



**MARMARA ÜNİVERSİTESİ**  
**FEN BİLİMLERİ ENSTİTÜSÜ**



# **İNSANSIZ HAVA ARACI İLE ELDE EDİLEN GÖRÜNTÜLERİN DERİN ÖĞRENME YÖNTEMLERİ İLE ANALİZİ**

---

**ÖZGÜR KUTLU**

**YÜKSEK LİSANS TEZİ**

Elektronik-Bilgisayar Eğitimi Anabilim Dalı  
Bilgisayar-Kontrol Eğitimi Programı

**DANIŞMAN**

Dr. Öğr. Üyesi Önder DEMİR

**EŞ-DANIŞMAN**

Dr. Barış DOĞAN

**İSTANBUL, 2019**

---



**MARMARA ÜNİVERSİTESİ**  
**FEN BİLİMLERİ ENSTİTÜSÜ**



# **İNSANSIZ HAVA ARACI İLE ELDE EDİLEN GÖRÜNTÜLERİN DERİN ÖĞRENME YÖNTEMLERİ İLE ANALİZİ**

---

**ÖZGÜR KUTLU**

522103981

**YÜKSEK LİSANS TEZİ**

Elektronik-Bilgisayar Eğitimi Anabilim Dalı

Bilgisayar-Kontrol Eğitimi Programı

**DANIŞMAN**

Dr. Öğr. Üyesi Önder DEMİR

**EŞ-DANIŞMAN**

Dr. Barış DOĞAN

**İSTANBUL, 2019**

---



## **ÖNSÖZ**

Tez çalışmam boyunca benden bilgisini, desteğini ve sabrını esirgemeyen hocalarım Dr. Öğr. Üyesi Önder Demir'e ve Dr. Barış Doğan'a

Tüm çalışmalarım sırasında verdikleri destek ve sabırları için sevgili eşim Yasemin ve çocuklarım Ahmet Alp ve Birce'ye, beni yüreklendiren Babam ve Anneme, fikir ve yönlendirmeleriyle yardımcı olan arkadaşlarıma ve hocalarıma,

Teşekkürü bir borç bilirim.

**Haziran, 2019**

**Özgür KUTLU**

# İÇİNDEKİLER

ÖNSÖZ .....	i
İÇİNDEKİLER .....	ii
ÖZET.....	iv
ABSTRACT.....	v
SEMBOLLER .....	vi
KISALTMALAR .....	vii
ŞEKİL LİSTESİ.....	viii
TABLO LİSTESİ.....	x
1. GİRİŞ .....	1
1.1. İnsansız Hava Araçlarının Kullanım Alanları .....	1
1.1.1. İnsansız hava araçlarının çeşitleri.....	2
1.1.2. İnsansız hava araçlarının yapısı.....	2
1.1.3. Döner kanat insansız hava aracını oluşturan bileşenler.....	3
1.2. Derin Öğrenme Nedir? .....	3
1.3. İnsansız Hava Araçları Kullanarak Yapılmış Akademik Çalışmalar .....	5
2. MATERYAL VE YÖNTEM .....	7
2.1. Kullanılan Veri Seti.....	7
2.2. Veri Etiketleme .....	9
2.3. Derin Öğrenme Mimarileri.....	10
2.3.1. Evrişimsel sinir ağları (Convolutional Neural Network, CNN) .....	10
2.3.2. Tekrarlayan sinir ağları (Recurrent Neural Network, RNN).....	11
2.3.3. Uzun kısa vadeli hafıza ağları (Long Short-Term Memory, LSTM) .....	11
2.4. Evrişimsel Sinir Ağlarının Yapısı .....	11
2.4.1. Evrişim katmanı .....	12
2.4.2. Kaydırma Adımı.....	14
2.4.3. Dolgulama .....	15
2.4.4. Aktivasyon fonksiyonu .....	15
2.4.5. Havuzlama katmanı(Pooling).....	16

2.4.6.	Tamamen bağı katman(Fully Connected Layer).....	17
2.5.	Derin Öğrenme Kütüphaneleri .....	17
2.6.	Bölgesel Önerimli Evrişimsel Sinir Ağları (R-CNN) .....	18
2.7.	Faster R-CNN Sinir Ağları.....	20
2.8.	Derin Öğrenme Modelleri .....	22
2.8.1.	Yaygın olarak kullanılan evrişimli sinir ağı tasarımları .....	22
2.8.2.	ResNet Mimarisi .....	23
2.8.3.	GoogLeNet .....	26
2.9.	Geliştirilen Sistemin İşleyişi .....	31
2.9.1.	Etiketlenmiş veri dönüşümü.....	32
2.9.2.	Etiket Haritasının Oluşturulması .....	33
2.9.3.	Model konfigürasyon dosyası ayarları .....	33
1.1.1.	Model eğitimi gerçekleştirme.....	34
1.1.2.	Çıkarım grafiği oluşturma .....	34
1.1.3.	Eğitilen modelin test edilmesi .....	35
3.	DENEYSEL ÇALIŞMALAR .....	37
3.1.	Hata Matrisi.....	38
3.2.	Kesinlik .....	39
3.3.	Duyarlılık .....	39
3.4.	Birleşmede Üzerinde Kesişim(IoU) .....	39
3.5.	Kesinlik-Duyarlılık Eğrisi .....	40
3.6.	Ortalama Hassasiyet.....	40
3.7.	Deney Sonuçları .....	41
4.	SONUÇLAR .....	49

## ÖZET

### İNSANSIZ HAVA ARACI İLE ELDE EDİLEN GÖRÜNTÜLERİN DERİN ÖĞRENME YÖNTEMLERİ İLE ANALİZİ

Günümüz insansız hava aracı (İHA) teknolojisi savunma sanayii, eğlence ve film endüstrisi, taşımacılık(paket teslimatı) gibi çok farklı sektörlerde yer edinmiş durumdadır. Savunma ve havacılık sektörüne yönelik araştırmalar yapan Teal Group'un 16 Temmuz 2018 tarihli raporunda Sivil İnsansız Hava Sistemlerinin (UAS) rekor düzeyde girişim sermayesi çektiği ve bu on yılda dünya havacılık endüstrisinin en dinamik büyüme sektörü olacağı belirtiliyor. İnsansız hava araçlarının kontrol ve yönlendirilmesinde kullanılacak akıllı sistemler ise yapay zeka çalışmalarıyla yakından ilişkilidir. Son yıllarda yapay zekâ algoritmaları en çok araştırma yapılan ve uygulama geliştirilen çalışma alanlarından biridir.

Yapay zekâ uygulamalarında makine öğrenmesi, yapay sinir ağları, sınıflandırma, kümeleme algoritmaları gibi birçok alt yöntem kullanılmaktadır. Bu yöntemlerden biri de derin öğrenmedir. Derin öğrenme gelişmiş bir makine öğrenmesi sınıfıdır. Derin Öğrenme yöntemi kullanılarak Video analizi, görüntü sınıflandırma, konuşma tanıma ve doğal dil işleme gibi alanlarda çok başarılı sonuçlar elde edilmektedir. Derin Öğrenme ve İnsansız hava araçları alanlarını kapsayacak projelerin sağlayacağı veriler ve tecrübeler, bu konularda yapılan nitelikli çalışmaların sayısını arttırarak ülkemizin bu teknolojilerde katma değeri yüksek ürünler geliştirilmesine katkı sağlayacaktır. Bu çalışmada, insansız hava araçlarından alınacak görüntü verisini değerlendiren ve çeşitli çıkarımlar (Sınıflandırma, konumlandırma, işaretleme) yapan bir kontrol yazılımı oluşturulmuştur. Önceden eğitilmiş yapay sinir ağı modellerinin son katmanlarının veri setimizle tekrar eğitilmesi yöntemi kullanarak eğitim süresi azaltılmaya ve başarı arttırılmaya çalışılmıştır. Yapılan çalışmalarda 2 ön eğitilmiş model kullanılmış bu modellerin eğitilmesi sonucu yapılan testlerde 190 bin adımlık eğitim sonucunda mAP değerleri olarak 25.39 ve 27.87 değerlerine ulaşılmıştır.

**Haziran, 2019**

**Özgür KUTLU**

## **ABSTRACT**

### **ANALYSIS OF IMAGES OBTAINED BY UNMANNED AERIAL VEHICLE BY DEEP LEARNING METHODS**

Today's unmanned aerial vehicle (UAV) technology has taken place in many different sectors such as defense industry, entertainment and film industry, transportation (package delivery). Teal Group's research on the defense and aeronautics sector, July 16, 2018 report, Civil Unmanned Air Systems (UAS) is a record level of venture capital and the world's most dynamic growth sector in this decade is stated to be the most dynamic. Intelligent systems for controlling and directing unmanned aerial vehicles are closely related to artificial intelligence studies. In recent years, artificial intelligence algorithms are one of the most researched and developed applications.

In artificial intelligence applications, many sub-methods such as machine learning, artificial neural networks, classification, clustering algorithms are used. One of these methods is deep learning. Deep learning is an advanced machine learning class. Using the Deep Learning method, video analysis, image classification, speech recognition and natural language processing are very successful. The data and experiences to be provided by the projects covering Deep Learning and Unmanned Aerial Vehicles will increase the number of qualified studies on these issues and contribute to the development of high value-added products in these technologies. In this study, a control software that evaluates image data from unmanned aerial vehicles and makes various inferences (classification, positioning, marking) is created. By using the method of retraining the last layers of the pre-trained artificial neural network models with our data set, it has been tried to reduce the training time and increase the success. In these studies, 2 pre-educated models were used and as a result of training of these models, as a result of 190 thousand steps of training, 25.39 and 27.87 mAP values were reached.

**June, 2019**

**Özgür KUTLU**



## **SEMBOLLER**

**AP** : Average Precision

**d** : Dimension

**h** : Height

**mAP** : mean Average Precision

**w** : Width

## KISALTMALAR

<b>ANN</b>	: Artificial Neural Network
<b>CNN</b>	: Convolutional Neural Network
<b>DNN</b>	: Deep Neural Network
<b>DP</b>	: Doğru Pozitif
<b>YP</b>	: Yanlış Pozitif
<b>DN</b>	: Doğru Negatif
<b>YN</b>	: Yanlış Negatif
<b>FC</b>	: Fully Connected
<b>GPU</b>	: Graphic Processing Unit
<b>HOG</b>	: Histogram of Oriented Gradients
<b>IoU</b>	: Intersection Over Union
<b>İHA</b>	: İnsansız Hava Aracı
<b>LSTM</b>	: Long Short Term Memory
<b>NLP</b>	: Natural Language Processing
<b>R-CNN</b>	: Regions with CNN
<b>ReLU</b>	: Rectified Linear Units
<b>RNN</b>	: Recurrent Neural Networks
<b>RRPN</b>	: Rotation Region Proposal Networks
<b>SBC</b>	: Single Board Computer
<b>SİHA</b>	: Silahsız İnsansız Hava Aracı
<b>SSD</b>	: Single Shot MultiBox Detector
<b>SVM</b>	: Support Vector Machine
<b>UAV</b>	: Unmanned Aerial Vehicles
<b>YOLO</b>	: You Only Look Once

## ŞEKİL LİSTESİ

Şekil 1.1. Döner Kanat İnsansız Hava Aracı.....	1
Şekil 1.2 Yüz Tanıma Problemi Başarı Oranları.....	4
Şekil 2.1 Sensefly Data Setinden Bir Örnek.....	8
Şekil 2.2. COCO Veri Setinden Bir Örnek.....	9
Şekil 2.3. labelImg Programı.....	9
Şekil 2.4. RGB Görüntü Matrisi.....	11
Şekil 2.5. CNN İş Akışı.....	12
Şekil 2.6. Evrişim Katmanı.....	12
Şekil 2.7. Evrişim ve Filtre.....	13
Şekil 2.8. Özellik Haritasının Oluşumu.....	13
Şekil 2.9. Filtre Çeşitleri.....	14
Şekil 2.10. Kaydırma Adımı.....	15
Şekil 2.11. ReLU Aktivasyon Fonksiyonu.....	16
Şekil 2.12. Maksimum Havuzlama.....	16
Şekil 2.13. Tamamen Bağlı Katman Yapısı.....	17
Şekil 2.14. Seçici Arama Algoritması Aşamaları.....	19
Şekil 2.15. Bölgesel Önerimli Evrişimsel Sinir Ağ İş Akışı.....	19
Şekil 2.16. R-CNN.....	20
Şekil 2.17. Faster R-CNN Yapısı.....	21
Şekil 2.18. Nesne algılama algoritmalarının karşılaştırılması (Sn).....	21
Şekil 2.19. ILSVRC Yarışması Yıllara Göre Birinciler.....	23
Şekil 2.20. CNN ile Residual bağlantılı CNN karşılaştırması.....	24
Şekil 2.21. ResNet Mimarisi.....	25
Şekil 2.22. Farklı Büyüklüklerde Nesneler.....	26
Şekil 2.23. Inception Modülü.....	27
Şekil 2.24. Boyut Küçültülmüş Inception Modülü.....	28
Şekil 2.25. GoogLeNet(Inception v1) .....	29
Şekil 2.26. Inception v2 Modülü.....	30
Şekil 2.27. Yatay Genişletilmiş Inception v2 Modülü.....	31
Şekil 2.28. Nesne Tanıma İşlem Sırası.....	32
Şekil 2.29. Etiket haritası dosyası içeriği.....	33
Şekil 2.30. Konfigürasyon dosyası.....	34
Şekil 2.31. Object_Detection_Image.py Dosyası.....	35
Şekil 2.32. Örnek Çıkarım Görüntüsü.....	36
Şekil 3.1. Eval_image Dosya Çıktısı.....	37
Şekil 3.2. Birleşmede Üzerinde Kesişim(IoU) .....	39
Şekil 3.3. Kesinlik-Duyarlılık Eğrisi.....	40
Şekil 3.4. Faster_rcnn_inception_v2_coco Toplam Kayıp.....	41
Şekil 3.5. Faster_rcnn_inception_v2_coco Sınıflandırma Kaybı.....	41
Şekil 3.6. Faster_rcnn_inception_v2_coco Konumlandırma Kaybı.....	42
Şekil 3.7. Faster_rcnn_inception_v2_coco modelinin Kesinlik-Duyarlılık Eğrileri.....	43
Şekil 3.8. Faster_rcnn_inception_v2_coco modeli Ortalama Kesinlik Değerleri.....	44
Şekil 3.9. Faster_rcnn_resnet101_coco Toplam Kayıp.....	44
Şekil 3.10. Faster_rcnn_resnet101_coco Sınıflandırma Kaybı.....	45
Şekil 3.11. Faster_rcnn_resnet101_coco Konumlandırma Kaybı.....	45

<b>Şekil 3.12.</b> Faster_rcnn_resnet101_coco modelinin Kesinlik-Duyarlık Eğrileri.....	47
<b>Şekil 3.13.</b> Faster_rcnn_resnet101_coco modeli Ortalama Kesinlik Değerleri.....	48

## TABLO LİSTESİ

<b>Tablo 2.1.</b> Sensefly Data Setleri.....	8
<b>Tablo 2.2.</b> Nesne Sınıfları-Etiket Sayıları.....	10
<b>Tablo 2.3.</b> Derin Öğrenme Kütüphaneleri.....	18
<b>Tablo 2.4.</b> Farklı ResNet Katman Mimarileri.....	26
<b>Tablo 2.5.</b> Kullanılan Model Özellikleri.....	32
<b>Tablo 3.1</b> Hata Matrisi.....	38
<b>Tablo 3.2.</b> Faster_rcnn_inception_v2_coco Ortalama Kesinlik Değerleri.....	42
<b>Tablo 3.3.</b> Faster_rcnn_resnet101_coco Ortalama Kesinlik Değerleri.....	45
<b>Tablo 3.4.</b> Test edilen Modellerin Sınıflara Göre Başarıları.....	48
<b>Tablo 4.1.</b> Model mAP Değerleri.....	49

# 1. GİRİŞ

İnsansız Hava Araçları (İHA, Unmanned Aerial Vehicle-UAV), uzaktan kontrol edilebilen bir tür uçak teknolojisi olup ilk çıktığında gözlem ve savunma gibi amaçlarla kullanılmıştır. İnsansız hava araçları yüksek manevra kabiliyetleri, olduğu yerden dikine kalkış/iniş yapabilmeleri, açık ve kapalı alanda kullanılabilmeleri sayesinde son yıllarda askeri, sivil ve ticari birçok alanda yaygınlaşan uygulama alanına sahip popüler bir teknoloji alanıdır.



**Şekil 3.1.** Döner Kanat İnsansız Hava Aracı.[1]

Yakın geçmişte mikro işlemci dünyasında yaşanan gelişmeler, yüksek işlem gücünü düşük enerji tüketimi ile birleştirerek taşınabilir cihazlarda kullanılabilir hale getirmiştir. Kendi kendine gidebilen otomobiller, model kara araçları, model deniz araçları ve insansız hava araçları bu küçük ve güçlü elektronik kartların başlıca kullanım alanlarıdır. Özellikle insansız hava araçları gibi genellikle batarya ile çalışan, ağırlık ve enerji tüketiminin ön planda olduğu cihazlarda kullanılabilecek SBC(Single Board Computer) ekipmanlar araçların kalkış, iniş, rota izleme, kontrol ve görev süreçlerini insan kontrolüne gerek kalmadan otonom olarak yapabilmelerine olanak sağlamıştır. Yüksek işlem gücü sayesinde makine öğrenmesi yöntemlerinden birisi olan yapay sinir ağları ANN (Artificial Neural Network) ve çok katmanlı mimari model DNN (Deep Neural Network) hava ve kara araçlarında kullanılabilmektedir.

## 1.1. İnsansız Hava Araçlarının Kullanım Alanları

Günümüzde savunma sanayi firmalarınca üretilmiş İha ve silahlı insansız hava araçları (SİHA) askeri ve istihbari amaçlarla aktif olarak kullanılmaktadır. Askeri alandaki başarılı uygulamalarla İHA'ların ülke savunmamıza önemli derecede başarılı etkileri olduğu gözlemlenmektedir. Ülkemiz için bir teknolojik sıçrama noktası olabilecek İHA

teknolojisinin ticari amaçlarla kullanımına hizmet edecek yazılım ve donanımların geliştirilmesi de hayati önem taşımaktadır. Bu kapsamda İHA'lar özel şirketler, üniversiteler ve bireysel girişimciler tarafından günlük alanlardaki kullanımlara da adapte edilmeye çalışılmaktadırlar. Havadan görüntüleme ve haritalama, yangın söndürme, ilk yardım ve can kurtarma, tarımsal ilaçlama, enerji nakil hatlarının bakım ve kontrolü, küçük ölçekli kargo taşıma gibi amaçlarla kullanılan İHA'lar bunlara örnektir.

