T.C.
BEYKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
BİLGİSAYAR MÜHENDİSLİĞİ BİLİM DALI

DERİN ÖĞRENME İLE RAKAM ÖĞRETME

Yüksek Lisans Tezi

Tezi Hazırlayan :

Ali KAYA

İstanbul, 2017

T.C.
BEYKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
BİLGİSAYAR MÜHENDİSLİĞİ BİLİM DALI

DERİN ÖĞRENME İLE RAKAM ÖĞRETME

Yüksek Lisans Tezi

Tezi Hazırlayan:

Ali KAYA

Öğrenci No:

160820803

Danışman:

Yrd. Doç. Dr. Ediz ŞAYKOL

İstanbul, 2017

YEMİN METNİ

Yüksek lisans tezi olarak sunduğum Derin Öğrenme ile Rakam Öğretme başlıklı bu çalışmanın, bilimsel ahlâk ve geleneklere uygun şekilde tarafımdan yazıldığını, yararlandığım eserlerin tamamının kaynaklarda gösterildiğini ve çalışmamın içinde kullanıldıkları her yerde bunlara atıf yapıldığını belirtir ve bunu onurumla doğrularım.11.05.2017

Ali KAYA

T.C.
BEYKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZ SAVUNMA SINAVI SONUÇ TUTANAĞI

Beykent Üniversitesi
Fen Bilimleri Enstitüsü Müdürlüğü'ne,

Aşağıda tez adı belirtilen yüksek lisans öğrencisi 160820803. no'lu ALİ KAYA'in 11/05/2017 tarihinde yapılan tez savunma sınavı[1] sonucunda 45 dakika süreyle sunduğu ve savunduğu tezi hakkında[2] oybirliğiyle KABUL kararı verilmiştir.

Bilgilerinize saygılarımızla arz ederiz.

Anabilim Dalı : BİLGİSAYAR MÜHENDİSLİĞİ
Programı : BİLGİSAYAR MÜHENDİSLİĞİ
Tez Başlığı[3] : DERİN ÖĞRENME İLE KAKAM ÖĞRETME

Tez Sınav Jürisi		**Öğretim Üyesi**	**İmza**
Danışman	:	Yrd. Doç. Dr. Ediz ŞAYKOL	
Üye	:	Doç. Dr. Gökhan SİLAHTAROĞLU	
Üye	:	Yrd. Doç. Dr. Atılay YILMAZ	

---

[1] Jüri üyeleri söz konusu tezin kendilerine teslim edildiği tarihten itibaren en geç bir ay içinde toplanarak öğrenciyi tez savunma sınavına alır. Belirlenen günde yapılamayan jüri toplantısı, katılanların hazırladığı bir tutanakla enstitü yönetimine bildirilir. Bu durumda jüri en geç onbeş gün içinde toplanarak adayı tez savunma sınavına alır. Tez savunma sınav süresi en az 45 dakikadır. Yüksek lisans tez savunma sınavı, tez çalışmasının sunulması ve bunu izleyen soru-yanıt bölümlerinden oluşur ve dinleyiciye açıktır. (Beykent Lisansüstü eğitim ve Öğretim Yönetmeliği-Madde30-3)

[2] Tez sınavının tamamlanmasından sonra jüri, tez hakkında "kabul", "düzeltme" veya "red" kararı verir. Jüri başkanı, jüri üyelerince imzalanmış sınav tutanağını, tez sınavını izleyen üç gün içinde ilgili enstitü yönetimine teslim eder. Tezi başarısız bulunan öğrencinin Enstitü ile ilişiği kesilir. Tezi hakkında düzeltme kararı verilen öğrenci en geç üç ay içinde gerekli düzeltmeleri yaparak ve yönetmelikte belirtilen usullere uygun olarak tezini aynı jüri önünde yeniden savunur. Bu savunma sınavında da tezi kabul edilmeyen öğrencinin enstitü ile ilişiği kesilir.(Beykent Lisansüstü eğitim ve Öğretim Yönetmeliği-Madde30-4)

[3] İleride doğabilecek aksaklıkların engellenmesi için tezin başlığının yazılması gerekmektedir.

TEŞEKKÜR

Bu günlere gelmemi sağlayan aileme, çalışmamın her safhasında gayretini ve desteklerini esirgemeyen danışmanım Sayın Yrd. Doç. Dr. Ediz ŞAYKOL'a teşekkür ederim.

Adı ve Soyadı : Ali KAYA
Danışmanı : Yrd. Doç. Dr. Ediz ŞAYKOL
Türü ve Tarihi : Yüksek Lisans /Tez 2017
Alanı : Bilgisayar Mühendisliği
Anahtar Kelimeler : Derin Öğrenme, Caffe, Python, Mobil Teknolojiler, Rakam Öğrenme

ÖZET

DERİN ÖĞRENME İLE RAKAM ÖĞRETME

Bu tez çalışmasında günümüz popüler yapay zekâ felsefesi olan derin öğrenme ile anaokulu öğrencilerine rakam öğretme amaçlanmıştır. Proje kapsamında Açık Kaynak Lisanslı Caffe Derin Öğrenme alt yapısı, Flask uygulama alt yapısı, Python ve Objective-C programlama dilleri, Web Servis ve mobil uygulama teknolojileri kullanılarak rakam öğretme uygulaması gerçeklenmiştir. Uygulama mobil platformlar (iOS işletim sistemi ile çalışan telefon ve tabletler) üzerinde öğrenci kullanımına sunulacak ve Python tabanlı web servis ile derin öğrenme sağlanacaktır.

Name and Surname : Ali KAYA
Supervisor : Assist. Prof. Dr. Ediz ŞAYKOL
DegreeandDate : Master /Thesis 2017
Major : Computer Engineering
KeyWords : Deep Learning, Caffe, Python, Mobile Technologies, Number Learning

ABSTRACT

NUMBER TEACH WITH DEEP LEARNING

In this thesis study, it was aimed to teach the number of kindergarten students with deep learning which is today's popular artificial intelligence philosophy. Within the scope of the project, numerical teaching application was realized by using Open Source Licensed Caffe Deep Learning Framework, Flask application framework, Python and Objective-C programming languages, Web Service and mobile application technologies. The application will be made available for student use on mobile platforms (phones and tablets running the iOS operating system) and deep learning will be provided through Python-based web service.

İÇİNDEKİLER

ŞEKİL LİSTESİ

Sayfa No.

KISALTMALAR

AWS	:	Amazon Web Services
GA	:	Genetik Algoritmalar
CAFFE	:	Convolutional Architecture for Fast Feature Embedding
BVLC	:	Berkeley Vision and Learning Center
ROC	:	Receiver Operating Characteristics Curve

1. GİRİŞ

Günümüzün büyük yazılım firmalarından Google, Microsoft, Facebook, Amazon, IBM gibi yazılım devleri ar-ge çalışmalarının büyük bir kısmını derin öğrenme üzerine yapmaktadır. Buradaki amaç, yazılımların insanlar gibi öğrenmesini sağlamaktır. Yakın geçmişte Google kendi geliştirmiş olduğu derin öğrenme alt yapısı TensorFlow ile Go oyununu hiç kod yazmadan makineye öğretmiş ve Go oyunu şampiyonunu yenmeyi başarmıştır. Bu yapay zekâ alanında son yıllarda gerçekleşen çok büyük bir devrimdir.

Bu bağlamda büyük firmalar her geçen gün yeni ürünlerini piyasaya sürmektedir. Facebook videolara ve resimlere derin öğrenme ile resim filtreleri uygulayarak gerçek zamanlı video ve resim dönüşümleri yapmaktadır. Tüm bunları 0.3 ms altında yaparak insanların göz kırpmasından daha kısa bir sürede filtre uygulayarak değişimi göz açıp kapamadan daha hızlı yaptıklarını iddia etmektedir. Büyük firmalar sadece resim işleme ya da makinelere oyun öğretme amacı ile bu felsefeyi kullanmamaktadır. Pinterest, IBM gibi firmalar Sahne Yorumlama, Sınıflandırma gibi ürünler geliştirmektedirler.

Derin öğrenme ile artık, resimdeki köpektir, kedidir, insandır gibi sonuçlar yerine, resimde "Motor süren bir insan var.", "Dağların arasından güneş doğuyor..", "Arka planında gökdelenler olan şehir manzarası." gibi sonuçlar vermektedir. Tüm bunların vardığı teknolojik nokta sürücüsüz arabalar, röntgenleri yorumlayıp kanser teşhisi yapan makineler, bu tez çalışmasının da konusu olan, eğitimde yardımcı makineler geliştirmektedir. Tüm bu bilgiler ışığında, derin öğrenme tüm yazılım dünyası tarafından geleceğin teknolojisi olarak görülmektedir. Derin öğrenme, yapay zekânın kaybettiği popülerliği tekrar kazandırmış ve üzerinde yoğun çalışmalar yapılmasını sağlamıştır.

Bu tez çalışması altı bölümden oluşmaktadır. İkinci bölümde bu tez çalışmasının konusu olan derin öğrenme teknolojisinin tarihsel gelişiminin yanı sıra, öğrenme, yapay zekâ ve derin öğrenmenin tanıtılması ve çalışma alanları hakkında detaylı bilgi verilmektedir. Üçüncü bölümde bir derin öğrenme alt yapısı olan Caffe hakkında detaylı bilgiler verilmektedir. Dördüncü bölümde geliştirilen sistemin tüm parçaları hakkında bilgiler verilecektir. Bu parçalar

başlıca mobil uygulamalar, web projesi ve servis projesidir. Projeler hakkında bilgi verilmeden önce kullanılan teknolojiler hakkında kısa bilgilendirmeler yapılacaktır. Beşinci bölümde geliştirilen uygulamanın kullanım alanı anlatılmakta ve örnekler üzerinden rakam öğretmesi yapılacaktır. Son bölüm olan altıncı bölümde ise projenin amaçları ve bu amaçların hangisinin tamamlandığı, elde edilen başarı yüzdeleri ile birlikte proje sonucu açıklanacaktır.

2. YAPAY ZEKÂ VE ÖĞRENME

2.1 Giriş

Bu bölümde yapay zekâ için kullanılan teknolojiler hakkında genel bilgiler verilecek ve bu teknolojilerin gelişiminden bahsedilmektedir. Yapay zekânın tam anlaşılabilir olması için başlangıçtan günümüze kadar olan gelişimini hakkında bilgi verilmektedir. Bu yüzden insan öğrenmesinden başlayarak, makinelerin öğrenmesi ve en son derin öğrenme anlatılacaktır.

Derin öğrenmenin temelini oluşturan yapay zekânın tarihsel gelişimi yanı sıra günümüze kadarki farklı uygulanış şekilleri ve teknikleri hakkında bilgi verilmektedir. Derin öğrenmenin tam olarak anlaşılabilmesi için bu konular hakkında bilgi sahibi olmak önem arz etmektedir.

2.2 Öğrenme Nedir?

Öğrenme belli durumlar ve sorunlar karşısında tepki ve davranış oluşturma, gerektiğinde bunları değiştirip yenilerini edinebilme yeteneğidir. Yaşantı sonucu davranışta meydana gelen nispeten sürekli değişikliktir, çevreye uyum sürecidir. Bu bakımdan, ihtiyaçları daha iyi karşılayacak biçimde düzene koyma ya da yeni bir durum karşısında bunları yeniden örgütleme anlamına gelir. Davranışta öğrenme sonucu meydana gelen değişmeleri, olgunlaşmanın etkilerinden ve geçici fizyolojik değişmelerden ayırt etmek gerekir. Organizmanın içinde var olan yeteneklerin kendiliğinden gelişmesine ve varabileceği düzeye varmalarına ‘olgunlaşma’ denmektedir. Olgunlaşma, organizmanın temelindeki potansiyel güçlerin göreve hazır bir duruma ulaşmasıdır.

Öğrenme aktif bir oluşumdur, yaşantılar sonucu meydana gelir. Öğrenmeyi bireyin kendi tepkileri, etkinlikleri ve yaşantıları yoluyla çevresine uyum tarzını değiştiren davranışlar geliştirmesi veya davranışlarının farklılaşması olarak tanımlayabiliriz. Bireyin çok fazla

tekrarlanmış uyarıcılara karşı alışma sonucu duyarlılığını kaybetmesi şeklindeki bir davranış değişikliğine öğrenme diyemeyiz.

2.3 Yapay Zekâ

Yapay zekâ konusundaki ilk çalışmalar ise McCulloch ve Pitts tarafından yapılmıştır. 2000'lerin başından beri Derin Öğrenme, yapay zekâ alanında çığır açmış ve yaygınlaşmaya başlamıştır. Yapay zekâ alanında yeni bir teknoloji olan derin öğrenme algoritması makineye görüntü ve ses tanımada farklı bir boyut kazandırmaktadır. İnsanlar için nesneleri tanımak ve onları belli bir kategoriye dâhil etmek oldukça basittir. Mesela bir insan elma ve armudun farkını ayırt edebilir ve bu iki nesneyi sınıflandırabilir. Ancak makineler için durum daha farklıdır. Derin öğrenmenin kilit noktası bir nesnenin gösterimindeki farklı katmanlardır.

Yapay zeka günümüz teknolojisinde gerek uygulama alanları gerek ise düşünebilen cihazlar oluşturabilmek açısından oldukça önemli bir bilim dalıdır. İnsan gibi karar verebilen modeller üretebilme düşüncesi insanlar için çok eskiye dayanan bir durumdur. Zaman içerisinde değişik düşünce yapıları ile insan beynini modelleme fikri uygulamaya konulmuştur. Yapay zeka, insan beynini, düşünce yapısını, öğrenme, karar verme gibi yeteneklerini taklit ederek makineler üzerinde modellemeyi sağlamaktadır. Yapay zeka mantığı üzerinde birçok farklı kavramlardan söz edilebilinmektedir. Zeka, klasik mantık, sezgisellik, algoritma, optimizasyon, öğrenme gibi kavramlar bunlara örnek olabilmektedir. Yapay zekanın ne olduğunu ortaya koymak için öncelikle bu kavramların ne olduğunun ortaya konması gerekmektedir. Daha sonrasında ise yapay zekanın temeli anlaşılabilir hale gelir. Bunun nedeni yapay zekanın temelinin insan beynini taklit etmesidir.[1]

İnsanlar için basit bir olay gibi gözüken bu fark, yapay zekâ geliştirmelerinde çok büyük farklılıklar oluşturmaktadır. Bu farklılıkları anlamamız için bu alanın tamamına zekâdan başlayarak geniş bir perspektifte bakmak önemlidir.

Zekâ araştırmaları derin, geniş, çok boyutlu, çok disiplinli bir alandır. Günümüzde insan zekâsının taklit edilmesinden henüz çok uzakta olduğumuzu söyleyebiliriz. Ancak zekânın değişik şekilleri ve boyutları, değişik yapay zekâ teknolojileri tarafından taklit edilebilmektedir. Son yıllarda bu teknolojiler kullanılarak yeni melez teknolojiler elde edilmekte ve bir teknolojinin eksikliği bir diğer teknolojinin üstünlükleri ile kapatılmaktadır. Bu gelişmeler bize daha yetenekli ve üstün özellikli yapay zekâ teknolojileri sunmakta, insan zekâsına biraz daha yaklaşılmaktadır. Yine de, insan zekâsının henüz uzağında olduğumuz ve atılacak çok adım olduğunu bilinen bir gerçektir. [2]

Bu bölümde öncelikle yapay zekâ teknolojilerinden uzman sistemler, bulanık mantık ve genetik algoritma ele alınacaktır. Aşağıda her teknik ayrı ayrı anlatılmıştır.

2.3.1 Uzman Sistemler

Bir uzman sistem, insan uzmanın bilgi birikimine sahip ve normal şartlarda ancak insan uzman tarafından çözülebilecek problemleri çözmekte kullanılan bilgisayar sistemleridir. [2]

Yapay zekâ çalışmaları ilk günden itibaren insan beynini taklit etme üzerine kuruludur. Günümüzde bu taklit için derin öğrenme, yapay sinir ağları gibi yapılar kullanılmaktadır fakat yapay zekânın ilk dönemlerinde bu tekniklerden söz etmek mümkün değildir. İlk zamanlarda direk taklit yerine bilgiye dayalı sistemler geliştirilmiştir. Bilgiye dayalı geliştirilen ve alanında uzman ve bilgili kişileri taklit eden ilk sistemler uzman sistemlerdir.

Uzman sistemlerin kökeni, uzman sistem kavramıyla ters bir şekilde "Genel Amaçlı Problem Çözücü" adlı ilk yapay zekâ projelerinden birine dayanmaktadır. Ancak her ne kadar Genel Amaçlı Problem Çözücü kendisinden bekleneni verememiş olsa da, bu çalışma uzman sistemlerin doğuşu için gerekli alt yapıyı hazırlamıştır. Genel Amaçlı Problem Çözücünün geliştirilmesi çalışmaları sırasında insan bilincinin çok sayıda "Eğer...ise.." şeklindeki kuralla modellenebileceği anlaşılmıştır. Genel Amaçlı Problem Çözücü, genel amaçlı olmasına rağmen

elde edilen teknoloji daha dar bir alanda kullanılarak uzman sistemlerin ortaya çıkmasını sağlamıştır. [2]

Uzman sistemlerin bilinen ilk uygulaması, 1960'lar da geliştirilmeye başlanan ve ancak 1970'lerin başında faal olarak kullanılmaya başlanan Dendral adlı uzman sistemdir. Joshua Lederberg (Kimya alanında Nobel ödülü almıştır.), Edward Feigenbaum ve Bruce Buchanan tarafından geliştirilen Dendral, bileşimi bilinmeyen maddelerin moleküler yapılarının tanınması konusunda uzmandır. Bugün hala kullanılan bu uzman sistemin, bazen insan uzmanlardan daha iyi olduğu bilinmektedir.[3]

2.3.2. Bulanık Mantık

Bulanık mantık sistemlerinden bahsetmeden önce klasik mantığın yapısını anlamak bu bölümün anlaşılması için önem arz etmektedir. Klasik mantık ve bulanık mantık birbirlerinden farklı gözükseler de temelde aynı işlev üzerine kurulmuşlardır.

Klasik mantık, cevabın ya "doğru" ya "yanlış" olduğu durumlardaki çalışmalardır. Çalışma alanı keskin cevaplar vermek üzerine kuruludur.

Klasik önermeler ve dereceli önermeler arasındaki temel fark, dereceli önermelerin doğruluk değerlerinin [0,1] aralığında olmasıdır. Her bir klasik önerme doğru yada yanlış olmak zorundadır. Fakat, dereceli önermelerin doğruluğu yada yanlışlığı, [0,1] aralığında bir sayı ile ifade edilen bir doğruluk derecesiyle belirtilir. [4]

2.3.3. Genetik Algoritmalar

Genetik Algoritmalar biyolojideki evrimin bir parçası olan mutasyonun bilgisayar programlamasına uyarlanmasıdır. En iyi sonucu bulmak için programımız mutasyona uğrar ve yeni nesiller oluşturur. Kodumuz en iyi sonuca ulaşana kadar evrimleşecek ve mutasyona uğrayacaktır.

