 50 iterations each with 8 rollouts per training iteration and 1500 simulation steps per rollout.

3.3 Fairness

Previous work focused of learning RL policies in the bottleneck environment without any fairness considerations. As seen in Figure 2(a), this results in the controller learning a policy which blocks some lanes and reserves a high throughput to maximize the average throughput through the bottleneck [9]. To avoid learning similar behavior, we define fairness as all vehicles spending similar amounts of



(a) Without Fairness

(b) With Fairness

Figure 2: Behavior learnt when training: (a) without regard for fairness: autonomous vehicles (red) learn to block the upper lanes in order to reserve the lower lane as a high-throughput lane; (b) with regard for fairness: all lanes are travel at roughly the same speeds.

time in the bottleneck - e.g. all vehicles traveling through the bottleneck at approximately the same speed, no matter which lane they are in.

We add fairness considerations to training through reward shaping. More specifically, we add a fairness penalty to the rewards as follows:

$$\mu_{e_3} = \frac{1}{4}\sum_{i=0}^{3} v^i_{avg,e_3}; \qquad \mu_{e_4} = \frac{1}{2}\sum_{i=0}^{1} v^i_{avg,e_4}$$

$$\sigma_{e_3} = \sqrt{\frac{\sum(v^i_{avg,e_3} - \mu_{e_3})^2}{4}} \qquad \sigma_{e_4} = \sqrt{\frac{\sum(v^i_{avg,e_4} - \mu_{e_4})^2}{2}}$$

$$r_f = \alpha_{e_3} \cdot \sigma_{e_3} + \alpha_{e_4} \cdot \sigma_{e_4}$$

$$R^f_t = r_t + r_f$$

Here, μ_{e_3} and μ_{e_4} represent the average velocities across lanes in the segments where four lanes merge into two lanes (edge 3 in the simulator) and where two lanes merge into one lane (edge 4 in the simulator), respectively. We then evaluate the standard deviations of average velocities across the lanes in edge 3 (σ_{e_3}) and edge 4 (σ_{e_4}). These standard deviations are scaled to a similar magnitude as the original reward r_t by setting $\alpha_{e_3} = \alpha_{e_4} = -0.1$. We get the total reward R^f_t as the sum of the original reward and the fairness penalty terms.

We also experimented with adding a fairness penalty on only edge 4 ($r_f = \alpha_{e_4} \cdot \sigma^f_{e_4}$) as an alternative to the reward r_f described above.

3.4 Human driving behavior variation

As previous work assumes all humans follow the same driving model, we wanted to test the performance robustness of the learnt RL policy by exposing the model to different models of human driving behavior during training and test times. We assume there are "types" of human drivers that are drawn from some distribution, and autonomous agents can take actions to infer the parameters of the humans around them to then employ the optimal control policy that responds to that model. For example,

Parameter	Sampling Distribution
Max acceleration (m/s^2)	$\mathcal{N}(2.7, 0.1)$
Minimum desired following headway (s)	$\min(\max(0.5, \mathcal{N}(1,1)), 4)$
Speed gain	$\min\left(\max\left(0, \mathcal{N}\left(1,1\right)\right), 2\right)$
Speed gain lookahead (s)	$\max(5, Poisson(1))$
Pushiness	0.3
Impatience	$\min\left(\max\left(-0.5, \mathcal{N}\left(1, 0.16\right)\right), 0.5\right)$
Cooperativeness	$\mathcal{U}(0,1)$

Table 1: Distributions used to sample human driver parameters.

humans may have different car-following behavior (different acceleration, different desired following distance) in a high-traffic setting, and autonomous agents may be able to minimize the stopping and starting of different human drivers behind them if they can successfully learn models of different human driving behaviors. Justification for the chosen sampling distributions came from SUMO [8]'s documentation on vehicle types, lane-changing models, and car-following models, which in turn was based on the literature on human traffic modeling.

We randomly created 5 different types of human drivers to add to the bottleneck environments. Since the `bottleneck0` environment doesn't allow the human drivers to change lanes, we only altered the car-following behavior: each human driver type had a maximum acceleration (m/s^2) and minimum desired following headway (a reaction time, measured in seconds) sampled from the parameters in (Table 1). We set the minimum possible value for headway as $0.5s$ as it is the simulation step size.