#### **1.1.1. İnsansız hava araçlarının çeşitleri**

Temel olarak insansız hava araçları döner kanatlı, sabit kanatlı ve hibrit olmak üzere 3 ana gruba ayrılabilir. Aracı havada tutan kanatları hareketsiz ve sabit olan İHA'lara sabit kanatlılar denir. Sabit kanatlı İHA'ların havada kalabilmeleri gövdenin sürekli hareket etmesine bağlıdır. Hareket içten yanmalı motor ya da elektrik motoru kullanılarak sağlanır. Genellikle tek motorlu olan bu araçların üretim maliyetleri diğer İHA modellerine göre düşüktür. İniş kalkış için geniş alanlara ihtiyaç duyulması sabit kanatlı İHA'ların dezavantajı olmakla birlikte uçuş menzillerinin oldukça uzun olması en büyük avantajlarıdır. Aracı havada tutan pervane kanatları yerçekimi doğrultusuna zıt yönde ve sürekli dönen İHA'lara ise döner kanatlılar denmektedir. Sahip oldukları pervane sayısına göre Helicopter, Tricopter, Quadcopter, Hexacopter, Octocopter gibi isimler alırlar. Döner kanatlılarda gövde sabit olup kanatlar döndüğünden sürekli hareket etme zorunluluğu yoktur. Bu sayede döner kanatlıların havadaki hareketleri daha kontrollü olup havada asılı kalabilir ve küçük alanlara iniş kalkış yapabilirler. Döner kanatların üretim ve tasarımında yük batarya planlaması ve işçilik çok önemlidir. Döner kanat sayısına göre motor, motor sürücü, batarya sayısı arttığı için üretim maliyetleri sabit kanatlıya göre yüksektir. Uçuş menzilleri kısadır. Hibrit İHA'lar hem döner kanat hem sabit kanat İHA'ların avantajlarını kullanabilmek amacıyla tasarlanmış araçlardır.

#### **1.1.2. İnsansız hava araçlarının yapısı**

İnsansız hava araçlarının tasarımları kullanım amacına bağlı olarak değişmektedir. Yarışma, gözetleme, ilaçlama, savunma ve eğlence amaçlı geliştirilmiş İHA'lar farklı büyüklük ve teknik özelliklere sahiptir. Bu tez çalışmasında havada sabit kalabilme özelliği sayesinde kaliteli fotoğraf ve video çekebilen döner kanat insansız hava araçları

üzerinde bulunan kameralardan elde edilen görüntüler kullanılmıştır.

### **1.1.3. Döner kanat insansız hava aracını oluşturan bileşenler**

Döner kanat insansız hava aracı genel olarak şu bileşenlerden oluşur:

- Gövde
- Motor
- Motor sürücü
- Uçuş Denetleyicisi
- Güç Dağıtıcısı
- OSD Modülü,
- Pervane
- Batarya ve Batarya Alarmı,
- Görüntü alınmak isteniyorsa Video Kamera

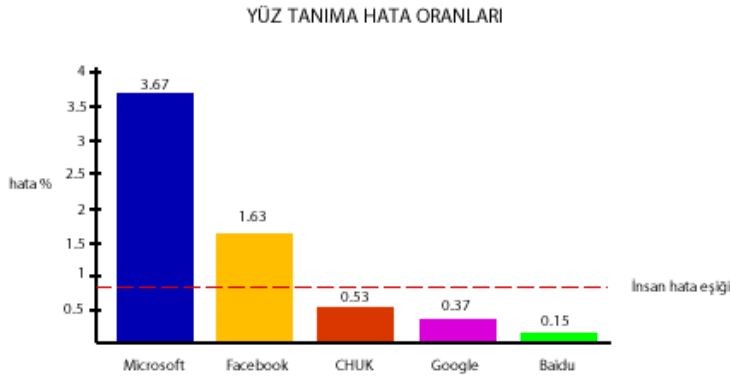
Bu bileşenlerin dışında İHA'lara eklenecek tek kartlı bilgisayar (SBC) ile sağlanan işlem gücü sayesinde İHA üzerinden alınacak görüntü kullanılarak gerçek zamanlı çalışabilen birçok bilgisayarlı görü uygulaması gerçekleştirilebilir.

## **1.2. Derin Öğrenme Nedir?**

Son yıllarda yapay zekâ üzerine en çok araştırma yapılan ve uygulama geliştirilen çalışma alanlarından biridir. Bunun sebebi bilim insanları ve endüstride çalışan geliştiricilerin ürettikleri ürünlerde ve yazılımlarda yapay zekâ teknolojilerini her geçen gün daha fazla kullanması ve kullanılan bu teknolojinin insanların hayatını kolaylaştırmasıdır. Yapay zekâ uygulamalarında makine öğrenmesi, yapay sinir ağları, sınıflandırma, kümeleme algoritmaları gibi birçok alt yöntem kullanılmaktadır. Bu yöntemlerden biri de derin öğrenmedir. Derin öğrenme gelişmiş bir makine öğrenmesi sınıfıdır. Öznitelik çıkarımı işlemini gizli katmanlar içerisinde etkin olarak kendi yapan katmanlı bir yapıya sahiptir. Her katman bir önceki katmandaki çıktıyı girdi olarak alır[2]. Derin öğrenme temel olarak veriyi tanımlayan seçilmiş özniteliklerden öğrenmeye dayalıdır. Bir görselde piksel yoğunluk vektörü, kenar kümeleri gibi öznitelikler temsil için örnek olarak düşünülebilir. Bu özniteliklerden bazıları veriyi diğerlerinden daha iyi temsil edebilir. Derin öğrenme ile en iyi öznitelikleri derecelendirebilecek genetik algoritmalar, temel bileşen analizi gibi yöntemler kullanılabilir[3]. Derin öğrenme uygulamaları için yüksek işlem gücü gerektiğinden



GPU donanımına sahip bilgisayarların kullanılması gereklidir. GPU donanımlarının özellikleri geliştikçe çok daha başarılı uygulamalar gerçekleştirilebilmektedir. Video analizi, görüntü sınıflandırma, konuşma tanıma ve doğal dil işleme gibi alanlarda çok başarılı sonuçlar elde edilmektedir. Yakın gelecekte derin öğrenme uygulamalarının daha da gelişerek yaygınlaşacağı öngörülmektedir. Bilgisayarlı görü bugün derin öğrenmenin kullanıldığı en önemli alanlardan biridir. Günümüzde hayatın her alanında artan kamera sayısı, kalitesi ve kullanımı dikkate alındığında, görüntü içindeki nesnelerin birbiriyle olan ilişkisinin bir insan gibi makineler tarafından anlamlı bir şekilde ortaya konması görüntüleri yorumlama konusunda kolaylık sağlamaktadır. Bu yöntemler sayesinde sayısal görüntülerin ve videoların bilgisayarlar tarafından bir insanın nesne tanıma seviyesinin üzerinde bir başarıyla değerlendirilebildiği Şekil 1.2’de görülmektedir.



**Şekil 1.2** Yüz Tanıma Problemi Başarı Oranları

Ciresan MNIST, CIFAR10 veri kümesi üzerinde yaptığı çalışmada çok sütunlu derin sinir ağı kullanarak görüntü sınıflandırma işleminde hata oranını en son çalışmalara göre %41 ve %39 oranında azaltarak MNIST için %0,23, CIFAR10 için %11,21 'e kadar düşürmeyi başarmıştır.[4]

2010 yılından beri her yıl yapılan ImageNet Large Scale Visual Recognition Challenge (ILSVRC) yarışmasının 2014 yılı mücadelesinde başarılı ekipler CNN algoritmalarının türevlerini kullanmıştır[5]. CNN algoritmaları görüntü işleme dışında NLP alanında da başarılı sonuçlar elde etmiştir. Evrimsel sinir ağları anlamsal ayrıştırma [6], cümle modelleme [7], sınıflandırma [8], tahmin problemleri [9] gibi alanlarda da kullanılmıştır. Google DeepMind tarafından CNN kullanılarak geliştirilen AlphaGo Go

oyununda profesyonel bir oyuncuyu Mart 2016 da yenmiştir [10].

Graves ve Jaitly ses verilerinin fonetik sunumuna gerek kalmadan metne çeviren bir uygulamayı RNN kullanılarak %6,7 lik hatayla gerçekleştirmiştir [11]. Karpaty ve Fei-Fei çalışmalarında RNN ve CNN birlikte kullanılarak verilen görselde nesnelerin tanınması ve konumlarının bulunmasını gerçekleştirmiştir.[12]

Çerçeve tabanlı ses sınıflandırma çalışmasında TIMIT veri kümesi kullanılarak %70 başarı oranı elde edilmiştir[13]. LSTM kullanarak Blues türünde besteler yapabilen model geliştirilmiştir.[14]

### **1.3. İnsansız Hava Araçları Kullanarak Yapılmış Akademik Çalışmalar**

Sarıbaş, Çevikalp ve Kahvecioğlu İHA'lerden alınan görüntülerdeki araçların konumlarını, gerçek zamanlı olarak bulabilen 8 - 4 kök sezicili EPCC, YOLOv2 ve YOLO Tiny algoritmalarını kullanan yöntemleri ile yüksek başarımlar(%84,49) elde etmiştir[15]. Jinwang Wang ve arkadaşları alçak irtifada uçan İHA ile elde edilmiş görüntülerden doğaya atılmış plastik şişeleri bulan ve konumlandıran bir uygulama gerçekleştirmişlerdir. Oluşturdukları veri seti Faster R-CNN, SSD, YOLOv2 ve RRPN gibi algoritmalarla test etmiş ve bu data set için RRPN (Rotation Region Proposal Networks) algoritmasının en yüksek konumlandırma doğruluğunu(%88.6) ve en düşük yanlış keşif oranını verdiğini gözlemlemişlerdir[16]. Joshua Kaster ve arkadaşları küçük bir insansız hava sisteminin yük boşaltma alanını kullanırken araç tespitini Faster R-CNN algoritması, etkili bir hava görüntü veri seti, süper bilgisayarlar ve özel bir ekip ile gerçek zamanlı olarak gerçekleştirmiştir[17]. Junjie Chen ve arkadaşları devriye görevi yapan İHA'ların inişini otomatikleştirme çalışmasında Faster R-CNN algoritması kullanmışlardır. Çalışmada iniş alanı tanımlama hızında 81 mili saniye ve doğru tanımda 97,8% başarı elde etmişlerdir[18]. Nguyen Cao Tri ve arkadaşları çeltik tarlalarının verim değerlendirmesinde İHA kullanmışlar elde ettikleri alçak irtifa görüntülerini kullanarak çeltik tarlasını 3 sınıfa ayırmış ve bu sınıfların geçmiş veri istatistiklerine bağlı olarak verim hesaplaması yapmışlardır[19]. Maria Romero ve arkadaşları üzüm bağlarının su ihtiyacının belirlenmesi ve sulama planlaması yapılmasında İHA'larla elde edilen multispektral görüntüleri yapay sinir ağlarında eğiterek kullanmışlardır[20]. Jian-Shu Zhang ve arkadaşları trafik akışı izlemede İHA

kullanmışlardır. Faster-RCNN algoritması kullanarak eğittikleri modellerle havadan elde edilen görüntülerde araçları tespit etmiş ve izlemişler böylece trafik yoğunluğu ile ilgili değerlendirmeler yapmışlardır. Araç tespitinde %96,5 doğruluk değerine trafik durum tespitinde %92,7 doğruluk değerine ulaşmışlardır[21]. Haodi Yao ve arkadaşları İHA'dan elde ettikleri görüntüde hareket eden hedefin pozisyonunu ve hareket yönünü Fully Convolutional Network kullanarak tespit etmiş ve Kalman filtresi sayesinde doğruluğu arttırmışlardır[22]. Dinale Zhou ve arkadaşları uçak gemilerine İHA'ların güvenli inişi için deniz durumunu, dalga hareketlerini, güverte hareketlerini ve rüzgar hareketlerini derin öğrenme ile modelleyen bir simülasyon üzerinde çalışmıştır[23]. Tianyu Tang ve arkadaşları İHA'larla elde ettikleri görüntüler üzerinde gerçek zamanlı araç tespitini derin öğrenme yönteminin Yolov2 algoritmasını kullanarak gerçekleştirmişler ve % 77,12 mAP – 0,048 saniye / kare değerlerine ulaşmışlardır[24]. Georgy V. Konoplich ve arkadaşları da İHA ile elde edilen görüntüler üzerinden gerçek zamanlı araç tespiti üzerine çalışmışlar, bu amaçla HDNN(Adapted Hybrid Neural Network) kullanmışlardır. HDNN in diğer denenen yöntemlerden (DNN, HOG+SVM, LBP+SVM, Adaboost) daha başarılı sonuçlar (%15,5 yanlış tespit oranı) verdiğini gözlemlemişlerdir[25]. Nicolas Rey ve arkadaşları Kuzikus tabiat parkında yaban hayatı gözlemek memeli ve kuşları sayarak vahşi yaşamın korunmasına katkıda bulunmak için İHALardan alınan görüntüleri makine öğrenmesi metotları ile işlemiştir[26]. A. Ellenberg ve arkadaşları köprü gövdelerinde bulunan bozulmaların tespitinde İHA kullanmışlardır. Optik ve akustik değerlendirmeler yerine renkli ve infrared kamera kullanarak hasarın yerini ve büyüklüğünü trafik akışını kesmeden hızlıca ve daha ucuza belirleyebilmişlerdir[27]. Mohamed Kerkecha ve arkadaşları İHA görüntülerinden üzüm yapraklarındaki hastalık belirtilerini görüntü renk bilgilerini kullanarak tespit etme çalışmalarında CNN kullanmış farklı renk uzaylarında ve bitki örtüsü indexlerinde yaptıkları uygulamalarda en iyi sonucun ExR, ExG, ExGR bitki örtüsü kombinasyonu ve 16x16 yama boyutuyla %95,80 başarımlı olduğunu gözlemlemişlerdir[28].

## 2. MATERYAL VE YÖNTEM

Bu bölümde İHA ile tespit edilen görüntülerin derin öğrenme yöntemleriyle analizinde kullanılan materyal ve yöntemler açıklanmıştır.

Tez çalışmasında cihaz olarak dijital kameraya sahip döner kanatlı bir İHA'dan elde edilmiş fotoğraflar ve derin öğrenme bilgisayarı kullanılmıştır.

Geliştirilmek istenen görsel değerlendirme uygulamasında çoklu sınıflandırma problemleri ele alınacaktır. Algoritma çıktısı verilen bir görselin içerisinde istenen nesnelerden bulunup bulunmadığı, bulunuyorsa konumunun neresi olduğunun gösterilmesi şeklindedir.

İstenilen nesnenin tespit edilebilmesi için ham veriye ihtiyaç vardır. Ham veri nesnenin aranacağı sayısal görüntü ve aranan nesnelerin farklı durumlarda çekilmiş sayısal görüntülerden oluşur. Ham veriler üzerinde yapılan etiketleme, dönüşüm ve eğitim gibi işlemler sonucunda üretilen dosyalar(xml, csv, proto, inference\_graph, label map gibi) derin öğrenme algoritmaları tarafından işlenmek üzere kullanılabilir.

Oluşturacak uygulama yazılımında Tensorflow kütüphanesi kullanılmıştır. Uygulamada daha önceden eğitilmiş modeller tekrar eğitilerek istenilen nesneler için bir sınıflandırıcı elde edilmiştir. Bu sayede modelin baştan oluşturulması için gerekli olan işlem gücü ve zaman sarfiyatı minimum seviyede olmuştur.

### 2.1. Kullanılan Veri Seti

Derin Öğrenme algoritmalarının başarı oranını etkileyen en önemli faktörlerden biri modelin eğitilmesinde kullanılan veri kümelerinin büyüklüğü, kalitesi ve zenginliğidir. Derin öğrenme veri kümesinin büyüklüğü ile doğru orantılı olarak başarılı sonuç verecektir. Kapsamlı bir veri kümesi oluşturmak zaman alan ve maliyetli bir iş olduğundan daha önceden üretilmiş veri kümeleri yapılan çalışmalarda sıklıkla kullanılır. Yapılan çalışmanın niteliğine uygun olarak veri kümesi seçilmelidir. Özel uygulamalarda özgün bir veri seti oluşturmak gerekebilir. Bu çalışmada sensefly.com

sitesinde yayınlanan veri setlerinden airport[29], Small Village, Industrial Estate isimli veri setlerinden modelin eğitilmesi için 203 adet görüntü, modelin test edilmesi için 37 adet görüntü toplam 240 görüntü seçilmiştir. Görüntüler üzerinde 3589 adet Araba, 2015 adet Ağaç, 1195 adet Ev, 196 Uçak, 130 adet İş makinesi, 380 adet Kamyon, 36 adet Helikopter etiketlenmiştir. Bu üç data setin yeryüzü çözünürlükleri ve uçuş yükseklikleri Tablo 2.1 de verilmiştir.

**Tablo 2.1.** Sensefly Data Setleri

Veri Seti	Görüntü Sayısı	Yeryüzü Çözünürlüğü	Uçuş Yüksekliği
Airport	557	3.14 cm (1.23 in)/px	120 Metre
Small Village	297	5 cm (1.96 in)/px	162 Metre
Industrial Estate	113	2.1 cm (0.82 in)/px	100 Metre



**Şekil 4.1** Sensefly Data Setinden Bir Örnek

Ayrıca bu çalışmada kullanılacak transfer öğrenme modellerinde coco veri kümesinde eğitilmiş derin öğrenme modellerinin ağırlıkları kullanılacaktır.



Eğitim ve test veri setlerindeki toplam 240 görüntü eğitim süresini azaltmak amacıyla yükseklik ve genişlikleri 4'te 1 e düşürüldükten sonra labelImg programı ile 7 farklı nesne sınıfını tanımak üzere etiketlenmiştir. Bu sınıflar ve etiket sayıları Tablo 2.2. de gösterilmiştir.

