# AKILLI TELEFONLAR İLE NESNE TESPİTİ

Cumali TOPRAK

Mühendislik Fakültesi, Bilgisayar Mühendisliği Kocaeli Üniversitesi Kocaeli, Türkiye cumalitoprakk@gmail.com

Özetçe— Bu projede akıllı telefonlar kullanılarak nesne tespiti yapmak amaclanmıştır. Bu kapsamda mobil programlama teknikleri, bulut veritabani kullanımı, ve sunucuda uygulama barindirma gibi konularda bilgi sahibi olunmasi amaçlanmıştır. Projeyi özetlersek telefondan çekilen bir görüntünün veya telefonun galerisinden seçilen bir görüntünün react native uygulaması kullanılarak sunucuya gönderilip burada görüntü işleme uygulamasıyla nesnelerin tespitinin yapılması ve verilerin burada barındırılması amaclanmıştır. Son olarak işlenen görüntünün de telefona tekrar geri gönderilmesi amaçlanmıştır. Projemizde React native mobil uygulama programlama framework'u, amazon uzak sunucusu ve depolama alani kullanılmıstır.

Anahtar Kelimeler—React native, Amazon, Goruntu Isleme, Nesne tespiti.

Abstract— In this project, it is aimed to detect objects using smart phones. In this context, it is aimed to have information about mobile programming techniques, cloud database usage, and application hosting on the server. If we summarize the project, it is aimed that an image taken from the phone or an image selected from the phone's gallery is sent to the server using the react native application, and the objects are detected with the image processing application and the data is stored here. Finally, it is aimed to send the processed image back to the phone. React native mobile application programming framework, amazon remote server and storage area were used in our project.

Keywords—React native, Amazon, Image Processing, Object Detection.

## I. GİRİS

Projemizde react native mobil programlama frameworku kullanmis bulunmaktayiz. Projemizi olustururken herhangi bir ucuncu parti uygulama kullanmadik(orn. Expo). Server tarafında ise bize en uygun olan barindirma ortaminin amazon olduguna karar verdik ve amazon hesabi actik. Back-End uygulamamiz olan nesne tanima uygulamamizi sunucuya tasidik. Ayni zamanda nesne depolama islemini de bu platformda gerceklestirdik. Projemizin detaylarina girecek olursak Goruntu isleme kisminda bircok alternatif teknolejiyi arastirdik. Fakat bircogu ya maliyetli ya da tam olarak bizim ihtiyaclarimizi karsilamiyordu. Biz bu yuzden oncelikle YOLOv4 algoritmasinin deep learning kutuphanesi olan keras ile kullanimini projemize uyguladik. Fakat sonradan algimiz sonuclarda bu algoritma goruntu isleme kismini sorunsuz yapmakta fakat 20 saniyelere varan islem suresiyle cok uzun zaman almaktaydi. Biz de bu amacla Farkli Berkay Efe ÖZCAN Mühendislik Fakültesi, Bilgisayar Mühendisliği Kocaeli Üniversitesi

