T.C.
İSTANBUL AYDIN ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YAPAY ZEKA TEKNİKLERİ KULLANARAK KEMİK YAŞI TESPİTİ ÜZERİNDE BİR UYGULAMA

YÜKSEK LİSANS TEZİ

Nur ZAKİROĞLU

Bilgisayar Mühendisliği Ana Bilim Dalı
Bilgisayar Mühendisliği Programı

HAZİRAN 2019

T.C.
İSTANBUL AYDIN ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YAPAY ZEKA TEKNİKLERİ KULLANARAK KEMİK YAŞI TESPİTİ ÜZERİNDE BİR UYGULAMA

YÜKSEK LİSANS TEZİ

Nur ZAKİROĞLU
(Y1713.010027)

Bilgisayar Mühendisliği Ana Bilim Dalı
Bilgisayar Mühendisliği Programı

Tez Danışmanı: Prof. Dr. Ali GÜNEŞ

HAZİRAN 2019

T.C.
İSTANBUL AYDIN ÜNİVERSİTESİ
FEN BİLİMLER ENSTİTÜSÜ MÜDÜRLÜĞÜ

Yüksek Lisans Tez Onay Belgesi

Enstitümüz Bilgisayar Mühendisliği Ana Bilim Dalı Bilgisayar Mühendisliği Tezli Yüksek Lisans Programı Y1713.010027 numaralı öğrencisi **Nur ZAKİROĞLU MOUKET** ' ın **"YAPAY ZEKA TEKNİKLERİ KULLANILARAK KEMİK YAŞI TESPİTİ VE BİR UYGULAMA"** adlı tez çalışması Enstitümüz Yönetim Kurulunun 26/06/2019 tarih ve 2019/13 sayılı kararıyla oluşturulan jüri tarafından oybirliği ile Tezli Yüksek Lisans tezi olarak kabul.....edilmiştir.

Öğretim Üyesi Adı Soyadı	İmzası
Tez Savunma Tarihi : 09/07/2019	
1)Tez Danışmanı: Prof. Dr. Ali GÜNEŞ	.....
2) Jüri Üyesi : Dr. Öğr. Üyesi Ahmet GÜRHANLI	.....
3) Jüri Üyesi : Doç. Dr. Metin ZONTUL	.....

Not: Öğrencinin Tez savunmasında **Başarılı** olması halinde bu form **imzalanacaktır.** Aksi halde geçersizdir.

YEMİN METNİ

Yüksek Lisans tezi olarak sunduğum "Yapay Zekâ Teknikleri Kullanarak Kemik Yaşı Tespiti Üzerinde Bir Uygulama" 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. (…/…/2019)

Nur ZAKİROĞLU

ÖNSÖZ

Yüksek lisans eğitimim sürecinde her türlü kolaylığı ve anlayışı benden esirgemeyen ve bilimsel katkılarıyla beni yönlendiren, ders alma sürecinde alanındaki derin bilgi ve motivasyonla makine öğrenmesi konusunda ufkumu açan ve bu alanda çalışmak için bende eşsiz bir istek uyandıran tez danışmanım sayın Prof. Dr. Ali GÜNEŞ', ders alma sürecinde eşsiz bilgilerinden yaralandığım Aydın Üniversitesi Bilgisayar Mühendisliğindeki tüm değerli hocalarıma sonsuz minnet ve şükranlarımı sunarım. Ayrıca tez çalışmam sürecinde yanımda olan, bana maddi ve manevi desteklerini hiçbir zaman esirgemeyen babam Fevzi ZAKİROĞLU, annem Hüda ZAKİROĞLU ve işlerime herzaman koşturan kardeşim Abdullah ZAKİROĞLU'ya, yaşama kaynağım olan kızım Sidra'ya sonsuz teşekkürlerimi sunuyorum.

Haziran 2019 **Nur ZAKİROĞLU**

İÇİNDEKİLER **Sayfa**

ŞEKİL LİSTESİ

Sayfa

YAPAY ZEKA TEKNİKLERİ KULLANARAK KEMİK YAŞI TESPİTİ ÜZERİNDE BİR UYGULAMA

ÖZET

Kemik yaşı, çeşitli hastalıkların teşhisinde ve tedavinin zamanlamasının belirlenmesinde etkili bir göstergedir. Bir kişinin iki yaşı, kronolojik yaş ve kemik yaşı vardır. Kronolojik yaş, kişinin doğum tarihinden itibaren belirlenen gerçek yaştır. Kemik yaşı ise kişinin kemiklerinin olgunlaşma derecesini tanımlar.
Büyüme bozuklukları, kromozomal bozukluklar, endokrin bozukluklar ve endokrinolojik problemler gibi birçok hastalık kemik yaşı ile kronolojik yaş arasındaki tutarsızlık ile keşfedilebilir.
Bu çalışmada, öncelikle kişinin sol eline yönelik olarak x-ışınlarını kullanarak sayısal görüntü alınmaktadır. Daha sonra, bu görüntü üzerinde makine öğrenmesi teknikleri kullanarak kişinin kemik yaşı tespit edilmektedir.
Çalışma kapsamında 2500 çocuğa ait sayısal el görüntüsü verileri ele alınmıştır. Bu verilere, yapay zeka tekniği olarak, CNN VGG16 ve CNN Inception V3 kümeleme algoritmaları uygulanmış, sonrasında da elde edilen kümeler değerlendirilmiştir.

Anahtar Kelimeler: *Makine Öğrenmesi, Görüntü İşleme, CNN Algoritması, Kemik Yaşı*

AN APPLICATION ON BONE AGE DETECTION USING ARTIFICIAL INTELLIGENCE TECHNIQUES

ABSTRACT

Bone age is an effective indicator for the diagnosis of various diseases. A person has two ages, chronological age and bone age. Chronological age is the actual age determined from the date of birth of the person. Bone age describes the degree of maturation of the Person's bones.
Many diseases, such as growth disorders, chromosomal disorders, endocrine disorders, and endocrinological problems, can be discovered by finding inconsistency between bone age and chronological age.
In this article bone age is determined by applying image processing techniques on people hands x-rays with techniques of machine learning.
The hand x-rays image data of 2500 children were studied. CNN VGG16 and CNN Inception V3 clustering algorithms were applied to these data and then clusters were evaluated.

Keywords: *Machine Learning, Image Processing, CNN Algorithm, Bone Age*

1.GİRİŞ

Kemik yaşı değerlendirmesi sıklıkla sadece pediatrik endokrinolojide değil aynı zamanda pediatrik ortopedi lerde de yapılır. Kemik yaşı, çeşitli hastalıkların teşhisinde ve tedavinin zamanlamasının belirlenmesinde etkili bir göstergedir. Kemik yaşı değerlendirmesinin amacı, büyümeyi ve olgunluğu değerlendirmek ve pediatrik hastalıkları teşhis ve yönetmektir. Bu nedenle, kemik yaşı değerlendirmesinin doğruluğu çok önemlidir. Manuel kemik yaşı değerlendirme yöntemleri uzun süredir kullanılmasına rağmen, bu yöntemlerle ilgili temel problem gözlemciler arası değişkenlik göstermektedir.

Son zamanlarda, kemik yaşı değerlendirmesi için çeşitli bilgisayarlı sistemler geliştirilmiştir. Bu makalede, değerlendirme yöntemlerinde ve kemik çağının klinik uygulamalarındaki ilerlemeyi açıklanmaktadır. Bu makalede x-ışınlarını kullanarak bir kişinin kemik yaşını makine öğrenmesi teknikleri ile x-ışınları üzerinde görüntü işleme tekniklerini uygulayarak tespit edilmektedir.

1.1 Kemik Yaşı Nedir?

Bir çocuğun iki yaşı, kronolojik yaş ve kemik yaşı vardır. Kronolojik yaş, çocuğun doğum tarihinden itibaren belirlenen gerçek yaştır. Kemik yaşı ise çocuğun kemiklerinin olgunlaşma derecesini tanımlar. İnsan iskeleti gelişimindeki değişiklikler temel olarak benzerdir, çünkü her kemiğin gelişim süreci aynı aşamalardan geçer. Her aşamada, kemiklerin kendine has özellikleri vardır. Bu nedenle, kronolojik yaş ile kıyaslandığında kemik yaşı, bireysel büyüme gelişme düzeyini ve olgunlaşma derecesini yansıtmanın daha doğru bir yoludur [1].

1.2 Kemik Yaşı Değerlendirme Yöntemleri

İskelet olgunluğu değerlendirmesi veya kemik yaşı değerlendirmesi (BAA), sol el bileği kemikleşme gelişimini incelemek ve yüzlerce standart görüntüden oluşan bir atlas ile karşılaştırmalar yaparak kemiğin yaşını tahmin etmek için radyolojik bir işlemdir. Çocuklarda büyüme bozuklukları, kromozomal bozukluklar, endokrin bozukluklar ve endokrinolojik problemler gibi birçok hastalık kemik yaşı ile kronolojik yaş arasındaki tutarsızlık ile keşfedilebilir.

Kemik yaşı değerlendirmesi klinik rutinde önemli bir süreçtir. Kemik yaşı, bir bireyin iskelet ve biyolojik olgunluğunun bir göstergesidir. Bu, bir bireyin doğum tarihi kullanılarak hesaplanan kronolojik yaştan farklıdır. Çocuklarda ve endokrinologlar tarafından çocuklarda uzun veya kısa boylanmaya neden olan hastalıkların teşhisinde kronolojik yaş ile karşılaştırılması için kemik yaşı istenmektedir. Sağlıklı çocuklarda da kronolojik yaşı ile kemik yaşı arasında 1-2 yıla varan farklar mevcut olabilir [2].

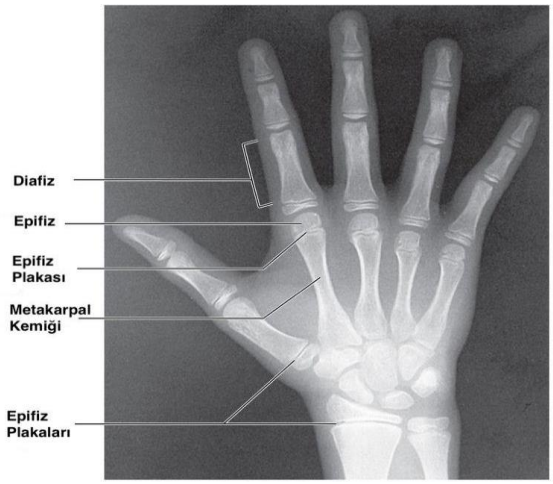
İskelet olgunluğu değerlendirmesi veya kemik yaşı değerlendirmesi (BAA) için uygulanan iki iyi bilinen yöntem vardır: Greulich-Pyle (GP) ve Tanner-Whitehouse (TW2) yöntemleri. TW2 sistemi, 20 berili kemiği inceleyen bir puanlama sistemine dayanırken, GP, sisteminde, el kemiği röntgenini atlastan standartlaştırılmış röntgenler ile karşılaştırıp değerlendirme yapılmaktadır. El radyografisi kullanılarak yapılan kemik yaşı değerlendirmesi, pediatri alanında, özellikle endokrinolojik problemler ve büyüme bozuklukları ile ilgili olarak, önemli bir klinik araçtır [3].

Çocukların kemik yaşı, cinsiyet, ırk, beslenme durumu, yaşam alanları vb. Tarafından etkilenmektedir. Sol el bileği iskelet gelişiminin radyolojik incelenmesine dayanarak, kemik yaşı değerlendirilir ve kronolojik yaşla karşılaştırılır. Bu iki değer arasındaki tutarsızlık iskelet gelişimindeki anormallikleri gösterir. İskelet kemiği yaşı tahmini için temel klinik yöntemler Greulich & Pyle (GP) yöntemi ve Tanner & Whitehouse (TW) yöntemidir.GP yönteminin kullanımı TW yönteminden daha hızlı ve kolaydır [4].

GP yönteminde sol el bileği grafisi, atlasta yaş ve cinsiyete göre gruplandırılmış bir dizi radyografi ile karşılaştırılır. Yüzeysel olarak klinik görüntüye benzeyen atlas örneği seçilir. TW metodu, her bir kemik için detaylı bir analiz kullanır ve onun gelişim aşamasını yansıtan sekiz sınıftan birine tahsis eder. Bu, her bir kemiğin

puanlar açısından tanımlanmasına yol açar. Tüm puanların toplamı kemik yaşını değerlendirir. Bu yöntem en güvenilir sonuçları verir. TW2, özellikle her aşamaya ilişkin puanlar ve her iki cinsiyet arasındaki fark ile ilgili olarak TW1'in bir revizyonudur. Ayrıntılı olarak, TW2 yönteminde, ana kemiklerde bulunan yirmi ilgili bölge (YG), kemik yaşı değerlendirmesi için göz önünde bulundurulur. Her YG üç bölüme ayrılır: Epifiz, Metafiz ve Diyafiz. Her YG'nin gelişimi ayrı aşamalara ayrılır [3].

Baskın olmayan el'in kemikleşme aşamalarının ayırt edici doğası nedeniyle genellikle sol elin radyolojik incelenmesi ile gerçekleştirilir ve daha sonra kronolojik yaşla karşılaştırılır. iki değer arasındaki tutarsızlık anormallikleri gösterir. Doğumdan sonra, epifizler yavaş yavaş tahmin edilebilir bir düzende kemikleşir ve iskelet olgunluğunda kemiğin ana gövdesi ile birleşir. Birincil merkezden kemikleşmiş kemik diyafizdir, ikincil merkezden kemikleşmiş kemik ise epifizdir. İkincil merkez aşamalı olarak kemikleşirken, kıkırdak, sadece ince bir kıkırdak tabakası olan epifiz plakası, diyafiz kemiğini epifizden ayırana kadar kemik ile değiştirilir. Diyafizin epifizde dayanan kısmı metafiz olarak kabul edilir ve kemiğin büyüyen ucunu temsil eder. Epifiz kıkırdak plakası devam ettiği sürece, hem diyafiz hem de epifiz büyümeye devam eder, fakat sonunda diyafiz ve epifizin kemikli yapıları kaynaşır ve büyüme durur [5]. Şekil 1.1'de el kemikleri anatomisi gösterilmektedir.



Şekil 1.1: El Kemikleri Anatomisi

2. GÖRÜNTÜ İŞLEME

Görüntü işleme, artık günümüzde pek çok alanda kullanılmaktadır. Örneğin görüntü geliştirmede, görüntü bozulmalarını gidermek ve önemli görüntü bilgisini çıkarmak için çeşitli yöntemler mevcuttur. Bilgisayar grafiklerinde, çok çeşitli görsel efektler ile dijital görüntüler üretilebilir, değiştirilebilir ve birleştirilebilir. Veri sıkıştırmada, sıkı bir dijital koda çevrildiğinde görüntüler verimli bir şekilde saklanabilir ve iletilebilir. Görüntüler üzerinde görüntü işleme yöntemleri uygulayarak görüntüleri iyileştirebilir ve görüntülerden bilgi alınabilir.

