T.C.

BİLECİK ŞEYH EDEBALİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

DERİN ÖĞRENME İLE İHA GÖRÜNTÜLERİNDEN NESNE TESPİTİNİN YAPILMASI

YÜKSEK LİSANS TEZİ

EMİR ALBAYRAK

TEZ DANIŞMANI

PROF.DR. UĞUR YÜZGEÇ

BİLECİK, 2021

10408563

T.C.

BİLECİK ŞEYH EDEBALİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

DERİN ÖĞRENME İLE İHA GÖRÜNTÜLERİNDEN NESNE TESPİTİNİN YAPILMASI

YÜKSEK LİSANS TEZİ

EMİR ALBAYRAK

TEZ DANIŞMANI

PROF.DR. UĞUR YÜZGEÇ

BİLECİK, 2021

10408563

BEYAN

"Derin Öğrenme ile İHA Görüntülerinden Nesne Tespitinin Yapılması" adlı yüksek lisans tezinin hazırlık ve yazımı sırasında bilimsel ahlak kurallarına uyduğumu, başkalarının eserlerinden yararlandığım bölümlerde bilimsel kurallara uygun olarak atıfta bulunduğumu, kullandığım verilerde herhangi bir tahrifat yapmadığımı, tezin herhangi bir kısmının Bilecik Şeyh Edebali Üniversitesi veya başka bir üniversitede başka bir tez çalışması olarak sunulmadığını beyan ederim.

<table>
<tr><td colspan="4">Bu çalışmanın,
Bilimsel Araştırmalar Projeleri (BAP), TÜBİTAK veya benzeri kuruluşlarca desteklenmesi durumunda; projenin ve destekleyen kurumun adı proje numarası ile birlikte beyan edilmelidir.</td></tr>
<tr><td colspan="2">DESTEK ALINMIŞTIR</td><td></td><td>DESTEK ALINMAMIŞTIR</td><td></td></tr>
<tr><td colspan="5">Destek alındı ise;</td></tr>
<tr><td colspan="5">Destekleyen Kurum:</td></tr>
<tr><td colspan="3">Desteğin Türü</td><td colspan="2">Proje Numarası</td></tr>
<tr><td colspan="2">1- BAP (Bilimsel Araştırma Projesi)</td><td></td><td colspan="2" rowspan="3"></td></tr>
<tr><td colspan="2">2- TÜBİTAK</td><td></td></tr>
<tr><td colspan="3">Diğer; .....</td></tr>
</table>

Emir ALBAYRAK

…../…../2021

İmza

ÖN SÖZ

Bu tez yeterlik çalışmasının hazırlanmasında, çalışmamı sahiplenerek takip eden danışmanım Sayın Prof. Dr. Uğur YÜZGEÇ ve Dr. Rıdvan YAYLA'ya değerli katkı ve emekleri için teşekkürlerimi ve saygılarımı sunarım.

Son olarak bugünlere ulaşmamdaki emekleri adına değerli aileme teşekkür ederim.

Emir ALBAYRAK

ÖZET

DERİN ÖĞRENME İLE İHA GÖRÜNTÜLERİNDEN NESNE TESPİTİNİN YAPILMASI

Günümüzde insansız hava araçlarından alınan görüntüler askeri ve ticari alanlar başta olmak üzere tarım, güvenlik, keşif alanlarında ve daha birçok alanda yaygın olarak kullanılmaktadır. Son yıllarda, yapay zekâ teknolojilerinin gelişimi ile birlikte görüntülerdeki nesne tespiti, doğru ve hızlı bir şekilde yapılabilmektedir. İnsansız hava araçlarında en çok tespit edilen nesne türleri, hareket eden nesneler ve araç görüntüleridir. Araç tespiti, askeri operasyonlarda tespit edilen hedefin takip edilip koordinatlarının öğrenilmesinde, üstü açık bir otoparktaki araç yoğunluğunun belirlenip otoparka gelen araçların boş yerlere yönlendirilmesinde, bir kavşaktaki trafik yoğunluğunun hesaplanmasında yaygın olarak kullanılmaktadır. Son yıllarda, derin öğrenme mimarilerinin gelişimi ile birlikte bir derin öğrenme mimarisi olan evrişimsel sinir ağları kullanılarak fotoğraf ve video görüntüleri üzerinden nesne tespiti, hızlı ve doğru bir şekilde yapılabilmektedir. Google tarafından geliştirilen ve bir derin öğrenme kütüphanesi olan Tensorflow kütüphanesindeki Faster R-CNN, Mask R-CNN gibi popüler modeller ile nesne tespitinde başarılı sonuçların alınabildiği gösterilmiştir. Evrişimsel sinir ağları temelinde oluşturulan nesne tespitine yönelik modeller, belli bir eğitim sürecinden geçtikten sonra test aşamasında önce nesne tahmini yapar ve daha sonra belirlenmiş nesneyi bir çerçeve içine alarak nesneyi tespit eder. Bununla birlikte daha ileri düzeydeki modeller, tespit edilen nesneyi ayrıca renklendirerek görüntünün arka plan dokusundan ayrılmasını sağlar. Mask R-CNN, örnek bölütleme özelliği sayesinde, bir görüntüde bulunan tek bir kategoriye ait birden fazla görüntüyü farklı renkler ile renklendirerek diğer algoritmalardan ayrılmaktadır. Diğer yandan YOLO yapısı ise görüntüye tek bir sinir ağı uygulayarak nesne tespit etmedeki hızı ile ön plana çıkmaktadır. Bu çalışma ile araç tespiti, birer derin öğrenme modeli olan YOLO ve Mask R-CNN modeli ile yapılmış olup bahsi geçen çözümler örnek veriler üzerinde kıyaslama (benchmark) testlerine tabi tutularak en etkili çözümün ortaya konulması amaçlanmıştır. Modellerin eğitiminde görüntüler, insansız hava aracı kullanılarak farklı yükseklik ve lokasyonlardan alınmıştır.

Anahtar Kelimeler: Derin öğrenme, Evrişimsel sinir ağları, İnsansız hava aracı, Nesne tespiti, Mask-R-CNN, YOLO.

ABSTRACT

OBJECT DETECTION FROM UAV IMAGES WITH DEEP LEARNING

Today, images taken from unmanned aerial vehicles are widely used in military, commercial fields, agriculture, security, and many more fields. In recent years, with the development of artificial intelligence technologies, object detection in images can be done accurately and quickly. The most frequently detected object types in unmanned aerial vehicles are moving objects and vehicle images. Vehicle detection is widely used in tracking and learning the coordinates of the target detected in military operations, in determining the density of vehicles in an open car park and directing the vehicles to empty parking places, and in calculating the traffic density at an intersection. It has shown that successful results can be obtained in object detection with popular models, such as Faster R-CNN, Mask R-CNN in Tensorflow library developed by Google. Models created on the basis of convolutional neural networks for object detection, firstly estimate the object in the test phase after training period, and then detect the object by enclosing the specified object in a frame. However, more advanced models let the image to be separated from the background texture by coloring the detected object. Mask R-CNN, thanks to its sample segmentation feature, differs from other algorithms by coloring multiple images with different colors, belonging to a single category in an image. On the other hand, the YOLO structure stands out with its speed in object detection by applying a single neural network to the image. In this study, vehicle detection has been done with YOLO and Mask R-CNN models, which are deep learning models, and it is aimed to reveal the most effective solution by subjecting the mentioned solutions to benchmark tests on sample data. In the training of the models, images were taken from different heights and locations using an unmanned aerial vehicle.

Keywords: Deep Learning, Convolutional Neural Networks, Unmanned air vehicle, Object Detection, Mask-R-CNN, YOLO.

İÇİNDEKİLER

SİMGELER VE KISALTMALAR LİSTESİ

CNN: Convolutional Neural Network (Evrişimsel Sinir Ağları)

R-CNN: Region Based Convolutional Neural Network (Bölgesel Evrişimsel Sinir Ağları)

Fast R-CNN: Fast Region Based Convolutional Neural Network (Hızlı Bölgesel Evrişimsel Sinir Ağları)

Faster R-CNN: Faster Region Based Convolutional Neural Network (Daha Hızlı Bölgesel Evrişimsel Sinir Ağları)

YOLO: You Only Look Once (Sadece Bir Kez Bakarsın)

Mask R-CNN: Mask Region Based Convolutional Neural Network (Maskelenmiş Hızlı Bölgesel Evrişimsel Sinir Ağları)

NHTSA: National Highway Traffic Safety Administration **(**Amerikan Ulusal Karayolu Trafiği Güvenliği İdaresi**)**

MLP: MultiLayer Perceptron (Çok Katmanlı Algılayıcı)

ReLU: Rectified Linear Unit (Doğrutulmuş Doğrusal Birim)

RPN: Region Proposal Network (Bölge Teklif Ağı)

VGG: Visual Geometry Group (Görsel Geometri Grubu)

ResNet: Residual Neural Network

ZFNet: Zeiler’s and Fergus’ Network (Zeiler ve Fergus’un Ağı)

TABLOLAR LİSTESİ

GRAFİKLER LİSTESİ

ŞEKİLLER LİSTESİ

1. GİRİŞ

Günümüzde derin öğrenme mimarilerinin geliştirilmesi ile nesne tespiti doğru ve hızlı bir şekilde yapılabilmektedir. Bu alanda yapılan sürdürülebilir çalışmalar sayesinde, son yıllarda nesne tespiti popüler bir alan haline gelmiş, pek çok alanda hayatımızı kolaylaştıran bir unsur olmuştur. Diğer yandan özellikle son beş yılda ülkemizde başarılı bir şekilde yürütülen çalışmalar sonucunda çeşitli türdeki insansız hava araçları (İHA) askeri alanda başarılı bir şekilde uygulanmaktadır (Korkmaz,2020). İHA görüntülerinde hareketli nesnelerin tespiti, özellikle silahlı insansız hava araçlarında (SİHA) hareket kabiliyetini artıran ve başarılı sonuçlar elde edilmesini sağlayan önemli bir faktördür. Kuşbakışı çekilen bir İHA görüntüsünde en fazla karşılaşılan hareketli nesneler araçlardır. Araç tespiti askeri alanların dışında araç güvenlik sistemleri ve trafik problemlerinin çözülmesi gibi benzer alanlarda da yaygın olarak kullanılmaktadır.

Yapay Sinir Ağları (YSA)'nın geliştirilmiş bir dalı olan derin öğrenme, nesne tespiti için son yıllarda Python, C ve C++ dili temelinde geliştirilen OpenCV, Tensorflow vb. kütüphaneler ile kullanılabilen bir makine öğrenmesi yöntemidir (Şeker vd., 2017: 47-64). Derin öğrenme temel olarak Multi Layer Perceptron (MLP) olarak adlandırılan çok katmalı algılayıcıda bulunan gizli katman sayısının çok fazla sayıda arttırılmış halidir. CNN Mimarisi temelinde geliştirilmiş YOLO Modeli ile R-CNN temelinde geliştirilen Fast R-CNN, Faster R-CNN ve son versiyon olarak Mask R-CNN modelleri nesne tespitinde tercih edilen, nesne tahmini yapabilen ve algılanan nesne etrafında bir çerçeve çizerek nesneyi belirleyen modellerdir. Bu modellerde kullanılan segmentasyon yöntemlerine ek olarak, Mask R-CNN modelinde algılanan nesne renklendirilerek nesnenin tamamının arka plan dokusundan ayrılmasını sağlar.

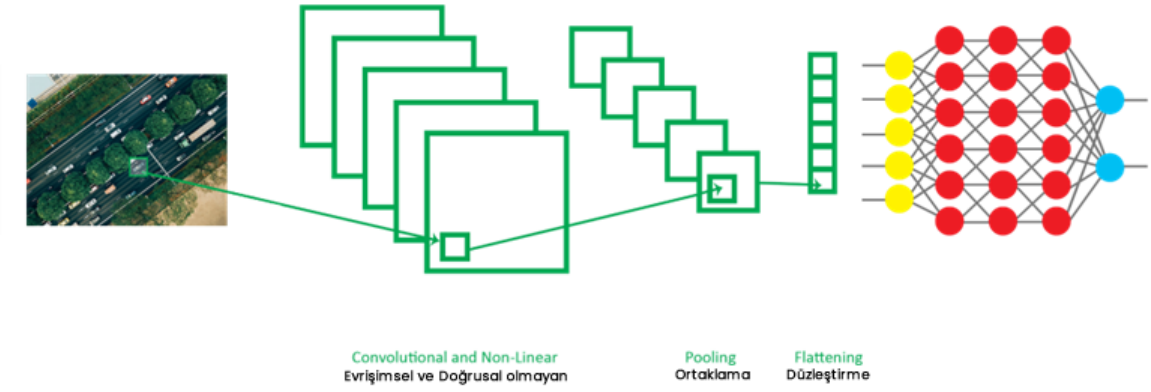
Yeni nesil araçlarda derin öğrenme temelinde kullanılan nesne tespit algoritmaları, şerit takip sistemi, kör nokta uyarı sistemi, yaya algılama sistemi, trafik levhası tanıma vb. sistemlerde kullanılmaktadır. Ayrıca Amerikan Ulusal Karayolu Trafiği Güvenliği İdaresi (NHTSA)'nin Seviye 0 ile Seviye 4 olarak sınıflandırdığı otonom araçlar, araca entegre edilen kameralar ile elde edilen görüntüleri işleyerek Seviye 4'te tam otonom araçlar olarak karşımıza çıkmaktadır (NHTSA, 2016: 4). Bununla birlikte bir kavşaktaki trafik yoğunluğunun hesaplanmasında görüntü işleme algoritmaları kullanılmaktadır (Ozdag, 2019: 58). Liu, Wang vd. insansız hava aracı (İHA) görüntülerinden küçük nesneleri YOLO algoritması ile tespit eden bir uygulama geliştirmiştir (Liu vd., 2020: 1-20). Vemula ve Frye,

Mask R-CNN algoritması yardımıyla İHA görüntüleri aracılığıyla bir bölgedeki elektrik hattı direklerini tespit eden bir çalışma gerçekleştirmiştir (Vemula vd., 2020: 1-6).

