PEKİŞTİRMELİ ÖĞRENME KONTROLDE OTOKODLAYICLAR İLE ÖRNEK VERİMLİLİĞİN ARTTIRMA

Burak Er[1]*, Mustafa Doğan*[2]

[1]Kontrol ve Otomasyon Mühendisliği Bölümü
İstanbul Teknik Üniversitesi
erb20@itu.edu.tr

[2]Kontrol ve Otomasyon Mühendisliği Bölümü
İstanbul Teknik Üniversitesi
mustafadogan@itu.edu.tr

Özetçe

Pekiştirmeli öğrenme yöntemlerinde eğitim sırasında çok sayıda örnek gerekmesi zaman kaybına, bütçesel problemlere neden olmakta ve gerçek dünya uygulamalarında risk alınmasına sebep olmaktadır. Bu çalışmada otokodlayıcılar yardımıyla sistemin durumlarının kodlanması ile elde edilen gizli uzay durumlarla birlikte kullanılarak herhangi bir pekiştirmeli öğrenme yönteminin eğitimi için örnek verimliliğinin arttırılması amaçlanmaktadır. Literatürde sadece gizli uzayın kullanıldığı durumlarda bilgi kaybı, sistem dinamiğinin değişmesi durumunda gizli uzayın durumların güncel olmayan bir temsilini içermesi gibi problemlere neden olmaktadır. Bu çalışmada sunulan yeni algoritma ile örnek olarak Derin Q-Ağı (DQN) yönteminin verimliliği arttırılmıştır. Temel DQN ve Otokodlayıcı ile Geliştirilmiş DQN algoritması OpenAI Gym Lunar Lander sisteminde denenerek birbirleriyle karşılaştırılmış, hem örnekleme verimliliğinin arttırdığı hem de önceki çalışmalardaki sorunların üstesinden gelindiği gösterilmiştir.

Abstract

Reinforcement learning algorithms learn efficient strategies through many interactions with the environment. This is particularly problematic in the control of dynamic systems, where each interaction can be expensive, time-consuming and risky. This study presents a novel approach for improving the sample efficiency of reinforcement learning (RL) control of dynamic systems by using states and the latent space of states using autoencoders. The latent space of states is obtained from the encoding of states. In literature, most applications use only the autoencoder's latent space while learning. This approach can cause loss of information, difficulty in interpreting latent space, difficulty in handling dynamic environments and outdated representation. In this study, the proposed novel approach overcomes these problems and enhances sample efficiency using both states and their latent space while learning. It has been demonstrated that the algorithm not only enhances sampling efficiency, but also successfully addresses the issues identified in previous studies.

1. Giriş

Pekiştirmeli öğrenme (RL), makine öğreniminin bir alt kategorisi olup, ajanların çevre etkileşimi yoluyla ardışık kararlar vermeyi öğrenmesini sağlar [9]. Bu ajanlar, zaman içinde kararlarını veya eylemlerini optimize etmek için birikimli ödülü maksimize etmeyi öğrenirler. Bu kavram, oyun oynama, kaynak yönetimi ve kontrol görevleri gibi birçok alanda özellikle etkili olmuştur [6, 8]. Dinamik sistemlerin kontrolü, RL'nin önemli bir potansiyel gösterdiği alanlardan biridir. Dinamik sistemler, robotik, ekonomi ve biyoloji gibi çeşitli alanlarda yaygın olarak görülen sistemlerdir. Sıklıkla karmaşık, doğrusal olmayan davranışları içerirler ve geleneksel yöntemlerle modellemek ve kontrol etmek zordur. Çevre ile etkileşim yoluyla karmaşık davranışları öğrenebilme yeteneği sayesinde, RL bu zorluklara çekici bir yaklaşım sunar. Ancak, dinamik sistem kontrolünde RL'nin uygulanmasındaki büyük bir zorluk, örnekleme verimliliği sorunudur. Etkin bir strateji öğrenmek için RL tekniklerinde sıklıkla çevreyle birçok etkileşim gerekir. Bu durum özellikle her etkileşimin maliyetli, zaman alıcı ve riskli olabileceği dinamik sistemlerde sorun teşkil etmektedir [3].

Otokodlayıcılar, giriş verilerinin kodlanması için kullanılan bir yapay sinir ağı türüdür. Otokodlayıcılar, yüksek boyutlu durumları daha az boyutta bir gizli uzayda temsil etme yeteneği geliştirebilir, bu da RL ajanı için öğrenme görevini potansiyel olarak basitleştirebilir. Ancak, otokodlayıcıların dinamik sistem kontrolünde RL için kullanımı henüz tam olarak keşfedilmemiş bir alanı oluşturur.

