T.C.
YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

PEKİŞTİRMELİ ÖĞRENME PROBLEMLERİNDE KEŞİF VE GENELLEŞTİRME YÖNTEMLERİ

Kıvanç GÜÇKIRAN

YÜKSEK LİSANS TEZİ
Elektronik ve Haberleşme Mühendisliği Anabilim Dalı
Elektronik Programı

Danışman
Dr. Öğr. Üyesi Bülent BOLAT

Şubat, 2020

T.C.
YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

PEKİŞTİRMELİ ÖĞRENME PROBLEMLERİNDE KEŞİF VE GENELLEŞTİRME YÖNTEMLERİ

Kıvanç GÜÇKIRAN tarafından hazırlanan tez çalışması 05.02.2020 tarihinde aşağıdaki jüri tarafından Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Elektronik ve Haberleşme Mühendisliği Anabilim Dalı Elektronik Programı **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Dr. Öğr. Üyesi Bülent BOLAT
Yıldız Teknik Üniversitesi
Danışman

Jüri Üyeleri

Dr. Öğr. Üyesi Bülent BOLAT, Danışman
Yıldız Teknik Üniversitesi _____

Doç. Dr. Umut Engin AYTEN, Üye
Yıldız Teknik Üniversitesi _____

Prof. Dr. Mehmet SAĞBAŞ, Üye
Yeni Yüzyıl Üniversitesi _____

Danışmanım Dr. Öğr. Üyesi Bülent BOLAT sorumluluğunda tarafımca hazırlanan Pekiştirmeli Öğrenme Problemlerinde Keşif ve Genelleştirme Yöntemleri başlıklı çalışmada veri toplama ve veri kullanımında gerekli yasal izinleri aldığımı, diğer kaynaklardan aldığım bilgileri ana metin ve referanslarda eksiksiz gösterdiğimi, araştırma verilerine ve sonuçlarına ilişkin çarpıtma ve/veya sahtecilik yapmadığımı, çalışmam süresince bilimsel araştırma ve etik ilkelerine uygun davrandığımı beyan ederim. Beyanımın aksinin ispatı halinde her türlü yasal sonucu kabul ederim.

Kıvanç GÜÇKIRAN

İmza

TEŞEKKÜR

Yüksek lisans eğitim dönemi boyunca desteğini esirgemeyen tüm Yıldız Teknik Üniversitesi bünyesindeki akademisyenlere minnettar olduğumu belirtmek isterim. Çağdaş eğitim bilinciyle yoğrulmuş ve güncel bilimsel faaliyetlerin farkında olarak akademik katkılarda bulunan üniversitemizde eğitim almış olmaktan memnuniyet duymaktayım.

Gerek enstitü, gerek fakülte personeline her türlü destekleri için teşekkür ederim. Beni bir çok açıdan yönlendiren Prof. Dr. Tülay YILDIRIM'a ve elbette danışmanım Dr. Öğr. Üyesi Bülent BOLAT'a tez, akademik, idari ve bir çok diğer konuda yardımı ve desteği müteşekkirim.

Son olarak, bu çalışmanın çıkmasına vesile olan İstanbul Teknik Üniversitesi Bilgisayar Mühendisliği bölümüne, akademisyen Doç. Dr. Nazım Kemal ÜRE'ye, takım arkadaşlarım Can ERHAN ve Onur KARADELİ'ye teşekkür ederim.

Kıvanç GÜÇKIRAN

İÇİNDEKİLER

SİMGE LİSTESİ vi

KISALTMA LİSTESİ vii

ŞEKİL LİSTESİ viii

TABLO LİSTESİ ix

ÖZET x

ABSTRACT xi

1 GİRİŞ 1

1.1 Literatür Özeti . . . 1
1.2 Tezin Amacı . . . 3
1.3 Hipotez . . . 3
1.4 Temel Kavramlar . . . 3
1.4.1 Pekiştirmeli Öğrenme . . . 3
1.4.2 Keşif Yöntemleri . . . 6
1.4.3 Genelleştirme Yöntemleri . . . 7
1.4.4 Algoritmalar . . . 8
1.4.5 TORCS . . . 9

2 METODOLOJİ 11

2.1 Problem Tanımı . . . 11
2.2 Mimari . . . 12
2.2.1 SAC . . . 12
2.2.2 Rainbow DQN . . . 13
2.3 TORCS . . . 15
2.3.1 Ödül Düzenlemesi . . . 15
2.3.2 Keşif . . . 17
2.3.3 Genelleştirme . . . 17
2.3.4 Eylem Uzayı . . . 18

3 TESTLER ve KARŞILAŞTIRMA **20**

4 SONUÇ ve ÖNERİLER **26**

Kaynakça **28**

Tezden Üretilmiş Yayınlar **31**

SİMGE LİSTESİ

θ	Araba ile yol arasındaki açı
$trackpos$	Arabanın yolun ortasına olan uzaklığı
s'	Bir sonraki durum
V_x	Doğrusal hız
s	Durum
$P_a(s,s')$	Durum geçiş dağılımı
V	Durum değeri
S	Durum uzayı
ϵ	Epsilon
a	Eylem
Q	Eylem değeri
A	Eylem uzayı
G	Getiri
γ	İndirim oranı
T_a	Ortam sonlanma fonksiyonu
R	Ödül
$R(s)$	Ödül fonksiyonu
$sigmoid$	Sigmoid fonksiyonu
V_y	Yanal hız
t	Zaman adımı

KISALTMA LİSTESİ

CNN	Convolutional Neural Network
CPU	Central Processing Unit
C51	Categorical Deep Q Network
DDPG	Deep Deterministic Policy Gradient
DQN	Deep-Q Network
GPU	Graphics Processing Unit
Impala	Importance Weighted Actor-Learner
KL	Kullback-Leibler
LSTM	Long-Short Term Memory
NEAT	Neuro Evolution of Augmenting Topologies
PER	Priority Experience Replay
ReLU	Rectified Linear Unit
RL	Reinforcement Learning
RPM	Rotations Per Minute
rtNEAT	Real-Time Neuro Evolution of Augmenting
SAC	Soft-Actor-Critic
TanH	Hyperbolic Tangent Function Architecture
TORCS	The Open Racing Car Simulator

ŞEKİL LİSTESİ

Şekil 1.1 ϵ değerinin zaman içinde değişim örnekleri 7
Şekil 1.2 Örnek bir yarışma 10
Şekil 2.1 SAC mimarisi 13
Şekil 2.2 DQN mimarisi 14
Şekil 2.3 Fren-Dene dağılımı 18
Şekil 3.1 Eğitim süresince skor değişimi 20
Şekil 3.2 Eğitim süresince maksimum ve ortalama hız değişimi 21

TABLO LİSTESİ

Tablo 2.1 Ayrık eylem uzayı 19
Tablo 3.1 Skorların standart sapma ve azami değerleri 21
Tablo 3.2 En yüksek ve ortalama hızların standart sapma ve azami değerleri . 21
Tablo 3.3 Pistlerdeki hız ve skor değerleri 22
Tablo 3.4 Cardamone ve ekibinin çalışması ile tur sürelerini karşılaştırma . . . 23
Tablo 3.5 Munoz ve ekibinin çalışması ile tur sürelerini karşılaştırma 23
Tablo 3.6 Verma ve ekibinin çalışması ile tur sürelerini karşılaştırma 24

ÖZET

Pekiştirmeli Öğrenme Problemlerinde Keşif ve Genelleştirme Yöntemleri

Kıvanç GÜÇKIRAN

Elektronik ve Haberleşme Mühendisliği Anabilim Dalı
Yüksek Lisans Tezi

Danışman: Dr. Öğr. Üyesi Bülent BOLAT

Makine öğrenmesinin alt başlığı olan pekiştirmeli öğrenme, bir hedefe dair en uygun politikayı bulmayı amaçlayan ve çeşitli algoritmaların birleşiminden oluşan bir yöntemler bütünüdür. Bu hedef bir çok alt görev tanımından oluşabileceği gibi yalnızca bir başarımdan oluşabilir.

Bir çok makine öğrenmesi konsepti gibi, pekiştirmeli öğrenmenin de kendisine has bazı problemleri bulunmaktadır. Bu problemler hedeflenen politikanın en uygun olmayan yapıya yakınsamasına yol açabilmektedir. Bu yakınsama, ortamın dinamiklerini yeterince keşfetmeden karara ulaşmasından yahut, karşılaşılmayan dinamikler ile karşılaşıldığı durumda genelleyememesinden kaynaklanmaktadır.

Bu çalışmada, TORCS ortamında otonom bir şekilde yarışan bir başarılı ve uygun bir politika bulunmaya çalışıldı. Keşif ve genelleştirme problemlerini en aza indirgemek için bir çok yöntem dikkate alındı ve kullanıldı. Bir çok iyileştirmeyle beraber Soft-Actor-Critic algoritması kullanılarak başarılı bir politika elde edildi.

Anahtar Kelimeler: Pekiştirmeli öğrenme, torcs, sürücüsüz araba, keşif, genelleştirme

ABSTRACT

Exploration and Generalization Methods in Reinforcement Learning Problems

Kıvanç GÜÇKIRAN

Department of Electronics and Communication Engineering
Master of Science Thesis

Advisor: Dr. Bülent BOLAT

Reinforcement Learning, which is a sub-component of machine learning algorithms, is a collection of algorithms which try to find the optimal policy regarding a goal. This goal might be overcoming a plethora of tasks or a single achievement.

Like many machine learning concepts, Reinforcement Learning algorithms have some drawbacks which make the method converge to a non-optimal policy. These drawbacks mainly occur because of the early exploitation of the environment by the policy or lacking the ability of generalizing to unseen dynamics of the environments.

In this study, we have tried to achieve an optimal autonomous car racing policy in The Open Racing Car Simulator environment. Multiple approaches are taken into account to achieve exploration and generalization over environment. We had significant success with Soft-Actor-Critic algorithm with improvements over several environmental factors.

Keywords: Reinforcement learning, torcs, self driving car, exploration, generalization

YILDIZ TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

1
GİRİŞ

Bu bölümde, tezin üzerine bina edildiği sistemler konusunda kısa bilgiler verilecek, sonrasında literatür özeti, tezin amacı ve hipotez açıklanacaktır.

Endüstri ve bilimin, elektronik ve bilişim konularında çok hızlı gelişmesini takiben otonom sistemler akademinin gündemde önemli bir yer oluşturmaktadır. Derin öğrenme ve bilgisayarlı görü tekniklerinde yaşanan gelişmelerin sonrasında özellikle otonom araçların sokaklarda görülmesi uzun zaman almayacaktır. Fakat bu sistemlerin öncelikle deneme ve simülasyon ortamlarında eğitilmesi şarttır. Zira gerçek hayatta deneyimlerden öğrenmek hem yavaş öğrenmeye yol açacak hem de büyük zararlara yol açabilecek riskler taşıyabilecektir.

