NÖHÜ Müh. Bilim. Derg. / NOHU J. Eng. Sci., 2024; 13(2), 429-439
Niğde Ömer Halisdemir Üniversitesi Mühendislik Bilimleri Dergisi
Niğde Ömer Halisdemir University Journal of Engineering Sciences
Araştırma makalesi / Research article
www.dergipark.org.tr/tr/pub/ngumuh / www.dergipark.org.tr/en/pub/ngumuh

Otonom araçlarda derin pekiştirmeli öğrenme yöntemleri ile sollama

Overtaking with deep reinforcement learning methods in autonomous vehicles

Fehim Köylü[1,*], Yasin Atılkan[2]

[1] *Erciyes Üniversitesi, Bilgisayar Mühendisliği Bölümü, 38030, Kayseri, Türkiye*
[2] *Ankara Üniversitesi, Yapay Zeka ve Veri Mühendisliği Bölümü, 06830, Ankara, Türkiye*
[1] *Kayseri Robotik, Otonom Sistemler ve Yapay Zeka Lab., 38030, Kayseri, Türkiye*

Öz

Pekiştirmeli öğrenme ile derin öğrenme yaklaşımlarını birleştiren derin pekiştirmeli öğrenme algoritmaları zorlu otonom araç görevlerinde kullanılmaktadır. Öndeki aracı geçme, içerisinde barındırdığı farklı türden alt görevler nedeni ile en zorlu otonom araç görevlerinden biridir. Literatürdeki güncel çalışmalar zorlu görevleri çözmek için müfredat öğrenme yaklaşımını derin pekiştirmeli öğrenme ile kullanmaktadır. Bu çalışmada, özgün olarak oluşturulmuş ortamda, yaygın olarak kullanılan derin Q-ağları, avantaj aktör kritik ve proksimal politika optimizasyonu algoritmaları ile yarısı müfredat öğrenme yaklaşımına uğramış 12 model eğitilmiştir. Modellerin değerlendirilmesinde modellerin eğitim süreci ve modellerin ortamda test edilmesi birlikte kullanılmıştır. Çalışmada, tüm modellerde olmasa da derin Q-ağları ve proksimal politika optimizasyonu yöntemleri ile başarılı modeller eğitilmiştir. Başarılı modeller içerisinde müfredat öğrenimi ile bir derin Q-ağları modelinin performansı artırılarak yaklaşımın olumlu etkisi görülmüştür.

Anahtar kelimeler: Derin pekiştirmeli öğrenme, Müfredat öğrenme, Otonom araçlar

Abstract

Deep reinforcement algorithms that combine reinforcement learning and deep learning approaches are used in challenging autonomous vehicle tasks. Passing the vehicle in front is one of the most challenging autonomous vehicle tasks due to the different types of subtasks involved. Recent studies in the literature use the curriculum learning approach with deep reinforcement learning to solve challenging tasks. In this study, 12 models, half of which have undergone a curriculum learning approach, are trained in a uniquely constructed environment with commonly used deep Q-networks, advantage actor critic and proximal policy optimization algorithms. The evaluation of the models is based on both the training process and the testing of the models in the environment. In the study, successful models were trained with deep Q-networks and proximal policy optimization methods, although not for all models. Among the successful models, the performance of a deep Q-network model was improved with curriculum learning, showing the positive impact of the approach.

Keywords: Deep reinforcement learning, Curriculum learning, Autonomous vehicles

1 Giriş

Bir otonom aracın insan müdahalesi olmaksızın çok hızlı değişen ve önceden bilinemeyen durumlara sahip trafik ortamında, maddi ve manevi zararları önleyecek şekilde hareket etmesi çok önemlidir. Kiran vd. [1] tarafından yapılan çalışmada bir otonom aracın görevleri, şerit takibi, şerit değiştirme, yol birleştirme, öndeki aracı geçme, kavşak hareketleri ve hareket planlama olmak üzere sınıflandırılmıştır. Bunlar arasında en karmaşık olanlardan biri öndeki aracı geçme problemidir. Bu görev hızlanma, şerit değiştirme ve daha sonra tekrardan geçilen aracın önünde eski şeride geri dönme olmak üzere bir dizi alt görevi içermektedir. Ek olarak bu görev ülkelere ait trafik kuralları ile farklı davranışları da gerektirebilir. Türkiye'de trafik içerisinde öndeki aracı geçme davranışı sol sinyalin verilişi, şerit kontrolü, sol şeride geçme, öndeki aracı geçme ve sağ şeride geçme davranışlarını bir arada yapılmasını gerektirmektedir [2].

Öndeki aracı geçme problemine yönelik yapılan çalışmalar iki sınıfa ayrılabilir. Bu sınıflar modüler boru hattı yaklaşımları ile uçtan uca öğrenme yaklaşımlarıdır [3]. Ek bir yaklaşım olarak Chen vd. [4] oyun ortamındaki görüntü verilerini kullanarak evrişimli sinir ağı modellerini doğrudan algı yaklaşımları adını verdikleri yöntem ile eğitmişlerdir.

Modüler boru hattı yaklaşımlarında, görüntü verilerinden gaz, fren ve direksiyon eylem kararlarını verirken yol planlama, araç kontrolü vb. birimleri ayrı olarak ele alınmaktadır. Uçtan uca öğrenmede ise sinir ağları ile görüntü verileri doğrudan eylem kararlarına eşlenmektedir. Bu nedenle otonom bir araçta; gaz, fren ve direksiyon olmak üzere verilmesi gereken kontrol kararları için derin sinir ağlarının pekiştirmeli öğrenme yaklaşımı ile eğitilmesi bir uçtan uca öğrenmedir. Klasik pekiştirmeli öğrenme algoritmalarının aksine derin pekiştirmeli öğrenme yöntemlerinin bilinmeyen durumlara genelleme yetenekleri daha yüksektir. Bu durum öndeki aracı geçme de dahil olmak üzere değişken ortamlarda çalışması beklenen otonom görevlerin çözümünde pekiştirmeli öğrenme ile derin öğrenmenin kullanımını zamanla artırmıştır.

Bu çalışmada, yol şerit tipleri, trafik ışıkları, trafik levhaları, hava durumları, diğer araç ya da engeller vb. faktörler göz ardı edilerek bir otonom aracın yavaş giden

* Sorumlu yazar / Corresponding author, e-posta / e-mail: fehimkoylu@erciyes.edu.tr (F. Köylü)
Geliş / Recieved: 22.07.2023 Kabul / Accepted: 28.12.2023 Yayımlanma / Published: 15.04.2024
doi: 10.28948/ngumuh.1331354

diğer aracı geçmesi problemine odaklanılmıştır. İlk olarak özgün bir pekiştirmeli öğrenme ortamının kurulması için kullanılabilecek olan bir uygulama programlama arayüzü (API) kullanılarak bir pekiştirmeli öğrenme ortamı kurulmuştur. Daha sonra müfredat öğrenimi yaklaşımı ile eğitilecek modeller için ilk ortamın kısıtlanmış versiyonu olan alt müfredat öğrenme ortamı ikinci ortam olarak kurulmuştur. Bu iki ortam için ortam dinamikleri tamamen özgün olup pekiştirmeli öğrenme ödül fonksiyonu basit tutularak 3 popüler derin pekiştirmeli öğrenme algoritması derin Q-ağları, avantaj aktör kritik ve proksimal politika optimizasyonu ile oluşturulan modellerinin mevcut problem üzerindeki etkinliği incelenmiştir. Kullanılan algoritmalar hiper-parametre yönünden çeşitlendirilerek aynı algoritma modelleri farklı ayarlarda karşılaştırılmıştır. Daha sonra eğitilen modeller aynı hiper-parametre ayarlarında tutularak müfredat öğrenme yaklaşımı ile eğitilmiş ve seyrek ödül anlarına sahip otonom araç geçme görevinde müfredat öğreniminin performansa etkisi incelenmiştir.

Çalışma sonucunda derin Q-ağları ve proksimal politika optimizasyonu ile eğitilen başarılı modeller, yapılması muhtemel kapsamlı otonom araç çalışmalarında derin pekiştirmeli öğrenme yöntemlerinin kullanılması için motivasyonu artıracaktır. Bu duruma karşın hiper-parametre çeşitlendirmeleri ile aynı algoritma modelleri arasındaki oluşan başarım farkları, algoritmaların parametre duyarlılığı konusundaki hassasiyetlerini göstermektedir. Bu durum algoritmaların hiper-parametre kırılganlığı yönünden geliştirebileceği sonucunu ortaya çıkarmaktadır. Başarılı kararlar alabilen modeller içerisinde bir derin Q-ağları modelinin performansı müfredat pekiştirmeli öğrenme ile belirgin şekilde artırılmıştır. Bu durum yaklaşımın olumlu etkisini gösterse de diğer modeller için başarılı modellerin elde edilememesi müfredat öğreniminde ya da farklı tarzda farklı optimizasyon yaklaşımlarının da kullanılabileceği sonucunu ortaya çıkarmaktadır.