Dijital röntgen, görüntüleri yakalamak, görüntüyü taramak veya görüntüyü jpeg, png gibi dijital formatta saklamak için kullanılır, bu nedenle hastaların hastalığının teşhisinde faydalıdır. Ancak dijital röntgen görüntüleri gauss gürültüsü veya tuz ve biber gürültüsü gibi görültüleri içerir, böylece bazen röntgen görüntüsünü net bir şekilde vermez. Bu nedenle görüntülerin kalitesini iyileştirmek için görüntüdeki gürültüyü kaldırmak amacıyla görüntü işleme yöntemleri önemlidir [6].

2.1 Gürültü Giderme

Gürültü, görüntünün kalitesini etkileyen istenmeyen pikseller olarak tanımlanabilir. Gürültü şöyle yazılabilir:

$$f(x, y) = g(x, y) + \eta(x, y)$$

f (x, y) orijinal görüntü, g (x, y) ise çıkış görüntüdür ve η (x, y) gürültü modelidir. Farklı gürültü türleri vardır. Tuz ve karabiber gürültüsü, x-ışını görüntülerinde bulunabilecek en yaygın gürültü türlerinden biridir [7]. Bu tür bir gürültü genellikle görüntüde açık ve siyah noktalar olarak görünen yakalama veya aktarmadaki başarısızlıktan kaynaklanır. Tuz ve karabiber gürültüsü, x-ışını görüntüsüne matematiksel bir dönüşüm T uygulayarak aşağıdaki gibi ele alınmaktadır:

$$g(x, y) = T[f(x, y)] , F(x, y)$$

g (x, y), tuz ve biber gürültüsüne sahip orijinal x-ışını görüntüsüdür ve üzerine T uygulandıktan sonra çıkan görüntüdür. Çalışmamızda, görüntü kenarlarını ve keskinliğini koruyarak tuz ve biber gürültüsünü azaltmak için medyan filtre kullanılmıştır. Medyan filtre ayrıca kenarları ve görüntünün keskinliğini korurken görüntüden gelen gürültüyü azaltmak için de kullanılır. medyan filtre görüntü deki her pikseli alır ve komşu piksellerden ne kadar farklı olduğunu kontrol etmektedir, "Çok farklı" ise, değeri çevresindeki piksellerin orta değeriyle değiştirilir [8].

Medyan filtre, görüntüdeki her pikseli sırayla filtreler ve yakındaki komşuları, çevresini temsil edip etmediğine karar vermek için kullanır. Medyan filtresi, piksel değerini bu değerlerin ortalaması ile değiştirir. Yani, çevreleyen komşu piksellerden gelen değerler önce sayısal sıraya göre sıralanır ve ardından söz konusu pikselin değeri, orta (medyan) piksel değeri ile değiştirilir. Komşu pikseller pencere olarak adlandırılır [9].

Şekil 2.1.b, bir x-ışını el görüntüsünde gürültü giderme ve görüntü düzgünleştirme uygulamasının bir örneğini göstermektedir. Medyan filtreleme, görüntüdeki gürültüyü gidermek için kullanılan doğrusal olmayan bir yöntemdir. Kenarları korurken gürültüyü gidermede çok etkili olduğu için yaygın olarak kullanılır.

2.2 Görüntü Kenarı Tespiti

Kenar algılama işleminde pikseller düşürülür ve görüntüler korulur. Kenar tespiti, görüntü parlaklığının keskin bir şekilde değiştiği veya daha fazla bulanıklaştığı noktaları belirleme yöntemidir. Kenar algılama, görüntünün sınırını işaretleyen ve dijital görüntüdeki diğer yerlerden veya nesnelerden ayrılan çizgilerin keşfi olarak tanımlanabilir. Kenar algılama, görüntü noktalarında yoğunluk değişimlerinin kenar olarak bildirildiği bir yaklaşım kullanır. Yoğunlukta net ve tanımlanmış değişikliklerin gerçekleştiği bir resimdeki noktaları tanımlamak için kullanılan bir dizi eylemdir.

Bu eylem dizisi, örneğin görüntüyle ilgili bilgileri çıkarmak için gereklidir. Kenar tespiti için yöntem iki kategoride sınıflandırılır; Birincisi degrade tabanlı, ikincisi Laplacian merkezlidir.

Degrade tabanlı yöntemde, görüntünün birinci dereceden türevi alınarak kenarlar tespit edilir. Degrade büyüklüğü, bir kenar kuvveti ölçüsü hesaplamak için kullanılır [10].

Laplacian esaslı yöntemde, görüntü, sıfır geçişi olan ikinci mertebeden türev ifadesini hesaplamak için kullanılır. Genellikle, doğrusal olmayan bir farklılığı sıfır geçişi aranarak kenarları bulunur. Tipik olarak kenar tespiti için ön işleme aşaması, genellikle bir düzeltme aşaması olan Gauss yumuşatma uygulanır [10].

2.3 Özellik Çıkarma ve Seçme

Görüntüyü düzelttikten ve el kemiğinin kenarlarını tespit ettikten sonra, önerilen sistem el kemiği görüntüsünün kullanışlı ve ayırt edici özelliklerini çıkarmaktır. Özellik çıkarma, çeşitli görüntü işleme uygulamalarında ana adımdır. Dalgacık dönüşümünün özellikleri, Curvelet dönüşümünün özellikleri ve diğer dokusal özelliklerin gibi farklı özellik setlerinin bir kombinasyonu kullanılır [8].

2.3.1 Dalgacık özellikleri

Wavelet dönüşümü (veya Dalgacılık), fizik, matematik ve mühendislikteki problemleri çözmek için geliştirilmiş ilginç bir tekniktir. Bu yöntem, özellik çıkarma, doku kategorizasyonu, sinyal analizi ve görüntü işleme konularında en modern uygulamaları sağlar. Çeşitli tıbbi görüntüler, dalgacıkları kullanarak bir görüntüyü analiz ederken, sonuç analiz edilen görüntü için bir dizi katsayıdır. Şekil 2.2, bir x-ışını görüntüsü üzerinde Wavelet dönüşümünün uygulanmasının bir örneğini göstermektedir [8].

2.3.2 Curvelets özellikleri

Curvelet dönüşümü matematiksel, sinyal işlemede ve biyolojik uygulamalarda kullanılan Wavelet dönüşümünden türetilmiş çok ölçekli bir yöntemdir. Curvelet dönüşümü ilginç bir yöntemdir, çünkü nesneleri düzgün eğri içinde temsil etmek için uyarlanabilir bir yolu olan matematiksel bir çerçeve sağlar. Burada, bir Xray görüntüsüne Dalgacık veya Curvelet dönüşümlerinin uygulanmasının, özellikler olarak kullanılabilecek onbinlerce katsayı üreteceği belirtilmelidir. Bu kadar çok sayıda özellik, yüksek boyutlu veri kümelerinin zayıf kullanımı nedeniyle birkaç

önemli sınıflandırıcının kullanılmasını zorlaştırmaktadır. Bu nedenle, veri setine bir özellik seçim tekniği uygulanmalıdır [8].

(a) Orijinal Görüntü (b) Gürültü Giderildikten Sonraki Görüntü (c) Kenar algılama Sonucu

Şekil 2.1: Görüntü ön işleme adımları

Şekil 2.2: Dalgacık Dönüşümünü Kullanarak Görüntü Analizi

2.4 Gri Ölçek İşleme

Önemli bir görüntü manipülasyon sınıfı, teşhis açısından önemli bilgileri daha iyi görüntülemek için gri tonlama değerlerini değiştirmeyi içerir. Bununla birlikte, dijital bir görüntü ile çok daha kolay bir işlemdir ve ayrıca yapay nesneleri bastırmak için rutin olarak kullanılır.

Gri ölçekli manipülasyonlara daha genel olarak nokta işlemleri denir, çünkü giriş görüntüsündeki her piksel değeri, çıkış görüntüsüne her çıkış pikselinin değerinin yalnızca karşılık gelen giriş pikselinin değerine bağlı olduğu şekilde işlenir. Belirli bir nokta işlemi, giriş gri seviyelerinin çıkış görüntü gri seviyelerine eşlenmesini belirleyen fonksiyonel bir ilişki tarafından tamamıyla belirtilir.

Tekrarlanan nokta işlemlerini yapmak için genellikle bir arama tablosu kullanılır. Böyle bir tablo, nokta işleminin işlevsel ilişkisini içeren sıralı bir girdi ve çıktı piksel değeri çiftinden oluşur. Böylece çıktı piksel değerini girdi piksel değerinden hesaplamak yerine çıktı değeri bellekten alınır. Kişi, büyük ve hatta orta boyutlu görüntülerle uğraşırken, bu önemli miktarda zaman kazandırabilir [11].

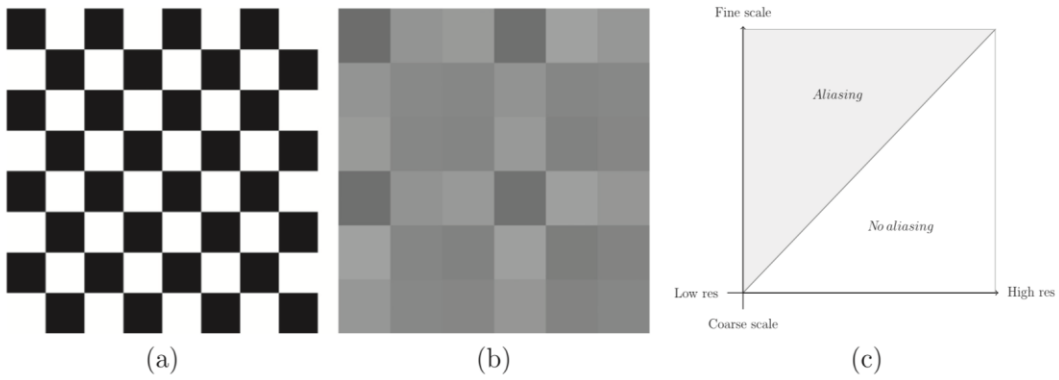
2.5 Görüntü Çözünürlüğü, Nesne Boyutu ve Görüntü Ölçeği:

Bir görüntü göz önüne alındığında çözünürlüğü piksel sayısı (yani görsel kaynaktan alınan örnek sayısı) cinsinden ifade edilebilir; düşük çözünürlüklü görüntüler, yüksek çözünürlüklü görüntülerden daha az piksele sahiptir. Bir görüntünün ölçeği, mekansal frekans içeriğini ifade eder. İnce ölçekli görüntüler, yüksek uzaysal frekanslardan (küçük görsel yapılarla ilişkili) ve düşük uzaysal frekanslara (büyük görsel yapılarla) kadar olan aralığı içerir. Kalın ölçekli görüntüler yalnızca düşük uzaysal frekansları içerir. Bir görüntünün uzaysal düzeltme (veya bulanıklaştırma) işlemi, düşük geçişli bir filtrenin çalışmasına karşılık gelir.

Yüksek uzaysal frekanslar kaldırılır ve düşük uzaysal frekanslar korunur. Böylece, ince ölçekli bir görüntünün uzaysal düzleştirilmesi daha kaba bir ölçek görüntüsü verir.

Bir görüntünün çözünürlüğü ile ölçeği arasındaki ilişki, bir görüntünün görsel ayrıntıları göstermek için takip edildiği teknik, görüntü ayrıntıların temsil edilmesini sağlamak için yeterince yüksek bir çözünürlüğe sahip olması gerekir. Şekil 2.3'de

gösterilen satranç tahtası modelini göz önünde bulundurursak, Şekil 2.3.b, satranç tahtası modelinin 6 * 6 piksel çoğaltılmasını göstermektedir. Reprodüksiyonun çözünürlüğü satranç tahtası deseni ince yapısını göstermek için yetersizdir. yetersiz çözünürlük (veya örnekleme) nedeniyle bir orijinal görüntünün bozulmasına 'Aliasing' denir [12].



Şekil 2.3: Aliasing İşlemin Gösterimi. (a) Bir satranç tahtasının görüntüsü. (b) Düşük çözünürlüklü (6*6 piksel) bir resimle satranç tahtasının kopyaları (c) Resim çözünürlüğü ve resim ölçeğiile kaplanan alan.

Görüntü çözünürlüğü görsel yapının temsil edilebileceği ölçeğe bir sınır getirmektedir. Şekil 2.3.c, çözünürlük (yatay eksen) ve ölçek (dikey eksen) ile kaplanan alanı gösterir. Sınır, gölgeli ve gölgesiz bölgelerin ayrılması ile temsil edilir. Gölgeli alanda bir ölçek ve çözünürlüğü birleştiren herhangi bir görüntü yumuşatmaya uğrar. En net görüntüler gölgeli-gölgesiz sınırında yer almaktadır.

Görüntü çözünürlüğü ile nesne boyutu arasındaki ilişki, çözünürlüğün görüntüde temsil edilebilecek nesnelerin boyutuna daha düşük bir sınır koymasıdır. Çözünürlük çok düşükse, daha küçük nesneler artık ayırt edilemez. Benzer şekilde, görüntü ölçeği ve nesne boyutu arasındaki ilişki, ölçek çok kaba olursa, daha küçük nesnelerin artık ayırt edilememesi dir. Görüntü yumuşatma, küçük nesnelerin görsel özellikleriyle ilişkili yüksek uzamsal frekansları kaldırır.

3. MAKİNE ÖĞRENMESİ VE KULLANILAN ALGORİTMALAR

Makine öğrenmesi, bilgisayar sistemlerinin doğrudan örneklerden, verilerden ve deneyimlerden öğrenmesini sağlayan bir yapay zeka dalıdır. Bilgisayarların belirli görevleri akıllıca yapmalarını sağlayarak makine öğrenme sistemleri, önceden programlanmış kuralları takip etmek yerine verilerden öğrenerek karmaşık işlemler gerçekleştirebilir [13].

Makine öğrenmesi, bilgisayarlara insanların ve hayvanların doğal olarak günlük hayatta yaptıklarını yapmalarını öğreten bir veri analizi tekniğidir. Yazılım uygulamalarının, açıkça programlanmadan sonuçları tahmin etmekte daha doğru olmalarını sağlayan yapay zekanın (AI) bir uygulamasıdır. Makine öğrenme algoritmaları bir model olarak önceden belirlenmiş denklemeye dayanmadan, bilgileri doğrudan veriden "öğrenmek" için hesaplama yöntemlerini kullanmaktadır.

Algoritma performansı, öğrenme için mevcut örnek sayısı arttıkça artmaktadır. Makinelerle ilgili olarak, çok geniş bir şekilde, bir makinenin yapısını, programını veya verilerini ne zaman değiştireceğini gelecekteki performansının artacağı şekilde öğrendiğini söylenebilir.

Artık birçok kişi, her gün makine öğrenmeye tabanlı sistemler ile etkileşime geçmektedir, örneğin sosyal medyada kullanılan görüntü tanıma sistemleri, sanal kişisel asistanlar tarafından kullanılan ses tanıma sistemleri, çevrimiçi perakendeciler tarafından kullanılan tavsiye sistemleri , Sağlık alanında doktorların belirli durumlar için daha doğru ve ya etkili teşhisler vermesine yardımcı olabilecek sistemler, bilim, biyoloji, fizik, tıp, sosyal bilimler ve daha fazla dallarda artık makine öğrenmesi yardımcı olmaktadır [13].

