YALOVA ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

- BİTİRME TEZİ -

DERİN EVRİŞİMLİ Q ÖĞRENME İLE PONG OYUNU

Samet KOCABAŞ

Bilal KORKMAZ

Bitirme Tezi Danışmanı: Dr. Öğr. Üyesi Adem Tuncer

YALOVA, 2020

YALOVA ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

DERİN EVRİŞİMLİ Q ÖĞRENME İLE PONG OYUNU

Samet KOCABAŞ

150101028

Bilal KORKMAZ

150101022

1. Bitirme Tezi Danışmanı : Dr. Öğr. Üyesi Adem TUNCER

2. Jüri Üyesi : Dr. Öğr. Üyesi Osman Hilmi Koçal

3. Jüri Üyesi : Dr. Öğr. Üyesi Murat Okkalıoğlu

Bitirme Tezinin Dönemi: 2019 – 2020 Bahar Yarıyılı

İÇİNDEKİLER

Sayfa

SİMGE LİSTESİ

ε	Epsilon
Γ, γ	Gama
V(s)	Durumun değeri
V(a)	Gidilecek aksiyon
V(s')	Gelecek durum değeri
V(a')	Bir sonraki aksiyon
R(a,s)	's' durumunda 'a' hareketini yaparak elde edeği ödül
Q(s,a)	Şuanda bulunan durum
max()	Parantez içinin maksimum değeri
P()	Parantez içinin olasılığı
∑	Toplam sembolü

KISALTMA LİSTESİ

DQL	Deep Q-Learning (Derin Q öğrenmesi)
DCQL	Deep Convolution Q-Learning (Derin evrişimli Q öğrenmesi)
TD	Temporal Difference (Geçici Fark)
DNN	Deep Neural Network (Derin sinir ağları)
PY	Pygame kütüphanesi

ŞEKİL LİSTESİ

ÖNSÖZ

Bu proje Yalova Üniversitesi Bilgisayar Mühendisliği bölümünde Bilgisayar Mühendisliği Bitirme Tezi olarak hazırlanmıştır. Bu projenin amacı, Derin Evrişimli Q Öğrenme algoritması kullanılarak Pong oyununu bilgisayara karşı oynayacak ajan eğitimini gerçekleştirmektir.Bu projede danışmanlığımızı yapan, bizi yönlendiren ve her konuda bizden yardımını esirgemeyen saygıdeğer hocamız Dr. Öğr. Üyesi Adem Tuncer ve okul hayatımız boyunca desteğini bizden esirgemeyen ailemize, hocalarımıza ve arkadaşlarımıza teşekkür eder, saygılarımızı sunarız.

Bilal KORKMAZ

Samet KOCABAŞ

ÖZET

Günümüzün ve geleceğin en önemli algoritmalarından bir tanesi olan Deep Learning algoritması sınıfının en önemli alt sınıflarından bir tanesi Derin Evrişimli Q öğrenmesi (DCQL)'dir. Yaygınlaşan ve gelişen bu algoritma sayesinde bir çok yapay zeka projeleri geliştirilmektedir. Anaconda yazılımı ve python dili aracılığıyla DCQL kullanılarak Pong oyununda ajanımızın bilgisayara karşı minimum hata ile oyunu oynaması amaçlanmıştır.

Anahtar Sözcükler: Q Öğrenmesi, Derin Q Öğrenmesi, Derin Evrişimli Q Öğrenmesi

ABSTRACT

DCQL is the one of the most important subclasses of deep learning algorithm which is one of the most important algorithms of today and future. Thanks to this widespread and developing algorithm, many artificial intelligence projects are being developed. Our agent is aimed to play the game with minimum error against computer by using Deep Convolutional Q-Learning through Anaconda software and python language.

Keywords: Q-Learning, Deep Q-Learning, Deep Convolutional Q-Learning

1.GİRİŞ

Teknolojinin bu kadar geliştiği bir dönemde akıllı cihazlar, kendi kendine hareket eden robotlar, insansız hava araçları ortaya çıkmış ve yapay zeka sistemler son derece önem teşkil etmeye başlamıştır. Gelişen teknoloji ile artan yapay zeka uygulamaları ile yakın gelecekte her işi, robotlar devralabilir. Pekiştirmeli Öğrenme ise bu amaçla gerçekleştirilen ve bir makinenin, deneme yanılma yolu ile kendi kendine öğrenmesini amaçlayan bir yapıdır. Pekiştirmeli öğrenme yapay zekaya en yakın konu olarak adlandırılır. Pekiştirmeli öğrenme, aslında daha yeterince gelişmemiş bir yapıdır ve dünyada kullanım alanları sınırlıdır. Bunun nedeni, bir önce ki cümlede bahsettiğimiz gibi, makine kendi kendine öğrenir be bu neden tam gelişmemiş olması için yeterli çünkü destekli öğrenmede hazır verilerle makine öğretmek çok daha kolaydır. Bu yapıda ise hazır veri kullanmadan, makinenin verilen ortamı keşfetmesiyle, öğrenimini gerçekleştirir ve bu ortamı hazırlamak her iş için mümkün değildir ve bundan dolayı kullanım alanları sınırlıdır ama bu teknolojinin gelişmesi ve farklı tekniklerin oluşturulması ile pekiştirmeli öğrenmenin yakın gelecekte her işi öğretmek mümkün olacaktır. O seviyeye gelene kadar, şimdilik bu alanda gelinen en son seviyeye nasıl gelindiğini ve en son seviyede nasıl bir çalışma sistemi olduğunu bu tezde ele alalım.

2. PEKİŞTİRMELİ ÖĞRENME

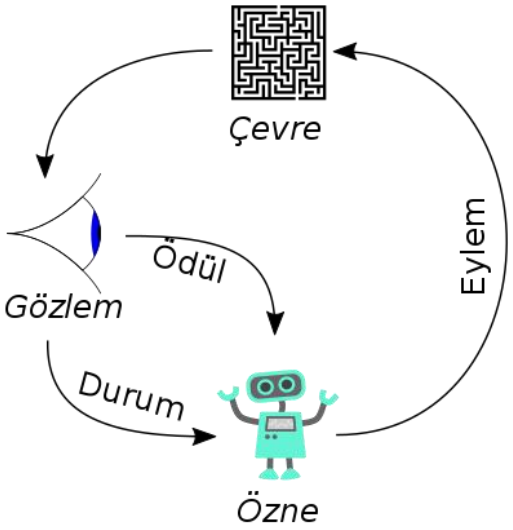
Pekiştirmeli öğrenme, amaca yönelik ne yapılması gerektiğini öğrenen bir makine öğrenmesi yaklaşımıdır. Pekiştirmeli öğrenmede ajan adı verilen öğrenen makinemiz karşılaştığı durumlara bir tepki verir ve bunun karşılığında da sayısal bir ödül sinyali alır. Ajan aldığı bu ödül puanını maksimuma çıkartmak için çalışır. Bu şekilde çalışan deneme yanılma yöntemi, pekiştirmeli öğrenmenin en ayırt edici özelliğidir.

2.1 Q Öğrenme

2.1.1 Politika, ödül ve durum

Politika; ajanın içinde bulunduğu durumda alabileceği aksiyonu belirler. Bir nevi etki tepki eşleşmesi olarak düşünülebilir. İçinde bulunulan durum bir etki olarak kabul edilirse ajan buna karşılık bir tepki (aksiyon) verir. Bu politika basit bir aksiyon olarak tanımlanabileceği gibi bütün durumları karşılayan bir arama tablosu şeklinde de tanımlanabilir. Politika dinamik olarak da nitelenebilir. Bunun temel nedeni, ajanın içinde bulunduğu durumu değerlendirerek alabileceği aksiyonları aramasından kaynaklanmaktadır.

Ödül; ajanının gerçekleştirmiş olduğu bir aksiyona karşılık çevreden aldığı puandır. Bir pekiştirmeli öğrenme ajanının amacı, uzun vadede aldığı ödülleri maksimum seviyeye ulaştırmaktır. Ödül alınan aksiyonun ne kadar iyi veya kötü olduğunu belirleyen değerdir (basit bir şekilde mutluluk veya acı ile eşleştirilebilir). Ajan, izlemiş olduğu politikayı bu ödülleri esas alarak zaman içerisinde değiştirir. Örneğin alınan bir aksiyonun sonrasında düşük bir puan elde ediliyorsa, gelecekte ajan aynı duruma geldiğinde farklı bir aksiyon almayı tercih edebilir.



