BURSA TEKNİK ÜNİVERSİTESİ ❖ FEN BİLİMLERİ ENSTİTÜSÜ

DERİN ÖĞRENME ALGORİTMALARI KULLANARAK GÜVENLİ ÇOCUK OYUN ALANLARINDA KARŞILAŞTIRMALI TEHLİKELİ NESNE TESPİTİ YAPILMASI

YÜKSEK LİSANS TEZİ

Mehmet Fatih İNKAYA

Elektrik-Elektronik Mühendisliği Anabilim Dalı

Ocak 2021

BURSA TEKNİK ÜNİVERSİTESİ ❖ FEN BİLİMLERİ ENSTİTÜSÜ

DERİN ÖĞRENME ALGORİTMALARI KULLANARAK GÜVENLİ ÇOCUK OYUN ALANLARINDA KARŞILAŞTIRMALI TEHLİKELİ NESNE TESPİTİ YAPILMASI

YÜKSEK LİSANS TEZİ

Mehmet Fatih İNKAYA
(181278034001)
ORCID: 0000-0003-4807-4298

Elektrik-Elektronik Mühendisliği Anabilim Dalı

Tez Danışmanı: Prof. Dr. Hakan GÜRKAN
ORCID: 0000-0002-7008-4778

Ocak 2021

BTÜ, Fen Bilimleri Enstitüsü'nün 181278034001 numaralı Yüksek Lisans Öğrencisi Mehmet Fatih İNKAYA, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı "DERİN ÖĞRENME ALGORİTMALARI KULLANARAK GÜVENLİ ÇOCUK OYUN ALANLARINDA KARŞILAŞTIRMALI TEHLİKELİ NESNE TESPİTİ YAPILMASI" başlıklı tezini aşağıda imzaları olan jüri önünde başarı ile sunmuştur.

Tez Danışmanı : **Prof. Dr. Hakan GÜRKAN** .....
Bursa Teknik Üniversitesi

Jüri Üyeleri : **Prof. Dr. Erdoğan DİLAVEROĞLU** .....
Bursa Uludağ Üniversitesi

Dr. Öğr. Üyesi Mustafa ÖZDEN .....
Bursa Teknik Üniversitesi

Teslim Tarihi : 27.01.2021
Savunma Tarihi : 08.02.2021

20.04.2016 tarihli Resmi Gazete'de yayımlanan Lisansüstü Eğitim ve Öğretim Yönetmeliğinin 9/2 ve 22/2 maddeleri gereğince; Bu Lisansüstü teze, Bursa Teknik Üniversitesi'nin abonesi olduğu intihal yazılım programı kullanılarak Fen Bilimleri Enstitüsü'nün belirlemiş olduğu ölçütlere göre uygun rapor alınmıştır.

İNTİHAL BEYANI

Bu tezde görsel, işitsel ve yazılı biçimde sunulan tüm bilgi ve sonuçların akademik ve etik kurallara uyularak tarafımdan elde edildiğini, tez içinde yer alan ancak bu çalışmaya özgü olmayan tüm sonuç ve bilgileri tezde kaynak göstererek belgelediğimi, aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul ettiğimi beyan ederim.

Mehmet Fatih İNKAYA

Oğluma, kızıma ve tüm çocuklara,

ÖNSÖZ

Hazineler dolusu bir servetten daha değerli bir şey varsa eğer, o da iyi bir çocuk yetiştirmektir. Bunun önemini kavrayan toplumlar çocukları daha iyi yetiştirilebilmek için tüm imkanlarını seferber eder ve geleceklerini teminat altına alır. Güvenli ve sağlıklı bir ortamda büyüyen çocuklar daha mutlu bir geleceği inşa edecektir. Bu çalışmanın her aşamasında kendilerini düşünerek motive olduğum oğluma ve kızıma sonsuz sevgilerimi sunarım.

Çalışmanın hazırlanmasında bilgisi, donanımı ve tecrübeleriyle desteğini esirgemeyen tez danışmanım Prof. Dr. Hakan GÜRKAN'a, tez konusunun belirlenmesi aşamasında bana yol gösteren Bursa TTO yöneticisi Doç. Dr. Nadir YILDIRIM'a, hayatımda bana her zaman örnek olan annem Şerife İNKAYA ve babam Mehmet İNKAYA'ya, çalışmanın her safhasında beni motive eden fedakar eşim Sibel İNKAYA'ya, neşesi ve heyecanıyla bana destek olan kardeşim Semiha AKBULUT'a ve çalışma esnasında bana imkan sağlayan Bursa Büyükşehir Belediyesi yöneticilerine teşekkür ederim.

Ocak 2021

Mehmet Fatih İNKAYA
Elektronik Mühendisi

İÇİNDEKİLER

Sayfa

ÖNSÖZ ..... **v**
İÇİNDEKİLER ..... **v**
KISALTMALAR ..... **vii**
SEMBOLLER ..... **ix**
ÇİZELGE LİSTESİ ..... **x**
ŞEKİL LİSTESİ ..... **xi**
ÖZET ..... **xiii**
SUMMARY ..... **xiv**
1. GİRİŞ **1**
1.1 Tezin Amacı 2
1.2 Literatür Araştırması 4
1.2.1 Geleneksel nesne tespit algoritmaları 4
1.2.1.1 Veri girişi ve ön işleme 5
1.2.1.2 Öznitelik çıkarımı 5
1.2.1.3 Sınıflandırma 7
1.2.2 Derin öğrenme temelli algoritmalar 11
1.2.2.1 Tanımlar 12
1.2.2.2 CNN (ESA) mimarisi 13
1.2.2.3 CNN katmanları 13
1.2.2.4 CNN eğitimi 17
1.2.2.5 Derin öğrenme temelli görüntü sınıflandırma modelleri 18
1.2.2.6 Derin öğrenme temelli bölge önermeli nesne tanıma modelleri 19
1.2.2.7 Derin öğrenme temelli tek aşamalı nesne tanıma modelleri 26
1.2.3 YOLOv3 nesne tanıma algoritması 26
2. MATERYAL VE YÖNTEM **31**
3.1 Giriş 31
3.2 Darknet ve Nesne Tanıma Ortamlarının Kurulumu 32
3.2.1 OpenCV 32
3.2.2 Visual studio (VS) 32
3.2.3 CUDA 32
3.2.4 CuDNN 32
3.3 Hazır Veri Seti ve Ağırlıklarla Nesne Tanıma 32
3.4 YOLOv3 Modelinin Optimizasyonu 33
3.4.1 Yeni veri setinin hazırlanması 34
3.4.2 YOLOv3 modelinin eğitilmesi 35
3.4.3 Eğitim sonuçlarının değerlendirilmesi 37
3. DENEYSEL SONUÇLAR **40**
4. SONUÇLAR VE ÖNERİLER **50**
KAYNAKLAR **51**
ÖZGEÇMİŞ **55**

KISALTMALAR

AP : Ortalama Sınıf Hassasiyet Değeri
CDC : Centers for Disease Control and Prevention
CNN : Convolutional Neural Networks
CPU : Central Process Unit
DVM : Destek Vektör Makineleri
ESA : Evrişimsel Sinir Ağları
FP : False Positive (Yanlış Pozitif)
FN : False Negative (Yanlış Negatif)
GPS : Global Positioning System
GPU : Graphics Processing Unit
HOG : Histogram of Oriented Gradients
IoT : Internet of Things
IoU : Intersection over Union
mAP : Mean Average Precision
MLP : Multi Layer Perceptron
NMS : Non-Maximum Suppression
ReLU : Rectified Linear Units
R-CNN : Region Based Convolutional Neural Network
RGB : Red, Green, Blue
RNA : Ribo Nükleik Asit
RNN : Recurrent Neural Network
RoI : Region of Interest
RPN : Region Proposal Network
SIFT : Scale-Invariant Feature Transform
SSD : Single-Shot Detector
SURF : Speeded Up Robust Features
TP : True Positive (Doğru Pozitif)
TN : True Negative (Doğru Negatif)
VGG : Visual Geometry Group
VOC : Visual Object Classes

VS : Visual Studio

YOLO : You Only Look Once

SEMBOLLER

α	: Öğrenme Katsayısı
E	: Toplam Hata
g	: Gradyan Genliği
$g(x)$	: Aktivasyon Fonksiyonu
$p(r)$	: Hassasiyet-Duyarlılık Fonksiyonu
t_j	: Hedef Değer
w	**:** Ağırlıklar
x_i	: Giriş Değeri
y_j	: Çıkış Değeri
θ	: Gradyan Açısı

ÇİZELGE LİSTESİ

Sayfa

Çizelge 1.1: Pascal VOC verisetinde R-CNN, Fast R-CNN ve Faster R-CNN testi. **24**
Çizelge 2.1: YOLOv3 etiketleme dosyası içeriği…… **36**
Çizelge 2.2: YOLOv3 data dosya yapısı **36**
Çizelge 2.3: YOLOv3 cfg dosyası önemli parametreleri **37**
Çizelge 2.4: Hata Matrisi **38**
Çizelge 3.1: Farklı hiperparametrelerle yapılan eğitimlerin başarım değerleri. **40**
Çizelge 3.2: Eğitimlerin hassasiyet, duyarlılık, F1-skoru ve IoU değerleri. **41**
Çizelge 3.3: Eğitim-2 ve Eğitim-6 güven skoru karşılaştırması. **43**
Çizelge 3.4: Eğitimlerin AP, mAP ve ortalama hata değerleri. **44**
Çizelge 3.5: Önerilen modelin benzer çalışmalarla karşılaştırılması. **48**

ŞEKİL LİSTESİ

Sayfa

Şekil 1.1 : Geleneksel nesne tespit aşamaları **5**
Şekil 1.2 : HOG aşamaları **5**
Şekil 1.3 : Yapay sinir hücresi yapısı **8**
Şekil 1.4 : Adaline mimarisi **8**
Şekil 1.5 : Çok katmanlı yapay sinir ağı mimarisi **9**
Şekil 1.6 : Gradyan iniş algoritması **10**
Şekil 1.7 : Tanımlar **12**
Şekil 1.8 : CNN (ESA) mimarisi **13**
Şekil 1.9 : RGB girdi görüntüsüne konvolüsyon işleminin uygulanması **14**
Şekil 1.10 : Yaygın kullanılan filtre tipleri **15**
Şekil 1.11 : Girdi görüntüsünün konvolüsyon ve ReLU çıktı görüntüsü **15**
Şekil 1.12 : Maksimum havuzlama işlemi **16**
Şekil 1.13 : ReLu çıktı görüntüsünün havuzlama çıktı görüntüsü **16**
Şekil 1.14 : DropOut işlemi **17**
Şekil 1.15 : CNN eğitim algoritması **17**
Şekil 1.16 : ImageNet yarışmasının kazananları **18**
Şekil 1.17 : Görüntü sınıflandırma işlemi **19**
Şekil 1.18 : Nesne tanıma işlemi **20**
Şekil 1.19 : IoU hesaplanması **20**
Şekil 1.20 : NMS işlemi **21**
Şekil 1.21 : R-CNN aşamaları **21**
Şekil 1.22 : Fast R-CNN aşamaları **22**
Şekil 1.23 : RoI havuzlama işlemi **22**
Şekil 1.24 : Faster R-CNN mimarisi **23**
Şekil 1.25 : Ankorlar ile bölge önerimi **24**
Şekil 1.26 : Mask R-CNN Algoritması **25**
Şekil 1.27 : Mask R-CNN Çıkış Görüntüsü **25**
Şekil 1.28 : SSD mimarisi **26**
Şekil 1.29 : Darknet-53 mimarisi **27**
Şekil 1.30 : YOLOv3 mimarisi **27**
Şekil 1.31 : Izgaralar ile bölge önerisi **28**
Şekil 1.32 : YOLOv3 ile sınır kutusu tahmini **29**
Şekil 1.33 : YOLOv3 çıkışındaki hücre derinliği **29**
Şekil 1.34 : Derin öğrenme algoritmalarının COCO veri setinde performansları **30**
Şekil 2.1 : Hızlı yardım istasyonu uygulaması **31**
Şekil 2.2 : Hatalı nesne tanıma sahnesi-1 ..... **33**
Şekil 2.3 : Hatalı nesne tanıma sahnesi-2 ..... **33**
Şekil 2.4 : Hızlı yardım istasyonu ile hayvan barınağından çekilen görüntüler **34**
Şekil 2.5 : Hızlı yardım istasyonu ile çocuk oyun alanında çekilen görüntüler **34**

Şekil 2.6 : Open Images V6 veri setinden alınan örnek görüntüler **35**
Şekil 2.7 : LabelImg etiketleme aracı kullanılarak resimlerin etiketlenmesi **35**
Şekil 2.8 : Hassasiyet-duyarlılık eğrisi **39**
Şekil 3.1 : Eğitimlerin en iyi hassasiyet-duyarlılık noktaları **42**
Şekil 3.2 : Eğitim-2 ağırlıkları ile validasyon testi **42**
Şekil 3.3 : Eğitim-6 ağırlıkları ile validasyon testi **43**
Şekil 3.4 : Eğitim-1 ortalama hata ve mAP eğrisi **44**
Şekil 3.5 : Eğitim-2 ortalama hata ve mAP eğrisi **45**
Şekil 3.6 : Eğitim-3 ortalama hata ve mAP eğrisi **45**
Şekil 3.7 : Eğitim-4 ortalama hata ve mAP eğrisi **46**
Şekil 3.8 : Eğitim-5 ortalama hata ve mAP eğrisi **46**
Şekil 3.9 : Eğitim-6 ortalama hata ve mAP eğrisi **47**
Şekil 3.10 : Önerilen modelle doğru köpek tespiti sahnesi-1 ..... **47**
Şekil 3.11 : Önerilen modelle doğru köpek tespiti sahnesi-2 ..... **48**

DERİN ÖĞRENME ALGORİTMALARI KULLANARAK GÜVENLİ ÇOCUK OYUN ALANLARINDA KARŞILAŞTIRMALI TEHLİKELİ NESNE TESPİTİ YAPILMASI

ÖZET

Türkiye Cumhuriyeti Sağlık Bakanlığı Halk Sağlığı Genel Müdürlüğü verilerine göre Türkiye'de her yıl yaklaşık 250.000 kuduz riskli vaka görülmektedir. Bu vakaların büyük bir kısmı çocukların tatilde olduğu yaz aylarında ve çocuk oyun alanlarında köpek saldırısı şeklinde gerçekleşmektedir. Derin öğrenme temelli bilgisayar görüsü teknolojisinin gelişmesiyle birlikte çocuk oyun alanlarındaki tehlikeli nesnelerin (silah, bıçak, köpek vb.) otomatik tespiti önemli bir güvenlik uygulaması olarak ortaya çıkmıştır.

Bu çalışmada daha önceden çocuk oyun alanlarında konumlandırılmış olan hızlı yardım istasyonları kullanılarak otomatik nesne (köpek) tespitinin yapılması amaçlanmıştır. Hem geleneksel hem de bölge önermeli ve tek aşamalı derin öğrenme temelli nesne tespit algoritmaları incelenmiş, başarım değerleri karşılaştırılmıştır. Karşılaştırma sonrasında bu tez çalışmasında yüksek fps ve mAP değeri nedeniyle YOLOv3 algoritmasının kullanılması tercih edilmiştir. Tek aşamalı nesne tespit algoritmaları görüntüyü tek bir seferde regresyon işlemine tabi tutarak nesnelerin sınıfını ve konumlarını aynı anda tahmin ettiklerinden bölge önermeli modellerden daha hızlıdır ve gerçek zamanlı uygulamalar için kullanımı daha uygundur. Güvenlik sistemlerindeki gerçek zamanlı ve anlık tepki verilmesi ihtiyacı ile çocuk oyun alanlarındaki çocuk ve köpeklerin hızlı ve hareketli yapıları göz önüne alınarak çocuk oyun alanları için YOLOv3 algoritması temelli nesne tespit modeli önerilmiştir.

Öncelikle, orijinal veri setlerindeki sahipsiz köpek görüntü verisi eksikliğinin giderilmesi için bir hayvan barınağındaki sahipsiz köpeklerin görüntüleri çekilerek veri setine eklenmiştir. Sonrasında, çocuk oyun alanında mevcut hızlı yardım istasyonunun kamera kayıtları ve açık kaynak veri setinden alınan görüntüler de bu veri setine eklenerek toplam 878 görüntüden oluşan yeni bir veri seti elde edilmiştir. Bu yeni görüntüler eğitim verisi çeşitliliğini etkili bir şekilde arttırmış ve önerilen modelin eğitim performansını iyileştirmiştir. Önerilen model, çeşitli hiperparametreler kullanılarak optimize edilmiş ve sonuçlar birbiriyle karşılaştırılmıştır. En iyi değerlendirme sonuçlarına sahip model yeni bir akıllı şehir güvenlik uygulaması olarak önerilmiş ve seçilen çocuk oyun alanında bulunan hızlı yardım istasyonuna entegre edilerek köpeklerin otomatik olarak tespit edilmesi için uygulanmıştır.

Optimize edilen hiperparametrelerle birlikte gerçek zamanlı köpek tespit uygulaması çocuk oyun alanında ihtiyaç duyulan hız ve doğruluğu da karşılayacak şekilde %82,59 AP @25 fps değerine ulaşmıştır.

Anahtar kelimeler: YOLOv3, Akıllı Şehir, Çocuk Oyun Alanı, Köpek Tespiti, Hızlı Yardım İstasyonu

COMPARATIVE SUSPICIOUS OBJECT DETECTION IN SAFE PLAYGROUNDS USING DEEP LEARNING ALGORITHMS

SUMMARY

According to the reports of Turkish Public Health Institution, approximately 250,000 rabies-risk animal bites occur per year in Turkey. Most of these bites are caused by dogs and most of the victims are the children who play in playgrounds in summer holidays. With the development of deep learning-based computer vision technology, autonomous detection of dangerous objects (handguns, knives, dogs, etc.) in these children's playgrounds has become a crucial security application.

In this study, it is aimed to perform autonomous object (dog) detection by using fast emergency station located in the selected children's playground. Both traditional methods and region-proposal and single shot deep learning-based object detection algorithms have been examined. The evaluation scores of these algorithms are compared. After the comparison, in this thesis the YOLOv3 algorithm has been preferred due to its high speed and high mAP value. Single shot algorithms regress the image at once, estimate the class of objects and their position at the same time. Therefore, they are faster than region-proposal algorithms and more suitable for real-time detection. The YOLOv3 algorithm-based object detection model is proposed for children's playgrounds, due to the need for real-time and instant response in security systems and the fast moving of children and dogs in children's playgrounds.

Firstly, in view of the problem of insufficient stray dog image data in the original datasets, new images of stray dogs have been taken from an animal shelter and they have been added to the dataset. Then, a new dataset that contains 878 images has been created by adding video recordings of the fast emergency station and images from an open images dataset. These new images have effectively enriched the diversity of training data and improved the training performance of the proposed model. The proposed model has been optimized by utilizing various hyperparameters and the results have been compared with each other. The model with the best evaluation scores is proposed as a new smart city security application and applied to detect dogs automatically by the fast emergency station located in the selected children's playground.

The real-time application achieved 82.59% of AP @25 fps with adjusted hyperparameters to meet the speed and accuracy needs in the children's playgorund.

Keywords: YOLOv3, Smart City, Children's Playground, Dog Detection, Fast Emergency Station

1. GİRİŞ

Maslow'un ihtiyaçlar hiyerarşisine göre insanların ihtiyaçları sınırsızdır ve birey alt düzeydeki ihtiyacının karşılanmasının ardından bir üst düzeydeki ihtiyaç düzeyine ulaşır. Yapılması gereken bireyin en temel ihtiyacı olan fizyolojik ihtiyaçlardan kendini gerçekleştirme ihtiyacına kadar olan süreçte engelleri ortadan kaldırmasına yardım etmektir [1].

Birleşmiş Milletler kentleşme raporuna göre 2019 yılı itibariyle dünya nüfusunun %56'sı şehirlerde yaşamakta olup 2050 yılında bu oranın %70'e ulaşacağı öngörülmektedir. Bu hızlı nüfus artışına rağmen insanların günümüz ve gelecekteki tüm ihtiyaçlarını karşılayabilmeleri için daha yaşanabilir ve sürdürülebilir Akıllı Şehir (Smart City) kavramı ortaya çıkmıştır.

Akıllı şehir bireylerin yaşam kalitesini yükseltmek, kaynakları etkin ve verimli kullanmak amacı ile teknolojik imkânlardan ve verilerden en ileri seviyede yararlanır. Büyük veri, sosyal ağlar, açık veri, blok zincir, nesnelerin interneti (IoT), yapay zeka ve bulut bilişim akıllı şehirlerde yükselen küresel eğilimler ve yeni teknolojilerdendir. Özellikle işlemci güçlerinin artmasına bağlı olarak akıllı şehirlerde derin öğrenme temelli akıllı uygulamalar tüm fonksiyon alanlarında kullanılır hale gelmiştir.

Akıllı hareketlilik alanında trafik kazası, duran araç, aşırı hız ve kırmızı ışık ihlal tespitleri yapılmakta sürücüler mobil uygulamalar, değişken mesaj ve trafik işaretleri ile uyarılmaktadır. Polis, itfaiye ve ambulans gibi geçiş üstünlüğüne sahip araçlar kavşaklara yaklaştığında trafik sinyalizasyon sistemleri otomatik geçiş sağlamaktadır. Ayrıca, yollar üzerindeki trafik işaretleri, saha panoları, hidrant vb. saha unsurları derin öğrenme temelli nesne tespit uygulamaları kullanılarak GPS koordinatları ile birlikte şehir haritalarına işlenmekte ve periyodik kontrolleri yapılmaktadır.

Akıllı sağlık alanında biyoinformatik (kanser teşhisi, gen seçimi, çeşitliliği ve sınıflandırması, ilaç tasarımı, bileşim protein etkileşimi, RNA ile protein ilişkisi, DNA metilasyonu vb.), medikal görüntüleme (3 boyutlu beyin rekonstrüksiyonu, nöral hücre ve beyin doku sınıflandırması, Alzheimer teşhisi, doku sınıflandırması, organ

bölümleme, hücre kümeleme, iç kanama tespiti, tümör tespiti vb.) ve medikal bilişim (hastalık tahmini, insan davranışı izleme, veri madenciliği vb.) konularındaki akıllı uygulamalar ile daha başarılı tanı ve sonuçlar elde edilmektedir [2].

Akıllı güvenlik alanında derin öğrenme temelli kayıp/bırakılan nesne [3], duman tespiti [4], yüz tanıma [5], anormal olay tespiti [6] uygulamaları hatalı alarmların önlenmesi ve insana dayalı doğrulamanın azaltılması için yoğun bir şekilde kullanılmaktadır.

Tehlikeli nesnelerin tespiti alanında yapılan derin öğrenme temelli bilimsel çalışmalarla silah [7], tabanca [8] ve köpek [9] gibi nesneler daha başarılı şekilde tespit edilmeye başlanmıştır.

Çocuk oyun alanlarında olası tehlikelerin (çocuk kaybolma, kaçırılma ve istismarı, ateşli silahlar ve bıçaklar, köpekler) güvenlik merkezine manuel olarak bildirildiği hızlı yardım istasyonu uygulaması Bursa Büyükşehir Belediyesi tarafından Mart 2019'da hayata geçirilmiştir [10].

