Gazi Üniversitesi Bilgisayar Mühendisliği

BM496-Proje Raporu

Düşük İşlem Gücüne Sahip Sistemler İçin Gerçek Zamanlı Görseller Kullanılarak Atıkların Geri Dönüşüm Kategorilerine Sınıflandırılması

Grup Danışmanı

Doç. Dr. Hacer Karacan

Grup Üyeleri

Burak TOKMAK - 161180057

Mustafa Serhat USLU – 161180061

İçindekiler

Düşük İşlem Gücüne Sahip Sistemler İçin Gerçek Zamanlı Görseller Kullanılarak Atıkların Geri Dönüşüm Kategorilerine Sınıflandırılması

Mustafa Serhat USLU[1], Burak TOKMAK[2]

1- 161180061
uslu.mserhat2@gmail.com

2- 161180057
buraktokmak07@gmail.com

ÖZETÇE

Çoğu bölgelerde yüzde 10'un altında kalan geri dönüşüm oranına getirilebilecek çok farla çözüm seçeneği bulunmaktadır. Buna rağmen geri dönüşüm için yapılan yatırımların büyük oranı masraflı alt yapılar kurulumunu gerektirdiğinden, çoğu zaman bu alandaki gelişim ertelenmektedir. Bu çalışmada bahsedilen probleme çözüm oluşturabilecek ya da çözümün bir parçası olabilecek bir sistem sunulmaktadır. Düşük işlem gücüne sahip olan sistemlere entegre edilebilecek bir gerçek zamanlı görüntü sınıflandırma sistemi sunulmaktadır. Farklı modeller ve veri seti kombinasyonları sonucunda, MobileNet ağ mimarisine sahip ve TrashNet, Google Images ve kendimizin oluşturduğu görsellerin bir arada kullanıldığı denemede başarılı bir şekilde atıklar 7 farklı sınıfa ayrılabilmiştir. Ayrıca, bazı atık sınıfları da kendi içinde alt sınıflara ayrıştırılarak son denemede daha yüksek başarı oranları elde edilebilmiştir. Ek olarak bu tür benzer niteliklere sahip sınıflar içeren problemlerde (karton kutu ve kağıt), "intersection over union" değerinin 0.5'in üzerine çıkarılmasının model başarısı için faydalı olabileceği görülmüştür.

Anahtar Kelimeler: *Evrişimli sinir ağları (CNN), Gerçek Zamanlı Görsel Sınıflandırma, MobileNet*

Classifying Household Waste with Low Processing Power Systems Using Real Time Images

There are many many solutions to the problem of recycling in our time, but even with all the options the recycling rate of most countries stays below %10. One problem concerning this problem is the cost, since the majority of investments made for recycling requires the installation of costly infrastructures, the development in this area is delayed. In this study, we present a partial solution to this problem. Our solution consists of an image classification system that can be integrated into systems with low processing power. In return the low processing power results in cheaper infrastructure costs. Different models and data set combinations were experimented with, at the end MobileNet network architecture and a data set that consists of TrashNet, Google Images and the visuals created by us were utilized. Waste was successfully divided into 7 different classes. In addition, some waste classes were also subdivided, resulting in higher success rates. In addition, it was seen that increasing the "intersection over union" value above 0.5 in such problems where some classes have similar features, can be beneficial for model success.

Keywords: *Convolutional Neural Networks, Real Time Image Classification, MobileNet*

1. GİRİŞ

Toplumumuzun hızla artan gelişimi ve zamanla çoğalan insan nüfusunun sonucu olarak ortaya çıkan yüksek miktarda atık çevremiz için büyük bir problem haline gelmiştir. Bu problemi hafifletme yöntemlerinden biri de geri dönüşüm oranımızı arttırmaktır. Türkiye çapında inceleme yapıldığında genel geri dönüşüm oranının %7 olduğu görülmektedir. İncelen anketlere göre ise insanların geri dönüşüm yapmamasına yol açan sebepler arasında %25'i yaşadığı bölgede geri dönüşüm altyapısı olmadığını, %10'u çok zaman aldığını, %10'u unuttuğunu, %8'i yüksek maliyetli olduğunu, %6'sı ise geri dönüşüm hakkında bilgisi olmadığını belirtmektedir.

Bu projede, bahsedilen probleme yönelik bir çözüm ya da çözümün bir parçası olabilecek bir sistem sunulmaktadır. Çözüm, düşük işlem gücüne sahip cihazlar akılda bulundurularak, alt yapı kurulum aşamasında yüksek masrafları önleyecek, gerçek zamanlı görüntü işlemeyle sınıflandırma yapabilen bir otonom yapıyı ele almaktadır.

1.1. Proje Amacı

Bu proje, toplam %54'lük önlenebilir geri dönüşüm açığının kalıcı şekilde giderilmesi için, ortak alanlardan başlamak üzere atık toplama ünitelerine entegrasyon hedefli, bir yapay zeka (Görüntü İşleme) çözümü üzerinde durmaktadır. Projenin amaçları arasında; gerçek zamanda yüksek başarı oranıyla kağıt, karton, cam, metal ve plastik geri döşüm sınıflarının tespit edilmesi vardır. Ayrıca, bu çözümün ulaşılabilirliğinin arttırılması için, oluşturulan modelin düşük işlem gücüne sahip mobil cihazlara uyarlanması sağlanacaktır.

1.2. Alanda Yapılmış Çalışmalar

Raporun bu kısmında nesne tanıma alanında geçmişte yapılmış benzer çalışmalardan bahsedilecektir.

S. Christian, T. Alexander ve E. Dumitru [15] 2013 yılında yaptıkları çalışmada nesne tanıma alanındaki gerekliliklerden biri olan nesnelerin etrafını sınırlayan kutuları (bounding box) birden fazla nesne için ve elle çizme gerektirmeyecek bir şekilde yüksek kesinlikle çizebilmek için temelinde derin sinir ağları bulunan bir regresyon modeli kullanarak nesnelerin etrafına sınırlayıcı kutu çizme konusunda yüksek başarı elde etmişlerdir.

2016 yılında Y. Keiji, T. Ryosuke ve O. Koichi [16] tarafından evrişimsel sinir ağları (convolutional neural networks) kullanılarak yapılan nesne tanımanın etkin bir şekilde mobil platformlara taşınması üzerine çalışmışlardır. Araştırmacıların ağ içinde ağ (Network-in-Nerwork/NIN) yapısını kullanarak oluşturdukları evrişimsel sinir ağında hiç sık (dense) katman bulunmazken 12 tane evrişimsel katman bulunmaktadır. Ağ içinde ağın (NIN) sahip olduğu bu basit mimari sayesinde araştırmacılar mobil cihazların daha hızlı bir şekilde nesneleri tanıyabilmelerini sağlamışlardır.

2014 yılında Singapor Nöroteknoloji Enstitüsü ve Singapor Üniversitesi [17] tarafından hareket halindeki nesnelerin gerçek zamanlı tanımak için gerçekleştirilen çalışmada araştırmacılar asenkron zaman tabanlı görüntü sensörü (Asynchronous Time-based Image Sensor/ATIS) ile birlikte Neuflow adlı evrişimsel sinir ağını birlikte kullanan araştırmacılar nesneleri tanımada %99.10'a varan başarı elde etmişlerdir.

2015 yılında R. Joseph, D. Santosh, G.Ross ve F.Ali [18] tarafından yapılan çalışmada araştırmacılarnesne tanıma problemi için GoogLeNet'ten esinlenerek geliştirdikleri ve "You Only Look Once (YOLO)" adını verdikleri bir evrişimsel sinir ağı modeli geliştirmişlerdir. YOLO 24 adet evrişimsel katman ve 2 sık katmandan meydana gelmektedir.

2012 yılında J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, A. W. M. Smeulders [19] tarafından yapılan çalışmada araştırmacılar nesne tanıma probleminde nesnelerin yerini belirleyebilmek için seçici arama (Selective Search) adını verdikleri yöntemden yaralanmışlardır. Bu yöntem tam kapsamlı arama (exhaustive search) ve segmantasyon (segmentation) algoritmalarının iyi yanlarını bir arada kullanarak nesnelerin yerini belirlemede % 87,9 başarı oranı elde etmişlerdir.

M. Gaurav, Y. Kaushal, G. Mohit ve K. Narayanan [20] 2016 yılında yaptıkları çalışmada kullanıcılarının çektiği fotoğraflarda çöp yoğunluğu fazla olan alanları tespit eden ve fotoğrafları coğrafi olarak etiketleyen SpotGarbage adlı mobil uygulamayı geliştirmişlerdir. Uygulamanın atıkları tespit etmekte kullandığı evrişimsel sinir ağı Garbage In Images (GINI) adlı veri seti üzerinde eğitilmiş olup %87.69 başarı oranı yakalamayı başarmıştır.