Bu çalışmada, İHA kamerası ile çekilmiş resim ve videolar üzerinden hareketli bir nesne olan araçların tespiti gerçekleştirilmiştir. Mask R-CNN modeli, bölge tabanlı segmentasyonda yaygın bir şekilde kullanılması ve tespit ettiği nesneleri örnek bölütleme özelliği sayesinde hem maskelemesi hem de çerçeve içine alması nedeniyle YOLO ise nesne tespit etmede hızı ile ön plana çıkması nedeniyle tercih edilmiştir.

2. EVRİŞİMSEL SİNİR AĞLARI (ESA)

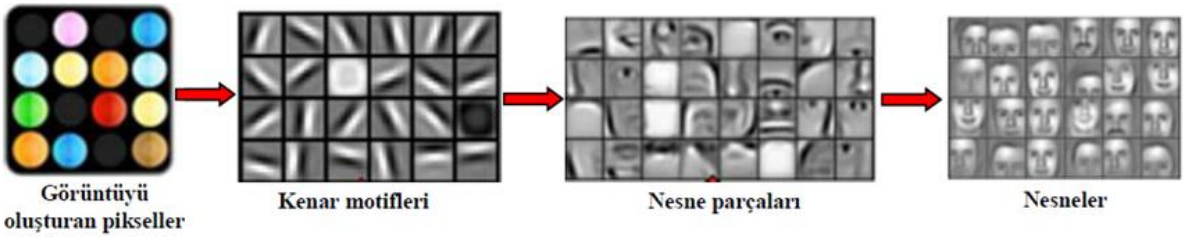
Literatürde Convolutional Neural Network (CNN) olarak isimlendirilen Evrişimsel Sinir Ağları (ESA) mimarisine görüntüler bir girdi olarak matris formatında verilir (Deshpande, 2019: 1). Bu mimariye göre ağda, Şekil 2.1'de gösterildiği gibi evrişim işlemi yapılır. ReLU işlemi ile nesne tahminleri arka plan dokusunda ayrılarak, ortaklama katmanlarından oluşan nesne tahminleri belirlenen eşik değerler ile son tahmin katmanı olan tam bağlantılı MLP'ye giriş verisi olarak sunulur (Prabhu, 2018: 1). Son aşamada da MLP'de ileri ve geri yayılım yapılarak hata oranı tespit edilip nesnelerin sınıflandırılması sağlanır (Saha, 2018: 1).



Şekil 2.1. ESA Mimarisinin Yapısı

Kaynak: (Rubik's Code, 2018)

Evrişimsel Sinir Ağları, Sobel, Prewitt ve Canny filtreleri gibi diğer resim öznitelik çıkarma yöntemlerine göre daha az ön işlem gerektirir. Evrişimsel sinir ağları farklı katmanlarda farklı temsiller oluşturur (Kızrak, 2018). Şekil 2.2'de görüldüğü üzere bir katmanda görüntüyü oluşturan pikseller üzerinde işlem yaparken bir katmanda kenar motifleri diğer bir katmanda ise nesne parçaları ile ilgilenir. Bu katmanlarda oluşturulan her yeni çıktıya öznitelik haritası, yapılan bu işleme ise öznitelik çıkarma işlemi denir.

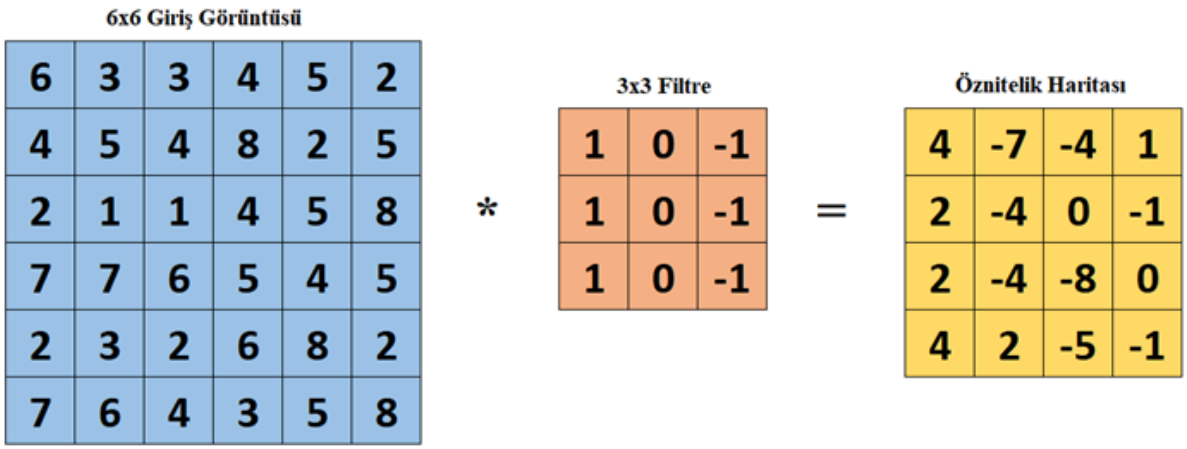


Şekil 2.2. ESA'nın Farklı Katmanlarda Nesne İle İlgili Oluşturduğu Farklı Temsiller

Kaynak: (İnik ve Ülker, 2017: 85-104)

2.1 Evrişim Katmanı

Konvolüsyon katmanı olarak da adlandırılan bu katmanda temel amaç giriş görüntüsünden öznitelik haritaları çıkarmaktır. Bu işlemi gerçekleştirirken giriş görüntüsü üzerine bir filtre uygulanır. Uygulanan filtre ile hem giriş görüntüsünün boyutu küçültülür hem de görüntüye ait öznitelik haritaları çıkartılmış olur. Bu haritalar bir katmanda görüntüdeki nesnelerin kenarları hakkında bilgi verirken bir diğer katmanda nesnelerin belirli bir parçasına odaklanabilir. Şekil 2.3'te 6x6 matris boyutundaki bir giriş görüntüsüne 3x3 boyutundaki filtre bir adım kaydırılarak uygulanmış ve elde edilen 4x4 matris boyutundaki öznitelik haritası gösterilmiştir. Bu katmanda kullanılan önemli hiper parametreler; giriş görüntüsünün matris boyutu, uygulanacak filtrenin matris boyutu ve filtrenin hareket ettiği adım sayısıdır (Amidi ve Amidi, 2019: 1).



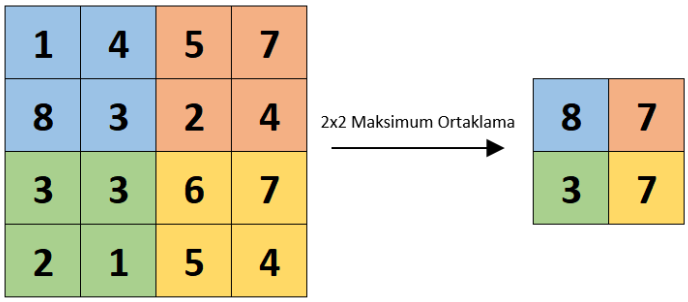
Şekil 2.3. Evrişim Katmanında Giriş Görüntüsüne Filtre Uygulanması.

2.2 Ortaklama Katmanı (Pooling)

Bu katman havuzlama katmanı olarak da adlandırılır. Burada temel amaç giriş görüntüsünü minimum kayıp ile daha küçük boyutlu bir hale getirmek ve görüntünün bilgisayar tarafından anlamlandırılmasını kolaylaştırmaktır. Literatürde birden fazla ortaklama işlemi mevcuttur (Sharma,2019). Bunlardan önde gelenleri Maksimum Ortaklama (Max Pooling), Ortalama Ortaklama (Average Pooling) ve Toplam Ortaklama (Sum Pooling)'dir.

2.2.1 Maksimum Ortaklama (Max Pooling)

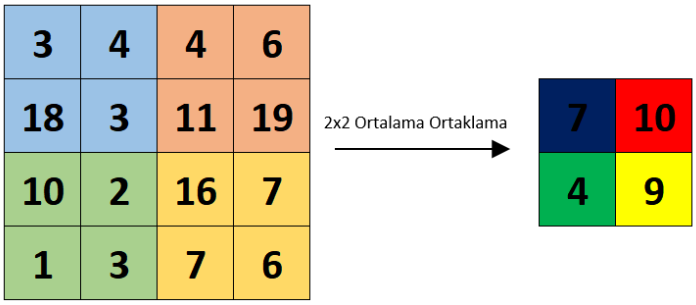
Maksimum ortaklama işleminde uygulanan filtrenin denk geldiği matris elemanlarının içinden en büyük olanı çıkış olarak belirlenir. Şekil 2.4'te 4x4 boyutundaki görüntü üzerinde 2x2 boyutundaki bir filtre iki adım kaydırılarak elde edilen çıktı gösterilmiştir. Görüldüğü üzere filtrenin denk geldiği elemanlar içerisinden en büyük değer çıkış olarak belirlenir. Bu sayede giriş görüntüsünün boyutu küçültülür ve bilgisayar üzerine yüklenen işlem boyutu yarı yarıya azaltılır.



Şekil 2.4. Maksimum Ortaklama İşlemi

2.2.2 Ortalama Ortaklama (Average Pooling)

Ortalama ortaklama işleminde uygulanan filtrenin denk geldiği matris elemanlarının aritmetik ortalaması alınarak elde edilen değer çıkış olarak belirlenir. Şekil 2.5'te 4x4 boyutundaki görüntü üzerinde 2x2 boyutundaki bir filtre iki adım kaydırılarak elde edilen çıktı gösterilmiştir. Görüldüğü üzere filtrenin denk geldiği elemanların aritmetik ortalaması alınarak elde edilen değer çıkış olarak belirlenir. Bu sayede giriş görüntüsünün boyutu küçültülür ve bilgisayar üzerine yüklenen işlem boyutu yarı yarıya azaltılır. Ortalama Ortaklama evrişimli sinir ağlarının ilk başarılı örneklerinden biri olan 1998 yılında yayınlanan Lenet sinir ağı modeli ile kullanılmaya başlanmıştır (Lecun vd., 1998).

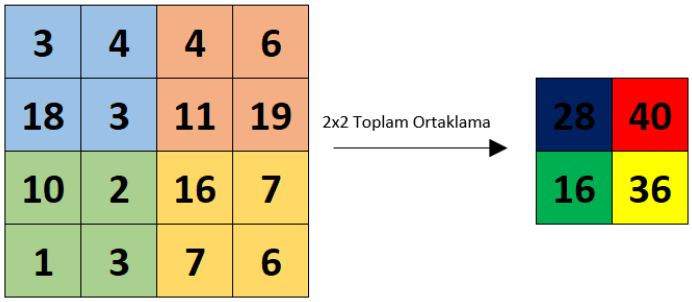


Şekil 2.5. Ortalama Ortaklama İşlemi

Ortalama Ortaklama yöntemindeki çıkış haritası giriş görüntüsü üzerinde bulunan piksellerden seçilmediğinden (ortalama alındığından) ortaya farklı bir görüntü meydana gelir. Bu yüzden Maksimum ortaklama yöntemi daha yaygın bir şekilde kullanılır.

2.2.3 Toplam Ortaklama (Sum Pooling)

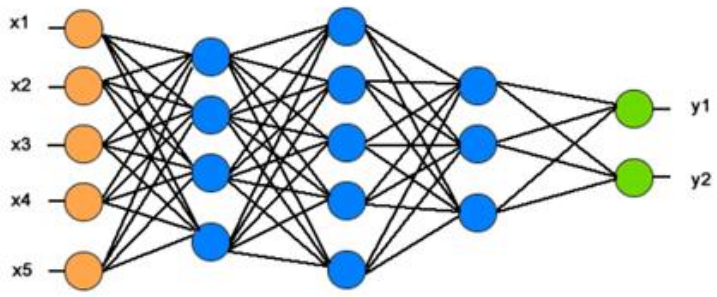
Toplam ortaklama işleminde uygulanan filtrenin denk geldiği matris elemanlarının toplamı çıkış olarak belirlenir. Şekil 2.6'da 4x4 boyutundaki görüntü üzerinde 2x2 boyutundaki bir filtre iki adım kaydırılarak elde edilen çıktı gösterilmiştir. Görüldüğü üzere filtrenin denk geldiği elemanların toplamı çıkış olarak belirlenir. Bu sayede giriş görüntüsünün boyutu küçültülür ve bilgisayar üzerine yüklenen işlem boyutu yarı yarıya azaltılır.



Şekil 2.6. Toplam Ortaklama İşlemi

2.3 Tam Bağlantılı Katman

Tam bağlantılı katman evrişimli sinir ağının son katmanıdır. Bu katmanın çıktıları ile tespit edilen nesnelerin neler olduğuna karar verilir. Şekil 2.7'de görüldüğü üzere bu katmana gelen matris verileri düzleştirilip işleme sokulur ardından nesnelerin sınıflandırılması yapılır.

