T.C.

TRAKYA ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

KONVOLÜSYONEL SİNİR AĞI KULLANILARAK TRAFİK İŞARETLERİNİ GERÇEK ZAMANLI BULMA VE TANIMA

SAVAŞ HASÇELİK

YÜKSEK LİSANS TEZİ

HESAPLAMALI BİLİMLER ANABİLİM DALI

TEZ DANIŞMANI : PROF. DR. ŞABAN AKTAŞ

EDİRNE-2021

SAVAŞ HASÇELİK'in hazırladığı "**KONVOLÜSYONEL SİNİR AĞI KULLANILARAK TRAFİK İŞARETLERİNİ GERÇEK ZAMANLI BULMA VE TANIMA"** başlıklı bu tez, tarafımızca okunmuş, kapsam ve niteliği açısından Hesaplamalı Bilimler. Anabilim Dalında bir **Yüksek lisans tezi** olarak kabul edilmiştir.

Jüri Üyeleri (Ünvan, Ad, Soyad): İmza

Prof. Dr. Şaban AKTAŞ ……

Doç. Dr. İlhan UMUT ……

Dr. Öğr. Üyesi Mustafa ULAŞ ……

Tez Savunma Tarihi: 30/07/2021

Bu tezin Yüksek Lisans tezi olarak gerekli şartları sağladığını onaylarım.

İmza

(Prof. Dr. Şaban AKTAŞ)

Tez Danışman ……

Trakya Üniversitesi Fen Bilimleri Enstitüsü onayı

Prof. Dr. Hüseyin Rıza Ferhat KARABULUT

Fen Bilimleri Enstitüsü Müdürü

T.Ü.FEN BİLİMLERİ ENSTİTÜSÜ
HESAPLAMALI BİLİMLER ANABİLİM DALI YÜKSEK LİSANS PROGRAMI
DOĞRULUK BEYANI

Trakya Üniversitesi Fen Bilimleri Enstitüsü, tez yazım kurallarına uygun olarak hazırladığım bu tez çalışmasında, tüm verilerin bilimsel ve akademik kurallar çerçevesinde elde edildiğini, kullanılan verilerde tahrifat yapılmadığını, tezin akademik ve etik kurallara uygun olarak yazıldığını, kullanılan tüm literatür bilgilerinin bilimsel normlara uygun bir şekilde kaynak gösterilerek ilgili tezde yer aldığını ve bu tezin tamamı ya da herhangi bir bölümünün daha önceden Trakya Üniversitesi ya da farklı bir üniversitede tez çalışması olarak sunulmadığını beyan ederim.

30 / 07 / 2021
Savaş HASÇELİK

Yüksek Lisans Tezi

Konvolüsyonel Sinir Ağı Kullanılarak Trafik İşaretlerini Gerçek Zamanlı Bulma ve Tanıma

T.Ü. Fen Bilimleri Enstitüsü

Hesaplamalı Bilimler Anabilim Dalı

ÖZET

Akıllı araç teknolojilerinin hızla gelişmekte olduğu dünyamızda makine öğrenim yöntemi ile mevcut araçlarda da kullanılmaya başlanan ve gün geçtikçe geliştirilen trafik işaretleri tanımlama problemine değinilmiştir.

Bu amaçla gerçek zamanlı olarak trafik işaretlerini tanımak için, katkı yapılan ve yararlanılan veri seti Türkiye Trafik Veri Seti (TTVS 2020) ile iki ayrı model eğitilecek olup bu modellerin farklı ortamdaki (bilgisayar-mobil-IoT geliştirme kartı) durumları değerlendirilecektir. Çalışmanın trafik işaretlerini tanıma ve ülkemizde bu probleme yönelik oluşturulan veri setine katkı sağlanacağı düşünülmektedir.

Yıl : 2021

Sayfa Sayısı : 59

Anahtar Kelimler : Yapay Sinir Ağları, Trafik İşaretleri, Nesne Tanıma

Master's Thesis
Real-Time Finding and Recognition of Traffic Signs Using Convolutional Neural Network
Trakya University Institute of Natural Sciences
Department of Computational Sciences

ABSTRACT

In our world where smart vehicle technologies are developing rapidly, the problem of identifying traffic signs, which is being used in existing vehicles with the machine learning method, and which is being developed day by day, is mentioned.

For this purpose, the knowledge of real-time traffic signs, made contributions and used data set Turkey Traffic Data Set (TTVS Github 2020), two models with these models in different media (PC-mobile-IoT development board) will be trained. the situation will be evaluated. It is thought that the study will contribute to the recognition of traffic signs in our country and the data set created for this problem.

Year : 2021

Number of Pages : 59

Keywords : Artificial Neural Networks, Traffic Signs, Object Recognition

TEŞEKKÜR

Bu tez, Trakya Üniversitesi Fen Bilimleri Enstitüsü Hesaplamalı Bilimler Anabilim dalı, Hesaplamalı Bilimler Yüksek Lisans Programı'nda hazırlanmıştır. Çalışmam da Konvolüsyonel (Evrişimli) Sinir Ağı Kullanılarak Trafik İşaretlerini Gerçek Zamanlı Bulma Ve Tanıma incelenmiştir. Tez çalışmam süresince bana her aşamada yardımcı olan danışmanım Sayın Prof. Dr. Şaban AKTAŞ'a teşekkür ederim.

Tüm eğitim-öğretim hayatım boyunca maddi ve manevi desteklerini esirgemeyen aileme, sabırları için sevgili eşim Melek HASÇELİK'e, kızım Berfin İkra HASÇELİK'e ve özellikle yaptığı fedakarlıklarla desteğini her daim hissettiğim sevgili abim Barış HASÇELİK'e teşekkür ederim. Bu tezin, bundan sonraki çalışmalara katkı sağlamasını temenni ederim.

İÇİNDEKİLER

ÖZET iv

ABSTRACT v

TEŞEKKÜR vi

İÇİNDEKİLER vii

KISALTMALAR DİZİNİ ix

ŞEKİLLER DİZİNİ x

ÇİZELGELER DİZİNİ xii

BÖLÜM 1 1

GİRİŞ 1

BÖLÜM 2 3

2. Literatür Araştırması 3

2.1. Yapay Sinir Ağları 3

2.2. Derin Öğrenme 5

2.2.1. Tarihçe 5

2.2.2. Nesne Algılama 6

2.2.3. Konvolüsyonel (Evrişimli) Sinir Ağı(CNN) 7

2.2.3.1. Konvolüsyonel (Evrişim) katmanı (CONV) 8

2.2.3.2. Havuzlama Katmanı(Pooling) 9

2.2.3.3. Tam Bağlantılı (Full Connected Layer) 10

2.2.3.4. Kesiştirilmiş Bölgeler(IoU) 11

2.2.3.5. Öneri Kutuları (Anchor Boxes) 11

2.2.3.6. Maksimum Olmayan Bastırma(NMS) 12

2.2.4. Öneri Kutuları Tabanlı Nesne Tanıma Algoritmaları 13

2.2.4.1. R-CNN, Fast R-CNN ve Faster R-CNN Algoritmaları 13

2.2.4.2. Tek Atış Çoklu Kutu Algılama (SSD) 16

2.2.4.3. Mask R-CNN Algoritması 16

2.2.4.4. YOLO(versiyon 5) Algoritması 17

2.2.5. Öneri Kutuları İçermeyen Nesne Tanıma Algoritmaları 18

2.2.5.1. YOLO(versiyon 1) Algoritması 19

2.2.5.2. FCOS Algoritması 19

2.2.6. Trafik İşaretleri 19

BÖLÜM 3 21

3. Materyal ve Yöntem 21

3.1. Veri Seti 21

3.2. Modellerin Eğitilmesi 23

3.2.1. YOLOv5 Modeli 24

3.2.2. Nanodet Modeli 27

3.3. Android Uygulaması 30

3.4. Jetson Nano Uygulaması 33

BÖLÜM 4 38

SONUÇLAR VE TARTIŞMA 38

KAYNAKLAR 44

ÖZGEÇMİŞ 47

KISALTMALAR DİZİNİ

ADAS : Advanced driver-assistance systems (Gelişmiş Sürücü Destek Sistemi).

CNN : Convolutional Neural Network (Konvolüsyonel-Evrişimli Sinir Ağı).

FCOS : Fully Convolutional One-Stage Object Detection.

FPS : Frame Per Second (Saniyedeki Kare Sayısı).

GTSRB : German Traffic Sign Recognition Benchmark (Alman Trafik İşaretleri Seti).

ISA : Intelligent Speed Assistance (Akıllı Hız Yardımı).

TSR : Traffic Sign Recognition.

TTVS : Türkiye Trafik İşaretleri Veri Seti.

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

YSA : Yapay Sinir Ağı.

ŞEKİLLER DİZİNİ

Şekil 2.1. Biyolojik sinir sistemi. 3
Şekil 2.2. Basit yapay sinir ağı yapısı (Prateek 2017) 4
Şekil 2.3. Bilinen ilk derin ağ. (Dettmers, 2015; Şeker vd. 2017) 5
Şekil 2.4. 3x vektör. 7
Şekil 2.5. Geleneksel CNN yapısı. (Amidi & Amidi, 2018) 8
Şekil 2.6. 7x7x3 özellik haritası 9
Şekil 2.7. Havuzlama katmanı 10
Şekil 2.8. Tam bağlantılı katman 10
Şekil 2.9. Kesiştirilmiş bölgeler(IoU) 11
Şekil 2.10. Maksimum olmayan bastırma (NMS) 12
Şekil 2.11. R-CNN mimarisi.s 13
Şekil 2.12. Fast-R-CNN mimarisi. 14
Şekil 2.13. Faster R-CNN mimarisi 15
Şekil 2.14. Algoritmaların nesne algılama hızları(saniye). 15
Şekil 2.15. SDD mimarisi. 16
Şekil 2.16. Mask RCNN sonucu 17
Şekil 2.17. YOLOv5 algoritmasının hızı. .(Jocher 2020) 18
Şekil 3.1. Veri setinden örnekler. 23
Şekil 3.2. Model eğitim diagramı 24
Şekil 3.3. YOLOv5 model eğitim süreci 24
Şekil 3.4. LabelImg görüntü etiketleme 25
Şekil 3.5. YOLOv5 model eğitimi kayıp oranı 26
Şekil 3.6. YOLOv5 mAP grafiği 26
Şekil 3.7. YOLOv5 eğitim sırasında nesne tanıma 27
Şekil 3.8. Nanodet model eğitimi kayıp oranı 28

Şekil 3.9. Nanodet model eğitimi 28
Şekil 3.10. Nanodet mAP Grafiği 29
Şekil 3.11. Nanodet örnek nesne tanıma 29
Şekil 3.12. Modellerin ncnn'e dönüştürme 30
Şekil 3.13. Android uygulamasının arayüzü 32
Şekil 3.14. Android uygulamasından örnekler 33
Şekil 3.15. Modelerin kendi ortamlarında Jetson Nano üzerinde çalıştırılması 34
Şekil 3.16. Jetson Nano genel görünümü 35
Şekil 3.17. Jetson Nano ekranı genel görünüm 35
Şekil 3.18. Detay ekranı 36
Şekil 3.19. Jetson Nano ile örnek tespit 37
Şekil 4.1. YOLOv5 örnek işaret tanıma 39
Şekil 4.2. Nanodet örnek işaret tanıma 39
Şekil 4.3. Android uygulamasında nanodet modeli 40
Şekil 4.4. Android uygulamasında YOLOv5 modeli 41
Şekil 4.5. Jetson Nano üzerinde YOLOv5 uygulaması 41
Şekil 4.6. Jetson Nano üzerinde nanodet uygulaması 42

ÇİZELGELER DİZİNİ

Çizelge 2.1. Biyolojik ve yapay sinir sistemi bileşenleri 3
Çizelge 3.1. Veri setini oluşturan sınıflar 22
Çizelge 3.2. Android ortamını destekleyen kütüphaneler 31

BÖLÜM 1

GİRİŞ

İlk çağlardan itibaren insanoğlunun bilgiyi işleme macerası, canlılara has özellikleri makine ya da elektronik sistemlere kazandırmaya çalışmasıyla devam etmektedir. Bu amaç doğrultusunda çeşitli yollar denemekte ve birçok sistem geliştirmekte.

İnsanoğlu; canlı özelliklerinden olan görme duyusunu makinelere kazandırma yolculuğunda, canlıların sinir sisteminden yararlanarak modellediği Yapay Sinir Ağlarını kullanarak görüntü işleme, doğal dil işleme, medikal uygulamalar, askeri projeler gibi daha pek çok alanda gelişme kaydetmiştir.

Bir sürücünün seyir halindeyken yollara konulmuş olan trafik işaretlerinin doğru bir şekilde tanımlanması hayati öneme sahiptir. Ayrıca işareti tanımlamak üzere sürücünün harcayacağı zaman ve dikkat birçok faktörden de etkilenmektedir. Hava şartları, trafik yoğunluğu, trafik işaretinin fark edildiği açı gibi birçok faktör sürücünün doğru işareti tespit etmesinde etkilidir.

Bu noktada sürücülere sürüş ve park etme gibi işlevlerde yardımcı olan ADAS (Advanced driver-assistance systems) gelişmiş sürücü destek sistemleri geliştirilmektedir. Bu sistemlerin içinde barındırdığı özelliklerden bir tanesi de TSR (Traffic sign recognition) trafik işareti tanıma sistemleridir. Trafik işareti tanıma için çeşitli algoritmalar vardır. Gelişmiş bilgisayar görüşü ve sinir ağları sayesinde trafik işaretlerini tanıyan sistemler geliştirilebilir (ADAS, 2020).

Sürüş güvenliğini arttıracak teknolojiler yoğun ilgi görmekte ve bu konuda firmalar müşteri memnuniyetini sağlamak adına sürücü destek sistemleri (ADAS) geliştirmektedirler. Bu sistemler tarafından desteklenen araçlara olan rağbet ve piyasa değerlenmesi otonom araç üreten Tesla şirketine yansımış durumda. (Tesla 100 milyar dolarlık piyasa değeriyle dünyanın 2. en büyük otomotiv şirketi oldu, 2021)

Mayıs 2022'den itibaren AB'de (Avrupa Birliği) ISA (Intelligent Speed Assistance) (Briefing: Intelligent Speed Assistance (ISA) | ETSC, 2017.) akıllı hız yardımı sisteminin bütün arabalarda zorunlu olması beklenmektedir. (EUR-Lex, t.y) Hız sınırı işaretlerini tanımayı sağlayan bu sistemin arabalarda bir standart haline getirilmesi üzerinde çalışılan konunun aslında ne kadar önemli olduğuna ait ciddi ipuçları vermektedir.

Bu çalışma kapsamında derin öğrenme yöntemleri kullanılarak özellikle araçların hareketli ve yollarda hız durumları da dikkate alınarak trafik işaretlerinin gerçek zamanlı olarak tanınması üzerine durulmuştur. Bu noktada son yıllarda nesne tanıma alanında hızları ve yüksek FPS'teki (saniyedeki kare sayısı) başarıları ile öne çıkan YOLOv5(Ultralytics, 2021) ve FCOS (Fully Convolutional One-Stage Object Detection) (Tian, Shen, Chen & He, 2019) algoritmasını kullanan Nanodet (Nanodet, 2020) kullanılmıştır. Mobil platformlarda sinir ağı çalıştırmamızı sağlayan ncnn (Tencent:ncnn, 2021) altyapısı kullanılarak bir mobil uygulama üzerinde örnek durumlar tartışılmıştır.

