

**TÜRKİYE CUMHURİYETİ  
YILDIZ TEKNİK ÜNİVERSİTESİ  
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**



**DERİN ÖĞRENME YÖNTEMLERİ İLE MARKA İÇEREN  
NESNELERİN TESPİTİ VE BULANIKLAŞTIRILMASI**

15011702 – Özgür KAN  
15011111 – Eray CINCI

**BİLGİSAYAR PROJESİ**

Danışman  
Doç. Dr. Mine Elif KARSLIGİL YAVUZ

Ocak, 2020



## **TEŞEKKÜR**

---

Bize bu tür projeler yapma fırsatı sağlayan Yıldız Teknik Üniversitesi Bilgisayar Mühendisliği Bölümüne teşekkürlerimizi sunarız.

Lisans eğitimimde bilgisayar projesi dersi kapsamında beraber çalışma fırsatı bulduğumuz ve proje konumuzlarındaki bilgi ve tecrübesiyle bize katkıda bulunan değerli hocamız Sayın Doç.Dr. Mine Elif KARSLIGİL YAVUZ'a sonsuz teşekkürlerimizi sunarız.

Bu projede, derin öğrenme dünyasında kullanılan nesne tespiti yönteminden yararlanılmıştır. Açık kaynak kodun geliştirilmesini destekleyen, yöntem ve paketlere erişimi sağlayan herkese teşekkür ediyoruz.

Hayatımız boyunca bize olan desteklerinden ve bu çalışmamız süresince bize gösterdikleri hoşgörüden ötürü değerli ailelerimize teşekkür ederiz.

Özgür KAN  
Eray CINCI

# İÇİNDEKİLER

---

<b>KISALTMA LİSTESİ</b>	v
<b>ŞEKİL LİSTESİ</b>	vi
<b>TABLO LİSTESİ</b>	vii
<b>ÖZET</b>	viii
<b>ABSTRACT</b>	ix
<b>1 Giriş</b>	1
1.1 Nesne Tespiti Ağları(Object Detection Networks) . . . . .	2
<b>2 Ön İnceleme</b>	4
<b>3 Fizibilite</b>	6
3.1 Teknik Fizibilite . . . . .	6
<b>4 Sistem Analizi</b>	10
<b>5 Sistem Tasarımı</b>	11
5.1 Veri Kümesinin Hazırlanması . . . . .	11
5.2 Nesne Tespiti . . . . .	12
5.2.1 R-CNN . . . . .	12
5.2.2 Fast R-CNN . . . . .	13
5.2.3 Faster R-CNN . . . . .	14
5.3 Bulanıklaştırma . . . . .	15
<b>6 Uygulama</b>	17
6.1 Veri Kümesi . . . . .	17
6.2 Başarılı ve Başarısız Durumlara Örnekler . . . . .	18
6.3 Deneysel Sonuçlar . . . . .	19
6.4 Performans Analizi . . . . .	22
<b>7 Sonuç</b>	23

**Referanslar** **24**

**Özgeçmiş** **25**

## KISALTMA LİSTESİ

---

KSA	:Konvolüsyonel Sinir Ağı
CNN	:Evrişimsel Sinir Ağı (Convolutional Neural Network)
R-CNN	:Bölgesel Evrişimli Sinir Ağı (Regional Convolutional Neural Network)
Fast R-CNN	:Hızlı Bölgesel Evrişimli Sinir Ağı (Fast Regional Convolutional Neural Network)
Faster R-CNN	:Daha Hızlı Bölgesel Evrişimli Sinir Ağı (Fast Regional Convolutional Neural Network)
RPN	:Bölge Teklif Ağrı (Region Proposal Network)
SVM	:Destek Vektör Makinesi (Support Vector Machine)
ROI	:İlgilenilen Bölge (Region of interest)
YOLO	:Sadece bir kez bak (You only look once)
CPU	:Merkezi İşlem Birimi (Central Process Unit)
GPU	:Grafik İşlem Birimi (Graphics Process Unit)
CUDA	:Birleşik Aygıt Mimarısını Hesaplama(Compute Unified Device Architecture)
cuDNN	:Derin Sinir Ağı Kütüphanesi(CUDA Deep Neural Network Library)

## **ŞEKİL LİSTESİ**

---

Şekil 3.1 Gantt Diagram . . . . .	8
Şekil 4.1 Use Case Diagram . . . . .	10
Şekil 5.1 Veri Artırma Yöntemleri . . . . .	11
Şekil 5.2 R-CNN . . . . .	13
Şekil 5.3 Fast R-CNN . . . . .	13
Şekil 5.4 Faster R-CNN . . . . .	14

## **TABLO LİSTESİ**

---

Tablo 3.1 Projede çalışan kişilerin maliyeti . . . . .	8
Tablo 3.2 Projede kullanılan sistemin ve yazılımların maliyetleri . . . . .	9
Tablo 5.1 Nesne Tespit Yöntemlerinin Karşılaştırılması . . . . .	14
Tablo 6.1 Projede Kullanılan Sınıflar ve Resim Sayıları . . . . .	17
Tablo 6.2 Projede Başarılı Tespit Edilen Nesneler . . . . .	18
Tablo 6.3 Projede Başarısız Tespit Edilen Nesneler . . . . .	19
Tablo 6.4 Projedeki Sınıfların Başarı Oranları-1 . . . . .	20
Tablo 6.5 Projedeki Sınıfların Başarı Oranları-2 . . . . .	21
Tablo 6.6 Sistemin Çıktıları Ve Çalışma Hızı . . . . .	22

## ÖZET

---

# Derin Öğrenme Yöntemleri ile Marka İçeren Nesnelerin Tespiti ve Bulanıklaştırılması

Özgür KAN  
Eray CINCI

Bilgisayar Mühendisliği Bölümü  
Bilgisayar Projesi

Danışman: Doç. Dr. Mine Elif KARSLIGİL YAVUZ

Günümüzde popüler markalar sürekli bir yarış halindedir ve sektörlerinde lider olmayı hedeflerler. Markaların, satış rakamlarını artttırmak için kullandığı yöntemlerin başında ürün yerleştirme gelir. Ürün yerleştirme basit, etkili ve insanlara ulaşma oranı çok yüksek olan bir yöntem olduğu için son dönemlerde popülerliği giderek artmıştır. Genellikle dizi veya film sektörlerinde kullanılan bu yöntem sayesinde yapım şirketleri önemli gelirler elde etmektedir. Ayrıca ürünlerin dizilerde veya filmlerde gösterilmesi reklam veren markaların prestij ve popülerliğini arttırmır. Bu nedenden dolayı dizi ve film sektöründe ürün yerleştirme yaygın olarak kullanılmaktadır.

Bu projede nesne tespiti ağlarından biri olan Faster R-CNN kullanılarak videolarda istenmeyen markaların tespit edilmesi ve tespit edilen bu markaların bulanıklaştırılmasını sağlayan bir sistem gerçekleştirılmıştır.

**Anahtar Kelimeler:** Marka, Ürün Yerleştirme, Bulanıklaştırma, Faster R-CNN

## **ABSTRACT**

---

# **BLURRING OPERATION OF BRAND INCLUDING OBJECTS WITH DEEP LEARNING METHODS**

Özgür KAN

Eray CINCI

Department of Computer Engineering

Computer Project

Advisor: Assoc. Prof. Mine Elif KARSLIGIL YAVUZ

Most popular brands are in a constant race with each other in order to be the leader of their own sectors. Wishing to increase the number of product sales, brands use product placement as their primary method. Product placement is a highly simple and effective method for making their product seen. It also has a significant reach to anyone, making it one of the most popular method. Mostly used in movies and television series, brands make a essential revenue with product placement. Moreover, with making their product seen on television, brands increase their prestige and popularity. Thus, product placement is widely used in television series and movie sectors.

In this project, unwanted brands are located with a object detection network called Faster R-CNN and blurred.

**Keywords:** Product placement, brand, blur, Faster R-CNN

# 1

## Giriş

---

Ürün yerleştirme, büyük bir kitleyi hedefleyen bir video produksiyonunda markalı ürünlerin ve hizmetlerin yer aldığı bir reklam türüdür. Ürün yerleştirme, diğer pazarlama türlerinden daha ekonomik ve daha kolay bir şekilde uygulanan bir yöntemdir. Genellikle filmlerde, televizyon şovlarında, kişisel videolarda, radyoda ve daha az yaygın olan canlı performanslarda bulunur. Ürün yerleştirme hakları karşılığında yapım şirketleri önemli gelirler elde eder.