Literatürde, halihazırdaki verilerden model bazlı algoritma geliştirme ve bu modelden yeni örnekler üretme veya sadece gizli uzayın kullanılması gibi metodlar kullanılmıştır. Fakat bu metodlar için sadece çalışmanın başlarında olsa bile veri toplanması için sistem üzerinde deney yapılması gerekmektedir. Ayrıca bazı yöntemlerde derin takviyeli öğrenmeyi iyileştirmek için denetimsiz temsil öğrenme teknikleri kullanılmaktadır [1]. Ancak, bu yöntemler genellikle görüntüler gibi yüksek boyut-

lara sahip girişleri işlemek için kullanılır. Ayrıca, bu tekniklerde yalnızca gizli uzay kullanılır. Bu yaklaşım bilgi kaybına, gizli uzayın yorumlanmasındaki zorluklara, dinamik ortamların işlenmesindeki zorluklara ve sistemin değişmesiyle birlikte güncellenmemiş temsillere neden olabilir.

Bu çalışma, otokodlayıcıların dinamik sistem kontrolünde RL'nin örnekleme verimliliğini artırmak için uygulanabilirliğini araştırmayı amaçlamaktadır. Karmaşık, yüksek boyutlu durumları daha verimli bir şekilde temsil etme potansiyelinden yararlanarak, RL ajanlarının kontrol politikalarını daha hızlı, daha az örnek ve gözlem gürültüsüne karşı dayanıklı bir şekilde öğrenme yeteneklerini artırmayı ummaktadır. Ayrıca otokodlayıcıyı eğitirken sistemle etkileşime geçmeye gerek yoktur. Sadece sistem durumlarının ne olduğun ve sınırlarını bilmek yeterlidir. Ayrıca, yalnızca gizli uzay kullanmanın sınırlamalarını aşmak da bu çalışmanın kapsamı dahilindedir.

2. Temel Bilgiler

Pekiştirmeli öğrenmede, ajan ortamla etkileşimde bulunarak ve ödül biçiminde geri bildirimler alarak optimal aksiyonları almayı öğrenir. Ajan ortamda kararlar alıp eylemde bulunan varlıktır. Pekiştirmeli öğrenmede ajanın amacı, her eylemden sonra elde edilen ödülün birikimli toplamını zamanla maksimize eden davranışı sergilemektir. Ajanın faaliyet gösterdiği alana ortam denir. Ortam, prosesle ilgili dinamikleri içeren sistemdir. Ajanın faaliyetlerine yanıt olarak, ortam ajana durumlar, gözlemler ve ödüller sunar.

Ortamın durum bilgisi belirli bir zaman diliminde ortamın durumunu veya yapılandırmasını tam olarak tarif eder ve $\boldsymbol{s}$ harfi ile temsil edilir. Ajanın ortamı değiştirme seçeneği bir eylem (aksiyon) olarak adlandırılır ve $\boldsymbol{a}$ ile temsil edilir. Bir politika, durumları eylemlere eşleyerek ajanın davranışını tanımlar ve $\boldsymbol{\pi}$ ile temsil edilir.

Eğer gelecekteki durumların olasılık dağılımı sadece mevcut duruma bağlıysa bu prosese Markov özelliğine sahiptir denir. Pekiştirmeli öğrenmede ajan-ortam etkileşimi için olan matematiksel model Markov özelliğine sahip ise bu süreç Markov Karar Süreci olarak adlandırılır. Bu çalışmadaki simülasyon ortamı olan OpenAI Gym Lunar Lander ortamı Markov Karar Süreci olarak kabul edilmiştir.

Ajanın ana hedefi Denklem 1'de verilen birikimli ödülü maksimize etmektir.

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}, \quad (1)$$

Denklem 1'de G_t indirimli birikimli ödül dönüşü, R_t ise t anındaki anlık ödül dönüşüdür. γ ise gelecekteki ödüllerin ağırlığını belirleyen indirim faktörüdür. Matematiksel olarak deterministik bir politika Denklem 2'deki gibi temsil edilir.

$$\pi(s) = a, \quad (2)$$

Ajanın hedefi, her durum için beklenen indirimli birikimli ödülü maksimize eden optimal politikayı $\pi' *$ bulmaktır. Optimal politikanın tanımı Denklerm 3'de verilmiştir.

$$\pi^* = \arg\max_{\pi} \mathbb{E}[G_t | s_t = s, \pi], \quad (3)$$

Pekiştirmeli öğrenmede değer (V) ve eylem-değer (Q) fonksiyonları, ajanın beklenen gelecek ödülleri tahmin etmesine ve kararlar vermesine yardımcı olur. Bu fonksiyonlar, her durum ve durum-eylem çifti için beklenen ödül dönüşünü temsil eder ve ajanın mevcut politikasının kalitesini değerlendirmek için kullanılır. Bu fonksiyonlar Denklem 4-5'de verilmiştir.

