Akıllı Telefonlar ile Nesne Tespiti

Furkan Aydoğan
Bilgisayar Mühendisliği, Kocaeli Üniversitesi
180202085

Sefa Berke Kara
Bilgisayar Mühendisliği, Kocaeli Üniversitesi
180202086

GİRİŞ

Bu uygulamanın amacı kullanıcı tarafından React Native ile geliştirilen uygulama üzerinden anlık görüntü alınarak bulut veritabanlarından birine gönderme,veritabanına gelen resim üzerinden nesne tanıma yapabilmek için görüntü işleme algoritmalarını kullanabilme ve veritabanı üzerinden tekrar uygulama ile haberleştirerek işlenmiş resmi uygulamaya gönderebilmektir

Özet

Akıllı telefon ile nesne tespiti uygulaması iki bölümden oluşmaktadır. Bunların birincisi Server kısmı ikincisi ise frontend kısmıdır. Server kısmında nesne tespiti ve işlenen resmin bulutta depolanması sağlanır. Frontend kısmı ise JavaScript React Native kütüphanesi ile yazılmış olup kullanıcı arayüzünde galeriden fotoğraf seçebilme, kameradan fotoğraf çekebilme ve de işlenmiş görüntünün gösterilmesi sağlanır.

Temel Bilgiler

Windows 10 işletim sistemine sahip bilgisayar ile geliştirme yapılmıştır. Program JavaScript dilinde React Native kütüphanesi ile geliştirilmiş olup, tümleşik geliştirme ortamı olarak “Visual Studio Code” kullanılmıştır. Uygulama android 7.0+ işletim sisteminde çalışmaktadır.

Tasarım

Akıllı telefon ile nesne tespiti projesi programlanma aşamaları altta belirtilen başlıklar altında açıklanmıştır.

Algoritma

Akıllı telefon ile nesne tespiti projesi App.js’e ek olarak “Camera.js, PicturePage.js, UploadImages.js, Style.js” modüllerinden oluşmaktadır.İlk olarak App.js’deki ana fonksiyon içerisinde resim url’lerini dinamik olarak tutulabilmesi ve görüntünün ekrana basılabilmesi için bir adet useState tanımlanmıştır. Altında return içerisinde bütün uygulamanın frontend iskeleti,arka plan resmi, işlenmiş görüntünün döndüreleceği alan, kamera butonu ve galeri butonu bu ana view etiketi içerisinde uygulamada bastırılır. View etiketinin içerisinde iki tane modül vardır. Bunlardan birincisi kameranın çağırıldığı kamera modülüdür. İkinicisi ise galeriden resim seçilebilmesi için kullanılan “PicturePage” modülüdür.Camera.js içerisinde “Camera” adlı bir adet buton bulunmaktadır. Bu butona tıklandığında fotoCek fonksiyonu aktif edilir. “fotoCek” fonksiyonun içerindeki “imagePicker” kütüphanesinden çağrılan “openCamera” metodu ile kamera açılır ve fotoğraf çekilmesi beklenir. Fotoğraf çekilmesi durumunda “UploadImages” fonksiyonuna çekilen resmin yolu dinamik olarak resmin yolunu tutan useState ve de kameradan çekildiğine dair “Camera” stringi döndürülür. “UploadImages” fonksiyonu girilmesiyle birlikte kameradan çekilen görüntünün uri’si “uploadUri” değişkenine eşitlenir. Bu url’nin dosya adını almak için “substring” ve “lastIndexOf” metodu ile dosya adı alınır ve “filename” değişkenine eşitlenir. Alınan bu dosya adı ile birlikte Firebase de “images-camera” klasörüne eklenir eğer bu klasör daha önceden oluşturulmamışsa da klasör otomatik olarak oluşturulur. Ardından gönderilen resmin link url alınır ve bundan sonraki aşama da “postData” fonksiyonu ile “herokuapp.com” sunucunda tutulan server’a bağlantı yapılarak yüklenen resmin dosya adı

