T.C.
İSTANBUL AYDIN ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

YAPAY SİNİR AĞLARI İLE GÖRME ENGELLİLER İÇİN İLAÇ TANIMA VE SESLİ BİLGİ VERME UYGULAMASI

YÜKSEK LİSANS TEZİ

Onur YILMAZ

Bilgisayar Mühendisliği Ana Bilim Dalı

Bilgisayar Mühendisliği Programı

ŞUBAT 2020

T.C.
İSTANBUL AYDIN ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

YAPAY SİNİR AĞLARI İLE GÖRME ENGELLİLER İÇİN İLAÇ TANIMA VE SESLİ BİLGİ VERME UYGULAMASI

YÜKSEK LİSANS TEZİ

Onur YILMAZ
(Y1613.010033)

Bilgisayar Mühendisliği Ana Bilim Dalı

Bilgisayar Mühendisliği Programı

Tez Danışmanı: Dr. Öğr. Üyesi Ahmet GÜRHANLI

ŞUBAT 2020

T.C.
İSTANBUL AYDIN ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ MÜDÜRLÜĞÜ

YÜKSEK LİSANS TEZ ONAY FORMU

Bilgisayar Mühendisliği Anabilim Dalı Bilgisayar Mühendisliği Tezli Yüksek Lisans Programı Y1613.010033 numaralı öğrencisi Onur YILMAZ'ın **"Yapay Sinir Ağları ile Görme Engelliler için İlaç Tanıma ve Sesli Bilgi Verme Uygulaması"** adlı tez çalışması Enstitümüz Yönetim Kurulunun 14.01.2020 tarihli ve 2020/01 sayılı kararıyla oluşturulan jüri tarafından oybirliği/oyçokluğu ile Tezli Yüksek Lisans tezi 10.02.2020 tarihinde kabul edilmiştir.

	Unvan	Adı Soyadı	Üniversite	İmza
		ASIL ÜYELER		
Danışman	Dr. Öğr. Üyesi	Ahmet GÜRHANLI	İstanbul Aydın Üniversitesi	
1. Üye	Prof. Dr.	Ali GÜNEŞ	İstanbul Aydın Üniversitesi	
2. Üye	Dr. Öğr. Üyesi	Ali HAMİTOĞLU	İstanbul Sabahattin Zaim Üniversitesi	
		YEDEK ÜYELER		
1. Üye	Dr. Öğr. Üyesi	Adem ÖZYAVAŞ	İstanbul Aydın Üniversitesi	
2. Üye	Dr. Öğr. Üyesi	Ahmet Feyzi ATEŞ	İstinye Üniversitesi	

ONAY

Prof. Dr. Ragıp Kutay KARACA
Enstitü Müdürü

YEMİN METNİ

Yüksek Lisans tezi olarak sunduğum “Yapay Sinir Ağları ile Görme Engelliler için İlaç Tanıma ve Sesli Bilgi Verme Uygulaması” adlı çalışmanın, tezin proje safhasından sonuçlanmasına kadarki bütün süreçlerde bilimsel ahlak ve geleneklere aykırı düşecek bir yardıma başvurulmaksızın yazıldığını ve yararlandığım eserlerin Bibliyografya’da gösterilenlerden oluştuğunu, bunlara atıf yapılarak yararlanılmış olduğunu belirtir ve onurumla beyan ederim. (10/02/2020)

Onur YILMAZ

Canım Annem, Babam, Kardeşim ve Sevgili Eşim Asel'e

ÖNSÖZ

Görme engelli bireyler, yaşamları boyunca birçok problemle karşılaşmaktadırlar. Yetersiz eğitim, yetersiz görme engelli olanakları gibi durumlar görme engellilerin yaşantılarını olumsuz yönde etkilemektedir. Günümüz teknolojisi ile birlikte, görme engellilerin gündelik yaşantısını kolaylaştırabilecek, herhangi bir yakınına ihtiyaç duymadan yaşantısına devam edebilecek birçok çalışma yapılmaktadır. Bu çalışmada, akıllı telefon kamerası kullanılarak, kameraya gösterilen ilacı algılayıp, görme engelli bireye sesli olarak ilacın tanıtımı amaçlanmıştır.

Tez konusunun belirlenmesinde, baştan sona kadar büyük bir sabırla bana gösterdikleri destek ve yardımlarından dolayı saygıdeğer hocam ve tez danışmanım olan Dr. Öğr. Üyesi Ahmet GÜRHANLI'ya teşekkür ederim. Sayın Gürhanlı'nın bilgi, tecrübe ve yönlendirmeleri çalışmanın yürütülmesinde son derece faydalı olmuştur. Bu zorlu çalışma döneminde uygun çalışma ortamını sağlayan ve desteğini esirgemeyen değerli aileme de teşekkürlerimi sunarım.

Şubat 2020 **Onur YILMAZ**

İÇİNDEKİLER

Sayfa

ÖNSÖZ ..... **ix**
İÇİNDEKİLER ..... **xi**
KISALTMALAR LİSTESİ ..... **xiii**
ÇİZELGE LİSTESİ ..... **xv**
ŞEKİL LİSTESİ ..... **xvii**
ÖZET ..... **xix**
ABSTRACT ..... **xxi**
1. GİRİŞ…. **1**
1.1 Tezin Amacı 2
1.2 Literatür Araştırması 2
2. KAYNAK ARAŞTIRMASI **3**
2.1 Görüntü İşleme 3
2.2 Yapay Sinir Ağları 4
2.2.1 İleri Beslemeli Yapay Sinir Ağları 5
2.2.2 Geri Beslemeli Yapay Sinir Ağları 5
2.3 Aktivasyon Fonksiyonu 6
2.3.1 Sigmoid Fonksiyonu 6
2.3.2 ReLU Fonksiyonu 6
2.3.3 Leaky ReLU Fonksiyonu 7
2.4 Optimizasyon Algoritmaları 8
2.4.1 Adagrad Optimizasyon Algoritması 8
2.4.2 Rmsprop Optimizasyon Algoritması 9
2.4.3 Adam Optimizasyon Algoritması 9
2.4.4 Nadam Optimizasyon Algoritması 9
2.5 Sınıflandırma 10
2.5.1 Karar Ağacı 11
2.5.2 Destek Vektör Makineleri 11
2.5.3 K-En Yakın Komşu Algoritması 13
2.5.4 Tekrarlayan Sinir Ağ (RNN) 14
2.5.5 Konvolüsyonel Sinir Ağ (CNN) 15
2.5.5.1 Girdi Katmanı 16
2.5.5.2 Konvolüsyon Katmanı 16
2.5.5.3 Havuzlama Katmanı 16
2.5.5.4 Tam Bağlantılı Katman 17
2.5.6 Görüntü Sınıflandırma Algoritmalarının Performansı 18
2.6 iOS Yapısı ve Katmanları 20
2.6.1 Cocoa Touch Katmanı 20
2.6.2 Medya Katmanı 22
2.6.3 Core Service Katmanı 23
2.6.4 Core OS Katmanı 24
2.7 Swift Programlama Dili 25

2.8 Xcode Yazılım Geliştirme Ortamı 26
3. MATERYAL VE YÖNTEM 29
3.1 Veri Seti Kararı 29
3.2 Veri Setinin Oluşturulması 30
3.3 Optimizasyon Algoritmalarının Performansı 31
3.4 Veri Setinin Eğitilmesi 31
3.5 Uygulamanın Geliştirilmesi 34
4. DEĞERLENDİRME VE ÖNERİLER 35
4.1 Değerlendirme 35
4.2 Öneriler 37
5. SONUÇ… 39
KAYNAKLAR 41
EKLER 45
ÖZGEÇMİŞ 57

KISALTMALAR LİSTESİ

AAC : Advanced Audio Coding
AÇSHB : Aile, Çalışma ve Sosyal Hizmetler Bakanlığı
Adagrad : Adaptive Gradient Algorithm
Adam : Adaptive Moment Estimation
ALAC : Apple Lossless Audio Codec
ANN : Artificial Neural Networks
API : Application Programming Interface
CNN : Convolutional Neural Networks
CPU : Central Processing Unit
FTP : File Transfer Protocol
GIF : Graphics Interchange Format
IP : Internet Protocol
JPEG : Joint Photographic Experts Group
K-NN : K-En Yakın Komşu Algoritması
MIDI : Musical Instrument Digital Interface
MLP : Multi-Layer Perceptron
NAG : Nesterov Accelerated Gradient
PDF : Portable Document Format
PNG : Portable Network Graphics
RAM : Random Access Memory
ReLU : Rectified Linear Unit
Rmsprop : Root Mean Square Propagation
RNN : Recursive Neural Network
SDK : Software Development Kit
SGD : Stochastic Gradient Descent
SSL : Secure Sockets Layer
SVM : Support Vector Machine
TCP : Transmission Control Protocol
TIFF : Tagged Image File Format
TSL : Transport Layer Security
WWDC : Worldwide Developers Conference

ÇİZELGE LİSTESİ

Sayfa

Çizelge 2.1 : Yaprak Veri Seti 18
Çizelge 2.2 : SVM Algoritması Performansı 18
Çizelge 2.3 : ANN Algoritması Performansı 19
Çizelge 2.4 : CNN Algoritması Performansı 19
Çizelge 3.1 : Değerlendirmeye Alınan Algoritmalar 31
Çizelge 3.2 : Create ML İçerisinde Bulunan Sınıflandırmalar 32
Çizelge 4.1 : Optimizasyon Algoritmaları Performans Karşılaştırması 35
Çizelge 4.2 : Uygulama Testi 37

ŞEKİL LİSTESİ

Sayfa

Şekil 2.1 : Dijital Görüntü İşlemenin Temel Basamakları 3
Şekil 2.2 : Derin Sinir Ağ Mimarisi (Waheed, Behery, & Elshewey, 2016)..... 4
Şekil 2.3 : İleri Beslemeli Nöron Ağ Yapısı (Yavuz & Deveci, 2012) 5
Şekil 2.4 : Geri Beslemeli Nöron Ağ Yapısı (Yavuz & Deveci, 2012)..... 5
Şekil 2.5 : Sigmoid Fonksiyonu Şekilsel Gösterimi..... 6
Şekil 2.6 : ReLU Fonksiyonu Şekilsel Gösterimi..... 7
Şekil 2.7 : Leaky ReLU Fonksiyonu Şekilsel Gösterimi..... 7
Şekil 2.8 : Sınıflandırma Örneği..... 10
Şekil 2.9 : Karar Ağacı Örneği 11
Şekil 2.10 : Destek Vektörlerinin Hiperdüzlem ile Ayrımı..... 12
Şekil 2.11 : (8,4) Noktasına Yakın Komşular 13
Şekil 2.12 : Tekrarlayan Sinir Ağ (Güneş, 2018)..... 14
Şekil 2.13 : RNN'lerin İleri Beslemeli Ağlardan Farkı 14
Şekil 2.14 : Konvolüsyonel Sinir Ağ Yapısı (Rahmon, 2018) 15
Şekil 2.15 : Konvolüsyon İşlemi (Url-1)..... 16
Şekil 2.16 : Havuzlama İşlemi (Url-2) 17
Şekil 2.17 : Tam Bağlantılı Katman (Xu, Chen, & Li, 2015)..... 17
Şekil 2.18 : iOS Katmanları (Smyth, 2012)..... 20
Şekil 2.19 : Rakam Sıralama Performans Değerlendirmesi 26
Şekil 2.20 : Storyboard Genel Görünümü 27
Şekil 3.1 : En Çok Satılan İlaç Listesi..... 29
Şekil 3.2 : Veri Seti Örneği 30
Şekil 3.3 : Klasör Yapısı..... 32
Şekil 3.4 : İlaç Modeli Klasör Yapısı 33
Şekil 3.5 : Create ML ile Eğitim İşlemi..... 33
Şekil 3.6 : Create ML ile Eğitim Sonrası Model Performansı..... 34
Şekil 3.7 : Uygulamanın Çalışma Şekli..... 34
Şekil 4.1 : Adam - Rmsprop Karşılaştırması – 1 36
Şekil 4.2 : Adam - Rmsprop Karşılaştırması - 2..... 36
Şekil A.1 : Tez Çalışması Akış Diyagramı..... 47
Şekil A.2 : Xcode Main.storyboard Görüntüsü 48

YAPAY SİNİR AĞLARI İLE GÖRME ENGELLİLER İÇİN İLAÇ TANIMA VE SESLİ BİLGİ VERME UYGULAMASI

ÖZET

Görme engelli bireyler, yaşamları boyunca birçok problemle karşılaşmaktadırlar. Yetersiz eğitim, yetersiz görme engelli olanakları gibi durumlar görme engellilerin yaşantılarını olumsuz yönde etkilemektedir. Aile, Çalışma ve Sosyal Hizmetler Bakanlığı'nın 2011 yılında yapmış olduğu çalışmaya göre, Türkiye nüfusunun %1,4'ü yani 1.039.000 kişi engel olarak görmede zorluk yaşayanlar grubuna girmektedir. Günümüz teknolojisi ile birlikte, görme engellilerin gündelik yaşantısını kolaylaştırabilecek, herhangi bir yakınına ihtiyaç duymadan yaşantısına devam edebilecek birçok çalışma yapılmaktadır. 2011 yılında Apple tarafından duyurulan Siri'den, Siirt Zübeyde Hanım Mesleki ve Teknik Anadolu Lisesi öğrencilerinin "TÜBİTAK 4006 Bilim Fuarı Sergisi" kapsamında tasarladığı sensörlü bastona kadar yapay zeka yardımı ile görme engellilere yardımcı olabilecek birçok ürün ve yazılım tasarlanmaktadır. Bu çalışmada, görme engellilerin kendi başına ilaç kullanabilmesi için bir uygulama geliştirilmiştir. Uygulamanın geliştirilebilmesi için her bir ilaç için 100 fotoğraf kullanılmıştır ve toplamda 10 sınıflı bir veri kümesi oluşturulmuştur. Bu verilere CNN, Adam, Rmsprop, Adagrad, Nadam algoritmaları uygulanmıştır. Optimizasyon algoritmalarından en iyi sonucu veren ile model geliştirilmiş ve iOS tabanlı uygulamaya eklenmiştir.

Anahtar Kelimeler : *Makine Öğrenmesi, Resim Sınıflandırma, Yapay Sinir Ağları, Optimizasyon Algoritmaları, İlaç Tanıma*

DRUG RECOGNITION AND VOICE INFORMATION APPLICATION FOR VISUAL IMPAIRED WITH ARTIFICIAL NEURAL NETWORKS

ABSTRACT

Visually impaired individuals face many problems throughout their lives. Conditions such as inadequate education and inadequate visual impairments affect the lives of the visually impaired. According to a study made in 2011 Ministry of Family, Labor and Social Services, 1.4% of Turkey's population is 1.039 million people fall into groups that have difficulty in seeing the obstacle. With today's technology, there are many studies that can facilitate the daily life of visually impaired people and continue their lives without the need of any relatives. Many products and software that can help visually impaired people are being designed with the help of artificial intelligence from Siri, which is announced by Apple in 2011, to the walking stick with sensor that Siirt Zübeyde Hanım Vocational and Technical Anatolian High School students have designed within the scope of "TÜBİTAK 4006 Science Fair Exhibition". In this thesis, an application has been developed for visually impaired individuals to use drugs on their own. 100 photographs were used for each drug to create the application and a total of 10 class datasets were created. CNN, Adam, Rmsprop, Adagrad, Nadam algorithms were applied to this data. With the best results from optimization algorithms, the model was developed and added to the iOS based application.

Keywords: *Machine Learning, Image Classification, Multiple Linear Regression, Optimization Algorithm, Medicine Detection*

1. GİRİŞ

Işığı algılayamayan veya iki gözü ile birlikte 3 metre uzaktan bir elin parmaklarını sayabilecek kadar ışığı algılayamayan insanlar görme engelli olarak kabul edilir (Yücel & Acatürk, 2006). Dünya Sağlık Örgütü'nün 2010 yılında yapmış olduğu çalışmaya göre 285 milyon görme engelli yaşamaktadır. Tamamen görme yetisini kaybetmiş bireyler ise 39 milyonu bulmaktadır (WHO, 2013). Görme engelli bireylerin, günlük yaşamlarında rahatça yürümeleri, engelleri önceden tespit edebilmeleri, bilmedikleri bir alanda önlerine çıkan nesnelerin ne olduğunu anlamaları oldukça zordur. Bu zorlukların üstesinden gelebilmek için çeşitli araç ve gereçler kullanırlar. En yaygın araçlardan birisi olan baston ile sadece yakınında bir cisimin olup olmadığını, dış mekanda bir engelle karşılaşıp karşılaşamadığı anlaşılabilmektedir. Görme engellilerin günlük yaşamlarında ihtiyaç duyduğu adres sorma, nesne bulma ve yön bulma gibi benzeri durumlar için başka bir insanın desteğine ihtiyaç duyabilirler. Günümüz teknolojisi ile akıllı telefonlar belli bir seviyeye kadar destek olabilmektedirler. Apple'ın geliştirmiş olduğu Siri ve Google'ın geliştirmiş olduğu Google Asistan günümüzün en yardımcı geliştirmeleridir. Uygulama geliştiricileri de, nesne tanıma, yazı okuma ve yön tarif etme gibi birçok alanda yardımcı uygulamalar geliştirmektedir.

