T.C.
GEBZE TEKNİK ÜNİVERSİTESİ

Bilgisayar Mühendisliği Bölümü

Competitive Planning on Autonomous Drones with Reinforcement Learning

Uğurkan ATEŞ

Danışman
Dr.Öğr.Üyesi Gökhan KAYA

Ocak, 2020
Gebze, KOCAELİ

T.C.
GEBZE TEKNİK ÜNİVERSİTESİ

Bilgisayar Mühendisliği Bölümü

Competitive Planning on Autonomous Drones with Reinforcement Learning

Uğurkan ATEŞ

Danışman
Dr.Öğr.Üyesi Gökhan KAYA

Ocak, 2020
Gebze, KOCAELİ

Bu çalışma ..../..../2019 tarihinde aşağıdaki jüri tarafından Bilgisayar Mühendisliği Bölümü'nde Lisans Bitirme Projesi olarak kabul edilmiştir.

Bitirme Projesi Jürisi

Danışman Adı	Dr.Öğr.Üyesi Gökhan KAYA	
Üniversite	Gebze Teknik Üniversitesi	
Fakülte	Mühendislik Fakültesi	

Jüri Adı	Doç.Dr.Mehmet GÖKTÜRK	
Üniversite	Gebze Teknik Üniversitesi	
Fakülte	Mühendislik Fakültesi	

ÖNSÖZ

Bu kılavuzun ilk taslaklarının hazırlanmasında emeği geçenlere, kılavuzun son halini almasında yol gösterici olan Sayın Dr.Öğr.Üyesi Gökhan KAYA hocama ve bu çalışmayı destekleyen Gebze Teknik Üniversitesi'ne içten teşekkürlerimi sunarım.

Ayrıca eğitimim süresince bana her konuda tam destek veren aileme ve bana hayatlarıyla örnek olan tüm hocalarıma saygı ve sevgilerimi sunarım.

Ocak, 2020 **Uğurkan ATEŞ**

İÇİNDEKİLER

ŞEKİL LİSTESİ

KISALTMA LİSTESİ

API **:** Uygulama Programlama Arayüzü (Application Programming Interface)
G.T.Ü **:** Gebze Teknik Üniversitesi
POLICY **:** Pekiştirmeli öğrenme algoritmalarında ajan(agent'in) yönetilmesini sağlayan karar mekanizması
FPV : First Person View , Drone'larda kullanılan birinci kamera açısı
CNN : Convolutional Neural Network , görüntü tanımada kullanılan bir yapay sinir ağı modeli
MDP : Markov Decision Process
MDT : Markov Decision Tree
VRAM : Video Hafızası /Memory
PPO : Policy Proximal Optimization , bir çeşit pekiştirmeli öğrenme algoritması

ÖZET

Drone(İnsansız Hava Aracı) yarışları son 15 yılda sadece bu konuya ilgili insanların uğraşmakta olduğu bir alandan çıkıp kontrol mühendisliği, bilgisayar mühendisliği ve bilgisayar bilimcilerinin uğraşmaya başladığı ve yeni yöntemler geliştirmeye başladığı bir alan haline gelmiştir.Yapısı gereği içerisinde bulunan hızlı rota planlama gereksinimi , kontrol mekanizmaları ve quadrocopterlerin konumunun tahmin edilmeye çalışılması son 10 yıl içerisinde sadece robotik ile uğraşan araştırmacıların alanı olmaktan çıkıp bilgisayarlı görü,makine öğrenmesi ve pekiştirmeli öğrenme,baştan uca öğrenme gibi yapay zeka tabanlı alanların konusu olmaya başlamıştır. Bu projeyi gerçekleştirmekte ki amacım pekiştirmeli öğrenme kullanarak herhangi bir etiketli veriye sahip olmaksızın drone'nun optimal rotayı tespit edebilmesini sağlamak , bunu klasik robotik teknikleriyle karşılaştırmak , farklı ortamlarda generalize olabilmesini sağlamak ve pekiştirmeli öğrenme ile robotların zorlu kontrol koşullarında stabil hareket edebilme yetisini kazanmasıdır.

Bu çalışmalar kapsamında Microsoft AirSim isimli otonom araç ve drone simülasyonları için arkataban sağlayan altyapı kullanılmıştır ve 4 kanatlı drone'nun drone yarışları için tasarlanmış pistte hiç bir ön bilgi verilmeden , sensörler verilerini kullanarak tasarlanmış yapay zeka sinir ağları modeli ve pekiştirmeli öğrenme algoritması ile optimal rota'yı keşfetmesi sağlanmıştır.

Bu çalışma ile birlikte pekiştirmeli öğrenme sistemlerinin veri toplaması zor olan koşullarda ve kontrol problemlerinde kullanılabileceği gösterilmiş ayrıca Microsoft firmasının düzenlediği NeurIPS 2019 yarışmasında katılım sağlanmıştır.

Anahtar kelimeler : **Pekiştirmeli Öğrenme, Yapay Sinir Ağları , AirSim , Robotik , Drone , Quadrotor**

SUMMARY

Drone racing has transformed from a niche activitiy done by enthusiastic hobbyists to an internationally and multidisciplinary area in last 15 years.Many disciplines such as control engineering , computer engineering and computer scientists have tackled problem and developed their solutions which all distinct from each other. Drone racing requires agile trajectory planning , control Yapısı gereği içerisinde bulunan hızlı rota planlama gereksinimi , control loop mechanism and estimation of quadrocopter's state which all 3 areas interested by people in computer vision , machine learning , reinforcement learning and end to end learning. By implementing this project I was able to teach a drone to detect and learn most optimal route given its internal sensors without any labeled data. Comparing this model learned by reinforcement learning to classical approaches from robotic algorithms and gain ability to generalize this algorithm in different scenearios with hard to solve planning tasks.

In this project I have used Microsoft AirSim open source autonomous car and drone backend simulation. Quadrocopter without any additional information given learned how to drive on drone racing environment with its own internal sensors using neural networks and reinforcement learning.

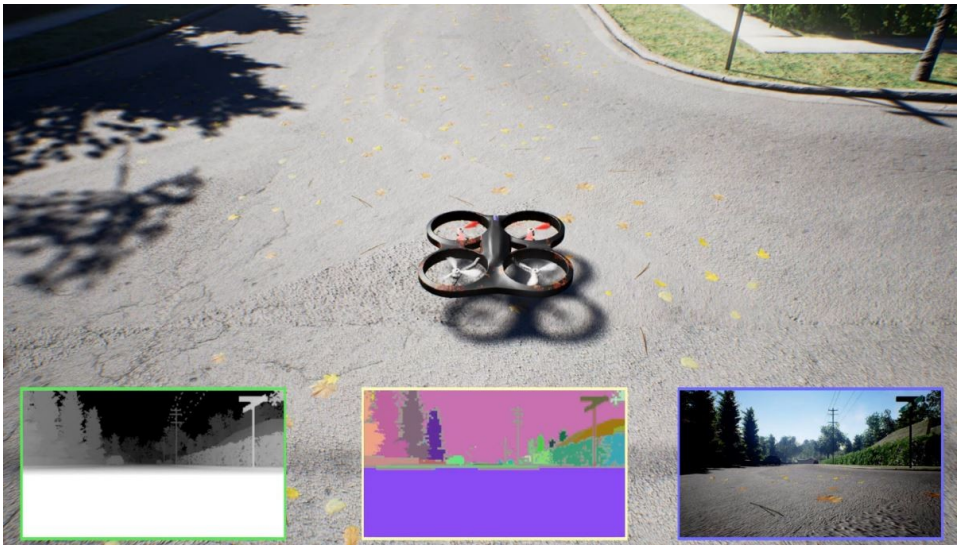
With this work I show how reinforcement learning systems could be used in planning tasks that are very hard to collect data and I also attended Microsoft's NeurIPS 2019 drone racing challenge .