1.1 Tezin Amacı

Hızlı yardım istasyonu üzerinden Mart 2019 ile Eylül 2019 arasında yapılan tüm görüşmelere ait kayıtlar geriye dönük olarak incelenmiş ve iki önemli sorun tespit edilmiştir. Birinci sorun, tehlike anında çocukların hızlı yardım istasyonunu kullanmak yerine kaçmayı tercih etmesi ve tehlike olmadığında manuel butonu amacı dışında oyun olarak kullanmalarıdır. İkinci sorun ise çocuk oyun alanında aşılanmamış sahipsiz köpeklerin bulunmasıdır. Bu tez çalışması ile mevcut hızlı yardım istasyonunun köpekleri otomatik olarak tespit edecek şekilde geliştirilmesi neticesinde söz konusu sorunların giderilmesi amaçlanmıştır.

Köpekler, çocukları yetişkinlerden daha fazla ısırma eğilimindedir çünkü çocuklar göz seviyesindedir ve bu durum hayvan tarafından bir tehdit olarak algılanır. Bu yüzden bir çocuğun kafasından ya da boynundan ısırılma olasılığı daha fazladır. Çocuğun boynu ve kafasında meydana gelen bir ısırık ölümcül olabilir. Günümüze kadar çocuk oyun alanlarında birçok ısırma vakası görülmüş olup ısırılan çocuklarda ölüm, kalıcı fiziksel sakatlıklar ve duygusal travmalar meydana gelmiştir.

ABD Hastalık Kontrol ve Önleme Merkezi (CDC) verilerine göre ABD'de her yıl yaklaşık olarak 4,5 milyon köpek ısırma vakası görülmektedir. Bu da genel nüfusa

oranla 73 kişiden 1'inin köpek saldırısına uğradığını göstermektedir. Ayrıca, 2005 ile 2018 yılları arasında Amerika'da köpek ısırması nedeniyle 471 ölüm vakası görülmüştür [11].

Ankara'da 2005–2009 yılları arasında meydana gelen ve bildirimi yapılan toplam 25.480 hayvan ısırığının %79,1'inden köpekler sorumlu bulunmuştur [12].

2011 yılında yayınlanan ve 5 yıllık süreci kapsayan Eskişehir Osmangazi Üniversitesi Eğitim, Uygulama ve Araştırma Hastanesi kuduz riskli temas raporunda başvuran hastaların %71,6'sının köpek ısırıklarından şikayetçi olduğu belirtilmiştir. Isırıklar mevsimsel değişim gösterse de çocukların tatilde olduğu yaz aylarında daha fazla gözlenmiştir. Sorumlu köpeklerin %48,2'si sahipsizken, %51,8'i sahiplidir [13].

Bursa Yıldırım ilçesinde 2011-2012 yıllarına ait 648 kuduz riskli vakanın %50,1'i 0-19 yaş grubunda görülmüş, %39,8'i yaz mevsiminde bildirilmiş, %70,7'sinden köpekler sorumlu bulunmuştur. Sorumlu köpeklerin %75,9'u sahipsizken, %27,6'sı aşılanmamıştır [14].

Haydarpaşa Numune Hastanesi Acil Tıp Kliniği'nde 2017-2018 yıllarına ait 10.974 kuduz riskli vakanın %33,7'si 0-19 yaş grubunda görülmüş, %29,8'i yaz mevsiminde bildirilmiş, %35,2'sinden köpekler sorumlu bulunmuştur. Sorumlu köpeklerin %80,8'i sahipsiz, %19,2'si sahiplidir. Hastanenin hizmet verdiği ilçelerin sahil kesiminde olması, insanların yürüyüş ve spor yapmak için kullandığı parkurlarda çok sayıda sokak hayvanının olması, okul çağındaki çocukların yaz tatilinde olması, iyi havalarda park ve bahçelerde hayvanlarla karşılaşmanın sık olması yaz mevsimindeki temasları arttırdığını düşündürmüştür [15].

Dünya üzerinde 900 milyon köpek olduğu ve bu sayının %83'ünün sahipsiz köpeklerden oluştuğu tahmin edilmektedir [16]. Türkiye'de ise sahipli ve sahipsiz köpeklerin sayısı bilinmemekle birlikte sahipli hayvanlar yaşlanmaları, hasta olmaları, davranış bozuklukları, sahiplerinin farklı bir şehre veya daha küçük evlere taşınmaları, bakım maliyetleri ve zorlukları, günümüzde tüm dünyayı etkisi altına alarak yeni bir yaşam düzeni getiren Covid-19 pandemisi gibi nedenlerle sahipleri tarafından terkedilebilmekte ve bu durum halk sağlığı ve güvenliği problemi haline gelebilmektedir.

Türkiye'de bugüne kadar 37 bakteriyel, 13 mantar, 29 viral, 28 parazitik (3 trematod, 7 cestod, 10 nematod ve 8 protozoon) olmak üzere toplam 107 farklı zoonotik

enfeksiyon bildirilmiştir. Buna ilaveten 15 farklı artropod grubu ve 1 sülük enfestasyonu olmak üzere birçok ektoparazitik zoonozlar da günümüze kadar Türkiye'den bildirilmiştir [17].

Dünya Sağlık Örgütü verilerine göre pek çok vahşi hayvan, kedi, domuz ve diğer çiftlik hayvanlarından hastalık bulaşsa da hala kuduza bağlı ölümlerde en önemli kaynak köpeklerdir. Dünyada her yıl yaklaşık olarak 55.000 insana kuduz tanısı konulmakla birlikte ülkemiz halen kuduz enfeksiyonu yönünden endemik bir bölgedir [18]. Türkiye Halk Sağlığı Kurumu verilerine göre 2018 yılında ülke genelinde 283.185 kuduz riskli temas gerçekleşmiştir [19].

Bu tez çalışmasında nesne tespit algoritmaları karşılaştırmalı şekilde incelenmiş olup önerilen model için yüksek fps ve mAP değeri nedeniyle YOLOv3 algoritması tercih edilmiştir.

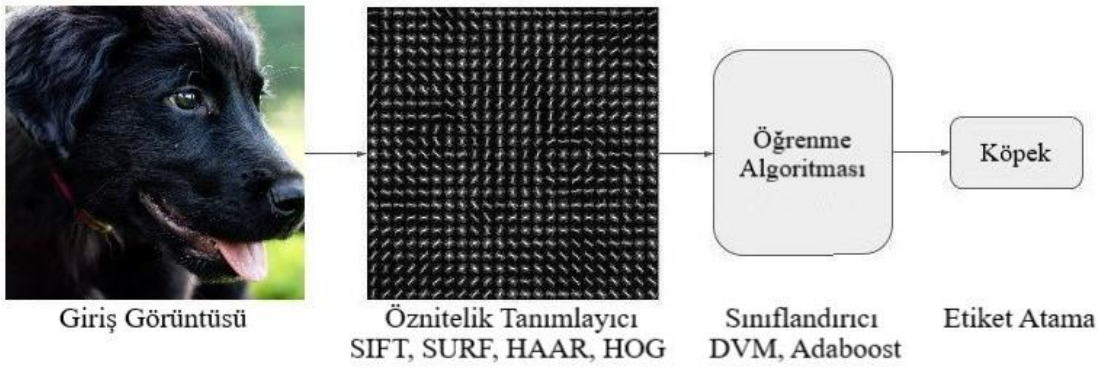
Öncelikle, orijinal eğitim veri setlerindeki sahipsiz köpek görüntü verisi eksikliğinin giderilmesi için bir hayvan barınağındaki sahipsiz köpeklerin görüntüleri çekilerek eğitim veri setine eklenmiştir. Sonrasında farklı hiperparametrelerle eğitimler gerçekleştirilmiş ve eğitim sonuçları birbirleriyle karşılaştırılmıştır. En iyi değerlendirme sonuçlarına sahip model çocuk oyun alanlarında gerçek zamanlı köpek tespiti için önerilmiş ve seçilen çocuk oyun alanında bulunan hızlı yardım istasyonuna entegre edilerek uygulanmıştır. Bu tez çalışmasında derin öğrenme temelli yeni bir akıllı şehir güvenlik uygulaması önerilmiş ve daha yüksek AP değeri hedeflenmiştir.

1.2 Literatür Araştırması

Nesne tespiti ve nesne tanımada amaç, daha önceden tanımlanmış sınıflara ait nesnelerin (araba, insan yüzü, köpek vb.) eğitilmiş bir algoritma ile hareketli veya sabit bir görüntüden gerçek zamanlı olarak yüksek hız ve doğrulukla tahmin edilmesidir.

1.2.1 Geleneksel nesne tespit algoritmaları

Geleneksel nesne tespit yöntemlerinde giriş görüntüsü üzerinde doğrudan nesne tespit işlemi gerçekleştirilemez. Veri girişi ve önişleme, öznitelik çıkarımı, sınıflandırma ve etiketleme olmak üzere 4 aşama takip edilmektedir. Geleneksel nesne tespit aşamaları Şekil 1.1'de gösterilmiştir.



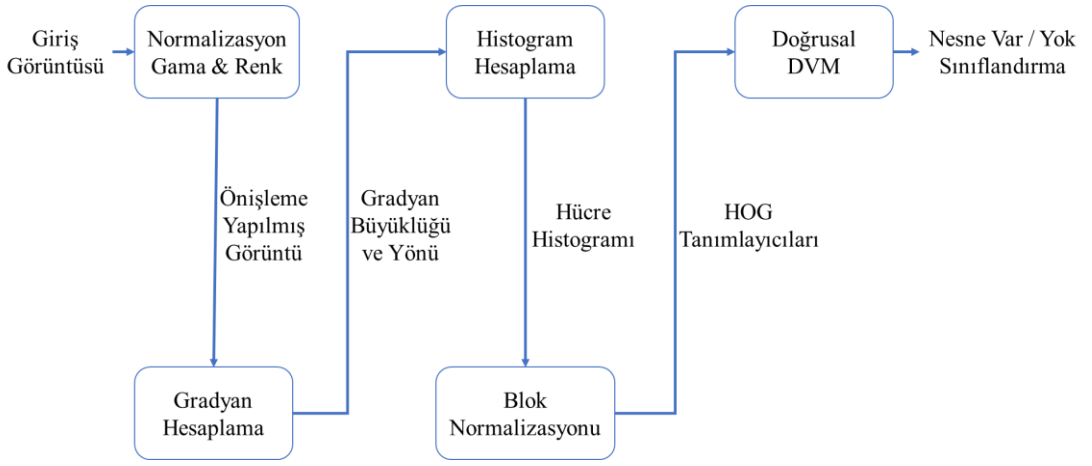
Şekil 1.1 : Geleneksel nesne tespit aşamaları

1.2.1.1 Veri girişi ve ön işleme

Görüntünün elde edilerek geçerli formatta matris haline getirilmesi ile görüntü üzerindeki gürültülerin temizlenmesi, görüntü keskinliğin arttırılması ve görüntünün öznitelik tanımlayıcısı tarafından belirtilen en boy oranı ve boyutlara getirilmesidir.

1.2.1.2 Öznitelik çıkarımı

Gereksiz bilgilerin atılarak görüntüyü en iyi şekilde tanımlayan öznitelik vektörünün elde edilmesi, ışık değişimlerine karşı bu vektörün normalize edilmesidir. Öznitelik çıkarımı öznitelik tanımlayıcıları tarafından gerçekleştirilir. Çok fazla sayıda öznitelik tanımlayıcısı olmakla birlikte SIFT[20], SURF[21], Haar Cascades[22] ve HOG[23] literatürde en çok kullanılmış olanlarıdır. HOG öznitelik tanımlayıcısının aşamaları Şekil 1.2'de gösterilmiştir.



Şekil 1.2 : HOG aşamaları **[24]**

HOG algoritması ile görüntü normalize edilerek ön işleme tabi tutulur, gama ve renk düzeltmeleri yapılır ve görüntü 1:2 en boy oranına getirilir. Yatay ve dikey gradyanlar

kernel kullanılarak hesaplanır. Gradyanların genlik ve açılarının bulunması için 1.1 ve 1.2 eşitlikleri kullanılır.

$$g = \sqrt{g_x^2 + g_y^2} \tag{1.1}$$

$$\theta = \tan^{-1}\frac{g_y}{g_x} \tag{1.2}$$

(g: gradyan genlik, θ:gradyan açısı)

X gradyanları yatay çizgileri y gradyanları ise dikey çizgileri daha belirgin hale getirir. Böylece sabit arka plan gibi gereksiz öznitelikler çıkartılmış olur. Her pikselin x ve y düzleminde bir gradyan genliği ve açısı bulunmaktadır. Gradyan genliği pikselin x ve y gradyan kanalındaki maksimum genlik değeri ile buna karşılık gelen açının alınması ile elde edilir.

Görüntü 8x8 hücrelere bölünür ve her bir 8x8 hücre için gradyan histogramları elde edilir. Bir 8x8 RGB görüntüde 8x8x3=192 piksel değeri bulunmakta iken bu görüntünün gradyanı genlik ve açı değerlerine sahip olduğundan 8x8x2=128 piksel değerine sahiptir. Bu şekilde görüntü daha kompakt ve gürültüye karşı daha dayanıklı hale gelmektedir.

Gradyan açıları diğer algoritmaların aksine 0-180 arasındadır. Görüntü içerisinde bulunan nesnelerin kenar ve köşelerinde gradyan genlik ve açı değerleri de yüksektir.

Açılara karşılık gelen genlik değerleri gradyan histogramına yazılır ve 9'lu histogram elde edilir.

Aydınlatmadan bağımsız bir öznitelik tanımlayıcı elde etmek için histogram normalize edilir. Hesaplama kolaylığı sağlaması açısından normalizasyon 16x16 hücrelerde yapılır.

64x128 bir görüntüde 16x16 hücre 7 yatay ve 15 dikey olmak üzere 7x15=105 farklı pozisyonda bulunabilir. Her bir 16x16 hücre 36x1 vektör ile tanımlanır. Böylece 36x105=3780 boyutlu vektör elde edilir. Dolayısıyla 64x128 bir RGB görüntü 64x128x3=24.576 piksel değerine sahipken yalnızca 3780 HOG boyutlu bir vektör ile tanımlanabilmektedir.

Öznitelik çıkarımı işleminin ardından sınıflandırma işlemi uygulanır.

1.2.1.3 Sınıflandırma

Öznitelik vektörünün çok fazla sayıda pozitif ve negatif örnek ile eğitilmiş makine öğrenmesi algoritmaları ile en verimli, en kesin ve en hızlı biçimde etiketlenmesi ve sınıflandırılmasıdır.

Temel sınıflandırıcılardan Naive Bayes modeli görsel nesne tespitinde k-en yakın komşu algoritmasının daha verimli çalışması için bu algoritmayla birlikte kullanılmıştır [25] [26].

Doğrusal regresyon modeli R-CNN algoritmasında konumlandırma probleminin çözülmesinde sınırlandırma kutularının yer tespitinde kullanılarak konumlandırma problemi büyük oranda çözülmüş ve mAP değeri 3-4 puan arttırılmıştır [27].

Destek vektör makineleri modeli yüz tanıma, karakter tanıma, el yazısı tanıma, tıbbi tahminleme alanlarında kullanılmıştır. Yüz ifadesi analizinde derin öğrenme modelleri ile kullanılan softmax aktivasyon fonksiyonu yerine sınıflandırma problemi için doğrusal DVM kullanılmasıyla %9,5 oranında iyileşme sağlanmıştır [28].

Karar ağaçları modeli gerçek zamanlı görsel nesne tanıma probleminde sınıflandırma işlemi için kullanılmıştır [29].

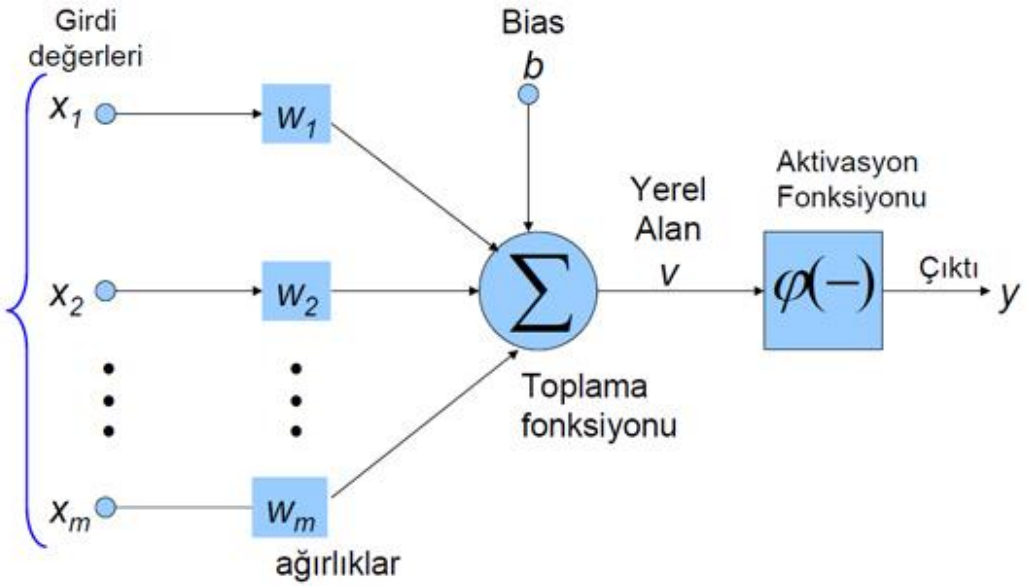
Artan ağaçlar modeli Orb öznitelik tanımlayıcıları ile birlikte nesne tespitinde de kullanılmıştır [30].

Rastgele ormanlar modeli çok sınıflı nesne tespitinde [31] ve insan vücut hareketlerinin takibi [32] amacıyla Microsoft Xbox'larda kullanılmıştır.

K-en yakın komşu modeli görsel nesne sınıflandırma alanında öznitelik kombinasyonunun oluşturularak daha verimli ve başarılı sınıflandırma yapılması amacıyla kullanılmıştır [33].

Yapay sinir ağları insan beyin yapısından esinlenerek geliştirilmiş, giriş verileri ile ilgili bilgileri toplayarak sürekli öğrenebilen, genellemeler yapabilen, öğrendiği bilgileri kullanarak yeni veriler hakkında kararlar verebilen bir bilgi işlem teknolojisidir [34].

Yapay sinir ağlarının en küçük parçası olan yapay sinir hücresi Şekil 1.3'teki gibi 5 temel elemandan (girdi değerleri, ağırlıklar, toplama fonksiyonu, aktivasyon fonksiyonu, çıktı) oluşmaktadır.



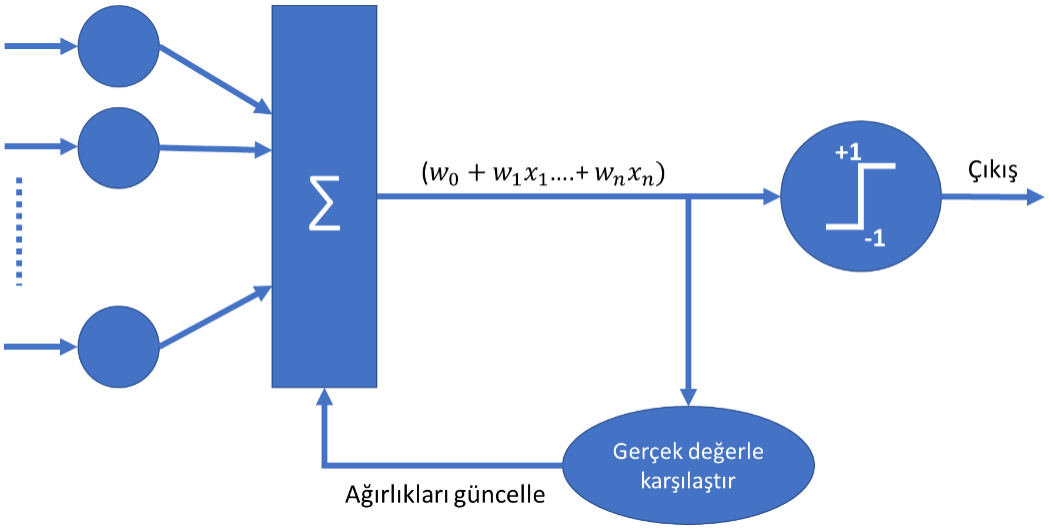
Şekil 1.3 : Yapay sinir hücresi yapısı

Tek katmanlı yapay sinir ağının kullanıldığı ilk gerçek problem olan telefon hatları üzerindeki gürültülerin yok edilmesinde en basit sinir ağı sınıflandırma modeli olan Adaline [35] (Adaptive Linear Element) algoritması kullanılmıştır.

Şekil 1.4'te mimarisi gösterilen Adaline algoritmasında her iterasyon için gerçek değer ile o anki çıktı değeri arasındaki hata hesaplanarak ağırlıklar güncellenir. Güncellenen yeni ağırlık 1.3 eşitliği kullanılarak hesaplanır.

$$w_{ij}(yeni) = w_{ij}(eski) + \alpha(t_j - y_j)x_i \tag{1.3}$$

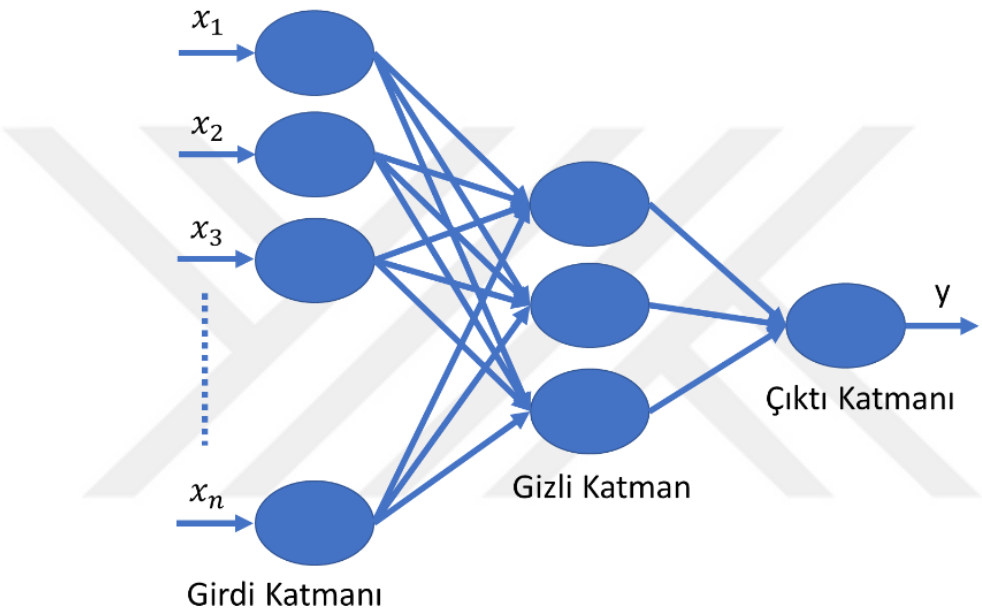
(w=ağırlıklar, α=öğrenme katsayısı, t_j=gerçek değer, y_j=çıkış değeri, x_i=giriş değeri)



Şekil 1.4 : Adaline mimarisi

Böylece her iterasyonda hata azaltılmış, gerçek değere daha çok yaklaşılmış olur. Adaline bir çok uygulama için kullanılsa da doğrusal olmayan çözümler üretemediğinden çok katmanlı yapay sinir ağları (MLP) önerilmiştir.

Şekil 1.5'te mimarisi verilen MLP, girdi katmanı ile çıktı katmanı arasındaki gizli katmana sahiptir. Gizli katmana sahip ve doğrusal olmayan aktivasyon fonksiyonu kullanan bir yapay sinir ağı ile giriş ve çıkış arasındaki tüm fonksiyonların tahmin edilebilmesi mümkündür. Gizli katman tipinin seçimine bağlı olarak yapay sinir ağları CNN, RNN vb. şekillerde isimlendirilir.