Günümüzde büyük markalar, ürünlerinin popülerliğini ve satış rakamlarını arttırmak için popüler olan dizi veya filmlerde ürününün görünmesi için yapım şirketleriyle belirli bir ücret karşılığında anlaşmaktadır. Dizi ve film sektörleri için önemli bir gelir kaynağı oluşturmaktadır. Bu nedenden dolayı dizi ve film sektöründe ürün yerleştirme yöntemi yaygın olarak kullanılmaktadır. Ürünlerin dizi veya filmlerde gösterilmesi reklam veren markaların prestij ve popülerliğini artttır.

Bazı durumlarda istenmeyen markalar sahnelerde gözükebilir ve bilinçsiz olarak o markanın reklamı yapılmış olur. Bu durumda editörler tarafından o sahnede bulunan ve istenmeyen markaların ürünleri bir video düzenleme programı kullanılarak bulanıklaştırılır. Böyle yapılarak oluşan bu sorunun önüne geçilmeye çalışılır. Bu yöntem hem iş yükü olarak bir problem ortaya çıkarır hem de zaman açısından daha uzun süre gerektirir. Ayrıca canlı ortamlarda markaların bu yöntem ile çok kısa bir sürede tespit edilip bulanıklaştırılması imkansızdır.

Projedeki amacımız derin öğrenme yöntemleri kullanarak videolarda istenmeyen markaların tespit edilmesi ve bu markaların otomatik olarak bulanıklaştırılmasıdır. Projede manuel olarak ürünlerin bulanıklaştırılmasında oluşan zaman ve iş yükü gibi sorunların önüne geçilmesi hedeflenmiştir.

Proje oldukça geniş bir kullanım alanına sahiptir. Dizi veya film sektöründe yapım şirketleri, canlı video içerikleri üreten insanlar ise bireysel olarak bu projeden yararlanabilecektir.

Derin Öğrenme(Deep Learning), yapay sinir ağları denilen beynin yapısından ve işlevinden ilham alan algoritmalarla ilgili bir makine öğrenmesi alt alanıdır. Derin Öğrenme bir makine öğrenme yöntemidir. Verilen bir veri kümesi ile çıktıları tahmin edecek yapay zekayı eğitmemize olanak sağlar. Projede Derin öğrenme (Deep Learning) kütüphanesi olarak TensorFlow kullanılacaktır. Google'in açık kaynak kodlu olarak sunduğu bu kütüphane özellikle derin öğrenme için yaygın olarak kullanılmaktadır. Projemizde hem açık kaynak kodlu olması nedeniyle hem de internet ortamında kaynaklarının daha fazla olması nedeniyle Tensorflow kütüphanesini kullanmayı tercih etti. Tensorflow dışında Keras,Caffe,TFLearn gibi çeşitli kütüphanelerde mevcuttur.

Nesne Tespiti(Object Detection), araba, bisiklet, trafik ışıkları, çiçekler, markalar ve insanlar gibi gerçek dünyadaki nesne örneklerini hareketsiz görüntülerde veya videolarda bulma işlemidir. Bir görüntüyü bir bütün olarak daha iyi anlayabilmemizi sağlayan, görüntü içindeki birden fazla nesnenin tanınmasına, yerelleştirilmesine ve algılanmasına olanak tanır. Nesne tespiti, birden çok yöntemle yapılabilir:

- Özellik Tabanlı Nesne Tespiti
- Viola Jones Nesne Tespiti
- HOG Özellikleri ile SVM Sınıflandırması
- Derin Öğrenme Nesne Tespiti

Projemizde Derin Öğrenme(Deep Learning) ve Tensorflow kütüphanesi kullanacağımızdan biz bu adımda Derin Öğrenme Nesne Tespitini kullanacağız.

## 1.1 Nesne Tespiti Ağları(Object Detection Networks)

Son yıllarda, derin öğrenme yöntemleri nesne tespitinde yaygın olarak kullanılmıştır. Resim ve videolardaki nesne tespiti çoğunlukla CNN tarafından gerçekleştirilmiştir. Derin sinir ağları farklı şekillerde optimize edilmeye çalışılmıştır. Bunlar; R-CNN, Fast R-CNN[1], Faster R-CNN[2] gibi olup nesne tespiti için geniş ölçekte kullanılır. Bu yöntemlerin avantajı resmi parçalara ayırıp her parçaya bakmaktansa, her pixeli ağa bir kere verip daha hızlı sonuç almaktır.

Nesne tespiti ağlarından(object detection networks) biri olan Faster R-CNN'in hızlı çalışma yapısı sebebiyle, canlı(Real-time) durumlarda bile hem hız hem de doğruluk açısından belirli markaların bulanıklaştırılmasını hedefliyoruz. Yapacağımız bu proje film ve dizi sektöründe birçok yerde kullanılabilir. Kullanacağımız Faster R-CNN mimarisinin hızlı olması bunu gerçek kılmaktadır. Gerçek zamanda çok hızlı ve doğru sonuç vermesi Faster R-CNN'i bu sektörde kullanılan en iyi yapılaradan biri olmasını sağlamıştır.

## 2 Ön İnceleme

---

Son yıllarda, derin öğrenme(Deep Learning) ile nesne tespiti(Object Detection) projelerinin başarıları giderek artmıştır. Bunun en önemli nedeni günümüzde derin öğrenme ağlarının(Deep Learning Network) başarı oranının artması ve veri toplamalarının, yaygınlaşan sosyal medya kullanımıyla birlikte daha kolay hale gelmesi olarak gösterilebilir. Bu bölümde bizim projemize benzer projeleri inceleyip kullandıkları yapıları, veri kümelerini, başarı oranlarını ve sistemlerini ele alacağız.

İlk olarak projemize çok benzeyen ve Franck FOTSO [3] tarafından geliştirilen logo tespiti projesini inceledik. Bu projede logoyu tespit etmek için Faster R-CNN ağı kullanılmış olup bizim projemizde kullanacağımız ağ yapısıyla benzerlik göstermektedir. Bizim projemizden farklı olarak Tensorflow kütüphanesi yerine Caffe kütüphanesini kullanılmıştır. Projede sadece 20 sınıfı küçük bir veri seti kullanılmıştır. Projede işletim sistemi Ubuntu 16.04 64 bit, ekran kartı Nvidia GTX 950M 4G kullanılmıştır. Yazılım olarak ise Cuda 8.0, CuDNN 3.0.8, Python 2.7.12, OpenCV 3.1.0 ve Caffe yazılımları kullanılmıştır. Bu projede veri seti olarak ROMYNY Logo 2016 kullanılmıştır. Son olarak yapılan bu projede yaklaşık olarak 30000 adımda eğilmiştir. Başarı oranı olarak %75'lik bir başarı oranı elde edilmiştir.

İkinci olarak inceledğimiz proje ise 2016 yılında International Journal of Innovative Research in Computer and Communication Engineering tarafından yapılmış olan projede elle yazılmış arapça 0 dan 9'a kadar olan sayıların tespiti edilmeye çalışılmıştır. Veriseti olarak 45000 örnek kullanılmıştır. Bu projede CNN kullanılmıştır. Girilen resimde sadece bir tane sayı olduğu için bölgesel bir tespite gerek yoktur. CNN'in seçilmesinin asıl sebebi budur. CNN görüntüyü çeşitli katmanlarla işler. Bu katmanlar; Convolutional Layer, Pooling Layer, Fully-Connected Layer(Sınıflamada kullanılır). Başarı oranı %95.7'dir.[4]

2017'de yapılmaya başlanan ve günümüzde hala devam etmekte olan bir diğer proje ise Derin öğrenme tekniklerinden biri olan Faster R-CNN ile biyolojik resimlerin içindeki parazitlerin tespiti üzerindedir. Bu projeyi zor yapan durum ise hücrelerin şekillerinde olan değişimlerle birlikte renk, yoğunluktur. Veriseti kitleyi yüzünden bu proje üzerinde günümüzde hala çalışılmaktadır.[5]

Bir başka yapılmış proje de yaya geçidinde olan insanların tespitidir. İnsanların aynı hızda yürümemesi ve dolayısıyla birbirinin önüne geçmesi insan tespitini zorlaştırmaktadır. Bu problemin önüne geçmek için yeni bir ROI yöntemi tasarlanmıştır. Proje Fast R-CNN mimarisyle gerçekleşmiştir. Veriseti 2120 resimden oluşmaktadır. Bunların 1832'si eğitime, 288 ise test için kullanılmıştır. Verisetinin ismi INRIA'dır. Başarı oranı ise %93.6'dır.[6]

# 3

## Fizibilite

---

Bu projede bir video içerisindeki istenmeyen bir markanın tespit edilip daha sonra o markanın bulunduğu koordinatlara bulanıklaştırma işlemi yapıyoruz. Bu işlemleri sağlamak için projede derin öğrenme kütüphanesi, nesne tespiti ağı ve bulanıklaştırma fonksiyonuna ihtiyacımız olduğundan dolayı bu yapıları kullanıyoruz.

