# DERİN TAKVİYELİ ÖĞRENME TABANLI OTONOM SÜRÜŞ

**Mehmet Yaşar Osman ÖZTURAN**




T.C.
BURSA ULUDAĞ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

# DERİN TAKVİYELİ ÖĞRENME TABANLI OTONOM SÜRÜŞ

Mehmet Yaşar Osman ÖZTURAN
0009-0006-3978-0034

Prof. Dr. Ahmet Emir DİRİK
(Danışman)

YÜKSEK LİSANS
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

BURSA – 2025
**Her Hakkı Saklıdır**

# ÖZET

Yüksek Lisans

## DERİN TAKVİYELİ ÖĞRENME TABANLI OTONOM SÜRÜŞ

**Mehmet Yaşar Osman ÖZTURAN**

Bursa Uludağ Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

**Danışman:** Prof. Dr. Ahmet Emir Dirik

Otonom araçlar, trafik güvenliğini artırmak, insan hatalarını azaltmak ve ulaşımda süreklilik sağlamak gibi hedeflerle, günümüzde önemli bir araştırma alanı haline gelmiştir. Ancak mevcut otonom sürüş sistemleri, çoğunlukla önceden tanımlanmış kurallara ve klasik algoritmalara dayandıkları için, insan sürücüler kadar esnek ve doğal kararlar alma konusunda yetersiz kalmakta; bu da öngörülebilirlik ve güvenlik açısından sorun yaratmaktadır. Özellikle insan benzeri sürüş davranışlarının modellenmesi; yol takibi, çevresel etkileşim, konforlu manevra üretimi ve güvenli sürüş performansı açısından büyük önem taşımaktadır.

Bu tez çalışmasında, insan benzeri pürüzsüz sürüş davranışını hedefleyen, derin takviyeli öğrenme tabanlı modüler bir otonom sürüş mimarisi geliştirilmiştir. Sistem; bilgi çıkarımı, karar verme, aksiyon filtreleme gibi işlevsel bileşenlerin ayrıldığı, esnek modüllerden oluşmakta ve hiyerarşik bir akış sergilemektedir. Sensör verileri, oto-kodlayıcılar kullanılarak düşük boyutlu temsillere dönüştürülmüş ve araç dinamikleriyle birlikte karar ve kontrol modülüne aktarılmıştır. Sistemin karar ve kontrol mekanizması, Soft Aktör-Kritik (SAC) algoritması ile eğitilmiş olup, ajanın daha pürüzsüz ve doğal manevralar üretmesi amacıyla çeşitli aksiyon filtreleme teknikleri entegre edilmiştir. Simülasyon sonuçları, önerilen yaklaşımın insan benzeri, pürüzsüz ve güvenli sürüş davranışları elde etmede etkili olduğunu göstermektedir.

**Anahtar Kelimeler:** Otonom sürüş, insan benzeri sürüş davranışı, derin takviyeli öğrenme, soft aktör-kritik, aksiyon filtreleme
**2025, xvi + 87 sayfa.**

## ABSTRACT

MSc

**DEEP REINFORCEMENT LEARNING BASED AUTONOMOUS DRIVING**

**Mehmet Yaşar Osman ÖZTURAN**

Bursa Uludağ University
Graduate School of Natural and Applied Sciences
Department of Computer Engineering

**Supervisor:** Prof. Dr. Ahmet Emir Dirik

Autonomous vehicles constitute a prominent field of research, driven by the objectives of enhancing traffic safety, mitigating human errors, and improving transportation efficiency. However, contemporary autonomous driving systems, which are predominantly reliant on predefined rules and classical algorithms, exhibit significant limitations in emulating the flexible and naturalistic decision-making characteristic of human drivers. This deficiency compromises both predictability and safety. Specifically, the modeling of human-like driving behavior is of paramount importance for key functionalities including lane keeping, interaction with the environment, smooth maneuver generation, and overall safe driving performance.

This thesis introduces a modular autonomous driving architecture founded on deep reinforcement learning, developed with the objective of achieving human-like smooth driving behavior. The system is composed of flexible modules with distinct functional responsibilities—namely, information extraction, decision-making, and action filtering—and is structured in a hierarchical manner. Sensor data are first encoded into low-dimensional representations using autoencoders, subsequently fused with vehicle dynamics data, and then fed into the decision and control module. The system's decision and control mechanism is trained using the Soft Actor-Critic (SAC) algorithm. Furthermore, a suite of action filtering techniques is integrated to facilitate the agent's generation of smoother and more naturalistic maneuvers. Simulation results demonstrate that the proposed architecture is effective in realizing human-like, smooth, and safe driving behaviors.



**Key words:** Autonomous driving, human-like driving behaviour, deep reinforcement learning, soft actor-critic, action filtering
**2025, xvi + 87 pages.**

## TEŞEKKÜR

Bu tezin hazırlanmasında başta değerli danışmanım Prof. Dr. Ahmet Emir DİRİK'e sonsuz teşekkürlerimi sunarım. Kendisinin engin bilgi birikimi, yapıcı eleştirileri ve her daim gösterdiği sabır ile yol göstericiliği, bu çalışmayı başarıyla tamamlamamda kilit rol oynamıştır.

Araştırma sürecim boyunca bana destek olan, sabırları ve anlayışlarıyla yanımda duran aileme şükranlarımı sunarım. Onların koşulsuz desteği, bu zorlu süreçte en büyük motivasyon kaynağım olmuştur. Ayrıca, her zorlandığımda bana ilham veren ve varlığıyla güç katan sevgili aile köpeğimiz Ice'a da özel teşekkürlerimi iletmek isterim.

Son olarak, akademik yolculuğumda bana destek olan tüm hocalarıma ve isimlerini tek tek zikredemediğim herkese teşekkür ederim.

Mehmet Yaşar Osman ÖZTURAN
……/……/…….

# İÇİNDEKİLER

**Sayfa**

ÖZET ..... viii
ABSTRACT ..... ix
TEŞEKKÜR ..... x
SİMGELER ve KISALTMALAR DİZİNİ ..... xii
ŞEKİLLER DİZİNİ ..... xiv
ÇİZELGELER DİZİNİ ..... xvi
1. GİRİŞ ..... 1
2. KURAMSAL TEMELLER ..... 5
2.1. Otonom Araçların Donanımsal ve Yazılımsal Mimarisi ..... 5
2.2. Takviyeli Öğrenme ..... 10
2.2.1. Derin takviyeli öğrenme ..... 15
2.2.2. Takviyeli öğrenme yaklaşımları içinde sınıflandırmalar ..... 18
2.2.3. Keşif ve mevcudu kullanma ikilemi ..... 20
2.3. Otonom Araçlar Literatüründe Takviyeli Öğrenme ..... 22
3. MATERYAL ve YÖNTEM ..... 27
3.1. Donanımsal ve Yazılımsal Bileşenler ..... 27
3.2. Simülasyon Ortamı ..... 28
3.3. Durum Uzayı Tasarımı ..... 30
3.4. Karar ve Kontrol Algoritması ..... 35
3.5. Aksiyon Gürültüsü ..... 41
3.6. Aksiyon Filtreleme ..... 41
3.7. Ödül Denklemi Tasarımı ..... 43
3.8. Başarı Metrikleri ..... 46
4. BULGULAR ..... 47
4.1. Şerit Takibi Deneyleri ..... 50
4.1.1. Eğitim süreci performansları ..... 51
4.1.2. Test performansları ..... 58
4.2. Şerit Takibi ve Engelden Kaçma Deneyleri ..... 61
4.2.1. Eğitim süreci performansları ..... 62
4.2.1. Test performansları ..... 68
4.3. Donanımsal Performanslar ..... 71
5. TARTIŞMA ve SONUÇ ..... 75
KAYNAKLAR ..... 81
ÖZGEÇMİŞ ..... 87

## SİMGELER ve KISALTMALAR DİZİNİ

| Simgeler | Açıklama |
|---|---|
| $a$ | Aksiyon uzayı |
| $a_t$ | Bir *t* zaman adımında gerçekleştirilen aksiyon |
| $\tilde{a}_t$ | Bir *t* zaman adımında filtrelenmiş aksiyon |
| $d$ | Bölüm bitim sinyali |
| $G_t$ | Bir *t* zaman adımındaki kümülatif ödül |
| $k$ | Bölüm adımı |
| $n$ | Geriye doğru dikkate alınan aksiyon sayısı ve ağırlıklandırma için kullanılan en büyük katsayı |
| $s$ | Durum uzayı |
| $s_t$ | Bir *t* zaman adımında elde edilen durum |
| $T$ | Bir bölümü oluşturan toplam adım sayısı |
| $t$ | Zaman adımı |
| $w$ | Denge (trade-off) katsayısı |
| $y$ | DQN için hedef kalite değeri |
| $\alpha$ | Entropi ağırlık katsayısı |
| $\gamma$ | İndirim katsayısı |
| $\epsilon_t$ | Bir *t* zaman adımında normal dağılımdan örneklenen gürültü |
| $\theta$ | SAC algoritmasında kritik ağının parametreleri |
| $\mu$ | Ortalama |
| $\pi$ | Politika |
| $\sigma$ | Standart sapma |
| $\tau$ | Polyak ortalaması için yumuşak güncelleme katsayısı |
| $\phi$ | SAC algoritmasında aktör (politika) ağının parametreleri |
| $\psi$ | SAC algoritmasında ana durum değer ağının parametreleri |
| $\overline{\psi}$ | SAC algoritmasında hedef durum değer ağının parametreleri |

| Kısaltmalar | Açıklama |
|---|---|
| ADAS | Gelişmiş sürücü destek sistemleri |
| CLF | Özelleştirilmiş Kayıp Fonksiyonu (Customized Loss Function) |
| CNN | Evrişimli Sinir Ağları (Convolutional Neural Network) |
| CPU | Merkezi İşlem Birimi (Central Processing Unit) |
| CTE | Çapraz Yörünge Hatası (Cross Track Error) |
| DDDQN | Çift Derin Düello Q-Ağı (Deep Double Duelling Q-Network) |
| DDPG | Derin Deterministik Politika Gradyanı (Deep Deterministic Policy Gradient) |
| DDQN | Çift Derin Q-Ağı (Deep Double Q-Network) |
| DRL | Derin Takviyeli Öğrenme (Deep Reinforcement Learning) |
| DQN | Derin Q-Ağı (Deep Q-Network) |
| E2E | Uçtan Uca (End-To-End) |
| EMA | Üstel Hareketli Ortalama (Exponential Moving Average) |
| GPU | Grafik İşlem Birimi (Graphics Processing Unit) |
| LIDAR | Işık Algılama ve Uzaklık Tayini (Light Detection And Ranging) |
| MDP | Markov Karar Süreci (Markov Decision Process) |

| | |
|---|---|
| ME | Ortalama Hata (Mean Error) |
| ML | Makine Öğrenimi (Machine Learning) |
| OU | Ornstein-Uhlenbeck Süreci/Gürültüsü |
| PPO | Proksimal Politika Optimizasyonu (Proximal Policy Optimization) |
| RAM | Rastgele Erişimli Bellek (Random-Access Memory) |
| RL | Takviyeli Öğrenme (Reinforcement Learning) |
| SAC | Soft Aktör-Kritik (Soft Actor-Critic) |
| SL | Denetimli Öğrenme (Supervised Learning) |
| USL | Denetimsiz Öğrenme (Unsupervised Learning) |
| WMA | Ağırlıklı Hareketli Ortalama (Weighted Moving Average) |

# ŞEKİLLER DİZİNİ


| | | Sayfa |
|---|---|---|
| Şekil 2.1. | Otonom sistemlerde kullanılan yazılım mimarilerinin temsili yapıları | 9 |
| Şekil 2.2. | Makine öğrenimi paradigmalarını ve temel özelliklerini gösteren diyagram | 10 |
| Şekil 2.3. | Takviyeli öğrenmenin temelini oluşturan Markov karar süreci modeli | 11 |
| Şekil 3.1. | Simülasyon platformunun arayüzü | 29 |
| Şekil 3.2. | Simülasyon içi görüntüler | 29 |
| Şekil 3.3. | Oto-kodlayıcıların temsili yapısı | 31 |
| Şekil 3.4. | Kamera görüntüleri için oluşturulan oto-kodlayıcı | 32 |
| Şekil 3.5. | Kamera görüntüleri için oluşturulan oto-kodlayıcının farklı eğitim döngüsü değerlerinde yeniden oluşturulan görüntüler oluşturdukları görüntüler | 32 |
| Şekil 3.6. | LIDAR nokta bulutu verileri için oluşturulan oto-kodlayıcı yapısı | 33 |
| Şekil 3.7. | LIDAR verileri için oluşturulan oto-kodlayıcının farklı eğitim döngüsü değerlerinde yeniden oluşturulan sinyaller | 33 |
| Şekil 3.8. | Bir *t* anı için oluşturulan $s_t$ durum uzayı vektörünün yapısı gösterilmektedir. | 34 |
| Şekil 3.9. | Aktör-Kritik yapısına sahip takviyeli öğrenme algoritmaları için temsili eğitim süreci | 35 |
| Şekil 3.10. | Çalışmada SAC için oluşturulan aktör, kritik ve değer ağlarının yapıları | 36 |
| Şekil 3.11. | Şerit merkezine olan uzaklık (CTE) değerinin temsili | 43 |
| Şekil 3.12. | CTE ödülü $R_{CTE}$ için dağılım grafiği | 44 |
| Şekil 3.13. | Hız ödülü $R_{hız}$ için dağılım grafiği | 45 |
| Şekil 4.1. | Oluşturulan modüler sistemin mimarisi | 48 |
| Şekil 4.2. | Şerit takibi görevi eğitiminde modellerin bölüm başına adım sayısı grafiği | 52 |
| Şekil 4.3. | Şerit takibi görevi eğitimlerinde modellerin ödül değişim grafikleri | 53 |
| Şekil 4.4. | Şerit takibi görevi eğitiminde modellerin CTE ve hız değişim grafikleri | 54 |
| Şekil 4.5. | Şerit takibi görevi eğitiminde modellerin bölüm başına ortalama hata değişim grafiği | 55 |
| Şekil 4.6. | Şerit takibi görevi eğitiminde modellerin bölüm başına ortalama ardışık aksiyon değişim grafiği | 56 |
| Şekil 4.7. | Şerit takibi görevi eğitiminde modellerin toplam hata ortalaması ve toplam aksiyon değişim ortalaması değerleri dağılım grafiği | 57 |
| Şekil 4.8. | Şerit takibi görevi testlerinde modellerin ortalama hata ve ortalama aksiyon değişim değerleri dağılım grafiği | 58 |
| Şekil 4.9. | Şerit takibi ve engelden kaçma görevi eğitiminde modellerin bölüm başına adım sayısı grafiği | 63 |

| | | |
|---|---|---|
| Şekil 4.10. | Şerit takibi ve engelden kaçma görevi eğitiminde modellerin ödül değişim grafikleri | 64 |
| Şekil 4.11. | Şerit takibi ve engelden kaçma görevi eğitiminde modellerin CTE ve hız değişim grafikleri | 65 |
| Şekil 4.12. | Şerit takibi ve engelden kaçma görevi eğitiminde modellerin bölüm başına ortalama hata grafiği | 66 |
| Şekil 4.13. | Şerit takibi ve engelden kaçma görevi eğitiminde modellerin bölüm başına ortalama ardışık aksiyon değişim grafiği | 66 |
| Şekil 4.14. | Şerit takibi ve engelden kaçma görevi eğitiminde modellerin ortalama toplam hata ve ortalama toplam aksiyon değişim değerleri dağılım grafiği | 67 |
| Şekil 4.15. | Şerit takibi ve engelden kaçma görevi testlerinde modellerin ortalama hata ve ortalama ardışık aksiyon değişim değerleri dağılım grafiği | 69 |
| Şekil 4.16. | Şerit takibi görevi eğitim süreci için GPU belleği kullanımı | 71 |
| Şekil 4.17. | Şerit takibi görevi eğitim süreci için kullanılabilir RAM grafiği | 72 |
| Şekil 4.18. | Şerit takibi ve engelden kaçma görevi eğitim süreci için GPU belleği kullanımı | 73 |
| Şekil 4.19. | Şerit takibi ve engelden kaçma görevi eğitim süreci için kullanılabilir RAM grafiği | 73 |

# ÇİZELGELER DİZİNİ


| | | Sayfa |
|---|---|---|
| Çizelge 2.1. | Otonom araçlarda kullanılan sensörler ve temel kullanım amaçları | 6 |
| Çizelge 2.2. | Otonom sürüş literatüründeki RL çalışmaları | 23 |
| Çizelge 4.1 | Eğitimler için oluşturulan modellerin yapıları | 49 |
| Çizelge 4.2. | Şerit takibi görevinde SAC algoritması için kullanılan eğitim parametreleri | 51 |
| Çizelge 4.3. | Şerit takibi görevinde modellerin eğitim süreleri | 51 |
| Çizelge 4.4. | Şerit takibi görevi eğitiminde modellerin toplam hata ortalaması ve toplam aksiyon değişim ortalaması yüzdesel değerleri | 58 |
| Çizelge 4.5. | Şerit takibi görevi testlerinde modellerin ortalama hata ve ortalama ardışık aksiyon değişim yüzdeleri | 59 |
| Çizelge 4.6. | Şerit takibi ve engelden kaçma görevi eğitiminde SAC algoritması için kullanılan parametreler | 62 |
| Çizelge 4.7. | Şerit takibi ve engelden kaçma görevi için modellerin eğitim süreleri | 62 |
| Çizelge 4.8. | Şerit takibi ve engelden kaçma görevi eğitiminde modellerin ortalama toplam hata ve ortalama toplam ardışık aksiyon değişim yüzdeleri | 68 |
| Çizelge 4.9. | Şerit takibi ve engelden kaçma görevi testlerinde modellerin ortalama hata ve ortalama ardışık aksiyon değişim yüzdeleri | 70 |
| Çizelge 4.10. | Şerit takibi görevi eğitim sürecinde toplanan deneyim adedi | 72 |
| Çizelge 4.11. | Şerit takibi ve engelden kaçma görevi eğitim sürecinde toplanan deneyim adedi | 74 |

# 1. GİRİŞ

21. yüzyılın başından itibaren ulaşım endüstrisindeki gelişmeler önemli dönüşümlere yol açarak gelişmiş sürücü destek sistemlerinin (ADAS) ve otonom araçların toplum için erişilebilirliğini artırmıştır. Günümüzde toplumun önemli bir kısmı tarafından benimsenen otonom araçlar ve otonom sürüş teknolojileri hızlı bir şekilde gelişmeye devam ederken ulaşım alışkanlıklarını daha güvenli, konforlu ve verimli bir geleceğe taşımayı vaat etmektedir. Bu teknolojinin sunduğu yeniliklerin aynı zamanda ekonomik, sosyal ve çevresel faktörler başta olmak üzere çeşitli alanlarda insan yaşamına avantajlar sağlayacağı öngörülmektedir. Buna ilişkin olarak Duarte ve Ratti (2018) tarafından hazırlanan bir çalışmada otonom araçların insan ve şehir yaşamında yaratabilecek etkileri detaylı bir şekilde ele alınmıştır. Söz konusu çalışmada otonom araçların çok yoğun nüfusa sahip bölgelerde kolay erişim imkanları yaratabileceği ve trafikte daha az zaman geçirmeyi sağlayabileceği belirtilmektedir. Bu olanakların kentsel yayılmayı artırarak insanların kırsal bölgelere veya banliyölere yönelmesini teşvik edeceği öngörülmektedir. Bu durumun şehir merkezlerindeki yoğunluğu azaltarak hem yaşam hem de çevre kalitesine katkıda bulunabileceği düşünülmektedir. Benzer şekilde otonom araçların araç paylaşım planları içerisinde kullanılması ile şehirlerdeki araç sayısının ve buna bağlı olarak park alanı ihtiyacının azalması öngörülmektedir. Etkileri bunlarla sınırlı kalmayan otonom araç teknolojilerinin zaman tasarrufu, enerji tüketimi optimizasyonu, karbon salınımında azalma gibi avantajlar da sunarak bireysel ve toplu ulaşım sistemlerinde köklü dönüşümlere yol açacak potansiyele sahip olduğu bilinmektedir. Bu sebeplerle otonom araçların günümüz şehir yaşamının ve tasarımının yeniden gözden geçirilmesine neden olacağı düşünülmektedir.

Bunlarla beraber ADAS ve otonom sürüşün Avrupa'da giderek daha fazla tercih edilmesinin trafik güvenliğini artırarak gerçekleşen kaza sayısını 2030 yılına kadar yaklaşık %17 azaltabileceğini gösteren çalışmalar da bulumaktadır (Deichmann vd., 2023). Bu durum otonom sürüş teknolojilerinin sürücüler, endüstri ve toplum için önemli değerler üretebileceğinin sinyallerini vermektedir. Otonom araç endüstrisi verileri üzerinde yapılan analizlerde 2035 yılı itibari ile otomotiv pazarının yaklaşık %17-57'sine üçüncü ve dördüncü seviye otonom araçların hakim olması

beklenmektedir. Bu durum otonom araçların toplum tarafından kabul görme oranının artması sonucu otonom araçlara yönelik giderek artan bir talebe işaret etmektedir.

Pazar payı artmakta olan otonom araçların yeni teknolojiler ile dönüşümü sürecinde çeşitli yapay zeka yaklaşımları benimsenmektedir. Bu yaklaşımlara ilişkin çalışmalar akademi ve endüstride hızlı bir şekilde yer bulmaktadır. Yapay zeka yaklaşımları sayesinde otonom araçlarda karar alma mekanizmalarının daha etkili bir seviyeye ulaşması sağlanmaktadır (Kuutti vd., 2021). Günümüzde yapay zeka teknolojileri otonom sürüş sistemlerinin daha güvenilir, konforlu ve çevik bir şekilde çalışmasını sağlamak adına veri işleme, sensör füzyonu, karar verme, tahmin etme ve kontrol gibi temel alanlarda yenilikçi çözümler sunabilmektedir. Bu teknolojilerin otonom sürüş sistemlerinde hızla uyarlanması hem akademik çalışmaların ivme kazanmasına hem de endüstriyel uygulamalarda rekabetin yoğunlaşmasına yol açmaktadır (Bachute & Subhedar, 2021).

Günümüzde yapay zeka teknolojileri ile entegre edilmiş otonom araçlar için kullanılan sistem mimarileri uçtan uca (end-to-end, E2E) ve modüler olmak üzere iki temel kategoride ele alınmaktadır (Grigorescu vd., 2020). Bu bağlamda Bachute ve Subhedar (2021) tarafından yapılan çalışmanın yanı sıra Huang ve Chen (2020) tarafından gerçekleştirilen çalışmalarda da otonom sürüş uygulamalarında hem uçtan uca hem de modüler sistem mimarileri (ilgili modüller açısından) kapsamlı bir şekilde değerlendirilmiştir. Bu sistemlerde evrişimli sinir ağlarından (convolutional neural network, CNN) uzun kısa süreli bellek (long short-term memory) sinir ağlarına kadar birçok farklı yapay zeka modeli kullanılabilmektedir.

Alandaki öncü kurum ve şirketler tarafından otonom sürüş performansını mükemmelleştirme amacıyla çeşitli yapay zeka yaklaşımları hızla geliştirilmeye devam etmektedir. Geliştirilen yapay zeka yaklaşımları insan faktörünü ortadan kaldırma ve sürüş teknolojilerini insan yeteneklerinin ötesine taşıma hedefiyle şekillenmektedir. Bu amaçlar doğrultusunda sıklıkla insandan ilham alan yaklaşımlar incelenmekte ve denenmektedir. Yapılan çalışmalar (Plebe vd., 2024), otonom araçlarda karar verme, kontrol ve hatta algılama alanlarında insan davranışlarından, bilişsel mekanizmalarından

ve çok çeşitli sinirsel mimarilerinden yararlanarak otonom sürüş sistemleri tasarımı için farklı yöntemler ele alındığını göstermektedir. Bu durum akademik alanda insan benzeri bilişsel yeteneklere sahip daha güvenli, doğal ve verimli otonom sistemler geliştirme çabalarını vurgulamaktadır. İnsan benzeri otonom sürüş deneyimi geliştirmek amacıyla mantık, istatistik ya da model tabanlı klasik yaklaşımlardan yeni nesil makine öğrenimi (Machine Learning, ML) yaklaşımlarına kadar birçok farklı yöntemden destek alınabilmektedir. Gerçekleştirilen çalışmalarda bu yaklaşımların özellikleri, avantaj ve dezavantajları ele alınmış, özellikle de yapay zeka temelli yaklaşımlardan taklit öğrenmenin (imitation learning) başarısı vurgulanmıştır (T. Zhang vd., 2023). Taklit öğrenmenin insan sürüş deneyimlerinden elde edilen yüksek boyutlu veri setleri ile eğitimi sonucu, insan sürüş stratejisine olabildiğince yaklaşık bir sürüş deneyimi sağladığı gözlemlenebilmektedir. Bunların yanı sıra son yıllarda akademik araştırmalarda önemli bir gelişim kateden ve potansiyel vadeden derin takviyeli öğrenme (Deep Reinforcement Learning, DRL) tabanlı yaklaşımların otonom araçlarda kullanılmasının insan benzeri sürüş deneyimine ulaşmayı hızlandıracağı öngörülmektedir. DRL yaklaşımları yapısı gereği diğer yapay zeka tabanlı yaklaşımlardan önemli noktalarda farklılık göstermekte ve robotik sistemler için büyük avantajlar sağlamaktadır.

Bu tezde, modüler bir sistem mimarisinden yararlanılarak DRL tabanlı bir otonom sürüş sistemi geliştirmek ve bu yolla insan benzeri pürüzsüz bir sürüş deneyimini sağlamaktır. Bu bağlamda sistemin karar verme ve kontrol süreçlerini gerçekleştirmek için güncel ve başarısını kanıtlamış bir DRL yaklaşımı olan Soft Aktör-Kritik (Soft Actor-Critic, SAC) algoritması tercih edilmiştir. Oluşturulan sisteme çeşitli aksiyon filtreleme yöntemlerini içeren bir modül entegre edilerek pürüzsüz bir sürüş deneyimi hedeflenmiştir.

Bu tez çalışmasının “Kuramsal Temeller” bölümünde, çalışmanın altyapısını oluşturmak amacıyla otonom araç sistemleri ele alınmakta, takviyeli öğrenmenin (Reinforcement Learning, RL) temel kavramları tanıtılmakta, otonom sürüşteki yeri incelenmekte ve bu tezin literatüre katkısına değinilmektedir. “Materyal ve Yöntem” bölümünde tezin temelini oluşturan deneysel tasarım ele alınmakta, kullanılan DRL algoritmasının ayrıntılarına ve oluşturulan modüllerin yapısına odaklanılmıştır.

“Bulgular” bölümünde gerçekleştirilen deneyler ve bu deneyler sonucunda elde edilen bulgular sunulmuş ve analizleri yapılmıştır. “Tartışma ve Sonuç” bölümünde deney sonuçlarından elde edilen bilgiler ışığında çıkarımlar yapılmakta, kullanılan materyal ve yöntemler değerlendirilmekte ve gelecek çalışmalara yönelik öneriler sunulmaktadır.

# 2. KURAMSAL TEMELLER

Bu bölümde, çalışmanın altyapısını anlamak ve oluşturmak amacıyla otonom araç sistemlerine değinilmekte, çalışmanın temelini oluşturan takviyeli öğrenme (Reinforcement Learning, RL) kavramı tanıtılmakta ve takviyeli öğrenmenin otonom sürüş içerisindeki yeri ele alınmaktadır.

Otonom araçlar, çevresel algılama, karar verme ve hareket planlama yeteneklerini bir araya getirerek belli seviyelerde bir insan müdahalesine ihtiyaç duymadan güvenli bir şekilde hareket edebilen sistemlerdir. Bu tür araçlar, sensörlerden elde edilen verileri işleyerek yol durumu, trafik yoğunluğu ve olası engeller gibi çevresel faktörleri değerlendirebilmektedirler. Otonom araçlar bu değerlendirmelere bağlı olarak bir karar süreci gerçekleştirerek uygun bir hareket planı oluşturmaktadır. Kamera (Krizhevsky vd., 2012) ve ışık algılama ve uzaklık tayini (Light Detection and Ranging, LIDAR) (Schwarz, 2010) gibi sensörlerden veri elde etme sürecinin derin öğrenme teknikleriyle kolaylaştırılması, otonom araçların günümüz robotik sahasında en çok dikkat çeken ve en yüksek potansiyele sahip teknolojilerden biri olarak öne çıkmasını sağlamaktadır.

## 2.1. Otonom Araçların Donanımsal ve Yazılımsal Mimarisi

Donanımsal açıdan bir otonom araç mimarisi çeşitli sensörler, güç ve işlem üniteleri, haberleşme araçları ve benzeri yapılardan oluşmaktadır. Yazılımsal mimari, donanımsal yapıların verilerinden yararlanarak çeşitli işlevleri yerine getirir (Zong vd., 2018). Bu işlevler arasında çevreyi algılama, konum tespiti, yol planlama, karar verme ve hareket kontrolü bulunur. Bu bütünleşik işlevler, aracın belirli bir seviyede insan müdahalesi olmadan güvenli ve verimli hareket etmesini sağlar. Tasarlanan mimarilere bağlı olarak bu donanımsal yapıların ve yazılımsal işlevlerin gerçekleştirdiği görevler bir ölçüde birbirleri ile örtüşebilir. Bu durum işlevlerin arasındaki sınırların belirsizleşmesine sebep olabilmektedir. Bu belirsizlik durumu endüstride hem yazılımsal olarak hem de donanımsal olarak otonom araç mimarisinin çeşitlilik göstermesine olanak sağlamaktadır.

Otonom araçlarda en temel donanımsal ihtiyaç ortam verilerini elde etme görevinde olan sensörlerdir. Otonom araçlardaki sensör tipleri ve bu sensörlerin sayıları otonom aracın seviyesine, aracın gerçekleştirmesi beklenen görevin türüne, aracın yazılımsal yapısına ve benzeri durumlara göre değişkenlik gösterebilmektedir. Bununla birlikte profesyonel otonom araçlarda sensör seçimlerini etkileyen en önemli faktör sensörlerin sahip olduğu sınırlılıklardır (Parekh vd., 2022). Bu tür sınırlamalara örnek olarak kamera sensörlerinin karanlık ortamlarda verimli çalışmadığı gösterilebilmektedir. Bu sınırlılığı çözebilmek için kızılötesi kameraların ek olarak kullanılması ya da karanlık ortamlarda kamera verilerini destekleyecek farklı sensörlerin ilave edilmesi çözüm olarak sunulmaktadır. Bu bağlamda otonom araçlarda en sık kullanılan sensörlere ve bu sensörlerden elde edilen verilerin yazılımsal fonksiyonlarla sağladığı bilgilere, Ignatious (2022) ve Yeong (2021) tarafından gerçekleştirilen çalışmaların ışığında, Çizelge 2.1'de değinilmiştir.

**Çizelge 2.1.** Otonom araçlarda kullanılan sensörler ve kullanım amaçları

| Sensör | Kullanım amaçları |
|---|---|
| Kameralar | • Trafik ışığı ve işareti tanıma<br>• Şerit tespiti<br>• Araç ve yaya tespiti ve takibi<br>• Yol segmentasyonu<br>• Engel tespiti<br>• Haritalama |
| LIDAR ve RADAR | • Engel tespiti<br>• Araç ve yaya tespiti ve takibi<br>• Sürülebilir alan tespiti<br>• Haritalama |
| GPS ve IMU | • Araç konum tespiti |

Otonom sistemlerde bir sensörden elde edilen veriler ile çevreyi algılama hatasını en aza indirmek için birden fazla sensörden elde edilen veriler entegre edilebilmektedir. Buna örnek olarak Çizelge 2.1'de de gösterildiği gibi engel tespiti için kamera ve LIDAR

sensörleri kullanılabilmektedir. Tespit edilen engel ile ilgili daha doğru bir bilgi oluşturabilmek için iki sensörün bilgileri çeşitli yollarla birleştirilmektedir. Bu yaklaşım sensör füzyonu (sensor fusion) olarak tanımlanmaktadır ve bu şekilde çevreyi doğru bir şekilde tanımlayan daha tutarlı bir model geliştirmek hedeflenmektedir (Yeong vd., 2021). Sensör füzyonu, kendi içinde çok kapsamlı bir alan olmakla birlikte literatürde farklı sınıflandırma şekilleri mevcuttur. David ve Llinas (1997) tarafından ele alınan çalışmada çoklu sensör füzyonu üzerinde önemli tanımlamalar yapılmış ve sensör füzyonu kabaca üç farklı seviyede ele alınmıştır:

- Ham veri seviyesi: Uyumlu sensör verilerinin doğrudan birleştirilmesi,
- Özellik/durum vektörü seviyesi: Uyumlu ve/veya uyumsuz sensör verilerinden temsili özelliklerinin çıkarılarak birleştirilmesi,
- Karar seviyesi: Her sensörün bir varlığın konum, nitelik ve kimliği hakkında ön belirlemesinden sonraki veri eşlemesidir.