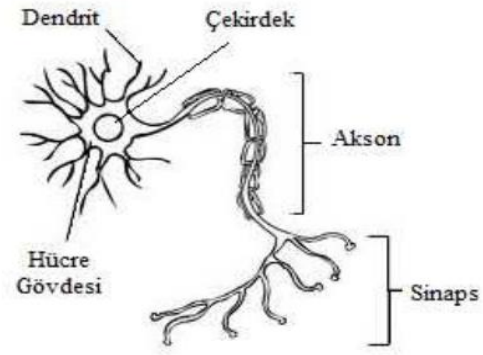
Yapılan çalışmaya mobil uygulamanın da dahil edilmesinin nedeni; yapay sinir ağlarının gerektirdiği hesaplama gücünün taşınabilir cihazlar tarafından karşılanmaya başlandığı günümüzde, eğitilen modellerin uygulanma aşamalarında ne gibi zorlukların olacağı, kısıtlı imkanlar söz konusu olduğunda güvenirliklerinin ne denli etkilendiğiyle ilgili çıkarımlarda bulunmak ve ayrıca trafik işaretlerini tanıyan bir sistemin taşınabilirliğini sağlamaktır.

BÖLÜM 2

2. Literatür Araştırması

2.1. Yapay Sinir Ağları

Yapay sinir ağlarının tarihine kısaca değinmek gerekirse ilk model 1943 yılında, Warren McCulloch ile Walter Pitts tarafından gerçekleştirilmiştir. Biyolojik sinir sisteminden modellenmiştir. Şekil 2.1'de sinir hücresi ve Şekil 2.2'de yapay sinir ağı yapısı bileşenleri görülmektedir.



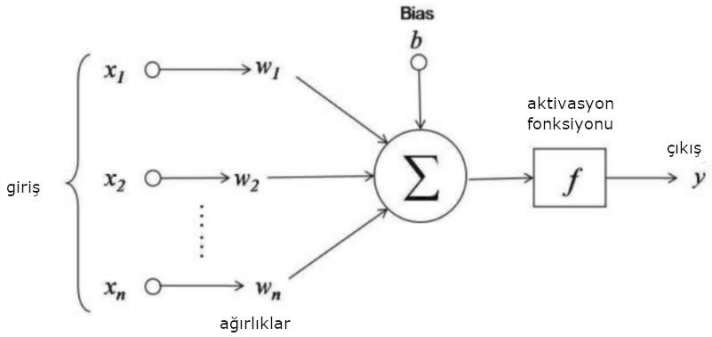
Şekil 2.1. Biyolojik sinir sistemi.

Çizelge 2.1. Biyolojik ve yapay sinir sistemi bileşenleri.

Biyolojik Sinir sistemi	Yapay Sinir Sistemi
Nöron	İşlemci eleman
Dentrit	Toplama fonksiyonu
Hücre gövdesi	Transfer fonksiyonu
Aksonlar	Yapay nöron çıkışı
Sinapslar	Ağırlıklar

Bu bileşenler arasındaki dönüşüm Çizelge 2.1.'deki gibi gerçekleşir. Yukarıdaki model, insan beynindeki süreçlerin taklit edildiği, yapay sinir ağlarının uygulanmasını oluşturan bir yapıdır (McCulloch & Pitts, 1943).

Yapay Sinir Ağları (YSA)'nın en temel taşı hücredir. Bu hücreler içinde bir veya birden fazla aldığı veriler ile çeşitli hesaplamalar yapılır ve ağ modeline göre sonuçta bir çıktı verir. Hücreye bağlı bileşenleri Şekil 2.2'deki gibi görselleştirebiliriz.



Şekil 2.2. Basit yapay sinir ağı yapısı (Prateek, 2017).

Dış ortamdan gelen girdi ilk adımda el yordamı ile verilen daha sonra öğrenmeye bağlı olarak değişen ağırlığı ile çarpılarak toplama fonksiyonuna gönderilir. Toplama fonksiyonu gelen bütün değerleri birleştirerek aktivasyon fonksiyonuna gönderir. Aktivasyon fonksiyonu kendi yapısına göre bir çıktı üretir.

Çeşitli matematiksel modelleri çıkarılan YSA'lar; arıza tespiti, tıp alanında, savunma sanayi, üretim, haberleşme ve kontrol, otomasyon, görüntü işleme örüntü tanıma, , işaret işleme, finansal sistemler, planlama, askeri sistemler, yapay zeka, güç sistemleri, kontrol ve araştırma, gibi birçok alanda etkin şekilde kullanılmaktadır (Erbaş, 2005).

Özellikle 2000 yıllardan sonra bilgisayar donanımında meydana gelen gelişmeler, görüntü işleme ve örüntü tanıma alanlarının birlikte kullanımı ile var olan bir görüntü, video ya da gerçek zamanlı bir kamera görüntüsü üzerinde, istenen nesne tanınabilmektedir.

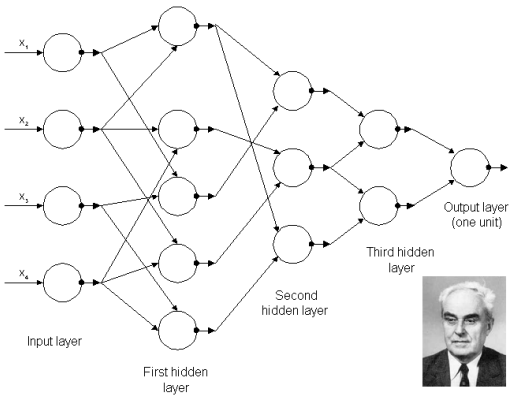
2.2. Derin Öğrenme

Derin yapılandırılmış öğrenim yada derin öğrenme makine öğrenimine dayalı yöntemler ve yapay sinir ağlarını temel alır. Derin öğrenmedeki derin sıfatı yapay sinir ağlarında birden çok katmanın kullanılmasından gelir (Deep Learning, 2020).

Doğrusal olmayan problemlerde kullanılan derin öğrenme denetimli(sınıf) veya denetimsiz algoritmalar olabilir. Verinin temsilinden öğrenmeye dayalı bir yöntem olan derin öğrenmeye bir görüntü için baktığımızda; başlıca özellikler piksel başına yoğunluk değerleri, kenar kümeleri, özel şekiller gibi özellikler olabilir. Derin öğrenmenin diğer bir avantajı, manuel çıkarılan özellikler yerine denetimli-denetimsiz yapı kullanarak özellik öğrenmesi veya hiyerarşik özellik haritası çıkarımı için etkin algoritmalar kullanmasıdır (Song & Lee, 2013; Şeker, Diri & Balık, 2017).

2.2.1. Tarihçe

Çok katmanlı denetimli derin beslemeli yapay sinir ağı için ilk öğrenme algoritması Ivakhnenko ve Lapa tarafından 1966'da yayınlanmıştır (Ivakhnenko & Lapa, 1966). Her katmanda, en iyi özellikler seçilip bir sonraki katmana iletilmektedir. Oluşturulan ağın eğitilmesi en küçük kareler yöntemi benimsenmiştir (Şeker vd., 2017).



Şekil 2.3. Bilinen ilk derin ağ (Dettmers, 2015; Şeker vd., 2017).

Yapılan ilk genel öğrenme algoritmasından sonra yapılan çalışmaları sıralayacak olursak aşağıdaki gibidir.

- Fukushima “Neokognitron” 1980 Derin Öğrenme mimarisi (Fukushima, 1980).
- Yann LeCun Elle yazılmış posta kodu okuma (LeCun, 1989).
- LeNet El yazısı rakamlarını tanıma (LeCun, 1989).
- Wake-sleep algoritması ile sinir ağının eğitilmesinin mümkün olması (Hinston, Dayan,Frey & Neal 1995).
- Tekrarlayan sinir ağlarında "Derin Öğrenme" 2000 yılında Igor Aizenberg ve arkadaşları tarafında ifade edilmiştir.
- Geoffrey Hinton 2006 yılında yayınlanan bir çalışmasında ileri beslemeli yapay sinir ağının herhangi bir aşamasının ağırlıklarını güncelleyebildiği, ardından denetimli geri yayılım yöntemi ile ayarlama yapabileceklerini göstermiştir (Hinton, vd., 2006; Şeker vd., 2017).

Yukarıdaki gelişmeler ışığında Google, Facebook ve Microsoft gibi büyük teknoloji firmaları derin öğrenme alanına yatırım yaptıkça derin öğrenme alanındaki gelişmeler bir hayli hızlanmıştır (Şeker vd., 2017).

2.2.2. Nesne Algılama

Nesne tanıma problemlerinde derin öğrenme algoritmaları oldukça başarılıdır. 2000 yılı başlarından itibaren görüntü işlem alanında Viola-Jones algoritması ve geliştirilen Opencv açık kaynak kütüphanesi ile görüntü işleme alanında inanılmaz hızlı gelişmeler meydana gelmiştir.

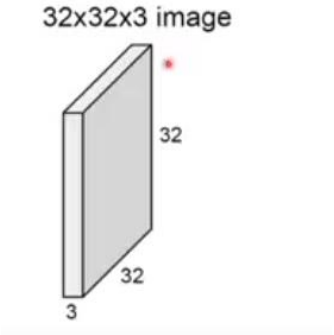
Nesne tanıma problemlerinde derin öğrenme algoritmaları oldukça önemlidir. Nesne tanıma temelde bir sınıflandırma problemidir. Öncelikle ağımızın nesneyi tanıyabilmesi için danışmanlı öğrenme yapmış olması gerekir. Viola-Jones algoritmasının başarısı ve beraberinde GPU kullanımı ile derin öğrenme algoritmalarının görüntü tanımada kullanımı çok başarılı sonuçlar vermiştir. Derin öğrenme algoritmalarından olan Convolutional Neural Network (CNN) (Konvolüsyonel (Evrişimli) Sinir Ağları) algoritmasının 2012 yılında yakaladığı başarı nesne tanımaya yeni bir soluk getirmiştir. Bu gelişme ile artık derin öğrenme ile görüntü içinde nesne algılamanın önü açılmış ve bugün otonom araçların ilerlerken çevreleri ile etkileşim içinde güvenli sürüşünden bahsetmek mümkün hale gelmiştir.

Derin öğrenme algoritmalarıyla nesne tanıma algılama yapılırken karşımıza çıkan kavramlara şöyle bir göz atacak olursak.

2.2.3. Konvolüsyonel (Evrişimli) Sinir Ağı (CNN)

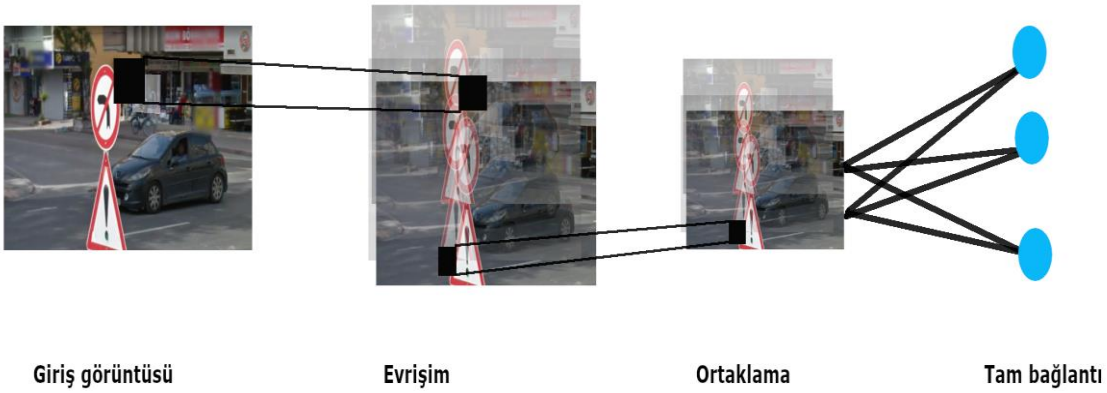
Resmin her pikseline belirlediğiniz boyutta ve sayıda filtre ile bakmamızı sağlar. Adının konvolüsyon olması, matematikteki matris konvolüsyon çarpımından gelmektedir. Bu katman ile resimde yüzlerce farklı minik patern bulunur. CNN, görüntünün bir seferde küçük parçalar halinde işlenmesiyle ilgilenir.

Normal sinir ağları tek vektörlü bir yapıdadır ve bu özellik ulaşılması gereken nesne özelliklerini engelleyen bir özelliktir. Normal sinir ağları görüntülere iyi derecede ölçeklenemez çünkü CNN katmanları 3 boyutta düzenlenmiş nöronlara sahiptir. Yani 32 x 32 boyutunda bir resmimiz olduğunu varsayalım. Normal bir yapay sinir ağında denediğimizde 1024 px bir veri girişi elde ederiz yani resmi düzleştirmiş oluruz fakat nesnelerin tanınabilmesi için derinlik gibi pek çok özelliğe bakmamız gerekecek işte bu durumda resmi CNN ağına verdiğimizde RGB kanalları işin içine gireceğinden yeni elde edeceğimiz veri 32 x 32 x 3 şekline dönüşecektir. Bu durumu Şekil 2.4'te görüldüğü gibi küpe benzetebiliriz.



Şekil 2.4. 3x vektör.

Konvolüsyonel (Evrişimli) sinir ağları yapısı Şekil 2.5'te görüldüğü gibidir. Bununla beraber karmaşık nesne tanımlama problemlerinde daha fazla katman gerektirecektir.



Şekil 2.5. Geleneksel CNN yapısı (Amidi & Amidi, 2018).

Evrişimli yapay sinir algoritmaları görüntü, ses işleme, biyomedikal problemler, doğal dil işleme gibi farklı birçok alanda uygulama alanı bulmaktadır. Görüntü işleme alanında MNIST verileri üzerinde Cireşan yaptığı uygulamada, Evrişimli yapay sinir ağları ile hata oranı %2'lere kadar azaltabilmiştir (Cireşan, Meier, & Schmidhuber, 2012).