Projemizde derin öğrenme kullanacağımız için bu noktada Google tarafından geliştirilen ve açık kaynaklı olan TensorFlow kütüphanesini kullanıyoruz. Tensorflow kütüphanesi bize bu noktada iki seçenek sunuyor. Bunlar CPU VE GPU sürümleri olarak ikiye ayrılıyor. CPU sürümü daha küçük ağlarda kullanılıyor ve daha yavaş çalışıyor. Biz TensorFlow kütüphanesinin GPU versiyonunu kullanıyoruz. Bunun nedeni hem geniş bir ağ kullacanak olmamız, ayrıca CPU sürümüne göre çok daha hızlı olması ve projemiz amacı video üzerindeki sahnelerde tespit edilen nesneleri bulanıklaştırma olduğu için bize daha hızlı bir seçenek sunduğundan dolayı GPU sürümünü kullanıyoruz.

Markaların bulunduğu alanlar tespit edilecek ve daha sonra tespit edilen alanlar üzerine bulanıklaştırma işlemi uygulanıp son kullanıcıya bu hali verilecektir. Nesnelerin tespiti için nesne tespiti ağlarından olan Faster R-CNN ağını kullanıyoruz. Markaların tespiti için modelimiz hızlı ve duyarlı olması açısından Faster R-CNN bizim ihtiyacımızı karşılıyor.

Bulanıklaştırma için ise neredeyse tüm görüntü işleme işlemlerinde kullanılan ve bu alanın öncüsü olan OpenCV kütüphanesinin 3.4.0 versiyonu ve kütüphanenin içinde bulunan Gaussian Filter fonksiyonu kullanılacaktır. Bu işlemler görüntü üzerinde olacağı için grafik kartı ne kadar iyi olursa çalışma hızı o kadar artacaktr.

### 3.1 Teknik Fizibilite

Teknik fizibilite üç temel başlık altında incelenmiştir. Bunlar Yazılım Fizibilitesi, Donanım Fizibilitesi ve Haberleşme (İletişim) Fizibilitesidir.

- **Yazılım Fizibilitesi:** GPU'nun donanımsal hesaplama gücünden faydalanan makam amacıyla NVIDIA firmasının 2006 yılında geliştirdiği paralel hesaplama mimarisi olan CUDA(cuda\_9.0.176\_win10 sürümü) programı kullanılmıştır. Linux, Windows ve Mac Osx platformları üzerinde çalışabilmektedir. FORTRAN, C ve Python gibi dilleri destekler. Rakiplerine göre avantajları paylaşımı bellek kullanımı, GPU'dan daha hızlı veri okuma ve bit düzeyinde işlem yapılabilmesine olanak sağlama olarak sayılabilir.

Derin öğrenme için gerekli işlemleri yapabilmemizi sağlayan yardımcı bir kütüphane olan CuDNN (cudnn9.0windows10x64v7 sürümü) kütüphanesinden faydalaniyoruz. CuDNN, derin sinir ağları için GPU ile hızlandırılmış bir kütüphanedir. CuDNN, ileri ve geri evrişim, havuzlama, normalizasyon ve aktivasyon katmanları gibi standart rutinler için ayarlanmış uygulamalar sunar.

Projemizde kodlama ve arayüz kısmı için Python programla dili kullanılmıştır. Zengin kütüphane olanakları sunar ve görüntü işleme alanında yaygın kullanılan bir programlama dilidir. Python kullanmak için Anaconda (Anaconda35.2.0-Windows-x86\_64) programı kullanılmıştır. Anaconda, veri bilimi ve benzeri bilimsel uygulamalar için python kullanmak isteyenlere hazırlanmış tümleşik bir python dağıtımıdır. Veri bilimi, yapay zeka vb. konularda sıkça kullanılan kütüphanelerin yanı sıra jupiter notebook ve spyder gibi araçları da barındırır.

Projenin sonuçlarını görüntülemek için Google tarafından geliştirilen Tensorboard aracı kullanılmıştır. Ağımızın yapısını ve birçok parametreyi bize basit bir arayüzle gösterir. Aynı zamanda interaktif grafikler çizebiliyor olması işimizi kolaylaştırır. Eğitim yapılırken sonuçları görmek ve müdahale edebilmemize olanak sağlar.

Resimlerde bulunan nesneleri etiketlemek için LabelIMG programı kullanılmıştır.

Projenin daha rahat ve çoğu kişi tarafından anlaşılabilir olması için işletim sistemi Windows 10 olarak seçilmiştir. Projeyi kullanmak isteyen kişilerde Windows 10 işletim sistemine sahip olmalıdır.

- Donanım Fizibilitesi:** Görüntü işleme performans gerektiren bir işlem olduğu için donanım özellikleri iyi olan bir bilgisayar gereklidir. Projede iki bilgisayar kullanılmıştır.

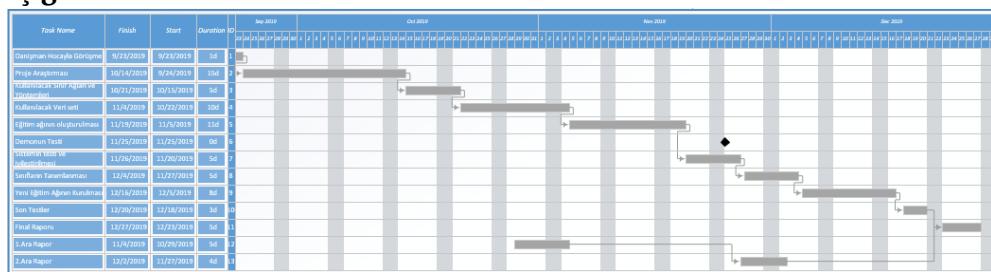
Birinci bilgisayarın özellikleri; Nvidia GTX-N960M 4GB ekran kartı, intel i7-6700HQ işlemci ve 8GB RAM olarak projede kullanılmıştır.

İkinci bilgisayarın özellikleri; Nvidia GTX-660M 2GB ekran kartı,intel i7-3630QM işlemci ve 8GB RAM olarak projede kullanılmıştır.

Projeyi kullanmak isteyen kişiler için grafik kartlarının CUDA programını ve cuDNN kütüphansını desteklemesi gerekmektedir.Bu yüzden minimum sistem gereksinimleri;Nvidia GTX-650M 2GB ve üstü ekran kartı,intel i5-3630QM ve üstü işlemci ve 4GB RAM ve üstü bir sistem kullanarak projeyi sorunsuz bir şekilde kullanabilirler.

- İletişim Fizibilitesi:** İletişim Fizibilitesi üç temel altbaşlık altında incelenmiştir. Bunlar İş Gücü ve Zaman Planlaması, Yasal Fizibilite, Ekonomik Fizibilite'dir.

#### - İş gücü ve Zaman Planlaması



Şekil 3.1 Gantt Diagram

- Yasal Fizibilite:** İnternette bulunan çeşitli veri setlerini kullanmak için, kullanım şartları ve gizlilik politikaları kabul edilmiştir. Ayrıca nesne tespiti için eğitimde kullanılacak resimlere dikkat edilmesi gerekmektedir. Projemizde kişisel verileri kullanmadığımız ve okulumuz tarafından desteklendiği için projemiz yasaldır.

- Ekonomik Fizibilite:**

Tablo 3.1 Projede çalışan kişilerin maliyeti

İsim	Haftalık Çalışma Süresi	Proje Süresi	Aylık Ücret	Toplam Maliyet
ÖZGÜR KAN	13 saat	3 AY	5000 TL	15000 TL
ERAY CINCI	13 saat	3 AY	5000 TL	15000 TL

Projede çalışan kişilere toplam 30.000 TL ücret ödenecektir.