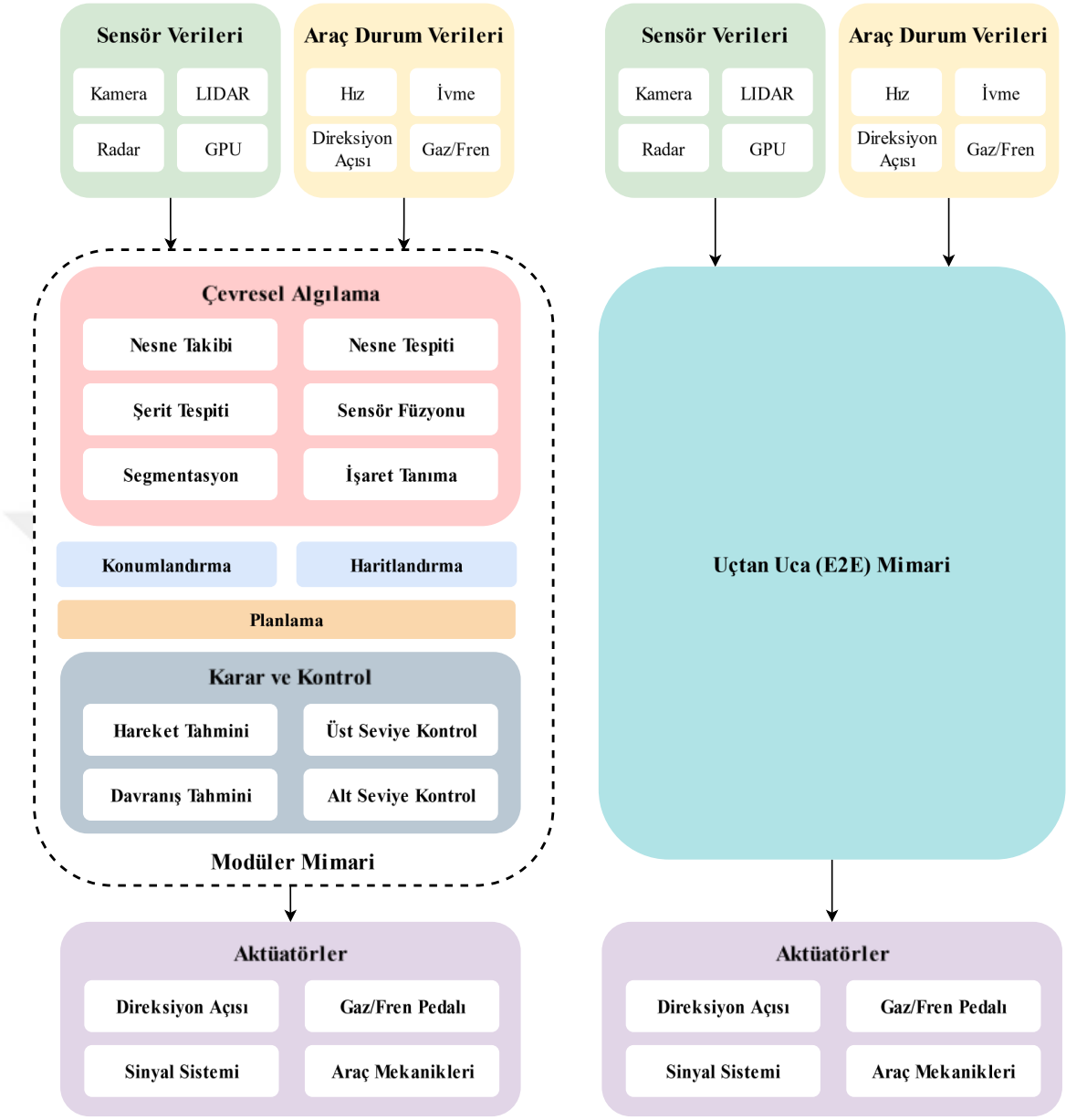
Günümüzde otonom araçlarda sensör füzyonu ileri seviyelerde gerçekleşmektedir ve bu birleşimin (sensör veri eşlemenin) yapılması için ileri füzyon teknikleri kullanılmaktadır. Bu tekniklerden bazıları; maksimum olabilirlik, kalman filtresi, parçacık filtresi, kovaryans kesişimi ve birleşimidir (Castanedo, 2013).

Bahsedilen sensörler dışında bir otonom araçta diğer otonom sistemlerle ya da akıllı şehir elemanları (trafik ışıkları ve benzeri öğeler) ile haberleşmek için bağlantı (connectivity) araçları da bulunabilmektedir. Otonom araçlarda sensörlerden elde edilen bilgiler doğrultusunda gerekli fonksiyonları gerçekleştirmek için bir bilgi işleme platformu bulunmaktadır (Gao vd., 2019). Bu platformlar aracın beyni olarak ifade edilen yüksek hesaplama gücüne sahip bilgisayarlardır. Bu platformlar araçlara monte edilmiş yerleşik bir bilgisayar ünitesi olabileceği gibi araç verileri ile iletişime geçebilen bulut tabanlı bilgisayarlar da olabilmektedir (S. Liu vd., 2019).

Otonom araçlarda ve benzeri robot sistemlerinde, bahsi geçen yazılımsal fonksiyonları gerçekleştirmek için kullanılan sistem mimarileri genellikle iki ana mimari olarak ayrılmaktadır (Yurtsever vd., 2020). Bunlar uçtan uca (end-to-end, E2E) ve modüler sistem mimarileridir (X. Wang vd., 2024). Otonom araçlarda uçtan uca mimariler olarak

adlandırılan yaklaşımda ham sensör verilerinin (bu sensör verileri kamera görüntüsü, LIDAR nokta bulutu gibi çeşitli sensör verileri olabilmektedir) tek bir modelden geçerek doğruca aracın kontrol bileşenlerine (bu kontrol bileşenleri direksiyon, gaz pedalı, sinyal sistemi gibi çeşitli aktüatörler olabilmektedir) etki etmesi hedeflenmektedir (Bojarski vd., 2016). Bu yaklaşımda girdiler ve çıktılar arasında doğrudan bir haritalama işlemi söz konusudur. Bu sistemde aracın hareket etmesi için gerekli olan tüm fonksiyonlar, tüm süreç, uçtan uca bağlı olan tek bir modelde öğrenilip optimize edilmektedir. Öte yandan modüler olarak adlandırılan karşıt mimarilerde öğrenim süreci için gerekli olan işlemler birbirinden ayrılarak modüllere bölünmektedir (Serban vd., 2018). Bu durumda her bir modül belirli bir fonksiyonu yerine getirmek ile görevlidir. Bu modüllerin birleştirilmesi ile istenilen mimari yapı oluşturulmaktadır. Burada bahsedilen uçtan uca ve modüler sistem mimarileri arasındaki yapısal fark Şekil 2.1'de görselleştirilmiştir. Uçtan uca mimarinin modüler mimariye göre bazı sınırları mevcuttur. Bu sınırlar Glasmachers (2017) tarafından ele alınmıştır. Bu bağlamda iki yazılım mimarisi karşılaştırmasını aşağıdaki gibi ele alınabilmektedir:

- Uçtan uca mimariler ara fonksiyonlar için bir müdahaleye ihtiyaç duymazken, modüler mimarilerde amaç ara fonksiyonları müdahale edilebilir bir hale getirmektir. Bu durumda modüler mimarilerde her bir modül için hata ayıklaması (debugging) kolaylaşmaktadır. Buna karşın uçtan uca mimariler "kara kutu" (blackbox) olarak değerlendirilmektedir.
- Modüler mimariler daha karmaşık bir tasarım ve entegrasyon süreci gerektirir. Buna karşın uçtan uca mimariler oldukça minimalist bir sistem tasarımı oluşturmaktadır. Bu nedenle özellikle akademik çalışmalarda uçtan uca mimariler iş yükünü azalttığı için daha fazla tercih edilebilmektedir.
- Uçtan uca mimarilerde etkili bir eğitim için büyük veri setleri ve uzun süreli eğitimler söz konusu olabilir. Buna karşın modüler mimarilerde istenilen modüller önceden eğitilerek eğitim süreci hızlandırılabilmektedir.

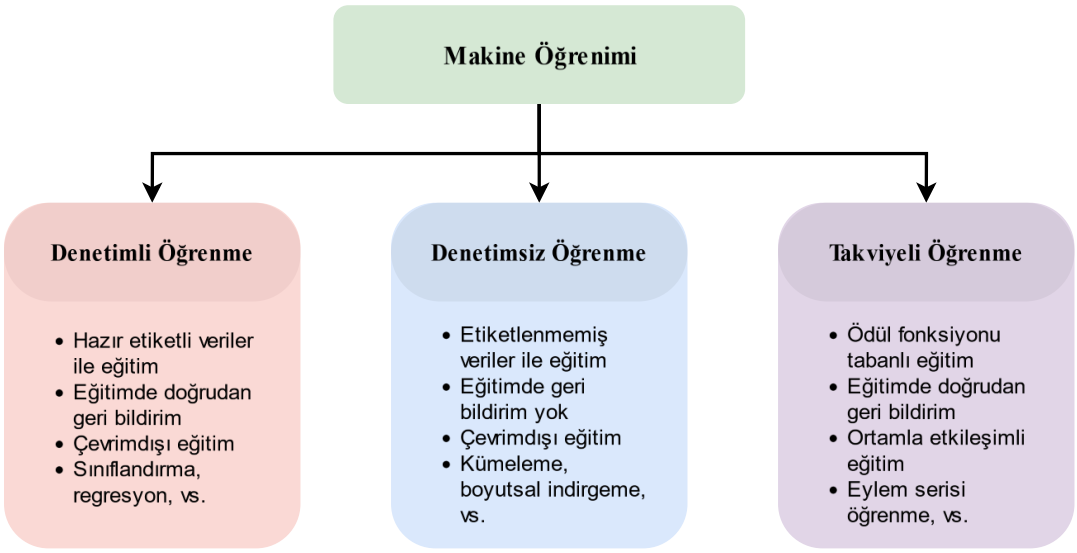



**Şekil 2.1.** Otonom sistemlerde kullanılan yazılım mimarilerinin temsili yapıları

Karmaşık görevleri daha verimli şekilde tamamlamak amacıyla endüstride modüler sistemler tercih edilmektedir (L. Liu vd., 2021). Google ve Uber gibi şirketlerin geliştirdiği otonom araçlar da modüler sistemlere dayanmaktadır. Bunlara dayanarak, bu tezin konu edindiği çalışmada modüler bir sistem yazılım mimarisi oluşturulmuş olup, oluşturulan mimarinin yapısı detaylı bir şekilde "Materyal ve Yöntem" bölümünde tanıtılmaktadır.

## 2.2. Takviyeli Öğrenme

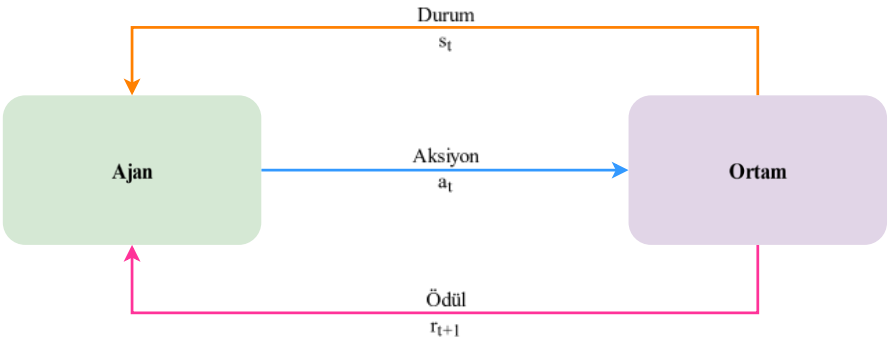
Geleneksel anlamda makine öğrenimi yaklaşımları; denetimli öğrenme (Supervised Learning, SL), denetimsiz öğrenme (Unsupervised Learning, USL) ve takviyeli öğrenme (Reinforcement Learning, RL) olmak üzere kabaca üç ana paradigmada sınıflandırılmaktadır (El Naqa vd., 2015). RL'nin makine öğrenimi içerisindeki yeri ve diğer makine öğrenimi paradigmaları olan denetimli ve denetimsiz öğrenme ile aralarındaki temel farklar Şekil 2.2'de gösterilmektedir. Bu doğrultuda makine öğrenimi içerisinde her üç paradigmanın kendine has yaklaşımları, avantajları ve kısıtları mevcuttur (Ethem, 2020). Denetimli öğrenmede eğitim veri seti her bir örnek için giriş verilerini ve karşılık gelen çıkış etiketlerini içerir. Denetimli öğrenme modeli, bu etiketli veri seti üzerinden eğitilir ve giriş verilerini çıkış değerlerine doğru haritalayan bir fonksiyon öğrenir. Denetimsiz öğrenme ise, etiketli veri eksik olduğunda veya hiç olmadığında ya da etiketli veriye ihtiyaç duyulmadığında kullanılır. Bu yöntemde modelin eğitildiği veri setindeki yapıları ve desenleri öğrenmesi beklenmektedir.




**Şekil 2.2.** Makine öğrenimi paradigmalarını ve temel özelliklerini gösteren diyagram

Bunlara karşın RL, öğrenme ve karar verme işlevlerine sahip bir ajanın (agent) bulunduğu çevre (environment) ile etkileşime girerek belirli bir dizi görevi optimal şekilde nasıl tamamlayacağını öğrenmeye odaklanan bir yapay zeka sistemi olarak tanımlanmaktadır (Sutton & Barto, 2018). RL'de ajan, çevresiyle etkileşimi boyunca

kazandığı deneyimlerden öğrenerek kendini optimize eder. Ajan öğrendiği bilgileri gelecekteki kararlarında kullanarak görevini tamamlamaya odaklanır. RL, davranışsal psikolojiye dayanan deneyim tabanlı bir yaklaşımdır. Bu sistem robotik, istatistik, bilgi teorisi, oyun teorisi ve benzeri alanlarca benimsenmiş disiplinler arası bir makine öğrenimi yaklaşımıdır.




**Şekil 2.3.** Takviyeli öğrenmenin temelini oluşturan Markov karar süreci modeli

RL'de öğrenim süreci bir Markov karar süreci (Markov Decision Process, MDP) olarak ifade edilir. Markov karar süreci bir durum dizisinin, durumlar arası geçiş olasılıkları ve her bir durumda alınabilecek eylemlerin o durum, eylem ve geçişe özel bir ödül ile bağıntı kurduğu ayrık zamanlı ve stokastik bir matematiksel bir modeldir (Van Otterlo & Wiering, 2012). Bu çalışmada ele alınan sistem Şekil 2.3'te gösterildiği üzere MDP çerçevesi üzerine inşa edilmiştir (Mausam & Kolobov, 2012). Bir Markov karar süreci olarak RL 4 elemanlı (*S, A, P, R*) bir dizi (tuple) olarak ifade edilmektedir. Bu elemanlara yönelik açıklamalara aşağıda yer verilmiştir.

- *S*; durum uzayı (state space), ajanın bulunabileceği tüm olası durumların kümesini temsil eder.
- *A*; aksiyon uzayı (action space), ajanın gerçekleştirebileceği aksiyonların kümesini temsil eder.
- $R(s, a, s')$; $a$ aksiyonu sonucu $s$ durumundan $s'$ durumuna geçişte elde edilen anlık ödül (reward) miktarını temsil eder.
- $P(s' \mid s, a)$; $s$ durumundayken $a$ aksiyonu gerçekleştirildiğinde $s'$ durumuna geçiş olasılığı temsil eder. Bunlarla birlikte uygulama düzeyinde eğitim bölümünün bitip bitmediğini ifade eden *d* değeri bulunur.

RL'de, ajan etkileşimde bulunduğu ortamda görevini tamamlayana ya da başarısız olana kadar aksiyonlar sergiler. Bu süreç bir bölüm (episode) olarak adlandırılmaktadır. Bir bölümde gerçekleştirilen her durum-aksiyon-ödül-durum döngüsü adım (step) olarak tanımlanmaktadır. Bu süreçte ajan maksimum ödüle ulaşmayı, yani beklenen kümülatif ödülü en üst düzeye çıkarmayı amaçlar. RL'de bir ödül denklemi (Denklem 2.1) ajan ile ortam arasındaki ilişkinin nasıl tanımlandığını ifade etmektedir (Sewak, 2019). Ödül denklemi, ajanın $t$ zamanında bir $s_t$ durumundan $s_{t+1}$ durumuna $a_t$ aksiyonu ile geçişinden aldığı ödülü, başka bir deyişle geri bildirimi temsil eder ($R$: $S \times A \times S \rightarrow \mathbb{R}$). Ajanın uzun vadede aldığı ödüllerin toplamı olan kümülatif ödül $G_t$ ise Denklem 2.2'deki şekilde ifade edilmektedir (Sutton & Barto, 2005). Burada $k$, bölüm içerisindeki adım numarası; $t$, zaman adımı; $\gamma$, gelecekteki ödüllerin önemini belirleyen indirim katsayısıdır (discount factor, $\gamma \in [0,1]$). RL, $\gamma=0$ olması durumunda ajan miyop olarak değerlendirilir ve buna bağlı olarak yalnızca anlık ödülü maksimize etmekle ilgilenir.

$$R_{t+1} = R(s_t, a_t, s_{t+1}) \tag{2.1}$$

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{2.2}$$

RL'de ajanın kümülatif ödül miktarını maksimize etme hedefi ile herhangi bir durumda hangi aksiyonu seçmesi gerektiğini belirleyen bir strateji geliştirmesi beklenmektedir. Bu durum-aksiyon eşlemesi politika (policy, $\pi$) olarak adlandırılmaktadır. Bu doğrultuda bir politika denklemi (Denklem 2.3) stokastik olarak ajanın herhangi bir $s$ durumundayken $a$ aksiyonunu seçme olasılığını temsil eder ($\pi: S \times A \rightarrow [0,1]$) (Sutton & Barto, 2005). Deterministik bir politika ise $\pi(s) = a$ olarak tanımlanır.

$$\pi(a|s) = P\,(a|s) \tag{2.3}$$

RL'de birçok algoritma bir politikanın ne kadar değerli olduğunu ya da maksimum kümülatif ödül miktarını veren optimal politikayı öğrenmek adına değer fonksiyonlarından yararlanmaktadır. Bu fonksiyonlardan biri Denklem 2.4'te

gösterildiği şekilde durum değeri (value function, V-function) denklemidir (Sutton & Barto, 2005). Verilen herhangi bir $s$ durumunun değeri o $s$ durumundan başlanmak üzere, belli bir $\pi$ politikası altında elde edilebilecek olan toplam indirgenmiş ödül olarak tanımlanır.

$$V^{\pi}(s) = \mathbb{E}[G_t | s_t = s, \pi] = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid s_t = s, \pi\right] \tag{2.4}$$

RL'de başka bir değer fonksiyonu olarak bir durum ve aksiyonu ikilisi için beklenen kümülatif ödülü temsil eden kalite değeri fonksiyonu (quality function, Q-function) (Denklem 2.5) mevcuttur (Sutton & Barto, 2005). Kalite değeri fonksiyonu durum değeri denkleminden farklı olarak verilen bir $s$ durumunda $a$ aksiyonunu gerçekleştirmenin getirisini (indirgenmiş kümülatif ödülü) belirtmektedir. Optimal politikayı bulmada kullanılan kalite denklemi birçok algoritmanın teorik temelini oluşturmaktadır (Van Otterlo & Wiering, 2012).

$$Q^{\pi}(s, a) = \mathbb{E}[G_t | s_t = s, a_t = a, \pi] = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid s_t = s, a_t = a, \pi\right] \tag{2.5}$$

RL ajanları, durum değeri ve kalite değeri hesaplamalarını 1950'lerin ortasında geliştirilmiş olan Bellman denklemlerine dayandırmaktadır (Sutton & Barto, 2005). Bu denklemler ajanın anlık kazandığı ödül ile gelecekte kazanılması beklenen toplam ödül arasında matematiksel bir bağlantı kurmasını sağlar. Bellman denklemleri dinamik programlamanın prensiplerini kullanarak özyinelemeli bir şekilde değer hesaplamalarını gerçekleştirir. Bu doğrultuda, Denklem 2.6'da verilen herhangi bir $s$ durumunun π politikası altındaki değerine ilişkin Bellman denklemi ifade edilmektedir (Sutton & Barto, 2005). Burada $\pi(a|s)$, $s$ durumuna $a$ aksiyonu için seçilme olasılığı; $P(s'|s, a)$, bir $s$ durumundayken $a$ aksiyonu ile $s'$ durumuna geçme olasılığı; $r(s, a, s')$, bir $s$ durumundan $s'$ durumuna $a$ aksiyonu ile geçişinden aldığı ödülü; $\gamma$, gelecek ödüllerinin önemini belirleyen indirim katsayısı; $V^{\pi}(s')$, $s'$ durumunun $\pi$ politikası altındaki değeri olarak tanımlanmaktadır.

$$V^{\pi}(s) = \sum_{a} \pi(a|s) \sum_{s\prime} P(s'|s,a)\,[R(s,a,s') + \gamma V^{\pi}(s')] \tag{2.6}$$

Benzer şekilde Denklem 2.7 herhangi bir $s$ durumunda $a$ aksiyonunu gerçekleştirmenin kalite değerine ilişkin Bellman denklemi verilmiştir (Sutton & Barto, 2005). Burada $P(s'|s,a)$, bir *s* durumundayken *a* aksiyonu ile $s'$ durumuna geçme olasılığı; $r(s,a,s')$, bir *s* durumundan $s'$ durumuna *a* aksiyonu ile geçişinden aldığı ödülü; $\gamma$, gelecek ödüllerinin önemini belirleyen indirim katsayısı; $\pi(a'|s')$, $s'$ durumuna $a'$ aksiyonu için seçilme olasılığı; $Q^{\pi}(s',a')$, $s'$ durumuna $a'$ eylemini gerçekleştirmenin değerini ifade etmektedir.

$$Q^{\pi}(s,a) = \sum_{s\prime} P(s'|s,a) \left[ R(s,a,s') + \gamma \sum_{a\prime} \pi(a'|s')\, Q^{\pi}(s',a') \right] \tag{2.7}$$

Bu durumda eğer $V^{\pi}(s')$ ifadesi biliniyorsa durum $s$ ve aksiyon $a$ ikilisi için kalite değeri Bellman prensiplerine göre Denklem 2.8'deki gibi hesaplanmaktadır (Sutton & Barto, 2005).

$$Q^{\pi}(s,a) = \sum_{s\prime} P(s'|s,a)\,[R(s,a,s') + \gamma V^{\pi}(s')] \tag{2.8}$$

Bir RL ajanının en temel amacı optimal politikaya ($\pi^{*}$) ulaşmaktır (Meyn, 2024) (Sutton & Barto, 2018). Bunun için optimal durum ya da kalite değerlerinden yardım alınabilir. Bir durum $s$ ve aksiyon $a$ ikilisi için optimal kalite değeri (Denklem 2.9), $Q^{*}(s,a)$, Bellman optimallik denklemi ile yinelemeli olarak hesaplanabilmektedir (Sutton & Barto, 2005). Burada $Q^{*}(s,a)$ ajanın en iyi politikayı kullanması durumunda beklenen toplam ödülü temsil ederken $r(s,a,s')$ ajanın *s* durumunda *a* aksiyonunu gerçekleştirmesi ile aldığı anlık ödülü belirtir. $P(s'|s,a)$ ifadesi ajanın *s* durumunda *a* aksiyonunu gerçekleştirmesi ile $s'$ duruma geçme olasılığını, $\max_{a\prime} Q^{*}(s',a')$ ise ajanın sonraki durum olan $s'$ için maksimum beklenen ödül değerini veren $a'$ aksiyonu seçtiğindeki kalite değerini belirtmektedir.

$$Q^*(s,a) = \sum_{s\prime} P(s'|s,a)\left[R(s,a,s') + \gamma \max_{a'} Q^*(s',a')\right] \quad (2.9)$$

Burada optimal kalite değer denklemi optimal bir sonraki durum değeri, $V^*(s')$, kullanılarak (Denklem 2.10) ifade edilebilmektedir (Sutton & Barto, 2005).

$$Q^*(s,a) = \sum_{s\prime} P(s'|s,a)\,[R(s,a,s') + \gamma\, V^*(s')] \quad (2.10)$$

Bu bilgiler doğrultusunda kalite fonksiyonuna dayalı optimal politika denkleminde (Denklem 2.11) $Q^*(s,a)$ bir $s$ durumu ve $a$ aksiyonu ikilisi için en iyi beklenen toplam ödülü temsil ederken $arg\max_a$ en büyük Q değerini veren aksiyonu seçen operatör olarak tanımlanabilir (Sutton & Barto, 2005).

$$\pi^*(s) = arg\max_a Q^*(s,a) \quad (2.11)$$

Van Otterlo & Wiering (2012) optimal politikaya ilişkin olarak en iyi eylemi "o eylemin gerçekleştirilmesiyle ortaya çıkan olası sonraki durumlara dayalı olarak en yüksek beklenen faydaya sahip eylemdir." şeklinde ifade etmektedir.

### 2.2.1. Derin takviyeli öğrenme

Derin takviyeli öğrenme (Deep Reinforcement Learning, DRL), klasik takviyeli öğrenme prensiplerini kullanarak karmaşık görevleri çözmek ve öğrenme sürecini optimize etmek için derin öğrenme tekniklerini bütünleştiren bir yaklaşımdır (Sewak, 2019).

DRL, genellikle büyük ve karmaşık verileri işleyebilen derin sinir ağlarını kullanarak öğrenme sürecini iyileştirir. Bu yaklaşım özellikle yüksek boyutlu girdilere veya büyük aksiyon uzaylarına sahip problemlerde etkili olabilmektedir. DRL, derin öğrenme mimarilerinin entegre edilmesi sayesinde geleneksel RL yöntemlerinin çözmekte zorlandığı veya sınırlı kaldığı bir dizi soruna çözüm getirmekte ve bazı avantajlar

sunmaktadır (Wani vd., 2019). Bu doğrultuda DRL'nin geleneksel RL'ye kıyasla avantaj sağladığı bazı durumlar aşağıda verilmiştir:

- **Yüksek boyutlu veriler ile uyumluluk**: Geleneksel anlamda RL yöntemleri, yüksek boyutlu girdi verilerinin kullanımı için özellik mühendisliğine (feature engineering) ihtiyaç duymaktadır. DRL, derin sinir ağlarını kullanarak büyük ve karmaşık durum uzayları üzerinde doğrudan çalışabilmektedir. Bu duruma benzer olarak DRL, yüksek boyutlu aksiyon uzaylarına sahip problemlere rahatça uyum sağlayabilmektedir.
- **Model genelleme:** Geleneksel anlamda RL yöntemleri, öğrenilen durumlar dışında kalan durum uzayında genelleme yapmakta zorlanmaktadır. DRL yapay sinir ağlarının avantajlarından yararlanarak genelleme sorunu ile daha iyi başa çıkabilmektedir.
- **Gizlenmiş ve karmaşık olguları fark edebilme:** DRL, yapay sinir ağları sayesinde örtük ve karmaşık yapıları öğrenmeye yatkın durumdadır. Bu durum daha gelişmiş stratejilerin öğrenilmesi ve örüntülerin tanınmasını sağlamaktadır.

DRL yaklaşımları 2010'lu yılların başından bu yana üzerinde yoğun bir şekilde çalışılan bir alan olmaya devam etmektedir. Özellikle 2015 yılında (Mnih vd., 2015) DeepMind ekibi tarafından geliştirilen Derin Q-Ağı (Deep Q-Networks, DQN) algoritması makine öğrenimi alanında önemli dönüşümlere yol açmıştır. Gerçekleştirilen çalışmanın yayınlanan kapsamlı raporunda DQN algoritması, seçilmiş bir dizi Atari oyunlarında ekran görüntülerini ve önceki deneyimlerini kullanarak insan düzeyinde bir başarı elde etmiştir. DQN algoritmasının sürekli ve büyük durum uzaylarındaki bu başarısı DRL yöntemleri için önemli bir kapı aralamıştır. Kendisinden sonra gelen daha gelişmiş yaklaşımlar için temel görevi görmüş ve ilham kaynağı olmuştur.

DQN algoritması bir klasik RL yaklaşımı olan Q-Öğrenme (Q-Learning) algoritmasını temel alır ve bu algoritma ile ölçeklenemeyen problemlerin çözümüne olanak tanır. Başlangıçta kusurları bulunan DQN algoritması daha sonra geliştirilen versiyonu ile literatürde Çift Derin Q-Ağı (Double Deep Q-Network, DDQN) olarak da adlandırılabilmektedir.

DQN (ya da DDQN) algoritmasında ana ağ ve hedef ağ olmak üzere iki adet sinir ağı bulunur. DQN algoritmasında ana ağ ile durum ve aksiyon ikililerinin kalite değerlerinin tahmin edilmesi sağlanır. Hedef ağ yardımı ile de ana ağın güncellenmesi için gerekli hedef kalite değerleri hesaplanır. Bu algoritmada ana ağ $\varphi$ parametrelerine sahiptir. Ana ağın güncellenmesi işleminde tahmin edilen kalite değerleri ile hesaplanmış hedef kalite değeri arasındaki fark kayıp denklemi (Denklem 2.12) olarak kullanılmaktadır (Mnih vd., 2015). Algoritmanın amacı buradaki farkı minimize etmektir. Burada $Q_{\varphi}(s,a)$ durum $s$ ve aksiyon $a$ için ana ağ tarafından tahmin edilen kalite değerini; y ise gerçek kalite değerini ifade etmektedir.

$$L(\varphi) = \mathbb{E}\left[\left(y - Q_{\varphi}(s,a)\right)^2\right] \tag{2.12}$$

Burada y ile ifade edilen gerçek kalite değerinin (Denklem 2.13) hesaplanması Bellman denklemlerine uygun olarak gerçekleştirilmektedir (Mnih vd., 2015). Bu anlayışla, $r$ anlık elde edilen ödülü; $Q_{\bar{\varphi}}(s',a')$ ise parametreleri $\bar{\varphi}$ olarak ifade edilen hedef ağın tahmin ettiği gelecekteki ödülü ifade eder. Burada $\max_{a'}$ en iyi kalite değerini seçmek için kullanılan operatör iken $\gamma$ gelecekteki ödüllerin önemini belirten indirim faktörüdür. Bu yaklaşımda hedef kalite değerlerinin hesabı parametreleri sabit tutulan bir hedef ağında gerçekleşmektedir. Belirli her birkaç adımda bir ana ağın parametreleri ile bu hedef ağı güncellenir.

$$y = R(s,a,s') + \gamma \max_{a'} Q_{\bar{\varphi}}(s',a') \tag{2.13}$$

DRL alanında elde edilen ilk başarılı sonuçların ardından, farklı DRL yaklaşımlarının geliştirilmesi uzun sürmemiştir (Arulkumaran vd., 2017). DQN algoritmasında olduğu gibi çeşitli değer denklemleri kullanarak öğrenen yaklaşımların yanı sıra politikayı ağırlıklandırarak durumlardan aksiyon haritalama yapabilen çeşitli yaklaşımlar da geliştirilmiştir. Bu gelişim sürecinde öne çıkan farklı yaklaşımlar DRL'nin robotik sistemlerde devrim yaratacak bir niteliğe sahip olduğuna dikkat çekmiştir. Bu doğrultuda mevcut araştırma alanları genişletilmiş ve DRL'nin zorlukları incelenmiştir.

Günümüzde belli problemlere çözüm olabilmek adına DRL yaklaşımlarının çeşitliliği artmaya devam etmektedir.

### 2.2.2. Takviyeli öğrenme yaklaşımları içinde sınıflandırmalar

RL içerisindeki yaklaşımların sınıflandırılması kullanılan öğrenme denklemleri, ortam özellikleri ve hesaplama stratejileri başta olmak üzere birden fazla kritere dayanmaktadır. Bu durum literatürde çeşitli sınıflandırma diyagramlarının oluşmasına neden olmaktadır. Bu nedenle RL içerisindeki sınıflandırmalar tek bir ağaç yapısı ile ifade edilemez. Bu sınıflandırmalar aynı zamanda RL algoritmalarının işleyişi için tanımlayıcı kavramlar niteliğindedir. RL algoritmalarının sınıflandırıldığı temel konseptler aşağıda açıklanmıştır.

**a) Politika tabanlı ve değer tabanlı yaklaşımlar**

Buradaki sınıflandırma RL'nin en temel ayrımlarından birini oluşturmaktadır (Nachum vd., 2017). Politika tabanlı yaklaşımlarda, bir RL ajanı, durum değerlerine ihtiyaç duymadan doğrudan olarak bir politika öğrenir. Bu yaklaşımda ajan eğitim süreci boyunca $s$ durumlarını $a$ aksiyonlarına haritalandıran ($\pi : s \rightarrow a$) bir $\pi$ politikasının parametrelerini doğrudan günceller. Bu durum boyutsal olarak büyük ve sürekli aksiyon uzaylarında avantaj sağlamakta ve stokastik politika öğrenimini kolaylaştırmaktadır. Bu avantajlara karşın politika tabanlı yaklaşım yüksek varyansa neden olabildiği için öğrenme sürecini uzatabilmektedir ve daha fazla veriye ihtiyaç duymaktadır.

Değer tabanlı yaklaşımlarda ise, bir RL ajanı, durum değerini ya da kalite değerini öğrenerek bu bilgiler doğrultusunda dolaylı bir şekilde optimal politikayı oluşturur. Bu yaklaşımda optimal politikaya ulaşmak için en iyi değere sahip aksiyonu seçmesi beklenmektedir. Politika tabanlı yaklaşımlara göre yapısal olarak daha basittir. Durum değerini öğrenmek ayrık aksiyon uzaylarında daha verimli olsa da sürekli aksiyon uzayları için uyarlanabilirliği zordur.

**b) Model tabanlı ve modelden bağımsız yaklaşımlar**

RL'de bir ortamın modelinde kasıt durum geçişlerinin ve bu geçişlerden elde edilen ödüllerin bilinmesidir (Spinning Up, 2018). RL algoritmaları, RL ajanının ortam modeline sahip olup olmadığına bağlı olarak model tabanlı ve modelden bağımsız olarak sınıflandırılabilir (Gläscher vd., 2010).

Model tabanlı yaklaşımlarda, RL ajanı ortamın (çözülmesi gereken problemin) modelinden haberdardır. Bu durum ajanın olası seçenekleri önceden değerlendirmesine ve bu değerlendirme sonuçlarını öğrenilmiş bir politikaya uyarlamasına olanak sağlamaktadır. Bu şekilde model tabanlı yöntemler elde bulunan örnekleri verimli (sample efficiency) kullanmaktadır. Go oyunu dünya şampiyonunu yenen *AlphaGo* RL algoritması model tabanlı bir yaklaşımdır.

Modelden bağımsız yaklaşımlarda ise ajan ortamın bir modelini bilmez ve modele ihtiyaç duymaz. Ajan bir politika ya da durum değerini öğrenmeye çalışır. Bunun için ortamla etkileşime girerek deneyim elde eder. Model tabanlı yaklaşımlara kıyasla örnek kullanımı konusunda daha verimsizdir. Örnek verimsizliğine karşın uygulanması model tabanlı yöntemlere göre daha kolaydır. Bu yaklaşım, ortam modelinin karmaşık veya bilinmediği durumlarda tercih edilir.

