YALOVA ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

- BİTİRME TEZİ -

TAKLİTLE ÖĞRENME İLE GERÇEK ZAMANLI STRATEJİ BOTU

Uğur İPEKDÜZEN

Bitirme Tezi Danışmanı: Dr. Öğr. Üyesi Adem TUNCER

YALOVA, 2022

YALOVA ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

TAKLİTLE ÖĞRENME İLE GERÇEK ZAMANLI STRATEJİ BOTU

Uğur İPEKDÜZEN
150101045

Bitirme Tezi Danışmanı : Dr. Öğr. Üyesi Adem TUNCER
2. Jüri Üyesi : Dr. Öğr. Üyesi Murat OKKALIOĞLU
3. Jüri Üyesi : Dr. Öğr. Üyesi Yunus ÖZEN

Bitirme Tezinin Dönemi : 2021 – 2022 Güz Yarıyılı

İÇİNDEKİLER

KISALTMA LİSTESİ

AI	Artificial Intelligence
AIIDE	Artificial Intelligence and Interactive Digital Entertainment
ALVINN	An Autonomous Land Vehicle In A Neural Network
BCO	Behavioural Cloning from Observation
CARL	Case-Based Reinforcement Learner
CBR	Case-Based Reasoning
CIG	Computational Intelligence and Games
CNN	Convolutional Neural Network
DQN	Deep Q-Network
DRL	Deep Reinforcement Learning
FDG	Foundations of Digital Games
GGP	General Game Playing
GTS	Game Tree Search
GVGAI	General Video Game AI
IL	Imitation Learning
IMAI	Influence Map Based Artificial Intelligence Player
i.i.d	Independent and Identically Distributed
JSON	JavaScript Object Notation
LfD	Learning from Demonstration
MDP	Markov Decision Process
NPC	Non-Player Character
RTS	Real Time Strategy
SC2LE	StarCraft II Learning Environment
SVM	Support Vector Machine
TL	Transfer Learning

ŞEKİL LİSTESİ

ÇİZELGE LİSTESİ

ÖNSÖZ

Video oyunları sosyal hayatın bir parçası haline gelmiştir. Amaçları insanları eğlendirmek olan bu sanal ortamlar, zaman içinde gelişerek sundukları dünya ve oyun içi nesnelerle veya oyuncularla etkileşim çeşitliliği beraberinde dünya keşfi, kısıtlı yol bulma veya takım taktikleri ve koordinasyon gibi oyun içinde çözülmesi gereken problemlerin doğmasına sebep olmuştur. Bununla birlikte gerçek dünya problemlerine uygulanmadan önce problem çözme tekniklerinin geliştirilip değerlendirilebildiği, kısıtlı bir ortam ve sabit bir kurallar dizisi içinde yapay zeka yöntemleri geliştirebilmek ideal bir çalışma alanı sağladığı için araştırmacıların ilgi odağı haline gelmiştir. İlk masa oyunlarına uygulanan yöntemlerden alınan başarılı sonuçlarla birlikte video oyunlarının da yaygınlaşmasıyla, giderek daha karmaşık senaryolarda yinelemeli ilerleme için potansiyel bir alan olarak gördüler ve sonunda insan düzeyinde yapay zekanın gelişmesine yol açtılar. Gerçek Zamanlı Strateji (RTS) oyunları genel olarak bu karmaşık senaryoları içerdikleri için araştırmacılar tarafından daha fazla araştırma yapılması çağrısında bulunulmuştur.

Bu tez çalışmasında basit bir RTS oyununda, daha önceki karşılaşmalarda yaptığı stratejik hamleler sonucu galip gelmiş uzman oyunculardan elde edilen gözlem verilerini taklit ederek, oyundaki temel eylemleri (kaynak toparlama, inşa etme, savunma, gelişme vs.) gerçekleştiren bir botun ne kadar başarılı olabileceği incelenmiştir. Geliştirilen çözüm, Kaggle platformunda yayınlanmış olan Lux AI Yarışması ortamı kullanılarak test edilmiştir.

Bu bitirme tezinin hazırlanması sürecinde, araştırmanın her bir aşamasında görüşleriyle beni destekleyen, samimiyetini her zaman hissettiren ve beni doğru yönde yönlendiren danışman hocam Dr. Öğr. Üyesi Adem TUNCER'e ve desteklerini hiçbir zaman eksik etmeyen, bana olan güvenlerini hiç kaybetmeyen aileme teşekkür ederim.

Şubat 2022 Uğur İPEKDÜZEN

ÖZET

Taklitle öğrenme, belirli bir görevde insan davranışını taklit etmeyi amaçlar. Bir öğrenme makinesi, gözlemler ve eylemler arasında bir eşleme öğrenerek gösterilerden bir görevi gerçekleştirmek üzere eğitilir. Taklit yoluyla öğretme fikri uzun yıllardır ortalıkta dolaşmaktadır, ancak bilgi işlem ve algılamadaki ilerlemelerin yanı sıra akıllı uygulamalara yönelik artan talep nedeniyle bu alan son zamanlarda dikkat çekmektedir. Taklitle öğrenme paradigması gün geçtikçe popülerlik kazanmaktadır çünkü karmaşık görevlerin, görevler hakkında asgari düzeyde uzman bilgisi ile öğretilmesini kolaylaştırmaktadır. Eğitilecek makine için kullanılacak verinin çok büyük miktarda olması, eğitimden elde edilecek kesin sonucu olumlu yönde etkiler. Ancak bununla birlikte, veriseti büyüdükçe eğitimin zaman maliyeti de artar. Bu yüzden, zaman maliyetinden tasarruf etmek için ön eğitimli modellerden elde edilen bilgileri başka bir makineye aktarmak için transfer öğrenimi kullanımı, günümüz teknolojisiyle büyüyen veriler üzerinde yapılan yapay zeka çalışmaları için hayat kurtarıcı olabilmektedir. Bu bitirme tezinde, Kaggle'da 16 Ağustos 2021 tarihinde yayınlanmış olan Lux AI Yarışması'nda, yarışmanın adını veren bir RTS oyun ortamı üzerinde gerçekleştirilen 1v1 senaryolarda rakip takıma kıyasla olabildiğince çok fazla alana sahip olabilmek için birim ve kaynak problemini çözen bir bot geliştirilmiştir. Bu problemi çözmek için, taklitle öğrenmenin en basit yöntemi olan davranışsal klonlama ile yarışmacıların günlük karşılaşmalarda uyguladıkları stratejik hamlelerin, bir CNN ağında eğitimi sonucunda üretilen ön eğitimli modellerdeki verilerin transfer öğrenimi ile tez kapsamında geliştirilen bota aktarılmasıyla yarışma sıralamasında en yüksek puana sahip olmak hedeflenmiştir.

Anahtar Kelimeler: Gerçek Zamanlı Strateji, Taklitle Öğrenme, Davranışsal Klonlama, Lux AI Challenge 2021, Kaggle

ABSTRACT

Imitation learning aims to imitate human behavior in a particular task. A learning machine is trained to perform a task from demonstrations by learning a mapping between observations and actions. The idea of teaching by imitation has been around for many years, but this area has been gaining attention recently due to the increasing demand for smart applications as well as advances in computing and perception. The imitation learning paradigm is gaining popularity day by day because it facilitates the teaching of complex tasks with minimal expert knowledge of the tasks. The large amount of data to be used for the machine to be trained positively affects the final result to be obtained from the training. However, the larger the dataset, the higher the time cost of training. Therefore, the use of transfer learning to transfer the information obtained from pre-trained models to another machine to save time costs can be life-saving for artificial intelligence studies on data growing with today's technology. In this dissertation, a bot that solves the unit and resource problem in order to have as much space as possible compared to the opposing team in 1v1 scenarios performed on an RTS game environment, which gives the name of the competition, in the Lux AI Challenge published on Kaggle on August 16, 2021, has been developed. In order to solve this problem, it is aimed to have the highest score in the competition ranking by transferring the strategic moves applied by the competitors in daily encounters, which is the simplest method of learning by imitation, to the bot developed within the scope of the thesis with the transfer learning of the data in the pre-trained models produced as a result of training on a CNN network.

Keywords: Real Time Strategy, Imitation Learning, Behavioral Cloning, Lux AI Challenge 2021, Kaggle

1 GİRİŞ

Günümüz sosyal hayatın bir parçası haline gelen video oyunları, değişen derecelerde karmaşıklığa sahip iyi tanımlanmış zorlukları temsil ettiği için yapay zeka araştırmaları için kullanılan, anlaşılması kolay ve yapay zeka algoritmaları ile insan oyuncuların performansını karşılaştırma fırsatı sunan bir alandır (Šustr vd., 2018; Rohlfshagen vd., 2018). Bir oyunun arka planında iyi bir yapay zeka tekniğinin bulunması, ticari bilgisayar oyunlarında eğlence ve tekrar oynanabilirlik için en önemli faktörlerden biridir.

Oyunlar, akademide yapay zeka çalışmalarında her zaman bir geliştirme ve test ortamı olarak yaygın olarak kullanılmıştır. Oyun yapay zekâsı üzerine yapılan ilk araştırmalar özellikle satranç ve dama gibi klasik masa oyunlarına odaklanmış olsa da, sonraki zamanlarda video oyunları araştırmacıların ilgisini çekmiştir. GVGAI yarışması (Perez-Liebana vd., 2019), GGP yarışması (Genesereth ve Björnsson, 2013) ve Geometry Friends Game AI yarışması (Prada, Lopes vd., 2015) gibi uluslararası yarışmalar ve CIG, AIIDE ve FDG konferansları gibi her yıl oyun yapay zekâsına adanmış uluslararası konferanslar bu durumun en büyük kanıtlarıdır. Bu artan ilgi, oyun endüstrisinin daha eğlenceli, zorlu, ilgi çekici ve nihayetinde insan oyunculara şu şekilde daha kişiselleştirilmiş bir deneyim sağlayan oyunlar yaratmak için daha karmaşık davranışlar ve beceriler sergileyen, daha doğal, insan gibi davranan daha karmaşık NPC'ler (Rohlfshagen vd., 2018; Gomez ve Miikkulainen, 1997; Llargues Asensio vd., 2014; Mahmoud vd., 2014; Torrey, 2010) gibi çözümler bulmaya yönelik artan ticari ilgisiyle de körüklenmiştir.

Yapay zekâ Go (Silver vd., 2016), satranç, tavla veya dama gibi çeşitli oyunlarda gerçek zamanlı oyunlar söz konusu olduğunda başarılı bir şekilde uygulanmış olsa da, gerçek zamanlı oyunlarda kararların gerçek zamanlı olarak verilmesi gerektiği ve arama uzayının çok büyük olması ve bu nedenle öğrenme için gerçek bir yapay zeka içermemeleri nedeniyle tamamen işe yarayan bir çözüm olamamıştır. Dolayısıyla araştırmacıların ilgisi, gerçek zamanlı strateji (RTS) oyunları tarafından temsil edilen daha karmaşık bir mücadeleye kaymıştır.

Kişinin kendi zamanını alma yeteneğine sahip olduğu sıra tabanlı strateji oyunlarının aksine, RTS oyunlarında tüm hareket, inşaat, dövüş vb. gerçek zamanlı olarak gerçekleşir. Tipik bir RTS oyununda ekran, binalar, birimler ve arazi ile oyun dünyasından oluşan bir harita alanı içerir. Bir RTS oyununda genellikle birkaç oyuncu vardır. Oyuncular dışında katılımcılar, birimler ve yapılar olarak adlandırılan çeşitli oyun varlıkları vardır. Bunlar oyuncuların kontrolü altındadır ve oyuncuların varlıklar üzerindeki kontrollerini kullanarak rakip oyuncuların varlıklarını kurtarması veya yok etmesi gerekir.

Günümüzde, önceki paragraflarda bahsedilen yarışmalardan ilham alınarak oluşturulan örneklerden bir tanesi de Lux AI Yarışması'dır [1]. Lux AI tamamen piyasadaki rakip RTS oyunlarından bağımsız, sadece bir RTS oyunundan beklenen temel yapıya sahip bir ortam sunan, kendine özgü, minimalist bir RTS botu geliştirme ortamıdır. Bu yarışmanın Kaggle üzerinden de erişilebilir olması RTS oyunları kullanılarak geliştirilen yapay zeka çalışmalarının öne çıkmasına olanak sağlamaktadır. Bu tez çalışmasında, bu ortamı kullanarak 1v1 senaryoda galip gelmiş rakiplerin hamlelerinin taklitle öğrenme paradigmasından davranışsal klonlama kullanılarak birden fazla ön eğitimli model üretimi gerçekleştirilmiştir ve yarışacak botun

transfer öğrenimi yardımıyla, bu modellerin iyi sonuç vermiş çıktılarıyla, karşı rakibe üstünlük kurabilecek ortalama bir sonuç elde edilmek istenmiştir.

Gerçek zamanlı strateji oyunu kavramının ne olduğu, geleneksel masa oyunlarıyla arasındaki farklar ve yapılan yapay zeka araştırmaları doğrultusunda karşılaşılan ortak zorluklar tezin literatür araştırmasının ilk bölümünde anlatılmıştır.

Tezin literatür araştırmasının ikinci bölümünde, temel makine öğrenmesi paradigmaları olan denetimli öğrenme, denetimsiz öğrenme ve pekiştirmeli öğrenme, bunları oluşturan temel unsurlar ve bu paradigmalar kullanılarak çözülen problem çeşitleri tanıtılmıştır. Son olarak, bu paradigmalardan esinlenerek ortaya çıkan taklitle öğrenmenin tanımından ve bu paradigmanın en basit şekli olan davranışsal klonlamadan bahsedilmiştir.

Tezin literatür araştırmasının üçüncü bölümünde, transfer öğrenimi ve süreci, geleneksel makine öğrenmesi ile arasındaki farkı, aktarım çeşidine gore transfer öğrenimi stratejileri ve teknikleri anlatılmıştır.

Tezin literatür araştırmasının son bölümünde ise araştırmada yer alan kavramlarla ilgili ve tezde savunulan yöntemi destekleyen örnek çalışmalara değinilmiştir.

Tezin deneysel çalışmasnın ilk bölümünde, tezde anlatılan çözümün uygulanacağı Kaggle platformuna özgü Lux AI Yarışması'nın amacı, kuralları, yarışma ortamındaki inşa edilebilen yapılar, toplanılması gereken kaynaklar ve kullanılabilen birimler tanıtılmıştır.

Tezin deneysel çalışmasnın ikinci bölümünde, yarışma için geliştirilen davranışsal klonlama için kullanılan derin ağ mimarisi, eğitilmesinde kullanılacak verisetinin içeriği ve uygulanan hiperparametre değerleri hakkında bilgiler verilmiştir.

Genel sonuçlar ve öneriler kısmında ise çalışmada elde edilen sonuçlar yorumlanmış ve gelecek çalışmalar için birtakım öneriler sunulmuştur.

2 LİTERATÜR ARAŞTIRMASI

2.1 Gerçek Zamanlı Strateji

Gerçek Zamanlı Strateji (RTS), oyuncuların kaynak toplayarak ve üsler kurarak, ekonomik ve stratejik görevleri yönettikleri, yeni teknolojiler ve eğitim birimleri araştırarak askeri güçlerini artırdıkları ve onları rakiplerine karşı savaşa yönlendirdikleri bir video oyunu türüdür (Certicky ve Churchill, 2017). Yapay zeka araştırması perspektifinden bakıldığında, RTS oyunları planlama, belirsizlikle başa çıkma, alan bilgisi kullanımı, görev ayrıştırma, mekânsal akıl yürütme ve makine öğrenimi alanlarında ilginç zorluklar ortaya çıkarmaktadır (Ontañón vd., 2013; Churchill vd., 2016). Teorik olarak bakıldığında, RTS oyunları ile geleneksel masa oyunları arasındaki temel farklar şunlardır:

- Birden fazla oyuncunun aynı anda hareket edebildiği eşzamanlı hareket oyunlarıdır ve oyunda gerçekleştirilen eylemler kalıcıdır, yani eylemler anlık değildir ancak tamamlanması biraz zaman alır.
- RTS oyunları gerçek zamanlıdır, yani aslında her oyuncunun bir sonraki hamleye karar vermek için çok az zamanı vardır.
- Çoğu RTS oyunu kısmen gözlemlenebilir, oyuncular haritanın yalnızca keşfedilen kısmını görebilir. Buna savaş sisi denir.
- Çoğu RTS oyunu deterministik değildir. Bazı eylemlerin başarı şansı vardır.
- RTS oyun alanının yüksek karmaşıklığı, savaşta birim mikro yönetimi, tehdide duyarlı yol bulma, kaynak tahsisi, rakip modelleme veya yapı siparişi optimizasyonu gibi daha küçük alt problemlere ayrışmasını teşvik eder. Bununla birlikte, bu alt problemlerin çoğu aşırı basitleştirmeden diğerlerinden ayrıştırılamaz, yani bir problemi etkin bir şekilde çözmek için genellikle diğer problemlerin de dikkate alınması gerekir. Örneğin, savaş birimlerinin mikro yönetimini çözmek için belirli bir savaş durumlarında rakibin nasıl tepki vereceğini tahmin etmek için tehdide duyarlı bir yol bulma algoritmasına (düşman birimlerini öldürülmeden kuşatabilmek için) veya rakip modelleme algoritmasına ihtiyaç duyulabilir.

Bu nedenlerden dolayı, oyun ağacı arama gibi klasik masa oyunlarını oynamak için kullanılan standart teknikler, bir düzeyde soyutlama veya başka bir basitleştirme tanımı olmadan RTS oyunlarını çözmek için doğrudan uygulanamaz.