kameradan mı yoksa galeriden mi seçildiğine dair bir string ve yüklenen resmin Firebase url'i JSON tipinde post metodu ile gönderilir. "Herokuapp.com" sunucusunda barındırılan server'a post atılması ile birlikte gelen istekteki veriler server da değişkenlerde tutulur. Bu değişkenler Tensorflow adlı fonksiyona parametre olarak verilir. Tensorflow fonksiyonunda ilk olarak Google Cloud Storage'dan erişim sağlanabilmesi için uygun link formatına dönüştürülür ve Google Cloud'a "APIkey.json" kimlik bilgileri ile bağlantı kurulur. Nesne algılama yapılabilmesi için "ObjectLocalization" metoduna yukarıda Google Cloud Storage için uygun formata dönüştürülen link verilerek nesne tespiti yaptırılır. Bu nesne tespitinin arka planında yapay zeka ve Tensorflow ile görüntü işleme bulunmaktadır. Buradaki nesne tespitinde şu tür algoritmalar kullanılmaktadır. Her bir görüntü BGR form dan Gri formuna dönüştürülür bunun yapılmasının nedeni kullanılan süreksizlik ve benzerlik bölütleme algoritmalarının gri renkli ile çalışabilmesidir. Bu algoritmalar *izolenokta*, *ince çizgi* veya *resim kenarları* gibi (gri seviye değerleri ani değişen) süreksizlikleri, düşük ve yüksek filtrelemedekine benzer maskeler kullanarak tespit edebilmeye dayanır. Bu maskelerin temelini türev işlemleri (1. ve 2. türev) oluşturur. Canny Kenar Tanımlaması ile resimdeki kenarlar belirlenir ve ardından Guassian Blur işlemi uygulanır.Kenarları belirlenen nesneler yapay zeka ile karşılaştırılarak en uygun objenin etiketi atanır.Sonuç olarakta Google Cloud Vision API'ndan bilgilerin olduğu bir obje döndürülür. Bu obje içinde koordinatlar ve etiketler "object" adlı bir değişkende tutulur. "Canvas" kütüphanesinden çağrılan "loadImage" fonksiyonuna Firebase Bulut Servisin'de yüklenen resmin url'si parametre olarak gönderili. Bu fonksiyon resmi indirir ve indirilen resim üzerinde Google Cloud Vision API tarafından döndürülen her bir tespit edilmiş nesneye ait koordinat değerleri ve o nesneye ait label(etiket) ismin resmin üzerine çizdirilir. Ve son olarak tespit edilen nesne sayısı resmin sol alt köşesine bastırılır. Çizdirme işlemleri bittikten sonra işlenmiş görüntü tekrardan Firebase'e depolanması için "basla" fonksiyonuna gönderilir. Ve içindeki "dosyaYukle" fonksiyonu ile Firebase'e yüklenmesi sağlanır. İşlem olumlu sonuçlandığında "Nesne Tespit Edildi" uyarısı bastırılır. İşlem bittiğine dair sunucudan React Native'deki post metoduna "response" iletilir. Hemen ardından Firebase'deki işlenmiş görüntü Front-end kısmında Firebase'den indirilerek kullanıcı ara yüzünde bastırılır. Kullanıcının galeriden resim seçmek istemesi durumunda "GaleridenSec" fonksiyonuna gidilir. Bu fonksiyon içerisinde "Image-crop-picker" kütüphanesinden "openPicker" metodu ile kullanıcının galeriden resim seçmesi beklenir. Resim seçilmesi durumunda kamera için yapılan bütün işlemler galeri içinde aynı şekilde tekrar edilerek nesne tespiti yapılır ve kullanıcı ara yüzüne bastırılır.