Keywords **: Reinforcement Learning , Neural Networks , Robotics , AirSim , Drone, Quadrotor**

1. GİRİŞ

1.1 PROJENİN ANA YAPISI

Son zamanlarda oldukça hızlı bir şekilde gelişmekte olan otonom sürüş kategorileri içerisinde bulunan otonom hızlı uçuş alanı makine öğrenmesi ve robotik alanlarının ilgilendiği birçok problemi içerisinde kapsamaktadır. Bunlardan bazıları stabil olmayan konum tahmininin etkisini en aşağıya indirebilmek , kamera görüntüsü kullanılarak yapılan fark etme algoritmalarında ki gerçeklik payını arttırabilmek ve sınırlı zaman koşullarında dinamik bir sürüş gerçekleştirebilmektir. Bu alanda kullanılan CNN- Konvolüsyonel Sinir Ağları algoritmaları oldukça fazla elle etiketlenmiş veri istemektedir ve çoğu ortamda bunu yapabilmek mümkün değildir.



Şekil 1.1 : AirSim Simülasyonu Drone FPV görüntüsü

Proje kapsamında pekiştirmeli öğrenme (RL) ile bu problemin çözümü sağlanmıştır. Problem kapsamında simülasyon ortamında bulunan 4 kanatlı drone gerçek zamanlı olarak pisti tamamlayabilmekte bunu yaparken stabil bir rota çizerek kapılara

çarpmamaktadır. Drone sensörlerinden toplanan anlık linear hız , açısal hız , linear ivmelenme,açısal ivmelenme,X-Y-Z cinsinden pozisyon verisine ek olarak daha hızlı optimal rotayı keşfedebilmesi için klasik robotik ile kodlanmış bir drone eğitim süresinde kendisi ile birlikte aynı ortamda bulundurulmuş ve bu hazır rotayı giden 4 kanatlının ivme ve hız verileride aynı şekilde yapay sinir ağı modeline parametre olarak eklenmiştir.

Üzerinde geliştirilme yapılan AirSim simülasyon arka tabanı pekiştirmeli öğrenme problemleri için geliştirilmediği için simülasyon ortamına paket geliştirilerek pekiştirmeli öğrenme problemlerinde algoritmaları koşacak hale getirilmiştir.

Pekiştirmeli öğrenmenin çalışabilmesi için en önemli 5 etken olan aksiyon fonksonu(drone'nun alması gereken hareket) için diskret hareket veya sürekli hareket kararlarından sürekli hareket seçilmiştir. Bunun sebebi disrkret hareketin drone'nun kapılardan geçerken aldığı sert dönüş ve manevralardan ötürü kapılara çarpmasını engellemektir. X,Y,Z konumlarında şuan da bulunan hızına pekiştirmeli öğrenme modelinin çıktısı doğrultusunca ekleme yapılmış ve drone'nun çok hızlı veya çok yavaş manevralar yapması engellenmeye çalışılmıştır.

Simülasyona reset modülü eklenmiş ve drone gereken rotadan sapınca (mesafe tahmini ve kapılara yakınlık) ortamın resetlenmesi ve başlangıç konumuna getirilmesi sağlanmıştır.

Pekiştirmeli öğrenme gereğince yaptığı her hareket için alması gereken ceza mekanizması simülasyon içine eklenmiştir. Yaptığı her hareket için şuan ki sensör verileri ile hesaplanan ceza fonksiyonu çalışmaktadır. Bu fonksiyonun çıktısına göre drone'nun gerektiği kapıya yaklaşıp yaklaşmadığı ve rotada bulunup bulunmadığı tespit edilmektedir. Ceza fonksiyonunun detaylı tasarımı ilerleyen kısımlarda anlatılacaktır.

Modelin öğrendiği modeli belirli aralıklarla test edebilmesi için test modülü eklenmiştir. O ana kadar eğitilen model dosyası alınarak belirli aralıklara(epoch) modelin hareketi puanlandırılmakta ve bundan önceki en iyi süre ve ödülden iyiyse en iyi model olarak saklanmaktadır.

Modelin karar mekanizması 4 katmanlı yapay sinir ağı üzerinde çalışmaktadır. Alınan girdi verileri (hız , ivmelenme , pozisyon ve rakip bilgileri) her hareket sonunda ağa girdi olarak girmektedir. Ağda aktor – kritik modeli kullanılmıştır. Bu model yapısında ağın bir kısmı yeni kararlar almak için çıktılar üretirken(aktör) diğer kısmı aktör yapısının ürettiği tahminlerin kritiğini yapmaktadır. Bu yapı Markov Karar Ağaçları(Markov Decision Tree) yapısının benzeridir. MDT'ler pekiştirmeli öğrenmenin temeli olduğu için bu yapıyı yapay sinir ağları ile birleştirince yüksek başarı alınabilmektedir. Aktivasyon fonksiyonu olarak ReLU aktivasyon fonksiyonu kullanılmıştır. Kayıp optimize fonksiyonu olarak Adam kullanılmıştır. Adam benzer çalışmalarda kullanılan ve başarılı sonuçlar veren bir fonksiyondur.

Kullanılan sinir ağı ardışık(sequential) sinir ağı modelindedir. Eğitim süresince Google Bulut bilgisayarları üzerinden çalışıtırlma gerçeklenmiştir. Bu sinir ağlarının eğitilmesi ve simülasyonun aynı anda çalışması için GPU (paralel matriks çarpımları) ve CPU(multi thread haberleşme ve fizik motoru) seviyesinin belirli kapasiteye sahip olması gerekmektedir. Oluşturulan model üzerinde eğitim yapmadan test amaçlı çalıştırmak için 2-3GB VRAM (video hafıza)'ya sahip bir NVIDIA ekran kartı ve Intel/AMD serisi modern sayılabilecek işlemci ile 8GB ve üzeri RAM ihtiyacı olmaktadır. Eğitim süresi ve detaylarından ilerleyen kısımlarda bahsedilecektir.

1.1 RAPORUN AKIŞI

Bölüm 1’de giriş,projenin amacı , proje kapsamında kullanılan AirSim simülasyonu hakkında bilgi , 4 kanatlı drone’larda çözülmeye çalışılan sorunlar ve pekiştirmeli öğrenmenin bunlara nasıl katkı sağlayacağı belirtilmiştir.

Bölüm 2’de proje kapsamında kullanılan yapay sinir ağlarından ve proje içinde kullanılan ağın yapısından bahsedilmiştir.

Bölüm 3’de pekiştirmeli öğrenme , pekiştirme öğrenme modelleri arasında ki farklar ve proje kapsamında seçilen PPO(Proximal Policy Optimization) algoritmasından bahsedilmiştir.

Bölüm 4’de AirSim simülasyonuna pekiştirmeli öğrenme altyapısının nasıl eklendiğinden bahsedilmiştir

Bölüm 5’de gerçekleştirilen deneyin hazırlanma süreci ve sonuçlarından bahsedilmiştir.

Bölüm 6’da projenin eksiklerinden ve projede yapılabilecek geliştirilmelerden bahsedilmiştir.

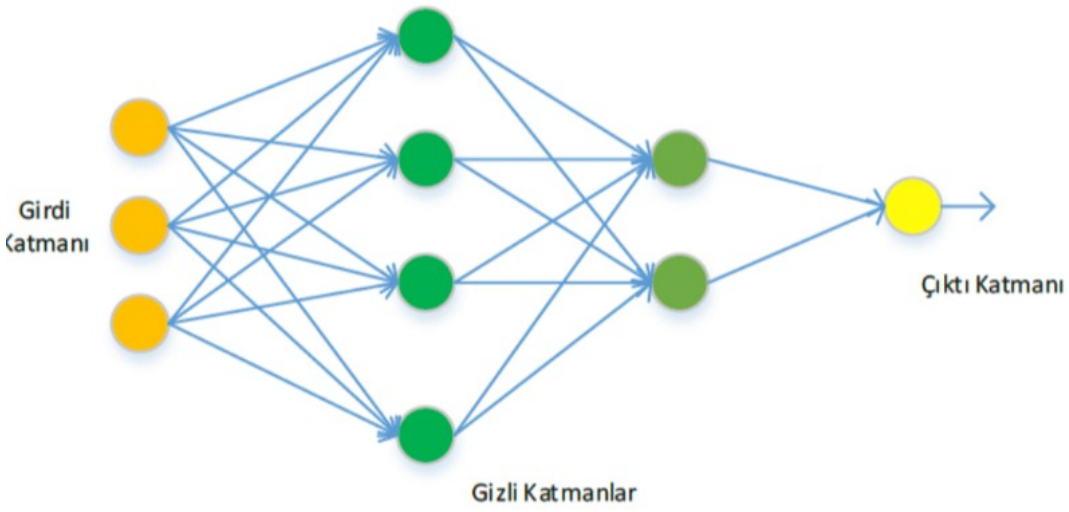
2. YAPAY SİNİR AĞLARI

2.1 YAPAY SİNİR AĞI NEDİR

Yapay sinir ağları basit biyolojik sinir sisteminin çalışma şeklinden esinlenerek tasarlanan makine öğrenmesi yaklaşımıdır. Temek olarak girdi katmanı, ara katman

ve çıktı katmanlarında oluşan yapay sinir ağları; ağırlıklandırılmış şekilde birbirlerine bağlanmış birçok işlem biriminden (nöronlar) oluşan matematiksel sistemlerdir.