3.1 Denetimli Makine Öğrenmesi

Pratik makine öğrenmesinin çoğunluğu denetimli öğrenmeyi kullanır. Denetimli öğrenme, girdi değişkenlerinin (x) ve bir çıktı değişkeninin (Y) olduğu ve girdiden çıktıya eşleme işlevini öğrenmek için bir algoritma kullanıldığı öğrenmedir.

$$Y = f(X)$$

Buna denetimli öğrenme denir, çünkü eğitim veri setinden algoritma öğrenme işlemi, öğrenme sürecini denetleyen bir öğretmen olarak düşünülebilir. Doğru sonuçlar mevcut, algoritma tekrarlı olarak eğitim verileri üzerinde tahminler yapıp bir öğretmen tarafından düzeltilir. Algoritma kabul edilebilir bir performans seviyesine ulaştığında öğrenme durmaktadır [14].

Denetimli makine öğrenmesi algoritmalarından en çok yaygın örnekler:

- Regresyon problemleri için doğrusal regresyon.
- Sınıflandırma ve regresyon problemleri için rastgele ormanlar.
- Sınıflandırma problemleri için destek vektör makineleri.

3.2 Denetimsiz Makine Öğrenmesi

Denetimsiz öğrenme, yalnızca girdi verilerin (X) olduğu ve karşılık gelen çıktı değişkenlerinin olmadığı öğrenmedir. Denetimsiz öğrenmenin amacı, veriler hakkında daha fazla bilgi edinmek için verilerdeki temel yapıyı veya dağılımı modellemektir. Bunlara denetlenmeyen öğrenme denir, çünkü yukarıda denetlenen öğrenmeden farklı olarak bilinen sonuçlar yoktur. Algoritmalar, verilerdeki yapıyı keşfetmek ve sunmak için kendi aygıtlarına bırakılmıştır [14].

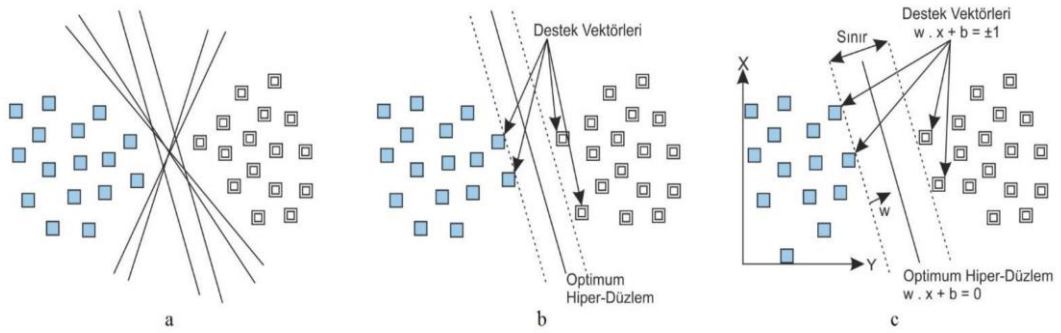
3.3 Destek Vektör Makinesi (DVM)

Destek Vektör Makinesi hem sınıflandırma hem de regresyon zorlukları için kullanılabilecek denetimli bir makine öğrenme algoritmasıdır. DVM çoğunlukla sınıflandırma problemlerinde kullanılır. Bu algoritmada, her bir veri öğesini n-boyutlu uzayda bir nokta olarak çizilir, her özelliğin değeri belirli bir koordinatın değerini alır. Ve iki sınıfı çok iyi ayıran optimum hiper düzlemi bularak sınıflandırma yapılır [15].

Destek vektör makinesi, veri analizi ve Görüntü tanıma için yaygın olarak kullanılan bir makine öğrenme yöntemidir. Destek vektör makinesinin ana fikri, hangi sınıfa ait olduğunu göstermek için veri kümeleri arasında bir hiper düzlem oluşturmaktır. Zorluk, makineyi verilerden yapıyı anlama ve doğru sınıf etiketi ile haritalama konusunda eğitmektir. En iyi sonuç, hiper düzlem herhangi bir sınıfın en yakın eğitim veri noktalarına en büyük mesafeye sahip olduğunda oluşur.

Destek vektörleri, hiper düzlemine daha yakın olan ve hiper düzlemin konumunu ve yönünü etkileyen veri noktalarıdır. Destek vektörlerini kullanarak, sınıflandırıcının marjını maksimize edilir. Destek vektörlerini silmek, hiper düzlemin konumunu değiştirir. Bunlar, DVM'i oluşturmaya yardımcı olan hususlardır Hiper düzlemler, veri noktalarını sınıflandırmaya yardımcı olan karar sınırlarıdır. Hiper düzlem her iki tarafına düşen veri noktaları farklı sınıflara bağlanabilir. Ayrıca, hiper düzlemin boyutu, özelliklerin sayısına bağlıdır. Giriş özelliklerinin sayısı 2 ise, hiper düzlem sadece bir çizgidir. Giriş özelliklerinin sayısı 3 ise, hiper düzlem iki boyutlu bir düzlem haline gelir.

Şekil 3.1.a'da gösterildiği gibi iki sınıf arasında sonsuz adet düzlem çizilebilmektedir. Oluşturulan bu düzlemlerden iki sınıfın arasındaki en büyük ayrımı gösteren düzlem istenilen uygun değerde düzlemdir, sınıf aralığını belirten vektörler ise destek vektörleri olarak bilinir ve Şekil 3.2.b'de gösterilmektedir. Sınıflar arası optimum hiper düzlem Şekil 3.3.c'de gösterildiği gibi belirlenmektedir.



Şekil 3.1: (a) İki Sınıf Arasındaki Sonsuz Düzlem (b) Destek Vektörler (c) Optimum Hiper Düzlem

3.3.1 Destek vektör makinesi ile öğreneme

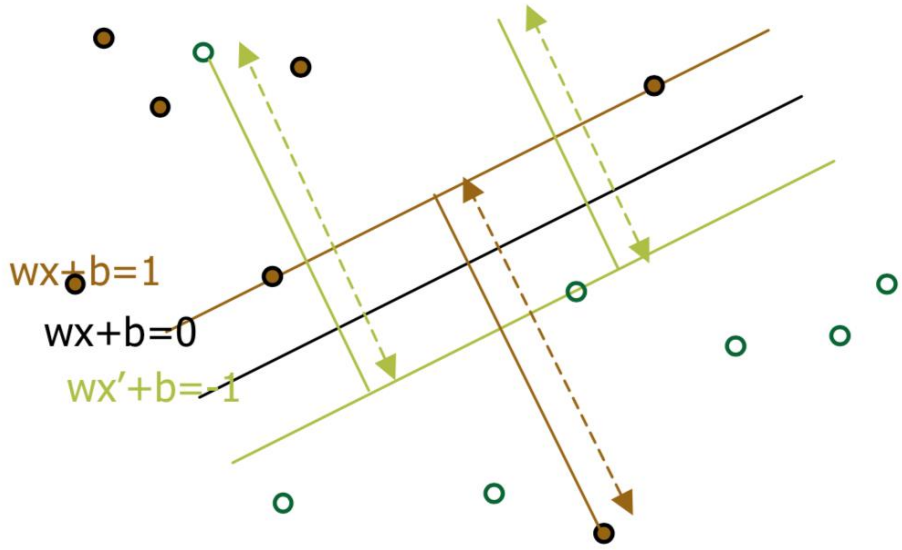
DVM'in temel güçlü yönleri, eğitimi nispeten kolaydır. Yüksek boyutlu verilere nispeten iyi ayırır ve sınıflandırıcı karmaşıklığı ile hata arasındaki denge açıkça kontrol edilebilir. DVM matematiksel ve mühendislik problemleri, el yazısı tanıma, nesne tanıma konuşmacı tanıma, görüntülerde yüz tanıma gibi birçok makine öğrenme algoritmalarında kullanmaktadır. SVM'nin amacı verileri hiper düzlemle ayırmaktır. SVM'yi hesaplamak için hedefin tüm verileri doğru şekilde sınıflandırmaktır. Matematiksel hesaplamalar için:

[a] Yi= +1 => $wxi + b \geq 1$

[b] Yi= -1 => $wxi + b \leq 1$

[c]Tüm i için; $yi\ (wi + b) \geq 1$

Bu denklemde x bir vektör noktasıdır ve w ağırlıktır ve aynı zamanda bir vektördür [16]. Bu yüzden verileri ayırmak için [a] daima sıfırdan büyük olmalıdır. Tüm olası hiper düzlemler arasından SVM, hiper düzlemin mesafesinin mümkün olduğu kadar büyük olduğu yeri seçer. Marjı maksimize eden bu optimum hiper düzlem aynı zamanda iki veri setinin en yakın noktalar arasındaki çizgileri ikiye böler. Böylece [a], [b] ve [c] Şekil 3.2'da görüldüğü gibi çizilmiştir.



Şekil 3.2: Hiper Düzlemler

Hiper düzlemde orijin noktasına en yakın noktanın mesafesi x'in hiper düzlem üzerinde olduğu gibi maksimize edilmesi ile bulunabilir. Benzer şekilde diğer yan noktaları için de aynı senaryo uygulanır. Böylece iki mesafeyi çözerek ve çıkartarak, optimum ayırma hiper düzleminden en yakın noktalara kadar olan toplam mesafeyi elde edilir.

Maksimum Marj = M= 2 / ||w|

Şimdi marjı maksimize etmek minimum ile aynıdır. İkinci dereceden bir optimizasyon problemi mevcuttur, w ve b'yi için çözmemiz gerekmektedir. Bunu çözmek için ikinci dereceden fonksiyonu doğrusal kısıtlarla optimize etmemiz gerekir. Çözüm, Langlier'ın çarpanı αi'nin ilişkilendirildiği yerden ikili bir problem oluşturmak. W ve b'yi

Φ (w) =1⁄2 |w'||w| minimize edecek şekilde bulmamız gerekmektedir.

Tümü için {(xi, yi)}: *yi* (w * xi + *b*) ≥ 1

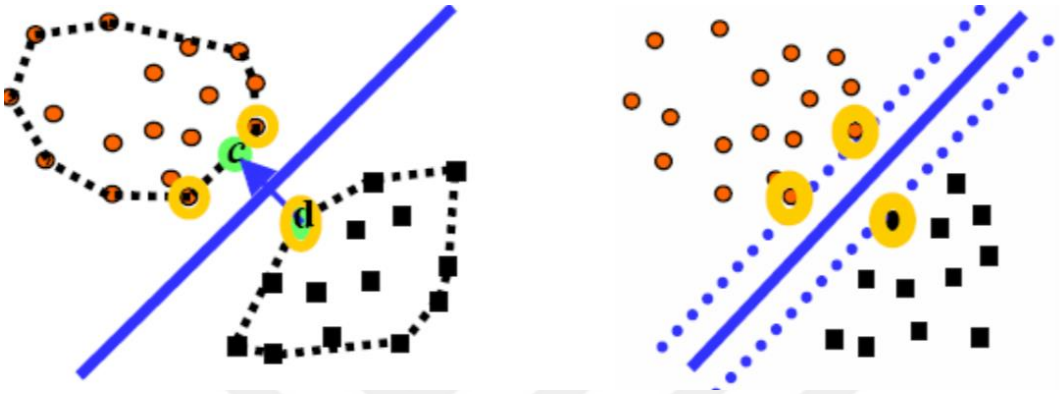
w =Σ*αi* * xi; *b*= *yk*- w *xk ; αk ≠ 0 olacak şekilde herhangi bir xk için

Sonuç olarak DVM sınıflandırma fonksiyonu aşağıdaki forma sahip olacaktır [16]:

$$f(x) = \Sigma \alpha i \; yi \; xi * x + b$$

3.3.2 Destek vektörleri

Hiper düzlemi seçerken tüm eğitim noktaları önemli değildir. Şekil 3.3'de görüldüğü gibi, karar alma sınırını seçmek için arka noktaların önemli olmadığı açıktır. Resimlerde ilgili noktalar sarı renkle işaretlenmiştir. Bu noktalara Destek Vectros adı verilmektedir.



Şekil 3.3: Destek Vektörleri

Tüm eğitim noktaları, bunlarla ilişkili katsayılara sahiptir. Katsayılar, verilen test puanları için nihai karar fonksiyonuna bu noktaların dahil edilme gücünü ifade eder. Ayırıcı hiper düzlemine en yakın olan tüm destek vektörleri için katsayılar 0'dan büyüktür. Noktaların geri kalanı için karşılık gelen katsayılar sıfıra eşittir.

Aşağıdaki denklem, eğitim noktaları ve karar sınırı arasındaki bağımlılığı açıklar:

$$w = \sum_i \alpha_i y_i \phi(\mathbf{X}_i)$$

Burada αi pozitif gerçek sayılardır - katsayılar. Katsayıların aşağıdaki koşulları sağlaması gerekir:

$$\sum_i \alpha_i - \sum_{ij} \alpha_i \alpha_j y_i y_j \langle \phi(\mathbf{X}_i), \phi(\mathbf{X}_j) \rangle \qquad \sum \alpha_i y_i = 0, \alpha_i > 0.$$

İlk denklemi maksimuma çıkarmak için katsayılar seçilmelidir [17].

3.4 En Yakin Komşu (KNN)

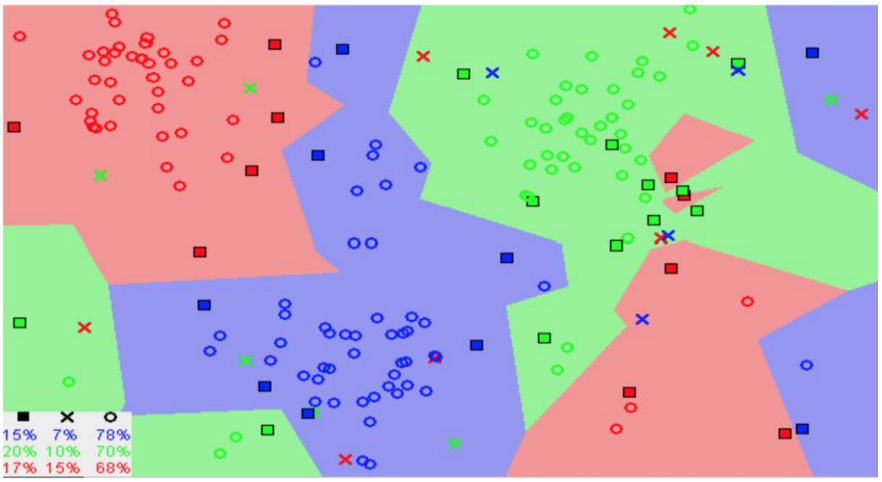
K En Yakın Komşu (KNN) yöntemi, basit uygulama ve seçkin performansı nedeniyle veri madenciliği ve makine öğrenmesi uygulamalarında yaygın olarak kullanılmaktadır. Sınıflandırma, veri madenciliğindeki en önemli araştırma konularından biridir. Sınıflandırmanın ana görevi, tüm eğitim veri noktalarını içererek test veri noktalarının etiketlerini tahmin etmektir.

Son birkaç on yılda, gerçek uygulamalarda birçok sınıflandırma yöntemi geliştirilmiştir. Arasında k En Yakın Komşular (KNN) sınıflandırması ilk 10 veri madenciliği algoritmasından biri olarak, sadeliği ve verimliliği nedeniyle, kabul edilmiştir. Standart bir KNN yönteminin ana fikri, bir test verisi noktasının etiketini çoğunluk kuralı ile tahmin etmektir, yani, test verisi noktasının etiketi, özellikteki en benzer eğitim verisi noktalarının ana sınıfı ile tahmin edilir.