Gerçek dünya problemlerinin büyük bir kısmında ortam modeli oluşturulamaz. Bu durumda ajan ortam modelini bilme imkanına sahip değildir. Otonom sürüş senaryoları için ortamın modeli bilinmemektedir. Ortamın büyüklüğü nedeni ile bir modelinin oluşturulması mümkün değildir. SAC algoritması modelden bağımsız bir algoritmadır.

**c) Politika dahilinde ve politika dışında yaklaşımlar**

Politika dahilinde (on-policy) ve politika dışındaki (off-policy) yöntemler arasındaki fark kalite değerleri güncellenirken kullanılan politikalardır. Öğrenme sürecinde rastgele bir politika izleyen RL ajanı bir $s$ durumunda $s'$ durumuna aksiyon $a$ ile geçerken durum $s$ ve aksiyon $a$ için kalite değeri $Q(s,a)$ güncellenir. Bu güncelleme $s'$ durumu için önceki politikalardan elde edilen bilgiler doğrultusunda

maksimum kalite değerini veren $a'$ aksiyonu seçilerek yapılması durumunda politika dışı bir yaklaşım olarak değerlendirilmektedir.

Bu güncelleme o an kullanılan politikanın karar verdiği a' aksiyonu ile elde edilen kalite değeri ile güncelleniyor ise politika dahilinde bir yaklaşım olduğu anlamına gelmektedir. SAC algoritması politika dışında bir algoritmadır.

**d) Ayrık aksiyon uzayı ve sürekli aksiyon uzayı**

Aksiyon uzayı bir RL ajanın, sistem tarafından izin verilen, gerçekleştirebileceği tüm aksiyonların kümesi olarak tanımlanır. Aksiyon uzayı, problemlerin türüne göre, ayrık (discrete) ve sürekli (continuous) olarak ikiye ayrılmaktadır (Almahamid & Grolinger, 2021). Eleman sayısı *n* olan ayrık aksiyon uzayı (Denklem 2.14), sonlu sayılı aksiyon kümesini ifade etmektedir.

$$A = \{a_1, a_2, \dots, a_n\} \tag{2.14}$$

Buna karşın sürekli aksiyon uzayları (Denklem 2.15), tanımlanmış bir aralıkta sonsuz değer alabilen aksiyonların kümesini ifade etmektedir. Burada *n* değişkeni aksiyon türü sayısının, $\mathbb{R}$ ise gerçek sayılar kümesini ifade etmektedir.

$$A \in \mathbb{R}^n \tag{2.15}$$

Sürekli aksiyon uzaylarının ayrıklaştırılarak dönüşümü ve ayrık aksiyon uzayları benzetiminin yapılması mümkündür (Tang & Agrawal, 2020). Bu benzetim ortam probleminin yapısına göre avantaj ya da dezavantaj olabilmektedir. Bir aktör-kritik yaklaşımı olan SAC algoritması sürekli aksiyon uzaylarında etkilidir.

### 2.2.3. Keşif ve mevcudu kullanma ikilemi

Keşif (exploration) ve mevcudu kullanma (exploitation) ikilemi RL içerisinde algoritmaların öğrenim performansını doğrudan etkileyen temel bir konsept olarak bulunmaktadır (Berger-Tal vd., 2014).

Mevcudu kullanma, ajanın halihazırda elinde bulunan (öğrenmiş olduğu) bilgiyi kullanarak en iyi ödülü verecek aksiyonları seçmesini ifade etmektedir. Öte yandan keşif kavramı, ajanın daha önce keşfedilmemiş aksiyonları ödül bazında daha iyi sonuçlar verme ihtimaline dayanarak gerçekleştirmesini ifade etmektedir.

Keşif ile ajan çevre hakkında daha fazla bilgi elde edebilir. RL'de bu iki olgu arasındaki dengenin oluşturulması optimal politikayı bulmak konusunda önemli bir sorundur. Bu denge sorununda ajanın keşfe yatkınlığı optimal politikaya geç ulaşmasına, ya da ulaşamadan yerel optimumda kalmasına yol açabilmektedir. Buna karşın, mevcudu kullanmaya meyilli bir ajan da benzer sorunlarla karşılaşıp daha iyi aksiyonları bulma konusunda başarısız olabilir.

Keşif ve mevcudu kullanma dengesini sağlamak için RL algoritmaları farklı yaklaşımları benimsemiştir. Bu durum için en sık kullanılan temel bileşenlerden biri aksiyon gürültüsüdür (Hollenstein vd., 2022a). Özellikle sürekli aksiyon uzaylarında önemli bir rol oynayan aksiyon gürültüsü, ajanı keşfe teşvik ederek yerel optimumda sıkışmasını önleyebilmekte ve politikanın sağlamlığını artırmaktadır.

RL'de normal dağılımlı rastgele bir aksiyon gürültüsü olarak Gauss gürültüsü (Gaussian noise), zaman korelasyonlu bir aksiyon gürültüsü olarak Ornstein–Uhlenbeck gürültüsü ya da bir aralıktan rastgele tekdüze bir şekilde örneklenen bir aksiyon gürültüsü başta olmak üzere birçok farklı gürültü türü kullanılabilmektedir. Bu gürültüler bazı senaryolarda yalnızca aksiyonlar için değil gözlemlenen durumlar ya da ağ parametreleri için de uygulanabilir (Plappert vd., 2018).

Gürültüler dışında keşif ve mevcudu kullanma dengesi için en sık kullanılan stratejilerden biri "Epsilon-Greedy" stratejisidir (Carton, 2021). Bu stratejide ajan bir rastgele örneklene bir $\epsilon$ olasılığı ile keşif amaçlı rastgele bir aksiyonu ve 1- $\epsilon$ olasılığı ile mevcut bilgiyi kullanarak en iyi aksiyonu gerçekleştirmektedir. Keşif olasılığı $\epsilon$ değerinin sınırları zamanla azalmaktadır. Azalan olasılık değişkeni ajanın zamanla keşiften mevcudu kullanmaya geçmesine olanak sağlar. Bu strateji ayrık zamanlı aksiyon uzaylarında kullanmak için daha uygundur (Ladosz vd., 2022). Sürekli zamanlı

eylem alanları için genellikle Gauss dağılımı kullanılarak rastgele bir aksiyon örneklemesi yapılmaktadır. Bununla beraber kayıp fonksiyonuna bir entropi değeri eklenerek ajanın takip ettiği politikada daha çeşitli aksiyonlar gerçekleştirmesi sağlanabilir.

### 2.3. Otonom Araçlar Literatüründe Takviyeli Öğrenme

Denetimli derin öğrenme yöntemleri uzun yıllardır otonom araçlarda birçok görev için kullanılsa da (Kuutti vd., 2021) DRL 2010'lu yıllardan itibaren geliştirilmiş ve hızla robotik sistemlere uyarlanarak özellikle karar verme ve kontrol görevlerinde büyük başarılar elde etmiştir (Kiran vd., 2022).

DRL otonom araçlarda şerit takibi, şerit değiştirme, hareket planlama, engelden kaçma, trafik akış kontrolü gibi birçok alanda üzerinde çalışılan popüler bir konu olma niteliğini sürdürmektedir. DRL, otonom sürüş alanında hem uçtan uca hem de modüler öğrenme mimarilerine başarıyla uyarlanmış ve bu sayede çeşitli önemli sonuçlar elde edilmiştir (Elallid vd., 2022). Mevcut araştırmalar yalnızca bu temel sürüş görevlerine odaklanmakla kalmayıp, aynı zamanda otonom araçlar için daha konforlu ve insan benzeri sürüş davranışları geliştirmeyi de hedeflemektedir (Emuna vd., 2020). Bu bölümde, ilgili literatür incelenerek mevcut tez çalışmasının alana olan özgün katkısı ve literatürdeki yeri ortaya konulacaktır.

Bu bağlamda Çizelge 2.2, otonom araçlarda takviyeli öğrenme kullanılarak yapılan temel ve insan benzeri sürüş görevlerine dair bazı önemli çalışmalar sunulmaktadır.

**Çizelge 2.2.** Otonom sürüş literatüründeki RL çalışmaları

| Çalışma | Amaç | Yöntem (Ortam) | Katkı (Girdi) |
|---|---|---|---|
| Bojarski vd. (2016) | Şerit takibi (E2E) | Aktör-kritik (TORCS) | DRL'nin fizibilitesini kanıtladı (Kamera). |
| Nosrati vd. (2018) | Engelden kaçınma | Hiyerarşik DRL (Özel Sim.) | Hiyerarşik yapı ile öğrenmeyi hızlandırdı (Kinematik). |
| Wang, P. vd. (2018) | Pürüzsüz şerit değiştirme | Q-Öğrenme (Özel Sim.) | Konfor odaklı ödül fonksiyonu (Kinematik). |
| Isele vd. (2018) | Kavşaklarda gezinme | DQN (Sumo) | Kural tabanlı yöntemlerden üstün (Izgara Haritası). |
| Zhang, Y. vd. (2018) | Doğal hız kontrolü | DDQN (NGSIM Sim.) | ACC'den daha doğal hız profili (Gerçek Veri). |
| Huang, Z. vd. (2019) | Konforlu hız kontrolü | Aktör-kritik (Gerçek Araç) | Gerçek araçta hassasiyet/konfor dengesi (Hız/İvme). |
| Li, D. vd. (2019) | Yanal kontrol | DDPG (TORCS) | Model tabanlı yöntemlerden üstün (Kamera/Sensör). |
| Kendall vd. (2019) | Simülasyonsuz hızlı öğrenme | DDPG (Gerçek Araç) | <20 dk sürüşle öğrendi (Kamera). |
| Lu vd. (2019) | Kişiselleştirilmiş hız kontrolü | NRL (PreScan/Matlab) | Sürücü tarzını çevrimiçi öğrendi (İnsan Girdisi). |
| Liao vd. (2020) | Sollama kararı | Hiyerarşik DDQN (Özel Sim.) | Hiyerarşik karar stratejisi (Göreceli Veri). |
| Peng vd. (2021) | Yarışta en hızlı tur | DDDQN (TORCS) | İnsan sürücülerden daha hızlı (Kamera/Kinematik). |
| Andrade vd. (2022) | Yol çalışmasında sürüş | DDPG + YOLO (CARLA) | Algı ve kontrolü birleştirdi (İşaret/Sensör). |
| Pérez-Gil vd. (2022) | Algoritma karşılaştırma | DDPG & DQN (CARLA) | DDPG'nin üstünlüğünü gösterdi (İşlenmiş Görüntü). |
| Wang, W. vd. (2024) | Pürüzsüz politika öğrenimi | Smonet (Çeşitli Sim.) | Dalgalanmaları azaltan ağ mimarisi (Eksik Bilgi). |

2016 yılında TORCS yarış simülatöründe bir aktör-kritik DRL algoritması eğitilerek bir yarış aracının yarış parkurunu başarı ile tamamlaması sağlanmıştır (Bojarski vd., 2016).

Gerçekleştirilen çalışmada kamera görüntülerinden aracın direksiyon açısı ve gaz pedalı girdilerine haritalandırma yapan uçtan uca bir eğitim sistemi uygulanmıştır. Böylece DRL'nin otonom araçlar üzerinde uygulanmasına kademeli olarak başlanmıştır.

Nosrati vd. (2018) DQN algoritmasını çoklu şerit senaryosuna sahip bir simülasyonda eğiterek engelden kaçınma konusunda başarı sağlamıştır. Wang ve ekibi (P. Wang vd., 2018) tarafından gerçekleştirilen bir çalışmada ise trafik akış senaryosunda DRL 'nin şerit değiştirme görevindeki başarısı belgelenmiştir. Isele ve ekibi (Isele vd., 2018) DQN algoritmasını kullanarak simülatörde bir aracın kavşağa giriş ve çıkışını kontrol etmeyi sağlamış ve DRL'nin üst seviye kontrollerdeki başarısını kanıtlamıştır. Çin'de gerçekleştirilen bir çalışmada (Y. Zhang vd., 2018) insan sürücülerin gerçek dünya sürüş davranışlarını yansıtan verilerin yardımı ile bir DQN algoritması eğitilmiş ve bu sayede doğal bir hız kontrolü sağlanmıştır.

Z. Huang vd. (2019) gerçek bir aracın boylamsal (hız) kontrolünü sağlamak amacıyla bir aktör-kritik DRL algoritması eğitmiştir. Bu çalışmada algoritmayı; aracın mevcut hızı, ivmesi ve hedeflenen değerleri girdi olarak kullanmış ve hedeflere başarıyla ulaşmışlardır. Li vd. (2019) TORCS simülatöründe sürekli aksiyon uzaylarına uygun bir DRL yöntemi olan Derin Deterministik Politika Gradyanı (Deep Deterministic Policy Gradient, DDPG) algoritmasını araç kamera görüntüleri ve çeşitli sensörler ile eğiterek direksiyon açısını kontrol etmiş ve bu yaklaşımın model tabanlı kontrol yöntemlerinden daha üstün bir performans sergilediğini ortaya koymuştur.

DRL'nin dikkat çekici uygulamalarından biri de gerçek dünya ortamında gerçekleştirilmiştir (Kendall vd., 2019). 2019 yılında yayınlanan bu çalışmada Kendall ve ekibi otonom bir aracın yaklaşık 250 metrelik bir yolu araç kamera görüntülerini kullanan DDPG algoritması kontrolünde tamamlamasını sağlamıştır. Bu çalışma, DRL'nin gerçek dünya senaryolarına uyarlanabilirliğini göstermesi açısından önemli bir adımdır.

Lu vd. gerçekleştirmiş olduğu bir çalışmada (2019) otonom bir araç sinirsel takviyeli öğrenme (Neural Reinforcement Learning, NRL) ile çevrimiçi olarak insan

sürücülerden öğrenerek kişiselleştirilmiş insan benzeri sürüş davranışları sergilemiş ve etkili bir hız kontrolü sağlamıştır.

Liao vd. (2020) tarafından gerçekleştirilen çalışmada hiyerarşik bir sistem kullanarak hem üst seviye hem de alt seviye kontrolü için DQN kullanmış ve simülasyon ortamında çeşitli karayolu sürüş görevlerini başarıyla tamamlamıştır.

2021 yılında gerçekleştirilen bir çalışmada (Peng vd., 2021a), araç kamera görüntüsü ile araç hareket bilgileri de kullanılarak eğitilen Çift Derin Düello Q-ağı (Deep Double Duelling Q-Network, DDDQN) algoritmasının TORCS simülatöründe insan sürücülerden daha iyi bir performans sergilediği rapor edilmiştir. Geliştirilen çalışmada TORCS simülatöründe araç kamera görüntüleri ve hız verileri ile beslenen DRL algoritması sayesinde araç yarış parkurunu başarıyla tamamlamıştır (Peng vd., 2021b).

2022 yılında kamu yol işlerinde kullanılmak için gerçekleştirilen bir otonom araç çalışmasında (Andrade vd., 2022) YOLO algoritması ile yol işareti tespiti yapılmış ve DDPG algoritması ile araç kontrolü sağlanmıştır. Gelişmiş bir otonom sürüş platformu olan CARLA simülatörü kullanılarak gerçekleştirilen bir çalışmada (Pérez-Gil vd., 2022) kamera görüntülerinin farklı şekillerde işlenmesiyle elde edilen çeşitli şerit bilgileri ile beslenmiş DDPG algoritmaları eğitilerek bir binanın çevresini dolaşması sağlanmış ve bu algoritmalara ilişkin performans analizi gerçekleştirilmiştir.

Bu çalışmaların yanı sıra W. Wang vd. (2024) tarafından gizli katmanlar sayesinde aksiyonlardaki dalgalanmaları azaltarak politikanın pürüzsüzlüğünü sağlayan yeni bir sinir ağı mimarisi olan Smonet tanıtılmıştır. Otonom araçları geleceğe hazırlamak amacıyla daha doğal ve insan benzeri sürüş deneyimi kazandırmak amacıyla akademik araştırmalar gerçekleştirilmeye halen devam edilmektedir.

Literatürdeki mevcut çalışmalar, otonom sürüşün şerit takibi ve engelden kaçınma gibi temel görevlerinde derin takviyeli öğrenmenin (DRL) etkinliğini kanıtlamış olsa da bu çalışmaların birçoğu görevi tamamlama başarısına odaklanırken, sürüşün kalitesi, doğallığı ve pürüzsüzlüğü gibi insan benzeri davranışsal nitelikleri ikinci planda

bırakmaktadır. Bununla beraber insan benzeri sürüş performansı hedefleyen çalışmalar genellikle karmaşık ödül denklemleri kullanmakta ya da insan sürücülerin deneyimlerinden yardım almaktadırlar. Bu tez çalışması, literatüre bu noktada önemli bir katkı sunmaktadır. Önerilen modüler mimari, özellikle "insan benzeri pürüzsüz sürüş" hedefine yönelik olarak tasarlanmıştır. Literatürdeki benzer DRL tabanlı sürüş çalışmalarından temel farkı ve en önemli avantajı, sürüş pürüzsüzlüğünü sağlamak için literatürdeki yöntemlerden farklı bir strateji olarak harici aksiyon filtreleme modüllerini kullanması, bu modülleri sistematik olarak karşılaştırması ve bu yaklaşımın sürüş hatası ile pürüzsüzlük arasındaki denge üzerindeki etkilerini nicel olarak analiz etmesidir. Bu bütüncül yaklaşım, otonom araçlar için daha konforlu, öngörülebilir ve toplumsal olarak kabul edilebilir sürüş politikaları geliştirme yolunda literatüre özgün bir metodolojik çerçeve ve deneysel bir bakış açısı kazandırmaktadır.

# 3. MATERYAL ve YÖNTEM

Bu bölümde yapılan çalışmanın amacı, kullanılan materyallerin ve yöntemlerin yapısı ele alınmaktadır.

Bu tezin konu aldığı çalışmanın ana amacı derin takviyeli öğrenme (Deep Reinforcement Learning, DRL) tabanlı modüler bir otonom sürüş sistemi tasarımı oluşturarak otonom bir aracın pürüzsüz ve insan benzeri bir sürüş deneyimi elde etmesini sağlamaktır. Eğitim ve testleri simülasyon ortamında gerçekleştirilen bu çalışma iki ana aşamada oluşmaktadır. "Şerit takibi" olarak adlandırılan ilk aşamada oluşturulan sistem ile kontrol edilen bir aracın iki şeritli bir parkuru hedef hızdan ve ona ayrılan şeritten sapmayacak şekilde tamamlaması beklenmektedir. İkinci aşamada aracımızın bu görevi gerçekleştirirken aynı zamanda yol üzerindeki sabit bir engelden kaçınması beklenmektedir ve bu aşama "şerit takibi ve engelden kaçma" olarak adlandırılmaktadır.

Gerçekleştirilen deneyler için oluşturulan otonom sürüş sistemlerinde karar ve kontrol görevini gerçekleştiren modül için bir DRL yöntemi uygulanmıştır. Bu karar ve kontrol algoritması için $s$ durum uzayı kamera görüntüleri, çeşitli araç bilgileri ve/veya LIDAR nokta bulutu verilerinden oluşmaktadır. Durum uzayındaki bu yüksek boyutlu sensör verileri için oto-kodlayıcı modüller tasarlanarak boyut indirgeme gerçekleştirilmiştir. Karar ve kontrol algoritmasının ürettiği aksiyonların devamlılığını sağlamak adına aksiyon filtreleme modülleri oluşturulmuştur. Aksiyon filtreleme modüllerinin SAC algoritmasının ürettiği aksiyonları pürüzsüzleştirerek otonom sürüş sistemine insan benzeri bir davranış kazandırması hedeflenmiştir.

## 3.1. Donanımsal ve Yazılımsal Bileşenler

Takviyeli öğrenme (Reinforcement Learning, RL) algoritmaları öğrenim süreci boyunca etkileşimde bulunduğu ortamda yüksek miktarda etkileşim verisi toplamaktadır. Bu durum gerçek dünya eğitim ve testlerinin donanımsal ve zamansal açıdan maliyetli kılmaktadır. Bu sorunlara çözüm olarak RL çalışmaları simülasyon ortamlarında

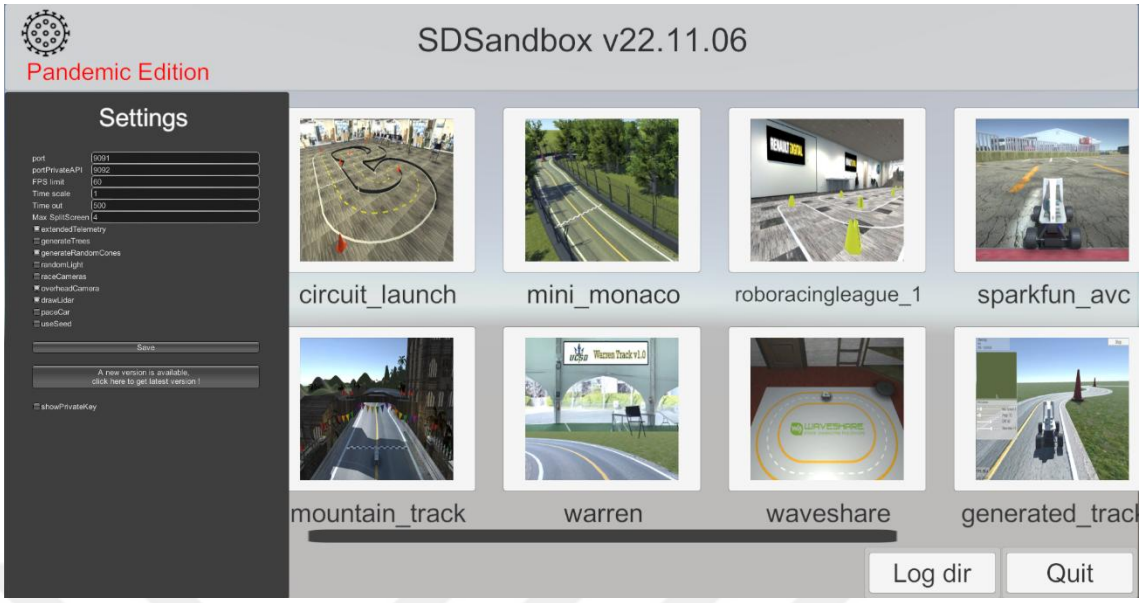
gerçekleştirilmektedir. Bu simülasyon ortamları ayrıca tekrarlanabilirlikleri ve ölçeklenebilirlikleri ile gerçek dünya sistemlerine göre büyük avantajlar sağlamaktadır. Bu tezin konu aldığı çalışma da üç boyutlu bir otonom sürüş simülatörü olan Donkey Car (*About Donkey*®, t.y.) kullanılarak gerçekleştirilmiştir. Bu tür simülasyonlar ve eğitimler gerektirdiği hesaplama maliyetleri nedeni ile güçlü bilgisayar sistemlerine ihtiyaç duymaktadır.

Deneylerin gerçekleştiği bilgisayar sisteminde gerçek zaman simülasyonu ile başa çıkabilmesi amacıyla 4.4 GHz hıza sahip 6 çekirdekli i5-12400F işlemci (CPU) kullanılmıştır. Grafik işlem ünitesi (GPU) olarak 12 GB hafızaya sahip RTX 3060 ekran kartı tercih edilmiştir. Ekran kartı sahip olduğu 3584 CUDA çekirdeği ile durum uzayı üzerindeki işlemleri etkili bir şekilde gerçekleştirebilir. Ortam ve model parametrelerine, büyük deneyim yığınlarına hızlı bir şekilde erişebilmek ve hafızada rahatça tutabilmek için 32 GB boyutunda rastgele erişimli bellek (RAM) kullanılmıştır.

Kullanılan bilgisayar sistemi için bir Linux dağıtımı olan Fedora işletim sistemi tercih edilmiştir. Simülatör arayüz iletişimi için Python ortamını desteklemektedir. Tüm sistem yazılımı Python dilinde oluşturulmuştur. Karar ve kontrol algoritması SAC ve oto-kodlayıcıların oluşturulması için bir derin öğrenme kütüphanesi olan Tensorflow temel alınmıştır.

### 3.2. Simülasyon Ortamı

Bu tezin konu aldığı çalışma için gerekli olan simülasyon platformu olarak Donkey Car simülatörü (*About Donkey*®, t.y.) kullanılmıştır. Donkey Car projesi mühendis, araştırmacı ve bilim insanları tarafından robot araç yapımı amacıyla kurulmuş otonom robot araç geliştirme platformudur. Bu platformun sağladığı Donkey Car simülatörü RL yöntemleri için geliştirilen bir Python çerçevesinde kullanılabilmektedir. Donkey Car simülatörünün arayüzü Şekil 3.1'de gösterilmektedir.




**Şekil 3.1.** Simülasyon platformunun arayüzü

Donkey Car simülatör ortamı çeşitli otonom sürüş yöntemleri, algılama, kontrol ve benzeri çalışmalar için araştırmacılara alan sağlamaktadır (*About Donkey®*, t.y.). Simülatör arayüzünde ajanların eğitimi için birden fazla ortam seçeneği mevcuttur. Bu çalışmada gerçekleştirilmek istenen görevlere uygun olması adına Donkey Car simülatöründe "*mountain-track*" isimli ortam tercih edilmiştir. Şekil 3.2'de tercih edilen ortamın kuşbakışı simülasyon içi görüntüsü verilmiştir.

**A**



**B**




**Şekil 3.2.** Simülasyon içi görüntüler
**A)** Deneyler için kullanılan parkur **B)** Simülasyon içi sürüş görüntüsü

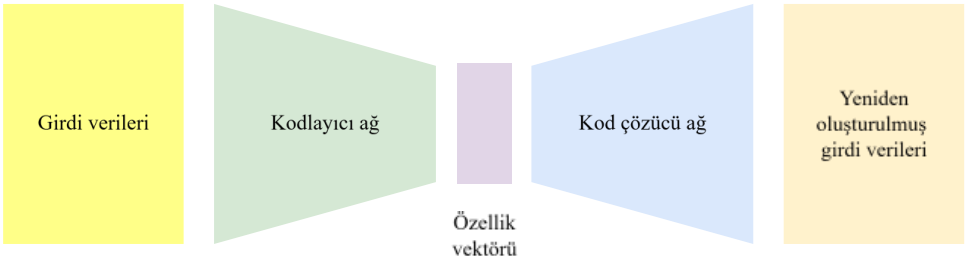
Donkey Car simülasyon ortamı araca ait bir takım sensör verilerine ve ek bilgilere ulaşmayı sağlamaktadır (*About Donkey*®, t.y.). Bu sensörler ve bilgiler aşağıda listelenmektedir:

- **Kamera**: Aracın önünü gösteren bir RGB kamerası mevcuttur. Kamera görüntüleri 3 kanallı olup 160x180 boyutundadır.
- **IMU**: Araç içinde aracın eksenel (x, y, z) ivmesini veren bir sensör mevcuttur.
- **GPS**: Aracın ortamdaki konumunu (x, y, z) veren bir GPS modülü mevcuttur.
- **LIDAR**: Aracın etrafındaki engelleri tespit etmek amacı ile 2 boyutlu bir LIDAR mevcuttur. LIDAR 360 derecede etkilidir.
- **Hız bilgisi**: Aracın hızını belirtmektedir.
- **İvme bilgisi**: Aracın eksenel ivmesini belirtmektedir.
- **Çarpışma bilgisi**: Aracın bir engele çarpıp çarpmadığını ifade etmektedir.
- **Şeritten merkezinden sapma (cross track error, CTE) bilgisi**: Aracın şerit merkezine olan uzaklığını belirtmektedir.

Simülasyon ortamındaki aracın kontrolü, iki aktüatör aracılığıyla sağlanır: direksiyon simidi açısı ve gaz/fren pedalı. Her iki aktüatör de -1 ile 1 arasında sürekli bir değer alabildiğinden, ajanımız iki boyutlu, sürekli bir aksiyon uzayında hareket etmektedir. Bu yapıda gaz ve fren, tek bir aktüatör olarak modellenmiştir.

### 3.3. Durum Uzayı Tasarımı

Gerçekleştirilen deneylerde otonom sürüş için kamera görüntüleri ve/veya LIDAR nokta bulutu temel veriler olarak kullanılmıştır. Kamera ve LIDAR verilerinin boyutlarının büyük olması dolayısı ile karar ve kontrol için kullanılan DRL algoritmaları düşük verimli bir öğrenme sergilemektedir. Yavaş ve verimsiz öğrenmenin önüne geçebilmek için modüler yapının avantajından yararlanılıp büyük boyutlu sensör verileri için bir ön işleme başvurulmuştur. Bu ön işlem için otonom araçlarda daha önceki çalışmalarda yapılanlara benzer olarak kamera ve LIDAR sensör verileri için birer oto-kodlayıcı yapısı oluşturulmuştur.




**Şekil 3.3.** Oto-kodlayıcıların temsili yapısı

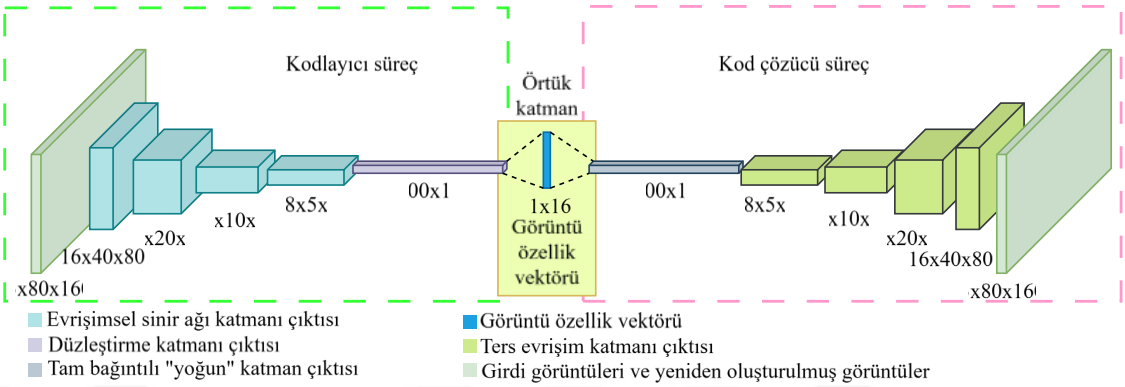
Şekil 3.3.'de genel yapısı verilmiş olan bir oto-kodlayıcı, genellikle denetimsiz öğrenmede kullanılan bir derin ağdır (Bank vd., 2023). Oto-kodlayıcılar kodlama ve kod çözücü olarak adlandırılabilen iki işlem için kullanılır. Kodlayıcı ağ kademeli olarak azalan boyutlu sinir ağı mimarisine sahipken kod çözücü ağ kademeli olarak artan boyutlu bir sinir ağı mimarisine sahiptir.

Oto-kodlayıcı eğitimi sonucunda kodlayıcı ağ girdi verilerini temsil eden çok daha küçük boyutlu bir vektör oluşturabilmektedir. Oto-kodlayıcının bu çalışmadaki kullanım amacı ve avantajları şu şekildedir:

- **Boyut indirgeme:** Özellikle üç boyutlu bir veri olan kamera görüntüsü, oto-kodlayıcı sayesinde tek boyutlu küçük ölçekli bir diziye dönüştürülerek kompakt hale getirilir (Hinton & Salakhutdinov, 2006). Aynı işlem 360 derece açıyı tarayan LIDAR nokta bulutu verileri üzerinde de gerçekleştirilmektedir. Bu sayede boyutu indirgenen sensör verileri sayesinde derin ağların hesaplama maliyeti azaltılmakta ve verilerin bellekte kapladığı alan küçülmektedir.
- **Özellik çıkarma:** Sinir ağları ile oluşturulan oto-kodlayıcı eğitimi sayesinde kamera görüntüleri ve LIDAR nokta bulutu verileri üzerinde özellik çıkarımı işlemi uygulanmaktadır (Bengio, 2009). Ham verilerden öğrenmenin zor olması nedeni ile bu özellik çıkarımı işlemi DRL'nin verimini ve hızını arttırmaktadır.
- **Doğal gürültü giderme:** Oto-kodlayıcılar yapısı gereği doğal bir gürültü giderme yöntemi olarak da çalışabilmektedirler (Kramer, 1992). Bu durum olası sensör ve sensör dışı gürültünün giderilmesine olanak sağlamaktadır.

Oluşturulan oto-kodlayıcıların eğitimi için simülasyon içinde insan sürüşünden elde edilen on bin adet örnek kamera ve LIDAR verisi kullanılmıştır. Kamera görüntüleri

ilgili alan kırpılarak 160x80 boyutuna getirilmiştir. Şekil 3.4'te bu kamera görüntüleri için oluşturulan oto-kodlayıcının yapısı gösterilmektedir.



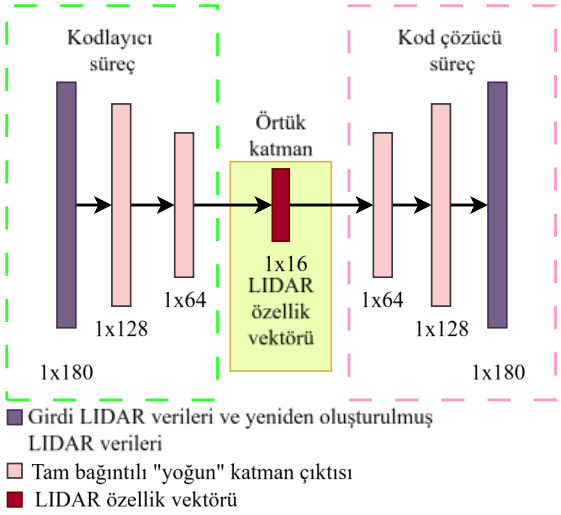

**Şekil 3.4.** Kamera görüntüleri için oluşturulan oto-kodlayıcı

Bu çalışmada üç boyutlu kamera görüntüleri kodlayıcı ağdaki evrişimli sinir ağları sayesinde kademeli olarak küçültülerek 1x16 boyutunda bir vektöre dönüştürülmüştür. Bu vektör kod çözücü ağ ile tekrar üç boyutlu görüntülere geri dönüştürülür. Eğitim sürecinde oto-kodlayıcı orijinal görüntü ve geri dönüştürülen görüntü arasındaki ortalama hata ile geri beslenerek güncellenir. Yeterli eğitim süreci sonucunda kodlayıcı ağın oluşturmuş olduğu 1x16 boyutundaki vektör orijinal görüntünün sahip olduğu özellikleri önemli ölçüde temsil edebilmektedir. Kamera görüntüleri ile eğitilen oto-kodlayıcının farklı eğitim döngüsü (epoch) değerlerindeki sonuçlarına ait karşılaştırması Şekil 3.5'te gösterilmiştir. Orijinal görüntüye en yakın olanı oluşturabilmesi adına 100 eğitim döngüsü değeri seçilmiştir.