Sinir ağları katmanlara bağlı nöronların bir yapısı olarak modellenmiştir. İleri beslemeli yapay sinir ağı modelinde; belirli bir katmanın nöronlarının çıktıları, bir sonraki katmanın girdilerini bağlantı ağırlıkları ile oluşturur. Girdi her zaman ileri doğru (aktivasyon yönü) iletilmektedir. Bu yüzden aynı katmanda yer alan katmanlar ise ara katman ya da gizli katman olarak adlandırılmaktadır. Tipik bir gizli katmana sahip YSA modeli Şekil 4-1'de verilmiştir. YSA modeline bağlı olarak; ilk katmanlarda girdi sinyalinden basit bilgiler çıkarılırken ikinci ve üçüncü gizli katmanlarda ise veri kümesine özel detaylı bilgiler çıkarılmaktadır. Derin sinir ağları yapılarında ilgili derinliği sağlayan katmanlar gizli katmanlardır. Gizli katmanların sayısının artması durumunda oluşturulan sinir ağı modeli daha derin ve karmaşıklığı artan bir yapıya dönüşmektedir. Buna ek olarak oluşturulan gizli katmanda yer alan nöron sayısı da YSA başarımını ve karmaşıklığını etkileyen önemli parametrelerden bir tanesidir.



Şekil 2-1 : İki gizli katmanlı Yapay Sinir Ağı

2.2 TAM BAĞLI SİNİR AĞI

Tam bağlı katmandaki nöronlar bir önceki katmandaki tüm etkinleşmelere tam bağlantılara sahiptir ve üretilen aktivasyon haritalarına dayalı büyük sınıflandırma işleminden büyük sınıflandırma işleminde sorumludurlar. Bu katmanda ileri besleme esnasında oluşan matematiksel ifade aşağıdaki denklem ile verilmektedir.

$$u_i^l = \sum w_{ji}^{l-1}\ y_j^{l-1}$$

$$y_i^l = f\left(u_i^l\right) + b^{(l)}$$

Şekil 2.2 İleri Besleme Matematiksel Formülü

Bu denklemde yer alan l, (l-1) katman numarası, i, j ise nöron numarasını, $yi\ l$ oluşturulan çıktı katmanındaki değeri, $wji\ l-1$ gizli katmanda yer alan ağırlık değerini, $yj\ l-1$ girdi nöronlarının değerlerini, $ui\ l$ aktivasyon fonksiyonu öncesi çıktı katmanı değeri ve $b\ (l)$ ise sapma değerini vermektedir. Burada belirtilen $f(.)$ fonksiyonu ise aktivasyon fonksiyonunu ifade etmektedir.

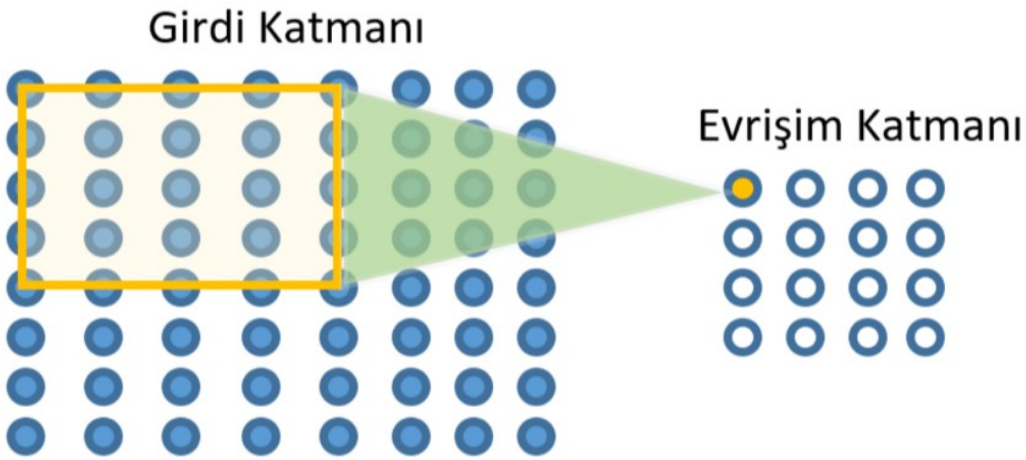
2.3 EVRİŞİMSEL SİNİR AĞLARI

Günümüzde evrişimsel sinir ağları ve derin öğrenme tabanlı çözümler pek çok sorunda iyi sonuçlar vermektedir. Görüntü tanıma, ses tanıma ve doğal dil işleme gibi birçok farklı alanda uygulanabilmektedir. Özellikle görünür bantta elde edilmiş görüntülerin sınıflandırılmasında; olgun, başarılı ve popüler yöntem olan DESA modeli resim üzerinde bir pikseli sınıflandırmaktan ziyade imgeyi genel olarak sınıflandırmaya yönelik çalışmalarda kullanılmıştır.

Derin evrişimsel sinir ağlarında; girdi katmanı, evrişim katmanı, havuzlama katmanı, tam bağlı katman ve çıktı katmanı gibi katmanlar yer almaktadır. Ancak evrişimsel sinir ağı genellikle belirli bileşenler grubundan oluşmaktadır. Bu

bileşenler, soruna göre özelleştirilmekte ve bahsi geçen bütün katmanları içermeyebilmektedir. Derin evrişimsel sinir ağı temel olarak 3 yaklaşıma sahiptir.

1-) Yerel Alıcı Alanlar (Local Receptive Fields): Girdi sinyali ile gizli evrişim katmanında yer alan filtreler bölgesel alanlarda Şekil 4-2'de gösterilen şekilde bağlılık kurmaktadır. Bağlılık kurulan bu alanlar ise yerel alıcı alan olarak adlandırılmaktadır. Her bir yerel alıcı alanın evrişim katmanında sonuç oluşturduğu bir nöron bulunmaktadır.



Şekil 2.3 : Yerel Alıcı Alan Gösterimi

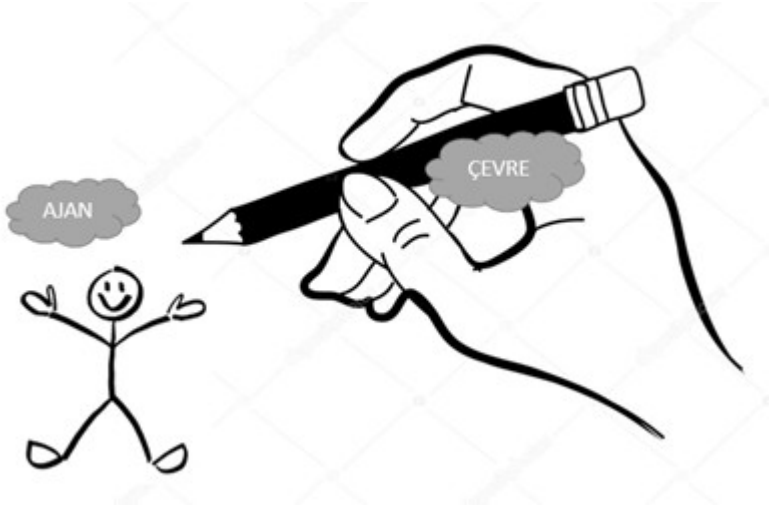
2-) Paylaşılan Ağırlıklar (Shared Weights) : Evrişimsel sinir ağlarında her bir evrişim filtresine özel bir öznitelik haritası oluşmaktadır. Bu öznitelik haritası oluşturulurken imgenin bütün bölgesinden benzer özellikleri çıkaran filtreler kullanılmaktadır. Evrişim filtresinde yer alan parametreler ise paylaşılan ağırlık değerleri olarak adlandırılmaktadır. Evrişimsel sinir ağları tarafından sağlanan bu özellik ile öğrenilen serbest parametre sayısı büyük ölçüde azaltılmaktadır. Bu sayede model üzerinde daha iyi bir genelleme sağlanabilmektedir.