> Kocaeli, Türkiye berkayefeozcan@gmail.com

alternatiflere yonelmeye karar verdik. YOLOv3 algoritmasinin keras kutuphanesine implementasyonunu goruntu isleme uygulamamiz icin kullanmaya karar verdik. Bu algortmanin islem suresi cok daha kisa surede (yaklasik 1.sn) aliyordu. Tek dezavantaji olan sadece 80 nesne tanimasiydi. Bizim amacimiz tum nesneleri taniyan bir modeldi. Bunun icin arastirmalar yaptigimiz zaman gorduk ki bunun icin model egitmemiz gerekiyordu. Fakat model egitmek icin de elimizde cok fazla dataset olmasi gerekiyordu. Bunu elde etmek cok zor olmasinin yanisira bu datasetleri egitmek icin islem gucu cok yuksek bilgisayarlarin kullanilmasi gerekiyordu. Biz de bu sebepten dolayi bu algoritmayi kullanmaya karar verdik. Taniyacagi nesne sayisi sinirli olmasina ragmen uygulama o belirli nesneler uzerinden yuzde 95'i askin dogrulukla calisiyor. Bu algoritmayla eslesen nesnelerin konumunu bulmus olduk. Sirada bu nesneleri diktortgen bir kutu icerisinde gostermek gerekiyordu. Bunun icin arastirmalar yaptigimiz zaman gorduk ki ihtiyacimiz olan sey OpenCV kutuphanesiymis. Bu kutuphaneyi de kullanarak konumlarini daha once tespit ettigimiz nesneleri yesil bir dikdortgen kutu icerisine alip bu nesneyi yuzde kac dogru tanidigini belirten oran yaninda nesnenin ismini de yazdırdık. Aynı zamanda gonderilen resim uzerinde toplam kac nesnenin tanindigini da resmin sol ust kosesine yazdirdik. Bunu tekrar istemci olan mobil uygulamaya geri gonderdik. Projemizde istek yapmak icin ayrica rest api kullandik. Sunucu tarafında python kullandigimiz icin flask frameworkunu kullandik. Aslinda oncelikle bir diger python frameworku olan django kullanmistik, fakat sonra yaptigimiz arastirmalar da ogrendik ki flask frameworku kucuk olcekli ve orta olcekli uygulamalar icin daha verimli calisan bir frameworkmus biz de bu yuzden uygulamamizi flask framworke dondurduk. Uygulamamizin detaylarina daha sonraki bolumlerde girecegiz.

### II. TEMEL BİLGİLER

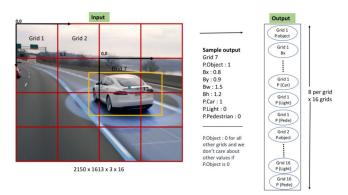
Projeye baslarken oncelikle arayuz kismini yapmaya basladik. Bu amacla arayuz kisminda resim gonderme islemi icin react native'de bulunan hazir bir modul kullandik. Bu modul yardimiyla hem galeriden fotolardan hem de anlik olarak cektigimiz bir resmi api uzerinden istekte bulunarak sunucuda calisan uygulamaya gonderebildik. Sunucu tarafinda YOLOv3 algoritmasi keras deep learning kutuphanesi ve bir goruntu isleme kutuphanesi olan OpenCV

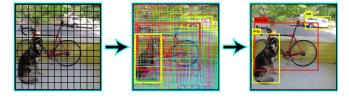
kullandik. Yolo object detection islemini sahip oldugu iyi algoritma sayesinde cok kisa surede gerceklestirebilmektedir. Sunucuyu amazonun 1 dolar karsiliginda free olarak ayirdigi paketlerden birini kullandik. Bu bizim icin cok kullanisli bir secim oldu. Sunucuyu olustururken ubuntu isletim sistemini de kurduk. Ubuntu sunucusunu secmemizin sebebi hem cok fazla forumda bununla ilgili bilginin bulunmasinin yanisira bizim de ubuntu isleti sistemi hakkinda bilgilimizin ve daha onceki tecrubelerimizin bulunmasidir. Daha sonra python diliyle calisacagimiz icin flask frameworku kuruldu ve uygulamamiz sunucuya tasindi.

## III. YÖNTEM

#### A. YOLOv3 Kullanimi

YOLO (You Only Look Once) görüntüdeki nesneleri konumlarıyla bereber tek seferde tespit edebilen bir yapay sinir ağıdır. Görüntüyü hücrelere bölerek her bir hücrede nesneler için confidence değerlerini ve bounding boxları tahmin etmeye çalışan çıktı vektörleri üretir.





Şekil-1

Buna göre YOLO her ızgara için ayrı bir tahmin vektörü oluşturur. Bunların her birinin içinde:

Güven skoru: Bu skor modelin geçerli ızgara içinde nesne bulunup bulunmadığından ne kadar emin olduğunu gösterir. (0 ise kesinlikle yok 1 ise kesinlikle var) Eğer nesne olduğunu düşünürse de bu nesnenin gerçekten o nesne olup olmadığından ve etrafındaki kutunun koordinatlarından ne kadar emin olduğunu gösterir.

