İNSANSIZ HAVA ARACINDAN ÇEKİLEN VİDEOLAR KULLANILARAK DERİN ÖĞRENME YAKLAŞIMI İLE NESNE TESPİTİ

AYŞAN USTA

AĞUSTOS 2022

DİYARBAKIR

T.C.
DİCLE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

İNSANSIZ HAVA ARACINDAN ÇEKİLEN VİDEOLAR KULLANILARAK DERİN ÖĞRENME YAKLAŞIMI İLE NESNE TESPİTİ

AYŞAN USTA

DİCLE ÜNİVERSİTESİ LİSANSÜSTÜ EĞİTİM-ÖĞRETİM VE SINAV YÖNETMELİĞİNİN BİR PARÇASI OLARAK ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ ANA BİLİM DALI YÜKSEK LİSANS TEZİ OLARAK HAZIRLANMIŞTIR

AĞUSTOS 2022

DİYARBAKIR

İNSANSIZ HAVA ARACINDAN ÇEKİLEN VİDEOLAR KULLANILARAK DERİN ÖĞRENME YAKLAŞIMI İLE NESNE TESPİTİ

Ayşan USTA tarafından Dicle Üniversitesi Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliği'nin bir parçası olarak hazırlanan bu çalışma, aşağıda bilgileri yazılı jüri üyeleri tarafından değerlendirilerek **Elektrik Elektronik Mühendisliği Ana Bilim Dalı**'nda **Yüksek Lisans Tezi** olarak kabul edilmiştir.

Prof. Dr. Neslihan Dalkılıç _____
Müdür, **Fen Bilimleri Enstitüsü**

Dr.Öğr. Üyesi M. Ali ARSERİM …. _____
Danışman,
Elektrik Elektronik Mühendisliği Bölümü,
Dicle Üniversitesi

Sınav Jürisi:

Dr.Öğr. Üyesi Emrullah ACAR **(*)** _____
Elektrik Elektronik Mühendisliği Bölümü,
Batman Üniversitesi

Dr.Öğr. Üyesi M. Ali ARSERİM **(**)** _____
Elektrik Elektronik Mühendisliği Bölümü,
Dicle Üniversitesi

Dr.Öğr. Üyesi Hüseyin ACAR _____
Elektrik Elektronik Mühendisliği Bölümü,
Dicle Üniversitesi

ONAY

Savunma Tarihi: 31 /08/ 2022

(*) Sınav Jürisi kısmının birinci satırına Jüri Başkanının bilgilerini yazınız.
(**) Sınav Jürisi kısmının ikinci satırına Tez Danışmanının bilgilerini yazınız.

Anneme ve Babama ...

Dicle Üniversitesi Fen Bilimleri Enstitüsü Tez Yazım Kurallarına uygun olarak hazırlanan bu tez çalışmasında yer alan tüm bilgilerin akademik kurallara ve etik ilkelere uygun olarak elde edildiğini ve sunulduğunu beyan ederim. Ayrıca, bahse konu bu kural ve ilkelerin gerektirdiği üzere, bu çalışmada özgün olmayan tüm bilimsel içerikleri kurallara uygun biçimde alıntılayıp kaynak gösterdiğimi beyan ederim. Beyanımla çelişen herhangi bir delil bulunduğu takdirde tüm sorumluluğu üstleneceğimi kabul ederim.

Ad, Soyad: Ayşan USTA

İmza: …….

TEŞEKKÜR

Bu yoğun ve heyecanlı sürecin her aşamasında bana destek olan Aileme ve İş arkaşlarıma, çalışmamın farklı aşamalarında yardımlarını esirgemeyen eşim Doğaç Usta'ya, babam İsmail Usta'ya, patronum Umut Ünal'a, kardeşlerim Berivan Çetin ve Elif Nur Usta'ya, iş arkadaşım Samet Çetin ve Uğurcan Usta'ya, üzerimde emeği olan çok sevdiğim öğretmenlerim Prof. Dr Ayşegül Uçar'a, Prof. Dr. Mehmet Siraç Özerdem'e ve ismini sayamadığım diğer tüm öğretmenlerime, özellikle yüksek lisans programına başlamam için teşvik eden, yol gösteren, cesaret veren Danışmanım Dr. Öğr. Üyesi M. Ali Arserim'e çalışmam süresince verdiği destek ve gösterdiği sabırdan dolayı sonsuz teşekkürler. Tezin daha nitelikli bir hale gelmesine katkı sağlayan Jüri üyelerine yaptıkları eleştiri ve verdikleri öneriler için ayrıca teşekkür ederim.

En önemlisi, bana ve kardeşlerime daima inanan ve güvenen, bizi bugünlere getiren, her durumda yanımızda olan, sevgi ve merhamet dolu annem İlknur Çetin'e, babam Remzi Çetin'e ve abim Fırat Çetin'e sonsuz teşekkürler…

İÇİNDEKİLER

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

SİMGELER ve KISALTMALAR LİSTESİ

Simge	Açıklama
s	: saniye
ms	: mili saniye
%	: yüzde

Kısaltma	Açıklama
İHA	: İnsansız Hava Aracı
YZ	: Yapay Zeka
YSA	: Yapay Sinir Ağları
ESA	: Evrişimsel Sinir Ağları
YOLO	: You Only Look Once
DÖ	: Derin Öğrenme
GPU	: Graphics Processing Unit
B-ESA	: Bölgesel Evrişimsel Sinir Ağları
MÖ	: Makine Öğrenmesi
FPS	: Frame Per Second
TBK	: Tam Bağlı Katman
KCF	: Kernelized Correlation Filter
EPCC	: Extended Polyhedral Classifier
ReLU	: Rectified Linear Unit
SGD	: Stochastic Gradient Descent
AP	: Average Precision
mAP	: mean Average Precision
IoU	: Intersection over Union

ÖZET

İNSANSIZ HAVA ARACINDAN ÇEKİLEN VİDEOLAR KULLANILARAK DERİN ÖĞRENME YAKLAŞIMI İLE NESNE TESPİTİ

Usta, Ayşan
Yüksek Lisans, Elektrik Elektronik Mühendisliği Bölümü
Danışman: Dr. Öğr. Üyesi M. Ali ARSERİM
Eylül 2022, 88 sayfa

Günümüzde İnsansız Hava Araçları (İHA) sınır güvenliği, sahil güvenliği, savunma, saldırı başta olmak üzere keşif, arama kurtarma, hava fotoğrafçılığı, lojistik, zirai ilaçlama, yangın söndürme gibi geniş bir kullanım alanına sahiptir. Bununla beraber otonom karar ve aksiyon mekanizması, komuta merkezi ile iletişim anında mesafeden veya başka sebeplerden dolayı oluşabilecek ve sonuçları tolere edilemeyecek gecikmenin yaşanmaması adına üzerinde durulması gereken önemli bir detaydır. Özellikle sınır güvenliği, sahil güvenliği, savunma, saldırı, yangın söndürme gibi durumlarda çok kritiktir. İHA'ların bazı görevleri otonom bir şekilde yerine getirebilmesi ise Bilgisayarlı Görü alanının İHA'lara entegresi ile olur. Bilgisayarlı Görü alanı uygulamalarından biri olan havadan nesne tespiti uygulamaları uzaklık, yakınlık kavramlarına bağlı olarak farklı boyutlardaki nesneleri algılayamama, yavaş algılama, yanlış tahminleme gibi birkaç hata içerir. Bu hatalar, farklı boyuttaki nesneleri, sabitleme kutularını (Anchor Box) kullanarak öğrenen ve maksimum olmayanı önleme (Non-Maximum Suppression) tekniğini kullanarak maksimum güven puanına sahip sınırlayıcı kutu dışındaki sınırlayıcı kutuları silen YOLO algoritması ile en aza indirilir. Çalışmanın kapsamı Derin Öğrenme (DÖ) yöntemlerinden YOLOv3 sinir ağı kullanılarak İHA'dan alınan görüntüdeki araçların ve yayaların tespitidir. Ve bu çalışmada, İHA'dan alınan video, sinir ağına verilmiş ve videodaki araçların ve yayaların tespiti sağlanmıştır. İHA olarak da DJI Mavic 2 Zoom Drone kullanılmıştır. Veri seti olarak VIRAT [50] video veri setinden alınan 500 görüntü eğitim için, Mavic 2 Zoom videolarından alınan 377 görüntü ise yeniden eğitim için kullanılmıştır. Ağ, 500 görüntü ile eğitilmiş, ortalama loss değeri 2.345 ve mAP değeri %79 olacak şekilde bir başarım elde edilmiştir. 377 görüntü ile ağ yeniden eğitilmiş, loss değeri 1'e yaklaşmıştır, bununla beraber mAP değeri önceki değere kıyasla %70.9 a düşmüştür. Eğitim ve test süreci Google Colab Tesla T4 GPU makinesinde gerçekleştirilmiştir. Modelin performansı Loss grafiği ve mAP grafiği ile değerlendirilmiştir.

Anahtar Kelimeler: İha, YoloV3, Darknet-53, Evrişimsel sinir ağları, Derin öğrenme

ABSTRACT

OBJECT DETECTION by DEEP LEARNING APPROACH USING VIDEOS TAKEN FROM UNMANNED AERIAL VEHICLE

Usta, Ayşan
Master of Science in Department of Electrical and Electronics Engineering Department
Supervisor: Dr. M. Ali ARSERİM
September 2022, 88 pages

Today, Unmanned Aerial Vehicles (UAV) have a wide range of uses such as border security, coast guard, defense, attack, reconnaissance, search and rescue, aerial photography, logistics, agricultural spraying, fire extinguishing. However the autonomous decision and action mechanism is a very important detail that should be emphasized, in order to avoid any intolerable delay that may occur during communication with the command center due to distance or other reasons. It is especially critical in situations such as border security, coast guard, defense, attack, fire extinguishing. The ability of UAVs to perform some tasks autonomously is can be possible by integrating the Computer Vision field with UAVs. Aerial object detection applications, which are one of the computer vision field applications, contain several errors such as not being able to detect objects of different sizes, slow detection, wrong estimation, depending on the concepts of distance and proximity. These errors are minimized by the YOLO algorithm, which learns objects of different sizes with using "Anchor Boxes" and delete bounding boxes other than the bounding box that with the maximum confidence score with using the "Non-Maximum Suppression" technique. The scope of the study is the detection of vehicles and pedestrians in the image taken from the UAV using the YOLOv3 neural network, one of the Deep Learning (DL) methods. And in this study, the video taken from the UAV was given to the neural network and the vehicles and pedestrians in the video were detected. DJI Mavic 2 Zoom Drone was used as the UAV. As dataset, 500 images taken from VIRAT [50] video datasets were used for training, and 377 images taken from the Mavic 2 Zoom videos were used for retraining the average loss value was 2,345 and the mAP value was 79%. The network was retrained with 377 images, the loss value approached 1, however, the mAP value decreased to 70.9% compared to the previous value. The training and testing process was carried out on the Google Colab Tesla T4 GPU machine. The performance of the model was also evaluated with the loss graph and mAP graph.

Keywords: Uav, YoloV3, Darknet-53, Convolutional neural networks, Deep learning

1. GİRİŞ

1.1 İnsansız Hava Araçları

İnsansız Hava Aracı (İHA), bir operatör tarafından uzaktan kumanda edilebilen veya bir uçuş rotası boyunca otomatik hareket edebilen, göreve ilişkin faydalı yük taşıyabilen ve bir takım görevi otonom olarak yerine getirebilen bir hava aracıdır [1]. Döner kanatlı, sabit kanatlı ve hibrit şeklinde 3 türü vardır. Döner kanatlı İHAlar, Şekil 1.1'de de görüldüğü gibi yerçekimi doğrultusunun zıttı yönde dönen ve aracı havada tutan pervane veya pervanelerden oluşup, sabit kanatlı İHAların aksine, gövdesi sabit, kanatlar yani pervaneler hareketli olduğu için havada asılı kalabilen, havadaki kontrolü rahat olan ve dar alanlarda iniş kalkış yapabilen araçlardır.

Şekil 1.1 Döner kanatlı insansız hava aracı [2]

Sabit kanatlı İHAlar adından da anlaşılacağı üzere sabit kanatlardan, havada kalabilmesi için gövdeyi hareket ettirecek içten yanmalı motordan veya elektrikli motordan oluşan, dolayısı ile havada asılı kalamayan bunun yanında iniş ve kalkış için geniş alanlara ihtiyaç duyan ancak uçuş menzili döner kanatlılara nazaran oldukça uzun olan hava taşıtıdır. Şekil 1.2'de sabit kanatlı bir İHA görülmektedir.

Şekil 1.2 Sabit kanatlı insansız hava aracı [3]

Hibrit İHAlar ise Şekil 1.3'deki gibi döner kanat ve sabit kanat İHAların bir kombinasyonu olup ikisinin de avantajlarından faydalanabilmek adına geliştirilmiştir.

Şekil 1.3 Hibrit insansız hava aracı [4]

1849 da Avusturya ordusunun Venedik kuşatması esnasında 200 tane yangın balonunu İHA ile taşıması tarihte ilk İHA kullanımı olarak bilinmektedir [5]. Bu alanda yapılan çalışmalar Birinci Dünya Savaşı döneminde hız kazanmıştır ve bu dönemin en önemli İHAsı belirlenen hedefe patlayıcı bırakabilen "Kettering Bug" olmuştur [6]. Belirlenen zamanda patlayan pilotsuz (insansız) hava torpidosu da bu dönemde geliştirilen bir diğer İHAlardan biridir [7] Ve yine ilk motorlu İHA denemesi Birinci Dünya Savaşı döneminde "Aerial Target" in uçurulması ile yapılmıştır [6]. Bu gelişmeler İkinci Dünya savaşında da artarak devam etmiştir. 1967 ile 1970 yılları arasında keşif kameralı ilk İHAlar İsrail tarafından Süveyş Kanalı ve çevresini görüntüleyerek kullanıldı. Böylece savaşta ilk defa kısa iniş pistlerini kullanabilen jet bazlı ve ağır olmayan İHA'lar kullanılmış oldu [8]. 1973 de yine İsrail tarafından gerçek zamanlı gözetleme yapan İHA geliştirildi [9].

İHAlar, son yıllarda birçok ülkenin hava kuvvetleri tarafından kullanılmaktadır ancak sivillere ait İHAların askeri İHAlardan sayıca fazla olduğu bilinmektedir [10]. İHAlar başlangıçta gözlem, saldırı ve savunma amaçlı üretilip kullanılsa da günümüzde gözetleme, keşif, taşımacılık, zirai ilaçlama, yangın söndürme, enerji nakil hatlarının bakım ve kontrolü, muhabirlik, fotoğraf ve video çekme gibi geniş bir kullanım yelpazesine sahiptir. Bu kullanım alanlarının bir çoğunda İHAnın otonom olması, yani bir insan gibi görüp, değerlendirip, karar verip, aksiyon alması beklenir. Özellikle de yangın söndürme, sınır güvenliği, sahil güvenliği gibi kritik durumlarda, mesafeden veya başka sebeplerden dolayı komuta merkezi ile iletişim anında oluşabilecek ve sonuçları tolere edilemeyecek gecikmenin yaşanmaması adına, otonom karar ve

aksiyon mekanizması, üzerinde durulması gereken çok önemli bir detaydır ve temelde İHAlar ile obje algılama, tanıma ve takibi işlemlerinin yapılabilmesi ile mümkündür.

1.2. İHA ile Obje Algılama, Tanıma ve Takibi

Bir cismi görmek, algılamaya çalışmak, ayırt edici yanlarını seçip tahminlerde bulunmak, cismi anlamlandırmak ve bu yolda yeni tahminler öne sürmek, destekleyici düşünceler geliştirmek, fikirler ortaya atmak ve sonuçta yanlış veya doğru bir karara varmak. Bir insanın obje algılama, tanıma ve tahminde bulunma süreci üç aşağı beş yukarı bu şekildedir. Peki bir makine bu sürecin ne kadarını bir insan gibi yapabilir? Yada aslında ne kadarını bir insanın erişemeyeceği bir başarımla, tam bir makine gibi kusursuz yapmasını isteriz?

Birçok alanda makinelerin olaylara, insanlar gibi sağduyu ile, önsezi ile, içgüdüsel bir şekilde empati kurarak yaklaşmasına, o yetiye erişebilmesine uğraşılabilir. Martı Jonathan Livingston'un öğrencisi Fletch'e verdiği öğüt "Sevgili Fletch! Gözlerinle gördüğüne inanma, gördüklerin yalnızca sınırlı olandır. Göz, her şeyin özünü göremez. Sezgilerinle bak. Öğrendiklerinin bilincine varmaya çalış [11]." elbette makinelere de verilebilir. Ancak bu çalışmanın konusu ve gece görüş cihazlarının insan gözünden daha hassas algılama yeteneğine sahip olması düşünülünce, makinenin, bahsi geçen insancıl özelliklerden uzak, bir insandan daha iyi tespit yapan, neredeyse hatasız bir sonuca ulaşan bir makine olması üzerine çalışılmıştır ve çalışılacaktır. Çünkü bahsi geçen çalışma nesne tanımaya yöneliktir ve nesnenin tespiti anında neredeyse hiçbir önsezi, içgüdü, sağduyu vs gerekmemektedir.

Obje algılama, tanıma ve takibi işlemleri oldukça zor bir süreçtir. Makine öğrenmesi (MÖ) yöntemlerinin en dikkat çekici uygulamalarını içeren, özellikle de görüntüler ile yapılan uygulamalarda, bir DÖ algoritması ve aynı zamanda bir YSA olan ESA'lar tercih edilir. ESAlar, 1990'lı yıllarda geliştirilmesine rağmen Grafik İşleme Üniteleri (Graphics Processing Unit - GPU) ile birlikte tekrar gündeme gelmiştir [12][13]. 2012 yılında ImageNet Büyük Ölçekli Görsel Tanıma Yarışmasında oldukça yüksek nesne sınıflandırma doğruluğu gösteren ESA'lar kullanılmıştır [14]. B-ESA'lar (Bölgesel Evrişimsel Sinir Ağları) ile ilk defa 2014 yılında nesne algılama için çalışılmış ve bu tarihten sonra nesne algılama için en ümit veren ağ yapılarından biri olmuştur [14].

Son yıllarda obje tespiti ve takibi konularında adından sıkça bahsettiren ve bu çalışmada kullanılan YOLO da ESA'ları kullanan bir algoritmadır. Süre, doğruluk ve hız bazında değerlendirilince de diğer ESA algoritmalarına karşı en çok tercih edilen algoritmadır.

YOLO "You Only Look Once", kendisine girdi olarak verilen resmi veya görüntüyü ilk önce SxS'lik (3x3 5x5 19x19 vs.) hücrelere böler. Resim hücrelere bölündükten sonra her hücre, kendi içerisinde nesne olup olmadığını, nesne var ise merkez noktasının kendi alanında olup olmadığını, merkez noktası kendi alanında ise nesnenin uzunluğunu, genişliğini ve hangi sınıftan olduğunu bulmakla sorumludur. Nesnenin merkez noktasına sahip olduğunu tespit eden her hücre o nesnenin sınıfını, yüksekliğini ve genişliğini bulup çevresine sınırlayıcı kutu (bounding box) çizer. Tüm bu işlemler ağır hesapsal yük dolayısı ile fazla zaman gerektirir. Zaman içinde bilgisayar ve bilgisayar donanımlarında meydana gelen büyük gelişmeler sayesinde YZ alanında büyük ölçekli araştırmalar, deneyler, tespitler yapılmış, çığır açıcı YSAlar oluşturulmuş, anlık tespitler yapan DÖ algoritmaları (modeller) geliştirilmiştir. Tüm bu gelişmeler ile yoğun hesaplama yükü, fazla zaman ihtiyacı ve diğer birçok problem bir anda çözüme kavuşmuştur. Hatta öyle ki gelişen modeller sayesinde önceden binlerce hatta yüzbinlerce veri seti ile elde edilen başarımlar şimdilerde sadece birkaç yüz veri seti ile elde edilebilecek duruma gelmiştir.
Bahsedilen zirai ilaçlama, yangın söndürme, sınır güvenliği gibi algılama, tespit ve takip gerektiren uygulamaların bir minyatürü olabilecek bu çalışmada, İHA ile havadan araç ve yaya tespiti yapılmıştır.

1.3 Benzer Çalışmalar

İHA ile DÖ temelli tespit ve tanıma yöntemleri üzerine yapılan çalışmada Daha Hızlı Bölgesel Evrişimsel Sinir Ağları (Daha Hızlı B-ESA - Faster R-CNN) ve YOLOv4 mimarileri 2595 adet görüntü ile eğitilmiş, Daha Hızlı B-ESA mimarisinde %93 doğruluk oranı elde edilirken YOLOv4 mimarisinde %88 doğruluk oranı elde edilmiştir [15].

YOLOv3 DarkNet-53 mimarisi kullanılarak drone kamerası ile kaydedilen 25 FPS (saniyede 25 kare) videolardaki insan ve araçların tespiti üzerine yapılan çalışmada %78 civarında bir doğruluk oranına ulaşılmıştır [16].

İHAlar ile hareket halindeki İHA ların tespiti ve takibi üzerine yapılan çalışmada DÖ tabanlı YOLOv3 ve YOLOv3-Tiny algoritmaları, algılanan nesnelerin gerçek zamanlı takibi için ise KCF (Kernelized Correlation Filter - Çekirdekleşmiş İlinti Filtresi) kullanılmıştır. YOLOv3 ve YOLOv3-Tiny algoritmalarının eğitilmesi ve test edilmesi için farklı İHAların bulnduğu veri seti oluşturulmuştur. YOLOv3 modelinde %48 başarım elde edilirken, YOLOv3-Tiny modelinde %46 başarım elde edilmiştir [17].

DÖ tekniklerinden YOLOv2 modeli kullanılarak İHAlardan elde edilen görüntülerdeki araçların tespiti üzerine yaapılan çalışmada %77 doğruluk oranı elde edilmiştir [18].

İHAdan alınan görüntüler ile oluşturulan yaklaşık 10000 imgeli veri seti ile 8 - 4 kök sezicili YOLO-Tiny, YOLOv2 ve EPCC (Çok Yüzlü Konik Sınıflandırıcı - Extended Polyhedral Classifier) algoritmalarını kullanan yöntemleri ile İHAlardan alınan görüntüdeki araçlar ve konumları gerçek zamanlı tespit edilmiştir ve %84,49 başarım elde edilmiştir [19].

İHA ile gerçek zamanlı ve bulut tabanlı nesne tespiti için Pascal VOC2012 veri seti ile eğitilen algoritmalardan Faster R-CNN ile %83.9, SSD300 ile %81.6, SSD500 %82.6, YOLO ile %78.3 ve Fast YOLO ile %79.4 doğruluk oranına ulaşılmıştır [20].

2. MATERYAL VE METOT

2.1 Beyin, Akıl, Zekâ, Yapay Zekâ

YZ kavramını anlayabilmek için beyin, zekâ, akıl gibi kavramları anlamak etkili olacaktır;

Beyin

Yaklaşık bir buçuk kilo ağırlığında olan ceviz görüntüsündeki bu organ, 60 yıllık bir ömürde saniyede 600 birimlik hafıza kaydedip, işleyip programlamak kapasitesine sahiptir. Bu, dakikada 3.600, saatte 2.160.000 günde 51.840.000 bitlik bilgi demektir [21].

Dr. V. Grey Walter'in incelemelerine göre, insan beynine benzeyen bir makinanın yapılabilmesi için 300 trilyon dolardan fazla para gerekmektedir. Böyle bir makinenin çalışabilmesi için ise 1 trilyon wattlık elektrik enerjisine ihtiyaç vardır.
Scott Witt adlı yazarın bu konudaki tespiti ise "yaşamımız boyunca beyin, gözlerinizle, kulaklarınızla burnunuzla, parmaklarınızla ve diğer duyu organlarınızla, devamlı olarak elektrik sinyalleri şeklinde, bilgi alır, depolar, gönderir. Beyninizden geçen milyarlarca gerçek ve hayal, doksan milyon kalın kitabı doldururdu." şeklindedir.

