T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

SÜRÜCÜ ASİSTAN SİSTEMLERİ İÇİN MOBİL GPU TABANLI GERÇEK ZAMANLI DURUM ANALİZİ VE TESPİT UYGULAMALARI

YÜKSEK LİSANS TEZİ

Emin GÜNEY

Enstitü Anabilim Dalı	**:**	**BİLGİSAYAR VE BİLİŞİM MÜHENDİSLİĞİ**
Tez Danışmanı	**:**	**Doç. Dr. Cüneyt BAYILMIŞ**

Ağustos 2021

T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

SÜRÜCÜ ASİSTAN SİSTEMLERİ İÇİN MOBİL GPU TABANLI GERÇEK ZAMANLI DURUM ANALİZİ VE TESPİT UYGULAMALARI

YÜKSEK LİSANS TEZİ

Emin GÜNEY

Enstitü Anabilim Dalı : BİLGİSAYAR VE BİLİŞİM MÜHENDİSLİĞİ

Bu tez 18/08/2021 tarihinde aşağıdaki jüri tarafından oybirliği ile kabul edilmiştir.

Jüri Başkanı **Üye** **Üye**

BEYAN

Tez içindeki tüm verilerin akademik kurallar çerçevesinde tarafımdan elde edildiğini, görsel ve yazılı tüm bilgi ve sonuçların akademik ve etik kurallara uygun şekilde sunulduğunu, kullanılan verilerde herhangi bir tahrifat yapılmadığını, başkalarının eserlerinden yararlanılması durumunda bilimsel normlara uygun olarak atıfta bulunulduğunu, tezde yer alan verilerin bu üniversite veya başka bir üniversitede herhangi bir tez çalışmasında kullanılmadığını beyan ederim.

Emin GÜNEY

18.08.2021

TEŞEKKÜR

Yüksek lisans tez çalışması süresince her türlü destek ve yardımlarını esirgemeyen araştırmanın planlanmasından yazılmasına kadar tüm aşamalarında deneyim ve bilgilerinden faydalandığım değerli danışman hocam Doç. Dr. Cüneyt BAYILMIŞ'a ve derin öğrenme teknolojileri konusunda bilgilerinden faydalandığım kıymetli hocam Dr. Öğr. Üyesi Süleyman UZUN'a ve çalışmam boyunca verisetlerinin hazırlanması ve sistemin geliştirilmesinde bana yardımcı olan gayretli ekip arkadaşlarım Neslihan ÇAKIRBAŞ, Havva Selin ÇAKMAK, Ali Göktuğ YALÇIN ve Dilara KOCA'ya teşekkür ederim.

Hayatımın her safhasında maddi ve manevi desteklerini benden esirgemeyen babam Metin GÜNEY ve annem Ayşe GÜNEY'e teşekkürü bir borç bilirim.

Bu çalışma, Sakarya Üniversitesi Bilimsel Araştırma Projeleri Koordinatörlüğü (BAPK) tarafından (Proje No: 2021-7-24-20) proje desteği ile gerçekleşmiştir.

İÇİNDEKİLER

TEŞEKKÜR i
İÇİNDEKİLER ii
SİMGELER VE KISALTMALAR LİSTESİ v
ŞEKİLLER LİSTESİ i
TABLOLAR LİSTESİ ix
ÖZET x
SUMMARY xi

BÖLÜM 1.
GİRİŞ 1
1.1. Literatür Araştırması 2
1.2. Motivasyon 8
1.3. Tezin Amacı ve Katkıları 8
1.4. Tez Organizasyonu 9

BÖLÜM 2.
SÜRÜCÜ ASİSTAN SİSTEMLERİNDE NESNE TESPİTİ İÇİN KULLANILAN TEKNOLOJİLER 10
2.1. Yapay Sinir Ağları (YSA) ve Çok Katmanlı Ağlar 10
2.2. Derin Öğrenme ve Derin Sinir Ağları (DNN) 11
2.3. Evrişimsel Sinir Ağları (CNN) ve Ana Bileşenleri 12
2.3.1. Evrişim katmanı 13
2.3.2. Havuzlama katmanı 14
2.3.3. Tam bağlı katman 14
2.4. Evrişimsel Sinir Ağlarının Gelişimi ve Temel Mimariler 15
2.4.1. LeNet mimarisi 15

2.4.2. AlexNet mimarisi 16
2.4.3. VGGNet mimarisi 19
2.4.4. GoogLeNet mimarisi 21
2.5. Nesne Algılama Modelleri 21
2.5.1. İki aşamalı algılayıcılar 22
2.5.1.1. R-CNN 22
2.5.1.2. Fast R-CNN 22
2.5.1.3. Faster R-CNN 23
2.5.2. Tek aşamalı algılayıcılar 23
2.5.2.1. SSD 24
2.5.2.2. YOLO 24
2.6. Yolo İçin Kayıp Fonskiyonu 27
2.7. YOLO v5 Nesne Tespit Mimarisi 28
2.7.1. Omurga kısmı 29
2.7.2. Boyun kısmı 30
2.7.3. Baş kısmı 30

BÖLÜM 3.
SÜRÜCÜ ASİSTAN SİSTEMİ UYGULAMALARININ GELİŞTİRİLMESİ... 31
3.1. Yazılım Geliştirme Ortamının Hazırlanması 32
3.2. Verisetlerinin Hazırlanması 34
3.2.1. Verileri etiketleme süreci 36
3.2.2. Verilerin eğitim ve test için ayırılması 38
3.3. Yazılım Ortamının Konfigürasyonu 39
3.4. Verilerin Eğitilerek Modellerin Oluşturulması 39
3.5. Modellerin Eğitim ve Test Sonuçlarının Analiz Edilmesi 43
3.5.1. Performans Değerlendirme Ölçütleri 43

BÖLÜM 4.
GELİŞTİRİLEN SÜRÜCÜ ASİSTAN SİSTEMİNİN GÖMÜLÜ SİSTEM ÜZERİNDE TEST EDİLMESİ 53
4.1. Uygulama İçin Mobil Platformların Hazırlanması 53

4.1.1. Nvidia Jetson xavier nx 54
4.1.2. Nvidia jetson nano 55
4.2. Sistemin Araç İçinde Test Edilmesi 58

BÖLÜM 5.
SONUÇ VE ÖNERİLER 59

KAYNAKLAR 61
ÖZGEÇMİŞ 68

SİMGELER VE KISALTMALAR LİSTESİ

ABS	: Adaptive Braking Systems
ACC	: Adaptive Cruise Control
ADAS	: Advanced Driver Assistance Systems
BTSC	: Belgian Traffic Sign Dataset
CNN	: Evrişimsel Sinir Ağı
conv	: Konvolüsyon işlemi
CSPNet	: Çapraz Aşamalı Kısmi Sinir Ağı
DenseNet	: Yoğun Evrişimsel Sinir Ağı
DNN	: Derin Sinir Ağı
Fast R-CNN	: Hızlı Bölge Tabanlı Evrişimsel Sinir Ağı
Faster R-CNN	: Daha-hızlı Bölge Tabanlı Evrişimsel Sinir Ağı
FC	: Full-Connected (Tam Bağlı Katman)
FPNet	: Özellik Piramit Ağı
GPU	: Grafik İşlem Birimi
GTSRB	: German Traffic Sign Recognition Benchmark
ImageNet	: Büyük Ölçekli Görsel Tanıma Yarışması
IoU	: Intersection Over Union
LDW	: Lane Departure Warning
mAP	: Mean Average Precision
PANet	: Yol Toplama Sinir Ağı
SPPNet	: Mekansal Piramit Havuzlama Sinir Ağı
SSD	: Single-Shot Multibox Detector
STSD	: Swedish Traffic Sign Dataset
TSR	: Traffic Sign Recognition
XYWH	: X-Y koordinatı Genişlik Yükseklik Formatı
YOLO	: You Only Look Once Algoritması
YSA	: Yapay Sinir Ağı

ŞEKİLLER LİSTESİ

Şekil 2.1. Yapay bir nöronun temel gösterimi 10
Şekil 2.2. Çok katmanlı bir YSA yapısı 11
Şekil 2.3. İkiden fazla gizli katmana sahip DNN yapısı 12
Şekil 2.4. Nesne algılama için örnek bir CNN Mimarisi [46] 12
Şekil 2.5. Görüntüde kernel hücresi kullanılarak yapılan evrişim işlemi 13
Şekil 2.6. Havuzlama işlemlerinden max-havuzlama işleminin yapılması [49].. 14
Şekil 2.7. LeNet mimarisi genel yapısı 16
Şekil 2.8. Alexnet mimarisi sınıflandırma adımları [57] 17
Şekil 2.9. AlexNet genel yapısı 18
Şekil 2.10. VGGNet mimarisi genel yapısı 19
Şekil 2.11. GoogLeNet genel yapısı 21
Şekil 2.12. Nesne Algılama yaklaşımlarının yöntem açısından sınıflandırılması 21
Şekil 2.13. R-CNN mimarisi sınıflandırma aşamaları 22
Şekil 2.14. Faster R-CNN modelinin çalışma prensibi [66] 23
Şekil 2.15. SSD mimarisi genel yapısı [67] 24
Şekil 2.16. YOLO'nun nesne tespitine yönelik çalışma prensibi 25
Şekil 2.17. YOLO'nun güven skoru ve sınıf olasılıklarını tahmin adımları 25
Şekil 2.18. Nesne tespitinden sorumlu YOLO katmanlı mimarisi [68] 27
Şekil 2.19. YOLO v5 mimarisinin alt mimarileri [71] 28
Şekil 2.20. YOLO v5 mimarisinin genel yapısı 29
Şekil 2.21. Büyüme hızı k=4 olan 5 katmanlı bir DenseNet bloğu [77] 30
Şekil 3.1. ADAS uygulamaları için model geliştirme aşamaları 31
Şekil 3.2. Colab'ın bulut servisinde Tesla P100- PCIE GPU ile eğitim 32
Şekil 3.3. YOLO v5 reposunun Github'dan klonlanması ve Colab'a yükleme ... 32
Şekil 3.4. Dosyaların çekilerek Drive'dan Colab'a aktarılması 33
Şekil 3.5. YOLO v5s mimarisi katmanları kod blokları 33

Şekil 3.6. GTSRB verisetinden örnek resimler [12] 34
Şekil 3.7. Araç dışı tespit uygulaması için oluşturulan lokal veriseti 34
Şekil 3.8. Trafik işareti ve nesne tespiti sınıflarının verisetindeki miktarları 35
Şekil 3.9. Sürücü davranış tespiti sınıflarının verisetindeki miktarları 35
Şekil 3.10. Araç içi sürücü davranışlarını tespit için kullanılan verisetinden örnek resimler 36
Şekil 3.11. Araç dışı ADAS uygulaması için insan, araç ve trafik işaretlerinin etiketlenmesi 36
Şekil 3.12. Araç dışı ADAS uygulaması için etiketleme işlemi 37
Şekil 3.13. Araç içi ADAS uygulaması için sürücü durumlarının etiketlenmesi. 37
Şekil 3.14. Sınırlayıcı kutunun resim üzerinde gösterilmesi..... 38
Şekil 3.15. YOLO formatına göre hazırlanmış dosya içeriğindeki alanlar..... 39
Şekil 3.16. Araç dışı nesne tespiti için hazırlanan custom-yaml dosya içeriği 40
Şekil 3.17. Araç içi nesne tespiti için hazırlanan custom-yaml dosya içeriği..... 40
Şekil 3.18. Önceden ağırlıklarndırılmış YOLO v5s ile modelin eğitilmesi..... 41
Şekil 3.19. Epoch sayısı = 850 için devam eden eğitim işlemi..... 41
Şekil 3.20. last.pt adlı model dosyasının yaklaşık 7 saat sonunda elde edilmesi. 42
Şekil 3.21. Öngörülen sınırlayıcı kutu gerçek sınırlayıcı kutu arasındaki ilişki .. 44
Şekil 3.22. Çeşitli sınırlayıcı kutular için IoU'nun hesaplanması..... 44
Şekil 3.23. Modelin eğitimi için (a) IoU değeri 0,5 için oluşan ortalama kesinlik grafiği (b) IoU değeri 0,5 ila 0,95 arasındaki adımlarda oluşan ortalama kesinlik grafiği (c) Total kesinlik grafiği 45
Şekil 3.24. Modelin eğitim sonucu performansı (a) Sınıflandırma doğruluğunun kayıp grafiği (b) Sınırlandırıcı kutunun doğruluğu için oluşan kutu kaybı (c) Oluşan nesnellik kayıp grafiği 45
Şekil 3.25. Modelin test sonucuna ilişkin tahmin doğruluğu grafiği 46
Şekil 3.26. Araç dışı nesne algılama için eğitim grafikleri 47
Şekil 3.27. Araç dışı nesne algılama görevinde karşılaştırmalı karışıklık matrisi 47
Şekil 3.28. Modelin başarımının test sonuçları -1..... 48
Şekil 3.29. Modelin başarımının test sonuçları -2..... 48
Şekil 3.30. Modelin başarımının test sonuçları -3..... 49

Şekil 3.31. Modelin test sonucu performansı (a) Sınıflandırma doğruluğunun kayıp grafiği (b) Sınırlandırıcı kutunun doğruluğu için oluşan kutu kaybı (c) Oluşan nesnellik kayıp grafiği..... 49
Şekil 3.32. Sınıflandırma doğruluğunun kayıp grafiği (b) Sınırlandırıcı kutunun doğruluğu için oluşan kutu kaybı (c)Oluşan nesnellik kayıp grafiği .. 50
Şekil 3.33. Modelin eğitim performansına ilişkin oluşan (a) IoU değeri 0,5 için oluşan ortalama kesinlik (b)IoU değeri 0,5 ila 0,95 arasındaki adımlarda oluşan ortalama kesinlik (c) Kesinlik grafikleri..... 51
Şekil 3.34. Modelin test sonucu performansı (a) Sınıflandırma doğruluğunun kayıp grafiği (b) Sınırlandırıcı kutunun doğruluğu için oluşan kutu kaybı (c) Oluşan nesnellik kayıp grafiği..... 51
Şekil 3.35. Araç içi modelin çeşitli test sonuçlarından görüntüler 52
Şekil 4.1. Xavier Nx geliştirme kiti ve bağlantıları 53
Şekil 4.2. Xavier Nx özellikleri..... 54
Şekil 4.3. Nvidia Jetson Nano özellikleri..... 55
Şekil 4.4. ADAS uygulamaları gerçek zamanlı tespit hızları FPS kıyaslaması... 57
Şekil 4.5. ADAS uygulamaları gerçek zamanlı tespit hızları karşılaştırması..... 57
Şekil 4.6. Araç içerisinde test ekipmanlarının görüntüleri..... 58

TABLOLAR LİSTESİ

Tablo 1.1. ADAS'ın araç dışı uygulamaları için incelenen literatür çalışmaları ... 4
Tablo 1.2. ADAS'ın araç içi uygulamaları için incelenen literatür çalışmaları 7
Tablo 2.1. LeNet mimarisi katmanları ve katman özellikleri 16
Tablo 2.2. AlexNet mimarisi katmanları ve katman özellikleri 17
Tablo 2.3. VGG-16 mimarisinin katman ve görevleri 20
Tablo 2.4. VGG-16 mimarisinin hassasiyet dereceleri [58] 20
Tablo 4.1. Yapılan çalışmadaki test ortamlarının karşılaştıırlması 56

ÖZET

Anahtar kelimeler: Gelişmiş Sürücü Asistan Sistemleri (ADAS), Derin Öğrenme (DNN), Evrişimsel Sinir Ağları (CNN), Trafik İşareti Tespiti (TSR), Nesne Tespiti, Sürücü Davranış Tespiti.

Bilgisayarlı görme teknolojilerinin gelişmesi ve derin öğrenme çalışmalarının hız kazanmasıyla sürücü asistan sistemleri son zamanlarda oldukça yaygınlaşmıştır. Bu sistemler güvenlik ve sürüş kolaylığı sağlamak amacıyla sürücü ve araç çevresinden gerekli verileri toplayarak kritik durumları tespit etmeyi hedefler. Sürücü ve çevrenin anlık olarak izlenmesi uyarı tespit sistemi için önemlidir. Bunun yapılabilmesi ancak sistemin gerçek zamanlı çalışmasıyla mümkün olacaktır.

Bu tezde, gelişmiş sürücü asistan sistemleri (ADAS) için görüntüye dayalı olarak gerçek zamanlı çalışan araç içi ve araç dışı durumları gömülü platforma bağlı kameralar ile tespit eden bir çalışma gerçekleştirilmiştir. Hem araç içinde hem de araç dışında sürücü ve çevreye odaklı çalışan iki adet uygulama gerçekleştirilmiştir. Geliştirilen sistem araç dışında trafik işareti, yaya ve nesneleri tespit ederken, araç içinde ise sürücü durumlarını analiz ederek, telefon ve sigara kullanımını ve göz takibi yaparak yorgunluk ve uyku tespiti ile sürücüye uyarı sağlamaktadır.

Çalışmada, hazır veri seti ile çalışmaya özgü elde edilmiş veri setleri kullanılarak grafik kartı (GPU) üzerinde eğitimler ile modeller oluşturulmuştur. Tespit hızlarını karşılaştırmak amacıyla sistem düşük güç ve yüksek performansa sahip iki gömülü platform (Jetson Xavier Nx, Nvidia Jetson Nano) ve bilgisayar ortamında test edilerek sonuçlar analiz edilmiştir. Uygulamaların sonucunda gerçek zamanlı çalışan bir ADAS prototipi gerçekleştirilmiştir.

MOBILE GPU BASED REAL-TIME STATUS ANALYSIS AND DETECTION APPLICATIONS FOR DRIVER ASSISTANT SYSTEMS

SUMMARY

Keywords: Advanced Driver Assistance Systems, Deep Neural Network (DNN), Traffic Sign Detection (TSR), Object Detection, Driver Behavior Detection.

With the development of computer vision technologies and the acceleration of deep learning studies, driver assistance systems have become quite widespread recently. These systems aim to detect critical situations by collecting the necessary data from the driver and vehicle environment in order to provide safety and driving convenience. Instant monitoring of the driver and the environment is important for the warning detection system. This will only be possible with the real-time operation of the system.

In this thesis, a study has been carried out for advanced driver assistance systems (ADAS) that detects real-time in-vehicle and out-of-vehicle situations based on images with cameras connected to the embedded platform. Two applications focused on the driver and the environment, both inside and outside the vehicle, were carried out. While the developed system detects traffic signs, pedestrians and objects outside the vehicle, it provides warnings to the driver with fatigue and sleep detection by analyzing the driver's status inside the vehicle, monitoring phone and cigarette use and eye tracking.

In the study, models were created with trainings on the graphics card (GPU) using the ready data set and the data sets specific to the study. In order to compare the detection rates, the system was tested in two low power and high performance embedded platforms (Jetson Xavier Nx, Nvidia Jetson Nano) and computer environment and the results were analyzed. As a result of the applications, a real-time ADAS prototype has been realized.

BÖLÜM 1. GİRİŞ

Her geçen gün artan nüfus sayısı ile beraber trafiğe dahil olan araç sayısında da büyük bir artış olmaktadır. Kaza risklerini azaltmak ve sürüş güvenliğini arttırmak amacıyla geçmişten bu yana pek çok çalışma ve yatırımlar yapılmıştır. Bu bağlamda geliştirilen teknolojiler genel olarak sürücü destek sistemlerinin (Advanced Driver Assistance Systems, ADAS) altyapısını oluşturmaktadır.

ADAS sürücü ve aracın tehlikeli trafik durumlarını tanımasına ve bunlara doğru ve hızlı tepki vermesine yardımcı olan sistemlerdir. Otomotiv endüstrisi tarafında güvenlik ve konforu arttırmak amacıyla günümüzde gelişen teknolojiyle birlikte yaygınlaşarak önemli bir çalışma konusu haline gelmiştir. Kilitlenme önleyici fren sistemleri (Anti-lock Braking System, ABS), adaptif hız sabitleyicileri (Adaptive Cruise Control System, ACC) ve şeritten ayrılmayı uyarma (Lane Departure Warning, LDW) gibi sistemler sadece araç merkezli kontrolü sağlamayı odaklanırken, yol çevresindeki yaya ve trafik işaretlerini tanıma (Traffic Sign Recognition, TSR), şeritleri tespit etme ve sürücü davranışlarını analiz ederek yorgunluk ve uyku gibi durumlarda sürücüyü uyarma gibi görüntüye dayalı olarak yapılan işlemler ise sürücü merkezli çalışan sistemler arasında yer almaktadır.

Maliyet ve performans bakımından otomotiv üreticileri çeşitli uygulamalar gerçekleştirmişlerdir. ADAS geliştiricileri araç içi ve sürüş çevresindeki ortam durumunu belirlemede kamera [1-3], radar [4,5], lidar [6-8] gibi çeşitli sensörler ile uygulamalar yapmaktadırlar. Bunlara ek olarak istekler doğrultusunda GPS sistemleri, araç ağ yapısı ve araçtan araca harici bilgi kaynakları da kullanılmaktadır. Bunlar içerisinde kamera tabanlı çalışan sistemler özellikle maliyet açısından büyük avantaj sağlamakla beraber hızla gelişen bilgisayarlı görme teknolojilerini kullanarak görüntü işleme ile analiz sağlamaktadır.

Araç dışında, dış çevreyi algılama sürüş ortamı hakkında bilgi toplamayı içerir. En önemli dış çevre bilgileri; şerit ve şerit sınırlarını, yakındaki araçları, yayaları, trafik işaretlerini, trafik ışıklarını ve bazı nesneleri içerir. Ortam bilgilerinin algılanma kalitesi hava, yol ve ışık koşullarından etkilendiğinden sistemin geliştirilmesinde bu faktörlerin dikkate alınması gerekmektedir.

Araç içi sürücü tespit sistemlerinde ise sürücünün kamera ve çeşitli sensörler vasıtasıyla davranışlarının izlenmesi ve buna bağlı olarak sürücü farkındalığının arttırılması hedeflenmektedir. Sürücü davranışlarının izlenmesi sürücüden alınan verilerin işlenerek analiz edilmesini gerektirmektedir. Sürücünün uyarılmasını gerektiren durumlardan bazıları dikkat dağınıklığına sebep olan telefonla konuşma veya mesajlaşma, uyku durumuna girme, yolcularla konuşma, sigara içme ve benzeri durumlardır. Bu sorunları çözmek veya en aza indirgemek için trafikte ADAS sistemlerinin geliştirilmesi hayati derecede büyük ölçüde önem taşımaktadır.

1.1. Literatür Araştırması

Son yıllarda sürüş deneyimini arttırmaya yönelik çeşitli çalışmalar ADAS sistemlerinin hızlı gelişmesine katkıda bulunmuştur. ADAS sistemleri sürüş konforu sağlayarak araç teknolojilerini otomatikleştirmenin yanında, istenmeyen durumlarda sürücüyü uyararak güvenliği sağlamaktadır. Kameralar aracılığıyla gerek sürüş çevresi gerekse sürücü izlenerek elde edilen veriler güvenliğin sağlanarak kazaların azaltılması için önemlidir. Literatür incelendiğinde ADAS sistemleri için araç içi ve sürüş ortamı temel alınarak yapılan birçok çalışma ile karşılaşılmaktadır.

Sürüş çevresindeki trafik işareti tespit ve tanıma (Traffic Sign Detection and Recognition, TSD&R) görevi için son on yılda büyük ölçüde araştırmalar yürütülmüştür. Yoldaki işaretlerin tanınması tespit ve tanımlama olarak iki adımda gerçekleşmektedir. Hedeflenen nesnenin tespit edilmesi algılanan nesnenin konumuna odaklanmayı hedeflerken tanımlama işleminde ise tespit edilen hedeflerin ait olduğu sınıfı belirlemek için tahmini bir sınıflandırma işlemi uygulanmaktadır.

Genel olarak TSD&R'de yapılan çalışmalar geleneksel yaklaşımlarla birlikte, (i) görüş özelliği tabanlı ve (ii) evrişim tabanlı yaklaşımlar şeklinde sınıflandırılabilir [9].

Görüş özelliğine dayalı yapılan çalışmalar işaretlerin şekil, kenar, renk ve aydınlatma özelliklerinden faydalanarak tespit sağlamaktadır. 2015'te Li ve arkadaşlarının önerdikleri renk tabanlı sistem, görüşü engelleyen hava durumlarında bile çalışabilemesi ile yenilikçidir [10]. Aynı yıl Yin vd. tarafından sunulan TSD&R çalışmalarına yönelik üç aşamadan oluşan yaklaşımları da tanıma doğruluğu ve işlem hızını arttırmıştır [11]. Güncel çalışmalardan 2019'da Xu ve arkadaşları karmaşık trafik ortamında TSD&R için renk eşiği bölütleme ve şekil simetrisinin hipotez testini kullanan yenilikçi bir tespit yöntemi önermiştir. Veri analizi için şekil ve renk tabanlı çalışan bir algoritma oluşturarak GTSRB [12] veriseti üzerinde test sağlanmış ve yaklaşık %94 oranında bir doğruluk elde edilmiştir [13].

2015'te Qian ve ekibi TSD&R için Derin Evrişimsel Sinir Ağları tabanlı çalışarak algılama açısından yüksek performans ve tanıma doğruluğu elde eden bir sistem önermişlerdir [14]. 2016'da Changzhen ve arkadaşları Bölge Öneri Ağı (RPN) kullanarak Derin Evrişimli Sinir Ağı (DCNN) temelli bir yöntem sunmuşlardır [15]. 2017'de Zhang ve arkadaşları Çin'deki trafik işaretlerini hızlı ve doğru tespit etmede yüksek hıza erişerek tespit hızını 0,017 saniyeye kadar düşürebilmişlerdir. YOLO v2 tabanlı çalışan sistemleri gerçek zamanlı tespit işlemlerinde oldukça başarılı sonuçlara ulaşmıştır [16]. Ammour ve arkadaşları lineer SVM sınıflandırıcısı ve ortalama kaydırma algoritmasını kullanarak insansız hava araçları tespiti için Evrişimsel Sinir Ağlarını (Convolutional Neural Network, CNN) kullanmışlardır [17]. 2018'de ise Ćorović vd. farklı hava koşullarında gerçek zamanlı çalışan YOLO v3 tabanlı bir sistem sunmuşlardır [18]. 2021 yılında Jamtsho ve arkadaşlarının plaka, insan ve kask tespiti için %98 civarında bir tespit doğruluğu sağlayarak başarım göstermiştir [19].

Bu tez kapsamında ise ADAS'ın araç dışı uygulaması için araç, yaya ve trafik işaretlerini gerçek zamanlı tespit eden YOLO v5 tabanlı bir sistem gerçekleştirilmiştir. Tablo 1.1.'de bahsedilen literatür çalışmaları ve tezde yapılan uygulamalar verilmiştir.

Tablo 1.1. ADAS'ın araç dışı uygulamaları için incelenen literatür çalışmaları

No	Yazar	Uygulama Ortamı	Yapılan Uygulama	Kategori	Kullanılan Teknikler	Kullanılan Veriseti
[10]	Li vd. 2015	Video dizisi	Trafik İşareti Tespiti	Renk segmentasyonu, Şekil simetrisi tabanlı	Yönlendirilmiş gradyanların piramit histogramı	Özgün veriseti
[11]	Yin vd. 2015	Video dizisi	Trafik İşareti Tanıma	Özellik tabanlı sabit ikili desen rotasyonu	Hough ve SIFT dönüşümleri, Yapay sinir ağları (YSA)	GTSRB [12] ve STS [20]
[13]	Xu vd. 2019	Video dizisi	Trafik İşareti Tespiti	Adaptif renk eşiği ve Şekil simetrisi tabanlı	Histogram dağılım fonksiyonu, Şekil simetrisi algılama algoritması	GTSRB [12]
[14]	Qian vd. 2015	Video dizisi	Trafik İşareti Tespiti	Derin Evrişimsel Sinir Ağları tabanlı	Bölge önerisi, Kenar algılama ve bağlı bileşen analizi	GTSRB [12], MNIST[21], CASIA[22]
[15]	Changzhen vd. 2016	Video dizisi	Trafik İşareti Tespit	Faster R-CNN tabanlı	Bölge teklif ağı	Özgün veriseti
[16]	Zhang vd. 2017	Gerçek zamanlı	Trafik İşareti Tespit ve Tanıma	Geliştirilmiş YOLO v2 tabanlı	Izgara bölümleme tekniği	GTSRB [12] ve CCTSDB [23]
[17]	Ammour vd. 2017	Fotoğraf	Araba Tespiti	Evrişimsel Sinir Ağları tabanlı	Lineer SVM sınıflandırıcısı, Ortalama kaydırma algoritması	Özgün veriseti
[18]	Ćorović vd. 2018	Gerçek zamanlı	Araç, Yaya ve Trafik İşareti Tespiti	Darknet ve YOLO v3 tabanlı	Izgara bölümleme tekniği	Berkeley BDD100K [24]
[19]	Jamtsho vd. 2021	Gerçek zamanlı	Plaka, İnsan ve Kask Tespiti	YOLO v2 tabanlı, Darknet19 yapısı	Izgara bölümleme tekniği	Özgün veriseti
Tez Uygulaması (1)		Gerçek zamanlı	Araç, Yaya ve Trafik işaretleri tespiti	Derin Evrişimsel Sinir Ağları tabanlı	YOLO v5 mimarisi	Özgün veriseti ve GTSRB[12]

Nesnelerin daha iyi şekilde algılanması için ağın verisetleri ile eğitiminin gerçekleşmesi gerekmektedir. Literatürdeki çalışmalar incelendiğinde trafik işaretlerinin algılanmasına yönelik pek çok algoritma uyarlanmış ve farklı verisetleri oluşturulmuştur. 2009 yılından önce, test edebilmek adına herhangi bir açık kaynak ve ek açıklamaların olduğu veri seti bulunmamaktaydı [25]. Ancak bu yıldan sonra araştırmacılar tarafından Belçika Trafik İşareti Sınıflandırma [26] (BTSC), LISA Trafik İşareti Veriseti [27], Alman Trafik İşareti Tanıma [12] (GTSRB) ve benzeri veritabanları oluşturulmuştur.