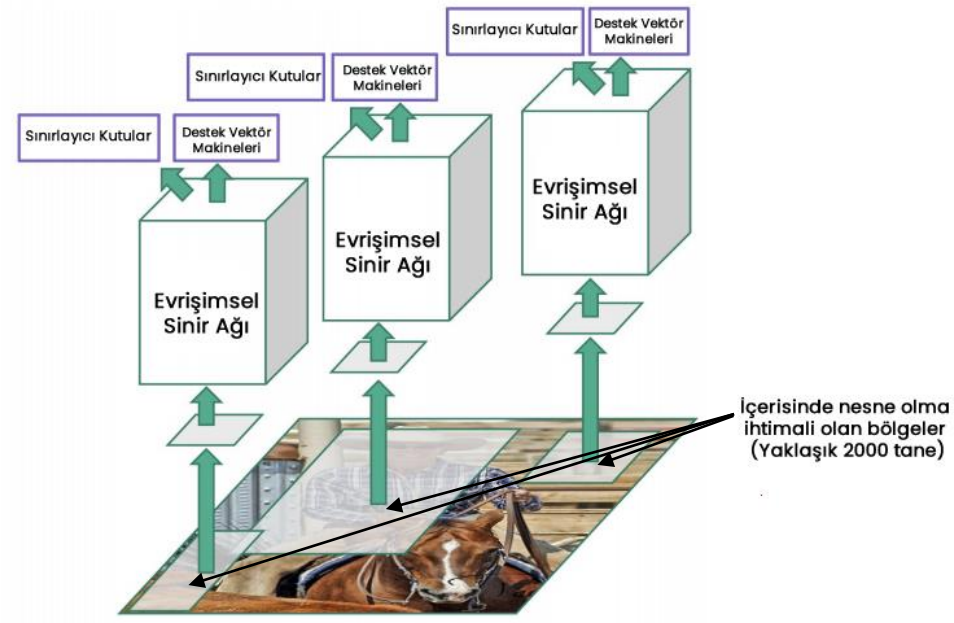


Şekil 2.7. Tam Bağlantılı Katman

3. BÖLGE TABANLI EVRİŞİMSEL SİNİR AĞLARI

3.1. Bölgesel Evrişimsel Sinir Ağları (R-CNN)

R-CNN mimarisinde Şekil 3.1'de görüldüğü üzere görüntü, seçici arama (selective search) ile içerisinde nesne olabilecek yaklaşık 2000 bölgeye ayrılır. Ardından her bölgeye evrişimsel sinir ağı (CNN) uygulanıp tespit edilen nesnelerin sınıfları tahmin edilmeye çalışılır.



Şekil 3.1. R-CNN Mimari Yapısı

Kaynak: (Li vd., 2017: 65)

Bölgesel evrişimsel sinir ağlarının (R-CNN) en büyük dezavantajı, her aday bölgeye ayrı bir evrişim ağı uygulandığı için eğitim süresi ve tahmin süresidir. Bu süreler eğitim için yaklaşık olarak 84 saat ve tahmin için ise bir görüntü için 49 saniyedir (Gandhi, 2018).

3.2. Hızlı Bölgesel Evrişimsel Sinir Ağları (Fast R-CNN)

R-CNN'in en büyük dezavantajı olan zaman sorununu ortadan kaldırmak için tasarlanmıştır. Bölgesel evrişimsel sinir ağlarında içerisinde nesne olabilecek her aday bölgeye uygulanan (yaklaşık 2000 tane) evrişim ağı yerine görüntüye tek bir evrişim ağı uygulanır. Bu sayede eğitim süresi yaklaşık 10 kat, tespit süresi ise yaklaşık 20 kat daha hızlı olmaktadır (Girshick, 2015: 6). Bu süreler eğitim için yaklaşık 8,75 saat, tahmin için 2,3

saniyedir (Gandhi, 2018). Şekil 3.2'de Fast R-CNN'in temel yapısı görünmektedir. Bu yapıyı bölgesel evrişimsel sinir ağlarından ayıran farklardan birisi görüntüye uygulanan 2000 tane ayrı evrişim ağı, diğeri ise sınıflandırma kısmındaki farklılıklardır. Burada Destek Vektör Makineleri yerine ayrı bir derin öğrenme sınıflandırması olan Softmax yapısı kullanılmıştır.

Şekil 3.2. Fast R-CNN Mimarisi

Kaynak: (Li vd., 2017: 74)

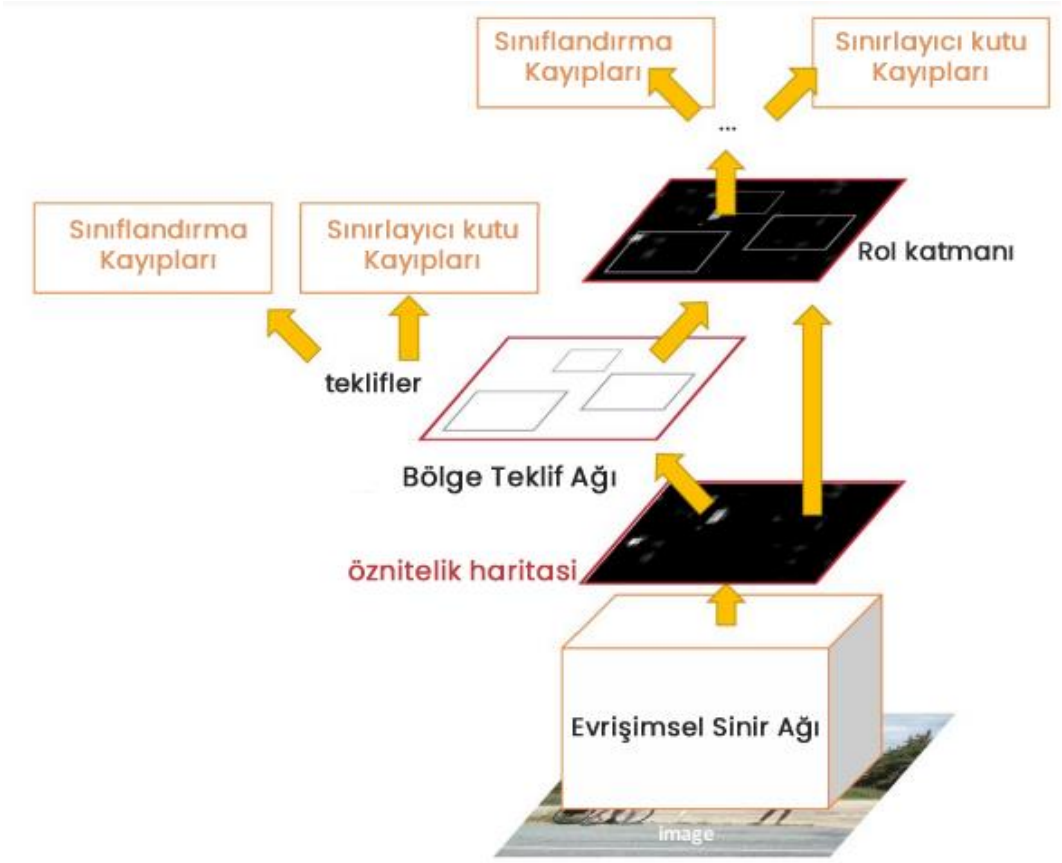
3.3. Daha Hızlı Bölgesel Evrişimsel Sinir Ağları (Faster R-CNN)

Faster R-CNN'i öncüllerine göre daha başarılı ve cazip kılan, görüntülerde nesnenin yer aldığı düşünülen bölgenin tahminine yönelik bir mekanizma (Region Proposal Network) ortaya koymasıdır. 2015 yılında Çinli bilgisayar bilimcisi Shaoqing Ren tarafından Microsoft firmasının Araştırma-Geliştirme (Ar-Ge) çalışması olarak nesne tanıma alanında en gelişmiş ve en ileri teknik olarak ortaya atılmıştır (Shaoqing, 2017). Bu ağların Google'ın geliştirdiği Tensorflow kütüphanesinde yer alan bir uygulaması ile gerçekleştirilen test sonuçlarının başarılı olduğu görülmüştür.

Bölge Teklif Ağı (Region Proposal Network) mekanizması sayesinde nesnenin yer aldığı düşünülen bölgenin tahminine yönelik işlemin maliyeti öncüllerine kıyasla çok daha az olup bu özellik algoritmayı muadillerine göre daha hızlı ve performanslı çalışma noktasında bir adım öne çıkarmıştır. Daha hızlı bölgesel evrişimsel sinir ağları için ortalama eğitim süresi ortalama 8,75 saat, tespit süresi ise 0,2 saniyedir. Bu verilere göre daha hızlı bölgesel

evrişimsel sinir ağları bir önceki sürümüne göre yaklaşık 10 kat, ilk sürümüne göre yaklaşık 100 kat daha hızlı tespit yapabilmektedir (Shaoqing, 2017).

Şekil 3.3'te daha hızlı bölgesel evrişimsel sinir ağlarının yapısı görülmektedir. Buna göre bölge teklif ağından alınan nesne teklifleri Faster R-CNN mimarisine giriş verisi olarak sunulur ve tam bağlantılı sinir ağı ile nesnelerin sınıflandırılması yapılır.



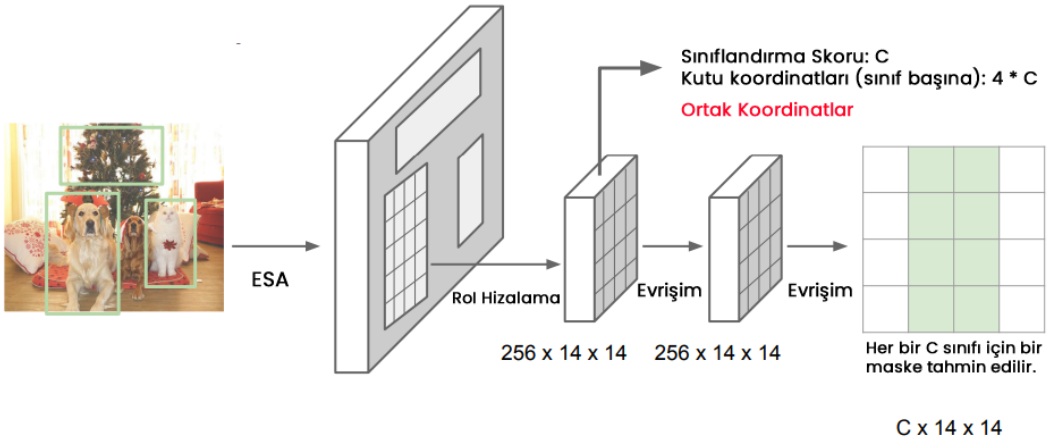
Şekil 3.3. Daha Hızlı Bölgesel Evrişimsel Sinir Ağı

Kaynak: (Li vd., 2017: 81)

3.4. Maskelenmiş Hızlı Bölgesel Evrişimsel Sinir Ağları (Mask R-CNN)

Bölge-tabanlı evrişimsel sinir ağı (Mask Region-based Convolutional Neural Network: Mask R-CNN) Faster R-CNN modeli temelinde geliştirilmiş sinir ağı modelidir. Faster R-CNN'de giriş resimlerinden özellik haritası (feature map) çıkarılır. Bir görüntünün potansiyel olarak bir nesne içeren bölgelerini belirlemek için bölgesel öneri ağından (Region Proposal Network: RPN) yararlanılır. Daha sonra bölgesel önerilerde havuzlama işlemi yapılır ve nesne tahmini için en sonunda tam bağlantılı MLP'ye giriş verisi olarak sunularak, önceden

belirlenen sınıf ya da sınıflar arasından bir tahmin yapılır (He vd., 2014: 2380-750). Yapılan bu işlemler Fast R-CNN'de daha yavaş olarak yürütülürken Faster R-CNN'de daha hızlı sonuçlar elde edilir (Shaoqing vd., 2017: 1137-1149).



Şekil 3.4. Mask R-CNN Mimarisi

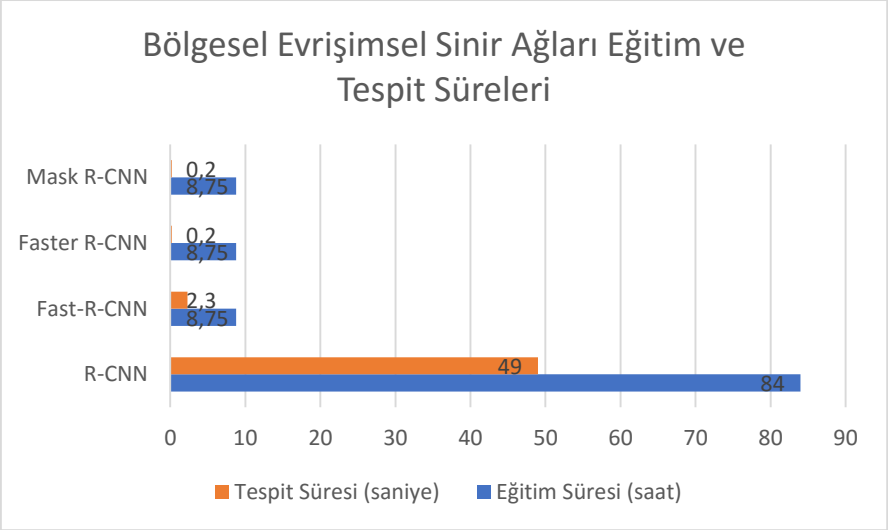
Mask R-CNN algoritmasında ise Şekil 3.4'te de görüldüğü üzere yüksek doğrulukla tahmin edilen nesnenin yüzeyi renklendirilerek maskeleme işlemi yapılır. Bu bölütleme işleminde örnek bölütleme (instance segmentation) özelliğinden yararlanılmaktadır. Örnek bölütlemede, eğer görüntüde aynı tür nesneye ait birden fazla nesne varsa, bu nesneler farklı tür renkler ile birbirinden ayrılır (Yayla ve Şen, 2019: 244-249). Çalışmamızda, İHA görüntüleri üzerinden araç tespiti yapıldığı yalnızca tek bir sınıf oluşturulmuştur. Mask R-CNN'de bulunan örnek bölütleme sayesinde görüntüde tespit edilen birden çok araç, farklı renkler ile maskelenmektedir. Bu sayede tespit edilen araçlar, farklı renkler ile birbirinden ayırt edilebilmektedir. Mask R-CNN'de algoritma kayıpları (L) sınıflandırma kayıpları (L_{cls}), maskeleme kayıpları (L_{mask}) ve çerçeve kayıpları (L_{box}) olarak 3 ana kayıptan meydana gelir ve algoritma kayıpları Denklem (3.1)'de gösterilmiştir (Yayla ve ŞEN, 2019: 1-4).

$$L = L_{cls} + L_{box} + L_{mask} \tag{3.1}$$

Algoritma kayıpları, Mask R-CNN eğitim sürecinde en aza indirgenerek, segmentasyonda oluşan hataların en aza düşürülmesi gerekmektedir. Eğitim sürecinde farklı iterasyonlar denenerek yapılan testler sonucunda toplamda 50 iterasyon sonunda hataların daha aza düşürülemediği gözlemlenmiş ve bu iterasyonda gözlenen hata / doğruluk oranı eşik değer olarak belirlenmiştir.

