

**ANKARA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ**

YÜKSEK LİSANS TEZİ

**PEKİŞTİRMELİ ÖĞRENME TABANLI ROBOTLAR İLE YENİ BİR
ROBOCODE SAVAŞ STRATEJİSİ**

Hakan KAYAKÖKÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

**ANKARA
2019**

Her hakkı saklıdır

TEZ ONAYI

Hakan KAYAKÖKÜ tarafından hazırlanan **Pekiştirmeli Öğrenme Tabanlı Robotlar ile Yeni Bir Robocode Savaş Stratejisi** adlı tez çalışması 09/10/2019 tarihinde aşağıdaki jüri tarafından oy birliği / ~~oy çokluğu~~ ile Ankara Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Danışman : Doç. Dr. Mehmet Serdar GÜZEL

Ankara Üniversitesi Bilgisayar Mühendisliği Anabilim Dalı

Jüri Üyeleri:

Başkan: Dr. Öğr. Üyesi Mehmet DİKMEN

Başkent Üniversitesi Bilgisayar Mühendisliği Bölümü

Üye : Doç. Dr. Gazi Erkan BOSTANCI

Ankara Üniversitesi Bilgisayar Mühendisliği Anabilim Dalı

Üye : Doç. Dr. Mehmet Serdar GÜZEL

Ankara Üniversitesi Bilgisayar Mühendisliği Anabilim Dalı

Yukarıdaki sonucu onaylarım.

Prof. Dr. Özlem YILDIRIM
Enstitü Müdürü

ETİK

Ankara Üniversitesi Fen Bilimleri Enstitüsü tez yazım kurallarına uygun olarak hazırladığım bu tez içindeki bütün bilgilerin doğru ve tam olduğunu, bilgilerin üretilmesi aşamasında bilimsel etiğe uygun davrandığımı, yararlandığım bütün kaynakları atıf yaparak belirttiğimi beyan ederim.

04/09/2019


Hakan KAYAKÖKÜ

ÖZET

Yüksek Lisans Tezi

PEKİŞTİRMELİ ÖĞRENME TABANLI ROBOTLAR İLE YENİ BİR ROBOCODE SAVAŞ STRATEJİSİ

Hakan KAYAKÖKÜ

Ankara Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Mehmet Serdar GÜZEL

Son yıllarda yapay zeka, makine öğrenmesi ve robotik konularındaki araştırmalar teknolojinin gelişmesi ve sektörün istekleri ile birlikte önceki senelere göre dikkate değer gelişmeler göstermiştir. Literatürde bahsi geçen konular üzerine yapılan araştırmalarda görülmüştür ki, yapay zeka uygulamalarının en önemli uygulama alanlarından birisi de yapay zeka birimlerinin dışarıdan herhangi bir müdahale olmadan hareket edebilmesi anlamı gelen otomatik kontroldür. Bu kavrama çözüm niteliğinde geliştirilen pekiştirmeli öğrenme, davranış biliminden ilham alınan, ajanların tanımlı bir çevrede en yüksek ödül miktarına ulaşabilmesi için hangi eylemleri yapması gerektiği bilgisini ajana sunan oldukça başarılı bir makine öğrenmesi modelidir.

Bu çalışmada makine öğrenmesi konusu kapsamında geliştirilen pekiştirmeli öğrenme tabanlı algoritmaların RoboCode adı verilen 2001 yılında tasarlanmış, Java programlama dili tabanlı, içerisindeki sanal robotların programlanarak yönlendirilebildiği bir ortam sunan simülasyon içerisinde geliştirilmesini konusu incelenmiştir. Yapay zeka alanındaki gelişmeler ile birlikte RoboCode simülasyonu yapay zeka algoritmalarının uygulandığı, kendi kendine öğrenebilen robotların savaştırıldığı, geliştirilen yapay zekâ algoritmalarının güçlerinin ölçüldüğü bir ortam haline gelmiştir. Simülasyon, literatürde geliştirilen yapay zeka algoritmaların sonuçlarının görülmesi ve diğer algoritmalarla sahip robotlar ile savaştırılarak algoritma analizi yapılabilmesi açısından büyük önem taşımaktadır.

Önerilen tez konusunun amacı RoboCode simülasyonunda geliştirilecek evrimsel sinirsel ağ destekli pekiştirmeli öğrenme algoritması ile hedef robotu arena üzerinde eğitmek ve eğitilen robotun arenada diğer robotlara karşı galip gelmesidir. Bu kapsamda, arenadan elde edilen görüntüler dinamik olarak oluşturulmuş ve geliştirilen robot bu görüntüler vasıtasıyla eğitilmiştir. Eğitilen model, önceden tanımlı rakip robotlar ile birebir olarak savaştırılmış ve bu savaşlar sonucunda rakip robotlara karşı elde edilen skorlar ile modelin önemli ölçüde başarılı olduğu gözlemlenmiştir.

Eylül 2019, 54 Sayfa

Anahtar Kelimeler: Yapay Zeka, Pekiştirmeli Öğrenme, Makine Öğrenmesi, Evrimsel Sinir Ağları, RoboCode, Derin Sinir Öğrenme, Q Öğrenme

ABSTRACT

Master Thesis

A NEW BATTLE STRATEGY FOR ROBOCODE BASED ON REINFORCEMENT LEARNING

Hakan KAYAKÖKÜ

Ankara University
Graduate School of Natural and Applied Sciences
Department of Computer Engineering

Supervisor: Assoc. Prof. Dr. Mehmet Serdar GÜZEL

In the last decade, researches on artificial intelligence, machine learning and robotics have shown remarkable improvements compared to previous years with the development of technology and the demands of the sector. Researches on the subjects mentioned in the literature have shown that one of the most important application areas of artificial intelligence applications is the automatic control, which means that artificial intelligence units can move without any external intervention. Reinforcement learning, developed as a solution to this concept, is a highly successful model of machine learning, inspired by behavioral science, which provides the agent with information on what actions agents must take to achieve the highest amount of rewards in a defined environment.

In this study, reinforcement learning based algorithms examined in the scope of machine learning for RoboCode simulation, which is designed in 2001 and based on Java programming language which provides an environment for virtual robots that can be programmed and controlled with program. In the simulation, robots with different characters are fought in an arena provided by the simulation. With the developments in the field of artificial intelligence, RoboCode simulation has become an environment where artificial intelligence algorithms are applied. Using RoboCode, self-learning robots are fought and the power of artificial intelligence algorithms is measured. This simulation is great importance in order to see the results of artificial intelligence algorithms developed in the literature and to perform algorithm analysis by combating robots with other algorithms.

This thesis proposed to train the target robot on the arena with the convolutional neural network supported reinforcement learning algorithm to be developed in RoboCode simulation and train the robot to win over the other robots in the arena. In this context, the images obtained from the arena were created dynamically and the developed robot was trained by these images. The trained model was battled one-on-one with the pre-defined rival robots, and as a result of these battles the scores obtained against the rival robots were observed to be significantly successful.

September 2019, 54 Pages

Key Words: Artificial Intelligence, Reinforcement Learning, Machine Learning, Convolutional Neural Networks, RoboCode, Deep Neural Learning, Q Learning

TEŞEKKÜR

Yüksek lisans eğitimim süresince bu tez çalışmasının hazırlanması ve şekillenmesi birçok kişinin yardım ve destekleriyle gerçekleşmiştir. Öncelikle tez çalışmasının her aşamasında bana destek olup yol gösteren, mesleki bilgilerini ve deneyimlerini öğrencilerinden esirgemeyip önerileri ile katkıda bulunan değerli tez danışmanım Sayın Doç. Dr. Mehmet Serdar GÜZEL (Ankara Üniversitesi Bilgisayar Mühendisliği Anabilim Dalı) hocama, yoğun çalışmalarım sırasında sabır gösterdiği ve bana katlandığı için eşime, sürekli çalışmama izin vermediği için küçük oğluma, çalışma boyunca bilgi birikimleri ile ilham veren, destek olan ASELSAN A.Ş.'de çalışan iş arkadaşlarıma sonsuz teşekkürlerimi sunarım.

Son olarak, beni her konuda destekleyen, yetiştirip bu günlere gelmemi sağlayan, üzerimdeki emeklerini hiçbir zaman ödeyemeyeceğim kıymetli aileme sonsuz teşekkürlerimi sunarım.

Hakan KAYAKÖKÜ
Ankara, Eylül 2019

İÇİNDEKİLER

TEZ ONAY SAFASI

ETİK.....	i
ÖZET.....	ii
ABSTRACT	iii
TEŞEKKÜR	iv
KISALTMALAR DİZİNİ	vi
ŞEKİLLER DİZİNİ	vii
1. GİRİŞ	1
2. PEKİŞTİRMELİ ÖĞRENME	7
2.1 Markov Karar Süreci	10
2.2 Q öğrenme.....	12
2.3 Derin Q-öğrenme.....	13
3. ROBOCODE	15
3.1 Oynanış	16
3.2 Oyun Dinamikleri	17
3.3 Savaş Stratejileri	21
3.3.1 Radar	22
3.3.2 Hedef Planlama	22
3.3.3 Hareket.....	23
3.3.4 Hedefleme ve Atış	25
4. LİTERATÜR.....	26
5. METODOLOJİ	31
5.1 Model Mimarisi	32
6. DENEYLER	40
7. SONUÇ.....	49
8. ÖNERİLER	51
KAYNAKLAR	52
ÖZGEÇMİŞ.....	54

KISALTMALAR DİZİNİ

TD	Temporal Difference
PÖ	Pekiştirmeli Öğrenme
YZ	Yapay Zeka
YSA	Yapay Sinir Ağları
ESA	Evrişimsel Sinir Ağları
RC	RoboCode



ŞEKİLLER DİZİNİ

Şekil 1.1 Mnih ve diğerlerinin (2013) makale kapsamında kullandıkları atari 2600 oyunlarından bazıları. (soldan sağa) Pong, Breakout, Space Invaders, Seaquest, Beam Rider.....	6
Şekil 2.1 Bir Pekiştirmeli Öğrenme Modeline Ait Ajan Çevre İlişkisi.....	7
Şekil 2.2 Markov karar süreci grafik modeli	11
Şekil 2.3 Derin Q-öğrenme ve Q-öğrenme arasındaki karşılaştırma	14
Şekil 3.1 RoboCode savaş simülatörüne ait arenadan bir ekran görüntüsü	16
Şekil 3.2 RoboCode simülatöründeki bir robotun anatomisi	17
Şekil 3.3 RoboCode motoru tarafından kullanılan işlem döngüsüne ait akış diyagramı	19
Şekil 5.1 Ajan ve sunucu arasındaki ilişkiyi gösterir akış şeması.....	35
Şekil 5.2 RoboCode arenasına ait bir ekran görüntüsü (solda) ve bu görüntüye ait sinir ağı girdisi (sağda)	36
Şekil 5.3 Evrişimsel derin Q ağı model	39
Şekil 6.1 SittingDuck isimli robotla yapılan mücadeleye ait ajanı öğrenme performansı.....	41
Şekil 6.2 SittingDuck isimli robotla savaşta model eğitimi boyunca seçilen eylemlerin yüzdesi.....	42
Şekil 6.3 Eğitim 1'inci, 2'nci, 3'üncü ve 4'üncü çeyreklerindeki seçilen eylemlerin yüzdesi	42
Şekil 6.4 Corners robotu ile yapılan savaşlar sonucunda elde edilen RoboCode skorları.	43
Şekil 6.5 Corners robotu için uygulanan epsilon stratejisi, başlangıçta lineer azalma devam etmekte ve sonrasında her 100 iterasyonda epsilon değeri güncellenmektedir.	44
Şekil 6.6 Corners robotu ile yapılan savaşlar sonucunda elde edilen yeni RoboCode skorları.	45
Şekil 6.7 Eğitim 1'inci, 2'nci, 3'üncü ve 4'üncü çeyreklerindeki seçilen eylemlerin yüzdesi	45
Şekil 6.8 Ajanın farklı ön tanımlı Robotlara karşı göstermiş olduğu performans	47
Şekil 6.9 Eğitilen ajanın Bl4ck, Net ve QBot isimli robotlara karşı almış olduğu skorlar	48
Şekil 6.10 Makine öğrenmesi tabanlı robotların ön tanımlı robotlar ile mücadelesiyle elde edilen skorlar.....	48

1. GİRİŞ

Teknolojik gelişmelerin hızla ilerlediği günümüzde, bilgisayar bilimlerinin yanı sıra çok farklı disiplinler tarafından da ele alınmaya başlanan, oldukça popülerleşen ve üzerinde sıkça durulan konulardan en önemlisi hiç şüphesiz yapay zeka konularıdır. Son yıllarda çeşitli yapay zeka uygulamaları, farklı akademik alanlarda kendine yer bulabilmiş, oldukça olumlu sonuçlar alınmış ve alınmaya devam etmektedir. Bu kavram özellikle son yıllarda, sadece bilgisayar bilimleri konusuna ait bir kavram olmaktan çıkmış; matematik, istatistik, felsefe, biyoloji, psikoloji gibi birçok bilim dalıyla karşılıklı etkileşim içerisinde bulunan disiplinler arası bir bilim haline gelmiştir. Örneğin; yapay zeka biliminin alt konularından biri olan yapay sinir ağları, insan beyninde bulunan nöronların bir modellemesi niteliğinde olup bu konunun geliştirilmesinde biyoloji biliminden büyük ölçüde faydalanılmıştır. Benzer şekilde yine yapay zekanın alt konularından makine öğrenmesi algoritmalarının çoğu istatistik ve psikoloji biliminde yer alan teoremlerin bilgisayar bilimine uyarlanmasıdır (Hastie vd. 2009).

Yapay zeka konusu bilgisayar bilimlerinde yaklaşık yüz yıllık bir geçmişe sahiptir. Kelime itibarıyla ilk kullanımı John McCarthy (1956) tarafından olmuştur. Bu tarihten itibaren özellikle yapay sinir ağları birçok problemin çözümünde sıklıkla kullanılmıştır. Ancak zaman içerisinde ortaya atılan karmaşık problemlerin çözümü için gereken donanım yetersizliği 2000'li yılların başına kadar yapay zeka problemlerinin adeta kıs uykusuna yatmasına sebep olmuştur. Bu süreçte geliştirilen problemlere ait çözümler teori aşamasında kalmış donanım yetersizliğinden uygulama alanı bulamamıştır. Bu sebeple geliştirilen algoritmaların doğruluğu sorgulanmış ve bir süre yapay zeka konularına duyulan ilgi azalmıştır. Teknolojideki gelişime paralel olarak son yıllarda yeterli donanıma sahip cihazların da varlığı ve bu cihazlara ulaşım kolaylığı ile birlikte yapay zeka çalışmaları altın çağını yaşamaktadır. Bu süreçte yapay zeka konusu için çeşitli tanımlar ortaya atılmıştır. Biraz önce bahsi geçen John McCarthy tarafında yazılan makalede, yapay zeka “akıllı makineler üretme bilimi ve mühendisliği” olarak tanımlanmıştır. McCarthy’nin yapmış olduğu bu tanım, makalenin yazıldığı tarih düşünüldüğünde bilimsel mecrada kabul görülmüş bir tanım halini almış olsa da, tarih içerisinde geçerliliğini yitirmiş yeterli görülmemiş ve yapay zeka biliminin gelişmesi ile

daha kapsayıcı ifadeler kullanılmaya çalışılmıştır. Yapay zeka, günümüze kadar çeşitli kavram kargaşalarına uğraşmış, tam olarak hangi birimlerin yapay zeka birimi olduğuna kesin karar verilmemiştir. Son yıllarda yapılan gelişmeler düşünüldüğünde yapay zeka, insanmış gibi makineler üretme hedefinin çok ötesinde olup insan düşünce ve davranışlarını simule eden, çıkarımlarda bulunan, var olan problemi geçmiş tecrübelerinden yararlanarak çözebilen ve gelecekte karşılaşılabileceği problemler için sürekli geliştirmelerde bulunan sistemler haline gelmiştir. Bu bağlamda yapay zeka, günümüzde insan zeka ve davranış süreçlerinin makinelerle, özellikle bilgisayar sistemleriyle simülasyonu olarak tanımlanmaktadır. Bu sisteme sahip birimlere ise yapay zeka birimi veya yapay zeka ajanı (agent) denmektedir. Tez boyunca bu tanıma uygun olarak, ajan kavramı yapay zeka birimi anlamı ile çokça kullanılmaktadır.

Yapay zeka konusunu, tarihi süreçteki gelişimi, yükselişi ve günümüzde geldiği nokta düşünüldüğünde tek başına ele almak diğer tüm disiplinler arası bilimlerde olduğu gibi yanlış bir önermedir. Günümüzde bu konu makine öğrenmesi, derin öğrenme gibi çeşitli alt konularda incelenmeye başlanmıştır. Bu kavramlar yapay zekayı oluşturan kapsayıcı terimler olmuştur ve bu konular da son zamanlarda oldukça popüler hale gelmektedir. Makine öğrenmesi konusu, yapay zeka birimine açıkça programlanmadan otomatik olarak deneyimlerden öğrenme, diğer bir deyişle makinenin, sağlanan verilerle mantıklı ve rasyonel sonuçlar türetmesini sağlayan algoritmalar içermekte olup yapay zeka biliminin bir alt konusudur. Derin öğrenme ise yapay sinir ağları denilen ve beynin yapısından ve işlevinden ilham alan algoritmalarla ilgili bir makine öğrenmesi alt sınıfıdır. Ek olarak derin öğrenme yapay sinir ağları kavramının geliştirilmiş bir modeli niteliğindedir de denilebilir.

Bu tez kapsamında yapay zeka biliminin alt konularından biri olan makine öğrenmesi dalına ait çalışmalar kullanılmıştır. Makine öğrenmesi, son zamanlarda oldukça yaygınlaşır hale gelmiş bir yapay zeka disiplini olup büyük veri analizi, otonom arabalar ve sezgisel yapay zeka gibi birçok alanda kullanılmaktadır. Bahsi geçen öğrenme şeklinin insan davranışlarını ve öğrenme şeklini en iyi simule eden öğrenim şekli olduğu düşünülmektedir. Makine öğrenmesi kavramı gözetimli (supervised) öğrenme,

gözetimsiz (unsupervised) öğrenme ve pekiştirmeli (reinforcement) öğrenme olarak üç ana grupta incelenmektedir. Bu üç gruba ait tanımlar aşağıda belirtilmiştir:

Gözetimli Öğrenme: Gözetimli öğrenme metodunda elde bulunan verilerin (input) hangi girdi değerinde hangi çıktıyı (output) vereceği etiketlenmektedir. Bu kapsamda eğitime tabi tutulan ağ, gösterilen girdiler için uygun çıktılar üretebilmek adına ağırlıklarını değiştirmektedir. Ağın tahmin edilen çıktıları ile gerçekte beklenen çıktılar arasındaki hata payı hesaplanarak ağın ağırlıkları güncellenmektedir.

Gözetimsiz Öğrenme: Gözetimsiz öğrenme metodunda öğrenme sürecinde girdilere karşılık gelen çıktı değeri belirtilmemektedir ve ajanın girilen verilere en yakınsak sınıfı bulması istenmektedir. Bu girdi değerlerinin çıktı değerleri ağa verilmemektedir. Bu yöntemde, gösterilen girdi bilgilerine göre ağ sunulan örneklerin kendi içerisinde sınıflandırmasını yapacak şekilde bir kural belirlemektedir.

Pekiştirmeli Öğrenme: Pekiştirmeli öğrenme bu iki yaklaşımın dışında üçüncü bir yaklaşım olarak yerini almaktadır. Bu öğrenmede ise ajana birtakım geri bildirimler verilmektedir ancak gözetimli öğrenmede olduğu gibi çıktıyı denetleyen etiketler yoktur. Ağın eğitim sürecindeki her tekrarında elde edilen çıktının iyi bir tahmin olup olmadığı konusunda bilgi sunulmaktadır. Ağa gelen bu bilgiler doğrultusunda ağ yeniden düzenlenmektedir. Bu özellik sayesinde, ağ hem eğitimini tamamlar hem de çıktı üretir.

