T.C.
BİTLİS EREN ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI
YÜKSEK LİSANS TEZİ

İNSANSIZ HAVA ARACI İLE GERÇEK ZAMANLI NESNE TANIMA

Gökhan KESKİNTAŞ

ŞUBAT 2021

ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI
YÜKSEK LİSANS TEZİ

İNSANSIZ HAVA ARACI İLE GERÇEK ZAMANLI NESNE TANIMA

Hazırlayan
Gökhan KESKİNTAŞ

Danışman
Dr. Öğr. Üyesi Musa ÇIBUK

Jüri Üyeleri
Prof. Dr. Sabir RÜSTEMLİ
Dr. Öğr. Üyesi Musa ÇIBUK
Dr. Öğr. Üyesi İhsan TUĞAL

ŞUBAT 2021

ONAY

Gökhan KESKİNTAŞ tarafından hazırlanan "**İnsansız Hava Aracı ile Gerçek Zamanlı Nesne Tanıma**" adlı tez çalışması 18 / 02 / 2021 tarihinde yapılan sınavla aşağıdaki jüri tarafından oybirliği/oyçokluğu ile Bitlis Eren Üniversitesi Lisansüstü Eğitim Enstitüsü Elektrik Elektronik Mühendisliği Anabilim Dalı'nda YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

Jüri Üyeleri	**İmza**
Dr. Öğr. Üyesi Musa ÇIBUK (Danışman)	_____
Prof. Dr. Sabir RÜSTEMLİ (Başkan)	_____
Dr. Öğr. Üyesi İhsan TUĞAL (Üye)	_____

Bu tezin kabulü, Lisansüstü Eğitim Enstitüsü Yönetim Kurulu'nun …. /… /2021 gün ve …./…. sayılı kararı ile onaylanmıştır.

Prof. Dr. Zeki ARGUNHAN
Enstitü Müdürü

BİTLİS EREN ÜNİVERSİTESİ LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ
YÜKSEK LİSANS TEZ ÇALIŞMASI
ETİK BEYANI

Bitlis Eren Üniversitesi Lisansüstü Eğitim Enstitüsü tez yazım kılavuzuna göre hazırlamış olduğum "**İnsansız Hava Aracı ile Gerçek Zamanlı Nesne Tanıma**" adlı tezimin özgün bir çalışma olduğunu, tez hazırlanırken tüm aşamalarda bilimsel etik ilkelerine uygun davrandığımı, tez kapsamında sunulan tüm verileri bilimsel etik ilkelerine uygun elde ettiğimi, tezde faydalandığım tüm eserlere atıf yaptığımı ve kaynaklar kısmında bu eserleri gösterdiğimi beyan ederim. 18 / 02 / 2021

Gökhan KESKİNTAŞ

ÖZET

İNSANSIZ HAVA ARACI İLE GERÇEK ZAMANLI NESNE TANIMA

Gökhan KESKİNTAŞ

Yüksek Lisans Tezi

Bitlis Eren Üniversitesi Lisansüstü Eğitim Enstitüsü
Elektrik Elektronik Mühendisliği Anabilim Dalı
Danışman: Dr. Öğr. Üyesi Musa ÇIBUK
Şubat 2021, 65 sayfa

Yapay zekâ alanında bugüne kadar birçok çalışma yapılmış ve bu çalışmalar neticesinde birçok algoritma geliştirilmiştir. Donanım üreticileri de her geçen gün daha iyi donanımlar üretmektedirler. Bugün geldiğimiz noktada yüksek donanım gücü sayesinde, daha önce gerçekleştirilmesi mümkün olmayan algoritmaları kullanan uygulamalar geliştirilmektedir. Bilgisayarla görü alanındaki denetimli öğrenme algoritmalarının gerçekleştirilmesi için ihtiyaç duyulan büyük veri miktarı, cep telefonu kameraları ve internet sayesinde elde edilebilir bir hale gelmiştir. Bu sayede birçok farklı sınıfa ait görüntüler içeren PASCAL VOC, ILSVRC, COCO gibi açık erişimli görsel veri setleri oluşturulmuştur. Fakat bu veri setleri insansız hava aracıyla görüntü işleme için ihtiyaç duyulan kuşbakışı perspektifinden çekilen görüntüleri içermemektedir.

Bu tez çalışmasında insansız hava araçları ile gerçek zamanlı görüntü işleme amaçlanmıştır. Bu amaçla insansız hava araçları ile çekilmiş görüntülerden oluşan özgün bir veri seti oluşturulmuştur. Gerçek zamanlı görüntü işleme hızına sahip bir evrişimsel sinir ağı modeli olan YOLO v4 seçilmiş ve oluşturulan veri seti ile eğitilmiştir. Geliştirilen uygulama ile insansız hava aracından alınan görüntüler YOLO v4 Modeli ile işlenerek, veri setinin içerdiği “insan” ve “koyun” sınıfları özelinde, gerçek zamanlı nesne tanıma yapılmıştır.

Anahtar kelimeler: İnsansız Hava Araçları, Evrişimsel Sinir Ağları, Yolo, Nesne Tanıma

ABSTRACT

REAL TIME OBJECT DETECTION WITH UNMANNED AERIAL VEHICLE

Gökhan KESKİNTAŞ

Master Thesis

Bitlis Eren University Graduate Education Institute
Department of Electrical and Electronics Engineering
Supervisor: Asst. Prof. Dr. Musa ÇIBUK
February 2021, 65 pages

Many studies have been carried out in the field of artificial intelligence and many algorithms have been developed as a result of these studies. Hardware manufacturers are also producing better hardware every day. Thanks to hardware power we have reached today, applications using algorithms that couldn't be run on hardware before are being developed. The large amount of data needed for the realization of supervised learning algorithms in the field of computer vision has become available thanks to mobile phone cameras and the internet. With these data, publicly accessible visual data sets such as PASCAL VOC, ILSVRC, COCO containing images of many classes have been created. However, these datasets do not include the bird's-eye perspective images needed for drone image processing.

In this thesis, real-time image processing with unmanned aerial vehicles is aimed. For this purpose, a unique data set consisting of images taken by unmanned aerial vehicles was created. YOLO v4, a convolutional neural network model with real time image processing speed, was chosen and trained with the created dataset. With the developed application, the images taken from the unmanned aerial vehicle were processed with the YOLO v4 Model and real-time object detection was achieved on "person" and "sheep" classes that are included in dataset.

Keywords: Unmanned Aerial Vehicles, Convolutional Neural Network, Yolo, Object Recognition

TEŞEKKÜR

Yoğun geçen tez çalışması sırasında, tez konusunun belirlenmesinden başlayarak son aşamaya kadar her türlü bilgi, teşvik ve deneyimlerini benden esirgemeyen danışman hocam sayın Dr. Öğr. Üyesi Musa ÇIBUK'a şükranlarımı sunarım. Yüksek lisans eğitimim boyunca yardımlarından dolayı Elektrik-Elektronik Mühendisliği Anabilim dalı başkanı sayın Prof. Dr. Sabir RÜSTEMLİ hocama teşekkür ederim.

Bu günlere gelmemde büyük emekleri olan başta ailem olmak üzere, çalışma temposu boyunca her daim bana destek olan eşime teşekkürlerimi sunarım.

ÖNSÖZ

Bu tez çalışmasında, son yıllarda oldukça popüler olan evrişimsel sinir ağları kullanılarak insansız hava aracından alınan görüntüler üzerinde gerçek zamanlı nesne tanıma yapılmıştır. İnsansız hava aracından anlık olarak alınan görüntüleri, gerçek zamanlı işleyebilmek için, tek aşamalı evrişimsel sinir ağı nesne tanıma modellerinden, YOLO v4 modeli tercih edilmiştir.

İnsansız hava aracından alınan görüntüler üzerinde nesne tanımada yüksek başarım oranı elde edebilmek için insansız hava aracı ile çekilen görüntülerden oluşan orijinal bir veri seti oluşturulmuş ve YOLO v4 modeli, bu eğitim setiyle eğitilmiştir. İnsansız hava aracının uçuşunu gerçekleştirebilmek, kamerasından anlık olarak görüntüleri elde etmek ve nesne tespiti yapabilmek için Visual Studio platformunda C# dili ile UWP mimarisinde bir uygulama geliştirilmiştir.

İÇİNDEKİLER DİZİNİ

Sayfa

ÇİZELGELER DİZİNİ

ÇİZELGE **Sayfa**

ŞEKİLLER DİZİNİ

SİMGELER DİZİNİ

Σ	Toplam Sembolü
n	Örnek Sayısı
log	Logaritma
$tanh$	Hiperbolic Tanjant
$\in$	Elemanı
R	Gerçek Sayılar
σ	Sigmoid Fonksiyonu
λ_{coord}	Fonksiyon Ağırlık dengeliyici
w	Genişlik değeri
$\hat{w}$	Genişlik değeri tahmini
h	Yükseklik değeri
$\hat{h}$	Yükseklik değeri tahmini
x	Yatay koordinat değeri
$\hat{x}$	Yatay koordinat tahmini
y	Düşey koordinat değeri
$\hat{y}$	Düşey koordinat değeri tahmini
y_i	i indisli elemanın yükseklik değeri
$\acute{y}_i$	i indisli elemanın yükseklik tahmini
1_{ij}^{obj}	i hücresindeki j sınırlayıcı kutusundaki nesne varlığı
1_{ij}^{noobj}	i hücresindeki j sınırlayıcı kutusundaki nesne yokluğu
$\widehat{\boldsymbol{C}}_{\boldsymbol{i}}$	i hücresinin güven tahmini
$\boldsymbol{C}_{\boldsymbol{i}}$	i hücresinin güven skoru
Pr(object)	Nesne olma ihtimali

KISALTMALAR DİZİNİ

ILSVRC	ImageNet Geniş Ölçekli Görsel Tanımlama Yarışması	ImageNet Large Scale Visual Recognition Competition
CPU	Merkezi İşlem Birimi	Central Processing Unit
GPU	Grafik İşlem Birimi	Graphics Processing Unit
İHA	İnsansız Hava Aracı	Unmanned Air Vehicle
SİHA	Silahlı İnsansız Hava Aracı	Armed Unmanned Air Vehicle
DC	Doğru Akım	Direct Current
ESC	Elektronik Hız Kontrolörü	Electronic Speed Controller
Bec	Batarya Eleme Devresi	Battery Eliminator Circuit
GPS	Küresel Konumlama Sistemi	Global Positioning System
PWM	Darbe Genişlik Modülasyonu	Pulse Width Modulation
Lipo	Lityum polimer pil	Lithium Polymer Battery
mAh	Miliamper-Saat	Miliamper-Hour
YSA	Yapay Sinir Ağları	Artificial Neural Network (ANN)
ESA	Evrişimsel Sinir Ağı	Convolutional Neural Network (CNN)
ReLU	Düzleştirilmiş Lineer Birim	Rectified Linear Unit
LReLU	Sızıntılı ReLU	Leaky ReLU
IoU	Birleşim Üzerindeki Kesişim	Intersection over Union
DVM	Destek Vektör Makinesi	Support Vector Machine
Fps	Saniyedeki Çerçeve Sayısı	Frames Per Second
DPM	Deforme Edilebilir Parça Bazlı Model	Deformable Part Based Model
UWP	Evrensel Windows Platformu	Universal Windows Platform

1. GİRİŞ

Makineler uzun yıllardır hayatımızı kolaylaştıran ve hayat konforumuzu oluşturan en önemli unsurlardan biridir. İnsan hayatını kolaylaştırmak için her alanda çeşitli makineler kullanılmaktadır. Makineler sayesinde artık haftalar, aylar süren uzun yolculuklar geçmişte kalmıştır. Günümüzde arabalar ve uçaklar gibi insan yapımı makineler sayesinde saatler içinde istediğimiz yerlere gidebilmekteyiz. Makineler, elektronik ve elektroniğin ruhu olan yazılımları sayesinde birçok problemi çözer niteliklere kavuşurlar. Bu yazılımlar, programcı tarafından öngörülen durumlara karşı makinenin vereceği reaksiyonları belirler. Birçok durumda insan müdahalesine muhtaç durumdadır.

Makinelerde insan müdahalesini minimuma indirmek önemli bir konudur. Öngörülmemiş durumlarda daha önce karşılaştığı durumlara bakarak makinenin karar vermesini sağlamak amacıyla yapay zekâ alanında birçok çalışma yapılmıştır.

Uzun yıllar boyu yapılan çalışmalar ve algoritmalar bilgisayar donanımı yetersizliğinden uygulamada gerçekleştirilememiştir. Günümüzde, bilgisayar donanımının gelişmesi, Merkezi İşlem Birimi (CPU) ve özellikle Grafik İşlem Birimi (GPU) alanındaki yapılan gelişmeler sayesinde yüksek işlem gücü gerektiren algoritmaların uygulanması mümkün hale gelmiştir. Nvidia [1], intel [2] gibi donanım alanında dünya devi olan firmalar yapay zekâ destekli yongalar üretmektedirler. Yazılım, donanım üzerinde çalıştığı için bu yonga setlerini destekleyen kütüphanelerin yazılım dünyasına dâhil olması, yüksek hız gerektiren mevcut algoritma ve modellerin yanı sıra, yeni algoritma ve modellerin tasarlanmasına, uygulanmasına ve bu alanda yapılan çalışmaların artmasına sebep olmuştur.

Yapay zekânın bir alt dalı olan denetimli makine öğrenmesi için gerekli algoritma ve modeller yüksek miktarda veriye ihtiyaç duyar. İnternetin gelişmesi neticesinde ihtiyaç duyulan verinin ortaya çıkması ve kullanılması kolaylaşmıştır. Kameraların gelişmesi ve ucuzlaması, cep telefonu ile çekilen görüntülerin internet ortamına yüklenmesi, video içerik sağlayıcılarına üretilen videolar neticesinde, özellikle bilgisayarla görü alanında ihtiyaç duyulan resim, görüntü gibi verilerin elde edilmesi çok daha kolaylaşmıştır.

Son yıllarda insansız hava aracı (İHA) üzerinde de büyük gelişmeler yaşanmıştır. Daha önceleri yalnızca askeri amaçla kullanılan insansız hava araçları, günümüzde sivil kullanımda da yaygınlaşmıştır [3]. Özellikle insansız hava araçlarının döner kanatlı modelleri birçok firma tarafından sivil kullanım amacıyla üretilmekte ve talep görmektedirler. Motor, motor sürücüsü, batarya gibi gerekli donanımlar satın alınarak istenilen faydalı yüklerle(kamera, sensörler…)

gerekli tasarım yapılarak ihtiyaca göre İHA yapılabiliyor olması, birçok alanda kullanılabilirliğinin olması, yazılım ile kontrol edilebilir olması bu alanda yapılan çalışmaları değerli kılmaktadır.

1.1. Literatür Taraması

Makinelere zekâ özelliğini katabilmek için insan ve hayvanların sinir sistemi taklit edilmeye çalışılmıştır. Sinir sistemi üzerine yapılan çalışmalar dokunma, tatma, duyma, koklama ve görme sistemlerinin veri iletimi ve çalışma ilkelerini daha iyi anlamamızı sağlamıştır.

İlk yapay sinir ağı modeli Warren McCulloch ve Walter Pitts tarafından 1943 yılında geliştirilmiştir [4].

1950'lerde Alan Turing tarafından ortaya atılan Turing Testi [5], bir makinenin insan gibi düşünebildiğini söyleyebilmenin mantıksal olarak mümkün olup olmadığını test etmeye yarar. Bu testte deneyde yer alan kişi, bir insan ve bir bilgisayarla, hangisinin bilgisayar hangisinin insan olduğunu bilmeden bir klavye aracılığıyla yazışır. Yapılan birkaç testte deneydeki kişi tutarlı bir şekilde hangisinin bilgisayar olduğunu bilemezse Turing Testi başarılmış sayılır.

Bernand Widrow ve Marcian Hoff tarafından 1959 yılında Stanford Üniversitesinde, yapay sinir ağlarının mühendislikteki uygulamalara yönelik ilk çalışmalar olan ADALINE ve MADALINE yapay sinir ağı modelleri geliştirilmiştir [6]. ADALINE modeli kendinden sonraki yapay sinir ağı çalışmalarında referans alınmıştır. MADALINE telefon hatlarında oluşan ses yankılarını yok eden filtre amacıyla kullanılmıştır. Gerçek problemlere uygulanmış ilk sinir ağı olup hala kullanılmaktadır [7].

Hubel ve Wiesel, 1950'lerde ve 1960'larda yaptıkları çalışmalarda, kedi ve maymun görsel kortekslerinin belirli bölgelerinin, görme alanının belirli bölgelerine reaksiyon verdiklerini keşfettiler [8].

Fukushima, Hubel ve Wiesel'in hayvan görsel korteksi üzerindeki çalışmalarından esinlenerek 1980 yılında Neocognitron modelini geliştirmiştir [9].

1986 yılında Rumelhart ve arkadaşları tarafından, yapay sinir ağları için büyük öneme sahip olan kayıp fonksiyonunun kayıp oranını önceki katmanlara dağıtan geri yayılım öğrenme algoritması ortaya konulmuştur [10].

Rumelhart ve arkadaşları tarafından tanıtılan geriyayılımlı öğrenme algoritması, 1989 yılında Y. Lecun tarafından, elle yazılmış posta kodu numaralarını tanımlayan evrişimsel sinir ağının eğitimi için kullanılmıştır [11].

1997 yılında satranç dünya şampiyonu Kasparov, IBM tarafından geliştirilen Deep Blue adlı süper bilgisayar tarafından yenilmiş ve yapay zekânın insana karşı başardığı bir olgu olarak büyük yankı uyandırmıştır [12].

2004 yılında Oh ve Jung sinir ağlarının GPU kullanılarak hızlandırılabileceğini göstermişlerdir. Oluşturdukları sinir ağının GPU versiyonu, CPU versiyonuna göre 20 kat daha hızlı sonuç üretmiştir [13].

2005 yılında Pascal Voc Görsel Nesne Tanıma yarışması yapılmış ve 2012 yılında Alex Krizhevsky ImageNet yarışmasını, geliştirdiği evrişimsel sinir ağını kullanarak kazanmıştır. Bu tarihten itibaren bu alanda yapılan çalışmalar ivme kazanmış, Evrişimsel Sinir Ağları'nın eğitim ve test başarısını arttıran birçok çalışma yapılmıştır.

2014 yılında GoogleNet [14], R-CNN Evrişimsel Sinir Ağı Modeli [15], 2015 yılında ise YOLO Evrişimsel Sinir Ağı Modeli [16] tanıtılmıştır.

Öte yandan bu alandaki çalışmalar her geçen gün artmakla beraber araştırmacılar tarafından birçok yeni evrişimsel sinir ağı modeli literatüre kazandırılmıştır [17].

1.2. Tez Organizasyonu

Bölüm 1'de tez çalışması konusunun önemi ile ilgili giriş bilgilerine ve literatür taramasına yer verilmiştir.

Bölüm 2'de IHA tanımı, uygulama alanları ve donanımı ile ilgili genel bilgilere yer verilmiştir. Ayrıca yapay sinir ağları, evrişimsel sinir ağları, nesne tespiti için kullanılan bazı evrişimsel sinir ağları modellerinden bahsedilmiştir. Nesne tanıma için kullanılan İHA, donanımlar, veri seti ve uygulama ile ilgili bilgiler verilmiştir.

Bölüm 3'te tezde kullanılan veri setinin başarım oranı değerlendirilmiş ve İHA'dan alınan görüntülerin mevcut donanımımızla işlenme hızı ile ilgili bilgiler verilmiştir. Bölüm 4'te Sonuç kısmına ve ilerde neler yapılabileceğine dair yoruma yer verilmiştir.

2. MATERYAL VE YÖNTEM

2.1. İnsansız Hava Araçları

Birçok kurum ve kuruluş tarafından birbirinden farklı İHA tanımları yapılmıştır. NATO tarafından yapılan tanımlamaya göre; "İHA, içerisinde insan olmayan, uzaktan kumanda ile yönlendirilen veya otonom olarak kendisini yönlendiren motorlu itki gücü olan, silah veya faydalı yükleri ana gövdesine yüklenip çıkarılabilen, görev sonu geri dönerek iniş yapabilen veya hedefte silah olarak kendini imha edebilen araçlardır [18]." Uçan balonlar ve seyrüsefer füzeleri İHA olarak kabul edilmemektedir.

Ülkemizde sivil hava sahasında kullanılanan İHA'ların uçuş izinleri, kayıt ve tescil işlemleri Sivil Havacılık Genel Müdürlüğü [19] tarafından yürütülmektedir.

2.1.1. İnsansız Hava Araçlarının Sınıflandırması

Günümüzde yalnızca ayrılmış hava sahalarında faaliyet gösteren İHA'ların değişik özellikleri dikkate alınarak, çeşitli kuruluş ve ülkeler tarafından birbirinden farklı sınıflandırmalar yapılmıştır. Bu sınıflandırma modellerine göre bir İHA modeli birden fazla sınıflama modeli içinde bulunabildiği gibi aynı sınıflama modelinde birden fazla kategoriye ait özellikleri de taşıyabilmektedir [18].

Bu sınıflandırmalar kısaca şu şekilde özetlenebilir.

- Büyüklük, irtifa, uçuş süresi ve faydalı yük kapasitesine göre (Micro-Mini-Küçük-Taktik- Operatif-Stratejik)
- Faydalı yük türüne göre (Silahlı İHA'lar – Silahsız İHA'lar)
- Yakıt türüne göre (İçten yanmalı motorlu – Elektrik motorlu)
- Uçuş yöntemine göre (Sabit Kanatlı – Döner Kanatlı)
- Komuta biçimine göre (Otomatik pilotlu – Uzaktan komutalı)
- Kullanım amacına göre (Sahte/Hedef – Keşif Gözetleme – Atak/Saldırı – Lojistik destek)
- Kalkış ve iniş yöntemine göre (Rampadan kalkan/fırlatılan – Pistten kalkan – Uçaktan bırakılan – elle fırlatılan gövde üzerine iniş yapan - paraşütle iniş yapan) [18]

2.1.2. İHA Kullanım Alanları

İHA'nın birçok avantajı ve potansiyel uygulama alanı mevcuttur. Hava aracı, engebeli yüzeyler, dağlar gibi yeryüzü şekillerinden etkilenmez. Karayolundan gitmenin saatler sürebileceği bir yere, hava yolu ile dakikalar içinde gidilebilir. Bu özelliğinden dolayı günümüzde İHA ile kargo taşımacılığı üzerine çalışmalar yapılmaktadır [20].

