
GAZİ ÜNİVERSİTESİ
Gazi
1926

GERÇEK ZAMANLI NESNE ALGILAMA ALGORİTMALARININ KARŞILAŞTIRILMASI

Elif Nur YILMAZ

YÜKSEK LİSANS TEZİ
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANA BİLİM DALI

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

HAZİRAN 2025

ETİK BEYAN

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

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

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

Elif Nur YILMAZ

26/06/2025

GERÇEK ZAMANLI NESNE ALGILAMA ALGORİTMALARININ KARŞILAŞTIRILMASI

(Yüksek Lisans Tezi)

Elif Nur YILMAZ

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

Haziran 2025

ÖZET

Son yıllarda yapay zekâ ve derin öğrenme alanlarında yaşanan teknolojik ilerlemeler, bilgisayarla görme sistemlerinin özellikle nesne algılama konusunda büyük bir gelişim göstermesini sağlamıştır. Bu tez çalışması kapsamında, karşılaştırmaya dâhil edilen modeller arasında en güncel YOLO ailesine ait YOLOv8, YOLOv9, YOLOv10 ve YOLO11 modellerinin küçük (s), orta (m) ve büyük (l) varyantları, geleneksel ancak yaygın olarak kullanılan SSD300-VGG16, ve ölçeklenebilir yapısıyla dikkat çeken EfficientDet (D0–D7) serisi yer almaktadır. Tüm modeller, MS COCO 2017 veri kümesi kullanılarak, PyTorch 2.0.1 çerçevesinde aynı donanım (Intel i7 CPU, 16 GB RAM, NVIDIA RTX 3050 GPU) ve yazılım konfigürasyonları altında test edilmiştir. Karşılaştırma, yalnızca doğruluk metrikleriyle sınırlı kalmamış; saniyedeki kare sayısı (FPS), çıkarım süresi (inference time), kesinlik (precision), duyarlılık (recall), F1-skoru, Ortalama Ortalama Hassasiyet (mAP), mAP50:95 ve Kesişim-Birleşim Oranı (IoU) gibi performans parametreleri üzerinden gerçekleştirilmiştir. Deneysel sonuçlar, özellikle YOLOv8s ve YOLOv10s modellerinin hem yüksek FPS değerleri hem de tatmin edici doğruluk skorları ile ön plana çıktığını göstermektedir. Buna karşın, SSD300-VGG16 modeli, rakiplerinin oldukça gerisinde kalmaktadır. EfficientDet D4-D7 modelleri yüksek yerelleştirme başarısıyla dikkat çekmektedir. Tezde ayrıca, modellerin zorlu koşullardaki dayanıklılığı da test edilmiştir. YOLOv10 ve YOLO11 modellerinin daha kararlı sonuçlar üretmiştir. Bunlara ek olarak, tezde kod seviyesinde yapılan optimizasyonların da performans üzerindeki etkisi ayrıntılı olarak ele alınmıştır. Bu sonuçlar ile yazılım optimizasyonunun da gerçek zamanlı sistemlerde önemli rol oynadığını açıkça ortaya koymaktadır. Sonuç olarak bu çalışma, farklı kullanım senaryoları için en uygun modeli belirlemede kapsamlı bir rehber sunmayı amaçlamaktadır. Gerçek zamanlılık, doğruluk ve sistem kaynaklarının dengeli kullanımı gibi pratik kaygıları ele alan bu tez; hem akademik alanda araştırma yapanlar hem de endüstriyel çözümler geliştiren profesyoneller için yol gösterici niteliktedir.

Bilim Kodu	:	93436
Anahtar Kelimeler	:	Nesne algılama, gerçek zamanlı nesne algılama, YOLO, SSD, EfficientDet, MS COCO veri seti
Sayfa Adedi	:	107
Danışman	:	Prof. Dr. Tuğba Selcen NAVRUZ

COMPARISION OF REAL-TIME OBJECT DETECTION ALGORITHMS

(M. Sc. Thesis)