Eğitim veri setindeki K örneklerinden hangisinin yeni bir girişe en çok benzeyeceğini belirlemek için bir uzaklık ölçüsü kullanılır. Gerçek değerli giriş değişkenleri için en yaygın uzaklık ölçüsü Öklid mesafesidir [18].

$$d(i,j)=\sqrt{\sum_{k=1}^{p}(X_{ik}-X_{jk})^2}$$

KNN algoritması, benzer şeylerin yakınlarda bulunduğunu varsayar. Başka bir deyişle, benzer şeyler birbirine yakındır. Şekil 3.4'de KKN ile Benzer veri noktalarının birbirine nasıl yakın olduğunu gösteren resim gösterilmektedir.

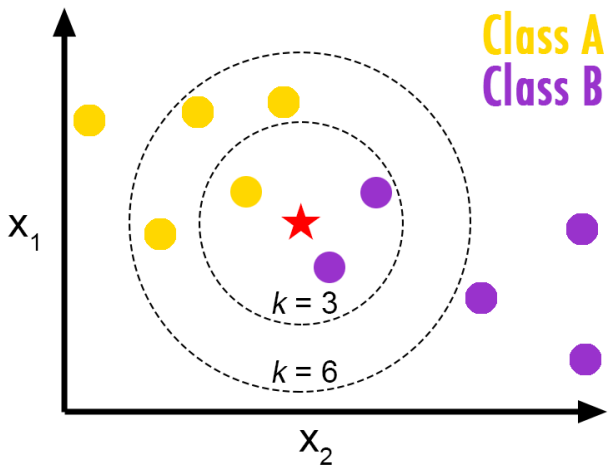


Şekil 3.4: Benzer Veri Noktalarının Birbirine Nasıl Yakın Olduğunu Gösteren Resim

KNN algoritması en doğru modellerle rekabet edebilir çünkü yüksek doğrulukta tahminler yapar. Bu nedenle, KNN algoritmasını yüksek doğruluk gerektiren ancak insan tarafından okunabilen bir model gerektirmeyen uygulamalar için kullanılabilir. Tahminlerin kalitesi mesafe ölçere bağlıdır. Bu nedenle, KNN algoritması, yeterli alan bilgisinin mevcut olduğu uygulamalar için uygundur. Finans, sağlık, siyaset bilimi, el yazısı tanıma, görüntü tanıma ve video tanıma gibi çeşitli uygulamalarda KNN kullanılabilir [19].

3.4.1 KNN ile öğrenme

Bir nesne, komşularının çoğunluk oyuyla sınıflandırılır, nesne, en yakın komşuları arasında en yaygın sınıfa atanır (k, tipik olarak küçük olan bir tamsayıdır). Eğer k = 1 ise, nesne basitçe en yakın komşunun sınıfına atanır.



Şekil 3.5: KNN Komşu Sayısı Seçimi

Çalışma süresini azaltmak için, ilgi çekici bir ağaç inşa etme hattına alternatif yaklaşımlar icat edildi. Bu yöntemler, test için toplam mesafe bilgisini verimli bir şekilde kodlayarak gerekli mesafe hesaplama sayısını azaltmaya çalışır. Şekil 3.5'de gösterildiği gibi, temel fikir, eğer A noktası B noktasından çok uzaksa ve B noktası C noktasına çok yakınsa, A ve C noktalarının mesafelerini açıkça hesaplamak zorunda kalmadan çok uzak olduğu anlaşılır.

Bu şekilde, en yakın komşular aramasının hesaplama maliyeti D boyutlarındaki N örnekleri için O[D N * log (N)] veya altına düşürülebilir [18].

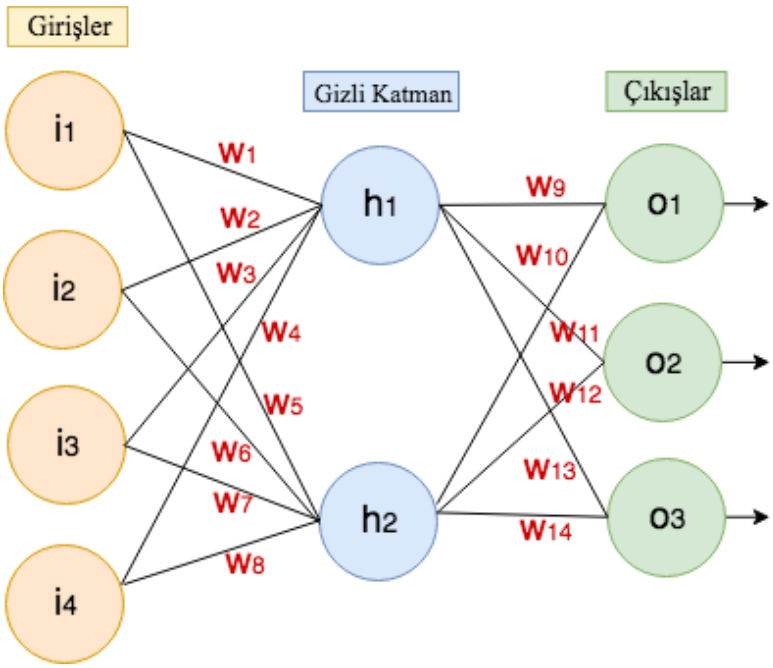
KNN'nin sınıflandırma ana avantajları:

- Çok basit bir uygulama.
- Arama alanı açısından sağlam; örneğin, sınıfların doğrusal olarak ayrılabilmesi gerekmez.
- Sınıflandırıcı, bilinen sınıflara sahip yeni örnekler sunulduğundan, çok az bir maliyetle güncellenebilir.
- Ayarlanacak az sayıda parametre mevcuttur, mesafe ölçümü ve k.

3.5 Konvolüsyonal Sinir Ağları (CNN)

Yapay Sinir Ağları (YSA), biyolojik sinir sistemlerinin (insan beyni gibi) çalışmasıyla yoğun şekilde ilham alan işlemsel sistemleridir. YSA'lar, esas olarak nihai çıktıyı optimize etmek amacıyla girdilerden topluca öğrenmek üzere işlerin dağıtık bir şekilde yerleştirildiği çok sayıda birbirine bağlı hesaplama düğümünden (nöronlar) oluşur.

Bir YSA'nın temel yapısı, Şekil 3.6'da gösterildiği gibi modellenebilir. Girdiyi, genellikle çok boyutlu bir vektör biçiminde girdi katmanını, gizli katmanlara dağıtacak şekilde yüklenir. Gizli katmanlar daha sonra önceki katmandan kararlar alır ve kendi içindeki stokastik bir değişimin nihai çıktının nasıl zarar verdiğini veya iyileştirdiğini tartıştır, buna öğrenme süreci olarak adlandırılır.



Şekil 3.6: YSA Temel Yapısı

Konvolüsyonel Sinir Ağları (CNN'ler), geleneksel YSA'lara benzer, çünkü öğrenme yoluyla kendini optimize eden nöronlardan oluşurlar. Her bir nöron yine bir girdi alacak ve sayısız YSA'nın temeli olan bir işlem gerçekleştirecektir. Son katman, sınıflarla ilişkili kayıp fonksiyonlarını içerir. CNN'ler ile YSA'lar arasındaki tek önemli fark, CNN'lerin öncelikle görüntülerdeki örüntü tanıma alanında kullanılmasıdır. Bu, görüntüye özgü özellikleri mimariye kodlamamıza izin vererek ağın görüntü odaklı görevler için daha uygun olmasını sağlarken, modeli ayarlamak için gereken parametreleri daha da azaltır. Geleneksel YSA formlarının en büyük sınırlamalarından biri, görüntü verilerini hesaplamak için gereken hesaplama karmaşıklığı ile mücadele etme eğiliminde olmalarıdır [20].

Konvolüsyonel Sinir Ağı, son on yılda, örüntü tanıma ile ilgili çeşitli alanlarda çığır açan sonuçlar vermiştir. CNN'lerin en faydalı yönü, YSA'da parametre sayısını azaltmaktır. Bu başarı hem araştırmacıları hem de geliştiricileri, klasik YSA'larla mümkün olmayan karmaşık görevleri çözmek için daha büyük modellere yaklaşmaya yöneltti. CNN tarafından çözülen problemlerle ilgili en önemli varsayım, mekansal olarak bağımlı olan özelliklere sahip olmamalıdır. Başka bir deyişle, örneğin bir yüz tanıma uygulamasında, yüzlerin resimlerde bulunduğu yere dikkat etmemize gerek yoktur. Tek endişe, verilen resimlerdeki konumlarından bağımsız olarak onları tespit etmektir. CNN'nin bir başka önemli yönü, giriş daha derin katmanlara doğru yayıldığında soyut özellikler elde etmektir [21].

CNN'ler temel olarak girdilerin görüntülerle karşılaştırılması temelinde odaklanır. Belirli veri türleriyle uğraşma ihtiyacına en iyi şekilde uyacak mimariyi bulmakla odaklanır. YSA ile en önemli farklardan biri, CNN içindeki katmanların nöronların üç boyutta düzenlenen nöronlardan oluşmasıdır; girdilerin uazaysal boyutları yükseklik, genişlik ve derinliktir. Derinlik, YSA içindeki toplam katman sayısına değil, bir aktivasyon hacminin üçüncü boyutuna karşılık gelmektedir. Herhangi bir katmanın içindeki nöronlar, sadece ondan önceki katmanın küçük bir bölgesine bağlanmaktadır [20].

3.5.1 CNN ile öğrenme

Bir sinir ağı, aralarında veri alışverişi yapan, birbirine bağlı yapay “nöronlar”dan oluşan bir sistemidir. Bağlantılar, eğitim sürecinde ayarlanan sayısal ağırlıklara sahiptir, böylece düzgün bir şekilde eğitilmiş bir ağ tanımak için bir görüntü veya

desen sunulduğunda doğru sonuç verecektir. Ağ, birden fazla özellik algılayan "nöron" katmanından oluşur. Her katman önceki katmanlardan farklı girdi kombinasyonlarına cevap veren birçok nörona sahiptir.

Tipik CNN'ler 5 ila 25 farklı desen tanıma katmanı kullanır. Eğitim, hedeflenen çıktı ile etiketlenen geniş bir girdi modelleri çeşitliliği içinde girdilerin "etiketli" veri seti kullanılarak gerçekleştirilir. Eğitim, ara ve son özellik nöronlarının ağırlıklarını belirlemek için genel amaçlı yöntemler kullanır. Bir CNN, standart bir sinir ağında olduğu gibi bir veya daha fazla tam bağlı katman tarafından takip edilen, genellikle bir alt-örnekleme katmanına sahip bir veya daha fazla katmandan oluşur. Geleneksel örüntü / görüntü tanıma modelinde, elle tasarlanmış bir özellik çıkarıcı girdiden ilgili bilgileri toplar ve önemsiz değişkenlikleri ortadan kaldırılmaktadır.

CNN'ler, görüntü ve örüntü tanıma, konuşma tanıma, doğal dil işleme ve video analizi de dahil olmak üzere çeşitli alanlarda kullnılmaktadır. Konvolüsyonel sinir ağlarının önem kazanmasının birkaç nedeni vardır. Örüntü tanıma için geleneksel modellerde, özellik çıkarıcılar elle tasarlanmıştır. CNN'lerde, özellik çıkarımı için kullanılan evrişimli katmanın ağırlıkları ve sınıflandırma için kullanılan tam bağlı katman eğitim süreci boyunca belirlenir. CNN'lerin gelişmiş ağ yapıları, bellek gereksinimlerinde ve hesaplama karmaşıklığı gerekliliklerinde tasarruf sağlar ve aynı zamanda, girdinin yerel korelasyona sahip olduğu uygulamalar için (örneğin görüntü ve konuşma) daha iyi performans sağlamaktadır [22].

Bir CNN'de sınıflandırma problemlerini çözmek için çoklu ve farklı katmanlardan oluşarak, karmaşık mimariler oluşturulur. Dört tür katman en yaygındır: Evrişim Katmanları, Havuzlama / Alt Örnekleme Katmanları, Doğrusal Olmayan Katmanlar ve Tam Bağlı Katmanlar.

3.5.2 Tam bağlı katmanlar

Tam bağlı katmanlar, bir CNN'nin son katları olarak kullanılır. Bu katmanlar matematiksel olarak önceki bir özellik katmanının ağırlığını toplayarak, belirli bir hedef çıktı sonucunu belirlemek için "bileşenlerin" tam karışımını gösterir. Tam bağlı bir katman olması durumunda, önceki katmanın tüm özelliklerinin tüm elemanları, her çıkış özelliğinin her bir elemanının hesaplanmasında kullanılır.

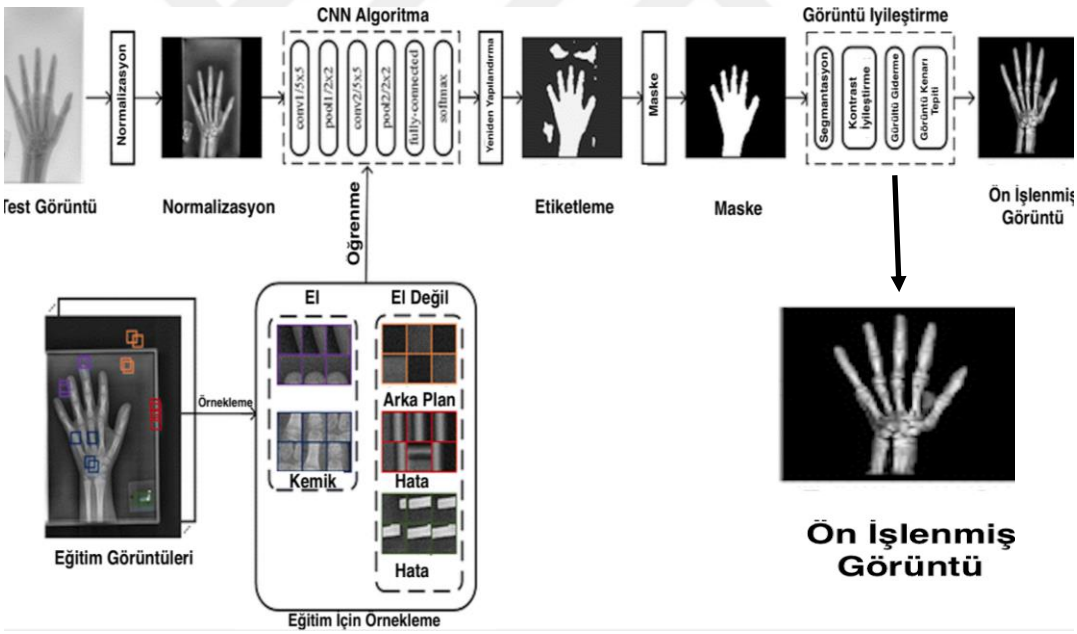
D boyutunda bir ağ da n giriş koordinatı varsa, m çıkışlı tam bağlı bir katman n · m parametresini gerektirir; tipik çalışma sistemlerde O (n ^ 2) parametrelerinin

karmaşıklığını gösterir. Genel olarak tamamen bağlanmış katmanlar yerine rastgele filtreler kullanmak, karmaşıklığı, her bir yerel harita için "yerel olarak bağlanmış" bir ağ oluşturarak metrik yapıyı kullanarak yaptığı gibi, özellik haritası başına O (n) parametrelerine düşürür. İkisinin birlikte kullanılması, O (k · S) parametrelerini verir; k, özellik haritalarının sayısıdır ve S filtrelerin desteğidir, sonuç olarak öğrenme karmaşıklığı tam bağlı katmanlarda N'den bağımsızdır [23].