Akıl - Zekâ

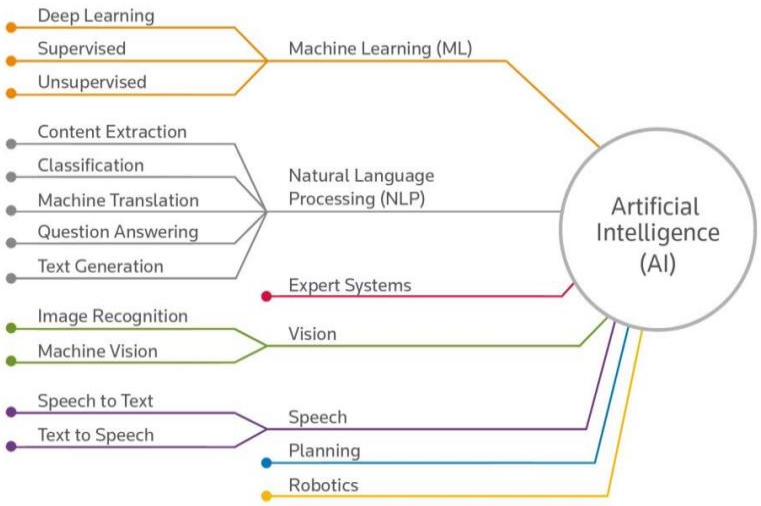
Akıl ve Zekâ birbirleriyle karıştırılmaması gereken kavramlardır. Akıl, düşünme, kavrama, idrak etme, görüş bildirme, karar verme, önlem alma yeteneğidir. Zekâ ise genel anlamda, bir bilgiyi alma, bir olayı algılama, anlama, hızlı ve doğru bir şekilde analiz etme, yargılama ve çözmeyi kapsar.

Yapay Zekâ

Bir çalışmada makinelerin insanların yaptığı işleri yapmasına YZ denmiştir [26]. Axe göre ise YZ, akıllı programları hedefleyen bir bilimdir [27]. YZ alanında akıllı davranışlar üzerine çalışılır, çeşitli canlıların davranışları yapay olarak oluşturulur [28].

Günümüz ihtiyaçlarından ve gelinen noktadan yola çıkarak bu tanımlamaları genişletip aslında olması istenen YZ'yı; Verilen görevleri yerine getirmek için insan zekâsını taklit eden ve topladıkları bilgilere göre yinelemeli olarak kendilerini iyileştirebilen sistemler veya makineler olarak tanımlayabiliriz. Hatta en basit şekliyle düşünürsek, YZ diye adlandırılan kavramın aslında insanların düşünebilme, anlayabilme, yorumlayabilme, genelleyebilme, çözüm yolu bulabilme, geçmişteki deneyimlerinden öğrenebilme gibi yeteneklerinin taklit edilip oluşturulan algoritmalardan, programlardan başka bir şey olmadığını çok net görürüz.

İnsanların çevresinde olup biteni algılaması ve düşünmesi, akıl yürütmesi, problemi tespit etmesi, tespit ettiği problemi analiz edip sebeplerini anlamaya çalışması, kavraması, yargılaması, çözüm için plan-program yapması, yapılan planları uygulaması, hatalar yapması, hatalardan ders alması, tekrar denemeler yapıp deneyimlerden öğrenmesi, mantık yürütmesi, sonuçları optimize etmesi, konuşması, konuşulanları anlaması, duygusallaşması, duygusallığı algılaması, sezgisel yaklaşması, öngörebilmesi, durumlara uygun hareket edebilmesi, bir bilgiyi sentezleyip özetleyebilmesi, matematik teoremlerini ispatlayabilmesi, zorlu oyunlar oynayabilmesi, sanat ya da müzik eserleri oluşturabilmesi, tüm bunlar YZ'nın alt dallarını oluşturan ve makinelere kazandırılmak istenen niteliklerdir. Şekil 2.1'de YZ'nın ayrıldığı alt dalların bir kısmını görebiliriz.



Şekil 2.1 Yapay zeka ve bazı alt dalları

2.1.1 Yapay zekânın bazı alt dalları

2.1.1.1 Makine öğrenmesi

Makinelerin öğrenmesini sağlayan ve bu amaçla geliştirilen teknolojidir. Sistem, verilen örnek veriler üzerinden girdi ve çıktıları arasındaki bağlantının mantığını, algoritmasını öğrenerek, öğrendiklerini deneyimlemelerle iyileştiriyor ise, her deneyimden öğrenmeye devam ediyor ise MÖ gerçekleşiyor denilebilir.

2.1.1.2 Doğal dil işleme

Bir bilgisayarın, insanın konuşmasını anlayabilmesi ve insanın anlayabileceği dilde konuşması için dilin tüm özelliklerini bilmesi gerekir. Ve bu da doğal dil işleme ile olur.

2.1.1.3 Uzman sistemler

Bir problemi, o problemin uzmanları gibi çözebilen bilgisayar programları geliştiren teknolojidir. Bir "bilgi mühendisi" bazı görevleri yerine getirmek için belirli alanlardaki uzmanlarla görüşür ve edindiği bilgileri belli bir algoritma ile somutlaştırmaya çalışır ve bilgisayar programına döker.

2.1.1.4 Görme - Bilgisayarlı görme

Bilgisayarlı görme, temelde görme ve gördüklerini analiz etme anlamında bir insanın yapabileceği görevleri bilgisayarlı sistemlerin yapabilmesini sağlamaktır. Dijital veya video görüntüleri üzerinde sayısal veya sembolik bilgi üretmek için, görüntü üzerinde işlemler yapma, çıkan sonucu analiz etme, anlama ve bir insan gibi karar verebilmeyi içermektedir.

Program, bir tür gözlem yaptığında, genellikle gördüklerini seçilen bir desenle karşılaştırmaya programlanır. Bu işlem Örüntü Tanıma (Pattern Recognition) olarak tanımlanır. Örneğin bir yüz tanıma programı, bir yüzü bulmak için göz, burun veya çene eşleştirmesi, karşılaştırması yapabilir. Daha karmaşık işlemlerde ise, örneğin bir metinde, bir satranç pozisyonunda karşılaştırmalar yapmak, üzerinde en çok çalışılan basit kalıplardan daha farklı yöntemler gerektirir [29].

2.1.1.5 Planlama

Planlama programları, dünya hakkındaki genel gerçekler, özellikle eylemlerin etkileri hakkındaki gerçekler, belirli bir durum hakkındaki gerçekler ve bir hedef beyanı ile başlar. Bunlardan, hedefe ulaşmak için bir planlama stratejisi, eylemler dizisi oluşturulur.

2.1.1.6 Sağduyu bilgisi - Akıl yürütme

1950'lerden beri aktif bir araştırma alanı olmasına rağmen YZ'nın insan seviyesinden en uzak olduğu alandır. Önemli ilerleme kaydedilmiş olsa da, örneğin monoton olmayan akıl yürütme sistemleri ve eylem teorileri geliştirmek için daha fazla yeni fikre ihtiyaç vardır. Sağduyu bilgisini yakalamayı uman Cyc projesinin amacı, insan sağduyusunu oluşturan milyonlarca bilgiyi makinede kullanılabilir biçimde kodlamaktır [34].

2.1.1.7 Mantıksal zeka

Bir programın genel olarak dünya hakkında ne bildiği, içinde hareket etmesi gereken belirli bir durumun gerçekleri ve hedeflerinin tümünü matematiksel mantıksal cümlelerle temsil etmeyi içerir. Program, belirlenen eylemlerin, hedeflerine ulaşmak için uygun olup olmadığı sonucuna vararak ne yapılacağına karar verir [29].

2.1.1.8 Çıkarım

Bazı gerçeklerden hareketle başka gerçeklere erişilmesine denir. Matematiksel mantıksal tümdengelim bazı çıkarımlar için yeterli olabilir ancak, 1970'lerden beri mantığa yeni monoton olmayan akıl yürütme (non-monotonic reasoning) yöntemleri eklenmiştir. Monoton olmayan akıl yürütmeyi basit bir şekilde tanımlarsak, aksine kanıt olmadığı sürece varsayılan doğrunun kabul görülmesi, aksine kanıt olması durumunda ise sonucun geri çekilebilir olmasıdır. Örneğin, bir kuşu duyduğumuzda onun uçabileceği çıkarımına varırız, ancak onun bir penguen olduğunu duyduğumuzda bu sonuç tersine dönebilir [29].

2.1.1.9 Genetik programlama

Genetik programlama, mutasyon, kromozom, gen gibi genetik fonksiyonların bilgisayar programlarına uygulanması ve sonuç elde etmek için bilgisayar

programlarının çalışma şeklini bulmayı hedefleyen MÖ yaklaşımıdır denebilir. Tanımdan da anlaşılacağı gibi karmaşık optimizasyon problemlerinin çözülmesinde kullanılan bir teknolojidir. Bir problemi çözebilmek için öncelikle rastgele başlangıç çözümleri belirlenmektedir. Daha sonra bu çözümler birbirleri ile eşleştirilerek performansı yüksek çözümler üretebilmektedir.

2.1.1.10 Bulanık mantık

Bulanık mantık, küme teorisinde üyelik derecesi kavramını geliştirmiştir. Yani %100 ve %0 ın ötesinde ara değerleri görünür kılmıştır. Örneğin gençler kümesine 25 yaşındaki bir insan %100 üye iken, 60 yaşındaki bir insan %30 üyedir şeklinde ifadeleri vardır. Böylesine bir açılım subjektif verilere dayansa da kazandırdığı esneklik ve gerçek hayat olaylarına daha iyi çözüm önerebilme itibariyle çok taraftar toplamıştır. Günümüzde de bulanık mantıkla çalışan ev aletleri, arabaların çeşitli kısımları ve elektronik cihazlar üretilmektedir.

2.1.2 Yapay zekanın bazı uygulama alanları

2.1.2.1 Geliştirilen ilk uzman MYCIN

MYCIN ilk uzman sistem olarak kabul edilmiştir ve 1970 yıında bir grup araştırmacı tarafından Stanford Üniversitesi'nde tasarlanmıştır. Tasarlanma sebebi çeşitli hastalıkların teşhis ve tedavisidir.

2.1.2.2 Google duplex

Google asistanın sağladığı bir imkan olup insan sesini taklit ederek insan gibi konuşan bir uygulamadır. Bu yazılım, insanlara benzer bir lisanı konuşmak için tasarlanmıştır.

2.1.2.3 Tesla autopilot

Tesla firması tarafından kendi ürettikleri araç için sürücüsüz veya sürücülü olarak trafiğin akış esnasında yol şerit takibi, adaptif hız kontrol, otomatik park gibi çeşitli manevraları otomatik olarak yapmak için tasarlanmıştır. Bu yazılım ile hedef otomatik olarak sürücüsüz araba kontrolü sağlamaktır.

2.1.2.4 Facebook deepface

Bu yazılım görüntü veya video frameleri kullanarak çerçevelerde olan kişileri tanımlar. Bu amaçla kullanılabilen birçok yöntem veya yazılım bulunmaktadır ve verilen görüntüden seçilen kişilerin simalarını bir veri tabanındaki simalar ile karşılaştırarak çalışır.

2.1.2.5 Sophia

Sophia, Hong Kong merkezli Şirket Hanson Robotics tarafından geliştirilen bir sosyal insansı robottur. Gözlerindeki kameralar bilgisayar algoritmalarıyla birleşerek görmesine izin vermektedir. Böylece yüzleri takip edebiliyor, göz teması kurabiliyor ve bireyleri tanıyabiliyor. Fonksiyonel yürüme kabiliyetine sahip olmakla birlikte konuşabilmektedir.

2.1.2.6 IBM Watson

IBM Watson, sistemsel problemleri insancıl bakış açısı ile anlamakta ve sonrasında 4 adımda karar vererek çözümler sunmaktadır. Watson, bilgiyi alıp kullanırken doğal dil kullanır ve dil bilgisi kurallarına uyar. Kullanıcının asıl amacını anlamaya çalışır ve mantıksal bütünlüğü sağlayarak, sahip olduğu dilsel modeller ve algoritmalar aracılığıyla beklenen cevapları oluşturur.

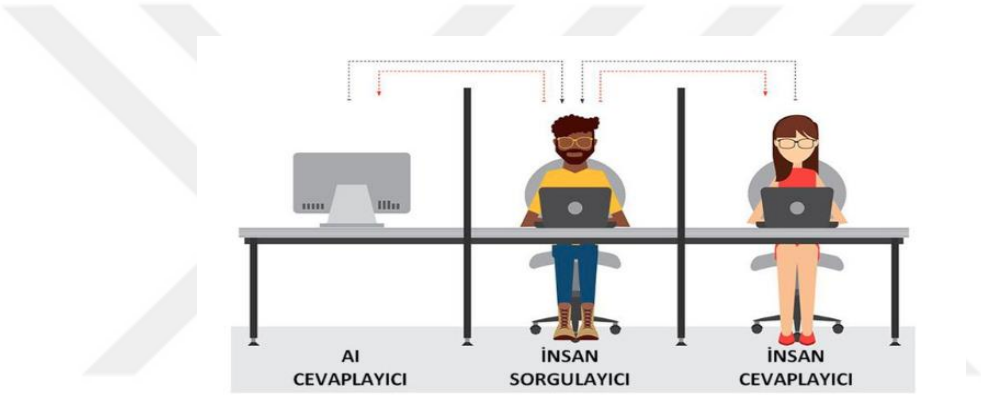
2.1.2.7 Bilgisayar oyunları (Computer games)

Çok iyi satranç oynayabilen, yüksek hesaplamalar ile saniyede 200 milyon pozisyona bakıp bir dünya şampiyonunu yenebilen makineler ile genetik algoritmalar kullanılarak, gerçek hayatın kopyasının oluşturulmaya çalışıldığı, stratejik yaşam simülasyonu video (*The Sims*) oyununu YZ teknolojisinin kullanıldığı oyunlara örnek olarak gösterilebilir. Bunlara ek olarak YZ'ya sahip bazı oyun botları, çoğu ölçütü göz önünde bulundurarak kullanıcıların hamlelerini tahmin edebilir ve kuvvetli birer rakibe dönüşebilir.

Bahsi geçen uygulama alanlarına ek olarak, YZ'nın birçok alt dalının kullanıldığı İHA ile yapılan çeşitli çalışmalar bulunmaktadır. Bu çalışmalar arasında yer alabilecek, İHA ile araç ve yaya tespiti yapılan bu uygulamada, YZ alt dallarından Bilgisayarlı Görü ve MÖ'nün alt dalı olan DÖ'den faydalanılmıştır.

2.2. Makine Öğrenmesi

Makineler düşünebilir mi? Makinelerin gözünü kulağını açan esasında bu soru olmuştur. Turing aslında bu sorunun cevabını, 1936 yılında, matematiksel mantığın soyut bir problemi konulu makalesinde yoğunlaştığı problemi çözerken bulmuştur ve Turing Makinesi diye adlandırılan bilgisayarı kuramsal olarak icat etmeyi başarmıştır. Alan Turing ayrıca Turing testi olarak adlandırılan bir test geliştirmiştir. Bu test bir makinenin düşünebildiğini söylemenin mantıksal olarak mümkün olup olmadığıdır. Turing testine göre makine ve insan Şekil 2.2'deki gibi bir sorgulayıcının görüş alanının dışında bir yere konumlandırılır. Sorgulayıcı yalnız soru sormak suretiyle hangisinin insan hangisinin bilgisayar olduğunu saptamaya çalışır.



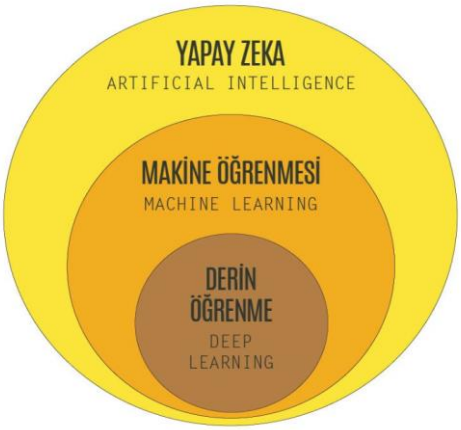
Şekil 2.2 Turing testi

Düzenli olarak tekrarlanan testler neticesinde sorgulayıcı aldığı cevaplara göre, makine ve insanı ayırt edemediği takdirde makine Turing testini geçmiş olur ki Turing testinde başarılı olan makineler insan gibi davranan sistemler olarak tanımlanır. Daha önce de değinildiği gibi istenilen aslında makinelerin zeki davranabilme kabiliyetinin yanında insan gibi davranabilmesidir. Genel olarak geliştirilen algoritmalar ve bu algoritmalar ile geliştirilen uygulamaların öğrenme, öğrendiği bilgiyi genelleyip yeni durumlara uyarlayabilme, tanıma ve tahminde bulunabilme görevlerini yerine getirebilmesi MÖ olarak adlandırılmaktadır.

MÖ de, öncelikle alan belirlenir. Sonrasında bu alanla ilgili datalar elde edilir. Bu aşamada datalara yani ham verilere ön işlem uygulanır, kullanmaya uygun hale getirilir. Daha sonra, bu veriler eğitim ve test aşamalarında kullanılmak üzere kısımlara ayrılır, eğitim ve test işlemi yapılır. Eğitim aşamasında ve sonrasında, sonuçların uygun hale getirilmesi aşamasında çeşitli yöntemler kullanılır.

2.3 Derin Öğrenme

DÖ, dijital sistemlerin öğrenmesi temeline dayalı bir MÖ türüdür, bunun yanında DÖ Şekil 2.3'de gösterildiği gibi MÖ nün bir alt dalıdır.



Şekil 2.3 Dijital sistemlerin öğrenme hiyerarşisi

Ancak DÖ'nün MÖ'ye göre yetenekleri farklılaşmaktadır. Genelde MÖ'de bir model, verilerden alınan örnekleri inceleyerek desenleri tanıyacak, öneriler sunacak ve uyum sağlayacak biçimde eğitilirken, DÖ'de model, doğrudan görüntülerden, metinlerden veya seslerden eğitir. Böylece dijital sistemler, yalnızca kural kümelerine yanıt vermek yerine, örneklerden faydalanarak bilgi edinir ve ardından bu bilgileri kullanarak insanlar gibi tepki verir, davranış gösterir ve performans sergiler.

MÖ modelleri, yalnızca eğitildiği konularda karar alabilecek düzeyde sınırlı iken, DÖ modelleri kendi kendine öğrenmeye, akıllı kararlar almaya devam edebilecek biçimde yapılandırılmıştır. DÖ'deki gelişim, güçlü bilgisayarların tasarlanmasıyla ilerleme kaydetmiştir.

DÖ'nün temelini Sinir Ağları oluşturmaktadır. Sinir ağlarının tarihsel gelişimine bakıldığında 1943'de YSA'ya, 1986 da YSA da Geri Yayılım Algoritmasına, 1996 da ise ESA'lara rastlanmaktadır. 1998 de ESA'nın belge tanıma üzerine ilk uygulaması gerçekleştirilmiştir. Ancak ESA'ları popüler hale getiren, bir ESA modeli olan AlexNet in 2012 yılında ImageNet yarışmasındaki başarısıdır.

2.3.1 Gerçek ve yapay sinir ağları

Nöronlar, insan vücudundaki en ilginç hücrelerdendir. Bunun sebebi bu hücreler düşüncelerden, duygulardan ve bütün diğer sezilerden sorumlu olup insan beynini ve tüm sinir sistemini oluşturur. Nöronlar (sinir hücreleri), veri içeren elektrik sinyallerini alan ve diğer nöronlara ileten özelleşmiş hücrelerdir [30].

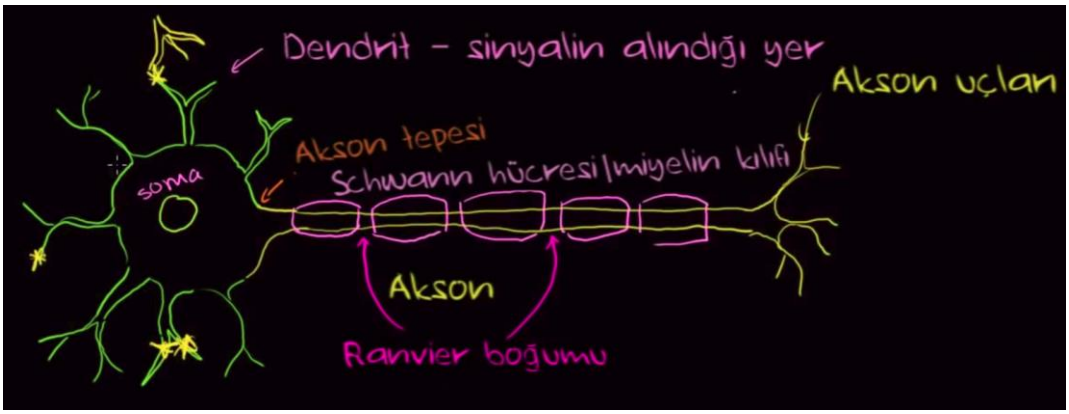
Nöronların bölümleri ve anatomik yapısı

Şekil 2.4'de detaylı bir şekilde gösterildiği gibi nöronlar üç temel bölümden oluşur; dendrit, hücre gövdesi ve akson.

Dendrit: Yunancada ağaç anlamına gelir. Hücre gövdesinden dışarı doğru uzanır ve ağaç dallarına benzer. Diğer hücrelerden gelen sinyali (bilgiyi,veriyi) alır ve hücre gövdesine taşır.

Hücre Gövdesi (Soma): Hücreyi denetler ve yönetir.

Akson (Axon): Sinyalin seyahat ettiği, yol aldığı uzantıdır. Hücre gövdesindeki sinyali diğer nöronların dendritlerine taşır. Son bölümü ağaçsal yapıyı andırır ve dallarının sonunda da sinaptik terminaller vardır.



Şekil 2.4 Nöronun anatomik yapısı ve bölümleri [30]

Sinaps (Sinaptik terminal): Aksondan akson uçlarına doğru giden sinyalin diğer nöronun dentritine ya da hücre gövdesine aktarıldığı iletişim noktasıdır daha doğrusu boşluktur. Sinapslar, gelen sinyallerin tek yönlü iletilmesinden sorumludur.

Akson Uçları: Bir nöronun aksonunun diğer nöronun gövdesine bağlandığı, sinaps oluşturan noktaya denilir.

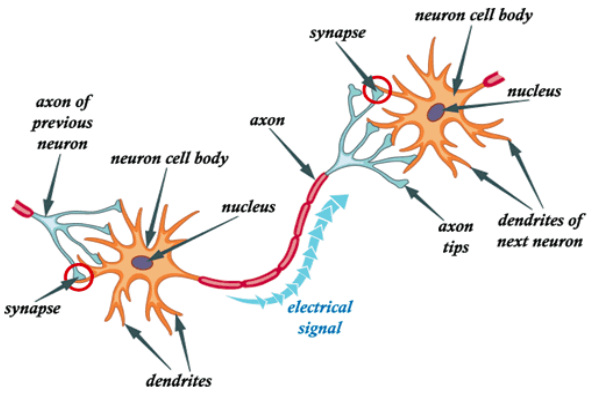
Miyelin Kılıf : Aksonların çoğunun çevresinde miyelin kılıf denilen bir örtü vardır ve sinyalin hızlı bir şekilde taşınmasını sağlar. Miyelin kılıf, schwann hücrelerinden oluşur.

Nöronların çalışma şekli

Şekil 2.5'te de görüldüğü üzere nöronlara gelen sinyaller dendritler aracılığı ile alınıp, hücre gövdesine gönderilir oradan da etkileri hesaplanan sinyaller akson tepesine iletilir. Eğer sinyal yeterince güçlüyse akson üzerinden geçmesine izin verilir. En son akson uçlarına gelen sinyaller sinapslardan (sinaptik boşluklardan) başka dendritlere veya kaslara iletilir.