2 Geçmiş çalışmalar

Loiacono vd. [5] TORCS simülatöründe araç geçme görevini çözmek için pekiştirmeli öğrenmede tablo tabanlı Q-öğrenme algoritmasını kullanmışlardır. Çalışmada 4 katmanlı bir mimari tanımlamışlardır ve düz ya da virajlı yollar için iki farklı senaryo gerçekleştirmişlerdir. Sonuç olarak Q-öğrenme algoritması, iyi sürücülerden birinin uyguladığı sürüş politikasından daha iyi bir performans göstermiştir. Ngai ve Yung [6] tarafından yapılan çalışmada Çok Amaçlı Pekiştirmeli Öğrenme yöntemi önerilmiştir. Yazarlar algoritma olarak Q-öğrenme ve çift eylemli Q-öğrenme algoritmasını kullanarak oluşturdukları modellerin doğru eylem kararları aldıklarını göstermişlerdir. Li vd. [7] yaptıkları çalışma ile araç geçme görevinde uygun politikaları öğrenmek için model oluşturmada Q-öğrenme algoritmasını kullanmışlardır. Oluşturdukları ödül fonksiyonlarında insan sürüş deneyimlerini kullanmışlar ve sonuç olarak Q-öğrenme algoritması ile kural tabanlı yöntemlerden daha iyi performans gösteren modelleri geliştirmişlerdir. Xia ve Han [8] tarafından yapılan çalışmada TORCS simülatöründe otonom aracın uygun hızda sürmeyi öğrenmesi ve araç geçme görevlerini gerçekleştirmesi için derin pekiştirmeli öğrenme algoritmaları kullanılmıştır. Çalışmada politika gradyan yöntemleri ile değer tabanlı yöntemler karşılaştırılmış ve politika gradyan tabanlı yöntemler daha iyi performans göstermiştir. Ek olarak çalışmada karmaşık sürüş görevleri alt kümelere ayrılarak "Seçenek Çerçevesi" adı verilen bir yöntem kullanılmış ve sonuç olarak pekiştirmeli öğrenme yaklaşımlarının otonom araç geçmede başarılı sonuçlar verdiği görülmüştür. Kaushik vd. [9] eğitim anında ham sensör verilerini de kullanarak derin pekiştirmeli öğrenme tabanlı derin deterministik politika gradyanı (DDPG) yöntemi ile öndeki aracı geçme problemini çözmeyi amaçlamışlardır. Bu çalışmada, çözümü daha basit problemlerden zor problemlere aşamalı olarak ayırıp müfredat pekiştirmeli öğrenme yaklaşımı kullanılarak başarılı sonuçlar alınmıştır. Li vd. [10] tarafından DDPG algoritması kullanarak yapılan bir çalışmada, öndeki aracı geçmek için yeni bir model önerilmiştir. Bu çalışmada otonom aracın yörüngesi üzerinde öndeki araç ve diğer engeller ile olan mesafeleri hesaplanıp hassas matematiksel ölçümler yapılarak pekiştirmeli öğrenme ödül fonksiyonları oluşturulmuştur. Modelin geleneksel algoritmalardan daha iyi sonuçlar verdiği gösterilmiştir. Song vd. [11] tarafından yapılan çalışmada müfredat öğrenimi pekiştirmeli öğrenme yaklaşımı çalışma içerisinde kullanılan oyun ortamındaki yapay zekâ modeli ile karşılaştırılmış ve müfredat öğrenimi ile performansın arttığı gözlenmiştir. Liu vd. [12] tarafından yapılan çalışmada otonom aracın öndeki aracı geçmesindeki zorluğun üstesinden gelmek için özgün bir müfredat öğrenme yaklaşımı kullanılmıştır. Esnek aktör-kritik (EAK) yöntemi tabanlı bir mimari oluşturarak TORCS simülatöründe yüksek hızda sollama elde etmek için görev iki aşamaya ayrılmıştır. Yapılan ilk eğitimde parkuru hızlı tamamlama üzerine odaklanılmış, daha sonra elde edilen model ikinci eğitim olan parkurda tüm araçların geçilmesinde kullanılmıştır. Bu şekilde ödül seyrekliği sorununun üstesinden gelinerek sonuçların müfredat öğrenme yaklaşımı ile iyileştirildiği gösterilmiştir.

Pekiştirmeli öğrenme karmaşık modellere sahip birçok farklı problemde çevre şartlarına uygun en iyi kontrolü sağlamak amacıyla literatürde kullanılmıştır. Aslan vd. [13] tarafından yapılan çalışmada insansı robotların yürüyüşünde dış kuvvetler karşısında denge kontrolünde, Uc-Cetina vd. [14] tarafından yapılan derleme makalesinde belirtildiği gibi doğal dil işleme alanında yapılan makine çevirisi, dil anlama, metin üretme vb. uygulamalarda kullanılmıştır.

3 Materyal ve metot

Mevcut problemi çözmek için pekiştirmeli öğrenme ortamı ile alt müfredat pekiştirmeli öğrenme ortamının bileşenleri ve tasarımı anlatılacaktır. İki şeritli yolda öndeki aracı geçmek için hızlanma, şerit değiştirme, öndeki aracın dikey eksende önüne geçilmesi ve aynı şeritte geçilecek aracın önüne geçilmesi ile tamamlanacak şekilde diğer birtakım çalışmalardan farklı olarak tam bir otonom araç geçme görevi tanımlanacaktır.

Çalışmada, tanımlanan mevcut görev üzerinde 3 adet literatürde yaygın olarak kullanan derin pekiştirmeli öğrenme algoritması kullanılmıştır. Bu algoritmalar iki farklı

hiper-parametre ayarı ile çeşitlendirilerek her bir algoritmadan 2 tane olmak üzere toplam 6 model eğitilmiştir. Aynı modeller müfredat pekiştirmeli öğrenme yaklaşımı ile de eğitilip ek 6 model oluşturulmuş ve toplamda 12 model elde edilmiştir.

3.1 Pekiştirmeli öğrenme

Pekiştirmeli öğrenme, makine öğrenmesinin alt yaklaşımı olarak herhangi bir ortamda bulunan aracının (agent) bir problemi çözmek ya da o ortamda varlığını uygun bir şekilde devam ettirmek için yaptığı eylemlerden aldığı ortam dönütlerine göre kendini eğitmesidir. Pekiştirmeli öğrenme ortamları, Markov karar süreçleri (MDP) kullanarak modellenirken *(S, A, p, R)* olarak gösterilmektedir [15]. Bir ortam üzerinde t zaman adımında, aracının bulunduğu durum $S_t \in S$, bu durum içerisinde aracının yapabileceği eylem $A_t \in A$ ve eylemi sonucunda aldığı ödül ise $r \in R$ olarak gösterilmektedir. MDP içerisinde durum – eylem çiftleri için aracının bir sonraki durum *s'* durumuna *r* ödülü ile geçme olasılığı ise *p(s', r | s, a)* olarak gösterilmektedir. Aracının ortam üzerindeki eylemleri seçim tarzı politika olarak adlandırılmaktadır. Aracının politikasına göre durum – eylem dizisi aracı başarılı veya başarısız olana kadar ya da ortam şartlarına kadar belirli bir süre boyunca devam eder. Bu süreç bölüm (episode) olarak adlandırılmaktadır. Aracının nihai amacı bölüm boyunca aldığı ödüllerin toplamı olarak adlandırılan getiriyi (return) en büyük yapan politikayı bulmaktır. Getiri formülü Denklem (1)'de gösterilmektedir.

$$R_t = \sum_{k=0}^{\infty} \gamma^k \, r_{t+k+1}, \tag{1}$$

Getiri hesaplanmasında, sonraki zaman adımlarındaki ödüllerin etkisini azaltmak için γ indirim faktörü olarak kullanılmaktadır.

Makine öğrenmesi uygulamalarında modelleri geliştirmek için kullanılan müfredat öğrenme, zor olan görevlerinden üstesinden gelmek için önce daha basit olan görevleri dikkate almayı temel almaktadır [16]. Bu kavram müfredat pekiştirmeli öğrenme adı ile düşünüldüğünde başarı anları seyrek olan pekiştirmeli öğrenme görevlerinde görev alt aşamalara ayrılmaktadır. Modeller ilk olarak daha basit olan alt görevde eğitilmektedir. Daha sonra eğitilen modeller ana problemi çözmek için tekrardan eğitilmektedirler.