For `bottleneck1`, we sampled values for all the parameters listed in (Table 1) for each human driver type. The speed gain, pushiness, impatience, and cooperativeness parameters are unit-less values used by SUMO to characterize different lane changing behaviors, whe re higher values indicate more dramatic behaviors. Speed gain represents a driver's eagerness to change lanes to gain speed. Speed gain lookahead controls a driver's lookahead time for anticipating slowdowns. Pushiness indicates a driver's willigness to encroach laterally on other drivers, and is multiplied by impatience. Cooperativeness represents willingness to cooperatively change lanes.

3.5 Evaluation

We adopt the same evaluation procedure as [9], and measure outflow (vehicles per hour) and throughput efficiency (defined as $\frac{outflow}{inflow}$) over the last 500 seconds of a 1000 second rollout, averaging results over 40 rollouts. As a baseline, we estimate human-level performance by running simulations of the traffic environment with no learning agents with the same default human driver model used in the fairness experiments.

4 Experiment Results

4.1 Fairness experiment results

In both `bottleneck0` (Table 2) and `bottleneck1` (Table 3), adding the fairness penalty to the reward reduces the standard deviation of average velocities across the lanes, ensuring vehicles travel through the bottleneck in approximately the same time. Notably, there is only a slight reduction in the throughput efficiency when the fairness penalty terms are added relative to the No Fairness baseline and performs better than the No AV baseline across all metrics.

4.2 Human driver variation experiment results

In both `bottleneck0` (Table 4) and `bottleneck1` (Table 5), adding a variety of human drivers to the environment during training and evaluation had little significant impact when compared to RL policies learned with less human driver variation during training and evaluation. These results held both with and without considering fairness.

Metric	No AVs	No Fairness	Fairness Edge 4	Fairness Edge 3+4
$\sigma^*_{e_3}$	2.9357 ± 1.0053	6.0091 ± 1.9731	**0.0644 ± 0.0534**	0.0650 ± 0.0866
$\sigma^*_{e_4}$	0.0703 ± 0.0844	5.1810 ± 2.2494	**0.0318 ± 0.0285**	0.2625 ± 0.5773
Outflow† (veh/hr)	1402.0 ± 22.249	**1589.9 ± 68.7569**	1489.5 ± 9.4135	1514.7 ± 30.6185
Throughput efficiency† (veh/hr)	0.5608 ± 0.0089	**0.6345 ± 0.0274**	0.5944 ± 0.0037	0.6045 ± 0.0122

Table 2: Results from experimenting with fairness penalties in the bottleneck 0 environment. σ_{e_3} and σ_{e_4} are the standard deviations of average velocities across the lanes in edge 3 and 4 of the bottleneck, respectively. $((.)^*$- lower the better; $(.)^\dagger$- higher the better)

Metric	No AVs	No Fairness	Fairness Edge 4	Fairness Edge 3+4
$\sigma^*_{e_3}$	1.6114 ± 0.1683	1.0757 ± 0.5137	**0.1491 ± 0.0848**	0.2565 ± 0.1455
$\sigma^*_{e_4}$	0.3630 ± 0.0738	7.9292 ± 1.5106	0.2092 ± 0.2180	**0.1217 ± 0.0786**
Outflow† (veh/hr)	1427.3 ± 20.935	**1733.0 ± 71.2798**	1495.9 ± 26.3179	1499.7 ± 29.9638
Throughput efficiency† (veh/hr)	0.5709 ± 0.0084	**0.6915 ± 0.0285**	0.5969 ± 0.0110	0.5984 ± 0.0121

Table 3: Results from experimenting with fairness penalties in the bottleneck 1 environment. σ_{e_3} and σ_{e_4} are the standard deviations of average velocities across the lanes in edge 3 and 4 of the bottleneck, respectively. $((.)^*$- lower the better; $(.)^\dagger$- higher the better)

5 Discussion

As a caveat, we note that all of our results are only preliminary and should be validated (as described in Section 5.1.1) drawing conclusive implications.

Our preliminary results for the fairness experiments indicate that fairness should be non-trivially taken into account when training AVs, as it can lead to significantly different learned behaviors that can drastically impact the ease of integrating AVs among human drivers.

Our preliminary results for the human driver model experiments indicate no significant change between training and testing with less human driver models and training and testing with more human driver models. This result may require more examination because we only added five new human driver models due to some errors in SUMO and FLOW, and due to time constraints did not separately characterize the impact of changing individual human driver model parameters.

