OTONOM ARAÇLAR İÇİN ARAÇ KONTROL SİSTEMİ GELİŞTİRİLMESİ

Ahmet Çağrı Kılıç, Furkan Akşit
Bilgisayar Mühendisliği Bölümü
Yıldız Teknik Üniversitesi, 34220 Istanbul, Türkiye
{ahmet.cagri.kilic, furkan.aksit }@std.yildiz.edu.tr

***Özetçe* —Otonom Yarışlar, otomobil yarışlarının gelişen bir türüdür. Bu tür yarışlarda araçlar yarış pilotları yerine geçen özelleştirilmiş bilgisayarlar tarafından kontrol edilir. Bu yarışların asıl amacı sürücüsüz araçların geliştirilmesine yönelik ilgi ve rekabetin arttırılması ve güvenli, konforlu sürüç deneyimleri yaşatabilecek sürücüsüz araç kontrol sistemlerinin ortaya çıkarılmasına katkıda bulunmaktır. Bu çalışmada sürücüsüz araç yarışlarına katılınması planlanarak, sürücüsüz araç kontrol sistemi üzerinde çalışılmıştır. Çalışmanın yürütülebilmesi için TORCS simülasyon ortamı kullanılmış ve sistem Derin Pekiştirmeli Öğrenme algoritması temel alınarak geliştirilmiştir. Çalışmada literatürdeki diğer çalışmalar incelenmiş vefarklı yapay sinir ağı mimarileri birbirleriyle detaylıca karşılaştırılmıştır.**

Anahtar Kelimeler—Anahtar Kelimeler— otonom sürüş sistemi, derin pekiştirmeli öğrenme, torcs, yapay sinir ağı

***Abstract*—Autonomous or self-driving racing is an evolving sport of racing ground-based wheeled vehicles, controlled by computer. The main purpose of these races is to increase the interest and competition for the development of autonomous vehicles and to contribute to the development of autonomous vehicle control systems that can provide safe and comfortable driving experiences. In this study, it was planned to participate in autonomous vehicle races, and the autonomous vehicle control system was studied. TORCS[1] simulation environment was used to carry out the study and the system was developed based on the Deep Reinforcement Learning algorithm. In the study, other studies in the literature were examined and different artificial neural network architectures were compared in detail.**

Keywords—autonomous driving, deep reinforcement learning,torc, artificial neural network

I. Introduction

Cars have been an indispensable medium of transportation in our modern lives. Advances in the automotive industry made cars more affordable and easily accessible. The number of deaths on the world's roads remains unacceptably high, with an estimated 1.35 million people dying each year according to the report on road safety published by the World Health Organization. [2] The main cause of the high number of death and injuries on the road is the human factor. Elimination of human factors from the road is thought to be the most promising solution to the high number of deaths considering road safety.

Partially automated safety features such as lane-keeping assists, adaptive cruise control, traffic jam assist, and self-parking will be part of roads in the United States of America until 2025 according to The National Highway Traffic Safety Administration of United States of America[3]. Fully automated vehicles and roads are thought to be a part of our lives after 2025.

Several autonomous car racing competitions aim to take people's attention to the development of automated driving. The TORCS Racing Board hosts a competition on its website among players in the TORCS community. We have developed a vehicle control system using a deep reinforcement learning approach.

In our project, we adopted a real-life approach that simulates a Formula 1 pilot's journey before the event that is being undertaken. Track testing is the process of collecting data while driving on a race circuit. This is the simplest form of testing since all it involves is the driver driving around a circuit several times to be able to get used to the race track. Besides, they collect telemetry data which allows a team to work out if a new part of the car improves the car. The vehicle control system uses the experiences which are collected by driving around a track to move inside the track. It is a positive feedback loop in which the vehicle is moving by its experience in track and at the same time, it gains new experiences by its every movement.

Our trials conclude the system's ability to learn depends on several factors such as the difficulty of the race track, whether a race track has lanes, or even if the race track is sloped or not. In our project, we analyze different factors that interfere with the vehicle control system's ability to learn. In the end, we were able to have a base model that can reach the end of a map and that can be used for a base model for transfer learning in other maps.

