TC
BARTIN ÜNİVERSİTESİ
MÜHENDİSLİK, MİMARLIK VE TASARIMFAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ

Mangala Oyunun Yapay Zeka Entegrasyonu

Lisans Bitirme Tezi

Burhan GÜL
Tez Danışmanı: Dr. Evrim GÜLER

BARTIN-2024

ONAY

Öğrenci Adı Soyadı:	**Burhan Gül**
Öğrenci numarası:	**20010310035**
Bölümü:	**Bilgisayar Mühendisliği**

Yukarda açık bilgileri verilen öğrencilerin "**Mangala Oyunu Yapay Zeka Entegrasyonu**" başlıklı çalışması lisans bitirme projesi olarak kabul edilmiştir.

Danışman Onayı
(Adı Soyadı ve İmza)

Bölüm Başkanı Onayı
(Adı Soyadı ve İmza)

Onay Tarihi:

Onay Tarihi:

TEZ VE ETİK BİLDİRİM

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada orijinal olmayan her türlü kaynağa eksiksiz atıf yapıldığını, aksinin ortaya çıkması durumunda her türlü yasal sorumluluğu üstlendiğime dair beyanda bulunuyorum.

Öğrenci Adı Soyadı: Burhan Gül

İmza

Tarih

İçindekiler

TEŞEKKÜR

Çalışmamda bana rehberlik eden ve destek ve yardımlarını esirgemeyen Dr. Öğr. Üyesi Evrim Güler'e teşekkür ederim.

ÖZET

Bu çalışmada, Deep Q-Network (DQN) algoritması kullanılarak Mancala oyunu için bir yapay zeka modeli eğitildi. Pygame kütüphanesi kullanılarak oyunun görsel arayüzü geliştirildi. DQN algoritması, modelin oyun stratejisini öğrenmesi için kullanıldı ve Pygame ile entegre edilen görüntü kısmıyla birleştirildi.

1.Mangala

Tarihi araştırmalar, Mangala Oyunu'nun Sakalar, Hunlar ve Göktürkler döneminde oynandığını göstermektedir. Dünyada "Coffee House" kültürünün temelleri 1554 yılında İstanbul'da atılmıştır; İstanbul'dan sonra günümüz Avrupa'sında kahve kültürünün izleri ise 1650 yılından itibaren, bizden 96 yıl sonra görülmeye başlanmıştır. Günümüze kadar, bu güçlü temellerin atıldığı dönemin kahve kültürünü yansıtan sadece iki adet görsel kaynak miras kalmıştır. İki kaynak da İstanbul'da resmedilmiştir. Mangala oyunu, bu iki görsel kaynakta da bulunan oyunlardan birisidir.

Resim1:18. yüzyılda Mangala oynayan iki kadın[1]

Dünyanın farklı ülkelerinde mangala türü oyunlar oynanmaktadır ancak Türk Mangalası'nı diğer mangala oyunlarından ayıran bazı özellikler vardır. Özellikle Mancala oyunu ile karşılaştırıldığında, Mangalada seçilen kuyudan alınan taşlar önce alınan kuyuya bir tane bırakılarak, sonrasında saat yönünde kuyulara bırakılmaya başlanır. Mancala'da ise bu durum, alınan taşların bir tanesi bırakılmadan saat yönünden dağıtılmaya başlanmasıyla farklılık gösterir. Başka bir farklılık olarak Mangalaya özgü olan kapma yöntemidir. Eğer oyuncunun taşlarını dağıtırken bıraktığı son taş, rakibinin kuyusuna denk gelip o kuyunun toplam taş sayısını çift yapıyorsa, oyuncu taşların tamamını kendi hazinesine atma hakkını kazanır. Bu gibi farklılıklarla Mangala, birden fazla varyasyona sahip bir oyun olarak karşımıza çıkar.

.

Resim2:Mangala tahtası[1]

Genel oynanış kuralları şu şekildedir: Mangala Türk Zeka ve Strateji Oyunu iki kişi ile oynanır. Oyun tahtası üzerinde karşılıklı 6'şar adet olmak üzere 12 küçük kuyu ve her oyuncunun taşlarını toplayacağı birer büyük hazine bulunmaktadır. Mangala Oyunu 48 taş ile oynanır. Oyuncular 48 taşı her bir kuyuya 4'er adet olmak üzere dağıtırlar. Oyunda her oyuncunun önünde bulunan yan yana 6 küçük kuyu, o oyuncunun bölgesidir. Karşısında bulunan 6 küçük kuyu ise rakibinin bölgesidir. Oyuncular, hazinelerinde en fazla taşı biriktirmeye çalışırlar. Oyun sonunda en çok taşı toplayan oyuncu oyun setini kazanmış olur. Oyuna kura ile başlanır. Oyunda 4 ana temel kural vardır.

(1. TEMEL KURAL: Kura neticesinde başlama hakkı kazanan oyuncu kendi bölgesinde bulunan istediği kuyudan 4 adet taşı alır. Bir adet taşı aldığı kuyuya bırakıp saatin tersi yönünde, yani sağa doğru her bir kuyuya birer adet taş bırakarak elindeki taşlar bitene kadar dağıtır. Elindeki son taş hazinesine denk gelirse, oyuncu tekrar oynama hakkına sahip olur. Oyuncunun kuyusunda tek taş varsa, sırası geldiğinde bu taşı sağındaki kuyuya taşıyabilir. Hamle sırası rakibine geçer. Her seferinde oyuncunun elinde kalan son taş oyunun kaderini belirler.