3.5. Bölgesel Evrişimsel Sinir Ağlarının Karşılaştırılması

Bölgesel evrişimsel sinir ağları hızlı ve doğru nesne tespiti yapabilmek için geliştirilmişlerdir. Ancak ilk versiyon doğru tespitler gerçekleştirilebilmesine rağmen hızlı tespitler gerçekleştirememiştir. Ayrıca eğitim süresi de oldukça fazladır. Bu dezavantajı ortadan kaldırmak için Ross Girshick ve arkadaşları bölgesel evrişimsel ağları hızlandırma çalışmalarına gitmişlerdir (Girshick vd., 2016: 3). Grafik 3.1'de de görüldüğü üzere yapılan iyileştirmeler ve geliştirilen yeni yapılarla (Bölge Teklif Ağları vb.) hızlandırılmış versiyonlar hem daha hızlı tespitler yapmakta hem de daha kısa sürede eğitilmektedir.



Grafik 3.1. Bölgesel Evrişimsel Sinir Ağlarının Karşılaştırılması

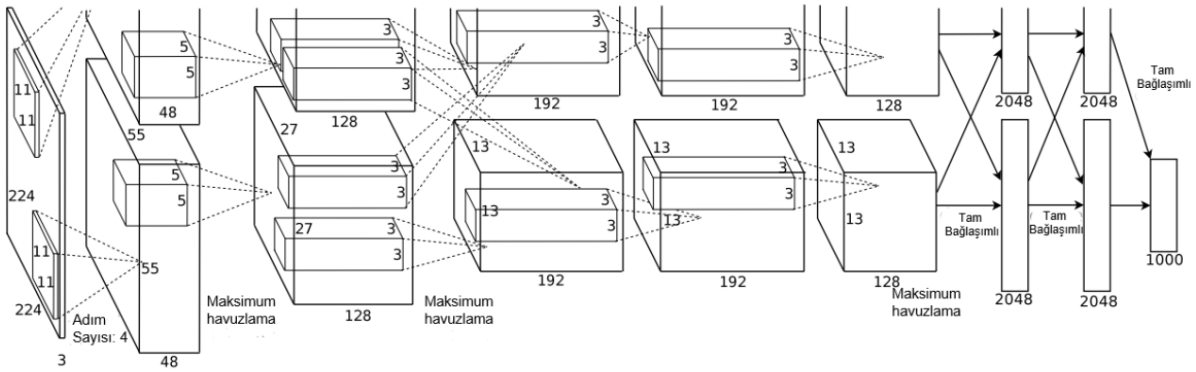
Kaynak: (Girshick vd., 2014: 5-6)

4. MEVCUT DERİN ÖĞRENME MODELLERİ

4.1. AlexNet Ağı

Alex Krizhevsky, Ilya Sutskever and Geoffrey Hinton tarafından geliştirilmiştir. AlexNet Şekil 4.1'de görüldüğü üzere 5 evrişim, 3 tanede tam bağlantılı katmandan oluşur. Aktivasyon fonksiyonu olarak ReLU (Rectified Linear Unit), havuzlama katmanlarında da max-pooling kullanılmaktadır. ReLU aktivasyon fonksiyonu her katmandan sonra uygulanır.

224x224x3 boyutundaki bir görüntüde 11x11 boyutundaki filtre 4 adım ile uygulanır. Ardından gerçekleştirilen 5 evrişim katmanı ve her katmandan sonra uygulanan maksimum havuzlama (maksimum ortaklama) ve ReLU aktivasyon fonksiyonunun ardından tam bağlantılı katmanlarda 4096 x 2 nöron elde edilir (Krizhevsky, 2012: 1097-1105).

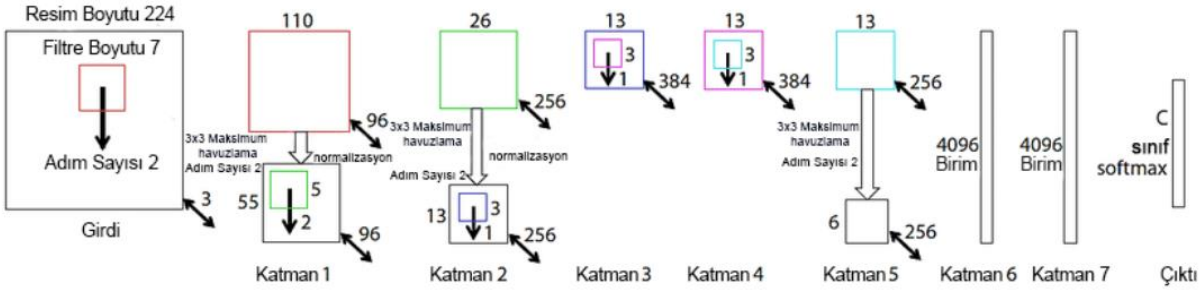


Şekil 4.1. AlexNet

Kaynak: (Krizhevsky, 2012: 5)

5.2. ZF-NET ağı

AlexNet üzerine geliştirilmiştir. Ağın elde ettiği sonuçlar ile gerçek değerler arasında karşılaştırma yapmak için çapraz düzensizlik maliyet fonksiyonunu kullanmaktadır. Şekil 4.2'de Zf-Net'in genel yapısı gösterilmektedir. AlexNet'ten farkı, orta katmanlardaki filtrelerin boyutu artırılırken ilk katmandaki filtre boyutu ve adım sayısı düşürülmüştür (Kurt, 2018: 42).



Şekil 4.2. ZFNet

Kaynak: (Kurt, 2018: 43)

4.3. VGG-NET Ağı

VggNet ağı AlexNet gibi temelde iki kısımda incelenebilir. Birincisi evrişim ve ortaklama katmanlarının bulunduğu kısım, ikincisi ise tam bağlantılı katmanların olduğu kısımdır. Ağa 224x224x3 boyutunda RGB formatında giriş görüntüsü sunulur. Genellikle kullanılan evrişim boyutları 3x3'tür. Sadece Vgg-16'nın bir konfigürasyonunda ek olarak 1x1 evrişim katmanı uygulanmıştır. Vgg-16'da 138 milyon, Vgg-19'da 144 milyon parametre hesabı yapılmaktadır (Kurt, 2018: 43). 60 milyon parametrenin hesaplandığı AlexNet'ten 2,5 kat daha fazla hesap yükü gerektirir.

Tablo 4.1'de ağ konfigürasyonu verilen VggNet'in Vgg-11, Vgg-13,Vgg-16 ve Vgg-19 gibi versiyonları mevcuttur. Burada yer alan sayılar ağın mimarisinde kullanılan katman sayılarını göstermektedir. Tabloda belirtilen konfigürasyonlardan sadece Vgg-11'de lokal cevap normalizasyonu (LRN) kullanılmıştır. Hafıza tüketimini arttırması dolayısıyla performansı olumsuz etkilemesi sebebiyle LRN, diğer konfigürasyonlardan çıkarılmıştır.

Tablo 4.1. Vgg Ağ Konfigürasyonları

VGG Konfigürasyonları					
A	A-LRN	B	C	D	E
11 Ağırlık Katmanı	11 Ağırlık Katmanı	13 Ağırlık Katmanı	16 Ağırlık Katmanı	16 Ağırlık Katmanı	19 Ağırlık Katmanı
Girdi (224 x 224 RGB Resim)					

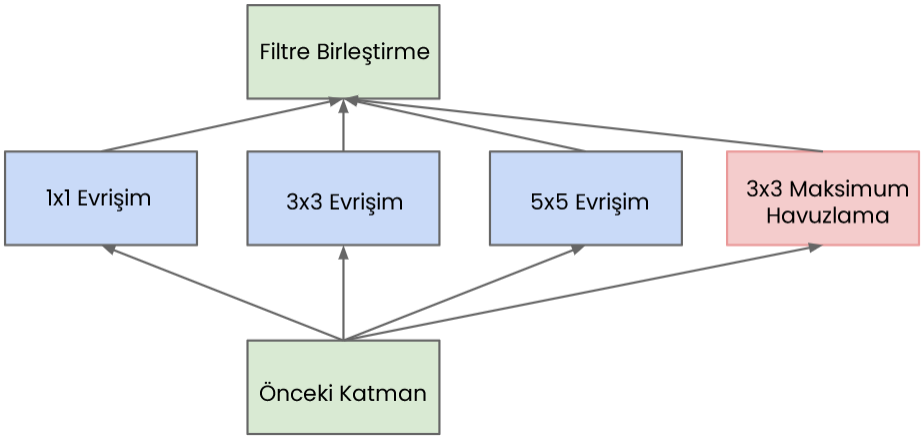
Tablo 4.1. devam ediyor.

evrişim3-64	evrişim3-64 LRN	evrişim3-64 evrişim3-64	evrişim3-64 evrişim3-64	evrişim3-64 evrişim3-64	evrişim3-64 evrişim3-64
Havuzlama					
evrişim3-128	evrişim3-128	evrişim3-128 evrişim3-128	evrişim3-128 evrişim3-128	evrişim3-128 evrişim3-128	evrişim3-128 evrişim3-128
Havuzlama					
evrişim3-256 evrişim3-256	evrişim3-256 evrişim3-256	evrişim3-256 evrişim3-256	evrişim3-256 evrişim3-256 evrişim1-256	evrişim3-256 evrişim3-256 evrişim3-256	evrişim3-256 evrişim3-256 evrişim3-256 evrişim3-256
Havuzlama					
evrişim3-512 evrişim3-512	evrişim3-512 evrişim3-512	evrişim3-512 evrişim3-512	evrişim3-512 evrişim3-512 evrişim1-512	evrişim3-512 evrişim3-512 evrişim3-512	evrişim3-512 evrişim3-512 evrişim3-512 evrişim3-512
Havuzlama					
evrişim3-512 evrişim3-512	evrişim3-512 evrişim3-512	evrişim3-512 evrişim3-512	evrişim3-512 evrişim3-512 evrişim1-512	evrişim3-512 evrişim3-512 evrişim3-512	evrişim3-512 evrişim3-512 evrişim3-512 evrişim3-512
Havuzlama					
TB-4096					
TB-4096					
TB-1000					
softmax					

Kaynak: (Kurt, 2018: 44)

4.4 GOOGLE-NET Ağı

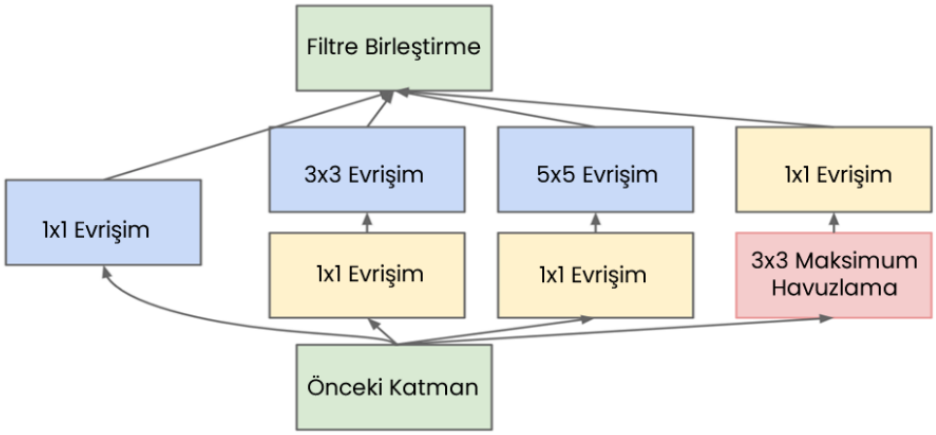
GoogleNet daha az hesaplama maliyeti ile daha iyi sonuçlar elde edebilmek için geliştirilmiştir. 22 adet katman bulunur. Ağın yapısı karmaşık ve anlaşılması zordur. Bu karmaşıklık ağa hız ve başarım getirmektedir. Bu ağa geliştiricileri tarafından Inception adı verilmiştir. Diğer ağlardan farkı giriş görüntüsüne uygulanan filtre ve işlemlerin sıralı bir şekilde uygulanması yerine aynı anda uygulanmasıdır. Aynı anda uygulanan bu işlemlere modül adı verilmektedir. Şekil 4.3'te görüldüğü üzere giriş görüntüsüne aynı anda birden fazla farklı boyutlarda evrişim işlemi ve maksimum havuzlama işlemi uygulanır. Ardından bu katmanlar birleştirilerek tek bir sonuç elde edilir ve diğer modüle giriş görüntüsü olarak sunulur. Bahsedilen bu modül uygulama işlemi sayesinde ağ giriş görüntüsünden hem özel hem de genel özellikler çıkarabilmektedir (Szegedy vd., 2014: 4).