İçerisinde insan taşımadığı için; insanın fiziksel olarak ihtiyaç duyacağı oksijen ayarlayan, basınç dengeleyen yaşam üniteleri İHA'larda bulunmaz. Bu nedenle insanlı modellere göre İHA'ların karmaşıklıklığı ve maliyeti düşük olur. Karmaşıklığın azaltılmış olması maliyetlerin yanı sıra hata ve arıza oranının da azalmasını sağlar. Bu yaşam destek ünitelerinin yokluğu aynı zamanda hava aracının daha hafif olmasını sağlayacağından, havada kalmak için ihtiyaç duyulan güç azalır ve bunun sonucu olarak, hava araçlarında enerji tüketim miktarı azalır. Bu nedenle çok daha uzun süreler havada kalabilir. Bugün itibariyle en uzun süre havada kalabilen uçakların rekoru devasa yakıt tanklarına rağmen; 20 saatin altında iken, İHA'lar 20 saatin çok üzerinde sürelerde başarıyla havada kalabileceklerini göstermişlerdir [21]. Bu nedenle uzun süren keşif/gözetleme görevlerinde İHA tercih edilir. Örnek bir İHA Şekil 2.1'de gösterilmiştir.

Şekil 2.1. Tusaş Anka

Pilotun hava aracının içinde değil de Yer Kontrol İstasyonunda bulunması neticesinde hava aracı kırıma uğrasa bile pilotu bir zarar görmeyecektir. Pilotun yetişmesi yıllar süren, maliyetli bir süreç olduğundan pilot kaybından kaynaklı maddi ve manevi kayıp yaşanmayacaktır. Pilotun yer kontrol istasyonunda bulunması özellikle uzun uçuş görevleri sırasında pilotun değiştirilebilmesine de imkân sağlar. Böylelikle bir insanın çalışabileceği maksimum saatin aşılması, yorgunluk, akli/bedeni rahatsızlıklar, fiziksel ihtiyaçlar gibi etkenlerin görev başarımı üzerine negatif etkisi azaltılmış olur. Ayrıca pilot, insanın fiziksel sınırlarından kaynaklı çok yüksek g kuvvetine dayanamaz. Bundan dolayı yapamayacağı keskin manevralar, İHA tarafından kolayca yapılabilir.

Günümüz modern savaş uçakları doğası ve tasarımı gereği yüksek miktarda yakıt tükettiğinden üzerindeki mühimmatı bir an önce hedefe bırakıp, piste geri dönmelidir. Bu nedenle savaş uçakları için hedef seçimi uçak pistten havalanmadan önce yapılır. Hedef seçiminin sağlıklı ve atışların isabetli olabilmesi için hedef bölgesinin gözetlenebiliyor olması, daha da iyisi hedefin bir müttefik tarafından lazerle işaretlenmiş olması gerekir. Uzun süre havada kalabilme özelliğinden dolayı gözetleme amacıyla kullanılan İHA tarafından lazerle işaretlenen hedefler savaş uçakları tarafından vurulabilir. Savaş uçaklarının operasyon maliyeti yüksek olduğu için, saatlerce havada kalabilen, kanatlarına füze entegre edilmiş silahlı insansız hava aracı(SİHA), çok daha düşük maliyetle hem gözetleme hem de saldırı amacıyla kullanılabilir.

İHA, ihtiyaca göre istenilen boyutlarda ve malzemeden tasarlanabilir. Askeri alanda kullanılan radarlar; belirli bir mesafede, belirli bir büyüklüğün üzerindeki cisimleri tespit edebilir, mesafe arttıkça küçük nesneleri tespit etmekte zorlanırlar. Küçük boyutlu İHA'ların radarlar tarafından fark edilebilmesi ve vurulma ihtimali daha düşüktür. İnsanlı modellere göre düşük maliyette üretilebiliyor olmalarından dolayı, gözetleme amacıyla kullanılan bir İHA'yı düşürmek için fırlatılan füze, hava aracının kendinden pahalı bile olabilir. Küçük boyutlu İHA'ların, kırımı durumunda büyük hasarlar oluşturmaması, sosyal yaşam bölgelerinde bile kullanılmasını mümkün kılmıştır. Piyasada kapalı ve dar alanlarda bile sorunsuz uçabilen görece küçük boyutlu değişik model ve markalarda İHA'lar mevcuttur. İHA'lar, tasarımına göre küçük bir yüzey alanına sahip düz bir yerden, elle veya mancınık ile kalkış yapabilir, iniş sırasında da küçük düz bir yüzey alanına inebilir veya elle tutulabilir, bu nedenle pist ihtiyacı da ortadan kalkabilir. Şekil 2.2'de zirai amaçla kullanılan döner kanatlı bir İHA gösterilmiştir.

Şekil 2.2. Zirai Amaçla Kullanılan Döner Kanatlı Bir İHA

Yukarıda anlatılanların yanı sıra; İHA'lar kamera çekimi, yangın söndürme, taşımacılık, afet yönetimi, arkeolojik araştırma, harita yapımı, telekomünikasyon, meteoroloji, arama-kurtarma faaliyetleri, veri toplama, deniz devriyesi, balıkçılık izleme, kıyı şeridi izleme, uluslararası sınır devriyesi, uyuşturucu trafiği izleme, yol trafiği izleme ve kontrolü, kanun uygulamaları, orman

yangını tespiti, ekin ve hasat durumu izleme, çevre durumu izleme, arazi haritalama, yüksek voltajlı güç hattı izleme, yağ kaçağı gözleme, sel izleme, kasırga izleme, afet operasyon yönetimi, felaket durum değerlendirmesi, arama kurtarma, yangınla mücadele, nükleer radyasyon gözleme, deprem gözleme, volkan gözleme, atmosfer araştırması, okyanus gözlemleri, jeolojik araştırmalar, hava durumu tahmini gibi pek çok alanda kullanılabilir [3, 18, 22].

2.1.3. İHA Sistemi

İHA Sistemi en temelde, Hava aracı ve Yer Kontrol İstasyonundan oluşur. İHA Sistemi'nde Hava aracı ile görev esnasında Yer Kontrol İstasyonu arasında iletişim kurulmak isteniyorsa bunların yanısıra Veri Linki de içermelidir [18]. Şekil 2.3'te Yer Kontrol İstasyonu ve İHA arasında Veri Linki aracılığıyla görüş hattında iletişim gösterilmiştir.

Şekil 2.3. Yer Kontrol İstasyonu-Veri Linki-Hava Aracı

2.1.3.1. Yer Kontrol İstasyonu

Yer Kontrol İstasyonu; uçuşun ve görevin belirlendiği birimdir. Yer Kontrol İstasyonu anlık olarak hava aracı ile veri iletişimi sağlayabilir ya da istasyonda otonom uçuş rotası ayarlanan hava aracı, veri bağlantısı ihtiyacı olmadan belirlenen rota güzergâhını takip edebilir. Telemetri verilerinin değerlendirilmesi, görev esnasında hava aracının kontrölü ve görevin takibi buradan yapılır.

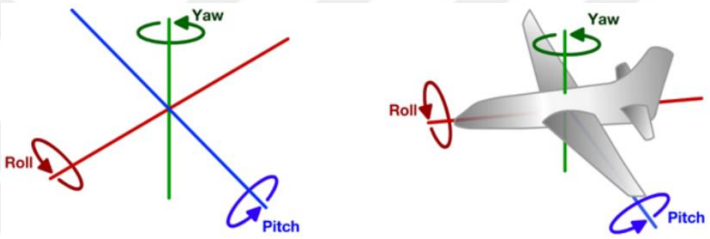
2.1.3.2. Veri Linki

Hava Aracı ile anlık veri iletişimi sağlanmak isteniyorsa, Hava aracı üzerinde ve Yer Kontrol İstasyonunda, birbiriyle iletişimi sağlayacak şekilde özelleşmiş veri aktarım donanımları ve antenleri bulunmak zorundadır. Bu donanım ve antenlerin Hava aracı ile Yer Kontrol İstasyonu

arasında veri iletimi tasarıma ve ihtiyaca göre görüş hattında ya da uydu aracılığıyla görüş hattının ötesinde gerçekleşebilir.

2.1.3.3. Hava Aracı

Hava aracı; aerodinamik ilkeleri ve kuralları göz önünde bulundurularak tasarlanan güçlü ve hafif olması istenen iskelet sistemi ve bu iskelet sistemi üzerinde taşınan; itki sağlayan elektrik motoru ve batarya ya da jet motoru ve yakıt tankı, gerek uçuşun sağlıklı gerçekleşmesini sağlayan gerekse görev için gerekli olan algılayıcı sensörler, kameralar, veri bağlantı sistemleri, füzeler gibi faydalı yüklerden oluşur. Bu faydalı yüklerin neler olacağı insansız hava aracının gerçekleştireceği göreve ve tasarımına göre değişkenlik gösterir.



Şekil 2.4. Hava Aracının 3 Eksenli Hareketleri

Şekil 2.4'te İHA'nın havada gerçekleştirebileceği hareketler gösterilmiştir. Bu hareketler Yuvarlanma, Yunuslama ve Dönme olmak üzere 3'e ayrılır.

Roll (Yuvarlanma) ekseni: Hava aracının ön ile arka kısmını birleştiren hayali hattır. Hava aracının, sağa sola yatış hareketleri bu eksen üzerinde olur.
Pitch(Yunuslama) ekseni: Hava aracının sağ ve sol kısımlarını birleştiren hayali hattır. Hava aracının, burun aşağı ve burun yukarı hareketleri bu eksen üzerinde gerçekleşir.
Yaw (Dönme) ekseni: Hava aracını üstten alta doğru delen eksendeki hayali hattır. Hava aracının sağa ve sola dönüş hareketleri bu eksen üzerinde olur.

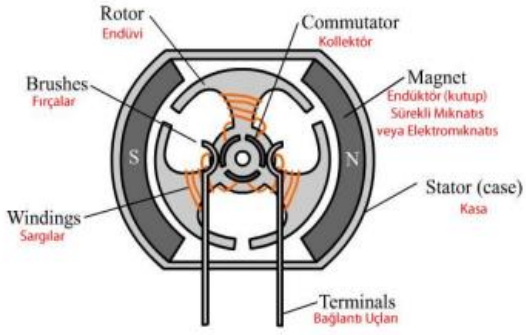
2.1.4. Quadrotorlar

Şekil 2.5'te bir örneği olan, dört adet DC (Doğru Akım) motor ile itki gücü sağlayan, Quadrotorlar, insansız hava araçları arasında en çok tercih edilendir. Quadrotorlar, hızlı manevra kabiliyeti ve küçük yapısıyla birçok alanda kullanılmaktadır. Quadcopter olarak da bilinen

quadrotorların, üç motorlu tricopter, altı motorlu hexacopter ve sekiz motorlu octocopter gibi farklı motor sayısında çalışan modelleri bulunur. Quadrotor, fırçasız doğru akım motoru, ESC (Electronic Speed Controller), batarya, mikrokontroller ve sensörler (ivmeölçer, gyroscope, manyetometre, barometre, gps vb.) olmak üzere 5 temel bileşenden oluşmaktadır [23].

Şekil 2.5. Quadrotor

Quadrotorların birçoğu elektrik motorlarıyla itki gücü sağlarlar. Elektrik Motorları elektrik enerjisini mekanik enerjiye dönüştürür. Her elektrik motoru biri sabit (Stator) ve diğeri kendi çevresinde dönen (Rotor) iki ana parçadan oluşur. Doğru akım motorlarının çalışma ilkesi ilk kez Faraday tarafından, "Manyetik alan içerisinde bulunan bir iletkenden elektrik akımı geçirilirse iletken hareket eder." şeklinde açıklanmıştır [24]. İnsansız hava araçlarında etkili performans ve kolay kontrolünden dolayı sıklıkla DC motorlar kullanılır. DC motorların dönüş hızı kaynak gerilimi ile doğru orantılıdır. Motora uygulanan ortalama gerilim arttırılarak ya da azaltılarak hız kontrolü sağlanabilir. Kaynak gerilim, motorun hissetmeyeceği frekansta, transistörler aracılığıyla anahtarlanır, PWM(Darbe Genişlik Modülasyonu) değiştirilir ve motor istenilen hızda sürülür. Motorlar; fırçalı ve fırçasız olmak üzere 2 kategoriye ayrılır. Örnek bir fırçalı DC motor içyapısı Şekil 2.6'da verilmiştir.



Şekil 2.6. Fırçalı DC Motor Yapısı [24]

Fırçalı DC motorlarda, komütasyon fırça ile yapılır. Fırça fiziksel temasta bulunduğu için zamanla sürtünmeden kaynaklı aşınmanın sonucu olarak; fırçanın bakımına, değiştirilmesine ya

da komple motorun değiştirilmesine ihtiyaç duyulur. Bu yüzden fırçalı motorların bakım maliyeti yüksektir. Fırçalı motorlarda fırçanın sürtünmesinden kaynaklı elektrik kıvılcımları ve gürültü de meydana gelir. Maliyetinin düşük olması, sürülmesi için harici devreye ihtiyaç duyulmamasından dolayı hobi amaçlı üretilen küçük ve düşük maliyetli İHA modellerinde sıklıkla fırçalı DC motorlar tercih edilir.

Fırçasız DC motorlarda komütasyon için sürücü devreleri kullanılır. Fırçanın sürtünmesinden kaynaklı kayıplar olmadığı için fırçalı dc motorlara göre daha verimlidir. Rotor mıknatısları taşırken, stator kısmı sargılardan oluşur. Rotorun sargı etrafında dönmesi ya da sargının içinde dönmesine göre "outrunner" ve "inrunner" olarak ayrılır. Fırçasız DC motorun sürücü devresinin, stator üzerindeki sargılara doğru anda ve sırada enerji verip rotoru döndürebilmesi için; rotorun pozisyonu sürücü devresi tarafından bilinmelidir. Sürücü devresi, rotorun pozisyon bilgisini, motor içinde yer alan konum sensörlerinden alır. Bu sensörler tasarıma göre rotorun sensörün yanından geçerken oluşturduğu manyetik alandan etkilenen hall effect sensör ya da rotorun konumunu ışıkla algılayan optik sensör olabilir. Sensörlerden gelen rotor pozisyon bilgisine göre, sürücü devre üzerinde yer alan anahtarlar(genellikle power mosfet) açılır-kapanır. Böylelikle doğru anda ve sırada statorlara enerji verilerek kesintisiz dönüş sağlanır. Quadrotorlarda motor sürücüsü olarak sıklıkla kullanılan ESC (Electronic Speed Control) modülünün Şekil 2.7'de bir örneği verilmiştir. Verilen örnekteki ESC 40 amper değere kadar akım verebilir. ESC üzerindeki Bec (Battery Eliminator Circuit), içerdiği regülatör devresi aracılığıyla çıkışta belirli bir gerilim verir. Örneğin ESC 24 volt ile beslenip, motorlar 24 volt ile çalışıyorken, Bec devresinin uçlarından 5 volt değer alınabilir. Bu nedenle uçuş kontrol kartı gibi modüller harici bir güç kaynağına ya da devreye ihtiyaç kalmadan Bec üzerinden beslenebilirler. Fırçasız DC motorların en yaygın olanı 3 fazlı olan modelidir. Üç fazdan ikisine enerji verilirken, boşta olan üçüncü kablodan sensörlerden elde edilen rotor pozisyon bilgisi okunur.

Şekil 2.7. Örnek bir ESC Modülü

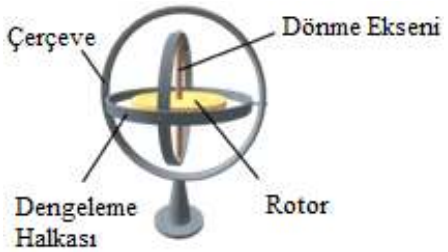
Motorların farklı gerilim değerlerinde, çekebileceği akım miktarı ve bu gerilim, akım ikilisinin farklı pervane ölçüleriyle oluşturabileceği maksimum kaldırma kuvveti üreticisi

tarafından paylaşılır. Quadcopter tasarlanırken, motorların oluşturduğu toplam kaldırma gücünün toplam ağırlıktan fazla olması gerekmektedir. Bu nedenle motor tercih edilirken kaldırma gücünün yanı sıra, motor ağırlığı, akım ve voltaj değerleri, iskelet sistemi, gerekli elektronik devreler, batarya, faydalı yükler de dâhil olmak üzere bütün bileşenlerin toplam ağırlığı göz önünde bulundurulur.

Sistem üzerinde güç çekecek bütün elektronik devrelerin ve motorların ihtiyaç duyacağı gerilim ve akım değerlerine göre batarya seçilir. Genellikle hafif olmasından, anlık yüksek güç verebilmesinden ve kapasitesinin yüksek olmasından dolayı LiPo(Lityum Polimer) piller tercih edilir. İnsansız hava araçlarının temel bileşenlerinden geliştirilmesine en çok ihtiyaç duyulan bileşendir. Döner kanatlı insansız hava araçlarının havada kalabilmesi için motorlar çok fazla güç tüketir. LiPo bataryaların zamanla performansının düşmesi, soğuk havada düşük performans göstermesi, üreticisinin belirlediği akım sınırlarının ötesine geçilmesi halinde ya da güneşte bırakılması durumunda ısınma ve patlama gibi dezavantajları vardır. Şekil 2.8'de 6200 mAh kapasiteli bir LiPo Batarya örneği verilmiştir.

Şekil 2.8. Örnek Bir LiPo Batarya

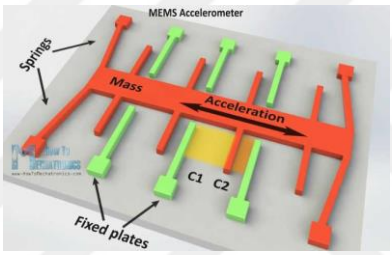
Gerek otonom uçuş gerçekleştirmesi beklenen, gerekse Yer kontrol istasyonundaki pilot tarafından uçurulan İHA, rüzgâr gibi dış etmenlerden etkileneceği için hava aracını havada dengede tutabilmeyi sağlayan sensörlere ihtiyaç duyulur. Hava aracınının x,y,z ekseninde açısal hızlarını elde etmek ve hava aracını dengede tutabilmek için ataletsel ölçme birimleri olarak adlandırılan jiroskop, ivmeölçer ve manyetometre kullanılır. Şekil 2.9'da mekanik bir jiroskopun yapısı verilmiştir.



Şekil 2.9. Mekanik Jiroskop

Jiroskop, x, y ve z eksenleri üzerindeki açısal hızları verir. Yönelim kestiriminde bulunurken, algılayıcılardan alınan ölçümlerde verinin doğruluğunu azaltacak seviyede yüksek gürültüler bulunur. Bu nedenle tek algılayıcıdan alınan verilerle yön kestirimi yapılmaz. Jiroskoplara ek olarak ivmeölçer ve manyetometreden alınan değerler ile yer çekimi ve manyetik alan da hesaplanarak yüksek bir doğruluk oranı yakalanabilir [22].

Yere paralel olduğu konumda 3 eksenli bir ivmeölçerden, z ekseninde yerçekimi ivmesinin büyüklüğünde (9.80665 m/s^2) bir çıkış vermesi beklenir. Hareketsizken x ve y eksenlerinden bir değer vermesi beklenmez. İvmeölçer herhangi bir yöne doğru eğilirse yerçekimi ivmesi bu eksenlere etki edecek ve bu eksenlerdeki ölçümlerde bir sapma gözlenecektir [25]. Analog veya dijital çıkış veren, basınca duyarlı veya kapasitif modelleri vardır. Şekil 2.10'da örnek bir kapasitif ivmeölçer gösterilmiştir.

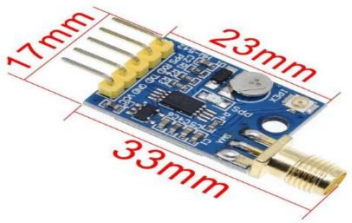


Şekil 2.10. Kapasitif İvmeölçer Örneği

Manyetometre ile dünyanın manyetik alanından faydalanarak pozisyonun enleminin bulunması sağlanır. Mıknatıslanma ile çalıştığı için etrafındaki manyetik cisimlerden etkilenip hatalı sonuçlar verebilir. Jiroskop, ivmeölçer ve manyetometre her biri kendi başına kullanıldığında belli bir hata miktarına sahiptir. Mikro düzeyde üretilen sensörlerin birleştirilmesi(Sensör Füzyonu) ile bu 3 sensörü içeren entegreler üretilmiştir. Yazılım tarafında seri iletişim ile bu 3 sensörün her birinden ayrı ayrı değer okunabilir. Elde edilen değerlerin filtrelenmesi ile mevcut konum hesaplanabilir. Şekil 2.11'de bu 3 modülü tek bir çatıda toplayan örnek bir entegre verilmiştir.

Şekil 2.11. 9 Eksen Jiroskop, İvmeölçer, Manyetometre

Bu ataletsel ölçme birimlerinin yanı sıra seyrüsefer desteği için hava araçlarında faydalı yük olarak, uydulardan faydalanarak dünya üzerinde bulunulan pozisyonun bulunmasını sağlayan, GPS sistemi de yer alır. GPS, uydulara ilettiği ve uydudan alınan sinyaller üzerinde saat bilgisi ile birlikte ışık hızının saniyede aldığı yol bilgisinden faydalanarak konum hesaplaması yapar. Hesaplamaların sağlıklı olması için en az 3, ideal olarak 4 uydudan sinyal alınması gerekir. Hava aracının ve uyduların hareket halinde olması, atmosferik etkiler, sinyallerin cisimlerden yansıması gibi etkenler GPS verisinde hata payı meydana getirir. Bu hata payının minimize edilmesi amacıyla geçmiş yıllarda birçok çalışma yapılmıştır. Bu çalışmalar ilkelerinde çalışan modern GPS sistemleri, birkaç santimetre hassasiyetle tam konum belirlenmesini sağlar. Şekil 2.12'deki örnekteki gibi entegre GPS modülleri quadcopterler tarafından taşınabilecek küçük ölçülerdedir.



Şekil 2.12. Neo-6m Arduino Shield Mini Gps Modülü

Termal kamera, mesafe ölçer, barometre, termometre, nemölçer, mikrofon, hoparlör gibi ihtiyaca göre birçok faydalı yük bataryadan çekeceği güç, ağırlık ve aerodinamik ilkelerine uygunluk kriterlerine göre hava aracına entegre edilebilir.

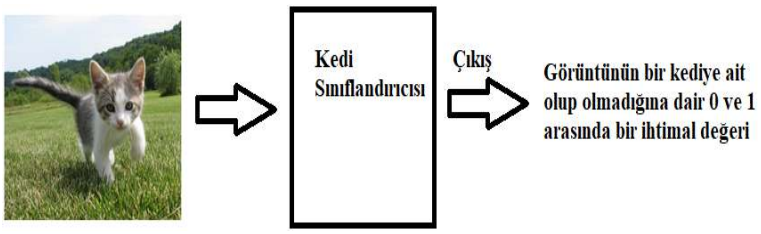
2.2. Nesne Tanıma

2.2.1. Sınıflandırma

Görüntü sınıflandırma, verilen bir görüntünün içerdiği bir nesneyi önceden belirlenmiş sınıflardan birine kategorize etme işlemidir. Görüntü sınıflandırma için geleneksel yöntemlerde, ilk aşamada ilgili sınıfın ayırt edici görüntü özelliklerinin, kenar tespiti, köşe tespiti, eşik değere göre piksel temelli bölümleme gibi görüntü işleme algoritmalarının da yardımıyla uzman kişiler tarafından çıkarılması gerekmektedir. Test aşamasında, sınıflandırılması istenen görüntüde ilgili sınıfa ait özellikler araştırılarak, görüntüde belirli bir sayıda ilgili sınıfa ait özellik tespit edilirse, görüntünün o sınıfa ait olduğu kabul edilir. Bu geleneksel yöntemlerde görüntünün hangi

özelliklerinin önemli olduğuna karar verilmesi büyük bir zorluk teşkil eder. Sınıf sayısı arttıkça bu zorluk daha da artar [26].

