MASA TENİSİ SİMÜLASYONU İÇİN DERİN ÖĞRENME ALGORİTMALARININ KARŞILAŞTIRILMASI

Ahmed Naji Musleh NUSARI

Danışman: Doç. Dr. İbrahim Yücel ÖZBEK

Yüksek Lisans Tezi

Elektrik-Elektronik Mühendisliği Ana Bilim Dalı

2021

(Her hakkı saklıdır.)

T.C.
ATATÜRK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANA BİLİM DALI

MASA TENİSİ SİMÜLASYONU İÇİN DERİN ÖĞRENME ALGORİTMALARININ KARŞILAŞTIRILMASI

(Comparison of Deep Learning Algorithms for Table Tennis Simulation)

YÜKSEK LİSANS TEZİ

Ahmed Naji Musleh NUSARI

Danışman: Doç. Dr. İbrahim Yücel ÖZBEK

Erzurum

Haziran, 2021

KABUL VE ONAY TUTANAĞI

Doç. Dr. İbrahim Yücel ÖZBEK danışmanlığında, Ahmed Naji Musleh NUSARI tarafından hazırlanan bu çalışma, 25/06/2021 tarihinde aşağıdaki jüri tarafından Elektrik-Elektronik Mühendisliği Anabilim Dalı Anabilim Dalı Haberleşme Bilim Dalı'nda Yüksek Lisans tezi olarak **oybirliği (3 / 3)** ile kabul edilmiştir.

Jüri Başkanı: (Danışman)	Doç. Dr. İbrahim Yücel ÖZBEK *Atatürk Üniversitesi*	…….....
Jüri Üyesi:	Dr. Öğr. Üyesi Emin Argun ORAL *Atatürk Üniversitesi*	…….....
Jüri Üyesi:	Doç. Dr. Kağan Koray AYTEN *Erzurum Teknik Üniversitesi*	…….....

Enstitü Yönetim Kurulunun
…./…/…. tarih ve ……
sayılı kararı.

Bu tezin Atatürk Üniversitesi Lisansüstü Eğitim ve Öğretim Yönetmeliği'nin ilgili maddelerinde belirtilen şartları yerine getirdiğini onaylarım.

Prof. Dr. Saltuk Buğrahan CEYHUN

Enstitü Müdürü

Not: Bu tezde kullanılan özgün ve başka kaynaklardan yapılan bildiriş, çizelge, şekil ve fotoğrafların kaynak olarak kullanımı, 5846 sayılı Fikir ve Sanat Eserleri Kanunundaki hükümlere tabidir.

ETİK BİLDİRİM VE İNTİHAL BEYAN FORMU

Yüksek Lisans Tezi olarak Doç Dr. İbrahim Yücel ÖZBEK danışmanlığında sunulan "Masa Tenisi Simülasyonu İçin Derin Öğrenme Algoritmalarının Karşılaştırılması" başlıklı çalışmanın tarafımızdan bilimsel etik ilkelere uyularak yazıldığını, yararlanılan eserlerin kaynakçada gösterildiğini, Fen Bilimleri Enstitüsü tarafından belirlenmiş olan Turnitin Programı benzerlik oranlarının aşılmadığını ve aşağıdaki oranlarda olduğunu beyan ederiz.

Tez Bölümleri	Tezin Benzerlik Oranı (%)	Maksimum Oran (%)
Giriş	*12*	**30**
Kuramsal Temeller	*1*	**30**
Materyal ve Yöntem	*0*	**35**
Bulgular	*0*	**20**
Tartışma	*0*	**20**
Tezin Geneli	*11*	**25**

Not: *Yedi kelimeye kadar benzerlikler ile Başlık, Kaynakça, İçindekiler, Teşekkür, Dizin ve Ekler kısımları tarama dışı bırakılabilir. Yukarıdaki azami benzerlik oranları yanında tek bir kaynaktan olan benzerlik oranlarının* ***%5'den*** *büyük olmaması gerekir.*

Beyan edilen bilgilerin doğru olduğunu, aksi halde doğacak hukuki sorumlulukları kabul ve beyan ederiz.

Tez Yazarı (Öğrenci)	**Tez Danışmanı**
Ahmed Naji Musleh NUSARI	Doç Dr. İbrahim Yücel ÖZBEK
9.7.2021	9.7.2021
İmza:	İmza:

***** Tez ile ilgili YÖKTEZ'de yayınlamasına ilişkin bir engelleme var ise aşağıdaki alanı doldurunuz.

☐ Tezle ilgili patent başvurusu yapılması / patent alma sürecinin devam etmesi sebebiyle Enstitü Yönetim Kurulunun …./…/…. tarih ve ……. sayılı kararı ile teze erişim 2 (iki) yıl süreyle engellenmiştir.

☐ Enstitü Yönetim Kurulunun …./…/…. tarih ve ……. sayılı kararı ile teze erişim 6 (altı) ay süreyle engellenmiştir.

TEŞEKKÜR

Tez çalışmam süresince her zaman yanımda olduğunu hissettirip desteklerini esirgemeyen, tecrübe ve bilgi birikimlerini benimle paylaşan danışman hocam Sayın Doç. Dr. İbrahim Yücel ÖZBEK'e sonsuz teşekkürlerimi sunarım. Bilgilerini, tecrübelerini esirgemeyen ve önerileriyle yol gösteren kıymetli hocam Sayın Dr. Öğr. Üyesi Emin Argun ORAL'a göstermiş oldukları hoşgörü, sabır ve emeklerinden dolayı teşekkür ederim.

Gerçekleştirdiğim bu tez çalışmasının uygulama aşamalarında bana yardımlarını esirgemeyen Merve POLAT, Aslı Nur ÖMEROĞLU, Burcu TİRYAKİ, Hussein ALAMERİ, Nida KUMBASAR ve Rabiye KILIÇ'a şükran borçluyum. Öğrenim hayatım boyunca gösterdikleri sabır ve fedakârlıklarından dolayı aileme sonsuz teşekkür ederim.

Ahmed Naji Musleh NUSARI

ÖZET

YÜKSEK LİSANS TEZİ

MASA TENİSİ SİMÜLASYONU İÇİN DERİN ÖĞRENME ALGORİTMALARININ KARŞILAŞTIRILMASI

Ahmed Naji Musleh NUSARI

Danışman: Doç. Dr. İbrahim Yücel ÖZBEK

Amaç: Bu tez çalışmasında, masa tenisi (Ping Pong) oyun simülasyonunda derin öğrenme yöntemlerinin performansları karşılaştırılmıştır. Ping Pong oyun simülasyonunda birbiriyle oynayan iki oyuncu bulunmaktadır. Bunlardan birisi oyunu oynayan insan, diğeri ise yapay zekâ tarafından kontrol edilen oyuncu (ajan) olarak kabul edilmiştir. Yapay zekâ tarafından kontrol edilen oyuncu CNN, LSTM ve DQN yöntemleri kullanılarak eğitilmiştir.

Yöntem: Çalışma genel olarak CNN, LSTM ve DQN algoritmalarıyla gerçekleştirilmiştir. CNN ve LSTM modellerinin eğitilmesi üç adımdan oluşmaktadır. İlk adımda, eğitim veri kümesi toplanmaktadır. Veri kümesi iki elementten oluşmaktadır. Bunlardan ilki oyun ekran resmi, ikincisi ise etikettir. İkinci adımda, görüntüler iyileştirilir. Bu adımda, yeniden boyutlandırma ve kontrast artırma işlemi yapılır. Ayrıca eğitim süreçlerini daha hızlı, işlevsel ve güvenilir hale getirmek için eğitim veri setlerinden gereksiz kısım (rakip oyuncu barı) silinir. Son adımda ise farklı CNN ve LSTM modelleri bu veri setleri kullanılarak en iyi sonuçları elde etmek için eğitilir. DQN algoritması için ise kullanılan veri kümesi gerçek zamanlı olarak doğrudan ortamdan alınır. Daha sonra bu veriler iyileştirilir. Ayrıca, öğrenmeyi daha hızlı hale getirmek için resimler yeniden boyutlandırılır.

Bulgular: Deneyler ve eğitim sonuçlarına göre CNN ve LSTM yöntemleri kullanıldığında zaman gecikmesi olduğu görülmüştür. Oyun çalıştırıldığında orijinal hızdan daha yavaş çalıştığı ancak modellerin doğru sınıflandırma verdiği belirlenmiştir. DQN yönteminde ise bu zaman gecikmesi sorunu çözülmüştür. Oyun farklı hızlarda test edilerek çalıştırılmış ve performansı ölçülmüştür.

Sonuç: CNN ve LSTM yöntemlerinde bilgisayarın hesaplama gücüyle ters orantılı olarak bir gecikme yaşanmıştır. Öte yandan zaman gecikmesi olmadan yüksek performans gösteren DQN yöntemi kullanılarak bar ve topun farklı hız değerleri denenmiş, sorunsuz ve verimli çalıştığı gözlenmiştir.

Anahtar Kelimeler: Derin öğrenme, DQN, CNN, LSTM, Ping Pong, Yapay zekâ, Pekiştirmeli öğrenme.

Haziran 2021, 64 sayfa

ABSTRACT

MASTER'S THESIS

TABLE TENNIS GAME APPLICATION BASED ON DEEP REINFORCEMENT LEARNING

Ahmed Naji Musleh NUSARI

Supervisor: Assoc. Prof. Dr. İbrahim Yücel ÖZBEK

Purpose: In this thesis, the performances of deep learning methods in table tennis (Ping Pong) game simulation were compared. In the Ping Pong game simulation, there are two players playing with each other. One of them was accepted as the human playing the game, and the other as the player (agent) controlled by artificial intelligence. Controlled by artificial intelligence, the player is trained using CNN, LSTM and DQN methods.

Method: The study was generally carried out with CNN, LSTM and DQN algorithms. Training the CNN and LSTM models consists of three steps. In the first step, the training dataset is collected. The dataset consists of two elements. The first of these is the game screenshot, and the second is the sticker. In the second step, the images are enhanced. In this step, resizing and contrast enhancement are done. In addition, the unnecessary part (opponent player bar) is deleted from the training datasets to make the training processes faster, functional and reliable. In the last step, different CNN and LSTM models are trained using these datasets to obtain the best results. For the DQN algorithm, the dataset used is taken directly from the environment in real time. This data is then refined. Also, images are resized to make learning faster.

Finds: According to the results of the experiments and training, it was observed that there was a time delay when CNN and LSTM methods were used. It has been determined that when the game is run, it runs slower than the original speed, but the models give correct classification. In the DQN method, this time delay problem is solved. The game has been tested and run at different speeds and its performance has been measured.

Results: There was a delay in CNN and LSTM methods inversely proportional to the computing power of the computer. On the other hand, using the DQN method, which shows high performance without time delay, different speed values of the bar and ball were tried and it was observed that it worked smoothly and efficiently.

Keywords: artificial intelligence, CNN, DQN, LSTM, deep learning, Ping Pong, reinforcement learning

June 2021, 64 pages

İÇİNDEKİLER

TABLOLAR DİZİNİ

ŞEKİLLER DİZİNİ

KISALTMALAR VE SİMGELER DİZİNİ

AGI	Yapay Genel Zekâ (Artificial General İntelligence)
AI	Yapay Zekâ (Artificial İntelligence)
ANI	Yapay Dar Zekâ (Artificial Narrow Intelligence)
ANN	Yapay Sinir Ağları (Artificial Neural Network)
ASI	Yapay Süper Zekâ (Artificial Super İntelligence**)**
CNN	Evrişimli Sinir Ağı (Convolutional Neural Network)
DDPG	Derin Belirleyici Politika Gradyanı (Deep Deterministic Policy Gradient)
DQN	Derin Q-Öğrenme Ağı (Deep Q-Learning Netwok)
LEAKY RELU	Leaky Rektifiye Lineer Birim (Leaky Rectified Linear Unit)
LSTM	Uzun Kısa Süreli Bellek Ağları (Long Short-Term Memory Networks)
ML	Makine Öğrenmesi (Machine Learning)
NLP	Doğal Dil İşleme (Natural Language Processing)
RELU	Rektifiye Lineer Birim (Rectified Linear Unit)
RL	Pekiştirmeli Öğrenimi (Reinforcement Learning)
RNN	Tekrarlayan Sinir Ağları (Recurrent Neural Network)
SARSA	Durum-Eylem-Ödül-Durum-Eylem (State-Action-Reward-State-Action)
SDL	Basit Direct Media Katmanı Simple Direct Media Layer
SRN	Basit Tekrarlayen Ağ (Recurrent Neural Network)

GİRİŞ

Yapay Zekâ (AI)

Yapay zekâ (AI) programlanabilir makinelerin insan zekâsını simüle etme yeteneğini ifade eden bir bilgisayar bilimleri dalıdır. Yapay zekâ, makinelerin sergilediği özelliklere ek olarak problem çözme, öğrenme, konuşma tanıma ve planlama gibi insan zihniyle ilişkilendirilebilir uygulamalarda kullanılır.

Yapay zekânın ideal özelliği, belirli bir hedefe ulaşma şansı en yüksek olan eylemleri rasyonelleştirme ve gerçekleştirme yeteneğidir. Yapay zekânın alt kümesi olan, makine öğrenmesi ve derin öğrenme teknikleri metin, görüntü veya video gibi büyük ve işlenmemiş verileri öğrenmek için kullanılır (Jake Frankenfield 2021).

Günlük hayatımızda, AI uygulamaları sıklıkla görülmektedir. AI türleri çeşitli alt başlıklara ayrılarak incelenir (Şekil 1). Ancak genel olarak, bunları yeteneklere ve işlevseliğe göre iki kategoriye ayırabiliriz.

Yeteneklere Göre Yapay Zekâ Türleri

Yeteneklere göre Yapay Zekâ; Yapay Dar Zekâ (ANI), Yapay Genel Zekâ (AGI) ve Yapay Süper Zekâ (ASI) olarak sınıflandırılır.

Yapay Dar Zekâ (ANI): özel bir görevi yerine getirebilen bir yapay zekâ türüdür. Yapay Zekâ uygulamalarında yaygın olarak kullanılmaktadır. Dar Yapay Zekâ, yalnızca belirli bir görev için eğitildiğinden istenilen çalışma alanının ötesinde performans gösteremez. Google Arama, Siri, Alexa, Cortana ve IBM'in Watson'ı, dar yapay zekâ uygulamaları için örnek verilebilir.

Yapay Genel Zekâ (AGI): Güçlü AI olarak da isimlendirilen Yapay Genel Zekâ, insan zekâsını ve / veya davranışlarını taklit eden, herhangi bir sorunu çözmek için öğrenme ve uygulama becerisine sahip bir makine kavramıdır. Şu anda, genel AI kapsamına girebilecek ve bir insan gibi herhangi bir görevi mükemmel yerine getirebilecek bir sistem yoktur.

Yapay Süper Zekâ (ASI): Makinelerin insan zekâsını geçebileceği ve bilişsel özelliklerle herhangi bir görevi insandan daha iyi gerçekleştirebileceği bir Sistemler Zekâsı seviyesidir. ASI'nın, kendi başına düşünme, akıl yürütme, bulmacayı çözme, yargılarda bulunma, planlama, öğrenme ve iletişim kurma becerisi gibi temel özellikleri vardır. Bununla birlikte, ASI varsayımsal bir Yapay Zekâ kavramıdır ve geliştirilebilir (Eban Escott 2017).

İşlevselliğe Göre Yapay Zekâ Türleri

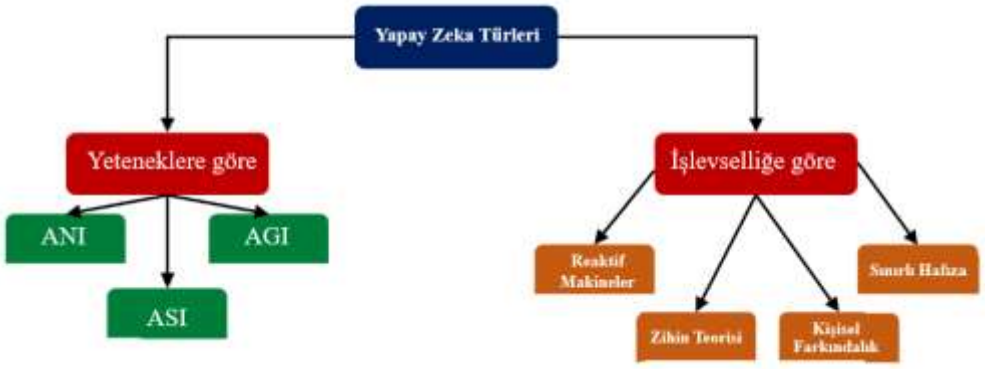
İşlevselliğine göre Yapay zekâ türleri Reaktif Makineler, Sınırlı Hafıza, Zihin Teorisi ve Kişisel Farkındalık olarak sınıflandırılmaktadır.

Reaktif Makineler: Tamamen reaktif makineler, en temel Yapay Zekâ türlerindendir. Gelecekteki eylemler için geçmiş deneyimleri saklayamazlar. Bu makineler yalnızca mevcut senaryolara odaklanır ve mümkün olan en iyi eyleme göre tepki verir. IBM'in Deep Blue sistemi ve Google'ın AlphaGo'su reaktif makinelere örnektir.

Sınırlı Hafıza: Sınırlı hafızalı makineler geçmiş deneyimleri veya bazı verileri kısa bir süre için saklayabilir. Kendi kendine giden arabalar sınırlı hafıza sistemlerinin en iyi örneklerinden biridir. Bu arabalar, yolda gezinmek için yakındaki arabaların son hızını, diğer arabaların mesafesini, hız sınırını ve diğer bilgileri saklayabilir.

Zihin Teorisi: Bu teoriye göre makinelerin insan duygularını, hislerini, inançlarını anlayıp insanlar gibi sosyal olarak etkileşime girebilmesi beklenmektedir. Bu tipteki AI makineleri hala geliştirilebilmiş değildir, ancak araştırmacılar bu tür AI makinelerini geliştirmek için çalışmalar yapılmaktadır.

Kişisel Farkındalık: Yapay Zekânın geleceğidir. Bu makinelerın çok zeki olmaları ve kendi duygu, düşünce ve öz farkındalıklarına sahip olmaları beklenmektedir. Kişisel Farkındalık Yapay Zekâ (AI) 'sı şu anda mevcut olmamakla birlikte varsayımsal bir kavramdır (Avijeet Biswal 2021).



Şekil 1. Yapay zekâ türleri

Makine Öğrenmesi (ML)

Cihazlara belirli bir veriden öğrenme yeteneği kazandırmak ve insan zekâsını simüle eden algoritmaları kullanarak zaman içinde doğruluklarını artırmak amacıyla uygulamalar oluşturmaya odaklanan bir yapay zekâ (AI) dalıdır. Ayrıca, makinelerin geçmişte

gözlemlediklerine ve öğrendiklerine dayanarak gelecek hakkında tahminlerde bulunabilmesi beklenmektedir (IBM Cloud Education 2020).