Genetik algoritmalar, en iyinin korunumu ve doğal seçilim ilkesinin benzetim yoluyla bilgisayarlara uygulanması ile elde edilen bir arama yöntemidir. Standart bir GA'da, aday sonuçlar eşit boyutlu vektörler olarak ifade edilir. Başlangıçta, bu vektörlerden bir grup, rastlantısal olarak seçilerek belirli büyüklükte bir popülasyon oluşturulur. Kromozom adı verilen bu vektörler, yeni nesiller oluşturarak değişikliklere uğrar. Bir kromozomun üzerindeki genler, n boyutlu vektörlerin bir boyutuna karşılık gelmektedir. [5]

Genetik algoritmalarda tüm bu mutasyon, yeniden üretim, çaprazlama işlemlerini gerçekleştiren operatörler bulunur. Her olay için özel tipte geliştirilmiş operatörlerdir. Bu operatör, her yerni nesilde kromozomların iyiliğini ölçer. Çıkan sonuca göre bazı kromozomlar değiştirilerek yeniden üretilir.

2.4. Derin Öğrenme

Derin öğrenme yapay sinir ağları üzerinde gerçekleştirilen öğrenme teknikleri felsefesidir. Makine öğrenmesinden farklı olarak derin öğrenmede çok katmanlı yapay sinir ağları kullanılmaktadır. Derin öğrenmeyi başarılı kılan da bu çok katmanlı yapay sinir ağları teknikleridir. Derin öğrenmede yapay sinir ağları tıpkı beynimizdeki gibi sinyal aktarımı yaparak çalışır. Bu sinyal aktarımları beyin hücrelerimizdeki eşik değerlerin simüle edilmesi ile gerçekleştirilir. Girişten alınan değerler ağa verilir ve bu değerler belli başlı işlemlerden sonra hala eşik değerini aşabiliyorsa bir sonraki nörona aktarım yapılır. Derin öğrenmede tıpkı beynimizdeki gibi bu şekilde çalışır.

Tüm bunları sağlayan çok katmanlı yapay sinir ağları temelde üç bölümden oluşur. Giriş katmanı, gizli katmanlar, çıkış katmanı. Öğrenmeyi sağlayan tüm karmaşık işlemler ve öğrenme algoritmaları gizli katmanlarda uygulanır. Giriş ve çıkış katmanlarında herhangi bir işlem yapılmaz. Bu katmanların yapay zekâ alanında yaptığı en büyük devrim, bir ve sıfırdan başka cevaplar üretebilmektir. Derin öğrenmeden önce makine öğrenmesi dediğimiz yapay zekâda, sonuç her zaman ya bir ya da sıfır olmuştur. Derin öğrenme ile 0.4, 0.6 gibi sonuçlarda üretilebilmektedir. Bunun bizlere sağladığı kazanç bir problemi daha özel ele alıp daha detaylı çözümleyebilmektir. Örneğin bir tatile gitme problemimizin olduğunu varsayalım. Makine öğrenmesinde ki cevaplar "Tatile gidilir.", "Tatile gidilmez." şeklinde olacaktır. Derin öğrenme ile buradaki cevaplar, "Vize alınırsa tatile gidilir.", "Havalar kötü olursa tatile gidilmez." şeklinde daha detaylı ve problemin çözümüne daha yakın ve özel yaklaşımlar olacaktır.

Bu anlattıklarımızın ışığında, yapay zekâdaki bu gelişim büyük firmalar tarafından çok büyük projeler ile özelleştiriliyor. Google'dan Tobias Weyand ve ekibi derin öğrenme ile görsel üzerinden konum bilgisi verecek şekilde bir ağı eğitiyorlar. Konum saptaması tamamen görsel üzerinde bulunan pikseller üzerinden yapılıyor. Dünya yirmi altı bin farklı parçaya bölünüyor ve bölgelerin önemine göre veriler ağa veriliyor. Ayırt edici veriler bulunduramayan okyanus ve kutup bölgeleri bu çalışmanın dışında tutulmuş. Ekip 90 milyon görselden oluşan bir veri ağı oluşturarak ağı tamamlıyor. PlaNet ismini verdikleri makineye 2.3 milyon fotoğraf gösteriliyor. Şu an itibari ile PlaNet rastgele seçilen bu fotoğraflarda %10 oranla şehir, %28 oranla ülkeyi doğru bir şekilde tespit ediyor. [6]

Yapay zekânın kendisine öğretilen bilgiye odaklanması ve tüm işlem gücünü buna harcaması ileride yapılacak farklı uygulamalarla iş dünyasında yepyeni kapıları aralayabilir. IBM'in yapay zekâ programı Watson'ın kitapları okuyarak analiz yapabilmesi gibi gerçeklerden söz ederken bunun olmaması için bir neden de yok. Geçtiğimiz aylarda Watson veri ağını kullanarak Harry Potter serisinin kitaplarını okumuş, filmlerini izlemiş ve bu eylemlerinin sonucunda filmler ve kitapları karşılaştıran bir karakter analizi yapmıştı. Üstelik ortaya döktüğü veriler hikayeye konu olan kahramanların özellikleriyle örtüşüyordu.[6]

3. CAFFE

3.1. Giriş

Caffe ifade, hız ve modülerlik dikkate alınarak geliştirilmiş bir derin öğrenme alt yapısıdır. Berkeley Vizyon ve Öğrenme Merkezi (BVLC) ve açık kaynak kod topluluğuna katkıda bulunan kişiler tarafından geliştirilmiştir.

Yangqing Jia, UC Berkeley'deki doktorası sırasında geliştirmiş ve başlangıcını oluşturmuştur. Bugüne kadar 1.000 yakın kişi tarafından katkıda bulunulmuştur. [3]

3.2. Faydaları

Caffe, tüm bu görevler için belgelenmiş örneklerle, eğitim, test etme, ince ayar yapma ve modelleri dağıtma için eksiksiz bir araç seti sunar. Bu nedenle, en son teknolojiye sahip makine öğrenimine atlamak isteyen araştırmacılar ve diğer geliştiriciler için ideal bir başlangıç noktasıdır. Aynı zamanda, muhtemelen bu algoritmaların mevcutta en hızlı uygulanması olduğu için, endüstriyel dağıtım için daha yararlı olmasını sağlıyor. Bunların dışında öne çıkan diğer özellikleri aşağıda incelenmiştir.

Anlamlı mimari: Uygulama ve yeniliği teşvik eder. Modeller ve optimizasyon, sabit kodlama olmadan yapılandırma ile tanımlanır. Bir GPU makinesinde eğitim yapmak için tek bir bayrak ayarladıktan sonra sunuculara veya mobil cihazlara yerleştirilebilir. İsteğe bağlı olarak daha sonra CPU ve GPU arasında geçiş yapmanıza olanak sağlar.

Genişletilebilir kod: Aktif gelişimi teşvik eder. Caffe ilk yılında, 1.000'in üzerinde geliştirici tarafından fork edilmiş ve çok önemli değişiklikler yüklenmiştir. Bu katkı sağlayıcılar sayesinde alt yapı, hem kod hem de modellerde en gelişmiş teknolojiyi yakından takip ederek sürekli güncel kalmaktadır.

Hız: Caffe'yi araştırma deneyleri ve endüstride kullanım için mükemmel kılar. Caffe, Şekil 1. de gösterilen tek bir *NVIDIA K40 GPU* ile günlük 60 milyondan fazla görüntü işleyebilmektedir. Bu da ortalama görüntü için 1 ms/görüntü, öğrenme için 4 ms/görüntü demektir. Berkeley geliştiricileri Caffe'nin mevcut en hızlı convnet uygulaması olduğuna inanmaktadır.

Şekil.1. NVIDIA K40 GPU

Topluluk: Caffe, akademik araştırma projelerine, başlangıç prototiplerine, hatta görme, konuşma ve multimedya alanındaki büyük ölçekli endüstriyel uygulamalara güç vermektedir.

Modülerlik: Yazılım baştan itibaren yeni veri formatlarına, nets katmanlarına ve loss fonksiyonlarına kolaylıkla genişletilebilecek şekilde modüler olabilmesi için tasarlanmıştır. Çok sayıda katman ve loss fonksiyonu zaten uygulanmıştır ve bunların çeşitli görevler için nasıl eğitileceği, tanıma sistemleri içine nasıl dahil edildiğini bol örnekler ile gösterilmiştir.

Sunum ve uygulama ayrımı: Caffe modeli tanımları, Protocol Buffer dilini kullanarak yapılandırma dosyaları olarak yazılmaktadır. Caffe, net mimarilerini yönlendirilmiş düz graph'lar biçiminde desteklemektedir. Örnekleme sırasında, Caffe net için gereken miktarda

bellek tutar ve ana bilgisayar veya GPU'yu alttaki konumundan soyutlamaktadır. CPU ve GPU uygulaması arasında geçiş yapmak sadece konfigüratif bir işlem olmaktadır.

Test kapsamı: Caffe'deki her modül bir teste sahiptir ve o modüle karşılık gelen testler olmadan projeye yeni bir kod kabul edilmemektedir. Bu, kod tabanının hızlı iyileştirilmesine ve yeniden yapılandırılmasına olanak tanımaktadır. Böylece kodu kullanan araştırmacılar daha güvenilir hissetmektedir.

Python ve MATLAB Bağlantı Noktaları: Hızlı prototip yapma ve mevcut araştırma koduyla ara birim (interface) kurmak için Caffe, Python ve MATLAB bağlantı noktaları sağlamaktadır. Ağları oluşturmak ve girdileri sınıflandırmak için her iki dil de kullanılabilmektedir. Python bağlantı noktası, yeni eğitim prosedürlerinin kolay prototipi için çözücü (solver) modülünü de ortaya çıkarmaktadır.

Eğitimli referans modeller: Caffe, daha önce eğitilmiş modeller üzerinde çalışma olanağı sağlamaktadır. Böylece akademik çalışmalar tüm dünya üzerinde ortak modeller üzerinde yürütülebilmektedir. Aynı şekilde bir çalışmaya başlayan kişi her zaman sıfırdan başlamak zorunda kalmamaktadır. Var olan çalışmaları daha ileriye taşıyabilmekte ve bunu başlangıçta zaman kaybı olmadan daha hızlı yapmaktadır. Bu hız sayesinde bu alandaki ilerlemeler daha hızlı olacaktır. Model paylaşımı şu an için akademik çalışmalar açık kaynak lisanslı değildir. Üzerinde çalışmak için açık kaynak lisanslı modeller bulunmalıdır.

3.3. Bileşenleri

Caffe'nin bileşenlerinin listesi ve kısa açıklamaları aşağıda verilmiştir.

- Nets (Ağlar), Layers (Katmanlar) ve Blobs (Bloblar): Caffe modelinin anatomisini oluşturur.
- Forward (İleri) / Backward (Geri): Katmanlı kompozisyon modellerinin temel hesaplamalarını oluştururlar.
- Loss (Kaybetme): Öğrenilecek görev loss tarafından tanımlanır.
- Solver (Çözücü): Solver, model optimizasyonunu koordine eder.
- Layer Catalogue (Katman Kataloğu): Katman, modelleme ve hesaplamanın temel birimidir.
- Interfaces (Arayüzler): Python, Matlab, Command Line
- Data (Veri): Model girişi için caffe yöntemleri.

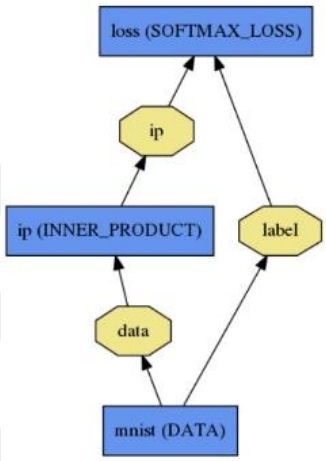
3.3.1. Nets

Net, katmanların toplanması ve birleştirilmesi olarak görülebilir. Caffe, kendi model şemasında net katman tanımlar. Net, giriş verisinden kaybolma noktasına kadar tüm modeli alttan üste tanımlar. Caffe veri ve türevleri net üzerinden ileri ve geriye doğru akarken, bilgiyi blob olarak depolar, iletir ve yönlendirir.

Net, bileşimi ve oto-türevlendirmeyi kullanarak bir fonksiyonu ve gradyanı ortaklaşa tanımlar. Her katman çıktısının bileşimi belirli bir görevi yerine getirme işlevini hesaplar. Katmanın bileşeni backward, görevi öğrenmek için loss üzerinden gradyanı hesaplar. Caffe modelleri uçtan uca makine öğrenme motorlarıdır. Net, bir hesaplama graph'ı ile bağlı katmanlar dizisidir (Tam olarak yönlendirilmiş bir düz graph (DAG)). Caffe, forward ve backward geçişlerin doğruluğundan emin olmak için katmanların herhangi bir DAG'ının tüm

hesaplamasını yapar. Tipik bir net, diskten yüklenen sınıflandırma veya yeniden yapılandırma gibi bir görevi hesaplayan bir kayıp katmanı ile biten bir veri katmanı ile başlar.

Net, katmanlar kümesi ve bağlantılarını düz metin modelleme dilinde tanımlar. Basit bir lojistik regresyon sınıflandırıcısı aşağıdaki gibidir:



Şekil.2. Lojistik Regresyon Sınıflandırıcısı

Yukarıdaki net'i tanımlamak için aşağıdaki Caffe dosyası yazılır:

```
name: "LogReg"
layer {
 name: "mnist"
 type: "Data"
 top: "data"
 top: "label"
 data_param {
  source: "input_leveldb"
  batch_size: 64
 }
}
layer {
 name: "ip"
 type: "InnerProduct"
 bottom: "data"
 top: "ip"
 inner_product_param {
  num_output: 2
 }
}
layer {
 name: "loss"
 type: "SoftmaxWithLoss"
 bottom: "ip"
 bottom: "label"
 top: "loss"
}
```

Yukarıdaki tanımlama kod tarafında bir takım alt yapı fonksiyonlarının çağırımı ile gerçekleşir. Alt yapıda bu kod çağırımı Net::Init() tarafından gerçekleştirilir. Başlatma, esas olarak iki işlem yapar: DAG'ın Bloblarını ve Katmanlarını (Layers) oluşturmak ve katmanlar (Layers) için Setup() fonksiyonunu çağırmak. Oluşturulan Blob ve Katmanlar LifeCycle boyunca bellek üzerinde ve class'lar üzerinde saklanır. Ayrıca genel net mimarisinin doğrulanması gibi bir dizi başka kayıt tutma işlemi de yapar.

Net yapımının cihaz bağımsız olduğunu unutmamalıyız. (Daha önceki açıklamalarımızda, Blobs ve katmanların (layers) uygulama ayrıntılarını model tanımından gizlediğini hatırlayalım.). Net yapımından sonra net'in ne üzerinde çalışacağı ile ilgili GPU veya CPU seçimi yapılır. Seçim Caffe::mode() ile yapılır. Katmanlar, aynı sonuçları üreten ilgili CPU ve GPU rutinleri ile birlikte gelir. Böylelikle CPU / GPU anahtarı modelden bağımsız olarak sorunsuz ve bağımsızdır. Tüm bu özellikleri ile araştırma ve uygulama için hem modeli hem de uygulamayı bölmede Caffe en iyisidir.

3.3.2. Layers

Katman, bir sonraki modelin ve hesaplamanın temeli olarak geliyor. Bir Caffe katmanı bir sinir ağı katmanının özüdür. Giriş olarak bir veya daha fazla blob alır ve çıktı olarak bir veya daha fazla blob üretir.

Katman bir modelin özü ve hesaplamanın temel birimidir. Katmanlar filtreleri evriştirir, iç ürünleri alır, düzeltilmiş doğrusal ve sigmoid gibi doğrusal olmayan uygulamaları ve diğer elemanlı dönüşümleri uygular, normalleştirir, verileri yükler ve softmax gibi kayıpları hesaplar. Tüm işlemler için katman kataloğu incelenebilir.

Bir katman, alt bağlantılardan girdi alır ve çıktıyı üst bağlantılardan geçirir. Her bir katman türü, üç kritik hesaplama tanımlar:

- Setup (Kur): Modül başlatıldığında katmanı ve bağlantılarını bir kez başlatır.
- Forward (İleri): Alttan verilen girdi çıktıyı hesaplar ve üste gönderir.
- Backward (Geriye): Gradyan verildiğinde, üst çıktı, girişe göre gradyanı hesaplar ve alta gönderir. Parametreli bir katman, parametrelere olan gradyanı hesaplar ve dahili olarak depolar.

Katmanların bu yapısından dolayı özel katmanları geliştirmek, net'in bileşimi ve kodun modülerliği sayesinde kolayca gerçekleştirilebilir. Katman için kurulumu ileriye ve geriye doğru tanımladıktan sonra katman, bir net'e dahil edilmeye hazırdır.

3.3.3. Blobs

Blob, alt yapı için standart dizi ve birleşik bellek ara birimidir. Blob'un ayrıntıları, bilgilerin katmanlar ve nets (ağlar) içinde nasıl depolandığı ve iletildiğini açıklar.

3.3.3.1. Blob Storage and Communication

Bir Blob, işlenmekte olan ve Caffe tarafından geçilen gerçek verinin üzerindeki bir sarmalayıcıdır. Ayrıca işlemci(CPU) ve GPU arasında senkronizasyon sağlar. Matematiksel olarak blob, C-contiguous biçimde saklanan N boyutlu bir dizidir. Caffe, blob'ları kullanarak verileri depolar ve iletir. Bloblar, optimizasyon için görüntü yığınları, model parametreleri ve türevleri gibi veri tutan birleşik bir bellek ara birimi sağlar.

Bloblar, CPU ana bilgisayarından GPU cihazına gerektiğinde senkronize ederek karışık CPU / GPU işleminin hesaplama ve zihinsel yükünü otomatik olarak azaltır. Ana bilgisayarda ve cihazdaki hafıza, verimli bellek kullanımı için gerektiğinde kullanılır.

Görüntü verilerinin yığınları için geleneksel "blob" boyutları, N x channel K x height H x width W dir. (N x kanal K x yükseklik Y x genişlik G)

Blob bellek düzende sıra büyüklüğündedir, bu nedenle son/en sağdaki boyut en hızlı değişir. Örneğin, 4D blob'da, (n, k, h, w) indeksteki değer fiziksel olarak ((n * K + k) * H + h) * W + w indekste bulunur.

- *Sayı / N*, verilerin toplu boyutudur. Toplu işleme, iletişim ve aygıt işleme için daha iyi sonuç elde eder. 256 resimden oluşan ImageNet eğitim topluluğu için N = 256.
- *Kanal / K*, özellik boyutu. Örneğin; RGB görüntüleri için K = 3.