Beynimize ulaştırılan bilgilerin çoğu göz tarafından iletilmektedir (Erdil & Elbaş, 2012). Göz, cisimlerden yansıyan fotonları beynin algılayabileceği forma dönüştürür. Görüntü işlemenin de temeli göz ve beyin arasındaki iletişim yapısının görüntü ve bilgisayar üzerine kurulmasıdır. Bir çok alanda karşımıza çıkan görüntü işleme; sağlık, savunma, biyoloji, astronomi ve bunun gibi bir çok benzer alanda yeni çözümler sunmaktadır (Acharya & Ray, 2005). Görüntünün işlenebilmesi için görüntünün bilgisayar ortamına aktarılarak sayısal hale getirilmesi gerekmektedir. Sayısal hale getirilen görüntü yapay sinir ağları (YSA) yardımıyla anlamlı hale getirilmektedir. Yapay sinir ağları, insan beyninden esinlenerek ortaya atılmıştır ve sinir hücrelerinin çalışma şekli simüle edilerek katmanlı bir ağ yapısı oluşturulmuştur (Doğan G. , 2010).

YSA'nın işleyişi, yeni bir bilgiyi kaydetme ve eski bilgiler arasında ilişkiler kurma temeline dayanır. Bu sistem almış olduğu her yeni bilgi ile eğitilmektedir. Eğitilen model ile birlikte günümüz teknolojilerinde kullanılan sıkça karşımıza çıkmaktadır.

Bu çalışmada, görme engelliler için ilaç tanıma yapabilen ve bulduğu sonucu sesli olarak kullanıcıya aktaran sistem geliştirilmiştir. Literatürdeki görüntü sınıflandırma problemleri hazır modeller kullanımıyla çözülmeye çalışılmıştır. Bu çalışmada, oluşturulan veri seti ile birlikte en yüksek performans değerine sahip optimizasyon algoritması kullanılmıştır. Bu kararın ortaya çıkmasına neden olan araştırmalar paylaşılmıştır.

1.1 Tezin Amacı

Günümüzde görme engellilere sağlanan faydaların yeterli olmadığı görülmektedir. Her geçen gün yeni icatlar ve faydalar artarak devam etse de, yeteri kadar destek görmemektedir. Aile, Çalışma ve Sosyal Hizmetler Bakanlığı'nın (AÇSHB) yayınlamış olduğu Engelli ve Yaşlı İstatistik Bülteni'nde, Türkiye'deki görme engelli oranını toplam nüfusa oranla %1.4 olarak belirtmiştir. Beslenme şekilleri, kötü yaşam koşulları, güvensiz çalışma alanları gibi etkenlerle bu oranın yükselmesi beklenmektedir. Yazılan bitirme tezinde, görme engelli bireylere kimseye ihtiyaç duymadan ilaç kullanımlarını kolaylaştırmaktır. Tez çalışmasına ait akış şeması Şekil A.1'de gösterilmiştir.

1.2 Literatür Araştırması

Tezde aşamalı olarak görüntü işleme ve yapay sinir ağları alanında yapılan çalışmalar incelenmiş ve bu tezde birleştirilmiştir. Bu alanda ülkemizde ve yabancı ülkelerde uygulanan örnekler incelenmiş, çıkarılan bilgiler bu tezde verilmiştir.

2. KAYNAK ARAŞTIRMASI

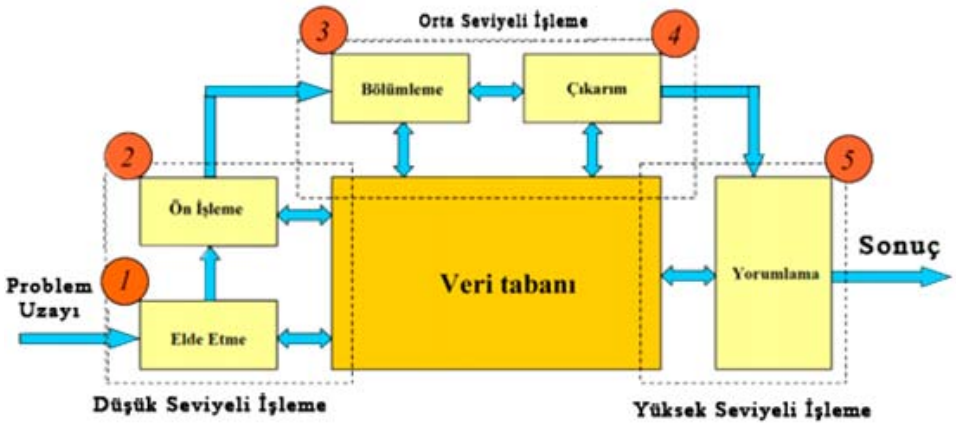
2.1 Görüntü İşleme

Görüntü, depolanan ve görüntülenen iki boyutlu görsel bilgi olarak tanımlanabilir (Sinecen, 2016). Görüntü, 3 boyutlu nesnenin iki boyutlu yüzeyinden yansıma ışını yakalayan ışığa duyarlı cihazlar sayesinde oluşur. Görüntülerden anlamlı anlamlı bilgiler çıkarmak için yapılan işlemlere de görüntü işleme denilmektedir. Görüntü işleme üç başıkta incelenebilir (Aytan, Öztürk, & Örgev, 1993).

Optik görüntü işleme, optiklerin dizilimi ile yapılmaktadır. Gözlükler ve fotoğrafçıların kullandığı karanlık oda optik görüntü işleme için örnek olarak verilebilir. Karanlık odanın öncüleri, görüntü işleme tekniklerinin ilk kullananları olarak kabul edilebilir.

Analog görüntü işleme, görüntülerin elektriksel olarak değişimine bağlıdır. Elektriksel formda görüntü işleyen teknik için televizyon örneği verilebilir.

Dijital (Sayısal) görüntü işleme, analog bir görüntünün bilgisayarlarla işlenmesi ile birlikte sayılsal forma dönüştürülmesine denir. Temel basamakları Şekil 2.1'de gösterilmiştir.



Şekil 2.1 : Dijital Görüntü İşlemenin Temel Basamakları

Kaynak : (Çayıroğlu, 2018)

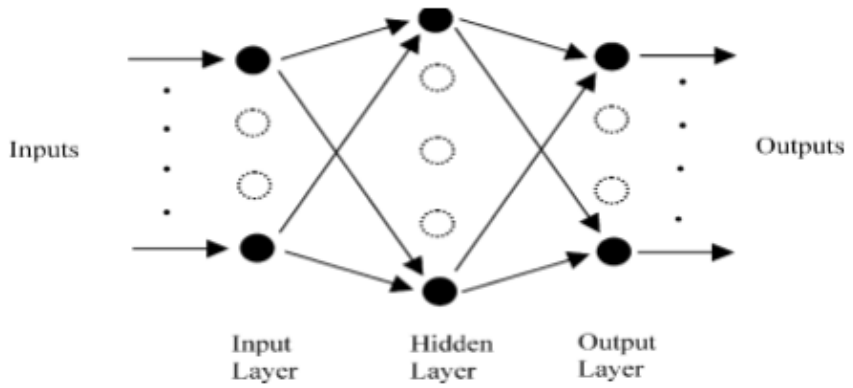
2.2 Yapay Sinir Ağları

Tarihte ilk yapay sinir ağ modeli 1940'ların başında geliştirilmiştir. Amerikalı Warren McCulloch ve öğrencisi Walter Pitts ile birlikte bugün birçok ağ için ana fikir olduğu basit bir sinir hücresi modeli tasarladı. 1957'de Frank Rosenblatt harf okuyabilen bir YSA geliştirdi.

YSA, insan beyninden esinlenerek geliştirilen yapay sinir ağlarının çok katmanlı halidir. İlk geliştirilen ve en ilkel yapay sinir ağı olan tek katmanlı algılayıcı (single layer perceptron), doğrusal olmayan problem çözümünde yeterli olmadığı için çok katmanlı algılayıcı (MLP) geliştirilmiştir (Fernandez, E., Martin, & A.J., 2006). Çok katmanlı ağlar, XOR problemine çözüm arayışı ile ortaya çıkmıştır. Çok katmanlı ağlar 3 katmandan oluşmaktadır.

- Giriş Katmanı: İşlenecek verinin ağa dahil edildi katmandır.
- Ara Katmanlar: Giriş katmanından gelen bilgilerin işlendiği katmandır. Ara katmanlar gizli katman (Hidden Layer) olarak da adlandırılır. Ağ içerisinde birden fazla olarak bulunabilirler. Ağın türüne göre katman içerisindeki nöron sayıları da değişiklik gösterebilir.
- Çıkış Katmanı: Ara katmanlarda üretilen bilginin sonucunu veren katmandır. İhtiyaca göre olan çıktıyı üretir.

Şekil 2.2'de iki gizli katmandan oluşan derin sinir ağ mimarisi verilmiştir.



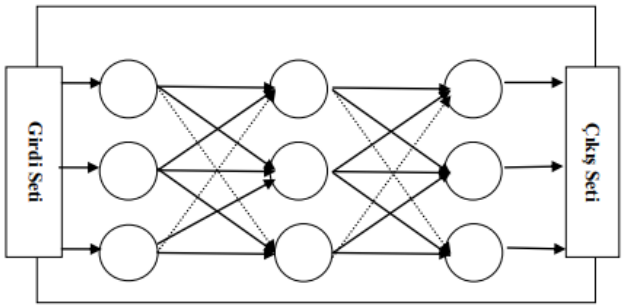
Şekil 2.2 : Derin Sinir Ağ Mimarisi

Kaynak : (Waheed, Behery, & Elshewey, 2016)

YSA'lar nöronların birbirine bağlanış şekline göre ileri beslemeli ve geri beslemeli olarak ikiye ayrılır.

2.2.1 İleri Beslemeli Yapay Sinir Ağları

İleri beslemeli sinir ağ yapılarında ters yöne doğru yönelim yoktur. Verilerin ilk önce giriş katmanına sonra ara katmana ve son olarak çıktı katmanına doğru ilerlediği bir ağ yapısıdır. Herhangi bir katmandaki yapay sinir hücreleri kendinden bir önceki hücrelerden beslenir (Özveren, 2006). Şekil 2.3'de ileri beslemeli nöron ağlarının yapısı gösterilmiştir.

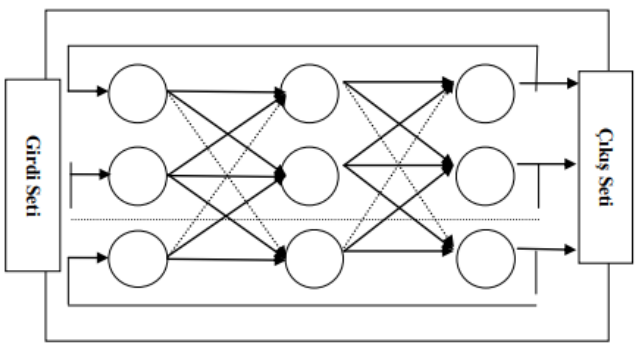


Şekil 2.3 : İleri Beslemeli Nöron Ağ Yapısı

Kaynak : (Yavuz & Deveci, 2012)

2.2.2 Geri Beslemeli Yapay Sinir Ağları

Geri beslemeli sinir ağ yapılarında ise veri kendinden önceki katmanlara da iletilebilir. Çıktıdaki bir veri kendinden önceki katmana ya da kendi katmanındaki bir nörona girdi olarak verilebilir. Geri beslemeli sinir ağın yapısı Şekil 2.4'de gösterilmiştir.



Şekil 2.4 : Geri Beslemeli Nöron Ağ Yapısı

Kaynak : (Yavuz & Deveci, 2012)

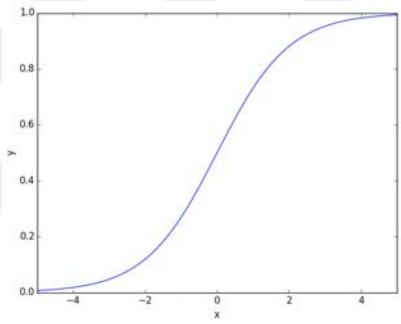
2.3 Aktivasyon Fonksiyonu

2.3.1 Sigmoid Fonksiyonu

Hücreye gelen net girdiyi işleyerek hücrenin bu girdiye karşılık üreteceği çıktıyı belirlemek için kullanılır Günümüzde MLP modellerinde en yaygın kullanılan aktivasyon fonksiyonu sigmoid fonksiyonudur (Öztemel, 2006). Sigmoid fonksiyonu Denklem (2.1)'de gösterilmektedir.

$$F(NET) = \frac{1}{1 + e^{-NET}} \tag{2.1}$$

Denklemde görülen NET girdi değerini göstermektedir. Bu değer toplama fonksiyonu kullanılarak belirlenmektedir (Öztemel, 2006). Sigmoid fonksiyonu şekilsel olarak Şekil 2.5'de gösterilmiştir.



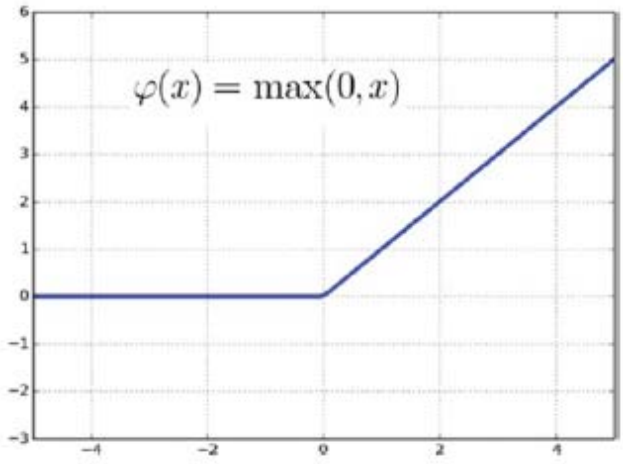
Şekil 2.5 : Sigmoid Fonksiyonu Şekilsel Gösterimi

Kaynak : (Öztemel, 2006)

2.3.2 ReLU Fonksiyonu

ReLU, günümüzün en popüler bir diğer aktivasyon fonksiyonlarından birisidir. Gelen girdileri sıfır ile sonsuz arasında çıktıya dönüştürür. Gelen değer negatif ise işlem sonucu olarak sıfır verir. Eğer sıfırdan büyük herhangi bir değiştirme işlemi uygulamaz, fonksiyondan olduğu gibi geçer. ReLU fonksiyonu denklem olarak Denklem (2.2)'de, şekilsel olarak da Şekil 2.6'da gösterilmiştir (Kim, 2017).

$$\varphi'(x) = \begin{cases} 1, x > 0 \\ 0, x \leq 0 \end{cases} \tag{2.2}$$

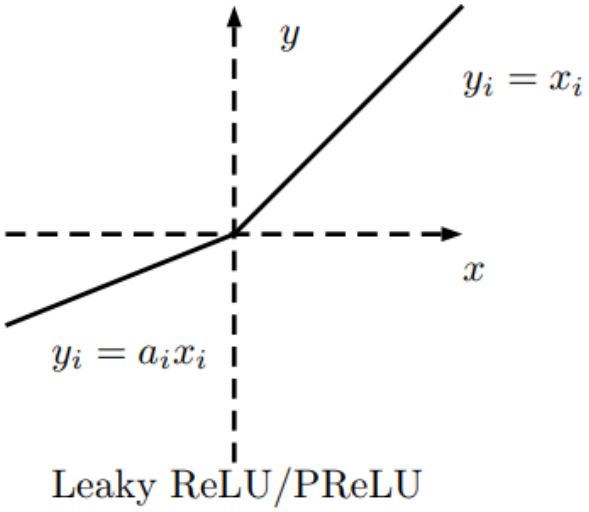


Şekil 2.6 : ReLU Fonksiyonu Şekilsel Gösterimi

2.3.3 Leaky ReLU Fonksiyonu

ReLu aktivayon fonksiyonunun negatif gördüğü değeri sıfıra eşitlediği için ölü Relu olarak da adlandırılır. Bu durum negatif değerlerin fazla olduğu ağlarda katmanların aktif hale gelmemesine ve sağlıklı bir eğitim gerçekleşmemesine neden olmaktadır. Bu durumun önüne geçebilmek için negatif değerlerde fonksiyona eğim eklemek amacıyla sabit bir parametre seçilir. Eklenen sabit parametre (a_i) 0.01 gibi küçük ve sabit bir değer alırsa Leaky ReLU oluşturulur. Leaky ReLU denklem olarak Denklem (2.3)'te, şekilsel olarak da Şekil 2.7'de gösterilmiştir.

$$y_i = \begin{cases} x_i, IF\ x_i > 0 \\ a_i x_i, IF\ x_i \leq 0 \end{cases} \quad \textbf{(2.3)}$$



Şekil 2.7 : Leaky ReLU Fonksiyonu Şekilsel Gösterimi

Kaynak : (Kuş, 2019)

2.4 Optimizasyon Algoritmaları

Derin öğrenme uygulamalarında optimizasyon, uygulamanın sağlıklı bir şekilde sonuç verebilmesi için hata fonksiyonunun mutlak minimum değerini bulan yöntemdir. Amacı, eğitim sürecinde ağın ağırlığını (w) ve bias (b) değerlerini güncelleyerek optimum çözümü bulma mantığında çalışır. YSA optimizasyonunda kullanılan yöntemlerden birisi Gradient Descent'tir. Gradient Descent yöntemi, YSA'da θ parametlelerine bağlı maliyet fonksiyonunu ($J(\theta)$) minimize etmek için kullanılır. Öğrenme işlemi, başlangıçta rastgele belirlenen θ parametresinin her adımda gradientin tersi yönünde güncellenerek eğitilme işleminin tekrar edilmesi ile gerçekleştirilir (Yazan & Talu, 2017). Gradient Descent'in, toplu eğim azaltma "Batch Gradient Descent", parçalı eğim azaltma "Mini-Batch Gradient Descent" ve rastgele eğim azaltma "Stochastic Gradient Descent" olmak üzere 3 farklı türü bulunur (Bengio, Boulanger-Lewandowski, & Pascanu, 2012).