**Tablo 3.2** Projede kullanılan sistemin ve yazılımların maliyetleri

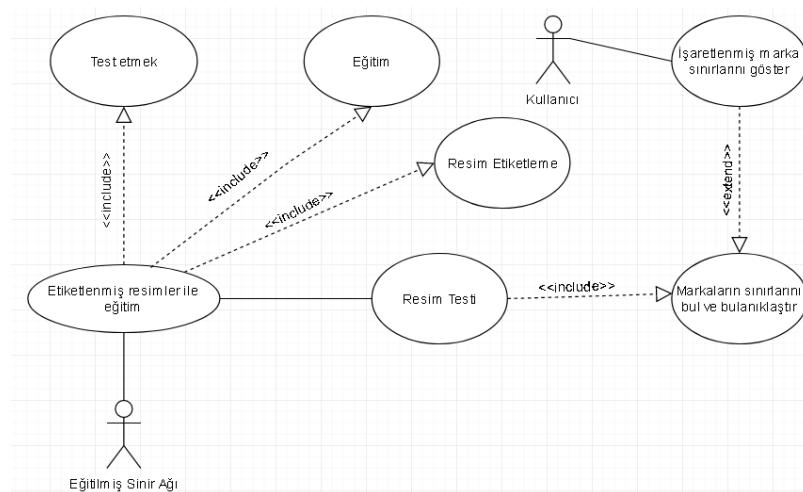
	Fiyat
i7-6700HQ	2419 TL
Nvidia GTX-960M 4GB	1500 TL
Microsoft Windows 10	889 TL
Microsoft Visio Professional	889 TL

Sistem ve yazılım fiyatları 6 Ocak 2020 tarihindeki piyasa değerleri üzerinden hesaplanmıştır. Bu fiyatlara göre proje için gerekli sistemin ve yazılımın toplam fiyatı 8574 TL olarak hesaplanmıştır.

Son olarak projenin toplam maliyeti yaklaşık olarak 40.000 TL olarak hesaplanmıştır.

## 4 Sistem Analizi

Sistem kullanıcının bir arayüzden videosunu seçmesiyle çalışmaya başlar. Girdi videosunu alan sistem bu video üzerindeki çerçeveleri (frame) işleyerek derin öğrenme ağına gönderir. Derin öğrenme ağı Tensorflow-GPU kütüphanesi ve nesneleri tespit etmek için Faster R-CNN mimarisi kullanarak marka içeren bölgeleri tespit eder. Tespit edilen bu bölge bir çerçeve içine alınır ve hangi sınıfı ait ise o sınıfın ismi çerçevenin yukarısına yazılır. Bu noktada nesnenin bulunduğu koordinatlara da ulaşmış olduk. Daha sonra bu koordinatlar OpenCV kütüphanesinin Gaussian bulanıklaştırma [7] fonksiyonuna gönderilerek tespit edilen nesne bulanıklaştırılmış olur. Bir video içerisinde istenmeyen her nesne tespit edilir ve yukarıdaki yöntemler uygulanarak bulanıklaştırılır. Son olarak kullanıcıya videonun bulanıklaştırılmış hali arka planda kaydedilerek kullanıcıya bulanıklaştırılmış video sunulur.



Şekil 4.1 Use Case Diagram

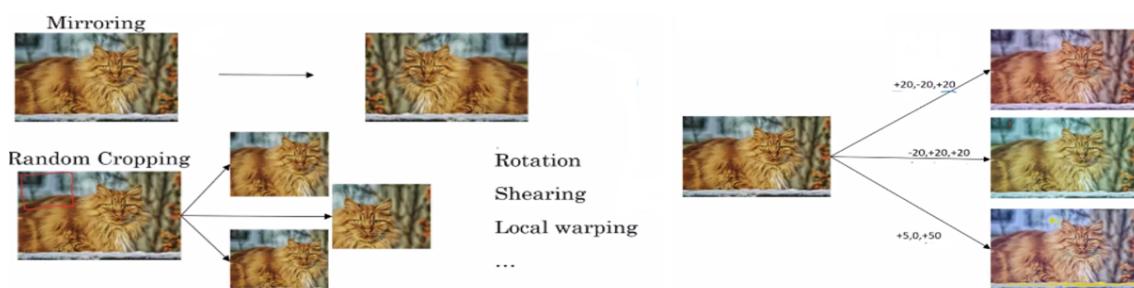
# 5 Sistem Tasarımı

Oluşturulacak sistemde, kullanıcı sisteme bir video verecek, sistem daha önceden belirlenmiş sınıfları içeren nesnelerin tespiti ve bulanıklaştırılmasını yapacaktır. Sistemin eğitilmesinde Faster R-CNN derin öğrenme ağı kullanılmıştır. Faster R-CNN diğer derin öğrenme ağlarına göre daha hızlı ve iyi sonuçlar verdiği için kullanılmıştır.

## 5.1 Veri Kümesinin Hazırlanması

Veri seti olarak Open Logo Dataset kullanılmıştır[8]. Bu veri seti çok geniş bir veri seti olduğu için biz bu veri setinin sadece bir bölümünü kullandık. Bu veri seti içerisinde projemizde kullanılmaya uygun olanlar derin öğrenme ağında kullanmak için ayırtırıldı. Bu işlem yapılırken bazı resimlere veri artırma (Data Augmentation) yöntemi uygulanmıştır.

Veri artırma (Data Augmentation) yöntemi küçük veri setlerinde özellikle başarımı artırmak için verilerin çeşitli bozulma etkilerine maruz bırakılarak artırılmasıdır. Bu şekilde modelin farklı koşulları da öğrenmesi sağlanmaktadır. Görüntünün çeşitli eksenlere göre simetriklerinin alınması, rastgele bir örnek parçasının kesilip alınması, eksenlerinin değiştirilmesi, renk oranlarının değiştirilmesi, gibi birçok farklı şekilde veriden elde edilen yeni veri parçaları oluşturularak çoğaltma/artırma yapılabilir. Böylece model ezberleme (overfit) eğiliminden uzaklaşmaktadır.



Şekil 5.1 Veri Artırma Yöntemleri

Sınıfların sayısı 17 olarak belirlenip her biri için 300 tane resim seçilmiştir. Ağın hızlı eğitilmesindeki en önemli faktörlerden biri resimlerin boyutlarıdır. Büyük boyutlu resimler daha yavaş işlendiği için resimlerin hepsi 500x500 şeklinde boyutlandırılmıştır.

## 5.2 Nesne Tespiti

Bu projede nesne tespiti Faster R-CNN ağı ile yapılmıştır. Aşağıda Fater R-CNN'den önce kullanılan ağlar ve en son detaylı olarak Faster R-CNN mimarisi hakkında bilgi verilmiştir.

### 5.2.1 R-CNN

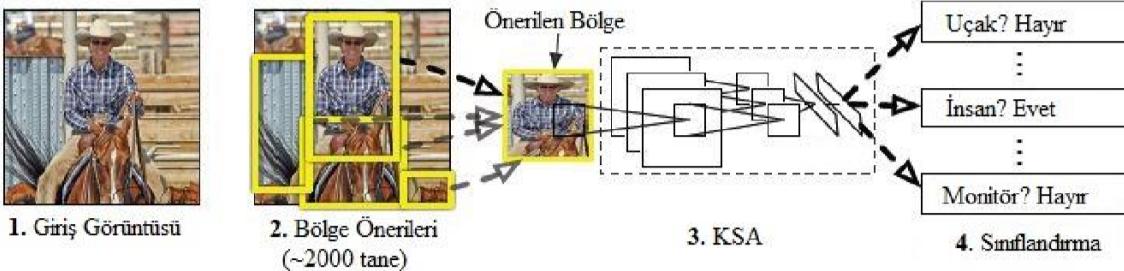
Bir nesnenin görüntünün neresinde olduğunu tespit etmek ve sınırlarını belirlemek için kullanılan ilk yöntemlerden birisidir ve 2014 yılında geliştirilmiştir.[1] Bu yöntem temel olarak 4 bölümden oluşur. Yöntem ilk olarak giriş görüntüsünü alır.

İkinci bölümde, giriş görüntüsü üzerinde nesne olma ihtimali olan yaklaşık 2000 bölgeyi seçmek için Ross Girshick [2] ve arkadaşları seçici arama (selective search) adında bir yöntem önerdi ve seçilen bu alt bölgelere bölge önerileri (region proposals) adını verdi. Böylece çok sayıda bölgeyi sınıflandırmaya çalışmak yerine, sadece 2000 bölgeyle çalışabilmeye olanak sağlandı. Bu 2000 bölge önerileri, aşağıda yazılmış olan seçici arama (selective search) algoritması kullanılarak oluşturulur.

