Imitation Learning for Autonomous Racing in F1 Tenth Cars

Adison De la Cruz, The University of British Columbia, Canada
Justin Kim, The University of British Columbia, Canada
Yanna Kwok, The University of British Columbia, Canada
Mo Ngai, The University of British Columbia, Canada

1 Introduction

In F1Tenth autonomous racing, traditional algorithms such as wall follow, gap follow, and pure pursuit follow a set of pre-defined rules and are designed to navigate in controlled scenarios. While rule-based algorithms require fewer computation resources and perform well in predictable scenarios, such as hallways without obstacles or a single car on the track, they struggle to handle highly dynamic environments. On the other hand, imitation learning (IL) learns from expert behaviour and is more adaptable to complex scenarios. Our goal is to explore the possibility of using imitation learning instead of rule-based algorithms in autonomous racing, specifically in the context of F1Tenth racing. We aim to create a neural network model that can learn from an expert to improve its abilities in path planning and speed adjustment.

2 Related Works

A previous study proposes MEGA-DAgger, a type of data aggregation (DAgger) designed to learn from multiple imperfect experts. The core of MEGA-DAgger lies in the data filter and conflict resolution. Since it relies on multiple imperfect experts, the data filter is essential in removing unsafe demonstrations and mitigating the risks of learning from undesired actions. However, interactive imitation learning approaches including DAgger, normally require expert feedback for every action it takes, making it resource-intensive and computationally heavy. We aim to develop an approach that is more computationally efficient while providing optimal performance. Furthermore, MEGA-DAgger learns from experts at a low speed, which leads to the model running at a low speed of around 1.5 m/s in the simulator and 0.75 m/s in the physical track. In evaluation, MEGA-DAgger competes against an opponent car using the pure pursuit algorithm at an extremely low speed of 0.75 m/s, yet it still shows a highly inefficient trajectory during overtaking. We plan to train the IL model with experts racing at a higher speed and focus on enhancing dynamic obstacle avoidance to have a more effective speed adjustment and overtaking capability than previous works.

Another research examines the performance of behaviour cloning in mini-autonomous racing. They develop a convolutional neural network (CNN) that relies on camera vision to capture the environment. In contrast, we implement behaviour cloning using laser scans instead of cameras

to track the surroundings. Thus, we replaced the CNN with a multilayer perceptron (MLP), which is more suitable when working with one-dimensional laser scan input.

3 Methodology

Our approach consists of three main components: data collection, data filtering, and training a multilayer perceptron (MLP).

3.1 Data Collection

We create a node that subscribes to both the scan and drive topic of the ego car to obtain information including laser scan, vehicle speed and steering angle. The scan topic provides an array of laser scans with a length of 1080 and the drive topic provides the speed and steering angle. Data is collected by running this node with the preferred algorithm as the expert, which is the gap follow algorithm in our case. Laser scan data captures the vehicle's current state and surrounding environment, whereas speed and steering angle provide information about the vehicle's actions and control signals. We also store the data in a pickle file instead of a CSV file. Pickle is optimized for saving and loading Python objects and is faster for round-trip serialization compared to CSV. Fig. 1 and Fig. 2 show the maps utilized for expert data collection.

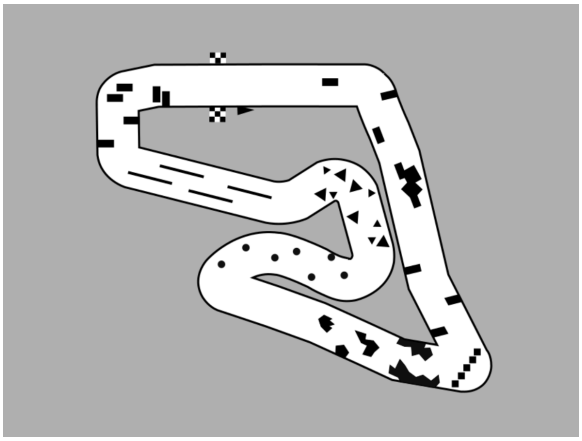

Fig. 1: Redbull Ring Obs map.