3.2 Derin öğrenme

Doğal bir yapı olarak beyin, içerisindeki sinir hücrelerinin oluşturduğu ağ yapısı ile herhangi bir problem için görmediği durumlara karşı sorun çözme yeteneğine sahiptir. Bu yeteneğe yapay zekâ literatüründeki çalışmalara bakıldığında genelleme denmektedir. Genelleme yaparak daha güçlü modeller üretmek amacı ile doğal sinir ağlarının çalışmasını örnek alarak hesaplama yapabilen sinir hücreleri ve bu hücrelerin arka arkaya gelmesiyle çok katmanlı mimari ile matematiksel bir yapı oluşturarak problemleri hiyerarşik bir yapıda çözmeyi amaçlayan makine öğrenmesi alt yaklaşımına derin öğrenme denir [17]. Derin ağlardaki her bir sinir hücresi ağırlıkları olan bağlantılar ile kendinden önceki birtakım sinir hücrelerine bağlıdır. Önceki katmandan gelen sayısal değerler ve geldiği bağlantının ağırlıkları çarpılarak her bir bağlantı için değer bulunur. Her bağlantının değeri giriş yaptığı sinir hücresi için toplanarak toplam değeri bulunur. Daha sonra bu toplam değeri sigmoid, düzeltilmiş doğrusal birim (ReLU) vb. aktivasyon fonksiyonlarına uğrayarak nihai sonuç değeri diğer katmandaki sinir hücrelerine gönderilmektedir. Literatürde sıklıkla kullanılan ReLU aktivasyon fonksiyonu girişi negatif bir değer ise sıfır değerine eşitlemektedir. Bu fonksiyonun yaptığı işlem Denklem (2)'de gösterilmektedir.

$$ReLU(x) = \max(0, x), \tag{2}$$

Sinir hücreleri arasındaki ağırlıklar model sonucunu etkiler ve derin öğrenme modellerinin eğitilmesindeki amaç bağlantılardaki doğru ağırlık parametrelerini bulmaktır. Modeller eğitilirken farklı optimizasyon algoritmaları kullanılmaktadır. Bu algoritmalar; stokastik gradyan inişi, Adam, RMSprop vb. algoritmalardır. Adam optimizasyon algoritması derin öğrenme yöntemlerinde yaygın olarak kullanılan algoritmalardan bir tanesidir [18].

Zaman içerisinde belirli amaçlar doğrultusunda birçok sinir ağı çeşitleri tasarlanmıştır. Pekiştirmeli öğrenme yaklaşımı ile en çok kullanılan sinir ağı çeşitleri ise evrişimli sinir ağları (CNN), otokodlayıcılar (auto-encoders) ve tekrarlayan sinir ağlarıdır (RNN) [19]. Evrişimli sinir ağları piksel tabanlı ortamda çalışan modellerin eğitiminde kullanılmaktadır.

3.2.1 Evrişimli sinir ağları

Sinir ağlarının içerisinde evrişim adı verilen işlemi kullanarak özel bir ağ yapısı geliştirilmiştir. LeCun vd. [20] tarafından yapılan çalışma ile oluşturulan LeNet-5 adlı evrişimli sinir ağı mimarisi görüntü verileri üzerinde başarı göstererek diğer çalışmaların öncüsü olmuştur. Evrişimli sinir ağları, giriş verisi üzerinde işlemler yaparak bu veriyi hesaplama maliyeti açısından daha uygun bir duruma getirirken verinin önemli özelliklerini haritalayan bir yapı oluştururlar. Evrişimli sinir ağlarının temel bileşenleri sırası ile evrişim işlemi, aktivasyon fonksiyonu ve biriktirme (pooling) işlemidir [17].

Evrişim işleminde çekirdek (kernel) ve adım (stride) adı verilen iki parametre kullanılmaktadır. Çekirdek bir matris yapısı olarak giriş verisi üzerinde dolaşmaktadır. Adım parametresi, çekirdeğin giriş verisi matrisi üzerinde yatay ve dikey eksende atlama büyüklüğünü belirler. Evrişim işlemi olarak kullanılabilecek matematiksel yapı Denklem (3)'te verilmiştir.

$$S(i,j) = \sum_{m} \sum_{n} I(i+m, j+n) K(m,n), \tag{3}$$

S(i, j) evrişim işlemi sonucu oluşan sıkıştırılmış matristeki *(i, j)* konumundaki sonuçtur. *I* giriş verisini, *K* ise çekirdeği temsil etmektedir. Evrişim işlemindeki satır ve sütun indisleri ise sırası ile *m* ve *n* harfleri ile temsil edilmektedir. Evrişim işlemi sonrası giriş verisinden çıkan

sonuçlar öznitelik haritası olarak da adlandırılmaktadır. İkinci aşamada ise Denklem (2)'de gösterilen ReLU aktivasyon fonksiyonu vb. bir aktivasyon fonksiyonu veriler üzerinde kullanılmaktadır. Daha sonra hesaplama verimliliği vb. amaçlar için genellikle biriktirme adı verilen işlem ağ mimarisinde kullanılır. Bu işlem ile çıkan öznitelik haritası daha da küçültülmektedir. Biriktirme işlemi için literatürde kullanılan en büyükleri biriktirme veya ortalama biriktirme yöntemleri vardır. Sonuç olarak, evrişimli sinir ağından geçerek işlenen giriş verisi sonraki katmanlara kullanılmak üzere gönderilmektedir.

3.3 Derin pekiştirmeli öğrenme

Pekiştirmeli öğrenmede aracı politikaları, derin öğrenme mimarileri ile temsil edilebilir. Derin sinir ağları MDP çerçevesi içerisinde eğitilmektedirler. Son 10 yıl içerisinde farklı pekiştirmeli öğrenme tarzlarını benimseyen güçlü derin pekiştirmeli öğrenme algoritmaları tasarlanmıştır. Bu kısımda derin Q-ağları (DQA), avantaj aktör kritik (A2K) ve proksimal politika optimizasyonu (PPO) olmak üzere çalışmada kullanılan 3 derin pekiştirmeli öğrenme algoritması anlatılacaktır. Algoritmaların, hangi eylem uzayında çalışabileceği, politika tabanlı veya politikadan bağımsız oluşu, eylem-durum değeri veya avantaj değeri tabanlı iyileştirme yapması açısından birbirine göre karşılaştırması Tablo 1'de verilmiştir.

Tablo 1. Derin pekiştirmeli öğrenme algoritmalarının karşılaştırılması

Özellikler	DQA	AAK	PPO
Fonksiyon Yakınsaması	✓	✓	✓
Modelden bağımsız	✓	✓	✓
Kesikli eylem uzayı	✓	✓	✓
Sürekli eylem uzayı	-	✓	✓
Politika tabanlı öğrenme	-	✓	✓
Politikadan bağımsız öğrenme	✓	-	-
Durum-Eylem (Q) değeri tabanlı	✓	-	-
Avantaj (A) hesabı tabanlı	-	✓	✓

3.3.1 Derin Q-ağları yöntemi

Mnih vd. [21] tarafından yapılan çalışma ile pekiştirmeli öğrenme aracısının politika öğrenimi için evrişimli sinir ağları kullanımı önerilmiştir. Bu yöntem Watkins ve Dayan [22] tarafından önerilen Q-öğrenme algoritmasını temel almaktadır. Derin Q-ağları (DQA) yönteminde politika olarak kullanılan sinir ağı, ham piksel verilerini girdi olarak alırken ağın çıktısı ise durum – eylem çiftinin değer fonksiyonudur. Bu ağı eğitmek için bir gradyan inişi vb. optimizasyon algoritması ile Q-öğrenme algoritması yaklaşımı kullanılabilmektedir. Optimizasyon sürecinde kayıp fonksiyonunda uygulanan gradyan işleminin hesaplanması Denklem (4)'te verilmiştir.

$$\nabla_{\theta_i} L_i(\theta_i) = E_{s,a\sim\rho(\cdot);s\sim\epsilon}\left[\left(r + \gamma \max_{a'} Q\,(s', a'; \theta_{i-1}) - Q(s, a; \theta_i)\right)\nabla_{\theta_i} Q(s, a; \theta_i)\right], \tag{4}$$

Bu işlem her bir döngüde hedef değer y kullanmaktadır. E beklenen değeri, r ödül değerini, γ indirim faktörünü ve Q fonksiyonları ise θ sinir ağı parametrelerine göre göre eylem – durum değerlerini göstermektedir. Denklem (4) içerisindeki hedef değerin hesaplanması Denklem (5)'te gösterilmektedir.

$$y_i = E_{s'\sim\epsilon}\left[r + \gamma \max_{a'} Q\,(s', a'; \theta_{i-1})\middle| s, a\right], \tag{5}$$

DQA yönteminin fayda sağlayan özelliklerinden biri de deneyim tekrarı verilen yapıyı kullanmasıdır. Aracının yaptığı eylem, eylemi sonucundaki aldığı ödül ve bir sonraki durumu geçiş (transition) adı verilen yapıları oluştururlar. Birçok geçiş deneyimi deneyim tekrarı belleğinde depolanır ve DQA yöntemi bu geçişleri gradyan güncellemelerinde tekrarlı şekilde kullanırlar. Bu şekilde veri kullanımı açısından daha verimli bir yöntem tasarlanmıştır.