Şekil 1.5 : Çok katmanlı yapay sinir ağı mimarisi

MLP ileri beslemeli ve geri beslemeli yapay sinir ağları olarak ikiye ayrılmaktadır. İleri beslemeli ağlar yalnızca ileri yönde bilgi akışına izin verirken geri beslemeli ağlarda en az bir yapay sinir hücresinin çıkışı kendisine ya da farklı bir sinir hücresine genellikle bir geciktirme elemanı üzerinden giriş olarak verilir. Böylece geri beslemeli ağlarda her iki yönde de akış oluşturulur.

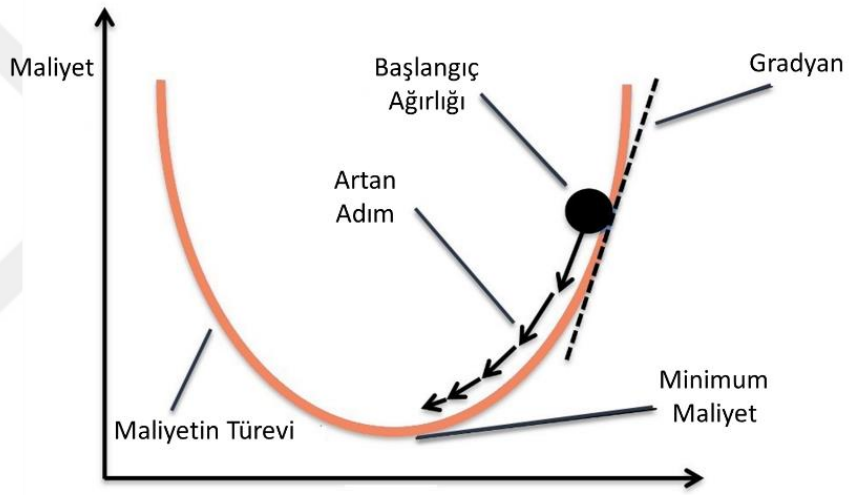
İleri beslemeli ağlarda girdi değerleri 1. gizli katmana iletilir, 1. gizli katmanın çıkışları 2. gizli katmana, 2. gizli katmanın çıkışları ise 3. gizli katmana giriş olarak verilir. Bu böylece devam eder ve çıktı katmanından önceki gizli katmanın çıkışları çıktı katmanında işlenerek yapay sinir ağının çıktı değerleri elde edilir. Toplam hata, çıktı değerleri ile hedef değerleri arasındaki hata fonksiyonu ile elde edilir. Regresyon problemlerinde genellikle 1.4'teki eşitlikteki ortalama karesel hata fonksiyonu kullanılır.

$$E = \frac{1}{n}\sum_{i=1}^{n}(t_j - y_j)^2 \tag{1.4}$$

(E: toplam hata, t_j: hedef değer, y_j: mevcut çıktı değeri)

Geri beslemeli ağlarda geri yayılım (backpropagation) [36] algoritması kullanılarak hata, çıkış katmanından gizli katmanlara iletilir ve gizli katmanların ağırlıkları güncellenir.

Hata fonksiyonunun minimumunu elde etmek için optimizasyon algoritmaları kullanılır. Optimizasyon algoritmalarından en yaygın kullanılanı Şekil 1.6'da gösterilen gradyan inişi algoritmasıdır. Bu algoritmada lokal minimuma yakınlaşmak için birinci türev kullanılır.



Şekil 1.6 : Gradyan iniş algoritması

Gradyan inişi algoritmasında 1.5 ve 1.6 eşitliklerindeki zincir kuralı kullanılarak wij ağırlığının toplam hataya olan etkisi hesaplanır.

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial o_j}\frac{\partial o_j}{\partial w_{ij}} = \frac{\partial E}{\partial o_j}\frac{\partial o_j}{\partial net_j}\frac{\partial net_j}{\partial w_{ij}} \tag{1.5}$$

$$\frac{\partial E}{\partial w_{ij}} = -(t_j - y_j)g'(h_j)x_i \tag{1.6}$$

(E: toplam hata, w: ağırlık, t_j: hedef değer, y_j: mevcut çıktı değeri, g(x): aktivasyon fonksiyonu, x_i: girdi değeri)

Bu değer ile öğrenme katsayısı çarpılarak ağırlığın değişim miktarı elde edilir ve değişim miktarının da mevcut ağırlığa eklenmesiyle yeni ağırlık elde edilmiş olur. 1.7 ve 1.8 eşitlikleriyle belirtilen bu öğrenme kuralına Delta kuralı da denir.

$$\Delta w_{ij} = -\alpha \frac{\partial E}{\partial w_{ij}} \tag{1.7}$$

$$w_{ij}^{yeni} = w_{ij}^{eski} + \Delta w_{ij} \tag{1.8}$$

(E: toplam hata, w: ağırlık, α=öğrenme katsayısı)

Öğrenme katsayısı büyük seçilirse daha az döngüde (epoch) lokal minimuma ulaşılabileceği gibi lokal minimuma erişememe problemi de görülebilir. Öğrenme katsayısı küçük seçilirse lokal minimuma erişme ihtimali yükseldiği gibi bu erişme daha fazla döngüde sağlanacaktır.

Yapay sinir ağlarının doğrusal olmayan çözümler üretebilmesi, eğitim sırasında verilen veya verilmeyen girdi ile çıktı arasındaki tüm fonksiyonları tahmin edebilmesi, kendi kendine öğrenebilmesi, basit işlemlerden oluşan hücrelerden meydana geldiğinden gerçeklenmesinin kolay olması, bu hücreler arasındaki görev paylaşımı ile karmaşık problemlerin kısımlara ayrılarak çözülebilmesi, değişen şartlara adaptasyonu, yapay sinir hücrelerinden birisinin zarar görmesi ve işlevsiz hale gelmesi durumunda dahi hataya karşı toleranslı olması avantajları iken, donanım bağımlı olması, ağ yapısının deneme yanılma yöntemi ile tasarlanması, ağ yapısının doğruluğunun kontrol edilebileceği bir mekanizmanın bulunmaması ve her zaman doğru karar verilmesinin garanti edilmemesi en büyük dezavantajlarıdır.

Yapay sinir ağları birçok alanda çözümü karmaşık ve zor olan problemlerin çözümünde kullanılmakta olup genellikle başarılı sonuçlar elde edilmiştir.

LeNet [37] ve Sınırlı Boltzman makinesi [38] ile popüler hale gelen yapay sinir ağlarının 2012'de AlexNet[39] tarafından ILSVRC 2012 yarışmasında kullanılması ile elde edilen büyük başarı neticesinde modern görüntü sınıflandırma tarihi başlamıştır.

1.2.2 Derin öğrenme temelli algoritmalar

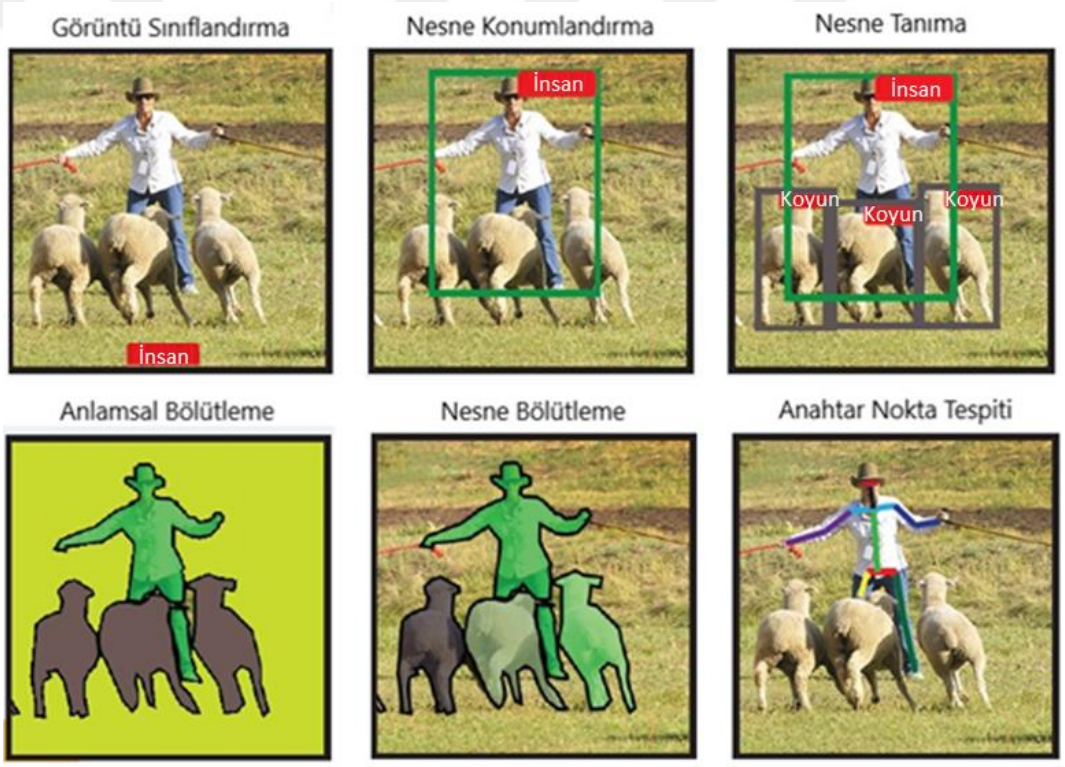
Geleneksel nesne tespit algoritmaları ile ham görüntü işlenemez. Öncelikle ham görüntüden öznitelik çıkarımı işlemi ile öznitelik vektörünün elde edilmesi

gerekmektedir. Bu işlem uzun zaman almakta ve kullanıcılar için iş yükü oluşturmaktadır.

Derin öğrenme temelli algoritmalar ise geleneksel algoritmaların aksine öznitelik çıkarımına ihtiyaç duymadan ham görüntüyü işleme yeteneğine sahiptir. Bilgisayar donanım altyapısının gelişmesi ve günlük hayatın dijitalleşerek daha çok verinin elde edilmesiyle birlikte derin öğrenme temelli algoritmalar tercih edilir olmuştur.

1.2.2.1 Tanımlar

Bilgisayar görüsü uygulamalarından görüntü sınıflandırma, nesne konumlandırma, nesne tanıma, anlamsal bölütleme, nesne bölütleme ve anahtar nokta tespiti işlemleri Şekil 1.7'de gösterilmiştir.



Şekil 1.7 : Tanımlar

Görüntü sınıflandırma, görüntünün, içerisinde yer alan dominant nesneye göre görüntünün sınıflandırılması işlemidir.

Nesne konumlandırma, görüntünün içerisinde yer alan dominant nesnenin bulunduğu görüntü parçasının tahmin edilmesidir. Sonrasında bu tahmin edilen parça içerisindeki görüntü sınıflandırılabilir.

Nesne tanıma, görüntü içerisindeki tüm nesnelerin konumlandırılması ve sınıflandırılmasıdır. Bu işlem genellikle görüntü parçasının önerilmesi ve bu önerilen bölge içerisindeki nesnenin sınıflandırılmasını içerir.

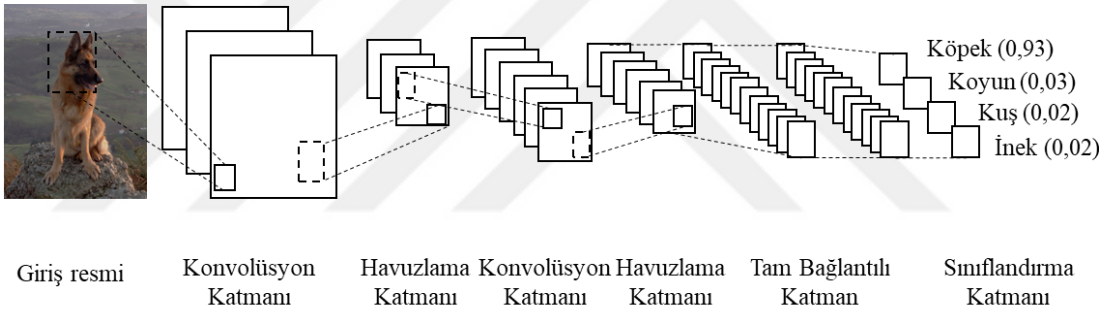
Anlamsal bölütleme, görüntünün tüm piksellerinin belirli sınıflara etiketlenmesidir.

Nesne bölütleme, görüntü içerisindeki her bir nesneye ait tüm piksellerin aynı sınıfa dahil başka bir nesneden de ayrılarak farklı olarak etiketlenmesidir.

Anahtar nokta tespiti, nesneye ait anahtar noktaların (insan yüzü, kolu, vücudu vb.) konumlarının tespit edilmesidir.

1.2.2.2 CNN (ESA) mimarisi

Tüm derin öğrenme temelli algoritmaların temel modeli evreşimli sinir ağlarıdır. CNN mimarisi Şekil 1.8'de gösterilmiş olup giriş, konvolüsyon, havuzlama (pooling), tam bağlantılı ve sınıflandırma katmanlarından oluşur.



Şekil 1.8 : CNN (ESA) mimarisi

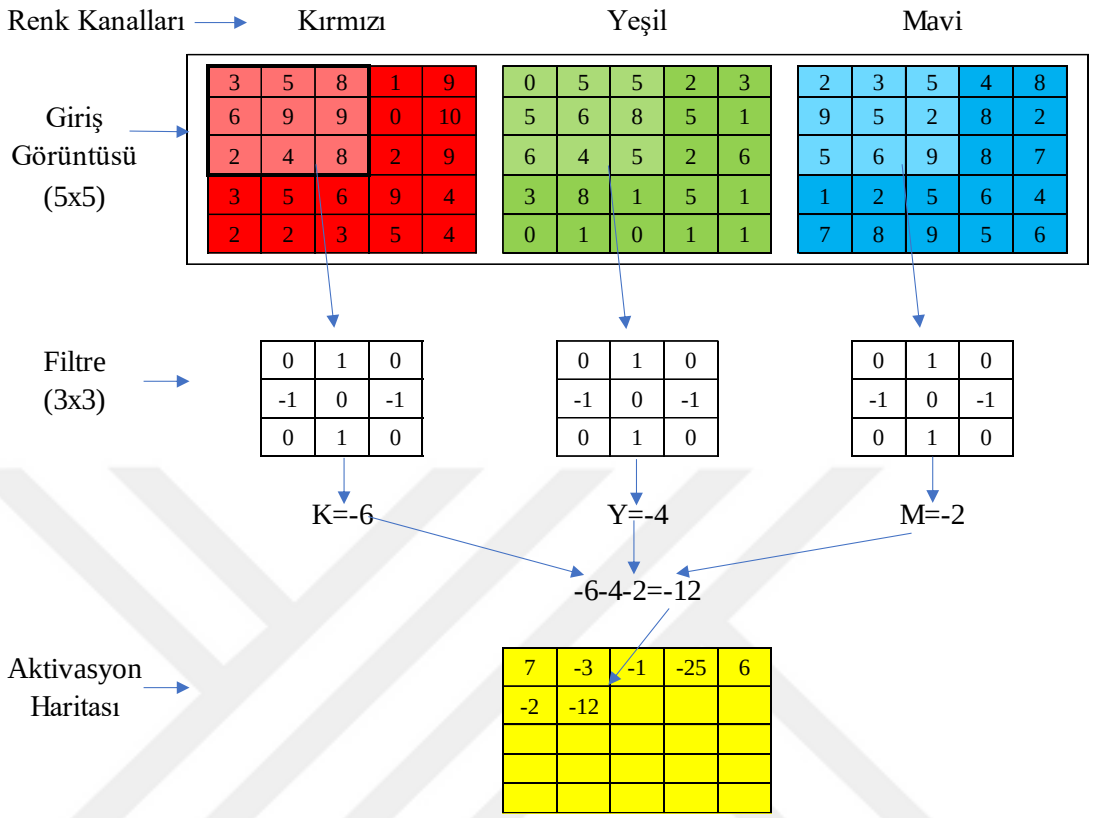
1.2.2.3 CNN katmanları

Girdi katmanı, görüntünün ham olarak ağa verildiği katmandır. Ağa verilecek görüntünün boyutu hem ağ derinliği hem de donanımsal yeterlilikler göz önüne alınarak belirlenir.

Konvolüsyon katmanı, belirli filtrelerin (kernel) bir önceki katmandan gelen görüntü üzerinde adım aralıkları ile (stride) gezdirilmesi yöntemiyle konvolüsyon işleminin uygulandığı dönüşüm katmanıdır. Filtreler 2x2, 3x3, 5x5 gibi farklı boyutlarda olabilir.

Konvolüsyon işlemi sonucunda aktivasyon haritası elde edilir. Aktivasyon haritası her bir filtreye ait özniteliklerin çıkarıldığı bölgedir. Şekil 1.9'da 5x5x3 boyutunda renkli

(RGB) bir görüntü üzerinde 3x3 bir filtre ile konvolüsyon işlemi neticesinde elde edilen 5x5 aktivasyon haritası gösterilmiştir.

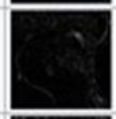
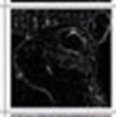


Şekil 1.9 : RGB girdi görüntüsüne konvolüsyon işleminin uygulanması

Girdi görüntüsü kenarlarına sıfır ilavesi (zero padding) yapılarak her bir pikselin eşit oranda konvolüsyona katılması ve girdi görüntüsü ile aktivasyon haritasının boyutlarının eşitlenmesi sağlanır. Yukarıdaki örnekte girdi görüntüsüne sıfır ilavesiyle 7x7x3 bir matris elde edilir ve aynı filtre ile konvolüsyon işlemi neticesinde girdi görüntüsü ile aynı boyutta 5x5 aktivasyon haritası elde edilir. Adım aralıkları yukarıdaki örnekteki şekilde 1 olabileceği gibi farklı aralıklarda da olabilir. Adım aralıklarına bağlı olarak eklenmesi gereken sıfır sayısı da değişecektir.

Farklı filtreler kullanılarak aktivasyon haritası dizisi elde edilir. Görüntü üzerine 32 adet filtre uygulanması durumunda 32 adet aktivasyon haritası elde edilecektir. Aktivasyon haritasındaki değerler filtre katsayılarının toplamına bölünerek normalize edilir. Yukarıdaki örnekte filtrenin katsayıları toplamı sıfır olduğundan bu işlem yapılmaz.

Şekil 1.10'da aktivasyon haritasının elde edilmesinde yaygın olarak kullanılan filtreler ve çıkış görüntüleri verilmiştir.

İşlem	Filtre	Çıkış Görüntüsü	İşlem	Filtre	Çıkış Görüntüsü
Belirleyici	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$		Bulanıklaştırma	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Kenar algılama	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$		3x3 Gauss Bulanıklaştırma	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$		5x5 Gauss Bulanıklaştırma	$\frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$		Keskin Olmayan Kısmı Maskele	$\frac{-1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & -476 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$	
Keskinleştirme	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$				

Şekil 1.10 : Yaygın kullanılan filtre tipleri

Konvolüsyon katmanında matematiksel işlemler yapıldığından ağ doğrusal bir yapıdadır. Genellikle öğrenilmek istenilen fonksiyon doğrusal olmadığından sigmoid, tanh ve ReLU gibi doğrusal olmayan aktivasyon fonksiyonları kullanılarak ağ doğrusal olmayan bir yapıya dönüştürülür. CNN'de her konvolüsyon katmanından sonra ReLU fonksiyonu uygulanarak ağ doğrusal olmayan bir yapıya dönüştürülür. ReLU'nun diğer aktivasyon fonksiyonlardan farkı satürasyona uğramadan daha hızlı öğrenme sağlamasıdır. Örnek girdi görüntüsünün konvolüsyon ve ReLU katmanı çıkışı Şekil 1.11'de gösterilmiştir.

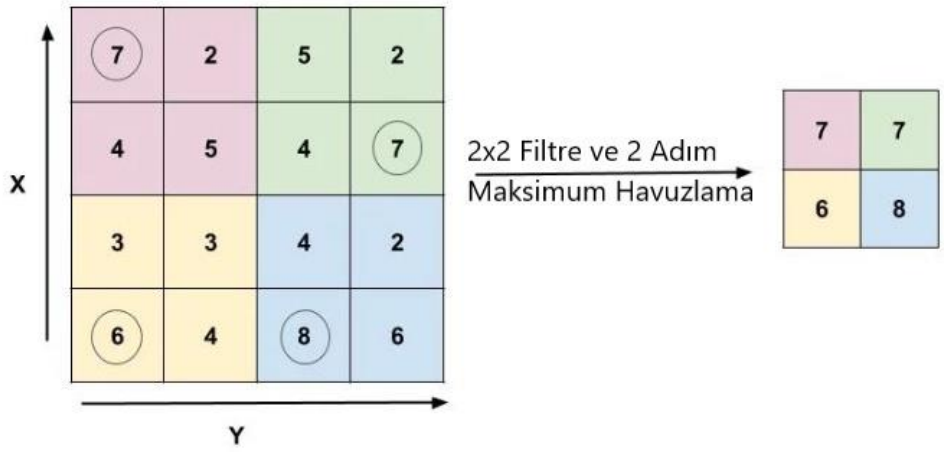


(a) **(b)** **(c)**

Şekil 1.11 : (a) Girdi görüntüsü (b) Konvolüsyon katmanı çıktı görüntüsü (c) ReLU katmanı çıktı görüntüsü

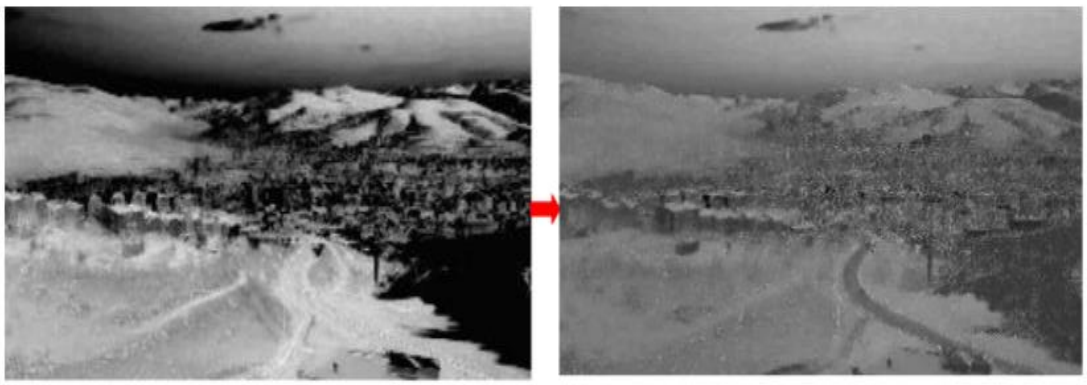
Havuzlama katmanı genellikle konvolüsyon katmanının hemen ardından görüntünün genişlik ve yükseklik boyutunun azaltılması için kullanılır. Bu katman sonrasında bilgi kaybı meydana gelir. Bilgi kaybı bir sonraki katman için hesaplamanın azaltılması ve ezbere öğrenmenin önlenmesi açısından faydalıdır. Havuzlama işleminde de konvolüsyona benzer şekilde belirli filtreler görüntü üzerinde belirli adım

aralıkları ile gezdirilerek görüntüdeki piksel değerlerinin maksimum değerleri veya değerlerin ortalaması hesaplanır. Daha iyi performans verdiği için genellikle 2 adım aralıklarla ve 2x2 filtre kullanılarak yapılan maksimum havuzlama tercih edilir. Şekil 1.12'de maksimum havuzlama işlemi neticesinde boyutun düşürüldüğü görülmektedir.



Şekil 1.12 : Maksimum havuzlama işlemi

Şekil 1.13'te maksimum havuzlama sonrası elde edilen görüntü verilmiştir.