- İlk olarak görüntüden birçok alt bölge oluşturulur..
- Benzer bölgeleri yinelemeli olarak daha büyük bölgelere birleştirmek için greedy algoritması kullanılır.
- Son olarak bu bölgeler kullanılarak nihai alt bölgeler üretilir.

Üçüncü bölümde, ikinci bölümde seçilen bu bölgeler sırasıyla KSA'nın (CNN) girişine verilir ve her bölge için sırasıyla KSA(CNN) uygulanır.

Dördüncü bölümde ise, KSA'nın çıkışı sınıflandırma işlemine tabi tutulur ve çıkan özellik haritaları(features map) için sınıflandırma algoritmalarından SVM kullanılıp nesnenin sınıfına karar verilir.Nesnenin varlık ve yokluğu belirlenir daha sonrasında linear regression modelleme ile bölgelerin boyutu belirlenir ve doğru bölge yapay sinir ağına sokulur. Bu yönteminin en büyük problemi yavaş olması ve gerçek zamanlı sistemlerde uygulanmasının imkansız olmasıdır. Bu yapı Şekil 5.1'de gösterilmiştir.

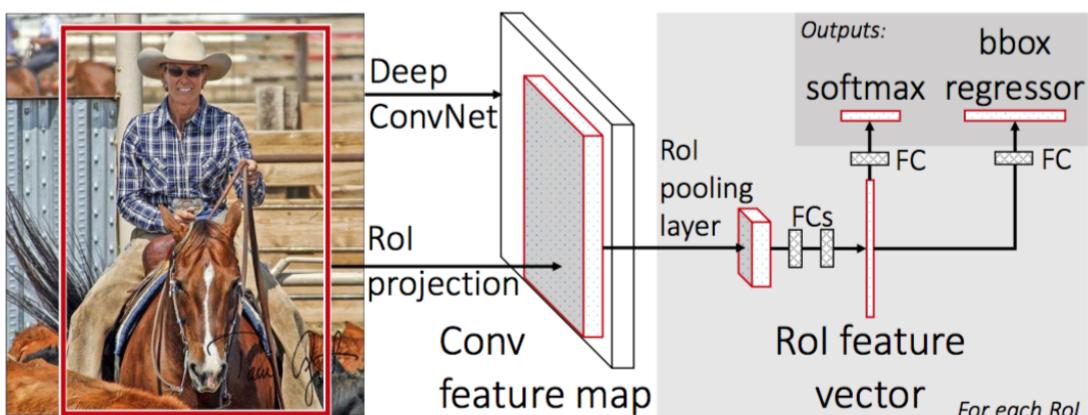


Şekil 5.2 R-CNN

### 5.2.2 Fast R-CNN

Bu yöntemin R-CNN'e kıyasen en büyük farkı CNN özelliklerini her parça için değil bir bütün resim için bir kere bulmak ve bölgeleri ona göre seçmektir(Selective Search ile). Ayırca sınıflandırma kısmında metot olarak SVM değil yapay sinir ağı katmanları içinde gerçekleştirebileceği bir derin öğrenme sınıflandırması softmax classification kullanmasıdır. CNN'i bir kere kullandığı için zamanda büyük bir kazanç sağlanmaktadır.

- Girdi resim CNN'e beslenir.
- Konvolüsyon özellik haritasından, önerileri tespit eder ve bunları karelere çarptırırız ve bir ROI havuzlama katmanı kullanarak, bunları tam bağlı bir katmana beslenebilmeleri için sabit bir boyutta yeniden sekillendiririz.
- ROI özellik vektöründen, önerilen bölgenin sınıfını ve sınırlayıcı kutu için offset değerlerini tahmin etmek için softmax katmanını kullanırız.

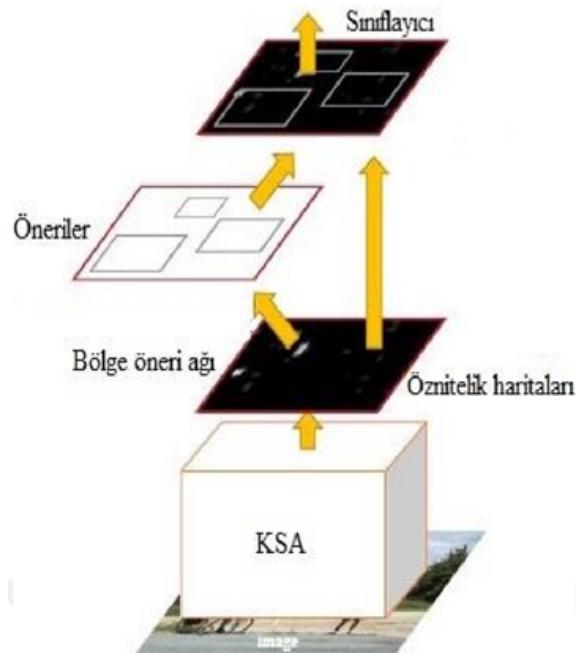


Şekil 5.3 Fast R-CNN

### 5.2.3 Faster R-CNN

Faster R-CNN, RPN ile Fast R-CNN modellerinin birleştirilmiş halidir. Bu yöntemin diğerlerinden farkı, bölge öneri ağı yapısını içermesidir. Diğer yöntemlere göre Selective Search algoritması yerine RPN kullanır. RPN tam olarak evrişimsel bir sinir ağıdır. Girdisi bir görüntü olan önerilen bölgeleri çıkartmak için kullanılır ve önerilen bölgelere aynı zamanda skorlar (tahmin etme doğruluğu) verilir. Önerilen bölgelerin sınıflama katmanı ve skor değerlerini tahmin katmanı aşamasında Fast R-CNN ağında birleştirilir. Fast R-CNN ağının da iki çıkış katmanı bulunmaktadır. Bunlardan ilki Softmax sınıflayıcı katmanı diğeri ise tespit edilen bölgenin tespit edilme doğruluğunu veren regresyon katmanıdır.

Faster R-CNN ile giriş görüntüsü sınıflandırma süreci boyunca sadece bir kez hesaplanır ve daha sonra tespit edilmesi gereken bölgelerin bulundukları yeri belirleyen kutuların sınırlarını düzeltir.



**Şekil 5.4** Faster R-CNN

**Tablo 5.1** Nesne Tespit Yöntemlerinin Karşılaştırılması

	R-CNN	FAST R-CNN	FASTER R-CNN
Öneriler ile birlikte bir resmin işleme süresi	50 saniye	2 saniye	0.2 saniye
Hızı	1X	25X	250X
Hassaslık Oranı	66.0	66.9	66.9

Tablo-5.1, nesne tespitinde yaygın olarak kullanılan üç yöntemin kıyaslanması göstermektedir.

### 5.3 Bulanıklaştırma

Projemizin son adımı olan tespit edilen nesneyi bulanıklaştırma işlemi projemiz için önemli bir rol oynamaktadır. Bulanıklaştırma için kullandığımız OprenCV kütüphanesi bize 3 adet farklı bulanıklaştırma fonksiyonu sunmaktadır. Bunlar; Ortalama(Averaging), Medyan Bulanıklaştırma(Median Blurring) ve son olarak Gauss Bulanıklaştırma(Gaussian Blurring) fonksiyonlarıdır. Filtreleme yapmak için resmin Konvolüsyon işleminden geçmesi gereklidir. Konvolüsyon işlemi; bir çekirdek şablonun (matrisin/kernel) resim üzerindeki piksellerle kaydırma ve çarpma işlemi yapılması olarak tanımlanabilir. Bu işlemde çekirdek şablon resim üzerinde kaydırılır ve değeri resim üzerindeki uygun piksellerle çarpılır. Belli özel uygulamalar için çeşitli standart şablonlar vardır. Burada şablonun büyülüğu ve şekli, işlemin özelliklerini belirler.