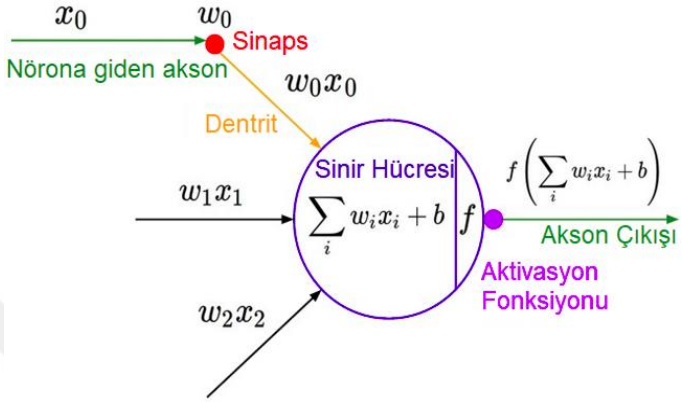
Sinaptik boşluk içerisinde "sinaptik kesecikler" bulunur. Sinaptik kesecikler, gelen sinyallerin diğer dendritler veya kaslara geçmesini koşullayan elemanlardır. Sinaptik boşluğa, "sinaptik kesecikler" tarafından sağlanan nöro-iletken maddenin dolması sinyallerin diğer nöronlara geçişini koşullar.

Nöronlara gelen sinyallerle uyumlu olarak nöronlar arasındaki mevcut sinaptik ilişkilerin değişimi veya hücreler arasında yeni sinaptik ilişkilerin kurulması "öğrenme" sürecine karşılık gelir.



Şekil 2.5 Biyolojik nöron

İlk yapay sinir ağı modeli 1943'de sinir hekimi Warren McCulloch ve matematikçi Walter Pitts tarafından "Sinir Aktivitesinde Düşüncelere Ait Bir Mantıksal Hesap (A Logical Calculus of Ideas Immanent in Nervous Activity)" başlıklı makale ile gündeme gelmiştir.



Şekil 2.6 Yapay nöron (biyolojik nöronun matematiksel modeli)

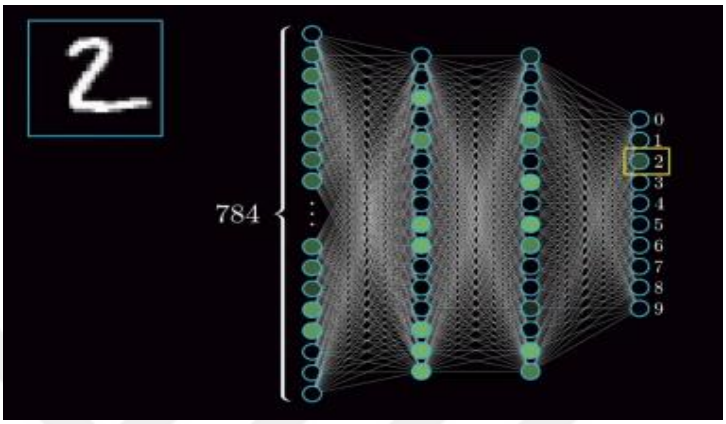
YSA'lar, öğrenme, hatırlama, genelleme yapma yolu ile topladığı verilerden yeni veri üretebilme gibi temel işlevlerin gerçekleştirildiği bilgisayar yazılımlarıdır. Diğer bir ifade ile, Şekil 2.6'da da görüldüğü gibi, nöronların anatomik yapısının, çalışma şeklinin taklit edilmesi ve beynin öğrenme işleminin matematiksel modellenmesi sonucu oluşturulan algoritmalardır.

Yapay sinir ağlarının çalışma şekli

YSA (Model) bir giriş bilgisi alıp bunu sinir ağından geçirir ve bir olasılık değeri döner. Sonra da en yüksek olasılığa bakıp tahminleme yapar. Loss fonksiyonu da bu olasılığın ne kadar doğru olduğunu hesaplayıp Modele hatasının büyüklüğünü söyler. Model hatasına bakar ve geri gidip (geri yayılım - back propagation) "ağırlık (weight)" ve "eşik (bias)" parametrelerini düzenler. Eğitim esnasında bu işlem "dataset"e bağlı olarak binlerce hatta milyonlarca kez tekrarlanır. Böylece model öğrenmesi gereken şeyi öğrenir. Bütün "dataset"in 1 kere bu işlemi gerçekleştirmesine epoch (iterasyon) denir [31].

Şekil 2.7'de MNIST veri setinden alınan 28 piksel genişliğinde ve 28 piksel

uzunluğunda bir resmin düzleştirilip (28*28 den) 784 piksel uzunluğunda bir vektör şeklinde, 1 giriş, 1 çıkış ve 2 gizli katmandan oluşan sembolik bir yapay sinir ağına verildiği ve yapay sinir ağının çıkış katmanının verdiği sinyale göre tahminleme yaptığı görülmektedir.



Şekil 2.7 Bir yapay sinir ağının sembolik çalışması

Şekil 2.7'deki sinir ağı, katman türlerinden Tam Bağlı Katmana (TBK) bir örnektir ve nöronlar birbirine x*w+b formülü ile bağlanır.

x: Gelen girdi (input) değeri (Resim)

w: ağırlık (weight), Öğrenme ağırlık değişimi ile sağlanır.

b: bias (eşik değeri, yönelim), ağ çıkışının 0 olmasını önleyen değer.

Modele girdi olarak x verilir ve eğitim esnasında model w ve b'yi güncelleyerek daha doğru sonuçlar çıkarmaya çalışır.

Tablo 2.1 Biyolojik nöronun bölümleri ve yapay nörondaki karşılıkları

Biyolojik nöron	**Yapay nöron**
Akson uçları (Gelen sinyal)	Nöron çıkışı gelen sinyal (x_0, x_1, x_2)
Sinaptik kesecikler (nöro-iletken madde)	Ağırlıklar (w_0, w_1, w_2)
Dentrit (Sinaptik kesecikten geçebilen sinyaller)	x_0*w_0, x_1*w_1, x_2*w_2
Nöron (Sinir hücresi)	Toplam fonksiyonu $\sum x \boxtimes w \boxplus b = x_0*w_0 + x_1 *w_1 + x_2 *w_2 + b$
Hücre gövdesi	Transfer fonksiyonu => $\square (\sum x_n*w_n + b)$ (Toplam fonksiyonu + aktivasyon fonksiyonu)

2.3.1.1 Yapay sinir ağını oluşturan önemli parametreler

YSAlar, katmanların yani farklı düzeylerde yer alan nöron gruplarının bir araya gelmesi ile oluşur, tek katmanlı algılayıcılar ve çok katmalı algılayıcılar olarak da ikiye ayrılır. Tek katmanlı algılayıcılar giriş katmanı ve çıkış katmanından oluşur. Çok katmanlı algılayıcılar ise giriş katmanı, çıkış katmanı ve gizli katmanlardan oluşur.

Yapay sinir ağının katmanları

YSA'lar üç ana katmandan oluşur;

Giriş Katmanı: YSA'nın ilk katmanıdır ve girdiler bu katmana uygulanır. Daha sonra giriş verileri doğrudan gizli katmana aktarılır.

Gizli Katman: Giriş değerleri belirli ağırlıklarla çarpılır ve toplanır ve elde edilen sonuçlar eşik fonksiyonlarına sokulur. Gizli katmanlar birden fazla nörona sahip olabilr ve birden fazla gizli katman kullanılabilir [32]. Öğrenme bu katmanda olur.

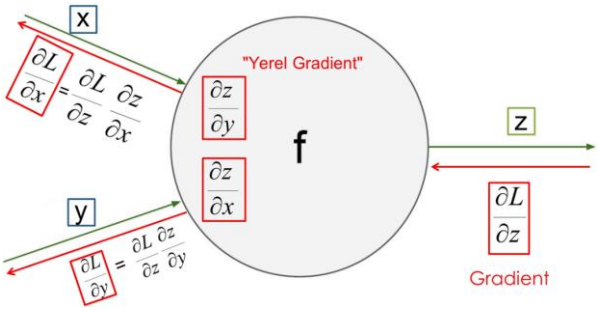
Çıkış Katmanı: Yapay Sinir ağının son katmanıdır ve YSA'nın çıkışları bu katmandan alınır.

Öğrenme algoritması

YSA'ların en önemli özelliklerinin başında, veriyi kaynağından öğrenme gelir. Veriler, biyolojik ağlarda sinapslarda, YSA'larda ise ağırlık (weight) diye adlandırılan uçlarda muhafaza edilir. Bu da ağırlıkların aldığı değerlerin ne kadar önemli olduğunu gösterir, çünkü öğrenme ağırlık değişimi ile sağlanır. Tüm ağ ele alınacak olursa, ağırlıkların en uygun değerileri alması için geri yayılımlı öğrenme (backpropagation learning) yöntemi kullanılır ve bu işlem defalarca tekrar edilerek ağın eğitimi sağlanır.

Geri yayılım (Backpropagation)

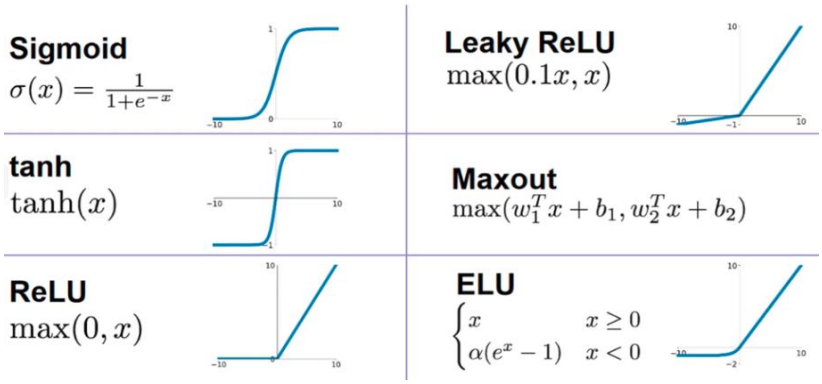
Bu yöntem yapay sinir ağı parametrelerinin güncellenmesi için en çok kullanılan yöntemdir. Başlangıçta rastgele olarak "ağırlık (weight)" ve "eşik (bias)" için değerler atanır. Model girdi (input) değerini alır, ileriye doğru giderek ağırlık ve bias değerleri ile bir çıktı (output) değeri verir. Bu çıktı değeri (ağın çıktısı) amaçlanan çıktı ile karşılaştırılır, aradaki fark ile hata payı elde edilir. Şekil 2.8'deki gibi Geri Yayılım metodu ile model geri gider daha isabetli sonuç için ağırlık ve bias değerlerini günceller. Bu güncelleme işlemi türevin zincir kuralı ile olur.



Şekil 2.8 Tek nöron üzerinde geri yayılım işlemi [31]

Aktivasyon fonksiyonları

Aktivasyon fonksiyonlarının amacı "ağırlık (weight)" ve "eşik (bias)" değerlerini ayarlamaktır. x*w+b işlemi yapıldıktan sonra elde edilen sonuç aktivasyon fonksiyonundan geçirilir. Şekil 2.9'da bazı aktivasyon fonksiyonları yer almaktadır. Aktivasyon fonksiyonları gelen sonucu değerlendirir ve sonraki nörona iletip iletilmemesine karar verir. Aktivasyon fonksiyonları, nöronların tetiklenmesi olarak düşünülebilir, yüksek bir değer geldiği zaman nöron daha yüksek bir sinyal verir. Bu yüzden fonksiyonun doğru seçilmesi çok önemlidir. Çünkü sonuç performansı etkileyecektir.



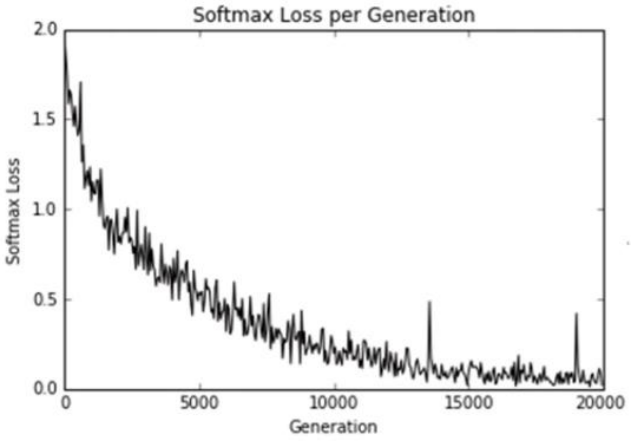
Şekil 2.9 Bazı aktivasyon fonksiyonları [31]

ReLU diğer aktivasyon fonksiyonları gibi yoğun hesaplama yapmaz, gelen değer pozitif ya da negatif olmasına bakar. Eğer gelen değer negatif ise fonksiyon sonuç olarak sıfır döner, gelen değer pozitif ise fonksiyon değeri olduğu gibi döner. ReLU, Sigmoid ve tanh'e göre daha iyi sonuçlar vermesinin yanında bilgisayarlar için de oldukça performanslıdır. Bilgisayar, Sigmoid ve tanh fonksiyonlarında karmaşık

hesaplamalar yaparken ReLU fonksiyonunda sadece değerin pozitif mi negatif mi olduğuna baktığı için ReLU fonksiyonunu sigmoid ve tanh e göre 6 kat daha hızlı hesaplar. Aktivasyon fonksiyonunun eğitim sırasında milyonlarca defa hesaplandığı düşünülürse ReLU'nun eğitimin hızlanmasına büyük katkısı olacaktır. Buna ek olarak ReLU nun çalışma yapısı, insan sinir sisteminde bulunan biyolojik nöronlarla benzerdir. Biyolojik nöronlar belli seviyenin altında gelen sinyalleri görmezden gelir. Bu seviyeye eşik şiddeti denir, sinyal bu eşik şiddetini geçmezse tıpkı ReLU fonksiyonunda sıfırın altında gelen değerlerin sıfır sayılması gibi yok sayılır. Sonuç olarak öğrenme oranı (learning rate) iyi ayarlanırsa ReLU en iyi sonucu verecek olan aktivasyon fonksiyonu olabilir.

Loss (Cost) fonksiyonu

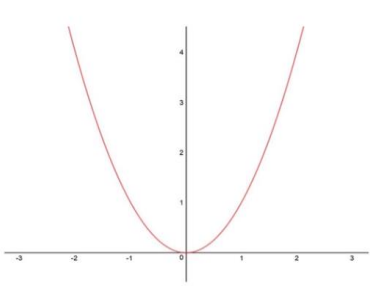
Model, bir giriş bilgisi alıp sinir ağından geçirdikten sonra döndüğü olasılık değerine göre tahminleme yapar. Loss fonksiyonu da bu tahminin doğruluk derecesinin matematiksel hesabını yapar yani tahmin değeri ve gerçek değer arasındaki farkı hesaplar. Böylece model loss fonksiyonuyla hatalarını görüp optimizasyonla minimuma indirmeye çalışır. Loss değeri en küçük değere ulaşana kadar eğitim devam eder. Şekil 2.10'da örnek bir loss grafiği görülmektedir.



Şekil 2.10 Loss fonksiyonu grafiği örneği [31]

Kullanılacak olan algoritmalara uygun birçok loss fonksiyonu vardır ve tüm bu loss fonksiyonlarının amacı modele ne kadar hata yaptığını göstermektir.

L2 Loss Fonksiyonu:



$$S = \sum_{i=1}^{n} (y_i - f(x_i))^2$$

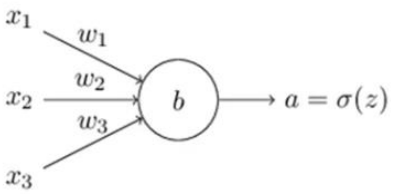
y: Doğru olan değer f(x): Tahmin

Şekil 2.11 L2 loss fonksiyonu [31]

Şekil 2.11'de L2 loss fonksiyonu en sık kullanılan fonksiyonlardan biridir. Hedeflenen sonuçtan tahmin edilen sonucu çıkarıp (hata) karesini alırız. Böylece hatanın karesi kadar cezalandırma yapılır.

Cross Entropy Loss Fonksiyonu:

Şekil 2.12'de ise cross entropy loss fonksiyonunun grafiği ve formülü görülmektedir.



$$z = \sum_j w_j x_j + b$$

σ: sigmoid

$$C = -\frac{1}{n} \sum_x [y \ln a + (1-y) \ln(1-a)]$$

n: Eğitim verisi sayısı y: Doğru olan değer

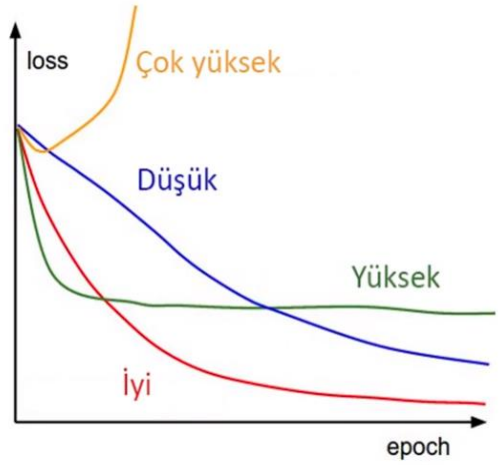
Şekil 2.12 Cross entropy loss fonksiyonu [31]

Optimizasyon

Loss değeri ile doğru tahmin sayısı arasında ters bir orantı vardır. Loss değeri ne kadar düşükse doğru tahmin sayısı o kadar fazladır denilebilir. Optimizasyonun burdaki rolü ise Loss değerini sıfıra yaklaştırmaktır ve bunun için kullanılan yöntemler gradyan iniş (Gradient Descent) olarak tanımlanır. Gradyan iniş bir süreçtir ve adım adım işletilir, bahsi geçen adımın büyüklüğüne de öğrenme oranı denir. Özetle öğrenme oranı

optimizasyon aşamasındaki en önemli parametrelerden biridir.

Öğrenme oranı (learning rate), Ağırlık (weight), Loss değeri ve Optimizasyon kavramlarını ve birbirleri ile olan ilişkiyi daha rahat kavrayabilmek adına dağlık tepelik bir arazi düşünülebilir. Bu arazideki her bir nokta ağırlık parametresinin herhangi bir ayarına eşittir. O noktaların yükseklikleri ise ağırlık ayarlarının loss değeridir. Bu arazinin herhangi bir yerinden loss değerinin en düşük olduğu yere yani arazinin en alçak noktasına gidilmek istenirse aşağıya giden yön hesaplanır o yönde adım atılır. Her adımda bu işlem tekrarlanarak en düşük nokta bulunabilir. Ancak en düşük noktayı bulunabilmesi atılan adımın büyüklüğüne bağlıdır. Eğer öğrenme oranı çok büyük seçilirse atılan her adımda tepeden tepeye gidilir ve en düşük nokta bulunamaz ki bu başarısız bir öğrenme olur. Eğer öğrenme oranı çok küçük seçilirse bu defa da en düşük noktaya ulaşmak ya çok zaman alır ya da hiç ulaşılmaz. Öğrenme oranının ne derece iyi olduğu loss değeri incelenerek anlaşılabilir. Nasıl bir loss grafiği beklenmesi gerektiği Şekil 2.13'deki grafiğe bakılarak anlaşılabilir [31].



Şekil 2.13 Seçilen öğrenme oranına göre ideal loss grafiği [33]

Sonuç olarak, öğrenme oranı, modeldeki en önemli parametrelerdendir. Doğru öğrenme oranı modelin başarısını direkt olarak etkileyecektir.

Gradyan ve gradyan iniş

Türev, bir fonksiyonun herhangi bir değişkenindeki değişimin, fonksiyonda yarattığı artış veya azalışın, değişkendeki değişime oranıdır. Gradyan ise bir fonksiyonun kısmi türevlerinin vektörüdür. Kısmi türevlerin bileşkesi gradyanın yönünü yani değişkenlerin hangi yönde değişmeleri halinde fonksiyonun artacağını gösterir.
Gradyan yönünde ilerlenirse maksimuma, gradyanın tersi yönde ilerlenirse minimuma yaklaşılır. Nabla veya del (ters üçgen) işlemcisi de bir fonksiyonun gradyanını temsil eder [35].

Matematiksel olarak gradient hesaplaması:

$$\Delta C \approx -\eta \nabla C \cdot \nabla C = -\eta \|\nabla C\|^2 \tag{2.1}$$

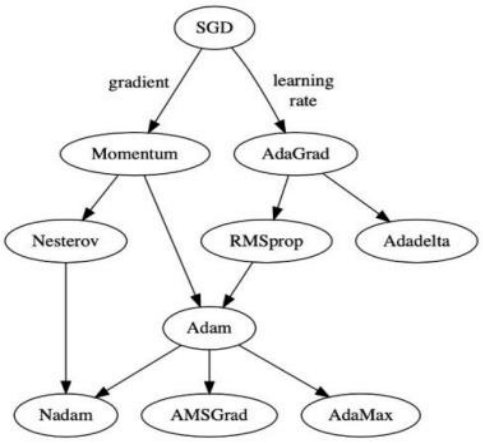
Formülde;

$\boldsymbol{\Delta C}$: Elde edilen loss değeri,

$\boldsymbol{\nabla C}$: Loss değerinin gradienti,

η : Seçilen öğrenme oranı (learning rate), çok küçük pozitif bir sayı olmalı.

Formülde görüldüğü gibi $\boldsymbol{\Delta C}$ sıfırdan küçüktür. Sonuç olarak bu işlem sürekli tekrarlanır ve C yani loss değeri sürekli düşürülerek minimuma ulaşılır [31].



Şekil 2.14 Gradyan iniş (gradient descent) yöntemlerinin evrişimsel haritası [36]

Şekil 2.14'de Gradyan iniş yöntemlerinin evrişimsel haritası, Tablo 2.2'de ise

karşılaştırması sunulmuştur.

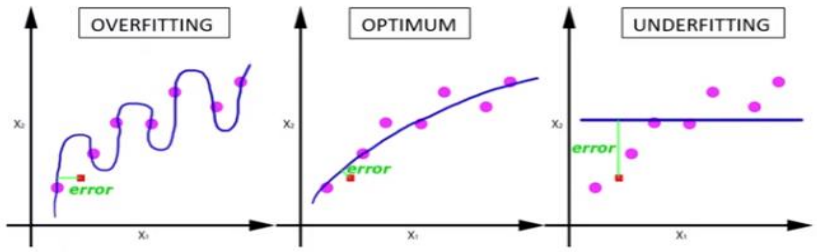
Tablo 2.2 Gradient iniş yöntemlerinin karşılaştırılması [37]

Yöntem	Tarih	Öğrenme Katsayısı	Gradient
SGD	1951	√	√
Momentum	1964		√
Adam	2014	√	√
AdaGrad	2011	√	
RMSprop	2012	√	
Adadelta	2012	√	

Aşırı öğrenme ve düzenlileştirme

Eğitilen modelin eğitim setini çok iyi öğrenip (ezberleyip) genelleştirme yapamamasına aşırı öğrenme (overfitting) denir. Aşırı öğrenmenin tam tersi ise modelin eksik öğrenmesidir (underfitting).

Tüm bu tanımlar Şekil 2.15'de daha net görülmektedir.