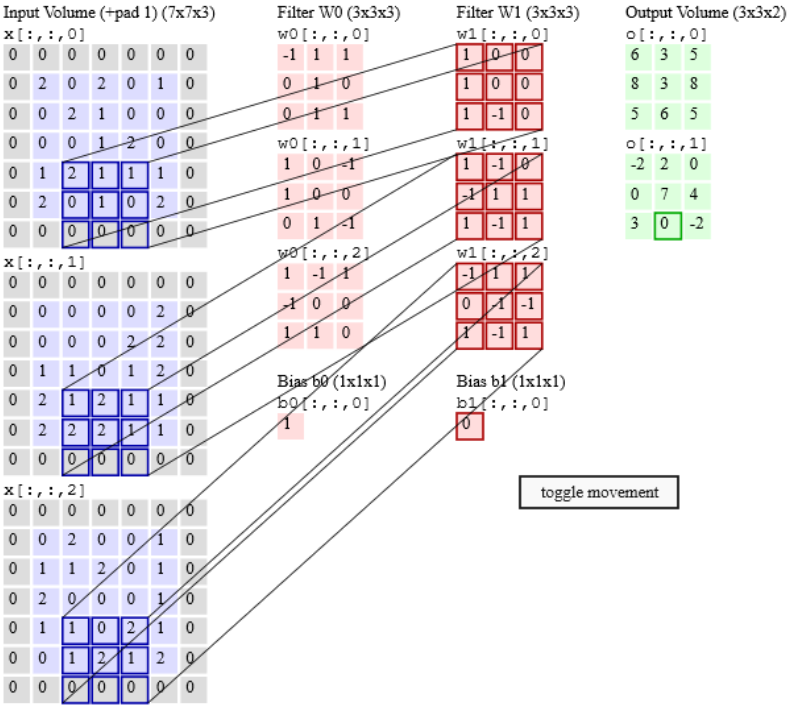
Burada bir parantez açmak gerekirse MNIST veri kümesi nesne tanıma dünyasının "Merhaba Dünya"'sıdır. Bu veri kümesi üzerinde 0-9 a kadar olan rakamların 6000 adet elle yazılmış bir veri kümesi mevcuttur. Genellikle tasarlanan bir yapay sinir algoritması bu veriler üzerinde denenir.

Evrişimli yapay sinir ağı modelleri çeşitli doğal dil işleme problemleri için de etkilidirler. Örnek olarak anlamsal ayrıştırmada (Grefenstette & Blunsom, 2014) arama sorgusu elde etmede (Shen, He, Gao, Deng, & Mesnil, 2014), cümle modelleme (Kalchbrenner, Grefenstette, & Blunsom, 2014), sınıflandırma problemlerinde mükemmel sonuçlar elde edilmiştir (Şeker vd., 2017).

2.2.3.1. Konvolüsyonel (Evrişim) katmanı (CONV)

2x2, 3x3, 5x5 gibi farklı boyutlardaki filtrelerin, bir önceki aşamadan gelen resimlere evrişim işlemini gerçekleştirerek sonuç verisi oluşturduğu katmandır. Özellik

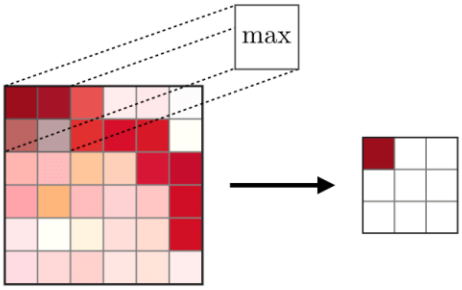
haritası ile her filtreye özgü özellikler keşfedilir. Bu katmandaki ağırlıklar her adımda güncellendiğinde öğrenilen özellik haritası eğitim boyunca özelleşir. Böylelikle verinin hangi bölgelerinin önem taşıdığı anlaşılır. Bu durum Şekil 2.6'te gösterilmiştir.



Şekil 2.6. 7x7x3 özellik haritası (CS231n, 2017).

2.2.3.2. Havuzlama Katmanı (Pooling)

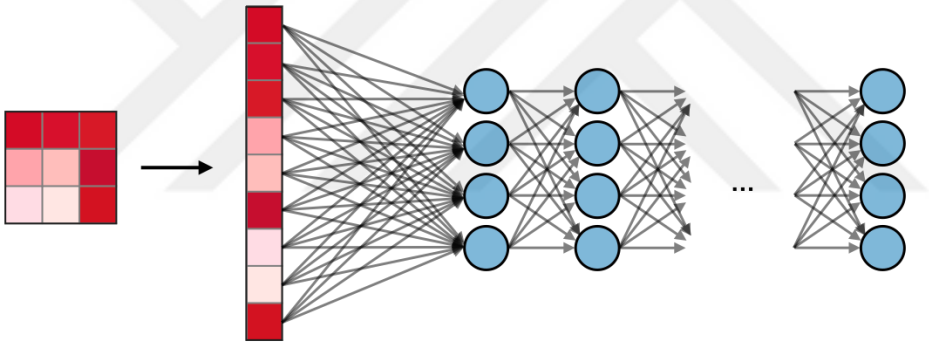
Aşağı örneklem olarak da adlandırılan bu katman evrişim katmanından sonra alınan girdiyi küçültmek için kullanılır. Sonradan uygulanacak filtre boyutuna göre genellikle en çok kullanılan maksimum havuzlamayı kullanılır. Bu durum Şekil 2.7'de gösterilmiştir.



Şekil 2.7. Havuzlama katmanı (CS231n, 2017).

2.2.3.3. Tam Bağlantılı (Full Connected Layer)

Sınıflara ait puanların optimize edildiği ve tüm sinir ağı nöronlarının birbirine bağlandığı katmandır. Burada gelen girdilere göre olası sınıflara ait son sonuçları verir. Bu durum Şekil 2.8'de gösterilmiştir



Şekil 2.8. Tam bağlantılı katman (Amidi & Amidi, 2018).

Klasik anlamdaki yapay sinir ağına benzer bir durum vardır burada. En doğru ağırlıkları belirlemek için eğitim boyunca geri yayılım kullanır ve sonuç olarak verilen görüntüde ki nesnenin sınıfı belirlenmiş olur.

2.2.3.4. Kesiştirilmiş Bölgeler(IoU)

Model eğitimlerinde eğitim verisinin yanında bir de test verisi veririz. Modelimiz bir nesneyi öğrenirken ayrıca ne kadar öğrendiğine dair bilgi edinmek için aynı nesneyi test verisinde test eder. Buna kesin referans denir. IoU hesabı, iki dikdörtgenin kesiştikleri alanın (intersection) bu iki dikdörtgenin bileşiminin (union) alanına bölümü olarak hesaplanır. Tahminin ne kadar doğru olduğuyla ilgili bilgi verir. Şekil 2.9'da gösterildiği şekli ile verilen test verisi üzerinde yapılan tahmin ve kesin değer üzerinden IoU hesaplanarak modelin doğru tahmin oranı hesaplanmaya çalışılır.



Şekil 2.9. Kesiştirilmiş bölgeler (IoU).

2.2.3.5. Öneri Kutuları (Anchor Boxes)

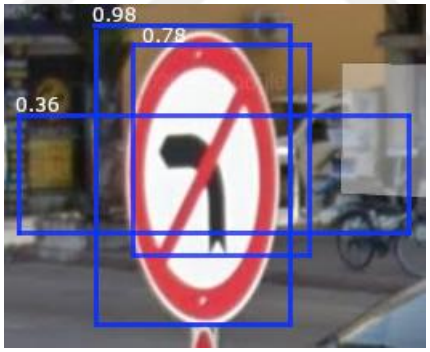
Sinir ağının görüntüde ki nesneleri yakalayabilmesi için görüntü üzerinde farklı boyutlarda pencereler oluşturup bu alanlarda ki nesnelerin tanınması işlemi yapması gerekecektir. Bütün bir resmi düşündüğümüzde burada nesneyi yakalamak için büyük-küçük birçok pencere oluşturması gerekecektir.

Günümüzde pek çok nesne algılama yöntemi (Fast-RCNN, YOLOv3, SSD, RetinaNet, vb.) bu yöntemi kullanır. Nesne ile örtüşen sınırlayıcı kutuları tahmin etmek için kullanılan bir tekniktir. Uygulamada, ağın aynı anda birden fazla kutuyu tahmin etmesine izin verilir, Örneğin, ilk tahmin potansiyel olarak verilen bir sınıfın sınırlayıcı bir kutusudur, ikincisi ise farklı bir sınıfın başka bir sınırlayıcı kutusudur. Bölge tabanlı algoritmalarda önceden belirlenmelidirler. Her ne kadar kayan noktalı nesne tespiti

sorununa çözüm getirmiş olsa da öneri kutusu kullanmadan nesne tanıma algoritmaları da son dönemlerde hızları ile öne çıkmaktadırlar (Amidi & Amidi, 2018).

2.2.3.6. Maksimum Olmayan Bastırma (NMS)

Birkaç farklı öneri kutusu, Şekil 2.10'da olduğu gibi aynı nesneyi kapsıyor olabilir ya da aynı alan içinde farklı nesnelere işaret eden farklı boyutlarda önerilmiş kutu bulunabilir. Bu noktada nesneyi tam olarak gösteren kutuyu seçip diğerlerini elemek adına maksimum bastırma tekniği kullanılır. IoU tahmini 0.7 dan düşük olan kutuları eledikten sonra kalan kutuları sınıfa ait en yüksek IoU'ya sahip kapsayıcı kutuyu bırakana kadar eleyerek seçer. Burada ki döngünün sınıflar üzerinde yapıldığı unutulmamalıdır aksi durumda başka sınıfa ait bir kapsayıcı kutu atılmış olur. NMS, hesaplama açısından çok pahalıdır: n kutu varsa, algoritma O (n^2) karşılaştırmaları gerçekleştirir ve özellikle giriş görüntüsü büyükse ve işlemci zayıfsa, hızla ciddi bir darboğaz haline gelir.



Şekil 2.10. Maksimum olmayan bastırma (NMS).

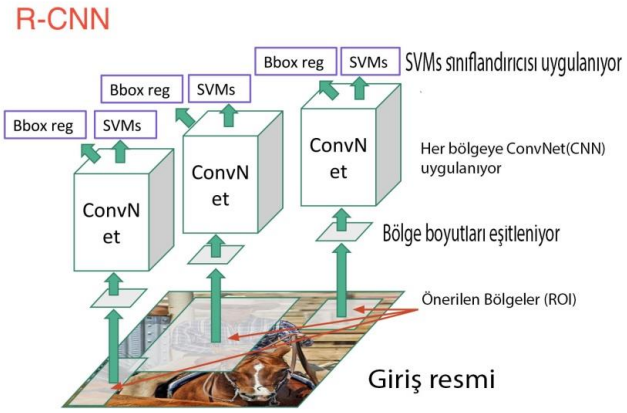
2.2.4. Öneri Kutuları Tabanlı Nesne Tanıma Algoritmaları

Klasik anlamdaki konvolüsyonel nesne tanıma algoritmaları (Alex Net, ZF Net, GoogLeNet, Microsoft RestNet) nesne tanıma dünyasında önemli başarılar elde etmiş olsalar da gerçek zamanlı nesne tanımada ki yavaşlıkları ve gerektirdikleri hesaplama gücü nedeniyle çalışmamızın dışındadırlar. Gerçek zamanlı nesne tanımadaki başarılarının yanında getirdikleri bölge önerisi sayesinde nesne tanımadaki hızları ile öne çıkan algoritmalar şunlardır.

- RCNN
- FAST RCNN
- FASTER RCNN
- SSD
- MASK RCNN
- YOLO(versiyon5)

2.2.4.1. R-CNN, Fast R-CNN ve Faster R-CNN Algoritmaları

Şekil 2.11'de yapısı görülebilecek olan R-CNN (Region-based CNN) bölge önerisi kullanan algoritmadır. R-CN'de buna Region Of İnterest (ROI) denir. Görüntü üzerinde yaklaşık olarak 2000 bölge önerisi üzerinde nesne araması yapar. Bu önerdiği her bir bölgenin üzerinde CNN algoritmasını çalıştırır.



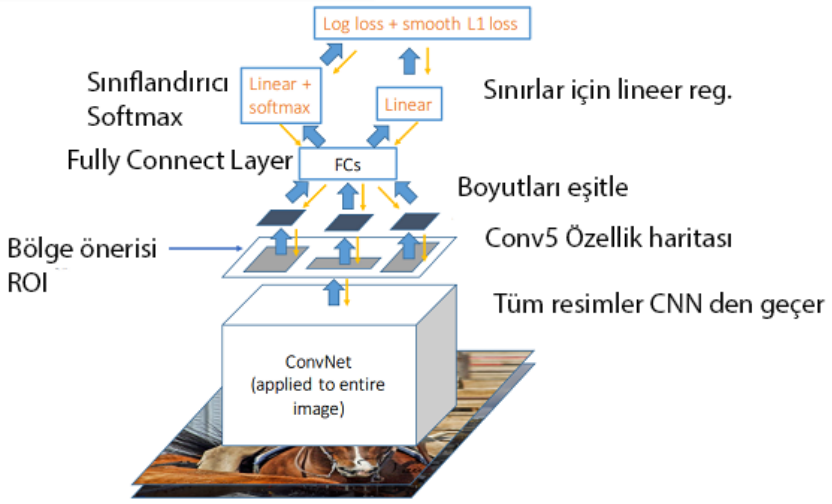
Şekil 2.11. R-CNN mimarisi.

SVMs: Support Vektör Machine Gözetimli Öğrenme makine öğrenimi algoritması. Fast R-CNN ve Faster R-CNN'de kullanılmıyor.

Bbox reg: Lineer regresyon ile nesnenin tam olarak bulunduğu bölgenin tespiti yapılır.

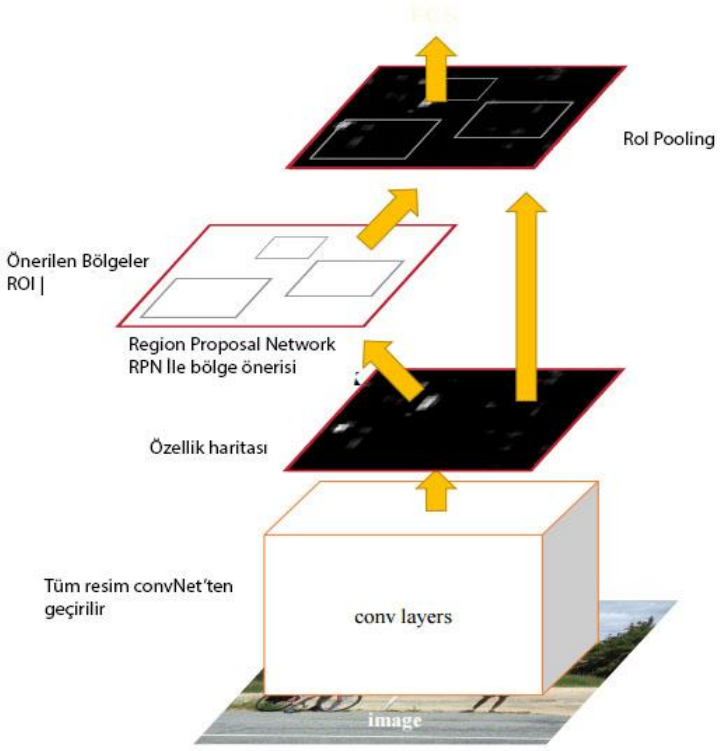
R-CNN'nin eğitim süresi 84 saat ile oldukça yavaştır. Ayrıca Softmax sınıflandırıcı ile ağ eğitilir, Softmax aktivasyon fonksiyonun değeri 0-1 arasında olduğundan kayıplar meydana gelir. Bu algoritmada nesne tanıma Şekil 2.12'de görüldüğü gibi yaklaşık 49 saniyedir ve bu gerçek zamanlı bir nesne tanıma için oldukça yavaştır.