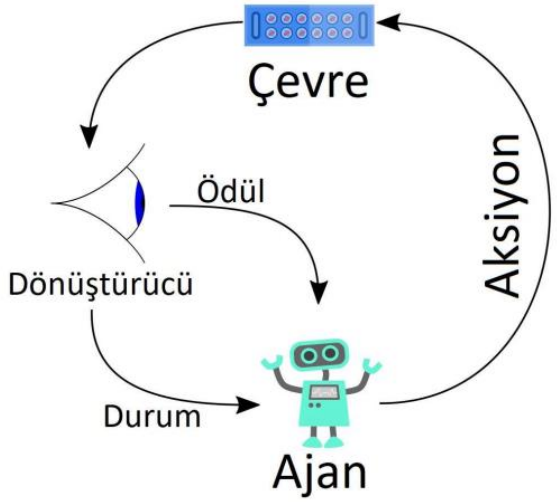
2. TEMEL KURAL: Hamle sırası gelen oyuncu kendi kuyusundan aldığı taşları dağıtırken elinde taş kaldıysa, rakibinin bölgesindeki kuyulara da taş bırakmaya devam eder. Oyuncunun elindeki son taş, rakibinin bölgesinde denk geldiği kuyudaki taşların sayısını çift sayı yaparsa (2, 4, 6, 8 gibi) oyuncu bu kuyuda yer alan tüm taşların sahibi olur ve onları kendi hazinesine koyar. Hamle sırası rakibine geçer.

3. TEMEL KURAL: Oyuncu taşları dağıtırken elinde kalan son taş, yine kendi bölgesinde yer alan boş bir kuyuya denk gelirse ve eğer boş kuyusunun karşısındaki kuyuda da rakibine ait taş varsa, hem rakibinin kuyusundaki taşları alır, hem de kendi boş kuyusuna bıraktığı taşı alıp hazinesine koyar. Hamle sırası rakibine geçer.

4. TEMEL KURAL: Oyunculardan herhangi birinin bölgesinde yer alan taşlar bittiğinde oyun seti biter. Oyunda kendi bölgesinde taşları ilk biten oyuncu, rakibinin bölgesinde bulunan tüm taşları da kazanır. Dolayısıyla, oyunun dinamiği son ana kadar hiç düşmez.)[2]

2. Takviyeli Öğrenme

Takviyeli öğrenme, bir ajanın eylemleri gerçekleştimesi sonucu oluşan ortamdaki tepkileri gözlemleyerek davranışını geliştirmeyi amaçlayan geri bildirim tabanlı bir makine öğrenme tekniğidir. Bu yöntemde, ajan, her bir başarılı eylem için olumlu geri bildirim alırken, her hatalı eylem için olumsuz geri bildirim veya ceza alır. Takviyeli öğrenme, denetimli öğrenmenin aksine, herhangi bir etiketli veri olmadan sadece geri bildirimleri kullanarak otomatik olarak öğrenme yeteneğine sahiptir. Bu durumda, ajanın öğrenmesi tamamen kendi deneyimine dayanmaktadır, çünkü etiketli veri bulunmamaktadır.