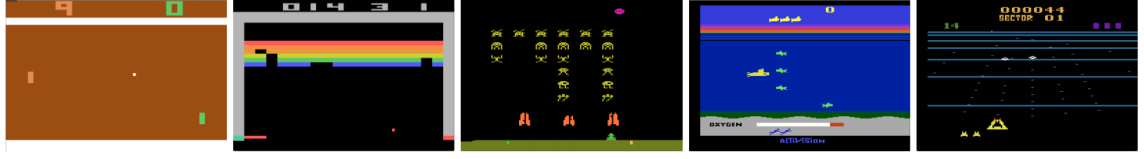
Makine öğrenmesi konusuna ait araştırmalar incelendiğinde, son zamanlarda akademik alanda özellikle pekiştirmeli öğrenme (reinforcement learning) tekniklerine ilgi duyulmaktadır (Kaelbling vd. 1996, Sutton vd. 1998). Pekiştirmeli öğrenme, yapay zeka birimini (ajan) bir dizi ödüllendirme/cezalandırma sistemi kullanarak kendi kendine düşünebilmesini sağlayan bir öğrenme disiplini. Bu yöntemde ajana hangi eylemlerin gerçekleştirileceği söylenmez, bunun yerine ajanın hangi eylemlerin en çok ödülü verdiğini denemesi gerekmektedir. Bilgisayar bilimleri literatürüne psikoloji biliminden kazandırılmıştır. Psikoloji dalındaki çalışmalarla ünlenen Skinner'e (1938) göre davranışlar sonuçlardan etkilenir. Bu bağlamda, pekiştirmeli öğrenme, davranışların

sonuçlarını kontrol ederek davranışı şekillendirme sürecidir. Ceza ve ödül ile davranışları değiştirmeyi önerir. Buna göre ödüller, istenilen davranışı güçlendirmek için kullanılır ve cezalar istenilmeyen davranışları önlemek için kullanılır. Bu teori bilgisayar bilimlerinde de bahsi geçen prensiplere uygun olarak kullanılmıştır. Her hareket sonunda bir ceza ya da ödül verilerek bir ajanın belli bir harekete ya da hareket dizisine koşullanabileceği fikrinden esinlenilmiştir.

Pekiştirmeli öğrenme teknikleri içerisinde özel bir formasyonu olan Q-öğrenme, (Watkins, 1989, Dayan vd. 1992) başarılı bir modeldir ve Markov karar süreçleri teorisinde sağlam bir temele sahiptir. Bu öğrenme metodu ajan ile çevre arasındaki etkileşimi, durumlar, eylemler ve ödüller açısından tanımlamak için Markov karar süreçlerinin resmi çerçevesini kullanır. Bu çerçeve yapay zeka probleminin temel özelliklerini temsil etmenin basit bir yoludur. (Sutton vd. 2017) Ayrıca uygulanması oldukça kolay olup robotik kontrol, bilgisayarla görü, yüz tanıma gibi geniş bir kullanım alanı bulmuştur. Son zamanlarda özellikle video oyunlarında yaygın olarak kullanılmakta olup oyunu oynayarak model geliştirmek için gerekli verilerin toplanmasını sağlar (Mnih vd. 2013). Bu algoritmadaki temel amaç, bir ajanın tanımlı olduğu çevrede uygulayacağı hareketleri inceleyip ajanın bu hareketlere göre kazanacağı ödülü görmek ve bu ödülü maksimize edip maksimize edilen ödüle uygun olarak hareket etmektir. Ajanın tanımlı olduğu çevrede kazanacağı ödüller, gitmesinin istendiği veya istenmediği yerler kullanıcı tarafında daha önceden belirlenir ve bu değerler bir ödül tablosuna yazılır. Ajanın ileriye dönük tecrübeleri ve uygulayacağı hamleler bu ödül tablosuna göre belirlenir. Ajan kullanıcının belirlemiş olduğu ödüle giderken uyguladığı her hamleye dair edindiği tecrübeleri, uygulayacağı hamleleri seçmek için kullanır. Bu tecrübeleri ise Q-Tablosu (Q-Table) adı verilen bir tabloda tutar. Bahsi geçen q tablosu ilk durumda ajanın çevre hakkında herhangi bir tecrübesi olmadığından sıfıra atanmaktadır. Bu durum ajana tanımlı olduğu çevrede bir ödül bulana kadar rastgele hareket edebilmesini sağlamaktadır. Bulunan ilk ödül ile birlikte ajan ödüle gelmeden önceki konumunu ve bu yerin değerini Q tablosuna yazmaktadır. Böylece ajan her iterasyonda ileriye dönük yapılan hamlelere ait değerleri maksimize ederek tahmin etmeye ve ödüle ulaşmaya başlamaktadır.

Bütün bunlar değerlendirildiğinde Q öğrenme tabanlı yapay zeka modelleri son yıllarda oldukça başarılı işler çıkartmışlardır. Öğrenme modeli oldukça güçlü ve öğrenmede kararlı bir yapı sergilemesine rağmen Q-öğrenme algoritması, durum ve eylemlerin sayısı çok arttığında; bu durum ve eylemler için oluşturulacak Q-tablosunun fazla büyümesine yol açacağından tabloya erişim ve tablonun güncellenmesinde sorunlar yaşatmaktadır. Örneğin; 10 000 durum ve 1 000 eylemli bir çevre için ajanın uygulayacağı her eylem için erişeceği Q-tablosunun boyutu 10 milyon hücreden oluşması gerekmektedir ki bu büyüklükte bir tablo hem hafıza alanı hem de işlem süresi açısından sorunlara yol açmaktadır. Bu sorunun çözümü için Q-tablosu ile elde edilen değerleri yaklaşık olarak belirleyebilecek bir yapay sinir ağı kullanılması öngörülmüştür. Böylece derin öğrenme ile Q-öğrenme algoritmalarının birleştirilmesi ile derin Q-öğrenme kavramı ortaya çıkmıştır (Mnih 2015). Derin Q-öğrenme, ajanın belirlenen çevrede uygulayacağı hamleyi kestirebilmesi için gerekli olan Q değerini, yaklaşık olarak belirlemek için derin yapay sinir ağları kullanan Q-öğrenme algoritmasıdır. Bahsedilen ağ ile Q değeri tahmini için gözetimli (supervised) öğrenme kullanılmaktadır. Derin Q-öğrenme bu sayede, daha istikrarlı bir eğitim ortamında pekiştirmeli öğrenme yöntemlerini çözmek için Q değerinin öğrenilmesini sağlar.

Son yıllarda derin Q-öğrenme konseptinin özellikle video oyunlarında kullanıldığı ve başarılı sonuçlar alındığı görülmüştür. Özellikle Mnih ve diğerlerinin (2013) yazmış olduğu makale bu konuya öncü niteliğindedir. Bahsi geçen makale kapsamında 7 adet popüler Atari 2600 oyunu derin pekiştirmeli öğrenme modeli kullanılarak evrişimli sinir ağları (Convolutional Neural network) vasıtasıyla atari oyunlarından gelen görseller ile eğitilmiş ve 7 oyunu da oldukça yüksek skorlar alarak oynayabilen ajanlar elde edilmiştir. **(Şekil 1.1)** Elde edilen sonuçların tatmin edici düzeyde olması ve pekiştirmeli öğrenme ile evrişimli sinir ağlarının beraber kullanılması açısından makale devrim niteliğinde olmuştur. Makalede bahsedilen konsept, sonralarında görüntü işleme, bilgisayarla görme, robotik gibi çeşitli alanlarda da kullanılmıştır.



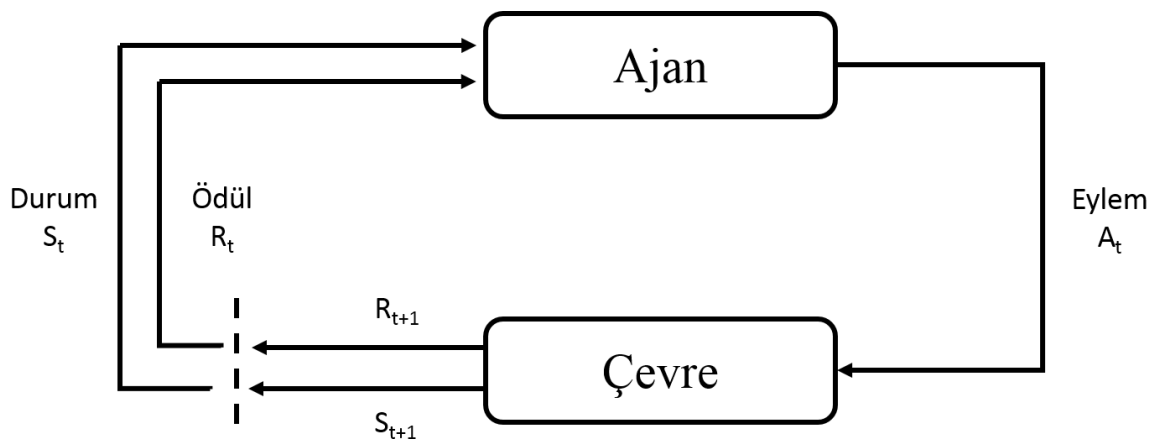
Şekil 1.1 Mnih ve diğerlerinin (2013) makale kapsamında kullandıkları atari 2600 oyunlarından bazıları. (soldan sağa) Pong, Breakout, Space Invaders, Seaquest, Beam Rider

Bu tez kapsamında bahsedilen ve son yıllarda başarılı olduğu kanıtlanan derin evrişimli Q öğrenme algoritmasının (deep convolutional Q-learning) farklı alanlara uygulanabilirliği üzerine çalışmalar yapılmıştır. Bu kapsamda RoboCode isimli simülasyon üzerinde derin evrişimli q öğrenme algoritması uygulanarak modelin performansı test edilmiştir. RoboCode IBM tarafından 2001 yılında Java programlama dili tabanında geliştirilen, açık kaynak kodlu bir savaş simülatörü programıdır. Bu simülatörde amaç, iki boyutlu ortamda platform tarafından kullanıcılara sunulan sınıfları kullanarak bir savaş robotu programlamak ve bu robotları, yine bu simülatör tarafından sağlanan ve arena ismi verilen ortamda birbirleriyle mücadele ettirerek kullanıcının programladığı robotun performansını ölçmektir. Her bir robot *gövde*, *namlu* ve *radar* isimleri verilen parçalardan oluşmakta olup simülasyonda belirli ara yüzlerle kullanıcıların bu parçaları kontrol etmesi sağlanmıştır. Bahsedilen robotun anatomisi ve hareket kabiliyetleri düşünüldüğünde simülasyonun durum ve eylem sayısı Atari 2600 oyunlarından daha fazla ve komplekstir. Bu bakımdan çalışma boyunca durum ve eylem sayısını optimize edecek kurallar belirlenmiştir. Belirlenen durum ve eylemler ile birlikte ödül ve ceza formülleri de uygulanarak evrişimli sinir ağları oluşturulmuş ve simülasyonun ekran görüntüleri kullanılarak ağ eğitilmiştir. Eğitilen ağ ile birlikte oluşan q değerleri ile robotların sonraki hareketleri uygulanarak simülasyondan en yüksek puan alması hedeflenmiştir. Eğitilen robotlar, RoboCode uygulamasında önceden tanımlı olan robotlar ile arenada savaştırılmış ve skor ölçümleri yapılarak model test edilmiştir.

Bu tez, temel olarak iki kısımdan oluşmaktadır. İlk kısımda pekiştirmeli öğrenme, Q-öğrenme, evrişimli sinirsel ağlar ve RoboCode hakkında birtakım araştırmalar ve bilgilendirmeler mevcuttur. İkinci kısım ise derin pekiştirmeli q öğrenme tabanlı robotun implementasyonunu ve oluşturulan modele ait test sonuçlarını içermektedir.

2. PEKİŞTİRMELİ ÖĞRENME

Pekiştirmeli Öğrenme, sayısal olarak ödüllendirme ile belirlenen çevrede denemeyanılma etkileşimi yoluyla bir görevi yerine getirmeyi öğrenen ajanlar olarak tanımlanmaktadır (Sutton vd. 2017). Bu yaklaşım, bir ajanın ilk etapta çevre ile alakalı herhangi bir bilgisi olmamasına karşın ödüllendirmeler yoluyla çevre ile etkileşime girip belirlenen sorunları çözmek anlamına gelmektedir. Ajan, çevrede uygulamış olduğu her eylem sonrasında çevreden bir ödül alır. Bu ödül genellikle bir sayı olarak gösterilir. Ajanın her eyleme karşılık bir ödül alması ajan ve tanımlı çevrenin öğrenme süreci boyunca ayrık (discrete) bir zaman içerisinde sürekli olarak birbirleri ile etkileşim halinde olmasını gerektirmektedir. Bu etkileşime dair tanıtıcı çizim şekil 2.1’de gösterilmektedir. Şekil 2.1’de ajanın sahip olduğu durumlar kümesi S ve ajanın uygulayacağı eylem kümesi A ile ifade edilmektedir. Bu kapsamda ajan ve çevre, $t = 0, 1, 2, 3, \dots$ ile ifade edilen bir dizi kesikli zaman durumu dizisinde etkileşime girer. Bu zaman dizisine ait her t anında ajan $S_t \in S$ ile ifade edilen bir durum setine sahiptir ve bu durum seti temelinde $A_t \in A(s)$ eylemini seçer. Bir zaman adımı sonra, gerçekleştirdiği eylemin bir sonucu olarak, ajan, sayısal bir ödül olan $R_{t+1} \in R$ ’yi alır ve kendisini yeni bir S_{t+1} durumunda bulur. Bu durum, ajanın çevreden aldığı ödül sinyalleri maksimize olana kadar devam eder.



Şekil 2.1 Bir Pekiştirmeli Öğrenme Modeline Ait Ajan Çevre İlişkisi

Ajan ve çevre elemanlarından başka, bir pekiştirmeli öğrenme sisteminin dört ana alt ögesi tanımlanabilir: politika (policy), ödül sinyali (reward signal), değer işlevi (value function) ve opsiyonel olarak tanımlı çevrenin modeli (Sutton vd. 2017). Politika, ajanın belirli bir zamandaki davranış şeklini tanımlamaktadır. Farklı bir deyişle, politika tanımlı çevreye ait durumlardan yapılacak eylemlere ait bir karardır. Belirli bir durum seti içerisinde ajanın o an içinde bulunduğu durumda olduğu göz önüne alındığında eylemlerin gerçekleşme olasılığını ifade eder. Matematiksel modelde π ile gösterilir ve deterministik çevrelerde eylem (a) üzerine etkisi (2.1), stokastik çevrelerdeki etkisi (2.2) ile ifade edilir:

$$a = \pi(s) \quad (2-1)$$

$$\pi(a|s) = P[A_t = a \mid S_t = s] \quad (2-2)$$

Politika, pekiştirmeli öğrenme algoritmalarının çekirdeği konumunda olup ajanın tanımlı çevredeki davranışını belirlemenin yoludur.

Pekiştirmeli öğrenme sisteminin bir diğer alt ögesi olan ödül sinyali, pekiştirmeli öğrenme problemlerindeki hedefi tanımlar. Ajan kendisine verilen bu ödül sinyalleri sayesinde kullanıcının isteklerini algılar. Her bir zaman diliminde ajana ödül olarak adlandırılan bir sinyal gönderilir. Burada sinyalden kastedilen rasyonel bir sayı değeridir. Ajanın kendisine tanımlanan çevredeki tek amacı uzun vadede eylemlerin sonucunda kazandığı toplam ödülleri en üst seviyeye çıkarmaktır. Esasen ödül sinyali ajan için iyi ve kötü olayların ne olduklarını tanımlamaktadır. Kullanıcı açısından ödül sinyali ajanın belirlemiş olduğu politikayı değiştirmenin tek yoludur. Ajan, politika tarafından seçmiş olduğu bir eylemin düşük ödül alması durumunda seçimlerini daha yüksek ödül alabileceği başka bir politikaya yakınsayabilir. Mevcut durum göz önüne alındığında, gerçekleştirilen eylem ile ilgili bir sonraki ödülü öngören ödül işlevi R formül (2.3) ile ifade edilmiştir.

$$R_s^a = E [R_{t+1} \mid S_t = s, A_t = a] \quad (2-3)$$

Pekiştirmeli öğrenme sisteminin bir diğer alt ögesi değer işlevidir (value function). Ödül sinyali ajanın anlık olarak neyin iyi olduğunu söylemekte iken değer işlevi uzun vadede neyin iyi olduğunu belirtir. Diğer bir deyişle, ajanın belirlenen çevrede kendisine uygun eylemi uygulayabilmesi için, mevcut durumdan başlayarak gelecek boyunca biriktirmeyi bekleyeceği toplam ödül tutarına ihtiyacı olmasına rağmen ödül sinyali anlık değere sahiptir. Örneğin bir ajan belirlediği politika üzerinde sırasıyla yüksek ödül almış olmasına rağmen anlık olarak düşük ödüllü bir duruma sahip olabilir. Böyle bir durumda ajan hala yüksek değere sahip olabilir. Bu sebeple sadece ödül sinyaline bağımlı kalmak ajanın yanlış seçim yapmasına yol açmaktadır. Bu açıdan bakıldığında ajan için en yüksek değeri elde edebileceği fakat o an için en yüksek ödülü olmayan durumları ortaya çıkaran eylemlerin varlığından söz edilebilir, çünkü bu eylemlerin uzun vadede ajan için en büyük ödülü alma ihtimali vardır. Bir pekiştirmeli öğrenme modelinde ödülleri temel olarak doğrudan çevre tarafından verilir, ancak bir ajanın tüm ömrü boyunca yaptığı gözlemler dizisinden değer işlevi yeniden ve yeniden tahmin edilmelidir. Hemen hemen tüm pekiştirmeli öğrenme algoritmalarının en önemli bileşeni, verimli bir şekilde değer işlevi tahmininde bulunan metotlardır. Kısacası değer işlevi tanımlı durumların iyiliğini / kötülüğünü değerlendirmek için kullanılan gelecekteki ödüllerin bir tahminidir, bu nedenle ajanın farklı eylemler arasında seçim yapmasını sağlar. Değer işlevine ait matematiksel model (2.4) ile gösterilmiştir.

$$V_{\pi}(s) = E_{\pi}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | S_t = s] \quad (2-4)$$

Formül (2-3) ile ifade edilen γ değeri indirim faktörü olup $[0,1]$ değer aralığında tanımlıdır. Geleceğe ait ödüllendirmelerin ne kadar önemli olması gerektiğini belirten katsayıdır. Eğer ($\gamma = 0$) olursa bu ajanın kısa görüşlü olduğu manasına gelir ve yalnızca ilk ödül önemser. Eğer ($\gamma = 1$) olursa bu ajanın gelecekteki tüm ödülleri önemsendiği anlamına gelmektedir. Bu değer pekiştirmeli öğrenme modellerinde bir hiper-parametre olup değeri deneme yanılma yolu ile problem spesifik olarak tespit edilir.

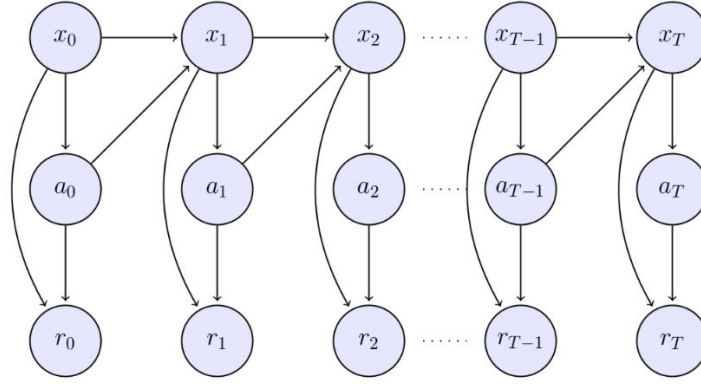
Bazı pekiştirmeli öğrenme sistemlerinin dördüncü ve son elemanı ise çevre modelidir. Model, tanımlanan çevrenin davranışını taklit eden veya daha genel olarak çevrenin nasıl davranacağına dair çıkarımlar yapılmasını sağlayan birimdir. Örneğin; model,

ajana bir durum ve eylem verildiğinde, sonuçta ortaya çıkan bir sonraki durumu ve bir sonraki ödülü tahmin edebilir. Modeller ajanın yapacağı hamleleri planlama için kullanılır, bununla birlikte yaşanmadan önce gelecekteki muhtemel durumları göz önünde bulundurarak ajanın herhangi bir hareket tarzına karar vermektedir. Özetle bir model, kullanıcıya şu anki durum ve ajan tarafından yapılan işlem göz önüne alındığında, bir sonraki olası hedefler üzerindeki olasılık dağılımını açıklamaktadır.