Şekil 2.1 Ajan, durum, aksiyon ve politika

Durum; ise ajanın içinde bulunduğu durumdan ve o durumu takip eden diğer durumlardan bekleyebileceği ödüllerin toplamıdır. Ödüller anlık olarak neyin iyi neyin kötü olduğunu ifade ederken, durum değeri uzun vadede neyin iyi neyin kötü olduğunu ifade eder. Örneğin; bir durum, düşük bir ödüle fakat yüksek bir değere sahip olabilir. Bunun nedeni düşük ödül veren durumu takip eden yüksek ödüllü diğer durumlardır. Tam tersi de mümkündür. Yüksek ödül veren bir durumdan sonra sürekli olarak düşük ödüller veren durumlar da olabilir. Buradaki durum "ileri görüşlülük" gibi düşünülebilir.

2.1.2 Deterministik ve Stokastik

Deterministik : Bir iş yaptığımızda çıkışın beklediğimiz şekilde olmasıdır. Yani ben boş bir odadayım öne bir adım atmak istiyorum. Adım atıyorum ve bir adım öndeyim o noktaya gidememem gibi bir durum söz konusu değil. Yani ben %100 olasılıkla öne gidebilmeliyim.İşte bu tür harekete(action) deterministic action denir.

Stokastik: Burada ise satrançtaki veziri düşünelim. Vezire %100 öne gider diyemeyiz. Sağa ,sola veya çapraz gidebilir. Yani ortada bir olasılık var. Değişkenliğe örnek olarak günlük hayatı düşünebiliriz.

2.1.3 Bellman denklemi

Ajanın ayak izlerini iki farklı yönde görülmesi durumunda, hedefe ulaşmak için hangi yoldan gidileceğine karar veremez. Çünkü ajanın ilerleyecek talimatları hatırlamanın bir yolu yoktur. Ajanı bir hafıza ile etkinleştirmek için Bellman Denklemi devreye girer. Bu denklem sayesinde ajanın çevre içinde rastgele aksiyonlar yaparak değil mantıklı bir yol çıkarımı yapmasını sağlar. Bellman denklemi q learning algoritmasının temelini oluşturur.

2.1.4 Markov karar süreci

Markov Karar Süreçleri, eylemlerin sadece bir sonraki ödülü değil gelecek durumları da etkilediği sıralı karar verme sürecinin klasik bir formülasyonudur. Aynı zamanda pekiştirmeli öğrenme probleminin matematiksel olarak idealleştirilmiş biçimleridir. Markov karar süreçlerinde; durum ve ödül sadece bir önceki durum ve aksiyona bağlı olan, çevrenin dinamikleri tarafından belirlenen, iyi tanımlanmış ayrık bir olasılık dağılıma sahiptir. Bu etkileşimin geçmiş durumlardan sadece bir önceki duruma bağlı olması, durumların geleceğe etki edebilecek tüm geçmiş ajan-çevre etkileşimlerini kapsaması sebebi ile olmaktadır. Durumların bu özelliğine Markov Niteliği denir.

Pekiştirmeli öğrenmede veya markov karar süreçlerinde öğrenen ve karar veren elemana ajan denir. Bu ajanın etrafını kapsayan ve onun etkileşimde bulunabileceği her şeyi içinde barındıran alana ise çevre denir. Çevre, ajanın eylemlerine karşılık gelen ödülü verir ve ajan için eylem alması gereken yeni bir durum oluşturur. Bu şekilde, çevre ve ajan ikisi sürekli etkileşim halinde kalır.

Bellman denklemi

$$V(s) = {}^{max}_{a}(R(a,s) + \gamma V(s')) \tag{2.1}$$

Markov karar süreci denklemi

$$V(s) = {}^{max}_{a}(R(a,s) + \gamma \sum_{s'} (\mathrm{P}(\mathrm{s},\mathrm{a},s')V(s'))) \tag{2.2}$$

2.1.5 Geçici fark

Pekiştirmeli öğrenmenin temel yöntemlerinden biri olan zamansal fark öğrenmesi, dinamik programlamada olduğu gibi en son sonucu beklemeden diğer tahminlerden yardım alarak tahminlerini günceller. Ortamın modelini öğrenmeye ihtiyaç duyulmaz. Herhangi bir sabit politika için yeteri kadar küçük adım boyu kullanılarak en iyi politikaya yakınsar.

Geçici fark denklemi

$$R(s,a) + \gamma max\, Q'(s',a') - Q(s,a) \tag{2.3}$$

2.1.6 Q öğrenme algoritması ve q tablosu

Q öğrenmesi algoritması, pekiştirmeli öğrenmenin en çok bilinen algoritmalarından biridir. Algoritmadaki temel amaç bir sonraki hareketleri inceleyip yapacağı hareketlere göre kazanacağı ödülü görmek ve bu ödülü maksimize ederek buna göre hareket etmektir. Ajan ödüle giderken her tekrarlamada edindiği tecrübeleri gidebildiği yerleri seçerken maksimize etmek için kullanır. Bu tecrübeleri ise Q tablosu adı verdiğimiz bir tabloda tutar.

Q tablomuz başlangıçta ajanımızın hiçbir tecrübesi olmadığı için sıfırlarla doludur ve bu yüzden ajanımız ilk seçimlerinde Q tablosundaki sıfırları maksimize edeceğinden rastgele hareket edecektir. Bu rastgelelik ajanın ilk ödülü bulmasına kadar sürecektir. Ajan ödülün olduğu bir yere geldiği anda ödüle gelmeden önceki yerini bilir ve bu yerin değerini kendi

tecrübelerini biriktirdiği Q tablosuna yazar. Bu yazma işleminin belirli bir algoritması bulunmaktadır.

Q öğrenmesi denklemi

$$Q(s,a) = R(s,a) + \gamma \sum_{s'} \left(\mathrm{P}(\mathrm{s},\mathrm{a},s') \max_{a'} (Q(s',a')) \right) \tag{2.4}$$

Her seferinde ajan bu algoritmaya göre bir ilerisini tahmin edip hareket eder ve ödüle ulaşır. Ödüle ulaştıktan sonra ise ajan tekrardan rastgele bir yerden harekete başlar ve tekrardan ödülü bulmaya çalışır. Bu işlem devam ettikçe ajan hangi durumda nereye gideceğini bilmeye başlar.

2.1.6 Keşif ve Uygulama

Pekiştirmeli Öğrenme sürecindeki en önemli zorluklar, keşif ve uygulama kavramlarının uygulamaya geçirilmesidir. Keşif-Uygulama ikilemi optimum bir şekilde çözülmeye çalışılmaktadır. Ajanın daha fazla ödül elde etmesi için geçmişte denediği ve pozitif ödül aldığı eylemleri seçmelidir. Ajan ödül elde etmek için daha önce deneyimlediği eylemlerden yararlanır, ancak karşılaştığı bir durumda daha fazla ödül alabileceği eylemler varsa bunları da keşfetmelidir. Böylece ajan, çeşitli eylemler denemeli ve en iyi sonuç/ödül alabildiklerini aşamalı olarak desteklemelidir.

Şekil 2.2 Keşif ve uygulama

2.1.6 Yaşam Cezası

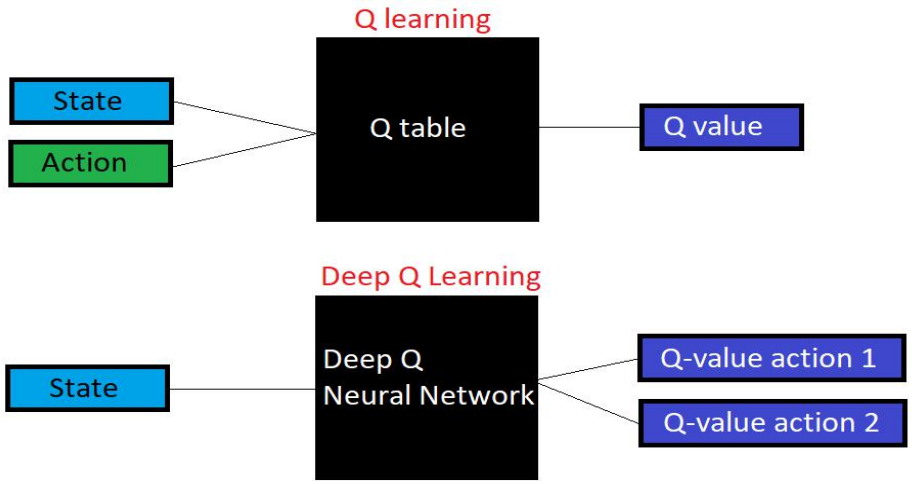
Ajanı sadece hedefe ulaştığında ödüllendiririz ve ideal olarak, ajanın eylemlerinin kalitesini daha iyi değerlendirmesine yardımcı olmak için aldığı her işlem için bir ödül

olmalıdır. Ödüllerin her zaman aynı olması gerekmez. Ancak, aksiyonlar için bir miktar ödül almak, hiç ödül almamaktan çok daha iyidir. Bu fikir yaşam cezası olarak bilinir.