II. related works

According to our researches, we have found that the issue of developing a vehicle control system for autonomous vehicles has been studied using different algorithms such as Imitation Learning [4], Monte Carlo Tree Search [5] and Deep Q Learning [4]. While some of these studies are carried out in the simulation environment, some of them are carried out in the physical world using appropriate equipment. In addition, line tracking [6], detection of other vehicles [7] etc., developed to assist autonomous vehicle control systems. There are also many auxiliary systems developed in the fields. In general, the creation of a suitable dataset in machine learning projects can take quite a time in the real world, while the creation of the relevant training dataset in

the simulation environment can be carried out very quickly. According to the information obtained from the project named CAD2RL [8], it has been seen that when the vehicle control system developed in the simulation environment is tested in the real world, very successful results can be obtained. In addition to the autonomous vehicle control systems, projects on other topics were studied using Deep Q Learning like Playing Atari using Deep Q Learning by OpenAI that is capable of human-level performance on many Atari video games using unprocessed pixels for input. To do so, deep neural network function approximators were used to estimate the action-value function. [9].

III. Autonomous Driving System

Deep Reinforcement Learning or Deep Q Learning is a reinforcement learning algorithm that combines artificial neural networks with a reinforcement learning architecture that enables software-defined agents to learn the best actions possible in the environment in order to attain their goals. It takes a high dimensional input like sensory inputs or images and approximates q values using deep neural networks. Real q values can be calculated using the formula shown in (1). This method allows developers to train deep neural networks by trial and error approach.

$$Q^{new}(s_t, a_t) = r_t + \gamma \times \max Q(s_{t+1}, \alpha) \quad (1)$$

Figure 1 Q Learning Formula

In our study, we developed an autonomous driving system that can learn from its mistakes using a deep q learning algorithm. It uses images from the simulation environment as current state input. The agent chooses the best action to move based on the agent's previous experiences. Our deep neural network architecture is shown in Figure 7. Our neural network uses 84x84 RGB images as input states and outputs the best action among 12 possible discrete actions to move to the next state. It decides the best action by calculating each action's possible rewards and it tries to get the highest reward possible. During training, the decision for action to move the car is not always made by the deep neural network. The system uses the epsilon-greedy approach to takes random actions to explore the environment and tries to get higher rewards. The randomness of the action choice dependent on a variable called epsilon. At the beginning of the training, action choice is completely random. Later in training, it gets less random and it is more dependent on deep neural network choices. TORCS does not have an internal reward calculator, so we need to design our reward function.This function aims to maximize longitudinal velocity, minimize transverse velocity, and we also penalize the agent if it constantly drives the very off-center of the track. We formulate our reward function shown in Formula (2):

$$R_t = V_x \cos(\theta) - V_x \sin(\theta) - V_x |trackPos| \quad (2)$$

Figure 2 Reward Function

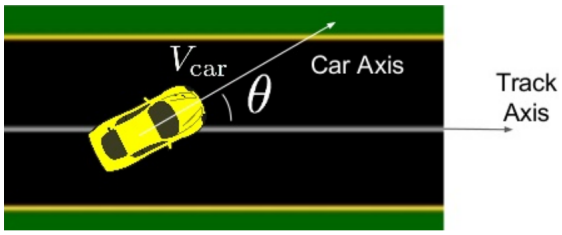

Figure 3 A visualization of car

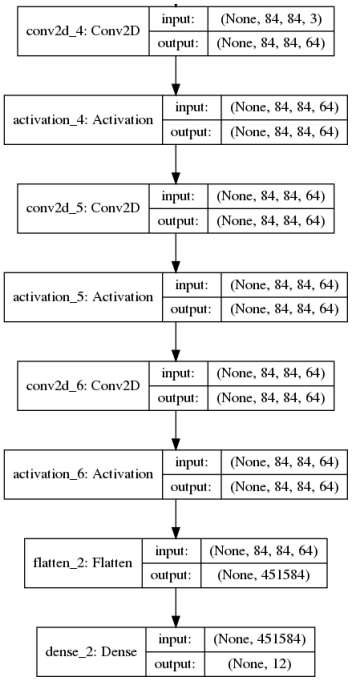

Figure 4 Convolutional Neural Network Architecture

A. TORCS Simulation System

Autonomous driving system is developed in a simulation system because of safety measures and economical reasons. In choosing the right simulation system, we had several criteria as following:

- Low Usage of System Resources
- Client/Server Architecture
- Python API
- Constant Image Feed From Sensors on Car
- Open Source

1) Low Usage of System Resources: These criteria have been chosen considering our previous study on the same subject with a different simulation system. In our previous system, the high usage of system resources was our limiting factor in developing such a system. A simulation system that had advanced graphics and physics engines required high usage of graphics cards and memory. Advanced graphics engine provides better quality images and that requires several steps of pre-processing. In the end, our deep reinforcement learning approach requires training of a convolutional neural network and that requires high usage of graphics card and memory. It required us to use a cloud computing machine with advanced graphics cards. It is a costly solution that is difficult to deploy and maintain.