Yapay zeka sistemlerinde üç tür eğitim yöntemi bulunmaktadır. Bunlardan ilki veriye ve bu veriye ait etiketlerin bulunduğu eğitim olan eğiticili öğrenmedir. Bir diğeri, veriye sahip olunduğu fakat etiketlerin olmadığı, eğiticisiz öğrenmedir. Pekiştirmeli öğrenmede ise ne veri ne de etiket bulunmaktadır. Otonom ajanlar eğitimlerini ortamdan aldıkları gözlemler vasıtasıyla oluştururlar. Derin öğrenme pratiklerinin pekiştirmeli öğrenme ile buluşmasıyla beraber, pekiştirmeli öğrenme problemlerinde gerçek hayata taşınabilecek başarılar elde edilmeye başlanmıştır.

1.1 Literatür Özeti

TORCS (The Open Racing Car Simulator) ortamı kullanılarak birçok Pekiştirmeli öğrenme modeli kendini kanıtlama ve yarışma fırsatı bulmuştur. Birçok çalışma.

Sıkıştırılmış ağ yapılarını evrimsel algoritmalarla geliştirerek sonuç arayan bir çalışmada [1], TORCS'un sağladığı görüntülerle başarılı bir eğitim sağlanmıştır. Fakat burada, TORCS üzerinde elde edilen sonuçtan daha çok sıkıştırılmış ağ yapılarının performansı incelenmiştir. Zira eğitim yalnızca tek pist üzerinde yapılmıştır.

Çalışmalar arasında belki de en ilgi çekici olanı, sürüş davranışlarını öğrenmeye

çalışan araştırma [2] olmaktadır. İlgili çalışmada, ortamın aerodinamik davranışları, geç fren davranışı gibi öğeler, Tablolu Q Öğrenmesi yöntemiyle öğrenmeye çalışılmış ve bunda başarılı olunmuştur. Fakat eğitim yapılan pist sayısı aracın hızı gibi bilgiler çalışmaya eklenmemiştir, zira çalışmanın amacı sürüş davranışlarını öğrenebilmektir. Ayrıca sürüş boyunca engelden kaçınmak da önemli olmaktadır. Bu konuda yapılan diğer bir çalışmada Deep Deterministic Policy Gradient algoritması kullanılarak çarpışmadan kaçınma amaçlanmıştır [3]. Yine sürücü kararlarını modelleyen bir çalışmada [4], otoyol sürüşü ve trafikte sürüş davranışları ile eğitim yapılmıştır. Benzer bir çalışmada, hızlı gitmek değil yolu düzgün takip eden sürüş davranışına sahip bir ajan eğitmek amaçlanmıştır [5].

TORCS üzerine yapılan çalışmalardan NEAT [6] ve rtNEAT [7] algoritmalarını kullanarak evrimsel bir şekilde eğitim sağlayan bu araştırma [8], ajanların performanslarını ve genelleştirme başarılarını, bir pist üzerinde eğitim yaparak ve başka bir pist üzerinde deneyerek göstermiştir. Genelleştirmeye ek olarak bu çalışmada, yeni pistte eğitim yapılmak istendiğinde, diğer pistte eğitilen ajandan bir öncel alarak eğitime başlanmış ve bu şekilde başarı sağlamıştır.

Derin Pekiştirmeli Öğrenme kullanarak otonom yarışabilen bir araç eğitmeyi hedefleyen Stanford'dan çalışmada [9], eğitimler zor aalborg pistinde, testler speedway ve alpine-1 pistinde yapılmıştır. Ajan başarılı bir şekilde sürüşü gerçekleştirebilmiştir.

Carneggie Mellon Üniversitesinde yapılan çalışmada [10], benzer bir mimari kullanılmış ve DDPG ile eğitim sağlamıştır. Eğitilen ajanın, düzlüklerde problem yaşadığı fakat dönüşlerde bu açıklarını kapattığı gözlemlenmiştir. Eğitimler ve test aalborg pistinde yapılmıştır. Fakat testlerde ajan tek başına değildir.

Verma ve ekibi tarafından yürütülen bir diğer çalışmada [11], yorumlanabilir ve teyit edilebilir pekiştirmeli öğrenme altyapısı hazırlanmaya çalışılmıştır. Bu çalışmada test sistemi olarak TORCS ortamı kullanılmış ve başarılı sonuçlar elde edilmiştir.

Pekiştirmeli Öğrenme algoritmaları arasında büyük önem arz eden DDPG [12] algoritması ilk formüle edildiği zaman TORCS üzerinde denenmiştir. İlgili çalışmada, bu çalışmada olduğu gibi hem sağlanan özelliklerle hem de görüntülerle eğitim yapılmıştır. Özelliklerle sağlanan başarı, görüntülerle alınan başarıdan fazladır. Fakat burada yapılan karşılaştırmalar diğer çalışma gibi, yalnızca tek pist üzerinde ve algoritma temelinde yapılmıştır. DDPG'ye benzer şekilde Pekiştirmeli Öğrenme denince akla gelen ilk algoritmalardan biri olan A3C [13] algoritması da DDPG gibi başarısını TORCS üzerinde aynı şekilde denemiştir.

Bu çalışmaların yanında, TORCS daha çok, yapılan bir iyileştirmeyi, iyileştirmenin yapıldığı temel algoritma ile karşılaştırmak için kullanılmıştır. Bu çalışmalara örnek olarak [14], [15], gösterilebilir. Benzer şekilde, imitasyon öğrenme yöntemini kullanarak insan davranışını da modellemeye çalışan çalışmalar da bulunmaktadır [16], [8], [17].

1.2 Tezin Amacı

Bu çalışmada, pekiştirmeli öğrenme problemi olarak modellenebilecek olan, simülasyon ortamında bir araba yarışında başarı elde edilmesi amaçlanmaktadır. İlgili ortam, probleme uygun The Open Racing Car Simulator olarak seçilmiştir. Bu problemi çözmek için Rainbow DQN ve Soft-Actor-Critic algoritmalarına başvurulacak ve çeşitli keşif ve genelleştirme yöntemleri ile karşılaşılan problemler çözülecek, performans iyileştirme imkanları aranacaktır.

1.3 Hipotez

Derin Pekiştirmeli Öğrenme yöntemleri arasından Rainbow DQN ve Soft-Actor-Critic algoritmaları ile çeşitli keşif ve genelleştirme yöntemleri ile kullanarak simülasyon ortamında yapılacak otonom araba yarışlarında, literatürde yapılan diğer çalışmalara göre gürbüz ve hızlı bir ajan ile başarı elde edilebilir.

1.4 Temel Kavramlar

1.4.1 Pekiştirmeli Öğrenme

Pekiştirmeli Öğrenme, otonom bir iş yapılması istenen sistem içerisinde, en uygun eylemin net bilinememesi durumlarında kullanılagelmiş bir yöntemler bütünüdür. Pekiştirmeli Öğrenme tabanlı sistemleri genellikle oyunlar üzerinden örneklendirmek anlaşılmayı kolaylaştırmaktadır. Herhangi bir satranç oyununda en iyi hamleyi belirlemek, tüm olası hamleleri ve sonuçlarının ulaşabileceği kazanma ve kaybetme ihtimallerini hesaplamayı gerektirecektir. Fakat böyle bir hesaplama ve bellek kapasitesine sahip bir bilgisayar bulunmamaktadır.

Pekiştirmeli Öğrenme yöntemleri içerisinde, ödül ve ceza işaretleri kullanılarak, ajanın (eylemleri yerine getiren birim) en çok ödül almayı beklediği durumlara ve eylemlere yönelmesini sağlamak önceliktir. Bu tarz problemlerin genelde ardışıl karar sistemleri olarak modellendirilmesi sebebiyle, ilgili problemler Pekiştirmeli Öğrenme ortamlarında Markov Karar Süreçleri olarak tasarlanırlar.

1.4.1.1 Markov Karar Süreçleri

Markov Karar Süreçleri, ardışıl stokastik karar süreçleridir. Karar alma süreçlerinin matematiksel olarak ifade ve modellenmesinde kullanılır. Dört başat unsuru bulunmaktadır, bunlar;

- S: Durum uzayı, ajanın alıcılarıyla gözlemleyebileceği, ortamın gözlemlenebilir her türlü durumunu barındırmaktadır.
- A: Eylem uzayı, ajanın duyargalarıyla ortamdan aldığı bilgilere göre tepki verebileceği eylemlerin bütününü simgeler.
- $P_a(s, s')$: Durum geçiş dağılımı, herhangi bir durumda olası eylemler sonucunda ajanın hangi duruma geçiş yapacağını gösterir.
- $R(s)$: Ödül fonksiyonu, ajanın her durumda ne kadar ödül sinyali alacağını işaret eder.

Bunlara ek olarak, hangi durumda sürecin sonlanacağını gösteren, T_a dağılımı ve gelecekteki ödüllerin bulunulan duruma olan etkisini belirten γ indirim faktörü, Markov Karar Süreçlerinde kullanılan öğelerdir.

Herhangi bir t anından itibaren elde edilen ödüller kümülatif olarak Denklem 1.1 ile hesaplanmaktadır. Bu hesaplama, getiri olarak isimlendirilmektedir [18].

$$G_t = R_{t+1} + \gamma R_{t+2} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{1.1}$$

İndirim faktörü, 0 ila 1 arasında bir değer alır. 1'e ne kadar yakın olursa, gelecekte edinilen ödüllerin getiriye katkısı o kadar önemli olmaya başlamaktadır. Bu değerin diğer bir kullanım amacı da getirilerin hesaplamalarının geometrik serilerin limiti kullanılarak yakınsaklığı elde etmektir. Denklem 1.2'de bu durum gösterilmiştir.

$$\sum_{k=0}^{\infty} \gamma^k R_{max} = \frac{R_{max}}{1-\gamma} \qquad \gamma \epsilon [0, 1) \tag{1.2}$$

Denklem 1.1 kullanılarak, her bir duruma ait, o durumun getirisi bulunabilecektir. Bu şekilde Bellman denklemlerinin ilk aşaması elde edilecektir.

$$
\begin{aligned}
V(s) &= \mathrm{E}[G_t|S_t = s] \\
V(s) &= \mathrm{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3}...] \\
V(s) &= \mathrm{E}[R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3}...)] \\
V(s) &= \mathrm{E}[R_{t+1} + \gamma G_{t+1}] \\
V(s) &= \mathrm{E}[R_{t+1} + \gamma G_{t+1}|S_t = s] \\
V(s) &= \mathrm{E}[R_{t+1} + \gamma V(S_{t+1})|S_t = s]
\end{aligned}
\tag{1.3}
$$

Bu formülasyona 1.3, durum değer fonksiyonu denmektedir. $V(s)$ terimi, s'nin durumları işaret ettiği halde, o durumun değerini ifade eder.