Şekil 4.3. Basit Başlangıç (Inception) Modül Yapısı

Kaynak: (Szegedy vd., 2014: 5)

Şekil 4.3'te görüldüğü üzere bir önceki katmandan gelen görüntüye 3x3 ve 5x5 boyutunda evrişim işlemleri uygulanır ve bu her modülde tekrarlanır. Bu hesaplama maliyetini azaltmak için Şekil 4.4'te gösterildiği üzere 3x3 ve 5x5 boyutunda evrişim uygulanmadan önce 1x1 boyutunda evrişim işlemi uygulanarak boyut düşürme işlemi gerçekleştirilir.



Şekil 4.4. Boyut Düşürme İşleminden Sonra Başlangıç (Inception) Modül Yapısı

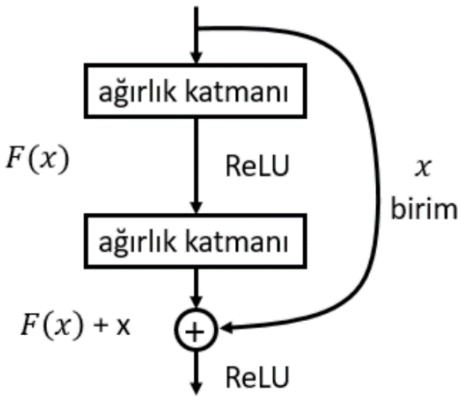
Kaynak: (Szegedy vd., 2014: 5)

4.5 Microsoft RES-NET Ağı

Microsoft firması tarafından geliştirilen ve ILSVRC 2015 yarışmasında birinci olan Resnet (Residual Network), Residual Blocks (Kalıntı Blokları) ya da diğer ismiyle Artık Bloklardan oluşmaktadır. Temelde birkaç katman önceki çıkışı o anki çıkış ile toplayıp aktivasyon fonksiyonuna giriş değeri olarak sunar. Bu çözüm gradient değerinin sıfıra yakınsaması problemini ortadan kaldırır.

$$h(x) = f(x) + x \tag{4.1}$$

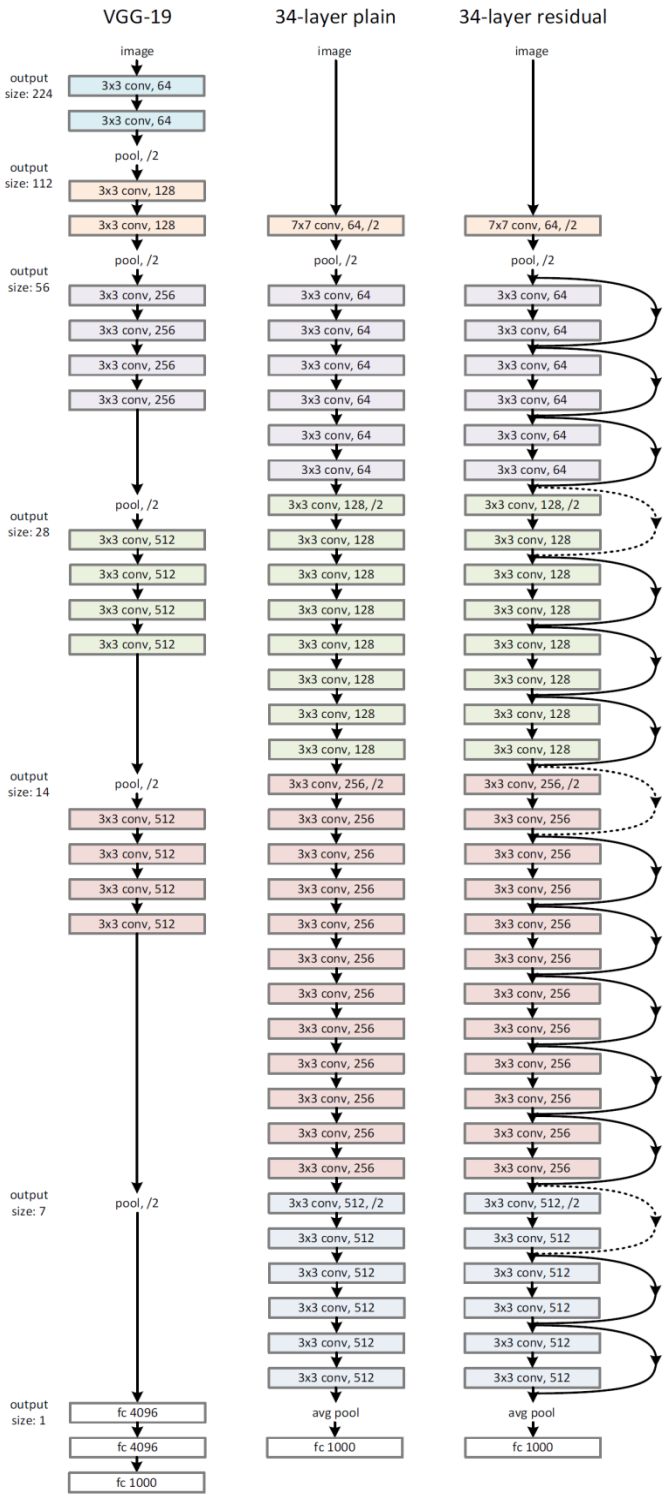
Kalıntı bloğu yukarıdaki formül ile hesaplanır. Daha önceki katmanın çıkışı olan x değeri en son katmandaki çıkış olan $f(x)$ değeri ile toplanıp RELU aktivasyon işlemine girdi olarak sunulur (He vd., 2015: 3). Kalıntı bloğunun genel yapısı Şekil 4.5'te gösterilmiştir.



Şekil 4.5. Kalıntı (Residual) Bloğunun Genel Yapısı

Kaynak: (He vd., 2015: 2)

Şekil 4.6'da ise Microsoft ResNet Ağının Vgg-19 Ağ mimarisi ile karşılaştırılması gösterilmiştir.



Şekil 4.6. 34 Katmanlı ResNet ve Vgg-19 Ağ Modelinin Karşılaştırılması

(**Kaynak:** He vd., 2015: 4).

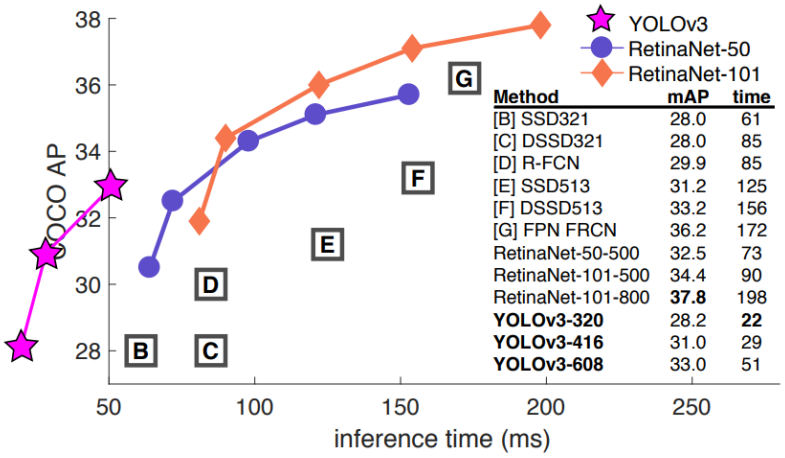
Şeklin sol tarafında 5.3. bölümde bahsedilen klasik Vgg-19 Ağ mimarisi, ortada 34 parametre katmanına sahip sade bir ağ modeli en sağda ise 34 parametre katmanına sahip kalıntı ağ modeli (ResNet-34) gösterilmiştir. Şekilde görüldüğü üzere kalıntı ağ modelinde yeni çıkış kendisinden önceki iki katmanın değeri ile toplanarak bulunur.

ResNet modelinin getirdiği kalıntı bloğu çözümü ile ağda kullanılan ve hesap yükünü arttıran parametre sayısı azaltmayı başarmıştır. ResNet modelinin daha derin ağlar için geliştirilmiş ResNet-50, ResNet-101 ve Resnet-152 model mimarileri de mevcuttur.

ResNet-152 son teknoloji derin öğrenme ağ modelleri arasında 152 katmanlı yapısı ile en derin olan modeldir.

4.6. YOLO Derin Sinir Ağı

YOLO temelde evrişimsel sinir ağları ile geliştirilen, nesne tespiti yapabilen bir derin sinir ağıdır. You Only Look Once (Sadece bir kez bakarsın.) cümlesinin baş harflerinin kısaltmasıdır. Sebebi ise görüntü üzerinde tek bir seferde ağın çalışmasıdır. Bu sayede çok hızlı tespitler yapabilmektedir. Şekil 4.7'de YOLO derin sinir ağının diğer metotlara göre çıkarım süresinin kıyaslaması yapılmıştır. Ayrıca YOLO'nun en büyük artılarından biri de nesne tespiti yaparken hız ve doğruluk arasında değiş tokuş yapılabilmesidir. (Redmon vd., 2018)



Şekil 4.7. YOLO ve Benzer Metotların Çıkarım Süreleri

Kaynak: (Redmon ve Farhadi, 2018: 1)

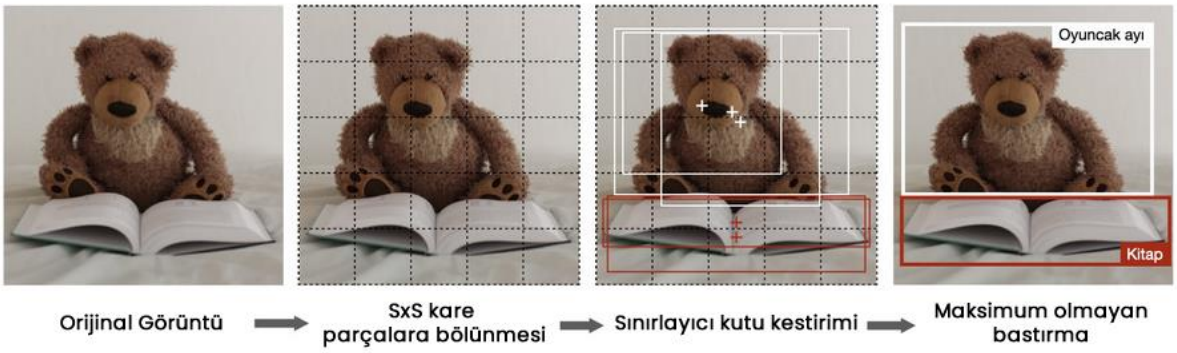
YOLO'nun çalışması temelde 3 kısımda incelenebilir.

a) Görüntü SxS kare parçalara bölünür.

b) Her bir parça için aşağıda görülen fonksiyonu öneren bir Evrişimsel Sinir Ağı çalıştırılır. Fonsiyonda, p_c'nin bir nesneyi algılama olasılığı olduğu durumlarda, b_x, b_y, b_h, b_w tespit edilen olası sınırlayıcı kutusunun özellikleridir. Burada yer alan "x" ve "y", sınırlayıcı kutunun görüntüdeki koordinatlarını, "h" ve "w" ise sınırlayıcı kutunun genişlik ve yüksekliğini temsil etmektedir. $c_1, c_2, \dots, c_p$ sınıflarının tespit edilen one-hot temsildir ve k öneri (anchor) kutularının sayısıdır. Fonksiyonun işletilmesinin sonucunda her parça kendi içinde nesne olup olmadığını nesne var ise merkez noktasının kendi içinde olup olmadığını kontrol eder. Ardından da ilgili parça, nesnenin merkez noktasının kendi içinde olduğunu düşünüyorsa sınırlayıcı kutu kestirimi yapar ve çerçeve içerisine alır. Denklem 4.2'de yer alan (1+p) ifadesindeki "1" değeri tespit edilecek sınıf sayısını temsil etmektedir.

$$y = \underbrace{\left[p_c, b_x, b_y, b_h, b_w, c_1, c_2, \dots, c_p, \dots\right]^T}_{k\ kez\ tekrarlanır} \in R^{SxSxkx(1+p)} \qquad (4.2)$$

c) Birden fazla parça, nesnenin merkezinin kendi içinde olduğunu varsayıp çerçeveleme yapabilir. Bu hatanın önüne geçmek için Non-Maximum Suppression (Maksimum Olmayan Bastırma) işlemi gerçekleştirilir. Bu işlem, görüntü üzerinde parçalar tarafından çizilen çerçevelerin güven skoru en yüksek olanını ekrana çizdirme işlemidir. YOLO'nun genel çalışma yapısının gösterildiği Şekil 4.8'de görüldüğü üzere oyuncak ayı için 3 adet çerçeveleme, kitap için ise 2 adet çerçeveleme yapılmıştır. Bu sınırlayıcı kutu kestirimlerden güven skoru en yüksek olan, maksimum olmayan bastırma aşamasında seçilerek ekrana çizdirilmiştir.



