T.C.

KOCAELİ ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

DERİN ÖĞRENMEDEKİ GELİŞMELER DERSİ

PROJESİ ÖDEVİ

**PEKİŞTİRMELİ SÜREKLİ ÖĞRENME**

Ödevi Hazırlayan:

215112006 Aslı YİĞİT

Dersin Öğretim Üyesi:

# Dr. Öğr. Üyesi Ayhan KÜÇÜKMANİSA

KOCAELİ, 2022

**1. Konu Tanıtımı**

**1.1 Özet**

Sürekli öğrenmenin temel amacı, öğrenilen görevlerin unutulmasının üstesinden gelmek ve yeni gelen görevlerde daha iyi performans veya daha hızlı yakınsama/eğitim hızı elde etmek için önceki bilgileri kullanmaktır. Bu çalışmada Pekiştirmeli Sürekli Öğrenme ile karmaşık bir şekilde tasarlanmış pekiştirmeli öğrenme stratejileri aracılığıyla gelecek her görev için en iyi sinir mimarisini arayan sürekli öğrenme için bir yaklaşım çözülecektir. MNIST ve CIFAR-100 veri kümelerinin varyantları için sıralı sınıflandırma ile çalışmanın derin ağlar için mevcut sürekli öğrenmenin nasıl performansı gösterilecektir.

Bu çalışmada yeni bir görevle karşı karşıya kalındığında, her katman için eklenecek en uygun düğüm/filtre sayısına karar vermek, bir kombinatoryal optimizasyon problemi olarak ortaya çıkar. Bu sorunu çözmek için gelişmiş bir şekilde tasarlanmış pekiştirmeli öğrenme yöntemi sunulur. Bu nedenle, Pekiştirmeli Sürekli Öğrenme (RCL) olarak adlandırılır. RCL'de, her görev için sinir ağlarının en iyi mimari hiper parametrelerini belirlemek için tekrarlayan bir sinir ağı olarak uygulanan bir denetleyici benimsenmiştir. Denetleyiciyi, hem doğrulama doğruluğundan hem de ağ karmaşıklığından kaynaklanan bir ödül sinyali tarafından yönlendirilen bir aktör-eleştirmen stratejisiyle eğitiyoruz. Bu, genel model karmaşıklığını azaltırken, eski görevlerde tahmin doğruluğunu mümkün olduğunca koruyabilir. Bildiğimiz kadarıyla, öneri, sürekli öğrenme problemlerini çözmek için pekiştirmeli öğrenmeyi kullanan ilk girişimdir.

RCL, yalnızca optimal olmayan ve hesaplama açısından pahalı olabilen yeni bir görevi çözmek için eski ağa sabit sayıda birim eklemekten [1] farklı olmakla kalmaz, aynı zamanda eklenen parametreler üzerinde grup seyrekliği düzenlemesi gerçekleştiren [2].'ten de farklıdır. Çeşitli sıralı görevlerde RCL'nin etkinliğini doğrularız. Ve sonuçlar, RCL'nin çok daha az birim ekleyerek bile mevcut yöntemlerden daha iyi performans elde edebileceğini gösteriyor.

2. **Derin Öğrenme**

Deep Learning (Derin öğrenme), yapay sinir ağlarının en önemli ve olgun modelidir. Yapay sinir ağları ile ilgili ilk çalışmalar 1943 yılına kadar uzanır (Kleene, 1956) ve yapay sinir ağlarının bir prototipi olan algılayıcı (perceptron) F. Rosenblatt tarafından 1957’de oluşturulmuştur (Rosenblatt, 1958). 1970’lerde (Werbos & John) hakkında bahsedilmeye başlanan geri yayılım (backpropagation) öğrenme algoritması 1980’den (Lecun, Bengio, & Hinton, Deep learning, 2015) itibaren sinir algoritmalarına uygulanarak büyük etki sağlamıştır. Araştırmacılar son yıllarda gelişen donanım teknolojisiyle daha derin sinir ağlarını eğitebilecek çalışmalar yapmaya başlamışlardır. Donanım alanındaki yenilikler bu çalışmaların hızla yayılmasına hizmet etmiştir [4].

Derin Öğrenme, verilen bir veri seti ile sonuçları tahmin eden birden fazla katmandan oluşan bir makine öğrenme yöntemidir. Derin öğrenme, makine öğrenmesi ve yapay zekâ birbirinden farklı anlamları olan terimlerdir. Derin öğrenme, makine öğrenmesinin; makine öğrenmesi ise yapay zekânın alt dalı olarak özetlenebilir.



