KÜTAHYA
SAĞLIK BİLİMLERİ
ÜNİVERSİTESİ

ML-Agents Rapor

Yazar: Mert Koca
April 18, 2024

Özet

Bu proje, Unity'nin ML-Agents kütüphanesi kullanılarak Python ve PyTorch ile geliştirilen bir yapay zeka projesidir. Bu proje, Unity ortamında eğitilmiş bir yapay zeka modeli oluşturarak araba sürmeyi öğretmeyi amaçlamaktadır.

1 Giriş

Bu rapor, Unity'nin ML-Agents kütüphanesi kullanılarak Python ve PyTorch ile geliştirilen bir yapay zeka projesine odaklanmaktadır. Proje, yapay zeka modellerinin eğitimini sağlamak amacıyla Unity ortamında bir simülasyon üzerinde çalışmaktadır. Hedef, araç sürmeyi öğrenen ve geliştiren bir yapay zeka modeli oluşturmaktır. Bu rapor, projenin amaçlarını, gereksinimlerini, kullanılan literatürü, algoritmaları, kurulum adımlarını, bileşenleri, bölüm tasarımını, kontrolleri ve sonuçları detaylı bir şekilde sunmaktadır.

2 Gereksinimler

1. Python 3.9.13
2. Unity
3. PyTorch
4. Visual Studio
5. ML-Agents Kütüphanesi

3 Literatür

Python 3.9.13: Genel amaçlı, yüksek seviyeli, etkileşimli bir programlama dilidir. Basit ve okunabilir sözdizimine sahiptir. Python, modüler yapısı ve geniş standart kütüphanesiyle birçok farklı alanda kullanılabilir.

PyTorch: Python tabanlı ve açık kaynaklı makine öğrenmesi kütüphanesidir.

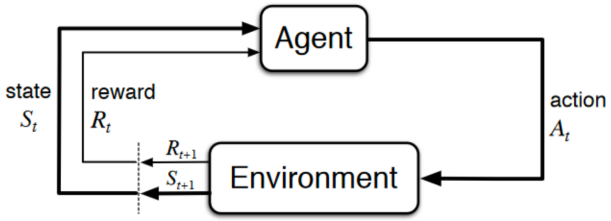
Unity: Oyun ve grafik uygulamaları geliştirmek için kullanılan bir oyun motorudur.

ML-Agents Kütüphanesi: Oyun ve simülasyonlarda ajanların eğitimi için ortamlar sağlayan açık kaynaklı bir projedir. Python API'ı kullanılarak teşvik öğrenmesi yöntemiyle ajanlar eğitilebilir. PyTorch tabanlı uygulamalar sunar. Ajanlar, NPC davranışlarını kontrol etmek ve oyun yapılarının otomatik test edilmesi gibi çeşitli amaçlar için kullanılabilir.

4 Teşvik Öğrenme Algoritması

Teşvik öğrenme algoritmasında, ajan çevresiyle etkileşime girer, belirli eylemler gerçekleştirir ve bu eylemlerin sonuçlarına göre ödüller veya ceza alır. Resimde gösterilen şema, ajanın çevresiyle olan etkileşimini ve bu süreçteki temel unsurları açıkça göstermektedir. Ajan, deneyimlediği ödül ve cezaları kullanarak en iyi eylem stratejilerini geliştirmeye çalışır.

Ibrahim Sobh Ibrahim'in yürüttüğü araştırma[1], Reinforcement Learning (RL) algoritmalarını kullanarak ajanların dinamik ortamlarda hızlı bir şekilde öğrenmesini ve adapte olmasını vurgulamaktadır. Derin RL'yi kullanarak, çalışma, özellikle müzakere gerektiren ve dinamik etkileşimler gerektiren senaryolarda otonom ajanların karar verme yeteneklerini artırmayı amaçlamaktadır.r verme yeteneklerini artırmayı amaçlamaktadır.