Bazı araştırmacılar, çeşitli tehlikeli koşullar altında iyi performans gösterebilen özellik tabanlı kararlı çalışan trafik işareti algılama yöntemi geliştirmeye odaklandılar. Haloi vd. çalışmalarında GTSRB [12] verisetini kullanarak derin öğrenme ağ yapısını optimum parametre ve bellek kullanarak eğitip altı işaret sınıfında yüksek tespit başarımları elde etmişlerdir [3]. Timofte vd. BTSC [26] veriseti ile renklerin ayırımı için eşikleme yaparken şekillerin tespiti için Hough dönüşümü kullanmışlardır [4].

Araç içi kısım için tespit işlevinin gerçekleştirilmesine gelindiğinde ise trafik kazalarının önemli bir kısmı sürücünün dikkat dağınıklığından kaynaklanmaktadır**.** Sürüş sırasında cep telefonu kullanımı, dikkat dağıtan unsurlardan biridir ve farkında olmadan can kaybı ve güvenlik için bir tehdit oluşturmaktadır. Bir istatistiğe göre trafik kazalarının yaklaşık %25-%50'si sürüş esnasında telefonla arama ve mesajlaşma gibi dikkati dağıtan eylemlerden kaynaklanmaktadır [28].

Mourant ve Rockwell, çarpışma deneyimi yaşayan yaklaşık 700 sürücüyü araştırdı. Bir çarpışma kazasından on dakika önce arama yapan sürücünün, normal sürüşe göre 4,3 kat daha fazla çarpışma olasılığına sahip olduğunu bulmuşlardır [29]. Bu nedenle, araç kullanırken telefon kullanımını gerçek zamanlı tespit etmek ve bu türdeki davranışlara karşı uyarılar vermek güvenlik açısından büyük önem taşır. Bu konuda literatür incelendiğinde araştırmacılar tespit için cep telefonu sinyal algılama ve bilgisayarlı görme metotlarını kullanarak çözümler önermişlerdir [30,31].

Derin öğrenme teknolojilerinin ilerlemesiyle beraber hedef tespiti başarımı yüksek ölçüde artmıştır. Xiong vd, çalışmalarında derin öğrenmeye dayalı arama davranışı algılama algoritmasını önermişlerdir. Arama işleminin aday bölgesini belirlemek için yüz algılama kullanırlarken, izlemeyi belirlemek için ise Aşamalı Kalibrasyon Ağları [32] (Progressive Calibration Networks, PCN) kullanmışlardır. Daha sonra aday bölgedeki cep telefonunu algılamak için CNN tabanlı arama algılama algoritmasını kullanarak %96 oranında tespit sağlamışlardır [33].

Sürücünün cep telefonu kullanımını tespit etmek için yapılan çalışmalardan, He ve arkadaşları CornerNet-Lite ağına dayalı bir yöntem sunmuşlardır. Önceki çalışmalarda tespit doğruluğunun düşük olması problemini çözmek için veri setleri manuel olarak kurulmuş buna ek olarak çeşitli yöntemler kullanmışlardır. Sürücü arama davranışını gerçek zamanlı tespit etmede %96'ya yakın bir doğruluğa ulaşarak işlem hızını arttırmışlardır [34]. Zhang ve ekibi ise SmokingNet adlı bir sinir ağı modelini tanıtarak görüntü özellikleri ve sigara içme hareketlerini kullanarak sigara içilen görüntüleri algılayan ve yüksek doğrulukta çalışan bir uygulama yapmışlardır [35].

2020 yılında Cho sigara kullanımını tespit için makine öğrenmesi tabanlı doğrusal olmayan SVM ve kNN algoritmalarını kullanarak yaptığı çalışmada %93 oranında bir doğruluk elde etmiştir [38]. Aynı yıl Inthanon ve arkadaşları video dizileri üzerinden kişilerin uyku halini tespit edebilmek üzere görüntü işleme tekniklerini çalışmalarına uyarlamışlardır [40]. Craye vd. 2015 Sürücülerin Dikkat Dağınıklığını tespit ve tanıma görevi için Ada Boost sınıflandırıcısı ve Gizli Markov Modelini kullanarak Makine öğrenmesi tabanlı çalışan bir uygulama önererek ortalama %89 tespit doğruluğuna ulaştıklarını belirtmişlerdir [39]. 2021'de Savaş ve Becerikli sürücü yorgunluğunu tespit için çok görevli CNN tabanlı çalışan gerçek zamanlı sistemlerinde %98 oranında bir doğruluğa ulaşarak başarıyla bir uygulama gerçekleştirilmiştir [41].

Tablo 1.2.'de yapılan literatür incelemeleri ve tez kapsamında gerçekleştirilen ADAS sisteminin araç içi uygulamaları gösterilmiştir. YOLO v5 ile sürücünün dikkat dağınıklığı, sigara içmesi ve telefonu kullanımının tespit işlemleri gerçekleştirilmiştir.

Tablo 1.2. ADAS'ın araç içi uygulamaları için incelenen literatür çalışmaları

No	Yazar	Uygulama Ortamı	Yapılan Uygulama	Kategori	Kullanılan Teknikler	Kullanılan Veriseti
[33]	Xiong vd. 2019	Gerçek zamanlı	Sürücünün yüzü ve Cep telefonu kullanımının Tespiti	Evrişimsel Sinir Ağları ve Aşamalı Kalibrasyon Ağları (PCN) tabanlı	Özgün algılama algoritması	CBDS [33]
[34]	He vd. 2021	Gerçek zamanlı	Cep telefonu kullanımı Tespiti	CornetNet-Lite tabanlı	Kum saati modülü (Hourglass Module)	Özgün veriseti
[35]	Zhang vd. 2018	Gerçek zamanlı	Sigara kullanımı Tespiti	Evrişimsel Sinir Ağları (CNN) tabanlı	SmokingNet	ImageNet[36] ve 1MHand[37] verisetleri
[38]	Cho 2020	Gerçek zamanlı	Sigara kullanımı Tespiti	Makine öğrenmesi tabanlı	Doğrusal olmayan SVM, kNN sınıflandırıcısı, Çok katmanlı algılayıcılar	Özgün veriseti
[39]	Craye vd. 2015	Video dizisi	Sürücü Dikkat Dağınıklığı Tespit ve Tanıma	Makine öğrenmesi tabanlı	Ada Boost sınıflandırıcısı ve Gizli Markov Modeli	Özgün veriseti
[40]	Inthanon 2020	Video dizisi	Uykusuzluk Tespiti	Bilgisayarlı görme tabanlı (Computer Vision, CV)	Görüntü işleme teknikleri (Image Processing, IP)	Özgün veriseti
[41]	Savaş vd. 2021	Gerçek zamanlı	Sürücü Yorgunluk Tespiti	Çok görevli CNN modeli tabanlı	Dlib algoritması	YawdDD[42] ve NthuDDD[41] verisetleri
Tez Uygulaması (2)		Gerçek zamanlı	Sürücü dikkat dağınıklığı tespiti, Sigara tespiti ve Telefon kullanımı tespiti	Derin Evrişimsel Sinir Ağları tabanlı	YOLO v5 mimarisi	Özgün veriseti

1.2. Motivasyon

Araç kazalarının ana nedeni dikkati dağılmış, yorgun veya sarhoş sürücülerin trafiğe dahil olmasından ve sürüş çevresini fark edememesinden kaynaklanmaktadır. Trafikte seyir halindeki araç ve sürücülerin çeşitli faaliyetlerini izleyerek ikaz eden bir sistem kaza riskinin azaltılmasına yardımcı olmaktadır. Araçların üst segmentleri sürücülerin yol farkındalığını arttıran ve sürüş konforunu iyileştiren bu gibi ADAS sistemlerine gömülü şekilde sahipken, diğer araçlarda ise araca gömülü böyle bir sistem bulunmamaktadır. Bu da sürücülerin trafikteki güvenlik riskini arttırarak tehlike potansiyelini tetiklemektedir. Bu problemin çözümüne yönelik araçlara sonradan takılabilecek ADAS sistemlerinin geliştirilmesi bu tez çalışmasının motivasyonudur.

1.3. Tezin Amacı ve Katkıları

Bu tezin amacı sürücü güvenliği ve sürüş deneyimini arttırmaya yönelik derin öğrenme tabanlı çalışan bir sürücü asistan sistemi (ADAS) geliştirmektir. Çalışmada görüntüye dayalı olarak araç, çevre ve sürücü davranışları analiz ve tespit edilerek sürücüye uyarı sağlanmıştır. Bununla birlikte geliştirilen sistem düşük güç ve yüksek hesaplama gücüne sahip gömülü platform (prototip) üzerinde test edilerek gerçek zamanlı çalışan bir uygulama elde etmeye odaklanılmıştır. Bu bakımdan, sistemin düşük maliyet ve taşınabilirlik açısından verimli bir uygulama olması hedeflenmiştir.

Bu amaçlar doğrultusunda yapılan tez çalışmasının ana katkıları şu şekilde sayılabilir:

- Literatürde sadece araç içi veya araç dışı kısımlar dikkate alınarak ADAS sisteminin uygulamaları geliştirilmişken, bu tez çalışmasında ise her iki işlevi de yerine getirebilen tek bir sistemin tasarım ve uygulaması gerçekleştirilmiştir.

- Hem araç içinde hem de araç dışında gerçek zamanlı olarak durum tespitinin yapılabilmesi kritik durumların anlık olarak fark edilmesine katkı sağlamıştır.

- Geliştirilen ADAS sistemi uygulamaları güncel derin öğrenme tabanlı algoritma kullanılarak çalıştırılmıştır (YOLO v5).

- Çalışmaya özgü toplanmış gerçek görüntülerle veriseti oluşturularak literatüre katkı sağlanmıştır.

- Tespit işlemlerinin gömülü sistem üzerinde gerçekleştirilmesi mobiliteyi arttırarak maliyeti düşürmüş ve çalışmayı kolayca monte / entegre edilebilir bir ürün haline getirmiştir.

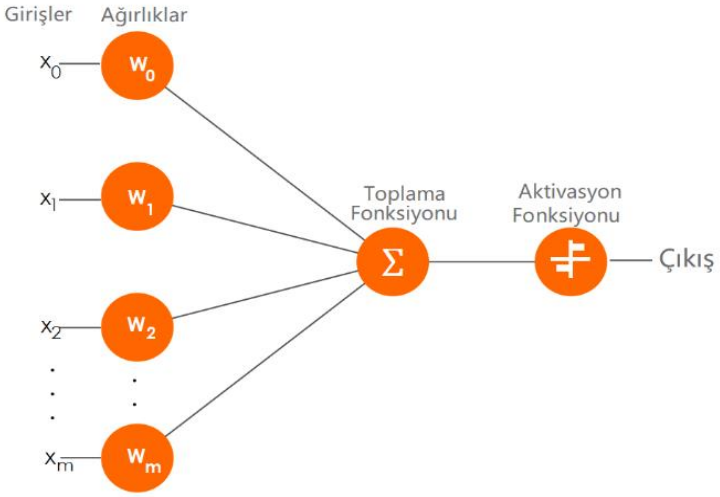
1.4. Tez Organizasyonu

Bu tez çalışması beş bölümden oluşmaktadır. Giriş bölümünde gelişmiş sürücü asistan sistemleri (ADAS) tanıtılmış ve yapısından bahsedilmiştir. Ardından ADAS sistemiyle ilgili literatürde geçmişten bu yana yapılmış olan çeşitli çalışmalara değinilerek araç içi ve araç dışı uygulamalarından bahsedilmiştir. İkinci bölümde gerçekleştirilen uygulamanın temelini oluşturan nesne tespit teknolojileri tanıtılmıştır. Derin öğrenme uygulamaları için kullanılan algoritmaların genel yapısı anlatılarak nesne tespiti açısından gelişimi gösterilmiştir. Üçüncü bölümde gerçekleştirilen uygulamalar için kullanılacak ortam hakkında bilgiler verilerek hazırlanma aşamaları şekillerle anlatılmıştır. İki uygulama için de sırasıyla modeller oluşturularak grafiklerle performans ölçümü yapılmıştır. Dördüncü bölümde modeller iki gömülü platform ve test bilgisayarı üzerinde çalıştırılarak algılama hızları elde edilip analiz sonuçları karşılaştırılmıştır. Son bölümde ise çalışma sonucunda elde edilen bulgular yorumlanmış ve gelecek çalışmalar için öneriler sunulmuştur.

BÖLÜM 2. SÜRÜCÜ ASİSTAN SİSTEMLERİNDE NESNE TESPİTİ İÇİN KULLANILAN TEKNOLOJİLER

2.1. Yapay Sinir Ağları (YSA) ve Çok Katmanlı Ağlar

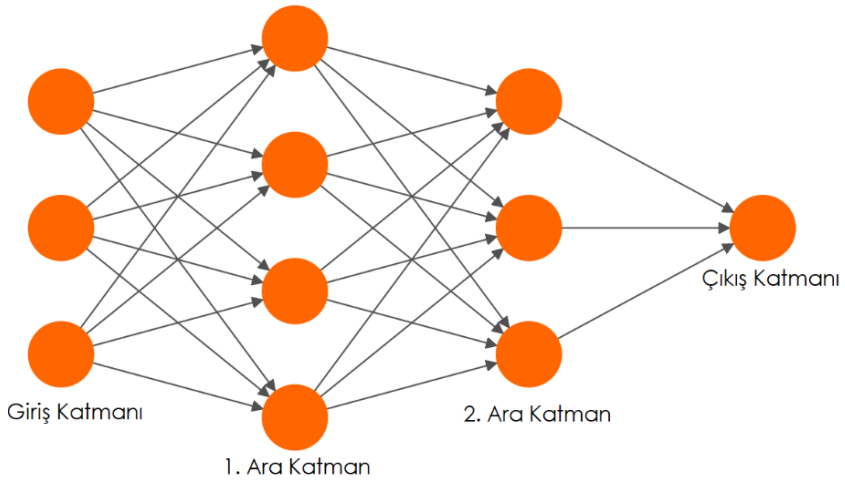
Yapay sinir ağları (YSA), insan beynini modelleyerek öğrenme kabiliyeti kazanmayı hedefleyen yapay zeka yöntemlerinden biridir. Yapısında bulunan temel birim olan nöronlar sayesinde bilgi kendi hafızasında tutulmakta ve ağırlıklarla katmanlar içerisindeki nöronlar birbirleriyle bağlanmaktadır. Şekil 2.1.'de YSA'da yapay bir nöronun modellenmesi gösterilmiştir. YSA, biyolojik sinir ağlarını taklit ederek bilgiyi paralel olarak işler ve dağıtık bir mimariye sahiptir [43].



Şekil 2.1. Yapay bir nöronun temel gösterimi

YSA'da her katman farklı sayıda nöron içerebilir. Şekil 2.2.'de birden fazla katmana sahip bir YSA modeli görülmektedir. Modelin katmanlarındaki nöronlar verilerden gelen girdiyi alarak, bu girdileri arttıran veya azaltan bir dizi ağırlıkla çarparak işlem yapmaktadır. Böylelikle algoritmanın verileri ağırlıklandırarak öğrenme sağlanır. Bir nöron öncelikle girdi katmanda giriş değerleri ile aktive edilmektedir. Girdi YSA için harici bir tetikleyici veya diğer yapay nöronların çıktılarından gelen değerlerdir. Bundan sonraki katmanda ağırlıklar yine kendilerine karşılık gelen girdilerle çarpılır

ve bu işlem çıkış katmanına kadar devam etmektedir. Nöron kısmında girdi ve ağırlıklar, herbir girdinin ağırlığı ile çarpımının toplamı olarak işleme alınır. Bunun sonucunda giriş nöronunun sinir ağı çıkışı üzerindeki etkisinin ölçülerini veren aktivasyon fonksiyonundan geçirilir [44]. Aktivasyon fonksiyonu çıktının belirli koşullara göre aktive edilip edilmeyeceğine karar vermektedir. Aktivasyon fonksiyonlarına sigmoid, tanh, softmax ve ReLU gibi sık kullanılan fonksiyonlar örnek verilebilir.



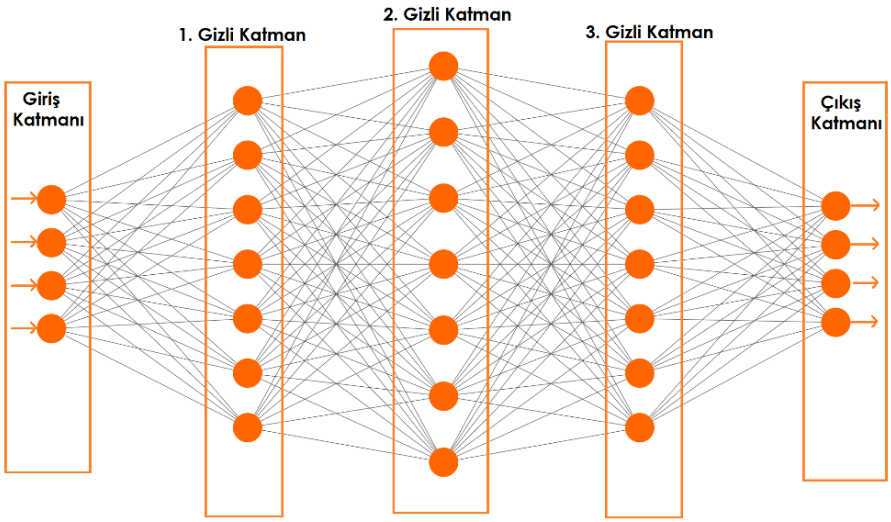
Şekil 2.2. Çok katmanlı bir YSA yapısı

En son adımda ise nöronun çıktısı oluşturulur. Bu çıktı başka bir nörona iletilebilir veya dış ortama verilip sonlandırılabilir. Elde edilen çıktı değeri kullanılan aktivasyon fonksiyonuna bağlı olarak değişim göstermektedir.

2.2. Derin Öğrenme ve Derin Sinir Ağları (DNN)

Derin öğrenme; nesne algılama, ses tanıma ve doğal dil işleme gibi pek çok alanda birbirini takip eden katmanlarla veriyi işleyerek daha kullanışlı gösterimler elde edebilen makine öğrenmesi çeşitlerinden biridir. Geleneksel makine öğrenmesi yöntemlerinden farklı olarak veriyi kodlanmış kurallar ile öğrenmekten ziyade çok sayıda katmana sahip YSA'yı kullanır. Bununla beraber derin öğrenme, öğrenme işlevini örnekler üzerinden gerçekleştirmektedir [43,44]. Derin Sinir Ağları (DNN), genellikle sınıflandırma, regresyon işlemlerinde kullanılan, ikiden fazla gizli katmana

sahip çok katmanlı sinir ağı yapısıdır. Şekil 2.3.'de üç gizli katmana sahip örnek bir DNN yapısı verilmiştir.

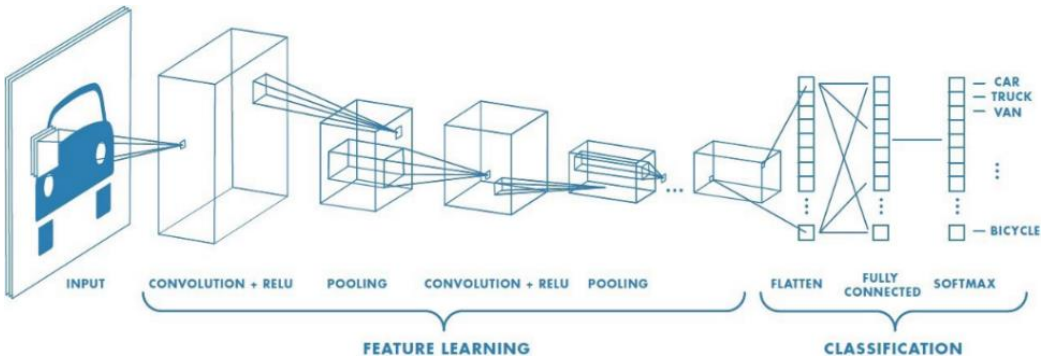


Şekil 2.3. İkiden fazla gizli katmana sahip DNN yapısı

DNN'lerin eğitiminde geri yayılım işlemi yapılmaktadır. Bu işlem ağın başarımını arttırmakla beraber hata değerini düşürmektedir. Karmaşık yapısıyla birlikte birçok alanda kullanımı başarıya ulaşmıştır.

2.3. Evrişimsel Sinir Ağları (CNN) ve Ana Bileşenleri

Mimarinin birçok alt modeli olmakla beraber genel olarak Evrişimsel Sinir Ağları veya CNN nöronlar arasındaki bağlantıları ve görüntü üzerindeki pikselleri düzenleyerek bir dizi filtreyi kullanan derin sinir ağlarıdır (Şekil 2.4.). Herbir katmanda giriş resmini belirli nöron işlemleri sonucunda soldan sağa doğru ileten CNN, son aşamada tam bağlı katman aracığıyla sınıflandırılma yapmaktadır [45].



Şekil 2.4. Nesne Algılama için örnek bir CNN Mimarisi [46].

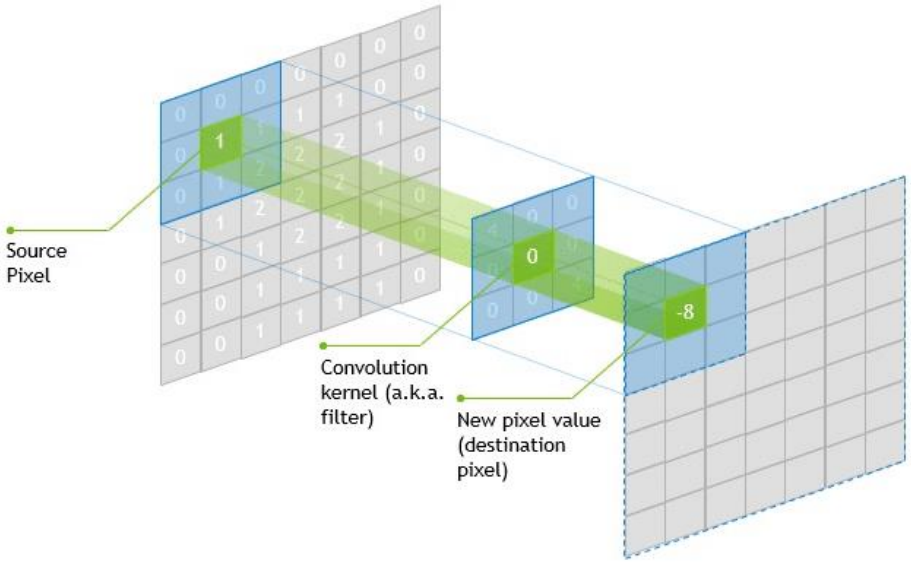
Temelde CNN mimarisi birbirinden farklı görevlere sahip üç ana katmandan oluşmaktadır. Bunlar evrişim, havuzlama ve tam bağlı katmanlardır.

2.3.1. Evrişim katmanı

Giriş görüntüsünün özelliklerinin çıkarıldığı evrişim katmanında görüntü üzerinde ilerleyen küçük kareler kullanılarak pikseller arasındaki ilişki öğrenilmektedir. Bu giriş verisi karelerine filtre veya kernel adı verilir. Filtrenin görüntünün üzerinde kaydırılması ve merkez noktanın hesaplanmasıyla oluşturulan matrise ise özellik haritası denilmektedir. Sahip olunan filtre sayısı ne kadar fazlaysa o kadar çok görüntü özelliği çıkartılır ve bu da ağın görüntülerdeki kalıpları tanımasına fayda sağlamaktadır. Özellik haritasının boyutu üç parametre ile kontrol edilmektedir [47].

- Derinlik, kullanılan filtre sayısını,
- Adım, giriş matrisi üzerinde kayan piksel sayısını,
- Sıfır dolgusu ise giriş matrisinin sınır çevresinde 0'larla doldurma işlemini

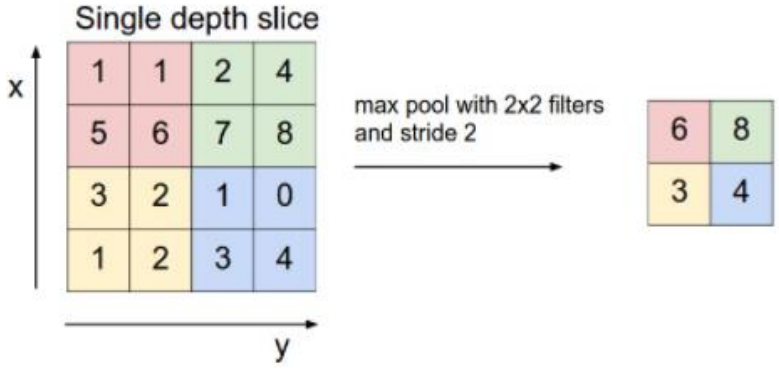
ifade etmektedir. Kernel'ın merkez hücresi kaynak pikselin üzerine yerleştirilir. Kaynak piksel daha sonra kendisinin ve yakındaki piksellerin ağırlıklı toplamı ile değiştirilir. Bu işlem Şekil 2.5.'te gösterilmiştir.



Şekil 2.5. Görüntü üzerinden kernel hücresi kullanılarak yapılan evrişim işlemi [48]

2.3.2. Havuzlama katmanı

Özellik haritasının boyutsallığını azaltmak için havuzlama işlemi gerçekleştirilmektedir. Bu işlem sinir ağındaki parametre ve hesaplamaların azaltılmasına ve overfitting (aşırı uyum) probleminin kontrol edilmesine yardımcı olur. Şekil 2.6.'da en yaygın havuzlama işlemlerinden olan max-havuzlama işlemi 4x4'lük karenin her 4 adet 2x2'lik hücresinden en büyüğünün alınması işlemi gösterilmiştir. Havuzlama katmanı sonucunda sinir ağı, giriş görüntüsündeki küçük dönüşümlere, bozulmalara ve uyumsuzluklara karşı stabil hale gelmektedir [47].



Şekil 2.6. Havuzlama işlemlerinden max-havuzlama işleminin yapılması [49]

2.3.3. Tam bağlı katman

CNN mimarisinde, evrişim ve havuzlama aşamalarından sonra ağın tamamlanması için tam bağlı katman eklenmektedir. Evrişim ve havuzlama katmanlarından elde edilen çıktı, giriş görüntüsünün üst düzey özelliklerini temsil etmektedir. Tam bağlı katmanda, giriş görüntüsünün eğitim veri kümesine göre çeşitli sınıflar sınıflandırılması için bu özellikler kullanılır. Sınıflandırmanın yanı sıra tam bağlı katmanların eklenmesi bu özelliklerin doğrusal olmayan kombinasyonlarının öğrenilmesine yardımcı olmaktadır. Şu halde bir CNN mimarisi iki ana görevi yerine getirmektedir. Öznitelik çıkarımı, evrişim ve katmanların birleştirilmesine yardımcı olurken, sınıflandırma işlemi ise tam bağlı katmanlarda eşleştirmeyi gerçekleştirmektedir. Genel olarak, ağın sahip olduğu evrişim adımları ne kadar fazla ise o ölçüde karmaşık özelliklerin öğrenilmesi kolaylaşmaktadır [47,49].