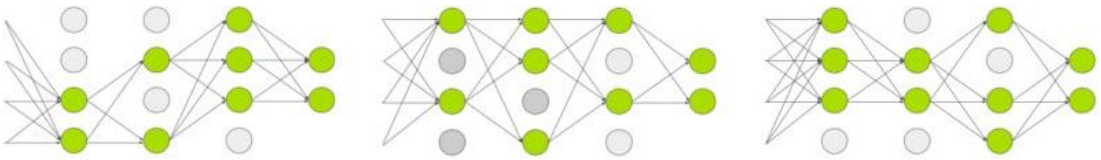


(a) **(b)**

Şekil 1.13 : (a) ReLu katmanı çıktı görüntüsü (b) Havuzlama katmanı çıktı görüntüsü

Tam bağlantılı (fully connected) katman kendisinden önceki katmanın tüm alanlarına bağlıdır. Katman sayısı ağın mimarisine göre farklılıklar gösterebilir. Bir önceki katmanın çıkış matris boyutu ile tam bağlantılı katmanın nöron sayısı çarpımı kadar boyutta ağırlık matrisi oluşur.

Ağın ezbere öğrenme probleminin çözümü için dropout [40] yöntemi kullanılır. Şekil 1.14'te gösterildiği şekilde eğitim sırasında her bir girdi için nöronlar belirli olasılıkla sistemden atılır, ileri ve geri yayılıma katılmaz. Böylece tüm girdiler farklı bir ağdan geçerek eğitim gerçekleştirildiğinden sistem ezbere öğrenmeye karşı daha dayanıklı olur.

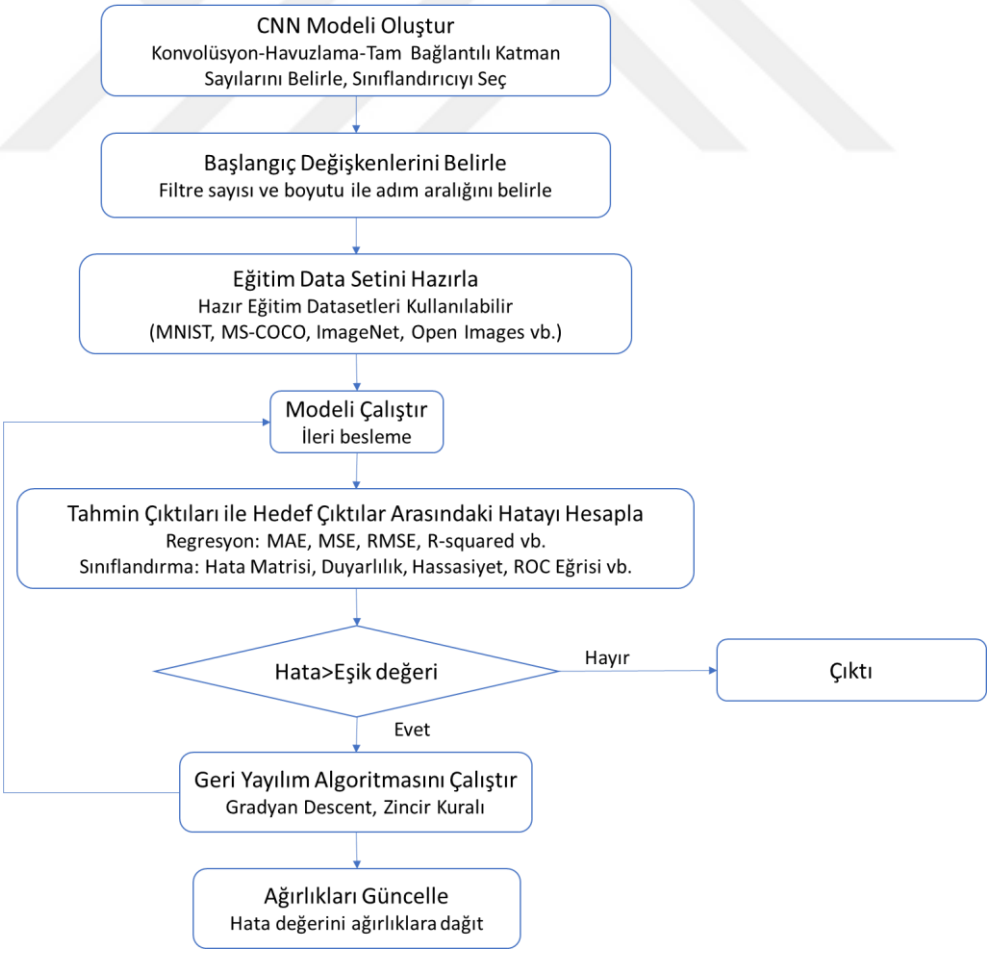


Şekil 1.14 : DropOut işlemi

Sınıflandırma katmanı, diğer katmanlarca elde edilen öznitelikler üzerinde doğrusal olmayan dönüşümlerin çalıştırılarak sınıflandırma işleminin yapıldığı katmandır. Farklı sınıflandırıcılar da kullanılabileceği gibi genellikle Softmax sınıflandırıcısı tercih edilir. Sınıflandırma yapılacak nesne sayısı kadar çıktıya sahiptir. Çıktılar 0 ile 1 aralığındadır ve toplamları 1'e eşittir. 1'e yakın olan çıktı ağın tahmin ettiği nesne olarak kabul edilir.

1.2.2.4 CNN eğitimi

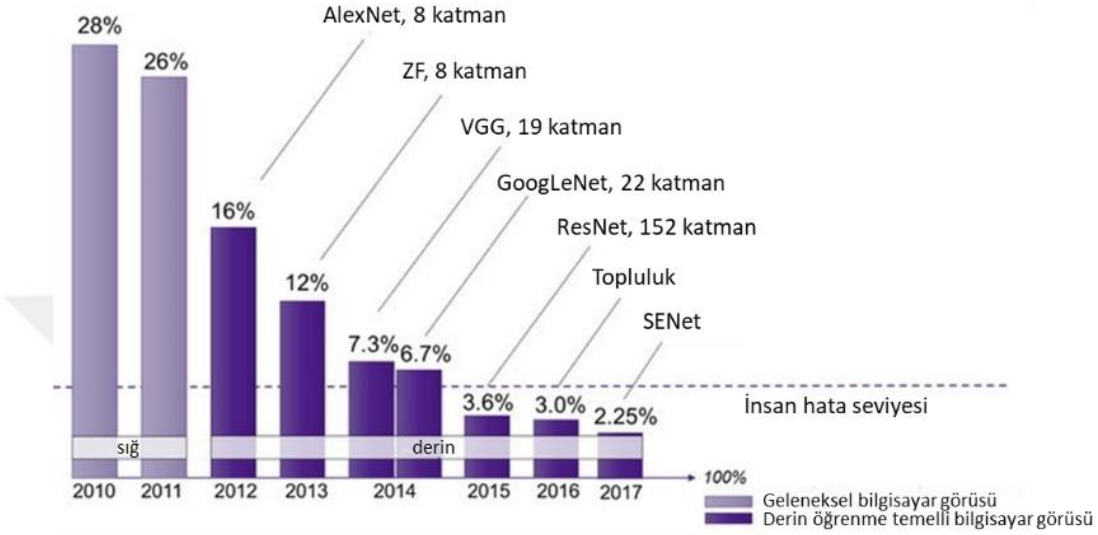
Evrişimli sinir ağı eğitim algoritması Şekil 1.15'te gösterilmiştir. Paralel işlemler neticesinde daha hızlı eğitim yapılabilir [41].



Şekil 1.15 : CNN eğitim algoritması

1.2.2.5 Derin öğrenme temelli görüntü sınıflandırma modelleri

AlexNet'in 2012 yılında ImageNet yarışmasını kazanmasının ardından yapılan yarışmalarda geleneksel yöntemler yerine derin öğrenme modelleri kullanılmaya başlanmış ve Şekil 1.16'dan de görüleceği gibi her yıl daha başarılı sonuçlar elde edilmiştir.



Şekil 1.16 : ImageNet yarışmasının kazananları

AlexNet 5 adet konvolüsyon, 3 adet havuzlama, 2 adet tam bağlantılı ve son olarak sınıflandırma katmanından oluşan mimarisiyle 1000 nesneyi %15,4 hata oranı ile sınıflandırmıştır.

AlexNet'in geliştirilmiş bir modeli olan ZF Net [42] mimarisinde ReLU aktivasyon fonksiyonu, Cross-Entropy Loss hata fonksiyonu ve eğitim için Stokastik Gradyan İnişi (SGD) kullanılmıştır. Havuzlama katmanında 2 adım aralığı ve ilk katmanda daha küçük boyutlu (7x7) filtreler kullanılarak girdi görüntüsünün orijinal piksel değerleri korunmuştur.

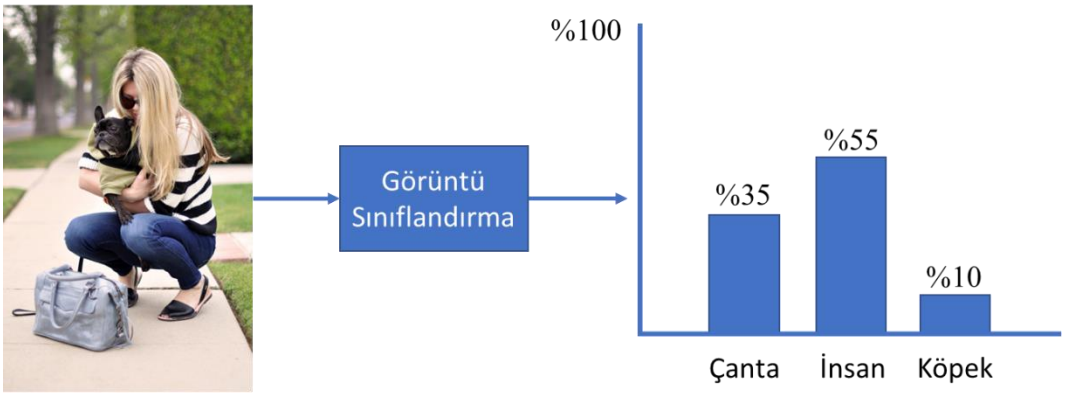
VGGNets'in 16 ve 19 ağırlık katmanlı VGG16 (13 konvolüsyon ve 3 tam bağlantılı katman) ve VGG19 (16 konvolüsyon ve 3 tam bağlantılı katman) versiyonları bulunmaktadır. Konvolüsyon katmanında 3x3 boyutlu filtreler ve havuzlama katmanında 2 adım aralığı kullanılmıştır. Her bir konvolüsyon katmanın ardından havuzlama katmanı gelmemekte sıralı konvolüsyon katmanları kullanılmaktadır. Havuzlama katmanında diğer modellerdeki gibi genişlik ve yükseklik küçülmekte farklı olarak derinlik gittikçe artmaktadır.

GoogLeNet [43] algoritması diğer ağ modellerinden farklı bir yapıya sahiptir. Ardışık konvolüsyon ve havuzlama katmanları yerine paralel olarak birbirine bağlı 9 adet inception modülü kullanılmış olup 22 katmandan oluşmaktadır. İnception modüllerinde farklı boyutlu konvolüsyon ve havuzlama işlemleri yapılmaktadır. Her konvolüsyon katmanından önce 1x1 ve düşük derinlikli konvolüsyon katmanı kullanılarak derinlik azaltılmıştır. Yığın normalleştirme (batch normalization) ve RMSprop kullanılarak hesaplanan parametre sayısı düşürülerek ağın hızı arttırılmıştır.

ResNet [44] diğer modellere göre daha derin şekilde tasarlanmıştır. 152 katmandan ve artık (residual) bloklardan oluşmaktadır. Doğrusal ve ReLU fonksiyonları arasına eklenen artık değer ile öğrenme hatası optimize edilir. İnsanlar görüntü sınıflandırmada uzmanlıklarına bağlı olarak %5 ile %10 arasında hata yapmaktadır. ResNet %3,6 hata oranını yakalayarak insandan daha iyi nesne tanıyan ilk model olmuştur.

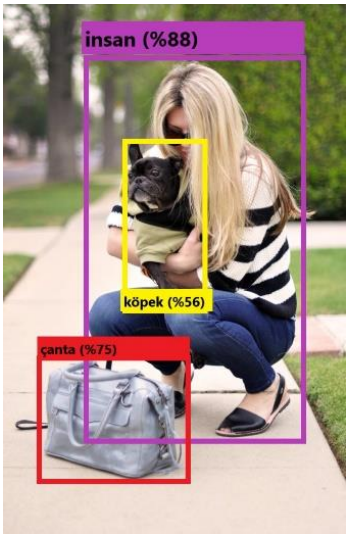
1.2.2.6 Derin öğrenme temelli bölge önermeli nesne tanıma modelleri

Görüntü sınıflandırma girdi görüntüsü içerisindeki dominant nesneye göre görüntünün sınıflandırılmasıdır. Şekil 1.17'deki görüntü için %55 skoruyla insan, %35 skoruyla çanta ve %10 skoruyla köpek tahminleri yapıldığından görüntü insan olarak sınıflandırılır.



Şekil 1.17 : Görüntü sınıflandırma işlemi

Nesne tanıma ise, girdi görüntüsü içerisindeki nesne bulunması muhtemel bölgelerin önerilmesi ve önerilen her bir bölgedeki tüm nesnelerin sınıflandırılmasıdır. Şekil 1.18'deki görüntüde içerisinde nesne bulunan üç adet sınırlayıcı kutu önerilmiş ve sınırlayıcı kutular içerisindeki insan, çanta ve köpek sırasıyla %88, %75 ve %56 güven skoruyla tanınmıştır.



Şekil 1.18: Nesne tanıma işlemi

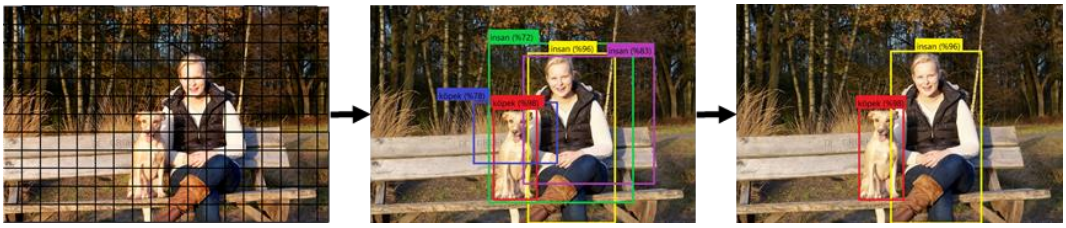
Nesne tanımada görüntü içerisindeki tüm nesneler için sınırlayıcı kutular önerilir. Sınırlayıcı kutu çizimi için sol üst köşenin veya nesnenin bulunduğu merkez noktanın (x,y) koordinatları ile yükseklik ve genişlik olmak üzere 4 farklı parametre gerekmektedir. Bu parametrelerin doğru şekilde tahmin edilmesi durumunda tahmin edilen sınırlayıcı kutu ile kesin referans (ground truth) sınırlayıcı kutusu büyük oranda çakışır ve yüksek IoU (Intersection over Union veya Jaccard indeksi) elde edilir. IoU değeri Şekil 1.19'daki eşitlikle hesaplanır.



Şekil 1.19: IoU hesaplanması

Görüntü üzerinde en iyi sınırlayıcı kutuların önerilmesi amacıyla öncelikle belirli güven skorunun altındaki tüm sınırlayıcı kutular elenir, Ardından en yüksek güven skoruna sahip sınırlayıcı kutular seçilirek bu kutularla belirli eşik değerin üzerinde IoU değeri ile çakışan diğer tüm kutular atılır. Bu işleme NMS (non max suppression) adı

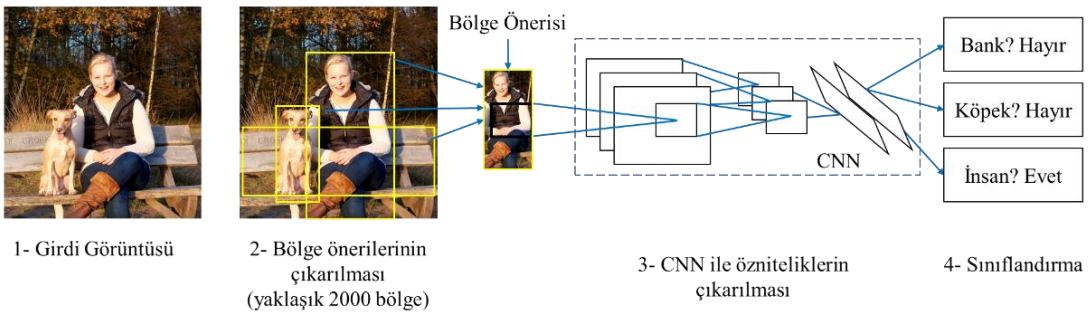
verilir. Şekil 1.20'de örnek görüntü üzerinde uygulanan NMS işlemiyle en iyi sınırlayıcı kutunun önerilmesi aşamaları gösterilmiştir.



Şekil 1.20: NMS işlemi

Bölge önermeli nesne tanıma algoritmaları önce nesne bulunması muhtemel alanları belirleyip ardından bu bölgelerde sınıflandırıcıyı çalıştırmaktadır. Bu algoritmalar her ne kadar doğru tahminler verse de görüntü üzerinde 2 sıralı işlem yapılmasından dolayı yavaştır.

R-CNN [27] (Regions with CNN features) sezgisel bölge önerimi metodu ile CNN'in bir kombinasyonudur. Girdi görüntüsü üzerinde seçici arama yöntemi ile nesne olma ihtimali bulunan yaklaşık 2000 adet bölge tespit edilir. Bu bölgeler sınırlayıcı kutu (bounding box) içerisine alınır ve görüntüden kesilir. Kesilen bölgeler 227x227 boyutuna getirilir ve her bir kesilen parça AlexNet benzeri bir CNN modeline (Caffe[45]) uygulanarak parçaların her biri için 4096 adet öznitelik çıkartılır. Bu öznitelikler ile nesne sınıflandırma yapılabilmesi için bir DVM modeli eğitilir. Sınırlayıcı kutu regresyon modeli eğitilerek sınırlayıcı kutuların 4096 öznitelikle beraber nesneyi daha iyi bir şekilde çevrelemesi sağlanır. R-CNN aşamaları Şekil 1.21'de gösterilmiştir.

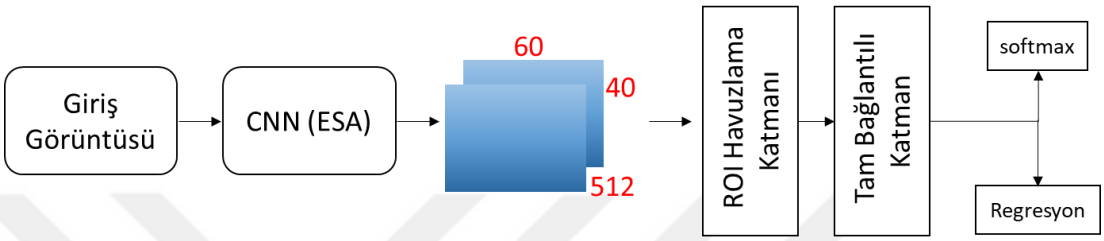


Şekil 1.21: R-CNN aşamaları

R-CNN test aşamasında her bir görüntüdeki her bir bölge önerisi için görüntü CNN'e girdi olarak verilir. Böylece her bir bölge önerisi için CNN toplam 2000 kez çalıştırılmış olur. Bu işlem çok büyük iş yüküne sebep olur ve uzun zaman alır. Ayrıca, öznitelik çıkartılması için CNN modelinin, sınırlayıcı kutuların düzenlenmesi için

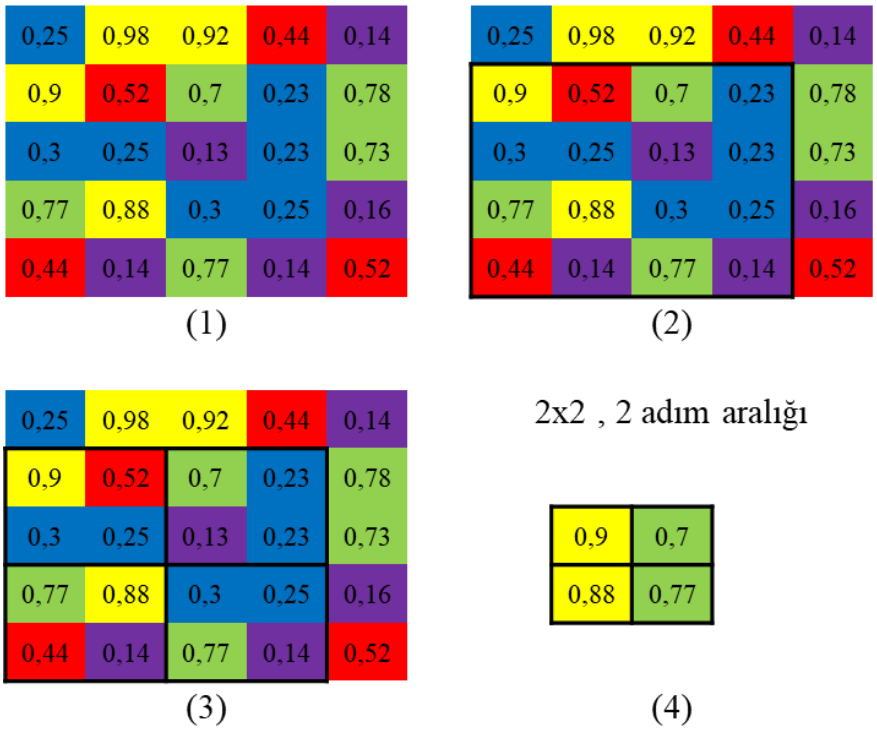
regresyon modelinin ve nesne sınıflarını tahmin etmek için sınıflandırıcının eğitimlerinin sıralı şekilde 3 aşamada yapılması da eğitim işlem zamanını uzatır. R-CNN üzerinde yapılan geliştirmeler ile iş yükü ve işlem zamanı problemleri Fast R-CNN ve Faster R-CNN modelleri ile giderilmeye çalışılmıştır.

Şekil 1.22'de mimarisi verilen Fast R-CNN [46] modelinde her bir görüntü için 2000 kez CNN çalıştırılması yerine 1 kez çalıştırılarak 2000 öneri arasında bu hesaplama paylaştırılır.