$$V(s) = \mathbb{E}[G_t | s_t = s, \pi] \quad (4)$$

$$Q(s,a) = \mathbb{E}[G_t | s_t = s, a_t = a, \pi] \quad (5)$$

Bellman denklemleri, bir durumun veya durum-eylem çiftinin değer ve eylem-değer fonksiyonları ile onların sonraki adımdaki değerleri arasındaki matematiksel ilişkiyi sağlar. Bu denklemler, değer ve eylem-değer fonksiyonları tekrar tekrar değerlendirmek ve güncellemek için kullanılır. Bellman denklemleri Denklem 6-7'de verilmiştir.

$$V(s) = \sum_a \pi(a|s) \left(R(s,a) + \gamma \sum_{s'} P(s'|s,a) V(s') \right) \quad (6)$$

$$Q(s,a) = R(s,a) + \gamma \sum_{s'} P(s'|s,a) \sum_{a'} \pi(a'|s') Q(s',a') \quad (7)$$

$V(s)$ ve $Q(s,a)$ için Bellman optimallik denklemleri Denklem 8-9'de verilmiştir.

$$V^*(s) = \max_a \left(R(s,a) + \gamma \sum_{s'} P(s'|s,a) V^*(s') \right) \quad (8)$$

$$Q^*(s,a) = R(s,a) + \gamma \sum_{s'} P(s'|s,a) \max_{a'} Q^*(s',a') \quad (9)$$

Zaman farkı (temporal difference - TD) öğrenme olarak bilinen modelden bağımsız pekiştirmeli öğrenme algoritmaları, dinamik programlama prensiplerini Monte Carlo yaklaşımları ile birleştirir. TD öğrenme algoritmaları, değer fonksiyonu tahminlerini güncellerken bir sonraki durumun değerini ve gözlemlenen ödülü kullanır. Bu, TD öğrenme algoritmalarının tamamlanmamış simülasyonlardan öğrenmesini ve değer tahminlerini çevrimiçi olarak güncellenmesini sağlar. Buna karşılık Monte Carlo yöntemleri bir simülasyonun tam bitmesini gerektirir. TD hatasının hesaplanması Denklem 10'de verilmiştir [9].

$$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \quad (10)$$

$V(s)$, TD hatası ile güncellenir. Bu güncelleme Denklem 11'de verilmiştir.

$$V(S_t) \leftarrow V(S_t) + \alpha \delta_t \quad (11)$$

Q-öğrenme, bir TD kontrol yöntemidir. Burada eylem-değer fonksiyonu $Q(s,a)$ iteratif bir şekilde TD hatası ile güncellenir. Bu durum, yeni eylem-değer tahminlerine yanıt olarak politikayı optimize eder. $Q(s,a)$ için TD hatası Denklem 12'de verildiği gibi hesaplanır [9].

$$\delta_t = R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t) \quad (12)$$

$Q(s,a)$ değeri TD hatası kullanılarak Denklem 13'de belirtildiği gibi güncellenir.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \delta_t \quad (13)$$

Derin Q-Ağı algoritmasında, eylem-değer fonksiyonu $Q(s,a)$ derin sinir ağı yapısıyla tahmin edilir [5]. Bu algoritmada derin öğrenme ve pekiştirmeli öğrenmenin dikkatsizce

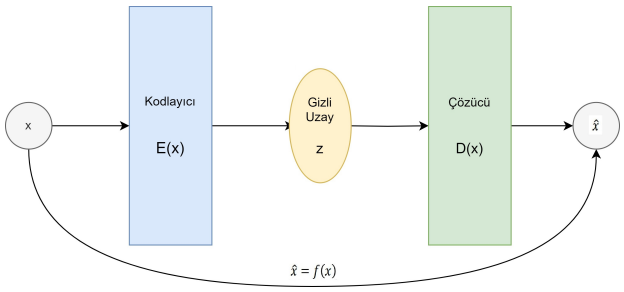
birleştirilmesinden kaynaklanan kararsızlık sorunlarını azaltmak için deneyim tekrar belleği (experience replay buffer) ve hedef ağları gibi yaklaşımlar sunar. Temel bir Derin Q-Ağı algoritması için eğitim algoritması Şekil 1'de verilmiştir.