Gradient Descent yöntemini esas alan çeşitli algoritmalar vardır. Bunlardan en popüler olanları; Root Mean Square Propagation (Rmsprop), Adaptive Gradient Algorithm (Adagrad), Adaptive Moment Estimation (Adam), Stochastic Gradient Descent (SGD)'tir (Kurt, 2018).

2.4.1 Adagrad Optimizasyon Algoritması

Adagrad, gradient descent yöntemindeki sabit öğrenme katsayısı problemini, her adımda öğrenme katsayısını güncelleyerek çözen bir yöntemdir. Adagrad, her bir parametre θ_i için her bir t işleminde farklı katsayı kullanır. Adagrad algoritmasının katsayı güncelleme yöntemi Denklem (2.4)'te gösterilmiştir (Ruder, 2016).

$$g_{t,i} = \nabla_{\theta_t} J(\theta_{t,i})$$
$$\theta_{t+1,i} = \theta_{t,i} - \frac{n}{\sqrt{G_{t,ii} + \epsilon}}\, g_{t,i} \qquad \textbf{(2.4)}$$

2.4.2 Rmsprop Optimizasyon Algoritması

Rmsprop algoritması, Adagrad algoritmasının radikal olarak küçülülen öğrenme katsayısı sorununa çözüm olarak geliştirilmiştir. Öğrenme oranını üstel olarak azalan eğimin karesinin ortalamasına böler. Böylelikle daha etkin bir öğrenme elde edilmiş olur. Rmsprop algoritmasının güncelleme yöntemi Denklem (2.5)'te gösterilimiştir (Ruder, 2016).

$$E[g^2]_t = 0.9E[g^2]_{t-1} + 0.1g_t^2$$
$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_{t+\epsilon}}} g_t \quad \textbf{(2.5)}$$

2.4.3 Adam Optimizasyon Algoritması

Rmsprop'ta yapıldı gibi geçmiş eğimlerin karelerinin üssel olarak ağırlıklandırılmış ortalamalarının (v_t) bulunmasının yanında, momentumdaki geçmiş değişikliklerini de (m_t) önbellekte tutar. Adam algoritmasının güncelleme yöntemi Denklem (2.6)'da gösterilmiştir.

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_{t^2}$$
$$m'_t = \frac{m_t}{1-\beta_1^t}, v'_t = \frac{v_t}{1-\beta_2^t} \quad \textbf{(2.6)}$$
$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v'_t + \epsilon}} m'_t$$

Bu eşitliği varsayılan değerleri β_1 için 0.9, β_2 için 0.999, ϵ için 10^{-8} olarak belirtilmiştir (Kingma & Ba, 2017).

2.4.4 Nadam Optimizasyon Algoritması

Nadam, Nesterov accelerated gradient (NAG) ve Adam kombinasyonundan oluşmuştur. Adam algoritmasını ile NAG algoritması kombine edebilmek için momentum ifadesi üzerinde değişiklik yapılması ile birlikte Nadam algoritması

oluşturulur. NAG ile birlikte eğim değeri hesaplamadan önce parametreler momentum ifadesiyle güncellenerek eğim yönünde daha tutarlı bir sonuç alınır. Nadam algoritmasının yöntemi Denklem (2.7)'de gösterilmiştir (Ruder, 2016).

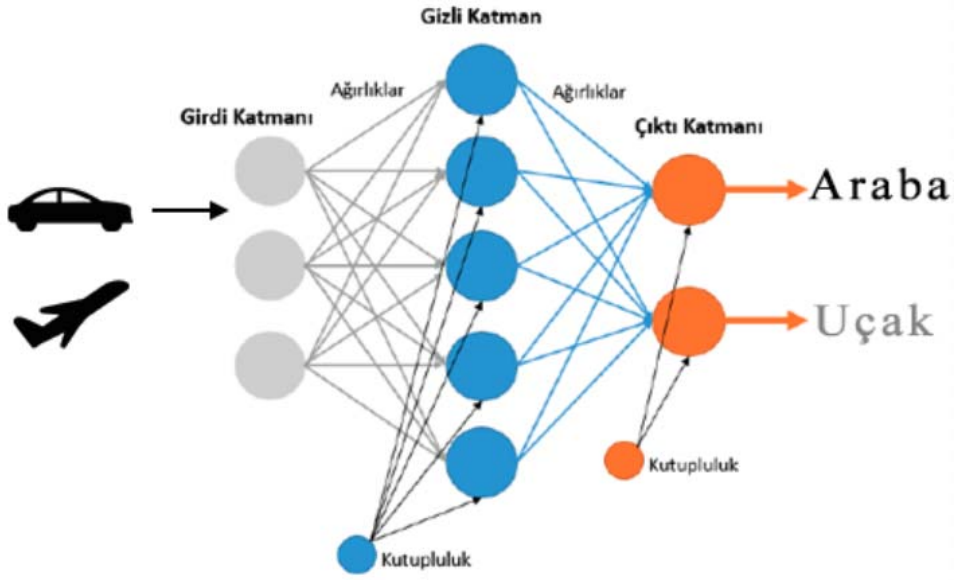
$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v'_t + \epsilon}}(\beta_1 m'_t + \frac{(1-\beta_1)g_t}{1-\beta_1^t}) \quad \textbf{(2.7)}$$

2.5 Sınıflandırma

Karşımıza ilk defa çıkan bir gözlemin özelliklerini, önceki sınıfını bildiğimiz gözlemlerle kıyaslayarak bir sınıf belirleme işlemine sınıflandırma işlemi denir. Matematiksel ifadesi Denklem (2.8)'deki gibi gösterilmiştir (Yılmaz, 2013).

$$(x_1, y_1), (x_2, y_2), \ldots\ldots, (x_n, y_n) \in X \times \{-1, +1\} \quad \textbf{(2.8)}$$

Denklem (2.8)'te (+1) ve (-1) sınıfları, $y_i \in$ {-1, +1} sınıf etiketi ve $x_i \in X$ kümesine ait bir denklem verilmiştir. Her bir x_i gözleminin y_i sınıfına ait olduğu bu türe ikili sınıflandırma denilmektedir (Yılmaz, 2013). Sınıflandırmaya ait bir görüntü Şekil 2.8'de gösterilmiştir.

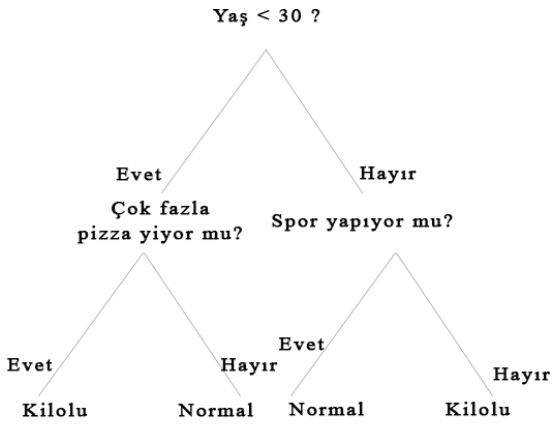


Şekil 2.8 : Sınıflandırma Örneği

Kaynak : (Yılmaz, 2013)

2.5.1 Karar Ağacı

Karar ağacı, bir hedef değişkene ihtiyaç duyan öğrenme modelidir. Günümüzde sınıflandırma ve tahmin için oldukça sık kullanılan bir yöntemdir. Karar ağacı kullanılarak verinin sınıflandırılması öğrenme ve sınıflama şeklinde iki basamaklı bir işlemden oluşur (Alemdar, 2019). Sınıflandırma işlemi, belirli özellik değerlerine sahip veriler ile gerçekleştirilir. Bu işlem ile birlikte algoritmaya sunulan verilerin bir kısmı girdi, diğer kısmı çıktı olarak verilir. Karar ağaçları Ağaç Oluşturulması ve Ağaç Budama olmak üzere iki aşamadan oluşur (Köklü, 2014). Karar ağacına bir örnek Şekil 2.9'da gösterilmiştir.



Şekil 2.9 : Karar Ağacı Örneği

Şekil 2.9'da gösterilen karar ağacı örneğinin kuralları şu şekildedir:

Eğer yaş ≤30 = "Evet" ve Çok fazla pizza yiyor mu? = "Evet" Sonuç: Kilolu

Eğer yaş ≤30 = "Evet" ve Çok fazla pizza yiyor mu? = "Hayır" Sonuç: Normal

Eğer yaş ≤30 = "Hayır" ve Spor yapıyor mu? = "Evet" Sonuç: Normal

Eğer yaş ≤30 = "Hayır" ve Spor yapıyor mu? = "Hayır" Sonuç: Kilolu

2.5.2 Destek Vektör Makineleri

Destek vektör makineleri (SVM), hem sınıflandırma hem de regresyon için kullanılabilen bir makine öğrenmesi yöntemidir. Eğitim verisi üzerinde işlem eğitim yapılarak yeni girdiler üzerinde tahmin yapılır (Kavuncu, 2018).

SVM, 1960'ların sonlarında Vladimir Vapnik tarafından bulunan bir yöntemdir. El yazısını tanıma, ses tanıma ve veri analizi gibi sınıflandırmalarda kullanışmıştır.

Doğrusal ve doğrusal olmayan iki sınıflı veri kümelerinin sınıflandırılmasında sınıf üyelerini üye olmayanlardan ayıracak şekilde sınıflandırma yapar (Kadiroğlu, 2019).

SVM dizayn adımları aşağıdaki şekilde verilebilir (Yıldırım, 2006);

1.Adım: Sınıflandırma probleminde karar fonksiyonunun biçimini belirleyen kernek fonksiyonunun belirlenmesi

2.Adım: Seçilen kernel fonksiyonu için parametresinin belirlenmesi

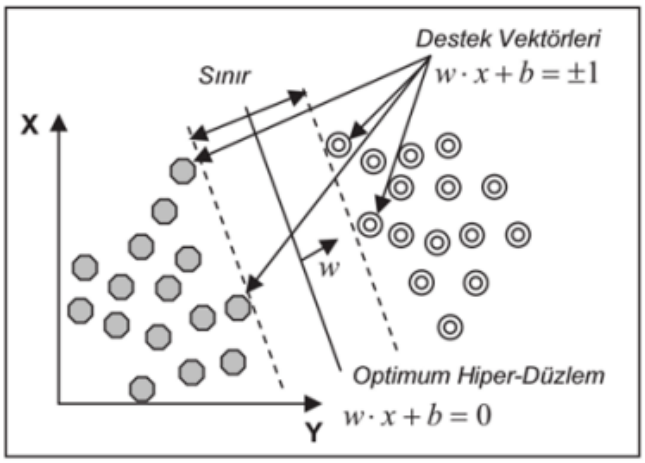
3.Adım: C yaptırım faktörünün belirlenmesi

4.Adım: İkinci derecenden problem çözümü

SVM'lerde amaç yeni verilerin tanıtılmasında hatayı en aza indirecek sınır çizgisinin çizilmesidir. Sınır çizgisi Denklem (2.9)'da gösterilen eğim formülü ile oluşturulur.

$$f(x) = w.x + b \tag{2.9}$$

Formülde gösterilen w ağırlık vektörü, x girdi verilerini, b hiperdüzleme dik olan vektörün eğim değeri ve f(x) veri noktalarının sınıflarını göstermektedir. İki kategorili sınıflandırma probleminden oluşturulmuş hiperdüzlemin gösterimi Şekil 2.10'da gösterilmiştir.



Şekil 2.10 : Destek Vektörlerinin Hiperdüzlem ile Ayrımı

Kaynak : (Kavuncu, 2018)

2.5.3 K-En Yakın Komşu Algoritması

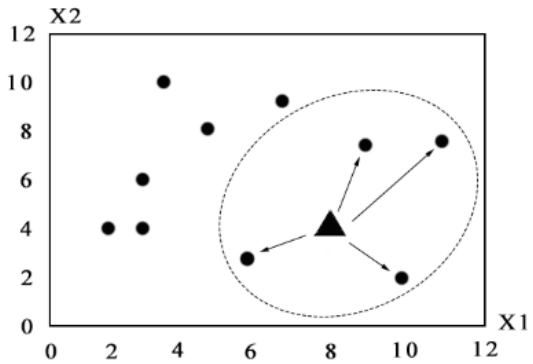
K-En yakın komşu algoritması, veri sınıflandırma ve regresyon için en çok kullanılan algoritmalardan birisidir (Barbe, Tewfik, & Khodursky, 2007). Sınıflandırılma yapılırken veri setindeki noktaların diğer noktalar ile mesafe hesaplaması yapılır. K komşu sayısını ifade etmektedir. Gözlem değerlerinden oluşan bir küme için aşağıdaki adımlardan oluşur:

- K parametresi belirlenir. K verilen bir noktaya en yakın komşuların sayısıdır.
- Algoritma, verilen her noktaya en yakın komşuları belirleyeceği için, bahsedilen nokta ile diğer noktalar arasındaki uzaklıklar tek tek hesaplanır.
- Hesaplanan uzaklıklara göre satırlar sıralanır. Bunlardan en küçük olanı k tanesi seçilir.
- Seçilen satırların kategorileri belirlenir ve en çok tekrarlanan değer seçilir.
- Seçilen kategori, tahmin edilmesi beklenen gözlem değerinin kategorisi olarak belirlenir (Özkan, 2008)

K=3 için yeni bir sınıf belirlenildiği düşünülürse, eski sınıflandırılmış elemanlardan en yakın 3 tanesi alınır. Alınan elemanlar hangi sınıfa dahilse, yeni gelen eleman da o sınıfa dahil edilir. Uzaklıkların hesaplanması yönteminde Öklid uzaklık formülü kullanılır. Formül Denklem (2.10)'da gösterilmiştir (Tekeli & Aşlıyan, 2016).

$$d(x,y) = \sqrt{\sum_{i=1}^{p}(x_i - y_i)^2} \qquad \textbf{(2.10)}$$

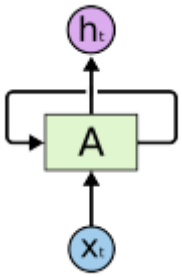
K=4 verildiği (8,4) noktasına en yakın 4 komşunu Şekil 2.11'de gösterilmiştir.