Bir durum karşısında alınacak eylemin değerini ifade eden $Q(s,a)$ terimi, benzer şekilde denklem 1.4 ile hesaplanabilmektedir. Bu formülasyona da eylem değer fonksiyonu denmektedir.

$$
\begin{aligned}
Q(s,a) &= \mathrm{E}[G_t|S_t = s, A_t = a] \\
Q(s,a) &= \mathrm{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3}...] \\
Q(s,a) &= \mathrm{E}[R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3}...)] \\
Q(s,a) &= \mathrm{E}[R_{t+1} + \gamma G_{t+1}] \\
Q(s,a) &= \mathrm{E}[R_{t+1} + \gamma G_{t+1}|S_t = s] \\
Q(s,a) &= \mathrm{E}[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1})|S_t = s, A_t = a]
\end{aligned}
\tag{1.4}
$$

Bu denklemler sayesinde, pekiştirmeli öğrenme için iterasyon yeteneği sağlanmıştır. Her bir t anında başlayan ve ortam içinde ajanın izlediği izleklerin döndüğü ödüller kullanılarak, durum değer ve eylem değer fonksiyonları bulunabilir.

Her bir durumun ve bu durumlardaki eylemlerden beklenen değerlere ulaşıldıkça doğru politika bulunması kolaylaşacaktır. Bu değer fonksiyonları, tablo tabanlı olabileceği gibi, giriş olarak durum yahut durum ve eylem alan ve çıktı olarak değeri veren doğrusal veya doğrusal olmayan fonksiyon yaklaşıklıkları ile bulunabilir. Derin öğrenme pratiklerinin kullanılmasıyla beraber bu fonksiyon yaklaşıklıkları sinir ağları ile çözülmeye başlanmıştır [19].

Değer fonksiyonları kullanılarak en yüksek değer veren eylem seçilebilir ve bu sayede en uygun politika bulunabilir. Benzer şekilde giriş olarak durumları alan ve çıktı olarak

eylemlerin olasılıksal dağılımını veren politika tabanlı yöntemler bulunmaktadır. Bu yöntemler eylemler ayrık olmadığı zaman kullanışlı olmaktadır. Yaklaşıklık kullanılan politika tabanlı ilk çalışmalar Politika Gradyeni ile başlamış olup, baz olarak değer fonksiyonlarını da kullanan daha etkin yöntemler geliştirilmiştir [20]. Bunlar Aktör-Kritik yöntemler olarak adlandırılmakta olup, güncel çalışmalar genellikle bu yöntemleri kullanmaktadır [21].

1.4.2 Keşif Yöntemleri

Pekiştirmeli Öğrenme ajanları, ortamdan pozitif bir karşılık aldığı andan itibaren o duruma tekrar ulaşmaya çalışacağından ötürü, daha iyi ödül bulunması ihtimaline rağmen, bulduğu ödülü sömüreceği için, daha iyi bir ödül bulamayabilir. Bu sebeple her pekiştirmeli öğrenme yönteminin gürbüz bir keşif stratejisine ihtiyacı olmaktadır.

1.4.2.1 Epsilon Açgözlü Keşif

Q-Learning gibi değer fonksiyonu yöntemlerinde, en uygun politika Q değerlerine göre açgözlü davranarak elde edilir. Fakat bu durum bahsedildiği üzere sömürüye sebep olacağı için, ajanın belli durumlarda rastgele davranması sağlanabilir. Bu tarz keşif sistemleri epsilon-açgözlü keşif algoritması denilir (ϵ) [18]. Epsilon değeri, 0 ila 1 arasında bir değer alır ve 1 değeri tamamen rastgele, 0 değeri ise Q tamamen açgözlü davranmasına yol açar. Aradaki değerler eğitim süresi boyunca Şekil 1.1'te görülebileceği gibi türlü şekillerde hazırlanabilir. Şekilde görünen stratejiler birer örnek teşkil etmekte olup, probleme özel stratejiler de tasarlanabilir.

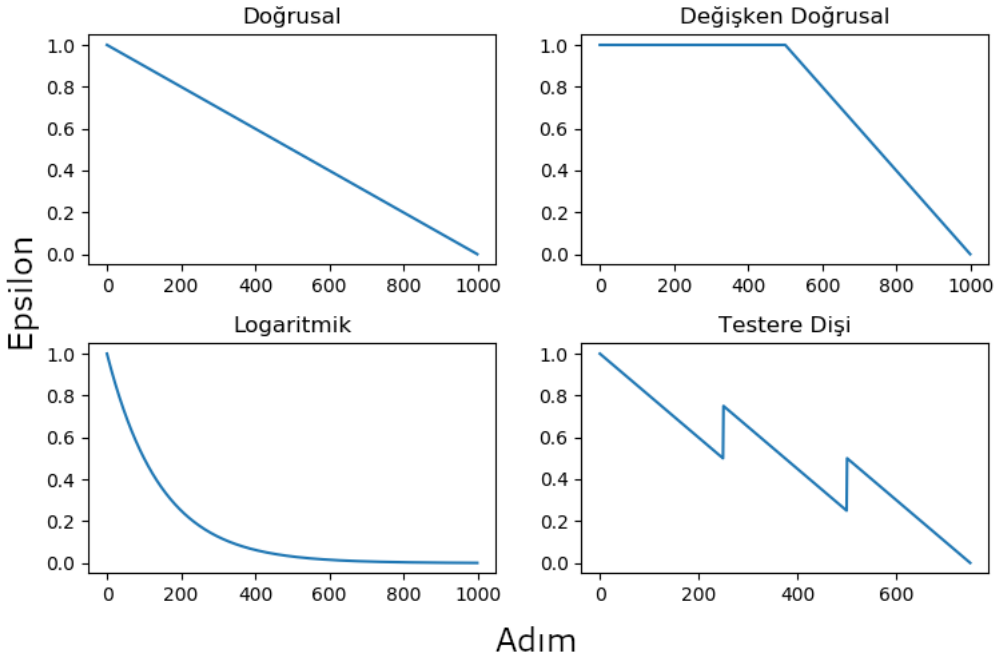
1.4.2.2 Gürültülü Ağlar

Gürültülü Ağlar, değer yaklaşıklığında yahut, politika tabanlı yöntemlerde kullanılan sinir ağlarının ağırlıklarına eğitim boyunca stokastik bir gürültü ekleyerek keşif yapılmasını sağlar [22]. Gürültüyü kontrol eden parametre, ağın ağırlıkları gibi optimize edilir.

1.4.2.3 Entropi Tabanlı Keşif

Politika tabanlı yahut Aktor-Kritik yöntemlerde, çıktı olarak eylemler üzerine olasılıksal dağılım oluşturulmaktadır. Bu olasılıksal dağılımın entropisini bulmak mümkün olmaktadır. Entropi sistemin ortama dair ne kadar bilgisi olduğunu ölçmek için önemli bir değer olması sebebiyle, bunu keşif için kullanmak mümkündür.

Sistemin entropisini maksimize etmeye çalışarak, sisteme dair görevleri ve sistem



Şekil 1.1 ϵ değerinin zaman içinde değişim örnekleri

dinamiklerini öğrenmek amaçlanmaktadır. Aynı zamanda, sistemin eğitimi süresince entropi değeri yüksek olan durumlarda eğitim yavaşlatılarak, ilgili bölgeleri daha çok öğrenmeye yönlendirme sağlanabilmektedir [23].

1.4.3 Genelleştirme Yöntemleri

Yeni, daha önce deneyimlenmemiş veya görülmemiş ortamlara kolay adapte olmak, pekiştirmeli öğrenme ajanlarının üstesinden gelmesi gereken önemli bir özelliktir. Eğer ajanlar genelleştirmeyi iyi yapmazlarsa, eğitim süresince görülen ortamlardaki en uygun politikayı ezberler ve yeni ortamlarda istenilen sonucu veremezler. Bunun önüne geçmek için kullanılan stratejilere genelleştirme yöntemleri denilmektedir.

1.4.3.1 Büyük Mimariler

Ağ topolojileri, genelleştirme için önemli bir imkan sunmaktadır. Deneylerle görülmüştür ki, daha fazla işlem yükü getirmesine karşın büyük ağ topolojileri, küçük ağlara göre daha iyi genelleştirme sağlamaktadır. Aynı ortamlarda ve aynı eğitim adımlarına tabi tutulan iki algoritma (Impala-CNN [24], Nature-CNN [19]) arasında karşılaştırma yapıldığı zaman, daha büyük mimariye sahip olan Impala-CNN yönteminin çok daha başarılı genelleştirme sağladığı görülmüştür [25].

1.4.3.2 Veri Artırımı

Veri artırımı, Derin Öğrenme yöntemlerinin de genelleştirme konusunda sıkıntı yaşadığı durumlarda sıklıkla başvurulan bir yöntemdir [26]. Ezberlemenin önüne geçmek için derin öğrenmede kullanıldığı gibi, veri artırımı yöntemleri pekiştirmeli öğrenme sistemlerinde de kullanılmaktadır. Bu yöntem, ajanı daha fazla ortamla eğitmek olabileceği gibi, ortama müdahil olunarak domen rastgeleliği [27] gibi yöntemlerle de sağlanabilir.

1.4.4 Algoritmalar

Tez içerisinde kullanılan eğitim algoritmaları Rainbow DQN ve Soft-Actor-Critic algoritmalarıdır.

1.4.4.1 Rainbow DQN

Nature dergisinde yayınlanarak büyük ses getiren Atari-DQN [19] çalışmasını takiben zaman içerisinde DQN üzerine birçok iyileştirme yapılmıştır. Rainbow DQN [28] çalışması, tüm bu iyileştirmeleri bir araya getirerek güçlü bir değer tabanlı eğitim algoritması oluşturmuştur. Bu geliştirmeler şu şekildedir:

- Double Q Learning [29] - Bu iyileştirme, öğrenilen Q değerlerinin politika ile kullanıldığında doğru eylemi sunsa bile, beklediği ödülün çok fazla çıkması durumunun önüne geçmektedir. Algoritma bir ağı eğitirken eylemi o ağ değerlerinden seçmekte, fakat bir sonraki durumdan beklenen değeri diğer (double) ağdan almaktadır. Ağların eğitimi her bir adımda sırayla yapılır. Bu sayede eylemler arası değer dengesi korunmuştur.
- Priority Experience Replay [30] - Standart DQN ile, eğitim yapılacak deneyimler hafızadan rastgele olarak seçilmektedir. Bu yöntem içerisinde bu deneyimler belli bir parametre ile seçilmektedir. Bu parametre genellikle en çok hata hesaplanan (öğrenim sağlayan) deneyimler olmaktadır.
- Dueling Networks [31] - Bu yöntemde, değer fonksiyonu ağının çıkışında durum değer ve eylem değer avantajları ayrı hesaplanır ve sonrasında aritmetik işleme tabi tutulur. Bu sayede eylemin çok önemli olmadığı durumlarda da durum değeri hesabı ve eğitimi yapılmış olur.
- Noisy Networks [22] - Q Learning yöntemlerinde keşif genellikle eylem tabanında epsilon-açgözlü yöntemler kullanılarak yapılmaktadır. Gürültülü Ağlar, parametre uzayında keşif yapılmasını sağlayarak dolaylı olarak durum ve eylem keşfini de sağlamaktadır.