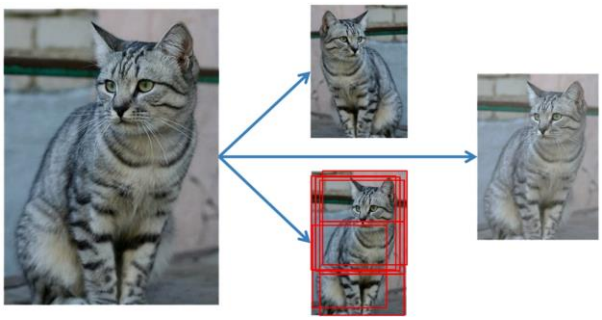


Şekil 2.15 Aşırı öğrenme - optimum - eksik öğrenmenin grafiksel ifadesi [31]

Aşırı öğrenme problemini çözmek için Düzenlileştirme metodları kullanılır. Bu metodlar;

Veri arttırma (Data augmentation) işlemi

Veri arttırma, var olan verileri değiştirerek yeni veriler üretmektir. Bu işlem eğitim esnasında yapılıp modelin daha iyi genelleştirme yapabilmesi sağlanabilir. Şekil 2.16'daki gibi resim ters çevrilebilir, parlaklığı değiştirilebilir ve resmin herhangi bir yeri yakınlaştırılabilir.

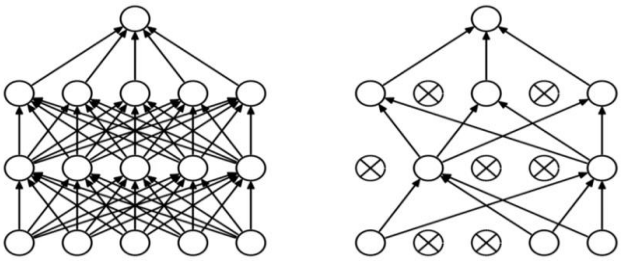


Şekil 2.16 Veri arttırma (data augmentation) [31]

Veri arttırma işlemi resimler grafiğe besleme yapılırken rastgele yapılır. Sebebi ise modele farklı resimler verebilme ihtiyacıdır.

Dropout yöntemi

Şekil 2.17'deki gibi, eğitim sırasında her ileri gidişte rassal olarak katmandaki nöronlardan bazıları sıfırlanır. Bu işleme dropout denir.



Şekil 2.17 Standart sinir ağı - dropout uygulanmış ağ [31]

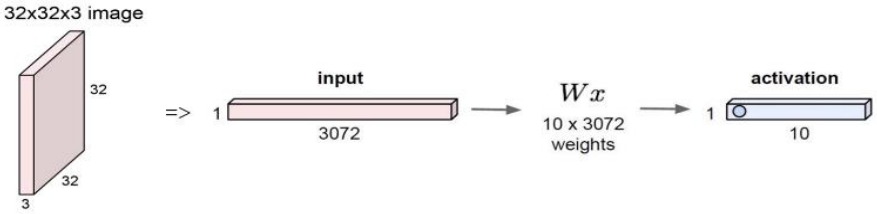
Bu şekilde modelin verinin tek bir noktasına odaklanması engellenip daha iyi genelleştirme yapılması sağlanır. Eğitim bittikten sonra yani test esnasında ise tüm nöronlar aktif hale getirilir [32].

2.3.2 Evrişimsel sinir ağları

İnsanlar bir resmi algılamak için köşelere, yuvarlak hatlara, çizgilere bakar. Ancak bir resim YSA'ya verildiğinde, o resim düzleştirilip vektör haline getirilir.

Örneğin Şekil 2.18'deki gibi 32x32x3 boyutunda bir resim (32x32 piksel boyutunda ve kanal sayısı 3, yani renklidir) yapay sinir ağından geçirilmek istenirse önce

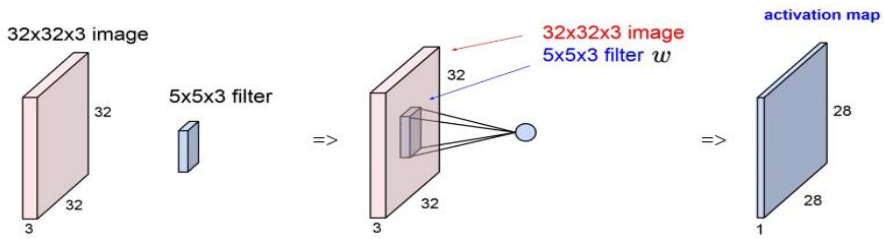
düzleştirilip vektör haline getirilir (3072x1) sonra girdi (input) olarak ağa verilir.



Şekil 2.18 32x32x3 boyutunda bir resim ve vektör halinde YSA'ya verilmesi [31]

Ancak resim üzerinde düzleştirme (flatten) işlemi yapıldığında tüm ince ayrıntılar yok olur. Bu probleme çözüm olarak ESA'lar kullanılır [31]. ESA'lar, objelerin önce kenarları gibi alt düzey özelliklerini sonra da detaya inerek diğer özelliklerini tespit edip, tanımaya çalışır. Resim vektör haline getirilmek yerine 3 boyutlu yapısı korunur ve ağırlık (weight) olarak da küçük filtreler kullanılır.

Şekil 2.19'da görüldüğü gibi 32x32x3 boyutundaki aynı resim ESA dan geçirilmek istenirse, 5x5x3 boyutundaki filtre, 32x32x3 boyutundaki resim üzerinde piksel piksel kaydırılır ve çıktı olarak 28x28x1 boyutunda bir öznitelik haritası (öznitelik matrisi) oluşur.



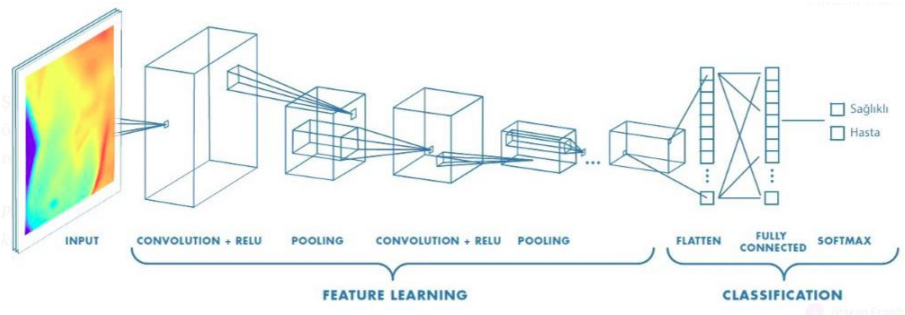
Şekil 2.19 32x32x3 boyutundaki resme 5x5x3 boyutunda filtrenin uygulanması [31]

Ayrıca normal YSA'ların kullanıldığı durumlarda resimdeki her bir piksel için modele bir giriş katmanı eklenir ve gereken bağlantılar sağlanır. Bu durum çok fazla bağlantı ve işlem gücü gerektirir. YSA yerine ESA'ların kullanıldığı durumlarda ise öznitelik haritası çıkartılarak bağlantı sayısı ve işlem gücü azaltılır.

ESAlar biyoloji ile bilgisayar bilimlerinin bir karışımı gibi görünse de, görüntü işlemede ve tanımada kullanılan özelleştirilmiş bir DÖ algoritmasıdır ve gösterdiği başarıdan dolayı da yaygın olarak kullanılmaktadır.

ESA'ya verilen resimlerin tanınması ve işlenebilir olması için ilk olarak resimler matris formatına çevrilir. Resimler (input) ESA'ya matris olarak verildiğinden, en küçük detaylarına kadar incelenir ve öğrenilirler. Dolayısıyla matrislerdeki farklılıklar baz alınarak hangi resmin hangi etikete ait olduğu sinir ağı tarafından saptanır. Görüntülerdeki yani matrislerdeki farklılıklar ve etiketler eğitim aşamasında öğrenilir ve sonrasında bunlar kullanılarak yeni resimler için tahminlerde bulunulur. Tüm bu işlemleri etkili bir şekilde gerçekleştiren ESA'lar, görüntüleri farklı katmanlarda işler. Bu katmanlar; Evrişimsel katman (convolutional layer), öznitelik seçme katmanı (pooling layer), düzleştirme katmanı (flattening) ve TBK'dan (fully connected layer) oluşur. Bunlara ek olarak aktivasyon fonksiyonları ve filtreler de kullanılır.

Şekil 2.20'de görüldüğü gibi bir ESA işlevsel olarak 2 kısımdan oluşur. Birinci kısım özniteliklerin çıkartılıp öğrenildiği kısımdır, bir veya birden fazla evrişimsel katmandan (convolutional layer) ve her bir evrişimsel katmanı takip eden aktivasyon fonksiyonundan (RELU) ve öznitelik seçme katmanından (pooling layer) oluşur. İkinci kısım ise sınıflandırma yani tahminleme yapılan kısımdır, düzleştirme katmanı (flattening), TBK ve [0,1] arasında herhangi bir sınıfa ait olma olasılığı döndüren aktivasyon fonksiyonundan (Softmax) oluşur. ESA'nın tahminleme yaptığı son kısım normal bir YSA'dan farklı değildir.



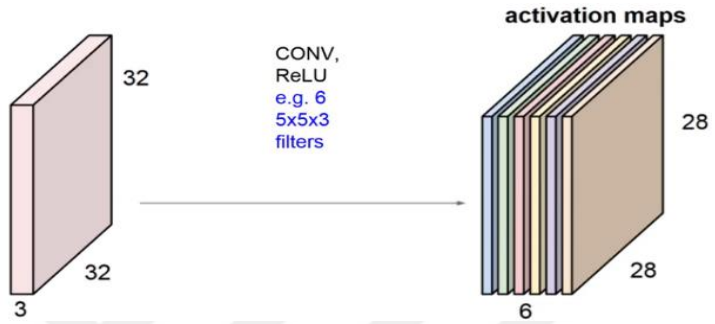
Şekil 2.20 Evrişimsel sinir ağı [38]

2.3.2.1. Evrişimsel sinir ağını oluşturan önemli parametreler

Evrişimsel katman (convolutional layer)

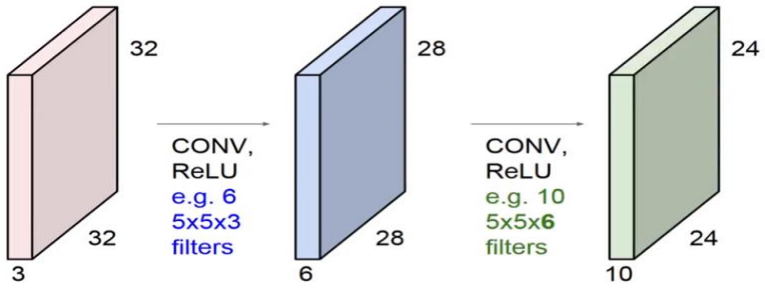
MÖ değil de DÖ yapıldığı için, MÖ deki gibi modele öznitelikler (features) verilmez. Model bu öznitelikleri bu katmanda kendisi çıkarır. Yani bu katman sayesinde görüntüler üzerine rastgele filtreler uygulanarak öznitelikler çıkartılır [38].

Şekil 2.19'da görüldüğü gibi 5x5x3 boyutundaki filtre, 32x32x3 boyutundaki resim üzerinde piksel piksel kaydırılır ve sonuç olarak 28x28x1 boyutunda bir öznitelik haritası oluşur. Bu sadece tek bir filtreden çıkan öznitelik haritasıdır. Katman (Layer) içerisinde kaç tane filtre varsa hepsi aynı şekilde uygulanır ve hepsi için birer öznitelik haritası çıkar. Ve elde edilen çıktının derinliği filtre sayısına eşit olur.



Şekil 2.21 32x32x3 boyutundaki resme 5x5x3 boyutunda 6 tane filtrenin uygulanması [31]

Şekil 2.21'e bakıldığında 32x32x3 resim üzerine 6 tane 5x5x3 lük filtre uygulanmış, 28x28x6 boyutunda bir çıktı alınmıştır (6 filtre uygulandığı için derinlik 6 dır). Uygulanan 6 tane 5x5x3 lük filtrenin tamamı 1 tane Evrişimsel Katman (CONV Layer) dır. ESA'larda da YSA'larda olduğu gibi birden fazla katman eklenebilir. Mesela ilk katmanın çıkışına 10 tane 5x5x6 boyutunda filtreden oluşan başka bir evrişimsel katman eklenebilir.

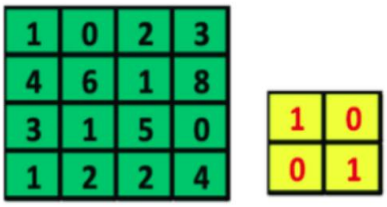


Şekil 2.22 Resmin 6 tane 5x5x3 lük filtrelerden oluşan ve10 tane 5x5x6 lık filtrelerden oluşan evrişimsel katmanlardan geçirilmesi [31]

Dikkat edilmelidir ki son eklenen, 10 filtreden oluşan evrişimsel katmanda bahsedilen filtrenin derinliği 6'dır (5x5x6). Çünkü filtrenin derinliği, uygulanacağı çıktının

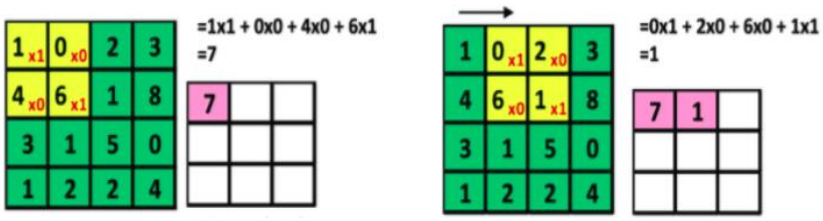
derinliği (28x28x6) ile ya da uygulanacağı katmanın (6 tane 5x5x3 boyutunda filtreden oluşan COVN layer) içerdiği filtre sayısı ile aynı olmak zorundadır. İstenirse 5x5 değiştirilip 2x2 veya 7x7 yapılabilir ancak derinlik değiştirilemez. Derinlik 6 olmak zorundadır. Yani sonuç olarak Şekil 2.22'de görüldüğü gibi 10 tane 5x5x6 lık filtrenin uygulandığı resmin derinliği 6, elde edilen çıktının derinliği de 10 dur.

Görüntülerin Filtreden Geçirilmesi:



Şekil 2.23 Girdi (input) olarak gelen resim ve filtre [32]

Gelen resimlerin (input) filtrelerden geçirilmesi sonucunda elde edilen değerler öznitelik haritasını oluşturur. Şekil 2.23'de 4x4'lük matris (resim) 2x2'lik filtre ile taranır.



Şekil 2.24 2x2'lik filtrenin 4x4'lük görüntü üzerinde kaydırılması ve 3x3 lük öznitelik haritasının (öznitelik matrisinin) oluşması [32]

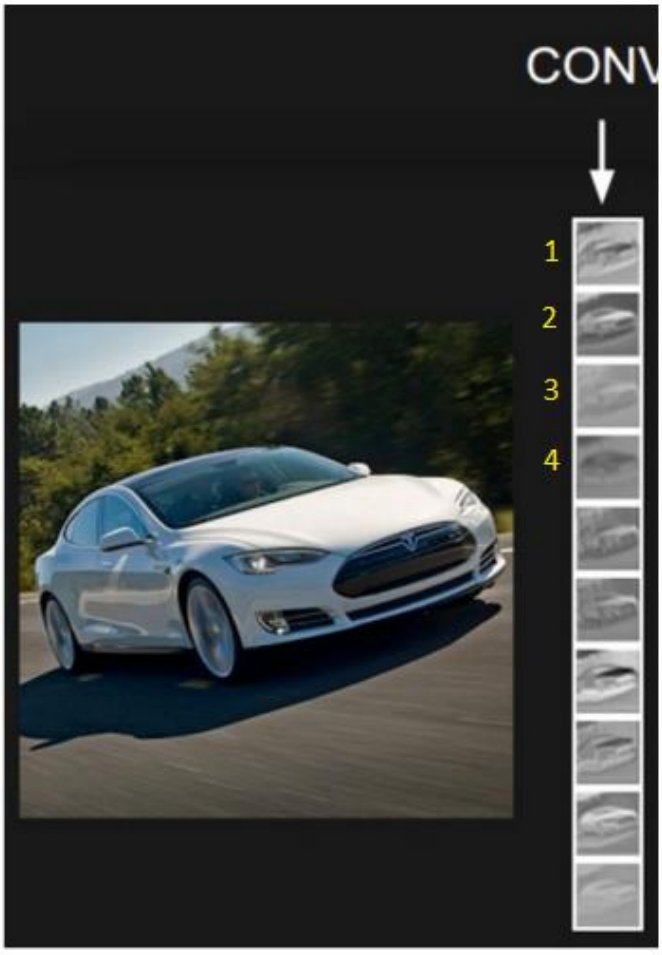
Şekil 2.24'de detaylı görüleceği üzere 2x2 lik filtrenin 4x4 lük matris üzerinde kayarken örtüştüğü her 2x2'lik matrisin piksel değerleri, filtre matrisinin değerleri ile çarpılır, toplanır ve elde edilen değerler öznitelik haritasını oluşturur. Birden fazla filtre olması durumunda ise bu işlemler her bir filtre için tekrar tekrar uygulanır.

Şekil 2.25'de, bir resmin orijinal hali (solda) ve filtre uygulanmış hali (sağda) görülmektedir. Filtre uygulanmış resimde bazı hatlar daha da belirginleşmiştir ki

öznitelik çıkarımı yapıldığında buna benzer görüntüler elde edilir [38].

Şekil 2.25 Orijinal resim (solda) ve filtre uygulanmış hali (sağda) [38]

Şekil 2.26'da ise bir evrişimsel katmanın çıktısı olarak 10 tane öznitelik haritası görülmektedir. Bu da resme 10 farklı filtre uygulandığı yani evrişimsel katmanın 10 filtreden oluştuğu anlamına gelmektedir. Öznitelik haritaları incelendiğinde ise her haritada resmin farklı bir bölümünüm netleştirildiği görülmektedir.

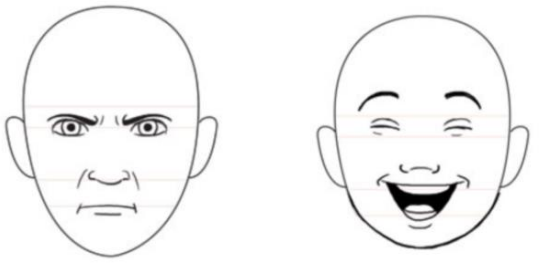


Şekil 2.26 Resme 10 kere evrişim işlemi uygulanması [38]

Görüntü işleme problemlerinde resimler genelde siyah-beyaz yapılıp modele girdi olarak verilir. Ancak bazı durumlarda renklerin etiket üzerindeki etkiler çok büyüktür (Bir meyve veya sebzenin kalitesinin rengiyle doğrudan alakalı olduğu durumlar gibi). Bu gibi durumlarda resimleri renkli şekilde işlemek zorunludur. Eğer modele renkli bir girdi verilirse yukarıdaki işlemler her bir renk katmanı için (RGB) ayrı ayrı yapılır.

Öznitelik Haritası (Öznitelik Matrisi):

ESA'lar, tıpkı insan beyninde olduğu gibi, verilen görüntüleri ayırt etmek için benzersiz özellikleri kullanır. Bu benzersiz özellikler aslında resimdeki özniteliklerdir ve evrişim işlemi uygulandıkça bu özniteliklere çok daha fazla odaklanılır. Örnek olarak insan yüzlerinden oluşan bir veri tabanındaki yüzleri 'mutlu' ve 'mutsuz' şekline sınıflandırmak istersek,



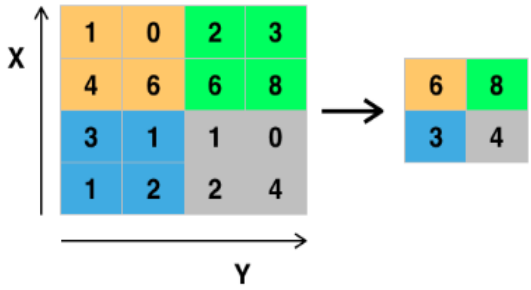
Şekil 2.27 Kızgın yüz - gülen yüz örnek resimler [32]

Şekil 2.27'de görüldüğü üzere sinir ağı, ağız, göz ve kaş çevrelerindeki matris farklılıklarının, etiket üzerinde büyük etkisinin olduğunu öğrenerek eğitimini buna göre tamamlayacaktır. Bununla beraber alın ve kulak bölgeleri her iki yüz resminde ve veri tabanındaki diğer yüz resimleri ile neredeyse aynı olduğu için sinir ağı bu bölgelerin eğitim ve tahmin üzerindeki etkisinin çok az olacağını veya tamamen etkisiz olacağını bilir, bu alanları hesaba katmaz. Tüm bu öznitelik haritası çıkarma işlemlerini sinir ağındaki evrişimsel katman yapar. Yapılan bu evrişim işlemleri birkaç kere uygulanır ve her uygulandığında objenin belli bir tarafı belirginleştirilmiş şekilde öznitelik haritası çıkartılır. Özetle evrişimsel katman ile resimlerdeki sadece gerekli ve önemli olan öznitelikler belirlenir. Farklılık oluşturmayan öznitelikler ise hesaba dahil edilmez.

Öznitelik seçme katmanı (pooling layer)

Bir ESA mimarisinde, genellikle ardışıl evrişim katmanlarından sonra birer öznitelik seçme katmanı periyodik olarak eklenir. Bu katman sayesinde, resmin temel özellikleri kaybedilmeden boyutu düşürülür böylece ağdaki parametrelerin ve hesaplamaların miktarları azaltılır. Dolayısıyla öğrenme hızı artar ve aşırı öğrenme durumunu kontrol etmek için de ağ boyutu kademeli olarak azaltılır.

Öznitelik seçme katmanı, oluşturulan öznitelik haritalarına uygulanır. Şekil 2.28'de 4x4 lük bir öznitelik haritasına 2x2 lik bir maksimum öznitelik seçme (max pooling) uygulanması sonucu 2x2 lik matris oluşur.



Şekil 2.28 4x4 lük bir öznitelik haritasına 2x2 lik max. pooling uygulanması [32]

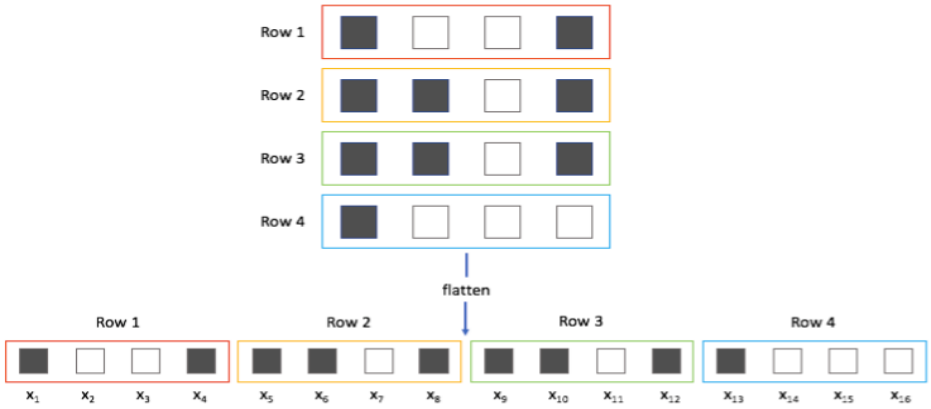
Öncelikle öznitelik seçme (pooling) işlemi için belli boyutlarda filtreler oluşturularak öznitelik haritası üzerinde kaydırılır ve filtre ile öznitelik haritasının örtüştüğü her matrisin en büyük değeri alınır. Şekil 2.28 incelendiğinde turuncu olarak gösterilen ilk matriste bu değer 6 dir, yeşilde ise 8 dir.

Genellikle kullanılan yöntem maksimum öznitelik seçme (max pooling) olsa da ortalama öznitelik seçme (mean pooling) ve minimum öznitelik seçme (min. pooling) gibi yöntemler de mevcuttur. Ortalama öznitelik seçme (mean pooling) yönteminde filtrenin örtüştüğü matrisin değerlerinin ortalaması alınırken, minimum öznitelik seçme (min. pooling) yönteminde ise en düşük değeri alınarak işlem gerçekleştirilir.

Düzleştirme katmanı (flattening):

Görüntü matrise çevrilip evrişimsel sinir ağına verildikten sonra birkaç evrişimsel

katman, aktivasyon fonksiyonu ve öznitelik seçme katmanından geçer ve en son elde edilen matris TBK'da (fully connected layer) kullanılmak üzere Şekil 2.29'daki gibi düzleştirilir.