3-) Alt Örnekleme: Alt örnekleme katmanında ise öznitelik haritalarının çözünürlüğü düşürülmektedir

3. PEKİŞTİRMELİ ÖĞRENME

3.1. PEKİŞTİRMELİ ÖĞRENME NEDİR

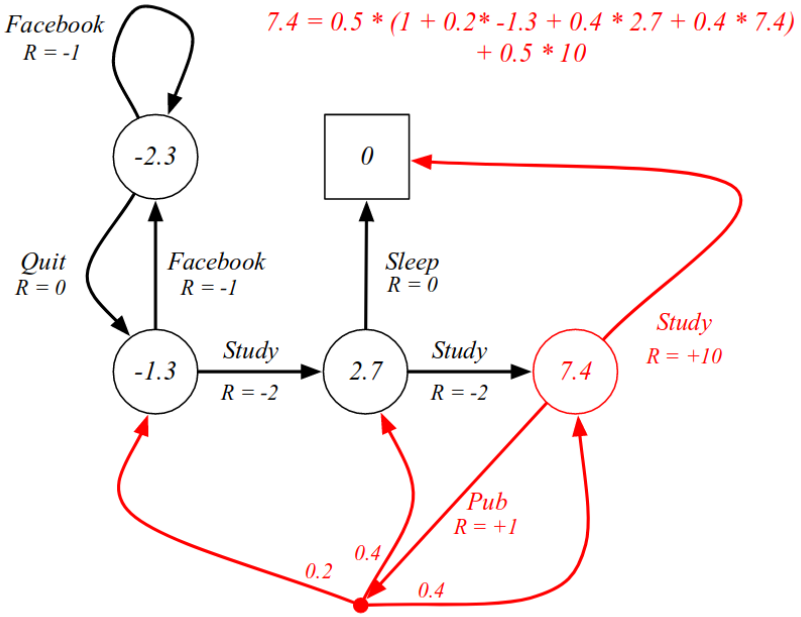
Pekiştirmeli öğrenme, amaca yönelik ne yapılması gerektiğini öğrenen bir makine öğrenmesi yaklaşımıdır. Pekiştirmeli öğrenmede *ajan (agent)* adı verilen öğrenen makinemiz karşılaştığı durumlara bir tepki verir ve bunun karşılığında da sayısal bir ödül sinyali alır. Ajan/öğrenen makine aldığı bu ödül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



Şekil 3.1a Çevre – Ajan figürü

Pekiştirmeli Öğrenme, Markov karar süreci denilen bir model kullanmaktadır. Markov karar süreçlerinin en önemli 3 özelliği; algılama (sensation), eylem (action) ve hedef (goal). Bu 3 hedefi gerçek uygulamalarda kullanmamıza 4 karşılığı pekiştirmeli öğrenme standartı haline gelmiştir bunlar Action- Step- Next Step- Reward olmaktadır. Ayrıca eğitim süresince gerekli olan bir kavram Reset(eğitim konumuna tekrar gelinmesi) durumudur.

Pekiştirmeli Öğrenme sürecindeki en önemli zorluklar, keşif (exploration) ve sömürü(exploitation) 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/ödül alabildiklerini aşamalı olarak desteklemelidir. Keşif-Sömürü ikilemi matematikçiler tarafından yıllardır optimum bir şekilde çözülmeye çalışılmaktadır.



Şekil 3.1b Markov Karar Ağacı (Markov Decision Tree)

Tüm pekiştirmeli öğrenme ajanları açık hedeflere sahip olup çevrenin özelliklerini algılayabilir ve çevrelerini etkileyebilecek eylemleri seçebilirler.*Ajan denildiği zaman bir organizma veya robot gibi bir varlık ifade edilmemektedir. Ajan, eylemi gerçekleştiren ve öğrenen etkendir.*

3.2 PEKİŞTİRMELİ ÖĞRENME ÖĞELERİ

Keşif-Sömürü ikilemi matematikçiler tarafından yıllardır optimum bir şekilde çözülmeye çalışılmaktadır.Bir Pekiştirmeli Öğrenme sisteminde ajan ve çevre (environment) dışında biri opsiyonel olmak üzere dört unsur bulunur:

- Politika (policy)
- Ödül (reward signal)
- Değer/Durum Değeri (value function)
- Çevre modeli (model)

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

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

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

Son unsurumuz olan **model** ise, isteğe bağlı olarak sisteme dahil edilen bir unsurdur. Çevrenin bir simülasyonu olup ajanın bir aksiyonu gerçekleştirmeden önce bu aksiyon sonucunda alabileceği ödülü ve doğuracağı durumu tahmin etmesini sağlamaktadır. Bu sayede bir planlama yapılarak ajanın davranışında değişiklik meydana gelebilecektir.

3.3 PEKİŞTİRMELİ ÖĞRENME MODELLERİ KARŞILAŞTIRMASI

Şuanda bulunan en popüler Pekiştirmeli Öğrenme algoritmaları iki sınıfın içine girmektedir. Bunlar **Q Learning** tabanlı algoritmalar ve **Policy Gradient(Karara Eğilimli)**, algoritma sınıflarıdır.

Q Learning tabanlı algoritmalarda amacımız ortamda bulunan agent'in ortamla etkileşim kurduğunda kazandığı state/action/reward/nextstate değişkenlerini kullanarak olabildiğince en yüksek Q değerine ulaşmasını sağlamaktır. Q değerinin matematiksel karşılığı şimdiki aldığı karar ve duruma bakarak gelecekte alabileceği beklenen toplam ödül miktarıdır.Yani kısacası şimdi aldığı karara bakarak ileri de en yüksek puanı toplamasını sağlamak. Bu işlem yapılırken bir sonraki durumda vereceği kararın yükseliği en yüksek (0.9) ve geriye kalan tüm kararlar(expected 0.1) oranında hesaplanmaktadır.

$$Q^{\pi}(s,a) = E[R_t]$$

Şekil 3.3a : Q (kümülatif reward) fonksiyonu

Burda pi ile gösterilen Policy/Karar mekanizmasının temsilidir. Policy mekanizmasının sonuçlarına göre bir sonraki durumda ne karar alacağını seçmektedir.

Rt dediğimiz yapı ise beklenen ödüllerin öncelik sırasına göre sıralanmış halidir. Şuan ki kendisine yarayan en yüksek ödül bir sonraki aşamada kazanacağı ödüldür , ve en son ödül en düşük önceliklidir.

$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots$$

Şekil 3.3b : Beklenen ödüller fonksiyonu (Expected Rewards)

Q learning'in temel amacı eğer elimizde yeterince tutarlı bir Q fonksiyonu varsa optimal karar mekanizasmasını en fazla ödülü vericek olan karar mekanizması olarak seçebiliriz şeklindedir.

Burda ki ideal Q değerinin hesaplanmasınıda yapay sinir ağları devreye girmektedir. Herhangi bir kayıp fonksiyonu(Adam, Min squared error gibi) seçilerek bu değer her sinir ağı iterasyonunda optimize edilmekte ve istenilen ödül değerlerine yaklaşana kadar ağ eğitilmektedir.