Figür 1. Teşvik Öğrenme Algoritması Modeli [2]

1. **Agent (Ajan):** Karar veren, öğrenen, eylemleri belirleyen ve ödülü maksimize etmeye çalışan yapay zekadır.
2. **Environment (Çevre):** Ajanın kararlarının etkileşimde bulunduğu ve ona geri bildirim sağlayan simülasyon ortamıdır.
3. **State:** Zaman (t) anındaki çevrenin durumu ya da ajanın o andaki gözlemini temsil eder.
4. **Action:** Ajanın zaman (t) anında yapmayı seçtiği eylemdir.
5. **Reward:** Ajanın (t+1) anındaki eylemi sonucunda elde ettiği ödüldür
6. **Next State:** Ajanın eylemi sonucunda çevrenin ulaştığı sonraki durumdur.

5 Kurulum Aşamaları

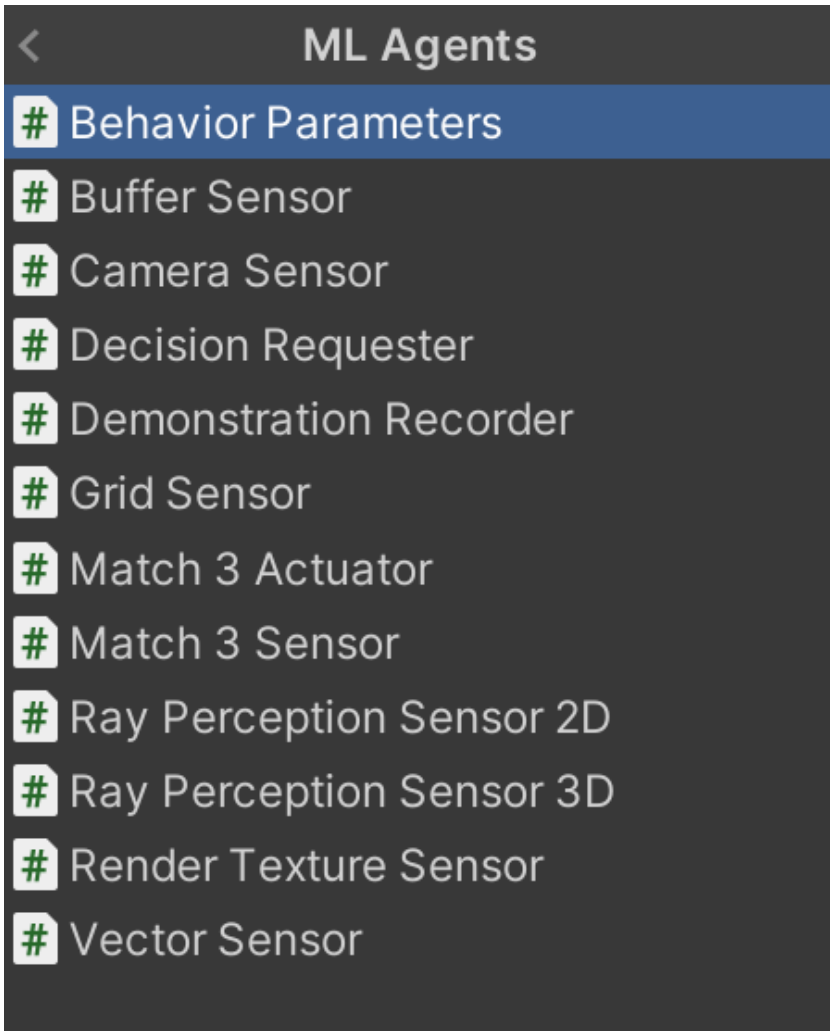
---

1. cmd yani komut istemi çalıştırılır ve cd (dizini değiştir) komutuyla projenin olduğu dosya açılır.
2. Dosyanın içinde venv adında bir virtual environment (sanal ortam) oluşturulur. Bu sayede python kullanılan başka bir proje ile karışıklık yaşanmasını önlenir.
3. Oluşturulan sanal ortam aktive edilir.
4. Python Package Installer (pip) güncel olup olmadığı kontrol edilir.
5. ML-Agents paketi kurulur.
6. Python torch kütüphanesi ve torch içindeki görüntü ve ses işleme kütüphanesi kurulur.
7. protobuf 3.20.3 sürümü kurlur. Protobuf, veri transfer protokolü olup diğer protokollere göre daha hızlıdır.
8. Packaging kütüphanesi kurulur. Bu kütüphane, python paketleri oluşturma, dağıtma ve sürümleme işlemlerini kolaylaştırır.
9. Unity projesi açılıp Package Manager kısmından mlagents paketi aktif edilir.
10. Boş bir nesne üretip bu nesnenin component kısmından ML-Agents aktif edilip edilmediği kontrol edilir.