RTS oyunları (Buro, 2003) için yapay zekâda yapılan ilk araştırmalar ışığında, aşağıdaki altı zorluk ortaya çıkmıştır:

- Kaynak yönetimi
- Belirsizlik altında karar verme
- Mekansal ve zamansal akıl yürütme
- İşbirliği (birden çok yapay zekâ arasında)
- Rakip modelleme ve öğrenme
- Düşmanca gerçek zamanlı planlama

Birçoğunda önemli çalışmalar varken, işbirliği gibi diğer başlıklar henüz önemsenmemiştir. Ayrıca, bu alandaki son araştırmalar, var olan büyük miktardaki alan bilgisinden (stratejiler, inşa emirleri, tekrarlar vb.) nasıl yararlanılacağı gibi birkaç ek araştırma zorluğu belirlemiştir. Bir RTS oyunun yapay zekâsındaki mevcut zorluklar, altı ana farklı alanda gruplandırılmış şekilde aşağıda yer almaktadır.

2.1.1 Planlama

RTS oyunlarındaki durum uzayının boyutu, satranç veya Go gibi geleneksel masa oyunlarından çok daha büyüktür. Bununla birlikte, belirli bir zamanda gerçekleştirilebilecek eylemlerin sayısı da çok daha fazladır. Bu nedenle, GTS gibi standart çekişmeli planlama yaklaşımları doğrudan uygulanamaz. RTS oyunlarında planlama, birden fazla soyutlama seviyesine sahip olarak görülebilmektedir. Daha yüksek bir seviyede, oyuncular oyunda güçlü bir ekonomi geliştirmek için uzun vadeli planlama yeteneklerine ihtiyaç duyarlar; düşük seviyede, araziyi ve rakibi dikkate alarak savaşmak için bireysel birimlerin koordinasyon içinde hareket ettirilmesi gerekir. Bu büyük planlama problemlerini örnekleme veya hiyerarşik ayrıştırma yoluyla çözebilecek teknikler henüz mevcut değildir.

2.1.2 Öğrenme

Doğrudan çekişmeli planlama tekniklerini kullanarak RTS oyunlarını oynamanın zorlukları göz önüne alındığında, birçok araştırma grubu dikkatlerini öğrenme tekniklerine çevirmiştir. RTS oyunlarında öğrenme problemi üç çeşittir:

- **Önceden öğrenme:** Mevcut tekrarlar gibi mevcut verilerden veya önceden uygun stratejileri öğrenmek için belirli haritalar hakkındaki bilgilerden nasıl yararlanılabilir?
- **Oyun içi öğrenme:** Botlar, oyun oynarken oyunlarını geliştirmelerine olanak tanıyan çevrimiçi öğrenme tekniklerini nasıl uygulayabilir?
- **Oyunlar arası öğrenme:** Bir sonraki oyunda zafer şansını artırmak için kullanılabilecek bir oyundan neler öğrenilebilir?

2.1.3 Belirsizlik

RTS oyunlarında iki ana belirsizlik türü vardır. İlk olarak, oyun kısmen gözlemlenebilir ve oyuncular tüm oyun haritasını satrançta olduğu gibi gözlemleyemezler, ancak rakibin ne yaptığını görmek için keşif yapmaları gerekir. Bu tür bir belirsizlik, görülmüş olandan neyin mümkün olduğunu çıkarmak için iyi bir keşif ve bilgi temsili ile azaltılabilir. İkincisi, oyunların çekişmeli olmasından kaynaklanan bir belirsizlik de vardır ve bir oyuncu, rakiplerin gerçekleştireceği eylemleri tahmin edemez. Bu tür bir belirsizlik için yapay zekâ, insan oyuncu olarak, rakibin yapması muhtemel olan eylemlere dair yalnızca mantıklı bir model oluşturabilir.

2.1.4 Mekânsal ve zamansal akıl yürütme

Mekânsal akıl yürütme, arazi kullanımının her yönü ile ilgilidir. Bina yerleştirme veya üs genişletme gibi görevlerde yer alır. İlkinde, oyuncunun hem işgallere karşı bir duvar oluşturarak onları korumak hem de büyük birimlerin sıkışabileceği kötü konfigürasyonlardan kaçınmak için kendi üslerine konumlandırma inşa etmeyi dikkatlice düşünmesi gerekir. Üs genişletmede, oyuncu, kendi konumu ve rakibin üsleri ile ilgili olarak yeni bir üs inşa etmek için uygun yerleri seçmelidir. Son olarak, mekânsal akıl yürütme taktiksel akıl yürütmenin anahtarıdır.

Oyuncuların savaş için birimleri nereye yerleştireceklerine karar vermeleri gerekir. Örneğin, rakibin birimleri bir darboğaza girdiğinde çatışmaya girmeyi tercih eder.

Bir başka örnek ise, düşman alçak yerdeyken kendi birimlerinin yüksek yerde olması her zaman bir avantajdır, çünkü alçak zemindeki birimlerin yüksek zeminde görüşü yoktur.

Benzer şekilde, zamansal akıl yürütme, taktiksel veya stratejik akıl yürütmenin olmazsa olmazıdır. Örneğin, bir üstünlük elde etmek için saldırıların ve geri çekilmelerin zamanlamasına doğru karar verilmelidir. Daha yüksek bir stratejik düzeyde, oyuncuların yükseltme, bina inşaatı, strateji değiştirme vb. uzun vadeli ekonomik eylemleri ne zaman gerçekleştirecekleri konusunda muhakeme yapmaları gerekir.

2.1.5 Etki alanı bilgi sömürüsü

Satranç gibi geleneksel masa oyunlarında araştırmacılar, alfa-beta arama algoritmaları, kapsamlı açılış kitapları veya oyun sonu tabloları tarafından kullanılacak iyi değerlendirme işlevleri oluşturmak için mevcut alan bilgisinden büyük miktarda yararlanırlarken, RTS oyunları söz konusu olduğunda önemli ölçüde büyük miktardaki alan bilgisinin (formlarda veya strateji kılavuzlarında, tekrarlarda vb.) botlar tarafından nasıl sömürülebileceği hala belirsizdir. Bu alandaki çalışmaların çoğu iki ana yöne odaklanmıştır: Bir yandan araştırmacılar tarafından mevcut stratejileri botlara kodlamanın yolları aranır. Böylece botların her bir zaman adımında her bir birim tarafından hangi eylemlerin yürütüleceğine karar verme sorununu çözmek yerine, yalnızca hangi stratejilerin konuşlandırılacağına karar vermesi gerekmektedir. Öte yandan stratejilerin, eğilimlerin veya planların öğrenilmeye çalışıldığı büyük tekrar verisetleri oluşturulmaktadır (Weber ve Mateas, 2009; Synnaeve ve Bessiere, 2012). Bununla birlikte, StarCraft gibi oyunlar oldukça karmaşıktır ve bu tür verisetlerinden otomatik olarak nasıl öğrenileceği hala açık bir sorundur.

Önceki tüm bu nedenlerden dolayı, StarCraft olarak oyun oynamaya yönelik mevcut yaklaşımların çoğu, bir RTS oyunu oynama problemini bağımsız olarak çözülecek daha küçük problemler koleksiyonuna ayrıştırarak çalışır. Spesifik olarak, ortak alt bölümler şu şekildedir:

- **Strateji**: Üst düzey karar verme sürecine karşılık gelir. Bu, oyun anlayışı için en yüksek soyutlama seviyesidir. Belirli bir rakibe karşı etkili bir strateji veya karşı strateji bulmak, RTS oyunlarının kilit özelliğidir. Bir oyuncunun sahip olduğu tüm birimler ve binalarla ilgilidir.
- **Taktikler:** Mevcut stratejinin uygulanmasıdır. Ordu ve bina konumlandırma, hareketler, zamanlama vb. taktikler bir grup birimle ilgilidir.
- **Reaktif kontrol:** Taktiklerin uygulanmasıdır. Bu, savaş sırasında hareket etme, hedef alma, ateş etme, kaçma, vur-kaç tekniklerinden oluşur ve belirli bir birime odaklanır.
- **Arazi analizi:** Haritayı oluşturan bölgelerin analizinden oluşur: dar noktalar, mineraller ve gaz yerleşimleri, alçak ve yüksek yürünebilir alanlar, adalar, vb.
- **İstihbarat toplama:** Rakip hakkında toplanan bilgilere karşılık gelir. Savaş sisi nedeniyle, oyuncular düşman üslerini tespit etmek ve gözetlemek için düzenli olarak gözcüler göndermelidir.

Buna karşılık, insanlar StarCraft oynadığında, karar verme süreçlerini genellikle çok farklı bir şekilde bölerler. StarCraft oyuncuları tipik olarak iki görevden bahseder:

- **Mikro:** Birimleri ayrı ayrı kontrol etme yeteneğidir. İyi bir mikro oyuncu genellikle birimlerini daha uzun süre canlı tutar.
- **Makro:** Birim üretme ve birim üretiminizin akışını sürdürmek için uygun zamanlarda genişletme yeteneğidir. İyi bir makro oyuncu genellikle daha büyük bir orduya sahiptir.

Yaygın olarak bir önceki görev ayrıştırması kullanılıyor olsa da, bu görevlerin her birini ele alan bireysel yapay zeka tekniklerinin iletişim kurabilmesi ve birlikte etkili bir şekilde çalışabilmesi, çakışmaları çözmesi, aralarındaki kaynaklara öncelik vermesi vb. için mimariler tasarlamak önemli bir zorluk oluşturmaktadır. Ayrıca, yukarıdaki görev ayrıştırması tek olası yaklaşım sayılmaz. IMAI (Miles, 2007) gibi bazı sistemler, oyunu çok daha küçük görevlere böler ve daha sonra her bir görevi başarmanın beklenen faydalarına bağlı olarak kaynaklara atanır.

2.2 Taklitle Öğrenme

Taklitle öğrenmeyi anlatmaya başlamadan makine öğrenmesindeki öğrenme paradigmalarını hatırlamak gerekmektedir. Bu paradigmalar temel olarak üç çeşittir: Denetimli Öğrenme, Denetimsiz Öğrenme ve Pekiştirmeli Öğrenme. Detaylı bilgi için [2] ve [3] websitelerine gidilebilir.

Denetimli öğrenme, çıktı değişkenlerinden çıktı değişkenine eşleme işlevini öğrenmek için bir algoritma kullanılmasıdır. Amaç, eşleme işlevini o kadar iyi tahmin etmektir ki, yeni girdi verileri olduğunda, bu veriler için çıktı değişkenlerini tahmin edebilir hale gelmektir.

Denetimli öğrenmede, eğitim verisetinden bir algoritma öğrenme süreci, öğrenme sürecini denetleyen bir uzman olarak düşünüldüğünde, algoritma eğitim verileri üzerinde iteratif olarak tahminler yapılır ve uzman tarafından düzeltilir. Algoritma kabul edilebilir bir performans düzeyine ulaştığında öğrenme durur.

Denetimli öğrenme problemleri, regresyon ve sınıflandırma problemleri olarak gruplandırılabilir. Sınıflandırma ve regresyon üzerine kurulu bazı yaygın problem türleri, sırasıyla öneri ve zaman serisi tahminini içerir. Denetimli makine öğrenimi algoritmalarının bazı popüler örnekleri şunlardır:

- Regresyon problemleri için lineer regresyon
- Sınıflandırma ve regresyon problemleri için rastgele orman
- Sınıflandırma problemleri için SVM

Denetimsiz öğrenme, yalnızca girdi verilerinin olduğu ama karşılık gelen çıktı değişkenlerinin olmamasıdır. Amaç, veriler hakkında daha fazla bilgi edinmek için verilerdeki temel yapıyı veya dağılımı modellemektir. Denetimli öğrenmenin aksine ne doğru cevaplar ne de uzman

vardır. Algoritmalar, verilerdeki ilginç yapıyı keşfetmek ve sunmak için kendi tasarımlarına bırakılmıştır.

Denetimsiz öğrenme problemleri ise, kümeleme ve ilişkilendirme problemleri olarak gruplandırılabilir. Denetimsiz öğrenme algoritmalarının bazı popüler örnekleri şunlardır:

- k-kümeleme problemleri için araçlar.
- Birliktelik kuralı öğrenme problemleri için Apriori algoritması.

Pekiştirmeli öğrenme, amaca yönelik ne yapılması gerektiğini öğrenen bir makine öğrenmesi yaklaşımıdır. Pekiştirmeli öğrenmede ajan adı verilen öğrenen makine, karşılaştığı durumlara göre bir politika izleyerek tepki verir ve bunun karşılığında da sayısal bir ödül sinyali alır. Ajan, aldığı bu ödül puanını maksimuma çıkartmak için çalışır. İçinde bulunan durumdan ve o durumu takip eden diğer durumlardan beklenen ödüller toplanarak bir durum değeri oluşturur ve bu uzun vadede neyin iyi neyin kötü olduğunu ifade eder. Bu şekilde çalışan deneme yanılma yöntemi, pekiştirmeli öğrenmenin en ayırt edici özelliğidir.

Pekiştirmeli öğrenme, MDP denilen bir model kullanmaktadır. MDP'nin en önemli üç özelliği; algılama, eylem ve hedeftir. Pekiştirmeli öğrenme yaklaşımı makine öğrenmesinin denetimli öğrenme ve denetimsiz öğrenme yaklaşımlarından farklıdır.

Pekiştirmeli Öğrenme sürecindeki en önemli zorluklar, keşif ve sömürü kavramlarının uygulamaya geçirilmesidir. Ajanın daha fazla ödül elde etmesi için geçmişte denediği ve pozitif ödül aldığı eylemleri seçmelidir. Ajan ödül elde etmek için daha önce deneyimlediği eylemlerden yararlanır, ancak karşılaştığı bir durumda daha fazla ödül alabileceği eylemler varsa bunları da keşfetmelidir. Böylece ajan, çeşitli eylemler denemeli ve en iyi sonuç alabildiklerini aşamalı olarak desteklemelidir (Sutton ve Barto, 2018).

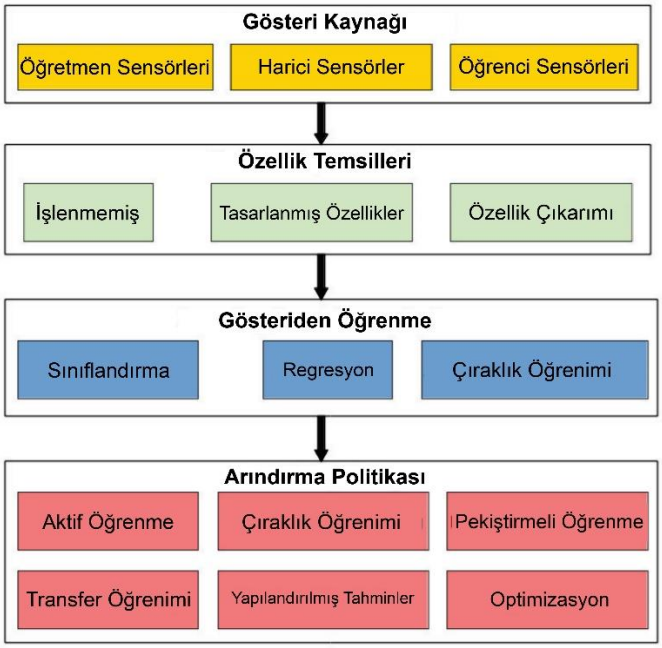
Taklitle öğrenme, belirli bir görevi gösteren bir uzmanı gözlemleyerek bir ajanın beceri veya davranış edinmesini ifade eder. Nörobilimden kaynaklanan ilham ve temel ile taklit öğrenme, makine zekası ve insan bilgisayar etkileşiminin önemli bir parçasıdır ve erken bir noktadan itibaren robotiğin geleceğinin ayrılmaz bir parçası olarak görülmüştür (Schaal, 1999). Bazen öğrencinin eylemlerinin inandırıcı olması ve doğal görünmesi önemlidir. Bu, öğrencinin performansının yalnızca bir insan gözlemcinin algısı kadar iyi olduğu birçok robotik alanda ve insan bilgisayar etkileşiminde gereklidir. Bu nedenle, bir öğrenciye bir dizi toplanmış örnekten istenen davranışı öğretmek daha uygundur. Bununla birlikte, nesnelerin konumları veya yetersiz gösterimler gibi görevdeki farklılıklar nedeniyle, uzmanın hareketinin doğrudan taklidinin yeterli olmadığı sıklıkla görülen bir durumdur. Bu nedenle, transferle öğrenme tekniklerinin, verilen gösterimlerden görünmeyen senaryolara genelleyebilecek bir politika öğrenebilmesi gerekir. Bu nedenle bir taklitle öğrenme ajanı, belirleyici bir şekilde uzmanı kopyalamak yerine görevi yerine getirmeyi öğrenir.

Taklitle öğrenme, uzmanın davranışı ve manipüle edilmiş nesneler de dâhil olmak üzere çevredeki çevre hakkında bilgi çıkararak ve durum ile gösterilen davranış arasında bir haritalamayı öğrenerek çalışır. Geleneksel makine öğrenmesi algoritmaları, yüksek serbestlik derecesine sahip yüksek boyutlu ajanlara ölçeklenememesi nedeniyle (Kober ve Peters, 2010)

insanlarda motor fonksiyonları taklit edebilmek için yeterli temsiller ve tahminler oluşturmak için özel algoritmalara ihtiyaç vardır.

Örneklerin özellik ve etiket çiftlerini temsil ettiği geleneksel denetimli öğrenmeye benzer şekilde, taklitle öğrenmede örnekler durum ve eylem çiftlerini gösterir. Durum, ilgili eklemlerin konumu ve hızları ve varsa bir hedef nesnenin durumu (konum, hız, geometrik bilgi vb.) dâhil olmak üzere ajanın mevcut pozunu temsil eder. Bu nedenle, MDP'ler kendilerini doğal olarak taklitle öğrenme problemlerine borçludur ve genellikle uzman gösterimlerini temsil etmek için kullanılır. Markov özelliği, sonraki durumun yalnızca önceki duruma ve eyleme bağlı olduğunu belirtir, bu da önceki durumları durum temsiline dâhil etme ihtiyacını hafifletir (Kober vd., 2013).