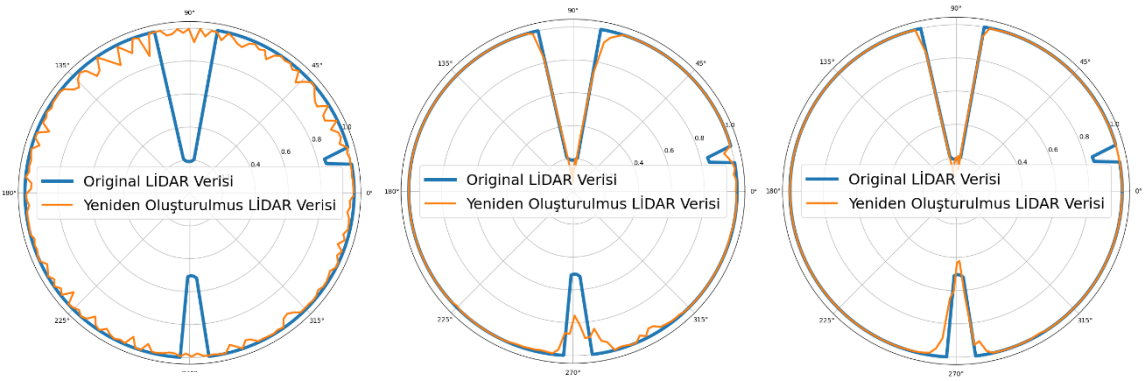

**Şekil 3.5.** Kamera görüntüleri için oluşturulan oto-kodlayıcının farklı eğitim döngüsü değerlerinde yeniden oluşturulan görüntüler

LIDAR nokta bulutu verileri için oluşturulan oto-kodlayıcının yapısı Şekil 3.6'da gösterilmektedir. Burada kodlayıcı ağın çıktısı 1x16 boyutunda bir vektördür.




**Şekil 3.6.** LIDAR nokta bulutu verileri için oluşturulan oto-kodlayıcı yapısı

Şekil 3.7'de 360 dereceyi kapsayan 1x180 boyundaki LIDAR nokta bulutu verilerinin orijinal halinin 10, 50 ve 100 eğitim döngüsü değerlerinde eğitilen oto-kodlayıcı tarafından yeniden oluşturulmuş hali şekillendirilmiştir. Burada farklı eğitim döngüsü sonucunda oluşturulan LIDAR sinyali ile orijinal sinyalin arasındaki fark gözlenmektedir.




**Eğitim döngüsü: 10** **Eğitim döngüsü: 50** **Eğitim döngüsü: 100**

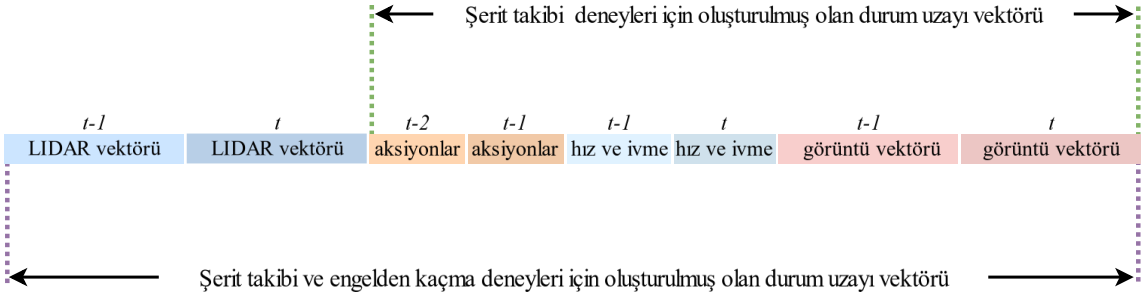
**Şekil 3.7.** LIDAR verileri için oluşturulan oto-kodlayıcının farklı eğitim döngüsü değerlerinde yeniden oluşturulan sinyaller

Bu çalışmada oluşturulmuş olan oto-kodlayıcı mimarilerinin tasarımı ve parametrelerinin seçimi, görevin gereksinimleri ile literatürdeki en iyi uygulamalar arasında bir denge kurmayı hedefleyen, sistematik ve deneysel bir sürece

dayanmaktadır. Her iki sensör verisi için de kodlayıcı ağın çıktısı olan özellik vektörü boyutu, deneysel bir validasyon süreci sonucunda 16 olarak belirlenmiştir. Oto-kodlayıcıların eğitimi, 100 eğitim döngüsü sonunda sonlandırılmıştır. Bu karar, eğitim kayıp değerlerinin yakınsadığı ve yeniden oluşturulan çıktıların kalitesinde anlamlı bir iyileşmenin gözlenmediği noktada, modelin eğitim verisine aşırı uyum sağlamasını önlemek ve genelleştirilebilir bir özellik temsili elde etmek amacıyla alınmış standart bir erken durdurma işlemidir.

Otonom sürüş için karar ve kontrolü sağlayan SAC algoritması için bir durum uzayı vektörü oluşturmak amacıyla *t ve t-1* anında elde edilen kamera ve LIDAR sensörlerine ait kodlanmış özellik vektörleri kullanılmıştır. Bu vektörlerin yanında aracın *t* ve *t-1* anındaki hızı ve ileri yöndeki ivmesi bilgileri eklenmiştir. Son olarak *t* anı durum vektörü *t-1* ve *t-2* anlarında ajanın gerçekleştirmiş olduğu aksiyon bilgileri eklenerek genişletilmiştir. Durum uzayı vektöründe *t-1* anlarına ait her bilgi ajanın ortamı anlama yetisini arttırarak daha iyi bir politika geliştirmesine yardımcı olması adına eklenmiştir.

Deneyler için oluşturulan *t* durum uzayı vektörünün temsili Şekil 3.8.'da verilmektedir. Bu tezin konu aldığı çalışmalarda LIDAR verileri yalnızca "şerit takibi ve engelden kaçma" deneylerinde kullanılmıştır.

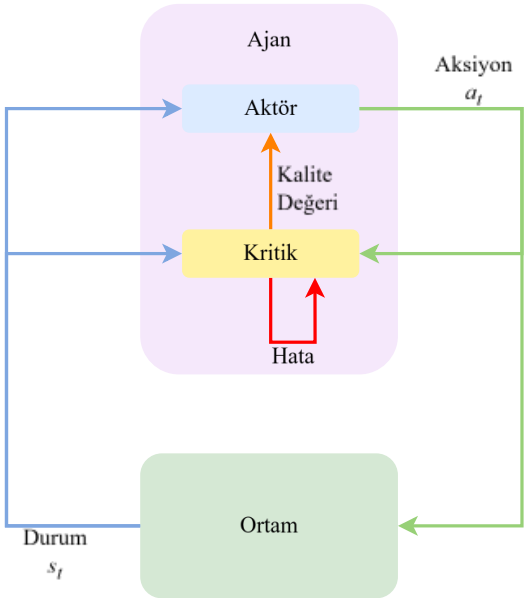



**Şekil 3.8.** Bir *t* anı için oluşturulan $s_t$ durum uzayı vektörünün yapısı gösterilmektedir.

### 3.4. Karar ve Kontrol Algoritması

Bu çalışmada tasarlanan otonom sürüş sistemde karar ve kontrol işlevlerini yönetmek için bir DRL algoritması olan Soft Aktör-Kritik (SAC) algoritması uygulanmıştır. SAC algoritması bu sistem içinde otonom sürüşü öğrenen modül olmakla yükümlüdür.

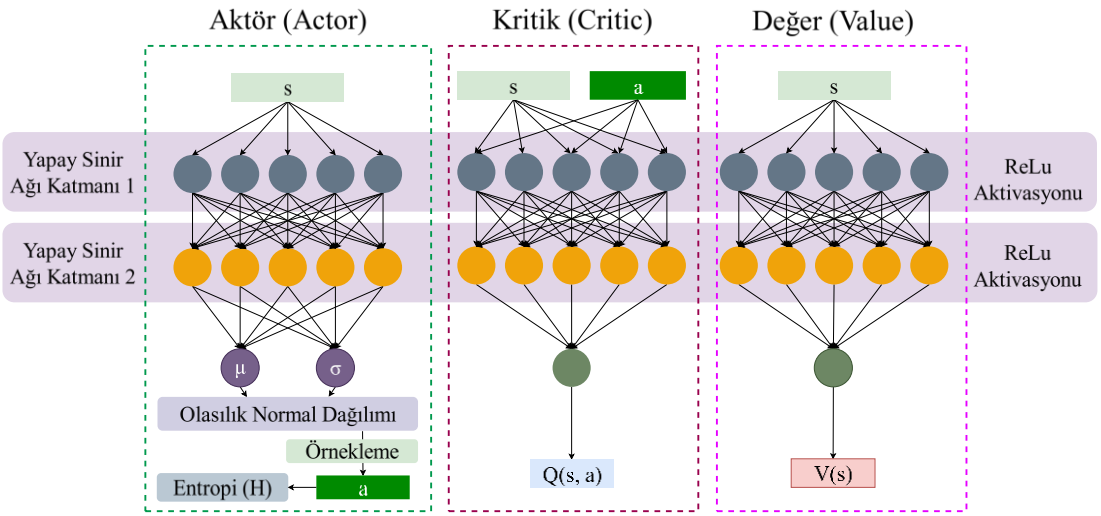
SAC, 2018 yılında geliştirilmiş olup sürekli eylem alanlarında etkili, modelden bağımsız, politika dışı bir algoritmadır (Haarnoja vd., 2018). Aynı zamanda politika tabanlı ve değer tabanlı yaklaşımları başarılı şekilde birleştirerek yenilikçi bir yapı sunmaktadır. SAC algoritması bir aktör-kritik yaklaşımıdır. Bir aktör-kritik algoritması olarak SAC politika ve değer tabanlı yaklaşımları Şekil 3.9'da gösterilen yapıyı kullanarak birleştirmektedir (Dutta & Upreti, 2022). Aktör-kritik yaklaşımlarda temelde iki ağ mevcuttur. Burada aktör ağı optimal politikayı politika tabanlı yaklaşımla öğrenmek ile görevlendirilmektedir. Aktör, gerçekleştirilecek aksiyonu belirlemektedir. Kritik ağı ise aktör ağı tarafından gerçekleştirilmiş olan aksiyonların değerini değer tabanlı yaklaşımlar ile hesaplamaktadır. Bu aksiyon değerleri aktör ağının gerçekleştirmiş olduğu aksiyonların ne derece iyi olduğunu ifade etmekte ve aktör ağının politikasının güncellenmesi için yol göstermektedir (Grondman vd., 2012).




**Şekil 3.9.** Aktör-Kritik yapısına sahip takviyeli öğrenme algoritmaları için temsili eğitim süreci

Bu tezin konu aldığı çalışmada uygulanmış SAC algoritmasında üçüncü bir yardımcı bileşen olan değer ağı bulunmaktadır. Bu değer ağı kritik ve aktör yapılarına yardımcı olabilmek ve algoritmanın hesaplamalarına katkıda bulunabilmek için durumların değer tahminini sağlamaktadır. Değer tahmini kritik ağın dengeli bir şekilde kendisini güncellemesine destek sağlamaktadır. Bu değer ağı algoritmanın sonraki versiyonlarında hesaplama yükünü bir miktar artırdığı gerekçesi ile kaldırılmış ve kritik ağı kendini optimize edecek şekilde güncellenmiştir. Bu çalışmada deneyler sırasında kodlama düzeyinde sistemin çalışmasını daha kolay denetleyebilmek ve durum değerlerini takip ederek algoritmanın sağlıklı çalıştığını teyit edebilmek adına değer ağına sahip olan 2018 yılında yayınlanmış algoritma versiyonu (Haarnoja vd., 2018) kullanılmıştır.

SAC algoritmasında bu aktör, kritik ve değer ağları derin sinir ağlarından oluşmaktadırlar (Haarnoja vd., 2018). Bu şekilde SAC algoritması sinir ağlarının ve derin öğrenmenin hesaplama ve genellemedeki avantajlarını kullanabilmektedir.




**Şekil 3.10.** Çalışmada SAC için oluşturulan aktör, kritik ve değer ağlarının yapıları

Şekil 3.10'da bu çalışma için kullanılan aktör, kritik ve değer ağlarının yapıları gösterilmektedir. Her bir ağ için benzer şekilde ReLu aktivasyonuna sahip ikişer gizli sinir ağı katmanları mevcuttur.

$$u_t = \mu_\phi(s_t) + \sigma_\phi(s_t) \cdot \epsilon_t \qquad \epsilon \sim N(0,1) \tag{3.1}$$

SAC algoritmasında aksiyon seçimi stokastik bir süreç olarak tanımlanmaktadır. Deterministik bir aksiyon üretmek yerine bir olasılık dağılımı üretilmektedir. Verilen bir $s_t$ durumunda $\phi$ parametrelerine sahip bir aktör ağı (ya da politika ağı), gerçekleştirilecek aksiyon seçimi için ortalama ($\mu_\phi$) ve aksiyon etrafındaki belirsizliği belirten standart sapma ($\sigma_\phi$) değerlerini hesaplamaktadır. Aktör ağı tarafından hesaplanan bu değerler ile normal dağılım kullanılarak aksiyon değerleri $u_t$ örneklenmektedir (Denklem 3.1) (Haarnoja vd., 2018). Bu aksiyon örnekleme denkleminde $\epsilon_t$ ifadesi $t$ zaman adımında rastgele bir gürültüyü belirten sabittir. Elde edilen aksiyon değerleri ortamın izin verdiği sınır değer aralığına uygun olması için hiperbolik tanjant fonksiyonu ile dönüştürülerek örneklenen aksiyon $a_t$ belirlenir (Denklem 3.2). Burada $a_t$ değeri deterministik bir denklem olarak $f_\phi(\epsilon_t; s_t)$ şeklinde ifade edilmektedir.

$$a_t = \tanh(u_t) = f_\phi(\epsilon_t; s_t) \tag{3.2}$$

Bu süreçte örneklenen aksiyonlar ile ajan ortamla etkileşime geçerek $s_t, a_t, r_t, s_{t+1}, d_t$ olarak ifade edilen deneyim verilerini toplamaktadır (Denklem 3.3). Ajan ortam içinde etkileşimini sürdürürken her zaman adımında bu veri yığınından ($\mathcal{D}$) rastgele olarak küçük yığınlar örneklenir ve bu yığınlarla eğitim gerçekleştirilir. Bu durum deneyim tekrarı (experience replay) olarak adlandırılmaktadır.

$$(s_t, a_t, r_{t+1}, s_{t+1}, d_t) \sim \mathcal{D} \tag{3.3}$$

SAC algoritmasında aktör ağı ile bir aksiyon örneklenirken aynı zamanda örneklenen aksiyonun seçilme olasılığının logaritması da hesaplanmaktadır. Bu logaritmik değer entropi olarak ifade edilmektedir. Entropi burada aksiyon olasılık dağılımındaki belirsizliği nitelendirmektedir. Bir politikanın entropisi, yani rastgeleliği denklemi (Denklem 3.4) $H\left(\pi_\phi(\cdot\,|s_t)\right)$ ile ifade edilmektedir (Haarnoja vd., 2018).

$$H\left(\pi_{\phi}(\cdot\,|s_t)\right) = -\mathbb{E}_{a_t\sim\pi_{\phi}}\left[\log \pi_{\phi}(a_t|s_t)\right] \tag{3.4}$$

SAC algoritmasında entropi, ağların güncellenmesinde ajanın keşif isteğini teşvik eden bir kavram olarak kullanılmaktadır. Bu keşif teşviği sayesinde aksiyonlar için deterministik bir seçim yapılmasının önüne geçmek hedeflenmektedir. Entropi yükseldikçe rastgelelik artmaktadır (Haarnoja vd., 2018). Bu durumda ajanın amacı sadece beklenen ödülü maksimize etmek değil aynı zamanda entropiyi de maksimize etmektir. SAC algoritmasında özellikle eğitimin başlarında yerel optimuma düşmemek için yüksek entropi değerine sahip olmak istenmektedir.

SAC algoritmasında $\theta$ parametrelerine sahip kritik ağını oluşturan derin sinir ağı sayesinde rastgele örneklenmiş küçük yığındaki her bir $s_t$ durumunda gerçekleştirilen $a_t$ aksiyonunun kalite değeri olarak $Q_{\theta}(s_t, a_t)$ tahmin edilmektedir[1] (Haarnoja vd., 2018).

Son olarak SAC algoritmasında durum değer ağı mevcuttur (Haarnoja vd., 2018). Burada durum değer ağı gerçekte birbirine özdeş yapıda iki derin sinir ağından oluşmaktadır.[2] Bunlardan ilki ψ parametrelerine sahip ana değer ağıdır. Ana değer ağı $s_t$ durumu için $V_{\psi}(s_t)$ durum değerini, yani o durum için beklenen ödülü tahmin etmektedir. Diğer bir durum değer ağı ise $\bar{\psi}$ parametrelerine sahip hedef değer ağıdır. Bu ağ ise $s_{t+1}$ durumu için $V_{\bar{\psi}}(s_{t+1})$ durum değerini tahmin etmekle görevlidir.

[1] SAC algoritmasında birbiri ile aynı yapıda ancak ayrı ayrı güncellenen iki adet kritik ağı mevcuttur. Bu algoritmada kritik ağının ürettiği kalite değerleri kullanılmak istendiğinde bu iki kritik ağ tarafından üretilen değerler arasında en az olanı tercih edilmektedir. Özellikle eğitim başında yüksek kalite değerlerinin tahmin edilmesi ile daha istikrarsız bir eğitimin önüne geçmek için önlem olarak bu yaklaşım oluşturulmuştur.

[2] Derin takviyeli öğrenme algoritmaları içerisinde ana ve hedef durum değer ya da kalite değer ağları kullanmak ve bu hedef ağlarını polyak ortalaması ile güncellemek öğrenme sürecini çok daha kararlı hale getirmektedir. Bu işlem birçok derin takviyeli öğrenme algoritmasında standart olarak uygulanmaktadır.

Eğitim süreci boyunca kritik ağının $Q_{hedef}(s_t, a_t)$ olarak ifade edilen gerçek kalite değerlerine göre kendini güncellemesi gerekmektedir. Hedef kalite değerleri hesaplanırken durum değeri ağlarından yardım alınmaktadır. Hedef kalite değeri denkleminde (Denklem 3.5), $r(s_t, a_t)$ ifadesi $s_t$ durumunda $a_t$ aksiyonunun gerçekleştirilmesi ile elde edilen ödülü ve $\gamma$ indirim katsayısını temsil etmektedir (Haarnoja vd., 2018). Burada $V_{\overline{\psi}}(s_{t+1})$ ifadesi küçük yığındaki her bir $s_{t+1}$ için durum değerini belirtmektedir ve bu değer hedef durum değer ağı tarafından tahmin edilmektedir.

$$Q_{hedef}(s_t, a_t) = R(s_t, a_t, s_{t+1}) + \gamma \cdot \mathbb{E}_{s_{t+1} \sim p}\left[V_{\overline{\psi}}(s_{t+1})\right] \tag{3.5}$$

Kritik ağının kendini hafızadan örneklenmiş küçük yığın üzerinden güncellemesi için gerekli kayıp denklemi (Denklem 3.6) tahmin edilen kalite değerleri ile hesaplanan hedef kalite değerleri arasındaki ortalama hata karesidir (Haarnoja vd., 2018).

$$J_Q(\theta) = \mathbb{E}_{(s_t, a_t) \sim \mathcal{D}}\left[\frac{1}{2}\left(Q_\theta(s_t, a_t) - Q_{hedef}(s_t, a_t)\right)^2\right] \tag{3.6}$$

SAC algoritmasında ana durum değer ağının kendini güncellenmesi için gerekli hedef durum değeri hesabı için aktör ağı güncel politikasını ($\pi_\phi$) kullanarak küçük yığındaki her $s_t$ durumu için yeni $a_t$ aksiyonları üretmekte ve bu aksiyonların entropi değerini hesaplamaktadır. Beklenti yeni üretilen aksiyonlar üzerinden alınmaktadır. Buna göre $V_{hedef}(s_t)$ olarak ifade edilen hedef durum değeri denkleminde (Denklem 3.7) $Q_\theta(s_t, a_t)$ kritik ağı tarafından tahmin edilen kalite değeri, $log\, \pi_\phi(a_t|s_t)$ ise hesaplanmış olan entropi değerini ifade etmektedir. Burada $\alpha$, entropi için ağırlıklandırma katsayısıdır (Haarnoja vd., 2018).

$$V_{hedef}(s_t) = \mathbb{E}_{a_t \sim \pi_\phi}\left[Q_\theta(s_t, a_t) - \alpha \cdot \log \pi_\phi(a_t|s_t)\right] \tag{3.7}$$

Hedef durum değer denkleminde entropi eğitim başlangıcında yüksek kalite ve durum değerleri için düzenleyici görev almaktadır. Bu sayede politikanın deterministik hale gelmesi yavaşlatılmaktadır. Bu durum keşif isteğini arttırmaktadır.

$$J_V(\psi) = \mathbb{E}_{s_t \sim \mathcal{D}}\left[\frac{1}{2}(V_\psi(s_t) - V_{hedef}(s_t))^2\right] \tag{3.8}$$

Ana durum değer ağının güncellenmesi için gerekli kayıp denklemi (Denklem 3.8) tahmin edilen değer ile hedef değer arasındaki ortalama hata karesidir (Haarnoja vd., 2018). Ana durum değer ağı, hedef durum değer ağının güncellenmesi için eğitilmektedir. Eğitim süreci boyunca $\bar{\psi}$ olarak ifade edilen hedef durum değer ağının parametreleri bir $\tau$ (polyak averaging) sabiti ile $\psi$ olarak ifade edilen ana değer ağının parametrelerinden yavaşça (soft update) güncellenmektedir (Denklem 3.9) (Haarnoja vd., 2018). Bu durum eğitim süreci boyunca durum değer ağlarının kendisine referans vermesini engelleyerek daha kararlı bir öğrenme sağlamaktadır.

$$\bar{\psi} \leftarrow \tau \cdot \psi + (1-\tau) \cdot \bar{\psi} \qquad \tau \in [0,1] \tag{3.9}$$

Aksiyonlara karar veren aktör ağının güncellenmesi aşamasında aktör ağı güncel politika ile küçük yığındaki her bir örnek için yeni aksiyon üretmektedir. Üretilen aksiyonların entropi değeri hesaplanmaktadır. Bununla beraber küçük yığındaki her bir durum ve yeni aksiyon değerleri için kritik ağı $Q_\theta(s_t, a_t)$ kalite değerini tahmin etmektedir. Denklem 3.10'da (Haarnoja vd., 2018) aktör ağının optimizasyonu için gerekli kayıp denklemi verilmiştir. Bu kayıp denkleminde aktör ağı maksimum kalite değerlerini bulmak isterken entropi terimi aynı değer ağının kayıp denkleminde olduğu gibi aktörün deterministik davranmasına engel olabilmek için kullanılmaktadır. Bu şekilde ajanın keşif isteği korunmaktadır.

$$J_\pi(\phi) = \mathbb{E}_{s_t \sim \mathcal{D}, a_t \sim \pi_\phi}\left[\alpha \cdot \log \pi_\phi(a_t|s_t) - Q_\theta(s_t, a_t)\right] \tag{3.10}$$

### 3.5. Aksiyon Gürültüsü

Bu çalışma için oluşturulan otonom sürüş sistemlerinde ajanların keşfe teşvikini arttırmak ve bu sayede yerel optimumlardan kaçınabilmek için aksiyon gürültüsü bir modül olarak eklenmiştir. Aksiyon gürültüsü, DRL algoritmaları içerisinde sık kullanılan ve kullanılması tavsiye edilen bir bileşendir (Hollenstein vd., 2022b).

DRL içerisinde birçok farklı aksiyon gürültüsü kullanılabilmektedir. Bu çalışmadaki deneylerde aksiyon gürültüsü olarak robotik alanında sık tercih edilen *Ornstein-Uhlenbeck* (OU) süreci kullanılmaktadır. OU gürültü modeli sürecinde, geçmiş değerlere bağlı şekilde bir gürültü sinyali oluşturulmaktadır (Bibbona vd., 2008). Bu süreçte üretilen gürültü değerleri ortalamaya doğru eğilim göstermektedir. Ardışık zaman sinyalleri arasında sağlanan korelasyon daha yumuşak değişimler oluşturarak gürültü değerleri arasında keskin değişimleri engellemektedir.

$$dx_t = \theta(\mu - x_t)\, dt + \sigma\, dW_t \tag{3.11}$$

OU süreci için kullanılan denklemde (Denklem 3.11), $x_t$, $t$ zamanındaki gürültü sinyalini; $\theta$, sürecin ortalamaya yönelim hızını kontrol eden değişken; $\mu$, süreç için ortalama değeri; σ, rastgelelik şiddetini kontrol eden değişken; $W_t$, *Wiener* sürecini ifade etmektedir. *Wiener* süreci OU gürültüsünde rastgeleliği temsil etmektedir. Programlama düzeyinde OU süreci denkleminde (Denklem 3.12) *Wiener* süreci Gauss dağılımından örneklenen rastgele bir sayı olarak ifade edilmektedir.

$$x_{t+1} = x_t + \theta(\mu - x_t)dt + \sigma\sqrt{dt}\epsilon \quad \epsilon \sim N(0,1) \tag{3.12}$$

### 3.6. Aksiyon Filtreleme

Bu çalışmada konu alınan otonom araç sistemlerinin insan benzeri pürüzsüz bir davranış elde edebilmesi için kullanılan aksiyon filtreleme teknikleri bu bölümde tanıtılmaktadır. Aksiyon filtreleme modülünün görevi karar ve kontrol algoritmasının ürettiği aksiyonu önceki adımlarda gerçekleştirilen aksiyonları dikkate alarak filtreleyerek

pürüzsüzleştirmektir. Bu işlemle birlikte otonom sürüşte aksiyonlar arası sürekliliğin sağlanması hedeflenmektedir. Aksiyon filtreleme modülü ile daha konforlu ve insansı bir sürüş deneyimi sağlamak için ajanların içerisinde bulunduğu durumlara karşı aşırı tepkisel aksiyon almasının önüne geçmek amaçlanmıştır. Aksiyon filtreleme modülü, ortamın dinamiklerini değiştirip karmaşıklaştırmaktadır. Bu modül aracılığıyla SAC algoritmasının daha karmaşık ortamlardaki performansını incelerken insan benzeri bir sürüş deneyimi elde edilmesi hedeflenmektedir.

Aksiyon filtreleme modülü iki farklı alçak geçiren filtre seçeneği sunmaktadır. Bunlardan ilki üstel hareketli ortalama (exponential moving average, EMA) tekniğidir. Üstel hareketli ortalama filtresi denkleminde (Denklem 3.13), $\tilde{a}_t$, $t$ zamanında gerçekleştirilecek olan filtrelenmiş aksiyon; $a_t$, $t$ zamanında karar ve kontrol algoritmasının gerçekleştirmek istediği aksiyon; $\tilde{a}_{t-1}$, önceki adımda gerçekleştirilmiş olan filtrelenmiş aksiyon; $w$, denge (trade-off) katsayısıdır. EMA filtresi sonucu ajan gerçekleştirmek istediği aksiyon değeri ile en son adımda gerçekleştirmiş olduğu aksiyon değeri arasında bir denge kurarak filtrelenmiş aksiyonu oluşturmaktadır.

$$\tilde{a}_t = w \cdot a_t + (1 - w) \cdot \tilde{a}_{t-1} \tag{3.13}$$

Bu modülde kullanılabilen ikinci filtre ağırlıklı hareketli ortalama (weighted moving average, WMA) tekniğidir. WMA filtresi ajanın gerçekleştirmiş olduğu aksiyonları geriye doğru ağırlıklandırarak ortalamasını almaktadır. Ağırlıklı hareketli ortalama denkleminde (Denklem 3.14), $\tilde{a}_t$, gerçekleştirilecek olan filtrelenmiş aksiyonu; $a_t$, $t$ zamanında karar ve kontrol algoritmasının gerçekleştirmek istediği aksiyonu; $n, (n-1), \ldots, 1$, geriye doğru aksiyon ağırlıklarını; $\tilde{a}_{t-1}, \tilde{a}_{t-2}, \ldots, \tilde{a}_{(t-n)+1}$, geriye yönelik filtrelenmiş aksiyonları temsil etmektedir. Bu işlem sonucunda filtrelenmiş aksiyon değeri, en son gerçekleştirilen aksiyon değerleri dikkate alınarak elde edilir.

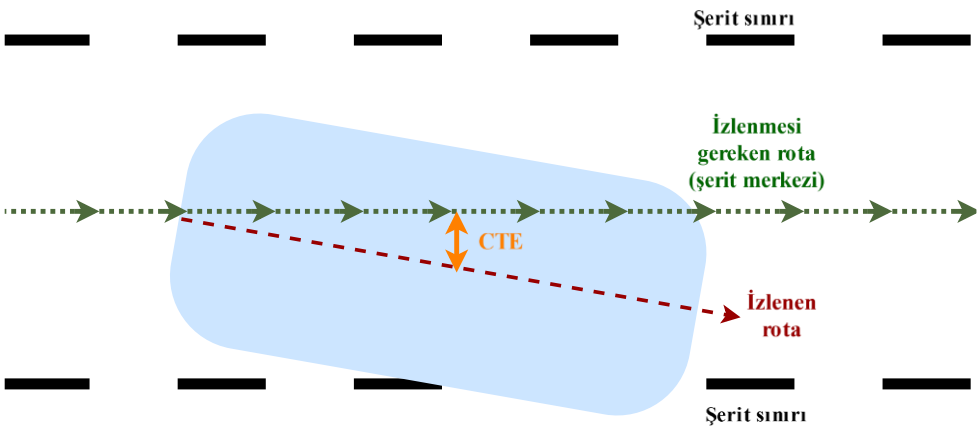
$$\tilde{a}_t = \frac{n \cdot a_t + (n-1) \cdot \tilde{a}_{t-1} + \cdots + (1) \cdot \tilde{a}_{(t-n)+1}}{n + (n-1) + \cdots + 1} \tag{3.14}$$

Otonom sürüşün başlangıç adımında $\tilde{a}_{t-1} = \emptyset$ olacağından tüm filtreler için $\tilde{a}_t = a_t$ olarak kabul edilmektedir. Bu iki denklem özelinde, çalışmada gerçekleştirilen deneyler için $n = 5$, ve $w = 0{,}5$ olarak tercih edilmiştir.

### 3.7. Ödül Denklemi Tasarımı

RL içerisinde ödül denklemi, ajanın gerçekleştirmeyi öğrenmesi gereken görevler için rehber niteliğinde yönlendirici bir yapıdır. Ödül tasarımı ortam, görev, durum bilgileri, aktüatörler ele alınarak tasarlanmaktadır. Bu tasarım sonucunda pozitif ya da negatif ödül yapıları oluşturulabilmektedir. Hedeflenen görev doğrultusunda gerçekleştirilmesi istenen aksiyonlar pozitif ödül yapısını ve gerçekleştirmesi istenmeyen aksiyonlar negatif ödül yapısını şekillendirmektedir (reward shaping).

Karar ve kontrol algoritması olan SAC için gerekli görevleri ve istenmeyen davranışları merkeze alarak üç bileşen içeren bir ödül denklemi tasarlanmıştır. Söz konusu bileşenlerden ilki $R_{CTE}$ değeri, aracın şerit içerisindeki pozisyonuna odaklanmaktadır. Aracın şerit merkezine olan uzaklığı "*cross track error*" (CTE) olarak ifade edilmektedir. Şekil 3.4.1'de yol üzerindeki aracın CTE değeri gösterilmektedir.