Resim3: Takviyeli Öğrenme'nin Basit Diyagram Halinde Gösterimi

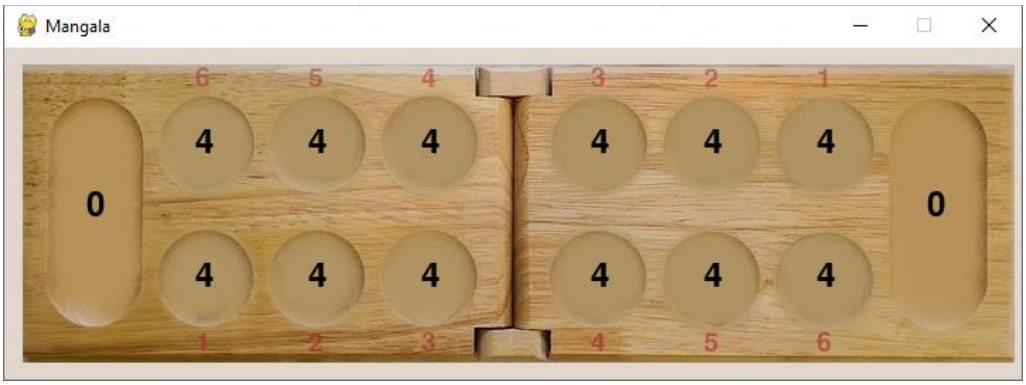
3.Kullanılan Teknolojiler

3.1 PyGame

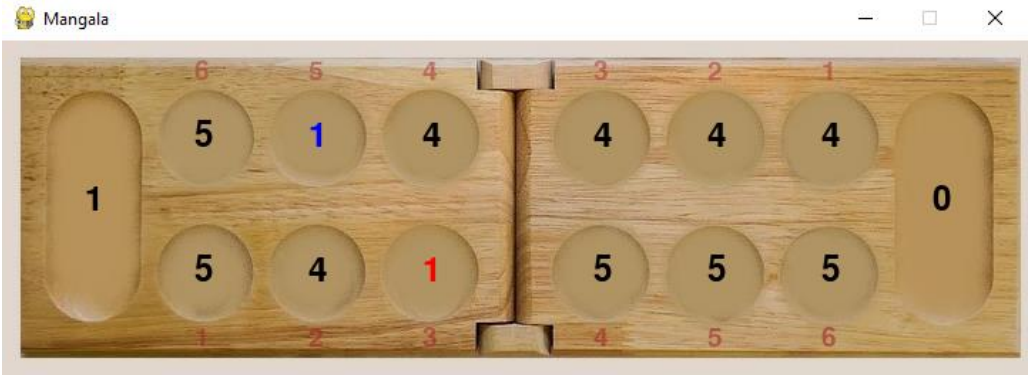
Pygame, video oyunları yazmak için tasarlanmış, platformlar arası çalışabilen bir Python modül setidir. Python programlama dili ile kullanılmak üzere hazırlanan bu modül seti, bilgisayar grafikleri ve ses kütüphanelerini içermektedir.

Proje kapsamında, mangala oyununun görsel boyutta oynanmasını sağlamak amacıyla Pygame kullanılmıştır. Oyun, iki ekranlı bir yapıya sahiptir. Ana ekran, oyunun oynandığı bölümü gösterir ve oyuncular, 1-6 numeric tuşlarını kullanarak hamle yapabilirler. Karşı tarafın geçmiş hamleleri mavi renkle, kendi geçmiş hamleleri ise kırmızı renkle etiketlenmiştir. Geriye kalan tüm kuyular ve hazineler siyah renkle gösterilmektedir. İkinci ekran olan sonuç ekranı, oyunculardan birinin kuyularının bitmesi sonucunda geçilir ve oyunu kimin kazandığını ve son hazine değerlerini içeren bilgileri sunar.

Ana Ekran:

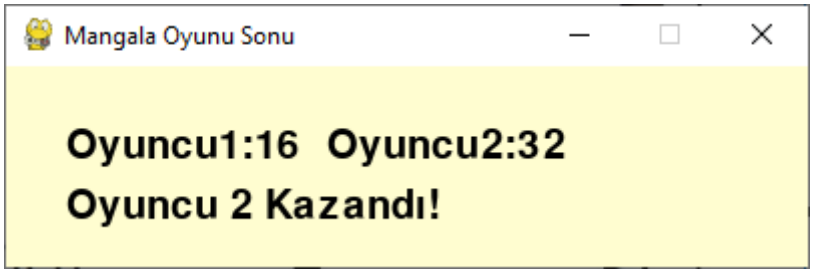


Resim4:Pygame ile oluşturulmuş başlangıç durumundaki mangala tahtası



Resim5: Oyuncu 1'in 3 numaralı hamleyi Oyuncu 2'nin 5 numaralı hamleyi oynadığı durum

Sonuç Ekranı:



Resim6: PyGame ile oluşturulmuş sonuç ekranı

3.2 OpenAI GYM

GYM, öğrenme algoritmaları ile ortamlar arasında iletişim kurmak için standart bir API ve bu API ile uyumlu standart bir ortam kümesi sağlayarak takviyeli öğrenme algoritmalarını geliştirmek ve karşılaştırmak için kullanılan açık kaynaklı bir Python kütüphanesidir.

Öncelikle, modelin oyunu nasıl algıladığına dair temel konulara hakim olmamız gerekiyor. Model seçimine göre değişiklik gösterebilir, ancak mangala oyunu üzerinden konuşmak

gerekirse, doğrudan ekran görüntüsünü kullanmak ve piksel değişikliklerinden anlam çıkartma yöntemi oldukça verimsizdir. Bu nedenle, modelin oyunu bir dizi formatında görmesini ve eylemlerini bir tamsayı sabiti üzerinden ifade etmesini tercih ediyoruz.

Anlattığımız model-mangala oyunu etkileşimini daha kolay hale getirmek için ara katman olarak GYM kütüphanesini kullanıyoruz. Oyundan alınan ve verilen verilerin uygun formatta ve karmaşık bir yapıda olmadan alışveriş yapılmasını sağlıyoruz.

Bu noktada, OpenAI GYM'in hazır bir mangala ortamına sahip olmaması nedeniyle mangala oyununu kodlamamız gerekiyor. Mangala oyununun temel kurallarını ve fonksiyonlarını kodladıktan sonra, OpenAI ile çevrelemek için bazı temel fonksiyonları override etmek ve bazı değerleri tanımlamak gerekiyor.

Tanımlanması gereken başlıca değişkenler şunlardır:

Action Space ve Observation Space

Action Space: Yani eylem uzayının tanımlanması; modelin seçim yaparken hangi aralıkta ve hangi değer biçiminde değerleri deneyebileceğini bildirdiğimiz kısımdır. Mangala oyunu için şu şekilde tanımlanır:

self.action_space = spaces.Discrete(6)

Burada, "Discrete" değerinin bir tamsayı olduğu ve maksimum değerin 6 olduğu varsayılmıştır. Ayrıca, varsayılan olarak tanımlanmış minimum değerin 0 olduğu için 0-6 arası değerlerin kabul gördüğü belirtilmiştir.