Fast R-CNN, R-CNN'nin yavaşlığını giderme amaçlı ortaya çıkmıştır. R-CNN den temel farkları Şekil 2.12'de görülen özellik haritası kullanması ve sınıflandırma için softmax algoritmasını kullanmasıdır.

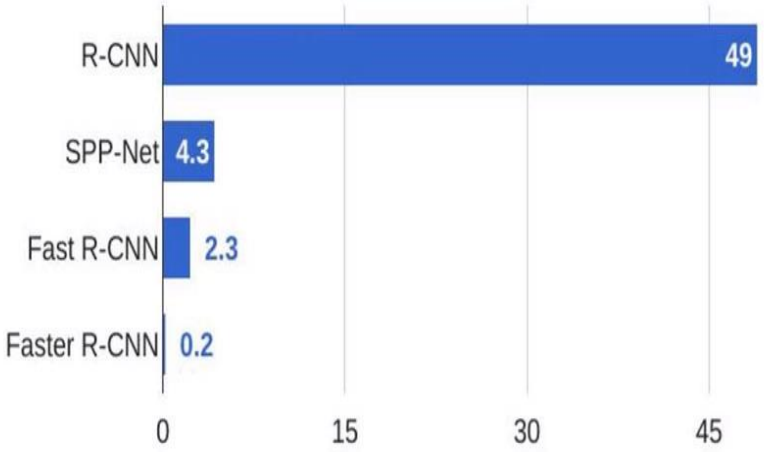


Şekil 2.12. Fast-R-CNN mimarisi.

Faster R-CNN, Fast R-CNN algoritmasının en çok hız kaybettiği bölge önerisi sorununa çözüm olarak geliştirilmiştir. R-CNN ve Fast R-CNN de bölge önerileri sinir ağının dışında yapılırken Faster R-CNN Şekil 2.13'te görüldüğü gibi bölge önerisini de sinir ağından geçirerek yapar bu da oldukça hızlı bir yöntem olarak karşımıza çıkmaktadır. Bölge önerisi ile oluşan yavaşlık böylece giderilmiş oluyor.



Şekil 2.13. Faster R-CNN mimarisi.

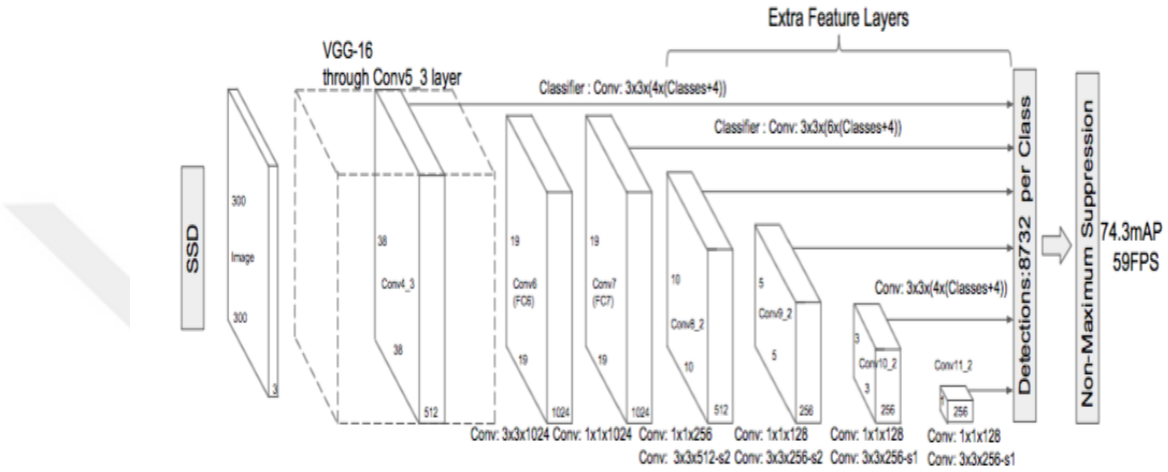


Şekil 2.14. Algoritmaların nesne algılama hızları(saniye).

Şekil 2.14 incelendiğinde bölge önerisinin sinir ağı ile oluşturulmasının etkisi açıkça gözükmektedir.

2.2.4.2. Tek Atış Çoklu Kutu Algılama (SSD)

SSD (Single Shot MultiBox Detector) bölge önerisi ve sınıflandırma aşamasını tek aşamada birleştirerek Faster R-CNN ‘den daha hızlı bir sinir ağı ile nesne yakalamayı mümkün hale getirmiştir. Bu sayede gerçek zamanlı nesne yakalamakta mümkün hale gelmiştir.

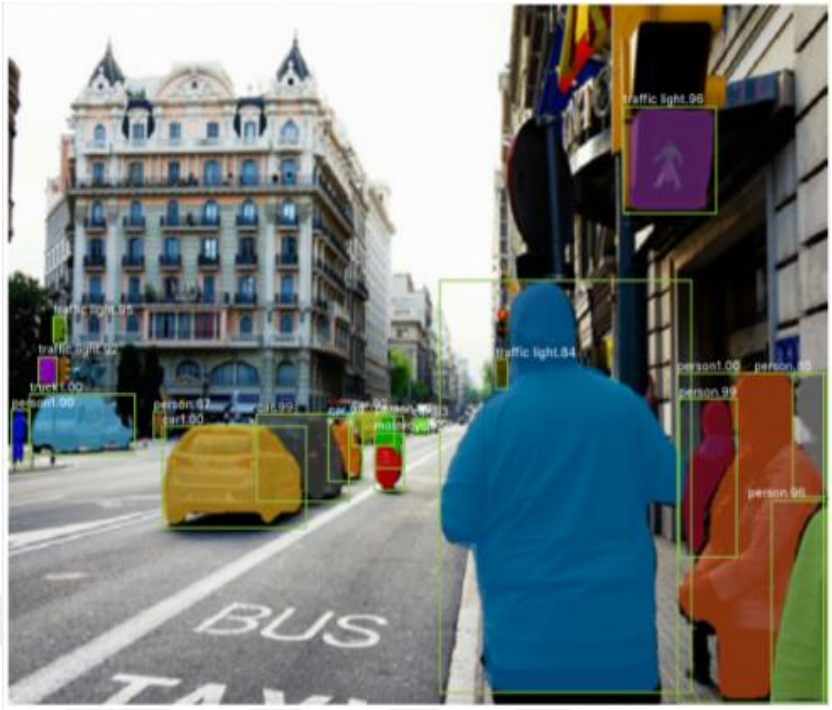


Şekil 2.15 SDD mimarisi.

Konvolüsyonel ağlar ile filtre uygulayarak özellik haritaları çıkarır. Şekil 2.15’te görüldüğü gibi her filtrede boyutları giderek daha çok küçültür bu sayede büyük küçük bütün nesneler tanınabilir. SDD yapısı sayesinde düşük donanımlarda da tercih edilebiliyor.

2.2.4.3. Mask R-CNN Algoritması

Nesne tanıma yapılan görüntüde hem nesnenin bulunduğu alanı kare içine alıp hem de kare içinde ki nesnenin image segmention ile her pikselinin belirlenmesi işlemini yapar. Faster R-CNN kullanır Şekil 2.16’da görüldüğü gibi son aşamada kare içine alıp tanıdığı alanın piksellerini 0-1 ile işaretleyerek o alanın 1 olan kısımlarını maskeler.

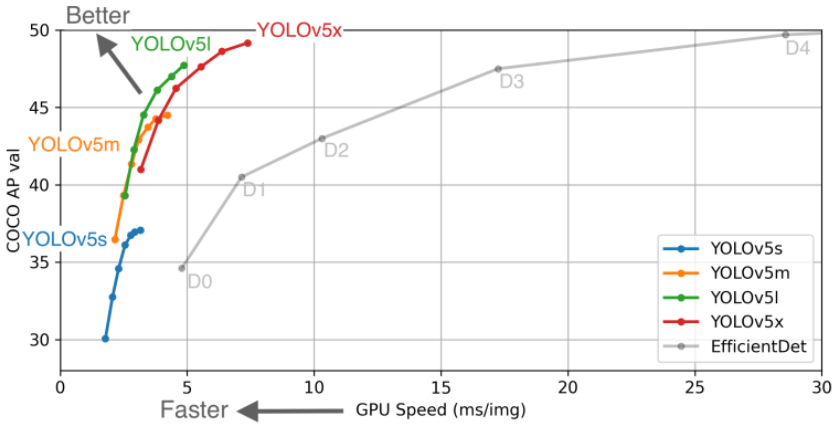


Şekil 2.16. Mask RCNN sonucu

2.2.4.4. YOLO (versiyon 5) Algoritması

YOLO, konvolüsyonel sinir ağlarını kullanarak nesne tespiti yapan bir algoritmadır. İlk olarak 2016 yılında Joseph Redmon ve arkadaşları tarafından yayınlanan YOLO (Redmon vd., 2016) açılımı "You Only Look Once", yani "Sadece Bir Kez Bak"'tir. Bu adın seçilmesinin nedeni algoritmanın nesne tespitini tek seferde yapabilecek kadar hızlı olmasıdır. Çeşitli sürümleri bulunan YOLO algoritmasının güncel ve en hızlı algoritması YOLOv5'tir (Jocher, 2020).

R-CNN gibi bölge önerisi uygulayan algoritmalar öncelikle nesne bulunabilecek alanları belirleyip daha sonra ilgili bölgelerin her birinde CNN sınıflandırıcıları yürütür. Bu işlem her ne kadar yararlı sonuçlar elde etse de bir görüntü iki kez işlemden geçtiği için işlem sayımız artıyor ve saniyede işlenen sahne sayımızda azalmalara neden olmaktadır.



Şekil 2.17. YOLOv5 algoritmasının hızı (Jocher, 2020).

Şekil 2.17'ye bakıldığı zaman YOLOv5'in COCO dataseti (AP 50) son derece hızlı olduğu görülmektedir.

YOLOv5 algoritması için bu tezin hazırlandığı an itibariyle herhangi resmi bir makale çalışması bulunmamakla beraber bu çalışmada değerlendirilmesinin nedeni yayınlanan kod deposunda Şekil 2.17'daki gibi doğruluk oranını sağlaması ve YOLOv5-s modelinin model boyutunun 27 mb'lık alan işgal ediyor oluşudur. Mobil uygulama üzerinde de nesne tespiti yapılmak istendiğinden bu model seçilmiştir.

YOLOv5 algoritmasıyla ilgili herhangi resmi bir makale bulunmadığı halde bu algoritmanın öneri kutuları içeren algoritmalar sınıfına dahil edilmesinin nedeni, eğitimden önce tıpkı YOLOv2-3-4 sürümlerinde olduğu gibi anchor ayarlarının yapılmasıdır.

2.2.5. Öneri Kutuları İçermeyen Nesne Tanıma Algoritmaları

Öneri kutularının getirdiği hesaplama yükünden kurtularak hız kazanmak isteyen modellerdir.

- YOLO (versiyon 1) algoritması.
- FCOS algoritması.

2.2.5.1. YOLO (versiyon 1) Algoritması

Öneri kutuları kullanmak yerine YOLOv1, nesnelerin merkezine yakın noktalardaki sınırlayıcı kutuları tahmin eder. Daha yüksek kalitede algılama sağlayabilecekleri düşünüldüğünden sadece merkeze yakın noktalar kullanılmaktadır. Bununla birlikte, sınırlayıcı kutuları tahmin etmek için yalnızca merkeze yakın noktalar kullanıldığından üst üste ya da çok yakın nesnelerde kayıp yaşar bu durum YOLOv2'de getirilen öneri kutusu aşaması ile giderilmiştir.

2.2.5.2. FCOS (Fully Convolutional One-Stage Object Detection) Algoritması

Segmentasyon işlemi gibi benzer işlemler kullanarak nesne tespiti yapan algoritmadır, piksel başına tahmin tarzında nesne tespitini çözmek için tamamen evrişimli tek aşamalı bir nesne detektörüdür. RetinaNet, SSD, YOLOv2-3-4-5 ve Faster R-CNN gibi algoritmalar, önceden tanımlanmış öneri kutularını kullanır. FCOS algoritması öneri kutusu ve bölge önerisi içermez. FCOS, eğitim sırasında üst üste binmeyi hesaplamak gibi (YOLOv1'in yaşadığı sorun) öneri kutuları ile ilgili karmaşık hesaplamaları tamamen ortadan kaldırır. ResNeXt-64x4d-101'e sahip FCOS, tek işlem sonrası maksimum bastırma (NMS) ile tek model ve tek ölçekli testlerde AP'de % 44,7'ye ulaşarak, çok daha basit olma avantajıyla önceki tek aşamalı nesne yakalama modellerini geride bırakıyor (Tian vd., 2019).

2.2.6. Trafik İşaretleri

Trafik işaretlerini tanımak için literatüre baktığımızda karşımıza çıkan bazı çalışmalar aşağıda verilmiştir.

Timofte ve arkadaşları, kamuya açık üç zorlu veri tabanını oluşturmuşlardır. Bu veri tabanları sırasıyla Belçika Trafik İşareti Sınıflandırması (BTSC), İsveç Trafik İşareti ve Alman Trafik İşareti Tanıma Benchmark (GTSRB) veritabanları olarak adlandırılmaktadır (Aghdam vd., 2016).

Cireşan ve arkadaşları 2012 yılında, GTSRB yarışmasında trafik işareti tanımada CNN'i kullanmışlardır. Önerdikleri CNN test görüntülerini %99,46 doğrulukla sınıflandırmayı başarmışlardır (Cireşan vd., 2012).

Shustanov ve Yakimov, 2017 yılında GTSRB veri setini kullanarak 16 trafik işaretini gerçek zamanlı tanımak için derin öğrenme kütüphanesi olan TensorFlow'u kullanarak bir CNN modelini tasarlamışlardır. Sınıflandırma başarısı %99.94 olarak elde edilmiştir (Shustanov & Yakimov, 2017).

Saha ve arkadaşları. 2018, çalışmalarında trafik işaretlerini sınıflandırmak için üç farklı sinir ağı önermişlerdir. Veri seti olarak GTSRB kullanılmıştır. Sınıflandırma sonucunda %99.92 oranında doğruluk elde edilmiştir (Saha, Islam, Khaled, & Tairin, 2019).

Changzhenve arkadaşları. 2016, Çin trafik işaretlerini kullanılarak R-CNN yönteminden faydalanarak trafik işareti tanınması üzerinde bir çalışma gerçekleştirmiştir. Bu sonuç olarak %99 civarında bir doğruluk oranı elde edildiği rapor edilmiştir (Changzhen vd., 2016).

BÖLÜM 3

3. Materyal ve Yöntem

Bu tez kapsamında derin öğrenme modellerinden Yolov5 ve FCOS algoritmasına dayanan nanodet modeli kullanılmıştır. Bu iki modelin eğitimi ve masaüstü ortamdaki tespit hassasiyetleri, mobil uygulamadaki (android) ve jetson nano iot kartındaki tespit hassasiyetleri tartışılmıştır.