2.1 Markov Karar Süreci

Tez kapsamında bahsi geçtiği üzere, pekiştirmeli öğrenme algoritmalarındaki temel amaç ajanların eylemleri (action) için biriken ödülü en üst düzeye çıkaracak bir politika (policy) bulmaktır. Bu amaca yönelik olarak pekiştirmeli öğrenme algoritmalarının temelinde Markov Karar Süreci matematiksel modeli kullanılmaktadır. Markov karar süreçleri, eylemlerin sadece o anlık ödülleri değil, müteakip durumlar için gelecekteki ödüller yoluyla etkiledikleri sıralı karar vermenin klasik bir formalizasyonudur. Aynı zamanda markov karar süreçleri, kesin teorik açıklamalar yapılabilecek pekiştirmeli öğrenme probleminin matematiksel olarak idealleştirilmiş bir şeklidir.

Pekiştirmeli öğrenme modellerinde ajan ile tanımlı çevrenin sürekli iletişim halinde olduğu ve ajanın bu çevrede bir eylem seçimi yaptığını buna karşılık tanımlı çevrenin bir ödül döndüğü belirtilmiştir. Daha spesifik olarak; ajan ve çevre, bir dizi kesikli zaman durumu dizisinde etkileşime girer. Bu zaman dizisine ait her t anında bir durum setine sahiptir. Bir zaman adımı sonra, gerçekleştirdiği eylemin bir sonucu olarak, ajan, sayısal bir ödül alır ve çevreye ait bulunan durumunu günceller. Bu durum ajanın çevreden aldığı ödül sinyalleri maksimize olana kadar devam eder. Markov karar süreci, bu bilgiler ışığında aşağıda belirtilen formül ve şekil 2.2 ile belirtilen çizelge ile bir dizi zaman yörüngesine sahiptir.



Şekil 2.2 Markov karar süreci grafik modeli

Bir Markov karar sürecinde, durum, eylemler ve ödüllerin (S, A ve R) tümü sınırlı sayıda öğeye sahiptir. Bu durumda bir t anında R_t ve S_t rasgele değişkenleri yalnızca önceki duruma ve eyleme bağlı olan ayrık olasılık dağılımlarına sahiptir. Diğer bir deyişle t anındaki bu değişkenleri değerleri için, bir önceki durumun ve eylemin değerlerine bağımlılık söz konusudur ve bu durum göz önüne alındığı takdirde, t zamanında meydana gelen değerlerin gerçekleşme olasılığından söz edilebilir. Bir durumun bir önceki duruma bağlı olma olasılığını ifade eden matematiksel formül (2.5) ile gösterilmiştir.

$$p(s', r | s, \alpha) = Pr\{S_t = s', R_t = r | S_{t-1} = s, A_{t-1} = \alpha\} \quad (2-5)$$

Bu durumda, p olasılığının bütün durum ve ödüller için değeri aşağıdaki formül ile gösterilmiştir.

$$\sum_{s' \in S} \sum_{r \in R} p(s', r | s, \alpha) = 1, \text{ tüm } s \in S \text{ ve } \alpha \in A(s) \text{ için}$$

Bunun yanı sıra Markov karar süreçleri için tanımlanan T zaman aralıklarının sonsuza gittiği düşünüldüğünde, ajanın sahip olacağı toplam ödül değeri ıraksamaya başlayacaktır. Toplam ödül değerinin ıraksamaması için $0 < \gamma < 1$ değer aralığında bir indirim faktörü tanımlanmaktadır. Böylece ajanın sahip olacağı ödüller de bu indirim

faktörüne göre üssel olarak azaltılmaktadır. Bu durum aşağıdaki formül (2-6) ile modellenmiştir.

$$\pi^* = \operatorname{argmax} \langle \sum_{t=0}^T \gamma^t r_t \rangle \quad (2-6)$$

Markov karar süreci yapısı soyut ve esnektir ve birçok farklı soruna farklı şekillerde uygulanabilir. Bu yapı, birtakım etkileşimler (ödül) ile hedefe yönelik öğrenme probleminin çözüme ulaştırıldığı kayda değer bir modeldir. Pekiştirmeli öğrenme uygulamalarının temeli olan hedef odaklı davranış öğrenme probleminin, bir ajan ve çevresi arasında ileri geri geçen üç sinyale indirgenebileceğini öne sürmektedir: ajan tarafından yapılan seçimleri temsil eden bir sinyal (eylemler), seçimlerin yapıldığı temeli temsil eden bir sinyal (durumlar) ve aracının hedefini tanımlamak için sinyal (ödülleri). Bu yapı, tüm karar öğrenme problemlerini faydalı bir şekilde göstermek için yeterli olmayabilir, ancak yaygın olarak faydalı ve uygulanabilir olduğu kanıtlanmıştır.

2.2 Q öğrenme

Q-öğrenme, modelsiz ve değerler tabanlı bir pekiştirici öğrenme algoritmasıdır. Bu öğrenme modelinin amacı bir ajana hangi şartlarda hangi işlemi yapması gerektiğini söyleyen bir politika öğrenmektir. Çevrenin bir modelini gerektirmez (dolayısıyla “modelsiz” denilmektedir) ve uyarlama gerektirmeden Stokastik geçişler ve ödülleri ile ilgili sorunları çözebilir. Herhangi bir sonlu Markov karar süreci için, Q-öğrenme, mevcut durumdan başlayarak, tüm ardışık adımların toplam ödülünün beklenen değerini en üst düzeye çıkarması anlamında optimal olan bir politika bulur. Q-öğrenme, sınırsız keşif süresi ve kısmen rastgele bir politika göz önüne alındığında, herhangi bir markov karar süreci için en uygun eylem seçim politikasını belirleyebilmektedir. Q-öğrenme algoritmasında kullanılan q-fonksiyonu Bellman eşitliğini kullanır ve durum (s) ve eylem (a) olmak üzere iki girdi almaktadır. Q-fonksiyonu formül (2.7) ile gösterilmiştir. Formülde ifade edilen eşitlik t anındaki bir durum ve eylem için indirim faktörü uygulanmış, beklenen toplam ödülü ifade etmektedir.

$$Q^\pi(s_t, a_t) = E[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | S_t, a_t] \quad (2.7)$$

Formül (2.6) ile elde edilen q-değerleri, q-tablosu denilen durum-eylem boyutlu bir tabloda tutularak ajanın uyguladığı her harekete karşılık aldığı ödül değerine göre ilgili tablo değerleri güncellenmektedir (formül (2.8)). Ajan, belirli durumda yapacağı eylemin hangisi olacağına q tablosundaki değerlere bakarak en optimal değere göre karar vermektedir.

$$yeni\ Q(s, a) = Q(s, a) + \sigma[R(s, a) + \gamma \max_{a'} Q'(s', a') - Q(s, a)] \quad (2.8)$$

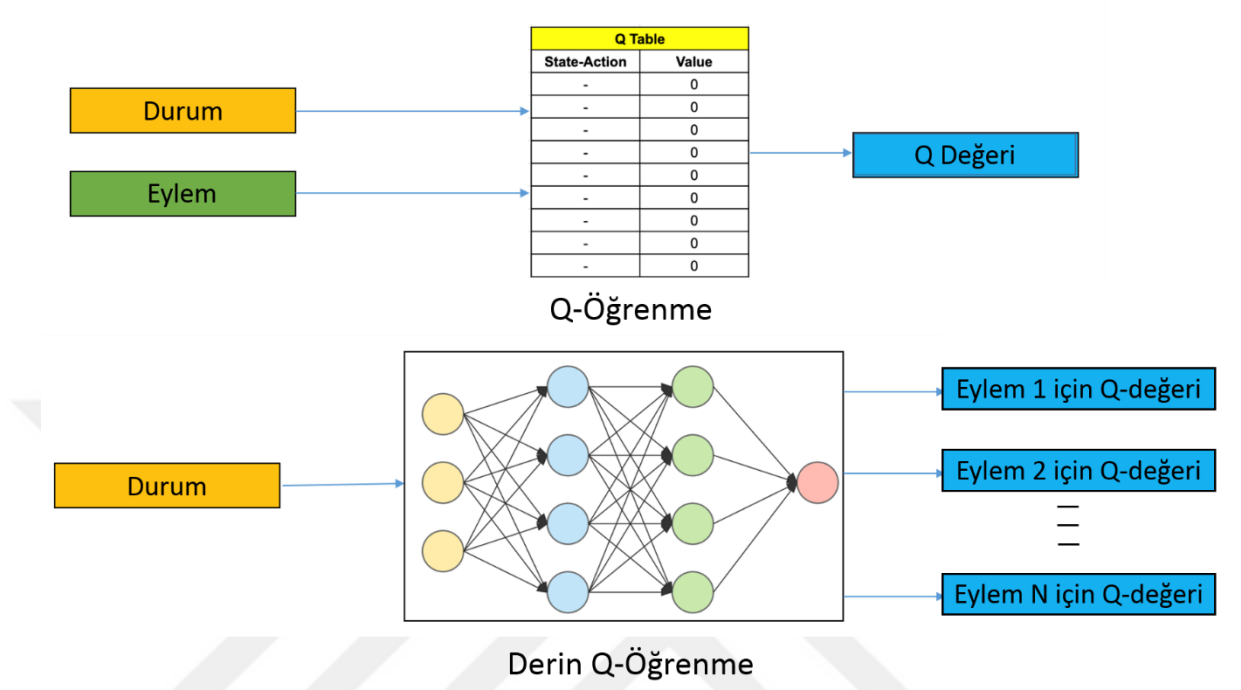
Burada σ ile belirtilen değer öğrenme katsayısı olup yeni edinilen bilgilerin ne kadar eski bilgileri geçersiz kıldığını belirleyen bir hiperparametredir. $Q(s, a)$ değeri o anki q-değerini ifade etmektedir. γ , indirim faktörüdür. $R(s, a)$ ifadesi belirtilen durumda bu eylemi gerçekleştirmenin ödülünü ifade eder. $\max_{a'} Q'(s', a')$ ifadesi o anki durum için bütün olası eylemler kümesinde, her bir eyleme ait kazanılan ödüller içerisindeki en yüksek ödül değerini tanımlar.

2.3 Derin Q-öğrenme

Q-öğrenme algoritması ile ajan elde ettiği q-tablosu ile çevrede tanımlı durumlar için en büyük ödülü alabileceği eylemi bulabilmektedir. Bu basit ama oldukça güçlü bir algoritma olmasına rağmen, her durum ve eylem için *durum x eylem* boyutlu bir q-tablosu oluşturulması küçük veri setlerinde bir problem oluşturmaya da çok sayıdaki durum ve eylem için hafıza ve işlem süresi problemi meydana getirmektedir. Bu sorun q değerlerinin hesaplanıp q-tablosunda saklanması yerine, q değerlerinin derin sinir ağları gibi makine öğrenmesi modellerini kullanarak elde edilmesi fikri ile çözüme ulaşılmıştır.

Derin Q-öğrenmede, q-değer fonksiyonunu yaklaşık olarak belirlemek için derin sinir ağı modeli kullanılmıştır. Bu ağa, durum girdi olarak verilmekte ve olası tüm işlemlerin Q-değeri çıktı olarak oluşturulmaktadır. Elde edilen Q değerleri arasından en yüksek veriler seçilerek ajanın öğrenmesi sağlanmaktadır. Q-öğrenme ve derin Q-öğrenme algoritmaları arasındaki karşılaştırma Şekil 2.3 ile gösterilmiştir. Derin Q-öğrenme ile

Q-öğrenme algoritmasına ait dezavantajlardan arındırılmış, oldukça güçlü ve başarılı bir öğrenme modeli elde edilmiştir.



Şekil 2.3 Derin Q-öğrenme ve Q-öğrenme arasındaki karşılaştırma

3. ROBOCODE

Tez kapsamında geliştirilen pekiştirmeli öğrenme tabanlı model için RoboCode savaş simülatörü kullanılmıştır.

RoboCode, Java programlama dili tabanında geliştirilen, IBM AlphaWorks tarafından 2001 yılında ilk sürümü çıkartılan, belirlenen bir savaş arenasında düşman robotlara karşı mücadele edebilmek için yapılması gerekenin bir robot savaş tankını kodlamak olduğu bir savaş simülatörüdür. Bu simülatörde oyuncu, robotun programcısıdır ve kurallar gereği simülatörde robotun üzerinde doğrudan bir etkiye sahip değildir. Bunun yerine, oyuncu, robotun yapay zeka kabiliyetlerini diğer bir deyişle, savaş alanında meydana gelen robotun olaylara nasıl davranacağını ve nasıl tepki vereceğini kodlayarak arena ismi verilen tanımlı çevrede geliştirdiği modeli diğer robotlara karşı çalıştırmaktadır. Arenada gerçekleştirilen savaşlar gerçek zamanlıdır ve sonuçları ekranda anlık olarak yayınlanmaktadır.

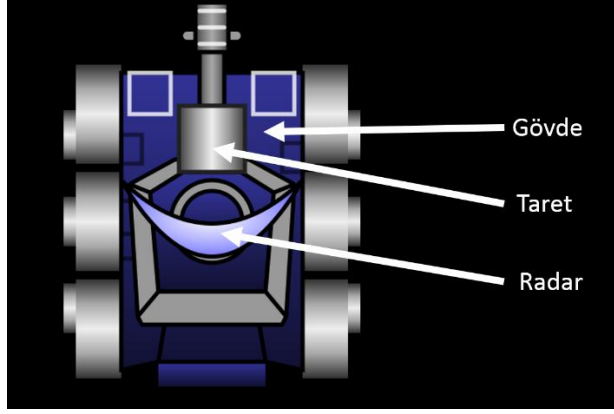
RoboCode simülatörüne, günümüzde 2001'deki ilk sürümden bu yana, çok fazla geliştirme yapılmış ve bugün robot tasarım fikirlerinin ve kaynak kodlarının paylaşıldığı büyük topluluklar meydan gelmiştir. Bu topluluklarda turnuvalar ve devam eden ligler düzenlenmekte ve robot sıralamalarına sahip web siteleri sürekli güncellenmektedir. Son zamanlarda bu uygulama üniversitelerde yapay zeka konularına ait çalışmalar için de kullanılmaktadır (Morten vd 2017). Simülatördeki robotların arena üzerindeki hamlelerini Şekil 3.1'deki gibi gerçek zamanlı olarak izlemek mümkündür. Alternatif olarak, hesaplama süresini azaltmak için savaşları arabirimsiz çalıştırmak da mümkündür.



Şekil 3.1 RoboCode savaş simülatörüne ait arenadan bir ekran görüntüsü

3.1 Oynanış

RoboCode robot simülatöründeki bir robot, Şekil 3.2’de gösterildiği gibi gövde taret ve radar olmak üzere üç ana parçadan oluşmaktadır. Robotların eklendiği çevreye arena denmektedir. Eklenen robotlar arenada son robot kalana kadar birbirleriyle mücadele etmektedirler. Bir savaş birkaç turdan oluşmaktadır ve tur sonunda savaşan robota simülatör tarafından bir puan verilir. Oyunun galibi olan robot savaş sonunda en çok puanı alan robottur. Arenadaki robotlar başlangıçta 100 enerji seviyesine sahiptir ve enerji seviyesi sıfırın altına düştüğünde ölürler. Robotlar gövdeye monte edilmiş taretler sayesinde birbirlerine mermi atışları ile savaşabilirler. Robotlar 1200 piksel uzaklığa kadar diğer robotları tarayabilen bir radar ile donatılmıştır. Herhangi bir robot, radarı ile tarama alanına giren diğer bir robot hakkında, uzaklık, rota, hız, isim ve enerji hakkında bilgi alabilir.



Şekil 3.2 RoboCode simülâtöründeki bir robotun anatomisi

Radar, taret ve robot gövdesi birbirinden bağımsız olarak 360 derece dönebilmektedir. İstenirse taret aracın rotasyonlarını veya radar, taret ve araç rotasyonlarını takip etmek için birbirlerine kilitlenebilir. Taretin sahip olduğu ateş gücü değişken olabilir. Bu sayede taret, oyuncu tarafından ayarlanan ateş gücüne bağlı olarak hedef robotlara farklı değerlerde hasarlar verebilmektedir. Simülasyona ateş gücünün arttırılmasının bir dezavantajı olarak taretin ısınma özelliği de eklenmiştir. Yüksek ateş gücüne sahip mermiler robotun taretinin daha fazla ısınmasına sebep olmaktadır ve bu ısınma robotun bir süre atış yapamaması manasına gelmektedir. Bunun yanı sıra düşük güçlü mermiler daha hızlı hareket ederlerken yüksek güçlü mermiler daha yavaş hareket etmektedirler. Tüm savaşlar, değişken büyüklükte, dikdörtgen, duvarlarla çevrili bir arenada gerçekleşir. Varsayılan boyutu 800×600 pikseldir. Robotların boyutu 36×45 pikseldir. Bir robot, radarı çevirirken aynı anda sabit hızlanma ve değişken hızda (ileri ve geri) arenada hareket edebilir. RoboCode simülâtöründe pozitif hız ileri bir harekettir ve negatif hız geriye doğru harekettir. Hız değerinin negatifliği yön belirtmekte olur hızın büyüklüğü değerin mutlak değeridir. Bir robotun maksimum hızı her tur için 8 pikseldir.

3.2 Oyun Dinamikleri

RoboCode, robotların bazı olası eylemlerini sınırlayan bir fizik modeline de sahiptir. Örneğin robot dönerken tam hızda hareket edemez. Silahı ateşlemek, bir robotun atış gücüyle aynı miktarda enerjiyi kaybetmesine neden olur. Eğer mermi bir hedefi vurursa kaybedilen enerji mermi gücünün 3 katı ile geri kazanılır. RoboCode tek bir iş parçacığı

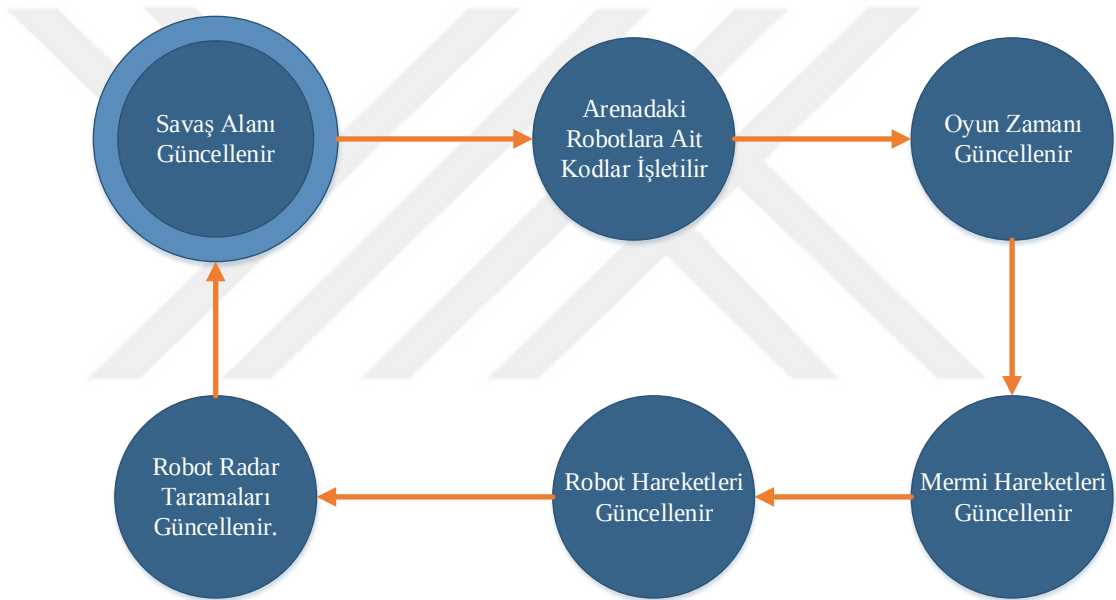
(thread) üzerinde çalıştırıldığından robotların çalışması sıraya dayalıdır. Her sırada robotun hamlelerini gerçekleştirebilmesi için sınırlı bir süre verilir. Tüm robotlar sırasını tamamladığında eylemlerin gerçekleştirilmesi işlemi tekrar eder. Bu, bir robotun uygulayacağı eylemlerinin karmaşıklığı ile geçen zaman arasında bir uzlaşma olması gerektiği anlamına gelir. Bu sebeple, aşırı karmaşık hesaplamalara sahip olan bir robotu, kendisine ayrılan zaman dilimini yapacağı eyleme ait hesaplamalar ile geçirmesi, RoboCode tarafından hamlenin ihmal edilmesine sebebiyet vereceğinden robot arenada kaybedecektir.