**Tablo 2.2.** Nesne Sınıfları-Etiket Sayıları

Nesne Sınıfı	Araba	Ağaç	Ev	Uçak	İş makinesi	Kamyon	Helikopter	Toplam
Eğitim	2972	1613	973	172	103	345	34	6212
Test	617	402	222	24	27	35	2	1329

### 2.3. Derin Öğrenme Mimarileri

Derin öğrenme uygulamalarının geliştirilmesinde farklı makine öğrenmesi ve yapay zeka mimarileri kullanılabilmektedir. Yapay Sinir ağları insan beyninin öğrenme fonksiyonunu gerçekleştirebilen yapay zeka algoritmalarıdır. Yapay sinir ağlarının eğitilmesi uygulama alanı ile ilgili örnekler yardımı ile gerçekleştirilir. Derin öğrenme için kullanılan bazı yapay sinir ağı çeşitleri şunlardır:

#### 2.3.1. Evrişimsel sinir ağları (Convolutional Neural Network, CNN)

Evrişimsel Sinir Ağları çok katmanlı algılayıcıların (MLP) bir türüdür. Evrişimsel Sinir Ağları görüntü işleme alanında kullanılmaktadır. Evrişimsel sinir ağları bir dizi evrişim katmanı ve bunlara yardımcı ek katmanlardan oluşmaktadır. Evrişim katmanı CNN'nin ana yapı taşıdır. Görüntünün özelliklerini algılamaktan sorumludur. Bu katman, görüntüdeki düşük ve yüksek seviyeli özellikleri çıkarmak için görüntüye bazı filtreler uygular. Filtre kenarları yada köşeleri algılayacak bir filtre olabilir. Bu filtreler genellikle çok boyutludur ve piksel değerleri içerirler.(5x5x3) 5 sayıları matrisin yükseklik ve genişliğini, 3 sayısı matrisin derinliğini temsil eder.



### 2.3.2. Tekrarlayan sinir ağıları (Recurrent Neural Network, RNN)

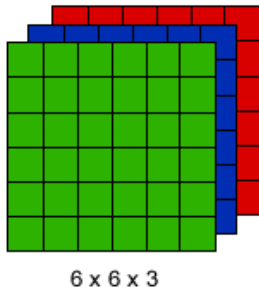
Tekrarlayan Sinir Ağı düğümler arasındaki bağlantıların yönlendirilmiş bir döngü oluşturduğu yapay sinir ağı sınıfıdır. Tekrarlayan sinir ağlarının asıl amacı ardışık bilgileri kullanmaktır. NLP’de kullanmak için uygundur. Bir cümlede bir sonraki kelimeyi tahmin etmek için o kelimenin öncesinde hangi sözcüklerin geldiğinin bilinmesi gerekir. RNN’nin farkı cümledeki her bir kelime için aynı görevi önceki çıktıları değerlendirerek tekrar yerine getirmesidir.

### 2.3.3. Uzun kısa vadeli hafıza ağıları (Long Short-Term Memory, LSTM)

RNN modellerinin geçmişten gelen bilgiyi kullanmakta zorlandığı durumlar için özel bir RNN türü olan Uzun Kısa Vadeli Bellek ağıları Hochreiter ve Schmidhuber tarafından tanıtılmıştır[32]. LSTM mimarileri konuşma ve metin işlemede oldukça iyi sonuçlar vermektedir.

## 2.4. Evrişimsel Sinir Ağlarının Yapısı

Evrişimsel sinir ağıları, yapay sinir ağlarında resim tanıma ve sınıflandırma kullanılan ana yöntemlerden biridir. CNN giriş olarak bir görüntü alır ve görüntüyü işleyerek sınıflandırır(Ör. Kedi, Köpek) . Bilgisayar giriş görüntüsünü görüntünün çözünürlüğüne bağlı bir pixel dizisi olarak algılar. Şekil 2.4. de  $h \times w \times d$  (  $h$  = Height,  $w$  = Width,  $d$  = Dimension )  $6 \times 6 \times 3$  RGB görüntü matrisi görünmektedir[33].

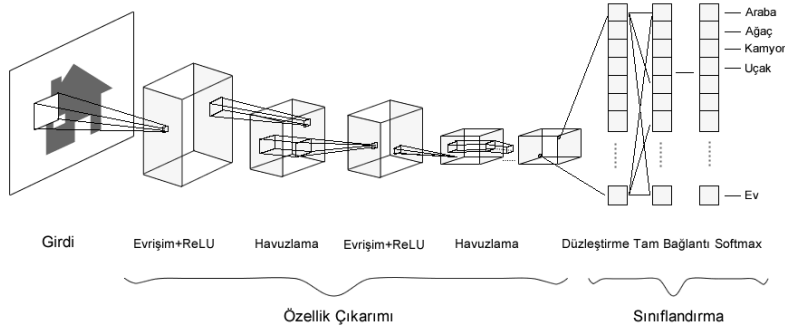


Şekil 2.4. RGB Görüntü Matrisi

Teknik olarak, derin öğrenme CNN modelleri eğitim ve test için, her bir giriş görüntüsünü filtreler(Kernel), Havuzlama(Pooling), tam bağlantılı katmanlardan (FC) oluşan bir dizi evrişim katmanından geçirerek ve Softmax işlevini uygulayarak 0 ile 1



arasındaki olasılıksal bir sınıflandırma değerleri elde eder. Şekil 2.5 de bir girdi görüntüsünü işleyen tam bir CNN akışı görülmektedir ve nesneleri olasılık değerlerine göre sınıflandırır.



Şekil 2.5. CNN İş Akışı

#### 2.4.1. Evrişim katmanı

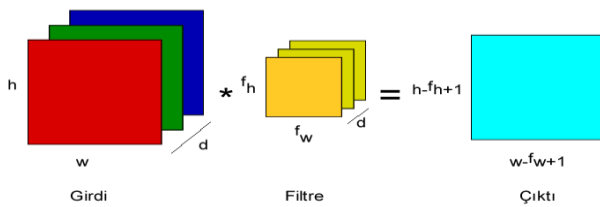
Evrişim katmanı görüntüden özellik çıkarılan ilk katmandır. Evrişim katmanı filtreler (kernel) aracılığıyla resimlerle özellikler arasındaki bağlantıyı kurmaya yarar.

Matematiksel olarak evrişim katmanı 2 giriş alır; görüntü matrisi ve filtre Şekil 2.6.'da gösterilmektedir.

Görüntü matrisi: yükseklik, genişlik, boyut ( $h \times w \times d$ )

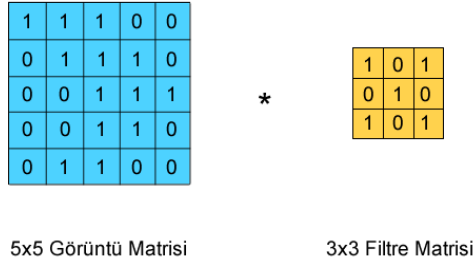
Filtre :yükseklik, genişlik, boyut( $f_h \times f_w \times d$ )

Çıktı boyutları:  $(h - f_h + 1) \times (w - f_w + 1) \times 1$



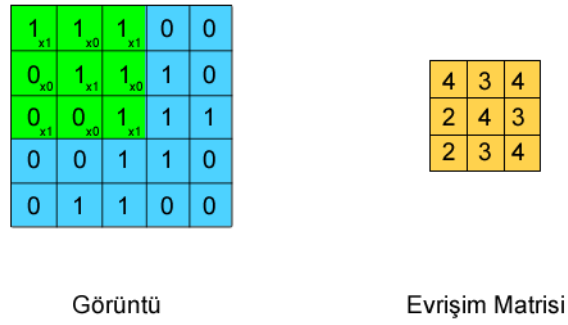
Şekil 2.6. Evrişim Katmanı

Şekil 2.7. de gördüğümüz gibi  $5 \times 5$  boyutlarında 0 ve 1'lerden oluşan bir giriş görüntüsü ve  $3 \times 3$  boyutlarında bir filtre olduğu düşünülürse.



**Şekil 2.7.** Evrişim ve Filtre

Giriş görüntü matrisi ile filtre matrisi çarpılarak özellik haritası (Feature Map) olarak adlandırılan ve Şekil 2.8 de gösterilen sonuç elde edilir.



**Şekil 2.8.** Özellik Haritasının Oluşumu

Görüntülere değişik filtreler uygulanarak köşe bulma, bulanıklaştırma, keskinleştirme gibi işlemler gerçekleştirilebilir. Şekil 2.9. da farklı filtreler uygulanarak elde edilen sonuçlar gösterilmiştir.

İşlem	Filtre	Görüntü									
Köşe Tespit	<table border="1"> <tr><td>1</td><td>0</td><td>-1</td></tr> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>-1</td><td>0</td><td>1</td></tr> </table>	1	0	-1	0	0	0	-1	0	1	
1	0	-1									
0	0	0									
-1	0	1									
Köşe Tespit	<table border="1"> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>-4</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> </table>	0	1	0	1	-4	1	0	1	0	
0	1	0									
1	-4	1									
0	1	0									
Köşe Tespit	<table border="1"> <tr><td>-1</td><td>-1</td><td>-1</td></tr> <tr><td>-1</td><td>8</td><td>-1</td></tr> <tr><td>-1</td><td>-1</td><td>-1</td></tr> </table>	-1	-1	-1	-1	8	-1	-1	-1	-1	
-1	-1	-1									
-1	8	-1									
-1	-1	-1									
Keskinleştirme	<table border="1"> <tr><td>0</td><td>-1</td><td>0</td></tr> <tr><td>-1</td><td>5</td><td>-1</td></tr> <tr><td>0</td><td>-1</td><td>0</td></tr> </table>	0	-1	0	-1	5	-1	0	-1	0	
0	-1	0									
-1	5	-1									
0	-1	0									
Gaussian Bulanıklaştırması	$\frac{1}{16} \star$ <table border="1"> <tr><td>1</td><td>2</td><td>1</td></tr> <tr><td>2</td><td>4</td><td>2</td></tr> <tr><td>1</td><td>2</td><td>1</td></tr> </table>	1	2	1	2	4	2	1	2	1	
1	2	1									
2	4	2									
1	2	1									

**Şekil 2.9.** Filtre Çeşitleri

#### 2.4.2. Kaydırma Adımı

Kaydırma adımı giriş görüntü matrisine uygulanan filtrenin kaç piksellik kaydırmalar yaparak görüntü üzerinde hareket edeceğini belirleyen bir parametredir. Şekil 2.10’da kaydırma adım değeri 2 olan evrişim katmanının çalışması gösterilmiştir.

1	2	3	4	5	6	7
11	12	13	14	15	16	17
21	5	10	2	1	5	3
10	12	20	15	13	12	18
15	13	1	33	1	14	1
1	1	14	1	1	3	21
1	6	7	1	1	9	1

\*

1	1	1
1	1	1
1	1	1

=

78	67	75
107	96	68
59	58	52

**Şekil 2.10.** Kaydırma Adımı

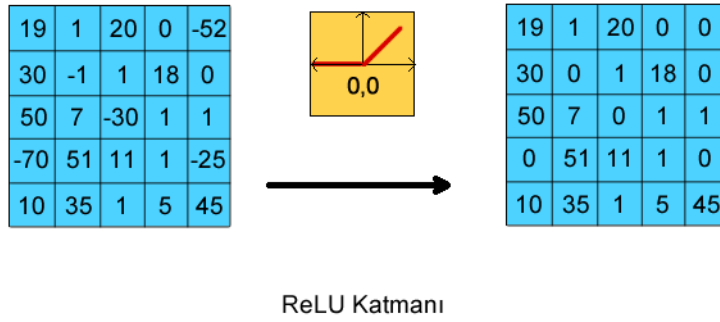
### 2.4.3. Dolgulama

Girdi görüntüleri ile filtreler arasında boyut uyumsuzlukları olduğu durumlarda kenar piksellerdeki görüntülerin yok sayılmaması için Sıfır-Dolgulama denilen işlem ile içeriği sıfır olan pikseller görüntüye eklenir. Diğer bir seçenek filtreye uymayan girdi görüntüsünün çıkarılmasıdır. Bu yöntem Geçerli-Dolgulama işlemi denilir.

### 2.4.4. Aktivasyon fonksiyonu

Aktivasyon fonksiyonları çok katmanlı yapay sinir ağlarında doğrusal olmayan dönüşüm işlemleri için kullanılmaktadır. Gizli katmanlarda geri türev alınabilmesi için gizli katmanların çıktısı aktivasyon fonksiyonları ile normalize edilmektedir. Yapay sinir ağlarında birçok aktivasyon fonksiyonu kullanılmakla birlikte evrişim katmanlarında Doğrultulmuş Doğrusal Birim (ReLU) fonksiyonu tercih edilir. ReLU, doğrusal olmayan işlemler için kullanılan bir fonksiyondur ve Rectified Linear Unit ‘in kısaltılmış halidir. Şekil 2.11.’de ReLU fonksiyonun çalışması görülmektedir. Fonksiyon negatif giriş alırsa, 0 değerini döndürür, ancak x pozitif bir değer ise x değerini geri döndürür.

Çıktısı  $f(x) = \max(0, x)$ . Formülü ile ifade edilir



**Şekil 2.11.** ReLU Aktivasyon Fonksiyonu

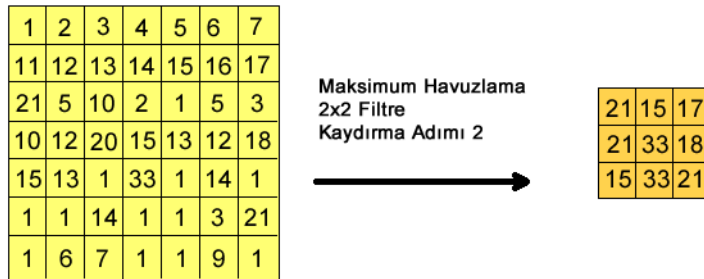
ReLU yerine tanh veya sigmoid aktivasyon fonksiyonları da kullanılabilir.

#### 2.4.5. Havuzlama katmanı(Pooling)

Havuzlama katmanı resim boyutu büyük olduğunda parametre sayısını düşürmek için kullanılır. Mekansal(Spatial) havuzlama ayrıca her haritanın boyutunu azaltan ancak önemli bilgileri koruyan alt örnekleme(subsample) olarak adlandırılır. Mekansal havuzlamanın çeşitleri şunlardır

- Max Pooling
- Average Pooling
- Sum Pooling

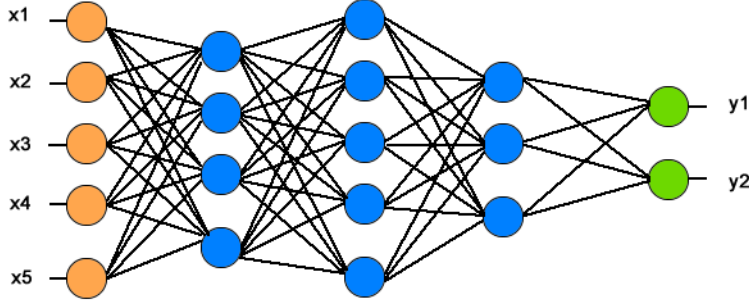
Maksimum havuzlama, özellik haritasındaki en büyük öğeyi alır. Öğelerin ortalamasını almak ortalama havuzlama, Özellik haritasındaki öğelerin toplamını almak toplam havuzlama adlandırılır. Şekil 2.12.'de Maksimum havuzlama işlemi gösterilmektedir.



**Şekil 2.12.** Maksimum Havuzlama

#### 2.4.6. Tamamen bağı katman(Fully Connected Layer)

FC katmanı olarak adlandırılan bu katmanlarla, Yapay sinir ağlarında, ortadaki katmanlar “gizli katman” olarak adlandırılırken, evrişimsel bağlamda “tamamen bağı katman” terimini kullanılır



Şekil 2.13. Tamamen Bağı Katman Yapısı

Şekil 2.13’de, özellik haritası matrisi vektör olarak dönüştürülecektir ( $x_1, x_2, x_3, \dots$ ). Tamamen birbirine bağı katmanlarla, bu özellikler bir model oluşturmak için bir araya getirilir. Son olarak, çıktıları kedi, köpek, araba, kamyon vb. olarak sınıflandırmak için softmax veya sigmoid gibi bir etkinleştirme fonksiyonu kullanılır

#### 2.5. Derin Öğrenme Kütüphaneleri

Tablo 2.3.’de görüldüğü gibi derin öğrenme için geliştirilmiş birbirinden farklı avantajları bulunan kütüphaneler mevcuttur[34]. Hız, çoklu GPU desteği, kolay kullanım, esneklik gibi faktörlere göre ihtiyaca uygun olan kütüphane seçilir. Kütüphane kullanımı çok katmanlı yapay sinir ağının oluşturulması, eğitilmesi ve test edilmesi sürecinde hesaplama işlemlerini kütüphane bünyesinde bulunan hazır fonksiyonlara gerçekleştirerek programcının iş yükünü azaltır. Kütüphane kullanılmaması durumun da tüm kodlama işlemlerinin sıfırdan oluşturulması gerekir.

**Tablo 2.3. Derin Öğrenme Kütüphaneleri**

Kütüphane	Yazıldığı dil	Geliştirici	Öne Çıkan Özellikleri
Theano	Python	MILA Lab	- Etkin dokümantasyon. – Keras gibi uygulama arayüzleri ile matematiksel hesap kolaylığı. - Grafik işlemci desteği.
Caffe	Python	Berkeley Vision and Learning Center (BVLC)	- Çeşitli modeller ön eğitimi yapılmış ağların olması. - Grafik işlemci desteği.
Torch	Lua	Ronan Collobert, Clement Farabet, ...	- Algoritma tasarımı konusunda esnek ve hızlı – Grafik işlemci desteği. (CUDA) – Kullanım kolaylığı
Digits	C++	NVIDIA	- Paralel GPU'lar üzerinde yapay sinir ağı tasarım ve eğitim özelliği – Güçlü görselleştirme yeteneği ile gerçek zamanlı etkileşim
TensorFlow	Python	Google	- Bir uygulama arayüzü ile desktop, server yada taşınabilir cihazlardaki birden fazla CPU ve GPU'ya görev dağıtma yeteneği.
DeepLearning	Java	Adam Gibson	- Java sanal makinesi tabanlı
KNET	Julia	Deniz Yuret	- Kolay kodlama, programlama için yüksek ifade gücü . – Grafik işlemci Desteği

## 2.6. Bölgesel Önerimli Evrişimsel Sinir Ağları (R-CNN)

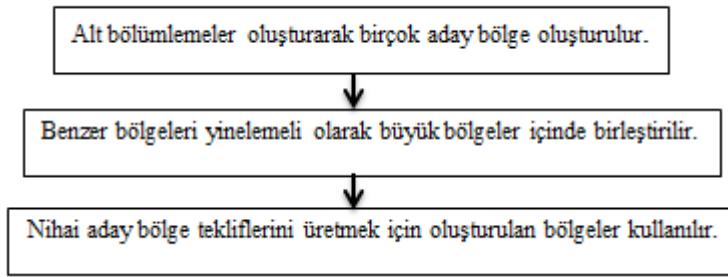
Bilgisayarlı görmenin önemli parçalarından biri nesne algılamadır. Nesne tespiti poz tahmini, araç tespiti, gözetim gibi konulara yardımcı olur. Nesne tespit algoritmaları ve sınıflandırma algoritmaları arasındaki fark, tespit algoritmalarında, görüntüde ilgilenilen nesnenin etrafına sınırlayıcı bir kutu çizmeye çalışılmasıdır. Hatta bu işlemin aynı görüntü için birden fazla defa yapılması gerekebilir.