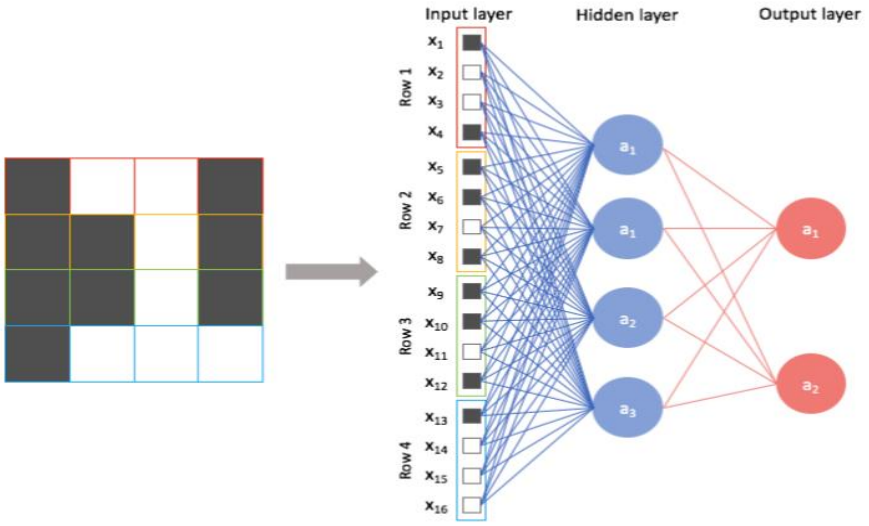


Şekil 2.29 Elde edilen matrisin tam bağlı katmana verilmeden önce düzleştirilmesi [38]

Düzleştirme işlemi gerçekleştikten sonra TBK'nın girdileri (x1, x2, x3, x4, x5, ...) hazırlanmış olur.

Tam bağlı katman (fully connected layer):

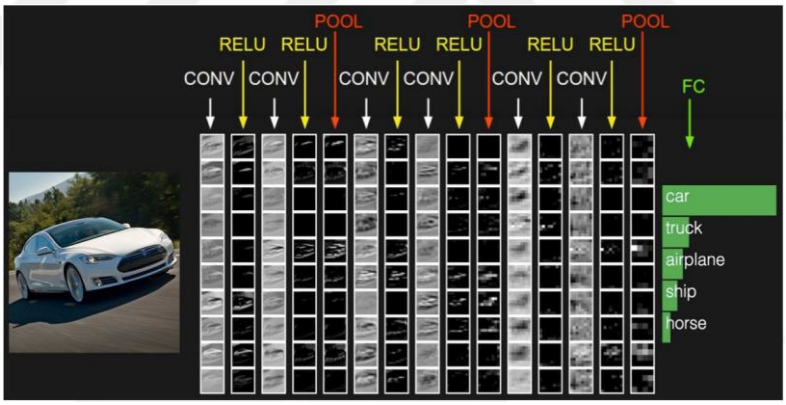
Bu katman sınıflandırmada kullanılan normal bir YSA ağı ile aynıdır.



Şekil 2.30 Tam bağlı katman ve düzleştirilen matrisin bu katmana verilmesi [38]

Şekil 2.30 Basit bir YSA yapısını yani TBK'nın basitleştirilmiş halini temsil etmektedir. Bu yapıyı anlamak TBK'nın çalışma prensibinin anlaşılmasına yardımcı olacaktır. Öncelikle x1, x2, x3, x4, x5, ... ile giriş katmanı (input layer), a1, a1, a2, a3 ile gizli katman (hidden layer) ve a1, a2 ile çıkış katmanı (output layer) oluşturulur. Giriş katmanından gelen değerler (x1, x2, x3, x4, x5, ...), gizli katmanda model tarafından belirlenen katsayılar (a1, a1, a2, a3) ile işleme girer. Bu işlem sonucunda belirlenen aktivasyon fonksiyonuna göre çıkış değeri (a1, a2) üretilir.

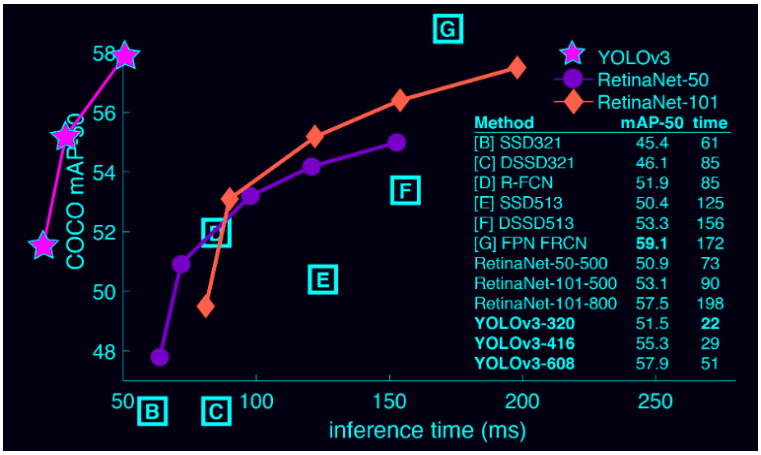
Şekil 2.31'de 6 tane evrişimsel katmandan oluşan bir ESA ve bu ağa verilen bir araba resmi görülmektedir. Resim ESA'ya verilirken önce matrise dönüştürülür ve sırası ile evrişimsel katmandan, aktivasyon fonksiyonundan ve model aralarına serpiştirilen öznitelik seçme katmanlarından birkaç defa geçirilir. Elde edilen öznitelik matrisi düzleştirilip vektör haline getirilir ve modelin tahmin yürütmesi için TBK'ya verilir, dönen sonuç ise resmin büyük oranda araba olduğunu ifade eder.



Şekil 2.31 Örnek bir evrişimsel model [38]

2.4 Sadece Bir Kez Bak (You Only Look Once - YOLO)

YOLO verilen görüntüyü tek seferde nöral ağdan geçirip resimdeki nesneleri, sınıfları ve konumlarıyla beraber oldukça hızlı bir şekilde ve tek seferde tespit edebilen bir Evrişimsel Sinir Ağıdır.

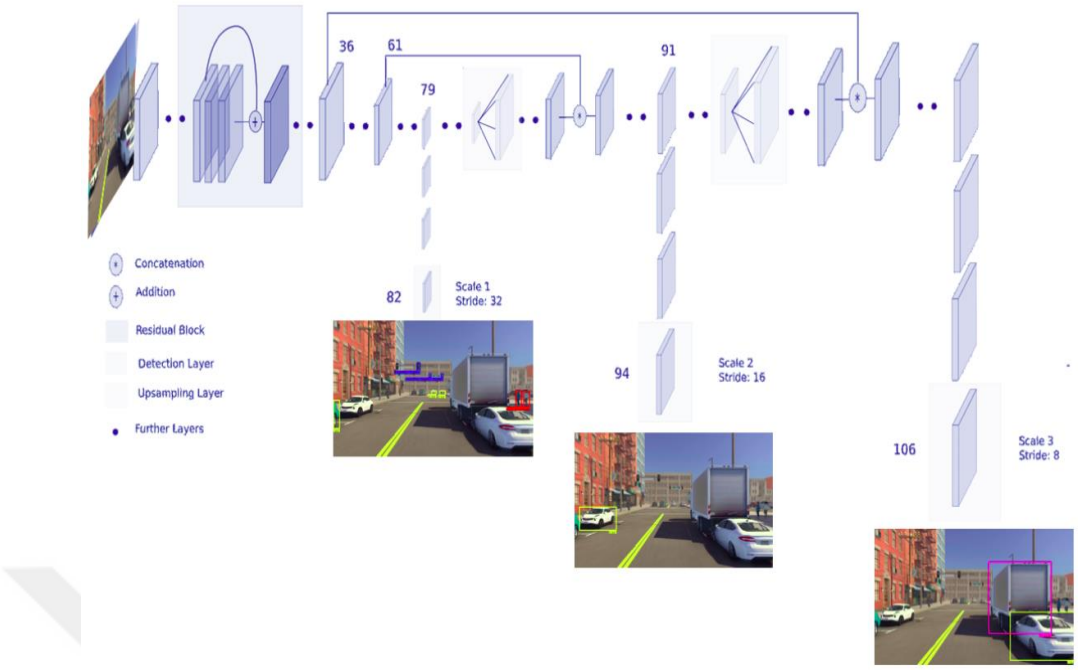


Şekil 2.32 YOLOv3 vs diğer algoritmalar [12]

Şekil 2.32'de YOLOv3 ve diğer algoritmaların COCO veri setinde 0.5 IoU (mAP-50) ile karşılaştırması görülmektedir. Grafikten de anlaşılacağı üzere YOLO rakiplerine karşı süre ve doğruluk açısından çok iyi durumdadır [39].

YOLO v3, C/Cuda kullanılarak geliştirilmiş olan, temel katman sayısı 53 olan Darknet-53 framework'ü üzerinde çalışır. Bu sayede oldukça yüksek performans gösterir [40]. Darknet-53, girdi olarak bir görüntü alan ve öznitelik haritasını çıkaran, YOLOv3 için bir omurgadır [41].

Şekil 2.33'de YOLOv3 algoritmasının mimarisi ve YOLOv3'ün üç ölçekte tahminler yaptığı gösterilmektedir. 13×13 katmanı (Bölge 82) büyük maske kullanan en büyük tahmin ölçeğidir ancak küçük nesneleri tahmin edebilir. 26×26 katmanı (Bölge 94) orta maske kullanan orta büyüklükteki tahmin ölçeğidir, orta büyüklükteki nesneleri tahmin edebilir. 52×52 katmanı (Bölge 106) en küçük tahmin ölçeğidir, daha küçük bir maske ile daha büyük nesneleri tahmin edebilir [41]. Yani Bölge 82, Bölge 94, Bölge 106, üç farklı ölçekte (82, 94, 106) tahmin edilen farklı büyüklükteki parametreleri temsil eder [42].



Şekil 2.33 YOLOv3 ağ mimarisi [41]

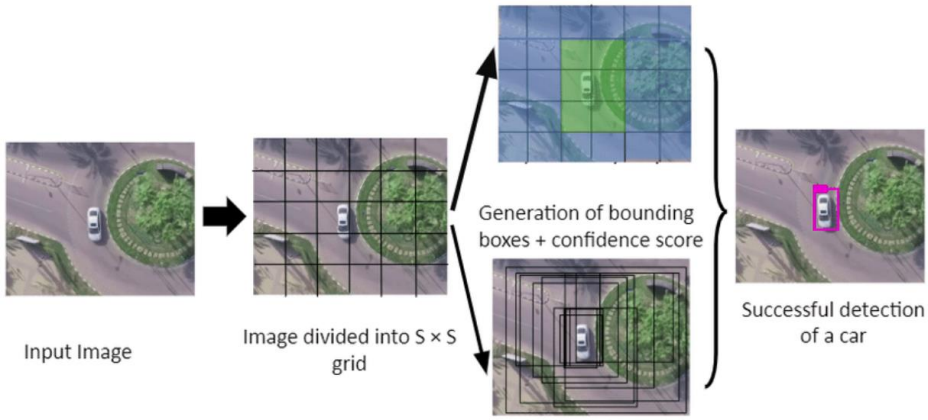
2.4.1. YOLOv3 çalışma prensibi:

Son yıllarda obje tespiti ve takibi konularında adından sıkça bahsettiren YOLO, evrişimsel katmanlardan oluşan bir algoritmadır. Süre, doğruluk ve hız bazında değerlendirildiğinde ise diğer ESA algoritmalarına karşı en çok tercih edilen algoritma olmuştur.

B-ESA ve B-ESA gibi bölge bazlı nesne tespit algoritmaları önce nesne bulunma ihtimali olan alanları belirler sonra da oralarda ayrı ayrı ESA sınıflandırıcıları yürütür. Bu yöntem her ne kadar iyi sonuçlar verse de iki ayrı işlem uygulanmış olur ve düşük FPS (Frame per second, saniye başına kare) alınmasına sebep olur. Daha Hızlı B-ESA algoritması bile gerçek zamanda ortalama 7 FPS ile çalışır [39].

YOLOv3 kendisine girdi olarak verilen resmi, görüntüyü ilk önce SxS'lik (3x3 5x5 19x19 vs.) hücrelere böler. Resim hücrelere bölündükten sonra her hücre kendi içerisinde nesne olup olmadığını, nesne var ise merkez noktasının kendi alanında olup olmadığını, merkez noktası kendi alanında ise nesnenin uzunluğunu, genişliğini ve hangi sınıftan olduğunu bulmakla sorumludur. Nesnenin merkez noktasına sahip olduğunu tespit eden her hücre o nesnenin sınıfını, yüksekliğini ve genişliğini bulup

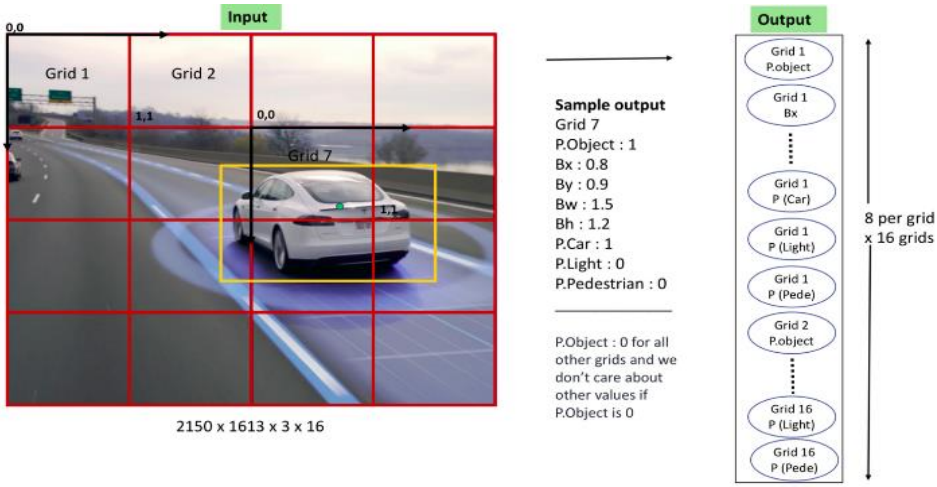
çevresine sınırlayıcı kutu (bounding box) çizer. Bahsedilen aşamalar Şekil 2.34'de gösterilmiştir.



Şekil 2.34 YOLOv3 tespit aşamaları [43]

Görüntü hücrelere bölünüp nöral ağdan geçirildikten sonra oluşan tahmin vektörü Şekil 2.35'deki gibi olur [44].

2.4.2 YOLO'nun tahmin vektörü:



Şekil 2.35 YOLO tahmin vektörü [45]

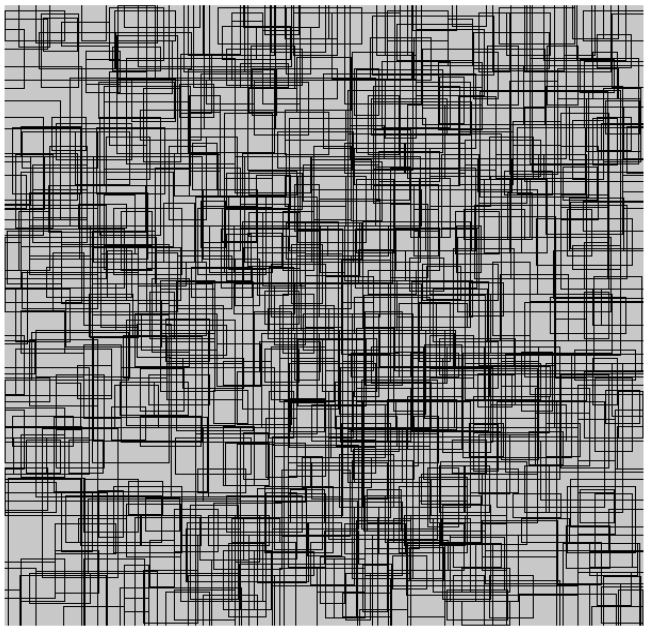
YOLO her hücre için ayrı bir tahmin vektörü oluşturur. Bunların her birinin içinde Güven Skoru, Bx, By, Bw, Bh ve Bağlı Sınıf Olasılığı vardır [46].

Güven Skoru (P.Object): Bu skor modelin, bahsi geçen hücre içerisinde nesne bulunup bulunmadığından, nesnenin gerçekten o nesne olup olmadığından ve sınırlayıcı kutunun (bounding box) koordinat değerlerinden emin olma derecesini gösterir [46].
Bx ve By nesnenin orta noktasının x ve y koordinatını,
Bw ve Bh: nesnenin genişliğini ve yüksekliğini gösterir.
Bağlı Sınıf Olasılığı: Kullanılan modelde sınıf sayısı kadar tahmin değeri gösterilir [46]. Şekil 2.35'deki modelde Car, Pedestrian ve Light olarak 3 sınıf için P.Car, P.Light, P.Pedestrian şeklinde ayrı ayrı tahmin değerleri gösterildiğini görürüz.

Örneğin Şekil 2.35'deki tahmin vektörü incelendiğinde toplamda 16 tane hücre olduğu (Bkz. Şekil 2.35 Input) ve her bir hücre için 1 tane tahmin değeri olmak üzere toplamda 16 tane tahmin değeri (Bkz. Şekil 2.35 Output) olduğu görülmektedir. Bunun yanında hücrelere ayrılmış resim ve resmin tahmin vektörü incelendiğinde 6. , 7. , 8. , 10. , 11. , 12. hücrelerde nesnenin farklı kısımlarına rastlandığını ama nesne merkezinin sadece 7. hücrede (Grid 7'de) olduğu, P.Object değerinin de sadece 7. hücrede 1 olduğu (P.Object=1), diğer hücrelerde 0 olduğu (P.Object=0) görülmektedir (Bkz. sample output). Buradan da modelin 7. Hücrede bir nesne bulduğundan emin olduğunu ve bulduğu nesnenin araba olduğundan (P.Car:1, P.Light=0, P.Pedestrian=0) ve arabanın koordinatlarının Bx=0.8, By=0.9, Bw=1.5, Bh=1.2 olduğundan emin olduğunu görüyoruz. Modelde 3 sınıf olduğu için 3 tane tahmin değeri olduğu görülmektedir. Bunlar; P.Car:1, P.Light:0, P.Pedestrian:0 dır. [46].

Sabitleme Kutusu (Anchor Box): Sabitleme Kutuları, gelişmiş performans için ayarlanması gereken en önemli parametrelerden biridir, kaliteli obje tespitinin anahtarıdır. Doğru ayarlanmadığı takdirde sinir ağı, küçük, büyük veya düzensiz nesnelerin var olduğunu bile bilemez ve onları algılama şansı asla olmaz. Mantığında ise, seçilmiş belli kalıplar (Anchor Box'lar) yardımı ile nesneyi çevreleyecek olan kutunun (Bounding Box'ın) tahmin edilmesi yatar.
Obje tespiti yapan, son teknoloji (state-of-art) obje tespit sistemleri, tahmininde uzmanlaştığı nesnenin ideal konumunu, şeklini ve boyutunu temsil eden binlerce "sabitleme kutusu" oluşturur. Tespit sırasında her sabitleme kutusu için IOU hesabı yapıp en yüksek örtüşmeye sahip olan sabitleme kutusunu bulur. Hesaplanan en yüksek IOU değeri %50'den büyükse, sabitleme kutusuna en yüksek IOU yu veren

nesneyi algılaması gerektiğini söyler. Eğer hesaplanan en yüksek IOU değeri %40 ile %49 arasında ise, sinir ağına gerçek tespitin belirsiz olduğunu ve bu örnekten ders almamasını söyler. Tüm bunların aksine, hesaplanan en yüksek IOU %40'tan azsa, sabitleme kutusuna nesne olmadığını tahmin etmesini söyler, yani bir nesne tahmini yapmamasını söyler [46].

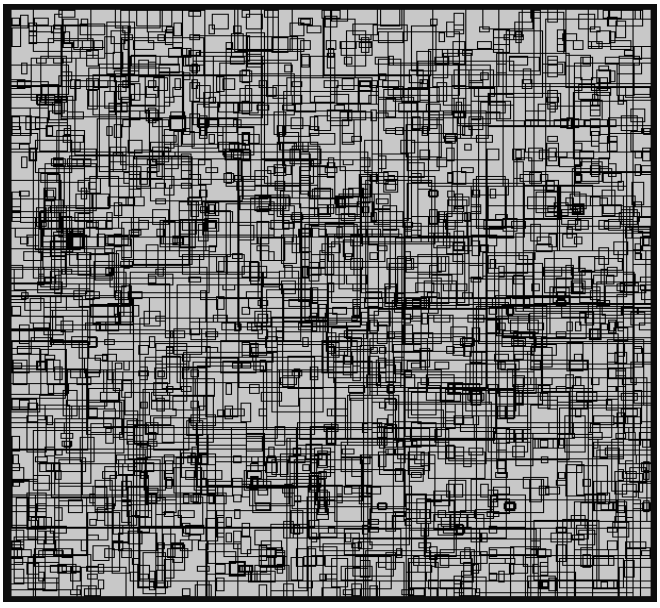


Şekil 2.36 RetinaNet sabitleme kutularının %1'i [46]

Son teknoloji nesne dedektörü olan RetinaNet'in açık kaynaklı bir uygulamasına göz atarak sabitleme kutularının sadece %1 i görselleştirilebilir (Şekil 2.36). Bu sabitleme kutularının uygulandığı görüntüdeki nesneler, sabitleme kutularıdan herhangi biri ile %50'lik bir IOU elde edemeyebilir. Dolayısıyla sinir ağı o nesneleri öğrenemez ve tahmin edemez. Mesela RetinaNet yapılandırmasında en küçük sabitleme kutusu boyutu 32x32'dir ve bu, 32x32 den daha küçük boyuttaki birçok nesnenin algılanmayacağı anlamına gelir [46]. Şekil 2.37'de RetinaNet'e girdi (input) olarak verilen bir resmin çıktısı (output) görülmektedir.

Şekil 2.37 RetinaNet'in 32x32 lik sabitleme kutuları ile yaptığı tahmin [46]

Şekil 2.36'dan anlaşılacağı üzere sınırlayıcı kutulardan (bounding box'lardan) yalnızca dördü, sabitleme kutularından herhangi biri ile örtüşür. Sinir ağı, diğer yüzleri tahmin etmeyi asla öğrenemez. Bu durumda varsayılan sabitleme kutusu yapılandırmalarını değiştirerek (Şekil 2.38), sabitleme kutuları çok daha küçük olacak şekilde ayarlanabilir.



Şekil 2.38 RetinaNet daha küçük sabitleme kutuları

Şekil 2.39 Daha küçük sınırlayıcı kutulara sahip retinaNet tahmini [46]

Böylece (Şekil 2.39) tüm yüzler sabitleme kutularından en az biri ile aynı hizaya gelir ve sinir ağı bunları algılamayı öğrenir [46].

Maksimum Olmayanı Önleme (Non-Maximum Suppression): Algoritma çalışırken gereksiz sınırlayıcı kutular oluşacaktır, hatta sadece bir nesne için birden fazla sınırlayıcı kutu çıkabilir [46].