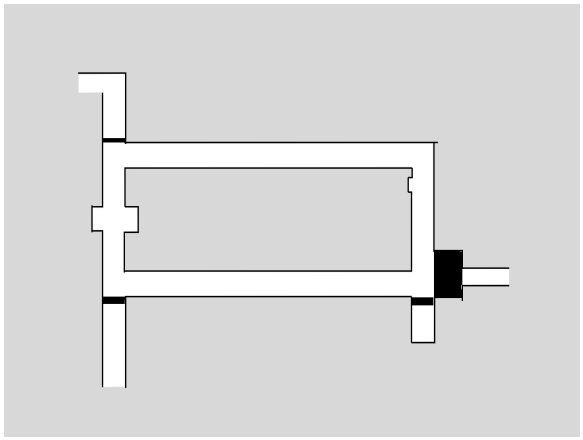

Fig. 2: Levine Blocked map.

3.2 Data Filter

To ensure the quality of the data, a data filter is applied before it is passed into the model. The data filter compares the minimum distance from the laser scans with a safety threshold. If the minimum distance is lower than the safety threshold, it indicates that the vehicle is either too close to an obstacle or has experienced a collision. The filter then removes the specified amount of data points collected before the trigger since we do not want to record past actions leading up to this trigger. Thus, the corresponding data is removed to prevent the model from learning from

undesired demonstrations. Moreover, a delay timer is added to pause the data collection process for a specified delay before resuming. This provides extra time for the imperfect expert to recover from the collision, either by itself or with external help, without recording these irrelevant data points.

3.3 Multilayer Perceptron

The construction of the MLP involves multiple hyperparameters as listed in Table 1. The model is designed to process laser scan data as input and predict outputs, speed and steering angle. To ensure compatibility with the input layer, we take the average of every $laser\ scan\ length\ /\ input\ layer\ size$ laser scan, resulting in laser scans with a length of input layer size. The data is then divided into smaller batches, defined by a predefined batch size. During the forward pass, input is passed into the neural network to produce outputs. Then, the loss function is applied to compute the prediction error and the weights are adjusted during the backward pass based on this error. The process is repeated for a specified number of epochs to improve the performance of the network, with the number of epochs carefully chosen to minimize loss without overfitting.

Number of hidden layers	Size of hidden layer
Size of the input layer	Size of the output layer
Loss function	Optimizer
Learning rate	Batch size
Number of epochs	

Table 1: List of MLP hyperparameters.

4 Result & Discussion

4.1 MLP Construction

To achieve optimal training outcomes for the IL model, we experiment with tuning various hyperparameters of the MLP. We begin by selecting the mean squared error (MSE) as the loss function, the adaptive moment estimation (Adam) optimizer, and a learning rate of 0.001. Over 60 combinations of layer sizes and the optimal hidden layer size are tested, with the optimal hidden layer size being around twice the input layer size. Based on the identified relationship, we further examine four possible combinations of input and hidden layer size and their performance as described in Table 2. The input layer size of 216 and hidden layer size of 512 are selected as the model imitates the expert's behaviour accurately without overfitting.

Input Layer Size	Hidden Layer Size	Performance
108	256	The model fails to clone the expert's behaviour in certain scenarios.
216	512	The model clones the expert's behaviour well and has the best performance.
540	1024	The model is overfitted and has a higher probability of crashing.
1080	2048	The model is overfitted and has a higher probability of crashing.

Table 2: Model performance with different input and hidden layer sizes.

The number of epochs determines how many times the entire dataset is passed through the network during training. A small value results in underfitting, where the model is not trained enough to identify patterns in data. On the other hand, a high value leads to overfitting, in which the model has poor generalizability when given unseen data. Through experimentation, we observe that the model underfits when the number of epochs is lower than 50 and overfits when the number of epochs is above 60. Therefore, the ideal number of epochs is found to be 55 in this case. Based on the findings as illustrated above, the complete MLP hyperparameters are presented in Table 3.

Number of hidden layers	2
Size of the input layer	216
Size of hidden layer	512
Size of the output layer	2
Loss function	MSE
Optimizer	Adam
Learning rate	0.001
Batch size	64
Number of epochs	55

Table 3: Values for MLP hyperparameters.