4. UYGULAMA MODELİ

El X-Işınları üzerinde görüntü işleme teknikleri ile makine öğrenmesi algoritmaları uygulanarak girilen görüntülere ait kemik yaşı tahmin edilmiştir. Giriş görüntüleri üzerinde görüntü işleme teknikleri yukarıda açıklandığı gibi, gürültü giderme, görüntü kenar tespiti ve gri ölçek işlemleri işlenmiştir. Model eğitiminde CNN algoritması kullanılmıştır. Transfer öğrenme ile önceden eğitilmiş VGG16 ve Inception V3 algoritmaları özellik olarak eklenmiştir. İki öğrenmede de en iyi model seçilmiş ve sonuçlar arasındaki kemik yaşı belirleme doğruluğu kıyaslanmıştır.

CNN model eğitiminde uygulanan aşamalar Şekil 4.1'de gösterilmektedir.



Şekil 4.1: CNN Eğitim Süreci

4.1 Kullanılan Veri Seti

Kemik yaşı tespitinde kullanılan veri seti Stanford Üniversitesi, Colorado Üniversitesi ve California Üniversitesi'nin katkılarıyla oluşturulmuştur. Veri setleri üniversiteler tarafından web sayfalarında yayınlamış, oradan alınmıştır. Kullanılan

veri seti çocuklara ait el x-ışınlarından ibarettir. 0 - 18 yaş arası kız ve erkek el x ışınları kemik yaşı tespiti için kullanılmıştır. Tam 2500 x-ışınlarından oluşan görüntü veri setleri eğitilmiş, iyi sonuçlar göstermiştir.

Konvolüsyonel Sinir Ağları (CNN'ler) görüntü sınıflandırma görevleri için kullanılmıştır. CNN'leri eğitmek için çok fazla veri ve zaman gerekmektedir. Bununla birlikte, bazen veri kümesi sınırlı olabilir ve bir CNN'yi sıfırdan eğitmek için yeterli olmayabilir. Böyle bir senaryoda, büyük bir veri kümesi üzerinde önceden eğitilmiş bir CNN kullanmak yardımcı olmaktadır. Imagenet'te eğitilmiş 1 katmanlı bir ağ olan VGG-16 önceden eğitilmiş CNN'i kullanılmıştır. Daha sonra aynı veri setleri üzerinde Imagenet'te eğitilmiş 1 katmanlı bir ağ olan Inception V3 önceden eğitilmiş CNN'i kullanılmış, VGG16 sonuçları ile kıyaslanmıştır.

Önceden eğitilmiş bir model aşağıdakiler için kullanılır:

- Görüntü Sınıflandırması: Yeni veri setinin eğitim veri setiyle aynı sınıflara sahip olması durumunda, önceden eğitilmiş CNN, yeni veri setindeki görüntülerin sınıfını tahmin etmek için doğrudan kullanılabilir.
- Özellik Çıkarma: CNN'ler ayrıca sınıflandırıcı yerine özellik çıkarıcı olarak da kullanılabilir. CNN'nin son tabakası çıkarılabilir ve özellik vektörünü elde etmek için ağın geri kalanından bir görüntü geçirilebilir. Örneğin, VGG-16 modelinde, son katman (1000 boyutlu) çıkarılabilir ve tam bağlı katman (fc2), bir giriş görüntüsünün 4096 boyutlu bir özellik vektörü temsiliyle sonuçlanır. Tüm eğitim görüntülerinden özellikler çıkarıldıktan sonra, görüntü sınıflandırması için SVM veya lojistik regresyon gibi bir sınıflayıcı eğitilebilir.
- Transfer eğitimi için önceden eğitilmiş CNN'leri kullanmanın bir başka yolu, önceden eğitilmiş bir ağdan ağ ağırlıkları başlatarak ve ardından yeni veri seti ile ağı yeniden eğiterek CNN'lere ince ayar yapmaktır. Modellemede bu yolu kullanarak önceden eğitilmiş VGG16 ile Inception V3 ağırlıkları ile model başlatılmış CNN ile ağ yeniden eğitimiştir.

4.2 Normalizasyon

Ön İşleme tekniğinin ilk adımı, radyografileri CNN'e göndermeden önce gri tonlamalı bir taban ve görüntü boyutu için radyografileri normalleştirmektir. Bazı

görüntülerde beyaz arka plana sahip siyah, bazılarında ise siyah arka plana sahip beyaz kemikler vardır. Görüntü boyutu birkaç bin ile birkaç yüz piksel arasında değişebilir. Farklı gri tonlu tabanları normalleştirmek için, her görüntünün dört köşesindeki 10 x 10 görüntü yamasının piksel araçlarını hesaplanmış ve bunları belirli bir görüntü çözünürlüğü için maksimum değerin yarı değeriyle karşılaştırılmıştır (örneğin, 8 için 128 -bit çözünürlük). Bu, bir görüntünün beyaz veya siyah bir arka plana sahip olup olmadığını belirleyerek hepsini siyah arka plana göre normalleştirmemize olanak tanımıştır.

Bir sonraki adım, giriş görüntülerinin boyutlarını normalleştirme. Neredeyse tüm el radyografileri yükseklikte dikdörtgenler dir. Buna göre, tüm görüntülerin yüksekliğini 384 piksele çevirip, ardından en boy oranlarını koruyarak ve sıfır dolgusu kullanarak birleştirilmiştir. Genişliklerin tümü 384 piksel yapılarak sonuçta standartlaştırılmış 384 × 384 görüntüler elde edilmiştir. Bu boyut iki nedenden dolayı seçilmiştir; sinir ağı için gereken giriş boyutu (224 × 224) boyutundan daha büyük olması gerekmektedir, ve bu boyut, CNN saptama performansı ve ön işleme hızı için en uygun dengedir. Daha büyük kareler daha yavaş dağıtım süresi pahasına, CNN performansını artırırken, daha küçük kareler test süresini hızlandırır, ancak daha kötü görüntü ön işleme ile sonuçlanır.

4.3 Modelleme

Derin CNN'ler, girdi görüntülerden katmanlı hiyerarşiği öğrenmek için değişen evrişim ve havuzlama katmanlarından oluşur, ardından daha sonra önceki katmanlardan elde edilen özellik vektörleri ile eğitilebilir olan tam bağlantılı sınıflandırma katmanları takip edilir. Nesne sınıflandırma, algılama ve anlamsal bölümleme dahil olmak üzere birçok bilgisayar vizyon görevinde önemli başarılar elde edilmiştir. Birçok yenilikçi derin sinir ağları ve yeni eğitim yöntemleri, görüntü sınıflandırma görevleri için etkileyici performans göstermiştir. Yaygın bir VGG model ailesinden esinlenerek, regresyon çıktısı olan derin bir evrişimsel sinir ağı olarak uygulanmaktadır. VGG modülü, Üstel Doğrusal Birim (ELU) aktivasyon fonksiyonu, küme normalizasyonu ve maksimum havuzlamaya sahip iki evrişimli tabakadan oluşmaktadır.

Giriş görüntüsü, üç VGG bloğundan oluşan bir ağdan ve ardından üç Tamamen Bağlı katmandan geçirilir. VGG blokları sırasıyla 64, 128, 256 konvolüsyon

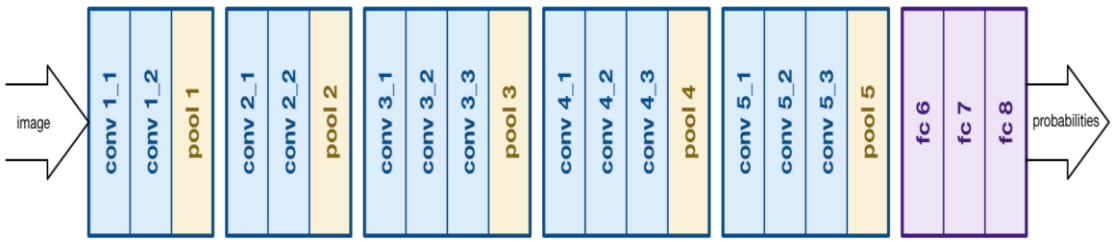
katmanlarından oluşur. Model, Adam en iyi duruma getiricisi kullanılarak Ortalama Kare Hata kaybı işlevi (MSE) ile eğitilmiştir:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y_i}|$$

4.3.1 VGG16 - sınıflandırma ve tespiti için konvolüsyon ağı

VGG16, Oxford Üniversitesi'nden K. Simonyan ve A. Zisserman tarafından "Büyük Ölçekli Görüntü Tanıma için Çok Derin Konvolüsyon Şebekeleri" başlıklı makalesinde tanıtılan konvolüsyonel bir sinir ağı modelidir. Model, 1000 sınıfa ait 14 milyondan fazla görüntünün veri seti olan ImageNet'te %92,7'lik ilk 5 test doğruluğunu elde etmiştir.

Büyük çekirdek boyutundaki filtreleri (sırasıyla birinci ve ikinci evrimli katmanda) birbiri ardına birden fazla 3 x 3 çekirdek boyutundaki filtreyle değiştirerek AlexNet'teki geliştirmeyi yapmaktadır [24] Şekil 4.2'de VGG16 mimarisi gösterilmektedir.



Şekil 4.2: VGG16 Mimarisi

4.3.2 VGG16 mimarisi

Conv1 katmanına giriş sabit büyüklükte 224 x 224 RGB görüntüdür. Görüntü, filtrelerin çok küçük bir alıcı alana sahip olduğu bir evrişimli (konv.) Katmanından geçirilir: 3 x 3 (sol / sağ, yukarı / aşağı, merkez kavramını yakalamak için en küçük boyuttur).

Yapılandırmalardan birinde, giriş kanallarının doğrusal olmayan bir dönüşümü olarak görülebilen 1 x 1 evrişim filtreleri de kullanmaktadır. katmanlar. Mekansal havuzlama, conv'un bir kısmını takip eden beş adet maksimum havuzlama katmanı tarafından gerçekleştirilir. Üç tam bağlantılı (FC) katman, bir evrişimli katmanı takip

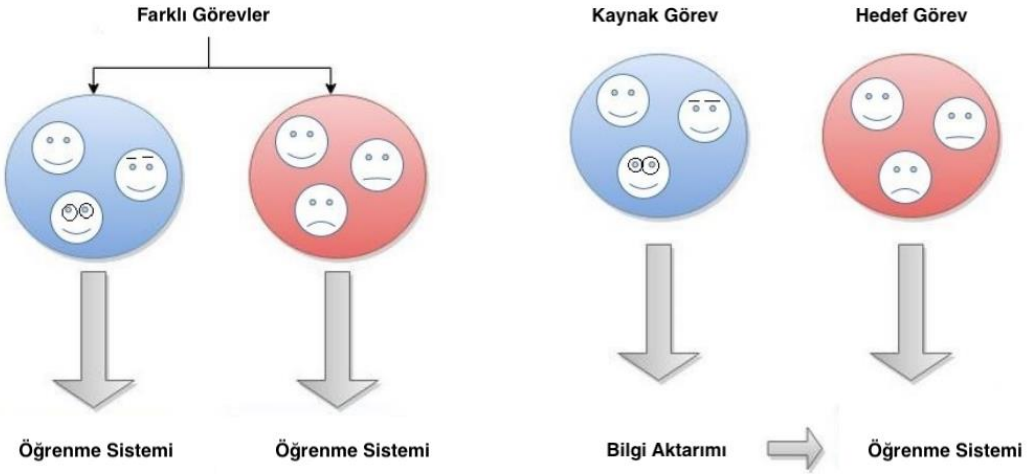
eder. ilk ikisi, her biri 4096 kanala sahiptir. Son katman yumuşak maksimum katmandır. Tam olarak bağlı katmanların yapılandırması tüm ağlarda aynıdır [24] .

4.3.3 Transfer öğrenme

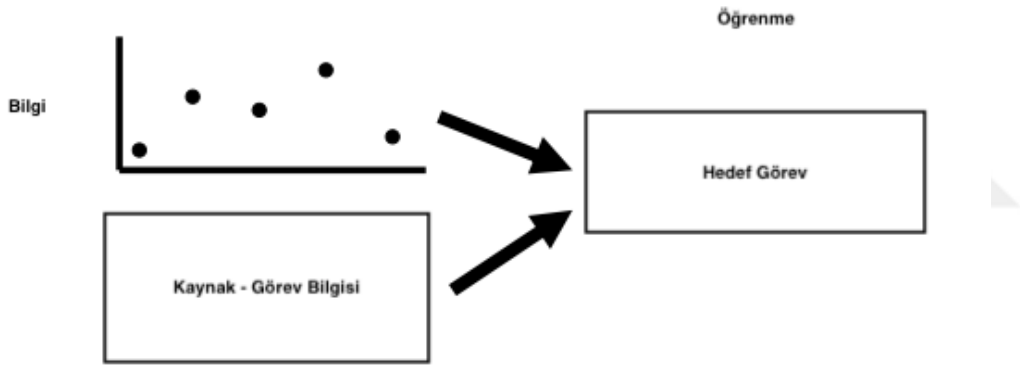
Transfer öğrenme, yeni bir görevi yerine getirirken öğrenme sürecini kolaylaştırmak için geçmişle ilgili görevlerden gelen bilgileri tekrar kullanır. Transfer öğrenmenin amacı, daha verimli bir şekilde öğrenmek için önceki öğrenimden ve deneyimden yararlanmaktır. Transfer öğreniminin faydası, önceki öğrenimlerden kullanılan bilgileri kullandığıdan daha hızlı bir öğrenim sonuçlanmasıdır. Örneğin, İngilizce (Çince vs) öğrenmiş biri için Fransızca öğrenmek biraz daha kolaydır, ve Portekizce (Almanca vs) biliyorsa İspanyolca öğrenmek daha da kolaydır. Bu nedenle, bir öğrenme makinesinin geçmiş öğrenmelerin bir yan ürünü olarak elde edilen soyut bilgiden yararlanmasını, gelecekteki öğrenme problemleri üzerindeki performansını arttırmasını sağlamaktadır [25].

Verilerin doğru iç gösterimi, geleneksel makine öğrenmesi algoritmalarının performansı üzerinde büyük bir etkiye sahip olabilir. Burada, temel varsayımlar, eğitim ve test verilerinin aynı dağıtımdan ya da özellik alanından geldiğidir. Ds = Dt, burada Ds kaynak veri alanı ve Dt hedef veri alanıdır. Bununla birlikte, eğer bu dağılım değiştirilirse

Ds <> Dt , makine öğrenme modelinin pahalı olan yeni etiketli eğitim verilerini kullanarak yeniden oluşturulması gerekecektir. Bu nedenle, bu tür durumlarda bir veya daha fazla kaynak görevleri ekstre edilmesi ve bir hedef görev aktarılması faydalı olur. Şekil 4.3, geleneksel makine öğrenme algoritmaları ile transfer öğrenme teknikleri arasındaki farkı göstermektedir. Şekil 4.4'de transfer öğrenimi ile bir veya daha fazla ilgili görevden gelen bilgi gösterimi mevcuttur.