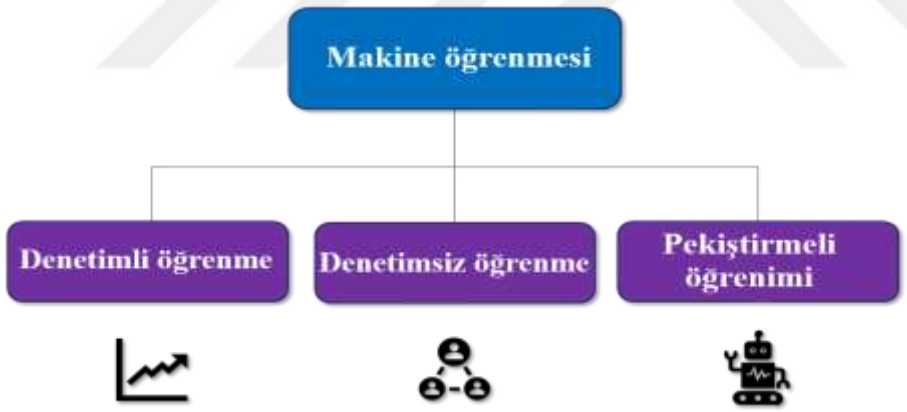
Her öğrenme modeli üç bileşen içermektedir:

- Eğitim seti: Sınıflandırıcı parametrelerinin bulunmasını sağlar ve öğrenme bu set ile gerçekleştirir.
- Doğrulama seti: Eğitim aşamasında modelin performansını değerlendirmek için kullanılır. Ayrıca en uygun sınıflandırıcı parametreleri bu veri seti ile ayarlanır.
- Test seti: Bir sınıflandırıcının performansını değerlendirmek için kullanılan veri setinin bir bölümüdür.

Üç tür makine eğimi vardır:

- Denetimli öğrenme
- Denetimsiz öğrenme
- Pekiştirmeli öğrenimi

Şekil 2'de öğrenme türleri gösterilmiştir.



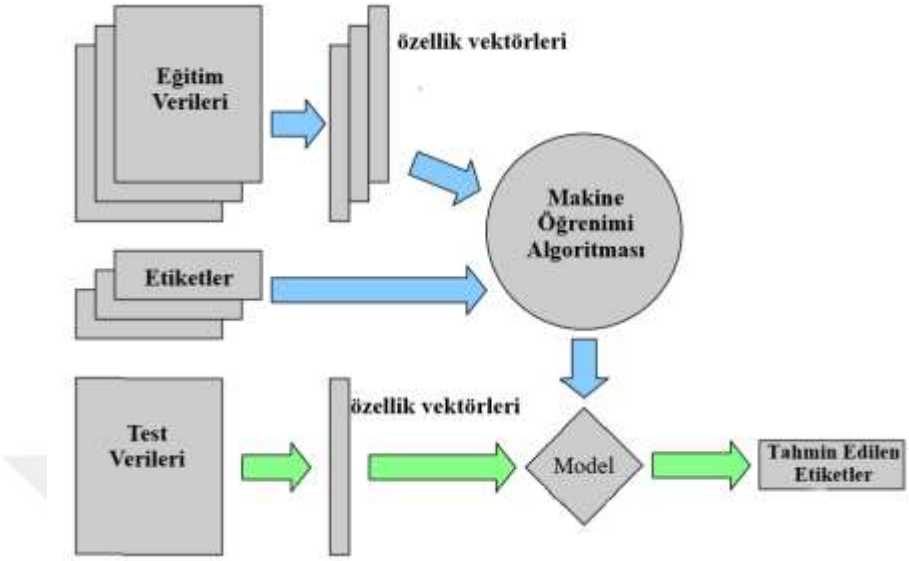
Şekil 2. Makine öğrenmesi türleri (Piyush & Rishabh 2020)

Denetimli öğrenme:

Bu öğrenme yönteminde veri kümeleri etiket bilgileri ile birlikte kullanılmaktadır. Şekil 3'te denetimli öğrenme mekanizması gösterilmiştir. Eğitim verilerinin etiket bilgileri ile birlikte algoritmaya girmesi sağlanır. Böylece oluşturulan modele test verileri girdi olarak verildiğinde tahmin edilen etiket çıktıları elde edilmektedir (Ed Burns 2021).

Örneğin, e-posta ve istenmeyen e-postaların (spam) sınıflandırılmasında, geçmiş spam ve e-postaları girdi olarak alır. Böylece, istenmeyen postaları sınıflandırmak için önceki

verilerden faydalanılır. Hisse Senedi Fiyat Tahmininde (Regresyon) geçmiş iş piyasası verileri bu yöntemde algoritmayı besler ve uygun regresyon analizi ile gelecek için yeni fiyat tahmin edilir *(*Piyush & Rishabh 2020).

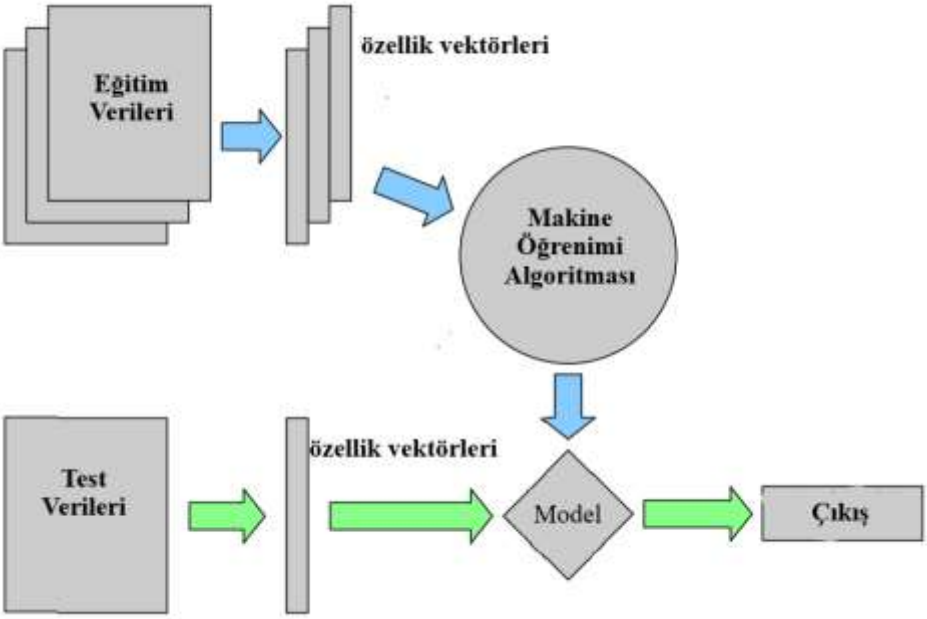


Şekil 3. Denetimli öğrenme blok diyagramı

Denetimsiz öğrenme:

Bu tür makine öğrenimi, etiketlenmemiş veriler üzerinde çalışan algoritmaları içerir. Algoritma, anlamlı bir bağlantısı olan veri kümelerini tarar. Algoritmaların eğitim verileri, tahminleri ve önerileri önceden belirlenir.

Benzer özelliklere sahip verilerin algoritmaya göre gruplandırılması istenir. Bu gruplara küme adı verilir ve yapılan işlem de kümeleme olarak adlandırılır. Perakende analitiğinde, çeşitli müşteriler genellikle satın alma ve diğer davranışlarına göre kümelenir (Fischer & Winkler 2020). Şekil 4'te denetimsiz öğrenme akış diyagramı gösterilmektedir.



Şekil 4. Denetimsiz öğrenme blok diyagramı

Pekiştirmeli öğrenme:

Bu türde makine öğrenimi modelleri, eylemleri için aldıkları ödüllere ve geri bildirimlere dayalı olarak bir dizi karar vermek üzere eğitilir. Makine, karmaşık ve belirsiz durumlarda bir hedefe ulaşmayı öğrenir ve öğrenme döneminde bunu her gerçekleştirdiğinde ödüllendirilir (Vaishali Advani 2021).

Pekiştirmeli öğrenmeye bilgisyarların video oyunlarını kendi kendilerine oyanamalarını öğrenmeleri örnek olarak verilebilir. Algoritma, bir dizi eylem yoluyla oyun ortamı ile etkileşime girer. Bu ortam, sırayla, yapılan eylemin niteliğine göre bir ödül veya ceza verir.

Pekiştirmeli öğrenmede kullanılan bazı önemli terimler aşağıdaki gibidir:

- **Ajan (Agent):** Ödül kazanmak için çevreyle etkileşen ve ödülleri maksimuma çıkartarak optimum policy'i (hareket tarzı) bulan bir öğrenen etkendir.
- **Çevre (Environment):** Çevre, ajanın kendisiyle etkileşime geçmek için uygun yolu bulmaya çalıştığı bir görev veya simülasyondur.
- **Ödül (Reward):** Bir ajana belirli bir eylemi veya görevi yerine getirdiğinde çevre tarafından verilen anında geri dönüştür (pozitif veya negatif olabilir).
- **Durum (State):** Ajanın bulunduğu durumu ifade eder.
- **Politika (Policy):** Ajan'nın herhangi bir zamana kadar olan davranışıdır. Mevcut duruma göre bir sonraki eyleme karar vermek için ajan tarafından uygulanan bir stratejidir.
- **Geri Dönüş (Return):** Herhangi bir action'ı gerçekleştirdikten sonra çevreden alınan ödül formudur.

- **Değer Fonksiyonu (Value Function) :** Bir ajan için bir durumun ne kadar iyi olduğunu temsil eder.

Şekil 5. Pekiştirmeli öğrenme

Farklı pekiştirmeli öğrenme türleri vardır; Q-öğrenme, SARSA (Durum-eylem-ödül-durum-eylem), DQN (derin Q-öğrenme ağı) ve DDPG (Derin Belirleyici Politika Gradyanı). Bu çalışmada Q-öğrenme yönteminin bir uzantısı olarak kabul edilen DQN yöntemi kullanılmıştır.

Derin Öğrenme

Derin sinir ağı olarak da bilinen, yapay sinir ağları (ANN) adı verilen beynin yapısından ve işlevinden esinlenen algoritmalarla ilgilenen bir makine öğrenimi alt alanıdır. Genellenebilirlikleri ve herhangi bir sürekli gerçek değerli fonksiyona yaklaşma yetenekleri mevcuttur. Mimariye göre, ANN'ler kabaca iki ana kategoriye ayrılır:

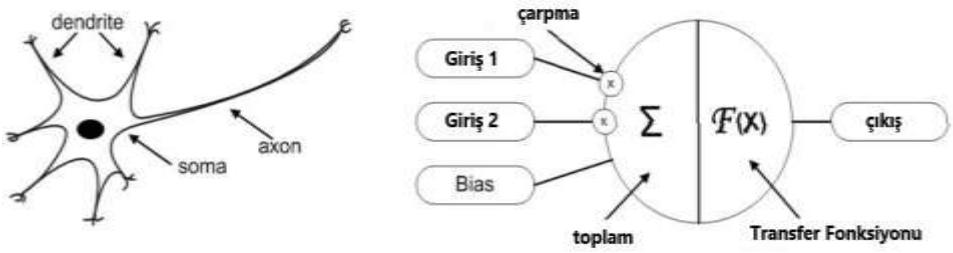
- **Feedforward Ağı:** Düğümler arasındaki bağlantıların döngü oluşturmadığı yapay bir sinir ağıdır. Tek bir girdi alır. Örneğin oyun durumunun bir temsilini ve her olası eylem için olasılık değerlerini çıkarır.
- **Tekrarlayan Sinir Ağları (RNN'ler):** Önceki adımın çıktısının mevcut adıma girdi olarak beslendiği bir ağ türüdür. Genel olarak, RNN'ler, ağın çıkışının ağın girişine bağlı olabileceği zaman serilerine uygulanır (Jason Brownlee 2019).

RNN'ler eğitim sürecinde, ağın önceki çıktısının bir sonraki ağa giriş olarak verilmesi yönüyle ileri beslemeli ağlara benzer. Örneğin, bir oyunda tek bir gözlemin tüm oyun durumunu temsil etmediği durumlarda her bir katman çıktısı diğer katmanın ne yapacağı hakkında bilgi vererek oyunun ilerlemesi sağlanır.

Yapay sinir ağlarının ana bileşenleri vardır:

Yapay Nöron: Verileri işleyebilen biyolojik bir nöronunun matematiksel modeli yapay sinir ağının temel yapı taşı olan yapay nöronun oluşturulmasında örnek olmuştur. Yapay

nöronların tasarımı ve işlevsellikleri, beyin, omurilik ve periferik gangliyonları içeren biyolojik sinir ağlarının (sistemlerin) temel yapı taşı olan biyolojik bir nöronun gözlemlenmesi doğrultusunda elde edilir. Tasarım ve işlevselliklerindeki benzerlikler Şekil 6'da gösterilmektedir.



Şekil 6. Biyolojik ve Yapay Nöron yapısının şematik gösterimi (Andrej Krenker1 *et. el.* 2011)

Yapay nöronların elde edilmesindeki matematiksel süreçler aşağıdaki gibidir:

- Öncelikle bilgi, ağırlıklı girdiler aracılığıyla yapay bir nöronun gövdesine gelir (her girdi ayrı ayrı bir ağırlık ile çarpılır).
- Daha sonra, yapay bir nöronun gövdesi ağırlıklı girdileri toplar ve toplamı bir transfer fonksiyonu ile işler.
- Son olarak, yapay bir nöron, işlenen bilgiyi çıktı (lar) yoluyla iletir.

Yapay nöron modelinin matematiksel açıklaması aşağıdaki gibidir:

$$y_i = f\left(\sum_{i=0}^{m} w_i \cdot x_i + b_i\right) \quad (1)$$

- x_i , giriş değeridir.
- w_i , ağırlık değeridir.
- b_i , bias'dır.
- $f\ (.)$, bir transfer fonksiyonudur.
- y_i, çıkış değeridir.

Aktivasyon Fonksiyonları:

Aktivasyon Fonksiyonları özel olarak yapay sinir ağlarında, bir giriş sinyalini sırayla bir sonraki katmana girdi olarak verip çıkış sinyaline dönüştürmek için kullanılmaktadır. Doğrusal (Linear), Sigmoid, Tanh, ReLU ve Leaky ReLU gibi farklı aktivasyon fonksiyon türleri vardır (Sharma, *et al* ,2020).

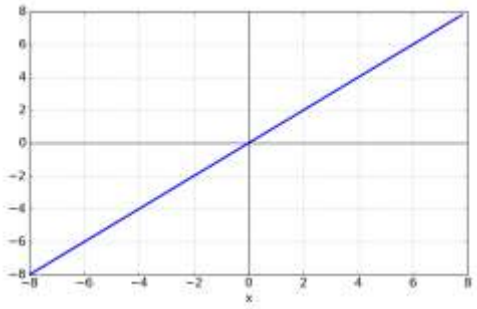
Doğrusal (Linear) Aktivasyon Fonksiyonu:

Doğrusal aktivasyon fonksiyonu, girişle doğru orantılıdır. Matematiksel ifadesi denklem 2'de verilmişir. Grafiksel olarak Şekil 7'de gösterilmektedir.

Sigmoid Aktivasyon Fonksiyonu:

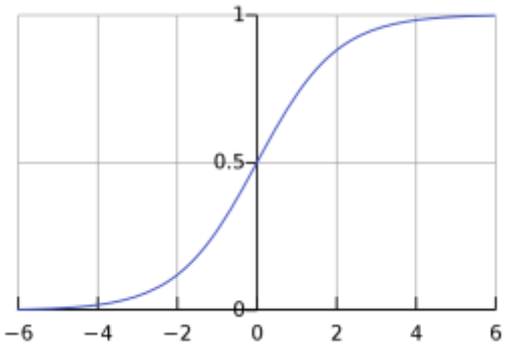
Sigmoid aktivasyon fonksiyonu, doğrusal olmayan bir fonksiyon olduğu için en yaygın kullanılan aktivasyon fonksiyonudur. Sigmoid fonksiyonu girilen değerleri 0 ile 1 aralığındaki değerlere dönüştürür. Matematiksel ifadesi denklem 3'te verilmişir. Grafiksel olarak Şekil 8'de gösterilmektedir.

$$f(x) = x \tag{2}$$



Şekil 7. Doğrusal aktivasyon fonksiyonu

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3}$$



Şekil 8. Sigmoid aktivasyon fonksiyonu

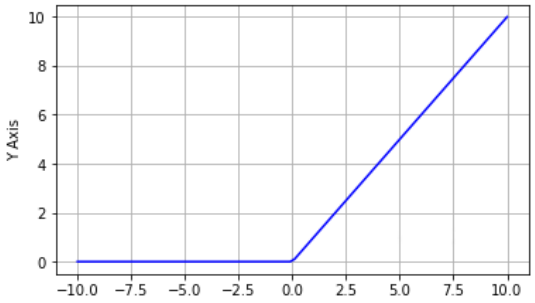
ReLU (Rectified Linear Unit) Fonksiyonu:

ReLU, doğrultulmuş doğrusal birim anlamına gelir ve sinir ağlarında yaygın olarak kullanılan doğrusal olmayan bir etkinleştirme işlevidir. ReLU fonkiyonunu kullanmanın avantajı, tüm nöronların aynı anda etkinleştirilmemesidir. Bu, bir nöronun yalnızca doğrusal dönüşümünün çıktısı sıfır olduğunda devre dışı bırakılacağı anlamına gelir. Matematiksel ifadesi denklem 4'te verilmişir. Grafiksel olarak Şekil 9'da gösterilmektedir.

Leaky Relu Fonksiyonu:

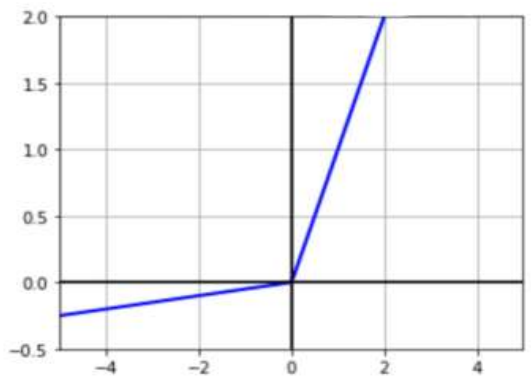
Leaky ReLU, x'in negatif değerleri için ReLU işlevlerinin değerini sıfır olarak tanımlamak yerine, x'in çok küçük doğrusal bileşeni olarak tanımlandığı ReLU işlevinin geliştirilmiş bir versiyonudur. Matematiksel ifadesi denklem 5'te verilmişir. Grafiksel olarak Şekil 10'da gösterilmektedir.

$$f(x) = \max(0, x) \tag{4}$$



Şekil 9. ReLU aktivasyon fonksiyonu

$$f(x) = \begin{cases} 0.01x & x < 0 \\ x & x > 0 \end{cases} \tag{5}$$

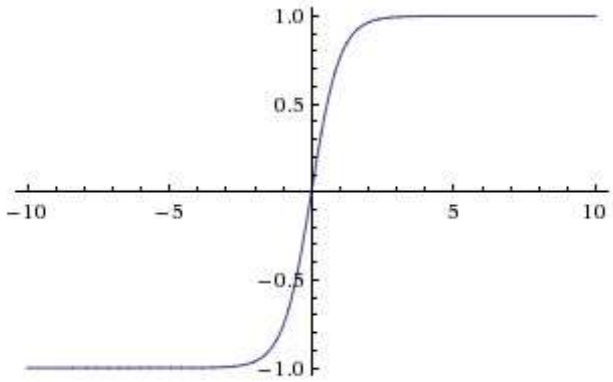


Şekil 10. Leaky ReLU aktivasyon fonksiyonu

Tanh Fonksiyonu:

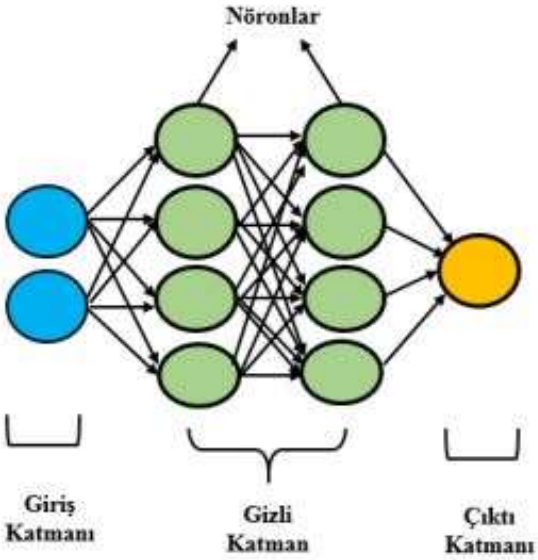
Hiperbolik tanjant işlevidir. Tanh işlevi sigmoid işlevine benzer ve orijine göre simetriktir. [-1,1] aralığında çıktı üreten doğrusal olmayan bir fonksiyondur. Matematiksel ifadesi denklem 6 ‘da verilmişir. Grafiksel olarak Şekil 11’de gösterilmektedir.

$$f(x) = \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{6}$$



Şekil 11. Tanh aktivasyon fonksiyonu

Sinir ağları katmanlar halinde tasarlanılır. Katmanlar bir 'aktivasyon işlevi' içeren birbirine bağlı bir dizi nörondan oluşur. Modeller, giriş katmanıyla bağlantılı bir veya daha fazla gizli katmandan oluşabilmektedir. Giriş katmanında verileri işleme, ağırlıklı bir bağlantı sistemi aracılığıyla yapılır. Benzer şekilde gizli katmanlar, Şekil 12’de gösterildiği gibi ard arda bağlanarak cevabın elde edileceği bir çıktı katmanına bağlanır.