Algorithm 1: Temel Derin Q-Öğrenme Eğitimi

Tekrar Belleği Oluşrur: R
Eylem-Değer Ağını (Q) θ rastgele ağırlıkları ile oluştur
Hedef Eylem-Değer Ağını ($\hat{Q}$) θ^{-} rastgele ağırlıkları ile oluştur
Keşfetme faktörü ϵ oluştur
for *simulasyon adımı = 1, N* **do**
  Ortamı (Simulasyonu) Resetle
  while *not done* **do**
    ϵ olasılığı ile rastgele aksiyon a_t seç
    aksi halde $a_t = \arg\max_a Q(s_t, a; \theta)$ aksiyonunu seç
    a_t aksiyonunu ortam üzerinde uygula ve sonraki durum s_{t+1} ile ödül r_t elde et
    Bu geçişleri (s_t, a_t, r_t, s_{t+1}) R'de sakla
    if *R'de yeterli örnek bulunuyorsa* **then**
      R'den Rastgele bir grup örnekle (s_k, a_k, r_k, s_{k+1})
      Eğer simulasyon k+1 adımında bitiyorsa $y_k = r_k$
      aksi halde $y_k = r_k + \gamma \max_a' \hat{Q}(s_{k+1}, a'; \theta^{-})$
      $(y_k - Q(s_k; \theta))^2$ üzerinde Gradyan azalma gerçekleştir.
      $\theta^{-} = \tau * \theta + (1 - \tau) * \theta^{-}$
    end
    $s_t = s_{t+1}$
    $\epsilon = \epsilon * \delta\epsilon$
  end
end

Şekil 1: Temel Derin Q-Ağı Algoritması

Otokodlayıcılar, yapay sinir ağları kullanarak etkin kodlamaların denetimsiz öğrenilmesi için kullanılır [4]. Bir otokodlayıcı, bir giriş verisinin bir temsilini (kodlamasını) öğrenir. Otokodlayıcılar, giriş verisinden anlamlı özellikler çıkarabilir ve böylece daha iyi temsiller elde edebilir. Kodlayıcı kısmı girişi gizli bir uzaya kodlar ve çözücü kısmı bu girişi orjinal boyutlarına yeniden inşa eder. Otokodlayıcının bir şeması Şekil 2'de verilmiştir.



Şekil 2: Otokodlayıcı Şeması

Öğrenme süreci, yeniden yapılandırma hatası olarak bilinen bir kayıp fonksiyonunu en aza indirgemeyi içerir. Yeniden yapılandırma, girişin kodlandıktan sonra çözülerek orjinal girişi elde etmektir. Otokodlayıcılar için en popüler kayıp fonksiyonu ortalama karesel hata olup bu hata orjinal giriş ile yeniden yapılandırılan çıkış arasındaki mesafeyi değerlendirir. Bu hata fonksiyonu Denklem 14'de verilmiştir. Burada $\hat{x}$, giriş x'in yeniden yapılandırılmış versiyonudur. Giriş verilerindeki gürültüye karşı dayanıklılık sağlamak için gürültü giderici (denoising) otokodlayıcı türü kullanılabilir [10]. Bu türde yeniden yapılandırma kaybı gürültü eklenmiş girişin yeniden yapılandırılmış hali ile girişin gürültüsüz versiyonu arasında hesaplanır.

$$L(x, \hat{x}) = \|x - \hat{x}\|^2 \tag{14}$$

Ayrıca modelin giriş değerlerindeki küçük değişikliklere dayanıklı olmasını sağlamak için daraltıcı (contractive) otokodlayıcı türü kullanılabilir [7]. Bunu için Denklem 14'deki kayıp fonksiyonuna düzenleme terimi Denklem 15'deki gibi eklenebilir. Burada kodlayıcı gizli uzay katmanının girişe göre olan Jacobian'ının Frobenius normu hesaplanır. Burada λ bir hiperparametre, h kodlayıcının gizli uzay katmanı ve x giriştir.

$$L(x, \hat{x}) = \|x - \hat{x}\|^2 + \lambda \|\nabla_x h(x)\|_F^2 \tag{15}$$

3. Metodoloji

Bu çalışmada simülasyon ortamı olarak OpenAI Gym tarafından geliştirilen Lunar Lander isimli simülasyon ortamı kullanılmıştır [11]. Bu ortam klasik bir roket yörünge optimizasyon problemi olan, Ay'ın yüzeyine güvenli iniş yapmayı amaçlayan bir Ay'a iniş modülünü simüle etmektedir. Bu modül her simülasyon başladığında havadan başlar ve $(0, 0)$ X, Y koordinatlarına inmeye çalışır.

Lunar Lander ortamı 8 boyutlu sürekli bir durum uzayına sahiptir. Bu durumlar modülün X, Y koordinatlarını ve hızlarını, açısını ve açısal hızını ve iki bacağın yere değip değmediğini gösteren değişkenleri içermektedir. Bu durumların özeti Tablo 1'de verilmiştir.

Tablo 1: Lunar Lander Durum Uzayı

Durumlar	Sınırlar
x,y koordinatları	[-90, 90] m
x,y hızlar	[-5, 5] m/s
açı	[-3.14, 3.14] rad
açısal hız	[-5, 5] rad/s
2 Adet Bacak Temas Değişkeni	[0, 1]

Sistemin aksiyon uzayı 4 boyutlu ayrık bir uzaydır. Bu uzay hiç aksiyon almama, alttaki ana motoru çalıştırma, sağdaki veya soldaki motorları çalıştırmayı içermektedir. Her aksiyon modülün durumlarını dolayısıyla alınan ödülü etkilemektedir. Bu aksiyonların özeti Tablo 2'de verilmiştir.