Şekil 2.1'de bir taklitle öğrenme sürecinin iş akışını göstermektedir. Süreç, daha sonra durum-eylem çiftleri olarak kodlanan bir uzmandan örnek deliller alarak başlar. Bu, farklı algılama yöntemleriyle gerçekleştirilebilir. Sensörlerden gelen veriler daha sonra ajanın durumunu ve çevresini tanımlayan özellikleri çıkarmak için işlenir. Özellikler, gösterilen davranışı taklit edecek bir politika öğrenmek için kullanılır. Bununla birlikte, durum ve eylem arasında doğrudan bir eşleştirmeyi öğrenmek, gerekli davranışı elde etmek için genellikle yeterli değildir. Bu, gösterimlerin elde edilmesindeki hatalar, uzman ve öğrenen iskeletlerindeki farklılıklar veya yetersiz gösterimler gibi birtakım sorunlardan kaynaklanabilir. Ayrıca, öğrenci tarafından gerçekleştirilen görev, ortamdaki değişiklikler, engeller veya hedefler nedeniyle gösterilen görevden biraz farklı olabilir. Bu nedenle, taklitle öğrenme sıklıkla öğrencinin öğrenilen eylemi gerçekleştirmesini ve öğrenilen politikayı görevin performansına göre yeniden optimize etmesini gerektiren başka bir adımı içerir. Bu kendini geliştirme, ölçülebilir bir ödülle ilgili olarak elde edilebilir veya örneklerden öğrenilebilir. Bu yaklaşımların çoğu, pekiştirmeli öğrenme başlığı altında toplanır. Son olarak, ajanın politikayı gerçekleştirmesine ve performansına dayalı olarak iyileştirmesine izin vererek politika geliştirilebilir. Bu adım bir uzmanın girdisini gerektirebilir veya gerektirmeyebilir. Politika geliştirmeyi öğrenme sonrası bir adım olarak düşünmek sezgisel olabilir, ancak çoğu durumda gösterilerden öğrenme ile bağlantılı olarak gerçekleşir.



Şekil 2.1 Taklitle Öğrenme Akış Şeması (Hussein vd., 2017)

Taklitle öğrenmenin en basit biçimi, denetimli öğrenmeyi kullanarak uzmanın politikasını öğrenmeye odaklanan davranışsal klonlamadır. Davranışsal klonlamaya önemli bir örnek, sensör girdilerini direksiyon açılarıyla eşleştirmeyi ve otonom sürüşü öğrenen sensörlerle donatılmış bir araç olan ALVINN'dir (Pomerleau, 1989) ve aynı zamanda genel olarak taklitle öğrenmenin ilk uygulamasıdır.

Davranışsal klonlamanın çalışma şekli oldukça basittir: Uzmandan deliller toplanır, toplanan deliller i.i.d durum-eylem çiftlerine dönüştürülür ve denetimli öğrenme yardımıyla kayıp fonksiyonu en aza indirilerek hedef politika öğrenilir.

Bazı uygulamalarda davranışsal klonlama mükemmel bir şekilde çalışabilir. Bununla birlikte, vakaların çoğu için davranışsal klonlama oldukça sorunlu olabilir. Bunun ana nedeni i.i.d varsayımından kaynaklanır: Denetimli öğrenme, durum-eylem çiftlerinin i.i.d'ye dağıtıldığını varsayarken, MDP'de belirli bir durumdaki bir eylem, önceki varsayımı bozan bir sonraki durumu tetikler. Bu aynı zamanda, farklı durumlarda yapılan hataların toplandığı anlamına gelir, bu nedenle ajan tarafından yapılan bir hata, onu uzmanın hiç ziyaret etmediği ve ajanın hiçbir zaman eğitim almadığı bir duruma kolayca getirebilir. Bu tür durumlarda davranış tanımsızdır ve ciddi hatalara yol açabilir.

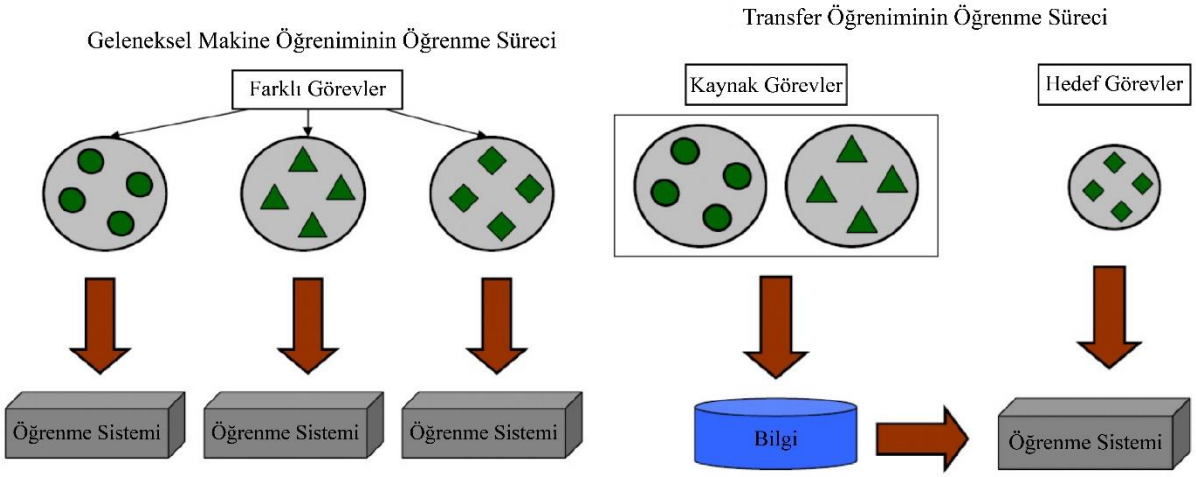
Yine de, davranışsal klonlama belirli uygulamalarda oldukça iyi çalışabilir. Başlıca avantajları basitliği ve verimliliğidir. Uygun uygulamalar, uzun vadeli planlamaya ihtiyaç duyulmayan, uzmanların yörüngelerinin durum uzayını kapsadığı ve bir hatanın ölümcül sonuçlara yol açmadığı uygulamalar olabilir.

2.3 Transfer Öğrenimi

Transfer öğrenimi, makine öğrenmesi yöntemlerinin de aynı insanda olduğu gibi bir problemi çözerken elde edilen bilgiyi saklayıp, başka bir problem ile karşılaştığında o bilgiyi

kullanmasıdır. Transfer öğrenimi ile önceki bilgiler kullanılarak daha az eğitim verisi ile daha yüksek başarı gösteren ve daha hızlı öğrenen modeller elde edilir.

Şekil 2.2'de de görüldüğü üzere, geleneksel makine öğrenmesi modelleri oluşturma ve eğitme yaklaşımı ile transfer öğrenimi ilkelerini izleyen bir metodoloji kullanma arasında büyük bir fark vardır. Normalde makine öğrenmesinde her bir görev için ayrı bir "sıfırdan öğrenme" gerçekleştirilir. Ancak bazı görevlerden öğrenilen bir takım bilgileri başka görevlerde de kullanmak mümkün ve avantajlı olacağından kaynak görevlerden elde edilen bilgi hedef görevin çözümü için kullanılmaktadır. Transfer öğrenimi ile daha önceden eğitilmiş modellerden elde edilen özellikler, ağırlıklar vb. yeni bir görev için kullanılmaktadır. Bu yöntemin işe yaraması için aktarılacak bilgilerin genel bilgiler olması gerekir, yani kaynak göreve özgü olmak yerine hem kaynak hem de hedef görevler için uygun olan bilgiler aktarılır.



Şekil 2.2 Geleneksel Makine Öğrenmesi ve Transfer Öğrenimi Arasındaki Farklı Öğrenme Süreçleri **(Pan & Yang, 2010)**

Bir örnek yardımıyla anlatmak gerekirse: Bir restoranın sınırlı bir alanı içindeki görüntülerdeki nesnelerin tanımlandığı bir görev T1 olsun. Bu görev için veriseti göz önüne alındığında, bir model eğitilir ve aynı etki alanından (restoran) görünmeyen veri noktalarında iyi performans göstermesi (genelleme) için ayarlanır. Geleneksel denetimli makine öğrenmesi algoritmaları, belirli alanlarda gerekli görevler için yeterli eğitim örneği mevcut olmadığında bozulmaya uğrar. Şimdi bir park veya kafedeki görüntülerden nesneleri algılama üzerine bir görev T2 olsun. İdeal olarak, T1 için ön eğitimli model uygulanabilmeli ancak gerçekte performans düşüşü ve iyi genellemeyen modeller söz konusu olduğu için, modelin eğitim verilerine ve alana yönelik önyargısı olarak özgürce ve toplu olarak adlandırabilecek çeşitli nedenlerle olur.

Transfer öğrenimi, daha önce öğrenilen görevlerden elde edilen bilgilerin kullanılmasını ve bunları daha yeni, ilgili görevlerde uygulanmasını sağlamalıdır. T1 görevi için önemli ölçüde daha fazla veriye sahip olunduğunda, öğrenmesi kullanılabilir ve bu bilgi (özellikler, ağırlıklar) T2 görevi (önemli ölçüde daha az veriye sahip) için genellenebilir.

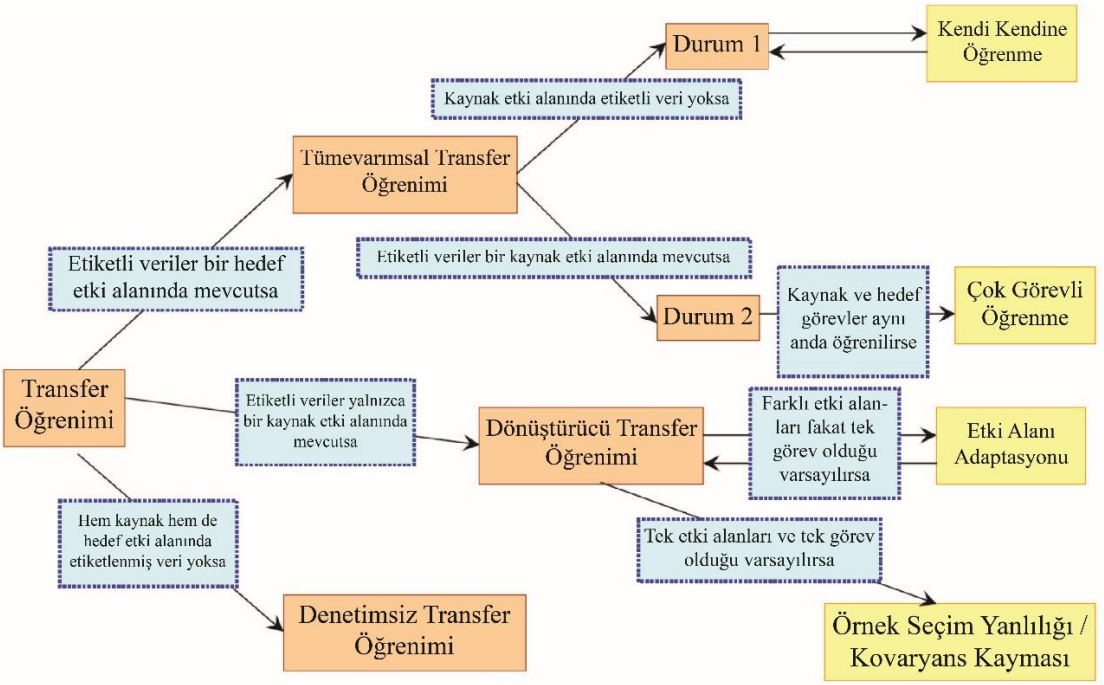
Transfer öğrenimi sürecinde aşağıda verilen 3 soru sorulur:

1. **Ne aktarılacak? :** Transfer öğreniminin ilk ve en önemlisi olan bu aşamada, kaynak veriden hedefe hangi bilgi aktarılınca hedef görevin başarısının artacağına cevap aranır. Bazı bilgiler, kullanılan veriye özgü olabilir ve hedef görev için performansı artırmaya yardımcı olabilecek şekilde farklı alanlar arasında ortak olabilir.
2. **Nasıl aktarılacak? :** Burada bilginin aktarımı için hangi yöntemlerin kullanılacağına karar verilir. Var olan yöntemlerin direkt kullanılması veya değiştirilmesi yaklaşımı izlenir.
3. **Ne zaman aktarılacak? :** Bu aşamada verinin hangi durumlarda aktarılacağı ya da hangi durumlarda aktarılmaması gerektiği durumları belirlenir. Bazı durumlarda kaynak veri ve hedef veri birbirinden çok farklı olabilir. Bu gibi durumlarda "olumsuz aktarım" adı verilen bir etmenden dolayı, kaynak veri hedef veri üzerindeki eğitimin başarısını düşürebilir ve aktarım başarısız olabilir. Önemli bir konu olmasına rağmen literatürde çok az çalışma mevcuttur.

Transfer öğrenmesinde kaynak veri amaca hizmet edecek bilgileri öğrenecek şekilde tasarlanarak veya önceden eğitilmiş bir model kullanılarak iki farklı şekilde tahsis edilebilir. Ön eğitimli modellerdeki ağırlıklar birçok bilgiyi barındırması sebebiyle, bu bilgiler üzerinden ince ayar yapılarak yeni model daha hızlı bir eğitilebilir. Sıfırdan oluşturulan modellerin eğitimi için büyük ölçekli bir veriseti gerekmektedir ve bu verisetlerinin oluşturulması ciddi bir zaman maliyetine sebep olur. Bunun yerine ön eğitimli modellere ince ayar yapılması daha az veri kullanarak yüksek performanslar elde edilmesini sağlar. Bu yüzden ön eğitimli model yaklaşımı derin öğrenme uygulamalarında diğerine göre daha yaygındır (Brownlee, 2019).

Bu tarz uygulamaların en güzel örneklerinden biri ImageNet yarışmasıdır [4]. Bu yarışmada 1000 sınıflı fotoğrafları sınıflandırma gibi büyük ve zorlu bir görüntü sınıflandırma görevi vardır. Bu yarışma için modeller geliştiren araştırma kuruluşları genellikle son modellerinin bir lisans kapsamında yeniden kullanılmasına izin vermektedir. Bu modellerin modern donanımlar ile eğitilmesinin günler veya haftalar alabildiği düşünüldüğünde önceki yıl eğitilmiş en iyi modelin bir sonraki yılda transfer öğrenimi ile kullanılması oldukça mantıklı ve yaygın bir uygulamadır. Bilgisayarla görme alanında Oxford VGG Model, Google Inception Model, Microsoft ResNet Model, Caffe Model Zoo; doğal dil işleme alanında Google Word2Vec Model, Stanford GloVe Model, Facebook fastText gibi herkese açık eğitilmiş modeller mevcuttur [5].

Alana, eldeki göreve ve verilerin kullanılabilirliğine bağlı olarak uygulanabilecek farklı transfer öğrenimi stratejileri ve teknikleri vardır. Bu nedenle Şekil 2.3'ye dayalı olarak, transfer öğrenimi yöntemleri ilgili geleneksel makine öğrenmesi algoritmalarının türüne göre kategorize edilebilir:



Şekil 2.3 Transfer Öğrenimi Stratejileri (Pan ve Yang, 2010)

- **Tümevarımsal transfer öğrenimi:** Kaynak ve hedef etki alanları aynıdır, ancak kaynak ve hedef görevler birbirinden farklıdır. Algoritmalar, hedef görevi geliştirmeye yardımcı olmak için kaynak alanın tümevarımsal önyargılarını kullanmaya çalışır. Kaynak etki alanında çok sayıda etiketli veri mevcutsa, tümevarımsal transfer öğrenimi çok görevli öğrenme ortamına benzer. Bu da hedef ve kaynak görevi aynı anda öğrenmeye çalışması anlamına gelir. Kaynak etki alanında etiketlenmiş veri olmaması durumunda ise tümevarımsal transfer öğrenme, kendi kendine öğretilen öğrenme ortamına benzer. Kendi kendine öğrenme ortamında, kaynak ve hedef alanlar arasındaki etiket boşlukları farklı olabilir, bu da kaynak alanın yan bilgilerinin doğrudan kullanılamayacağı anlamına gelir.
- **Denetimsiz transfer öğrenimi:** Kaynak ve hedef etki alanları benzer, ancak görevler farklıdır. Bununla birlikte, denetimsiz transfer öğrenimi, kümeleme, boyutsallık azaltma ve yoğunluk tahmini gibi hedef alandaki denetimsiz öğrenme görevlerini çözmeye odaklanır.
- **Dönüştürücü transfer öğrenimi:** Kaynak ve hedef görevler arasında benzerlikler vardır, ancak etki alanları farklıdır. Bu ayarda, kaynak etki alanında çok sayıda etiketlenmiş veri bulunurken hedef etki alanında hiç veri yoktur. Bu, özellik uzaylarının farklı olduğu veya alanlar arasındaki özellik uzayları aynı olup marjinal olasılıkların farklı olduğu ayarlara atıfta bulunarak alt kategorilere ayrılabilir.