Şekil 12. Temel sinir ağlari katmanları

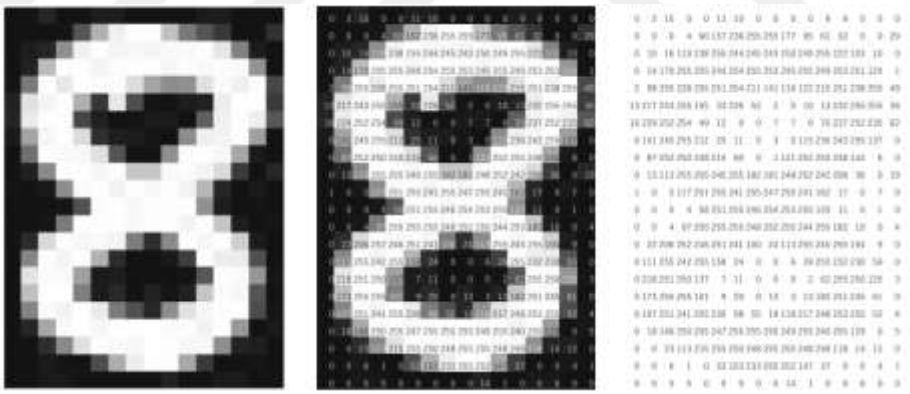
KURAMSAL TEMELLER

Evrişimli Sinir Ağı (CNN)

Evrişimli Sinir Ağı: Görüntü, ses gibi girdi alabilen, girdideki çeşitli yönlere/nesnelere önem (öğrenilebilir ağırlıklar ve önyargılar) atayabilen bir ızgara modeline sahip bir derin öğrenme modeli türü ve algoritmasıdır. CNN, tipik olarak üç tür katmandan (veya yapı taşı) oluşan matematiksel bir yapıdır:

* Evrişim Katmanı
* Havuzlama Katmanı
* Tamamen Bağlı Katmanlar.

Bilgisayarlarda bir görüntü piksel olarak görülmekte ve görüntü 0 ile 255 arasında bir sayı dizisi olarak temsil edilmektedir. Şekil 13'te bir görüntü (sağda) ve bu görüntünün bilgisayarda okunması ile edilen görüntü (solda) gösterilmiştir. Burada gösterilen sayıların her biri pikselin parlaklığını ifade eder.



Şekil 13. Görüntünün bilgisayardaki içeriği

Evrişim Katmanı:

Evrişim katmanı, özellik çıkarımını gerçekleştiren CNN mimarisinin temel bir bileşenidir. Tipik olarak, evrişim işlemi ve aktivasyon işlevi gibi doğrusal ve doğrusal olmayan işlemlerin bir kombinasyonundan oluşur.

Evrişim İşlemi:

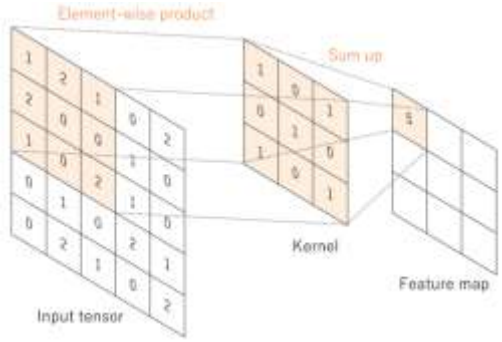
Evrişim, çekirdek adı verilen küçük bir sayı dizisinin, tensör adı verilen bir sayı dizisi olan girişe uygulandığı, özellik çıkarımı için kullanılan özel bir doğrusal işlem türüdür.

Çekirdeğin her bir öğesi ile giriş tensörü arasındaki öğe bazlı bir ürün, tensörün her konumunda hesaplanır ve çıkış tensörüne karşılık gelen yerde, özellik haritası adı verilen çıktı değerini elde etmek için toplanır.

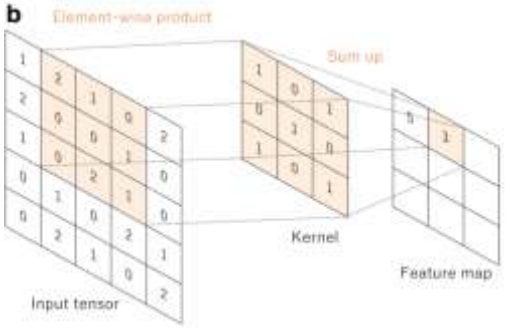
Çekirdeğin bir girdi görüntüsündeki hareketi, piksellerin girdi matrisi üzerindeki kayma sayısı olan Evrişim Katmanı parametresi 'Stride' tarafından gerçekleştirilir. Örneğin; Stride 1 olduğunda, çekirdek bir seferde 1 piksele taşınır. Stride 2 olduğunda, çekirdek bir seferde 2 piksele taşınır ve bu şekilde devam eder.

Şekil 14 çekirdek boyutu 3 × 3 olan, dolgu içermeyen ve adımı 1 olan evrişim işlemine örnektir. Giriş tensörüne bir çekirdek uygulanır ve her bir öğenin arasına öğe bazlı bir ürün uygulanır. Çekirdek ve giriş tensörü her konumda hesaplanır ve öznitelik haritası adı verilen çıkış tensörüne karşılık gelen konumdaki çıkış değerini elde etmek için toplanır. Şekil 14'de Evrişim katmanlarındaki çekirdeklerin bir giriş tensöründen özellikleri nasıl çıkardığına dair örnekler gösterilmektedir. Çoklu çekirdek, yatay kenar algılayıcı (üst), dikey kenar algılayıcı (orta) ve anahat algılayıcı (alt) gibi farklı özellik çıkarıcılar olarak çalışır. Soldaki görüntü bir girdi, ortadaki çekirdekler ve sağdaki görüntü ise çıktı özelliği haritalarıdır.

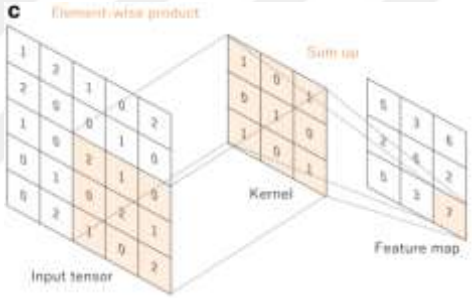
Ancak, evrişimli katmanları uygularken görüntü çevresindeki pikselleri kaybetme sorunu vardır. Bu, özellikle görüntünün çevresinde temel özelliklere sahip görüntüler söz konusu olduğunda bazı görüntü özelliklerini kaybetme anlamına gelir. Bu sorun Padding tekniği ile çözülebilir; burada tipik olarak sıfır olan satırlar ve sütunlar, en dıştaki öğeye bir çekirdeğin merkezini sığdırmak ve aynı tutmak için evrişim işlemi yoluyla giriş tensörünün her iki tarafına eklenir.



a



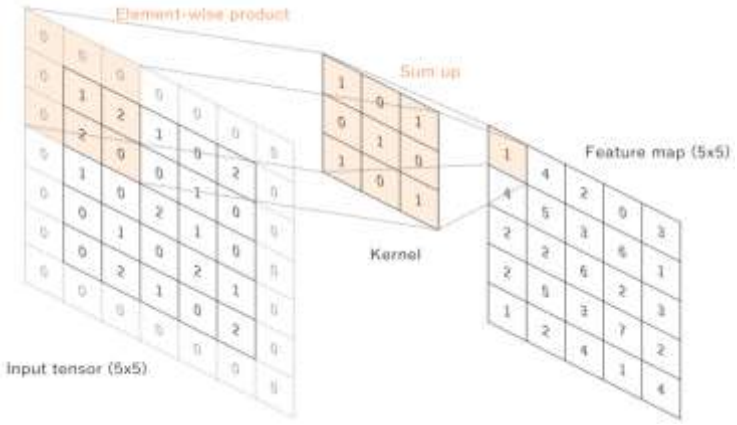
b



c

Şekil 14. Evrişim işlemi (Yamashita *et al*. 2018)

Şekil 15'te, Düzlem içi boyutları korumak için sıfır dolgulu bir evrişim işlemi gösterilmiştir. Çıktı özelliği haritasında 5 × 5 giriş boyutu tutulmuştur. Bu örnekte, bir çekirdek boyutu 3x3 ve adım 1 olarak ayarlanmıştır (Yamashita *et al*., 2018).



Şekil 15. Padding işlemi (Yamashita *et al*. 2018)

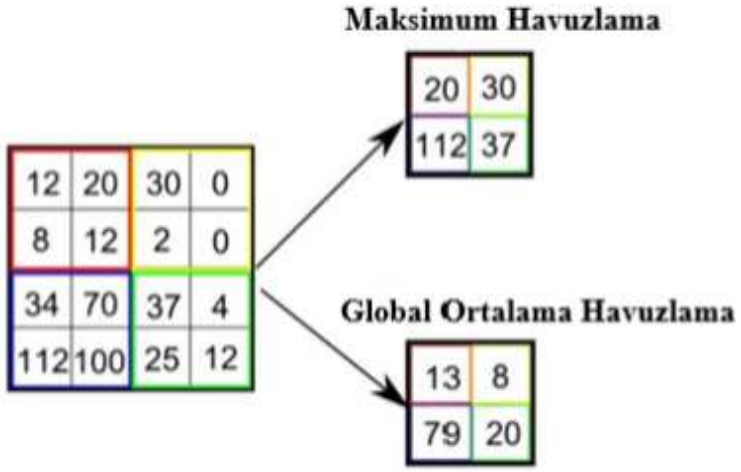
Aktivasyon Fonksiyonu:

Evrişim gibi doğrusal bir işlemin çıktıları daha sonra doğrusal olmayan bir aktivasyon fonksiyonundan geçirilir. Sigmoid veya hiperbolik tanjant (tanh) işlevi gibi doğrusal olmayan işlevler daha önce biyolojik bir nöron davranışının matematiksel temsilleri oldukları için kullanılmış olsa da, şu anda kullanılan en yaygın doğrusal olmayan aktivasyon işlevi düzeltilmiş doğrusal birimdir (ReLU).

Havuzlama Katmanı:

Bir havuzlama katmanı, küçük kaymalara ve bozulmalara bir öteleme değişmezliği katmak ve sonraki öğrenilebilir parametrelerin sayısını azaltmak için özellik haritalarının düzlem içi boyutluluğunu azaltan tipik bir alt örnekleme işlemi sağlar. Iki havuzlama katmanı türü vardır: Maksimum havuzlama ve global ortalama havuzlama.

Maksimum havuzlama, giriş özelliği haritalarından pencereler çıkaran, her pencerede maksimum değeri veren ve diğer tüm değerleri atan en popüler havuzlama işlemidir. Diğer bir havuzlama işlemi de, global ortalama havuzlamadır. Global ortalama havuzlama, en uç boyutta bir alt örnekleme gerçekleştirir; burada yükseklik × genişlik boyutuna sahip bir özellik haritasının, her özellik haritasındaki tüm öğelerinin ortalamasını alarak 1 × 1 diziye alt örneklemesi yapılırken, özellik haritalarının derinliği korunur. Şekil 16'da havuz katmanı gösterilmektedir.



Şekil 16. Havuzlama katmanı

Tam Bağlantılı Katman (Fully Connected Layer):

Bu aşamada son evrişim veya havuzlama katmanının çıktı öznitelikleri düzleştirilir. Yani tek boyutlu (1D) bir sayı dizisine veya vektöre dönüştürülür. Bu katmanlar, her girişin her

çıkışa öğrenilebilir bir ağırlık değeri ile bağlandığı, yoğun katmanlar olarak da bilinen bir veya daha fazla tamamen bağlı katmandan oluşabilir.

Son Katman Aktivasyon İşlevi (Last Layer Activation Function):

Tamamen bağlı son katmana uygulanan aktivasyon işlevi, genellikle diğerlerinden farklıdır. Her göreve bağlı olarak uygun bir etkinleştirme işlevinin seçilmesi gerekir. Çok sınıflı sınıflandırma görevine uygulanan bir etkinleştirme işlevi, son tam bağlı katmandan çıktı gerçek değerlerini, her bir değerin 0 ile 1 arasında olduğu ve tüm değerlerin toplamının 1 olduğu hedef sınıf olasılıklarına normalleştiren bir softmax işlevidir.

Ağın Eğitilmesi:

Ağın eğitilmesi, evrişim katmanlarında çekirdekleri bulma işlemidir. Bir eğitim veri kümesinde çıktı tahminleri ile verilen temel doğruluk etiketleri arasındaki farklılıkları en aza indiren tamamen bağlantılı katmanlardaki ağırlıklar bulunur. Geri yayılım algoritması, kayıp işlevi ve gradyan iniş optimizasyon algoritmasının temel görevler üstlendiği sinir ağlarını eğitmek için yaygın olarak kullanılan bir yöntemdir. Belirli çekirdekler ve ağırlıklar altındaki bir model performansı, bir eğitim veri setinde ileri yayılma yoluyla bir kayıp fonksiyonu tarafından hesaplanır ve öğrenilebilir parametreler, yani çekirdekler ve ağırlıklar, geri yayılım ve gradyan inişi adı verilen bir optimizasyon algoritması aracılığıyla kayıp değerine göre güncellenir.

Kayıp Fonksiyonu:

Maliyet işlevi olarak da adlandırılan bir kayıp işlevi, ileriye doğru yayılma ve verilen kesin referans etiketleri aracılığıyla ağın çıktı tahminleri arasındaki uyumluluğunu ölçer. Çok sınıflı sınıflandırma için yaygın olarak kullanılan kayıp fonksiyonu çapraz entropidir. Çapraz entropi bir tür kayıp işlevi olan hiper parametrelerden biridir ve verilen görevlere göre belirlenmesi gerekir.

Kademeli Alçalma:

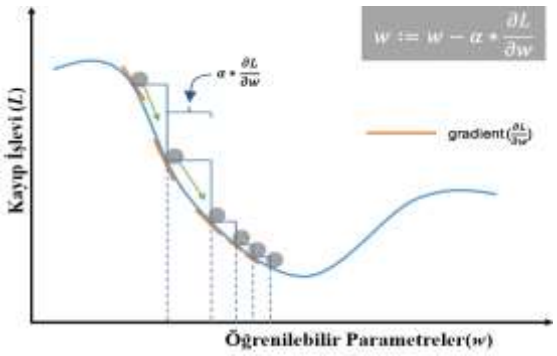
Gradyan inişi genellikle kaybı en aza indirmek için ağın öğrenilebilir parametrelerini, yani çekirdek ve ağırlıkları yinelemeli olarak güncelleyen bir optimizasyon algoritması olarak kullanılır. Kayıp fonksiyonunun gradyanı fonksiyonun en yüksek artış hızına sahip olduğu yönü sağlar ve her öğrenilebilir parametre, öğrenme hızı adı verilen bir hiper parametreye göre belirlenen rastgele bir adım boyutu ile gradyanın negatif yönünde güncellenir. Matematiksel olarak gradyan, öğrenilebilir her parametreye göre kaybın kısmi bir türevidir ve bir parametrenin tek bir güncellemesi aşağıdaki formülle ifade edilir:

$$w := w - \propto \frac{\partial L}{\partial w} \qquad (7)$$

• w: öğrenilebilir parameterler,

• α: öğrenme oranı,

• L: kayıp işlevini temsil etmektedir.



Şekil 17. Kademeli azalma (Yamashita *et al*., 2018)

Veriler ve Kesin Referans Etiketleri:

Veriler ve kesin referans etiketleri, derin öğrenme veya diğer makine öğrenimi yöntemlerini uygulayan araştırmalardaki en önemli bileşenlerdir. Mevcut veriler eğitim, doğrulama ve test seti olarak üç gruba ayrılır. Ancak çapraz doğrulama gibi bazı değişkenler vardır. Kayıp değerlerinin ileriye doğru yayılma yoluyla hesaplandığı ve öğrenilebilir parametrelerin geri yayılım yoluyla güncellendiği bir ağı eğitmek için eğitim seti kullanılır. Eğitim süreci sırasında modeli değerlendirmek, hiper parametrelere ince ayar yapmak ve model seçimini gerçekleştirmek için bir doğrulama seti kullanılır. Eğitim sürecinde ince ayar yapılan ve seçilen nihai modelin performansını eğitim ve doğrulama setleriyle değerlendirmek için ideal olarak çalışmanın en sonunda yalnızca bir kez bir test seti kullanılır.

Son olarak, evrişimli sinir ağının evrişim katmanı, havuzlama katmanı ve tam bağlantılı katman olmak üzere üç katmanı vardır. Her katmanda parametreler ve hiperparametre bulunur. Parametre, eğitim süreci sırasında otomatik olarak optimize edilen bir değişken iken hiperparametre, önceden ayarlanması gereken bir değişkendir.