Bu sorunu çözmek için kullanılacak yöntemlerden biri, görüntüden farklı ilgi alanlarını almak ve o bölgedeki nesnenin varlığını sınıflandırmak için bir CNN kullanmak olabilir. Bu yöntemle ilgili sorun, ilgilenilen nesnelerin görüntü içinde farklı uzamsal konumlara ve farklı en boy oranlarına sahip olabilmesidir. Bu nedenle, çok sayıda bölge seçmek gerekir ve bu işlem yoğun hesaplama gerektirir. Bu nedenle, R-CNN, YOLO gibi algoritmalar bu bölgeleri hızlı bir şekilde bulmak için geliştirilmiştir.[35]

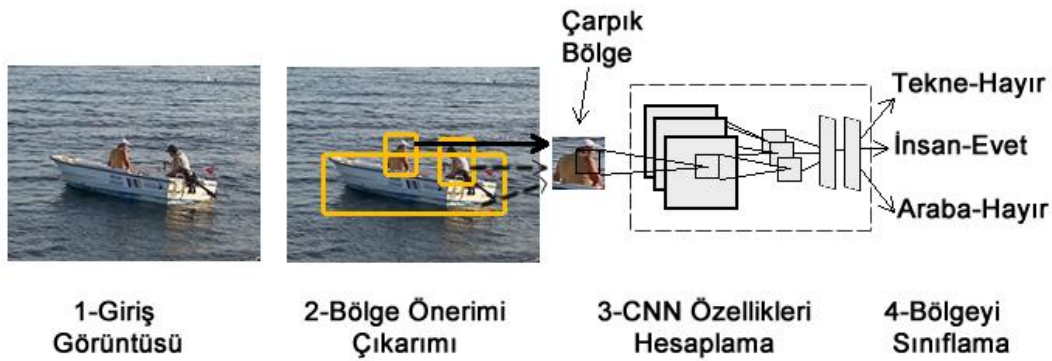
Çok sayıda bölge seçmenin oluşturduğu hesaplama zorluğunu azaltmak için, Ross Girshick ve arkadaşları[36] seçici arama (Selective Search)[37] algoritması kullanarak görüntü üzerinde nesnenin bulunma ihtimali olan 2000 bölgeyi bulan, bölge önerileri

(Region Proposals) olarak adlandırılan bir yöntem önermiştir. Şekil 2.15.'de bu yöntem kullanımı gösterilmiştir. Seçilmiş bölgeler yerine tüm görüntü üzerinde evrişim işlemini gerçekleştirmek işlem zamanını arttırarak özellikle gerçek zamanlı nesne tespiti işlemlerinin yapılması için yüksek işlem kapasitesi gerektirmektedir. Bu yöntemle çok sayıda bölgeyi sınıflandırmaya çalışmak yerine, yalnızca 2000 bölgeyle çalışabilir. Bu 2000 bölge önerisi, aşağıda yazılan seçici arama algoritması kullanılarak üretilmiştir.

Seçici arama algoritması işlem basamakları şekil 2.14.'de gösterilmiştir.



**Şekil 2.14.** Seçici Arama Algoritması Aşamaları

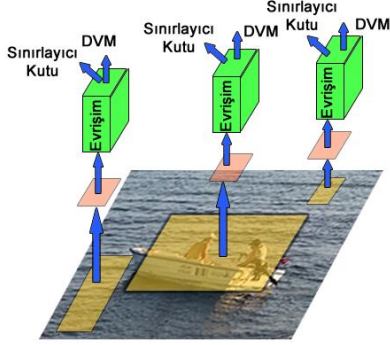


**Şekil 2.15.** Bölgesel Önerimli Evrişimsel Sinir Ağ İş Akışı

Bu 2000 aday bölge önerisi boyut olarak bir kareye dönüştürülür ve çıktı olarak 4096 boyutlu bir özellik vektörü üreten evrişimli bir sinir ağına girdi olarak verilir. Evrişimli sinir ağı bir özellik çıkarıcı olarak işlev görür ve çıktı yoğun katman, görüntüden çıkartılan özelliklerden oluşur ve elde edilen özellikler, o aday bölge teklifindeki nesnenin varlığını sınıflandırmak için bir Destek Vektör Makinesi sınıflandırıcıya girdi olarak verilir. Bölge önerileri dâhilinde bir nesnenin varlığını öngörmeye ek olarak, algoritma ayrıca sınırlayıcı kutunun hassasiyetini arttırmak için dengeleme değerleri



olan dört değeri de belirler. Örneğin, bir bölge önerisi verildiğinde, algoritma bir kişinin varlığını tahmin edebilir, ancak o bölge teklifindeki insan yüzünün yarısı teklif edilen bölgede olmayabilir. Bu nedenle, ofset değerleri, bölge teklifinin sınırlayıcı kutusunun ayarlanmasına yardımcı olur. Şekil 2.16'da Bölgesel önerimli evrişimsel sinir ağı şeması görülmektedir.

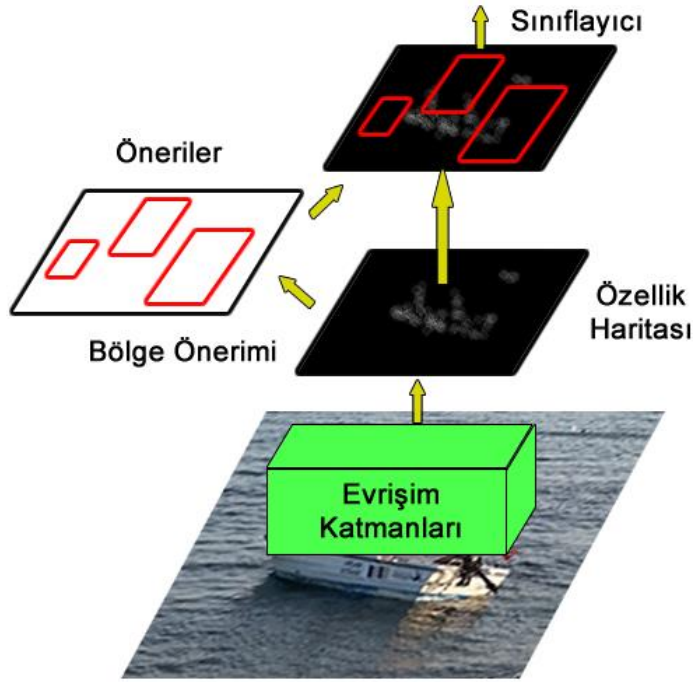


**Şekil 2.16. R-CNN**

Bölgesel önerimli evrişimsel sinir ağının zayıf olduğu alan görüntüyü başına 2000 bölge teklifi sınıflandırılmak zorunda kalınacağından ağı eğitmek hala çok zaman alır. Gerçek zamanlı kullanımlar için yavaştır. Seçici arama algoritması sabit bir algoritmadır, kötü aday bölgesi tekliflerinin oluşturulmasına yol açabilir.[38]

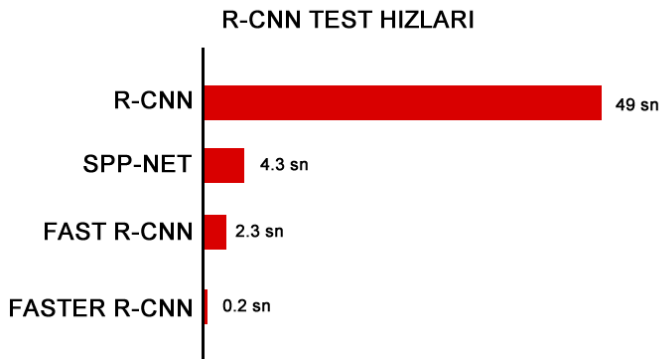
## **2.7. Faster R-CNN Sinir Ağları**

R-CNN ve Hızlı R-CNN bölge önerilerini bulmak için seçici bir arama kullanır. Seçici arama, ağın performansını etkileyen yavaş ve zaman alan bir işlemdir. Bu nedenle, Shaoqing Ren ve arkadaşları seçici arama algoritmasını ortadan kaldıran ve ağın bölge önerilerini öğrenmesini sağlayan bir nesne algılama algoritması geliştirmişlerdir.[39] Şekil 2.17'de Faster R-CNN algoritmasının yapısı gösterilmektedir.



**Şekil 2.17.** Faster R-CNN Yapısı

Fast R-CNN'ye benzer şekilde, görüntü bir evrişimsel özellik haritası sağlayan bir evrişimsel ağı girdi olarak sağlanır. Bölge tekliflerini belirlemek için özellik haritasında seçici arama algoritması kullanmak yerine, bölge tekliflerini tahmin etmek için ayrı bir ağı kullanılır. Tahmini bölge önerileri daha sonra önerilen bölge içindeki görüntüyü sınıflandırmak ve sınırlayıcı kutular için ofset değerlerini tahmin etmek için kullanılan bir ROI havuzlama katmanı kullanılarak yeniden şekillendirilir.



**Şekil 2.18.** Nesne algılama algoritmalarının karşılaştırılması (Saniye)

Şekil 2.18. de görüldüğü gibi Faster R-CNN en az test süresine ulaşarak gerçek zamanlı

uygulamalarda kullanılabileceğini göstermiştir. İnce ayar yapılmış Coco tespit modelinin PASCAL VOC 2012 test seti üzerinde görüntü başına yaklaşık 200 ms hızında çalıştığı görülmüştür [39].

## **2.8. Derin Öğrenme Modelleri**

Basit bir Evrişimli sinir ağı kurmanın yolu, birkaç evrişim katmanını arka arkaya koymak ve her birinden sonra ReLU katmanı eklemektir. Bundan sonra havuzlama katmanları ve düzleştirme katmanı eklenmelidir. Daha sonra ReLU katmanı kadar tamamen bağlı katman eklenir. Evrişimli sinir ağının son katmanı tamamen bağlı katman olmalıdır.[40]

### **2.8.1. Yaygın olarak kullanılan evrişimli sinir ağı tasarımları**

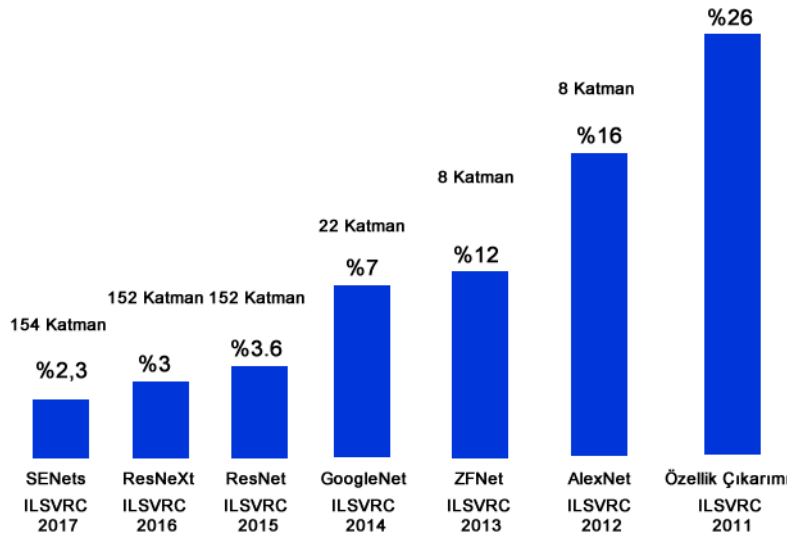
**LeNet**—Bu ağ, Evrişimsel sinir ağının ilk başarılı uygulaması kabul edilir. 1990'lı yıllarda Yann LeCun tarafından geliştirildi ve posta kodlarını, basit basamakları vb. okumak için kullanıldı.

**AlexNet**—Bu ağ, 2012'de ImageNet Large Scale Visual Recognition Challenge (ILSVRC) yarışmasında sunuldu. Diğer ağlardan oldukça başarılı bir performans göstermiştir.

**GoogLeNet**—ILSVRC 2014'ün kazananı bu ağ olmuştur. Ağdaki parametrelerin sayısını önemli ölçüde azaltmak için ortalama havuzlama katmanları kullanılmıştır.

**VGGNet**—Bu ağ, ağ derinliğinin Sinir ağları için ne kadar önemli olduğunu kanıtlamıştır. 16 tane Evrişim katmanı bulunur.

Şekil 2.19'de 2010 yılından beri düzenlenen ILSVRC yarışmasını kazanan Evrişimsel sinir ağı mimarileri ve katman sayıları gösterilmektedir



**Şekil 2.19.** ILSVRC Yarışması Yıllara Göre Birinciler

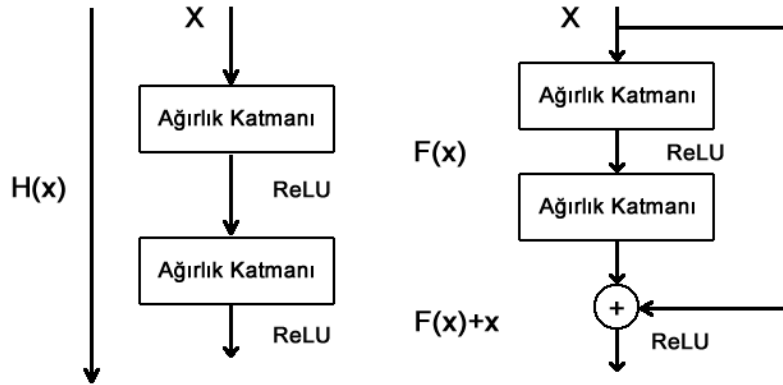
### 2.8.2. ResNet Mimarisi

Yapılan çalışmalarda ortak çıkarımlardan birisi : Evrişimli yapay sinir ağlarında ağ derinliği (katman sayısı) parametresinin başarımı önemli oranda etkilediğidir. Çok sayıda katman kullanmak teorik olarak CNN’lerde temsil kapasitesinin yükselterek başarılı ağ mimarileri oluşturulmasını sağlar. Ancak bu durum her ağ uygulamasında geçerli değildir. Bu durumda oluşacak iki sorun şunlardır.[41]

- Patlayan Degradeler ( Vanishing/Exploding Gradients) Sinir ağlarının eğitilmesi işleminde bazı nöronların etkisizleştiği görülür. Bu durum geri doğru yayılım (backpropagation) sırasında güçlü aktivasyonları olmayan nöronların değerlerinin uzun katman zinciri arasında azalarak eğitim sürecine katkı verememesi olarak ifade edilebilir.
- Optimizasyonun Zorlaşması : Evrişimli yapay sinir ağlarında parametre paylaşımı yöntemiyle parametre sayısı azaltılır. Ağın derinliği artarsa farklı yapılardan dolayı oluşan yeni parametrelerle karşılaşılacaktır. Yüksek parametre sayısı eğitim sürecini ve uygun değer çözümü bulunmasını zorlaştıracaktır.

Patlayan degradeler ve optimizasyonun zorlaşmasından dolayı oluşabilecek sorunları engelleyebilecek bir mimari olan ResNet’in en önemli katkılarından birisi Yapay sinir ağının katman sayısını arttırırken daha hızlı ve başarılı şekilde ağı eğitebilmemizi

sağlamasıdır.



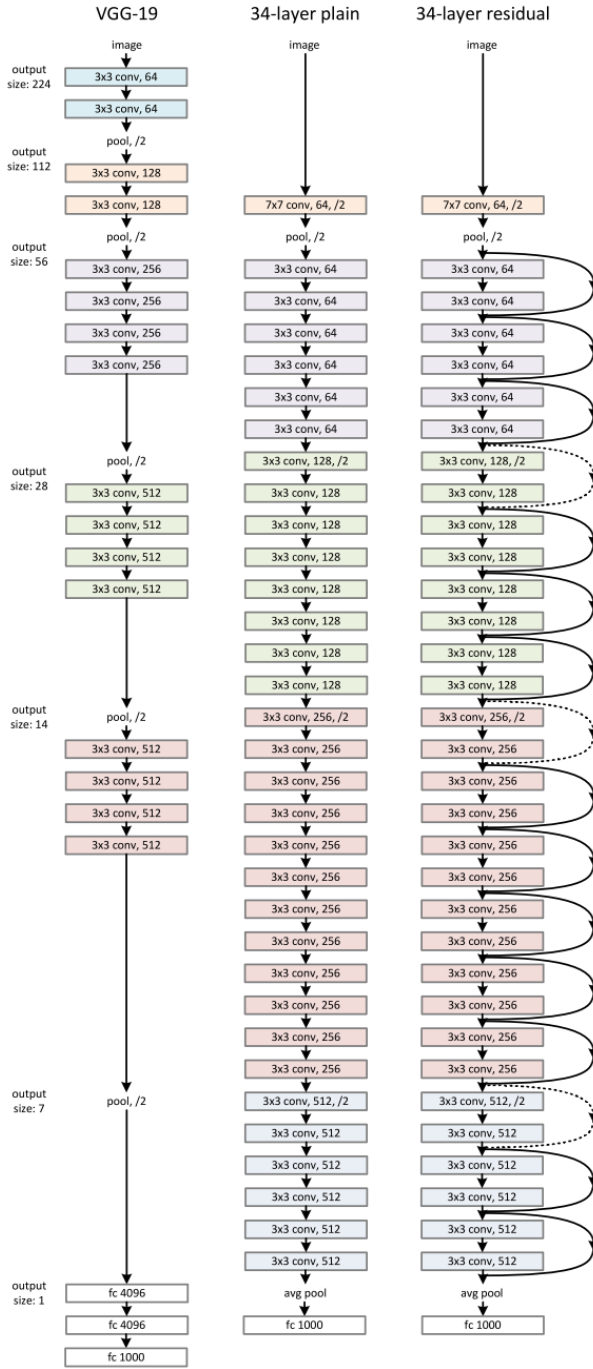
**Şekil 2.20.** CNN ile Residual bağlantılı CNN karşılaştırması

Normal şartlarda girişten çıkışa kadar olan kısmı doğrusal olmayan bir  $H(x)$  fonksiyonu ile haritalandırılabilir. ResNet mimarisinde bu yol,  $H(x)$  yerine  $F(x) := H(x) - x$  olarak tanımlanan başka bir doğrusal olmayan fonksiyon ile haritalandırılır. Buna ek olarak da girişten çıkışa bir kısa yol bağlantısı yapılarak,  $x$  (giriş) değeri  $F(x)$  fonksiyonuna aritmetik olarak eklenir. Daha sonra  $F(x) + x$  fonksiyonu birlikte ReLU'den geçirilir. Böylece ikinci katmanın çıkışına giriş verisi de eklenerek, geçmiş katman verilerinin ileri katmanlara etkin olarak iletilmesi hedeflenir. Yapılan deneyler bu iki yaklaşımın da aynı fonksiyona ulaşılmasını sağladığı ama ağı eğitme kolaylığının aynı olmadığı görülmüştür.