6 ML-Agents Bileşenleri

---

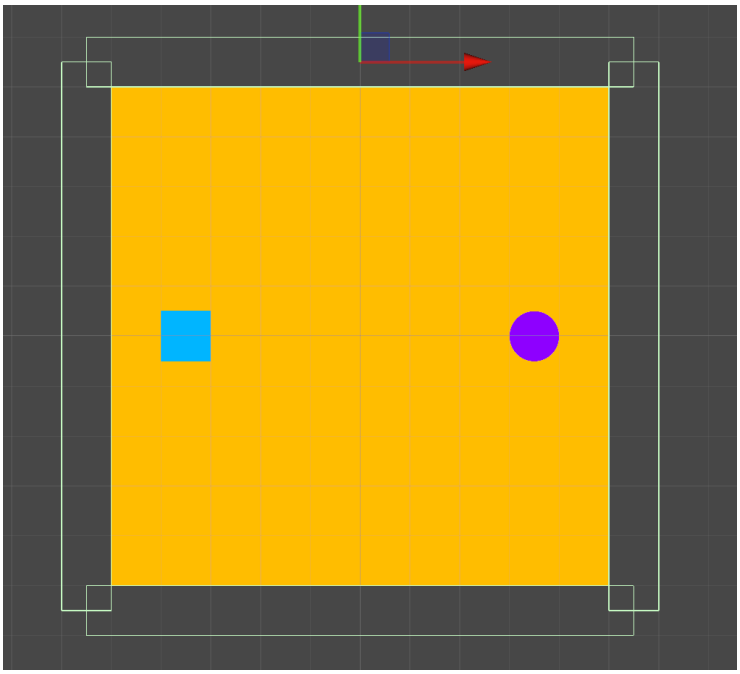
1. **Behavior Parameters:** Ajanların davranışlarını ve öğrenme süreçlerini kontrol etmek için kullanılan parametrelerin ayarlandığı bir bileşendir.
2. **Buffer Sensor:** Çeşitli veri türlerini depolamak ve işlemek için kullanılan bir sensördür.
3. **Camera Sensor:** Oyun dünyasını görsel veri olarak almak için kamera görüntülerini kullanan bir sensördür.
4. **Decision Requester:** Ajanların kararlarını gerektiğinde talep etmelerini sağlayan bir bileşendir.
5. **Demonstration Recorder:** Eğitim için insanların gerçekleştirdiği oyun hareketlerini kaydetmek ve kullanmak için bir kayıt cihazıdır.
6. **Grid Sensor:** Grid tabanlı oyunlarda çevre bilgisini algılamak için kullanılan bir sensördür.
7. **Match 3 Actuator:** Match 3 tarzı oyunlarda (Candy Crush) eylemleri gerçekleştirmek için kullanılan bir bileşendir.
8. **Match 3 Sensor:** Match 3 tarzı oyunlarda oyun durumunu algılamak için kullanılan bir sensördür.
9. **Ray Perception Sensor 2D/3D:** 2 boyutlu veya 3 boyutlu ortamlarda nesneleri algılamak için kullanılan bir sensördür.
10. **Render Texture Sensor:** Görüntüleri işlemek için render texture'ları kullanarak veri sağlayan bir sensördür.
11. **Vector Sensor:** Özel bir vektör verisiyle ajanlara çevre bilgisi sağlayan bir sensördür.