Ortalama(Averaging)filtresi, görüntüleri yumuşatmadı kullanılan basit ve uygulanması kolay bir yöntemdir. Diğer bir deyişle, bir piksel ile diğerleri arasındaki değişim miktarını azaltmaktadır. Genellikle görüntülerdeki gürültüyü azaltmak için kullanılır. Ortalama(Averaging)filtresi, bir görüntünün her bir piksel değerini komşularının ve kendisinin dahil olduğu ortalama değer ile değiştirmektedir. Bu durum, çevresindekileri temsil etmeyen piksel değerlerinin ortadan kalkmasına yol açar. Ortalama filtresi bir konvolüsyon filtresidir. Konvolüsyon filtreleri çekirdek şablon (kernel) temeline dayanır. Çoğunlukla  $3 \times 3$  kare çekirdek şablon kullanılır. Bazı yumuşatma işlemlerinde daha büyük şablonlar ( $5 \times 5$ ,  $7 \times 7$  gibi) kullanılabilir. Büyük şablonun tek bir taramadaki etkisine benzer bir etki, küçük şablonun birden fazla geçisi ile de sağlanabilir.

Ortalama filtresi, bir görüntüdeki gürültüyü azaltmak için kullanılan en basit yöntemdir. Ancak gürültü daha az belirgin hale getirilirken, görüntüde yumuşatılmış olmaktadır. Kullanılan çekirdek şablonun (matrisin) boyutu artırılırsa yumuşatma daha da artacaktır. Ortalamafiltrelemeye ilgili iki ana sorun bulunmaktadır:

- Resmi çok iyi temsil etmeyen değere sahip bir piksel, yakın bölgedeki tüm piksellerin ortalama değerini önemli ölçüde etkiler. Bu da resmin değişimine sebep olur.
- Filtre (şablon) bir kenar üzerinden geçerken, kenarın her iki tarafındaki pikseller için yeni değerler üreticektir ve bu durum kenarın bulanıklaşmasına sebep olacaktır. Eğer keskin kenarların kaybolması istenmiyorsa bu bir sorun olabilir.

Bu iki problemi gidermek için Ortalama filtresi (mean) yerine, Medyanfiltresi (Median Filter) geliştirilmiştir. Fakat bu filtrenin hesaplama süresi uzun sürmektedir.

Medyan Bulanıklaştırma(Median Blurring),normal olarak ortalama filtresi gibi bir resimdeki gürültüyü azaltmak için kullanılır. Ancak resim üzerindeki detayların kaybolmaması noktasında ortalama filtreden çok daha iyi sonuç verir.Medyan filtre de ortalama filtре gibi her pikselin değerini hesaplamak için yakınındaki komşularına bakar. Medyan filtresinde piksel değeri komşu piksel değerlerinin ortalaması ile değiştirmek yerine, komşu pikselleri sıralayıp sıranın ortasındaki değeri alır. Eğer incelenen bölge (şablonun içerişi) çift sayıda piksel varsa,orta değer olarak, ortada bulunan iki pikselin ortalaması kullanılır.Medyan filtrenin ortalama filtreye nazaran iki avantajı vardır.

- Orta değeri (median) kullanmak, ortalama değeri kullanmaktan (mean) daha güçlü olarak şablonu temsil eder. Temsil yeteneği uzak bir piksel sıralanan dizinin uçlarında kalacağından (hiç bir zaman ortada bulunmayacaktır) oradaki komşuların genel temsilini etkilemesi imkansız hale gelmiş olur.
- Medyan değer (orta değer), komşu piksellerin birinin değeri olması gerektiği için, kenar boyunca hareket ettiğinde gerçekçi olmayan piksel değerleri oluşturmaz. Bu nedenle, medyanfiltre, keskin kenarları ortalama filtreden daha iyi korur. Örnekleyecek olursa, siyah ve beyazdan oluşan bir sınırdı ortadaki değer ya siyah olur yada beyaz olur. İkisinin ortalaması olan gri olmayacağı. Böylece kenar üzerindeki keskinlik kaybolmamış olacaktır.

Medyan filtresinin dezavantajı olarak şundan bahsedilebilir. Resim üzerinde büyük boyutlarda gürültü varsa,Medyanfiltresi küçük matrislerle (şablonla) gürültüyü tam olarak kaldırılamaz. Bu gibi durumlarda daha büyük matrisler kullanmak gereklidir yada bir kaç kez aynı filtreden geçirmek gerekebilir.Bir diğer dezavantaj ise hesaplama zamanın uzun olmasıdır. Çünkü komşu piksellerin değerlerini alıp bunları sıralamaya tabi tutar. Bazı akıllı algoritmalar ile sıralama hızı artırılabilir.

Bizim projemizde kullandığımız ve resim bulanıklaştırma için yaygın kullanılan fonksiyonlardan biri de Gauss bulanıklaştırma fonksiyonudur.(Gaussian Blurring) Gauss farklı bir çekirdek şablon (matris) kullanır.Bu anlamda, ortalama filtreye benzer.Gauss'un resim düzgünleştirme etkisi, bir görüntüyü ortalama filtreye benzer şekilde bulanıklaştırmaktır.Düzeltebilme derecesi Gaussian'ın standart sapması ile belirlenir.Gauss, her piksel bölgesinin ağırlıklı ortalamasını çıkarır. Merkez piksel değerine doğru yaklaşıkça ağırlıklandırma artar. Bu durum, ortalama filtrenin aksine daha ince bir düzeltme sağlar, kenarları benzer büyüklükteki bir ortalama filtreden daha iyi korur. Ayrıca diğer filtreleme yöntemlerine göre daha hızlı olduğundan dolayı projemizde Gaussian Bulanıklaştırma kullanmayı tercih ettim.

# 6

## Uygulama

Markaların yerlerini tespit etmek için Faster R-CNN mimarisi kullanılmıştır. Bu aşamada Tensorflow'un object detection yapısından yararlanılmıştır. Böylece kullanılacak yapının daha kararları ve sonuçların daha iyi görselleştirilmesi sağlanmıştır.

### 6.1 Veri Kümesi

Sistem birçok nesneyi tespit edip bulanıklaştıracaktır. Bu nesneler bizim önceden bulduğumuz nesneler olacaktır. Projenin uygulanması için 17 tane nesne belirlenmiştir. Her bir nesne sınıfı için 300 tane resim toplanmıştır. Bunlardan 240'ı eğitim için kullanılacak(%80) ve 60'ı test için kullanılacaktır(%20).

**Tablo 6.1** Projede Kullanılan Sınıflar ve Resim Sayıları

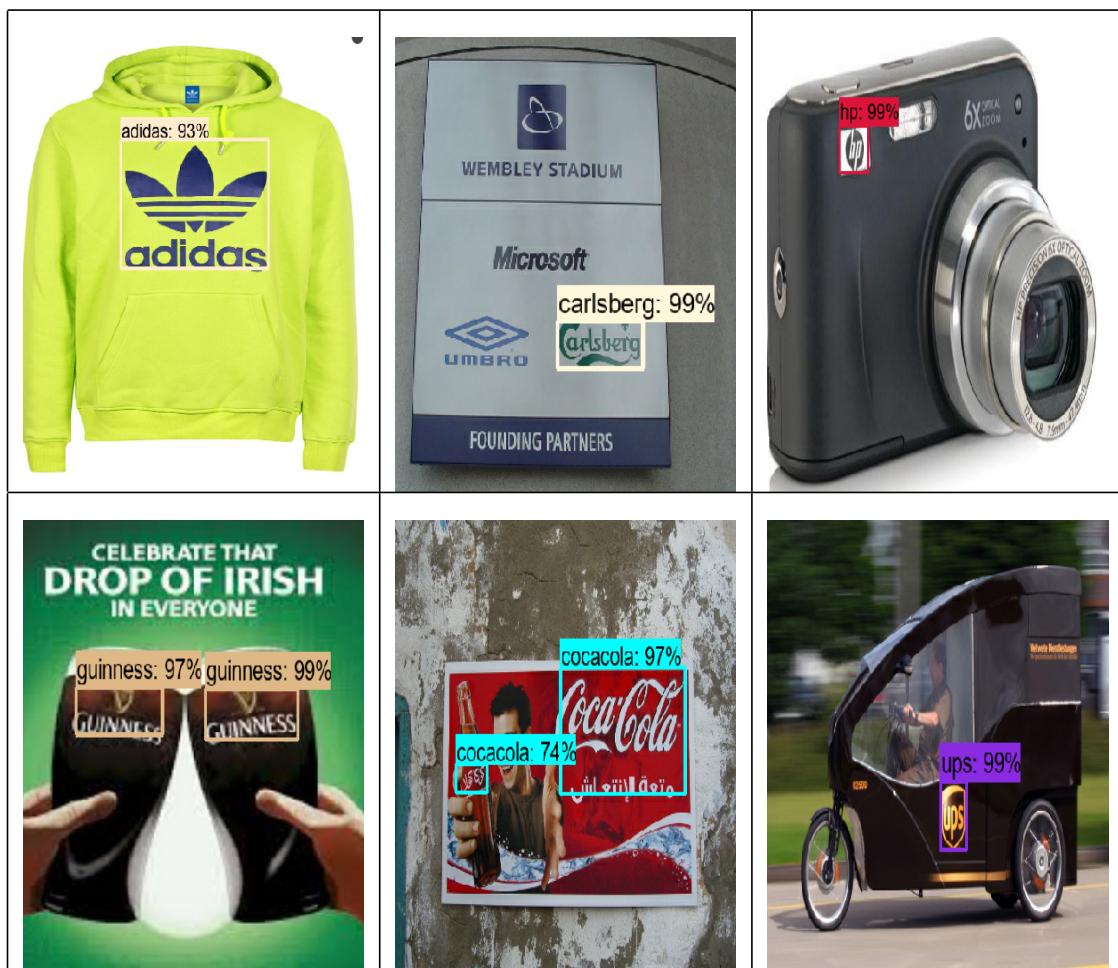
Sınıf	Eğitim için resim sayısı	Test için resim sayısı	Toplam
Apple	240	60	300
CocaCola	240	60	300
Dhl	240	60	300
Starbucks	240	60	300
Bmw	240	60	300
Volkswagen	240	60	300
Adidas	240	60	300
Ups	240	60	300
Guinness	240	60	300
Shell	240	60	300
Chimay	240	60	300
Ferrari	240	60	300
Fedex	240	60	300
Corona	240	60	300
Carlsberg	240	60	300
Hp	240	60	300
Erdinger	240	60	300