Şekil 2.20. 'da sağ taraftaki ağ yapısındaki giriş-çıkış arasındaki bağlantı “shortcut connections” olarak isimlendirilir. Aralarındaki fark şekil 2.20.'da gösterilen “shortcut” bağlantılar olan iki derin nöral ağın ImageNet ve CIFAR-10 datasetleri üzerindeki karşılaştırmaları, Kaiming He ve arkadaşlarının “Deep Residual Learning for Image Recognition” isimli çalışmasında bir şekilde belirtilmiştir[42].

2015 yılında yayınlanan bu çalışmada 152 katmana sahip olan ResNet'in o ana dek ImageNet üzerinde koşturulan en derin ağ özelliğini taşımakta olduğu belirtilir. ResNet'ten 8 kat daha az katmanlı bir yapı olan VGG Netten daha az parametreye sahiptir. Bu da yapay sinir ağının eğitim sürecini hızlandırır.

Şekil 2.21'de ResNet mimarisi gösterilmiştir. Tablo 2.4. de ResNet mimarisinin diğer türlerinin katman yapıları gösterilmiştir.



Şekil 2.21. ResNet Mimarisi[43]

**Tablo 2.4.** Farklı ResNet Katman Mimarileri

Katman Adı	Çıkış Büyükliği	18 Katman	34 Katman	50 Katman	101 Katman	152 Katman
Evrişim 1	112x112	7x7,64 , Kaydırma Adımı 2				
Evrişim 2	56x56	3x3 Maksimum Havuzlama , Kaydırma Adımı 2				
		3x3,64	3x3,64	1x1,64	1x1,64	1x1,64
		3x3,64 x2	3x3,64	3x3,64	3x3,64	3x3,64
				1x1,256 x 3	1x1,256 x 3	1x1,256 x 3
Evrişim 3	28x28	3x3,128	3x3,128	1x1,128	1x1,128	1x1,128
		3x3,128 x2	3x3,128	3x3,128	3x3,128	3x3,128
				1x1,512 x 4	1x1,512 x 4	1x1,512 x 8
Evrişim 4	14x14	3x3,256	3x3,256	1x1,256	1x1,256	1x1,256
		3x3,256 x2	3x3,256	3x3,256	3x3,256	3x3,256
				1x1,1024 x 6	1x1,1024 x 23	1x1,1024 x 36
Evrişim 5	7x7	3x3,512	3x3,512	1x1,512	1x1,512	1x1,512
		3x3,512 x2	3x3,512	3x3,512	3x3,512	3x3,512
				1x1,2028 x 3	1x1,2028 x 3	1x1,2028 x 3
	1x1	Ortalama Havuzlama 1000 Tamamen bağlı softmax				

### 2.8.3. GoogLeNet

Christian Szegedy ve arkadaşları[44] tarafından geliştirilen diğer adı Inception olan mimari kompleks bir yapıya sahiptir. Hız ve doğruluğu arttıracak yöntemlere sahiptir. Geliştirilerek farklı sürümler geliştirilmiştir.

Inception v1[44], Inception v2, Inception v3[45], Inception v4 and Inception-ResNet.[46]

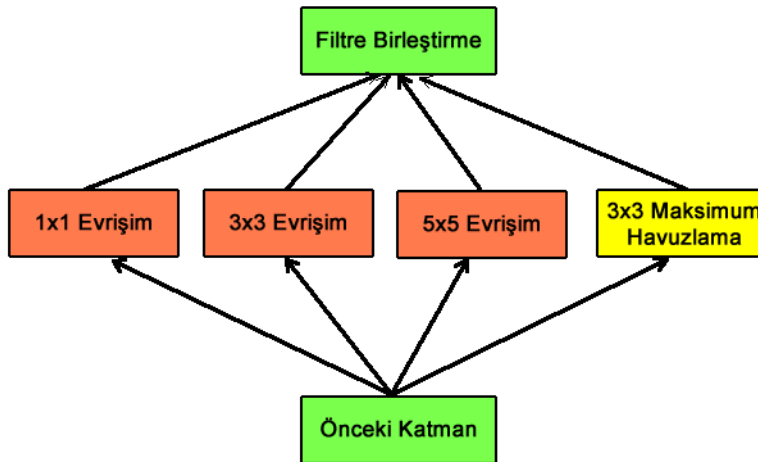
Her sürüm bir önceki sürümde değişiklikler yapılarak geliştirilmiştir. Sürüm yükseltmelerini anlamak, hem hız hem de hassasiyeti optimize edilmiş özel sınıflandırıcılar oluşturmada faydalı olabilir. Kullanılacak verilere bağlı olarak, daha düşük bir sürüm kullanmak daha iyi sonuçlar verebilir.



**Şekil 2.22.** Farklı Büyüklüklerde Nesneler

Şekil 2.22.de görülen üç görüntüde tespit edilmesi istenen köpeğin görüntü üzerinde kapladığı alan farklı büyüklüktedir. Nesnenin bu boyut farklılık nedeniyle, evrişim işlemi için doğru filtre boyutunu seçmek zorlaşır. Daha büyük olarak görülen bilgiler için daha büyük bir filtre, daha küçük görüntüler için daha küçük bir filtre tercih edilmelidir. Çok derin ağlar ezberlemeye(Overfitting) eğilimlidir. Ağırlık güncellemelerini tüm ağ üzerinden geçirmek de zordur ve evrişim işlemlerinin yığında tutulması hesaplama açısından kaynak gerektirir.

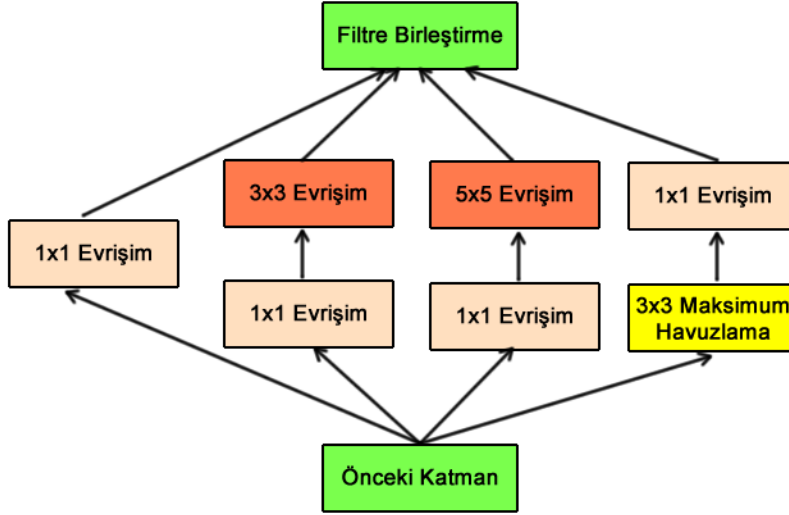
Inception mimarisi bunlara çözüm olarak birden fazla filtreyi girdi üzerinde kullanmayı ve çıktıyı bir sonraki inception modülüne göndermeyi önermektedir. Şekil 2.23’de 3 farklı boyutta filtre (1x1, 3x3, 5x5) ile giriş üzerinde evrişim gerçekleştirilmiş ve maksimum havuzlama yapılmıştır. Çıktılar birleştirilip ve bir sonraki Inception modülüne gönderilmiştir.



**Şekil 2.23.** Inception Modülü

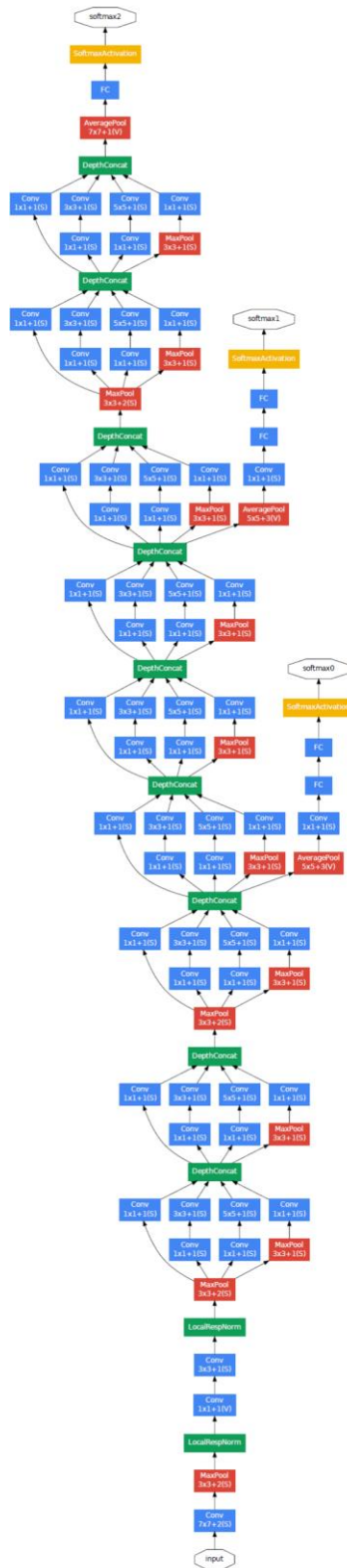


Şekil 2.24’de derin sinir ağındaki hesaplama işlevini azaltmak için 3x3 ve 5x5 evrişimlerinden önce ekstra 1x1 evrişim ekleyerek giriş kanallarının sayısı sınırlandırılmıştır. 1x1 evrişim maksimum havuzlama katmanından sonra eklenmiştir.



**Şekil 2.24.** Boyut Küçültülmüş Inception Modülü

Boyut küçültülmüş Inception modülü kullanarak inşa edilen sinir ağı mimarisi GoogLeNet (Inception v1) olarak bilinir. Mimari Şekil 2.25’de gösterilmiştir.

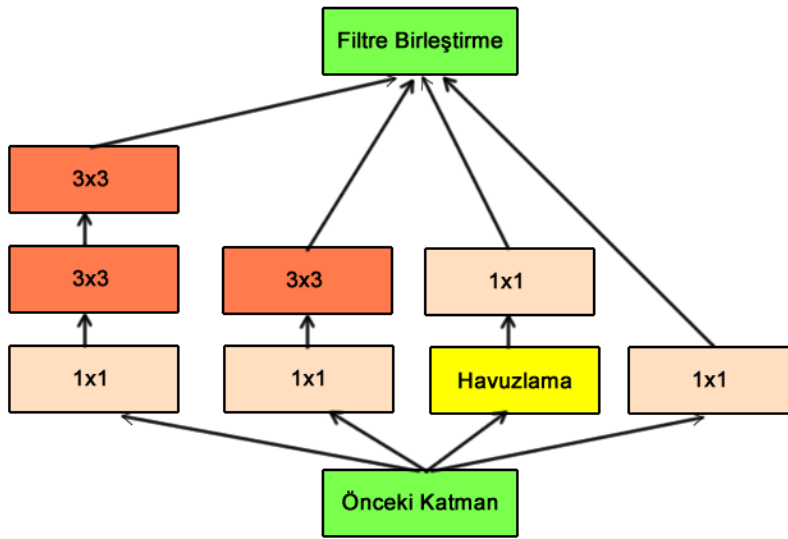


Şekil 2.25. GoogLeNet(Inception v1)[47]

GoogLeNet, doğrusal şekilde dizilmiş 9 adet inception modülüne sahiptir. 22 katman derinliğindedir. Havuz katmanları dahil edildiğinde katman sayısı 27 olur.

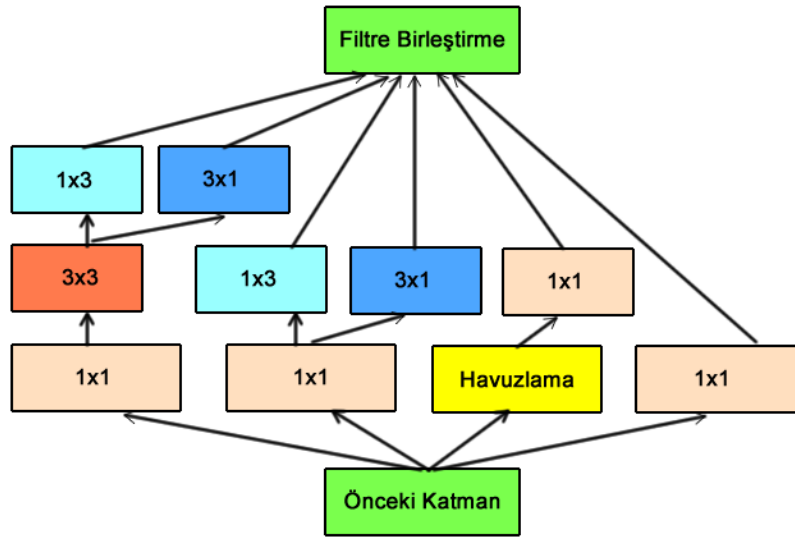
Inception v1(GoogLeNet)'nin ardından Christian Szegedy ve arkadaşları doğruluğu arttırmak ve hesaplama karmaşıklığını azaltmak için Inception v2 mimarisini tasarlamışlardır[45]. Tecrübeler evrişim katmanları girdilerin boyutunu büyük ölçüde değiştirmede sinir ağlarının daha başarılı olduğunu göstermektedir. Çünkü boyutları çok fazla azaltmak bilgi kaybına neden olur. Ayrıca akıllı çarpanlar yöntemlerini kullanarak, hesaplamalar karmaşıklık açısından daha verimli hale getirilebilir.

Inception v2 mimarisinde hesaplama hızını iyileştirmek için 5x5 evrişimi iki 3x3 evrişim işlemiyle değiştirilmiştir. 5x5 evrişim işlemi 3x3 evrişim işleminden 2,78 kat daha çok hesaplama gerektirir. Bu nedenle, iki 3x3 evrişim işlevinin arka arkaya kullanılması performansta artışa neden olur. Bu işlem Şekil 2.26'de gösterilmiştir.



**Şekil 2.26.** Inception v2 Modülü

Ayrıca 1 x n evrişimleri kullanarak ve ağın derinliği yerine genişliği artırılarak şekil 2.27'da görünen modül yapısı elde edilmiştir. Bu mimari ile hesaplama yükü azaltılıp aynı zamanda bilgi kaybı önlemiştir.



**Şekil 2.27.** Yatay Genişletilmiş Inception v2 Modülü

## 2.9. Geliştirilen Sistemin İşleyişi

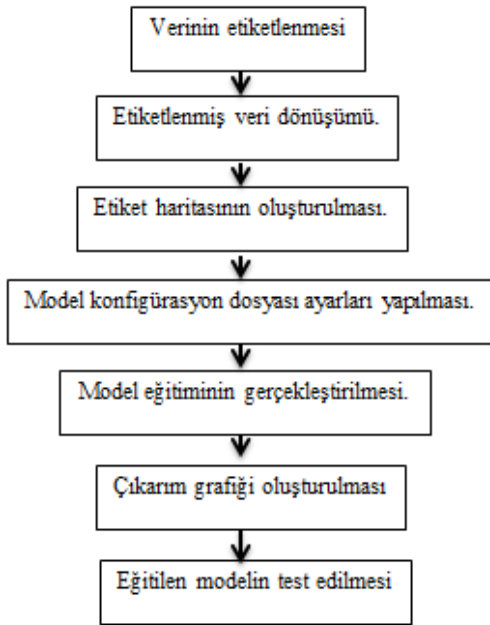
Geliştirilen sistemde insansız hava aracı üzerinde bulunan bir kameradan elde edilmiş ve etiketlenmiş görüntüler eğitim ve test kümelerine ayrılmış daha sonra Google Tensorflow kütüphanesini kullanan Nesne tespit uygulama ara yüzü kullanılarak eğitim işlemi gerçekleştirilmiştir. Eğitimde Google tarafından sunulan ön eğitimle ağırlık ve bias değerleri belirlenmiş nesne bulma modellerinin tamamen bağlı katmanlarının kendi veri setimizle yeniden eğitilmesi yöntemi kullanılmıştır. Bu sayede eğitim için gerekli adım sayısı ve eğitim süresi azaltılmıştır. Eğitim için Google model havuzunda bulunan farklı mimarilere sahip faster\_rcnn\_inception\_v2\_coco ve faster\_rcnn\_resnet101\_coco modelleri seçilmiştir. Bu modellerin seçilmesinin nedeni Tensorflow model havuzunda bulunan önceden eğitilmiş modeller içerisinde kısa çıkarım zamanına ve yüksek mAP değerine sahip modeller olmalarıdır. Kullanılan modellerin hızları ve mAP değerleri Tablo 2.5’de gösterilmiştir.

**Tablo 2.5.** Kullanılan Model Özellikleri

Model Adı	Hız(mili saniye)	mAP
faster_rcnn_inception_v2_coco	58	28
faster_rcnn_resnet101_coco	106	32

Google Nesne tespit uygulama ara yüzünü kullanabilmek için github sitesindeki tensorflow hesabının deposunun indirilmesi gereklidir[48]. Ayrıca gerekli diğer python kütüphaneleri de bilgisayarımıza Anaconda yada başka bir IDE yardımıyla kurulmalıdır[49]. Bu kütüphaneler şunlardır: Opencv-python, matplotlib

Ayrıca GPU üzerinde eğitim yapılacağı için cuda, cudann sürücüleri ve Visual Studio 2015 Community Edition Update 3 w. Windows Kit 10.0.10240.0 programının da sisteme kurulmalıdır. Şekil 2.28’de Nesne tanıma işlem sıralaması gösterilmiştir.



**Şekil 2.28.** Nesne Tanıma İşlem Sırası

### 2.9.1. Etiketlenmiş veri dönüşümü

labelImg programı etiketlenmiş her görüntü için bir xml uzantılı dosya oluşturur. Bu

xml dosyalarını nesne tespit uygulama ara yüzünün kullanabilmesi için Record dosyasına dönüştürülmeleri gerekir. Bunun için ilk olarak xml dosyalar csv dosya türüne `xml_to_csv.py` dosyası çalıştırılarak dönüştürülür. Daha sonra `generate_tfrecord.py` dosyası çalıştırılarak eğitim ve test için ayırdığımız resimlere ait record uzantılı dosyalar oluşturulur.