Transfer öğrenimi sürecinde bahsedilen üç transfer kategorisi, transfer öğreniminin uygulanabileceği ve ayrıntılı olarak çalışılabileceği farklı ayarları özetlemektedir. Bu kategoriler arasında nelerin aktarılacağı sorusunu yanıtlamak için aşağıdaki yaklaşımlardan bazıları uygulanabilir:

- **Örnek aktarımı:** Kaynak verideki bazı örneklerin hedef verisinde kullanılmasıdır. Çoğu durumda kaynak veri doğrudan kullanılamaz. Fakat kaynak verisindeki belirli örneklere uygun ağırlık değerleri verilmesi ve hedef veri üzerinde kullanılmasıyla başarı artırılabilir. Kaynak ve hedef verisinin ortak birçok özelliğe sahip olması durumunda geçerlidir.
- **Özellik temsili aktarımı:** Kaynak verideki özellik temsillerinin hedef verisinde kullanılmasıdır. Kaynak ve hedef verisinin ortak özelliklere sahip olması durumunda geçerlidir. Ortak özelliklerin yanı sıra yalnızca kaynak veya yalnızca hedefte olan özellikler de bulunabilir. Kaynak verisinden hedef verisine daha iyi özellik gösterimleri bularak hedef görevdeki performansın artması beklenir.
- **Parametre aktarımı:** Kaynak görevlerin ve hedef görevlerin bazı parametreleri veya modellerin hiperparametrelerinin önceki dağılımlarını paylaştığı varsayılarak, aktarılan bilgi paylaşılan parametrelere veya önceliklere kodlanır. Böylece, paylaşılan parametreleri veya öncelikleri keşfederek bilginin görevler arasında aktarımı sağlanır.
- **İlişkisel bilgi aktarımı:** Kaynak verideki bazı ilişkilerin hedefteki ilişkilere benzetilmesi durumudur. Kaynak veri ve hedef veri arasında bazı ilişkilerin olduğu varsayılır, veri kaynakları arasında ilişkisel bilginin eşleştirilmesi yapılır.

2.4 İlgili Çalışmalar

RTS oyunları ve yapay zeka üzerine yapılan çalışmalarda literatürde en çok üzerinde test yapılan oyun Starcraft oyunudur. (Vinyals vd., 2017) çalışmasında, StarCraft II oyununa dayalı bir pekiştirmeli öğrenme ortamı olan geliştirilmiştir. StarCraft II etki alanı için gözlem, eylem ve ödül özelliklerini açıklanmaktadır ve oyun motoruyla iletişim kurmak için açık kaynaklı Python tabanlı bir arayüz sağlanır.

(Andersen vd., 2018) konferans tutanağında, RTS oyunları için son teknoloji yapay zeka algoritmalarını test etmek için Deep RTS oyun ortamı tanıtılmaktadır. Deep RTS, Blizzard Entertainment'ın ünlü StarCraft II video oyunundan esinlenen özellikle yapay zeka araştırmaları için yapılmış yüksek performanslı bir RTS simülatörüdür. Hızlandırılmış öğrenmeyi destekler, yani mevcut RTS oyunlarına kıyasla 50.000 kat daha hızlı öğrenebilir. Deep RTS, kısmen gözlemlenebilir durum uzayları ve harita karmaşıklığı dâhil olmak üzere birkaç farklı RTS senaryosunda araştırma yapılmasını sağlayan esnek bir yapılandırmaya sahiptir. Deep RTS kullanılarak, bir DQN ajanının rastgele oynayan ajanları zamanın %70'inden fazlasında yendiği tespit edilmiştir.

Son yıllarda araştırmalarda, DRL algoritmalarını RTS oyunlarına uygulanmasıyla, StarCraft II' de profesyonel oyuncuları yenebilecek güçlü özerk ajanlar oluşturarak büyük başarı elde edilmiştir. Bunlarda en önemlisi, DeepMind tarafından popüler RTS oyunu StarCraft II (Vinyals vd., 2019) için DRL ile AlphaStar adlı usta düzeyinde bir yapay zekânın eğitilmesidir. AlphaStar, etkileyici bir strateji ve oyun kontrolü gösterir, birçok insan benzeri davranış sergiler ve profesyonel oyuncuları sürekli olarak yenebilmektedir. Önceden tasarlanmış çoğu botun tam oyunda insanlara karşı iyi performans göstermediği göz önüne alındığında (Ontañón vd., 2013), AlphaStar açıkça bu alanda önemli bir kilometre taşını temsil etmektedir. Bu başarı etkileyici olsa da, yüksek hesaplama maliyetleri ile birlikte gelir. Özellikle AlphaStar ve hatta diğer ekiplerin hesaplama maliyetlerini düşürmeye yönelik daha fazla girişimleri (Lei vd., 2020),

ajanları uzun bir süre boyunca eğitmek için hala binlerce CPU ve GPU/TPU gerektirir ve bu çoğu araştırmacının finansal gücünün ötesidedir.

(Huang vd., 2021) çalışmasında bu sorunu ele almak için iki ana katkıda bulunulmuştur: İlk ana katkı Gym-μRTS adında, RTS oyunları için çeşitli yapay zeka tekniklerini test etmek için popüler bir platform olan RTS oyunu μRTS (Ontañón, 2013) için bir pekiştirmeli öğrenme arayüzü geliştirilmiştir. Basit görsellerine rağmen μRTS, RTS oyunlarının temel zorluklarını yakalar. Gym-μRTS, PySC2 (SC2LE) (Vinyals vd., 2017) ile pek çok benzerliğe sahip olsa da, birçok önemli farklılık da vardır (örneğin, PySC2 insan benzeri bir hareket alanı kullanırken Gym-μRTS daha düşük seviyeli bir hareket alanı kullanır). Gym-μRTS aracılığıyla, yüksek performanslı bilgi işlem kümeleri gibi kapsamlı teknik kaynaklar olmadan DRL kullanarak tam oyun RTS araştırmaları yürütülebilmektedir. μRTS' de yapılan basitleştirmelere rağmen, DRL aracılığıyla 1v1 rekabetçi maçlar oynamak hala göz korkutucu bir iş olduğu için ikinci ana katkı olarak, tam oyun μRTS oynamak için DRL'yi ölçeklendirmek için bir dizi teknikler ve bunların deneysel önemini göstermek için birtakım çalışmalar yapılmıştır.

(Torabi vd., 2018) çalışmasında, BCO adı verilen yeni bir taklitle öğrenme algoritması geliştirilmiştir. İlk çıktığı zaman araştırma topluluğunda büyük ilgi gören LfD paradigmasının entegrasyonunda iki önemli yönünün büyük ölçüde gözden kaçırılmış olmasına dikkat çekilmektedir. (Schaal, 1997; Argall vd., 2009) Birincisi, klasik LfD ortamından farklı olarak, insanlar tipik olarak göstericilerin davranışlarını yönlendirmek için kullandıkları dâhili kontrol sinyalleri hakkında bilgiye sahip değildir. İkincisi ise insanlar, gösterim sağlandıktan sonra çevreleriyle etkileşime girmek için çok fazla zaman harcamak zorunda kalmadan taklit yapabilirler. BCO, bu tartışılan konuların her ikisini de eşzamanlı olarak ele alır, yani yalnızca durum yörüngeli gösterileri gözlemler üzerine hemen hemen makul taklit politikaları sağlar.

(Gemine vd., 2012) çalışmasında, RTS oyun türünün çeşitli yönleri çok farklı sorunlar oluşturduğundan, ajanlara oyunun tüm yönlerinde öğrenme yeteneği vermek için çeşitli öğrenme teknolojileri gerekli olacağını savunarak, bir ajanın bir yapı inşa etmek veya bir teknolojiyi araştırmak gibi üretimle ilgili kararları nasıl aldığına odaklanılır. Taklitle öğrenme kullanarak bir dizi kayıtlı oyundan bir ajan üretim stratejilerini öğretmek için genel bir yöntem kullanılmıştır. Tam teşekküllü bir oyun düzenleyiciye sahip olan bu yeni tür öğrenme ajanlarını değerlendirmek için ideal bir platform olduğu için test ortamı olarak StarCraft II kullanılmıştır.

(Kelly ve Churchill, 2020) yazısında, SC2LE üzerindeki deneylerde eğitimli politikaların değişen karmaşıklıktaki RTS savaş senaryoları arasında etkin bir şekilde aktarabileceği kanıtlanmıştır. İlk olarak, çeşitli sayıda StarCraft II birimi üzerinde savaş politikaları eğitilmiştir ve ardından bu politikaları benzer kazanma oranlarını koruyarak daha büyük ölçekli savaşlarda uygulanmıştır. Daha sonra, savaş politikaları eğitme yeteneği oyundaki Terran Marine birim tipi üzerinde denenmiştir ve ardından bu politikaları benzer başarı ile Protoss Stalker adlı başka bir birim tipine uygulanmıştır.

(Sharma vd., 2007) çalışmasında, ticari bir RTS oyunu olan MadRTS kullanılarak, RTS oyunlarında transfer öğrenimi sağlamak için bir mekanizma geliştirilmiştir. Oyun ortamının stratejik ve taktiksel yönlerinin yakalanmasına, ayırıştırılmasına ve transfer öğreniminin modüler yapısından yararlanılmasına olanak tanıyan, CARL adlı mimari tanıtılmıştır. Böylece,

CBR'nin pekiştimeli öğrenme için örnek tabanlı durum işlevi tahmincisi ve pekiştimeli öğrenmenin CBR için zamansal fark tabanlı revizyon algoritması olarak kullanıldığı bir kombinasyon oluşturulmuştur. Yapılan deneyler sonucunda, CARL'ın yalnızca bireysel görevlerde iyi performans göstermediği, aynı zamanda önceki görevlerden bilgi aktarımına izin verildiğinde önemli performans kazanımları sergilediği görülmüştür.

3 DENEYSEL / TEORİK ÇALIŞMA

3.1 Lux AI Yarışması ve Kuralları

Lux AI Yarışması, rakiplerin diğer rakiplere karşı 1v1 senaryosunda çok değişkenli bir optimizasyon, kaynak toplama ve tahsis problemini çözmek için ajanlar tasarladığı bir yarışmadır. Optimizasyona ek olarak, başarılı ajanlar rakiplerini analiz edebilmeli ve üstünlük sağlamak için uygun politikalar geliştirebilmelidir.

Yarışma ekipleri her gün yarışmaya en fazla beş ajan (bot) gönderebilir. Her gönderilen bot, benzer beceri derecesine sahip lider tablosundaki diğer botlara karşı oyunlar oynar. Zamanla beceri puanları galibiyetlerle artar veya mağlubiyetle azalır ve beraberliklerle eşitlenir.

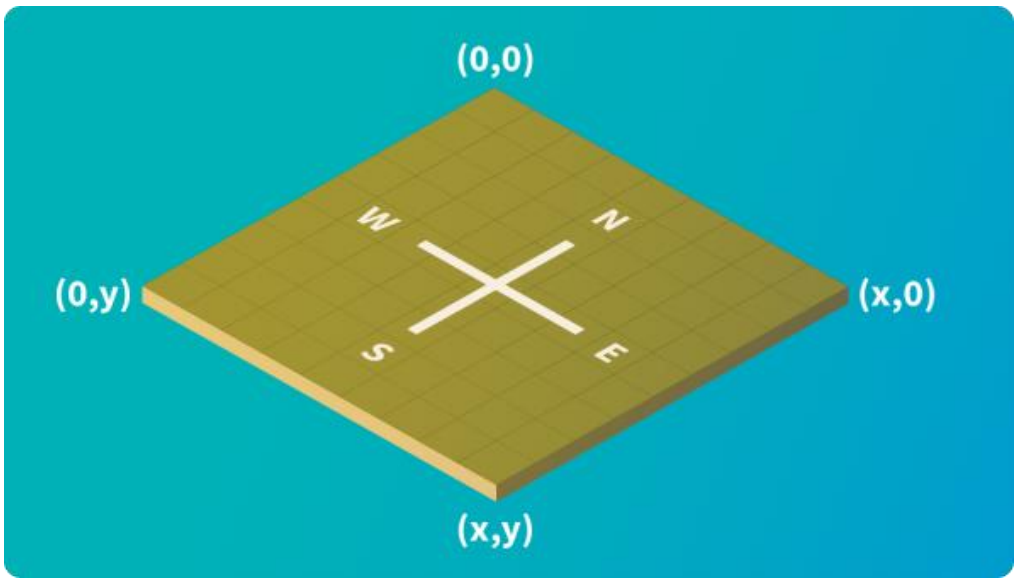
3.1.1 Yarışma ortamı

İki rakip takım, sıra tabanlı oyunun sonunda mümkün olduğunca çok şehir karosuna sahip olmak için ana hedefi olan şehirlerini beslemek için kaynak toplayan birimleri ve şehir karolarını sürekli kontrol etmelidir. Her iki takım da tüm oyun durumu hakkında eksiksiz bilgiye sahiptir ve kaynak toplamayı optimize etmek, kıt kaynaklar için rakibe karşı rekabet etmek ve puan kazanmak için şehirler inşa etmek için bu bilgileri kullanmalıdır.

Her turda, takım ajanının aksiyon alması üç saniye sürer, turlar arasında fazla zaman kaydedilmez. Her oyunda, bir turun üç saniyelik sınırı her aşıldığında açılan 60 saniyelik bir havuz verilir. 60 saniyenin tamamını kullanıp üç saniye sınırı aşıldığında, ajan donar ve artık ek eylemler gönderemez.

3.1.2 Harita

Lux dünyası Şekil 3.2'deki gibi, iki boyutlu bir ızgara olarak temsil edilir. Koordinatlar Doğu (sağ) ve Güney (aşağı) tarafına doğru artar. Harita her zaman bir karedir ve 12, 16, 24 veya 32 karo uzunluğunda olabilir. (0, 0) koordinatı sol üsttedir.



Şekil 3.1 Lux AI harita ızgarası

Harita, kaynaklar (odun, kömür, uranyum), birimler (işçiler, el arabaları), şehir karoları ve yol dâhil olmak üzere çeşitli özelliklere sahiptir. Haritaların bir oyuncuyu diğerine tercih etmesini önlemek için, haritaların her zaman dikey veya yatay yansıma ile simetrik olması garanti edilir. Her oyuncu, o şehir karosunda tek bir şehir karosu ve tek bir işçi ile başlamalıdır.

3.1.3 Kaynaklar

Yakıt verimliliğini arttırma sırasına göre odun, kömür ve uranyum olarak üç çeşit kaynak vardır. Bu kaynaklar işçiler tarafından toplanır, daha sonra bir işçi şehir karosunun üzerine çıktığında şehir için yakıta dönüştürülmek üzere bırakılır. Bazı kaynaklar, toplanmaları mümkün olmadan önce araştırma puanları gerektirir. Çizelge 3.1'de bu puanlar ile ilgili bilgiler mevcuttur.

Özellikle ahşap yeniden büyüyebilir. Her tur, her ahşap karonun ahşap miktarı, yuvarlanan mevcut ahşap miktarının %2,5'i kadar artar. Tükenmiş ahşap karolar yeniden büyümeyecektir. Yalnızca 500'den az ahşap içeren ahşap karolar yeniden büyüyecektir.

Çizelge 3.1 Lux AI Kaynak Tablosu

Kaynak Tipi	**Araştırma Puanları Ön Koşulu**	**Birim Başına Yakıt Değeri**	**Tur Başına Toplanan Birimler**
Odun	0	1	20
Kömür	50	10	5
Uranyum	200	40	2

3.1.4 Toplama mekanikleri

Her turun sonunda işçiler, kaynak toplayabildikleri tüm bitişik (Kuzey, Doğu, Güney, Batı veya Merkez) kaynak karolarından otomatik olarak kaynakları alır. Uranyum, kömür ve ardından odun kaynaklarında yineleme yapılırken,

- Her birim, mevcut yinelenen kaynağın her bitişik kutucuğundan çift sayıda kaynak toplamak için kaynak toplama talepleri yapar, öyle ki toplanan miktar birimin yükünü kapasitenin üzerine çıkarır. Örneğin, üç ahşap karoya bitişik 60 odunla çalışan işçi her birinden 14 tane ister, 40 odun alır ve iki tanesini boşa harcar.
- Geçerli yinelenen kaynağın tüm karoları daha sonra istekleri yerine getirmeye çalışır, yerine getirilmeyen tüm isteklerin eşit miktarda olmasını sağlayamazlarsa, kalan boşa harcanır. Örneğin, dört işçi 25 odunluk bir kiremit kazıyorsa, ancak bunlardan biri sadece beş, diğerleri her biri 20 odun istiyorsa, sonra ilk önce tüm işçiler her biri için beş odun alır, kalan beş odunu üç işçiye daha bırakır. Bu, son üç işçiye her birine bir odun vererek ve daha sonra boşa harcanan iki odun bırakarak eşit olarak dağıtılabilir.

İşçiler şehir karolarındayken madencilik yapamazlar. Bunun yerine, bir şehir karosu üzerinde en az bir işçi varsa, o şehir karosu otomatik olarak bitişik kaynakları bir işçiyle aynı oranda toplayacak ve hepsini doğrudan yakıta dönüştürecektir. Bir şehir karosu için kullanılan toplama mekaniği bir işçi ile aynıdır.

3.1.5 Eylemler

Birimler ve şehir karoları, belirli koşullar altında her turda eylemler gerçekleştirebilir. Genel olarak, tüm eylemler aynı anda uygulanır ve bir turun başlangıcında oyunun durumuna göre doğrulanır.

3.1.6 Şehir karoları

Şehir karoları, bir karoluk alan kaplayan bir yapıdır. Bitişik şehir karoları topluca bir şehir oluşturur. Her şehir karosunda, bekleme süresi birden küçük olması koşuluyla tek bir eylem gerçekleştirilebilir. Bu eylemler:

- **İşçi oluşturma:** Mevcuttaki şehir karosunun üzerinde işçi birimi oluşturulur. Eğer sahip olunan işçi ve el arabası sayısı, sahip olunan şehir karoları sayısına eşitse bir işçi inşa edilemez.
- **El arabası oluşturma:** Mevcuttaki şehir karosunun üzerinde el arabası birimi inşa edilir. Eğer mevcut sahip olunan işçi ve el arabası sayısı, sahip olunan şehir karoları sayısına eşitse bir el arabası inşa edilemez.
- **Araştırma:** Ekibin araştırma puanı bir artırılır

3.1.7 Birimler

İşçiler ve el arabaları olmak üzere iki birim türü vardır. Her birim, bekleme süresi birden küçük olduğunda tek bir eylem gerçekleştirebilir.