2015 yılında G. Ross [21] tarafından nesne tanıma için Hızlı Bölge Tabanlı Evrişimsel ağ (Fast Region-based Convolutional Network/Fast R-CNN) kullandığı çalışmasında diğer R-CNN modellerine göre daha hızlı ve daha doğru sonuçlar elde etmeyi başarmış ve PASCAL VOC 2012 veri setinde %66 başarı oranı elde etmiştir

H. Kaiming, Z. Xiangyu, R Shaoqing ve S. Jian [22] tarafından 2015 yılında gerçekleştirilen çalışmada araştırmacılar görsel tanıma sistemlerindeki başarıyı arttırmak için geliştirdikleri "Spatial Pyramid Pooling (SPP)" adlı yöntemi kullanmışlardır. SPP' yi kullanan araştırmacılar mevcut R-CNN'lere "Spatial Pyramid Pooling Layer" ekleyerek Pascal VOC 2007 veri setinde mevcut sistemlerin başarı skorlarını arttırmayı başarmışlardır.

O. Wanli et al. [23] 2014 yılında gerçekleştirdikleri çalışmada deforme olmuş nesnelerin tanınmasındaki başarı oranını arttırmak için deformasyon kısıtlı pooling (deformation constrained pooling/def-pooling) adlı havuz (pooling) katmanını geliştirmişler ve def-pooling kullanmayan R-CNN'lerin daha önce %31 olan başarı oranlarını %45'e kadar çıkarmayı başarmışlardır.

1.3. Önerilen Çözüm Yolu ve Rapor Taslağı

Bu proje kapsamında önerilen çözüm yolu gerçek zamanlı bir görüntü sınıflandırma sistemidir. Sistem mimarisi olarak işlem hızından dolayı MobileNet seçilmiştir. Günlük yaşama dair bir veri setiyle eğitilen bu ağ yapısı geri dönüşüm sınıfları için tekrar eğitilecektir. Bazı geri dönüşüm sınıfları renk ve şekil farklılıklarından dolayı kendi içlerinde de alt sınıflara ayrılmıştır. Rapor taslağı ise giriş bölümünden sonra; kullanılan araç ve yöntemler, gerçekleştirilen çalışma, elde edilenlerin değerlendirilmesi ve sonuç kısımlarını içerir.

2. KULLANILAN ARAÇ VE YÖNTEMLER

Projemizin kurulumunda kullanıcağımız yazılım geliştirme modeli Şelale yazılım geliştirme modelidir. Verilerden değer üretmek için kullnılacak yöntem ise “Convolutional Neural Networks”(CNN)’dir. Derin öğrenmenin bir sınıfı olan CNN görsel verilerin işlenmesinde sıklıkla kullanılan bir yöntemdir.

Gerçek zamanlı değer üretebilmek içinse lokalizasyon ve sıflandırma algoritmalarını birleştiren, cnn’lerden oluşan MobileNet ağ yapısından yararlanılmıştır. Spesifik olarak SSD_MOBILENET modelinin ‘COCO’ veri seti üzerinde eğitilmiş versyonunu ‘trasfer learning’ metoduyla kendi veri setimize uygun biçimde eğittik.

Projeyi gerçekleştirmek için kullanılacak temel yazılım dili altyapı desteği ve kullanım kolaylığından dolayı Pythondur. Python kullanılırken yararlanması planlanan kütüphaneler ise: Pandas, Sklearn, Matplotlib, Keras ve TensorFlow’dur.

Keras, Python'da yazılmış açık kaynaklı bir sinir ağı kütüphanesidir. TensorFlow, Microsoft Cognitive Toolkit, R, Theano veya PlaidML'in üzerinde çalışabilir. Bizim durumumuzda burada TensorFlow devreye girer.

TensorFlow, Google tarafından oluşturulan ve hızlı sayısal hesaplamalar yapmak için kullanılan bir Python kütüphanesidir. Doğrudan Derin Öğrenme modelleri oluşturmak için veya TensorFlow'un üzerine inşa edilen süreci basitleştiren sarmalayıcı kütüphaneler ile birlikte kullanılabilecek bir kütüphanedir.

Pandas, veri manipülasyonu ve analizi için Python programlama diline odaklı yazılmış bir yazılım kütüphanesidir. Özellikle, sayısal tabloları ve zaman serilerini değiştirmek için veri yapıları ve işlemleri sunar.

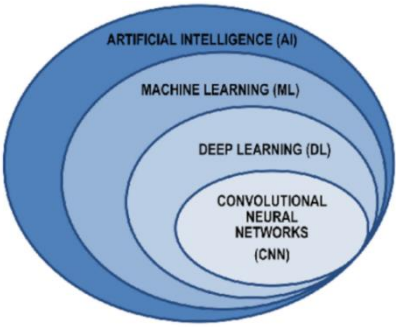
Sklearn, çeşitli sınıflandırma, regresyon ve kümeleme algoritmalarına sahiptir. Bundan dolayı kullanımı projemiz için temel bir gereksinimdir.

Matplotlib, genel amaçlı GUI araç takımlarını kullanarak uygulamalara çizim yerleştirmek için nesne yönelimli bir API sağlar.

En son aşamada ise oluşturulan model optimize edildikten sonra, düşük işlem gücü olan olan cihazlarda test edilebilmesi için Android Studio kullanılmıştır. Bu araç Android işletim sistemi kullanan mobil cihazlar için önemlidir.

2.1) Evrişimsel Sinir Ağlarını Oluşturan Teknolojiler

Günümüzde birbirinin yerini alabilecek şekilde kullanılabilen makine öğrenmesi, yapay zeka ve derin öğrenme gibi kavramlar, birbirleriyle ilişkili olduğundan tam olarak anlaşılabilmeleri için detaylarıyla incelenmelilerdir. Öncelikle bahsedilmesi gerekir ki, makine öğrenmesi ve derin öğrenme arasında da bir hiyerarşik yapı bulunmasına rağmen, ikiside yapay zeka tekniklerini uygulamanın yöntemleridir, diğer bir deyişle yapay zekanın alt dallarını oluşturan parçalarıdır. Bir alt kademeye inildiğinde ise makine öğrenmesi ve derin öğrenme arasındaki ilişkiyle karşılaşırız, burada da temel olarak bilmemiz gereken, derin öğrenmenin yapay sinir ağları tarafından uygulanan bir makine öğrenmesi tekniği olduğudur. Bundan dolayı, makine öğrenmesi de derin öğrenmeyi kapsamaktadır. Son olarak ta artık şu sonuca varabiliriz, evrişimli sinir ağları derin öğrenme algoritmalarına örnektir.



Şekil 1: *Evrişimli sinir ağlarını oluşturan teknolojiler arasındaki ilişki* [4].

Bu raporun ana konusu olan, evrişimli sinir ağlarına (CNN) detaylı bir giriş yapabilmemiz için, ilk olarak, birbiriyle iç içe geçmiş olan, derin öğrenme ve makine öğrenmesi kavramları incelenecektir. Böylece, yapay sinir ağlarından oluşan bir derin öğrenme algoritması örneği CNN'leri daha kapsamlı bir şekilde inceleme fırsatımız olacaktır. Dolayısıyla, CNN'leri oluşturan teknolojiler hakkında ön bilgi vermeye makine öğrenmesinden başlayarak, genelden özele doğru bir akış sağlayabiliriz.

2.1.1) Makine öğrenmesi

Makine öğrenmesi, kullanılan algoritmalar yardımıyla makinenin veriyi analiz etmesi, veriden öğrenmesi ve daha sonra yeni veriler hakkında kararlar vermesi ya da tahminlerde bulunmasıdır [3]. Makine öğrenmesinin geleneksel programlamaya kıyasla farkı geleneksel programlamadaki gibi makinenin belirli bir görevi yerine getirmesi için elle tek tek girilmesi gerekli bir dizi talimatı yazmaktansa söz konusu makinenin çok sayıda veri kullanılarak eğitilmesi ve kullanılan algoritmaların makineye görevi yerine getirmesi için tek tek elle bir dizi talimat girilmesine gerek kalmadan o görevi nasıl yerine getirileceği öğrenmesini sağlamasıdır.

Daha da kısası, makine öğrenmesi, verilerden bilgi çıkarmakla ilgilidir. Makinelerin açıkça programlanmadan geçmiş verilerden veya deneyimlerden öğrenmesini sağlayan yapay zekanın bir alt alanı olarak da tanımlanabilir [1].

Makine öğrenmesinin kullanım amacı bir bilgisayar sisteminin açıkça programlanmadan geçmiş verilerini kullanarak tahminlerde bulunmasına veya bazı kararlar almasına olanak tanımaktır. Makine öğrenme modelinin doğru sonuç üretebilmesi veya bu verilere dayalı tahminler verebilmesi için çok miktarda yapılandırılmış ve yarı yapılandırılmış veri kullanır [1].

Makine öğrenmesi temel olarak kullanılan verilerin getirdiği algoritma gereksinimlerine göre ikiye ayrılabilir:[2]

- Gözetimli Öğrenim (Supervised Learning)
- Gözetimsiz Öğrenim (Unsupervised Learning)

Gözetimli öğrenim, makine öğrenmesi algoritmasının hâlihazırda etiketlenmiş veriler kullanılarak öğrenmesi ve bu etiketlenmiş verilerden bir sonuca varması ile son bulan öğrenimdir. Gözetimsiz öğrenim ise makine öğrenmesi algoritmasının etiketsiz veriler kullanarak öğrenmesi ve bu etiketsiz verilerden bir sonuca varmasıyla son bulan öğrenimdir.