RoboCode, bir maçın kazananını belirlemek için bir puanlama sistemi kullanır. Bir robotun kazanan olabilmesi için bütün turlarda hayatta kalması şart değildir, çünkü robotlar birtakım hücum eylemleri sebebi ile kazananın daha fazla puan kazanabilmektedirler. RoboCode simülatörüne dair birtakım kurallar aşağıda belirtilmiştir:

- Arena ölen her robot için bütün canlı robotlara 50 puan eklenir.
- Hayatta kalan son robota her ölü robot için 10 puanlık ekstra ödül verilir.
- Verilen her hasan puanı 1 puan ile ödüllendirilir.
- Hedef robot bir düşmanı öldürürse, düşman robota vermiş olduğu toplam hasarın %20'si kadar miktar puan ile ödüllendirilir.
- Bir rakibe çarpma, çarpmadan kaynaklanan hasar başına 2 puan ile ödüllendirilir.
- Rakibe çarparak bir rakibi öldürmek, o robotun verdiği toplam hasarın %30'u kadar miktar puan ile ödüllendirilir.
- Oyunda son kazanan tüm turlar oynandıktan sonra en çok puanın sahibi olan robottur.

RoboCode oyunu tek bir iş parçasığı üzerinde çalıştırıldığından her tur için oyunda bir dizi işlemler sırasıyla gerçekleştirilir. Her turu yönetebilmek için RoboCode motoru tarafından kullanılan işlem döngüsü aşağıda belirtilmiştir. İşlem döngüsüne ait akış diyagramı ise Şekil 3.3 ile gösterilmiştir:

- Tüm robotların kodları RoboCode'un öngördüğü maksimum işlem zamanı içerisinde ise işlem bitene kadar çalıştırılır.
- Oyun zamanı güncellenir.
- Tüm mermilerin hareketleri kontrol edilir ve konum güncellemeleri yapılır.
- Mermi çarpışmaları kontrol edilir.
- Tüm robotların hareketleri kontrol edilir ve rota, hız, mesafe bilgileri ışığında konum güncellemeleri yapılır.
- Tüm robotların radar taramaları yapılır ve radar mesajları ilgili robotlara iletilir.
- Savaş alanı yeniden çizilir.



Şekil 3.3 RoboCode motoru tarafından kullanılan işlem döngüsüne ait akış diyagramı

RoboCode, her robotun kendi başına bir olay kuyruğuna (örneğin bir mermiye çarptığında veya başka bir robotu tararken) sahip olduğu olaya dayalı bir sisteme dayanmaktadır. Bu olayları gerçekleştiren robotlar *Robot* veya *AdvancedRobot* sınıfı genişletilerek oluşturulur. *AdvancedRobot* Java sınıfını temel alan robotlar asenkrondur; bu, engelleme yapmayan yöntemler kullanarak aynı anda eylemler gerçekleştirmenin mümkün olduğu anlamına gelir. *AdvancedRobot*, temel olarak aşağıdaki olayları alır:

BulletHitEvent: Robotun mermilerinden biri başka bir robota çarptığında olay aktifleşir. Merminin çarptığı rakibin adını, sahibini, gücünü, hızını, merminin hala aktif olup olmadığını belirten bir durum bayrağını tutan bir *Bullet* nesnesiyle birlikte rakibin adı ve kalan enerjisi hakkında bilgi verir.

BulletHitBulletEvent: Robota ait mermilerinden biri başka bir mermiye çarptığında ve imha edildiğinde olay aktifleşir. Fonksiyon iki kurşun nesnesini de getirebilir.

BulletMissedEvent: Robotun mermilerinden biri bir duvara çarptığında olay aktifleşir

HitByBulletEvent: Bir mermi tarafından vurulduktan sonra olay aktifleşir ve atıcının ismi, merminin doğrultusu, gücü ve hızı gibi bilgileri bulundurur.

HitRobotEvent: Başka bir robotla çarpıştığında olay aktifleşir ve diğer robotun adını, enerjisini, yönünü ve robotun diğer robota doğru gidip gitmediği bilgisini verir.

HitWallEvent: Robot bir duvarla çarpıştığında alınır. Robotun duvara olan açısını verir.

ScannedRobotEvent: Radar arenadaki herhangi bir robotu taradığında olay aktifleşir. Diğer robotun adını, enerjisini, yönünü, mesafesini, doğrultusunu ve hızını verir.

WinEvent: Robot arenada oynanan tur kazandığında olay aktifleşir. Bu fonksiyon herhangi bir bilgi dönmez. Genellikle savaşa ait bilgilerin kaydedilmesi işlemlerin kullanılır.

DeathEvent: Robot arenada oynanan tur esnasında öldüğünde olay aktifleşir. Bu fonksiyon herhangi bir bilgi dönmez. Genellikle savaşa ait bilgilerin kaydedilmesi işlemlerin kullanılır.

SkippedTurnEvent: Robot, o tur için belirli bir süre içerisinde bir eyleme karar vermemişse olay aktifleşir. Fonksiyona herhangi bir bilgi dönüşü yoktur.

RobotDeathEvent: Başka bir robot öldüğünde olay aktifleşir. Ölen robotun adını bilgisine ulaşılabilir.

3.3 Savaş Stratejileri

Savaşlarda, stratejiler belirli bir durumda nasıl hareket edileceğine dair kılavuz ilkelerdir. RoboCode savaş simülatörüne ait her yıl uygulanmakta olan uluslararası turnuvalar ile birlikte arenada uygulanacak etkili stratejiler büyük önem taşımaktadır.

Bu sebeple tez kapsamında problemin çözümünde fayda sağlayacağından RoboCode solüsyonu ile ilgili bir dizi strateji araştırmalarına yer verilmiştir.

İlk olarak robot arenasında 1-1 savaşlar ve çoklu robota ait savaşa ait farklı stratejiler benimsenmelidir. Bir robot birebir savaşta iyi sonuç verebilir, ancak yakın dövüşler için başka stratejiler gerekir. RoboCode savaş arenasında yakın dövüş savaşında iyi bir stratejinin, savaş alanından uzak durmaya ve hiç dikkat çekmemeye yönelik olması genel kanıdır. (Kobayashi vd. 2013) Bu metot ile, diğer robotların birbirlerini öldürmesi beklenir. Sonrasında, kalan robotları aldığı muhtemel hasarlardan dolayı öldürmek kolaylaşmaktadır. Rakip tam enerjiye sahip olduğunda, bu strateji birebir savaşlarda eşit derecede iyi çalışmamaktadır. Bu stratejiye karşı diğer bir argüman RoboCode 'da kullanılan puanlama sistemidir. RoboCode simülatörünün prensipleri gereği savaş, sadece ayakta kalan son kişi olmaya dayanılarak kazanılmamaktadır. Simülatörde ek olarak, verilen hasarın ve robotların öldürdüğü robot miktarı parametrelerinin de oldukça büyük etkisi olmaktadır.

Aşağıda, bir robotu kontrol etmek için çeşitli stratejiler, görünen her avantaj ve dezavantaj ile birlikte sunulmuştur:

3.3.1 Radar

En basit yol olarak rakip robotları taramanın temel yolu, radarı sürekli olarak 360 derece döndürmek ve diğer tüm rakiplerin konumlarını kaydetmektir. Bu yola alternatif olarak bir diğer yön tarama stratejisi, robotun bir yönde taradığı ve diğer tüm robotlar bulunduğu, tarama yönünün tersine çevrildiği bir stratejidir. Fikir, robotların küçük alanlarda savaşmaya meyilli olduğu ve bunun rakipleri bulmak için harcanan zamanı azaltmak için kullanılabileceği yönündedir.

Birebir savaşlarda robot taraması için farklı stratejiler kullanılmaktadır. Özellikle bire bir savaşlarda, radarın tek bir hedefte sabit kaldığı kilitli bir tarama stratejisi çok etkilidir. Bu, hedefin kaybedilmemesi için radarı her turda biraz ileri geri hareket ettirerek yapılabilmektedir. Bahsedilen strateji çoklu robotlu arenalar düşünüldüğünde iyi bir strateji olmayacaktır, çünkü bu savaşlarda enerjisi daha düşük olan robotlara hamle yapma daha avantajlı olmaktadır. Bu sebeple radarı bir robota kilitlemek yakında bulunan düşük güçlü robotun göz ardı edilmesine yol açacaktır.

RoboCode yapısında başka bir robotun bir mermi fırlattığını gösteren herhangi bir olay (event) olmadığından, radar bunu tahmin etmek için kullanılabilmektedir. Bir rakip robotun iki tarama arasında belirli miktarda enerji kaybetmesi eyleminden çıkarımla, rakibin bir atış yapmış olduğu belirlenebilmektedir. Rakip robotun yapmış olduğu bu atıştaki ateş gücü, kaybedilen enerjinin miktarına göre de tahmin edilebilmektedir. Elbette, atış yapmak enerji kaybının tek nedeni olmayabilir fakat bu kestirim, eğitilmiş bir tahminde bulunmanın etkili bir yoludur.

3.3.2 Hedef Planlama

Hedefleri elimine etmeden galip gelme ihtimali olsa da RoboCode'daki öncelikli hedef, arenadaki rakiplere saldırarak ve onları öldürerek kazanmak olduğundan, arenadaki bir robot, eğer saldırmak isterse hangi rakibe saldıracağına karar vermelidir. Bu konuda kullanılan en temel yaklaşım, basitçe arenadaki en düşük enerjili robota saldırmak veya

tüm rakipler robottan daha yüksek enerjiye sahipse saklanmaya çalışmaktır. Fakat bununla birlikte, bahsi geçen stratejiye göre tayin edilen hedef robot arenanın diğer ucunda olabilmesi durumunda, robotun hedefi vurmak için oldukça uzağa hareket etmesi gerekmektedir. Bu durum belirlenen hedef üzerinde yol alırken, seçili olan hedefi başka bir rakibin öldürmesine yol açabilmektedir. Ayrıca, yine en düşük enerjiye sahip olan uzak hedefe doğru ilerlerken başka bir robot bu süreç içerisinde daha da zayıflama ihtimali vardır. Bu durumda makul olan eylem robotun düşük enerjili yeni bir hedef seçmesi ve ona doğru hareket etmeye başlaması olmalıdır. Bahsi geçen stratejiye sahip robot bu sebeplerden ötürü arenada öldürülene kadar stratejiyi tekrar edebilir ve hiçbir zaman tek bir mermi ateşleyemeyebilir.

Başka bir yaklaşım olarak robotun hedef robot olarak kendisine en yakın hedefi seçmesi gösterilebilir. Fakat bu yaklaşımın da birtakım dezavantajları olmaktadır. Örneğin, en yakın hedef gerçekten savaşta en iyi performans gösteren robot olabilir ve bu sebeple seçilecek en iyi hedef olmayabilir.

Üçüncü bir yaklaşım bahsedilen iki yaklaşıma hibrit bir model niteliğinde olup, hedef mesafe ile hedef enerji arasında bir dengeden söz edilmektedir. Ancak, doğru oranı bulmak karmaşık bir problemidir ve kararın verilmesi gereken duruma bağlıdır. RoboCode, taramanın aksine, robot sınıflarının hiçbirinde planlamada yardımcı olacak bir yöntem sunmadığından bu stratejiler için kullanıcının birtakım matematiksel modeller geliştirmesi gerekmektedir.

3.3.3 Hareket

Robotların sahip olduğu hareket kabiliyetini etkili kullanmak, arenadaki saldırılardan korunmak ve hedeflenen robotlara saldırılarda bulunmak için büyük önem taşımaktadır.

İlk olarak, arenada bulunan herhangi bir robotun aynı yerde hareket etmeden kalması, saldırı anında mermilerden kaçamaması gibi durumlardan dolayı kötü bir fikirdir.

Robotu arenada düz çizgilerle hareket ettirmek de kötü bir stratejidir; çünkü böyle bir robotun rakibinin rotasını tahmin etmekte zorlanmayacak ve böylece gelecekteki pozisyonları hedeflemesi kolay olacaktır. Bu şekilde hareket eden robotların rakiplerinden çok fazla vuruş alması oldukça yüksek ihtimallidir.

Arenada bahsi geçen düz çizgi üzerine hareketten farklı olarak dairesel hareket, düz çizgiler halinde hareket etmekten daha avantajlıdır; fakat sürekli konumları bir hızda verilen bir çember hareket stratejisinin, rakip robotlar tarafından fark edilip hesaplanması zor değildir. Bu nedenle bu strateji arenadaki gelişmiş robotlara karşı dezavantajlı olmaktadır.

Rakip robotların, kullanılan hareket desenlerini tahmin etmeye çalışmasını zorlaştırmak amacı ile rastgele hareketler kullanılması bir diğer hareket stratejisidir. Fakat bu kapsamda tamamen rastgele hareket etmenin faydasından çok zararı olmaktadır, çünkü bu diğer robotları takip etmeyi zorlaştırmakta ve robotun duvarlara çarpma riskini arttırmaktadır.

Anti-yerçekimi birçok robotta uygulanan ve basit ve etkili bir çözüm olarak gösterilen genel bir hareket stratejisidir. Robotu birçok robotun olduğu tehlikeli bölgelerden uzaklaştırmak ve güvenli veya diğer avantajlı alanlara çekmek için kullanılır. Mermilerin ve diğer robotların olduğu bölgelerde negatif çekim kuvveti varmış gibi davranılır ve bu bütün çekim kuvvetlerinin bileşkesi alınarak sıcak temasın olduğu bölgeden kaçmak amaçlanır. Hareketin gerekli olduğu her durumda kullanılabilir ve algoritmalar yoluyla savaşlarda dinamik olarak kaçınılması gereken alanlara karar verilmesi mümkündür.

RoboCode hareket stratejilerinden bir diğeri toslama tekniğidir. Toslama, hedef robotun rakip robotun bulunduğu konuma gidip robota çarparak yüksek mermi gücü ile ateş etme eylemine denilmektedir. Özellikle bire bir savaşta toslamak rakibinden daha fazla enerjiye sahip bir robot için iyi bir atış desteği sağlayabilir, çünkü böyle bir düelloda arenada ilk olarak ölecek robot rakip robottur.

3.3.4 Hedefleme ve Atış

Hedeflemenin görevi, silah taretinin döndürme açısını hesaplamak ve böylece bir düşman robotunu vurmak için doğru yöne işaret etmektir. Hedefleme yaparken meydana gelen başlıca komplikasyonlardan biri, Arenada atılan bir merminin hedefe ulaşabilmesi için bir mesafe kat etmesini gerektirdiğinden, silah taretini sadece düşmanın yönüne doğrultmanın yeterli olmamasıdır. Bu durum merminin uçuş süresi boyunca, düşmanın orijinal konumundan uzaklaşması ihtimalini hesaplamayı gerektirmektedir. Bu konudaki diğer bir komplikasyon, kurşunların atış gücüne göre farklı hızlara sahip olmasıdır.

Basit bir strateji rakipleri taramak ve bir rakip bulunduktan hemen sonra tam güçle rakibin yönünde atış yapmaktır. Fakat bu teknik hareket eden robotlarla iyi çalışmamaktadır.

Bir diğer teknik, yapılacak atışların en azından bir kısmının rakibe çarpacağı ihtimalinden hareketle robotun düşük mermi gücü ile her yöne ateş ettiği, basit bir stratejidir.

Daha gelişmiş bir strateji ise, hedefin gelecekteki konumlarını radar taramasından döndürülen bilgilerden hareketle, robot doğrultusu ve hız değerlerinden tahmin etmeye çalışmaktır. Bu strateji rakip robotun hareket desenlerini takip etmeyi de beraberinde getirmektedir. Rakip düz çizgiler halinde hareket ederse, nasıl ateş edileceğini belirlemek oldukça kolaydır. Ek olarak rakip bu yöntem ile arena duvarlarını takip etme, köşelerde gizleme veya dairesel hareket etme gibi belirli bir davranış sergileyen robotların gelecekteki konumları tahmin edilebilir. Desen eşleştirme, hangi stratejilerin kullanılacağına karar vermede yardımcı olabilecek rakipler tarafından kullanılan mevcut stratejiyi belirlemek için de kullanılabilir.

4. LİTERATÜR

Yapay zeka ile ilgili gelişmeler daha eski tarihlere dayanmasına rağmen makine öğrenmesi ve özellikle tezin ana konusunu oluşturan pekiştirmeli öğrenme tabanlı araştırmaların oldukça yeni olduğu gözlemlenmektedir. Bu alanındaki ilk projelerden birisi Arthur Samuel'in (1959) dama oyunu için modellediği ajanlardır. Arthur Samuel, çalışmasında kendi kendisiyle dama oynayan bir ajana yer vermiştir. Bu çalışmanın sonunda ajanın dama hakkında makale sahibinden daha fazla şey öğrendiği belirtirmiştir.

Pekiştirmeli öğrenmenin belki de en başarılı uygulaması, tamamen pekiştirmeli öğrenme ve kendi kendine oynama ile öğrenilen ve insanüstü bir oyun seviyesine ulaşan bir tavla oyun programı olan TD-gammon'dur (Tesauro 1995). TD-gammon ajanında, Q-öğrenmeye benzer bir modelsiz takviye öğrenme algoritması kullanılmış ve gizli katmanı olan çok katmanlı bir algılayıcı kullanarak değer işlevine (value function) yaklaşmıştır.

TD-gammon'da uygulanan yöntemin başarılı olmasının ardından aynı yöntem satranç, Go ve dama da dahil olmak üzere diğer bilinen oyunlara da uygulanmış ancak bu ajanlar tavla oyunundan daha az başarılı olmuşlardır. Bu durum, tavla oyunundaki zarların stokastik oluşunun ajan tarafından durum uzayının keşfedilmesine yardımcı olduğunu düşündürmüştür ve TD-gammon yaklaşımının sadece tavlada çalışan özel bir durum olduğuna dair yaygın bir inanca yol açmıştır. (Pollack vd. 1996)

Son zamanlarda derin öğrenmeyi pekiştirmeli öğrenme ile birleştirmeye yönelik bir ilgi canlanmıştır. Öne sürülen bu hibrit modelde tanımlı çevreyi tahmin edebilmek için derin sinir ağları mimarisi ve değer işlevini (Sallans vd 2004) veya ilgili politikayı (Hess vd 2012) tahmin edebilmek için ise kısıtlı Boltzmann makineleri kullanılmıştır.

Özellikle robotik ve sensor teknolojilerindeki gelişmelerle birlikte ajanları doğrudan görme ve konuşma gibi yüksek boyutlu duyuşal girdilerden kontrol etmeyi öğrenmek,

pekiştirmeli öğrenmenin uzun zamandır karşılaştığı zorluklardan biridir. Bu alanlarda çalışan en başarılı pekiştirmeli öğrenme uygulamaları, doğrusal değer işlevleri veya politika gösterimleriyle birlikte el yapımı özelliklere dayanmaktadır. Günümüzde derin öğrenmedeki son ilerlemeler ile birlikte ham duyuşal verilerden yüksek düzeyde özelliklerin çıkarılması mümkün kılınarak, bilgisayarla görü (Krizhevsky vd. 2012), (Sermanet vd. 2013), (Mnih vd. 2013) ve konuşma tanımada (Dahl vd. 2012), (Graves vd. 2013) büyük atılımlara yol açmaktadır. Bahsi geçen çalışmalarda kullanılan yöntemlerde evrişimli ağlar, çok katmanlı algılayıcılar, sınırlı Boltzmann makineleri ve tekrarlayan sinir ağları dahil olmak üzere bir dizi sinir ağı mimarisi kullanılmakta ve hem denetimli hem de denetimsiz öğrenmeden yararlanılmaktadır.