- Multi-Step Learning [32] - Çok adımlı Q hesaplamaları, bir adımlı hesaplamalara göre daha doğru ve hızlı öğrenme sağladığı gözönüne alınarak bu iyileştirme eklenmiştir.
- Distributional RL [33] - Q hesaplamalarını sürekli bir değere atamaya çalışmak ve buna göre ağı eğitmek yerine, belli bir sayıda ayrık çıkışa atayarak bir dağılım bulunması amaçlanmıştır. Bu şekilde ikili çıkışlar ile hızlı bir eğitim ve daha doğru Q değerleri elde edilmiştir.

Bu iyileştirmelerin bütünü Rainbow DQN mekanizmasını oluşturmaktadır.

1.4.4.2 Soft Actor Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor (SAC)

SAC, isimlendirilmesinden anlaşılabileceği üzere Aktör-Kritik bir pekiştirmeli öğrenme algoritmasıdır. Yalnızca getirileri maksimize etmeye çalışan standart pekiştirmeli öğrenme hedefi yerine entropi değerini de maksimize etmeye çalışan, dönüştürülmüş bir hedef kullanmaktadır. Bu sayede keşif ve genelleştirme noktalarında başarı sağlamaktadır.

Double Q learning [29]'de uygulanan ikili Q ağını SAC algoritması da değer hesaplaması için kullanmaktadır. Bu eylem değer fonksiyonu, eğitim için getirilerin yanında durum değer ağı da eğitmekte ve eğitimde bu durum değer fonksiyonunu kullanmaktadır. Fakat bu fonksiyonların birbirlerine bağımlı olması kararsızlığa yol açabileceği için, durum değer fonksiyonunun eğitiminde Polyak ortalaması kullanılmaktadır [34].

Politika eğitilirken, Q değerlerinin hedef yoğunluğu kulanılmaktadır. Politika güncellemesinde türev imkanı sağladığı için re-parameterizasyon tekniği kullanılmaktadır. Bununla beraber politikanın her bir eğitimde büyük değişikliğe uğramaması için Kullback–Leibler uzaklığı kontrol edilmektedir.

1.4.5 TORCS

TORCS (The Open Racing Car Simulator), Açık Araba Yarışı Simülasyonu, insanların oynayabileceği bir araba yarışı olduğu gibi, yapay zeka tabanlı ajanların eğitimi için kullanımına sıklıkla başvurulan bir ortamdır. Örnek bir yarıştan kare Şekil 1.2'de görülebilir. Ajanların etkileşimi için birçok gözlem ve eylem imkanı sunmaktadır. Aşağıda ortamın sunduğu gözlemler bulunmaktadır.

- Açı (Angle) - 1 boyut - Yolun tanjantı ve araba arasındaki açı



Şekil 1.2 Örnek bir yarışma

- Yol (Track) - 19 boyut - Arabanın önünü 180 derecelik bir kapsama alanıyla tarayan Lidar tarayıcısı
- Yol Pozisyonu (TrackPos) - 1 boyut - Arabanın yolun ortasına uzaklığı
- Hız (Speed) - 3 boyut - Arabanın 3 boyuttaki kartezyen hızları
- Tekerlek Hızları (Wheel Speeds) - 4 boyut - Her bir tekerin açısal hızı
- Motor Dönüş Hızı (RPM) - 4 boyut - Motorun dönüş hızı

Bu gözlemlere ek olarak istenirse TORCS ortamından arabanın görüşü de alınabilmektedir. Tezde kullanılan 3 eylem tipi bulunmaktadır, bunlar gaz, fren ve direksiyon olmaktadır.

Sistem, ajanların yarışabileceği elle kodlanmış bot yarışmacılar bulundurmaktadır. Bu sayede gerçek yarışmaya benzer bir ortam sağlanabilmektedir.

2
METODOLOJİ

Bu bölümde, öncelikle problem tanımı genişletilecek, keşif ve genelleştirme yöntemleri detaylandırılacaktır. Çalışmayı başarıya ulaştıran algoritmalar anlatılacak ve mevzubahis algoritmalar ve iyileştirmelerin TORCS ortamına hangi yöntemlerle uygulandığı açıklanacaktır. Algoritmaların mimarisi, ödül düzenleme ve ortamın sonlandırılmasından bahsedilecektir. Sistemin kaynak koduna Github [35] linki vasıtasıyla erişilebilir.

2.1 Problem Tanımı

TORCS ortamı, problemin Markov Karar Süreci olarak modellenmesine olanak sağlamaktadır. Ortamın Python uyumluluk paketi kullanılarak, ajanın karar verdiği her eylem sonrası ortamın sağladığı veriler kullanılarak ödül hesabı yapılabilmekte, ajana ödül ve bir sonraki gözlem sağlanabilmektedir. Bu sayede pekiştirmeli öğrenme yöntemleri kullanılarak araba yarışı problemine TORCS ortamı ile çözüm aranabilmektedir.

Araba yarışı problemi, sürüş dinamiklerine ek olarak hızlı gitmeyi gerektirmektedir. Yalnızca bir hedefe varmak ajan için çözüm olmayacaktır. Bu sebeple, ajan yoldan çıkmadan hızlı gidebilmeli, düzlüklerde hızlanabilmeli, viraja yaklaştığı zaman yavaşlamayı öğrenebilmelidir. Ajanın ortam ve kendisine dair anlık gözlemi ve eyleminin yanında, bir veya birkaç önceki durumunda ne olduğuna dair bilgisi de olabilmelidir. Zira, yalnızca yarış için değil, herhangi bir sürüş dinamiği için de zamansal bilgi önem arz etmektedir.

Ajan yalnızca eğitim aldığı ortamlarda değil, yeni gördüğü ortamlarda da başarı sağlayabilmeli, genelleştirme yapabilmelidir. Bununla beraber eğitimde görmüş olduğu ortamı ve modelini iyi keşfetmeli, bu sayede yerel minimumlardan bir nebze de olsa kurtulabilmelidir.

Çalışmada bu problemlere dair cevaplar aranacak, Rainbow DQN ve SAC gibi

güncel algoritmalar, keşif ve genelleştirme yöntemleriyle iyileştirilecek, hızlı sürüş ve kararlılık hedefiyle yarışta başarılı olmak amaçlanacaktır.

2.2 Mimari

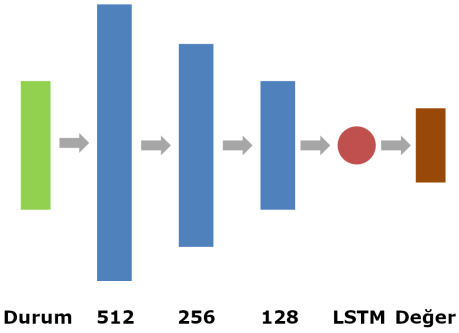
2.2.1 SAC

SAC'ın ağ topolojisi tümüyle bağlı üç katmandan oluşmaktadır. Bu üç katman, Şekil 2.1'de görülebileceği gibi sırasıyla 512, 256 ve 128 ağırlıklara sahiptir. Aktivasyon fonksiyonu olarak, ara katmanlarda ReLU ve çıkış katmanında TanH kullanılmaktadır. Çalışmada, ağın son katmanından önce bir LSTM katmanı eklenmesinin sistemi daha kararlı bir noktaya çektiği gözlemlenmiştir. Algoritmanın hiper-parametreleri grid arama (grid search) yöntemiyle aşağıda görülebileceği şekilde belirlenmiştir.

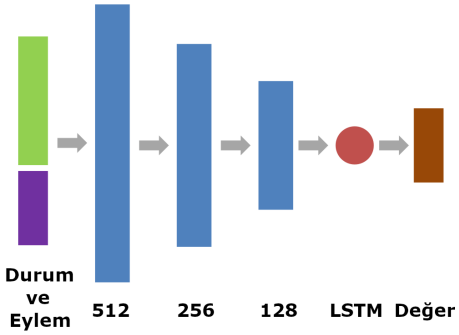
- Gamma: 0.99
- Tau: 10^{-3}
- Yığın Boyutu (Batch Size): 32
- Adım Boyutu (Step Size): 16
- Bölüm Belleği (Episode Buffer): 10^{3}
- Aktör Öğrenme Hızı (Actor Learning Rate): 3.10^{-4}
- Durum Değer Öğrenme Hızı (Value Learning Rate): 3.10^{-4}
- Eylem Değer Öğrenme Hızı (Q Learning Rate): 3.10^{-4}
- Entropi Öğrenme Hızı (Entropy Learning Rate): 3.10^{-4}
- Politika Güncelleme Sıklığı (Policy Update Interval): 2
- Eğitim Öncesi Rastgele Adım Sayısı (Initial Random Action): 10^{4}

2.2.1.1 Sıralı Deneyim Yinelemesi

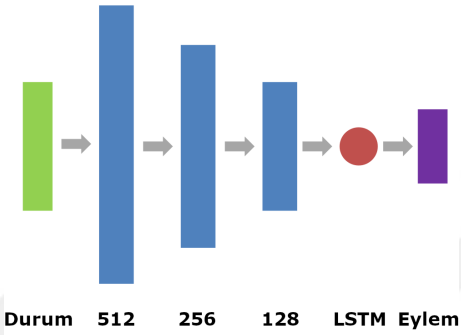
SAC algoritması içerisinde deneyim yinelemesi sistemi bulunması sebebiyle LSTM için değişiklik gerekmektedir. Standart deneyim yinelemesinde deneyimler rastgele zamanlardan seçildiği için LSTM'in herhangi bir şekilde eğitilmesine olanak tanımamaktadır. Bu sebeple sıralı deneyim yinelemesi eklemesi yapılmıştır. Her bölüm bütünüyle ve sıralı bir şekilde kaydedilmekte, yığın (batch) boyutuna göre



(a) SAC V Network



(b) SAC Q Network



(c) SAC Policy Network

Şekil 2.1 SAC mimarisi

rastgele bir şekilde eğitim zamanında bu bölümlerden seçilmektedir. Sonrasında her bölümden rastgele bir zaman seçilmekte ve bu zamanı takip eden 16 örnek ile eğitim yapılmaktadır.

