Tim Komperda

Final Project CSC580

Word Count: 2656

**Exploring DisCor Parameters in Hot Lap Racing in Assetto Corsa**

Win on Sunday, sell on Monday is an old car manufacturing saying that described how winning in motorsports then leads to selling more cars. This led to manufacturers researching better technology which would then be proved on stages like the 24 hours of Le Mans. This led to the production of more efficient and reliable cars that could handle the harsh conditions of these chaotic conditions on the edge of grip while going as fast as possible. With such improvements happening due to the impact of racing, I think that autonomous driving should be a similar technology. There are many projects that try to solve the case of autonomous driving in a public environment, but very few that try to solve applications of autonomous driving in racing. This may be because many see self-driving in a normal case as much more valuable than obtaining an autonomous racing model. However, racing provides benefits that self-driving in a public environment does not. It provides a controlled environment that can benefit from iteration as racing is done on tracks where outside influence is low besides from track officials. It can be iterated by going from time trials which is a single car on a track going as fast as it can to grassroots wheel to wheel racing which allows for slower speed to be able to analyze how self-driving is reacting in multiple agent environments to endurance events where there are similar situations happening over long periods of time, but you have other cars that have different speeds, sizes, and cars can change in performance due to wear or change in conditions at any time.

However, track time and cars to run can get expensive, especially when using neural networks as neural networks take time to train and can perform maneuvers that can damage or even destroy the car. There is a cost-effective solution to this problem, simulation racing. Simulation racing has been around in some sense since 1982 when Namco developed Pole Position an arcade game that aimed to simulate Formula 1. This game did not have the in-depth physics simulation that modern simulators now have, such as tire wear, weather changing track conditions, weight transfer, etc. With that said modern simulations do a fairly good job of simulating reality and have been used to train autonomous driving models and one such paper has created an environment that can be used to compare and experiment: *A Simulation Benchmark for Autonomous Racing with Large-Scale Human Data*. In this project, I will be exploring the different parameters of the reinforcement learning model used by the paper to see what parameters impact training and results.

**Preliminary information**

The model used in the paper is a version of a Soft Actor Critic (SAC) algorithm called Distribution Correction (DisCor). “Soft Actor Critic (SAC) is an algorithm that optimizes a stochastic policy in an off-policy way, forming a bridge between stochastic policy optimization and DDPG-style approaches” (OpenAI Spinning Up). That is to say that Soft Actor Critic optimizes the Action policy in continuous action spaces by relying on an entropy factor or in this project it is known as the error learning rate. This allows the prediction model more leeway when exploring and means it does not end up over correcting and missing an optimal weight. Early versions of SAC also had value state optimization as well, but newer versions got rid of this as it was found to reduce performance and overcomplicate the model. Distribution Correction was made to additionally enhance performance by correcting possible issues that the training data may have by correcting the training data and reweighting transitions to provide improvements when there is multiple actions that need to be input at the same time and when the rewards are a bit messy which in this case is needed as the reward for continuous driving may not end up with giving optimal racing training.

There are some motorsports terms that may cause confusion when reading this paper and I would like to provide some clarity regarding these terms. The track or racecourse is the layout of the road being driven; in this project I used Autodromo Nazionale Monza also known as Monza for the city in Italy that this course is located in. Below is an image of the track’s layout and describes the different turns and where they are located and the names that they have been given over time.