Q Learning ve Deep Q Network adı verilen bu yapının temel sorunlarından biri öğrenilmesi gereken ve optimize edilen ödül fonksiyonunun ağın öğrenmesi için çok kompleks yapıda olmasıdır. Böyle durumlarda DQN ağı hiçbir zaman istenilen değer aralığına converge/yaklaşma olmayacak ve DQN başarısız kalacaktır.

Policy Gradient(Karar Eğilimli) algoritmalar ise Q learning modellerinde görülen bu sorunları gidermek ve kompleks problemlere uygulanabilmesini sağlamaktır. Burada kompleks problemden anlatılmak istenen sınırlı sayıda hareket bulunmayan ve alınabilecek kararların sınırlı olmadığı ortamlardır. Örneğin bir oyun ortamında belirli sayıda (kontrolcüde ki tuş sayısı) girdi ve belirli sayıda çıktı mevcuttur. Fakat otonom sürüş gibi bir problemde sınırsız sayıda girdi ve çıktı bulunmaktadır. (Aracın x,y,z değerleri sabit sayılar almamaktadır)
Policy Gradient methodlar Q değerini optimize etmek yerine onu hesaplayan Policy (Pi) değerlerini optimize etmeye çalışmaktadır. PG algoritmalarında yapay sinir ağı gibi fonksiyon tahmincilerini kullanarak modelin alabileceği aksiyon olasılıklarını hesaplıyoruz. Ajan her seferinde ortamla etkileşime geçtiğinde bir çıktı (durum/sonraki durum/ ödül / aksiyon) hesaplamaktadır. PG'de ki temel fark Q'yi

optimize etmek yerine policy'i artırıacak ve maksimum hale getirecek kararlar verilmesini sağlamaktır. Bunu yaptığımızda oluşan policy zaten bize optimal bir ödül getirecektir. Gradient formülü aşağıda verilmiştir.

$$\nabla_\theta E[R_t] = E[\nabla_\theta log P(a) R_t]$$

Şekil 3.3c : Policy Gradient Formülü

Bu formülde bulunan E[RT] çarpımına **REINFORCE** denilmektedir. Rt değişkeni burada büyüme faktörü görevinde rol oynamaktadır. Yani aksiyon (a) iyi bir aksiyon ise Rt değeri büyüyecek ve bizim karar mekanizmamız olan P(a) büyüyecektir. Doğru kararı aldığımzı ve böyle bir durumla tekrar karşılaştığıdna vermesi gereken kararda ki yüksek öncelikli olasılığın değerini bulabiliyoruz. Eğer kötü bir karar alınmışsa bu sefer tam tersi işlem yapılacak ve P(a) azalacaktır. Bu sayede öğrenme işlemi gerçekleşmiş oluyor.

PG direk ödül fonksiyonuna odaklanmadığı için ve sadece karar mekanizmasını optimize etmek tabanlı olduğu için çok zor problemlere adapte olabilmektedir. PG algoritmaları DQN tabanlı algoritmalara göre çok uzun eğitim süreçlerine sahiptir. Bu sebeple doğru algoritmanın seçilmesi kritik değer taşımaktadır.

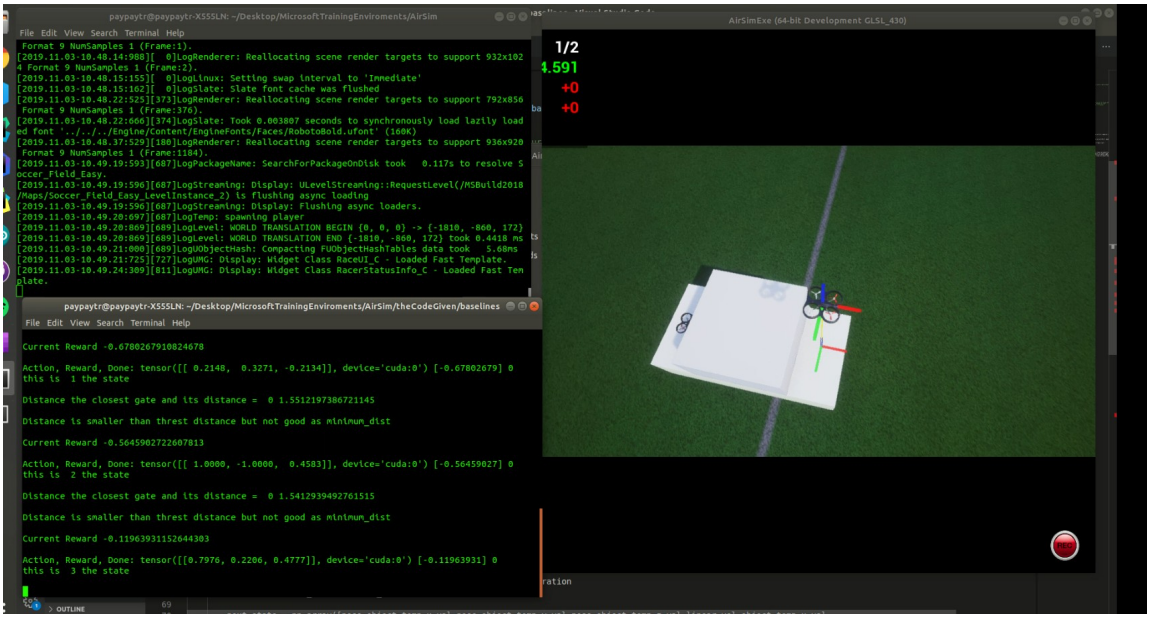
Benim seçtiğim algoritma Policy Gradient algoritmalarından olan PPO(Policy Proximal Optimization) algoritmasıdır. Kendisinden önce çıkmış olan TRPO(Trust Region Policy Optimization)'dan farkı çok daha az değiştirilebilir parametre olmasıdır. Bu sayede anlaşılması daha kolay ve gerçekleşemesi daha basittir.

PPO aktor kritik bir ağ yağısında çalışmaktadır. Öğrenmesi gereken policy optimizasyonu işlemini yaparken yapay sinir ağı üzerinde bir model tahminleri üretmekte(optimal karar tahminlerini) diğer model ise aktörün verdiği tahminlerin değerlendirmesini yapmakta ve gelecekte ne kadar başarılı olabileceğini aynı şekilde

tahmin yapmaktadır. Bu sayede model hiç görmediği ortamlarda beklenmedik hareketlerde bulunmamakta ve belli sınırlar içinde öğrenmektedir.

3.4 PPO MODELİ PARAMETRE SEÇİMİ

2017'de yayınlanan PPO makalesinde verilen parametreleri olabildiğince kullanmaya çalıştım. Gamma değeri (Discount / Sonraki ödülleri umursamama miktarı) **0.99 ,** Ağın Öğrenme Oranı (**1e-4 = 0.0001**) , Entropi değeri(ağın sürekli takılmaması ve çevresinde bulunan hiç gezmediği hareketleri gezmesini teşvik eden oran) **0.001** , Aktor kritik modelde Kritiğin kararlarının etkisi **0.5 ,** yapay sinir ağında her katmanda bulunan nöron sayısı **256 (** makalede 128 belirtilmişti fakat benim problemimde 256 daha başarılı çalıştı) , eğitim süresince hesaplanan test aralığı **10**(10 deneme'de bir test yapılmaktaydı), PPO'nun eğitim deneme adımlarını ise simülasyon çerçevesinde modifiye ettim. Normal'de OpenAI şirketinin robotik ve oyun ortamlarında denendiği için biraz adım sayıları önerilmiş örneğin bir robotun yürümeyi bilmeden yürümeye çalışması için 256 ve 128 adım yeterlidir. Fakat benim simülasyonumda 10 kapıdan geçmesi için 1024 ve üzeri adımlar denendi.