Belirtilen çalışmalarla birlikte derin öğrenme ve pekiştirme öğrenmedeki gelişmeler, doğrudan ham girdilerden (örneğin, görüntülerden) yapılacak eylemler dağılımına eşleyen uçtan uca politikaları öğrenmeyi mümkün kılmıştır. Bu konudaki öncü niteliğindeki çalışma evrişimsel sinir ağları modelini pekiştirmeli öğrenme algoritmaları ile birlikte kullanılarak olmuştur (Mnih vd. 2013). Evrişimsel sinir ağları alt yapısı ilk kez Fukushima tarafından önerildi (1988). Ancak, ağı eğitmek için gerekli olan donanımların o yıllardaki yetersizliği nedeniyle yaygın olarak kullanılmadı. Sonrasında LeCun ve diğerleri (1998) ESA'na gradyan tabanlı bir öğrenme algoritması uygulamış ve el yazısı basamaklı sınıflandırma problemi için başarılı sonuçlar elde etmiştir. Evrişimsel sinir ağlarının, Derin sinir ağlarına kıyasla, insan görsel işleme sistemine benzer olması, 2D ve 3D görüntüleri işlemek için yüksek oranda optimize edilmiş olması ve 2D özelliklerinin soyutlamasını öğrenmek ve çıkarmak için etkili olması gibi birçok avantajı vardır. Mnih ve diğerleri (2013) bu ağ yapısı ile pekiştirmeli öğrenme modelini beslemiş ve klasik Atari 2600 video oyunlarında insanüstü performans sağlayan politikaların üretilmesi ile alakalı çalışmalar yapmışlardır. Bu çalışma ile görüntü tabanlı derin öğrenme modeli ile pekiştirmeli öğrenme modelinin beraber çalışması ile klasik Atari oyunlarını oynayan ajanların insandan daha başarılı olduğu görülmüştür. Öncü niteliğindeki bu çalışmanın sunmuş olduğu bu başarılı öğrenme modeli, sonrasında Go isimli geleneksel masa oyununda da denenmiş ve oyunu en iyi oynayan oyunculara karşı bilgisayar galip gelmiştir (Silver vd. 2016).

Her ne kadar pekiştirmeli öğrenme tabanlı algoritmalarının genel bir başarı gösterdiği düşünülse de bu alanda olumsuz örnekler mevcuttur. Hung ve diğerlerinin (2017) yaptığı bu çalışmada Atari çalışmasında kullanılan hedef video oyunlarında ajanı olumsuz etkilere sebep oldurabilecek resimler ile karıştırmalar yapılmış, bu sayede oyundaki ajan hipnotize edilmiş ve ajanın gelecekte daha kötü sonuçlar elde etmesinin mümkün olduğu gösterilmiştir.

Bu makale kapsamında yapılan çalışma, bahsedilen Atari çalışmasında (Mnih vd. 2013) kullanılan evrimsel bir sinir ağının, karmaşık pekiştirmeli öğrenme ortamlarındaki ham fotoğraf verilerinden başarılı kontrol politikaları öğrenmek için bu zorlukların üstesinden gelebileceğini göstermektedir. Bu kapsamda Atari oyunlarından farklı olarak RoboCode savaş simülatörü kullanılmış, bu simülatör vasıtası ile savaş verilerinden ham fotoğraf verileri alınarak evrimsel sinir ağı tabanlı pekiştirmeli öğrenme modeli eğitilmiştir. Eğitilen bu model ile robotlar arenada savaşmış ve RoboCode savaş arenasında performans testleri yapılmıştır.

Günümüzde RoboCode savaş simülasyonu, ciddi oyun kategorisinde olup, dünyanın birçok yerinde yapay zeka eğitim aracı olarak kullanılmaktadır. Kullanıcılar tarafından geliştirilen bu robotlar RobotRumble isimli platformda yayınlanmakta, arenalarda savaşırılarak puanlandırılmakta ve en iyiler tablosu oluşturulmaktadır. Bu turnuvalarda en yüksek skorlu robotlar yapay zeka destekli ajanlar olmaktadır. RobotRumble isimli platforma ait robotların birkaçı haricinde büyük bir çoğunluğu akademik olarak belgelenmemiş uygulamalardan oluşmaktadır. Bu alandaki ilk akademik makaleler RoboCode simülatörünün çıkış tarihinden birkaç yıl sonrasına dayanmaktadır. İlk olarak Eisenstein (2003) genetik algoritmalar tabanlı bir robot üzerine çalışmalar yapmış ve REX gibi varsayım ve davranış yönelimli dilleri temel alarak en iyi şekilde savaş robotunun nasıl uygulanabileceğini üzerine araştırmalar yapmıştır.

Sonraki yıllara bakıldığında RoboCode savaş simülatörü için geliştirilen yapay zeka robotlarının çoğunun genetik algoritmalar tabanlı olduğu görülmektedir. Shichel ve diğerleri (2005), evrimsel olarak tasarlanan robotları uluslararası bir RoboCode yarışmasına tanıtmak için ilk girişimde bulunmuşlardır. Amaç, bu genetik evrilmiş

robotların, insan yapımı “zor kodlanmış” robotlara karşı nasıl bir güç alacağını görmekti. Ekstra bir meydan okuma olarak, robotlara ait kodların sadece 4 satır ile sınırlı olduğu bire bir HaikuBot mücadelesini seçmişlerdir. Tasarladıkları robotları geçmiş HaikuBot mücadelelerinin en iyi oyuncularına karşı savaştırmışlar ve bu robot 27 robot içerisinde üçüncü sırada yer almışlardır.

Makine öğrenmesi tabanlı ilk RoboCode tankı Gade ve diğerleri (2003) tarafından RoboCode'un ana özelliklerinin her biri için doğru makine öğrenmesi tipini (yapay zeka algoritması) seçerek modüler bir hibrid ajan mimarisi ile geliştirilmiştir. Amaç, yakın dövüş savaşında iyi performans gösteren bir robot geliştirmektir. Bu hibrit mimari, durum uzayı, tüketim katmanları ve hakem bileşeni olmak üzere üç katmandan oluşmaktadır. Alt katmanların her biri, her modülde bulunan bir modül kümesi kullanarak, RoboCode'un belirli bir kontrol probleminin üstesinden gelmek için tasarlanmıştır. Örnek olarak, Hedefleme ve Hedef Seçimi problemleri iki farklı katmandır. Bu katmanların her biri bir istekte bulunmaktadır. Örneğin “A Robotunu öldür”. Bu dilek, her katman tarafından düzenlendikten sonra, hakem bileşenine sunulmaktadır. Hakem bileşeni ise, RoboCode robot yapısının çözümleyebileceği komuta çevirmektedir.

Aalborg üniversitesinden belgelenen bir başka uygulama Frøjhær ve diğerleri (2004) tarafından yine çeşitli AI yöntemlerini uygulamak için bir başka hibrit hiyerarşik mimarisi kullanmaktadır. Aalbot'tan farklı bir yönüyle Frøkjær ve diğerleri tarafından bir robot ekibi içinde diğer robotlar ile takım halinde çalışabilecek şekilde uygulanmıştır. Robot, her makine öğrenme yöntemini, yöntemin en uygun olduğu sıradüzensel düzeyde kullanmak üzere tasarlanmıştır. Model, Genel Strateji, Strateji Modülleri ve Etkileşim Modülleri olmak üzere üç genel katmana sahiptir. Genel Strateji, saldırgan veya savunmacı bir strateji arasında seçim yapmak için bir Bayesian ağı kullanır. Bunu, Bayesian ağına belirli faktörlerin olasılığını söyleyen Strateji Modüllerinden (Geri Çekilme, Saldırı, Hedef Alma) geri bildirimle yapar. Son olarak, gerçek davranış her parçanın (Radar, Silah, gövde ve takım) etkileşim modülleri tarafından belirlenir.

Sonuç olarak, Yapılan arařtırmalar neticesinde RoboCode simülasyonunda genetik algoritmalar, yapay sinir ağıları ve makine öğrenmesi gibi konularda birtakım çalışmalar ve bu modeller ile birtakım robotlar geliştirilmiş olsa da bu tezin konusu dahilinde olan derin evriřimsel sinir ağıları kullanılarak RoboCode simülatöründen alınan görüntülerin işlenip pekiřtirmeli öğrenme modeli ile beraber kullanıldığı bir çalışmaya ve bu yönde geliştirilmiş bir robota rastlanmamıştır. Bu sebeple Mnih ve diğerkleri (2013) tarafından başarısı kanıtlanan modelin Atari oyunları tanımlı çevresi ek olarak RoboCode gibi durum ve eylem uzayı daha büyük ve kompleks bir çevrede uygulanması bu bakımdan akademik bir yenilik taşımaktadır.



5. METODOLOJİ

Bilgisayarla görü ve konuşma tanıma alanındaki son gelişmeler ile birlikte derin sinir ağlarının çok geniş bir eğitim setinde etkili bir şekilde eğitildiği görülmüştür. En başarılı yaklaşımlar, stokastik geriye gradyan işlemine dayanan hafif güncellemeler kullanılarak doğrudan ham girdilerden eğitilir. Derin sinir ağlarına yeterli veri besleyerek, el yapımı özelliklerden daha iyi sunumlar öğrenmek çoğu zaman mümkündür (Krizhevsky vd 2012). Bütün bunlara ek olarak pekiştirme öğrenme algoritmasını doğrudan RGB görüntüleri üzerinde çalışan derin sinir ağına bağlayarak ve stokastik geriye gradyan güncellemeleri kullanarak eğitim verilerini verimli bir şekilde işleyen ve bu sayede video oyunlarını insanüstü performans ile oynayabilen ajanlar geliştiren çalışmalar da oldukça başarılı olmuştur (Mnih vd. 2013). Bahsi geçen bu çalışmalar ve bu çalışmalarda alınan başarılar, tez kapsamında RoboCode simülasyonuna ait robotların eğitilmesi için simülatöre ait RGB görüntüleri üzerinde çalışan derin sinir ağına bağlı pekiştirici öğrenme modeli kullanma fikrini doğurmuştur.

Tesauro'nun (1995) TD-Gammon ve Mnih'in (2013) Atari mimarileri, tez kapsamındaki yaklaşım için bir başlangıç noktası olmaktadır. Bu mimariler değer işlevini tahmin eden bir ağı parametrelerini, algoritmanın çevre ile olan etkileşimlerini kullanarak güncellemektedirler. Atari mimarisinde TD-Gammon'dan farklı olarak, ajanın geçmiş deneyimlerine ait değerlerin saklanarak ağı eğitildiği deneyim tekrarı konsepti kullanılmıştır (Lin. 1993). Deneyim tekrarı ile algoritmanın iç döngüsü sırasında, Q-öğrenme güncellemeleri veya parçalı (minibatch) eğitim güncellemeleri, saklanan bu deneyim tekrarı örnekler havuzunda rastgele çekilen deneyim örneklerine uygulanarak ağ eğitilmektedir. Eğitim sonrasında ajan epsilon greedy politikasına uygun olarak bir eylem seçmekte ve bu eylemi yürütmektedir. Deneyim tekrarı yönteminin birçok avantajı bulunmaktadır (Sutton vd. 2017). İlk olarak, deneyim tekrarı yönteminin aksine, ardışık girdiler kullanılarak doğrudan öğrenme modelinde, girdiler arasındaki güçlü korelasyonlar nedeniyle çeşitlilik yetersizdir. Girdilerin randomize edilmesi bu korelasyonları kırmakta ve bu nedenle ağ güncellemelerinin varyansını azaltmaktadır. Bu sayede deneyim tekrarı, önceki durumlara ait davranış dağılımlarının birçoğunun ortalamasını almakta ve öğrenmeyi yumuşatarak parametrelerde salınım veya sapmaları

önlemektedir. Bahsi geçen konseptin Atari mimarisinde başarılı sonuç verdiği görülmüştür. Bu sebeple bu tez kapsamında da derin sinir ağı yapısının öğreniminde deneyim tekrarı kullanılmıştır. Bu kapsamda kullanılan algoritma sadece son n sayıda deneyim dizisini hafızada saklar ve güncellemeleri gerçekleştirirken rastgele olarak d' den örnekler alır. Bu yaklaşım bazı açılardan problemlidir, çünkü bellek tamponu önemli geçişleri ayırt edememekte ve bellek boyunun (n) sonlu olması nedeniyle her zaman son geçişlerin üzerine yazılmaktadır.(Mnih vd. 2013).

Tez kapsamında RoboCode simülatörü kullanılarak derin sinir ağıları tabanlı pekiştirmeli öğrenme algoritması uygulanması hedeflenmiştir. Pekiştirmeli öğrenme modellerinden Q-öğrenme tekniği kullanılmış ve derin öğrenme tekniklerinden evrişimli sinir ağıları ile q-tablosu oluşturulmuştur. Evrişimli sinir ağıları eğitimi için RoboCode simülatörü kullanılarak arenadaki robotların ekran görüntüleri ağa girdi olarak verilmiş ve ödül fonksiyonları ile ağın istendik yönde eğitilmesi sağlanmıştır. RoboCode arenası ön tanımlı olarak 800 piksel genişlik ve 600 piksel yükseklik çözünürlüğüne sahiptir. Robotların mücadele ettiği arenanın çözünürlük değerleri için bu ön tanımlı ayarlara sadık kalınmıştır.

5.1 Model Mimarisi

Çalışma kapsamında evrişimli sinir ağıları girdisi için ilk olarak arenaya ait ekran görüntüleri periyodik olarak kaydedilmiş ve bu görüntüler üzerinden işlem yapılmaya karar verilmiştir. RoboCode uygulamasına ait herhangi bir periyodik ekran görüntüsü alma özelliği olmadığından ilk olarak RoboCode programını çalıştırarak ekran görüntülerini periyodik alabilen bir program geliştirilmiştir. Geliştirilen programın sonucundan bahsedilen işlemin birkaç dezavantajı olduğu görülmüştür. İlk olarak ekran görüntüsü alan program ile RoboCode arenasındaki kare hızı (fps) farklılığından dolayı arenaya ait bütün karelerin kaydedilememesi sorunu meydana gelmiştir. İkinci olarak, uygulama grafik arayüzü olmadan çalışabilmektedir ve bu sayede arenadaki savaşlar çok hızlı sonuçlanabilmektedir. Uygulamanın görsel arayüzü kullanılarak ekran görüntüsü alınması ile modelin eğitilmesi, modelin öğrenme süresini ciddi miktarda uzatmaktadır. Bu sebeple RoboCode arenasındaki robotların konum ve doğrultu bilgileri

ile arenada atılan mermilerin konum bilgileri kullanılarak ağı girdi olarak verilecek görüntünün üretilmesini sağlayan bir program yazılmıştır. Bu program ile arenanın gösterimine benzer görüntüler elde edilmiş olup ekran görüntüsü almanın getirmiş olduğu dezavantajlar elimine edilmiştir.

Bir sinir ağı kullanılarak eğitilecek q-öğrenme algoritmasında q değerinin elde edilebilmesi için ağı verilecek girdiler için birkaç olası yol vardır. Tarih ve eylem, Q-değerlerinin skaler tahminlerine eşleştiği için, önceki bazı yaklaşımlar tarafından iki değer birden sinir ağına girdi olarak kullanılmıştır (Riedmiller 2005). Bu tür mimarinin ana dezavantajı, her bir eyleme ait Q değerini hesaplamak için ağı ayrı bir ileri geçiş yapılması gerektiği ve bu işlemin de eylem sayısı ile doğrusal olarak ölçeklenen bir maliyete yol açmasıdır. Bu modelin yerine proje kapsamında, her olası eylem için ayrı bir çıkış biriminin bulunduğu bir mimari kullanılmakta ve sadece ajanın durumu sinir ağına girdi olarak verilmektedir. Böylece ağı ait her bir çıkış değeri, giriş durumu için bireysel eylemin öngörülen q değerlerine karşılık gelir. Bu tür tanımlanan bir mimarinin temel avantajı, belirli bir durumda tüm olası eylemler için q değerinin ağıda yalnızca bir ileri geçişle hesaplayabilme yeteneğidir.

Evrişimsel sinir ağları mimarisine girdi olarak verilecek ekran görüntüleri RoboCode arenasına ait o anki durumu ifade etmekte olup ağı eğitilmesi için kısıtlı bir bilgiye sahiptir. Farklı bir deyişle t anındaki ekran görüntüsü girdisi için robotların ve mermilerin bulunduğu konumlar algılanabilir fakat mermilerden kaçabilmek ve etkili saldırılar yapabilmek için robot ve mermilerin hareket yönleri bilgisine de ihtiyaç vardır. Bu bilginin elde edilebilmesi için t+1 anındaki ekran görüntüsü kullanılabilir. Birden fazla kareyi aynı anda işleyerek tanımlanacak evrişimsel sinir ağına girdi olarak vermek bu kareler arasındaki akış bilgisini de koruyacak ve ağı daha güçlü kestirmelerde bulunmasını sağlayacaktır. Bu sebeple oluşturulan modelde bir t anına ait kare ile birlikte t+1, t+2, t+3 kareleri de kullanılmış ve ağı girdi olarak verilmiştir.

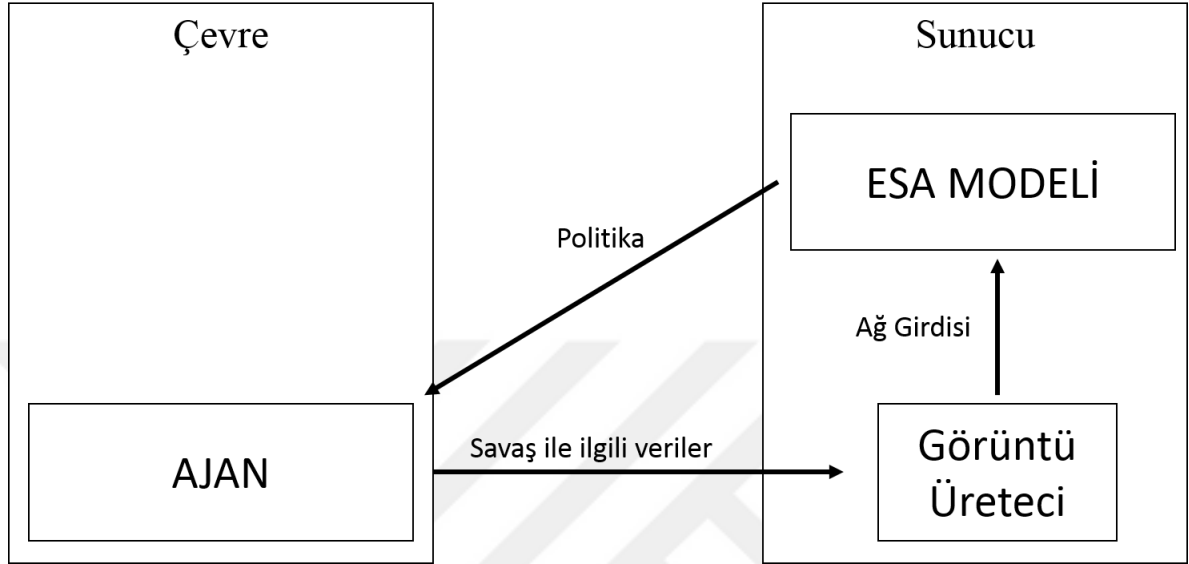
Her ne kadar RoboCode arenada istenilen sayıda robot ile savaşmaya izin verse de, tez kapsamında yapılan araştırmalar ve oluşturulan öğrenme modellerinin geliştirilmesi için sadece bire bir savaşlar tercih edilmiştir. İlk olarak ağı eğitimi için herhangi bir mermi

atışında bulunmayan ve hareket etmeyen SittingDuck isimli önceden tanımlı robot kullanılmıştır. Bu görev basit görünmesine rağmen ajana başarılı bir politikaya yaklaşmayı öğretmek için birkaç temel uygulama detayına sahiptir. İlk olarak bu robotla mücadele ile yumurtlama noktaları her bölümde rastgele olan RoboCode’da, ağın rakip robota yönelmeyi, uygun bir mesafeye girmeyi öğrenmesi amaçlanmaktadır.

RoboCode uygulamasında pekiştirmeli öğrenme ajanı yetiştirme konusundaki zorluklardan birisi de, ajanın oyunlar arasında paylaşılması garanti edilmeyen ağır kum havuzunda (sandbox) çalıştığıdır. Robotlar güvenlik ve robotların oyunda hile yapılmasını önlemek nedeni ile ana bilgisayardaki kaynaklara sınırlı erişime sahiptir ve dışarıdan herhangi bir Java kütüphanesi kullanılamamaktadır. Bu sebeple, robotların anlık konumları, sağlık durumları, kazandıkları puanlar ve ödüller gibi derin sinir ağıları modeli için gerekli bilgileri ana bilgisayardaki herhangi bir konuma raporlayabilmesi mümkün değildir. Bu sorunun çözümü için yapay sinir ağı modelinin eğitiminin ve eğitilen modelin kaydedilmesi işlemlerinin, robot geliştirme sürecinin dışında gerçekleştirilmesi sonucuna varılmıştır.