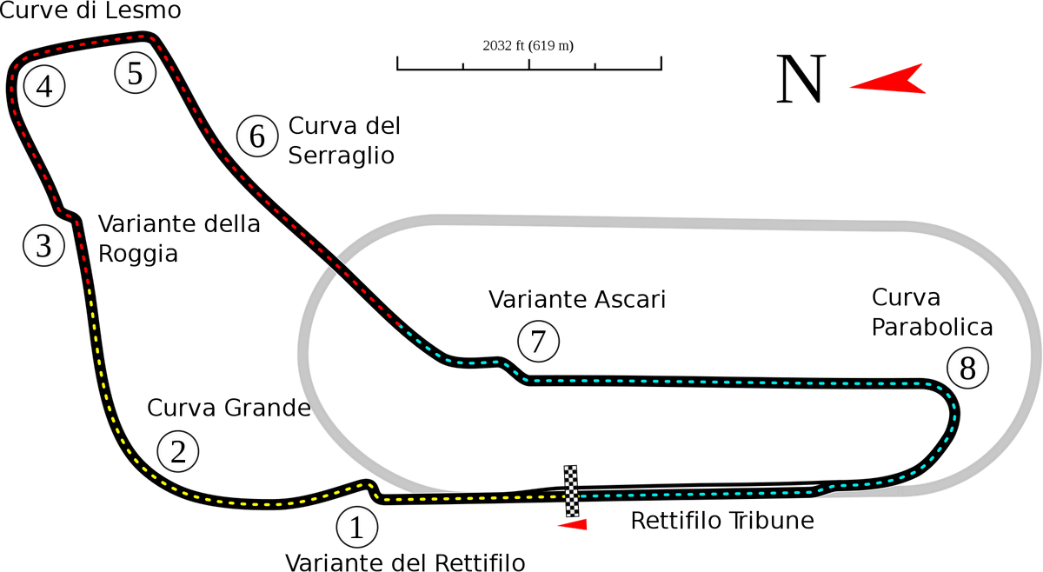


Figure 1 Layout of the Autodromo Nazionale Monza race track

There are many different types of cars that can be used and for this project I am using the BMW z4 GT3 car that is similar to a road going car with some modifications that allow it to go faster such as a large wing on the back to provide more downforce for grip. The term for going around a track in the fastest manner is often said to be quick that is that someone is quick if they can go around a track in the lowest amount of time. As fast is typically used to describe top speed attained rather than the optimization of driving around a track. When someone makes a race ending mistake or is unable to complete a lap this is known as a DNF which is a shorthand for did not finish.



Figure 2 The BMW z4 GT3 race car being used in training

**Why do I care to do this project?**

I am performing this testing because in the homework 5 assignment that regarded reinforcement learning I struggled to understand why my Q-learning neural network did not learn well enough to produce results. This struggle really made me want to test a reinforcement learning neural network that was already trained and see what parameters effect the training process. My thought process was to test all the parameters, but that was not feasible in such a short period of time.

**Methodology of experiment**

I ended up with testing 4 different parameters of the Soft Actor Critic: discount factor, number of combined steps, target update coefficient, and error learning rate. I am using the BMW z4 gt3 on the Monza track and using the model given from the gymnasium project that was trained with the SAC with DisCor for 54 million steps. From there I trained the model for 1 million more steps using the given parameters which consisted of a discount factor of 0.992, number of previous steps of 3, target update coefficient of 0.005, and an error learning rate of 0.0003 and then tested each of the other tested parameters to see what makes a difference and what does not.

**Results**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Control** | **Gamma = 0.999** | **Gamma = 0.985** | **Nstep =5** | **Nstep = 1** | **Target Update Coefficient = 0.007** | **Target Update Coefficient = 0.003** | **Error Learning Rate = 0.0006** | **Error**  **Learning Rate = 0.00015** |
| **Best Training Lap** | 1:52:789 | DNF | DNF | 1:52:872 | 1:55:773 | DNF | 1:52:565 | 1:53:195 | 1:53:743 |
| **Best Evaluation Lap** | 1:52:085 | 1:52:184 | 1:52:123 | 1:52:130 | 1:52:001 | 1:52:041 | 1:52:063 | 1:52:211 | 1:52:131 |

**Analysis**

*Discount Factor Experiment (Gamma)*

The gamma experiment was the one that I thought was the most obvious test to make this experiment also made the most sense as far as the results. The control gamma was the most optimal as expected. I will note that the experimental gammas were unable to complete a lap during the training, mainly struggling during the first chicane. But it was quite interesting to see that the lower gamma ended up performing better than the higher gamma.

*Number of Steps Experiment*

For this test, I wholly expected more steps to be better at connecting laps together. And this was indeed the case in the training example, the network managed to put up a lap that counted and was quick around the course. However, when it came to evaluation, more steps were detrimental to results. The network when trained on one step at a time managed to pull out the best result overall just 1 thousandth of a second above the 1 minute and 52 second mark. This network also seemed to perform the best around the chicanes, where other experiments lost some time.

*Target Update Coefficient Experiment*

This experiment was the most intriguing to me, the decreased target update coefficient performed the best in training and was quite quick and still improved in evaluation, but not by as much as other experiments, but it did lay down a better time than the control. The increased target update coefficient on the other hand could not get through the chicane at turn 1 during training. Every time it would just try to cut through the track and would get reset to the previous point which is what caused the DNF. In the evaluation though it put up the second-best time out of all experiments, as it was able to cleanly do that turn 1 chicane, although not as optimally as other experiments, so if it could train that chicane more, I’m certain it would break the 1 minute and 52 second barrier.

*Error Learning Rate Experiment*

For this experiment I asked ChatGPT what additional fields I should experiment with, and this was one of the responses where it recommended halving or doubling the original learning rate. And this experiment was not as exciting, as it was altering DisCor specifically, but it was interesting to see how that affected results. This ended up being the least successful in the evaluation portion, but it did manage to put novice level times in training.

Overall, the results of the experiments led that increasing the decreasing the target update coefficient in other words, slowing how much the target updates from iteration to iteration led to the best times when looking at training and evaluation combined. However, the number of steps when evaluating seemed to perform the best, I do wonder if this came down to a bit of randomness as all of the evaluation times have completed in better time then the Soft Actor Critic results described in the paper with this track and car.