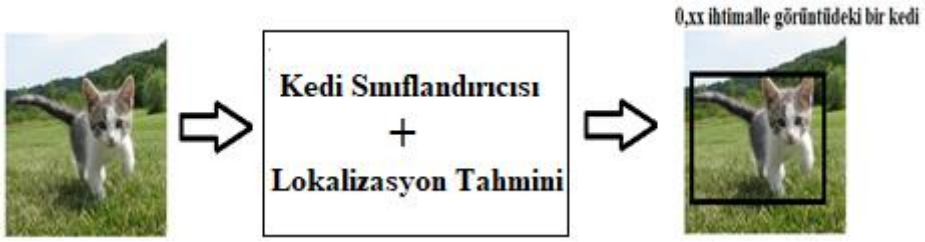
Şekil 2.13'te verilen örnekte gösterildiği üzere sınıflandırıcı çıkış olarak, görüntünün bir sınıfa ait olduğuna dair 0 ve 1 arasında bir olasılık değeri üretir. Sınıflandırmaya dair bu olasılık değeri ve eşik değeri kullanılarak; evet/hayır, var/yok gibi bir sonuç elde edilebilir.



Şekil 2.13. Sınıflandırma Örneği

2.2.2. Lokalizasyon

Görüntü sınıflandırma işleminde sonuç olarak yalnızca görüntünün hangi sınıfa ait olduğuna dair bir tahmin üretilir. Tahmin edilen sınıfa ait nesnenin görüntünün içindeki konumuna dair bir sonuç üretilmez. Şekil 2.14'te verilen örnekteki gibi, lokalizasyon işleminde nesnenin görüntü içindeki koordinatlarını belirten sınırlayıcı kutu değerleri(x, y, genişlik, yükseklik) tahmin edilir.



Şekil 2.14. Sınıflandırma ve Lokalizasyon

2.2.3. Nesne Tespiti

Sınıflandırma işlemi bütün görüntü üzerinde gerçekleştirildiğinden, bir görüntüde birden fazla nesne ve sınıf olduğu zaman ilgili görüntünün tek bir çıkış etiketiyle sınıflandırılması doğru olmaz. Nesne tespiti, görüntüde yer alan nesnelerin her birinin sınıfının ve görüntü içindeki konumunun ayrı ayrı tespit edilmesidir. Şekil 2.15'deki örnekte gösterildiği üzere, görüntüde yer

alan kedinin ve köpeğin, herbirinin sınıf olasılık değeri ile birlikte sınırlayıcı kutuları ayrı ayrı tespit edilmiştir.

Şekil 2.15. Nesne Tespit Örneği

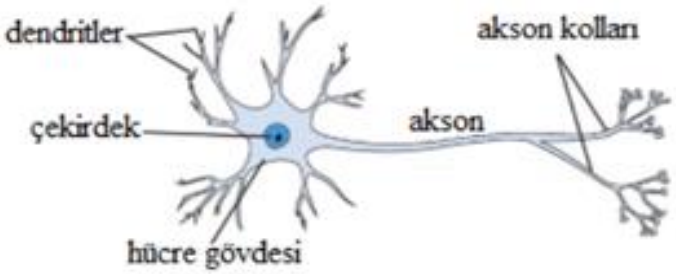
2.2.4. Segmentasyon

Nesne tespit işlemlerinde görüntüde yer alan nesnelerin sınırlayıcı kutuları bulunur. Nesnelerin görüntü içindeki gerçek sınırları genellikle dikdörtgenlerle ifade edilemeyecek kadar girintili çıkıntılıdır. Nesnelerin görüntü içerisinde bulunduğu yerlerin piksel temelli belirtilmesine segmentasyon denir. Şekil 2.16'da segmentasyon örneği verilmiştir.

Şekil 2.16. Segmentasyon Örneği

2.3. Yapay Sinir Ağları

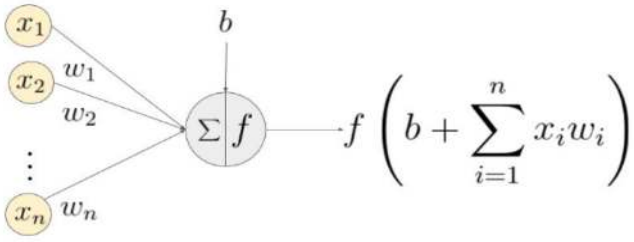
Yapay sinir ağları biyolojik sinir sisteminin modellenmesi hedefinden esinlenmiştir. İlk olarak nörofizyolog McCulloch ve matematikçi Pitts 1943 yılında beyindeki nöronların nasıl çalıştığı ile ilgili basit bir sinir ağı oluşturarak siniri matematiksel olarak modellemişlerdir [27]. Şekil 2.17'de biyolojik sinir hücresinin yapısı gösterilmiştir.



Şekil 2.17. Sinir Hücresi

Nöronlar dentrit, gövde, akson ve çekirdekten oluşur. Nöronlar dendritleri aracılığıyla aldıkları sinyalleri aksona iletir. Akson kolları bir sonraki nöronun dentritine, sinapsları aracılığıyla bağlanır.

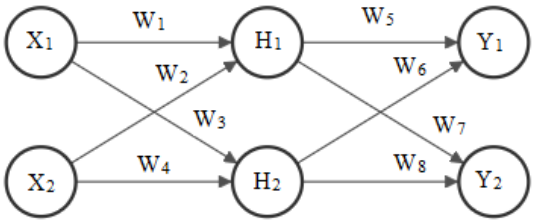
Şekil 2.18'de Toplayıcı Fonksiyon yapısı verilmiştir. Nöron girişlerinden aldığı sinyallerin her birini o sinyale ait ağırlıklarla çarpıp toplamını alır. Daha sonra bu toplam değere Bayes değeri eklenerek nöronun değeri bulunur. Bir nöronun çıkışının aktif olup olmayacağını aktivasyon fonksiyonu belirler. Aktivasyon fonksiyonu, değer üzerinde non-lineer bir dönüşüm gerçekleştirir.



Şekil 2.18. Toplayıcı Fonksiyon [28]

2.3.1. Tam Bağlı Katmanlar

Bir katmandaki bütün nöronlar kendinden sonraki katmandaki bütün nöronlara bağlıysa bu katman "Tam Bağlı Katman" adını alır. Evrişimsel Sinir Ağlarının son katmanlarında bir ya da iki tane tam bağlı katman yer alır. Tam bağlı katmanların görevi evrişimlerden çıkarılan görüntü özelliklerine dayanarak sınıflandırma işlemi yapmaktır. Şekil 2.19'daki gizli katmandaki bütün nöronlar kendinden sonraki katmandaki bütün nöronlara bağlı olduğu için, bu katman bir Tam Bağlı Katman örneğidir.



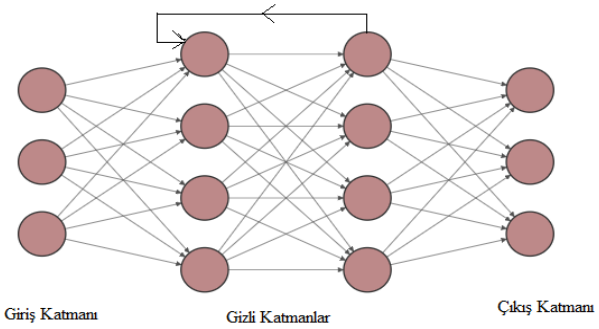
Şekil 2.19. Tam Bağlı Katman Örneği

2.3.2. İleri Beslemeli Yapay Sinir Ağları

İleri beslemeli yapay sinir ağları yalnız tek yönde veri iletimi yapılmasına olanak sağlar. Girişlerden verilen veriler, bir katmandaki nöronlardan bir sonraki katmandaki nöronlara doğru ilerleyerek çıkışa ulaşır. Şekil 2.19'daki tam bağlı katmanlardan oluşmuş sinir ağı aynı zamanda ileri beslemeli yapay sinir ağının bir örneğidir.

2.3.3. Geri Beslemeli Yapay Sinir Ağları

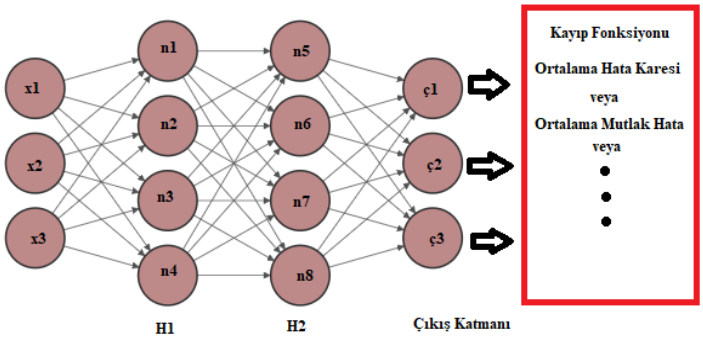
Geri beslemeli yapay sinir ağlarında, veri iletimi 2 yönlü gerçekleştirilebilir. Şekil 2.20'deki gibi bir katmandaki nöron kendinden önceki katmanlardaki bir nöronun girişi olabilir.



Şekil 2.20. Geri Beslemeli Yapay Sinir Ağı

2.3.4. Kayıp Fonksiyonları

Kayıp Fonksiyonları, sinir ağının çıkışından elde edilen değerlerin, bu ağın çıkışından beklenen değerlerden kaybını/uzaklığını/farkını hesaplamak için kullanılır. Şekil 2.21'deki örnek sinir ağının çıkışları Kayıp Fonksiyonu'nun parameterleridir.



Şekil 2.21. Kayıp Fonksiyonu

Eğitimli derin sinir ağı modelinde, eğitim aşamasında modele veri ve veriye dair etiketler sağlanır. Görüntü işlemede bu veriler, görüntüler ve bu görüntülerin hangi sınıfa ait(insan, kedi, köpek...) olduğuna dair bir etikettir. Modele test aşamasında, bir görüntü verildiği zaman model sonuç olarak, giriş katmanında verilen görüntüye karşılık çıkış katmanında tespit edilmesi istenen sınıflara dair olasılıksal değerler üretir. Bu değerlerin başarılı bir şekilde oluşturulması için sinir ağı modelinin iyi eğitilmesi gerekir. Eğitim aşamasında, sinir ağına hâlihazırda verilen görüntünün hangi sınıfa ait olduğu etiketlenerek verildiği için, model ulaşılan sonuçlar ile hedeflenen sonuçlar arasındaki hata payını hesaplar. Hata payını düşürebilmek için sinir ağı güncellenmiş ağırlıklar ile kayıp fonksiyonu belirlenen bir hata payına düşene kadar tekrar tekrar test edilir. Hata payının hesaplanması için kullanılan çeşitli fonksiyonlar vardır.

Ortalama Hata Karesi(mean squared error) yönteminde, beklenen gerçek değerlerle eğitim aşamasında sinir ağının mevcut ağırlıklarla verdiği çıkış arasındaki farkların karesinin ortalaması alınır. Denklem (2.1)'de formülü verilmiştir.

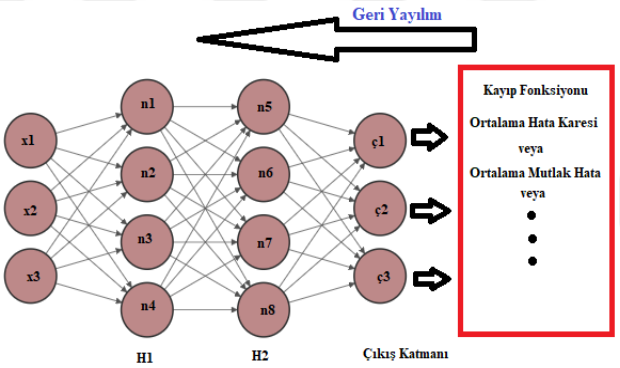
$$ORTALAMA\ HATA\ KARESİ = \frac{\sum_{i=1}^{n}(y_i - \acute{y}_i)^2}{n} \tag{2.1}$$

Ortalama Mutlak Hata(Mean Absolute Error/L1 Loss) yönteminde ise, beklenen gerçek değerlerle, tahmin arasındaki farkların mutlak değerlerinin toplamının ortalaması hata oranı olarak değerlendirilir. Denklem (2.2)'de formülü verilmiştir.

$$ORTALAMA\ MUTLAK\ HATA = \frac{\sum_{i=1}^{n}|y_i - \acute{y}_i|}{n} \tag{2.2}$$

2.3.5. Geri Yayılım

İleri beslemeli derin sinir ağlarında, ilerleme, girişten başlayarak bir katmandaki nöronlardan bir sonraki katmandaki nöronlara doğru aktığı için, kayıp fonksiyonunun çıktısı son katmanda elde edilen parametreler ile hesaplanır. Eğitim zamanında, sinir ağındaki nöronların ağırlıklarının başarılı bir şekilde eğitilmesi istenir. Kayıp fonksiyonunun hesapladığı kayıp miktarı bu ağırlıkların aranılan ağırlıklara ne kadar yakın olduğunu gösterir. Kayıp miktarı azaldıkça ağırlıklar aranılan ağırlıklara o denli yakınsamış demektir. Kayıp fonksiyonu yalnız çıkıştaki değerlere göre hesaplanır. Çıkıştaki değerlerin değiştirilmesi için bir önceki katmandaki ağırlıkların değiştirilmesi gerekir. Bir katmandan önceki katmana doğru, ağırlıkların hesaplanıp, güncellenerek geriye doğru akmasına "Geri Yayılım" denir. Şekil 2.22'de geri yayılımın bir örneği verilmiştir.



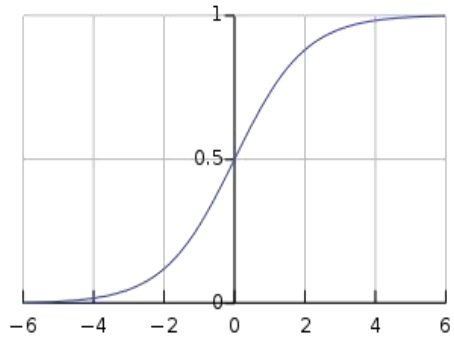
Şekil 2.22. Geri Yayılım

Her bir nöron girişi, bir önceki katmandaki nöronların çıkış değerlerine bağlı olduğu için, kayıp fonksiyonunun geriye doğru ilgili katmandaki nörona göre kısmi türevi alınarak, geriye doğru ağırlık güncellemesi yapılır. Geri Yayılım ile hesaplanan yeni değerlere göre sinir ağı çıkış üretir. Kabul edilen kayıp miktarına erişilinceye kadar işlem tekrarlanır.

2.3.6. Aktivasyon Fonksiyonları

Sinir ağının, nöronları arasındaki lineer yapıyı kırmak nonlineer bir dönüşüm gerçekleştirebilmek [29], böylece verilerin genelleştirilmesini sağlamak amacıyla kullanılır. Nonlineer dönüşüm gerçekleştirilmesi, sinir ağının eğitim seti dışındaki veriler ile karşılaştığında doğru sonuçlar verebilmesi için önemlidir. Aktivasyon fonksiyonları transfer fonksiyonları olarak da adlandırılır.

2.3.6.1. Sigmoid

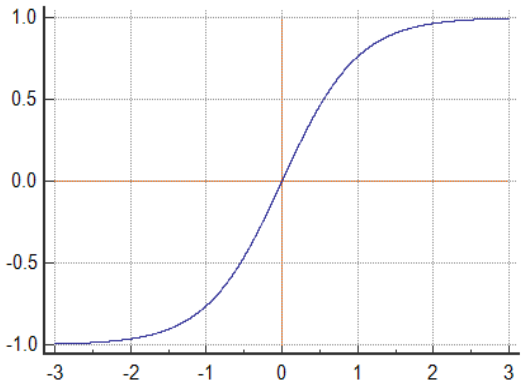


Şekil 2.23. Sigmoid Grafik Gösterimi

$$S(x) = \frac{e^x}{1 + e^x} \tag{2.3}$$

Denklem (2.3)'te formülü verilen Sigmoid fonksiyonun Şekil 2.23'te grafik gösterimi verilmiştir. Sigmoid aktivasyon fonksiyonu gerçek değerli bir girdi alır ve [0,1] aralığında bir değer üretir. Bu nedenle büyük negatif sayılar 0, büyük pozitif sayılar 1 değerini alır. 0 değerini alması nöronun aktif olmaması, 1 değeri ise aktif olduğu anlamına geldiğinden geçmişte yapay sinir ağlarında yaygın olarak kullanılan bir fonksiyondur. Ancak günümüzde geri yayılım sırasında kaybolan gradyanlara sebebiyet vermesi ve başlangıç ağırlıkları çok büyük olduğunda ağın öğrenememesi gibi sorunlar nedeniyle nadir kullanılmaktadır [27].

2.3.6.2. Tanh

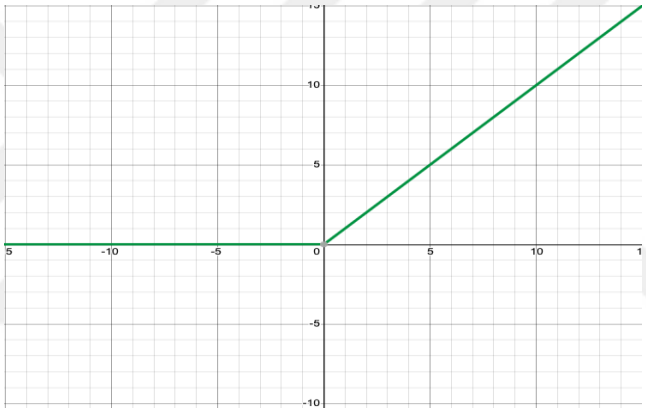


Şekil 2.24. Tanh Grafik Gösterimi

Şekil 2.24'te grafiği verilen Tanh aktivasyon fonksiyonu, gerçek değerli girdiyi alıp [-1,1] aralığında bir değer üretmektedir. Pratikte sigmoide tercih edilmektedir [27]. Tanh fonksiyonu negatif değer için negatif sonuç döndürürken, pozitif değer için pozitif değer döndürür.

2.3.6.3. ReLU

ReLU (Rectified Linear Unit) günümüzün en popüler aktivasyon fonksiyonlarından biridir. Bu fonksiyon gelen değerlere pozitif mi negatif mi diye bakar. Eğer gelen değer sıfırdan küçükse işlem sonucu olarak sıfır verirken değer sıfırdan büyükse bu değeri sonuç olarak verir. Biyolojik nöronlar belli bir seviyenin altında gelen sinyalleri görmezden geldiği için bu fonksiyonun sinir sistemimizde bulunan biyolojik nöronlara yakın bir yapısı vardır.



Şekil 2.25. ReLU

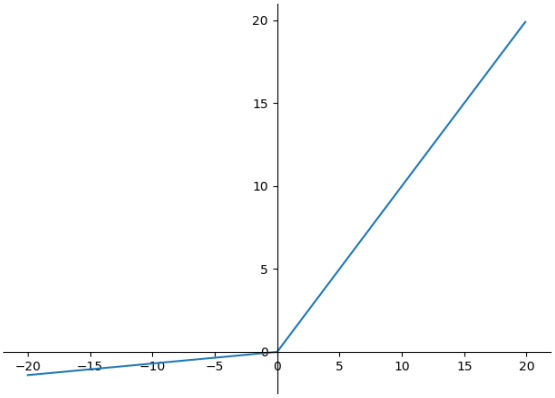
$$F(x) = \max(0, x) \tag{2.4}$$

Denklem (2.4)'te formülü verilen ReLU aktivasyon fonksiyonunun, Şekil 2.25'de grafik çizimi verilmiştir. Sigmoid ve tanh aktivasyonlarına göre daha hızlı yakınsama sağlamakta ve daha az maliyetli işlemler içerir. Bu fonksiyon ilk olarak 2012 yılında ImageNet Geniş Ölçekli Görsel Tanımlama Yarışması'nda (ILSVRC) en iyi sonuçları veren AlexNet'te kullanılmıştır. Bu tarihten itibaren fonksiyonun popülerliği de giderek artmıştır [13].

2.3.6.4. Leaky ReLU

Leaky ReLU (LReLU) fonksiyonu, ReLU aktivasyon fonksiyonundan negatif noktada ayrışır. ReLU ile X≤0 olduğunda fonksiyon sonucu 0 olur. Şekil 2.26'da grafik çizimi verilen

LReLU aktivasyon fonksiyonunda X≤0 olduğunda denklem (2.5)'te verildiği üzere a katsayısı küçük tutularak Y değerinin negatif eksende küçük bir değer alması sağlanır.

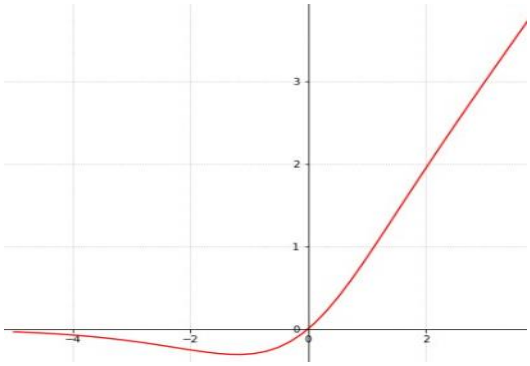


Şekil 2.26. Leaky ReLU

$$f(x) = \begin{cases} x & x > 0, \\ 0.ax & x \leq 0 \end{cases} \tag{2.5}$$

Denklem (2.5)'te LReLU aktivasyon fonksiyonunun matematiksel bağıntısı verilmiştir.

2.3.6.5. Mish



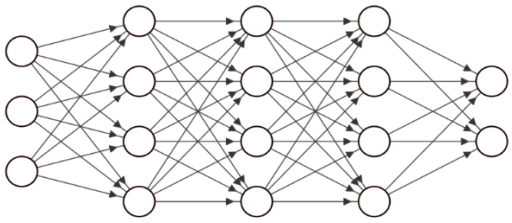
Şekil 2.27. Mish

$$f(x) = xtanh\big(softplus(x)\big) = xtanh(ln(1 + e^x)) \tag{2.6}$$

Denklem (2.6)'da formülü verilen Mish [30] aktivasyon fonksiyonunun, Şekil 2.27'de grafik gösterimi verilmiştir. Diganta Misra, tarafından 2019 yılında ortaya atılan bu aktivasyon fonksiyonu, derin öğrenme modellerindeki yapılan karşılaştırmalı testlerdeki yüksek başarısından dolayı YOLO v4 modelinde de kullanılmıştır.