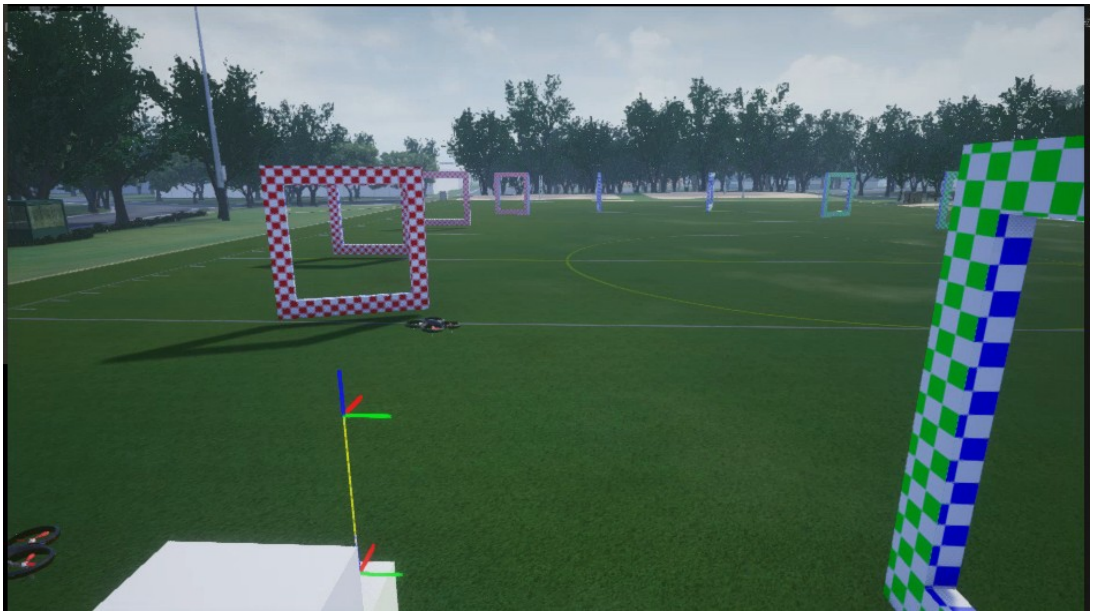
Şekil 3.4 – PPO modeli simülasyona bağlıyken tahmin ettiği hız değerleri

PPO modeli ve aktor kritik yapısı PyTorch kütüphanesiyle gerçekleştirildi. Modelin oluşturduğu hareket çıktıları (drone'un x,y,z hızlarına eklenmesi gereken hız miktarı) **[-1,+1]** arasında sıkıştırıldı. Bunun sebebi drone'un olabildiğince optimal rotayı hızlı öğrenmesidir. Bu tarz normalize işlemleri derin öğrenme ve uygulamalarında çok sık görülmektedir. Eğer normalize işlemi yapılmazsa matriks çarpımları uzun sürmekte ve çoğu zaman converge/yakınlaşma işlemini yapamamaktadırlar.

4 AIRSIM VE PEKIŞTIRMELI ÖĞRENME

4.1 AIRSIM SİMÜLASYONUNUN TEKNIK YAPISI

Airsim Microsoft firmasının satın aldığı ve geliştirmelerine devam ettiği bir otonom araç ve drone simülasyon platformudur. Açık kaynaklı yapısı ve sürekli geliştirilmeler yapılmasından dolayı yeni teknolojilere açık ve dinamik bir yapıda geliştirilmektedir. Bu simülasyonu seçmem de ki kararlardan biri sürekli aktif özellik ve geliştirilmeler eklenmesi , yarışma kapsamında kullanılması , drone çalışmalarında rakiplerine göre daha ileride olması, gerçekçi görüntü(Unreal Engine 4 oyun motoru) sayesinde sunabilmesidir.



Şekil 4.1 Drone Yarış Simülasyon ortamı

Simülasyon alt tabanı ne kadar gerçekçi fizik motoru ve görüntüler sağlayabiliyorsa burda eğitilen modellerin gerçek ortama aktarıldığında ki başarısı o kadar yüksek olacaktır. Simülasyon ile gerçek dünyanın birleştirilmesi için yapılan Sim-to-Real transfer learning algoritmaları başlı başına büyük bir araştırma konusudur. Bu konuda yapılan en başarılı çalışmalar gerçek dünya ile simülasyon ortamında ki farkları en aza indirmek için yapılan çalışmalardır. Gerçek ortamda üretilen veriler ile simülasyon verileri harmanlanarak randomize bir şekilde ağ eğitilmektedir. Bu şekilde veriler üretebilmek için gerçek dünyada çok efor isteyen etiketleme çabaları ve ekipmanı gerekmektedir.

Bu sebeple etiketsiz öğrenme ve pekiştirmeli öğrenme gibi iki yöntem büyük vaadler içermektedir. AirSim simülasyonu resmi olarak herhangi bir model veya algoritma yapısına direkt destek vermemektedir. Fakat drone üzerinde bulunan FPV kamera görüntüleri , drone sensörleri ve çeşitli dış ortam değişkenlerini ayarlayabilmekte ve istenilen algoritma için ortamı sağlayabilmektedir.

Unreal Engine 4 üzerinde çalışan simülasyon ortamıyla haberleşmek , modele veri toplamak ve modelden oluşturulan karar mekanizmasını(hız değerinin değiştirilmesi) sağlamak amacıyla Python dilinde yazılmış olan msgpack adlı uygulamayı kullandım.

MSGpack çoklu threadler ile çalışmayı destekliyor, bu sayede drone kamera görüntülerini , diğer drone bilgilerini ve kendi drone bilgilerimi ayrı threadlerde tuttum. Arada ki gecikme miktarı oldukça düşük ve farkedilmeyecek şekilde tasarlanmış. İleri de network üzerinden eğitilmiş bir sisteme API bağlantısı yapılırsa gecikme bilgisi daha önemli hale gelebilir.

4.2 PEKİŞTİRMELİ ÖĞRENME ALTYAPISININ EKLENMESİ

Pekiştirmeli öğrenmenin oluşması için sıfırdan environment/çevre ortamı oluşturmam gerekti. Drone'nun öğreneceği haritalar modellenmiş şekilde simülasyonda mevcut bulunmaktadır.

Şekil 4.2a 4 Kanatlı Drone kırmızı kapıdan geçiş yaparken

Drone eğitim süresince PPO algoritmasında belirlenmiş adım sayısı kadar hareket almakta ve her hareketi sonucu 21 adet sensör verisi ve kamera görüntüsü(gerektiğinde) modele yüklenmektedir. Model bu sensör verileri

sonucunda x,y,z eksenlerinde şuan ki hızına eklenmesi gereken hız miktarını tahmin etmekte ve drone bu veriyle hareket etmektedir. Sinir ağından aldığı değerle hareketiğini sağladıktan sonra hesaplanan ödül fonksiyonuna göre bir ödül almakta ve olabildildiğince yüksek ödül çıkacak şekilde öğrenme süreci devam etmektedir.

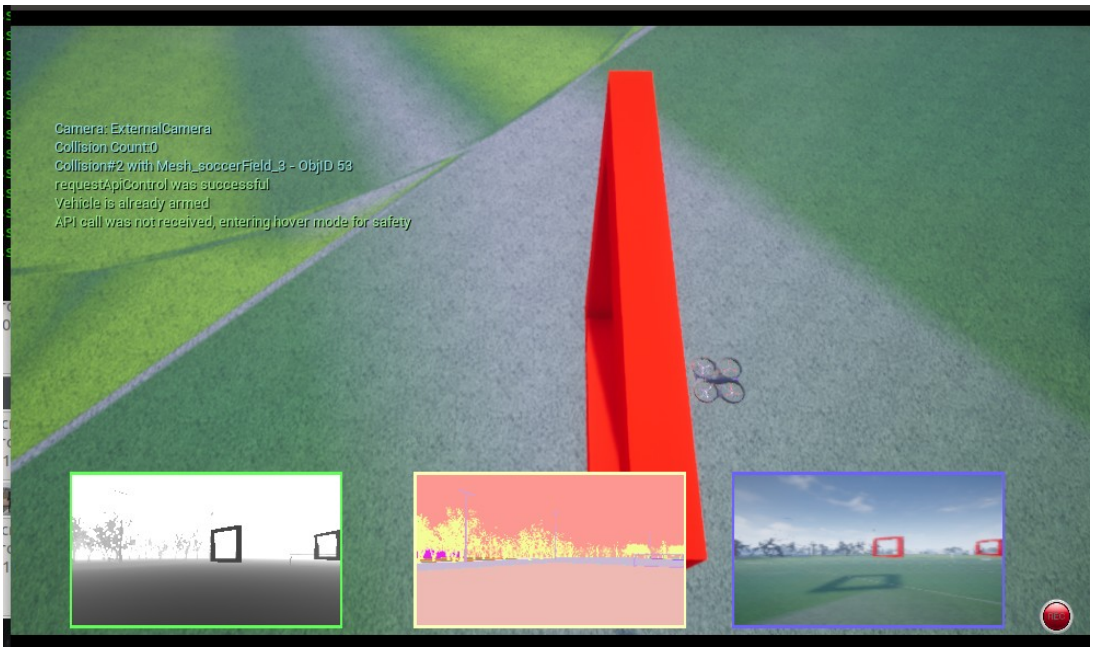
Drone'nun öğrenmesini sağlamak için 21 değişkenden oluşturulan durum/state değişkeni oluşturuldu. Durum değişkeni oluşturulurken robotik algoritmalarda ne gibi verilerin kullanıldığı incelendi ve pekiştirmeli öğrenme için olabildiğince çok veri yararlı olduğundan dolayı anlık verilerin hepsi eklendi.