2.2 Derin Q Öğrenme

2.2.1 Derin q öğrenme ve q öğrenme

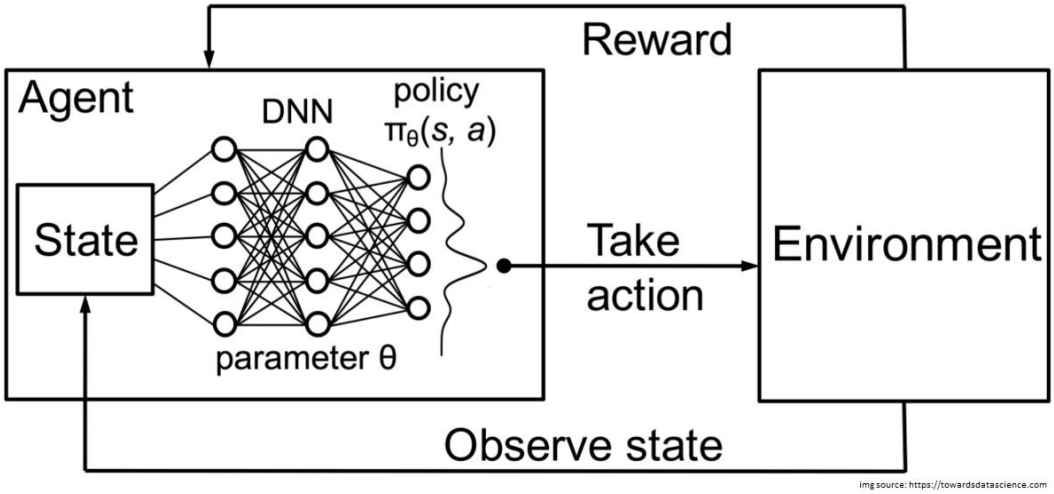
Neden Q öğrenmesi yeterli olmuyor ve derin Q öğrenmesine ihtiyaç duyarız ? Bu sorunun üç temel nedeni vardır. Birincisi durum kümesi, ajanımızın çevre içinde dolandığı durumlar vardır. Bu durum kümelerinin on binlere, yüz binlere hatta milyonlara çıktığı durumlar olabiliyor. Durumların artmasında zamansal olsun, zaman olarak q tablosu geçte olsa oluşturulur ama donanımsal olarak, q tablosundaki değerlerin alıp verilmesi durum kümesi arttıkça imkansızlaşıyor.İkincisi kaynak kitlenmesi,q tablosunda olabilecek her durum oluşturulur ama bazı çevrelerde bu durumlar yetersiz kalabilir ve ajanın gittiği duruma karşılık gelen değer q tablosunda bulunamayabilir.Üçüncüsü ise q öğrenmesinde kompleks çevrelerde çalışmasında büyük performans kayıplarına neden olur. Örneğin kompleks bir oyunda sağ sol yukarı aşağı yönleri dışında milimlik dönüşlerde oyununun sonucuna etkisi fazla olabilir. Bu nedenle q tablomuzda milimleri oluşturmak istediğimizde, aşırı dolar ve performans kaybına neden olur. Bu üç ana nedenden dolayı derin q öğrenmesi geliştirilmiştir.



Şekil 2.3 Q Öğrenmesi ve derin q öğrenmesi farkı

Derin Q öğrenmesinin, q öğrenmesinden temel farkı q tablosu yerine sinir ağlarının kullanılmasıdır. Sinir ağları modelini çevrenin ve ajanın özelliklerine göre en ideal model şekilde oluşturulur. Çevreden gelen gözlem durumları, ajana durum olarak iletilir. İletilen

durumlar, sinir ağlarında giriş değerleri olarak verilir. Giriş değerleri model içinde işlemini tamamladıktan sonra ajanın kaç tane aksiyon değeri varsa modelden de o kadar çıktı elde ederiz. En yüksek çıkan çıkışı ise sonucu olarak ve çevre içinde ona göre hareket eder.



Şekil 2.4 Derin q öğrenmesi model

2.2.2 Deneyim tekrarı

Derin q öğrenmesinde zamanla öğrenen ajanımızın, kazandığı, öğrendiği tecrübelerini unutmamasını sağlamak ve geçmişte öğrendiği bilgilerinden yararlanmasını sağlayan, deneyimlerin saklandığı alandır. Bu alan sınırsız yapılarak ajanımızın bütün öğrendikleri tutabilir ama bu bize alandan kayıp ettireceği için genellikle sınırlandırılır ve son öğrenilen tecrübe eklenerek, ilk öğrenilen tecrübe alandan çıkarılır.

2.2.3 Uyarlanabilir Epsilon Değeri

Q öğrenmesi konusunda gördüğümüz keşif ve uygulama kısmını derin q öğrenmesinde uyarlanabilir olarak epsilon değerini tek bir değer olarak değil, ajan öğrenimini geliştirdikçe keşif kısmını azaltabilmemiz için zamanla epsilon değerini de azaltırız. Bunun amacı, en ajanımızın başta çevreyi iyice araştırmasını, daha sonrasında zamanla öğrenen, deneyimleri artan ajanımız çevreye uyum sağlamasını ve öğrendiği tecrübelerden devam edilmesini sağlamış oluruz.

2.3 Derin Evrişimsel Q Öğrenme

Derin Q öğrenme işleri çözmenin harika bir yoludur, peki bir adım daha ileri gidebilir miyiz? Ajana bir video oyunu oynamasını öğretebilir miyiz? Videoyu izleyerek, bunu yaptığımız şekilde yapmayı öğretebilir miyiz? Burada özel tür sinir ağları, Evrişimli Sinir Ağları (CNN) kullanılır. Bunun için, ajanının video oyunundaki kareleri bir girdi olarak kullanmasını sağlanarak, oyunun nasıl oynatacağını öğretiriz.

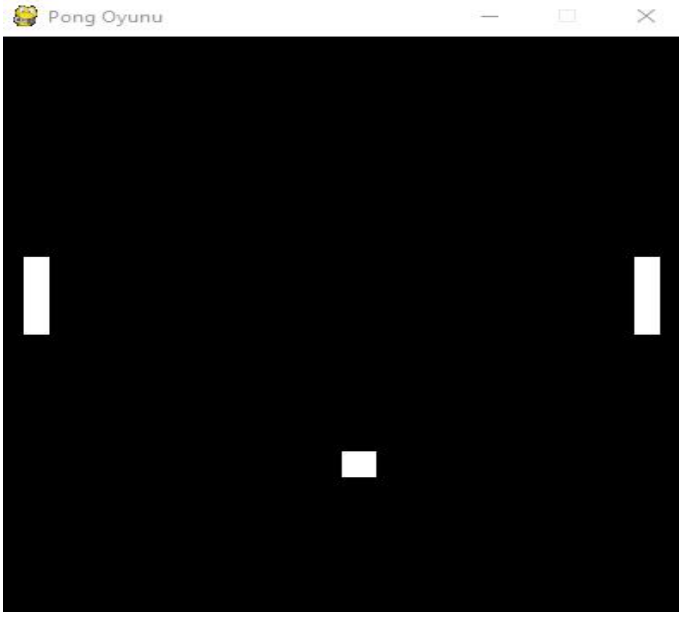
Özetlemek gerekirse, görüntüyü önce işlemek ve standart sinir ağının çalışabileceği bir şekle sokmak için CNN katmanları kullanılır. Bunu yapmanın ilk adımı , giriş görüntüsü üzerindeki belirli özellikleri veya nitelikleri tespit etmektir. Bu CNN tabaka ile yapılır. Bu tabaka, kenarlar ve eğriler gibi düşük düzeyli özelliklerin yanı sıra yüz veya el gibi daha yüksek düzeyli özellikleri algılamak için filtreler kullanır. Evrimsel Sinir Ağı, görüntüden doğrusallığı kaldırmak için ek katmanlar kullanır, bu da aşırı sığmaya neden olabilecek bir şeydir. Doğrusallık kaldırıldığında, görüntüyü sıkıştırmak ve verileri düzleştirmek için ek katmanlar kullanılır. Son olarak, bu bilgi CNN dünyasında tam bağlantılı katman adı verilen bir sinir ağına aktarılır. Ancak, bu tezin asıl amacı olan, bu kavramları Q öğrenme’ ile birleştirerek 3. bölümde kodumuzu inceleyeceğiz.