Çoğu zaman, makine öğrenmesi ve yapay zeka birbirinin yerine geçebilecek kavramlar gibi zaman zaman kullanılsa da teknik olarak aynı kavramlar değildir. Yapay zeka, makine öğrenmesini kapsamaktadır. Diğer bir ifadeyle, yapay zeka uygulamalarında makine öğrenmesi kullanılabilir. Makine öğrenmesi yapay zeka tasarımının bir parçası olarak da düşünülebilir. Teknik açıdan bakıldığında, Yapay zeka, problemleri öğrenebilen ve çözebilen programlar geliştirmeyi amaçlayan bir dizi tekniği kapsayan çok geniş bir araştırma alanını ifade eder. Makine öğrenmesi ise, verilerden kendi kendine öğrenme ile ilgili yapay zeka alanıdır.

2.1.2) Yapay Sinir Ağları

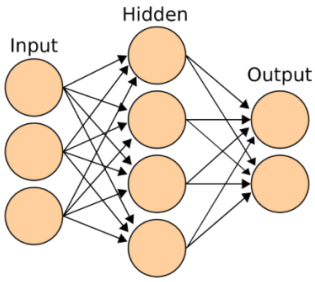
Yapay sinir ağları, insan beyninde yer alan sinir ağlarından esinlenerek tasarlanmış makine öğrenmesi sistemleridir [8]. Yapay sinir ağlarını oluşturan yapı birbirine bağlı yapay nöron adı verilen birimler oluşturur. Bu nöronlar arasındaki her bir bağ bir nörondan diğerine veri iletebilir, sinyali alan nöron kendisine iletilen bu sinyali işler ve kendisine bağlı diğer nöronlara da bu sinyali iletir.
Nöronlar katmanlar halinde organize edilmiştir, bu katmanlar 3'e ayrılırlar [9]:

- Girdi (Input) Katmanı
- Gizli (Hidden) Katman(lar)
- Çıktı (Output) Katmanı

Her bir katman kendi girdisi üzerinde farkı dönüşümler uygulayabilir yapıdadır. Katmanlardaki veri akışı girdi katmanından çıktı katmanına doğru olur. Girdi ve çıktı katmanı arasında bulunan her katman gizli katman olarak adlandırılır. Yapay sinir ağlarında bu katmanlara ek olarak kullanılan başka katmanlarda bulunmaktadır. Bu katmanlar [9]:

- Sık (Dense) Katman
- Evrişimli (Convolutional) Katman
- Havuzlama (Pooling) Katmanı
- Tekrarlı (Recurrent) Katmanlar
- Normalizasyon (Normalization) Katmanı

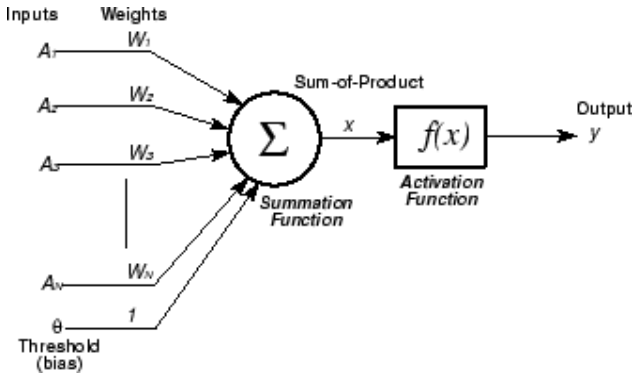
Evrişimli Sinir Ağlarının detayları anlatılan bölümde bu katmanlar detaylı bir şekilde incelenecektir.



Şekil 2: *Girdi, Gizli ve Çıktı katmanlarından oluşan basit bir yapay sinir ağı örneği [9].*

Yukarıdaki görselde basit bir yapay sinir ağı örneği görülmektedir. Burada yer alan, girdi katmanında bulunan 3 düğüm (node), veri setindeki her bir örneğe ait ayrı birer özelliği (feature) temsil etmektedir. Girdi katmanı ve çıktı katmanı arasındaki her bir bağlantı, bir önceki katmandaki çıktıyı bir sonraki katmana girdi olarak iletir. Bağlantıların çıktı olarak alıcı katmana ilettikleri değerin ne olduğunu anlamak için önce bağlantıların sahip oldukları ağırlıklardan (weights) söz edilmelidir.

Bir nörondan diğerine olan her bir bağlantı kendisine ait rastgele atanmış bir ağırlık değerine sahiptir ve bu ağırlık değerleri 0 ile 1 arasında değişim gösterebilmektedir. Ağırlıkları aynı zamanda nöronlar arasındaki bağlantıların kuvveti olarak da düşünebiliriz. Girdi katmanı girdiyi ilk aldığında o girdi bir sonraki katmana nöronlar arasındaki bağlantılar aracılığıyla ve o bağlantılara atanmış olan ağırlık değerleriyle çarpılarak iletilecektir. Hedef nörona bağlı her bir bağlantının bu çarpım ile elde ettiği değerler toplanıp bu toplam değeri hedef nöronun aktivasyon fonksiyonuna (Activation Function) verilir [6]. Aktivasyon fonksiyonu kendisine verilen bu toplam değerini 0 ile 1 arasındaki bir sayıya dönüştürür. Aktivasyon fonksiyonundan çıktı olarak aldığımız bu değer bir sonraki katmanda bulunan hedef nöronun girdisidir. Bağlantıların ağırlıkları ve aktivasyon fonksiyonunun işleyişi hakkında tüm bu anlatılanların görsel ile ifade edilmiş hali Şekil 3'te gösterilmektedir.



Şekil 3*: Nöronun temel yapısı [6].*

Yapay sinir ağları eğitilirken sistemin temel olarak yapmaya çalıştığı şey, bir optimizasyon problemini çözmektir. Sistemin optimize etmeye çalıştığı şey işe bu bağlantılar arasındaki ağırlık değerleridir. Ağırlıkların nasıl optimize edileceği modelde kullanılan optimizasyon algoritmasına bağlı olarak değişir. Yapay sinir ağı modellerinde en çok kullanılan optimizasyon algoritması "Stochastic Gradient Descent" (SGD) algoritmasıdır. SGD algoritmasının amacı modelin kayıp fonksiyonunu (Loss Function) minimize etmektir. Minimize etmeye çalıştığımız bu loss değeri yapay sinir ağımızın fotoğrafın tahmin ettiği etiket değeri ile fotoğrafın asıl etiket değeri arasındaki farktır. SGD algoritması nöronlar arasındaki bağlantıların ağırlıklarını bu loss fonksiyonunu sıfıra olabildiğince yaklaştıracak bir şekilde yeniden atamaya/güncellemeye çalışır. SGD'in yaptığı bu ağırlık güncellemeleri "back propagation" olarak adlandırılır [9].

Ayrıca, yapay sinir ağı modellerinin eğitim sürecinde verinin modelden her bir geçişine devir (Epoch) denir. SGD her bir devir ile nöronlar arasındaki bağlantıların ağırlıkları üzerinde yapılan bu yeniden değer atama işleminin loss fonksiyonun değeri sıfıra mümkün olduğunca yaklaştırılıncaya kadar devam etmesi sonucu yapay sinir ağı öğrenmiş (nöronlar arasındaki bağlantıların ağırlıkları loss fonksiyonunu minimize eden değerlere ulaşmış) demektir [6].

2.1.3) Derin Öğrenme

Derin öğrenme, makine öğrenmesinin insan beynindeki sinir ağlarının yapısı ve fonksiyonlarından esinlenilerek oluşturulmuş algoritmaları kullanan bir alt alandır. Derin öğrenmede de verilerden öğrenen algoritmalar kullanılmaktadır, fakat kullanılan algoritmaların yapısı beynin sinir ağlarının yapısı ve fonksiyonlarına dayanarak oluşturulmuştur.

Derin öğrenme daha teknik bir bakış açısıyla tanımlanacak olursa; verileri doğrusal olmayan çok katmanlı bir düzende işleyerek, gözetimli ve gözetimsiz durumlarda nitelik çıkarımı, nitelik dönüşümü, desen analizi ve genel sınıflandırma yapabilen yapay sinir ağları olarak tanımlayabiliriz [3].

Günümüzde kullanılan derin öğrenme mimarisi temel olarak özellik çıkarma ve dönüştürme için birden fazla doğrusal olmayan işleme katmanı kullanan yapay sinir ağlarına dayanmaktadır, buna evrişimli sinir ağları da dahildir.

Makine öğrenmesi ve derin öğrenme arasında seçim yaparken, yüksek performanslı bir GPU ya ve çok sayıda etiketlenmiş veriye erişimimizin olup olmadığı değerlendirilmelidir. Elimizde bunlar olmadığında, derin öğrenme yerine makine öğrenimini kullanmak daha mantıklı olabilir. Derin öğrenme genellikle daha karmaşıktır, bu nedenle güvenilir sonuçlar elde etmek için yüklü miktarda veriye ihtiyacımız olur.