Şekil 4.8. YOLO'nun Genel Çalışma Yapısı

Kaynak: (Amidi ve Amidi, 2019)

5. İHA GÖRÜNTÜLERİNDEN ARAÇ TESPİTİNİN YAPILMASI

İnsansız hava aracından alınan görüntülerden araç tespitinin yapılması literatürde öne çıkan iki model olan Mask R-CNN ile YOLO derin sinir ağları ile gerçekleştirilmiş ve sonuçlar karşılaştırılmıştır. Her iki ağ için de Bilecik ili Pazaryeri ilçesinde DJI Mavic Pro ile çekilen ve Github'da bulunan açık kaynaklı bir veri setinden toplam 282 adet görüntü içerisinde bulunan 789 örnek kullanılmıştır (Kharuzhy, 2018). Ağlar her bir adımda 1000 iterasyon, toplamda 50 adımlık bir eğitimden geçirilerek İHA görüntüsünde bulunan araçların kuş bakışı tespiti yapılmıştır. Öncelikle İHA görüntüsünden alınan fotoğraflar (282 Adet) %90-%10 oranında eğitim ve test veri seti olarak ikiye ayrılmıştır. 282 fotoğrafın 254 adedi eğitim, 28 adedi ise test için ağların ezberleme ihtimallerini düşürmek amacıyla farklı lokasyonlardan denk gelecek şekilde ayarlanarak ayrılmıştır. Tüm görüntülerde bulunan araçlar Oxford Üniversitesinden *VGG Image Annonator* olarak adlandırılan kullanışlı ve sade bir çizim aracıyla eğitime hazırlanmıştır.

Çalışma için gerekli olan ve yüksek hesaplama kabiliyeti gerektiren eğitim ve testler, Google firmasının ücretsiz olarak kullanıma sunduğu Google Colaboratory üzerinde gerçekleştirilmiştir. Google Colaboratory, Nvidia Tesla K80 GPU'ya sahip ve içerisinde birçok Python ve derin öğrenme kütüphanesini hazır bulunduran ücretsiz bir ortamdır. Ayrıca Python dilinde Keras ve Tensorflow kütüphanelerinden de yararlanılmıştır (Abdulla, 2017).

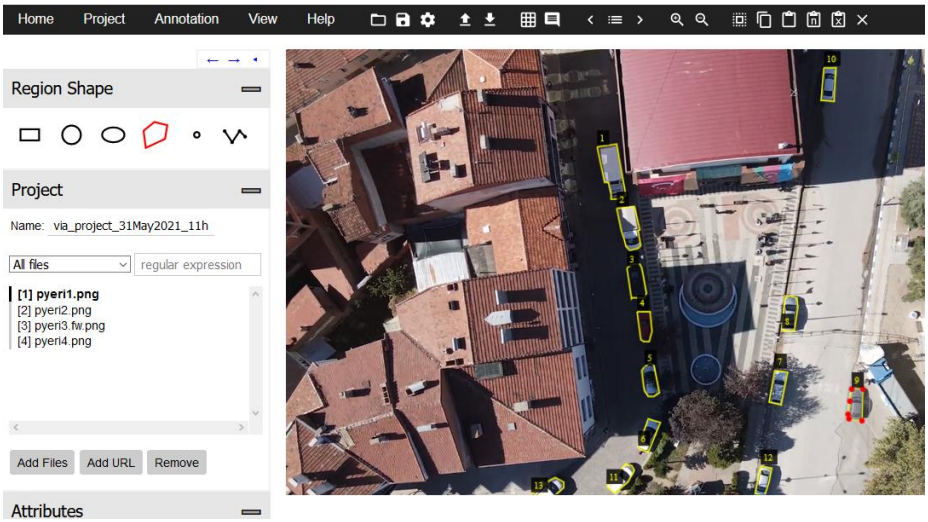
5.1. Mask R-CNN Modeli ile Araç Tespitinin Yapılması

5.1.1. Veri Setinin Hazırlanması

Görüntülerin araç tespitine yönelik eğitime hazırlanması için sade ve kullanışlı bir web çizim aracı olan VGG Image Annotator yazılımı kullanılmıştır (Dutta ve Zisserman, 2019: 2276–2279). VGG çizim aracı sayesinde eğitime verilecek görüntüler, test ve eğitim seti olarak iki aşamalı hazırlanır. Her iki sete ait görüntülerdeki araçlar, poligonal olarak çizilerek koordinatları belirlenir. Bu işlemi gerçekleştirmek için aracın sol tarafında yer alan, Şekil 5.1'de de görülen "Region Shape" bölümünden "Polygon Region Shape" kısmı işaretlenir. Ayrıca şekilde 13 adet aracın nasıl işaretlendiği gösterilmiştir.

Etiketleme işlemiden sonra Şekil 5.1'de de görülen "Annotation" menüsünden Matterport Mask R-CNN çerçevesinde kullanılacak olan JSON dosyası kaydedilir. Bu işlem

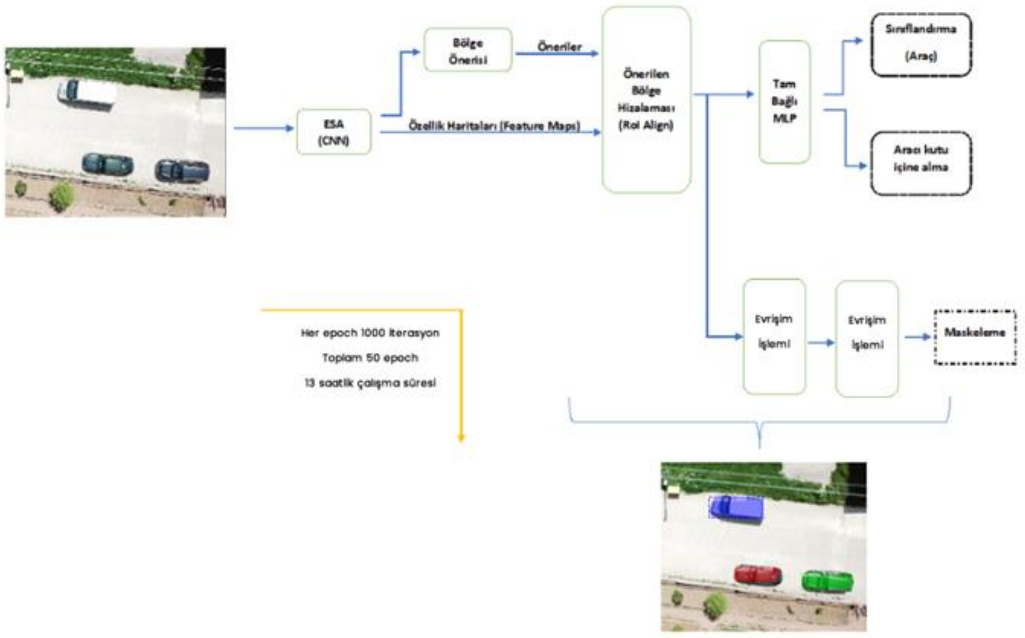
hem eğitim hem de test görüntüleri için ayrı ayrı yapılır. Elde edilen JSON formatlı dosyalar görüntülerle birlikte eğitim aşamasında kullanılır.



Şekil 5.1. Vgg İmage Annotator Web Aracının Kullanılması

5.1.2. Sistem Tasarımı ve Bileşenleri

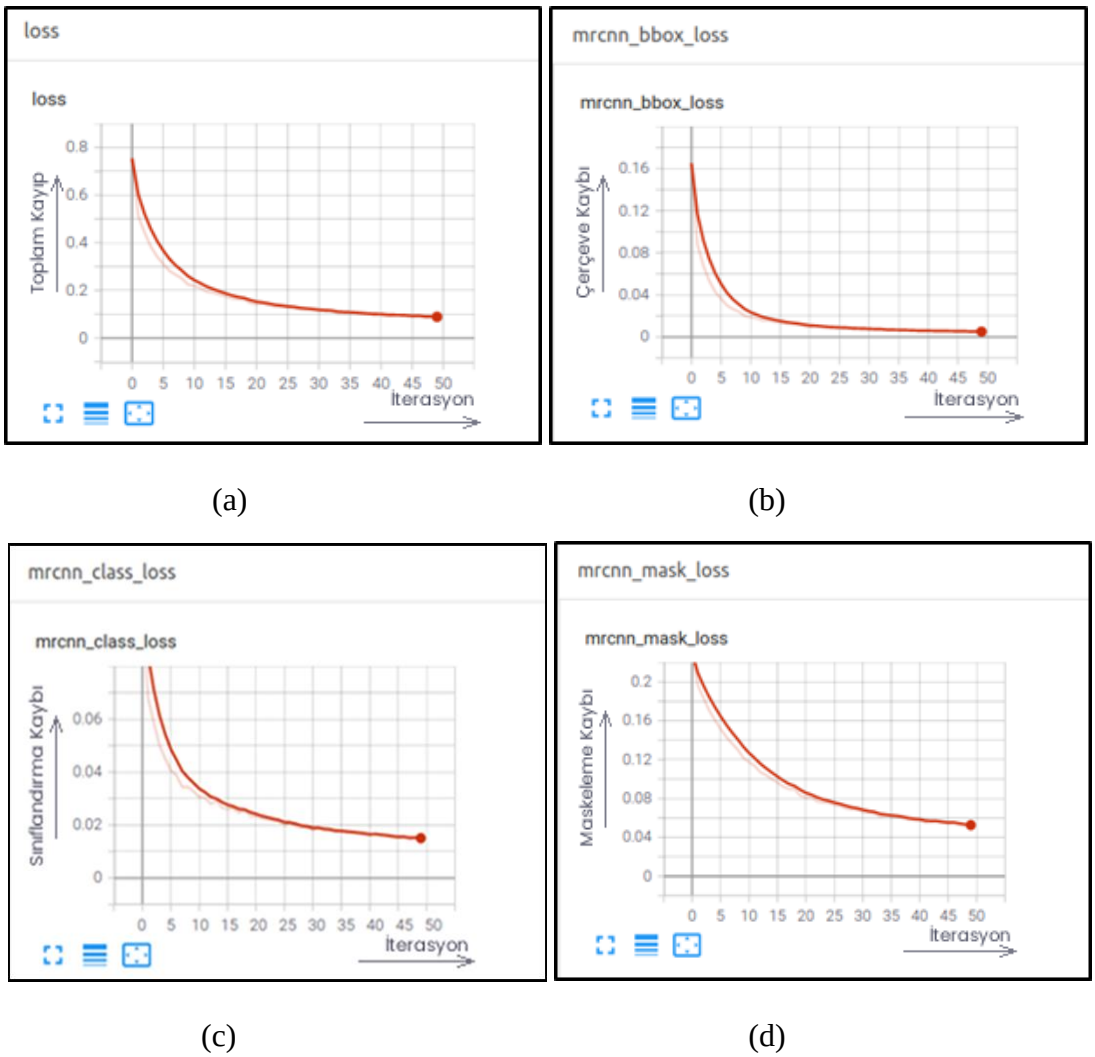
Sistem tasarımı için kullanımı kolay ve popüler çerçevelerden biri olan Matterport Mask R-CNN seçilmiştir (Abdulla, 2017). Matterport Mask R-CNN 4.5. bölümde bahsedilen ResNet-101 mimarisi üzerine kurulmuştur. Veri seti için önceden eğitilmiş MS COCO veri setinden yararlanılmıştır. Sistemin çalışma prensibi Şekil 5.2'de gösterilmiştir.



Şekil 5.2. Mask R-CNN Çalışma Prensibi

5.1.3. Eğitim ve Test İşlemlerinin Gerçekleştirilmesi

Google Colaboratory üzerinde Matterport Mask R-CNN çerçevesi ile öğrenme oranı (0.001) ve momentum (0.9) seçilerek eğitim gerçekleştirilmiştir. Eğitim sonunda hata oranlarının tespit edilebilmesi için Tensorflow kütüphanesinde bulunan Tensorboard grafik arayüzünden yararlanılarak, hata oranlarının iterasyonlara göre gözlemi yapılabilmektedir. Eğitim sonucunda elde edilen algoritma kayıplarının grafiksel gösterimi Şekil 5.3'te gösterilmiştir. Araç tespitinde Mask R-CNN algoritma kayıpları en aza indirgenerek, segmentasyon hatalarının azaltılması, segmentasyon kalitesinin arttırılması için gereklidir. Bu amaçla her epoch 1000 iterasyon ve toplamda 50 epoch ile yapılan testler sonucunda algoritma hatalarının en aza indirildiği gözlemlenmiştir.



(a) (b)

(c) (d)

Şekil 5.3. Algoritma Kayıpları

(a) Toplam kayıp, (b) Çerçeve kaybı, (c) Sınıflandırma kaybı, (d) Maskeleme Kaybı

Çalışmada, 50 dönem (epoch) sonucunda ulaşılan minimum hata oranları Tablo 5.1'de, Bilecik ili Pazaryeri ilçesinde farklı lokasyonlarda ve farklı yükseklikte İHA kamerasından elde edilen görüntülerin segmentasyon sonuçları Şekil 5.4'te gösterilmiştir. Toplam kayıp oranı diğer kayıp oranlarının (Çerçeve kaybı, sınıflandırma kaybı ve maskeleme kaybı) toplamı ile hesaplanır.