3. PONG OYUNU

Pong oyunu en eski arcade video oyunlarından bir tanesidir.Amaç masa tenisi simülasyonuna benzeyen oyunda daha yüksek puan kazanarak rakibi yenmektir. Pong oyunu ve ajanı nasıl oluşturacağımız hakkında bilgileri ve oluşturduğumuz kodu detaylarıyla bu bölümde inceleyelim.

3.1 Kodlama Planı

Pong oyunumuzun, yani çevremizin ve çevre içindeki bir raket olacak ajanın tasarımından biraz bahsedelim. Oyunumuzun görünümü için siyah ekran kullanacağız. Oyun içinde yukarı ve aşağı hareket edebilen, sağ ve sol kenarlarda iki raketimiz ve ekranın tam ortasında her açılışında rastgele bir noktadan başlayacak kare bir topumuz olacak. Sağ taraftaki raketimiz için topu takip edebilmesi için matematiksel kod yazacağız. Sol taraftaki raketimiz, bizim ajanımız olacak ve oymayı öğreteceğiz. Çevre ve ajan arasındaki etkileşimi kurabilmemiz için öncelikle iki model oluşturacağız. Birinci modelimiz, pygame kütüphanesi kullanarak yapacağımız pong oyunumuz, ikincisi ise CNN yapısı ile oluşturacağımız ajan.Ajan ve çevremiz direk etkileşim içinde olmayacağı için ayrı bir eğitim alanı oluşturup birbirileri ile etkileşim gerçekleştirerek,ajanımızın eğitimini gerçekleştireceğiz. Çevre, ajan ve eğitim için gerekli metotlardan sırasıyla alt bölümlerde bahsedeceğiz.



Şekil 3.1 Pong oyunu görseli

3.2 Çevre Tasarımı

Çevre, pygame kütüphanesi kullanılarak pong oyunumuzun oluşturulduğu sınıf olacak. Bu sınıfımız için gerekli değişken ve fonksiyonları açıklayalım.

3.2.1 Sabit değişkenler

Bu sabit değişkenlerimiz için, öncelikle oluşturacağımız oyun için windows ekranı genişlik ve yüksekliği tanımlanacak. Belirlenen ekran yüksekliğine eşit olacak şekilde oyun ekranını yüksekliği tanımlanacak ve bunlar aynı değerlerde olacak. Aynı değerde olmasının sebebi, eğitim için oyundan alacağımız ekran görüntülerinin kare olmasını sağlamak. Oyunumuz için gerekli raketler ve topumuz için genişlik, yükseklik,hız ve sadece raketler için ayrıca tampon tanımlanacak. Tampon, raketlerin oyun ekranından sağ ve sol kenarlarındaki mesafeyi belirtecek değerdir. Raket ve topumuzun değişkenlerini de belirledikten sonra oyunun rengi için siyah ve beyaz renklerini tanımlayacağız. Siyah oyun ekranı, beyaz ise raketlerin ve topun rengi için kullanacağız. Sonrasında Frame Per Second (FPS) belirliyoruz. FPS, saniyede ki kare değişim hızını temsil edecek. Bütün oyun içi değişkenlerimizi oluşturduk ve son sabit değişken olarak pygame kütüphanesini kullanarak windows ekranımızı oluşturmak ve kullanmak için ekran oluşturuyoruz.

```
#Ekrenın ve oyunun genişliğini,yukseklıgını tanımlama
EkranGenisligi = 400
EkranYuksekligi = 500
OyunYuksekligi = 400

#Raketlerin genislik yukseklik ve tamponunu tanımlama
RaketGenisligi = 15
RaketYuksekligi = 60
RaketAraligi = 15

# Topun genisligini ve yuksekligini tnaımla
TopGenisligi = 20
TopYuksekligi = 20

#Top ve Raketlerin hızlarını belirleme
RaketHareketHizi = 3
TopXEksendeHizi = 2
TopYEksendeHizi = 2

#Oyun ekranı ve raketleri için RGB renkler
RGBBeyazRenk = (255,255,255)
RGBSiyahRenk = (0,0,0)

#Skorların ekranda gözüken yazı tipi
YaziFontu = YaziTipi.match_font('arial')

OyunEkrani = Goruntule.set_mode((EkranGenisligi,EkranYuksekligi))
```

Şekil 3.2 Pong için sabit değişkenler

3.2.2 Başlatıcı metodu

Burada pong oyunu için sınıfımıza oluşturmaya başlayacağız. Bir sınıf tanımlıyoruz ve sınıfımızın ilk çağrıldığında ekranın ve çevrenin oluşturulmasını sağlayacak başlatıcı fonksiyonunu kodluyoruz. Bu fonksiyon için ilk olarak pygame'i başlatıyoruz. Önceki bölümlerde de bahsettiğimiz gibi pygame kütüphanesi bize istediğimiz ortamın oluşturulmasında yardımcı olacak. Windows ekranımız için yani, oyunumuz açıldığında barda gözükmesi istediğimiz isim için pygame set_caption'ı çağırıyoruz ve isim belirliyoruz. Daha sonrasında raketlerimiz ve topumuz için başlangıç pozisyonlarını ve ayrıca topumuza ekstra random olarak belirlenecek yönünü tanımlayacağız. Topun yönünün random olarak seçilmesinin nedeni, aslında ajanımız için işleri zorlaştırmaktır. Topu hep şekilde harekete başlatırsak, ajanımız bunu bi kaç eğitimde öğrenir. Ayrıca skoru ve geçen zamanı tutmamız için zaman ve skor değişkenleri tanımlıyoruz.

```
class PongOyunu:

    def __init__(self):
        PY.init()
        Goruntule.set_caption("Pong Oyunu")

        self.AjanRaketYEkseni = OyunYuksekligi/2 - RaketYuksekligi/2
        self.NormalRaketYEkseni = OyunYuksekligi/2 - RaketYuksekligi/2

        self.AjanSkor = 0
        self.NormalSkor = 0
        self.OyunSkor = 0.0

        self.TopXYonu = random.sample([-1,1],1)[0]
        self.TopYYonu = random.sample([-1,1],1)[0]

        self.TopXEkseni = EkranGenisligi/2
        self.TopYEkseni = random.randint(0,9)*(EkranYuksekligi - TopYuksekligi)/9
```

Şekil 3.3 Başlatıcı fonksiyonu

3.2.3 Görüntüleme metodu

Pong Oyunu için oluşturduğumuz sınıfta, pozisyonlarını ve yönlerini belirlediğimiz raketlerimizin ve topun, sabit değişken oluşturduğumuz ekran üzerinde görüntülenmesini sağlayacak. Bu fonksiyonda da öncelikle pygame kütüphanesinin .event.pump() fonksiyonu

yardımıyla, yazdığımız oyunun işletim işlemiyle uyumlu olarak çalışmasını sağlayacağız. Sabit değişkenlerde oluşturduğumuz ekranın içini seçtiğimiz renk ile dolduruyoruz. Sonrasında raketlerimiz ve topumuz için ekrana çizilmesini sağlayacak fonksiyonlarımızı çağırıyoruz ve sonunda pygame'in display.flip() fonksiyonunu çağırarak, şimdiye kadar yarattığımız her şeyin ekranda sürekli güncellenecek şekilde bize gösterilmesini sağlayacak.

```
def EkrandaGoruntuOlustur(self):

    Olay.pump()

    OyunEkrani.fill(RGBSiyahRenk)

    RaketOlustur("Ajan", self.AjanRaketYEkseni)
    RaketOlustur("Normal",self.NormalRaketYEkseni)

    TopOlustur(self.TopXEkseni, self.TopYEkseni)

    Goruntule.flip()
```