Evrişimli sinir ağındaki parametrelerin ve hiperparametrelerin bir listesi Tablo 1'de listelenmiştir:

Tablo 1. Parametrelerin ve Hiperparametrelerin Listesi (Yamashita, Rikiya, et al 2018)

	Parametre	**Hiperparametre**
Evrişim katmanı	Çekirdek	Çekirdek boyutu, çekirdek sayısı, Stride, Padding, Aktivasyon Fonksiyonu.
Havuzlama katmanı	-	Havuzlama yöntemi, filtre boyutu, Padding
Tam bağlantılı katman	Ağırlıklar	Ağırlık sayısı, aktivasyon fonksiyonu
Diğerleri		Model mimarisi, optimize edici, öğrenme hızı, kayıp işlevi, mini batch boyutu, dönemler, düzenlileştirme, ağırlık başlatma, veri kümesi bölme

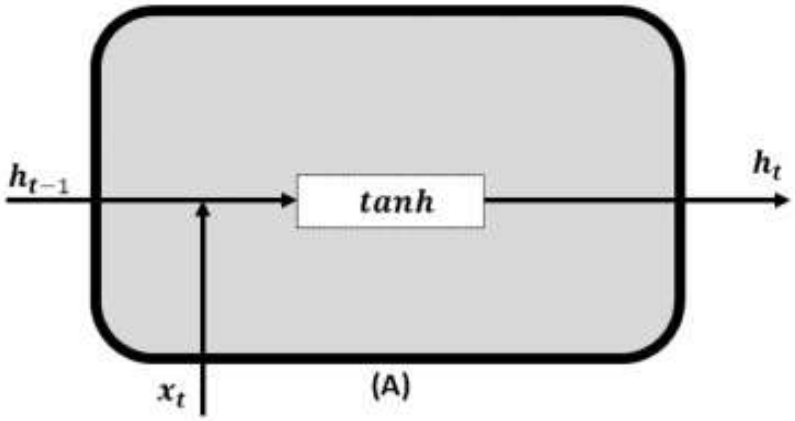
Yinelenen Sinir Ağı (RNN)

Yinelenen sinir ağı bir tür yapay sinir ağıdır. Ancak gizli katmanlarında yinelenen bağlantılara sahiptir. Bu ağ eğitim sürecinde zaman serisi verilerini veya sıralı verileri kullanır. Bu algoritmalar genellikle dil çevirisi, doğal dil işleme (NLP), konuşma tanıma gibi zaman serisi problemlerinde kullanılır.

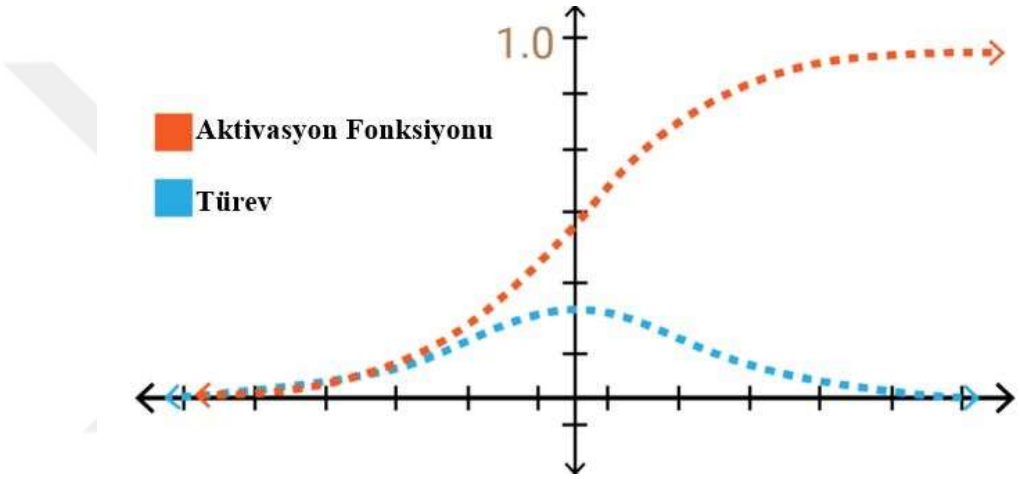
Basit Tekrarlayan Ağ (SRN)

SRN, ilk olarak Elman (1990) tarafından tanıtılan, giriş katmanı, gizli katman ve çıkış katmanlarından oluşan en basit yinelenen sinir ağı türüdür. Geleneksel sinir ağlarıyla karşılaştırıldığında SRN, yeni bir katman oluşturan gizli katmandan ek girdilere sahiptir (Şekil 18). Bununla birlikte, SRN, kaybolan gradyan problemi nedeniyle birçok sınıflandırma adımında bilgiyi entegre etmekte zorlanır.

Kaybolan gradyan sorunu, gradyan tabanlı yöntemlerle belirli Yapay Sinir Ağlarını eğitmede bulunan bir zorluktur. Sigmoid fonksiyonunun grafiği Şekil 19'da gösterilmiştir. Sigmoid fonksiyonu için çok büyük değerlerde türevin çok düşük değer aldığı görülmektedir. Sinir ağının birçok gizli katmanı varsa, her katmanın türevlerini çarptığımızda önceki katmanlardaki gradyanlar çok düşük olacaktır. Sonuç olarak, önceki katmanlarda öğrenme çok yavaş hale gelmektedir.



Şekil 18. Basit tekrarlayan ağ hücresi (ArunKumar *et al.*, 2021)



Şekil 19. Sigmoid aktivasyon fonksiyonu ve türevi (ArunKumar *et al.*, 2021)

Girdi x = (x_1, x_2, · · ·, x_t) ve çıktı y = (y_1, y_2, . . . y_t) olarak bir dizi verildiğinde, RNN'nin tekrarlayan gizli durumu h_t'yi şu şekilde güncellediği varsayılırsa:

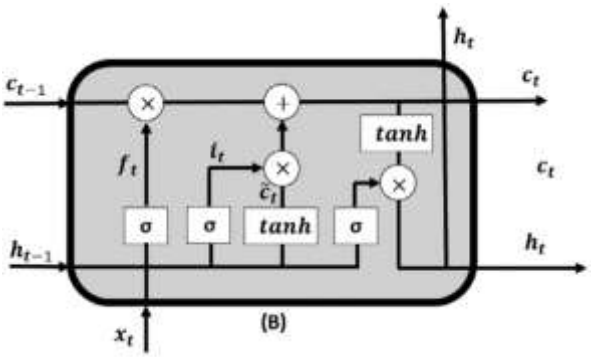
$$h_t = \begin{cases} 0 & t = 0 \\ g(Wx_t + Uh_{t-1} + b) \end{cases} \quad (8)$$

Burada, afin dönüşümü lojistik sigmoid fonksiyonundan geçirilerek doğrusal olmayan bir fonksiyon haline getirilir.

Uzun Kısa Süreli Bellek Ağları (LSTM)

LSTM, bilginin bir zaman adımından diğerine nasıl geçeceğini belirleyen karmaşık bir geçit mekanizmasına sahip bir tekrarlayan ağdır. Başlangıçta, bu geçit mekanizması, SRN'lerdeki kaybolan gradyan problemini çözmek veya eşdeğer olarak, ağın uzun vadeli bağımlılıkları hatırlamasını kolaylaştırmak için tasarlanmıştır.

Hochreiter ve Schmidhuber, kaybolan gradyan probleminin üstesinden gelmek için LSTM'yi önermişlerdir. LSTM hücresinin genel mimarisi Şekil 20 'de gösterilmiştir.



Şekil 20. LSTM hücresinin genel mimarisi (ArunKumar *et al.*, 2021)

LSTM hücresi, unutma kapısı (forget gate), çıkış geçidi, giriş geçidi ve güncelleme kapılarından oluşur. Adından da anlaşılacağı gibi, unutma kapısı önceki bellek birimlerinden neyi unutacağına karar verir, giriş kapısı nöronun neyi kabul edeceğine karar verir, güncelleme geçidi hücreyi günceller ve çıkış kapısı ise yeni uzun süreli belleği oluşturur (ArunKumar *et al.*, 2021).

LSTM adımları aşağıdaki gibi çalışır:

• Giriş kapısı matematiksel olarak aşağıdaki gibi temsil edilir:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{9}$$

- Önceki bellekten unutulacak bilgi, matematiksel olarak denklem 10'daki gibi tanımlanan unutma kapısı tarafından kontrol edilir:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_c) \tag{10}$$

- Hücre durumu, aşağıdaki denklemlerle matematiksel olarak ifade edilen güncelleme geçidi tarafından güncellenir:

$$C_t^* = tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{11}$$

$$C_t = f_t * C_{t-1} + i_t * C_t^* \tag{12}$$

- Önceki zaman adımının gizli katmanı, çıkış geçidi tarafından güncellenir ve çıkış kapısı tarafından verildiği gibi, çıktının güncellenmesinden de sorumludur:

$$O_t = \sigma(W_o * x_t + U_o * h_{t-1} + b_o) \tag{13}$$

$$h_t = O_t * \tanh(C_t) \tag{14}$$

Q-Öğrenme Algoritması

Q-öğrenme

Veri değişimine ve gerçek zamanlı planlamaya dayalı bir algoritmadır. Q-öğrenme algoritması, herhangi bir t anında belirli bir durum için bir eylem gerçekleştirerek elde edilebilecek beklenen ödülleri ölçmek üzere bir Q işlevi oluşturmak için Bellman Denklemini kullanır. Matematiksel olarak şu şekilde yazılabilir:

$$Q(s_t, a_t) = E[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots | (s_t, a_t)] \quad (15)$$

s_t: t anındaki durum,

a_t: t anındaki eylem

E: beklenen değer

R_{t+1}: t+1 anındaki ödül

ve γ indirim faktörüdür.

Amaç, tüm olası durum ve eylem kombinasyonlarını keşfederek bu Q işlevi değerlerini yinelemeli bir süreçle güncellemektir. Yöntem, olası tüm durum ve eylem seçenekleri kombinasyonları için ödülleri depolayan bir Q tablosu oluşturmayı içerir. Q-öğrenme algoritma süreci Şekil 21 'de gösterilmektedir. Bir Q öğrenmenin aşamaları aşağıdaki dört ana adım ile özetlenebilir:

1- Başlangıç değeri 0 olan Q değerleri matrisi oluşturulur. Satırların durumları, sütunların ise eylemleri temsil etmesi sağlanır.
2- Rastgele bir eylem seçilir.
3- Seçilen eylem ortama uygulanır, ajan bir sonraki duruma geçerek R ödülünü alır.
4- Aşağıdaki formülasyon kullanılarak Q tablosu güncellenir:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \propto \left[R_{t+1} + \gamma max Q(s_{t+1}, a_{t)} - Q(s_t, a_t)\right] \quad (16)$$

burada α, eğitim ilerledikçe monoton olarak 1.0'dan 0.1'e düşen öğrenme oranıdır.

5- Mevcut durum güncellenir ve bir sonraki iterasyonda yukarıdaki adımları tekrarlanır.

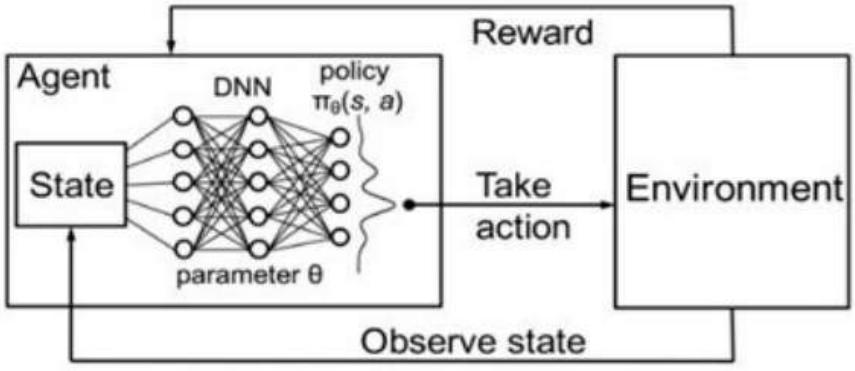
Şekil 21. Q-öğrenme algoritma süreci (Chathurangi Shyalika, 2019)

Derin Q ağı (DQN):

Giriş olarak görüntü kullanan ve çıkışında bir eylem ortaya çıkaran ilk pekiştirmeli öğrenme algoritmasıdır. Bu yöntemde ilk adımda görüntüyü işlemek için CNN kullanılır ve ardından olası eylemler için Q (s, a) hesaplanır. Böyle bit yöntemde bir ajanın temel amacı, içinde bulunduğu ortamdan ödüller toplamak ve puanını en üst düzeye çıkarmaktır.

Q-öğrenme algoritması, Q-tablosunu oluşturmak için ayrı durumlar gerektirdiğinden boyutsallık zorluğuna sahiptir. Q-öğrenmenin hesaplama karmaşıklığı, durum ve eylem vektörünün artan boyutu ile üssel olarak artar. Derin Q öğrenme algoritması, bu sorunu yapay bir sinir ağıyla yaklaşık çözer.

DQN algoritmasının genel blok diyagramı Şekil 23 'te gösterilmektedir:



Şekil 22. DQN algoritmasının genel blok diyagramı (Tan & Karakose, 2020)

(Yamini & Jain, 2020) çalışmalarında, oyun geliştiricileri tarafından şu anda AI sistemleri yapmak için kullanılan davranış ağacı modelini değiştirme potansiyeline sahip sinir ağları ve genetik algoritma kavramlarına dayalı yeni bir model tanıtmıştır. Bu model, bir

sonraki hareketi belirlemek için karmaşık iç içe geçmiş ağaç oluşturmak yerine, AI sistemlerinin oyunda eğitilmesi olanağını getirmiştir.

(Zhang, N.D., 2020) çalışmalarında, bir Snake oyunu ortamı yarattılar ve daha sonra ajanlarını aynı ortamda farklı derin öğrenme algoritması ile eğiterek, Double-DQN ve PPO'nun iki sonucunu elde ettiler. Sonuçta double-DQN yöntemi performansının daha kararlı olduğunu gözlemlediler.

(Vu & Tran, 2020), SARSA ve Q-Learning algoritmalarının değiştirilmiş varyantları tarafından eğitilen aracıların performansını uyguladı ve karşılaştırdı. Ayrıklaştırma tekniklerinin ajanın 8000 yineleme içinde daha hızlı bir şekilde yakınlaşmasına yardımcı olduğu ve 2069 gibi oldukça yüksek bir puan elde edildiği fark edildi. Ayrıca, ayrıklaştırma ile epsilon-açgözlü politikaların genellikle Q-Learning için ortalama olarak daha kötü performans gösterdiği görüldü. Son olarak, geriye doğru güncellemenin, Q değerlerinin gerçek değerlerini daha hızlı bir araya getirmesine yardımcı olduğu tespit edildi.

(Mnih & Silver, et. al.2013), pekiştirmeli öğrenme için yeni bir derin öğrenme modeli tanıttılar. Girdi olarak yalnızca ham pikselleri kullanarak Atari 2600 bilgisayar oyunları için zor kontrol politikalarında ustalaşma becerisini gösterdiler. Ayrıca RL için derin ağların eğitimini kolaylaştırmak amacıyla stokastik mini parti güncellemelerini deneyim tekrarlama belleğiyle birleştiren bir çevrimiçi Q-öğrenme çeşidi sundular. Yaklaşımları, üzerinde test edildiği yedi oyundan altısında mimari veya hiperparametrelerde herhangi bir ayarlama yapılmadan son teknoloji sonuçlar verdi.

(Dewan *et al* 2020) ajanın hedeflerine ulaşmak için sanal bir ortamda mümkün olan en iyi eylemleri öğrenmesine olanak tanıyan yapay sinir ağlarını ve takviye öğrenme mimarisini birleştiren bir yaklaşım kullandılar. Breakout, Enduro, Time Pilot vb. gibi diğer oyunlara kıyasla insanüstü bir oyun elde ettiği için Breakout'u seçmişlerdir. Bu makale, Deep Q Network (DQN) ve Double Deep Q Network (DDQN) algoritmaları arasında isabet oranlarına göre karşılaştırmalı bir analiz sağlar. DDQN'nin Breakout oyunu için daha iyi olduğunu kanıtlamışlardır.

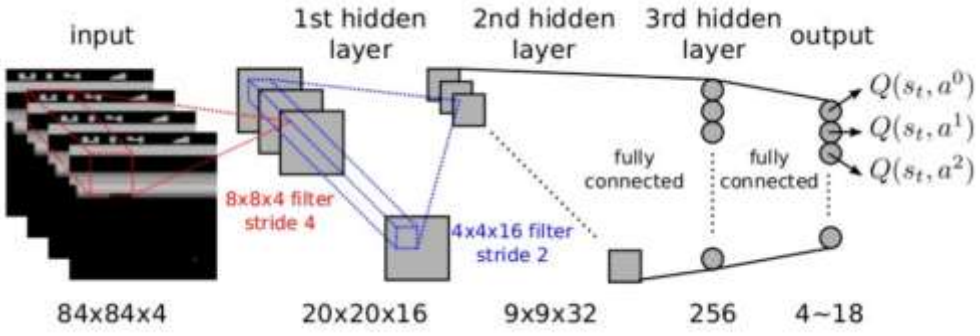
(Mnih & Silver, *et.al* 2015), derin bir Q-ağı olarak adlandırılan yeni bir yapay ajan geliştirmek için derin sinir ağlarını eğitmedeki son gelişmeleri kullanmışlardır. Doğrudan yüksek boyutlu duyusal girdilerden uçtan uca pekiştirmeli öğrenme ile başarılı politikalar elde etmişlerdir. Bu aracıyı klasik Atari 2600 oyunlarının zorlu alanında test ettiler. Girdi olarak yalnızca pikselleri ve oyun puanını alan derin Q-ağı aracısının, önceki algoritmaların performansını aşabildiğini ve 49 oyunluk bir sette insana karşı karşılaştırılabilir bir seviyeye ulaştığını gösterdiler.

(Kumar, 2020), Cart-Pole problemini çözmek için kullanılan birkaç pekiştirme öğrenme algoritmasının uygulama ayrıntılarını sağladı. Uygulama kodu Python'da yazılmıştır ve OpenAI / Gym simülasyon çerçevesini ve Keras derin öğrenme araçları kullanılmıştır. DQN'nin standart Qlearning algoritmalarına göre oldukça hızlı olduğu ve sürekli durum kullanımına izin verdiği görülmüştür. DDQN ve Dueling mimarileri, bu tür karmaşık mimarileri garanti etmek için çok basit olduğundan, DQN üzerinde önemli bir gelişme sağlamanaktadır.

Bir Derin Q ağının adımları aşağıdaki gibidir:

• Evrişimli Sinir Ağı (CNN):

Evrişim işleminin temel amacı, öznitelik haritasını giriş durumundan çıkarmaktır. Kullanılan ağ filtresi, adım uzunluğuna bağlı olarak giriş durumu üzerinde gezdirilirken, her adımda çakışan değerler birer birer çarpılarak tüm değerlerin toplamı çıktı matrisinin değeri olarak kaydedilir. İşlemin basitliği, giriş verilerinin sayısına ve formatına göre değişir. CNN'nin son nöronları, ajanın seçilmiş olduğu eylemi temsil edecektir. Şekil 22 'de DQN modelindeki Evrişimli Sinir Ağı gösterilmektedir.



Şekil 23. DQN modelinde evrişimli sinir ağı (Dewan *et al*. 2020)

• Deneyim Tekrarı

Deneyim Tekrarı, pekiştirmeli öğrenme algoritmasının öğrenebileceği ortam durumlarını (durum, eylem, ödül, sonraki_durum) saklama ve yeniden oynatma eylemidir. Deneyimler, mevcut durumu, mevcut eylemi, ödülü ve her yinelemeden sonraki durumu içeren demet (tuple) biçiminde sabit boyutlu bir tekrarlama belleğinde saklanır.

• Keşif (Exploration) : Ajanın çevreyi rastgele eylemlerle keşfettiği süreçtir.

• Faydalanma (Exploitation) : Ajanın, ağdan öngörülen eylemle çevreyi keşfetmesi sürecidir.

Genel olarak, Bir Derin Q ağının adımları aşağıdaki gibidir:

1- Deneyim tekrarı ile ortam başlatılır,

2- Ön işleme, başlangıç durumunu DQN'ye girdi olarak verilir. Bu, durumda tüm olası eylemlerin Q değerleri döndürülür.