Parametre blob boyutları katmanın türüne ve yapılandırmasına göre değişir. Örnek olarak 11 x 11 boyutlu ve 3 girişli, 96 filtreli convolution bir katman için blob boyutu 96 x 3 x 11 x 11 dir. Bir başka örnekte 1000 çıktı kanalı ve 1024 giriş kanalı olan bir iç ürün / tamamen bağlı katman için blob parametresi boyutu 1000 x 1024'tür. Özel veriler için veri girişi için gerekli veri katmanı giriş araçlarını kendiniz yapmanız gerekebilir. Veri bir defa tanımladıktan sonra geri kalan işlemleri yine Caffe otomatik olarak yapacaktır.

3.3.3.2. Implementation Details

Blob gradyanlar kadar değerlerle de ilgilendiğinden bellekte, veri ve farklılık yığınını saklar. Birinci yığın normal verilerdir. İkincisi yığın net tarafından hesaplanan gradyanlardır. Ayrıca, gerçek değerler CPU'da ve GPU'da depolanabileceğinden, bunlara erişmenin değerleri değiştirmeyen const yolu ve değerleri değiştiren değiştirilebilir yol olarak iki farklı yolu vardır.

```
const Dtype* cpu_data() const;
Dtype* mutable_cpu_data();
```

Bu tasarımın nedeni, bir Blob senkronizasyon ayrıntılarını gizlemek, veri aktarımını en aza indirgemek ve CPU ile GPU arasındaki değerleri senkronize etmek için bir SyncedMem sınıfı kullanmasıdır. Temel kurallar, değerlerin değiştirilmesi istenmiyorsa daima const çağrısı kullanılmalı ve pointer'lar asla kendi nesnelerimizde saklanmamalı. Bir blob üzerinde çalışırken her pointer'i almak için pointer'i çağırmak gerekir çünkü SyncedMem, verileri ne zaman kopyalayacağımızı anlamamız için buna ihtiyaç duyacaktır.

Pratikte GPU'lar mevcut olduğunda, birisi diskten CPU kodunda blob'a veri yükler, GPU hesaplamasını yapmak için bir aygıt çekirdeği çağırır ve blob'u bir sonraki katmana taşır. Düşük düzeyli ayrıntıları göz ardı ederken yüksek performans seviyesini korur. Tüm katmanların GPU uygulamaları (implements) olduğu sürece, ara verilerin ve eğimlerin hepsi GPU'da kalacaktır. Blob'un verileri ne zaman kopyalayacağı kontrol edilmek isterse, aşağıdaki örnekteki gibi yapılabilir:

```
// Başlangıçta verinin CPU üzerinde olduğunu varsayarsan Bir blob'muz olur.
const Dtype* foo;
Dtype* bar;
foo = blob.gpu_data(); // veri kopyalama cpu->gpu.
foo = blob.cpu_data(); // her ikisi de güncel içeriği olduğundan hiçbir veri kopyalanmadı.
bar = blob.mutable_gpu_data(); // veri kopyalanmaz
// .. Bazı işlemler yapılır.. //
bar = blob.mutable_gpu_data(); // hâlâ GPU'dayken hiçbir veri kopyalanmadı.
foo = blob.cpu_data(); // gpu tarafı veriyi değiştirdiği içinveri kopyalama gpu->cpu
foo = blob.gpu_data(); // her ikisi de güncel içeriği olduğundan hiçbir veri kopyalanmadı
```

bar = blob.mutable_cpu_data(); // hâlâ hiçbir veri kopyalanmadı.

bar = blob.mutable_gpu_data(); // veri kopyalama cpu->gpu.

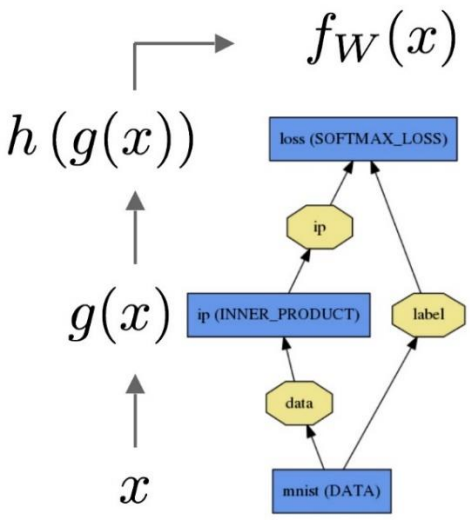
bar = blob.mutable_cpu_data(); // veri kopyalama gpu->cpu.

3.3.3.3. Model Format

Öğrenilen modeller binary protocol buffer (binaryproto) “.caffe” model dosyaları olarak serileştirilirken, modeller düz metin olarak prototip protocol buffer şemasında (prototxt) tanımlanır. Model biçimi, "caffe.proto'daki protobuf şeması tarafından tanımlanır.

3.3.4. Forward / Backward

Forward ve backward geçişler, net’in temel hesaplamalarıdır. Basit bir lojistik regresyon sınıflandırıcısı düşünelim. Forward geçiş, çıkarım için girdi verilen çıktıyı hesaplar. Forward Caffe model tarafından temsil edilen "fonksiyon" u hesaplamak için her katmanın hesaplamasını oluşturur. Bu geçiş alttan üste gider.



Şekil.3. Forward

Veri x, g (x) için bir iç ürün katmanından (INNER_PRODUCT) sonra h (g (x)) için bir softmax üzerinden ve fw(x) vermek için softmax kaybından geçer. Backward geçiş, öğrenme kaybına verilen gradyanı hesaplar. Backward geçişte Caffe, otomatik modellemeyle tüm modelin gradyanını hesaplamak için her tabakanın gradyanını ters oluşturur. Bu geri yayılımdır. Bu geçiş yukarıdan aşağıya doğru gider.

Caffe, sizin için forward ve backward geçişlerini planlar ve uygular. Framework tarafındaki kod çağırımları ve zamanlamaları aşağıdaki gibidir.

- *Layer::Forward()* ve *Layer::Backward()* her adımda hesaplanırken, *Net::Forward()* ve *Net::Backward()* metotları ilgili geçişleri gerçekleştirir.
- Her katman türü, hesaplama moduna göre adımlarını hesaplamak için *forward_ {cpu, gpu}()* ve *backward_ {cpu, gpu}()* yöntemlerine sahiptir. Bir katman, kısıtlamalar veya kolaylık nedeniyle *CPU* veya *GPU* modunu uygulayabilir.

Solver (Çözücü), ilk önce çıktı ve kaybı vermek için forward çağırarak modeli en iyi duruma getirir. Daha sonra modelin gradyanını oluşturmak için backward çağırır ve ardından gradyan kaybı en aza indirgeye çalışan bir ağırlık güncellemesine dahil eder. Solver (Çözücü), Net ve Katman arasındaki iş bölünmesi Caffe'yi modüler ve gelişime açık tutuyor.

3.3.5. Loss

Caffe'de, makine öğrenmenin çoğunda olduğu gibi öğrenme, bir kayıp fonksiyonu (aynı zamanda bir hata, maliyet veya nesnel fonksiyon olarak da bilinir) tarafından yönlendirilir. Bir loss fonksiyonu, parametre ayarlarını (diğer bir deyişle mevcut net ağırlıklarını) bu parametre ayarlarının "kötü" olduğunu belirten bir skalar değerine haritalanması suretiyle öğrenmenin amacını belirtir. Dolayısıyla, öğrenmenin amacı, loss fonksiyonunu en aza indirgeyen ağırlık ayarlarını bulmaktır. Caffe'deki loss, net'in forward geçişi ile hesaplanır. Her katman bir dizi giriş (alt) blob alır ve bir çıktı (üst) blob oluşturur. Bu katmanların bazılarının çıktıları loss fonksiyonunda kullanılabilir. One-vs-All tipinde tüm

sınıflandırma görevleri için tipik bir loss fonksiyonu seçimi, aşağıdaki gibi bir net tanımında kullanılan SoftmaxWithLoss işlevidir:

```
layer {
  name: "loss"
  type: "SoftmaxWithLoss"
  bottom: "pred"
  bottom: "label"
  top: "loss"
}
```

Bir SoftmaxWithLoss fonksiyonu içinde, üstteki blob, tüm mini toplu işlem boyunca kaybı (öngörülen etiketlerden ve fiili etiketlerden hesaplanan) bir skalaya (boş bir şekle) sahiptir.

Loss Weights: Birden fazla katmana sahip olan ağlar (örneğin, bir SoftmaxWithLoss katmanı kullanarak girdiyi sınıflandırıp bir EuclideanLoss katmanı kullanarak yeniden yapılandıran bir ağ gibi) için kaybetme ağırlıkları göreli önemlerini belirtmek için kullanılabilir. Kurallara göre, loss fonksiyonuna (Kaybetme) sahip Caffe katman türleri, loss fonksiyonuna katkıda bulunur, ancak diğer katmanları sadece ara hesaplamalar için kullanılır. Bununla birlikte, herhangi bir katman, katmanın ürettiği üst blob için katman tanımına loss_weight: <float> alanını ekleyerek bir katmanı kayıp olarak kullanılabilir.

```
layer {
  name: "loss"
  type: "SoftmaxWithLoss"
  bottom: "pred"
  bottom: "label"
  top: "loss"
  loss_weight: 1
}
```

Backward yayma yapabilen herhangi bir katmana, eğer istenirse, ağın bazı ara katmanları / grupları tarafından üretilen aktivasyonları düzenli hale getirmek için sıfır olmayan bir loss_weight verilebilir. Sıfırdan farklı bir loss ile ilişkili tek olmayan çıktılar için, loss sadece blob'un tüm girdileri üzerinden toplanarak hesaplanır. Son olarak Caffe'deki loss, aşağıdaki pseudo-code'da olduğu gibi net üzerindeki toplam ağırlıklı loss'un toplanmasıyla hesaplanır:

```
loss := 0
for layer in layers:
  for top, loss_weight in layer.tops, layer.loss_weights:
    loss += loss_weight * sum(top)
```

3.3.6. Solver

Solver, loss iyileştirmeye çalışan parametre güncelleştirmelerini oluşturmak için net'in forward çıkarımını ve backward gradyanlarını koordine ederek model optimizasyonunu düzenler. Öğrenme sorumlulukları, optimizasyonun denetimi için Solver ile parametre güncellemeleri, loss ve gradyanlara neden olan Net arasında bölünmüştür. Caffe modelleri gibi, Caffe çözücüleri CPU / GPU modlarında çalışır.

Caffe çözücüleri;

- Stochastic Gradient Descent (type: "SGD")
- AdaDelta (type: "AdaDelta")
- Adaptive Gradient (type: "AdaGrad")
- Adam (type: "Adam")
- Nesterov's Accelerated Gradient (type: "Nesterov")
- RMSprop (type: "RMSProp")

Çözücüler;

- Optimizasyon muhasebesini destekler ve öğrenme ve test ağı ve değerlendirme için eğitim ağı oluşturur.
- Forward / Backward çağırarak ve parametreleri güncelleyerek yinelemeli olarak optimize eder.
- Test ağlarını periyodik olarak değerlendirir.
- Optimizasyon boyunca model ve çözücü durumlarını anlık olarak görüntüler.

Her iterasyonda;

- Çıktının ve loss hesaplanması için net'i forward eder.
- Gradyanları hesaplamak için net'i backward eder.
- Gradyanları Solver yöntemine göre parametre güncellemelerine dahil eder.
- Öğrenme hızı, geçmişi ve metoduna göre Solver durumunu günceller.

Solver yöntemleri loss küçültmesi için genel optimizasyon problemini ele alır. 'D' veri kümesi için, optimizasyon hedefi tüm '|D|' üzerindeki ortalama loss'dur.

3.3.7. Layer Catalogue

Caffe modeli oluşturmak için model mimarisini bir *protocol buffer* tanım dosyasında (prototxt) tanımlamanız gerekir. Caffe katmanları ve parametreleri, caffe.proto'daki proje için *protocol buffer* tanımlarında tanımlanmıştır.

3.3.7.1. Vision Layers

Görme katmanları genellikle görüntüleri girdi olarak alır ve diğer görüntüleri çıktı olarak üretir. Gerçek dünyadaki tipik bir "görüntü", gri tonlamalı bir resimdeki gibi bir renk kanalına (c = 1) veya RGB (kırmızı, yeşil, mavi) görüntüdeki gibi üç renk kanalına (c = 3) sahip olabilir. Fakat bu bağlamda bir görüntünün ayırt edici özelliği, mekânsal yapısıdır. Genellikle bir görüntünün yüksekliği h> 1 ve genişliği w> 1'dir ve önemsizdir. Bu 2D geometri de, doğal olarak girdinin nasıl işleneceği ile ilgili belirli kararları verir. Özellikle, görme katmanlarının çoğu, çıktının bir bölgesini üretmek için girdinin bazı bölgesine belirli bir işlem uygulayarak çalışır. Buna karşın, diğer katmanların (birkaç istisna dışında) girdinin mekânsal yapısını görmezden gelmesi, buna chw (channel*height*weight) boyutlu büyük bir vektör olarak etkili bir şekilde davranması gerekir.

Vision Layer Çeşitleri;

Convolution:

- Katman tipi: Convolution
- CPU uygulama: ./src/caffe/layers/convolution_layer.cpp
- CUDA GPU uygulama: ./src/caffe/layers/convolution_layer.cu
- Girdi : n * c_i * h_i * w_i
- Çıktı: n * c_o * h_o * w_o ‘dır. h_o = (h_i + 2 * pad_h - kernel_h) / stride_h + 1 ve w_o = h_o

Örnek tanımlama;

```
layer {
  name: "conv1"
  type: "Convolution"
  bottom: "data"
  top: "conv1"
  param { lr_mult: 1 decay_mult: 1 }
  param { lr_mult: 2 decay_mult: 0 }
  convolution_param {
    num_output: 96     # 96 filtre
    kernel_size: 11    # 11x11 filtreler
    stride: 4          # Her filtre uygulaması arasına 4pixel adım
    weight_filler {
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
      type: "constant"
      value: 0
    }
  }
}
```

Pooling:

- Katman tipi: Pooling
- CPU uygulama: ./src/caffe/layers/pooling_layer.cpp
- CUDA GPU uygulama: ./src/caffe/layers/pooling_layer.cu
- Girdi : n * c_i * h_i * w_i
- Çıktı: n * c_o * h_o * w_o ‘dır. h_o = (h_i + 2 * pad_h - kernel_h) / stride_h + 1 ve w_o = h_o

Örnek tanımlama;

```
layer {
  name: "pool1"
  type: "Pooling"
  bottom: "conv1"
  top: "pool1"
  pooling_param {
    pool: MAX
    kernel_size: 3 # pool over a 3x3 region
    stride: 2      # step two pixels (in the bottom blob) between pooling regions
  }
}
```

Local Response Normalization (LRN):

- Katman tipi: LRN
- CPU uygulama: ./src/caffe/layers/lrn_layer.cpp
- CUDA GPU uygulama: ./src/caffe/layers/lrn_ayer.cu
- Girdi : n * c_i * h_i * w_i
- Çıktı: n * c_o * h_o * w_o ‘dır. h_o = (h_i + 2 * pad_h - kernel_h) / stride_h + 1 ve w_o = h_o

Yerel yanıt normalizasyon katmanı, lokal giriş bölgelerinde normalleştirme yaparak bir tür "yanal ketleme" gerçekleştirir. ACROSS_CHANNELS modunda, yerel bölgeler yakınlardaki kanallara yayılır ancak mekânsal kapsamı yoktur. WITHIN_CHANNEL modunda yerel bölgeler uzaysal olarak uzar ancak ayrı kanallardır. Her bir girdi değeri, n'nin her yerel bölgenin boyutu olduğu bölüme ayrılır ve bu toplam, o değere ortalanmış bölge üzerinden alınır.

Im2Col: im2Col Caffe'in orijinal konvolüsyonunda, tüm yamaları bir matris içine yerleştirerek matris çarpması yapmak için kullanılır.

3.3.7.2. Loss Layers

Loss katman, kaybı bir çıktıyla öğrenmek için bir hedefe yönlendirir ve maliyeti en aza indirgemek için atama yapar. Kaybın kendisi forward ile hesaplanır ve kaybolma gradyanı backward geçiş ile hesaplanır.

Loss Layer çeşitleri;

Softmax: Softmax loss katmanı, girişlerin softmax'ının çoklu-lojistik kaybını hesaplar. Kavramsal olarak softmax katmanının ardından çoklu terimli lojistik loss katmanı ile aynıdır ancak sayısal olarak daha istikrarlı bir geçiş derecesi sağlar. Katman Tipi SoftmaxWithLoss'dur.

Sum-of-Squares / Euclidean: Öklid kayıp tabakası (Sum-of-Squares / Euclidean) iki girdisinin farklılıklarının kareler toplamını hesaplar. Katman Tipi EuclideanLoss'dur.

Hinge / Margin:

- Katman tipi: HingeLoss
- CPU uygulama: ./src/caffe/layers/hinge_loss_layer.cpp
- CUDA GPU uygulama: ./src/caffe/layers/hinge_loss_layer.cu
- Girdi :
  - N * c * h * w Tahminler (Predictions)
  - N * 1 * 1 * 1 Etiketler (Labels)
- Çıktı: 1 * 1 * 1 * 1 Hesaplanan Kayıp (Computed Loss)

Örnek tanımlama;

```
# L1 Norm
        layer {
          name: "loss"
          type: "HingeLoss"
          bottom: "pred"
          bottom: "label"
        }

# L2 Norm
        layer {
          name: "loss"
          type: "HingeLoss"
          bottom: "pred"
          bottom: "label"
          top: "loss"
          hinge_loss_param {
            norm: L2
          }
        }
```

Sigmoid Cross-Entropy: Katman tipi SigmoidCrossEntropyLoss'dur.

Infogain: Katman tipi InfogainLoss'dur.

Accuracy and Top-k: Accuracy, çıktıyı hedefe göre çıktı hassasiyeti olarak puanlandırır. Aslında bir loss değildir ve backward'ı yoktur.

3.3.7.3. Activation / Neuron Layers

Genel olarak, Activation / Neuron katmanları eleman bazlı operatörler olup, bir alt blob alır ve aynı boyutta bir üst blob üretirler. Aşağıdaki katmanlarda, giriş ve çıkış boyutlarını özdeş oldukları için görmezden geleceğiz.

- Girdi : n * c * h * w
- Çıktı : n * c * h * w

Activation / Neuron Layers çeşitleri;

ReLU Rectified-Linear and Leaky: Bir giriş değeri x göz önüne alındığında, ReLU katmanı çıktıyı x> 0 ise x olarak, x <= 0 ise negatif_slope * x olarak hesaplar. Negatif eğim parametresi ayarlanmadığında, standart ReLU fonksiyonunun max (x , 0) aynı zamanda, yerinde hesaplamayı da destekler. Bu, alt ve üst blobun bellek tüketimini korumak için aynı olabileceği anlamına gelir.

- Katman tipi: ReLU
- CPU uygulama: ./src/caffe/layers/relu_layer.cpp
- CUDA GPU uygulama: ./src/caffe/layers/relu_layer.cu

Örnek tanımlama;

```
layer {
  name: "relu1"
  type: "ReLU"
  bottom: "conv1"
  top: "conv1"
}
```

Sigmoid: Sigmoid katmanı, her girdi elemanı x için sigmoid (x) olarak çıktıyı hesaplar.