Şekil 2.11 : (8,4) Noktasına Yakın Komşular

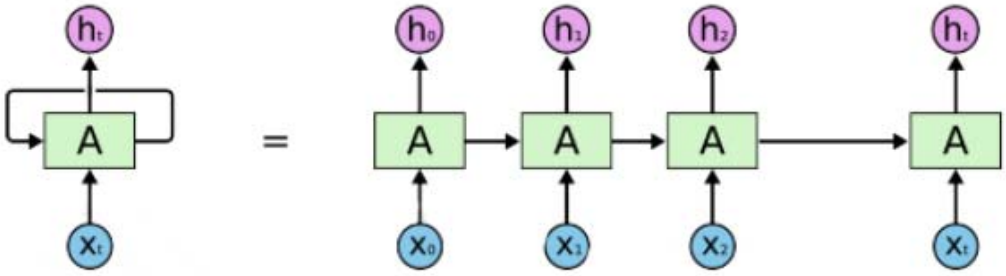
2.5.4 Tekrarlayan Sinir Ağ (RNN)

Yenilenen sinir ağlar olarak da bilinen bu ağ, 1980'lerde ortaya çıkmış dizi üzerinde işlem yapmak üzere geliştirilmiş bir yapay sinir ağdır (Güneş, 2018). RNN'ler, birimler arasındaki bağlantılarla döngü oluşturulan ağlardır. İleri beslemeli sinir ağlara yapı olarak benzemektedirler. İleri beslemeli ağdan farkı, RNN'ler giriş belleklerini girdileri işlemek için kullanabilirler. Bu özelliği sayesinden el yazısı tanıma ve konuşma algılamada diğer sinir ağlardan üstündür (Alp, 2018). Tekrarlayan sinir ağlara ait bir örnek Şekil 2.12'de gösterilmiştir. İleri beslemeli ağlardan farkı Şekil 2.13'de gösterilmiştir.



Şekil 2.12 : Tekrarlayan Sinir Ağ

Kaynak : (Güneş, 2018)



Şekil 2.13 : RNN'lerin İleri Beslemeli Ağlardan Farkı

Kaynak : (Güneş, 2018)

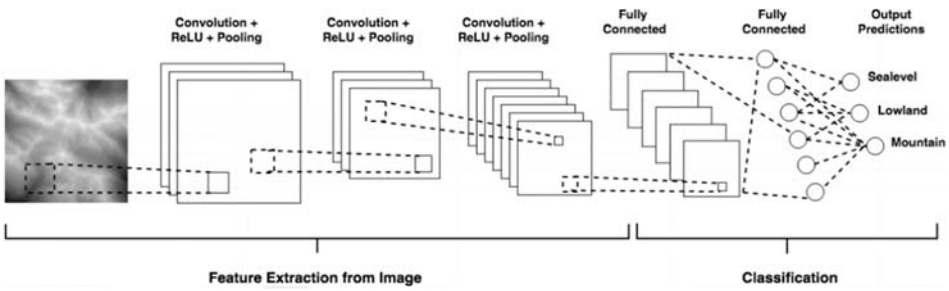
Yukarıda verilen Şekil 2.13'de, ilk önce tekrarlayan yapay sinir ağ gösterilmiştir. Eşitliğin karşı tarafında ise gösterilen RNN yapısının ileri beslemeli ağ karşılığına gelmektedir. RNN modelinde X_t giriş, h_t çıkış ve A RNN sinir ağı göstermektedir. RNN'ler geçmiş ile bağlantı kurup anlamlandırma özelliğinden dolayı zaman bazlı problemlerde yüksek başarım göstermektedir. Ancak geçmiş girdiler hakkında çok

uzun süre bilgi saklayamamaktadırlar. RNN daha uzun süreli bir belleğe sahip olabilmek için tahmin yeteneğini kullanır. Yakın geçmişini anlamamış olsa bile tahmin formüle edebilir.

2.5.5 Konvolüsyonel Sinir Ağ (CNN)

Konvolüsyonel sinir ağları (CNN), MLP'nin bir türüdür. Görme merkezinde bulunan hücreler tüm görseli kapsayacak şekilde alt bölgelere ayrılmıştır. İleri yönlü bir yapay sinir ağ olan CNN, bir veya daha fazla konvolüsyonel katman, alt örnekleme katmanı ve çok katmanlı bir sinir ağı gibi bir veya tamamen bağlı katmanlardan oluşur (LeCun, Bengio, & Hinton, 2015). CNN'lerin önemli özelliklerinden birisi, aynı sayıda gizli birimle bağlı ağlardan daha az sayıda eğitime ve daha az sayıda parametreye sahip olmalarıdır.

CNN algoritmaları ses ve görüntü işleme alanları başta olmak üzere günümüzde bir çok farklı alanda kullanılmakta ve uygulanmaktadır. 2014 yılında yapılan ImageNet Büyük Ölçekli Görsel Tanıma Yarışması'nda, en başarılı sonuçların alan ekiplerin hepsi temelde CNN algoritmalarını kullanmıştır (Girshick, Donahue, Darrell, & Malik, 2014). CNN algoritmaları sağlık sektöründe de başarılı sonuçlar elde etmiştir. 2015 yılında Atomwise şirketinin geliştirmiş olduğu AtomNet, ilaç tasarımı tasarımı alanında ilk derin sinir ağ olmuştur (Wallach, Dzamba, & Heifets, 2015). Ebola, skleroz gibi hastalıkların yeni biyomoleküler keşiflerinde kullanılmıştır (Yosinski, Clune, Nguyen, Fuchs, & Lipson, 2015). Konvolüsyonel sinir ağının yapısı Şekil 2.14'te gösterilmiştir.



Şekil 2.14 : Konvolüsyonel Sinir Ağ Yapısı

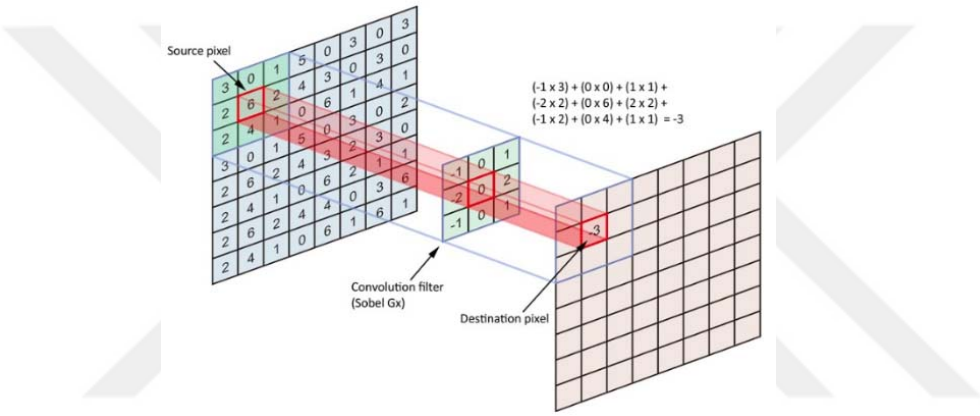
Kaynak : (Rahmon, 2018)

2.5.5.1 Girdi Katmanı

Konvolüsyonel sinir ağın başlangıcıdır. Giriş görüntülerinin piksel değerlerini tutar.

2.5.5.2 Konvolüsyon Katmanı

Konvolüsyonel sinir ağının en önemli temel yapı taşıdır. Dönüşüm katmanı olarak da bilinmektedir. Dönüşüm işlemi, belirli bir filtrenin görüntü üzerinde dolaştırılması şeklinde yapılır. Filtrelerin boyutları 2x2, 3x3, 5x5 matrisler halinde olabilir. Filtreler girdi üzerinde dolaşarak görüntüdeki nitelikleri belirler. Bu şekilde girdi boyutunda yeni bir veri elde edilir (Liu, Shen, & Hengel, 2015). Konvolüsyon katmanında bulunan filtreler öğrenebilir yapıdadırlar. Örnek Şekil 2.15'da gösterilmiştir.



Şekil 2.15 : Konvolüsyon İşlemi

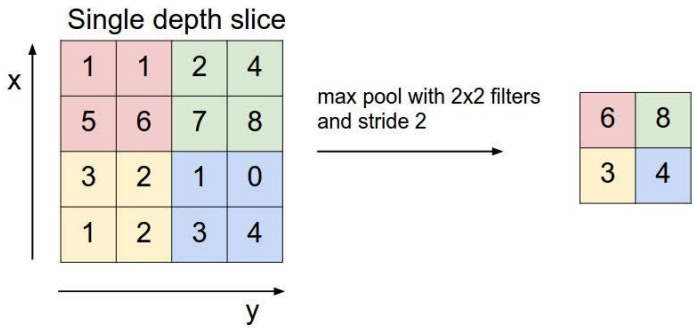
Kaynak : (Url-1)

Konvolüsyon işleminde kullanılan formül Denklem (2.11)'de gösterilmiştir. Formülde *Ç* işlem sonucu elde edilen çıktıyı, *F* filtre matrisinin boyutunu, *B* filtre matrisini ve *G* girdi matrisini göstermektedir (Doğan M. , 2019)

$$Ç_{i,j} = \sum_{k=1}^{F}\sum_{l=1}^{F} B_{k,l} G_{i+k-1,j+l-1} \quad \textbf{(2.11)}$$

2.5.5.3 Havuzlama Katmanı

Bu işlemin bir diğer adı "Aşağı Örnekleme" olarak da geçmektedir. İlk basamakta olan kovolüsyon katmanında olduğu gibi bu katmanda da bir takım filtreler uygulanır. Diğer katmanda olduğu gibi filtreler görüntü üzerinde gezdirilir, çoğunlukla görüntüdeki piksellerin maksimum değerleri alınır (Hijazi, Kumar, & Chris, 2015) Havuzlama işleminin adımları Şekil 2.16'de gösterilmiştir.

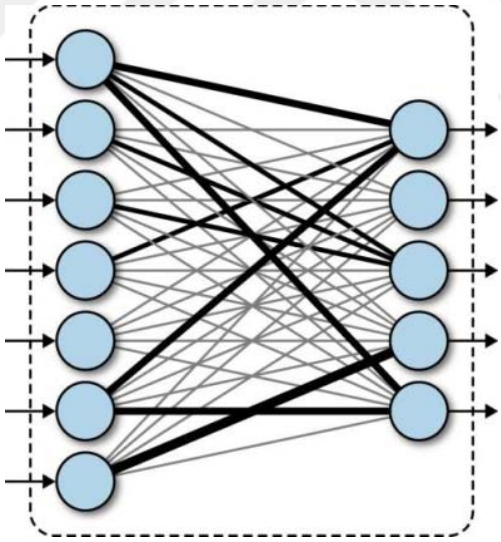


Şekil 2.16 : Havuzlama İşlemi

Kaynak : (Url-2)

2.5.5.4 Tam Bağlantılı Katman

Tam bağlantılı katmanda bulunan nöronlar, YSA'daki gibi önceki katmanlara tam olarak bağlanır. Burada elde edilen çıktıya göre hesaplama yapılır. Tamamen bağlantılı aslında MLP'ye karşılık gelmektedir (Şeker, 2017). Tam bağlantılı katmana örnek Şekil 2.17'de gösterilmiştir.



Şekil 2.17 : Tam Bağlantılı Katman

Kaynak : (Xu, Chen, & Li, 2015)

2.5.6 Görüntü Sınıflandırma Algoritmalarının Performansı

2010 Eylül tarihinde toplanmış yaprak görüntülerinden oluşan bir veriseti ile SVM, ANN ve CNN üzerinden performans testi yapılmıştır. Verisetinin içeriği Çizelge 2.1'de verilmiştir.

Çizelge 2.1 : Yaprak Veri Seti

Yaprak Adı	Kısa Kodu	Örnek Sayısı
Akçaağaç Yaprağı	AP	35
Asplenium Nidus	AN	35
Kırmızı Kızılcık	CS	35
Japon Madımağı	FJ	35
Ginkgo Biloba	GB	35
Duvar Sarmaşığı	HH	35
American Holly	IL	35
Sığla Ağacı	LS	35
Doğu Çınarı	PO	35
Karayemiş	PL	35
Kafkas Ormangülü	RH	35
Yelken Çiçeği	SP	35
Ihlamur	TP	35

SVM algoritmasının en yüksek performans gösteren kernel (Line Kernek) seçimi ile performansı Çizelge 2.2'de gösterilmiştir.

Çizelge 2.2 : SVM Algoritması Performansı

	AP	AN	CS	FJ	GB	HH	IL	LS	PO	PL	RH	SP	TP	T	YD
AP	32	0	0	0	0	0	0	0	0	1	0	0	0	33	96
AN	0	30	0	2	2	0	0	1	0	0	0	0	0	35	85
CS	0	0	28	0	0	0	0	0	1	0	0	1	0	30	93
FJ	0	0	0	30	0	0	4	0	0	0	1	0	0	35	85
GB	0	0	0	1	29	0	0	0	0	0	2	0	0	32	90
HH	0	0	0	0	0	32	0	1	0	2	0	0	0	35	91
IL	0	0	0	0	2	0	30	0	0	0	0	0	0	32	93
LS	1	0	0	0	0	0	0	32	0	0	0	0	0	33	96
PO	0	0	0	0	0	0	0	0	29	0	1	2	0	32	90
PL	1	1	0	0	0	0	0	0	0	33	0	0	0	35	94
RH	0	0	0	0	0	0	1	0	0	0	31	1	0	33	93
SP	0	0	0	0	0	1	0	0	0	0	0	32	0	33	96
TP	0	0	0	0	0	0	0	2	0	0	4	1	28	35	80
T	34	31	29	33	33	33	35	34	30	38	39	37	28	433	
TD	94	96	96	90	87	96	85	94	96	86	79	86	100	91	

ANN algoritması ile yapılmış sınıflandırma çalışmasının performansı

Çizelge **2.3**'te gösterilmiştir.

Çizelge 2.3 : ANN Algoritması Performansı

	AP	AN	CS	FJ	GB	HH	IL	LS	PO	PL	RH	SP	TP	T	YD
AP	10	0	0	0	0	0	0	0	0	0	0	0	0	10	100
AN	0	12	1	0	0	0	0	0	0	0	0	0	0	13	92
CS	0	0	4	0	0	0	0	0	0	0	0	0	0	4	100
FJ	0	0	0	3	0	0	1	0	0	0	0	0	0	4	75
GB	0	0	0	0	2	0	0	0	0	0	0	0	0	2	100
HH	0	0	0	0	0	6	0	0	0	0	0	0	0	6	100
IL	0	0	0	0	0	0	2	0	0	0	0	0	0	2	100
LS	0	0	1	0	0	0	0	9	0	0	0	0	0	10	90
PO	0	0	0	0	0	0	0	0	8	0	0	0	0	8	100
PL	0	0	0	0	0	0	0	0	0	5	0	0	0	5	100
RH	0	0	0	0	0	0	0	0	1	0	6	0	0	6	100
SP	0	0	0	0	0	0	0	0	0	0	0	15	0	17	88
TP	0	0	0	0	0	0	0	0	0	0	0	0	6	6	100
T	10	12	5	3	2	6	3	9	9	5	6	15	8	93	
TD	100	100	80	100	100	100	66	100	88	100	100	100	75	94	

CNN algoritması ile yapılmış sınıflandırma çalışmasının performansı Çizelge 2.4'te gösterilmiştir.

Çizelge 2.4 : CNN Algoritması Performansı

	AP	AN	CS	FJ	GB	HH	IL	LS	PO	PL	RH	SP	TP	T	YD
AP	33	0	0	0	0	1	0	0	0	0	0	0	0	33	97
AN	0	35	0	0	0	0	0	0	0	0	0	0	0	35	100
CS	0	0	34	0	0	0	0	0	0	0	0	0	0	34	100
FJ	0	0	0	35	0	0	1	0	0	0	0	0	0	33	96
GB	0	0	0	0	34	0	0	0	0	0	0	0	0	34	100
HH	0	0	1	0	0	33	0	0	0	0	0	0	0	34	97
IL	1	0	0	0	0	0	30	0	0	0	0	0	0	31	96
LS	0	0	0	1	0	0	0	34	0	0	0	0	0	35	97
PO	0	0	0	0	0	0	0	0	30	0	0	0	0	30	100
PL	0	0	0	0	0	0	0	0	0	29	0	0	0	29	100
RH	1	0	0	0	0	0	0	0	0	0	34	0	0	35	97
SP	0	0	0	0	0	0	0	0	0	0	0	29	0	29	100
TP	0	0	0	0	0	0	0	0	0	0	0	0	34	34	100
T	35	35	35	33	34	34	31	34	30	29	34	29	34	426	
TD	94	100	97	96	100	97	96	100	100	100	100	100	100	99	

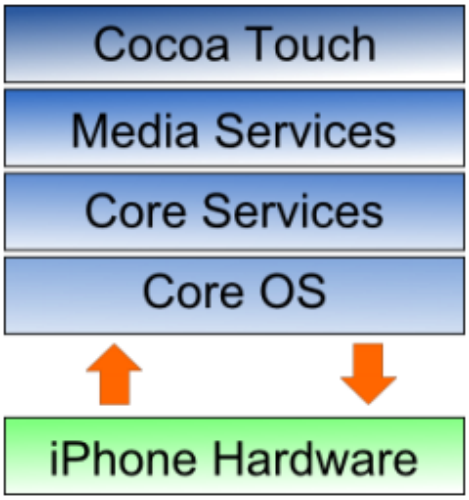
Onüç orman bitkisi üzerinde yapılan bu çalışmada incelenmiş, CNN algoritmasının diğer yöntemlerden daha iyi olduğu belirtilmiştir (Hasan, Ullah, Khan, & Khurshid, 2019).