3- Epsilon-greedy politikasını kullanarak bir eylem seçilir: epsilon olasılığı ile rastgele bir A eylemi kullanılır.

$$A = argmax(\mathrm{Q}\ (\mathrm{S}, \mathrm{A})) \tag{17}$$

Denklem 17'deki gibi maksimum Q değerine sahip bir eylem seçilir.

4- A eylemini seçtikten sonra, Ajan seçilen eylemi S durumunda gerçekleştirir ve yeni bir S durumuna geçer ve bir R ödülü alır.

5- Geçişi Deneyim Tekrarı grubunda olarak kaydeder.

6- Deneyim Tekrarından bazı rastgele geçiş grupları örneklenir, İşlev Q, formülde gösterildiği gibi beklenen gelecekteki kümülatif ağırlıklı ödül olarak tanımlanır:

$$y_t = \begin{cases} R_t \qquad terminal\ durumu\ s_t\ için \\ Q(s_t, a) = R_{t+1} + \gamma\ argmax\big(Q(s_{t+1}, a)\big)\ terminal\ olmayan\ durum\ s_t\ için \end{cases} \tag{18}$$

ve aşağıdaki formülü kullanarak kayıp hesaplanır:

$$LOSS = [y_t - \ Q(s_t, a)]^2 \tag{19}$$

7- Bu kaybı en aza indirmek için gerçek ağ parametrelerine göre gradyan inişi gerçekleştirilir.

8- Her k adımdan sonra, gerçek ağ ağırlıkları hedefe kopyalanır.

9- M bölüm sayısı için bu adımlar tekrarlanır.

MATERYAL VE METOT

Pygame kütüphanesi, video oyunları oluşturmak için tasarlanmış modüller sağlayan Python programlama dili için açık kaynaklı bir modüldür. Pek çok platformda ve işletim sisteminde çalışabilen SDL (*Simple DirectMedia Layer,* Basit DirectMedia Katmanı) geliştirme kütüphanesinin üzerine inşa edilmektedir.

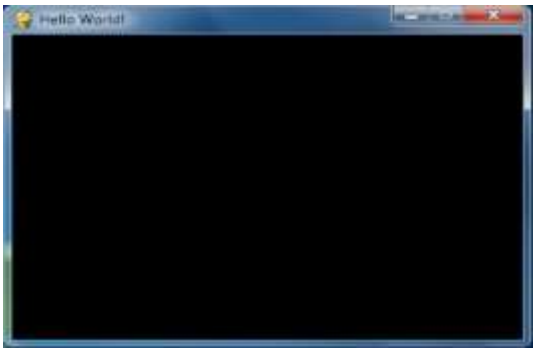
Pygame modülü kullanılarak, video ve ses ile çalışmak için gereken arka uç karmaşıklıkları hakkında oyunların mantığı ve grafikler kontrol edilebilmektedir. Pygame'in dört ana bileşeni bulunmaktadır. Bunlar;

- Çerçeve (Frame): Oyun döngüsünün bir tekrarıdır.
- Hareketli Grafik (Sprite): Herhangi bir 2 boyutlu oyun ortamında ekranda görünen tüm nesnelerdir. Hareketli grafikler canlandırılabilir, oyuncu tarafından kontrol edilebilir ve birbirleriyle iletişimle girilebilir.
- Etkinlik (Event): Kullanıcının oyunla etkileşim kurma şeklidir.
- Queue (Sıra) :Olayların depolandığı yerdir.

Şekil 24'te Temel Pygame kodu ve çıktısı gösterilmektedir.

```
import pygame, sys
from pygame.locals import *

pygame.init()
DISPLAYSURF = pygame.display.set_mode((400, 300))
pygame.display.set_caption('Hello World!')
while True: # main game loop
    for event in pygame.event.get():
        if event.type == QUIT:
            pygame.quit()
            sys.exit()
    pygame.display.update()
```



Şekil 24. Temel Pygame kod örneği

Yapay Zekâ (Artificial Intelligence, AI) tekniklerini oyunlara uygulamak artık birden fazla konferans ve özel dergilerde yer alan bir araştırma alanı olmuştur (Justesen et al., 2020). Yapay zekâ sistemlerinin herhangi bir insan yardımı olmadan bilgisayar oyunlarının kurallarını anlamasını sağlamak için çaba sarf edilmiştir (Yamini & Jain, 2020).

Video oyunlarında yapay zekâ, uzun süredir devam eden bir araştırma alanı olmuştur. Oyun oynarken insan seviyesinde performans elde etmek için yapay zekâ teknolojilerinin nasıl kullanılacağı sürekli araştırılmıştır. Daha genel olarak, araştırmacılar ajan ve oyun ortamı

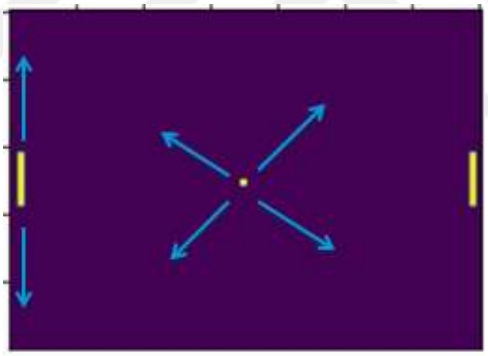
arasındaki karmaşık etkileşimler üzerinde çalışmışlardır. Çeşitli oyunlar, ajanların çözmesi için ilginç ve karmaşık sorunlar sağlayarak video oyunlarını AI araştırması için mükemmel ortamlar haline getirmiştir. Bunu başarmak için, bir ajan oluşturulmalı ve bu ajanı yapay zekâ algoritmalarını kullanarak her durumda hareket etme, oyunda etkileşim ve uygun kararı alma becerisine sahip olmalıdır. Bu nedenle, aracı topun hareket yönünü belirlemeli ve aynı yöne gitmeli, topun yönünün değişmesiyle ne zaman durduğunu ve yönünü değiştirdiğini ayırt etmelidir (Barros et al., 2015).

Önerilen Masa Tenisi Oyunu

Bu tez çalışmasında geliştirilen masa tenisi oyununda biri insan diğeri yapay zekâ destekli olmak üzere iki oyuncu bulunmaktadır. Önerilen oyun ortamında iki nesne vardır:

1- Barlar (Paddles) : Oyuncular tarafından kontrol edilen iki bar bulunmaktadır. Bu barlar yukarı ve aşağı olmak üzere iki yönde hareket ettirilebilmektedir.

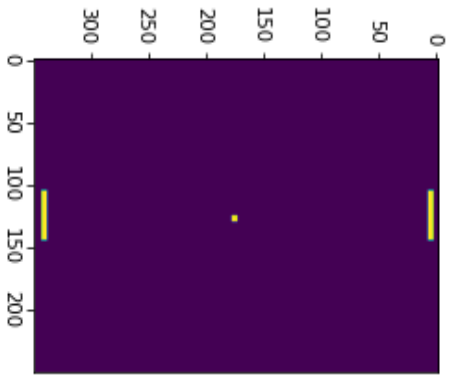
2- Top: Oyun ortamında bir top bulunmaktadır. Oyuncular barı kontrol ederek topun hareket yönünü değiştirir. Burada topun hareket edebileceği dört yön vardır. Barların ve topun başlangıç durum konumları Şekil 25 'te gösterilmektedir.



Şekil 25. Oluşturulan çevrede bar ve topun hareket yönleri

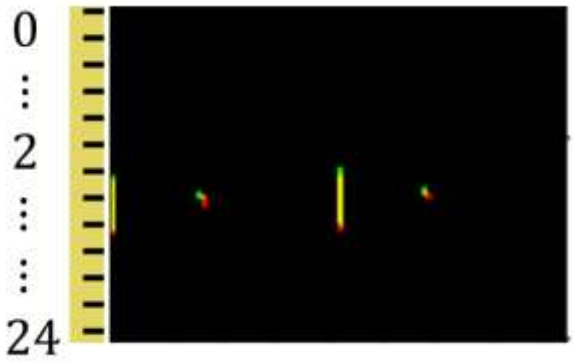
Veri Oluşturulması

Önerilen sanal masa tenisi oyunu saniyede 20 oyun ortamı üretecek şekilde tasarlanmıştır. Bu bir saniyede top ve barların her biri için 20 hareket olduğu anlamına gelmektedir. Her karenin boyutları Şekil 26'da görüldüğü gibi 350x250 (pikselxpiksel) ve topun hızı 80 piksel/saniye'dir. Top herhangi bir bara her çarptığında, topun hızı 0,25 piksel/saniye artmaktadır. Başlangıçta, barlar en fazla sayıda örneği yakalamak için merkezleri topun merkezine eşit olacak şekilde programlanmıştır. Bu, oyun sırasında barların topun hareketine bağlı olarak yukarı ve aşağı hareket edeceği anlamına gelmektedir. Tüm olası top ve bar yönleri için oyun dört kez oynanmış ve her seferinde topun ilk yönü farklı seçilmiştir.



Şekil 26. Örnek çevre görüntüsü

Oyun oyananırken, ortamın her karesi, top ve barları içeren bir durum olarak değerlendirilmiştir; barlardan birisi insan diğeri ise yapay zekâ tarafından kontrol edilen ajandır. Bu tez çalışmasında ajanın konumuna göre 250 piksel olan çerçeve yüksekliği 25 parçaya bölünmektedir. Bu durum Şekil 27'de görülmektedir. Oluşturulan sanal masa tenisi ortamında barın (ajan) konumu bu 25 parça (sınıf) ile temsil edilmektedir. Her çevre durumu alındığında ajanın konumu da alınmaktadır. Örneğin, bir durum alındığında ajan konumu düşeyde 50. piksele denk geldi ise ölçekleme ile konum 5. sınıf olmakta ve bu böyle devam etmektedir. Son olarak, Ping Pong ortamından 5462 durum kaydı toplanmıştır, her kayıt çevre durumu resmini ve ajan eylemini içermektedir. Bu veriler eğitim veri kümesi olarak kabul edilmiştir, ancak daha iyi eğitim yapabilmek için verilerin iyileştirilmesi gerekmektedir.



Şekil 27. Ajanın konumunu belirleyen sınıflar

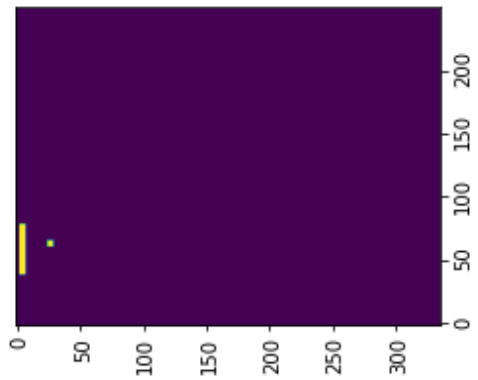
Oluşturulan Verilerin İyileştirilmesi:

Yapay zekâya dayalı sistemlerin doğru bir makine öğrenmesi modeli oluşturması, doğru yanıtları öğrenmesi ve formüle etmesi için kapsamlı, yüksek kaliteli bir eğitim veri kümesi oluşturmak çok önemlidir. Özellikle denetimli makine öğrenimi algoritmalarında, eğitim verilerinin doğru bilgilerle etiketlenmesi gerekmektedir. Makine öğrenimi modelleri çevrelerindeki dünyayı anlamak için verilere ihtiyaç duyduğundan, etiketli eğitim verileri,

algoritmalar tarafından bilinmeyen bilgiler ve örüntüler sağlamaktadır. Eğitim verilerinin önemini anlamak ve kalitesine öncelik vermek, modellerde doğruluk elde edebilmek için büyük ölçüde yardımcı olmaktadır. Kaliteli eğitim verileri elde etmenin ilk adımı, eğitim verilerini etiketlemek için doğru süreçleri ve araçları bulmakla başlamaktadır. (Li et al., 2020)

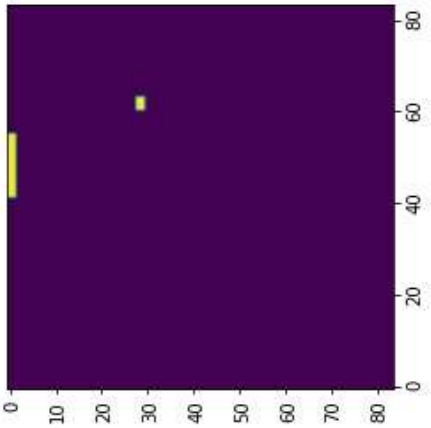
Verimli ve hızlı bir eğitim modeli için veri kümesi yeniden işlenmelidir. Bu durumda yapılan tez çalışması için çözülmesi gereken iki ana konu bulunmaktadır:

1- Rakibin bar hareketi: Bu hareket, bir insan tarafından kontrol edilmektedir. Bu nedenle rakibin hareket zamanı tam olarak bilinemez. Bu, oyun sırasında kayda alınan durumun daha önce kayıt edilmiş durumlardan farklı olmasına neden olmaktadır. Bu durum, ağdan yanlış bir eylem tahmini yapılmasına yol açmaktadır. Sorunun çözümü, rakibin orta sahasını içeren alanı kaldırmak ve çerçevenin geri kalanını korumaktır. Şekil 28, rakibin orta sahasını çıkardıktan sonraki durumu göstermektedir.



Şekil 28. Rakip barı kaldırıldıktan sonra çerçeve durumu

2- Görüntü boyutları: Oyunu oluşturmak için kullanılan ortam, saniyede 20 kare üretmektedir. Her kare 350 x 250 boyutlarına sahiptir. Görsellerin özelliklerinden biri, ağın eğitim sürecini daha hızlı hale getirmek için görüntüyü küçültebilmek ve ağ modelini orijinal boyutta eğitirken aynı verimlilikle alabilmektir. Bu nedenle sürekli gözlem ve deneylerle görüntü boyutu 84 x 84'e düşürülmüştür. Görüntünün bu boyuttan daha fazla küçültülmesi durumunda görüntü özelliklerinin değiştiği fark edilmiş olup, bu da ağa hatalı veri verilmesine neden olmuştur. Şekil 29, küçültülen boyutta yeni ortam durumunu göstermektedir.



Şekil 29. 84x84 boyutunda yeni ortam görüntüsü

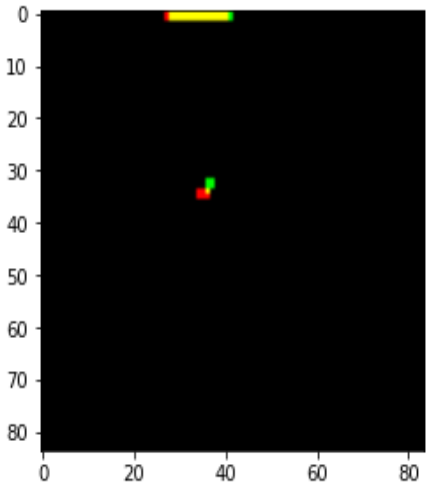
Evrişimli Sinir Ağı

Genel olarak CNN, görüntüleri sınıflandırmak ve makineye görüntü özelliklerini çıkarma ve bunları giriş eğitim verilerinde verilen çıktıyla ilişkilendirme yeteneği vermek için kullanılmaktadır. Önceki bölümde bahsedildiği gibi, bu tür bir yapay zekâ ağı, ortamın durumunu temsil ettiği bir ağ girişi olarak yalnızca bir görüntü alabilmektedir ve ağın çıkışında, ajanın içinde hareket ettiği eylemi temsil etmesini sağlamaktadır.

Evrişimli sinir ağı için veri hazırlama

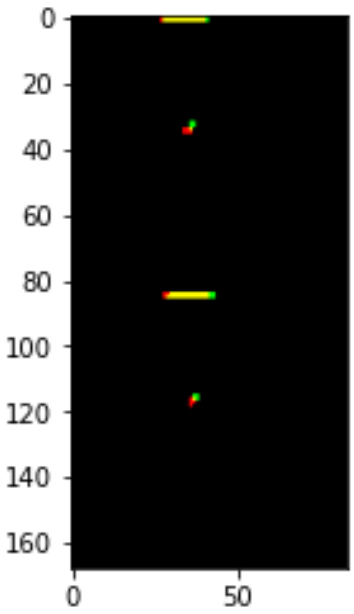
Bu tez çalışmasında Evrişimli Sinir Ağını eğitmeden önce görüntülere iki görüntü iyileştirmesi yöntemi uygulanmıştır:

1- Her çerçeve, siyah ve beyaz olmak üzere iki renk veri içermekte, ancak siyah renkte beyazdan daha fazla veri bulunmaktadır. Bu nedenle eğitim sırasında ağın performansını iyileştirmek için t anındaki 84x84x1 boyutlarındaki görüntüye t+1 anındaki görüntüde ilave edilerek giriş resminin görüntüsü 84x84x2 olarak elde edilmektedir. Bir önceki görüntü için, Şekil 30'da gösterildiği gibi, bir sonraki görüntüyü temsil eden orijinal görüntüye boyut eklenmiştir.



Şekil 30. Birbirini takip eden zamanda iki resmin arka arkaya birleştirilmesiyle elde edilen görüntü

2- Topun hareket yönü bilgisinin de gözlenebilmesi için de t+1 anındaki resim ile t+2 anındaki resim arka arkaya birleştirildikten sonra Şekil 29'da görülen birleştirilmiş resim ile bu yeni elde edilen birleştirilmiş resim uç uca eklenerek 168x84x2 boyutlarında yeni birleştirilmiş resim görüntüsü elde edilmiştir. Bu yeni resimde orjinalde üç farklı zamanda elde edilen resimlerin birleştirilmesi elde edilmiştir ve Şekil 31'de gösterildiği gibidir. Bu resim CNN algoritmasına giriş olarak verilmektedir.



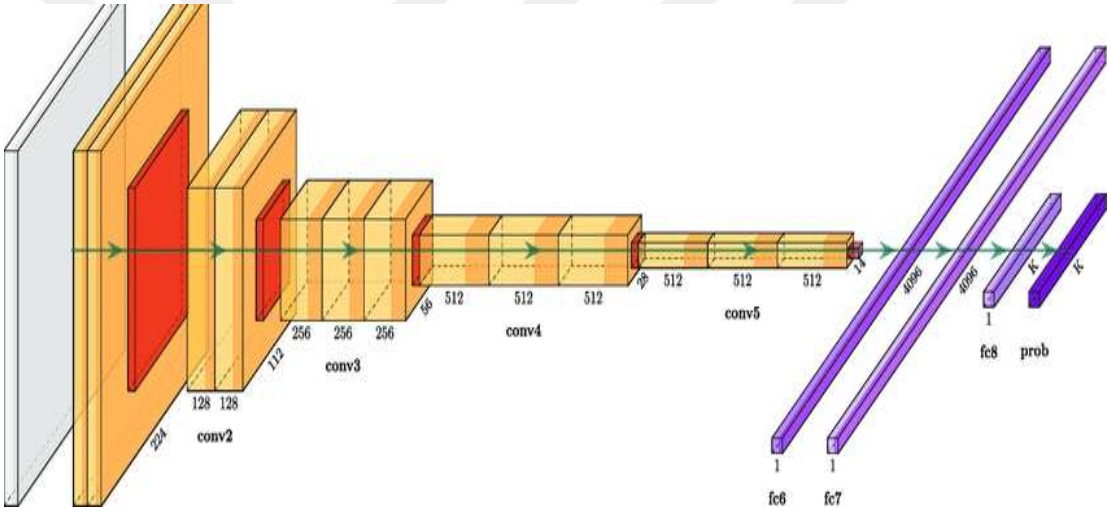
Şekil 31. Şekil 30'da verilen resimlerin yan yana birleştirerek elde edilen CNN giriş görüntüsü

Ağ modelleri

Bu çalışmada, performans karşılaştırması yapmak ve hangi ağın eğitim ajanı için uygun olduğunu bulmak için iki farklı CNN ağı kullanılmıştır. Bu ağlardan ilki VGG16 ile transfer öğrenme yapılarak elde edilen evrişimli sinir ağı diğeri ise bu tez çalışması kapsamında tasarlanan yeni bir evrişimsel sinir ağı modelidir (Simonyan & Zisserman, 2014). Bu tez çalışmasında kullanılan her bir ağ, bar konumuna dayalı olarak bir kez eğitilmiştir.