Observation Space: Daha önce bahsedilen, modelin oyunu nasıl göreceğini belirleyen kısımdır.

self.observation_space = spaces.Box(low=0, high=48, shape=(14,), dtype=int)

Bu kod, 14 uzunluğuna sahip bir dizinin her bir değerinin 0 ile 48 arasında değişebileceğini belirtir. Buradaki 14 değeri, 6 şardan (12 kuyu) ve 2 hazine sonucundan oluşan toplam 14 gözlem değeri olduğunu ifade eder. 0 ile 48 arasındaki aralık, bir dizinin elemanlarının değer aralığını tanımlar ve 48 değeri, başlangıç durumunda bulunan tüm taşların toplamına karşılık gelir.

Tanımlanması gereken başlıca fonksiyonlar şunlardır:

__init__(): Bu fonksiyon, ortamınızı varsayılan değerlerle başlatmamıza yarar.

reset(): Bu fonksiyon, ortamı varsayılan ayarlara sıfırlamak içindir.

step(): Bu fonksiyon, ajanın bir eylem gerçekleştirdiğinde ortamın nasıl değişeceğini yürütür. Genellikle, ödül fonksiyonu da step() içine dahil edilir veya çağrılır.

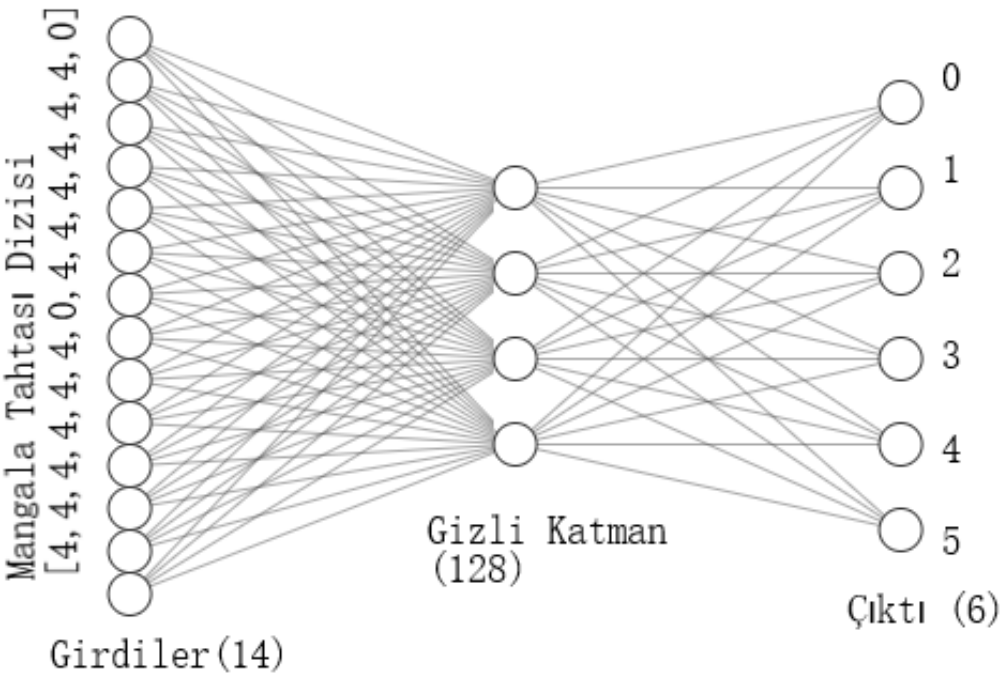
Girdi olarak eylem alır -> Eylem sonucunda yeni mangala tahtası konumu, eylem sonucunda varsa kazanılan ödül, oyunun eylem sonrasında bitip bitmediği durumunu döndürür.

render(): İsteğe bağlı olarak oyunu görüntülemek için kodların yazılabileceği kısım.

3.3 PyTorch

PyTorch, açık kaynaklı bir makine öğrenimi kütüphanesi olup, özellikle takviyeli öğrenme modelleri geliştirmek ve eğitmek için kullanılır. Temel olarak Facebook'un Yapay Zeka araştırma grubu tarafından geliştirilen PyTorch, Python yanı sıra C++ ile de entegre edilebilen bir yapıya sahiptir

Projede Pytorch DQN modelini kodlarken 14x128x6 lık bir yapay sinir ağı oluşturmak için kullanılmıştır.

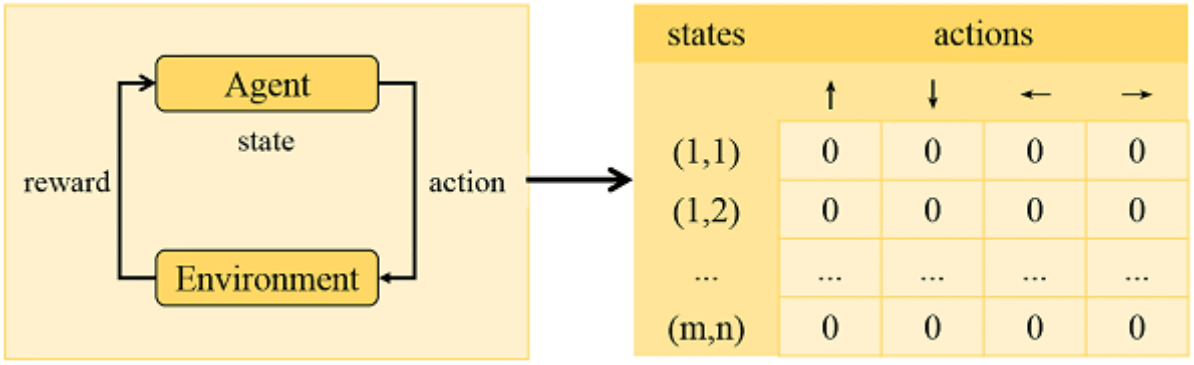


Resim7: PyTorch ile oluşturulan yapay sinir ağı modelinin 128'den 4 e gösterim kolaylığı için indirgenmiş hali

4. Deep Q-Network(DQN)

Deep Q-Network pekiştirmeli öğrenmenin bir modeli olan Q-Learning algoritmasının daha kapsamlı uzaylarda çalışılabilmesi için ismine *deep* kısmını da vermesine sebep olan Derin öğrenmesi eklenmiş halidir.