Elif Nur YILMAZ

GAZİ UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

June 2025

ABSTRACT

In recent years, technological advances in the fields of artificial intelligence and deep learning have enabled computer vision systems to make significant progress especially in object detection. Within the scope of this thesis, the models included in the comparison include small (s), medium (m) and large (l) variants of the latest YOLO family YOLOv8, YOLOv9, YOLOv10 and YOLO11 models, the traditional but widely used SSD300-VGG16, and the EfficientDet (D0–D7) series, which stands out with its scalable structure. All models were tested using the MS COCO 2017 dataset within the PyTorch 2.0.1 framework under the same hardware (Intel i7 CPU, 16 GB RAM, NVIDIA RTX 3050 GPU) and software configurations. The comparison was not limited to accuracy metrics only; The performance parameters such as frames per second (FPS), inference time, precision, recall, F1-score, Mean Average Precision (mAP), mAP50:95 and Intersection-Joinage Ratio (IoU) were used. Experimental results show that especially YOLOv8s and YOLOv10s models stand out with both high FPS values and satisfactory accuracy scores. On the other hand, SSD300-VGG16 model lags far behind its competitors. EfficientDet D4-D7 models stand out with their high localization success. The thesis also tested the durability of the models under difficult conditions. YOLOv10 and YOLO11 models produced more stable results. In addition, the effect of optimizations made at the code level on performance is discussed in detail in the thesis. These results clearly show that software optimization also plays an important role in real-time systems. As a result, this study aims to provide a comprehensive guide to determine the most suitable model for different usage scenarios. Addressing practical concerns such as real-time, accuracy and balanced use of system resources, this thesis is a guide for both academic researchers and professionals developing industrial solutions.

Science Code	:	93436
Key Words	:	Object detection, real-time object detection, YOLO, SSD, EfficientDet, MS COCO dataset
Page Number	:	107
Supervisor	:	Prof. Dr. Tuğba Selcen NAVRUZ

TEŞEKKÜR

Çalışmalarım boyunca değerli yardım ve katkılarıyla beni yönlendiren, hatalarımı sabırla düzelten, akademik hayat ile ilgili tecrübelerini aktaran, hiçbir konuda yardımını esirgemeyen ve her zaman desteğini hissettiren kıymetli Hocam Prof. Dr. Tuğba Selcen NAVRUZ'a, bugünlere gelmemi sağlayan ve benim için hiçbir fedakârlıktan kaçınmayan sevgili aileme, bana inanan ve desteğini her zaman gösteren sevgili eşim Mustafa YILMAZ'a ve dostlarıma sonsuz teşekkürlerimi sunarım.

İÇİNDEKİLER

Sayfa

ÖZET iv

ABSTRACT v

TEŞEKKÜR vii

İÇİNDEKİLER vii

ÇİZELGELERİN LİSTESİ ix

ŞEKİLLERİN LİSTESİ x

SİMGELER VE KISALTMALAR xiii

1. GİRİŞ 1

2. LİTERATÜR TARAMASI 5

3. TEORİK ALTYAPI 9

3.1. MS COCO Veri Seti ve Ön İşleme Adımları 9

3.2. Derin Öğrenme Modelleri 11

3.2.1. YOLO 12

3.2.2. SSD 27

3.2.3. EfficientDet 30

3.3. Performans Metrikleri 34

3.3.1. Hız Metrikleri: FPS ve çıkarım süresi 34

3.3.2. Doğruluk metrikleri: kesinlik (precision), duyarlılık (recall) ve f1-skoru (f1-score) 35

3.3.3. Nesne yerelleştirme metrikleri: ortalama ortalama hassasiyet (mean average precision - mAP) ve kesişim / birleşim orani (intersection over union - IoU) 37

4. DENEYSEL SONUÇLAR VE TARTIŞMA 39

4.1. Deney Ortamı ve Test Koşulları 39

Sayfa

4.2. YOLO Modelleri İle Nesne Algılama Performans Analizi 40

4.3. SSD Modelleri İle Nesne Algılama Performans Analizi 49

4.4. EfficientDet Modelleri İle Nesne Algılama Performans Analizi 52

4.5. YOLO, SSD ve EfficientDet Modellerinin Performans Metrikleri Karşılaştırması 59

4.5.1. FPS karşılaştırması 60

4.5.2. Çıkarım süresi (inference time) karşılaştırması 61

4.5.3. Kesinlik (precision) karşılaştırması 63

4.5.4. Duyarlılık (recall) karşılaştırması 64

4.5.5. F1-Skoru (f1-score) karşılaştırması 65

4.5.6. mAP karşılaştırması 67

4.5.7. IoU karşılaştırması 69

4.5.8. Performans metriklerinin grafiksel analizi ve genel değerlendirme 71

4.6. Zorlu Senaryolarda YOLO, SSD ve EfficientDet Modellerinin Karşılaştırılması 78

4.7. Kod Optimizasyonunun Gerçek Zamanlı Nesne Algılama Performansına Etkileri 86

5. SONUÇLAR VE ÖNERİLER 97

KAYNAKLAR 101

EKLER 105

EK-1. MS COCO 2017 Veri Setinde Kullanılan Nesne Sınıfları Listesi 106

ÖZGEÇMİŞ 107

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

Çizelge 3.1. Maksimum olmayan bastırma (NMS) algoritması..... 15

Çizelge 3.2. COCO veri kümesinde YOLOv8 modelleri için performans ölçümleri 19

Çizelge 3.3. COCO veri kümesinde YOLOv9 modelleri için performans ölçümleri 21

Çizelge 3.4. COCO veri kümesinde YOLOv10 modelleri için performans ölçümleri..... 24