Şekil 3.4 Görüntüleme fonksiyonu

Görüntüleme metodunu bahsettiğimiz şekilde tamamladık. Şimdi içindeki kodlamamız gereken raketlerin çizimi için gerekli fonksiyondan bahsedelim. Bu fonksiyondan biri sağ roket, yani ajan, çizimi diğeri sol raket çizimi için iki kere aynı fonksiyonu çağıracağız ve parametrelerinden biri sağ veya sol raket olduğunu belirtecek, diğeri ise gönderdiğimiz raketin y ekseninde ki pozisyonunu, hareketini sağlamak için, parametre olarak vermemiz gerekiyor.Fonksiyonumuz içinde öncelikle sağ veya sol raket için if komutu ayrım yapıyoruz. Daha sonra her raketimiz için pygame kütüphanesinden yararlanarak konumlarını parametre vererek .Rect() fonskiyonu ile dikdörtgen raketlerimizi oluşturuyor ve ekrana çizimlerini gerçekleştirmek için tekrar pygame kütüphanesinden .draw.rect() fonksiyonunu ekran, renk ve oluşturduğumuz raketleri parametre vererek bu fonksiyonumuzu tamamlıyoruz.

```
#Ekrana Raket çizdirme Fonksiyonu
def RaketOlustur(RaketTipi, RaketYEkseni):

    #hangi raketin cizilmesi gerektigini kontrol etmek için if komutu yap ve ona göre sag ve solda raketleri çizdir
    if RaketTipi == "Ajan":
        Raket = PY.Rect(RaketAraligi, RaketYEkseni, RaketGenisligi, RaketYuksekligi)
    elif RaketTipi == "Normal":
        Raket = PY.Rect(EkranGenisligi - RaketAraligi - RaketGenisligi, RaketYEkseni, RaketGenisligi, RaketYuksekligi)

    Ciz.rect(OyunEkrani, RGBBeyazRenk, Raket)
```

Şekil 3.5 Raket oluştur fonksiyonu

Raketlerimizin çizimlerini gerçekleştirdikten sonra şimdi topun çizimi için gerekli fonksiyondan bahsedelim. Bu fonksiyonumuz için öncellikle, x ve y pozisyonlarımızı parametre olarak fonksiyon içinde gönderiyoruz. Topumuz içinde raketlerde kullandığımız pygame fonksiyonlarını kullanarak, topu oluşturuyor ve çizimini gerçekleştiriyoruz.

```
#Topun ekranda görüntelenmesi için çizen fonksiyon
def TopOlustur(TopXEkseni, TopYEkseni):

    Top = PY.Rect(TopXEkseni, TopYEkseni, TopGenisligi, TopYuksekligi)

    Ciz.rect(OyunEkrani, RGBBeyazRenk, Top)
```

Şekil 3.6 Top oluştur fonksiyonu

3.2.4 Güncelleme ve Hareket

Bu bölümde, fonksiyonumuzdan bahsetmeden önce güncellemenin gerekli method ve parametrelerinden bahsedelim. Öncelikle raketlerimiz için gerekli aksiyon kümemizi belirleyelim. Aksiyon kümemiz için bu oyunda raketimiz için 3 tane aksiyonu olabilir; yukarı, aşağı ve hareketsiz kalmak. Raketlerin hareket edeceği yönümüzü belirleyip, fizikteki en temel konulardan hız hesabı ile yani, x = v.t +x kullanarak raketin konumunu güncelliyoruz. Konumunu güncellediğimiz raketin ekranda görüntüsünü de güncellememiz yani, ekranda tekrar çizdirme işlemi uygulamamız gerekir. Aynı işlemleri topumuz için de yapacağız ve

skorları da burada güncelleyeceğiz. Öncelikle delta zamanını belirlememiz lazım ve .event.pump() fonksiyonunu çağırıyoruz. Skor değerini tanımlıyoruz. Her güncellemede ekranımızı yeniden oluşturmamış gerektiğini unutmayalım ve bu yüzden ekran için her güncellemede tekrardan çizdirmemiz gerekecek. Bu yüzden, ekran için tekrardan rengini doldurma kodumuzu yazıyoruz. Ve burada hareket fonksiyonu için belirlenen aksiyon parametresini kullanarak, raketlerin nasıl ve hangi yönde hareket ettiğini güncelleyecek ve bize raketin y pozisyonunu geri döndürecek fonksiyonumuzu yazıyor ve başlatıcı da belirlediğimiz raketlerin y pozisyonuna eşitliyor ve bir önceki bölümde yaptığımız raket çizme metodu ile tekrar çizilmesini sağlıyoruz.Aynı şekilde topumuzu güncelleyeceğiz ama topun bize yönü ve pozisyonu dışında, bize skor döndürecek ve top güncelleme fonksiyonu ile topumuzun konumunu güncelleyeceğiz.

```
def HareketEttir(self, Aksiyon):

    Olay.pump()

    Skor = 0

    OyunEkrani.fill(RGBSiyahRenk)

    self.AjanRaketYEkseni = RaketGuncelle("Ajan", Aksiyon, self.AjanRaketYEkseni, self.TopYEkseni)
    RaketOlustur("Ajan", self.AjanRaketYEkseni)

    self.NormalRaketYEkseni = RaketGuncelle("Normal", Aksiyon, self.NormalRaketYEkseni, self.TopYEkseni)
    RaketOlustur("Normal", self.NormalRaketYEkseni)

    [Skor, self.TopXEkseni, self.TopYEkseni, self.TopXYonu, self.TopYYonu] = TopGuncelle(self.AjanRaketYEkseni,
    self.NormalRaketYEkseni, self.TopXEkseni, self.TopYEkseni, self.TopXYonu, self.TopYYonu)

    TopOlustur(self.TopXEkseni, self.TopYEkseni)

    if Skor == 0.05:
        self.AjanSkor +=1
    if Skor == -10:
        self.NormalSkor +=1

    if ( Skor > 0.5 or Skor < -0.5):
        self.OyunSkor = self.OyunSkor*0.9 + 0.1*Skor

    EkranGoruntusu = DiziEkran.array3d(Goruntule.get_surface())
    SkorYazdir(OyunEkrani,str(self.AjanSkor),0)
    SkorYazdir(OyunEkrani,str(self.NormalSkor),1)
    Goruntule.flip()

    return [Skor, EkranGoruntusu]
```

Şekil 3.7 Hareket ettir fonksiyonu

Şimdi raketlerin y pozisyonunu güncelleyecek fonksiyonumuzdan bahsedelim. Parametreleri sırasıyla, sağ veya sol raketin olduğunu belirten değişken, hangi aksiyonu gerçekleştireceğini

belirten değişken,raketin ve topun y pozisyonlarını parametre olarak belirliyoruz. İçinde ise delta zamanı sabit olarak 7.5 alıyoruz. Pyhton'dan aldığımız saat değerini kullanarak oluşturulan delta zamanı sağlıklı sonuç alamıyoruz. Sonrasında çizimde gerçekleştirdiğimiz gibi if komutu ile raket ayrımını yapıyoruz. Ajanımız için aksiyonlara göre pozisyon belirleyeceğimiz için, hangi aksiyonunun parametre olarak gönderildiğini kontrol ediyoruz ve gelen aksiyona göre y pozisyonumuzu yukarı veya aşağı olacak şekilde güncelliyoruz. Raketin hareket etmemesi durumda kendi değerine eşit olacağı için bunu kodumuzda yazmaya gerek yoktur. Ayrıca raketlerimizin ekrandan çıkmaması için y pozisyonunun sıfırdan küçük olması durumda y pozisyonumuzu sıfıra eşitliyoruz ve aynı işlemi aşağıdan dışarı çıkmaması için yükseklik kontrolü yapıyor ve yüksekliği geçtiği durumda, maksimum alabileceği yüksekliğe eşitliyoruz. İkinci raketimiz, yani matematiksel olarak sadece topu takip edecek şekilde kodumuzu yazıyor ve ajanda yaptığımız gibi ekrandan çıkmaması için kontrolleri gerçekleştiriyoruz. Bu fonksiyonumuz geri dönen fonksiyon olduğu için güncellenmiş y pozisyonumuzu geri döndürüyoruz.

```
#Raketlerin konumu guncelleme
def RaketGuncelle(RaketTipi, Aksiyon , RaketYEkseni, TopXEkseni):
    DeltaZaman = 7.5

    #ajan raket için aldığı aksiyonlara göre konumunu güncelle
    if RaketTipi == "Ajan":
        if Aksiyon == 1:
            RaketYEkseni = RaketYEkseni - RaketHareketHizi*DeltaZaman
        if Aksiyon == 2:
            RaketYEkseni = RaketYEkseni + RaketHareketHizi*DeltaZaman

        if RaketYEkseni < 0:
            RaketYEkseni = 0
        if RaketYEkseni > OyunYuksekligi - RaketYuksekligi:
            RaketYEkseni = OyunYuksekligi - RaketYuksekligi
    #normal raket için topun takibini sağlayacak sekilde konum guncelle
    elif RaketTipi == "Normal":
        if RaketYEkseni + RaketYuksekligi/2 < TopXEkseni + TopYuksekligi/2:
            RaketYEkseni = RaketYEkseni + RaketHareketHizi*DeltaZaman
        if RaketYEkseni + RaketYuksekligi/2 > TopXEkseni + TopYuksekligi/2:
            RaketYEkseni = RaketYEkseni - RaketHareketHizi*DeltaZaman

        if RaketYEkseni < 0:
            RaketYEkseni = 0
        if RaketYEkseni > OyunYuksekligi - RaketYuksekligi:
            RaketYEkseni = OyunYuksekligi - RaketYuksekligi
    return RaketYEkseni
```