Mnih vd. [23] yaptıkları diğer çalışma ile DQA yönteminin farklı bir versiyonunu tasarlamışlardır. Bu çalışmada farklı olarak hedef Q-ağı adı verilen bir yapı kullanılmıştır. Hedef Q-ağı asıl Q-ağından farklı olarak algoritmada her adım yerine belirli aralıklarla güncellenmektedir. Bu şekilde hedef değerlerin daha az sıklıkla değiştirilmesi sağlanarak öğrenme süreci verimliliği artırılmaktadır.

3.3.2 Avantaj aktör kritik yöntemi

Mnih vd. [24] tarafından yapılan çalışmada derin pekiştirmeli öğrenmede asenkron olarak çalışan 4 adet yaklaşım gösterilmiştir. Bunlar; tek-adım SARSA, tek-adım Q-öğrenme, n-adım Q-öğrenme ve avantaj aktör kritik yönteminin çok iş parçacıklı asenkron çalışan yöntemleridir. Yöntemler arasında asenkron avantaj aktör kritik (A3K) en kapsamlı ve iyi performans gösteren yöntemdir. Yöntem deneyim tekrarı mekanizmasını kullanmadığı için bellek açısından verimli çalışmaktadır. Çalışmada, deneyim tekrarı mekanizmasının kullanmamasına karşın benzer bir faydadan yararlanmak için pekiştirmeli öğrenme ortamının birden fazla kopyası çok çekirdekli merkezi işlem birimi (CPU) üzerinde birden fazla iş parçacığı kullanarak aynı anda çalıştırılmaktadır. Bu şekilde CPU donanımını kullanarak grafik işlem birimi (GPU) kullanımından kaçınıldığı için donanım yönünden de avantaj sağlanması yöntemin ayrı bir özelliği olmaktadır.

A3K yönteminde asenkron çalışan iş parçacıklarının senkron çalışması ile avantaj aktör kritik (A2K) yöntemi elde edilmektedir. Burada iki önemli yapı olarak aktör ve kritik vardır. Aktör bir derin sinir ağı olarak aracının politikasını belirleyen fonksiyon, kritik ağı ise mevcut durum değerini belirleyen fonksiyondur. Aktör ve kritik sinir ağlarının parametreleri sırası ile θ' ve θ'_v olarak gösterilmektedir. Yönteme göre bu ağ parametrelerini güncellerken kullanılan

gradyan hesaplamaları Denklem (6) ve Denklem (7)'de verilmiştir.

$$\theta' : d\theta \leftarrow d\theta + \nabla_{\theta'} \log \pi\,(a_i|s_i;\theta')\big(R - V(s_i;\theta'_v)\big), \qquad (6)$$

$$\theta'_v : d\theta_v \leftarrow d\theta_v + \frac{\partial\big(R - V(s_i;\theta'_v)\big)^2}{\partial\theta'_v}, \qquad (7)$$

Burada π politika fonksiyon, R bir bölüm içerisindeki getiri ve $V(s_i;\theta'_v)$ durum s_i için değer fonksiyonudur.

3.3.3 *Proksimal politika optimizasyonu*

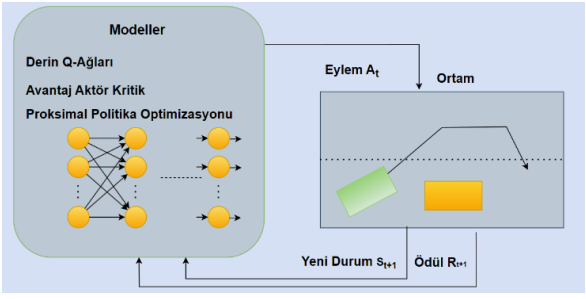
Schulman vd. [25] tarafından yapılan çalışmada ölçeklenebilirlik, veri verimliliği ve hiper-parametre değişikliklerine kırılganlık yönünden mevcut algoritmalara göre daha iyi bir yöntem geliştirilmesi amaçlanmıştır. Proksimal politika optimizasyonu (PPO) yönteminde kırpılmış amaç fonksiyonu ile önceki yöntemlerin olumsuz etkilendiği tek seferde büyük politika güncellemelerinin engellenmesi amaçlanmıştır. Kırpılmış amaç fonksiyonu Denklem (8)'de gösterilmektedir.

$$L^{\text{CLIP}}(\theta) = \hat{E}_t\big[\min\big(r_t(\theta)\widehat{A_t}, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\widehat{A_t}\big)\big], \qquad (8)$$

Burada $\widehat{A_t}$ bir t anında eylemi gerçekleştirmenin değeri olarak hesaplanan avantaj fonksiyonudur. Çok fazla değişim farkı olmaması amaçlanan yeni ve eski politikalar arasındaki fark $r_t(\theta)$ olarak temsil edilir. Bu ifadenin hesaplanmasında yeni ve eski politikalar için sırası ile θ ve θ_{old} ağ parametreleri kullanılmaktadır. Kırpılma işlemi için ϵ sabit hiper-parametre olarak kullanılır. Sonuç olarak yöntemde bu amaç fonksiyonu ile birlikte tek yinelemede büyük politika değişimleri engellenerek daha istikrarlı bir öğrenme süreci elde edilir.

3.4 *Otonom araç geçme ortamlarının kurulumu*

Otonom aracın yapması gereken davranışlar hızlanma, şerit değiştirme, öndeki aracı geçme ve güvenli şekilde sağ şeride geri gelmesidir. Otonom aracın ortamdaki MDP yapısı Şekil 1'de gösterilmiştir.



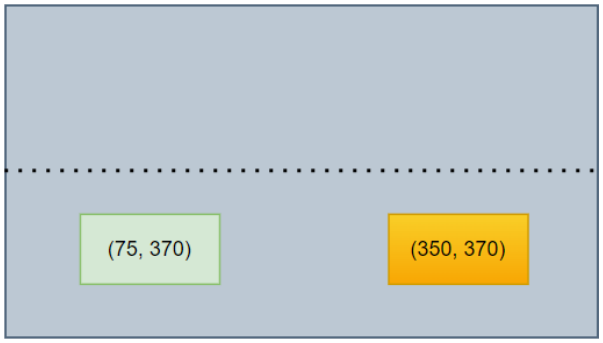
Şekil 1. Öndeki aracı geçmenin MDP çerçevesi

Şekil 1'deki tüm özellikleri sağlayan ana ortam ile mevcut görevi alt aşamaya ayıran müfredat öğrenme alt ortamı olmak üzere iki ortam kurulmuştur. Ortamların kurulumu ve deneylerin gerçekleştirilmesi aşamasında x86_64 mimari ile 16 çekirdeğe sahip AMD Ryzen 7 5800H işlemci, 16 gigabayt bellek ve 6 gigabayt hafızaya sahip GeForce RTX 3060 Mobile marka GPU özelliklerine sahip bilgisayar kullanılmıştır. Çalışma Python programlama dili ile Anaconda [26] platformu üzerinde gerçekleştirilmiştir. MDP çerçevesi ile ortamların kurulumunda OpenAI tarafından geliştirilen OpenAI GYM [27] API kullanılmıştır. Daha sonra oluşan hata çözme gereksinimleri nedeni ile OpenAI GYM uzantısı olan ve Farama-Foundation tarafından geliştirilen Gymnasium [28] çalışmaya eklenmiştir. Python programlama dili ile piksel tabanlı işlemlerin gerçekleştirilmesi amacı ile OpenCV-Python [29] ve sayısal birtakım işlemlerin gerçekleştirilmesi için NumPy [30] kütüphaneleri kullanılmıştır.

3.4.1 *Otonom araç geçme ana ortamının kurulumu*

Ortamın tuval boyutu (600, 800, 3) boyutudur. Değerler sırası ile piksel cinsinden yüksekliği, piksel cinsinden genişliği ve kırmızı, yeşil ve mavi renk boyutlarını temsil etmektedir. Hesaplama maliyetini azaltmak için aracının gözlem uzayı (150, 200, 3) olarak yeniden boyutlandırılmıştır.