2.6 iOS Yapısı ve Katmanları

iOS Mac OS'tan türetilmiş bir kapanık kaynak kodlu işletim sistemidir. Uygulama iOS tabanlı olduğu için Swift programlama dilinini destekleyen Xcode kullanılmıştır. 2007 yılında iPhone ilk duyurulduğunda kendine özgü yazılım geliştirme kiti (SDK)'sı yoktu. Ancak 2003 yılında Xcode 1.0'ı yayınladı. Hemen ardından gelen Xcode 1.5 ile daha iyi kod deneyimi ve daha iyi hata ayıklayıcı özellikleri kazandı (Allan, 2013). Optimizasyon algoritmalarının karşılaştırılmasında ise python dili kullanılmıştır. Yapısal olarak derin öğrenme yöntemine uygun olduğu için derin öğrenme projelerinde en çok tercih edilen dillerin başında gelir.

iOS UNIX tabanlı bir işletim sistemidir. İşletim sisteminin temeli C, C++ ve Objective-C'den oluştuğu için yüksek veririmliliğe sahiptir ve birçok işletim sistemine göre daha kararlı çalışmaktadır.

iOS'un sistem yapısı 4 katmandan oluşur. Bunlar; çekirdek CocoaTouch, Medya, Çekirdek ve Çekirdek İşletim Sistemi katmanlarıdır ve Şekil 2.18'da gösterilmiştir. (Lv L. , 2016).



Şekil 2.18 : iOS Katmanları

2.6.1 Cocoa Touch Katmanı

Cocoa Touch katmanı, kullanıcılar için çeşitli kontrollerin tanımlandığı, temel altyapı uygulamalarının sunulduğu, hareket algılayıcılar, çoklu görev gibi özelliklere sahip katmandır. Cocoa Touch katmanında iAD, UIKit, Message UI, Map Kit, GameKit, AddressBook UI, Event Kit UI, Social ve Push Notification Service uygulama çatıları yer almaktadır (Smyth, 2012).

- iAD Framework'ün amacı, geliştiricilerin uygulamalarına banner reklamlarını dahil etmelerini sağlamaktır. Tüm reklamlar Apple'ın kendi reklam hizmeti tarafından sunulur.

- UIKit Framework geniş ve zengin özelliklere sahip Objective-C tabanlı bir programlama arabirimidir. Yazılım geliştirme sırasında kuşkusuz en çok vakit harcanan yerdir. UIKit'in bazı temel özellikler; Kullanıcı arayüzü oluşturma ve yönetimi (metin alanları, butonlar, etikerler, renkler, yazı tipleri vb.) işlemlerinde kullanılır.

- Message UI Framework'ün amacı, uygulama içinde e-posta iletileri oluşturulması ve gönderilmesi için gerekli her şeyi sağlanmasıdır. Bu çatıda e-posta adresleme bilgileri ve mesaj içeriği gibi öğeleri de içinde bulundurur.

- Map Kit Framework, uygulamada harita tabanlı işlemlerin oluşturulmasını sağlayan bir arayüz sağlar.

- GameKit Framework, birden fazla kullanıcı ve cihaz arasında bağlantı ve sesli iletişim sağlayarak, aynı uygulamanın çalıştırılmasını sağlar.

- AddressBook UI Framework'ün amacı, adres defterinde bulunan iletişim bilgilerine uygulama üzerinden erişilmesini, görüntülenmesini, düzenlemesini ve ekleme yapabilmesini sağlar.

- Event Kit UI Framework'ün amacı, uygulama içerisinden takvim verilerine ulaşabilmelerini ve yönetebilmelerini sağlar.

- Social Framework; Facebook, twitter gibi sosyal medya hesapları için URL oluşturmayı ve bu servislere ulaşmayı sağlar.

- Push Notification Service'in amacı, cihazın çalışmadığı yani tuş kilidinde olduğu anda uygulamaların kullanıcıları bir olay hakkında bilgilendirmesine olanak

sağlamaktır. Bu uyarılar mesaj yoluyla olmakla birlikte, ses ve titreşim de bu uyarılara eşlik eder (Smyth, 2012).

2.6.2 Medya Katmanı

Uygulama geliştiriciler için uygulamalarına ses, görüntü veya video gibi özelliklerin eklenmesini sağlayan katmandır. Cocoa Touch katmanında olduğu gibi Medya katmanına kendi içinde birçok çatı bulundurur (Smyth, 2012).

- Core Video Framework, medya katmanı için arabelleğe alma desteği sağlar. Her ne kadar geliştiriciler tarafından kullanılabilse de, bu çatının kullanılması gerekli değildir.

- Core Text Framework, ileri düzey metin düzenlemelerini ve metin içerisindeki font yönetimini sağlar. C tabanlı bir Application Programming Interface (API) yani uygulama programlama arayüzüdür.

- Image I/O Framework'ün amacı, görüntü verilerinin ve görüntü meta verilerinin içe ve dışa aktarılmasını kolaylaştırmaktır. PNG, JPEG, TIFF ve GIF gibi çeşitli görüntü formatlarını desteklemektedir.

- Asset Library Framework'ün amacı, cihaz içerisinde bulunan fotoğraf ve videolara uşalımını ve bu veriler üzerinde değişiklik yapılabilmesine olanak sağlamasıdır.

- Core Graphics Framework, 2 boyutlu grafiklerin oluşturulmasını sağlar. Ayrıca PDF belgesi oluşturma, sunma, vektör tabanlı çizim, saydam katmanlar gibi özellikler de bulunur.

- Core Image Framework, iOS 5 ile ilk defa sunulmuş olan bu çatı, video, görüntü filtreleme ve manipülasyon yeteneği sağlar.

- Quartz Core Framework'ün amacı, iPhone'da animasyon özellikleri sağlamaktır. UIKit Framework'ü ile oluşturulan görsel efektlerin ve animasyonun temelini oluşturur. Objective-C tabanlı bir API'dir.

- OpenGL ES Framework, 2 veya 3 boyutlu grafikler çizmek için kullanılan OpenGL kütüphanesini yüksek performansla kullanılmasını sağlayan çatıdır.

- GLKit Framework, OpenGL ES tabanlı uygulamalar oluşturma görevini kolaylaştırmak için tasarlanmış Objective-C tabanlı bir API'dir.

- NewstandKit Framework, iOS 5 ile birlikte gelen bu çatı, kullanıcıların gazete ve dergilere erişebilmesi için tasarlanmıştır.

- iOS Audio Support, AAC, ALAC gibi ses formatlarını destekleyen çatıdır.

- AV Foundation Framework, ses içeriğinin oynatılmasını, kaydedilmesini ve yönetilmesini sağlamak için tasarlanmış, Objective-C tabanlı bir çatıdır.

- Core Audio Framework, desteklenen ses türlerini, ses dosyalarını ve kaydedilmesini tanımlar. Ayrıca cihazın ses işleme birimlerine erişim sağlar.

- Open Audio Library, yüksek kalitede 3D ses efektleri sağlamak için kullanılan bir çapraz platform teknolojisidir. Genellikle oyunlarda ses efektleri sağlamak için kullanılır.

- Media Player Framework, .mov, .mp4, .m4v, .3gp gibi çeşitli sıkıştırılmış videoların oynatılmasında kullanılır.

- Core MIDI Framework, klavye gibi MIDI uyumlu cihazların kullanımını sağlayan çatıdır (Smyth, 2012).

2.6.3 Core Service Katmanı

Uygulamaların kullandığı, daha önce referans alınan katmanların oluşturulduğu temel yapıları içeren katmandır. Core Service çatıları aşağıda tanımlanmıştır (Smyth, 2012).

- Address Book Framework, uygulamanın adres defteri veri tabanına erişimesini sağlayan, değiştirmesine izin veren çatıdır.

- CFNetwork Framework, C tabanlı bir TCP/IP network protokolü yönetimini sağlar. FTP ve etki alanı ile çalışan uygulamalara, SSL ve TSL bağlantılarının kurulmasına olanak tanır.

- Core Data Framework, veri modelleme ve depolama oluşturmak için kullanılır. Durum bilgisini kaydetme imkanı sunar.

- Core Foundation Framework, Objective-C ile uygulama geliştirenler için NSObject, NSArray, NSDictionary sınıflarını içeren bir çatıdır.

- Core Media Framework, AV Foundation üzerine inşa edilen bir alt katmandır. Uygulama geliştirici tarafından düşük denemetimin gerekli olduğu durumlarda kullanılır.

- Core Telephony Framework, mevcut cep telefonu servis sağlayıcısı hakkında bilgi almak için sorgulama ve telefonla ilgili olaylarda bildirim almasını sağlayan çatıdır.

- EventKit Framework, uygulamalara takvim, hatırlatıcı ve alarm gibi erişimleri sağlamak için tasarlanmış bir API'dir (Smyth, 2012).

2.6.4 Core OS Katmanı

iOS katmanlarının en altında yer alır ve bu nedenle doğrudan cihaz donanımına direkt erişim sağlayabilen katmandır. Düşük düzeyli ağ oluşturma, harici aksesuarlara erişim, bellek yönetimi, dosya sistemi kullanımı ve iş parçacıkları gibi temel işletim sistemleri hizmetlerini sunar Core OS çatıları aşağıda tanımlanmıştır.

- Accelerate Framework, görüntü işleme, sayısal işaret işleme ve matematiksel işlemler yapmak için kullanılan C tabanlı bir API'dir.

- External Accessory Framework, bluetooth üzerinden kablosuz olarak bağlı harici aksesuarları sorgulama ve iletişim kurma olanağı sağlar.

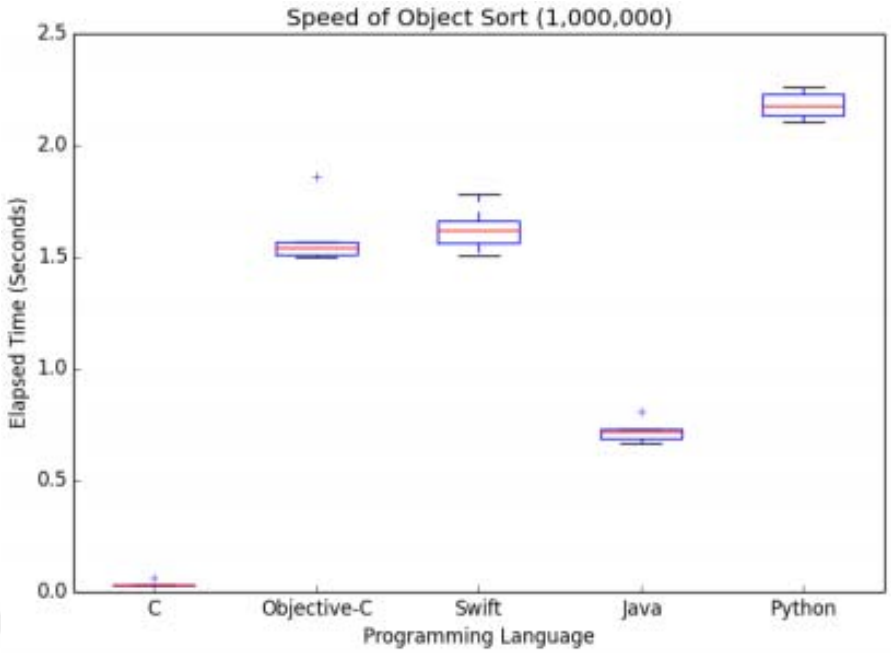
- Security Framework, Uygulama içerisinde bulunan verileri korumak için geliştirilen kütüphaneleri bulundurur. Bu uygulama çatısı ile güvenlik sertifikaları, özel anahtarlar, güvenlik politikaları oluşturulabilir ve yönetimi sağlanabilir.

- System, daha önce de belirtildiği gibi iOS UNIX tabanlı bir işletim sistemidir. Bu katman işletim sistemi çekirdeğini ve aygıt sürücülerini içerir. Çekirdek bellek tahsisi, süreç yaşam döngüsü yönetimi, giriş/çıkış süreçleri arası iletişim, iş parçacığı yönetimi, dosya sistemi yönetimlerinden sorumludur (Smyth, 2012).

2.7 Swift Programlama Dili

Swift programlama dilinin ilk versiyonu olan Swift 1.0 GM, 6 Haziran 2014 tarihinde duyuruldu. GM'nin açılımının Golden Master olduğunu, eklemelerin geliştirmelerin devam edeceğinden dolayı böyle bir isim ile çıkardı. 22 Ekim 2014 yılında ise ikinci versiyon olan Swift 1.1 duyuruldu. Bu versiyon ile bazı protokoller ve bazı işlevsellikler değiştirildi. 8 Nisan 2015'te ise üçüncü versiyon olan Swift 1.2 duyuruldu. Büyük bir güncelleme olarak görüldü ve Xcode 6.3 ile birlikte sulundu. Derleyicide farklı iyileştirmeler yapıldı. En önemli iyileştirmeler, daha iyi derleme hızı, hata ve uyarı mesajlarının iyileştirmesidir (García, Espada, G-Bustelo, & Lovelle, 2015).

Swift dili geliştirilirken, Rust, Haskell, Ruby, Python ve C# gibi birçok programlama dilinden faydalanılmıştır. Yapılan bir testte her bir dilde algoritma yazılmış ve -1000 ile +1000 arasında oluşan 1.000.000 rakamın sıralanması istenmiştir. Performans grafiği Şekil 2.19'de gösterilmiştir.

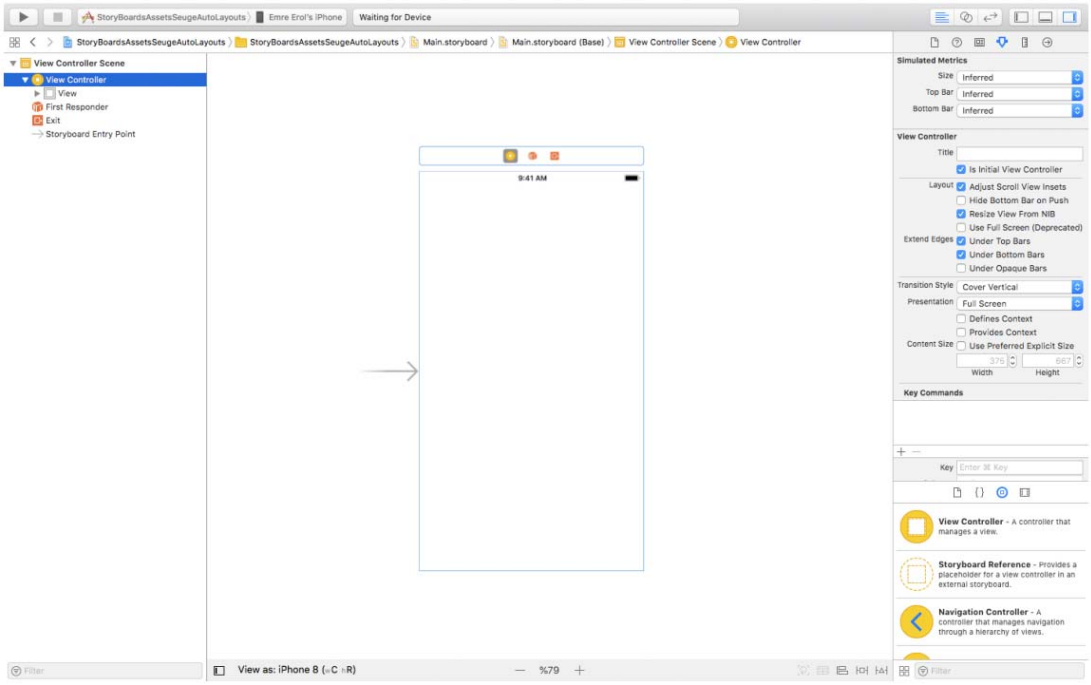


Şekil 2.19 : Rakam Sıralama Performans Değerlendirmesi

Swift ve Objective-C Python'dan 1.4 kat daha hızlı çalışırken, kendi aralarında eşit performansa yakın performans gösterdi. Swift ve Objective-C, Java ve C karşısında ise ortalama iki kat kadar düşük kalmıştır (Wells, 2015).