Şekil 4.3: Geleneksel Makine Öğrenmesi Süreci ile Transfer Öğrenmesi Arasındaki Fark



Şekil 4.4: Transfer Öğrenmi ile Bir veya Daha Fazla ilgili Görevden Gelen Bilgi Gösterimi.

Transfer öğrenimi, standart eğitim verilerinden ayrı olarak ek bir bilgi kaynağına sahip makine öğrenimidir; bir veya daha fazla ilgili görevden bilgi aktarılır.
Bununla birlikte, derin öğrenme modellerinin eğitilmesi için gerekli olan muazzam kaynaklar veya derin öğrenme modellerinin eğitildiği büyük ve zorlu veri setleri göz önüne alındığında, derin öğrenme konusunda transfer öğrenme popülerdir. Transfer öğreniminde, öncelikle bir temel veri kümesi ve görevi üzerine bir temel ağ eğitilir ve daha sonra öğrenilen özellikleri yeniden hedeflenir veya bunları bir hedef veri kümesi ve görevi için eğitilmek üzere ikinci bir hedef ağa aktarılır [26].

Transfer Öğrenimi İki yaygın yaklaşımı vardır.

- Model geliştirme yaklaşımı
- Önceden eğitilmiş model yaklaşımı

4.3.4. Model geliştirme yaklaşımı

Kaynak Görevi Seçimi: Girdi verisi, çıktı verisi ve / veya haritalama sırasında girdiden çıktı verisine kadar öğrenilen kavramlar arasında bir ilişki bulunan veri bolluğu ile ilgili bir tahmin modellemesi problemi seçilir.

Kaynak Model Geliştirme: Daha sonra, bu ilk görev için yetenekli bir model geliştirilir.

Yeniden Model Kaynak göreve uygun olan model daha sonra, ilgilenilen ikinci görevdeki bir model için başlangıç noktası olarak kullanılabilir. Bu, kullanılan modelleme tekniğine bağlı olarak modelin tamamını veya bir kısmını kullanmayı içerebilir.

4.3.5 Önceden eğitilmiş model yaklaşımı

Kaynak Modeli Seçimi: Mevcut modellerden önceden eğitilmiş bir kaynak model seçilir. Birçok araştırma kurumu, seçilecek aday model havuzuna dahil edilebilecek olan büyük ve zorlu veri setlerine ilişkin modeller yayınlar.

Yeniden Modelleme: Önceden eğitilmiş model, daha sonra, ilgilenilen ikinci görevdeki bir model için başlangıç noktası olarak kullanılabilir. Bu, kullanılan modelleme tekniğine bağlı olarak modelin tamamını veya bir kısmını kullanmayı içerebilir.

Ayarlama Modeli: İsteğe bağlı olarak, modelin ilgili görev için mevcut olan girdi-çıktı çifti verilerine uyarlanması gerekebilir. Bu ikinci tip transfer öğrenmesi derin öğrenme alanında yaygındır [27].

4.4 Inception V3

Inception v3, ImageNet veri setinde% 78,1'den daha fazla doğruluk elde ettiği gösterilen yaygın olarak kullanılan bir görüntü tanıma modelidir. Modelin kendisi, konvolüsyonlar, ortalama havuzlama, maksimum havuzlama, ve tamamen birbirine bağlı katmanlar dahil simetrik ve asimetrik yapı taşlarından oluşur. Inception-v3, ImageNet veritabanından bir milyondan fazla resim üzerinde eğitilmiş evrişimli bir sinir ağıdır.

Ağ 48 katman derinliğindedir ve görüntüleri klavye, fare, kurşun kalem ve birçok hayvan gibi 1000 nesne kategorisine ayırabilir. Sonuç olarak, ağ çok çeşitli görüntüler için zengin özellik gösterimleri öğrenmiştir. Ağın 299'a 299'luk bir resim giriş boyutu vardır [28].

Google, 2014 ILSVRC'yi kazanan ve yüzde 6,67'lik bir 5 hata oranıyla Inception V1 olarak bilinen GoogleNet adlı kendi CNN'ini kurdu. Daha sonra model birkaç kez geliştirildi ve değiştirildi. Inception sinir ağının dört versiyonu vardır. Bu makalede, ImageNet veri setinde önceden tanımlanmış bir görüntü tanıma için CNN modeli olan Inception V3 kullanılmıştır [28].

4.4.1 Görüntü sınıflandırması için inception v3 kullanımı

Inception V3 ile transfer oğrenme, mevcut görüntü sinir ağını özel görüntü sınıflandırma görevlerini çözmek ve kullanabilmek için yeniden eğitmeye izin verir. Önceden belirlenmiş Inception V3 modeline yeni veri sınıfları eklemek için, Tensorflow-Görüntü-Sınıflandırma kaynağını kullandık.

Bu kaynak, Inception V3 modelinin varsayılan sürümünü indirmek ve Python 3, Tensorflow ve Keras kullanarak yeni bir resim kümesini sınıflandırmak için yeniden eğitmeyi içerir.

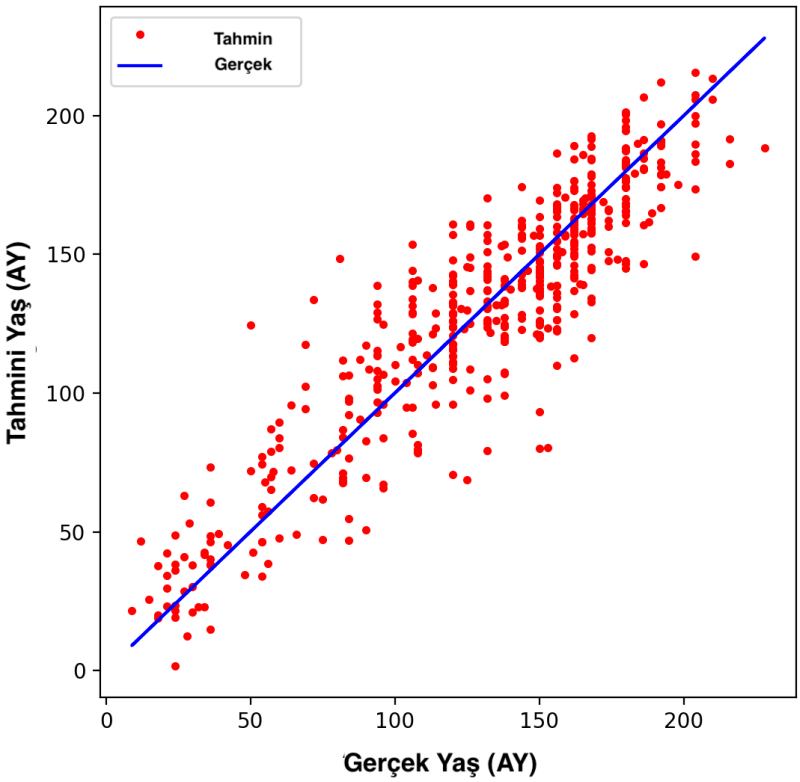
5. SONUÇ VE TARTIŞMA

Aynı görüntü ön işleme ve özellik seçim yöntemleri CNN transfer öğrenme modellerine, VGG16 ile Inception V3 önceden eğitilmiş ağırlıklarını kullanarak, veri setlerine beslemeden önce uygulanmıştır. Tüm veri setleri üzerinde eğitim denilmiştir. Tahmin edilen yaş ile gerçek yaş arasındaki farkı ölçmek için ayların ortalama mutlak hataları (MAE) kullanılmıştır. Veri setinin yüzde 20'si rastgele doğrulama seti olarak seçilmiş, MAE doğrulama setleri üzerindeki iki model kullanılarak elde edilmiştir. VGG16 Transfer Öğrenimi de MAE (Mutlak Hata) sonucu 14.43 gösterirken Inception V3 Transfer Öğrenimi de MAE sonucu 37.29 olarak hesaplanmıştır.

Incepition V3 transfer modelin VGG16 transfer modelinden daha kötü performans sergilediği açıktır. Her iki modelde Adam opitmazyosunu kullanarak eğitilmiştir.

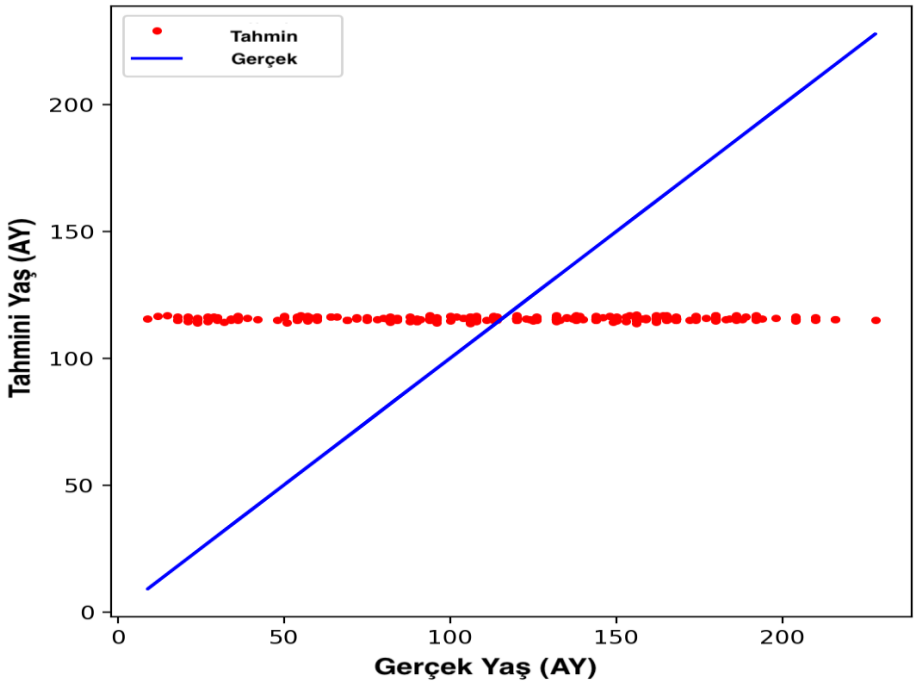
Eğitim sürecinde CNN transfer öğrenme kullanarak önceden eğitilmiş ağırlıkları başlangıç ağırlıklar olarak seçilmiş, model yeni veri setleri üzerinde tekrar eğitilmiş ve en iyi model ağırlığı tespit edilmiştir. Transfer öğrenme de hem önceden eğitilmiş VGG16 hem de önceden eğitilmiş Inception V3 ve her ikiside ImageNet'ten alınmış ağırlıklar, başlangıç ağırlıklar olarak seçilmiş, öğrenim durumu iki yöntem arası kıyaslanmıştır.

Sonuç olarak CNN transfer öğrenme yöntemi ile eğitilmiş model , VGG16 önceden eğitilmiş ağırlıkları 2500 El X-Işını eğitiminde kullanıldığında doğruluk payı %86'ya kadar ulaşmış en iyi yöntem olarak seçilmiştir. Şekil 5.1'de eğitim sonucu veriler üzerinde tahmin edilen yaş ile gerçek yaş arasındaki farkı göstermektedir.



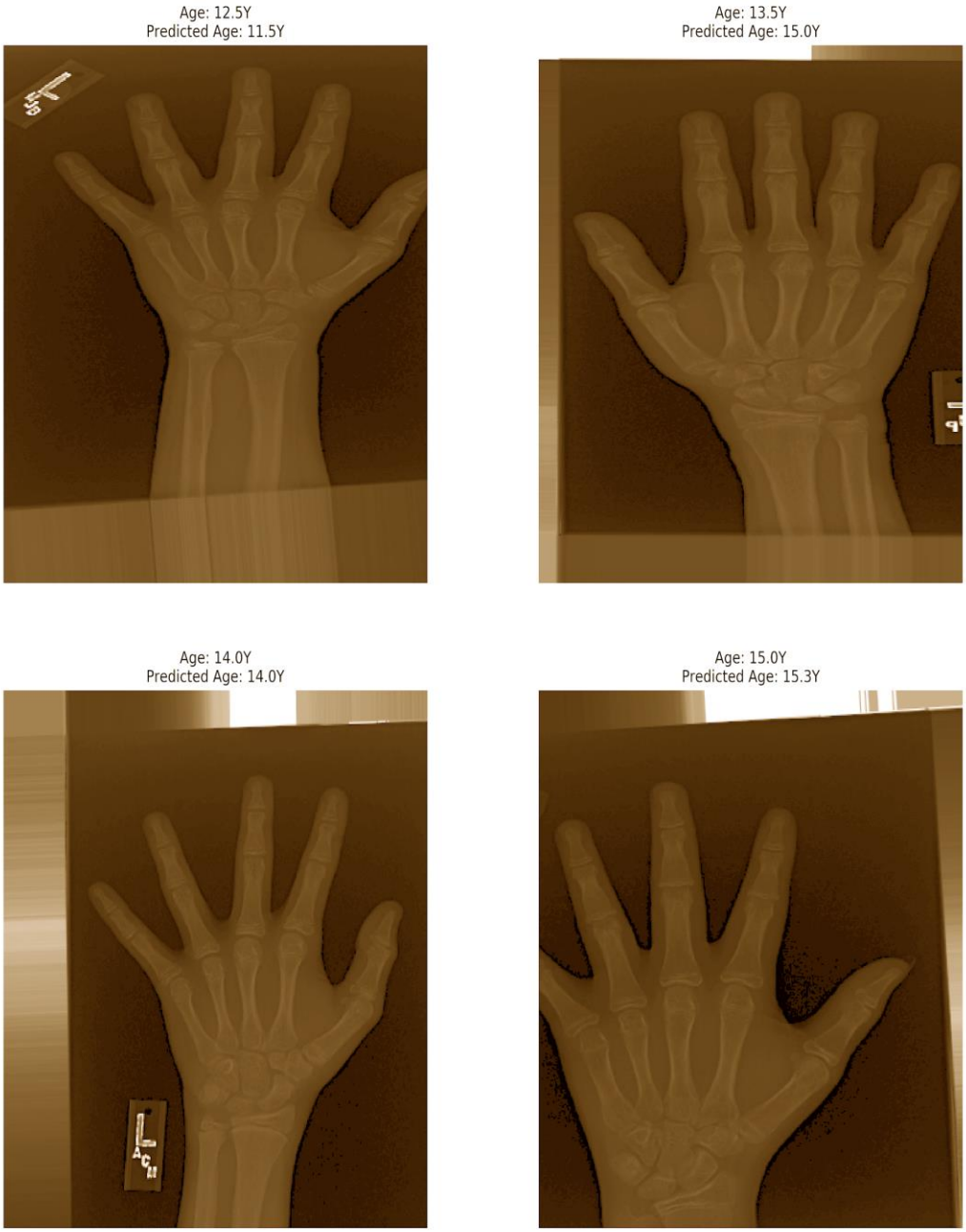
Şekil 5.1: VGG16 ile CNN Transfer Öğrenme Sonucu Gerçek Yaş ile Tahmini Yaş Arasındaki Fark

Inception V3 ile önceden eğitilmiş ağırlıkları 2500 El X-Işını eğitiminde kullanıldığında doğruluk payı %63'e kadar ulaşmıştır. Bu yöntem sonucu doğruluk payı az olarak kabul edilmiştir. Şekil 5.2'da eğitim sonucu veriler üzerinde tahmin edilen yaş ile gerçek yaş arasındaki farkı göstermektedir.