Tablo 2: Lunar Lander Aksiyon Uzayı

Aksiyonlar	Aksiyon Numarası
Aksiyon Yok	0
Sol Motor Ateşlemesi	1
Ana Motor Ateşlemesi	2
Sağ Motor Ateşlemesi	3

Ortamın ödülleri, güvenli ve verimli inişleri teşvik etmek için tasarlanmıştır. Başarılı bir iniş için ödül +100, çarpışma durumunda ise ceza -100 olarak belirlenmiştir. Ayrıca, motorları ateşleme veya iniş platformundan uzaklaşma gibi eylemler

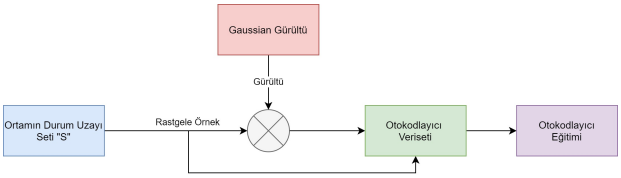
için ara ödüller ve cezalar da bulunmaktadır. Lunar Lander ortamının ödüllerine ilişkin özet bilgiler Tablo 3'de verilmiştir.

Tablo 3: Lunar Lander Ödül Uzayı

Vaka	Her Adım için Ödül
Çarpma	-100
Güvenli İniş	+100
Bacak Kontağı	+10
Yan Motor Ateşlemesi	-0.03
Ana Motor Ateşlemesi	-0.3
x,y koordinatları	$-100\sqrt{x^2+y^2}$
x,y hızları	$-100\sqrt{\dot{x}^2+\dot{y}^2}$
açı	$-100\|\|angle\|\|$

Bu çalışmada, Lunar Lander ortamına bir Gym fonksiyonu olan ObservationWrapper uygulanarak bir dereceye kadar rastlantılılık tanıtılmıştır. Bu fonksiyon, gözlemlere Gaussian gürültü ekleyerek, gerçek bir senaryoda karşılaşılabilecek türden sensör gürültüsünü taklit etmektedir.

Bu çalışmadaki otokodlayıcı, çevreden gelen gözlemleri işlemek için kullanılır ve takviyeli öğrenme modelinin veriyi anlamasına yardımcı olur. Otokodlayıcıyı eğitmek için çevreyle etkileşime geçmeye gerek yoktur. Yalnızca çevrenin durumlarını ve kısıtlamalarını bilmek yeterlidir. Bu durumlar arasından rastgele örnekler alarak otokodlayıcıyı eğitmek için bir veri seti oluşturulabilir. Otokodlayıcıyı eğitmek için veri seti oluşturma yapısı Şekil 3'de verilmiştir.



Şekil 3: Otokodlayıcı Eğitimi için Veri Seti Oluşturma

Bu çalışmada kullanılan otokodlayıcı tamamen bağlı katmanlardan oluşan ileri beslemeli bir sinir ağıdır. Otokodlayıcının kodlayıcı kısmı 3 katmandan oluşur. İlk ve ikinci katman giriş verisini işleyerek ReLU aktivasyon fonksiyonundan geçirir. Son katman ise gizli uzayın boyutunu belirleyen katmandır ve ikinci katmandan geçmiş olan veriyi işler. Otokodlayıcının çözücü kısmı ise kodlayıcının tam tersi işlemi yapar ve gizli uzayı orjinal boyuttaki uzaya dönüştürür.

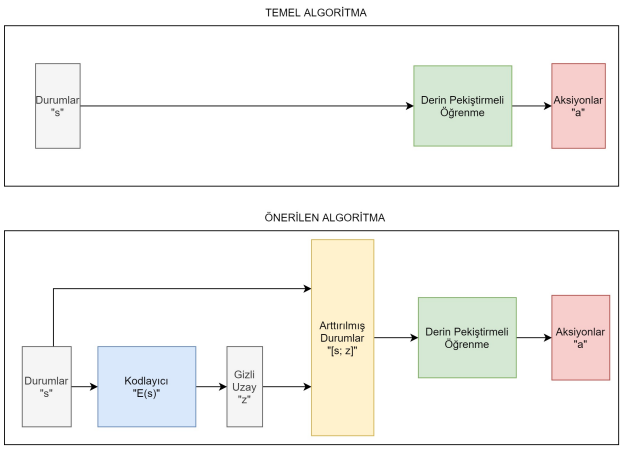
Bu çalışmada kullanılan otokodlayıcı gürültü giderici ve daraltıcı otokodlayıcıdır [2]. Otokodlayıcı, gürültü ile bozulmuş gözlemler ve girişteki küçük değişimlere karşı dayanıklılık sağlamaya yardım eden daraltıcı bir fonksiyonla birlikte eğitilir. Bu otokodlayıcının, orijinal veriye yakından eşleşen bir yeniden yapılandırma üretmesini teşvik eden bir yeniden yapılandırma kaybı vardır. Ayrıca, daraltıcı kayıp, otokodlayıcının giriş verisindeki küçük değişikliklere karşı hassasiyetini azaltan güvenilir ve tutarlı bir temsil geliştirmesini sağlar. Otokodlayıcının eğitim algoritması Şekil 4'de verilmiştir.