Literatüre bakıldığında oluşturulan derin öğrenme modellerinin genellikle yüksek hassasiyet ve hız arasında bir seçim yapmak durumunda kaldıkları görülmüştür. Hızlı çalışan modellerin doğruluk oranlarında belli bir kayıp yaşadıkları ortadadır. Bu çalışmalarda günümüzde gittikçe yaygınlaşan gömülü ya da mobil sistemlerde derin öğrenme modellerinin çalıştırılmaya çalışılması dikkate alınmış derin öğrenme modeli ile tespit edilen trafik işaretlerin taşınabilir bir cihaz üzerinde de çalıştırılmasının daha yararlı olacağı düşünülmektedir.

3.1. Veri Seti

Derin öğrenmede ki en önemli adım veri setidir. Ne kadar iyi bir modeliniz olursa olsun oluşturulan veri setinin probleme uymaması modelin baştan başarısız olmasına neden olacaktır. Trafik işaretlerine dair literatürde pek çok sete rastlanılmış olsa da (GTSRB: Alman Trafik İşaretleri Seti gibi) ülkemizde özel veri setleri yok denecek kadar az ve tüm işaretleri temsilden şimdilik uzaktır.

Veri setini oluşturmak adına Google Street View, araç içi çekilen videolar ve github üzerinde oluşturulan TTVS (TTVS, 2020) veri setinden yararlanılmıştır. Ülkemizde trafik işaretlerini tanıma amacıyla çalışmalar yapan çeşitli takım ve kişilerin bir araya gelerek oluşturdukları bir veri setidir. İlgili veri setinden yararlanmak için kendi oluşturduğunuz ve etiketlemesini yaptığınız 500 adet görüntü sunmanız

beklenmekte. Kendi imkanlarım ve TTVS'den alınan görüntüler ile toplamda 12629 adet görüntü kullanılmış bu görüntülerin 11366 tanesi eğitim için ve toplamın %1'i olan 1263 adedi de test için ayrılmıştır. Veri seti toplamda 17 adet işareti tanıması için işaretlenmiş ve sınıfların görüntülerde geçme sayıları şu şekildedir:

Çizelge 3.1. Veri setini oluşturan sınıflar.

Sınıflar	Eğitim	Test	Toplam
Dur Tabelası	608	69	677
Durak Tabelası	1330	167	1497
Sağa Dönülebilir veya İleri Gidilebilir Tabelası	499	49	548
Sağa Dönüş Tabelası	452	48	500
Sağa Dönüş Yasak Tabelası	1052	125	1177
Sola Dönülebilir veya İleri Gidilebilir Tabelası	456	57	513
Sola Dönüş Tabelası	428	52	480
Sola Dönüş Yasak Tabelası	2561	298	2859
Park Tabelası	2241	252	2493
Park Yasak Tabelası	576	55	631
Geçiş Yok Tabelası	2377	253	2630
Taşıt Giremez Tabelası	126	11	137
Yol Ver Tabelası	30	7	37
Dönüş Adası Tabelası	119	23	142
Yon Tabelası	218	65	283
Yaya Geçidi	214	21	235
Trafik Işığı	638	38	676

Çizelge 3.1'de sınıflar arasında ciddi sayı farklarının olması, karayollarında bu işaretlerinde aslında bu oranda kullanılmasından kaynaklıdır. Toplanan 500 görüntü ve TTVS veri setinden yararlanılmasına karşılık bazı sınıflarda istenen örnek sayıları yakalanamamıştır. Bu durumun TTVS veri setine eklemeler oldukça düzeleceği düşünülmektedir. Şekil 3.1'de veri setinde bulunan bazı örnekler listelenmiştir.







Şekil 3.1. Veri setinden örnekler.

3.2. Modellerin Eğitilmesi

Gerçek zamanlı nesne tanıma problemlerinde, gerçek zamanlı alınan görüntüde FPS ve modelin gerektirdiği işlem gücü çok önemlidir. Herhangi bir kısıtlama olmadan gerçek zamanlı nesne tanıma yapmak istediğinizde doğal olarak en fazla doğruluk oranına sahip modelin seçilmesi beklenir fakat gerçek hayat problemlerinde herhangi bir kısıtlamanın olmadığı bir durum ancak laboratuvar ortamlarında gerçekleşir. Bu durumdan hareketle problemimiz olan trafik işaretlerini tanımak için model boyutu küçük ve aynı zamanda işleyebilecekleri FPS oranıyla öne çıkan ve mobil cihazlarda da çalıştırılabilecek YOLOv5 (Jocher, 2020) ve FCOS temelli nanodet (Nanodet, 2020) modelleri seçilmiştir. Modellerin işlenişi ile ilgili izlenen yol Şekil 3.2'de belirtilmiştir.

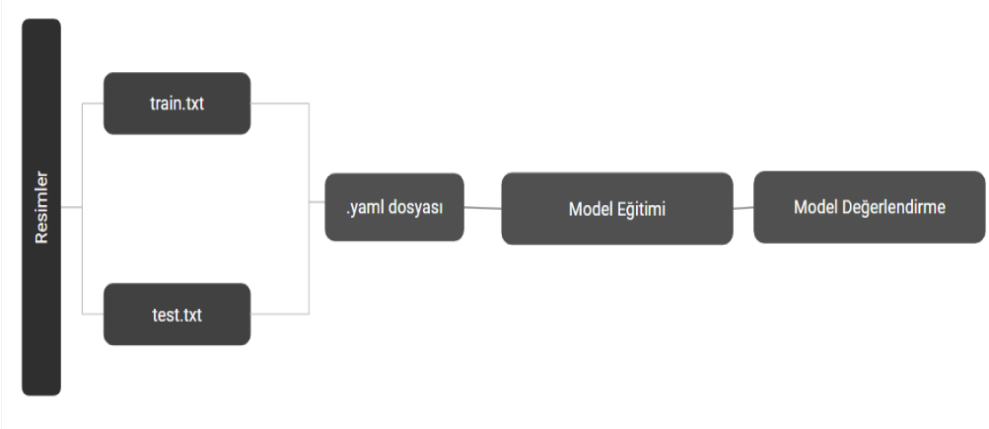
Modellerin eğitimi Intel Core i7 CPU işlemciye 16 Gb Ram ve 4 GB Nvidia Geforce 1060Tİ ekran kartı sahip bilgisayar üzerinde gerçekleştirilmiştir.



Şekil 3.2. Model eğitim diagramı.

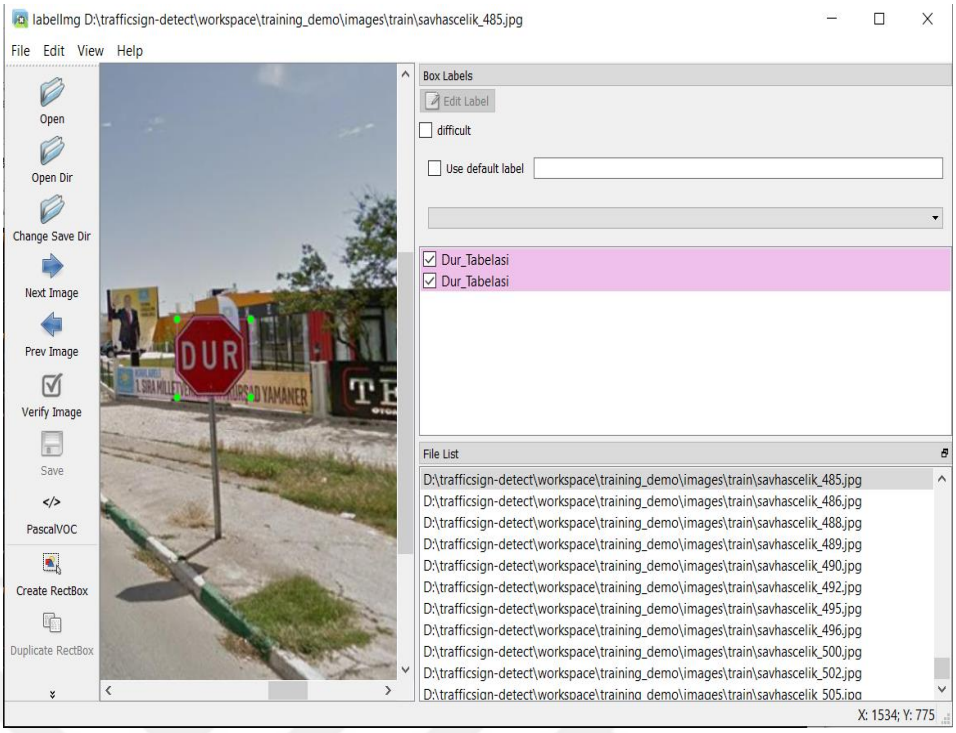
3.2.1. YOLOv5 Modeli

YOLOv5 sürümü PyTorch altyapısını kullanırken önceki tüm sürümler Darknet altyapısına dayanır. Önceki sürümlerinden farklı olarak yüksek FPS oranında nesne tanımı yapabilmesiyle öne çıkan YOLOv5 aynı zamanda eğitim sonunda oluşturduğu modelin boyutu ile tercih edilmede bir adım daha öne çıkmıştır.



Şekil 3.3. YOLOv5 model eğitim süreci.

Veri setini belirledikten sonra Şekil 3.3'de belirtildiği şekilde ikinci aşama verilerin etiketlenmesidir. Etiketleme amacıyla farklı araçlar bulunmakla beraber bu tezde bu işlem için Şekil 3.4'de göründüğü şekli ile LabelImg (LabelImg, 2020) aracı kullanılmıştır.



Şekil 3.4. LabelImg görüntü etiketleme.

Etiketleme aşamalarında veri setinin etiketleme türü PascalVOC –YOLO eğitim yapacağınız modele göre değişmekle beraber bu tezde farklı veri modelleri kullanan iki farklı model kullanıldığından modeller arasında etiketlemeyi çevirebilmek amacıyla PascalVOC etiketleme türü kullanılmıştır.

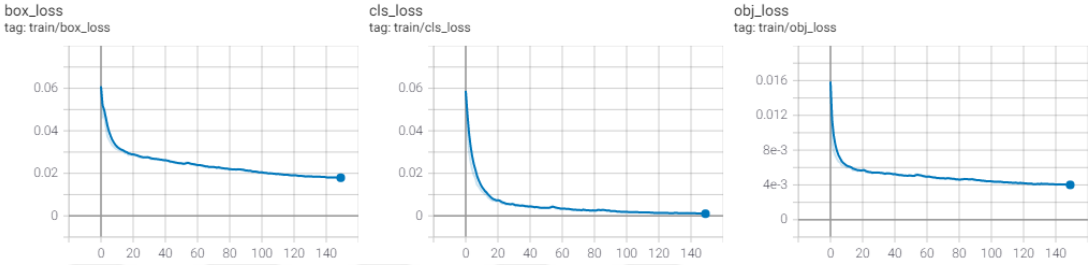
Etiketleme işlemi bittikten sonra yapılması gereken veri setini eğitim verisi ve test verisi şeklinde ikiye bölmektir. Toplam veri setinin yüzde 10-20 oranında ayırmak yeterli olmakla beraber dikkat edilmesi gereken nokta eğitim veri setindeki görüntülerin test verisi içinde bulunmamasına dikkat edilmelidir.

YOLOv5 modelinin eğitim sırasında kullanacağı ayarların bulunduğu yalm dosyası oluşturulur. Burada en kritik noktaların başında sınıf sayısı (nc:17) ve YOLOv2'den beri kullanılan öneri kutuları ayarlarını içermektedir. Öneri kutuları daha önce değindiğimiz gibi görüntüde tespit edilen nesnenin tam olarak konumunu belirlememize yarayan bir algoritmaydı.

YOLOv5 modelinin eğitilmesi sırasında görüntü boyutları 640x640 olarak ayarlanmıştır. Bunu nedenlerinin başında modelin mobil ortamda da test edilmesi

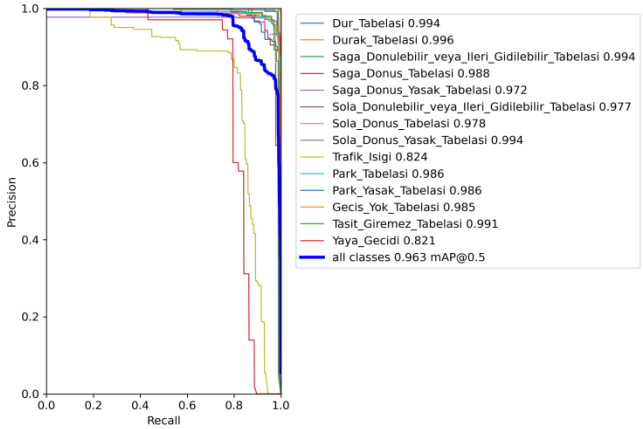
gelmektedir. Görüntünün öncelikle büyük boyutlu olarak eğitime sokulması hem eğitim süresini uzatacak hem de mobilde işlem gücünden kaynaklanan mecburi boyut küçültmeden fazlaca etkilenecektir.

150 dönemde (epoch) gerçekleştirilen eğitim sonunda eğitimin elde edilen hata değerlerinin değişimi Şekil 3.5'de gösterilmiştir. Hata değerleri incelendiğinde dönem boyunca hata oralarının düştüğü ve genel kabul gören 0.02'nin altına indiği görülmüştür.



Şekil 3.5. YOLOv5 model eğitimi kayıp oranı.

Kesinlik (precision) 'tahmin edilen nesneler arasında kaç tanesinin doğru olduğunu hesaplarken hassasiyet (recall) 'nesneye ait örneklerden kaç tanesinin doğru tahmin edildiği' sorusuna cevap veririr . mAP (Mean Average Precision) ortalama keskinlik değeri nesne algılama modellerinin doğruluğunun ölçülmesinde kullanılan bir standarttır.



Şekil 3.6. YOLOv5 mAP grafiği.

Şekil 3.6 incelendiğinde doğruluk oranın sınıflar bazında gayet kabul edilebilir seviyelerde olduğu gözlemlenmiştir. mAP >0.5 durumunda orta ölçekli nesnelerde bir isabet olduğundan söz edilebilir.

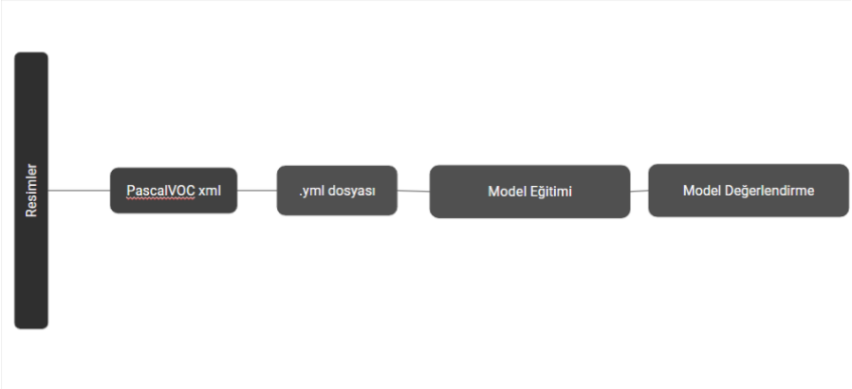


Şekil 3.7. YOLOv5 eğitim sırasında nesne tanıma.