2.4. Evrişimsel Sinir Ağlarının Gelişimi ve Temel Mimariler

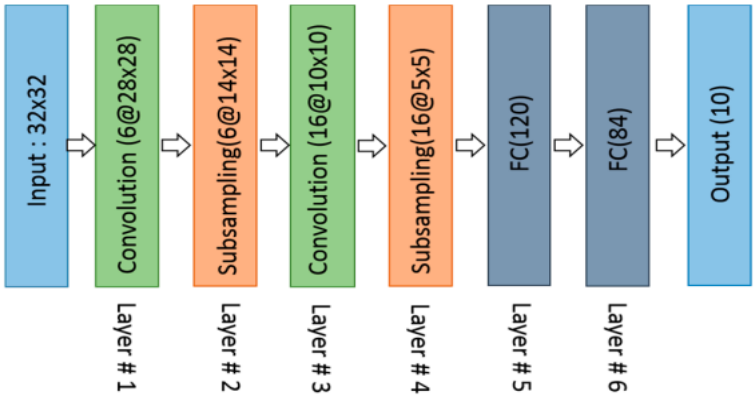
Evrişimsel Sinir Ağları (CNN), biyolojik görme çalışmalarının ilk bulgularından esinlenerek geliştirilmiş ve görüntülerdeki verileri işlemek için katmanlı mimaride tasarlanmış derin öğrenme modelleridir. Sinirbilimciler beynin görmesine yönelik işlevlerden hareketle matematiksel analize dayalı hesaplama modelleri geliştirmek için çeşitli çalışmalar yürütmüş ve birçok model ortaya atmışlardır.

Tarihsel olarak incelendiğinde, bilgisayarlı görme çalışmaları aslında 1950'lerde başlayarak devam etmiş ve çeşitli biyolojik deneylerle desteklenmiştir. 1959'de Hubel ve Wiesel görme korteks hücrelerindeki alıcıların parlayan ışık noktalarına verdiği tepkileri gözlemleyerek çeşitli çıkarımlar yapmışlar ve 1962'de ise retinaların basit ve karmaşık hücreleri tanımasının örüntü tanımada kullanılabileceğini ileri sürmüşlerdir [50,51].

Ardından 1980 yılında Fukushima, Hubel ve Wiesel'in bulgularını görsel sistemin çalışan bir modeline dönüştürmek için neocognitron adındaki ilk CNN ağını tasarlamıştır. Çok sayıda evrişim ve havuzlama katmanından oluşan ağ modeli elle yazılmış karakterleri tanımak için kullanılmıştır. Fakat eğitim için gereken hesaplama donanımının sınırları nedeniyle bu ağ modeli çok fazla yaygınlaşamamıştır [52,53].

2.4.1. LeNet mimarisi

1990'lara gelindiğinde ise Fukushima'nın çalışmalarının ilk modern CNN uygulaması gerçekleşmiştir. Yann LeCun ve arkadaşları el yazısıyla yazılan rakamları sınıflandırmak için CNN'lere gradyan tabanlı bir öğrenme algoritması uygulayarak LeNet mimarisini önermişlerdir [54]. Şekil 2.7.'de genel mimarisi ve Tablo 2.1.'de katman görevleri verilen LeNet mimarisi, yedi katmandan oluşmaktadır. Sınıflandırma yapmak için dönüşümlü olarak iki evrişim ve iki havuzlama katmanı ardından iki tam bağlı katman ile bir çıktı katmanından oluşmaktadır.



Şekil 2.7. LeNet mimarisi genel yapısı

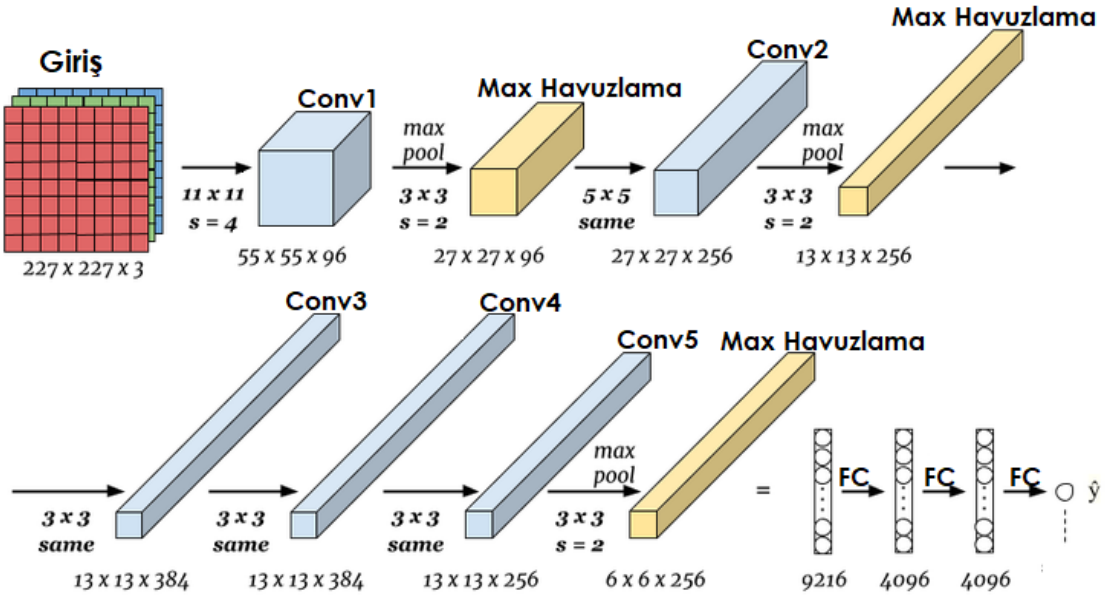
Tablo 2.1. LeNet mimarisi katmanları ve katman özellikleri

Katmanlar		***Özellik Haritası***	***Boyut***	***Çekirdek Boyutu***	***Adım***	***Aktivasyon Fonksiyonu***
Katman	**Katman Adı**	*1*	*32 x 32*	-	-	-
1	Convolution	6	28 x 28	5 x 5	1	tanh
2	Ortalama Havuzlama	6	14 x 14	2 x 2	2	tanh
3	Convolution	16	10 x 10	5 x 5	1	tanh
4	Ortalama Havuzlama	16	5 x 5	2 x 2	2	tanh
5	Convolution	120	1 x 1	5 x 5	1	tanh
6	FC	-	84	-	-	tanh
Çıktı	FC	-	10	-	-	Softmax

Tasarlanan bu ağ modeli görüntü üzerindeki rakamların tahmin edilmesi için MNIST veriseti ile eğitilmiş ve kullanılan bu CNN ağının önceki tüm tekniklerden daha iyi performans gösterdiği belirtilmiştir [54,55].

2.4.2. AlexNet mimarisi

2012'de LeNet'e kıyasla daha derin ve daha geniş bir CNN modeli olan AlexNet [56] tanıtılmıştır. AlexNet, önceki mimarilere göre görsel tanıma ve sınıflandırmada hata oranını %15'lere kadar düşürerek büyük bir başarı göstermiştir. İki adet Nvidia GTX 580 GPU kullanılarak 1.2M görüntüye sahip ImageNet veriseti ile eğitildi ve gelişmiş bir performans elde etti. Karmaşık nesneleri öğrenmek için büyük bir sinir ağı yapısında tasarlanan AlexNet'in genel mimarisi Şekil 2.8.'de gösterilmiştir.



Şekil 2.8. Alexnet mimarisi sınıflandırma adımları [57]

Bununla birlikte, katmanların yapmış oldukları işlevler Tablo 2.2.'de verilmiştir. Mimari beş evrişimli ve üç tam bağlı katman olmak üzere toplam 8 katmandan oluşur. Tam bağlı son katmanın çıktısında softmax fonksiyonu ile 1000 sınıf etiketi arasından tahmin yapılmaktadır. Birbirinden farklı boyut ve çekirdeklere sahip her katmanın çıkışında aktivasyon fonksiyonu bulunur ve ilk iki katman ile beşinci katmandan sonra boyut indirgemesi amacıyla havuzlama işlemi uygulanmaktadır [56]. Ağın ilk evrişimli katmanı öncelikle 224 x 244 x 3 boyutundaki giriş görüntüsünü 11 x 11 x 3 boyutundaki 96 çekirdek ile filtreler. İkinci evrişimli katmanda, 5 x 5 x 48 boyutundaki 256 çekirdek ile ilk evrişimli katmanın çıktısı filtreler.

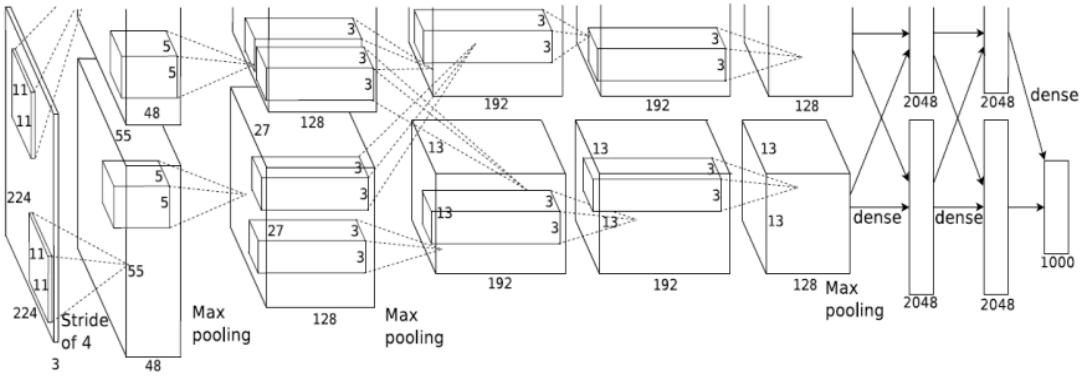
Tablo 2.2. AlexNet mimarisi katmanları ve katman özellikleri

Katmanlar		***Özellik Haritası***	***Boyut***	***Çekirdek Boyutu***	***Adım***	***Aktivasyon Fonksiyonu***
Katman	***Katman Adı***	1	227 x 227 x 3	-	-	-
1	Convolution	96	55 x 55 x 96	11 x 11	4	ReLU
	Max Havuzlama	96	27 x 27 x 96	3 x 3	2	ReLU
2	Convolution	256	27 x 27 x 256	5 x 5	1	ReLU
	Max Havuzlama	256	13 x 13 x 256	3 x 3	2	ReLU

Tablo 2.2. (Devamı)

	Katmanlar	*Özellik Haritası*	*Boyut*	*Çekirdek Boyutu*	*Adım*	*Aktivasyon Fonksiyonu*
3	Convolution	384	13 x 13 x 384	3 x 3	1	ReLU
4	Convolution	384	13 x 13 x 384	3 x 3	1	ReLU
5	Convolution	256	13 x 13 x 256	3 x 3	1	ReLU
	Max Havuzlama	256	6 x 6 x 256	3 x 3	2	ReLU
6	FC	-	9216	-	-	ReLU
7	FC	-	4096	-	-	ReLU
8	FC	-	4096			ReLU
Çıktı	FC	-	1000	-	-	Softmax

Üçüncü evrişimli katman, ikinci evrişimli katmanın çıktılarına bağlı 3 × 3 × 256 boyutunda 384 çekirdeğe sahiptir. Dördüncü evrişimli katman, 3x3x192 boyutunda 384 çekirdeğe ve beşinci evrişimli katman ise 3x3x192 boyutunda 256 çekirdeğe sahiptir. Bahsedilen üç, dört ve beşinci evrişim katmanında, araya giren herhangi bir havuzlama veya normalleştirme katmanı olmaksızın ardarda birbirlerine bağlıdırlar. Son olarak tam bağlı katmanların her birinde 4096 nöron bulunmaktadır (Şekil 2.9.).

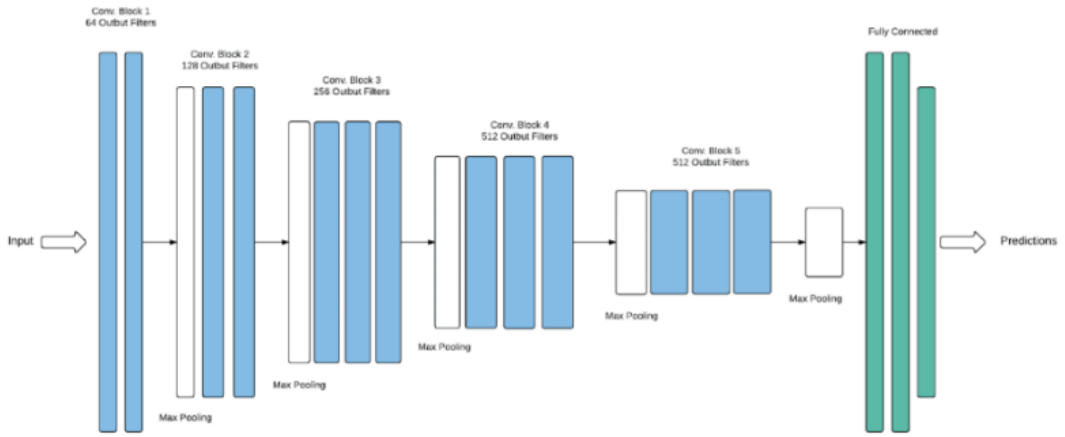


Şekil 2.9. AlexNet genel yapısı [56]

AlexNet sinir ağı mimarisi 60M parametre ile verisetinde 1000 farklı etiket sınıfına sahiptir. Çift GPU ve LeNet'e göre iyileştirilmiş bir mimari olmasına rağmen, parametrenin bu denli fazla olması nedeniyle yazarlar mimarinin aşırı uyum problemiyle karşılaştığını ve bu sebeple öğrenme yetersizliğine uğradını belirtmişlerdir. Bu sorunlarla mücadele edebilmek için iki ana yoldan bahsetmişlerdir. İlki veri sayısını genişletme, diğeri ise dropout yöntemi denilen gizli nöronların çıktısını %50 olasılıkla sıfırlamayı içermektedir.

2.4.3. VGGNET

2014'e gelindiğinde, Simonyan ve Zisserman tarafından görüntü tanıma için derin bir evrişimli sinir ağı olan VGGNet tanıtılmıştır [58]. 11 ila 19 katman arasında altı ağ modeline sahip olan mimarinin 16 ve 19 ağırlıklı modeli sınıflandırma ve lokasyon için en iyi sonucu vermiştir. Şekil 2.10.'da genel yapısı Tablo 2.3.'de katman özellikleri verilen mimari 3x3'lük kernel boyutuna sahip 5 evrişim katmanından oluşmaktadır. Görüntü her evrişim katmanından sonra havuz katmanı 2x2 boyutlarında olan max-havuzlama katmanından geçirilir. En sonuncu max-havuzlama katmanından sonra ise görüntünün özelliklerini çıkarmak için üç tam bağlı katmandan daha geçirilir. Bununla birlikte ağın son katmanı sınıflandırma ve normalleştirme için Softmax katmanını içermektedir [59].



Şekil 2.10. VGGNet mimarisi genel yapısı

Önceki CNN modelleriyle karşılaştırıldığında VGGNet'in en belirgin iyileştirmesi ise şu şekilde sayılabilir;

- Evrişim çekirdeğinin ve havuz çekirdeğinin boyutunu azaltmak,
- Evrişim katmanlarının sayısını ve derinliği arttırmak,
- Hızı arttırmak için parametre sayısını azaltmak,
- Parametreleri başlatmak için önceden eğitilmiş verileri kullanmaktır.

Tablo 2.3. VGG-16 mimarisinin katmanları ve katman özellikleri

Katmanlar		***Map***	***Boyut***	***Kernel Boyutu***	***Stride***	***Aktivasyon Fonksiyonu***
Katmanlar	**Katman Adı**	1	224 x 224 x 3	-	-	-
1	2 x Convolution	64	224 x 224 x 64	3 x 3	1	ReLU
	Max Havuzlama	64	112 x 112 x 64	3 x 3	2	ReLU
3	2 x Convolution	128	112 x 112 x 128	3 x 3	1	ReLU
	Max Havuzlama	128	56 x 56 x 128	3 x 3	2	ReLU
5	2 x Convolution	256	56 x 56 x 256	3 x 3	1	ReLU
	Max Havuzlama	256	28 x 28 x 256	3 x 3	2	ReLU
7	3 x Convolution	512	28 x 28 x 512	3 x 3	1	ReLU
	Max Havuzlama	512	14 x 14 x 512	3 x 3	2	ReLU
10	3 x Convolution	512	14 x 14 x 512	3 x 3	1	ReLU
	Max Havuzlama	512	7 x 7 x 512	3 x 3	2	ReLU
13	FC	-	25088	-	-	ReLU
14	FC	-	4096	-	-	ReLU
15	FC	-	4096	-	-	ReLU
Çıkış	FC		1000	-	-	Softmax

VGGNet modelinde tüm proseslerde 5x5 veya 7x7 gibi büyük evrişim çekirdekleri kullanmak yerine 3x3'lük küçük evrişim çekirdekleri kullanılmıştır. Ağın test aşamasında çok sayıda küçük evrişim çekirdeğini ard arda kullanmanın performans artışına büyük katkıda bulunduğu belirtilmiştir. Bunun nedeni ise görüntüden daha fazla ayrıntı yakalayabilmektir.

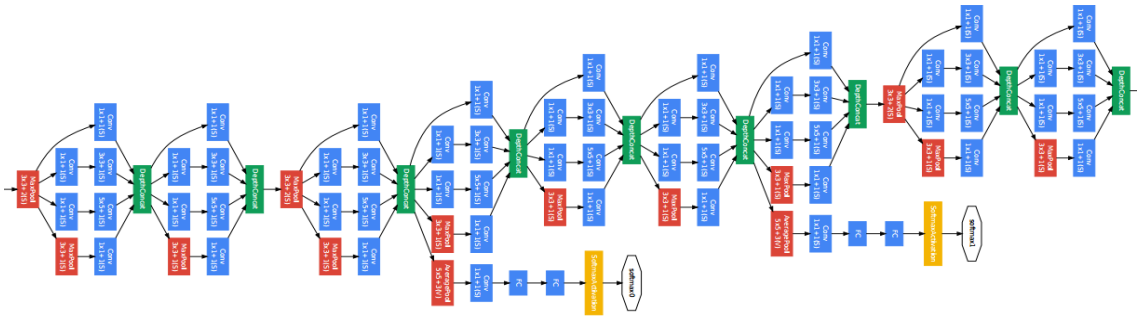
VGGNet'in farklı derinliklerindeki alt modellerinde ağ derinliği arttıkça performansta iyileşme görülmüştür. Açık bir şekilde 19 katmanlı VGGNet modeli görüntüyü 11 katmanlı olandan daha iyi tanımıştır. Örneğin, VGG16 haftalarca NVIDIA Titan Black GPU kullanarak eğitilmiştir. Model, 1000 sınıfa ait olan 14M'dan fazla görüntüye sahip ImageNet verisetinde ilk 5 testi % 92,7 doğruluğa ulaşmıştır (Tablo 2.4.).

Tablo 2.4. VGG-16 mimarisinin hassasiyet dereceleri [58]

Model	**VOC-2007 (Ortalama Hassasiyet)**	**VOC-2012 (Ortalama Hassasiyet)**	**Caltech-101 (Ortalama Hassasiyet)**	**Caltech-256 (Ortalama Hassasiyet)**
VGG -16 katmanlı	89,3	89,0	91.8±1,0	85.0±0,2
VGG -19 katmanlı	89,3	89,0	92.3±0,5	85.1±0,3
Model Comb.	89,7	89,3	92.7±0,5	86.2±0,3

2.4.4. GoogLeNet

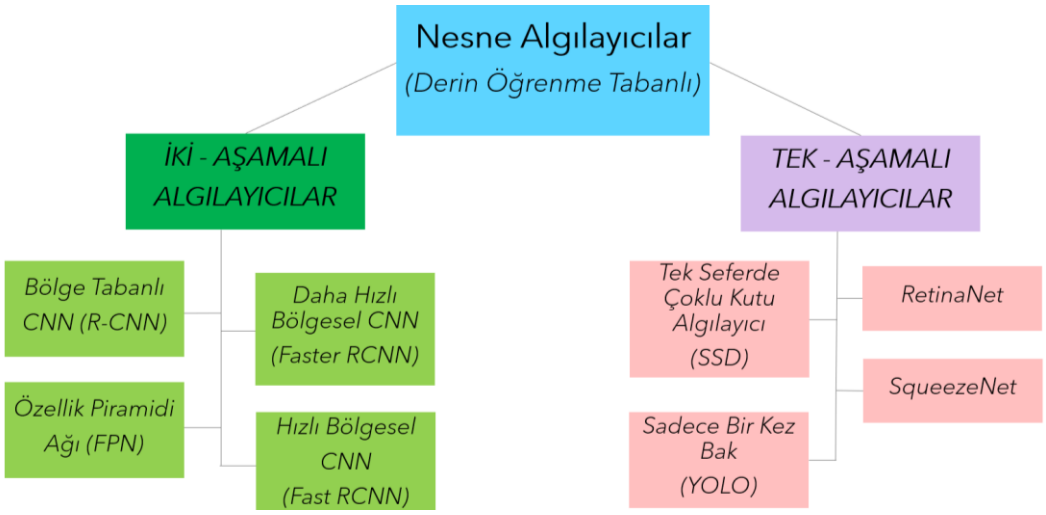
2014'de geliştirilen bir diğer ağ modeli ise ImageNet yarışmasında birinciliği kazanan GoogLeNet'dir. Mimari giriş görüntüsünü farklı filtreler ile filtreleyip daha sonra ağırlıklarla birleştirerek bir sonraki katmana aktarmaktadır. Evrişim, havuzlama ve diğer katmanlarla beraber toplamda 22 katmana sahiptir (Şekil 2.11.). Ağın en büyük avantajı daha derin ağlar için hesaplama maliyeti olan toplam parametre sayısını azaltmasıdır. Bununla beraber aynı derinlik seviyesinde farklı filtreler kullanılması da görüntülerden daha kaliteli öznitelik çıkarılmasına katkı sağlamaktadır [60].



Şekil 2.11. GoogLeNet genel yapısı

2.5. Derin Öğrenme Tabanlı Nesne Algılama Modelleri

Nesne algılama bir görüntüdeki nesneleri bulma ve sınıflandırma işlemidir. Doğruluk oranı ve hız açısından geliştirilen pek çok algoritma ve model mevcuttur. Genel olarak, nesne algılama için yapılan yaklaşım açısından iki gruba ayırılabilir (Şekil 2.12.).



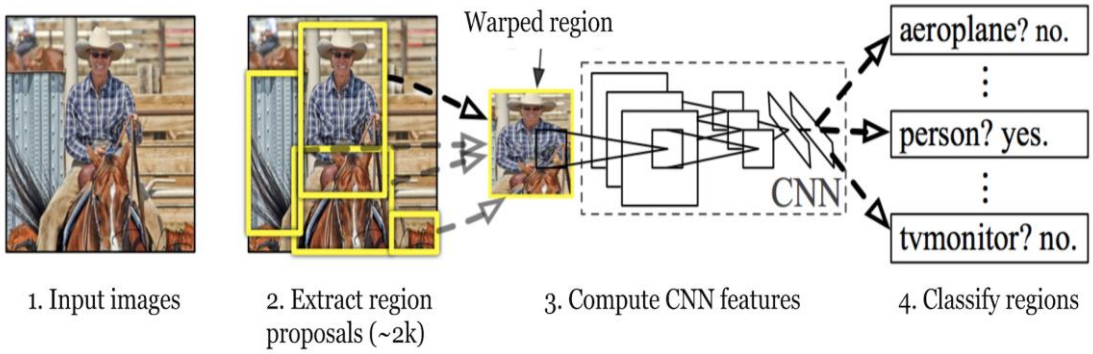
Şekil 2.12. Nesne algılama yaklaşımlarının yöntem açısından sınıflandırılması

2.5.1. İki aşamalı algılayıcılar

Sınıflandırma işlemi için bu algılayıcılarda, ilk olarak seçici arama kullanılarak yönetilebilir sayıda sınırlayıcı kutu nesne bölgesi adayı tanımlanır. Ardındanbağımsız olarak her bölgeden CNN öznitelikleri çıkarılarak sınıflandırma yapılmaktadır.

2.5.1.1. R-CNN

Girshick vd. tarafından Bölge Tabanlı Evrişimli Sinir Ağları (R-CNN) yaklaşımı iki aşamadan oluşmaktadır. Şekil 2.13.'te gösterilen R-CNN mimarisi, girdi görüntüsünü alarak her sınırlayıcı kutunun bir nesneyi ve o nesnenin kategorisini (örneğin araç, bisiklet veya insan) içerdiği sınırlayıcı kutuyu üreterek nesneyi çıktı olarak tahmin etmesiyle gerçekleşmektedir. Daha sonra yakın zamanlarda R-CNN ailesi, diğer bilgisayarla görme görevlerini gerçekleştirmek için genişletilmiştir [61-63].



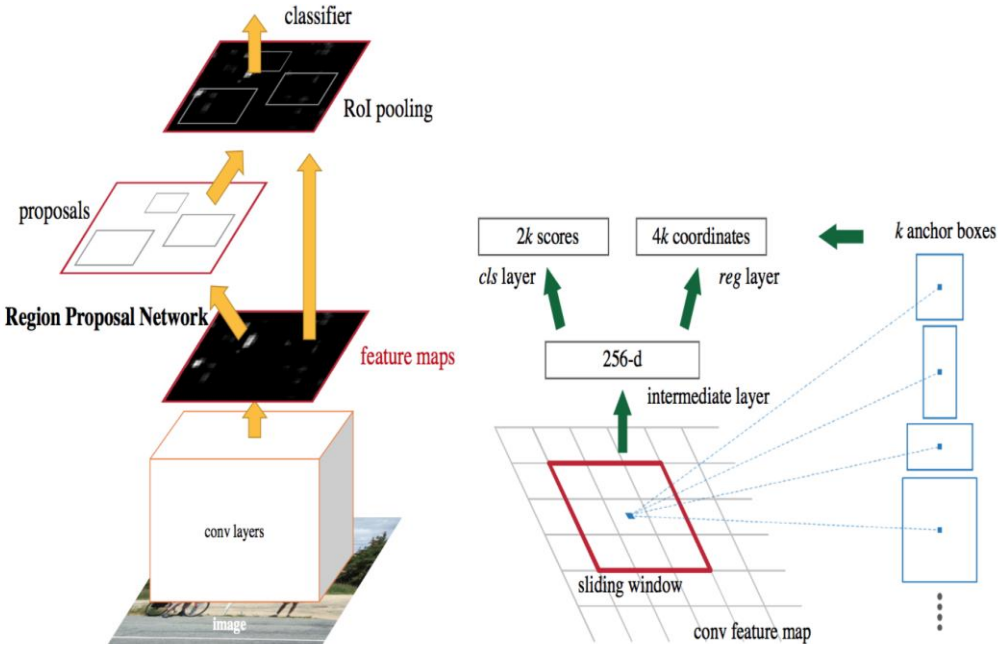
Şekil 2.13. R-CNN mimarisi sınıflandırma aşamaları

2.5.1.2. Fast R-CNN

Fast R-CNN nesne algılama yöntemi [64] Girshick tarafından uygulanmış ve SPPNet [65] ile R-CNN'nin geliştirilmiş hali olarak görülmektedir. Hem tespit edici hem de sınırlayıcı kutu regresyonunun aynı anda eğitilmesine izin veren bu model VOC-2007 verisetinde yapılan testlerde doğruluk oranını %58'den (R-CNN) %70'lere (Fast R-CNN) kadar yükseltmiştir. R-CNN ve SPPNet'in tüm avantajları Fast R-CNN ile başarılı bir şekilde birleştirilmiştir ancak yine de algılama hızı sınırlı kalmıştır. Bu dezavantajlar Faster R-CNN[66] ile ortadan kaldırılmıştır.

2.5.1.3. Faster R-CNN

R-CNN ve Fast R-CNN, bölge önerilerini bulmak için Seçici Arama Algoritması (SAA) kullanmakta ve bu algoritma ağ performansını etkileyen yavaş ve zaman alıcı bir süreç olduğu gözükmektedir. Bu nedenle, SAA' yı ortadan kaldıran ve ağın bölge önerilerini öğrenmesi için Bölge Teklif Ağı (RPN) yöntemi kullanan Faster R-CNN [66] algoritması geliştirilmiştir. Fast R-CNN'e benzer şekilde görüntü, bir evrişimsel özellik haritası sağlayan evrişimli ağa girdi olarak verilir (Şekil 2.14.).