Algorithm 2: Gürültü Giderici-Daraltıcı Otokodlayıcı Eğitimi

Öğrenme veriseti T'yi rastgele örneklenmiş ve gürültü eklenmiş durumlarla oluştur
Validasyon veriseti V'yi rastgele örneklenmiş ve gürültü eklenmiş durumlarla oluştur
Kodlayıcı E'yi rastgele ağırlıklarla Θ oluştur
Çözücü D'yi rastgele ağırlıklarla Φ oluştur
for $t = 1, N$ **do**
  Bir grup gürültüsüz x_t and gürültülü $\tilde{x}_t$ durumlarını T'den rastgele örnekle
  Daraltıcı kaybı $\lambda\|\nabla_{x_t}\mathbf{h}(x_t)\|_F^2$ hesapla, burada $\mathbf{h}$ E'nin gizli katmanıdır
  Yeniden yapılandırılmış örneği gürültülü durumlardan elde et $\hat{x}_t = D(E(\tilde{x}_t, \Theta), \Phi)$
  Yeniden yapılandırma kaybını gürültüsüz durumları kullanarak hesapla $L(x_t, \hat{x}_t)$
  Toplam kayıp üzerinde gradyan azalma uygula $L(x_t, \hat{x}_t) + \lambda\|\nabla_{x_t}\mathbf{h}(x_t)\|_F^2$ Benzer işlemleri validasyon seti V'yi kullanarak yap **if** *minimum validasyon kaybı 1000 adım boyunca değişmediyse* **then**
    Erken dur
  end
end

Şekil 4: Otokodlayıcı Eğitim Algoritması

Bu çalışmada kullanılan RL modeli, bir Derin Q-Ağı (DQN) ajanıdır. Metodoloji ayrıca deneyimleri depolamak ve örneklemek için bir yeniden oynatma belleği kullanımını içerir. Bu, modelin geçmiş deneyimlerden öğrenmesine olanak sağlar ve zamanla performansını geliştirir. Deneyimler bellekte depolanır ve ihtiyaç duyulduğunda rastgele örneklemelerle toplu olarak seçilir. Önerilen ve temel pekiştirmeli öğrenme kontrol algoritmalarının yapısal karşılaştırması Şekil 5'de verilmiştir. Temel pekiştirmeli öğrenme kontrol algoritmaları doğrudan durumları kullanırken, önerilen algoritmalar bu durumları kodlama yoluyla elde edilen gizli uzaylarla tamamlar. Temel DQN algoritmasının eğitim algoritması Şekil 1'de (Algoritma 1) verilmiştir. Önerilen teknik için eğitim algoritması Şekil 6'de (Algoritma 3) verilmiştir.



Şekil 5: Temel ve Geliştirilmiş Tekniğin Karşılaştırılması

Algorithm 3: Kodlayıcı ile Geliştilimiş Derin Q-Öğrenme Eğitimi

Kodlayıcı E'yi ön eğitimden geçir
Tekrar Belleği Oluşrur: R
Eylem-Değer Ağını (Q) θ rastgele ağırlıkları ile oluştur
Hedef Eylem-Değer Ağını ($\hat{Q}$) θ^- rastgele ağırlıkları ile oluştur
Keşfetme faktörü ϵ oluştur
for *simulasyon adımı* $= 1, N$ **do**
  Ortamı (Simulasyonu) Resetle
  while *not done* **do**
    ϵ olasılığı ile rastgele aksiyon a_t seç
    Gizli durumları elde et $z_t = E(s_t)$
    aksi halde $a_t = \arg\max_a Q(s_t, z_t, a; \theta)$ aksiyonunu seç
    a_t aksiyonunu ortam üzerinde uygula ve sonraki durum s_{t+1} ile ödül r_t elde et
    Bu geçişleri (s_t, a_t, r_t, s_{t+1}) R'de sakla
    if *R'de yeterli örnek bulunuyorsa* **then**
      R'den Rastgele bir grup örnekle (s_k, a_k, r_k, s_{k+1})
      Gizli uzayı elde et $z_k = E(s_k)$ ve $z_{k+1} = E(s_{k+1})$
      Eğer simulasyon k+1 adımında bitiyorsa $y_k = r_k$
      aksi halde $y_k = r_k + \gamma \max_a' \hat{Q}(s_{k+1}, z_{k+1}, a'; \theta^-)$
      $(y_k - Q(s_k, z_k; \theta))^2$ üzerinde gradyan azalma uygula
      $\theta^- = \tau * \theta + (1 - \tau) * \theta^-$
    end
    $s_t = s_{t+1}$
    $\epsilon = \epsilon * \delta\epsilon$
  end
end