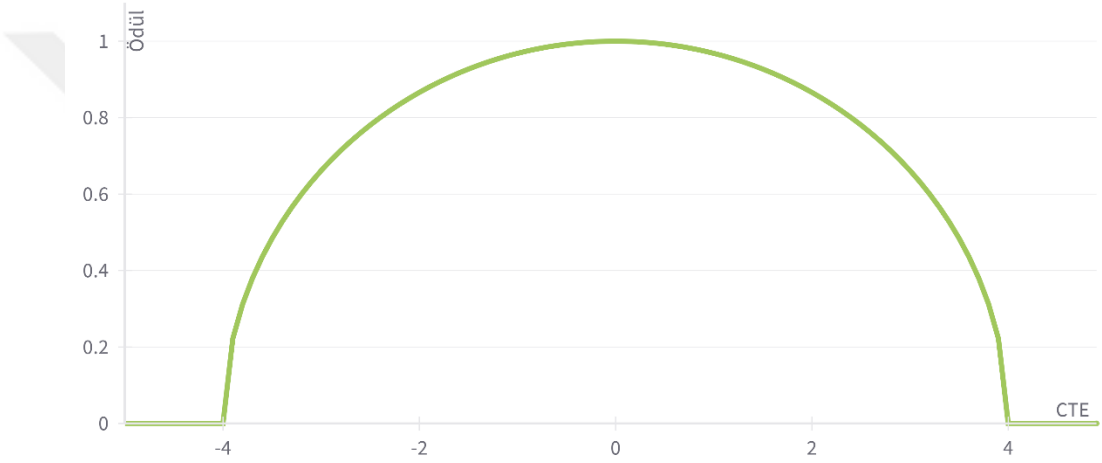



**Şekil 3.11.** Şerit merkezine olan uzaklık (CTE) değerinin temsili

Bir $t$ anında elde edilebilecek $R_{CTE}$ ödül bileşeni denkleminde (Denklem 3.15) $CTE_t$ değişkeni Şekil 3.11'de görselleştirilmiştir. Burada $CTE_t$ ifadesi aracın $t$ anında şerit

merkezine olan uzaklığını ifade etmektedir. Bu bilgi simülatör tarafından sağlanmaktadır. $CTE_{max}$ ifadesi ise şerit merkezine izin verilen maksimum uzaklık olarak tanımlanmıştır. $R_{CTE}$ bileşeni ile aracın şerit merkezine olan yakınlığı ödüllendirilmektedir ($R_{CTE} \in [0,1]$). Simülasyon ortamı için $CTE_{max} = 4.0$ birim olarak seçilmiş olup $R_{CTE}$ ödül bileşeninin grafiği Şekil 3.12'de verilmektedir.

$$R_{CTE}(\mathrm{t}) = \sqrt{1 - \left(\frac{|CTE_t|}{CTE_{\ max}}\right)^2} \tag{3.15}$$

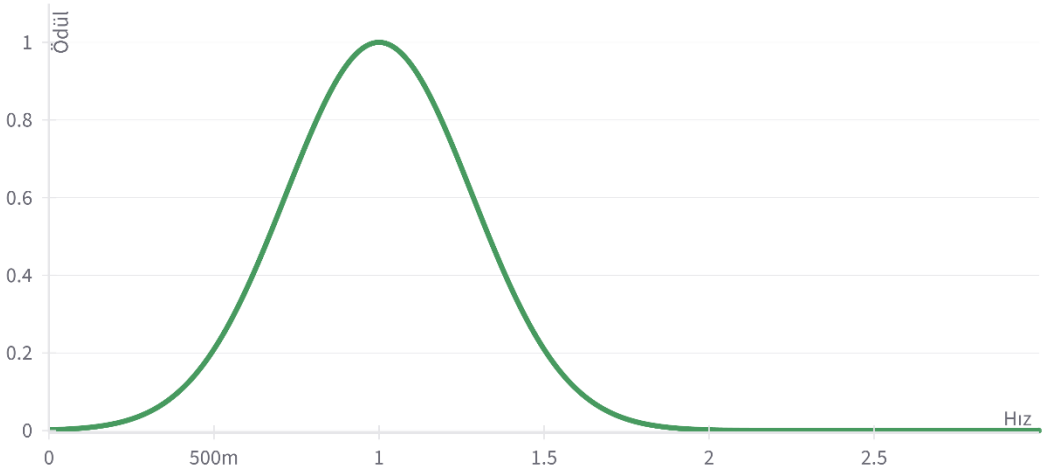



**Şekil 3.12.** CTE ödülü $R_{CTE}$ için dağılım grafiği

İkinci ödül bileşeni aracın hızı ile ilişkilendirilmektedir. Hedef hıza yakınlığın ödüllendirildiği bileşen $R_{hız}$, bir *Gauss* denkleminden oluşmaktadır. Bu denklemde (Denklem 3.16) $hız_{hedef}$, arzu edilen hız; $hız_t$, aracın $t$ anındaki hızı; $\sigma$, varyans değeridir ($R_{hız} \in [0,1]$).

$$R_{hız}(\mathrm{t}) = \mathrm{e}^{\left(-\frac{(hız_t - hız_{hedef})^2}{2\sigma^2}\right)} \tag{3.16}$$

Simülasyon ortamı için $hız_{hedef} = 1.0$ birim ve $\sigma = 0.2$ olarak tercih edilmiştir. Buna ilişkin olarak $R_{hız}$ ödül bileşeninin grafiği Şekil 3.13'te verilmektedir.




**Şekil 3.13.** Hız ödülü $R_{hız}$ için dağılım grafiği

Bu iki ödül bileşeni otonom sürüş görevlerinde en temel görevler olup yol üzerinde hatayı nitelemektedir. Buna karşın üçüncü ödül bileşeni ajanın pürüzsüz bir sürüş gerçekleştirip aksiyon sürekliliğini korumasına yardımcı olmak adına ardışık aksiyon değişim oranı ile ilişkilendirilmektedir (Abouelazm vd., 2024).

$$R_{aksiyon}(t) = \left(\frac{a_t - a_{t-1}}{\Delta a_{max}}\right)^2 \cdot j \tag{3.17}$$

Aksiyona bağlı ödül bileşeni olan $R_{aksiyon}$ denkleminde (Denklem 3.17) $a_t$, aracın $t$ anında gerçekleştirdiği aksiyon; $a_{t-1}$, *t-1* anındaki aksiyon; $\Delta a_{max}$, ardışık aksiyonların alabileceği maksimum fark değeri; $j$, aksiyon değişim ödülü bileşeni için maliyet katsayısıdır. Burada $j$=0,5 olarak tercih edilmiştir.

$$R(t) = \begin{cases} -1{,}0 & , maksimum\ CTE\ aşımı\ ya\ da\ engele\ çarpma \\ \dfrac{R_{CTE}(t) \cdot R_{hız}(t)}{R_{aksiyon}(t) + 1} & , sürüş\ durumu \end{cases} \tag{3.18}$$

Ödül denklemi (Denklem 3.18) bu üç bileşenin birleşiminden oluşmaktadır. Sürüş anında şerit merkezine ve hedef hıza olan yakınlık ödülü arttırmaktadır. Buna karşın $R_{aksiyon}$ sisteme denge sağlamak için ceza işlevi görmektedir. $R_{aksiyon}$ ödül bileşeni burada aksiyon filtreleri ile aynı amaç doğrultusunda, aksiyonlar arasındaki sürekliliği korumaya yardımcı olması amacıyla eklenmiştir.

### 3.8. Başarı Metrikleri

Deneyler için oluşturulan ajanların eğitim sürecinde performanslarını gözlemleyebilmek adına RL literatüründe olduğu gibi ajanların;

- Bölümlerde gerçekleştirdiği adım sayısı (bölüm uzunluğu),
- Bölümlerde aldığı toplam ödül miktarı,
- Bölümlerdeki ortalama CTE,
- Bölümlerdeki ortalama hız değerleri eğitim ile eş zamanlı olarak izlenmiş analiz edilmiştir.

İzlenen bu değerler dışında ajanın görevini ne kadar iyi icra ettiğini gösteren metrikler oluşturulmuştur. Burada ajanların en temel görevi şerit içinde kalacak ve hedef hızı koruyacak şekilde bir otonom sürüş gerçekleştirmektir. Buna istinaden eğitim için oluşturulan ödül denkleminin bileşenleri olan $R_{CTE}$ ve $R_{hız}$ denklemlerinden yararlanılarak bir metrik tasarlanmıştır. Aracın deneyimlediği bir bölüm başına şerit merkezine ve hedef hıza uzaklığını temsil eden ortalama hata (mean error, ME) bir başarı metriği olarak kullanılmıştır. Ortalama hata denkleminde (Denklem 3.19) *T* değeri bölümü oluşturan adım sayısını ifade etmektedir.

$$Hata = \frac{1}{T}\sum_{t=1}^{T}\left(1 - \frac{R_{CTE}(t) + R_{hız}(t)}{2}\right) \tag{3.19}$$

Ajanların gerçekleştirmesi istenilen bir diğer temel görev ise pürüzsüz sürüş deneyimi elde etmeleridir. Buradaki başarısını gözlemleyebilmek için ardışık iki aksiyon arasındaki değişim miktarı bir başarı metriği olarak kullanılmıştır. Ardışık iki aksiyon arası değişim miktarı denkleminde (Denklem 3.20) $a_t$, aracın *t* anında gerçekleştirdiği aksiyon; $a_{t-1}$, *t-1* anındaki aksiyon; $\Delta a_{max}$, ardışık aksiyonların alabileceği maksimum fark değeri; *T*, bölümü oluşturan adım sayısını ifade etmektedir.

$$\bar{a} = \frac{1}{T}\sum_{t=1}^{T}\left(\frac{a_t - a_{t-1}}{\Delta a_{max}}\right)^2 \tag{3.20}$$

# 4. BULGULAR

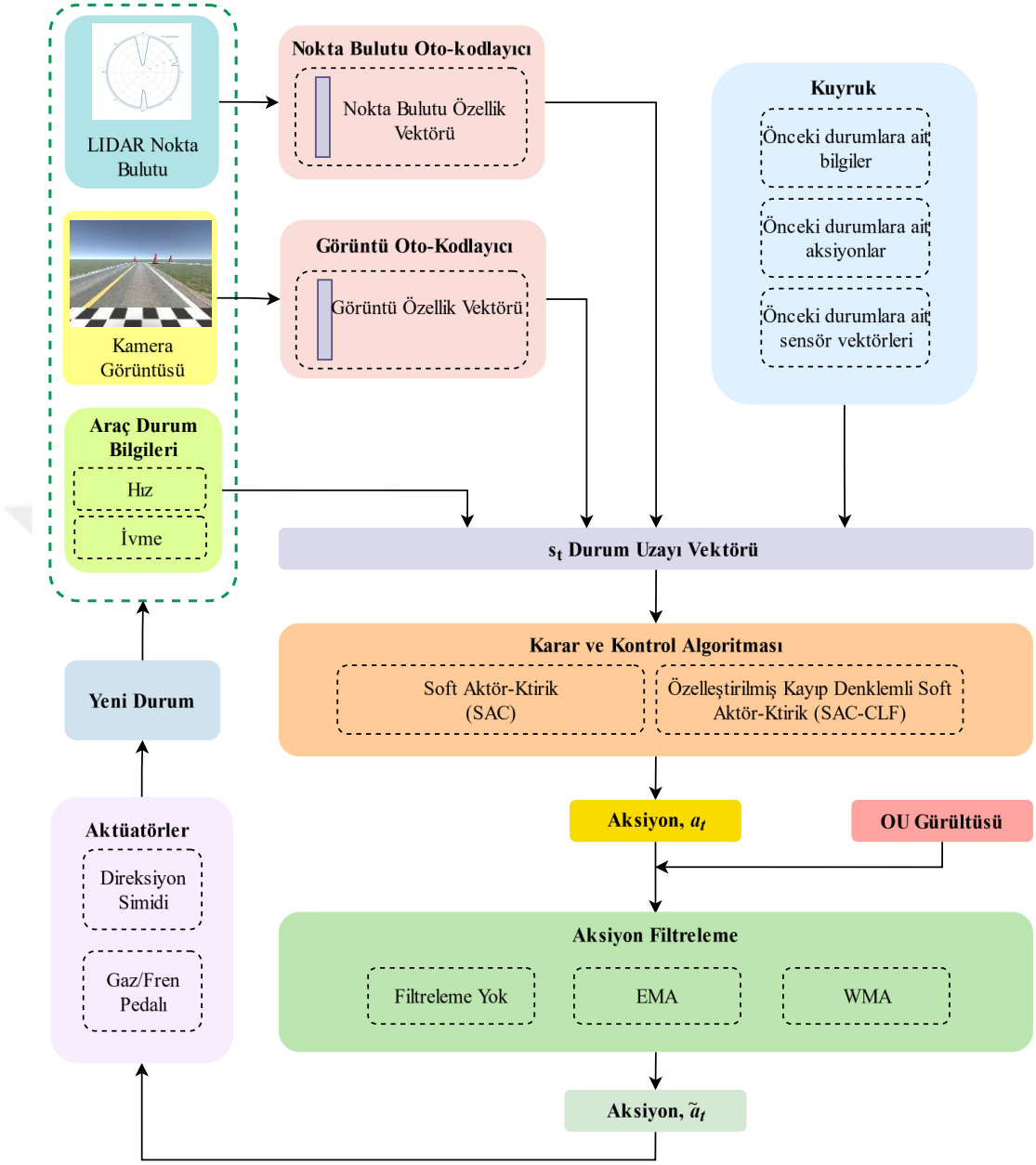
Bu bölümde çalışma için oluşturulan otonom sürüş sistemlerinin mimarilerine genel bir bakış sağlanarak bu sistemlerin özelliklerine değinilmiş ve her bir sistem için eğitim ve test performansları gözlenmiştir. Elde edilen bulgular güncel literatür ve teknoloji doğrultusunda tartışılmıştır.

Bu tez çalışması için gerçekleştirilen deneyler iki aşamadan oluşmaktadır. İlk aşama aracın şerit takibi yaparak parkuru tamamlaması üzerine gerçekleştirilen "şerit takibi" deneyleridir. İkinci aşama aracın şerit takibi yaparken yol üzerindeki bir engelden kaçmaya çalışması üzerine gerçekleştirilen "şerit takibi ve engelden kaçma" deneyleridir. Her iki aşamada gerçekleştirilen deneyler için oluşturulan tüm modellerinin yapıları için genel bir bakış Şekil 4.1'deki akış şemasında sağlanmıştır. Şekil 4.1'deki akış şeması doğrultusunda:

- Bir $t$ anı için ilk aşama deneylerinde kamera görüntülerinin ikinci aşama deneylerinde ise hem kamera görüntülerinin hem de LIDAR nokta bulutu verilerinin ön işlemesi yapılmakta ve gerekli araç durum bilgileri ile durum uzayı vektörü $s_t$ oluşturulmaktadır.
- Oluşturulan durum uzayı vektörü karar ve kontrol yöntemi olan SAC algoritmasına beslenmektedir. SAC algoritmasında aktör ağı için orijinal kayıp denklemi dışında özelleştirilmiş bir kayıp denklemi seçeneği de oluşturulmuştur. Bu özelleşmiş kayıp denkleminde (Denklem 4.1) orijinal kayıp denklemi ve ardışık aksiyon değişimi ($\Delta a$) arasında bir denge katsayısı ($w$) kullanılarak EMA filtresine benzetim sağlanmıştır.

$$J_{CLF} = (1 - w) \cdot J_{\pi}(\phi) - w \cdot \Delta a \tag{4.1}$$

- SAC algoritması gerçekleştirmeyi arzu ettiği bir aksiyon değeri üretmektedir. Bu aksiyon değerine "OU Gürültüsü" modülü ile gürültü eklenebilmektedir.
- Son olarak aksiyon filtreleme bloğu isteğe göre iki farklı aksiyon filtreleme seçeneği sunabilmektedir. Filtrelenen aksiyon değeri $\tilde{a}_t$ aktüatörlere iletilerek yeni durum elde edilir. Böylece döngü tekrarlanır.




**Şekil 4.1.** Oluşturulan modüler sistemin mimarisi

Bu bilgiler doğrultusunda tüm deneylerde eğitim için oluşturulan modellerin yapısına dair bilgilere Çizelge 4.1'de değinilmektedir. SAC olarak ifade edilen modelde eğitim süreci boyunca orijinal kayıp denklemine sahip SAC karar ve kontrol algoritmasının gerçekleştirmek istediği aksiyon değerleri doğrudan ortama iletilmektedir. SAC+Gürültü modelinde eğitim süreci boyunca orijinal kayıp denklemine sahip SAC karar ve kontrol algoritmasının gerçekleştirmek istediği aksiyon değerleri gürültü

eklenerek ortama iletilmektedir. SAC-CLF+ Gürültü modelinde eğitim süreci boyunca özelleştirilmiş kayıp denklemine sahip SAC karar ve kontrol algoritmasının gerçekleştirmek istediği aksiyon değerleri gürültü eklenerek ortama iletilmektedir. SAC+Gürültü+EMA modelinde eğitim süreci boyunca orijinal kayıp denklemine sahip SAC karar ve kontrol algoritmasının gerçekleştirmek istediği aksiyon değerlerine gürültü eklendikten sonra bu değerler EMA aksiyon filtresi ile filtrelenmektedir. Bu modelde filtrelenmiş aksiyon değeri ortama iletilmektedir. SAC+Gürültü+WMA modelinde eğitim süreci boyunca orijinal kayıp denklemine sahip SAC karar ve kontrol algoritmasının gerçekleştirmek istediği aksiyon değerlerine gürültü eklendikten sonra bu değerler WMA aksiyon filtresi ile filtrelenmektedir. Filtrelenmiş aksiyon değeri ortama iletilmektedir.

**Çizelge 4.1.** Eğitimler için oluşturulan modellerin yapıları

| Sistem/Model | Açıklama |
|---|---|
| SAC | • Aktör Kayıp Fonksiyonu: Orijinal<br>• Gürültü: Hayır<br>• Aksiyon Filtresi: Hayır |
| SAC+Gürültü | • Aktör Kayıp Fonksiyonu: Orijinal<br>• Gürültü: Evet<br>• Aksiyon Filtresi: Hayır |
| SAC-CLF+Gürültü | • Aktör Kayıp Fonksiyonu: Özelleştirilmiş<br>• Gürültü: Evet<br>• Aksiyon Filtresi: Hayır |
| SAC+Gürültü+EMA | • Aktör Kayıp Fonksiyonu: Orijinal<br>• Gürültü: Evet<br>• Aksiyon Filtresi: EMA |
| SAC+Gürültü+WMA | • Aktör Kayıp Fonksiyonu: Orijinal<br>• Gürültü: Evet<br>• Aksiyon Filtresi: WMA |

Eğitilen tüm modellerin gürültüsüz şekilde test sürüşleri gerçekleştirilmiş olup bu sürüşlerin performansı oluşturulan başarı metrikleri ile ölçülmüştür. Test sonuçlarını hata ve pürüzsüzlük açısından daha iyi değerlendirebilmek için bu test sürüşlerin ekran kaydı alınarak performansları insan gözlemi ile de değerlendirilmiştir.

Bu bölümde verilmiş olan grafiklerdeki değerler okunurluğu arttırmak amacı ile zaman ağırlıklı üstel hareketli ortalama tekniği kullanılarak yumuşatılmıştır.

### 4.1. Şerit Takibi Deneyleri

Şerit takibi deneyleri için oluşturulan tüm modellerde kullanılan SAC algoritmaları aynı hiperparametre değerlerine sahiptir ve bu parametre değerleri Çizelge 4.2'de verilmiştir. Bu parametreler algoritmanın öğrenme dinamiklerini ve performansını doğrudan etkileme potansiyeline sahiptir.

Şerit takibi görevi için kullanılan SAC algoritmasının hiperparametreleri, robotik ve sürekli kontrol alanlarındaki DRL literatüründe kabul görmüş standartlar ve bu çalışmanın özelindeki simülasyon ortamının dinamiklerine yönelik yapılan deneysel ayarlamalar sonucunda belirlenmiştir. SAC algoritmasında 256 olarak belirlenen yığın boyutu (batch size), gradyan tahminlerinin kararlılığı ile hesaplama verimliliği arasında yaygın olarak tercih edilen bir denge sunmaktadır. Kullanılan SAC algoritmasında ortamın karmaşıklığı dikkate alınarak 512 birimlik tam bağıntılı iki adet sinir ağı katmanı bulunmaktadır. İndirim katsayısı olarak gamma değeri 0,97 olarak ayarlanmıştır; 1'e yakın bu değer uzun vadeli planlamayı teşvik etmektedir. Entropi katsayısı ajanın yerel optimal noktalara takılıp kalmasını önleyecek kadar keşfe olanak tanırken, aynı zamanda etkili bir politika yakınsamasına da izin veren 0,2 değeriyle deneysel olarak saptanmıştır. Değer ağı için yumuşak güncelleme katsayısı $\tau$ ise 0,005 olarak tercih edilmiştir. Bu düşük değer, hedef ağların yavaş ve yumuşak bir şekilde güncellenmesini sağlayarak, politika dışı algoritmalarda sıkça karşılaşılan eğitim istikrarsızlıklarını önlemede etkili bir rol oynamaktadır. SAC algoritması için *Adam* optimizasyon tekniği kullanılmıştır.

**Çizelge 4.2.** Şerit takibi görevinde SAC algoritması için kullanılan eğitim parametreleri

| Parametre | Değer |
|---|---|
| Yığın Boyutu | 256 |
| Gamma (γ) | 0,97 |
| Tau (τ) | 0,005 |
| Sıcaklık (α) | 0,2 |
| Sinir ağı katmanları boyutu | 512 (x2) |

Deneyler için oluşturulan modeller 260 bölüm boyunca eğitilmiştir. Bir eğitim bölümü eğitilen ajan izin verilen maksimum şerit merkezine uzaklığı aştığında veya toplamda 300 adım sayısına ulaştığında sonlandırılmıştır.

### 4.1.1. Eğitim süreci performansları

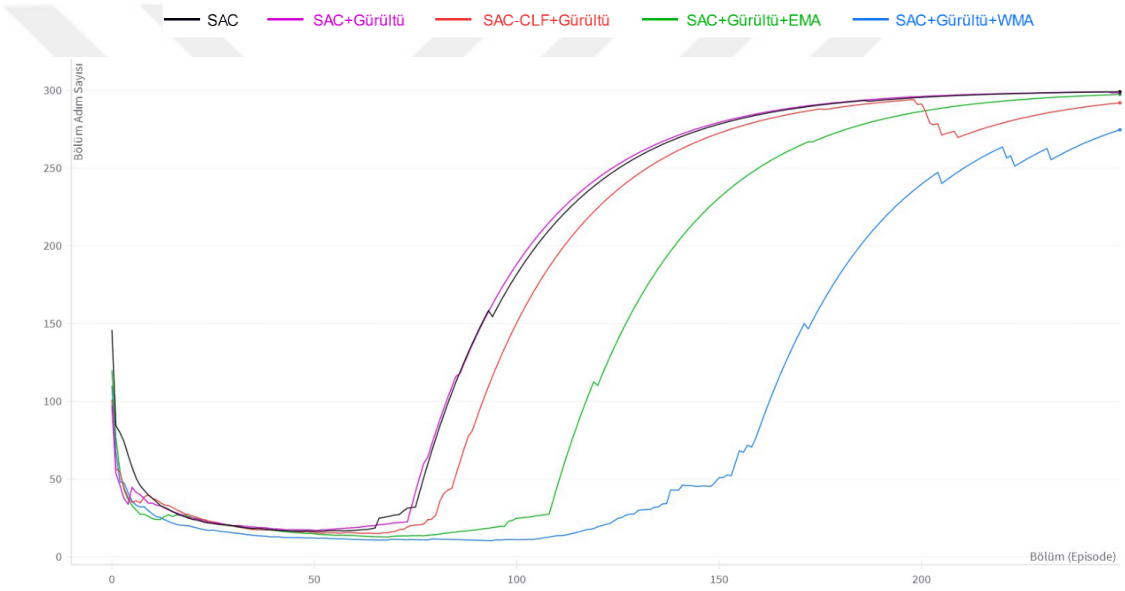
Her bir modelin 260 bölümden oluşan eğitim sürecini tamamlama süreleri Çizelge 4.3'te verilmiştir. Şerit takibi deneyleri için toplam eğitim süresi 14 saat 20 dakikadır.

**Çizelge 4.3.** Şerit takibi görevinde modellerin eğitim süreleri

| Model | Eğitim Süresi |
|---|---|
| SAC | 3 Saat 16 Dakika |
| SAC+Gürültü | 3 Saat 20 Dakika |
| SAC-CLF+Gürültü | 3 Saat 10 Dakika |
| SAC+Gürültü+EMA | 2 Saat 41 Dakika |
| SAC+Gürültü+WMA | 1 Saat 53 Dakika |

Çizelge 4.3 incelendiğinde, aksiyon filtresi (EMA veya WMA) kullanan modellerin (SAC+Gürültü+EMA ve SAC+Gürültü+WMA) eğitimlerini diğer modellere kıyasla daha kısa sürede tamamladığı görülmektedir. Ancak bu bilgi tek başına, modellerin verimliliğine dair yanıltıcı çıkarım ifade etmektedir. Eğitim sürelerine ilişkin farkın nedeni Şekil 4.2'de verilmiş olan modellerin eğitim süreci boyunca bölüm başına adım sayısı grafiği ile açıklanabilmektedir. Bu grafik doğrultusunda aksiyon filtreli

modellerin eğitim süreci boyunca özellikle erken aşamalarda bölümleri daha erken bitirdiği gözlenmiştir. Bu durum aksiyon filtreli modellerin karmaşıklaşan ortam dinamiklerine bağlı olarak daha fazla hata yaparak ortama uyum sağlamakta zorlandığını göstermektedir. Bunun sonucunda aksiyon filtreli modeller diğer modellere kıyasla bir bölüm için izin verilen maksimum adım sayısına daha geç yakınsamıştır. Dolayısı ile "daha kısa toplam eğitim süresi", bu modellerin daha hızlı veya daha iyi öğrendiği anlamına gelmeyip, öğrenme sürecinin daha çalkantılı olduğunu ve stabil bir politika geliştirmek için daha fazla "başarısız denemeye" ihtiyaç duyduklarını işaret edebilir.

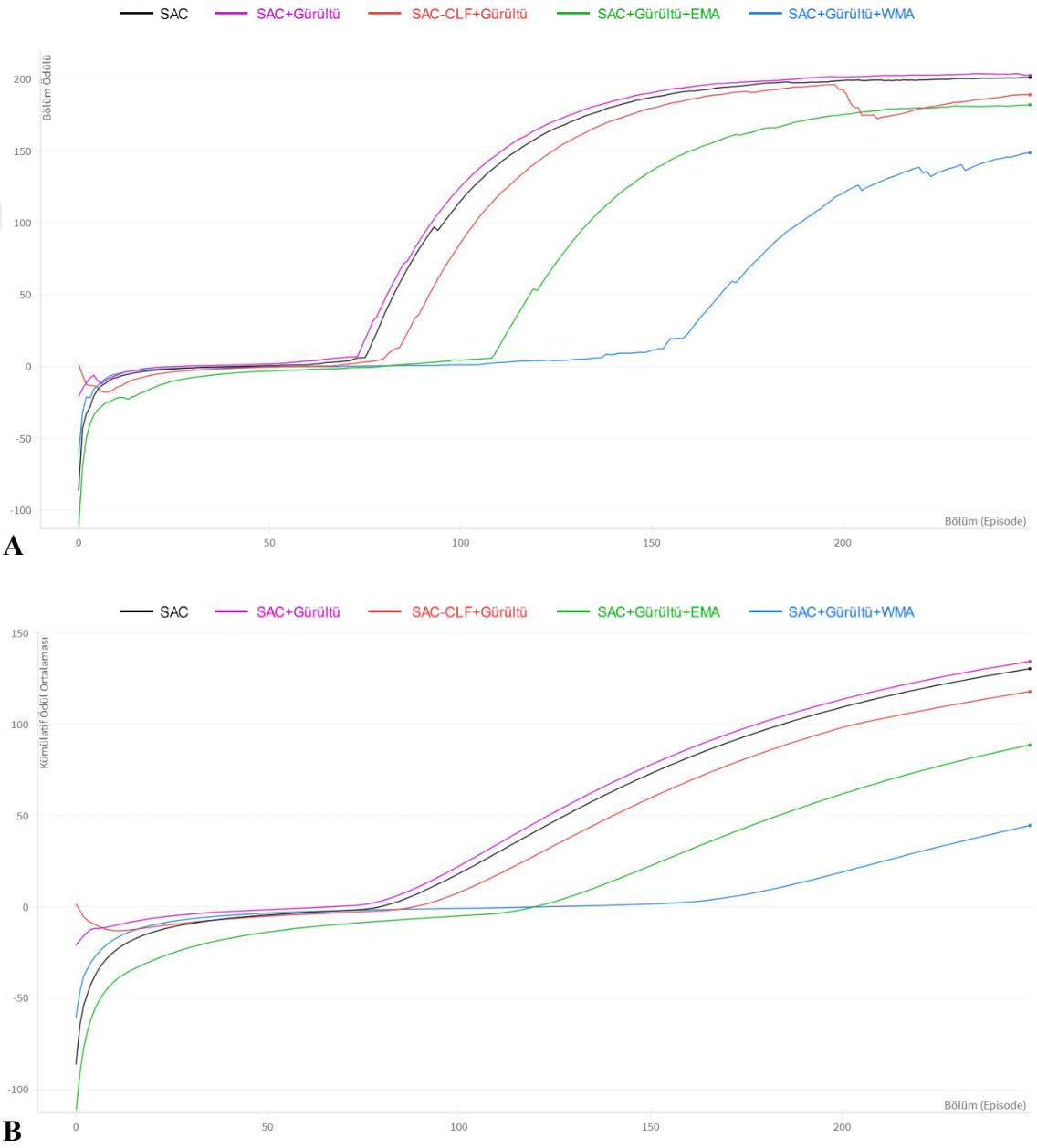



**Şekil 4.2.** Şerit takibi görevi eğitiminde modellerin bölüm başına adım sayısı grafiği

Şekil 4.3.A'da eğitim süreci boyunca bölüm başına toplanan ödül grafiği ve Şekil 4.3.B'de kümülatif ödül ortalaması grafiği gösterilmektedir. Bu grafiklerde bölüm başına adım sayısı grafiğine benzer bir eğilim gözlenmiştir.

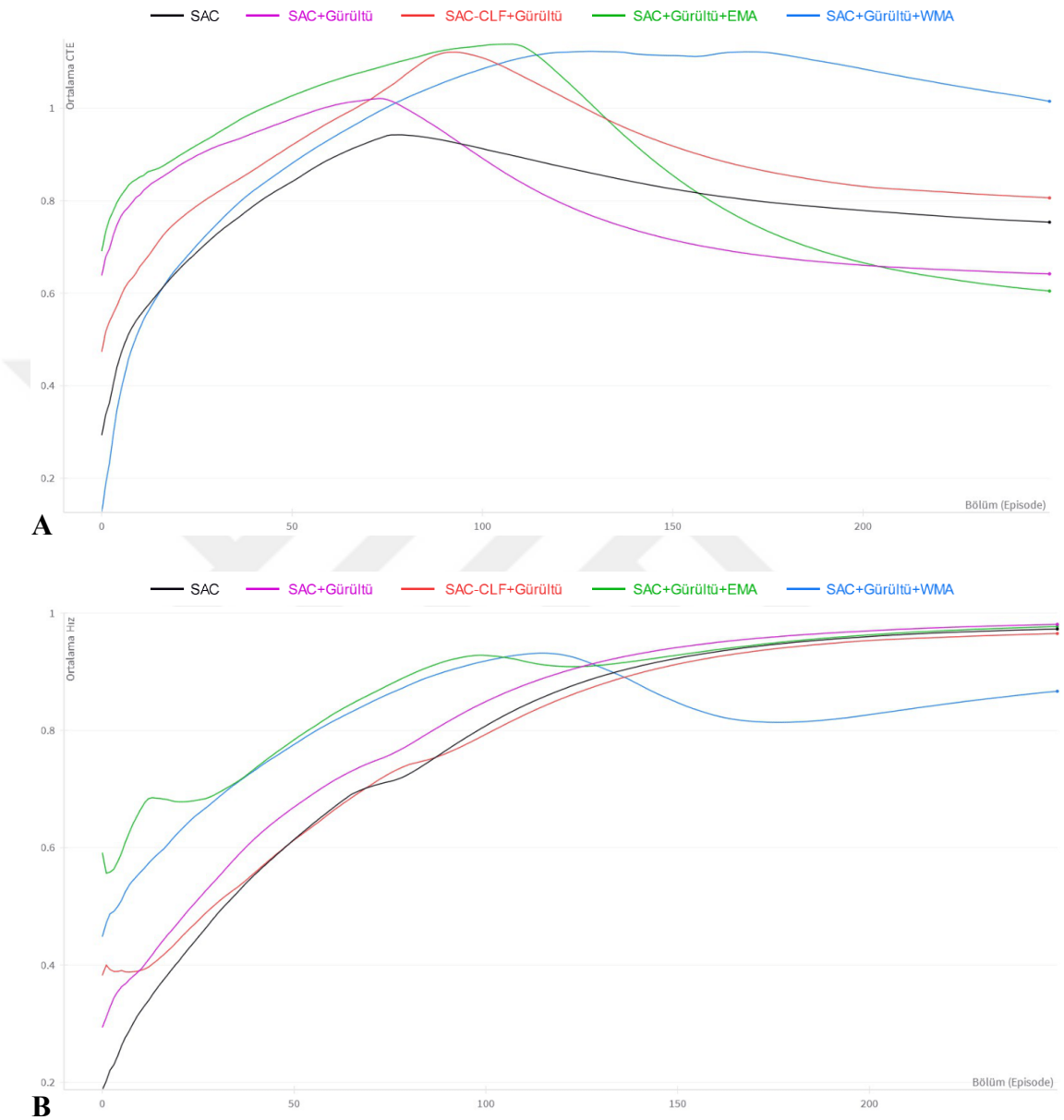
Aksiyon filtresine sahip sistem modellerinin maksimum ödül miktarına diğer modellere göre daha geç yakınsadığı görülmektedir. Özellikle SAC+Gürültü+WMA modelinin, ödülleri maksimize etme konusunda diğer modellere kıyasla en fazla zorlandığı tespit edilmiştir. Bu durum, WMA filtresinin uyguladığı güçlü pürüzsüzleştirme etkisinin, ajanın anlık olarak en yüksek ödülü getirebilecek optimal aksiyonları seçme esnekliğini

kısıtladığını göstermektedir. Ajan, filtrenin getirdiği kısıtlamalar dahilinde hareket etmek zorunda kaldığından, özellikle eğitimin erken aşamalarında çevreyi keşfederken ve temel kontrol stratejilerini öğrenirken bu bir engel teşkil edebilir. Ancak, bu durumun test aşamasındaki sürüş pürüzsüzlüğüne olumlu yansıma potansiyeli de bulunmaktadır.




**Şekil 4.3.** Şerit takibi görevi eğitimlerinde modellerin ödül değişim grafikleri
**A)** Bölüm başına toplam ödül grafiği **B)** Kümülatif ödül ortalaması grafiği

Eğitim süreci boyunca modellerin bölüm başına ortalama CTE değeri grafiği Şekil 4.4.A'da ve ortalama hız değeri grafiği Şekil 4.4.B'de verilmiştir.