Tüm birimler hareket eylemini seçebilir ve Kuzey, Doğu, Güney, Batı ve Merkez olmak üzere beş yönden herhangi birinde hareket edebilir. Ayrıca tüm birimler, otomatik madencilik veya kaynak aktarımından elde edilen ham kaynakları taşıyabilir. İşçiler 100 birim kaynakla ve el arabaları 2000 birim kaynakla sınırlandırılmıştır.

Bir birim, ait olduğu takımın bir şehir karosunun üzerine çıktığında, şehir karosunun oluşturduğu şehir, taşınan tüm kaynakları yakıta dönüştürür. Şehir karosu olmadan karolar üzerinde en fazla bir birim olabilir. Ayrıca, birimler rakip takımın şehir karosunun üzerine çıkamaz. Bununla birlikte, birimler ait olduğu takımın şehir karosu üzerinde üst üste istiflenebilir. İki birim şehir karosu olmayan aynı karoya hareket etmeye çalışırsa, bu bir çarpışma olarak kabul edilir ve hareket eylemi iptal edilir.

3.1.8 İşçiler

İşçi birimlerin oyun içinde gerçekleştirebilecekleri eylemler şunlardır:

- **Hareket Etme:** Kuzey, Doğu, Güney, Batı ve Merkez olmak üzere beş yönden birine hareket edebilir.
- **Yağmalama:** Birimin üzerinde bulunduğu karonun yol seviyesini 0,5 oranda azaltır
- **Transfer:** Bir birimin kargosundan diğer komşu birime, sonrakinin kargo kapasitesine kadar, tek bir kaynak türünden herhangi bir miktarda gönderim yapabilir. Fazlalık orijinal birime iade edilir.
- **Şehir Karosu İnşa Etme:** İşçinin kargosunda her türden toplam 100 kaynağa sahip olması ve karosunun boş olması koşuluyla, bu işçinin hemen altında bir şehir karosu inşa edebilir. İnşa başarılı olursa, taşınan tüm kaynaklar tüketilir ve sıfır başlangıç kaynağı ile yeni bir şehir karosu inşa edilir.

3.1.9 El arabaları

El arabası birimlerinin oyun içinde gerçekleştirebilecekleri eylemler şunlardır:

- **Hareket Etme:** Kuzey, Doğu, Güney, Batı ve Merkez olmak üzere beş yönden birine hareket edebilir.
- **Transfer:** Bir birimin kargosundan diğer komşu birime, sonrakinin kargo kapasitesine kadar, tek bir kaynak türünden herhangi bir miktarda gönderim yapabilir. Fazlalık orijinal birime iade edilir.

3.1.10 Bekleme süresi

Çizelge 3.2'de görüldüğü gibi Şehir karoları, işçiler ve el arabalarının hepsinde her eylemden sonra bir bekleme süresi mekaniği bulunur. Birimler ve şehir karoları, yalnızca 1'den küçük bekleme süresine sahip olduklarında bir eylem gerçekleştirebilir.

Her turun sonunda, yol inşa edildikten ve yağmalandıktan sonra, her birimin bekleme süresi bir birim azalır ve dönüşün sonunda birimin üzerinde bulunduğu yolun seviyesi kadar daha da azalır. Şehir karoları yol seviyelerinden etkilenmez ve bekleme süresi her zaman bir birim azalır. Minimum bekleme süresi 0'dır. Bir eylem gerçekleştirildikten sonra, birimin bekleme süresi bir temel bekleme süresi kadar artar.

Çizelge 3.2 Birimlerin Temel Bekleme Süresi

Birim Tipi	Temel Bekleme Süresi
Şehir Karosu	10
İşçi	2
El Arabası	3

3.1.11 Yollar

El arabaları harita üzerinde gezinirken, tüm birimlerin daha hızlı hareket etmesini sağlayan yollar oluşturmaya başlarlar. Her turun sonunda el arabası, bittiği karonun yol seviyesini 0.75 oranda yükseltecektir. Yol seviyesi ne kadar yüksek olursa, birimler o kadar hızlı hareket edebilir ve eylemler gerçekleştirebilir. Tüm karolar yol seviyesi 0 ile başlar ve 6 ile sınırlandırılır. Ayrıca, şehir karolarının otomatik olarak maksimum yol seviyesi 6'dır. Yollar, her seferinde yol seviyesini 0,5 oranda azaltan yağma eylemiyle işçiler tarafından da yok edilebilir. Bir şehir karanlığa teslim olursa, şehrin tüm karolarının yol seviyesi 0'a düşer.

3.1.12 Gece/gündüz döngüsü

Gece/gündüz döngüsü, ilk 30'u gündüz, son 10'u gece olmak üzere 40 turluk bir döngüden oluşur. Bir maçta 9 döngü oluşturan toplam 360 tur vardır. Gece boyunca birimler ve şehirler hayatta kalabilmek için ışık üretmelidir. Gecenin her dönüşü, her birim ve şehir karosu bir miktar yakıt tüketecektir, yakıt miktarları Çizelge 3.3'te mevcuttur. Özellikle birimler, ışık üretmek için taşınan kaynaklarını kullanacak, şehir karoları ise yakıtlarını ışık üretmek için kullanacak.

İşçiler ve el arabaları, yalnızca şehir karosu üzerinde değillerse kaynakları tüketmeye ihtiyaç duyacaklar. Şehir dışındayken, İşçiler ve Arabalar gece ihtiyaçlarını karşılamak için tüm kaynak birimlerini tüketmelidir, örneğin bir işçi üzerinde 1 odun ve 5 uranyum taşıyorsa, 1

yakıt için tam odun tüketecek, ardından son 3 birim yakıt ihtiyacını karşılamak için tam bir uranyum tüketecek ve 37 birim yakıt israfı yapacaktır. Birimler her zaman önce en az verimli kaynakları tüketecektir. Son olarak, geceleri birimler iki kat daha fazla temel bekleme süresi kazanır.

Gece boyunca herhangi bir birimin yakıtı biterse, oyundan çıkarılır ve sonsuza dek gecenin içinde kaybolur. Ancak bir şehrin yakıtı biterse, sahip olduğu tüm şehir karolarıyla birlikte tüm şehir karanlığa düşer ve oyundan çıkarılır.

Çizelge 3.3 Birimlerin Yakması Gereken Yakıt Miktarları

Birim	**Şehirde Yakılan Yakıt**	**Şehir Dışında Yakıt Yakma**
Şehir karosu	23 - 5 * bitişik takıma ait şehir karosu sayısı	Yok
El arabası	0	10
İşçi	0	4

3.1.13 Oyun önerge sırası

Oyundaki eylemlerin tümü, geçerli olup olmadıklarını görmek için öncelikle mevcut oyun durumuna göre doğrulanır. Ardından, oyun etkinlikleriyle birlikte eylemler aşağıdaki sırayla ve her adımda aynı anda çözülür.

1) Artan bekleme süresiyle birlikte şehir karosu eylemleri
2) Artan bekleme süresi ile birlikte birim eylemleri
3) Yollar oluşturma
4) Kaynak toplama
5) Şehir karosunda kaynak düşmesi
6) Gece ise, birimlerin kaynakları ve şehir karolarının yakıt tüketmesini sağlama
7) Sıfıra kadar tükenmeyen odun karolarını yeniden büyütme
8) Her birim ve şehir karosu için bekleme süreleri işleme/hesaplama

Doğrulama kriterlerinin tek istisnası, birimlerin boşluklar arasında sorunsuz bir şekilde hareket edebilmesidir, yani iki birim bitişikse, bir sırayla yer değiştirebilirler.

Aksi takdirde, bir birimin şehir karosu inşa etmesi, ardından başka bir birimin yeni şehir karosunun üzerinde hareket etmesi gibi eylemlere izin verilmez, çünkü mevcut durumda bu yeni inşa edilmiş şehir yoktur ve birimler şehir karoları dışındaki diğer birimlerin üzerinde hareket edemez.

3.1.14 Kazanma koşulları

360 turdan sonra haritada en fazla şehir karosuna sahip olan takım oyunu kazanır. Her iki takımın şehir karosu sayısı eşitse, tahtada en fazla birime sahip olan takım kazanır. Birim sayısı da eşitse, oyun beraberlik olarak işaretlenir. Bir takımda artık birim veya şehir karosu yoksa oyun erken sona erebilir ve diğer takım kazanır.

3.2 Veri Toplama ve Uygulanacak Çözüm

Kaggle yarışmalarında yazılan notebook'lar genelde iki parçadan oluşmaktadır: Train ve Submission. Train kısmında, tercih edilen bir makine öğrenimi modelinin uygun verisetiyle eğitimi yapılır ve eğitilmiş ağırlık değerlerine sahip bir model ortaya çıkar. Submission

kısmında ise elde edilen ön eğitimli model kullanılarak yarışması beklenen bota aktarılıp yarışma ortamında test edilir. Kaggle'da eğitim süresi sınırlı olduğu için eğitim genellikle yerel makine üzerinde gerçekleştirilmesi daha pratiktir. Böylece sadece eğitilen modelin Kaggle ortamına entegre edilmesi sağlanır.

Veriseti olarak, yarışmacıların yarışma süresi içinde gönderdikleri botların diğer yarışmacı botlarla oynadıkları oyunların yeniden oynatma bilgilerini tutan JSON formatlı kayıt dosyaları kullanılmıştır. Kayıt dosyaları Kaggle'ın kendi sisteminde kayıtlı olan tüm yarışma, kullanıcı, veriseti, yazılan kodlardan elde edilen skorları içeren ve herkes tarafından kullanıma açık olan Meta Kaggle [6] verisetinden elde edilmiştir. Lux AI Yarışması'yla ilgili bilgileri daha sonra bir yerel makinede kullanmak için sistemden çekip buluta kaydetmeye yarayan bir notebook kullanıldı [7].

- **EpisodeId**: Yarışmaya gönderilen botun ID numarası
- **rewards:** Oyun sonunda iki takımın toplam ödül puanları [1.takım ödül puanı, 2.takım ödül puanı]
- **steps:** 360 tur boyunca yarışan iki takımın oyun içi bilgileri
    - **status:** aktiflik durumu
    - **action:** Oynatılan oyun sırasında ilgili birimin gerçekleştirdiği eylem [gözlem id, birim id, eylem etiketi]
    - **observation:** Oyunu oynayan iki takımından en yüksek ödül puanına sahip olanın gözlem bilgileri
        - **height:** Oyun haritası yüksekliği
        - **width:** Oyun haritası genişliği
        - **step:** Tur sayısı
        - **player:** Oyuncu indisi
        - **updates:** Birim eylemleri (ilerleyen kısımlarda bahsedilecektir)

Bu dosya içinde en önem teşkil eden değerler, yarışan iki takımın botunun sıra tabanlı yaptıkları gözlem ve eylem güncellemelerinden oluşan "steps" değerleridir. Bu değerler, davranışsal klonlamada taklit edilecek uzmanlardan toplanacak delillerdir. "steps" değerlerinden aktif durumda olan davranışlar seçilir ardından gözlem ve eylem çiftinden oluşan i.i.d örneklerine dönüştürülür. Lux AI içinde birimlerin gerçekleştirebildiği eylem sabitleri şu şekildedir:

- **m (hareket et):** [m, birim id, yön (Güney, Kuzey, Doğu, Batı, Merkez)]
- **bcity (şehir inşa et):** [bcity, birim id]
- **bw (işçi oluştur):** [bw, x koordinatı (şehir karosu), y koordinatı (şehir karosu)]
- **r (araştırma yap):** [r, x koordinatı (şehir karosu), y koordinatı (şehir karosu)]
- **t (aktar):** [t, kaynak kullanıcı id, hedef kullanıcı id, kaynak tipi (odun, kömür, uranyum), miktar]

Yapılan önişleme sonucunda 237170 tane gözlem, 1460691 tane eylem verisi elde edilmiştir. Kullanılan örnek davranışlar, yarışmayı birinci olarak tamamlayan Toad Brigade adlı takımın oyun kayıtlarından elde edilmiştir.

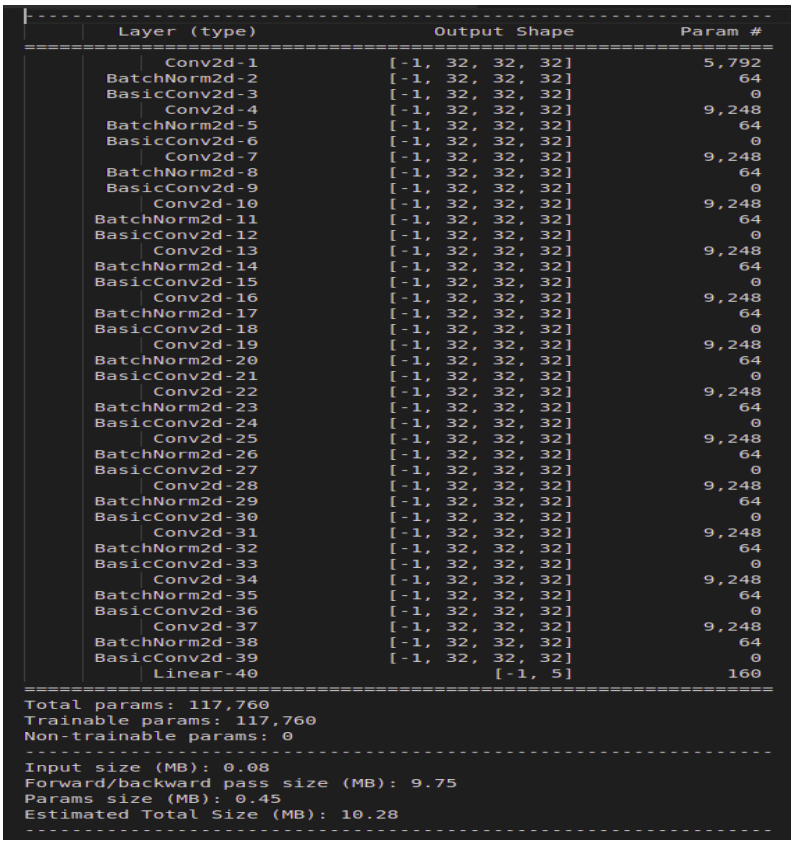
Davranışsal klonlama ile uzman davranışlarını eğitmek için bir CNN mimarisi kullanılmıştır, tezin geri kalanında bu ağdan "LuxNet" olarak bahsedilecektir. LuxNet, daha önce yine Kaggle üzerinden gerçekleştirilmiş Hungry Geese adlı yarışmada birinci olmuş HandyRL tarafından geliştirilen mimari ile benzerdir [8]. 20 özellik önce 32 kanala ve bir evrişim katmanına dönüştürülür ve ardından 3x3 filtreler ve toplu normalleştirme ile 12 kat evrişim katmanı haline gelir. Katmanlar boyunca, oyun haritasının boyutları sabit kalır ve lineer katman ile birim/şehir

koordinatındaki özellikler üzerinde tahmin yapılır. LuxNet'in tam hali Şekil 3.2'de mevcuttur. Toplu normalleştirme için kullanılacak yığın en fazla 64 parametre alacak şekilde ayarlandı. Meta Kaggle verisetinin Lux AI kayıtlarından ayıklanan gözlemler ve eylemler LuxNet'ten geçirilirken, eylemler %90 eğitim %10 doğrulama şeklinde kullanılır. Gözlemler ise eğitim ve doğrulama için bölünmeksizin her iki aşamada aynen kullanılır.

Dört yönün her biri için ve bir şehir inşa etmek için bir tane olmak üzere beş olası eylem vardır: Kuzey, Güney, Doğu, Batı ve bcity. Kurulan bu sinir ağına gönderilen ilgili birimin girdi değerleri sonucunda çıktı katmanında gerçekleştirileceği beş eylem için lineer dönüşüm uygulanarak izlenecek politika oluşturulur. Politika içinde elde edilen değerler arasında, en büyük olan değer uzman kişinin hangi eylemde bulunduğunun tahminini yapmakta kullanılır.