Ortamın başlangıç konumunda otonom aracı temsil eden araç (x, y) eksenlerinde (75, 370) konumunda, otonom aracın geçeceği araç ise (350, 370) konumunda başlamaktadır. Otonom aracın hedefi 275 piksellik farkı kapatarak diğer aracı geçmektir. Başlangıca ait örnek temsili gösterim Şekil 2'de gösterilmektedir.

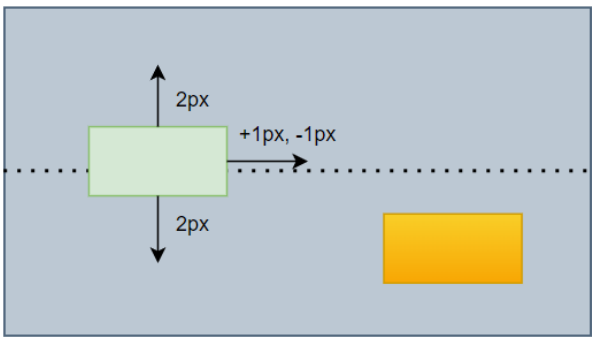


Şekil 2. Otonom araç geçme ortamının başlangıcı

Otonom araç dikey eksen y ekseninde 160 ile 370 konumları arasında hareket etmektedir. Bu büyüklükleri aşacak bir yön değiştirme eylemi gerçekleştirdiğinde kırpma işlemi yapılmaktadır. Yatay eksen x ekseninde ise sınırlandırma 0 ve 740 piksel değerleri arasındadır. Aracının yaptığı her eylem 1 zaman adımına eşittir. Bir bölüm en fazla 500 adım sürmektedir. Geçilmesi gereken araç sadece yatay eksende 10 zaman adımında 1 ileri hareket etmektedir. Bir bölüm çarpışma, öndeki aracın başarı ile geçilmesi, bölüm süresinin dolması ya da aracının yatay eksen x ekseninde 740 pikseli aşması ile sona ermektedir.

Ortamda bulunan otonom araç her zaman adımında 5 eylemden sadece 1 tanesini seçerek mevcut hızını ve direksiyon yönünü değiştirebilmektedir. Bu nedenle aracının eylem uzayı kesikli uzaydır. Bu eylemler hızını ileriye doğru 1 piksel artırma ya da azaltma veya direksiyon yönü için sağa ya da sola 2 piksel yön değiştirme şeklindedir. Son eylem ise aracın zaman adımında hiçbir eylemde bulunmamasıdır.

Bölüm içerisinde mevcut hız değeri 1 ve 10 piksel değerleri arasında tutulmaktadır. Bu hız aralıklarının dışına çıkılma durumlarında kırpma işlemi yapılmaktadır. Benzer şekilde direksiyon yönü için sınırlandırma -14 ve +14 piksel değerleri arasındadır. 5 eylemden 4 tanesi Şekil 3'te temsili olarak gösterilmektedir.



Şekil 3. Otonom aracın hız ve yön eylemleri

Otonom araç geçme görevi için her zaman adımında kullanılan ödül fonksiyonlarında $r_{adım}$ değeri -1, çarpışma anında kullanılacak $r_{çarpışma}$ değeri -100, iki aracın yan yana gelme anında kullanılacak $r_{yanyana}$ değeri 100 ve öndeki aracı tam olarak geçme durumunda kullanılacak $r_{sollama}$ değeri ise 200 olarak kullanılmaktadır. Aracının eylemde bulunduğu her zaman adımında çarpışma gerçekleşirse Denklem (9), yan yana gelme durumu gerçekleşirse Denklem (10) veya o adımda araç geçme görevi başarı ile tamamlanırsa Denklem (11) kullanılmaktadır.

$$r_{ödül} = r_{adım} + r_{çarpışma}, \quad (9)$$

$$r_{ödül} = r_{adım} + r_{yanyana}, \quad (10)$$

$$r_{ödül} = r_{adım} + r_{sollama}, \quad (11)$$

Her zaman adımında araçlar arasındaki mesafeler (x, y) konumları alınarak ölçülmektedir. Aynı zaman adımında yatay eksen x ekseninde iki araç arasındaki mesafe 110 pikselden daha az ve yatay eksen y ekseninde iki araç arasındaki mesafe 43 pikselden daha az olursa iki araç arasında çarpışma işlemi gerçekleşmiş olur. Araçların yan yana gelmesi durumu ise otonom aracın yatay eksen x eksenindeki konum değerinin sayısal olarak diğer araçtan daha yüksek olması ve otonom aracın dikey eksen y eksenindeki konumunun diğer araçtan 100 pikselden daha büyük bir değerde olması ile gerçekleşir. Bir bölüm içerisinde yan yana gelme durumuna sadece bir defa ödül verilmektedir. Otonom araç için ana hedef olan öndeki aracı geçme durumu ise otonom aracın yatay eksen x eksenindeki konumunun diğer araçtan 50 pikselden daha fazla olması ve dikey eksen y eksenindeki konumları arasındaki farkın 5 pikselden daha az olması ile gerçekleşmektedir.

3.4.2 Müfredat öğrenme alt ortamının kurulumu

Bu ortamda tuval boyutu, gözlem ve eylem uzayı, ödül fonksiyonları vb. ana ortam için anlatılan ayarlar aynıdır. Bu ortamın ana ortamdan farkı, ana hedef otonom aracı geçme görevinin bu ortamda bulunmaması ve Denklem (11)'in kullanılmamasıdır. Bu ortamdaki nihai hedef iki aracın yan yana gelmesidir. Bu ortamda yan yana gelindiğinde bölüm sonlanmaktadır.

3.5 Derin pekiştirmeli öğrenme modellerinin tasarımı

Bu kısımda 3 algoritma için 4'er tane olmak üzere eğitilen 12 modelin hiper-parametre tanımlamaları ve derin sinir ağları açıklanacaktır. Modellerin oluşturulmasında PyTorch çerçevesi tabanlı Stable Baselines3 [31] kütüphanesi kullanılmıştır. Stable Baselines3 derin pekiştirmeli öğrenme algoritmalarının istenilen parametrelerde uygulanmasını sağlayan kütüphanedir.

Yöntemlerin her birinde 2 hiper-parametre ayarı oluşturulmuştur. 2 ayar ile oluşturulan 2 model ana ortamda 1000000 zaman adımı eğitilmiştir. Diğer 2 model ise müfredat öğrenme alt ortamında ilk olarak 500000 zaman adımında eğitilmiştir. Daha sonra eğitilen 2 modelin eğitimine 500000 zaman adımı daha ana ortam üzerinde devam edilerek müfredat öğrenme yaklaşımına uğramış modeller oluşturulmuştur. Sonuç olarak 2 tanesi müfredat öğrenimi yaklaşımına uğramış ve 1000000 zaman adımı boyunca eğitilmiş 4'er adet DQA, A2K ve PPO modelleri oluşturulmuştur.

3.5.1 Derin Q-ağları modellerinin tasarımı

DQA modelleri için iki hiper-parametre ayarı Tablo 2'de gösterilmektedir.

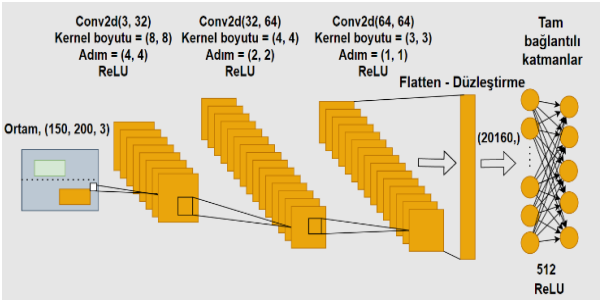
Tablo 2. DQA modelleri için kullanılan 2 farklı ayar

Hiper-parametreler	Ayar 1	Ayar 2
Policy	CnnPolicy	CnnPolicy
Learning_rate	0.00025	0.000001
Buffer_size	50000	50000
Learning_starts	50000	50000
Batch_size	32	32
Tau	1.0	1.0
Gamma	0.99	0.99
Train_freq	4	4
Gradient_steps	1	1
Replay_buffer_kwargs	-	-
Optimize_memory_usage	False	False
Target_update_interval	10000	10000
Exploration_fraction	0.1	0.1
Exploration_initial_eps	1.0	1.0
Exploration_final_eps	0.05	0.05
Max_grad_norm	10	10
Policy_kwargs	-	-
Seed	None	None
Device	cuda	cuda

Tablo 2'deki policy politikayı, learning_rate optimizasyonda her adımda ilerleme büyüklüğünü, buffer_size deneyim tekrarı mekanizmasında depolanacak veri büyüklüğünü, learning_starts öğrenme başlamadan önceki adım sayısını, batch_size gradyan güncellemelerindeki veri topluluğu büyüklüğünü, tau yumuşak güncelleme katsayısını, gamma indirim faktörünü, train_freq modellerin güncellenme sıklığını, gradient_steps her bir güncellemede atılacak olan gradyan güncelleme adım sayısını, replay_buffer_kwargs deneyim tekrarı mekanizması deposu olan

tekrarlama arabelleğine gönderilen parametreleri, optimize_memory_usage tekrarlama arabelleğinin belleği verimli kullanma durumunu, target_update_interval hedef ağın güncellenme sıklığını, exploration_fraction keşif oranı azaltımının zaman dilim aralığını, exploration_initial_eps rastgele eylem olasılığının başlangıç değerini, exploration_final_eps rastgele eylem olasılığının son değerini, max_grad_norm gradyan kırpma için en yüksek değeri, policy_kwargs politika oluşturulurken eklenebilecek parametreleri, seed rastgele üreteç sabit değerini ve device ise kodun çalıştırılacağı donanımı belirlemektedir.