### 2.9.2. Etiket Haritasının Oluşturulması

Yapay sinir ağı modelimizin tamamen bağlantılı katmanları sınıflandırmak istediğimiz nesneler için yeniden eğitilirken etiket haritası(Label Map) denilen dosyaların içerisindeki sınıflandırma bilgilerine ihtiyaç duyar. Bu dosya içerisinde görüntüde tespit etmek istediğimiz her nesne sınıfının adı ve tanımlama numarası(Id) bulunur. Şekil 2.29'de etiket haritası dosya içeriği gösterilmiştir.

```
item {
  id: 1
  name: 'araba'
}

item {
  id: 2
  name: 'agac'
}

item {
  id: 3
  name: 'ev'
}

item {
  id: 4
  name: 'ismakinesi'
}
```

**Şekil 2.29.** Etiket haritası dosyası içeriği

### 2.9.3. Model konfigürasyon dosyası ayarları

Model konfigürasyon dosyası oluşturulacak yapay sinir ağı ile ilgili tüm yapısal bilgilerin tutulduğu dosyaya verilen isimdir. Google Tensorflow kütüphanesi tarafından kullanılabilen önceden eğitilmiş modellerin konfigürasyon dosyaları indirilmelidir. İndirilen bu konfigürasyon dosyası üzerinde kendi veri setimize ve sınıf sayımıza göre gerekli değişiklikler yapılmalıdır. Şekil 2.30'de örnek bir konfigürasyon dosyasının başlangıç satırları görülmektedir.

```

model {
  faster_rcnn {
    num_classes: 7
    image_resizer {
      keep_aspect_ratio_resizer {
        min_dimension: 600
        max_dimension: 1024
      }
    }
    feature_extractor {
      type: 'faster_rcnn_inception_v2'
      first_stage_features_stride: 16
    }
    first_stage_anchor_generator {
      grid_anchor_generator {
        scales: [0.25, 0.5, 1.0, 2.0]
        aspect_ratios: [0.5, 1.0, 2.0]
        height_stride: 16
        width_stride: 16
      }
    }
    first_stage_box_predictor_conv_hyperparams {
      op: CONV
    }
  }
}

```

**Şekil 2.30.** Konfigürasyon dosyası

Konfigürasyon dosyası içerisinde sınıf sayısı, kaydırma adım sayısı (Stride), filtre büyüklükleri, yığın büyüklüğü (Batch Size), öğrenme oranı (Learning Rate) gibi parametreler değiştirilebilir. Ön eğitilmiş model ağırlıklarının tutulduğu dosya yolu belirtilir.

### 1.1.1. Model eğitimi gerçekleştirme

Konfigürasyon dosyasında istenen değişiklikler yapıldıktan sonra kendi veri setimizle eğitime başlamak ve yeni ağırlık değerlerini hesaplamak için konfigürasyon dosyasının adı ve yolu belirtilerek train.py dosyası çalıştırılır.

```
python train.py --logtostderr --train_dir=training/ --pipeline_config_path=training/
faster_rcnn_inception_v2_coco.config
```

Eğitimi Tensorboard'da takip etmek için yeni bir komut penceresi açılıp

```
C:\tensorflow1\models\research\object_detection>tensorboard --logdir=training
```

Komutu çalıştırılır.

### 1.1.2. Çıkarım grafiği oluşturma

Eğitim sırasında kayıp (Loss) değerinin artık azalmadığı görüldüğünde eğitim sonlandırılır. Eğittikten sonra modeli kullanabilmek için dondurulmuş çıkarım grafiği (frozen inference graph) oluşturulması gerekir. Bunun için

```
python export_inference_graph.py --input_type image_tensor --pipeline_config_path
```

```
training/faster_rcnn_inception_v2_coco.config --trained_checkpoint_prefix
training/model.ckpt-XXXX --output_directory inference_graph
```

komutu çalıştırılır. Komutta XXXX yerine training klasöründe .ckpt dosyalarından sayısı en büyük olan dosyanın değeri yazılır. Training klasöründen en büyük sayı olarak "model.ckpt-190534" görülüyorsa XXXX yerine 190534 yazılmalıdır. Bu işlem sayesinde en son hesaplanan ağırlıklar kullanılarak çıkarım dosyası oluşturulacaktır.

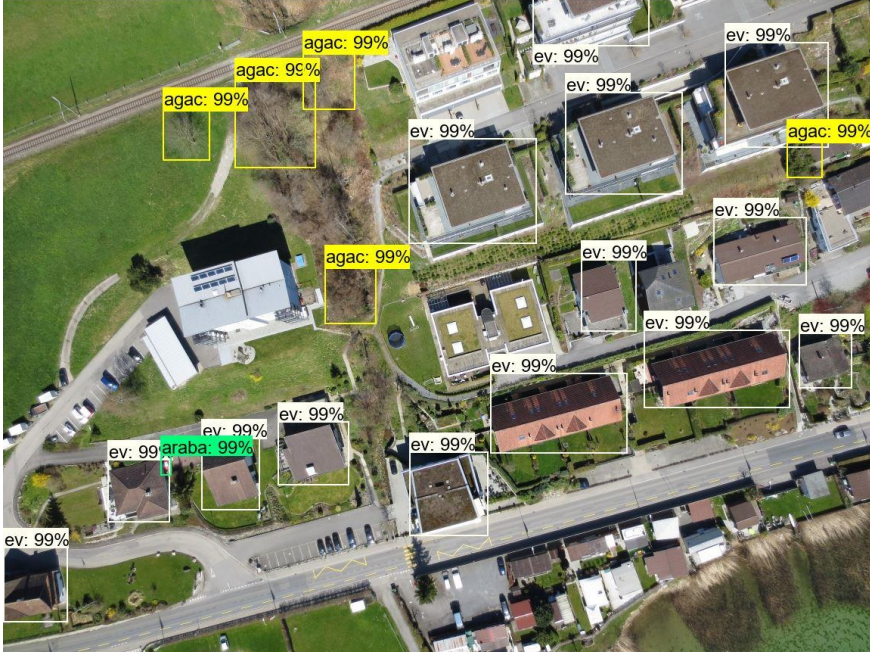
### 1.1.3. Eğitilen modelin test edilmesi

Eğitilen modelin havadan çekilmiş görüntüler üzerinde nasıl bir performans gösterdiğini görmek için Google Nesne tespit uygulama ara yüzünün bir parçası olan Object\_detection\_image.py dosyası kullanılır . Bu dosya üzerinde sınıflandırma yapılacak görüntü dosyasının adı ve uzantısı, çıkarım grafiğinin adı ve dosya yolu, etiket haritasının adı ve yolu gibi bilgiler bulunur. Örnek bir Object\_detection\_image.py dosyası içeriği Şekil 2.31.'da gösterilmektedir. Şekil 2.32'da sınıflandırma işlemi yapılmış bir görüntü dosyası üzerinde tespit edilmiş nesneler kutucuklar ile işaretlenmiş olarak gösterilmiştir.

```
import os
import cv2
import numpy as np
import tensorflow as tf
import sys
sys.path.append(".")
from utils import label_map_util
from utils import visualization_utils as vis_util
MODEL_NAME = 'inference_graph'
IMAGE_NAME = '3.jpg'
CWD_PATH = os.getcwd()
PATH_TO_CKPT = os.path.join(CWD_PATH,MODEL_NAME,'frozen_inference_graph.pb')
PATH_TO_LABELS = os.path.join(CWD_PATH,'training','labelmap.pbtxt')
PATH_TO_IMAGE = os.path.join(CWD_PATH,IMAGE_NAME)
NUM_CLASSES = 7
label_map = label_map_util.load_labelmap(PATH_TO_LABELS)
categories = label_map_util.convert_label_map_to_categories(label_map, max_num_classes=NUM_CLASSES, use_display_name=True)
category_index = label_map_util.create_category_index(categories)
detection_graph = tf.Graph()
with detection_graph.as_default():
    od_graph_def = tf.GraphDef()
    with tf.gfile.GFile(PATH_TO_CKPT, 'rb') as fid:
        serialized_graph = fid.read()
        od_graph_def.ParseFromString(serialized_graph)
        tf.import_graph_def(od_graph_def, name='')
    sess = tf.Session(graph=detection_graph)
```

Şekil 2.31. Object\_Detection\_Image.py Dosyası





Şekil 2.32. Örnek Çıkarım Görüntüsü

### 3. DENEYSEL ÇALIŞMALAR

Bu çalışmada, İnsansız Hava Aracından elde edilmiş görüntü üzerinde 7 adet sınıfa ait nesneleri bulan ve kutu içerisine alarak işaretleyen bir sistem geliştirilmiştir. Geliştirilen sistemde sınıflandırma yöntemleri denenmiştir. Sınıflandırma problemleri, eğitilmiş makine öğrenmesi problemlerinden birisidir. Sınıflandırma problemlerinde sistem, eğitim aşamasından geçirilir ve ardından başarı değerlendirmesi için Test aşamasına sokulur.

Bölüm 2.9’da anlatılan işlem basamakları gerçekleştirilerek eğitilen ve test için hazır duruma getirilen yapay sinir ağı modellerinin başarılarının test edilebilmesi için eval\_image.py isimli dosya Nesne tespit uygulama ara yüzü ile gelen Object\_Detection\_Image.py dosyasına eklemeler yapılarak geliştirilmiştir. Bu dosya eval klasörü içerisinde bulunan test için ayrılmış 37 adet görüntü dosyasını sınıflandırma işlemine tabi tutmaktadır. Her görüntü dosyası için ayrı ayrı olmak üzere görüntüde bulunduğu her tanımlı nesne türü için; nesne türü, güven skoru ve nesneyi kapsayan kutu koordinatlarını içeren txt uzantılı dosya oluşturur. Şekil 3.1’de bir görüntü dosyasında tespit edilen her nesne için sırasıyla nesne sınıfı, güven skoru, sol üst köşenin x ve y koordinatları, sağ alt köşenin x ve y koordinatları görülmektedir.

```
agac 0.9928625 724 273 814 344  
agac 0.96881795 36 228 151 316  
araba 0.90062255 575 794 593 838  
agac 0.7118741 93 491 235 658
```

**Şekil 3.1.** Eval\_image Dosya Çıktısı

Daha sonra bu değerlendirme dosyaları uzman kişiler tarafından etiketlenmiş dosyalardan üretilen txt uzantılı dosyalarla karşılaştırılarak ortalama kesinlik (Average Precision) denilen değer bulunur. Veri setinde tespit edilmesi gereken sınıf sayısı birden fazla ise tüm nesne sınıflarına ait ortalama kesinlik değerlerinin ortalaması alınır. Bulunan bu değere mAP (Mean Average Precision) denir. Bu değer sistemimizin eğitim işleminde ne kadar başarılı olduğunu gösteren bir ölçüttür.

Ortalama Kesinlik Faster R-CNN, SSD gibi nesne tespit algoritmalarının doğruluğunu ölçmede yaygın olarak kullanılan bir ölçüm yöntemidir. Geliştirilen sistemde başarıyı

ölçmek için her nesne sınıfına ait ortalama kesinlik değeri hesaplanacak ve bunların ortalaması alınarak mAP değerinin hesaplanmasında kullanılacaktır.

### 3.1. Hata Matrisi

Kullanılan sınıflandırma modellerinin performansını değerlendirmek için tahminlerin ve gerçek değerlerin karşılaştırıldığı hata matrisi sıklıkla kullanılmaktadır. Medikal karar destek sistemlerinde sıklıkla kullanılan sınıflandırma tahminleri şu dört değerlendirmeden birine sahip olacaktır.

- Doğru Pozitif (DP)
- Yanlış Pozitif (YP)
- Doğru Negatif (DN)
- Yanlış Negatif (YN)

- Doğru Pozitif (DP): Hasta olan kişinin hasta olarak sınıflandırılması durumudur.
- Yanlış Pozitif (YP): Hasta olmayan kişiyi hasta olarak sınıflandırılması durumudur.
- Doğru Negatif (DN): Hasta olmayan kişinin hasta değil olarak sınıflandırılması durumudur.
- Yanlış Negatif (YN): Hasta olan kişinin hasta değil olarak sınıflandırılması durumudur.

Şekil 3.2’ de örnek bir hata matrisi görülmektedir. Hasta ve hasta olmayan kişilere ait bir veri kümesinde tüm veriler hasta ya da hasta değil olarak sınıflandırılmış ve Hata matrisinde gösterilmiştir. Veri kümesinde toplam 100 kişinin bilgisi vardır.

**Tablo 3.1.** Hata Matrisi

HATA MATRİSİ		TAHMİN	
		HASTA DEĞİL	HASTA
GERÇEK	HASTA DEĞİL	45(DN)	6(YP)
	HASTA	3(YN)	46(DP)

Sınıflandırıcı model bu hastalardan 91 tanesin hastalık durumunu doğru olarak bilmıştır. Üç kişiye hasta olduğu halde hasta değil olarak sınıflamış, 6 kişiyi hasta olmadığı halde hasta olarak sınıflandırmıştır.

### 3.2. Kesinlik

Kesinlik, bir modelin yalnızca ilgili nesneleri tanımlama yeteneğidir. Doğru pozitif tahminlerin yüzdesidir. Denklem III.1’de formülü verilmiştir.

$$\text{Kesinlik} = \text{DP} / (\text{DP} + \text{YP}) \quad (\text{III.1})$$

### 3.3. Duyarlılık

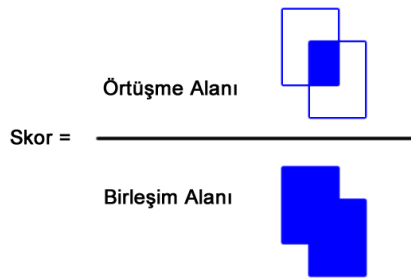
Duyarlılık, bir modelin tüm ilgili vakaları bulma yeteneğidir. Gerçek referans değerler arasında tespit edilen gerçek pozitiflerin yüzdesidir. Denklem III.2’de formülü verilmiştir.

$$\text{Duyarlılık} = \text{DP} / (\text{DP} + \text{YN}) \quad (\text{III.2})$$

### 3.4. Birleşmede Üzerinde Kesişim(IoU)

Gerçek referans değerler ile model tahmini arasındaki benzerlik Jaccard indeksi ile ölçülür. Jaccard indeksi formülü Denklem III.3’de verilmiştir. IoU hesabı, iki dikdörtgenin kesiştikleri alanın (intersection) bu iki dikdörtgenin bileşiminin (union) alanına bölümü olarak hesaplanır. Şekil 3.2.’de bu hesaplama işlemi grafiksel olarak gösterilmiştir.

$$J(X,Y) = |X \cap Y| / |X \cup Y| \quad (\text{III.3})$$



**Şekil 3.2.** Birleşmede Üzerinde Kesişim(IoU)

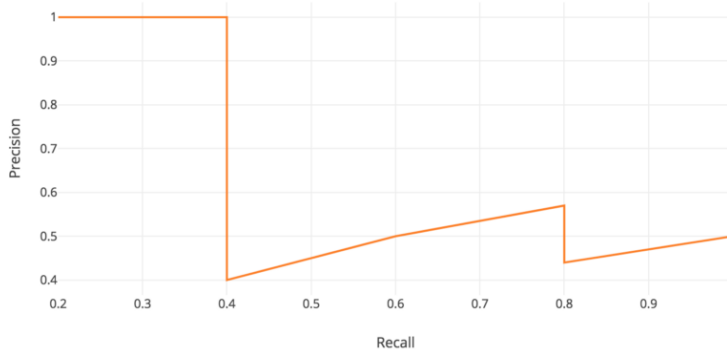
Bu değeri, öngörülen sınırlarımızın Gerçek referans değer ile ne kadar örtüştüğünü ölçmek için kullanılır. Geliştirdiğimiz sistemde, tahminin gerçek pozitif mi yoksa yanlış pozitif

mi olduğunu belirlemede IoU eşik değeri kullanılır. Örneğin bu değer 0.5 olabilir.

### 3.5. Kesinlik-Duyarlılık Eğrisi

Her nesne sınıfı için çizilen Kesinlik-Duyarlılık eğrisi nesne sınıflayıcıların performanslarını ölçmede kullanılır. Nesne sınıfları için çizilen eğride duyarlılık artarken kesinlik değeri ne kadar yüksekse sınıflayıcı o kadar iyi kabul edilir. İyi bir nesne algılayıcısını tanımlamanın başka bir yolu, yalnızca ilgili nesneleri tanımlayabilen bir algılayıcı olması ve tüm gerçek referans değerlerini bulmasıdır. Ne kadar az yanlış pozitif tahmin yaparsa kesinlik daha fazla artacaktır. Ayrıca 0 yanlış negatif duyarlılığı arttıracaktır.

Kötü bir nesne algılayıcısı tüm gerçek referans nesnelerini yakalamak için algılanan nesnelerin sayısını arttırması gerekir. Bu durum yanlış pozitif değerlerini arttırırken düşük kesinlik ancak yüksek duyarlılık değerlerine sebep olacaktır. Kesinlik-Duyarlılık eğrisine örnek şekil 3.3.'de gösterilmiştir.



Şekil 3.3. Kesinlik-Duyarlılık Eğrisi

### 3.6. Ortalama Hassasiyet

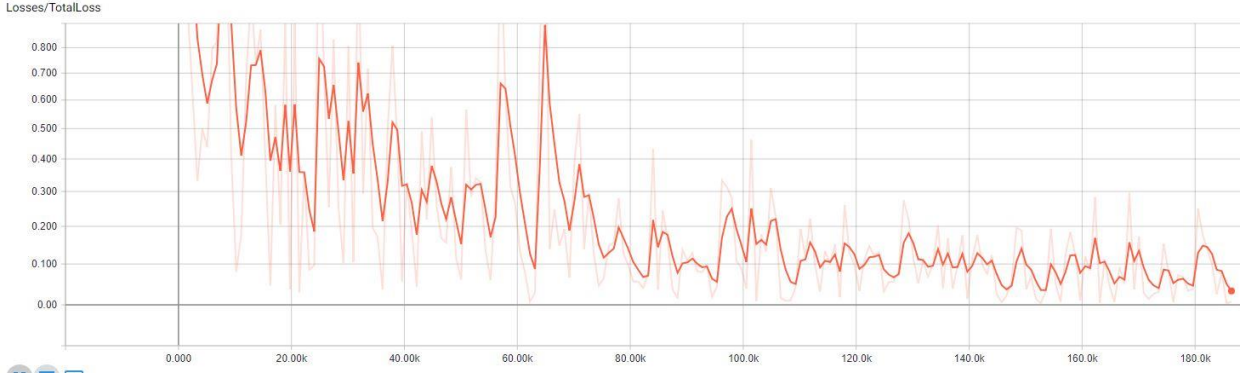
Nesne sınıflayıcıları birbirleri ile kıyaslayabilmek için Kesinlik-Duyarlılık eğrileri kullanılabilir fakat grafikleri yorumlamak çoğu zaman zordur. Bunun yerine çizilen eğrinin altında kalan alanın hesaplanması ile elde edilen sayısal değer karşılaştırma için kullanılır. Hesaplama kullanılan denklem III.4'de gösterilmiştir.

$$\sum_{r=0}^1 (r_{n+1} - r_n) \rho_{interp}(r_{n+1})$$

$$\rho_{interp}(r_{n+1}) = \max_{\tilde{r}: \tilde{r} \geq r_{n+1}} \rho(\tilde{r}) \quad (\text{III.4})$$

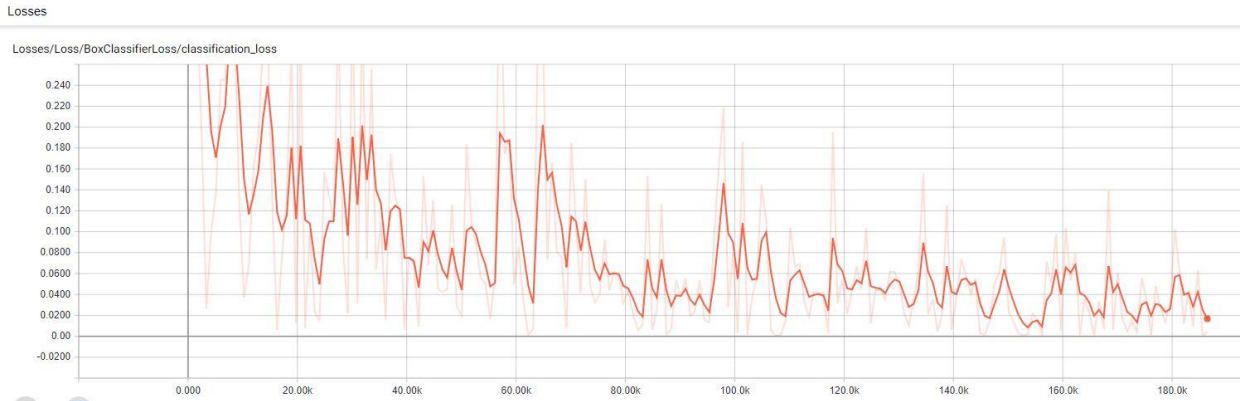
### 3.7. Deney Sonuçları

Model eğitimi ve testleri coco veri seti üzerinde ön eğitim uygulanmış faster\_rcnn\_resnet101\_coco ve Faster\_rcnn\_inception\_v2\_coco isimlerinde 2 farklı modelle gerçekleştirilmiştir. Faster\_rcnn\_inception\_v2\_coco modeli ile gerçekleştirilen eğitime ilişkin toplam kayıp grafiği Şekil 3.4’de verilmiştir. Eğitim 190000 adım sürdürülmüş total kayıp değeri 0.05 olduğu bir noktada sonlandırılmıştır. Grafikler Tensorboard kullanılarak elde edilmiştir.