Q-Learnig kısaca oyunun bulunan durumunda (state) yapılabilecek en iyi (en yüksek Q değerini veren) hamlesini bulabilmek için state (action) ->Q Value (Toplam Ödül) değerlerini Q-Table adı verilen bir tabloda tutması üzerine çalışıyor. Oyunun mevcut durumunda, eğer tabloda daha düşük bir değere sahip bir State-Action (Durum-Eylem) çifti varsa, yapılan yeni hamlenin tabloda yer bulmasını göz önünde bulundurarak, State-Action çiftlerinin deneme yanılma sürecinde bulduğu en büyük değerlerle oyunu oynamak için bir talimat tablosu tutuyor diyebiliriz.



Resim8: Q-Table [3]

8. resimde de görüldüğü üzere, oyunun o anki durumuna bağlı olarak her eylemi karşılayacak şekilde bir tablo oluşturuyoruz.

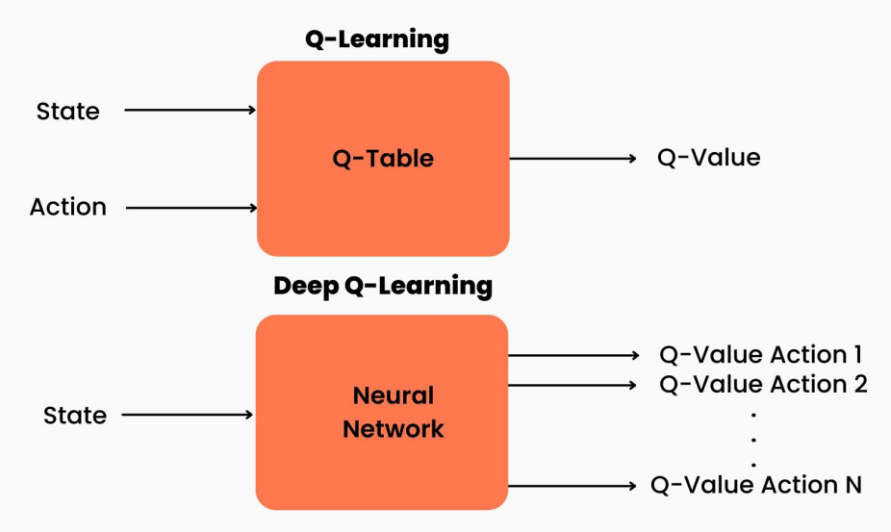
Mangala üzerinden örnek vermek gerekirse, başlangıç durumunda tablomuzun tüm değerleri 0 ile dolduruluyor. Herhangi bir state-action çiftini oynadığımızda; örneğin, oyunun başlangıcında tahta durumu (State = [4,4,4,4,4,4,0,4,4,4,4,4,4,0]) iken alınan 4 (action=4) numaralı eylemin bize Q-value olarak 0.01 döndürdüğünü düşünelim. Elde ettiğimiz bu değeri, ilgili state'in ve action'ın karşılık geldiği kısmına kaydediyoruz. Ardından, eğitim sırasında oyunun başlangıç durumuna geri dönülüp başka bir hamle yapıldığını ve bu hamlenin 5 olduğunu varsayalım. Ancak, bu sefer, öncekine göre Q-value'nun 0.001 değer aldığını düşünelim. Tablomuza yine state-action çiftini dolduruyoruz. Böylece, Q-learning modelimiz, bir test zamanında oyunun başlangıç durumuna denk gelen bir anda daha yüksek Q-value'lu 4 (action=4)'ü seçmesi gerektiğini anlayarak işlemini tamamlayacak.

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

Resim9: Q-Learning Optimum Policy formülü[4]

9. resimde de görüldüğü üzere optimum policy'e verilen state için optimum table'in içeren State-Action çifti için maksimum değeri döndürmesi ile gerçekleşiyor

Tahmin edileceği üzere, Q-Learning kompleks yapılar üzerinde düşük verimli bir model olucaktır. Çünkü tüm State-Action çiftlerini oynamadığı sürece yüksek başarı oranları elde etmesi zor olabilir. Bu yüzden, geliştirilmiş bir versiyon olan DQN modeli, tüm State-Action çiftlerini tutmak yerine onlara yakınsamak amacıyla yapay sinir ağı kullanıyor.