Stable Baselines3 üzerindeki varsayılan derin sinir ağı DQA mimarisi 4 modelde de aynı şekilde kullanılmıştır. Hedef Q-ağı için de mimari aynı olup DQA modelleri için ağ mimarisi Şekil 4'te gösterilmektedir. DQA'da optimizasyon sürecinde varsayılan olarak Adam algoritması kullanılmaktadır.



Şekil 4. DQA modelleri için ağ mimarisi

Şekilde gösterilen ağ yapısında, giriş, çıkış kanal sayıları, çekirdek ve adım parametreleri değiştirilmemiştir [32]. Katmanlar içerisinde ReLU aktivasyon fonksiyonu kullanılmaktadır. Ağ içerisinde giriş verisi tam bağlantılı katmanlar içerisinde kullanılmadan önce (20160,) uzunluğunda vektöre dönüştürülmektedir. Bu dönüşüm için düzleştirme adı verilen katman kullanılmıştır.

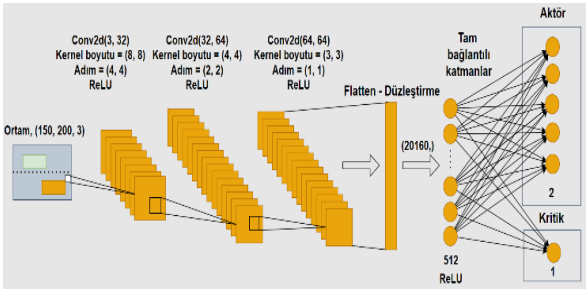
3.5.2 Avantaj aktör kritik modellerinin tasarımı

A2K modelleri için iki farklı hiper-parametre ayarı Tablo 3'te gösterilmektedir. DQA yöntemi ile ortak olan hiper-parametre tanımlamaları olsa da A2K yöntemi için farklı olan hiper-parametreler vardır. Farklı olarak n_envs paralelde çalışan kopya ortamlarının sayısı, n_steps her bir güncellemede alınacak eylem sayısını, gae_lambda avantaj tahmininde önyargı ve varyans arasındaki değişim faktörünü, ent_coef ve vf_coef kayıp fonksiyonu hesaplanırken sırası ile entropi ve değer fonksiyonu katsayısını, use_sde genelleştirilmiş duruma bağlı keşif (gSDE) mekanizmasının kullanılma durumunu, sde_sample_freq kullanılma durumunda gürültü matrisinin kullanılma sıklığını ve normalize_advantage ise avantaj tahmininde normalizasyon yapılma durumunu belirlemektedir.

Stable Baselines3 üzerindeki varsayılan derin sinir ağı A2K mimarisi 4 modelde de aynı şekilde kullanılmıştır. A2K modelleri için ağ mimarisi Şekil 5'te verilmiştir. A2K'de optimizasyon sürecinde RMSprop yöntemi kullanılmaktadır.

Tablo 3. A2K modelleri için kullanılan 2 farklı ayar

Hiper-parametreler	Ayar 1	Ayar 2
Policy	ActorCriticCnnPolicy	ActorCriticCnnPolicy
Learning_rate	0.000001	0.00025
N_envs	1	1
N_steps	32	32
Gamma	0.99	0.99
Gae_lambda	1.0	1.0
Ent_coef	0.0	0.0
Vf_coef	0.5	0.5
Max_grad_norm	0.5	0.5
Use_sde	False	False
Sde_sample_freq	-1	-1
Normalize advantage	False	False
Policy_kwargs	Optimizer: RMSprop Aplha:0.99 Eps: 0.00001 Weight decay: 0	Optimizer: RMSprop Alpha: 0.99 Eps: 0.00001 Weight decay: 0
Seed	None	None
Device	cuda	cuda



Şekil 5. A2K modelleri için ağ mimarisi

A2K'nin evrişimli sinir ağı yapısı DQA modeli ile aynıdır. Farklı olarak aktör ve kritik katmanları son aşamada kullanılmaktadır. Katmanlar içerisinde ReLU aktivasyon fonksiyonu kullanılmaktadır. Aktör ağı ile 5 eylem için eylem değerleri hesaplanırken kritik katmanı ile mevcut durum değeri hesaplanmaktadır.

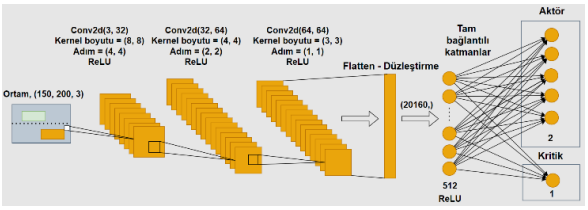
3.5.3 Proksimal politika optimizasyonu modellerinin tasarımı

PPO modelleri için iki farklı hiper-parametre ayarı Tablo 4'te verilmiştir. DQA ve A2K modellerinden farklı olarak bu yöntemde kullanılan n_epochs parametresi kayıp fonksiyonu güncellenirken aynı veri topluluğunun tekrar kullanılma sayısını, clip_range kırpma büyüklüğünü, clip_range_vf değer fonksiyonu için kırpma büyüklüğünü ve target_kl kullanılıyor ise güncellemeler arasındaki KL uzaklığını tanımlar.

Stable Baselines3 üzerindeki varsayılan derin sinir ağı PPO mimarisi 4 modelde de aynı şekilde kullanılmıştır. PPO modelleri için ağ mimarisi Şekil 6'da verilmiştir. PPO'da optimizasyon sürecinde Adam algoritması kullanılmaktadır.

Tablo 4. PPO modelleri için kullanılan 2 farklı ayar

Hiper-parametreler	Ayar 1	Ayar 2
Policy	ActorCriticCnnPolicy	ActorCriticCnnPolicy
Learning_rate	0.000001	0.00025
N_envs	1	1
N_steps	2048	2048
Batch_size	32	32
N_epochs	10	10
Gamma	0.99	0.99
Gae_lambda	0.95	0.95
Clip_range	Sabit fonksiyon	Sabit fonksiyon
Clip_range_vf	None	None
Normalize advantage	True	True
Ent_coef	0.0	0.0
Vf_coef	0.5	0.5
Max_grad_norm	0.5	0.5
Use_sde	False	False
Sde_sample_freq	-1	-1
Target_kl	None	None
Policy_kwargs	-	-
Seed	None	None
Device	cuda	cuda



Şekil 6. PPO modelleri için ağ mimarisi

Buradaki evrişimli sinir ağı, aktör ve kritik mimarisi Şekil 5'te verilen A2K mimarisi ile aynı olup (20160,) uzunluğundaki vektör ile tanımlanan veri tam bağlantılı katmandan geçerek aktör ve kritik katmanlarına beslenmektedir. Katmanlar içerisine ReLU aktivasyon fonksiyonu kullanılmaktadır.