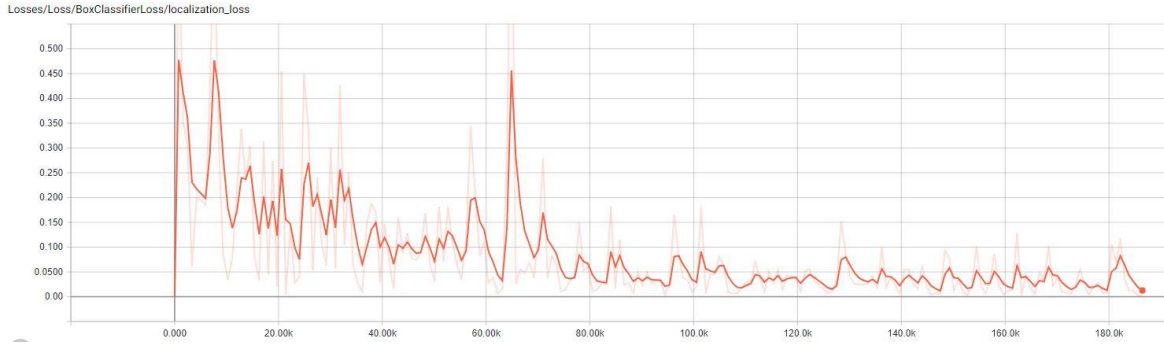
Şekil 3.4. Faster\_rcnn\_inception\_v2\_coco Toplam Kayıp

Şekil 3.5’de modelin eğitim sırasındaki sınıflandırma kaybını gösteren grafik görülmektedir. Sınıflandırma kaybının eğitim durdurulduğunda 0.02 olduğu görülmüştür.



Şekil 3.5. Faster\_rcnn\_inception\_v2\_coco Sınıflandırma Kaybı

Şekil 3.6’da Nesne konumlandırma kaybının değişimini gösteren grafik görülmektedir. Konumlandırma kaybının eğitim durdurulduğunda 0.01 olduğu görülmüştür.



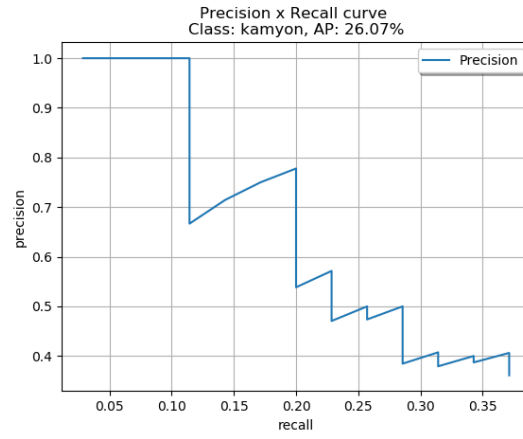
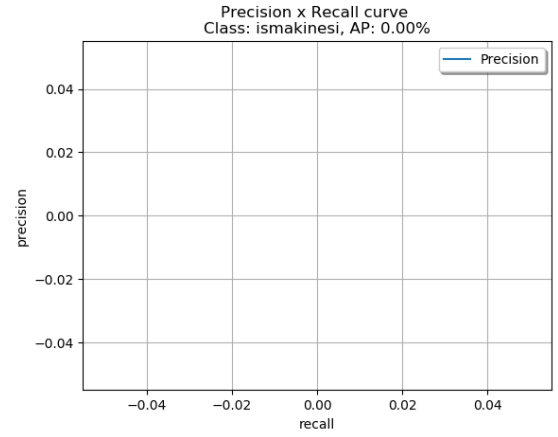
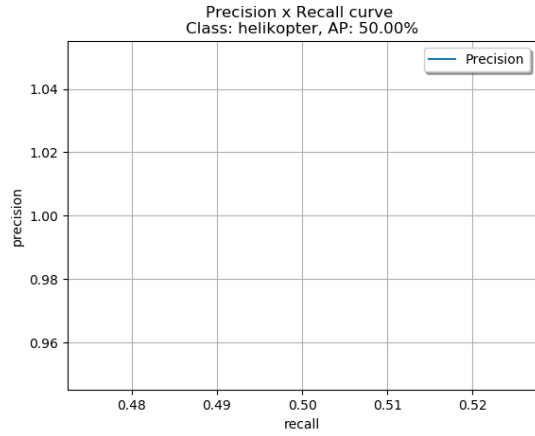
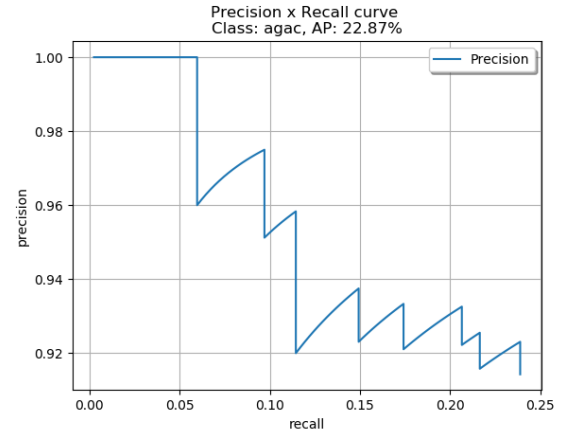
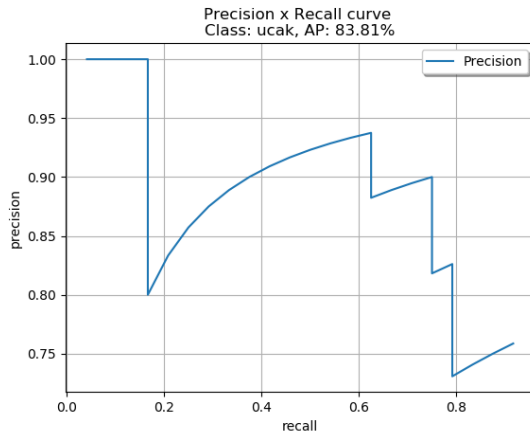
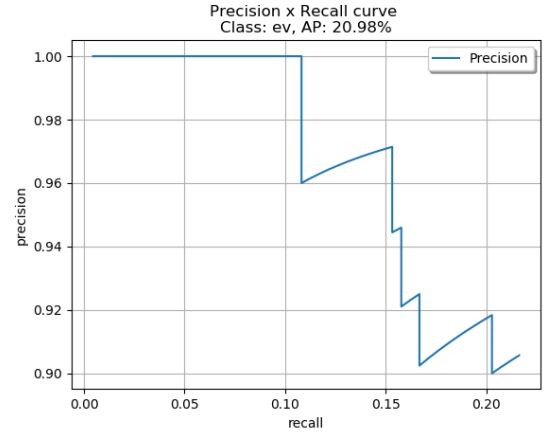
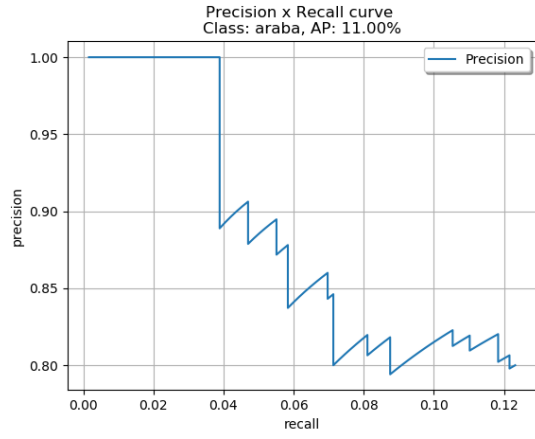
**Şekil 3.6.** Faster\_rcnn\_inception\_v2\_coco Konumlandırma Kaybı

Eğitim sonucu elde edilen model dosyası kullanılarak yapılan testlerde modelin başarısını ölçmek için Birleşmede Üzerinde Kesişim (IoU) değerleri 0.5 den başlayarak 0.05 artışlarla 0.95’e kadar giden 10 farklı eşik değeri için ortalama kesinlik değerleri bulunarak ortalamaları alınmıştır. Faster\_rcnn\_inception\_v2\_coco modeli için Birleşmede Üzerinde Kesişim değerlerine bağlı olarak bulunan ortalama kesinlik değerleri Tablo 3.2’de gösterilmiştir.

**Tablo 3.2.** Faster\_rcnn\_inception\_v2\_coco Ortalama Kesinlik Değerleri

IoU	0.5	0.55	0.60	0.65	0.70	0.75	0.80	0.85	0.90	0.95	Ortalama
mAP	30.68	30.68	30.68	30.40	30.35	28.95	27.8	24.54	13.76	6.10	25.39

Faster\_rcnn\_inception\_v2\_coco modelinin test sırasında 0.5 IoU değeri için elde edilmiş Ortalama Keskinlik değerlerinin hesaplanmasında kullanılan Kesinlik-Duyarlılık eğrilerini gösteren grafikler Şekil 3.7’de gösterilmiştir. Grafikler incelendiğinde en yüksek değer %83.81 ile uçak sınıfına ait olduğu görülmektedir. En düşük değerin ise %0 ile iş makinesi sınıfına ait olduğu gözlemlenmiştir. Uçak sınıfının diğer sınıflardaki nesnelere benzemeyen kanat ve burun yapısı gibi belirleyici farklarının olması başarının yüksek olmasına sebep olmuştur. İş makinesi sınıfının başarımının düşük olmasının sebebi bu sınıftaki nesnelerin özelliklerinin araba ve kamyon sınıflardaki nesnelere benzerliği ve etiketlenmiş verideki nesne sayısının sadece 103 olmasıdır.



Şekil 3.7. Faster\_rcnn\_inception\_v2\_coco modelinin Kesinlik-Duyarlık Eğrileri

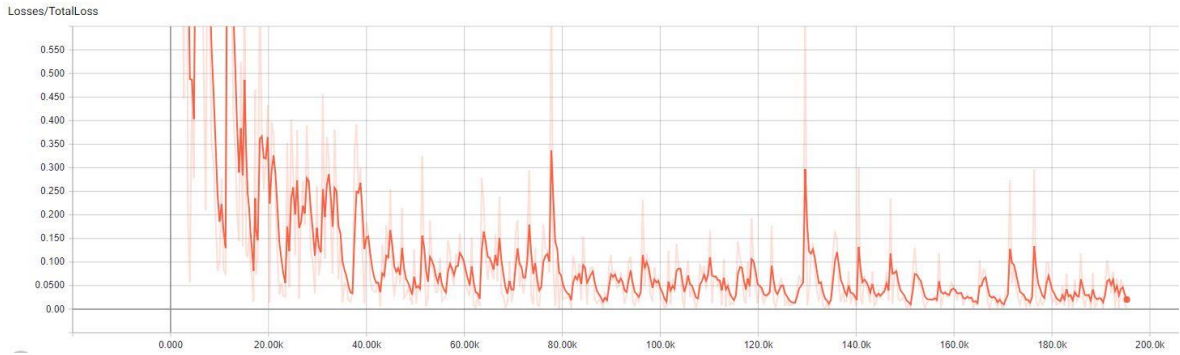


Şekil 3.8’de Faster\_rcnn\_inception\_v2\_coco modelinin test sırasında 0.5 IoU değeri için her sınıfta elde ettiği başarı değerleri gösterilmiştir.

```
C:\ozgur\Object-Detection-Metrics>python pascalvoc.py -t 0.50
AP: 22.87% (agac)
AP: 11.00% (araba)
AP: 20.98% (ev)
AP: 50.00% (helikopter)
AP: 0.00% (ismakinesi)
AP: 26.07% (kamyon)
AP: 83.81% (ucak)
mAP: 30.68%
```

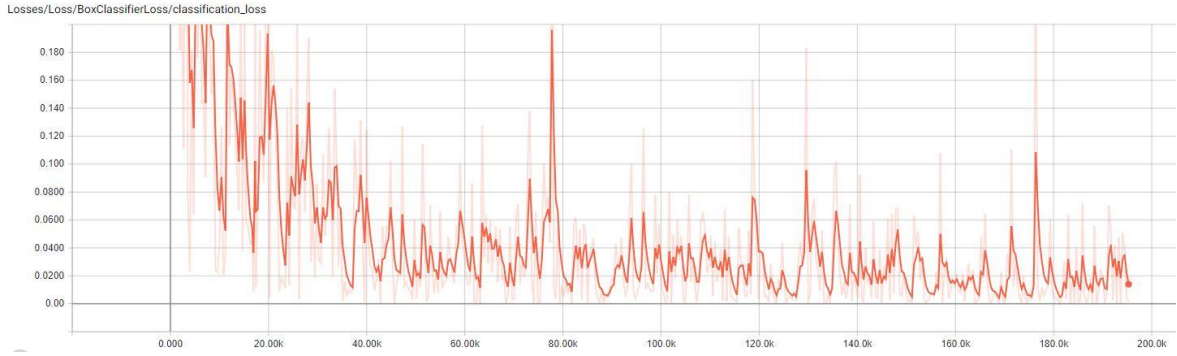
**Şekil 3.8.** Faster\_rcnn\_inception\_v2\_coco modeli Ortalama Kesinlik Değerleri

faster\_rcnn\_resnet101\_coco modeli ile gerçekleştirilen eğitime ilişkin toplam kayıp grafiği Şekil 3.9’de verilmiştir. Eğitim 196000 adım sürdürülmüş total kayıp değeri 0.025 olduğu bir noktada sonlandırılmıştır.



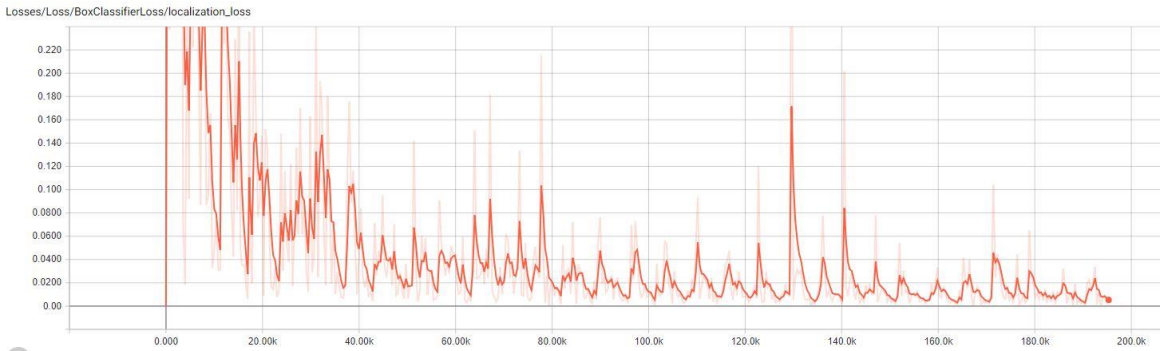
**Şekil 3.9.** Faster\_rcnn\_resnet101\_coco Toplam Kayıp

Şekil 3.10’de modelin eğitim sırasındaki sınıflandırma kaybını gösteren grafik görülmektedir. Sınıflandırma kaybının eğitim durdurulduğunda 0.015 olduğu görülmüştür.



**Şekil 3.10.** Faster\_rcnn\_resnet101\_coco Sınıflandırma Kaybı

Şekil 3.11’da Nesne konumlandırma kaybının değişimini gösteren grafik görülmektedir. Konumlandırma kaybının eğitim durdurulduğunda 0.006 olduğu görülmüştür.