Sinir ağını eğitmek için girdi katmanına gönderilecek 20 özellik ise şu şekildedir:

1) Kararı veren birim bilgisi
2) Birim kargo seviyesi
3) Öz birimin varlığı (karar verme birimi hariç)
4) Öz birimin bekleme süresi seviyesi
5) Kendinden birim kargo seviyesi
6) Rakip birimin varlığı
7) Rakip birim bekleme süresi seviyesi
8) Rakip birim kargo seviyesi
9) Öz şehrin varlığı
10) Öz şehir karosunun gece hayatta kalma süresi
11) Rakip şehir karosunun varlığı
12) Rakip şehir karosu gece hayatta kalma süresi
13) Kaynak odun seviyesi
14) Kaynak kömür seviyesi
15) Kaynak uranyum seviyesi
16) Kendi kendine araştırma noktası
17) Rakip araştırma noktası
18) Gündüz gece döngü numarası
19) Mevcut tur numarası
20) Haritada sınır dışı olup olmadığı



```
----------------------------------------------------------------
      Layer (type)               Output Shape         Param #
================================================================
           Conv2d-1          [-1, 32, 32, 32]           5,792
      BatchNorm2d-2          [-1, 32, 32, 32]              64
      BasicConv2d-3          [-1, 32, 32, 32]               0
           Conv2d-4          [-1, 32, 32, 32]           9,248
      BatchNorm2d-5          [-1, 32, 32, 32]              64
      BasicConv2d-6          [-1, 32, 32, 32]               0
           Conv2d-7          [-1, 32, 32, 32]           9,248
      BatchNorm2d-8          [-1, 32, 32, 32]              64
      BasicConv2d-9          [-1, 32, 32, 32]               0
          Conv2d-10          [-1, 32, 32, 32]           9,248
     BatchNorm2d-11          [-1, 32, 32, 32]              64
     BasicConv2d-12          [-1, 32, 32, 32]               0
          Conv2d-13          [-1, 32, 32, 32]           9,248
     BatchNorm2d-14          [-1, 32, 32, 32]              64
     BasicConv2d-15          [-1, 32, 32, 32]               0
          Conv2d-16          [-1, 32, 32, 32]           9,248
     BatchNorm2d-17          [-1, 32, 32, 32]              64
     BasicConv2d-18          [-1, 32, 32, 32]               0
          Conv2d-19          [-1, 32, 32, 32]           9,248
     BatchNorm2d-20          [-1, 32, 32, 32]              64
     BasicConv2d-21          [-1, 32, 32, 32]               0
          Conv2d-22          [-1, 32, 32, 32]           9,248
     BatchNorm2d-23          [-1, 32, 32, 32]              64
     BasicConv2d-24          [-1, 32, 32, 32]               0
          Conv2d-25          [-1, 32, 32, 32]           9,248
     BatchNorm2d-26          [-1, 32, 32, 32]              64
     BasicConv2d-27          [-1, 32, 32, 32]               0
          Conv2d-28          [-1, 32, 32, 32]           9,248
     BatchNorm2d-29          [-1, 32, 32, 32]              64
     BasicConv2d-30          [-1, 32, 32, 32]               0
          Conv2d-31          [-1, 32, 32, 32]           9,248
     BatchNorm2d-32          [-1, 32, 32, 32]              64
     BasicConv2d-33          [-1, 32, 32, 32]               0
          Conv2d-34          [-1, 32, 32, 32]           9,248
     BatchNorm2d-35          [-1, 32, 32, 32]              64
     BasicConv2d-36          [-1, 32, 32, 32]               0
          Conv2d-37          [-1, 32, 32, 32]           9,248
     BatchNorm2d-38          [-1, 32, 32, 32]              64
     BasicConv2d-39          [-1, 32, 32, 32]               0
           Linear-40                 [-1, 5]             160
================================================================
Total params: 117,760
Trainable params: 117,760
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.08
Forward/backward pass size (MB): 9.75
Params size (MB): 0.45
Estimated Total Size (MB): 10.28
----------------------------------------------------------------
```

Şekil 3.2 LuxNet ağı özeti

Yukarıda belirtilen 20 özellik, daha önce ayrılmış gözlem verileri içindeki “width”, “height”, “updates” ve “step” anahtar kelimelerine bağlı değerler kullanılarak elde edilir. Örnek bir “updates” satırı Şekil 3.3’teki gibidir:

```
['rp 0 0','rp 1 1','r uranium 0 0 326','r wood 0 5 780','r wood 0 6 800','r
coal 0 10 386','r coal 0 11 366','r wood 1 5 780','r wood 4 2 381','r wood 4
3 349','r wood 5 1 407','r wood 5 2 353','r wood 5 3 335','r wood 5 10 387','r
wood 5 11 400','r wood 6 1 407','r wood 6 2 353','r wood 6 3 335','r wood 6
10 387','r wood 6 11 400','r wood 7 2 381','r wood 7 3 349','r wood 10 5
780','r uranium 11 0 326','r wood 11 5 780','r wood 11 6 800','r coal 11 10
386','r coal 11 11 366','u 0 0 u_1 1 5 1 40 0 0','u 0 1 u_2 10 5 1 40 0 0','c
0 c_1 0 23','c 1 c_2 0 23','ct 0 c_1 1 6 0','ct 1 c_2 10 6 9','ccd 1 6 6','ccd
10 6 6','D_DONE']
```

Şekil 3.3 Örnek bir 'updates' satırı

Bu satır içindeki tuple değer şeklindeki durum güncellemeleri, Lux AI içinde kullanılabilen altı farklı güncelleme değerine karşılık gelir ve kullanımları şu şekildedir:

- **rp (araştırma noktası):** [rp, oyuncu id, miktar]
- **r (kaynaklar):** [r, kaynak tipi, x koordinatı, y koordinatı, miktar]
- **u (birim):** [u, birim tipi (0 - işçi, 1 – el arabası), oyuncu id, birim id, x koordinatı, y koordinatı, bekleme süresi, odun, kömür, uranyum]
- **c (şehir):** [c, oyuncu id, şehir id, yakıt, aydınlıkta kalma]
- **ct (şehir karosu):** [ct, oyuncu id, şehir id, x koordinatı, y koordinatı, bekleme süresi]
- **ccd (yol):** [ccd, x koordinatı, y koordinatı, yol seviyesi]

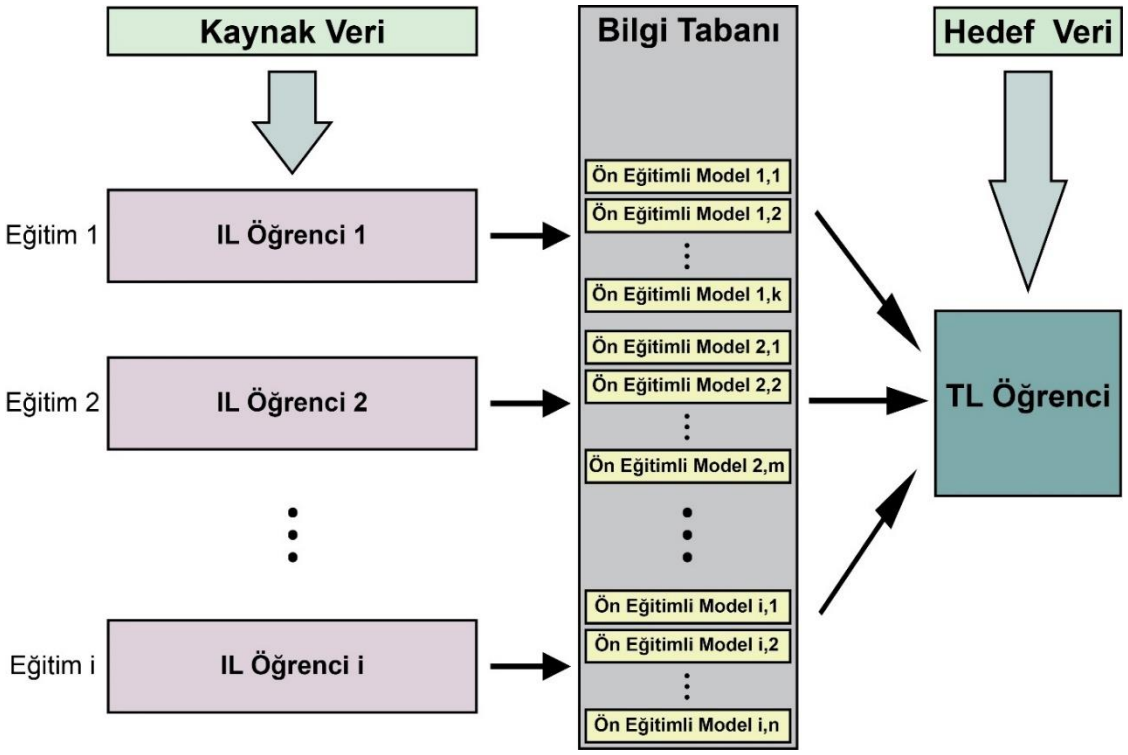
Gece gündüz döngüsü için gereken değerlerin eldesi için “step” değişkenine bağlı 1’den 360’a kadar olan tur sayıları önce 40’a bölümünden kalanı hesaplanarak 1’den 40’a kadar gece gündüz tur sayılarına çevirilir daha sonra da 40’a bölünerek 40 tur içindeki yüzdelik değerine dönüştürülür. Oyun tarafından tanından 360 tur ise sadece 360’a bölünerek yüzdelik değere dönüştürülür.

Optimizasyon algoritması olarak iki farklı algoritma kullanıldı. Algoritma seçimi sırasında, eğitim süresinden tasarruf etmek için daha küçük biri veriseti kurgulandı. Eğitimden elde edilen istatistiksel bilgiler sonucunda AdamW ve SGD optimizasyon algoritmalarından daha fazla eğtim kesinliği elde edildiği gözlemlendi. AdamW optimizasyonu Adam optimizasyonundan farklı olarak ağırlık azaltma kullanır. AdamW kullanılan bir ağ daha küçük ağırlıklara sahip olduğu zaman daha az aşırı öğrenme ve daha iyi genelleşme durumu gözlemlenmiştir (Loshchilov ve Hutter, 2019).

Daha küçük veriseti beraberinde doğrulama kaybını arttırdığı için büyük epok değerleri kullanıldığında bir noktadan sonra model aşırı öğrenme durumuna geçmeye başlar. Bu soruna çözüm olarak, tez kapsamında kullanılacak verisetine geçilip aynı iki farklı optimizasyon algoritması için ayrı eğitim yapmanın daha uygun olacağına karar verildi. Yapılan eğitim, elde edilen çıktılar kategorik değerlere sahip olduğu için bir çok sınıflı sınıflandırma uygulamasıdır. Bu yüzden eğitimdeki kayıp, çapraz entropi [9] ile hesaplanır.

Makine öğreniminde önemli olan bir diğer parametre ise öğrenme oranıdır. Bu eğitimdeki amaç olabildiğince farklı eğitilmiş model üretebilmek olduğu için bir öğrenme oranı düzenleyicisi kullanarak uyarlanabilir hale getirildi. Düzenleyiciler, eğitimde kullanılan mimari üzerine aşırı öğrenmeyi azaltmaya yönelik yapılan ince ayarlamayı kolaylaştırıp eğitim için belirlenen azaltma oranına göre en uygun öğrenme oranı değerini bulmaya çalışır. Düzenleyici olarak LambdaLR formülü kullanıldı [10]. Öğrenme oranının başlangıç değeri ve lambda katsayısı 0.1 olarak belirlendi. Birbirinden farklı eğitilmiş modelleri oluşturmak için istenen kontrol koşulu şöyledir: Bir epok anında elde edilen kesinlik değeri, daha önce kaydedilen en iyi kesinlik değerinden daha büyük olup olmadığına bakılır. Eğer büyükse en son hesaplanan epok kesinlik değerine kadar eğitilen model kaydedilir ve en iyi kesinlik değeri güncellenir. Bu sayede eğitilen modelde elde edilen olumlu sonuçlar içeren birden fazla model oluşturulmuş olur. Kurulan LuxNet, Intel(R) Core i7-8750H CPU 2.20 GHz, Nvidia GeForce GTX 1050 Mobile 4GB, 16 GB DDR4 RAM, 128 GB SSD donanıma sahip bir makine üzerinde eğitilmiştir.

Tez kapsamında uygulanan istenen çözüm Şekil 3.4’te de görülebilmektedir. Kaynak veri, uzman oyunculardan edinilen gözlem ve eylem bilgilerini temsil eder. Taklitle öğrenme yapan öğrenci makinelerden edinilen ön eğitimli modeller bilgi tabanını oluşturacaktır. Eğitim tamamlandığı zaman oluşan ön eğitimli modeller, Lux AI ortamında test edilmek üzere Kaggle’a yüklenecektir ve yarışmaya sokulacak RTS botunun uzmanlardan edinilen bilgilerin transfer öğrenimi ile aktarımı sağlanacaktır. Bot oyun sırasında, aktif olarak gözlem yapacaktır. Bu da transfer öğrenimi uygulanan RTS botunun hedef verisidir.



Şekil 3.4 Uzman verilerinden oluşan veriseti kullanılarak yapılan eğitimden elde edilen eğitilmiş modeller ile transfer öğrenimi gösterimi

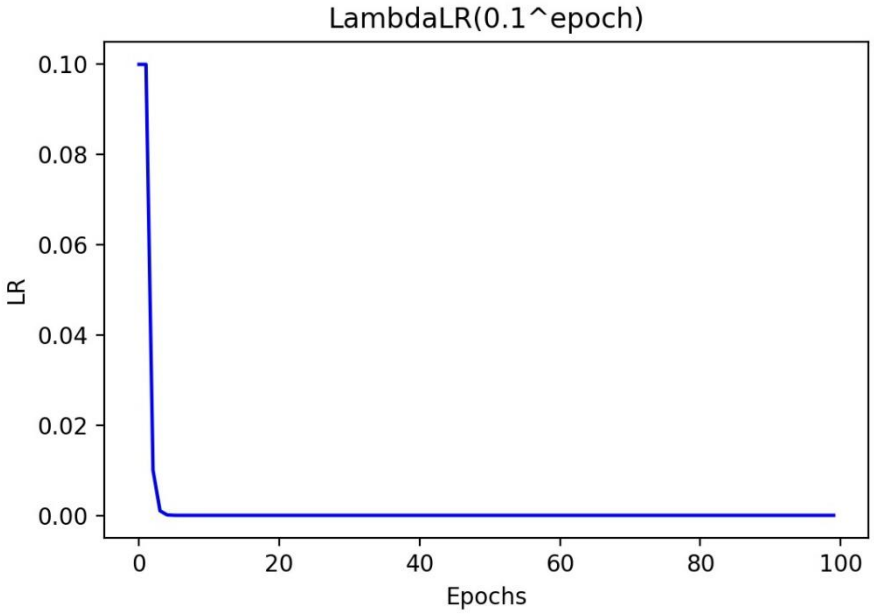
Yapılan bu gözlemler ile botun haritadaki işçilerinin LuxNet'i eğitmek için kullanılan 20 özelliğe göre durum bilgisi çıkartılır. Daha sonra eklenmiş ön eğitimli modellerin her biri için o duruma uygun politikalar oluşturulur ve oluşturulan politikalara uygun ilgili birimin bir sonraki muhtemel eylemleri ve bir sonraki pozisyonları belirlenir. Fakat en son hesaplanan pozisyon hedef pozisyon olarak seçilir, bu da rastgeleliliği artırır. İşçiler için belirlenen eylem şeması şu şekildedir:

- ('move', 'n')
- ('move', 's')
- ('move', 'w')
- ('move', 'e')
- ('build_city',)

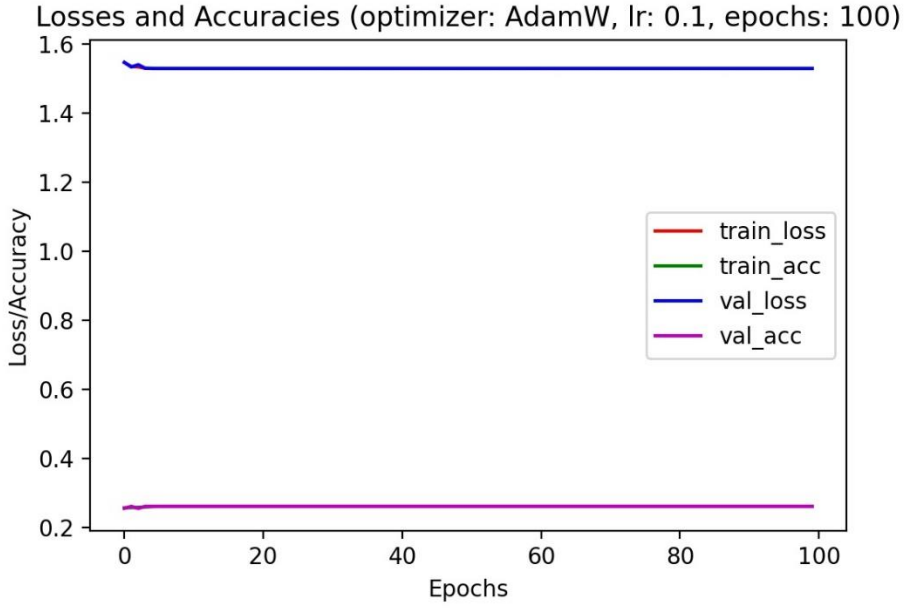
Gönderilen politikalara şemadaki eylemlerden en çok hangisi dönüş değeri olarak gelirse, o eylem seçilir. Bu işlem, 360 tur boyunca her 30 gündüz turu boyunca devam eder. Şehirler ise mümkün olan her zamanda birimler inşa eder. Aksi takdirde şehir, uranyum seviyesine ulaşılıp ulaşılmadığı araştırılır ve şehirlerin eylemleri taklit edilemez.

4 GENEL SONUÇLAR VE ÖNERİLER

Şekil 4.1‘deki eğitim sonuçlarına göre mevcut CNN modelinin AdamW 0.1 değerindeki öğrenme oranında yapılan eğitim sonucunda öğrenme oranının çok hızlı şekilde sıfıra yaklaşmıştır. Öğrenme oranındaki bu hızlı düşüş, eğitimin neredeyse başlamadan sabitlenmesine sebep olmuştur. Şekil 4.2’de de gözlemlenebilen bu eğitim sonucunda en yüksek eğitim kesinliği değeri %26.19 olarak bulunmuştur ve toplamda 2 tane ön eğitimli model üretilmiştir. Ortaya çıkan sonuç beklenenin altında kaldığı için transfer öğrenimi sonucunda alınmak istenen yüksek skora ulaşmak adına, AdamW ile yapılan eğitimde öğrenme oranı 0.01 olarak güncellenerek yeniden eğitilmiştir.