Şekil 2.14. Faster R-CNN modelinin çalışma prensibi [66]

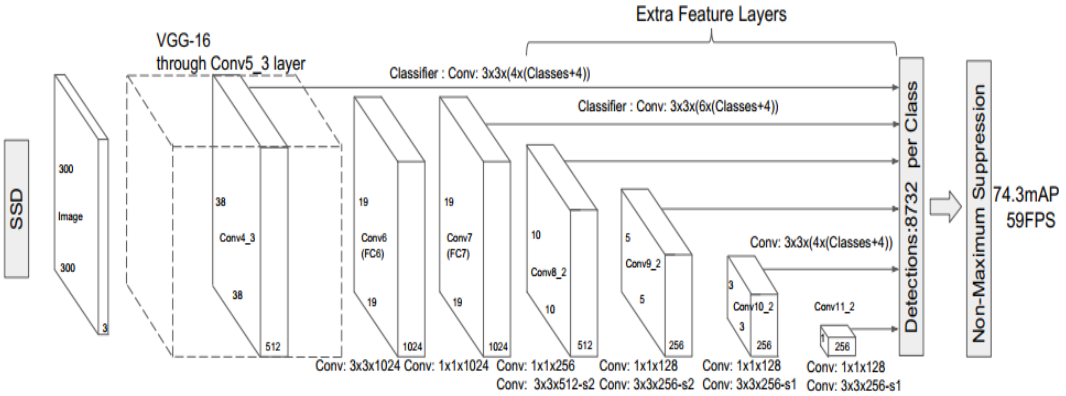
Ardından Faster R-CNN bölge önerisi için ayrı bir sinir ağı yapısı oluşturarak bölge önerileri yapmaktadır. Bundan sonraki ağın geri kalan kısımları Fast R-CNN ile benzerlik göstermektedir. Son aşamada ise sınıflandırma yapılarak ağ yapısı tespit işlemini tamamlamaktadır.

2.5.2. Tek aşamalı algılayıcılar

Tek aşamalı algılayıcılar, giriş görüntüsünün sinir ağından tek bir geçiş ile tahmin edildiği yaklaşımdır (SSD ve YOLO gibi). Bu yaklaşımda tespit işlemi ilk yaklaşıma göre çok daha hızlı, gerçek-zamanlı işlemlerde başarılı bir şekilde gerçekleşmektedir.

2.5.2.1. SSD

Liu vd. Tek-Seferde Çoklu Kutu Algılayıcı (Single-Shot Detector, SSD) mimarisini önerdiler [67]. SSD, derin öğrenme algoritmaları içerisinde gerçek zamanlı nesne algılamak için tasarlanmıştır. R-CNN ’de olduğu gibi bölge önerisi oluşturmak ve bu önerilerin herbirinin nesnesini algılamak için ağdan iki defa geçiş yapmak yerine, bir görüntüdeki birden fazla nesneyi algılamak için tek geçiş kullanır (Şekil 2.15.).



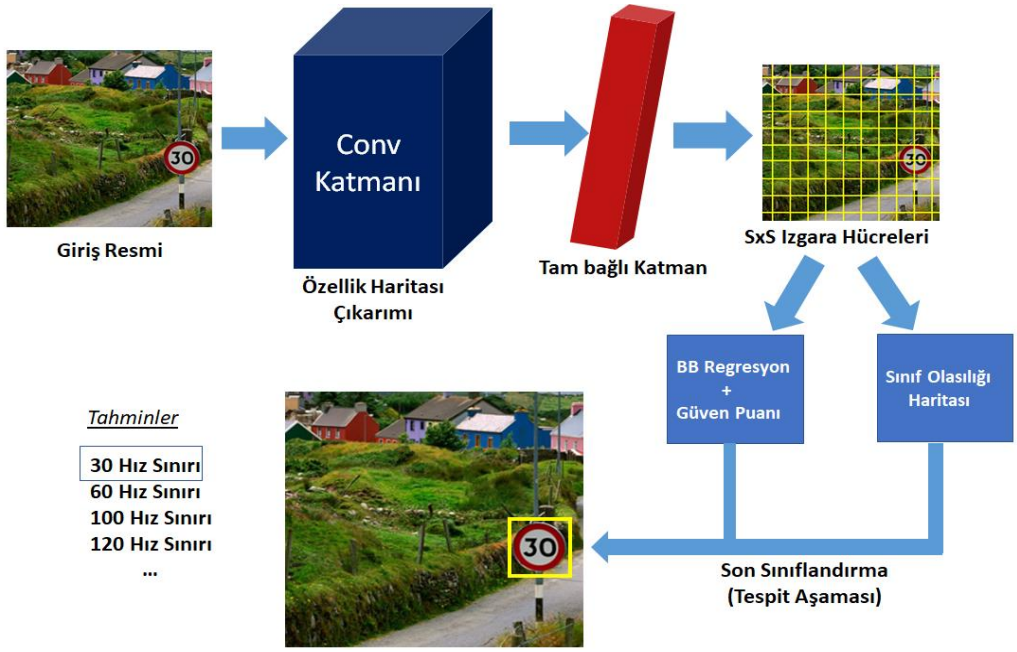
Şekil 2.15. SSD mimarisi genel yapısı [67]

SSD’nin algılama doğruluğunu özellikle de küçük nesneleri algılamada geliştirmek için çoklu referans ve çok çözünürlüklü algılama teknikleri de ayrıca tanıtılmıştır. Bununla birlikte SSD, Fast R-CNN'nin gerçek zamanlı hız algılama doğruluğunu iyileştirmek için Bölge Teklif Ağı’nı (RPN) ortadan kaldırmıştır.

2.5.2.2. YOLO

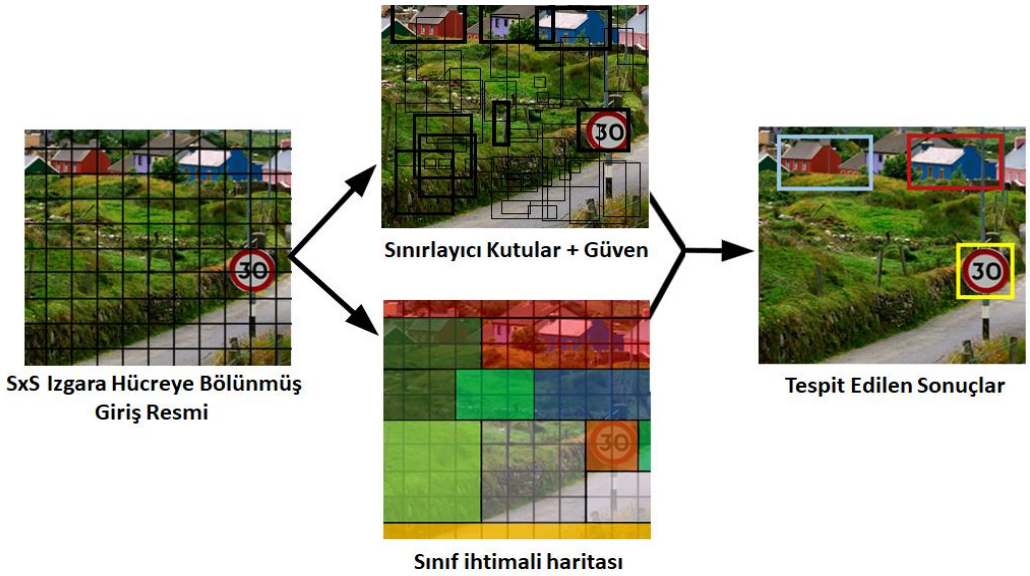
YOLO nesne algılama işlemi için CNN kullanan hızlı ve başarım oranı yüksek bir algoritmadır. Gerçek zamanlı tespit gerektiren işlemlerde sıklıkla kullanılmaktadır. Yöntem olarak tespit işlemini önceki algoritmalardan farklı kılarak regresyona dayalı bir yaklaşımla yapmaktadır [68]. R-CNN vd. geleneksel nesne tespit modellerinde görüntü içerisindeki nesneleri konumlandırmak için bölgeler kullanılırken [62,64], YOLO’da ise bu konumlandırma giriş görüntüsünün tek bir CNN ağından geçirilmesiyle sağlanmaktadır. Daha sonrasında görüntü bölgelere ayırılarak herbir

bölge için sınırlayıcı kutu ve sınıf olasılıkları tahmini yapılır. Bunun sonucunda Şekil 2.16.'da gösterildiği gibi tespit ve sınıflandırma işlemi tamamlanır.



Şekil 2.16. YOLO'nun nesne tespitine yönelik çalışma prensibi

Algoritmanın ağ modelinde herbir sınırlayıcı kutu aynı anda tahmin edilerek giriş görüntüsünden tespit amaçlanmaktadır. Sistem bu işlemi yaparken öncelikle giriş görüntüsünü S × S' lik ızgaralara bölmektedir. Bu bölmenin sonucunda bir nesnenin merkezi bir ızgara hücresinde ise o nesneyi tespit etmekten artık o ızgara hücresi sorumlu olacaktır. Şekil 2.17.'de YOLO'nun tespit etme işlemi gösterilmiştir.



Şekil 2.17. YOLO'nun girdi görüntüsü için güven skoru ve sınıf olasılıklarını tahmin adımları

Daha sonrasında herbir ızgara hücresi, B sınırlayıcı kutuları ve bu kutular için güven puanlarını tahmin eder. Bu güven puanları (confidence score), modelin kutunun bir nesne içerdiğinden ne kadar emin olduğunu ve ayrıca kutunun ne kadar doğru tahmin edildiğini yansıtmaktadır. Şu halde eğer bir hücrede nesne yoksa güven skoru sıfır olmaktadır. Bunun dışındaki durumlarda ise güven skorunun öngörülen kutu ile temel gerçek (ground truth) arasındaki birleşim üzerinden kesişme (IoU) ile eşit olmasını istenilir. Güven skorunun hesaplanma yöntemi (Denklem 2.1) 'de verildiği gibidir.

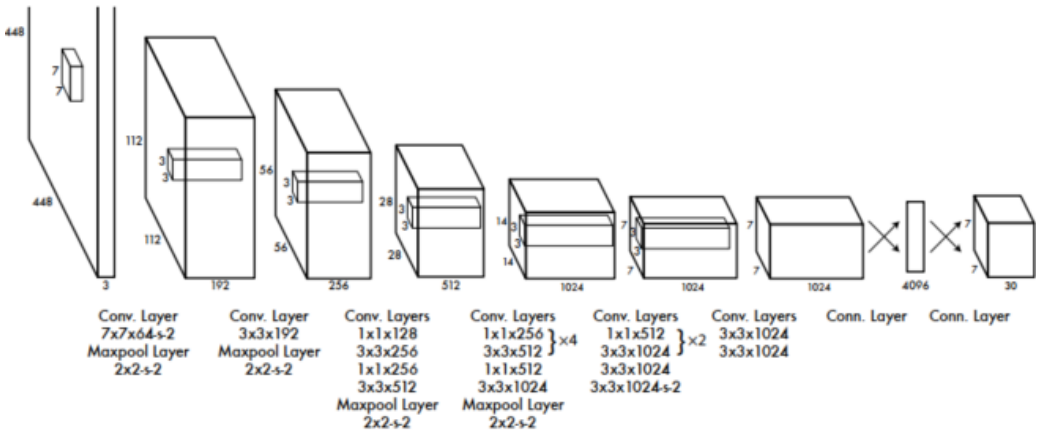
$$Güven\ Skoru = Pr\ (Nesne) * IOU_{pred}^{truth} \tag{2.1}$$

Sınırlayıcı kutuların herbirisi beş parametreyi tahmin etmekten sorumludur. Bunlar görüntü koordinatı (x,y), ebat (w,h) ve güven skorudur. Koordinatlardan x ve y ızgara hücresinin sınırlarına göre kutunun merkezini temsil ederken; w ve h ise genişlik ve yükseklikliktir. Son olarak güven skoru ise tahmin edilen kutu ile temel gerçek kutusu (ground truth box) arasındaki birleşim üzerinden kesişmeyi (IoU) temsil etmektedir. Her grid hücresi ayrıca C koşullu sınıf olasılıklarını, "$Pr\ (Class_i \mid Object)$" tahmin etmektedir. Bu olasılıklar, bir nesne içeren ızgara hücresi için (Denklem 2.2) 'deki gibi hesaplanır.

$$\begin{aligned} &\mathbf{Pr}(\boldsymbol{Class_i} \mid \boldsymbol{Object}) * \mathbf{Pr}(\boldsymbol{Object}) * \boldsymbol{IOU_{pred}^{truth}} \\ &\qquad = \mathbf{Pr}(\boldsymbol{Class_i}) * \boldsymbol{IOU_{pred}^{truth}} \end{aligned} \tag{2.2}$$

Kutuların sayısından bağımsız olarak, ızgara hücresi başına yalnızca bir sınıf olasılıkları kümesi tahmin edilmektedir. Test zamanında da koşullu sınıf olasılıkları ve kutu güven skoru tahminleri her kutu için sınıfa özgü güven puanları verilerek hesaplanır. Bu skorlar hem o sınıfın kutuda görünme olasılığını hem de tahmin edilen kutunun nesneye ne kadar iyi uyduğunu tutmaktadır. Ağ tasarımı olarak YOLO sırasıyla 24 evrişimli katman ve 2 tam bağlı katmandan oluşmaktadır. Evrişimli katmanlar görüntüden özellik çıkarırken, tam bağlı katmanlar ise çıktı olasılıkları ve koordinatları tahmin etmektedir. Mimarinin tam yapısı Şekil 2.18.'de verildiği gibidir. Bununla birlikte katman sayısı azaltılarak hızın arttırılması için Fast-YOLO modeli de geliştirilmiştir.

Bu versiyonda evrişim ve filtre sayısı azaltılarak (9 evrişim katmanı) ağı yapısı tasarlanmıştır. Ağın boyutu dışında, tüm eğitim ve test parametreleri YOLO ve Fast-YOLO için aynıdır.



Şekil 2.18. Nesne tespitinden sorumlu YOLO katmanlı mimarisi [68]

2.6. Yolo İçin Kayıp Fonksiyonu

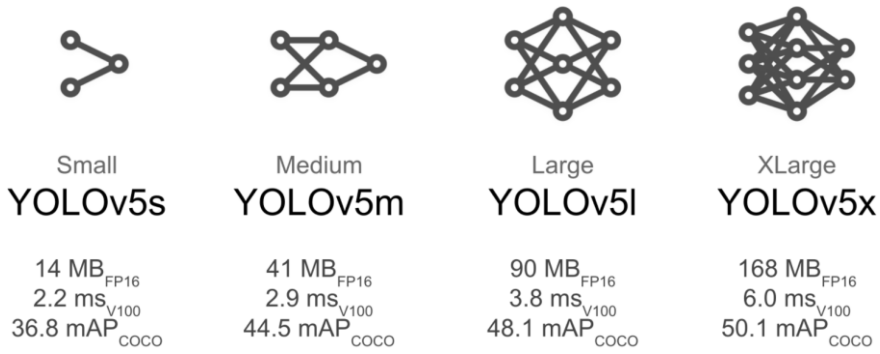
YOLO algoritması ızgara hücresi başına birden çok sınırlayıcı kutuyu öngörmektedir. Ancak eğitim sırasında her nesneden yalnızca bir sınırlayıcı kutu tahmincisinin sorumlu olması istenir. Bu yüzden gerçek doğruluğa en yakın IoU'ya sahip, bir nesneyi tahmin etmekten sorumlu olacak bir tahminci atanmaktadır. Bu işlem sınırlayıcı kutu tahmin edicileri arasında en doğrusunu bulmak için yapılmaktadır. Her tahmin edici belirli boyutları, en-boy oranlarını ve sınıfları tahmin etmeye çalışır. (Denklem 2.3)'de verilen çoklu kayıp fonksiyonu eğitim sırasında YOLO için optimize edilmektedir.

$$
\begin{aligned}
\textit{Kayıp Fonksiyonu} = {} & \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathrm{I}_{ij}^{obj} \left[\left(x_i - \hat{x}_i \right)^2 + \left(y_i - \hat{y}_i \right)^2 \right] \\
& + \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathrm{I}_{ij}^{obj} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
& + \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathrm{I}_{ij}^{obj} \left(C_i - \hat{C}_i \right)^2 + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathrm{I}_{ij}^{noobj} \left(C_i - \hat{C}_i \right)^2 \\
& + \sum_{i=0}^{S^2} \mathrm{I}_{i}^{obj} \sum_{c \in \text{ classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2
\end{aligned}
\tag{2.3}
$$

2.7. YOLO v5 Nesne Tespit Mimarisi

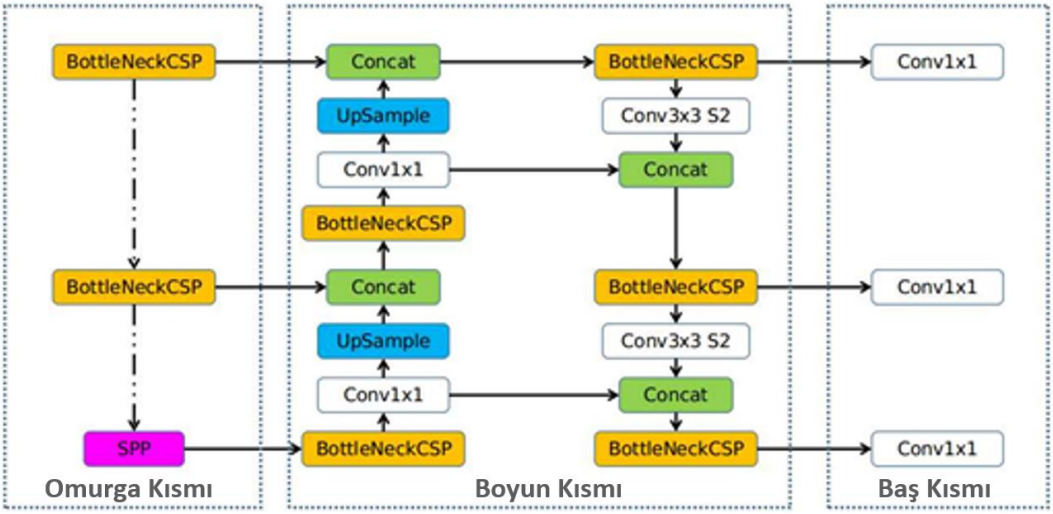
YOLO v4'ün yayınlanmasından kısa bir süre sonra Glenn ve ekibi tarafından mimarinin yeni sürümü olan YOLO v5 yayınlanmıştır. Mimarinin önceki sürümlerine göre işlem süresini önemli ölçüde düşürmesiyle birlikte YOLO v5, Pytorch'ta yeni bir eğitim ortamı altında derlenerek eğitim sürecini Darknet'ten daha kolay bir hale getirmiştir [69,70]. Bu ağ modelinin algılama doğruluğu ve hızı önceki sürümlere kıyasla oldukça yüksektir. Buna ek olarak, YOLO v5'in ağ modelinin ağırlık dosyasının boyutları da önceki sürüm olan YOLO v4'ten yaklaşık %90 oranla daha küçüktür [71].

Bütün bu avantajlar YOLO v5 modelinin gerçek zamanlı tespit uygulamaları için gömülü cihazlar üzerinde çalışması için uygun olduğunu göstermektedir [71,72]. YOLO v5 mimarisi boyut ve model parametrelerinin miktarına göre artan dört alt mimariyi içermektedir. Bunlar; YOLO v5s, YOLO v5m, YOLO v5l ve YOLO v5x mimarileridir. Bu mimariler sinir ağının belirli bir yerindeki özellik çıkarma modüllerinin ve evrişim çekirdek miktarlarının farklı olmasıyla birbirinden ayrılırlar. Model boyutları ve parametre sayısıda sırayla artmaktadır.



Şekil 2.19. YOLO v5 mimarisinin alt mimarileri [71].

Bu tez kapsamında geliştirilen sürücü asistan sistemi uygulamaları gömülü platform üzerinde gerçek zamanlı olarak çalışacağından ve sistemin minimum kaynak kullanımı ile maksimum performansa ulaşması istendiğinden ağın eğitimi YOLO v5s [71] ile gerçekleştirilmiştir. Genel yapısı Şekil 2.20.'de verilen YOLO v5 ağ mimarisi omurga, boyun ve baş kısmı olmak üzere üç bölümden oluşmaktadır.



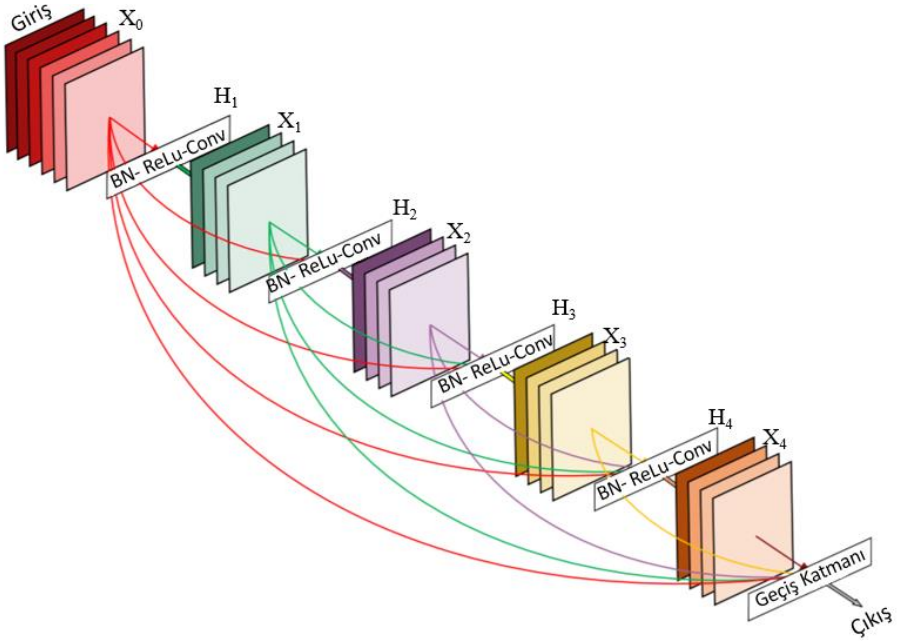
Şekil 2.20. YOLO v5 mimarisinin genel yapısı

Omurga kısmında CSPDarknet [73], boyun kısmında PANet [74] ve baş kısmında ise YOLO katmanı kullanılmaktadır. Özetle, veriler özellik çıkarımı için ilk olarak CSPDarknet'den geçirilerek özellik birleştirme için PANet'e verilir. Bunun sonucunda da YOLO katmanı sınıf, skor, konum ve boyut bilgisi gibi çıktıları vermektedir.

2.7.1. Omurga kısmı

YOLO v4'te omurga olarak kullanılan CSPNet aynı şekilde YOLO v5'te de bazı katmanlarda güncelleme yapılarak kullanılmıştır [75]. CSPNet modülü, büyük ölçekli omurgalarda tekrarlanan gradyan problemlerini çözer ve gradyan değişikliklerini özellik haritasına entegre ederek model parametrelerini azaltır. Bu işlem çıkarım hızını ve doğruluğu arttırmakla beraber model boyutunu da azaltır [76]. CSPNet modelleri DenseNet'e dayanmaktadır. DenseNet [77], CNN ağ katmanlarını birbirine bağlamak için tasarlanmıştır. Şekil 2.21.'de görülebileceği gibi her katman önceki katmanların tümünden ek girdiler alır ve kendi özellik haritalarını sonraki tüm katmanlara iletir [77]. (Denklem 2.4)'te verildiği gibi DenseNet, herbiri doğrusal olmayan H_i dönüşümünü uygulayan L katmanlarını içerir ve nihayetinde L'ninci katman girdi olarak x_0 ,x_1 ,x_2 ,…., x_L olmak üzere tüm önceki katmanların özellik haritalarını alır ki bu işlem özellik haritalarını ağın iyi bir şekilde özümsemesini sağlamaktadır.

$$x_i = H_L[(x_{L-1}, x_{L-2}, x_{L-3}, \ldots, x_0)] \quad (2.4)$$



Şekil 2.21. Büyüme hızı k=4 olan 5 katmanlı bir DenseNet bloğu [77].

2.7.2. Boyun kısmı

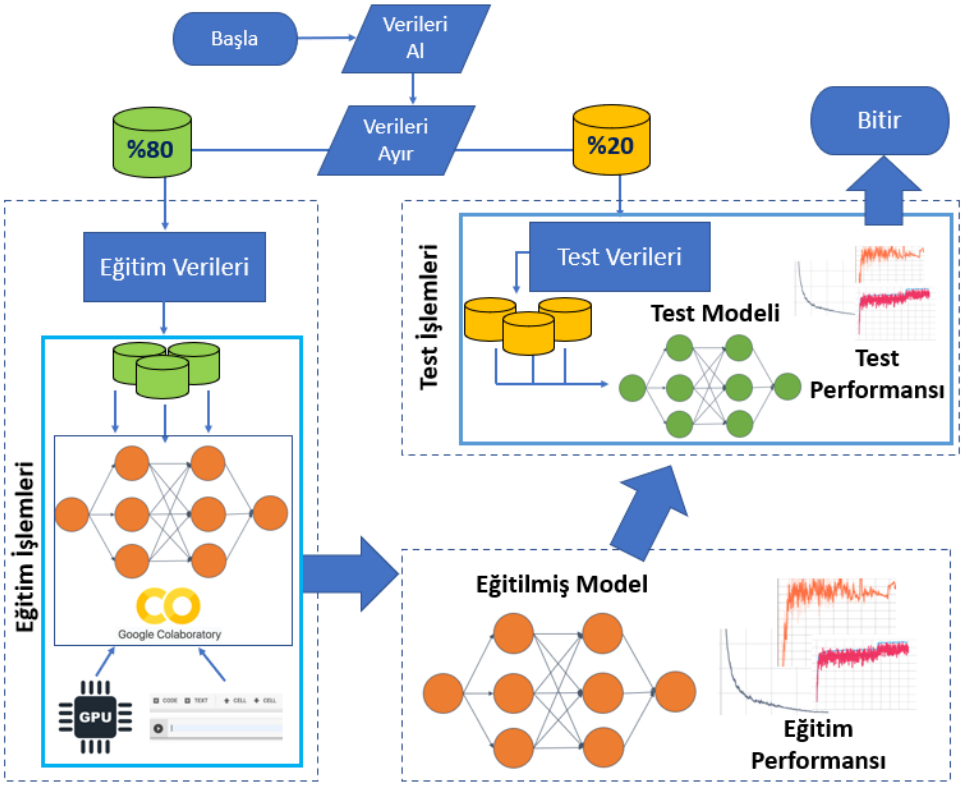
YOLO v5 boyun kısmında bilgi akışını arttırmak amacıyla PANet'i kullanmaktadır. PANet düşük seviyeli özelliklerin yayılmasını iyileştiren, aşağıdan-yukarıya yayılan FPNet[78] yapısını benimsemektedir. Özellik ızgarasını ve tüm özellik seviyelerini birbirine bağlayan özellik havuzu her bir özellik seviyesindeki yararlı verilerin alt ağa yayılmasını sağlamak için kullanılr. PANet nesnenin konum doğruluğunu arttırabilen alt katmanlarda doğru yerelleştirme sinyallerinin kullanımını iyileştirmektedir [74]. PANet, YOLOv5'te özellik seviyelerini korumak için kullanılmaktadır.

2.7.3. Baş kısmı

YOLO v5'in baş kısmı olan YOLO katmanı çok ölçekli tahmin elde etmek için 3 farklı boyutta (18 × 18, 36 × 36, 72 × 72) özellik haritası oluşturur ve modelin küçük, orta ve büyük nesnelerin tahmin edilmesi için kullanılan kısımdır. Baş kısmı olan YOLO katmanında veriler özellik çıkarımı için önce CSP Darknet'e girer ve sonrasında özellik birleştirme için PANet ile beslenir. Son olarak, YOLO katmanı algılama sonuçlarını (sınıf, puan, konum, boyut) vermektedir [80].