Bx: Nesnenin orta noktasının x koordinatı

By: Nesnenin orta noktasının y koordinatı

Bw: Nesnenin genişliği

Bh: Nesnenin yüksekliği

**Bağlı Sınıf Olasılığı:** Modelimizde kaç farklı sınıf varsa o kadar sayıda tahmin değeri. Örn;

Yukarıdaki resimde Grid 7' ye baktığımızda eğer araba olduğundan kesin olarak eminse:

Araba: 1, Yaya: 0 olacaktır.

Devam etmeden önce güven skoru nasıl hesaplanıyor onu görelim:

Güven skoru = Kutu Güven Skoru x Bağlı Sınıf Olasılığı

Kutu Güven Skoru = P(nesne) . IoU

P(nesne) = Kutunun nesneyi kapsayıp kapsamadığının olasılığı. (Yani nesne var mı yok mu?)

**IoU** = Ground truth ile tahmin edilmiş kutu arasındaki IoU değeri

Yani aslında hiçbir nesne olmayan ızgaralarda bağlı sınıf olasılığı 0 olması gerektiği için (aslında arka plan olarak tespit ediliyor) güven skoru 0 olacaktır.

### B. OpenCV Kutüphanesinin Kullanimi

OpenCV (Open Source Computer Vision) açık kaynak kodlu görüntü işleme kütüphanesidir. 1999 yılında İntel tarafından geliştirilmeye başlanmış daha sonra Itseez, Willow, Nvidia, AMD, Google gibi şirket ve toplulukların desteği ile gelişim süreci devam etmektedir. İlk sürüm olan OpenCV alfa 2000 yılında piyasaya çıkmıştır. İlk etapta C programlama dili ile geliştirilmeye başlanmış ve daha sonra birçok algoritması C++ dili ile geliştirilmiştir. Open source yani açık kaynak kodlu bir kütüphanedir ve BSD lisansı ile altında geliştirilmektedir.

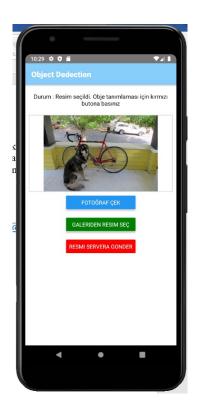
## C. React Native ile Mobil Uygulamanın Geliştirilmesi

Mobil uygulamada telefon galerisinden veya kameradan çekilen bir fotoğrafin bulut veri tabanına http post request ile gönderilmesi ve akabinde bulut bilgisayar gelen resmin üzerindeki nesnelerin tanımlaması yapılması amaçlanmıştır.

Bulut bilgisayara gönderilecek fotoğrafin elde edilmesinde React Native Image Pickerkütüphanesi kullanılmıştır. Elde edilen fotoğraf React Native Fetch Blob kütüphanesi yardımıyla bulut sunucuya gönderilmiştir. Elde edilen fotoğraf base64 tipine dönüştürülerek post request ile upload işlemi gerçekleştirilmiştir.



Şekil2: Uygulamanın başlangıç ekranı



Şekil3: Çekilmiş fotoğrafın gösterimi



Şekil4: İşlenmiş fotoğrafın gösterimi

## D. Bulut Veri Tabanı Üzerinde Görüntü İşlemenin Yapılması

Görüntü işleme sürecini bulutta çalıştırabilmek için bulut veri tabanları ilgili bir takım araştırmalarda bulunduk. Bizim için en iyi bulut sisteminin AWS olduğuna karar verdik. Bu bulut sunucu üzerinde çalışacak işletim sistemini Ubuntu Server 20.04 LTS olarak belirledik.

Bulut bilgisayara gelen resmi alabilmek için http kullandık. Bu http ile gelen requestleri handle etmek için python programlama dilinde Flask framework ünü kullandık. Post request ile gelen resimde yolov3 ile nesne tanımlaması yaptırdıktan sonra response olarak işlenmiş resim döndürülmüştür.