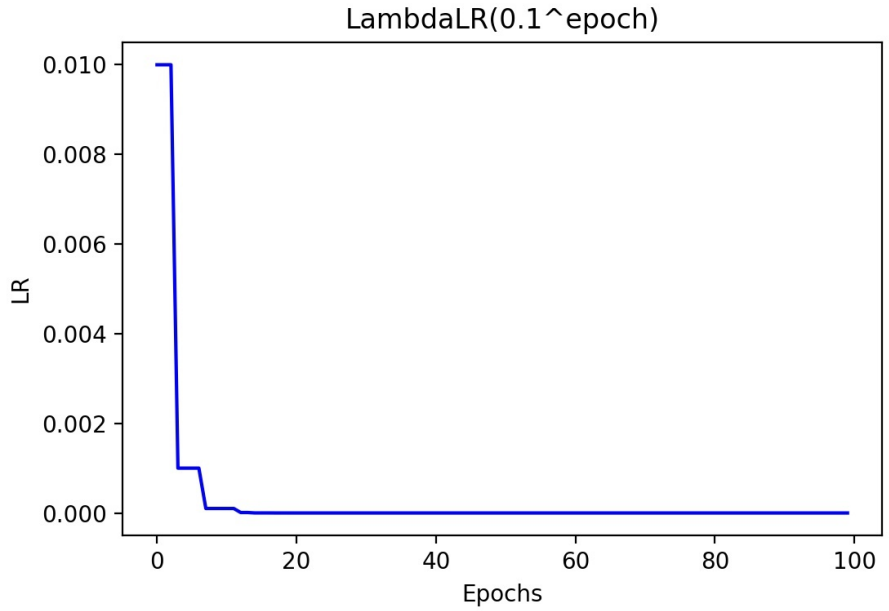


Şekil 4.1 LambdaLR algoritmasının değişim grafiği (AdamW, lr: 0.1, 100 epok)

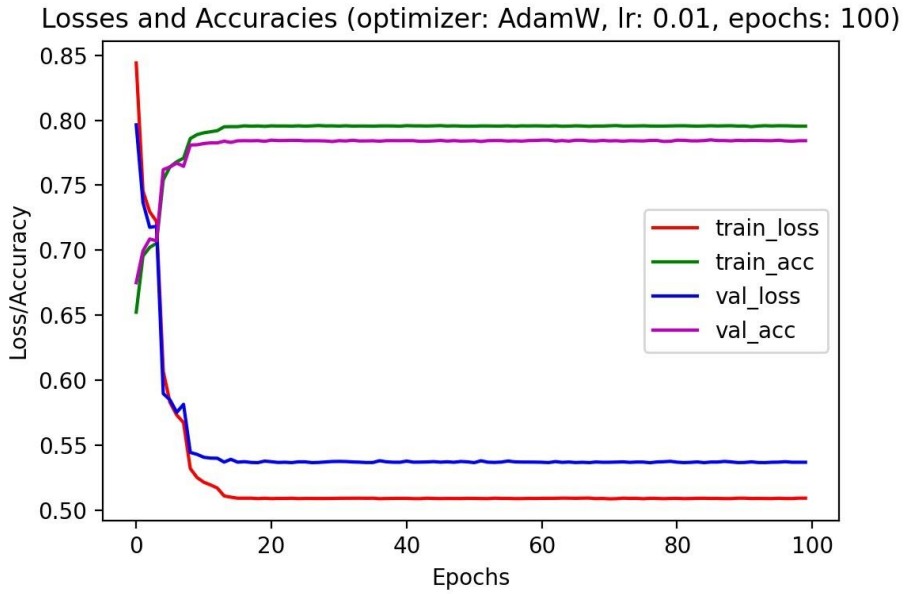


Şekil 4.2 Kayıp ve kesinlik değerleri grafiği (AdamW, lr: 0.1, 100 epok)

Şekil 4.3'teki grafik incelendiğinde öğrenme oranının önceki eğitime kıyasla, sıfıra daha geç yakınsamakla birlikte bazı epok değerleri arasında sabitlenme gözlemlenmektedir. Bu nokta Şekil 4.4'e bakıldığı zaman aynı epok değerleri arasında eğitim ve doğrulama kesinliği değerlerinin artmasına sebep olmaktadır. Bu da ön eğitimli model üretiminde çeşitliliği sağlayan bir etmendir. AdamW optimizasyonu kullanılarak yapılan ikinci eğitim, en yüksek eğitim kesinliği değeri %78.49 olarak sonuçlanıp bu değerden sonraki epoklarda ağın öğrenecek bir verisi kalmadığı için sabit bir seyir izlemiştir. Öğrenme oranı güncellenerek yapılan yeni eğitim sonucunda 18 tane ön eğitimli model üretilmiştir.

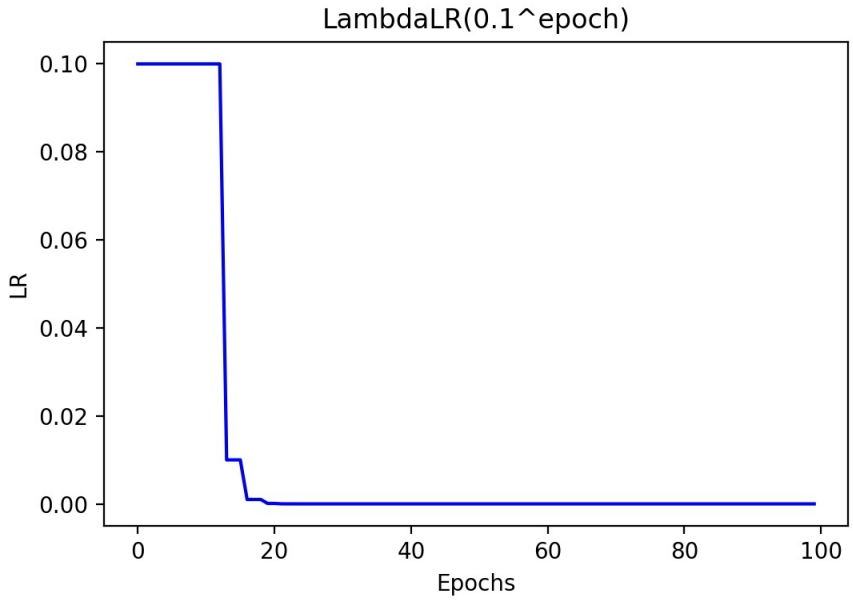


Şekil 4.3 LambdaLR algoritmasının değişim grafiği (AdamW, lr: 0.01, 100 epok)

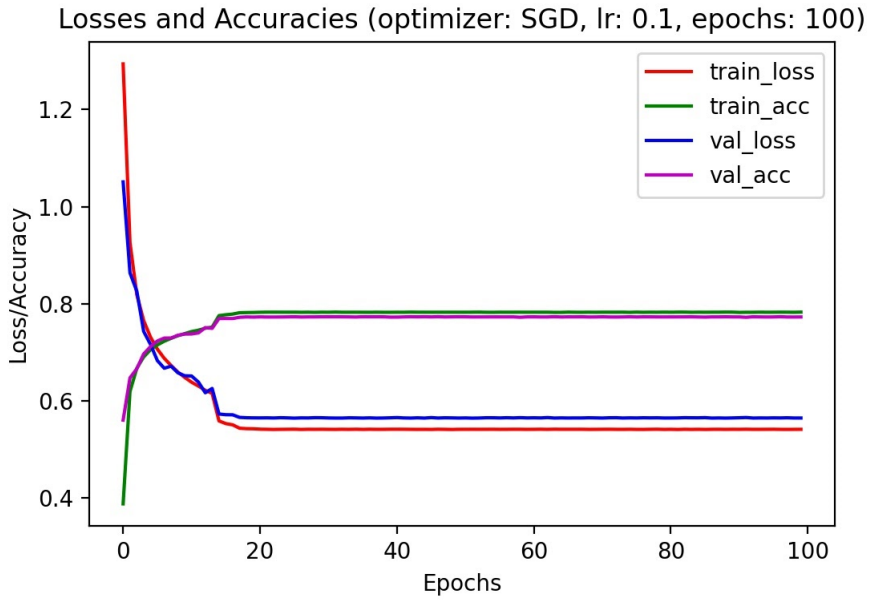


Şekil 4.4 Kayıp ve kesinlik değerleri grafiği (AdamW, lr: 0.01, 100 epok)

Diğer bir optimizasyon algoritması olan SGD kullanılarak yapılan eğitimde, önceki eğitimlere nazaran hiçbir güncelleme yapmaksızın daha sabit öğrenme oranına ve daha uzun epok süresine sahip bir öğrenme gerçekleşmiştir. Şekil 4.5'teki grafikte görüldüğü üzere, öğrenme oranı değişimi 20. epoktan sonra sıfıra yaklaşmıştır. Bu da davranışsal klonlamada kullanılan LuxNet'in gradiyan azalımlı optimizasyon algoritmalarıyla yapılan eğitimin, adaptif optimizasyon algoritmalarıyla yapılan eğitime göre daha yavaş ve uzun vadede daha etkili sonuç verdiğini göstermektedir. Şekil 4.6'daki grafikte de görüldüğü gibi eğitim kesinliği değeri %77.31 olarak sabitlenmiştir ve 23 tane ön eğitimli model üretilmiştir.



Şekil 4.5 LambdaLR algoritmasının değişim grafiği (SGD, lr: 0.1, 100 epok)



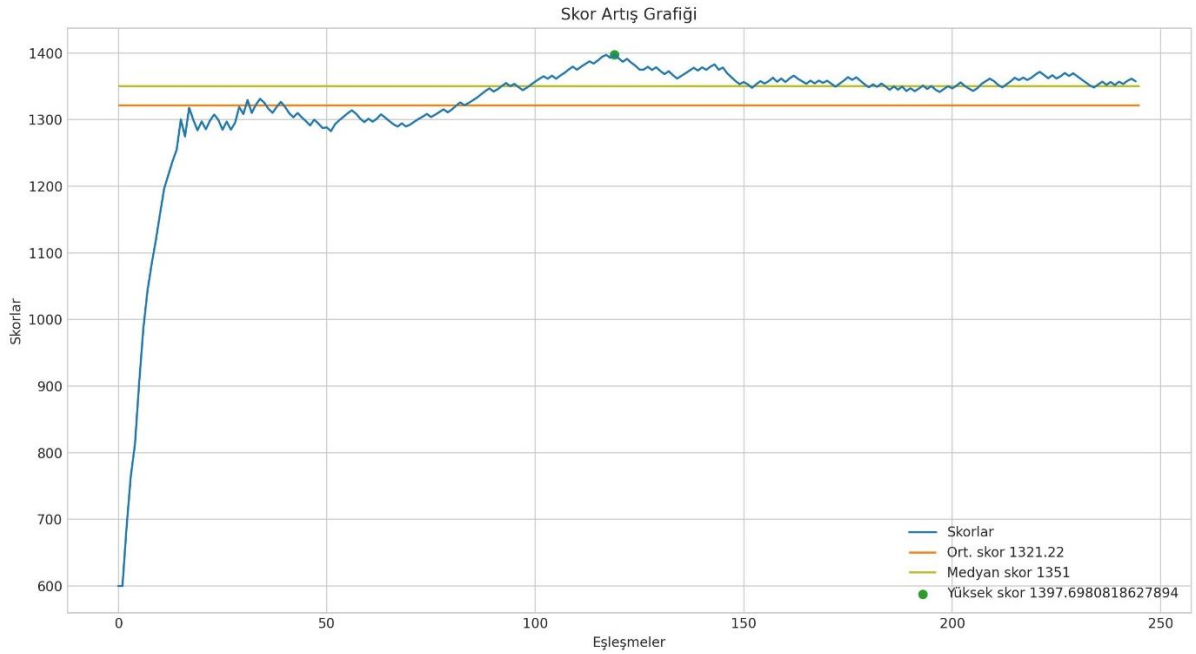
Şekil 4.6 Kayıp ve kesinlik değerleri grafiği (SGD, lr: 0.1, 100 epok)

Eğitim başı 2946 dakika süren üç farklı eğitim, toplamda 8838 dakika yani yaklaşık altı günde tamamlanmıştır. Ön eğtimli model üretimi için kullanılan kodlara için şu sayfadan erişilebilir

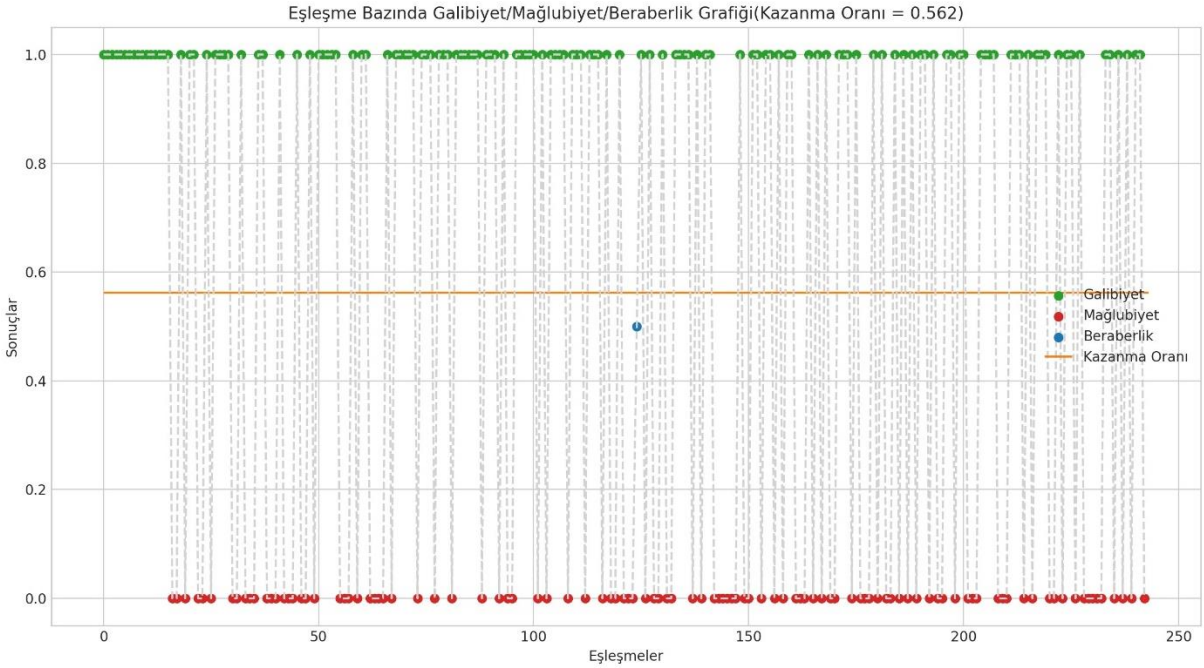
[11]. Bu üç eğitim sonucunda toplam 43 ön eğitimli model üretilmiştir. Bu modeller, yarışma ortamında test edilmek üzere Kaggle platformuna aktarılmıştır [12].

İlk olarak eğtimde elde edilen ön eğitimli modellerin tamamı kullanılmak istenmiştir. Fakat 43 modelin aynı anda kullanılması, botun doğrulama için ortam içinde çalıştırıldığında hata vermesine sebep olmuştur. Yapılan deneyden sonuç alabilmek için 21 farklı deneme yapılmıştır. Her denemede kullanılan ön eğitimli model sayısı azaltılmıştır ve son gönderim tarihi olan 6 Aralık 2021'e kadar denemeye devam edilmiştir. Yapılan denemeler sonucunda iki optimizasyondan en yüksek değerli üç tane olmak üzere altı modelli versiyon ve iki optimizasyondan en yüksek değerli birer tane olmak üzere iki modelli versiyon, doğrulama çalıştırması aşamasını geçebilmiştir. Bu da şunu göstermektedir, modelden aktarılacak veri çok fazla ise zaman maliyeti artar. Bu sorunu önlemek ve daha etkili bir çözüm için daha iyi sonuç alınan ön eğitimli modeller seçilerek zaman maliyeti düşürülmelidir. Tezin bu kısmından itibaren bu iki farklı versiyonun karşılaştırılması yapılacaktır. İki model kullanılan bottan Versiyon 6[13], altı model kullanılan bottan ise Versiyon 7[14] olarak bahsedilecektir.

Yarışmaya 7-20 Aralık 2021 tarihleri arasında son oyunları oynamak üzere gönderilmiş iki farklı bottan, Versiyon 6 toplam 243 tane rakiple eşleşmiştir. Şekil 4.8'deki grafikte görüldüğü üzere 136 tane galibiyet 1 tane beraberlik ve 106 tane mağlubiyet ile sonuçlanmıştır. Bu grafikteki verilerine dayanarak, geliştirilen RTS botunun %56.2 yani burun farkı denenebilecek bir kazanma oranına sahip olunduğu söylenebilir. Şekil 4.7'de gösterilen skor artış grafiğinin ilk %20'lik diliminin başlangıcında çok hızlı bir artış olduğu görülmektedir. İlk seferlerde hiç oyun kaybetmeyen RTS botu, sonraki oyunlarda karşısına daha zorlu rakipler çıktıkça skor artışı durulup daha inişli çıkışlı bir seyir izlemiştir. İkinci %20'lik dilimde tekrar artışa geçen skor artışı, üçüncü %20'lik dilimde en yüksek değerine ulaştıktan sonra medyan skoru civarında değerler alarak skor değişim küçültmüştür. Versiyon 6'nın skor artışı grafiğinden elde edilen istatistik sonuçlarına göre 243 eşleştirme sonucu elde edilen skorların ortalaması 1321.22, medyan skor 1351 ve en yüksek skor 1397.69 puandır.