Bu sayılarla ulaşmak için bazı resimlere Data Augmentation yöntemi adı verilen yöntem uygulanmıştır. Bu yöntemde resimleri sağa-sola çevirmek, contrast ayarlarıyla oynamak ve resmi küçültüp büyütmek dışında başka bir şey yapılmamıştır. Bunu yapmamızın sebebi, bir sınıf için yeterli sayıda örneğe ulaşılmazsa sistemin başarı oranını düşüreceğindendir.

## 6.2 Başarılı ve Başarısız Durumlara Örnekler

Projemizin önemli bir kısmını oluşturan nesne tespiti bölümünde elde ettiğimiz başarılı ve başarısız sonuçlar aşağıdaki tablolarda gösterilmiştir. Görüşlerdeki nesnelerin daha rahat anlaşılmasını sağlamak için tespit edilen nesnelerin bulanıklaştırılmış hali kullanılmıştır.

**Tablo 6.2** Projede Başarılı Tespit Edilen Nesneler



Tablo-6.2, projemizde bulunan test resimlerinin uygulanmasında başarılı olan durumlar

**Tablo 6.3** Projede Başarısız Tespit Edilen Nesneler

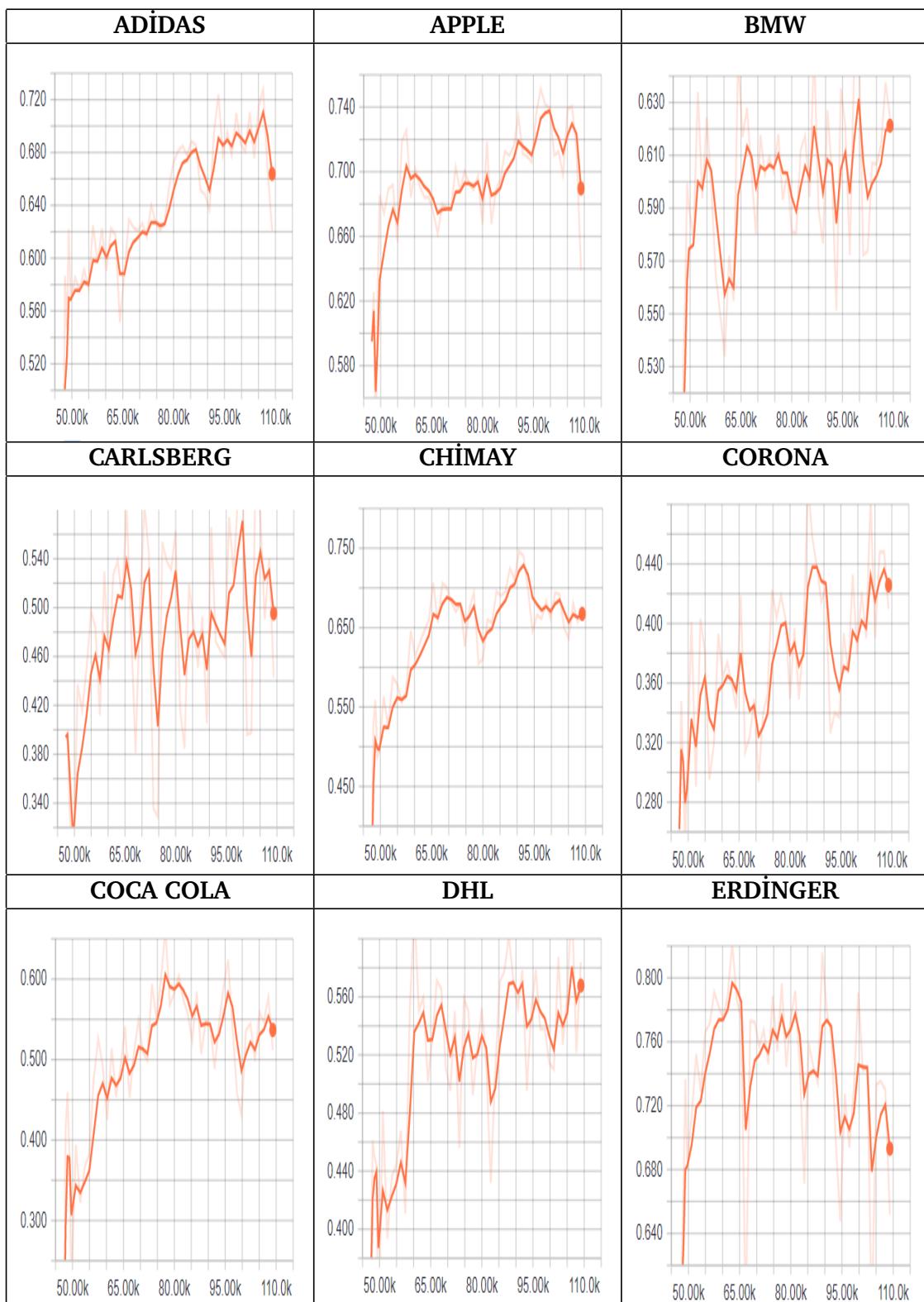


Tablo-6.3, projemizde bulunan test resimlerinin uygulanmasında başarısız olan durumlar