Şekil 2.40 Maksimum olmayanı önleme (non-maximum suppression) [46]

Şöyle ki, (Şekil 2.40) birden fazla hücre, nesnenin merkezinin kendi içerisinde olduğunu tespit edebilir ve ekranda gereksiz sınırlayıcı kutular oluşturabilir. Bu durumda maksimum olmayanı önleme (non-maximum suppression) denen bir teknik uygulanır. Bilindiği üzere, çizilmiş olan her bir sınırlayıcı kutu için bir tahmin vektörü oluşturulur ve bu vektörlerin içinde güven skoru (P.Confidence veya P.Object) yer alır ve bu teknik ile de YOLO tarafından tahmin edilen güven skoru düşük olan sınırlayıcı

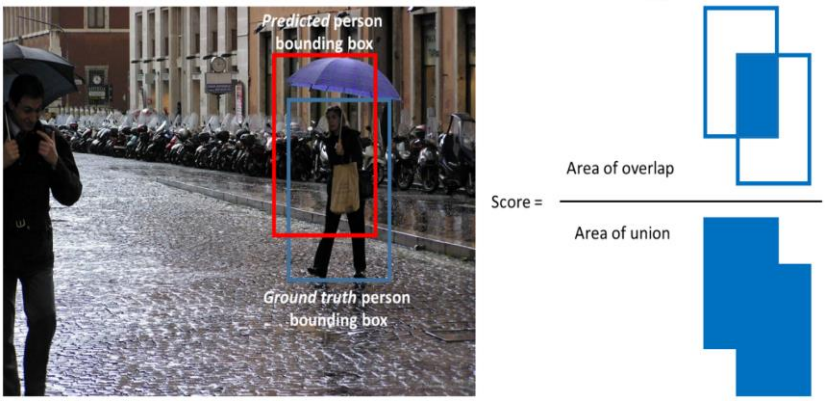
kutu görüntüden çıkarılır ve güven skoru en yüksek olan sınırlayıcı kutu görüntüde kalır [46].

2.5 Modelin Başarım Ölçütleri ve Hesaplanması:

Sınıflandırma çalışmalarında bir modelin başarısını ölçmek ve sayısal olarak ifade etmek için doğruluk (accuracy), kesinlik (precision), duyarlılık (recall) ve F1 score metrikleri kullanılır. Ancak, bu çalışmada objelerin sınıflandırılmasına ek olarak konumlarının bulunması da amaçlandığından, başarım ölçütü olarak genel ortalama kesinlik değeri (mean avarage precision - mAP) kullanılacaktır.

2.5.1 Kesin referans (Ground truth):

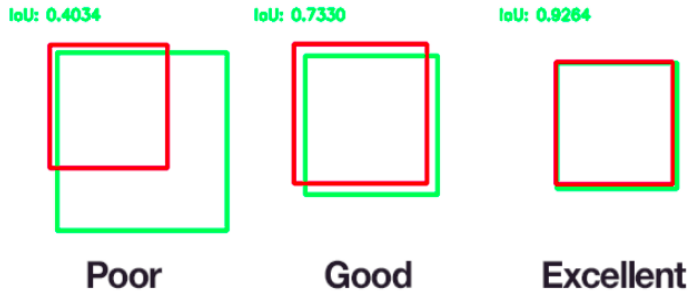
Nesne algılama çalışmalarında kullanılan modeller ile nesnelere yönelik bazı tahminler elde edilmeye çalışılır. Modellerin tahminleme başarısı ise bir test seti ile ölçülür. Test setinde yer alan, hedeflenen gerçek değerlere **kesin referans** denir. Bir nesne tanıma modelinin tahmini, kesin referans ve güven skoru (IoU) hesaplaması Şekil 2.41'de gözlemlenmektedir.



Şekil 2.41 Kesin referans ve güven skoru hesaplaması [47]

2.5.2 Güven skoru (Intersection over union - IoU):

Şekil 2.42'den anlaşılacağı üzere, YOLO yaptığı tahminin güvenilir veya güvenilmez olduğunu güven skoru ile ifade eder.



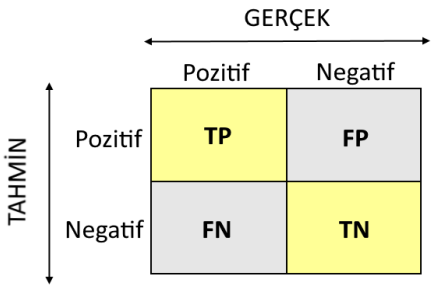
Şekil 2.42 IoU güven dereceleri [47]

Güven skoru, tahmin edilen sınırlayıcı kutu ile nesnenin gerçek alanının yani kesin referansının örtüşme oranı ile ölçülür, hesaplanır. Algılamanın doğru olup olmadığını yargılamak için kullanılan IoU aşağıdaki gibi hesaplanabilir [48];

$$\text{IoU} = \frac{Area(\mathbf{B}_p \cap \mathbf{B}_{gt})}{Area(\mathbf{B}_p \cup \mathbf{B}_{gt})} \tag{2.2}$$

2.5.3 Karşıtlık matrisi (Confusion matrix):

Sınıflandırma modellerinin performans değerlendirmelerinde, tahminlerin ve gerçek değerlerin karşılaştırmalarının yapıldığı Karşıtlık Matrisi sıklıkla kullanılmaktadır. Şekil 2.43'deki gibi karşıtlık matrisi bir sınıflandırma problemindeki gerçek değerleri ve tahmin edilen değerleri gösterir.



Şekil 2.43 Karşıtlık matrisi [49]

Kullanılan modelin 'yaya' sınıfına yönelik 'pozitif' tahminler yaptığı durum incelenirse:

TP (True Positive)

İlgili nesneye yönelik yapılan 'pozitif' tahminin 'doğru' olma durumunun sayısı.

FP (False Positive - Type 1 Error)

İlgili nesneye yönelik yapılan 'pozitif' tahminin 'yanlış' olma durumunun sayısı.

Modelin gördüğü obje gerçekte 'yaya' olsun. Modelin tahmini de 'yaya' ise bu True Positive dir. (Algılanan objenin 'yaya' sınıfında olmasının 'Positive' olduğuna yönelik yapılan tahminin doğru, 'True' olması.)
Modelin gördüğü obje gerçekte bir 'ağaç' olsun. Ancak modelin tahmini bu objenin 'yaya' olduğuna yönelik ise bu False Positive dir. (Algılanan objenin 'yaya' sınıfında olmasının 'Positive' olduğuna yönelik yapılan tahminin yanlış, 'False' olması.)

Kullanılan modelin 'yaya' sınıfına yönelik 'negatif' tahminler yaptığı durum incelenirse:

TN (True Negative)

İlgili nesnye yönelik yapılan 'negatif' tahminin 'doğru' olma durumunun sayısı.

FN (False Negative - Type 2 Error)

İlgili nesneye yönelik yapılan 'negatif' tahminin 'yanlış' olma durumunun sayısı.

Modelin gördüğü obje gerçekte bir 'ağaç' olsun. Modelin bu objeye yönelik tahmini 'yaya değil' yönünde ise bu True Negative dir. (Algılanan objenin 'yaya' sınıfında olmasının 'negative' olduğuna yönelik yapılan tahminin doğru, 'True' olması.)
Modelin gördüğü obje gerçekte bir 'yaya' olsun. Modelin bu objeye yönelik tahmini 'yaya değil' yönünde ise bu False Negative dir. (Algılanan objenin 'yaya' sınıfında olmasının 'negative' olduğuna yönelik yapılan tahminin yanlış, 'False' olması.)

2.5.4 Doğruluk (Accuracy), Kesinlik (Precision), Duyarlılık (Recall), F1-score

Doğruluk (Accuracy):

Bir modelin başarısını ölçmek için çok kullanılan ancak tek başına yeteli olmayan bir metriktir.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{2.3}$$

Kesinlik (Precision):

Modelin yaptığı "pozitif" tahminlerin, gerçekte kaçının "pozitif" olduğunu gösteren bir metriktir.

$$\text{Precision} = \frac{\mathbf{N}_{TP}}{\mathbf{N}_{TP} + \mathbf{N}_{FP}} \tag{2.4}$$

Duyarlılık (Recall):

Modelin yapması gereken "pozitif" tahminlerden, gerçekte kaçını pozitif tahminlediğini gösteren bir metriktir.

$$\text{Recall} = \frac{\mathbf{N}_{TP}}{\mathbf{N}_{TP} + \mathbf{N}_{FN}} \tag{2.5}$$

F1-score:

Eşit dağılmayan veri kümelerinde hatalı bir model seçimi yapmamak için doğruluk (accuracy) yerine tercih edilen bir metriktir.

İdeal bir modelde, duyarlılık (recall) değeri artarken kesinlik değeri de (precision) yüksek tutulmalıdır.

F1-skoru, kesinlik değerinin ve duyarlılık değerinin harmonik ortalaması alınarak hesaplanır.

$$\text{F1} - \text{score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.6}$$

2.5.5 Ortalama kesinlik değeri (Average precision)

Ortalama kesinlik değeri (AP), kesinlik-duyarlılık (precision-recall) eğrisi altında kalan alandır. Modelin tahminlerinin ne kadar doğru olduğunu ölçer yani doğru olan tahminlerin yüzdesel gösterimidir.

kesinlik-duyarlılık eğrisi, modelin güven eşiğinin (threshold) bir fonksiyonu olarak, modelin kesinlik (precision) ve duyarlılık (recall) değişimlerini çizme sürecidir.

$$\mathrm{AP} = \int_0^1 p(r)dr \tag{2.7}$$

2.5.6 Genel Ortalama kesinlik değeri (Mean average precision):

AP, her nesne sınıfı için hesaplanır. Sınıf sayısının birden fazla olması durumunda ise genel ortalama kesinlik değeri yani mAP değeri hesaplanır. Bu da tüm nesne sınıflarına ait AP değerinin ortalamasıdır.

mAP, modelin eğitim işleminin başarım ölçütüdür. Pratikte daha yüksek bir mAP değeri modelin daha iyi bir performans gösterdiğinin kanıtıdır. Model aslında mAP değerine göre değerlendirilir, mAP modelin hassasiyetinin iyi bir ölçüsüdür.

$$\mathrm{mAP} = \frac{\sum_{i=1}^{N} \mathrm{AP}_i}{N} \tag{2.8}$$

3. BULGULAR VE TARTIŞMA

Bu bölümde, DÖ yöntemi ile İHA'dan alınan video görüntülerindeki araçların ve yayaların tespiti üzerine yapılan çalışmanın aşamalarından ve elde edilen sonuçlardan bahsedilmiştir.

3.1 Veri Setinin Hazırlanması:

Drone'dan alınan görüntülerden biri olan FQQY4862.MP4 isimli video görüntüsü frame'lere ayrılır.

➔ *ffmpeg -i FQQY4862.MP4 -vf fps=1/5 drone3_%3d.jpg*

Videoların frame'lere ayrılması sonucu elde edilen resimlerin etiketleme işlemi için labelImg (Linux makineler için) kurulur ve başlatılır.

➔ *git clone https://github.com/tzutalin/labelImg.git*
➔ *cd labelImg*
➔ *sudo pip3 install -r requirements/requirements-linux-python3.txt*
➔ *make qt5py3*
➔ *cd labelImg-master*
➔ *python3 labelImg.py*

Etiketleme işlemi tamamlandıktan sonra etiketlenen resimlerin boyutları 608x608 olacak şekilde YOLO formatına uygun hale getirilir. Bu işlem için Şekil 3.1'deki kod bloğu yazılmıştır.

```
from cv2 import cv2
import os

folder = '/home/bildes/darknet/custom_data/images'
newfolder = '/home/bildes/darknet/custom_data/'

for filename in os.listdir(folder):
    img = cv2.imread(os.path.join(folder,filename))
    dim = (608,608)
    newImage = cv2.resize(img,dim,interpolation=cv2.INTER_AREA)
    newPath = newfolder + filename
    cv2.imwrite(newPath, newImage)
```

Şekil 3.1 resize.py dosyası

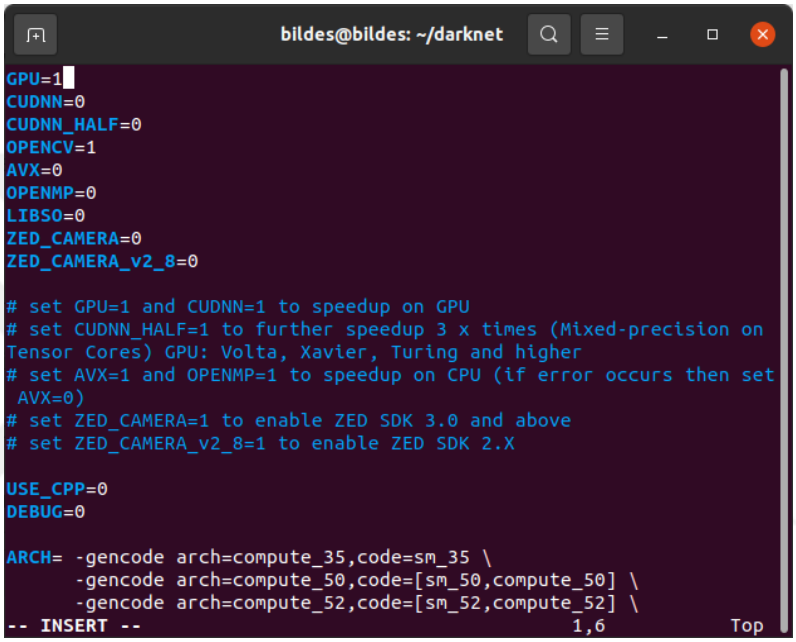
'images' adında bir klasör açılır, etiketlenen ve boyutları ayarlanan .jpg uzantılı resimler, .txt uzantılı etiketleri ile beraber bu klasöre atılır.

3.2 Darknet Kurulumu ve Konfigürasyonu:

Darknet klonlanır ve derlenir.

- ➔ *git clone https://github.com/pjreddie/darknet*
- ➔ *cd darknet*
- ➔ *make*

'make' komutu çalıştırıldıktan sonra 'Makefile' adında bir dosya oluşur. Bu dosya açılır ve Şekil 3.2'deki gibi GPU=1, OPENCV=1 şeklinde güncellenir.



Şekil 3.2 GPU ve OPENCV nin güncellenmesi

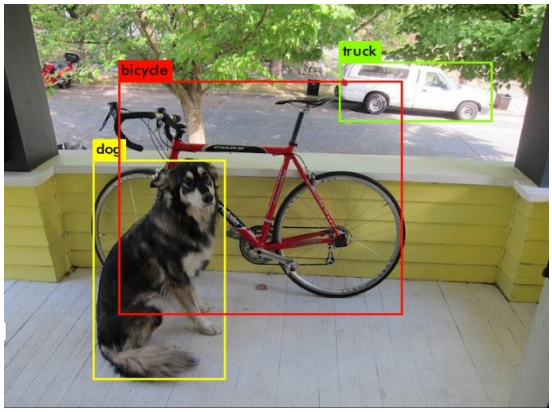
Darknet'in doğru bir şekilde kurulup derlendiğini test etmek için önceden eğitilmiş ağırlık (yolov3.weights) indirilir.

- ➔ *wget https://pjreddie.com/media/files/yolov3.weights*

İndirilen yolov3.weights ile darknet/data içerisindeki dog.jpg resmi üzerinde tespit işlemi yapılır

- ➔ *./darknet detect cfg/yolov3.cfg yolov3.weights data/dog.jpg*

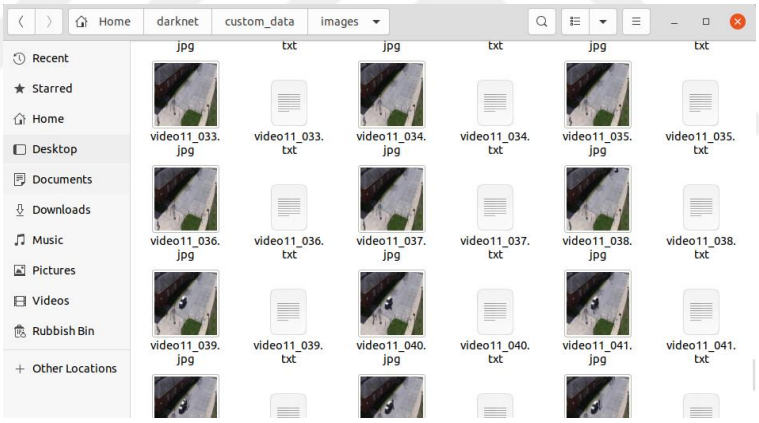
Şekil 3.3'deki resim derleme işleminin doğru yapıldığının bir kanıtıdır.



Şekil 3.3 Darknet derleme testi sonucu [12]

3.3. Eğitime Hazırlık - Eğitim ve Test İşlemleri:

Öncelikle ‘darknet’ klasörü içerisinde ‘custom_data’ adında bir klasör oluşturulur ve daha önce hazırlanmış ‘images’ klasörü Şekil 3.4’deki gibi ‘custom_data’ klasörünün içerisine atılır.



Şekil 3.4 images klasörüne atılan resimler ve anotasyonları

‘train.txt’ ve ‘test.txt’ adında iki dosya oluşturulur. Eğitim esnasında ‘images’ klasörü içerisindeki resimlere, bu iki dosyada yazılı olan dosya yolları (path’ler) ile gidilir. ‘train.txt’ içerisine eğitim için kullanılacak resimlerin yolları, tüm resimlerin %90 olacak şekilde rasgele seçilip yazılır. ‘test.txt’ içerisine de doğrulama için kullanılacak resimlerin yolları, tüm resimlerin %10’u olacak şekilde rasgele seçilip yazılır. Ve bu işlem Şekil 3.5’deki kod bloğu yardımı ile rahat bir şekilde yapılır.

```
import glob
import os
BASE_DIR = os.path.dirname(os.path.abspath(__file__))

current_dir = '/content/drive/MyDrive/darknet/custom_data/images'

# Create train.txt and valid.txt
file_train = open('train.txt', 'w')
file_test = open('test.txt', 'w')

counter = 1
percentage_test = 10;
index_test = round(100 / percentage_test)

for file in glob.iglob(os.path.join(current_dir, '*.jpg')):
    title, ext = os.path.splitext(os.path.basename(file))
    image_path = os.path.join(current_dir, '{}.jpg'.format(title))

    if counter == index_test:
        counter = 1
        file_test.write(image_path + "\n")
        # file_test.write(current_dir + "/" + title + '.jpg' + "\n")

    else:
        file_train.write(image_path + "\n")
        # file_train.write(current_dir + "/" + title + '.jpg' + "\n")
        counter = counter + 1
```

Şekil 3.5 split.py dosyası

Eğitim işlemi bilgisayarda yapılacak olsa

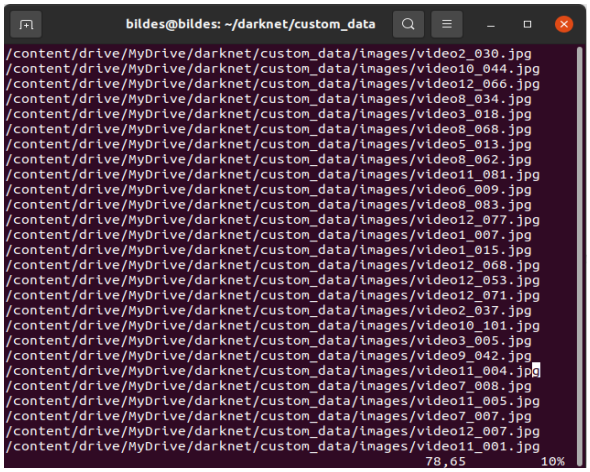
➔ `current_dir = os.path.join(BASE_DIR, 'images')`

kodu ile *current_dir = '/home/bildes/darknet/custom_data/images'* şeklinde ana yol olarak ‘images’ klasörünün bilgisayardaki yolu verilmiş olur. Ancak eğitim işlemi google colab’da yapılacağı için

➔ `current_dir = '/content/drive/MyDrive/darknet/custom_data/images'`

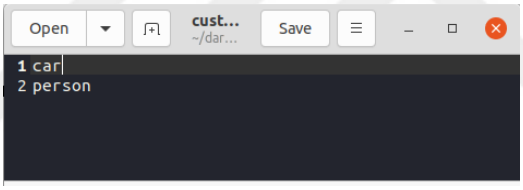
şeklinde bir yol verilir. Bu yol ileriki adımlarda ‘darknet’ klasörü drive’a atılıp colab’dan drive’daki darknet klasörüne erişim sağlanacağı zaman, ‘images’ klasöründeki resimlere erişim sağlanılabilecek yoldur. Şekil 3.5’deki kod bloğunun çalıştırılması sonucu oluşacak olan ‘train.txt’ ve ‘test.txt’ dosyasının örnek görünümü Şekil 3.6’da verilmiştir.

Oluşan ‘train.txt’ ve ‘test.txt’ dosyaları da ‘custom_data’ klasörü içerisine atılır.



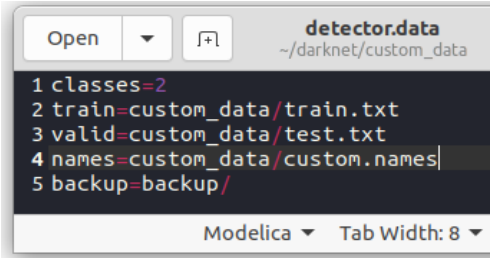
Şekil 3.6 train.txt veya test.txt örnek görüntüsü

Şekil 3.7'deki gibi 'custom_data' klasörü içerisine 'custom.names' adında bir dosya açılır, içerisine sınıf isimleri yazılır.



Şekil 3.7 custom.name dosyası

Yine 'custom_data' klasörü içerisine Şekil 3.8'deki gibi, 'detector_data' adında bir dosya oluşturulur, içerisine eğitim sırasında kullanılacak gerekli bilgiler yazılır.



Şekil 3.8 detector.data dosyası

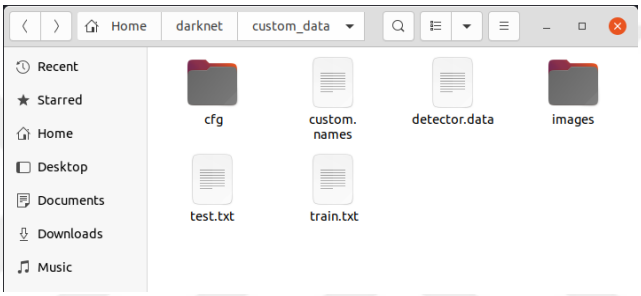
classes: Sınıf sayısı,

train: Eğitim için kullanılacak resimlerin yani 'train.txt' nin yolu,

valid: Doğrulama olarak kullanılacak resimlerin yani 'test.txt' nin yolu,

names: Sınıf isimlerinin yazıldığı dosyanın yolu,
backup: Eğitim esnasında yeni oluşacak olan ağırlıkların kaydedileceği dosyanın yoludur.

'darknet' klasörü içerisindeki 'cfg' adlı konfigürasyon klasöründeki 'yolov3.cfg' adlı konfigürasyon dosyası (*darknet/cfg/yolov3.cfg*) kopyalanır. 'custom_data' içeriside 'cfg' adlı bir klasör açılır. Kopyalanmış olan 'yolov3.cfg' konfigürasyon dosyası 'custom_data' içeriside 'cfg' klasörüne (*darknet/custom_data/cfg/yolov3.cfg*) yapıştırılır. Şekil 3.9'da 'custom_data' klasörünün son hali görülmektedir.



