



**HAVA VE UYDU GÖRÜNTÜLERİ ÜZERİNDE DERİN ÖĞRENME
TABANLI ARAÇ TESPİT VE SINIFLANDIRMASI**

Ömer ER

**YÜKSEK LİSANS TEZİ
ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ ANA BİLİM DALI**

**GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ**

AĞUSTOS 2020

Ömer ER tarafından hazırlanan “HAVA VE UYDU GÖRÜNTÜLERİ ÜZERİNDE DERİN ÖĞRENME TABANLI ARAÇ TESPİT VE SINIFLANDIRMASI” adlı tez çalışması aşağıdaki jüri tarafından OY BİRLİĞİ ile Gazi Üniversitesi Elektrik Elektronik Mühendisliği Ana Bilim Dalında YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

Danışman: Prof. Dr. Hasan Şakir BİLGE

Elektrik Elektronik Mühendisliği Ana Bilim Dalı, Gazi Üniversitesi

Bu tezin, kapsam ve kalite olarak Yüksek Lisans Tezi olduğunu onaylıyorum.

.....

Başkan: Doç. Dr. Ahmet Murat ÖZBAYOĞLU

Bilgisayar Mühendisliği Ana Bilim Dalı, TOBB-ETÜ

Bu tezin, kapsam ve kalite olarak Yüksek Lisans Tezi olduğunu onaylıyorum.

.....

Üye: Prof. Dr. Fırat HARDALAÇ

Elektrik Elektronik Mühendisliği Ana Bilim Dalı, Gazi Üniversitesi

Bu tezin, kapsam ve kalite olarak Yüksek Lisans Tezi olduğunu onaylıyorum.

.....

Tez Savunma Tarihi: 31/08/2020

Jüri tarafından kabul edilen bu tezin Yüksek Lisans Tezi olması için gerekli şartları yerine getirdiğini onaylıyorum.

.....

Prof. Dr. Cevriye GENCER
Fen Bilimleri Enstitüsü Müdürü

ETİK BEYAN

Gazi Üniversitesi Fen Bilimleri Enstitüsü Tez Yazım Kurallarına uygun olarak hazırladığım bu tez çalışmada;

- Tez içinde sunduğum verileri, bilgileri ve dokümanları akademik ve etik kurallar çerçevesinde elde ettiğimi,
- Tüm bilgi, belge, değerlendirme ve sonuçları bilimsel etik ve ahlak kurallarına uygun olarak sunduğumu,
- Tez çalışmada yararlandığım eserlerin tümüne uygun atıfta bulunarak kaynak gösterdiğimi,
- Kullanılan verilerde herhangi bir değişiklik yapmadığımı,
- Bu tezde sunduğum çalışmanın özgün olduğunu,

bildirir, aksi bir durumda aleyhime doğabilecek tüm hak kayıplarını kabullendiğimi beyan ederim.

.....
Ömer ER
31/08/2020

HAVA VE UYDU GÖRÜNTÜLERİ ÜZERİNDE DERİN ÖĞRENME TABANLI ARAÇ TESPİT VE SINIFLANDIRMASI

(Yüksek Lisans Tezi)

Ömer ER

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

Ağustos 2020

ÖZET

Hava görüntüleri üzerinde araç tespit ve sınıflandırması; istihbarat, keşif ve gözetleme açısından önemlidir. Ancak bu görev; düşük uzamsal çözünürlük, karmaşık arka plan, nesne üzerine düşen ışık/gölge farklılıkları ve nesnelerin çevre tarafından kamufle olması gibi sebeplerle zordur. Son zamanlarda geliştirilen CNN tabanlı ağlar umut vericidir ancak bu ağlar doğrudan küçük nesnelerin tespiti için yeterli değildirler ve ince ayara ihtiyaç duyarlar. Bu çalışmada daha hızlı RCNN algoritması ve görece büyük nesnelerin tespitinde başarısı kanıtlanmış ResNet modeli ile araba, pikap, kamyon, uçak, kamp arabası, panelvan, traktör, iş makinası ve tekne gibi görünüm ve boyut olarak birbirinden farklı 9 sınıfı barındıran VEDAI veri kümesi üzerinde çalışılmıştır. Nesnelerin görüntüdeki toplam piksellerin 0.5×10^{-3} 'ü kadar az yer kapladığı görüntüler üzerinde araç tespitinde başarıyı artırımı için daha hızlı RCNN algoritmasında değişiklikler ile çeşitli deneyler yapılmıştır. Deneyler sonucunda VEDAI veri kümesi üzerindeki araçların tespitinde %74.9 ortalama hassasiyet elde etmenin mümkün olduğu gösterilmiştir. Ayrıca küçük nesne tespiti için daha sıkı ağların daha faydalı olabileceği sonucuna ulaşılmıştır. Araç tespitinde en çok başarı sağlayan deney modeli ile araçların sınıflandırılması gerçekleştirilmiş ve araba için %72.4, kamyon için %58.6, pikap için %72, traktör için %57.6, kamp arabası için %69.9, tekne için %60, panelvan için %56.4, iş makinası için %53.1 ve uçak için %85.7 ortalama hassasiyet elde edilmiştir.

Bilim Kodu : 90542
Anahtar Kelimeler : Nesne tespiti, derin öğrenme, hava görüntüleri
Sayfa Adedi : 88
Danışman : Prof. Dr. Hasan Şakir BİLGE

DEEP LEARNING BASED VEHICLE DETECTION AND CLASSIFICATION ON AIR AND SATELLITE IMAGES

(M. Sc. Thesis)

Ömer ER

GAZİ UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

August 2020

ABSTRACT

Vehicle detection and classification on aerial images are important for intelligence, discovery and surveillance. However, this task is difficult due to low spatial resolution, complex background, light/shadow differences on the object and objects being camouflaged by the environment. Recently developed CNN-based networks are promising, but they are not sufficient for direct detection of small objects and need fine tuning. In this study, VEDAI dataset containing 9 different classes in terms of appearance and size such as car, pickup, truck, plane, camper, van, tractor, work machine and boat with faster RCNN algorithm and proven ResNet model was studied. Various experiments have been carried out with changes in the faster RCNN algorithm for performance enhancement in vehicle detection on images where objects occupy as little as 0.5×10^{-3} pixels of the total image. As a result of the experiments, it has been shown that it is possible to achieve an average sensitivity of 74.9% in the detection of vehicles on the VEDAI dataset. It was also concluded that shallower networks may be more useful for small object detection. The classification of the vehicles has been realized with the most successful experiment model in vehicle detection and 72.4% for the car, 58.6% for the truck, 72% for the pickup, 57.6% for the tractor, 69.9% for the camper, 60% for the boat, 56.4% for the van, work average precision of 53.1% for machine and 85.7% for airplane.

Science Code : 90542

Key Words : Object detection, deep learning, aerial images

Page Number : 88

Supervisor : Prof. Dr. Hasan Şakir BİLGE

TEŞEKKÜR

Tez süreci boyunca yardımlarını esirgemeyen danışman hocam Prof. Dr. Hasan Şakir BİLGE'ye, çalışma hayatım ve yüksek lisans eğitimim arasındaki dengeyi kolaylaştıran Aselsan'a ve desteğiyle hep yanımda olan Ali ER'e çok teşekkür ederim.

İÇİNDEKİLER

	Sayfa
ÖZET	iv
ABSTRACT.....	v
TEŞEKKÜR.....	vi
İÇİNDEKİLER	vii
ÇİZELGELERİN LİSTESİ.....	ix
ŞEKİLLERİN LİSTESİ.....	x
RESİMLERİN LİSTESİ	xii
SİMGELER VE KISALTMALAR.....	xiii
1. GİRİŞ.....	1
2. KONVOLÜSYONEL SİNİR AĞLARI.....	3
2.1. Giriş Katmanı	3
2.2. Konvolüsyon Katmanı	3
2.3. Yığın Normalizasyonu Katmanı	6
2.4. ReLu Katmanı	7
2.5. Havuzlama Katmanı.....	7
2.6. Tam Bağlantılı Katman	8
2.7. Çıktı Katmanları.....	9
2.7.1. Softmaks ve sınıflandırma katmanları	9
2.7.2. Regresyon katmanı.....	10
3. RESNET	11
4. DAHA HIZLI RCNN ALGORİTMASININ GELİŞİMİ VE UYGULANMASI	19
4.1. RCNN Algoritması.....	19

	Sayfa
4.2. Hızlı RCNN Algoritması.....	21
4.3. Daha Hızlı RCNN Algoritması	22
4.3.1. Bölge önerileri ağı.....	23
4.3.2. Tespit Ağı.....	25
4.4. Daha Hızlı RCNN Algoritmasının Uygulanması.....	25
5. VERİ KÜMESİ VE VERİ ARTIRIMI	29
6. DAHA HIZLI RCNN VE RESNET MODİFİKASYONLARI.....	33
6.1. Çapa Kutu Sayısı ve Boyutu	33
6.2. Konvolüsyon Katman Sayısı.....	34
6.3. Özellik Haritası Boyutu.....	39
7. DENEYSEL ÇALIŞMALAR	41
7.1. Çapa Kutu Sayısı ve Boyutunun Etkisi	43
7.2. Konvolüsyon Katman Sayısının Etkisi	47
7.3. Özellik Haritası Boyutunun Etkisi	53
7.4. Sonuçların Karşılaştırılması	55
8. ARAÇLARIN SINIFLANDIRILMASI.....	59
9. SONUÇLAR	77
KAYNAKLAR	79
EKLER.....	83
EK-1. ResNet101 Ağ Mimarisi	84
ÖZGEÇMİŞ.....	88

ÇİZELGELERİN LİSTESİ

Çizelge	Sayfa
Çizelge 3.1. ResNet ağı katman yapısı	14
Çizelge 5.1. VEDAI veri kümesi araç sayısı	30
Çizelge 6.1. Kullanılan çapa kutuları.....	34
Çizelge 7.1. Eğitim Parametreleri	41
Çizelge 7.2. Farklı bağlantı kutularına ait sonuçlar	46
Çizelge 7.2. Farklı derinlikteki ağlara ait sonuçlar	47
Çizelge 7.3. Daha hızlı RCNN ResNet18_conv3 modeli test sonuçları.....	54
Çizelge 7.4. Farklı çalışmaların karşılaştırılması.....	57
Çizelge 8.1. Daha hızlı RCNN ResNet50_classification modeline ait algıma sonuçları	60
Çizelge 8.2. Daha hızlı RCNN ResNet50_eksi6_tl_classification modeline ait algıma sonuçları	62

ŞEKİLLERİN LİSTESİ

Şekil	Sayfa
Şekil 2.1. $5 \times 5 \times 3$ görüntü ve 3×3 filtrenin konvolüsyon işlemi	4
Şekil 2.2. Konvolüsyon katmanında dolgu (sol) ve genişleme (sağ) işlemleri.....	5
Şekil 2.3. 5×5 konvolüsyonun işlemi; doğrudan(üst), öncesinde 1×1 konvolüsyon ile uygulanması (alt)	6
Şekil 2.4. 5×5 görüntüde 2×2 filtre ile 1 ve 2 adım kullanılarak maksimum havuzlama	8
Şekil 3.1. ResNet18’de kullanılan artık blok yapıları	12
Şekil 3.2. Resnet50 ve ResNet100’de kullanılan artık blok yapıları	13
Şekil 3.3. ResNet18 ağ mimarisi.....	15
Şekil 3.4. ResNet50 Ağ mimarisi	16
Şekil 4.1. Seçici arama algoritmasının işleyişi	20
Şekil 4.2. RCNN algoritması çalışma prensibi	20
Şekil 4.3. RoI havuzlama katmanı	21
Şekil 4.4. Hızlı RCNN algoritması çalışma prensibi	22
Şekil 4.5. Hızlı RCNN algoritmasının çalışma prensibi	23
Şekil 4.6. Bölge Önerileri Ağı çalışma prensibi	24
Şekil 4.7. Nesne tespitine uygun ResNet50 ağının çıktı katmanları.....	26
Şekil 4.8. RPN ağı.....	28
Şekil 5.1. Veri artırımının araç koordinatlarına etkisi	32
Şekil 6.1. Daha hızlı RCNN ResNet18 modeli bölge önerileri ağı	35
Şekil 6.2. Daha hızlı RCNN ResNet101 modeli bölge önerileri ağı	36
Şekil 6.3. Daha hızlı RCNN Resnet18_artı4 modeli conv2_x konvolüsyon bloğu.....	38
Şekil 6.4. Daha hızlı RCNN ResNet50_eksi6 modeli	39
Şekil 6.5. Daha hızlı RCNN ResNet18_conv3 modeli	40

Şekil	Sayfa
Şekil 7.1. a) Ankor1 modelinin geri çağırma-hassasiyet grafiği b) Ankor2 modelinin geri çağırma hassasiyet grafiği c) Ankor3 modelinin geri çağırma hassasiyet grafiği d) Ankor4 modelinin geri çağırma hassasiyet grafiği e) Ankor5 modelinin geri çağırma hassasiyet grafiği.....	43
Şekil 7.2. Farklı çapa boyut ve sayılarına sahip modellerin R-P grafiği	46
Şekil 7.3. a) Daha hızlı RCNN ResNet18 modelinin geri çağırma-hassasiyet grafiği b) Daha hızlı RCNN ResNet101 modelinin geri çağırma-hassasiyet grafiği c) Daha hızlı RCNN ResNet18_artı4 modelinin geri çağırma-hassasiyet grafiği d) Daha hızlı RCNN ResNet50_eksi6 modelinin geri çağırma-hassasiyet grafiği e) Daha hızlı RCNN ResNet50_eksi6_tl modelinin geri çağırma-hassasiyet grafiği	50
Şekil 7.4. Farklı konvolüsyon katman sayısına sahip modellerin R-P grafiği.....	52
Şekil 7.5. Daha hızlı RCNN ResNet18_conv3 modelinin geri çağırma – hassasiyet grafiği	54
Şekil 7.6. Farklı modellere ait geri çağırma – hassasiyet grafiği.....	54
Şekil 8.1. Daha hızlı RCNN ResNet50_classification modelinin her bir sınıfa ait R-P grafiği	60
Şekil 8.2. Daha hızlı RCNN ResNet50_eksi6_tl_classification modelinin her bir sınıfa ait R-P grafiği.....	62
Şekil 8.3. Ağ modellerinin araba sınıfı geri çağırma-hassasiyet grafiği.....	63
Şekil 8.4. Ağ modellerinin kamyon sınıfı geri çağırma-hassasiyet grafiği.....	65
Şekil 8.5. Ağ modellerinin pikap sınıfı geri çağırma-hassasiyet grafiği.....	66
Şekil 8.6. Ağ modellerinin traktör sınıfı geri çağırma-hassasiyet grafiği.....	68
Şekil 8.7. Ağ modellerinin kamp arabası sınıfı geri çağırma-hassasiyet grafiği	69
Şekil 8.8. Ağ modellerinin tekne sınıfı geri çağırma-hassasiyet grafiği.....	71
Şekil 8.9. Ağ modellerinin panelvan sınıfı geri çağırma-hassasiyet grafiği	73
Şekil 8.10. Ağ modellerinin iş makinesi sınıfı geri çağırma-hassasiyet grafiği	74
Şekil 8.11. Ağ modellerinin uçak sınıfı geri çağırma-hassasiyet grafiği.....	76

RESİMLERİN LİSTESİ

Resim	Sayfa
Resim 5.1. VEDAI veri kümesinden örnekler	30
Resim 5.2. Veri artırımı	31
Resim 7.1. Daha hızlı RCNN ResNet50_eksi6_tl modeline ait algılama sonuçları	56
Resim 8.1. Daha hızlı RCNN ResNet50 classification modeline ait bir araba tespit örneği(sol) ve aynı görüntü üzerinde daha hızlı RCNN ResNet50_eksi6_tl classification modeline ait bir araba tespit örneği(sağ)	64
Resim 8.2. Daha hızlı RCNN ResNet50 classification modeline ait bir kamyon tespit örneği(sol) ve aynı görüntü üzerinde daha hızlı RCNN ResNet50_eksi6_tl classification modeline ait bir kamyon tespit örneği(sağ)	65
Resim 8.3. Daha hızlı RCNN ResNet50 classification modeline ait bir pikap tespit örneği(sol) ve aynı görüntü üzerinde daha hızlı RCNN ResNet50_eksi6_tl classification modeline ait bir pikap tespit örneği(sağ)	67
Resim 8.4. Daha hızlı RCNN ResNet50 classification modeline ait bir traktör tespit örneği(sol) ve aynı görüntü üzerinde daha hızlı RCNN ResNet50_eksi6_tl classification modeline ait bir traktör tespit örneği(sağ)	69
Resim 8.5. Daha hızlı RCNN ResNet50 classification modeline ait bir kamp arabası tespit örneği(sol) ve aynı görüntü üzerinde daha hızlı RCNN ResNet50_eksi6_tl classification modeline ait bir kamp arabası tespit örneği(sağ)	70
Resim 8.6. Daha hızlı RCNN ResNet50 classification modeline ait bir tekne tespit örneği(sol) ve aynı görüntü üzerinde daha hızlı RCNN ResNet50_eksi6_tl classification modeline ait bir tekne tespit örneği(sağ)	72
Resim 8.7. Daha hızlı RCNN ResNet50 classification modeline ait bir panelvan tespit örneği(sol) ve aynı görüntü üzerinde daha hızlı RCNN ResNet50_eksi6_tl classification modeline ait bir panelvan tespit örneği(sağ)	74
Resim 8.8. Daha hızlı RCNN ResNet50 classification modeline ait bir iş makinesi tespit örneği(sol) ve aynı görüntü üzerinde daha hızlı RCNN ResNet50_eksi6_tl classification modeline ait bir iş makinesi tespit örneği(sağ)	75
Resim 8.9. Daha hızlı RCNN ResNet50 classification modeline ait bir uçak tespit örneği(sol) ve aynı görüntü üzerinde daha hızlı RCNN ResNet50_eksi6_tl classification modeline ait bir uçak tespit örneği(sağ)	76

SİMGELER VE KISALTMALAR

Bu çalışmada kullanılmış simgeler ve kısaltmalar, açıklamaları ile birlikte aşağıda sunulmuştur.

Simgeler

cm

°

GHz

GB

Açıklamalar

Santimetre

Derece

Giga Hertz

Gigabayt

Kısaltmalar

AGRC

AP

CNN

CPU

DPM

GPU

HOG

ILSVRC

IOU

İHA

MSCOCO

MSE

RCNN

RELU

RESNET

ROI

RPN

SIFT

SVM

Açıklamalar

Automated Geographic Reference Center

Average Precision

Convolutional Neural Network

Central Process Unit

Deformable Part Models

Graphic Processing Unit

Histogram of Oriented Gradient

ImageNet Large Scale Visual Recognition Challenge

Intersection Over Union

İnsansız Hava Aracı

Microsoft Common Objects in Context

Mean Square Error

Region-based Convolutional Network

Rectified Linear Unit

Residual Network

Region of Interest

Region Proposal Network

Scale Invariant Feature Transform

Support Vector Machine

Kısaltmalar**Açıklamalar****VEDAI**

Vehicle Detection in Aerial Imagery

YOLO

You Only Look Once

1. GİRİŞ

İnsansız hava aracı (İHA) ve uydu teknolojisinin ilerlemesi, bunların kullanımının yaygınlaşması; elde edilen hava görüntü verilerinin artmasına ve bu alanda çeşitli uygulamaların geliştirilmesine olanak sağlamıştır. Bu uygulamaların büyük bir kısmı geniş alanları kapsayan hava görüntülerinin taranmasına dayanmaktadır ve ilgili tüm nesnelerin doğru algılanması ihtiyacını paylaşırlar. Buradaki temel amaç, hedef tanıma görevinden insan faktörünü çıkarmak ve otomatik bir hedef tanıma sistemi geliştirmektir. Özel olarak hava görüntüleri üzerinden hedef tanıma görevlerinden bir tanesi de araç tespiti ve sınıflandırılmasıdır. Hava görüntüleri üzerinden araç tespit ve sınıflandırılması; trafik yönetimi, akıllı ulaşım sistemlerinin tasarımı, afet yönetimi, otopark yönetimi, şehir planlama gibi birçok alan için ve özellikle istihbarat, askeri keşif ve gözetleme için önemlidir. Ancak gelişen uydu ve İHA elektro optik sistemlerine rağmen halen bu zorlu bir iştir. Bunun sebeplerinden ilki, yatay görüntüleme ile alınan görüntülere kıyasla hava görüntülerinde nesnelerin çok daha az bilgi içerebiliyor olmasıdır. Bunun temel etmeni uzak mesafedir. Ayrıntılar çok yüksek çözünürlüklü görüntüde dahi çok sınırlıdır. Bir diğer sebep, hava görüntülerinde araç benzeri nesnelerin çokluğuudur. Evlerin çatıları, klima üniteleri, çöp kutuları ve elektrik üniteleri gibi birçok nesne yukarıdan bakıldığında araçlar ile karıştırılabilmektedir. Sebeplerden bir tanesi de nesnelerin geleneksel nesne algılama çalışmalarında olduğu gibi yerçekimin etkisi ile tek bir yön doğrultusunda bulunmamasıdır. Hava görüntülerinde araçlar rastgele yönlere sahiptir ve bu durum tespitlerini zorlaştırmaktadır [1]. Araçları hava görüntülerinde tespit etmenin diğer zorlukları olarak; karmaşık arka plan, araçların üzerine düşen ışık ve gölge farklılıkları ve araçların bazen bir ağaçla bazen bir trafik işareti ile çevre tarafından kamufle olması belirtilebilir.

Nesne tespit teknolojisi ile ilgili olarak bugüne kadar önerilen yöntemler; sık ve derin öğrenmeye dayalı olarak ikiye ayrılabilir. Sık öğrenmeye dayalı yöntemler, el yapımı özellik çıkarma metotları ve sınıflandırıcı veya sınıflandırıcıların kombinasyonlarından oluşan yöntemler uygulanarak geliştirilmiştir. Bu yöntemlere göre; ilk önce yönelimli gradyan histogramı (HOG) [2] ve ölçekten bağımsız öznelik dönüşümü (SIFT) [3] gibi çok yaygın özellik çıkarma metotları ile nesnelerin öznelikleri çıkarılmış, ardından destek vektör makinası (SVM) [4] ve Ada-Boost [5] gibi sınıflandırıcıların bir tanesi veya birden fazlası kullanılarak nesnelerin tespiti sağlanmıştır. Bu yöntemler nesneyi tek bir katı şablon olarak

modellerken, deęiřime uğrayabilen parça bazlı model (DPM) [6] nesneyi uzaysal olarak organize edilmiř parçaların kombinasyonu řeklinde modellemiřtir. DPM de dahil olmak üzere sığ öğrenmeye dayalı yöntemler, kayan pencere prensibi ile çalışırlar. Kayan pencere prensibine göre bir pencere tüm görüntü üzerinde kayarak nesnelerin tespitini sağlar, bu çok zaman alıcıdır. Derin öğrenmeye dayalı yöntemler ise Krizhevsky ve arkadaşlarının 2012 yılında ILSVRC2012 [7] yarışmasında büyük ses getiren derin konvolüsyonel sinir aęları (CNN) [8]'nı önermesiyle başlamaktadır. İlerleyen dönemlerde bölge tabanlı konvolüsyonel sinir aęları (RCNN) [9] nesne tespitinde önemli performans kazanımı elde etmiřtir. Ardından RCNN'nin görece hız sorununu çözen hızlı RCNN [10] önerilmiřtir. Bir sonraki adımda bölge önerilerini oluşturmak için kendi içinde bölge teklif aęı (RPN) ismi verilen, tamamen konvolüsyonel bir bölge teklif aęı kullanan, daha hızlı RCNN [11] önerilmiřtir. Bölge teklif tabanlı aęlar birkaç nedenden dolayı sığ öğrenmeden daha iyi performans göstermektedir [12]. Bunlardan biri, CNN'nin el yapımı özellik çıkarma metotlarının aksine özellik çıkarma işlemini daha güçlü bir řekilde otomatik olarak yapmasıdır. Dięer sebep ise sığ öğrenmeye dayalı yöntemlerin nesne tespiti için görüntünün tamamını arayan kayar pencere kullanmasının aksine bölge tabanlı CNN'lerin yalnızca önerilen bölgeler üzerinde nesneyi aramasıdır. Bu metot çok daha az zaman alıcıdır.

Son birkaç yılda [13], [14], [15] ve [16] gibi CNN tabanlı yeni algoritmalar geliştirilmiř olsa da bu çalışmada daha hızlı RCNN algoritmasının ve ILSVRC 2015 [17] sınıflandırma yarışmasında birincilik elde eden artık aę (ResNet) [18] modelinin kullanımına odaklanılmıřtır. Bahsedilen algoritmalar, büyük ölçekli nesneleri barındıran ImageNet [19] ve MSCOCO [20] gibi veri kümlelerinde başarı sağlamıřtır. Ancak küçük nesnelerin tespiti için yeterli deęildirler ve bu görev için bu algoritmalara ince ayar yapılması gerekmektedir. Bu kapsamda daha hızlı RCNN algoritması ve ResNet modeli ile hava görüntülerinden küçük nesne olarak deęerlendirebileceğimiz araçların tespitinin başarılı bir řekilde yapılması hedeflenmiř, bu algoritma ve model üzerinde bir takım deneysel deęişiklikler yapılmıřtır. Hava görüntüleri üzerinden araçların tespitinde başarılı bir model oluřturulduktan sonra, hava görüntüleri üzerinde araçların sınıflandırma nihai hedefi bu başarılı model üzerinden gerçekteřirilmıřtir.

2. KONVOLÜSYONEL SİNİR AĞLARI

Konvolüsyonel sinir ağları (CNN), yapay sinir ağlarından geliştirilen bir derin öğrenme modelidir. İlk olarak 1998 yılında LeNet [21] mimarisiyle ortaya atılsa da popülerliğini 2012 yılında Alex Krizhevsky ve arkadaşlarına daha derin bir CNN olan AlexNet [8] mimarisiyle bilgisayarlı görünün olimpiyatları olarak bilinen ImageNet görüntü sınıflandırma yarışmasında birincilik getirerek kazanmıştır. İçerisinde barındırdığı farklı katmanlarla, yapay sinir ağlarının parametre sayısının fazlalığından kaynaklanan işlem yükü ve yavaşlık dezavantajlarını yenmiştir [22]. Katmanların bir ağda bulunma sayısı, sırası vb. durumlar mimariden mimariye değişiklik göstermektedir. Bu bölümde bu çalışmanın bir parçası olan ResNet mimarisinde kullanılan konvolüsyonel sinir ağları katmanları üzerinde durulmaktadır.

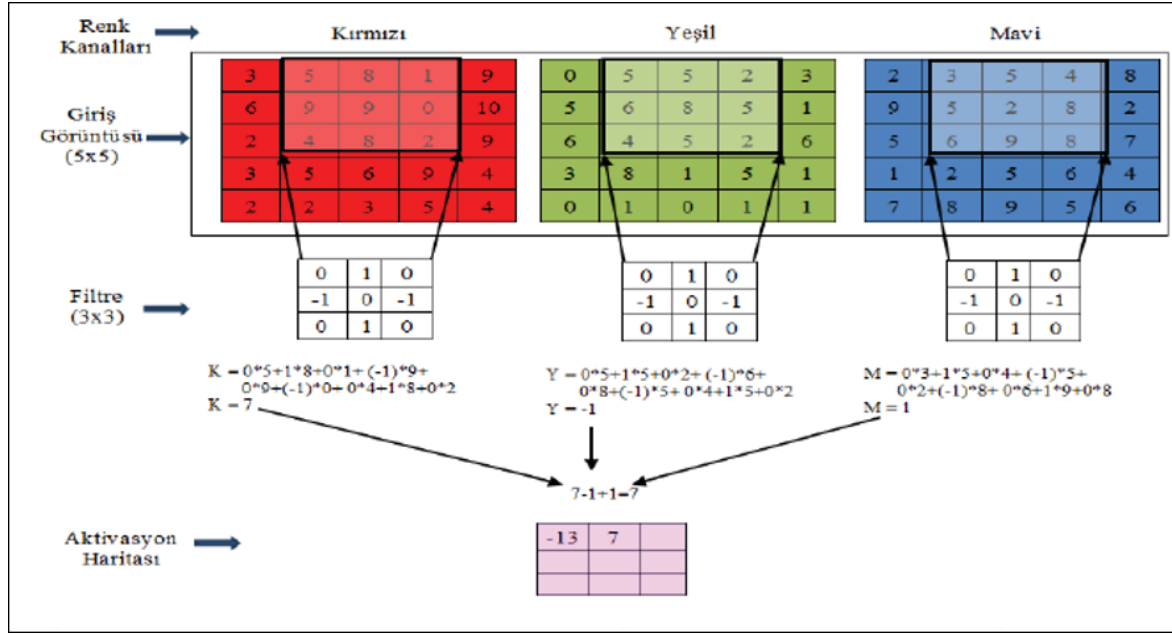
2.1. Giriş Katmanı

Görüntünün ham olarak ağa beslendiği katmandır. Bu katmanda kullanıcının seçimine bağlı olarak normalizasyon işlemi uygulanabilmektedir. Normalizasyon işleminin uygulanması durumunda, giriş görüntüsü sıfır ortalama ve birim varyansa sahip olmaktadır [23].

2.2. Konvolüsyon Katmanı

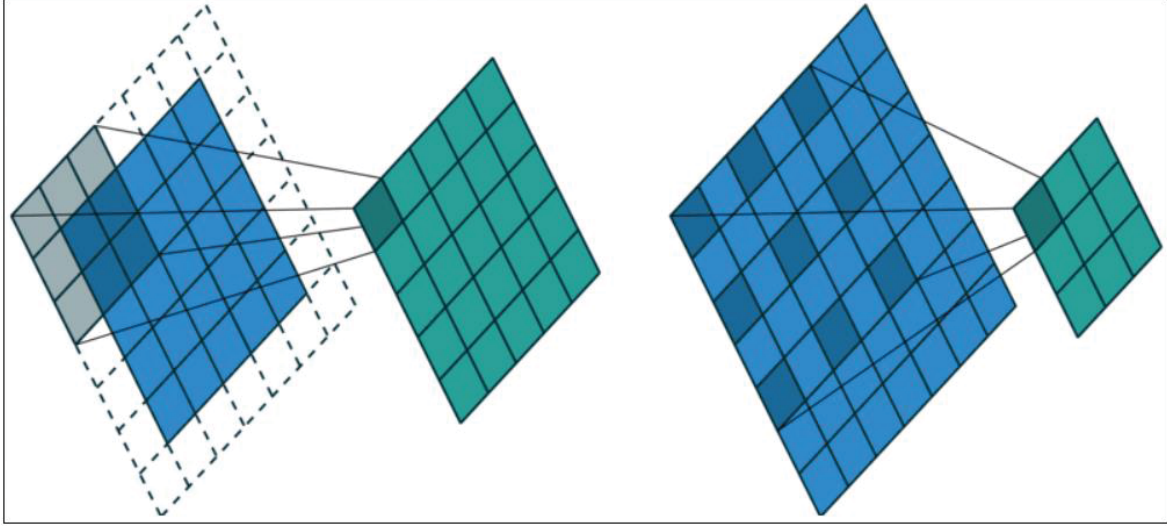
Bu katman belirli filtrelerin girdi üzerinde dolaştırılmasına dayanan bir dönüşüm katmanıdır. Bir önceki katmandan gelen girdiye, filtreler konvolüsyon işlemi uygular ve çıkış verisini oluştururlar. Konvolüsyon işlemi, filtre ağırlıkları ile filtrenin bulunduğu konuma karşılık gelen girdi verilerinin skalar çarpımıdır [24]. Girdi $G(i,j)$ ve filtre $F(i,j)$ 'nin konvolüsyon işlemi Eş. 2.1'de verilmiştir. Şekil 2.1 [25], $5 \times 5 \times 3$ boyutundaki bir girdiye 3×3 'lük bir filtre ile konvolüsyon işleminin uygulanmasını göstermektedir. Uygulanan her bir filtre sonucunda özellik haritası oluşur. Bu bakımdan; özellik haritaları, her bir filtrenin keşfettiği özelliği temsil eden bölgelerdir ve özellik haritalarının sayısı filtre sayısına eşittir. CNN'nin eğitimi esnasında filtre ağırlıkları her bir öğrenme yinelemesinde güncellenir. Böylelikle ağ, verinin hangi bölgesinin önem taşıdığını belirleyerek, verinin özelliklerini çıkarır [25].

$$O(i,j) = G(i,j) * F(i,j) = \sum_k \sum_l G(i,j)F(i-k,j-l) \quad (2.1)$$



Şekil 2.1. 5×5×3 görüntü ve 3×3 filtrenin konvolüsyon işlemi [25]

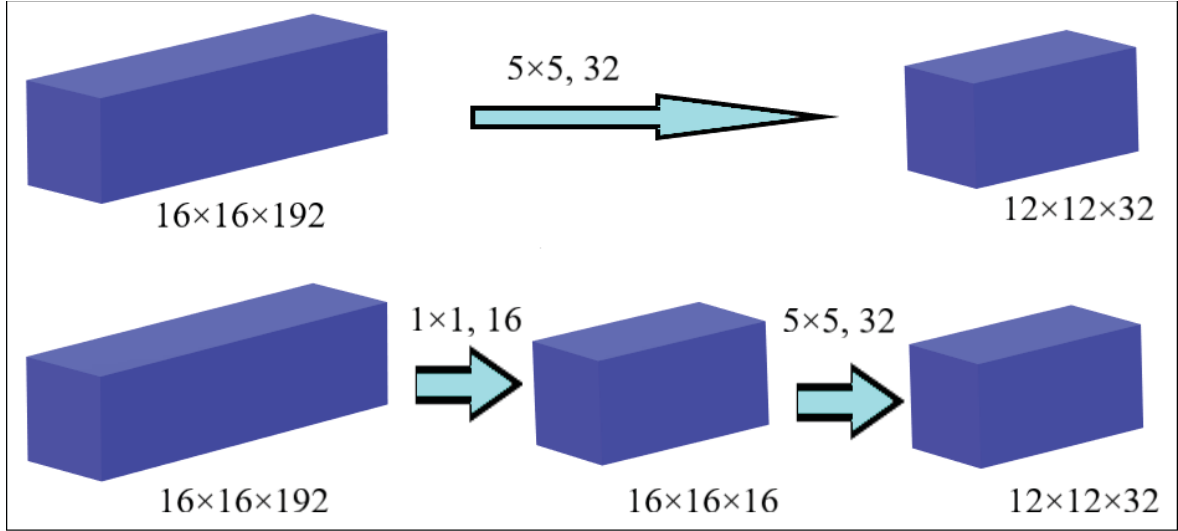
Konvolüsyon katmanının çıkış boyutunu belirleyen bir takım parametreler vardır. Bunlardan biri filtrenin adım (stride) sayısıdır. Adım sayısı [1,1] olduğunda filtre birer piksel hareket ederken, adım sayısı [2,2] iken filtre yatay ve dikey ekseninde iki piksel atlayarak hareket eder. Adım sayısının artması ağırlık parametre sayısını azaltmakta, daha küçük boyutlu bir çıktının oluşmasını sağlamaktadır [26]. Bu, bir sonraki katmandaki düğümün komşularıyla çok fazla örtüşme olması durumunda uygulanabilecek bir işlemdir [22]. Çıkış boyutunu belirleyen bir diğer parametre dolgudur (padding). Dolgu, bir görüntü girişinin dikey ve yatay sınırlarına sıfır piksellerinin belirli satır ve sütun adedince eklenmesi işlemidir. Bu işlem, konvolüsyonun görüntünün kenarlarında neden olacağı bilgi kaybını önlemek için faydalıdır [22]. Şekil 2.2 [27], [1,1]'lik dolgu uygulanan görüntü üzerinde 3×3'lik bir filtrenin işleyişini soldaki figürle göstermiştir. Genişleme (dilate) faktörü çıktı boyutunu belirleyen bir diğer parametredir. Genişleme faktörü filtre ağırlıkları arasına ne kadar sıfır eklenip, filtrenin hangi oranda genişleyeceğini belirlemektedir. Genişleme faktörünün [1,1] olduğu durum normal konvolüsyon işlemine denk düşerken, [2,2] olduğu durum filtrenin satır ve sütunlarının 1'er piksel, [4,4] olduğunda 3'er piksel kaymasına karşılık gelmektedir. Bu, ağırlık sayısını ve hesaplama maliyetini değiştirmeden daha geniş bir görüş alanı sağlar [27]. Şekil 2.2'de sağdaki figürde 3×3'lük bir filtre [2,2]'lik bir genişleme faktörü ile daha geniş bir görüş açısına sahip olurken, ağırlık sayısı yine 9 olarak kalmıştır. Bir konvolüsyon katmanının çıktı boyutu O, Eş. 2.2 ile ifade edilmektedir.