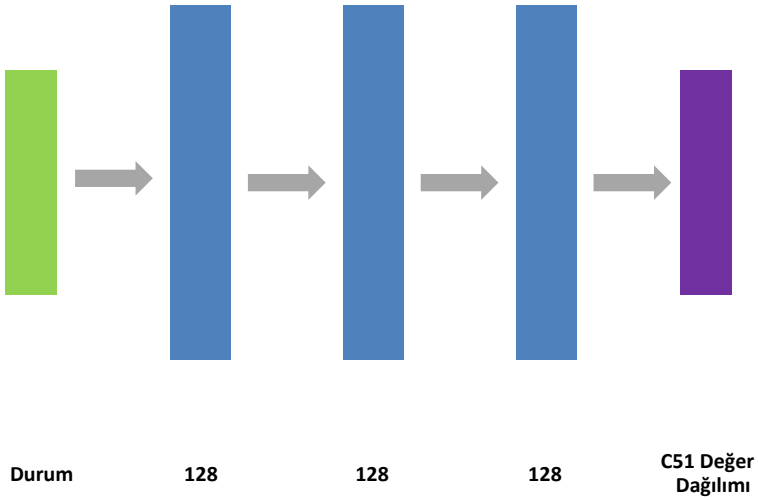
2.2.1.2 N-Stack Çalışması

Atari-DQN [13] çalışmasında, ajanın zamansal bilgiye erişimi için observation stack mekanizması kullanılmıştır. Benzeri tez içerisinde LSTM'den önce, N-Stack şeklinde uygulanmıştır. Giriş katmanı iki, üç, dört, beş katlarına çıkartılmış ve geçmiş durumlarla beraber zamansal bilginin kullanılması sağlanmıştır. Fakat LSTM'in daha performanslı olduğunun gözlemlenmesiyle beraber, N-Stack mekanizmasından vazgeçilmiştir.

2.2.2 Rainbow DQN

SAC algoritmasının topolojisine benzer şekilde DQN algoritmasının topolojisi de Şekil 2.2'de görülebileceği üzere tümüyle bağlı 3 katmandan oluşmaktadır. Bu 3 katmanın her biri 128 ağırlık barındırmaktadır. Ara katmandaki aktivasyon fonksiyonlarında yine ReLU kullanılmaktadır, fakat çıkış katmanında Q değerleri üzerine 51 atom dağılımı kullanılmıştır. Basit bir grid araması (grid search) sonrası

aşağıdaki hiper-parametrelere karar verilmiştir.



Şekil 2.2 DQN mimarisi

- Gamma: 0.99
- Tau: 10^{-3}
- N-Adım (N-Step): 3
- N-Adım Ağırlık Parametresi (N-Step Weight Parameter): 1
- N-Adım Q Düzenleme Parametresi (N-Step Q Regularization Parameter): 10^{-7}
- Bellek Boyutu (Buffer Size): 10^{5}
- Yığın Boyutu (Batch Size): 32
- Öğrenme Hızı (Learning Rate): 10^{-4}
- Adam Epsilon: 10^{-8}
- Adam Ağırlık Azalımı (Adam Weight Decay): 10^{-7}
- PER (Öncelikli Deneyim Yinelemesi) Alpha: 0.6
- PER Beta: 0.4
- PER Epsilon: 10^{-6}
- Gradyen Kesimi (Gradient Clip): 10
- Eğitim Öncesi Rastgele Adım Sayısı (Prefill Buffer Size): 10^{4}

- C51 - V Minimum: -300
- C51 - V Maximum: 300
- C51 - Atom Boyutu (Atom Size): 1530
- Gürültülü Ağlar Varyans Başlangıcı (Noisy-Net Initial Variance): 0.5

2.3 TORCS

TORCS ortamı olarak Gtihub [36] linkindeki uygulama baz alınmıştır. Uygulama bir sunucu gibi davranarak, istemcilerden (ajanlardan) eylem beklemektedir. Her bir eylem sonrasında yeni gözlemi istemcilere gönderir ve bu döngü yarış bitimine kadar devam eder. Bu ortam canlı yarışa da olanak sağlaması sebebiyle cevap süresi milisaniyeler mertebesindedir. Eğitim sırasında ise GPU ve CPU yoğun olarak kullanıldığı için bazen istemciden cevaplar geç gelmekte, bu da mesajların zaman aşımına uğramasına sebep olmaktadır. Eğitimde yarışların hem izlenmemesi hem de hızlı olması gerekliliği olmadığı için, bu çalışmada ortamın zaman aşımı süresini milisaniye mertebesinden 10 saniyeye çıkarmak bu zaman aşımı problemini çözmüştür. Aksi durumda ajanlar eylemin sonuca etkisini doğrudan göremeyecek, eğitim düzgün sağlanamayacaktır.

2.3.1 Ödül Düzenlemesi

Pekiştirmeli öğrenme sistemlerinde, ajanın istenilen hedefe ulaşabilmesi için ödül düzenlemesi çok büyük önem arz etmektedir. Yarışma gibi ortamlarda, sadece hız değil, yolda kalış, yolu ve virajları etkin kullanım, arabanın momentumuna göre davranma zorunluluğu ödül düzenlemenin önemini ortaya çıkarmaktadır. Bu sebeple, bu çalışmada ödül düzenlemesine çok önem verilmiştir.

TORCS üzerinde eğitim sağlamış birçok geliştirici gibi, yolun ortasında ve düz gitmeye çalışan ajanlar hızlı sağ sol (slalom) yapmaktadır. Bunu engellemek için birçok alternatif ödül düzenlemesi denenmiştir. Bununla beraber verili ortamda eğer ajan yoldan çıkarsa veya geri doğru dönerse ortam sıfırlanmaktadır. Tezde ise bu duruma izin verilmiş olup, ajan geri dönse yahut yoldan çıksa bile yola geri dönmeyi öğrenmesi sağlanmıştır.

2.3.1.1 Ödül Fonksiyonları

Ödül fonksiyonlarında kullanılan parametreler aşağıdaki şekilde tanımlanmıştır.

- V_x: Uzunlamasına hız
- V_y: Yanal hız
- θ: Araba ve yol arasındaki açı
- $trackpos$: Arabanın yolun ortasına uzaklığı

Bu parametreler kullanılarak aşağıdaki ödül fonksiyonları çalışılmıştır.

- No Trackpos: Yolun ortasına uzaklığı önemsenmemiştir.

$$R = V_x \cos\theta - |V_x \sin\theta| \tag{2.1}$$

- Trackpos: Yolun ortasına uzaklığı önemsenmiştir.

$$R = V_x \cos\theta - |V_x \sin\theta| - |V_x trackpos| \tag{2.2}$$

- EndToEnd [37]: Aracın açısı önemsenmemiştir.

$$R = V_x(\cos\theta - |trackpos|) \tag{2.3}$$

- DeepRLTorcs [38]: Yolun ortasına olan uzaklık cezası aracın açısıyla azaltılmıştır. Bu sayede araç düz gittiği müddetçe yolun ortasında durması şart değildir. Aracın açısı ayrıca ceza olarak kullanılmıştır. Ayrıca, yanal hız cezası aracın açısıyla azaltılmıştır. Bu sayede eğer araç virajda düzgün kaydırıyorsa (drift) ceza verilmemiştir.

$$R = V_x \cos\theta - |V_x \sin\theta| - |2V_x \sin\theta\, trackpos| - V_y \cos\theta \tag{2.4}$$

- Sigmoid: DeepRLTorcs fonksiyonuyla aynı fonksiyon fakat, slalomu engellemek için dairesel ödül fonksiyonları düzleştirilmiştir.

$$R = V_x sigmoid(3\cos\theta) - V_x \sin\theta - V_y sigmoid(3\cos\theta) \tag{2.5}$$

Çalışmada en iyi performans "DeepRLTorcs" ödül fonksiyonuyla alınmıştır. Sigmoid fonksiyonu dairesel ödülleri yumuşak kırptığı için umut vermektedir fakat herhangi bir iyileştirme sağlamamıştır, üzerinde daha fazla çalışılması gerekmektedir.

2.3.1.2 Ortam Sonlanması

Bölüm eğer ilk 100 adımda herhangi bir ilerleme sağlamadıysa yüksek bir cezayla sıfırlanmaktadır. Burada ilerlemeden kasıt 5km/s'dir. Daha önce bahsedildiği üzere, aracın geri dönmesi yahut yoldan çıkması sonrasında ajana kendini düzeltebilmesi için 100 adım imkan verilmektedir. Zira, aracın hızlı gitmesi kimi zaman savrulmasına yol açabilmektedir.

2.3.2 Keşif

Ajanın durum, eylem ve parametre keşfi, SAC algoritmasında entropi maksimizasyonu vasıtasıyla sağlanmıştır. DQN'de ise, Gürültülü ağlar kullanılarak dolaylı olarak durum ve eylem keşfi sağlanmıştır. Bu yöntemlerin faydası görülmekle birlikte, ajanın fren yapmayı öğrenmesi çok zor olmaktadır. Zira fren yapmak hem ödülü azaltmakta, hem de hızlı gitmek yoldan çıkmaya sebep olmaktadır. Bu sebeple ajan dönüşlerde fren yapmamak için yavaş gitmekte, uzun düzlüklerde hızlanmamaktadır. Bunun önüne geçebilmek için Fren-Dene yöntemi kullanılmıştır.

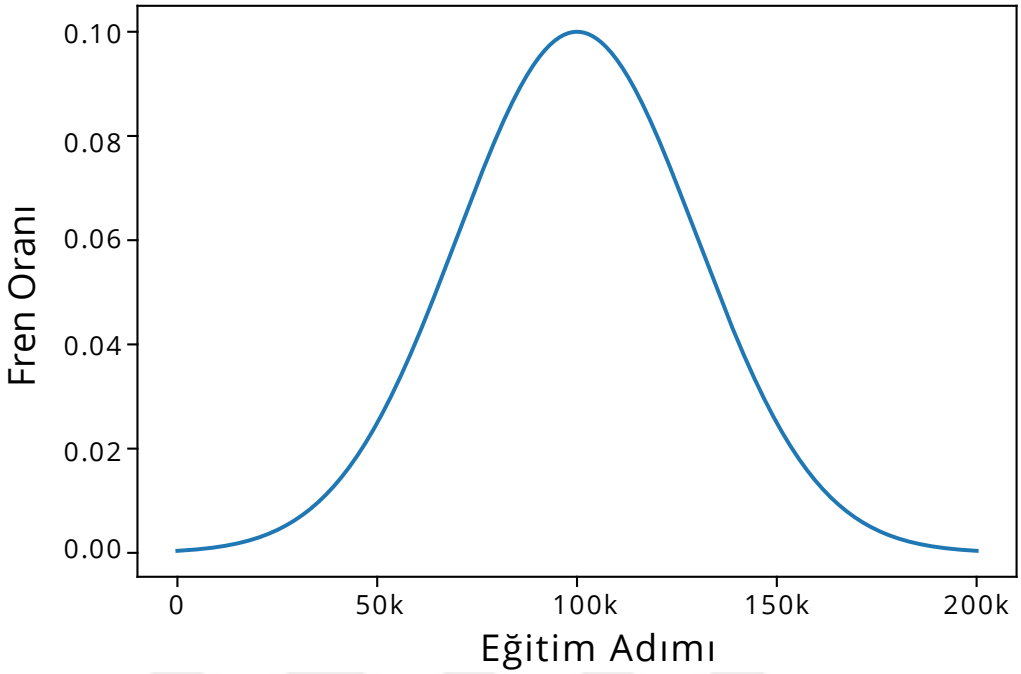
2.3.2.1 Fren-Dene

Fren-Dene mekanizması Stokastik Fren Deneme [38] sistemine benzemektedir. Belli bir eğitim adımı sonrası, ajan fren kullanmaya zorlanmaktadır. Fren kullanımı zorlandıktan yine belli bir müddet sonra bu zorlama ortadan kaldırılmaktadır. Fren denemesi boyunca fren kullanım zorlama sıklığı ve eğitim süresi dağılımı Şekil 2.3'de görülebilecek şekilde Gaussian dağılım ile gerçekleştirilmiştir. Bu sayede ajanın düzlükte hızlanması ve dönüşlerden önce fren kullanmayı öğrenmesi amaçlanmaktadır.