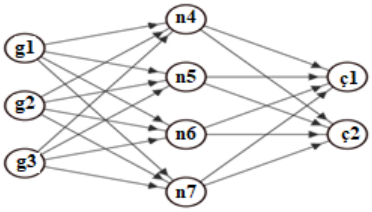
2.4. Evrişimsel Sinir Ağları

Evrişim işlemi özel bir lineer işlem türüdür. Evrişimsel Sinir Ağları (ESA), en az bir katmanında genel matrix çarpımı yerine evrişim işlemi gerçekleştiren sinir ağlarıdır [31]. ESA temelde evrişim, pooling ve aktivasyon katmanlarından oluşur [29]. Bu sinir ağları değişik modellerde tasarlanabilir. Kurulan modele göre değişik sayıda gizli katman ve her katmanda değişik sayıda nöron içerebilir. ESA'da Şekil 2.28'deki örnekteki gibi, nöronlar arasında bağlantılar katmanlar arasında gerçekleştirilir, aynı katmandaki nöronlar arasında bağlantı gerçekleştirilmez.



Şekil 2.28. Bir katmandan diğer katmana bağlanan nöronlar

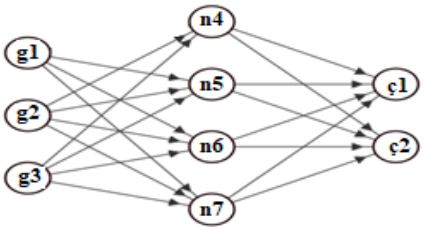
Sinir ağının katman sayısından bahsedilirken giriş katmanı sayılmaz. Bu nedenle gizli katmanları olmayan bir sinir ağı girişlerin direk çıkışlara bağlandığı tek katmanlı bir ağdır. Sinir ağı üzerinde yer alan her bir nörona ait ağırlıklar ve bu ağırlıkların toplamına eklenen bias değerleri hafızada tutulur. Böylece sinir ağı çalıştırıldığında; yani girişlerinden vektör şeklinde bir dizi veri verilip çıkıştan bir sonuç elde edilmek istendiğinde; ağdaki nöronlara ait ağırlıklar ve bias değerleri kullanılarak girişten çıkışa kadar bir katmandan bir sonrakine, herbir nöron değeri hesaplanarak çıkış değerleri üretilir.



Şekil 2.29. Örnek Bir Sinir Ağı

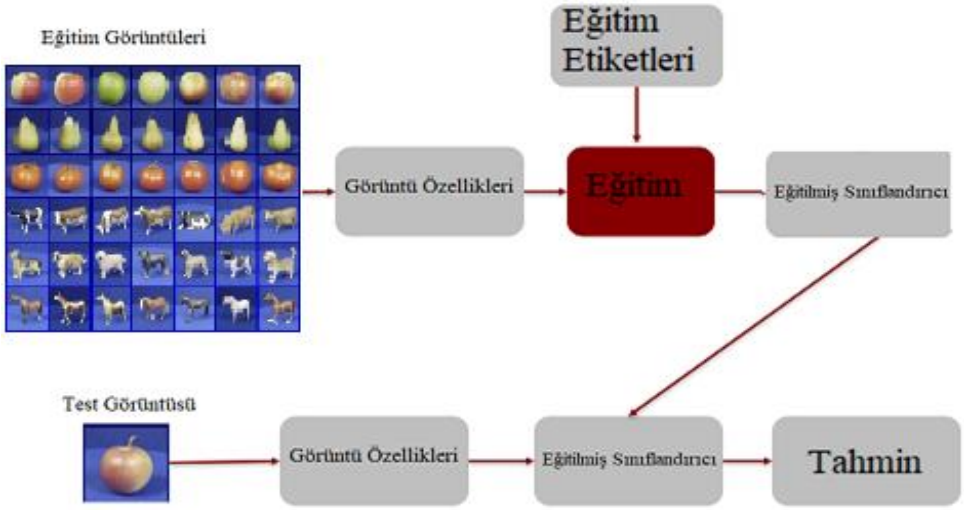
Giriş katmanı sayılmadığı için Şekil 2.29'daki örnek resimde 4 adet nöron gizli katmanda, 2 adet nöron çıkış katmanında olmak üzere toplam 6 adet nöron bulunur. Nöronların toplam bağlantı sayısı gizli katman tam bağlı katman olduğu için, (3x4)+(4x2)=20 adettir. Toplam bias sayısı; 4+2=6 adet bulunur. Bu durumda örnekteki sinir ağının toplam parametre sayısı 26 olur.

Şekil 2.29'daki örnek sinir ağının g1 girişi ile n4 nöronu arasındaki bağı koparırsak Şekil 2.30'daki sinir ağı oluşur. Nöron sayısı aynı kalmasına rağmen parametre sayısı azalır. Her parametre, hafızada bir alan işgal etmesinin yanı sıra donanım üzerinde de bir işlem maliyeti oluşturur. Bu nedenle sinir ağının ölçüsünden bahsedilirken, nöron sayısından ziyade parametre sayısından bahsedilir.



Şekil 2.30. Parametre Azaltılması

2012 yılında Alexey ve arkadaşlarının Evrişimsel Sinir Ağları (ESA)'nın sınıflandırma işlemlerindeki başarısını göstermesinin ardından görüntü sınıflandırma işlemleri üzerine olan çalışmalar, ESA üzerinde yoğunlaşmıştır. Eğitim işlemi için, eğitim setinde yer alan her bir görüntü tek tek etiketlenerek hangi sınıfa ait olduğu belirtilir. Bu görüntüler üzerinde eğitilen ESA, daha sonra test anında verilen görüntüye karşılık bu görüntünün belirlenmiş sınıflara ait olma olasılıklarına dair tahmin üretir. Şekil 2.31'de eğitim ve sınıflandırma aşamalarının genel bir gösterimi verilmiştir.



Şekil 2.31. ESA Eğitimi ve Sınıflandırma Amaçlı Kullanımı

Görüntüye ait piksel değerleriyle beslenen ESA görüntüye dair çeşitli özellikleri ortaya çıkarır. Katmanlardaki her bir nöron, bir grup görüntü piksellerini giriş olarak alır. Bu görüntü

piksellerini ağırlıklarıyla çarpar, toplar daha sonra aktivasyon fonksiyonuna iletir. Sinir ağındaki diğer katmanlardan farklı olarak çıkış katmanı, sınıflandırma sonuçlarını içerdiği için genelde aktivasyon fonksiyonu içermez.

Nesne tanıma amacıyla kullanılan ESA'dan beklenen, kendisine giriş olarak verilen görüntü içerisindeki nesneleri yüksek bir doğruluk oranıyla sınıflandırması ve bu nesnelerin görüntü içerisinde bulunduğu konumu tespit etmesidir.

2.4.1. Evrişim İşlemi

Evrişim işlemi için, görüntüyle kanal sayısı aynı filtreler kullanılır. 3 renk (rgb) kanalına sahip bir görüntü için, her bir kanal bir rengi ifade edecek şekilde kullanılan kanal sayısı 3, gri tonlarında bir görüntü için filtre kanal sayısı 1'dir. Filtrelerin her biri görüntü üzerindeki farklı bir özelliğin aranması için kullanılır. Filtreler, görüntü üzerinde sol üst başlangıç noktasından başlayarak, atlama miktarı(stride) kadar kaydırılarak görüntüye uygulanır. Filtrenin boyutu uygulanacak görüntü boyutundan küçük olmalıdır. Sonuçta orijinal görüntü üzerine uygulanan filtre sayısı kadar yeni görüntü elde edilmiş olur. Bulunan bu yeni görüntülere özellik haritası denir.

0	1	0
0	1	0
0	1	0

Şekil 2.32. Dikey Filtre

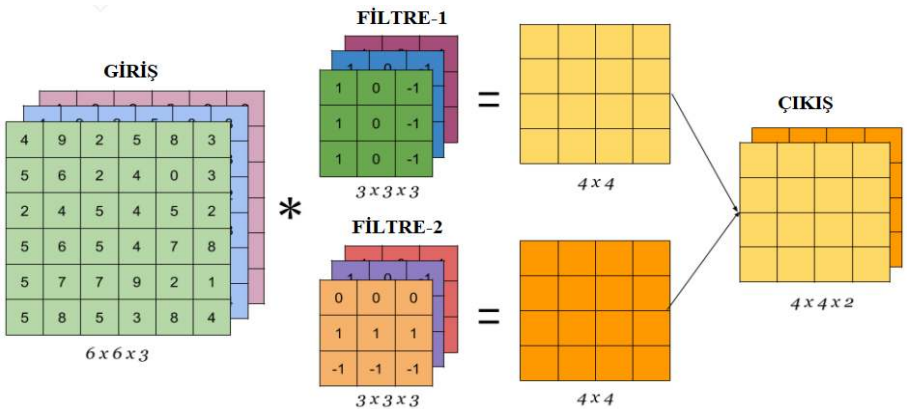
0	0	0
1	1	1
0	0	0

Şekil 2.33. Yatay Filtre

Şekil 2.32'deki filtrenin görüntü üzerine uygulanması, sonuçta görüntü üzerinde yalnızca dikey çizgilerin olduğu bir özellik haritası oluştururken, Şekil 2.33'deki filtre aynı görüntü üzerine uygulandığında yalnız yatay çizgilerin olduğu bir özellik haritası oluşur. Bu 2 ayrı filtre sonucu oluşan özellik haritaları birleştirilirse, görüntü üzerinde sadece yatay ve dikey çizgilerin görüldüğü

yeni bir özellik haritası oluşmuş olur. Görüntü üzerine birçok birbirinden farklı filtre uygulanarak, görüntünün farklı özelliklerini öne çıkaran özellik haritaları oluşturulabilir. Sinir ağı, filtrelerin uygulanması sonucu oluşan özellik haritaları üzerine her katmanda yeni filtreler uygulayarak görüntünün farklı özelliklerini ortaya çıkarmaya çalışır. Sinir ağı, ilk katmanlarda daha ziyade basit niteliklerin özellik haritalarını çıkarırken, katmanlar ilerledikçe orta ve nihayet üst düzey özelliklerin haritalarını oluşturur. Genel olarak küçük boyutlu filtreler nesnelerin ayrıntılı özelliklerinin çıkarılması ve küçük boyutlu nesnelerin tespit edilebilmesi için önemlidir.

Bir evrişim katmanından sonra aynı ölçülerde ve filtre sayısınca özellik haritası oluştuğundan, bir sonraki evrişim katmanında önceki katmanın oluşturduğu özellik haritası sayısınca kanal olur. Bu kanallarda her bir filtre için hesaplanan değer tek bir özellik haritası oluşturur. Sonuçta filtre sayısınca özellik haritası oluşturulmuş olur. Örnek olarak Şekil 2.34'te 3 kanallı bir görüntünün 2 adet (3x3) ölçülerinde 3 kanallı filtre ile evrişim işlemi sonucu gösterilmiştir. Filtrenin her bir kanalına "kernel" adı verilir. Her bir kernel birbirinden farklıdır ve her bir kernel girişten verilen görüntünün ilgili kanalı üzerinde işlem yapar. Şekil 2.34'te 1.filtrenin 1. kanalı giriş görüntüsünün 1. kanalı üzerinde kaydırılarak yeni bir görüntü elde edilir. Aynı şekilde 1. filtrenin 2. kanalı giriş görüntüsünün 2. ve 1. filtrenin 3. kanalı giriş görüntüsünün 3. kanalı üzerinde işlem yaparak yeni görüntüler oluşturulur. Bu oluşturulan görüntüler toplanarak tek kanallı bir çıkış görüntüsü elde edilir. Aynı şekilde 2. Filtre de görüntü üzerine uygulanarak farklı bir çıkış görüntüsü elde edilir. Bu durumda herbir filtre bir tek özellik haritası oluşturduğu için sonuçta filtre sayısınca yeni özellik haritaları oluşturulmuş olur. Kullanılan filtre sayısı 2 olduğu için sonuç olarak 2 adet görüntü oluşmuştur. Bu 2 adet görüntüye, 2 kanallı tek bir görüntü de denebilir. Bu çıkış görüntüsü üzerine evrişim işlemi uygulanacak olursa kullanılacak filtrelerin kanal sayısı ya da diğer bir deyişle derinliği 2 olmalıdır.



Şekil 2.34. 3 Kanallı Bir Görüntünün 2 Adet 3 Kanallı Filtre ile Evrişimi

ESA'da her evrişim katmanında kullanılacak filtre sayısı ve ölçüleri önceden belirlenir. Eğitim zamanında ilk başta filtrelerin içerdiği değerler rastgeledir. Sinir ağı eğitim zamanında, eğitim seti olarak verilen hâlihazırda sınıfları etiketlenmiş görüntüleri(insan, kedi, köpek…) işleyerek, her iterasyonda aranılan filtre değerlerine yaklaşır. Eğitim setinde bulunan her bir sınıfı diğer sınıflardan ayıran filtre değerleri o eğitim seti için aranılan değerlerdir.

2.4.2. Padding

NxN bir görüntünün FxF boyutlu bir filtre ile evrişim işleminden sonra sonuç olarak çıkan görüntünün en ve boy değerleri $\frac{(N-F)}{Atlama\ Miktarı} + 1$ olur.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Şekil 2.35. 4x4 Görüntü pixelleri

Örneğin Şekil 2.35'deki gibi (4x4) bir görüntünün, Şekil 2.36'daki gibi (2x2) filtre ile 1 atlamalı evrişiminde sol üst köşeden başlayarak filtre uygulandığında, filtre soldan sağa 3 defa, yukarıdan aşağıya 3 defa uygulanabilir. Bunun sonucunda (4x4) bir görüntüden, Şekil 2.37'deki gibi (3x3) bir görüntü meydana gelir.

1	1
1	1

Şekil 2.36. 1x1 Evrişim Filtresi

14	18	22
30	34	38
46	50	54

Şekil 2.37. Sonuç Görüntüsü

Şekil 2.37'deki gibi görüntünün her evrişiminden sonra küçülmesi sinir ağı derinleştikçe bir noktadan sonra görüntünün kaybolması anlamına gelir ve ağın derinleştirilmesine engel teşkil eder. Bu nedenle evrişim işleminden önce, görüntünün sağına/soluna/üstüne/altına değer eklenmesiyle görüntü büyütülür ve evrişim işleminden sonra görüntü boyutları korunmuş olur. Bu işleme padding denir. Eklenen değer için genelde 0 veya komşuluğundaki değer kullanılır. Şekil 2.35'deki (4x4) görüntüye padding uygulanarak, üstüne ve sağına 0 eklenmiş ve Şekil 2.38'deki (5x5) görüntü elde edilmiştir. Bu görüntüye (2x2) filtre 1 atlamalı uygulanırsa, sonuçta (4x4) bir görüntü elde edilmiş olur.

0	0	0	0	0
1	2	3	4	0
5	6	7	8	0
9	10	11	12	0
13	14	15	16	0

Şekil 2.38. Padding Örneği

Padding işleminin bir diğer önemi ise, evrişim yapılırken kenarlardaki ve köşelerdeki görüntü pixellerin orta kısımdaki pixeller kadar işleme girmesini sağlamaktır. Böylece görüntünün kenar uç ve köşelerin pixellerinin orta kısımdakilere nispeten değersiz olması ve kenar, köşelerdeki bilgilerin kaybolması engellenmiş olur.

2.4.3. Aktivasyon İşlemi

Evrişim işleminden sonra, çıkıştan önce nöronlar üzerinde aktivasyon işlemi gerçekleştirilir. Her nöron giriş katmanının bazı nöronlarından bağlantı alır. Aldığı bu bağlantılardan kendi ağırlığını hesaplar. Kendi ağırlık değerini, aktivasyon fonksiyonlarından birini kullanarak, aktivasyon işlemi ile çıkışa iletir. Aktivasyon işlemiyle nöronun çıkış değerinin ne olacağı belirlenir. Aktivasyon işlemi, nöron çıkış değerlerinin [0,1] arasında ya da [-1,1] arasında olmasını sağlar.

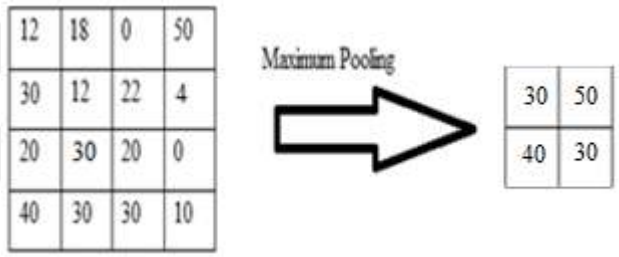
2.4.4. Pooling İşlemi

Sinir ağı, evrişim işlemiyle her katmanda filtre sayısı kadar yeni özellik haritası oluşturduğu için sinir ağının parametre sayısı artar. Bu parametre sayısını azaltmak için özellik

haritalarının küçültülmesi gerekir. Özellik haritalarının küçültülmesi işlemine "pooling" denir. Pooling için belirlenen atlama miktarı ve matris ölçüsüne bağlı olarak özellik haritası küçültülmüş olur. Pooling işleminde, özellik haritasının kenarlarındaki değerleri de işleme alabilmek için, gerekli olduğu takdirde padding uygulanabilir.

2.4.4.1. Maximum Pooling

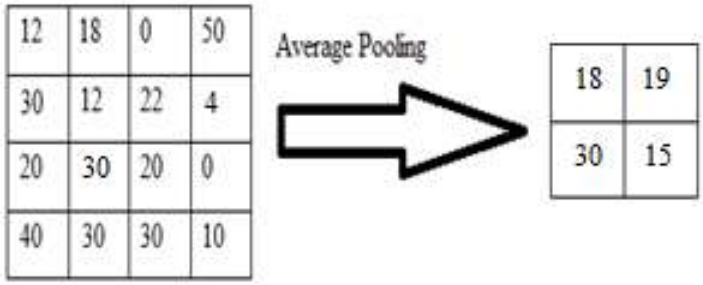
Maximum pooling işleminde, belirlenen matris boyutu temel alınarak, sol üst köşeden başlayarak ve stride kadar ilerleyerek, özellik haritası içindeki en büyük değer alınır. Şekil 2.39'da örnek özellik haritası üzerinde (2x2), stride 2 maximum pooling işlemi sonrasında elde edilen yeni özellik haritası verilmiştir.



Şekil 2.39. Maximum Pooling

2.4.4.2. Average Pooling

Average pooling işleminde, belirlenen matris boyutu temel alınarak, sol üst köşeden başlayarak ve stride kadar ilerleyerek, özellik haritası içindeki değerlerin aritmetik ortalaması alınır. Şekil 2.40'da örnek özellik haritası üzerinde (2x2), stride 2 average pooling işleminden sonra oluşan özellik haritası gözükmektedir.

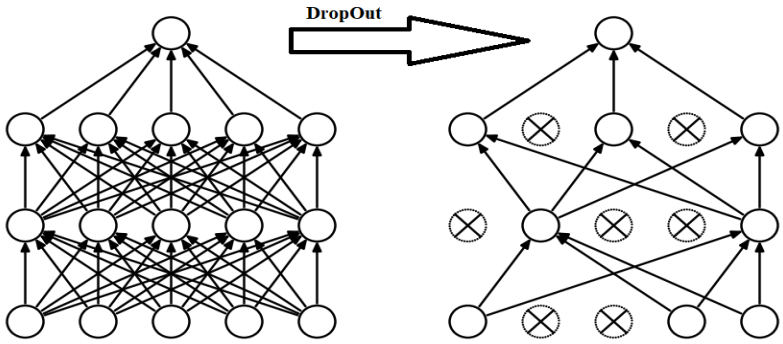


Şekil 2.40. Average Pooling

2.4.5. DropOut

Birçok parametreye sahip derin sinir ağlarında "aşırı uyma"(overfitting) istenmeyen bir durumdur. Aşırı uyma durumunda sinir ağı öğrenme yerine ezberlemiş olur. Bu da sinir ağının, eğitildiği veri seti dışında karşılaşılan örneklerde başarısız sonuçlar üretmesine neden olur.

Şekil 2.41'de bir örneği verilen DropOut [32] işlemi, aşırı uyma problemini çözmek için geliştirilmiş bir yöntemdir. Bu yöntemin temel fikri eğitim sırasında rastgele olarak bazı nöronları geçici olarak düşürmektir. Bir nöronun değeri veya ağırlığı, 0'a çekilerek nöron düşürülebilir.



Şekil 2.41. Sinir ağına DropOut Uygulanması Örneği [32]

2.4.6. DropBlock

DropOut genellikle tam bağlı katmanlarda kullanılırken, evrişimsel katmanlarda kullanımı genellikle daha az etkilidir. Rastgele düşürülen nöronlara rağmen, bilginin uzaysal bağlantılar sayesinde bir sonraki katmana aktarıldığı düşünülmektedir. DropBlock [33], DropOut yönteminin özel bir halidir. DropOut yönteminde nöronlar rastgele seçilirken, DropBlock yönteminde, bir özellik haritasındaki bitişik nöronlar birlikte düşürülür. Böylece nöronlar arasındaki uzaysal bağlantılar koparılarak sinir ağının aşırı uyumunun önüne geçilmek istenir.

2.4.7. CutMix

DropOut ve DropBlock yöntemlerinde nöronların düşürülmesi, modelin veriyi genelleştirebilmesini sağlar. Bu sayede model veri setinde yer almayan bir veri ile test edildiğinde doğruluk oranının artmasını sağlar. Fakat bu yöntemlerde eğitim setinde yer alan görüntünün belli bir miktar bilgi kaybetmesi söz konusudur. ESA modelleri, yüksek veri ihtiyacı olan veri arttıkça

başarımı iyileşen modellerdir. Bu nedenle veri içerisinde bilginin kaybolması olumsuz bir durum teşkil eder. CutMix [34] yönteminde, bu bilgi kaybını amorti etmek için görüntünün belli bölgeleri atılırken, bu bölgeler veri setinde yer alan başka bir görüntüden alınan yama ile örtülür. Şekil 2.42'de CutMix yöntemi uygulanmış örnek görüntüler yer almaktadır.

Şekil 2.42. CutMix Örnekleri [35]

2.4.8. Mozaik Veri Arttırımı

Mozaik veri arttırma yönteminde eğitim setindeki 4 adet görüntü birleştirilip tek görüntü haline getirilir. Bu da nesnelerin kendi normal çevreleri dışında da tespit edilme ihtimalini güçlendirir [36]. Şekil 2.43'te Mozaik Veri Arttırımı yöntemiyle oluşturulmuş 6 adet görüntü verilmiştir.