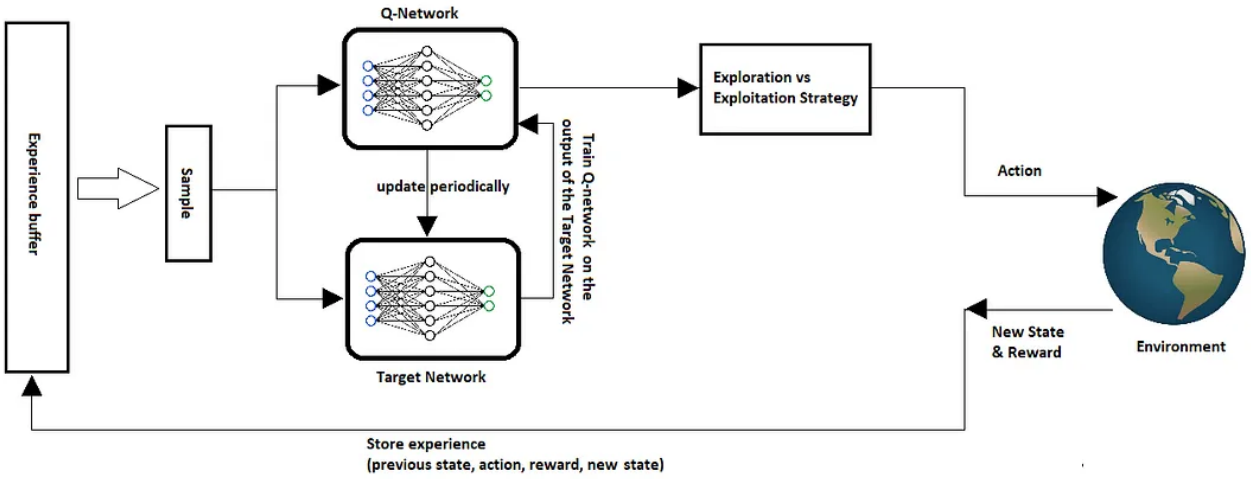
Resim10: Q-Learning ve DQN [5]

DQN ve Q-Learning modellerinin farklarının anlaşılabileceği 10. resimde, DQN için bir durum (state) verildiğinde yapay sinir ağı tarafından tüm aksiyonlar için Q-Value (Toplam ödül) değerleri yakınsanıyor ve model, bu Q-Value'ların en büyüğünü seçerek kararını alıyor.

$$Q(s,a) = Q(s,a) + lr*(r(s,a) + Y*max(Q(s',a')) - Q(s,a))$$

Resim11: Q-Value Bellman Denklemi [6]

(*"Q(s,a)" (durum, eylem) dediğimiz değer bizim şu anda {bulunduğumuz, gideceğimiz} dizin, "lr" ise değer öğrenme katsayısı (learning rate),"r(s, a)" değeri bizim {bulunduğumuz, gideceğimiz} ödül tablomuzdaki ödül değerimiz, "Y" değeri gamma, "max (Q (s',a'))" değeri ise gidebileceğimiz {gideceğimiz yerden gidebileceğimiz} yerlerin en yüksek Q değeridir.*)[6]



Resim12: DQN mimarisi[7]

12. resimde görüldüğü gibi, DQN mimarisinde kullanılan iki önemli bileşen bulunmaktadır: Q ağı (Q network) ve hedef ağı (target network).

Q Ağı (Q Network): Q ağı bir sinir ağıdır ve mevcut durumdan her aksiyonun Q değerini tahmin eder. Yani, ağa bir durum verildiğinde, her aksiyonun olası getirisini belirler.

Hedef Ağı (Target Network): Hedef ağı, başka bir sinir ağıdır ve Q ağının eğitilmesinde kullanılır. Ancak, bu ağın parametreleri belirli bir periyot boyunca (örneğin, birkaç adımda bir) sabitlenir. Hedef ağın sabitlenmiş olması, eğitim sürecini daha kararlı ve güvenilir kılar.

Bu iki ağın kullanılmasının temel nedeni, Q ağını güncellerken hedef değerlerin sabit kalmasını sağlamaktır. Eğer aynı ağı kullanırsanız, her güncelleme sonrasında hedef değerleri değişir ve bu durum algoritmanın kararlılığını zorlaştırabilir.

Bir sonraki aşamaya geçmeden önce bahsedilmesi gereken 3 önemli konu daha var;

1- Replay Buffer:

Replay buffer, ajanın geçmiş deneyimlerini depolayarak bu deneyimleri daha sonra tekrar kullanmasını sağlar. Bu durum, geçmişte öğrenilen deneyimleri unutmamayı ve bu işlem sırasında seçilen verileri rastgele bir şekilde yaparak veriler arasındaki korelasyonu azaltarak daha genelleyebilen bir modelin ortaya çıkmasını sağlar.

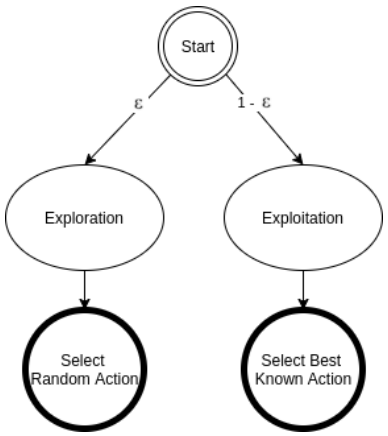
Projede replay buffer, namedtuple koleksiyon yapısı kullanılarak saklanıyor.