2.3.3 Genelleştirme

Ajanın genelleştirmesini güçlendirmek için, ağ topolojisi derinleştirilmiştir. Bu eğitim süresini uzatmakta olmasına karşın genelleştirme sağlamaktadır. Bununla beraber standart eğitim kümesine ek olarak TORCS için hazırlanmış diğer tüm "road track"ler eğitim setine eklenmiş ve bu sayede ajanın birçok alternatif yol görmesi sağlanmıştır. Bu sayede küçük verisetini ezberlemesi engellenmiştir. "Spring" pisti ise eğitim setine diğer yollardan çok farklı ve çok uzun olması sebebiyle eklenmemiştir.

Eğitim ve test için kullanılan pistler Tablo 3.3'de listelenmiştir. Ajanlar bu pistler üzerinde döngüsel bir şekilde eğitilmiştir. Her bir döngü içerisinde, her pistte 5 kez eğitim yapılmış ve diğer piste geçiş yapılmıştır.



Şekil 2.3 Fren-Dene dağılımı

2.3.4 Eylem Uzayı

Öğrenimi kolaylaştırmak için en uygun eylem uzayı hazırlanması için çalışılmıştır. Aşağıda ayrık ve sürekli eylem uzaylarının nasıl hazırlandığı gösterilmektedir. Durum uzayı gibi, eylem uzayı da -1 ila $+1$ arasında normalize edilmiştir.

2.3.4.1 Sürekli Eylem Uzayı

Bu eylem uzayı politika tabanlı Aktör-Kritik algoritma olan SAC için kullanılmıştır. Eylem uzayı birçok çalışmadan farklı olarak 3 yerine 2 olarak belirlenmiştir. İlk eylem fren ve gaz için kullanılmış, diğer eylem ise direksiyon için kullanılmıştır. İlk eylemde -1 ila 0 arasında fren yapılmış, 0 ila $+1$ arasında gaza basılmıştır. Bu sayede ajanın fren ve gaza aynı anda basmamayı öğrenmesine gerek kalmamıştır. Direksiyon eylemi için ise -1 tam sol, 0 düz ve $+1$ tam sağ olacak şekilde, arası sürekli olarak dağıtılmıştır.

2.3.4.2 Ayrık Eylem Uzayı

DQN algoritması sürekli eylem uzayı kullanamadığı için, eylem uzayı parçalara bölünmüştür. Her bir aksiyon belli bir gaz, fren ve direksiyon eylemi barındırmaktadır. Direksiyon eylemi 7'ye bölünmüş, gaz ve fren kombinasyonu 3 adet tutulmuştur. Bu sayede toplam 21 eylem ile DQN algoritmasının eğitimi yapılmıştır. Tablo 2.1 bu eylem

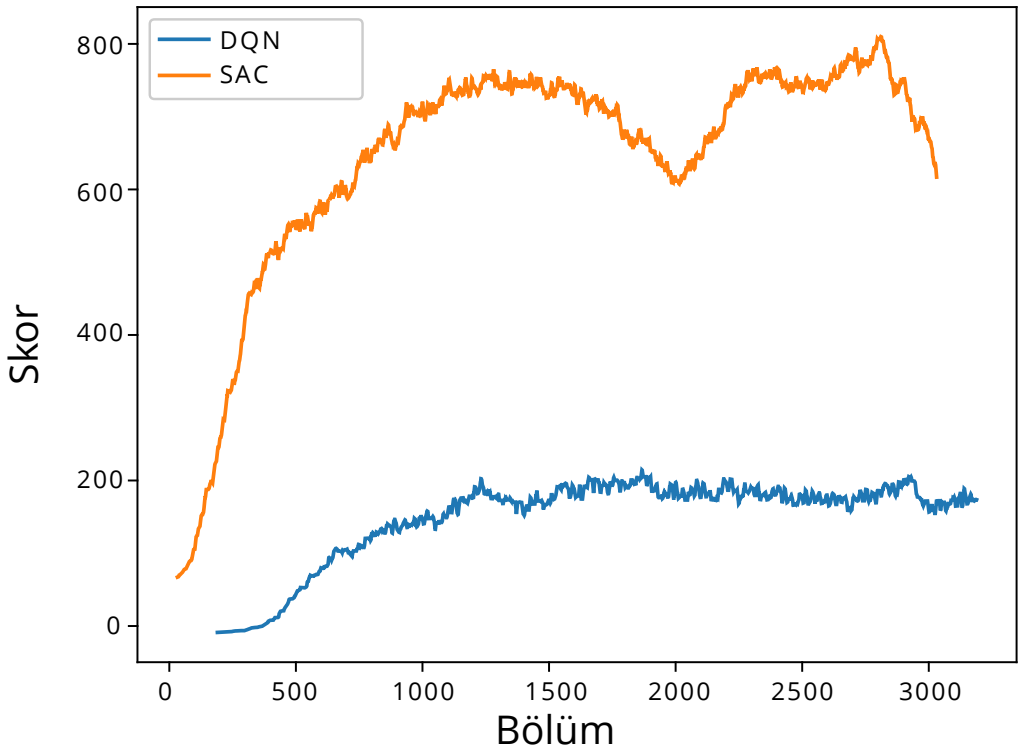
uzayının detaylarını göstermektedir. Direksiyon değerleri, −1'de sola tam dönüş, +1'de sağa tam dönüş ve 0 değerinde düz olarak ayrık şekilde hazırlanmıştır.

Tablo 2.1 Ayrık eylem uzayı

Gaz	**Fren**	**Direksiyon**
+1	-1	{-1, -0.66, -0.33, 0, 0.33, 0.66, 1}
-1	+1	{-1, -0.66, -0.33, 0, 0.33, 0.66, 1}
-1	-1	{-1, -0.66, -0.33, 0, 0.33, 0.66, 1}

3
TESTLER ve KARŞILAŞTIRMA

Rainbow DQN ve SAC algoritmalarının her ikisi de sürüş ve yarışma konusunda başarılı olmuşlardır. Hız ve yol performansı olarak SAC, Rainbow DQN'den de iyi performans elde etmiştir. Bölüm ile Skor arasındaki ilişki Şekil 3.1'den görülebilir. Değişimin azami değerleri ve standart sapması Tablo 3.1'den incelenebilir. Skor, bir yarıştaki ödüllerin toplamından oluşmaktadır. Çarpışma, yoldan çıkma ve savrulma durumlarında cezaların çok yüksek olması sebebiyle standart sapma çok yüksek çıkmaktadır. Bu sebeple görsellik ve yorumlanabilme açısından 250 bölümlük hareketli ortalama alınmıştır.



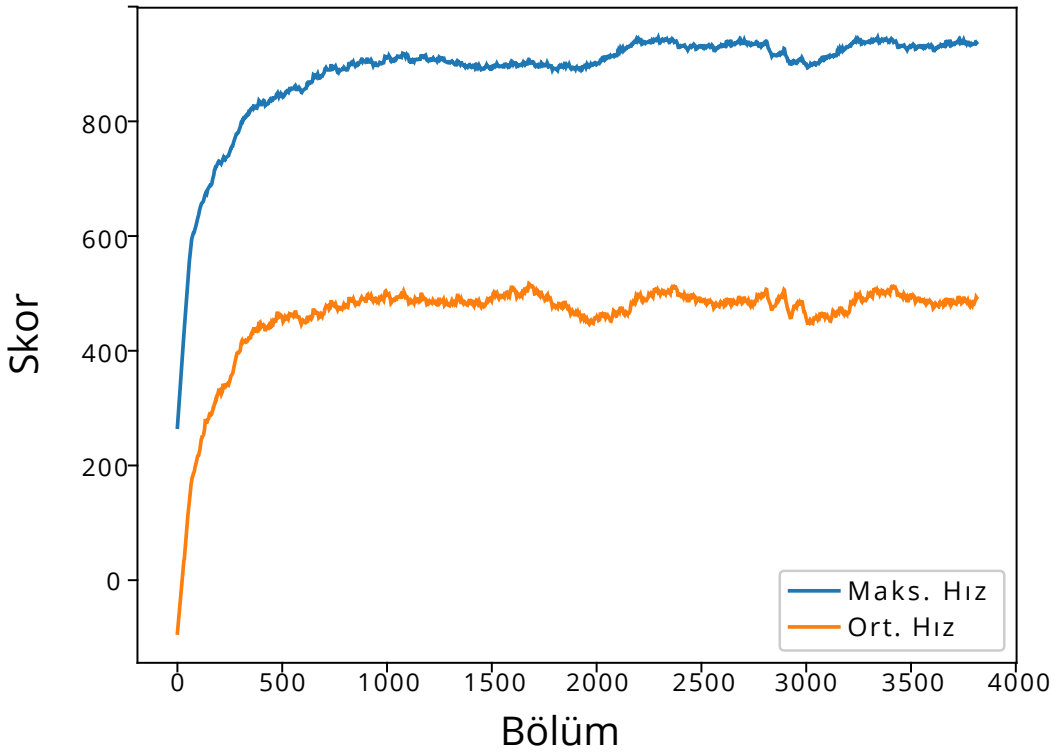
Şekil 3.1 Eğitim süresince skor değişimi

Daha başarılı olan SAC ajanı daha detaylı incelenecektir. Bu çalışma araba yarışı

Tablo 3.1 Skorların standart sapma ve azami değerleri

Algoritma	Standart Sapma	Azami Değer
DQN	306.25	1255.94
SAC-LSTM	433.22	1395.0

konusunda başarı sağlamak amacıyla yapıldığı için, arabanın hızı en önemli faktör olmaktadır. SAC ile eğitilen ajanın eğitim süresi boyunca ulaştığı ortalama ve azami hızlar Şekil 3.2'de görülebilir. Hız değerleri 250 bölümlük hareketli ortalama ile hesaplanmıştır. Değişimin standart sapması ve azami değerleri Tablo 3.2'den görülebilir.



Şekil 3.2 Eğitim süresince maksimum ve ortalama hız değişimi

Tablo 3.2 En yüksek ve ortalama hızların standart sapma ve azami değerleri

Tip	Standart Sapma	Azami Değer
En yüksek hız	24.94	216.42
Ortalama hız	28.52	164.59

Daha önce bahsedildiği üzere, ortam içerisinde 17 adet pist bulunmaktadır ve hepsinin zorluk derecesi farklıdır. Bu sebeple her biri için ortalama/azami hız ve ortalama/azami skor değerleri SAC ajanı için Tablo 3.3'de gösterilmiştir. Her sütunun en yüksek değeri kalın yazı tipi ile gösterilmiştir.