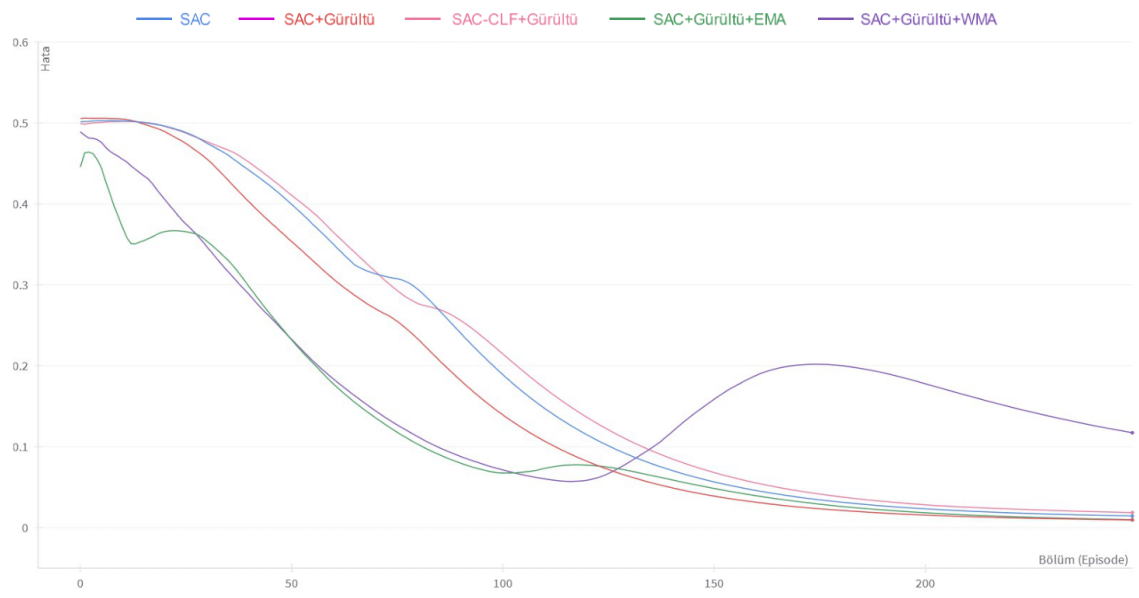



**Şekil 4.4.** Şerit takibi görevi eğitiminde modellerin CTE ve hız değişim grafikleri
**A)** Bölüm başına ortalama CTE grafiği **B)** Bölüm başına ortalama hız grafiği

Bu grafikler doğrultusunda SAC+Gürültü+EMA modelinin eğitimin sonlarına doğru CTE değerini diğer modellere kıyasla daha etkin bir şekilde minimize etmeyi başardığı gözlenmiştir. Buna karşın SAC+Gürültü+WMA modelinin eğitim süreci boyunca CTE değerini minimize etmekte belirgin bir zorluk yaşadığı ve hedef hız olan "1 birim"

değerine yakınsamakta diğer modellere kıyasla daha çok zorlandığı açıkça görülmektedir. SAC+Gürültü+EMA modelinin CTE minimizasyonundaki bu başarısı, EMA filtresinin, WMA filtresine göre daha az kısıtlayıcı bir pürüzsüzleştirme sağlayarak şerit takibi için gerekli olan daha hassas direksiyon ayarlamalarına olanak tanımasıyla açıklanabilmektedir. Bu bilgiler ışığında, aksiyon filtresi tercihinin, görevin farklı yönleri (CTE hassasiyeti, sürüş pürüzsüzlüğü, hız kontrolü) arasında bir dengeyi beraberinde getirdiğini göstermektedir.

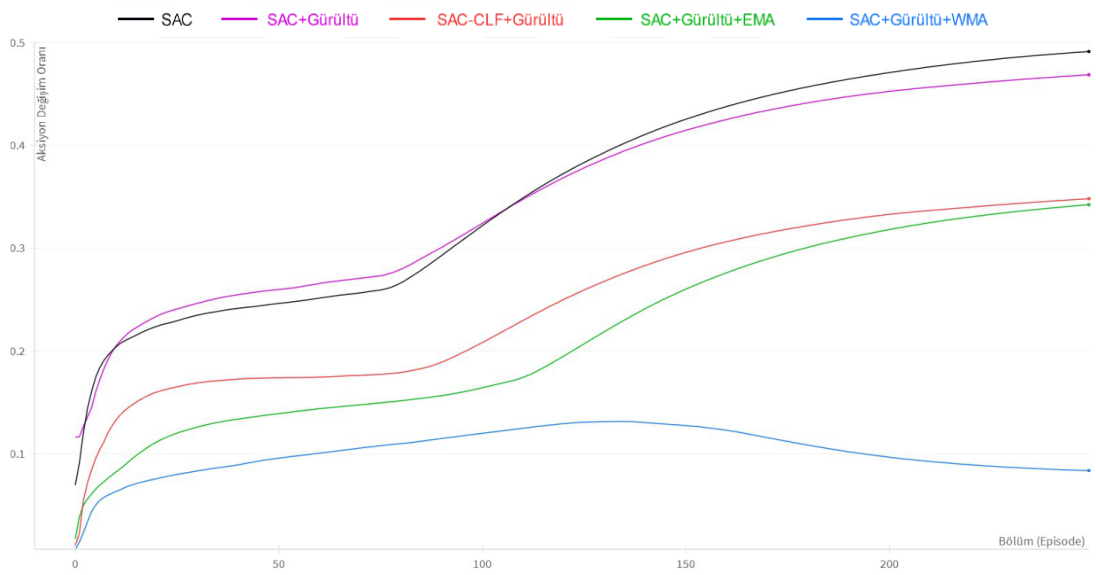
Modellerin eğitim sürecindeki genel performansını değerlendirmek amacıyla bölüm başına ortalama hata grafiği Şekil 4.5'te verilmiştir. Bu doğrultuda eğitim sonlarına doğru SAC+Gürültü+WMA modeli haricindeki diğer modellerin hata oranını birbirlerine yakın değerlerde minimize ettiği gözlenmektedir. SAC+Gürültü+WMA modeli ise hatayı minimize etme konusunda diğer modellere kıyasla daha zayıf bir performans sergilemiştir. Bu durum, WMA filtresinin aksiyon gerçekleştirme sürecindeki esnekliği sınırlandırmasıyla açıklanabilir. Bunun sonucu olarak WMA filtresine sahip SAC+Gürültü+WMA modelinin diğer modellere göre tepkisel davranmakta zorlandığı söylenebilir.




**Şekil 4.5.** Şerit takibi görevi eğitiminde modellerin bölüm başına ortalama hata değişim grafiği

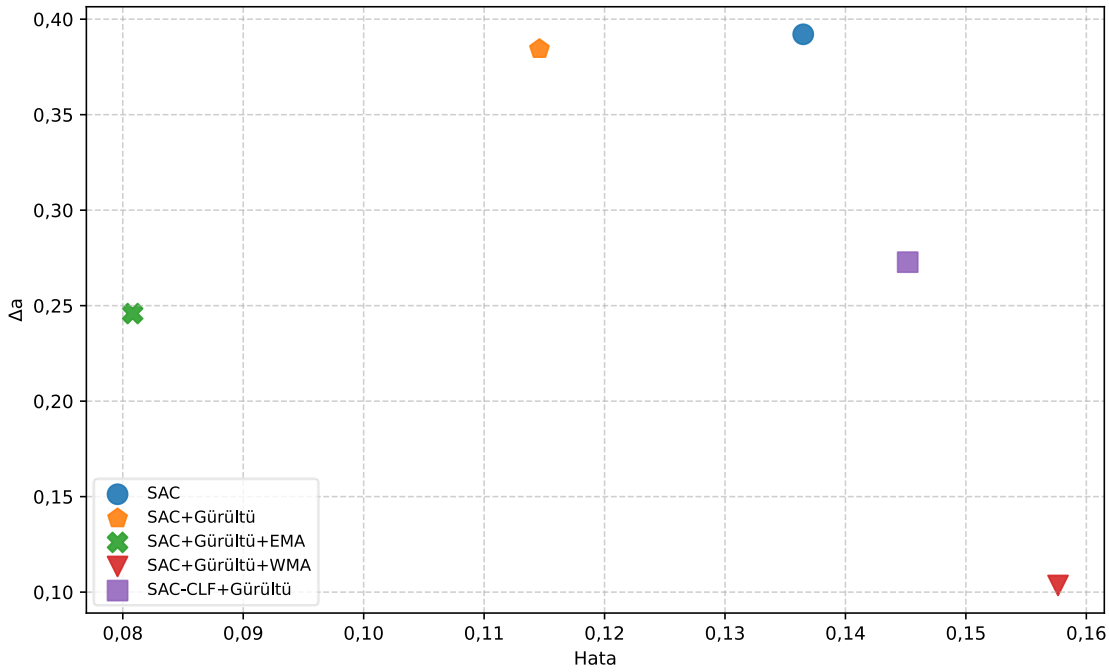
Eğitim süreci boyunca modellerin pürüzsüz sürüş deneyimi performansını incelemek adına ardışık aksiyon değişim değeri gözlenmiştir. Şekil 4.6'da bu değerin değişim grafiği verilmiştir. Bu grafik incelendiğinde ortalama hata grafiğinden farklı bir tablo ortaya çıkmaktadır.

SAC+Gürültü+WMA modelinin diğer tüm modellere kıyasla çok daha düşük bir ardışık aksiyon değişim değerine erişebildiği gözlenmektedir. Bu durum, WMA filtresinin aksiyonları önemli ölçüde pürüzsüzleştirdiğini göstermektedir. Hata grafiği ile kıyaslandığında diğer modellerin hatayı minimize etmek amacıyla ardışık aksiyon değişim değerlerini zamanla artırdığı söylenebilmektedir. SAC+Gürültü+EMA ve SAC-CLF+Gürültü modelleri ardışık aksiyon değişimi açısından eğitim süreci boyunca birbirine yakın performanslar sergilemişlerdir.




**Şekil 4.6.** Şerit takibi görevi eğitiminde modellerin bölüm başına ortalama ardışık aksiyon değişim grafiği

Bu grafikler ışığında Şekil 4.7'de tüm eğitim süreci boyunca toplam hata ortalaması ve toplam aksiyon değişim ortalaması değerleri için dağılım grafiği verilmiştir.




**Şekil 4.7.** Şerit takibi görevi eğitiminde modellerin toplam hata ortalaması ve toplam aksiyon değişim ortalaması değerleri dağılım grafiği

Şekil 4.7'ki dağılım grafiğindeki ardışık aksiyon değişimi ve hata değerlerine ilişkin yüzdelik oranlar Çizelge 4.4'te verilmiştir. Şekil 4.7 ve Çizelge 4.4 birlikte değerlendirildiğinde, eğitim süreci boyunca en düşük toplam hata ortalamasına (%8,08) SAC+Gürültü+EMA modelinin ulaştığı, en düşük toplam ardışık aksiyon değişim ortalamasını (%10,24) SAC+Gürültü+WMA modelinin elde ettiği görülmektedir.

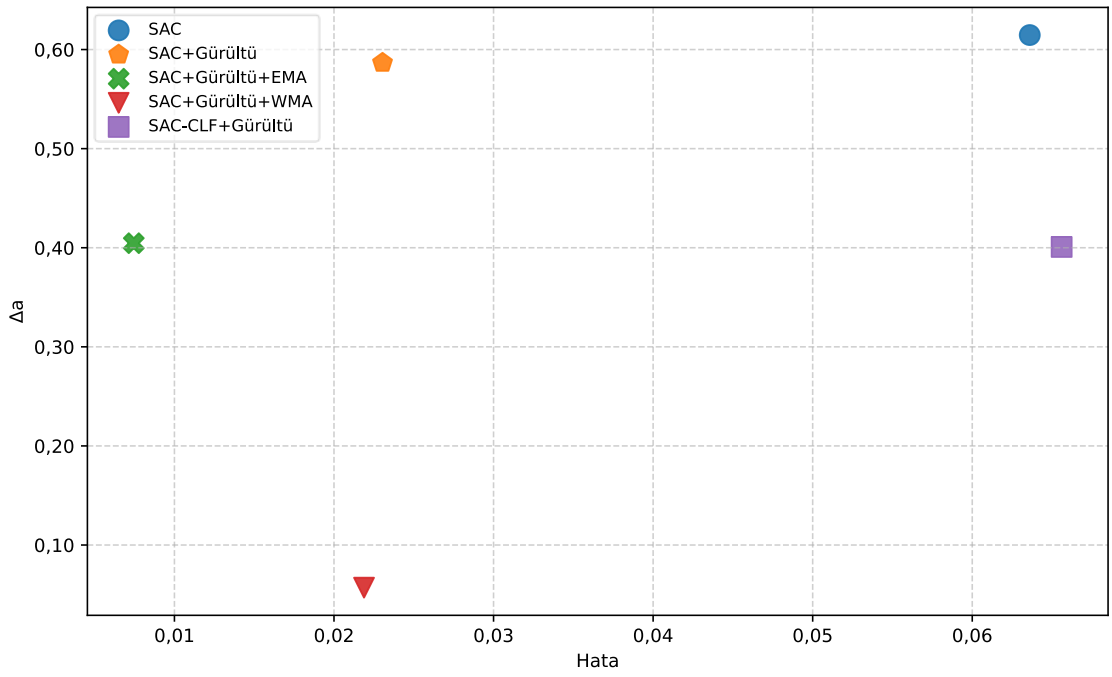
Bununla beraber en yüksek toplam hata ortalamasına (%15,76) SAC+Gürültü+WMA modeli sahiptir. Bu durum SAC+Gürültü+EMA modelinin, SAC+Gürültü+WMA modeline kıyasla hata minimizasyonuna öncelik vererek daha "tepkisel" bir kontrol stratejisi öğrenmeye çalıştığını göstermektedir. SAC+Gürültü+WMA modeli ise pürüzsüz bir kontrol stratejisi geliştirmeye öncelik vermiştir.

**Çizelge 4.4.** Şerit takibi görevi eğitiminde modellerin toplam hata ortalaması ve toplam aksiyon değişim ortalaması yüzdesel değerleri

| Model | Ortalama Δa (%) | Ortalama Hata (%) |
|---|---|---|
| SAC | 39,20 | 13,64 |
| SAC+Gürültü | 38,43 | 11,45 |
| SAC-CLF+Gürültü | 27,27 | 14,51 |
| SAC+Gürültü+EMA | 24,58 | 8,08 |
| SAC+Gürültü+WMA | 10,24 | 15,76 |

### 4.1.2. Test performansları

260 bölümlük eğitim sürecinin tamamlanmasının ardından, eğitilen modellerin nihai performanslarını değerlendirmek amacıyla en fazla 500 adımdan oluşan test sürüşleri gerçekleştirilmiş ve sürüş performansları analiz edilmiştir. Bu performans analizleri için başarı metrikleri olarak ortalama hata ve ortalama ardışık aksiyon değişim değerleri kullanılmıştır. Bu değerlerin dağılımı Şekil 4.8'deki grafikte gösterilmiştir.




**Şekil 4.8.** Şerit takibi görevi testlerinde modellerin ortalama hata ve ortalama aksiyon değişim değerleri dağılım grafiği

Bu grafik doğrultusunda modeller arası ortalama hata değerlerinin, ortalama ardışık aksiyon değişim değerlerine kıyasla daha küçük ölçekte farklılıklar sergilediği dikkat çekmektedir. Şekil 4.8'ki dağılım grafiğindeki test bölümleri için ardışık aksiyon değişimi ve hata değerlerine ilişkin yüzdelik oranlar Çizelge 4.5'te verilmiştir.

**Çizelge 4.5.** Şerit takibi görevi testlerinde modellerin ortalama hata ve ortalama ardışık aksiyon değişim yüzdeleri

| Model | Ortalama Δa (%) | Ortalama Hata (%) |
|---|---|---|
| SAC | 61,46 | 6,36 |
| SAC+Gürültü | 58,66 | 2,30 |
| SAC-CLF+Gürültü | 40,07 | 6,56 |
| SAC+Gürültü+EMA | 40,45 | 0,75 |
| SAC+Gürültü+WMA | 5,69 | 2,18 |

Test sürüşü sonuçlarına göre temel SAC modeli %6,36 hata oranı ve %61,46 ardışık değişim oranı elde ederken, gürültü eklenmiş SAC+Gürültü modeli %2,3 hata oranı ve %58,66 ardışık değişim oranı sergilemiştir. SAC+Gürültü modelinin, aksiyon uzayına eklenen gürültü sayesinde, SAC modeline kıyasla hem oluşan hatayı hem de ardışık aksiyonlardaki değişim oranını daha başarılı bir şekilde minimize ettiği gözlemlenmiştir. Bu durum, aksiyon gürültüsünün sadece eğitim sırasında keşfi artırmakla kalmayıp, aynı zamanda daha sağlam bir politika öğrenilmesine de katkıda bulunduğunu göstermektedir. Gürültüyle eğitilen bir ajan, eğitimde daha çeşitli durumlar ile karşılaşması sayesinde daha iyi genelleme yeteneği kazanarak belirsizliklere karşı daha dirençli bir politika geliştirmiştir. Her iki model için de hata oranları kabul edilebilir düzeyde düşük olmasına rağmen, ardışık aksiyon değişim oranları yüksektir ve birbirine yakındır. Bu durum sonucunda SAC modelinin test sürüş kaydı[1] ile SAC+Gürültü modelinin test sürüş kaydı[2] arasında insan gözlemi açısından belirgin bir pürüzsüzlük farkı deneyimlenmemiştir.

[1] SAC modelinin test sürüş kaydı: https://youtu.be/q2bM7ZJCVP4

[2] SAC+Gürültü modelinin test sürüş kaydı: https://youtu.be/yi-j8GRyi6c

Test sürüşlerinde SAC-CLF+Gürültü modeli %6,56 hata oranı ve %40,07 ardışık aksiyon değişim oranı elde ederken SAC+Gürültü+EMA modeli %0,75 gibi oldukça küçük bir hata oranı ve %40,45 ardışık aksiyon değişim oranı sergilemiştir. SAC+Gürültü+EMA modelinin, tüm modeller içinde en düşük hata oranına sahip olması ve ardışık aksiyon değişim oranında düşüş sağlaması EMA filtresinin pürüzsüzleştirme ve tepkisellik arasında etkili bir denge kurduğunu göstermektedir. Buna karşın SAC-CLF+Gürültü modelinin, SAC+Gürültü+EMA ile benzer bir aksiyon değişim oranına sahip olmasına rağmen daha yüksek bir hata oranı sergilemesi, özelleştirilmiş kayıp fonksiyonunun (CLF), bu özel görevde harici bir EMA filtresi kadar etkili bir hata minimizasyonu sağlayamadığını göstermektedir. Ardışık aksiyon değişim oranlarındaki düşüş SAC-CLF+Gürültü modeli test sürüş kaydı[1] ve SAC+Gürültü+EMA modeli test sürüş kaydında[2] da dikkat çekmektedir.

Test sürüşlerine ilişkin sonuçlarda SAC+Gürültü+WMA modelinin %5,69 ile açık ara en düşük aksiyon değişim oranına sahip olduğu gözlenmektedir. Bununla birlikte %2,18 hata oranı ile SAC+Gürültü+WMA modeli, SAC+Gürültü+EMA modelinden sonra en düşük ikinci hata oranını elde etmiştir.

SAC+Gürültü+WMA modelinin, eğitim sürecinde ödül maksimizasyonu ve hata minimizasyonunda zorlanmasına rağmen, test aşamasında hem oldukça düşük bir ardışık aksiyon değişim oranı hem de düşük bir hata oranına erişerek etkili bir sürüş sergilemesi dikkat çekmektedir.

Test sürüşü video kayıtları[3] SAC+Gürültü+WMA modelinin diğer modellere göre net bir şekilde daha az sarsıntılı ve daha akıcı manevralar yaptığını, yani ardışık aksiyon değişim değerini büyük ölçüde azaltırken ortalama hata oranını da kabul edilebilir bir seviyede tuttuğunu doğrulamaktadır.

[1] SAC-CLF+Gürültü modelinin test sürüş kaydı: https://youtu.be/J6eL30FHcF4

[2] SAC+Gürültü+EMA modelinin test sürüş kaydı: https://youtu.be/Qp418WPwZQE

[3] SAC+Gürültü+WMA modelinin test sürüş kaydı: https://youtu.be/3F8FPMGOuAc

İnsan sürücüler genellikle ani ve keskin manevralardan kaçınarak yumuşak kontrol girdileri üretirler; bu da düşük ardışık aksiyon değişimine karşılık gelir. Dolayısıyla, SAC+Gürültü+WMA modelinin bu performansı, "insan benzeri pürüzsüz sürüş" hedefine en yakın sonucu verdiğini düşündürmektedir.

WMA filtresinin getirdiği öğrenme zorluğunun, ajanın daha "dikkatli" ve "stabil" bir politika geliştirmesine yol açmış olabileceği ve bu politikanın test ortamında daha doğal bir davranışa dönüştüğü söylenebilir. Bu bulgu, WMA gibi güçlü aksiyon pürüzsüzleştirme filtrelerinin, otonom araçlarda sürüş kalitesini ve yolcu konforunu artırmada önemli bir rol oynayabileceğini göstermektedir.

### 4.2. Şerit Takibi ve Engelden Kaçma Deneyleri

Bu bölümde, çalışmanın ikinci aşaması olan ve şerit takibi görevine ek olarak dinamik olmayan bir engelden kaçınma görevini de içeren deneylerin sonuçları sunulmaktadır. Karmaşıklaşan senaryo ile, modellerin adaptasyon ve karar verme yeteneklerini daha zorlu koşullar altında test etmek amaçlanmıştır.

Daha karmaşık olan 'şerit takibi ve engelden kaçma' görevi için oluşturulan modellerde, temel “şerit takibi” deneyindeki hiperparametrelerin büyük bir bölümü tutarlılığı korumak amacıyla sabit tutulmuştur (Çizelge 4.6). Ancak, artan görev karmaşıklığına yanıt olarak bilinçli bir mimari değişiklik yapılmıştır: aktör ve kritik ağlarındaki gizli katmanların boyutu 512 nörondan 1024 nörona çıkarılmıştır. Bu ölçeklendirme kararı, görevin doğasındaki temel bir farklılıktan kaynaklanmaktadır. Statik bir engelin ve bu engeli tespit etmek için kullanılan LIDAR verilerinin sisteme dahil edilmesi, ajanın işlemesi gereken durum uzayını önemli ölçüde zenginleştirmiş ve daha karmaşık karar verme süreçlerini zorunlu kılmıştır. Hem şerit takibini sürdürmek hem de engelden kaçınmak için gereken daha incelikli politikayı öğrenebilmek, daha yüksek parametrik kapasiteye sahip bir sinir ağı gerektirmektedir.

**Çizelge 4.6.** Şerit takibi ve engelden kaçma görevi eğitiminde SAC algoritması için kullanılan parametreler

| Parametre | Değer |
|---|---|
| Yığın Boyutu | 256 |
| Gamma (γ) | 0,97 |
| Tau (τ) | 0,005 |
| Sıcaklık (α) | 0,2 |
| Sinir ağı katmanları boyutu | 1024 (x2) |

Deneyler için oluşturulan modeller 350 bölüm boyunca eğitilmiştir. Bir eğitim bölümü eğitilen ajan izin verilen maksimum şerit merkezine uzaklığı aştığında veya toplamda 300 adım sayısına ulaştığında sonlandırılmıştır.

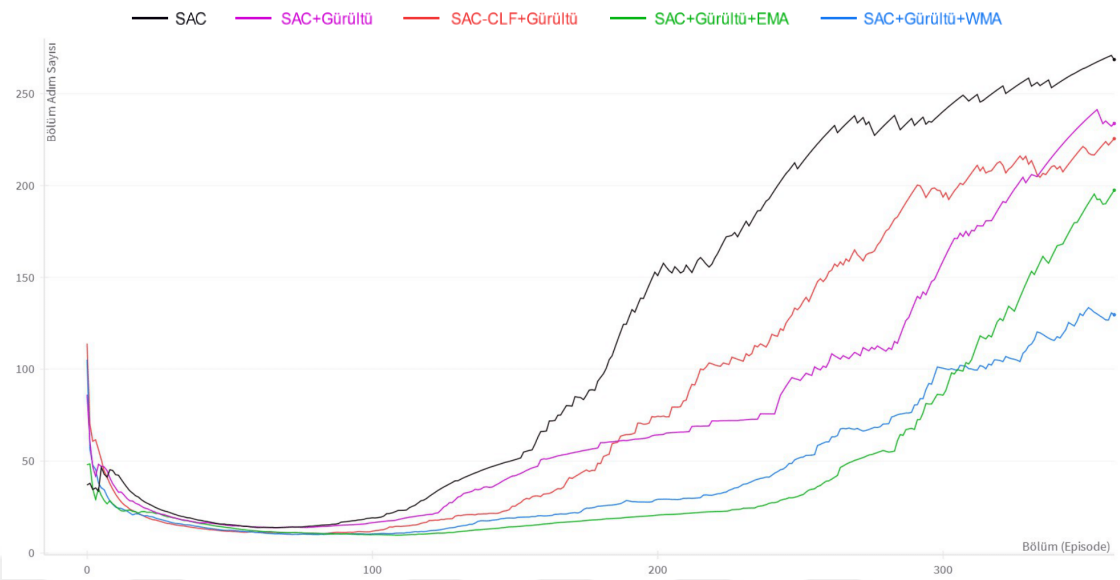
### 4.2.1. Eğitim süreci performansları

Her bir modelin 350 bölümlük eğitim sürecini tamamlama süreleri Çizelge 4.7'de sunulmuştur. Şerit takibi ve engelden kaçma deneyleri için toplam eğitim süresi 14 saat 32 dakika olarak kaydedilmiştir.

**Çizelge 4.7.** Şerit takibi ve engelden kaçma görevi için modellerin eğitim süreleri

| Model | Eğitim Süresi |
|---|---|
| SAC | 4 Saat 11 Dakika |
| SAC+Gürültü | 2 Saat 50 Dakika |
| SAC-CLF+Gürültü | 3 Saat 52 Dakika |
| SAC+Gürültü+EMA | 1 Saat 49 Dakika |
| SAC+Gürültü+WMA | 1 Saat 50 Dakika |

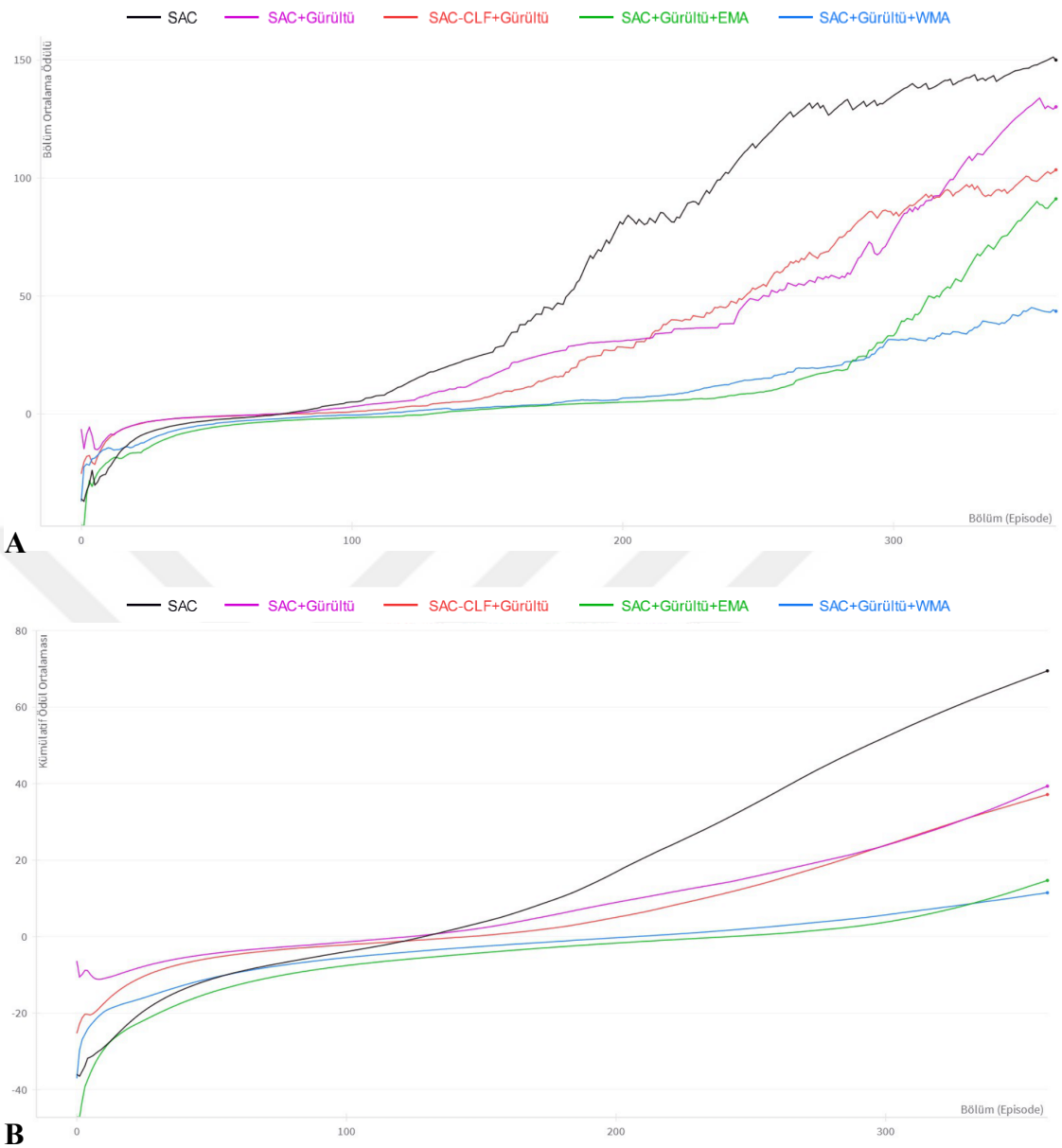
Tüm modellerin eğitim süreci boyunca bölüm başına adım sayısı grafiği Şekil 4.9'da verilmiştir.




**Şekil 4.9.** Şerit takibi ve engelden kaçma görevi eğitiminde modellerin bölüm başına adım sayısı grafiği

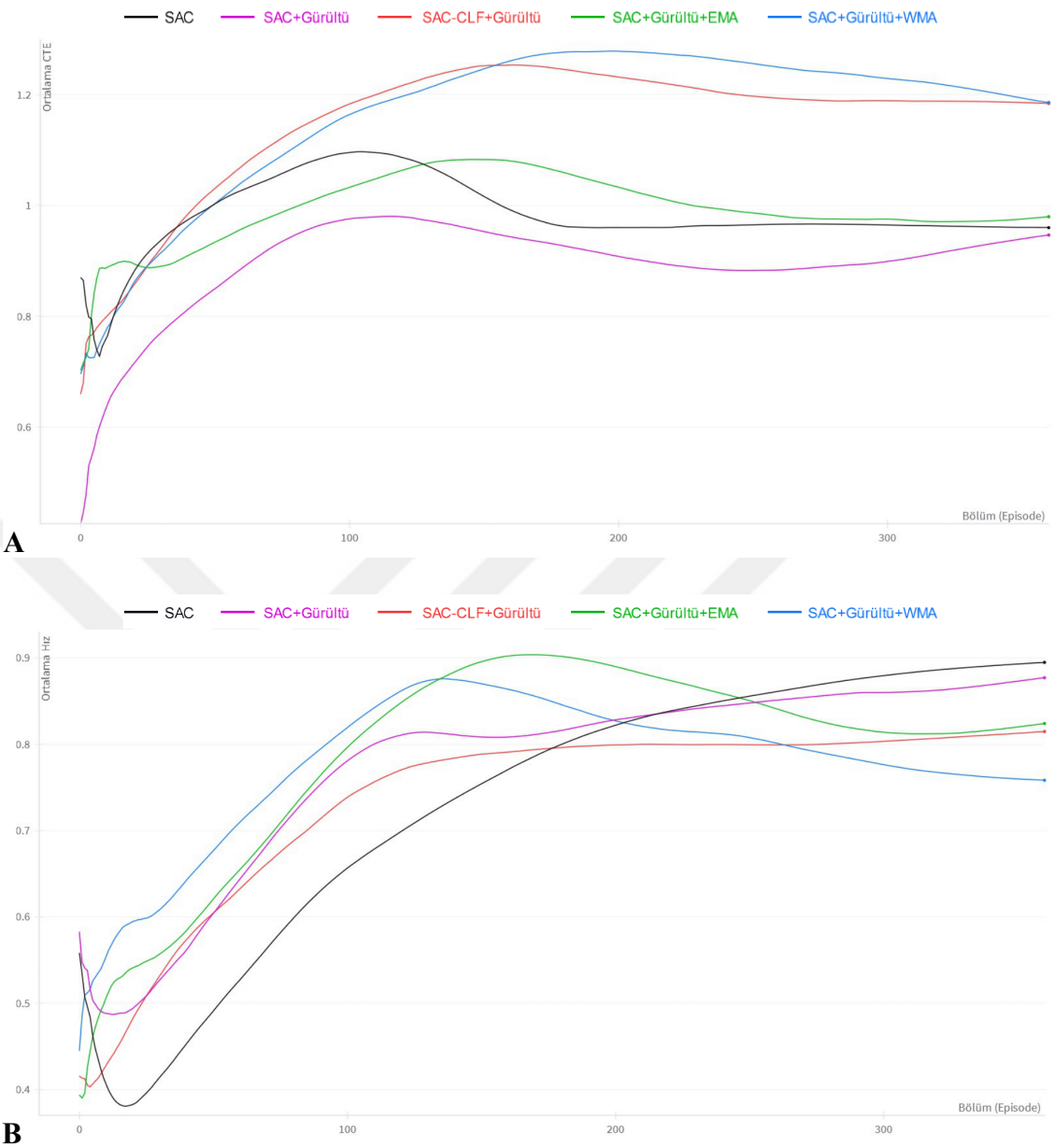
Bu grafik doğrultusunda tüm modellerin, "şerit takibi" deneylerine kıyasla, artan görev karmaşıklığı nedeniyle modellerin 300 adımlık maksimum bölüm uzunluğuna yakınsamakta zorlandığı görülmektedir. Özellikle aksiyon filtreleme modülü kullanan modellerin diğer modellere kıyasla daha sık hata yaparak bölümleri daha erken bitirdiği gözlenmektedir. Bu durum, aksiyon filtrelerinin sebep olduğu kontrol kısıtlamalarının, engelden kaçma gibi daha tepkisel aksiyonlar gerektiren durumlarda öğrenmeyi yavaşlatabileceğini göstermektedir.