4.2 Evaluation

To evaluate model performance, we compare the predictions of the IL model and the expert's outputs across the two maps, as shown in Fig. 1 and Fig. 2. Fig. 3 and Table 4 demonstrate that the IL model successfully navigates multiple obstacles and narrow lanes while maximizing the performance in speed adjustments with the maximum speed of 6 m/s, achieving an average accuracy of 77.561% on the Redbull Ring Obs map. The IL model also has a faster lap time and a better trajectory at certain corners than the expert on the Redbull Ring Obs map as detailed in Fig. 3, Fig. 4, and Table 5. Therefore, the IL model has a stronger performance in path decisions and speed control on the Redbull Ring Obs map than the expert.

Trial	Accuracy %
1	78.882%
2	74.914%
3	80.278%
4	79.164%
5	76.022%
6	77.076%
7	67.999%
8	82.38%
9	81.147%
10	77.748%
Average	77.561%

Table 4: Accuracy percentage of the imitation learning model compared to its expert on the Redbull Ring Obs map.

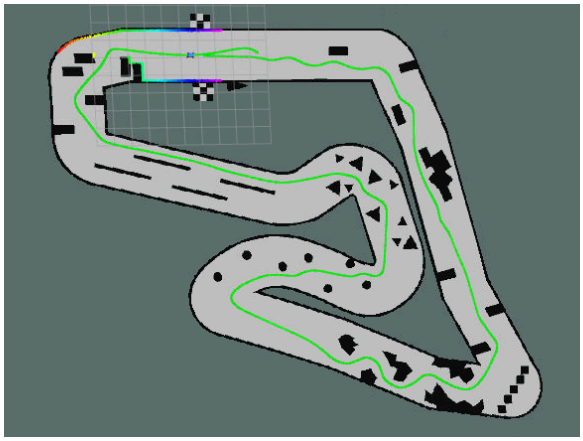

Fig. 3: IL model path on Redbull Ring Obs map.

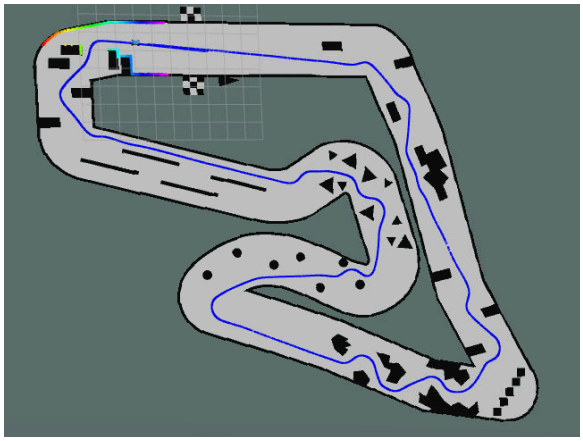

Fig. 4: Gap follow path on Redbull Ring Obs map.

Trial	Expert (Gap Follow)	Imitation Learning
1	33.684 sec	28.596 sec
2	33.599 sec	30.558 sec
3	35.155 sec	29.172 sec
Average Lap Time	34.146 sec	29.442 sec

Table 5: Lap time on Redbull Ring Obs map.

Unlike the Redbull Ring Obs map, the IL model has a slower lap time on the Levine Blocked map as shown in Table 6. However, as visualized in Fig. 5 and Fig. 6, the IL model exhibits a more efficient and smoother trajectory compared to the expert in the Levine Blocked map. Moreover, when the IL model competes in a head-to-head race with the expert, it not only successfully overtakes the expert but also showcases superior path planning in a dynamic environment.

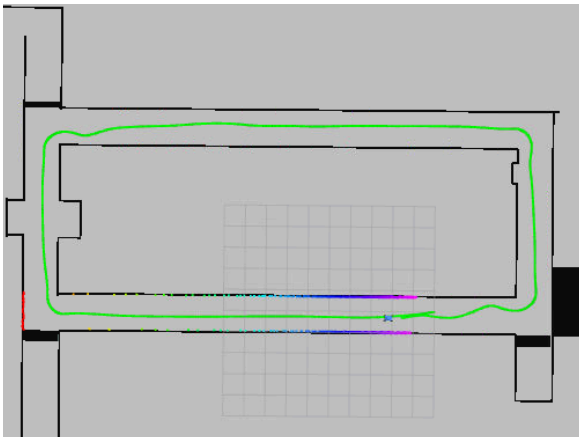

Fig. 5: IL model path on Levine Blocked map.