Derin öğrenme modelleri olarak değerlendirilebilecek sınıfta çeşitli algoritmalar kullanılabilmektedir, bu algoritmaların biri de bu raporda detaylı bir şekilde işlenecek olan CNN algoritmalarıdır. Hiçbir ağ mükemmel kabul edilmese de, bazı algoritmalar belirli görevleri yerine getirmek için daha uygundur.

2.2) Evrişimli Sinir Ağları

Bir derin öğrenme algoritması olan evrişimli sinir ağları, çok katmanlı yapay sinir ağlarının bir türüdür. Bu algoritma, hayvanların görme merkezinden esinlenerek tasarlanmıştır. Kullanım alanları incelendiğinde, görüntü ve ses işleme, doğal dil işleme ve biyomedikal gibi alanlar içermesine rağmen, en iyi sonuçları görüntü işleme alanında vermektedir [26].

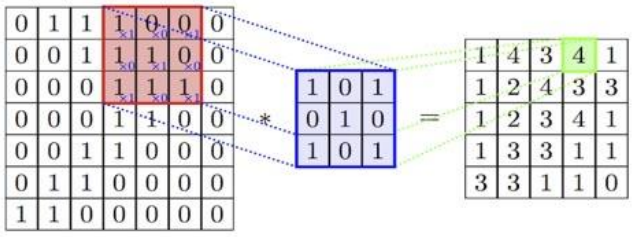
CNN'ler, işledikleri verilerde örüntü keşfedip bu örüntüleri anlamak ve bu örüntülerden bir anlam çıkarmakta uzmanlaşmış bir yapay sinir ağı olarak da düşünülebilir. Evrişimli sinir ağlarını çok katmanlı ileriye dönük bir yapay sinir ağı sınıfı olan "Çok Katmanlı Algılayıcı" dan (Multilayer Perceptron – MLP) ayıran özelliği evrişimli katman olarak da adlandırılan gizli katmanlarıdır. Evrişimli sinir ağlarında genellikle evrişimli katmanlar dışında başka katmanlar da kullanılmaktadır, ancak CNN'lerin temelini bu Evrişimli katmanlar oluşturur [14].

2.2.1) Evrişimli Sinir Ağları Nasıl Çalışır?

Tıpkı diğer derin öğrenme algoritmalarına benzer olarak, evrişimli sinir ağları da bir girdi alır ve girdi üzerinde birtakım dönüşümler uygulayarak bu dönüşüme uğramış girdiyi çıktı olarak bir sonraki katmana iletir.

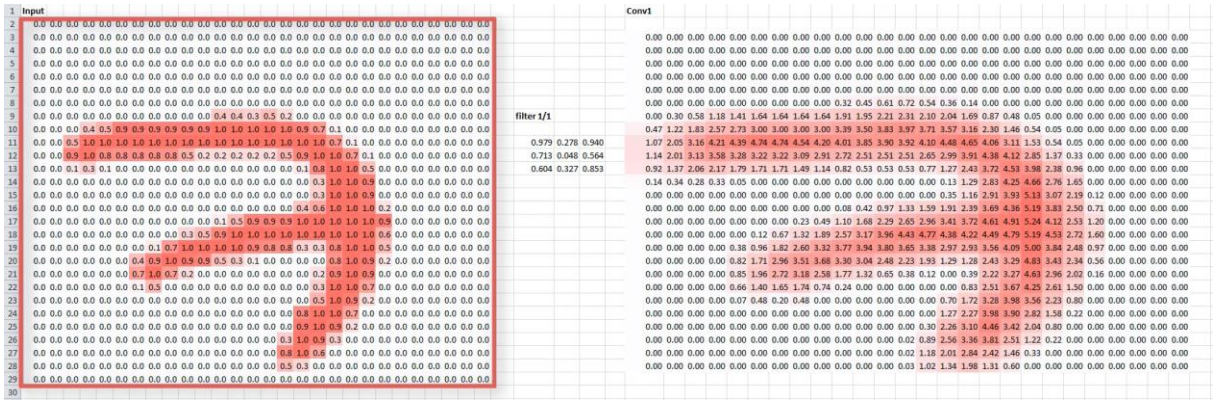
Evrişimli katmalar örüntülerin farkına varabilmektedirler ve bu özelliklerini her evrişimli katmanda kaç tane kullanılacağını belirtmemiz gereken filtrelere borçludurlar [6]. Örüntüleri tespit eden aslında bu filtrelerdir. Filtrelerin fark

ettikleri örüntülerden kasıt, verilen resimlerde bulunan birden çok kenar (Edge), şekil (Shape), doku (Texture), objelerden (Objects) herhangi biri olabilir. Filtrelerin tespit edebileceği örüntülerden biri olan kenarları tespit eden filtre kenar filtresi olarak adlandırılır, bazı filtreler köşelere ek olarak daire, kare gibi basit geometrik şekilleri de ayırt edebilirler. Bu basit ve bir bakımdan geometrik filtreler evrişimli sinir ağlarının ön kısımlarında görülen şeylerdir, Evrişimli sinir ağlarının derinlerine gittikçe bu filtreler giderek daha komplike hale gelirler. Böylece daha derinlerdeki katmanlarda filtrelerimiz kenar ve basit şekilleri ayırt etmektense göz, kulak gibi belirli objeleri ayırt etmeye başlayabilir. Daha derindeki katmanların filtreleri ise köpek, kedi, kertenkele gibi daha da karmaşık objeleri ayırt edebilirler. Filtreler satır ve sütun sayılarına bizim karar verdiğimiz ve rastgele değerler atanmış küçük bir matris olarak düşünülebilirler. Örnek olarak, şekil 4'te 3x3 boyutlarında bir filtre görülmektedir.



Şekil 4: *CNN'de 3x3 boyutlarında bir filtre* [6].

Şekil 4. de görülen 7x7 boyutundaki matris CNN'in girdi olarak aldığı resimdeki piksellerin 1 ve 0'lar ile ifade edilmiş halidir. Şekil 4. de görülen 3x3'lük filtre 7x7'lik matriste bulunan her bir 3x3'lük piksel bloğu üzerinde kayarak resim üzerindeki her bir 3x3'lük piksel bloğunu tarar. Bu kayma işlemine aynı zamanda "Evriştirme"de (Convolving) denilmektedir.



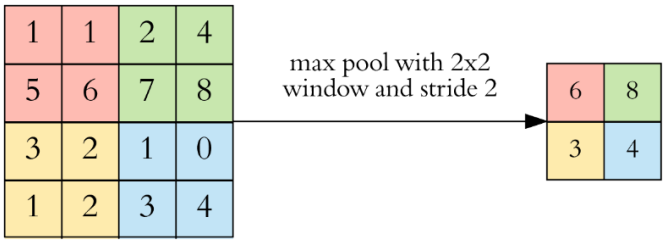
Şekil 5: *Filtreleme işlemi sonrası oluşan boyut azalması* [7].

Filtre orijinal resmin üzerinde kayarak resmin tamamını taradığında filtreleme sonucu oluşan çıktı matrisinin boyutu filtreleme öncesi resmin boyutundan daha küçüktür. Bunun görsel bir örneği de yukarıda verilen şekil 5 te görülebilir. Boyuttaki bu düşüşü hesaplamak için kullanılan denklem, orijinal resim boyutunun n x n olduğu ve filtre boyutu f x f kabul edilirse:

$$\underline{Çıktı\ Boyutu} = (n-f+1)\ x\ (n-f+1) \qquad (1)$$

Dolayısıyla bu modelde kullanılan resmin boyutunun her evrişimli katmandan geçerken bir küçülmeye uğrayacağı anlamına geliyor ve bu istenmeyen bir durumdur. Özellikle hâlihazırda görece küçük boyutlu resimlerle model eğitilmeye başlanmış ise sadece birkaç evrişimli katmandan geçtikten sonra resmin boyutu o kadar küçülebilir ki ortaya çıkan çıktı artık hiçbir anlam ifade etmez hale gelebilir. Bu durumun önüne geçmek için "Zero Padding" adlı yöntem kullanılmaktadır [7]. Zero Padding orijinal girdi boyutunu korumamızı sağlayan bir tekniktir. Zero Padding'de girdinin kenarlarına içindeki değerler 0 olan bir piksel matrisi ekleyerek orijinal resim girdisinin boyutu korunur. Bu sayede boyut küçülürken önemli olabilecek değerleri kaybetmek yerine sadece 0'lardan oluşan bu matrisler kaybolmuş olur.