- Katman tipi: Sigmoid
- CPU uygulama: ./src/caffe/layers/sigmoid_layer.cpp
- CUDA GPU uygulama: ./src/caffe/layers/sigmoid_layer.cu

Örnek tanımlama;

```
layer {
  name: "encode1neuron"
  bottom: "encode1"
  top: "encode1neuron"
  type: "Sigmoid"
}
```

TanH / Hyperbolic Tangent: TanH katmanı, çıktıyı her bir x girdi elemanı için tanh (x) olarak hesaplar.

- Katman tipi: TanH
- CPU uygulama: ./src/caffe/layers/tanh_layer.cpp
- CUDA GPU uygulama: ./src/caffe/layers/tanh_layer.cu

Örnek tanımlama;

```
layer {
  name: "layer"
  bottom: "in"
  top: "out"
  type: "TanH"
}
```

Absolute Value: AbsVal katmanı, çıktıyı her bir x girdi elemanı için AbsVal(x) olarak hesaplar.

- Katman tipi: AbsVal
- CPU uygulama: ./src/caffe/layers/absval_layer.cpp
- CUDA GPU uygulama: ./src/caffe/layers/absval_layer.cu

Örnek tanımlama;

```
layer {
  name: "layer"
  bottom: "in"
  top: "out"
  type: "AbsVal"
}
```

Power: Power katmanı, çıktıyı her bir x girdi elemanı için (shift + scale * x)^ power olarak hesaplar.

- Katman tipi: Power
- CPU uygulama: ./src/caffe/layers/power _layer.cpp
- CUDA GPU uygulama: ./src/caffe/layers/power _layer.cu

Örnek tanımlama;

```
layer {
  name: "layer"
  bottom: "in"
  top: "out"
  type: "Power"
  power_param {
    power: 1
    scale: 1
    shift: 0
  }
}
```

BNLL: BNLL katmanı, çıktıyı her bir x girdi elemanı için log(1 + exp(x)) olarak hesaplar.

- Katman tipi: BNLL
- CPU uygulama: ./src/caffe/layers/bnll _layer.cpp
- CUDA GPU uygulama: ./src/caffe/layers/bnll _layer.cu

Örnek tanımlama;

```
layer {
  name: "layer"
  bottom: "in"
  top: "out"
  type: "BNLL"
}
```

3.3.7.4. Data Layers

Veriler, veri katmanları aracılığıyla Caffe'ye girer, ağların altına uzanır. Veriler verimli veritabanlarından (*LevelDB* veya *LMDB*) doğrudan belleğe ya da etkinlik kritik olmadığında *HDF5'deki* diskten dosyalardan veya ortak resim formatlarından gelebilir. Ortak girdi ön işleme (ortalama çıkarma, ölçeklendirme, rastgele kırpma ve yansıtma) *TransformationParameters* belirterek kullanılabilir. Başlıca data layers tipleri aşağıda verilmiştir.

Database: Katman tipi Data'dır.

In-Memory: Bellek veri katmanı, verileri kopyalamaksızın doğrudan bellekten okur. Kullanabilmek için, bitişik veri kaynağı (4D satırdan büyük dizi) belirlemek için *MemoryDataLayer::Reset()(C++'dan) veya Net.set_input_arrays()(Python'dan)* çağrılmalıdır. Bu fonksiyonlar çağırıldığında yığın olarak toplu iş gibi bir okuma yapılır. Katman Tipi Memory Data'dır.

HDF5 Input: Katman tipi HDF5Data'dır.

HDF5 Output: HDF5 çıktı katmanı, bu bölümdeki diğer katmanların tersine okuma yerine giriş bloblarını diske yazar. Katman tipi HDF5Output'dır.

Images: Katman tipi ImageData'dır.

Windows: Katman tipi WindowData'dır.

Dummy: Dummy development için kullanılan veri katmanıdır. Bu katman için *DummyDataParemeter* parametresi kullanılmalıdır.

3.3.7.5. Common Layers

Common Layer çeşitleri aşağıda verilmiştir.

Inner Product: InnerProduct katmanı (genellikle tam bağlı katman olarak anılır) girdiyi basit bir vektör olarak ele alır ve tek bir vektör (blob'un yüksekliği ve genişliği 1 olarak ayarlanır) şeklinde bir çıktı üretir.

- Katman tipi: InnerProduct
- CPU uygulama: ./src/caffe/layers/inner_product_layer.cpp
- CUDA GPU uygulama: ./src/caffe/layers/inner_product_layer.cu
- Girdi : n * c_i * h_i * w_i
- Çıktı: n * c_o * 1 * 1

Örnek tanımlama;

```
layer {
  name: "fc8"
  type: "InnerProduct"
  # learning rate and decay multipliers for the weights
  param { lr_mult: 1 decay_mult: 1 }
  # learning rate and decay multipliers for the biases
  param { lr_mult: 2 decay_mult: 0 }
  inner_product_param {
    num_output: 1000
    weight_filler {
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
```

```
    type: "constant"
    value: 0
   }
  }
  bottom: "fc7"
  top: "fc8"
 }
```

Splitting: Split katmanı, girdi blobunu çoklu çıktı bloblarına bölen bir yardımcı katmandır. Bir blob birden fazla çıktı katmanına beslendiğinde kullanılır.

Flattening: Flattening(Düzleştirme) katmanı, n * c * h * w şeklindeki bir girdiyi, n * (c * h * w) şeklindeki basit bir vektör çıktısına yuvarlatan bir yardımcı katmandır.

Reshape: Reshape katmanı, verilerinin değiştirilmeden girişinin boyutlarını değiştirmek için kullanılabilir. Düzleştirme katmanı gibi yalnızca boyutlar da değiştirilir; İşlemde hiçbir veri kopyalanmaz. Çıktı boyutları, ReshapeParam parametresi ile belirtilir. Pozitif sayılar doğrudan çıktı blobunun ilgili boyutunu belirleyerek kullanılır. Buna ek olarak, hedef boyut değerlerinden herhangi biri için iki özel değer kabul edilir:

- 0 : "alt katmanın ilgili boyutunu kopyalamak" anlamına gelir. Diğer bir deyişle, birinci boyut olarak 2 tane varsa, üstteki birinci boyut olarak dim: 0 olarak verilen birinci boyutta da 2 olur.
- -1: Diğer boyutlardan çıkarmak anlamına gelir. Bu davranış, MATLAB'ın yeniden şekillendirmesi için numpy veya [] 'de -1'e benzer. Boyut, toplam öğe sayısını alt katmandaki gibi tutmak için hesaplanır. -1 yeniden şekillendirme işleminde de kullanılabilir.

Başka bir örnek olarak, reshape_param {shape {dim: 0 dim: -1}} belirtilerek bu katmanın Düzleştirme katmanı ile aynı şekilde davranması sağlanabilir.

- Katman tipi: Reshape
- CPU uygulama: ./src/caffe/layers/reshape_layer.cpp
- Girdi : Herhangi bir boyuttaki tek blob
- Çıktı: Parametre olarak verilen reshape boyutunda blob çıktısı üretir.

Örnek tanımlama;

```
layer {
  name: "reshape"
  type: "Reshape"
  bottom: "input"
  top: "output"
  reshape_param {
    shape {
      dim: 0  # copy the dimension from below
      dim: 2
      dim: 3
      dim: -1 # infer it from the other dimensions
    }
  }
}
```

Concatenation: Concat katmanı, çoklu girdi bloblarını tek bir çıkış blobuna bağlayan bir yardımcı katmandır.

- Katman tipi: Concat
- CPU uygulama: ./src/caffe/layers/concat_layer.cpp
- CUDA GPU uygulama: ./src/caffe/layers/concat_layer.cu
- Girdi : n_i * c_i * h * w, her girdi blob i için 1'den K'ya kadar.
- Çıktı:

- Eksen = 0 ise: (n_1 + n_2 + ... + n_K) * c_1 * h * w ve tüm girdi c_i aynı olmalıdır.
- Eksen = 1 ise: n_1 * (c_1 + c_2 + ... + c_K) * h * w ve tüm girdi n_i aynı olmalıdır.

Örnek tanımlama;

```
layer {
  name: "concat"
  bottom: "in1"
  bottom: "in2"
  top: "out"
  type: "Concat"
  concat_param {
    axis: 1
  }
}
```

Slicing: Slice katmanı, bir girdi katmanını, verilen dilim indeksleri ile belirli bir boyut (şu anki num veya channel) boyunca çoklu çıktı katmanlarına bölen bir hizmet katmanıdır.

Örnek tanımlama;

```
layer {
  name: "slicer_label"
  type: "Slice"
  bottom: "label"
  ## Example of label with a shape N x 3 x 1 x 1
  top: "label1"
  top: "label2"
  top: "label3"
  slice_param {
```

```
    axis: 1
    slice_point: 1
    slice_point: 2
  }
}
```

Elementwise Operations: Katman tipi Eltwise'dır.

Argmax: Katman tipi Argmax'dır.

Softmax: Katman tipi Softmax'dır.

Mean-Variance Normalization: Katman tipi MVN'dır.

3.3.8. Interfaces

Caffe, günlük kullanım için komut satırı, Python ve MATLAB ara birimlerine sahiptir. Caffe temelde bir C ++ kütüphanesi ve geliştirme için modüler bir ara yüze sahipken, her fırsatta özel derleme gerektirmez. Cmdcaffe, pycaffe ve matcaffe arabirimleri open source olarak kullanıma sunulmuştur.

3.3.8.1. Command Line

Komut satırı arabirimi - cmdcaffe - model eğitimi, skorlama ve tespit için kullanılan Caffe aracıdır. Yardım için herhangi bir argüman olmadan Caffe'yi çalıştırabilirsiniz. Bu araç ve diğerleri caffe/build/tools altında bulunur.

Örnek komutlar;

```
# training LeNet
    caffe train -solver examples/mnist/lenet_solver.prototxt
# training on GPU 2
    caffe train -solver examples/mnist/lenet_solver.prototxt -gpu 2
# resume training from the half-way point snapshot
    caffe train -solver examples/mnist/lenet_solver.prototxt -snapshot examples/mnist/lenet_iter_5000.solverstate
```

3.3.8.2. Python

Python ara birimi - pycaffe - Caffe modülü ve komut dosyaları caffe/python dizininde bulunur. Modelleri yüklemek, ileri geri hareket ettirmek, IO'yu idare etmek, ağları görselleştirmek ve hatta enstrüman modelini çözmek için Caffe ithal (import) edilir. Tüm model verileri, türevleri ve parametreleri okuma ve yazma için kullanılmaktadır.

- caffe.Net, modelleri yüklemek, yapılandırmak ve çalıştırmak için merkezi bir ara birimdir. Sınıflandırıcı ve caffe.Detector ortak görevler için kolaylık arayüzleri sağlar.
- caffe.SGDSolver solver arayüzünü sağlar.
- caffe.io, önişleme ve protokol tamponlarıyla girdi / çıktıyı işler.
- caffe.draw, net mimarilerini görselleştirir.
- Caffe blob'ları, kullanım kolaylığı ve verimlilik için numpy ve ndarrays olarak kullanılır.

3.3.8.3. Matlab

Matcaffe Caffe'yi matlab kodunuza entegre edebileceğiniz pakettir. caffe/matlab altında bulunur. Matcaffe ile yapabileceklerimiz aşağıdaki gibidir:

- Matlab'da birden çok nets oluşturma.
- Forward ve Backward hesaplama yapma.
- Bir nets içindeki herhangi bir katmana ve bir katmandaki parametre blobuna erişme.
- Bir nets'in parametrelerini dosyaya kaydetme ve parametreleri dosyadan yükleme.
- Bir reshape yeniden şekillendirme ve bir nets'i yeniden şekillendirme.
- Nets parametrelerini ve nets düzenleme.
- Eğitim için Matlab'da çoklu Solver oluşturma.
- Çözücü Snapshots'larından eğitimlere devam etme.
- Eğitim ve Solver nets'ine ulaşma.
- Belirli sayıda yineleme için çalışma ve kontrolü geri Matlab'a verme.
- matcaffe kullanmak için ModelZoo hazır modeli ile çalışabilirsiniz.

3.3.9. Data

Veriler Caffe'den Blob olarak akar. Veri katmanları, Blob'dan diğer biçimlere dönüştürerek girdi yükler ve çıktı kaydeder. Ortalama çıkarma ve özellik ölçekleme gibi yaygın dönüşümler veri katmanı yapılandırmasıyla yapılır. Yeni veri katmanı geliştirilerek yeni girdi türleri desteklenir.

Data katmanı tanımlama;

```
layer {
  name: "mnist"
  # Veri katmanı, yüksek verim için leveldb veya lmdb depolama DB'leri yükler.
  type: "Data"
  # İlk üstteki veri kendisidir: adı sadece örnek
  top: "data"
  # ikinci üstteki veri gerçek data'dır. adı sadece örnek.
  top: "label"
  # veri katmanı konfigurasyonu
  data_param {
    # DB yolu
    source: "examples/mnist/mnist_train_lmdb"
    # DB tipi LMBD veya LevelDB
    backend: LMDB
    # toplu işlem verimlilik boyutu.
    batch_size: 64
  }
  # ortak veri dönüşümleri
  transform_param {
    # Özellik ölçekleme katsayısı: model  verisini [0, 255] den  [0, 1] 'a
    scale: 0.00390625
  }
}
```

Katmanda kullanılan parametrelerin özellikleri aşağıdaki gibidir:

Tops ve Bottoms: Bir veri katmanı, üst blob'ları modele veri çıktısı haline getirir. Hiçbir girdi almadığı için alt kısımları yoktur.

Data ve Label: Bir veri katmanı en azından kanonik olarak adlandırılmış en az bir veriye sahiptir. Zemin gerçeği için, kanonik olarak etiket olarak adlandırılan ikinci bir üst tanımlanabilir. Her iki üstte sadece blob üretilir ve bu isimler hakkında özünde hiçbir şey özel değildir. (Data, Label) eşleştirme, classification modelleri için bir kolaylıktır.

Transformations: Veri ön işleme, veri katmanı tanımındaki dönüştürme mesajları ile parametreleştirilir.

Bölüm boyunca kullanmış olduğumuz derin öğrenme alt yapısı olan caffe ve tüm bileşenleri detaylı olarak anlatılmıştır.

4. RAKAM ÖĞRENME UYGULAMASI

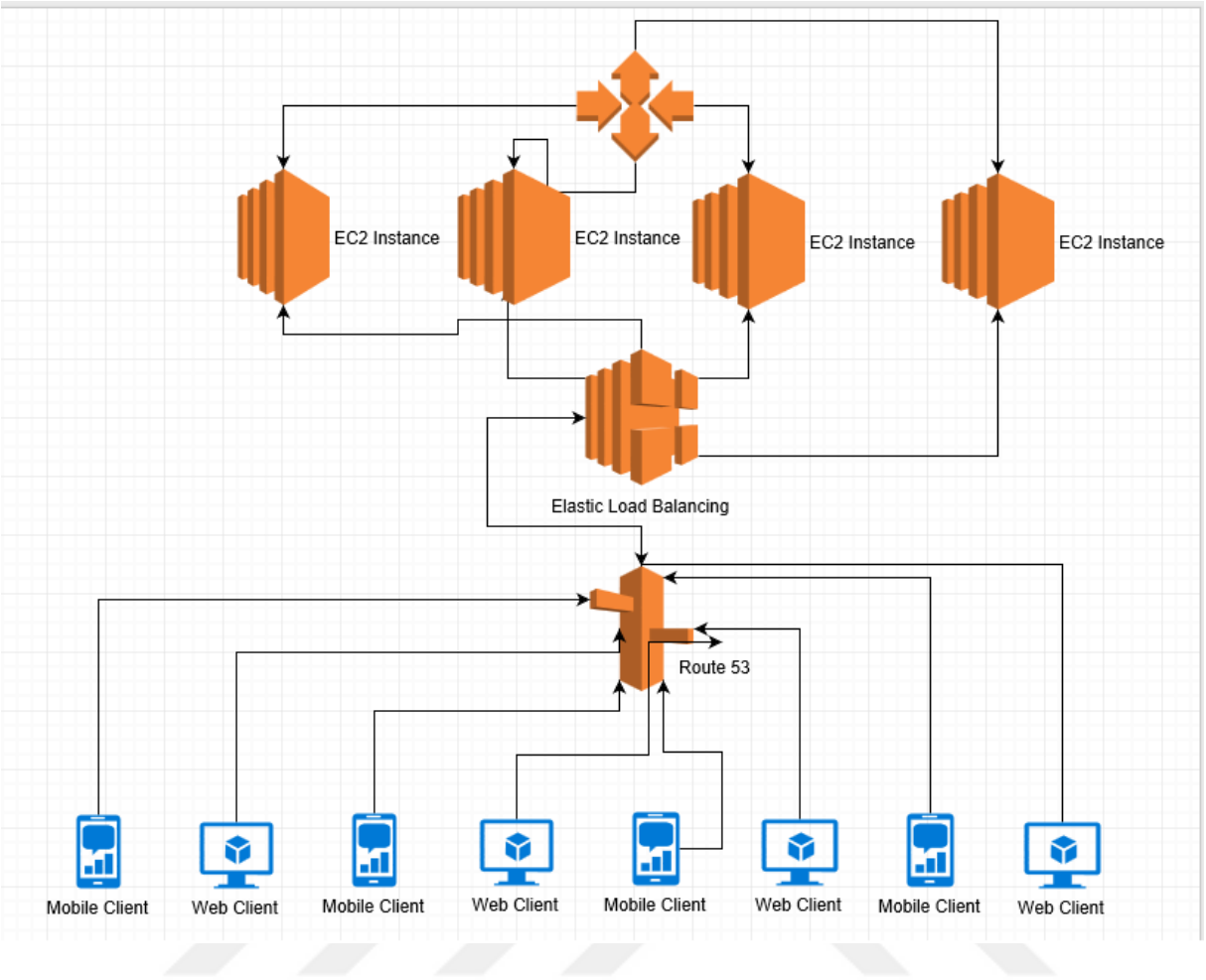
4.1. Giriş

Rakam Öğrenme, derin öğrenme, mobil teknolojiler ve web teknolojilerinin birbirleri ile haberleşerek eğitim alanındaki konulara çözüm bulmayı amaçlayan ve bu tez çalışmasında geliştirilen bir sistemdir.

Uygulama mobil ve web platformlarında çalışarak, kullanım alanlarını arttırır ve daha çok kullanıcıya hitap eder. Sistemin mobil kullanıcıları anaokul öğrencileri, web kullanıcıları öğretmenler olacak şekilde tasarlanmıştır. Öğrencilerin rakam öğrenmelerini destekleyici bir eğitim uygulaması tasarlanmıştır.