Tablo 3.3 Pistlerdeki hız ve skor değerleri

Pist	Azami Skor	Ortalama Skor	Azami Hız	Ortalama Hız
forza	**1395.0**	742.95	216.0	143.27
g-track-1	954.0	611.36	200.36	131.56
g-track-2	1182.0	**912.06**	205.65	149.32
g-track-3	1154.0	574.16	182.0	109.11
ole-road-1	1103.0	209.70	**216.42**	121.65
ruudskogen	1294.0	575.58	199.78	117.08
street-1	1208.0	597.32	198.0	117.44
wheel-1	1248.0	758.45	206.05	139.16
wheel-2	1152.0	636.49	216.19	128.87
aalborg	922.0	183.84	189.16	80.54
alpine-1	1264.0	822.09	203.18	118.94
alpine-2	1119.0	677.13	194.43	103.02
e-track-1	1080.0	325.40	207.54	119.02
e-track-2	1212.0	907.00	179.65	104.55
e-track-4	1293.0	881.47	214.73	**152.13**
e-track-6	1201.0	583.34	213.59	130.12
eroad	1264.0	883.87	200.59	131.87
e-track-3	1383.0	895.41	211.87	137.35

Cardamone ve ekibi tarafından yapılan çalışmada farklı üç pistte ayrı olarak eğitilmiş ajanlar diğer pistlerde test tabi tutulmuştur [8]. İki ayrı algoritma, dört seçim yöntemi ve eğitimin üç ayrı pistte yapılmış olması karşılaştırma yapılmasını zorlaştırmaktadır. Bu sebeple karşılaştırma yapılırken ilgili çalışmada her pistte tur süresinde en iyi sonuç alan algoritma ve seçim yöntemi karşılaştırılmıştır. Sonuçlar Tablo 3.4'de görülebilir. Cardamone sütununda parantez içerisinde detaylandırılan ilk madde eğitim yapılan pist, ikincisi eğitim algoritması ve üçüncüsü seçim stratejisini işaret etmektedir. Tur süresinin kısa olması sonucun daha iyi olduğunu göstermektedir. Tezde elde edilen süreler tüm algoritmalar ve seçim stratejileri göz önüne alındığında çok daha iyi sonuçlar üretmektedir. "-" işareti ile gösterilen kısımlarda araç yarışı tamamlayamamıştır. Tur süreleri saniye cinsindendir. Her pist için en iyi değer kalın yazı tipi ile gösterilmiştir.

Munoz ve ekibinin çalışmasında turnuvada başarı kazanan insanların ve elle kodlanmış hazır botların davranışları imitasyon öğrenmesi kullanılarak öğrenilmeye çalışılmıştır [16]. Öğrenilen ajanların geneli pistlerde tur tamamlayamamıştır. Aralarında kısmen en hızlı olan turnuvada başarılı olan Simmerson'un davranışları taklit edilerek öğrenilen ajandır. Fakat bu ajan birçok pistte yoldan çıkarak turu bitirememiştir. En çok turu tamamlayabilen elle kodlanmış hazır botların davranışlarını taklit eden ajan olmuştur. Yine karşılaştırma, tur bitirilebildiği ölçütte en hızlı olan ajanla yapılmıştır. Tur sürelerinin karşılaştırılması Tablo 3.5'den

Tablo 3.4 Cardamone ve ekibinin çalışması ile tur sürelerini karşılaştırma

Pist	Tez (s)	Cardamone (s)
e-track-1	**87**	125 (Ruudskogen, rtNEAT, ϵ greedy-improved)
e-track-2	**99**	151 (Ruudskogen, NEAT, ϵ-greedy-improved)
e-track-3	**119**	123 (Ruudskogen, NEAT, ϵ-greedy-improved)
e-track-4	148	**142** (Ruudskogen, NEAT, ϵ-greedy-improved)
e-track-6	**114**	139 (Ruudskogen, NEAT, ϵ-greedy-improved)
forza	**138**	141 (Ruudskogen, NEAT, ϵ-greedy-improved)
g-track-1	51	**50** (Ruudskogen, NEAT, ϵ-greedy-improved)
g-track-2	-	**72** (Ruudskogen, NEAT, ϵ-greedy-improved)
g-track-3	-	**95** (Ruudskogen, NEAT, ϵ-greedy-improved)
ole-road-1	**159**	172 (Ruudskogen, NEAT, ϵ-greedy-improved)
ruudskogen	-	**93** (Ruudskogen, NEAT, ϵ-greedy-improved)
wheel-1	**100**	109 (Ruudskogen, NEAT, ϵ-greedy-improved)
wheel-2	**149**	166 (Ruudskogen, NEAT, ϵ-greedy-improved)
alpine-1	**170**	174 (A-Speedway, NEAT, Softmax)
alpine-2	**125**	128 (CG-Track-1, rtNEAT, Softmax)

incelenebilir. Tur süreleri saniye cinsinden verilmiştir. Her pist için en iyi değer kalın yazı tipi ile gösterilmiştir.

Tablo 3.5 Munoz ve ekibinin çalışması ile tur sürelerini karşılaştırma

Pist	Tez (s)	Munoz (s)
e-track-1	**87**	98 (Elle kodlanmış)
e-track-2	**99**	125 (Elle kodlanmış)
e-track-3	119	**108** (Simmerson)
e-track-6	**114**	134 (Elle kodlanmış)
e-road	**83**	86 (Simmerson)
forza	**138**	158 (Elle Kodlanmış)
g-track-1	51	**47** (Simmerson)
g-track-2	-	**70** (Simmerson)
g-track-3	-	**76** (Simmerson)
ole-road-1	**159.30**	159.97 (Simmerson)
ruudskogen	-	**119** (Simmerson)
wheel-1	100	**99** (Simmerson)
wheel-2	**149**	194 (Elle kodlanmış)
alpine-1	170	**168** (Simmerson)
alpine-2	**125**	127 (Simmerson)

Verma ve ekibi tarafından yapılan çalışmada TORCS ortamı üzerinde yorumlanabilir ve teyit edilebilir ajanlar üretilmeye çalışılmıştır. İlgili çalışmada test edilen pistler ile tezdeki ajanın karşılaştırılması Tablo 3.6'de görülebilir. Parantez içinde verilen detaylar hangi algoritma ile eğitildiğini belirtmektedir. Benzer şekilde turu tamamlayan ajanlar arasındaki en iyisi karşılaştırma için seçilmiştir. Her pist için en iyi değer kalın yazı tipi ile gösterilmiştir.

Tablo 3.6 Verma ve ekibinin çalışması ile tur sürelerini karşılaştırma

Pist	Tez (s)	Verma (s)
g-track-1	**51**	54 (DRL)
g-track-2	-	**100** (NDPS)
e-road	**83**	111 (NDPS)
aalborg	**95**	109 (DRL)
alpine-2	**125**	196 (NDPS)
ruudskogen	-	**199** (NDPS)

Görülebileceği üzere tezdeki ajan diğer çalışmalara göre daha hızlı olmaktadır. Karşılaştırma yapılan tüm çalışmalarda ilgili turdaki en iyi ajan seçilerek karşılaştırma yapılmıştır. Bu sebeple zaman zaman bazı ajanlar tezdekinden iyi sonuçlar almıştır. Fakat ajanlardan herhangi biri sabit olarak seçilirse tezde eğitilen ajanın büyük farklarla önde olduğu görülebilecektir.

Munoz ve ekibinin çalışmasında [16] Simmerson'un davranışları ile eğitilen ajan tamamlayabildiği ölçütte hızlı olabilirken pistlerin yarısından çoğunu tamamlayamamıştır. Bu açıdan Simmerson ile karşılaştırılan sonuçları genele yaymak mümkün olmamaktadır. Elle kodlanmış botun taklit edildiği ajan ise pistleri tamamlamış fakat tezdeki ajanın çok gerisinde kalmıştır. Bu karşılaştırma düzleminde dahi tezde eğitilen ajan çoğunlukla pistleri daha hızlı tamamlamıştır.

Benzer şekilde Cardamone [8] ve ekibinin çalışmasında ağırlıklı olarak Ruudskogen pistinde, Neat algoritması ve ϵ-greedy-improved seçim stratejisiyle eğitilen ajan en iyi sonuçları vermiştir. Diğer sonuçlar tez ile karşılaştırılamayacak kadar geride kalmıştır.

Verma ve ekibinin çalışmasında [11] amaç hızlı bir ajan elde etmek olmamakla beraber, karşılaştırma yapmak için uygun bir aday olmaktadır. Diğer çalışmalarda yapıldığı gibi, turu tamamlayabilen ve en hızlı ajan seçilip karşılaştırma yapılmıştır. Zira algoritmalar test edilen pistlerin yarısında turu tamamlayamamıştır.

Ganesh ve ekibi tarafından hazırlanan çalışmada [9] paylaşılan Dropbox linki [39] vasıtası ile erişilebilecek görüntüden anlaşılabileceği üzere, ajan slalom açısından teze göre daha zayıf bir performans sergilemektedir. Aracın virajlarda yoldan çıkması da kararlılık açısından daha zayıf olduğunu gösterir niteliktedir. Fren-Dene mekanizmasının faydasını, görüntülerde yarışan ajanın fren yapmadığını gözlemleyerek de anlayabiliriz.

Wang ve ekibinin çalışmasında [10] paylaşılan Dropbox linki [40] üzerinden izlenebilecek diğer bir görüntüde ise kaliteden dolayı hüküm vermek çok mümkün değildir. Fakat eylemlerin dağılımından bu çalışmada eğitilen ajanın fren yapmadığı

ve slalom yaptığı görülebilmektedir.

Tezde elde edilen ajanın eğitimi yaklaşık 6.10^6 adım süresince, 5.10^3 bölüm için Intel i9-9900k işlemci ve GeForce RTX 2060 ekran kartı ile sağlanmıştır.

4
SONUÇ ve ÖNERİLER

TORCS kullanılarak hazırlanan yarışma ortamında 2 farklı algoritma ile başarı elde edilmiştir. Ajanlar 190 km/s hızlara ulaşmakta ve genelde ortalama 140 km/s'i tutturmaktadırlar. Ajanların daha önce görmediği pistlerde de başarılı olduğunu ve genelleme konusunda sorun yaşamadığı görülmüştür. Fren-Dene mekanizmasının başarılı olduğu, ajanların düzlüklerde hızlarını artırarak viraj öncesi yavaşladıkları gözlemlenmiştir.