Figür 2. Teşvik Öğrenme Algoritması Modeli

7 ML-Agents Hareket Projesi

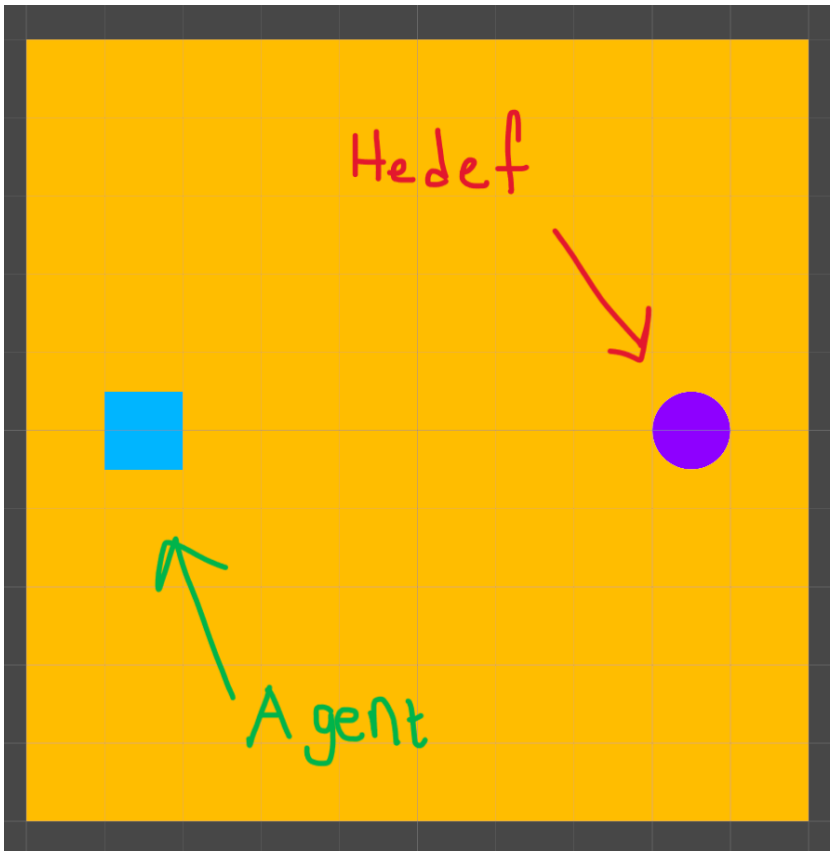
Platform üzerinde mavi agentin hedeflediği mor renkli ödüller yer alır [3]. Bunların yanında görünmeyen ancak temas edildiğinde ceza uygulayan duvarlar da bulunmaktadır. Platformun rengi, Agent'ın etkileşimlerine göre değişmektedir. Ödül alındığında platform yeşile, duvara çarpıldığında kırmızıya dönüşür. Bu tasarım, agentin belirlenen hedeflere ulaşmasına teşvik ederken, aynı zamanda negatif etkileşimlerden kaçınma yeteneğini geliştirmeyi amaçlamaktadır.