Eğitim sırasında modelin test ettiği Şekil 3.7'de ki gibi görüntülerde bazı nesnelerin doğruluk oranlarının düşük olduğu gözlemlenebilir. Burada kaçıncı dönemde olduğunun önemli olmasının yanında nesneye ait örneklem sayısının da önemi büyüktür.

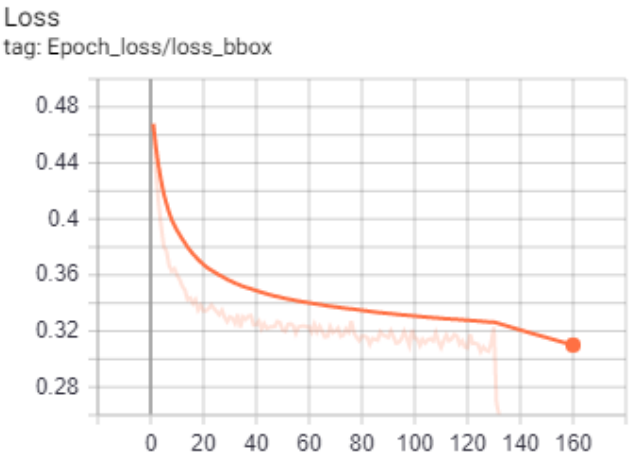
3.2.2. Nanodet Modeli

Gömülü ve mobil sistemlerde yüksek hızlı nesne tanıma yapmak amacıyla ortaya çıkan, FCOS (Tian vd., 2019) algoritmasını temel alan nanodet (Nanodet, 2020) modeli mobil cihazlarda 97 FPS'e gibi bir hıza ulaşabilmekte. Ayrıca model büyüklüğü 1.8 mb ve bu nesne tanıma modelleri için oldukça küçük bir boyut sayılır.



Şekil 3.8. Nanodet model eğitimi kayıp oranı.

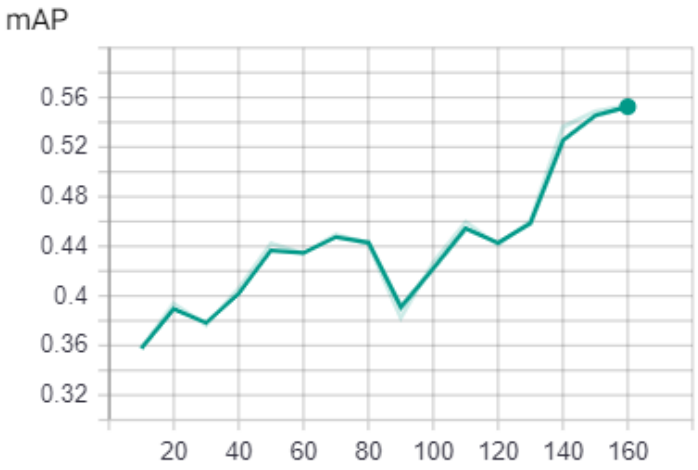
Şekil 3.8'de görüldüğü gibi aslında her model genellikle değişmeyen sıra veri setlerinin etiketlenmesi bir ayar dosyası ve sonrasında eğitime başlanması şeklinde sıralayacağımız bir hiyerarşik düzen vardır. Modelimiz görüntülerin ayrı dosyalarda (eğitim ve test) görüntü barındırılmasını gerektirmekte ayrıca YOLOv5'te olduğu gibi eğitim ve test datalarının adreslerinin listelendiği gibi bir .txt dosyası istememektedir. Ayarların yapıldığı .yml dosyasında eğitim ve test klasörlerinin yerlerinin gösterilmesi yeterli gelmektedir. Bu dosyalarda bulunan .xml dosyalarından kendisi gerekli bilgileri almaktadır.



Şekil 3.9. Nanodet model eğitimi.

Nanodet modelinin eğitilmesi sırasında görüntü boyutları 640x640 olarak ayarlanmıştır.

160 dönemde (epoch) gerçekleştirilen eğitim sonunda eğitimin elde edilen hata değerlerinin değişimi Şekil 3.9'da gösterilmiştir. Hata değerleri incelendiğinde dönem boyunca hata oralarının düştüğü ve genel kabul gören 0.02'ye yaklaştığı görülmüştür.



Şekil 3.10. Nanodet mAP grafiği.

mAP ortalama keskinlik değeri nesne algılama modellerinin doğruluğunun ölçülmesinde kullanılan bir standarttı. Şekil 3.10 incelendiğinde doğruluk kabul edilebilir seviyelerde olduğu gözlemlenmiştir. mAP >0.5 durumunda orta ölçekli nesnelerde bir isabet olduğundan söz edilebilir.

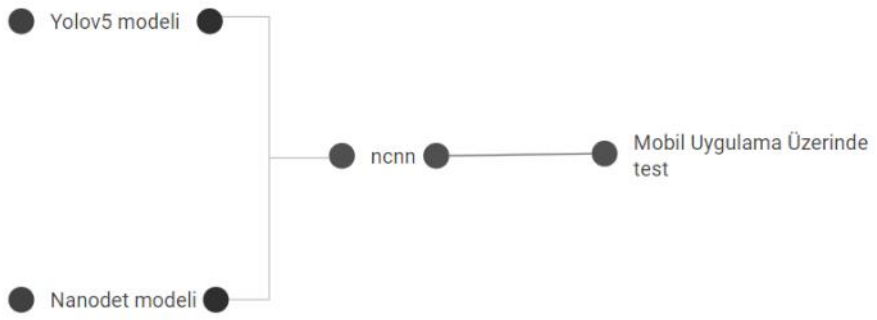


Şekil 3.11. Nanodet örnek nesne tanıma.

Test sırasında nanodet YOLOv5'e göre daha düşük doğrulukla nesne tanıma yapmıştır. Bu durum Şekil 3.11'de çok tanınan trafik işaretinde de(trafik ışığı %50.4) görülmektedir. FPS hızı açısından öne çıkan modellerin aslında doğrulukla ve hız arasında bir seçim yaptıkları söylenebilir.

3.3. Android Uygulaması

Eğitilen modeller eğitildikleri masaüstü ortamının yanında taşınabilirlik açısından test edilmeleri için ncnn altyapısında basit bir mobil uygulamada denenmişlerdir. Nanodet (Nanodet, 2020) demo uygulaması kullanılarak geliştirilen mobil uygulamayla özellikle kısıtlı imkanlarla modellerin FPS hızları ve doğrulukları test edilmek istenmiştir.



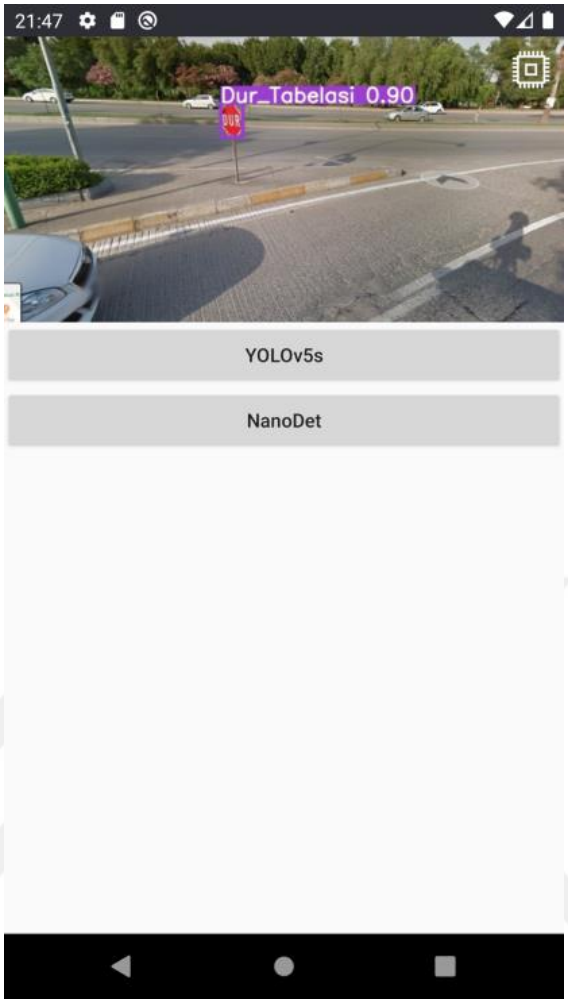
Şekil 3.12. Modellerin ncnn'e dönüştürme.

Eğitilen modellerin android ortamında çalıştırılması için çeşitli işlemlerden geçmesi gerekmektedir. Şekil 3.12'de görüldüğü gibi eğitilen modellerin android uygulamada kullanılmadan önce ncnn kütüphanesine aktarılması gösterilmiştir. Android ortamında yapay sinir ağlarının çalıştırılmasını sağlayan çeşitli kütüphaneler vardır.

Çizelge 3.2. Android ortamını destekleyen kütüphaneler.

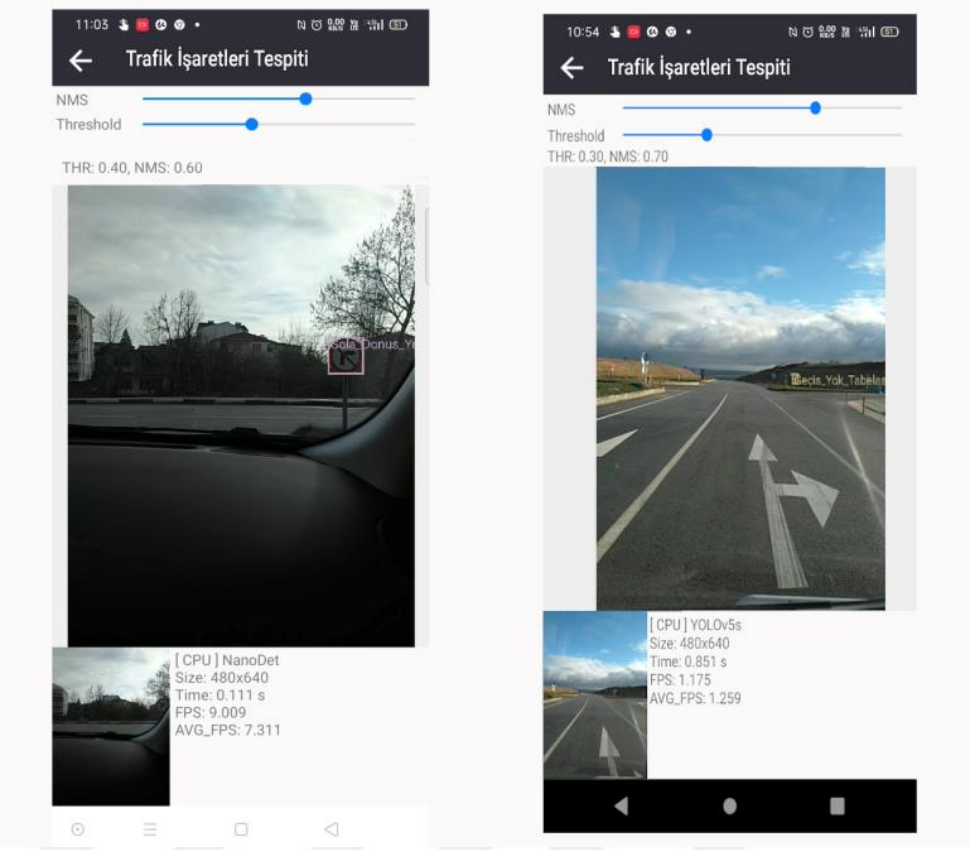
Tensorflow Lite	TensorFlow Lite, TensorFlow'un basitleştirilmiş bir sürümüdür.Akıllı telefon veya Raspberry Pi gibi küçük bir cihaz için tasarlanmıştır.
ncnn	ncnn çerçevesi, Çinli internet şirketi Tencent tarafından oluşturulan hızlı bir çerçevedir. Raspberry Pi'de bulunan ARM çekirdekleri, alternatifleri ve günümüzde hemen hemen her akıllı telefonda bulunan ARM çekirdekleri ile bu kütüphaneyi çalıştırmanız mümkündür
MNN	Alibaba şirketi tarafından oluşturulan MNN oldukça verimli ve hafif bir derin öğrenme çerçevesidir.Gömülü sistemlerde de bu kütüphaneyi çalıştırmanız mümkündür

Çizelge 3.2'de görülen kütüphaneler içinde ncnn'in seçilme nedeni eğitilen YOLOv5 (PyTorch tabanlı) ve nanonet modellerinin ncnn'nin android ortamında yürütülmesi için çevrilmesi gereken .bin ve .param dosyalarına kolaylıkla çevrilebiliyor oluşu ve ayrıca topluluk etkisi açısından oluşan sorunlara ncnn'de daha kolay cevap bulunuyor oluşudur.



Şekil 3.13. Android uygulamasının arayüzü.

Android uygulaması kullanıcıyı eğitilen modeli seçmesi için Şekil 3.13'te görülen hoş geldin ekranı ile karşılamaktadır. Burada eğitilen modellerden YOLOv5 ve Nanodet modellerinden birinin seçilmesinin ardından Şekil 3.14'de görülen ekrana geçmektedir. Bu ekranda da seçilen modele ait bilgiler o anki kamera görüntüsü, modele giden görüntünün boyutu ve kameradan alınan FPS bilgisi bulunmaktadır. Ayrıca nesne tanıma üzerinde önemli etkisi olan maksimum bastırma (NMS) ve nesne tanıma sınır değeri olan (threshold) değerleri de ayarlanabilmektedir. Bu değerler ekran geldiğinde varsayılan değerleri ile başlar



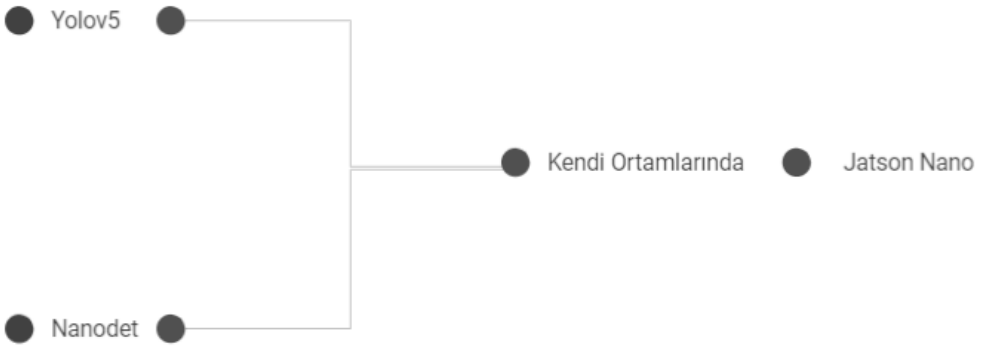
Şekil 3.14. Android uygulamasından örnekler.