Tablo 5.1. Kayıp Oranları

Toplam Kayıp	Çerçeve Kaybı	Sınıflandırma Kaybı	Maskeleme Kaybı
0,08652	4,9783e-3	0,01478	0,05261

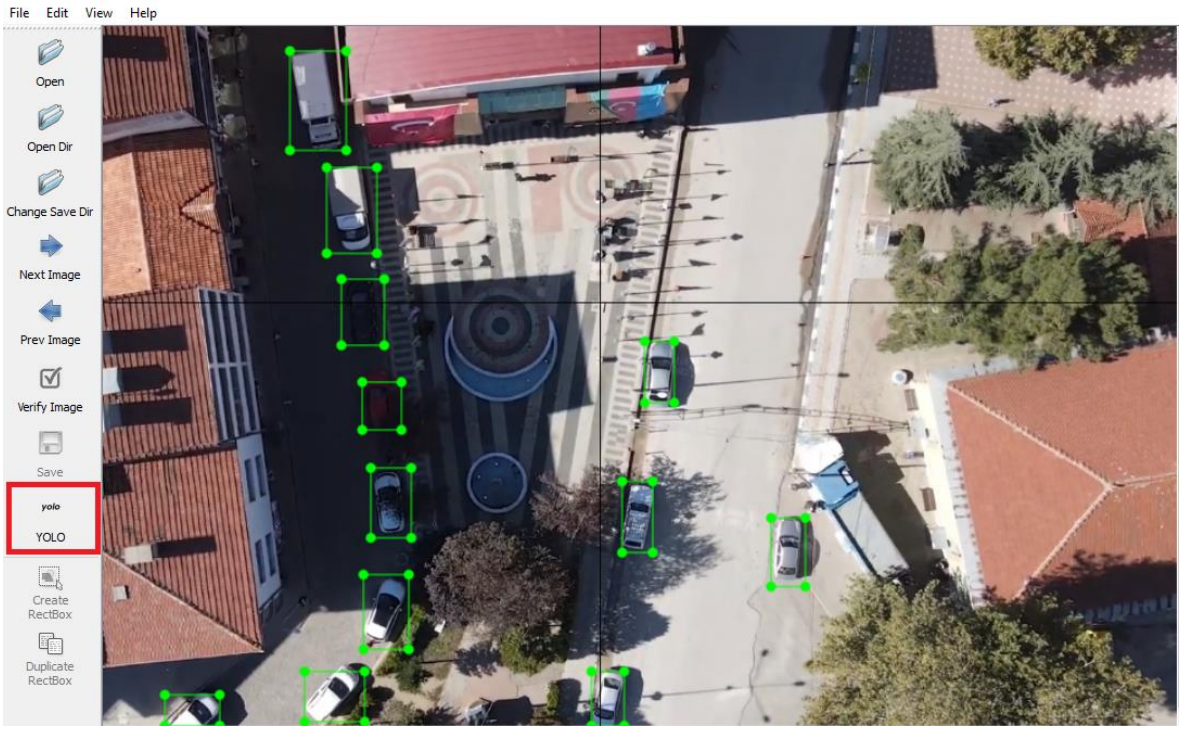
Şekil 5.4. Mask R-CNN ile Bilecik/Pazaryeri Görüntülerinden Araç Tespiti Örnekleri

5.2. YOLO Modeli ile Araç Tespitinin Yapılması

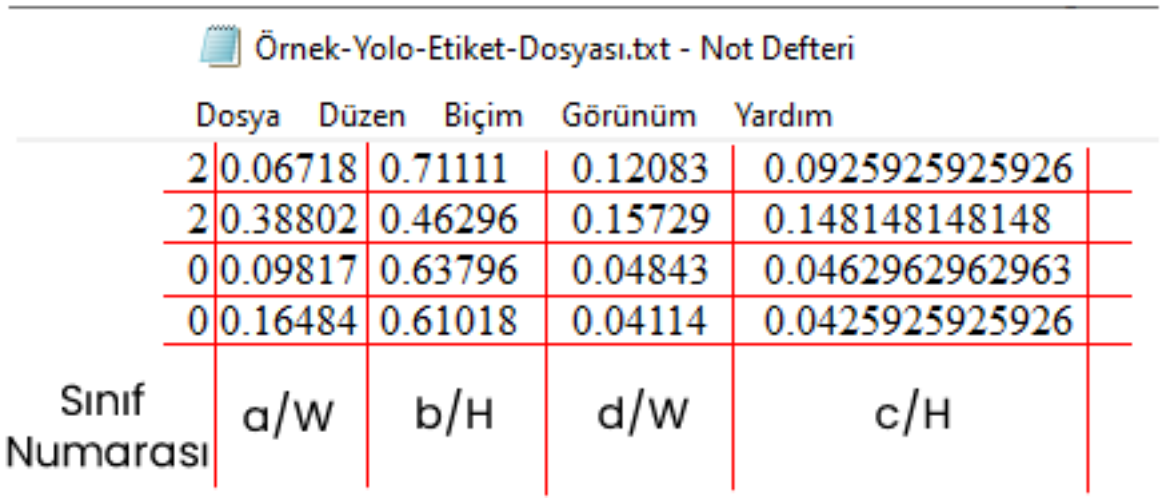
5.2.1. Veri Setinin Hazırlanması

YOLO derin sinir ağında kullanılmak üzere ilgili eğitim ve test görüntüleri grafiksel bir görüntü etiketleme aracı olan LabelImg programı ile etiketlenmiştir (Lin, 2017). LabelImg ayrıca PASCAL VOC ve CreateML formatını da destekler. LabelImg çizim aracı sayesinde eğitime verilecek görüntüler, test ve eğitim seti olarak iki aşamalı hazırlanır. Her iki sete ait görüntülerdeki araçlar, Mask R-CNN'in aksine poligonal olarak değil dikdörtgen biçiminde çizilerek koordinatları belirlenir. Ayrıca YOLO için yapılan etiketleme de her görüntü için ayrı bir txt dosyası oluşturulur. LabelImg ile YOLO için etiketleme yaparken aracın sol tarafında yer alan, Şekil 5.5'te da görülen YOLO kısmı seçilir. Ayrıca şekilde 13 adet aracın nasıl işaretlendiği gösterilmiştir.

Etiketleme işleminden sonra Şekil 5.5'te de görülen "File" menüsünden YOLO modelinde kullanılacak olan "txt" dosyası kaydedilir. Bu işlem hem eğitimdeki hem de testteki görüntüler için ayrı ayrı yapılır. Elde edilen "txt" formatlı dosyalar görüntülerle birlikte eğitim aşamasında kullanılır. Oluşturulan "txt" dosyasının içeriğinde bulunan değerlerin açıklamaları Şekil 5.6'da belirtilmiştir.



Şekil 5.5. LabelImg Programının Kullanılması



Şekil 5.6. YOLO Etiket Dosyası Örneği

Şekil 5.6'de belirtilen;

W: Görüntünün genişliğini

H: Görüntünün yüksekliğini

a: Nesnenin görüntüdeki x pozisyonunu

b: Nesnenin görüntüdeki y pozisyonunu

c: Nesnenin genişliğini

d: Nesnenin yüksekliğini

temsil etmektedir. Sınıf Numarası eğitim setindeki kategorileri temsil etmektedir. Tek sınıf ile nesne tespiti yapılacaksa bu alandaki değer sadece sıfır olacaktır.

5.2.2. Sistem Tasarımı ve Bileşenleri

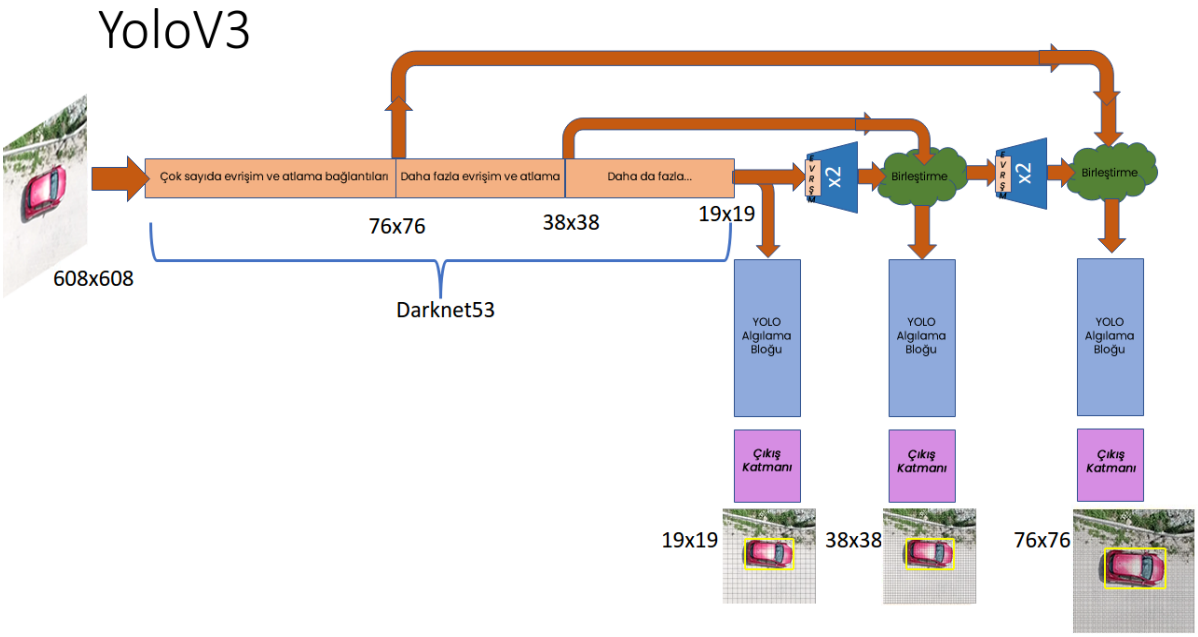
Sistem tasarımı için Washington Üniversitesinden Joseph Redmon ve Ali Farhadi'nin hazırlamış olduğu YOLOv3 çerçevesi kullanılmıştır (Redmon ve Farhadi, 2018). YOLOv2'de DarkNet-19 mimarisi kullanılırken, YOLOv3 DarkNet-53 mimarisi üzerine kurulmuştur. DarkNet-53, ResNet ve FPN (Feature-Pyramid Network) ağlarından ilham alınarak geliştirilmiştir. Bu iki ağın hibrit halinin olduğu da söylenebilir. Bu mimari barındırdığı atlama bağlantıları ile ResNet'e, 3 tahmin özelliği ile de FPN'e benzemektedir. DarkNet-53 3x3 ve 1x1 evrişim katmanlarından yararlanır. DarkNet-53 mimarisi Şekil 5.7'de, YOLO

çalışma prensibi de Şekil 5.8'de verilmiştir. Veri seti için ise önceden eğitilmiş MS COCO veri setinden yararlanılmıştır.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Şekil 5.7. DarkNet-53 Mimarisi

Kaynak: (Redmon ve Farhadi, 2018: 2)



Şekil 5.8. YOLO Çalışma Prensibi

Kaynak: (Almog, 2020)

5.2.3. Eğitim ve Test İşlemlerinin Gerçekleştirilmesi

YOLO, Mask R-CNN'deki gibi bölgesel tahmin ağı kullanmayıp giriş görüntüsüne tek seferde evrişimsel sinir ağı uyguladığından önem arz eden hiper paremetreleri de farklı olmaktadır. YOLO 'da ana hiper parametre giriş görüntüsünün boyutudur. Modelimizde (608x608) giriş görüntüsü boyutu kullanılmıştır. Gerçekleştirilen deneme yanılma yöntemi sayesinde edinilen tecrübeyle Momentum (0.9), öğrenme oranı (0.001), weight decay (0.0005) ve Batch boyutu (64) olarak seçilmiştir. Öğrenme işlemi, toplamda 50000 iterasyondan sonra ortalama kayıp 0,12 seviyelerine kadar düştüğünde sonlandırılmıştır. Bilecik ili Pazaryeri ilçesinde farklı lokasyonlarda ve farklı yükseklikte İHA kamerasından elde edilen görüntülerin YOLO ile araç tespit başarısı Şekil 5.9'da gösterilmiştir.

Şekil 5.9. YOLO ile Bilecik/Pazaryeri Araç Tespiti Görüntüsü

5.3. Yolo ve Mask R-CNN'in Karşılaştırılması

YOLO ve Mask R-CNN'in bu çalışma için seçilme nedenlerinden biri de YOLO'nun nesne tespit etmedeki hızı ve etkinliği ile Mask R-CNN'in tespit edilen nesneleri örnek bölütleme özelliği sayesinde arka plan dokusundan ayırmasıdır.

Literatürde YOLO'nun, Mask R-CNN'e göre daha hızlı tespit yaptığı ancak Mask R-CNN'in ise daha doğru sonuçlar ortaya koyduğu söylenmektedir (Buric vd., 2018). Nitekim bu çalışma ile de benzer sonuçlar elde edilmiştir. Her iki derin sinir ağı da aynı veri seti ile toplam kayıpları minimum seviyeye düşene kadar eğitilmişlerdir. YOLO'nun daha yüksekten çekilen görüntülerdeki başarısının Mask R-CNN'e göre daha düşük olduğu Şekil 5.4 ve Şekil

5.9'daki en altta bulunan görsellerde gözlemlenmiştir. Ancak bu dezavantajını YOLO, nesneleri Mask R-CNN'e göre daha hızlı tespit ederek azaltmıştır. Karşılaştırma için kesinlik (precision), duyarlılık (recall), F1-Skoru (F1-Score) ve kalite (quality) metrikleri kullanılmıştır (TBAV, 2020). Karşılaştırma için her iki ağın da daha önce görmediği Bilecik ili Pazaryeri ilçesinde farklı lokasyon ve yükseklikten çekilen görüntüler ile Github üzerinde bulunan açık kaynak olarak sunulan toplam 80 adet görüntü (212 örnek) kullanılmıştır (Kharuzhy, 2018).

Kesinlik metriği (5.1); pozitif olarak tahmin edilen değerlerin gerçekten kaç adedinin pozitif olduğunu göstermek için kullanılmıştır.

Duyarlılık metriği (5.2); pozitif olarak tahmin edilmesi gereken işlemlerin ne kadarının pozitif olarak tahmin edildiğini göstermek için kullanılmıştır.

F1-Skoru metriği (5.3); kesinlik ve hassaslık değerlerinin harmonik ortalamasını göstermektedir. Harmonik ortalama tercih edilmesinin sebebi, uç durumları da işleme almak içindir.