**Future Ideas**

Another thing I would like to do is train each of the experiments but with a smaller change and multiple times so that I could show a continuous graph to see where the peak of training is to find the peak of each variable in this experiment. Especially the discount factor as its one of the most common variables used in reinforcement learning.

Things that could be done in the future is evaluating training with nothing but getting to higher number of steps like was done in the original paper with 70 million steps rather than the small number of steps that I completed. I also would like to see if I could get the model to learn to drive a track without a preliminary track layout like in the original paper. Other ideas I have are if I could train on a single human run and translate that data as a layout of the track even if the run is not optimal.

An issue that this current algorithm has is that it is in a very ideal situation, where it does not have to care about gas, the wear on the tires, change in the environment (rain) or any possible damage that could affect the performance of the car and it would be great to extend this performance to be able to handle a change in these situations.

Another route I would love to try is running this model with two separate computers together at the same time and experiment with how multiple agents will interact with each other. This could also be extended to more and more agents to a point where agents are competing and racing against each other. Where at that point there needs to be an extension of the model where it can recognize safety controls such as yellow flags and red flags where yellow flags require everyone to drive cautiously and red flags for everyone to return to the pit lane usually to clean up an accident.

**Conclusion**

Overall, the results of the experiments provided some great insights into how to improve my future reinforcement learning models. I would love to have more time and computing power to perform more and thorough experiments, but I am proud of this project and the results I was able to pull out from this. The process of setting up this project was the toughest part. There was a problem with the gymnasium version that the original project required to use, so I had to improvise. I ended up using gymnasium 0.22, but also because of the cuda version I had installed for my graphics card I had to change the typing extensions to 4.10. After that training on a checkpoint of the model was a bit of a problem as it required removing the replay buffer and making some other changes with the training file. I had a really fun time making the car work. The small differences in each of the models based on the different variable changes was really interesting to follow and I wish I documented more of the training process in writing. Overall though I would recommend using a gamma of 0.992 in the future as it provided the most results while training. The other variables I would recommend using are keeping the number of steps from 2 to 4 so that you can keep previous the past information but not rely on that past information too much. While lowering the target update coefficient showed promising results in training and evaluation I believe that when training from scratch, lowering that value increases the amount of training you would need to perform to reach a similar level of training, even if the lower value can hit that optimal time around the track in the end.

**Addendum**

I recorded a bunch of my training iterations of the neural network on YouTube and although some sound stuff was off because I got distracted at points and had other audio playing… Training paid off so much that I felt so successful just from getting the car to drive a clean lap, and then for it to get really close to times that I got when evaluating the 54 million checkpoint made me ecstatic. But I definitely see improvements that need to be made to my experiments as the training overall wasn’t done with the different values, but the experiments did provide some insight in how changing training over time helps.

Thank you again for enthusiastically supporting this project and I hope that some day I can utilize the knowledge I learned in this class to make something remarkable! I really enjoyed this course even though it was a bit disorganized at times. There was a lot to learn, and I got a lot better at getting a write up in. I appreciate the different guest talks although it would be nice to have maybe an extra lecture on perhaps your specialty of NLP.

**Bibliography**

Remonda, Adrian, et al. "A simulation benchmark for autonomous racing with large-scale human data." *Advances in Neural Information Processing Systems* 37 (2024): 102078-102100.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, G. Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, P. Abbeel, and Sergey Levine. Soft actor-critic algorithms and applications. ArXiv, 1812.05905, 2018.

Kumar, Aviral, Abhishek Gupta, and Sergey Levine. "DisCor: Corrective feedback in reinforcement learning via distribution correction." *Advances in neural information processing systems* 33 (2020): 18560-18572.

OpenAI Spinning Up -- <https://spinningup.openai.com/en/latest/algorithms/sac.html> Accessed 2025-03-17

**YouTube training videos:**

**Discount Rate = 0.999** [**https://youtube.com/live/7gQ0yS7nvlk**](https://youtube.com/live/7gQ0yS7nvlk?feature=share) **Discount Rate = 0.985** [**https://youtube.com/live/T2wfTO6GpoA**](https://youtube.com/live/T2wfTO6GpoA?feature=share)

**Number of Steps = 5** [**https://youtube.com/live/bP2nf-58aGM**](https://youtube.com/live/bP2nf-58aGM?feature=share)

**Number of Steps = 1** [**https://youtube.com/live/b3\_Z4C1cCPc**](https://youtube.com/live/b3_Z4C1cCPc?feature=share)

**Target Update Coefficient = 0.007** [**https://youtube.com/live/UlJad7pdnXo**](https://youtube.com/live/UlJad7pdnXo?feature=share)

**Target Update Coefficient = 0.003** [**https://youtube.com/live/UlJad7pdnXo**](https://youtube.com/live/UlJad7pdnXo)

**Error Learning Rate = 0.0006** [**https://youtube.com/live/cCRKT8KUDBM**](https://youtube.com/live/cCRKT8KUDBM)

I was too tired and forgot to record the control and error rate of 0.00015 sorry!