Şekil 2.2. Konvolüsyon katmanında dolgu (sol) ve genişleme (sağ) işlemleri

$$O = \frac{\text{Giriş Boyutu} - ((\text{Filtre Boyutu} - 1) \times \text{Genişleme} + 1) + 2 \times \text{Dolgu}}{\text{Adım}} + 1 \quad (2.2)$$

Konvolüsyon katmanının özel bir kullanım amacı da verinin boyutunu (yükseklik, genişlik ve derinlik açısından) değiştirmektedir. Bu, 1×1 filtreler kullanılarak yapılmaktadır. Örneğin; $256 \times 256 \times 3$ boyutundaki bir veriye bir adet 1×1 filtre uygulanması durumunda çıktı $256 \times 256 \times 1$ boyutunda olacaktır. Aynı veriye 16 adet 1×1 filtre uygulanması durumunda çıktı boyutu $256 \times 256 \times 16$ olacaktır. Bir başka örnekle, $112 \times 112 \times 64$ boyutundaki bir veriye $[2,2]$ adım sayısına ve 64 adet 1×1 filtreye sahip bir konvolüsyon uygulanması durumunda çıktı $56 \times 56 \times 64$ olacaktır. 1×1 filtre veri boyutunu değiştirmesi bakımından, iki verinin toplanmasını gerektiren ağ mimarilerinde girdilerin aynı boyuta sahip olması gerekliliğini sağlayan önemli bir araç olarak kullanılmaktadır. 1×1 filtrenin bir diğer kullanım amacı ise, işlem yükünü azaltmaktır. Örneğin; Şekil 2.3'te üstte gösterildiği gibi $16 \times 16 \times 192$ boyutundaki bir özellik haritasına 32 adet 5×5 filtresinin uygulanması gerekliliğinde 39 milyon $((16 \times 16 \times 32) \times (5 \times 5 \times 192))$ 'den işlem maliyeti vardır. Aynı giriş özellik haritasına Şekil 2.3'te altta gösterildiği gibi 5×5 filtreden önce 16 adet 1×1 filtre uygulanması durumunda toplamda 4 milyon $((16 \times 16 \times 16) \times (1 \times 1 \times 192) + (16 \times 16 \times 32) \times (5 \times 5 \times 16))$ 'den işlem maliyeti ortaya çıkmaktadır.



Şekil 2.3. 5x5 konvolüsyonun işlemi; doğrudan(üst), öncesinde 1x1 konvolüsyon ile uygulanması (alt)

2.3. Yığın Normalizasyonu Katmanı

Giriş katmanında veriye bir normalizasyon yapılır, ancak ağı kendi içerisindeki dönüşümlerden dolayı ortalama ve varyans sürekli kayar. Bu değişime veriyi uyarlamak için veri yığınının tekrardan normalizasyon işleminin uygulanması yaklaşımı gelişmiştir. Bir başka deyişle, ağı eğitiminde geri yayılım sırasında ağırlıklar değişir ve bu değişime bağlı olarak çıktılar da değişir. Yığın normalizasyonu bu değişimdeki kaymayı azaltarak, eğitimi hızlandırmayı amaçlar. Böylece bu işlem aynı zamanda ağırlıkların ilklendirilmesine olan duyarlılığı da azaltır. Genel olarak konvolüsyon katmanı ile düzeltilmiş lineer birim (ReLU) katmanı gibi doğrusal olmayan aktivasyon fonksiyonları arasında kullanılmaktadır [28].

$B = (x_{1,2,...,m})$ bir mini yığının girdileri olsun. Eş. 2.3, mini yığının ortalamasını; Eş. 2.4, mini yığının varyansını göstermektedir. Normalizasyon Eş.2.5'te gösterildiği gibi, mini yığının her bir girdisine mini yığın ortalamasının çıkarılması ve ardından mini yığın standart sapmasına (σ) bölünmesi işlemlerinin uygulanması ile gerçekleşir.

$$\mu_B = \frac{1}{m} \sum_{i=1}^m x_i \quad (2.3)$$

$$\sigma_B^2 = \frac{1}{m} \sum_{i=1}^m (x_i - \mu_B)^2 \quad (2.4)$$

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \quad (2.5)$$

Eş. 2.5'te ϵ (epsilon) varyans değerinin çok küçük olması durumunda sayısal kararlılığı sağlamaktadır [29]. Daha sonra katman, Eş. 2.6'da gösterildiği gibi, normalize edilmiş girdiyi ölçek faktörü γ ile ölçeklendirmekte ve ofset değeri β ile kaydırmaktadır. Her iki parametre de ağ tarafından öğrenilebilir. Yani ağ eğitiminde her geri yayılımla γ ve β güncellenmektedir.

$$y_i = \gamma \hat{x}_i + \beta \quad (2.6)$$

2.4. ReLu Katmanı

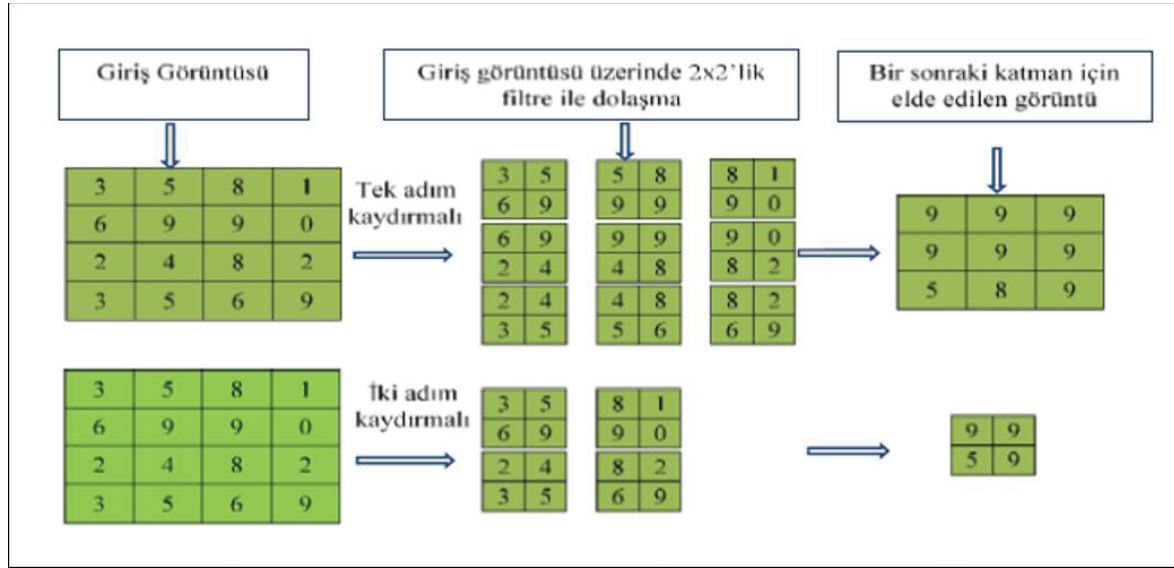
Düzeltilmiş lineer birim (ReLu) katmanı, konvolüsyon ve yığın normalizasyonu katmanlarından sonra kullanılarak, bu katmanların girdiye getirdiği lineerlikten ağı kurtarır. Böylece ağı hızlı öğrenmesine katkı sağlar [30]. Girdinin negatif değerlerini sıfıra çeker ve girdinin boyutuna etki etmez. Matematiksel olarak Eş. 2.7'deki gibi tanımlanır.

$$f(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases} \quad (2.7)$$

2.5. Havuzlama Katmanı

Konvolüsyon katmanında olduğu gibi havuzlama katmanında da bir filtre kullanılır. Filtre girdi üzerinde yatay ve dikey ekseninde belirli bir adım (stride) sayısınca kayarak girdi üzerinde filtrenin bulunduğu konuma karşılık gelen değerlerin maksimum değerini (maksimum havuzlama) veya ortalamasını (ortalama havuzlama) almaktadır. Genellikle ReLu katmanından sonra yerleştirilir. Bu katman sonucunda girdinin derinliği değişmez. Ancak girdinin boyutu azalır (havuzlama katmanının çıktı boyutu Eş. 2.8'de verilmiştir), yani bir bilgi kaybı gerçekleşir. Bu bilgi kaybı sonraki katmanların hesap yükünün azalması

ve sistemin ezberlemeyi önlemesi açısından önemlidir [25]. Ayrıca konvolüsyon katmanında olduğu gibi havuzlama katmanında da kenarlardaki bilgi kaybını azaltmak ya da çıkış boyutunu ayarlamak için girdinin yatay ve dikey kenarlarına dolgu işlemi uygulanabilmektedir. Şekil 2.4 [25]'te 5×5 bir görüntü üzerinde 2×2 bir filtrenin 1 ve 2 adım sayıları ile maksimum havuzlama işlemini uygulaması gösterilmektedir.



Şekil 2.4. 5×5 görüntüde 2×2 filtre ile 1 ve 2 adım kullanılarak maksimum havuzlama [25]

$$\text{Çıkış Boyutu} = (\text{Giriş Boyutu} - \text{Havuzlama Boyutu} + 2 \times \text{Dolgu}) / \text{Adım} + 1 \quad (2.8)$$

2.6. Tam Bağlantılı Katman

Tam bağlantılı katmanda geleneksel bir sinir ağında olduğu gibi, tüm nöronlar kendinden önceki katmandaki tüm nöronlara bağlıdır. Bu katman girdiyi bir ağırlık matrisi ile çarpar ve ardından bir yanlılık (bias) ekler [27]. Girdi X , ağırlıklar W , yanlılık B için çıktı Z Eş. 2.9'da verilmiştir.

$$Z = WX^T + B \quad (2.9)$$

Genel olarak bir sınıflandırma probleminde sınıflandırma katmanından önce ve bir regresyon probleminde regresyon katmanından önce kullanılır. Çünkü bu katman kendinden önceki katmanların öğrendiği tüm özellikleri birleştirmektedir. Bir sınıflandırma

probleminde, tam bağlantılı katmanın çıktı boyutu veri kümesinin sınıf boyutuna eşittir. Bir regresyon probleminde ise bu katmanın çıktı boyutu regresyon katmanından beklenen dönüt (response) adedinceidir. Buna göre; örneğin 10 sınıflı bir sınıflandırma probleminde tam bağlantılı katmanın çıktı boyutu 10×1 olacaktır ve kendinden önceki katmanın çıktısı; yani girdi, $7 \times 7 \times 256 = 1 \times 12544$ boyutunda ise Eş. 2.9'dan ağırlık boyutu 10×12544 olacaktır.

2.7. Çıktı Katmanları

Çıktı katmanı ağı kullanıldığı amaca göre değişiklik gösterir. Ağ bir sınıflandırma problemi için kullanılıyorsa, softmaks katmanını takip eden bir sınıflandırma katmanı kullanılmaktadır. Sınıflandırma problemi, örneğin; bir görüntüdeki rakamın tanınması olabilir. Ağ eğer bir regresyon problemi için kullanılıyorsa, regresyon katmanı kullanılmaktadır. Regresyon problemleri bir değeri tahmin etmeyi içermektedir. Örneğin; bir görüntü üzerindeki evin fiyatının tahmin edilmesi regresyon problemidir. Bir problem kendi içinde hem sınıflandırma hem de regresyon problemlerini içerebilmektedir. Nesne tespiti buna bir örnektir. Nesne tespiti amacıyla kullanılan bir ağ; nesnenin varlığının sınıflandırılması için sınıflandırma katmanı ve bu katmana paralel olarak nesnenin görüntüdeki koordinatlarının tespiti için bir regresyon katmanı içermelidir.

2.7.1. Softmaks ve sınıflandırma katmanları

Bir sınıflandırma probleminde; bir softmaks katmanı ve onun ardından gelen bir sınıflandırma katmanı, bir tam bağlantılı katmanı takip etmektedir [27]. Softmaks katmanı sınıflandırma işlemi için softmaks fonksiyonunu kullanmaktadır. Softmaks fonksiyonu; her bir sınıf için olasılıklarının toplamı 1 olacak şekilde, bir olasılık değeri hesaplamaktadır. Tam bağlantılı katmanın çıktı vektörü a , sınıf sayısı k olmak üzere r sınıfının olasılığı p_r . Eş. 2.10'da verilmiştir.

$$p_r = \frac{e^{a_r}}{\sum_{j=1}^k e^{a_j}} \quad (2.10)$$

Sınıflandırma katmanı; çapraz entropi kaybı fonksiyonu ile kayıp değeri hesaplayarak, softmaks katmanının 0 ile 1 arasında olasılık değerlerinden oluşturduğu sınıflandırma

modelinin performansını ölçer. Bunu, gerçek etiket vektörü ile hesaplanan olasılık vektörünü karşılaştırarak yapmaktadır. Gerçek etiket vektörü 0 ve 1'lerden oluşmaktadır. Örneğin üç sınıflı bir problemde bu vektör; [1,0,0], [0,1,0] veya [0,0,1] olabilmektedir. Eğer görüntü birinci sınıfa ait ise gerçek etiket vektörü [1,0,0]'dır. Gerçek etiket vektörü ile tahmin edilen olasılık arasındaki fark fazla ise kayıp fazla olacaktır [31]. Softmaks katmanının oluşturduğu olasılık vektörü p , gerçek etiket vektörü y , sınıf sayısı n için çok sınıflı bir ($n>2$) problemin çapraz entropi kaybı $H(p,y)$ Eş. 2.11'de verilmiştir. İki sınıflı ($n=2$) bir problem için çapraz entropi kaybı ise Eş. 2.12'de verilmiştir. Buna göre, softmaks katmanının çıktısı olan tahmin vektörü [0,7;0,2;0,1], gerçek etiket vektörü [1,0,0] için kayıp 5,15 olacaktır. Oluşan bu kaybın, ağırlık eğitimi sırasında geri yayılımla her iterasyonda azalması beklenmektedir.

$$H(p, y) = - \sum_{i=1}^n y_i \log(p_i) \quad (2.11)$$

$$H(p, y) = -(y \log(p)) + (1 - y) \log(1 - p) \quad (2.12)$$

2.7.2. Regresyon katmanı

Regresyon katmanı, regresyon problemleri için kayıp değeri hesaplamaktadır [27]. Bir tam bağlantılı katmanın ardından gelmekte ve kayıp değeri için ortalama kare hatası (mean squared error (MSE)) kullanmaktadır. Dönüt sayısı R , hedeflenen (ağa çıktı olarak gösterilen) değer t_i ve ağırlık hesapladığı değer y_i için ortalama kare hatası (MSE) Eş. 2.13'te verilmiştir.

$$MSE = \sum_{i=1}^R \frac{(t_i - y_i)^2}{R} \quad (2.13)$$

3. RESNET

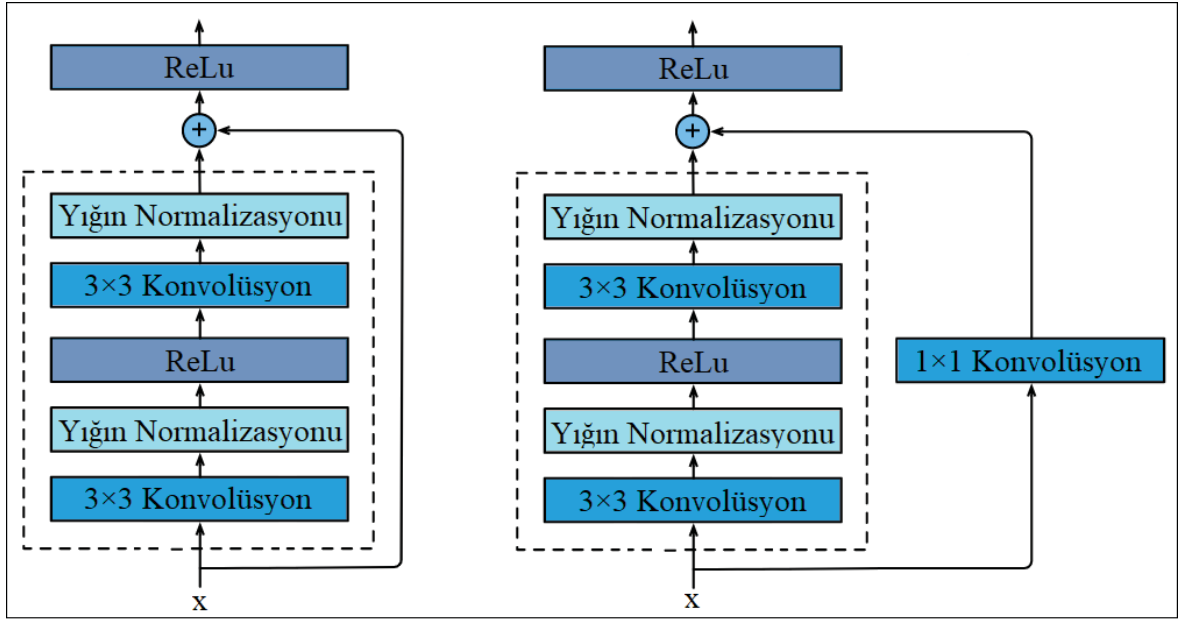
2012 yılında Krizhevsky ve arkadaşlarının AlexNet modeli [8] ile ILSVRC2012 [7] sınıflandırma yarışmasını kazanmasından bu yana CNN mimarisi derinleşme eğiliminde olmuştur. AlexNet yalnızca 5 konvolüsyon katmanına sahipken, VGG [32] ve GoogleNet [33] sırasıyla 19 ve 22 konvolüsyon katmanına sahiptir. Ancak katman sayısı arttıkça; gradyanın, geri yayılımda tekrar eden çarpımlar sonucunda sıfıra yaklaştığı, kaybolan gradyan sorunu ile karşılaşılır. Bu durum, ağı derinleşmesiyle performansın doyuma ulaşmasına ve hatta hızla bozulmaya başlamasına neden olur [18]. Çok derin ağları eğitmedeki bu sorun, ResNet ağı ile tanıtılan artık bloklarla çözülmüştür. Bu bloğun temel işlevi, önceki katmanın çıktısını bir sonraki katmana eklemektir. Buna göre; girdi X 'in ağırlık katmanından geçmesiyle oluşan $F(X)$ çıktısı, bir sonraki ağırlık katmanında önceki katmanın girdisi olan X ile birleşir. Ancak, çoğu kez konvolüsyon işlemine maruz kalan görüntüde olduğu gibi $F(X)$ 'de boyut daralması olur. Bu sebeple X ve $F(X)$ aynı boyuta sahip olmaz. X ; $F(X)$ ile eşleşecek şekilde genişlemesi için doğrusal W ile çarpılır. Bu, X ve $F(X)$ 'in bir sonraki katmana giriş olarak birleştirilmesini sağlar. Eş. 3.1 bu durumu matematiksel olarak ifade etmektedir. Ağı iki matrisin birleşme işlemini gerçekleştiren bu katmanına toplama katmanı (addition layer) ismi verilmiştir.

$$Y = F(X, W_i) + W_s X \quad (3.1)$$

Kaiming He ve arkadaşları, önerdikleri bu modelle kaybolan gradyan sorununu çözmüş, daha önce mümkün olandan çok daha derin ağların eğitilmesine imkan sağlamıştır. Bu avantajla ResNet modeli, ILSVRC2015 [17] sınıflandırma yarışmasında ImageNet veri kümesi ile %3.57'lik hata oranı ile birinci olmuştur. Bu çalışmada, 2015 yılında yayınlanan ResNet'in [18] daha rafine hali olan, ağ büyüklüğüne göre artık bloktaki kısa yol bağlantısının herhangi bir önceki katmana kurulmasına imkan sağlayan güncel versiyonu [34] kullanılmıştır.

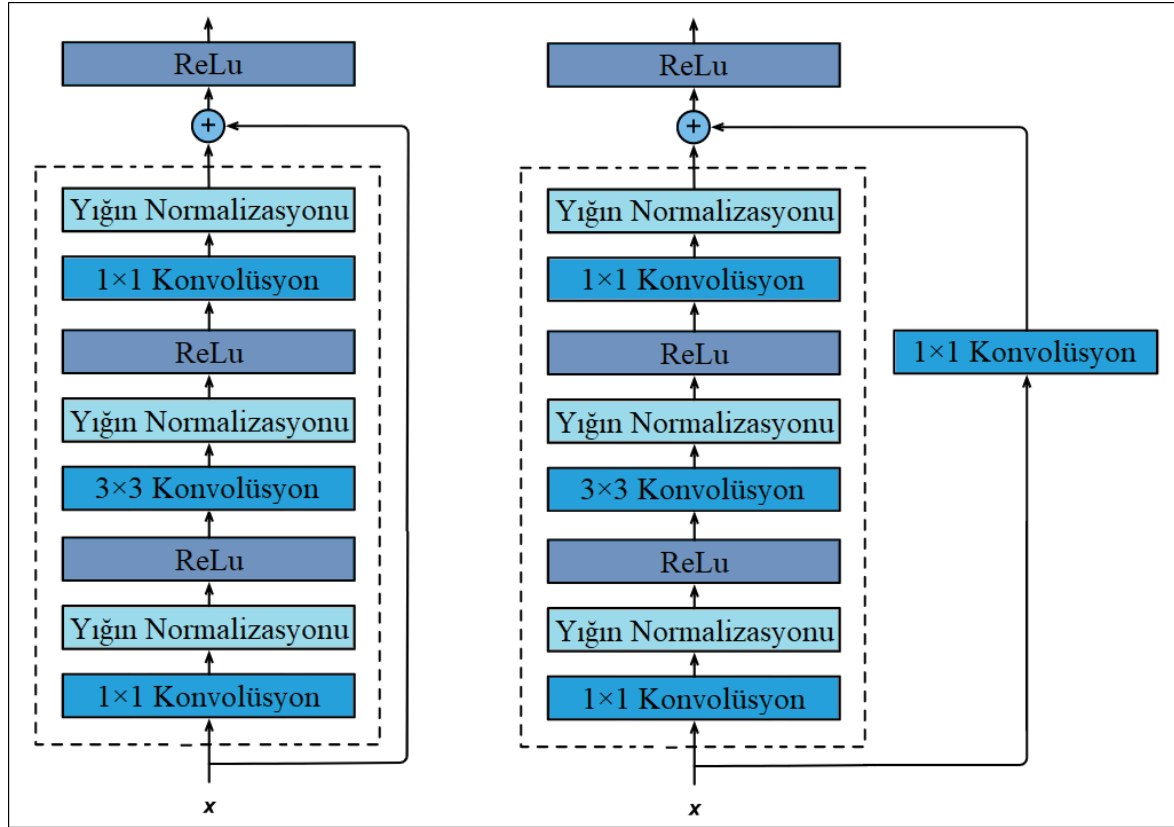
Şekil 3.1 artık blok yapısının yaygın kullanımlarından bir tanesini göstermektedir. 3×3 'lük iki konvolüsyon katmanı içermektedir [35]. Her konvolüsyon katmanını yığın normalizasyonu ve ReLu katmanları takip etmektedir. Her iki konvolüsyon katmanından sonra, ReLu katmanından önce giriş verisi eklenmektedir. Ekleme işlemi konvolüsyon

katmanının çıktısı ile giriş verisinin aynı boyutta olmasını gerektirir. Verinin yükseklik ve genişliğinin korunması bir konvolüsyon parametresi olan adım sayısı ile sağlanırken, derinliğinin korunması her iki konvolüsyon katmanında da derinlik adedince filtre kullanılarak sağlanmaktadır. Giriş verisinin derinliği veya yükseklik $\times$ genişlik boyutu (3×3 'lük iki konvolüsyon katmanında adım sayısının değiştirilmesi yolu ile) değiştirilmek istendiğinde, toplama işleminin gerçekleştirilmesi giriş verisini değişen boyutlara dönüştürmesi için 1×1 'lik bir konvolüsyon katmanı kullanılmaktadır.



Şekil 3.1. ResNet18’de kullanılan artık blok yapıları

Şekil 3.2 artık blok yapısının bir başka yaygın kullanımını göstermektedir. Blok üç konvolüsyon katmanı ve her bir konvolüsyon katmanının ardından gelen yığın normalizasyonu ve ReLu katmanlarını içermektedir. Giriş verisine ilk olarak 1×1 'lik bir konvlüsyon ile derinlik azaltılması uygulanır. Böylece ardından gelen 3×3 'lük konvolüsyon katmanının işlem yükü azaltılmaktadır. Takip eden 1×1 konvolüsyon ile de verinin derinliğinin giriş ile aynı olması sağlanmaktadır. Yine bir önceki yapıda olduğu gibi konvolüyon katmanlarından sonra, ReLu katmanından önce giriş verisi eklenmektedir. Giriş verisinin yükseklik $\times$ genişlik boyutu (3×3 'lük konvolüsyon katmanının adım sayısının değiştirilmesi yolu ile) değiştirilmek istendiğinde, toplama işleminin gerçekleştirilmesi giriş verisini değişen boyutlara dönüştürmesi için 1×1 'lik bir konvolüsyon katmanı kullanılmaktadır.



Şekil 3.2. Resnet50 ve ResNet100’de kullanılan artık blok yapıları

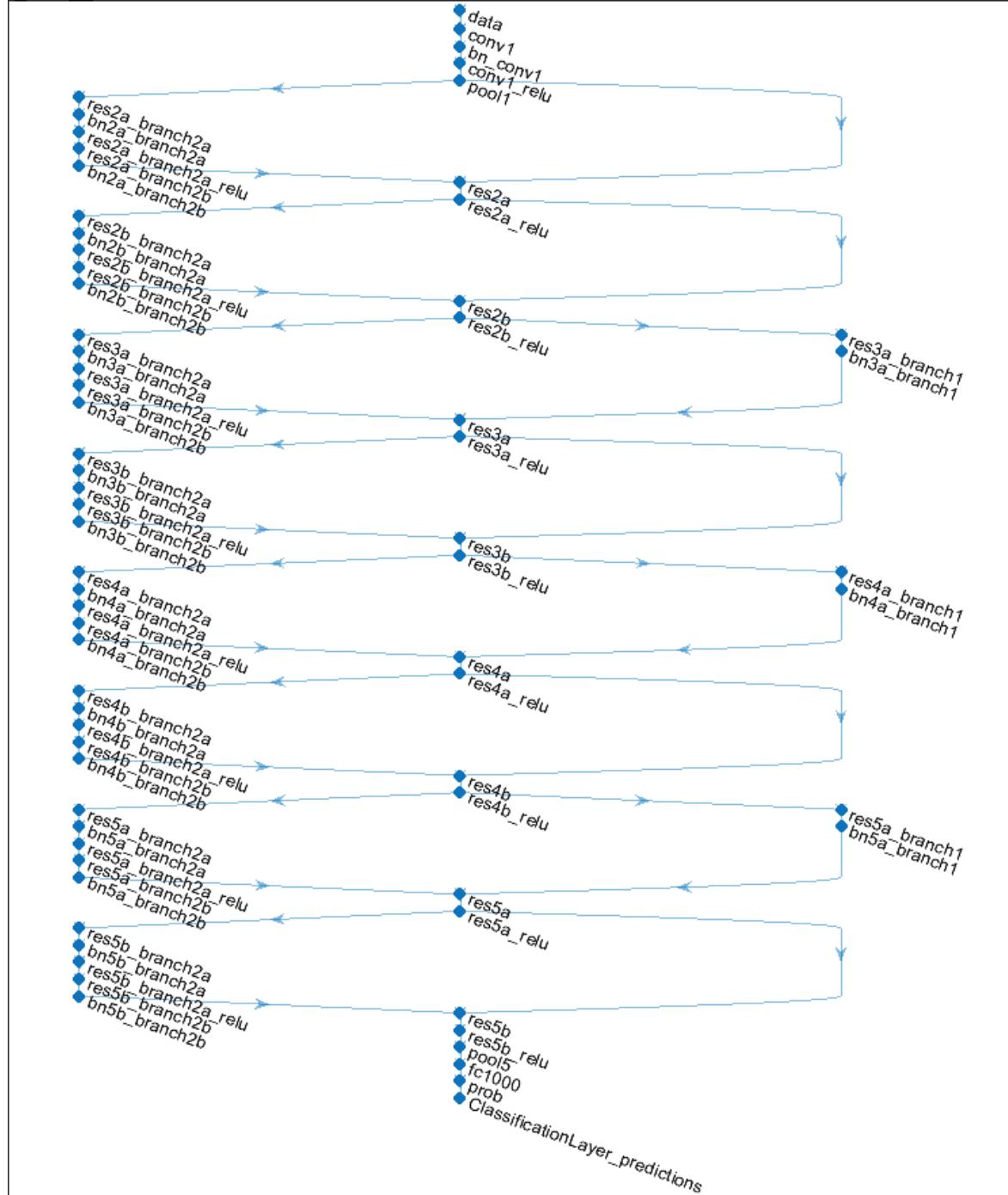
Çizelge 3.1’de gösterildiği gibi ResNet modeli içerdiği katman sayısına göre isim alır. Örneğin; ResNet50, kırk dokuz konvolüsyon katmanı ve bir tam bağlantılı katmandan oluşur. Benzer şekilde; ResNet18, on yedi konvolüsyon katmanı ve bir tam bağlantılı katmandan ve ResNet101, yüz konvolüsyon katmanı ve bir tam bağlantılı katmandan oluşur. Yığın normalizasyonu, ReLu ve maksimum havuzlama gibi diğer katmanlar isimlendirmede etkili olan katman sayısına katılmamaktadır. Her biri esasında beş ana konvolüsyon bloğundan oluşmaktadır. Bu blokların özelliği kendi içerisinde barındırdığı tüm konvolüsyon katmanlarının çıktı boyutunun eşit olmasıdır. Çizelge konvolüsyon bloklarının içerdiği artık blok sayılarını da göstermektedir. Örneğin; conv2_x konvolüsyon bloğunda, ResNet18 iki artık blok içerirken Resnet 50 ve Resnet101 üçer artık blok içermektedir. ResNet modelinin tüm varyasyonları için giriş katmanı 224×224 boyutundadır ve her bir konvolüsyon bloğunun çıktı genişlik ve yüksekliği diğer varyasyonları ile aynı boyuta sahiptir. ResNet ağ mimarisi, ImageNet yarışmasında 1000 adet sınıfa ait nesnenin sınıflandırılması amacıyla tasarlandığı için softmax katmanından önce 1000 çıkışlı tam bağlantılı katmana sahiptir. Doğal olarak; katman sayısı arttıkça, ağın hesap yükü de artmaktadır.

Çizelge 3.1. ResNet ağı katman yapısı

Katman İsmi	Çıkış Boyutu	18 Katman	50 Katman	101Katman
Conv1	112x112	7x7,64, adım 2		
Conv2_x	56x56	3x3 maks. Havuzlama, adım 2		
		$\begin{bmatrix} 3x3,64 \\ 3x3,64 \end{bmatrix} \times 2$	$\begin{bmatrix} 1x1,64 \\ 3x3,64 \\ 1x1,256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1x1,64 \\ 3x3,64 \\ 1x1,256 \end{bmatrix} \times 3$
Conv3_x	28x28	$\begin{bmatrix} 3x3,128 \\ 3x3,128 \end{bmatrix} \times 2$	$\begin{bmatrix} 1x1,128 \\ 3x3,128 \\ 1x1,512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1x1,128 \\ 3x3,128 \\ 1x1,512 \end{bmatrix} \times 4$
Conv4_x	14x14	$\begin{bmatrix} 3x3,256 \\ 3x3,256 \end{bmatrix} \times 2$	$\begin{bmatrix} 1x1,256 \\ 3x3,256 \\ 1x1,1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1x1,256 \\ 3x3,256 \\ 1x1,1024 \end{bmatrix} \times 23$
Conv5_x	7x7	$\begin{bmatrix} 3x3,512 \\ 3x3,512 \end{bmatrix} \times 2$	$\begin{bmatrix} 1x1,512 \\ 3x3,512 \\ 1x1,2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1x1,512 \\ 3x3,512 \\ 1x1,2048 \end{bmatrix} \times 3$
	1x1	Ort.Havuzlama,1000 tam bağlantı,softmax		
İşlem (flop) Sayısı		1.8×10^9	3.8×10^9	7.6×10^9

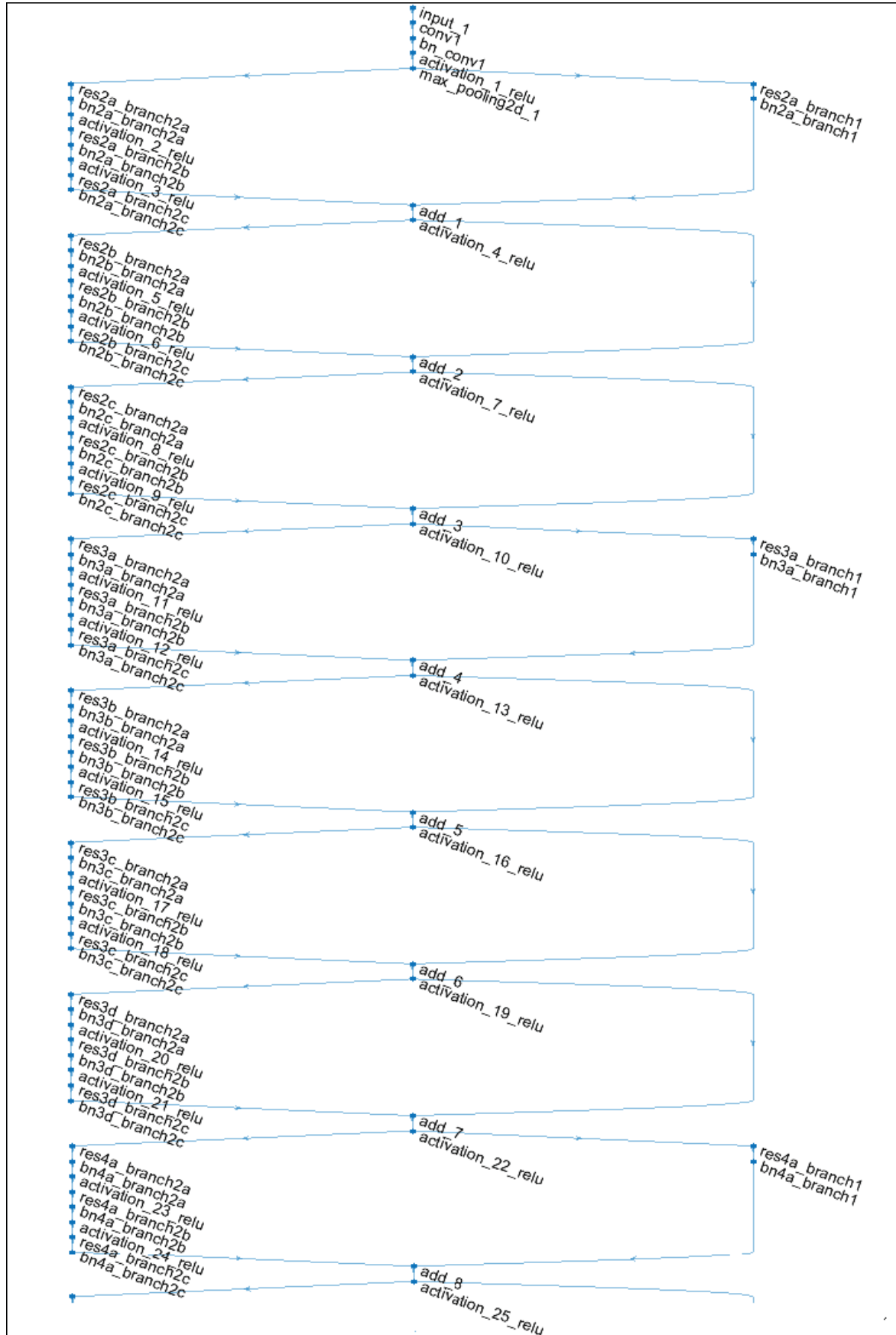
Şekil 3.3 ResNet18 ağına katmanlarını ve bağlantılarını göstermektedir. ResNet18 ağı, $224 \times 224 \times 3$ boyutunda “data” isimli giriş katmanı ile başlar. “conv1” isimli 64 adet [7,7] filtreye, [2,2] adım, [1,1] genişleme, [3,3] dolgu parametrelerine sahip bir konvolüsyon katmanı takip eder. Eş. 2.2’den çizelge 3.1’de gösterilen “Conv1” Konvolüsyon bloğunun çıktısı $112 \times 112 \times 64$ olmaktadır. Veri; “Conv2_x” konvolüsyon bloğunun başında “pool1” ismi ile [3,3] filtreye, [2,2] adım, [1,1] dolgu parametrelerine sahip maksimum havuzlamaya tabi tutulmakta ve Eş. 2.8’den $56 \times 56 \times 64$ boyutuna evirilmektedir. Ardından Şekil 3.1’deki artık blok yapıları kullanılmaktadır. Konvolüsyon blokları kendi içerisinde veri boyutunu korurken, konvolüsyon blokları arasındaki geçişlerden sonra veri boyutu değişikliklerinden dolayı 1×1 konvolüsyon katmanı giriş verisini değişen boyutlara dönüştürmesi için kullanılmıştır. Katman isimlendirilmelerinde; konvolüsyon katmanları için “res#_branch#”, yığın normalizasyonu katmanları için “bn#_branch#”, yığın normalizasyonu katmanından sonra gelen ReLu katmanları için “res#_branch#_relu”, toplama katmanları için “res#” ve toplama katmanlarından sonra gelen ReLu katmanları için “res#_relu” kullanılmıştır. Son konvolüsyon bloğundan veri $7 \times 7 \times 512$ boyutunda çıkmaktadır. Bunu “pool5” ismi ile [7,7] filtreye, [7,7] adım, [0,0] dolgu parametrelerine sahip ortalama havuzlama katmanı takip etmekte ve Eş. 3.8’den çıktı $1 \times 1 \times 512$ boyutuna evirilmektedir. Havuzlama katmanını 1000×512 boyutunda ağırlık matrisine ve 1000×1 boyutunda yanlılık içeren bir tam bağlantılı katman takip etmekte ve Eş. 2.9’dan çıktı 1000×1 boyutunda olmaktadır. Son

olarak “prob” ismi ile softmax ve onu takip eden “ClassificationLayer_predictions” ismi ile sınıflandırma katmanları çıktı katmanını oluşturmaktadır.