Evrişimli sinir ağları'nda kullanılan bir diğer katman (Filtre) ise "Max Pooling"dir [8]. Max pooling Evrişimli Sinir Ağları'nda genelde Evrişimli katmanların sonuna eklenir. Max pooling çalışma mantığı anlatılırken, aşağıdaki 6 nuramalı şekilden yararlanılacaktır. Max pooling işleminde önce n x n boyutunda Max Pooling yaparken kullanılacak filtrenin boyutuna karşılık gelen bir matris tanımlarız. Max Pooling şekil 6 üzerinden anlatılacağı için burada n = 2'dir. Sonra bu filtrenin atlama (Stride) miktarı belirtilir. Bu atlama miktarı ile filtremizin resim üzerinde kayarken kaç pixel boyu hareket edeceği belirtilir. Sonra boyutlarını 2 x 2 olarak belirlediğimiz filtremiz evrişimli katmandan çıktı olarak gelen 4 x 4 matrisin ilk 2 x 2'lik alanındaki değerler içinden en büyük değere sahip olan değeri kaydeder. Sonra 2 x 2 Max Pooling filtremiz daha önce belirttiğimiz kayma (Stride) miktarı kadar piksel yana ve sonrasında aşağıya kayarak 4 x 4'lük matristeki her bir 2 x 2'lik alandaki maksimum değerleri kaydeder. Max Pooling yapmaktaki amaç girdi boyutunu en önemli pikselleri koruyarak küçültmek ve bu küçülme sayesinde girdinin soyut bir formunu sisteme vererek "Over-Fitting" (ezberleme)'in önüne geçmektir. Max Pooling aynı zamanda girdideki parametre sayısını azaltarak sistemin yapması gereken işlem miktarını da düşürür.



Şekil 6: *Max Pooling katmanının işleyiş mantığını temsili bir örnek [7].*

Bir Evrişimli sinir ağı modelinde öğrenilebilir parametreleri veren denklem şu şekildedir:

$$\underline{\text{Öğrenilebilir Parametreler}} = \text{Girdi Sınıf Ağırlıkları x Çıktı Sınıf Ağırlıkları} + \text{Eğilimler} \quad (2)$$

Bu denklemdeki eğilimler (Bias) önceden hesaplanmış ağırlıklı toplam değeri alıcı nöronun aktivasyon fonksiyonuna verilmeden önce ağırlıklı toplam değerine eklenir. Böylece eğilim (Bias) kullanılarak daha önce eşik değerinin (Threshold) altında kaldığı için ateşlenmemiş bir nöronu ateşleyebilir ya da daha önce ateşlenmiş bir nöronun ateşlenmemesini sağlayabiliriz. Sisteme eğilim (Bias) değerlerinin eklenmesi modelin veriyi öğrenirken ki esnekliğini arttırır.

Denklem (2)'ye geri dönecek olursak eğer bir önceki katman dense layer ise Evrişimli katmana gelen girdi (Input) sayısı önceki katmandaki düğüm (Node) sayısına eşittir. Eğer bir önceki katman Evrişimli ise Evrişimli katmana gelen girdi (Input) sayısı bir önceki Evrişimli katmanda bulunan filtre sayısına eşittir.

Bir Evrişimli katmandan çıkan çıktı (Output) sayısını denkleme dökecek olursak:

$$\underline{\text{Evrişimli Katmanın Çıktı Sayısı}} = (\text{Katmanda Kullanılan Filtre Sayısı}) \times (\text{Kullanılan Filtrelerin Boyutu}) \quad (3)$$

Denkleminden yola çıkılarak, katmandaki eğilim (Bias) sayısı incelenecek olursa, o katmandaki filtre sayısına eşit olduğu gözlemlenir.

Böylece, CNN'lerin genel çalışma prensiplerini ve algoritmanın arkasındaki genel mantığı incelemiş olduk, bunun devamında artık literatürde popüler olmuş bazı CNN mimarilerinin yapısını ve ne için kullanıldıklarını inceleyebiliriz.

2.3) Gerçek Zamanlı Sınıflandırma İçin Kullanılan Teknolojiler

Bir görüntüdeki nesnenin ve sınırlarının tahmini, nesne belirlemesidir. Nesne sınıflandırma ve nesne belirlenmesi arasındaki en büyük fark; nesne belirlemede nesneyi tanımlamaya çalışır, bunu yapmak için bir sınırlayıcı kutu kullanır.

2.3.1) Cisimlerin Belirlenmesi (Object Localization)

Bir görüntüdeki nesnenin ve sınırlarının tahmini, nesne belirlemesidir. Nesne sınıflandırma ve nesne belirlenmesi arasındaki en büyük fark; nesne belirlemede nesneyi tanımlamaya çalışır, bunu yapmak için bir sınırlayıcı kutu kullanır.

2.3.2) Cisimlerin Sınıflandırılması (Object Classification)

Cisimlerin sınıflandırılması fonksyonu ise görseller içerisinde yerleri belirlenerek etraflarında sınır çizgileri çizilen cisimlerin önceden belirlenen sınıflara göre ayrılmasıdır. Bizim durumumuzda bu, kağıt, karton, cam, metal ve plastik geri döşüm sınıflarının tespit edilmesidir.

2.3.3) MobileNet Gerçek Zamanlı Cisim Sınıflandırma&Belirleme Ağı

Proje için kullanılan nesne tanımlama algoritması "Tensorflow Object Detection Zoo Respository" den seçilmiştir. Seçilen model önceden "COCO" veri seti üzerinde eğitilmiş olup, günlük yaşamda bol bulunan cisimlerin tanınması üzerine eğitilmiştir. Kullanılan modelin tam ismi ise, ssd_mobilenet_v1_fpn_coco'dır.

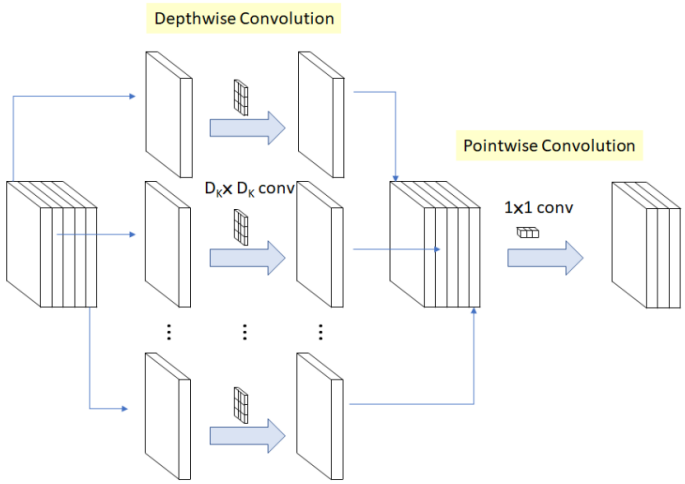
Projede kullanılan veri seti ise 3 farklı kaynaktan elde edilmiştir ve bir kısmı ise bizim tarafımızdan üretilmiştir. İlk olarak 2/5'i Kaggle.com sitesinde yer alan "Garbage Classification" adlı veri setinden oluşmaktadır. 1/5'lik kısmı Google Images'dan geri kalan 2/5'lik kısmı ise bizim tarafımızdan elde edilmiştir. Görsellerin etiketlenmesi de bizim tarafımızdan yapılmıştır.

Eğitim süreci tamamlandığında ise, modelimizi düşük işlemcili cihazlarda çalıştırabilmek için daha az yük gerektiren bir modele çevirip, bir android app'ine döşedik.

Kullanılan nesne tanımlama algoritmasının detaylarına daha yakından bakacak olursak. İlk Google tarafından tasarlanan MobilenetV1, model boyutlarını azaltmak ve işlem süresini minimize etmek için kullanılmaktadır. Bundan dolayı, özellikle mobil ve gömülü görüş uygulamaları için kullanışlıdır.

Diğer obje tanıma algoritmalarına kıyasla, küçük mimari yapısından dolayı daha az parametre üretmektedir, bu da karar aşamasının daha az işlemle, hızlı bir şekilde gerçekleşebilmesine olanak sağlamaktadır.

Derinlemesine ayrılabilir evrişimler MobileNet yapısının en önemli özelliklerindendir. Çalışma prensibi aşağıda şekilde de görülebilir:



Şekil 7: *Derinlemesine Ayrılabilir Evrişim Katmanı Çalışma Prensibi [5].*

1. Derinlemesine evrişim kanal yönünden DK × DK uzamsal evrişimdir. Yukarıdaki şekil üzerinden örnek verirsek, 5 kanalımız var, o zaman 5 DK × DK uzamsal evrişimimiz olacak.

2. Noktasal evrişim, gerçekte boyutu değiştirmek için 1 × 1 evrişimdir.

Bu bahsedilen işlemin maliyeti ise şöyle özetlenebilir:

$$D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F$$

Burada M: Giriş kanalı sayısı, N: Çıkış kanalı sayısı, DK: Çekirdek boyutu ve DF: Özellik harita boyutudur.

Normal evrişimli katmanlarda bu maliyet şu şekildedir:

$$D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F$$

Dolayısıyla işlem maliyetindeki düşüş bu şekilde özetlenebilir:

$$\frac{D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F}{D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F} = \frac{1}{N} + \frac{1}{D_K^2}$$

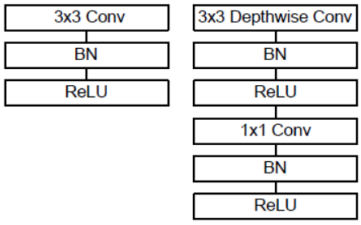
DK × DK 3 × 3 olduğunda, 8 ila 9 kat daha az hesaplama yapılabilir, ve toplam doğrulukta sadece küçük bir azalma oluşur.