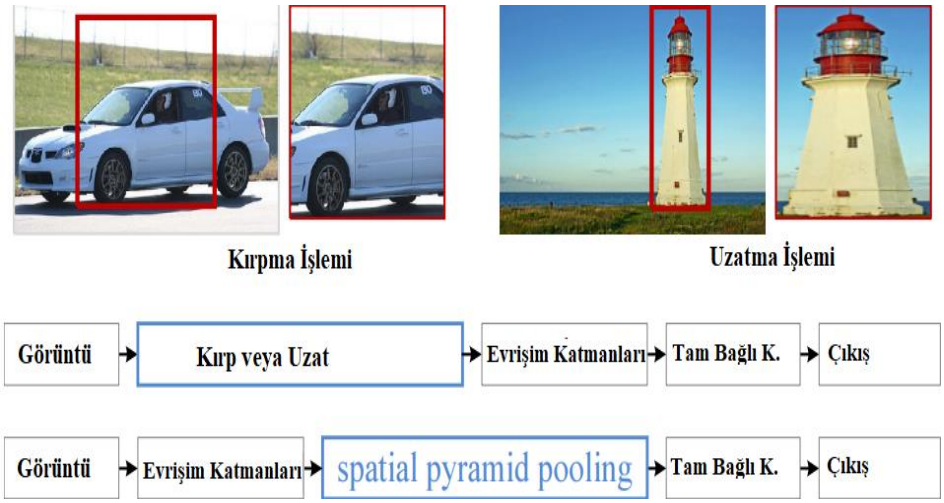
Şekil 2.43. Mozaik Veri Arttırımı [36]

2.4.9. Sınıf Etiketi Düzleme

Modelin yaptığı tahmin üzerinde 100% güven skoru oluşması, modelin veriyi öğrenmek yerine ezberlediği manasına gelebilir. Etiket düzleme yönteminde, tahminin hedefi isabet değerinin üst sınırı 1 yerine 0,9 gibi daha düşük bir değerle ifade edilir. Kayıp fonksiyonunda maksimum isabet değeri olarak 1 yerine, belirlenmiş 1'den küçük üst sınır değeri kullanılır. Bu yöntemle aşırı uyum azaltılmış olur [36].

2.4.10. Spatial Pyramid Pooling

ESA modelleri, sabit boyutlu görüntüleri giriş olarak kabul eder. Bu nedenle değişik ölçeklerdeki görüntüler ESA girişlerine ölçeklenir. Bu da belirli ölçeklerde görüntülerin tanımlanmasında doğruluk oranının azalmasına sebebiyet verir. ESA evrişimsel katmanlar ve tam bağlı katmanlar olmak üzere iki ana kısımdan oluşur. Evrişimsel katmanlar pencere tabanlı bir yaklaşımla görüntü üzerinde dolaşarak özellik haritaları oluşturur. Bu nedenle aslında sabit boyutlu olma zorunluluğu yoktur ve her ölçekte özellik haritaları oluşturabilir. Diğer taraftan tam bağlı katmanlar ise yapısı gereği sabit boyutlu girişleri kabul eder. Spatial Pyramid Pooling (SPP) [37] yönteminde ölçekleme işlemi en başta yapılmaz. Şekil 2.44'te görüldüğü üzere en başta ölçeklemek yerine, son evrişim katmanından sonra, tam bağlı katmandan önce araya SPP katmanı eklenerek, özellik haritalarının tam bağlı katmana uyacak şekilde sabit bir boyuta getirilmesi sağlanır.



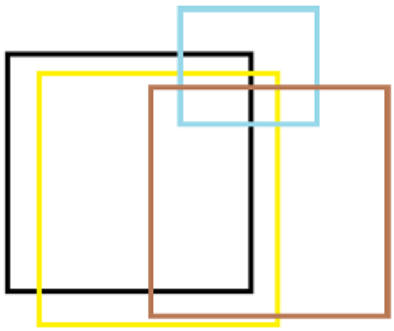
Şekil 2.44. SPP, Evrişim ve Tam Bağlı Katman Arasında Yer Alır

2.4.11. Intersection over Union

İki sınırlayıcı kutunun birbirine ne kadar örtüştüğünün bir ölçüsüdür. Sinir ağının tahmin ettiği sınırlayıcı kutunun, tahmin edilmesi istenen kutu(ground truth) ile karşılaştırılması için kullanılır. Intersection over Union (IoU), Denklem (2.7) formülünde verildiği üzere, sınırlayıcı kutuların kesişimlerinin birleşimlerine oranıdır. IoU formül gereği 0 ile 1 arasında bir değer alır. IoU, ne kadar yüksekse o kadar iyi bir tahmin yapılmış olur.

$$\textbf{IoU} = \frac{A \cap B}{A \cup B} \tag{2.7}$$

Şekil 2.45'deki siyah kutuyu görüntü üzerindeki nesneyi sınırlayan gerçek sınırlayıcı kutu olarak kabul edersek, siyah kutu ile kesişimlerinin birleşimlerine oranı en yüksek olan, en yüksek IoU oranına sahip olan sarı kutudur.



Şekil 2.45. Kesişen Tahminler

2.4.12. Complete Intersection over Union

IoU yöntemi, yanlızca sınırlayıcı kutuların kesişim ve birleşimini hesaba katar. Complete Intersection over Union (CIoU) [38] yöntemi ise sınırlayıcı kutuların kesişim alanı, merkez nokta uzaklığı ve birbirlerine göre en boy oranı olmak üzere 3 geometrik özelliği hesaba katar. Böylece sınırlayıcı kutu tahmininde daha hızlı yakınsama ve daha iyi bir performans sunar.

$$\text{CIoU} = \text{IoU} + \frac{p^2(k, k^g)}{c^2} + \alpha v \tag{2.8}$$

Denklem (2.8)'deki formülde, k tahmin edilen sınırlayıcı kutunun orta noktası, k^g gerçek sınırlayıcı kutu orta noktası değerleri ve $p(k, k^g)$ bu iki sınırlayıcı kutu merkez noktaları arasındaki öklid mesafesidir. Formülde c her iki sınırlayıcı kutuyu kapsayacak şekilde çizilebilen en küçük sınırlayıcı kutunun çapraz uzunluğu, α pozitif bir değer ve v en boy oranının tutarlığını belirtir.

En boy oranı tutarlılık değeri olan v değeri Denklem (2.9)'a göre hesaplanır. Formülde, w^g değeri gerçek sınırlayıcı kutunun genişliği, h^g değeri gerçek sınırlayıcı kutunun yüksekliği, w tahmini yapılan sınırlayıcı kutunun genişlik değeri, h tahmini yapılan sınırlayıcı kutunun yükseklik değeridir.

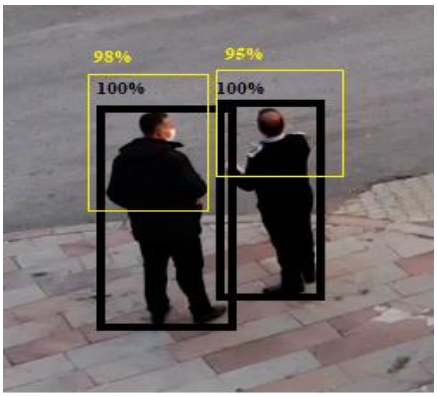
$$v = \frac{4}{\pi^2}\arctan\left(\frac{w^g}{h^g} - arctan\frac{w}{h}\right)^2 \tag{2.9}$$

α Değeri, Denklem (2.10)'a göre hesaplanır.

$$\alpha = \frac{v}{(1 - IoU) + v} \tag{2.10}$$

2.4.13. NonMaximum Supression

NonMaximum Supression, nesne tespit sistemlerinde son aşamada, nesne için en ideal sınırlayıcı kutuyu seçmek için kullanılır. Aynı nesneyi temsil eden düşük skorlu kutular elenerek, her bir nesnenin bir adet kutu ile ifade edilmesi sağlanır.

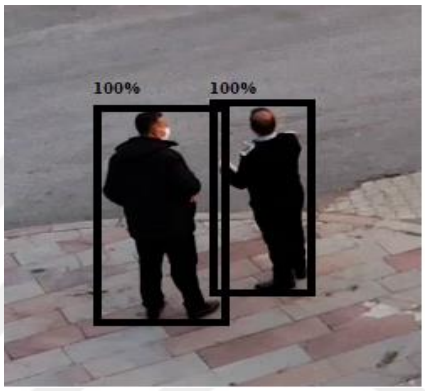


Şekil 2.46. Nesne tahmini yapılan sınırlayıcı kutular

Algoritması;

1. En yüksek skora sahip sınırlayıcı kutu S seçilir.
2. Daha sonra S ile diğer sınırlayıcı kutular arasında IoU karşılaştırması yapılır
3. S ile %50 den fazla olan IoU sahip sınırlayıcı kutular çıkarılır.(%50 oranı değiştirilebilir)
4. Daha sonra sıradaki en yüksek skora sahip sınırlayıcı kutu seçilir. Kalmamışsa bitirilir.
5. 2. Aşamaya git

Şekil 2.46'daki görüntüye NonMaximum Supression algoritması uygulanırsa Şekil 2.47'deki gibi görüntü üzerindeki nesneleri en iyi ifade edebilecek sınırlayıcı kutular kalır.



Şekil 2.47. NonMaximum Supression Algoritması Sonrası

2.5. Nesne Tespit Amacıyla Kullanılan ESA Modelleri

ESA modelleri 90'lı yıllarda sıklıkla kullanıldı, fakat destek vektör makinelerinin yükselişi ile gözden düştü. 2012 yılında Krizhevsky ve arkadaşlarının ILSVRC'deki başarısından sonra tekrar dikkatleri üzerine çekti. Bu başarının temelini 1.2 milyon etiketlenmiş görüntü üzerinde eğitim yaptırılması, nonlineerleştirme ve dropout normalleştirmesi gibi birtakım geliştirmeler oluşturur [15].

Model, görüntünün özellik haritalarını çıkarmak amacıyla ESA kullanır. ESA, modelin iskeletidir. Model bu iskeleti, adeta bir sanatçının eserini işlediği gibi işleyerek, nesne tespit sisteminden beklenen iki temel unsurda; nesne sınıflandırmada ve nesne lokalizasyonunda; maksimum doğruluğu elde etmeye çalışır. Model, ESA'nın kaç katmandan oluşacağını ve her katmanda kaç adet nöron olacağını belirler. Katman ve nöron sayısının fazla olması, modelin parametre sayısını ve donanımın işlem yükünü arttırır. AlexNet 5 evrişim katmanı, 3 tam bağlı katman içermekteyken [39], donanımın gelişmesiyle birlikte, güncel modellerde katman sayısı

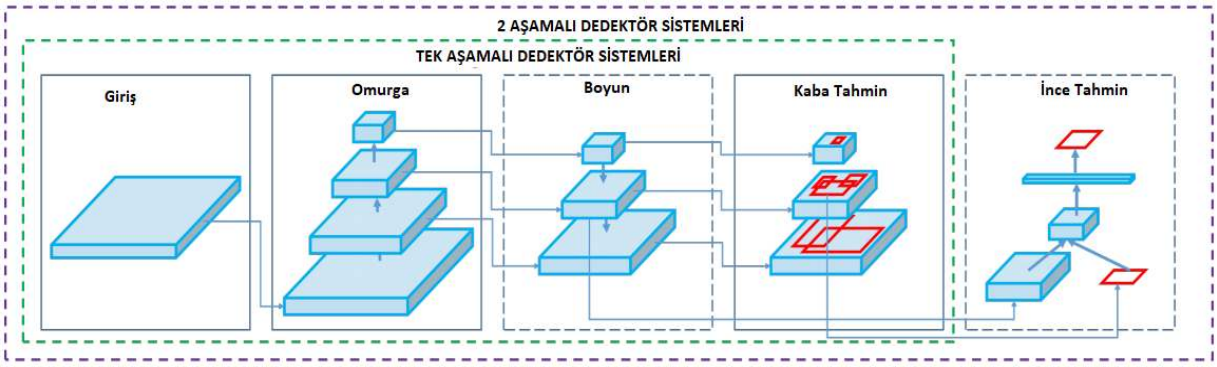
fazladır. Katmanlardaki evrişim işlemlerinde model; uygulanacak filtrelerin sayısını, ölçülerini ve kanal sayılarını, atlama miktarını, padding uygulanıp uygulanmayacağını, uygulanıyorsa nasıl bir teknik uygulanacağını belirler.

Evrişim işleminden sonra model tarafından belirlenen aktivasyon fonksiyonu aracılığıyla bir sonraki katmana aktarılacak nöron çıkışları elde edilir. Pooling işlemleri için ESA modellerinde genellikle maximum pooling ya da average pooling teknikleri model tarafından tercih edilir. Maximum pooling büyük nesnelerin özellik haritalarının çıkarımında daha başarılı iken, average pooling görece küçük nesnelerin özellik haritalarını çıkarmada başarılıdır.

Son katmanlarda genellikle tam bağlı katmanlar bulunur. Son evrişim ya da pooling işleminden sonra veri düzleştirilir ve tam bağlı katmanlar aracılığıyla sınıflandırma ve sınırlayıcı kutu tahminleri yapılır.

Model, kendine özel kayıp fonksiyonu aracılığıyla ESA'nın mevcut değerlerle ürettiği çıktının, hedeflenen çıktıdan hata payını hesaplayıp, geri yayılım teknikleri ile ağdaki filtre ağırlıklarının güncellenmesini sağlar. Böylece her iterasyonda hedeflenen değere daha yakın değerler elde edilir.

Nesne tespit amacıyla kullanılan ESA modelleri, İnce Tahmin ve Kaba Tahmin modelleri olmak üzere 2 farklı kategoriye ayrılır. İnce tahmin modelleri görüntü üzerinde nesne tahmin işlemini 2 aşamalı olarak gerçekleştirir. İlk aşamada görüntü üzerinde nesne olabilirliği yüksek bölgeler belirlenir. İkinci aşamada bu bölgeler üzerinde yoğunlaşılıp, nesnelerin sınıf tahmini yapılır ve sınırlayıcı kutu koordinatları belirlenir. Kaba tahmin modellerinde ise sinir ağının, nesnelerin sınıf tahminlerini ve sınırlayıcı kutu koordinatlarını tek aşamada bulması istenir. Bu nedenle tek aşamalı modeller, 2 aşamalı modellere göre genel olarak daha hızlıdır fakat doğruluk oranları genel olarak daha düşüktür. Şekil 2.48'de bu modellerin test anında görüntüyü nesne tespiti amacıyla ele alış şekli gösterilmiştir.



Şekil 2.48. İnce Tahmin ve Kaba Tahmin Farkı

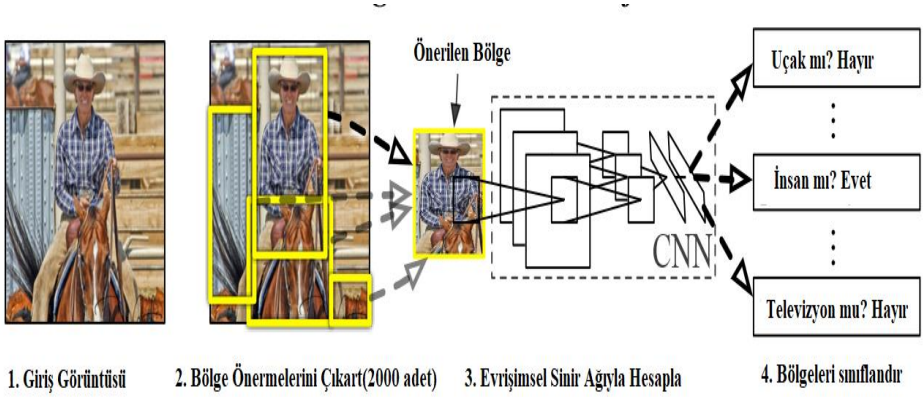
2.5.1. RCNN Ailesi

RCNN ailesi 2014 yılında Ross Girshick ve arkadaşları [15] tarafından geliştirilen "Bölge Temelli Evrişimsel Sinir Ağları" modelinin üyeleridir. R-CNN, Fast R-CNN ve Faster-RCNN modelleri nesne lokalizasyonu ve nesne sınıflandırma için kullanılır.

RCNN model ailesini, kayan pencere tabanlı nesne tespit sistemlerinden ayıran en belirleyici özelllik, görüntü üzerinde nesne tespiti yapılması istenen bölgelerin önişlemler ile tahmin edilmesidir.

2.5.1.1. R-CNN Nesne Tespit Modeli

Şekil 2.49'da nesne tespit aşamaları gösterilen R-CNN modeli [15], "Bölge önermeleri modülü", "Özellik çıkarıcı modül" ve "Sınıflandırıcı modül" olmak üzere 3 modülden oluşur. Bölge önermeleri modülünde, giriş görüntüsü üzerinde, kategorilerden ve sınıflardan bağımsız olarak aday bölgeler belirlenir. Nesne aranacak bu aday bölgeler yaklaşık 2000 adettir. Sonraki modülde, belirlenen bu bölgeler üzerinde ESA aracılığıyla, her bir bölge için ayrı ayrı sabit boyutlu görüntü özellik vektörleri çıkarılır. Son modülde, çıkarılan bu özelliklere göre şayet nesne tespit edilmişse, tespit edilen nesnenin bilinen sınıflardan hangisine ait olduğu, Lineer destek vektör makinaları aracılığıyla bulunur.



Şekil 2.49. R-CNN [15]

Test anında, R-CNN modeli bölge önermesi yapmak için "selective search" algoritmasını kullanır. Görüntü üzerinde bulunan bölgelerin genişlik ve yükseklik değerleri dönüştürülerek (227x227) girişli, 5 evrişim katmanı 2 tam bağlı katmana sahip olan evrişimsel sinir ağına verilir.

Daha sonra her bir sınıf için o sınıfa özel eğitilmiş Destek Vektör Makinası(DVM) tarafından görüntü vektörlerine skor değerleri verilir. Son olarak NonMaximum Supression uygulanır [15].

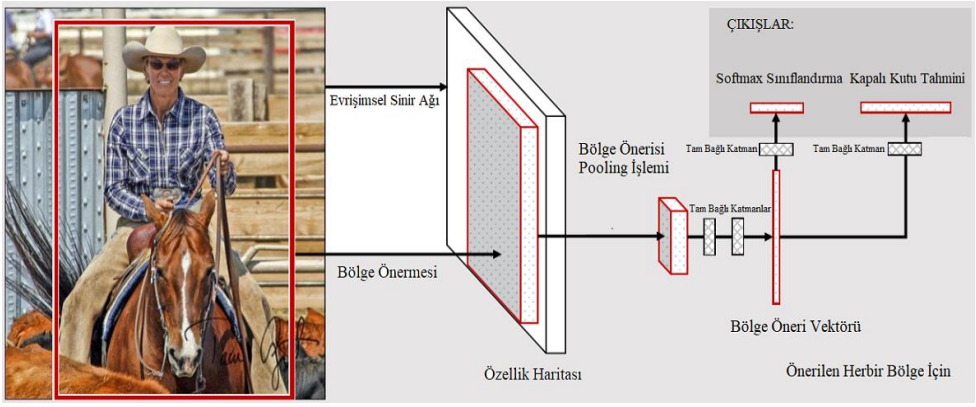
R-CNN modeli PASCAL VOC 2010 üzerinde 53.7% mAP başarı oranına sahiptir. Bölge önermeleri ve özellik çıkarımları görüntü başına modern bir CPU ile yaklaşık 53 saniye iken GPU ile yaklaşık 13 saniye sürer [15].

R-CNN Problemleri

- Her görüntü 2000 adet bölge önermesinde bulunduğundan eğitim uzun vakit alır.
- Görüntüdeki nesneleri tanımlaması GPU ile 13 saniye civarında vakit alır.
- Bölge önermelerinin özellik haritalarını kaydetmek için yüksek disk kapasitesi ve kullanımı gerekir.

2.5.1.2. Fast R-CNN Nesne Tespit Modeli

Fast R-CNN modelinde bölge önerileri için R-CNN modelinin aksine, Şekil 2.50'de gösterildiği üzere, görüntü ilk olarak evrişim katmanına aktarılarak bölge önerileri belirlenir. Bu sayede tek bir özellik haritası oluşturulacağı için disk alanında fazla yer kaplamaz. Bölge önerileri değişik ölçülerde olduğundan tahmin için sinir ağına verilmeden önce aynı ölçülere getirilir. Ayrıca sınıflandırma kısmında metot olarak DVM değil Softmax kullanılır [40].



Şekil 2.50. Fast R-CNN [41]

Fast R-CNN modeli, R-CNN modeline göre eğitim süresinde 9 kat, test anında 213 kat daha hızlıdır ve PASCAL VOC 2012 veri setinde daha yüksek bir doğruluk oranı sağlar [41].

2.5.1.3. Faster R-CNN Nesne Tespit Modeli

Ren ve arkadaşları [42] tarafından yapılan çalışmada, önceki modellerin hızını büyük ölçüde azaltan, modelin çıkmazı olarak görülen bölge önermesi algoritmalarının modele getirdiği yük azaltılmıştır. Bölge önermesi için kullanılan ESA 'nın katmanlarıyla, nesne tespitinden sorumlu ESA'nın katmanları ortak kullanılarak, birleşik bir sinir ağı modeli oluşturulmuştur. Böylece bölge önermesi için kullanılan ESA'nın külfeti nerdeyse ortadan kalkmış ve bu modelle 5-17 Fps değerlerine ulaşılmıştır.

2.5.2. YOLO Ailesi

Nesne tespit sistemleri, nesneleri tespit etmek için görüntü sınıflandırıcılarını kullanırlar. Sınıflandırıcılar, görüntüde belirli bir nesne için eğitilmiş sınıflandırıcı tarafından tespit edilebilen belirli bir nesne olup/olmadığına bakar ve görüntünün belirli bir sınıfa ait olduğuna dair bir ihtimal verir. Sınıflandırıcılar bütün resme bakar. Bu nedenle, nesne tespiti için, görüntü üzerindeki belirli çerçeveler seçilir, bu çerçeveler bütün bir görüntüymüş gibi, değişik ölçeklerde yeniden boyutlandırılır ve çerçeve içindeki nesneyi tanımlamak için, her bir değişik nesne için belirlenmiş sınıflandırıcı her bir ölçek için tekrar çağrılır ve belirlenmiş çerçeve içindeki sınıf tahmin edilir [43].

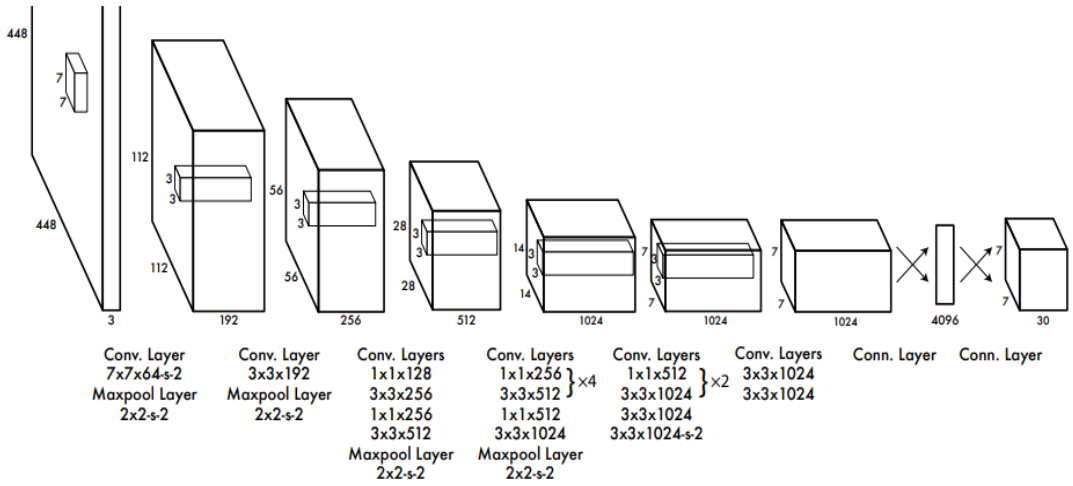
DPM modelinde, belirlenen bir çerçeve, resmin bütün koordinatlarında kaydırılarak her bir çerçeve için sınıflandırıcılar tekrar çağrılır. R-CNN modelinde görüntü üzerindeki potansiyel çerçeveler tahmin edilir, daha sonra bu çerçeveler için sınıflandırıcılar çağrılır [41].