Çalışma kapsamında bahsi geçen sorunun çözümü için, http veri protokolü kullanılarak, savaş esnasında ajandan edinilen deneyim örneklerini sürekli dinleyen, bunları bellekte saklayan, öğrenim modelini robottan gelen girdilere göre güncelleyen, robotun kendisine gönderdiği durum bilgilerini kullanarak optimal eylemin ne olduğuna karar vererek robota geri bildirimde bulunan, periyodik olarak ağıdan gelen bilgileri bir dosyaya kaydeden işlem dışı bir Python programı geliştirilmiştir. Bu program ile ağın eğitilmesi ve en uygun eylemin uygulanabilmesi için robot ile gerekli olan iletişim http sunucusu üzerinden gerçekleşmektedir. Ajan, oyuna başladıktan sonra sunucu ile arasında bir UDP bağlantısı açmakta ve modeli besleyecek girdileri sunucuya göndermektedir. Ajan aynı zamanda arenada savaşın başlamasından itibaren her durum için modelin verdiği eylem kararını sunucudan sorgulayarak dönen cevabı uygulamaktadır. RoboCode simülasyonuna ait kısıtlamaların nasıl üstesinden geldiğine dair çalışma Şekil 5.1 ile gösterilmiştir. Şekilde ifade edilen senaryoda ajan çevre ile ilgili verileri anlık olarak sunucuya göndermektedir. Sunucu, ajanın göndermiş olduğu bu verileri kullanarak evrişimli sinirsel ağ modelinin ihtiyaç duyduğu görüntüyü

üretmekte ve modele girdi olarak vermektedir. Verilen girdi sonucunda modelin sunduğu en optimal eylem sunucu üzerinden UDP bağlantısı ile ajana gönderilir. Ajanın sunucuya eğitim için gönderdiği savaş ile ilgili veriler aşağıda açıklanmaktadır:

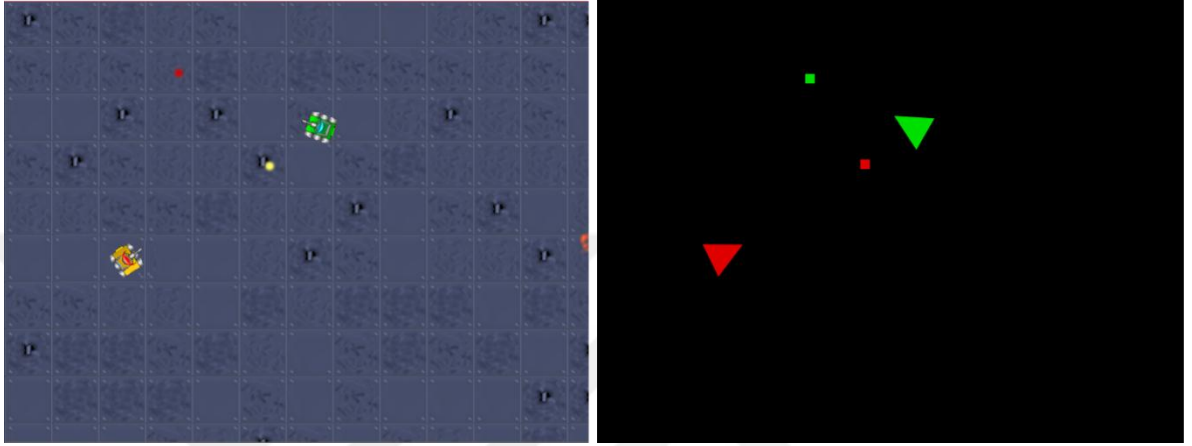


Şekil 5.1 Ajan ve sunucu arasındaki ilişkiyi gösterir akış şeması

- **X ve Y koordinatları:** Ajan kendisinin ve radar taraması sonucunda bulduğu rakip robotun bulunduğu x ve y koordinatlarını sunucuya gönderir.
- **Doğrultu:** Ajan Kendisinin ve radar taraması sonucunda bulduğu rakip robotun doğrultularını sunucuya gönderir.
- **Mermi:** Ajan düşmana ait mermilerin konumlarını sunucuya gönderir. Ajanın fırlatmış olduğu mermilerin koordinatları sunucuya gönderilen bilgiler içerisinde yer almamaktadır.
- **Enerji:** Ajan kendisinin ve radar taraması sonucunda bulduğu rakip robotun sahip olduğu enerjileri sunucuya gönderir. Bu bilgiler görüntü üretimi için değil, ödül sinyali için kullanılmaktadır.

Model sunucusuna robotun gönderdiği bilgiler, görüntü üretici yardımcı programı ile evrişimsel sinir ağları modeline uygun olan görüntülere dönüştürülmektedir. Ajanın arenada galip gelmesini sağlayacak robotların ve mermilerin konumları ile robotların doğrultuları bilgilerinin üretilen görüntünün içerisinde bulunması gerekmektedir.

Projede arenadaki robotlara ait dođrultu bilgilerini resim içerisinde ifade edebilmek için RoboCode uygulamasının aksine; robot nesnelerinin kare formasyonunda deđil üçgen formasyonunda üretilmesine karar verilmiştir. RoboCode arenasının sunduđu görüntü ve bu görüntünün evrişimsel sinir ađı girdisi olarak kullanılacak hale getirilmiş durumu Şekil 5.2 ile gösterilmiştir.



Şekil 5.2 RoboCode arenasına ait bir ekran görüntüsü (solda) ve bu görüntüye ait sinir ađı girdisi (sađda)

Görüntü üretcinin elde ettiđi resimlere ait 128 renk paleti ve 800×600 piksel görüntü olan ham savař görüntüleri ile doğrudan çalışmak ve ađı bu görüntüler ile eğitmek çok fazla hesaplama yükü getirmektedir. Bu nedenle girdiyi evrişimli sinir ađları modeline uygun hale getirebilmek için girdi boyutlandırmasını azaltmayı amaçlayan temel bir ön işleme adımı uygulanmaktadır. Ham görüntüler önce RGB renk yelpazesinden gri skalaya dönüřtürülmüş, boyutlandırma için 80×60 çözünürlükler kullanılmıştır. Evrişimli sinir ađları modeline uygun resim girdileri için gerekli matris operasyonlarının kolaylıđı ađ tasarımı kolaylařtırdıđı için kare çerçeveye sahip resimler ile çalışılması daha avantajlı olmaktadır.(Krizhevsky vd 2012) Bu sebeple belirlenen boyutlandırma ölçüleri kare olacak şekilde tekrar düzenlenmektedir. Arenada bulunan her noktaya robotların erişimi olduğundan resmin kırılması bilgi kaybına yol açmaktadır. Bu sebeple bu işlem resmi kıparak deđil resme ait küçük kenarı gererek çözülmüřtür. Bu işlem sonucunda hedef resim çözünürlüđu 80×80 olmuřtur.

RoboCode arenasındaki bir ajan rakip robotlara ait doğrultu, konum ve mermi bilgilerine yalnızca rakip robotu radarı ile taradığında ulaşabilmektedir. Görüntü üreticinin rakip robotlara ait bilgiler ile görüntü oluşturduğu ve ağı eğittiği düşünüldüğünde ajan yalnızca rakip robotu radarı ile yakaladığında sunucuya bilgi gönderebilmektedir. Robotun sunucuya rakip hakkında olabildiğince çok veri gönderebilmesi için projede robot radarı kendi eksenini etrafında sürekli hareket ettirilmektedir. Bu sayede ajan radar görüntüsü ile rakip robotu yakaladığı anda sunucuya gerekli bilgileri göndermektedir. Bu durum toplanacak bilgi miktarını azaltmakta iken rakip robotun olmadığı görüntülerin üretilmesinden doğacak tutarsızlıkların meydana gelmesini engeller.

RoboCode arenasında bir ajan için belirlenecek eylem setleri çok çeşitli olabilmektedir. Bu tez kapsamında aşağıdaki eylem setleri belirlenmiştir.

- Sağa 10 derece dön
- Sola 10 derece dön
- 5 piksel ileriye git
- 5 piksel geriye git
- 1 atış gücü ile ateş et

RoboCode ortamı sağa-sola dönüşler, ileri-geri hareketler ve merminin gücü için farklı değerlere izin verse de bu değerler problemi basitleştirebilmek için sabitlenmiştir.

Her ne kadar RoboCode ortamı savaş sonunda yapılan eylemlere göre bir puan veriyor olsa da bu puanı doğrudan almak ve ağı iletmek mümkün olmamaktadır. RoboCode doğrudan bir ödül sinyali sağlayamadığından yapay olarak bir ödül sinyali oluşturulması gerekmektedir. Bu konuda akla gelebilecek en standart yaklaşım, ajanın savaşı kazanıp kazanmadığına bağlı olarak sadece terminal durumlarında +1 veya -1 ödül vermektir. Ağ ilk bahsedilen yaklaşım ile eğitmeye çalışılmış fakat bu yöntem ile ödül sinyalinin oldukça seyrek frekansa sahip olduğu ve ajanın makul bir politika öğrenebilmesi için yeterince sinyal alamadığı görülmüştür.

RoboCode ortamında galip gelebilmek için uygulanan genel kazanma stratejileri incelendiğinde, bir arenada robotlara karşı üstün olabilmenin temel yolunun rakip robottan daha fazla enerjiye sahip olmasından geçtiği genel kanıdır. Bu sebeple ödül sinyali için temel odağın robotların sahip olduğu enerjilere bağlı bir işlev olması gerektiği düşünülmüştür. Sahip olunan enerjiler düşünüldüğünde robotların t anındaki enerjileri arasındaki farklar o an hangi robotun kazanma yolunda daha önde olduğuna dair bilgiler verse de robotların t-1 anındaki enerjilerin de ödül işlevine dahil olması düşünülmüştür. Bu durum ödül işlevinin, robotların merminin isabetinden kaynaklanan mermi düşüklüğü bilgisini de içermesi anlamına gelmektedir. Robotlara ait t ve t-1 anındaki enerjilerini içeren ödül işlevi aşağıdaki gibi belirlenmiştir:

$$R_t = A_t - A_{t-1} + R_{t-1} - R_t$$

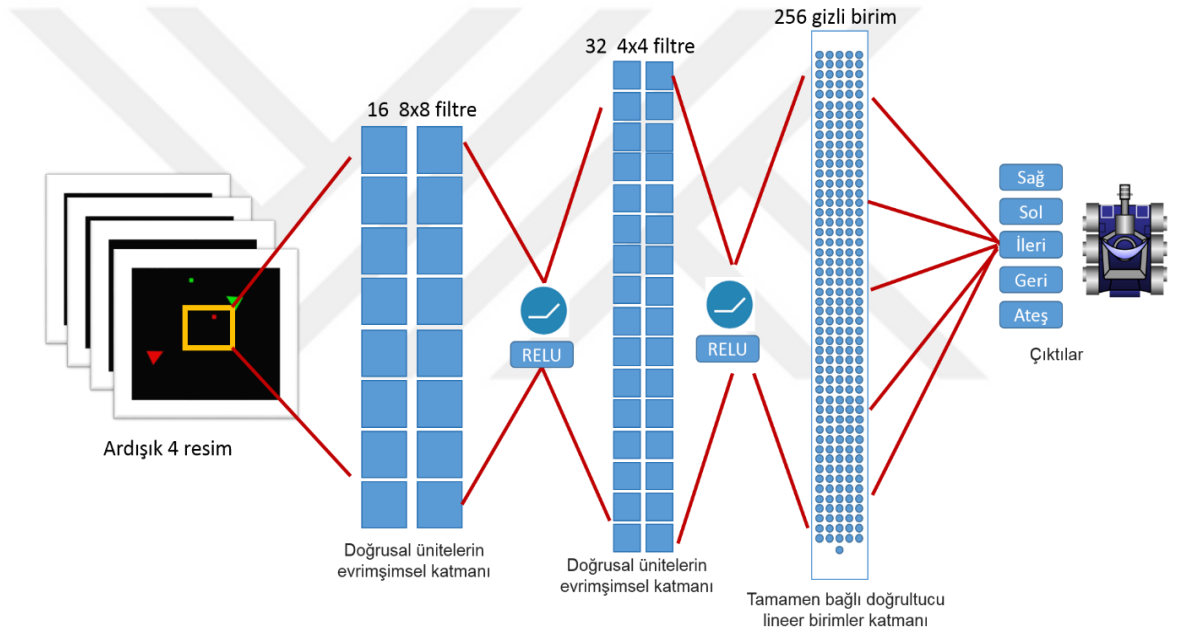
Fonksiyonda A_t ajanı R_t ise rakibi ifade etmektedir. Bu formül zamana bağlı olarak ajanın ilerledikçe tutarlı bir pozitif ödül almasını sağlamaktadır.

İndirim faktörü (γ) değeri 0.99 olarak tanımlanmış, bu değer ile ilk gelen ödüllere daha hafif bir öncelik verilmiştir.

Mnih ve diğerleri (2013)'nin Atari projesinde kullandığı gibi proje kapsamında ajanın eğitiminde epsilon-açgözlü politika uygulanmıştır. Bu kapsamda sinir ağı modeli bir eylemi seçerken uyguladığı maksimum q değerine ait eylemi seçmesi işlemi 1-epsilon seçim ihtimali ile sınırlandırılmaktadır. Bu sayede model epsilon ihtimali kadar rastgele eylem seçebilmektedir. Bu sayede ajanın öğrenimi sırasında epsilon değerine göre rastgelelik ile bir noktaya yakınsamasının önüne geçilmektedir. Projede yapılan deneyler sonucunda ilk 40.000 eğitim güncellemesinde epsilon değeri 1 ile 0.1 arasında oranlanmış ve daha sonra 0.1'de sabitlenmiştir. Bu durum ağın eğitim erken dönemde durum uzayını iyice keşfetmesine ve sonrasında doğru stratejiyi daha da geliştirebilmek için öğrendiği politikayı kullanmaya daha fazla odaklanmasına izin vermektedir.

Ağ yapısı 80x80 ekran görüntüleri ile ardışık 4 kare ile üretilen 80 x 80 x 4 lük girdi le oluşturulmuştur. Evrişimli sinirsel ağa ait ilk gizli katman, giriş görüntüsüyle 4 adımlık

(stride) 16 adet 8×8 filtrelerden oluşmakta olup her biri doğrusal olmayan bir doğrultucu (nonlinear rectifier) uygular (Nair vd. 2010). İkinci gizli katman 32 adet 4×4 filtreleri 2 adım ile tekrar birleştirmekte ve ardından bir kez daha doğrusal olmayan doğrultucu uygulamaktadır. Ağa ait son gizli katman birbirlerine tamamen bağlanmış 256 doğrultucu ünitesinden oluşan sinir ağı modelinden oluşmaktadır. Çıkış katmanı, her geçerli işlem için tek bir çıkışa sahip, tam olarak bağlanmış bir doğrusal katmandır. Ağa ait çıktılar daha önce belirtilen 5 aksiyondan (Sağa 10 derece dön, Sola 10 derece dön, 5 piksel ileriye git, 5 piksel geriye git, 1 atış gücü ile ateş et) oluşmaktadır. Belirtilen modelin yapısı Şekil 5.3 ile gösterilmiştir.



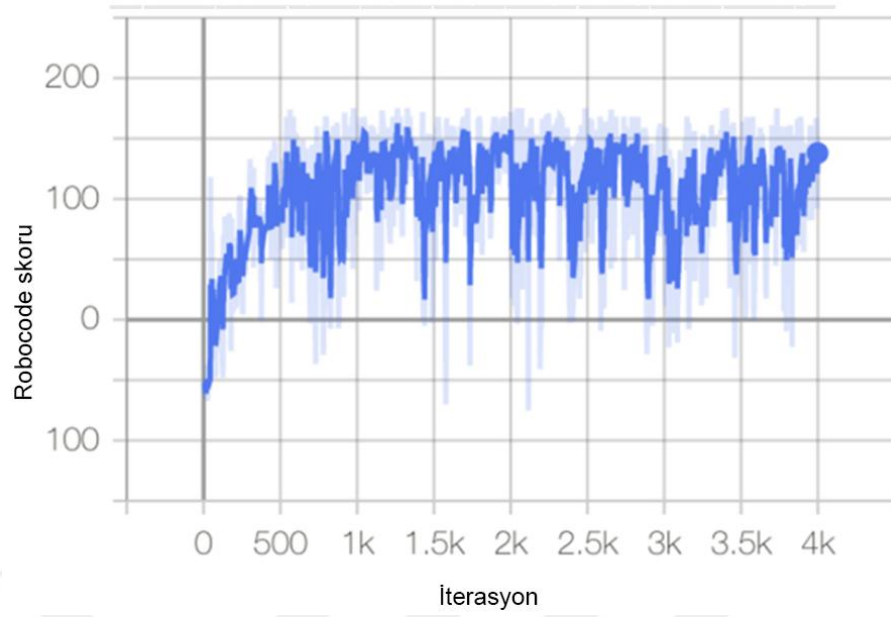
Şekil 5.3 Evrişimsel derin Q ağı model

6. DENEYLER

Ağ eğitimi kapsamında ilk olarak *SittingDuck* isimli hareket etmeyen ve herhangi bir mermi ateşlemesi eyleminde bulunmayan önceden tanımlı robot seçilerek evrişimli sinir ağı eğitilmesi amaçlanmıştır. Bahsedilen derin sinir ağları hücrelerine ait ağırlık değerleri, ajanlardan gelen girdiler ile her tur sonunda kaydedilmektedir. Kaydedilen bu değerler tur başlangıcı ile yeniden yüklenmekte ve robotlar daha önce eğitilmiş modeller ile eğitilmektedir. Bu sayede eğitilen modele ait tecrübelerin kümülatif artması ve ajanın arenada oynadığı her savaşta daha iyi skorlar elde etmesi amaçlanmıştır.

Denetimli öğrenme algoritmalarına ait uygulamalarda bir modelin eğitim sırasındaki performansı, eğitim ve validasyon setleri değerlendirilerek kolayca izlenebilmektedir. Ancak, pekiştirmeli öğrenme algoritmalarında, bir ajanın eğitimi sırasındaki ilerlemesini doğru bir şekilde değerlendirebilmek denetimli öğrenme kadar kolay olmamaktadır. Genel bir ölçüt olarak (Bellemare vd 2013) pekiştirmeli öğrenme algoritmalarındaki başarının, ajanın bir turda veya oyunun genelinde topladığı toplam ödül miktarı ile orantılı olduğu düşünülmektedir. Bu düşünceye uygun olarak toplam ödül miktarı eğitim sırasında periyodik olarak toplanarak performans ölçümü yapılmıştır. Toplam ödül metriği çok gürültülü olma eğilimindedir, çünkü bir politikanın ağırlığındaki küçük değişiklikler, politikaların ziyaret ettiği durumların dağılımında büyük değişikliklere yol açabilmektedir.

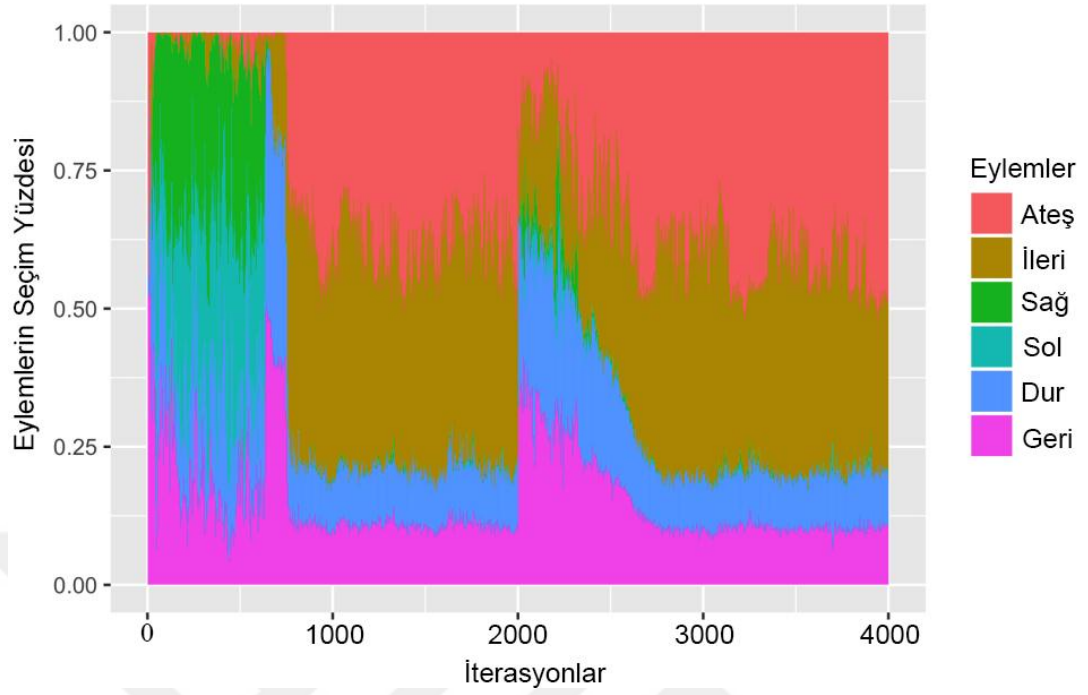
Deney kapsamında modelin eğitimine ilk olarak *SittingDuck* robotuyla başlanmıştır. Bu sayede bahsi geçen rakip robot ile savaşı ve galip gelmeye başlayan robotun, eğitim sonunda doğru mermi atışı eylemini öğrenme eğiliminde olacağı düşünülmüştür. Bu sebeple *SittingDuck* robotu ajan robot tarafından %100 başarı elde edilinceye kadar arenada savaştırılmış ve ajan 4000'in üzerinde mücadele sonrasında nihayet başarıyla hedeflenen politikayı öğrenmiştir. Eğitim sırasında ϵ -açgözlü strateji izlenmiş ve ϵ değeri 0.1 de tutulmuştur. Eğitim boyunca tur başına elde edilen toplan ödüllere ait grafik Şekil 6.1 ile gösterilmiştir.