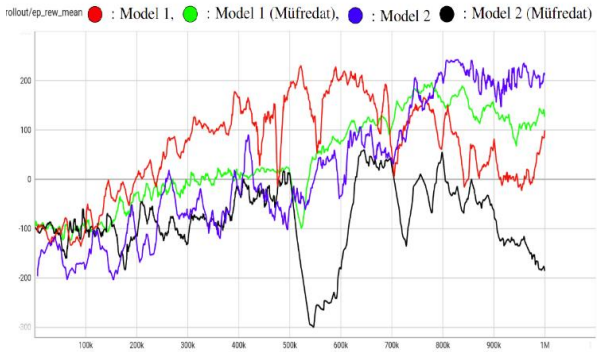
4 Bulgular ve tartışma

Bu çalışmada oluşturulan 12 derin pekiştirmeli öğrenme modelinin tasarlanan özgün otonom araç geçme senaryosunda değerlendirilmesi için iki ölçüm kriteri kullanılmıştır. Bu iki kriter eğitim ve test aşamasıdır. Eğitim kriteri olarak eğitim aşamasında her 100 bölümün ödül ortalaması verileri TensorBoard [33] görselleştirme aracı ile gösterilmektedir. Test aşamasında ise eğitimi tamamlanmış modeller ana ortamda 10 bölüm çalıştırılmıştır. Bu işlem 10 defa tekrarlanarak toplanan toplam ödül 100'e bölünmüş sonuç olarak her bir modelin test aşamasındaki ortalama ödül değeri belirlenmiştir. Tablolardaki Ayar 1 ile oluşturulan modeller sonuçlarda Model 1, Ayar 2 ile oluşturulan modeller Model 2 olarak belirtilmiştir. Müfredat öğrenimi yaklaşımı ile eğitilmiş modeller parantez içerisinde belirtilmiştir.

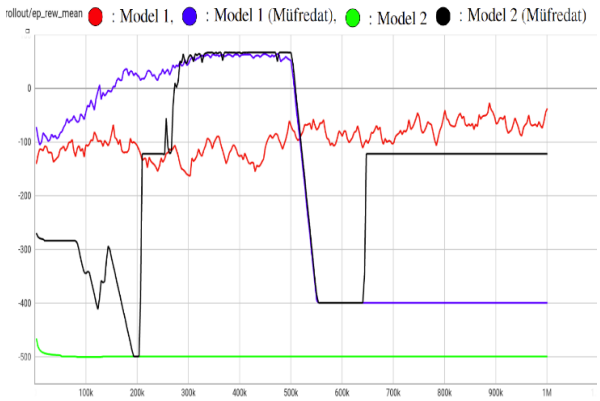
4.1 Eğitim aşaması ödül sonuçları

DQA modelleri için 1000000 zaman adımı boyunca eğitim anında alınan sonuçlar Şekil 7'de gösterilmektedir. Sonuçlara göre Tablo 2'de Ayar 1 ile eğitilen modellerde müfredat öğrenimi yaklaşımının eğitim performansını artırarak daha güçlü bir model sağladığı görülmektedir. Buna karşın Ayar 2 ile eğitilen modellerde müfredat öğrenimi yaklaşımı ile performansın büyük ölçüde düştüğü görülmektedir. Ayar 1 ile Ayar 2 farkını bakıldığında iki ayar arasındaki fark optimizasyon sürecindeki öğrenme oranından kaynaklanmaktadır. Küçük değerdeki öğrenme oranı ile eğitilen model her optimizasyon adımında daha çok öğrenme değişimi yaşayan modelden daha iyi bir eğitim performansı göstermiştir.

A2K modelleri için 1000000 zaman adımı boyunca eğitim sonuçları Şekil 8'de verilmektedir. Sonuçlara göre Tablo 3'te her iki ayarda da ilk 500000 zaman adımında müfredat öğrenme alt ortamında eğitilirken pozitif ödüllere ulaşılmıştır. Daha sonra modellerin eğitimine ana ortamda devam edildiğine 1000000 zaman adımına kadar olan eğitimde modellerin performansı yetersizdir. Eğitim süresince sadece ana ortamda eğitilen modeller hiçbir zaman pozitif ödüllere ulaşamamıştır. Sonuç olarak A2K modellerinin otonom araç görevinde yan yana gelme senaryosunda başarı elde ettiği görülse de ana görevde başarısız olduğu görülmektedir.

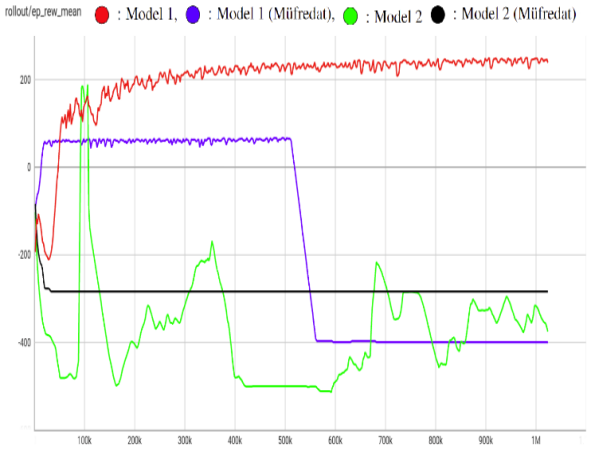


Şekil 7. DQA modellerinin eğitim sonuçları



Şekil 8. A2K modellerinin eğitim sonuçları

PPO modelleri için 1000000 zaman adımı boyunca eğitim anında alınan sonuçlar Şekil 9'da gösterilmektedir. Sonuçlara göre Tablo 4'te Ayar 1 ile eğitilen model başarılı bir eğitim performansı ile ortalama bölüm ödülünde 200 değerinin üzerine çıkmıştır. DQA modelinde olduğu gibi öğrenme oranı daha yüksek olan model büyük bir fark ile başarısız bir eğitim performansı göstermiştir. Müfredat öğrenimi her iki ayarda da başarısız olmuştur. Ayar 1 ile eğitilen model ilk 500000 zaman adımında başarılı olsa da ana ortama geçildiğinde asıl görevde başarısını sürdürememiştir.



Şekil 9. PPO modelleri eğitim sonuçları

Her üç algoritmada en iyi eğitim performansını gösteren modellerin eğitim sonuçları Şekil 10'da gösterilmektedir. Tablo 4'te Ayar 1 ile eğitilen PPO modeli en iyi eğitim performansını göstermiştir. DQA yönteminde bir ayarda performans artımını sağlasa da en iyi 3 model arasında müfredat öğrenimi yaklaşımı ile eğitilmiş bir model yoktur.

4.2 Test aşaması ödül sonuçları

DQA modelleri için test sonuçları Tablo 5'te verilmiştir.

Tablo 5. DQA modellerinin test sonuçları

Model 1	Model 1 (Müfredat)	**Model 2**	Model 2 (Müfredat)
11.53	165.29	**202.54**	-22.17

A2K modelleri için test sonuçları Tablo 6'da verilmiştir.

Tablo 6. A2K modellerinin test sonuçları

Model 1	Model 1 (Müfredat)	Model 2	Model 2 (Müfredat)
-58.48	-400.00	-500.00	-122.00

PPO modelleri için test sonuçları Tablo 7'de verilmiştir.

Tablo 7. PPO modellerinin test sonuçları

Model 1	Model 1 (Müfredat)	Model 2	Model 2 (Müfredat)
248.18	-400.00	-500.00	-284.00

Test sonuçları eğitim sonuçları ile paralellik göstermektedir. Tablo 4'teki Ayar 1 ile sadece ana ortamda eğitilen PPO modeli oluşturulan ödül fonksiyonuna göre yaklaşık olarak alabileceği en yüksek ödülü alarak tam bir otonom araç geçme görevinde başarılı olmaktadır. Ancak, sadece optimizasyon sürecindeki öğrenme oranını artırarak eğitilen diğer PPO modeli mevcut görevde başarısız olmuştur. DQA modelleri içerisinde Tablo 2'deki Ayar 1 ile oluşturulan modeller içerisinde müfredat öğrenimi yaklaşımı ile oluşturulan model bölüm ortalama ödülünü yaklaşık olarak 155 artırmış ve başarılı bir otonom araç geçme performansı göstermiştir. Tablo 6'daki A2K modellerinin test sonuçlarına bakıldığında hiçbir A2K modeli pozitif ödüle ulaşamamış ve tüm A2K modelleri başarısız olmuştur.