Drone'nun aksiyon alması için gerekli olan yapı eklendi. X,Y,Z linear hız değişkenlerine [-1,+1] arasında eklemeler yapılması sağlandı. Drone'nun maksimum hız sınırı ve minimum hız sınırları belirlendi. (3 m/s 0.5 m/s) Bu hareket asenkron bir şekilde yapıldığı için threadler arasında ki haberleşmede gecikme olursa drone takılıp kalmaması sağlandı.

Drone için ceza/ödül fonksiyonu eklendi. Bu fonksiyon RL algoritmalarının temel taşı olmaktadır. Bunun tasarımında oldukça uzun süre düşünüldü. PPO , DQN algoritmalarının tersine çok kompleks ödül fonksiyonu yapıları istemiyor fakat bu demek değildir ki her durumda kolay şekilde converge/yakınlaşma sağlıyor. Burada bitirme projesi süresince oldukça çok değişikliğe gidildi. İlk başta kullanılan negatif ödüllerin sisteme kötü etki yaptığı farkedildi ve araştırma yazılarını okuduğumda aynı yapıyı tekrar gördüm.

Şuan ki ödül fonksiyonu yapısı şu şekilde çalışmaktadır:
Drone kendisine en yakın kapıyı bulmaktadır. (Kapı bilgileri simülasyondan elde dilebilmektedir). Kendisine en yakın kapıya veya bu kapıdan bir ileri ve bir geri kapılardan birine hareket etmiyorsa drone tamamen yanlış bir rota saptamaktadır ve bu sırada o an ki eğitimin resetlenmesi gerekmektedir. Sadece bu durumda negatif ödül almaktadır. Eğer drone gitmesi gereken kapı yerine bir sonraki veya bir önceki kapıya gidiyorsa negatif değer vermemekteyim. Bunun yerine eğitimi devam

ettirdiğimde etrafı gezme parametresi(merak parametresi- PPO makalesi) farklı aksiyonlar denemekte ve eğitim süresince doğru kapının yakınından geçmektedir. Herhangi bir seferde doğru kapıya doğru hareket yaptığında(mesafe azaldığında) yüksek bir başlangıç pozitif ödülü verilmektedir. Herhangi bir defa buradan geçtiğinde yüksek bir pozitif ödül almaktadır. Buda drone'nun öğrenme mekanizmasının (Policy) bu harekete yüksek önem vermesini sağlamaktadır. Drone doğru kapıya yaklaştıkça pozitif ödül miktarı artmaktadır. 2.5 metre yaklaşlaştığında kapıya yaklaşmış olduğunu anlayıp bu alana kadar aldığı ödüllerden daha yüksek ödül almaya başlamaktadır. Yaklaşma durumuna geldiğinde CNN üzerinden kapının yeri saptanmaktadır. Kapıya doğru yaklaşmaya devam etmekte içerisinden geçtiğinde çok büyük bir bonus ödül almaktadır. Bu sırada hızıyla drone çok yavaş hareket etmemesi için hızı normalize etmeye çalışıldı. Fakat hız ile ilgili değişimler çok fazla dışarı taşmasını ve doğru optimize olamamasını sağladığı için olabildiğince az değiştirmeye çalıştım.



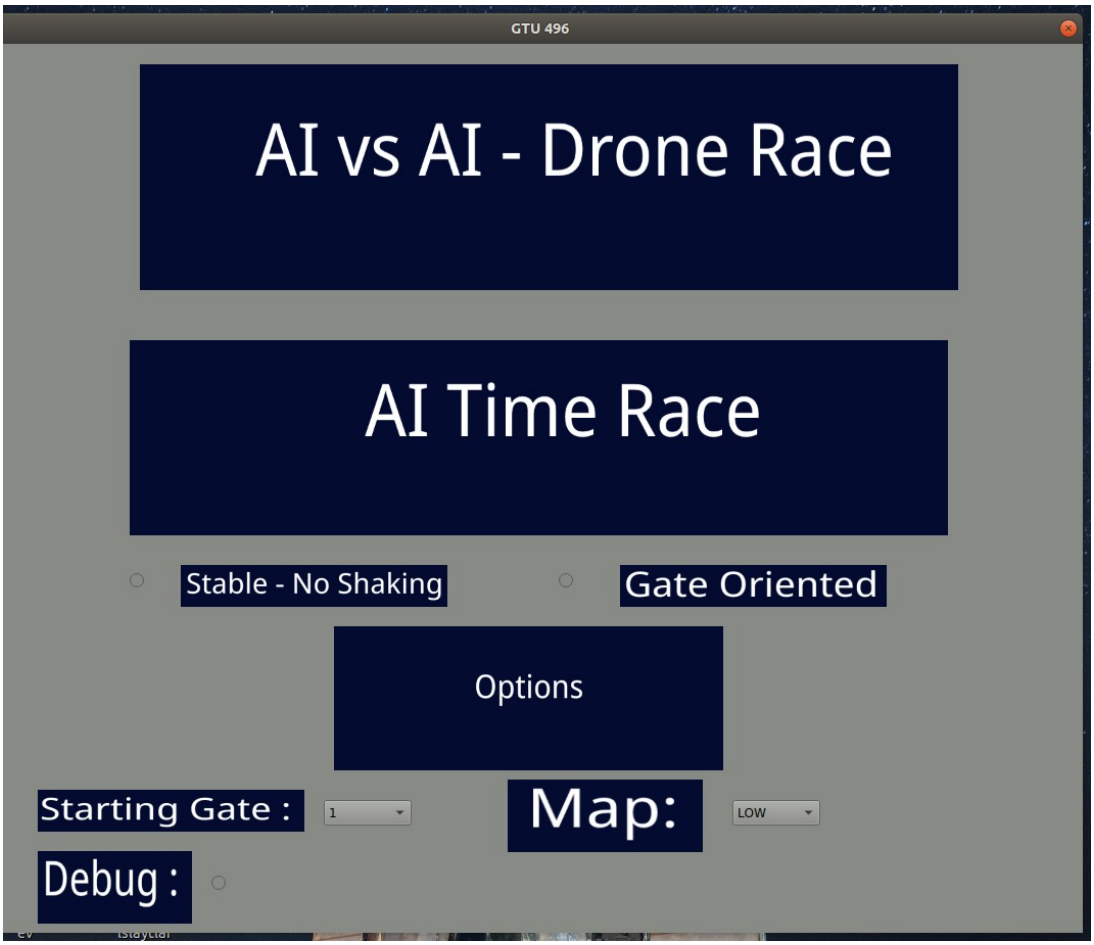
Şekil 4.2b Şekil 4.2a'da bulunan drone'nun kapıdan geçiş anı

Drone'un pekiştirmeli öğrenme ile deneyerek optimize rotayı bulması bizim ona otomatik olarak programladığımız kurallardan daha iyi çalışmaktadır. Bu sebepten

dolayı ödül fonksiyonu basit ama etkili tutulmalıdır. Negatif ödüller ise bir işe yaramamakla beraber modelin hesaplamalarında negatif etkiye sebep olmaktadırlar.