VGG16

VGG16, 224 × 224 × 3 giriş görüntüsünü alan ve bu giriş görüntüsünü hiyerarşik bir şekilde bir dizi çıkış sınıfı olasılığına dönüştüren bir evrişimli sinir ağı modelidir. Şekil 32'de görülmektedir ((Blauch et al., 2021). Bu tez çalışmasında 23 sınıflı bir problem ele alındığından dolayı orjinalde 1000 sınıf için ayarlanan VGG16'nın son katmanlarında gerekli ayarlamalar yapılmıştır.



Şekil 32. VGG16 mimarisi (Blauch *et al.*, 2021)

conv2d_1 (Conv2D)	(None, 168, 84, 64)	1216
conv2d_2 (Conv2D)	(None, 168, 84, 64)	36928
max_pooling2d_1 (MaxPooling2	(None, 84, 42, 64)	0
conv2d_3 (Conv2D)	(None, 84, 42, 128)	73856
conv2d_4 (Conv2D)	(None, 84, 42, 128)	147584
max_pooling2d_2 (MaxPooling2	(None, 42, 21, 128)	0
conv2d_5 (Conv2D)	(None, 42, 21, 256)	295168
conv2d_6 (Conv2D)	(None, 42, 21, 256)	590080
conv2d_7 (Conv2D)	(None, 42, 21, 256)	590080
max_pooling2d_3 (MaxPooling2	(None, 21, 10, 256)	0
conv2d_8 (Conv2D)	(None, 21, 10, 512)	1180160
conv2d_9 (Conv2D)	(None, 21, 10, 512)	2359808
conv2d_10 (Conv2D)	(None, 21, 10, 512)	2359808
max_pooling2d_4 (MaxPooling2	(None, 10, 5, 512)	0
conv2d_11 (Conv2D)	(None, 10, 5, 512)	2359808
conv2d_12 (Conv2D)	(None, 10, 5, 512)	2359808
conv2d_13 (Conv2D)	(None, 10, 5, 512)	2359808
max_pooling2d_5 (MaxPooling2	(None, 5, 2, 512)	0
flatten_1 (Flatten)	(None, 5120)	0
dense_1 (Dense)	(None, 4096)	20975616
dense_2 (Dense)	(None, 4096)	16781312

Windows'u Etkinleştir

Şekil 33. VGG16 mimari ve parametreleri

• Eğitim Verileri

Sanal Masa tenisi oyun ortamı oluşturmak için kullanılacak evrişimsel sinir ağının eğitimi için toplam 9920 veri toplanmıştır. Bu veriler 168 x 84 x 2 boyutunda olup topun pozisyonunun bilgisayar tarafından bilindiği ve her iki oyuncunun da bu bilgiye göre hareket ettiği durumda toplanmıştır. Bu tez çalışmasında eğitim, doğrulama ve test veri kümeleri sayısı sırasıyla 5460, 2230, 2230 olup, tümü rastgele dağıtılmıştır.

Transfer öğrenme için hiper parametreler:

Bar hareketlerine ve pozisyona dayalı kullanılan VGG16 mimarisi Şekil 33'te gösterilmektedir.

Bu ağda kullanılan eğitim parametreleri:

Kayıp Fonksiyonu: 'categorical_crossentropy'

Optimizasyon Algoritması: 'adam'

Parti(batch) Boyutu: 64

Epok Sayısı: 29.

Önerilen Yeni Evrişimli Sinir Ağı Model Mimarisi:

Bar hareketlerine ve pozisyona dayalı model Şekil 34'te gösterilmektedir.

Bu ağda kullanılan eğitim parametreleri:

Kayıp Fonksiyonu: 'categorical_crossentropy'

Optimizasyon Algoritması: ' adam'

Parti(batch) Boyutu: 64

Epok: 9

conv2d_1 (Conv2D)	(None, 168, 84, 50)	1400
conv2d_2 (Conv2D)	(None, 168, 84, 75)	33825
max_pooling2d_1 (MaxPooling2	(None, 84, 42, 75)	0
dropout_1 (Dropout)	(None, 84, 42, 75)	0
conv2d_3 (Conv2D)	(None, 84, 42, 125)	84500
max_pooling2d_2 (MaxPooling2	(None, 42, 21, 125)	0
dropout_2 (Dropout)	(None, 42, 21, 125)	0
flatten_1 (Flatten)	(None, 110250)	0
dense_1 (Dense)	(None, 500)	55125500
dropout_3 (Dropout)	(None, 500)	0
dense_2 (Dense)	(None, 250)	125250
dropout_4 (Dropout)	(None, 250)	0

Şekil 34. Önerilen yeni evrişimli sinir ağı model mimarisi

Uzun Kısa Süreli Bellek Ağları (LSTM)

LSTM algoritması zamana bağlı verilerin işlenmesinde kullanılmaktadır. Özellikle resim altyazıları, konuşma sentezi, müzik üretimi, video oyunlarının üretilmesi ve sınıflandırılması uygulamalarında sıklıkla kullanılmaktadır (Lipton *et al*. 2015)(Hochreiter & Schmidhuber, 1997).

Görüntü dizisi sınıflandırması, ayrı çerçevelerin görüntü içeriklerinin CNN ve LSTM gibi derin sinir ağları tarafından tek bir sınıfta özetlenmesiyle gerçekleştirilir. Sanal masa tenisi ortamının oluşturulduğu bu tez çalışmasında her bir çevre görüntüsü belirli bir zaman anında alındığından elde edilen veriler zaman serisi olarak değerlendirilebilir. Bu sebeple ajanın konumunu bulmak için CNN algoritması ile önerilen sınıflandırma problemi LSTM algoritması yardımıyla zamana bağlı veriler kullanılarak benzer şekilde çözülebilir.

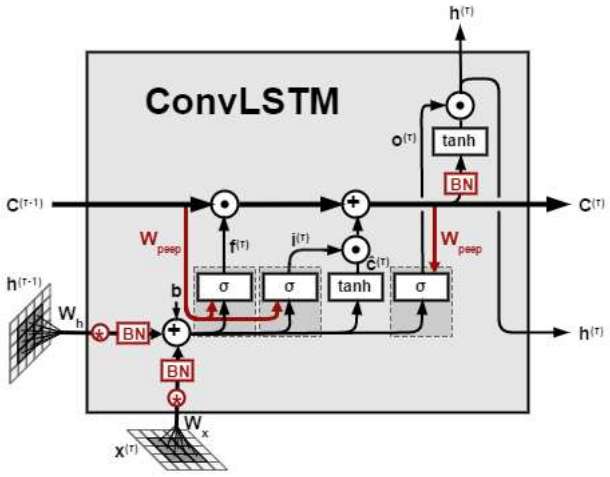
Bu tez çalışmasında önerilen çerçevede, sınıflandırma için çift yönlü bir LSTM ağına beslenen görüntü dizisinin özniteliklerini çıkarmak için 2D CNN ağları kullanılır. Bu amaç

doğrultusunda LSTM ağından sonra bir dikkat bloğu (Attention Block) eklenmiştir. Deneylerde birkaç farklı 2D CNN mimarisi test edilmiştir.

ConvLSTM:

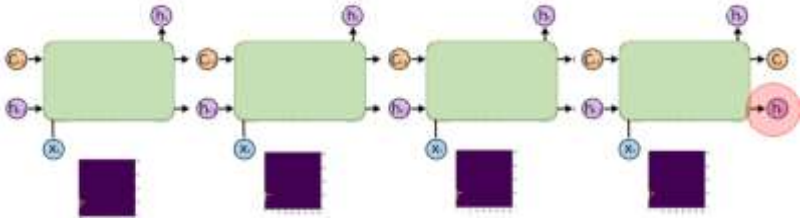
ConvLSTM, LSTM ağına dayalı geliştirilmiş bir algoritmadır. Bu algoritma geleneksel LSTM modelleri gibi zaman serisi karakteristiklerini oluşturmakla kalmaz, aynı zamanda CNN gibi yerel uzaysal özellikleri de dikkate alır. Girdi verileri CNN işlemlerinden geçirilerek verilerin uzay-zaman özelliği daha detaylı incelenir ve tam bağlantı katman aracılığıyla model çıktısı alınır(Chen *et al.* 2020).

ConvLSTM'nin temeli, önceki katmanın çıktısını bir sonraki katmanın girdisi olarak kullanan LSTM ile aynıdır. Aralarındaki fark, evrişim işlemini ekledikten sonra, yalnızca zamanlama ilişkisini elde etmek için değil, aynı zamanda evrişim katmanı gibi özellikleri çıkarmak için de uzamsal özellikleri çıkarmasıdır. Bu şekilde uzay-zaman özellikleri elde edilir. Durumlar arası geçişin yerini de evrişim hesaplaması alır. Şekil 35'te, ConvLSTM hücresi gösterilir.



Şekil 35. ConvLSTM Hücresi (Alexandre Xavier 2019)

ConvLSTM'in farklı bir yaklaşımı, Convolutional-LSTM modelidir. İlk olarak görüntü evrişim katmanlarından geçer. Daha sonra katmanların çıktısı elde edilen özelliklerle 1 boyutlu bir diziye düzleştirilmiş bir küme olur. Bu işlemi zaman kümesindeki tüm görüntülere tekrarlar. Sonuç olarak zamanla bir dizi özellik elde edilir ve bu LSTM katmanı girişidir. Şekil 36, ağın bir örneğini gösterir.



Şekil 36. Girdi olarak resim kullanan ConvLSTM mimarisi

• Eğitim Verileri

Evrişim katmanı girişi, örnek sayısı, kanal sayısı, satır ve sütundan oluşan 4D tensör olarak temsil edilen bir dizi görüntü dizisidir. Burada örnek sayısı, ağa toplam kaç örnek girdiğini temsil eder. ConvLSTM girişi ise örnek sayısı, zaman adımı, kanal sayısı, satır ve sütun olan bir 5D tensör olarak zaman içinde bir görüntü kümesidir. Burada zaman adımı (time_steps), ağa aynı anda kaç dizi görüntü girdiğini gösterir.

CNN yönteminde anlatılan (84 x 84 x 2) resimlerinden zaman içerisindeki dört tanesi birleştirilerek ConvLSTM ağ mimarisine giriş olarak verilmektedir. Bu ağ mimarisi için giriş görüntüsü (4 x 84 x 84 x 2) biçimindedir. Örnek sayısının 2732 olduğu dikkate alındığında 5D tensör yapısı (2732x4x84x84x2) şeklinde olmaktadır.

• LSTM Ağı Model Mimarisi:

Bu modelde CNN ve LSTM yöntemleri kullanılır. CNN'nin temel amacı, görüntünün özelliklerini çıkarmak ve onu 1D matris olarak temsil etmektir. Daha sonra, bu matris bir dizi olarak LSTM modeline verilir. CNN Modelinde, 3x3 boyutunda dolgu (padding) olmadan kullanılan ve atlama (stride) miktarı 1 olan 64 çekirdek (kernel) yer almaktadır. ConvLSTM model parametreleri, Şekil 37'de gösterildiği gibi özetlenebilir.

Layer (type)	Output Shape	Param #
conv_lst_m2d_3 (ConvLSTM2D)	(None, 82, 82, 64)	150016
dropout_5 (Dropout)	(None, 82, 82, 64)	0
flatten_3 (Flatten)	(None, 430336)	0
dense_5 (Dense)	(None, 256)	110166272
dropout_6 (Dropout)	(None, 256)	0
dense_6 (Dense)	(None, 3)	771

Total params: 110,317,059
Trainable params: 110,317,059
Non-trainable params: 0

Şekil 37. ConvLSTM model mimari ve parametreleri

Derin Q Ağı (DQN)

Sinir ağlarına ve genetik algoritmaya dayalı bu model, bilgisayar oyunlarının geliştirilmesi için ağaç şeklinde dinamik programlamanın yerini alma potansiyeline sahiptir.

Yapay zekâ sistemi için, çoğunlukla öğrenme hızını artıran sinir ağı algoritması nedeniyle yeterince erken ve iyi puan almaya başladığı için sığ bir öğrenme eğrisine sahiptir. Ancak, bu modelin eğitilmesi için diğer algoritmalara nazaran çok fazla eğitim verisine ihtiyaç vardır (Yamini & Jain, 2020).

Ajanların oyun oynamak için eğitilmesinin en yaygın yolu, pekiştirmeli öğrenme yöntemlerini kullanmaktır. Pekiştirmeli öğrenme, belirli bir durumdaki bir ajanın önceden tanımlanmış bir ortamda bir eylemde bulunmayı seçtiği ve bir miktar ödül aldığı süreçtir. Ajan daha fazla eylemde bulundukça, her bir durum için, puanını maksimize etmek istiyorsa hangi eylemin en iyi olduğunu belirleyebilmektedir (Vu & Tran, 2020).

Önceki yöntemlerde, ajan, toplanan etiketli verilerle eğitildi. Bu yöntemde ajan, çevreye eylem vererek, çevreden ceza veya ödül almaktadır. Ardından aynı ortamdaki eylemi tahmin etmektedir. Bu döngü en uygun ödüle kadar devam etmektedir. Çevreden örnek almak için iki yöntem kullanılmaktadır. Bunlardan ilki, çevreden alınan alınan resimlerdir. İkincisi ise, ajanın ve topun durum bilgileridir.

DQN ağı için veri hazırlama

Ajanın Durumuna Göre Alınan Çerçeveler:

350 × 250 boyutundaki resimlerle doğrudan çalışmak, işlem yükünü arttırmkatadır. Bu nedenle bu çalışmada karelerin boyutunu küçültmek için temel bir ön işleme adımı uygulanmıştır. Çerçeve aslen siyah ve beyaz çerçevedir. Ancak kenar çubuğu, eğitim işlemlerinde hareketi ihmal etmek için silinmiştir.

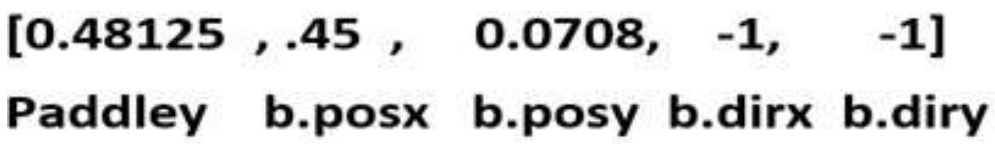
Bu silme işleminden sonra, yeni çerçevenin boyutu 335 x 250'dir. Ancak bu çerçeve boyutuyla öğrenmek hala zordur. Çünkü özellikle ajanı eğitmek için binlerce adım söz konusu olduğu için çok zaman almaktadır. Eğitim sürecini hızlandırmak için, çerçeve boyutu 84 x 84 olarak değiştirilmiş ve bu da aynı etkiyle süreyi azaltmıştır.

Son olarak, ağın topun hangi yöne hareket ettiğini algılaması ve anlaması için dört seri çerçeve alınmış ve boyutsal olarak düzenlenmiştir.

Ajanın Durumuna Göre Alınan Ajan Pozisyonu:

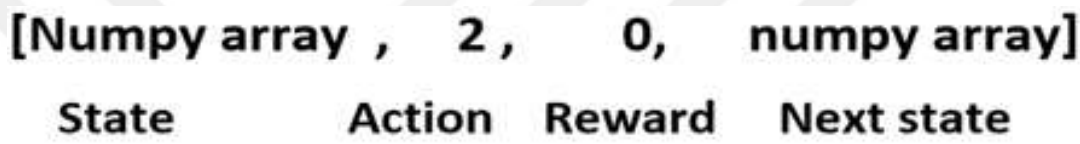
Ajan durumunu oluşturmak için yalnızca top konumu, top yönü ve ajan konumu gereklidir. 350 x 250 boyutlu giriş reminin yerine çok daha az sayıda giriş boyutuna sahip sinir

ağı etkin bir şekilde eğitilir. Burada ajan ve top koordinatları 0 ile 1 arasında olacak şekilde normalize edilmiştir. Ajan Y olan bir koordinatta hareket edebildiği için bu durum önemlidir. Yani sadece y koordinatındaki pozisyon bilgisi alınır. Bu bilgiye ilaveten top için yer ve yön bilgisinin belirtilmesi gerekmektedir. Bu bilgilerden satır vektörü, Şekil 38'de gösterilmektedir.



Şekil 38. DQN mimarisi için Çevre Durum vektörü

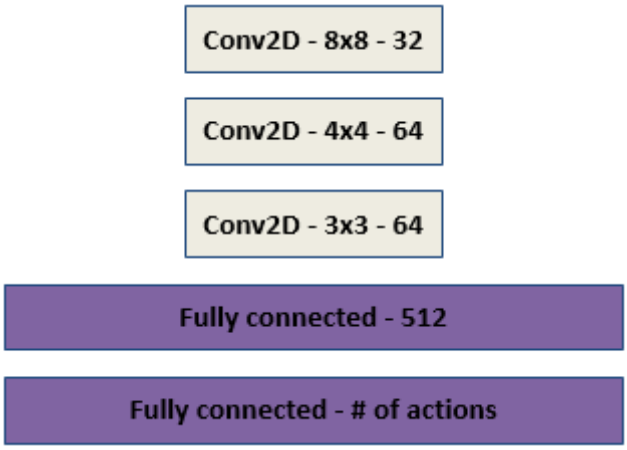
Çevreden alınan bir örnek ise şu şekilde tanımlanmaktadır; Durumun şuanki değeri, eylem, elde edilen ödül ve gelecek durumun değeridir. Tüm bu değerler birleştirildiğinde 12x1 boyutundaki örnek Şekil 39 'da gösterilmektedir.



Şekil 39. DQN mimarisi için Çevre Örnek vektörü

DQN Algoritması için Ağ Mimarisi (Ajanın Durumuna Göre Alınan Resimler Yöntemi için):

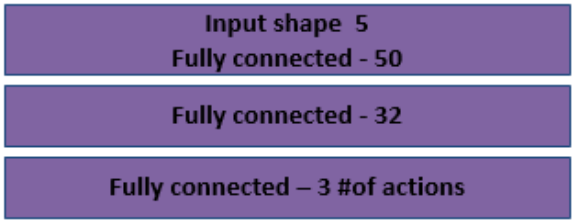
Masa tenisi oyunu için geliştirilen DQN mimarisinde eylemleri tahmin etmek için kullanılan CNN ağ mimarisi Şekil 40'da gösterilmiştir. Bu ağın giriş katmanı, 84 × 84 × 4 boyutlu görüntülerden oluşur. Tam bağlantılı katmandan üç farklı eylem için üç farklı çıkış vardır.