Şekil 1.22: Fast R-CNN aşamaları **[46]**

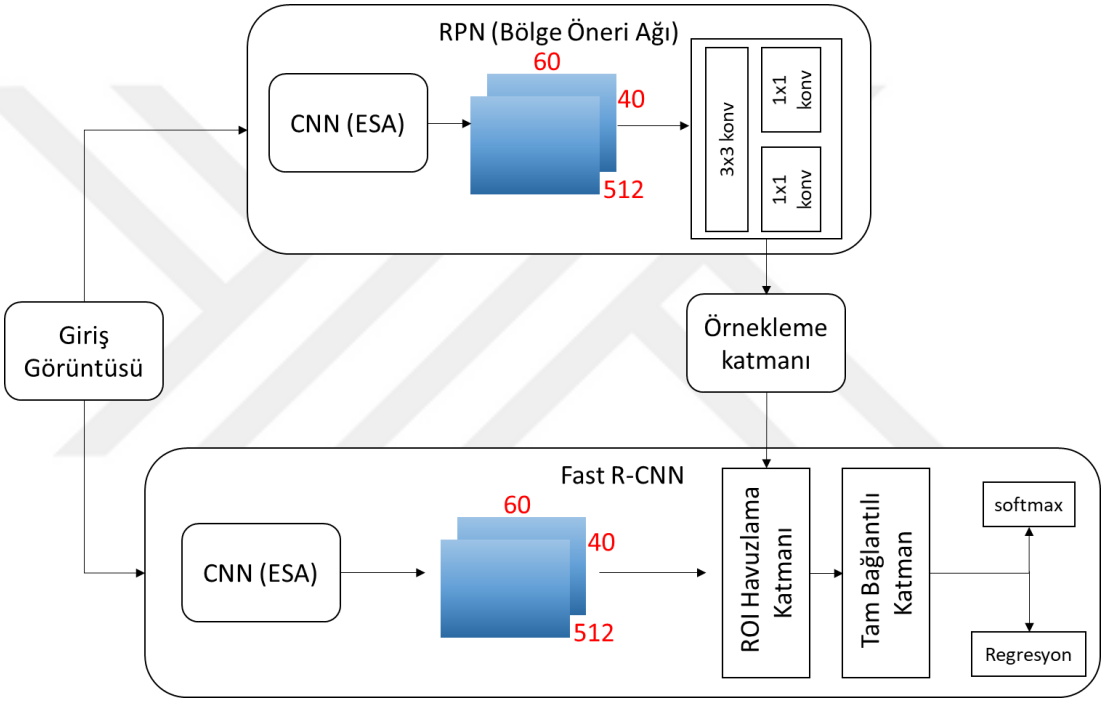
Fast R-CNN'de öncelikle görüntü CNN'e verilerek öznitelik haritası çıkartılır. Sonra her bir önerilen bölgeyle ilgili parça öznitelik haritasından çıkartılır ve bölge önermeli öznitelik haritası elde edilir. İlgili parça öznitelik haritasından kesilir ve havuzlama katmanı ile belirli boyutlarda sabitlenir. Bu havuzlama katmanına RoI (Region of Interest) havuzlama katmanı denir. RoI havuzlama işlemi Şekil 1.23'te gösterilmiştir.



Şekil 1.23: RoI havuzlama işlemi

Bölge önermeli öznitelik haritası öznitelik vektörüne dönüştürülerek ağın son kısmında yer alan ve 2 çıkışı bulunan tam bağlantılı katmana giriş olarak verilir. Birinci çıkış görüntü sınıflandırmanın yapıldığı Softmax çıkışı, diğer çıkış ise sınırlayıcı kutuların düzenlenmesi için regresyon çıkışıdır.

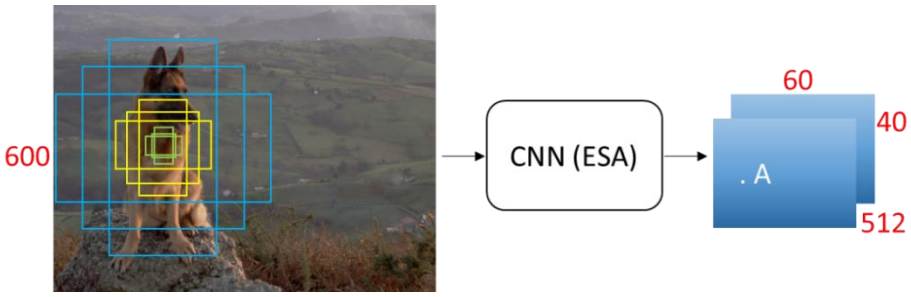
Faster R-CNN [47] modeli R-CNN ve Fast R-CNN'in karmaşık eğitim hattını düzenleyerek ve paralel işlem yapılarak işlem zamanında iyileştirme sağlamıştır. Faster R-CNN birbirine paralel bölge öneri ağı (RPN) ve Fast R-CNN ağından meydana gelmektedir. Şekil 1.24'te gösterildiği gibi Faster R-CNN ağında Fast R-CNN dedektör olarak kullanılmıştır.



Şekil 1.24: Faster R-CNN mimarisi **[47]**

RPN kısmında girdi görüntüsü omurga ağından geçirilerek öznitelik haritası elde edilir. RPN ağı öznitelik haritasının tüm noktalarında nesne bulunup bulunmadığının belirlenmesinden ve nesne var ise girdi görüntüsü üzerindeki bulunduğu alanın tahmin edilmesinden sorumludur. Bu işlem öznitelik haritasının her bir noktasına karşılık girdi görüntüsü üzerine farklı boyut ve en-boy oranlarında ankorlardan oluşan bir ankor grubunun ilave edilmesiyle yapılır.

Şekil 1.25'te öznitelik haritası üzerindeki A noktası için girdi görüntüsü üzerinde önerilen 3 farklı boyutta ve 1:1, 1:2, 2:1 en boy oranlarında toplam 9 adet ankor gösterilmiştir.



Şekil 1.25: Ankorlar ile bölge önerimi

RPN öznitelik haritasının her pikseli için k adet ankorun nesne içerip içermediğini kontrol eder ve ankorların boyut ve koordinatlarını revize ederek sınırlayıcı kutuları önerir. Bunun için RPN ağında öncelikle öznitelik haritasına 512 adet 3x3 filtre ile konvolüsyon işlemi yapılır, ardından paralel olarak sınıflandırma (ankorların nesne içerip içermediğinin kontrolü) için 18 boyutlu 1x1 konvolüsyon katmanından ve regresyon (nesnenin başlangıç x ve y koordinatları, genişlik, yükseklik) için 36 boyutlu 1x1 konvolüsyon katmanından geçirilerek sınırlayıcı kutu regresyonu için 60x40x36 (9 ankor için eşdeğeri 60x40x9x4), sınıflandırıcı için ise 60x40x18 (9 ankor için eşdeğeri 60x40x9x2) boyutlarında çıkışlar elde edilir. Ankor kutusu IoU değeri 0,7'den büyükse ve kesin referans için en büyük IoU değerine sahipse pozitif olarak, 0,3'den küçükse negatif olarak etiketlenir. 128 adet pozitif ve 128 adet negatif örnek rasgele seçilerek RPN eğitimi yapılır.

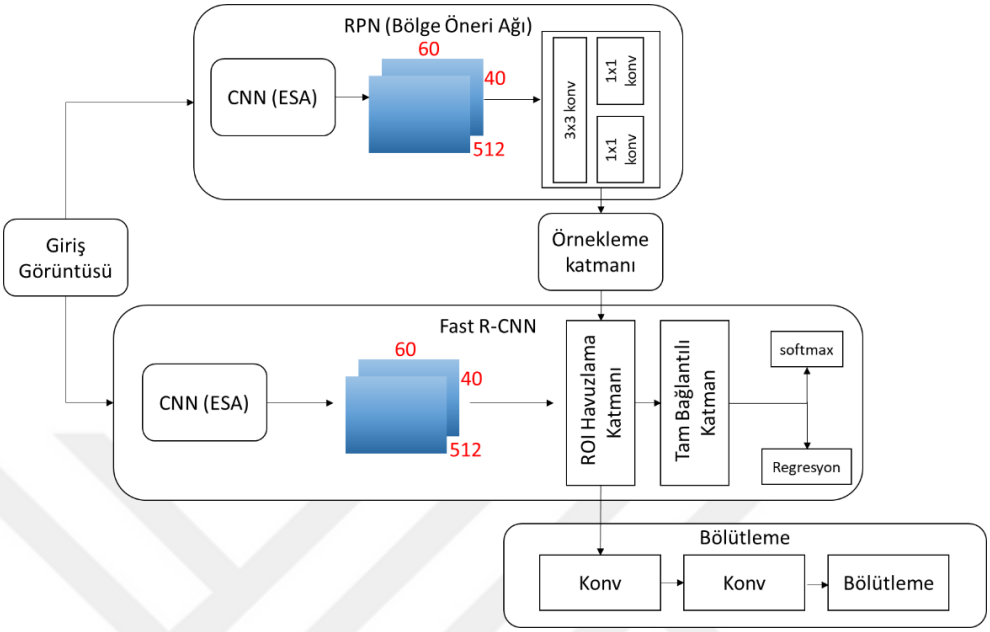
Ağın dedektör kısmında da RPN'den gelecek bölge önerilerinin çakıştırılması için öncelikle konvolüsyon işlemi yapılarak öznitelik haritası elde edilir. RoI havuzlama katmanından (N,7,7,512) boyutlu bölge önermeli öznitelik haritası elde edilir. N sayısı RPN tarafından önerilen sınırlayıcı kutu sayısıdır. Bölge önermeli öznitelik haritası öznitelik vektörüne dönüştürülerek ağın son kısmında yer alan ve 2 çıkışı bulunan tam bağlantılı katmana giriş olarak verilir. Regresyon çıkışı sınırlayıcı kutu tahminlerinin iyileştirilmesi için sınıflandırma çıkışı ise sınıflandırma tahminleri için kullanılır.

Bölge önermeli ağlarda bu geliştirmelerle birlikte işlem hızı 250 kat artmıştır. PASCAL VOC veri seti üzerinde çalıştırılan R-CNN, Fast R-CNN ve Faster R-CNN modellerinin sonuçları Çizelge 1.1'de gösterilmiştir.

Çizelge 1.1: Pascal VOC veri setinde R-CNN, Fast R-CNN ve Faster R-CNN testi.

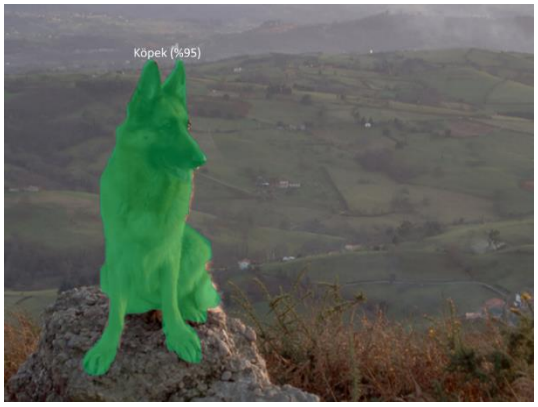
	R-CNN	Fast R-CNN	Faster R-CNN
Görüntü başına test süreleri	50 sn.	2 sn.	0,2 sn.
Hızlanma	1x	25x	250x
Ortalama tahmin (%)	66	66,9	66,9

Mask R-CNN [48] algoritması Faster R-CNN'in geliştirilmiş bir modelidir ve Şekil 1.26'da gösterildiği gibi sınıflama ve regresyonun yanı sıra bölütleme işlemini de gerçekleştirir.



Şekil 1.26: Mask R-CNN Algoritması **[48]**

Bu modelde sınırlayıcı kutular ile nesne tespiti ve konumlandırılması yapılırken, anlamsal bölütleme (semantik segmentasyon) ile girdi görüntüsüne ait tüm pikseller bir sınıfa etiketlenir. Faster R-CNN'e ek olarak RoI haritalarında bölütleme tahmini yapılır. Mask R-CNN sınıf etiketi, sınırlayıcı kutu ve nesne bölütleme olmak üzere 3 adet çıkış üretir. Mask R-CNN çıkışında Şekil 1.27'dekine benzer şekilde bölütleme görüntüsü elde edilir.



Şekil 1.27: Mask R-CNN Çıkış Görüntüsü

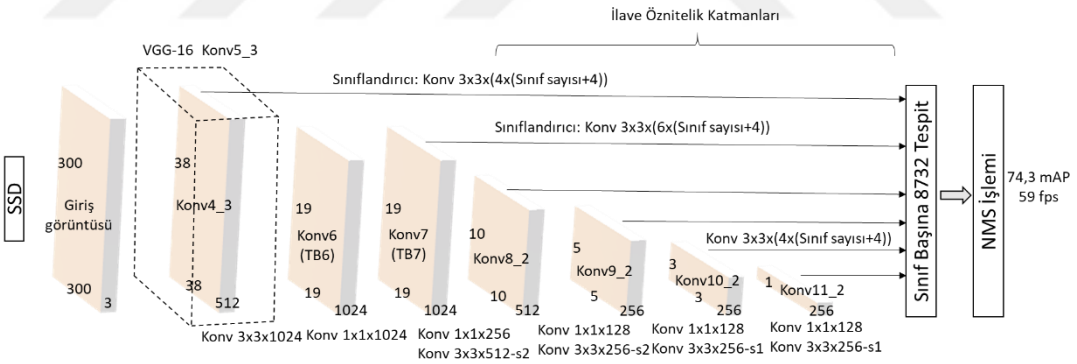
Overfeat [49] algoritması AlexNet'in geliştirilmiş bir modelidir. CNN önce sınıflandırıcı olarak eğitilmiş daha sonra softmax katmanı silinip yerine regresyon katmanı eklenerek nesnenin konumu tahmin edilmeye çalışılmıştır. Ağın

sınıflandırma, konumlandırma ve tespit için eğitilmesi ile daha başarılı olacağı önerilmiştir.

1.2.2.7 Derin öğrenme temelli tek aşamalı nesne tanıma modelleri

Tek aşamalı nesne tanıma modelleri görüntüyü tek bir seferde regresyon işlemine tabi tutarak nesnelerin sınıfını ve konumlarını aynı anda tahmin ettiklerinden bölge önermeli modellerden daha hızlı ve gerçek zamanlı uygulamalar için kullanımı daha uygundur.

Şekil 1.28'de mimarisi verilen SSD algoritması [50] VGG-16 mimarisine dayanmaktadır. VGG tam bağlantılı katmanları yerine ağın çıkışında konvolüsyon katmanları kullanılmıştır. Eklenen konvolüsyon katmanları farklı boyutlardaki özniteliklerin çıkartılmasını ve aşamalı olarak girdi boyutunun azaltılmasına olanak sağlar. Faster R-CNN'deki RPN ankorlarına benzer şekilde sınırlayıcı kutular kullanılır. Ancak bu kutuların farklı çözünürlükteki öznitelik haritalarına uygulanmasıyla kutu şekilleri daha verimli şekilde oluşturulabilmektedir. SSD algoritmasının VOC2007 veri seti üzerinde yapılan testlerinde yüksek hız (59 fps) ve mAP değerine (%87,32) ulaştığı görülmüştür.



Şekil 1.28: SSD mimarisi **[50]**

1.2.3 YOLOv3 nesne tanıma algoritması

YOLO (You Only Look Once) [51] algoritması tek aşamalı nesne tanıma algoritmalarındandır. Çoklu kutu kullanarak birden çok nesneyi gerçek zamanlı olarak yüksek frame hızlarında tanıyabilmektedir.

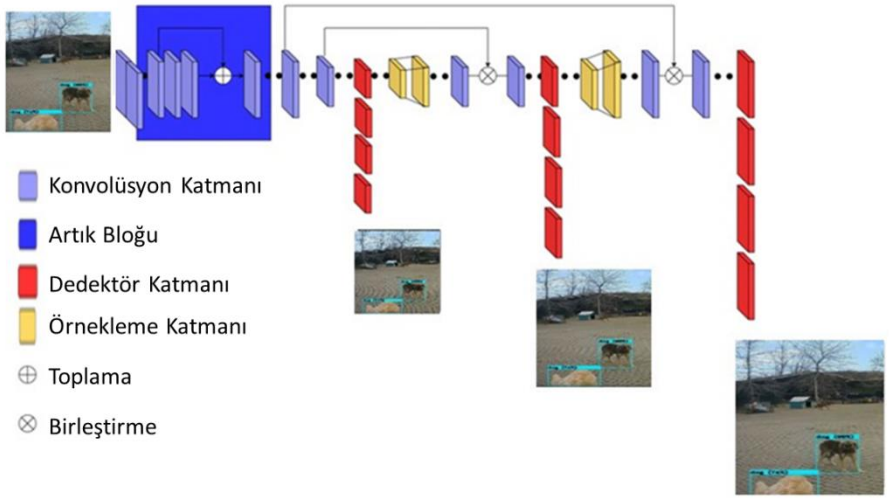
YOLOv3 [52] ağı Darknet-53 öznitelik tanımlayıcısı ve YOLO dedektörü olmak üzere iki kısımdan oluşmaktadır. Darknet-53 ImageNet'te eğitilmiş bir konvolüsyon ağı olup

toplam 53 adet ardışık 1x1 ve 3x3 boyutlarındaki konvolüsyon katmanını takip eden residual (artık) katmanlardan oluşur. Darknet-53 ağ yapısı Şekil 1.29'da gösterilmiştir.

	Katman	Filtre	Boyut	Çıktı
	Girdi Görüntüsü			416x416
	Konvolüsyon	32	3x3 / 1	416x416
	Konvolüsyon	64	3x3 / 2	208x208
	Konvolüsyon	32	1x1 / 1	208x208
1x	Konvolüsyon	64	3x3 / 1	208x208
	Artık			208x208
	Konvolüsyon	128	3x3 / 2	104x104
	Konvolüsyon	64	1x1 / 1	104x104
2x	Konvolüsyon	128	3x3 / 1	104x104
	Artık			104x104
	Konvolüsyon	256	3x3 / 2	52x52
	Konvolüsyon	128	1x1 / 1	52x52
8x	Konvolüsyon	256	3x3 / 1	52x52
	Artık			**52x52**
	Konvolüsyon	512	3x3 / 2	26x26
	Konvolüsyon	256	1x1 / 1	26x26
8x	Konvolüsyon	512	3x3 / 1	26x26
	Artık			**26x26**
	Konvolüsyon	1024	3x3 / 2	13x13
	Konvolüsyon	512	1x1 / 1	13x13
4x	Konvolüsyon	1024	3x3 / 1	13x13
	Artık			**13x13**
	Havuzlama			
	Tam Bağlantılı			
	Softmax			

Şekil 1.29: Darknet-53 mimarisi **[52]**

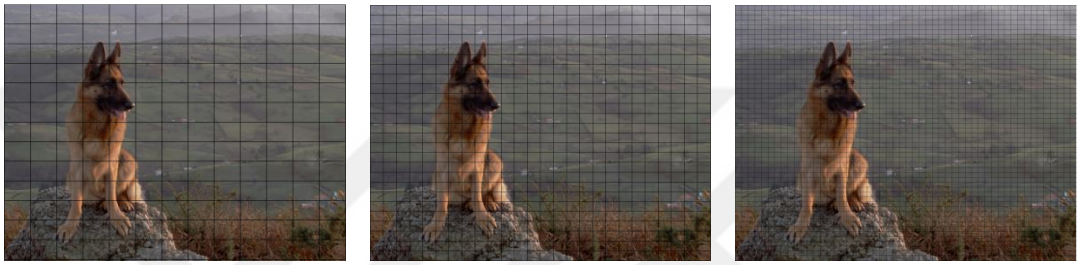
YOLOv3 algoritmasında Darknet-53 ağının öznitelik tanımlayıcısı olarak kullanılabilmesi amacıyla ağın son kısmında yer alan havuzlama, tam bağlantılı ve softmax katmanları Şekil 1.30'da gösterildiği gibi kaldırılmıştır. YOLOv3 ağı çok ölçekli dedektör özelliğindedir ve farklı boyutlardaki özniteliklere ihtiyaç duyar. Bu nedenle son 3 artık bloğu dedektör katmanlarına bağlanır. Her iki konvolüsyon katmanının girişi ile çıkışı arasına kısayol bağlantısı yapılarak artık değer ile öğrenme hatası optimize edilir.



Şekil 1.30: YOLOv3 mimarisi **[53]**

YOLOv3 algoritmasında ankorların algoritmaya eklenmesiyle ilk versiyonlardaki bağlı katmanlar mimariden kaldırılmıştır. Ağ yapısı 75'i konvolüsyon olmak üzere toplam 106 katmandan oluşur. Orijinal resim ağ tarafından 320x320x3, 416x416x3, 608x608x3 boyutuna getirilerek işlem yapılır.

YOLOv3 farklı boyutlardaki nesneleri algılamak için görüntüyü Şekil 1.31'deki gibi 3 farklı ölçekte ızgaralara böler. Girdi görüntüsü boyutunun 416x416 olması durumunda dedektör katmanlarına bağlanan vektör boyutları 13x13, 26x26 ve 52x52 olacaktır. 13x13'lük ızgaralar büyük, 26x26'lık ızgaralar orta ve 52x52'lik ızgaralar küçük boyutlu nesnelerin algılanmasını sağlar.

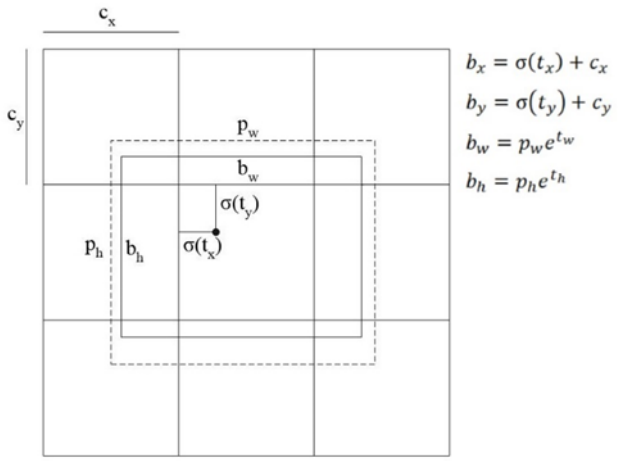


13x13 26x26 52x52

Şekil 1.31: Izgaralar ile bölge önerisi

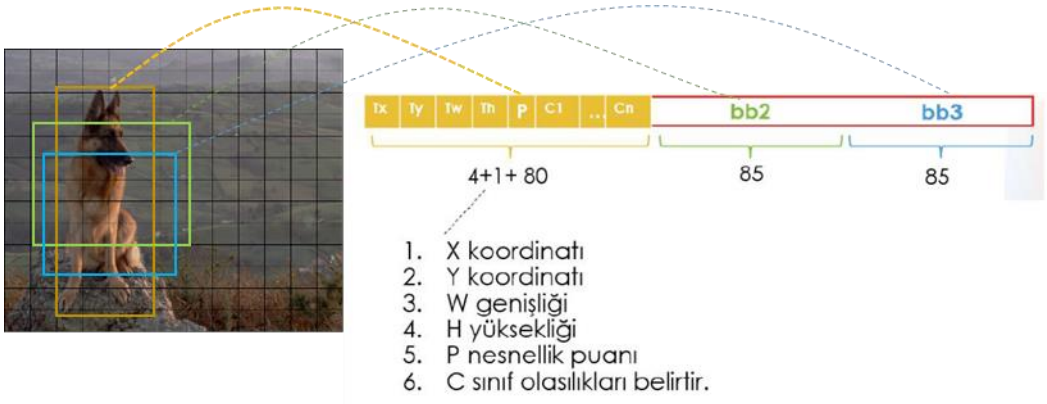
3 farklı ölçekteki (13x13, 26x26, 52x52) her bir ızgara hücresi 3 ankor kullanarak 3 farklı tahmin gerçekleştirebilir. Bu ankorlar çeşitli sınıflara ait farklı boyutlardaki nesnelerin tespit edilebilmesi amacıyla veri setleri üzerinde yapılan eğitimler sonucunda şekillenir. Örneğin, görüntü içerisindeki insanı tespit eden ankorlar dikey dikdörtgene benzer bir şekildeyken, köpeği tespit eden ankorlar yatay dikdörtgene benzer bir şekildedir.

YOLOv3 algoritmasında ankorların elle seçilmesinin yerine daha iyi ankorlar bulunması amacıyla veri seti üzerinde k-ortalama kümeleme yöntemi kullanılır. Sınır kutusunun genişliği ve yüksekliği (b_w, b_h) ile sınır kutusunun merkez koordinatları (b_x, b_y) Şekil 1.32'de verilen eşitliklerle hesaplanır. Sigmoid fonksiyonu merkez koordinat değerlerini (t_x, t_y) 0-1 aralığına düşürür. Kümelenme merkez noktasının, ızgara hücresinin sol üst köşesine olan uzaklığı $\sigma(t_x)$ ve $\sigma(t_y)$ ile hesaplanır. Bu değerlere ızgara hücresinin sol üst köşesinin, görüntünün sol üst köşesine olan uzaklıkları (c_x, c_y) eklenerek sınır kutusunun merkez koordinatları bulunur (b_x, b_y). Daha önceden tanımlanmış ankor boyutları (p_w, p_h) kullanılarak sınır kutusunun genişliği ve yüksekliği (b_w, b_h) hesaplanır. Negatif t_x ve t_y değerleriyle karşılaşma olasılığı göz önüne alınarak e^{tw} , e^{th} işlemiyle değer pozitif yapılır.