Kalite metriği (5.4); algoritmanın yeni bir girdi de göstereceği muhtemel başarı oranını ifade etmektedir. Metrikler tanımları şu şekildedir;

$$Kesinlik = \frac{DP}{DP + YP} \tag{5.1}$$

$$Duyarlılık = \frac{DP}{DP + YN} \tag{5.2}$$

$$F1\ Skoru = \frac{2 * Kesinlik * Duyarlılık}{Kesinlik + Duyarlılık} \tag{5.3}$$

$$Kalite = \frac{DP}{DP + YP + YN} \tag{5.4}$$

DP: Doğru Pozitif (Gerçekte pozitif olup modelin pozitif tahmin ettiği objeler)

YP: Yanlış Pozitif (Gerçekte pozitif olmayıp modelin pozitif tahmin ettiği objeler)

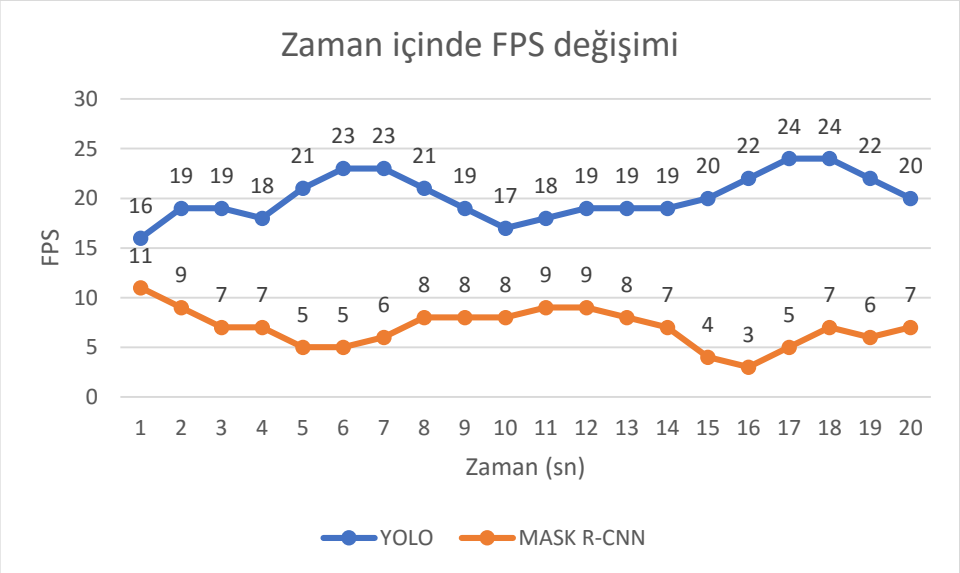
YN: Yanlış Negatif (Gerçekte pozitif olup modelin negatif tahmin ettiği objeler)

Bu kriterler doğrultusunda YOLO ve Mask R-CNN derin sinir ağlarının performanslarının karşılaştırılması Tablo 5.2'de gösterilmiştir. Elde edilen veriler göstermektedir ki Mask R-CNN derin sinir ağı YOLO derin sinir ağına göre daha doğru sonuçlar vermektedir.

Tablo 5.2. YOLO ve Mask R-CNN Derin Sinir Ağlarının Performanslarının Karşılaştırılması

Ağ	DP	YP	YN	Kesinlik	Duyarlılık	F1 Skoru	Kalite
YOLO	162	24	26	87,09	86,17	86,63	76,41
MASK R-CNN	168	21	23	88,88	87,95	88,42	79,24

FPS (Frame per Second) karşılaştırması için 20 saniye süren, Bilecik ili Pazaryeri ilçesinde çekilen ve her iki ağın da hiç görmediği video görüntüsü üzerinden tespit edilen sonuçlar Grafik 5.1'de gösterilmiştir. YOLO derin sinir ağı görüntüye tek bir sinir ağı uyguladığından en büyük avantajı nesneleri hızlı tespit edebilmesidir. Nitekim bu çalışmadaki hız testinde de benzer sonuçlar elde edilmiştir. YOLO test işleminde kullanılan donanıma bağlı olarak (Nvidia Geforce GTX 1070 Tİ) ortalama (20,15) FPS performansı gösterirken Mask R-CNN (6,95) FPS performansı göstermektedir.



Grafik 5.1. FPS Karşılaştırması

6. SONUÇ VE ÖNERİLER

Bu çalışmada, bir İHA kamerası üzerinden elde edilen görüntülerdeki araçların bölge-tabanlı segmentasyon ile bir derin öğrenme mimarisi olan ESA mimarisi ile geliştirilen Mask R-CNN algoritması ve DarkNet-53 mimarisi üzerine geliştirilen FPN ve ResNet ağlarının hibrit bir versiyonu olan YOLO aracılığıyla araç tespiti yapılmıştır.

Çalışma, öncelikle evrişimsel sinir ağlarının ve bölgesel evrişimsel sinir ağlarının teorik açıklamaları ile başlamaktadır. Ardından literatürde hâlihazırda bulunan evrişimsel sinir ağlarının yapısı incelenmiştir. Son kısımda ise her iki derin sinir ağı ile araç tespitinin nasıl yapıldığı detaylı bir şekilde anlatılmıştır.

Performans değerlendirmesi literatürde sıklıkla kullanılan şu metriklere göre yapılmıştır: Kesinlik, F1 Skoru, duyarlılık, kalite ve işlem süresi. Her iki derin sinir ağı ile aynı veri seti üzerinden eğitilen modeller ilgili metrikler doğrultusunda karşılaştırıldığında elde edilen sonuçlara göre Mask R-CNN'in araçları daha doğru tespit ettiği görülmüştür. YOLO ise araçları daha hızlı tespit etmiştir. Görüntülerden elde edilen araçların segmentasyonu, Mask R-CNN algoritmasında bulunan örnek bölütleme özelliği sayesinde farklı renklendirmeler yolu ile tespit edilmiş, hata oranları yapılan testler sonucunda en aza indirgenmiştir. Çalışmada tek bir sınıf olan araç tespitine odaklanılmıştır. Ancak çalışma çoklu sınıflar ile zenginleştirilerek yeryüzü şekilleri, yerleşim merkezleri, çoklu hedef tespiti gibi farklı amaçlarla kullanılarak geliştirilebilir. Ayrıca farklı açılardan elde edilebilecek daha fazla sayıda araç görüntüleri eğitim sürecinden geçirilerek, daha iyi bölütleme ve nesne tespit işlemleri yapılabilir. Veri setinde sadece gündüz çekilen görüntüler bulunmaktadır. Farklı ışık koşulları (Gece, sabah ve akşam saatleri) eklenerek veri seti genişletilebilir. Yine farklı çevresel koşullar (kar, yağmur vb.) eklenerek veri seti genişletilebilir.

Çalışmanın ek sistemler ve yazılımlarla şu şekilde geliştirilebileceği öngörülmektedir: İnsansız hava aracından GPS bilgileri alınıp tespit edilen araçların koordinatları belirlenebilir. Ayrıca modeller belirli bir araca göre eğitilip (sadece özel bir araç için örneğin siyah renkli sedan araç) takip edilebilir. Bununla beraber sabit duran araçlar dikkate alınmayıp sadece hareket halindeki araçlar maskelenebilir. Bahse konu olan bu senaryolar başka bir çalışmanın konusu olabileceği gibi mevcut modeller üzerinde çeşitli düzenlemeler yapılıp gerçekleştirilebilir.

KAYNAKÇA

Abdulla, W. (2017) . Mask R-CNN for Object Detection and Segmentation. [Erişim: 15.02.2021, https://github.com/matterport/Mask_RCNN]

Abdulla, W. (2017). Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow. [Erişim: 15.01.2021, https://github.com/matterport/Mask_RCNN]

Almog, U. (2020). Yolo v3 Explained. [Erişim: 20.02.2021, https://towardsdatascience.com/yolo-v3-explained-ff5b850390f]

Amidi, S., & A, A. (2019). Convolutional Neural Networks.[Erişim: 30.01.2021, https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks]

Buric, M. vd. (2018). Ball detection using Yolo and Mask R-CNN. 2018 International Conference on Computational Science and Computational Intelligence (CSCI)

Deshpande, A. (2020). A Beginner's Guide To Understanding Convolutional Neural Networks. [Erişim: 12.01.2021, https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/]

Dutta, A., & Zisserman, A. (2019). The VIA Annotation Software for Images, Audio and Video. Proc. 27th ACM International Conference on Multimedia (MM '19),2019, pp. 2276–2279

Gandhi, R. (2018). R-CNN, Fast R-CNN, Faster R-CNN, YOLO - Object Detection Algorithms. [Erişim: 03.02.2021, https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection -algorithms-36d53571365e]

Girshick, R. (2015). Fast R-CNN. International Conference on Computer Vision 2015

Girshick, R. vd. (2014). Rich feature hierarchies for accurate object detection and semantic segmentationTech report (v5). 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)

He, K. vd. (2015). Deep Residual Learning for Image Recognition . 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)

He, K. vd. (2018). Mask R-CNN. Computer Vision and Pattern Recognition

He, K. vd. (2014). Mask R-CNN. in Proc. IEEE Conference on Computer Vision and Pattern Recognition, 2014, p. 2380-750

İnik, Ö., & Ülker E. (2017). "Derin Öğrenme ve Görüntü Analizinde Kullanılan Derin Öğrenme Modelleri", Vol 6, pp. 85-104, Dec. 2017.

Kharuzhy, Y. (2018). Aerial Car Dataset. [Erişim: 15.02.2021, https://github.com/jekhor/aerial-cars-dataset]

Korkmaz, C. (2020). Donanmaların Etkinliğinin Artırılmasında İnsansız Hava Araçlarının Rolü, Türkiye İnsansız Hava Araçları Dergisi, 2(2); 49-54

Kurt, F. (2018). Evrişimli Sinir Ağlarında Hiper Parametrelerin Etkisinin İncelenmesi. [Erişim: 05.02.2021, http://www.openaccess.hacettepe.edu.tr:8080/xmlui/bitstream/hand le/ 11655/4536/10192378.pdf ?sequence=1&isAllowed=y]

Krizhevsky, A. vd., (2017). Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, pp. 1097-1105, 2012.

Lecun, Y. vd. (1998). Gradient Based Learning Applied to Document Recognition. Proc. of the IEEE November 1998

Li, F. vd. (2017). Detection and Segmentation. [Erişim: 01.02.2021, http://cs231n.stanford.edu/slides/ 2017/cs231n_2017_lecture11.pdf]

Liu, M. vd. (2020) "UAV-YOLO: Small Object Detection on Unmanned Aerial Vehicle Perspective", Sensors Journal, vol. 20 , pp. 1-12, Apr. 2020.

Lin, T. (2017). LabelImg. [Erişim: 17.02.2021, https://github.com/tzutalin/labelImg]

Lin, Y. vd. (2015). Microsoft COCO: Common Objects in Context. European Conference on Computer Vision

National Highway Traffic Safety Administration (NHTSA) (2020). Preliminary Statement of Policy Concerning Automated Vehicles, Current Issues in Education, [Erişim: 11.02.2021, http://www.nhtsa.gov/staticfiles/rulemaking/pdf/Automated_Vehicles_Policy.pdf]

Özdağ, M.E. (2019), "Derin Öğrenme Teknikleri Kullanılarak Anayol Trafik Analizi," Y.Lisans Tezi, Karabük Üniv. Lisansüstü Eğit. Enst. , Karabük, MA, 2019.

Prabhu, R. (2020) "Understanding of Convolutional Neural Network (CNN) - Deep Learning", Course Notes, [Erişim: 25.02.2021, https://medium.com/@RaghavPrabhu/ understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148]

Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. Tech Report

Rubik's Code. (2020). Introduction to Convolutional Neural Networks, [Erişim: 25.02.2021, https://rubikscode.net/2018/02/26/introduction-to-convolutional-neural-networks/].

Russakovsky, O. vd. (2015). ImageNet Large Scale Visual Recognition Challenge. Computer Vision and Pattern Recognition

Saha, S. (2020). A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way. [Erişim: 26.02.2021, https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53].

Sharma, P. (2019). Everything about Pooling layers and different types of Pooling, [Erişim: 25.02.2021, https://iq.opengenus.org/pooling-layers/].

Shaoqing, R. vd. (2017). "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol:39/6, pp 1137-1149, June 2017.

Szegedy, C. vd. (2014). Going deeper with convolutions. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)

Şeker, A. vd. (2017), "Derin Öğrenme Yöntemleri ve Uygulamaları Hakkında Bir İnceleme", Vol 3(3), pp. 47-64, Nov. 2017.

Türk Bilim Araştırma Vakfı (TBAV) (2020). Derin Öğrenm Yöntemleri Kullanarak Gerçek Zamanlı Araç Tespiti. [Erişim: 25.02.2021, https://dergipark.org.tr/tr/download/issue-file/34765]

Vemula, S., & Frye, **M.** (2020) "Mask R-CNN Powerline Detector: A Deep Learning approach with applications to a UAV" in Proc. IEEE 39th Digital Avionics Systems Conference (DASC), 2020, pp. 1-6

Yayla, R., & Şen, B. (2019a). "Research on Region-Based Convolutional Neural Network for Semantic Segmentation", in Proc. 8th International Conference on Advanced Technologies Conference (ICAT'19), 2019, pp. 244-249

Yayla, R., & Şen, B. (2019b). "Region-based Segmentation of Terrain Fields in SAR Images", in Proc 28th Signal Processing and Communications Applications Conference Conference (SIU2020), 2019, pp. 1-4