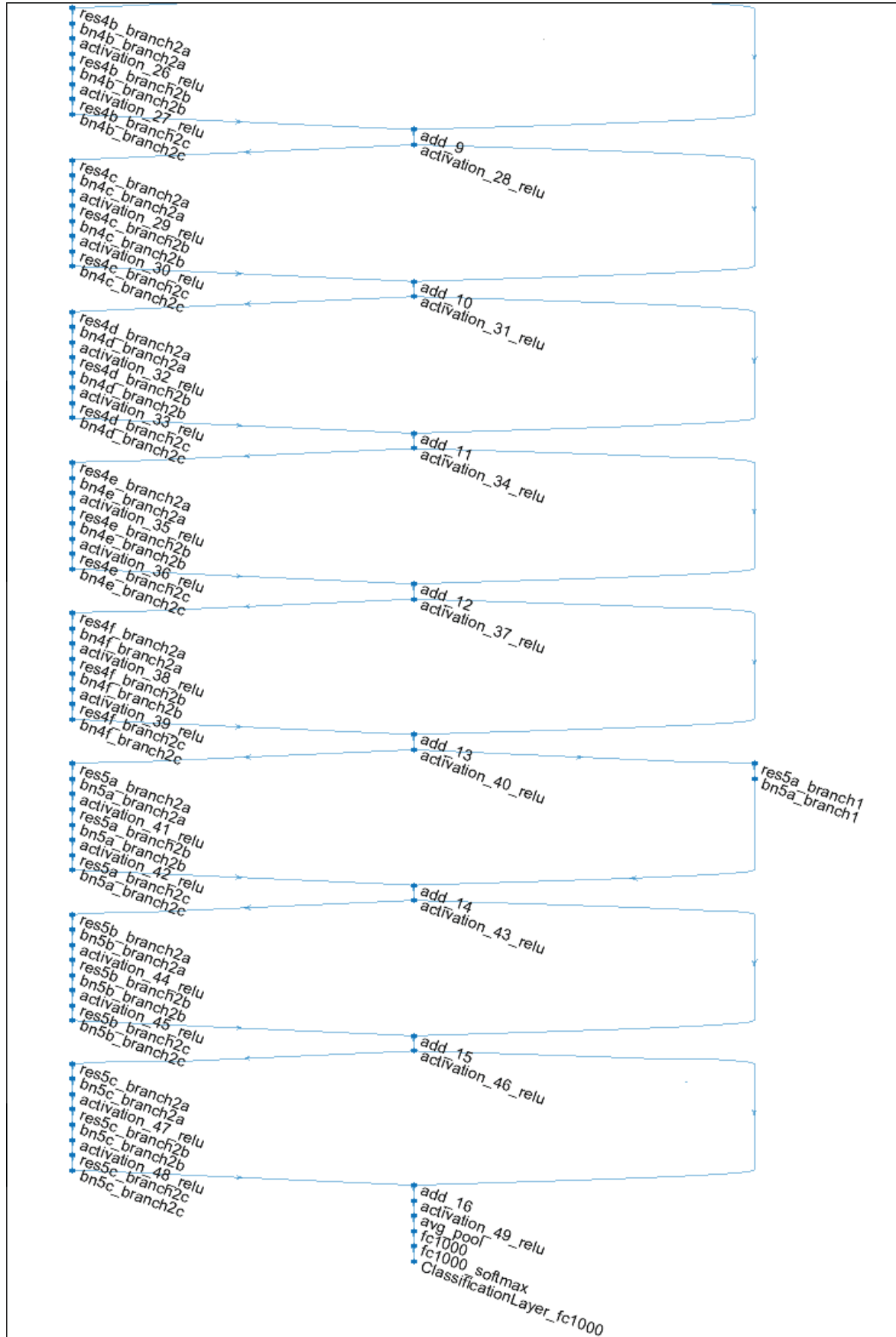


Şekil 3.3. ResNet18 ağ mimarisi

Şekil 3.4 ResNet50 ağının katmanlarını ve bağlantılarını göstermektedir. ResNet18’den farklı olarak Şekil 3.2’de gösterilen artık blok yapısını kullanmaktadır.



Şekil 3.4. ResNet50 Ağ mimarisi



Şekil 3.4. (devam) ResNet50 Ağ mimarisi

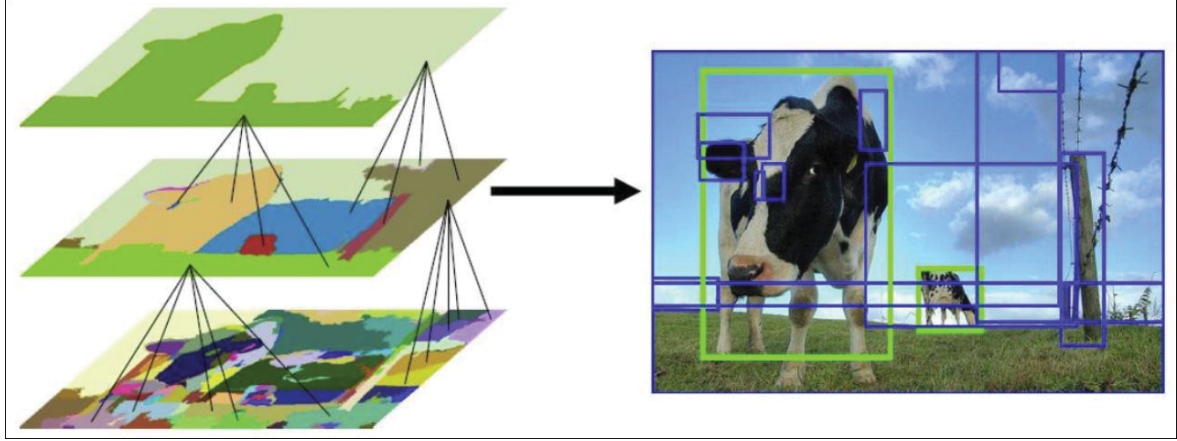
ResNet101 ađının katmanları ve bu katmanların bađlantıları EK-1’de verilmiřtir. ResNet50 ile benzer řekilde řekil 3.2’de gsterilen artık blok yapısını kullanmaktadır.

4. DAHA HIZLI RCNN ALGORİTMASININ GELİŞİMİ VE UYGULANMASI

Bilgisayarlı görü (computer vision), CNN'lerle birlikte büyük ivme kazanan disiplinler arası bir alandır. Temel olarak insanın görsel olarak yapabileceği görev ve işlevleri bilgisayar ortamına taşıyarak, bu görev ve işlevlerin otomatikleştirilmesini amaçlar. Nesne tespiti ise bilgisayarlı görünün önemli bir parçasıdır. Nesne tespit algoritmalarında genel olarak, görüntü içerisinde ilgilenilen nesnenin konumunun tespit edilip sınırlayıcı kutular ile etrafının çizilmesi hedeflenir. Hedef nesne görüntü içerisinde çok sayıda bulunabilir. Bu durumda çok sayıda sınırlayıcı kutuya ihtiyaç duyulur ve bunun kaç tane olacağı önceden bilinmez. Bu noktada standart CNN'lerin yetersizliği ortaya çıkmaktadır. Çünkü CNN'ler konumdan bağımsız olarak çalışırlar. CNN'lerin çözüm getirdiği problemlere bakıldığında, bu problemler konumsal olarak bağımlı özelliklere sahip değildirler [22]. Bu sonun çözümü için; görüntüden farklı ilgi alanlarını alma ve o bölgedeki nesnenin varlığını sınıflandırma için bir CNN kullanma yaklaşımı benimsenmiştir. Bu yaklaşımda, nesneler görüntü üzerinde farklı konumlarda ve boyutlarda olabileceği için çok sayıda bölge seçimine ihtiyaç duyulur. Önerilen bölge sayısının artması da hesaplama maliyetini arttıracığından zamandan kayıp yaşatır. RCNN ile başlayan daha hızlı RCNN'in gelişim süreci, bu parametrelerin iyileştirilmesi üzerine kurulmuştur.

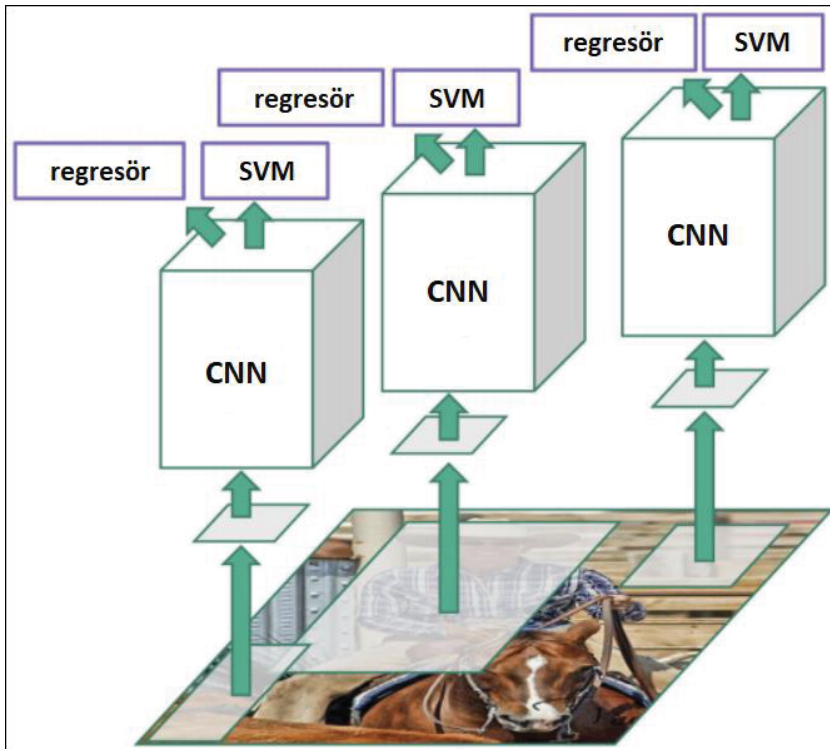
4.1. RCNN Algoritması

Ross Girshick ve arkadaşları, çok sayıda bölge önerilmesi yerine, seçici arama algoritması [36] kullanarak yalnızca 2000 bölge önerisi sunan bir yöntem önerdi [9]. Seçici arama algoritması, görüntü segmentasyonu ile görüntüdeki benzer pikselleri bölge oluşturmak üzere gruplandırmaya dayanmaktadır. Ancak farklı olarak, görüntüde tek bir segmentasyon yöntemini çalıştırmak yerine, çok sayıda durumu kapsayacak şekilde hiyerarşik grupta tabanlı segmentasyon algoritması kullanılmaktadır. Buna göre; ilk olarak Şekil 4.1'de sol altta görüldüğü gibi; görüntü, nesne tabanlı olmayan segmentasyon ile renk, doku, boyut ve şekilsel benzerliklerle küçük alanlara bölünür. Benzer bölgeler daha sonra yeni bölgeler oluşturmak için birleştirilir, bu işlem tüm görüntü tek bir bölgede temsil edilene kadar sürer. Her adımdaki bölgeler, bölge tekliflerine eklenir. Şekil 4.1 [36], seçici arama algoritmasının işleyişini göstermektedir.



Şekil 4.1. Seçici arama algoritmasının işleyişi [36]

2000 adet bölge önerisinin üretiminin ardından, önerilen her bir bölge sabit 227×227 boyutlarında kareye dönüştürülür. Kare formatındaki bölge önerileri, özelliklerin çıkarılması için ayrı ayrı CNN'ye beslenir. CNN, çıktı olarak 4096 boyutunda bir özellik vektörü oluşturur. Özellik vektörü, o bölgedeki nesnenin varlığını tahmin etmesi, sınıflandırma işlemini gerçekleştirmesi için SVM'ye [4] beslenir. Sınırlayıcı kutuların oluşturulması için ise regresör kullanılmaktadır. Şekil 4.2 [9] RCNN algoritmasının çalışma prensibini göstermektedir.

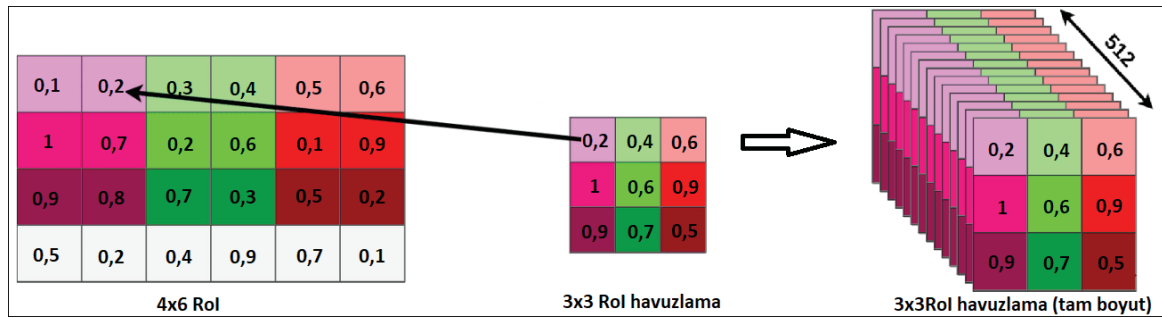


Şekil 4.2. RCNN algoritması çalışma prensibi [9]

RCNN algoritmasının dezavantajı, her bir görüntüde 2000 adet sınıflandırma işlemine ihtiyaç duymasındır. Bu hem ağı eğitiminin hem de eğitilmiş ağı test görüntüleri üzerinde uygulamasının zaman almasına neden olmaktadır.

4.2. Hızlı RCNN Algoritması

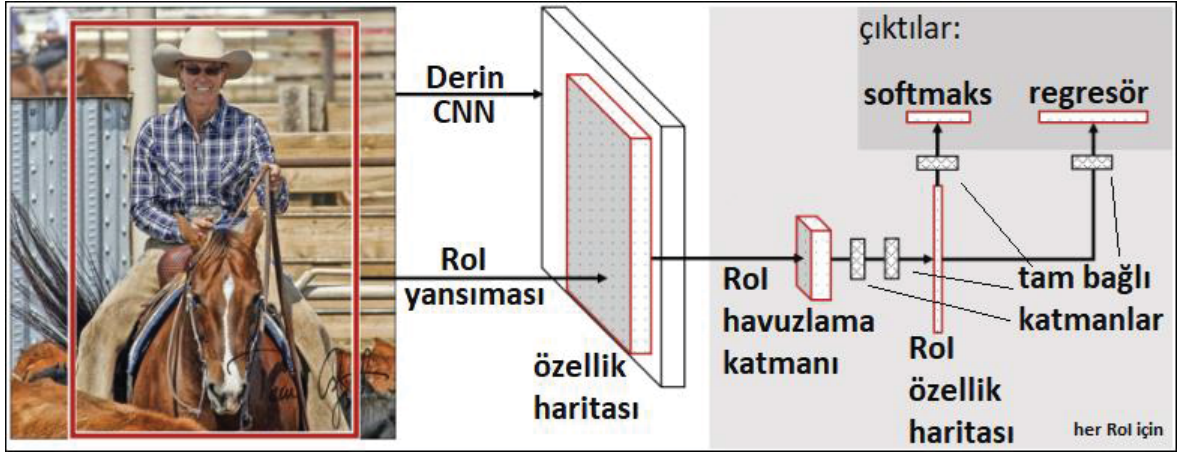
RCNN algoritmasının aynı yazarı, RCNN'nin dezavantajlı yönlerini çözerek daha hızlı bir algoritma oluşturmuş ve bu algoritmaya hızlı RCNN [10] ismini vermiştir. Bu algoritmaya göre, görüntü ilk olarak bir CNN'den geçerek özellik haritası oluşturmaktadır. Bölge önerileri, RCNN'de kullanılan seçici arama algoritması ile özellik haritası üzerinden çıkarılmaktadır. Özellik haritası üzerinden oluşturulan her bir bölge önerisine ilgi alanı (region of interest) (RoI) denmektedir. İlgi alanları farklı en boy oranlarında olabilmektedir. Bunların bir tam bağlı katmana beslenebilmesi için aynı boyuta sahip karelere dönüştürülmesi gerekmektedir. Bunun için RoI havuzlama katmanı kullanılmaktadır. RoI havuzlaması sonucu elde edilen matris iki ayrı tam bağlı katmandan geçirilerek RoI özellik vektörü oluşturulur. Ardından sınıflandırma işlemi için softmax ve sınırlandırıcı kutuların oluşturulması için regresör kullanılır. Bu işlem özellik haritasından çıkarılan her bir RoI için uygulanır. Şekil 4.4 [10] hızlı RCNN algoritmasının çalışma prensibini göstermektedir



Şekil 4.3. RoI havuzlama katmanı

RoI havuzlama katmanı, maksimum havuzlama katmanı ile aynı mantıkla çalışmaktadır (Şekil 4.3). Örneğin; bir CNN mimarisi olan VGG-16 [32] ağı 512×512×3 boyundaki bir görüntüyü girdi olarak aldığı anda çıktı olarak bunu 16×16×512 boyutunda bir özellik haritasına eşler. Orijinal görüntü üzerinde 145×200 boyutunda yer kaplayan bir nesne 32'ye 1 oranından dolayı özellik haritası üzerinde 4×6 boyutunda bir RoI'ya dönüşür (Bölge önerileri bir nesneyi temsil etmek zorunda değildir, anlaşılması için bu şekilde bir örnek

verilmiştir). Hedeflenen sabit kare boyutunun 3×3 olduğu varsayılırsa, $4/3=1$ ve $6/3=2$ 'den dolayı 1×2 'lik bir filtre ile uygulanacak maksimum havuzlama yeterli olacaktır (RoI'nin alt sırası kaybedilir). Bu işlem yalnızca en üst katmanda değil tüm RoI'da uygulandığında $3 \times 3 \times 512$ boyutunda bir matris elde edilir.



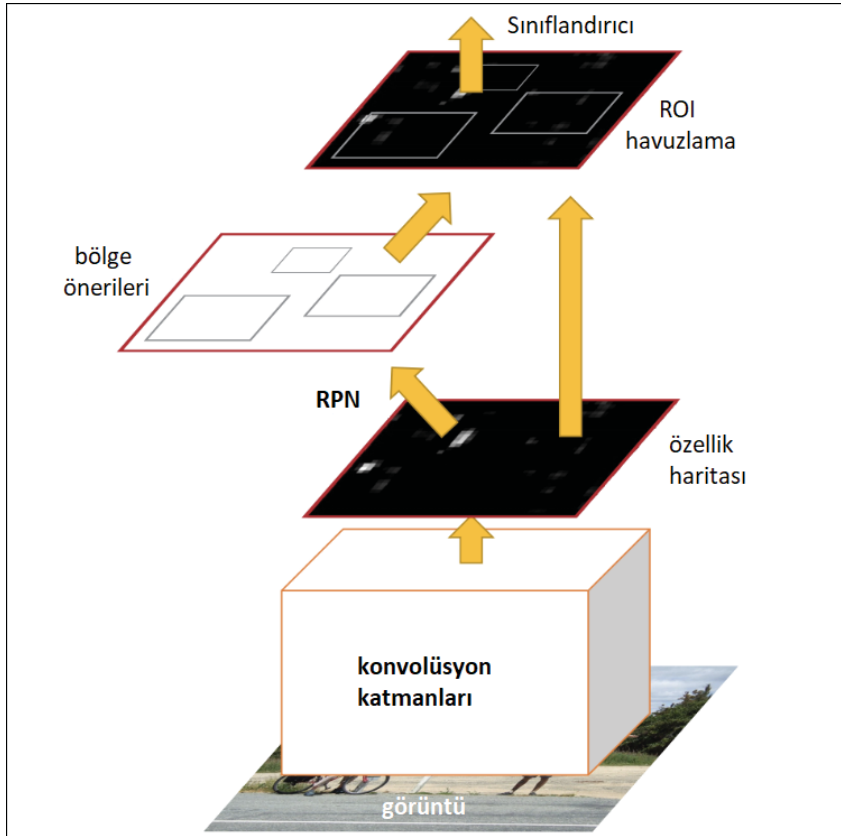
Şekil 4.4. Hızlı RCNN algoritması çalışma prensibi [10]

RCNN algoritmasında, bir görüntüde oluşturulan her bölge önerisi için bir CNN çalışması gerekmektedir. Bu da bir görüntü için 2000 adet CNN'nin çalışması anlamına gelir. Hızlı RCNN algoritmasında ise her görüntüde yalnızca bir CNN'nin çalışması yeterlidir. Bu durum; hızlı RCNN algoritmasını, RCNN algoritmasından daha hızlı yapar.

4.3. Daha Hızlı RCNN Algoritması

Daha hızlı RCNN [11]; RCNN ve hızlı RCNN algoritmalarının dezavantajlı yönlerini çözerek üretilmiş bir algoritmadır. RCNN ve hızlı RCNN'de önce seçici arama algoritması ile bölge önerileri oluşturulmakta, daha sonra bölge önerileri üzerinde nesne sınıfını sınıflandırmak ve sınırlayıcı kutuları oluşturmak için CNN tabanlı bir ağ kullanılmaktadır. Aralarındaki temel fark ise; RCNN bölge önerilerini piksel düzeyinde doğrudan girdi olarak aldığı görüntüden üretirken, hızlı RCNN bölge önerilerini girdi olarak aldığı özellik haritasından üretmektedir. Ancak her ikisinde de bölge önerisinin yapıldığı algoritma ile tespit ağı birbirinden ayrıştırılmıştır. Bu yaklaşım Shaoqing Ren vd.'ne göre iyi bir yaklaşım değildir. Bu yaklaşımda seçici arama yanlış negatif ürettiğinde, tespit ağı bundan zarar görmektedir. Bu sebeple, bu iki yapıyı birbiri ile ilişkilendirerek birleştirmek daha iyi bir fikirdir. Ayrıca RCNN ve Hızlı RCNN'de kullanılan seçici arama algoritması basit bir

algoritmadır. Bölge önerisinin oluşturulduğu aşamada, hiçbir öğrenme gerçekleşmez. Bu, kötü aday bölgelerinin oluşmasına yol açabilmektedir. Aynı zamanda bu algoritma yavaş çalışmaktadır. Tüm bunları göz önünde bulunduran Shaoqing Ren ve arkadaşları “daha hızlı RCNN” ismini verdikleri algoritmalarında, seçici arama yerine ağıın bölge önerilerini öğrenmesini sağlayan bölge önerileri ağı (RPN)’nı önermişlerdir. Bu CNN tabanlı ağ, tespit ağı ile paylaşımlıdır. Hızlı RCNN’de olduğu gibi Daha hızlı RCNN’de de görüntü doğrudan özellik haritasının oluşturulması için evrimsel bir ağıın girişı olarak ağı beslenir. Bölge önerilerinin özellik haritası üzerinden üretilmesi için ayrı bir ağ kullanılır. Son olarak öngörülen bölge önerileri, ilgili bölgedeki görüntüyü sınıflandırmak ve sınırlama kutularının oluşturulması için ROI havuzlama katmanı kullanılarak yeniden şekillendirilir. Şekil 4.5 [11] daha hızlı RCNN algoritmasının çalışma prensibini göstermektedir.

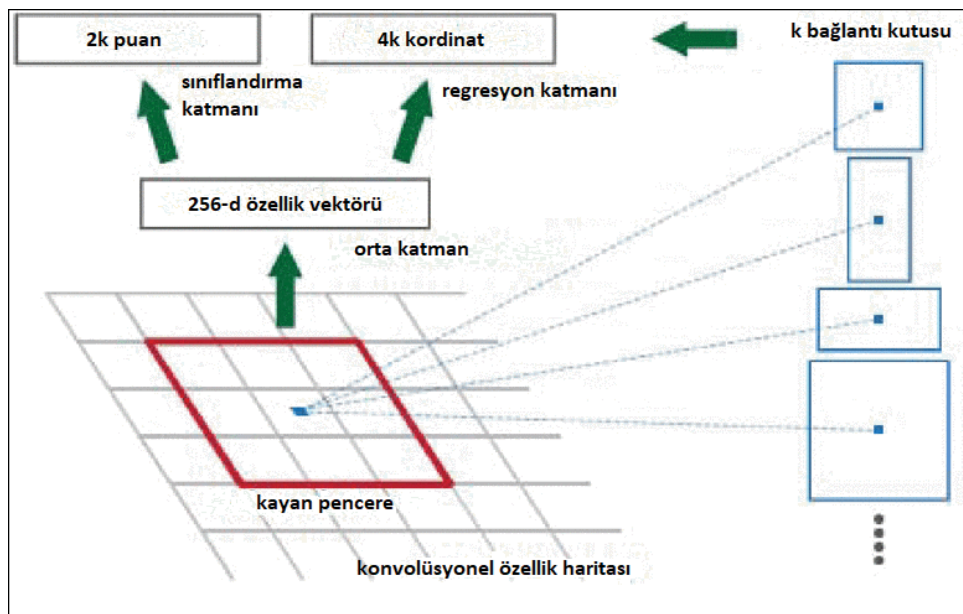


Şekil 4.5. Hızlı RCNN algoritmasının çalışma prensibi [11]

4.3.1. Bölge önerileri ağı

Buna göre, daha hızlı RCNN’de görüntü, ilk olarak konvolüsyon katmanlarından geçmekte ve özellik haritası oluşturmaktadır. Özellik haritasının oluşturulmasının ardından, bölge

öneri ağında özellik haritası üzerindeki her konum için kayan bir pencere kullanılmaktadır. Özellik haritası üzerindeki her bir piksele çapa ismi verilmektedir. Her çapa orijinal görüntü üzerinde daha büyük kare boyutundaki piksellere denk gelmektedir. Her bir çapada bölge önerileri oluşturmak için $k=9$ adet çapa kutusu kullanılmaktadır. Bu kutular standart bir daha hızlı RCNN’de 128, 256 ve 512 ölçeklerinde ve 1:1, 1:2, 2:1 en-boy oranlarındadır. Çapa kutularının bu boyutları orijinal görüntü boyutuna endeksli olarak belirtilir. Yani; ağ girişinde $512 \times 512 \times 3$ boyutunda, özellik haritası oluştuğunda $16 \times 16 \times 512$ boyutunda olan bir görüntü için; önceden belirlenen çapa kutularından bir tanesi 256×256 ise, bu çapa kutusu özellik haritasında 8×8 ’e (32 ’ye 1 oranından dolayı) denk gelecektir. Çapa kutuları sabit boyutlu kutulardır ve bölge teklif ağıının amacı bu kutulara karşılık gelen görüntüde nesnenin varlığının (nesne var veya yok şeklinde) sınıflandırılması ve nesne sınırlayıcı kutularının koordinatlarının oluşturulmasıdır. Bu amaçla, bir sınıflandırma katmanı özellik haritası üzerindeki her bir noktada k kutuları içinde nesne olup olmadığına dair $2 \times k$ adet puan üretmektedir. Bir regresyon katmanı da yine her nokta için k bağlantı kutuları içerisinde merkez koordinatlarını, genişlik ve yüksekliği içeren $4 \times k$ adet koordinat önerisi üretmektedir. Bir $W \times H$ özellik haritasında toplam $W \times H \times k$ adet çapa kutusu vardır. Böylece bölge öneri ağı hangi konumun nesne içerdiğini önceden kontrol etmektedir. Ardından karşılık gelen konumlar ve sınırlama kutuları, nesne sınıfını algılamak ve o nesnenin sınırlama kutusunu oluşturmak için tespit ağına geçecektir. Şekil 4.6 [11] bölge önerisi ağıının çalışma prensibini göstermektedir.



Şekil 4.6. Bölge Önerileri Ağı çalışma prensibi [11]

4.3.2. Tespit Ağı

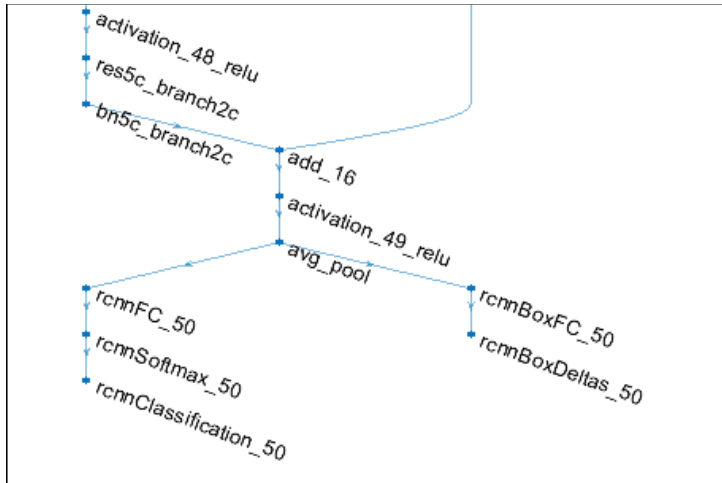
Bölge teklif ağı dışında kalan kısım, hızlı RCNN ile benzerdir. Önce ROI havuzlama yapılmaktadır. Ardından havuzlanmış alan CNN'ye girmekte ve iki farklı tam bağlantılı katman dalına ayrılmaktadır. Birinci tam bağlantılı katmanı sınıflandırma için softmax, diğer tam bağlantılı katmanı sınırlayıcı kutuların oluşturulması için regresyonu izlemekte ve ağ sonlanmaktadır.

4.4. Daha Hızlı RCNN Algoritmasının Uygulanması

Shaoqing Ren ve arkadaşları çalışmalarında daha hızlı RCNN algoritmasını uygulamak üzere VGG-16 [32] ve Zeiler ve Fergus (ZF) [37] ağlarını kullanmışlardır. Bu çalışmada ise, daha hızlı RCNN algoritmasını uygulamak üzere Bölüm 3'te anlatılan ResNet ağı kullanılmıştır. Deneysel çalışmalara ResNet50 ağı ile başlanılmıştır. Bu sebeple daha hızlı RCNN algoritmasının bir ağı uygulanması ResNet50 ağı üzerinden anlatılmıştır.

Bir nesne tespit algoritması olan daha hızlı RCNN'nin ResNet50 ağına uygulanması iki aşamadan oluşmaktadır. Birinci aşama; ağıın çıktı katmanının ağıın kullanım amacına uygun hale getirilmesini yani tespit ağıının oluşturulmasını içermektedir. ResNet ağı, ImageNet yarışmasında 1000 adet sınıfa ait nesnenin sınıflandırılması amacıyla kurgulanmış bir mimaridir. Dolayısıyla Şekil 3.4'te gösterildiği gibi "avg_pool" isimli ortalama havuzlama katmanından sonra 1000 adet nesne sınıfını sınıflandırmak üzere 1000 çıktıya sahip tam bağlantılı katmanı takip eden softmax ve sınıflandırma katmanları bulunmaktadır. Ayrıca ImageNet yarışmasının amacı yalnızca sınıflandırma olduğu, nesne etrafına bir sınırlayıcı kutu çizdirmenin ihtiyacı olmadığı için de ResNet ağı bir regresyon katmanına sahip değildir. Bu çalışmadaki amaç ilk olarak araçların tespitini içermektedir (bir sonraki amaç olan araçların araba, pikap, uçak vs. olarak tespit ve sınıflandırılması çalışması bölüm 8'de anlatılmıştır). Dolayısıyla bu çalışmanın ilk basamağı arka plan ile birlikte 2 sınıflı bir sınıflandırma problemini ve bir nesne tespiti çalışmasının gereği olarak nesnenin etrafına çizilmesini gerektiren sınırlayıcı kutu x (sınırlayıcı kutu sol üst köşe x koordinatı), y (sınırlayıcı kutu sol üst köşe y koordinatı), w (sınırlayıcı kutusunun piksel cinsinden genişliği) ve h (sınırlayıcı kutusunun piksel cinsinden yüksekliği) değerlerinin tahmini için bir regresyon problemini içinde barındırmaktadır. Bu sebeplerle orijinal ResNet50 ağıının "fc1000" isimli tam bağlantılı, "fc1000_softmax" isimli softmax ve

“ClassificationLayer_fc1000” isimli sınıflandırma katmanları ağdan atılmıştır. Ardından iki sınıflı sınıflandırma problemine uygun olarak 2 çıktıya sahip “rcnnFC_50” isimli tam bağlantılı, “rcnnSoftmax_50” isimli softmaks ve “rcnnClassification_50” isimli sınıflandırma katmanları tanımlanmış ve “avg_pool” isimli ortalama havuzlama katmanının çıktısı olarak ağa bağlanmıştır. Regresyon katmanından önce gelen tam bağlantılı katmanın çıktı sayısı sınırlayıcı kutunun dört koordinatından dolayı $4 \times$ sınıf sayısı olması gerekmektedir. Arka plan, gerçek bir nesne sınıfı olmadığı için sınıf sayısı 1 alınarak 4 çıktıya sahip “rcnnBoxFC_50” isimli tam bağlantılı ve “rcnnBoxDeltas_50” isimli regresyon katmanları tanımlanmış ve ortalama havuzlama katmanının çıktısı olarak ağa bağlanmıştır. ResNet50 ağının çıktı katmanlarında yapılan bu değişiklikler Şekil 4.7’de gösterilmiştir.

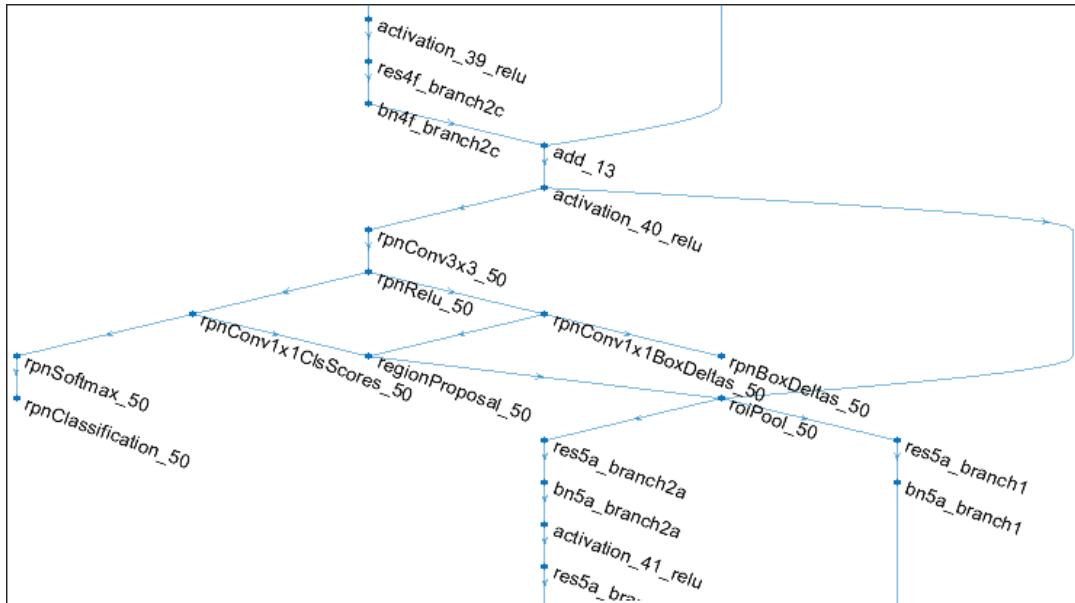


Şekil 4.7. Nesne tespitine uygun ResNet50 ağının çıktı katmanları

Daha hızlı RCNN’nin ResNet50 ağına uygulanmasında ikinci aşama, bölge önerileri ağının (RPN) eklenmesidir. RPN bölge önerilerini özellik haritası üzerinden oluşturmaktadır. Bu sebeple bu aşamada çıktısının özellik haritası olarak kullanılacağı bir katmanının belirlenmesi gerekmektedir. Özellik çıkarma katmanı, ResNet ağı için de farklı ağlar için de genel bir yaklaşım olarak 4. konvolüsyon bloğunun son katmanı olarak belirlenmiştir [38 ve 39]. ResNet50 ağında bu, “activation_40_relu” isimli ReLu katmanına karşılık gelmektedir. “activation_40_relu” katmanının çıktı boyutu, Çizelge 3.1’den çıkarılacağı gibi $14 \times 14 \times 1024$ tür. RPN ağının eklenmesi için özellik çıkarma katmanına bağlı “res5a_branch2a” ve “res5a_branch1” isimli katmanların bağlantısı kesilmiş, ardından özellik çıkarma katmanına 3×3 boyutunda filtreye, 1024 adet filtre sayısına ve [1,1] dolguya sahip ‘rpnConv3x3_50’

isimli bir konvolüsyon ve onu takip eden “rpnRelu_50” isimli bir ReLu katmanları tanımlanmış ve eklenmiştir. Özellik çıkarma katmanını takip eden bu konvolüsyon ve ReLu katmanları RPN ağını oluşturmaktadır. Konvolüsyon katmanındaki 1024 filtre adedi özellik haritasının derinliğini adedince seçilmektedir [38]. Ayrıca Eş. 2.2’den RPN ağının çıktısı, özellik haritasının boyutu korunarak $14 \times 14 \times 1024$ olmuştur. RPN ağı özellik haritasındaki her bir noktada çapa kutusu adedince “nesne” veya “arka plan” şeklinde sınıflandırma ve yine özellik haritasındaki her bir noktada çapa kutusu adedince 4 değerden oluşan sınırlayıcı kutu değerlerini üretmektedir. Sınıflandırma; RPN ağına bağlanmış olan 1×1 boyutunda filtreye ve ‘ $2 \times$ Çapa Sayısı’ adet filtre sayısına sahip “rpnConv1x1ClsScores_50” isimli bir konvolüsyon katmanı, konvolüsyon katmanını takip eden “rpnSoftmax_50” isimli bir softmaks katmanı ve softmaks katmanına bağlı “rpnClassification_50” isimli bir sınıflandırma katmanı ile sağlanmıştır. Sınırlayıcı kutu koordinatları ise; RPN ağına bağlanmış olan 1×1 boyutunda filtreye ve ‘ $4 \times$ Çapa Sayısı’ adet filtre sayısına sahip “rpnConv1x1BoxDeltas_50” isimli konvolüsyon katmanı ve konvolüsyon katmanını takip eden “rpnBoxDeltas_50” isimli regresyon katmanı ile sağlanmıştır. Farklı boyutlarda ağa önceden tanımlanmış 9 adet çapa kutusu için, her bir özellik haritası noktasında 9 adetten toplam $14 \times 14 \times 9$ adet çapa kutusu vardır. Ağa bir girdi olarak verilen çapa kutu sayısı RPN ağında sınıflandırma ve regresyondan önceki konvolüsyon katmanlarının derinliğini belirlemektedir. Her bir çapa kutusu için RPN iki sınıflı (nesne ve arka plan) bir sınıflandırma ve sınırlayıcı kutuların 4 koordinatı için dört regresyon problemini çözümlendiğinden dolayı; “rpnConv1x1ClsScores_50” katmanı $14 \times 14 \times 18$, “rpnConv1x1BoxDeltas_50” katmanı $14 \times 14 \times 36$ çıktı boyutlarına sahiptir. Yani “rpnConv1x1ClsScores_50” katmanının $14 \times 14 \times 18$ boyutlu çıktısının genişlik-yükseklik eksenindeki bir noktadaki 18 değer, özellik haritasının bir noktasındaki 9 çapa kutusunun her biri için nesne ve arka plan skor bilgilerine karşılık gelmektedir. “rpnConv1x1BoxDeltas_50” katmanının $14 \times 14 \times 36$ boyutlu çıktısının genişlik-yükseklik eksenindeki bir noktadaki 36 değer de, özellik haritasının bir noktasındaki 9 çapa kutusunun her biri için sınırlayıcı kutu koordinatlarına (sol üst köşe x koordinatı, sol üst köşe y koordinatı, genişlik, yükseklik) karşılık gelmektedir. Burada sınıflandırma ve regresyon katmanları ağın eğitimi için gerekli olan kayıp değerlerini oluşturmaktadırlar. Skor bilgisi olarak “rpnConv1x1ClsScores_50” katmanının çıktısını ve sınırlayıcı kutu koordinatı bilgisi olarak “rpnConv1x1BoxDeltas_50” katmanının çıktısını girdi olarak alan, bu bilgileri her bir çapa kutusuna ait özellik olarak tutan “regionProposal_50” isimli bölge öneri katmanı tanımlanmış ve ağa bağlanmıştır. Bölge öneri katmanının çıktıları çapa kutularının

kendisidir ve bu çapa kutularının boyutları bölge öneri katmanının bir girdisi olarak ağa önceden tanımlanmaktadır. Bölge öneri katmanının ardından “roiPool_50” isimli [14,14] çıktı boyutuna sahip (özellik haritası ile aynı boyuta sahiptir) RoI havuzlama katmanı tanımlanmış, “regionProposal_50” bölge önerileri katmanı ve “activation_40_relu” özellik çıkarma katmanları ile bağlanmıştır. RoI havuzlama katmanı, özellik haritasını ve bölge önerileri ile oluşun ilgi alanlarına ait bilgileri girdi olarak almaktadır. Özellik haritasında bu bilgilere karşılık gelen bölümleri almakta ve maksimum havuzlama ile Bölüm 4.2’de anlatıldığı gibi sabit boyuta ([14,14]) dönüştürmektedir. Son olarak; RoI havuzlama katmanının çıkışı, özellik çıkarma katmanı ile bağlantısı koparılmış olan, “res5a_branch2a” ve “res5a_branch1” katmanlarına bağlanmıştır. Şekil 4.8 ResNet50 ağında daha hızlı RCNN algoritmasının ikinci aşamasının uygulanmış halini göstermektedir. Şekil 4.7 ve 4.8’den görüleceği gibi daha hızlı RCNN’nin eğitimi; ikisi RPN ağıının, diğer ikisi tespit ağıın sınıflandırma ve regresyon katmanlarında hesaplanan 4 kayıp değeri ile gerçekleşmektedir.