Şekil-1: Derin Öğrenme Şeması

Derin Öğrenme algoritmalarının son zamanlarda görüntülerin ve sesin analizinde başarılı olduğu bildirildi. Bu algoritmalar, özellikle Convolutional Neural Network (CNN), tıbbi görüntüleme teknolojileri tarafından üretilen görüntülerde de son derece umut verici olduklarını kanıtladılar. Araştırmacılar, derin öğrenme algoritmalarını kullanarak, bu alanda görüntü sınıflandırma, nesne ve lezyon tespiti ve tıbbi bir görüntüdeki farklı dokuların segmentasyonu dahil olmak üzere birçok görevi yerine getirmişlerdir. Araştırmacılar, Derin Öğrenme yaklaşımından yararlanmak için çoğunlukla nöronlar, retina, akciğerler, dijital patoloji, meme, kalp, karın ve iskelet sisteminin tıbbi görüntülerine odaklandılar. Bu çalışma, ilgili literatürün genel bir resmini sunmak için tıbbi görüntüler üzerinde Derin Öğrenme algoritmalarını kullanan son yıllardaki literatür çalışmalarını gözden geçirmektedir.

Derin öğrenme denetimli, yarı denetimli veya denetimsiz olarak gerçekleştirilebilir. Derin öğrenmede çok sayıda veri girişiyle ayırt edici özellikleri kendisi öğrenir. Öğrenme işlemini için ne kadar çok veri girişi olursa o kadar çok başarılı olunur. Veriler birden çok katmandan geçer. Üst katmanlar daha çok ayrıntı çıkaran katmanlardır.

Denetimli öğrenme yönteminde, algoritmaya girdi verilir ve beklenen sonuç verilir. Algoritma sonucunda elde edilen sonuç ve beklenen sonuç algoritma tarafından kıyaslanıp hesaplama parametrelerinin yeniden hesaplanması sağlanır. 14 Denetimsiz öğrenme, beklenen çıktı algoritmaya verilmeden, girdi olarak verilen bilgilerin kümelerini kullanan yöntemdir. Bu yöntem özellikle büyük verilerin kümelenmesinde kullanılmaktadır.

Makine öğrenmesi günlük hayatımızın her noktasında, bazılarımız farkında olmasa bile kullanılmaktadır. Nesne tanıma, konuşmayı yazıya çeviren yazılımlar, kullanıcıların ilgi alanlarına ve geçmişte inceledikleri ürünleri baz alarak ürün önerileri yapan elektronik ticaret siteleri bu alanlara verilebilecek en temel örneklerdir.

Üç ana derin model çeşidi vardır:

- Çok katmanlı Perceptron (Multilayer Perceptrons)

- Evrişimsel Sinir Ağı (Convolutional Neural Networks)

- Yinelgeli Sinir Ağı (Recurrent Neural Networks)

**Derin öğrenmenin kullanım alanları:**

- Yüz tanıma sistemlerinde

- Ses tanıma sistemlerinde

- Araçlardaoto pilot özelliğinde veya sürücüsüz araçlarda kullanılmaktadır.

- Alarm sistemlerinde, kamera kayıtlarını sürekli kontrol etmek yerine, yalnızca olağan dışı hareketlerde alarm sisteminin devreye girmesi gibi teknolojiler derin öğrenme sayesinde mümkün olmaktadır.

- Sağlık sektöründe kanser araştırmalarında, zaman kaybını ortadan kaldırmaktadır. Kanserli hücre örneklerinin tanıtıldığı derin öğrenme algoritmaları, yeni hücrelerin kanserli olup olmadığı tanısını koymakta hem daha hızlı hem de daha başarılı oluyor.

- Görüntü iyileştirilmesinde

- Tavsiye sistemlerinde, beğenilebilecek müzik ve film önerileri sunmada

- Siber tehdit analizlerinde de derin öğrenme yöntemleri geliştirilebilir. Yukarıda verilen örneklerin dışında birçok örnek verilebilir.