Şekil 3.8 Raket konumu güncelle fonksiyonu

Raket güncelleme işleminden bahsettiğimize göre topun durumunu güncelleme işleminden bahsedebiliriz. Top güncelleme fonksiyonu bize 5 tane değer geri döndürecek. Bunlar; skor, topun x, y pozisyonu ve topun x, y hareket yönü. Fonksiyonun parametreleri ise; iki raketin pozisyonu, topun x, y pozisyonu ve topun x, y hareket yönü olacak şekilde fonksiyonu oluşturacağız. İçinde ise raket güncelleme metodunda yaptığımız delta zamanı sabit olarak 7.5 alıyoruz. Sonrasında ise parametreden aldığımız topun şimdiki pozisyonlarını delta time ile güncelliyoruz ve yeni pozisyonunu buluyoruz. Bu işlemi hem x hem de y pozisyonları için gerçekleştiriyoruz. Topumuzun konumunu güncelledik ve buna göre skorumuzu bulalım. Teorik kısımda gördüğümüz yaşam cezası için skorumuzu öncellikle -0.05 olarak belirliyoruz. Bir daha açıklamak gerekirse, yaşam cezası, ajanımızın boş geçirdiği veya oyalandığı durumlarda ceza vermektir. Daha sonrasında güncellenen topun konumuna göre raketlere denk gelip gelmemesini kontrol ederek, eğer ajanımızın olan raket topu yakalamışsa ajanımız +10 puan alacak ve topun yönü değişecek, eğer ajanımız topa karşılık verememiş ise -10 ceza verilecek ve top oyun ekranına çarptıktan sonra geri dönüşü sağlanacaktır. Aynı işlemleri diğer normal raketimiz için yapacağız ama tek fark skorlara etki etmeyecek. O yüzden ajanımız olmayan raketin eğer topun karşılanma durumda ilk başa verdiğimiz yaşam cezasını geri +0.05 olarak geri veriyoruz. Bu işlemleri gerçekleştirdikten sonra başta belirttiğimiz 5 değeri geri döndürüyoruz. Şekil 3.9'da fonksiyonu inceleyebiliriz.

Raket ve Top güncellemelerini gerçekleştirdik. Top güncellemesinden gelen skoru kontrol ederek 10 veya -10 gelmesi durumunda oyun skorumuzu güncelliyoruz. Oyun skorumuz ajanımızın öğrendiği skordur yani ödüldür. Diğer skorlar ise raketlerin kaç defa topu kaçırdıklarını gösteren skorlar. Derin Evrişimli Q Öğrenmede bahsettiğimiz gibi resimlerle öğrenme gerçekleştireceği için her hareket için görüntü almamıza yardımcı olacak pygame kütüphanesinden .surfarray.array3d() fonksiyonunu kullanıyoruz ve sonrasında skor ve aldığımız görüntünün eğitim alanında kullanacağımız için geri dönüşünü sağlıyoruz.

```
#Raketlerin konumu guncelleme
def TopGuncelle(AjanRaketYEkseni, NormalRaketYEkseni, TopXEkseni, TopYEkseni, TopXYonu, TopYYonu):

    dft = 7.5

    TopXEkseni = TopXEkseni + TopXYonu*TopXEksendeHizi*dft
    TopYEkseni = TopYEkseni + TopYYonu*TopYEksendeHizi*dft

    YasamCezasi = -0.05
    # agent
    if (TopXEkseni <= (RaketAraligi + RaketGenisligi + TopGenisligi/2)) and ((TopYEkseni + TopYuksekligi -1) >= AjanRaketYEkseni)
    and ((TopYEkseni-1) <= (AjanRaketYEkseni + RaketYuksekligi)) and (TopXYonu == -1):
        TopXYonu = 1

        YasamCezasi = 10

    elif (TopXEkseni <= TopGenisligi/2):

        TopXYonu = 1

        YasamCezasi = -10

        return [YasamCezasi, TopXEkseni ,TopYEkseni ,TopXYonu, TopYYonu]

    if (TopXEkseni >= (EkranGenisligi - RaketGenisligi - RaketAraligi - TopGenisligi))
    and ((TopYEkseni + TopYuksekligi)>= NormalRaketYEkseni) and (TopYEkseni <= (NormalRaketYEkseni + RaketYuksekligi))
    and (TopXYonu == 1):

        TopXYonu = -1

    elif(TopXEkseni >= EkranGenisligi - TopGenisligi):

        TopXYonu = -1
        YasamCezasi = 0.05
        return [YasamCezasi, TopXEkseni,TopYEkseni, TopXYonu, TopYYonu]

    if TopYEkseni <= 0:

        TopYEkseni = 0

        TopYYonu = 1

    elif TopYEkseni >= OyunYuksekligi - TopYuksekligi:

        TopYEkseni = OyunYuksekligi - TopYuksekligi

        TopYYonu = -1

    return [YasamCezasi, TopXEkseni,TopYEkseni,TopXYonu,TopYYonu]
```

Şekil 3.9 Top konumu güncelleme fonksiyonu

3.3 Ajan

Çevremizi yarattık, şimdi bu çevreyi kullanarak eğiteceğimiz ajan için bir sınıf oluşturacağız. Bu sınıfta ajanın öğrenmesi için gerekli değişkenler ve fonksiyonlar yer alacak.

3.3.1 Sabit değişkenler

Ajan sınıfımız için sabit değişkenlerimizi belirleyelim. Önceki bölümde bahsettiğimiz aksiyon sayımızı burada değişken olarak tanımlayalım. Kullanacağımız resim boyutlarını ve her deneyim tekrarı için kaç resmin tutulacağını belirten değişkenleri tanımlayalım. Oyunumuzun ekranı 400x420 e olacak şekilde ayarlandı ama eğitimde bu boyutu kullanmak,

performansımızı azaltacaktır ve hatta eğitimi hiç bir zaman bitiremeyebiliriz. O yüzden yakalanan resimlerin boyutunu modelde 40x40 boyutlarında işleteceğiz. Ayrıca, keşif yapmayı azaltmak zamanla azaltmak için yani uyarlanabilir epsilon değerini belirlemek için öncelikle alt sınır , gamma değeri ve deneyim tekrarı kapasitesi ve batch size tanımlıyoruz.

```
AksiyonSayisi = 3
ResimYuksekligi = 40
ResimGenisligi = 40
ResimSiniri = 4

Gozlem = 2500
GamaDegeri = 0.975
BatchBoyutu = 64

DeneyimTekrariKapasite = 2000
```

Şekil 3.10 Ajan sabit değişkenler

3.3.2 Başlatıcı metodu

Ajanımızın başlatıcı fonksiyonunda, öncelikle parametre olarak, kaydedilen hazır modelimiz varsa, onu kullanabilmek için değişken tanımlıyoruz. İçinde ise, modelimizi, deneyimleri, epsilon değerimizi ve uyarlanabilir epsilon değerini güncellemek için adım değeri değişkenleri oluşturuyoruz. Model değişkeni, duruma göre parametreden gelen yoksa bir sonraki bölümde bahsedeceğimiz sinir ağı modelinden eğitimle kendisi oluşturacak, eğer model zaten parametrede verilmiş ise oyunu o modele göre oynayacak.

```
class Ajan:

    def __init__(self,HazirModel=""):
        if(HazirModel == ""):
            self.Model = self.ModelOlustur()
        else:
            self.Model = keras.models.load_model(HazirModel)
        self.DeneyimTekrari = deque()
        self.AdimSayisi = 0
        self.EpsilonDegeri = 1.0
```