3.4. Jetson Nano Uygulaması

Eğitilen modeller android telefon ortamından çok daha güçlü bir şekilde donatılmış Nvida firmasının IoT uygulamaları için piyasaya sürdüğü kart olan Jetson Nano (Jetson Nano, 2020) 4 GB modeli üzerinde denenmiştir. Jetson Nano IoT uygulamaları için geliştirilmiş üzerinde bulunan işlem gücü sayesinde IoT projeleri için sıklıkla kullanılan bir geliştirme kartı olmuştur. Yapısal olarak Rasberry Pi ile benzerlikler taşısa da işlemci ve gpu desteği ile yapay sinir ağlarının kullanımı için ideal bir karttır.

Jetson Nano sunduğu olanakların android telefon ortamından en büyük farkı, ncnn üzerinde modelleri çalıştırmak için yapılan dönüşümün neden olduğu kaybın olmamasıdır. Jetson Nano üzerinde Ubuntu işletim sistemini yürütebilmenizi sağlayan güçlü bir yapıda olduğundan modellerin eğitimi sırasında kurulan kendi ortamlarını bu

kart üzerinde de kurmak mümkündür (Şekil 3.15). Bu nedenle Jetson Nano üzerinde denenen modellerin trafik işaretlerini daha doğru şekilde tanıması beklenmektedir.



Şekil 3.15. Modelerin kendi ortamlarında Jetson Nano üzerinde çalıştırılması.

Şekil 3.16'te görüldüğü şekli ile Jetson Nano kartı ile bilgisayar ortamında eğitimi tamamlanmış modellerin yürütülmesi amacıyla karta bir webcam, kablosuz bağlantı için wifi modülü ve ayrıca tanınan trafik işaretleri hakkında sürücüye geri bildirim verilmesini sağlayan 3 inçlik bir ekran bağlanmıştır. Jetson Nano 5v 4A güç ile çalışan bir kart olduğundan arabanın çakmaklık bağlantı noktasından bir çevirici yardımıyla karta güç sağlanmıştır. Yapılan deneme bir benzetim amacı taşıdığından arabalarda bu tür güç bağlantıları için bırakılan yedek besleme bağlantıları kullanılmamıştır. Ayrıca üzerine bağlanan webcam deneme sırasında serbest bırakılmış, gerçek hayat uygulamasında en iyi konumun dikiz aynası olabileceği ya da arabanın ön kısmında muhafaza edilebileceği düşünülmüştür.



Şekil 3.16. Jetson Nano genel görünümü.



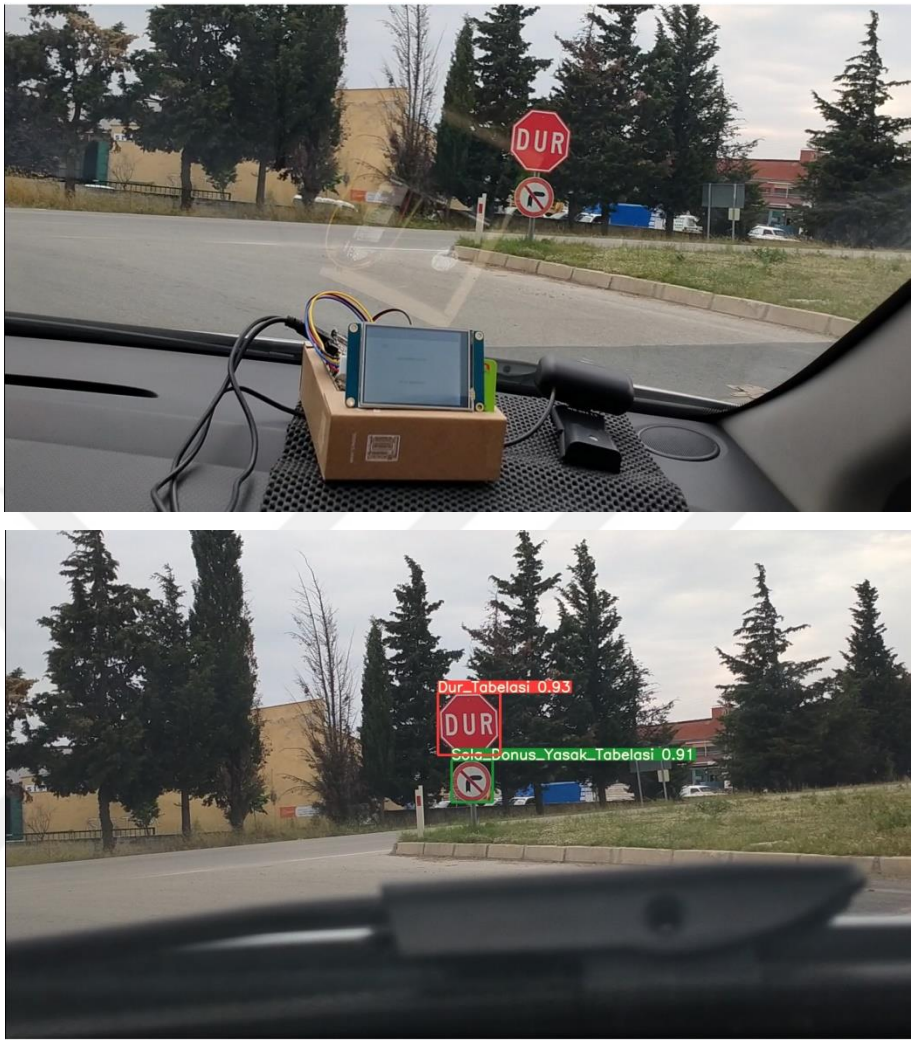
Şekil 3.17. Jetson Nano ekranı genel görünüm.

Şekil 3.17'de görüldüğü gibi kartın ekranında bulunan iki adet modelden istenilen model seçildikten sonra kartın ilgili modele ait geri bildirimleri görülebilir. Asıl amaç model denemeleri olduğundan bu ekranlara ait fazlaca tasarım detaylarına girilmemiş tasarım olabildiğince sade tutulmuştur.



Şekil 3.18. Detay ekranı.

Seçilen ilgili modelin geri dönütleri Şekil 3.18'de ki gibi detay ekranında görülebilmektedir. Tespit edilen işaret ekrana yazılı olarak gelmektedir Kullanılan ekranın kısıtlı hafızası nedeniyle ekrana işarete ait görüntü gönderilmemiştir farklı bir ekran ya da iletişim kanalıyla bu sorun kolaylıkla atlatılabilir, tezin konusunun kapsamına girmediği için bu tezde bu konuya detaylıca girilmemiştir.



Şekil 3.19. Jetson Nano ile örnek tespit.

Şekil 3.19'da görüldüğü gibi sistem arabanın ön tarafına yerleştirilen kamera ile karşılaştığı işaretleri tanımaya çalışmakta. Tabi burada sistemin tanıma başarısının her şeyden önce beslendiği veri setinden etkilendiğini unutmamak gerek.

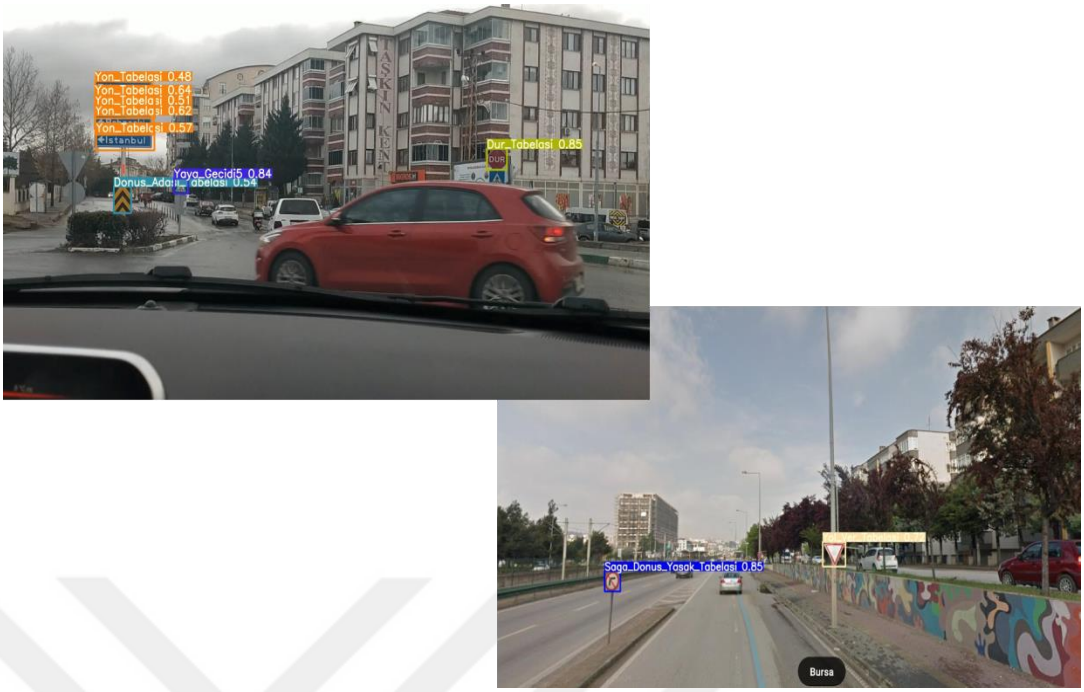
BÖLÜM 4

SONUÇLAR VE TARTIŞMA

Bu tez kapsamında yeni yeni oluşturulmaya başlanan TTVS (Türkiye Trafik İşaretleri Veri Seti) ile eğitilen ve trafik işaretlerini gerçek zamanlı olarak tespit eden uygulama geliştirilmeye çalışılmıştır. Tez kapsamında modellerin farklı ortamdaki hassasiyet ve hız kontrolü için masaüstü ortamda, mobil (android) ortamda ve geliştirme kartı olan Jetson Nano ile denemeler yapılmış ve sonuçları ortaya çıkarılmıştır.

Yapılan çalışmada mAP değerleri karşılaştırıldığında (Şekil 3.6) YOLOv5 modelinin, (Şekil 3.10) nanodet modeline göre daha fazla doğruluk sağladığı görülmüştür. Bunun temel nedenlerinden bir tanesi hız kazanmak isteyen modellerin genellikle doğruluktan feragat ettikleri görülmüştür. Göz önüne alınması gereken nokta karşı karşıya kalınan problemde modelin doğruluğunun ne kadarından vazgeçilebileceğidir. Problemin uygulandığı alanda sistem yeterliliklerinin izin verdiği ölçüde modelin nesne tanımadaki doğruluğu en üst düzeyde seçilmelidir. Fakat nesneleri yüksek doğrulukla tanıyan ama gerektirdiği işlem gücü nedeniyle gerçek hayat problemine uygulanamayan bir modelin varlığı da pek işimize yaramayacaktır.

Modelin eğitildiği ortam olan Core i7 CPU işlemciye 16 Gb Ram ve 4 GB Nvidia Geforce 1060Tİ ekran kartına sahip bilgisayarda yapılan testlerde FPS hızlarının eşit tutulduğu görüntüde YOLOv5 modelinin daha fazla ve daha yüksek bir doğrulukla nesne tanıma yaptığı gözlemlenmiştir.



Şekil 4.1. YOLOv5 örnek işaret tanıma.

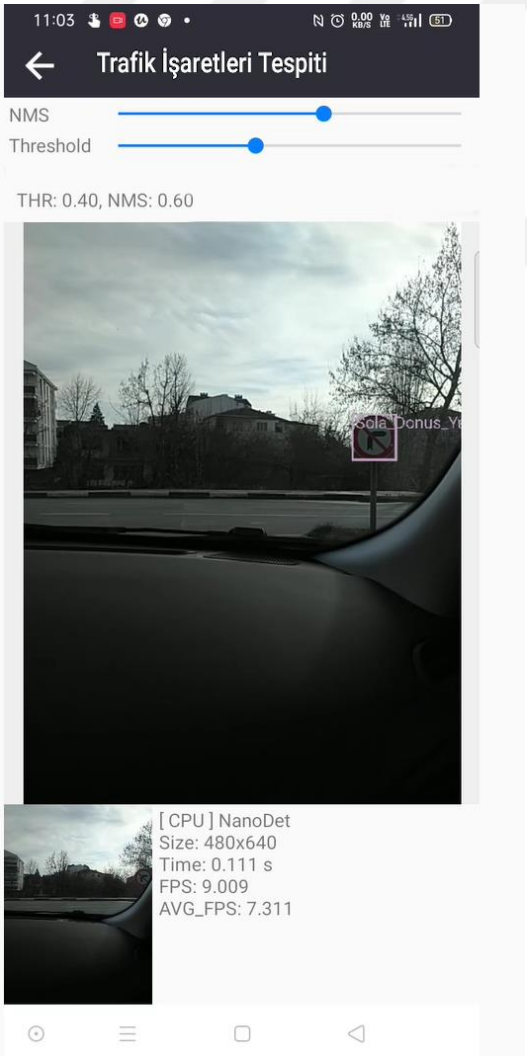


Şekil 4.2. Nanodet örnek işaret tanıma.

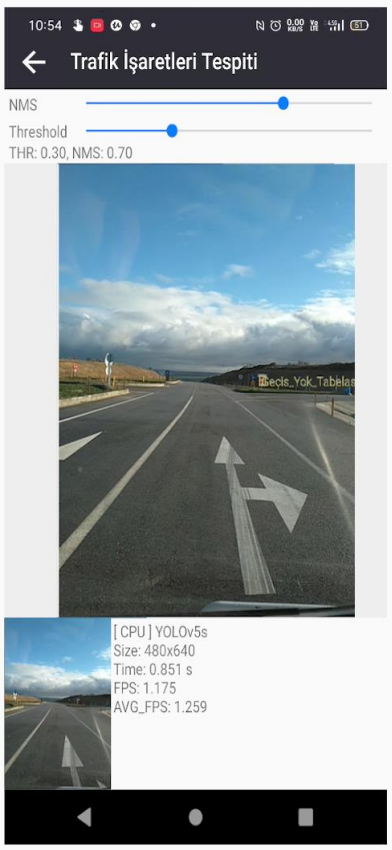
Şekil 4.1 ve Şekil 4.2 incelendiğinde YOLOv5'in nanodet'ten daha fazla işaret tespit ettiği ve doğruluk oranının 2 katından fazla olduğu görülmektedir. Bu farkın asıl

nedenin FCOS ve öneri kutusu farkından kaynaklanıp kaynaklanmadığı bu tezde irdelenmemiş olup bu konunun araştırılması gerektiği ortaya çıkmaktadır.