BÖLÜM 3. SÜRÜCÜ ASİSTAN SİSTEMİ UYGULAMALARININ GELİŞTİRİLMESİ

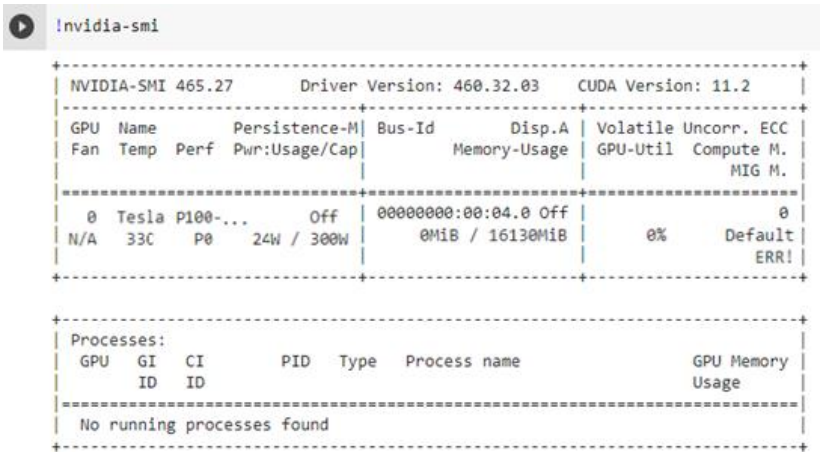
Tezin bu bölümünde araç içi ve araç dışı uygulamalara sahip ADAS sisteminin nesne ve davranışları tespit edebilmesi için oluşturulan modellerin gelişim aşamaları sunulmaktadır. Derin öğrenme uygulamaları yüksek hesaplama gücü ve işlem hızına ihtiyaç duymaktadır. Öğrenmenin sağlanabilmesi için gizli katman sayısının fazla olması ve ağırlıkların sürekli güncellenerek parametrelerin ayarlanması gereklidir. Google Colab, sunmuş olduğu GPU ve TPU birimleri ile donanım ve kurulum gerektirmeden online ortamda uygulamalar geliştirme imkanı sağlamaktadır. Çalışmada Colab Tesla P100 grafik işlemcisi kullanılarak her iki uygulama için de modeller geliştirilmiştir. Şekil 3.1.'de verilen geliştirme aşamaları bu bölümde detaylı olarak açıklanmaktadır.



Şekil 3.1. ADAS uygulamaları için model geliştirme aşamaları

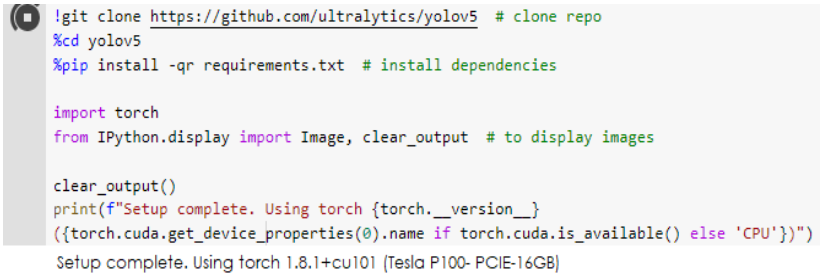
3.1. Yazılım Geliştirme Ortamının Hazırlanması

Google Colab tarayıcı aracılığıyla herhangi bir kurulum gerektirmeden makine öğrenmesi ve veri analizleri için python kodu yazılıp yürütülmesine izin veren entegre geliştirme ortamıdır (IDE) [82]. GPU ve TPU birimlerinin belirli limitlerde kullanımına izin vermektedir. Colab; Pytorch, Tensorflow, Keras ve OpenCV gibi pek çok kütüphaneye önceden yüklenmiş şekilde sahiptir. GPU (Tesla P100) ve TPU (TPU v2) desteği sunmakla beraber, 25 GB RAM ve 150 GB depolama alanına kadar birimin ücretsiz kullanımına imkan tanımaktadır [83,86]. Geliştirme ortamında GPU kullanım durumunu sorgulamak için !nvidia-smi komutu kullanılmaktadır (Şekil 3.2.).



Şekil 3.2. Colab'ın bulut servisinde Tesla P100- PCIE GPU ile eğitim

Önceki sürümlerde olduğu gibi, YOLO v5 mimarisi Gleen ve ekibi tarafından GitHub'da oluşturularak bir repoda [75] yayınlanmıştır. YOLO v5 PyTorch çerçevesi üzerine kuruludur. Bununla beraber geliştirilecek olan mimari katman ekleme, katman kaldırma, model optimizasyonu gibi problem tanımına bağlı olarak yapılandırabilir. Colab üzerinden YOLO v5 reposunun klonlanması Şekil 3.3.'te gösterilmiştir.



Şekil 3.3. YOLO v5 reposunun Github'dan klonlanması ve Colab'a yüklenmesi

Verisetleri Colab'a yüklenmeden önce Colab bir Google hizmeti olduğundan yüklemeler Google Drive hesabı üzerinden yapılmaktadır. Modeli eğitmek ve daha sonrasında sonuçları kaydederek Drive'dan veri almak için bu işlemin yapılması gerekmektedir. Bu nedenle verisetlerinin YOLO formatına uygun olarak etiketlemeleri yapıldıktan sonra hazırlanan bu dosyalar Drive'daki YOLO v5 adlı klasöre taşınmıştır (Bu ayırma işlemleri ilerleyen safhalarda anlatılmıştır). Daha sonrasında Colab'dan gerekli kodlarla veriler çekilerek aktarım sağlanmıştır (Şekil 3.4.).

```
!zip -r /content/yolo.zip /content/yolov5
#Proje dosyalarını çekmek için

from google.colab import files
files.download("/content/yolo.zip")
```

Şekil 3.4. Dosyaların çekilerek Drive'dan Colab'a aktarılması

YOLO v5'in sunduğu dört alt mimari içinden eğitim için YOLO v5s seçilmiştir. Sınırlı kaynak ve hesaplama için kullanılması uygun olan YOLO v5s'in Şekil 3.5.'te mimarisinin genel yapısı kod blokları ile verilmiştir. PyTorch üzerine kurulu olan YOLO v5 modeli, yaml dosyası formatındaki bu dosyayı okuyarak eğitim dosyasını verisetindeki verileri okuyarak ağırlıklandırır ve böylece modelin oluşmasını sağlar.

yolov5s.yaml

```yaml
# parameters
nc: 80  # number of classes
depth_multiple: 0.33  # model depth multiple
width_multiple: 0.50  # layer channel multiple

# anchors
anchors:
  - [10,13, 16,30, 33,23]  # P3/8
  - [30,61, 62,45, 59,119]  # P4/16
  - [116,90, 156,198, 373,326]  # P5/32
# YOLOv5 backbone
backbone:
  # [from, number, module, args]
  [[-1, 1, Focus, [64, 3]],  # 0-P1/2
   [-1, 1, Conv, [128, 3, 2]],  # 1-P2/4
   [-1, 3, C3, [128]],
   [-1, 1, Conv, [256, 3, 2]],  # 3-P3/8
   [-1, 9, C3, [256]],
   [-1, 1, Conv, [512, 3, 2]],  # 5-P4/16
   [-1, 9, C3, [512]],
   [-1, 1, Conv, [1024, 3, 2]],  # 7-P5/32
   [-1, 1, SPP, [1024, [5, 9, 13]]],
   [-1, 3, C3, [1024, False]],  # 9
  ]

# YOLOv5 head
head:
  [[-1, 1, Conv, [512, 1, 1]],
   [-1, 1, nn.Upsample, [None, 2, 'nearest']],
   [[-1, 6], 1, Concat, [1]],  # cat backbone P4
   [-1, 3, C3, [512, False]],  # 13

   [-1, 1, Conv, [256, 1, 1]],
   [-1, 1, nn.Upsample, [None, 2, 'nearest']],
   [[-1, 4], 1, Concat, [1]],  # cat backbone P3
   [-1, 3, C3, [256, False]],  # 17 (P3/8-small)

   [-1, 1, Conv, [256, 3, 2]],
   [[-1, 14], 1, Concat, [1]],  # cat head P4
   [-1, 3, C3, [512, False]],  # 20 (P4/16-medium)

   [-1, 1, Conv, [512, 3, 2]],
   [[-1, 10], 1, Concat, [1]],  # cat head P5
   [-1, 3, C3, [1024, False]],  # 23 (P5/32-large)

   [[17, 20, 23], 1, Detect, [nc, anchors]],
  ] # Detect(P3, P4, P5)
```

Şekil 3.5. YOLO v5s mimarisi katmanları kod blokları

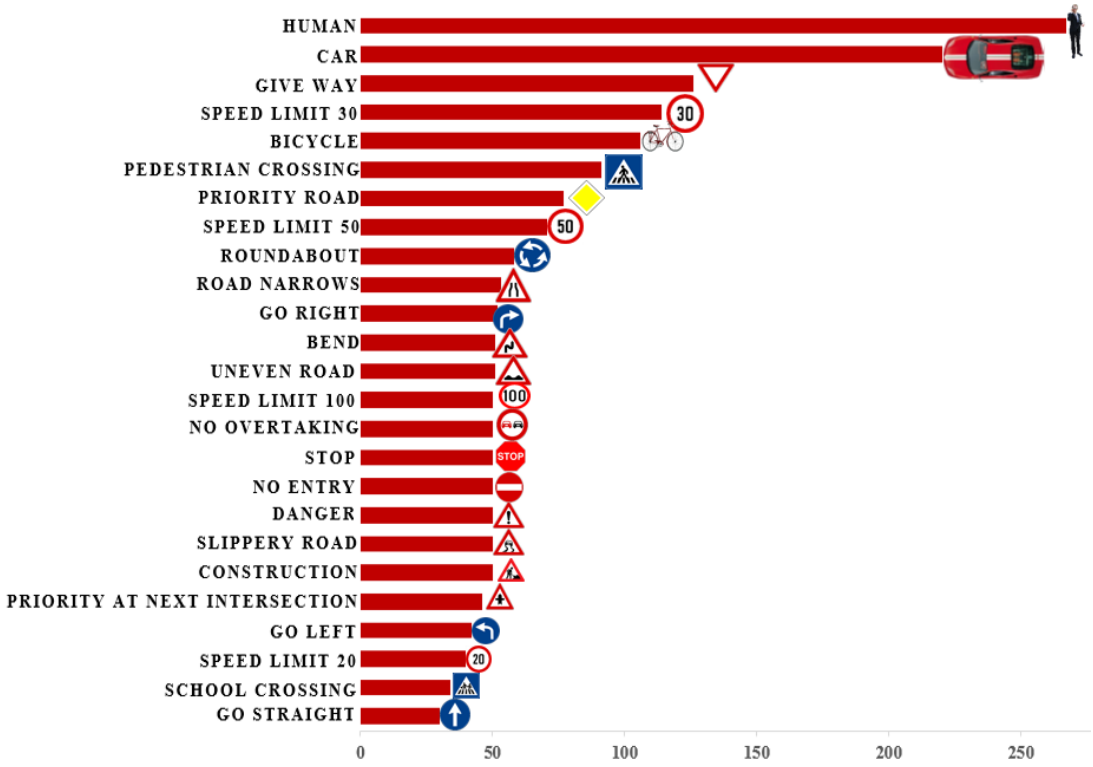
3.2. Verisetlerinin Hazırlanması

Araç dışındaki işaret ve nesneleri algılamak için oluşturulan model iki adet veriseti ile eğitilmiştir. Bunlardan ilki global bir veriseti Alman Trafik İşareti Algılama [12] (GTSRB) ve diğeri ise çalışmaya özgü toplanılmış gerçek görüntüleri içeren veri setimizdir. Verisetlerinden örnek görüntüler, Almanya'da çekilen GTSRB veriseti için Şekil 3.6.'da ve hazırlanılan özgün veriseti için ise Şekil 3.7.'de gösterilmiştir.

Şekil 3.6. GTSRB verisetinden örnek resimler [12]

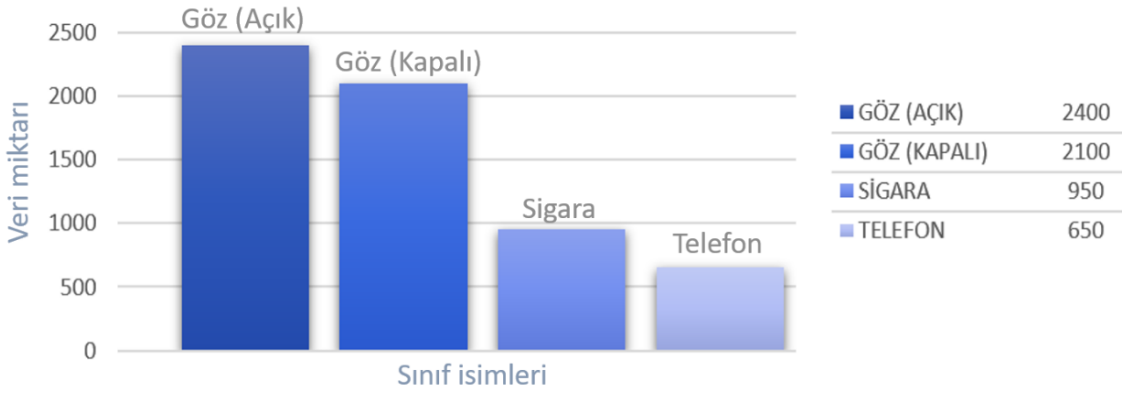
Şekil 3.7. Araç dışı tespit uygulaması için oluşturulan lokal veriseti

Araç dışındaki araba, insan, bisiklet ve 22 farklı trafik işaretini tespit etmek için bir araya getirilen verisetlerindeki (GTSRB ve lokal) herbir sınıf farklı miktarlarda veriler içermektedir. Eğitim-test işlemleri için bu veriler dosyalara ayırılmıştır. Şekil 3.8.'de araç dışı tespit için toplanılan sınıfların ad ve verisetindeki dağılımları gösterilmiştir.



Şekil 3.8. Trafik işareti ve nesne tespiti sınıflarının verisetindeki miktarları

Bununla birlikte sürücü davranışlarını tespit için geliştirilen uygulamanın veriseti ise açık ve kapalı göz, sigara içme ve telefon kullanma sınıfları için çeşitli sürücü fotoğraflarını içermektedir. Bu verilerin miktarca dağılımı ise Şekil 3.9.'da verilmiştir.



Şekil 3.9. Sürücü davranış tespiti sınıflarının verisetindeki miktarları

Tez çalışmasında araç dışı tespit için oluşturulan verisetinden sonra yapılan ikinci uygulama olan, araç içi davranış ve durumları tespit işlemi için ağın eğitiminde tamamen özgün sürücü görüntülerini içeren oluşturduğumuz veriseti kullanılmıştır. Eğitimde en iyi sonuçları elde edebilmek için resimler farklı ışık ve konumlarda elde edilmiştir. Kullanılan bu verilerden örnek görüntüler Şekil 3.10.' da gösterilmiştir.

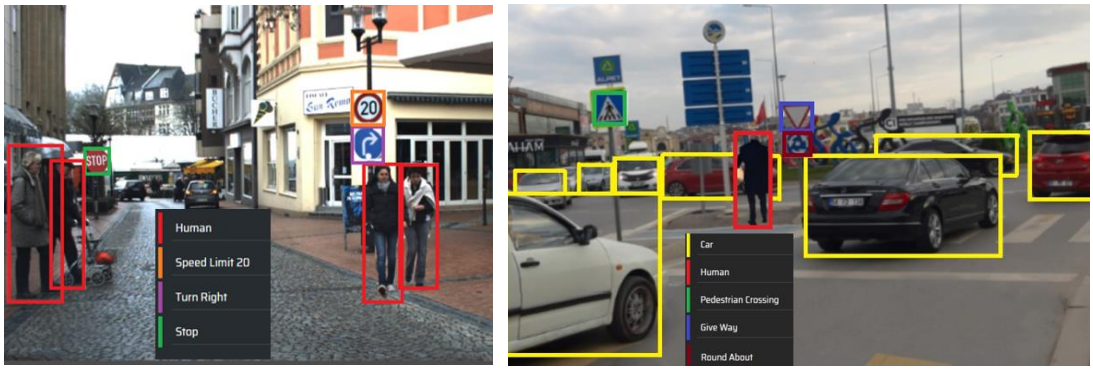
Şekil 3.10. Araç içi sürücü davranışlarını tespit için kullanılan verisetinden örnek resimler

3.2.1. Verileri etiketleme süreci

Verisetlerindeki resimler etiketlenirken, makesense.ai [81] adlı yazılım aracı kullanılmıştır. Makesense.ai fotoğrafları etiketlemek için kullanımı ücretsiz bir çevrimiçi araçtır. Bu programda görüntüler incelenerek nesneler sınırlayıcı kutular ile seçilerek etiketlenmiştir. Etiketleme sürecine ait örnek resim Şekil 3.11.'de verilmiştir.

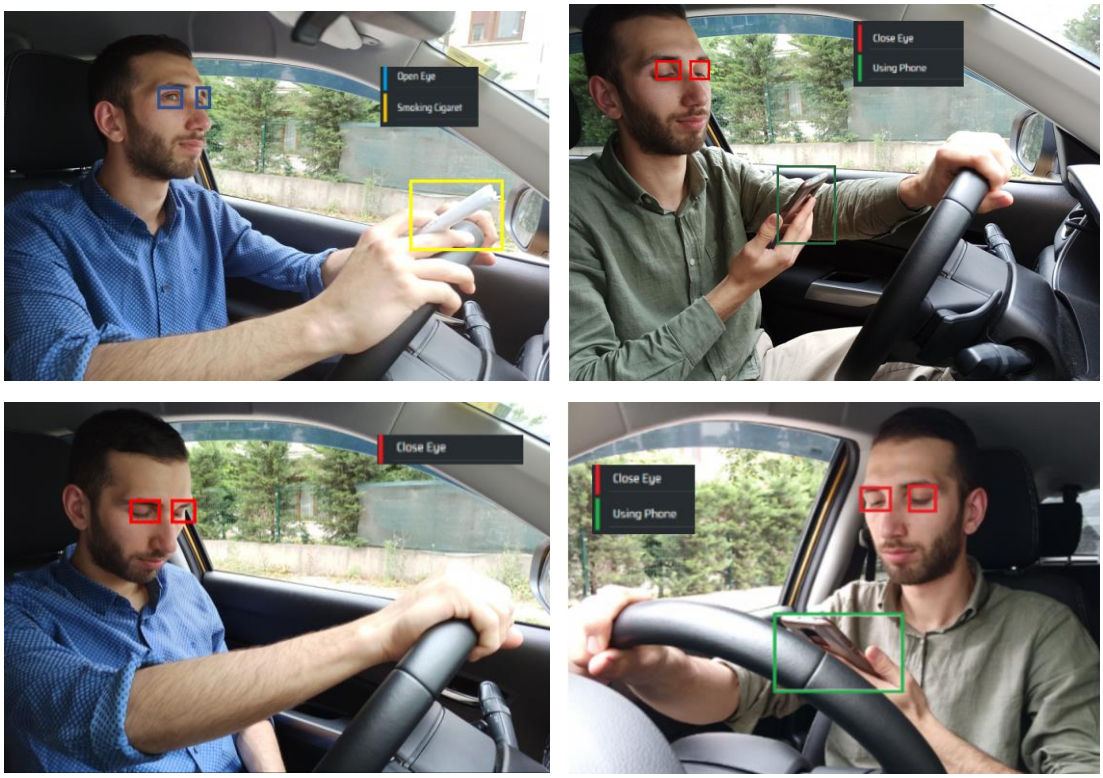
Şekil 3.11. Araç dışı ADAS uygulaması için insan, araç ve trafik işaretlerinin etiketlenmesi

Çalışmanın performansı etiketlenecek olan bu görüntülerle doğrudan ilişkili olduğundan, bu süreçte nesnelerin doğru şekilde seçilmesi önemlidir (Şekil 3.12.).



Şekil 3.12. Araç dışı ADAS uygulaması için etiketleme işlemi

Araç içi tespit işlemine gelindiğinde, sürücünün araç kullanırken gerçekleştirmiş olduğu farklı davranışları tespit etmek için çalışmada dört farklı sınıf kullanılmıştır. Bunlar gözlerin açık ve kapalı olma durumu, sigara ve telefon kullanımı tespitleridir. Etiketleme işlemine ait örnek görüntüler Şekil 3.13.'te gösterilmiştir. Böylece hazırlanacak olan model için gerekli veriler eğitim ve test için hazır hale getirilmiştir.



Şekil 3.13. Araç içi ADAS uygulaması için sürücü durumlarının etiketlenmesi

3.2.2. Verilerin eğitim ve test için ayırılması

Tanıtılan verisetlerinin eğitim ve test için ayırılmasına bakıldığında, araç içi ve araç dışı uygulamalar için;

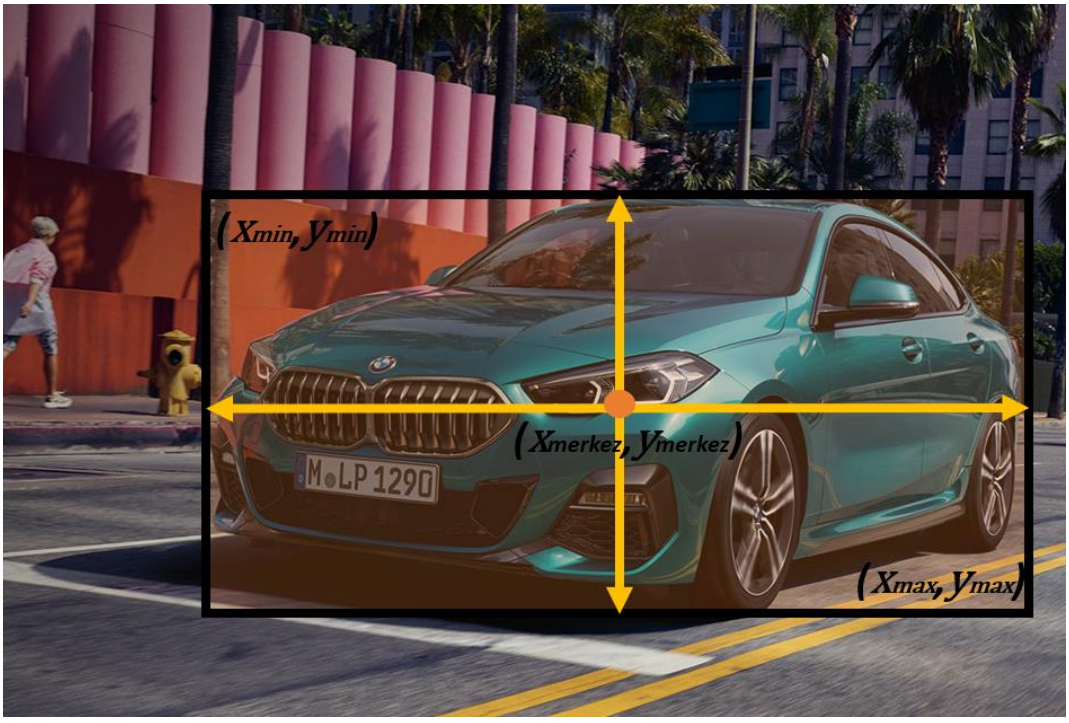
Trafik işareti ve nesne tespiti verisetinde toplanılan 1850 görüntüden;

- train.zip – 1350 görüntü eğitim için ayırıldı.
- test.zip – 500 görüntü test için seçildi.

Araç içi uygulamaları için toplanılan verisetinde 3500 görüntüden;

- train.zip – 2800 görüntü eğitim için ayırıldı.
- test.zip – 700 görüntü test için seçildi.

YOLO formatında hazırlanılan bir resim için sınırlayıcı kutuların nasıl göründüğü Şekil 3.14.'te gösterilmektedir. Verinin içeriği sınıflandırılacak nesneyi tam kapsayacak şekilde içeren sınırlayıcı kutuların koordinatlarıdır ve sınırlayıcı kutu koordinatları 0-1 arasında normalize edilmelidir. YOLO'daki sınırlayıcı kutu verileri [sınıf_no, x_merkezi, y_merkezi, genişlik, yükseklik] olarak formatlanmaktadır.



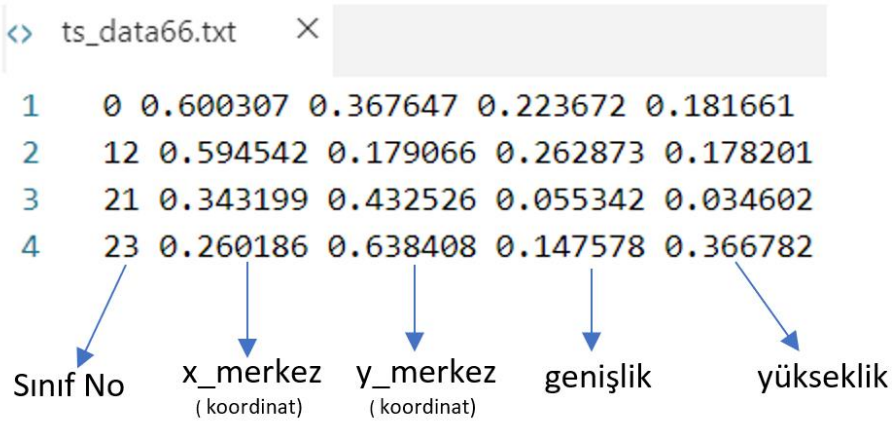
Şekil 3.14. Sınırlayıcı kutunun resim üzerinde gösterilmesi

Bu formattaki dosya içeriği sınırlayıcı kutunun bilgilerini tutmaktadır. Verinin içeriği sınıflandırılacak nesneyi tam kapsayacak şekilde içeren sınırlayıcı kutuların koordinatlarıdır. Minimum nokta olan (x_{min}, y_{min}) ile merkez noktası (x_{merkez}, y_{merkez}) (Denklem 3.1) ve (Denklem 3.2) 'deki gibi hesaplanmaktadır.

$$x_{merkez} = x_{min} + \frac{genişlik}{2} \tag{3.1}$$

$$y_{merkez} = y_{merkez} + \frac{yükseklik}{2} \tag{3.2}$$

Bu denklemlerden de görülebileceği gibi x_{merkez} ve y_{merkez} görüntünün x ve y koordinatlarının merkez noktaya olan uzaklığını temsil etmektedir. Train.cvs dosyasındaki eğitim verileri okunur. Buradaki herbir satır, bir sınırlayıcı kutunun verisini temsil etmektedir. Verilerin etiketlenme aşamasında gösterildiği gibi bir resim birden fazla nesne içerebilir, bu nedenle bir görüntüde birden çok sınırlayıcı kutu bulunabilmektedir. Her bir veri sütunu sırasıyla benzersiz görüntü kimliğine, görüntünün genişliğine, yüksekliğine, sınırlayıcı kutunun verilerine [$xmin$, $ymin$, $genişlik$, $yükseklik$] karşılık gelmektedir. Şekil 3.15.'de gösterilen veri dosyası, nesne algılama veri kümesindeki sınırlayıcı kutular için kullanılan formattır.



Şekil 3.15. YOLO formatına göre hazırlanmış dosya içeriğindeki alanlar

3.3. Yazılım Ortamının Konfigürasyonu

YOLO formatına uygun etiketlenen veriseti ardından yazılım ortamının konfigüre edilmesiyle Google Colab üzerinde eğitilmiştir. Verisetlerinin eğitimi için etiketlere

özel yaml dosyası hazırlanarak Google Colab'a yüklenmiştir. PyTorch üzerindeki YOLO v5 modeli görüntülere erişerek veri seti hakkında özet bilgiler içeren bir yaml dosyası aracılığıyla girdi olarak kullanır [84]. YOLO modelindeki data.yaml dosyasında belirlenmiş alanlar yapısal olarak vardır. Bu dosyada train değişkeni, eğitim verileri için; val değişkeni ise doğrulama verileri için dizin yolunu ifade etmektedir. Sınıf sayısını nc adındaki değişken tutmaktadır ve en son değişken names ise nesnelerin sırasıyla adının tutulduğu değişkendir. Buradaki veriler Şekil 3.15.'te gösterilen txt formatındaki YOLO verileriyle sınıf numarasına göre eşleşerek names değişkeni sınıfların ait olduğu indeks numarasına göre isimlerini belirlemektedir (Şekil 3.16. ve Şekil 3.17.).

custom_data.yaml

```yaml
# train-val data as
# 1) directory: path/images/,
# 2) file: path/images.txt, or
# 3) list:[path1/images]

train: ../train_data/images/train/
val: ../train_data/images/val/

# number of classes
nc: 25

# class names
names: [ 'Speed Limit 20', 'Speed Limit 30', 'Speed Limit 50',
         'Speed Limit 100', 'No Overtaking', 'Priority at Next
         Intersection', 'Priority Road', 'Give Way', 'Stop',
         'No Entry', 'Danger','Bend', 'Uneven Road', 'Slippery Road',
         'Road Narrows', 'Construction', 'School Crossing', 'Go Right',
         'Go Left','Go Straight','Roundabout', 'Pedestrian Crossing',
         'Car','Human', 'Bicycle']
```

Şekil 3.16. Araç dışı nesne tespiti için hazırlanan custom_data.yaml dosya içeriği

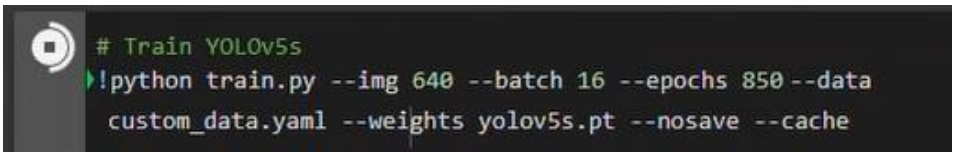
custom_data.yaml

```yaml
# train-val data as
# 1) directory: path/images/,
# 2) file: path/images.txt, or
# 3) list:[path1/images]

train: ../train_data/images/train/
val: ../train_data/images/val/

# number of classes
nc: 4

# class names
names: [ 'open_eye', 'phone', 'cigaret', 'close_eye']
```

Şekil 3.17. Araç içi nesne tespiti için hazırlanan custom_data.yaml dosya içeriği

3.4. Verilerin Eğitilerek Modellerin Oluşturulması

Her iki model de önceden ağırlıklandırılmış YOLO v5s ile Şekil 3.18.'de gösterilen komut satırında görüldüğü gibi eğitilerek daha sonra gömülü platform ve bilgisayar ortamında test edilecek olan .pt uzantılı modeller haline getirilmiştir.