Şekil 4.10.A'da bölüm başına toplanan ödül ve Şekil 4.10.B'de kümülatif ödül ortalaması grafikleri sunulmuştur. Bu grafikler doğrultusunda, "şerit takibi" deneylerine paralel olarak, aksiyon filtresine sahip modellerin maksimum ödül miktarına diğer modellere göre daha geç yakınsamaya başladığı görülmektedir. Ayrıca, tüm modeller için ödül miktarlarında, ortam karmaşıklığının artmasına bağlı olarak genel bir düşüş gözlenmektedir. Sinir ağı boyutunun artırılmasına rağmen, bölüm adım sayılarındaki ve ödül miktarlarındaki yavaş artışlar, bu daha karmaşık görevde optimum performansa ulaşmak için daha uzun eğitim sürelerinin gerekebileceğine işaret etmektedir.




**Şekil 4.10** Şerit takibi ve engelden kaçma görevi eğitiminde modellerin ödül değişim grafikleri
**A)** Bölüm başına toplam ödül grafiği **B)** Kümülatif ödül ortalaması grafiği

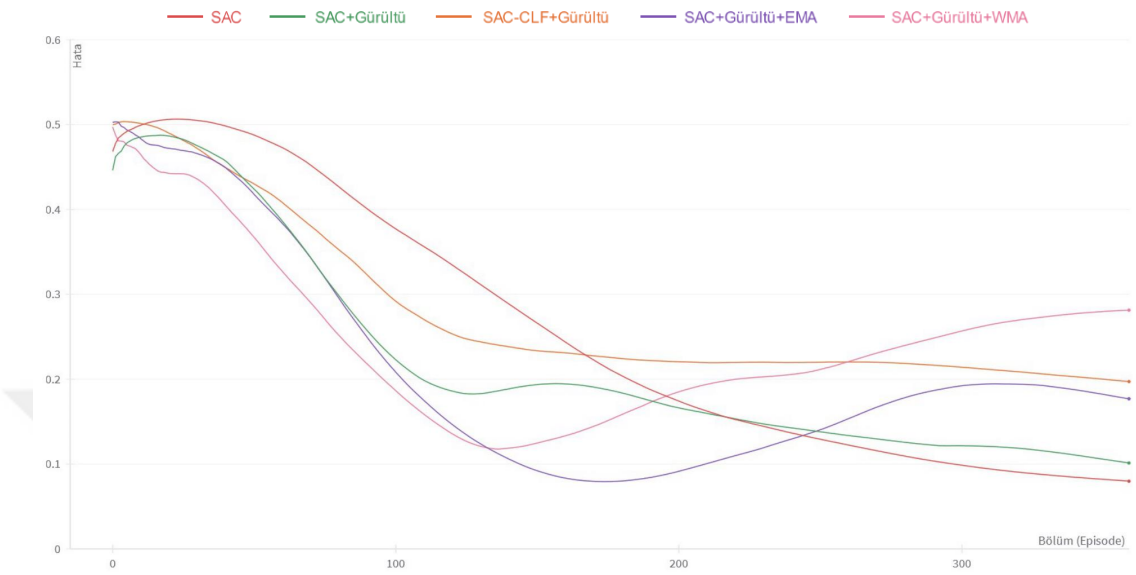
Modellerin eğitim süreci boyunca sergiledikleri bölüm başına ortalama CTE değeri grafiği Şekil 4.11.A'da ve ortalama hız değerleri grafiği Şekil 4.11.B'de verilmiştir. Bu grafiklerde SAC+Gürültü+WMA ve SAC-CLF+Gürültü modellerinin diğer modellere kıyasla daha yüksek CTE değerlerine sahip olduğu gözlenmektedir. Özellikle SAC+Gürültü+WMA modelinin eğitim sürecinde hedef hız olan 1 birim değerine ulaşmakta diğer modellere kıyasla daha zorlandığı görülmektedir.




**Şekil 4.11.** Şerit takibi ve engelden kaçma görevi eğitiminde modellerin CTE ve hız değişim grafikleri
**A)** Bölüm başına ortalama CTE grafiği **B)** Bölüm başına ortalama hız grafiği

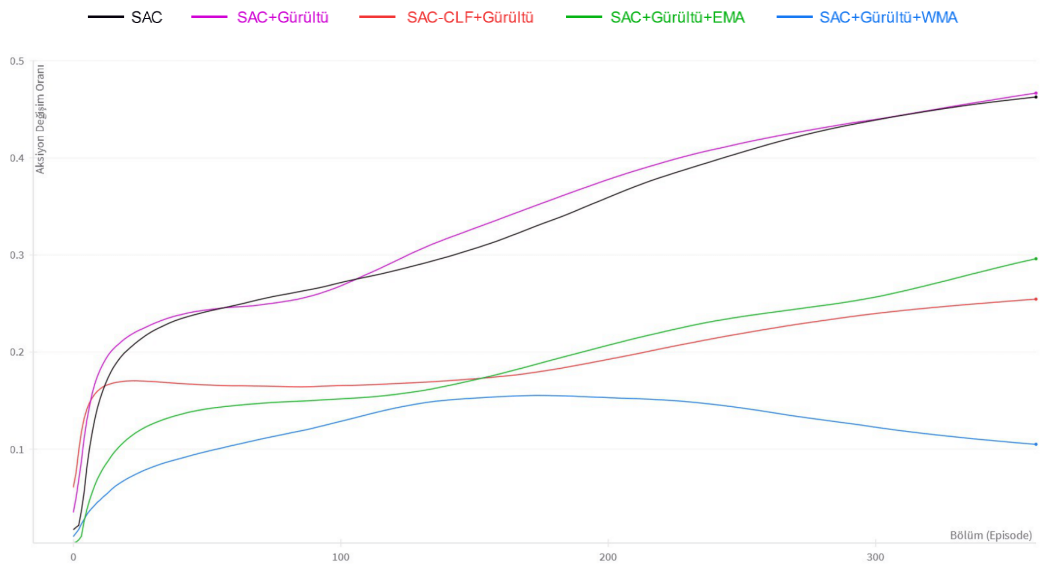
Modellerin eğitim sürecindeki genel hata performansını değerlendirmek amacıyla kullanılan bölüm başına ortalama hata değerlerine ilişkin grafik Şekil 4.12'de sunulmuştur. Bu doğrultuda SAC+Gürültü+WMA modelinin eğitimin sonlarına doğru hata oranını minimize etmekte diğer modellere kıyasla daha fazla zorlandığı görülmektedir. En düşük hata oranına temel SAC modelinin ulaştığı gözlenirken,

eğitimin sonlarına doğru SAC-CLF+Gürültü ve SAC+Gürültü+EMA modellerinin birbirine yakın hata değerleri sergilediği görülmüştür.



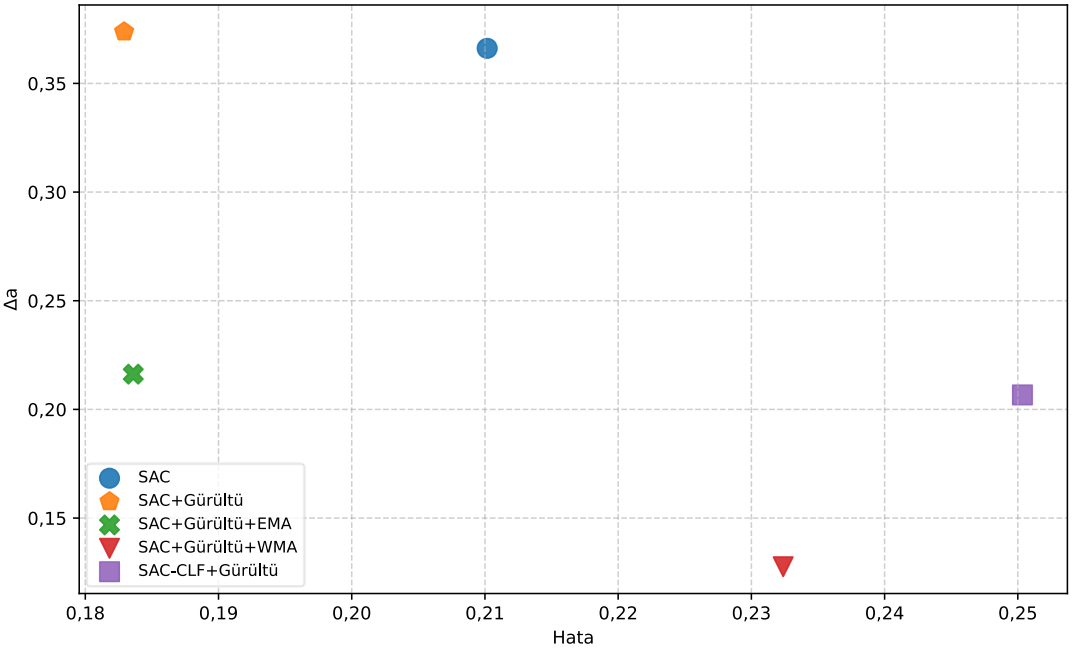

**Şekil 4.12.** Şerit takibi ve engelden kaçma görevi eğitiminde modellerin bölüm başına ortalama hata grafiği

Eğitim süreci boyunca tüm modellerin sürüş pürüzsüzlüğünü incelemek amacıyla bölüm başına ortalama ardışık aksiyon değişim değerleri grafiği Şekil 4.13'te sunulmuştur.




**Şekil 4.13.** Şerit takibi ve engelden kaçma görevi eğitiminde modellerin bölüm başına ortalama ardışık aksiyon değişim grafiği

Bu grafik doğrultusunda SAC ve SAC+Gürültü modelleri için ardışık aksiyon değişim değerleri eğitim boyunca birbirine yakın ve diğer modellere kıyasla daha yüksek seviyelerde seyretmiştir. Bu durum, aksiyon filtresiz modellerin engelden kaçma ve şerit takibi görevlerini yerine getirirken daha tepkisel ve potansiyel olarak daha sarsıntılı manevralar yaptığını göstermektedir. En düşük aksiyon değişim değerine ise SAC+Gürültü+WMA modelinin ulaştığı görülmektedir. SAC+Gürültü+EMA ve SAC-CLF+Gürültü modelleri ise bu metrik açısından benzer davranışlar sergileyerek eğitimi yakın değerlerde sonlandırmışlardır.




**Şekil 4.14.** Şerit takibi ve engelden kaçma görevi eğitiminde modellerin ortalama toplam hata ve ortalama toplam aksiyon değişim değerleri dağılım grafiği

Şekil 4.14'te eğitim süreci boyunca elde edilen ortalama hata ve ortalama aksiyon değişim değerlerinin dağılımı sunulmuştur. Her bir modelin eğitim süreci için toplam hata ve toplam ardışık aksiyon değişim ortalamalarına ait yüzdelik değerler Çizelge 4.8'de detaylandırılmıştır. Bu dağılım grafiği ve çizelge, modeller arasında hata oranlarında görece küçük farklılıklar olduğunu, ancak ardışık aksiyon değişimi açısından en iyi değeri %12,75 ile SAC+Gürültü+WMA modelinin elde ettiğini göstermektedir. Bununla beraber "şerit takibi" deneylerinden farklı olarak en düşük toplam hata ortalamasına %18,29 ile SAC+Gürültü modelinin ve ardından %18,36 ile

SAC+Gürültü+EMA filtresinin sahip olduğu görülmüştür. Burada "şerit takibi" deney sonuçların farklı gerçekleşen küçük detaylar, karmaşıklaşan göreve bağlı olarak bazı modellerin diğerlerine kıyasla beklenmedik şekilde zorlanabileceğini görülmektedir. Model performanslarının daha iyi analiz edilmesi için test sürecinin de incelenmesi gerekmektedir.

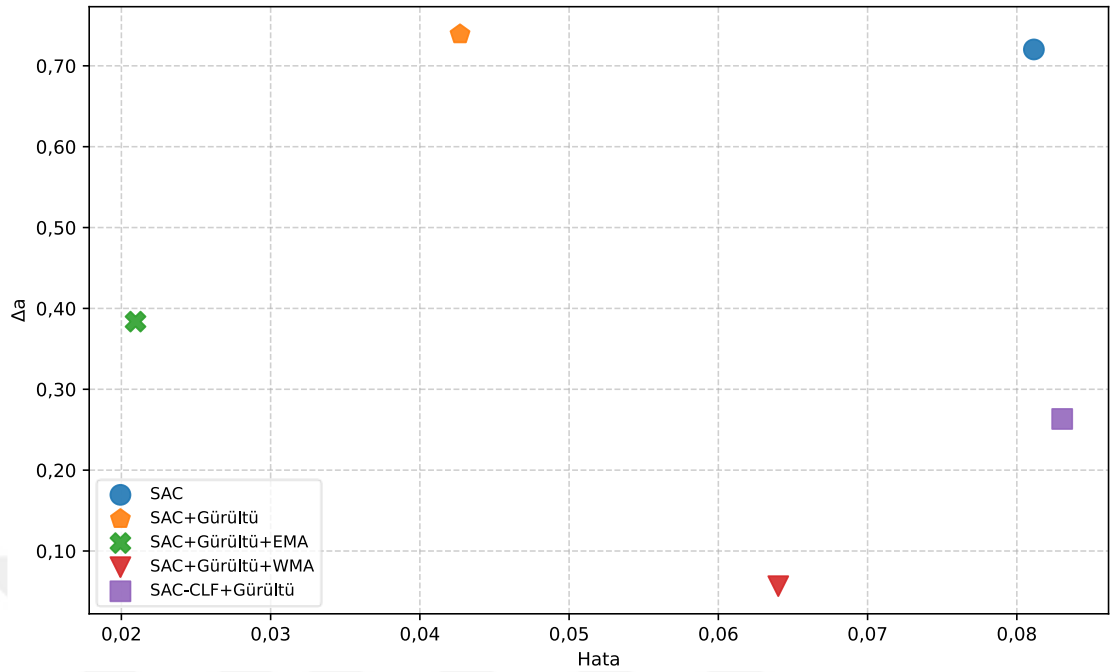
**Çizelge 4.8.** Şerit takibi ve engelden kaçma görevi eğitiminde modellerin ortalama toplam hata ve ortalama toplam ardışık aksiyon değişim yüzdeleri

| Model | Ortalama Δa (%) | Ortalama Hata (%) |
|---|---|---|
| SAC | 36,61 | 21,01 |
| SAC+Gürültü | 37,37 | 18,29 |
| SAC-CLF+Gürültü | 20,66 | 25,03 |
| SAC+Gürültü+EMA | 21,62 | 18,36 |
| SAC+Gürültü+WMA | 12,75 | 23,23 |

### 4.2.1. Test performansları

Toplam 350 bölümlük eğitim sürecinin tamamlanmasının ardından, eğitilen modellerin nihai performanslarını değerlendirmek amacıyla test sürüşleri gerçekleştirilmiştir. Bu testlerde, eğitilen modellerin en fazla 500 adımdan oluşan bir bölüm için sürüş performansları analiz edilmiştir. Bu performans analizleri için başarı metrikleri olarak hata değerleri, ardışık aksiyon değişim değerleri kullanılmıştır. Ayrıca sürüşlerin nitel bir değerlendirmesi için video kayıtları alınmıştır. Tüm sistem modelleri test sürüşünü başarı ile tamamlamıştır.

Test sürüşleri sonunda elde edilen ortalama hata ve ortalama ardışık aksiyon değişimi değerlerinin dağılımı Şekil 4.15'teki grafikte gösterilmiştir. Bu grafik doğrultusunda, "şerit takibi" deneylerinde olduğu gibi, modeller arası ortalama hata değerlerinin, ortalama ardışık aksiyon değişim değerlerine kıyasla daha küçük ölçekte farklılıklar sergilediği görülmüştür.




**Şekil 4.15.** Şerit takibi ve engelden kaçma görevi testlerinde modellerin ortalama hata ve ortalama ardışık aksiyon değişim değerleri dağılım grafiği

Şekil 4.15'te verilen her bir modelin test bölümü için ortalama hata ve ortalama ardışık aksiyon değişim oranlarına ait yüzdesel değerler Çizelge 4.9'da sunulmuştur. Temel SAC modeline kıyasla, SAC+Gürültü modelinin hata değerini daha etkin bir şekilde minimize etmesi (%8,11'e karşılık %4,26), aksiyon gürültüsü ekleme modülünün bu karmaşık görevde daha sağlam bir politika öğrenilmesine katkı sağladığını göstermektedir. Bununla birlikte, her iki modelin de ardışık aksiyon değişim oranları oldukça yüksektir, bu da filtresiz yaklaşımların pürüzsüzlükten ziyade tepkiselliğe öncelik verdiğini göstermektedir. Buradaki sonuçlar sebebiyle SAC modelinin test sürüş kaydı[1] ile SAC+Gürültü modelinin test sürüş kaydı[2] arasında insan gözlemi açısından belirgin bir pürüzsüzlük farkı deneyimlenmemiştir. Ancak her iki model için, "şerit takibi" deneyleri test görüntülerine kıyasla görevi tamamlamakta daha çok zorlandıkları söylenebilir.

[1] SAC modelinin test sürüş kaydı: https://youtu.be/hOeU_LdGnWM

[2] SAC+Gürültü modelinin test sürüş kaydı: https://youtu.be/u4oKBmLUo_c

**Çizelge 4.9.** Şerit takibi ve engelden kaçma görevi testlerinde modellerin ortalama hata ve ortalama ardışık aksiyon değişim yüzdeleri

| Model | Ortalama Δa (%) | Ortalama Hata (%) |
|---|---|---|
| SAC | 72,02 | 8,11 |
| SAC+Gürültü | 73,90 | 4,26 |
| SAC-CLF+Gürültü | 26,33 | 8,30 |
| SAC+Gürültü+EMA | 38,36 | 2,09 |
| SAC+Gürültü+WMA | 5,64 | 6,40 |

Test sonuçlarına göre en düşük hata oranını (%2,09) SAC+Gürültü+EMA modeli elde ederken, en düşük ardışık aksiyon değişim değerini (%5,64) ise SAC+Gürültü+WMA modeli sağlamıştır. Bu sonuçlar, Bölüm 4.1'deki "şerit takibi" deneyleriyle tutarlılık göstermekte ve aynı çıkarımları sağlamaktadır. SAC-CLF+Gürültü sistem modeli, orijinal kayıp denklemine sahip SAC ve SAC+Gürültü modellerine göre ardışık aksiyon değişim oranını önemli ölçüde azaltmayı başarmıştır (%26,33). İlginç bir şekilde, bu deneylerde SAC-CLF+Gürültü modeli, aksiyon değişim değeri açısından SAC+Gürültü+EMA modelinden (%38,36) daha iyi bir performans sergilemiştir. Ancak, hata oranı (%8,30) EMA ve WMA filtreli modellere göre daha yüksek kalmıştır. Bu durum, özelleştirilmiş kayıp fonksiyonunun pürüzsüzlüğe katkı sağlayabildiğini, ancak bu özel görevde hata minimizasyonu açısından modüler filtreli modeller kadar etkili olamadığını göstermektedir.

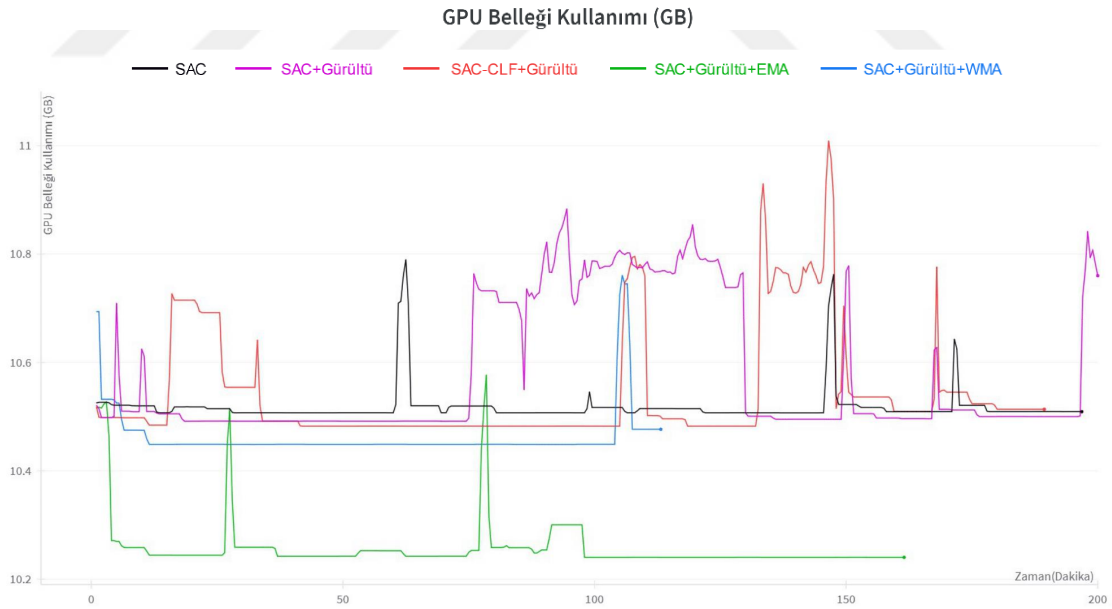
Test sürüşü görüntüleri incelendiğinde SAC-CLF+Gürültü[1] ve SAC+Gürültü+EMA[2] modellerinin ardışık aksiyon değişim oranlarındaki başarısı gözlenmektedir. Bununla birlikte SAC+Gürültü+WMA modeline ait görüntü kaydı[3] ve test sonuçları incelendiğinde, bu modelin diğer modellere kıyasla ardışık aksiyon değişim değerini büyük ölçüde azalttığı gözlenmiştir. Ayrıca SAC+Gürültü+WMA modeli ortalama hata oranını da kabul edilebilir bir seviyede tutarak başarısını ispatlamıştır.

[1] SAC-CLF+Gürültü modelinin test sürüş kaydı: https://youtu.be/7TJKru7aWqE
[2] SAC+Gürültü+EMA modelinin test sürüş kaydı: https://youtu.be/jfAU_vDxXNQ
[3] SAC+Gürültü+WMA modelinin test sürüş kaydı: https://youtu.be/itq300CbA68

Tüm modellerin test görüntüleri incelendiğinde, her bir modelin parkur içerisinde özellikle sağ ve sola dönüşlü kısımlarda şerit merkezine yakınlığı minimize etmekte zorlandığı görülmüştür. Bu durum hata oranının artmasına neden olmaktadır. Engelden kaçma aşamasında ise yine aksiyon filtreli modellerin çok daha pürüzsüz bir davranış sergilediği ve insan sürücülerin performansına yakınlık gösterdiği söylenebilmektedir. Yine de "şerit takibi ve engelden kaçma" görevleri için daha uzun süre eğitimin ve iyileştirmelerin çok daha iyi performans sağlayabileceği ifade edilebilir.

### 4.3. Donanımsal Performanslar

Deneylerin gerçekleştirildiği donanımsal sistemde ekran kartı (GPU) ve rastgele erişimli bellek (RAM) performansları izlenmiştir. Bu başlık altında donanımın yeterliliği ve gelecek çalışmalar için iyileştirmeler değerlendirilmektedir. Şekil 4.16'da "şerit takibi" deneyleri için gerçekleştirilen eğitim süreçlerine ilişkin GPU kullanım miktarı (A) ve yüzdesi (B) grafikleri verilmiştir.



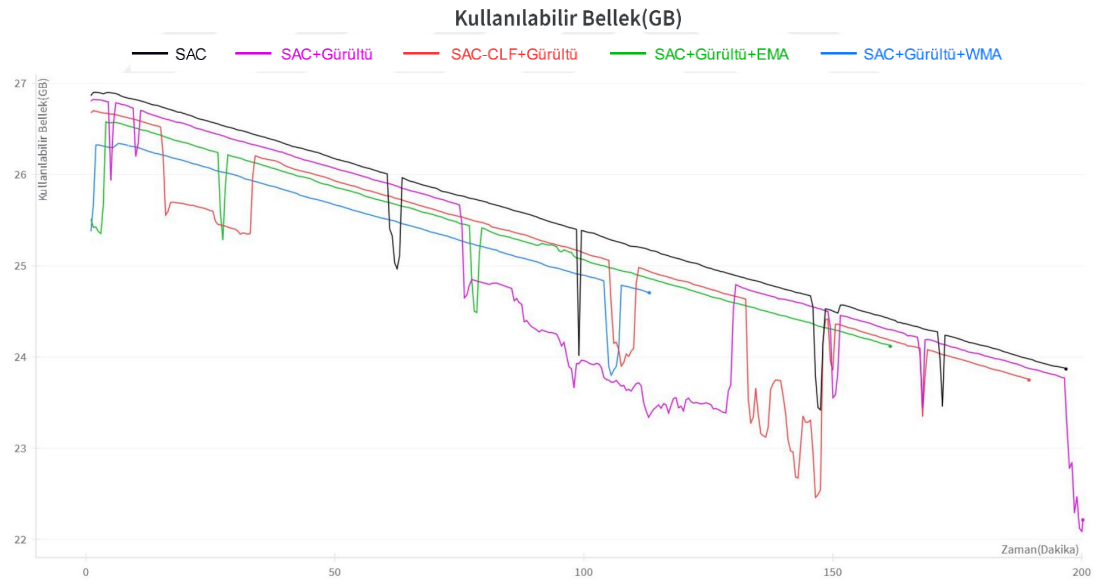

**Şekil 4.16.** Şerit takibi görevi eğitim süreci için GPU belleği kullanımı

Tüm modellerin eğitim sürecinde GPU belleği kullanımının 10,2-11,1 GB aralığında seyrettiği ve bu durumun, mevcut 12 GB GPU belleğinin yaklaşık %85'ine tekabül ettiği tespit edilmiştir.

Eğitim sonunda toplanan toplam deneyim sayıları Çizelge 4.10'da sunulmaktadır. Eğitim süreci boyunca toplanan bu deneyimler hızlı erişim sağlanabilmek için rastgele erişimli bellekte (RAM) tutulmaktadır. Buna ilişkin olarak şerit takibi deneyleri için eğitim sürecinde kullanılabilir RAM grafiği Şekil 4.17'de sunulmuştur.

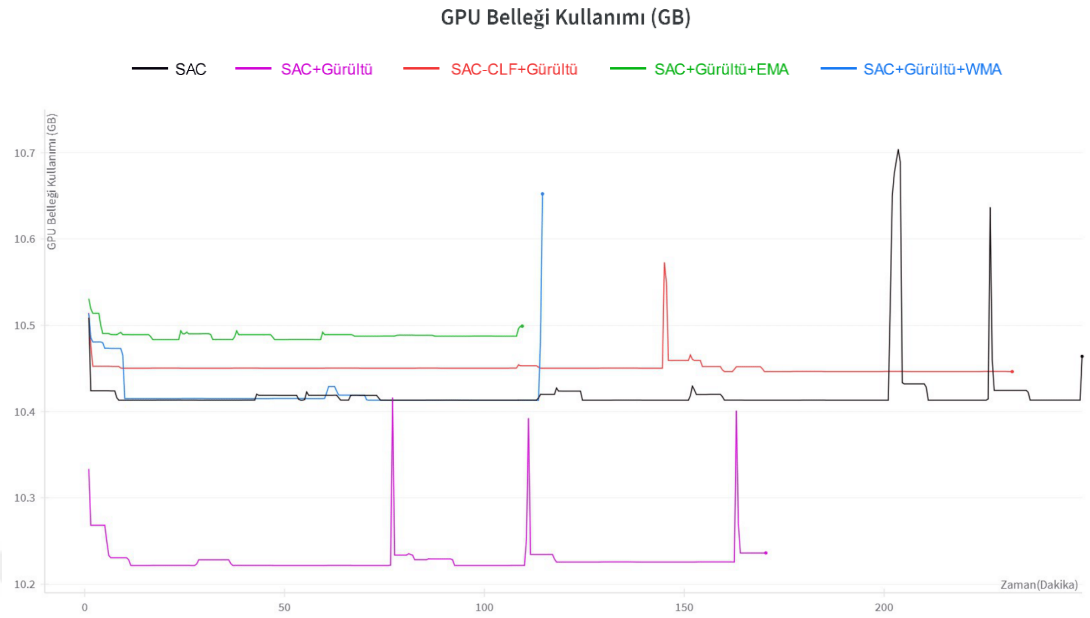
**Çizelge 4.10.** Şerit takibi görevi eğitim sürecinde toplanan deneyim adedi

| Model | Eğitim sonu toplanan deneyim adedi |
|---|---|
| SAC | 53 877 |
| SAC+Gürültü | 54 131 |
| SAC-CLF+Gürültü | 50 825 |
| SAC+Gürültü+EMA | 44 258 |
| SAC+Gürültü+WMA | 30 804 |



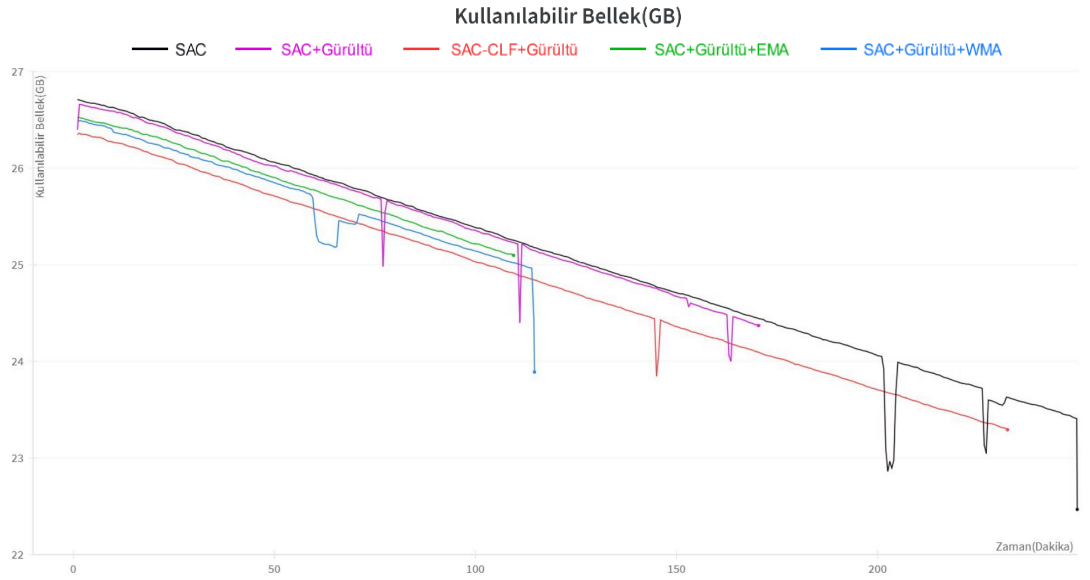

**Şekil 4.17.** Şerit takibi görevi eğitim süreci için kullanılabilir RAM grafiği

Şekil 4.18'de "şerit takibi ve engelden kaçma" deneyleri için gerçekleştirilen eğitim süreçlerine ilişkin GPU kullanım miktarı (A) ve yüzdesi (B) grafikleri verilmiştir. Bu bilgiler ışığında tüm modeller için eğitim sürecinde 10.2-10.8 GB aralığında bir GPU hafızası kullandığı gözlenmiştir. Bu değer aralığı doğrultusunda GPU belleğinin yaklaşık %85 oranında kullanıldığı görülmektedir.




**Şekil 4.18.** Şerit takibi ve engelden kaçma görevi eğitim süreci için GPU belleği kullanımı

Şekil 4.19, şerit takibi ve engelden kaçma deneylerinin eğitim sürecindeki kullanılabilir RAM miktarını göstermektedir. Bu deneylerin eğitimi sonunda toplanan toplam deneyim sayıları ise Çizelge 4.11'de özetlenmiştir.




**Şekil 4.19.** Şerit takibi ve engelden kaçma görevi eğitim süreci için kullanılabilir RAM grafiği

**Çizelge 4.11.** Şerit takibi ve engelden kaçma görevi eğitim sürecinde toplanan deneyim adedi

| Model | Eğitim sonu toplanan deneyim adeti |
|---|---|
| SAC | 54 432 |
| SAC+Gürültü | 36 579 |
| SAC-CLF+Gürültü | 39 752 |
| SAC+Gürültü+EMA | 23 260 |
| SAC+Gürültü+WMA | 20 320 |

Elde edilen veriler, kullanılan simülasyon ortamının ve tek boyutlu durum uzayı vektörünün mevcut 12 GB GPU belleği için yeterli bir yük oluşturduğunu göstermektedir. Ancak daha karmaşık ortamlar ve simülatörler göz önüne alınarak gelecekteki çalışmalar için yetersiz olacağı anlaşılmaktadır. Öte yandan 32 GB RAM boyutunun tek boyutlu durum uzayı vektörü ve toplanan deneyim adedi için yeterli geldiği tespit edilmiştir.

# 5. TARTIŞMA ve SONUÇ

Bu bölümde, "Derin Takviyeli Öğrenme Tabanlı Otonom Sürüş" başlıklı tez çalışmasında elde edilen bulgular, çalışmanın özgün katkıları ve sınırlılıkları tartışılmış ve gelecek araştırmalar için öneriler sunulmuştur.