Şekil 3.9 custom_data son hali

3.3.1 yolov3.cfg dosyası konfigürasyon ayarları:

Eğitim işleminin başarılı ve efektif bir şekilde tamamlanması için *darknet/custom_data/cfg/yolov3.cfg* dosyasında gerekli ayarlamalar yapılmalıdır.

batch

Eğitim sırasında her iterasyonda alınması istenen resim sayısıdır. Varsayılan değeri 64 tür ancak bu çalışmada batch=32 olarak ayarlandı.

learning_rate

Öğrenme oranı ya da hızı, parametrelerin optimum değere hareket etme hızını belirler. Daha iyi bir öğrenme oranı ayarlamak sürekli deneme gerektirir. Buna karşın bu değeri başlangıçta yüksek tutup belli iterasyonlarda belli oranlarda küçültmek en uygun yaklaşım olacaktır. Örneğin YOLO eğitimi 160 tur (epoch) da tamamlanır, ilk öğrenme oranı 0.001'dir. 60. ve 90. tur da öğrenme oranı 10'a bölünür ve eğitime öyle devam edilir.

momentum

Eğitim boyunca ulaşılmak istenen hedeflerden biri loss değerini sıfıra yaklaştırmaktır ve bunun için kullanılan optimizasyon yöntemi Gradyan iniş (Gradient descent) yöntemidir. Bu yöntemin en önemli parametrelerinden biri öğrenme oranı diğeri ise momentumdur. Gradyan iniş çok fazla salınım oluşturan, gürültü üreten yöntemlerden biridir ve kararlı hale getirmek için momentum katsayısı kullanılır. Yeni ağırlık değeri hesaplanırken, önceki ağırlık değerinin momentum katsayısı ile çarpılması ile elde edilen değer de hesaplamaya dahil edilir. Böylece kararlı ve hızlı bir yöntem oluşmuş olur. momentum değeri için 0.9 ile 0.95 arası değerler önerilir.

decay

Ağırlık azaltma (weight decay) aşırı öğrenmeyi engellemek için kullanılır. Aşırı öğrenmeden kaçınmak için her iterasyonda ağırlıklar küçük bir faktör ile azaltılır.

max_batches

Sinir ağının eğitilmesi gereken maksimum iterasyon sayısıdır. *sınıf_sayısı*2000* şeklinde hesaplanabilir. Dolayısı ile bu çalışmada ‘car’ ve ‘person’ olmak üzere iki sınıf olduğu için max_batches=4000 dir.

steps

max_batches’in %80 ve %90’ıdır. Dolayısı ile steps=3200,3600 dür.

[yolo] katmanlarında;

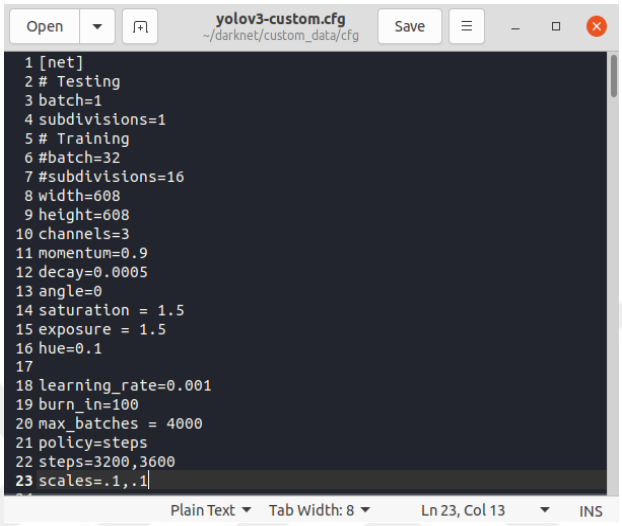
‘classes’ değişkenlerinin karşısına çalışmada kulanılan sınıf sayısı yani 2 girilir (610, 696, 783. satırlarda).

[yolo] dan hemen önceki [convolutional] katmanlarında;

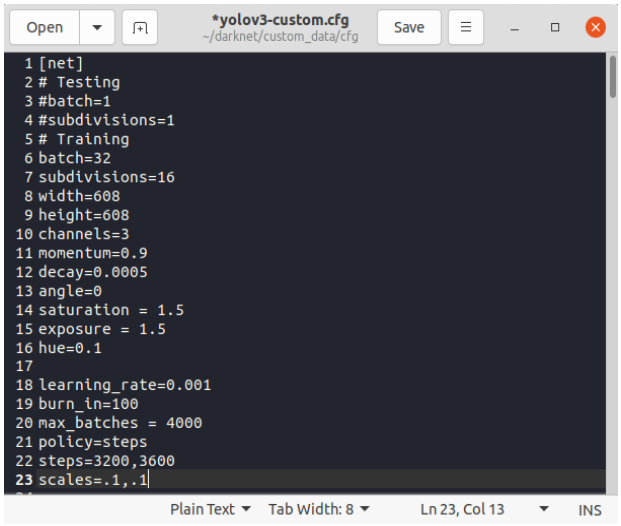
‘filters’ parametresi güncelenir. *filters = (classes + 5) * 3* formülü ile güncellenir ki bu çalışmada filters=21 dir (603, 689, 776. Satırlarda).

Değinilmesi gereken önemli bir diğer nokta;

Şekil 3.10 ve Şekil 3.11'de de gösterildiği gibi, eğitim için batch=32, subdivisions=16 olarak ayarlanmıştır. Ancak test yapılacağı zaman batch=1, subdivisions=1 şeklinde değiştirilmelidir.



Şekil 3.10 Eğitim sırasında batch, subdivisions değerleri



Şekil 3.11 Test sırasında batch, subdivisions değerleri

Ve son olarak 'darknet' kalsörü içerisine, eğitilecek olan ana ağırlık değerleri, "darknet53.conv.74" indirilir.

➔ *https://pjreddie.com/medi/files/darknet53.conv.74*

3.3.2 Google drive ve google colab ortamlarının hazırlanması

Eğitim için 'darknet' klasörü hazır. Eğitme işlemi google colab da yapılacağından bilgisayarda oluşturulan 'darknet' klasörü google drive' a atılır. Daha sonra google colab açılır ve google drive'ı google colab'a bağlamak için 'drive' API si kullanılır ve Şekil 3.12'deki gibi çalıştırılır.

```
from google.colab import drive
drive.mount('/content/drive', force_remount=True)
```

```
Mounted at /content/drive
```

Şekil 3.12 Google drive'ı google colab'a bağlama

- *from google.colab import drive*
- *drive.mount('/content/drive', force_remount=True)*

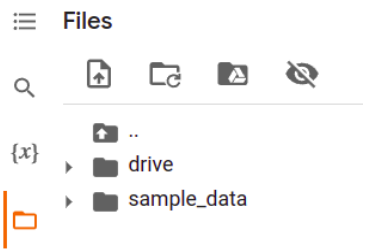
Daha sonra Şekil 3.13'deki kod ile çalıma dizini python, sys.path'e eklenir.

```
[5] import sys
    sys.path.append('/content/drive/MyDrive/Documents/Job documents/Xenlp/AR-nav/Detection/YOLOv3')
```

Şekil 3.13 Çalıma dizininin python sys.path'e eklenmesi

- *import sys*
- *sys.path.append('/content/drive/MyDrive/Documents/Job documents/Xenlp/AR-nav/Detection/YOLOv3')*

Artık google drive'daki 'darknet' klasörüne google colab üzerinden erişmek mümkün hale gelir. Sol panelde klasör ikonu tıklanınca Şekil 3.14'deki gibi 'drive' klasörü belirir. 'darknet' klasörü de 'drive' klasörü içerisinde yer alır.

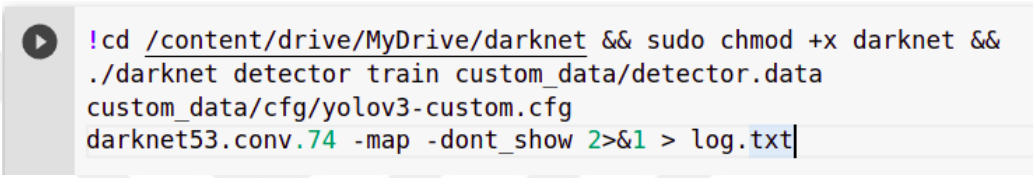


Şekil 3.14 Sol panel drive klasörü

3.3.3 Eğitim işlemi

Artık eğitim başlatılabilir. Bunun için Şekil 3.15'deki kod çalıştırılır.

- `!cd /content/drive/MyDrive/darknet`
- `sudo chmod +x darknet`
- `./darknet detector train custom_data/detector.data custom_data/cfg/yolov3-custom.cfg darknet53.conv.74 -map -dont_show 2>&1 > log.txt`



Şekil 3.15 Eğitim kodu

Eğitim sonrası başka bir veri seti ile ek bir eğitime ihtiyaç duyulduğu durumda, 'backup' klasöründeki eğitim esnasında oluşmuş olan 'yolov3-custom_last.weights' isimli ağırlık değerleri alınıp Şekil 3.16'deki gibi tekrar eğitim işlemi başlatılır

- `./darknet detector train custom_data/detector.data custom_data/cfg/yolov3-custom.cfg custom_data/yolov3-custom_last.weights -map -dont_show`

Şekil 3.16 Tekrar eğitim işlemi

Eğitim sonrası oluşan log.txt dosyası kullanılarak loss grafiği çizdirilebilir. Bunun için önce kullanılacak olan kod bloğu klonlanır.

- `git clone https://github.com/vovaekb/darknet_scripts.git`

Daha sonra loss grafiği çizdirilir.

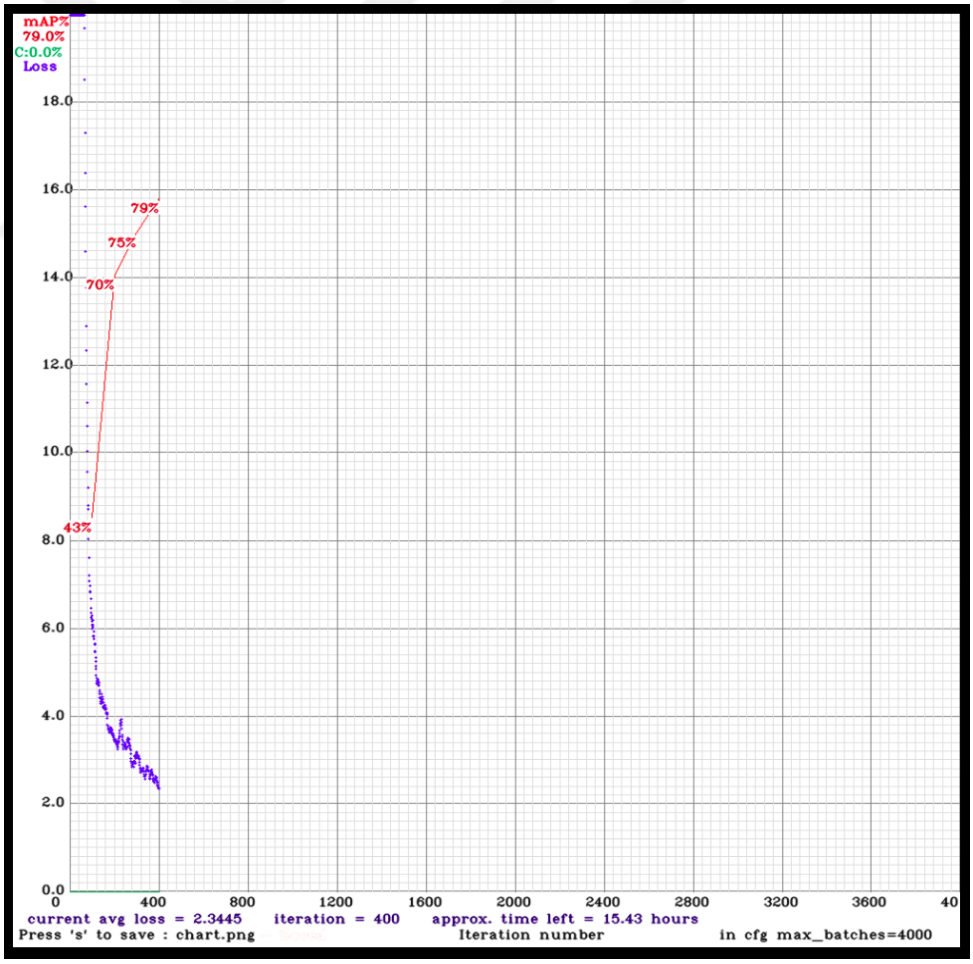
- `python plot_yolo_log.py ./darknet/log.txt`

Tablo 3.1'de eğitim ve yeniden eğitim süreçlerinde kullanılan veri seti sayıları ve eğitim sonrası elde edilen loss ve mAP değerleri görülmektedir.

Tablo 3.1 Eğitim - yeniden eğitim süreçlerinde kullanılan veri setleri ile eğitim sonrası elde edilen loss ve mAP değerleri

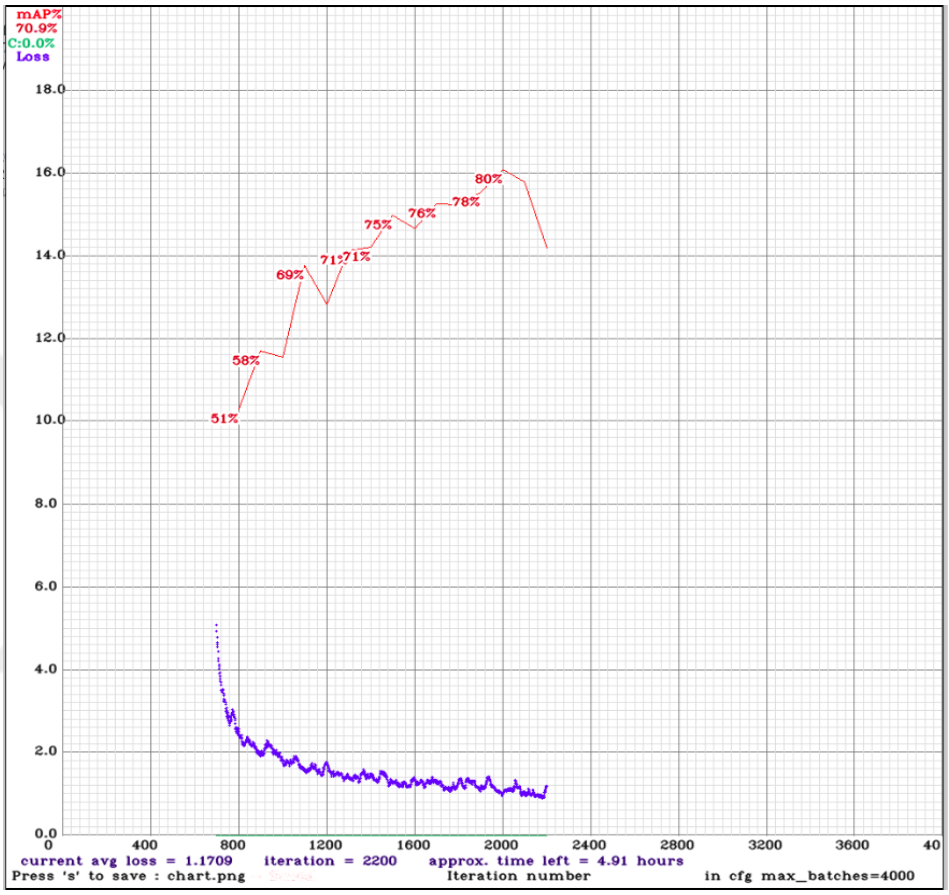
	Veri Seti	**Eğitim Seti**	**Doğrulama Seti**	**Loss Değeri**	**mAP Değeri (%)**
İlk Eğitim	500	450	50	2.345	79
Yeniden Eğitim	377	339	388	1.171	70.9

Şekil 3.17'daki mAP grafiğinde de görüldüğü gibi ilk eğitim 15.43 saat sürmüş, ortalama loss değeri 2.345 ve mAP değeri %79 olacak şekilde bir başarı elde edilmiştir.



Şekil 3.17 mAP grafiği

Ancak ilk eğitim sonrası oluşan ağırlığın yeniden eğitime tabi tutulması sonucu Şekil 3.18'deki grafik elde edilir. Grafikte görüldüğü üzere eğitim yaklaşık 5 saat sürmüş ve loss değeri 1'e yaklaşmıştır. Bununla beraber mAP değerinin önceki değere kıyasla %70.9 a düştüğü görülmektedir.



Şekil 3.18 Yeniden eğitim sonrası mAP grafiği

Tablo 3.2'de, elde edilen mAP sonuçlarının bu çalışma ile neredeyse birebir aynı olan, YOLOv3 DarkNet-53 kullanarak araç ve yaya tespiti yapan, "Drone ile Çekilmiş Videolarda Derin Öğrenme Tabanlı İnsan ve Araç Tespiti" adlı çalışma [16] ile karşılaştırma tablosu görülmektedir.

Tablo 3.2 Çalışma sonuçlarının 'Drone ile Çekilmiş Videolarda Derin Öğrenme Tabanlı İnsan ve Araç Tespiti' adlı çalışma [16] ile karşılaştırılması

Refer. No	Eğitim Seti	Öğrenme Katsayısı	Küme Boyutu	Loss Değeri	mAP Değeri (%)
İlk Eğitim	500	0.001	32	2.345	79

Yeniden Eğitim	377	0.001	32	1.171	70.9
[16]	1000	0.001	64	---	78.84
[16]	1000	0.001	128	---	77.23

3.3.4. Test işlemi:

Eğitim işlemi tamamlandıktan sonra Şekil 3.19'daki kod ile resim üzerinde, Şekil 3.20'deki kod ile de Video üzerinde test işlemini gerçekleştirilebilir.

- `!cd /content/drive/MyDrive/darknet`
- `sudo chmod +x darknet`
- `./darknet detector test custom_data/detector.data custom_data/cfg/yolov3-custom.cfg backup/yolov3-custom_last.weights custom_data/video13_012.jpg`

```
!cd /content/drive/MyDrive/darknet && sudo chmod +x darknet &&
./darknet detector test custom_data/detector.data custom_data/cfg/yolov3-custom.cfg
backup/yolov3-custom_last.weights custom_data/video13_012.jpg
```

Şekil 3.19 Resim için test kodu

- `!cd /content/drive/MyDrive/darknet`
- `sudo chmod +x darknet`
- `./darknet detector demo custom_data/detector.data custom_data/cfg/yolov3-custom.cfg backup/yolov3-custom_last.weights -thresh 0.20 -dont_show custom_data/video14.mp4 -out_filename VehPer_detection_20%.mp4`

```
!cd /content/drive/MyDrive/darknet && sudo chmod +x darknet &&
./darknet detector demo custom_data/detector.data
custom_data/cfg/yolov3-custom.cfg backup/yolov3-custom_last.weights
-thresh 0.20 -dont_show custom_data/video14.mp4 -out_filename VehPer_detection_20%.mp4
```

Şekil 3.20 Video için test kodu

Şekil 3.21'de yeniden eğitim sonrası oluşan ağırlıkların resim üzerindeki sonuçları görülmektedir. Şekil 3.22'de ise ilk eğitim sonrası oluşan ağırlıkların resim üzerindeki sonuçları görülmektedir.

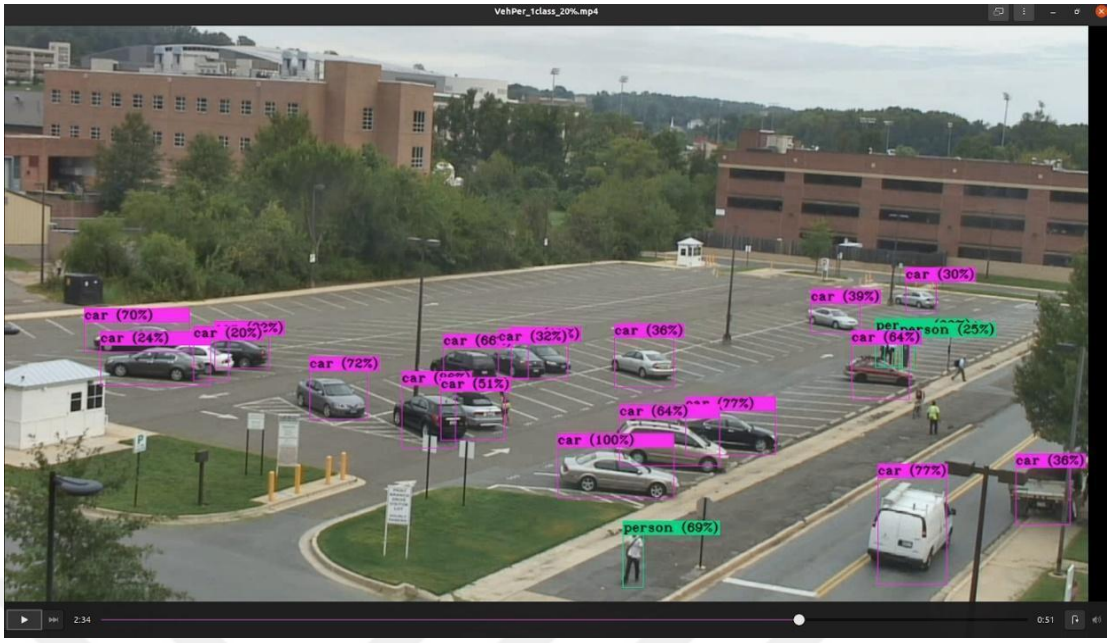
Şekil 3.21 Yeniden eğitim sonrası oluşan ağırlıkların resim üzerinde test edilmesi

Şekil 3.22 İlk eğitim sonrası oluşan ağırlıkların resim üzerinde test edilmesi

Bununla beraber Şekil 3.23 ve Şekil 3.24'de yeniden eğitim sonrası oluşan ağırlıkların video üzerindeki sonuçları görülmektedir. Şekil 3.25'de ise ilk eğitim sonrası oluşan ağırlıkların video üzerindeki sonuçları görülmektedir.

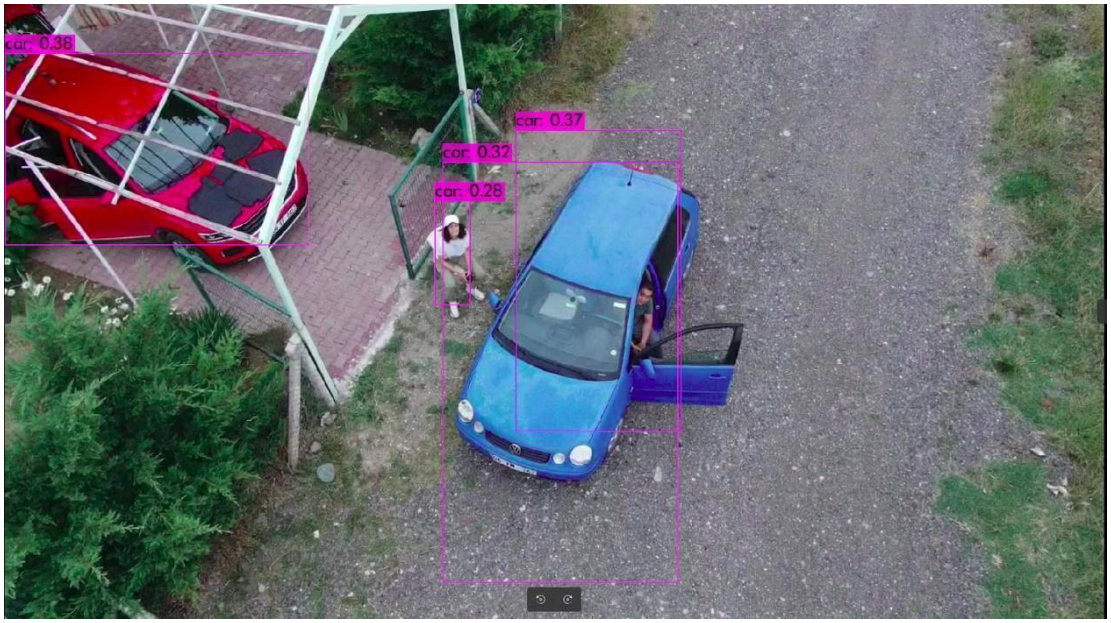
Şekil 3.23 Yeniden eğitim sonrası oluşan ağırlıkların video üzerinde test edilmesi

Şekil 3.24 Yeniden eğitim sonrası oluşan ağırlıkların video üzerinde test edilmesi



Şekil 3.25 İlk Eğitim sonrası oluşan ağırlıkların video üzerinde test edilmesi

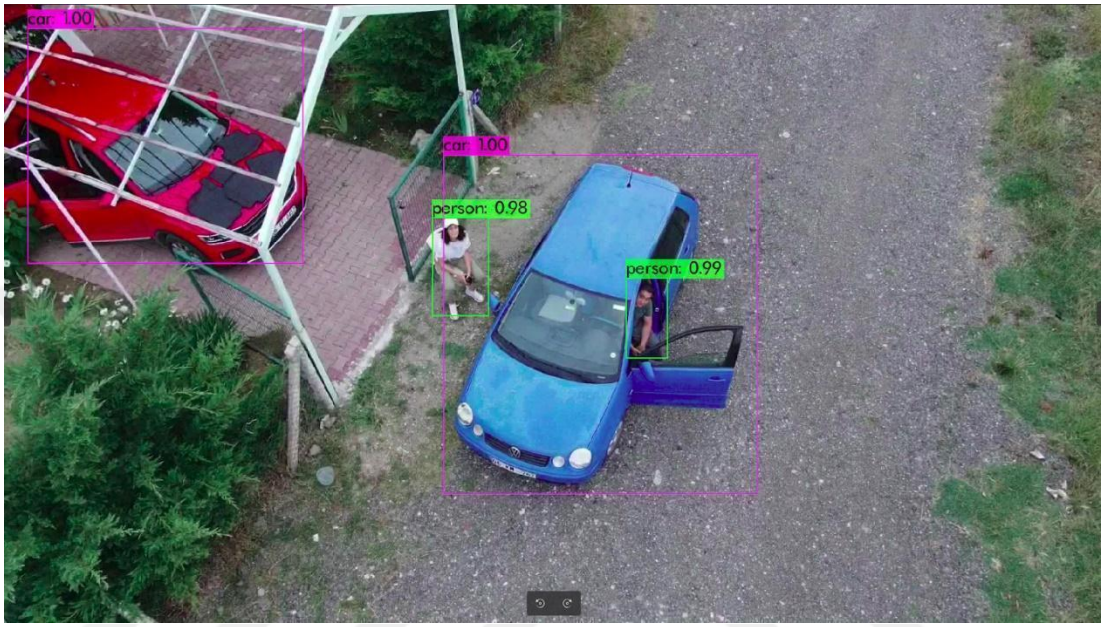
İlk eğitim sonrası oluşan ağırlıklar ile yeniden eğitim sonrası oluşan ağırlıklar aynı resim ile test edilmek istendiğinde ve resim olarak ilk eğitim setinde hiçbir şekilde benzeri olmayan, alışılmamış bir resim seçildiğinde, Şekil 3.26 ve Şekil 3.28'da testin resim üzerindeki sonucu, Şekil 3.27 ve Şekil 3.29'de ise testin süre ve yüzde bazında ağ çıktısı görülmektedir.