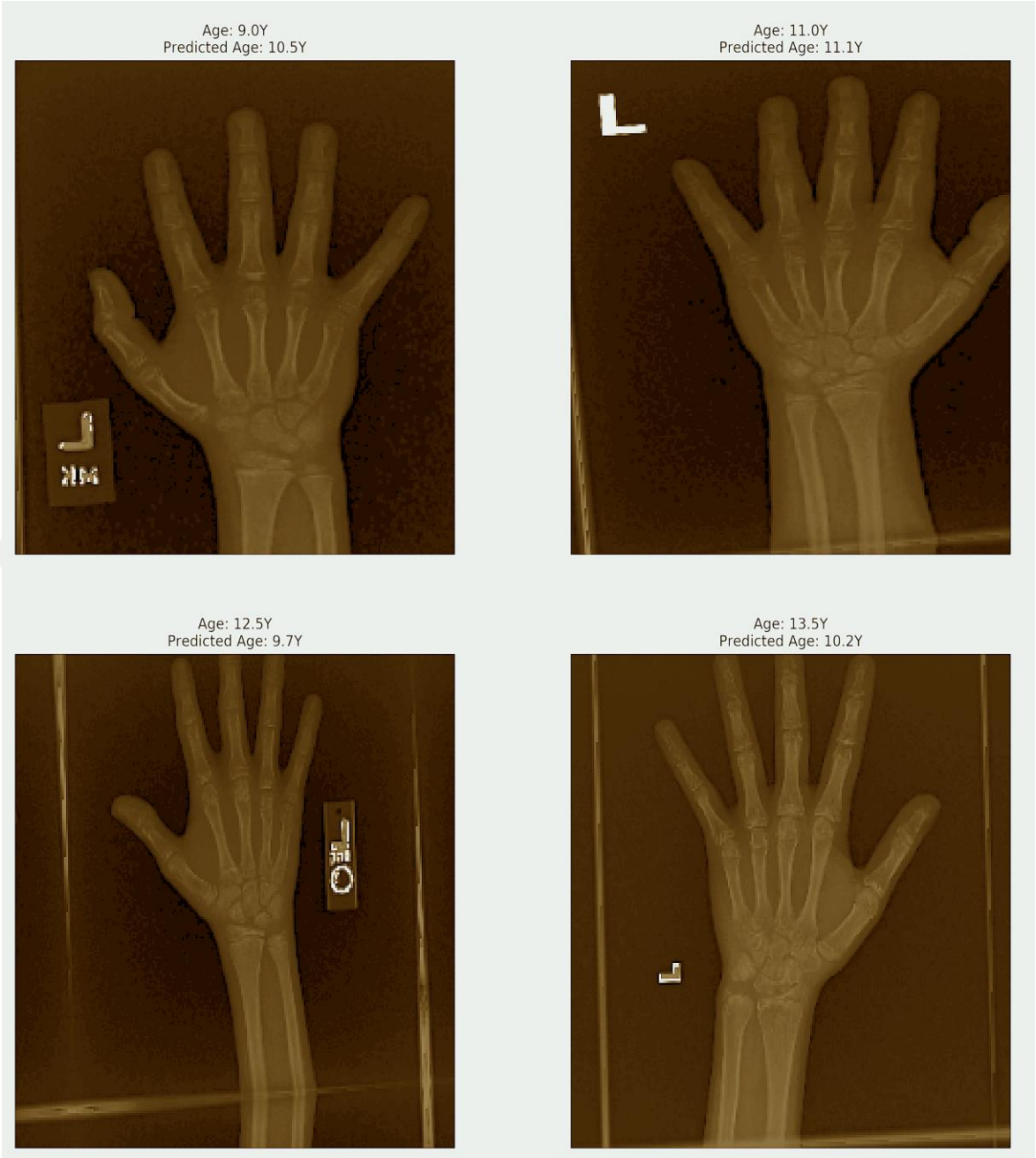
Şekil 5.2: Inception V3 ile CNN Transfer Öğrenme Sonucu Gerçek Yaş ile Tahmini Yaş Arasındaki Fark

CNN Tranfser öğrenme ile ağitilen ve en iyi sonucu alan VGG16 model, görüntü veri setleri üzerinde görüntü işleme teknikleri kullanarak kemik yaşını tahmin etmekte en iyi sonucu sergilemiştir. Aşağıda test verilerinden birkaç tanesi seçilmiştir. Şekil 5.3'de Görüldüğü gibi Gerçek Yaş ile Tahmin edilen yaş arası az bir fark vardır.



Şekil 5.3: Tahmin Edilen Yaş ile Gerçek Yaş VGG16 Transfer Öğrenme Sonucu Örnekler.

CNN Tranfser öğrenme ile ağitilen ve en iyi sonucu alan Inception V3 model, görüntü veri setleri üzerinde görüntü işleme teknikleri kullanarak kemik yaşını tahmin etmekte VGG16 ile kıyaslayarak daha düşük sonuç sergilemiştir. Aşağıda test verilerinden birkaç tanesi seçilmiştir. Şekil 18'de Görüldüğü gibi Gerçek Yaş ile Tahmin edilen yaş arası fark biraz daha yüksektir.



Şekil 5.4: Tahmin Edilen Yaş ile Gerçek Yaş Inception V3 Transfer Öğrenme Sonucu Örnekler

KAYNAKLAR

[1] De Sanctis V, Di Maio S, Soliman A.T, Raiola G, Elalaily R, Millimaggi G. (2019).Hand X-ray in pediatric endocrinology: Skeletal age assessment and beyond. DOI:10.4103/2230- 8210.145076

[2] Lee H, Alkasab T, Tajmir S , Choy G. (March 2017) . Fully Automated Deep Learning System for Bone Age Assessment. Journal of Digital Imaging

[3] Thangam P, Mahendiran T, Thanushkodi K. (2012).Skeletal Bone Age Assessment – Research Directions. Journal of Engineering Science and Technology Review 5 (1) (2012) 90 – 96 ,DOI: 10.25103/jestr.051.16

[4] Bull R.K, Edwards P.D, Kemp P.M, Fry S, Hughes I.A. (1999). Bone Age Assessment: a large scale comparison of the Greulich and Pyle, and Tanner and Whitehouse (TW2) methods. DOI:10.1136/adc.81.2.172

[5] Gilsanz V, Ratib O. (2005). Hand Bone Age A Digital Atlas of Skeletal Maturity. Springer-Verlag Berlin Heidelberg 2005

[6] Chikhalekar A.T. (2016). Analysis of Image Processing for Digital X-Ray. International Research Journal of Engineering and Technology (IRJET)

[7] Jassim, F. A. (2003). Kriging interpolation filter to reduce high density salt and pepper noise. World of Computer Science and Information Technology.

[8] Al-Ayyoub M, Hmeidi I, Rababah H. (2013). Detecting Hand Bone Fractures in X-Ray Images. Jordan University of Science and Technology

[9] Ahmed E, Elatif R, Alser Z. (2015). Median Filter Performance Based on Different Window Sizes for Salt and Pepper Noise Removal in Gray and RGB Images,DOI: 10.14257/ijsip.2015.8.10.34

[10] Dharampal, Mutneja V. (2015). Methods of Image Edge Detection: A Review. J Electr Electron Syst 4:150. doi:10.4172/2332-0796.1000150

[11] Barnes G. T, Karen L. (1989). mage Processing in Digital Radiography: Basic Concepts and Applications. Journal of Digital Imaging

[12] Noord N, Postma E. (2016). Learning scale-variant and scale-invariant features for deep image classification.arXiv:1602.01255

[13] Iliadis N. (2017). Machine learning: the power and promise of computers that learn by example. ISBN: 978-1-78252-259-1,The Royal Society

[14] Brownlee J. (2016). Master Machine Learning Algorithms Discover How They Work and Implement Them From Scratch, Machine Learning Mastery.

[15] Awad M, Khanna R. (2015). Efficient Learning Machines Theories, Concepts, and Applications for Engineers and System Designers,DOI : 978-1-4302-5990-9

[16] Cristianini N, Schölkopf B. (2002). Support vector machines and Kernel methods: the new generation of learning machines, DOI:10.1609/aimag.v23i3.1655

[17] Ben-Hur A, Weston J. (2010). A User's Guide to Support Vector Machines, DOI: 10.1007/978-1-60327-241-4_13

[18] Zhang S, Li X, Zhu X, Zong M, Cheng D .(2017). Learning k for kNN Classification Article in ACM Transactions on Intelligent Systems and Technology,DOI: 10.1145/2990508

[19](URL8)https://www.ibm.com/support/knowledgecenter/en/SS6NHC/com.ibm.swg.im.dashdb.analytics.doc/doc/r_knn_usage.html.

[20] O'Shea K.T, Nash R. (2015). An Introduction to Convolutional Neural Networks Article. arXiv:1511.08458

[21] Albawi S , Albawi S , Abed MOHAMMED T , ALZAWİ S. (2017). Understanding of a Convolutional Neural Network. DOI: 10.1109/ICEngTechnol.2017.8308186

[**22] Hijazi S, Kumar R, Rowen C.** (2015). Using Convolutional Neural Networks for Image Recognition. IP Group, Cadence

[23] Bruna J, Zaremba W, Lecun Y, Szlam A. (2013). Spectral Networks and Locally Connected Networks on Graphs.

[24] Liu I, Li X, Luo P, Loy C.C, Tang X. (2015). Semantic Image Segmentation via Deep Parsing Network.

[25] Yang L, Hanneke S, Carbonell J. (2012). A theory of transfer learning with applications to active learning,DOI 10.1007/s10994-012-5310-y

[26] Torrey L, Shavlik J. (2009). Transfer Learning. University of Wisconsin, Madison WI, USA,DOI: 10.4018/978-1-60566-766-9.ch011

[27] Guyon I, Aha W, Lemaire V, Taylor G, Dror G. (2011) .Unsupervised and transfer learning challenge,DOI: 10.1109/IJCNN.2011.6033302

[28] Barratt S, Sharma R. (2018). A Note on the Inception Score,arXiv:1801.01973

EKLER

EK A Python VGG16 Kodları

EK B Python INCEPTION V3 Kodları

EK A Python VGG16 Kodları

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt # showing and rendering figures
from skimage.io import imread
import os
from glob import glob
base_bone_dir = os.path.join('..', 'input', 'rsna-bone-age')
age_df = pd.read_csv(os.path.join(base_bone_dir, 'boneage-training-dataset.csv'))
age_df['path'] = age_df['id'].map(lambda x: os.path.join(base_bone_dir,
                                                         'boneage-training-dataset',
                                                         'boneage-training-dataset',
                                                         '{}.png'.format(x)))
age_df['exists'] = age_df['path'].map(os.path.exists)
print(age_df['exists'].sum(), 'images found of', age_df.shape[0], 'total')
age_df['gender'] = age_df['male'].map(lambda x: 'male' if x else 'female')
boneage_mean = age_df['boneage'].mean()
boneage_div = 2*age_df['boneage'].std()
# we don't want normalization for now
boneage_mean = 0
boneage_div = 1.0
age_df['boneage_zscore'] = age_df['boneage'].map(lambda x: (x-boneage_mean)/boneage_div)
age_df.dropna(inplace = True)
age_df.sample(3)
age_df[['boneage', 'male', 'boneage_zscore']].hist(figsize = (10, 5))
age_df['boneage_category'] = pd.cut(age_df['boneage'], 10)
from sklearn.model_selection import train_test_split
raw_train_df, valid_df = train_test_split(age_df,
                                  test_size = 0.25,
                                  random_state = 2018,
                                  stratify = age_df['boneage_category'])
print('train', raw_train_df.shape[0], 'validation', valid_df.shape[0])
train_df = raw_train_df.groupby(['boneage_category', 'male']).apply(lambda x: x.sample(500, replace = True)
                                                    ).reset_index(drop = True)
print('New Data Size:', train_df.shape[0], 'Old Size:', raw_train_df.shape[0])
train_df[['boneage', 'male']].hist(figsize = (10, 5))
```

```
from keras.preprocessing.image import ImageDataGenerator
from keras.applications.vgg16 import preprocess_input
IMG_SIZE = (384, 384) # slightly smaller than vgg16 normally expects
core_idg = ImageDataGenerator(samplewise_center=False,
                              samplewise_std_normalization=False,
                              horizontal_flip = True,
                              vertical_flip = False,
                              height_shift_range = 0.15,
                              width_shift_range = 0.15,
                              rotation_range = 5,
                              shear_range = 0.01,
                              fill_mode = 'nearest',
                              zoom_range=0.25,
                             preprocessing_function = preprocess_input)
def flow_from_dataframe(img_data_gen, in_df, path_col, y_col, **dflow_args):
    base_dir = os.path.dirname(in_df[path_col].values[0])
    print('## Ignore next message from keras, values are replaced anyways')
    df_gen = img_data_gen.flow_from_directory(base_dir,
                                    class_mode = 'sparse',
                                    **dflow_args)
    df_gen.filenames = in_df[path_col].values
    df_gen.classes = np.stack(in_df[y_col].values)
    df_gen.samples = in_df.shape[0]
    df_gen.n = in_df.shape[0]
    df_gen._set_index_array()
    df_gen.directory = '' # since we have the full path
    print('Reinserting dataframe: {} images'.format(in_df.shape[0]))
    return df_gen
train_gen = flow_from_dataframe(core_idg, train_df,
                                path_col = 'path',
                               y_col = 'boneage_zscore',
                               target_size = IMG_SIZE,
                                color_mode = 'rgb',
                               batch_size = 32)
```

```
valid_gen = flow_from_dataframe(core_idg, valid_df,
                                path_col = 'path',
                               y_col = 'boneage_zscore',
                               target_size = IMG_SIZE,
                                color_mode = 'rgb',
                               batch_size = 32) # we can use much larger batches for evaluation
# used a fixed dataset for evaluating the algorithm
test_X, test_Y = next(flow_from_dataframe(core_idg,
                                  valid_df,
                                path_col = 'path',
                               y_col = 'boneage_zscore',
                               target_size = IMG_SIZE,
                                color_mode = 'rgb',
                               batch_size = 32)) # one big batch
t_x, t_y = next(train_gen)
fig, m_axs = plt.subplots(2, 4, figsize = (16, 8))
for (c_x, c_y, c_ax) in zip(t_x, t_y, m_axs.flatten()):
    c_ax.imshow(c_x[:,:,0], cmap = 'bone', vmin = -127, vmax = 127)
    c_ax.set_title('%2.0f months' % (c_y*boneage_div+boneage_mean))
    c_ax.axis('off')
from keras.applications.vgg16 import VGG16
from keras.layers import GlobalAveragePooling2D, Dense, Dropout, Flatten, Input, Conv2D, multiply, LocallyConnected2D, Lambda
from keras.models import Model
in_lay = Input(t_x.shape[1:])
base_pretrained_model = VGG16(input_shape =  t_x.shape[1:], include_top = False, weights = 'imagenet')
base_pretrained_model.trainable = False
pt_depth = base_pretrained_model.get_output_shape_at(0)[-1]
pt_features = base_pretrained_model(in_lay)
from keras.layers import BatchNormalization
bn_features = BatchNormalization()(pt_features)
# here we do an attention mechanism to turn pixels in the GAP on an off
attn_layer = Conv2D(64, kernel_size = (1,1), padding = 'same', activation = 'relu')(bn_features)
attn_layer = Conv2D(16, kernel_size = (1,1), padding = 'same', activation = 'relu')(attn_layer)
```