Figür 3. Hareket Projesi Bölüm Tasarımı

8 Proje Nesneleri

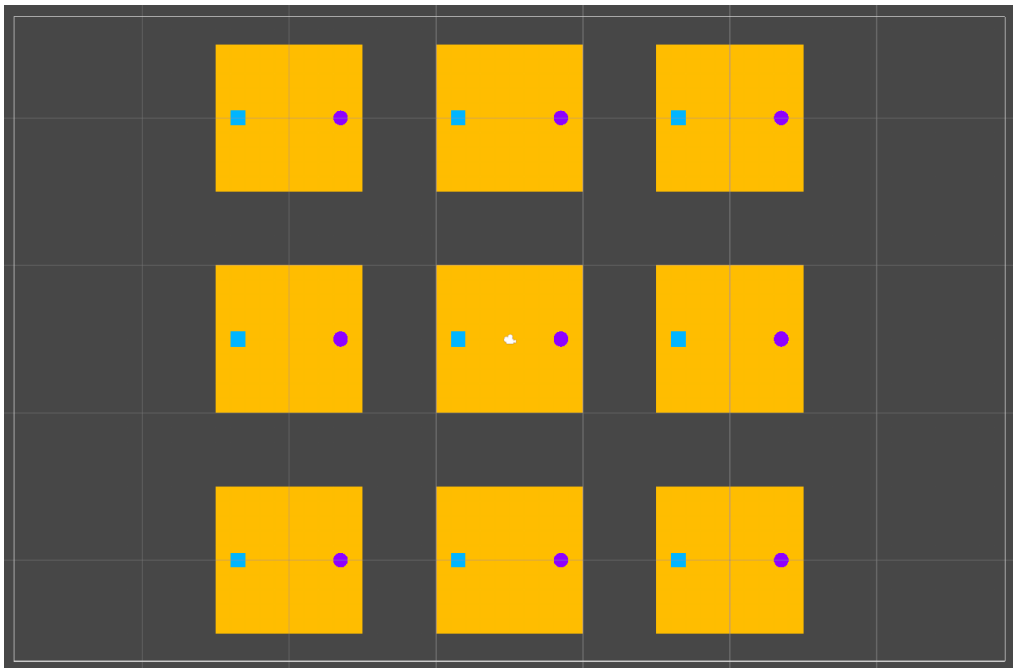
Projenin bu kısmında kullanmak üzere dört adet nesne seçilmiştir. Seçilen nesneler platform, agent, ödül ve duvarlardır. Agent hareket ederek ödüle ulaşmayı hedefler fakat bu süreçte duvarlara temas ederse en baştan başlamak zorundadır.



Figür 4. Agent ve Hedef

9 ML-Agents Davranışları

Agent hedefe ulaşmak için bir çok kez deneme yapmalıdır. Simulasyon hızlandırılmış olsada daha da hzılandırmak için oluşturulan bölüm koyalanarak toplamda aynı bölümden dokuz tane oluşturulmuştur. Böylece dokuz ajan aynı anda eğitilir.

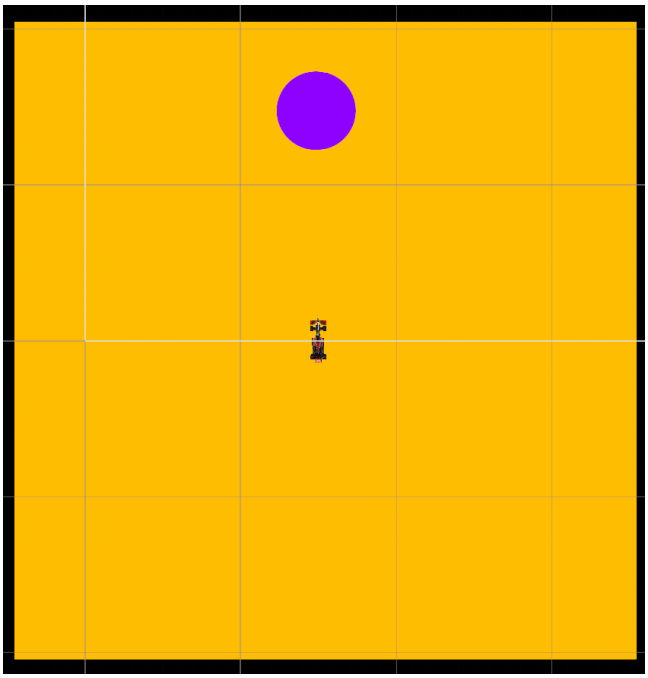


Figür 5. Çoklu Bölüm

10 Araç Kontrolleri Öğrenme Projesi

Hareket Projesine benzer olarak, platform üzerinde agentin hedeflediği mor renkli ödüller yer alır. Bu ödüllerin yeri her denemede rastgele olacak şekilde değişir. Araç duvara çarparsa ceza puanı alır ve yeniden başlar, araç hedefe ulaşırsa ödül puanı alır ve yeniden başlar. Platformun rengi, Agent'ın etkileşimlerine göre değişmektedir. Ödül alındığında platform yeşile, duvara temas edildiğinde kırmızıya dönüşür. Bu tasarım, ajanların basit araç kontrollerini öğrenmesini sağlar.

flyyufelix adlı github kullanıcısının yaptığı "donkeyrl" projesinde[4] Donkey Car adı verilen bir arabanın bir pist etrafında RL kullanarak kendi kendine dönmesini sağlamıştır. Bu projede Double Deep Q Learning (DDQN) kullanılmıştır. DDQN, bir ajanın çevresiyle etkileşime girerek belirli bir durumda alabileceği eylemler arasında en uygun olanını seçmeyi öğrenen bir güçlendirme öğrenme algoritmasıdır.