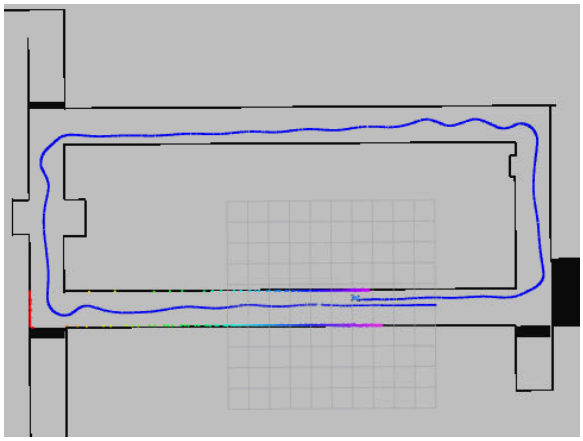

Fig. 6: Gap follow path on Levine Blocked map.

Trial	Expert (Gap Follow)	Imitation Learning
1	16.005 sec	17.432 sec
2	16.437 sec	16.714 sec
3	16.203 sec	16.871 sec
Average Lap Time	16.215 sec	17.006 sec

Table 6: Lap time on Levine Blocked map.

Ultimately, the IL model surpasses its expert in path planning and shows exceptional performance in challenging scenarios like the Redbull Ring Obs map. Hence, the precise replication of the expert's path decision and speed adjustment demonstrates the success of our IL model in behavioural cloning. Its consistent outperformance ensures safer and more effective driving in autonomous racing.

5 Limitations & Future Applications

5.1 Limitations

Although the imitation learning implementation successfully replicates the expert's behaviour and even outperforms the expert's performance in path planning and overtaking, it has significant limitations. A primary constraint is the absence of reinforcement learning. Since incorporating reinforcement learning is beyond the scope of our academic expertise in deep learning, we focus heavily on optimizing hyperparameters to construct the ideal neural network rather than accounting for erroneous behaviours. Consequently, the current approach lacks the capability to improve through experience and even crashes when faced with unseen situations in the worst case. Furthermore, the implementation lacks conflict resolution to decide the optimal action given an observation between multiple experts. This would result in the novice attempting to imitate multiple experts that have non-optimal actions. The data filter would ensure the dataset has no unsafe actions from experts, but non-optimal actions that pass the data filter will affect the performance of the novice policy in non-obvious ways. Lastly, the implementation can only be tested in the simulator due to technical challenges encountered during the process of obtaining the RViz map of the physical track. The RViz map is required to train the novice policy on observations mostly applicable to the specific circuit. Without the map, hardware limitations of the physical car and real-world constraints remain unaddressed, further narrowing the range of evaluation and applicability in real-world scenarios.

5.2 Future Applications

Overall, to construct a reliable and effective system for autonomous racing and real-world applications, further examination of the limitations is required. We aim to extend the current IL model to include reinforcement learning as well as conflict resolution to further optimize the performance and ensure robustness against unseen cases. In addition, we plan to develop a machine learning algorithm using a camera and convolutional neural network to evaluate its performance in comparison to imitation learning with LiDAR and MLP. This approach allows us to have a more comprehensive assessment of autonomous racing across multiple models and devices, which eventually enables a broader real-world application. Finally, we intend to train and evaluate the model in a physical setting to interact with real-world constraints. By transitioning to physical evaluation, we aim to assess the viability of imitation learning models beyond the limitations of simulations.

References

Sun, X., Yang, S., & Mangharam, R. (2023). MEGA-DAgger: Imitation Learning with Multiple Imperfect Experts. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2303.00638

Moraes, P., Peters, C., Sodre, H., Moraes, W., Barcelona, S., Deniz, J., Castelli, V., Guterres, B., & Grando, R. (2024). Behavior cloning for mini autonomous car path following. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2410.07209