Şekil 1.32: YOLOv3 ile sınır kutusu tahmini **[52]**

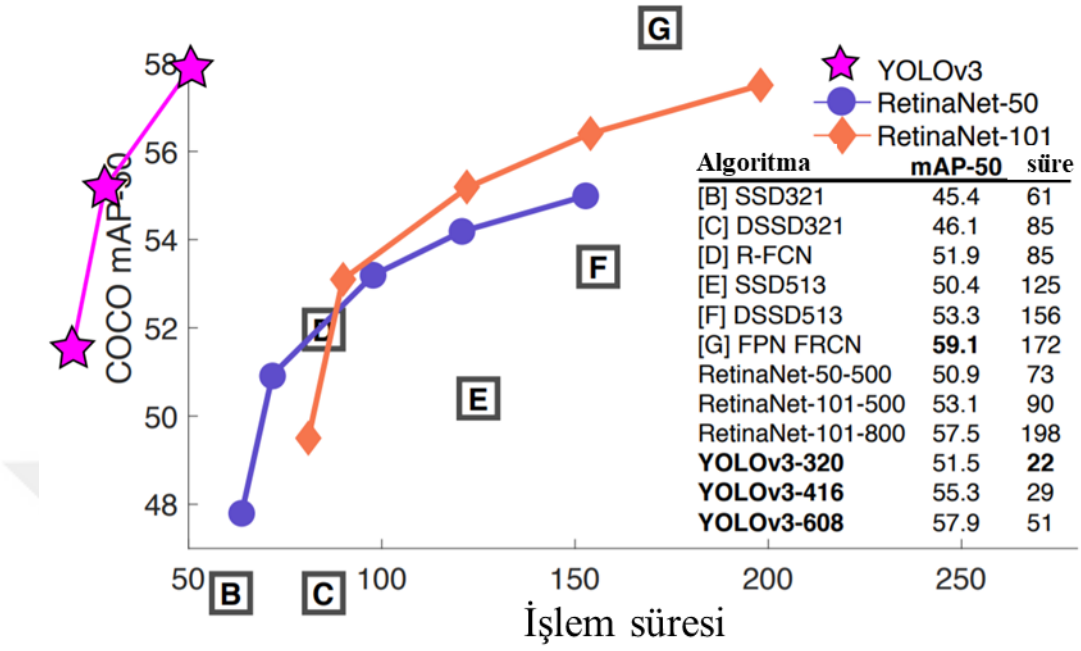
YOLOv3 ağı her bir ankor için güven skoru ve sınıf olasılığı tahmininde bulunur. Güven skoru sınırlayıcı kutunun nesne içerip içermediğini, sınıf olasılığı ise bu nesnenin hangi sınıfa ait olduğunu gösterir. Görüntüdeki her bir nesnenin tüm sınıflara ait olma olasılığı tek tek hesaplanır. Orijinal YOLOv3 algortiması, COCO [54] veri setini kullanmakta olup COCO veri setinde toplam 80 adet sınıf, 328.000 görüntü ve 2.500.000 nesne bulunmaktadır. Sonuç olarak YOLOv3 çıkışında 80 adet sınıf olasılığı tahmin edilmektedir. Her bir sınır kutusunun (x,y) merkez noktası ve (w,h) genişlik ve yüksekliği ile güven skorunun da dahil edilmesiyle her bir sınır kutusu için toplam 85 adet değer tutulur ve 3 farklı ızgara boyutu için 85x3 her bir hücrenin derinliği ise Şekil 1.33'te gösterildiği gibi 255 boyutlu olacaktır.



Şekil 1.33: YOLOv3 çıkışındaki hücre derinliği

3 farklı ızgara boyutunda toplam ((52x52)+(26x26)+(13x13))x3=10.647 adet sınır kutu tahmini yapılacak olup bu sınırlayıcı kutuların büyük bir kısmı çok düşük güven skoruna sahip olacaktır. NMS yöntemiyle bunlardan güven skoru ve IoU değeri eşik değerin (genellikle 0,5) altında olanlar elenir.

Şekil 1.34'ten de görüleceği üzere YOLOv3 COCO veri setinde 0,5 IoU (mAP-50) ile diğer algoritmalarla kıyaslanmasında özellikle süre açısından çok başarılı durumdadır.



Algoritma	mAP-50	süre
[B] SSD321	45.4	61
[C] DSSD321	46.1	85
[D] R-FCN	51.9	85
[E] SSD513	50.4	125
[F] DSSD513	53.3	156
[G] FPN FRCN	**59.1**	172
RetinaNet-50-500	50.9	73
RetinaNet-101-500	53.1	90
RetinaNet-101-800	57.5	198
YOLOv3-320	51.5	**22**
YOLOv3-416	55.3	29
YOLOv3-608	57.9	51

Şekil 1.34: Derin öğrenme algoritmalarının COCO veri setinde performansları **[52]**

Güvenlik sistemlerindeki gerçek zamanlı ve anlık tepki verilmesi ihtiyacı, çocuk oyun alanlarındaki çocuk ve köpeklerin hızlı ve hareketli yapıları ile YOLOv3 ağının tehlikeli nesne tespiti amacıyla kullanımının sınırlı olması nedeniyle bu tez kapsamında yapılan çalışmada YOLOv3 algoritması kullanılmıştır.

2. MATERYAL VE YÖNTEM

3.1 Giriş

Bursa Büyükşehir Belediyesi sorumluluğunda bulunan Hüdavendigar Kent Parkı, Botanik Park ve Kültürpark'ta hayata geçirilen Şekil 2.1'deki hızlı yardım istasyonu uygulaması ile çocuk parkları 24 saat boyunca kameralarla takip edilmekte, herhangi bir olumsuzlukta kameranın monte edildiği direk üzerindeki butona basılması halinde güvenlik noktası ile karşılıklı sesli ve görüntülü görüşme başlatılmakta, tehlikenin boyutuna göre bölgeye anında güvenlik birimleri sevk edilmektedir.



Şekil 2.1: Hızlı yardım istasyonu uygulaması **[10]**

Bu tez çalışması kapsamında söz konusu uygulamanın geliştirilerek olası tehlike oluşturabilecek nesnelerden köpeklerin manuel butonlar yerine kameralar ve YOLOv3 algoritması kullanılarak tespiti amaçlanmıştır. Bunun için öncelikle mevcut işletim sistemi üzerine Darknet kurulumu yapılarak hazır veri seti ve ağırlıklarla uygulama yapılmıştır. Uygulama esnasında karşılaşılan hataların giderilmesi için yeni eğitim ve test veri seti hazırlanmış ve çeşitli hiperparametrelerle YOLOv3 modeli eğitilmiştir. Son olarak modelin başarısı test edilmiş ve diğer modellerle kıyaslanmıştır.

3.2 Darknet ve Nesne Tanıma Ortamlarının Kurulumu

Darknet CUDA mimarisi ve C programlama dili ile geliştirilmiş açık kaynak kodlu yapay sinir ağı yazılım iskeleti (framework) olup standart fonksiyonların hazır olarak sunulduğu ancak bu fonksiyonların içerisinden istenen kısımlara ek kodlama yapılarak güncellenmesine ve geliştirilmesine imkan tanıyan bir sistemdir.

Darknet CPU veya GPU üzerinde kurulabilmektedir. GPU programlama görüntü işleme gibi farklı veriler üzerinde aynı işlemlerin yapıldığı uygulamalarda performans açısından daha iyi sonuçlar verdiğinden bu çalışmada da Darknet kurulumu GPU üzerinde yapılmıştır.

3.2.1 OpenCV

OpenCV gerçek zamanlı bilgisayar görüsü uygulamalarında görüntü işlemek için kullanılan C programlama dili ile geliştirilmiş açık kaynak kütüphanesidir. Yapılan çalışmada OpenCv v3.3.0 kullanılmıştır.

3.2.2 Visual studio (VS)

Visual Studio Windows işletim sistemi üzerinde kolay yazılım geliştirilmesini sağlayan ve geliştirme araçlarını içinde barındıran tümleşik geliştirme ortamıdır. Mevcut Windows işletim sisteminde çalışılması nedeniyle VS 2017 kurulmuştur.

3.2.3 CUDA

GPU için NVDIA tarafından geliştirilmiş C programlama dili üzerinde eklenti olarak sunulan mimaridir. C ile yazılmış algoritmaların GPU üzerinde çalışmasını sağlayan geliştirme araçları kümesidir. Yapılan çalışmada CUDA 10.0 kullanılmıştır.

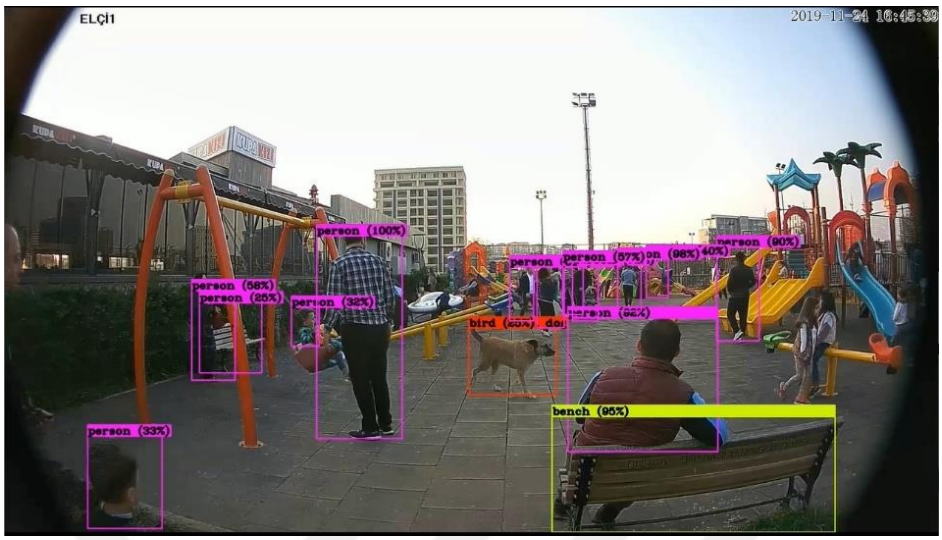
3.2.4 CuDNN

GPU üzerinde paralel işlem yapılmasını sağlayan derin sinir ağları için geliştirilmiş bir kütüphanedir. Bu çalışmada CuDNN 7.4.1 kullanılmıştır.

3.3 Hazır Veri Seti ve Ağırlıklarla Nesne Tanıma

Darknet ve nesne tanıma ortamlarının kurulumuyla standart olarak gelen YOLOv3 modeli kullanılarak çocuk oyun alanı ve hayvan barınağında nesne tanıma uygulaması yapılmış ve sonuçları incelenmiştir.

Köpeklerin kuş, koyun ve insan olarak hatalı şekilde tespit edildiği sahneler Şekil 2.2 ve Şekil 2.3'te gösterilmiştir.



Şekil 2.2: Hatalı nesne tanıma sahnesi-1



Şekil 2.3: Hatalı nesne tanıma sahnesi-2

COCO veri setinde 80 adet sınıfın bulunması ile veri seti içerisinde sağlıksız görünümlü, sakat ve farklı cins çiftleşmeleri nedeniyle kırma cinste sahipsiz köpek sayısının az olmasının bu hatalara neden olduğu değerlendirilmiştir.

3.4 YOLOv3 Modelinin Optimizasyonu

Sabit ve hareketli görüntüler üzerinden köpeklerin daha başarılı şekilde tespit edilebilmesi için yeni bir eğitim veri seti hazırlanmış ardından bu veri seti kullanılarak YOLOv3 ağının eğitimi yapılmıştır.

3.4.1 Yeni veri setinin hazırlanması

Hızlı yardım istasyonu Bursa Bademli Sevgi Hayvan Barınağı'na taşınarak burada istasyon üzerindeki kamera vasıtasıyla köpek görüntüleri 25 fps ve 1920x1080 çözünürlükte video şeklinde kaydedilmiştir. Sonrasında Python kod bloğu kullanılarak videolar framelerine ayrılmış ve resimlere dönüştürülmüştür. Bu şekilde farklı açılarda, ışık koşullarında ve arka plana sahip toplam 241 adet sahipsiz köpek görüntüsü elde edilmiştir. Çekilen örnek görüntüler Şekil 2.4'te gösterilmiştir.



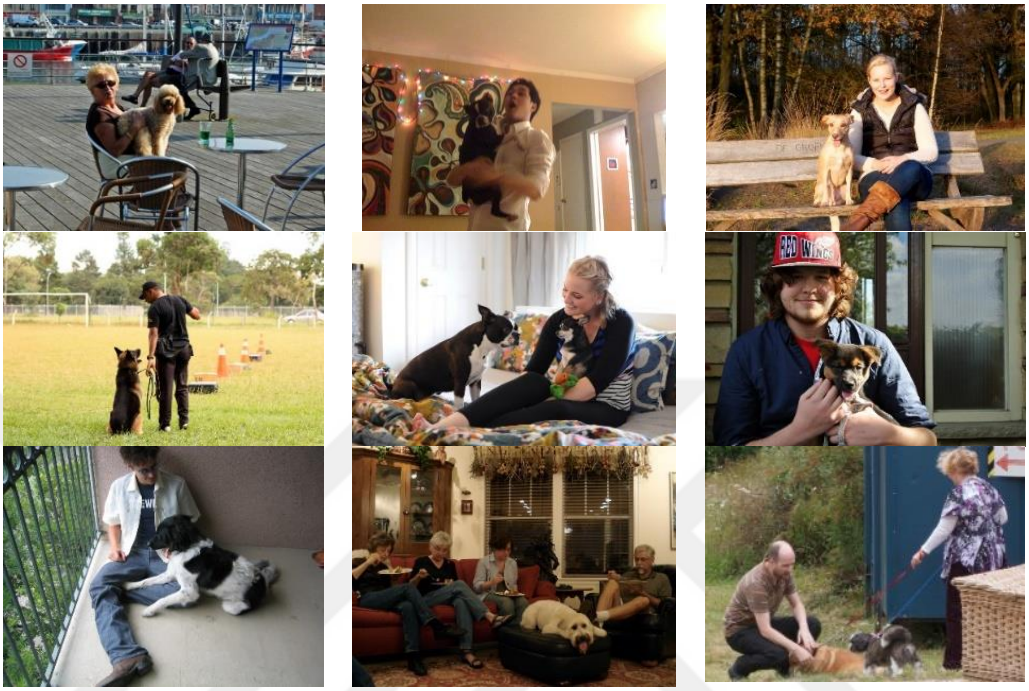
Şekil 2.4: Hızlı yardım istasyonu ile hayvan barınağından çekilen görüntüler

Çocuk oyun alanında hızlı yardım istasyonu ile farklı gün ve saatlerde çekilmiş kamera kayıtları resimlere dönüştürülerek içerisinden 236 adet çocuk görüntüsü seçilmiştir. Seçilen örnek görüntüler Şekil 2.5'te gösterilmiştir.



Şekil 2.5: Hızlı yardım istasyonu ile çocuk oyun alanında çekilen görüntüler

Open Images V6 [55] açık kaynak veri seti en geniş açık kaynak görüntü veri seti olup toplam 600 sınıf, 9.000.000 görüntü ve 16.000.000 nesne içermektedir. Bu veri setinden insan ve köpek bulunan 401 görüntü seçilmiştir. Seçilen örnek görüntüler Şekil 2.6'da gösterilmiştir.



Şekil 2.6: Open Images V6 veri setinden alınan örnek görüntüler **[55]**

Yukarıda belirtildiği şekilde elde edilen tüm görüntüler 878 görüntüden oluşan yeni bir veri setinde toplanmıştır.

3.4.2 YOLOv3 modelinin eğitilmesi

Yeni veri setinde bulunan her bir resimdeki köpek ve insanlar LabelImg açık kaynak etiketleme aracı kullanılarak Şekil 2.7'de gösterildiği gibi etiketlenmiştir.



Şekil 2.7: LabelImg etiketleme aracı kullanılarak resimlerin etiketlenmesi

Etiketleme sonucunda resim sayısıınca sınıf ve nesnelerin koordinat bilgilerini (nesne merkezi x, y koordinatı, nesne genişliği ve yüksekliği) içeren YOLOv3 formatında txt etiket dosyaları elde edilmiştir. Etiketleme dosya içeriği Çizelge 2.1'de gösterilmiştir.

Çizelge 2.1: YOLOv3 etiketleme dosyası içeriği

Labelled-Custom/image-1.txt				
1	0.874219	0.626389	0.126562	0.647222
0	0.719010	0.631019	0.076563	0.086111
0	0.772917	0.531944	0.044792	0.069444
0	0.603906	0.571759	0.056771	0.043519
0	0.422396	0.715741	0.092708	0.140741

YOLOv3 modelinin Darknet kullanılarak eğitimi için Çizelge 2.2'de yapısı gösterilen data uzantılı dosya ile Çizelge 2.3'te önemli parametreleri verilen cfg uzantılı eğitim ve test dosyalarının hazırlanması gerekmektedir.

Çizelge 2.2: YOLOv3 data dosya yapısı

Dosya grubu	Dosya adı	Dosya içeriği
Labelled_data.data		Classes=2
	Train.txt	Labelled-Custom/image-1.jpg Labelled-Custom/image-2.jpg Labelled-Custom/image-3.jpg …
	Test.txt	Labelled-Custom/image-101.jpg Labelled-Custom/image-102.jpg Labelled-Custom/image-103.jpg …
	Classes.names	Dog Person
		Backup = backup

Python kod bloğu kullanılarak etiketlenen resimlerin %85'i eğitim, %15'i test verisi olarak ayrılmış ve train.txt ile test.txt dosyaları elde edilmiştir.

Etiketleme sonucunda elde edilen ve sınıf isimlerini içeren txt dosyası da Python kod bloğu kullanılarak names uzantılı dosyaya dönüştürülmüştür.

Tanınacak nesne sayısı 2 olduğu için Classes değeri 2 olarak atanmış, eğitimde elde edilecek ağırlıkların saklanması için backup klasörü oluşturulmuştur. Data uzantılı dosya Python kod bloğu kullanılarak hazırlanmıştır.

Cfg uzantılı dosyalar ağın eğitimi ve testi için mimari yapısının, katmanların filtre sayısı, aktivasyon fonksiyonları, öğrenme oranı, giriş görüntüsü vb. özel parametreleri içeren dosyalardır. Çizelge 2.3'te YOLOv3 cfg uzantılı train ve test dosyalarında bu çalışma kapsamında değişiklik yapılan parametreleri gösterilmiştir.

Çizelge 2.3: YOLOv3 cfg dosyası önemli parametreleri

Parametre	Tanım	Train.cfg	Test.cfg
Batch	Bir yığında işlenen örneklem sayısı	64	1
Subdivisions	Bir yığın içerisindeki ağırlıkların GPU tarafından güncellendiği örneklem sıklığı	32	1
Width	Giriş resminin boyutlandırılacağı genişlik	320	320
Height	Giriş resminin boyutlandırılacağı yükseklik	320	320
Channels	Görüntü derinlik sayısı	3	3
Max_batches	Toplam iterasyon sayısı (Sınıf sayısı x 2000)	4000	4000
Steps	Öğrenme oranının güncellenme parametresi (İterasyon sayısının %80, %90'ı şeklinde seçilir)	3200/3600	3200/3600
Class	Her bir YOLO katmanı için sınıf sayısı	2	2

Daha önceden Imagenet'te eğitimi yapılmış olan Darknet-53 konvolüsyon ağırlıkları (darknet53conv.74) YOLOv3 ağının eğitiminde başlangıç ağırlıkları olarak verilmiştir.

3.4.3 Eğitim sonuçlarının değerlendirilmesi

İki veya daha fazla sınıfın bulunduğu sınıflandırma problemlerinde eğitilen modelin sonuçlarının değerlendirilmesi maksadıyla hata matrisleri kullanılır.

Tahmin edilen ve gerçek değerlerle 4 farklı kombinasyona (TP, FP, FN, TN) sahip Çizelge 2.4'teki hata matrisi elde edilir.

Çizelge 2.4: Hata Matrisi

Hata Matrisi		Gerçek Değerler	
		1	0
Tahmin Edilen Değerler	1	True Positive (TP) Doğru	False Positive (FP) Tip-1 hata
	0	False Negative (FN) Tip-2 hata	True Negative (TN) Doğru

IoU değeri kesin referans sınırlayıcı kutusu ile tahmin edilen sınırlayıcı kutu kesişimlerinin birleşimlerine oranı şeklinde tanımlanmıştır. IoU değerinin standart olarak 0,5 seçilmesi durumunda bu değerin üzerinde yapılan sınırlayıcı kutu tahminleri TP doğru tahminlere, altındaki tahminler ise FP Tip-1 hataya karşılık gelecektir.

Resimde kesin referans kutusu olmasına rağmen modelin nesne için sınırlayıcı kutu tahmini yapamaması FN Tip-2 hataya neden olmaktadır.

Resimde sınıflandırılmak istenilen nesneler dışında kalan diğer tüm arka plan TN'yi oluşturur ve performans hesaplamalarında kullanılmaz.

Hata matrisi kullanılarak mAP değeriyle birlikte 3 farklı başarım ölçütü elde edilir.

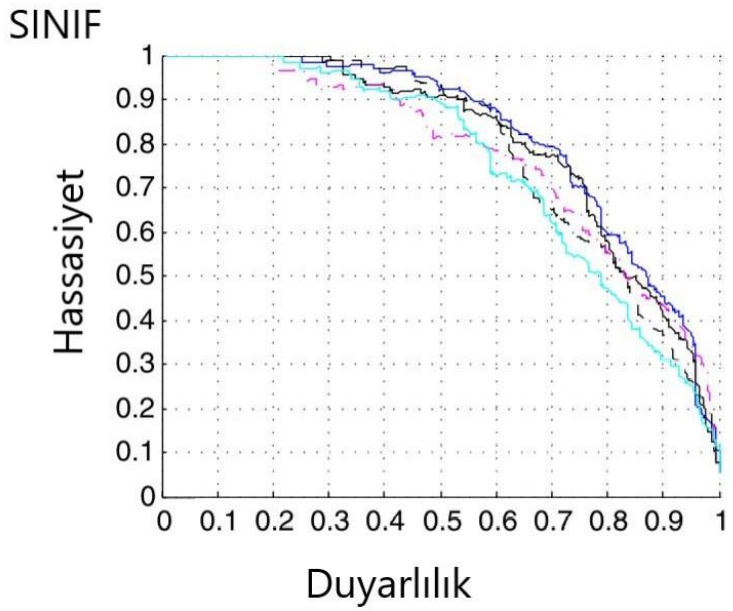
Hassasiyet yapılan doğru tahminlerin tüm tahminlere oranını verirken, duyarlılık nesne bulunması durumunda hangi doğruluk yüzdesiyle tahmin edildiğini gösterir. F1 skoru hassasiyet ve duyarlılık değerlerinin harmonik ortalaması olarak tek bir denklemde ifadesidir.

$$Hassasiyet\ (Precision) = \frac{TP}{TP + FP} \tag{2.1}$$

$$Duyarlılık\ (Recall) = \frac{TP}{TP + FN} \tag{2.2}$$

$$F1\ skoru = 2\text{x}\frac{HassasiyetxDuyarlılık}{Hassasiyet + Duyarlılık} = 2x\frac{TP}{2xTP + FN + FP} \tag{2.3}$$

Hassasiyet ve duyarlılık değerleri ile Şekil 2.8'deki hassasiyet-duyarlılık eğrisi elde edilir.