Sözde Kod

1- İlk olarak App.js'deki ana fonksiyon içerisinde resim url'lerini dinamik olarak tutulabilmesi ve görüntünün ekrana basılabilmesi için bir adet useState tanımla.
2- Return içerisinde bütün uygulamanın Front-end iskeleti,arka plan resmi, işlenmiş görüntünün döndüreleceği alan, kamera butonu ve galeri butonu bu ana view etiketi içerisinde uygulamada bastır.
3- View etiketinin içerisinde modüllerden camera modülüne giriş yap.
4- Camera.js içerisinde "Camera" adında bir buton oluştur.
5- Bu butona tıklandığında fotoCek fonksiyonu aktif et.
6- "fotoCek" fonksiyonun içerindeki "imagePicker" kütüphanesinden çağrılan "openCamera" metodu ile kamera aç.
7- Fotoğraf çekilmesi bekle.
8- Fotoğraf çekilmesi durumunda "UploadImages" fonksiyonuna çekilen resmin yolu dinamik olarak resmin yolunu tutan useState ve de kameradan çekildiğine dair "Camera" stringini döndür.
9- "UploadImages" fonksiyonu girilmesiyle birlikte kameradan çekilen görüntünün uri'si "uploadUri" değişkenine eşitle.

10- Bu url'nin dosya adını almak için "substring" ve "lastIndexOf" metodu ile dosya adı alınır ve "filename" değişkenine eşitle.
11- Alınan bu dosya adı ile birlikte Firebase de "images-camera" klasörüne ekle.
12- Eğer bu klasör daha önceden oluşturulmamışsa da klasör otomatik olarak oluştur.
13- Ardından gönderilen resmin link url al.
14- "postData" fonksiyonu ile "herokuapp.com" sunucunda tutulan server'a bağlantı yap.
15- Yüklenen resmin dosya adı kameradan mı yoksa galeriden mi seçildiğine dair bir string ve yüklenen resmin Firebase url'i JSON tipinde post metodu ile gönder.
16- . "Herokuapp.com" sunucusunda barındırılan server'a post atılması ile birlikte gelen istekteki veriler server da değişkenlerde tut.
17- Bu değişkenler Tensorflow adlı fonksiyona parametre olarak ver.
18- Tensorflow fonksiyonunda ilk olarak Google Cloud Storage'dan erişim sağlanabilmesi için uygun link formatına dönüştür.
19- Google Cloud'a "APIkey.json" kimlik bilgileri ile bağlantı kur.
20- .Nesne algılama yapılabilmesi için "ObjectLocalization" metoduna yukarıda Google Cloud Storage için uygun formata dönüştürülen link verilerek nesne tespiti yap.
21- Google Cloud Vision API'ndan döndürülen obje içindeki koordinatları ve etiketleri "object" adlı bir değişkende tut.
22- "Canvas" kütüphanesinden çağrılan "loadImage" fonksiyonuna Firebase Bulut Servisin'de yüklenen resmin url'si parametre olarak gönder.
23- Bu fonksiyon resmi indirmesini yap.
24- Ve indirilen resim üzerinde Google Cloud Vision API tarafından döndürülen her bir tespit edilmiş nesneye ait koordinat değerleri ve o nesneye ait label(etiket) ismin resmin üzerine çizdir.
25- Tespit edilen nesne sayısı resmin sol alt köşesine bastır.
26- Çizdirme işlemleri bittikten sonra işlenmiş görüntü tekrardan Firebase'e depolanması için "basla" fonksiyonuna gönder.
27- "dosyaYukle" fonksiyonu ile Firebase'e yüklenmesini sağla.
28- İşlem olumlu sonuçlandığında "Nesne Tespit Edildi" uyarısı bastır.
29- İşlem bittiğine dair sunucudan React Native'deki post metoduna "response" ilet.
30- Bu işlemden sonra Firebase'deki işlenmiş görüntü Front-end kısmında Firebase'den indirilerek kullanıcı ara yüzünde bastır.
31- Kullanıcının galeriden resim seçmek istemesi durumunda "GaleridenSec" fonksiyonuna git.
32- Bu fonksiyon içerisinde "Image-crop-picker" kütüphanesinden "openPicker" metodu ile kullanıcının galeriden resim seçmesi bekle.
33- Resim seçilmesi durumunda kamera için yapılan bütün işlemler galeri içinde aynı şekilde tekrar edilerek nesne tespiti yap ve kullanıcı ara yüzüne bastır.