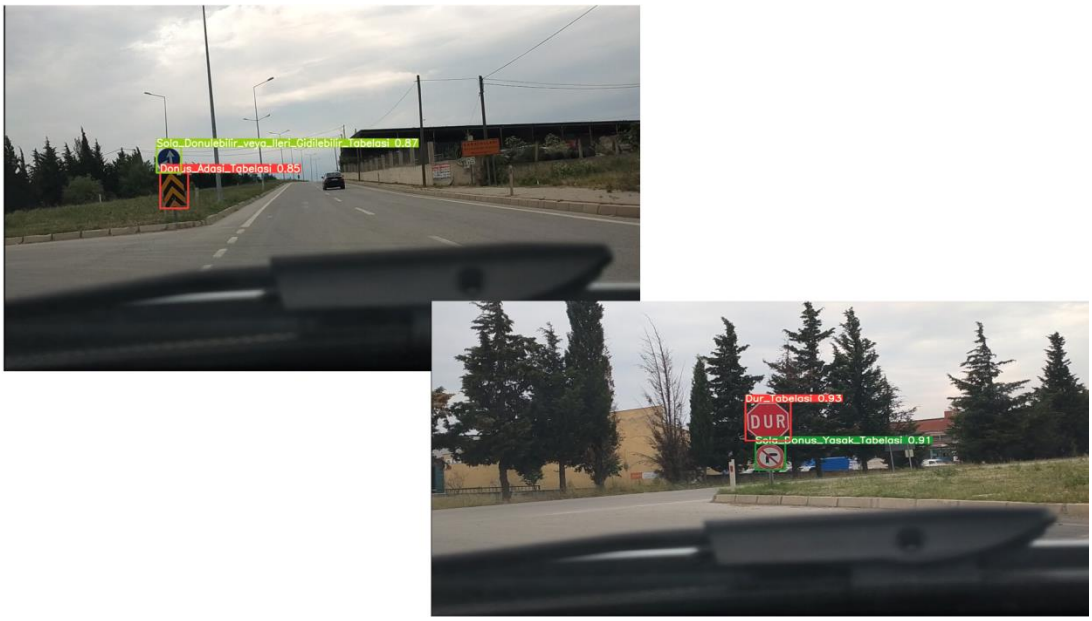
Şekil 4.3 ve Şekil 4.4 incelendiğinde eğitilen YOLOv5 ve nanodet modelleri android mobil uygulamasında test edildiğindeyse FPS hızı açısından öne çıkan nanodet modeli olmuştur. Nanodet modeli ortalama 7.3 FPS hıza ulaşabiliyorken YOLOv5 modeli 1.2 FPS hızında kalmıştır. Hareketli bir araba düşünüldüğünde FPS hızı ortaya konulan problem için hayati öneme sahip olmaktadır. Ortalama 60 km hızla giden bir arabada YOLOv5'in sahip olduğu FPS hızı nesne algılamayı mümkün kılmamışken nanodet modelinin sahip olduğu FPS hızı doğruluk oranı düşük dahi olsa YOLOv5'ten daha bir performans yakalamıştır.



Şekil 4.3. Android uygulamasında nanodet modeli.

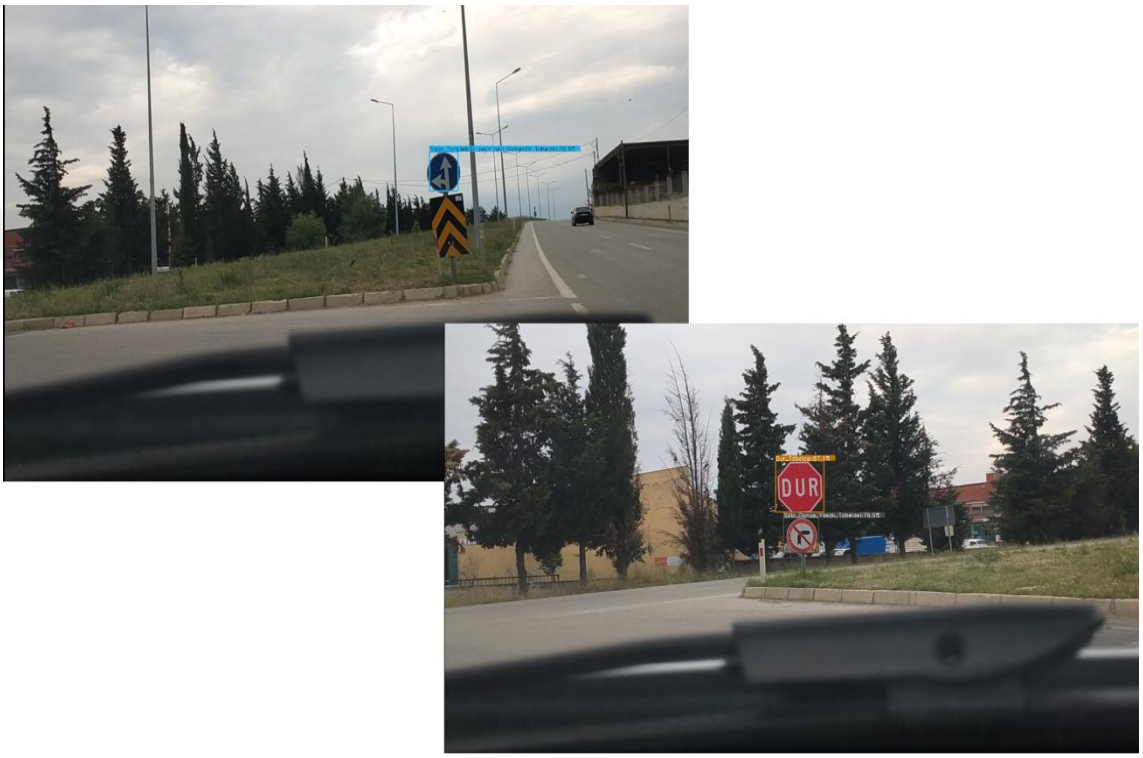


Şekil 4.4. Android uygulamasında YOLOv5 modeli.



Şekil 4.5. Jetson Nano üzerinde YOLOv5 uygulaması.

Şekil 4.5 incelendiğinde YOLOv5 ile eğitilen modelin gayet başarılı ve doğru sonuç verdiği gözlenmiştir. Fakat uygulama sırasında bellek durumuna bağlı olarak YOLOv5 tarafından işlenen görüntünün FPS değerinin istenen düzeyde olmadığı görülmüştür. Bu durum belli hızda seyreden araç içinde işaretlerin tanınmasını güçleştirecektir.



Şekil 4.6. Jetson Nano üzerinde nanodet uygulaması

Şekil 4.6 incelendiğinde nanodet modelinin tanıma oranının istenen düzeyde olmadığı ayrıca YOLOv5 ile aynı veri seti ile eğitildikleri halde yanlış tanıma, bazı işretleri hiç tanımama gibi durumlar görülmüştür. Yapılan testlerde nanodet modelinin FPS hızının istenenin çok çok üzerinde olduğu kameradan görüntü alımında herhangi bir kare atlama gibi bir durumun gözlemlenmemiştir.

Model eğitimleri ve sonrasında yapılan denemelerde öne çıkan önemli bilgiler şunlardır. Yeterli işlem ihtiyaçları karşılandığından YOLOv5 doğruluk ve hız anlamında nanodet modelinin kat kat üzerinde bir performans sergilemektedir. İşlem hacminin kısıtlandığı ortamlarda bizim örneğimizde mobil-jetson nano, YOLOv5 FPS

hızı çok fazla düştüğünden istenilen performansı gösterememiştir. Nanodet modelinin FPS hızı çok çok iyiyken tanıma ve doğruluk oranında YOLOv5'in altında performans göstermiştir.

YOLOv5'in tercih ettiği öneri kutusu algoritması tanıma oranında başarılı olmakla beraber işlem hacminin kısıtlı olduğu durumlarda uygulanmasını güçleştirmektedir. Nanodet modeli işlem hacminin kısıtlı olduğu ortamlarda YOLOv5'ten çok daha hızlı işlem yapmakta fakat nesne tespiti noktasında sağladığı doğruluk oranı tatmin etmemektedir.

Modellerin kullandığı algoritmaların yanında, kamera tabanlı teknoloji kullandığından, çok çeşitli faktörler sistemi daha az hassas hale getirebilir. Bunlar zayıf aydınlatma koşulları, aşırı hava koşullarını ve işaretin kısmen engellenmesini içerir.

Literatürde farklı ülkelere ait veri setleri mevcut olsa da ülkemize ait tam kapsayıcı bir veri seti oluşturma noktasında çalışmaların yapılması gerektiği ve aynı zamanda diğer ülkelerin (örneğin Almanya ve Belçika) veri setleri ile benzer ortamlarda teste tabi tutulmaları gerektiği ortaya çıkmaktadır.

Gelecekteki çalışmalarda mobil sistemlerin özelliklerinin artması yeni nesne tanıma algoritmaları ile TTVS'nin daha fazla örnek içermesiyle ele aldığımız konu daha başarılı sonuçlar elde edebilir. Ayrıca öneri kutusu tabanlı algoritmalar ve öneri kutusu içermeyen algoritmaların doğruluk ve hızları arasındaki farklılıklar ayrıca ele alınarak çalışmalar yapılabilir.

KAYNAKLAR

ADAS. (2020). Advanced Driver-Assistance Systems. *https://en.wikipedia.org/w/index.php?title= Advanced_driver-assistance_systems&oldid=1034822989 (22 Kasım 2020).*

Aizenberg, I. N., Aizenberg, N. N., & Vandewalle, J. (2000). Multiple-Valued threshold logic and multi-valued neurons. In *Multi-valued and universal binary neurons* (pp. 25-80). Springer, Boston, MA.

Aghdam, H. H., Heravi, E. J., & Puig, D. (2016). A practical approach for detection and classification of traffic signs using convolutional neural networks. *Robotics and autonomous systems*, *84*, 97-112.

Amidi, A., & Amidi, S. (2018) Convolutional Neural Networks cheatsheet. (Çev. A. Kızrak & Y. Kömeçoğlu) https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-*networks (20 Kasım 2020).*

Cireşan, D., Meier, U., & Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. In *2012 IEEE conference on computer vision and pattern recognition* (pp. 3642-3649). IEEE.

Changzhen, X., Cong, W., Weixin, M., & Yanmei, S. (2016). A traffic sign detection algorithm based on deep convolutional neural network. In *2016 IEEE International Conference on Signal and Image Processing (ICSIP)* (pp. 676-679). IEEE.

CS231n. (2017). Convolutional Neural Networks for Visual Recognition. *https://cs231n.github.io/convolutional-networks/ (20 Kasım 2020).*

Deep Learning. (2020). Deep Learning. *https://en.wikipedia.org/w/index.php?title= Deep_learning&oldid=1036897621 (20 Eylül 2020).*

Dettmers, T. (2015). Deep Learning in a Nutshell: History and Training. *NVIDIA Developer Blog. https://developer.nvidia.com/blog/deep-learning-nutshell-history-training/ (10 Şubat 2021).*

Erbaş, Ü. (2005). Yapay Sinir Ağları ve Uygulama Alanları. *Yüksek Lisans Tezi, Marmara Üniversitesi, Sosyal Bilimler Enstitüsü, İstanbul*.

ETSC. (2017). Briefing: ıntelligent speed assistance (ISA). *https://etsc.eu/briefing-intelligent-speed-assistance-isa/*

EUR-Lex. (t.y). Procedure 2018/0145/COD. *https://eur-lex.europa.eu/procedure/EN/2018 145 (20 Kasım 2020)*

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

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 580-587).

Grefenstette, E., Blunsom, P., De Freitas, N., & Hermann, K. M. (2014). A deep architecture for semantic parsing. *arXiv preprint arXiv:1404.7296*.

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision* (pp. 2961-2969).

Hinton, G. E., Dayan, P., Frey, B. J., & Neal, R. M. (1995). The" wake-sleep" algorithm for unsupervised neural networks. *Science*, *268*(5214), 1158-1161.

Hinton, G., Osindero, S., Welling, M., & Teh, Y. W. (2006). Unsupervised discovery of nonlinear structure using contrastive backpropagation. *Cognitive science*, *30*(4), 725-731.

Ivakhnenko, A. G., & Lapa, V. G. (1966). Cybernetic predicting devices. *PURDUE UNIV LAFAYETTE IND SCHOOL OF ELECTRICAL ENGINEERING*.

Jetson Nano (2020. *https://www.nvidia.com/tr-tr/autonomous-machines/embedded-systems/jetson-nano/ (11.Ekim.2020)*

Jocher G., (2020). YOLOv5. *https://zenodo.org/record/4154370#.X JRu9gzZPY (29 Ekim 2020)*

Kalchbrenner, N., Grefenstette, E., & Blunsom, P. (2014). A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

Kılıkçıer, Ç., & Yılmaz, E. (2018). Trafik Işığı Tespiti Yapan Bir Sürücü Güvenlik Destek Sistemi. *Politeknik Dergisi*, *21*(2), 419-426.

LabelImg (2020). LabelImg. *https://github.com/tzutalin/labelImg (11.Eylül.2020)*

LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., & Jackel, L. (1989). Handwritten digit recognition with a back-propagation network. *Advances in neural information processing systems*, *2*.

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.

Nanodet. (2020). Nanodet. *https://github.com/RangiLyu/nanodet (12.Kasım.2020)*

Prateek (2017) Statistics is Freaking Hard: WTF is Activation function. *https://towardsdatascience.com/statistics-is-freaking-hard-wtf-is-activation-function-df8342cdf292 (16 Ağustos 2017)*

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779-788).

Saha, S., Islam, M. S., Khaled, M. A. B., & Tairin, S. (2019). An efficient traffic sign recognition approach using a novel deep neural network selection architecture. In *Emerging Technologies in Data Mining and Information Security* (pp. 849-862). Springer, Singapore.

Shustanov, A., & Yakimov, P. (2017). CNN design for real-time traffic sign recognition. *Procedia engineering*, *201*, 718-725.

Song, H. A., & Lee, S. Y. (2013). Hierarchical representation using NMF. In *International conference on neural information processing* (pp. 466-473). Springer, Berlin, Heidelberg.

Şeker, A., Diri, B., & Balık, H. H. (2017). Derin öğrenme yöntemleri ve uygulamaları hakkında bir inceleme. *Gazi Mühendislik Bilimleri Dergisi (GMBD)*, *3*(3), 47-64.

Tencent:ncnn. (2021). Tencent/ncnn: ncnn is a high-performance neural network inference framework optimized for the mobile platform. *https://github.com/Tencent/ncnn (9 Kasım 2020).*

Tesla 100 milyar dolarlık piyasa değeriyle dünyanın 2. en büyük otomotiv şirketi oldu. (2021s). BBC News Türkçe. https://www.bbc.com/turkce/haberler-dunya-51220950 (10 Şubat 2021).

Tian, Z., Shen, C., Chen, H., & He, T. (2019). Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 9627-9636).

TTVS (Türkiye Trafik Veri Seti). (2021). TTVS || ITU Racing Driverless. *https://github.com/ituracingdriverless/TTVS (10 Ağustos 2021).*

Ultralytics (2021). YOLOv5. *https://github.com/ultralytics/yolov5 (10 Ağustos 2021).*

ÖZGEÇMİŞ

Ad-Soyad : Savaş HASÇELİK

Öğrenim Durumu: •

- **Lisans :** 2015, Yüzüncü Yıl Üniversitesi, Eğitim Fak., Bilgisayar ve Öğretim Teknolojileri Öğretmenliği
- **Yüksek Lisans :** 2021, Trakya, Hesaplamalı Bilimler ABD,

Mesleki Deneyim:

- 2016-2017 Kırklareli Cumhuriyet Ortaokulu Bilişim Teknolojileri Öğretmenliği
- 2017 yılından itibaren Kırklareli İl Milli Eğitim Müdürlüğünde BT İl Koordinatörlüğü görevini yürütmektedir.