Şekil 6: Geliştirilmiş Tekniğin Eğitim Algoritması

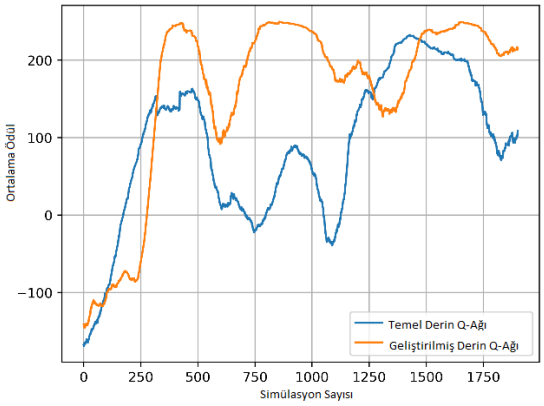
4. Bulgular ve Tartışma

Bu çalışmada gürültü önleyici ve daraltıcı otokodlayıcı Derin Q-Ağı algoritması ile birleştirilip Lunar Lander simülasyonu üzerindeki örnekleme verimliliği potansiyeli incelenmiştir.

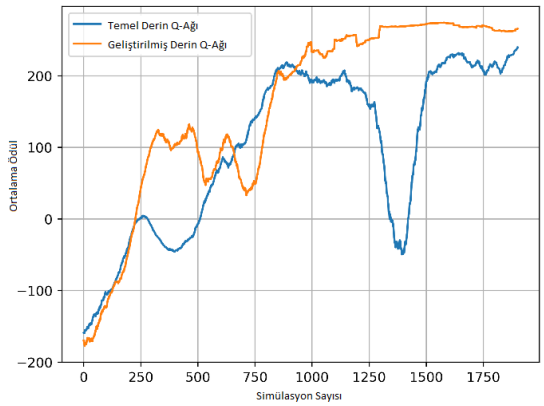
Otokodlayıcı, hem kodlayıcı hem de çözücü için üç katmana sahip olarak tasarlanmıştır. İlk ve ikinci katmanların boyutu 64 olarak ayarlanırken, gizli uzayın katman boyutu karşılaştırma yapmak için farklı boyutlarda (2, 4 ve 8) belirlenmiştir. Otokodlayıcı, özellikle girişteki küçük değişikliklere ve gürültüye karşı dayanıklı olan bir gürültü önleyici-daraltıcı otokodlayıcı olarak tasarlanmıştır. Daraltıcı kaybındaki λ değeri 10^{-4} olarak ayarlanırken, otokodlayıcının öğrenme oranı 10^{-3} olarak belirlenmiş ve aktivasyon fonksiyonu olarak ReLU kullanılmıştır. Otokodlayıcıyı eğitmek için küme boyutu 64 olarak ayarlanmıştır. Gürültü önleme işlemi için gözlemlere sıfır ortalamalı ve 0.01 standart sapmalı bir Gauss dağılımına sahip gürültü eklenmiştir.

Pekiştirmeli öğrenme sürecinde birikimli ödüllerin ağırlığını kontrol etmek için Derin Q-Ağı'nın indirim faktörü 0.99 olarak ayarlandı. Hedef Q değerinin hesaplandığı sinir ağının ağırlıklarıyla tahmin edilen Q değerinin sinir ağının ağırlıkları arasındaki güncelleme hızını kontrol eden τ değeri 10^{-3} olarak belirlendi. Derin Q-Ağı'nın eğitimi için yeniden oynatma belleğinin küme boyutu 64 olarak ayarlandı ve öğrenme oranı $5 * 10^{-4}$ olarak ayarlandı.

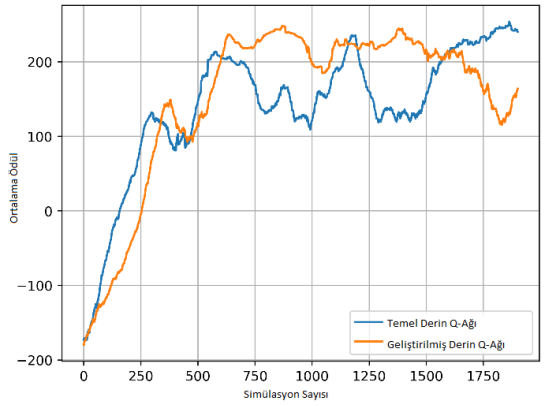
Otokodlayıcının farklı gizli uzay boyuları ve rastgele sayıların üretimi için farklı seed değerleri için bir dizi simülasyon gerçekleştirilmiştir. Her simülasyonda elde edilen, simülasyonun başarısını gösteren nümerik bir değer olan ortalama birikimli ödül değerleri temel Derin Q-Ağı ve bu çalışmada sunulan geliştirilmiş Derin Q-Ağı modeli için performans kriteri olarak değerlendirilmiştir. Bu değerlerin iki model için karşılaştırılması Şekil 7- 9'de verilmiştir.