Karşılaştığımız Sorunlar

React Native CLI çeşitli sürüm uyuşmazlık sorunları ile karşılaştık. Kurulumda yaşadığımız bu problemler yüzünden başlangıçta oldukça zaman kaybettik.Fakat tekrardan başa sararak ve temiz kurulum yaparak sorunu çözdük.

Firebase bulut ortamına resim kaydederken bir takım izin sorunları ile karşılaştık.

Gerekli hazır kütüphaneleri kullanırken şu anda kullandığımız uygulama sürümü ile uyuşmazlık yaşadık.

İlk defa API kullandığımız için API'yi server'a implement etmekte zorlandık.

İlk defa React Native ile uygulama yaptığımız için JavaScript diline adapte olmakta sıkıntılar yaşadık.

Projenin Bize Kattığı Yararlar

RestFul bir yapıda Android uygulaması yapmayı öğrendik.

Server yazma konusunda deneyim kazandık.

Bir API nasıl kullanılacağını öğrendik.

Görüntü işleme hakkında bilgi sahibi olduk.

Firebase hakkında deneyim kazandık.

Kullanılan Fonksiyonlar

const UploadImages = async (uri, setImage, isim);

Resim uri'sini aldığı uri değişkeni ile gönderilen isim ile Firebase'e resimleri yükler.setImage'i update ederek resmin ekrana basılmasını sağlar.

async function postData(url = "", data = {});

Sunucuya resim bilgilerini post atmak için kullanılan fonksiyon.

const fotoCek = (cropping, mediaType = 'photo');

Kullanıcı arayüzünde kameranın açılmasını ve ekranda gösterilmesini sağlayan fonksiyon.

GaleridenSec = (cropit, circular = false, mediaType);

Kullanıcı arayüzünde galeriden resim seçilmesini ve ekranda gösterilmesini sağlayan fonksiyon.

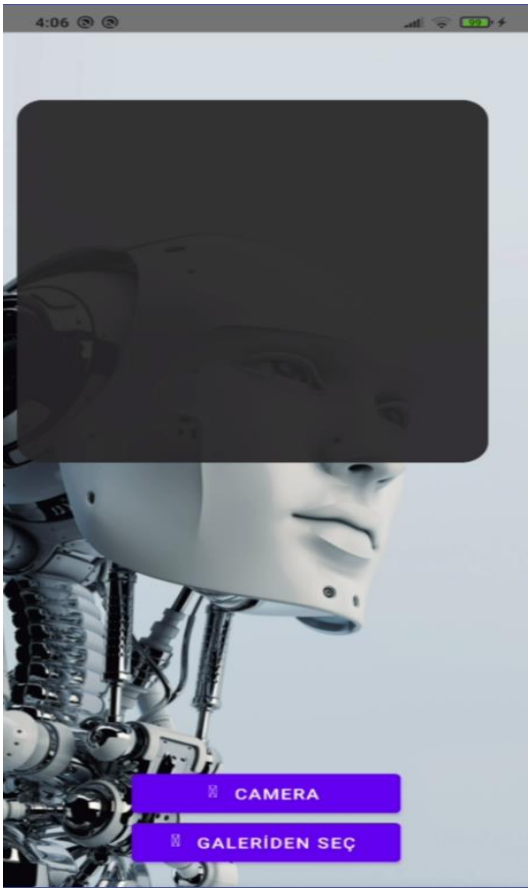
await tensorflow(answer, option, gonderilenUri, res);