**3. Pekiştirmeli Öğrenme**

Pekiştirmeli öğrenme, doğal öğrenme süreçlerine benzeyen öğrenme karakteristiği ve çevredeki sürekli değişimi ile öğrenme ihtiyacı olduğundan diğer makine öğrenmesi yaklaşımlarına göre daha fazla tercih edilmektedir [5]. Pekiştirmeli öğrenme modeli canlılara benzer bir şekilde çalışır. Örneğin, bir bebek yürümeyi öğrenirken defalarca kez düşer, her düşüşünde biraz daha deneyim kazanır ve en sonunda yürümeye başlar [6]⁠. Bir çocuğun bisiklet sürmeyi öğrenmesi gibi doğal bazı süreçlerde öğrenme olgusu öğreticisiz öğrenme şeklindedir ve istenen değer için hâli hazırda bir bilgi yoktur. Bu çocuk pekiştirmeli öğrenmede ajan (agent) adı verilen öğrenen makinedir [5]. Ajanın etkileşimde bulunduğu herşey çevre (environment) olarak nitelendirilir. İçerisinde bulunduğu çevreyi bilmez deneme yanılma yolu ile farklı algoritmalar kullanarak yeni şeyler öğrenmeye çalışır, karşılaştığı durumlara bir tepki verir. Ajana tanımlı olduğu çevrede bazı görevler, geri bildirimler verilir ve bu çevrede uygulayacağı hareketler incelenip ajana ödül veya ceza verilir. Kazanılan ödül maksimize edilir ve ajanında ödül kazanabilmek için uygun hareketlerde bulunması gerekir [7]⁠. Ajanın bulunduğu çevrede kazanacağı ödül veya cezalar, gitmesi istenen yerler daha önceden kullanıcı, programcı tarafından belirlenir ve ödül tablosuna yazılır. Ajan kazandığı ödül veya cezalar ile bulunduğu çevrede nereye gidip gitmeyeceğini öğrenmiş olacak ve belirli bir tecrübe edinmektedir.

Pekiştirmeli Öğrenme sisteminde ajan ve çevre dışında biri opsiyonel olmak üzere dört kavram bulunur. Bunlar:

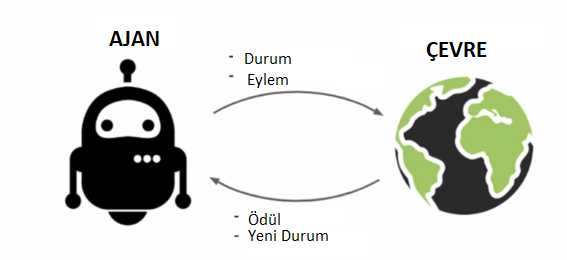
**Politika (Policy):**  Ajan çevreye göre bir eylem ortaya koyar, buna politika denir ve çevreden buna karşılık bir geri bildirim bekler. Gerçekleşen geri bildirimler önceden belirlenmiş ödül sistemine tabi tutulur. Kazanılan ya da kaybedilen ödül doğrultusunda ajan eğitilir, yaptığı eylemin ne kadar yanlış ya da doğru olduğunu anlar [6].⁠

**Ödül (Reward):** Ajanın bulunduğu durumda aldığı aksiyonlar sonucu kazanacağı/kaybedeceği ödül miktarıdır.

**Değer (Value):**Ajanın başarılı olabilmesi adına içinde bulunduğu durumun ne kadar değerli olduğudur.

**Ortam Modeli:** İsteğe bağlı olarak sisteme dahil edilen bir ögedir. Ortamın bir simülasyonu olup ajanın bir hareket gerçekleştirmeden önce bu hareket sonucunda alabileceği ödülü ve durumu tahmin etmesini sağlamaktadır. Bu sayede bir planlama yapılarak ajanın davranışında değişiklik meydana gelebilecektir.

Ajanın odanın içindeki hareketler eylem (action) olarak tanımlanır. Örneğin hareket kümesinde ajan için aşağı, yukarı, sağ, sol hareketleri tanımlansın. Ajanın hareket kümesi içerisinde bir yöne gitmesi eylemdir. Ajanın eylemlerine bağlı olarak verilen ödül veya cezaya ödül (reward) denir. Ajan çevre içerisinde belirli durumda bulunur ve hareket eder. Ortamda bunun karşılığında ajana ödül verir. Ortamdan okunan bilgiler ise gözlem (observation) olarak adlandırılır. Şekil 2’de durum, eylem, ödül kavramları gösterilmiştir.