Şekil 7: 2 Boyutlu Gizli Uzay için Ortalama Birikimli Ödüller



Şekil 8: 4 Boyutlu Gizli Uzay için Ortalama Birikimli Ödüller



Şekil 9: 8 Boyutlu Gizli Uzay için Ortalama Birikimli Ödüller

Şekil 7- 9'deki sonuçlar incelendiğinde simülasyon başarısı olarak sunulan modelin temel modele birçok durumda üstün geldiği görülmektedir. Maksimum toplam ödüle ulaşma hızı ve simülasyonlar boyunca ödülün stabilitesi açısından geliştirilmiş Derin Q-Ağı'nın, temel Derin Q-Ağı'dan daha yüksek bir performansa sahiptir. Bu sonuçlar otokodlayıcıların gözlemlere eklenen gürültülere karşı dayanıklı olduğunu ve gözlemlerden sistemle ilgili önemli içeriklerin çıkarılıp kullanılmasında başarılı olduğunu ve bu başarının sistemin kontrolünü iyileştirdiğini göstermektedir. Ayrıca sonuçlardan farklı gizli uzay boyutları için farklı performanslar elde edildiği görülmüştür. Yani gizli uzay boyutu bir tasarım parametresi olarak görülebilir.

5. Sonuç

Bu çalışmada otokodlayıcıların Derin Pekiştirmeli Öğrenme yöntemlerinin eğitimi sırasında örnekleme verimliliğini arttırmadaki kabiliyetlerinin değerlendirilmesi amçlanmıştır. Literatürde sadece gizli uzayın Derin Pekiştirmeli Öğrenme'de kullanıldığı yöntemler bulunmaktadır. Ancak bu yöntemler genellikle resimler gibi yüksek boyuttaki girişler için kullanılmaktadır ver kodlamadaki bilgi kaybı, otokodlayıcının kalitesine bağımlılık ve sistemin değişmesi durumunda gizli uzayın gözlemleri geçersiz temsil etmesi gibi problemler getirmektedir. Bu çalışmada Derin Pekiştirmeli Öğrenme'de kullanılan gözlem vektörü, bu gözlem vektörünün otokodlayıcı ile kodlanmış hali ile elde edilen gizli uzayı ile birleştirilerek pekiştirmeli öğrenme yönteminde kullanılmıştır. Böylece sadece gizli uzayın kullanıldığı yöntemlerdeki problemler aşılmış ve pekiştirmeli öğrenmenin eğitiminde örnekleme verimliliği geliştirilmiştir. Bunun için gürültü önleyici ve daraltıcı bir otokodlayıcı tasarlanmıştır ve bu otokodlayıcı temel Derin Q-Ağı'na entegre edilerek Lunar Lander simülasyonunda temel Derin Q-Ağı ile karşılaştırılmıştır. Sonuçlar geliştirilmiş algoritmanın temel algoritmaya göre yüksek ödül değerlerine daha düşük örnek sayılarında ulaşmadaki başarısını göstermiştir.

6. Kaynakça

[1] Botteghi, Nicolò, Mannes Poel, and Christoph Brune. "Unsupervised representation learning in deep reinforcement learning: A review." arXiv preprint arXiv:2208.14226 (2022).

[2] Chen, Fu-qiang, et al. "Contractive de-noising auto-encoder." Intelligent Computing Theory: 10th International Conference, ICIC 2014, Taiyuan, China, August 3-6, 2014. Proceedings 10. Springer International Publishing, 2014.

[3] Dulac-Arnold, Gabriel, et al. "Challenges of real-world reinforcement learning: definitions, benchmarks and analysis." Machine Learning 110.9 (2021): 2419-2468.

[4] Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. Deep learning. MIT press, 2016.

[5] Mnih, Volodymyr, et al. "Playing atari with deep reinforcement learning." arXiv preprint arXiv:1312.5602 (2013).

[6] Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." nature 518.7540 (2015): 529-533.

[7] Rifai, Salah, et al. "Contractive auto-encoders: Explicit invariance during feature extraction." Proceedings of the 28th international conference on international conference on machine learning. 2011.

[8] Silver, David, et al. "Mastering the game of go without human knowledge." nature 550.7676 (2017): 354-359.

[9] Sutton, Richard S., and Andrew G. Barto. "Reinforcement learning: an introduction MIT Press." Cambridge, MA 22447 (1998).

[10] Vincent, Pascal, et al. "Extracting and composing robust features with denoising autoencoders." Proceedings of the 25th international conference on Machine learning. 2008.

[11] https://www.gymlibrary.dev