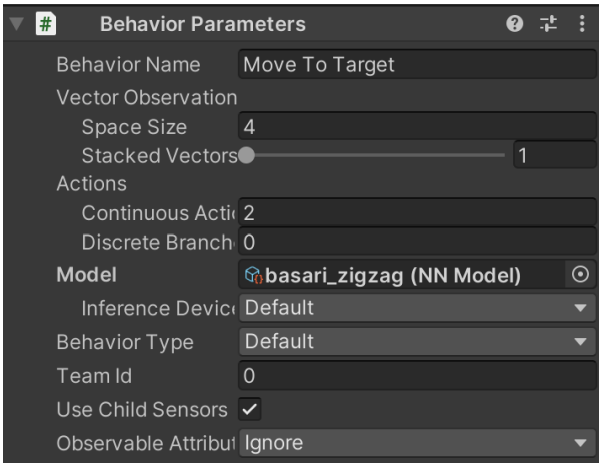
Figür 6. Araç Kontrolleri Öğrenme Projesi Bölüm Tasarımı

11 Araç ve Ajan Kontrolleri

Önceden carInput isminde başka bir koddan alınan girdiler, ajan kodunun içindeki X ekseni ve Y ekseni olarak ajan hareketlerinden alındı. Böylece ajanların araba kontrollerine erişimi sağlandı.

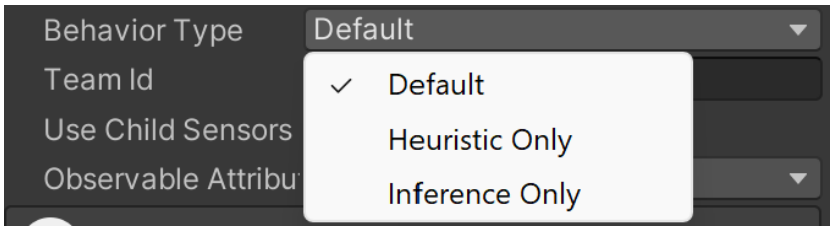
Ajanın kullancı tarafından kontrol edilip edilemediğini, ajanın kaç eksende hareket yaptığını, eğitilen beynin ajana aktarıldığı ve diğer bir çok ayarların yapıldığı kısım Behavior Parameters olarak adlandırılır. Bu özellik ML-Agents ile birlikte gelir.

Figür 7. Araba Kontrolleri



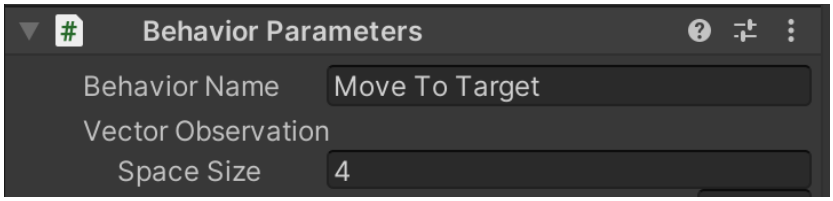
Figür 8. Behavior Parameters

Behavior Type: Agent'ın kullancı tarafından kontrol edilip edilemediğini değiştiren bir özelliktir. Agent'ı kullanıcının yönetmesi isteniyorsa heuristic only seçeneği seçilmelidir. Agent'a dışarıdan herhangi bir müdehalede bulunmadan hedefe ulaşması isteniliyorsa inference only seçeneği seçilmelidir.