Şekil 3.18. Önceden ağırlıklandırılmış YOLO v5s ile modelin eğitilmesi

Model eğitiminde kullanılan parametrelerden;

- img : Giriş görüntüsünün boyutunun verildiği parametredir. Orijinal giriş görüntüsünü sıkıştırarak eğitim sürecini hızlandırmayı amaçlar.

- batch : Binlerce görüntünün aynı anda sinir ağına iletilmesi, modelin bir epoch'ta öğrendiği ağırlık sayısının çok artmasına sebep olmaktadır. Bu nedenle, veri kümesi genellikle birden çok n görüntü grubuna ve toplu batch'lere bölünmektedir.

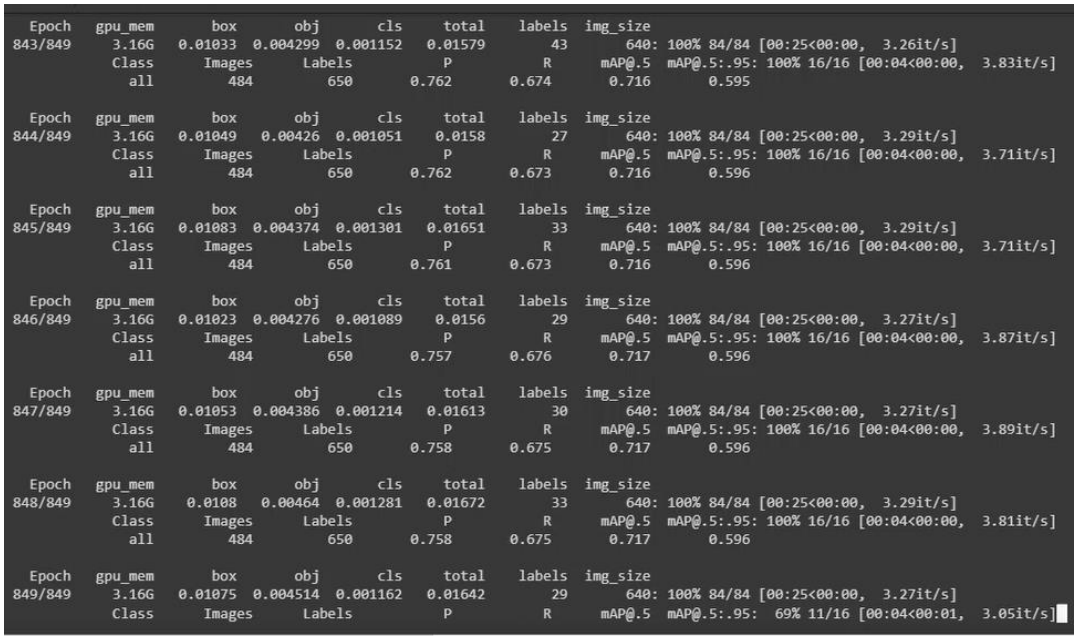
- epochs : Epoch sayısı öğrenme algoritmasının tüm eğitim veriseti boyunca geçiş yaptığı süreyi tanımlayan parametredir.

- data : Verisetinin içeriğindeki sınıfların listesini içeren, yaml uzantılı dosyanın yolunu belirtmektedir.

- weights : Önceden eğitilmiş bir ağırlık kullanmak eğitim süresini kısaltacaktır. Eğer bu kısım boş bırakılırsa model eğitim için rastgele ağırlıkları otomatik başlatmaktadır.

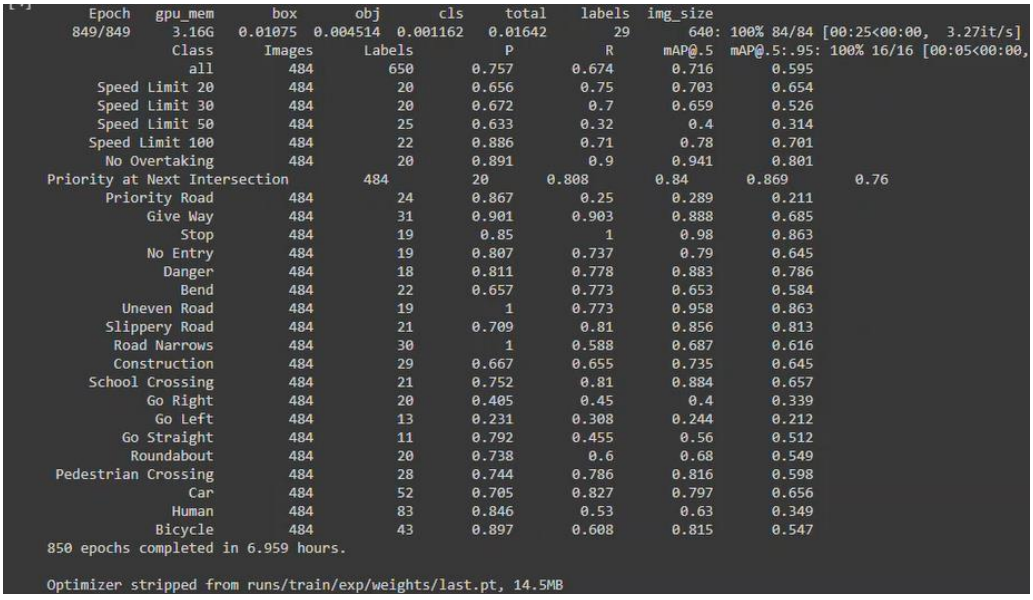
- cache : Görüntüleri önbelleğe alınması eğitim hızını arttırmaktadır.

Verisetinde araç dışı tespit 850 epoch için ortalama eğitim süresi yaklaşık 7 saat sürerken (Şekil 3.19.), bu durum araç içinde 700 epoch için yaklaşık 8,5 saat sürmüştür. Eğitim ve testlerin net bir şekilde analiz edilebilmesi için Tensorboard grafik eklentisi çalışmada kullanılarak modellerin performansları görselleştirilmiştir.



Şekil 3.19. Epoch sayısı = 850 için devam eden eğitim işlemi

Şekil 3.20.'de 25 adet nesnenin belirtilen epoch sayısı ile eğitiminden sonraki başarımları verilmiştir. Bunun sonucunda modeller test için hazır hale getirilmiştir.



Şekil 3.20. last.pt adlı model dosyasının yaklaşık 7 saat sonunda elde edilmesi

3.5. Modellerin Eğitim ve Test Sonuçlarının Performans Değerlendirmesi

Google Colab'da eğitimleri tamamlanarak test edilen modeller TensorBoard ile grafiklere dönüştürülmüştür. TensorBoard, makine öğrenimi deneyi için gereken görselleştirme ve araçları sağlayan araçtır. Kayıp ve doğruluk gibi metrikleri izleme ve model grafiklerini oluşturmak için oldukça popüler bir şekilde kullanılmaktadır.

3.5.1. Performans Değerlendirme Ölçütleri

Derin öğrenmede kullanılan algoritmalarının değerlendirilmesi için çeşitli parametreler kullanılmaktadır. Bunlardan;

- Kesinlik (Precision): Doğru tahmin edilen pozitifler (TP) ile tüm pozitifler (TP ve FP) arasındaki orandır (Denklem 3.3.). Ağın tahmin doğruluğunun ölçüsüdür.

$$Kesinlik = \frac{TP}{TP + FP} \tag{3.3}$$

- Hassasiyet (Recall) : Doğru tahmin edilen pozitiflerin) ne kadar doğru olduğunu ölçen orandır. Denklem 3.4.'da verilen formülle hesaplanmaktadır.

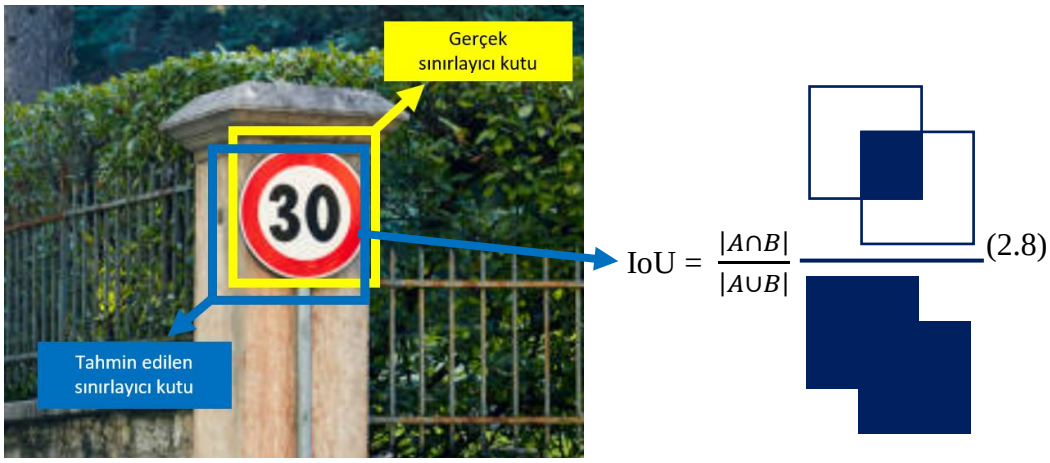
$$Hassasiyet = \frac{TP}{TP + FN} \tag{3.4}$$

- Ortalama Kesinlik (AP): Farklı güven eşikleri altında bir modelin genel performansını göstermektedir (Denklem 3.5). Denklemde verilen R_n ve P_n güven eşiğindeki kesinlik ve hassasiyeti ifade etmektedir. Her bir eşikteki kesinliklerin ağırlıklı ortalaması olarak bir kesinlik-geri çağırma eğrisini özetlemektedir.

$$\text{AP} = \sum_{n} (R_n - R_{n-1})\ P_n \tag{3.5}$$

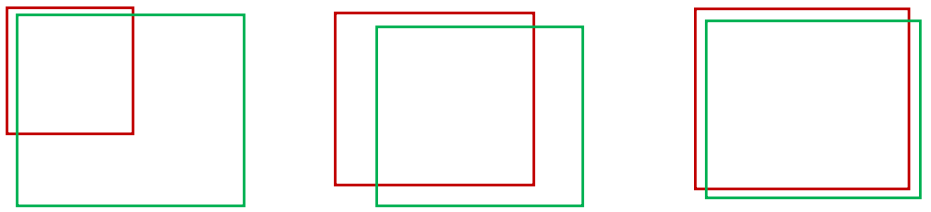
- Ortalama Kesinlik Ortalaması (mAP): mAP birimi ise ortalama kesinliğin ortalamasını ifade etmektedir. Bazı durumlarda her sınıf için hesaplanan AP'nin ortalamasıyken bazı durumlarda ise mAP ve AP aynı birimi ifade ederler.

- Birleşim Üzerinden Kesişim (IoU): Algılanan nesnenin gerçek nesneye ne kadar benzediğini belirleyen bir birimdir (Şekil 2.23.). Tahmin edilen sınırlayıcı kutu ile gerçek sınırlayıcı kutunun birleşimi üzerindeki kesişimdir (Denklem 2.8.).



Şekil 3.21. Öngörülen sınırlayıcı kutu ile gerçek sınırlayıcı kutu arasındaki ilişki

Kesişim sonucunda elde edilen alanın doğruluk yüzdesi Şekil 2.22.'de gösterildiği gibi yorumlanmaktadır. IoU'nun iki nesne arasındaki ilişkisi yüksek şekilde örtüşen sınırlayıcı kutuların düşük şekilde örtüşen sınırlayıcı kutulardan daha yüksek puanlanmasıyla hesaplanmaktadır.

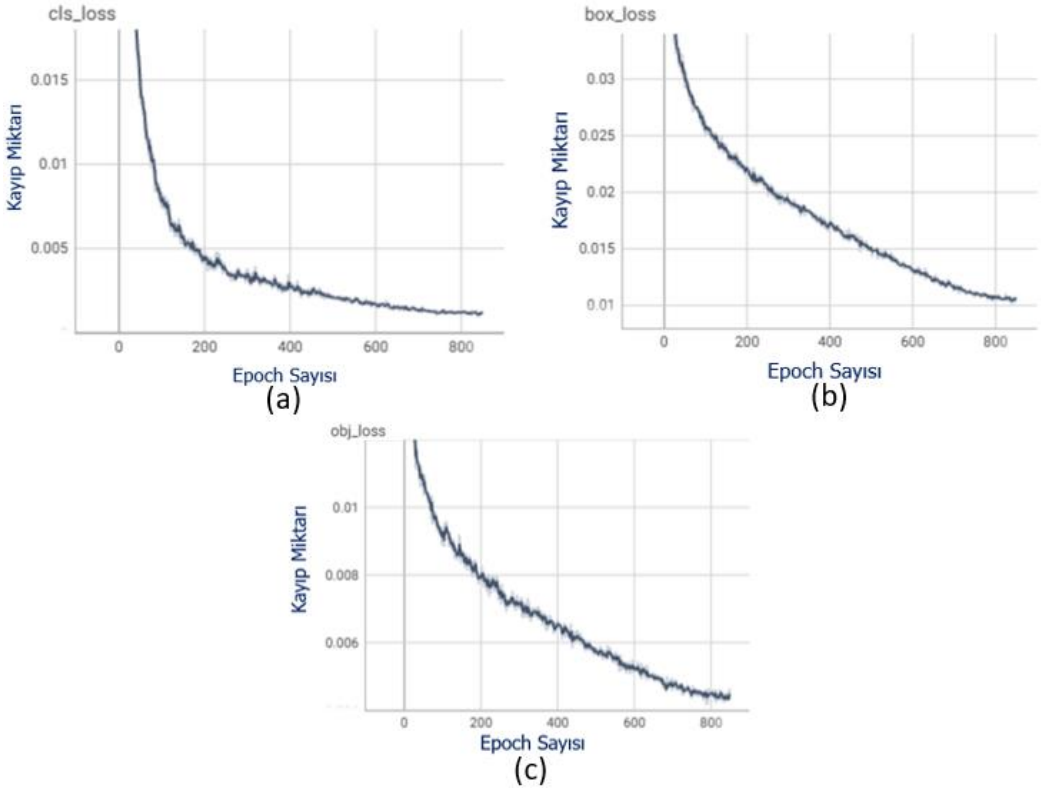


Şekil 3.22. Çeşitli sınırlayıcı kutular için Birleşimler Üzerinden Kesişimin (IoU) hesaplanması

3.5.2. Araç dışı model için performans analizleri

Araç dışı tespit işlemleri için modelin eğitim başarısının sonuçları 850 epoch için 7 saat sürdürülerek ortalama kayıp ve hata grafikleri hesaplanmıştır. Kayıp miktarlarının zamanla azalması ağın başarısı için önem taşımaktadır. Şekil 3.23.'te kolayca görüleceği gibi her üç parametre için de (sınıflandırma, kutu ve nesnellik kayıpları)

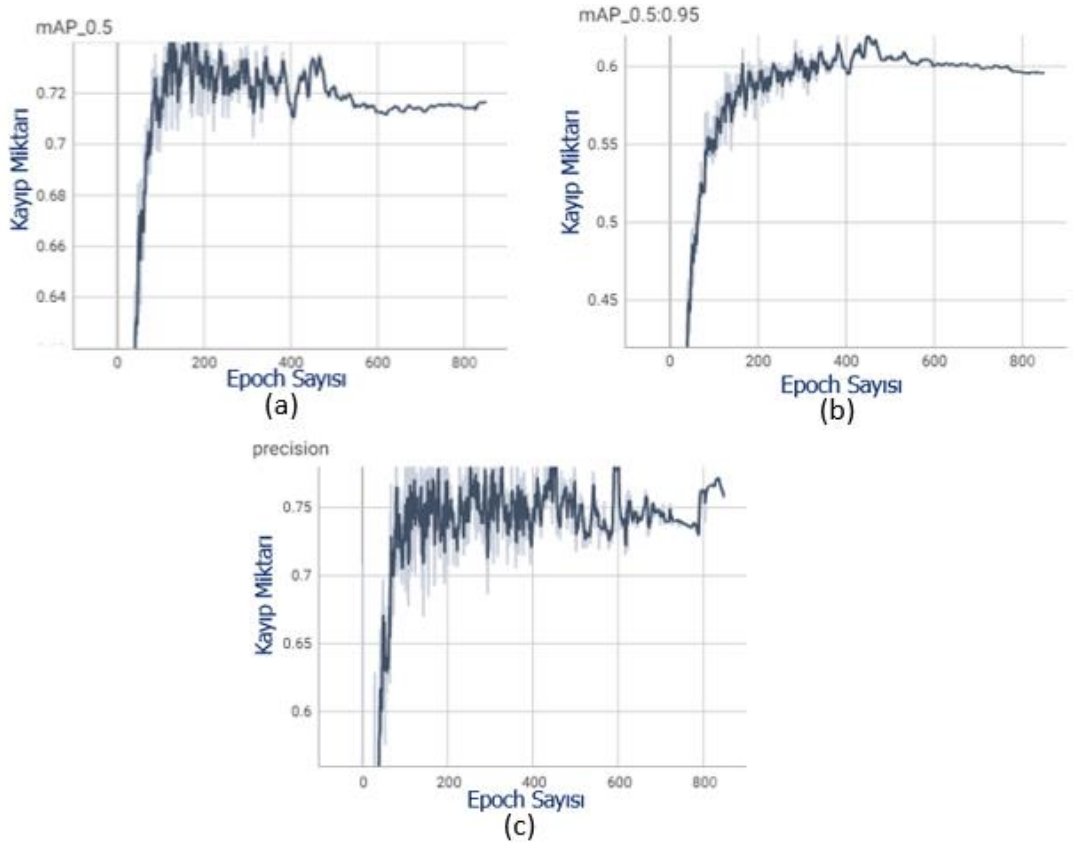
epoch sayısına bağlı olarak hata oranı önemli ölçüde düşürülmüştür. Böylelikle ağdan beklenilen tahmini kayıp oranı yaklaşık olarak 0,01 ila 0,005 ’in altına düşürülerek ağ performansını test için uygun hale getirmiştir.



Şekil 3.23. Modelin eğitim sonucu performansı (a) Sınıflandırma doğruluğunun kayıp grafiği (b) Sınırlandırıcı kutunun doğruluğu için oluşan kutu kaybı (c) Oluşan nesnellik kayıp grafiği

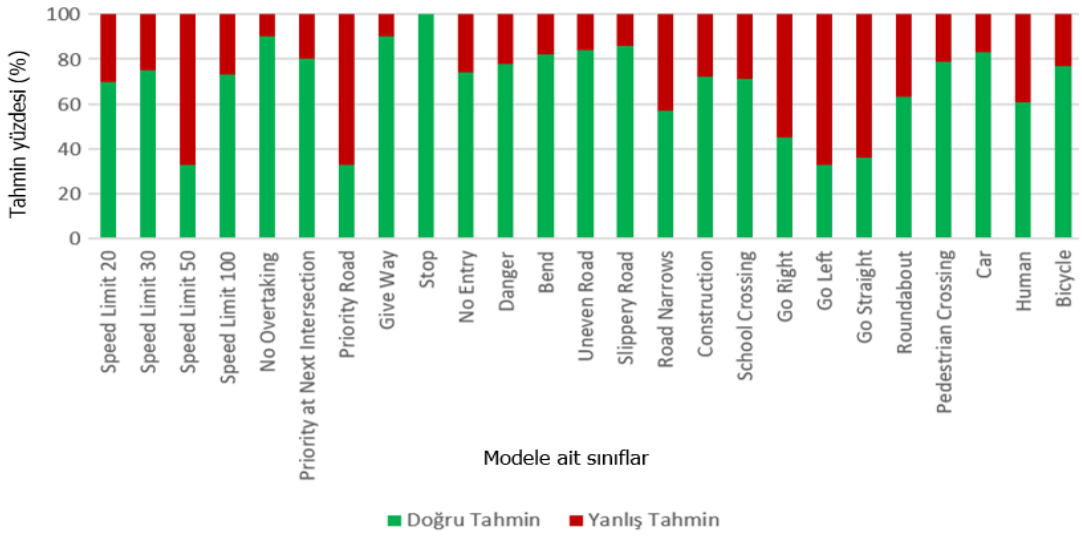
Modelin gerçek değerine en yakın değer olarak tahmin grafiği ise epoch sayısının artmasıyla Şekil 3.24.’de görüldüğü gibi bir artış göstermiştir. %80’in üzerine çıkarılarak doğruluğu korunmuştur. Grafikteki ani artış ve inişler muhtemelen kullanılan sınıf sayısının veriseti ile orantısı açısından sayıca farklılıktan kaynaklanmaktadır. Tüm bunlara ek olarak şekilden şu noktalar da gözlemlenebilir :

- Modelin eğitimi için verisetine ait olan 25 adet sınıf ağın performansı açısından daha fazla görüntüye ihtiyaç duymaktadır.
- Eğitimdeki epoch sayısının arttırılması her zaman için algılama doğruluğunun artacağı anlamına gelmez.
- Az sayıda epoch için algılama doğruluğu yüksek görünse de Şekil 3.23.’te görüleceği üzere kayıp miktarı epoch sayısıyla doğru orantılı düşmektedir.



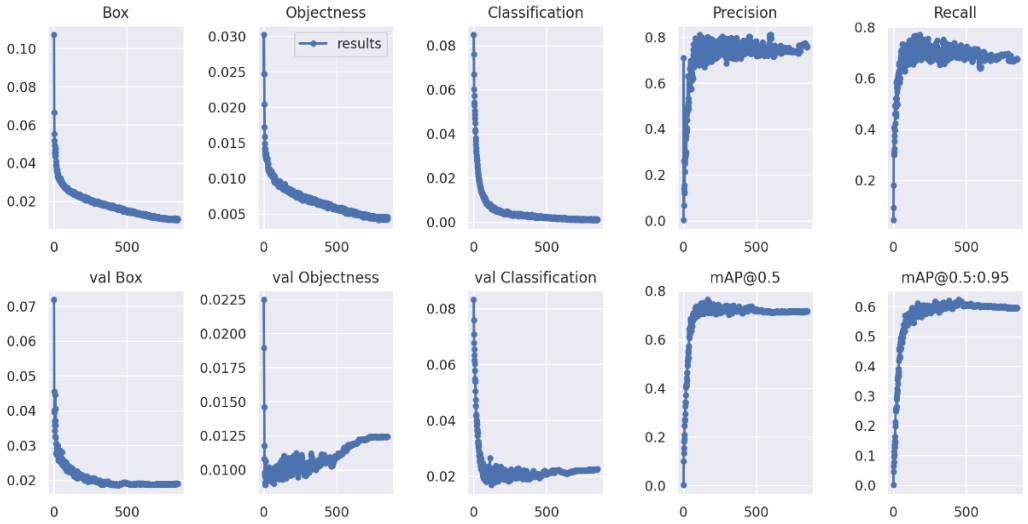
Şekil 3.24. Modelin eğitimi için (a) IoU değeri 0,5 için oluşan ortalama kesinlik grafiği (b)IoU değeri 0,5 ila 0,95 arasındaki adımlarda oluşan ortalama kesinlik grafiği (c) Total kesinlik grafiği

Trafik işareti ve nesneleri tespit etmek için toplanılan veriler ile model eğitimi gerçekleştirilmiştir. Şekil 3.25.'de bu durum her sınıf için algılama doğruluğunu özetleyecek şekilde gösterilmiştir. Her sınıfa ait çeşitli sayıda görüntü içeren veriseti doğruluk oranlarını farklılaştırmıştır. Bunun sebebi verisetindeki verilerin farklı dağılımından kaynaklanmaktadır.



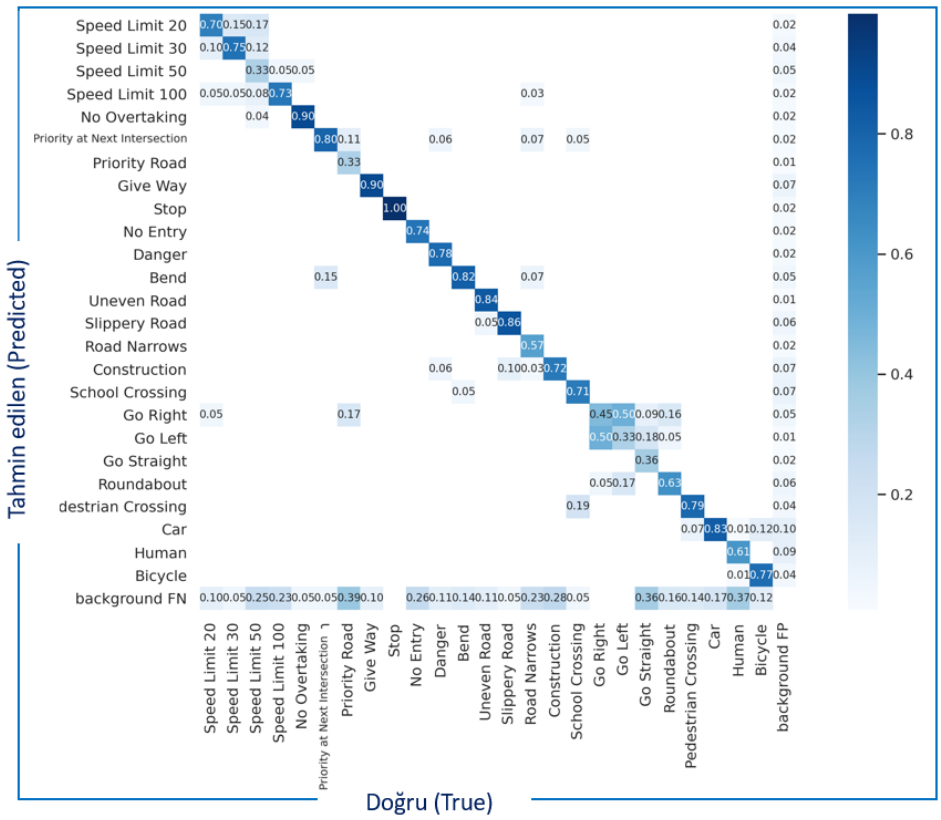
Şekil 3.25. Modelin test sonucuna ilişkin tahmin doğruluğu grafiği

Tamamlanan eğitim için doğrulama metrikleri Şekil 3.26.'da gösterildiği gibidir. Eğitim modelinin performans analizlerinin tüm ölçümleri burada değerlendirilmiştir.



Şekil 3.26. Araç dışı nesne algılama için eğitim grafikleri

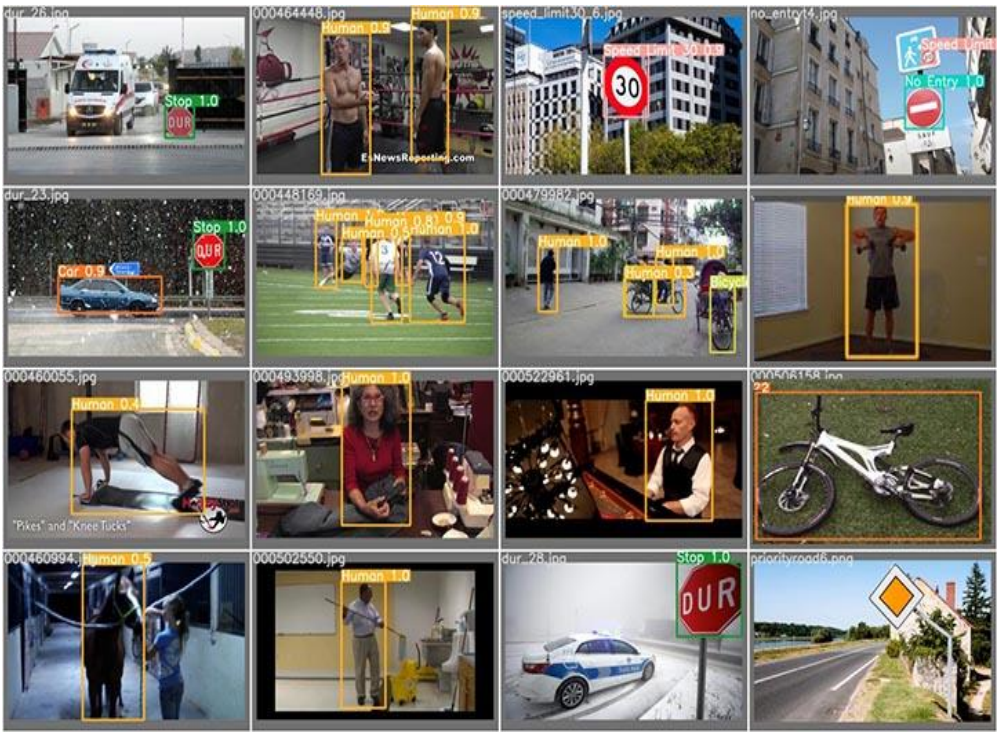
Tüm tespit edilen sınıfların tahmin edilen değerlerinin doğruluğunu gösteren karşılaştırmalı karışıklık (confusion) matrisi Şekil 3.27.'de gösterilmiştir.