Öğrencilerin okul dışında kendi evlerinde veya müsait oldukları herhangi bir yerde ders çalışmalarına ve rakam öğrenmelerine olanak sağlanmaktadır. Böylece öğrencilerin eğitime destek sağlayıcı bir sistem hayata geçirilmiştir.

Sistemin yazılım mimarisi servis ve web uygulamalarının AWS alt yapısını kullanacak şekilde kurgulanmıştır. Client kullanıcıları web ve mobil uygulamaları kullanan kişiler olarak kurgulanmıştır. Servise gelen istekler öncelikle en ön katmandaki Route53 dns servisine ulaşır. Burada gelen istek çözülerek, load balancer'a iletilir. Load balancer daha uygun pozisyondaki sunucuya isteği gönderir. Sunucuların üzerinde çalışan auto scaling servisi istek yoğunluğuna göre yeni sunucu açar veya kapatır. Tüm bu Client-Server mimarisi Şekil 4. te gösterilmiştir.



Şekil.4. Rakam öğrenme mimari yapısı

4.2. Kullanılan Teknolojiler

Bu bölümde rakam öğrenme sisteminde kullanılan tüm teknolojiler hakkında bilgiler verilecektir. Teknolojilerin sistemdeki kullanım amaçlarının anlatılması, hangi teknolojinin ne sebeple seçildiğinin anlaşılmasını kolaylaştıracaktır.

4.2.1. iOS

iOS, Amerikalı Apple firmasının telefon ve tablet cihazlarında kullanılmak üzere piyasaya çıkardığı mobil bir işletim sistemidir. Donanım ve yazılım uyumunun dikkate alınarak geliştirildiği işletim sisteminde en iyi performans elde edilmektedir. iOS rakiplerine oranla her zaman daha az donanım ile daha yüksek performans sağlamaktadır.

Geliştirilmiş olan Rakam Öğrenme uygulamasının öğrencilerin kullanacağı mobil kısımlar iOS platformu üzerinde çalışacak şekilde tasarlanmış ve geliştirilmiştir. iOS platformunun kararlığı ve kullanıcı deneyiminin rakibi konumundaki Android'e göre daha iyi durumda olması bu seçiminin sebebidir. Uygulama hem iOS telefonlarda hem de iOS tabletler üzerinde çalışacak şekilde tasarlanmıştır.

4.2.1.1. Objective-C

Objective-C iOS platformu üzerinde yazılım geliştirmek için kullanılan bir programlama dilidir. Sistemim geliştirilmesi için bir araç olarak kullanılmıştır. iOS üzerinde çalışan bir uygulamaya sahip olmak için gerekliliktir.

Objective-C, C'nin üzerine yazılmış, yansımalı, nesne yönelimli bir programlama dilidir. ObjC, Objective C ve Obj-C olarak da anılır.

Günümüzde OpenStep standardı üzerine kurulu olan Mac OS X ve GNUstep işletim sistemlerinde kullanılmaktadır. Objective-C'nin en yaygın olarak kullanıldığı alan Cocoa çatısının kullanıldığı yazılımlardır. Bu özel kütüphanelere erişime ihtiyaç duymayan bir Objective-C programı Objective-C derleyicisi içeren gcc ile derlenebilir.

4.2.1.2. Xcode

Xcode MacOS, iOS, watchOS ve tvOS platformları için yazılım geliştirme ortamıdır. İlk kez 2003 yılında piyasaya sürülen sürümü ilk kararlı sürümdür. Günümüzde Xcode 8.0 çıkmıştır. MacOS işletim sistemleri üzerinde çalışabilen Xcode işletim macOS işletim sistemleri gibi ücretsiz olarak geliştiricilere sunulmaktadır.

4.2.2. Python

Python derin öğrenme algoritmalarımızı, web servislerimizi, web uygulamalarımızı geliştirdiğimiz programlama dilidir. Bu programlama dilini seçmemizin sebebi, Caffe ile olan uyumu ve desteğinin olmasının en büyük etkendir. Ayrıca Python diğer programlama dillerine göre daha hızlı çalışır ve yüksek seviyeli güçlü bir programlama dilidir. Bundan dolayı dünya üzerinde geliştirilen askeri savunma uygulamaları, hacking network tool'ları bu dil ile geliştirilmektedir. Aşağıda bu dilin açıklamaları ve tarihsel gelişimi anlatılmaktadır.

Geliştirilmeye 1990 yılında Guido van Rossum tarafından Amsterdam'da başlanmıştır. Adını sanılanın aksine bir yılandan değil Guido van Rossum'un çok sevdiği, Monty Python adlı altı kişilik bir İngiliz komedi grubunun Monty Python's Flying Circus adlı gösterisinden almıştır. Günümüzde Python Yazılım Vakfı çevresinde toplanan gönüllülerin çabalarıyla sürdürülmektedir. Python 1.0 sürümüne Ocak 1994'te ulaşmıştır. Son kararlı sürümü, 2.x serisinde Python 2.7 ve 3.x serisinde Python 3.5.2'dir. 3 Aralık 2008 tarihinden itibaren 3.x serisi yayınlanmaya başlamıştır ancak 3.x serisi 2.x serisiyle uyumlu değildir. [10]

Python'un son derece kolay okunabilir olması düşünülmüştür. Bu yüzden örneğin küme parantezleri yerine girintileme işlemi kullanılır. Hatta bazı durumlarda girintileme işlemine dahi gerek kalmadan kodun ilgili bölümü tek satırda yazılabilir. Böylece Python, program kodunuzu en az çaba ile ve hızlıca yazmanıza imkân tanır. Sade sözdizimi ile diğer programlama dillerinden üstündür. [10]

Django, Zope uygulama sunucuları, YouTube ve orijinal BitTorrent istemcisi, Pardus Linux dağıtımı Python kullanan önemli projelerden bazılarıdır. Ayrıca Google, NASA ve CERN gibi büyük kurumlar da Python kullanmaktadır. Ayrıca OpenOffice.org, GIMP, Inkscape, Blender, Scribus ve Paint Shop Pro gibi bazı programlarda betik dili olarak kullanılır. Pek çok Linux dağıtımında Python ön tanımlı bir bileşen olarak gelir. [10]

4.2.3. Digits

Derin öğrenme konusunda donanım olarak çözümler sunan NVIDIA, ilgili bilim insanları için NVIDIA DiGiTS uygulamasını kullanıma sunmuştur. NVIDIA DiGiTS, yani Derin Öğrenme GPU Eğitim Sistemi (Deep Learning GPU Training System – DIGITS) Yangqing Jia tarafından geliştirilen programlama yapmaksızın kullanılan Caffe isimli derin öğrenme aracını web ara yüzü desteği sağlayarak görselleştirmiştir.

San Francisco'da (ABD) yapılan GPU Teknoloji Konferansı'nda NVIDIA CEO'su ve Kurucu Ortağı olan Jen-Hsun Huang tarafından Derin Öğrenme konusunda yine ilginç bir sunum yapılmıştır. Jen-Hsun Huang bilindik bir CEO'dan ziyade sadece firmanın ürettiği ürünleri değil ürünlerin kullanıldığı alanlarda ihtiyaç duyulan akademik bilgiyi bir akademisyen gibi izleyiciye aktarmıştır. Jen-Hsun Huang bu konuşmasında yeni nesil GPU TITAN X'in yanında derin öğrenmenin işleyişi ve bu kapsamda geliştirdikleri NVIDIA DIGITS ara yüzünü tanıtmıştır. [11]

4.2.4. Flask

Flask Python ile yazılmış ve Werkzeug web tool'u ve Jinja2 template sistemine dayalı bir mikro web alt yapısıdır. Flask, belirli bir araç veya kütüphane gerektirmeyen mikro bir alt yapı olarak adlandırılır. Veritabanı soyutlama katmanı, form doğrulama veya önceden var olan üçüncü parti kütüphanelerin ortak işlevler sağlayan diğer bileşenleri yoktur.

4.3. Mobil Uygulama

Mobil uygulama rakam öğrenme sisteminin istemci kısmı için geliştirilen bir uygulamadır. Daha önceki bölümlerde açıklamış olduğumuz iOS işletim sistemi üzerinde çalışan bir telefon ve tablet uygulamasıdır. Kullanıcılar bu uygulama üzerinden yazı yazar gibi ekran üzerinde çizim yaparak rakamları çizmektedir. Çizdikleri bu rakamları servise göndererek servisten çizilen şeklin doğruluk oranı ile birlikte cevabı gelmektedir.

Derin öğrenme servisinden gelen cevaplar, sesli komut olarak kullanıcılara okunmaktadır. Ekran üzerinde verilen ilk cevabın yanlış olma olasılığı için kullanıcıya diğer seçeneği dinlemek istediği sorulacak ve kullanıcı onaylarsa ikinci doğruluk oranındaki cevap okunmaktadır.

Rakamların çizimi parmak hareketlerinin takibi yapılarak gerçekleştirilmektedir. Kullanıcılar çizimi isterlerse telefon ve tabletlerine uygun kalemler ile de yapabilirler. Günümüzün artık kendini kabul ettirmiş teknolojisi olan dokunmatik ekran teknolojisi sayesinde öğrenciler için kağıt, kalem harcaması yapmadan sınırsız adet deneme imkanı mobil uygulamanın en büyük kazanımlarından biridir.

Kullanıcıların yeni çizimler yapmak ve eski çizimini silmeleri ortalama 5 sn. sürmektedir. Tüm bu işlemler iki farklı butona bağlanarak, kullanıcı deneyimi (User Experience

– UX) ön planda tutulmuştur. UX'e uygun geliştirilen mobil uygulama ile kullanıcılar daha rahat ve kolay bir mobil deneyim yaşayarak uygulamayı daha efektif olarak kullanmaktadır.

Mobil uygulama içerisinde bulunan menü ve butonlar ile kullanıcılar uygulamayı farklı şekillerde kullanabilmektedir. Uygulama içerisinde iki farklı çizim modu, dört farklı renk seçeneği bulunmaktadır. Bunun dışında kullanıcı uygulama içerisinde kendi el yazısı ile değil de farklı bir ortamdaki yazıyı da uygulamaya tanıtabilmek için fotoğraf özelliğini kullanabilir. Mobil uygulama ile çekilen fotoğraf rakam öğrenme servisine gönderilerek analizi yapılır. Analiz sonucunda rakam öğrenme servisi fotoğraf ile ilgili bilgiyi mobil uygulamaya gönderir. Buradan sonrası normal el yazısı gibi çalışır. Mobil uygulama yine cevabı kullanıcılara sesli olarak söyler.

4.4. Web Uygulaması

Web uygulaması tüm derin öğrenme işlemlerinin gerçekleştirildiği uygulamadır. Rakam öğrenme sistemimizin merkezi ve en önemli parçasıdır. Rakamların tahmininden önce tüm derin öğrenme işlemleri de bu uygulama ile gerçekleştirilir. Sistemin rakamları öğrenmesi için eğitimi, öğrendiği rakamların test edilmesi ve daha sonra mobil uygulama üzerinden gelen rakam sorularına cevap vermesi web uygulaması ile sağlanır.

Daha önceki bölümlerimizde açıkladığımız gibi derin öğrenme işlemlerini gerçekleştirebilmek için Berkeley Üniversitesinde geliştirilmekte olan Caffe adlı derin öğrenme alt yapısı kullanılmıştır. Bu alt yapının en önemli özelliklerinden biri olarak da GPU üzerinde çalışabiliyor olmasıdır. Derin öğrenmenin GPU'lar üzerinde daha başarılı ve performanslı olduğu, işlem bütününe bakıldığında temelde görüntü üzerinden işlem yapıldığından GPU'ların donanım olarak CPU'lara fark atacağı kaçınılmazdır.

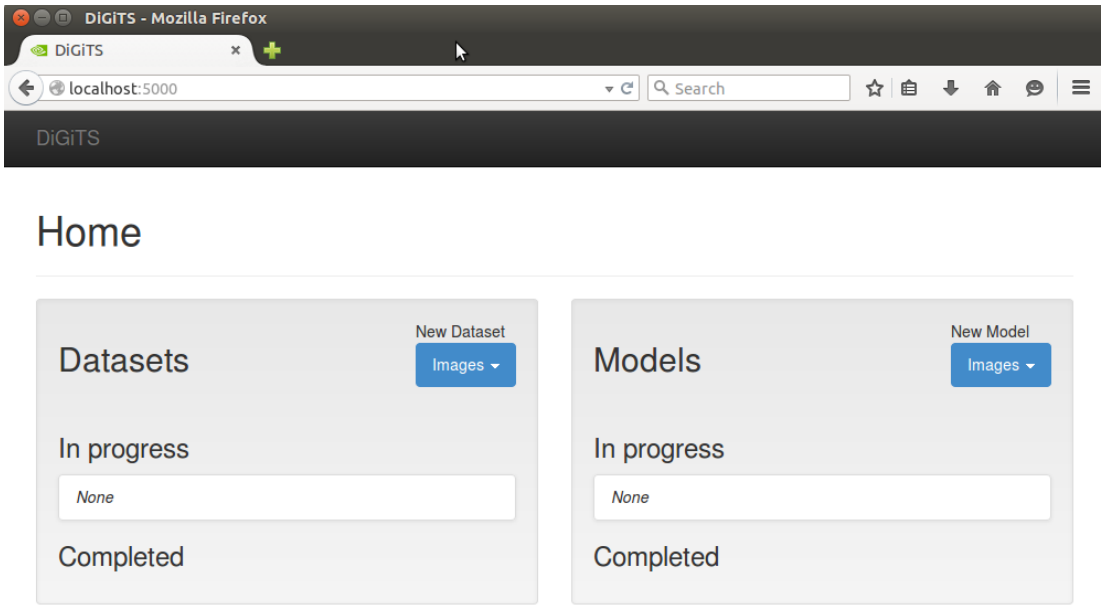
Derin öğrenme ve yapay zekâ çalışmalarında GPU'ların önemi arttığından belli NVDIA isimli şirket bu alana büyük yatırımlar yapmaktadır. Çıkardığı donanımsal ürünlerin yanında

yazılım ürünleri de piyasaya sürmektedir. NVDIA derin öğrenme ve akademik çalışmalara destek olmak amacı ile DIGITS isimli bir yazılımı ücretsiz olarak kullanıma sunmaktadır. Ücretsiz kullanım dışında bu uygulamayı sürekli olarak geliştirmekte ve insanlara dağıtımına, güncellemelere devam etmektedir. DIGITS yazılımından önceki konularda bahsetmiştik. Bu tezin ve akademik çalışmanın web uygulaması tarafında DIGITS kullanılmıştır. Sistemimiz derin öğrenme için Caffe, Caffe'nin son kullanıcı ile etkileşimi ve web uygulaması için DIGITS alt yapılarını kullanmaktadır. DIGITS bir derin öğrenme alt yapısı değildir, sadece bir ara yüzdür. Caffe üzerinde kod ile uzun sürede yapılacak işlemlerinin ara yüzler üzerinden daha kısa sürede yapılmasını sağlar.

Web uygulaması Flask uygulama sunucusu üzerinde Python dili ile geliştirilmiş olan DIGITS ile çalışmaktadır. Sistemimiz için belli başlı özelleştirmeler ekstra python kodu yazarak sağlanmıştır. Kodlarımız Aws üzerindeki Ubuntu işletim sisteminde çalışmaktadır. İşletim sistemimizin özellikleri aşağıdaki gibidir;

- Aws Instance Type : g2.2xlarge
- 8 çekirdek Gpu
- 15 Gb Bellek
- 60 GB SSD
- Intel Xeon E5-2670 işlemci

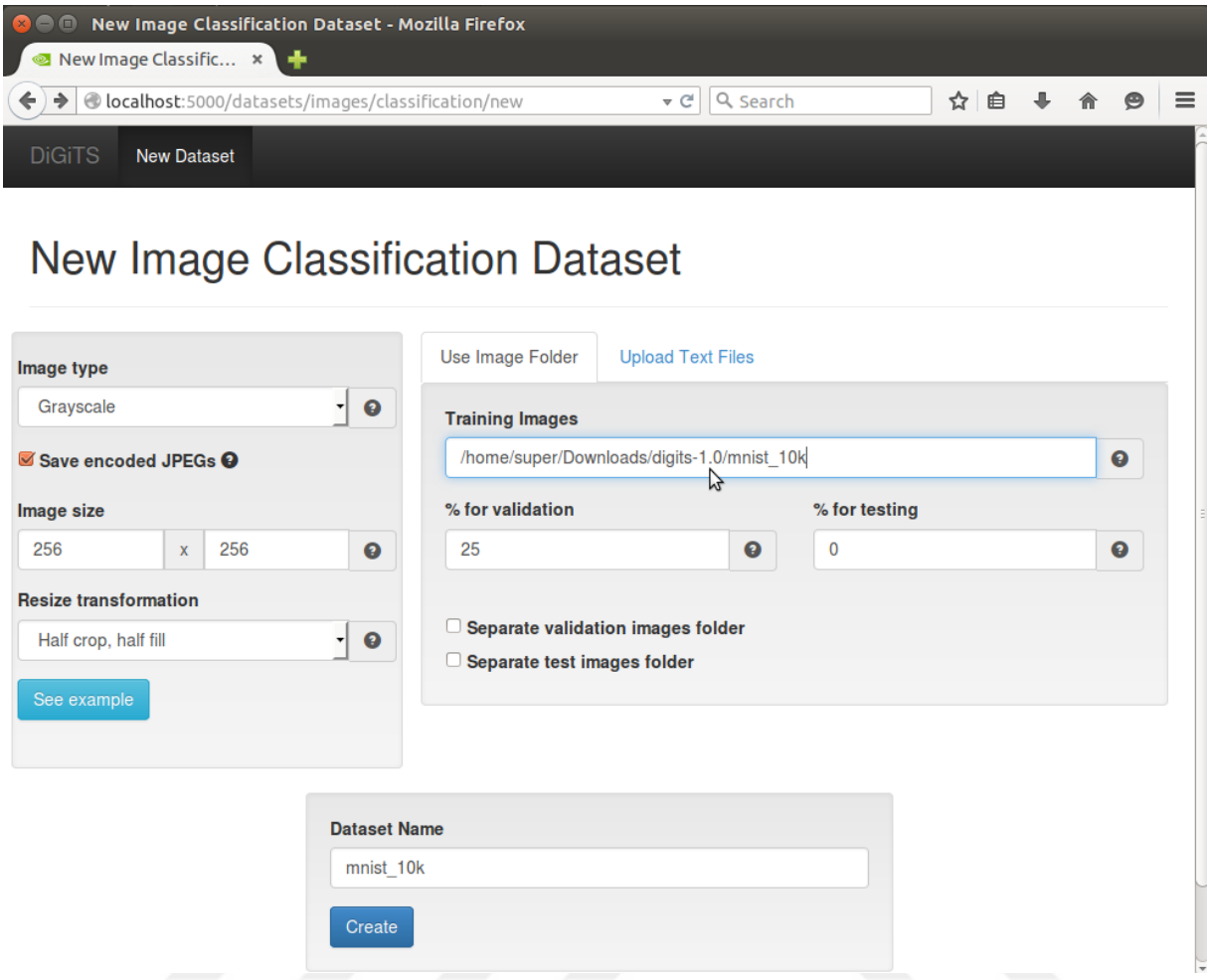
Sunucuya önce Berkeley üniversitesinin proje sayfasından kaynak kodlar ücretsiz olarak indirildikten sonra Caffe kurulumu gerçekleştirilmiştir. Caffe kurulduktan sonra GPU modda yapılacak çalışmalar ve DIGITS için nvdian'ın CUDA driver'larının kurulumu gerçekleştirilmiştir. CUDA driver'ları indirildikten sonra DIGITS yazılımının kurulum dosyaları indirilip sunucu üzerine kurulmuştur. Digits kurulumu da tamamlandıktan sonra sunucuya ssh üzerinden bağlanılmaktadır. Ssh üzerinden ./runme.sh dosyası çalıştırılarak DIGITS başlatılmaktadır. DIGITS başlatıldıktan sonra browser üzerinden Şekil 5'deki ekran açılmaktadır. Bu ekran başarı ile açıldığında tüm kurulumlar sorunsuz olarak gerçekleştirilmiş olur.