Tek aşamalı detektörlerden olan YOLO(You Only Look Once) modeli, sınıflandırma ve lokalizasyon aşamalarını bir arada gerçekleştirilir. Network defalarca kez değişik ölçeklerde çağrılmaz. Böylece yüksek hız elde edilmiş olur. Şekil 2.51'de görüldüğü üzere, YOLO modelinde nesne tanıma amacıyla ESA bir kez çalıştırılarak, tek seferde hem sınıf değerleri, hem de sınırlayıcı kutu koordinatları bulunur. Kayan pencere tabanlı ve bölge önermeli modellerin aksine YOLO modelinde eğitim ve test anında bütün görüntü üzerinde çalışılır.

Şekil 2.51. YOLO Tek Aşamalı Nesne Tanıma Modeli [43]

YOLO modeli, eğitim ve görüntüden özellik çıkarımı için modelin omurgası olarak, C programlama dili ve Nvidia Ekran Kartı üreticisi tarafından geliştirilen CUDA (Compute Unified Device Architecture) kütüphanesi ile yazılmış açık kaynak kodlu yapay sinir ağı kütüphanesi olan Darknet Framework'u [44] kullanır. YOLO modelinin diğer frameworklar üzerinde de uygulamaları bulunmaktadır. YOLO modelinde, eğitim ve test anında, giriş görüntüleri hangi ölçekte olursa olsun, model tarafından sinir ağının girişlerine ölçeklenir, en boy oranı korunmaya çalışılmaz. Modelin sinir ağı 3 renk kanalında(RGB) veya 1 kanallı görüntüleri kabul eder.

2.5.2.1. YOLO v1 Nesne Tespit Modeli

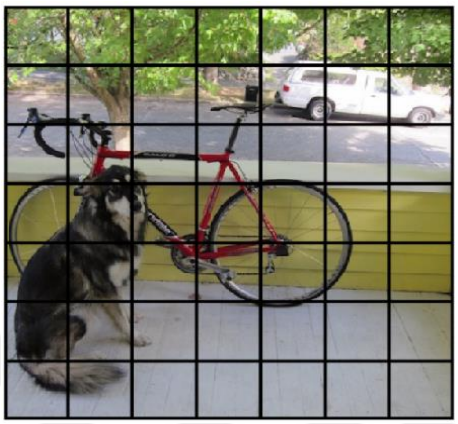


Şekil 2.52. Yolo v1 Modeli

YOLO v1 [43], 1000 sınıflı ImageNet üzerinde ön eğitime tutulmuştur. Ön eğitim aşamasında, Şekil 2.52'deki ilk 20 katmanın ardından, ortalama havuzlama katmanı ve tam bağlı katman kullanılmıştır. Nesne tespiti için, ön eğitilmiş ağlara evrişimsel katman ve tam bağlı katman eklemenin performansı arttırabileceğine dair yapılan çalışmalara istinaden, ön eğitimin ardından modele rastgele ağırlıklarla 4 evrişim katmanı ve 2 tam bağlı katman eklenmiştir. Şekil 2.52'de test anındaki katman yapısı verilen YOLO v1, nesne tespiti için 24 evrişim katmanı ve bu

katmanların ardında 2 tam bağlı katman içerir. Son katmanda lineer aktivasyon fonksiyonu kullanılır, diğer bütün katmanlarda LReLU kullanılır. Ağın evrişim katmanları görüntünün özellik haritalarını ortaya çıkarırken tam bağlı katmanlar çıkış olasılıklarını ve koordinatlarını tahmin eder. Eğitimin ardından network çözünürlüğü 224x224 ten 448x448 e yükseltilmiştir.

YOLO modelinde görüntü, NxN ızgara hücrelerine bölünür. Şekil 2.53'deki örnek görüntüde N=7 kabul edilmiş ve görüntü 7x7 ızgara hücrelerine bölünmüştür.



Şekil 2.53. SxS Izgara

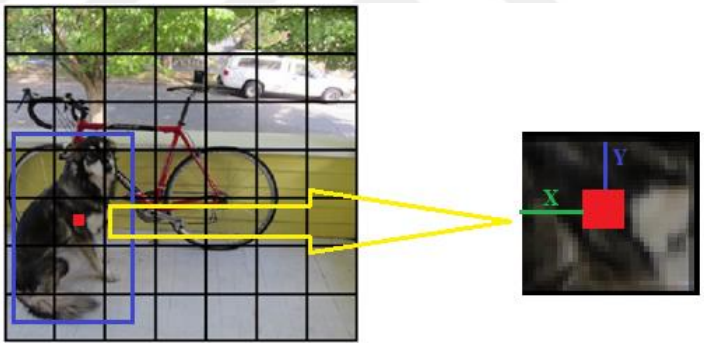
YOLO v1 modelinde, her ızgara hücresi için 2 adet sınırlayıcı kutu tahmin edilir. Toplamda NxNx2 adet sınırlayıcı kutu tahmin edilir. Her sınırlayıcı kutu, içinde nesne olma olasılığına dair bir güven skoru, nesnenin orta noktasını belirten (x,y) değerleri, nesnenin genişlik ve yüksekliğini içeren bir vektörle ifade edilir.

$$Sınırlayıcı\ Kutu\ Vektörü = \begin{bmatrix} p_c \\ s_x \\ s_y \\ s_h \\ s_w \end{bmatrix}$$

$p_c = Güven\ Skoru(Nesne\ Olasılığı),\ p_c \in R\ ve\ p_c \in [0,1]$
$s_x = Sınırlayıcı\ kutuyu\ içeren\ hücreye\ göre\ x\ eksenindeki\ uzaklığı, s_x \in R\ ve\ s_x \in [0,1]$
$s_y = Sınırlayıcı\ kutuyu\ içeren\ hücreye\ göre\ y\ eksenindeki\ uzaklığı,\ s_y \in R\ ve\ s_y \in [0,1]$
$s_h = Bütün\ görüntüye\ göre\ yüksekliği,\ s_h \in R\ ve\ s_h \in [0,1]$
$s_w = Bütün\ görüntüye\ göre\ genişliği,\ s_w \in R\ ve\ s_w \in [0,1]$

Modelde, sınırlayıcı kutunun güven skoru, hücrenin tahmini yapılan sınıfa ait bir nesne içerdiğine dair yapılan tahmin ve tahmin edilen sınırlayıcı kutunun doğruluğunun($IoU_{tahmin}^{gerçek}$) çarpımına eşittir. Hücre bir sınıfa ait bir nesne içermiyorsa sınırlayıcı kutunun güven skoru 0'dır. Güven skoru 0 olan bir sınırlayıcı kutunun diğer parametreleri önemsizdir.

Eğitim zamanında, bir nesnenin orta noktasının isabet ettiği hücre, model tarafından o nesneyi tahmin etmek üzere işaretlenir. Sınırlayıcı kutunun orta noktasını belirten koordinatlar, bütün görüntüye göre değil, sınırlayıcı kutunun ait olduğu hücreye göre hesaplanır. Sınırlayıcı kutunun merkezinin içinde yer aldığı hücrenin X eksenindeki boyu 1 kabul edilerek, sınırlayıcı kutu x değişkeni ilgili hücreye X eksenindeki uzaklığını ifade eden 0 ve 1 arası bir uzaklık değeri alır. Aynı şekilde y değişkeni de Y eksenine göre ilgili hücreye olan uzaklığı ifade eden 0 ve 1 arası bir uzaklık değeri alır. Şekil 2.54'te örnekte gösterildiği üzere, görüntüdeki nesneyi işaretleyen mavi renkte çizilmiş sınırlayıcı kutunun orta noktası kırmızıyla işaretlenmiştir. Mavi sınırlayıcı kutunun merkezini belirten x ve y değerleri, bu kırmızı noktanın, noktayı içeren hücreye olan mesafesine göre değer alır.



Şekil 2.54. Sınırlayıcı Kutu Merkez Nokta Hesaplanması

Sınırlayıcı kutunun genişlik ve yükseklik değerleri ise bütün görüntüye göre değer alır. Örneğin, 400x400 pixel ölçülerinde bir görüntünün içerisinde yer alan 100x200 ölçülerinde bir sınırlayıcı kutunun genişlik ve yükseklik değerleri, (0.25,0.50) şeklinde ele alınır.

Her hücre vektörü, sınırlayıcı kutu değerleri ve ilgili hücrenin tahmin edilmesi istenen sınıflara ait olasılık değerlerini içerir. YOLO v1 modelinde hücre başına 2 adet sınırlayıcı kutu tahmini yapıldığı için hücre vektörü 2 adet sınırlayıcı kutu ve sınırlayıcı kutu sayısından bağımsız olarak hücrenin sınıf olasılık değerlerini içerir. Test sırasında, modelin hedeflendiği, bulunması istenen sınıfların sayısı n olmak üzere, her sınırlayıcı kutu vektörü 5 adet parametre (p_c, s_x, s_y, s_w, s_h) içerdiğinden, her ızgara hücresi (2x5+n) adet parametre içerir. Yani her hücre (2x5+n)

elemanlık bir vektörle temsil edilir. ESA çıkışında, toplam S x S x (2 x 5+n) adet çıkış(tensor) oluşturulmuş olur.

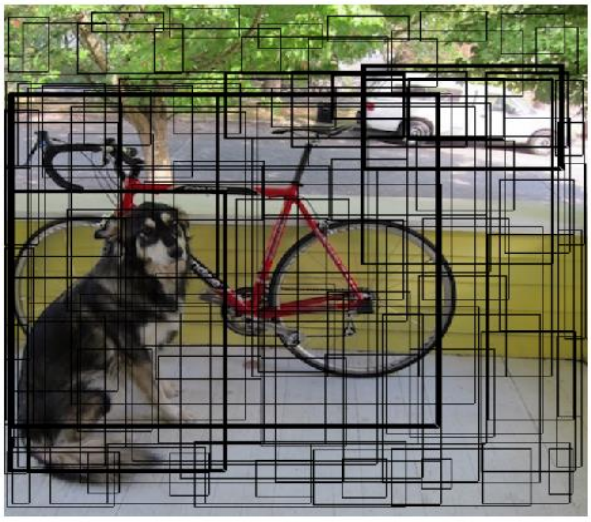
$$S_{i1}=\begin{bmatrix} p_c \\ s_x \\ s_y \\ s_h \\ s_w \end{bmatrix} \quad S_{i2}=\begin{bmatrix} p_c \\ s_x \\ s_y \\ s_h \\ s_w \end{bmatrix} \quad Hücre_i=\begin{bmatrix} p_c \\ s_x \\ s_y \\ s_h \\ s_w \\ p_c \\ s_x \\ s_y \\ s_h \\ s_w \\ c_1 \\ \cdot \\ c_n \end{bmatrix}$$

i nolu hücreye ait Sınırlayıcı Kutu1

i nolu hücreye ait Sınırlayıcı Kutu2

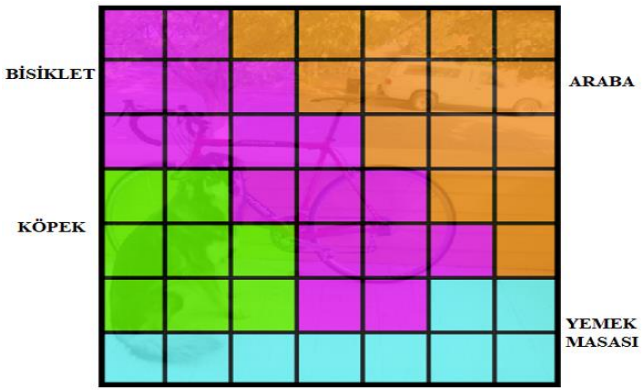
Tahmin edilmesi istenen sınıflara dair olasılıklar

Şekil 2.53'deki örnek görüntü üzerinde N=7 olmak üzere, modelin tahmin ettiği (7x7x2) adet sınırlayıcı kutu Şekil 2.55'te gösterilmektedir. Güven skoru yüksek olan sınırlayıcı kutular daha kalın çizilmiştir.



Şekil 2.55. Sınırlayıcı Kutu Tahminleri ve Güven Skoru

Modelde, sınıf olasılık değerleri sınırlayıcı kutu değerleriyle aynı anda birlikte tahmin edilir. Şekil 2.56'daki Sınıf tahmini haritası, her bir hücrenin bütün sınıflara ait olasılık değerlerinden en yüksek olanı temel alınarak oluşturulmuştur.

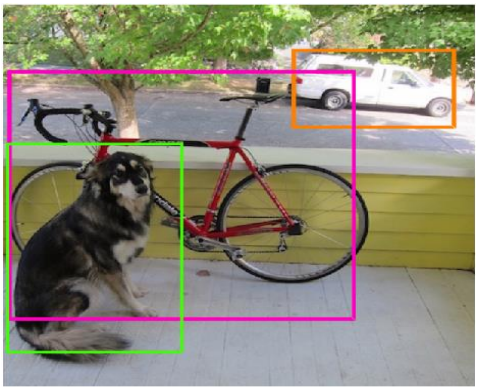


Şekil 2.56. Sınıf Haritası

Şekil 2.57'de örnek görüntüye ait Sınırlayıcı Kutu Tahminleri ve Sınıf Tahmini Haritası birleşimi gözükmektedir.

Şekil 2.57. Sınırlayıcı Kutu+Sınıf Tahmini Haritası

ESA tarafından tahmin edilen bu değerlere, NonMaximum Supression yöntemi uygulanarak gereksiz sınırlayıcı kutulardan arındırılır ve Şekil 2.58'deki görüntü elde edilir.



Şekil 2.58. Örneğin Son Hali

Eğitim sırasında, görüntü sinir ağına verilir ve sinir ağının, görüntü üzerinde önceden işaretlenmiş sınırlayıcı kutu ve bu sınırlayıcı kutu ile işaretlenmiş nesnenin sınıfını bulması istenir. Model, mevcut ağırlıklarla sinir ağının ürettiği sonucun, istediği sonuca ne kadar yakın olduğunu kayıp fonksiyonu aracılığıyla hesaplayabilir. Bu nedenle kayıp fonksiyonunda, bulunması istenen sınırlayıcı kutunun koordinat, en, boy oranının hesaba katılması gerektiği açıktır. Eğitim setindeki, herhangi bir görüntüde yer alan nesne, ızgara hücresinin bir ya da daha fazla hücresini kaplayabilir ya da kısmen yer alabilir. Nesneler değişik şekillerde olabilir, sınırlayıcı kutuların ise düzgün dikdörtgen bir şekli vardır ve nesne eğitim verisinde hâlihazırda sınırlayıcı kutu içinde ifade edilmiştir. YOLO modelinde, sinir ağının kayıp fonksiyonu, nesneyi çevreleyen sınırlayıcı kutunun orta noktasının düştüğü hücredeki sınırlayıcı kutu tahminlerine göre hesaplanır.

YOLO v1 modeli, 2015 yılında yayınlanan ilk makalesinde [16] her hücre yalnızca bir adet sınırlayıcı kutu tanımlayacak şekilde tasarlanmıştır. Bu yüzden Denklem (2.11)'de verilen kayıp fonksiyonu, aynı makalenin her hücrenin 2 adet sınırlayıcı kutu tanımladığı, Denklem (2.12) 'de verilen formülle hesaplanan 2016 yılı versiyonuna [43] göre daha sadedir. 2015 yılındaki versiyonu Denklem (2.11)'de gösterilmiştir. Bu versiyonda kayıp fonksiyonu 2 unsuru göz önünde bulundurur.

- Lokalizasyon Kaybı
- Sınıflandırma Kaybı

$$\sum_{i=0}^{48}\left(\lambda \mathbf{1}_i^{obj}\left((x_{i-}\hat{x}_i)^2+(y_{i-}\hat{y}_i)^2+\left(\sqrt{w}_{i-}\sqrt{\hat{w}_i}\right)^2+(\sqrt{h_i}-\sqrt{\hat{h}_i})^2\right)+\sum_{c\in \text{sınıflar}}(p_i(c)-\hat{p}_i(c))^2\right) \tag{2.11}$$

Model görüntüyü 7x7 ızgara hücrelerine böler. Denklem (2.11)'de verilen kayıp fonksiyonu 49 adet hücrenin her birinde meydana gelen kaybın toplamı olarak tasarlanmıştır. Eğitim zamanında nesneler sınırlayıcı kutu içinde etiketlenmiş bir şekilde modele verildiğinden, kayıp fonksiyonunda, model; sinir ağının nihayetinde ürettiği sınırlayıcı kutu tahminleri ve sınıf değeri olasılığının, gerçek değerle olan farkını hesaplar. Sınırlayıcı kutu tahmini, şayet bir hücre nesne içeriyorsa yapılmaktadır. Formülde bu yöntem, $\mathbf{1}_i^{obj}$ sembolüyle gösterilmiştir. Bir hücre nesne içeriyorsa ilgili i hücresine ait $\mathbf{1}_i^{obj}$ değeri 1, içermiyorsa $\mathbf{1}_i^{obj}$ değeri 0 olarak ele alınır. Formülde şapkalı ifadeler, sinir ağının çıkışından elde edilen, modelin eğitim zamanında yaptığı

tahmin değerleridir. Formülde x ve y değişkenleri sınırlayıcı kutu ile ifade edilen nesnenin merkez noktasını içeren hücreye, X ve Y eksenlerindeki uzaklık değerleridir. Sınırlayıcı kutu genişlik ve yükseklik değerleri sırasıyla w ve h değişkenleri ile belirtilmiştir. Sınırlayıcı kutu tahmininde meydana gelen kaybın önemini λ pozitif değeri belirler. Bu versiyonda λ=4 olarak seçilmiştir.

Sınıflandırma kaybı, ilgili hücrenin içerdiği nesneyi doğru sınıflandırması için kullanılır. Model tarafından hâlihazırda bilinen, i hücresinin c sınıfına ait olma değeri $\boldsymbol{p_i(c)}$ sembolü ile koşullu sınıf olasılığı ise $\boldsymbol{\hat{p}_i(c)}$ sembolüyle gösterilmiştir. İlgili i hücresi nesne içeriyorsa, nesnenin ait olduğu c sınıfına ait $\boldsymbol{p_i(c)}$ değeri 1'dir. Diğer bütün sınıfların $\boldsymbol{\hat{p}_i(c)}$ tahmin değerinin 0 olması istenir. İçinde herhangi bir nesne içermeyen hücreler arka plan verisi olarak değerlendirilir. Bu nedenle nesne içermeyen i indisli hücreye ait $\boldsymbol{p_i(c)}$ değeri 0 kabul edilerek $\boldsymbol{\hat{p}_i(c)}$ değeri 0'a yaklaştırılır.

Ortalama hata karesi formülü, küçük kutularda meydana gelen hata miktarıyla, büyük kutularda meydana gelen hata miktarını aynı değerde kabul eder. Küçük bir kutudaki 1 pixel hatanın, büyük bir kutudaki 1 pixel hatayla aynı önemde kabul edilmesi istenmez. Modelde sınırlayıcı kutunun genişlik ve yükseklik değeri yerine, genişlik ve yükseklik değerlerinin karekök değerlerinin tahmin edilmesi istenerek, bu problemin kısmen önüne geçilmesi amaçlanmıştır.

YOLO v1 modeli, hücre başına 2 adet sınırlayıcı kutu tanımlanan 2016 yılı versiyonunda [43] açıklanan kayıp fonksiyonu Denklem (2.12)'de verilmiştir. Modelde kayıp fonksiyonu 3 unsuru göz önünde bulundurur.

- Lokalizasyon Kaybı
- Güven Skoru Kaybı
- Sınıflandırma Kaybı

$$\lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbf{1}_{ij}^{obj}\left[(x_i-\hat{x}_i)^2+(y_i-\hat{y}_i)^2+\left(\sqrt{w_i}-\sqrt{\hat{w}_i}\right)^2+\left(\sqrt{h_i}-\sqrt{\hat{h}_i}\right)^2\right] \tag{2.12}$$
$$+\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbf{1}_{ij}^{obj}\left[(C_i-\hat{C}_i)^2\right]+\lambda_{noobj}\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbf{1}_{ij}^{noobj}\left[(C_i-\hat{C}_i)^2\right]$$
$$+\sum_{i=0}^{S^2}\mathbf{1}_{i}^{obj}\sum_{c\in\text{sınıflar}}(p_i(c)-\hat{p}_i(c))^2$$

Formülde, i nolu hücrede j. sınırlayıcı kutu nesneyi tespitten sorumluysa, yani bir nesnenin orta noktası i nolu hücreye denk geliyorsa ve ilgili hücredeki sınırlayıcı kutuların IoU değerleri arasından j indisli sınırlayıcı kutu en büyük değere sahipse, $\mathbf{1}_{ij}^{obj} = \mathbf{1}$ aksi takdirde $\mathbf{1}_{ij}^{obj} = \mathbf{0}$ 'dır. Sınırlayıcı kutu koordinatlarının, genişlik ve yükseklik değerlerinin tahmininin kayıp fonksiyonu için ağırlığını λ_{coord} pozitif değeri belirler. Bu versiyonda, λ_{coord} değeri 5 olarak seçilmiştir. $\mathbf{1}_{ij}^{noobj}$ değeri ise $\mathbf{1}_{ij}^{obj}$ değerinin tam tersidir. Nesne içermeyen arka plan görüntüsünün doğru hesaplanması amacıyla kullanılır. Arka plan tespitinin kayıp fonksiyonu için ağırlığını λ_{noobj} pozitif değeri belirler. Bu versiyonda, λ_{noobj} değeri 0.5 olarak seçilmiştir. Sınıflandırma kaybı eğer bir hücre bir nesne içeriyorsa hesaba katılır. Formülde i hücresi şayet nesne içeriyorsa $\mathbf{1}_{i}^{obj}$ değeri 1, nesne içermiyorsa $\mathbf{1}_{i}^{obj}$ değeri 0 kabul edilir. Eğer sınırlayıcı kutuda bir nesne tespit edilmişse, güven skoru kaybı denklem(2.13)'teki formüle göre hesaplanır:

$$\sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbf{1}_{ij}^{obj} \left[(\boldsymbol{C_i} - \widehat{\boldsymbol{C}}_i)^2 \right] \tag{2.13}$$

Eğitim zamanında, işaretlenmiş hücrelerin sınırlayıcı kutuları arasından IoU değeri en yüksek olan model tarafından seçilir ve o nesneyi tespit etmekten sorumlu olur. Yani sonuç vektöründe ilgili hücrenin ilgili sınırlayıcı kutu değer tahminlerini içeren bölüm model tarafından önemsenir, ilgili hücrenin diğer sınırlayıcı kutularına ait değerler model tarafından baskılanır. Bu nedenle kayıp fonksiyonunda, i nolu hücrede j nolu sınırlayıcı kutu nesneyi tespitten sorumluysa, $\mathbf{1}_{ij}^{obj}$**=1** aksi takdirde $\mathbf{1}_{ij}^{obj}$**=**0'dır. Yani i nolu hücre bir nesnenin orta noktasına tekabül ediyor ve ilgili hücredeki sınırlayıcı kutular arasından j indexli sınırlayıcı kutu model tarafından o nesneyi tespit etmekle görevli işaretlenmişse değeri 1'dir. Aksi takdirde 0 verilerek kayıp fonksiyonu değerlendirmesine katılmaz.