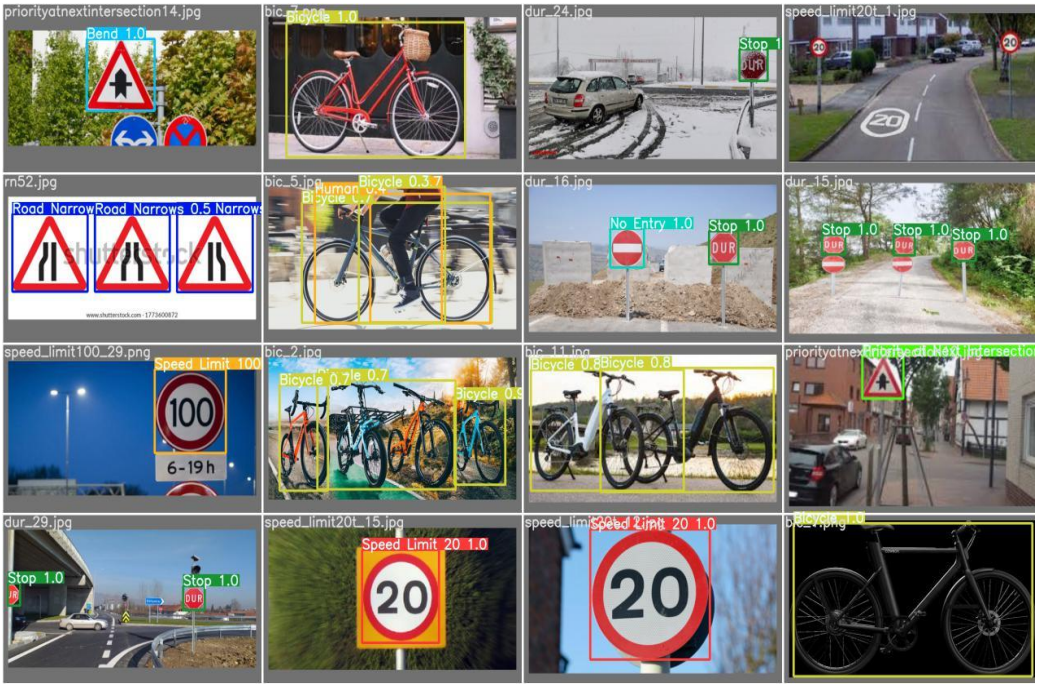
Şekil 3.27. Araç dışı nesne algılama görevinde karşılaştırmalı karışıklık matrisi

Araç dışı için oluşturulan model Şekil 3.28. ve Şekil 3.29.'a ilaveten Şekil 3.30.'da gösterildiği gibi tahmin edilen nesne konumları ve sınıfa aitlik olasılığı verilmiştir.

Şekil 3.28. Modelin başarımının test sonuçları -1

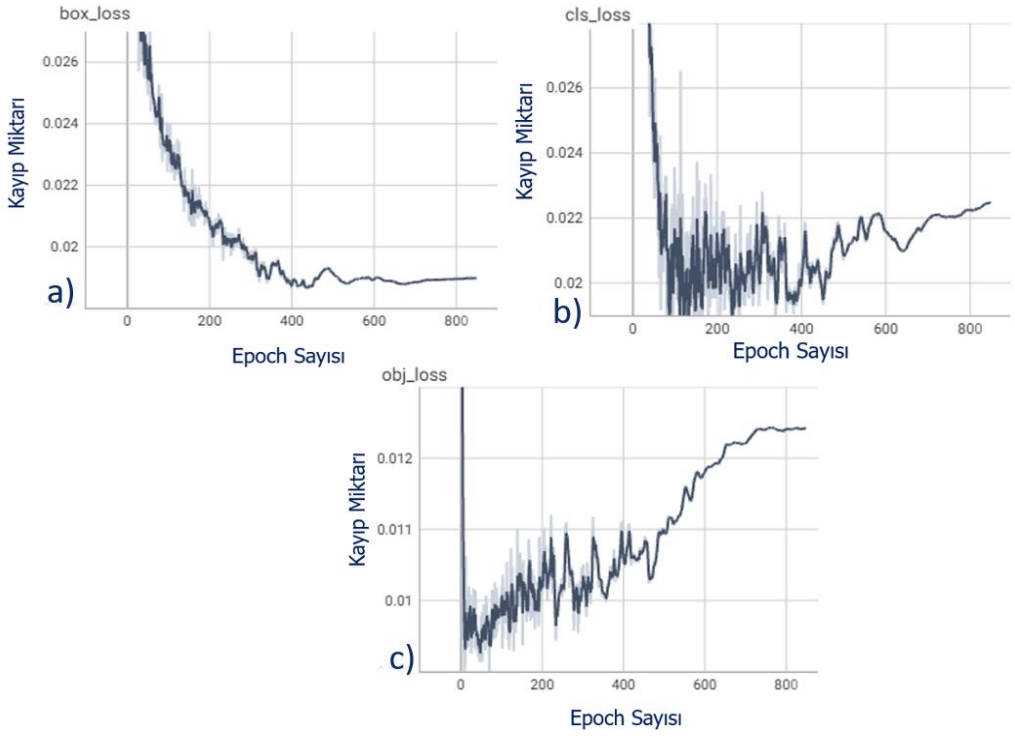


Şekil 3.29. Modelin başarımının test sonuçları-2



Şekil 3.30. Modelin başarımının test sonuçları-3

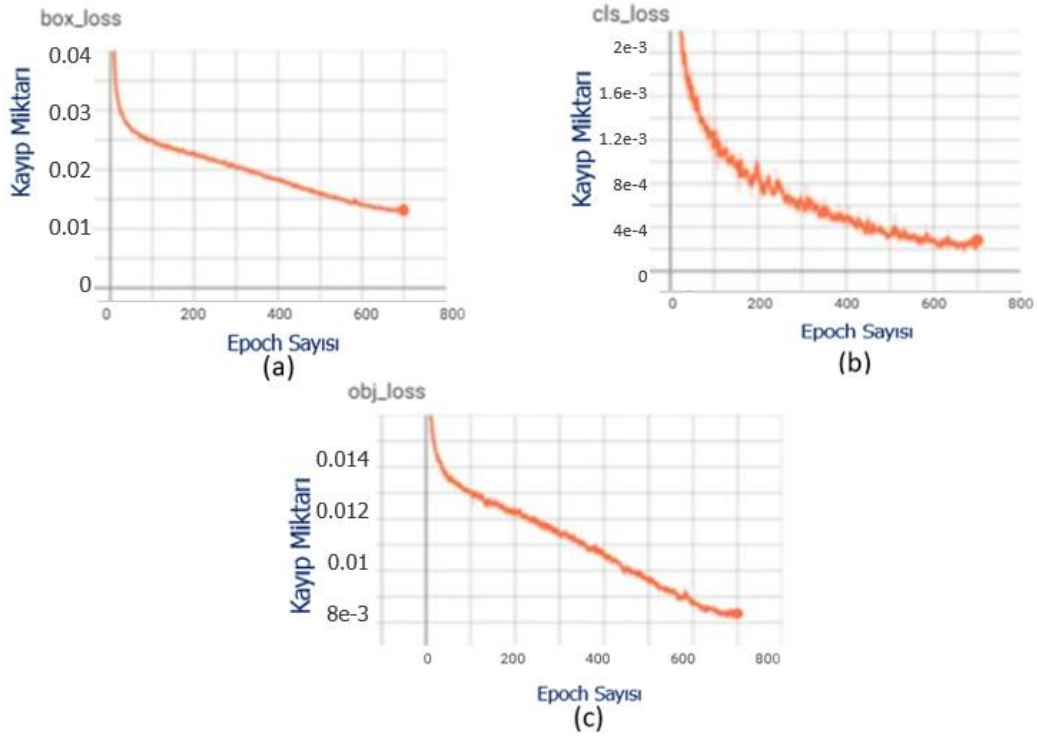
Yapılan eğitimlerin test sonucu performansı Şekil 3.31.'de gösterilmiştir. Şekilden de kolaylıkla görüleceği gibi, sınıflandırma doğruluğunun kaybı, kutu kaybı ve nesnellik kaybı grafikleri yaklaşık 500 epoch'a kadar doğru orantılı olarak hata değeri azalırken öğrenme oranı zamanla arttığı için ağ modeli muhtemelen aşırı öğrenmeye gidilmiştir.



Şekil 3.31. Modelin test sonucu performansı (a) Sınıflandırma doğruluğunun kayıp grafiği (b) Sınırlandırıcı kutunun doğruluğu için oluşan kutu kaybı (c) Oluşan nesnellik kayıp grafiği

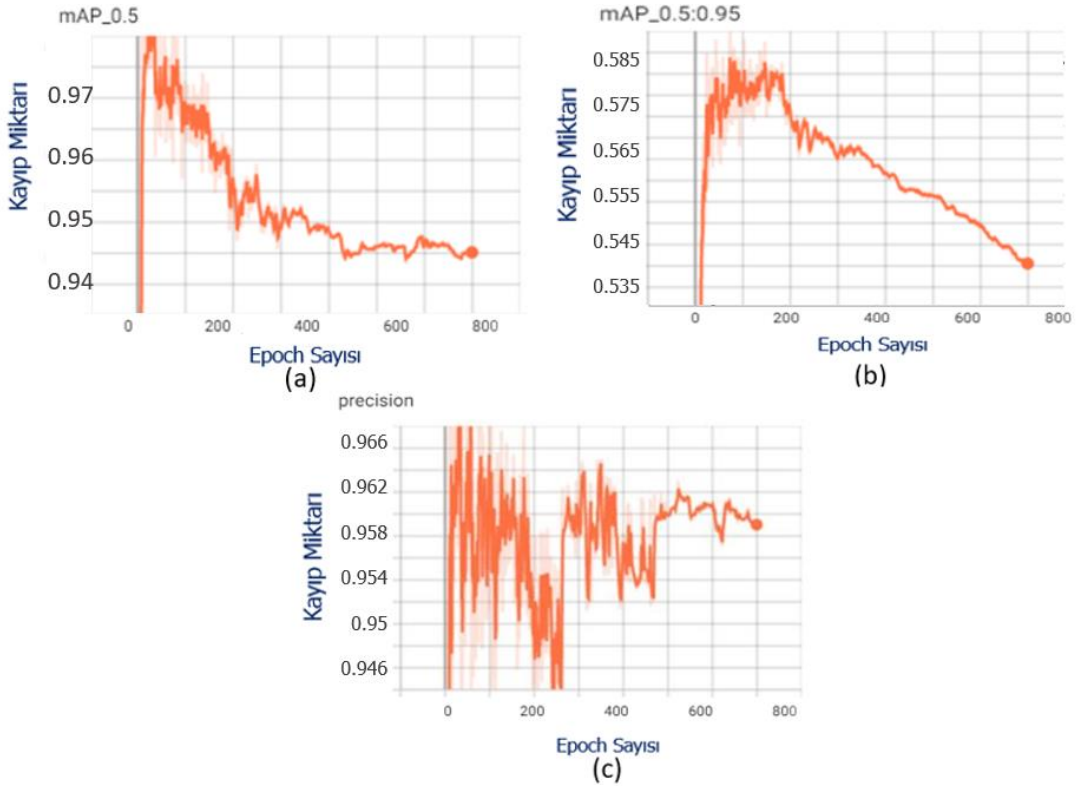
3.5.3. Araç içi model için performans analizleri

Araç içi tespit işlemleri için modelin eğitim başarısının sonuçları 700 epoch için 8,5 saat sürdürülerek ortalama kayıp ve hata grafikleri hesaplanmıştır (Şekil 3.32.). Modellerin nesne algılamadaki performansının ölçülmesi için tahmin edilen sınırlayıcı kutu ile etiketleme aşamasında konumlandırılan sınırlandırıcı kutunun koordinatları ve ait olunan sınıfın olasılığı gibi parametreler baz alınmaktadır. YOLO v5 modeli, 700 epoch'a kadar Şekilden de kolayca görülebileceği gibi gerçekten iyi bir performans sergilemiştir. Daha sonra dönemlerin artmasıyla sınıflandırma kaybı, kutu kaybı ve nesnellik kaybı gibi tüm kayıplar artarak model performansı azalmıştır.

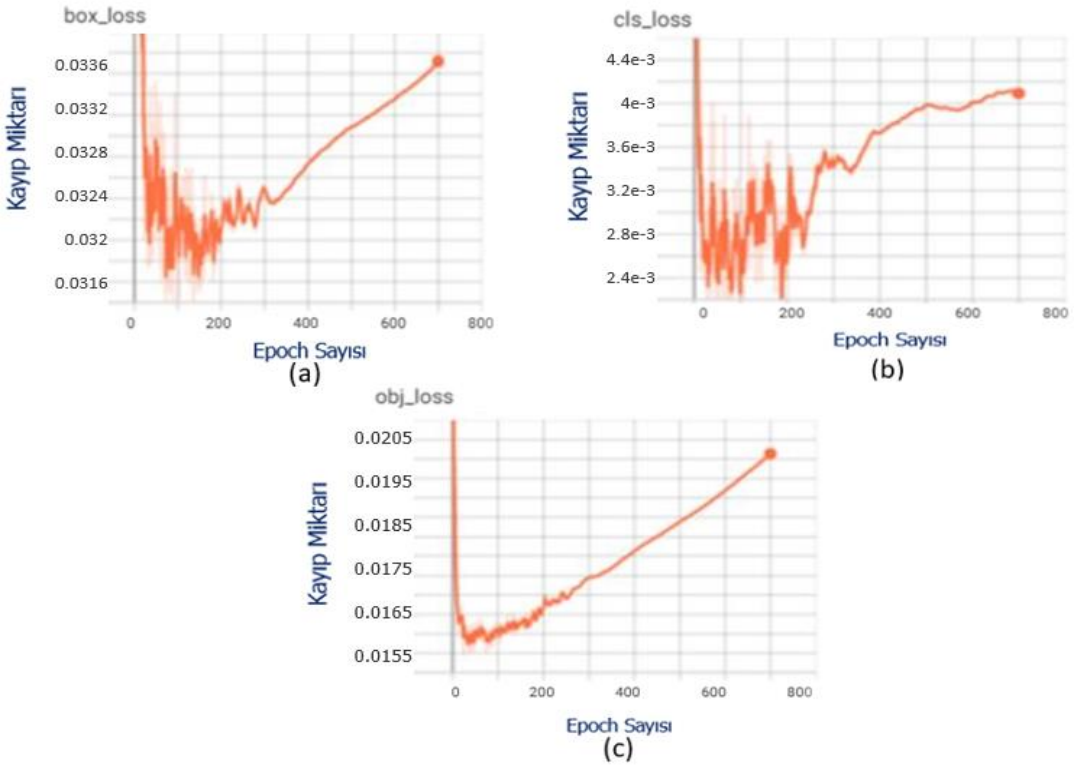


Şekil 3.32. Sınıflandırma doğruluğunun kayıp grafiği (b) Sınırlandırıcı kutunun doğruluğu için oluşan kutu kaybı (c) Oluşan nesnellik kayıp grafiği

Modeli eğitmek için çeşitli görüntü çözünürlükleri kullanılmış ve uygun görüntü çözünürlüğü seçilmiştir. Test için ayrılan veriler üzerinde tespit edilen nesnelerin kesinlik, geri çağırma, ortalama kesinlik (AP, mAP) hesaplanmıştır. Buna ek olarak test sonucu performansında oluşan kayıplar da diğer modellerle karşılaştırılmıştır. Metrikler Bölüm 3.6'da verilen denklemler yardımıyla hesaplanarak sonuçlar benzer şekilde Şekil 3.33. ve Şekil 3.34.'de gösterilmiştir.



Şekil 3.33. Modelin eğitim performansına ilişkin oluşan (a) IoU değeri 0,5 için oluşan ortalama kesinlik (b)IoU değeri 0,5 ila 0,95 arasındaki adımlarda oluşan ortalama kesinlik (c) Kesinlik grafikleri



Şekil 3.34. Modelin test sonucu performansı (a) Sınıflandırma doğruluğunun kayıp grafiği (b) Sınırlandırıcı kutunun doğruluğu için oluşan kutu kaybı (c) Oluşan nesnellik kayıp grafiği

ADAS sistemi veriseti oluşturulduktan sonra çeşitli sürücü aktiviteleri üzerinde durum tespiti için test edilmiştir. Şekil 3.35.'de farklı nesne ve durumlar üzerinde test edilen sistemden örnekler verilmiştir. Buna göre modelin verilen bu test görüntüleri üzerinde yüksek bir kesinlik ile tespit başarısı elde ettiği görülmektedir.

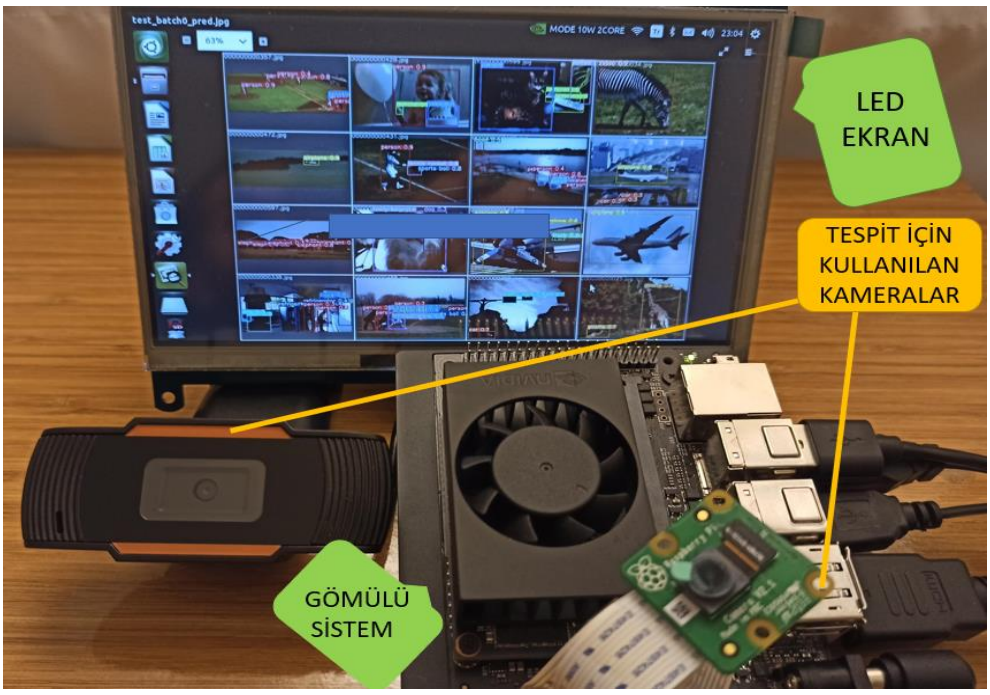
Şekil 3.35. Araç içi modelin çeşitli test sonuçlarından görüntüler

BÖLÜM 4. GELİŞTİRİLEN SÜRÜCÜ ASİSTAN SİSTEMİNİN MOBİL PLATFORM ÜZERİNDE TEST EDİLMESİ

Tezin bu bölümünde, nesne ve davranışları tespit için oluşturulan modellerin üç farklı cihaz üzerinde algılama hızları karşılaştırılmıştır. Sistemin gerçek zamanlı çalışması istenildiğinden uygulamalar yüksek performanslı GPU ihtiyacından dolayı iki mobil platform ve bir de test bilgisayarına yüklenerek analiz edilmiştir. Araç içerisinde de başarıyla çalışan uygulama nesneleri başarılı bir şekilde tahmin edebilmiştir.

4.1. Uygulama İçin Mobil Platformların Hazırlanması

Sürücüye yardımcı olmayı ve kaza riskini azaltmayı hedefleyen ADAS sistemlerinin temel sorunlarından biri düşük güç tüketimi ve gerçek zamanlı tanıma için yeterli hesaplama gücüne sahip uygun bir donanım bulmaktır. Bu tez çalışmasında araç içi ortamda kullanılabilecek gömülü sistem, ekran ve kameralara sahip nesne tespiti algoritmalarının koşturulabildiği bir elektronik sistem geliştirilmiştir (Şekil 4.1.).

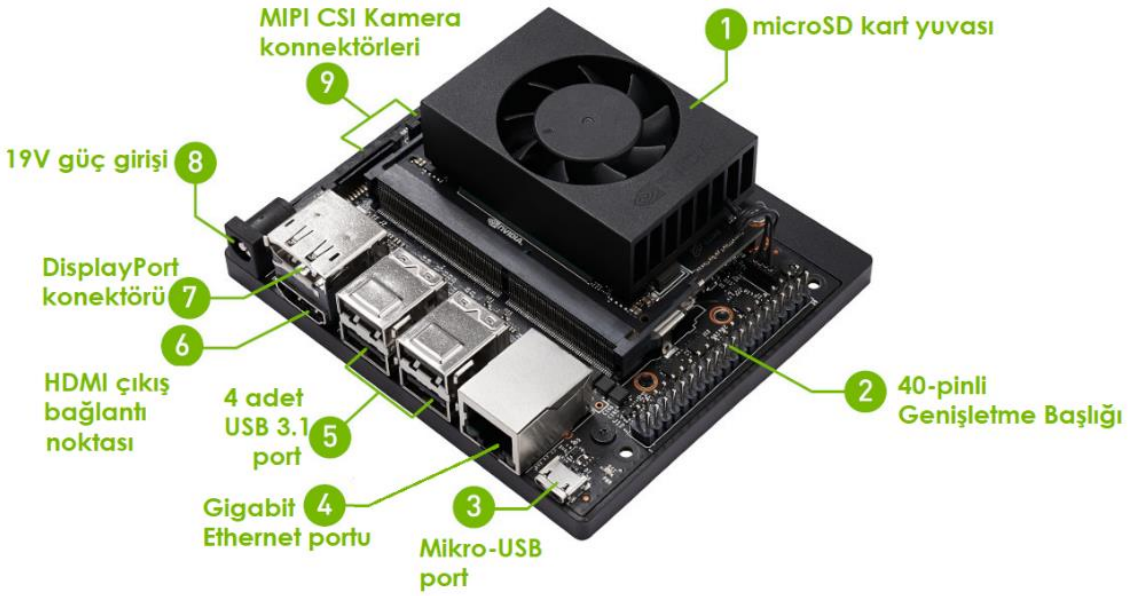


Şekil 4.1. Nvidia Xavier Nx geliştirme kiti ve uygulama için hazırlanan platform

Gömülü platform üzerinde sınırlı kaynaklar ile gerçek zamanlı çalışan bir sistemin hafif bir model boyutuna sahip olması gerekmektedir. Sınırlı kaynaklara sahip bir gömülü sistem için çalıştırılacak olan uygulamanın bellek boyutu ve hesaplama gücü oldukça önemlidir. Tüm bu kısıtlar dikkate alınarak geliştirilen model mobil platformlarda gerekli konfigürasyonlardan sonra çalıştırılabilmektedir. Bu tez çalışmasının deney kısmında kullanılacak cihazların teknik özellikleri şu şekildedir;

4.1.1. Nvidia jetson xavier nx

Deney çalışmasında kullanılan Nvidia Xavier NX geliştirme kiti GPU tabanlı uygulamalar için tasarlanmış enerji ve hesaplama gücü açısından verimli çok çekirdekli bir gömülü karttır. Nvidia Xavier Nx'e ait genel görünümü Şekil 4.2.'de gösterildiği gibidir.



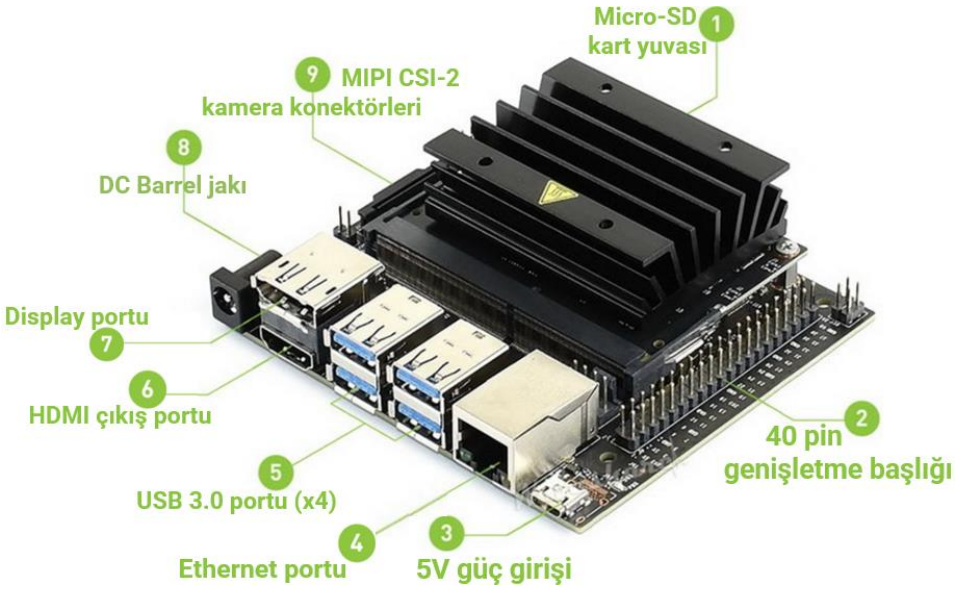
Şekil 4.2. Nvidia Xavier Nx özellikleri

Volta GPU mikro mimarisi ile güçlü bir grafik işlemciye sahip olan cihazın grafik işlemcisi saniyede 30 FPS hızında 1080p video akışına çıkabilmektedir [85]. Karta tümleşik olarak yerleştirilen iki adet Derin Öğrenme Hızlandırıcısı (DLA) sayesinde bilgisayarlı görme ve derin öğrenme uygulamaları için ideal olup Linux işletim sisteminde çalışabilmektedir. Nvidia Xavier NX geliştirme kitine ait teknik özellikler [86] aşağıdaki gibidir;

- 384 Nvidia CUDA ve 48 Tensor çekirdekli Nvidia Volta mimarisi,
- 6 çekirdekli NVIDIA Carmel ARM v8.2 64-bit CPU 6 MB L2,
- 2x NVDLA (Derin Öğrenme Hızlandırıcı),
- HDMI port desteği, 40 pinli GPIO,
- 4 adet USB 3.1, USB 2.0 Mikro-B ve 2 adet CSI-2 bağlantı noktası,
- 802.11ac Wi-Fi, Gigabite Ethernet desteği,
- 103 mm x 90,5 mm x 31 mm boyutlarındadır.

4.1.2. Nvidia jetson nano

NVIDIA Jetson Nano son teknoloji yapay zeka algoritmaları için yüksek hesaplama kabiliyetine sahip ve güç tüketimi açısından verimli gömülü platformdur. Nesne tespiti, görüntü sınıflandırma, segmentasyon ve dil işleme gibi uygulamalarda kullanılmaktadır. Jetson Nano için genel görünüm Şekil 4.3.'te gösterildiği gibidir.