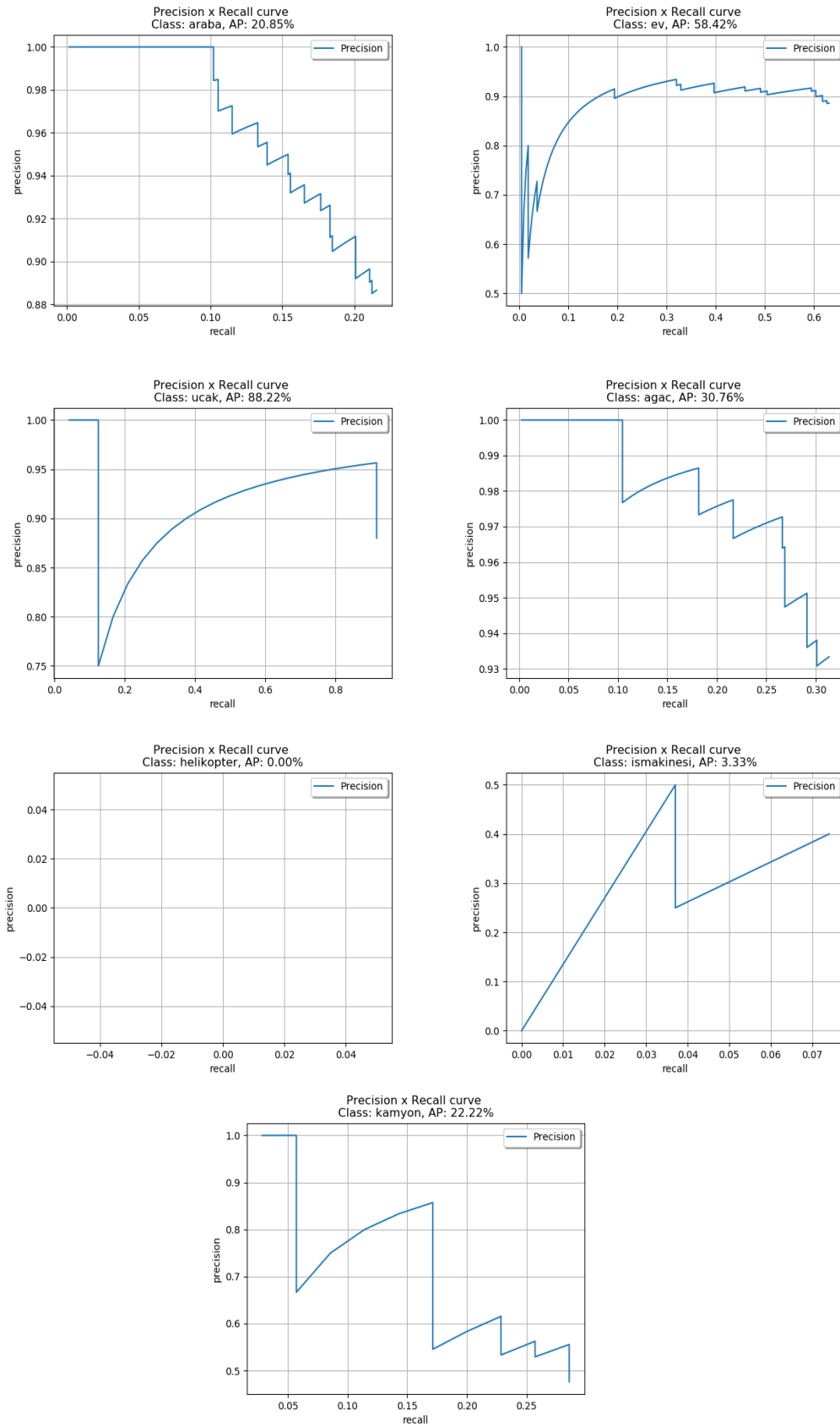
**Şekil 3.11.** Faster\_rcnn\_resnet101\_coco Konumlandırma Kaybı

Eğitim sonucu elde edilen model dosyası kullanılarak yapılan testlerde modelin başarısını ölçmek için Birleşmede Üzerinde Kesişim (IoU) değerleri 0.5 den başlayarak 0.05 artışlarla 0.95’e kadar giden 10 farklı eşik değeri için ortalama kesinlik değerleri bulunarak ortalamaları alınmıştır. Faster\_rcnn\_resnet101\_coco modeli için Birleşmede Üzerinde Kesişim değerlerine bağlı olarak bulunan ortalama kesinlik değerleri Tablo 3.3’te gösterilmiştir.

**Tablo 3.3.** Faster\_rcnn\_resnet101\_coco Ortalama Kesinlik Değerleri

IoU	0.5	0.55	0.60	0.65	0.70	0.75	0.80	0.85	0.90	0.95	Ortalama
mAP	31.97	31,79	31.66	31.55	30.58	29.94	29.53	27.92	22.7	11.07	27.87

Faster\_rcnn\_resnet101\_coco modelinin test sırasında 0.5 IoU değeri için elde edilmiş Ortalama Keskinlik değerlerinin hesaplanmasında kullanılan Keskinlik-Duyarlılık eğrilerini gösteren grafikler Şekil 3.12’de gösterilmiştir. Grafikler incelendiğinde en yüksek değerin %88.22 ile uçak sınıfına ait olduğu görülmektedir. En düşük değerin ise %0 ile helikopter sınıfına ait olduğu gözlemlenmiştir. Başarım oranı düşük olan sınıflara ait etiketlenmiş nesne sayılarının başarısı yüksek sınıflara kıyasla az olmasıdır. Etiketlenmiş nesne sayıları Tablo 2.2’de gösterilmiştir.



Şekil 3.12. Faster\_rcnn\_resnet101\_coco modelinin Kesinlik-Duyarlık Eğrileri

Şekil 3.13’de Faster\_rcnn\_resnet101\_coco modelinin test sırasında 0.5 IoU değeri için her sınıfta elde ettiği başarı değerleri gösterilmiştir.

```
(tensorflow1) C:\ozgur\Object-Detection-Metrics>python pascalvoc.py -t 0.5
AP: 30.76% (agac)
AP: 20.85% (araba)
AP: 58.42% (ev)
AP: 0.00% (helikopter)
AP: 3.33% (ismakinesi)
AP: 22.22% (kamyon)
AP: 88.22% (ucak)
mAP: 31.97%
```

**Şekil 3.13.** Faster\_rcnn\_resnet101\_coco modeli Ortalama Kesinlik Değerleri

Tablo 3.4’te test edilen 2 modelin 0.5 IoU değeri için sınıflara göre başarısı gösterilmiştir.

**Tablo 3.4.** Test edilen Modellerin Sınıflara Göre Başarıları

Model İsmi	Araba	Ev	Uçak	Ağaç	Helikopter	İş makinesi	Kamyon
Faster_rcnn_inception_v2	11	20.98	83.81	22.87	50	0	26.07
Faster_rcnn_resnet101	20.85	58.42	88.22	30.76	0	3.33	22.22

Faster\_rcnn\_resnet101\_coco modelinin Araba, Ev, Uçak, Ağaç, İş makinesi sınıflarında başarıyı arttırdığı gözlemlenmiştir. Faster\_rcnn\_inception\_v2\_coco modelinin ise helikopter sınıfına ait nesneleri tespit etmede daha başarılı olduğu görülmüştür.

#### 4. SONUÇLAR

Derin öğrenmenin yapay zekâ uygulamalarına getirdiği üstün kabiliyetler ve insansız hava araçlarının gelişimi güvenlik, trafik kontrol, arama-kurtarma, taşımacılık, tarım ve mühendislik alanlarında önemli gelişmeler sağlamıştır.

Geliştirilen sistemde, insansız hava aracı ile elde edilmiş görüntülerin derin öğrenme yöntemi kullanarak analiz edilmesiyle belirlenen özellikteki nesnelerin hızlı ve yüksek doğruluk oranıyla tespitini sağlayacak uygulama yazılımı geliştirilmiştir. Bu çalışma ile derin öğrenme ve bilgisayarlı görüyü birleştirerek bu teknolojilerin insansız hava araçları ile elde edilen görüntüler üzerindeki performansı değerlendirilmiştir.

Yapılan çalışmada; İHA ile elde edilmiş görsel veriler toplanmış, toplanan veriler etiketlenmiş ve derin öğrenme algoritmasının eğitilmesinde kullanılmıştır. Çalışma kapsamında yapay sinir ağı, sınıflandırma, öğrenme çeşitleri, uygulama alanına uygun derin öğrenme yöntemi belirleme konularında deneyim kazanılmıştır. Probleme uygun bir model seçilerek etiketlenmiş veri kullanılarak modelin tekrar eğitilmesi sağlanmıştır.

Yapılan deneylerde Faster\_rcnn\_inception\_v2\_coco ve Faster\_rcnn\_resnet101\_coco model mimarileri seçilmiş olup bu modeller Google Tensorflow model havuzunda bulunan en hızlı ve yüksek başarımlı modellerden ikisidir.

Modellerin tekrar eğitilmeleri sonucunda elde edilen mAP değerleri ve Google model havuzunda elde edilen mAP değerleri Tablo 4.1’de gösterilmiştir.

**Tablo 4.1.** Model mAP Değerleri

Model İsmi	Eğitim Sonucu mAP	Google mAP
Faster_rcnn_inception_v2_coco	25.39	28
Faster_rcnn_resnet101_coco	27.87	32

Yapılan deneylerde 240 adet görüntü üzerinde toplam 7541 adet nesne etiketlemesi yapılmıştır. Başarının artırmak için veri setinde bulunan görüntü sayısı ve etiketlenmiş

veri sayısı arttırılabilir. Ayrıca görüntünün alındığı yükseklik arttıkça küçük nesnelerin tespiti zorlaşmaktadır. 162 metreden elde edilen görüntülerde test edilen iki modelde de Arabalar tespit edilmekte zorlanmıştır. Bununla birlikte daha büyük yapıda olan ev sınıfı daha başarılı sonuçlar vermiştir. 100 ve 120 metreden çekilen resimlerde nesneler daha büyük ve net oldukları için arabalar daha iyi tespit edilebilmiştir.

Faster\_rcnn\_resnet101\_coco 88.22 AP ve Faster\_rcnn\_inception\_v2\_coco 83.81 değeriyle uçakları yüksek başarıyla tespit etmiştir. Faster\_rcnn\_resnet101\_coco 0 AP ve Faster\_rcnn\_inception\_v2\_coco 3.33 AP değeriyle iş makinelerini tespitte başarısız olmuştur. Bu sınıfa ait başarıyı artırmak için sınıfa ait etiketlenmiş veri sayısı arttırılabilir.

Tüm nesne sınıfları değerlendirildiğinde 27.87 mAP ile Faster\_rcnn\_resnet101\_coco modelinin başarısının 25.39 mAP ile Faster\_rcnn\_inception\_v2\_coco model başarısından daha yüksek olduğu görülmüştür.

Gelecek çalışmalarda, etiketlenmiş verilerin sayısı çoğaltılarak başarı değerlerinin artırılması hedeflenmektedir. Çalışmanın ilerleyen aşamalarında farklı yüksekliklerden görüntüler çekilerek daha geniş bir veri seti oluşturularak başarı değerlerinin artırılmasına ilişkin geliştirmeler yapılabilir.

## KAYNAKLAR/

- [1] Drone Resmi, <https://www.harveynorman.com.au/dji-phantom-4-pro-drone.html>, 15 Haziran 2019.
- [2] Deng, L., & Yu, D. (2014). Deep learning: methods and applications. Foundations and Trends® in Signal Processing, 7(3–4), 197-387.
- [3] Song, H. A., & Lee, S. Y. (2013, November). Hierarchical representation using NMF. In International conference on neural information processing (pp. 466-473). Springer, Berlin, Heidelberg.
- [4] Cireşan, D., Meier, U., & Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. arXiv preprint arXiv:1202.2745.
- [5] Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 580-587).
- [6] Grefenstette, E., Blunsom, P., de Freitas, N., & Hermann, K. M. (2014). A deep architecture for semantic parsing. arXiv preprint arXiv:1404.7296.
- [7] Kalchbrenner, N., Grefenstette, E., & Blunsom, P. (2014). A convolutional neural network for modelling sentences. arXiv preprint arXiv:1404.2188.
- [8] Kim, Y. (2014). Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882.
- [9] Collobert, R., & Weston, J. (2008, July). A unified architecture for natural language processing: Deep neural networks with multitask learning. In Proceedings of the 25th international conference on Machine learning (pp. 160-167). ACM.
- [10] Lee, C. S., Wang, M. H., Yen, S. J., Wei, T. H., Wu, I. C., Chou, P. C., ... & Yan, T. H. (2016). Human vs. computer go: Review and prospect [discussion forum]. IEEE Computational intelligence magazine, 11(3), 67-72.



- [11] Graves, A., & Jaitly, N. (2014, January). Towards end-to-end speech recognition with recurrent neural networks. In International conference on machine learning (pp. 1764-1772).
- [12] Karpathy, A., & Fei-Fei, L. (2015). Deep visual-semantic alignments for generating image descriptions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 3128-3137).
- [13] Graves, A., & Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural networks*, 18(5-6), 602-610.
- [14] Eck, D., & Schmidhuber, J. (2002, August). Learning the long-term structure of the blues. In International Conference on Artificial Neural Networks (pp. 284-289). Springer, Berlin, Heidelberg.
- [15] Sarıbaş, H., Çevikalp, H., & Kahvecioğlu, S. (2018, May). Car detection in images taken from unmanned aerial vehicles. In 2018 26th Signal Processing and Communications Applications Conference (SIU) (pp. 1-4). IEEE.
- [16] Wang, J., Guo, W., Pan, T., Yu, H., Duan, L., & Yang, W. (2018, July). Bottle Detection in the Wild Using Low-Altitude Unmanned Aerial Vehicles. In 2018 21st International Conference on Information Fusion (FUSION) (pp. 439-444). IEEE.
- [17] Kaster, J., Patrick, J., & Clouse, H. S. (2017, June). Convolutional neural networks on small unmanned aerial systems. In 2017 IEEE National Aerospace and Electronics Conference (NAECON) (pp. 149-154). IEEE.
- [18] Chen, J., Miao, X., Jiang, H., Chen, J., & Liu, X. (2017, October). Identification of autonomous landing sign for unmanned aerial vehicle based on faster regions with convolutional neural network. In 2017 Chinese Automation Congress (CAC) (pp. 2109-2114). IEEE.
- [19] Tri, N. C., Duong, H. N., Van Hoai, T., Van Hoa, T., Nguyen, V. H., Toan, N. T., & Snasel, V. (2017, October). A novel approach based on deep learning techniques and UAVs to yield assessment of paddy fields. In 2017 9th

International Conference on Knowledge and Systems Engineering (KSE) (pp. 257-262). IEEE.

- [20] Romero, M., Luo, Y., Su, B., & Fuentes, S. (2018). Vineyard water status estimation using multispectral imagery from an UAV platform and machine learning algorithms for irrigation scheduling management. *Computers and electronics in agriculture*, 147, 109-117.
- [21] Zhang, J. S., Cao, J., & Mao, B. (2017, July). Application of deep learning and unmanned aerial vehicle technology in traffic flow monitoring. In 2017 International Conference on Machine Learning and Cybernetics (ICMLC) (Vol. 1, pp. 189-194). IEEE.
- [22] Yao, H., Yu, Q., Xing, X., He, F., & Ma, J. (2017, July). Deep-learning-based moving target detection for unmanned air vehicles. In 2017 36th Chinese Control Conference (CCC) (pp. 11459-11463). IEEE.
- [23] Zhou, D., Zhou, J., Zhang, M., Xiang, D., & Zhong, Z. (2017, July). Deep learning for unmanned aerial vehicles landing carrier in different conditions. In 2017 18th International Conference on Advanced Robotics (ICAR) (pp. 469-475). IEEE.
- [24] Tang, T., Deng, Z., Zhou, S., Lei, L., & Zou, H. (2017, May). Fast vehicle detection in UAV images. In 2017 International Workshop on Remote Sensing with Intelligent Processing (RSIP) (pp. 1-5). IEEE.
- [25] Konoplich, G. V., Putin, E. O., & Filchenkov, A. A. (2016, May). Application of deep learning to the problem of vehicle detection in UAV images. In 2016 XIX IEEE International Conference on Soft Computing and Measurements (SCM) (pp. 4-6). IEEE.
- [26] Rey, N., Volpi, M., Joost, S., & Tuia, D. (2017). Detecting animals in African Savanna with UAVs and the crowds. *Remote sensing of environment*, 200, 341-351.
- [27] Ellenberg, A., Kontsos, A., Moon, F., & Bartoli, I. (2016). Bridge Deck delamination identification from unmanned aerial vehicle infrared imagery. *Automation in Construction*, 72, 155-165.

- [28] Kerkech, M., Hafiane, A., & Canals, R. (2018). Deep leaning approach with colorimetric spaces and vegetation indices for vine diseases detection in UAV images. *Computers and electronics in agriculture*, 155, 237-243.
- [29] Sensefly data seti, <https://www.sensefly.com/education/datasets/>, 5 Haziran 2019.
- [30] Coco Örnek Resim, <http://cocodataset.org/#explore>, 5 Haziran 2019.
- [31] LabelImg etiketleme yazılımı, <https://github.com/tzutalin/labelImg>, 5 Haziran 2019)
- [32] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8), 1735-1780.
- [33] Evrişimsel Ağlar, <https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148>, 5 Haziran 2019.
- [34] “Comparison of deep learning software.”, [https://en.wikipedia.org/wiki/Comparison\\_of\\_deep\\_learning\\_software](https://en.wikipedia.org/wiki/Comparison_of_deep_learning_software), 01.05.2019.
- [35] Nesne tespit algoritmaları, <https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e>, 5 Haziran 2019.
- [36] Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 580-587).
- [37] Uijlings, J. R., Van De Sande, K. E., Gevers, T., & Smeulders, A. W. (2013). Selective search for object recognition. *International journal of computer vision*, 104(2), 154-171.
- [38] Stanford Üniversitesi, [http://cs231n.stanford.edu/slides/2017/cs231n\\_2017\\_lecture11.pdf](http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture11.pdf), 5 Haziran 2019.

- [39] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In Advances in neural information processing systems (pp. 91-99).
- [40] Medium Web Sitesi, <https://medium.com/@tuncerergin/convolutional-neural-network-convnet-yada-cnn-nedir-nasil-calisir-97a0f5d34cad>, 5 Haziran 2019.
- [41] Medium Web Sitesi, <https://medium.com/@bakiiii/microsoft-sunar-deep-residual-network-d2970003ad8b>, 5 Haziran 2019.
- [42] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).
- [43] Towardsdatascience Web Sitesi, <https://towardsdatascience.com/review-resnet-winner-of-ilsvrc-2015-image-classification-localization-detection-e39402bfa5d8>, 5 Haziran 2019.
- [44] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).
- [45] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2818-2826).
- [46] Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. A. (2017, February). Inception-v4, inception-resnet and the impact of residual connections on learning. In Thirty-First AAAI Conference on Artificial Intelligence.
- [47] Towardsdatascience Web Sitesi, <https://towardsdatascience.com/an-intuitive-guide-to-deep-network-architectures-65fdc477db41>, 5 Haziran 2019.
- [48] Github Web Sitesi, <https://github.com/tensorflow/models>, 5 Haziran 2019.
- [49] Anaconda Web Sitesi, <https://www.anaconda.com/distribution/>, 5 Haziran 2019.

## ÖZGEÇMİŞ

**Adı Soyadı** : Özgür KUTLU  
**Doğum Yeri ve Tarihi** : Balıkesir - 1979  
**Yabancı Dili** : İngilizce  
**E-Posta** : kutlu\_ozgur@hotmail.com

### Öğrenim Durumu

Derece	Bölüm/Program	Üniversite/Lise	Mezuniyet Yılı
Lise	Teknik Lise	100.Yıl Meslek Lisesi	1997
Lisans	Bilgisayar-Kontrol Eğitimi	Marmara Üniversitesi	2002

### İş Deneyimi

Yıl	Firma/Kurum	Görevi
2002	M.E.B.	Öğretmen