Şekil.5. Digits kurulum sonrası ilk açılış ekranı

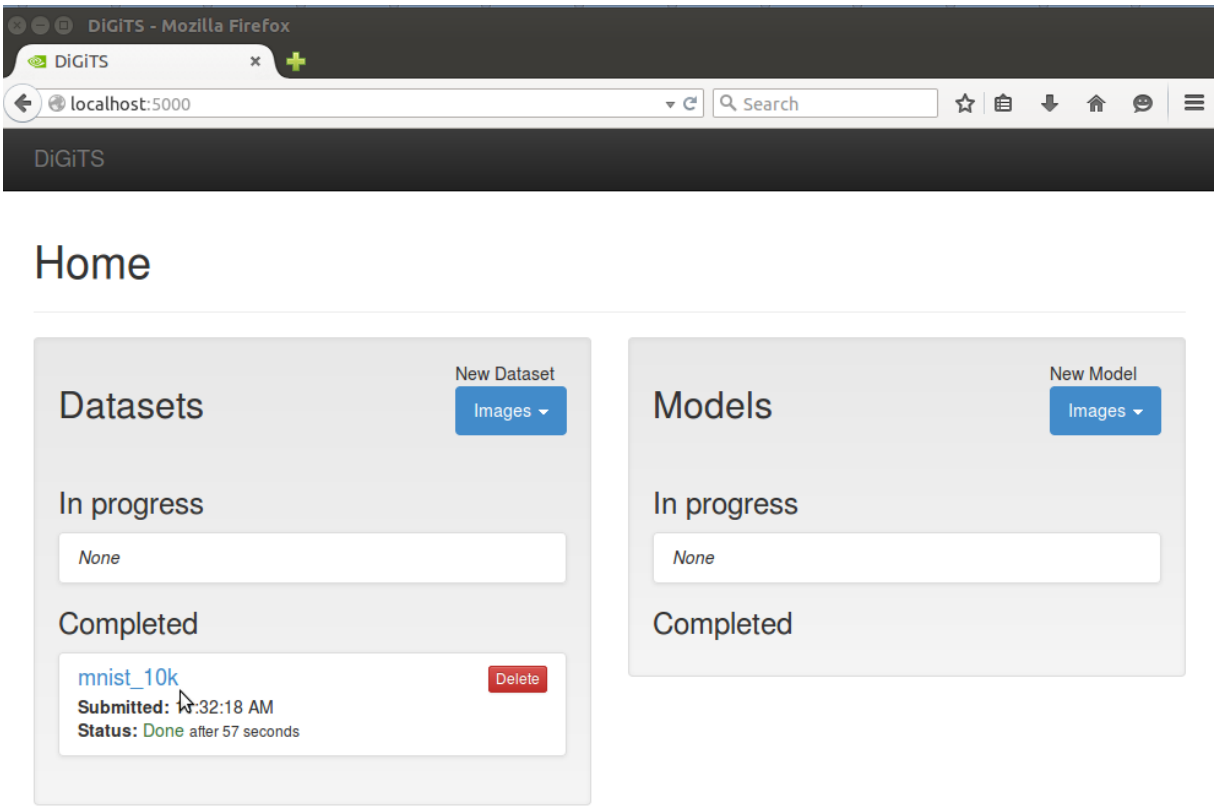
Derin öğrenme sistemimiz için öncelikle Veri seti oluşturulmalıdır. Veri seti tüm öğrenme işlemimizin yapıldığı settir. Caffe ile önce bu set eğitilmekte, daha sonra test edilerek öğrenmenin başarısını ölçülmektedir. Rakam öğrenme sistemde eğitim için 60.000 resimden oluşan bir veri seti kullanılmıştır. Test işlemleri için de 10.000 resimlik bir veri seti kullanılmıştır. Tüm bu veri setleri ile 3.000 iterasyonluk bir eğitim yapılmıştır. Eğitim sonucunda %99 doğruluk oranı yakalanmış; toplam hata değeri ise 0,005 çıkmıştır. Caffe ve Digits ile veri seti oluşturulmak için yapılması gereken adımlar aşağıdaki gibidir;

Digits uygulamamızın sol tarafında bulunan Datasets bölümünde, New Dataset altında Images butonu ile Classification seçeneği seçilir. Bu seçimden sonra New Image Classfication Dataset sayfasına otomatik olarak sistem yönlendirmesi yapılır. El yazısını tanımak için kullanacağımız MNIST isimli el yazısı rakam veritabanı örneği ile işlemlere devam edilir. Veritabanın durduğu fiziksel yol bilgileri de digits'e verildikten sonra ilk veri setimiz oluşturulur. Tüm bu işlemlerin yapıldığı ve alanların doldurulduğu ekran görüntüsü Şekil 6.'da gösterilmiştir.



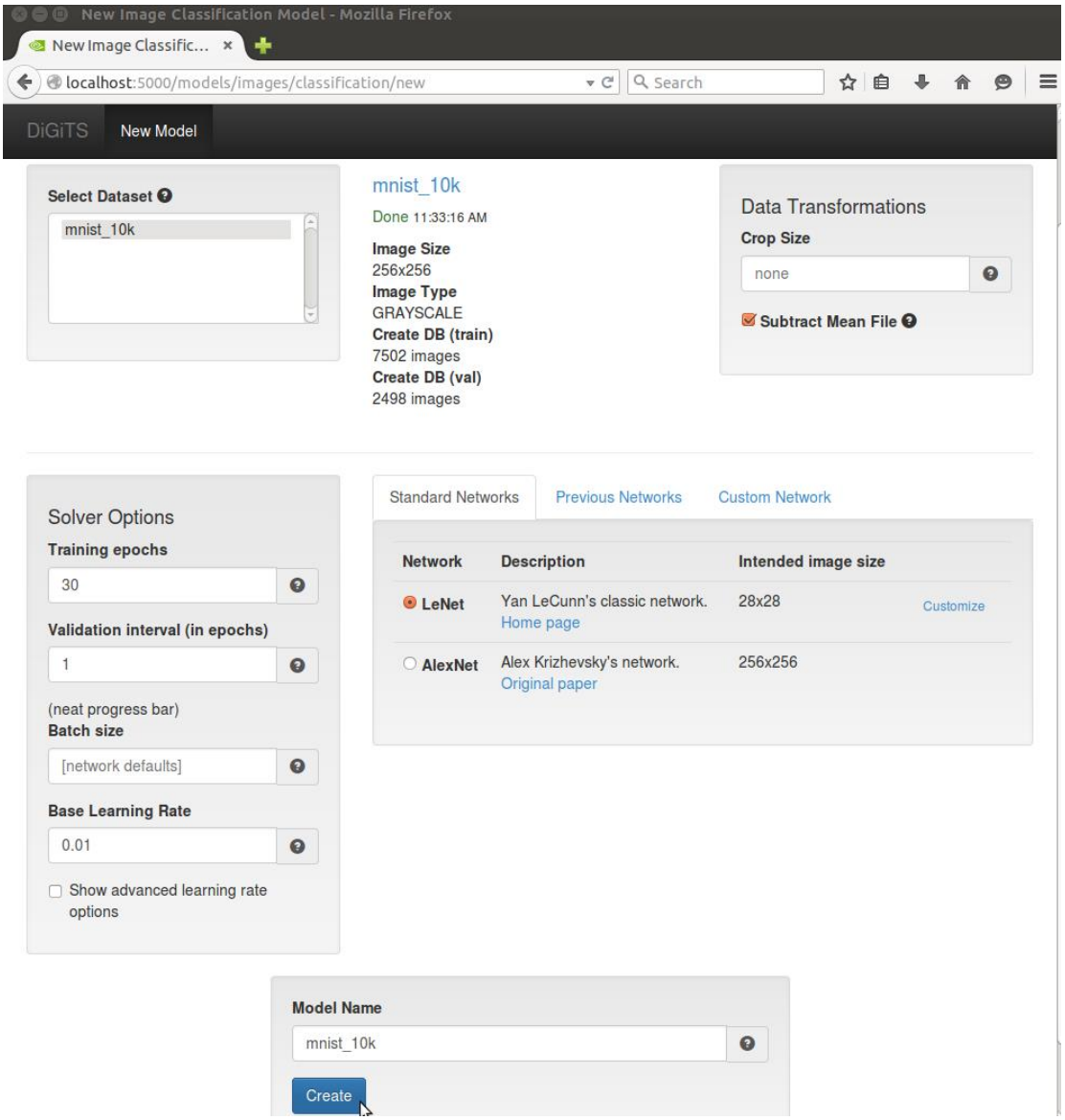
Şekil.6. Veri seti oluşturma

Veri seti oluşturduktan sonra Ana sayfaya geri döndüğümüz Şekil 7.'de gözüktüğü gibi yeni veri setimizi ekran üzerinde görürüz.



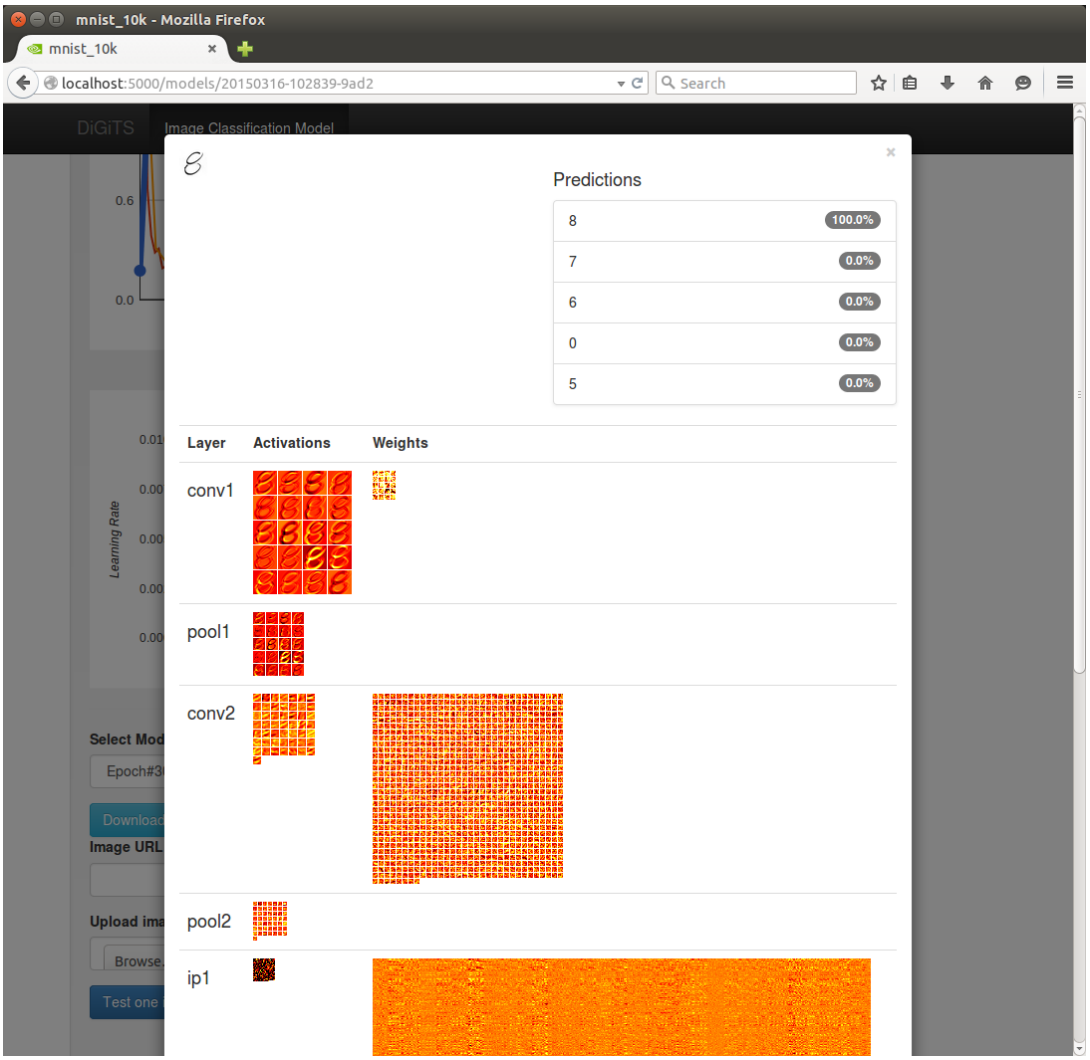
Şekil.7. Veri seti

Veri setini oluşturduktan sonra, veri setini eğitmemiz gerekir. Veri setini eğitmek için ana sayfadan veri setine tıklayarak detayına girilir. Burada yeni veri seti modeli oluşturulur. Ekran üzerinde New Image Classification Model adı ile bir buton bulunur. Bu butona tıklayarak alanlar doldurulduktan sonra eğitilmiş veri seti modelimiz oluşturulmuş olunur. İlgili tüm alanlar ve veri seti modeli oluşturma Şekil 8.'de gösterilmiştir.



Şekil.8. Veri seti modeli oluşturma

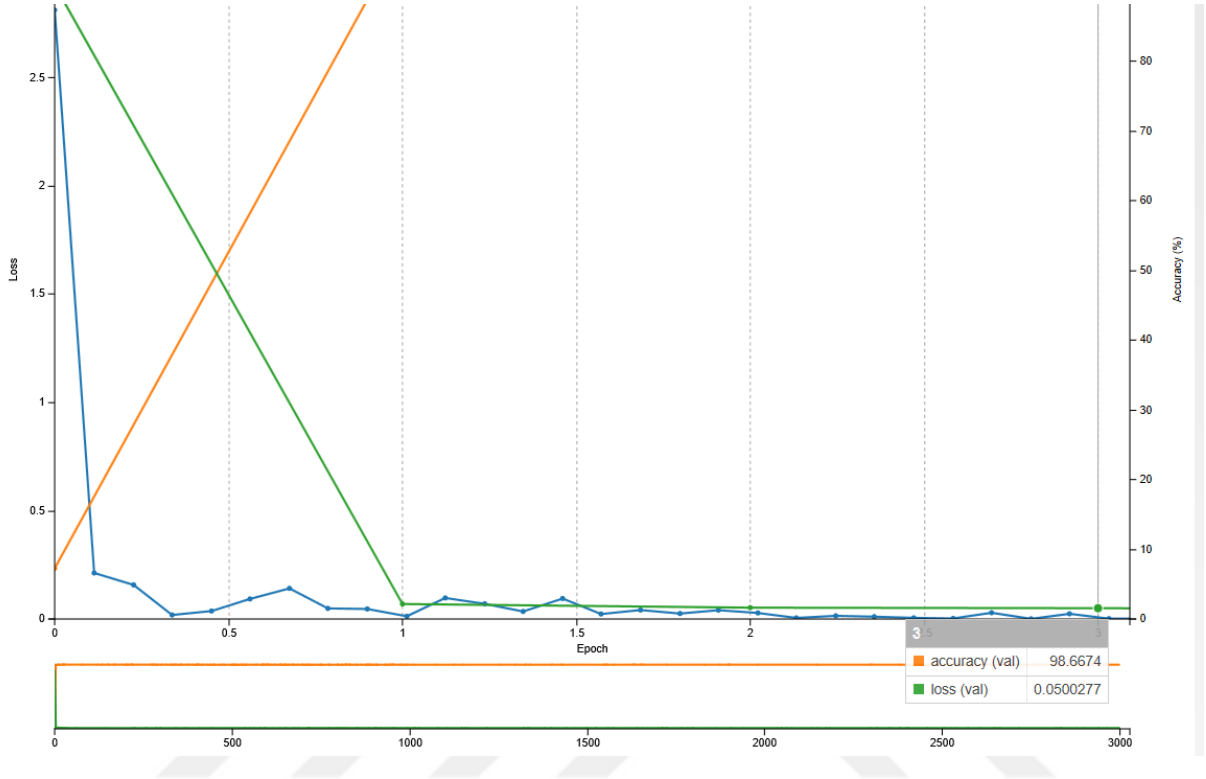
Yaptığımız işlemler sonucunda Digits ile veri setimizi oluşturup eğiterek ilk veri seti modelimizi de oluşturmuş olduk. Böylece mnist el yazısı veritabanını kullanarak Caffe alt yapımıza öğrenmesi için bilgiler sunduk. Eğitimi tamamladıktan sonra sırada sistemi test ederek, eğitimin başarısını ölçmek var. Test için web uygulamasını kullanarak, el yazısı ile oluşturulmuş görüntüler sisteme yükleyerek sistemin başarısı ölçülür. Sisteme sekiz rakamının görüntüsünü yüklediğimizde elde ettiğimiz sonuç Şekil 9. da gösterilmiştir. Sistem %100 doğruluk oranı ile sekiz cevabını vermiştir.