Güven skoru, bir hücrenin bir nesne içeriyor olması ile ilgili verilen bir değerdir. Tahmin edilmesi istenen sınıfların her biri için hücre vektöründe bir olasılık değeri hesaplanır. Sınıflandırma olasılık değerleri gözönünde bulundurularak, hücrenin nesne içermesine dair yaptığı $\boldsymbol{C_i}$ i hücresindeki j nolu sınırlayıcı kutunun güven skoru, $\widehat{\boldsymbol{C}}_i$ i hücresindeki j nolu sınırlayıcı kutunun gerçek değerle olan IoU değeridir.

Eğer sınırlayıcı kutuda bir nesne tespit edilmemişse, bu durumdaki güven skoru kaybı denklem(2.14)'te verilmiştir.

$$\lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbf{1}_{ij}^{noobj} \left[\left(C_i - \widehat{C}_i \right)^2 \right] \tag{2.14}$$

$\mathbf{1}_{ij}^{noobj}$, $\mathbf{1}_{ij}^{obj}$ 'nin tümleyeni, $\widehat{C}_i$ i nolu hücredeki j sınırlayıcı kutusunun güven skorunu ve λ_{noobj} ise arkaplan tespiti için, kayıp fonksiyonunda güven skoru kaybının ağırlığını azaltır.

2.5.2.2. YOLO v2 Nesne Tespit Modeli

YOLO v2 modelinde, YOLO v1 modelindeki katmanlar kullanılır. YOLO v1 modelinde batch normalizasyonu yer almaz. YOLO v2 modelinde bütün evrişim katmanlarına batch normalizasyonu uygulanarak, modelin başarı oranı arttırılmıştır [45].

Yolo v1 modelinde, her hücrede yalnızca 1 adet nesnenin tahmin edilebiliyor oluşu, aynı hücre içinde yer alan 2. bir nesnenin tahminini mümkün kılmamaktadır. Bir görüntüde yakın ve küçük nesnelerin orta noktaları aynı hücre içinde yer alabilir. YOLO v2 modelinde Anchor Box(Öncül Kutu) kullanılarak orta noktası aynı hücre içinde yer alan birden fazla nesnenin eğitimi ve tespiti sağlanabilmiştir.

Öncül Kutular çeşitli yükseklik ve genişlik değerlerine sahip öntanımlı dikdörtgen kutulardır. Bu kutuların görüntü içerisindeki yerleşimi model tarafından yapılır. YOLO v2 modelinde her hücre için 5 adet öncül kutu tanımlanır. Öncül kutuların ölçüleri, veri seti üzerindeki belirlenmiş sınırlayıcı kutuların üzerinde k-ortalama kümeleme yapılarak seçilmiştir.

Eğitim sırasında bir nesnenin orta noktasına tekabül eden hücre ve ilgili hücre ile en yüksek IoU oranına sahip öncül kutu, o nesnenin tespitinden sorumlu olur. Hedef etiketinde (hücre, öncül_kutu) olarak kodlanır.

2.5.2.3. YOLO v3 Nesne Tespit Modeli

Eğitim için, 53 evrişimsel katman (Darknet-53) içerir. YOLO v3 mimarisinde, nesne tespit görevleri için orijinal mimarinin üzerine 53 adet katman daha yığılır ve toplamda 106 katmandan oluşur. Her evrişim işleminin ardından Batch Normalization ve LReLU uygulanır [46]. Pooling katmanı bulunmaz. Onun yerine evrişimsel katman bulunur. Özellik haritalarını aşağı örnekleme için pooling yerine 2x2, 2 atlamalı filtre ile evrişimsel katman kullanır. Bu sayede max pooling katmanının görmezden geldiği alt düzey özelliklerin kaybolmasını önler. Bunun sonucu olarak alt düzey özelliklerin yakalanması ve küçük nesnelerin tespitinde başarıyı arttırmaya yardımcı olur.

YOLO v3 modelinde, sinir ağı varsayılan olarak (416x416) ölçülerinde 3 renk kanalında görüntüleri girişinden alır. Sinir ağının giriş ölçeği, 32'nin katı olacak şekilde değiştirilebilir. Eğitim ve tespit amacıyla herhangi bir ölçekteki görüntü sinir ağına parametre olarak verilebilir. Sinir ağı bunu kendi girişlerine göre, görüntünün en boy oranını koruma şartı olmadan ölçekleyecektir.

YOLO v3 modelindeki en büyük yenilik, diğer modellerden farklı olarak yalnız son katmanda değil aynı zamanda, ara katmanlarda da tespitler yapılmasıdır. 82.katman, 94.katman ve 106. katmanda nesne tespiti yapılır. 82. katmanda NxN görüntü üzerinde, (N/32) x (N/32) ölçülerinde, 94. katmanda (N/16) x (N/16) ölçülerinde ve 106. katmanda (N/8) x (N/8) ölçülerinde örneklenir. N=416 olursa bu katmanlarda, sırasıyla 13x13, 26x26 ve 52x52 pixel ölçeklerinde görüntü özellik haritaları elde edilir. 13x13 Büyük nesnelerin tespiti için, 26x26 orta büyüklükte nesnelerin tespiti için, 52x52 küçük nesnelerin tespiti için önemlidir. Yolov3 modelinde her hücre için 3 adet sınırlayıcı kutu tahmin edilir. Her sınırlayıcı kutu (p_c, b_x, b_y, b_w, b_h) şeklinde kodlandığı için, özellik haritalarının boyutları, n sınıf sayısı olmak üzere, [13,13, 3*(5+n)], [26,26, 3*(5+n)], [52,52, 3*(5+n)] olur. Yani bu 3 farklı ölçeğin her biri için 3*(5+n) adet özellik haritası elde edilmiş olur. Her bir hücre için 3 adet sınırlayıcı kutu tahmin edildiğine göre, (13x13) ölçeğinde 13x13x3=507 adet, (26x26) ölçeğinde 26x26x3=2028 adet 52x52x3=8112 adet sınırlayıcı kutu, toplamda 507+2028+8112=10647 adet sınırlayıcı kutu tahmin edilmiş olur. Bu sınırlayıcı kutular nonmaximum supression yöntemiyle filtrelenir. Her bir ölçek için hücre başına, 3 adet öncül kutu tanımlanmış, toplamda hücre başına 9 adet öncül kutu tanımlanmıştır. Her bir ölçekteki sınırlayıcı kutu koordinatları kendi ölçeğinde tanımlanmış öncül kutu temel alınarak kodlanır. Sınırlayıcı kutuların gerçek genişlik ve yükseklik değerlerini tespit edebilmek için, YOLO v3 modelinde ön tanımlı öncül kutulara olan offset değerleri temel alınır.

2.5.2.4. YOLO v4 Nesne Tespit Modeli

ESA modellerinin doğruluğunun ve hızının arttırılması üzerine her geçen gün birçok yeni çalışma yayınlanmaktadır. YOLO v4 modelinin doğruluk oranını ve hızını artıran güncel çalışmalar modele entegre edilip, modelin başarısı ve hızı arttırılmıştır. YOLO v4 modeli, omurga olarak CSPDarknet53, boyun kısmında SPP ve PAN, kafa kısmında ise YOLO v3 modelinden oluşur. Eğitim aşaması için, omurga kısmında CutMix ve Mosaic veri arttırma teknikleri, DropBlock regülasyonu, Sınıf etiketi düzleştirme, Mish aktivasyon fonksiyonu, CSP (Cross-stage partial connections) , MiWRC (Multiinput weighted residual connections) entegre edilmiştir [47].

Tespit aşaması için; CIoU-loss, CmBN, DropBlock regülasyonu, Mosaic veri arttırma, Self-Adversarial eğitim, Eliminate grid sensitivity, Cosine annealing scheduler, rastgele eğitim şekilleri, Mish aktivasyonu, SPP-block, SAM-block, PAN path-aggregation block, DIoU-NMS entegre edilmiştir [47].

Yüksek doğruluk oranı ve gerçek zamanlı görüntü işleme için gerekli olan yüksek Fps(frames per second) ihtiyacını karşılamasından dolayı bu tez çalışmasında YOLO v4 modeli tercih edilmiştir.

2.5.3. Kullanılan İHA Özellikleri

Bu tez çalışmasında İHA olarak, Çizelge 2.1'de özellikleri verilen DJI Firmasının Mavic Air [48] modeli tercih edilmiştir. Bu model 1/2.3" CMOS sensörlü 12 megapixel kamerasıyla saniyede 30 kare HD (1280x720) görüntü aktarabilme özelliğine sahiptir. Bu da gerçek zamanlı görüntü işleme için ihtiyaç duyulan video kare sayısını karşılar. Kameranın bağlı olduğu 3 eksenli gimbali sayesinde, çeşitli açılardan görüntü alınabilir ve rüzgâr gibi etmenlerden İHA etkilense bile gimbali hareket ederek kendini dengeler ve kameranın titremesi engellenmiş olur. Böylece daha net görüntüler alınabilmesi mümkündür.

Çizelge 2.1. DJI Mavic Air Özellikleri

Kalkış Ağırlığı	430g
Ölçüler	168x84x64mm
Max Uçuş Süresi	21 dakika
Max Uçuş Mesafesi	10 km
Çalışma Aralığı	0-40 derece
GNSS	GPS+GLONASS
Çalışma Frekansı	2.400-2.4835 Ghz 5.725-5.850 Ghz
Batarya	2375 mAH 3s Lipo Batarya
Photo Format	JPEG/DNG(RAW)
Video Format	MP4/MOV(H.264/MPEG-4 AVC)
Gecikme Süresi	250 milisaniye

DJI firmasının yazılımcılar için geliştirdiği, firmanın görece yeni modellerinde kullanılabilen SDK(Software Development Kit) kütüphanesi, hava aracından telemetri bilgilerinin ve kamera görüntülerinin alınmasını sağlayan ve yer kontrol istasyonundan verilen komutların da hava aracına iletimini sağlayan fonksiyonlar içerir. DJI SDK, Mavic Air sonrası modeller için radyo bağlantısı ve wifi modunu desteklerken, Mavic Air modelinde yalnızca wifi modunu destekler. Bu çalışmada hava aracı ile yer birimi arasında bu nedenle wifi bağlantısı kurulmuştur. DJI SDK; Windows SDK, Mobile SDK, OnBoard SDK, PayLoad SDK, UX SDK şeklinde farklı ihtiyaç ve platforma göre ayrı ayrı paketlenmiştir. Bu çalışmada İHA ile etkileşim için DJI Windows SDK Beta 0.3.2 versiyonu kullanılmıştır.

2.5.4. Bilgisayar Donanım Özellikleri

Model görüntü üzerinde nesne tanımlama işlemleri gerçekleştirirken yüksek performanslı donanıma ihtiyaç duyar. Nesne tanıma için kullanılan YOLO v4 modeli, Cuda destekli ekran kartları ile yüksek performans sağladığı için, Cuda destekli Pascal mimarisindeki NVIDIA QUADRO P5000 ekran kartlı donanım üzerinde çalışmalar gerçekleştirilmiştir. Çizelge 2.2'de bu çalışmada kullanılan bilgisayar donanım özellikleri belirtilmiştir.

Çizelge 2.2. Donanım Özellikleri

İşletim Sistemi	Windows 10 Enterprise 64 bit (10.0,Build 19042)
İşlemci	Intel(R) Xeon(R) CPU E5-2687W v3 @ 3.10GHz 3.10 GHz (2 processors)
RAM	64 GB
Ekran Kartı	NVIDIA QUADRO P5000
Wifi Bağlantısı	Tp-Link TL-WN822N(300 Mbps)

2.5.5. Veri Seti

Veri seti olarak insan ve koyun sınıfına ait görüntüler incelenmiştir. Görüntüsü alınmak istenen sınıfların İHA aracılığıyla havadan görüntüsü video şeklinde alınmıştır. Alınan bu video dosyaları, Adobe Premier Pro [49] aracılığıyla adım adım incelenmiş ve video içinden eğitim setinde kullanılacak görüntüler jpeg formatında dışa aktarılmıştır. Alınan görüntülerdeki sınıfların sağlıklı bir şekilde işaretlenebilmesi için sınıfların görüntüsünün bütün olarak görüntü içerisinde

yer almasına, yarım eksik olmamasına, birçok farklı yerde, farklı yüksekliklerde ve farklı açılardan görüntü alınmasına gayret edilmiştir.

Görüntülerin içindeki sınıfları işaretlemek için, YOLO v4 modeli tasarımcılarından Alex Bachoskiy tarafından yazılan açık kaynak kodlu yoloMark [50] isimli uygulamadan faydalanılmıştır. Görüntü içindeki sınıflar kutulara alınarak tek tek sınıf etiketiyle beraber, işaretlenmiştir. Şekil 2.59'da işaretleme yapılmış bir görüntü örneği yer almaktadır.

Şekil 2.59. Yolo Mark ile İşaretlenen Örnek Bir Görüntü

2.5.6. Platform Seçimi

Uygulama Microsoft Windows 10 x64 ortamında geliştirilmiştir. Uygulama Geliştirme Platformu olarak Microsoft Visual Studio 2019 Community Edition kullanılmıştır. Yazılım geliştirme için kullanılması gereken DJI Windows SDK Kütüphanesi, Microsoft UWP(Universal Windows Platform) mimarisine uyumlu yazılmıştır. Bu nedenle programlama dili olarak, DJI Windows SDK uyumu ve güçlü nesneye yönelik bir dil olmasından dolayı, C# dili tercih edilmiştir.

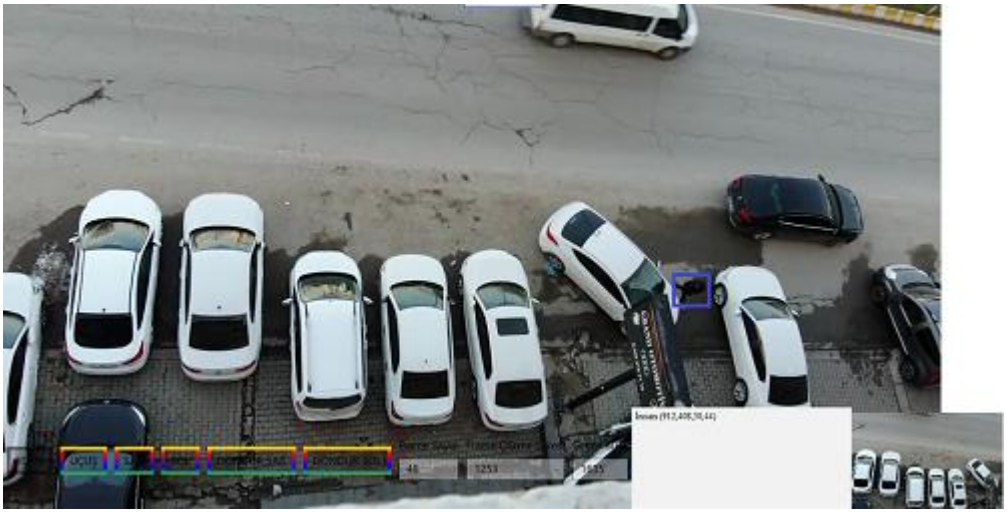
Nesne tespiti için, omurga olarak C dili ile yazılmış Darknet Framework kullanan, Alexey Bochkovskiy ve arkadaşları tarafından oluşturulan açık kaynak kodlu YOLO v4 [51] modeli kullanılmıştır. YOLO v4 modeli, görüntü işleme için birçok hazır fonksiyon bulunduran OPENCV [52] kütüphanesi ve nesne tespitini hızlandırmak için Nvidia tarafından oluşturulmuş Nvidia grafik işlemcilerinin bazı modelleriyle uyumlu CUDA mimarisi ve kütüphanelerinden faydalanılarak yazılmıştır. Cuda mimarisi olarak, Windows x64 uyumlu Cuda Toolkit 10.2 versiyonu [53] seçilmiştir. Cuda mimarisinde derin sinir ağları için oluşturulmuş cuDNN kütüphanesinin Cuda Toolkit 10.2 versiyonu ile uyumlu olan 7.6.5 versiyonu ve OpenCV Kütüphanesi olarak 4.1.0 versiyonu seçilmiştir. YOLO v4 modeli, Windows x64 ortamında, Visual Studio 2019 ile mevcut

donanımımıza uygun ayarlar tercih edilerek derlenmiş ve uygulamamızda ihtiyaç duyulan "yolo.dll" dosyası ve eğitim için kullanılan "darknet.exe" dosyası oluşturulmuştur. Oluşturulan dll dosyası, Windows platformunda unmanaged kod olarak kabul edilen C dili ile yazılmış olduğundan, UWP mimarisinde çalışacak uygulamamızda kullanabilmek için, bu dll dosyalarında dışa aktarılmış fonksiyonların bir sarmalayıcı proje(Wrapper) aracılığıyla çağrılması gerekmektedir. Uygulamamızda "yolo.dll" kütüphanesini sarmalayıcı proje olarak, açık kaynak kodlu AlturosYolo [54] projesi kullanılmıştır.

2.5.7. Uygulama

Bu çalışmada kullanılan İHA kamerası üzerinden video görüntüsü h264 video formatında saniyede 30 kare olmak üzere iletilir. Uygulamamızda, kullandığımız DJI Windows SDK tarafından alınan video görüntüsü, "DJIVideoParser.dll" aracılığıyla decode edilir. Uygulamamızda ihtiyaç duyduğumuz video görüntü karelerine, SDK tarafından çağrılan bir callBack fonksiyonu aracılığı ile RGBA (Red-Green-Blue-Alpha) renk uzayında 1280x720 ölçülerinde raw formatında erişilir. "yolo.dll' içerisinde yeralan fonksiyonları çağırabilmek için Wrapper proje olan AlturosYolo aracılığıyla görüntü, sinir ağına iletilir. Sinir ağı tarafından bulunan sınıflara ait güven skoru ve bunlara ait sınırlayıcı kutu genişlik, yükseklik, x ve y değerleri uygulamamız tarafından, güven eşik değerine göre filtrelenerek ekranda gösterilir.

İHA aracından video görüntülerinin alınmasının yanısıra, İHA'nın uçuş kontrolü de klavye aracılığıyla verilen komutlarla geliştirilen uygulama tarafından gerçekleştirilmektedir. Şekil 2.60'da geliştirilen uygulamanın örnek bir ekran görüntüsü verilmiştir.



Şekil 2.60. Uygulama Ekran Görüntüsü

3. BULGULAR VE TARTIŞMA

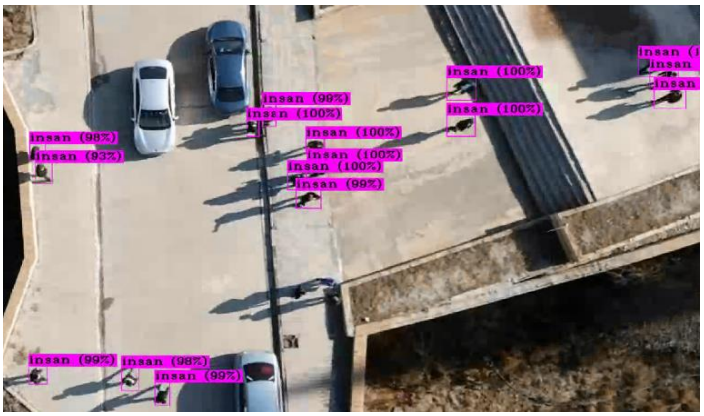
Bu tez çalışmasında İHA ile YOLO v4 modeli kullanılarak gerçek zamanlı nesne tespiti yapılmıştır. Modelin eğitimi amacıyla İHA tarafından alınan görüntülerle bir veri seti oluşturulup, YOLO v4 modeli bu eğitim seti ile eğitilmiştir.

Modelin doğruluk oranı başarısı, 5-Fold yaklaşımı ile test edilmiştir. Veri seti, her grupta aynı sayıda sınıf (insan ve koyun) içerecek şekilde 5 farklı alt kümeye bölünmüş. Bu 5 farklı grubun 4'lü kombinasyonları alınarak, 5 farklı eğitim seti oluşturulmuş ve her bir eğitim setiyle model, Darknet Framework ile ayrı ayrı eğitilmiştir. Gruba dâhil olmayan 5. Grup, modelin başarısını test amacıyla kullanılmıştır. Modelin eğitimi, 6000 iterasyon olacak şekilde seçilmiştir. Başarı oranları Çizelge 3.1'de gösterilmiştir.

Çizelge 3.1. 5-Fold Testleri

Test Grupları	İnsan Başarı Oranı	Koyun Başarı Oranı
Grup1	85.69%	93.01%
Grup2	98.10%	72.90%
Grup3	94.05%	82.62%
Grup4	93.92%	96.67%
Grup5	91.35%	97.06%
Ortalama	**92,62%**	**88.45%**

Oluşturulan veri setinin tamamı kullanılarak model eğitilmiş, eğitim yaklaşık 8 buçuk saat sürmüştür ve nihayi olarak model tarafından ağırlık dosyası oluşturulmuştur. Bu ağırlık dosyası ve YOLOv4 modelinin ayar dosyası, komut satırı uygulamasına(darknet.exe) parametre olarak aktarılarak, mevcut İHA ile çekilen 1280x720 ölçeğindeki video dosyalarını nesne tanıma işleme hızı test edilmiştir. Mevcut ağırlık dosyası ile kullanılan donanım üzerinde 31 Fps ortalama hız ile video görüntülerinin işlendiği gözlenmiştir. Şekil 3.1. Komut satırından yapılan teste dair örnek bir kare yer almaktadır.



Şekil 3.1. Komut satırından yapılan teste dair örnek bir kare

Görüntünün elde edilmesinden, tanımlanan nesnelerin sınırlayıcı kutularının ekranda gösterilmesi aşamasına kadar olan arka plan işlemlerinden ötürü, yapılan testlerde uygulama tarafında nesne tespit işleminin ortalama 20 Fps hızında olduğu ölçülmüştür.

Eğitim seti, daha az sayıda katman ve nöron içeren Yolov4 Tiny modeliyle eğitilmiş ve modelin doğruluk oranı, 5-Fold yaklaşımı ile test edilmiştir. Her bir grubun eğitimi yaklaşık 1 saat sürmüştür. Modelin eğitimi, 6000 iterasyon olacak şekilde seçilmiştir. Başarı oranları Çizelge 3.2'de gösterilmiştir.