Figür 9. Behavior Type

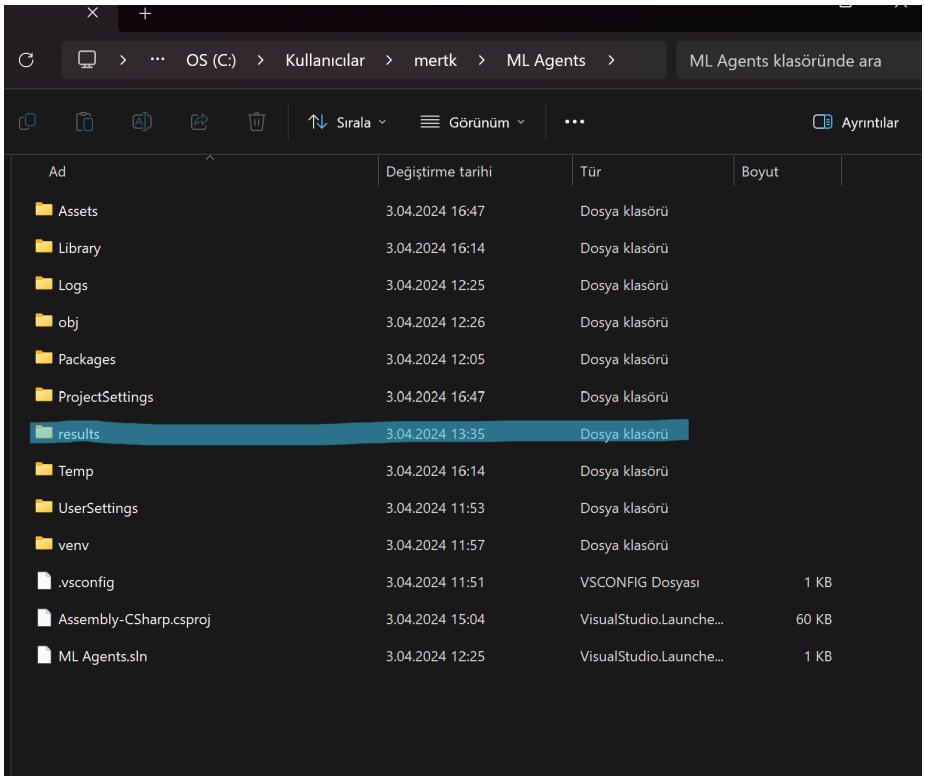
Space Size: Agent'ın sahip olduğu davranışları belirleyen ve yönlendiren bir özelliktir. Bazı durumlarda Agent'ın farklı davranışlar sergilemesi istenebilir. Bu durumda, her bir nesner için farklı davranışlar tanımlanması gerekir. Bu projede Agent ve Ödül olarak iki hareketli nesne olduğundan her iki nesnenin de x ve y eksenindeki hareketleri almak için Space Size özelliği dört olarak ayarlanır.



Figür 10. Space Size

12 Beyin Oluşturma Kaydetme ve Yükleme

1. Unity projesi çalıştırıldığından itibaren Agentlar öğrenmeye başlar. Öğrenmiş olan bu beyin, proje dosyasının içindedir



Figür 11. Proje Dosyaları

Figür 12. Beyin Dosyası

2. Result dosyasının içindeki beyin, proje dosyasının içindeki Assets klasörüne kopyalanır. Kopyalanan bu beyin Unity içerisinde Assets kısmında görülebilir.

Figür 13. Assets Klasörü

3. Agent altında, Behavior Parameters bileşeninin içindeki, Model adındaki değişkene Assets klasöründeki beyin atanır. Böylece önceden eğitilen bir Agent'ı projeye eklemiş oluruz.