Drone'nun dışarı çıktığı durumlarda reset işlemini yapıcak yapı simülasyona gömüldü. Ayrıca her test aralığında test yapılması ve ağın eğitilmemesi(backward propogation işlemi yapılmaması) sağlandı. Drone o ana kadar öğrendiği olasılık matriksleri ile kararlar verip ne kadar ileri gidebildiği ölçüldü ve bunların sistematik tutulması için loglama sistemleri yapıldı.

Simülasyon için demo ortamı hazırlandı ve eğitilmiş modellerin farklı kararlar aldığı versiyonları üzerinden arayüz ile seçim yapılması sağlandı.



EĞİTİM SÜRECİ VE SONUÇLAR

5.1 EĞITIM ORTAMI

Projenin geliştirilme sürecinde Ubuntu 18.04 LTS(stabil sürüm) kullanıldı. Python 3.7 versiyonu seçildi , PyTorch yapay sinir ağları kütüphanesinin 1.3 versiyonu(2019) kullanıldı. Bunlar dışında OpenCV, numpy ve simülasyon verilerine ulaşmak için AirSim geliştirme ekibinin tasarladığı AirSim paketi kullanıldı.

Kişisel bilgisayarımda bulunan Intel i5 1.8 GHZ 2 çekirdek , 8GB RAM ve NVIDIA GT 840M (1800 mb VRAM) bulunan cihaz aynı anda yapay sinir ağını eğitmek ve simülasyonu çalıştırmakta zorlandı. Simülasyonda GPU'da çalıştığı ve yapay sinir ağı'da GPU'da çalıştığı için CUDA tabanlı sistem üzerinde hafıza yetersizliği hataları almaya başladım. Bu sebepten kullandığım tüm paketleri ve klasör dizinlerini bulut sistem üzerinde eğitmeye karar verildi. Bunu sağlamak için Docker yapısı oluşturdum. Bu docker yapısı istenildiği bulut servis sağlayacısına dosyaları indirip doğru dizinlere yerleştirebilmektedir.

Proje süresince iki farklı derin öğrenmeye odaklı bulut sağlayıcı kullandım.Vast.ai isimli şirketin kiralık sunduğu bulut bilgisayarlar ile başladığım eğitim sürecinde yüksek ücretlerden dolayı bırakmak zorunda kaldım ve Google Cloud yapısına geçtim . Burada aylık 300$ 'e kadar yeni üye kullanım hakkı veriliyor. İki yapıda da grafik görüntüsü alma imkanı yoktu. Bu tarz derin öğrenme sağlayıcıları genelde veri üzerinden çalıştıkları için grafik tabanlı kütüphaneler bulunmuyor ve kullanılan serverler üzerinde ki grafik kartları ekran görüntüsü vermeye göre tasarlanmamışlar. Ayı anda başka kişiler de sizin ekran kartınız üzerinde çalışabiliyor. Bu sebepte sonuçlar belirli bir yüksekliğe gelene kadar model dosyalarını indirip kendi bilgisayarımda eğitilmiş dosyaları test etme yöntemini seçtim. Bu yöntemin kötü yani çok efor gerektiriyor fakat alternatif çözüm bulamadığım için böyle yapmak durumunda bulundum. Eylül ayından beri eğitim sürecinde çalıştırmaktayım belirli aralıklarla. Tabi herhangi bir fonksiyona (ödül özellikle) yapılan değişiklikler modelin tamamen baştan eğitilmesini gerektiriyor.

Modelin 10 kapıdan geçecek seviyeye gelmesi için eğitilmesi gereken süre 18-22 saat civarında olmaktadır. Oluşturulan model dosyasıını tekrar kullanarak (çok büyük değişiklikler yapılmadıysa) transfer learning methodu uygulanabilmektedir. Bu şekilde modelin öğrendiği yapıyı yeni yapıda kullanmasını sağlamaktadır.

Eğitilen sistemlerde bulunan GPU'lar NVIDIA 1080 ve daha üst modeller olmaktadır. Düşük VRAM'li makinelerde eğitimin yapılması mümkün değildir. CUDA seviyesinin 5.0+ üzeri ve yeterli ekran hafızasına sahip olan makinelerde eğitim gerçekleşebilmektedir.

AMD GPU makinelerinde NVIDIA Cuda ve CudaNeuralNetwork yapıları bulunmadığı için derin öğrenme problemlerinde başarısız kalmaktadırlar ve herhangi bir CPU'dan fark gösterememektedirler.

5.2 BAŞARI KRİTERLERİ

- AirSim sisteminin pekiştirmeli öğrenme yapısına uyumluı hale gelmesi sağlanmıştır.
- Üretilen model ve eğitim süreci ile 4 kanatlı drone'nun drone yarışlarında kullanılan 10 kapılı pist üzerinde kapılara çarpmadan otonom turu tamamlaması mümkün hale getirilmiştir.
- Oluşturulan model klasik robotik algoritmalarla karşılaştırılmış ve aynı pist üzerinde birlikte sürülmeleri aracılığıyla pekiştirmeli öğrenmeli modelin daha stabil ve optimize şekilde gittiği görülmüştür.
- Kapılar arası uzaklık değiştirilerek modelin bu tarz şartlara adapte olabildiği gözlenmiştir.
- 10 kez durmaksızın tur attığında 10x10 kapı üzerinde çarpmadan geçebildiği gözlenmiştir.
- Diğer ortamlarda modelin eğitilmesi için gereken ortam sağlanmıştır.

6 SONUÇ

Proje kapsamında zor bir kontrol algoritması probleminin pekiştirmeli öğrenme ile ajana ortamla ilgili herhangi bir koşul sağlanmadan otonom sürüş yapmayı ve bunu yaparken stabil bir kontrol içinde bulunması öğretildi. Ayrıca öğrenmiş modelin farklı kapı aralıklarından çalıştırıldığında kendini ortama adapte edebildiği fark edildi.

Bu kapsamdan yola çıkarak zor kontrol problemlerinde , veri toplanması zor derin öğrenme problemlerinde ve gerçekten bulunulması mümkün olmayan fakat simüle edilebilen (Mars görevleri ve otonom araçların şehir içi trafikte sürülmesi gibi) ortamlarda pekiştirmeli öğrenme modellerinin başarı vaad ettiği ortada denilebilir.

Uzun eğitim süreleri gelişen donanımlar ve teknolojiyle azalması sağlanacaktır , çözümün diğer ortamlara genelleştirilmesi için ise aynı anda CPU veya GPU tabanlı eğitim süreçleri makalelerde konuşulmaktadır. Bu makalelerde farklı noktalardan başlayan farklı ajanların aynı modeli beslemesi sağlanmaktadır.

Bunun dışında başlangıçta düşünülen farklı ortam genellenme kısmı yapılmaya çalışıldı fakat eğitimin çok uzun sürmesinden ötürü istenildiği şekilde tamamlanmadı. Bunun yerine aynı ortam üzerinde genelleştirilme çözümü elde edildi.(Kapıların farklı aralıklarla başlanması ve farklı yüksekliklerde denenmesi)

7. KAYNAKÇA

[1] Deep Drone Racing , Zurich 2018. Intel Labs https://arxiv.org/pdf/1806.08548.pdf

[2] Reinforcement Learning and Dynamic Programming , Lucian Busoniu ,2010 https://orbi.uliege.be/bitstream/2268/27963/1/book-FA-RL-DP.pdf

[3] Proximal Policy Optimization Algorithms , 2017 John Schulman , OpenAI https://arxiv.org/pdf/1707.06347.pdf

[4] Trust Region Policy Optimization , 2015 John Schulman OpenAI https://arxiv.org/pdf/1707.06347.pdf

[5] AirSim simulation Microsoft GitHub https://microsoft.github.io/AirSim-NeurIPS2019-Drone-Racing/