MobileNet'in genel mimarisine bakılacak olursa ise:

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Şekil 8: *MobileNet Mimari Açılımı [5].*

Aynı zamanda, görüldüğü üzere her evrişimden sonra Batch Normalization ve ReLU uygyulandığı görülür:



Şekil 3: *Standart Konvolüsyon (Sol), BN ve ReLU ile Derinlemesine Ayrılabilir Konvolüsyon (Sağ) [5].*

Ek olarak, MobileNet'te giriş katmanının kontrolü için genişlik çarpanı "a" kullanılmıştır. Bu da M'nin aM olmasını sağlar. Ve derinlemesine ayrılabilir evrişim maliyeti:

$$D_K \cdot D_K \cdot \alpha M \cdot D_F \cdot D_F + \alpha M \cdot \alpha N \cdot D_F \cdot D_F$$

Burada α, 1, 0.75, 0.5 ve 0.25 tipik ayarlarıyla 0 ila 1 arasındadır. A = 1 olduğunda, temel MobileNet'tir. Ve hesaplama maliyeti ve parametre sayısı kabaca α^2 ile kuadratik olarak azaltılabilir. Son olarakta, girdi görsellerinin çözünürlüklerini kontrol etmek için, çözünürlük çarpanı adı verilen "p" kullanılır:

$$D_K \cdot D_K \cdot \alpha M \cdot \rho D_F \cdot \rho D_F + \alpha M \cdot \alpha N \cdot \rho D_F \cdot \rho D_F$$

Burada ρ, 0 ile 1 arasındadır. Ve giriş çözünürlüğü 224, 192, 160 ve 128'dir. ρ = 1 MobileNet'te standart değer olarak kabul edilir.

2.4) Test için Kullanılan Teknolojiler

Nesneleri gerçek zamanlı olarak tanıması için oluşturduğumuz modelimizi test edebilmek için modelimizi mobil cihazlarda kullanılabilir hale getirdik. Modelimizin test edilmesi için mobil cihazları seçmemizin nedeni mobil cihazların test aşamasında sağladıkları kolaylıktan yararlanmaktır. Modelimizi mobil cihazlarda çalışabilir hale getirmek için Android işletim sistemine sahip telefonlar için uygulama oluşturmayı sağlayan Android Studio IDE (Integrated Development Environment) sini ve Android studio'da uygulamanın kodlanması aşamasında Java yazılım dilinden yararlandık. Aynı zamanda son ürünün oluşturulması aşamasında makine öğrenmesi modellerinin mobil ortamda kullanılabilmesini sağlayan TensorFlow Lite tarafından sağlanan framework den de yararlanıldı.

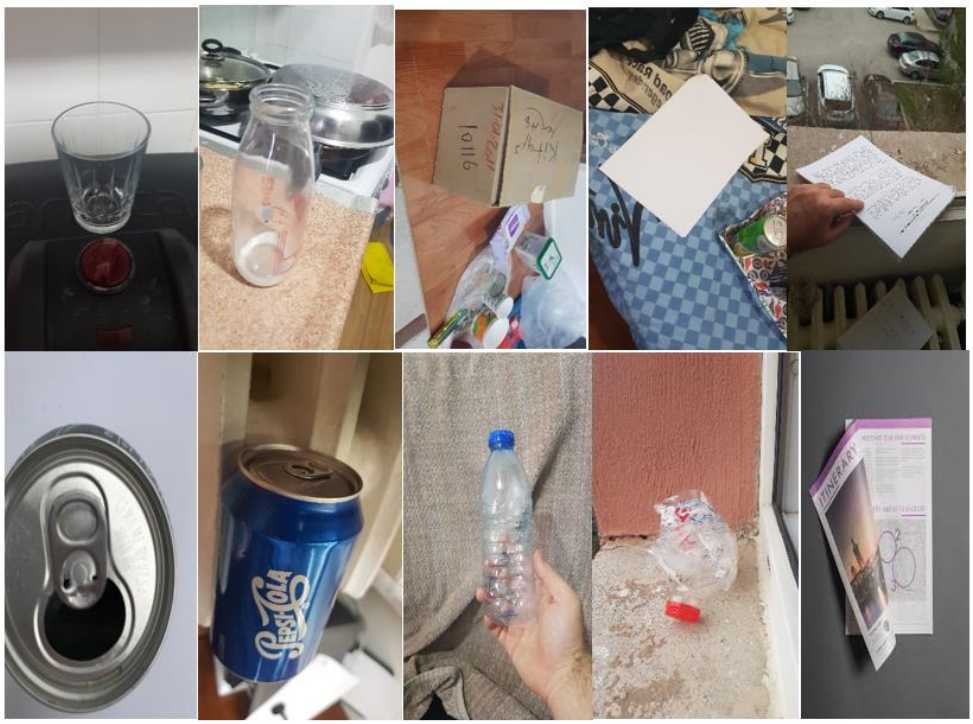
3. ÇALIŞMADA KULLANILAN VERİ SETİ

Projede kullanılan veri seti ise 3 farklı kaynaktan elde edilmiştir ve bir kısmı ise bizim tarafımızdan üretilmiştir. İlk olarak 2/5'i Kaggle.com sitesinde yer alan "Garbage Classification" adlı veri setinden oluşmaktadır. 1/5'lik kısmı Google Images'dan geri kalan 2/5'lik kısmı ise bizim tarafımızdan elde edilmiştir.

Görsellerin etiketlenmesi de bizim tarafımızdan yapılmıştır. Görsel etiketlenme sürecinde LabelImg adı verilen bir yazılım kullanılmıştır. Lokalizasyon etiketleri dikdörtgen şekillerinde cisimlerin sınırlarına uyumlu bir şekilde elle çizilmiştir. Etiketleme sonucunda ise her görsel için bir XML etiket dosyası elde edilmiştir. Bu dosyalarda cisimlerin yerlerini belirleyen koordinatlarla birlikte sınıfları da bulunmaktadır.

Modelimiz, veri setine bağlı olarak 7 sınıfı ayrıştıracak şekilde kurulmuştur. Bu sınıflar; renkli cam, cam, kağıt, renkli kağıt, plastik, metal ve kutu kağıdı (cardboard) olacak şekilde ayrılmıştır. Her sınıf için 205 eğitim görseli bulunmaktadır. Toplamda 1425 eğitim, 238 adet ise test görseli bulunmaktadır.

Görseller seçilirken ve bizim oluşturduğumuz veri setinde, farklı açı ve ışık seçeneklerinde görüntü bulunmasına dikkat edilmiştir. Aşağıda veri setine ait örnek görseller bulabilirsiniz.



Şekil 9: *Veri Setinden Örnek Görüntüler*.

4. GERÇEKLEŞTİRİLEN ÇALIŞMA

Gerçekleştirilen çalışma yaklaşık olarak 4 hafta süren, tam zamanlı araştırma ve deneme yanılma gerektiren bir süreçle gerçekleştirildi. Üç farklı kez veri seti üzerinde modifikasyon, iki farklı kez ise model mimarisi değiştirilmesine gerek duyuldu. Buna rağmen sonuçta, beklenilen seviyede bir başarıya ulaşıldığı söylenebilir. Yapılan çalışmanın aşamalarına bir liste halinde bakacak olursak;

- **Problemin tanımlanması (1 iş günü)**
  "Projemiz "object detection" algoritmaları yardımıyla ortak alanlarda bulunan küçük çaplı çöp kutularına atılan cisimleri geri dönüşüm sınıfları çerçevesinde, düşük işlem maliyetiyle (TensorflowLite) sınıflandırır."

- **Literatür çalışması (4 iş günü)**
  Benzer çalışmalardan ulaşılan başarı oranları ve kullandıkları yöntemler hakkında bilgi edinilmiştir.

- **Problemin çözümüne uygun niteliklere sahip görsellerin temin edilmesi (3 iş günü)**
  Probleme uygun niteliklere sahip görsellerin büyük kısmını kendimiz oluşturduk, bir kısmını 'Trashnet' veri setinden aldı, yaklaşık üçte birini ise Google görseller bölümüncen aldık.

- **Görsellerin lokalizasyon algoritması için etiketlenmesi (5 iş günü)**
  Object detection algoritmasının sınıflandıracağı cisimleri görsellerde belirleyebilmesi için kullanılan eğitim görsellerinde sınıflandırılacak cisimlerin koordinatları belirlendi. LabelImg yazılımı kullanıldı.

- **Kullanılacak araç ve yöntemlerin belirlenmesi ve kurulması (2 iş günü)**
  Genel olarak, model kurumu için ssd_mobilenet modelini, tensorflow Object Detection API aracılığıyla eğitim yapıldı ve sonrasında Android Studio kullanılarak bir mobil test ortamı gerçekleştirildi.