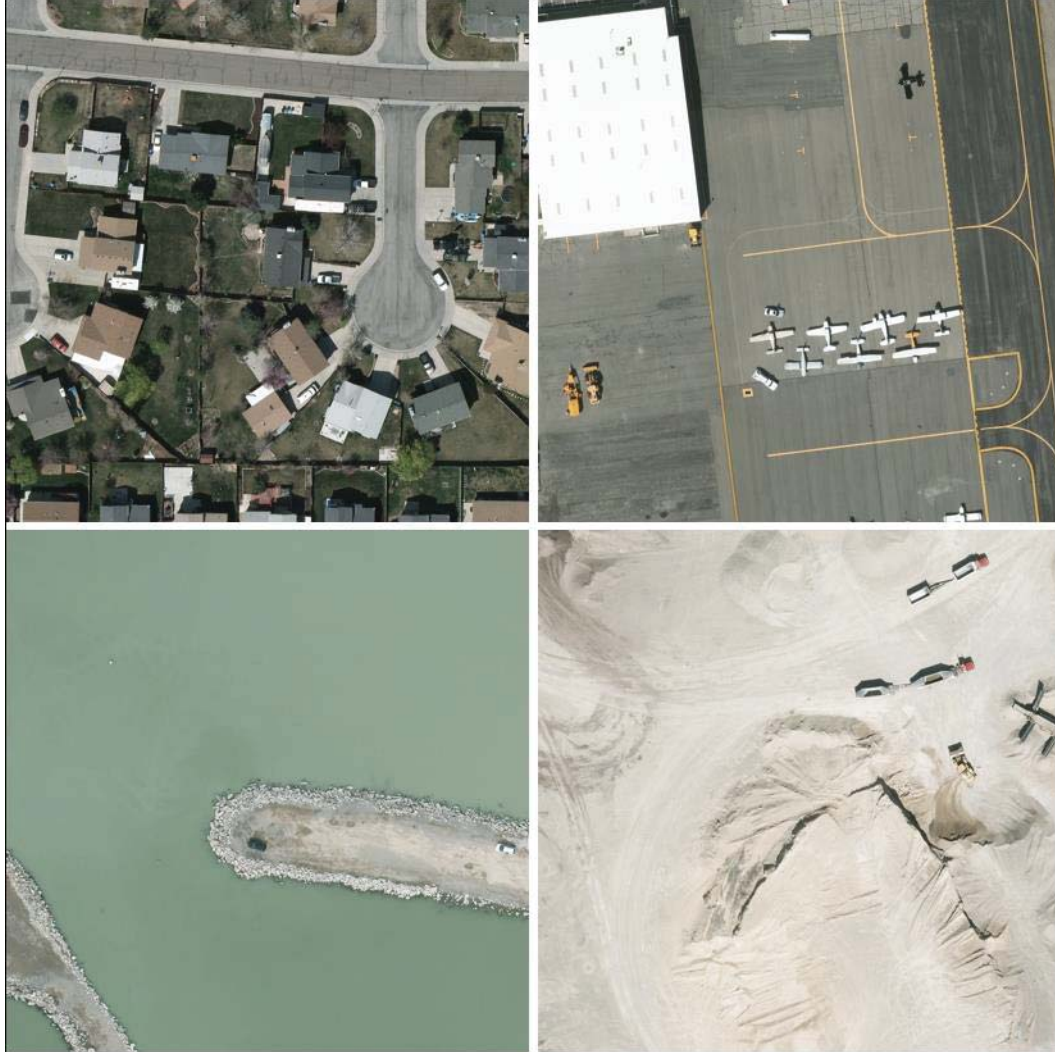


Şekil 4.8. RPN ağı

5. VERİ KÜMESİ VE VERİ ARTIRIMI

Giriş bölümünde belirtildiği gibi bu çalışmanın motivasyonu, küçük nesnelerin otomatik tespiti; özel olarak hava görüntüleri üzerinden araçların tespitidir. Bu motivasyonun gerçekleşmesi için uygun veri kümesinin seçimine ihtiyaç duyulmaktadır. Veri kümesinin özellikleri birtakım kısıtları karşılamalıdır; hava görüntüsü üretmek pahalı olacağı için görüntüler serbestçe kullanılabilmesi ve telif içermemelidir. İstihbarat, keşif, gözetleme gibi çeşitli uygulamaların ihtiyaçlarını karşılayacak kadar hedeflerin yeterli tür çeşitliliğine ve sayısına sahip olmalıdır. Arka planlar; yol ve binaların olduğu yalnızca kentsel alanlar olmamalıdır, mümkün olduğunca çeşitli olmalıdır. Hedefler görüntü üzerinde piksel olarak küçük yer kaplamalıdır. Bu çalışma için bu kısıtları sağlayan VEDAI [40] veri kümesi seçilmiştir. VEDAI, halka açık Utah AGRC [41] veri tabanından alınarak orto-normalize edilmiş görüntüler içermektedir. Görüntüler 2012 yılının baharında Amerika Birleşik Devletleri'nin Utah eyaleti üzerinde çekilmiştir. Görüntülerin yerden uzaklığı sabittir. Bu özellik kasten seçilmiştir. Çünkü istihbarat, keşif ve gözetleme senaryolarında gözetim sistemi genellikle belirli yükseklik verilerine sahiptir ve bu durum ilgili uygulamalar için aranan bir özelliktir. VEDAI veri kümesinin bir diğer aranan özelliği de çok farklı arka planları barındırmasıdır. Göl, tarla, orman, dağlar, çöl gibi kırsal alanların yanında otoban, otopark, yerleşim alanları ve şantiyeler gibi kentsel arka plan alanlarını içermektedir (Resim 5.1'de gösterilmiştir). VEDAI, her biri 1239 adet olmak üzere 512×512 ve 1024×1024 renkli ve kızılötesi görüntüden oluşmaktadır. Bu çalışmada 1024×1024 çözünürlüğündeki renkli görüntüler üzerinde çalışılmıştır. Dokuz farklı araç türü mevcuttur. Bunlar: uçak, araba, kamyon, panelvan, pikap, kamp arabası, traktör, gemi ve çoğunlukla iş makinelerini içeren diğerleri. Veri kümesi görüntü başına ortalama 5,5 araç sunmaktadır. Çizelge 5.1 her bir araç türünün veri kümesindeki bulunma adedini göstermektedir. Bir görüntüde araçlar, görüntünün ortalama %0,7'sini kaplamaktadır. En küçük boyuta sahip araba sınıfı ise görüntüde toplam piksellerin %0,5'ini kaplamaktadır. 1024×1024 görüntülerde zemin örnekleme mesafesi 25 cm/piksel'dir. Bu çalışmada bazı uygulamaların ([42 ve 43]) aksine farklı görüntü ve boyuttaki uçak, traktör, gemi ve iş makinesi araç türleri de kullanılmıştır. Bu durumun eğitim başarısında bir miktar düşüşe sebep olacağı ön görülse de gerçek zamanlı bir araç tespit uygulamasında bu tür araçların da tespit edilmesi gerekliliğinden dolayı mevcut tür çeşitliliği korunmuştur. VEDAI veri kümesi; değişen arka planları,

içerdiği küçük boyuttaki hedefleri ve bazı durumlarda hedeflerin çevre tarafından kamufle olmasıyla bu çalışma için seçilmiş uygun zorlukta bir veri kümesidir.



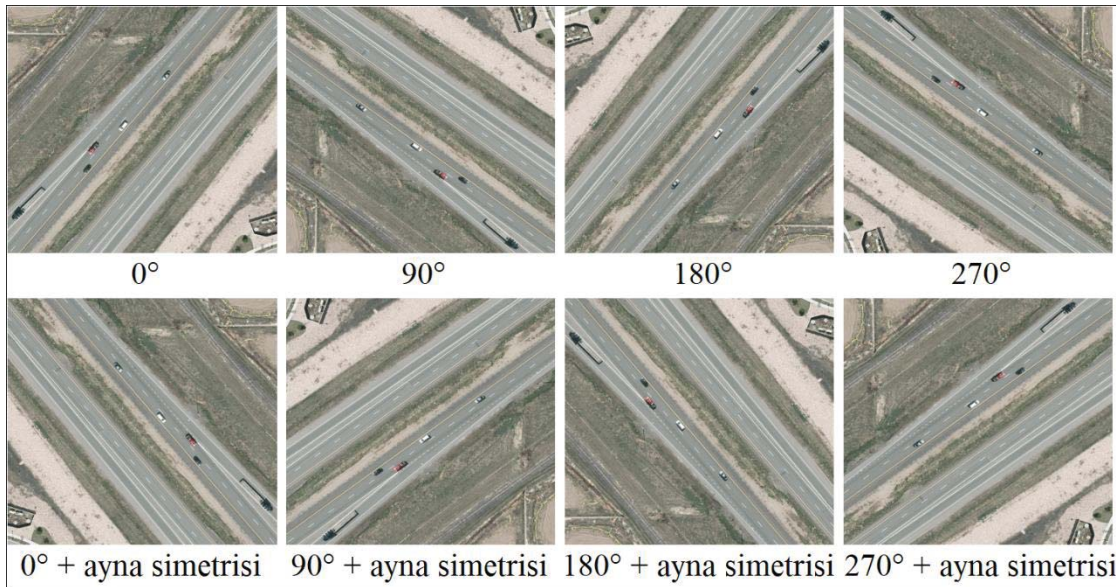
Resim 5.1. VEDAI veri kümesinden örnekler

Çizelge 5.1. VEDAI veri kümesi araç sayısı

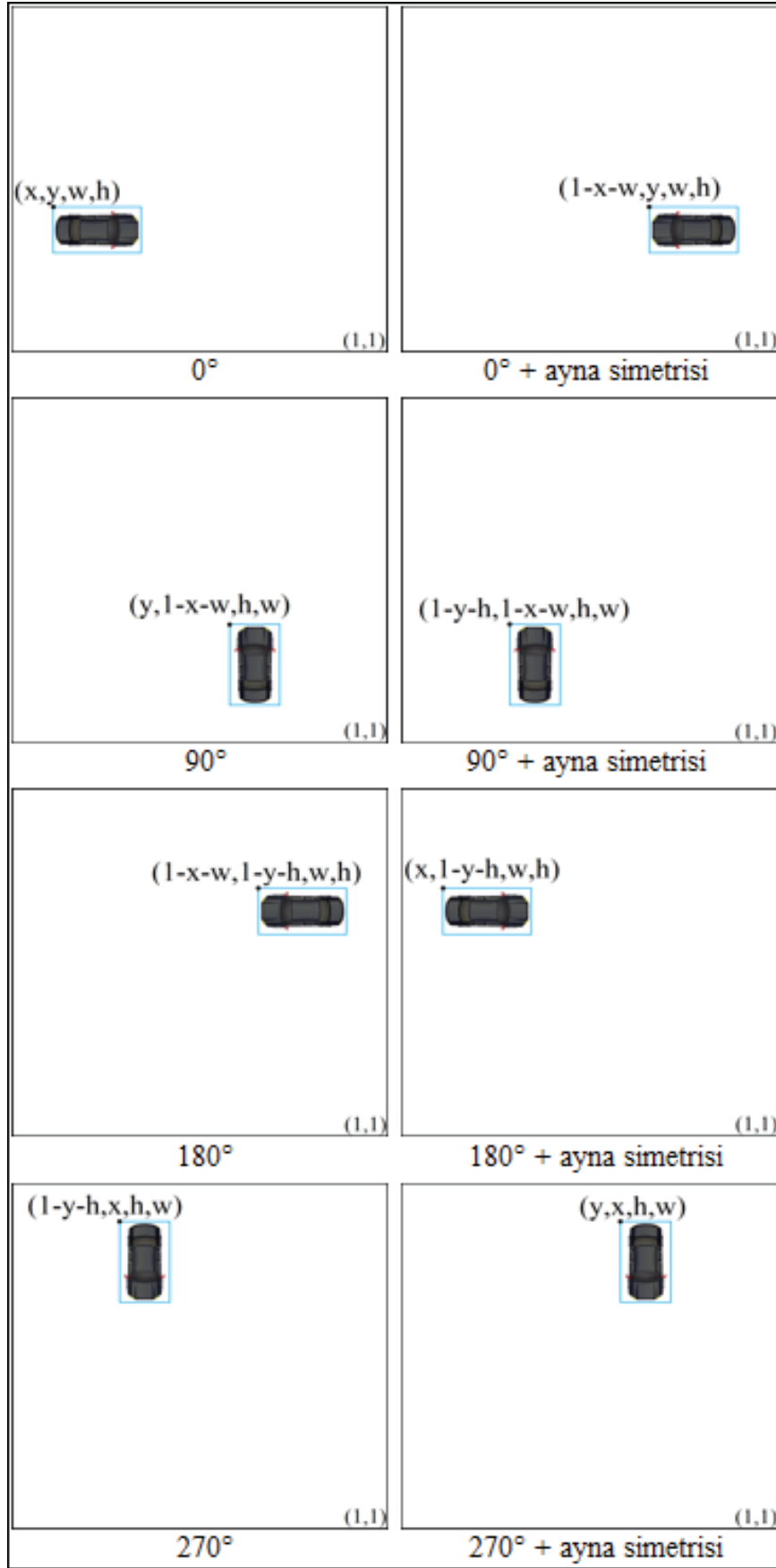
Sınıf	Adet
Araba	1340
Kamyon	300
Pikap	950
Traktör	190
Kamp arabası	390
Tekne	170
Panelvan	100
İş arabası	200
Uçak	47

VEDAI veri kümesi, iki dosya kümesinden oluşmaktadır. Birinci dosya kümesi 1239 adet görüntü dosyası, ikinci dosya kümesi her bir görüntü üzerindeki araçların sınıf ve koordinat bilgilerini içeren 1239 adet açıklama dosyası içermektedir. Açıklama dosyaları, belli bir görüntü üzerindeki her bir araç için beş adet bilgi tutmaktadır. Bu bilgiler; aracın sınıf etiketi ve araç koordinatlarını dikdörtgen bir kutu içerisinde belirten sınırlayıcı kutunun dört noktası x , y , w , h 'dır. Burada x sınırlayıcı kutu sol üst köşe yatay eksen koordinatı, y sınırlayıcı kutu sol üst köşe dikey eksen koordinatı, w sınırlayıcı kutusunun piksel cinsinden genişliği ve h sınırlayıcı kutusunun piksel cinsinden yüksekliğini belirtmektedir.

VEDAI veri kümesi, Çizelge 5.1'de belirtildiği gibi bazı sınıflar için yeterli miktarda örneğe sahip değildir. Örneğin; 148 panelvan araç sınıfı, 171 tekne araç sınıfı ve 190 traktör araç sınıfı için örneğe sahiptir. Bu durumun eğitim başarısına olumsuz etki edeceği düşünüldüğü için veri kümesi üzerinde veri artırma işlemi uygulanmıştır. Artırma işlemi için iki yöntem kullanılmıştır. İlk olarak, veri kümesindeki her bir görüntü saat yönünün tersinde 90° , 180° ve 270° 'lik açılarla döndürülmüştür. Ardından orijinal görüntünün ve döndürülmüş görüntülerin dikey ekseninde ayna simetrisi alınmıştır (Resim 5.2'de gösterilmiştir). Böylece orijinal veri kümesinden sekiz kat büyüklükte bir veri kümesi elde edilmiştir. Bu işlemler uygulanırken görüntü üzerindeki araçların koordinatları da Şekil 5.1'de gösterildiği gibi döndürme ve ayna simetrisi işlemlerine tabi tutulmuştur.



Resim 5.2. Veri artırımı



Şekil 5.1. Veri artırımının araç koordinatlarına etkisi

6. DAHA HIZLI RCNN VE RESNET MODİFİKASYONLARI

Bu bölümde, daha hızlı RCNN algoritmasının ve ResNet modelinin hava görüntüleri üzerinde başarılı bir şekilde uygulanabilmesi için yapılan değişiklikler açıklanmaktadır. İlk değişiklik, daha hızlı RCNN algoritmasının RPN modülünde kullanılan çapa kutularının sayısı ve boyutudur. İkinci değişiklik, ResNet modelinin konvolüsyon katman sayısının değiştirilmesini içermektedir. Üçüncü değişiklik ise, daha hızlı RCNN algoritmasında bölge önerilerini üretmesi için RPN ağına beslenen özellik haritası boyutunun değiştirilmesini içermektedir.

6.1. Çapa Kutu Sayısı ve Boyutu

Bölüm 4.3.1’de belirtildiği gibi daha hızlı RCNN algoritması, orijinalde bölge önerilerinin üretildiği RPN ağına {128,128; 128,256; 256,128; 256,256; 256,512; 512,256; 512,512; 512,1024 ve 1024,512} boyutlarında dokuz adet çapa kutusu kullanmaktadır. Bu çapa kutuları; geleneksel büyük boyutlu farklı en-boy oranlarına sahip nesne tespit uygulamaları için yeterlidir ancak nesnenin görüntü sensöründen sabit uzaklıkta olduğu ve daha küçük nesneler içeren bir veri kümesi için uygun değildir. Orijinal daha hızlı RCNN algoritmasının kullandığı çapa kutularından daha küçük boyutlardaki çapa kutularının kullanılması daha yararlı olabilir. Ayrıca kullanılan bağlantı kutu sayısı, üretilen bölge önerileri ile orantılı olduğu için eğitim başarısına doğrudan etki edecektir. Bu sebeplerle YOLOv2 algoritmasında [15] kullanılan metot daha hızlı RCNN algoritmasına uygulanmıştır. Daha hızlı RCNN’nin RPN ağı, farklı çapa kutu sayılarını ve boyutlarını kullanacak şekilde değiştirilmiştir. VEDAI veri kümesi üzerinden uygun bağlantı kutularının boyutlarını belirlemek için IoU mesafe metriği kullanılmıştır. Bağlantı kutularının IoU mesafe metriği kullanılarak veri kümesi üzerinden tahmin edilmesi, temel olarak benzer en-boy oranlarına sahip kutuların birlikte kümelenip verilere uyan bağlantı kutularının tahmin edilmesi prensibi ile çalışmaktadır [15]. Veri kümesinin içerdiği tüm görüntülere ait tüm sınırlayıcı kutuların ve çapa kutu sayısının girdi olarak kullanıldığı, IoU mesafe metriği yöntemi kullanılarak; bir, iki, üç, dört ve beş adetlerinde olacak şekilde çapa kutuları oluşturulmuştur. Çizelge 6.1 oluşturulan çapa kutularını göstermektedir. Bir, iki, üç, dört ve beş adetlerinde çapa kutularının kullanıldığı her bir senaryo için Bölüm 4.4’te anlatıldığı gibi daha hızlı RCNN algoritmasının ResNet50 ağı üzerinde uygulandığı daha hızlı RCNN ResNet50 ağ

modelleri oluşturulmuştur. Bir adet çapa kutusu içeren daha hızlı RCNN ResNet50 ağ modeline “Ankor1”, iki adet çapa kutusu içeren daha hızlı RCNN ResNet50 ağ modeline “Ankor2”, üç adet çapa kutusu içeren daha hızlı RCNN ResNet50 ağ modeline “Ankor3”, dört adet çapa kutusu içeren daha hızlı RCNN ResNet50 ağ modeline “Ankor4” ve beş adet çapa kutusu içeren daha hızlı RCNN ResNet50 ağ modeline “Ankor5” ismi verilmiştir.

Çizelge 6.1. Kullanılan çapa kutuları

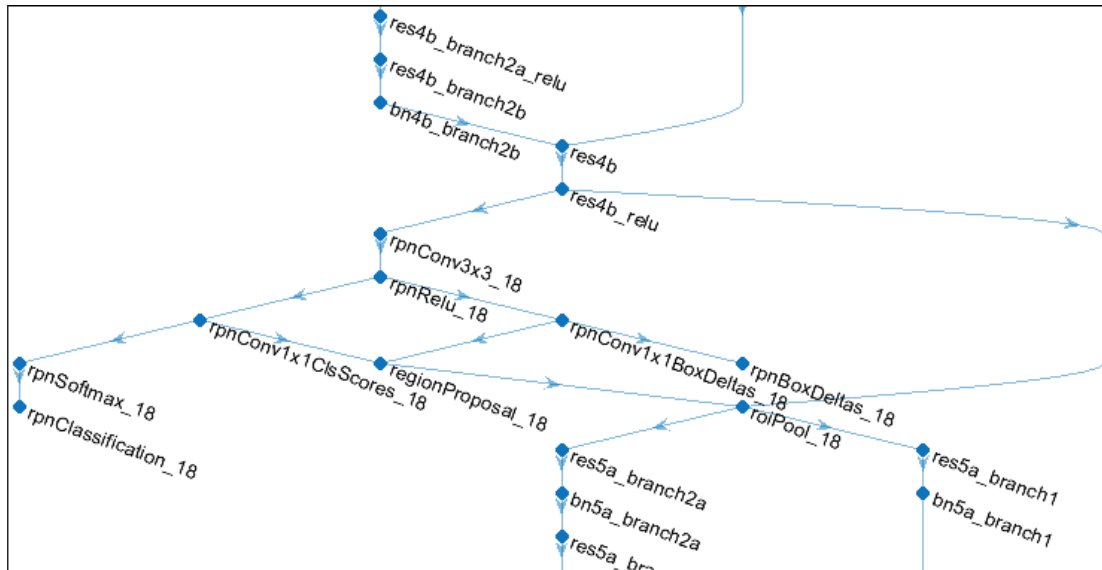
Model İsmi	İçerdiği Çapa Kutu Sayısı	Çapa Kutuları
Ankor1	1	[40,40]
Ankor2	2	[36,36;80,80]
Ankor3	3	[72,76;25,46;47,27]
Ankor4	4	[43,24;77,94;26,47;69,37]
Ankor5	5	[42,23;103,103;66,34;41,68;24,44]

6.2. Konvolüsyon Katman Sayısı

Çizelge 3.1’de gösterildiği gibi ResNet ağı içerdiği katman sayısına göre isim almaktadır. ResNet50; kırk dokuz konvolüsyon katmanı ve bir tam bağlantılı katmandan oluşmaktadır. Benzer şekilde ResNet18; on yedi konvolüsyon katmanı ve bir tam bağlantılı katmandan oluşmaktadır. Standart ResNet ağını kullanan daha hızlı RCNN algoritmasında özellik haritasının çıktısı ve RPN’nin girişi olarak conv4_x konvolüsyon bloğunun son konvolüsyon katmanı kullanılmaktadır. Bu katman; ResNet18 için conv4_b, ResNet50 için conv4_f ve ResNet101 için conv4_b22’dir. ResNet modelinin tüm varyasyonları için conv4_x’ün çıktısı, 14×14’lük özellik haritası çözünürlüğüne sahiptir. Bu, ağ derinliğinin özellik haritası boyutunu değiştirmedikçe göstermektedir. Ancak özellik haritasını oluştururken kullanılan konvolüsyon katman sayısı değiştiği için özellik haritasının öğrenme oranı da değişmektedir. Ayrıca bölge öneri ağı (RPN)’den sonra gelen tespit ağında kullanılan konvolüsyon katman sayısı da ağı öğrenmesini etkilemektedir. Farklı derinlikteki ResNet modellerinin kullanmasının faydalı olabileceği değerlendirilmektedir. Bu sebeplerle ImageNet’te daha önceden eğitilmiş ResNet18, ResNet50 ve ResNet101 ağlarının yanı sıra bu ağlara farklı sayıda konvolüsyon katmanlarını ekleme ve çıkarma yaparak yeni daha hızlı RCNN ResNet ağ modelleri deney amaçlı olarak oluşturulmuştur. Oluşturulan bu ağ modellerinde Çizelge 6.1’deki çapa kutularından ağ başarısı en yüksek olan kullanılmıştır. ResNet18 ağına daha hızlı RCNN algoritmasının uygulandığı modele “daha hızlı RCNN

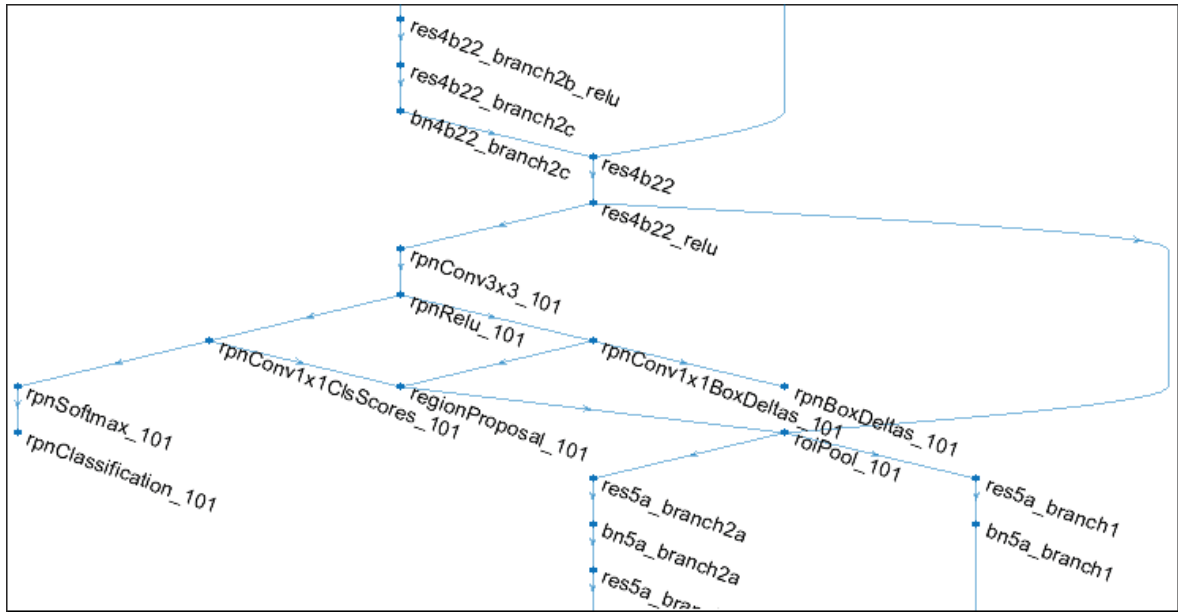
ResNet18” ve ResNet101 ağına daha hızlı RCNN algoritmasının uygulandığı modele “daha hızlı RCNN ResNet101” isimleri verilmiştir. Bölüm 6.1’de farklı çapa kutuları ile oluşturulan ResNet50 ağının kullanıldığı modellerinden başarımı en yüksek olan için “daha hızlı RCNN ResNet50” ismi kullanılmıştır. Orijinal ResNet ağlarında konvolüsyon katman sayısının değiştirme çalışmalarından biri ResNet18’de conv2_x bloğundaki iki artık bloğa, blokların çıkış boyutlarını değiştirmeyecek şekilde, 2’şer ilave konvolüsyon katmanının eklenmesidir. Konvolüsyon katmanı ekleme işleminden sonra daha hızlı RCNN algoritması uygulanmış ve bu modele “daha hızlı RCNN Resnet18_artı4” ismi verilmiştir. Bir diğer çalışma ise, ResNet50’de conv4_x konvolüsyon bloğunun barındırdığı 6 artık bloktan son iki artık bloğun ağdan çıkarılmasıdır. ResNet50’den iki artık blok yapısı atılmış ve böylece 6 konvolüsyon katmanı eksilmiştir. Konvolüsyon katmanı silme işleminden sonra daha hızlı RCNN algoritması uygulanmış ve bu model “daha hızlı RCNN ResNet50_eksi6” olarak isimlendirilmiştir.

Daha hızlı RCNN ResNet18 modelinin oluşturulması için, Bölüm 4.4’de ResNet50 ağına daha hızlı RCNN algoritmasının uygulanmasında anlatıldığı gibi, ResNet18 ağının çıktı katmanları güncellenmiş ve bölge öneri ağı eklenmiştir. Özellik çıkarıcı katman olarak belirlenen 4. konvolüsyon bloğunun (conv4_x) son katmanı ResNet18’de “res4b_relu” isimli ReLu katmanıdır. Bu sebeple “res4b_relu” katmanına bağlı “res5a_branch2a” ve “res5a_branch1” isimli katmanların bağlantısı kesilmiş ve RPN ağı eklenmiştir. Şekil 6.1 daha hızlı RCNN ResNet18 modelinin bölge önerileri ağını göstermektedir.



Şekil 6.1. Daha hızlı RCNN ResNet18 modeli bölge önerileri ağı

Daha hızlı RCNN ResNet101 modelinin oluşturulması için ise, yine Bölüm 4.4’de ResNet50 ağına daha hızlı RCNN algoritmasının uygulanmasında anlatıldığı gibi, ResNet101 ağının çıktı katmanları güncellenmiş ve bölge öneri ağı eklenmiştir. Özellik çıkarıcı katman olarak belirlenen 4. konvolüsyon bloğunun (conv4_x) son katmanı ResNet101’de “res4b22_relu” isimli ReLu katmanı olduğu için, “res4b22_relu” katmanına bağlı “res5a_branch2a” ve “res5a_branch1” isimli katmanların bağlantısı kesilmiş ve RPN ağı eklenmiştir. Şekil 6.2 daha hızlı RCNN ResNet101 modelinin bölge önerileri ağını göstermektedir.



Şekil 6.2. Daha hızlı RCNN ResNet101 modeli bölge önerileri ağı

Daha hızlı RCNN Resnet18_artı4 modelinde, ResNet18 ağının katman sayısında artırma gidilmiştir. Artırma işlemi için ağı artık blok eklemek yerine, artık blokların içerisindeki konvolüsyon katman sayısının artırılması tercih edilmiştir. Bunun için, ilk olarak ikinci konvolüsyon bloğu conv2_x’in birinci artık bloğundaki ilk konvolüsyon katmanından sonra gelen ReLu katmanı “res2a_branch2a_relu” ile ReLu katmanının ardından gelen “res2a_branch2b” isimli konvolüsyon katmanının bağlantısı kesilmiştir. Ardından sırasıyla birbirine bağlı bir şekilde; 64 adet 3×3 boyutunda filtreye, [1,1] dolgu ve 1 adım sayısına sahip “resnet2a_branch2c” isimli bir konvolüsyon katmanı, “bn2a_branch2c” isimli bir yığın normalizasyonu katmanı, “resnet2a_branch2c_relu” isimli bir ReLu katmanı, bir öncekiyle aynı özelliklere sahip “resnet2a_branch2d” isimli bir konvolüsyon katmanı, “bn2a_branch2d” isimli bir yığın normalizasyonu katmanı ve “resnet2a_branch2d_relu” isimli bir ReLu katmanı tanımlanmıştır. Ağın bütünlüğünü sağlamak için bağlantısı kesilen

“res2a_branch2a_relu” katmanı ile yeni tanımlanan katmanların ilki olan “resnet2a_branch2c” katmanının bağlantısı ve yeni tanımlanan katmanların sonuncusu olan “resnet2a_branch2d_relu” katmanı ile “res2a_branch2b” katmanının bağlantısı yapılmıştır. İki konvolüsyon katmanı daha eklemek üzere, yine conv2_x konvolüsyon bloğunda ancak bu sefer ikinci artık bloğundaki ilk konvolüsyon katmanından sonra gelen ReLu katmanı “res2b_branch2a_relu” ile bu ReLu katmanının ardından gelen “res2b_branch2b” isimli konvolüsyon katmanının bağlantısı kesilmiştir. Ardından sırasıyla birbirine bağlı bir şekilde; bir önceki artık bloğa eklenen konvolüsyon katmanları ile aynı özelliklere sahip “resnet2b_branch2c” isimli bir konvolüsyon katmanı, “bn2b_branch2c” isimli bir yığın normalizasyonu katmanı, “resnet2b_branch2c_relu” isimli bir ReLu katmanı, “resnet2b_branch2d” isimli bir öncekiyle aynı özelliklere sahip bir konvolüsyon katmanı, “bn2b_branch2d” isimli bir yığın normalizasyonu katmanı ve “resnet2b_branch2d_relu” isimli bir ReLu katmanı tanımlanmıştır. Ağın bütünlüğünü sağlamak için bağlantısı kesilen “res2b_branch2a_relu” katmanı ile bu artık blok için yeni tanımlanan katmanların ilki olan “resnet2b_branch2c” katmanının bağlantısı ve bu artık blok için yeni tanımlanan katmanların sonuncu olan “resnet2b_branch2d_relu” katmanı ile “res2b_branch2b” katmanının bağlantısı yapılmıştır. Her iki artık bloğa eklenen konvolüsyon katmanları sahip olduğu parametreler itibarı ile, ResNet’in her konvolüsyon bloğunun kendi içerisinde çıktı boyutunu koruması prensibini korumuştur. Şekil 6.3 ResNet18’in 4 ilave konvolüsyon katmanı eklenmiş halini göstermektedir. Bu işlemlerin ardından bu ağa daha hızlı RCNN algoritması uygulanarak, çıktı katmanları güncellenmiş ve bölge öneri ağı eklenmiştir.