2) Client/Server Architecture : Previous simulation system was developed considering Client/Server. This feature allows developers produce easily scalable solutions.

3) Python API: Uniformity in programming languages is required for easier maintenance of code. Learning agents are developed using Python programming language. Python programming has been chosen to interact with the simulation system to maintain the uniformity of programming languages in the project.

4) Constant Image Feed From Sensors on Car: In our approach, convolutional neural network is being fed with current state images to predict the next action. In order to do that, simulation ought to supply images of the road on current state.

5) Open Source: Simulation being open-source criterion has been added due to solely economical reasons.

IV. Experimental Results

In our study, we measure the success of the established system by analyzing our agent's behavior and the amount of reward that it can get in different conditions. We analyze our agent's behavior using different hyperparameters, different maps, and different training methods.

A. Changing hyperparameters

Firstly, we analyze the behavior of our agent with following learning rate values: 0.001,005 and 0.01 Increasing the learning rate from 0.001 to 0.005 enabled the learning agent to succeed earlier. However when the learning rate increased to 0.01, it will be seen that the learning agent always performs the same movement, it cannot distinguish between different situations in the required environment and remains at the local minimum point. In the training of the learning agent, we found out that randomness of the action choice effects model significantly. Dropping the randomness of the action choice very slowly by multiplying the value of 0.9985 causes the learning agent to take too many random actions. For this reason, the learning agent cannot get too far from the start of the race, so the reward it gets is around 400. Rapidly decreasing the randomness of the action choice by multiplying 0.95 enables the agent to act in a way that can receive more rewards by using the information obtained. In this way, the agent can complete a part of the race and can reach 10000-12000 points in points.

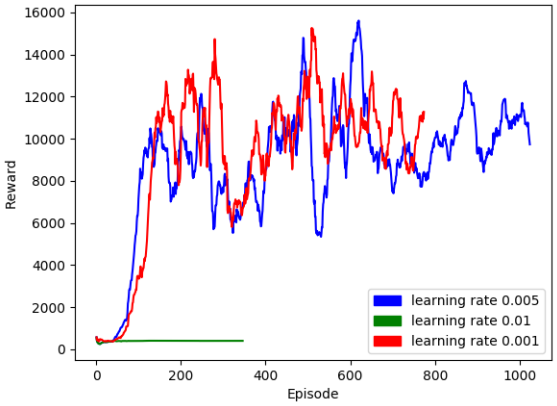

Figure 5 Learning Rate Comparison

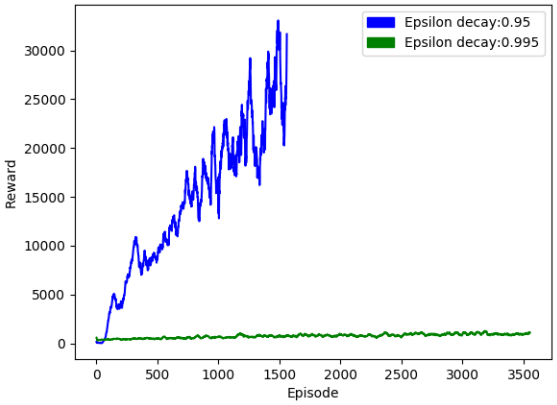

Figure 6 Randomness of action choice effects on agent

We analyze different neural network architectures to ensure convergence of the network. During development, we tried a few neural network architecture that could not gain enough reward to continue training, so we do not have enough data to visualize these neural network's success. We analyze two promising convolutional neural network architecture that is similar to each other. One architecture is shown in Figure 7, the other architecture the same but it has pooling layers between each convolution layer. We analyze that removing pooling layers has a major positive effect on the agent's behavior because the pooling layer removes too much necessary data from an already low pixel density image.