Çizelge 3.2. Tiny 5-Fold Testi

Test Grupları	İnsan Başarı Oranı	Koyun Başarı Oranı
Grup1	78.83%	85.57%
Grup2	95.08%	71.82%
Grup3	91.11%	85.88%
Grup4	87.48%	96.67%
Grup5	79.70%	81.53%
Ortalama	**86.44%**	**84,29%**

Tiny modeliyle eğitilen veri setine ait ağırlık dosyası ve YOLOv4 Tiny modelinin ayar dosyası, komut satırı uygulamasına(darknet.exe) parametre olarak aktarılarak, mevcut İHA ile çekilen video dosyalarını nesne tanıma işleme hızı test edilmiştir. Mevcut ağırlık dosyası ile kullanılan donanım üzerinde 84 Fps ortalama hız ile video görüntülerinin işlendiği gözlenmiştir. Görüntünün elde edilmesinden, tanımlanan nesnelerin sınırlayıcı kutularının ekranda gösterilmesi aşamasına kadar olan arka plan işlemlerinden ötürü, yapılan testlerde uygulama tarafında nesne tespit işleminin ortalama 30 Fps hızında olduğu ölçülmüştür.

Bu tez çalışmasında geliştirilen uygulama CPU üzerinde de çalışabilmektedir. Ancak CPU üzerinde görüntünün işlenmesi uzun sürdüğü, gerçek zamandan çok uzak bir sonuç elde edildiği için işlenen video kareleri arasında geçiş süreleri uzundur. Bu nedenle değerlendirilmemiştir.

Literatürde İHA ile gerçek zamanlı nesne tanıma üzerine, doğrudan hava aracı üzerinde nesne tanıma ve uzaktaki güçlü bir GPU'ya sahip bir bilgisayar aracılığıyla nesne tanıma olmak üzere 2 farklı yaklaşım benimsenmiştir. Her 2 yöntemin de avantajları ve dezavantajları mevcuttur.

Tijtgat ve arkadaşları [55] tarafından hava aracı üzerinde NVIDIA Jetson TX2 kartı kullanarak gerçek zamanlı görüntü işleme üzerine yapılan çalışmada, YOLO v2 modeliyle (416x256) ölçülerinde %25 doğrulukla, 17.5 Fps oranı yakalanmıştır.

Lee ve arkadaşları [56] tarafından yapılan çalışmada, 1 GHz ARM Cortex-A8P işlemci üzerinde çalışan gömülü Linux işletim sistemine sahip ARDrone 2.0 İHA kullanarak, onboard ve uzaktaki bilgisayar üzerinde nesne tanıma şeklinde hibrit bir yaklaşım benimsenmiştir. Bu yaklaşımda, İHA üzerindeki kamerayla elde edilen görüntünün içerisinde nesne olabilirliği onboard işlemci üzerinde koşan uygulama ile irdelenir ve şayet görüntünün nesne içerebileceğine dair yüksek bir değer elde edilirse, görüntü uzaktaki bilgisayara wifi ile aktarılır ve R-CNN modeliyle görüntü nesne tanıma amacıyla incelenir.

Tang ve arkadaşları tarafından İHA ile araç tanıma üzerine yapılan çalışmada [57], YOLO v2 modeli kullanılarak NVIDIA GTX-1060 GPU ekran kartlı donanım üzerinde görüntü başına 48 ms işleme süresi ile 77.12% başarım oranı elde edilmiştir.

Wang ve arkadaşları tarafından yapılan çalışmada [58], 2 adet Intel 10-core Xeon CPU E5-2630 işlemcili NVIDIA Tesla K80 ekran kartlı donanım üzerinde RetinaNet(ResNet-50) modeliyle 24.45 Fps hızı yakalanmıştır. Bu çalışmada kullanılan İHA, gecikme süresi ve İHA ile iletişim için kullanılan uygulamadan bahsedilmemektedir. Bununla beraber Fps ve doğruluk testinin aynı tabloda verilmesi, İHA tarafından kaydedilen görüntülerin, sonrasında bilgisayar üzerinde işlenmesiyle yapılan testlere dayandığı düşünülmektedir. Yukarıda bahsedilen çalışmaların ve bu tez çalışmasının karşılaştırmalı tablosu Çizelge 3.3'te verilmiştir.

Çizelge 3.3. İHA ile Görüntü İşleme Çalışmalarının Karşılaştırılması

Makale	Görüntü İşleme Birimi	Donanım	Model	Veri Seti	Çözünürlük	Fps	Başarı (%)
Embedded Real-Time Object Detection for a UAV Warning System [55]	İHA	Nvidia Jetson Tx2	TinyYOLO	UAV123	416x256	17.5	35.6
Fast Vehicle Detection in UAV Images [57]	Bilgisayar	Intel Core i7, NVIDIA GTX-1060 GPU,16 GB RAM, Ubuntu 14.04	YOLOv2	Orjinal	640x480	20.8	77.12
Fast and Accurate, Convolutional Neural Network Based Approach For Object Detection From UAV [58]	Bilgisayar	2 adet Intel 10-core Xeon CPU E5-2630, NVIDIA Tesla K80 GPU	RetinaNet (ResNet-50)	Stanford Drone Dataset	Belirtilmemiş	24.4	85.17
“	“	“	RetinaNet (ResNet-101)	“	Belirtilmemiş	22.3	86.58
Bu çalışma	**Bilgisayar**	**Intel(R) Xeon(R) CPU E5-2687W v3 @3.10 GHz (2 processors), NVIDIA QUADRO P5000, 64 GB RAM, Windows 10**	**YOLO v4**	**Orjinal**	**416x416**	**31 (Darknet.exe)** **20 (Geliştirilen Uygulama)**	**92.62**
Bu çalışma	Bilgisayar	Intel(R) Xeon(R) CPU E5-2687W v3 @3.10 GHz (2 processors), NVIDIA QUADRO P5000, 64 GB RAM, Windows 10	Yolov4 Tiny	Orjinal	416x416	84 (Darknet.exe) 30 (Geliştirilen uygulama)	86.44

Bu tez çalışmasında İHA kontrolü, görüntünün elde edilmesi ve nesne tanıma işlemleri yüksek donanımlı tek bir bilgisayar üzerinde yapıldığından, nesne tanıma işleminin hızı bilgisayar donanımı gücüyle doğru orantılıdır. Bilgisayarın donanım özellikleri artırılarak Fps oranı yükseltilebilir. DJI Windows SDK kütüphanesinin yanlızca UWP mimarisini destekliyor olması, uygulamanın C dili ile yazılmasını mümkün kılmamamaktadır. Diğer İHA marka ve modelleri incelenerek, doğrudan C dili ile kullanılabilecek SDK’ya sahip bir İHA modelinde Darknet Framework doğrudan kullanılabileceği için yüksek Fps oranı yakalanabileceği öngörülmüştür. Mevcut İHA marka ve modellerinin haricinde, gerçek zamanlı nesne tespiti amacıyla gerekli donanım bileşenleri ve yazılımlar seçilerek veya kodlanarak özel bir İHA tasarlanarak yüksek bir Fps oranı yakalanabilir.

4. SONUÇ

Bu tez çalışmasında İHA kamerası aracılığıyla elde edilen görüntüler üzerinde gerçek zamanlı nesne tanıma amaçlanmıştır. İHA ile YKİ arasındaki haberleşmede iletilen telemetri ve video verilerinin güvenli bir şekilde iletilmesi hedeflenmiştir.

Çalışmada İHA olarak kullanılan DJI Mavic Air modelinin kamerasından alınan görüntülerin bilgisayar ortamına aktarılabilmesi için, İHA ve bilgisayar arasında wifi bağlantısı kullanılmıştır. Bilgisayar ile İHA arasında telemetri ve video görüntülerinin aktarılması yazılım tarafında DJI Windows SDK kütüphanesinin içerdiği fonksiyonlar aracılığıyla gerçekleştirilmiştir. Video görüntülerinin elde edilmesi ve İHA'nın uçuşunun bilgisayar tarafında yönetilebilmesi için, DJI Windows SDK kullanan C# dili ile UWP mimarisinde bir uygulama geliştirilmiştir. Elde edilen görüntüler üzerinde nesne tanıma yapabilmek amacıyla, yüksek doğruluk oranı ve gerçek zamanlı görüntü işleme için gerekli olan yüksek Fps ihtiyacını karşılamasından dolayı iskelet olarak açık kaynak kodlu Darknet Framework'u kullanan YOLO v4 modeli kullanılmıştır. Geliştirilen uygulama tarafında YOLO v4 modelini kullanabilmek için ihtiyaç duyulan "yolo.dll" dosyası ve eğitim amacıyla kullanılan "darknet.exe" dosyası, modelin kaynak kodlarının Visual Studio ortamında donanıma uygun ayarlarla derlenmesi ile elde edilmiştir. Gerçek zamanlı görüntü işleme yüksek işlem gücü gerektirdiğinden, uygulama NVIDIA QUADRO P5000 ekran kartlı bilgisayar tarafında gerçekleştirilmiştir.

İHA tarafından iletilen görüntülerde yüksek doğruluk oranıyla nesne tanıma yapabilmek için orijinal bir veri seti oluşturulmuştur. Bu veri seti; insan ve koyun sınıflarını ayırt edebilecek şekilde yoloMark uygulaması aracılığıyla işaretlenmiş, "Darknet.exe" aracılığıyla eğitilmiş ve eğitim sonucu oluşan ağırlık dosyası geliştirilen uygulama tarafında kullanılmıştır. İHA görüntülerinden elde edilen veri setiyle eğitim yapıldığından, uygulamada yüksek bir doğruluk oranı yakalanmıştır. Geliştirilen uygulama tarafından İHA'nın uçuşu kontrol edilmiş ve İHA'dan alınan görüntüler üzerinde gerçek zamanlı nesne tanıma gerçekleştirilmiştir. Yapılan bu çalışma ile İHA ile yukarıdan insan tanıma konusunda %92.62 başarım oranı ile literatürdeki en iyi başarımlardan biri elde edilmiştir.

Gelecek çalışmalarda; bu tez çalışmasında başarılan gerçek zamanlı nesne tanımanın verdiği ortam algısı gücü, İHA'nın uçuşunun bilgisayar aracılığıyla gerçekleştirilebilmesinin verdiği kontrol edilebilirlik ve hava aracının mobilitesinden faydalanabilecek birçok uygulama geliştirilebilir. Örneğin yoğun insan gücü gerektiren zamanla yarışılan arama kurtarma faaliyetleri için yapılabilecek bir çalışma ile geliştirilebilecek bir uygulama aracılığıyla İHA, insan

müdahalesine gerek olmaksızın kayıp bir kişiyi ya da bir nesneyi arayabilir ve bulduğunda tespit ettiği noktayı YKİ'ye iletebilir. Öte yandan yüksek işlem gücüne sahip, paralel programlanmaya müsait ve İHA üzerinde taşınabilecek Nvidia Jetson benzeri kartlar üzerinde YOLO v4 modelinin uyarlanması çalışması gerçekleştirilebilir. Bu kartlar üzerinde yapılabilecek çalışmalar ile İHA'nın YKİ'den bağımsız, sağlıklı bir şekilde otonom uçuşunu sağlayan, aynı zamanda nesne tanıma işlemleri yapabilen uygulamalar geliştirilebilir. Tez için oluşturulan veri seti farklı yerlerde ve zamanlarda elde edilen görüntüler ile zenginleştirilerek, İHA ile görüntü işleme alanındaki güçlü ve güvenli veri seti eksikliği giderilebilir.

5. KAYNAKLAR

[1] Nvidia, 2020. Nvidia ai-data-science. https://www.nvidia.com/en-us/ai-data-science/ (Erişim Tarihi:15/11/2020).

[2] Intel, 2020. Intel artificial-intelligence. https://www.intel.com/content/www/us/en/artificial-intelligence/overview.html (Erişim Tarihi:15/11/2020).

[3] Can N, Kahveci M, 2017. İnsansız Hava Araçları Tarihçesi, Tanımı, Dünyada Ve Türkiye'deki Yasal Durumu. Selcuk University Journal of Engineering, Science and Technology, 5(4):511–535.

[4] McCulloch WS, Pitts W, 1943. A logical calculus of the ideas immanent in nervous activity. The Bulletin of Mathematical Biophysics, 5(4):115–133.

[5] Anonim, 2020. Alan Turing. https://tr.wikipedia.org/wiki/Alan_Turing (Erişim Tarihi:15/12/2020).

[6] Widrow B, Lehr MA, 1990. 30 years of adaptive neural networks: perceptron, Madaline, and backpropagation. Proceedings of the IEEE, 78(9):1415–1442.

[7] Keskenler MF, Keskenler EF, 2017. Geçmişten Günümüze Yapay Sinir Ağları ve Tarihçesi. Takvim-i Vekayi, 5(2):8–18.

[8] Hubel DH, Wiesel TN, 1959. Receptive fields of single neurones in the cat's striate cortex. The Journal of Physiology, 148(3):574–591.

[9] Fukushima K, 1980. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological Cybernetics, 36(4):193–202.

[10] Rumelhart DE, Hinton GE, Williams RJ, 1986. Learning representations by back-propagating errors. Nature, 323(6088):533–536.

[11] Lecun Y, Bottou L, Bengio Y, Haffner P, 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324.

[12] Anonim, Deep Blue. https://tr.wikipedia.org/wiki/Deep_Blue (Erişim Tarihi:15/12/2020).

[13] Oh KS, Jung K, 2004. GPU implementation of neural networks. Pattern Recognition, 37(6):1311–1314.

[14] Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A, 2015. Going deeper with convolutions. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 07-12 June 2015, Boston,

USA, s:1–9.

[15] Girshick R, Donahue J, Darrell T, Malik J, 2014. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. 2014 IEEE Conference on Computer Vision and Pattern Recognition, 23-28 June 2014, Columbus, s:580–587.

[16] Redmon J, Divvala S, Girshick R, Farhadi A, 2015. You only look once: Unified, real-time object detection. arXiv preprint arXiv:150602640.

[17] Budak Ü, Cömert Z, Çıbuk M, Şengür A, 2020. DCCMED-Net: Densely connected and concatenated multi Encoder-Decoder CNNs for retinal vessel extraction from fundus images. Medical Hypotheses, 134(August 2019):109426.

[18] Kök T, 2012. İnsansız Hava Araçlarının Güvenli Kullanımı için Spektrum İhtiyaçlarının Belirlenmesi ile İlgili Öneriler. Teknik Uzmanlık Tezi, Bilgi Teknolojileri ve İletişim Kurumu, İstanbul.

[19] Shgm, 2020. iha.shgm. https://iha.shgm.gov.tr/public/index (Erişim Tarihi:20/12/2020).

[20] Trt, 2020. TSK'nın lojistik ihtiyaçlarını 'Kargo İHA' karşılayacak. https://www.trthaber.com/haber/bilim-teknoloji/tsknin-lojistik-ihtiyaclarini-kargo-iha-karsilayacak-496846.html (Erişim Tarihi:18/12/2020).

[21] Kolukısa H, 2020. Aksungur İHA 49 saat havada kalarak rekor kırdı. https://www.defenceturk.net/aksungur-iha-yeni-rekorunu-gokyuzune-bayragimizi-cizerek-kutladi (Erişim Tarihi:12/12/2020).

[22] Çakıcı E, 2019. Mini İnsansız Hava Araçları İçin Bir Fırçasız Motor Test Sistemi Geliştirilmesi. Yüksek Lisans Tezi, Osmangazi Üniversitesi Fen Bilimleri Enstitüsü, Eskişehir.

[23] Karakuzu C, Kesler M, Ucar G, 2016. Dengeleme halkalari ile görüntü kararlılaştırması. Bap Sonuç Raporu, Şeyh Edebali Üniversitesi, Bilecik.

[24] Aktoğan A, 2011. Uzaktan Erişimli Sabit Mıknatıslı Doğru Akım Motor Kontrolü Deney Düzeneği. Yüksek Lisans Tezi, Karadeniz Teknik Üniversitesi Fen Bilimleri Enstitüsü, Trabzon.

[25] Kerimoğlu K, 2011. Sabit Kanatlı Bir İnsansız Hava Aracı İçin Düşük Bütçeli Bir Otopilot Sistemi Tasarımı. Yüksek Lisans Tezi, TOBB Ekonomi ve Teknoloji Üniversitesi Fen Bilimleri Enstitüsü, Ankara.

[26] O'Mahony N, Campbell S, Carvalho A, Harapanahalli S, Hernandez GV, Krpalkova L, Riordan D, Walsh J, 2020. Deep Learning vs. Traditional Computer Vision. Advances in Intelligent Systems and Computing, 943(Cv):128–144.

[27] Cantemir Z, 2019. Derin Öğrenme Yöntemleriyle Medikal Görüntü Sınıflandırma. Yüksek Lisans Tezi, Gazi Üniversitesi Fen Bilimleri Enstitüsü, Ankara.

[28] Mete BR, 2019. Derin Öğrenme ve Görüntü Sınıflandırma. Yüksek Lisans Tezi, İstanbul Üniversitesi Lisansüstü Eğitim Enstitüsü, İstanbul.

[29] Budak Ü, Cömert Z, Rashid ZN, Şengür A, Çıbuk M, 2019. Computer-aided diagnosis system combining FCN and Bi-LSTM model for efficient breast cancer detection from histopathological images. Applied Soft Computing Journal, 85:105765.

[30] Misra D, 2019. Mish: A self regularized non-monotonic neural activation function. arXiv preprint arXiv:1908.08681v3 .

[31] Heaton J, 2018. Ian Goodfellow, Yoshua Bengio, and Aaron Courville: Deep learning. Genetic Programming and Evolvable Machines, 19(1–2):305–307.

[32] Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R, 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research, 15(2014):1929–1958.

[33] Ghiasi G, Lin TY, Le Q V., 2018. Dropblock: A regularization method for convolutional networks. Advances in Neural Information Processing Systems, 2018–Decem:10727–10737.

[34] Yun S, Han D, Chun S, Oh SJ, Choe J, Yoo Y, 2019. CutMix: Regularization strategy to train strong classifiers with localizable features. Proceedings of the IEEE International Conference on Computer Vision, 2019–Octob:6022–6031.

[35] Rugery P, 2020. Explanation of YOLO v4 a one stage detector. https://becominghuman.ai/explaining-yolov4-a-one-stage-detector-cdac0826cbd7 (Erişim Tarihi:06/01/2021).

[36] Hui J, 2020. YOLO v4. https://jonathan-hui.medium.com/yolov4-c9901eaa8e61 (Erişim Tarihi:06/01/2021).

[37] He K, Zhang X, Ren S, Sun J, 2015. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. arXiv preprint arXiv:14064729v4.

[38] Zheng Z, Wang P, Liu W, Li J, Ye R, Ren D, 2019. Distance-IoU loss: Faster and better learning for bounding box regression. arXiv preprint arXiv:191108287.

[39] Krizhevsky BA, Sutskever I, Hinton GE, 2012. ImageNet Classification with Deep Convolutional Neural Networks. Communications of the ACM, 60(6):84–90.

[40] Bektaş M, 2020. Tensorflow Ortamında Nesne Tanıma Uygulaması (Defect Type Detection using Faster R-CNN). https://medium.com/@mbektas/tensorflow-ortamında-nesne-

tanıma-uygulaması-defect-type-detection-using-faster-r-cnn-1be1e1f59c8c (Erişim Tarihi:06/01/2021).

[41] Girshick R, 2015. Fast R-CNN. 2015 IEEE International Conference on Computer Vision (ICCV), 7-13 Dec 2015, Santiago, s:1440–1448.

[42] Ren S, He K, Girshick R, Sun J, 2017. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(6):1137–1149.

[43] Redmon J, Divvala S, Girshick R, Farhadi A, 2016. You Only Look Once: Unified, Real-Time Object Detection. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 27-30 June 2016, Las Vegas, 779-788.

[44] Redmon J, 2020. Darknet. https://pjreddie.com/darknet/ (Erişim Tarihi:10/11/2020).

[45] Redmon J, Farhadi A, 2017. YOLO9000: Better, faster, stronger. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 21-26 July 2017, Honolulu, s:6517-6525.

[46] Redmon J, Farhadi A, 2018. YOLOv3: An Incremental Improvement. Tech report, arXiv preprint arXiv:1804.02767 .

[47] Bochkovskiy A, Wang CY, Liao HYM, 2020. YOLOv4: Optimal Speed and Accuracy of Object Detection. arXiv preprint arXiv:2004.10934 .

[48] DJI, 2020. Mavic Air. https://www.dji.com/mavic-air (Erişim Tarihi:11/11/2020).

[49] Adobe, 2020. Adobe Premiere Pro. https://www.adobe.com/tr/products/premiere.html (Erişim Tarihi:10/11/2020).

[50] Bachoskiy A, 2020. yoloMark. https://github.com/AlexeyAB/Yolo_mark (Erişim Tarihi:10/10/2020).

[51] Bochkovskiy A, 2020. YOLO v4. https://github.com/AlexeyAB/darknet (Erişim Tarihi:11/11/2020).

[52] OpenCv, 2020. OpenCV. https://opencv.org/ (Erişim Tarihi:14/11/2020).

[53] Nvidia, 2020. Cuda Toolkit. https://developer.nvidia.com/cuda-toolkit (Erişim Tarihi:14/12/2020).

[54] Bricman M, 2020. AlturosYolo. https://github.com/AlturosDestinations/Alturos.Yolo (Erişim Tarihi:13/12/2020).

[55] Tijtgat N, Van Ranst W, Volckaert B, Goedemé T, De Turck F, 2017. Embedded real-time object detection for a UAV warning system. Proceedings - 2017 IEEE International Conference on Computer Vision Workshops, ICCVW 2017, 2018–Janua:2110–2118.

[56] Lee J, Wang J, Crandall D, Sabanovic S, Fox G, Real-time, cloud-based object detection for unmanned aerial vehicles. Proceedings - 2017 1st IEEE International Conference on Robotic Computing, IRC 2017, :36–43.

[57] Tang T, Deng Z, Zhou S, Lei L, Zou H, 2017. Fast vehicle detection in UAV images. RSIP 2017 - International Workshop on Remote Sensing with Intelligent Processing, Proceedings, 2017–June:16–20.

[58] Wang X, Cheng P, Liu X, Uzochukwu B, 2018. Fast and accurate, convolutional neural network based approach for object detection from UAV. arXiv preprint arXiv:1808.05756.

ÖZGEÇMİŞ

[redacted] yılında [redacted] Merkez'de dünyaya geldim. İlköğretim ve lise eğitimimi [redacted]'ta tamamladım. [redacted] Üniversitesi Bilgisayar Mühendisliği Bölümü'nden [redacted] yılında mezun oldum. [redacted] yılları arasında [redacted] Bilgi Teknolojileri Dairesi'nde çalıştım. Daha sonra [redacted]'nde çalışmaya başladım ve halen orda devam etmekteyim.

Gökhan KESKİNTAŞ