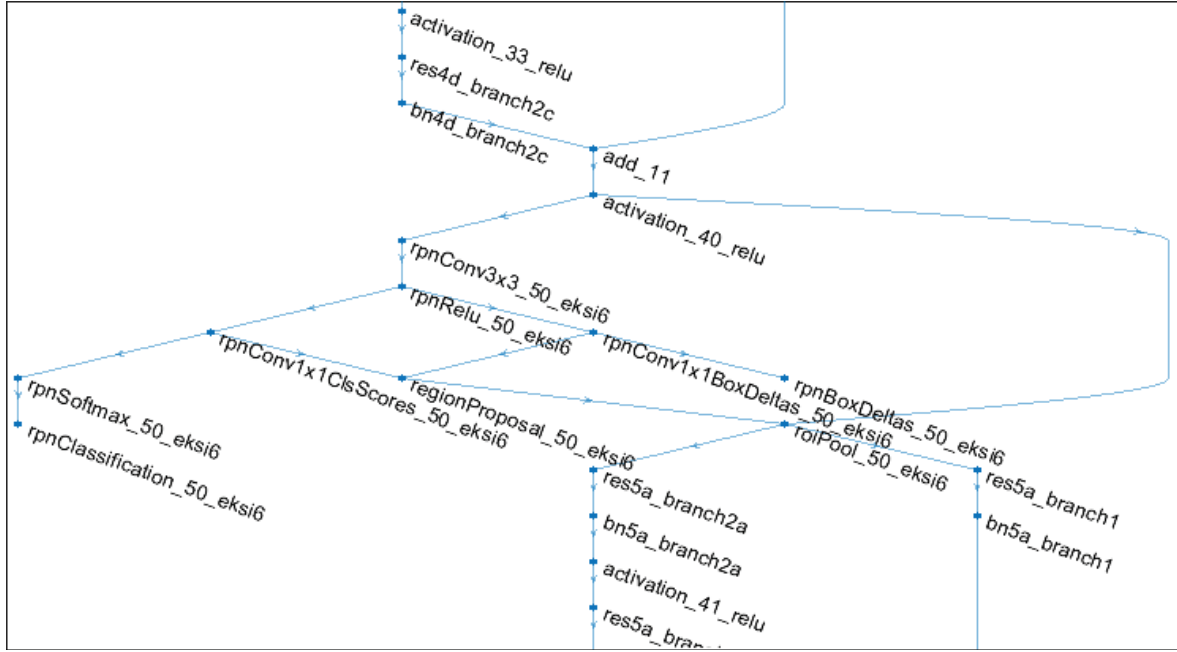
Katman sayısındaki değişikliklerle oluşturulan sonuncu deneysel ağ modeli olan daha hızlı RCNN ResNet50_eksi6’nin oluşturulması için, dördüncü konvolüsyon bloğundaki sondan 2 artık blok ağdan atılmıştır. Bunun için, “add_11” isimli toplama katmanı ile ona bağlı “activation_34_relu” Relu katmanının bağlantısı ve “add_13” isimli toplama katmanı ile ona bağlı “activation_40_relu” ReLu katmanının bağlantısı kesilmiştir. “add_11” ile “activation_40_relu” arasında bulunan; “activation_34_relu”, “res4e_branch2a”, “add_12”, “bn4e_branch2a”, “activation_35_relu”, “res4e_branch2b”, “bn4e_branch2b”, “activation_36_relu”, “res4e_branch2c”, “bn4e_branch2c”, “activation_37_relu”, “res4f_branch2a”, “add_13”, “bn4f_branch2a”, “activation_38_relu”, “res4f_branch2b”, “bn4f_branch2b”, “activation_39_relu”, “res4f_branch2c” ve “bn4f_branch2c” katmanları ağdan atılmıştır. Ardından “add_11” ile “activation_40_relu” katmanlarının bağlantısı sağlanmıştır. Yine bu işlemler sonucunda; konvolüsyon bloğu içerisindeki çıktı boyutu

değişmemiş, korunmuştur. Son olarak bu ağı daha hızlı RCNN algoritması uygulanmıştır. Şekil 6.4 daha hızlı RCNN ResNet50_eksi6 modelinin yukarıda anlatılan ağdan atılan katmanlar sonucunda oluşan yapıyı ve bölge öneri ağının bağlantılarını göstermektedir.

Konvolüsyon katman sayısının küçük nesnelerin tespitinde ağ başarısına etkisinin araştırılması amacıyla oluşturulan daha hızlı RCNN ResNet18, daha hızlı RCNN ResNet50 ve daha hızlı RCNN ResNet101 modellerinden sonra; neden daha hızlı RCNN Resnet18_artı4 ve daha hızlı RCNN ResNet50_eksi6 deneysel modellere ihtiyaç duyulduğu, Bölüm 7’de deneysel çalışmaların sonuçlarıyla bağlantılı olarak açıklanmaktadır.



Şekil 6.3. Daha hızlı RCNN Resnet18_artı4 modeli conv2_x konvolüsyon bloğu

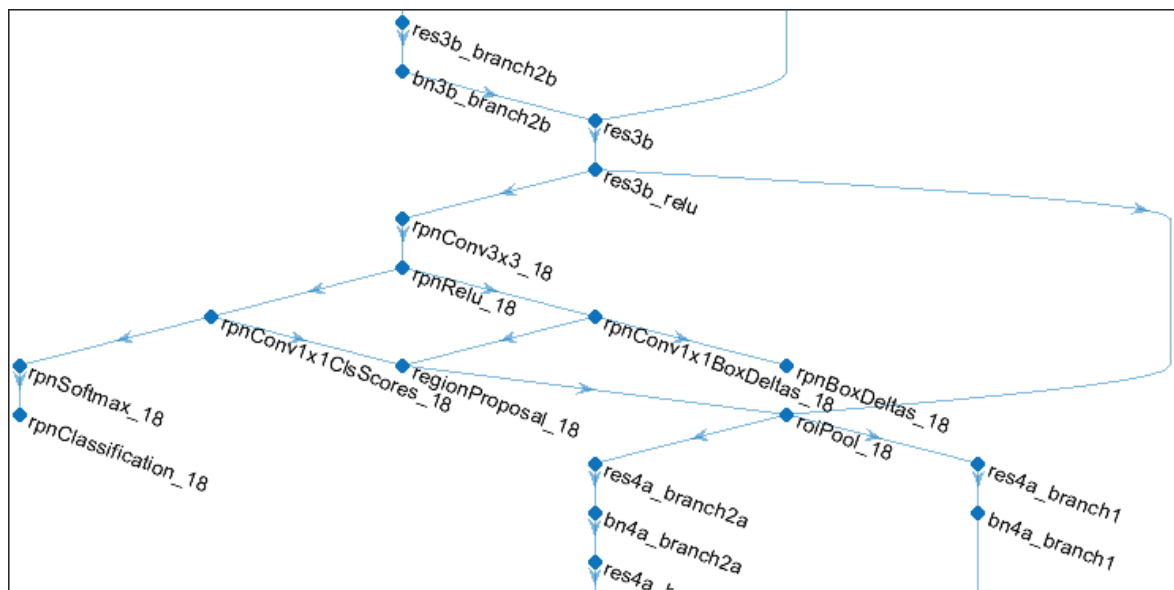


Şekil 6.4. Daha hızlı RCNN ResNet50_eksi6 modeli

6.3. Özellik Haritası Boyutu

Bölüm 3 ve 4.4'te anlatıldığı gibi ResNet ağını kullanan standart bir daha hızlı RCNN algoritmasında, RPN'in bölge önerilerini üretmek için kullandığı özellik haritası katmanı conv4_x'ün son konvolüsyon katmanıdır ve ağ derinliğinden bağımsız olarak bu katmanın çıktısı 14×14 boyutundadır. Buna göre VEDAI görüntülerindeki araçların özellik haritasında, bir aracın tek bir noktaya karşılık geldiği düşünülebilir. Bir aracın özellik haritasında birden fazla nokta ile temsil edilmesi ağın başarısında etkili olabileceği değerlendirilmektedir. Bir aracın özellik haritasında birden fazla nokta ile temsil edilmesi, özellik haritası boyutunun artırılması ile mümkündür. Özellik haritasının boyutunun artırılması için birkaç yöntem kullanılabilir. Birinci yöntem, standart ResNet'in 224×224 olan ağ giriş boyutunun 448×448 yapılmasıdır. Giriş boyutunun en ve boy için iki katına çıkması, conv4_x'in çıktı boyutunun dört katına çıkmasına yani 28×28 olmasına neden olacaktır. Ancak; bir aracın görüntüde kapladığı alan değişmediği için, özellik haritası boyutu dört katına çıkmış olsa da aracın özellik haritasında temsil edildiği nokta sayısı değişmeyecektir. Üstelik daha hızlı RCNN algoritması ile yapılan nesne tespit çalışmalarında ağın görüntünün daha küçük bölümlerini analiz etmesi için ağın giriş boyutunun veri kümesindeki nesnelere yakın olması önerilmektedir [44]. VEDAI veri kümesindeki en büyük boyutlu tır araç sınıfı ortalama 106×21 boyuta sahiptir. Dolayısıyla

giriş boyutunun artırılmasının ağ başarısını düşüreceği değerlendirilmektedir. Özellik haritası boyutunu artırmanın yöntemlerinden bir diğeri, RPN'in giriş olarak kullandığı özellik haritası katmanını 14×14 boyutundaki conv4_x konvolüsyon bloğunun son katmanı olarak seçmek yerine, 28×28 boyutundaki conv3_x konvolüsyon bloğunun son katmanı olarak seçilmesidir. Bu durumda özellik haritası daha sıkı bir ağ ile oluşturulacak ancak bölge önerilerinden sonraki tespit ağı daha derin olacaktır. Bu yöntem, görüntüde küçük araçları içeren VEDAI veri kümesinde başarıyı sağlayacağı değerlendirilerek, ResNet18 ağına uygulanmıştır. Bu yöntemin uygulanması için ResNet18 ağının seçilmesinin nedeni Bölüm 7'de deneysel sonuçlarla bağlantılı bir şekilde açıklanmaktadır. ResNet18 ağına, daha hızlı RCNN'nin ağın dördüncü konvolüsyon bloğunun son katmanının özellik çıkarma katmanı olarak seçilmesi geleneksel yaklaşımının değiştirilerek ağın üçüncü konvolüsyon bloğunun son katmanının özellik çıkarma katmanı olarak seçilmesi deneysel çalışmanın uygulanması sonucunda oluşturulan model “daha hızlı RCNN ResNet18_conv3” olarak isimlendirilmiştir. Bu modelde daha hızlı RCNN ResNet18 modelinden farklı olarak; ResNet18 ağına daha hızlı RCNN algoritması uygulanırken bölge önerileri ağının ağı eklenmesi aşamasında; “res3b_relu” isimli ReLu katmanı, özellik çıkarma katmanı olarak seçilmiş ve bu katmanın “res4a_branch2a” ve “res4a_branch1” katmanları ile bağlantısı kesilerek bölge önerileri ağının bağlantıları yapılmıştır. Şekil 6.5 daha hızlı RCNN ResNet18_conv3 modelinin bölge önerileri bağlantılarını göstermektedir. Oluşturulan bu ağ modelinde de Çizelge 6.1'deki çapa kutularından ağ başarısı en yüksek olan kullanılmıştır.



Şekil 6.5. Daha hızlı RCNN ResNet18_conv3 modeli

7. DENEYSEL ÇALIŞMALAR

Deneysel çalışmalar için gerekli olan ağ eğitim ve test kodları Matlab2019b yazılımında yazılmıştır. Tüm deneyler Google bulut sistemi üzerinden uzaktan bağlanılan Intel Xeon CPU (4 çekirdekli, 2,30 GHz) 15GB ram, NVIDIA Tesla T4 GPU (16GB ram) ve Windows Server 2019 ile donatılmış bir bilgisayar üzerinden gerçekleştirilmiştir. Veri kümesinin %80'i eğitim, %20'si test için kullanılmıştır. Eğitimlerde mini yığın boyutu (batch size) ismi verilen, bir görüntüden pozitif ve negatif nesne örneklerinin rastgele üretilmesine dayanan örnekleme stratejisi kullanılmıştır. Mini yığın boyutu, eğitimi hızlandıran aynı zamanda GPU bellek kullanımını önemli ölçüde artıran bir parametredir. Kullanılan GPU bellek boyutu sınırlamasından dolayı eğitimlerde mini parti boyutu 4 olarak seçilmiştir. Böylece her iterasyonda rastgele 4 pozitif ve negatif nesne önerisi, bir alt veri kümesi oluşturmak için algoritma tarafından üretilmiştir. Pozitif ve negatif nesne önerileri, Eş. 7.1'de tanımlanan IoU Jaccard indeksi kullanılarak belirlenmektedir:

$$IoU = \frac{A_{hedef} \cap A_{tahmin}}{A_{hedef} \cup A_{tahmin}} \quad (7.1)$$

Denklemden A_{hedef} , veri kümesinde nesnenin belirtilen koordinatlarını yani olması gereken alanı; A_{tahmin} ise tahmin edilen alanı temsil etmektedir. Eğitimlerde ağ IoU indeksi; 0,7'den büyük değerler için pozitif nesne; 0,1 ile 0,3 arasında olan değerler için negatif nesne atfında bulunulmuştur. Bu değerler, *Negatif Kesişim Aralığı* ve *Pozitif Kesişim Aralığı* parametreleri ile eğitimden önce ağda belirtilmiştir. Maksimum epok sayısı 25 seçilmiştir. Başlangıç öğrenme oranı 0,001 kullanılmış, bu değer her 12 epok'ta bir 0,1 çarpanı ile düşürülmüştür. Çizelge 7.1'de eğitimde kullanılan parametreler gösterilmektedir.

Çizelge 7.1. Eğitim Parametreleri

Parametreler	Değerler
Mini Parti Boyutu	4
Başlangıç öğrenme hızı	0,001
Öğrenme Hızı Azaltma Periyodu	12 epok
Öğrenme Hızı Azaltma Çarpanı	0,1
Maksimum Epok Sayısı	25
Negatif Kesişim Aralığı	[0; 0,3]
Pozitif Kesişim Aralığı	[0,7; 1]

Modellerin performansının nicel olarak değerlendirilmesi için yaygın olarak kullanılan hassasiyet, geri çağırma oranı, F1 skoru ve ortalama hassasiyet (AP) metrikleri kullanılmıştır. Hassasiyet, geri çağırma oranı ve F1 skoru sırasıyla Eş. 7.2, Eş. 7.3 ve Eş. 7.4 ile hesaplanmaktadır.

$$\text{Geri Çağırma (R)} = \frac{\text{Doğru Pozitif}}{\text{Doğru Pozitif} + \text{Yanlış Negatif}} \quad (7.2)$$

$$\text{Hassasiyet (P)} = \frac{\text{Doğru Pozitif}}{\text{Doğru Pozitif} + \text{Yanlış Pozitif}} \quad (7.3)$$

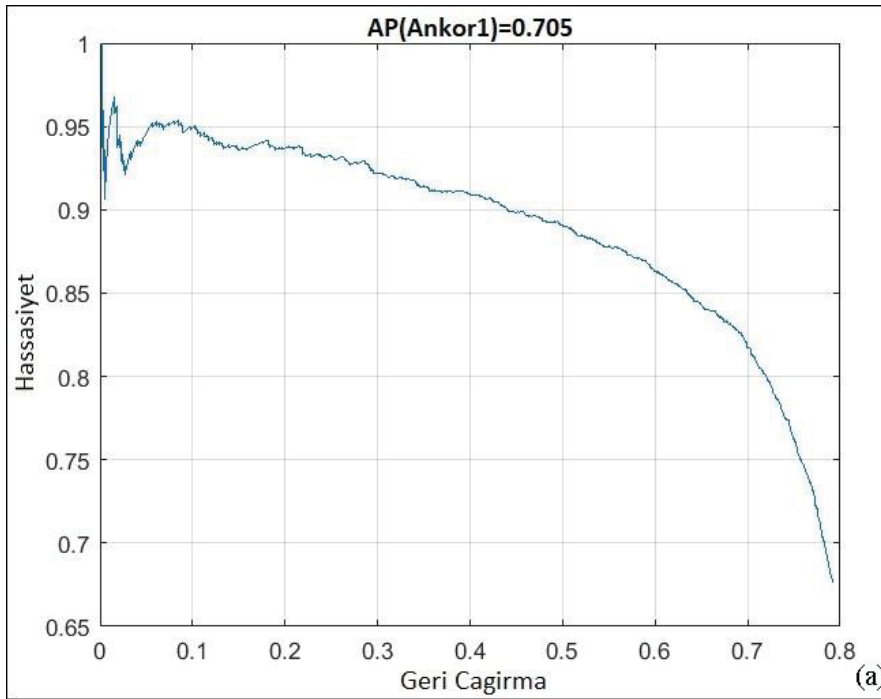
$$\text{F1 skoru} = \frac{2 * \text{Geri Çağırma} * \text{Hassasiyet}}{\text{Geri Çağırma} + \text{Hassasiyet}} \quad (7.4)$$

Test sırasında nesne tespit tahminleri, eğitilmiş ağın oluşturduğu sınırlayıcı kutuların veri kümesinde nesnenin konumunu belirten alan ile çakışmasına göre doğru veya yanlış pozitif olarak atanmaktadır. Yine bu atamada IoU Jaccard indeksi kullanılmıştır. IoU değerinin 0,5'ten büyük olduğu tahminler doğru pozitif olarak atanmaktadır. Bir nesne için birden fazla sınırlayıcı kutu oluşturulması cezalandırılmaktadır. Örneğin; eğitilmiş ağ test sırasında bir nesne için dört adet sınırlayıcı kutu oluşturmuş ise bunun biri IoU değeri 0,5'ten büyük olan doğru pozitif, geri kalan üçü yanlış pozitif olarak değerlendirilmektedir. Tespit edilemeyen bir araç algoritma tarafından arka plan olarak (yanlış olarak) atanmıştır ve bu durum yanlış negatif olarak değerlendirilmektedir. Değerlendirmeler hassasiyet ve geri çağırma oranlarının hesaplanmasında kullanılmaktadır. Bu çalışma için; geri çağırma araçların doğru tespit edilmesinin, hassasiyet ise araç olarak tespit edilenlerin gerçekte kaçının araç olduğunun bir değerlendirmesidir. Düşük hassasiyete ve yüksek geri çağırmaya ya da tam tersi verilere sahip iki modeli karşılaştırmak zordur. Bu zorluğu kolaylaştırmak, geri çağırma ve hassasiyeti tek bir veride değerlendirmek için F1 skoru kullanılmaktadır. Ortalama hassasiyet ise geri çağırma-hassasiyet grafiğinin altında kalan alanı tanımlamaktadır. Ortalama hassasiyet ve F1 skoru detektörün performansını ortaya koyan iki metriktir. Yüksek ortalama hassasiyet ve F1 skor değerleri, yüksek performans anlamına gelir. Geri çağırma-hassasiyet grafiğinde ise grafik üst yatay eksen ve sağ yan dikey eksene ne kadar yakınsa, modelin algılama başarısının o kadar yüksek olduğu anlamına

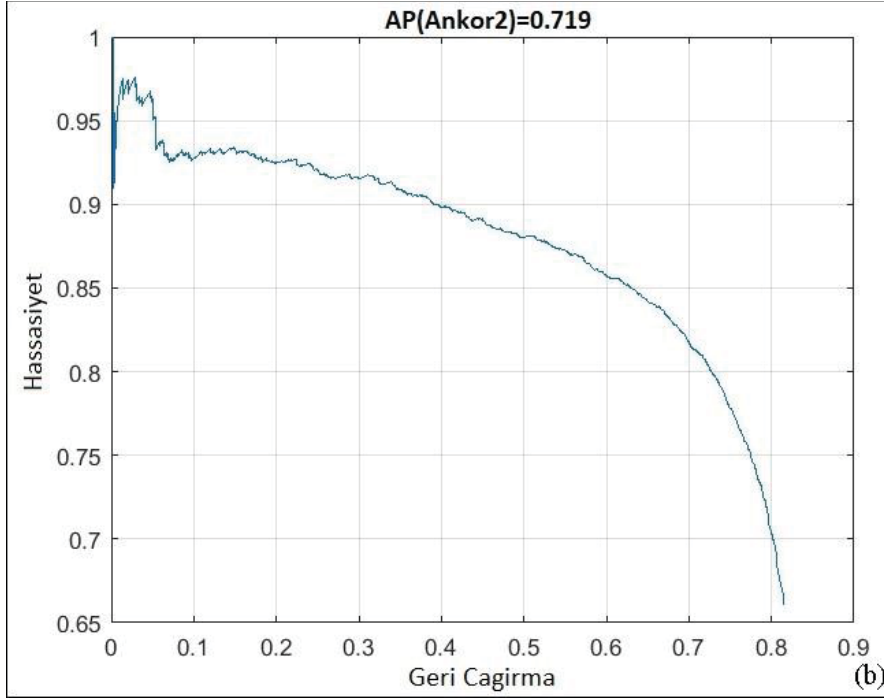
gelmektedir. Bu grafiklerin; sol üst kısmı yüksek hassasiyet açısından, sağ alt kısmı ise yüksek geri çağırma açısından modellerin kıyaslamasına imkân sağlamaktadır.

7.1. Çapa Kutu Sayısı ve Boyutunun Etkisi

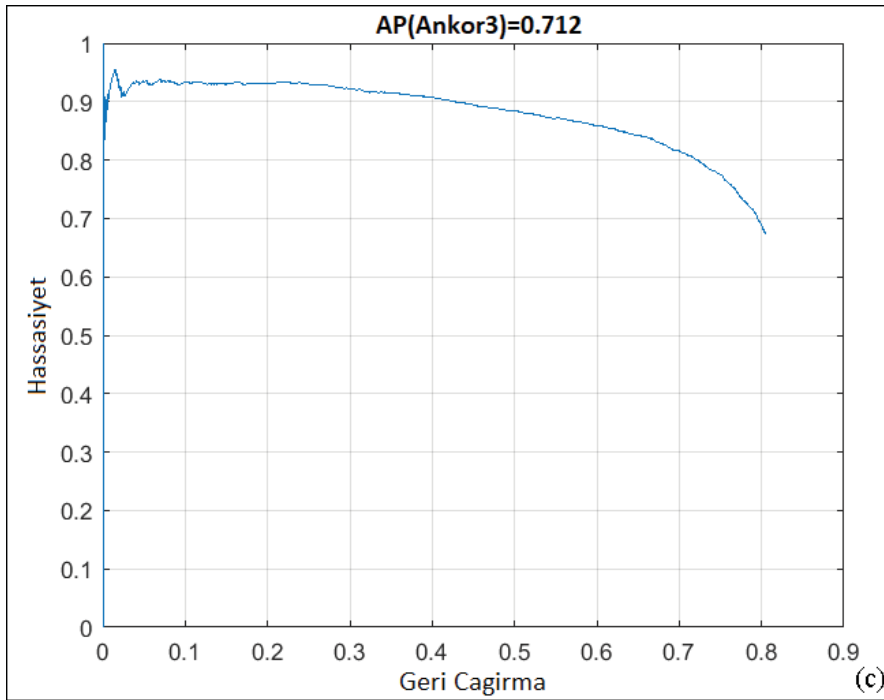
İlk deneyde çapa kutu sayısı ve bu kutuların boyutlarının algılama performansı üzerindeki etkisi araştırılmıştır. ImageNet ile önceden eğitilmiş ResNet50 ağı ile daha hızlı RCNN'nin RPN modülünde kullanılmak üzere Çizelge 6.1'de gösterilen farklı çapa kutu ve boyutunu içeren beş model ile beş farklı eğitim yapılmıştır. Ardından test için ayrılan veri kümesi kullanılarak Bölüm 7'de belirtilen metrikler hesaplanmıştır. Ayrıca her birinin eğitim süreleri kayıt edilmiştir. Şekil 7.1 her bir modele ait geri çağırma-hassasiyet grafiklerini göstermektedir.



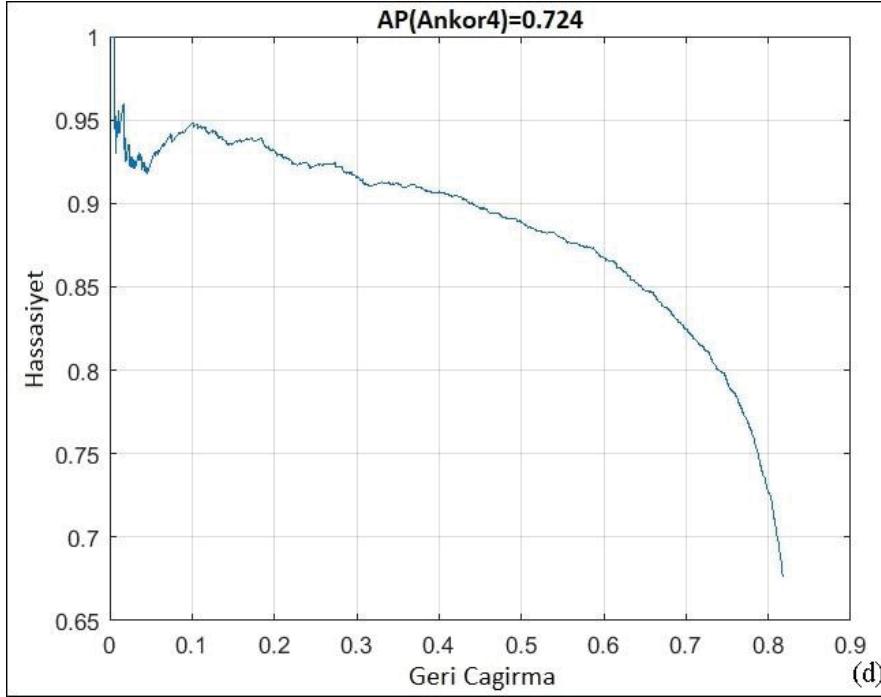
Şekil 7.1. a) Ankor1 modelinin geri çağırma-hassasiyet grafiği b) Ankor2 modelinin geri çağırma hassasiyet grafiği c) Ankor3 modelinin geri çağırma hassasiyet grafiği d) Ankor4 modelinin geri çağırma hassasiyet grafiği e) Ankor5 modelinin geri çağırma hassasiyet grafiği



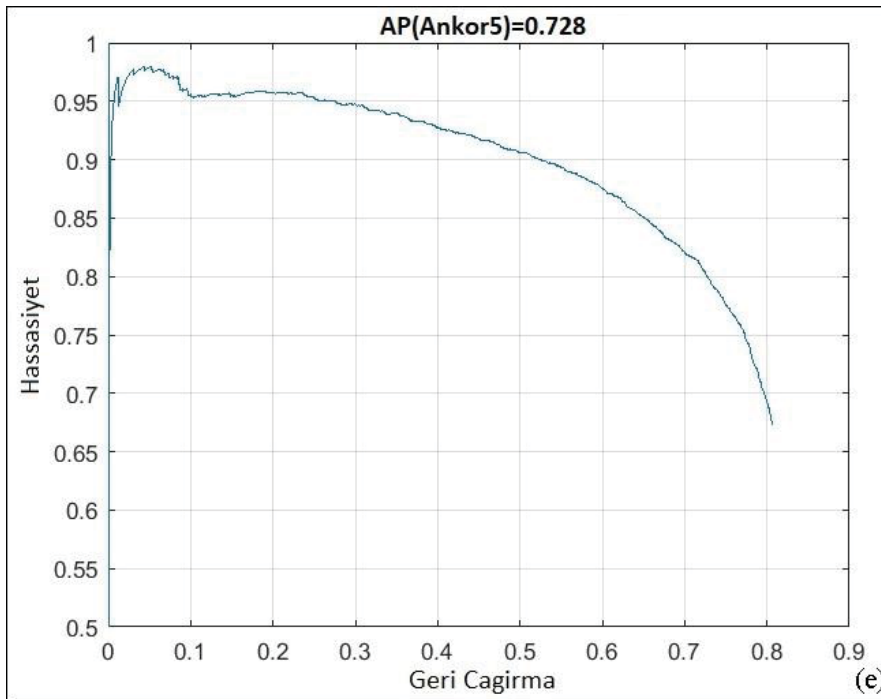
Şekil 7.1. (devam) a) Ankor1 modelinin geri çağırma-hassasiyet grafiği b) Ankor2 modelinin geri çağırma hassasiyet grafiği c) Ankor3 modelinin geri çağırma hassasiyet grafiği d) Ankor4 modelinin geri çağırma hassasiyet grafiği e) Ankor5 modelinin geri çağırma hassasiyet grafiği



Şekil 7.1. (devam) a) Ankor1 modelinin geri çağırma-hassasiyet grafiği b) Ankor2 modelinin geri çağırma hassasiyet grafiği c) Ankor3 modelinin geri çağırma hassasiyet grafiği d) Ankor4 modelinin geri çağırma hassasiyet grafiği e) Ankor5 modelinin geri çağırma hassasiyet grafiği

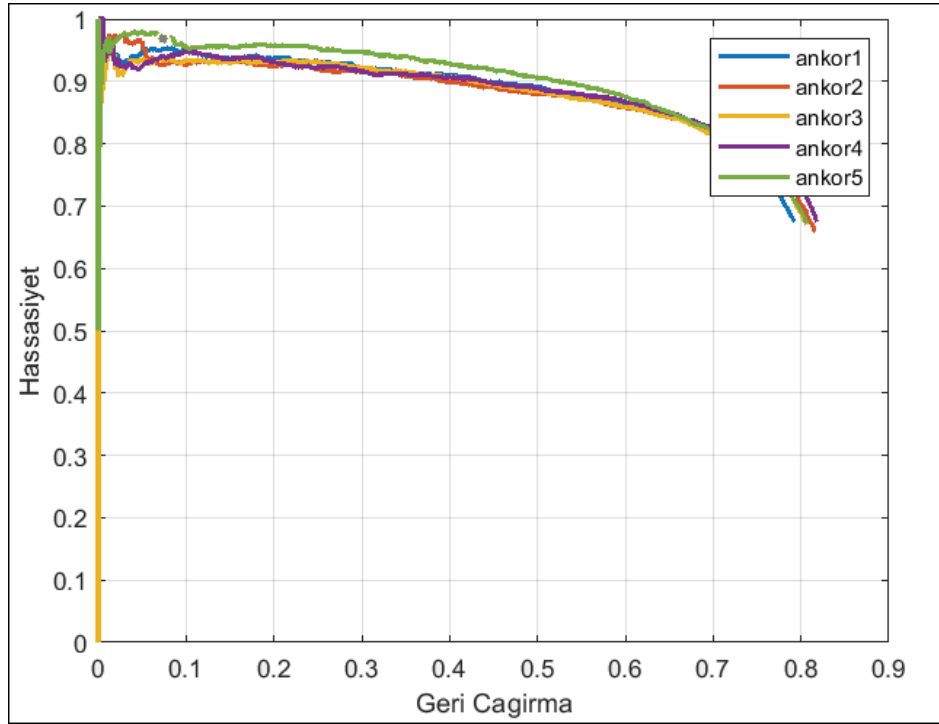


Şekil 7.1. (devam) a) Ankor1 modelinin geri çağırma-hassasiyet grafiği b) Ankor2 modelinin geri çağırma hassasiyet grafiği c) Ankor3 modelinin geri çağırma hassasiyet grafiği d) Ankor4 modelinin geri çağırma hassasiyet grafiği e) Ankor5 modelinin geri çağırma hassasiyet grafiği



Şekil 7.1. (devam) a) Ankor1 modelinin geri çağırma-hassasiyet grafiği b) Ankor2 modelinin geri çağırma hassasiyet grafiği c) Ankor3 modelinin geri çağırma hassasiyet grafiği d) Ankor4 modelinin geri çağırma hassasiyet grafiği e) Ankor5 modelinin geri çağırma hassasiyet grafiği

Karşılaştırmasının daha iyi yapılabilmesi için Şekil 7.2’de bu grafikler üst üste çizdirilmiştir.



Şekil 7.2. Farklı çapa boyut ve sayılarına sahip modellerin R-P grafiği

Çizelge 7.2, beş farklı bağlantı kutusu ile oluşturulan modellerin test sonuçlarını ve eğitim sürelerini göstermektedir. Eğitim süresi saat cinsinden verilmektedir.

Çizelge 7.2. Farklı bağlantı kutularına ait sonuçlar

Model	R	P	AP	F1	Eğitim Süresi
Ankor1	0,791	0,676	0,705	0,729	74,6
Ankor2	0,815	0,660	0,719	0,729	80,8
Ankor3	0,806	0,672	0,712	0,733	82,2
Ankor4	0,818	0,676	0,724	0,740	82,4
Ankor5	0,806	0,673	0,728	0,734	84,1

Şekil 7.1 ve Şekil 7.2’den, grafiklerin birbirine çok yakın olduğu gözlenmektedir. Grafiklerde yüksek hassasiyet bölgelerine (grafikğin sol üst kısmı) bakıldığında, Ankor3 ve Ankor5 modellerinin düşük geri çağırma değerlerinde düşük hassasiyet gösterdikleri gözlenmiştir. Grafiklerin yüksek geri çağırma bölgelerine (grafikğin sağ alt kısmı) bakıldığında; Ankor1 modelinin bu açıdan en kötü model olduğu, Ankor4 modelinin ise en iyi model olduğu gözlenmiştir. Çizelge 7.1’e göre ise, birkaç istisna dışında çapa kutu sayısı arttıkça; ortalama

hassasiyet ve F1 skor değerleri artmaktadır. Ancak bununla birlikte işlem yükü de arttığı için eğitim süreleri de artış eğilimindedir. Ankor4 modeli, algılama başarısında (hem ortalama hassasiyet hem de F1 skor açısından değerlendirildiğinde) Ankor1, Ankor2 ve Ankor3 modellerine göre radikal bir artış göstermiştir. Ancak Ankor4 modelinden sonra gelen Ankor5'in eğitim başarısında, ortalama hassasiyet açısından bakıldığında radikal bir artış görülmemiş aksine F1 skoru açısından bakıldığında bir düşüş görülmektedir. Bu sonuçlar ve eğitim süreleri dikkate alındığında bundan sonraki deneyler çapa kutu sayısı ve boyutu açısından en iyi model olduğu değerlendirilen Ankor4'te kullanılan çapa kutuları kullanılarak gerçekleştirilecektir.

7.2. Konvolüsyon Katman Sayısının Etkisi

Bu bölümde, ResNet ağ derinliğinin algılama performansına etkisi araştırılmıştır. ResNet50'den daha sığ olan ResNet18 ağını kullanan daha hızlı RCNN ResNet18 ve ResNet50'den daha derin olan ResNet101 ağını kullanan daha hızlı RCNN ResNet101 modelleri ile eğitimler yapılmıştır. Bu bölümdeki deneysel çalışmalarda kullanılan modellerin tüm test sonuçları Çizelge 7.2'de gösterilmiştir. Çizelgede; ResNet50 ağına kullanıldığı ve Bölüm 7.1'de en başarılı sonucu veren Ankor4 modeli “daha hızlı RCNN ResNet50” ismi ile belirtilmiştir.