5.1 Future work

5.1.1 Validating results

Due to time constraints and the computationally-intensive, time-consuming nature of training the RL agents for each experiment ($\sim$4 hours for 50 iterations of training), we did not train our agents using multiple random seeds and dramatically limited the number of training iterations. We also did not perform a hyperparameter search for training the PPO controller, instead using the default values listed in the Flow tutorials for hyperparameters (and verifying that they matched previously-published values). As can be seen in Figure 3, many training curves did not increase by very much over the course of training. The procedural changes outlined in this section can confirm or improve those results.

To validate and increase the robustness of our results, we would like to properly train each agent by performing a hyperparameter search, training with multiple random seeds for a statistical significance of the results, and training for more iterations for each experiment. We did try running experiments for 100 training iterations for `bottleneck0` but saw some strange behavior and did not have time to do the same for `bottleneck1`, and thus left those results out of this report. Thus, we emphasize the

Metric	RL Benchmark	With More Human Models	With Fairness Edges 3+4	With More Human Models + Fairness Edges 3+4
$\sigma^*_{e_3}$	6.0091 ± 1.9731	7.7162 ± 3.3361	**0.0650** ± **0.0866**	0.2632 ± 0.1758
$\sigma^*_{e_4}$	5.1810 ± 2.2494	7.8297 ± 2.6012	0.2625 ± 0.5773	**0.1548** ± **0.2224**
Outflow† (veh/hr)	**1589.9** ± **68.7569**	1587.4 ± 88.6358	1514.7 ± 30.6185	1416.0 ± 17.2189
Throughput efficiency† (veh/hr)	0.6345 ± 0.0274	**0.6353** ± **0.0354**	0.6045 ± 0.0122	0.5667 ± 0.0068

Table 4: Results from experimenting with more human driver types in the `bottleneck0` environment. σ_{e_3} and σ_{e_4} are the standard deviations of average velocities across the lanes in edge 3 and 4 of the bottleneck, respectively. ($(.)^*$- lower the better; $(.)^\dagger$- higher the better)

Metric	RL Benchmark	With More Human Models	With Fairness Edges 3+4	With More Human Models + Fairness Edges 3+4
$\sigma^*_{e_3}$	1.0757 ± 0.5137	2.2340 ± 1.2048	0.2565 ± 0.1455	**0.2077** ± **0.1928**
$\sigma^*_{e_4}$	7.9292 ± 1.5106	7.0668 ± 1.2711	0.1217 ± 0.0786	**0.0884** ± **0.0569**
Outflow† (veh/hr)	**1733.0** ± **71.2798**	1608.5 ± 91.8871	1499.7 ± 29.9638	1463.0 ± 11.7428
Throughput efficiency† (veh/hr)	**0.6915** ± **0.0285**	0.6434 ± 0.0373	0.5984 ± 0.0121	0.5855 ± 0.0047

Table 5: Results from experimenting with more human driver types in the `bottleneck1` environment. σ_{e_3} and σ_{e_4} are the standard deviations of average velocities across the lanes in edge 3 and 4 of the bottleneck, respectively. ($(.)^*$- lower the better; $(.)^\dagger$- higher the better)

need to do more hyperparameter tuning, to run experiments for more training iterations, and to try other RL algorithms such as TRPO.

5.1.2 Fairness

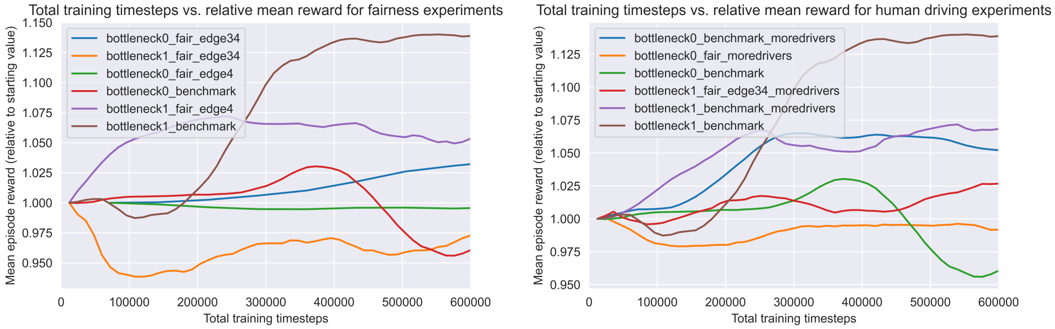
After validating results for the single-agent case, it could be interesting to explore how fairness considerations extend to the multi-agent setting, where it may be harder for agents to learn the coordinated lane-blocking behavior they exhibit when trained without fairness penalties. In particular, it may also be interesting to experiment with fairness penalties in the multi-agent setting where each agent is incentivized to travel through the bottleneck as quickly as possible and does not know the global throughput through the bottleneck.