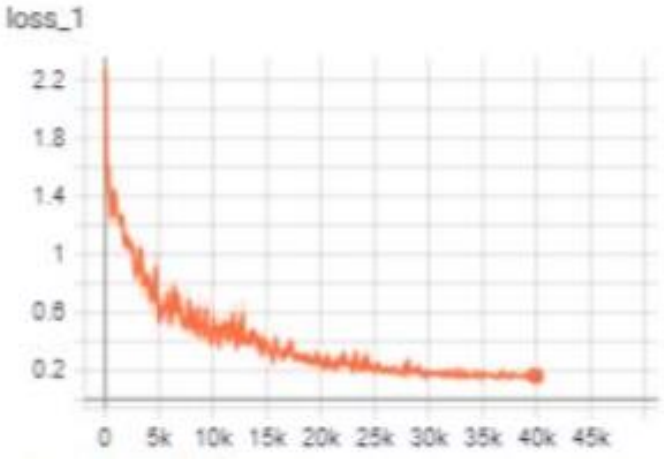
- **Optimal modele ulaşılması (16 iş günü)**
  İstediğimiz oranlarda başarıya ulaşabilmemiz için farklı verisetleriyle ve mimarilerle 3 model eğitmemiz gerekli oldu, ve son model 2 gün boyunca eğitildi. Veri seti üzerindeki oynamalar bu süreci uzattı. İlk olarak sadece Kaggle.com dan elde edilen veri seti kullanılarak düşük başarılı bir model elde edildi. Sonrasında Google Images'dan temin edilen görsellerle güçlendirilmiş bir model oluşturuldu.Fakat beklenen başarı oranı üçüncü denemede kendimizin oluşturduğu veri setinin eklenmesiyle elde edildi.

- **Elde edilen modelin işlem gücü düşük cihazlarda testi için TFlite formatına çevrilmesi (1 iş günü)**
  Modelimizi bir gpu'ya sahip bilgisayarda test ederek çalıştığını gördük, süremizin geri kalan kısmında ise düşük işlem gücüne sahip cihazlarda test yapabilmemiz için modelimizi hafiflettik.

- **Mobile testler için modelin testine uygun bir Android app'i oluşturulması (6 iş günü)**
  Bunu aşamanın amacı, modelin olası entegrasyonu durumunda gerçekçi maliyetlerde, düşük işlem gücüne sahip cihazlarla bir altyapı kurulabileceğini kanıtlamaktır.

4.1. Elde Edilen Çıktılar

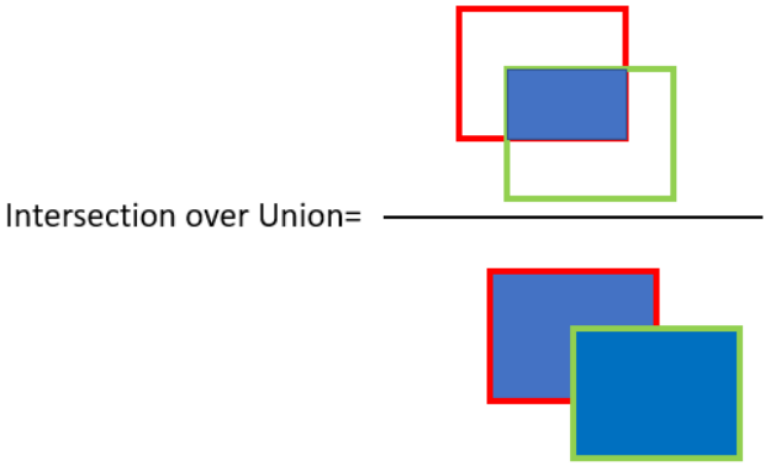
Daha önceden, COCO adı verilen günlük sık rastlanan objeleri içeren bir veri seti üzerinde eğitilmiş olan Mobilenet mimarisindeki ağımızı elimizdeki veri setine göre eğitmemiz yaklaşık olarak 2 gün sürmüştür. Eğitim sürecinde yaklaşık olarak 40 bin epoch döngüsü elde edilmiştir. Eğitim kriterleri arasında en çok dikkat edilmiş kriter Loss oranlarının doğrusal bir izleyiş seyredene kadar eğitilmesidir. Aşağıdaki grafikte de Loss oranın 40 epoch boyunca seyrini görebiliriz.



Şekil 10: *Eğitim Sürecindeki Loss Oranının Seyiri.*

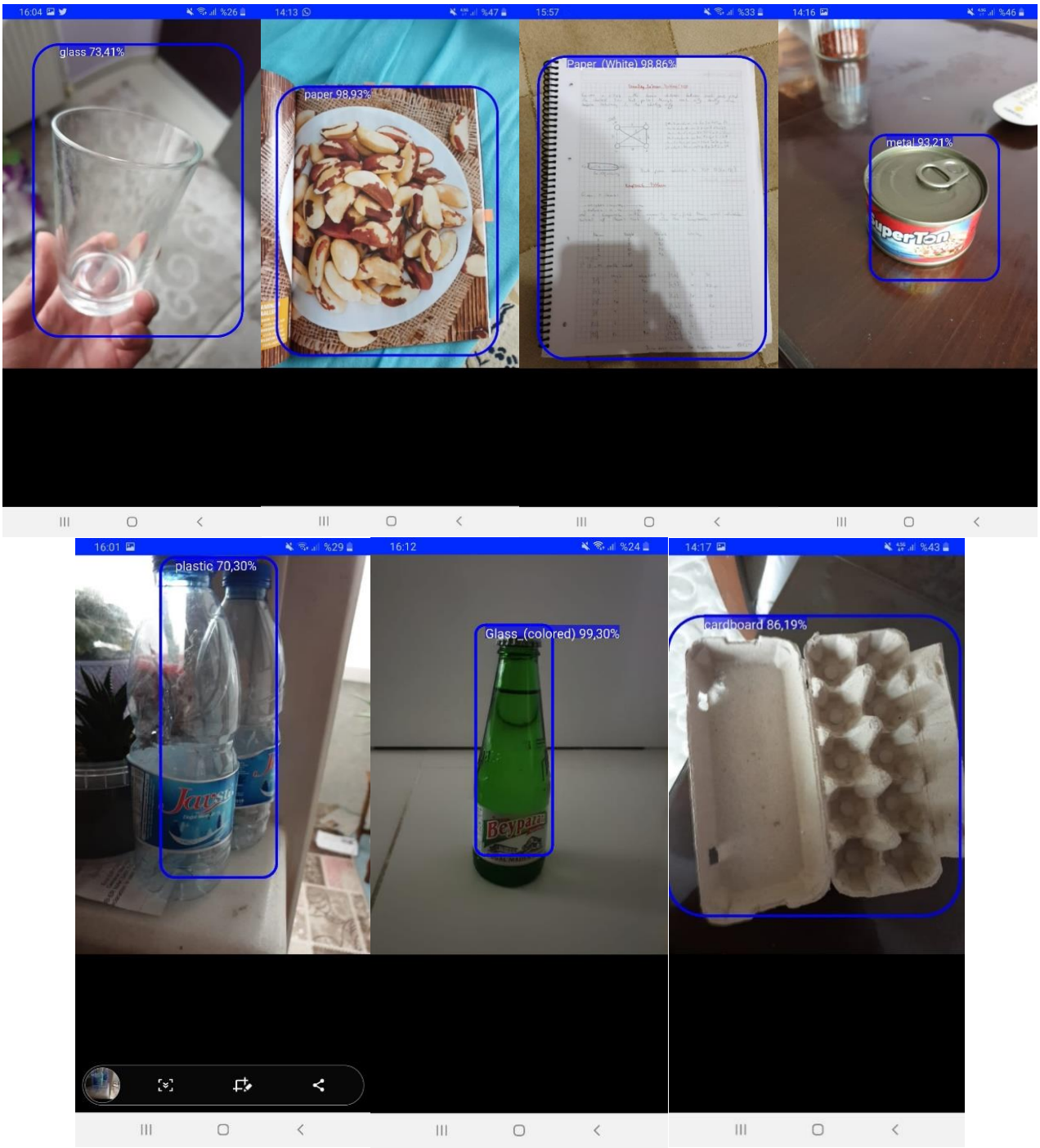
Yukarıda görüleceği üzere, eğitim sürecinde dikkat edilen diğer bir nokta ise Loss oranının 0.5'in altına inmiş olmasıdır. Kısacası, eğitim sürecinin bitirilmesi için dikkat edilen iki parametre kayıp oranının 0.5 altına inmesi ve uzun bir süre için doğrusal bir eğilim izlemesi.

Kurulan modelde dikkat edilen noktalardan bir tanesi de intersection over union oranının 0.5 yerine 0.60 değerinde tutulmasıdır. Bu oran karton kutu ve kâğıt gibi bazı benzer nitelikler içeren sınıfların karışık değer döndürmesini önlemiştir. Ayrıca eğitim sürecinde tahmin edilen değer sınırlarının gerçek değer sınırlarıyla %60 uyuşmasını ön görmüştür. İntersection over union parametresinin çalışma prensibi aşağıdaki görselde görülebilir.



Şekil 11: *Modelimizde 0.6 değerinde tutulan ıntersection over union parametresinin çalışma mantığı* [14].

Aşağıda da eğitim sonrası modelin düşük işlem gücündeki testi amacıyla kurulan Android uygulamasından çıktılar görülmektedir. Aşağıdaki görsellerde her geri dönüşüm sınıfı temsil edilmiştir. Sınıflandırılacak olan cisimlerin lokalizasyon işlemiyle ilk olarak yerlerinin belirlendiği sonrasında ise sınıflandırıldığı da görülebilmektedir.