Çizelge 7.2. Farklı derinlikteki ağlara ait sonuçlar

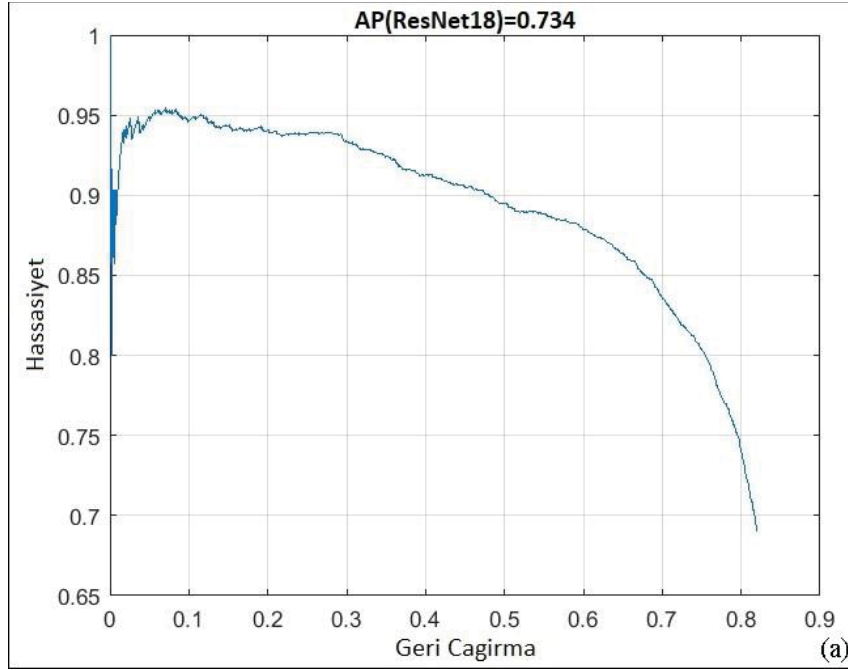
Model	R	P	AP	F1	Eğitim Süresi
Daha hızlı RCNN Resnet18	0,820	0,689	0,734	0,749	46,8
Daha hızlı RCNN Resnet50	0,818	0,676	0,724	0,740	82,4
Daha hızlı RCNN Resnet101	0,812	0,671	0,718	0,735	109,5
Daha hızlı RCNN Resnet18_artı4	0,641	0,495	0,475	0,559	51,2
Daha hızlı RCNN Resnet50_eksi6	0,800	0,677	0,702	0,732	86,5
Daha hızlı RCNN ResNet50_eksi6_tl	0,827	0,725	0,749	0,773	87,6

Sonuçlara göre; hem ortalama hassasiyet (AP) hem de F1 skoru açısından bakıldığında, daha hızlı RCNN ResNet18 modeli, hem daha hızlı RCNN ResNet50'den hem de daha hızlı ResNet101'den daha iyi performans göstermiştir. Bu sonuç [32,45] gibi daha derin ağların daha iyi olduğunu savunan geleneksel bilgiye ve hava görüntülerinde de bu bilginin iyi çalıştığını gösteren [43,46] gibi çalışmalara aykırıdır, ancak [47] ile paralellik göstermektedir. ResNet50'den daha sığ bir ağın, VEDAI veri kümesinde daha başarılı bir sonuç elde etmesinin ardından; ResNet18 ile ResNet50 ağ derinlikleri arasında bir ağın algılama başarısına etkisi araştırılmıştır. Bunun için ResNet18'in conv2_x konvolüsyon bloğunun her iki artık bloğuna 2'şer konvolüsyon bloğu eklenerek oluşturulan ResNet18'den daha derin, daha hızlı RCNN ResNet18_arti4 modeli eğitilmiştir. Eklenen konvolüsyon katmanlarının ağırlık ilklendirilmesi, sıfır ortalama ve sıfır varyans ile bir dağılım sağlayan, Xavier ilklendirici olarak da bilinen, Glorot [48] yöntemi ile yapılmıştır. Ayrıca ResNet50'nin conv4_x konvolüsyon bloğundan 2 artık bloğun yani 6 konvolüsyon katmanının çıkarılarak oluşturulduğu, ResNet50'den daha sığ, daha hızlı RCNN ResNet50_eksi6 modeli eğitilmiştir. Çizelge 7.2'de görüldüğü gibi daha hızlı RCNN ResNet18_arti4 modeli, daha önceden eğitilmiş standart ResNet18 ağını içeren daha hızlı RCNN ResNet18 modelinden çok daha az başarı sağlamıştır. Bu durumun; standart ResNet18'in tüm ağırlıklarının daha önceden eğitilerek optimize olmuş olmasına rağmen, daha hızlı RCNN ResNet18_arti4 modelinde ResNet18 ağına sonradan eklenen dört konvolüsyon katmanı nedeni ile eklenen konvolüsyon katmanı ve sonraki katmanlardaki ağırlıkların yeniden optimize olması gerekmesinden kaynaklandığı değerlendirilmiştir. ResNet'in Bölüm 3'te anlatılan artık blok yapılarının dışına çıkmış olmasının ağ başarısını olumsuz etkilediği de ayrıca gözlemlenmiştir. Aynı şekilde, daha hızlı RCNN ResNet50_eksi6 için de ResNet50 ağının ortasından bir kısım katman çıkarıldığı için yeni modelde çıkarılan katmandan sonraki katman ağırlıkları yine optimize olmaya ihtiyaç duyar. Ancak atılan katmanlardan sonra birleştirilen iki tarafın ağırlıkları arasındaki fark, görece az olduğu için ağırlıkların optimizasyonu çok daha az maliyetle olduğu değerlendirilmiştir. Bu sebepten dolayı daha hızlı RCNN ResNet50_eksi6 modelinin ağ başarısı, daha hızlı RCNN ResNet50 ve daha hızlı RCNN ResNet18'den az ancak hatırı sayılır bir değerde çıkmıştır. Bu aşamaya kadar yapılan tüm eğitimler, ILSVRC yarışmasında önceden eğitilmiş ResNet ağlarına ilk önce daha hızlı RCNN algoritmasına uygun olarak bölge önerileri ağının eklenmesi ve tespit ağının güncellenmesi ile yapılmıştır. Yani örneğin; daha hızlı RCNN ResNet18_arti4 modeli için ilk olarak önceden eğitilmiş hazır ResNet18 modeli daha hızlı RCNN algoritmasına uygun olarak değiştirilmiş, ardından 4 katman ağdan çıkartılmış ve son

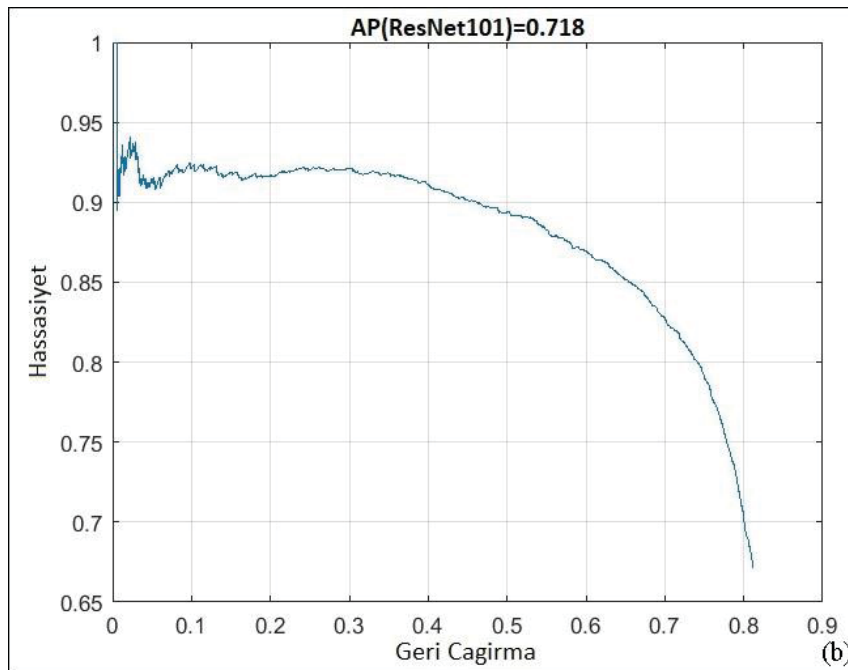
olarak deęişikliğe uğramış aę eğitilmiştir. Benzer şekilde daha hızlı RCNN ResNet50_eksi6 modelinde de ilk olarak önceden eğitilmiş ResNet50 aęı daha hızlı RCNN algoritmasına uygun olarak güncellenmiş, ardından 6 konvolüsyon katmanı atılmış ve son olarak deęişikliğe uğramış aę eğitilmiştir. Bu son iki deneyde görölmüştür ki önceden eğitilmiş bir aęı bir algoritmaya uyarlayıp birkaç katmanın eklenmesi veya çıkarılması işlemlerinden sonra eğitilmesi, önceden eğitilmiş bir aęın bir algoritmaya uyarlandıktan sonra eğitilmesine göre ağırlıkların çok daha büyük maliyetlerde optimizasyonuna ihtiyaç duyulduęu için algılama başarısında düşüřlere neden olmaktadır. Bu optimizasyon maliyetini azaltmak için bir sonraki deneyde öğrenmenin aktarılması (transfer learning) yöntemi denenmiştir. Önceki deneylerde eğitilen daha hızlı RCNN_ResNet50 modelinin conv4_x konvolüsyon bloęu üzerinden 6 konvolüsyon katmanı çıkarılarak tekrardan eğitimi sağlanmıştır. Yani daha hızlı RCNN_ResNet50 modelinin aę katsayıları eğitim için kullanılmıştır. Algılama test sonuçlarına bakıldığında (Çizelge 7.2’de gösterilmiştir) görölmüştür ki öğrenmenin aktarılması ile yapılan daha hızlı RCNN ResNet50_eksi6_tl modeli, hem daha hızlı RCNN ResNet50 modelinden hem de daha hızlı RCNN ResNet50_eksi6 modelinden başarılı bir sonuç elde etmiştir.

Yapılan deneylerde eğitim süreleri deęerlendirildiğinde, beklendięi gibi ResNet50’den daha sığ olan ResNet18 ile oluşturulan modelin eğitiminin çok daha az zaman aldıęı; ResNet50’den daha derin olan ResNet101 ile oluşturulan modelin eğitiminin çok daha fazla zaman aldıęı görölmüştür. Katmanların orijinal aęlardan eklenerek veya çıkarılarak deforme edilen modellerin eğitimlerinin ise orijinallerinden bir miktar daha fazla sürdüęü gözlenmiştir. Öğrenmenin aktarılması ile eğitilen daha hızlı RCNN ResNet50_eksi6_tl modeli ile daha hızlı RCNN ResNet50_eksi6 modeli yakın sürelerde eğitimlerini tamamlamışlardır.

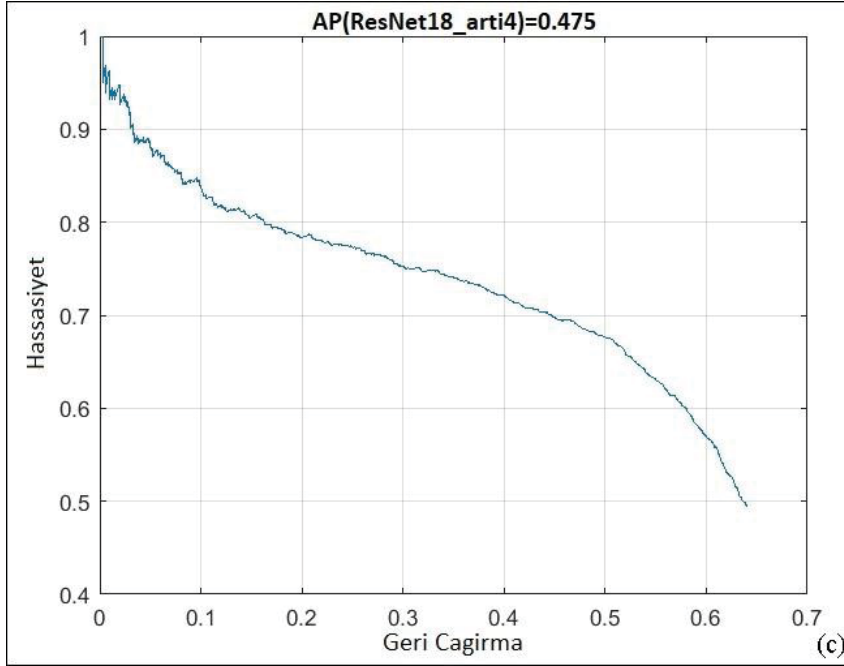
Konvolüsyon katman sayısının dolayısıyla aę derinliğinin küçük nesnelerin tespitine etkisinin araştırılması amacıyla kullanılan modellerin test aşamalarında elde edilen geri çağırma-hassasiyet grafikleri Şekil 7.3’te verilmiştir.



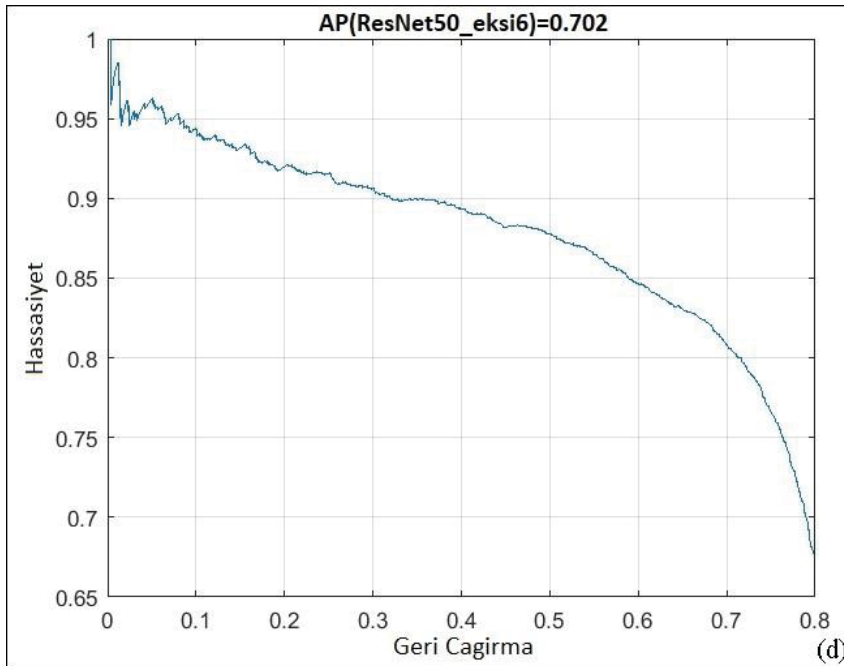
Şekil 7.3. a) Daha hızlı RCNN ResNet18 modelinin geri çağırma-hassasiyet grafiği b) Daha hızlı RCNN ResNet101 modelinin geri çağırma-hassasiyet grafiği c) Daha hızlı RCNN ResNet18_artı4 modelinin geri çağırma-hassasiyet grafiği d) Daha hızlı RCNN ResNet50_eksi6 modelinin geri çağırma-hassasiyet grafiği e) Daha hızlı RCNN ResNet50_eksi6_tl modelinin geri çağırma-hassasiyet grafiği



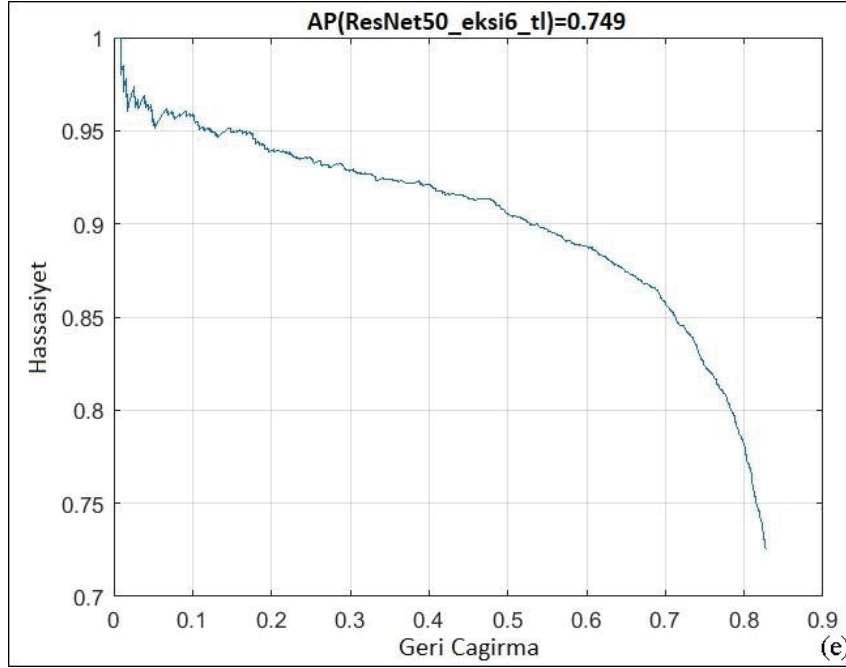
Şekil 7.3. (devam) a) Daha hızlı RCNN ResNet18 modelinin geri çağırma-hassasiyet grafiği b) Daha hızlı RCNN ResNet101 modelinin geri çağırma-hassasiyet grafiği c) Daha hızlı RCNN ResNet18_artı4 modelinin geri çağırma-hassasiyet grafiği d) Daha hızlı RCNN ResNet50_eksi6 modelinin geri çağırma-hassasiyet grafiği e) Daha hızlı RCNN ResNet50_eksi6_tl modelinin geri çağırma-hassasiyet grafiği



Şekil 7.3. (devam) a) Daha hızlı RCNN ResNet18 modelinin geri çağırma-hassasiyet grafiği b) Daha hızlı RCNN ResNet101 modelinin geri çağırma-hassasiyet grafiği c) Daha hızlı RCNN ResNet18_arti4 modelinin geri çağırma-hassasiyet grafiği d) Daha hızlı RCNN ResNet50_eksi6 modelinin geri çağırma-hassasiyet grafiği e) Daha hızlı RCNN ResNet50_eksi6_tl modelinin geri çağırma-hassasiyet grafiği

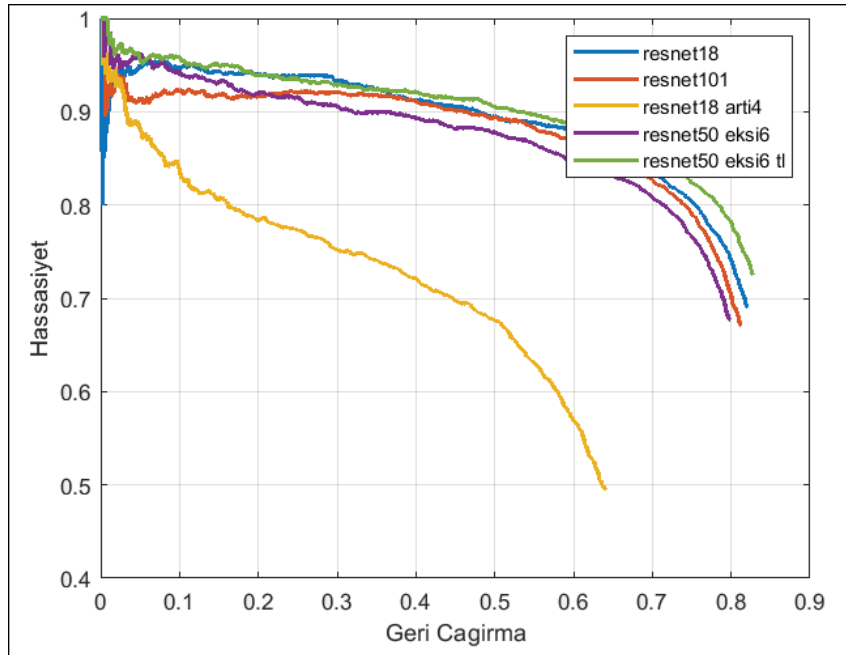


Şekil 7.3. (devam) a) Daha hızlı RCNN ResNet18 modelinin geri çağırma-hassasiyet grafiği b) Daha hızlı RCNN ResNet101 modelinin geri çağırma-hassasiyet grafiği c) Daha hızlı RCNN ResNet18_arti4 modelinin geri çağırma-hassasiyet grafiği d) Daha hızlı RCNN ResNet50_eksi6 modelinin geri çağırma-hassasiyet grafiği e) Daha hızlı RCNN ResNet50_eksi6_tl modelinin geri çağırma-hassasiyet grafiği



Şekil 7.3. (devam) a) Daha hızlı RCNN ResNet18 modelinin geri çağırma-hassasiyet grafiği b) Daha hızlı RCNN ResNet101 modelinin geri çağırma-hassasiyet grafiği c) Daha hızlı RCNN ResNet18_artı4 modelinin geri çağırma-hassasiyet grafiği d) Daha hızlı RCNN ResNet50_eksi6 modelinin geri çağırma-hassasiyet grafiği e) Daha hızlı RCNN ResNet50_eksi6_tl modelinin geri çağırma-hassasiyet grafiği

Karşılaştırmasının daha iyi yapılabilmesi için Şekil 7.4'te bu grafikler üst üste çizdirilmiştir.



Şekil 7.4. Farklı konvolüsyon katman sayısına sahip modellerin R-P grafiği

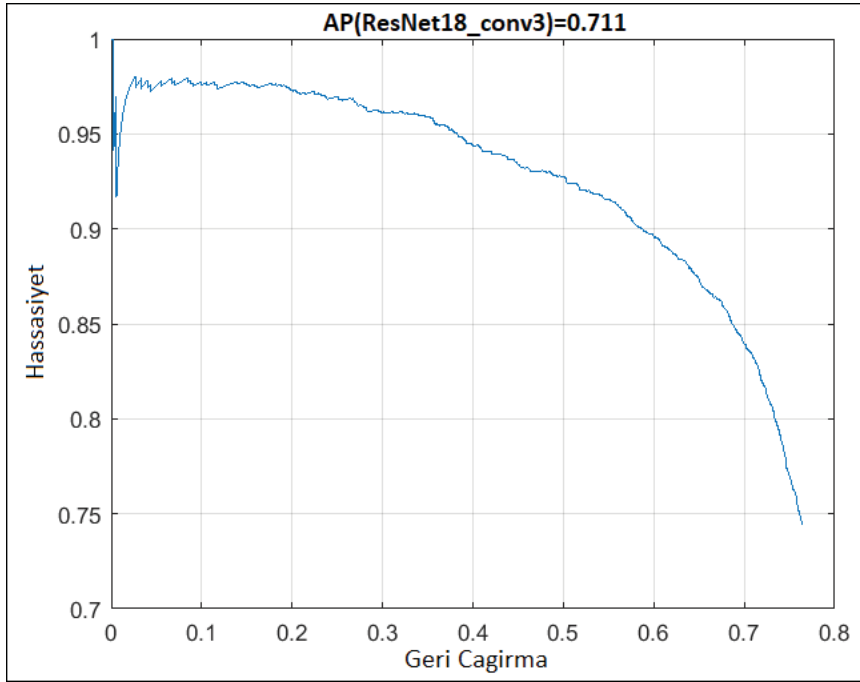
Grafiklere göre; algılama başarısı diğer modellere göre hem ortalama hassasiyet (AP) hem de F1 skoru açısından en yüksek olan daha hızlı RCNN ResNet50_eksi6_tl modeli, bir başarı kriteri olarak, grafikte üst yatay ve yan sağ dikey eksenlere en yakın seyreden model olduğu gözlenmiştir. Bu açıdan daha hızlı RCNN ResNet18_artı4 modelinin eksenlere en uzak seyreden grafiğiyle en başarısız model olduğu gözlenmiştir. Grafikler yüksek hassasiyet bakımından değerlendirildiğinde; daha hızlı RCNN ResNet18 ve daha hızlı RCNN ResNet101 modellerinin düşük geri çağırma diğer modellere göre görece düşük hassasiyete sahip olduğu gözlenmiştir. Yüksek geri çağırma açısından grafikler değerlendirildiğinde; daha hızlı RCNN ResNet50_eksi6_tl modelinin en iyi olduğu, ardından sırasıyla daha hızlı RCNN ResNet18, daha hızlı RCNN ResNet101, daha hızlı RCNN ResNet50_eksi6 modellerinin başarılı olduğu ve en kötü modelin daha hızlı RCNN ResNet18_artı4 modelinin olduğu gözlenmiştir. Çizelge 7.2’de modeller ortalama hassasiyet (AP) ve F1 skoru açısından değerlendirildiğinde; her iki metrik için de başarı sırasının sırasıyla daha hızlı RCNN ResNet50_eksi6_tl, daha hızlı RCNN ResNet18, daha hızlı RCNN ResNet101, hızlı RCNN ResNet50_eksi6 ve hızlı RCNN ResNet18_artı4 şeklinde olduğu gözlenmiştir.

7.3. Özellik Haritası Boyutunun Etkisi

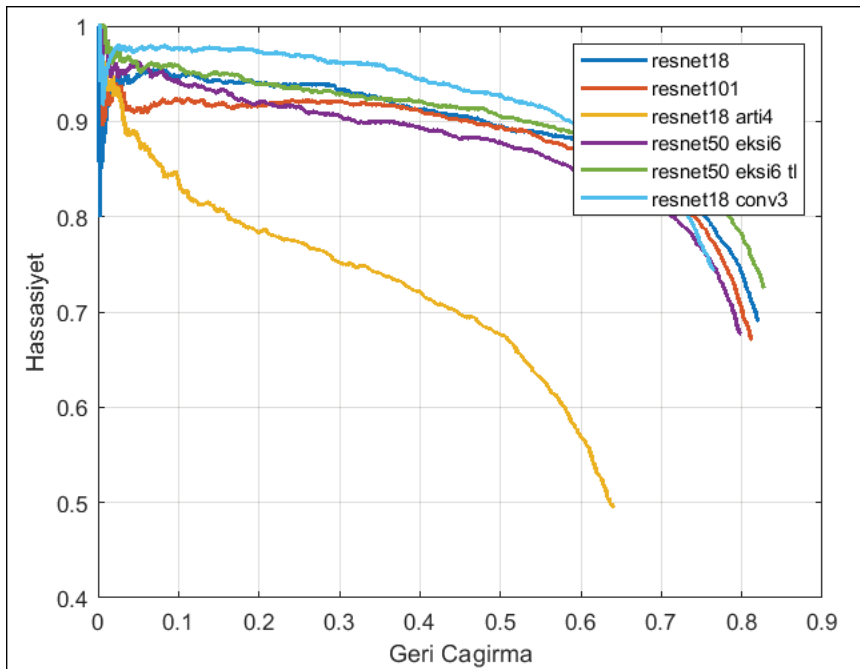
Bu bölümde, küçük nesnelerin özellik haritasında temsil ettiği nokta sayısını artırmak amacıyla özellik haritası boyutunun artırılmasının, algılama başarısına etkisi araştırılmıştır. Özellik haritası boyutunun artırılması için, nesnenin özellik haritasında temsil edildiği nokta sayısını da artıracak şekilde; ResNet ağının özellik çıkarması ve daha hızlı RCNN algoritmasının bölge önerileri ağına beslenmesi için standart olarak kullanılan conv4_x konvolüsyon bloğunun [14,14] çıktı boyutundaki son katmanı yerine, conv3_x konvolüsyon bloğunun [28,28] çıktı boyutundaki son katmanı seçilmiştir. Ağ olarak, Bölüm 7.2’de ResNet50 ve ResNet101’den daha yüksek bir algılama başarısı sağladığı için ResNet18 ağı kullanılmıştır. Bölge önerileri ağının çapa kutu olarak, Bölüm 7.1’de en yüksek başarıya sahip Ankor4 modelinde kullanılan 4 çapa kutusu kullanılmıştır. Bu kapsamda hazırlanan daha hızlı RCNN ResNet18_conv3 modeli eğitilmiştir. Modelin test aşamasında hesaplanan test parametreleri Çizelge 7.3’te ve geri çağırma-hassasiyet grafiği Şekil 7.5’te verilmiştir. Karşılaştırmasının daha iyi yapılabilmesi için bu grafiğin Bölüm 7.2’deki modellere ait grafiklerle birlikte çizdirildiği geri çağırma- hassasiyet grafiği Şekil 7.6’da verilmiştir.

Çizelge 7.3. Daha hızlı RCNN ResNet18_conv3 modeli test sonuçları

Model	R	P	AP	F1	Eğitim Süresi
Daha hızlı RCNN ResNet18_conv3	0,765	0,743	0,711	0,754	78,5



Şekil 7.5. Daha hızlı RCNN ResNet18_conv3 modelinin geri çağırma – hassasiyet grafiği



Şekil 7.6. Farklı modellere ait geri çağırma – hassasiyet grafiği

Şekil 7.6’da, daha hızlı RCNN ResNet18_conv3 modelinin yüksek hassasiyet açısından değerlendirildiğinde, Bölüm 7.2’deki tüm modellerden daha iyi bir sonuç gösterdiği gözlenmiştir. Ancak yüksek geri çağırma açısından değerlendirildiğinde; modelin, Bölüm 7.2’deki modellerden daha hızlı RCNN ResNet18_artı4 modeli hariç diğer tüm modellerden daha başarısız olduğu gözlenmiştir. Çizelge 7.3’te; ortalama hassasiyet (AP) değeri açısından daha hızlı RCNN ResNet101, daha hızlı RCNN ResNet18 ve daha hızlı RCNN ResNet50_eksi6_tl modellerinin gerisinde kalırken, F1 skoru açısından yalnızca daha hızlı RCNN ResNet50_eksi6_tl modelinin gerisinde kaldığı diğer modellerden daha başarılı olduğu gözlenmiştir. Bu modelin test sonuçlarında; yanlış pozitiflerin fazla, yanlış negatiflerin ise az olduğunu göstermektedir. Yani modelin gerçekte araç olmayan (arka plan olan) yerleri araç olarak gösterme oranı görece fazla ve görüntüdeki aracı tespit edememe oranı görece azdır. Çizelge 7.3’teki bir diğer gözlem ise, eğitim süresinin bu ağın ağ derinliği bakımından muadili daha hızlı RCNNResNet18 modelinden çok uzun olmasıdır. Bunun sebebinin, bölge önerileri ağının giriş boyutunun artmasından dolayı işlem yükünün artması olduğu değerlendirilmiştir.

7.4. Sonuçların Karşılaştırılması

Yapılan deneysel çalışmalarda en yüksek başarıyı daha hızlı RCNN ResNet50_eksi6_tl modeli sağlamıştır. Bu model ile elde edilen ortalama hassasiyet değeri, [42] ve [43]’teki sonuçlar ile karşılaştırılmıştır. [42]’de VEDAI veri setinden tekne, traktör ve uçak sınıfları örnek azlığı sebebi ile çıkarılmıştır. Ardından bu yazıda olduğu gibi döndürme ve ayna simetrisi alma yöntemleri ile mevcut veriler sekiz katına çıkarılmıştır. Hiçbir değişiklik yapılmadan orijinal daha hızlı RCNN algoritması ile farklı önceden eğitilmiş ağ modelleri ve kendi önerdikleri daha hızlı RCNN benzeri, bölge önerilerinin ayrı bir ağ tarafından yapıldığı ikili bir ağ yapısı içeren, modelin eğitim sonuçları Çizelge 7.4’te verilmiştir. Bu yazıda elde edilen sonuçların, hiçbir değişiklik yapılmadan kullanılan daha hızlı RCNN ve benzeri algoritmalarından çok daha başarılı olduğu görülmektedir. Bu da mevcut algoritmaların küçük nesne tespitlerinde doğrudan kullanımının yetersiz olduğu, küçük nesne tespiti için bu algoritmaların ince ayara ihtiyaç duyduğu tezini doğrular. [43]’te VEDAI veri kümesinden yalnızca üstten bakıldığında birbirine çok benzeyen, benzer boyutlara sahip araba, pikap ve panelvan araç sınıfları kullanılmış, diğer sınıflar atılmıştır. Ayrıca [43]’te veri kümesinin sağladığı görüntülerdeki araçların koordinatları, merkez noktaları korunarak yüksekliği ve genişliği 40’a 40 olacak şekilde bir değişikliğe gidilmiş,

eğitimlerde de daha hızlı RCNN'in çapa kutu boyutu 40'a 40 olarak kullanılmıştır. Farklı olarak, VGG-16 ağı kullanılmış ve veri artırımı yapılmamıştır. RPN'e bölge önerilerini üretmesi için giriş olarak beslenen özellik haritasının oluştuğu katmanlar, ağın farklı derinliklerinden seçilmiş ve yapılan deney sonuçları Çizelge 7.4'te verilmiştir. Sonuçların, bu yazıda elde edilen algılama başarılarından daha başarılı olduğu görülmektedir. Bu yazıda eğitimlerde birbirinden çok farklı görünen uçak, iş makinesi, traktör, tır gibi dokuz farklı araç sınıfı kullanılmıştır. [43]'te ise birbirine her açıdan çok benzeyen üç farklı araç sınıfı kullanılmıştır. Bunun öğrenmeyi kolaylaştırdığı ve algılama başarısını artırdığı değerlendirilmektedir.



Resim 7.1. Daha hızlı RCNN ResNet50_eksi6_t1 modeline ait algılama sonuçları

Çizelge 7.4. Farklı çalışmaların karşılaştırılması

Model	AP
Daha Hızlı R-CNN (Z&F)[42]	0,308
Daha Hızlı R-CNN (VGG-16)[42]	0,421
Önerilen Model(VGG-16)[42]	0,546
Daha Hızlı RCNN ResNet50 _eksi6 _tl	0,749
Daha Hızlı RCNN(VGG-16) conv3 _3[43]	0,972
Daha Hızlı RCNN(VGG-16) conv4 _3[43]	0,974
Daha Hızlı RCNN(VGG-16) conv5 _3[43]	0,967

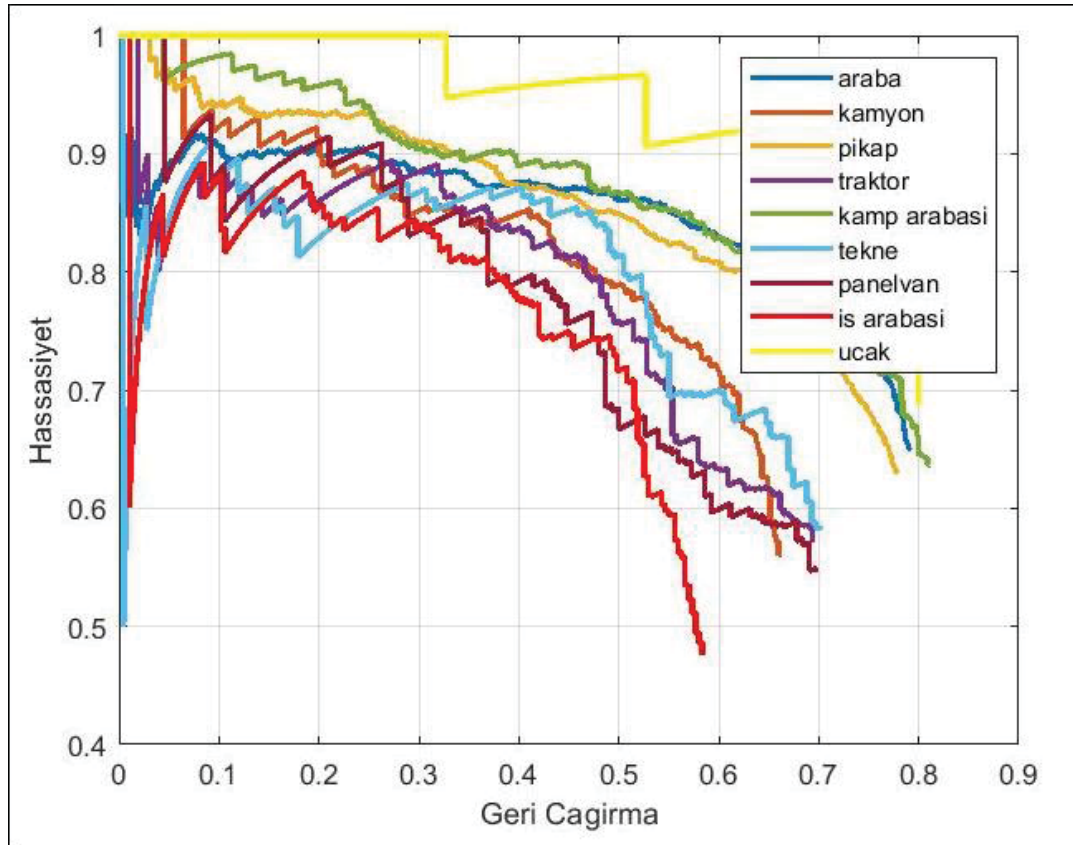
8. ARAÇLARIN SINIFLANDIRILMASI

Bu bölümde; araçların görüntüler üzerinden otomatik olarak uçak, araba, kamyon, panelvan, pikap, kamp arabası, traktör, gemi ve iş makinesi şeklinde sınıflandırılması çalışması yapılmıştır. Yapılan çalışmaların Bölüm 7’den temel farklı, sınıflandırma probleminin sınıf sayısının fazlalığıdır. Yani; Bölüm 7’de araç ve arka plan olmak üzere 2 sınıflı bir sınıflandırma ve sınırlayıcı kutular ile konum tespit edilmesi gerektiği için bir regresyon problemleri üzerinde, bu bölümde ise arka plan ile birlikte 10 sınıflı bir sınıflandırma ve yine sınırlayıcı kutu koordinatları için regresyon problemleri üzerinde çalışılmıştır. Bu kapsamda çalışmada Bölüm 7’de araçların otomatik tespit edilmesi deneysel çalışmalarında edinilen kazanımlar korunarak en yüksek algılama başarısına sahip daha hızlı RCNN ResNet50_eksi6_tl modeli temel alınmıştır. Daha hızlı RCNN ResNet50_eksi6_tl modelinin başarısında öğrenmenin aktarılması yöntemi etkili olmuş, ilk olarak ResNet50 ağına başarısı kanıtlanmış 4 çapa kutusu ile birlikte daha hızlı RCNN algoritması uygulanmış model eğitilmiş, ardından ağ eğitilmiş modelden conv4_x bloğunun son 2 artık bloğu (6 konvolüsyon katmanı) çıkartılarak tekrardan eğitilmiştir. Bu yöntemden yola çıkılarak burada da ilk olarak ResNet50 ağına daha hızlı RCNN algoritması uygulanmıştır. Daha hızlı RCNN algoritmasının bir parçası olan bölge önerileri ağına çapa kutusu olarak, yine Bölüm 7.1’de en yüksek algılama başarısına sahip Ankor4 modelinin 4 çapa kutusu kullanılmıştır. Farklı olarak; çıktı katmanları 10 sınıflı sınıflandırma problemine uygun hale getirilmiştir. Bunun için; orijinal ResNet50 ağının “fc1000” isimli tam bağlantılı, “fc1000_softmax” isimli softmaks ve “ClassificationLayer_fc1000” isimli sınıflandırma katmanları ağdan atılmıştır. Ardından 10 sınıflı sınıflandırma problemine uygun olarak 10 çıktıya sahip tam bağlantılı katman, bu katmanı takip eden softmaks ve sınıflandırma katmanları tanımlanmış ve “avg_pool” isimli ortalama havuzlama katmanının çıktısı olarak ağa bağlanmıştır. Ardından sınırlayıcı kutuların tespiti için; arka plan gerçek bir nesne sınıfı olmadığı için sınıf sayısı 9 alınarak 36 çıktıya sahip ($4 \times$ sınıf sayısından dolayı) tam bağlantılı katman, bu katmanı takip eden regresyon katmanları tanımlanmış ve ortalama havuzlama katmanının çıktısı olarak ağa bağlanmıştır. Bu modele daha hızlı RCNN ResNet50_classification ismi verilmiştir. Modelin eğitimi Çizelge 7.1’deki eğitim parametrelerine göre yapılmış ve her bir sınıf için Bölüm 7’de anlatılan test metrikleri hesaplanmış, geri çağırma – hassasiyet grafikleri üst üste çizdirilmiştir. Çizelge 8.1; daha hızlı RCNN ResNet50_classification

modelinin her bir sınıf için test sonuçlarını göstermektedir. Şekil 8.1 ise, her bir sınıf için üst üste bindirilmiş geri çağırma – hassasiyet grafiğini göstermektedir.