2.8 Xcode Yazılım Geliştirme Ortamı

Xcode, OS X, iOS, watchOS ve tvOS işletim sistemleri için uygulama geliştirmede kullanılan yazılım geliştirme platformudur. Yalnızca Mac OS X işletimde çalışmaktadır. Xcode Apple tarafından ücretsiz dağıtılmaktadır. Yazılımcıların hazırladığı uygulamalar iTunes Connect platformu aracılığıyla Apple Store'a yüklemektedir. iOS üzerinden geliştirme yapabilmek Xcode vasıtasıyla Swift veya Objective-C dilinde geliştirilmesi tavsiye edilmektedir. Arayüz tasarımı ViewController alanında yapılır. Uygulama içerisindeki elemanlar bu ekranda eklenir. Ekranlar arası geçiş için ise Storyboard ile tanımlanır. Uygulamada birden fazla ekran tasarlanmış olabilir. Bu ekranlar arasında geçişin sağlanması Storyboard ile tanımlanır (Erol, 2018). Storyboard genel görünümü Şekil 2.20'da gösterilmiştir.



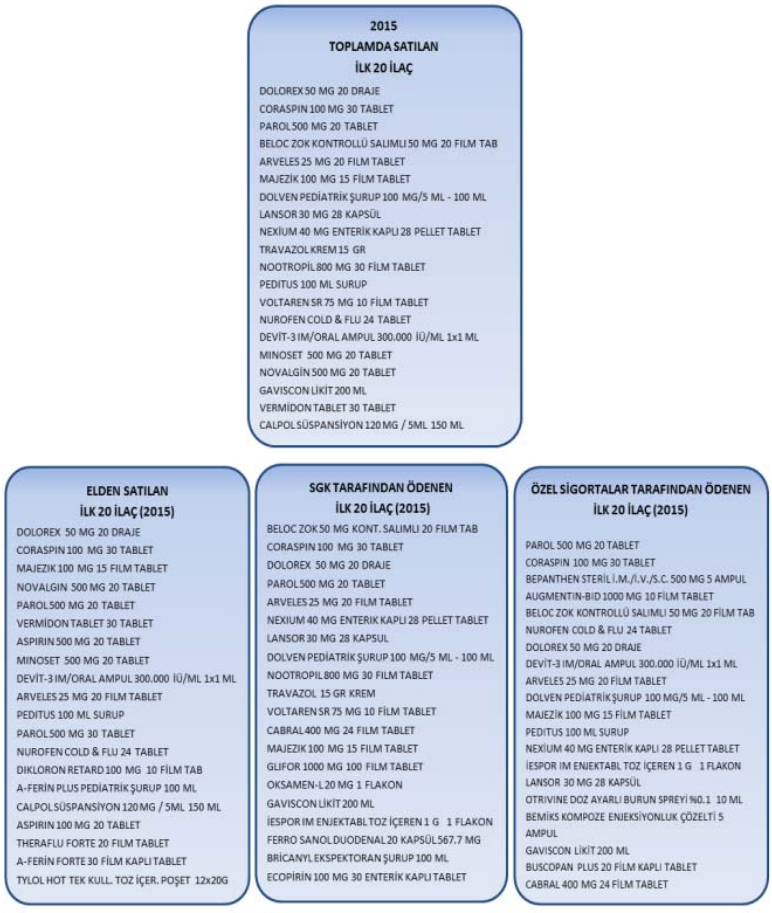
Şekil 2.20 : Storyboard Genel Görünümü

Kaynak : (Erol, 2019)

3. MATERYAL VE YÖNTEM

3.1 Veri Seti Kararı

Veri setinin oluşturulabilmesi için faydalı olabilmesi açısından en çok kullanılan ilaçlar seçilmeye çalışılmıştır. İlaç kararı için Sağlık Bakanlığı Türkite İlaç ve Tıbbi Cihaz Kurumu'nun yayınlamış olduğu Türkiye İlaç Pazarı Gözlem Raporu – 4 incelenmiş, toplamda satılan ilk 20 ilaçlardan yapılmıştır (Bayar, Yalçın, Sinan, Arslan, Tuc, & Atikeler, 2019). 2015 yılı satış hacimlerine göre ülke genelinde en çok satılan ilaçlar Şekil 3.1'de gösterilmiştir. Bu ilaç listesinden uygulama için 10 ilaç seçilmiştir.



Şekil 3.1 : En Çok Satılan İlaç Listesi

Kaynak : (Bayar, Yalçın, Sinan, Arslan, Tuc, & Atikeler, 2019)

3.2 Veri Setinin Oluşturulması

Veri setinin oluşturulabilmesi için farklı açılarda ve ışıklarda ilaç kutularının fotoğrafları çekilmiştir. Uygulama cep telefonu üzerinde çalışacağı için kalite farkı olmaması için veri seti cep telefonu kamerası ile oluşturulmuştur. En doğru kararı alabilmek için 10 sınıfın da verileri eşit tutulmuştur. Veri setinin örnek fotoğrafları Şekil 3.2'de gösterilmiştir.

Şekil 3.2 : Veri Seti Örneği

Veri seti olarak hazırlanan fotoğrafların %10'luk kısmı eğitim sırasında doğrulama için ayrılmıştır. Doğrulama için hazırlanan fotoğraflar, mümkün olduğunda veri setinden bağımsız ve benzeri olanları seçilmiştir. Derin öğrenme algoritmalarında veri setinin büyüklüğü ve çeşitliliği öğrenmenin performansı açısından en önemli faktördür. Sisteme vermiş olduğumuz fotoğraf sayısı ne kadar fazla olursa öğrenme de o kadar başarılı olacaktır. Tek bir arkaplan ve açı ile çekilen 10.000 fotoğrafın eğitim performansını etkilemesinde en ufak bir katkısı olmayacaktır. Fotoğrafların farklı açılarda ve benzersiz arkaplanlarda olması gerekmektedir. Veri setinin fazlalığının tek dezavantajı, saatler süren öğrenme sürecinin daha da uzaması olarak söylenebilir.

3.3 Optimizasyon Algoritmalarının Performansı

Optimizasyon seçiminde uygulanan ve performans değerlendirmeleri yapılan optimizasyon algoritmaları Çizelge 3.1'de verilmiştir.

Çizelge 3.1 : Değerlendirmeye Alınan Algoritmalar

Optimizer	Formula
Rmsprop	$E[g^2]_t = 0.9E[g^2]_{t-1} + 0.1g_t^2$ $\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_{t+\epsilon}}} g_t$ $g_t = \nabla_{\theta_t} J(\theta_t)$
Adam	$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$ $v_t = \beta_2 v_{t-1} + (1-\beta_2) g_{t^2}$ $m'_t = \frac{m_t}{1-\beta_1^t}$, $v'_t = \frac{v_t}{1-\beta_2^t}$ $\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v'_t + \epsilon}} m'_t$
Adagrad	$g_{t,i} = \nabla_{\theta_t} J(\theta_{t,i})$ $\theta_{t+1,i} = \theta_{t,i} - \frac{n}{\sqrt{G_{t,ii}+\epsilon}} g_{t,i}$
Nadam	$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v'_t + \epsilon}} (\beta_1 m'_t + \frac{(1-\beta_1) g_t}{1-\beta_1^t})$

Sınıflandırma ve eğitim öncesinde hangi optimizasyonun daha iyi olduğuna karar verebilmek için PyCharm ile python dilinde Keras ve Tensorflow kütüphaneleri kullanılarak oluşturulan sınıflar performans değerlendirilmesinden geçmiştir.

3.4 Veri Setinin Eğitilmesi

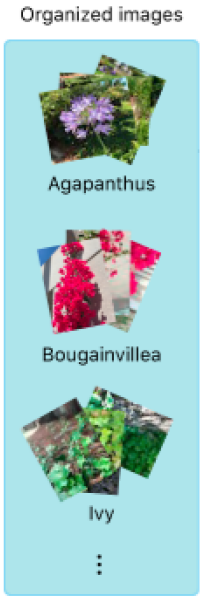
Çalışmada değişik açılardan ve değişik arkaplanların bulunduğu toplamda 1100 adet ilaç kutusu fotoğrafı çekilmiştir. Görüntü işleme ve sınıflandırma Create ML 3 ile yapılmıştır. 2018 yılında Worldwide Developers Conference (WWDC)

konferansında Apple Core ML 3 ile birlikte gelen Create ML 3'ü tanıttı. Create ML Apple'ın programlama dili olan Swift ile oluşturulmuştur (Apple Inc.1, 2018) Create ML ile gelen sınıflandırmalar Çizelge 3.2'de gösterilmiştir.

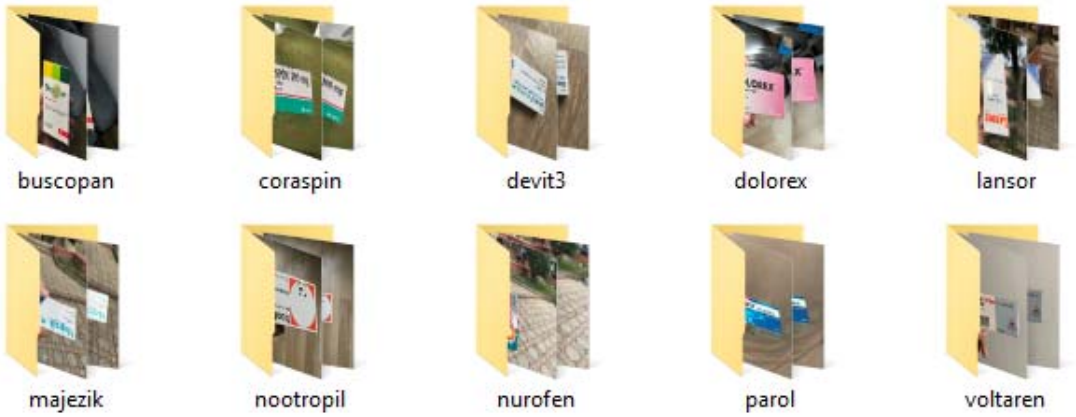
Çizelge 3.2 : Create ML İçerisinde Bulunan Sınıflandırmalar

Görüntü	Image Classifier	Object Detector
Ses	Sound Classifier	
Hareket	Activity Classifier	
Yazı	Text Classifier	Word Tagger
Tablo	Tabular Regressor	Tabular Classifier

Veri setinin doğru bir şekilde eğitim işlemine sokulabilmesi için klasör isimlendirmeleri ve klasör ayrımları çok düzgün yapılması gerekir. Şekil 3.3'de Apple'ın önderdiği klasör yapısı gösterilmiştir (Apple Inc.2, 2019).

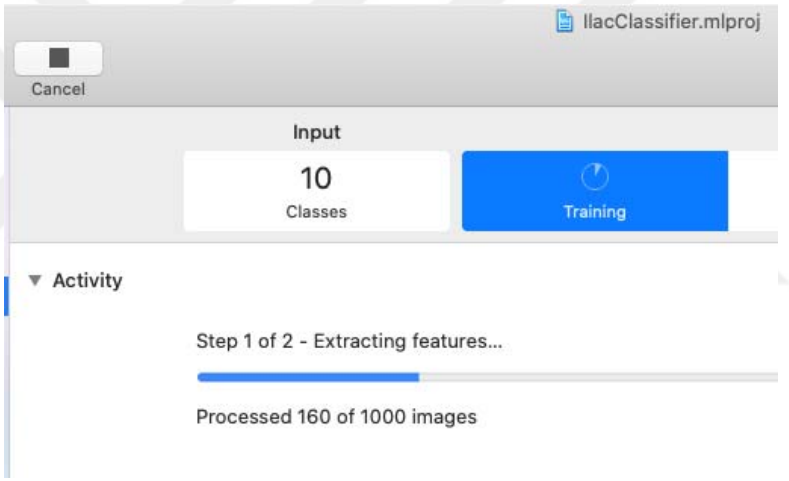


Şekil 3.3 : Klasör Yapısı



Şekil 3.4 : İlaç Modeli Klasör Yapısı

Core ML 3 ile birlikte gelen Create ML uygulaması ile birlikte girdiler ve test verileri uygulamaya gösterilerek Şekil 3.5'te de gösterildi gibi eğitime başlanır.



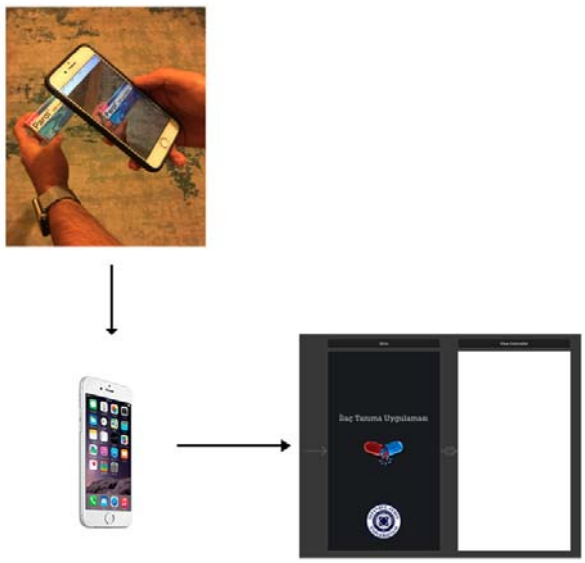
Şekil 3.5 : Create ML ile Eğitim İşlemi

Eğitim süresi, data sayısı, CPU ve GPU gücüne göre değişiklik göstermektedir. Eğitim sürecinde güçlü işlemci ve güçlü ekran kartı kullanılması tavsiye edilir. Bu eğitimde kullanılan bilgisayarın, Core i7 7700HQ CPU, Nvidia GTX 1060 ekran kartı ve 16 Gb RAM özellikleri bulunmaktadır. Güçlü özelliklere rağmen veri setinin büyüklüğü ile eğitimin süresinin saatleri bulabileceği gözlemlenmiştir. Core ML 3'e girdiler ve test verilerinin verdildiği eğitimin sonucu Şekil 3.6'da gösterilmiştir.

Şekil 3.6 : Create ML ile Eğitim Sonrası Model Performansı

3.5 Uygulamanın Geliştirilmesi

Tez kapsamında geliştirilen uygulama, görme engelli bireylerin, başka bir insandan yardım almadan akıllı telefonlarını kullanarak kullandıkları ilaçları tanımlayabilmesidir. Bu uygulama pazarda ciddi bir payı olan ve Siri gibi engelli bireyler için ciddi yardımcı olan iPhone için geliştirilmiştir. Uygulamanın çalışma şekli Şekil 3.7'de gösterilmiştir.



Şekil 3.7 : Uygulamanın Çalışma Şekli

4. DEĞERLENDİRME VE ÖNERİLER

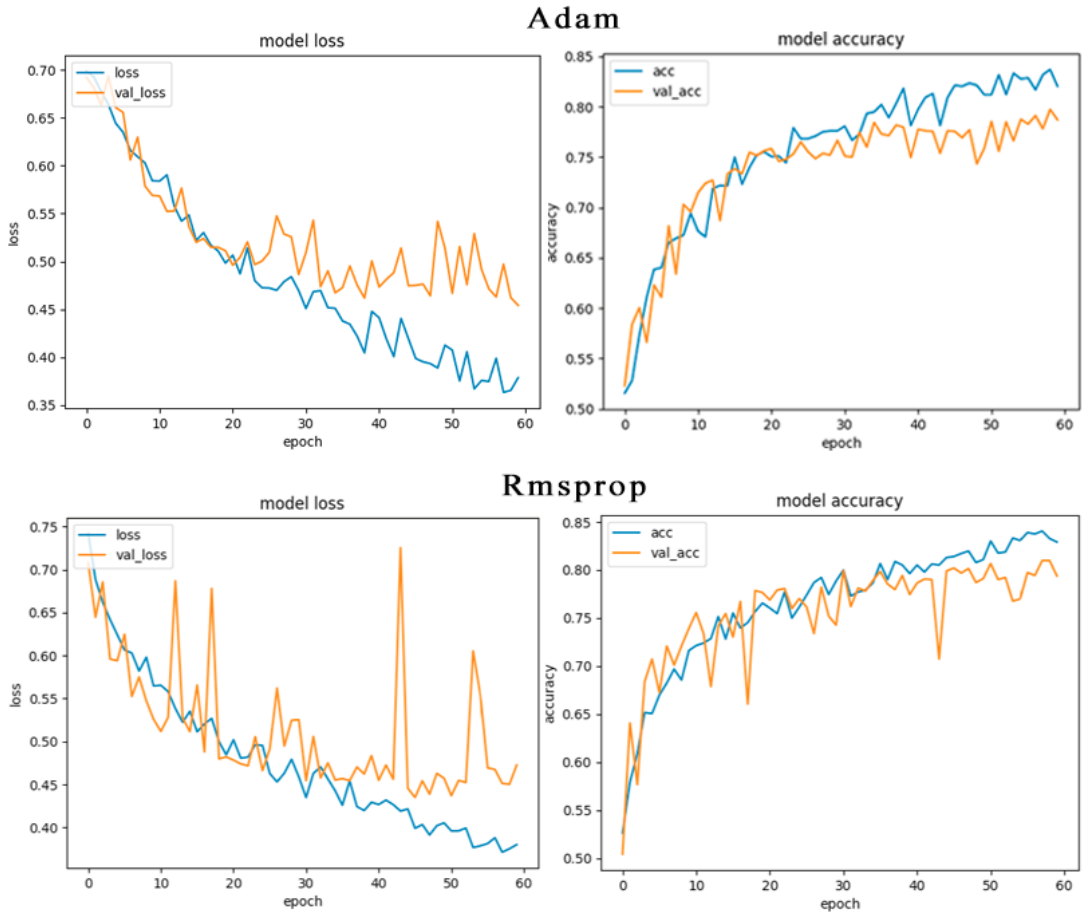
4.1 Değerlendirme

Model oluşturulmadan önce en doğru sonucu alabilmek için Adam, Rmsprop, Adagrad ve Nadam algorithmaları incelenmiş, örnek bir veri seti ve python dilinde yazılım bir kod ile optimizasyon algoritmaları test edilmiştir. Test edilen algoritmaların performansları Çizelge 4.2'de verilmiştir. Elde edilen sonuçlar ilaç tanıma yönteminde kullanılan Adam algoritmasının doğru seçildiğini ve uygulamada kullanılabileceğini göstermiştir.