Şekil 3.26 İlk eğitim sonrası oluşan ağırlıkların resim üzerindeki tespitleri

```
custom_data/drone3_001.jpg: Predicted in 602.322000 milli-seconds.
car: 38%
car: 28%
car: 32%
car: 37%
```

Şekil 3.27 Şekil 3.26'deki tespitlerin süre ve yüzde bazlı ağ çıktısı



Şekil 3.28 Yeniden eğitim sonrası oluşan ağırlıkların resim üzerindeki tespitleri

```
custom_data/drone3_001.jpg: Predicted in 599.079000 milli-seconds.
car: 100%
person: 98%
car: 100%
person: 99%
```

Şekil 3.29 Şekil 3.28'daki tespitlerin süre ve yüzde bazlı ağ çıktısı

Tablo 3.3'de ise Şekil 3.26, Şekil 3.27, Şekil 3.28 ve Şekil 3.29'deki ilk eğitilen ağırlıklar ile yeniden eğitilen ağırlıkların seçilen resim üzerindeki tahmin sonuçlarının değerlendirmesi görülmektedir.

Tablo 3.3 Eğitim - yeniden eğitim sonrası Şekil 3.26, Şekil 3.27, Şekil 3.28 ve Şekil 3.29'deki tahmin sonuçlarının değerlendirme tablosu

	Tespit Süresi (ms)	**Araç Sınıfı Tahminleme Başarımı Ortalama (%)**	**Yaya Sınıfı Tahminleme Başarımı Ortalama (%)**
İlk Eğitim	602.3	34	0
Yeniden Eğitim	599.1	100	99

4. SONUÇLAR

Bu çalışmada, DÖ yöntemi ile İHA'dan alınan video görüntülerindeki araçların ve yayaların tespiti amaçlanmıştır.

İHA olarak DJI Mavic 2 Zoom Drone kullanılmıştır. Mavic 2 Zoom 1/2.3-inch 12-megapixel sensör ve 4x zoom ile güçlendirilmiş Zoom Kamera, 2x optik zoom (24–48 mm) lense sahiptir [2].

DÖ algoritması olarak ESA mimarilerini kullanan ve nesne tanıma mimarileri arasında en fazla öne çıkan Darknet-53 framework'ü üzerinde çalışan YOLOv3 algoritması kullanılmıştır.

Veri seti olarak Mavic 2 Zoom dan alınan video görüntülerden 377 resim, VIRAT [50] tarafından üretilen video datasetlerinden 500 resim, toplamda 877 resim kullanılmıştır. Video'yu frame'lere ayırmak için ffmpeg yazılımı kullanılmıştır.
Resimlerin etiketleme işlemi LabelImg [13] aracı ile yapılmış olup, %90'i eğitim, % 10'u doğrulama (test) seti olacak şekilde bir kod bloğu yardımı ile rassal olarak ayrılmıştır.

Algoritma Google Colab Tesla T4 GPU makinesi kullanılarak eğitilmiştir. Eğitim yaklaşık 16 saat sürmüş olup, ortalama loss değeri 2.345 ve mAP değeri %79 olacak şekilde bir başarım elde edilmiştir. Ancak ağı, ilk eğitim setinde hiçbir şekilde benzeri olmayan resimler (Şekil 3.21, Şekil 3.24, Şekil 3.26) ile eğitmeye kalktığımızda, Şekil 3.21'deki spor aletlerinin bir kısmını yaya bir kısmını da araç olarak tespit etmesi, Şekil 3.24'deki ağaçları yaya olarak tespit etmesi ve Şekil 3.26'daki oturan ve araç içerisindeki yayaları tespit edememesinden dolayı ilk eğitim sonucunda oluşan ağırlık değerleri yeniden eğitim işlemine tabi tutulmuştur. Eğitim yaklaşık 5 saat sürmüş, loss değeri içtendiği gibi azalıp 1'e yaklaşmıştır. Bununla beraber mAP değerinin önceki değere kıyasla %70.9 a düşmesi istenmeyen ve beklenmeyen bir durumdur.

Tablo 3.1'de yukarıda bahsi geçen eğitim ve yeniden eğitim sonrası elde edilen sonuçlar detaylı bir şekilde görülmektedir. Bununla beraber Tablo 3.2'deki [16] referans nolu çalışma ile bu çalışmanın değerler incelendiğinde, öğrenme

katsayılarının iki çalışmada da aynı olduğu ancak küme boyutu (batch) ile kullanılan veri setlerinin farklı olduğu, bu çalışmadaki veri setinin sayıca daha az olduğu görülmektedir. Bununla berber [16] referans nolu çalışmanın eğitildiği ortam ve loss değeri ile ilgili detaylar bilinmemektedir.

Tablo 3.2'deki mAP değerleri incelendiğinde ise bu çalışmanın ilk eğitimi sonrası elde edilen mAP değerinin, [16] referans nolu çalışmanın mAP değeri ile aynı olduğu görülmektedir. Ancak ilk eğitim sonrası elde edilen ağırlıkların detaylı testi sonucu, ileriki paragraflarda bahsedilecek olan bazı yanlış tespitler, araştırmacıyı yeniden eğitim işlemine itmiştir ancak elde edilen sonuç kısmen beklendiği gibi olmayıp mAP değeri %70.9'a gerilemiştir. Eğitim setinin az olması ve çalışma boyunca Google Colab'da yaşanan problem yani Google Colab'da başlatılan eğitimin tamamlanmadan sonlanması ise tespit edilen sebepler arasında gösterilebilir.

Şekil 3.21, Şekil 3.23, Şekil 3.24'daki yeniden eğitim sonrası oluşan ağırlıkların resim üzerindeki tahmin sonuçları ile Şekil 3.22 ve Şekil 3.25'deki ilk eğitim sonrası oluşan ağırlıkların resim üzerindeki tahmin sonuçları karşılaştırıldığında ise sonuçlar ilk bakışta her ne kadar benzer gibi görünse de ilk eğitime kıyasla ikinci eğitimdeki tahmin oranları daha da artmıştır. Bu, ilk eğitim sonrası oluşan ağırlıklar ile yeniden eğitim sonrası oluşan ağırlıkların aynı resim üzerindeki test sonuçlarında, Şekil 3.26, Şekil 3.27, Şekil 3.28 ve Şekil 3.29'da çok daha net görülmektedir.

Şekil 3.26 ve Şekil 3.28'de testin resim üzerindeki sonucu, Şekil 3.27 ve Şekil 3.29'da ise testin süre ve yüzde bazında ağ çıktısı görülmektedir. Tablo 3.3'de, ilk eğitim sonrası oluşan ağırlıkların ve yeniden eğitim sonrası oluşan ağırlıkların tespit süreleri ile araç ve yaya bazında tespit oranları görülmektedir. Test için kullanılan resimdeki 2 aracın ve 2 yaya'nın tahmin ortalama başarım yüzdeleri incelendiğin araç ve yaya başarım oranlarının ilk eğitimde %34 ve %0 iken yeniden eğitimde %100 ve %99 ile istenen sonuca erişmiştir. Burda da ilk eğitim setine çok daha farklı resimlerin eklenip, eğitim setinin daha da zenginleştirilmesi ve yeniden eğitim yapılması Şekil 3.28 ve Şekil 3.29'deki tahmin başarımının Şekil 3.26 ve Şekil 3.27'daki başarımdan yüksek olmasının nedenlerinden biri olarak gösterilebilir.

Sonuç olarak, veri seti sayısının arttırılması, konfigürasyon dosyasındaki parametrelerin daha iyi ayarlanması ve eğitim işleminin daha iyi bir makinede yapılması çok daha iyi mAP değeri ve loss değerine erişilmesine olanak tanır.

İleriki süreçte varılan sonuçlar üzerinden bahsi geçen çalışmanın iyileştirilmesi üzerinde ilerlenecek ve ek olarak tespitlerin gerçek zamanlı (real time) yapılması üzerine çalışılacaktır.

KAYNAKLAR

[1] O. Bayraktar , F. Özdemir , Ö. Çetin ve G. Yılmaz , "İnsansız Hava Araçları İçin Otonom İniş Sistemi Simülatörü Tasarımı", Bilişim Teknolojileri Dergisi, c. 5, sayı. 2, ss. 1-8, Mayıs 2012.

[2] MAVIC 2 Specs - Mavic 2 Zoom, DJI, Erişim Tarihi:12 Ocak 2022. https://www.dji.com/mavic-2?site=brandsite&from=insite_search

[3] İnsansız Hava Araçları sektörlerin geleceğini nasıl değiştirecek?, Forum Makine, Erişim Tarihi: 31 Ağustos 2022. http://www.forummakina.com.tr/tr/haberler/insansiz-hava-araclari-sektorlerin-gelecegini-nasil-degistirecek-

[4] Portable Fixed Wing Vtol UAV Drone Long Endurance Drone For inspection drone, Erişim Tarihi: 31 Ağustos 2022. alibaba.com/product-detail/Portable-Fixed-Wing-VTOL-UAV-DRONE_1600191395838.html

[5] R. D. Layman, Naval Aviation in the First World War: Its Impact and Influence, Naval Inst Pr, 1 November 1996

[6] J. William ve R. Taylor, Jane's pocket book of remotely piloted vehicles: Robot aircraft today, Collier Books, 1 Ocak 1977

[7] R. Kanyike, History of U.S. Drones, Understanding Empire: Technology Power Politics, 2012. Erişim Tarihi: 31 Ağustos 2022. https://understandingempire.wordpress.com/2-0-a-brief-history-of-u-s-drones/

[8] S. Dunstan, Israeli Fortifications of the October War 1973, Osprey Publishing, 18 November 2008

[9] Y. Azoulai, Unmanned combat vehicles shaping future warfare, L3Harris, Erişim Tarihi:31 Ağustos 2022. https://en.globes.co.il/en/article-1000691790

[10] M. Koç, "Derin öğrenme kullanarak iha ile hareketli bir hedefin otonom olarak yakalanması", Master's thesis, İstanbul Sabahattin Zaim Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Bilimleri ve Mühendisliği Anabilim Dalı, 2020.

[11] R. Bach, Martı Jonathan Livingston, Epsilon Yayınları, 2018

[12] YOLO: Real-Time Object Detection, Erişim tarihi: 7 Aralık 2021. https://pjreddie.com/darknet/yolo/

[13] LabelImg, Erişim tarihi: 7 Aralık 2021. https://github.com/tzutalin/labelImg

[14] P. Vladimir, Training YOLOv3 for detecting vehicles on video from scratch, 28 Ağustos 2020. Erişim tarihi: 7 Aralık 2021. https://becominghuman.ai/training-yolov3-for-detecting-vehicles-on-video-from-scratch-6403059ca0c2

[15] E. Bayhan, Z. Ozkan, M. Namdar ve A. Basgumus, "Deep Learning Based Object Detection and Recognition of Unmanned Aerial Vehicles", 2021 3rd International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), (pp. 1-5), June 2021.

[16] B. Bender, M. E. Atasoy ve F. Semiz, "Deep Learning-Based Human and Vehicle Detection in Drone Videos", 2021 6th International Conference on Computer Science and Engineering (UBMK), (pp. 446-450), September 2021.

[17] B. Uzun, O. Eker, H. Saribaş ve H. Çevikalp, "Detection Based Tracking of Unmanned Aerial Vehicles", 2019 27th Signal Processing and Communications Applications Conference (SIU), (pp. 1-4), April 2019.

[18] T. Tang, Z. Deng, S. Zhou, L. Lei ve H. Zou, "Fast vehicle detection in UAV images", 2017 International Workshop on Remote Sensing with Intelligent Processing (RSIP), (pp. 1-5), May 2017.

[19] H. Sarıbaş, H. Çevıkalp ve S. Kahvecıoğlu, "Car detection in images taken from unmanned aerial vehicles", 2018 26th Signal Processing and Communications Applications Conference (SIU), (pp. 1-4), May 2018.

[20] J. Lee, J. Wang, D. Crandall, S. Šabanović ve G. Fox, "Real-time, cloud-based object detection for unmanned aerial vehicles", 2017 First IEEE International Conference on Robotic Computing (IRC), (pp. 36-43), April 2017.

[21] Yapay Zeka-Artificial Intelligence-AI, Erişim tarihi: 17 Mart 2022. https://www.haydarcan.com/yapay-zeka-artificial-intelligence-ai/

[22] G. Luger and W. Stubblefield, "Artificial Intelligence : Structures and Strategies for ComplexProblem Solving", 5th Edition, The Benjamin/Cummings Publishing Company, Inc., 2004.

[23] N. J. Nilsson and N. J. Nilsson, "Artificial Intelligence:A New Synthesis", Morgan Kaufmann Publishers, 1998.

[24] S. J. Russell and P. Norvig, "Artificial Intelligence : A Modern Approach, Prentice-Hall", 1995

[25] A. Uğur, A. C. Kınacı, "İnternet Üzerinde Yapay Zeka ve Yapay Sinir Ağları", COMPOTEK 2005 Bilişim Seminerleri Programı, İzmir, Türkiye, Kasım 2005.

[26] E. V. Popov, "Artificial intelligence, Expert Systems and Natural Language Processing", Moskova: Radio i Svyaz, s. 461,1990.

[27] J. Copeland, "Artificial Intelligence: A Philosophical", Blackwell: Oxford, 1993.

[28] E. Charniak and D. mcDermot, "Introduction to Artifical Intellingence", Boston: Addison-Wesley Company, 1985.

[29] J. McCarthy, “Concepts of Logical AI16”, Semantic Scholar, 1996.

[30] KhanAcademy, Nöronun Anatomisi, Erişim Tarihi:31 Aralık 2021. https://tr.khanacademy.org/science/high-school-biology/hs-human-body-systems/hs-the-nervous-and-endocrine-systems/v/anatomy-of-a-neuron

[31] İ. Cebeci, R. Tuna, Yapay Zeka ve Derin Öğrenme A-Z™: Tensorflow, 2019. Erişim Tarihi: 10 Ocak 2022. https://www.udemy.com/course/yapayzeka/

[32] A. Akdogan, Uygulamalı Evrişimsel Sinir Ağları (Convolutional Neural Network), 2020. Erişim Tarihi:16.01.2022. https://medium.com/bilişim-hareketi/uygulamalı-evrişimsel-sinir-ağları-convolutional-neural-network-7d643eaeb6a7

[33] A. Bosch, A. Zisserman and X. Munoz, “Image classification using random forests and ferns”, 2007 IEEE 11th international conference on computer vision (pp. 1-8), 2007.

[34] D. Lenat, Cyc Project, 1984. Erişim Tarihi:31 Aralık 2021. https://stringfixer.com/tr/Opencyc

[35] H. Atasoy, Gradyan ve Gradyan İniş, 2015. Erişim Tarihi:12.01.2022. http://www.atasoyweb.net/Gradyan-Ve-Gradyan-Inis

[36] A Seyyarer, T. Aydın, “Değişmez Momentler Kullanarak İçerik Tabanlı Görüntü Erişim Sistemi ve İmge Sınıflandırma Yöntemlerinin Karşılaştırılması”, Anatolian Science - Bilgisayar Bilimleri Dergisi, ss. 1-9, 2017.

[37] E. Alpaydın, “Yapay Öğrenme”, Boğaziçi Üniversitesi Yayınevi, 2018.

[38] M. S. Konca, Evrişimli Sinir Ağları (Convolutional Neural Networks - CNN), 2020. Erişim tarihi: 2 Ağustos 2022. https://mustafaserdarkonca.medium.com/evrişimli-sinir-ağları-convolutional-neural-networks-cnn-74b3d4a567f9

[39] Y. Mesci, YOLO Algoritmasını Anlamak, 2019. Erişim tarihi: 7 Aralık 2021. https://medium.com/deep-learning-turkiye/yolo-algoritmasını-anlamak-290f2152808f

[40] H. V. Karakuş, Darknet YOLOv3 hızlı bakış, 2020. Erişim tarihi: 23 Kasım 2021. https://medium.com/@karakus.haciveli/darknet-yolov3-hızlı-bakış-ddc9cd5582ea

[41] S. Dulepet, P. Maji, M. Harsh and K. Washabaugh, Deploying a Scalable Object Detection Inference Pipeline Part, 2020. Erişim Tarihi:21 Aralık 2020. https://developer.nvidia.com/blog/deploying-a-scalable-object-detection-inference-pipeline/

[42] The meaning of output parameters during YOLO v3 training. Erişim Tarihi: 31 Ağustos 2022. https://blog.actorsfit.com/a?ID=00850-083895a9-5fda-4738-939b-1149afe66ca1

[43] A. Ammar, A. Koubaa, M. Ahmed and A. Saad, "Vehicle Detection from Aerial Images Using Deep Learning: A Comparative Study", Electronics, 2021.

[44] Yolo Nedir?, Odak Arge Merkezi, Erişim tarihi: 23 Kasım 2021. https://www.odakarge.com/yolo-nedir.html

[45] A. Frome, Y. Singer and J. Malik, "Image retrieval and classification using local distance functions", Advances in neural information processing systems, (pp. 417-424), 2006

[46] A. Christiansen, Anchor Boxes - The key to quality object detection, 2018. Erişim tarihi: 7 Aralık 2021. https://towardsdatascience.com/anchor-boxes-the-key-to-quality-object-detection-ddf9d612d4f9

[47] Yolo Nedir?, Görsel Analiz, 2020. Erişim Tarihi: 23 Kasım 2021. http://gorselanaliz.com/yolo-nedir/#page-content

[48] X. Nie, M. Yang and R. W. Liu, "Deep Neural Network-Based Robust Ship Detection Under Different Weather Conditions", 2019 IEEE Intelligent Transportation Systems Conference (ITSC), (pp. 47-52), 2019

[49] B. Bilen, Karışıklık Matrisi (Confusion Matrix), 2021. Erişim Tarihi: 18 Ağustos 2022. https://burhanbilen.medium.com/karışıklık-matrisi-confusion-matrix-990dfc718653

[50] The Virat Video Dataset, 11 Jan 2012. Erişim Tarihi: 18 Mayıs 2020. https://viratdata.org/

ÖZGEÇMİŞ

Kişisel Bilgiler		
Soyad, Ad		Usta, Ayşan
Eğitim Bilgileri		
Derece : 3.18	Kurum : Dicle Üniversitesi	Mezuniyet Yılı: 2018
Derece : 2.90	Kurum : Fırat Üniversitesi	Mezuniyet Yılı: 2012
Yüksek Lisans		Dicle Üniversitesi
Lisans		Dicle Üniversitesi Fırat Üniversitesi
Lise		Güler Şevki Özbek Anadolu Lisesi
İş Deneyimi		
Dönem (Yıl): 2019 - 2020	Şirket, Kurum: Xena Vision	Görev: AR-GE Mühendisi
Dönem (Yıl): 2020 - Halen	Şirket, Kurum: N2 Mobil	Görev: Yazılım Geliştirici
Yabancı Dil		
İngilizce		
Yayınlar		
1. INESEC DERİN ÖĞRENME KULLANARAK ARAÇ ALGILAMA: BİR KARŞILAŞTIRMALI ÇALIŞMA		

DİCLE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
TEZ BENZERLİK BİLDİRİMİ FORMU

Öğrencinin Adı, Soyadı	Ayşan, Usta		
Öğrenci No	18805004		
Ana Bilim Dalı	Elektrik Elektronik Mühendisliği Anabilim Dalı		
Program Türü	Proje ☐	Yüksek Lisans X	Doktora ☐
Tez Danışmanı (Ünvanı, Adı, Soyadı)	Dr. Öğr. Üyesi Muhammet Ali Arserim		
(Varsa) II. Tez Danışmanı (Ünvanı, Adı, Soyadı)			
Tez Başlığı	İnsansız Hava Aracından Çekilen Videolar Kullanılarak Derin Öğrenme Yaklaşımı ile Nesne Tespiti		
RAPOR BİLGİLERİ			
Raporlama Aşaması	Tez Savunma Sınavı Sonrası		
Sayfa Sayısı	88		
Raporlama Tarihi	08.09.2022		
Benzerlik Oranı (%)	%13		

Yukarıda bilgileri verilen tez çalışmamın toplam 88 sayfalık kısmına ilişkin, 08/09/2022 tarihinde şahsım/tez danışmanım tarafından Turnitin isimli intihal tespit programından aşağıda belirtilen filtrelemeler uygulanarak alınmış olan intihal raporuna göre, tezimin benzerlik oranı % 13 olarak tespit edilmiştir.

Uygulanan filtrelemeler:

☐ Başlangıç Bölümleri (Kabul ve Onay sayfası, Teşekkür sayfası, Özet/Abstract) hariç

☐ Kaynaklar hariç

☐ Alıntılar hariç/dâhil

☒ Diğer (Her Şey Dahil)

Tezimin benzerlik oranı, Dicle Üniversitesi Fen Bilimleri Enstitüsü İntihal Raporu Uygulama Esaslarında belirtilen üst sınır benzerlik oranını aşmamaktadır. Benzerlik oranım üst sınır benzerlik oranının altında olsa dahi aksinin tespit edilmesi durumunda her türlü yasal sorumluğu kabul ettiğimi ve hukuki sonuçlarına razı olduğumu bildirir, gereğini arz ederim.

Öğrencinin Adı, Soyadı: Ayşan, Usta
Tarih: 08.09.2022

Danışman Dr. Öğr. Üyesi M. Ali ARSERİM İmza:

Tarih:08.09.2022

Ana Bilim Dalı Başkanı Doç.Dr. Bilal GÜMÜŞ İmza:

Tarih:08.09.2022