Şekil.9. Veri setinin test edilmesi

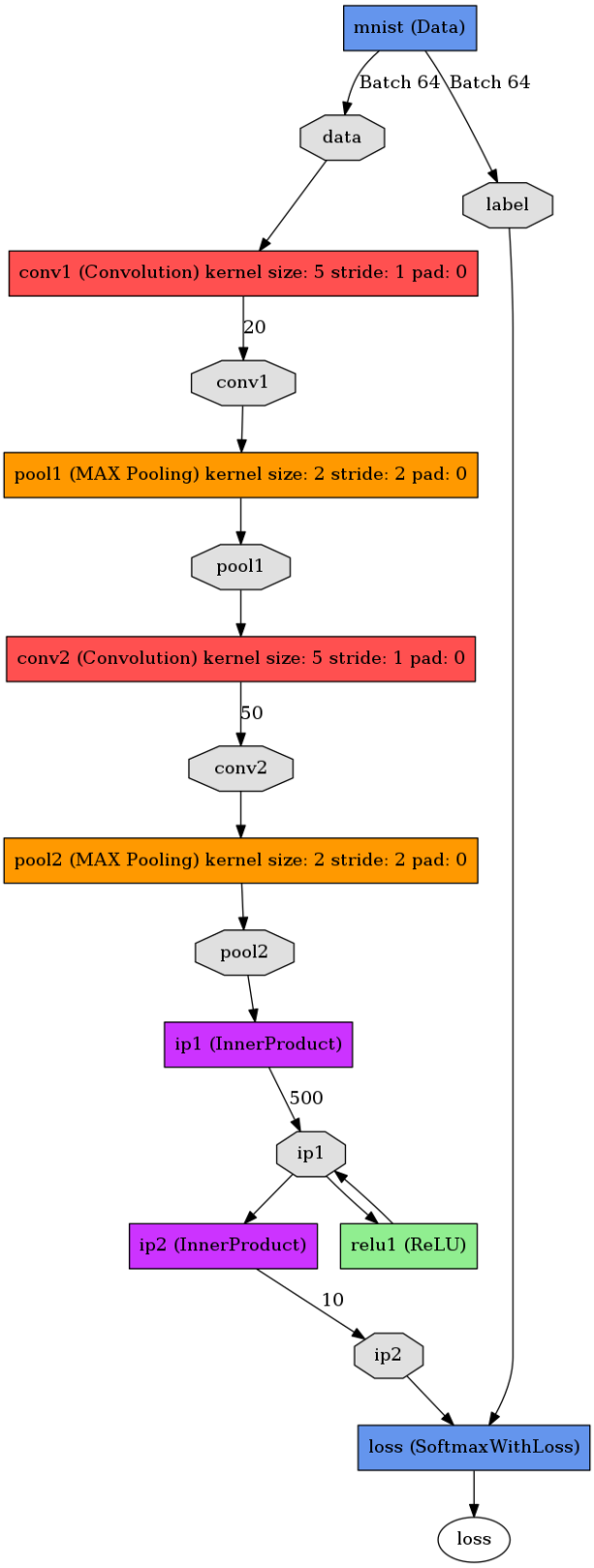
Yaptığımız geliştirmeler sonucunda web uygulamamız ve sistemimiz artık tüm rakamları öğrenmiştir. Sistemimize yapılacak tüm isteklere doğruluk oranları ile birlikte beş sonuç doğruluk oranına göre sıralanarak cevap olarak verilmektedir.

Sistemin öğrenme ve test sonrası iterasyonlara göre doğruluk oranı ve kaybı Şekil 10. da gösterilmiştir.



Şekil.10. İterasyonlara göre doğruluk oranları

Bu sonuçları elde ettiğimiz solver-model yapımız da Şekil 11. de gösterilmiştir.



Şekil.11. Model yapısı

5. RAKAM ÖĞRENME UYGULAMASI KULLANIMI

5.1. Giriş

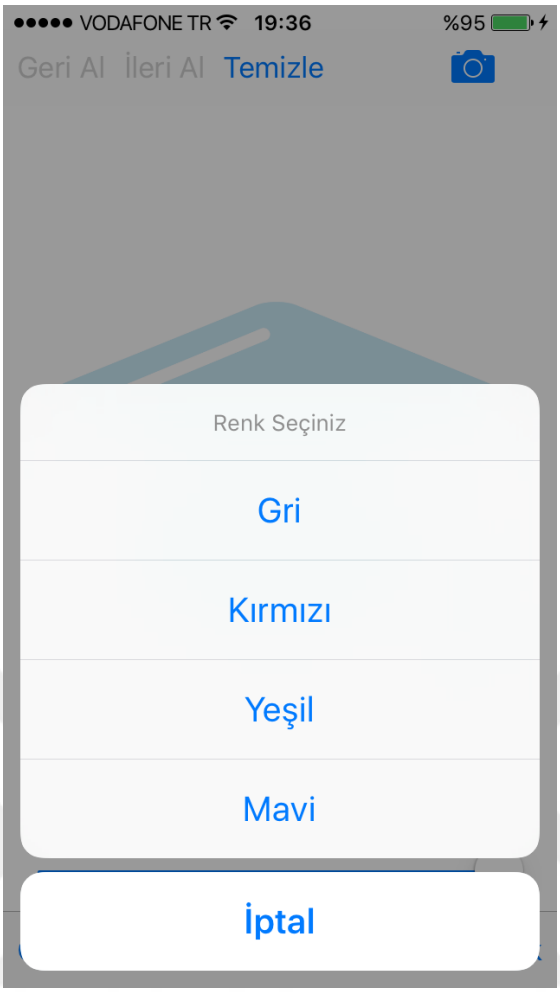
Bu bölümde rakam öğrenme sisteminin nasıl kullanıldığı ekran görüntüleri ile desteklenerek adım adım anlatılacaktır. Kullanıcının el yazısı ile rakam yazması, derin öğrenmeden kullanıcıya cevabın dönülmesi, kullanıcının yaptığı işlemlerin takip edilmesi ve web üzerinden derin öğrenme testlerinin yapılması bu bölümün konuları oluşturmaktadır.

5.2. Kullanıcı Mobil Kullanımı

Sistemin mobil kullanıcılarının iOS telefon veya tablet sahibi kişiler olduğu kabul edilmiştir. Kullanıcılar, sistemin iOS platformlar için geliştirilmiş olan uygulamasını kullanarak derin öğrenme uygulamasına el yazılarının sonuçlarını soran kişilerdir.

Kullanıcılar bu uygulamayı App Store üzerinden kendi cihazlarına kurduktan sonra sistemi kullanmaları için uygulamayı açmaları yeterlidir. Açılış ekranı uygulamanın logosu ile birlikte adının gösterildiği ekrandır. Açılış ekranından sonra kullanıcı otomatik olarak ana ekrana yönlendirilir. Kullanıcı bu ekran üzerinde el yazısı yazması ile birlikte, bu yazı işlemlerini konfigürasyonunu da bu ekran üzerinde yapmaktadır.

Ana ekranın sol alt tarafında Gri yazan butona tıklayarak renk seçimi menüsüne ulaşılır. Açılan menüden isteye bağlı olarak renk seçimi yapılır. Renk menüsünden yapılan seçime göre çizimin rengi değişir. Renk menüsü Şekil 12. de görülmektedir.

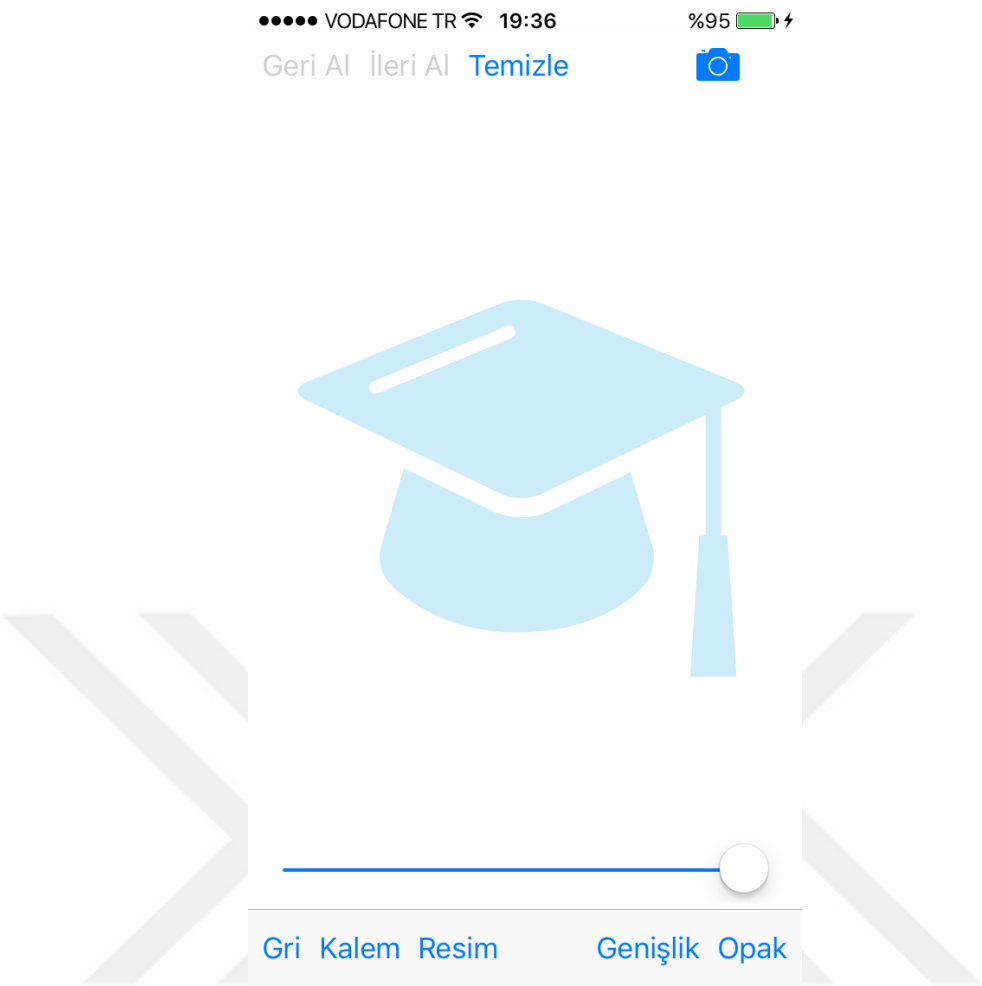


Şekil.12. Renk menüsü

Ekranın altında bulunan menüdeki bir diğer buton ile çizim aracı seçimi yapılır. Ana ekranda Kalem yazan butona tıklandığında açılır. Kalem ve çizgi şeklinde iki seçenek bulunur. Menüden yapılan seçime göre seçim şekli değişir. Kalem çizim aracı ile gerçek hayattaki kalem kullanımı sunulur. Çizgi çizim aracı ile yapılan çizimler doğrusal olarak gerçekleştirilir. Şekil 13.'de Çizim aracı menüsü gösterilmiştir.

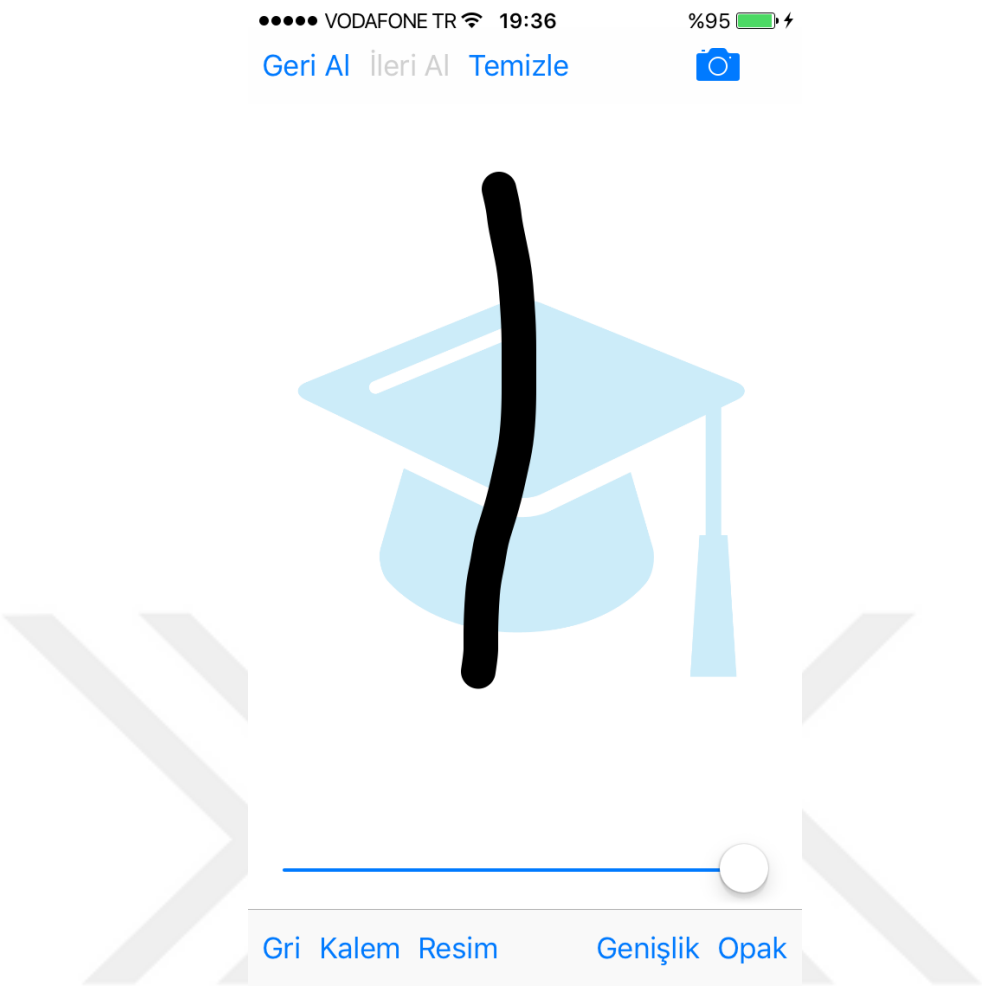
Şekil.13. Çizim aracı menüsü

Ekranın altında bulunan son menüler, Genişlik ve Opak menüleridir. Bu menüler vasıtası ile yapılan çizimlerin genişliği ve saydamlığı bir bar vasıtası ile ayarlanır. Barın sol tarafı değerleri azaltırken, sağ tarafı değerleri yükseltir. Daha kalın bir çizgi istediğimizde genişlik barını sağa doğru, daha saydam bir çizgi istediğimizde ise opak barını sola doğru kaydırmamız yeterlidir. Kullanıcılar kendi isteklerine göre bu menüler vasıtası ile çizimlerini özelleştirebilirler. Bu menüler şekil 14.'de gösterilmiştir.



Şekil.14. Genişlik barı

Ana ekranın ortasındaki boş alan çizim yapılması için ayrılmıştır. Kullanıcılar bu alan üzerine elleri ile kalem kullanır gibi çizim yaparak rakam yazarlar. Şekil 15 örnek bir rakam yazması gösterilmiştir.



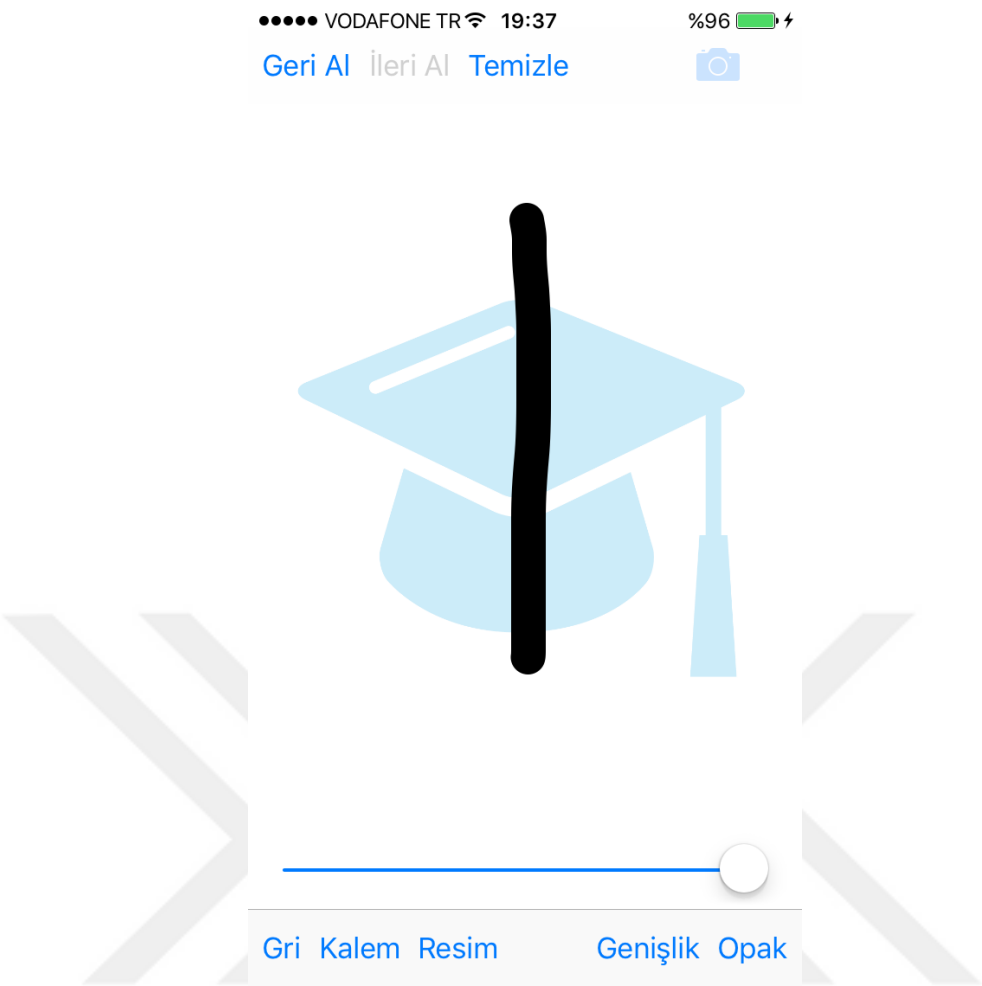
Şekil.15. Örnek rakam çizimi

Kullanıcılar bu alan üzerinde sınırsız sayıda çizim yapabilirler. Herhangi bir kısıtlama yoktur. İnsanlar yazı yazarken bile hata yaparlar ve bu yaptıkları hataları düzeltmek için silgi kullanırlar. Uygulama geliştirilirken bu tarz hata durumları göz önüne alınmıştır. Hata durumlarını yönetmek için yapılan her çizim hareketi tek tek kayıt altına alınarak, hata olması durumunda geriye dönülmesi ve bir önceki çizim anına gelinmesi sağlanmıştır. Bazı durumlarda kullanıcıların eski duruma döndükten sonra tekrar yeni duruma gelmesini isteyebilecekleri için geri dönüşlerin yanında ileri dönüşlerde tasarlanmış ve geliştirilmiştir.

Uygulamamızın üst kısmında bulunan Geri Al ve İleri Al menüleri sayesinde tüm bu çizimler arasındaki değişimler gerçekleştirilir. Çizimi yaptıktan sonra beğenmediğimiz her hareket bu menüler ile en doğru çizim yapılana kadar kullanılır. Şekil 16. da gösterildiği gibi İleri Al butonu çizim eski haline getirilerek tamamlanır.

Şekil.16. İleri Al menüsü

Yaptığımız çizimin sonucu öğrenmek için, çizimi bitirdikten sonra ekranın sağ üst tarafında bulunan fotoğraf makinesi simgeli menü yardımı ile çizimizi rakam öğrenme web uygulamasını göndermiş oluruz. Çizim uygulamaya eriştikten sonra işlenmeye başlanır. İşlemenin sonucunda web uygulaması, mobil uygulamaya cevap dönülür. Web uygulaması ile iletişimi sağlayan menü şekil 17. de gösterilmiştir.