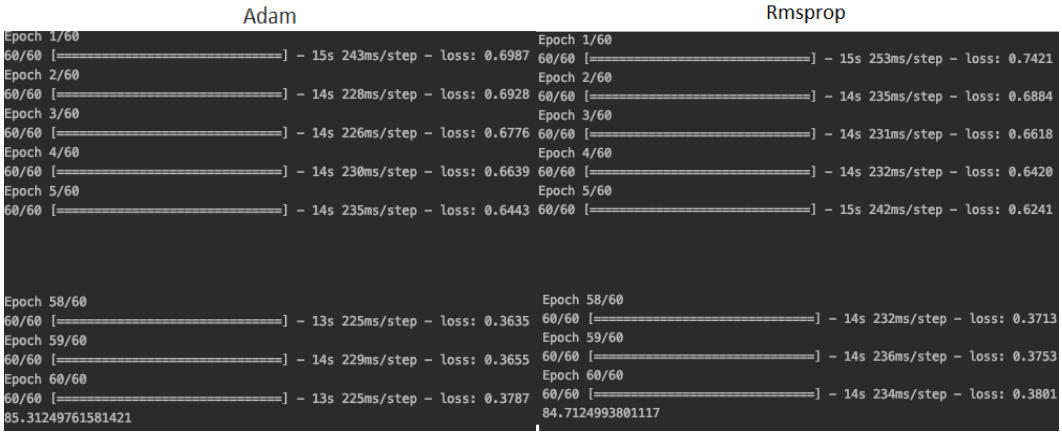
Çizelge 4.1 : Optimizasyon Algoritmaları Performans Karşılaştırması

Optimizer	Epoch Sayısı	Başarım %'si
Adam	60	%85.31
Rmsprop	60	%84.71
Nadam	60	%84.18
Adagrad	60	%72.41

Yukarıdaki sonuçlara bakıldığında başarım olarak en yüksek performansı veren algoritma Adam algoritması oldu. Birbirlerine yakın olan Adam ve Rmsprop algoritmalarının 60 iterasyon ile çalıştırılması sonucunda çıkan başarım ve kayıp grafikleri Şekil 4.1 ve Şekil 4.2'te gösterilmiştir.



Şekil 4.1 : Adam - Rmsprop Karşılaştırması – 1



Şekil 4.2 : Adam - Rmsprop Karşılaştırması - 2

Yapılan performans değerlendirmesi sonucunda Adam optimizasyon algorithmasının, Rmsprop'a göre daha kararlı ve daha performanslı olduğu görülmüştür. Geliştirilecek uygulamada Adam algoritmasını kullanan Create ML3 kullanılmasına bu yöntem ile karar verilmiştir. Uygulamada Create ML3 ile birlikte gelen CNN sinir ağından faydalanılmıştır. Create ML3 ile birlikte gelen

MLFeatureValue sınıfı ile CNN sinir ağında yapılması gereken piksel işlemlerini otomatik olarak manüpüle etmektedir (Apple Inc.4, 2019).

Hali hazırda Adam algoritmasını kullanan Apple Create ML (Apple Inc.3) ile eğitim ve model geliştirilmiştir. Uygulamada yapılan 10 testin sonucu Çizelge 4.2'de gösterilmiştir.

Çizelge 4.2 : Uygulama Testi

İlaç	Doğruluk Oranı
Nurofen	100%
Majezik	80%
Buscopan	80%
Devit-3	80%
Lansor	70%
Parol	90%
Coraspin	70%
Nootropil	100%
Dolorex	80%
Voltaren	90%

Apple iPhone 7 ile çekilen 1100 fotoğraftan oluşan bir veri seti ile model geliştilirmiş, geliştirilen model ile iOS tabanlı uygulama yazılmıştır.

4.2 Öneriler

Görme engelli bireyler için geliştirilen uygulamalar, gelişen teknoloji ve ulaşımı kolay donanımlar ile birlikte daha çok önem kazanmalıdır. Bu çalışmada geliştirilen uygulamanın daha istikrarlı ve daha çok nesneyi tanıyabilir hale gelebilmesi için veri setinin genişletilmesi düşünülmektedir. Veri setindeki sayısal oranda artış, hem sınıf

anlamında daha çok bilgi barındıracak hem de doğruluk oranına pozitif anlamda katkı sağlayacaktır. Uygulamanın dünyanın her yerinde kullanılabilmesi için iOS tarafında geliştirilen yazılıma dil konusunda ek özellikler getirilebilir. Böylelikle Türkçe dışındaki diller için de tanımlaması mümkün olacaktır.

5. SONUÇ

Bu çalışmada, görme engelli bireylerin insan yardımı almaksınız ilaç kullanımını kolaylaştırabilecek bir uygulama geliştirilmiştir. Uygulama temel olarak eğitim sırasında gösterilen nesneleri tanıma ve elde edilen sonucu sesli olarak kullanıcılara iletebilmektedir. Uygulamada kullanılan verisetinin her türlü fiziki ortama, her türlü açıya uygun olmabilmesi için oluşturulmuş ve bu şekilde uygulanmıştır.

Görüntü sınıflandırma algoritmaları arasında yapılan doğruluk oranı karşılaştırmalarında en yüksek oranı CNN algoritması almıştır. Görüntü işleme ve tanıma alanlarında yapılan birçok yarışmada genelde CNN ve CNN'e ek yöntemler kullanılmaktadır. 2017 yılında yapılan ImageNet yarışmasında GBD-Net, 2016 yılında yapılan aynı yarışmada fast-RCNN-GBD-Net ve 2015 yılında yapılan aynı yarışmada ise fast-RCNN algoritması ile yapılan çalışma birinci olmuştur. Bu yarışmalar ve yapılan testler ile birlikte, görüntü işleme konusunda günümüzün en kullanışlı ve en iyi performans veren algoritmasının CNN olduğunu söyleyebiliriz.

Aynı zamanda bu çalışma, Apple'ın sunmuş olduğu Core ML ile birlikte gelen Create ML3 uygulaması ile birlikte görüntü, ses, hareket, yazı gibi sınıflandırma işlemlerinin eski yöntemlere göre daha kolay yapabileceğini göstermektedir. Çalışmada gösterilen adımlarla farkı alanlardan verisetler tanıtılara yeni modeller oluşturulabilir, farklı alanlarda farklı faydalan sağlanabilir.

Günümüzde yapay sinir ağları ile geliştirilen uygulamalar çoğalmıştır. Bu çalışma ortaya çıkacak yeni fikirler için faydalı olacaktır. Özellikle engelli bireylerin yaşamlarını kolaylaştırabilmek adına bu çalışma önemli sonuçlara sahiptir. Bu çalışmada belirtilen yönlemler ile birlikte farklı verisetler işlenerek yeni yöntemler çıkması açısından yararlı olacak ve kaynak oluşturacaktır. Küçük ilaveler ya da sistem entegre edilecek farklı veriler ile birlikte farkı alanlara da uygulanabilir.

KAYNAKLAR

Bibliography

Acharya, T., & Ray, A. K. (2005). *Image processing Principles and Applications.* New Jersey: John Wiley & Sons.

Alemdar, C. (2019). Doğrudan Satış Sektöründe Veri Madenciliği Teknikleri ile Müşteri Kayıp Analizi. Beyken Üniversitesi, Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi.

Allan, A. (2013). *Learning iOS Programming: From Xcode to App Store.* O'Reilly Media Inc.

Alp, E. C. (2018). Makine Öğrenmesi Yöntemleriyle Akan Görüntülerden Otomatik Aktivite Sınıflandırma. Ankara Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Anabilim Dalı, Yüksek Lisans Tezi.

Aytan, A. E., Öztürk, Y., & Örgev, E. K. (1993). Görüntü İşleme. *İ.Ü. Diş Hekimliği Fakültesi Dergisi*, 273-277.

Barbe, J. F., Tewfik, A. H., & Khodursky, A. B. (2007). Transcription Factor Discovery using Support Vector Machines and Heterogeneous Data. IEEE.

Bayar, B., Yalçın, E., Sinan, B., Arslan, A., Tuc, B., & Atikeler, E. K. (2019). *Türkiye İlaç Pazarı Gözlem Raporu-4 Satış Hacmi ve Değeri Açısından 2015 ve 2016 Yılı Pazar Durumu.* Ankara: Sağlık Bakanlığı.

Bengio, Y., Boulanger-Lewandowski, N., & Pascanu, R. (2012). Advances in Optimizing Recurrent Networks. arXiv:1212.0901.

Çayıroğlu, İ. (2018). Görüntü İşleme. Karabük Üniversitesi, Mühendislik Fakültesi .

Doğan, G. (2010). Yapay Sinir Ağları Kullanılarak Türkiye'deki Özel Bir Sigorta Şirketinde Portföy Değerlendirmesi. Hacettepe University, Ankara.

Doğan, M. (2019). Bitkilerde Görülen Hastalıkların Derin Öğrenme Yöntemleriyle Tespiti ve Sınıflandırılması. Yalova Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Yüksek Lisans Tezi.

Erdil, F., & Elbaş, N. Ö. (2012). *Cerrahi Hastalıkları Hemşireliği.* Ankara: Aydoğdu Ofset.

Erol, E. (2018). Bilişim Toplumu Bağlamında Mobil Uygulama Geliştirme Platformu Olarak Xcode. *Yeni Medya Elektronik Dergi*, 160-167.

Erol, E. (2019). Swift ile iOS Uygulama Geliştirme: Yeni Başlayanlar İçin. s. 17.

Fernandez, C., E., S., Martin, J., & A.J., S. (2006). Neural networks for animal science applications: Two case studies. *Expert Systems with Applications*, 444-450.

García, C. G., Espada, J. P., G-Bustelo, C. B., & Lovelle, J. M. (2015). Swift vs. Objective-C: A New Programming Language. *IJIMAI*, 74-81.

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. *arXiv.org*, arXiv:1311.2524.

Güneş, A. (2018). Türkçe'de Varlık İsmi Tanıma. İstanbul Teknik Üniversitesi, Bilgisayar Mühendisliği Anabilim Dalı, Yüksek Lisans Tezi.

Hasan, M. u., Ullah, S., Khan, M. J., & Khurshid, K. (2019). Comparative Analysis of SVM, ANN and CNN for Classifying Vegetation Species Using Hyperspectral Thermal Infrared Data. *ISPRS*, 1861-1868.

Kadiroğlu, Z. (2019). Histopatolojik Meme Kanseri Görüntülerinin Evrişimsel Sinir Ağları Kullanılarak Sınıflandırılması. Fırat Üniversitesi, Elektrik ve Elektronik Anabilim Dalı, Yüksek Lisans Tezi.

Kavuncu, S. K. (2018). Makine Öğrenmesi ve Derin Öğrenme: Nesne Tanıma Uygulaması. Kırıkkale Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Anabilim Dalı, Yüksek Lisans Tezi.

Kim, P. (2017). *MATLAB Deep Learning: With Machine Learning, Neural Networks and Artificial Intelligence.* Apress.

Kingma, D. P., & Ba, J. L. (2017). Adam: A Method for Stochastic Optimization. *ArXiv e-prints*.

Köklü, M. (2014). Sınıflandırma Problemlerinde Kural Çıkarımı için Yeni Bir Yöntem Geliştirilmesi ve Uygulamaları. Selçuk Üniversitesi, Fen Bilimleri Enstitüsü, Doktora Tezi.

Kurt, F. (2018). Evrişimli Sinir Ağlarında Hiper Parametrelerin Etkisinin İncelenmesi. Hacettepe Üniversitesi, Fen Bilimleri Enstitüsü.

Kuş, Z. (2019). Mikrokanonikal Optimizasyon Algoritması ile Konvolüsyonel Sinir Ağlarında Hiper Parametrelerin Optimize Edilmesi. Fatih Sultan Mehmet Vakıf Üniversitesi, Yüksek Lisans Tezi.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep Learning. *Nature*, 436-444.

Liu, L., Shen, C., & Hengel, A. v. (2015). The Treasure beneath Convolutional Layers: Cross-convolutional-layer Pooling for Image Classification. *IEEE Journals*, 4749-4757.

Lv, H., & Liu, J. (2016). iOS System Based on Sina Microblogging Mobile APP Design. *6th International Conference on Electronic, Mechanical, Information and Management* (s. 1631-1635). Atlantis Press.

Lv, L. (2016). iOS System Based on Sina Microblogging Mobile APP Design. *6th International Conference on Electronic, Mechanical, Information and Management* (s. 1631-1635). Atlantis Press.

Özkan, Y. (2008). *Veri Madenciliği Yöntemleri.* Papatya Bilim.

Öztemel, E. (2006). *Yapay Sinir Ağları.* Papatya Yayıncılık.

Özveren, U. (2006). Pem Yakıt Hücrelerinin Yapay Sinir Ağları İle Modellenmesi. Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü.

Rahmon, G. (2018). Evaluation of Procedurally Generated Terrains via Artificial and Convolutional Neural Networks. İzmir Ekonomi Üniversitesi.

Ruder, S. (2016). An overview of gradient descent optimization. arXiv:1609.04747.

Sinecen, M. (2016). Digital Image Processing with MATLAB. *IntechOpen*, DOI: 10.5772/63028.

Smyth, N. (2012). *iPhone IOS 6 Development Essentials.* CreateSpace Independent Publishing Platform.

Şeker, A. (2017). Derin Öğrenme Yöntemleri ve Uygulamaları Hakkında Bir İnceleme. Yıldız Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Doktora Semineri, Bilgisayar Mühendisliği Anabilim Dalı.

Tekeli, K., & Aşlıyan, R. (2016). Çok Katmanlı Algılayıcı, K-NN ve C4.5 Metotlarıyla İstenmeyen E-postaların Tespiti. Adnan Menderes Üniversitesi.

Waheed, M. E., Behery, G. M., & Elshewey, A. M. (2016). Determining Mean Square Error and Standard Deviation Error for Measurement of Non-Invasive Blood Pressure Using ANN. *European Journal of Scientific Research*, 209-222.

Wallach, I., Dzamba, M., & Heifets, A. (2015). "AtomNet: A Deep Convolutional Neural Network for Bioactivity Prediction in Structure-based Drug Discovery. arXiv:1510.02855v1.

Wells, G. (2015). The Future of iOS Development: Evaluating the Swift Programming Language. Claremont McKenna College.

Xu, B. W., Chen, T., & Li, M. (2015). Empirical Evaluation of Rectified Activations in Convolution Network. arXiv:1505.00853v2.

Yavuz, S., & Deveci, M. (2012). İstatiksel Normalizasyon Tekniklerinin Yapay Sinir Ağın Performansına Etkisi. *Erciyes Üniversitesi İktisadi ve İdari Bilimler Fakültesi Dergisi*, 167-187.

Yazan, E., & Talu, F. M. (2017). *Stokastik Dereceli Alçalma Yöntemi Temelli Optimizasyon Tekniklerinin Karşılaştırılması.*

Yıldırım, S. (2006). Arıza Teşhisinde Destek Vektör Makinelerinin Kullanımı. Fırat Üniversitesi, Bilgisayar Mühendisliği Anabilim Dalı, Yüksek Lisans Tezi.