Çizelge 8.1. Daha hızlı RCNN ResNet50_classification modeline ait algıma sonuçları

Sınıf	R	P	AP	F1
Araba	0,792	0,649	0,676	0,713
Kamyon	0,662	0,561	0,559	0,607
Pikap	0,778	0,630	0,674	0,697
Traktör	0,694	0,571	0,557	0,627
Kamp arabası	0,811	0,637	0,712	0,713
Tekne	0,702	0,582	0,568	0,636
Panelvan	0,697	0,546	0,553	0,613
İş arabası	0,584	0,475	0,463	0,524
Uçak	0,800	0,687	0,762	0,740



Şekil 8.1. Daha hızlı RCNN ResNet50_classification modelinin her bir sınıfa ait R-P grafiği

Şekil 8.1'deki grafiğe göre; yüksek hassasiyet açısından bakıldığında, modelin uçak, kamp arabası, kamyon ve pikap sınıflarını algılamada iyi performans göstererek bu sınıfların

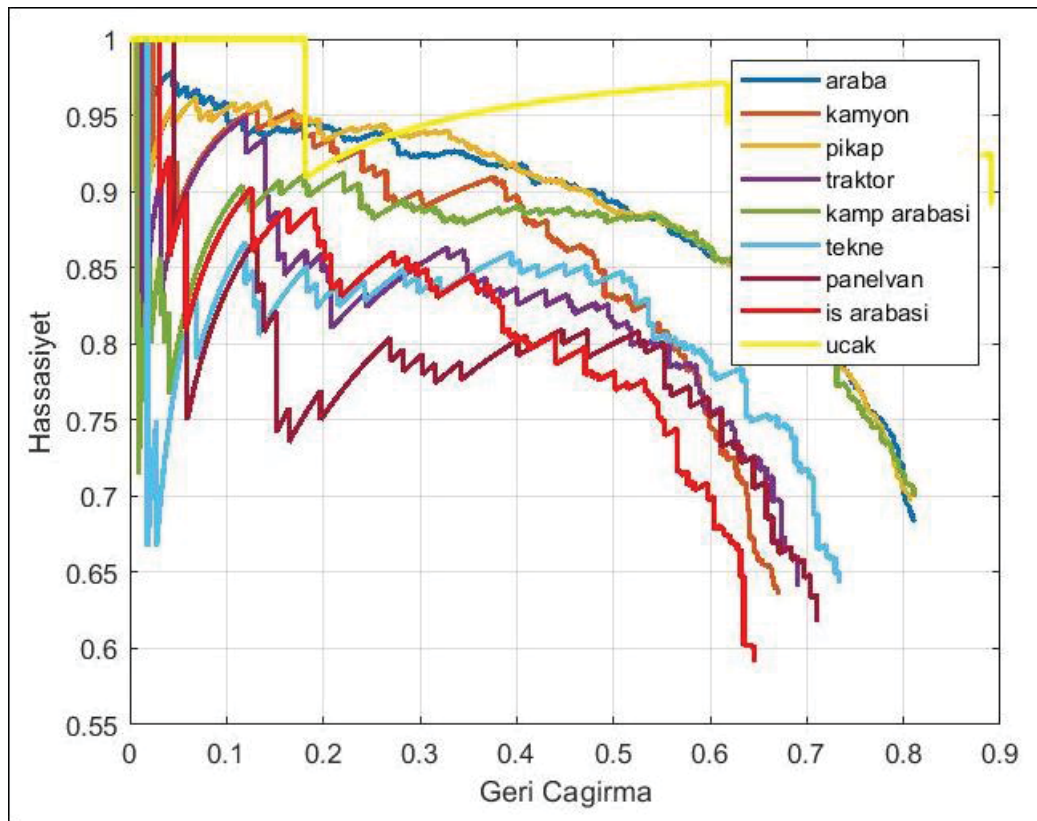
ayrıştığı gözlenmiştir. Yani; modelin bu sınıfları algılamadaki yanlış pozitif üretme oranı diğer sınıflara göre daha küçüktür. Bu da; bu sınıfların diğer sınıflarla karıştırılmasının, bir araç için birden fazla sınırlayıcı kutu çizdirilmesinin ve görüntüdeki araç olmayan arka planlarda araç tespit edilme oranlarının daha küçük olduğunu göstermektedir. Diğer sınıflar için ise; özellikle tekne ve iş arabası sınıfları için, bu oranların daha yüksek olduğu gözlenmiştir. Grafik yüksek geri çağırma açısından değerlendirildiğinde; uçak, kamp arabası, araba ve pikap sınıflarının model tarafından algılama başarısının daha iyi olduğu gözlenmektedir. Yani; model bu sınıflar için daha az yanlış negatif üretmekte, dolayısıyla modelin görüntü üzerindeki bu sınıflara ait araçları tespit edemeyip arka plan olarak değerlendirme oranının düşük olduğu gözlenmiştir. Bu sınıflar dışında kalan sınıflar için, özellikle de iş arabası sınıfı için, model yüksek oranda yanlış negatif üretmiştir. Grafiğin genelinden modelin; uçak, kamp arabası, araba ve pikap sınıflarını algılamada başarılı olduğu, iş arabası sınıfını algılamada başarısız olduğu ve diğer sınıflar için ortalama bir başarı gösterdiği gözlenmiştir. Çizelge 8.1'deki ortalama hassasiyet (AP) ve F1 değerleri de bu gözlemi teyit etmiştir. Modelin; araba, pikap ve kamp arabasını algılama başarısında veri kümesinin bu sınıflar için daha fazla örnek içermesi etkili olmuştur. En düşük algılanma başarısına sahip sınıfın iş makinesi olmasında ise, sınıfın bazı araçlarının traktör sınıfı ile benzerliği ve daha önemlisi kendi içerisinde çok farklı görünümde araçları barındırması etkili olmuştur. Burada uçak sınıfı veri kümesi içerisinde en az örneğe sahip sınıf olmasına rağmen, modelin bu sınıfı algılamadaki başarısı yüksek olmuştur. Uçak sınıfının diğer sınıflara göre görüntüde daha fazla alan kaplaması ve bu sınıfın diğer sınıflara hiç benzememesi, modelin bu sınıfı öğrenmesinde faydalı olduğu; dolayısıyla modelin algılanma başarısının yüksek olduğu değerlendirilmektedir.

Modelin sınıfları algılama başarısını artırmak için; Bölüm 7'de araçların tespitinde edilen kazanımlar baz alınarak, ağ daha sık hale getirilmiş ancak bu öğrenmenin aktarılması (transfer learning) yöntemi ile yapılmıştır. Yani ağı sık hale getirilmesi; daha önceden VEDAI veri kümesi ile eğitilen daha hızlı RCNN ResNet50_classification modeli kullanılarak, dolayısıyla bu eğitilmiş bu modelin ağ katsayıları eğitim için kullanılarak yapılmıştır. Bunun için, daha önce eğitilen daha hızlı RCNN_ResNet50_classification modelinin conv4_x konvolüsyon bloğunun son iki artık bloğu (6 konvolüsyon katmanı) çıkarılarak tekrardan eğitimi sağlanmıştır. Bu model daha hızlı RCNN ResNet50_eksi6_tl_classification olarak isimlendirilmiştir. Çizelge 8.2; daha hızlı RCNN ResNet50_eksi6_tl_classification modelinin her bir sınıf için test sonuçlarını

göstermektedir. Şekil 8.2 ise, her bir sınıf için üst üste bindirilmiş geri çağırma – hassasiyet grafiğini göstermektedir.

Çizelge 8.2. Daha hızlı RCNN ResNet50_eksi6_tl_classification modeline ait algıma sonuçları

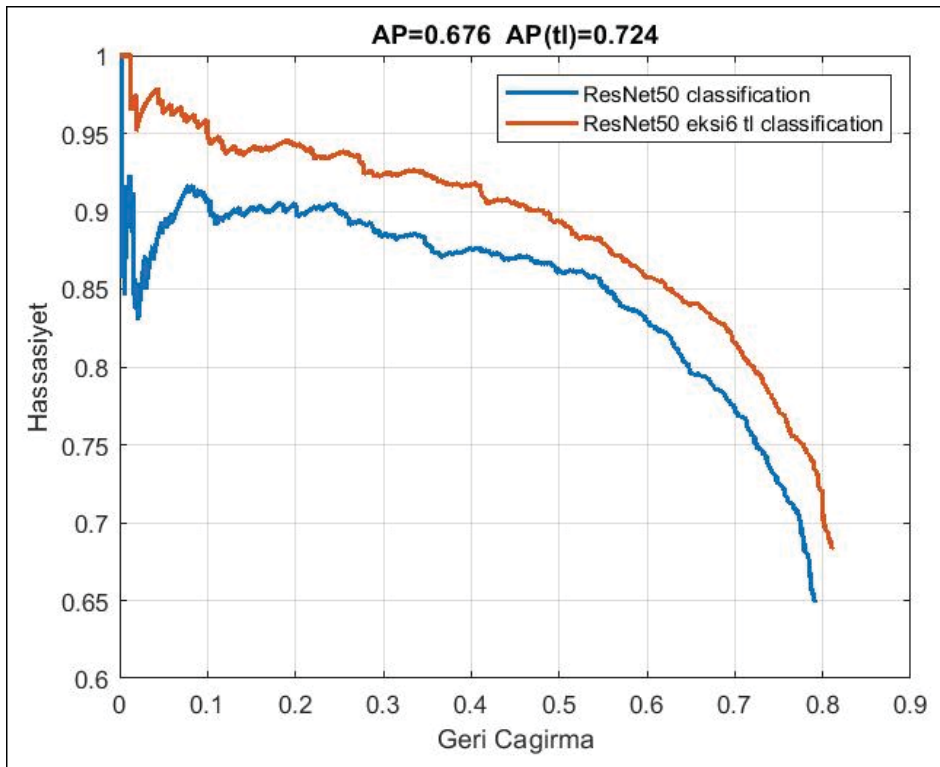
Sınıf	R	P	AP	F1
Araba	0,811	0,683	0,724	0,742
Kamyon	0,670	0,636	0,586	0,652
Pikap	0,807	0,697	0,720	0,748
Traktör	0,691	0,641	0,576	0,665
Kamp arabası	0,812	0,701	0,699	0,753
Tekne	0,734	0,643	0,600	0,685
Panelvan	0,711	0,617	0,564	0,661
İş arabası	0,645	0,591	0,530	0,617
Uçak	0,891	0,891	0,857	0,891



Şekil 8.2. Daha hızlı RCNN ResNet50_eksi6_tl_classification modelinin her bir sınıfa ait R-P grafiği

Çizelge 8.2 ve Şekil 8.2’den; daha hızlı RCNN ResNet50_eksi6_tl_classification modelinin algılama başarısının tüm sınıflar üzerinde bir önceki modele göre arttığı gözlenmektedir.

Şekil 8.3 daha hızlı RCNN ResNet50 classification ve daha hızlı RCNN ResNet50_eksi6_tl classification modellerinin araba sınıfı için geri çağırma – hassasiyet grafiklerini göstermektedir. Grafığe göre hem yüksek hassasiyet açısından bakıldığında hem de yüksek geri çağırma açısından bakıldığında, daha hızlı RCNN ResNet50_eksi6_tl classification modelinde araba sınıfı için daha hızlı RCNN ResNet50 classification modeline göre bir iyileşme gözlenmiştir. Bu durum; daha hızlı RCNN ResNet50_eksi6_tl classification modelinin araba sınıfı için testlerde hem daha az yanlış pozitif hem de daha az yanlış negatif ürettiğini göstermektedir. Yani; daha hızlı RCNN ResNet50_eksi6_tl classification modeli ile araba sınıfının tespitinde, bir araba için birden fazla sınırlayıcı kutu çizdirme, araba olmayan yerde araba tespit etme ve arabayı tespit edememe durumlarının azalması sağlanmıştır. Ortalama hassasiyet (AP) değerindeki artış, daha hızlı RCNN ResNet50_eksi6_tl classification modelinin araba algılama başarısının daha iyi olduğunu göstermektedir.



Şekil 8.3. Ağ modellerinin araba sınıfı geri çağırma-hassasiyet grafiği

Resim 8.1, daha hızlı RCNN ResNet50 classification ve daha hızlı RCNN ResNet50_eksi6_tl classification modellerinin aynı görüntü üzerinde araba tespit örneği olarak verilmiştir. Görüntüde bir araba ve bir kamyon mevcuttur. Soldaki görüntüde daha

hızlı RCNN ResNet50 classification modeli bir araba için iki sınırlayıcı kutu yani bir adet yanlış pozitif üretirken, sağdaki görüntüde daha hızlı RCNN ResNet50_eksi6_tl classification modeli hatasız bir şekilde araba ve kamyonu tespit etmiştir.

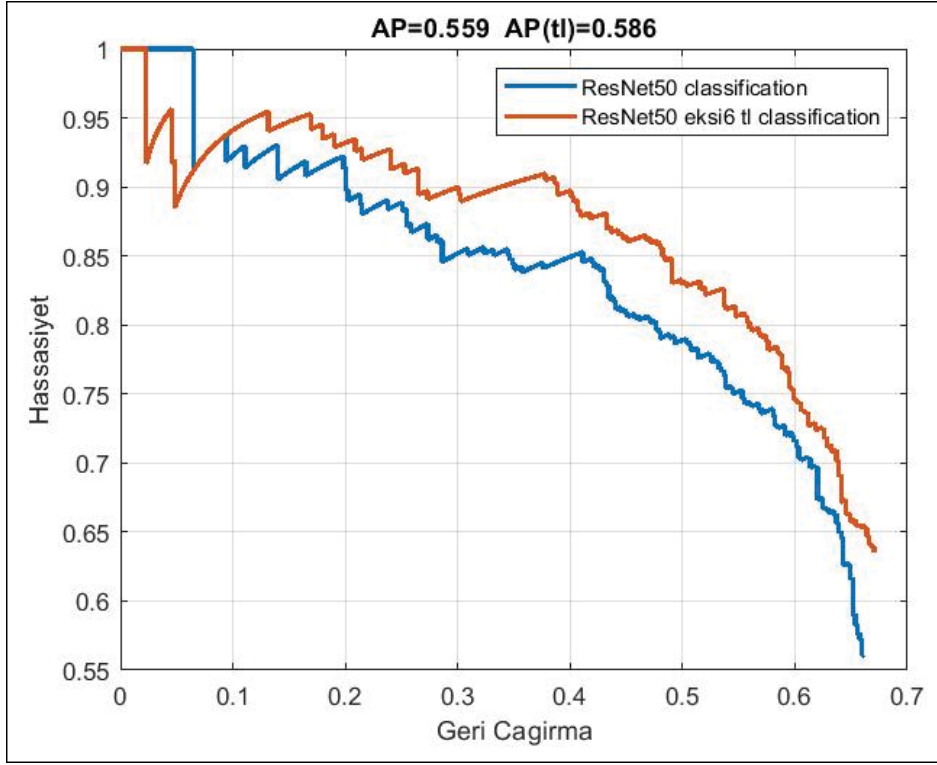


Resim 8.1. Daha hızlı RCNN ResNet50 classification modeline ait bir araba tespit örneği(sol) ve aynı görüntü üzerinde daha hızlı RCNN ResNet50_eksi6_tl classification modeline ait bir araba tespit örneği(sağ)

Şekil 8.4 daha hızlı RCNN ResNet50 classification ve daha hızlı RCNN ResNet50_eksi6_tl classification modellerinin kamyon sınıfı için geri çağırma – hassasiyet grafiklerini göstermektedir. Grafiğe göre; yüksek geri çağırma açısından bakıldığında daha hızlı RCNN ResNet50_eksi6_tl classification modelinin daha başarılı olduğu, daha az yanlış negatif ürettiği gözlenmektedir. Yüksek hassasiyet açısından bakıldığında ise; daha hızlı RCNN ResNet50_eksi6_tl classification modeli grafikte başlangıçta bir miktar düşüş sergilemiş olsa da, grafiğin sol tarafındaki altında kalan alanlar kıyaslandığında modelin daha az yanlış pozitif ürettiği değerlendirilmektedir. Ortalama hassasiyet (AP) değerindeki artış, daha hızlı RCNN ResNet50_eksi6_tl classification modelinin kamyon algılama başarısının daha iyi olduğunu göstermektedir.

Resim 8.2, daha hızlı RCNN ResNet50 classification ve daha hızlı RCNN ResNet50_eksi6_tl classification modellerinin aynı görüntü üzerinde kamyon tespit örneği olarak verilmiştir. Görüntüde bir kamyon mevcuttur. Soldaki görüntüde daha hızlı RCNN ResNet50 classification modeli fazladan kamyon olmayan bir yeri kamyon olarak

işaretlerken yani yanlış pozitif üretirken, sağdaki görüntüde daha hızlı RCNN ResNet50_eksi6_tl classification modeli hatasız bir şekilde kamyonu tespit etmiştir.

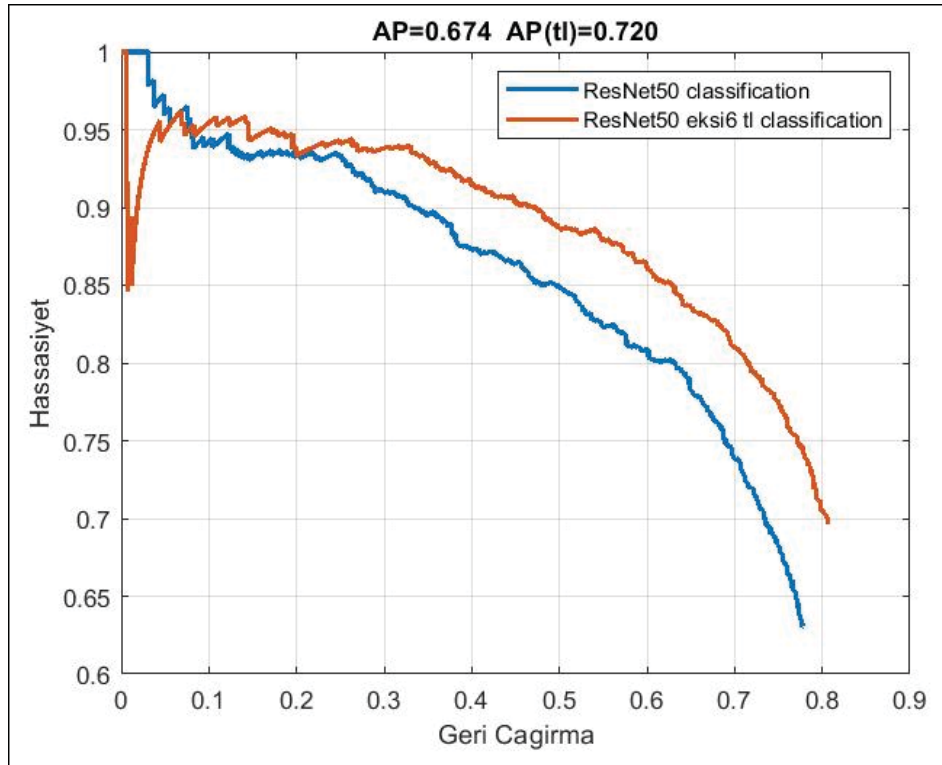


Şekil 8.4. Ağ modellerinin kamyon sınıfı geri çağırma-hassasiyet grafiği



Resim 8.2. Daha hızlı RCNN ResNet50 classification modeline ait bir kamyon tespit örneği(sol) ve aynı görüntü üzerinde daha hızlı RCNN ResNet50_eksi6_tl classification modeline ait bir kamyon tespit örneği(sağ)

Şekil 8.5 daha hızlı RCNN ResNet50 classification ve daha hızlı RCNN ResNet50_eksi6_tl classification modellerinin pikap sınıfı için geri çağırma – hassasiyet grafiklerini göstermektedir. Grafiğe göre; yüksek geri çağırma açısından bakıldığında daha hızlı RCNN ResNet50_eksi6_tl classification modelinin daha başarılı olduğu, daha az yanlış negatif ürettiği gözlenmektedir. Yüksek hassasiyet açısından bakıldığında ise; daha hızlı RCNN ResNet50_eksi6_tl classification modeli bir miktar daha hızlı RCNN ResNet50 classification modelinin gerisinde kaldığı gözlenmiştir. Ancak; ortalama hassasiyet (AP) değerindeki artış, daha hızlı RCNN ResNet50_eksi6_tl classification modelinin pikap algılama başarısının daha iyi olduğunu göstermektedir.



Şekil 8.5. Ağ modellerinin pikap sınıfı geri çağırma-hassasiyet grafiği

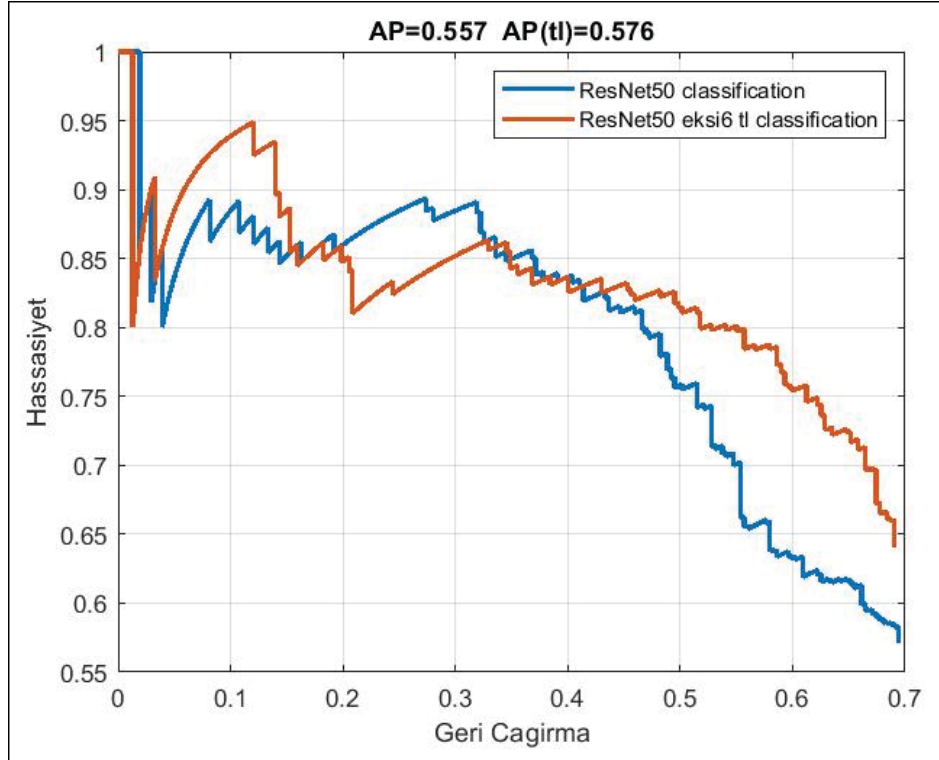
Resim 8.3, daha hızlı RCNN ResNet50 classification ve daha hızlı RCNN ResNet50_eksi6_tl classification modellerinin aynı görüntü üzerinde pikap tespit örneği olarak verilmiştir. Görüntüde beş pikap, bir traktör, bir iş makinesi ve bir kamp arabası mevcuttur. Soldaki görüntüde daha hızlı RCNN ResNet50 classification modeli tarlanın içerisindeki bir kırmızı pikabı algılayamamış yani yanlış negatif üretmiştir. Ayrıca arkasına römork bağlı pikap için hem pikap hem kamyon sınırlayıcı kutularını çizdirmiş, yani kamyon sınıfı için bir yanlış pozitif üretmiştir. Görüntünün sağındaki iş makinesi için ise

traktör sınırlayıcı kutusu çizdirmiş; yani traktör sınıfı için yanlış pozitif, iş makinesi sınıfı için yanlış negatif üretmiştir. Sağdaki görüntüde daha hızlı RCNN ResNet50_eksi6_t1 classification modeli bir önceki modelin yanlış negatif ve yanlış pozitiflerini gidererek, hatasız bir şekilde araçları tespit etmiştir.



Resim 8.3. Daha hızlı RCNN ResNet50 classification modeline ait bir pikap tespit örneği(sol) ve aynı görüntü üzerinde daha hızlı RCNN ResNet50_eksi6_t1 classification modeline ait bir pikap tespit örneği(sağ)

Şekil 8.6 daha hızlı RCNN ResNet50 classification ve daha hızlı RCNN ResNet50_eksi6_t1 classification modellerinin traktör sınıfı için geri çağırma – hassasiyet grafiklerini göstermektedir. Grafiğe yüksek hassasiyet açısından bakıldığında her iki modelinde benzer bir performans gösterdiği gözlenmiştir. Her iki model de bu sınıf için bir miktar yanlış pozitif üretme eğiliminde olmuştur. Buna, iş makinesi sınıfının bazı araçlarının traktör sınıfı ile benzerliğinin neden olduğu değerlendirilmektedir. Yüksek geri çağırma açısından bakıldığında ise; daha hızlı RCNN ResNet50_eksi6_t1 classification modelinin daha hızlı RCNN ResNet50 classification modelinden ayrılarak daha iyi performans gösterdiği gözlenmiştir. Yani modelin traktör sınıfını arka plandan ayırt ederek tespit etme durumu artmış, yanlış negatif üretme durumu azaltılmıştır. Sonuç olarak ortalama hassasiyet (AP) değerindeki artış, daha hızlı RCNN ResNet50_eksi6_t1 classification modelinin traktör algılama başarısının daha iyi olduğunu göstermektedir.



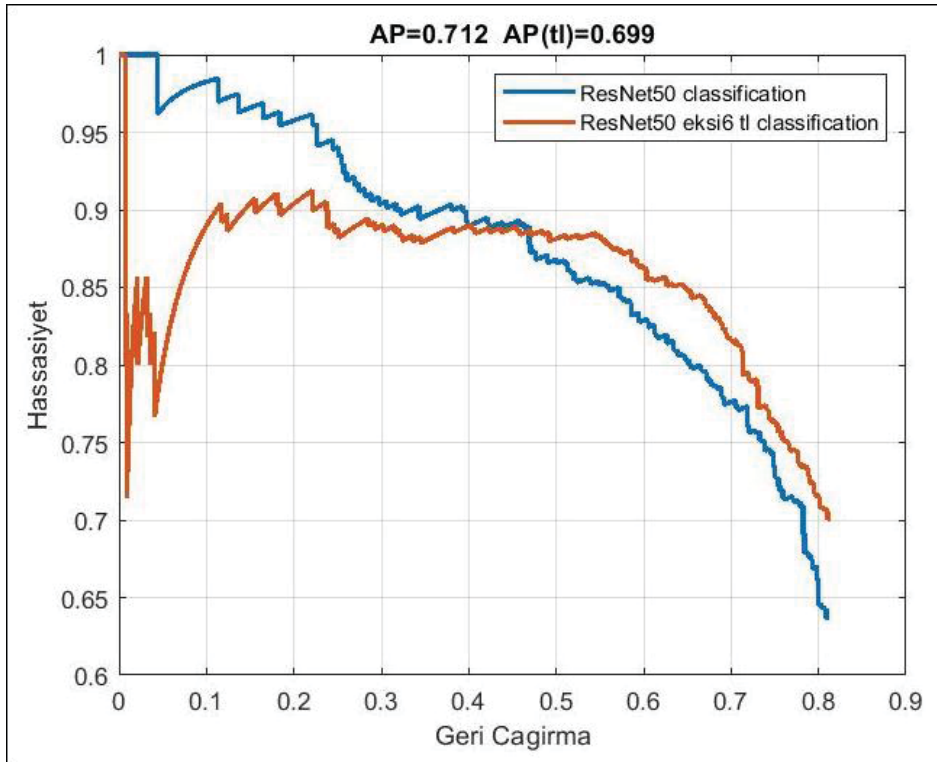
Şekil 8.6. Ağ modellerinin traktör sınıfı geri çağırma-hassasiyet grafiği

Resim 8.4, daha hızlı RCNN ResNet50 classification ve daha hızlı RCNN ResNet50_eksi6_tl classification modellerinin aynı görüntü üzerinde traktör tespit örneği olarak verilmiştir. Görüntüde üç pikap, bir traktör, iki araba ve iki kamp arabası mevcuttur. Soldaki görüntüde daha hızlı RCNN ResNet50 classification modeli sağdaki traktör için iki sınırlayıcı kutu çizdirmiş, traktör sınıfı için bir yanlış pozitif üretmiştir. Ayrıca alttaki kamp arabasını çeken pikap grubu için kamyon sınırlayıcı kutusunu oluşturarak, kamyon sınıfı için yanlış pozitif üretmiştir. Bu grubun önündeki pikap için iki sınırlayıcı kutu çizdirerek, pikap sınıfı için bir yanlış pozitif üretmiştir. Sağdaki görüntüde daha hızlı RCNN ResNet50_eksi6_tl classification modeli bir önceki modelin yanlış pozitiflerini gidererek, hatasız bir şekilde araçları tespit etmiştir.

Şekil 8.7 daha hızlı RCNN ResNet50 classification ve daha hızlı RCNN ResNet50_eksi6_tl classification modellerinin kamp arabası sınıfı için geri çağırma – hassasiyet grafiklerini göstermektedir. Grafiğe yüksek hassasiyet açısından bakıldığında, daha hızlı RCNN ResNet50_eksi6_tl classification modelinin yüksek orandaki yanlış pozitif tahminleri ile bir önceki modelin gerisine düştüğü gözlenmiştir. Buna; kamp arabası sınıfı için modelin aşırı öğrenmeye girerek, hava görüntülerinde kamp arabaları ile benzerliği yüksek olan



Resim 8.4. Daha hızlı RCNN ResNet50 classification modeline ait bir traktör tespit örneği(sol) ve aynı görüntü üzerinde daha hızlı RCNN ResNet50_eksi6_tl classification modeline ait bir traktör tespit örneği(sağ)



Şekil 8.7. Ağ modellerinin kamp arabası sınıfı geri çağırma-hassasiyet grafiği

konteynerler, elektrik üniteleri ve ev çatıları üzerinde yanlış tahminlerde bulunmasının neden olduğu değerlendirilmektedir. Yüksek geri çağırma açısından bakıldığında ise; daha

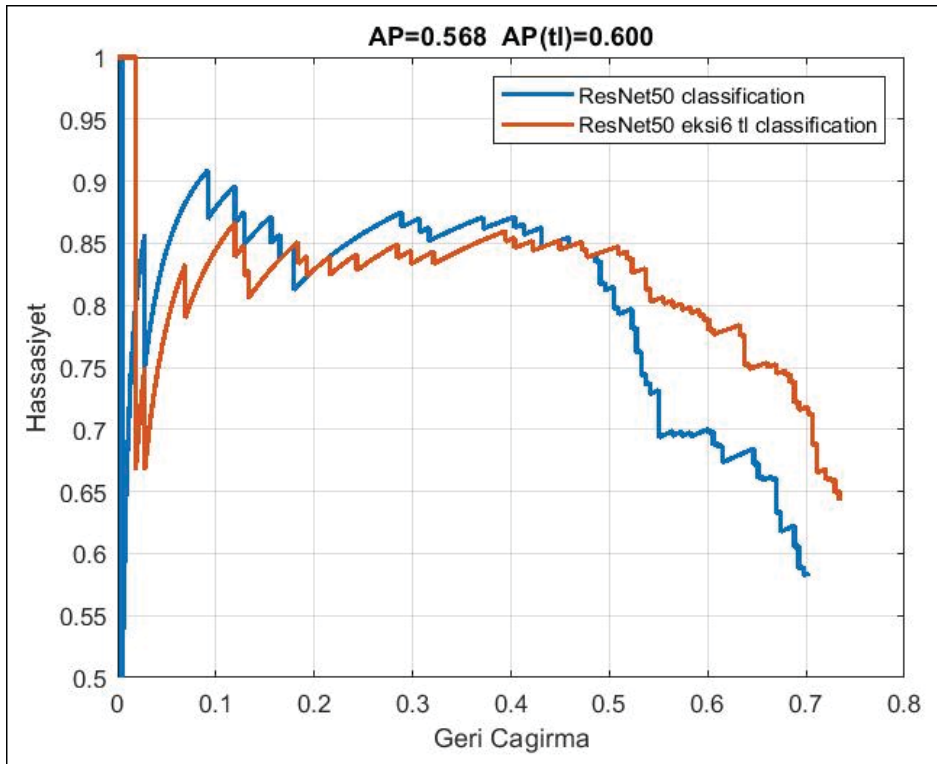
hızlı RCNN ResNet50_eksi6_tl classification modelinin daha hızlı RCNN ResNet50 classification modelinden ayrılarak daha iyi performans gösterdiği gözlenmiştir. Ancak yüksek hassasiyet açısından bakıldığındaki grafikteki düşüş; ortalama hassasiyet (AP) değerinde daha hızlı RCNN ResNet50_eksi6_tl classification modelinin kamp arabası için daha hızlı RCNN ResNet50 classification modeline göre düşüşüne neden olmuştur.

Resim 8.5, daha hızlı RCNN ResNet50 classification ve daha hızlı RCNN ResNet50_eksi6_tl classification modellerinin aynı görüntü üzerinde kamp arabası tespit örneği olarak verilmiştir. Görüntüde beş pikap, altı araba ve altı kamp arabası mevcuttur. Soldaki görüntüde daha hızlı RCNN ResNet50 classification modeli sol alttaki kamp arabasını tespit edememiş, kamp arabası sınıfı için yanlış negatif üretmiştir. Ayrıca; görüntünün sağındaki kamp arabası ve pikap gurubu için kamyon sınırlayıcı kutusu oluşturularak kamyon sınıfı için yanlış pozitif üretmiştir. Görüntünün ortasındaki iki araba için ikişer adet sınırlayıcı kutu oluşturmuş, araba sınıfı için iki yanlış pozitif üretmiştir. Sağdaki görüntüde daha hızlı RCNN ResNet50_eksi6_tl classification modeli bir önceki modelin tespit edemediği kamp arabasını tespit etmiş ancak araba ve iki pikabın yanındaki kamp arabasını tespit edememiştir. Ayrıca; sağ üstteki pikabı tespit edemeyerek pikap sınıfı için bir yanlış negatif ve üstteki araba için iki sınırlayıcı kutu oluşturularak araba sınıfı için bir yanlış pozitif üretmiştir.



Resim 8.5. Daha hızlı RCNN ResNet50 classification modeline ait bir kamp arabası tespit örneği(sol) ve aynı görüntü üzerinde daha hızlı RCNN ResNet50_eksi6_tl classification modeline ait bir kamp arabası tespit örneği(sağ)

Şekil 8.8 daha hızlı RCNN ResNet50 classification ve daha hızlı RCNN ResNet50_eksi6_tl classification modellerinin tekne sınıfı için geri çağırma – hassasiyet grafiklerini göstermektedir. Grafiğe yüksek hassasiyet açısından bakıldığında, her iki modelin de bu bakımdan başarısız olduğu ve modellerin yüksek yanlış pozitif üretme eğiliminde olduğu görülmektedir. Bu durumun veri kümesindeki tekne sınıfı örneklerinin azlığından kaynaklandığı değerlendirilmektedir. Ancak yine de daha hızlı RCNN ResNet50_eksi6_tl classification modelinin bir iyileşme sağladığı, yanlış pozitif üretme durumunun azaldığı gözlenmiştir. Grafiğe yüksek geri çağırma açısından bakıldığında ise; daha hızlı RCNN ResNet50_eksi6_tl classification modelinin daha hızlı RCNN ResNet50 classification modelinden ayrılarak daha iyi performans gösterdiği gözlenmiştir. Sonuç olarak ortalama hassasiyet (AP) değerindeki artış, daha hızlı RCNN ResNet50_eksi6_tl classification modelinin tekne algılama başarısının daha iyi olduğunu göstermektedir.



Şekil 8.8. Ağ modellerinin tekne sınıfı geri çağırma-hassasiyet grafiği

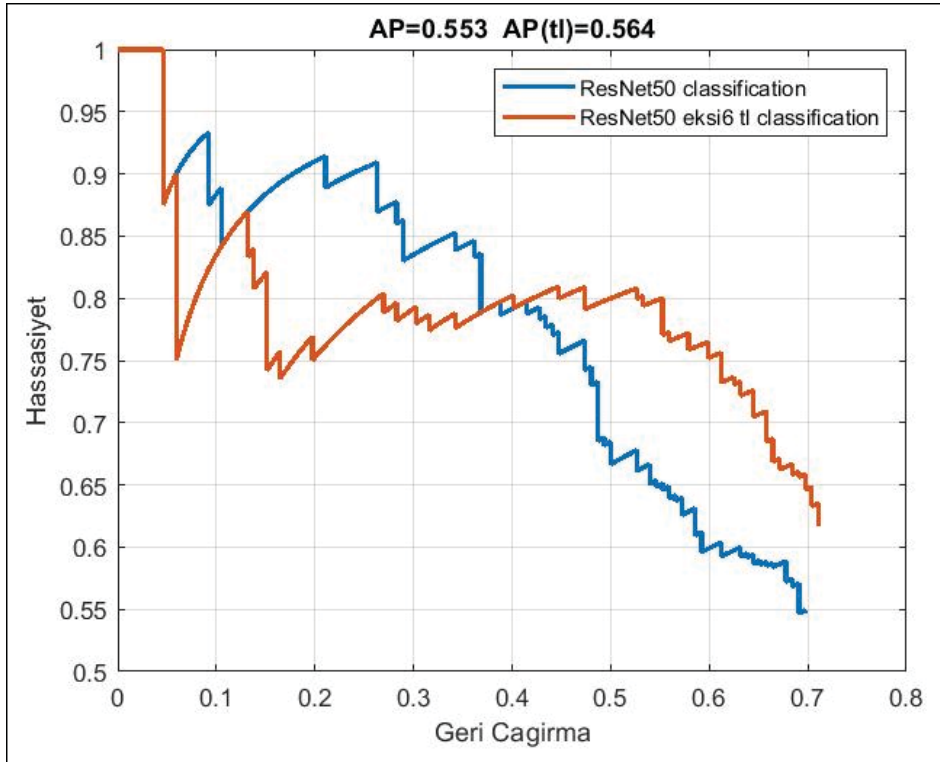
Resim 8.6, daha hızlı RCNN ResNet50 classification ve daha hızlı RCNN ResNet50_eksi6_tl classification modellerinin aynı görüntü üzerinde tekne tespit örneği olarak verilmiştir. Görüntüde beş adet tekne sınıfı araç mevcuttur. Soldaki görüntüde daha hızlı RCNN ResNet50 classification modeli en üstteki tekneyi algılayamayarak, bir yanlış

negatif üretmiştir. Sağdaki görüntüde daha hızlı RCNN ResNet50_eksi6_tl classification modeli bir önceki modelin yanlış negatifini gidererek, hatasız bir şekilde tekneleri tespit etmiştir.



Resim 8.6. Daha hızlı RCNN ResNet50 classification modeline ait bir tekne tespit örneği(sol) ve aynı görüntü üzerinde daha hızlı RCNN ResNet50_eksi6_tl classification modeline ait bir tekne tespit örneği(sağ)

Şekil 8.9 daha hızlı RCNN ResNet50 classification ve daha hızlı RCNN ResNet50_eksi6_tl classification modellerinin panelvan sınıfı için geri çağırma – hassasiyet grafiklerini göstermektedir. Grafiğe yüksek hassasiyet açısından bakıldığında; başlangıçta her iki model de aynı çizgide devam ederken, sonradan daha hızlı RCNN ResNet50_eksi6_tl classification modeli daha hızlı RCNN ResNet50 classification modelinin gerinde kaldığı gözlenmiştir. Yani daha hızlı RCNN ResNet50_eksi6_tl classification modeli, daha hızlı RCNN ResNet50 classification modeline göre daha fazla yanlış pozitif üretmiştir. Grafiğe yüksek geri çağırma açısından bakıldığında ise; daha hızlı RCNN ResNet50_eksi6_tl classification modelinin daha hızlı RCNN ResNet50 classification modelinden ayrılarak daha iyi performans gösterdiği gözlenmiştir. Dolayısıyla panelvan sınıfı araç algıma için; yüksek hassasiyet açısından daha önce eğitilen daha hızlı RCNN ResNet50 classification modeli daha başarılıyken, yüksek geri çağırma açısından daha hızlı RCNN ResNet50_eksi6_tl classification modeli daha başarılı olmuştur. Modeller genel olarak kıyaslandığında; ortalama hassasiyet (AP) değerindeki artış, daha hızlı RCNN ResNet50_eksi6_tl classification modelinin panelvan algılama başarısının daha iyi olduğunu göstermektedir.