Slalom problemi LSTM ile kısmen çözüme ulaşmıştır, fakat tümüyle çözülmemiştir. Bu problemin ana kaynağı ödül fonksiyonu olarak düşünülmektedir. Eğer daha iyi formüle edilebilirse, bahsi geçen yumuşak kırpma ile ödül fonksiyonlarıyla aracın slalomu azaltılabilir. Bu durum, yalnızca yüksek hızlarda oluştuğu için, normal sürüş pratiklerinde bir problem yaşanmayacaktır. SAC-LSTM'in önceden hazırlanmış yapay zeka kullanmayan ajanlarla yarışı ve Rainbow DQN ile SAC-LSTM arasında geçen yarışlar YouTube linkinden [41] izlenebilir. Gelecekteki çalışmalarda, ajanın zorlandığı pistlerde daha çok eğitim yapması sağlanarak daha gürbüz bir ajan elde edilmesi mümkün olabilir.

SAC'ın Rainbow DQN'den daha iyi performans sergilemesinin sebeplerinin entropi ve eylem uzayı olduğu düşünülmektedir. SAC algoritmasının entropiyi maksimize etmeye çalışması, sisteme dair emin olunamayan, bilgi sahibi olunamayan kısımlarını daha çok keşfetmesini sağlamaktadır. Bu sayede zor bölgeleri öğrenmesi kolaylaşmaktadır. Bununla beraber sürekli eylem uzayı gaz, fren ve direksiyon kabiliyeti açısından avantaj sağlamaktadır. Bununla beraber, iki algoritma için de aynı anda fren ve gaza basılmasının engellenmesi, eğitim için önemli bir faktör olmuştur.

Bu sistemin bir araba yarışı mekanizması olması sebebiyle literatürde yer alan diğer çalışmalar ile birebir karşılaştırmak mümkün olmamaktadır. Skor hesabının farklı olması skor karşılaştırmasını, amacın yarıştan öte olması da ajanların netlik içerisinde karşılaştırılmasını olanaksız kılmaktadır. Çoğu araştırmacının ajanı aynı pistlerde eğitmemesi ve eğitilmiş ajanını paylaşmaması sebebiyle ajanları birbirleriyle

yarıştırmak da olanaklı olmamaktadır.

Bir önceki bölümde yapılan karşılaştırmalardan görülebileceği üzere mevzubahis çalışmalarda, karşılaştırma için tek bir ajan seçimine gidilmemiştir. Böyle bir durumda tezdeki ajan çok büyük ölçüde hızlı ve kararlı gözükecektir. Bu da karşılaştırılan sonuçların yanlış anlaşılmasına sebep olabilir. Bu nedenle diğer çalışmalarda farklı algoritmalarla elde edilen sonuçların en iyi olanları elle özenle seçilmiştir. Bu halde dahi tezde eğitilen ajan çok büyük kısmında diğer ajanlardan daha hızlı olmaktadır.

Bu çalışma, otonom araçların eğitilmesinde derin pekiştirmeli öğrenme yöntemlerinin kullanımı konusunda bir adım olmaktadır. Bu yöntemlerin herhangi bir eğitici ve hazır veri olmadan sürüş/yarış öğrenebilmesi önemli bir temel oluşturmaktadır. Ayrıca, bu çalışmanın kolaylıkla fiziksel ortama aktarılabilir olması ve simülasyon ortamında eğitilen ajanların gerçek ortama taşınabilecek olması gelecekteki çalışmalar için fayda sağlamaktadır. Derin pekiştirmeli öğrenme algoritmalarının, diğer makine öğrenmesi, kontrol ve yazılım pratikleriyle harmanlanarak yakın zamanda otonom araç endüstrisinde güçlü bir şekilde kullanılması beklenmektedir.

Kaynakça

[1] J. Koutnik, G. Cuccu, J. Schmidhuber, F. Gomez, "Evolving large-scale neural networks for vision-based reinforcement learning," in *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, ACM, 2013, pp. 1061–1068.

[2] D. Loiacono, A. Prete, P. L. Lanzi, L. Cardamone, "Learning to overtake in torcs using simple reinforcement learning," in *IEEE Congress on Evolutionary Computation*, IEEE, 2010, pp. 1–8.

[3] X. Zong, G. Xu, G. Yu, H. Su, C. Hu, "Obstacle avoidance for self-driving vehicle with reinforcement learning," *SAE International Journal of Passenger Cars-Electronic and Electrical Systems*, vol. 11, no. 07-11-01-0003, pp. 30–39, 2017.

[4] X. Li, X. Xu, L. Zuo, "Reinforcement learning based overtaking decision-making for highway autonomous driving," in *2015 Sixth International Conference on Intelligent Control and Information Processing (ICICIP)*, IEEE, 2015, pp. 336–342.

[5] A. E. Sallab, M. Abdou, E. Perot, S. Yogamani, "End-to-end deep reinforcement learning for lane keeping assist," *arXiv preprint arXiv:1612.04340*, 2016.

[6] S. Whiteson P. Stone, "Evolutionary function approximation for reinforcement learning," *Journal of Machine Learning Research*, vol. 7, no. May, pp. 877–917, 2006.

[7] J. Reeder, R. Miguez, J. Sparks, M. Georgiopoulos, G. Anagnostopoulos, "Interactively evolved modular neural networks for game agent control," in *2008 IEEE Symposium On Computational Intelligence and Games*, IEEE, 2008, pp. 167–174.

[8] L. Cardamone, D. Loiacono, P. L. Lanzi, "Learning drivers for torcs through imitation using supervised methods," in *2009 IEEE symposium on computational intelligence and games*, IEEE, 2009, pp. 148–155.

[9] A. Ganesh, J. Charalel, M. D. Sarma, N. Xu, *Deep reinforcement learning for simulated autonomous driving*, 2016.

[10] S. Wang, D. Jia, X. Weng, "Deep reinforcement learning for autonomous driving," *arXiv preprint arXiv:1811.11329*, 2018.

[11] A. Verma, V. Murali, R. Singh, P. Kohli, S. Chaudhuri, "Programmatically interpretable reinforcement learning," *arXiv preprint arXiv:1804.02477*, 2018.

[12] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[13] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, 2016, pp. 1928–1937.

[14] S. Sharma, A. S. Lakshminarayanan, B. Ravindran, "Learning to repeat: Fine grained action repetition for deep reinforcement learning," *arXiv preprint arXiv:1702.06054*, 2017.

[15] J. Koutnik, J. Schmidhuber, F. Gomez, "Online evolution of deep convolutional network for vision-based reinforcement learning," in *International Conference on Simulation of Adaptive Behavior*, Springer, 2014, pp. 260–269.

[16] J. Munoz, G. Gutierrez, A. Sanchis, "Controller for torcs created by imitation," in *2009 IEEE Symposium on Computational Intelligence and Games*, IEEE, 2009, pp. 271–278.

[17] Y. Gao, J. Lin, F. Yu, S. Levine, T. Darrell, *et al.*, "Reinforcement learning from imperfect demonstrations," *arXiv preprint arXiv:1802.05313*, 2018.

[18] R. S. Sutton A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[19] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[20] R. S. Sutton, D. A. McAllester, S. P. Singh, Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in neural information processing systems*, 2000, pp. 1057–1063.

[21] V. R. Konda J. N. Tsitsiklis, "Actor-critic algorithms," in *Advances in neural information processing systems*, 2000, pp. 1008–1014.

[22] M. Fortunato, M. G. Azar, B. Piot, J. Menick, I. Osband, A. Graves, V. Mnih, R. Munos, D. Hassabis, O. Pietquin, *et al.*, "Noisy networks for exploration," *arXiv preprint arXiv:1706.10295*, 2017.

[23] T. Haarnoja, H. Tang, P. Abbeel, S. Levine, "Reinforcement learning with deep energy-based policies," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, JMLR. org, 2017, pp. 1352–1361.

[24] L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning, *et al.*, "Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures," *arXiv preprint arXiv:1802.01561*, 2018.

[25] K. Cobbe, O. Klimov, C. Hesse, T. Kim, J. Schulman, "Quantifying generalization in reinforcement learning," *arXiv preprint arXiv:1812.02341*, 2018.

[26] L. Perez J. Wang, "The effectiveness of data augmentation in image classification using deep learning," *arXiv preprint arXiv:1712.04621*, 2017.

[27] K. Lee, K. Lee, J. Shin, H. Lee, "A simple randomization technique for generalization in deep reinforcement learning," *arXiv preprint arXiv:1910.05396*, 2019.

[28] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, D. Silver, "Rainbow: Combining improvements in deep reinforcement learning," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[29] H. Van Hasselt, A. Guez, D. Silver, "Deep reinforcement learning with double q-learning," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[30] T. Schaul, J. Quan, I. Antonoglou, D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[31] Z. Wang, T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, N. De Freitas, "Dueling network architectures for deep reinforcement learning," *arXiv preprint arXiv:1511.06581*, 2015.

[32] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine learning*, vol. 3, no. 1, pp. 9–44, 1988.

[33] M. G. Bellemare, W. Dabney, R. Munos, "A distributional perspective on reinforcement learning," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, JMLR. org, 2017, pp. 449–458.

[34] B. T. Polyak A. B. Juditsky, "Acceleration of stochastic approximation by averaging," *SIAM Journal on Control and Optimization*, vol. 30, no. 4, pp. 838–855, 1992.

[35] K. Güçkıran, *Torcs-RL-Agent*, `https://github.com/kivancguckiran/torcs-rl-agent`, [Online; accessed 05-02-2020], 2020.

[36] N. Yoshida, *Gym-TORCS*, `https://github.com/ugo-nama-kun/gym_torcs`, [Online; accessed 05-02-2020], 2020.

[37] M. Jaritz, R. De Charette, M. Toromanoff, E. Perot, F. Nashashibi, "End-to-end race driving with deep reinforcement learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 2070–2075.

[38] B. Renukuntla, S. Sharma, S. Gadiyaram, V. Elango, V. Sakaray, *The road to be taken, a deep reinforcement learning approach towards autonomous navigation*, `https://github.com/charlespwd/project-title`, 2017.

[39] A. Ganesh, J. Charalel, M. D. Sarma, N. Xu, *TORCS-Autopilot-Footage-Final*, `https://www.dropbox.com/sh/b4623soznsjmp12/AAA1UD8_oaa94FgFC6eyxReya`, [Online; accessed 05-02-2020], 2020.

[40] S. Wang, D. Jia, X. Weng, *drive4.mov*, `https://www.dropbox.com/s/balm1vlajjf50p6/drive4.mov`, [Online; accessed 05-02-2020], 2020.

[41] K. Güçkıran, *TORCS - Deep Reinforcement Learning Agent - PESLA*, `https://youtu.be/f82EBvPKyDI`, [Online; accessed 05-02-2020], 2020.

Tezden Üretilmiş Yayınlar

İletişim Bilgileri: kivancguckiran@gmail.com

Konferans Bildirisi

1. Güçkıran, K., & Bolat, B. Autonomous Car Racing in Simulation Environment Using Deep Reinforcement Learning. In 2019 Innovations in Intelligent Systems and Applications Conference (ASYU) (pp. 1-6). IEEE.