Şekil 2.8: Hassasiyet-duyarlılık eğrisi

Ortalama hassasiyet (AP) hassasiyet – duyarlılık eğrisi altında kalan alan olarak tanımlanır. Eğrinin sağ üst köşeye ve altında kalan alanın 1'e yakınlaşması sınıfa ait tahminin başarım oranının arttığını gösterir. AP değeri 2.4'teki eşitlikle hesaplanır.

$$AP = \int_0^1 p(r)dr \tag{2.4}$$

($p(r)$: hassasiyet-duyarlılık fonksiyonu)

Her bir sınıfa ait AP'lerin toplanması ve sınıf sayısına bölünmesiyle mAP değeri 2.5'teki eşitlikle elde edilir.

$$mAP = \frac{Tüm\ sınıfların\ ortalama\ kesinliklerinin\ toplamı}{Sınıf\ sayısı} \tag{2.5}$$

Ağın öğrenmesi aşamasında belirli bir iterasyon sayısının üzerine çıkıldığında sistem ezberlemeye ve kötü tahminler yapmaya başlar. Buna bağlı olarak mAP değeri de küçülür.

Literatürde yapılan çalışmalara bakıldığında optimum iterasyon sayısının belirlenmesinde sınıf sayısının en az 2000 katı şeklinde seçildiği görülmektedir. Bu çalışmada da 2 sınıf bulunduğundan güvenli bölgede kalmak için iterasyon sayısı 6000 seçilmiştir.

3. DENEYSEL SONUÇLAR

Bu çalışmada YOLOv3 ağının eğitimleri Intel Core i7 işlemcili, 16 GB RAM ve NVIDIA Geforce GTX 1050Ti GPU'ya sahip bir bilgisayar üzerinde yapılmıştır.

Ağın en doğru şekilde nesne tespitini yapabilmesi amacıyla hiperparametreler (giriş resim boyutu (widthxheight), batch ve subdivisions) değiştirilerek mevcut donanımın elverdiği şekilde Çizelge 3.1'deki 6 farklı kombinasyonda eğitimler gerçekleştirilmiştir. Bu hiperparametrelerin daha yüksek giriş resim boyutu (608x608) ve batch-subdivisions kombinasyonları (64-16, 32-8) denenmiş olup mevcut donanımda CUDA bellek hatası alındığından hiperparametrelerin güncellenmesi gerekmiştir.

Eğitimlerin sonucunda elde edilen en iyi mAP değerine sahip ağırlıklar için nesne tespit başarımının hesaplanmasında kullanılan TP, FP ve FN değerleri Çizelge 3.1'de gösterilmiştir.

Çizelge 3.1: Farklı hiperparametrelerle yapılan eğitimlerin başarım değerleri.

	Resim Boyutu	Batch	Subdivisions	TP	FP	FN
Eğitim-1	320x320	64	64	247	54	194
Eğitim-2	320x320	64	32	291	65	150
Eğitim-3	320x320	32	32	220	47	221
Eğitim-4	320x320	32	16	294	73	147
Eğitim-5	416x416	64	64	263	78	178
Eğitim-6	416x416	32	32	247	79	194

Eğitimdeki kesin referans sayısı (TP + FN) 441 olmuştur. Eğitimlerin tahmin ortalaması ise 326 olarak gerçekleşmiştir.

En düşük Tip-1 hata Eğitim-3, en yüksek Tip-1 hata ise Eğitim-6 sonucunda gerçekleşmiştir.

En düşük Tip-2 hata Eğitim-4, en yüksek Tip-2 hata ise Eğitim-3 sonucunda gerçekleşmiştir.

TP, FP ve FN değerleri kullanılarak elde edilen hassasiyet, duyarlılık ve F1 skoru başarım değerleri ile eğitimlerin ortalama IoU değerleri Çizelge 3.2'de verilmiştir.

Çizelge 3.2: Eğitimlerin hassasiyet, duyarlılık, F1-skoru ve IoU değerleri.

	Hassasiyet	Duyarlılık	F1-Skoru	IoU (%)
Eğitim-1	0,82	0,56	0,67	61,42
Eğitim-2	0,82	0,66	0,73	62,07
Eğitim-3	0,82	0,50	0,62	60,60
Eğitim-4	0,80	0,67	0,73	60,89
Eğitim-5	0,77	0,60	0,67	56,89
Eğitim-6	0,76	0,56	0,64	54,58

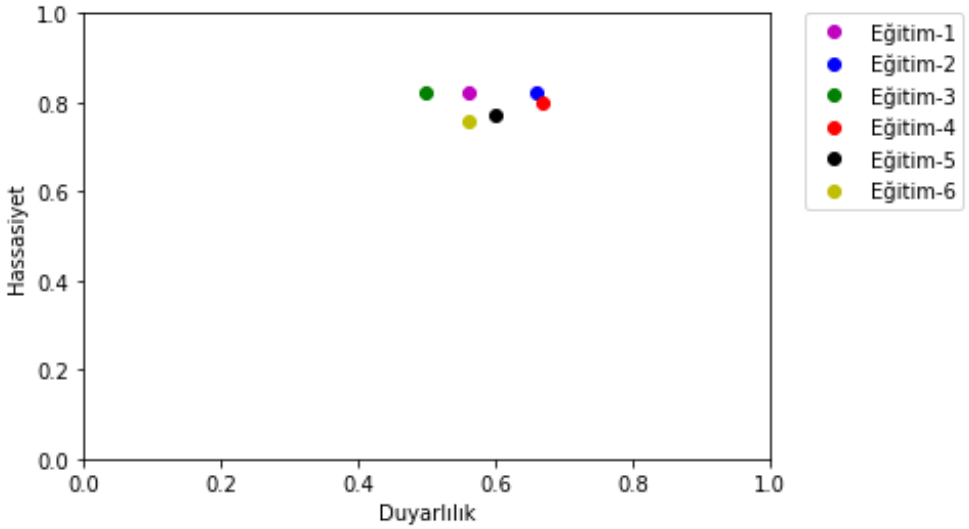
Hassasiyet ve duyarlılık arasında bir ödünleşim bulunmaktadır. Yüksek duyarlılığa ve düşük hassasiyete sahip ağlar çok sayıda tahmin üretse de bunlardan büyük bir kısmı hatalı olabilmektedir. Yüksek hassasiyet ve düşük duyarlılığa sahip ağlar ise az sayıda ancak nispeten daha doğru tahminler üretmektedir.

Çizelge 3.1'e göre en az sayıda tahmin (267 adet) üreten Eğitim-3'ün en yüksek hassasiyet değerine (0,82) sahip olduğu Çizelge 3.2'den de görülebilmektedir. Ancak, az sayıda tahmin üretimi beraberinde Tip-2 hatayı getirmiş ve böylece duyarlılık değerinin (0,50) düşük çıkmasına neden olmuştur. Sonuç olarak Eğitim-3, F1-skoru (0,62) bakımından sonuncu sırada yer almıştır.

En fazla sayıda tahmin üreten (367 adet) Eğitim-4'e ait hassasiyet değerinin (0,80) diğer eğitimlere göre nispeten düşük olmasına rağmen en yüksek duyarlılığa (0,67) sahip olması bu eğitimin en yüksek F1-skoruna sahip (0,73) iki eğitimden birisi olmasını sağlamıştır.

Ortalamaya yakın sayıda tahmin üreten (356 adet) Eğitim-2, en yüksek hassasiyet değerine (0,82) ve 2. en yüksek duyarlılık değerine (0,66) sahiptir. Bu dengeli durum Eğitim-2'nin de en yüksek F1-skoruna (0,73) ulaşmasını sağlamıştır.

Eğitimlerin en iyi hassasiyet-duyarlılık noktaları Şekil 3.1'deki gibi farklı renklerle gösterilmiştir.



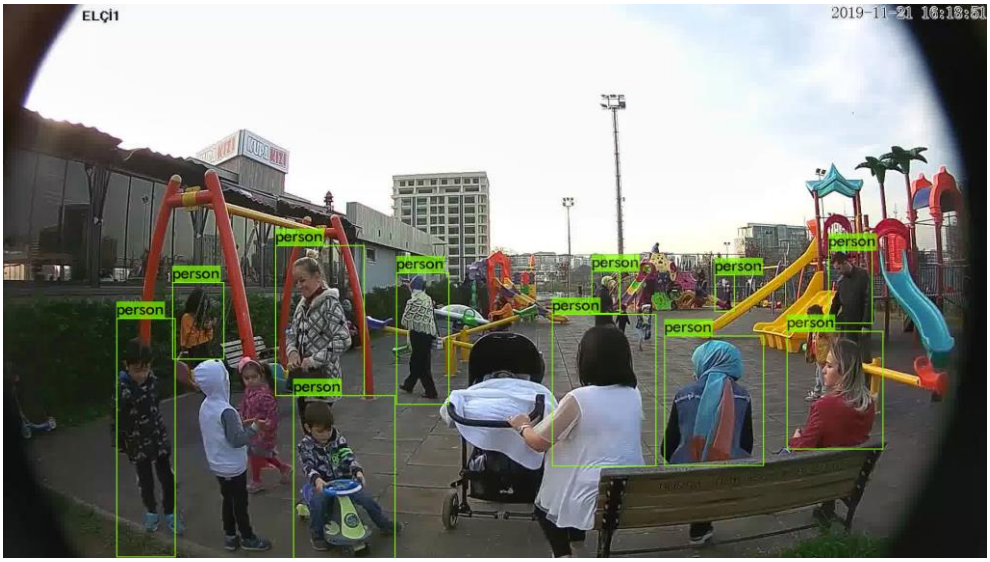
Şekil 3.1: Eğitimlerin en iyi hassasiyet-duyarlılık noktaları

Eğitim-3'e ait hassasiyet-duyarlılık noktasının diğer noktalara kıyasla sağ üst köşeye en uzakta olduğu Şekil 3.1'de görülmektedir. Bunun nedeni en düşük F1-skorunun bu eğitim sonucunda elde edilmiş olmasıdır.

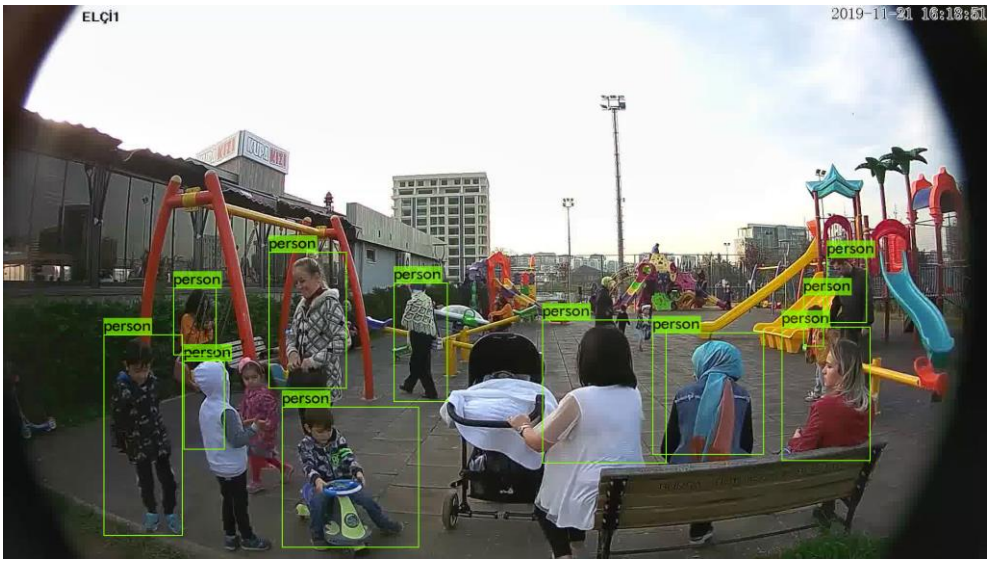
Eğitim-2 ve Eğitim-4'e ait hassasiyet-duyarlılık noktalarının diğer noktalara kıyasla sağ üst köşeye daha yakın olduğu Şekil 3.1'de görülmektedir. Bunun nedeni en yüksek F1-skorunun bu iki eğitimde elde edilmiş olmasıdır.

Eğitimlerin ortalama IoU değerleri Çizelge 3.2'de gösterilmiştir. En yüksek IoU değeri Eğitim-2, en düşük IoU değeri ise Eğitim-6 sonucunda elde edilmiştir.

En yüksek ve en düşük IoU değerlerine sahip Eğitim-2 ve Eğitim-6 ağırlıklarıyla yapılan validasyon testleri sırasıyla Şekil 3.2 ve Şekil 3.3'te gösterilmiştir.



Şekil 3.2: Eğitim-2 ağırlıkları ile validasyon testi



Şekil 3.3: Eğitim-6 ağırlıkları ile validasyon testi

Çocuk oyun alanına ait örnek görüntü üzerinde Eğitim-2 ve Eğitim-6 ağırlıklarıyla yapılan test sonuçları Çizelge 3.3'te verilmiştir.

Çizelge 3.3: Eğitim-2 ve Eğitim-6 güven skoru karşılaştırması.

Eğitim-2	Eğitim-6
insan: %82	insan: %91
insan: %81	insan: %80
insan: %97	insan: %28
insan: %94	insan: %83
insan: %98	insan: %63
insan: %92	insan: %69
insan: %40	insan: %69
insan: %99	insan: %98
insan: %71	insan: %77
insan: %97	insan: %26
insan: %91	insan: %65

Şekil 3.2 ve Şekil 3.3'teki örnek görüntü için Eğitim-2 ağırlığı %82,64, Eğitim-6 ağırlığı ise %68,09 ortalama güven skoru üretmiştir.

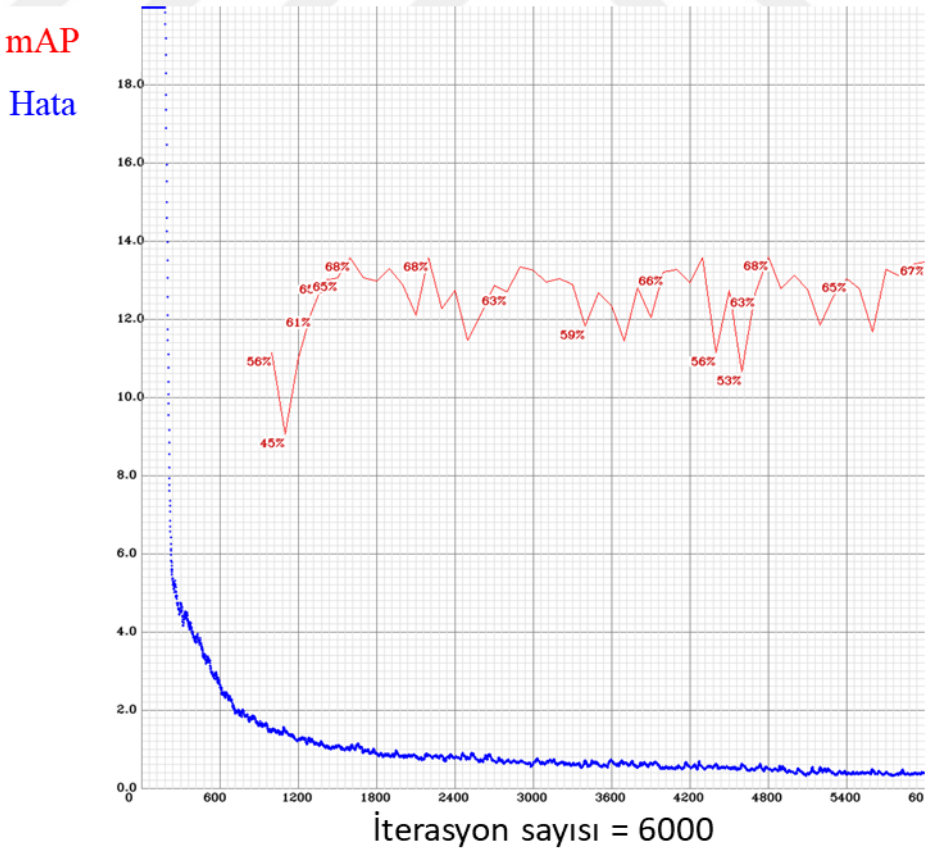
Eğitimlerde elde edilen AP, mAP ve ortalama hata değerleri Çizelge 3.4'te gösterilmiştir.

Çizelge 3.4: Eğitimlerin AP, mAP ve ortalama hata değerleri.

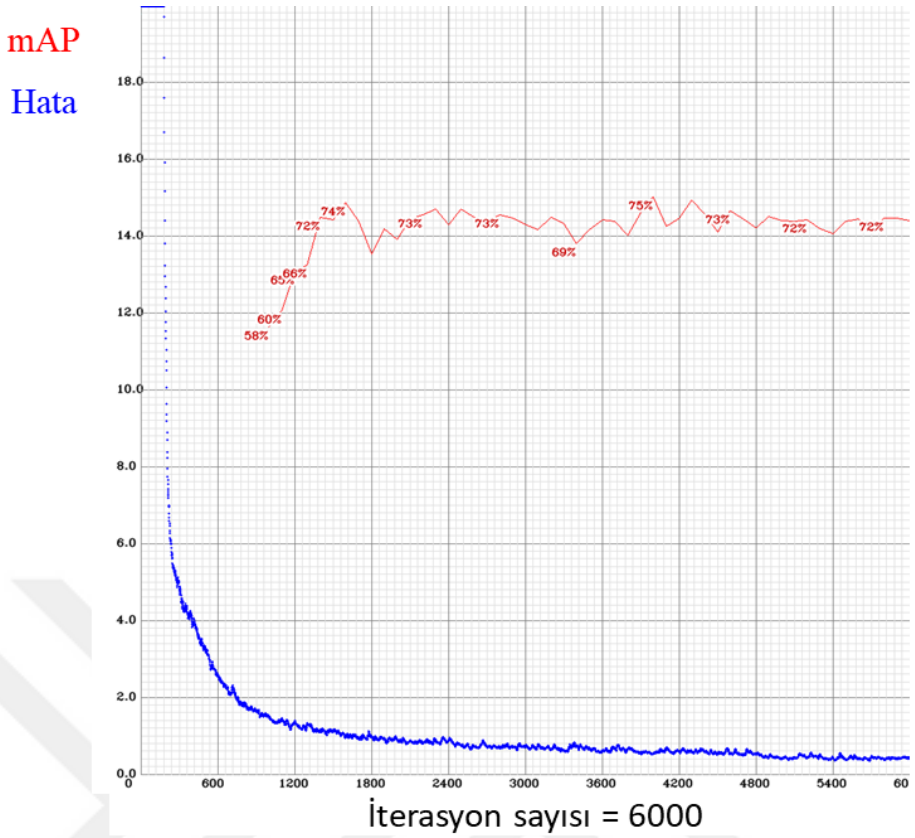
	AP_Köpek	AP_İnsan	mAP (%)	Hata
Eğitim-1	75,45	60,38	67,91	0,4034
Eğitim-2	82,59	67,52	75,05	0,4330
Eğitim-3	78,20	60,17	69,19	0,4979
Eğitim-4	83,06	65,51	74,28	0,5703
Eğitim-5	81,88	61,66	71,77	0,3391
Eğitim-6	81,03	60,83	70,93	0,4943

Batch ve subdivisions parametrelerinin sabit olması koşuluyla giriş resim boyutunun büyümesi ile ortalama hata değeri azalmaktadır. Resim boyutu sabit olması koşuluyla batch ve subdivisions değerlerinin artmasıyla ortalama hata değeri azalmaktadır.

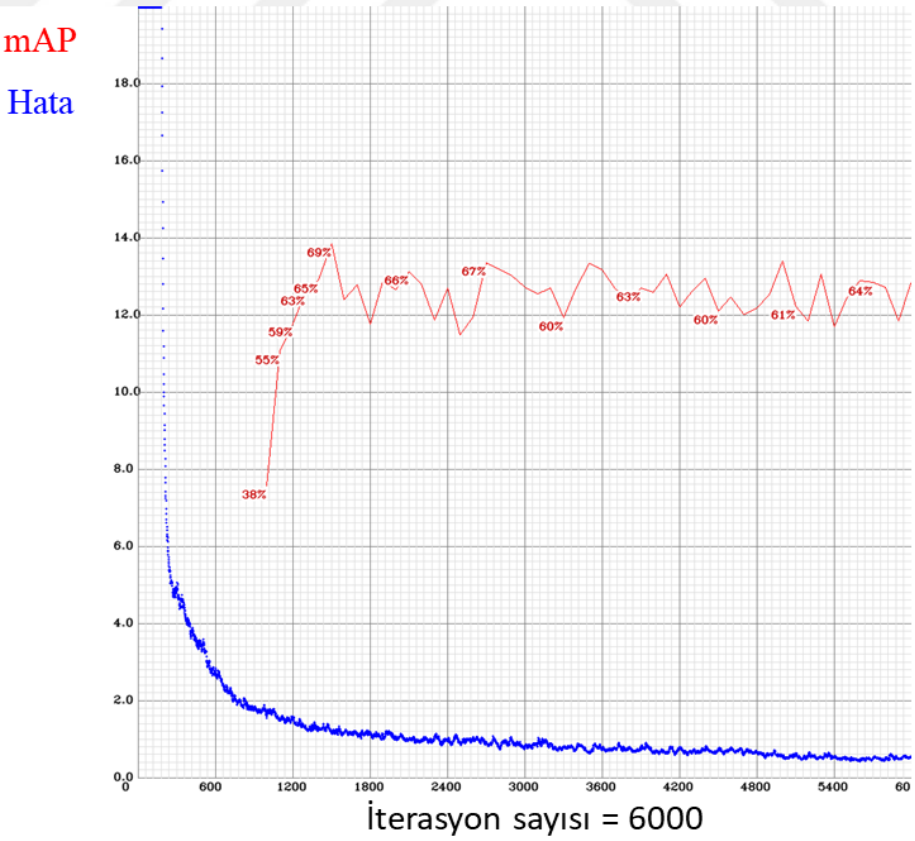
Tamamlanan tüm eğitimlerdeki iterasyon sayısına bağlı olarak mAP ve hata değerlerindeki değişimler Şekil 3.4, Şekil 3.5, Şekil 3.6, Şekil 3.7, Şekil 3.8 ve Şekil 3.9'da gösterilmiştir.