Yılmaz, R. (2013). Türkçe Dökümanların Sınıflandırılması. Adnan Menderes Üniversitesi Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, Aydın.

Yosinski, J., Clune, J., Nguyen, A., Fuchs, T., & Lipson, H. (2015). Understanding Neural Networks Through Deep Visualization. *Deep Learning Workshop, 31 st International.* Lille.

Yücel, C. Y., & Acatürk, C. (2006). Görme Engelliler için Web Sayfalarında Erişilebilirliğin Sağlanması. *8. Akademik Bilişim Konferansı*, (s. 09-11).

İnternet Kaynakları:

Apple Inc.1 *Create ML*. December 02, 2019 tarihinde Create ML: https://developer.apple.com/documentation/createml adresinden alındı. Erişim tarihi: 02.09.2019

Apple Inc.2 *Training a Create ML Model to Classify Flowers*. Eylül 23, 2019 tarihinde Training a Create ML Model to Classify Flowers: https://developer.apple.com/documentation/vision/training_a_create_ml_model_to_classify_flowers adresinden alındı. Erişim tarihi: 17.09.2019

WHO. *Universal eye health: a global action plan 2014-2019.* Ekim 14, 2019 tarihinde WHO International: https://www.who.int/blindness/AP2014_19_English.pdf?ua=1 adresinden alındı. Erişim tarihi: 07.10.2019

Apple Inc.3 *MLParameterKey* https://developer.apple.com/documentation/coreml/mlparameterkey adresinden alındı. Erişim Tarihi: 19.10.2019

Apple Inc.4 MLFeatureValue https://developer.apple.com/documentation/coreml/mlfeaturevalue adresinden alındı. Erişim Tarihi: 18.10.2019

URL-1 *Exploring Convolutional Neural Networks (CNNs) from an iOS Developer's Perspective* https://heartbeat.fritz.ai/exploring-convolutional-neural-networks-cnns-from-an-ios-developers-perspective-162664130d5b adresinden alındı. Erişim Tarihi: 18.08.2019

URL-2 *Pooling Layer* https://harangdev.github.io/deep-learning/convolutional-neural-networks/24/ adresinden alındı. Erişim Tarihi: 18.08.2019

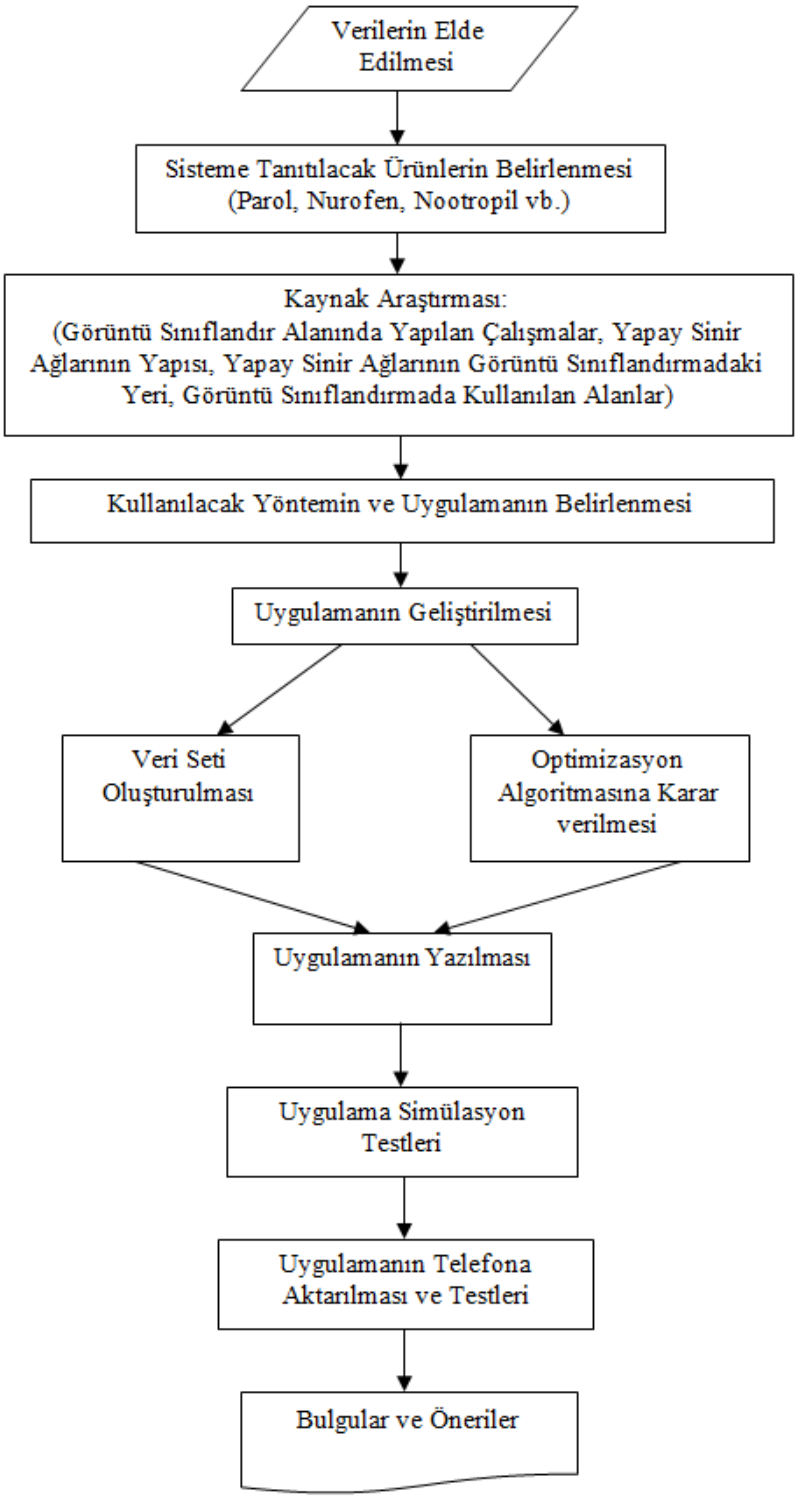
Hijazi, S., Kumar, R., & Chris, R. (2015). *Using Convolutional Neural Networks for Image Recognition*. https://ip.cadence.com/uploads/901/cnn_wp-pdf adresinden alındı. Erişim Tarihi: 03.09.2019

EKLER

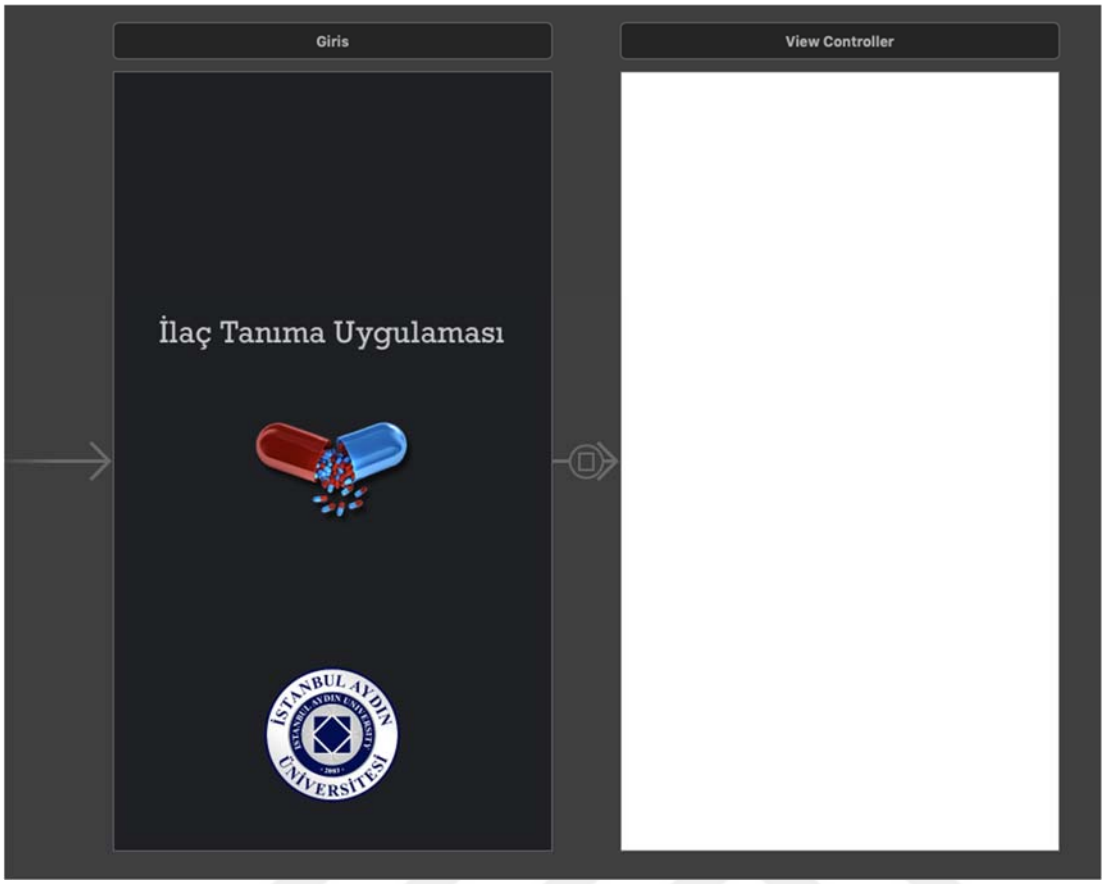
EK A : Şekiller

EK B : Kodlar

EK A



Şekil A.1 : Tez Çalışması Akış Diyagramı



Şekil A.2 : Xcode Main.storyboard Görüntüsü

EK B

iOS Uygulama Geliştirme için Kullanılan Kod

```
import UIKit
import AVKit
import Vision
import CoreML
import AVFoundation

class ViewController: UIViewController,
AVCaptureVideoDataOutputSampleBufferDelegate {

    @IBOutlet weak var innerView: UIView!
    @IBOutlet weak var viewLabel: UILabel!

    var previewLayer: AVCaptureVideoPreviewLayer?

    override func viewDidLoad() {
      super.viewDidLoad()

        updateLabel(newLabel: "new label")

        //Kamera açılışı
        let captureSession = AVCaptureSession()
        captureSession.sessionPreset = .photo

        guard let captureDevice = AVCaptureDevice.default(for: .video) else { return }
        guard let input = try? AVCaptureDeviceInput(device: captureDevice) else {
return }
        captureSession.addInput(input)
```

```
        captureSession.startRunning()

        self.previewLayer = AVCaptureVideoPreviewLayer(session: captureSession)
        self.previewLayer?.frame.size = self.innerView.frame.size
        self.previewLayer?.videoGravity = AVLayerVideoGravity.resizeAspectFill
        self.innerView.layer.addSublayer(self.previewLayer!)
        self.previewLayer?.frame = view.frame



        //video pencere değerlerinin alınması
        let dataOutput = AVCaptureVideoDataOutput()
        dataOutput.setSampleBufferDelegate(self, queue: DispatchQueue(label:
"VideoQueue"))
        captureSession.addOutput(dataOutput)


    }

    override func viewDidAppear(_ animated: Bool) {
        super.viewDidAppear(animated)
        self.previewLayer?.frame.size = self.innerView.frame.size
    }


    func captureOutput(_ output: AVCaptureOutput, didOutput sampleBuffer:
CMSampleBuffer, from connection: AVCaptureConnection) {


        guard let pixcelBuffer:CVPixelBuffer =
CMSampleBufferGetImageBuffer(sampleBuffer)
            else { self.quickErr(myLine: #line,inputStr: "") ; return }
        guard let model =  try? VNCoreMLModel(for: ilacModel10Crop().model)
            else { self.quickErr(myLine: #line,inputStr: "") ; return }
        let request = VNCoreMLRequest(model: model) { (finishedReq, err) in
```

```
        guard let results = finishedReq.results as? [VNClassificationObservation]
            else { self.quickErr(myLine: #line,inputStr: "") ; return }
        guard let firstObservation = results.first
            else { self.quickErr(myLine: #line,inputStr: "") ; return }

        var myMessage = ""
        var myConfident = 0

            if (firstObservation.confidence > 0.2 ) {
                myConfident = Int ( firstObservation.confidence * 100 )
                let myIdentifier = firstObservation.identifier.split(separator: ",")
                myMessage = "Bu ilaç : \(myIdentifier[0]) "
                }
             else {
                return
            }

            print(myMessage)
            self.updateLabel(newLabel: myMessage)
            if ( myConfident == 100 ){
                self.readyMe(myText: myMessage, myLang: "tr_TR")
                sleep(3)
            }
        }

        // Resim dosyası analizi
        try? VNImageRequestHandler(cvPixelBuffer: pixcelBuffer, options:
[:]).perform([request])
    }

    func readyMe(myText :String , myLang : String ) {
```

```
        let uttrace = AVSpeechUtterance(string: myText )
        uttrace.voice = AVSpeechSynthesisVoice(language: myLang)
        uttrace.rate = 0.5

        let synthesizer = AVSpeechSynthesizer()
        synthesizer.speak(uttrace)
    }

    func quickErr(myLine: Int , inputStr : String = "" ) {
        print("===> Guard Error \(inputStr) :\n    file:\(#file)\n    line:\(myLine)\n function:\(#function) ")
    }

    func updateLabel(newLabel: String){

        DispatchQueue.main.async { // Doğru ise
            self.viewLabel?.text = "[ " + newLabel + " ]"
        }
    }
}
```

Optimizasyon Karşılaştırılması için Kullanılan Python Dilinde Yazılmış Kod

```
# Import libraries from Keras.

import tensorflow as tf
import matplotlib.pyplot as plt
from tensorflow.keras import layers
from tensorflow.keras import preprocessing
from tensorflow.keras import models
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Flatten
from tensorflow.keras.preprocessing import image
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.models import Sequential
from keras.optimizers import SGD
import numpy as np

# CNN Konfigürasyonu

classifier = Sequential()

classifier.add(Conv2D(32, (3, 3), input_shape=(64, 64, 3),
               activation='relu'))

classifier.add(MaxPooling2D(pool_size=(2, 2)))

# İkinci CNN katmanın eklenmesi

classifier.add(Conv2D(32, (3, 3), activation='relu'))
```

```
classifier.add(MaxPooling2D(pool_size=(2, 2)))
classifier.add(Flatten())

classifier.add(Dense(units=128, activation='relu'))
classifier.add(Dense(units=1, activation='sigmoid'))

# CNN’in Derlenmesi ve Eğitim

classifier.compile(optimizer='adam', loss='binary_crossentropy',
                   metrics=['accuracy'])
from keras.preprocessing.image import ImageDataGenerator
train_imagedata = ImageDataGenerator(rescale=1. / 255, shear_range=0.2,
        zoom_range=0.2, horizontal_flip=True)
test_imagedata = ImageDataGenerator(rescale=1. / 255)
training_set = \
    train_imagedata.flow_from_directory('data/train'
        , target_size=(64, 64), batch_size=32, class_mode='binary')
val_set = \
    test_imagedata.flow_from_directory('data/validation'
        , target_size=(64, 64), batch_size=32, class_mode='binary')
history=classifier.fit_generator(training_set, steps_per_epoch=60, epochs=60,
                         validation_data=val_set,
                         validation_steps=60)

loss, acc = classifier.evaluate(training_set, verbose = 0)
print(acc * 100)

# Kayıp
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
```

```
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['loss', 'val_loss'], loc='upper left')
plt.show()

# Doğruluk
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['acc', 'val_acc'], loc='upper left')
plt.show()
```

ÖZGEÇMİŞ

Ad-Soyad **:** Onur YILMAZ

Doğum Yılı ve Yeri **:** 1990 - İstanbul

E-posta **:** info@yilmazonur.com

Öğrenim Durumu

- **Ön Lisans** **:** Bilecik Üniversitesi – Söğüt Meslek Yüksekokulu – Bilgisayar Programcılığı
- **Lisans** **:** Anadolu Üniversitesi – İşletme Fakültesi
- **Yüksek Lisans** **:** İstanbul Aydın Üniversitesi – Lisansüstü Eğitim Enstitüsü Bilgisayar Mühendisliği Programı

Mesleki Deneyim

- **Demirer Kablo Tesisleri Sanayi ve Ticaret A.Ş.**
  - 2010 – 2011 : Bilgi Teknolojileri Uzmanı
  - 2011 – 2016 : BT Bilgi Güvenliği ve Network Uzmanı
  - 2016 – 2018 : BT Kıdemli Bilgi Güvenliği ve Network Uzmanı
  - 2018 – 2019 : BT Operasyonları Şefi
  - 2019 – Günümüz : Bilgi Teknolojileri Müdürü

Yabancı Diller

- İngilizce