Figür 14. Model Değişkeni

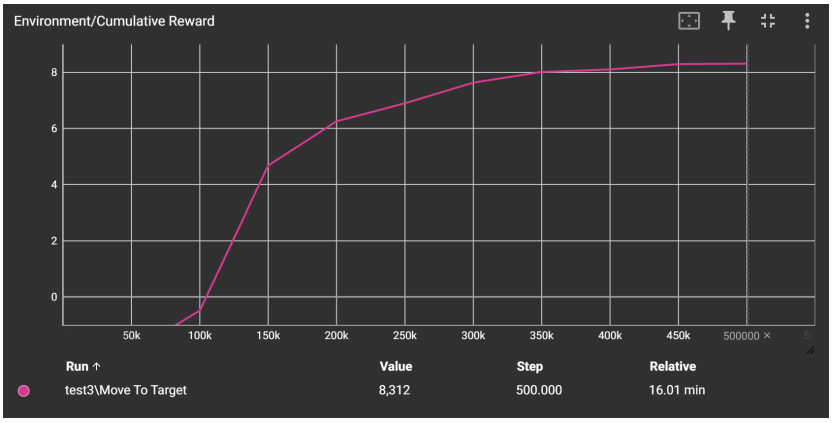
13 Başarılı ve Başarısız Oranı

Ekranın sol üst kısmında başarılı deneme sayısını, başarısız deneme sayısını, başarılı ve başarısız deneme sayısı arasındaki farkı ve başarı oranını gösteren bir panel yer almaktadır. Bu panel sayesinde deneme sayısı arttıkça başarı oranının arttığı gözlemlenebilir.

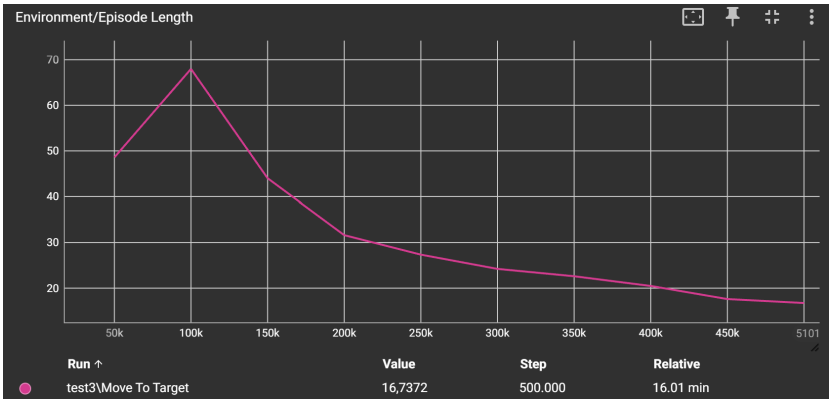
Figür 15. Az Deneme Sayısı

Figür 16. Çok Deneme Sayısı

Eğitim tamamlandıktan sonra TensorBoard yardımıyla sonuçlar grafik halinde görüntülenebilir. Aşağıdaki grafikler eğitilmiş bir beyine aittir.



Figür 17. Toplam Ödül Sayısı Grafiği



Figür 18. Bölüm Uzunluğu Grafiği

14 Sonuç

Bu çalışma, Unity'nin ML-Agents kütüphanesi kullanılarak Python ve PyTorch ile geliştirilen bir yapay zeka projesine odaklanmaktadır. Projenin amacı, araç sürmeyi öğrenen ve geliştiren bir yapay zeka modeli oluşturmaktır. Çalışmanın sonuçları, ajanların çevreleriyle etkileşime girerek araba kontrolü öğrenmesini sağladığını göstermektedir. Eğitim sonucunda elde edilen başarı oranları ve ödül grafiği incelendiğinde, ajanların belirlenen hedeflere ulaşma yeteneklerinin arttığı ve bölüm tamamlama süresinin azaldığı görülmektedir. Bu sonuçlar, ML-Agents kütüphanesi kullanarak yapay zeka modellerinin eğitilmesinin ve çeşitli görevlerin başarılı bir şekilde öğrenilmesinin mümkün olduğunu göstermektedir.

Kaynakça

[1] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, S. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 4909–4926, 2021.

[2] T. Simonini, "An introduction to unity ml-agents," *Medium*, 2020.

[3] T. A. Bot, "How to use unity ml agents 2.0.1," *Youtube*, 2023.

[4] flyyufelix, "Train donkey car in unity simulator with reinforcement learnin," *GitHub*, 2018.