Şekil 3.11 Ajan başlatıcı fonksiyonu

3.3.3 Sinir ağları modeli

Burada ajanımızın öğrenmek için yararlanacağı CNN oluşturulacak. CNN modelimizi oluştururken keras kütüphanesinden yararlanacağız. 32, 64, 64 tane nöron olmak ve aktivasyon fonksiyonları ‘relu’ olacak şekilde 3 katman oluşturuyoruz. Sonrasında 512’li Dense katmanı ve son olarak Dense ile aksiyon sayısı kadar sonuç verecek ve aktivasyon fonksiyonu ‘linear’ olacak şekilde çıkış katmanı oluşturuyoruz ve modelimizin kaybını ortalama kare hatası, ve iyileştirici olarak ‘adam’, öğrenme oranını otomatik olarak ayarlayan, olacak şekilde derliyoruz. Derlenen modelimizi geri döndürüyoruz.

```
def ModelOlustur(self):
    Model = Sequential()

    Model.add(Conv2D(32, kernel_size=4, strides = (2,2),
                     input_shape = (ResimYuksekligi,ResimGenisligi,ResimSiniri),padding = "same"))
    Model.add(Activation("relu"))
    Model.add(Conv2D(64,kernel_size=4,strides=(2,2),padding="same"))
    Model.add(Activation("relu"))
    Model.add(Conv2D(64,kernel_size=3,strides=(1,1),padding="same"))
    Model.add(Activation("relu"))
    Model.add(Flatten())
    Model.add(Dense(512))
    Model.add(Activation("relu"))
    Model.add(Dense(units= AksiyonSayisi, activation="linear"))

    Model.compile(loss = "mse", optimizer="adam")

    return Model
```

Şekil 3.12 Model oluştur fonksiyonu

3.3.4 Aksiyon seçimi

Bu fonksiyon ajanımız için bir sonra ki en iyi adımını bulmasını sağlayacaktır. En iyi adımı bulmak için keşif ve uygulama kısmını burada kullanacağız. Random 0 ile 1 arasında bir değer alınarak epsilon değeri ile karşılaştıracağız eğer ki epsilondan küçük olması durumda deneyerek yeni aksiyonlar üretecek, yani keşfedecek veya öğrendiği kısımlardan kendine yön bulacak ve bulunan adım geri döndürülecek. Şekil 3.13’de aksiyon seçme fonksiyonu kodunu görebiliriz.

```
def EnIyiAksiyonuBul(self, GoruntuGirdileri):
    if random.random() < self.EpsilonDegeri or self.AdimSayisi < Gozlem:
        return random.randint(0,AksiyonSayisi - 1)
    else:
        qvalue = self.Model.predict(GoruntuGirdileri)
        bestA = np.argmax(qvalue)
        return bestA
```

Şekil 3.13 Aksiyon seçme fonksiyonu

3.3.5 Depolama

Bu fonksiyonda, elde ettiğimiz resim örneklerini deneyim tekrarına depoluyoruz. Burada yapmamız gereken ekstra şey; eğer depomuz dolmuş ise ilk eklenenleri çıkarmak ve yenisi eklemek. Ayrıca bu fonksiyonda başlatıcı metodunda oluşturduğumuz adım sayılarını sayacağız ve ilerlenen adımlara göre zamanla epsilon değerimizi düşürme işlemini yapıyoruz.

```
def Depolama(self, Durum):
    self.DeneyimTekrari.append(Durum)
    if len(self.DeneyimTekrari) > DeneyimTekrariKapasite:
        self.DeneyimTekrari.popleft()

    self.AdimSayisi += 1

    self.EpsilonDegeri = 1.0
    if self.AdimSayisi > Gozlem:
        self.EpsilonDegeri = 0.75
        if self.AdimSayisi > 7000:
            self.EpsilonDegeri = 0.5
        if self.AdimSayisi > 14000:
            self.EpsilonDegeri = 0.25
        if self.AdimSayisi > 30000:
            self.EpsilonDegeri = 0.15
        if self.AdimSayisi > 45000:
            self.EpsilonDegeri = 0.1
        if self.AdimSayisi > 70000:
            self.EpsilonDegeri = 0.05
```

Şekil 3.14 Deneyim depolama fonksiyonu

3.3.6 Süreç metodu

Bu fonksiyon, ajanımızın eğitileceği süreci işlemektedir. Ajanımızın eğitime başlaması için öncelikle deneyim tekrarı kapasitesinin dolması lazım. Kapasite dolduktan sonra içinden sabit değişken olarak tanımladığımız batch sayısı kadar örnek alıyoruz. Bunları eğitim için kullanacağız. Batch sayımız kadar ve 4 sonuç verecek kadar girdi ve aksiyon sayımız kadar sonuç verecek hedef değerleri oluşturuyoruz. Eğitimi gerçekleştirmek için aldığımız örneklerden durumları, aksiyonları,ödülleri ve bir sonraki durumları alarak girdi değerlerimizi dolduruyoruz. Hedef değerlerini doldurmak için ise modelimizden tahmin alıyoruz ve aksiyon ve ödül değerleri ile dolduruyoruz. Sonra modelimizi tekrardan fit ediyoruz. Ayrıca her 10.000 adımda bir modelimizin h5 olarak kaydediyoruz.

```
def EgitimSureci(self):
    if self.AdimSayisi > DeneyimTekrariKapasite:
        Ornekler = random.sample(self.DeneyimTekrari, BatchBoyutu)
        OrnekUzunlugu = len(Ornekler)

        Girdiler = np.zeros((BatchBoyutu,ResimYuksekligi,ResimGenisligi,ResimSiniri))
        Hedefler = np.zeros((Girdiler.shape[0],AksiyonSayisi))

        for i in range(OrnekUzunlugu):
            Durum = Ornekler[i][0]
            Aksiyon = Ornekler[i][1]
            Odul = Ornekler[i][2]
            GelecekDurum = Ornekler[i][3]

            Girdiler[i:i + 1] = Durum
            Hedefler[i]  = self.Model.predict(Durum)

            if GelecekDurum is None:
                Hedefler[i,Aksiyon] = Odul
            else:
                Hedefler[i,Aksiyon] = Odul + GamaDegeri*np.max(self.Model.predict(GelecekDurum))

        self.Model.fit(Girdiler, Hedefler ,batch_size= BatchBoyutu, epochs=1, verbose=0)
    if self.AdimSayisi % 10000 == 0:
        self.ModelKaydet(self.AdimSayisi)
```

Şekil 3.15 Eğitim süreci fonksiyonu

3.4 Eğitim

Bu bölümde oluşturduğumuz çevre ve ajanı birbirleri ile etkileşime sokarak eğitimi gerçekleştireceğiz.

3.4.1 Sabit değişkenler

Sabit değişkenlerimiz, toplam eğitim süresi, eğitimde kullanılacak resim boyutları ve her deneyim tekrarı için kaç resim kullanılacağı değerleri tanımlıyoruz.

```
EgitimDonguSayisi = 400000

ResimYuksekligi = 40
ResimGenisligi = 40
ResimSiniri = 4
```

Şekil 3.16 Eğitim sayfası sabit değişkenler

3.4.2 Süreç öncesi resimleri

Normalde resimlerimiz 400x420'dir. Eğitimi gerçekleştirebilmemiz için gerekli olan resim boyutlarına yani 40x40'a dönüştüreceğiz. Bu işlemde öncelikle resimlerimizi gri resme çeviriyoruz. Daha sonra gerçek boyuttaki resimlerimizi skimage kütüphanesinden yararlanarak yeniden istediğimiz boyutlarda yeniden boyutlandırma yapıyoruz. Boyutlandırma işleminden sonra elde ettiğimiz resmi normalize ederek geri dönüşünü sağlıyoruz.

```
def GoruntuHazirlama( Goruntu ):

    GriGoruntu = Grilestir(Goruntu)

    KirpilanGoruntu = GriGoruntu[0:400,0:400]

    HazirGoruntu = YenidenBoyutlandir(KirpilanGoruntu,(ResimYuksekligi,ResimGenisligi))

    HazirGoruntu = YogunlukAyarla(HazirGoruntu, out_range = (0,255))

    HazirGoruntu = HazirGoruntu / 128

    return HazirGoruntu
```