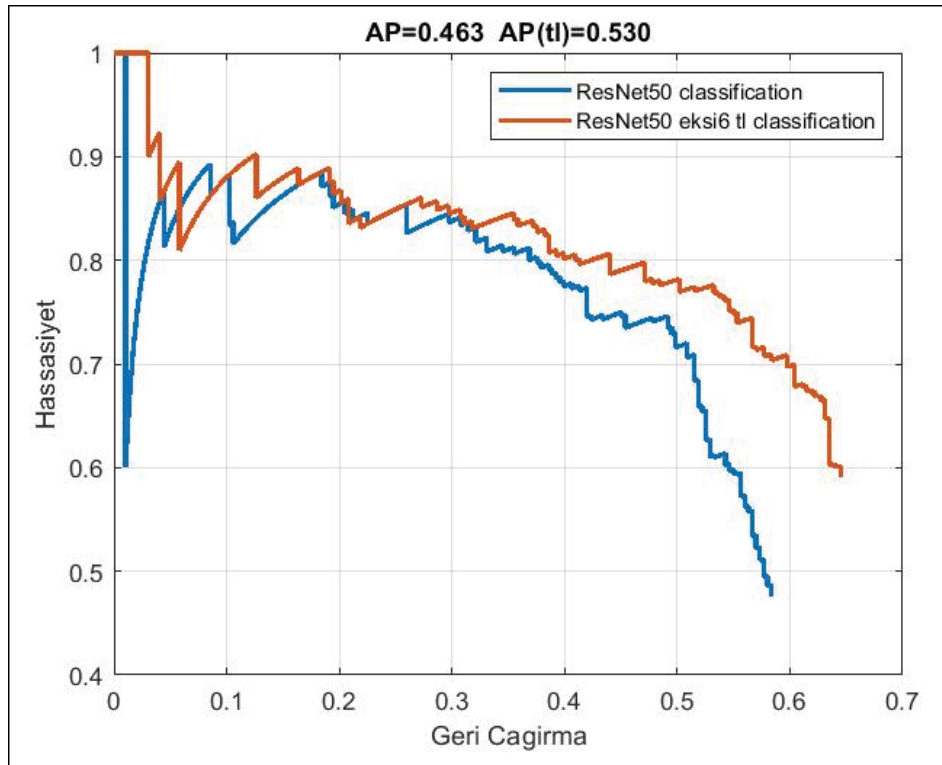
Şekil 8.9. Ağ modellerinin panelvan sınıfı geri çağırma-hassasiyet grafiği

Resim 8.7, daha hızlı RCNN ResNet50 classification ve daha hızlı RCNN ResNet50_eksi6_tl classification modellerinin aynı görüntü üzerinde panelvan tespit örneği olarak verilmiştir. Görüntüde üç araba ve bir panelvan sınıfı araç mevcuttur. Soldaki görüntüde daha hızlı RCNN ResNet50 classification modeli soldaki panelvan için iki sınırlayıcı kutu oluşturmuş, panelvan sınıfı için bir yanlış pozitif üretmiştir. Sağdaki görüntüde daha hızlı RCNN ResNet50_eksi6_tl classification modeli bir önceki modelin yanlış pozitifini gidererek, hatasız bir şekilde araçları tespit etmiştir.

Şekil 8.10 daha hızlı RCNN ResNet50 classification ve daha hızlı RCNN ResNet50_eksi6_tl classification modellerinin iş makinesi sınıfı için geri çağırma – hassasiyet grafiklerini göstermektedir. Grafikte daha hızlı RCNN ResNet50_eksi6_tl classification modelinin hem yüksek hassasiyet açısından hem de yüksek geri çağırma açısından iyileşme sağladığı gözlenmektedir. Yani daha hızlı RCNN ResNet50_eksi6_tl classification modeli panelvan araç sınıfı için, daha hızlı RCNN ResNet50 classification modeline göre daha az yanlış negatif ve yanlış pozitif üretmiştir. Ortalama hassasiyetteki artış da, daha hızlı RCNN ResNet50_eksi6_tl classification modelinin iş makinesi algılama başarısının daha iyi olduğunu göstermektedir.



Resim 8.7. Daha hızlı RCNN ResNet50 classification modeline ait bir panelvan tespit örneği(sol) ve aynı görüntü üzerinde daha hızlı RCNN ResNet50_eksi6_tl classification modeline ait bir panelvan tespit örneği(sağ)



Şekil 8.10. Ağ modellerinin iş makinesi sınıfı geri çağırma-hassasiyet grafiği

Resim 8.8, daha hızlı RCNN ResNet50 classification ve daha hızlı RCNN ResNet50_eksi6_tl classification modellerinin aynı görüntü üzerinde iş makinesi tespit

örneği olarak verilmiştir. Görüntüde altı adet iş makinesi mevcuttur. Soldaki görüntüde daha hızlı RCNN ResNet50 classification modeli ortadaki iş makinesi için iki sınırlayıcı kutu oluşturmuş, iş makinesi sınıfı için bir yanlış pozitif üretmiştir. Sağdaki görüntüde daha hızlı RCNN ResNet50_eksi6_tl classification modeli bir önceki modelin yanlış pozitifini gidererek, hatasız bir şekilde iş makinelerini tespit etmiştir.

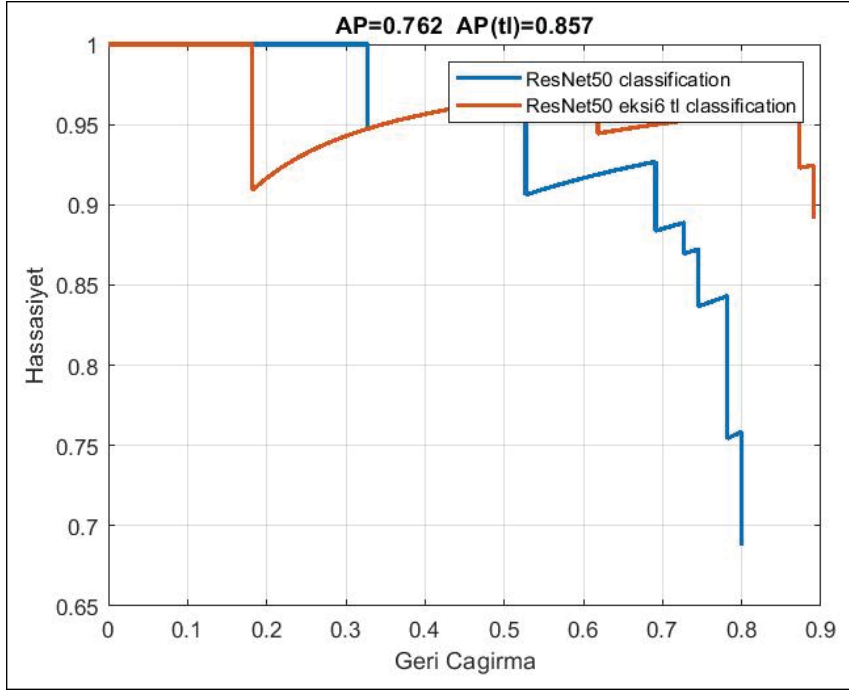


Resim 8.8. Daha hızlı RCNN ResNet50 classification modeline ait bir iş makinesi tespit örneği(sol) ve aynı görüntü üzerinde daha hızlı RCNN ResNet50_eksi6_tl classification modeline ait bir iş makinesi tespit örneği(sağ)

Şekil 8.11 daha hızlı RCNN ResNet50 classification ve daha hızlı RCNN ResNet50_eksi6_tl classification modellerinin uçak sınıfı için geri çağırma – hassasiyet grafiklerini göstermektedir. Grafığe göre; yüksek geri çağırma açısından bakıldığında, her iki modelin başlangıçta aynı çizgide devam ettiği sonrasında daha hızlı RCNN ResNet50_eksi6_tl classification modelinin bir miktar daha hızlı RCNN ResNet50 classification modelinin gerisinde kaldığı gözlenmiştir. Ancak yüksek hassasiyet açısından bakıldığında; daha hızlı RCNN ResNet50_eksi6_tl classification modelinin uçak sınıfı için çok büyük oranda bir iyileşme sağlamıştır. Ortalama hassasiyetteki artış, en fazla uçak sınıfı için gözlenmiştir.

Resim 8.9, daha hızlı RCNN ResNet50 classification ve daha hızlı RCNN ResNet50_eksi6_tl classification modellerinin aynı görüntü üzerinde uçak tespit örneği olarak verilmiştir. Görüntüde dört uçak, üç araba, dört kamyon, 1 panelvan ve 1 pikap mevcuttur. Soldaki görüntüde daha hızlı RCNN ResNet50 classification modeli soldan

ikinci uçak için iki sınırlayıcı kutu oluşturmuş, uçak sınıfı için bir yanlış pozitif üretmiştir. Sağdaki görüntüde daha hızlı RCNN ResNet50_eksi6_tl classification modeli bir önceki modelin yanlış pozitifini gidermiş, ancak her iki model de soldaki uçağı, en altta görüntüde yarım olarak yer alana bir panelvan ve bir kamyonu tespit edememiştir.



Şekil 8.11. Ağ modellerinin uçak sınıfı geri çağırma-hassasiyet grafiği



Resim 8.9. Daha hızlı RCNN ResNet50 classification modeline ait bir uçak tespit örneği(sol) ve aynı görüntü üzerinde daha hızlı RCNN ResNet50_eksi6_tl classification modeline ait bir uçak tespit örneği(sağ)

9. SONUÇLAR

Bu çalışmada, küçük nesne tespiti için uygun zorluktaki bir veri kümesi olan VEDAI ile araç algılama başarısını artırmak üzere ResNet ağı ve daha hızlı RCNN algoritmasında değişiklikler yapılmıştır. Daha hızlı RCNN algoritmasının bölge öneri ağına kullanılan bağlantı kutunu sayısı ve boyutunun, küçük nesneleri tespit etmekte olan etkisini araştırmak üzere yaptığımız deneysel çalışmalarda; genel olarak kullanılan bağlantı kutu sayısının artmasının algılama başarısını arttırdığı ancak işlem yükünün artmasından dolayı ağ eğitim süresini de arttırdığı gözlenmiştir. Daha hızlı RCNN algoritmasında önerilen orijinal boyutlarındaki bağlantı kutularının küçük nesne tespitinde yetersiz oldukları, daha küçük boyutlu ve veri kümesine uygun bağlantı kutularının IoU mesafe metriği ile hesaplanarak kullanılmasının algılama başarısında daha iyi sonuç gösterdiği; algılama başarı sonuçlarının orijinal bağlantı kutuları kullanan diğer çalışma sonuçları ile karşılaştırılmasında gösterilmiştir. Böylece küçük nesne tespiti için orijinal algoritmaların yetersiz olduğu, bu görevler için orijinal algoritmalara ince ayar yapılması gerektiği savı doğrulanmıştır. Ağ katman sayısının, küçük nesneleri tespit etmekte olan etkisini araştırmak üzere yaptığımız deneysel çalışmalarda ise; küçük nesne tespiti için daha sık ağların daha faydalı olabileceği sonucuna ulaşılmıştır. Beklenildiği gibi ağ derinliği arttıkça işlem sayısı da arttığı için ağ eğitim süresinin arttığı gözlenmiştir. ResNet18'in kullanıldığı ağ modelinin ResNet50'nin kullanıldığı ağ modelinden daha başarılı sonuç göstermesi üzerine, bu iki ağ derinliği arasındaki algılama başarısının araştırıldığı deneylerde; önceden eğitilmiş bir ağın bir algoritmaya uyarlanıp birkaç katmanın eklenmesi veya çıkarılması işlemlerinden sonra eğitilmesinin, önceden eğitilmiş bir ağın bir algoritmaya uyarlandıktan sonra eğitilmesine göre ağırlıkların çok daha büyük maliyetlerde optimizasyonuna ihtiyaç duyulduğu için algılama başarısında düşüşlere neden olduğu saptanmıştır. Bu optimizasyon maliyetinin, öğrenmenin aktarılması yöntemi ile azaltılabileceği gösterilmiştir. Yine öğrenmenin aktarımı yöntemi ile VEDAI veri kümesi üzerinde araçların tespitinde ortalama hassasiyette %74,9'a kadar algılama başarısının sağlanabileceği gösterilmiştir. Araçların sınıflandırılması çalışmalarında ise; hava görüntüleri üzerindeki araç tespitinde başarıyı artırımı için yapılan deneysel çalışmalarda elde edilen kazanımlar kullanılmıştır. Araç tespitinde en çok başarı sağlayan deney modeli ile araçların sınıflandırılması gerçekleştirilmiş ve araba için %72,4, kamyon için %58,6, pikap için %72, traktör için

%57,6, kamp arabası için %69,9, tekne için %60, panelvan için %56,4, iş makinası için %53,1 ve uçak için %85,7 ortalama hassasiyet sağlanmıştır.

Yapılan deneylerde hiper parametreler üzerinde eniyileme çalışmaları yapılmamıştır. Hiper parametreler üzerinde eniyileme çalışmaları yapılarak ve eğitimlere yerden yükseklik verileri aynı olan farklı kaynaklardan elde edilen hava görüntülerini içeren veri kümeleri dahil edilerek, önerilen modelin bir ürün haline getirilebileceği değerlendirilmektedir.

KAYNAKLAR

1. Liu, C., Ding, Y., Zhu, M., Xiu, J., Li, M, and Li, Q. (2019). Vehicle Detection in Aerial Images Using a Fast Oriented Region Search and the Vector of Locally Aggregated Descriptors. *Sensors*, 19, 3294.
2. Dalal, N., and Triggs, B. (2005). *Histograms of oriented gradients for human detection*. 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), 1, 886-893.
3. Lowe, D.G. (2004). Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 91–110.
4. Chang, C.C., and Lin, C.J. (2011). LIBSVM: A library for support vector machines. *ACM Trans. Intell. Syst. Technol*, 2(3), 27.
5. Viola, P.A. and Jones, M.J. (2001). *Rapid object detection using a boosted cascade of simple features*. Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1, I-I.
6. Felzenszwalb, P.F., Girshick, R.B., McAllester, D.A., and Ramanan, D. (2009). Object Detection with Discriminatively Trained Part Based Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32, 1627-1645.
7. Deng, J., Berg, A., Satheesh, S., Su, H., Khosla, A., and Li, F. ImageNet Large Scale Visual Recognition Competition 2012 (ILSVRC2012). URL:<http://image-net.org/challenges/LSVRC/2012/index>, Son Erişim Tarihi:04.07.2020.
8. Krizhevsky, A., Sutskever, I., and Hinton, G. (2012). *ImageNet classification with deep convolutional neural networks*. Proceedings of the 25th International Conference on Neural Information Processing System, 1097–1105.
9. Girshick, R.B., Donahue, J., Darrell, T., and Malik, J. (2014). *Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation*. 2014 IEEE Conference on Computer Vision and Pattern Recognition, 580-587.
10. Girshick, R.B. (2015). *Fast R-CNN*. 2015 IEEE International Conference on Computer Vision (ICCV), 1440-1448.
11. Ren, S., He, K., Girshick, R.B., and Sun, J. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39, 1137-1149.
12. Tayara, H., Soo, K.G., and Chong, K.T. (2018). Vehicle Detection and Counting in High-Resolution Aerial Images Using Convolutional Regression Neural Network. *IEEE Access*, 6, 2220-2230.
13. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S.E., Fu, C., and Berg, A.C. (2016). SSD: Single Shot MultiBox Detector. *ECCV*.

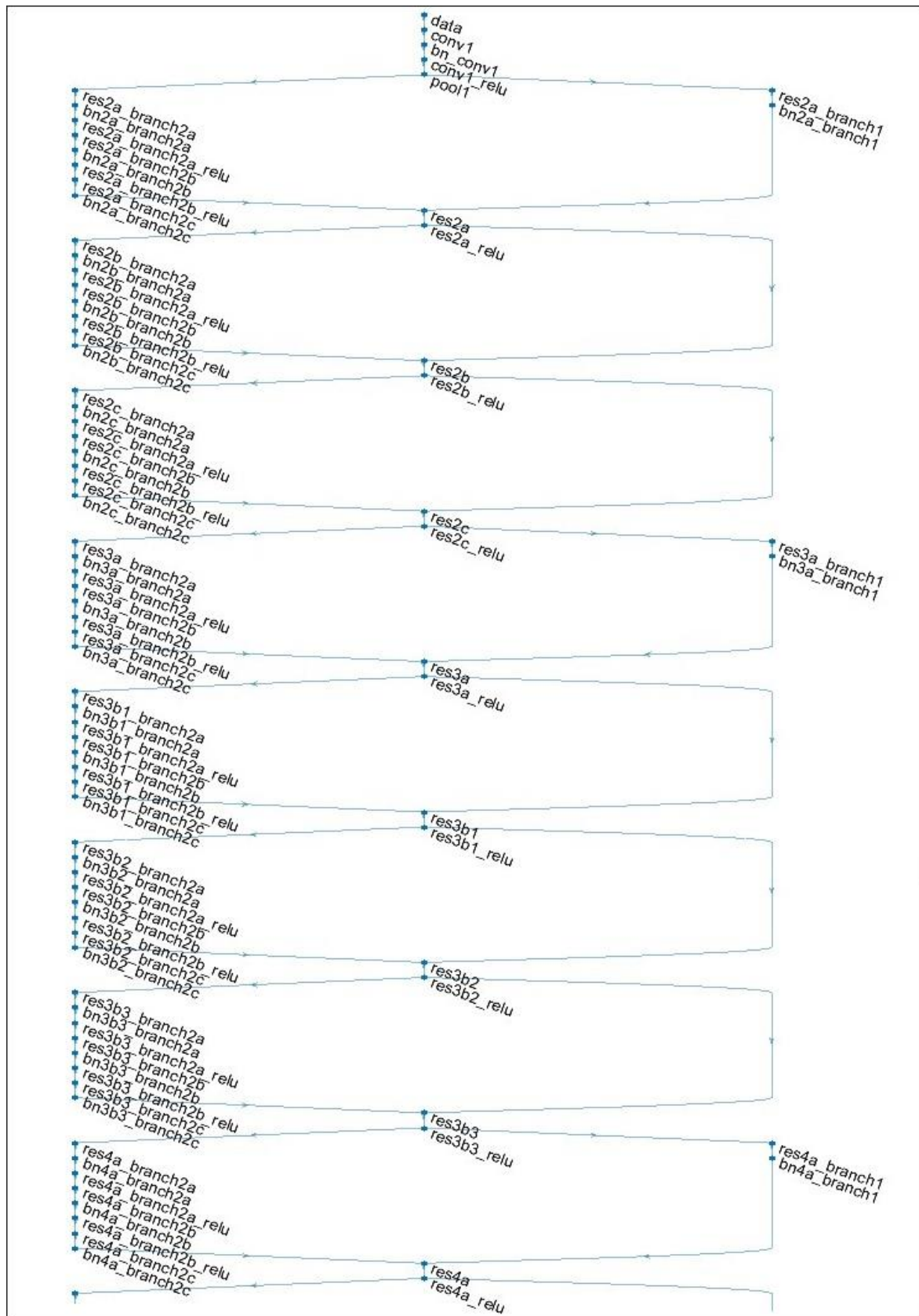
14. Redmon, J., Divvala, S.K., Girshick, R.B., and Farhadi, A. (2016). *You Only Look Once: Unified, Real-Time Object Detection*. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 779-788.
15. Redmon, J., and Farhadi, A. (2017). *YOLO9000: Better, Faster, Stronger*. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 6517-6525.
16. Redmon, J., and Farhadi, A. (2018). *YOLOv3: An Incremental Improvement*.
17. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M.S., Berg, A.C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115, 211-252.
18. He, K., Zhang, X., Ren, S., and Sun, J. (2016). *Deep Residual Learning for Image Recognition*. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770-778.
19. Deng, J., Dong, W., Socher, R., Li, L., Li, K., and Fei-Fei, L. (2009). *ImageNet: A large-scale hierarchical image database*. 2009 IEEE Conference on Computer Vision and Pattern Recognition, 248-255.
20. Lin, T., Maire, M., Belongie, S.J., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C.L. (2014). Microsoft COCO: Common Objects in Context. *ArXiv*, abs/1405.0312.
21. Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 2278–2324.
22. Albawi, S., Mohammed, T.A., and Al-Zawi, S. (2017). *Understanding of a convolutional neural network*. 2017 International Conference on Engineering and Technology (ICET), 1-6.
23. Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, G., Cai, J., and Chen, T. (2018). Recent advances in convolutional neural networks. *Pattern Recognit.*, 77, 354-377.
24. Bozkurt, S. (2018). *Derin Öğrenme Algoritmaları Kullanılarak Çay Alanlarının Otomatik Segmentasyonu*, Yüksek Lisans Tezi, Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü, İstanbul.
25. İnik, Ö., and Ülker, E. (2017). Derin Öğrenme ve Görüntü Analizinde Kullanılan Derin Öğrenme Modelleri. *Gaziosmanpaşa Bilimsel Araştırma Dergisi*, 6(3), 85-104.
26. İnternet: Convolutional Neural Networks (CNNs / ConvNets). (2020). URL:<https://cs231n.github.io/convolutional-networks/>, Son Erişim Tarihi: 04.07.2020
27. İnternet: MathWorks. (2020). Specify Layers of Convolutional Neural Network. URL: <https://www.mathworks.com/help/deeplearning/ug/layers-of-a-convolutional-neural-network.html>, Son Erişim Tarihi: 04.07.2020.

28. Ioffe, S., and Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *ArXiv*, abs/1502.03167.
29. İnternet: MathWorks. (2020). batchNormalizationLayer. URL: <https://www.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.batchnormalizationlayer.html>, Son Erişim Tarihi: 04.07.2020.
30. Nair, V., and Hinton, G.E. (2010). *Rectified Linear Units Improve Restricted Boltzmann Machines*. 27th International Conference on Machine Learning (ICML), 807-814.
31. Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. New York: Springer, 738.
32. Simonyan, K., and Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR*, abs/1409.1556.
33. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). *Going deeper with convolutions*. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 1-9.
34. He, K., Zhang, X., Ren, S., and Sun, J. (2016). Identity Mappings in Deep Residual Networks. *ArXiv*, abs/1603.05027.
35. Zhang, A., Lipton, Z.C., Li, M., and Smola, A.J. (2019). Dive into Deep Learning. URL: <https://d2l.ai/>, Son Erişim Tarihi: 04.07.2020.
36. Uijlings, J.R., Sande, K.E., Gevers, T., and Smeulders, A.W. (2013). Selective Search for Object Recognition. *International Journal of Computer Vision*, 104, 154-171.
37. Zeiler, M.D., and Fergus, R. (2014). *Visualizing and Understanding Convolutional Networks*. European Conference on Computer Vision (ECCV), 8689, 818-833.
38. İnternet: MathWorks. (2020). Faster R-CNN Examples. URL: <https://www.mathworks.com/help/vision/ug/faster-r-cnn-examples.html>, Son Erişim Tarihi: 04.07.2020.
39. İnternet: MathWorks. (2020). trainFasterRCNNObjectDetector. URL: <https://www.mathworks.com/help/vision/ref/trainfastercnnobjectdetector.html>, Son Erişim Tarihi: 04.07.2020.
40. Razakarivony, S., and Jurie, F. (2016). Vehicle detection in aerial imagery : A small target detection benchmark. *J. Vis. Commun. Image Represent.*, 34, 187-203.
41. İnternet: Utah Automated Geographic Reference Center. (2012). URL: <https://gis.utah.gov/>, Son Erişim Tarihi: 04.07.2020.
42. Zhong, J., Lei, T., and Yao, G. (2017). Robust Vehicle Detection in Aerial Images Based on Cascaded Convolutional Neural Networks. *Sensors*, 17(12), 2720.
43. Sakla, W.A., Konjevod, G., and Mundhenk, T.N. (2017). *Deep Multi-modal Vehicle Detection in Aerial ISR Imagery*. 2017 IEEE Winter Conference on Applications of Computer Vision (WACV), 916-923.

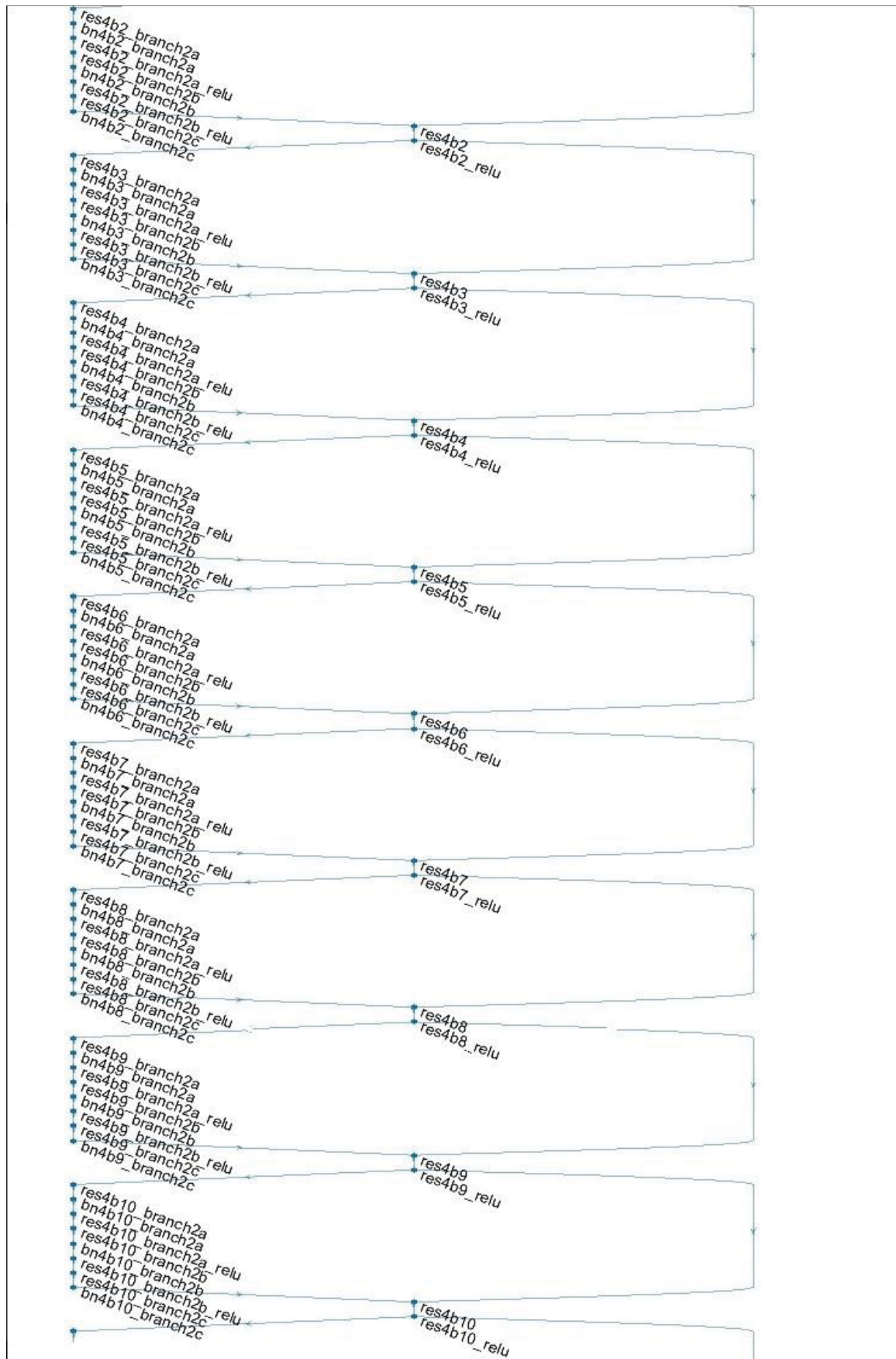
44. İnternet: MathWorks. (2020). Object Detection Using Faster R-CNN Deep Learning. URL: <https://uk.mathworks.com/help/vision/examples/object-detection-using-faster-r-cnn-deep-learning.html>, Son Erişim Tarihi: 04.07.2020.
45. Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. (2017). Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. *ArXiv*, abs/1602.07261.
46. Huster, T., and Gale, N.C. (2016). Deep learning for pedestrian detection in aerial imagery. *In MSS Passive Sensors*.
47. Carlet, J., and Abayowa, B. (2017). Fast Vehicle Detection in Aerial Imagery. *ArXiv*, abs/1709.08666.
48. Carlet, J., and Abayowa, B. (2017). Fast Vehicle Detection in Aerial Imagery. *ArXiv*, abs/1709.08666.

EKLER

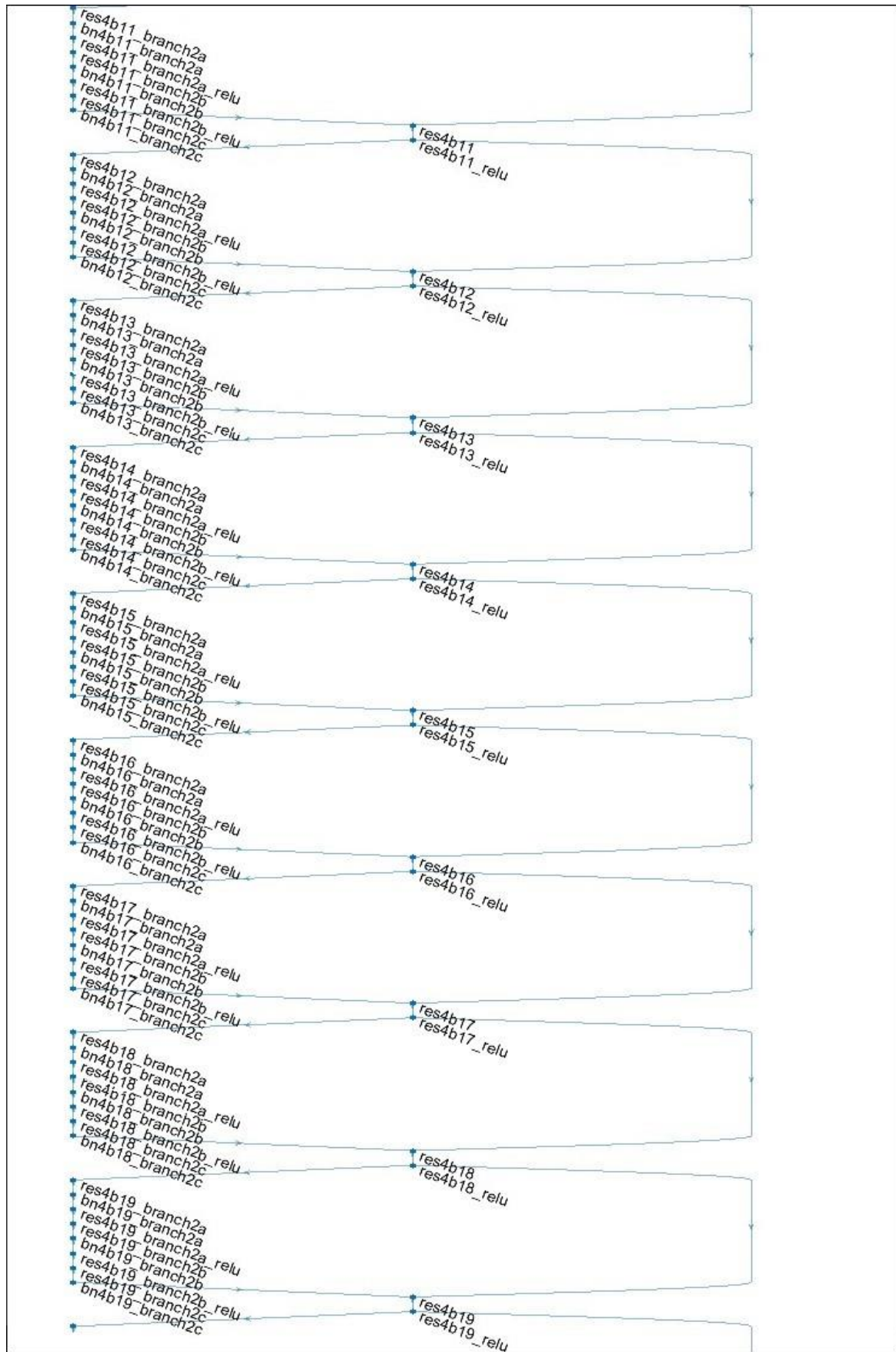
EK-1. ResNet101 Ağ Mimarisi



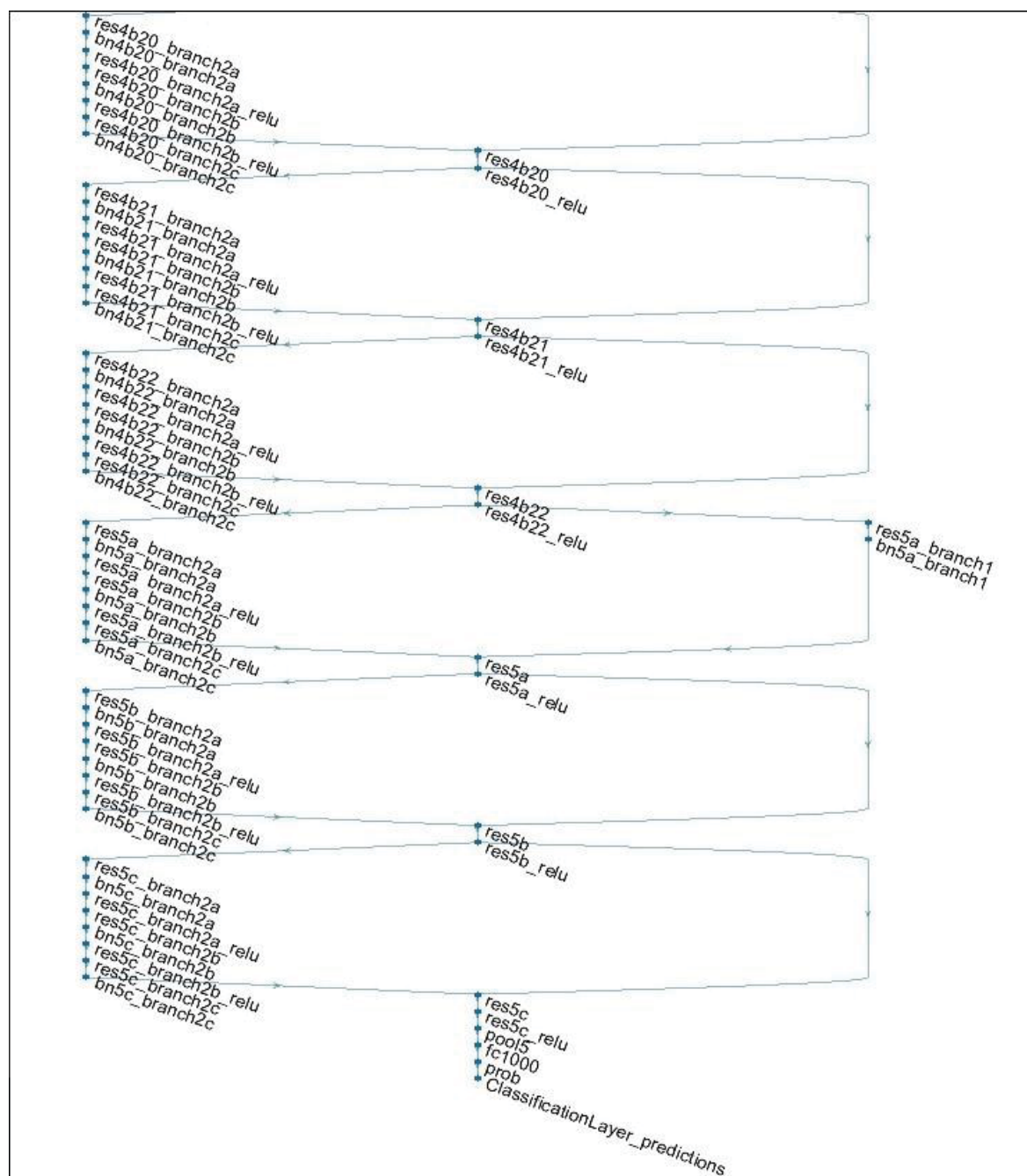
EK-1. (devam) ResNet101 Ağ Mimarisi



EK-1. (devam) ResNet101 Ağ Mimarisi



EK-1. (devam) ResNet101 Ağ Mimarisi



ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı : ER, Ömer
 Uyuğu : T.C.
 Doğum tarihi ve yeri : 31.08.1992, Aksaray
 Medeni hali : Bekar
 Telefon : 0 (312) 592 10 00
 e-mail : omerer.eee@gmail.com



Eğitim

Derece	Eğitim Birimi	Mezuniyet Tarihi
Yüksek lisans	Gazi Üniversitesi / Elektrik Elektronik Mühendisliği	Devam Ediyor
Lisans	Hacettepe Üniversitesi / Elektrik Elektronik Mühendisliği	2016
Lise	Antalya Lisesi	2010

İş Deneyimi

Yıl	Yer	Görev
2016-Halen	ASELSAN A.Ş.	Mühendis

Yabancı Dil

İngilizce

Yayınlar

1. ER, Ö., ve BİLGE H.Ş. (2020). *Küçük Nesne Tespit Zorluğu Olarak Hava Görüntüleri Üzerinde Araç Tespiti*. 3rd International Conference on Data Science and Applications (ICONDATA'20), 25.
2. ER, Ö., ve BİLGE H.Ş. (2020). Bir Küçük Nesne Tespit Zorluğu Olarak Hava Görüntülerinden Araç Tespiti. *Veri Bilimi*.

Hobiler

Yüzmek, seyahat etmek



GAZİ GELECEKTİR..