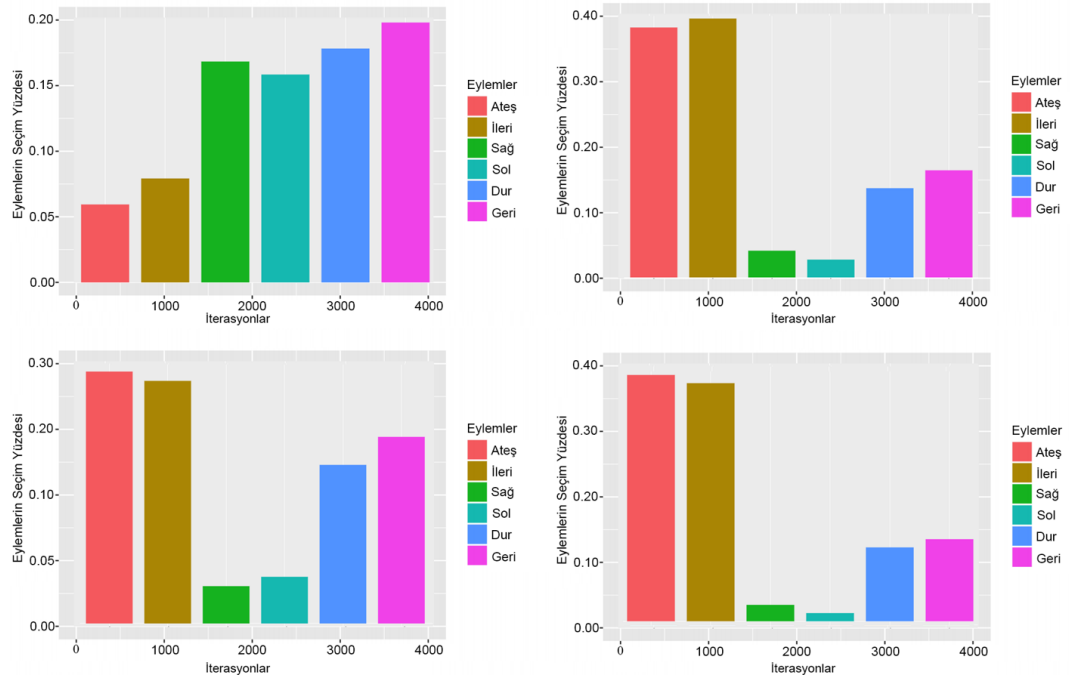
Şekil 6.1 SittingDuck isimli robotla yapılan mücadeleye ait ajanın öğrenme performansı

Öğrenme boyunca dikkat çekici bir diğer nokta ise hedeflenen şekilde robotun zaman içerisinde mermi ateşlemesi eylemine ağırlık vermeye ve sonrasında mermi atış eylemini giderek arttırmaya başlamasıdır. Bu durumu raporlayabilmek için ajanın uyguladığı bütün eylemler depolanmış ve sonrasında görselleştirilmiştir. Ajanın öğrenim boyunca uyguladığı eylemlerin miktarları Şekil 6.2 ile ve bu eylemlerin iterasyonlar aralığındaki dağılımları Şekil 6.3 ile verilmiştir.

Eğitilen ajanın SittingDuck robotunun ardından *Corners* isimli, arenada köşelere yönelerek radar ile tarama yapabilen ve radar alanına giren robota mermi atışında bulunan başka bir ön tanımlı robot ile savaştırılması hedeflenmiştir.

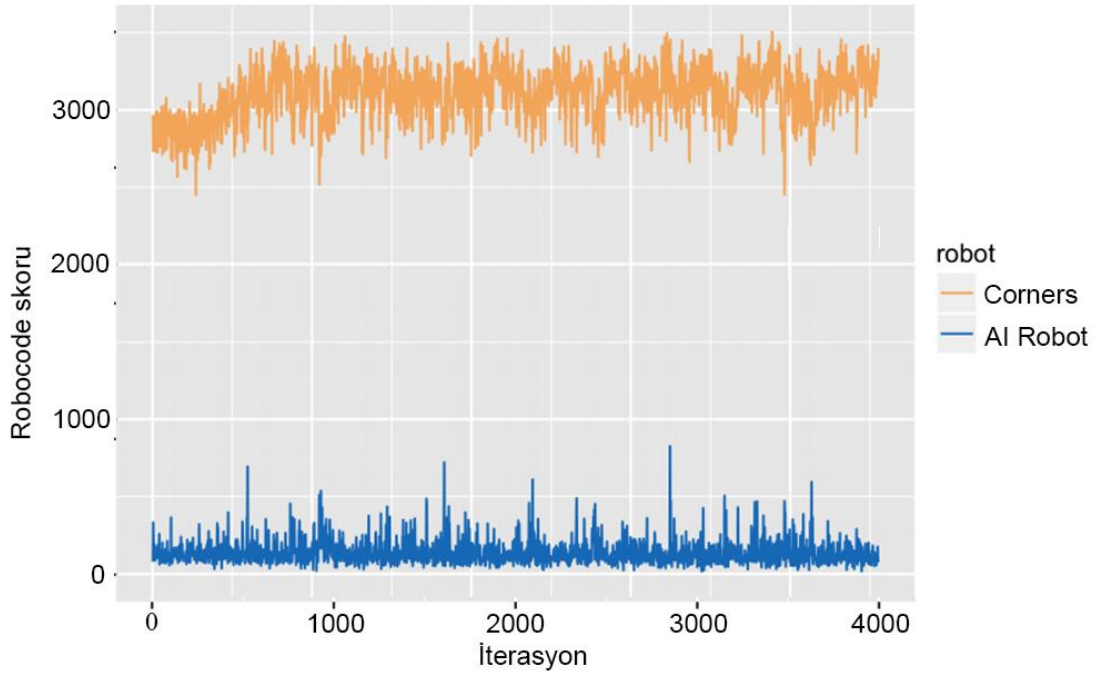


Şekil 6.2 SittingDuck isimli robotla savaşta model eğitimi boyunca seçilen eylemlerin yüzdesi



Şekil 6.3 Eğitim 1'inci, 2'nci, 3'üncü ve 4'üncü çeyreklerindeki seçilen eylemlerin yüzdesi

Hedeflenen eğitim için *SittingDuck* robotu ile önceden eğitilmiş ağ ve değerler kullanılarak arenada *Corners* robotu ile yaklaşık 4000 savaş yapılmış ve Şekil 6.4 ve şekil 6.5 de görüldüğü gibi modelin var olan yaklaşımlar ile *Corners* gibi agresif rakiplere karşı başarısız olduğu gözlemlenmiştir.

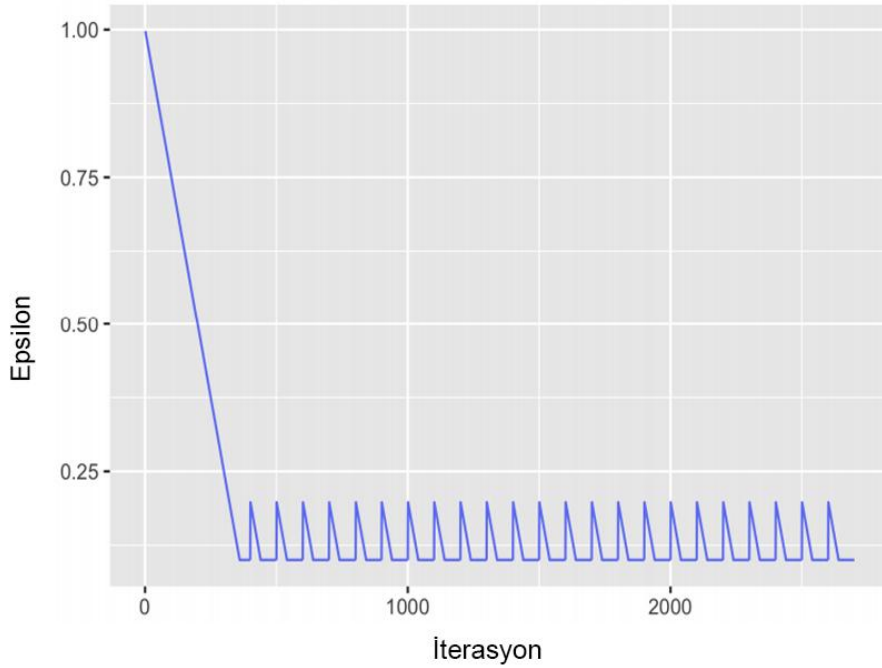


Şekil 6.4 Corners robotu ile yapılan savaşlar sonucunda elde edilen RoboCode skorları.

Eğitim sonucunda, ajan mermi ateşleyen rakiplerine karşı eğitilmediğinden, *Corners* robotuna karşı başarısız olmuştur. Çalışma kapsamında robotun *SittingDuck* robotuna karşı almış olduğu başarının sonrasında var olan eğitim modeli ile *Corners* robotuna karşı başarısızlığı üzerine yoğunlaşmıştır. Bu sebeple ağı eğitimi için birtakım optimizasyonlar uygulanmıştır. Bahsedilen optimizasyonlar aşağıda belirtilmiştir:

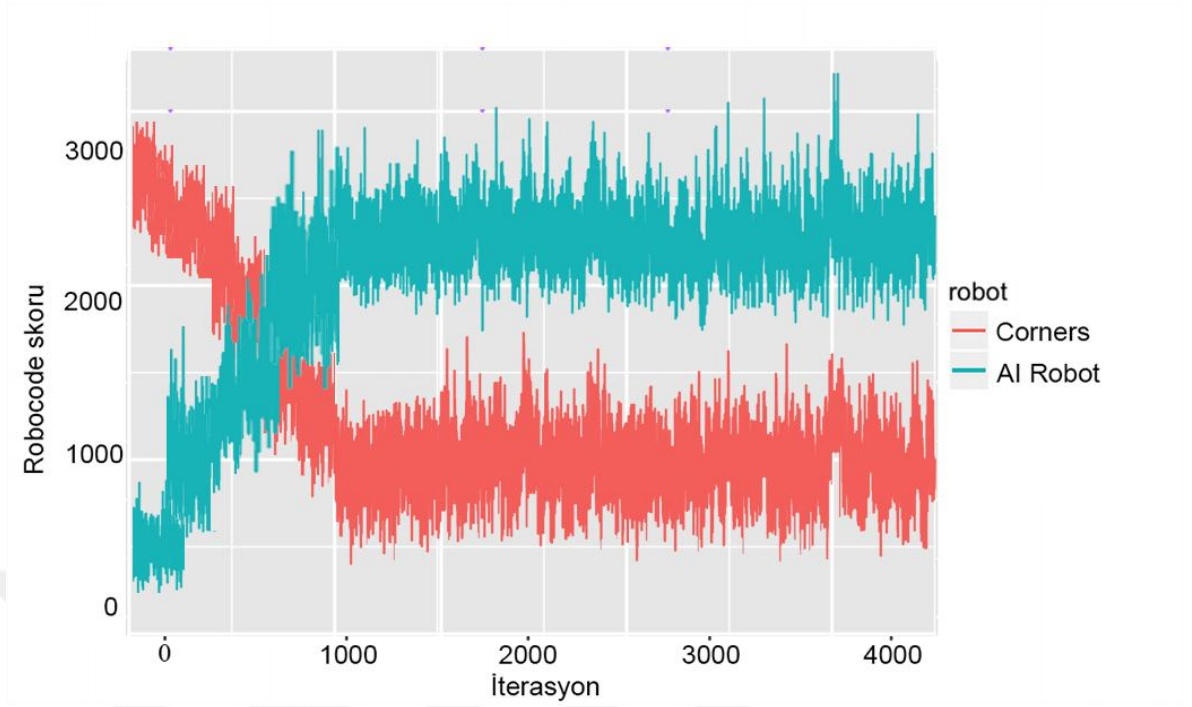
Bir RoboCode ajanın en büyük öğrenme zorluklarından biri, robotun ödüle daha önceden belirlenmiş bir ödül sinyalinin varlığında bile seyrek ulaşmasıdır. Bunun nedeni belirlenen ödül işlevi düşünüldüğünde, robotun herhangi bir olumlu ödül görebilmesi için rakip robotu rastgele yönlerde ateş ederek vurması gerektiğidir. Benzer olarak belirlenen ödül işlevi düşünüldüğünde robotun attığı mermi kendisinin de

enerjisini düşüreceğinden hedefe çarpmayan her mermi ajanın negatif ödül almasına sebep olacaktır. Sonuç olarak ajan erken dönemde sadece epsilon değeri 1 iken rastgelelik sayesinde olumlu birkaç ödül geçişi görebilecektir. Ajanın başlarda etkisi altına kalacağı rastgeleliği arttırabilmenin sonrasında ödüle ulaşmasında olumlu etkisi vardır. Bu sebeple epsilon değerinin eğitim başında 1 değeri ile başlayıp sonrasında doğrusal olarak azalarak her 100 iterasyonda 0.2 değerine sıçraması öngörülmüştür. İterasyon boyunca epsilon değerinin zamanla değişimi Şekil 6.5 ile gösterilmiştir.

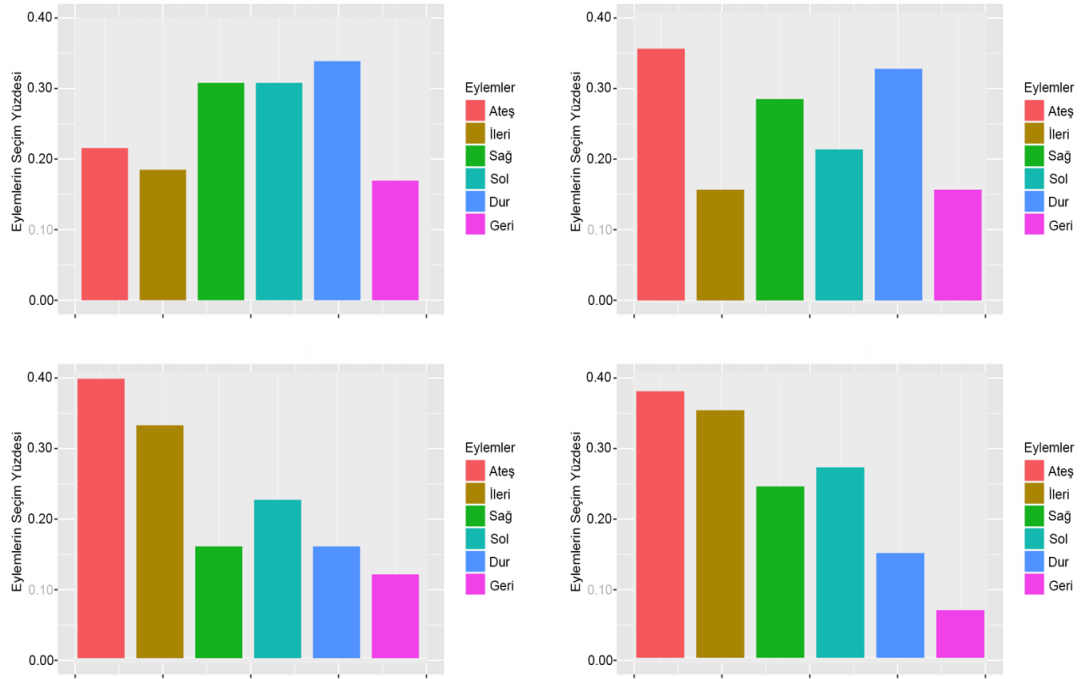


Şekil 6.5 Corners robotu için uygulanan epsilon stratejisi, başlangıçta lineer azalma devam etmekte ve sonrasında her 100 iterasyonda epsilon değeri güncellenmektedir

Bütün bunlara ek olarak robotun *SittingDuck* isimli robot ile öğrendiği ateş etme ve ileri gitme aksiyonlarının, saldırgan olan *Corners* robotu için başarıya ulaşamaması sebebi ile ağı bahsedilen optimizasyonlar ile baştan eğitilmesine karar verilmiş bu sayede ajanın öğrenme süreci boyunca önceki modele kıyasla daha dengeli aksiyonlar üretebilmesi hedeflenmiştir. *Corners* robotu ile önceden yapılan 4000 savaş tekrar yapılarak ağ eğitilmiştir. Eğitilen bu ağa ait skor değerleri ve uygulanan aksiyon dağılımları şekil 6.6 ve şekil 6.7 ile gösterilmiştir.



Şekil 6.6 Corners robotu ile yapılan savaşlar sonucunda elde edilen yeni RoboCode skorları.