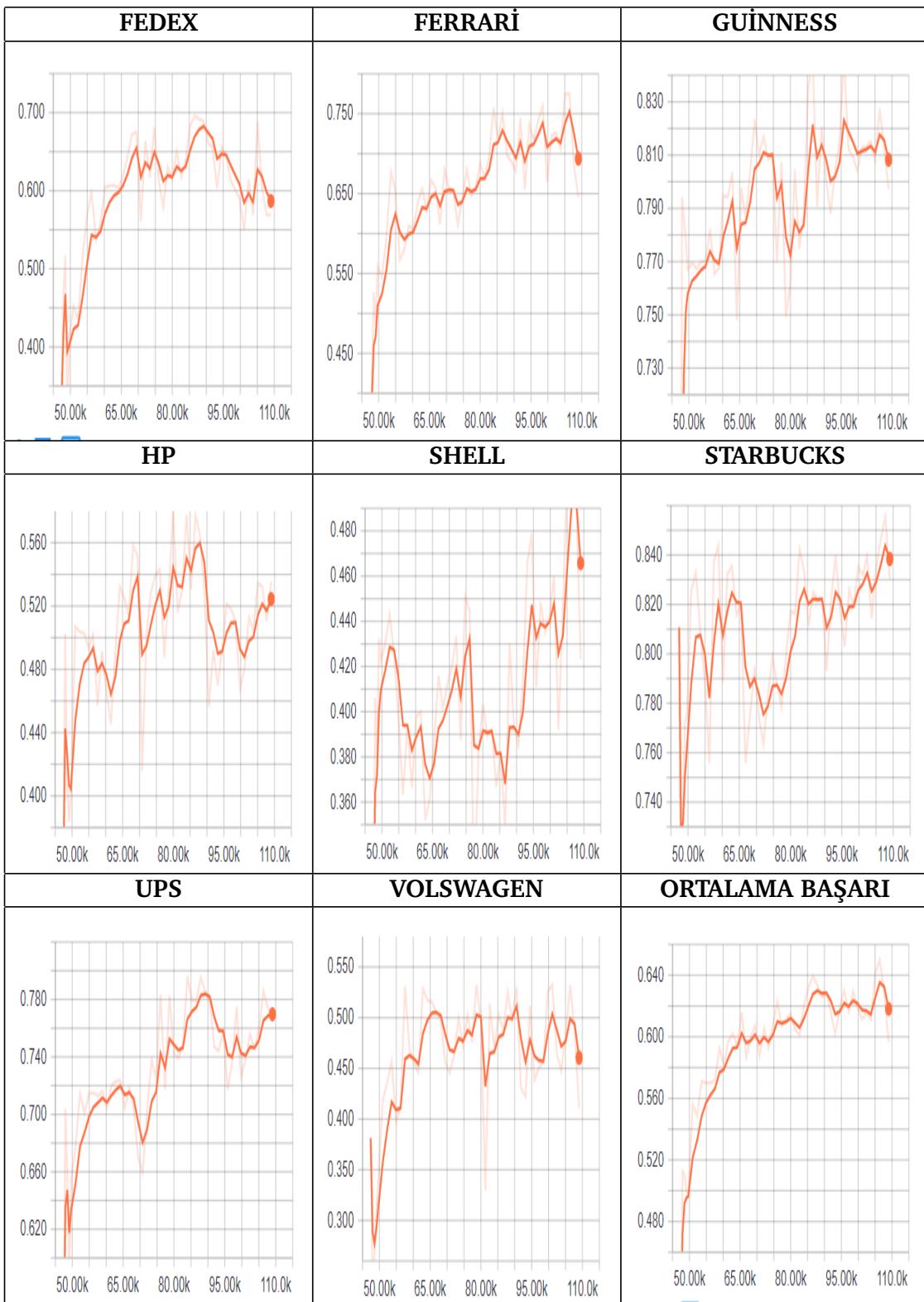
### 6.3 Deneysel Sonuçlar

Her bir sınıfın performansı ve toplam performansı saklanmıştır. Sistem 17 sınıf ve bu sınıflar için 110000(110k) adım eğitilmiştir. Deneysel sonuçlar Tensorboard’ın grafik arayüzünde gözlemlenmiştir. Ortalama kesinlik(precision) nesne tespiti alanında performans ölçüsüdür. Aşağıdaki resimlerde her bir sınıfın başarı oranları gösterilmiştir.

**Tablo 6.4** Projedeki Sınıfların Başarı Oranları-1



**Tablo 6.5 Projedeki Sınıfların Başarı Oranları-2**



## 6.4 Performans Analizi

Projede kullanılan modelin hem hızlı çalışması hem de tespit edilen nesnenin doğruluk yüzdesinin yüksek olmasını istediğimiz için nesne tespiti ağları olarak Faster R-CNN mimarisi kullandık. Model 2 farklı grafik kartı üzerinde denenmiştir. İlk model Nvidia GTX-960M (4GB) ta denenmiştir. Bir resmin süreçi 0.6 saniye sürmektedir. İkinci ve bizim son halini kullandığımız model Nvidia GTX-660M (2GB) ta bir resmin işlenmesi yaklaşık 1.1 saniye sürmektedir. Nvidia GTX-1660Ti gibi bir grafik kartı kullanılması üzerinde resmin işlenme zamanı 0,13'e kadar düşürülebilir.

Veri seti üzerinde çeşitli değişiklikler yapılarak bu zamanlar kısaltılabilir. Model 52 saatte eğitilmiştir.

Sisteme 22 saniyelik bir video verilip sistem test edildiğinde videonun tamamının işlenmesi için geçen süre yaklaşık 10 dakika olarak ölçülmüştür. Sistemin çıktıları aşağıdaki tabloda gösterilmiştir.

**Tablo 6.6** Sistemin Çıktıları Ve Çalışma Hızı

	Sisteme Verilen Video	Sistemden Alınan Video
<b>Uzunluk</b>	22 saniye	131 saniye
<b>Çerçeve Genişliği</b>	480	1280
<b>Çerçeve Yüksekliği</b>	848	720
<b>Veri Hızı</b>	1506 kb/sn	9216 kb/sn
<b>Toplam Bit Hızı</b>	1563 kb/sn	9216 kb/sn
<b>Resim Kare Hızı</b>	29.62 kare/saniye	5 kare/saniye
<b>Dosya Türü</b>	MP4	AVI

# 7 Sonuç

---

Bu projede videolarda gözükmesi istenmeyen markaların elle bulanıklaştırılmasının önüne geçilmeye çalışılmıştır. Bunu derin öğrenme ile yaparak hem zaman hem iş yükü azaltılmıştır. Öncelikle nesne tespiti için gerekli yapılar kodlanmış, ondan sonra 17 sınıfı dair veriler toplanmıştır. Bunlar yapıldıktan sonra kullanılıcak mimari olarak Faster R-CNN ağ yapısı seçilmiştir. Faster R-CNN iki ağdan oluşur. Birincisi regional proposal network(RPN), nesnenin resimde nerede olabileceğini hesaplayan ağ. İkincisi ise nesneyi bulanıklaştırması işini yapan bizim modellediğimiz ağ. "Custom size anchor boxes" yaklaşımı YOLO[9]dan esinlenmiştir. Bu yaklaşım Faster R-CNN'de bulunan RPN'nin özellikle eğitilmiş veri seti için daha iyi sonuçlar üretmesini sağlamıştır. Sistemin ortalama başarı oranı(Mean Average Precision) = %62 olarak ölçülmüştür. Son olarak kullanıcıların sistemi rahat kullanabilmesi için basit ve anlaşılır bir arayüz tasarlanmıştır.

Sonuç olarak bu sistem birçok film veya dizi yapımcısı tarafından kullanılabilir. Bu sistem sayesinde bilinçsiz yapılan ürün yerleştirmelerinin önüne kolaylıkla geçilebilir.

## Referanslar

---

- [1] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.
- [2] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” in *Advances in neural information processing systems*, 2015, pp. 91–99.
- [3] F. FOTSO, “Faster rcnn logo,” *GitHub repository*, 2018.
- [4] H. A. Alwzwazy, H. M. Albehadili, Y. S. Alwan, and N. E. Islam, “Handwritten digit recognition using convolutional neural networks,” *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 4, no. 2, pp. 1101–1106, 2016.
- [5] J. Hung and A. Carpenter, “Applying faster r-cnn for object detection on malaria images,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 56–61.
- [6] S. Zhang, L. Wen, X. Bian, Z. Lei, and S. Z. Li, “Occlusion-aware r-cnn: Detecting pedestrians in a crowd,” in *The European Conference on Computer Vision (ECCV)*, Sep. 2018.
- [7] A. Anuar, K. M. Saipullah, N. A. Ismail, and Y. Soo, “Opencv based real-time video processing using android smartphone,” *International Journal of Computer Technology and Electronics Engineering (IJCTEE)*, vol. 1, no. 3, 2011.
- [8] H. Su, X. Zhu, and S. Gong, “Open logo detection challenge,” in *British Machine Vision Conference*.
- [9] J. Redmon and A. Farhadi, “Yolov3: An incremental improvement,” *arXiv preprint arXiv:1804.02767*, 2018.

# Özgeçmiş

---

## BİRİNCİ ÜYE

**İsim-Soyisim:** Özgür KAN  
**Doğum Tarihi ve Yeri:** 04.08.1994, İstanbul  
**E-mail:** ozgur.kan@std.yildiz.edu.tr  
**Telefon:** 0534 396 64 45  
**Staj Tecrübeleri:**

## İKİNCİ ÜYE

**İsim-Soyisim:** Eray CINCI  
**Doğum Tarihi ve Yeri:** 04.07.1997, Prishtina  
**E-mail:** cincieray@gmail.com  
**Telefon:** 0553 173 97 11  
**Staj Tecrübeleri:**

## Proje Sistem Bilgileri

**Sistem ve Yazılım:** Windows İşletim Sistemi, Python, Cuda, cuDNN, Anaconda  
Virtual Enviroment  
**Gerekli RAM:** 8GB  
**Gerekli Disk:** 100GB