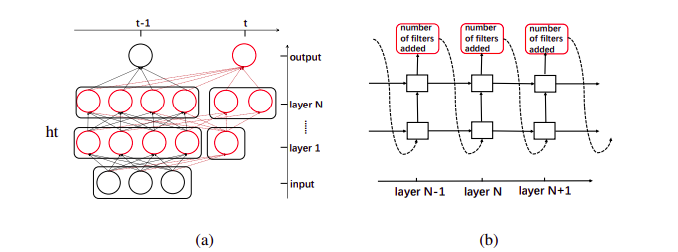
Şekil 2: Ajan-Çevre Etkileşimi

Her adımda, bir etmen ortamın mevcut durumunu st gözlemler, bir eyleme π(at|st) göre karar verir ve bir ödül sinyali rt+1 gözlemler. Aracının amacı, indirimli ödüllerin beklenen toplamını Rt, Rt = P∞ t 0=t+1 γ t 0−t−1 rt 0 maksimize eden bir politika bulmaktır, burada γ ∈ (0, 1] gelecekteki ödüllerin önemini belirleyen indirim faktörü Bir politikanın π değer fonksiyonu, beklenen getiri Vπ(s) = Eπ[ P∞ t=0 γ t rt+1|s0 = s] ve eylem değeri olarak tanımlanır. Qπ(s, a) = Eπ[ P∞ t=0 γ t rt+1|s0 = s, a0 = a] olarak işlev görür Politika gradyan yöntemleri, bir θ ile parametrelenen πθ(a|s) parametreli belirli bir stokastik politikalar ailesi üzerindeki performans hedefi. Politika gradyan teoremi, θ'ye göre ortalama ödül ve indirimli ödül hedeflerinin gradyanı için ifadeler sağlar.

**4. Pekiştirmeli Sürekli Öğrenme**

Sürekli öğrenme veya yaşam boyu öğrenme [3], daha önce eğitilmiş görevlerin nasıl gerçekleştirileceğini unutmadan ardışık görevleri öğrenme yeteneği, yapay zeka geliştirmek için önemli bir konudur. Sürekli öğrenmenin birincil amacı, öğrenilen görevlerin unutulmasının üstesinden gelmek ve yeni gelen görevlerde daha iyi performans veya daha hızlı yakınsama/eğitim hızı elde etmek için önceki bilgileri kullanmaktır. Derin öğrenme topluluğunda, önceden eğitilmiş görevleri unutma sorununu hafifletmek için, öğrenme sırasında ağ mimarisinin değişip değişmediğine göre ayırt edilen iki grup strateji geliştirilmiştir.

RCL (Reinforced Continual Learning), denetleyici, değer ağı ve görev ağı olmak üzere üç ağdan oluşur. Denetleyici, ilkeler oluşturmak ve her görev için kaç filtre veya düğüm ekleneceğini belirlemek için Uzun Kısa Süreli Bellek ağı (LSTM) olarak uygulanır.



Şekil 3: (a) RCL, t-th görevi geldiğinde ağın her katmanını uyarlamalı olarak genişletir. (b) Yeni görev için kaç tane filtre ekleneceğini belirlemek için bir RNN olarak uygulanan kontrolör.

Değer ağını, durumun değerine yaklaşan tam bağlantılı bir ağ olarak tasarlıyoruz. Görev ağı, görüntü sınıflandırma veya nesne algılama gibi belirli bir görevi çözmek için ilgi duyulan herhangi bir ağ olabilir. Bu çalışmada, yöntemimiz yalnızca evrişimli ağlara değil, aynı zamanda tam bağlantılı ağlara da uyum sağlayabilmesine rağmen, RCL'nin unutmayı önlemek için bu CNN'yi nasıl uyarlanabilir bir şekilde genişlettiğini göstermek için görev ağı olarak bir evrişimli ağ (CNN) kullanıyoruz.

Şekil 3(a), yeni bir görev geldiğinde RCL'nin ağı nasıl genişlettiğini görsel olarak gösterir. Görev t − 1'in öğrenme süreci bitip t görevi geldikten sonra, her katmana kaç tane filtre veya düğüm eklenmesi gerektiğine karar vermek için bir denetleyici kullanırız. Anlamsal sapmayı önlemek için, önceki görevler için ağ ağırlıklarının değiştirilmesini durdurur ve yalnızca yeni eklenen filtreleri eğitiriz. Modeli t görevi için eğittikten sonra, neden olan anlamsal sapmayı önlemek için yeni eklenen her filtreye her katmanın şekline göre zaman damgası koyarız. Çıkarım süresi boyunca, her görev yalnızca t aşamasında tanıtılan parametreleri kullanır ve sonraki görevlere eklenen yeni filtreleri dikkate almaz.