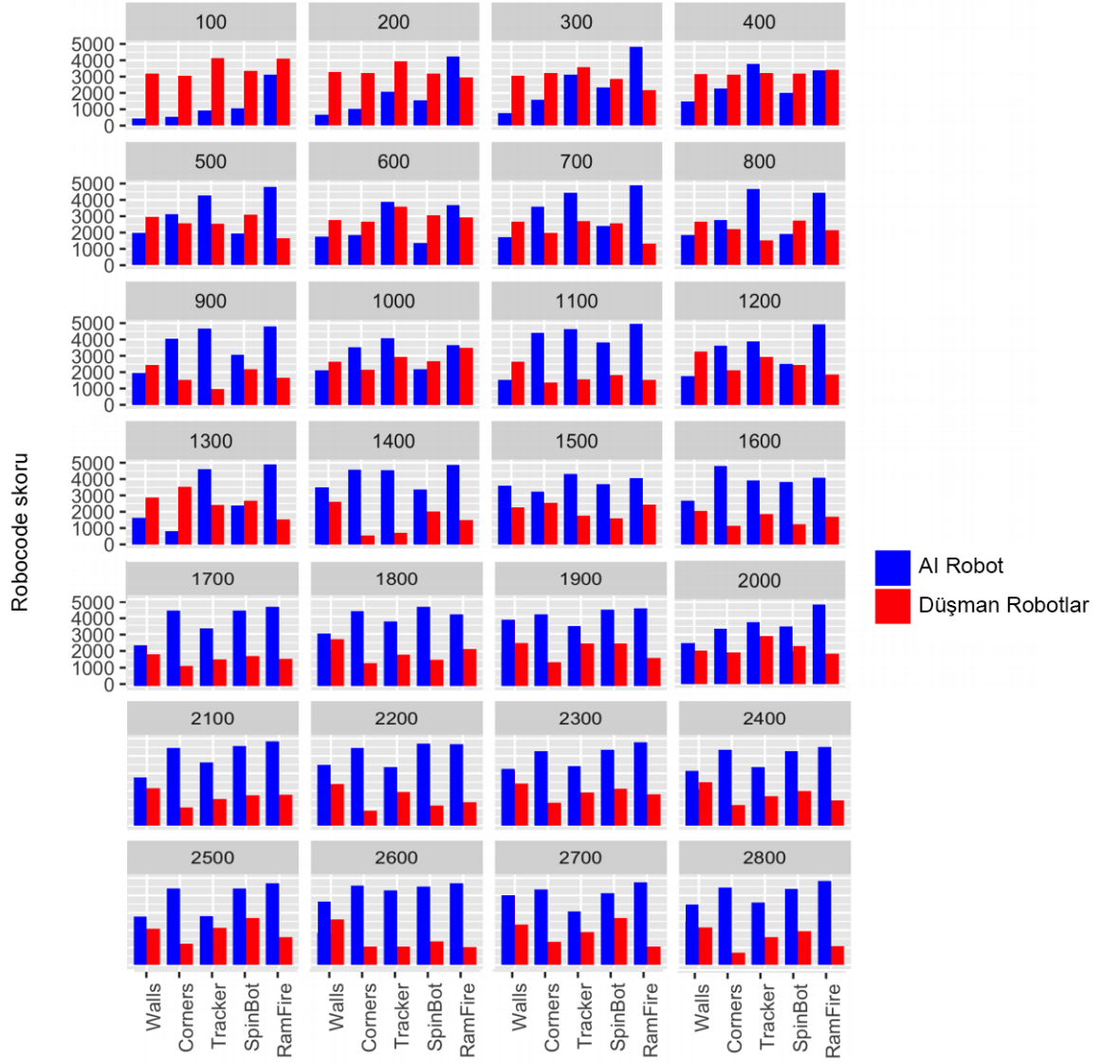


Şekil 6.7 Eğitim 1'inci, 2'nci, 3'üncü ve 4'üncü çeyreklerindeki seçilen eylemlerin yüzdesi

Şekil 6.6 ve şekil 6.7 grafiklerinin ışığında, *Corners* robotu ile yeniden eğitilen ajanın eğitim sonucunda vermiş olduğu dengeli karaların RoboCode skoruna da yansıdığı ve

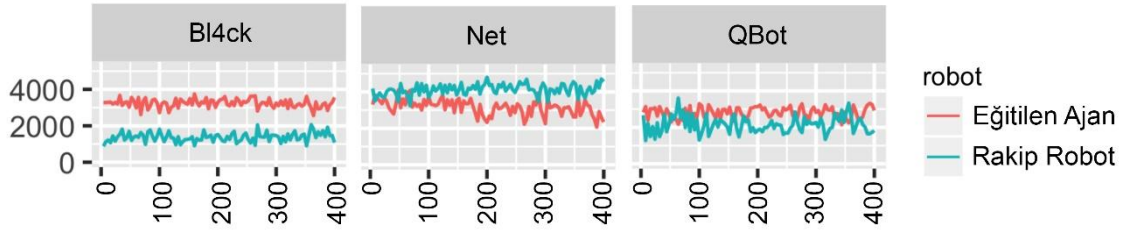
elde edilen modelin daha başarılı olduđu yorumu yapılmıştır. Elde edilen hedef eğitimi modeli kaydedilerek RoboCode simülatöründe önceden tanımlı robotlar ile arenada savaştırılmaya devam edilmiştir. Ajanın dengeli eğitilmesi için ön tanımlı robotların her biri ile ilk olarak 100'er iterasyon savaş yapılmış ve bu sayede modelin bütün robotlara ait tecrübelerden yararlanarak kümülatif olarak eğitilmesi amaçlanmıştır. Bahsedilen bu model sonraki her 100 iterasyon için uygulanarak robotun kümülatif öğrenmesi sağlanmıştır. Yapay zeka ajanı RoboCode simülatörüne ait ön tanımlı robotlardan her biri farklı savaş stratejisine sahip *Walls*, *Corners*, *Traker*, *SpinBot*, *RamFire* isimli robotlar ile ayrı ayrı 2800'er adet mücadeleye katılmış ve yapılan bu mücadelelere ait RoboCode skorları şekil 6.8 ile gösterilmiştir.





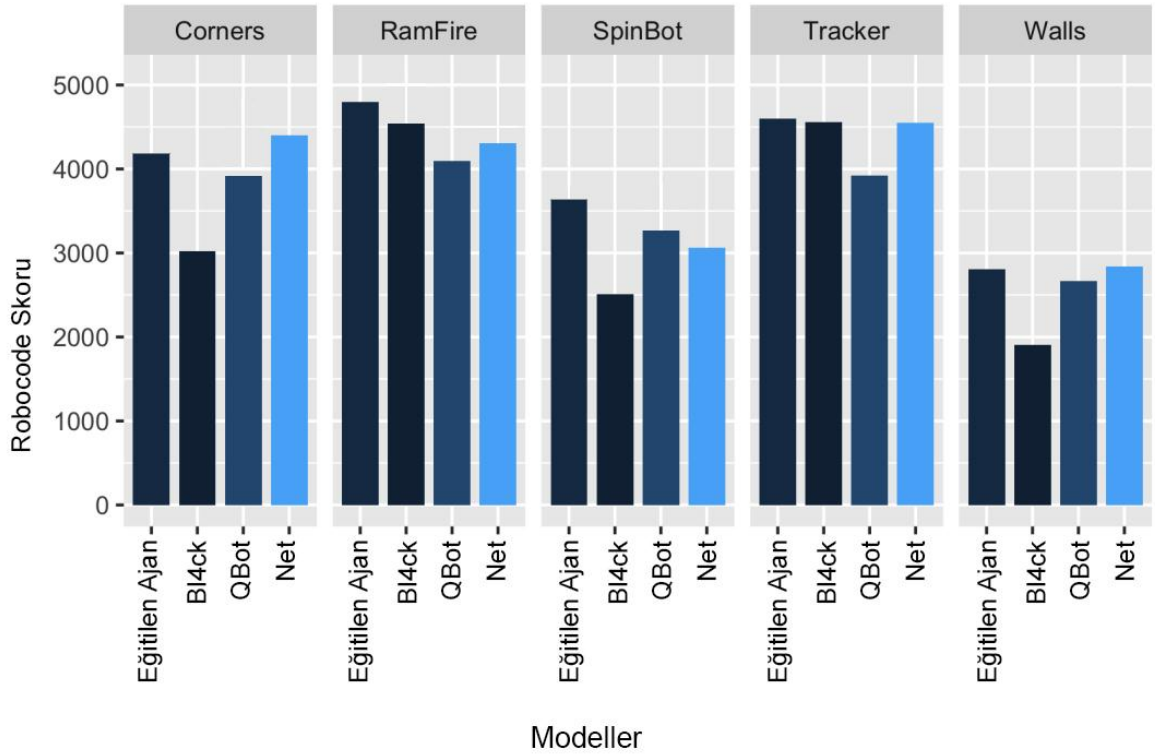
Şekil 6.8 Ajanın farklı ön tanımlı Robotlara karşı göstermiş olduğu performans

2800'er mücadele sonucu elde edilen model şekil 6.8 de görüldüğü gibi, ön tanımlı robotlar karşısında yüksek puanlar almayı başarabilmiştir. Modelin elde etmiş olduğu başarı sonrasında, tez kapsamında bahsedilen RobotRumble isimli, kullanıcıların geliştirmiş oldukları robotları yükleyebilmesine imkân tanıyan platformda bulunan robotlar ile mücadele etmesi hedeflenmiştir. Bu kapsamda eğitilen model ile, RobotRumble platformunda halihazırda bulunan makine öğrenmesi tabanlı *Black*, *QBot*, *Net* isimli 3 farklı robot ayrı ayrı mücadele etmiştir. Bu mücadelelere ait ilk 400 karşılaşmanın sonucunu gösteren grafik Şekil 6.9 ile gösterilmiştir.



Şekil 6.9 Eğitilen ajanın Bl4ck, Net ve QBot isimli robotlara karşı almış olduğu skorlar

Bahsedilen makine öğrenmesi tabanlı *Bl4ck*, *Net* ve *QBot* isimli robotların, bir diğer deney kapsamında, eğitilen ajanın daha önce eğitimi esnasında karşılaşmış olduğu ön tanımlı *Walls*, *Corners*, *Traker*, *SpinBot*, *RamFire* isimli robotlar ile mücadele ettirilmesi kararlaştırılmıştır. Bu kapsamda makine öğrenmesi tabanlı robotlar, arenada ayrı aynı ön tanımlı robotlar ile birebir olacak şekilde 100 iterasyon boyunca mücadele ettirilmiştir. Bütün bunlara ek olarak tez kapsamında elde edilen ajan da 100 iterasyon boyunca ön tanımlı robotlar ile mücadele ettirilmiş ve diğer robotlar ile skorları karşılaştırılmıştır. Elde edilen skorlara ait grafik Şekil 6.10 ile gösterilmiştir.



Şekil 6.10 Makine öğrenmesi tabanlı robotların ön tanımlı robotlar ile mücadelesiyle elde edilen skorlar

7. SONUÇ

Teknolojik gelişmelerin hızla ilerlediği günümüzde, yapay zeka, makine öğrenmesi ve robotik konularındaki araştırmalar sektörün istekleri ile birlikte önceki senelere göre dikkate değer gelişmeler göstermiştir. Bu konu akademik araştırmalarda da kendine oldukça fazla yer bulmuş ve bu konulara artan ilgi logaritmik hızda artarak devam etmektedir.

Tez kapsamında yapay zeka konusunun alt konularından olan makine öğrenmesi konusu dahilinde çalışmalar yapılmış, son zamanlarda derin sinir ağlarının erişilebilir ve güçlü bir araç haline gelmesi ile birlikte önemli atılımların olduğu pekiştirmeli öğrenme modellerinden Q öğrenme metodu kullanılarak RoboCode savaş simülasyonuna ait robotların eğitilmesi amaçlanmıştır. Bu konu ile alakalı literatür taraması yapıldığından q öğrenme modelinin güçlü varyasyonlarından olan derin evrimsel q öğrenme ile herhangi bir RoboCode robotunun geliştirilmediği görülmüş ve bu metod kullanılarak RoboCode savaş arenasında başarılı olabilecek robotun eğitilmesi hedeflenmiştir.

Bu makale, pekiştirmeli öğrenme modeli için daha önce kullanılan (Mnih vd. 2013) evrimsel sinir ağları mimarisini RoboCode simülasyonu için ortaya koymuş ve girdi olarak sadece arenadan gelen ham pikselleri kullanarak Atari 2600 bilgisayar oyunlarına ait çalışmaya benzer şekilde zor kontrol politikalarında ustalaşabildiğini göstermiştir. Atari oyunlarına kıyasla RoboCode, oldukça geniş eylem ve durum kümesine sahiptir. Ek olarak simülasyonun güvenlik sebebi ile bilgisayara ait kaynaklara erişimi konusunda çeşitli kısıtlamaları mevcuttur. Bu kısıtlamaların önüne geçebilmek için robotla haberleşebilen ve robotun gelecekte yapacağı eylemleri iletebilen bir sunucu modeli kurulmuş ve robotun bu sunucu ile sürekli konuşması sağlanmıştır. Robottan elde edilen bilgiler ışığında hedef evrimsel sinirsel ağ modeli arenayı sürekli deneyimlemiş ve ajanı yönlendirmiştir.

Bahsi geçen geniş eylem ve durum kümesine rağmen proje kapsamında Atari 2600 oyunlarında kullanılan modele ait hiperparametreleri kullanarak evrimsel sinir ağ modeli oluşturulmuştur. Model eğitimi esnasında sinir ağına girdi olarak verilen oyun

içi görüntülerin oldukça gürültü olması sebebi ile ajanın başarısız olduğu tespit edilerek oyun içi görüntüler yerine arenada savaş esnasında ajan tarafından elde edilen bilgiler kullanılarak görüntü üretme yoluna gidilmiştir. Bu sebeple ajan, arenada ilk doğduğu andan itibaren; kendi eksenini etrafında sürekli dönerek rakip robot hakkında bilgi elde etmekte, elde ettiği bütün bilgileri sunucuya göndererek sunucudan kendisi için ağ tarafından en optimal olduğu düşünülen eylemi beklemektedir.

Evrişimli sinir ağı ile oluşturulan model ilk olarak, arenada üstünlük kurmanın diğer robotlara kıyasla çok daha kolay olduğu, herhangi bir hareket ve mermi atışında bulunmayan *SittingDuck* isimli ön tanımlı robotlarda denenmiş ve %100 başarı elde edilmiştir. Bu başarı ile ajanın Q değerlerine bağlı kalmadan rassal olarak seçim yapmasını sağlayan epsilon açgözlü algoritmasının ajanın başarısına olan etkisi görülmüştür. Ajanın bahsi geçen robota karşı elde ettiği savaş deneyimlerini daha agresif robotlar için kullanması başarısız sonuçlar almasına yol açmıştır. Bu olay ajana öğrenme yolunda bir takım optimizasyonlar yapılamadan basit pekiştirmeli öğrenme görevlerinin bile özellikle seyrek ödül koşulları altında eğitilmesinin zor olabileceğini göstermiştir. Tespit edilen sorunların çözümü olarak yapılan birtakım optimizasyonlar ile birlikte ajan ön tanımlı robotlardan *Corners*, *RamFire* ve *Walls*, *SpinBot*, *Tracker* isimli farklı karakterlere ait robotlar ile savaştırılmış ve tez kapsamında performansları ölçülmüştür. Robotların eğitilmesi sırasında uygulanan kümülatif öğrenme yöntemi ile birlikte eğitim modelinin bahsi geçen ön tanımlı robotlara karşı önemli ölçüde başarı sağladığı görülmüş ve arenada aldığı skorlar raporlanmıştır. Eğitilen model sonrasında literatürde bulunan makine öğrenmesi tabanlı robotlar ile mücadele ettirilmiştir. Bu mücadelelerden elde edilen sonuçlar ile birlikte tez kapsamında önerilen modelin literatürde bulunan makine öğrenmesi tabanlı modellere yakın veya daha başarılı sonuçlar verdiği görülmüştür. Bu raporlar ışığında evrişimsel sinirsel ağ tabanlı derin Q-öğrenme algoritmasının Atari 2600 oyunlarında olduğu gibi RoboCode oyununda da başarılı olduğu kanıtlanmıştır.

8. ÖNERİLER

Tez kapsamında RoboCode simülatörüne ait ajanlar sadece birebir savařlar için eęitilmiřtir. RoboCode arenasına birden fazla robot eklenebilmekte, eklenen robotlar birbirleriyle takım halinde veya bireysel m¼cadele edebilmektedir. Takım oyununun karmařıklığı ve birden fazla d¼řman robota karřı belirlenecek aksiyonların fazlalığı d¼ř¼n¼ld¼ę¼nde birebir savařlar harici senaryolar tez kapsamı dıřına ¼ıkarılmıřtır. Tez kapsamında eęitilen mevcut model birden fazla robotlar için de test edilmiř olsa da, ajanın eęitilen model ile tutarlı eylemlerde bulunamadığı ve olduk¼a bařarısız olduęu g¼r¼lm¼řt¼r. Fakat birden fazla robot için yapılan akademik ¼alıřmaların azlığı d¼ř¼n¼ld¼ę¼nde bu durum bir sonraki ¼alıřmalarda ¼zerinde durulacak önemli bir k¼ře tařı olarak belirlenmiřtir.

Yapay zeka biriminin kendi kendileri ile oynatılarak devamlı surette eęitilmesi de son yılların olduk¼a pop¼ler bir kavramı olarak g¼ze ¼arpmaktadır. Bu konu modelin geliřtirilmesi sırasında denenmiřse de robotun deterministik davranıřlarından dolayı bařarı saęlanamamıřtır. Bu kavrama uygun olarak tez kapsamında bulunmayan fakat sonraki ¼alıřmalarda ¼zerine durulacak önemli bir konu olan robotun kendi kendine eęitilmesi de olduk¼a önemli bir kazanım olarak gelecek ¼alıřmalarda yerini alacaktır.

KAYNAKLAR

- Bellemare, G., Naddaf, Y., Veness, J. and Bowling, M. 2013. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279
- Dahl, G. E., Yu D., Deng L. and Acero, A. 2012. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *Audio, Speech, and Language Processing, IEEE Transactions on*, 20(1):30 –42.
- Eisenstein, J. 2003. Evolving robocode tank fighters. CSAIL Technical Reports, Massachusetts Institute of Technology Computer Science and Artificial Intelligence Laboratory.
- Frøjhær, J., Kristiansen, M. L., Hansen, P. B., Larsen, I. V. S., Malthesen, D. and Suurland, R. 2004. Robocode, development of a robocode team. Technical report, Department of Computer Science, Aalborg University.
- Fukushima, K. 1988. Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural Networks Vol 1*, 119–130
- Gade, M., Knudsen, M., Kjær, R. A., Christensen, T., Larsen, C. P., Pedersen, M. D. and Andersen, J. K. S. 2003. Applying machine learning to robocode. Technical report, Department of Computer Science, Aalborg University.
- Graves, A., Mohamed, A. and Hinton, G.E. 2013. Speech recognition with deep recurrent neural networks.
- Hastie, T., Tibshirani, R. and Friedman, J. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer
- Heess, N., Silver, D. and Teh, Y.W. 2012. Actor-critic reinforcement learning with energy-based policies. In *European Workshop on Reinforcement Learning*, 43.
- Huang, S., Papernot, N., Goodfellow, I., Duan, Y. and Abbeel, P. 2017. Adversarial Attacks on Neural Network Policies.
- IBM. 2019. Web Sitesi: <https://www.ibm.com/developerworks/library/j-tipstrats/index.html> Erişim tarihi: 24.08.2019
- Kobayashi, K., Uchida, Y. and Watanabe, K. 2003. A study of battle strategy for the Robocode, SICE Annual Conference in Fukui.
- Krizhevsky, A., Sutskever I. and Hinton, G. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, 1106–1114.
- LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE*, 2278–2324.
- Lin, L. 1993. Reinforcement learning for robots using neural networks. *Teknik Rapor, DTIC Dökümanı*.
- McCarthy, J., Shannon, C., and Goldstine, H. 1956. *Automata Studies*. Princeton: Princeton University Press.

- Mnih, V. 2013. Machine Learning for Aerial Image Labeling. Doktora Tezi, University of Toronto.
- Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D. and Riedmiller, M. 2013. Playing atari with deep reinforcement learning. In NIPS Workshop on Deep Learning.
- Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K. ve Ostrovski, G. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529– 533.
- Morten G., Michael K., Kjær R. A., Thomas C., Christian P.L., David M., Jens K. ve Søgaard, A. 2003. Applying machine learning to robocode. <http://www.dinbedstemedarbejder.dk/Dat3.pdf>. Erişim tarihi: 03.21.2017.
- Nair, V. ve Hinton, G. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML 2010)*, 807–814.
- Pollack, J. ve Blair, A. 1996. Why did td-gammon work. In *Advances in Neural Information Processing Systems 9*, 10–16.
- Riedmiller, M. 2005. Neural fitted q iteration—first experiences with a data efficient neural reinforcement learning method. In *Machine Learning: ECML 2005*, 317–328.
- Sallans, B. ve Hinton, G. 2004. Reinforcement learning with factored states and actions. *Journal of Machine Learning Research* 5, 1063–1088.
- Samuel, A. 1959. Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development*, 3-3
- Sermanet, P., Kavukcuoglu, K., Chintala S. ve LeCun, Y. 2013. Pedestrian detection with unsupervised multi-stage feature learning. In *Proc. International Conference on Computer Vision and Pattern Recognition (CVPR 2013)*.
- Shichel, Y., Ziserman, E. ve Sipper, M. 2005. Gp-robocode: Using genetic programming to evolve robocode players. Department of Computer Science, Ben Gurion University, Israil.
- Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman S., Grewe D., Nham J., Kalchbrenner N., Sutskever I., Lillicrap T., Leach M., Kavukcuoglu K., Graepel T. ve Hassabis, D. 2016. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–503.
- Sutton, R.S., Barto, A.G. 2017. Reinforcement Learning. An Introduction. Cambridge, MA: MIT Press.
- Tesauro, G. 1995. Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3):58–68.

ÖZGEÇMİŞ

Adı Soyadı : Hakan KAYAKÖKÜ

Doğum Yeri : Malatya, Ankara

Doğum Tarihi : 02/07/1993

Medeni Hali : Evli

Yabancı Dili : İngilizce

Eğitim Durumu (Kurum ve Yıl)

Lise : Malatya Fen Lisesi (2011)

Lisans : Ankara Üniversitesi Bilgisayar Mühendisliği Bölümü (2016)

Yüksek Lisans: Ankara Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği
Anabilim Dalı (Eylül 2016 - Eylül 2019)

Çalıştığı Kurum/Kurumlar ve Yıl

Simsoft Bilgisayar Teknolojileri Ltd. Şti. (2016 - 2016)

ASELSAN A.Ş. (2016 - ...)

Yayınlar

Güzel, M. S. ve Kayakökü, H. 2017. A Collective Behaviour Framework For Multi-Agent Systems. Mechatronics And Robotics Engineering For Advanced And Intelligent Manufacturing. Springer, 61-71

Hakemli Dergiler

Güzel, M. S., Kayakökü, H., Babaei, V. 2016. A Robotic Simulator Tool for Mobile Robots. 4th International Symposium on Innovative Technologies in Engineering and Science.