Şekil 3.17 Eğitim sayfası görüntü hazırlama fonksiyonu

3.4.3 Model eğitim

Çevre ve ajanı etkileşime sokarak eğitim tecrübeleri elde edeceğiz. Öncelikle depolamak için boş bir alan tanımlıyoruz. Oyunumuzu, yani çevremizi tanımlayıp, başlatıcı ile oyunumuzu ekranda görüntülüyoruz. Çevre ortamını ayarladıktan sonra ajanımızı yaratıyoruz. İlk başta bir aksiyon seçmemiz lazım. Bu aksiyon rastgele olarak veya kendimizde belirleyebiliriz. Belirlenen ilk aksiyon, oyun üzerinde hareket gerçekleştirilerek ekran görüntüsü ve skor alınır. Oluşturduğumuz deneyim tekrarı 4, en az 4 resimden yararlanarak sonuç çıkartacağı için alınan ekran görüntüsünü yığın haline getiriyoruz. Buraya yaptığımız işlem alsında, ilk adımı gerçekleştirmekti gibi düşünülebilir. Ajanımızın en iyi aksiyonun bulması için gerekli olan 4 resim görüntüsünü, eğitim döngüsüne girmeden önce manuel olarak yarattık. Daha sonra eğitim döngüsü oluşturacağız. Döngü içinde ilk olarak, ajana, manuel olarak oluşturulan 4 resim yığınını en iyi aksiyonu bulması için parametre olarak verilir ve bulunan en iyi aksiyon ise çevre içinde o aksiyon gerçekleştirilerek, yeni skor ve yeni görüntüyü elde ederiz. Yeni görüntümüzü süreç öncesi resim fonksiyonunu kullanarak dönüştürürüz. Dönüşen görüntüyü ve manuel olarak oluşturulan 4 görüntünün birincisini çıkartır ve yeni görüntümüzü ekleyerek, ilk gelecek durumu elde etmiş oluruz. Daha sonra ajanımız, elde edilen örnekleri, durumu, gelecek durumu,skoru ve en iyi aksiyonu depolama işlemi yaparız. Depolama işlemini de yaptıktan sonra elde ettiğimiz bilgiler doğrusunda ilk eğitim gerçekleştirilir. İlk eğitim gerçekleştirdikten sonra gelecek durumunu, yeni durum olarak belirlenir ve döngü bu şekilde devam eder. Eğitim geçmişini, ilk başta oluşturulan boş alana, istenilen aralıkta depolanması sağlanır. Artık her şey hazır. Eğitimi başlatabiliriz. Şekil 3.18’ de eğitim için ajan ve çevre etkileşimi için gerekli kodu görebiliriz.

```
def Egitim():

    EgitimGecmisDepolama = []

    Cevre = DCQL_Pong.PongOyunu()

    Cevre.EkrandaGoruntuOlustur()

    Ajan = DCQL_Agent.Ajan()

    RandomAksiyon = 0

    [BaslangicSkoru, BaslangicEkranGoruntusu] = Cevre.HareketEttir(RandomAksiyon)
    BaslangicOyunGoruntusu = GoruntuHazirlama(BaslangicEkranGoruntusu)

    GoruntuYigini = np.stack((BaslangicOyunGoruntusu,BaslangicOyunGoruntusu,BaslangicOyunGoruntusu,BaslangicOyunGoruntusu),axis = 2)

    GoruntuYigini = GoruntuYigini.reshape(1, GoruntuYigini.shape[0],GoruntuYigini.shape[1],GoruntuYigini.shape[2])

    for i in range(EgitimDonguSayisi):

        RandomAksiyon = Ajan.EnIyiAksiyonuBul(GoruntuYigini)
        [Skor, YeniEkranGoruntusu] = Cevre.HareketEttir(RandomAksiyon)

        YeniOyunGoruntusu = GoruntuHazirlama(YeniEkranGoruntusu)

        YeniOyunGoruntusu = YeniOyunGoruntusu.reshape(1,YeniOyunGoruntusu.shape[0],YeniOyunGoruntusu.shape[1],1)

        GelecekGoruntuYigini = np.append(YeniOyunGoruntusu, GoruntuYigini[:,:,:,:3], axis = 3)

        Ajan.Depolama((GoruntuYigini,RandomAksiyon,Skor,GelecekGoruntuYigini))

        Ajan.EgitimSureci()

        GoruntuYigini = GelecekGoruntuYigini

        if i % 100 == 0:
            print("Egitim Süresi: ",i, " Oyun Skoru: ",Cevre.OyunSkor)
            EgitimGecmisDepolama.append(Cevre.OyunSkor)
```

Şekil 3.18 Ajan ve çevre etkileşimiyle eğitim

4. SONUÇLAR VE TARTIŞMA

Bu tezde, yapay zekanın alanında makine öğrenmesi tekniklerinden pekiştirmeli öğrenmenin teorik olarak 2 algoritmasından ve pratik olarak ise DCQL algoritmasından ve mantığından bahsettik. Girişte bahsettiğimiz gibi, bu algoritma aslında çok gelişmemiş bir alandır ve kullanım alanları oldukça sınırlıdır ama bu değil ki bu teknoloji her zaman böyle kalacak. Gelişen teknoloji ve geliştirilen yeni algoritmalarla, bu alan yakın zamanda her işlemi yapabilecek bir şekilde geliştirilebilir. Örneğin alpha go, bu teknikle geliştirilmiş bir yapay zekadır. Bir kaç sene öncesine kadar bu teknik geliştirilmeden önce go oyununu oynayan bir bilgisayarın yapılması çok uzak gelecekte olabileceğini tahmin ediyorlardı. Pekiştirmeli öğrenmenin gelişmesi bu işlemi oldukça hızlandırdı ve alpha go, dünyanın en iyi go oynayan insanını yenmeyi başarabiliyor. Diğer bir örnek; spacex, uzay taşımacılığı şirketinde kullanılan roketlerin inişlerini bu teknik ile makineye öğreterek gerçekleştiriyorlar veya diğer bilinen sürücüsüz araçlar. Zamanla pekiştirmeli öğrenme daha da gelişecek ve bir insanın yapabileceği her şeyi bir makine yapacak, üstelik en kısa zamanda ve en iyi şekilde. Bu belki yakın bir gelecek değil ama gerçekleşmesi de imkansız değil. Bu tezde, yaptığımız uygulamada ise bir makinenin aynı bir insan gibi öğrenebileceğini gördük. Nasıl bir insan gibi ? Örneğin; sokakta yürürken topun size doğru geldiğinizi gördüğünüzde beynimiz bize tepki verir ve size çarpmasını engellemek için kaçarsınız veya topu tutmaya çalışırsınız. Peki topun bize geldiğini beynimiz nasıl algılar ve bu uygulama ile bağlantısı nedir ? Her saniyeyi bir fotoğraf olarak düşündüğümüzde bunu anlayabiliyoruz. Beynimizde o fotoğraflardan topun yönünü anlayıp bize doğru geldiğinin farkına varıyor. Aynı mantık ile makineye pon oyunumuzu öğrettik, oyun içinde alınan görüntülerle topun yönünü ve hatta hızını bile algılayarak bir insanın öğrenme süresinden, daha kısa sürede oynamasını öğrettik. Bu basit bir oyun olduğundan bir insanın, bu oyunu öğrenmesi daha kısa sürebilir ama bir makine kadar kusursuz öğrenme gerçekleştiremez. Buradan sonuçla, bu öğrenme tekniğin zamanla gelişmesi her şeyin bir makine tarafından öğrenilebileceğini mümkün kılar. Belki bir gün kendi kendine model oluşturan bir model bile yapılabilir. İşte o zaman yapay zekadan korkmaya başlayabiliriz.

KAYNAKLAR

[1]http://karpathy.github.io/2016/05/31/rl Erişim Tarihi:01 Mayıs 2020

[2]https://blog.floydhub.com/an-introduction-to-q-learning-reinforcement-learning/ Erişim Tarihi:01 Mayıs 2020

[3]https://medium.com/deep-learning-turkiye/q-learninge-giri%C5%9F-6742b3c5ed2b Erişim Tarihi:01 Mayıs 2020

[4]https://medium.com/deep-learning-turkiye/peki%CC%87%C5%9Fti%CC%87rmeli%CC%87-%C3%B6%C4%9Frenmeye-gi%CC%87ri%CC%87%C5%9F-seri%CC%87si%CC%87-1-8f5c35b6044 Erişim Tarihi:01 Mayıs 2020

[5]https://mc.ai/makine-ogrenmesi-q-learninge-giris/ Erişim Tarihi:02 Mayıs 2020

[6]https://towardsdatascience.com/states-actions-rewards-the-intuition-behind-reinforcement-learning-33d4aa2bbfaa Erişim Tarihi:02 Mayıs 2020

[7] https://yz-ai.github.io/blog/pekistirmeli-ogrenme/sonlu-markov-karar-surecleri-bolum-3 Erişim Tarihi:02 Mayıs 2020

[8] L. Acosta, J. J. Rodrigo, J. A. Mndez, G. N. Marichal and M. Sigut, "Ping-pong player prototype", IEEE Robot. Autom. Mag., vol. 10, no. 4, pp. 44-52, Dec. 2003.

[9] C. H. Lai and T. I. J. Tasy, "Self-learning for a humanoid robotic ping-pong player", Proc. Adv. Robot., pp. 1183-1208, 2011.