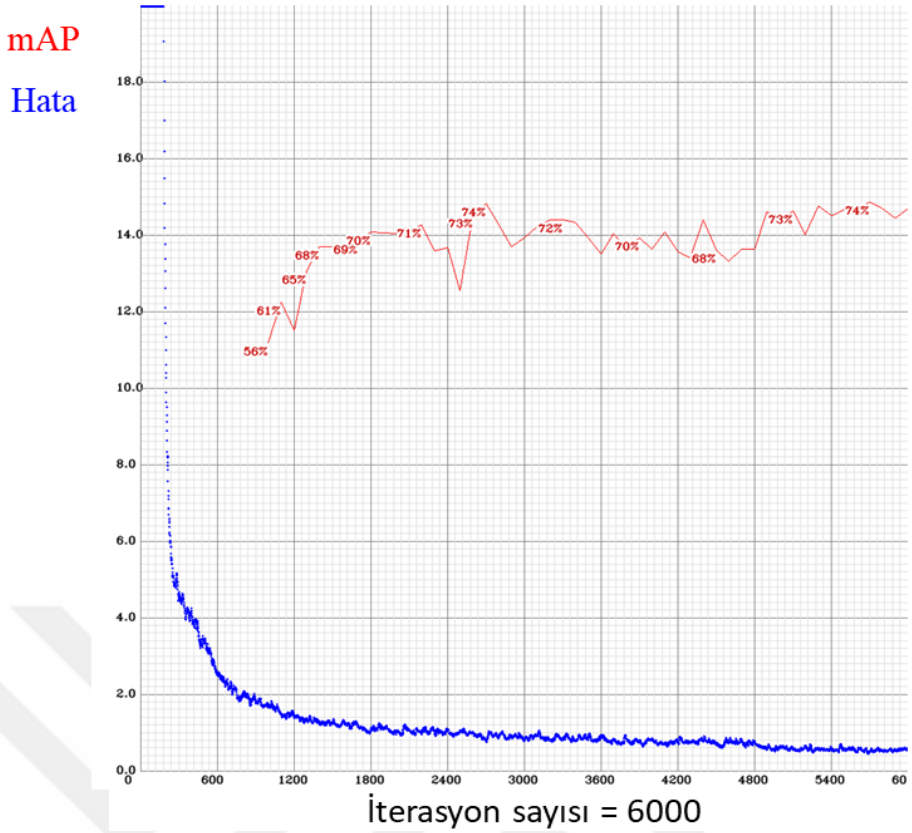
Şekil 3.4: Eğitim-1 ortalama hata ve mAP eğrisi



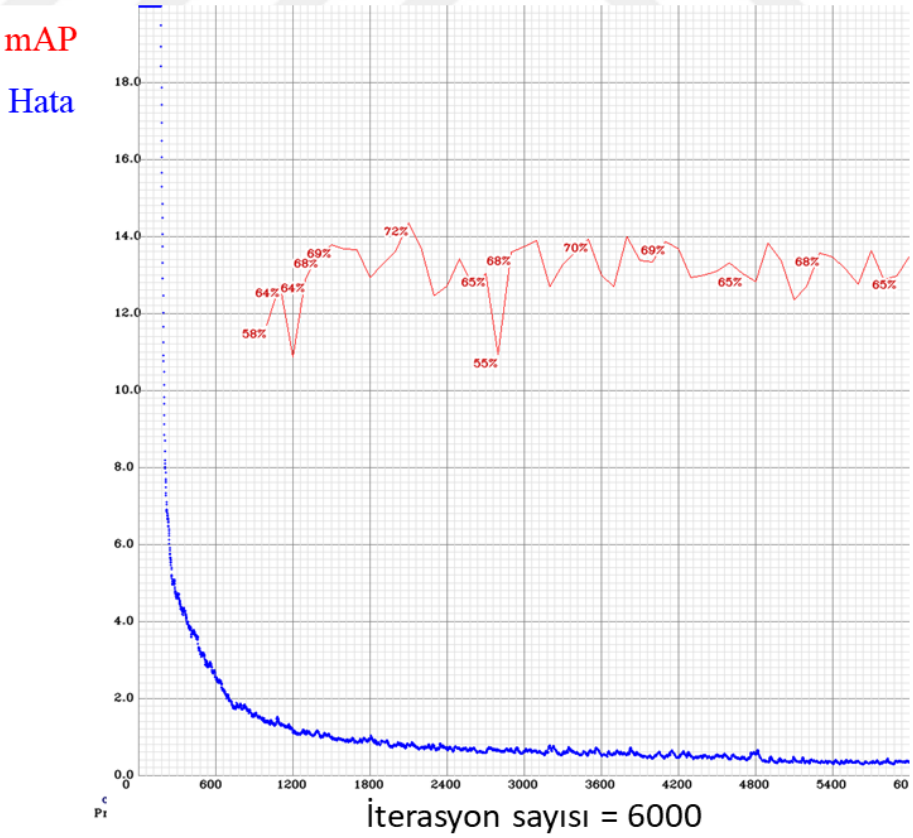
Şekil 3.5: Eğitim-2 ortalama hata ve mAP eğrisi



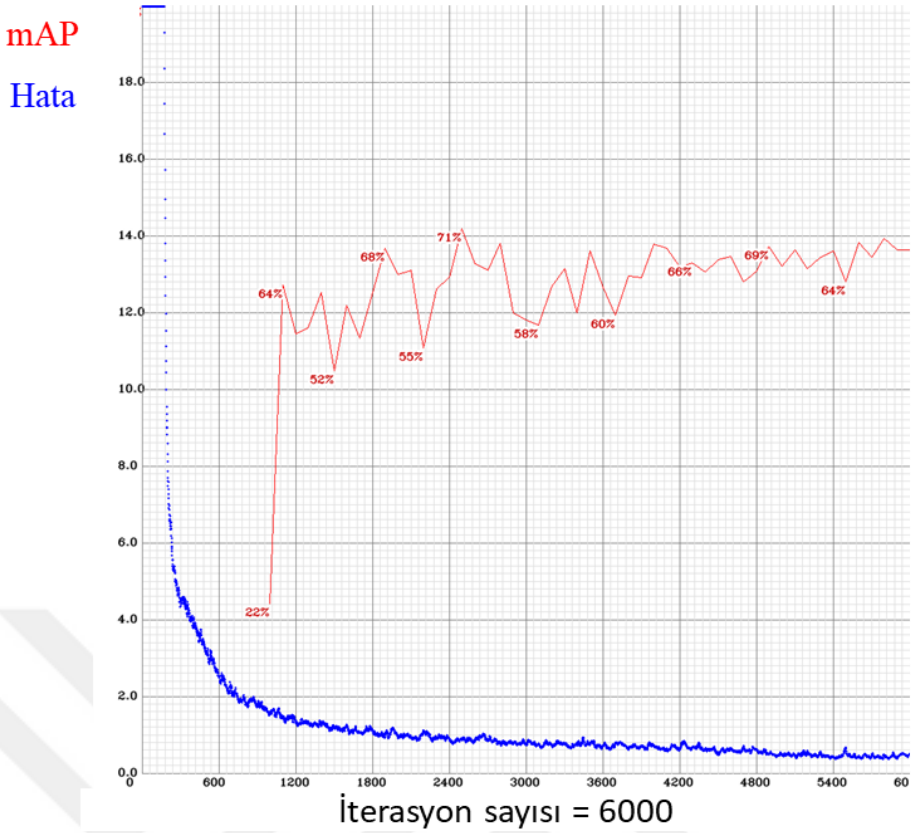
Şekil 3.6: Eğitim-3 ortalama hata ve mAP eğrisi



Şekil 3.7: Eğitim-4 ortalama hata ve mAP eğrisi

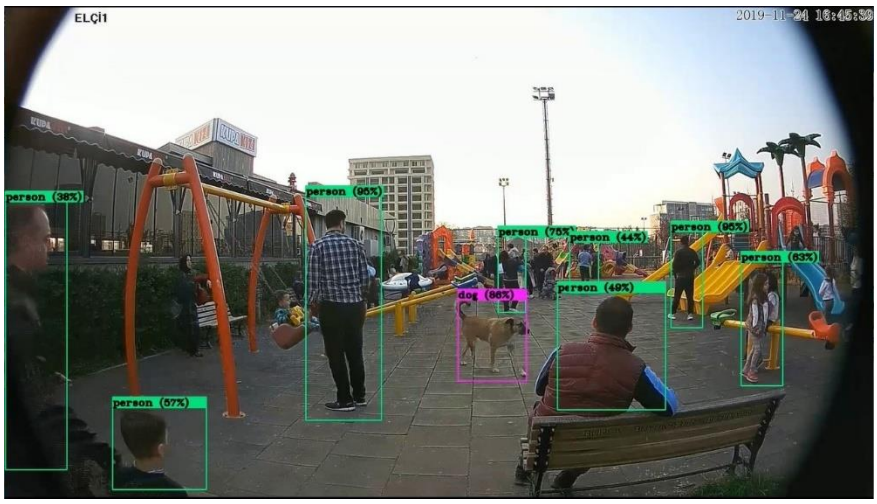


Şekil 3.8: Eğitim-5 ortalama hata ve mAP eğrisi

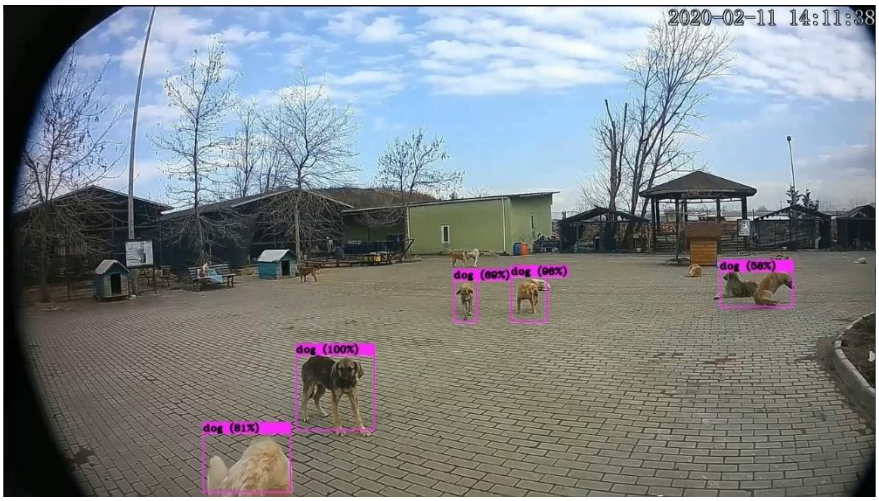


Şekil 3.9: Eğitim-6 ortalama hata ve mAP eğrisi

Yukarıdaki çizelge ve şekiller karşılaştırılarak en yüksek F1-skoru, IoU ve mAP değeri göz önüne alınarak ağın en başarılı şekilde Eğitim-2 sonucunda eğitildiği tespit edilmiştir. Eğitim-2 sonucu elde edilen ağırlıklar çocuk oyun alanlarında köpek tespiti için YOLOv3 ağının en iyi ağırlıkları olarak seçilmiştir. Seçilen ağırlıklarla çocuk oyun alanında ve hayvan barınağında yapılan validasyon testleriyle Şekil 2.2 ve Şekil 2.3'teki hataların düzeltildiği, Şekil 3.10 ve Şekil 3.11'de gösterildiği şekilde daha doğru tespitlerin yapıldığı gözlenmiştir.



Şekil 3.10: Önerilen modelle doğru köpek tespiti sahnesi-1



Şekil 3.11: Önerilen modelle doğru köpek tespiti sahnesi-2

Ayrıca, örnek bir videonun farklı çözünürlüklerinde (320x320 ve 608x608) seçilen ağırlıklar kullanılarak validasyon testleri gerçekleştirilmiştir. Bu videonun toplam 65 köpek içeren 30 sahnesi seçilmiştir. 320x320 çözünürlükteki videodan 52 adet köpek 25 fps ile doğru olarak tespit edilirken, 608x608 çözünürlükteki videodan 60 adet köpek 10,2 fps ile doğru tespit edilmiştir. Bu şekilde hızlı yardım istasyonunun kamera çözünürlüğü değiştirilerek doğruluk ile hız arasında takas yapılabildiği görülmüştür.

Önerilen modelin benzer çalışmalarla karşılaştırması Çizelge 3.5'te gösterilmiştir. Bu çalışmalarda kullanılan veri seti, çözünürlük ve tespit edilen hedef nesnenin birbirinden tamamen farklı olduğu belirtilmelidir. Ancak, çizelgedeki sonuçlar bu modeller arasında karşılaştırma imkanı sağlamaktadır. Çizelge 3.5'teki sonuçlar önerilen modelin özellikle gerçek zamanlı bir uygulamada (25 fps) diğer modellerden daha başarılı nesne tespiti yapabildiğini göstermektedir. Bunun yanı sıra önerilen model çocuk oyun alanındaki uygulamanın ihtiyaç duyduğu hız ve doğruluğu da karşılamaktadır.

Çizelge 3.5: Önerilen modelin benzer çalışmalarla karşılaştırılması.

Çalışmanın Adı	Nesne	Algoritma	Veri Seti	AP	Fps
Developing a Real-Time Gun Detection Classifier [7]	Silah	Overfeat	Imagenet	93%	0.77
Automatic Handgun Detection Alarm in Videos Using Deep Learning [8]	Tabanca	Faster R-CNN ve VGG-16 Temelli Sınıflandırıcı	Imagenet	84.21%	5

Çizelge 3.6 (devam) : Önerilen modelin benzer çalışmalarla karşılaştırılması.

Çalışmanın Adı	Nesne	Algoritma	Veri Seti	AP	Fps
Tinier-YOLO: A Real-Time Object Detection Method for Constrained Environments [9]	Köpek	Tinier YOLO	PASCAL VOC	67.9%	25.1
Önerilen Model	**Köpek**	**YOLOv3**	**Özel**	**82.59%**	**25**

4. SONUÇLAR VE ÖNERİLER

Bu tez çalışmasında çocuk oyun alanları için yeni bir akıllı şehir uygulaması olarak YOLOv3 temelli gerçek zamanlı bir köpek tespit modeli önerilmiştir. Öncelikle, veri setlerindeki sahipsiz görüntü verisi eksikliğinin giderilmesi için hayvan barınağından yeni görüntüler elde edilmiştir. Bu görüntüler eğitim verisi çeşitliliğini etkili bir şekilde arttırmış ve önerilen modelin performansını arttırmıştır. En iyi değerlendirme sonuçlarına sahip model önerilmiş ve seçilen bir çocuk oyun alanında hızlı yardım istasyonuna entegre edilerek uygulanmıştır. Gerçek zamanlı uygulama optimize edilmiş hiperparametrelerle %82,59 AP değerine ulaşmıştır. Ayrıca, validasyon testleri önerilen modelin Darkent kurulumuyla otomatik olarak gelen önceden eğitilmiş YOLOv3 modeline kıyasla daha başarılı köpek tespitleri yaptığını ortaya koymuştur.

Maalesef Covid-19 pandemisi nedeniyle çocuk oyun alanları uzun bir süre kapalı kalmış ve çocuk oyun alanının açıldığı zamanlarda da çok az sayıda çocuk bu çocuk oyun alanlarını kullanmıştır.

Gelecek çalışmalarda veri setine çocuk oyun alanlarında tehlike oluşturabilecek diğer nesnelerden ateşli silahlar ve bıçakların eklenmesi, insan hareketlerini anlamlandıracak algoritmalarla çocuğa şiddet ve çocuk kaçırılma gibi vakaların önüne geçilmesi hedeflenmektedir.

KAYNAKLAR

[1] **Maslow, A.H.** (1943). A theory of human motivation. *Psychologiical Review, 50*, 370-396.

[2] **Kaya, U., Yılmaz, A., & Dikmen, Y.** (2019). Sağlık Alanında Kullanılan Derin Öğrenme Yöntemleri. *European Journal of Science and Technology, 16,* 792-808.

[3] **Luna, E., San Miguel, J., Ortego, D., & Martínez, J.** (2018). Abandoned Object Detection in Video-Surveillance: Survey and Comparison, *Sensors, 18*(12), 4290.

[4] **Namozov, A., & Cho, Y. I.** (2018). An Efficient Deep Learning Algorithm for Fire and Smoke Detection with Limited Data, *AECE, 18*(4), 121-128.

[**5]** **Sun, Y., Liang, D., Wang, X., & Tang, X.** (2015). *DeepID3: Face Recognition with Very Deep Neural Networks*, Retrieved November 12, 2020 from https://arxiv.org/abs/1502.00873.

[6] **Wang, T., Miao, Z., Chen, Y., Zhou, Y., Shan, G., & Snoussi, H.** (2019). AED-Net: An Abnormal Event Detection Network, *Engineering, 5*(5), 930–939.

[7] **Lai, J., & Maples, S.** (2017). Developing a Real-Time Gun Detection Classifier, *World Academy of Science*, Stanford University.

[8] **Olmos, R., Tabik, S. & Herrera, F.** (2017). Automatic Handgun Detection Alarm in Videos Using Deep Learning, *Neurocomputing, 275,* 66-72.

[9] **Fang, W., Wang, L. & Ren, P.** (2020). Tinier-YOLO: A Real-Time Object Detection Method for Constrained Environments, *IEEE Access, 8,* 1935-1944,

[10] **Çocuklara Büyükşehir koruması | Haberler**. Erişim tarihi: 10 Ağustos 2020, https://www.bursa.bel.tr/cocuklara-buyuksehir-korumasi/haber/27347.

[11] **14 Year U.S. Dog Bite Fatality Chart - 2005 to 2018**. Erişim tarihi: 10 Ekim 2020, https://www.dogsbite.org/reports/14-year-dog-bite-fatality-chart-trends-in-3-periods-2005-2018.pdf.

[12] **Aksoy, M., Demirbaş, B., Maden, F., Şimşek, Ç., Özlü, A., Kaya, M., Çulha, G., Topaç, O., Arı, E., Yılmaz, H., & Gürbüz, F.** (2010). Ankara İlinde 2005–2009 yılları arasında görülen şüpheli ısırıkların ve kuduz aşılamasının değerlendirilmesi, *Türkiye 3. EKMUD Kongresi Özet Kitabı,* 199.

[13] **Gunduz, T., Elcioglu, O. & Balci, Y.** (2011). An Evaluation of Dog And Cat Bites For a Five Year Period: A Sample Case From Eskisehir, *Turkish Journal of Trauma and Emergency Surgery, 17*(2), 133–140.

[14] **Karadağ, M., Çatak, B., Baştürk, S. & Elmas, Ş.** (2014). Assessment of Rabies-Risk Contact Notifications In Yıldırım, District of Bursa, *Turkish Journal of Family Practice, 18*(3), 117–121.

[15] Tekyol, D. (2019). Evaluation of rabies risky contact cases admitted to emergency department last year, *Haydarpasa Numune Medical Journal*, *61*(2), 0.

[16] Gompper, M. E. (2013). Free-Ranging Dogs and Wildlife Conservation, *Ed. Oxford University Press, Oxford, United Kingdom*.

[17] İnci, A., Doğanay, M., Özdarendeli, A., Düzlü, Ö. & Yıldırım, A. (2018). Overview of Zoonotic Diseases in Turkey: The One Health Concept and Future Threats, *Türkiye Parazitoloji Dergisi, 42*(1), 39–80.

[18] Halk Sağlığı ve Sokak Hayvanları, Erişim tarihi: 15 Ekim 2020, http://www.skb.gov.tr/wp-content/uploads/2018/05/Halk-Sagligi-ve-Sokak-Hayvanlari-Doc.-Dr.-Sibel-Cevizci-Oymak.pdf.

[19] Aylan, O., Baykam, N., Güner, R., Kara, A., Köksal, İ., Seçer, M., Tülek, N., Ünal, N. (2019). T.C. Sağlık Bakanlığı Kuduz Profilaksi Rehberi.

[20] Lowe, D. G. (2004). Distinctive Image Features from Scale-Invariant Keypoints, *International Journal of Computer Vision*, *60*(2), 91-110.

[21] Bay, H., Ess, A., Tuytelaars, T., & Van Gool, L. (2008). Speeded Up Robust Features (SURF), *Computer Vision and Image Understanding*, *110*(3), 346-359.

[**22] Viola, P. , & Jones, M.** (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.CVPR 2001*, Kauai, HI, USA: December 8-14.

[23] Dalal, N., & Triggs, B. (2005). Histograms of Oriented Gradients for Human Detection, *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, San Diego, CA, USA: June 20-25.

[24] Özcan, A. R. (2013). G*ömülü yaya belirleme sistemi tasarımı ve gerçeklemesi*. (Yüksek lisans tezi). Yıldız Teknik Üniversitesi, Fen Bilimleri Enstitüsü, İstanbul.

[25] Boiman, O., Shechtman, E., & Irani, M. (2008). In defense of Nearest-Neighbor based image classification, *2008 IEEE Conference on Computer Vision and Pattern Recognition*, Anchorage, AK, USA: June 23-28.

[26] Timofte, R., Tuytelaars, T., & Van Gool, L. (2012). Naive Bayes Image Classification: Beyond Nearest Neighbors, *ACCV 2012*, Daejeon, Korea: November 5-9.

[27] Girshick, R., Donahue, J. , Darrell, T., & Malik, J. (2014). Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation, *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, OH, USA: June 23-28.

[28] Tang, Y. (2013). *Deep Learning using Linear Support Vector Machines*, Retrieved November 15, 2020 from https://arxiv.org/abs/1306.0239.

[29] Wilking, D., & Röfer, T. (2004). Realtime Object Recognition Using Decision Tree Learning, *RoboCup 2004: Robot Soccer World Cup VIII*.

[30] Dalai R., & Senapati, K. K. (2018). An Innovative Tree Gradient Boosting Method Based Image Object detection from a Uniform Background Scene, *Pattern Recognition and Image Analysis*, *28*(2), 217-224.

[31] Gall, J., Razavi, N., & Van Gool, L. (2012). An Introduction to Random Forests for Multi-class Object Detection, *Outdoor and Large-Scale Real-World Scene Analysis, 7474*, 243-263.

[32] Shotton, J., *et al*. (2011). Real-time human pose recognition in parts from single depth images, *CVPR 2011*, Providence, RI, USA: June 20-25.

[33] Hou, J. , Gao, H. , Xia, Q., & Qi, N. (2016). Feature Combination and the kNN Framework in Object Classification, *IEEE Transactions on Neural Networks and Learning Systems*, *27*(6), 1368-1378

[34] McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity, *The bulletin of mathematical biophysics*, *5*(4), 115-133.

[35] Widrow, B., & Lehr, M. A. (1990). 30 years of adaptive neural networks: perceptron, Madaline, and backpropagation, *Proceedings of the IEEE*, *78*(9), 1415-1442.

[36] Rumelhart, D. E., & Mcclelland J. L. (1986). *Parallel distributed processing: explorations in the microstructure of cognition*, MIT Press, Cambridge, Massachusetts.

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

[38] Hinton, G. E. (2006). Reducing the Dimensionality of Data with Neural Networks, *Science*, *313*(5786), 504-507.

[39] Krizhevsky, A., Sutskever, I., & Hinton, G.E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, *60*, 84-90.

[40] Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2012). *Improving neural networks by preventing co-adaptation of feature detectors*, Retrieved November 15, 2020 from https://arxiv.org/abs/1207.0580.

[41] Liu, T., Fang, S., Zhao, Y., Wang, P., & Zhang, J. (2015). *Implementation of Training Convolutional Neural Networks*. Retrieved December 10, 2020 from https://arxiv.org/abs/1506.01195.

[42] Zeiler, M.D., & Fergus, R. (2014). Visualizing and Understanding Convolutional Networks. ECCV.

[43] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1-9.

[44] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770-778.

[45] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R.B., Guadarrama, S., & Darrell, T. (2014). Caffe: Convolutional Architecture for Fast Feature Embedding. *Proceedings of the 22nd ACM international conference on Multimedia*.

[46] Girshick, R.B. (2015). Fast R-CNN. *2015 IEEE International Conference on Computer Vision (ICCV)*, 1440-1448.

[47] Ren, S., He, K., Girshick, R.B., & 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.

[48] He, K., Gkioxari, G., Dollár, P., & Girshick, R.B. (2017). Mask R-CNN. *2017 IEEE International Conference on Computer Vision (ICCV)*, 2980-2988.

[49] Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., & LeCun, Y. (2014). OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. *CoRR*, abs/1312.6229.

[50] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C., & Berg, A. (2016). SSD: Single Shot MultiBox Detector. *ECCV*.

[51] Redmon, J., Divvala, S., Girshick, R.B., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 779-788.

[52] Redmon, J., & Farhadi, A. (2018). *YOLOv3: An Incremental Improvement.* Retrieved December 3, 2020 from https://arxiv.org/abs/1804.02767.

[53] Deploying a Scalable Object Detection Inference Pipeline. Erişim tarihi: 27 Eylül 2020, https://developer.nvidia.com/blog/deploying-a-scalable-object-detection-inference-pipeline.

[54] COCO - Common Objects in Context. erişim tarihi: 13.12.2020, https://cocodataset.org/#home

[55] Open Images V6. erişim tarihi: 30.11.2020, https://storage.googleapis.com/openimages/web/index.html