Görev ağının m katmanı olduğunu varsayalım, yeni gelen bir görevle karşı karşıya kaldığında, her i katmanı için 0 ile ni − 1 aralığında eklenecek filtre sayısını belirtiriz. m katmanları için filtreler, eylemlerin tüm kombinatoryal kombinasyonlarını geçmektir. Bununla birlikte, bir m-katmanlı ağ için, en iyi eylem kombinasyonunu toplamanın zaman karmaşıklığı, NP-zor olan ve VGG ve ResNet gibi çok derin mimariler için kabul edilemez olan O(Qm 1 ni)'dir.

Bu sorunla başa çıkmak için, bir dizi eylemi sabit uzunlukta bir dize olarak ele alıyoruz. Her katmana kaç filtre eklenmesi gerektiğini gösteren böyle bir dize oluşturmak için bir denetleyici kullanmak mümkündür. Ardışık katmanlar arasında tekrar eden bir ilişki olduğundan, kontrolör doğal olarak bir LSTM ağı olarak tasarlanabilir. İlk adımda, kontrolör ağı, eğitim sırasında sabitlenecek olan mevcut görev için girdi olarak (yani durum s) boş bir yerleştirme alır. Her t görevi için, ağı, katman i için her eylemi örnekleme olasılıklarını, yani eklenecek filtre sayısını temsil eden softmax çıktısı, pt,i ∈ R ni ile donatıyoruz. LSTM'yi otoregresif bir şekilde tasarlıyoruz, Şekil 3(b)'de gösterildiği gibi, önceki adımdaki olasılık pt,i sonraki adıma girdi olarak beslenir. Bu süreç, tüm m katmanları için eylemler ve olasılıklar elde edilene kadar dolaştırılır. Ve a1:m eylem dizisinin politika olasılığı, çarpım kuralına uyar.

### **5. Veri Seti**

Projede MNIST ve CIFAR-100 veri setleri kullanılmıştır.

**5.1 MNIST:**

MNIST el yazısı rakamları veritabanı, 60.000 örneklik bir eğitim setine ve 10.000 örneklik bir test setine sahiptir. NIST tarafından sağlanan daha büyük bir kümenin alt kümesidir. Ön işleme ve biçimlendirme için minimum çaba harcarken gerçek dünya verileri üzerinde öğrenme teknikleri ve örüntü tanıma yöntemlerini denemek isteyenler için iyi bir veritabanıdır. Rakamlar boyut normalleştirildi ve sabit boyutlu bir görüntüde ortalandı.

Aşağıda verilen dosyalardan proje yapım aşamasında gerekli olan dosyalar kullanılmıştır.

train-images-idx3-ubyte.gz : eğitim seti görüntüleri (9912422 bayt)

train-labels-idx1-ubyte.gz : eğitim seti etiketleri (28881 bayt)

t10k-images-idx3-ubyte.gz : test seti görüntüleri (1648877 bayt)

t10k-labels-idx1-ubyte.gz : test seti etiketleri (4542 bayt)

**5.2 CIFAR-10:**

CIFAR-10 veri seti, sınıf başına 6000 görüntü olmak üzere 10 sınıfta 60000 32x32 renkli görüntüden oluşur. 50000 eğitim görüntüsü ve 10000 test görüntüsü vardır.

Veri seti, her biri 10000 görüntü içeren beş eğitim grubuna ve bir test grubuna bölünmüştür. Test grubu, her sınıftan tam olarak rastgele seçilmiş 1000 görüntü içerir. Eğitim grupları rastgele sırada kalan görüntüleri içerir, ancak bazı eğitim grupları bir sınıftan diğerinden daha fazla görüntü içerebilir. Bunların arasında, eğitim grupları her sınıftan tam olarak 5000 görüntü içerir.

Aşağıda verilen dosyalardan proje yapım aşamasında gerekli olan dosyalar kullanılmıştır.