In the bottleneck environment specifically, it could be interesting to explore fairness penalties that apply generally (e.g. penalize standard deviation of velocities among all lanes, not limited to certain segments of the bottleneck). The impact of transferring these penalties to other non-bottleneck environments can be another future area of research.

5.1.3 Human driver models

It would be interesting to test how our results change as a function of the number of human driver models added to the environment. Separately, it could be very informative to characterize the impact of specific parameter changes in the human driver model by running independent experiments for each parameter change. If a particular parameter is shown to be more influential, for example, that parameter could inform both future AV training and human driver education.

Finally, in this project we compared the results of previously-published training approaches (models trained and evaluated with only one human driver model) to the results of our experiments. But, it could be interesting to also approximate sim-to-real transfer capabilities by testing models trained



(a) Fairness experiments

(b) Human driver experiments

Figure 3: Mean episode rewards for experiments normalized by their initial values.

with one human driver model on environments containing many human driver models (approximating the real world).

Acknowledgements

We would like to thank the 6.884 Computational Sensorimotor Learning Spring 2021 course staff for helpful discussions and their comments on our project scope and progress. We would also like to thank Prof. Cathy Wu for helping us determine our project direction, and Zhongxia Yan for helping us get started with FLOW.

References

[1] Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst, 2018.

[2] Liting Sun, Cheng Peng, Wei Zhan, and Masayoshi Tomizuka. A fast integrated planning and control framework for autonomous driving via imitation learning, 2017.

[3] Sampo Kuutti, Richard Bowden, and Saber Fallah. Weakly supervised reinforcement learning for autonomous highway driving via virtual safety cages. *Sensors*, 21(6):2032, Mar 2021.

[4] Jianyu Chen, Zining Wang, and Masayoshi Tomizuka. Deep hierarchical reinforcement learning for autonomous driving with distinct behaviors. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1239–1244, 2018.

[5] Dean Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In D.S. Touretzky, editor, *Proceedings of Advances in Neural Information Processing Systems 1*, pages 305 –313. Morgan Kaufmann, December 1989.

[6] Florent Chiaroni, Mohamed-Cherif Rahal, Nicolas Hueber, and Frederic Dufaux. Self-supervised learning for autonomous vehicles perception: A conciliation between analytical and learning methods. *IEEE Signal Processing Magazine*, 38(1):31–41, 2021.

[7] Cathy Wu, Aboudy Kreidieh, Kanaad Parvate, Eugene Vinitsky, and Alexandre M. Bayen. Flow: Architecture and benchmarking for reinforcement learning in traffic control. *CoRR*, abs/1710.05465, 2017.

[8] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. Microscopic traffic simulation using sumo. In *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018.

[9] Eugene Vinitsky, Aboudy Kreidieh, Luc Le Flem, Nishant Kheterpal, Kathy Jang, Cathy Wu, Fangyu Wu, Richard Liaw, Eric Liang, and Alexandre M. Bayen. Benchmarks for reinforcement learning in mixed-autonomy traffic. In Aude Billard, Anca Dragan, Jan Peters, and Jun Morimoto, editors, *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pages 399–409. PMLR, 29–31 Oct 2018.

[10] Eugene Vinitsky, Nathan Lichtle, Kanaad Parvate, and Alexandre Bayen. Optimizing mixed autonomy traffic flow with decentralized autonomous vehicles and multi-agent rl. 2020.

[11] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1994.

[12] Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph Gonzalez, Michael Jordan, and Ion Stoica. RLlib: Abstractions for distributed reinforcement learning. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3053–3062. PMLR, 10–15 Jul 2018.

[13] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

Individual Contributions

All team members contributed to ideation and planning throughout the project. Ashay Athalye wrote code for human driver models, ran training experiments, and edited the presentation and report. Shannon Hwang wrote code for the initial benchmarks and evaluations, generated visualizations, and helped draft the presentation and report. Siddharth Nayak wrote code for the fairness reward penalties, performed the majority of evaluation, and drafted the presentation and report.