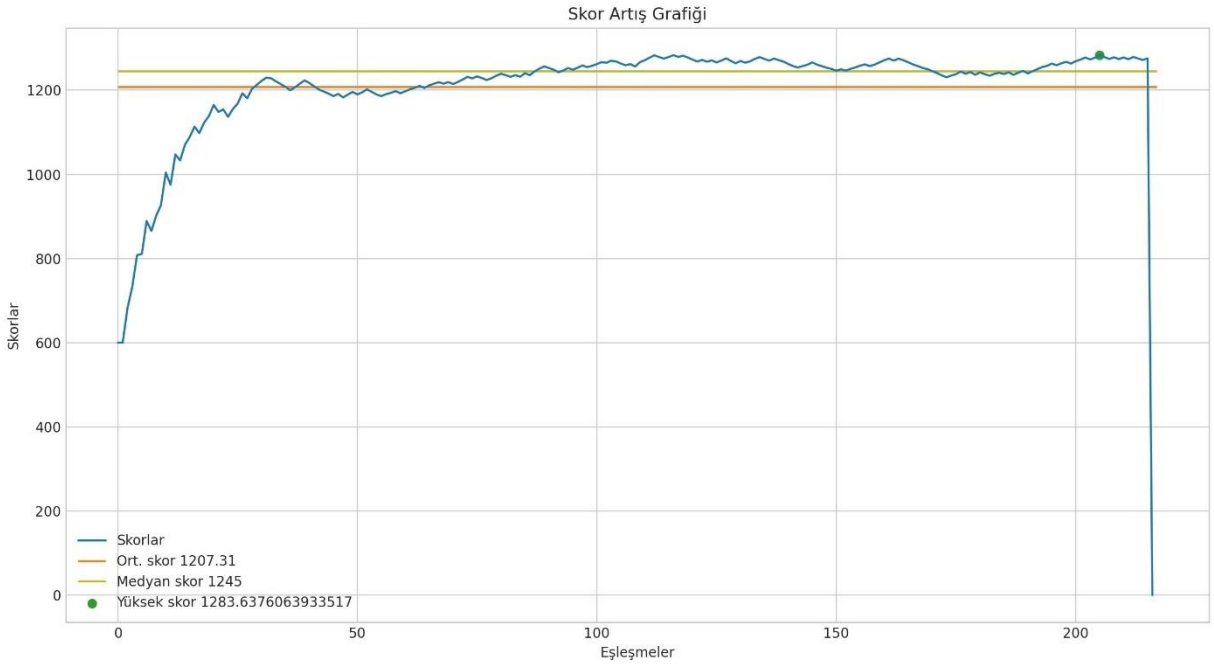
Şekil 4.7 Skor artış grafiği (Versiyon 6)



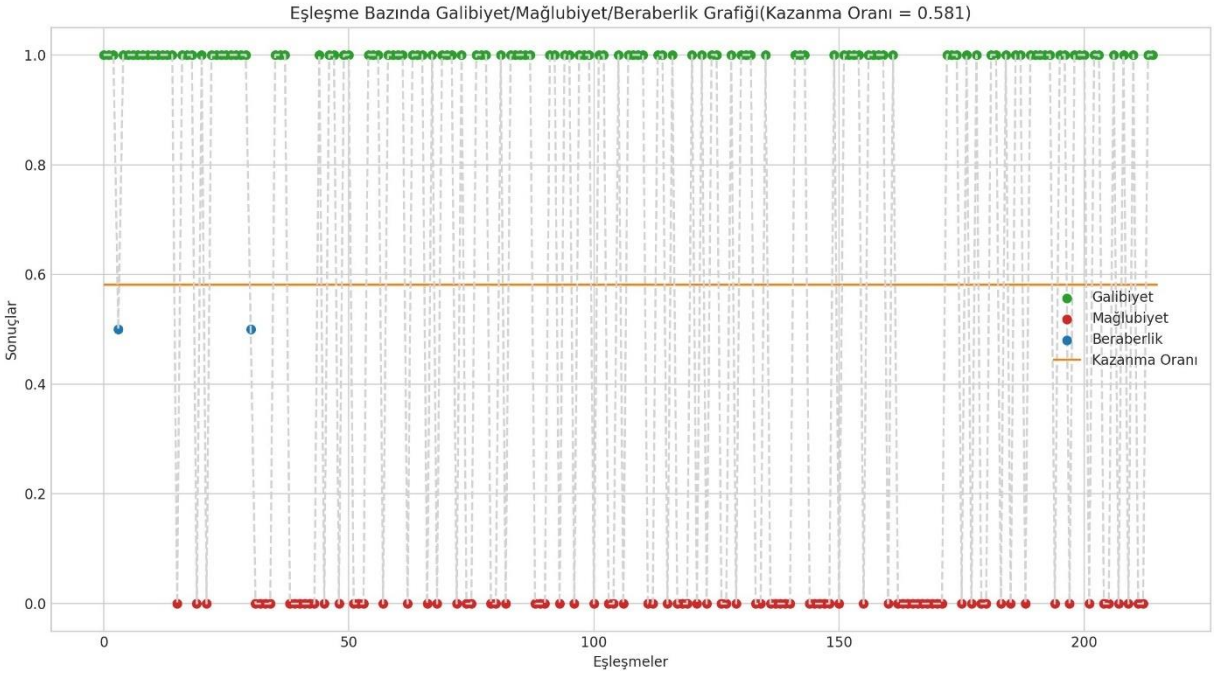
Şekil 4.8 Eşleşme bazında Galibiyet/Mağlubiyet/Beraberlik grafiği (Versiyon 6)

Versiyon 7'ye bakıldığı zaman ise toplam 215 tane rakiple eşleşmiştir. Şekil 4.10'daki grafikte görüldüğü üzere 124 tane galibiyet 2 tane beraberlik ve 89 mağlubiyet ile sonuçlanmıştır. Bu grafikteki verilere ise, geliştirilen RTS botunun %58.1'lik bir kazanma oranına sahip olmuştur. Versiyon 6 ile kıyaslandığında daha az bir eşleşme sayısına sahip olmasına rağmen daha yüksek bir kazanma oranına sahiptir. Bununla birlikte Versiyon 6'daki daha düz ilerleyen skor artışı, Şekil 4.9'daki grafiğe bakıldığı zaman Versiyon 7'te hemen hemen benzer şekilde ilerlemesine rağmen son %20'lik dilimden birdenbire düşüşe geçmiştir.

Versiyon 7'in skor artışı grafiğinden elde edilen istatistik sonuçlarına göre 215 eşleştirme sonucu elde edilen skorların ortalaması 1207.31, medyan skor 1245 ve en yüksek skor 1283.64 puandır. Bu sonuca göre Versiyon 7 yüksek kazanma oranına sahip olmasına rağmen skor olarak Versiyon 6'ya yaklaşamamıştır.



Şekil 4.9 Skor artış grafiği (Versiyon 7)



Şekil 4.10 Eşleşme bazında Galibiyet/Mağlubiyet/Beraberlik grafiği (Versiyon 7)

Bu çalışmada oynadığı oyunlarda galip gelen uzman oyunculardan toplanan gözlem verilerinin taklit edilerek edilen bilgileri rakibine karşı kullanacak bir RTS botu için kullanılan yöntemler tanıtıldı. Kullanılan uzman oyuncu verileri Kaggle'ın herkes açık Meta Kaggle adlı verisetinden Lux AI için yapılan paylaşımlar sonucu elde edilen yeniden oynatma verilerinden elde edilmiştir. Davranışsal klonlama kullanılarak eğitilen CNN mimarisinden sonucu elde edilen öneğitimli modellerden en yüksek kesinlik değerine sahip altı modelden elde edilen bilgiler, transfer öğrenimi uygulanarak tez kapsamında hazırlanan RTS botuna aktrarılmıştır. Yapılan deneyler sonucunda elde edilen Versiyon 6 ve Versiyon 7 botları arasından Versiyon 6, Lux AI

Yarışması'nda gösterdiği performanstan ötürü, yarışmayı 1357.4 puanla 190. sırada tamamlamıştır. Bu çalışmadaki adımlardan farklı olarak, verilerin eğitilmesinde kullanılan CNN ağı üzerinde değişiklik yaparak nöronlar arası aktarılan bilgilerin kaybolması azaltılabilir ve böylece daha yüksek eğitim ve doğrulama kesinliği ve daha düşük eğitim ve doğrulama kaybı elde edilebilir. Son olarak, bu çalışma şunu göstermiştir ki aynı eğitim modelinin varyasyonları çoğaldığı zaman aynı eylemlere karar verme olasığı artmaktadır. Karar aşamasında eylem çeşitliliği arttırabilmek için farklı eğitim modellerine başvurulabilir.

5 KAYNAKLAR

Andersen, P., Goodwin, M., & Granmo, O. (2018). Deep RTS: A Game Environment for Deep Reinforcement Learning in Real-Time Strategy Games. *2018 IEEE Conference on Computational Intelligence and Games (CIG)* (s. 1-8). IEEE.

Argall, B., Chernova, S., Veloso, M., & Browning, B. (2009). A survey of robot learning from demonstration. *Robotics and Autonomous Systems Volume 57, No 5*, 469-483.

Brownlee, J. (2019, Eylül 16). *A Gentle Introduction to Transfer Learning for Deep Learning*. Machine Learning Mastery (Erişim Tarihi: 16 Ocak 2022): https://machinelearningmastery.com/transfer-learning-for-deep-learning/ adresinden alındı

Buro, M. (2003). Real-Time Strategy Games: A New AI Research Challenge. *IJCAI-03*, 1534–1535.

Certicky, M., & Churchill, D. (2017). The Current State of StarCraft AI Competitions and Bots. *AIIDE 2017 Workshop on Artificial Intelligence for Strategy Games.*

Churchill, D., Preuss, M., Richoux, F., Synnaeve, G., Uriarte, A., Ontañón, S., & Certicky, M. (2016). StarCraft Bots and Competitions. L. Newton içinde, *Encyclopedia of Computer Graphics and Games* (s. 1-18). Springer International Publishing.

Gemine, Q., Safadi, F., Fonteneau, R., & Ernst, D. (2012). Imitative learning for real-time strategy games. *2012 IEEE Conference on Computational Intelligence and Games (CIG)* (s. 424-429). IEEE.

Genesereth, M., & Björnsson, Y. (2013). The International General Game Playing Competition. *AI Magazine, 34*, 107-111.

Gomez, F., & Miikkulainen, R. (1997). Incremental Evolution of Complex General Behavior. *Adaptive Behavior Volume 5 Issue 3-4*, 317–342.

Huang, S., Ontañón, S., Bamford, C., & Grela, L. (2021). Gym-μRTS: Toward Affordable Full Game Real-time Strategy Games Research with Deep Reinforcement Learning. *2021 IEEE Conference on Games (CoG)* (s. 1-8). IEEE.

Hussein, A., Gaber, M., Elyan, E., & Jayne, C. (2017). Imitation Learning: A Survey of Learning Methods. *ACM Computing Surveys Volume 50 No 2*, 35.

Kelly, R., & Churchill, D. (2020). Transfer Learning Between RTS Combat Scenarios Using Component-Action Deep Reinforcement Learning. *AIIDE Workshops*.

Kober, J., & Peters, J. (2010). Imitation and Reinforcement Learning. *IEEE Robotics Automation Magazine Volume 17 No 2*, 55-62.

Kober, J., Bagnell, J., & Peters, J. (2013). Reinforcement Learning in Robotics: A Survey. *International Journal of Robotics Research Volume 32 No 11*, 1238–1274.

Lei, H., Jiechao, X., Peng, S., Xinghai, S., Meng, F., Qingwei, G., . . . Zhengyou, Z. (2020). TStarBot-X: An Open-Sourced and Comprehensive Study for Efficient League Training in StarCraft II Full Game. *ArXiv*.

Llargues Asensio, J. M., Donate, J. P., & Cortez, P. (2014). Evolving Artificial Neural Networks applied to generate virtual characters. *2014 IEEE Conference on Computational Intelligence and Games* (s. 1-5). Dortmund, Germany: IEEE.

Loshchilov, I., & Hutter, F. (2019). Decoupled Weight Decay Regularization. *International Conference on Learning Representations (ICLR 2019).* Freiburg, Germany: arXiv.

Mahmoud, I., Li, L., Wloka, D., & Ali, M. (2014). Believable NPCs in serious games: HTN planning approach based on visual perception. *2014 IEEE Conference on Computational Intelligence and Games* (s. 1-8). Dortmund, Germany: IEEE.

Miles, C. E. (2007). *Co-evolving real-time strategy game players.* ProQuest .

Ontañón, S. (2013). The Combinatorial Multi-Armed Bandit Problem and Its Application to Real-Time Strategy Games. *Proceedings of the 9th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, AIIDE 2013* (s. 58-64). Boston, MA, USA: AAAI Press.

Ontañón, S., Synnaeve, G., Uriarte, A., Richoux, F., Churchill, D., & Preuss, M. (2013). A Survey of Real-Time Strategy Game AI Research and Competition in StarCraft. *IEEE Transactions on Computational Intelligence and AI in Games Volume 5 No 4*, 293-311.

Pan, S. J., & Yang, Q. (2010). A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, 1345-1359.

Perez-Liebana, D., Liu, J., Khalifa, A., Gaina, R. D., Togelius, J., & Lucas, S. M. (2019). General Video Game AI: A Multitrack Framework for Evaluating Agents, Games, and Content Generation Algorithms. *IEEE Transactions on Games Volume 11, No 3*, 195–214.

Pomerleau, D. A. (1989). ALVINN: An Autonomous Land Vehicle in a Neural Network. D. Touretzky içinde, *Advances in Neural Information Processing Systems.* Morgan-Kaufmann.

Prada, R., Lopes, P., Catarino, J., Quitério, J., & Melo, F. S. (2015). The geometry friends game AI competition. *2015 IEEE Conference on Computational Intelligence and Games* (s. 431-438). Tainan, Taiwan: IEEE.

Rohlfshagen, P., Liu, J., Perez-Liebana, D., & Lucas, S. M. (2018). Pac-Man Conquers Academia: Two Decades of Research Using a Classic Arcade Game. *IEEE Transactions on Games Volume 10, No 3*, 233-256.

Schaal, S. (1997). Learning from demonstration. *Advances in Neural Information Processing Systems 9* (s. 1040-1046). Cambridge, MA: MIT Press.

Schaal, S. (1999). Is imitation learning the route to humanoid robots? *Trends in Cognitive Sciences Volume 3 No 6*, 233-242.

Sharma, M., Holmes, M., Santamaría, J., Irani, A., Jr, C., & Ram, A. (2007). Transfer Learning in Real-Time Strategy Games Using Hybrid CBR/RL. *Proceedings of the 20th International Joint Conference on Artificial Intelligence* (s. 1041-1046). Hyderabad, India: DBLP.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., . . . Lanctot, M. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature 529*, 484–489.

Šustr, M., Malý, J., & Čertický, M. (2018). Multi-platform Version of StarCraft: Brood War in a Docker Container: Technical Report. *ArXiv*.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction 2nd Edition.* Cambridge, MA, USA: The MIT Press.

Synnaeve, G., & Bessiere, P. (2012). A Dataset for StarCraft AI \& an Example of Armies Clustering. *Artificial Intelligence in Adversarial Real-Time Games 2012*, 25-30.

Torabi, F., Warnell, G., & Stone, P. (2018). Behavioral Cloning from Observation. *Proceedings of the 27th International Joint Conference on Artificial Intelligence* (s. 4950–4957). Stockholm, Sweden: AAAI Press.

Torrey, L. (2010). Crowd Simulation Via Multi-Agent Reinforcement Learning. *Proceedings of the Sixth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* (s. 89–94). Stanford, CA, USA: AAAI Press.

Vinyals, O., Babuschkin, I., Czarnecki, W., Mathieu, M., Dudzik, A., Chung, J., . . . Dalibard, V. (2019). Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature Volume 575, No 7782*, 350-354.

Vinyals, O., Ewalds, T., Bartunov, S., Georgiev, P., Vezhnevets, A., Yeo, M., . . . Tsing, R. (2017). StarCraft II: A New Challenge for Reinforcement Learning. *CoRR*, 1-20.

Vinyals, O., Ewalds, T., Bartunov, S., Georgiev, P., Vezhnevets, A., Yeo, M., . . . Tsing, R. (2017). StarCraft II: A New Challenge for Reinforcement Learning. *CoRR*, 1-20.

Weber, B., & Mateas, M. (2009). A data mining approach to strategy prediction. *2009 IEEE Symposium on Computational Intelligence and Games*, 140-147.

[1] https://www.kaggle.com/c/lux-ai-2021, Erişim Tarihi: 17 Ocak 2022

[2] https://smartlabai.medium.com/a-brief-overview-of-imitation-learning-8a8a75c44a9c, Erişim Tarihi: 27 Aralık 2021

[3] https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms, Erişim Tarihi: 17 Ocak 2022
[4] https://image-net.org/challenges/LSVRC, Erişim Tarihi: 16 Ocak 2022
[5] https://machinelearningmastery.com/transfer-learning-for-deep-learning, Erişim Tarihi: 16 Ocak 2022
[6] https://www.kaggle.com/kaggle/meta-kaggle, Erişim Tarihi: 22 Ocak 2022
[7] https://www.kaggle.com/robga/simulations-episode-scraper-match-downloader, Erişim Tarihi: 26 Ocak 2022
[8] https://www.kaggle.com/c/hungry-geese/discussion/263279, Erişim Tarihi: 17 Ocak 2022
[9] https://towardsdatascience.com/cross-entropy-for-classification-d98e7f974451, Erişim Tarihi: 23 Ocak 2022
[10] https://www.kaggle.com/isbhargav/guide-to-pytorch-learning-rate-scheduling, Erişim Tarihi: 17 Ocak 2022
[11] https://github.com/UgurIpekduzen/Lux-AI-2021, Erişim Tarihi: 3 Şubat 2022
[12] https://www.kaggle.com/uipkdzn/luxai-il-pretrained-models-with-large-data, Erişim Tarihi: 17 Ocak 2022
[13] https://www.kaggle.com/uipkdzn/lux-ai-with-il-ensemble-of-two-optimizers?scriptVersionId=81554042, Erişim Tarihi: 3 Şubat 2022
[14] https://www.kaggle.com/uipkdzn/lux-ai-with-il-ensemble-of-two-optimizers?scriptVersionId=81691067, Erişim Tarihi: 3 Şubat 2022