Şekil 12: *Düşük İşlem Gücüne Sahip Cihazlarda Test Amaçlı Oluşturulan Android Uygulamasından Görüntüler.*

4.2. Başarısız Denemeler

Yukarıda görülen son modele ulaşılmadan önce 2 farklı model elde edilmiştir. Fakat, istenilen başarı oranına ulaşılamadığı için denemelere devam edilmek zorunda kalınmıştır.

İlk denemede, sadece Trashnet adı verilen veri setinden alınan görüntüler etiketlenerek ssd_mobilenet_v2_coco ağ mimarisiyle sonuç elde edilmeye çalışılmıştır, fakat istenilen başarı oranına ulaşılamamıştır.

İkinci denemde veri setine Google Images'dan eklemeler yapılarak veri setinin boyu iki katına çıkartılmıştır fakat, tekrardan istenilen başarıya ulaşılamamıştır.

Ek olarak, ilk iki denemeden sonra geri dönüşüm sınıfları da kendi içlerinde ayrıştırılarak daha başarılı sonuçlara varıldığı görülmüştür. Spesifik olarak, algoritmamız renk ve şekilleri baz alarak belirli bir sınıfın model içindeki ağılıklarını ayarladığından dolayı, kağıt ve renkli kağıt, cam ve renkli cam gibi bazı sınıflar kendi içinde de ayrılmıştır.

Son olarakta, model mimarisi değiştirilmiştir. Ssd_mobilenet_v1_fpn_coco mimari yapı olarak seçilerek işlem yapılmıştır. Ayrıca bu son denemede veri setinin boyutu tekrardan iki katına çıkarılmıştır. Bu sefer veri setine eklenen görseller bizim tarafımızdan elde edilmiştir. Farklı ışık ve açı olanaklarında görseller edinilmeye dikkat edilmiştir. Böylece istenen başarı oranına ulaşabilmiş bir model elde edilmiştir.

5. SONUÇ ve DEĞERLENDİRME

Bu çalışmada gerçek zamanlı nesne tanıma problemine Evrişimsel Sinir Ağlarıyla çözüm aranmıştır. Yaptığımız çalışmada kağıt, karton, cam, metal ve plastik yapıdaki atıkların hızlı ve doğru bir şekilde tanınması ve bu sayede bu atıkların ayrıştırılması aşamasında görüntü işlemeye dayalı bir çözüm üretmeye çalıştık. Bu amaçla geliştirdiğimiz modelimizin eğitim ve test aşamalarında bir kısmını kendimiz oluşturduğumuz, bir kısmını Google görseller ve "Garbage Classification" adlı veri setini kullanarak oluşturduğumuz verilerden yararlanılmıştır. Farklı modeller ve veri seti kombinasyonları sonucunda, MobileNet ağ mimarisine sahip ve TrashNet, Google Images ve kendimizin oluşturduğu görsellerin bir arada kullanıldığı denemede, başarılı bir şekilde atıklar 7 farklı sınıfa ayrılabilmiştir. Ayrıca, bazı atık sınıfları da kendi içinde alt sınıflara ayrıştırılarak son denemede daha yüksek başarı oranları elde edilebilmiştir. Ek olarak bu tür benzer niteliklere sahip sınıflar içeren problemlerde (karton kutu ve kağıt), "intersection over union" değerinin 0.5'in üzerine çıkarılmasının model başarısı için faydalı olabileceği tespit edilmiştir.

Ek olarak, modelimizin test sonuçlarından görülmüştür ki, modelimiz nesne tanıma modellerindeki sıkça karşılaşılan problemlerden birisi olan "false positive" sonuç üretmektedir. Modelin karşılaştığı bu sorunun önüne geçebilmek için gelecek çalışmalarda model eğitiminde kullanılan veri sayısının arttırılması ve modelin eğitimi için veri oluşturma aşamasında çok sayıda farklı arka plan kullanılarak modelin arka plan gürültülerini ayırt edebilmeyi öğrenebilmesi sağlanabilir.

6. KAZANIMLAR

Bu proje üzerinde çalışırken nesne tanıma metotları ve bu metotların mobil platformlarda nasıl kullanılabileceği hakkında bilgi sahibi olduk. Modelin geliştirilmesi ve mobil ortama aktarılması aşamasında karşılaştığımız sorunları çözebilmek için yaptığımız araştırmalar sayesinde nesne tanıma modellerinin karşı karşıya kalmaları muhtemel sorunlar ve bu sorunlarla ne şekilde baş edilebileceğine dair bilgi edindik. Mobil cihazların bilgisayarlara kıyasla çok daha az işlem gücüne sahip olmasından dolayı doğan kısıtlamaları ortadan kaldırmak için mobil cihazlarda makine öğrenmesi modellerini çalıştırabilmek için tasarlanmış olan TensorFlow Lite teknolojisi hakkında da bilgi sahibi olduk.

Bu proje nesne tanıma modeli geliştirme ve bu modeli mobil ortama aktarma konusunda edindiğimiz bilgileri pratiğe dökmemizi sağladığı.

7. KAYNAKÇA

[1] Talwar, Anish and Yogesh Kumar. “Machine Learning: An artificial intelligence methodology.” (2013).

[2] İnternet kaynağı: https://hackernoon.com/artificial-intelligence-vs-machine-learning-whats-the-difference-9e35u30a0 (Son Erişim: 23.09 09/05/2020)

[3] Choi, Hyungeun & Ryu, Seunghyoung & Kim, Hongseok. (2018). Short-Term Load Forecasting based on ResNet and LSTM. 1-6. 10.1109/SmartGridComm.2018.8587554.

[4] Borysiuk, Zbigniew & Konieczny, Mariusz & Kręcisz, Krzysztof & Pakosz, Paweł. (2018). Application of sEMG and Posturography as Tools in the Analysis of Biosignals of Aging Process of Subjects in the Post-production Age. 10.1007/978-3-319-75025-5_3.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in Advances in neural information processing systems, pp. 1097-1105, 2012.

[6] Choi, Hyungeun & Ryu, Seunghyoung & Kim, Hongseok. (2018). Short-Term Load Forecasting based on ResNet and LSTM. 1-6. 10.1109/SmartGridComm.2018.8587554.

[7] İnternet kaynağı: https://towardsdatascience.com/vgg-neural-networks-the-next-step-after-alexnet-3f91fa9ffe2c (Son Erişim: 23.09 09/05/2020)

[8] C. Szegedy et al., "Going deeper with convolutions," 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1-9, 2015

[9] Choi, Hyungeun & Ryu, Seunghyoung & Kim, Hongseok. (2018). Short-Term Load Forecasting based on ResNet and LSTM. 1-6. 10.1109/SmartGridComm.2018.8587554.

[10] M. Toğaçar And B. Ergen, "Deep Learnıng Approach For Classıfıcatıon Of Breast Cancer," 2018 International Conference On Artificial Intelligence And Data Processing (Idap), Malatya, Turkey, 2018, Pp. 1-5, Doi: 10.1109/Idap.2018.8620802.

[11] Boukaye Boubacar Traoré, Bernard Kamsu-Foguem, Fana Tangara. Deep convolution neural network for image recognition. *Ecological Informatics*, Elsevier, 2018, 48, pp.257-268.

[12] Dong Y, Jiang Z, Shen H, David Pan W, Williams LA, Reddy VVB, Benjamin WH, Bryan AW. 2017. Evaluations of deep convolutional neural networks for automatic identification of malaria infected cells. In: 2017 IEEE EMBS international conference on biomedical and health informatics, BHI 2017, 101-104.

[13] Kaya, A., Keçeli, A. S., & Can, A. B., Examination of various classification strategies in classification of lung nodule characteristics, Journal of the Faculty of Engineering and Architecture of Gazi University, 34(2), 709-725, 2019.

[14] Yildiz, O., Melanoma detection from dermoscopy images with deep learning methods: A comprehensive study, Journal of the Faculty of Engineering and Architecture of Gazi University, 34(4), 2241-2260, 2019.

[15] Deep Neural Networks for Object Detection (2013) Christian Szegedy Alexander Toshev Dumitru Erhan

[16] Efficient Mobile Implementation of A CNN-based Object Recognition System (2016**)** Keiji Yanai Ryosuke Tanno Koichi Okamoto

[17] Real-Time Object Recognition and Orientation Estimation Using an Event-Based Camera and CNN (2014) Rohan Ghosh, Abhishek Mishra, Garrick Orchard, and Nitish V. Thakor

[18] You Only Look Once: Unified, Real-Time Object Detection (2015) Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi

[19] Selective Search for Object Recognition (2012) J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, A. W. M. Smeulders

[20] SpotGarbage: Smartphone App to Detect Garbage Using Deep Learning (2016) M. Gaurav, Y. Kaushal, G. Mohit ve K. Narayanan

[21] Fast R-CNN (2015) G. Ross

[22] Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition (2015) H. Kaiming, Z. Xiangyu, R Shaoqing ve S. Jian

[23] Wanli Ouyang, et al., "Deepid-net: multi-stage and deformable deep convolutional neural networks for object detection", Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition, 20