Transition = namedtuple('Transition', ('state', 'action', 'next_state', 'reward'))

2- Exploration ve Exploitation Dengesi:

Modelin sürekli olarak öğrendiği doğrultuda hareket seçmesi durumunda, başarılı bir oyun stratejisi geliştirmek yerine düşük ödüllerle yetinmesine neden olabilir. Bu durum, istenmeyen bir durumdur, çünkü eğer keşif yapmaya devam ederse, daha büyük ödüllere ulaşabilir ve daha güçlü rakipleri yenebilir hale gelebilirdi. Bu durumu yönetmek için her

eylem seçilirken E(epsilon) değeri kullanılır. Bu değer, modelin keşif yapma oranını belirler; yani, random bir değerle hiç bilmediği bir hamleyi yapacak mı, yoksa öğrendiklerini kullanarak kendi stratejisine göre maksimum ödülü mü seçecek, bu denkleme tabi tutularak belirlenir.



Resim13: Exploration ve Exploitation Dengesi [8]

Epsilon (E) değeri başlangıçta 1 olarak seçilir. Bu seçim nedeniyle, oyunun başında model, keşif yapmayı daha çok tercih eder. Sonrasında, tahminlerimiz doğrulandıkça epsilon değerini kademe kademe düşürerek, modelin bilinen en iyi hareketi yapmayı tercih etmeye başlamasını sağlarız.

3. İndirimli (Discounted) Gamma Değeri

İndirimli gamma, gelecekteki ödüllerin bugünkü ödürlere göre nasıl değerlendirileceğini kontrol eder. Eğer γ değeri 0'a yakınsa, ajan sadece anlık ödüllere odaklanır ve gelecekteki ödüllerin etkisi azalır. Tam tersine, γ değeri 1'e yaklaşırsa, ajan gelecekteki ödüllere daha fazla ağırlık verir. Bu ödül gamma değeri ile belirlenir ve genellikle 0.99 gibi bir değer seçilir. Bu, indirimin az olduğu ve gelecek ödüllere daha fazla ilgi gösterilen bir durumu ifade eder.

5. Self-Play

Şu ana kadar tezimde paylaştıklarımda, modelin nasıl kurulduğunu ve nasıl tahmin yaptığını detaylı bir şekilde açıkladım. Ancak, mangala gibi iki kişilik bir oyunun öğrenilmesi için rakip olarak bir eylem seçiciye daha ihtiyaç duyuluyor. Bu durumda, bilgisayarın karşısına geçip bir oyuncu tarafından eğitilmesi, milyonlarca mangala oyununu insan tarafından oynaması anlamına gelir. Bu durum, oldukça gerçekçi bir yaklaşım olmayabilir. Bu nedenle, bu problemi çözmek için self-play yöntemini kullanıyoruz.

Temel amaç, karşı oyuncunun da bir model tarafından çevreyi kendi içinde işleyip sonuç döndürmesini sağlamaktır. Karşı hamleleri belirleyen yapay zeka da, aslında bizim eğittiğimiz modelin kullanımıyla gerçekleşir. Ancak, burada bir sorun ortaya çıkar: Doğrudan modeli kullanmak verimsiz olabilir, ana modelin daha düşük seviyeli bir halini kullanmak da

çok az bilgiye sahip bir model anlamına gelir, ana modelimizin savunmaya karşı etkili bir strateji geliştirmemesine yol açar .Güçlü bir rakip kullanmak da model karşısında doğrudan öğrenememesine neden olabilir. Bu sorunu çözmek için ana modelin, bir hiper parametre kadar belirlenmiş eski bir sürümü ile(örneğin, 20.000 epoch öncesini) oynamasını sağlayarak modelin kendi kendine oynaması sağlanabilir.

6. Yol Ayrımları

Proje kapsamında kararın tamamen bende olduğu deneme yanılma yoluyla kontrollerini sağladım iki durum açığa çıktı;

1. Geçersiz Hamle

Sinir ağını 14x128x6 olarak tasarladığımızda, durumlar için 6 çıktıdan birini baskın olarak seçebileceğimizi biliyoruz. Ancak, mangala oyunu gereği bazı durumlarda oyuncunun kendi kuyuları 0 taşa sahip oluyor, bu da o taşların oynanamayacağı anlamına geliyor. Bu durumda yapay sinir ağımız, boş kuyulardan birini oynamayı mantıklı bulursa, oyun sonsuz bir döngüye girip ileri bir duruma geçemez hale geliyor. Bu durumu düzeltmenin iki yöntemi bulunmaktadır:

1.Durumda, eğer geçersiz hamle (boş kuyuyu işaret eden) yapılırsa, bu hamleyi cezalı bir hamle olarak seçip oyuncuya yeniden hamle yapmasını isteyerek boş kuyuları seçmemesini bekleyebiliriz. Ancak, bu yöntem, hala daha küçük de olsa bir olasılıkla boş kuyuyu seçmeyi tercih edebileceğinden, ikinci yöntemi projede kullanmayı tercih ettim. Ayrıca, bu yöntem öğrenmeyi bir hayli yavaşlatabilirdi.

2.Durumda ise, modelin çıktılarının en yüksek olasılığı boş kuyuyu seçmesini engellemek için seçilen kuyunun doluluğunu kontrol ediyoruz. Eğer boşsa, modelin bir sonraki en yüksek olasılıklı hamleyi yapmasını ve boş olmayan bir kuyu bulana kadar bu işlemi döngüsel olarak tekrarlamasını sağlıyoruz, böylece en yüksek olasılıklı (Q-value) kuyunun seçimini gerçekleştiriyoruz.