# IV. GERÇEKLEŞTİRİLEN TESTLER VE DEĞERLENDİRME

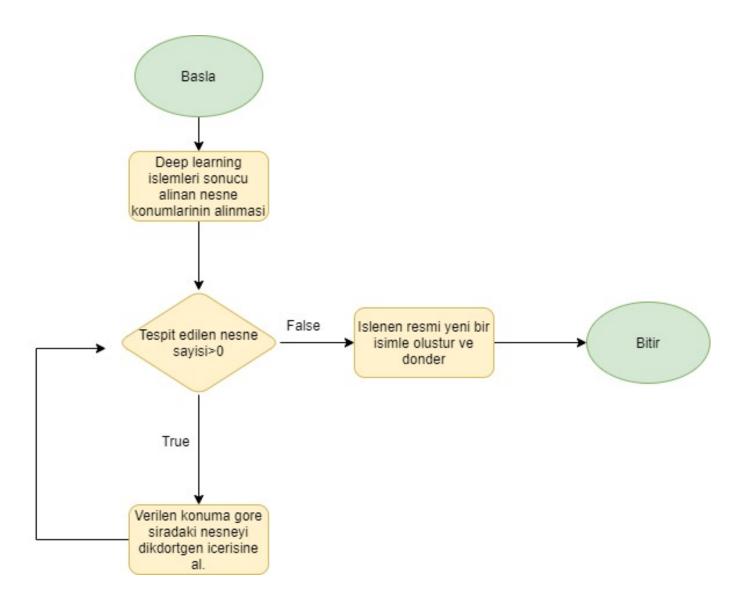
Gerçekleştirdiğimiz testler ve çözülen sorunlar sonucunda istemci-sunucu olarak gerçekleştirdiğimiz proje sorunsuz olarak çalışmaktadır. Programın doğru çalıştığından emin olmak için birden çok bilgisayarda birden çok kez çalıştırdık. Bu proje sonunda sunucu-client ve bulut kavramını anlamış, gerçek bir projeye uygulamış olduk.

# V. ÖZET VE SONUÇLAR

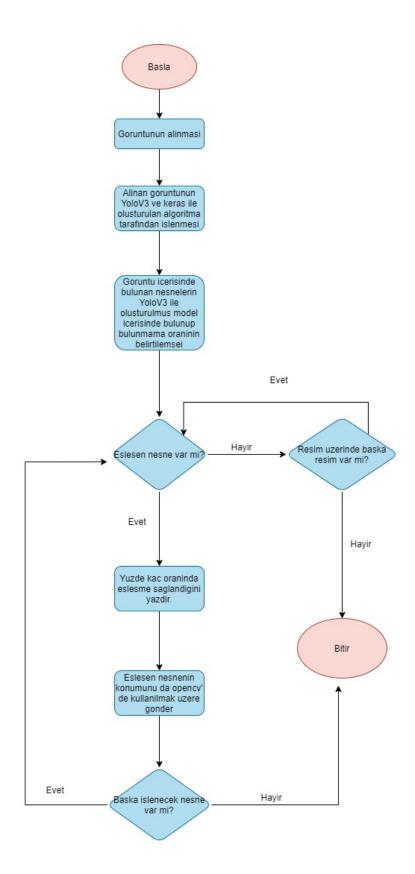
Yaptığımız bu proje sonucunda bizden istenilen tüm isterleri eksiksiz olarak uyguladık. Gerçekleştirilen testler sonucu da bunun hatasız olarak çalıştığını gördük. Bu projeyle birlikte istemci-sunucu kavramını daha iyi anlamış olduk. Bu proje an itibari ile gelişime açık durumdadır.

## VI. KAYNAKÇA

- [1] https://stackoverflow.com/
- [2]https://medium.com/@ahmetxgenc/pre-trained-modeller-kullanman
  - [3] https://www.youtube.com/
  - [4] https://www.udemy.com/
  - [5] https://github.com/theAIGuysCode/Object-Detection-API



Şekil5: Görüntü İşleme Akış Şeması



Şekil6: YoloV3 Algoritmasının Akış Şeması