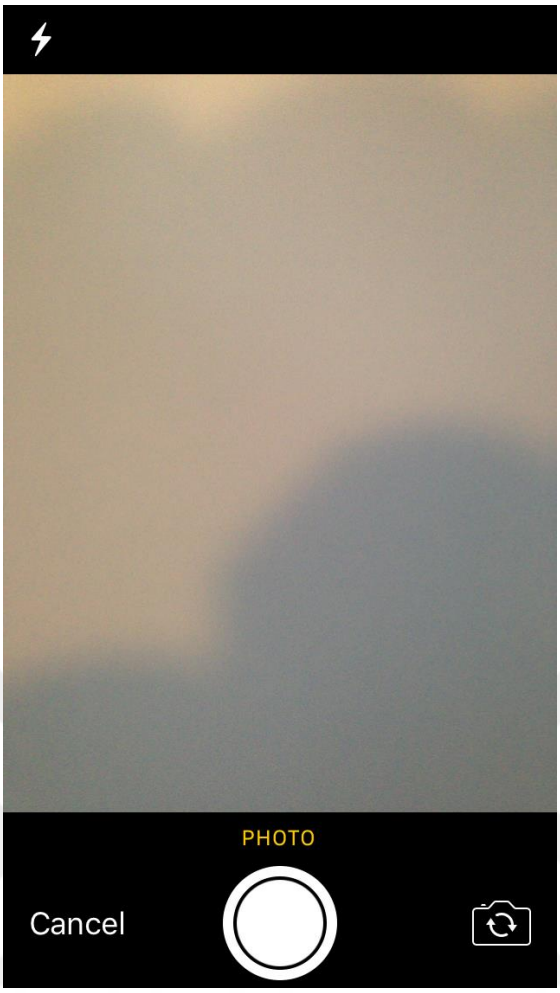
Şekil.17. Web uygulaması iletişim menüsü

Web uygulamasından cevap geldiğinde mobil uygulama gelen cevabı sesli bir şekilde kullanıcılara bildirir. Verilen bilginin yanlış olma olasılığı düşünülerek ikinci seçeneğin gösterilmesi kullanıcıya sorulur. Kullanıcının Evet cevabı vermesi sonucu diğer seçenek kullanıcılara okunur. Kullanıcıya sorulan seçenek ekranı şekil 18. de gösterilmiştir.

Şekil.18. Diğer seçenek sorusu ekranı

Mobil kullanıcılar cevapları da duyduktan sonra tüm işlemleri yeniden yapmak veya çizim ekranını sıfırlamak için yukarıda bulunan Temizle menüsünü kullanırlar. Temizle menüsü kullanıldığında ekrandaki tüm çizimler geri alınır ve ekran temizlenir.

Kullanıcılar kalem ile kâğıda yazdıkları rakamları da rakam öğrenme sistemine gönderebilirler. Bunun için ekranın altında bulunan Resim menüsünü kullanabilirler. Resim menüsüne tıklandığında iOS tablet veya telefonun kamerası açılır. Açılan kamera ekranında sisteme gönderilmek istenen rakamın fotoğrafı çekilir. Şekil 19. da bu kamera ekranı gösterilmiştir.



Şekil.19. Kamera ekranı

İstenilen görüntü yakalandığında mobil uygulama otomatik olarak çekilen fotoğrafı sisteme gönderir. Sistem üzerinden cevap gelir ve kullanıcıya okunur. Buradan sonraki tüm işlemler çizim yapılırken oluşan adımlar ile aynıdır. Kullanıcı isterse diğer seçeneği dinleyebilir veya temizle yaparak tüm işlemleri yeniden yapabilir. Yeni bir çizim yada yeni bir fotoğraf yükleyerek sisteme gönderebilir.

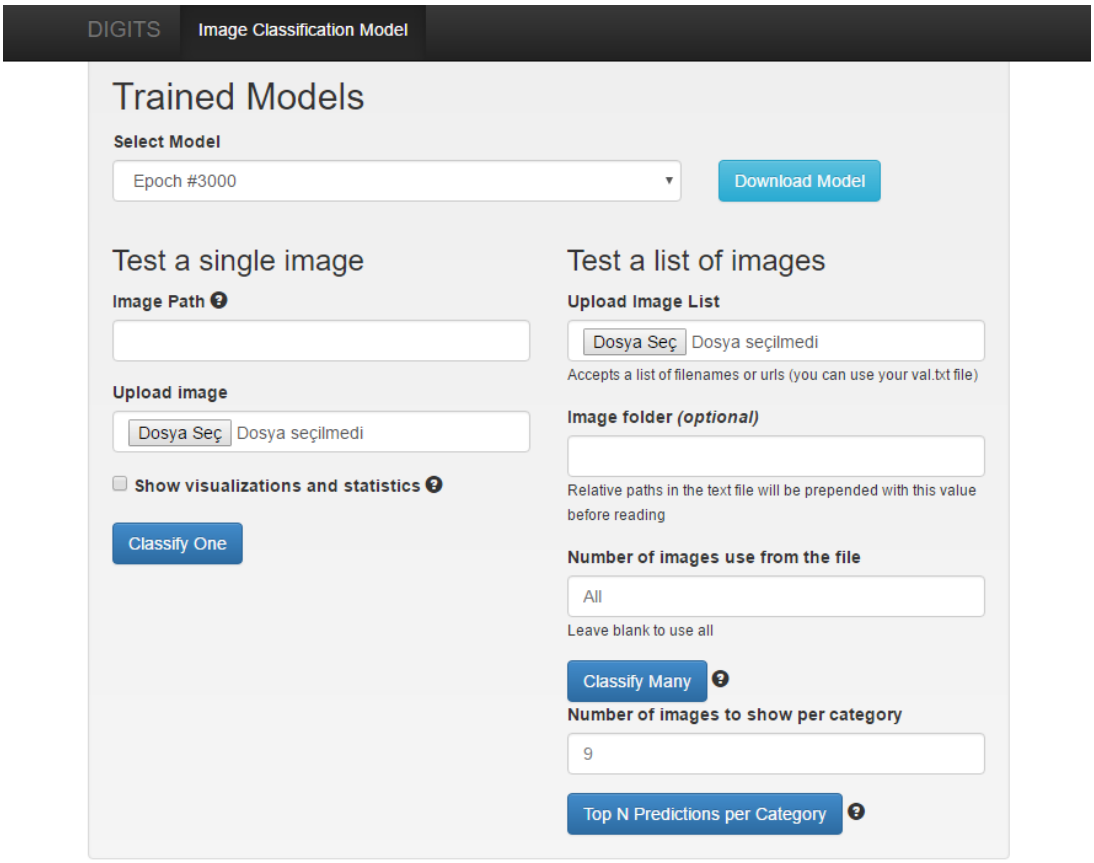
Mobil uygulamanın nasıl kullanılacağı ile ilgili bölümün sonuna geldik. Bu bölümde uygulamanın tüm ekranları ve menüleri gösterilmiştir. Ayrıca uygulamanın kullanım adımları gösterilmiş ve yaşam döngüsü anlatılmıştır.

5.3. Kullanıcı Web Kullanımı

Web uygulaması iki yönlü çalışmaktadır. İlk olarak mobil uygulamadan gelen istekleri karşılayan ve cevap dönen bir hizmet sağlayıcı servis olarak çalışmaktadır. İkinci olarak web üzerinden veri seti oluşturma, eğitme, test etme ve mobil uygulamadan gelen istekleri takip etme şeklinde hizmet verici olarak çalışmaktadır. Birinci görevi diğer bölümlerde anlatılmıştır. Bu bölümün konusu web uygulamasının ikinci görevleri olacaktır.

Web uygulamasında veri seti oluşturma, eğitme konuları da digits konusu anlatılırken bahsedilmişti. O bölümdeki konular bu bölüm için alt yapı olacağından önem arz etmektedir. Eğitmiş olduğumuz veri setimizi test etmek için mobil uygulamayı kullanabileceğimiz gibi web uygulamamızı da kullanabiliriz.

Uygulamamızı test etmek ve derin öğrenmemizi sınamak için web uygulaması üzerinden tekli veya çoklu görsel yükleyerek sınamalar yapabiliriz. Çoklu ve tekli görsel sınamanın dışında, web uygulamamıza görselleri bir link üzerinden, sunucu üzerindeki bir klasör üzerinden veya görsel yükleme metodu ile yapabiliriz. Üç metot ile de uygulamamız aynı görseli alacak ve işleme sokmaktadır. İşlem sonucunda bu görseller ile ilgili beş farklı sonuç sunmaktadır. Bu sonuçlar doğruluk oranına göre sıralanarak belirtilmektedir. Şekil 20. de web uygulaması üzerinden görsel test etme işlemi gösterilmiştir. Şekilde tüm görsel yükleme seçenekleri gösterilmiştir.



Şekil.20. Web uygulaması görsel yükleme ve test etme

Web uygulaması üzerinden sınırsız sayıda görsel test edilebilir.

6. SONUÇ

Yapay zekâya yönelik çalışmalar çok eski senelere dayanmaktadır. Bu çalışma alanına yönelik popülerlik her zaman üst düzeyde olmamıştır. Zamanla ilgi artmış, ilgi artışını istenilen seviyeye ulaşamayan çalışmalar ilgisizliğe dönüştürmüştür. Son zamanlarda yapay zekâ çalışmaları yeniden durağanlığa girmişti. Derin öğrenme felsefesi ile birlikte yapay zekâ alanında tekrardan bir popülerlik ve çalışma isteği artışı gözlemlenmeye başlanmıştır.

Derin öğrenme sayesinde bilim insanları, akademik çalışanlar ve diğer tüm geliştiriciler tekrardan yapay zekâ ile ilgilenmeye başlamıştır. Bu artış beraberinde başarılı çalışmaları da getirmiştir. Google, Facebook, IBM, Microsoft, Apple gibi büyük şirketler derin öğrenme alanına yaptıkları yatırımların meyvelerini toplamaya başlamıştır. Tüm bu yatırımlar, gelişmeler derin öğrenme felsefisinde kullanılmak üzere bir çok alt yapı geliştirilmesine yol açmıştır. Tensorflow, Keras, Cntk, Theano, Torch, Chainer ve bu akademik çalışmanın konularından biri olan Caffe bu geliştirmelerin bir kısmıdır.

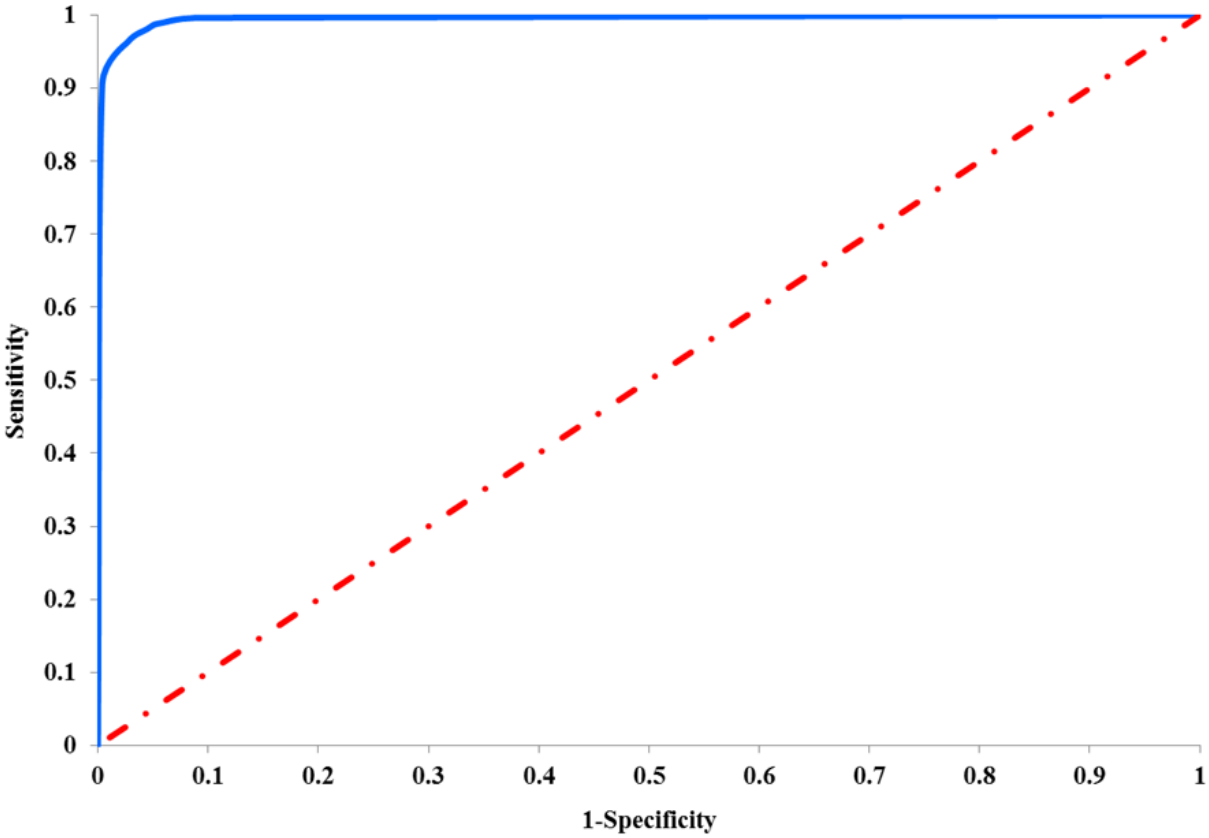
Bu tez çalışması Caffe üzerine kurulu bir derin öğrenme çalışmasıdır. Caffe ile rakamlar sisteme öğretilmiş ve mobil uygulama ile haberleşen bir sistem tasarlanmıştır. Bu sayede kullanıcılara her türlü çizimlerine karşılık bir cevap verilerek rakamların öğretilmesi amaçlanmıştır. Kullanıcılar olarak hedef kitle anaokulu öğrencileri seçilmiştir. Rakam öğrenme ile derin öğrenmenin eğitime uyarlanması amaçlanmıştır. Literatür çalışmalarını taradığımda el yazısı tanıma fikri ile benzer olarak bankacılık sektöründe uygulamalar geliştirilmiştir. Çek ve senet doğrulamada bu yöntemler kullanılmaktadır.

Bu tez çalışması kapsamında geliştirilen Rakam Öğrenme sistemi ile derin öğrenme ve yapay zekânın eğitim sektörü ile iç içe geçebileceği ve faydalı olabileceği amaçlanmış ve gösterilmiştir. İlerleyen zamanlarda yapay zekâ ve derin öğrenme, eğitime katkı sağlayacak bir çok konu ile geliştirmeler yapılabilir. Bu çalışmanın daha ileri seviyelere getirilmesi ve geliştirilmesi için bu yapının üzerine sayı öğrenme gerçekleştirilebilir. Sistem rakamlardan sonra sayılarıda öğrenebilir.

Çalışmanın doğruluğunun ispati için Şekil 21. de doğruluk tablosu ve Şekil 22. de ROC eğrisi gösterilmiştir. Çizdirilen ROC eğrisi sonucunda ROC alanı 0,96 olarak bulunmuştur. Bu değer sistemin geçerliliğini ortaya koymuştur.

	Path	Ground truth	Top predictions			
1	/home/saman/data/data/AD_019_S_4252_2012-04-24_021_001_001.png	AD	AD	79.72%	NC	20.28%
2	/home/saman/data/data/AD_019_S_4252_2012-04-24_021_001_002.png	AD	AD	94.67%	NC	5.33%
3	/home/saman/data/data/AD_019_S_4252_2012-04-24_021_001_003.png	AD	AD	91.96%	NC	8.04%
4	/home/saman/data/data/AD_019_S_4252_2012-04-24_021_001_004.png	AD	AD	81.63%	NC	18.37%
5	/home/saman/data/data/AD_019_S_4252_2012-04-24_021_001_005.png	AD	AD	89.44%	NC	10.56%
6	/home/saman/data/data/AD_019_S_4252_2012-04-24_021_001_006.png	AD	AD	90.56%	NC	9.44%
7	/home/saman/data/data/AD_019_S_4252_2012-04-24_021_001_007.png	AD	AD	85.15%	NC	14.85%
8	/home/saman/data/data/AD_019_S_4252_2012-04-24_021_001_008.png	AD	AD	93.94%	NC	6.06%
9	/home/saman/data/data/AD_019_S_4252_2012-04-24_021_001_009.png	AD	AD	89.53%	NC	10.47%
10	/home/saman/data/data/AD_019_S_4252_2012-04-24_021_001_010.png	AD	AD	92.96%	NC	7.04%

Şekil.21. Doğruluk oranları



Şekil.22. ROC eğrisi

7. KAYNAKLAR

[1] Atınç Yılmaz, Yapay Zeka, Kodlab Yayıncılık, 2017

[2] Birol Yıldız , Finansal Analizde Yapay Zeka, Detay Yayıncılık, 2009

[3] Ignizio J.P., Shivakumar V., A stochastic neural network for resource constrained scheduling, 1991

[4] Prof. Dr. İ. Burhan Türkşen, Dereceli (Bulanık) Sistem Modelleri, Abaküs Yayınevi, 2015

[5] Prof. Dr. Vasif Vagifoğlu Nabiyev, Yapay Zeka, Seçkin Yayınevi, 2012

[6] Emerging Technology, Google Unveils Neural Network with "Superhuman" Ability to Determine the Location of Almost Any Image, https://www.technologyreview.com/s/600889/google-unveils-neural-network-with-superhuman-ability-to-determine-the-location-of-almost/, 2016

[7] Yangqing Jia, Caffe, http://caffe.berkeleyvision.org/

[8] B. SAMANTA, K.R. AL-BALUSHI, ARTIFICIAL NEURAL NETWORK BASED FAULT DIAGNOSTICS OF ROLLING ELEMENT BEARINGS USING TIME-DOMAIN FEATURES, March 2003

[9] J.C. Hoskins, D.M. Himmelblau, Artificial neural network models of knowledge representation in chemical engineering, 1988

[10] Wikipedia, Python (programlama dili), https://tr.wikipedia.org/wiki/Python_(programlama_dili)

[11] Ferhat Kurt, NVIDIA DiGiTS, http://www.derinogrenme.com/nvidia-digits/], 2015

[12] W.G Baxt, Application of artificial neural networks to clinical medicine, 1995

[13] P. Burrascano, S. Fiori, M. Mongiardo, A review of artificial neural networks applications in microwave computer-aided design, 1999

ÖZGEÇMİŞ

28 Aralık 1989 tarihli, İstanbul İli Eminönü ilçesi doğumluyum. Lise son sınıftan itibaren yazılım geliştirmeye başladım. Yazılım geliştirme hayatımın her alanında ve anında yer buldu. Şu an uluslararası bir telekomünikasyon firmasında çalışıyorum. 2014 yılında Beykent Üniversitesi'nde yüksek lisans eğitimine başladım.

Yüksek lisans eğitimimi tamamladıktan sonra doktora eğitimime başlayıp akademik alanda çalışmalar yapmak istiyorum. Özel ilgi alanım yapay zeka teknolojileri üzerine çalışmalar yapmak.

Aday: Ali KAYA