2. Ödül Tasarımı

Ödül tasarımında birkaç yöntem bulunmaktadır. İkili oyunculu rekabet oyunlarında genellikle kullanılan yöntem, (1, 0, -1) yaklaşımıdır. Yani, kazanırsa +1 ödül, kaybederse -1 ödül alır ve berabere kalması durumunda 0 ödül verilerek sistemin yaptığı hamlelerin doğruluğunu anlamasını bekleriz. Ancak, mangala oyunu için bu tür bir öğrenme süreci oldukça uzun sürmekte ve modelin öğrenimde başarılı olduğunu kontrol ettiğimiz yakalama (capture), çift kuyu yakalama ve yeniden hamle kazanma gibi özelliklerini kazanmakta zorlandığını gözlemlemekteyiz. Bu sorunu çözmek amacıyla, sistemin daha hızlı ve başarılı bir model olabilmesi için alt hedefler (subgoals) kullanmaya karar verdim.

Alt Hedefler:

Capture (Yakalama): Oyuncunun son taşının kendi boş kuyusuna gelmesiyle o kuyuya attığı taş ve o kuyuya karşıt rakip kuyunun taşlarının toplamı alınması durumunda, veya tam tersi rakibinin böyle bir hamle yapması durumunda, artı/eksi hazineye geçen taş sayısı /100 kadar ödül verilir.

Capture_even (Yakalama_Çift): Oyuncunun son taşının rakip oyuncunun kuyusuna eklenmesinden sonra, eğer kuyudaki taş sayısı çift sayıya ulaşıyorsa, rakip kuyunun taşlarının hepsinin kendi hazinesine geçirilmesi durumunda, artı/eksi hazineye geçen taş sayısı /100 kadar ödül verilir.

Go_Again (Yeniden Hamle): Oyuncunun son taşının kendi hazinesinde bitmesi durumunda, oyuncu yeniden hamle yapma hakkı kazanır. Bu durumda 0.001 ödülü verilerek modelin yeniden oynama hakkı kazanması durumunun artırılması hedeflenir.

Ana Hedefler:

Kazanma Durumu: +50 ödül

Kaybetme Durumu: -50 ödül

Beraberlik Durumu: 0 ödül

Sonuç

Eğitim sırasında modelin performansını kontrol etmek için birkaç değişken durumunu izledim ve ödül değerlerini bu gözlemlere göre ayarladım. Bu ölçütler şunları içeriyordu:

Her 500 epoch'ta bir, modelin ortalama kaç kez "capture" ve "capture_even" (yakalama ve yakalama_çift) hamleleri yaptığını. Bunun yanı sıra, ortalama kaç kez "go_again" (yeniden oynama hakkı) kazandığını da gözlemledim. Son olarak, genel olarak hangi açılış hamleleri ile oyuna başladığını belirlemek için bir kod bölümü ekledim.

```
ÇAĞ: 324500 Reward: 104154.09999473534 Draw:27565 Player1:158854 Player2:138082 Capture_mean:17.412 Capture_mea
n_enem:6.62 Go_again_mean:9.702 opennig_moves:{'2-2': 80532, '5-5': 19320, '1-1': 15882, '0-0': 9859, '3-3': 81
30}
ÇAĞ: 325000 Reward: 104233.28999470171 Draw:27634 Player1:159153 Player2:138214 Capture_mean:16.538 Capture_mea
n_enem:6.244 Go_again_mean:9.64 opennig_moves:{'2-2': 80532, '5-5': 19330, '1-1': 15882, '0-0': 9859, '3-3': 81
30}
```

Resim14: Deploy Ekranı

14. Resimde görüldüğü üzere, modelin Capture yapma yeteneğini geliştirdiğini ve rakip oyuncunun capture'ına karşı kendini savunduğunu görmekteyiz. Ancak, açılış hamleleri kısmında yetersiz kaldığı görülüyor. Bu durumun sebebi, DQN mimarisinin istikrarsız olması ve DQN den ötürü oyunun başında modelin öğrendiği stratejilerin gelecekte çok fazla değişmemesidir.

Kaynakça

[1] Mangala. (Tarih yok). Nedir, Ne Demek? Mangala. https://www.mangala.com.tr/nedir-ne-demek-mangala

[2] Mangala. (Tarih yok). Mangala Nasıl Oynanır. https://www.mangala.com.tr/mangala-nasil-oynanir

[3] Zhao, M., Lu, H., Yang, S., & Guo, F. (2020, Mart). The Experience-Memory Q-Learning Algorithm for Robot Path Planning in Unknown Environment

[4] Hugging Face. [The Link Between Value and Policy]. Deep RL Course. https://huggingface.co/learn/deep-rl-course/unit2/q-learning-recap

[5] Turing. Differences between Q-learning and deep Q-learning. Turing.com. https://www.turing.com/kb/how-are-neural-networks-used-in-deep-q-learning

[6] Umutozel. (2019, Eylül 14). [Reinforcement Learning]. Umutozel.com. https://www.umutozel.com/reinforcement-learning

[7] Ziad SALLOUM.(2022, Ocak,7)[Practical Guide to DQN] https://towardsdatascience.com/practical-guide-for-dqn-3b70b1d759bf

[8] baeldung.(2023, Mart,24)[Epsilon-Greedy Q-learning]

https://www.baeldung.com/cs/epsilon-greedy-q-learning