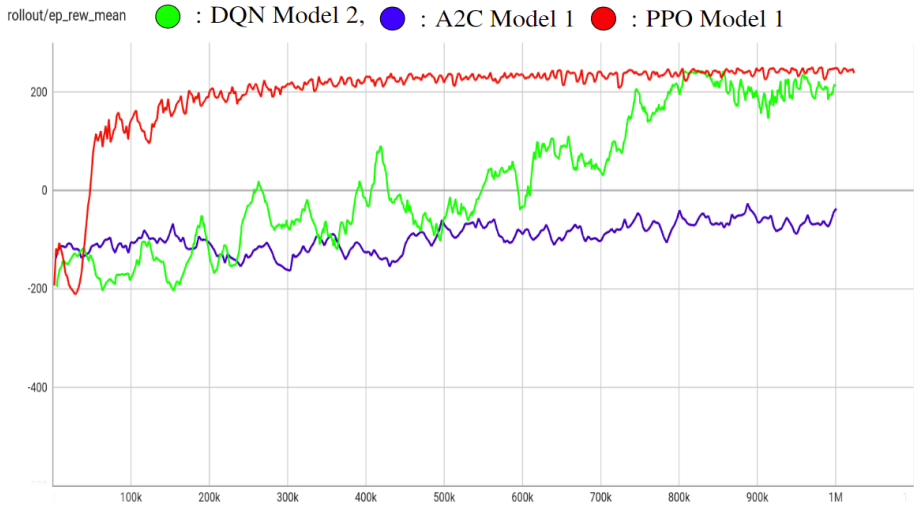
5 Sonuçlar

Otonom araç çalışmaları trafikte insan kaynaklı hatalar sonucu meydana gelen zararların azaltılması ve otonom araçların diğer sayılabilecek faydaları yönünden önemlidir. Bu doğrultuda yapılan çalışmada, ayrık eylem uzayında, otonom araç geçme görevinde, DQA ve PPO yöntemleri ile bazı hiper-parametre ayarlarında başarılı modeller elde ederek yapay zekâ tabanlı yöntemlerin otonom araç görevlerinde kullanılması fikrine katkı sağlanmıştır. Müfredat öğrenme yaklaşımı ile aynı ayarda daha iyi bir DQA modeli elde ederek yaklaşımın olumlu etkisi gözlenmiştir. Kullanılan ödül fonksiyonu keyfi değerler ile oluşturulmuş basit bir ödül fonksiyonudur. Ödül fonksiyonunun otonom araç geçme görevi için iyileştirilmesinin daha iyi modellerin eğitimini sağlayacağı öngörülmektedir.

Öğrenme oranı hiper-parametresinin değiştirilmesi ile aynı algoritma modellerinin başarı oranında büyük değişimler olduğu görülmektedir. Bu sonuç, algoritmaların parametre değişkenliğinden açıkça etkilendiğini göstermektedir.

A2K modellerinin başarısız olması çalışmada incelenmesi gereken diğer bir sonuçtur. GPU yerine CPU donanımının kullanımı veya kopya ortamların sayısı vb. diğer hiper-parametrelerin değişimi ile başarılı A2K modellerinin de eğitilebileceği öngörülmektedir.

Çalışmada tasarlanan ortam yayalar, hava durumu, şerit tipleri, levhalar, diğer araçlar ve farklı engeller vb. trafik ortamının birçok dinamiğinden yoksundur. Bu etmenler ile birlikte ayrıca sürekli eylem uzayı biçiminde davranışın ele alınması ile yapılabilecek daha kapsamlı ortamlarda derin pekiştirmeli öğrenme ve müfredat öğrenme yaklaşımlarının etkinliği ilerleyen çalışmalarda incelenebilecektir.



Şekil 10. Algoritmaların en iyi performans gösteren modelleri

Teşekkür

Bu çalışma Erciyes Üniversitesi Bilimsel Araştırma Projeleri Birimi tarafından FKB-2019-9388 proje numarası ile desteklenmiştir. Bu çalışmadaki sonuçlar Yasin Atılkan'ın yüksek lisans tezi kapsamında yaptığı deneylere ait verileri içermektedir.

Çıkar çatışması

Yazarlar çıkar çatışması olmadığını beyan etmektedir.

Benzerlik oranı (Turnitin): %8

Kaynaklar

[1] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, S. Yogamani, and P. Peréz, Deep reinforcement learning for autonomous driving: a survey. IEEE Transactions on Intelligent Transportation Systems, 23(6), 4909-4926, 2021. https://doi.org/10.1109/TITS.2021.3054625.

[2] Sürücü Eğitimi. https://www.taksimsurucukursu.com/assets/pdf/surucuegitimkitabi.pdf, Erişim: 11 Aralık 2023.

[3] J. Janai, F. Güney, A. Behl, and A. Geiger, Computer vision for autonomous vehicles: Problems, Datasets and State of the Art. 2017, *arXiv: 1704.05519*. https://doi.org/10.48550/arXiv.1704.05519.

[4] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, Deepdriving: Learning affordance for direct perception in autonomous driving. Proceedings of the IEEE International Conference on Computer Vision, 2722-2730, Santiago, Chile, 2015.

[5] D. Loiacono, A. Prete, P. L. Lanzi, and L. Cardamone, Learning to overtake in TORCS using simple reinforcement learning. IEEE Congress on Evolutionary Computation, pp. 1-8, Barcelona, Spain, 2010.

[6] D. C. K. Ngai and N. H. C. Yung, A multiple-goal reinforcement learning method for complex vehicle overtaking maneuvers. IEEE Transactions on Intelligent Transportation Systems, 12(2), 509-522, 2011. https://doi.org/10.1109/TITS.2011.2106158.

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

[8] T. Xia and Z. Han, Path planning using reinforcement learning and objective data. Master's Thesis, University of Gothenburg, Gothenburg, Sweden, 2017.

[9] M. Kaushik, V. Prasad, K. M. Krishna, and B. Ravindran, Overtaking maneuvers in simulated highway driving using deep reinforcement learning. 2018 IEEE Intelligent Vehicles Symposium (IV), pp. 1885-1890, Changshu, China, 2018.

[10] X. Li, X. Qiu, J. Wang, and Y. Shen, A deep reinforcement learning based approach for autonomous overtaking. 2020 IEEE International Conference on Communication Workshops (ICC Workshops), pp. 1-5, Dublin, Ireland, 2020.

[11] Y. Song, H. Lin, E. Kaufmann, P. Dürr, and D. Scaramuzza, Autonomous overtaking in gran turismo sport using curriculum reinforcement learning. 2021 IEEE International Conference on Robotics and Automation (ICRA), pp. 9403-9409, Xi'an, China, 2021.

[12] J. Liu, H. Li, Z.Yang, S. Dang, and Z. Huang, Deep dense network-based curriculum reinforcement learning for high-speed overtaking. IEEE Intelligent Transportation Systems Magazine, 15(1), 453-466, 2022. https://doi.org/10.1109/MITS.2022.3174410.

[13] E. Aslan, M. A. Arserim, and A. Uçar, Development of Push-Recovery control system for humanoid robots using deep reinforcement learning. Ain Shams Engineering Journal, 14(10), 102167, 2023. https://doi.org/10.1016/j.asej.2023.102167.

[14] V. Uc-Cetina, N. Navarro-Guerrero, A. Martin-Gonzalez, C. Weber, and S. Wermter, Survey on reinforcement learning for language

processing. Artificial Intelligence Review, 56(2), 1543-1575, 2023. https://doi.org/10.1007/s10462-022-10205-5.

[15] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction. MIT Press, 2018.

[16] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, Curriculum learning. International Conference on Machine Learning, pp. 41-48, Montreal, Canada, 2009.

[17] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, 2016.

[18] D. Soydaner, A comparison of optimization algorithms for deep learning. International Journal of Pattern Recognition and Artificial Intelligence, 34(13), 2020. https://doi.org/10.1142/S0218001420520138.

[19] S. S. Mousavi, M. Schukat, and Enda Howley, Deep reinforcement learning: an overview. In Proceedings of SAI Intelligent Systems Conference (IntelliSys), pp. 426-440, London, UK, 2016.

[20] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278 – 2324, 1998. https://doi.org/10.1109/5.726791.

[21] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, Playing atari with deep reinforcement learning. 2013, *arXiv: 1312.5602*. https://doi.org/10.48550/arXiv.1312.5602.

[22] C. J. Watkins and P. Dayan, Q-learning. Machine Learning, 8, 279-292, 1992. https://doi.org/10.1007/BF00992698.

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

[24] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Harley, T. P. Lillicrap, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In International Conference on Machine Learning, pp. 1928-1937, New York, USA, 2016.

[25] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, Proximal policy optimization algorithms. 2017, *arXiv: 1707.06347*. https://doi.org/10.48550/arXiv.1707.06347.

[26] Anaconda Software Distrubition. https://www.anaconda.org/, Erişim: 18 Temmuz 2023.

[27] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, OpenAI Gym. 2016, *arXiv: 1606.01540*. https://doi.org/10.48550/arXiv.1606.01540.

[28] Gymnasium. https://gymnasium.farama.org, Erişim: 18 Temmuz 2023.

[29] OpenCV-Python. https://pypi.org/project/opencv-python. Erişim: 18 Temmuz 2023.

[30] NumPy. https://numpy.org, Erişim: 18 Temmuz 2023.

[31] Stable-Baselines3. https://www.ai4europe.eu/sites/default/files/2021-06/README_5.pdf, Erişim: 18 Temmuz 2023.

[32] PyTorch Conv2d. https://pytorch.org/docs/stable/generated/torch.nn.Conv2d.html, Erişim: 18 Temmuz 2023.

[33] Tensorflow TensorBoard. https://github.com/tensorflow/tensorboard, Erişim: 18 Temmuz 2023.