Şekil 40. Çevre resminin girdi olarak kullanıldığı ağ mimarisi

DQN Algoritması için Ağ Mimarisi (Ajanın Durumuna Göre Alınan Ajan Pozisyonu Yöntemi için):

Ağ için girişi Şekil 39'da verilen 1x5 boyutlu vektördür. Bu nedenle Şekil 41'de gösterildiği gibi sadece tam bağlantılı katmanlar kullanılmaktadır.



Şekil 41. Ajan poziyonuna göre ağ mimarisi

Öğrenme Nasıl Gerçekleştirilir:

Denetimli öğrenmede, eğitim sırasında bir modelin performansı eğitim ve doğrulama setlerinde değerlendirilerek kolayca izlenebilir. Ancak pekiştirmeli öğrenmede, eğitim sırasında bir ajanın ilerlemesini doğru bir şekilde değerlendirmek zor olabilir. Değerlendirme metriği, ajanın bir bölümde veya oyunda topladığı toplam ödülün birkaç oyun üzerinden ortalamasıdır. Bu eğitim sırasında periyodik olarak hesaplanır. Ortalama toplam ödül metriği çok gürültülü olma eğilimindedir. Çünkü bir politikanın ağırlıklarındaki küçük değişiklikler, politika ziyaretlerinin yapıldığı durumların dağılımında büyük değişikliklere yol açabilir. Bir başka, daha istikrarlı metrik, politikanın, ajanın herhangi bir durumdan politikasını izleyerek ne kadar indirimli ödül alabileceğine dair bir tahmin sağlayan, politikanın tahmini eylem değeri işlevidir. Eğitim başlamadan önce rastgele bir politika yürüterek sabit bir dizi durum toplanır ve bu durumlar için maksimum tahmin edilen Q'nun ortalamasını izlenir (Mnih et al., 2015).

Pekiştirmeli öğrenmedeki en büyük zorluklardan biri, keşif ve faydalanma arasındaki temel ikilemdir. Ajan, çevreyi keşfetmek (ve potansiyel olarak modeli geliştirmek) için ne zaman optimal olmayan eylemleri denemeli (algılamalı) ve ne zaman optimalden yararlanmalı, yararlı bir ilerleme sağlamak için hangi eylemi gerçekleştirmeli şeklindeki sorular için DQN gibi politika dışı algoritmalar tipik olarak, olasılık $\in [0, 1]$ olan rastgele bir eylemi, ya da en uygun eylemi seçen basit açgözlü keşif politikasını kullanır. Zaman içinde ajan, faydalanmaya doğru ilerler. Keşif için bağımsız gürültü eklemek sürekli kontrol problemlerinde kullanılabilir olsa da, daha karmaşık stratejiler momentumu daha iyi korumak için zamanla ilişkilendirilen gürültüyü ilave eder. (Arulkumaran *et al.* 2017)

Bu yöntemde, bunlara bağlı olarak ajanı öğrenen çıktılar yoktur. Ancak ajanın içinde bulunduğu çevreyi anlamak ve onunla etkileşime geçmek için kullanılan eylemler vardır. Genel olarak, ortamdan sürekli olarak alınan bir dizi görüntü vardır ve her durumda ortamın durumu

sırasında en iyi eylemin alındığından emin olmak için çıktılar oluşturulmalı ve bunlar ortamda test edilmelidir.

DQN algoritmasının bileşenleri:

1. Yapı Modeli (Build model): Deep Q ağının yapısını oluşturmak için KERAS kütüphanesi kullanılmıştır. Geliştirilen modelin giriş katmanı durumlardan oluşmaktadır. Modelin çıkış katmanı, oyundaki 3 eylemin 3 Q değerini temsil eden 3 nörona sahiptir. Ajan elde edilen bu üç Q değerinden en büyük değere sahip olan eylemi seçer.
2. Hatırlatıcı (Remember): Durumu, eylemi, ödülü ve sonraki durumu tuple <s, a, r, s '> içinde saklar (Şekil 40). Ağ tüm bu bigiler dikkate alınarak eğitilir.
3. Eylem (Action): İlk olarak 0 ile 1 arasında rastgele bir sayı oluşturulur. Ardından bu rastgele sayının epsilon'dan büyük olup olmadığını kontrol edilir. Değilse, rastgele bir eylem çıkarılır. Epsilon'dan daha büyükse, giriş durumunun Deep Q ağından geçmesine izin verilir ve bir eylemi tahmin eder ve ardından bu eylem çıkarılır.
4. Replay (Replay): Ağ toplu olarak eğitilir ve Deep Q ağındaki parametreler güncellenir (Dewan et al., n.d. 2020).

Ajanın Durumuna Göre Alınan Çerçeveler:

Keşif:

Bu aşamada ajanın, rastgele eylemler gerçekleştirmesine, durumu, eylemi, ödülü ve sonraki durumu 10000 adım için saklamasına izin verilir.Keşif sırasında, eylem seçimi, keşif ve faydalanma arasında rastgele seçim yaparak keşif ve faydalanmayı dengelemek için Epsilon-Greedy yöntemi kullanılır.

Faydalanma:

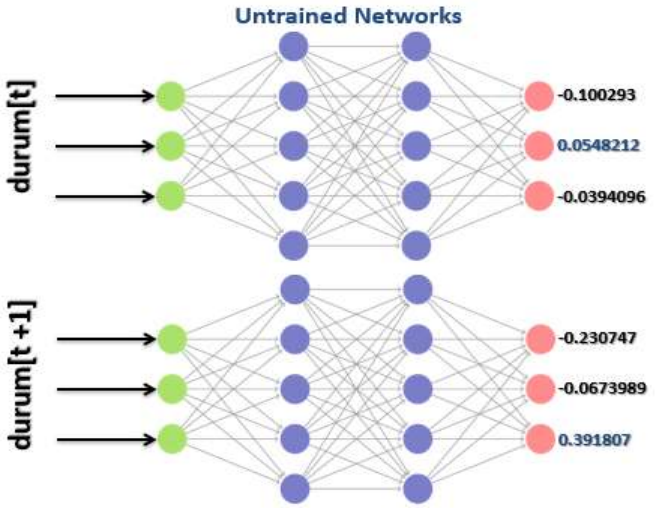
Bu aşama, CNN ağını kullanarak ağırlıkların güncellenmesinden sorumlu olan yeniden oynatma işlevini yürütür:

- Hatırlatıcı (Remember) deposundan rastgele (batch=32) örnekler alınarak CNN ağına verilir.
- Her örnek durum, sonraki durum, ödül ve eylemi içerir. Durum ve sonraki durum ağ üzerinden geçtikten sonra, durum için üç ve sonraki durum için bir tane olmak üzere Şekil 42 'de gösterildiği gibi altı çıktı alınır.
- Bellman denklemini kullanarak Q değerini hesaplanır.

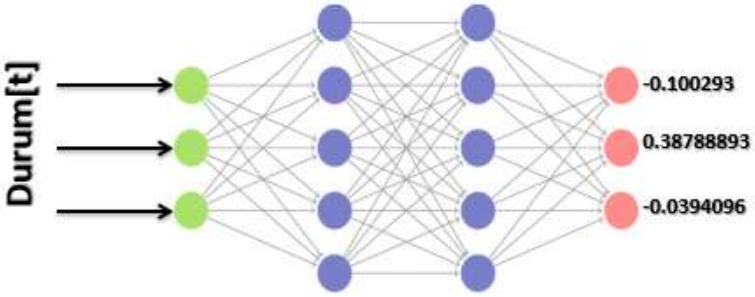
- Durum CNN ağ çıkışını güncellemek için eylem değerini kullanarak, şekilde gösterildiği gibi ödül değerinin 0 ve indirim faktörünün 0.99 olduğunu varsayıldığında güncelleme denklemi aşağıdaki gibi ifade edilir.

$Q(s_t, a) \leftarrow 0 + .99^* (0{,}391807)$

Çıkışın etiket olduğu ağ bilgisinin işlem parametreleri (durum[t] ,[-0,100293 0,38788893 -0,0394096]) Şekil 43'te gösterildiği gibidir.



Şekil 42. Eğitimsiz ağ üzerinden durum ve sonraki durum çıktıları



Şekil 43 . Güncellenmiş eylem ağırlığı değerleri, ödül 0 durumu

- Ödül değeri -1 ise q değeri aşağıdaki gibi hesaplanır.

$Q(s_t, a) \leftarrow -1 + .99 * (0{,}391807)$

Model dizisi parametreleri (durum[t] ,[-0,100293 -0,61211107 -0,0394096]) şeklinde ifade edilir. Bu parametreler çıktının etiket olduğu bilgisini verir.

- Kayıp ortalama kare hata fonksiyonunu kullanarak hesaplanır.
- Ağırlıkların geriye yayılması ve güncellenmesi için eğitim işlevi kullanılır.

- Faydalanmanın ilerlemesi sırasında eylemi varsaymak için, ortamdan çekilen resim ağa yüklenir, bu giriş için üç çıktı elde edilir. Elde edilen her bir eylemin olasılığını elde etmek için softmax işlemi kullanılır.
- Çevre üzerinde varsayılan eylemin uygulanması; bir sonraki durumu tekrarlayarak yeni örneği deneyim cevabı olarak kaydeder.

Ajanın Durumuna Göre Alınan Ajan Pozisyonu:

Keşif:

Başlangıçta, ajan rastgele 750 eylem seçer ve her durumun bir sonraki durumunu toplayarak, bundan sonra her durum için Şekil 43'te gösterildiği gibi eğitim örneği oluşturulmuştur.

Faydalanma:

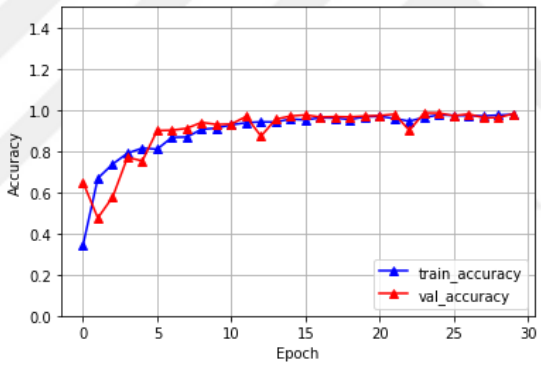
- 750 kayıt toplandıktan sonra, eğitim süreci için rastgele 64 kayıt seçilmektedir.
- Ardından, ağın başlangıç ağırlıklarını güncellemek için bu 64 kayıt ağa uygulanmaktadır.
- Eğitim sürecinde, ajan ortamdan bir durum alarak ve hangi eylemin olması gerektiğini tahmin etmektedir.
- Bir sonraki durumu ve ödülü almak için çevre üzerinde öngörülen eylemi uygulamaktadır.
- Yeni örneği deneyim yanıtında kaydetmektedir.
- Ağın çıkış ağırlıklarını güncellemek için Bullmen denklemi kullanılmaktadır. Denklem ödül tarafından kontrol edilirken, ödül 0 ise ağırlıklarda hiçbir değişiklik olmaz. Aksi takdirde ödül -1 olur, yani ağırlık -1 azalır ve bir dahaki sefere bu eylemden kaçınılır.
- Bu işlem 15000 kez tekrarlanır.

ARAŞTIRMA BULGULARI

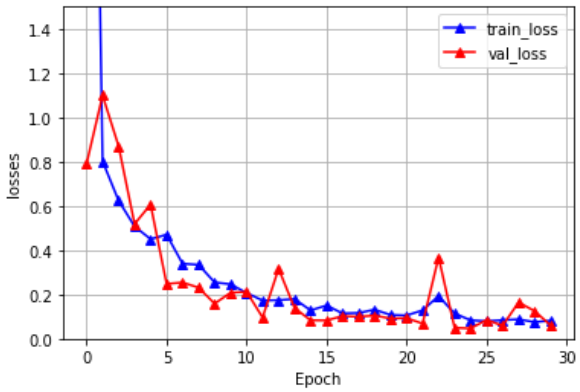
CNN Yöntemi ile Elde Edilen Sonuçlar

VGG16 Ağı ile Elde Edilen Sonuçlar

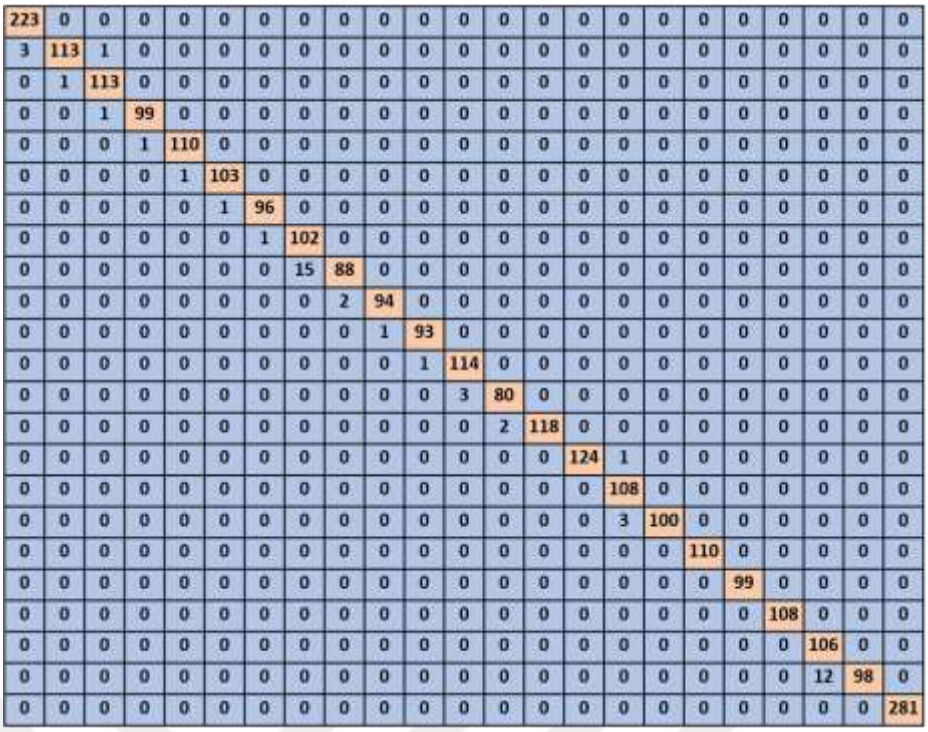
Eğitim sürecinde CNN ağı, optimizeri ‘Adam’, kayıp fonksiyonu 'categorical_crossentropy' ve epok sayısı 30 olarak ayarlanarak eğitilmiştir. Doğrulama için 2730 örnek kullanılmıştır. Eğitim sonuçlarında kayıp:0.08164 ve doğruluk:0.978 olarak hesaplanırken, doğrulama sonuçlarında ise kayıp: 0.06015 ve doğruluk: 0.9813 olarak bulunmuştur. Model 2730 örnekle test edilmiştir. Test sonucu %98.1 olarak elde edilmiştir. Şekil 44’te eğitim için doğruluk ve kayıp gösterilirken, Şekil 45’te doğrulama için doğruluk ve kayıp gösterilmektedir. Bir sınıflandırma modelinin test performansının karışıklık matrisi Şekil 46’da gösterilmiştir.



Şekil 44. VGG16 için Epok sayısına göre eğitim ve validasyon doğruluk performans grafiği



Şekil 45. VGG16 için Epok sayısına göre eğitim ve validasyon kayıp grafiği



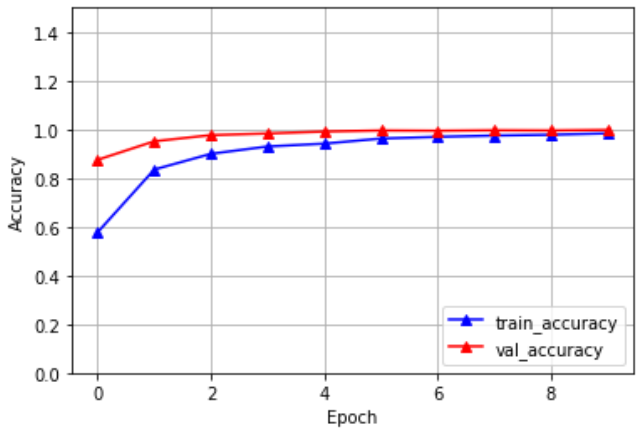
223	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	113	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	113	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	99	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	110	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	103	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	96	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	102	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	15	88	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	2	94	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	93	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	114	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	3	80	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	2	118	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	124	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	108	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	100	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	110	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	99	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	108	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	106	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	98	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	281

Şekil 46. VGG16 için krışıklık matrisi

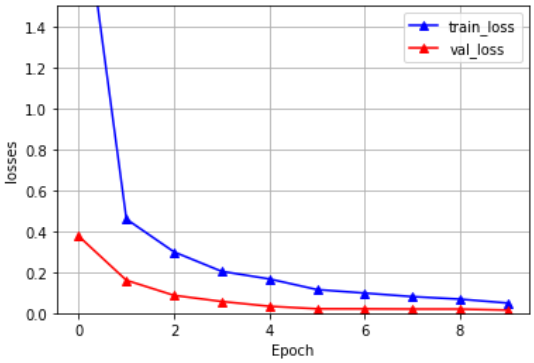
Bu model ile oyun oynanırken zaman gecikmesi olduğu görülmüştür. Bar hareketi oyun sırasında her zaman topun pozisyonunu takip etmektedir.

Önerilen Yeni Evrişimli Sinir Ağı ile Elde Edilen Sonuçlar

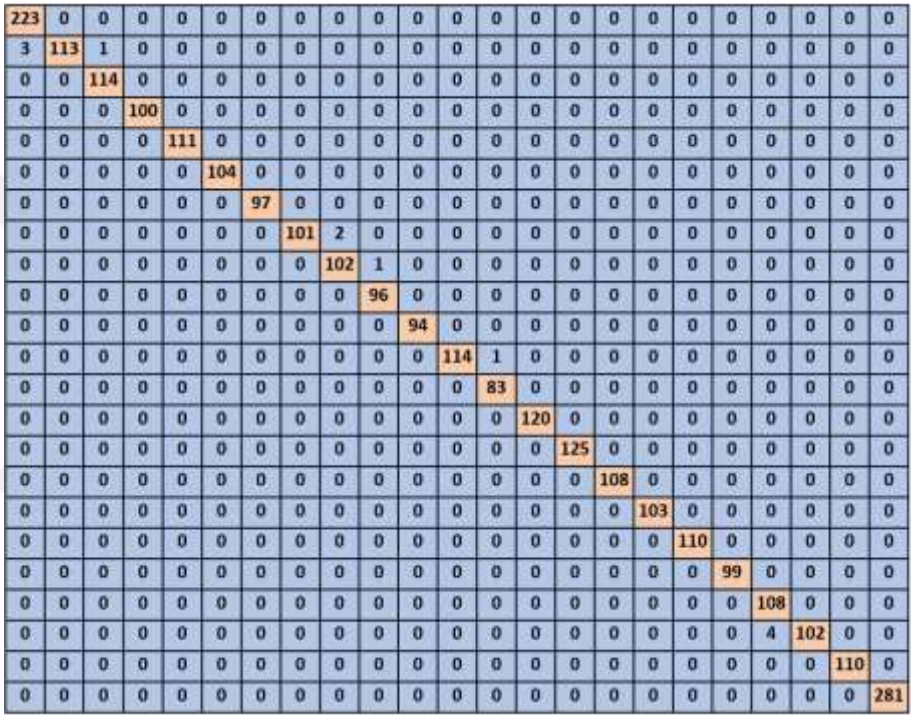
Eğitim sürecinde CNN ağı bir Adam optimizer kullanılarak eğitilmiştir, kayıp fonksiyonu 'categorical_crossentropy' ve epoch 10'dir. Doğrulama için 2730 örnek alınmıştır. Eğitim sonuçları için kayıp: 0.0489 - doğruluk: 0.9837, doğrulama için kayıp: 0.01436 - doğruluk: 0.9978 olarak elde edilmiştir. Bundan sonra model 2730 örnekle test edilmiştir. Test sonucu %99.6. Şekil 47'de eğitim için doğruluk ve kayıp performanslarını ve Şekil 48'de doğrulama için doğruluk ve kayıp performanslarını gösterilmektedir. Bu ağ ile eğitilen sınıflandırma modelinin test performansının karışıklık matrisi Şekil 49'da gösterilmiştir.