CIFAR-10 piton versiyonu 163 MB

CIFAR-10 Matlab versiyonu 175 MB

CIFAR-10 ikili versiyon (C programları için uygundur) 162 MB

**6. Kullanılan Teknolojiler**

Projede Python programlama dili ve pythonda bulunan numpy, tensorflow ve matplotlib kütüphaneleri ve **pickle** modülü kullanıldı. Numpy dizilerini oluşturmak için **numpy** modülünü, derin sinir ağlarının eğitimi ve çıkarımında **tensorflow** kütüphanesi kullanıldı.

#### **Gereksinimler**

#### python= 2.7.15

#### numpy

#### tensorflow==1.2.0

**6.1 Numpy**

NumPy, dizilerle çalışmak için kullanılan bir Python kitaplığıdır.Doğrusal cebir, fourier dönüşümü ve matrisler alanında çalışmak için işlevlere sahiptir. NumPy, [Python programlama diline](https://teknoloji.org/python-nedir/) ait çok boyutlu dizilerle ve matrislerle çalışmamıza yardım eden ileri düzey matematiksel işlemler yapabileceğimiz bir kütüphanedir.

**6.2 Tensorflow**

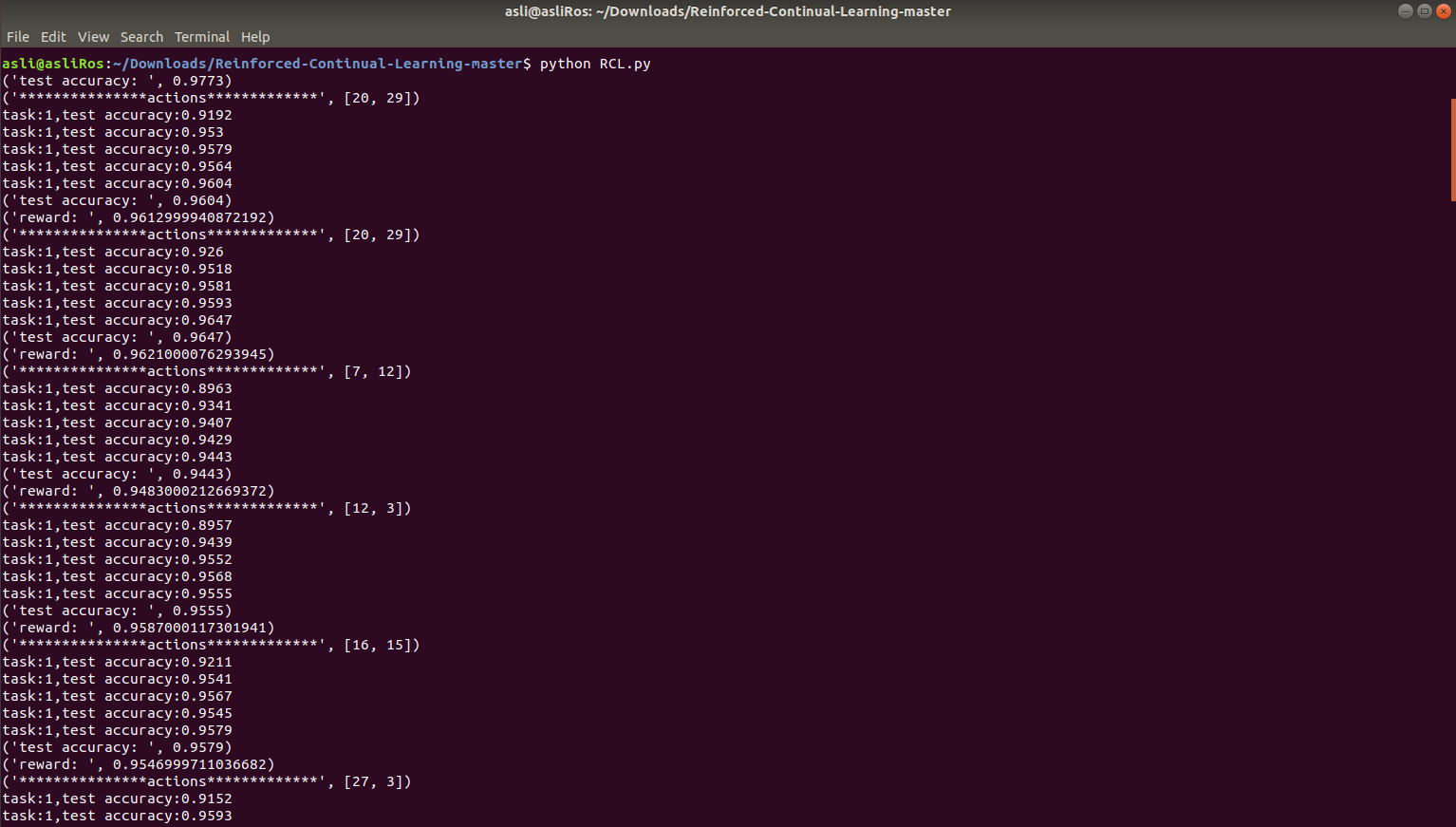
Açık kaynak kodlu bir derin öğrenme kütüphanesidir. Esnek yapısı sayesinde, tek bir API ile platform farketmeksizin hesaplamaları, bir veya birden fazla CPU, GPU kullanarak deploy etmenize olanak sağlar. Temelinde Python kullanılarak geliştirilen bu framework, günümüzde Python‘ın yanısıra C++, Java, C#, Javascript ve R gibi birçok dili desteklemektedir.