```
attn_layer = LocallyConnected2D(1,
                                kernel_size = (1,1),
                                padding = 'valid',
                                activation = 'sigmoid')(attn_layer)
# fan it out to all of the channels
up_c2_w = np.ones((1, 1, 1, pt_depth))
up_c2 = Conv2D(pt_depth, kernel_size = (1,1), padding = 'same',
               activation = 'linear', use_bias = False, weights = [up_c2_w])
up_c2.trainable = False
attn_layer = up_c2(attn_layer)
mask_features = multiply([attn_layer, bn_features])
gap_features = GlobalAveragePooling2D()(mask_features)
gap_mask = GlobalAveragePooling2D()(attn_layer)
# to account for missing values from the attention model
gap = Lambda(lambda x: x[0]/x[1], name = 'RescaleGAP')([gap_features, gap_mask])
gap_dr = Dropout(0.5)(gap)
dr_steps = Dropout(0.25)(Dense(1024, activation = 'elu')(gap_dr))
out_layer = Dense(1, activation = 'linear')(dr_steps) # linear is what 16bit did
bone_age_model = Model(inputs = [in_lay], outputs = [out_layer])
from keras.metrics import mean_absolute_error
def mae_months(in_gt, in_pred):
    return mean_absolute_error(boneage_div*in_gt, boneage_div*in_pred)
bone_age_model.compile(optimizer = 'adam', loss = 'mse',
                       metrics = [mae_months])
bone_age_model.summary()
from keras.callbacks import ModelCheckpoint, LearningRateScheduler, EarlyStopping, ReduceLROnPlateau
weight_path="bone_age_weights.best.hdf5"
checkpoint = ModelCheckpoint(weight_path, monitor='val_loss', verbose=1,
                             save_best_only=True, mode='min', save_weights_only = True)
reduceLROnPlat = ReduceLROnPlateau(monitor='val_loss', factor=0.8, patience=10, verbose=1, mode='auto', epsilon=0.0001, cooldown=5, min_lr=0.0001)
early = EarlyStopping(monitor="val_loss",
                      mode="min",
                      patience=5) # probably needs to be more patient, but kaggle time is limited
callbacks_list = [checkpoint, early, reduceLROnPlat]
```

```
bone_age_model.fit_generator(train_gen,validation_data=(test_X,test_Y), validation_steps=800,steps_per_epoch=300,epochs = 10,callbacks = callbacks_list)
bone_age_model.load_weights(weight_path)
# get the attention layer since it is the only one with a single output dim
for attn_layer in bone_age_model.layers:
    c_shape = attn_layer.get_output_shape_at(0)
    if len(c_shape)==4:
        if c_shape[-1]==1:
            print(attn_layer)
            break
import keras.backend as K
rand_idx = np.random.choice(range(len(test_X)), size = 6)
attn_func = K.function(inputs = [bone_age_model.get_input_at(0), K.learning_phase()],
           outputs = [attn_layer.get_output_at(0)]
          )
fig, m_axs = plt.subplots(len(rand_idx), 2, figsize = (8, 4*len(rand_idx)))
[c_ax.axis('off') for c_ax in m_axs.flatten()]
for c_idx, (img_ax, attn_ax) in zip(rand_idx, m_axs):
    cur_img = test_X[c_idx:(c_idx+1)]
    attn_img = attn_func([cur_img, 0])[0]
    img_ax.imshow(cur_img[0,:,:,0], cmap = 'bone')
    attn_ax.imshow(attn_img[0, :, :, 0], cmap = 'viridis',
                   vmin = 0, vmax = 1,
                   interpolation = 'lanczos')
    real_age = boneage_div*test_Y[c_idx]+boneage_mean
    img_ax.set_title('Hand Image\nAge:%2.2fY' % (real_age/12))
    pred_age = boneage_div*bone_age_model.predict(cur_img)+boneage_mean
    attn_ax.set_title('Attention Map\nPred:%2.2fY' % (pred_age/12))
fig.savefig('attention_map.png', dpi = 300)
pred_Y = boneage_div*bone_age_model.predict(test_X, batch_size = 32, verbose = True)+boneage_mean
test_Y_months = boneage_div*test_Y+boneage_mean
fig, ax1 = plt.subplots(1,1, figsize = (6,6))
ax1.plot(test_Y_months, pred_Y, 'r.', label = 'predictions')
ax1.plot(test_Y_months, test_Y_months, 'b-', label = 'actual')
ax1.legend()
```

```
ax1.set_xlabel('Actual Age (Months)')
ax1.set_ylabel('Predicted Age (Months)')
ord_idx = np.argsort(test_Y)
ord_idx = ord_idx[np.linspace(0, len(ord_idx)-1, 8).astype(int)] # take 8 evenly spaced ones
fig, m_axs = plt.subplots(4, 2, figsize = (16, 32))
for (idx, c_ax) in zip(ord_idx, m_axs.flatten()):
    c_ax.imshow(test_X[idx, :,:,0], cmap = 'bone')

    c_ax.set_title('Age: %2.1fY\nPredicted Age: %2.1fY' % (test_Y_months[idx]/12.0,
                                                          pred_Y[idx]/12.0))
    c_ax.axis('off')
fig.savefig('trained_img_predictions.png', dpi = 300)
fig.show('trained_img_predictions.png')
```

EK B Python INCEPTION V3 Kodları

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt # showing and rendering figures
# io related
from skimage.io import imread
import os
from glob import glob
# not needed in Kaggle, but required in Jupyter
#%matplotlib inline
base_bone_dir = os.path.join('..', 'input', 'rsna-bone-age')
age_df = pd.read_csv(os.path.join(base_bone_dir, 'boneage-training-dataset.csv'))
age_df['path'] = age_df['id'].map(lambda x: os.path.join(base_bone_dir,
                                                         'boneage-training-dataset',
                                                         'boneage-training-dataset',
                                                         '{}.png'.format(x)))
age_df['exists'] = age_df['path'].map(os.path.exists)
print(age_df['exists'].sum(), 'images found of', age_df.shape[0], 'total')
age_df['gender'] = age_df['male'].map(lambda x: 'male' if x else 'female')
boneage_mean = age_df['boneage'].mean()
boneage_div = 2*age_df['boneage'].std()
# we don't want normalization for now
boneage_mean = 0
boneage_div = 1.0
age_df['boneage_zscore'] = age_df['boneage'].map(lambda x: (x-boneage_mean)/boneage_div)
age_df.dropna(inplace = True)
age_df.sample(3)
age_df[['boneage', 'male', 'boneage_zscore']].hist(figsize = (10, 5))
age_df['boneage_category'] = pd.cut(age_df['boneage'], 10)
from sklearn.model_selection import train_test_split
raw_train_df, valid_df = train_test_split(age_df,
                                  test_size = 0.25,
                                  random_state = 2018,
                                  stratify = age_df['boneage_category'])
print('train', raw_train_df.shape[0], 'validation', valid_df.shape[0])
```

```
train_df = raw_train_df.groupby(['boneage_category', 'male']).apply(lambda x: x.sample(500, replace = True)
                                                    ).reset_index(drop = True)
print('New Data Size:', train_df.shape[0], 'Old Size:', raw_train_df.shape[0])
train_df[['boneage', 'male']].hist(figsize = (10, 5))
from keras.preprocessing.image import ImageDataGenerator
from keras.applications.imagenet_utils import preprocess_input
IMG_SIZE = (224, 224) # default size for inception_v3
core_idg = ImageDataGenerator(samplewise_center=False,
                              samplewise_std_normalization=False,
                              horizontal_flip = True,
                              vertical_flip = False,
                              height_shift_range = 0.15,
                              width_shift_range = 0.15,
                              rotation_range = 5,
                              shear_range = 0.01,
                              fill_mode = 'reflect',
                              zoom_range=0.25,
                             preprocessing_function = preprocess_input)
def flow_from_dataframe(img_data_gen, in_df, path_col, y_col, **dflow_args):
    base_dir = os.path.dirname(in_df[path_col].values[0])
    print('## Ignore next message from keras, values are replaced anyways')
    df_gen = img_data_gen.flow_from_directory(base_dir,
                                     class_mode = 'sparse',
                                    **dflow_args)
    df_gen.filenames = in_df[path_col].values
    df_gen.classes = np.stack(in_df[y_col].values)
    df_gen.samples = in_df.shape[0]
    df_gen.n = in_df.shape[0]
    df_gen._set_index_array()
    df_gen.directory = '' # since we have the full path
    print('Reinserting dataframe: {} images'.format(in_df.shape[0]))
    return df_gen
```

```
train_gen = flow_from_dataframe(core_idg, train_df,
                             path_col = 'path',
                            y_col = 'boneage_zscore',
                            target_size = IMG_SIZE,
                             color_mode = 'rgb',
                            batch_size = 8)

valid_gen = flow_from_dataframe(core_idg, valid_df,
                             path_col = 'path',
                            y_col = 'boneage_zscore',
                            target_size = IMG_SIZE,
                             color_mode = 'rgb',
                            batch_size = 256) # we can use much larger batches for evaluation
# used a fixed dataset for evaluating the algorithm
test_X, test_Y = next(flow_from_dataframe(core_idg,
                                valid_df,
                             path_col = 'path',
                            y_col = 'boneage_zscore',
                            target_size = IMG_SIZE,
                             color_mode = 'rgb',
                            batch_size = 1024)) # one big batch
t_x, t_y = next(train_gen)
fig, m_axs = plt.subplots(2, 4, figsize = (16, 8))
for (c_x, c_y, c_ax) in zip(t_x, t_y, m_axs.flatten()):
    c_ax.imshow(c_x[:,:,0], cmap = 'bone', vmin = -127, vmax = 127)
    c_ax.set_title('%2.0f months' % (c_y*boneage_div+boneage_mean))
    c_ax.axis('off')
from keras.applications.inception_v3 import InceptionV3
from keras.layers import GlobalAveragePooling2D, Dense, Dropout, Flatten
from keras.models import Sequential
base_iv3_model = InceptionV3(input_shape =  t_x.shape[1:], include_top = False, weights = 'imagenet')
base_iv3_model.trainable = False
bone_age_model = Sequential()
bone_age_model.add(base_iv3_model)
```

```
bone_age_model.add(GlobalAveragePooling2D())
bone_age_model.add(Dropout(0.5))
bone_age_model.add(Dense(1024, activation = 'tanh'))
bone_age_model.add(Dropout(0.25))
bone_age_model.add(Dense(1, activation = 'linear')) # linear is what 16bit did
from keras.metrics import mean_absolute_error
def mae_months(in_gt, in_pred):
    return mean_absolute_error(boneage_div*in_gt, boneage_div*in_pred)

bone_age_model.compile(optimizer = 'adam', loss = 'mse',
                       metrics = [mae_months])
bone_age_model.summary()
from keras.callbacks import ModelCheckpoint, LearningRateScheduler, EarlyStopping, ReduceLROnPlateau
weight_path="{}_weights.best.hdf5".format('bone_age')

checkpoint = ModelCheckpoint(weight_path, monitor='val_loss', verbose=1,
                             save_best_only=True, mode='min', save_weights_only = True)
reduceLROnPlat = ReduceLROnPlateau(monitor='val_loss', factor=0.8, patience=10, verbose=1, mode='auto', epsilon=0.0001, cooldown
early = EarlyStopping(monitor="val_loss",
                      mode="min",
                      patience=5) # probably needs to be more patient, but kaggle time is limited
callbacks_list = [checkpoint, early, reduceLROnPlat]
bone_age_model.fit_generator(train_gen,
                                 steps_per_epoch=300,
                                 validation_data = (test_X, test_Y),
                                 epochs = 10,
                                 callbacks = callbacks_list)
bone_age_model.load_weights(weight_path)
pred_Y = boneage_div*bone_age_model.predict(test_X, batch_size = 1024, verbose = True)+boneage_mean
test_Y_months = boneage_div*test_Y+boneage_mean
fig, ax1 = plt.subplots(1,1, figsize = (6,6))
ax1.plot(test_Y_months, pred_Y, 'r.', label = 'predictions')
ax1.scatter(test_Y_months, test_Y_months, 'b-', label = 'actual')
ax1.legend()
ax1.set_xlabel('Actual Age (Months)')
ax1.set_ylabel('Predicted Age (Months)')
plt.show()
ord_idx = np.argsort(test_Y)
ord_idx = ord_idx[np.linspace(0, len(ord_idx)-1, 8).astype(int)] # take 8 evenly spaced ones
fig, m_axs = plt.subplots(4, 2, figsize = (16, 32))
for (idx, c_ax) in zip(ord_idx, m_axs.flatten()):
    c_ax.imshow(test_X[idx, :,:,0], cmap = 'gray')

    c_ax.set_title('Age: %2.1fY\nPredicted Age: %2.1fY' % (test_Y_months[idx]/12.0,
                                                          pred_Y[idx]/12.0))
    c_ax.axis('off')
fig.savefig('trained_img_predictions.png', dpi = 300)
```

ÖZGEÇMİŞ

Ad Soyad **:** Nur ZAKİROĞLU
Doğum Tarihi ve Yeri **:** 25.04.1990 – İstanbul
Email **:** nzakiroglu@gmail.com

Eğitim Durumu

09.2017-06.2019 Fen Bilimleri Fakültesi - Bilgisayar Mühendisliği (YL) 2,94/4.00
İstanbul Aydın Üniversitesi İstanbul, Türkiye

09.2014-06.2016 Fen Bilimleri Fakültesi - Bilgisayar Mühendisliği 2,97/4.00
İstanbul Aydın Üniversitesi İstanbul, Türkiye

09.2007-06.2011 Fen Bilimleri Fakültesi - Bilgisayar Mühendisliği
Halep Üniversitesi Halep, Suriye

İş Deneyimi

07.2018-... Yazılım Geliştirme Uzmanı (1 Yıl)
Kalem Yazılım A.Ş. İstanbul, Türkiye

- C# ile Asp.net Core kullanarak uygulama geliştirme.
- Web service yazma.
- React Native dili ile mobil uygulma geliştirme.
- Full stack olarak çalışma.

01.2016-05.2018 C# Yazılım Geliştirme Uzmanı (2 Yıl)
Ice Bilişim Sistemleri İstanbul, Türkiye

- E-integrasyon sistemleri geliştirme(E-Fatura, E-Arşiv,
- E-İrsaliye, E-Müstahsil)

03.2014-12.2015 Dış Ticaret Bölge Satış Müdürü (1 yıl 8 Ay)
Tech Contactlens Ltd. İstanbul, Türkiye

- Yürt içi ve Yurt dışı müşteriler ile bağlantı kurmak, Broşür ve Katalog tasarımları yapmak.