Otonom araç teknolojilerinde insan benzeri sürüş davranışlarının elde edilmesi, yalnızca teknik bir başarıdan öte, aynı zamanda yol güvenliği, toplumsal kabul edilebilirlik ve yolcu konforu gibi çok boyutlu hedefler açısından da kritik öneme sahiptir. Çalışmanın temel amacı, otonom araçlar için insan benzeri, akıcı ve güvenli sürüş davranışları sergileyebilen modüler bir derin takviyeli öğrenme sistemi geliştirmekti. Bu hedef doğrultusunda, SAC algoritması temel alınmış; yüksek boyutlu sensör verilerinin işlenmesi için oto-kodlayıcılar ve sürüş pürüzsüzlüğünü artırmak amacıyla çeşitli aksiyon filtreleme teknikleri, spesifik olarak Üstel Hareketli Ortalama (EMA) ve Ağırlıklı Hareketli Ortalama (WMA) filtreleri, kullanılmıştır. İnsan benzeri sürüşün modellenmesi; yol takibi, çevresel etkileşim, konforlu manevra üretimi ve güvenli sürüş performansı açısından büyük önem taşımaktadır. Bu bağlamda, geliştirilen sistemin, özellikle aksiyon filtreleri aracılığıyla, sürüş kararlarının doğallığını ve sürekliliğini ne ölçüde iyileştirdiği, çalışmanın temel odak noktasını oluşturmuştur.

Bu çalışmanın en önemli bulgularından biri, sisteme entegre edilen aksiyon filtreleme modüllerinin, otonom aracın sürüş karakteristiği üzerinde belirleyici bir rol oynamasıdır. Özellikle Ağırlıklı Hareketli Ortalama (WMA) filtresi kullanılan SAC+Gürültü+WMA modelinin, hem "şerit takibi" hem de daha karmaşık olan "şerit takibi ve engelden kaçma" görevlerinde, sürüş pürüzsüzlüğünü ifade eden ortalama ardışık aksiyon değişimi ($\Delta a$) metriğinde diğer tüm modellere kıyasla açık ara en iyi performansı sergilediği gözlemlenmiştir.

"Şerit takibi" deneylerinin test aşamasında (bkz. Çizelge 4.5, Şekil 4.8), SAC+Gürültü+WMA modeli %5,69 gibi oldukça düşük bir $\Delta a$ değerine ulaşırken, ortalama hata oranı %2,18 olarak kaydedilmiştir. Bu, filtresiz SAC (%61,46 $\Delta a$) ve sadece gürültü eklenmiş SAC+Gürültü (%58,66 $\Delta a$) modellerine göre pürüzsüzlükte

kayda değer bir iyileşmeye işaret etmektedir. Benzer şekilde, "şerit takibi ve engelden kaçma" gibi daha dinamik ve tepkisel manevralar gerektiren görevde (bkz. Çizelge 4.9, Şekil 4.15), SAC+Gürültü+WMA modeli %5,64 ile yine en düşük Δa değerini sergilemiş ve hata oranını (%6,40) makul bir düzeyde tutmuştur. Bu tutarlı performans, WMA filtresinin, farklı karmaşıklık seviyelerindeki görevlerde aracın sarsıntısız ve akıcı hareket etmesine, dolayısıyla insan sürüşüne daha yakın bir profil sergilemesine önemli katkı sağladığını göstermektedir.

Üstel Hareketli Ortalama (EMA) filtresi kullanılan SAC+Gürültü+EMA modeli ise, hata minimizasyonu ile sürüş pürüzsüzlüğü arasında dengeli bir durum sergilemiştir. "Şerit takibi" testlerinde %0,75 ile tüm modeller arasında en düşük hata oranını elde ederken, Δa değeri (%40,45) WMA filtresine göre daha yüksek kalsa da filtresiz modellere kıyasla önemli bir iyileşme sağlamıştır (bkz. Çizelge 4.5). Daha karmaşık olan "şerit takibi ve engelden kaçma" testlerinde de %2,09 ile yine en düşük hata oranını vermiştir (bkz. Çizelge 4.9). Bu bulgular, EMA filtresinin, özellikle hataya daha duyarlı senaryolarda tepkiselliği korurken pürüzsüzlüğe de katkıda bulunabildiğini göstermektedir. WMA filtresinin eğitim sürecinde ödülleri maksimize etme ve şerit merkezine olan uzaklığı (CTE) minimize etme konusunda EMA filtresine kıyasla daha yavaş bir öğrenme eğrisi sergilemesi, WMA'nın getirdiği güçlü pürüzsüzleştirme etkisinin, ajanın anlık optimal aksiyonları seçme esnekliğini bir miktar kısıtlayabileceğini, ancak test aşamasında bu durumun daha stabil ve akıcı bir sürüşe dönüştüğünü düşündürmektedir. Dolayısıyla, aksiyon filtresi tercihi, görevin öncelikleri – örneğin, yolcu konforu için azami pürüzsüzlük mü, yoksa ani durumlara hızlı tepki için düşük hata mı – arasında bir denge kurulmasını gerektirmektedir.

Öte yandan, aktör ağının kayıp fonksiyonuna doğrudan pürüzsüzlük terimi eklenerek oluşturulan SAC-CLF+Gürültü modeli, filtresiz modellere göre Δa değerlerinde bir miktar iyileşme sağlamış olsa da (örneğin, şerit takibi testinde %40,07 Δa), genel sürüş kalitesi ve hata minimizasyonu açısından harici EMA ve WMA filtreli modeller kadar tutarlı ve etkili bir performans sergileyememiştir. Bu durum, aksiyonların ajanın temel öğrenme mekanizmasından bağımsız olarak harici bir modül tarafından filtrelenmesinin,

bu çalışmada incelenen görevler için daha esnek ve sonuç odaklı bir yaklaşım olabileceğine işaret etmektedir.

Bu çalışmada benimsenen modüler mimari, sistemin farklı işlevsel bileşenlerinin (algısal veri işleme, karar verme ve aksiyon filtreleme) bağımsız olarak geliştirilmesine ve esnek bir şekilde entegre edilmesine olanak tanımıştır. Bu yaklaşımın en önemli avantajlarından biri, yüksek boyutlu sensör verilerinin ön işlenmesi için oto-kodlayıcıların kullanılması ve bu işlenmiş verilerin SAC tabanlı karar verme mekanizmasına girdi olarak sunulması olarak gösterilebilir. Oto-kodlayıcılar, ham sensör verilerini daha düşük boyutlu, ancak temel bilgiyi koruyan özellik vektörlerine dönüştürerek SAC algoritmasının eğitim verimliliğini ve hızını artırmada kritik bir rol oynamıştır. Bu sadeleştirme, özellikle hesaplama kaynaklarının sınırlı olduğu durumlarda ve karmaşık görevlerde öğrenme sürecini kolaylaştırmaktadır.

Temel öğrenme algoritması olarak seçilen Soft Aktör-Kritik algoritması aksiyon filtreleme modüllerinin getirdiği karmaşık kontrol dinamiklerine rağmen hem şerit takibi hem de engelden kaçma gibi zorlu otonom sürüş görevlerinde başarılı ve kararlı sürüş politikaları öğrenebildiğini kanıtlamıştır. Ajanın, filtrelerin uyguladığı kısıtlamalar altında dahi çevreyle uyumlu ve kontrollü sürüş davranışları geliştirebilmesi, SAC algoritmasının karmaşık ve sürekli aksiyon uzaylarındaki etkinliğini bir kez daha teyit etmektedir. Ayrıca, aksiyon uzayına Ornstein-Uhlenbeck (OU) gürültüsü eklenmesinin (SAC+Gürültü modeli), özellikle daha karmaşık olan "şerit takibi ve engelden kaçma" görevinde, temel SAC modeline kıyasla daha düşük hata oranları elde edilmesini sağladığı görülmüştür (bkz. Çizelge 4.9: SAC %8,11 hata, SAC+Gürültü %4,26 hata). Bu durum, aksiyon gürültüsünün sadece eğitim sırasında ajanın çevreyi daha etkin bir şekilde keşfetmesine yardımcı olmakla kalmayıp, aynı zamanda öğrenilen politikanın daha sağlam ve genelleştirilebilir olmasına da katkıda bulunduğunu göstermektedir. Gürültüyle eğitilen bir ajan, eğitimde daha çeşitli durumlarla karşılaşması sayesinde belirsizliklere karşı daha dirençli bir politika geliştirebilmektedir.

Çalışmada ele alınan iki farklı deney senaryosu – "şerit takibi" ve "şerit takibi ve engelden kaçma" – görev karmaşıklığının öğrenme süreci ve model performansı

üzerindeki etkilerini net bir şekilde ortaya koymuştur. Beklendiği üzere, sabit bir engelden kaçma zorunluluğunun eklendiği ikinci senaryo, tüm modeller için daha zorlayıcı olmuştur. Bu senaryolar için tüm modeller incelendiğinde, aksiyon filtreleme modülü kullanan modellerin (özellikle SAC+Gürültü+EMA ve SAC+Gürültü+WMA), karmaşık görevde (bkz. Şekil 4.9) filtresiz modellere kıyasla bölümleri daha sık ve daha erken sonlandırma eğiliminde olduğu görülmektedir. Bu durum, aksiyon filtrelerinin getirdiği pürüzsüzleştirme etkisinin, engelden kaçma gibi ani ve tepkisel manevralar gerektiren durumlarda ajanın anlık optimal aksiyonları seçme esnekliğini bir miktar kısıtlayabileceğini düşündürmektedir. Filtreler, aksiyonlardaki ani değişimleri yumuşatarak bir tür "eylemsizlik" veya "gecikme" yaratır; bu durum basit şerit takibi görevinde faydalı olabilirken, hızla yaklaşan bir engele tepki verme süresini olumsuz etkileyebilir. Bu, filtre parametrelerinin görevin dinamiklerine veya çevresel bağlama göre uyarlanabilir olmasının önemine işaret etmektedir. İkinci senaryoda artan görev karmaşıklığına bir yanıt olarak, "şerit takibi ve engelden kaçma" deneylerinde kullanılan SAC algoritmasının sinir ağı katman boyutları 512 birimden 1024 birime çıkarılmıştır (bkz. Çizelge 4.2 ve Çizelge 4.6 karşılaştırması). Bu kapasite artışı, modelin daha karmaşık durum-aksiyon ilişkilerini öğrenebilmesi için bir gereklilik olarak değerlendirilmiştir. Buna rağmen, elde edilen sonuçlar oldukça iyi olsa da bu tür karmaşık görevlerde optimum performansa ulaşmak için daha uzun eğitim sürelerinin, daha sofistike ağ mimarilerinin veya daha gelişmiş keşif stratejilerinin gerekebileceği görülmektedir.

Tüm sonuçlar ve çıkarımlarla beraber çalışmanın bazı sınırlılıkları bulunmaktadır:

- **Simülasyondan Gerçek Dünyaya Genelleme Zorluğu**: Elde edilen tüm başarılı sonuçlar, Donkey Car simülasyon ortamında elde edilmiştir. Simülasyon ortamları, gerçek dünya dinamiklerinin, sensör gürültülerinin, öngörülemeyen olayların ve çeşitli yol ile hava koşullarının karmaşıklığını tam olarak yansıtmayabilir. Dolayısıyla, bu çalışmada geliştirilen modellerin gerçek otonom araç platformlarındaki performansı doğrudan garanti edilemez.
- **Veri ve Hesaplama Maliyeti**: Derin takviyeli öğrenme algoritmalarının eğitimi, genellikle yüksek miktarda etkileşim verisi (deneyim) ve önemli hesaplama kaynakları (CPU, GPU, RAM) gerektirmektedir. Bu çalışma kapsamında

gerçekleştirilen eğitimler dahi toplamda saatler sürmüş ve önemli miktarda deneyim verisi toplanmıştır (bkz. Çizelge 4.10, 4.11). Bu durum, özellikle daha karmaşık görevler, daha yüksek çözünürlüklü sensör verileri veya daha gelişmiş simülatörler kullanıldığında önemli bir kısıt oluşturabilir.

- **Donanım Kısıtları**: "Donanımsal Performanslar" bölümünde (bkz. Bölüm 4.3) belirtildiği üzere, mevcut 12 GB GPU belleğinin %85 civarında kullanılması, daha karmaşık modellerin veya daha fazla sayıda sensör verisinin eş zamanlı işlenmesinin mevcut donanım için zorlayıcı olabileceğine işaret etmektedir. Gerçek dünya otonom sürüş sistemleri genellikle birden fazla yüksek çözünürlüklü kamera, LIDAR ve radar verisini aynı anda işlemek durumundadır; bu da mevcut donanım kapasitesinin çok ötesinde kaynak gerektirebilir. Bu durum, geliştirilen yaklaşımların gerçek dünya uygulamalarına ölçeklenebilirliği konusunda dikkate alınması gereken bir faktördür.
- **Modüler Yapının Potansiyel Dezavantajları**: Çalışmada modüler yapının avantajları vurgulanmış olsa da bileşenler arası senkronizasyon problemleri yaşanabilir. Her ne kadar bu çalışmanın bulguları bu tür bir sorunu doğrudan göstermese de modül sayısının ve karmaşıklığının artmasıyla bu tür sorunların ortaya çıkma olasılığı göz önünde bulundurulmalıdır.

Bu çalışmanın bulguları ve ortaya konan sınırlılıklar ışığında, derin takviyeli öğrenme tabanlı otonom sürüş sistemlerinin daha da geliştirilmesi ve gerçek dünya uygulamalarına yaklaştırılması amacıyla aşağıdaki alanlarda yapılacak araştırmalar değerli olacaktır:

- **Gerçek Dünya Testleri ve Kapsamlı Doğrulama**: Simülasyon ortamında umut verici sonuçlar veren modellerin, sensör karakteristikleri, aktüatör tepkileri ve çevresel koşullar açısından farklılıklar gösterebilecek gerçek otonom araç platformlarına uyarlanması değerli bir çalışma konusu olabilir.
- **Sensör Füzyonu ve Ortam Algısı**: Mevcut çalışmada kullanılan kamera ve LIDAR verilerine ek olarak radar, GPS/IMU ve yüksek çözünürlüklü harita verileri gibi farklı modalitelerden gelen bilgilerin, daha gelişmiş füzyon teknikleriyle (örneğin, derin öğrenme tabanlı füzyon ağları) birleştirilerek ortam algısının zenginleştirilmesi ve sürüş güvenliğinin artırılması hedeflenmelidir.

- **Dinamik Aksiyon Filtreleme**: Bu çalışmada kullanılan EMA ve WMA filtrelerinin sabit parametrelerle çalışması bir sınırlılıktır. Gelecekte, filtre parametrelerinin (örneğin, pencere boyutu n veya ağırlık w) anlık sürüş koşullarına (hız, yol tipi, trafik yoğunluğu, hava durumu) veya ajanın o anki taktiksel ihtiyacına (örneğin, ani bir engelden kaçınma gereksinimi karşısında daha az filtreleme, sakin bir otoyol sürüşünde daha fazla pürüzsüzleştirme) göre dinamik olarak ayarlandığı adaptif filtreleme mekanizmalarının geliştirilmesi, sistemin tepkiselliği ve pürüzsüzlüğü arasında daha iyi bir denge kurmasını sağlayabilir.
- **Çok Katmanlı Karar Verme Mimarileri**: Otonom sürüş, farklı seviyelerde kararlar almayı gerektirir. Yüksek seviyeli stratejik kararlar (örneğin, rota planlama, şerit değiştirme stratejisi, sollama kararı) ile düşük seviyeli operasyonel kontrollerin (direksiyon açıları, gaz/fren komutları) daha etkin bir şekilde entegre edildiği çok katmanlı derin takviyeli öğrenme yaklaşımlarının araştırılması, daha karmaşık ve uzun vadeli sürüş senaryolarının üstesinden gelinmesine yardımcı olabilir.
- **Sosyal ve Etkileşimli Sürüş Davranışlarının Modellenmesi**: Gerçekçi ve güvenli otonom sürüş, sadece aracın kendi dinamiklerini kontrol etmesini değil, aynı zamanda diğer yol kullanıcılarıyla (araçlar, yayalar, bisikletliler) etkileşim kurmasını ve trafik kurallarına uymasını gerektirir. Diğer ajanların niyetlerini tahmin edebilen, onlarla işbirlikçi veya rekabetçi senaryolarda etkileşime (connectivity) girebilen ve sosyal açıdan kabul edilebilir sürüş davranışları sergileyebilen derin takviyeli öğrenme ajanlarının geliştirilmesi önemli bir araştırma yönüdür.
- **Gelişmiş Keşif Stratejileri**: Derin takviyeli öğrenmede etkili bir keşif stratejisi, özellikle karmaşık ortamlarda optimal politikaların bulunması için hayati öneme sahiptir. Farklı ve daha gelişmiş keşif stratejilerinin (merak tabanlı keşif, içsel motivasyon mekanizmaları veya hiyerarşik keşif gibi) SAC algoritmasına entegre edilmesi, öğrenme sürecini hızlandırabilir ve daha iyi sonuçlar elde edilmesini sağlayabilir.

## KAYNAKLAR


Abouelazm, A., Michel, J., & Zöllner, J. M. (2024). A Review of Reward Functions for Reinforcement Learning in the context of Autonomous Driving. *2024 IEEE Intelligent Vehicles Symposium (IV)*, 156-163. https://doi.org/10.1109/IV55156.2024.10588385

*About Donkey®*. (t.y.). https://docs.donkeycar.com/.

Almahamid, F., & Grolinger, K. (2021). Reinforcement Learning Algorithms: An Overview and Classification. *Canadian Conference on Electrical and Computer Engineering*, *2021-September*. https://doi.org/10.1109/CCECE53047.2021.9569056

Andrade, N., Ribeiro, T., Coelho, J., Lopes, G., & Ribeiro, A. F. (2022). Combining YOLO and Deep Reinforcement Learning for Autonomous Driving in Public Roadworks Scenarios. *International Conference on Agents and Artificial Intelligence*, *3*, 793-800. https://doi.org/10.5220/0010913600003116

Arulkumaran, K., Deisenroth, M. P., Brundage, M., & Bharath, A. A. (2017). Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, *34*(6), 26-38. https://doi.org/10.1109/MSP.2017.2743240

Bachute, M. R., & Subhedar, J. M. (2021). Autonomous Driving Architectures: Insights of Machine Learning and Deep Learning Algorithms. *Machine Learning with Applications*, *6*, 100164. https://doi.org/10.1016/J.MLWA.2021.100164

Bank, D., Koenigstein, N., & Giryes, R. (2023). Autoencoders. Içinde *Machine Learning for Data Science Handbook* (ss. 353-374). Springer International Publishing. https://doi.org/10.1007/978-3-031-24628-9_16

Bengio, Y. (2009). Learning Deep Architectures for AI. *Foundations and Trends® in Machine Learning*, *2*(1), 1-127. https://doi.org/10.1561/2200000006

Berger-Tal, O., Nathan, J., Meron, E., & Saltz, D. (2014). The exploration-exploitation dilemma: A multidisciplinary framework. *PLoS ONE*, *9*. https://doi.org/10.1371/journal.pone.0095693

Bibbona, E., Panfilo, G., & Tavella, P. (2008). The Ornstein–Uhlenbeck process as a model of a low pass filtered white noise. *Metrologia*, *45*(6), S117-S126. https://doi.org/10.1088/0026-1394/45/6/S17

Bojarski, M., Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L., Monfort, M., Muller, U., Zhang, J., Zhang, X., Zhao, J., & Zieba, K. (2016). End to End Learning for Self-Driving Cars. *arXiv.org*. https://doi.org/https://doi.org/10.48550/arXiv.1604.07316

Carton, F. (2021). *Exploration of reinforcement learning algorithms for autonomous vehicle visual perception and control*. https://theses.hal.science/tel-03273748

Castanedo, F. (2013). A review of data fusion techniques. *The Scientific World Journal*, *2013*. https://doi.org/10.1155/2013/704504

Deichmann, J., Ebel, E., Heineke, K., Heuss, R., Kellner, M., & Steiner, F. (2023, Ocak 6). *Autonomous driving's future: Convenient and connected*. https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/autonomous-drivings-future-convenient-and-connected#/

Duarte, F., & Ratti, C. (2018). The Impact of Autonomous Vehicles on Cities: A Review. *Journal of Urban Technology*, *25*(4), 3-18. https://doi.org/10.1080/10630732.2018.1493883

Dutta, D., & Upreti, S. R. (2022). A survey and comparative evaluation of actor-critic methods in process control. *Canadian Journal of Chemical Engineering*, *100*(9), 2028-2056. https://doi.org/10.1002/CJCE.24508

El Naqa, I., Murphy, M. J., El Naqa, I., & Murphy, M. J. (2015). What Is Machine Learning? *Machine Learning in Radiation Oncology*, 3-11. https://doi.org/10.1007/978-3-319-18305-3_1

Elallid, B. Ben, Benamar, N., Hafid, A. S., Rachidi, T., & Mrani, N. (2022). A Comprehensive Survey on the Application of Deep and Reinforcement Learning Approaches in Autonomous Driving. *Journal of King Saud University - Computer and Information Sciences*, *34*(9), 7366-7390. https://doi.org/10.1016/J.JKSUCI.2022.03.013

Emuna, R., Borowsky, A., & Biess, A. (2020). Deep Reinforcement Learning for Human-Like Driving Policies in Collision Avoidance Tasks of Self-Driving Cars. *arxiv.org*. https://doi.org/https://doi.org/10.48550/arXiv.2006.04218

Ethem, A. (2020). Introduction to Machine Learning in Adaptive Computation and Machine Learning series. *Massachusetts Institute of Technology*, 712.

Gao, H., Yu, H., Xie, G., Ma, H., Xu, Y., & Li, D. (2019). Hardware and software architecture of intelligent vehicles and road verification in typical traffic scenarios. *IET Intelligent Transport Systems*, *13*(6), 960-966. https://doi.org/10.1049/IET-ITS.2018.5351

Gläscher, J., Daw, N., Dayan, P., & O'Doherty, J. P. (2010). States versus rewards: Dissociable neural prediction error signals underlying model-based and model-free reinforcement learning. *Neuron*, *66*(4), 585-595. https://doi.org/10.1016/J.NEURON.2010.04.016/ATTACHMENT/C2A07BEA-46B6-4B19-A0DA-8C76BE7D18B4/MMC1.PDF

Glasmachers, T., Noh, Y.-K., & Zhang, M.-L. (2017). Limits of End-to-End Learning. Içinde *Proceedings of Machine Learning Research* (C. 77, ss. 17-32). PMLR. https://proceedings.mlr.press/v77/glasmachers17a.html

Grigorescu, S., Trasnea, B., Cocias, T., & Macesanu, G. (2020). A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics*, *37*(3), 362-386. https://doi.org/10.1002/ROB.21918

Grondman, I., Busoniu, L., Lopes, G. A. D., & Babuška, R. (2012). A survey of actor-critic reinforcement learning: Standard and natural policy gradients. Içinde *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews* (C. 42, ss. 1291-1307). https://doi.org/10.1109/TSMCC.2012.2218595

Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *35th International Conference on Machine Learning, ICML 2018*, *5*, 2976-2989.

Hall, D. L., & Llinas, J. (1997). An introduction to multisensor data fusion. *Proceedings of the IEEE*, *85*(1), 6-23. https://doi.org/10.1109/5.554205

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the Dimensionality of Data with Neural Networks. *Science*, *313*(5786), 504-507. https://doi.org/10.1126/science.1127647

Hollenstein, J., Auddy, S., Saveriano, M., Renaudo, E., & Piater, J. (2022a). *Action Noise in Off-Policy Deep Reinforcement Learning: Impact on Exploration and Performance*. https://arxiv.org/abs/2206.03787v3

Hollenstein, J., Auddy, S., Saveriano, M., Renaudo, E., & Piater, J. (2022b). *Action Noise in Off-Policy Deep Reinforcement Learning: Impact on Exploration and Performance*.

Huang, Y., & Chen, Y. (2020). Survey of State-of-Art Autonomous Driving Technologies with Deep Learning. *Proceedings - Companion of the 2020 IEEE 20th International Conference on Software Quality, Reliability, and Security, QRS-C 2020*, 221-228. https://doi.org/10.1109/QRS-C51114.2020.00045

Huang, Z., Xu, X., He, H., Tan, J., & Sun, Z. (2019). Parameterized Batch Reinforcement Learning for Longitudinal Control of Autonomous Land Vehicles. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *49*(4), 730-741. https://doi.org/10.1109/TSMC.2017.2712561

Ignatious, H. A., Sayed, H. El, & Khan, M. (2022). An overview of sensors in Autonomous Vehicles. *Procedia Computer Science*, *198*, 736-741. https://doi.org/10.1016/J.PROCS.2021.12.315

Isele, D., Rahimi, R., Cosgun, A., Subramanian, K., & Fujimura, K. (2018). Navigating Occluded Intersections with Autonomous Vehicles Using Deep Reinforcement Learning. *Proceedings - IEEE International Conference on Robotics and Automation*, 2034-2039. https://doi.org/10.1109/ICRA.2018.8461233

Kendall, A., Hawke, J., Janz, D., Mazur, P., Reda, D., Allen, J. M., Lam, V. D., Bewley, A., & Shah, A. (2019). Learning to drive in a day. *Proceedings - IEEE International Conference on Robotics and Automation*, *2019-May*, 8248-8254. https://doi.org/10.1109/ICRA.2019.8793742

Kiran, B. R., Sobh, I., Talpaert, V., Mannion, P., Sallab, A. A. A., Yogamani, S., & Perez, P. (2022). Deep Reinforcement Learning for Autonomous Driving: A Survey. *IEEE Transactions on Intelligent Transportation Systems*, *23*(6), 4909-4926. https://doi.org/10.1109/TITS.2021.3054625

Kramer, M. A. (1992). Autoassociative neural networks. *Computers & Chemical Engineering*, *16*(4), 313-328. https://doi.org/10.1016/0098-1354(92)80051-A

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84-90. https://doi.org/10.1145/3065386

Kuutti, S., Bowden, R., Jin, Y., Barber, P., & Fallah, S. (2021). A Survey of Deep Learning Applications to Autonomous Vehicle Control. *IEEE Transactions on Intelligent Transportation Systems*, *22*(2), 712-733. https://doi.org/10.1109/TITS.2019.2962338

Ladosz, P., Weng, L., Kim, M., & Oh, H. (2022). Exploration in deep reinforcement learning: A survey. *Information Fusion*, *85*, 1-22. https://doi.org/10.1016/j.inffus.2022.03.003

Li, D., Zhao, D., Zhang, Q., & Chen, Y. (2019). Reinforcement Learning and Deep Learning Based Lateral Control for Autonomous Driving [Application Notes]. *IEEE Computational Intelligence Magazine*, *14*(2), 83-98. https://doi.org/10.1109/MCI.2019.2901089

Liao, J., Liu, T., Tang, X., Mu, X., Huang, B., & Cao, D. (2020). Decision-making strategy on highway for autonomous vehicles using deep reinforcement learning. *IEEE Access*, *8*, 177804-177814. https://doi.org/10.1109/ACCESS.2020.3022755

Liu, L., Lu, S., Zhong, R., Wu, B., Yao, Y., Zhang, Q., & Shi, W. (2021). Computing Systems for Autonomous Driving: State of the Art and Challenges. *IEEE Internet of Things Journal*, *8*(8), 6469-6486. https://doi.org/10.1109/JIOT.2020.3043716

Liu, S., Liu, L., Tang, J., Yu, B., Wang, Y., & Shi, W. (2019). Edge Computing for Autonomous Driving: Opportunities and Challenges. *Proceedings of the IEEE*, *107*(8), 1697-1716. https://doi.org/10.1109/JPROC.2019.2915983

Lu, C., Gong, J., Lv, C., Chen, X., Cao, D., & Chen, Y. (2019). A Personalized Behavior Learning System for Human-Like Longitudinal Speed Control of Autonomous Vehicles. *Sensors 2019, Vol. 19, Page 3672*, *19*(17), 3672. https://doi.org/10.3390/S19173672

Mausam, & Kolobov, A. (2012). *Planning with Markov Decision Processes*. https://doi.org/10.1007/978-3-031-01559-5

Meyn, S. (2024). The Projected Bellman Equation in Reinforcement Learning. *IEEE Transactions on Automatic Control*. https://doi.org/10.1109/TAC.2024.3409647

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529-533. https://doi.org/10.1038/nature14236

Nachum, O., Norouzi, M., Xu, K., & Schuurmans, D. (2017). Bridging the Gap Between Value and Policy Based Reinforcement Learning. *In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17*, 2772-2782.

Nosrati, M., Abolfathi, E., Elmahgiubi, M., Yadmellat, P., Luo, J., Zhang, Y., Yao, H., Zhang, H., & Jamil, A. (2018). *Towards Practical Hierarchical Reinforcement Learning for Multi-lane Autonomous Driving*.

Parekh, D., Poddar, N., Rajpurkar, A., Chahal, M., Kumar, N., Joshi, G. P., & Cho, W. (2022). A Review on Autonomous Vehicles: Progress, Methods and Challenges. *Electronics 2022, Vol. 11, Page 2162*, *11*(14), 2162. https://doi.org/10.3390/ELECTRONICS11142162

*Part 2: Kinds of RL Algorithms*. (2018). https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html.

Peng, B., Sun, Q., Li, S. E., Kum, D., Yin, Y., Wei, J., & Gu, T. (2021a). End-to-End Autonomous Driving Through Dueling Double Deep Q-Network. *Automotive Innovation*, *4*(3), 328-337. https://doi.org/10.1007/S42154-021-00151-3/FIGURES/10

Peng, B., Sun, Q., Li, S. E., Kum, D., Yin, Y., Wei, J., & Gu, T. (2021b). End-to-End Autonomous Driving Through Dueling Double Deep Q-Network. *Automotive Innovation*, *4*(3), 328-337. https://doi.org/10.1007/S42154-021-00151-3

Pérez-Gil, Ó., Barea, R., López-Guillén, E., Bergasa, L. M., Gómez-Huélamo, C., Gutiérrez, R., & Díaz-Díaz, A. (2022). Deep reinforcement learning based control for Autonomous Vehicles in CARLA. *Multimedia Tools and Applications*, *81*(3), 3553-3576. https://doi.org/10.1007/S11042-021-11437-3

Plappert, M., Houthooft, R., Dhariwal, P., Sidor, S., Chen, R. Y., Chen, X., Asfour, T., Abbeel, P., & Andrychowicz, M. (2018). Parameter space noise for exploration. *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*.

Plebe, A., Svensson, H., Mahmoud, S., & Da Lio, M. (2024). Human-inspired autonomous driving: A survey. *Cognitive Systems Research*, *83*, 101169. https://doi.org/10.1016/j.cogsys.2023.101169

Schwarz, B. (2010). Lidar: Mapping the world in 3D. *Nature Photonics*, *4*(7), 429-430. https://doi.org/10.1038/NPHOTON.2010.148

Serban, A. C., Poll, E., & Visser, J. (2018). A Standard Driven Software Architecture for Fully Autonomous Vehicles. *Proceedings - 2018 IEEE 15th International Conference on Software Architecture Companion, ICSA-C 2018*, 120-127. https://doi.org/10.1109/ICSA-C.2018.00040

Sewak, M. (2019). Deep Reinforcement Learning. *Deep Reinforcement Learning*. https://doi.org/10.1007/978-981-13-8285-7

Sutton, R. S., & Barto, A. G. (2005). Reinforcement Learning: An Introduction. *IEEE Transactions on Neural Networks*, *9*, 1054-1054. https://doi.org/10.1109/tnn.1998.712192

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction Second edition, in progress*.

Tang, Y., & Agrawal, S. (2020). Discretizing Continuous Action Space for On-Policy Optimization. *Proceedings of the AAAI Conference on Artificial Intelligence*, *34*(04), 5981-5988. https://doi.org/10.1609/AAAI.V34I04.6059

Van Otterlo, M., & Wiering, M. (2012). Reinforcement Learning and Markov Decision Processes. *Adaptation, Learning, and Optimization*, *12*, 3-42. https://doi.org/10.1007/978-3-642-27645-3_1

Wang, P., Chan, C. Y., & De La Fortelle, A. (2018). A Reinforcement Learning Based Approach for Automated Lane Change Maneuvers. *2018 IEEE Intelligent Vehicles Symposium (IV)*, *2018-June*, 1379-1384. https://doi.org/10.1109/IVS.2018.8500556

Wang, W., Duan, J., Song, X., Xiao, L., Chen, L., Wang, Y., Cheng, B., & Li, S. E. (2024). Smooth Filtering Neural Network for Reinforcement Learning. *IEEE Transactions on Intelligent Vehicles*, 1-10. https://doi.org/10.1109/TIV.2024.3446823

Wang, X., Maleki, M. A., Azhar, M. W., & Trancoso, P. (2024). *Moving Forward: A Review of Autonomous Driving Software and Hardware Systems*. http://arxiv.org/abs/2411.10291

Wani, M. A., Bhat, F. A., Afzal, S., & Khan, A. I. (2019). *Advances in Deep Learning*. *57*. https://doi.org/10.1007/978-981-13-6794-6

Yeong, D. J., Velasco-hernandez, G., Barry, J., & Walsh, J. (2021). Sensor and Sensor Fusion Technology in Autonomous Vehicles: A Review. *Sensors 2021, Vol. 21, Page 2140*, *21*(6), 2140. https://doi.org/10.3390/S21062140

Yurtsever, E., Lambert, J., Carballo, A., & Takeda, K. (2020). A Survey of Autonomous Driving: *Common Practices and Emerging Technologies*. *IEEE Access*, *8*, 58443-58469. https://doi.org/10.1109/ACCESS.2020.2983149

Zhang, T., Zhan, J., Shi, J., Xin, J., & Zheng, N. (2023). Human-Like Decision-Making of Autonomous Vehicles in Dynamic Traffic Scenarios. *IEEE/CAA Journal of Automatica Sinica*, *10*(10), 1905-1917. https://doi.org/10.1109/JAS.2023.123696

Zhang, Y., Sun, P., Yin, Y., Lin, L., & Wang, X. (2018). Human-like Autonomous Vehicle Speed Control by Deep Reinforcement Learning with Double Q-Learning. *IEEE Intelligent Vehicles Symposium, Proceedings*, *2018-June*, 1251-1256. https://doi.org/10.1109/IVS.2018.8500630

Zong, W., Zhang, C., Wang, Z., Zhu, J., & Chen, Q. (2018). Architecture design and implementation of an autonomous vehicle. *IEEE Access*, *6*, 21956-21970. https://doi.org/10.1109/ACCESS.2018.2828260