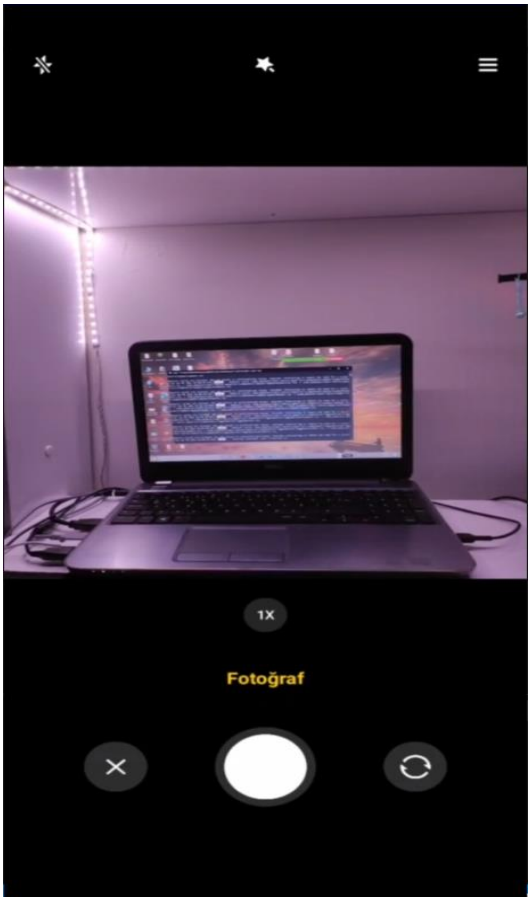
Görüntünün tespit edildiği ve çizdirildiği fonksiyondur.

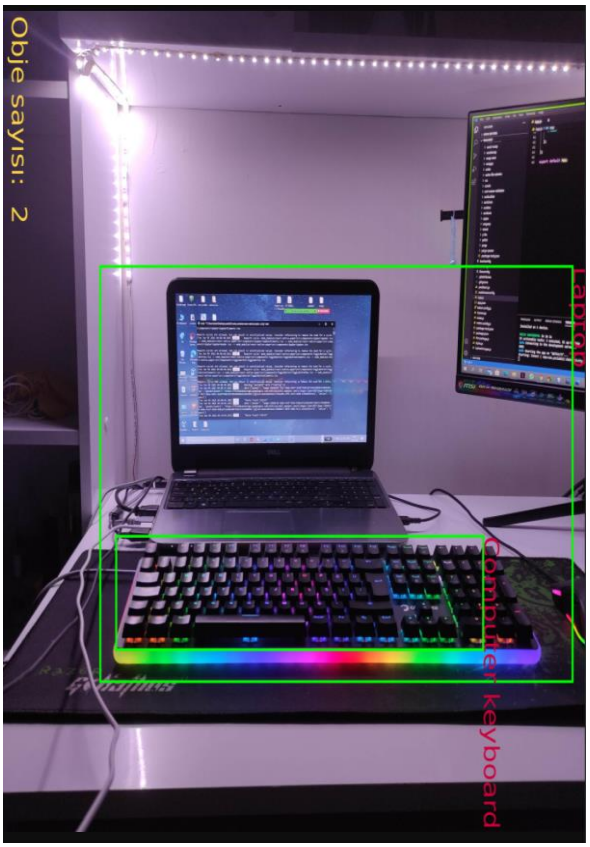
async function DosyaYukle();

İşlenmiş görüntüyü firebase yükleyen fonksiyondur.

EKRAN GÖRÜNTÜLERİ







KAYNAKÇA

https://cloud.google.com/vision/docs/object-localizer

https://programmingwithmosh.com/react-native/make-api-calls-in-react-native-using-fetch/#:~:text=In%20React%20Native%2C%20you%20can,method%20to%20make%20a%20request.

https://flaviocopes.com/canvas-node-generate-image/

https://www.npmjs.com/package/canvas

https://www.pluralsight.com/guides/upload-images-to-firebase-storage-in-react-native