Şekil 47. Önerilen Yeni Evrişimli Sinir Ağı için Epok sayısına göre eğitim ve validasyon doğruluk performans grafiği



Şekil 48. Önerilen Yeni Evrişimli Sinir Ağı için Epok sayısına göre eğitim ve validasyon kayıp grafiği



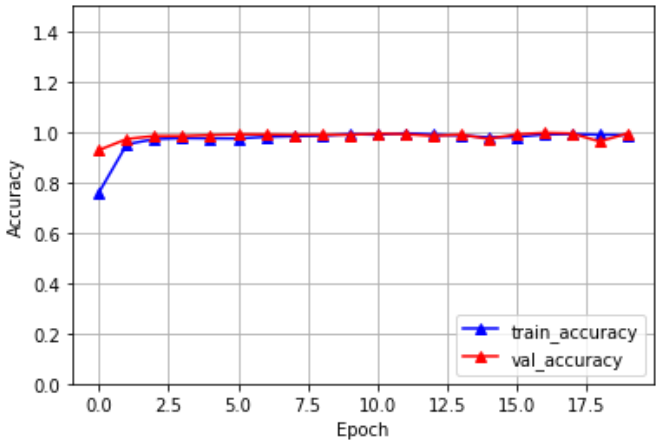
Şekil 49. Önerilen Yeni Evrişimli Sinir Ağı için karışıklık matrisi

Model oyun üzerinde çalışırken oyun hareketinde zaman gecikmesi olduğu görülmüştür. Bar hareketi oyun sırasında her zaman topun pozisyonunu takip etmektedir.

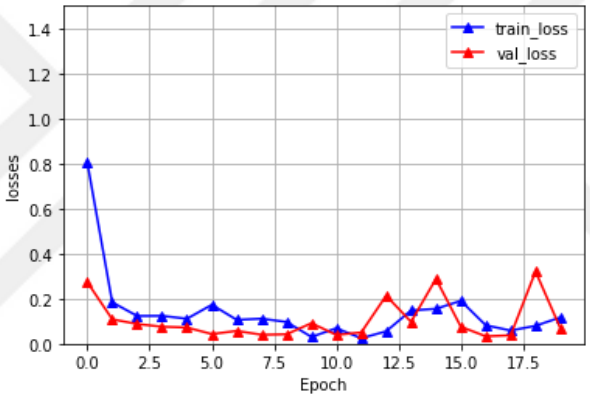
LSTM Yöntemi ile Elde Edilen Sonuçlar

Eğitim sürecinde, ConvLSTM ağı, 0,001 öğrenme oranı, 'categorical_crossentropy' kayıp fonksiyonu ve epok sayısı 20 olan bir Adam optimizer kullanılarak eğitilmiştir. Doğrulama için 1366 örnek alınmıştır. Eğitim sonuçları için hata: 0.0917 - doğruluk: 0.9892, doğrulama için hata: 0.0674 - doğruluk: 0.9941 olarak elde edilmiştir. Daha sonra model 1366 örnekle test edilmiştir. Test sonucu % 99.4 olarak bulunmuştur. Şekil 50'de eğitim için doğruluk ve kayıp performanslarını ve Şekil 51'de doğrulama için doğruluk ve hata oranları

gösterilmektedir. Bir sınıflandırma modelinin test performansının karışıklık matrisi Şekil 52'de gösterilmiştir.

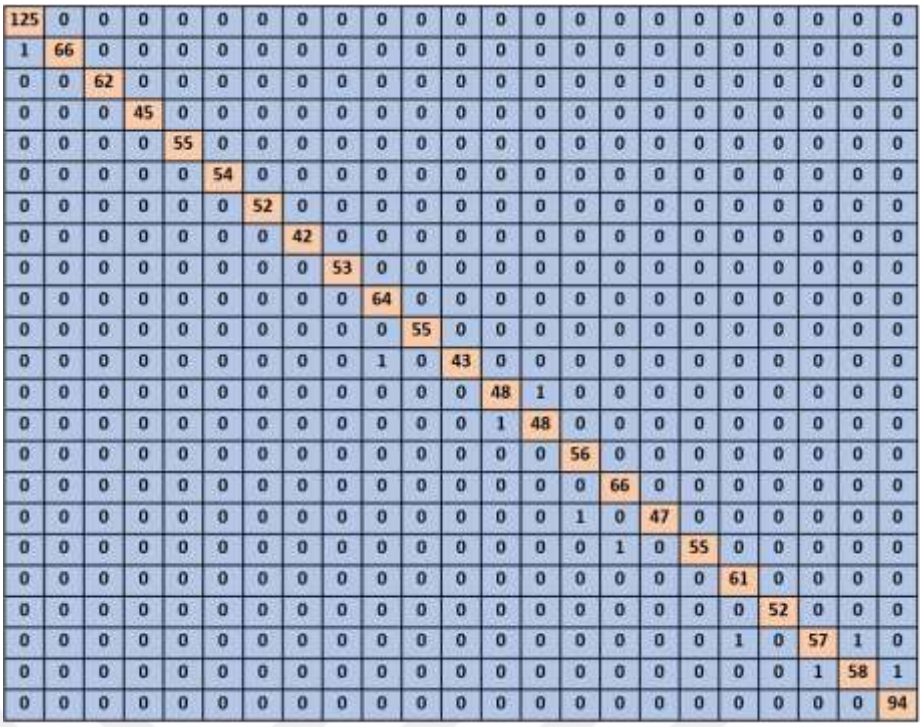


Şekil 50. LSTM için Epok sayısına göre eğitim ve validasyon doğruluk performans grafiği



Şekil 51. LSTM için Epok sayısına göre eğitim ve validasyon kayıp grafiği

Model oyun üzerinde çalışırken oyun hareketinde zaman gecikmesi olduğu görülmüştür. Bar hareketi oyun sırasında her zaman topun pozisyonunu takip etmektedir.



Şekil 52. LSTM Modeli için karışıklık matrisi

Evrişimli sinir ağları ve LSTM modelinden elde eidlen sonuçların özeti Tablo 2'de gösterilmektedir. Önerilen CNN mimarisi ile en iyi sonuçlar elde edilmiştir

Tablo 2. CNN ve LSTM Sonucunun Çıktısı

Yöntem	VGG16	Önerilen Model	ConvLstm
Doğruluk	%98.1	%99.6	% 99.4

DQN Yöntemi ile Elde Edilen Sonuçlar

Girdi olarak Resim kullanılması

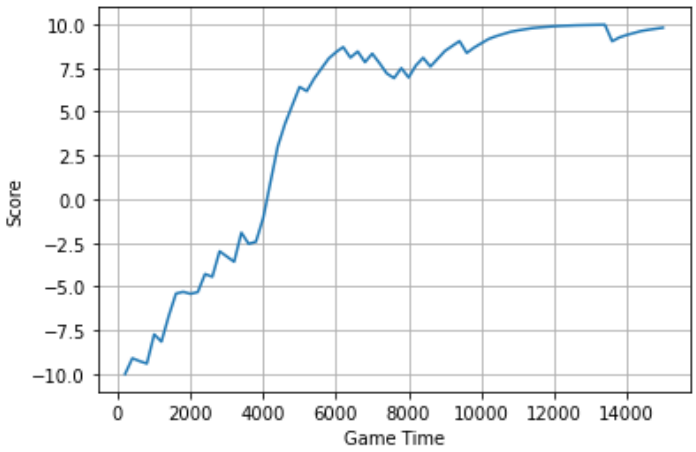
80 saatlik ağ eğitiminden sonra ağ çıkışı iki şekilde test edilmektedir:

İlk olarak, uzun süreli oyun testi; bu esnada bir tanesini bilgisayar olan oyuncunun topun doğru pozisyonu bildiği kabul edilerek topun merkezine göre barı ayarlandığında DQN'e dayalı eğitilmiş oyuncuya karşı üç saat oyun oynaması sağlanmıştır. Bu durumda her iki oyuncunun berabere kaldığı gözlemlenmiştir.

İkincisi, hız testidir. Eğitim esnasında oyun frekansı 60 seçilerek eğitim işlemi gerçekleştirilmiş, test aşamasında ise oyun frekansı 20,30,40…80 seçilerek farklı hızlarda çalışması test edilmiştir. Bu modelde oyunun sorunsuz ve gecikmesiz bir şekilde çalıştığı gözlemlenmiştir.

Girdi olarak durum vektörünün kullanılması

Oyuna, oyun frekansı 60 seçilerek başlanmıştır. Ajanın her seferinde öğrenmesini değerlendirmek ve topun ajan barını ne zaman geçtiğini algılamak için skor değeri olarak bir değişken kullanılmıştır. Bu değişken başlangıç değeri olarak -10 olarak seçilmiştir. Bundan sonra top ajan barı geçerse puanı düşmüş, aksi takdirde puanı artmıştır. Ajan eğitim puanı değeri değişim grafiği Şekil 53'te gösterilmektedir.



Şekil 53. Ajan Eğitim Puanı değişim grafiği

Yukarıdaki şekilde gösterildiği gibi, puan değeri sürekli olarak artmaktadır. Grafikte azalmanın görüldüğü noktalarda ajan puan kaybetmektedir. Ancak uzun süre devam eden oyun neticesinde gürültülü de olsa ortalama olarak ajanın puanın artma eğilimi vardır.

SONUÇLAR

Bu tez çalışmasında, masa tenisi oyun simülasyonu için CNN, LSTM ve DQN algoritmaları incelenmiş olup her bir yöntemin performansı incelenmiştir. CNN yönteminde, görüntülerin özniteliklerini çıkarmak için kullanılır. Bunlar veri kümesinde verilen etiketlerle eşleştirilir, ardından farklı bir veri kümesine uygulanarak ağ performansı test edilir. Öznitelik çıkarmak için, ağ girişine verilen görüntüye evrişim (convolution), havuzlama (maxpooling) ve düşürme (dropout) gibi farklı işlemler uygulanır. LSTM yönteminde, nesneleri video, metin veya ses gibi sıra durumundayken sınıflandırmak için kullanılır. Giriş sırasını hatırlamak için LSTM, unutma kapısı, çıkış kapısı, giriş kapısı ve güncelleme kapısı kullanılır. DQN yönteminde CNN ve LSTM'den farklı olarak Q-öğrenmenin bir uzantısıdır. DQN veri kümesine ihtiyaç duymaz, yalnızca ajanın ortamdaki durumunu alması ve verilen etiketli eylemlere bağlı olarak uyarlanmış çıktısını oluşturması için ortamda etkileşime girmesine izin verilir. Bu tez çalışmasında kullanılan CNN, LSTM ve DQN algoritmalarının karşılaştırması Tablo 3'de gösterilmiştir.

Tablo 3. CNN, LSTM ve DQN Karşılaştırması

Vektörler	CNN	LSTM	DQN
Konsept	Biyolojik Nöronlara Dayalı	Biyolojik Nöronlara Dayalı	İnsan beyni
Veri Kümesi	Etiketli Görüntüler (giriş olarak bir görüntü)	Sıra Verileri (Görüntü, Ses, Metin Gibi)	Veri kümesi yok
Tekrarlayan Bağlantı	Hayır	Evet	Kullanılan ağa bağlı
Kaybolan Gradyan	Evet	Evet	Evet
Öğrenme Yöntemleri	Eğitim veri kümesi analiz etme ve başka bir veri kümesinde test etme	Eğitim veri kümesini analiz etme ve başka bir veri kümesinde test etme	Hatalardan ders alma ve ödülü en üst düzeye çıkarma
Keşif	Hayır	Hayır	Keşfetmeden değişime uyarlanabilme
Optimal strateji	Veri kümesine bağlı	Veri kümesine bağlı	Keşiften optimal stratejiyi öğrenme

Yapılan deneyler sonucu CNN ve LSTM yöntemlerinde bilgisayarın hesaplama gücüyle ters orantılı olarak bir gecikme yaşandığı, öte yandan zaman gecikmesi olmadan yüksek performans gösteren DQN yöntemi kullanılarak bar ve topun farklı hız değerleri denenmiş, sorunsuz ve verimli çalıştığı gözlenmiştir.

KAYNAKLAR

Advani 2021. What is Machine Learning? How Machine Learning Works and future of it?, https://www.mygreatlearning.com/blog/what-is-machine-learning/ (22.4.2021).

Arulkumaran, K., Deisenroth, M. P., Brundage, M., & Bharath, A. A. (2017). Deep reinforcement learning: A brief survey. IEEE Signal Processing Magazine, 34(6), 26–38. https://doi.org/10.1109/MSP.2017.2743240

Arunkumar, K. E., Kalaga, D. V., Sai Kumar, C. M., Chilkoor, G., Kawaji, M., & Brenza, T. M. (2021). Forecasting the dynamics of cumulative COVID-19 cases (confirmed, recovered and deaths) for top-16 countries using statistical machine learning models: Auto-Regressive Integrated Moving Average (ARIMA) and Seasonal Auto-Regressive Integrated Moving Averag. Applied Soft Computing, 103(December 2019), 107161. https://doi.org/10.1016/j.asoc.2021.107161

Barros, P., Tanevska, A., Yalcin, O., Sciutti, A., & Nov, A. I. (n.d.). Incorporating Rivalry in Reinforcement Learning for a Competitive Game.

Biswal A., 2021. 7 Types of Artificial Intelligence That You Should Know in 2020, https://www.simplilearn.com/tutorials/artificial-intelligence-tutorial/types-of-artificial-intelligence (21.4.2021)

Blauch, N. M., Behrmann, M., & Plaut, D. C. (2021). Computational insights into human perceptual expertise for familiar and unfamiliar face recognition. Cognition, 208(November 2019), 104341. https://doi.org/10.1016/j.cognition.2020.104341

Brownlee J., 2019. What is Deep Learning?, https://machinelearningmastery.com/what-is-deep-learning/ (25.4.2021).

Burns E., 2021. machine learning, https://searchenterpriseai.techtarget.com/definition/machine-learning-ML (25.4.2021).

Chen, X., Xie, X., & Teng, D. (2020). Short-term Traffic Flow Prediction Based on ConvLSTM Model. Proceedings of 2020 IEEE 5th Information Technology and Mechatronics Engineering Conference, ITOEC 2020, Itoec, 846–850. https://doi.org/10.1109/ITOEC49072.2020.9141783

Dewan, T., Mohammed, S., & Chen, A. (n.d.). The Use of Reinforcement Learning in Gaming : The Breakout Game Case Study. 1–8.

Escott E.,2017. What are the 3 types of AI? A guide to narrow, general, and super artificial intelligence, https://codebots.com/artificial-intelligence/the-3-types-of-ai-is-the-third-even-possible (20.4.2021).

Fischer S. & Winkler C., 2020. Machine Learning – Basics And Definition Explained For Beginners And Managers, https://morethandigital.info/en/machine-learning-basics-and-definition-explained-for-beginners-and-managers/ (23.4.2021).

Frankenfield J., 2021. Artificial Intelligence (AI), https://www.investopedia.com/terms/a/artificial-intelligence-ai.asp (25.4.2021).

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-term Memory. Neural Computation, 9, 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

IBM Cloud Education., 2020. Machine Learning, https://www.ibm.com/cloud/learn/machine-learning (23.4.2021)

Justesen, N., Bontrager, P., Togelius, J., & Risi, S. (2020). Deep learning for video game playing. IEEE Transactions on Games, 12(1), 1–20. https://doi.org/10.1109/TG.2019.2896986

Krenker, A., Bešter, J., & Kos, A. (2011). Introduction to the artificial neural networks. Artificial Neural Networks: Methodological Advances and Biomedical Applications. InTech, 1-18.

Kumar, S. (2020). Balancing a CartPole System with Reinforcement Learning -- A Tutorial. 0. http://arxiv.org/abs/2006.04938

Li, H., Huang, J., Wang, Y., Wang, B., & Gu, C. (2020). DQN based Reinforcement Learning Algorithm for Scheduling Workflows in the Cloud DQN based Reinforcement Learning Algorithm for Scheduling Workflows in the Cloud. 1–6.

Lipton, Z. C., Berkowitz, J., & Elkan, C. (2015). A Critical Review of Recurrent Neural Networks for Sequence Learning arXiv : 1506 . 00019v4 [cs . LG] 17 Oct 2015. 1–38.

Mnih, V., & Silver D., (2013). Playing Atari with Deep Reinforcement Learning. 1–9.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). learning. Nature, 518(7540), 529–533. https://doi.org/10.1038/nature14236

Piyush & Rishabh., 2020. 3 Types of Machine Learning, https://www.newtechdojo.com/3-types-of-machine-learning/ (25.4.2021).

Shyalika C., 2019. A Beginners Guide to Q-Learning, https://towardsdatascience.com/a-beginners-guide-to-q-learning-c3e2a30a653c (28.4.2021).

Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. ArXiv 1409.1556.

Tan, Z., & Karakose, M. (2020). Comparative Study for Deep Reinforcement Learning with CNN, RNN, and LSTM in Autonomous Navigation. 2020 International Conference on Data Analytics for Business and Industry: Way Towards a Sustainable Economy, ICDABI 2020. https://doi.org/10.1109/ICDABI51230.2020.9325622

Vu, T., & Tran, L. (2020). FlapAI Bird: Training an Agent to Play Flappy Bird Using Reinforcement Learning Techniques. http://arxiv.org/abs/2003.09579

Xavier A., 2019. An introduction to ConvLSTM, https://medium.com/neuronio/an-introduction-to-convlstm-55c9025563a7 (29.4.2021).

Yamashita, R., Nishio, M., Do, R., & Togashi, K. (2018). Convolutional neural networks: an overview and application in radiology. Insights into Imaging, 9. https://doi.org/10.1007/s13244-018-0639-9

Yamini, R., & Jain, A. (2020). General artificial intelligence model for developing adaptive non playable characters in computer games. Journal of Critical Reviews, 7(6), 78–82. https://doi.org/10.31838/jcr.07.06.16

Zhang, R. (2017). Train a snake with reinforcement learning. 1–11.

ÖZGEÇMİŞ

Kişisel Bilgiler	
Adı Soyadı:	Ahmed Naji Musleh NUSARI
Doğum tarihi:	
Doğum Yeri:	
Uyruğu:	
Adres:	
Tel:	
E-mail:	
Eğitim	
Lise:	Teknik ve Fen Lisesi
Lisans:	Sana'a Üniversitesi, Mühendislik Fakültesi (Elektrik-Elektronik)
Yabancı Dil Bilgisi	
İngilizce:	Çok İyi
Arapça:	Ana dili
Türkçe	İyi
Üye Olunan Mesleki Kuruluşlar	
-	
Tezden Üretilmiş Yayınlar	
-	