Şekil 4.3. Nvidia Jetson Nano özellikleri

Ayrıca Jetson Nano üzerinde bulunan pinler sayesinde GPIO ve CSI işlemlerini mümkün kılmaktadır. Çeşitli sensörler bağlanarak haberleşme sağlabilir ve üzerinde bulunan wifi modülü sayesinde cihaz uzaktan kontrol edilebilmektedir. Bunlara ek olarak Nano Linux İşletim Sistemi'ne sahiptir. NVIDIA CUDA, cuDNN ve

TensorRT yazılım kütüphanelerini içeren NVIDIA JetPack paketi de cihazın hızlı ve kolay kullanımı desteklenmektedir. Teknik özellikleri ise şu şekildedir;

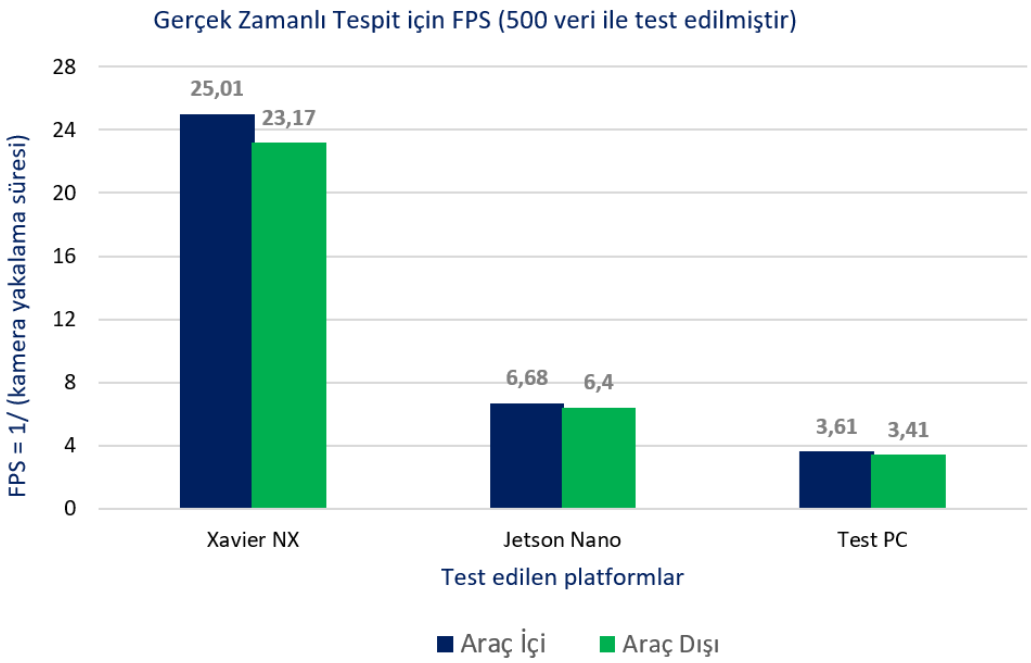
- 128-core NVIDIA Maxwell™ GPU,
- 64-bit Quad-core ARM A57 (1.43 GHz) CPU,
- 2 GB 64-bit LPDDR4 (25.6 GB/s bandwidth) hafıza birimi,
- 2x NVDLA (Derin Öğrenme Hızlandırıcı),
- 1x MIPI CSI-2 kamera konnektörü,
- 12 pinli UART başlığı ile 4 pinli fan başlığı,
- HDMI çıkış portu ve USB-C 5V 3A güç desteği,
- 100 mm x 80 mm x 29 mm boyutlarındadır.

Çalışmanın algılama hızını test edebilmek için oluşturulan modeller Jetson Xavier Nx, Jetson Nano ve test bilgisayarı üzerinde sırasıyla çalıştırılarak test edilmiştir. Tablo 4.1.'de kullanılan cihazların özellikleri gösterilmiştir.

Tablo 4.1. Yapılan çalışmadaki test ortamlarının karşılaştırılması

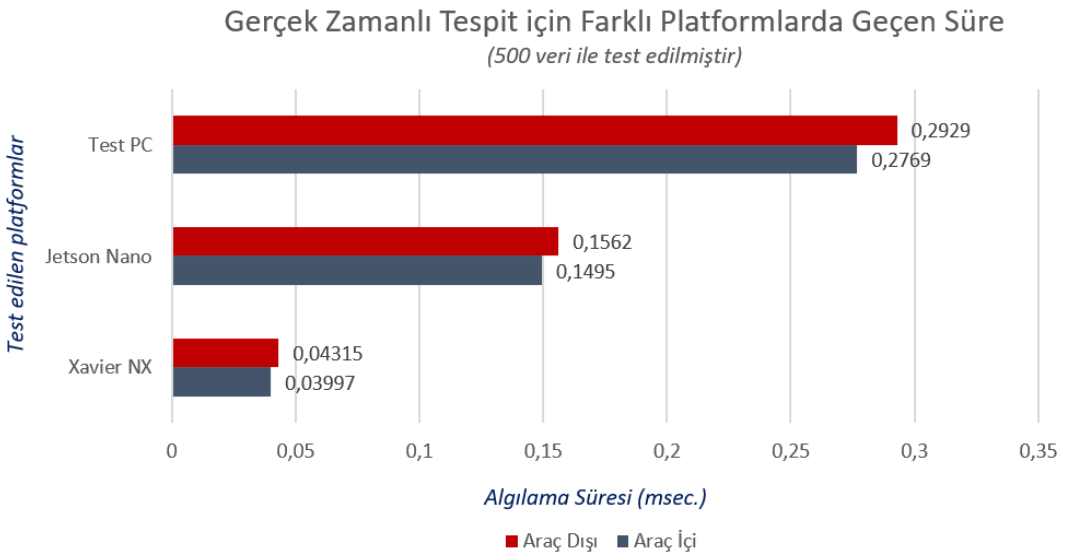
	Kişisel Bilgisayar	**Nvidia Xavier Nx**	**Jetson Nano**
CPU	Intel Core i7 6700 HQ	6-core Carmel ARM v8.2	Quad-core ARM A57
RAM	8 GB DDR4	8 GB	4 GB
GPU	Nvidia Geforce GTX 960 M 4 GB	384 Core Volta	128 Core Maxwell
Ağırlık	2700 gr.	771 gr.	140 gr.

Şekil 4.4.'de Jetson Xavier ve Jetson Nano gömülü platformlarına ek olarak kişisel bilgisayar üzerinde tespit işlemi için geçen sürelerin FPS cinsinden kıyaslaması verilmiştir. Şekilden görüleceği üzere Jetson Xavier Nx; Jetson Nano ve test bilgisayarından çok daha yüksek bir performans göstermiştir. Çalışmada Jetson Xavier Nx üzerinde derin öğrenme işlemlerinin yürütülerek gerçek zamanlı sonuçların alınması maksimum verimlilik ve performansa katkı sağlamıştır. Bununla birlikte gömülü platforma uygunluğunun görülmesi için eğitilen model NVIDIA Jetson Xavier Nx ve NVIDIA Jetson Nano üzerinde çalıştırılarak hızların farkı net bir şekilde görülmektedir.



Şekil 4.4. ADAS uygulamalarının gerçek zamanlı tespit hızlarının FPS cinsinden kıyaslanması

Çalışmanın tespit aşamasında da kolayca görülebileceği gibi son teknoloji derin öğrenme modellerinin NVIDIA Xavier Nx mobil donanım platformu üzerinde çalıştırılması hız ve taşınabilirliğe katkı sağlamıştır. Şekil 4.5.'te ise kullanılan platformların gerçek zamanlı olarak karşılaştırılması için geçen nesne tespit süreleri görülmektedir.



Şekil 4.5. ADAS uygulamalarının gerçek zamanlı tespit hızlarının karşılaştırması

4.2. Sistemin Araç İçinde Test Edilmesi

Yapılan uygulamayı değerlendirmek için araç içerisinde sürücüyü ve araç dışında sürüş ortamını gösteren ve en yüksek FPS hızının elde edildiği Nvidia Xavier cihazı üzerine iki adet kamera yerleştirilmiştir. Araçtaki algılama ekipmanları Şekil 4.6.'da gösterilmiştir. Eğitim sırasında kullanılan gömülü platform içerisine Jetpack Ubuntu 18.04 işletim sistemi kurulmuştur. Modelin çalıştırılması için PyTorch kullanılmıştır. Python dili üzerinde Jetson sürümü 4.4, CUDA 10.2, CUDNN 8.0 gibi çeşitli sürümlerde kütüphaneler yüklenerek başarılı bir şekilde sistem test edilmiştir. Gerçekleştirilen uygulama yine Şekil 4.6.'da verilmiştir. Araç içindeki kameralar vasıtasıyla sürücünün telefon, sigara ve uyku durumları tespit edilirken, araç dışı algılamada dış kamera ile trafik işaretleri, insan, araç ve nesne tespitleri yapılmıştır.

Şekil 4.6. Araç içerisinde test ekipmanlarının görüntüleri

BÖLÜM 5. SONUÇLAR VE ÖNERİLER

Trafik risklerini azaltmak ve sürüş deneyimini arttırmak amacıyla araçlarda ADAS sistemlerinin kullanımı hızla yaygınlaşmaktadır. Bu sistemler sürücü ve araç çevresinden çeşitli sensörler ile veriler elde ederek kritik durumları kontrol altına almayı hedefler. ADAS sistemleri içerisinde görüntü işlemeye dayalı uygulamalar ise son yıllarda oldukça popüler hale gelmiştir. Bu tez çalışmasında ADAS sistemlerine katkıda bulunmak amacıyla sürücü ve yol çevresi temel alınarak iki farklı uygulama geliştirilmiştir. Bu uygulamalar araç içi ve araç dışı kameralardan gelen görüntüleri derin öğrenme yöntemiyle gerçek zamanlı işleyerek tespit sağlamaktadır. İlk olarak araç içi kısımda sürücünün uyku / yorgunluk, cep telefonu kullanımı ve sigara kullanımı tespit edilmiştir. Araç dışı kısımda ise geliştirilen uygulama sürüş çevresindeki belirli nesneler ile yirmiden fazla trafik işaretinin tespitini gerçekleştirmektedir. Her iki uygulama da gömülü platform üzerinde çalıştırılarak kritik durumları gerçek zamanlı tespit ederek sonuçları ekranda gösteren bir ADAS prototipi elde edilmiştir.

Tespit işlemi için algılama hızı ve doğruluk açısından başarımı yüksek olan YOLO v5 algoritması kullanılmıştır. Bu algoritma kameradan gelen görüntüleri girdi olarak alarak katmanlar halinde görüntüleri işler ve çıktıları tahmin etmeye çalışır. Sistemin geliştirilmesi için GTSRB veriseti ile birlikte uygulamaya özgü veriseti elde edilerek etiketlemeler yapılmıştır. Ardından, veriler eğitim ve test işlemlerine hazır hale getirilerek eğitilmiştir. Çalışmada modelin eğitim işlemleri herhangi bir konfigürasyon gerektirmeden ücretsiz GPU ve TPU kullanımına izin vererek derin öğrenme uygulamaları geliştirmeye yardımcı olan Google Colab platformu üzerinde gerçekleşmiştir. Çalışma Colab'ın sunduğu Tesla P100 grafik kartı ile her iki uygulama için yaklaşık olarak toplamda 17 saat süren bir eğitim ile araç içi ve araç dışındaki tespitler için modeller oluşturulmuştur.

Eğitim işleminin sonucunda oluşan modeller önceden gerekli konfigürasyonlar yapılmış olan iki farklı gömülü platform (NVIDIA Jetson Nano, Jetson Xavier Nx) ve bir test bilgisayarına yüklenerek sistemin gerçek zamanlı çalışması analiz edilmiştir. Bu doğrultuda düşük güç ve yüksek hesaplama gücüne sahip NVIDIA Jetson Xavier Nx platformu tespit için en hızlı çıkarım hızına ulaşmasıyla araç içerisinde kullanımı tercih edilmiştir. Gömülü platform kullanımı uygulamanın mobilitesini arttırarak maliyeti düşürmüş ve çalışmayı kolayca monte / entegre edilebilir bir ürün haline getirmiştir.

Gelecek çalışmalarda tez çalışmasında elde edilen nesne tespiti ve gömülü sistem kullanım becerisi geliştirilerek bir takım yenilikler eklenebilir. Bunlar içerisinde son teknoloji derin öğrenme modelleri ile çalışılması ve kullanılan verisetlerinin farklı aydınlatma ve ortam koşullarına uygun olarak geliştirilmesi; ADAS sisteminin nesneleri daha iyi bir şekilde tespit etmesine katkı sağlayacaktır. Bu çalışmada nesne tespitine odaklanılmıştır. Ancak trafikteki uygulamalar düşünüldüğünde bu sisteme şerit ihlali de eklenebilir. Bunun yanında, gerçekleştirilen sistem araç içi ve araç dışında ayrı ayrı çalışmaktadır. Ancak bunu tek bir işlemcide eş-zamanlı koşturabilecek bir sistem tasarlanabilir. Böyle bir sistem için hesaplama gücü ve tespit süresinin dikkate alınması gereklidir. Bunlardan başka, taşınabilir ADAS sistemlerine katkı sağlamak amacıyla otonom sürüş için çeşitli sensörlerle durum tespiti yapılabilir.

KAYNAKLAR

[1] Dabral, S., Kamath, S., Appia, V., Mody, M., Zhang, B. and Batur, U., "Trends in camera based Automotive Driver Assistance Systems (ADAS)," *Midwest Symp. Circuits Syst.*, 1110–1115, 2014.

[2] Kakani, V. H. Kim, M. Kumbham, D. Park, C. Bin Jin, and V. H. Nguyen, "Feasible self-calibration of larger field-of-view (FOW) camera sensors for the advanced driver-assistance system," *Sensors (Switzerland)*, 19(15), 1–29, 2019.

[3] Zhong, Z., Liu, Z., Mathew, M. and Dubey, A., "Camera radar fusion for increased reliability in ADAS applications," *IS T Int. Symp. Electron. Imaging Sci. Technol.*, 1–4, 2018.

[4] Jang, B.J., Jung, I. T., Soo Chong, K., Sung, H.K. and Lim, S. "Rainfall Observation System using Automotive Radar Sensor for ADAS based Driving Safety," *2019 IEEE Intell. Transp. Syst. Conf. ITSC 2019*, 305–310, 2019.

[5] Zolock, J., Senatore, C., Yee, R., Larson, R, and Curry, B. "The Use of Stationary Object Radar Sensor Data from Advanced Driver Assistance Systems (ADAS) in Accident Reconstruction," *SAE Tech. Pap.*, 2016.

[6] Li, Y., Wang, Y., Deng, W., Li, X., Liu, Z. and Jiang, L., "LiDAR Sensor Modeling for ADAS Applications under a Virtual Driving Environment," *SAE Tech. Pap.*, vol. 2016-Septe, no. September, 2016.

[7] Goodin, C., Carruth, D., Doude, M. and Hudson, C., "Predicting the influence of rain on LIDAR in ADAS," *Electron.*, 8(1), 2019.

[8] Warren, M. E., "Automotive LIDAR Technology C254 C255," *2019 Symp. VLSI Circuits*, 254–255, 2019.

[9] Liu, Z., Qi, M., Shen, C., Fang, Y. and Zhao, X., "Cascade saccade machine learning network with hierarchical classes for traffic sign detection," 2021.

[10] Li, H., Sun, F., Liu, L. and Wang, L., "Neurocomputing A novel traffic sign detection method via color segmentation and robust shape matching," *Neurocomputing*, 77–88, 2015.

[11] Yin, S., Ouyang, P., Liu, L., Guo, Y. and Wei, S., “Fast Traffic Sign Recognition with a Rotation Invariant Binary Pattern Based Feature,” 2161–2180, 2015.

[12] Stallkamp, J., Schlipsing, M., Salmen, J. and Igel, C., “The German Traffic Sign Recognition Benchmark: A multi-class classification competition,” *Proc. Int. Jt. Conf. Neural Networks*, 1453–1460, 2011.

[13] Xu, X., Jin, J., Zhang, S., Zhang, L., Pu, S. and Chen, Z., “Smart data driven traffic sign detection method based on adaptive color threshold and shape symmetry,” *Futur. Gener. Comput. Syst.*, vol. 94, pp. 381–391, 2019.

[14] Qian, R., Zhang, B., Wang, Z. and Coenen, F., “Robust Chinese Traffic Sign Detection and Recognition with Deep Convolutional Neural Network,” 791–796, 2015.

[15] Changzhen, X., Cong, W., Weixin, M. and Yanmei, S., “A Traffic Sign Detection Algorithm Based on Deep Convolutional Neural Network,” 6–9, 2016.

[16] Zhang, J., Huang, M., Jin, X. and Li, X., “A Real-Time Chinese Traffic Sign Detection Algorithm Based on Modified YOLOv2,” 1–13, 2017.

[17] Ammour, N., Alhichri, H., Bazi, Y., Benjdira, B., Alajlan, N. and Zuair, M., “Deep Learning Approach for Car Detection in UAV Imagery,” 2017.

[18] Ćorović, A., Ilić, V., Đurić, S., Marijan, M. and Pavković, B., “The Real-Time Detection of Traffic Participants Using YOLO Algorithm,” 31–34, 2018.

[19] Jamtsho, Y., Riyamongkol, P. and Waranusast, R., “Real-time license plate detection for non-helmeted motorcyclist using YOLO,” *ICT Express*, 7(1), 104–109 , 2021.

[20] Kuo, W. J., Lin, C.C., "Two-stage road sign detection and recognition," Proceedings of the 2007 IEEE International Conference on Multimedia and Expo, Beijing, China, 1427–1430, 2007.

[21] Bengio, Y. and Haffner, P., “Gradient-Based Learning Applied to Document Recognition,” 86(11), 1998.

[22] Liu, C., Yin, F., Wang, D. and Wang, Q., “Chinese Handwriting Recognition Contest 2010,” 3–7, 2010.

[23] Yang, Y., Luo, H., Xu, H. and Wu, F. “Towards Real-Time Traffic Sign Detection and Classification,” *IEEE Trans. Intell. Transp. Syst.*, 17(7), 2016.

[24] Yu F. *et al.*, “BDD100K : A Diverse Driving Dataset for Heterogeneous Multitask Learning,” 2636-2645, 2020.

[25] Zou, Z. Shi, Z., Guo, Y., Ye, J. and Member S., “Object Detection in 20 Years : A Survey,” 1–39, 2019.

[26] Mathias, M. Timofte, R., Benenson, R. and Van Gool, L., “Traffic sign recognition - How far are we from the solution?,” *Proc. Int. Jt. Conf. Neural Networks*, 2013.

[27] Møgelmose, A., Trivedi, M.M. and Moeslund, T.B., “Vision-Based Traffic Sign Detection and Analysis for Intelligent Driver Assistance Systems : Perspectives and Survey,” 13(4) ,1484–1497 , 2012.

[28] Bach K.M. *et al.*, “You Can Touch , but You Can ’ t Look : Interacting with In-Vehicle Systems,” 1139–1148, 2008.

[29] Mourant, R. R. and Rockwell, T.,"Strategies of Visual Search by Novice and Experienced Drivers,” 14(4), pp. 325–335, 1972.

[30] Jin, C., Zhu, Z., Bai, Y., Jiang, G. and He, A., “A deep-learning-based scheme for detecting driver cell-phone use,” *IEEE Access*, vol. 8, 18580–18589, 2020.

[31] Žuraulis, V., Nagurnas, S., Pečeliunas, R., Pumputis, V. and Skačkauskas, P., “The analysis of drivers’ reaction time using cell phone in the case of vehicle stabilization task,” *Int. J. Occup. Med. Environ. Health*, 31(5), 633–648, 2018.

[32] Shi, X., Shan, S., Kan, M., Wu, S. and Chen, X., “Real-Time Rotation-Invariant Face Detection with Progressive Calibration Networks," 2018.

[33] Xiong, Q., Lin, J., Yue, W., Liu, S., Liu, Y. and Ding, C., “A Deep Learning Approach to Driver Distraction Detection of Using Mobile Phone,” 2019.

[34] He, A., Chen, G., Zheng, W., Ni, Z., Zhang, Q. and Zhu, Z., “Driver cell-phone use detection based on cornernet-lite,” *IOP Conf. Ser. Earth. Sci.*, 632(4) , 2021.

[35] Zhang, D., Jiao, C. and Wang, S. “Smoking image detection based on convolutional neural networks,” *2018 IEEE 4th Int. Conf. Comput. Commun. ICCC 2018*, 1509–1515, 2018.

[36] Jia Deng, Wei Dong, R. Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, “ImageNet: A large-scale hierarchical image database,” 248–255, 2009.

[37] https://www-i6.informatik.rwth-aachen.de/~koller/1miohands/., Erişim Tarihi: 01.08.2021.

[38] Cho, J. H., "Detection of smoking in indoor environment using machine learning," *Appl. Sci.*, 10(24), 1–17, 2020.

[39] Craye, C. and Karray, F. "Driver distraction detection and recognition using RGB-D sensor," 1–11, 2015.

[40] Inthanon, P. and Mungsing, S., "Detection of Drowsiness from Facial Images in Real-Time Video Media using Nvidia Jetson Nano," *17th Int. Conf. Electr. Eng. Comput. Telecommun. Inf. Technol. ECTI-CON 2020*, 246–249, 2020.

[41] Savaş, B. K. and Becerikli, Y. "Real Time Driver Fatigue Detection System Based on Multi-Task ConNN," *IEEE Access*, vol. 8, 12491–12498, 2020.

[42] Abtahi, S., Omidyeganeh, M., Shirmohammadi, S. and Hariri, B. "YawDD A yawning detection dataset. In Proceedings of the 5th ACM multimedia systems conference," 24–28, 2014.

[43] Yılmaz, K., Atınç, U., *Derin Öğrenme*. İstanbul: Kodlab Yayın Dağıtım Yazılım Ltd. Şti, 2019.

[44] François, C., *Deep learning with Python*. New York: Manning, 2018.

[45] Lindsay, G.W., "Convolutional Neural Networks as a Model of the Visual System : Past , Present , and Future," 1-15, 2020.

[46] Voulodimos, A., Doulamis, N., Doulamis, A. and Protopapadakis, E., "Deep Learning for Computer Vision : A Brief Review," 2018.

[47] Le, J., "Convolutional Neural Networks: The Biologically Inspired Model, Towards data science," 2018.

[48] www.edge-ai-vision.com/2018/09/whats-the-difference-between-a-cnn-and-an-rnn., Erişim Tarihi: 07.06.2021.

[49] Andrej, K., CS231n Convolutional Neural Networks for Visual Recognition, Stanford University, 2016.

[50] Hubel, T. N., Wıesel, D. H., Receptive fields of single neurones in the cat's striate cortex, J. Physiol, 574–591, 1959.

[51] Hubel, H., Wiesel, N., And Functional Architecture In The Cat's Visual Cortex From The Neurophysiolojy Laboratory , Department Of Pharmacology Central Nervous System is The Great Diversity Of its Cell Types And Inter- Receptive Fields Of A More Complex Type (Part I) And To , 106–154, 1962.

[52] Contribution, O., "Neocognitron : A Hierarchical Neural Network Capable of Visual Pattern Recognition," vol. 1, pp. 119–130, 1988.

[53] https://www.glassboxmedicine.com/2019/04/13/a-short-history-of-convolutional-neural-networks/., Erişim Tarihi: 07.06.2021.

[54] Lecun, Y., Bottou, L., Bengio, Y., Haffner, P., Gradient-Based Learning Applied to Document Recognition, Proc. IEEE, 86(11), 2278–2324, 1998.

[55] Alom, Z., Taha, T. M., Yakopcic, C., Westberg, S., Sidike, P., A State-of-the-Art Survey on Deep Learning Theory and Architectures, 2019.

[56] Krizhevsky, A., Sutskever, I., Hinton, E., ImageNet Classification with Deep Convolutional Neural Networks, Adv. Neu Inf. Process. Syst., 1097-1105, 2012.

[57] www.medium.com/srm-mic/convnet-architectures-for-beginners-part-i-233aa9d1761b., Erişim Tarihi: 07.06.2021.

[58] Simonyan, K., Zisserman, A., Very deep convolutional networks for large-scale image recognition, 3rd Int. Conf. Learn. Represent. ICLR 2015, 1–14, 2015.

[59] Jun, H., Shuai, L., Jinming, S., Yue, L., Jingwei, W., Peng, J., Facial expression recognition based on VGGNet convolutional neural network, Chinese Autom. Congr. (CAC). IEEE, 4146-4151, 2018.

[60] Szegedy, A., Liu, C., Jia, W., Sermanet, Y., Reed, P., Anguelov, S., Going deeper with convolutions, Proc. IEEE Conf. Comput. Vis. pattern Recognit, 1-9, 2015.

[61] Sultana, F., Sufian, A., Dutta, P., A review of object detection models based on convolutional neural network, Adv. Intell. Syst. Comput., 1–16, 2020.

[62] Girshick, R., Donahue, J., Darrell, T., Malik, J., "Rich feature hierarchies for accurate object detection and semantic segmentation," 2014.

[63] Gandhi, R., R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms, Towards Data Science. 2018.

[64] Girshick, R., Fast R-CNN," *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2015 Inter, 1440–1448, 2015.

[65] He, K., Zhang, X., Ren, S., Sun, J., Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition, 37(9), 1904–1916, 2015.

[66] Ren, S., He, K., Girshick, R., Sun, J., Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(6), 1137–1149, 2017.

[67] Liu, W. *et al.*, "SSD: Single shot multibox detector," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, 9905 LNCS, 21–37, 2016.

[68] Redmon, J., Divvala, S., Girshick, R., Farhadi, A., You only look once: Unified, Real-time Object Detection, *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 779–788, 2016.

[69] Li S., *et al.*, Detection of concealed cracks from ground penetrating radar images based on deep learning algorithm, *Constr. Build. Mater.*, 121949, 2021.

[70] Wang, Z., Wu, Y., Yang, L., Thikarasu, A., Fast Person Protive Equipm Detect for Real Construction Sites Using Deep Learning Approaches, 1–22, 2021.

[71] Jocher, G., Yolov5 in PyTorch, www.github.com/ultralytics/yolov5., Erişim Tarihi: 07.06.2021.

[72] Yan, B., Fan, P., Lei, X., Liu, Z., A Real-Time Apple Targets Detection Method for Picking Robot Based on Improved Yolov5, 1–23, 2021.

[73] Wang, C. Y., Liao, H. Y., Wu, Y. H., Chen, P. Y., Hsieh, J. W., Yeh, I. H., CSPNet: A new backbone that can enhance learning capability of CNN," IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Work., 1571–1580, 2020.

[74] Liu, S., Qi, L., Qin, H., Shi, J., Jia, J., Path Aggregation Network for Instance Segmentation, 8759–8768, 2018.

[75] www.github.com/ultralytics/yolov5., Erişim Tarihi: 07.06.2021.

[76] Huang, G., Liu, S. and van der Maaten, L., "CondenseNet: An Efficient DenseNet using Learned Group Convolutions," Proc. IEEE Conf. Comput. Vis. pattern recognition. 2018.

[77] Iandola, F., Moskewicz, M., Karayev, S., Girshick, R., Darrell, T., Keutzer, K., DenseNet: Implementing Efficient ConvNet Descriptor Pyramids, 1–11, 2014.

[78] Lin *et al.*, “Feature Pyramid Networks for Object Detection," 2117-2125, 2017.

[79] www.blog.roboflow.com/a-thorough-breakdown-of-yolov4., Erişim Tarihi: 16.06.2021.

[80] Xu, R., Lin, H., Lu, K., Cao, L., Liu, Y., A Forest Fire Detection System Based on Ensemble Learning. Forests, 12(2), 2021.

[81] www.makesense.ai., Erişim Tarihi: 16.06.2021.

[82] www.research.google.com/colaboratory/faq.html., Erişim Tarihi: 12.06.2021.

[83] Thuan, D., Evolution of Yolo Algorithm and Yolov5: the State-of-the-Art Object Detection Algorithm, 61, 2021.

[84] Kasper-Eulaers, M., Hahn, N., Berger, S., Sebulonsen, T., Myrland, Ø. and Kummervold, P. E. "Detecting Heavy Goods Vehicles in Rest Areas in Winter Conditions Using YOLOv5. Algorithms," 14(4), 114, 2021.

[85] Jetson Xavier NX Developer Kit | NVIDIA Developer, https://developer.nvidia.com/embedded/jetson-xavier-nx-devkit, Erişim Tarihi: 21.07.2021.

[86] User Guide Of Jetson Xavier Nx, https://elinux.org/Jetson_Nano, Erişim Tarihi: 21.07.2021.

ÖZGEÇMİŞ

Adı Soyadı **:** Emin Güney

ÖĞRENİM DURUMU

Derece	**Eğitim Birimi**	**Mezuniyet Yılı**
Yüksek Lisans	Sakarya Üniversitesi / Fen Bilimleri Enstitüsü / Bilgisayar Mühendisliği	Devam Ediyor
Lisans	Sakarya Üniversitesi / Mühendislik Fakültesi / Bilgisayar Mühendisliği	2019
Lise	Sultanahmet Endüstri Meslek Lisesi/ Bilgisayar Bölümü	2014

İŞ DENEYİMİ

Yıl	**Yer**	**Görev**
08.2020 – Halen	Sakarya Uygulamalı Bilimler Üniversitesi	Araştırma Görevlisi
03.2019 – 06.2019	Simurg Bilişim Mobil Haritacılık Ltd. Şti.	Yazılım Stajı
02.2018 – 05.2018	Inovenso Nanospinning Machine Ltd. Şti.	Donanım Stajı
09.2013 – 06.2014	Türk Hava Yolları A.O. Genel Yönetim	Yazılım Stajı

YABANCI DİL

İngilizce