B. Changing Maps

We analyze our agent's behavior in three different tracks. We found out that using only visual data as input state is not enough for agent to learn how to drive in different tracks. Change in the visual input causes agent to behave unexpectedly.In order to learn how to drive in different tracks, agent has to be trained in these tracks too. We trained learning agent in two different track and create

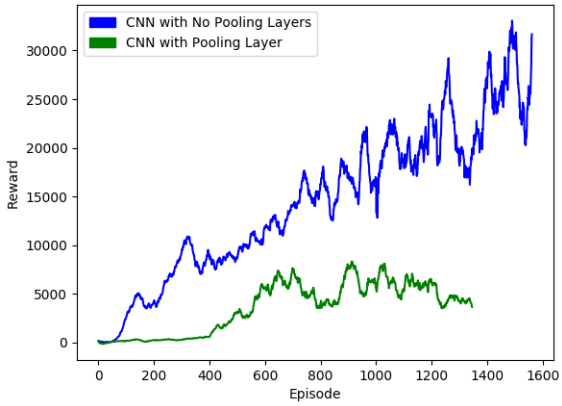

Figure 7 Convolutional Neural Network Architecture Comparison

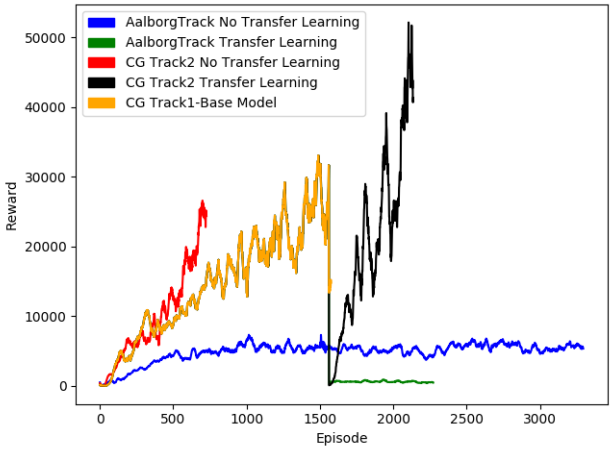

Figure 8 Rewards In All Models

new models for each track using transfer learning. One of these models trained in similar road conditions but other model trained in different road conditions. We also trained the agent additional two different track without transfer learning. We found out that if base model and the new track has similar road conditions, model that we developed using transfer learning was remarkably better than models that developed without transfer learning and gained high rewards earlier than others. If base model and the new track has different road conditions, agent could not learn how to drive in new environment. Rewards of all of the models mentioned shown in Figure 8

V. Conclusion and Future Work

This research analyzes a deep reinforcement approach that can autonomously drive in the TORCS simulation system. Trials with different hyperparameters have shown that the learning rate, randomness coefficient, and pooling layer has a great effect on the success of the model. While too high the learning rate has an effect of a car that made the car stop, too low learning rate has an effect that the model learns slower. The existence of a pooling layer has thought to remove too much necessary data from already low pixel density images. While too low randomness of the action choice has an effect of trying already known actions and not discovering possible actions that may have higher success, too high randomness has an effect of constantly trying new actions and not verifying the success of actions that are already taken.

During the training of models in different maps, It has been realized that difficulties of maps, the slope of the road, and event the differences in scenery affect the average reward of the model. The reason of these effects is thought to be the training method of models. Feeding only with state images makes the model vulnerable to little changes in the image. The difficulty of the map is defined by the number of bends on the track. The number of bends means a wider range of verified experiences which a model should have.

In this project, a base model has been created. This base model allows us to use the transfer learning method in different maps. The transfer learning method is proved to be more successful in new maps than starting without any experience.

In the future, we would like to use more sensory data than the only image to able to train a model that has little sensitivity to change in the scenery. More sensory data can allow a model to learn how to drive a car rather than to learn how to act in a situation that has been experienced before. We hope that we can have an effect on the development of autonomous vehicle control systems with this research.

References

[1] "Torcs simulation." [Online]. Available: http://www.berniw.org/trb/

[2] "Global status report on road safety 2018," *WHO Publications*, 2019. [Online]. Available: https://www.who.int/publications/i/item/global-status-report-on-road-safety-2018

[3] "Automated vehicles for safety," *The National Highway Traffic Safety Administration*. [Online]. Available: https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety

[4] M. K. Simon Kardell, "Autonomous vehicle control via deep reinforcement learning," 2017.

[5] J. Fischer, N. Falsted, M. Vielwerth, J. Togelius, and S. Risi, "Monte-carlo tree search for simulated car racing," 2015.

[6] R. Nourine and S. F. Khelifi, "Vision-based lane boundaries following for autonomous vehicle guidance," 2006.

[7] A. E. Tolga Birdal, "Real-time automated road, lane and car detection for autonomous driving," 2007. [Online]. Available: https://arxiv.org/pdf/1611.04201.pdf

[8] S. L. Fereshteh Sadeghi, "Cad2rl: Real single-image flight without a single real image," 2017. [Online]. Available: https://arxiv.org/pdf/1611.04201.pdf

[9] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *ArXiv*, 2013. [Online]. Available: https://arxiv.org/pdf/1312.5602.pdf