**6.3 Pickle**

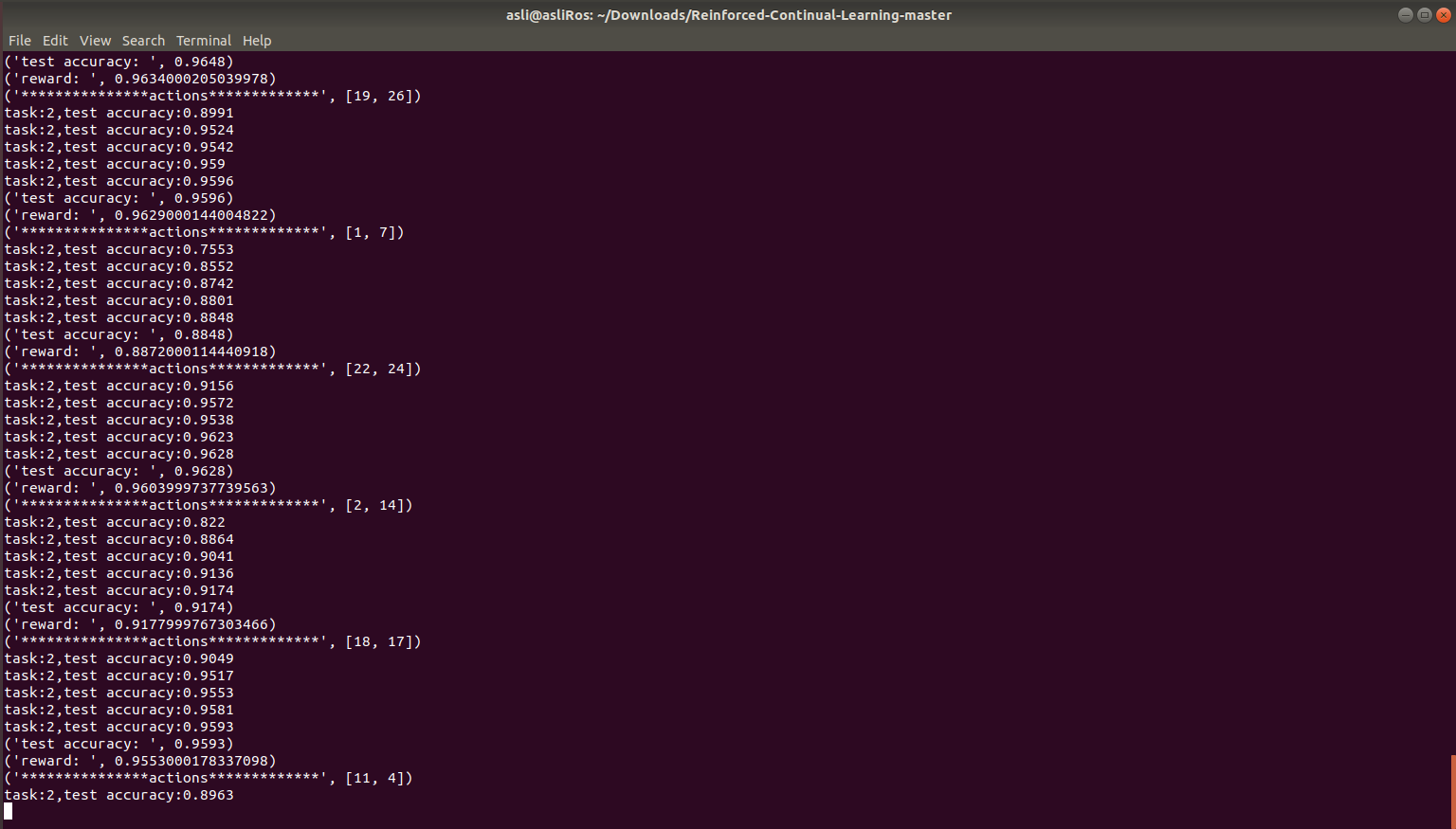
Uygulamanın oluşturduğu verileri dosyaya kaydedip, tekrardan kullanmayı sağlar. Kaydetme işlemi text formatında değil , **binary** formatında olur.

**7. Sonuçlar**

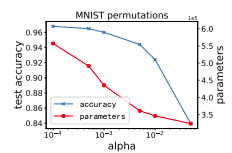
Veri setlerinin tümü için öğrenilecek görev sayısını T = 10 olarak belirlendi. MNIST veri setleri için her görev, 10 farklı sınıftan 60000 eğitim örneği ve 1000 test örneği içeriyor. CIFAR-100 veri kümeleri için her görev, 10 farklı sınıftan 5000 örnek ve 1000 örnek içerir. Model, görevleri tek tek gözlemler ve görev bir kez gözlemledikten sonra, eğitim sırasında daha sonra görev gözlenmez. Şekil 4 ve Şekil 5’te sonuçlar verilmiştir.

****

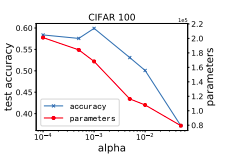
Şekil 4: Sonuç-1

****Şekil 5: Sonuç-2

Model performansı ve karmaşıklık arasındaki dengeyi, ödül fonksiyonundaki (6) katsayı aracılığıyla kontrol ederiz. Grafiklerde değişen α'nın test doğruluğunu ve model parametrelerinin sayısını nasıl etkilediğini göstermektedir. Beklendiği gibi, artan α ile model karmaşıklığı önemli ölçüde düşerken model performansı da yavaş yavaş bozulur. İlginç bir şekilde, α küçük olduğunda, parametre sayısındaki azalmaya kıyasla doğruluk çok daha yavaş düşer. Bu gözlem, orta ölçekli bir ağın hala nispeten iyi bir model performansı elde edebilmesi için uygun bir α seçilmesine yardımcı olabilir. Şekil 6’da test doğruluğu ve model karmaşıklığı arasındaki denge MNIST ile sonucu verilmiş olup Şekil 7’de ise test doğruluğu ve model karmaşıklığı arasındaki denge CIFAR 100 ile sonucu verilmiştir.

****

Şekil 6: Test doğruluğu ve model karmaşıklığı arasındaki denge MNIST ile sonucu



Şekil 7: Test doğruluğu ve model karmaşıklığı arasındaki denge CIFAR 100 ile sonucu

**Kaynakça**

[1] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. arXiv preprint arXiv:1606.04671, 2016.

[2] J. Yoon and E. Yang. Lifelong learning with dynamically expandable networks. arXiv preprint arXiv:1708.01547, 2017

[3] Sebastian Thrun. A lifelong learning perspective for mobile robot control. In International Conference on Intelligent Robots and Systems, 1995

[4] Z. Aslan, On The Use Of Deep Learning Methods On Medical Images, 2018

[5] Elif Degirmenci, “Fabrika Ortamındaki Otonom Taşıyıcı Araçlar için Durumsal Farkındalık Yöntemi Geliştirilmesi,” 2019.

[6] B. Ünlü, “DERİN PEKİŞTİRMELİ ÖĞRENME ALGORİTMALARIN İNCELENMESİ,” 2020.

[7] H. KAYAKÖKÜ, “PEKİŞTİRMELİ ÖĞRENME TABANLI ROBOTLAR İLE YENİ BİR ROBOCODE SAVAŞ STRATEJİSİ,” 2019.

[8] James Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. Proceedings of the National Academy of Sciences, 114(13):3521–3526, 2017.

[9] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. icarl: Incremental classifier and representation learning. In CVPR, pages 5533–5542. IEEE Computer Society, 2017.

[10] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, 1998.