Çizelge 3.5. COCO veri kümesinde YOLO11 modelleri için performans ölçümleri 27

Çizelge 3.6. COCO test-dev2015 SSD modelleri için performans ölçümleri 1..... 29

Çizelge 3.7. COCO test-dev2015 SSD modelleri için performans ölçümleri 2..... 29

Çizelge 3.8. EfficientDet D0-D6 için ölçekleme yapılandırmaları–φ diğer tüm ölçekleme boyutlarını kontrol eden bileşik katsayıdır; BiFPN, kutu/sınıf ağı ve giriş boyutu sırasıyla denklem 3.1, 3.2, 3.3 kullanılarak ölçeklenir 31

Çizelge 3.9. COCO veri kümesinde EfficientDet D0-D7 modelleri için performans ölçümleri..... 33

Çizelge 4.1. Hız performans parametreleri hesaplama sonuçları..... 73

Çizelge 4.2. Algılama doğruluğu parametreleri hesaplama sonuçları..... 75

Çizelge 4.3. Nesne yerelleştirme parametreleri hesaplama sonuçları 78

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

Şekil 3.1. YOLO mimarisi 12

Şekil 3.2. YOLO algılama sistemi 13

Şekil 3.3. YOLO model çalışma prensibi 14

Şekil 3.4. Maksimum olmayan bastırma (NMS) (a) Birden fazla örtüşen kutu içeren bir nesne algılama modelinin tipik çıktısı (b) NMS'den sonraki çıktı 15

Şekil 3.5. YOLOv8 mimarisi. 17

Şekil 3.6. YOLOv9 mimarisi - PGI esas olarak üç bileşenden oluşur (1) ana dal: çıkarım için kullanılan mimari (2) yardımcı geri dönüşümlü dal: geri iletim için ana dalı sağlamak üzere güvenilir gradyanlar üretir ve (3) çok seviyeli yardımcı bilgi: ana dalın planlanabilir çok seviyeli anlamsal bilgiyi öğrenmesini kontrol eder 20

Şekil 3.7. YOLOv10 mimarisi 22

Şekil 3.8. YOLO11 mimarisi 25

Şekil 3.9. SSD mimarisi 27

Şekil 3.10. EfficientDet mimarisi - omurga ağı olarak EfficientNet[36], özellik ağı olarak BiFPN ve paylaşılan sınıf/kutu tahmin ağı kullanır. Hem BiFPN katmanları hem de sınıf/kutu ağı katmanları, Çizelge 3.7'de gösterildiği gibi farklı kaynak kısıtlamalarına göre birden çok kez tekrarlanır 30

Şekil 4.1. YOLOv8 - s versiyonu ile nesne algılama ekran görüntüsü 42

Şekil 4.2. YOLOv8 - m versiyonu ile nesne algılama ekran görüntüsü 42

Şekil 4.3. YOLOv8 - l versiyonu ile nesne algılama ekran görüntüsü 43

Şekil 4.4. YOLOv9 - s versiyonu ile nesne algılama ekran görüntüsü 44

Şekil 4.5. YOLOv9 - m versiyonu ile nesne algılama ekran görüntüsü 44

Şekil 4.6. YOLOv9 - c versiyonu ile nesne algılama ekran görüntüsü 45

Şekil 4.7. YOLOv10 - s versiyonu ile nesne algılama ekran görüntüsü 46

Şekil 4.8. YOLOv10 - m versiyonu ile nesne algılama ekran görüntüsü 46

Şekil 4.9. YOLOv10 - l versiyonu ile nesne algılama ekran görüntüsü 47

Şekil **Sayfa**

Şekil 4.10. YOLO11 - s versiyonu ile nesne algılama ekran görüntüsü 48

Şekil 4.11. YOLO11 - m versiyonu ile nesne algılama ekran görüntüsü 48

Şekil 4.12. YOLO11 - l versiyonu ile nesne algılama ekran görüntüsü 49

Şekil 4.13. SSD300-VGG16 modeli ile nesne algılama ekran görüntüsü 1 51

Şekil 4.14. SSD300-VGG16 modeli ile nesne algılama ekran görüntüsü 2 51

Şekil 4.15. EfficientDet-D0 versiyonu ile nesne algılama ekran görüntüsü 54

Şekil 4.16. EfficientDet-D1 versiyonu ile nesne algılama ekran görüntüsü 54

Şekil 4.17. EfficientDet-D2 versiyonu ile nesne algılama ekran görüntüsü 55

Şekil 4.18. EfficientDet-D3 versiyonu ile nesne algılama ekran görüntüsü 56

Şekil 4.19. EfficientDet-D4 versiyonu ile nesne algılama ekran görüntüsü 57

Şekil 4.20. EfficientDet-D5 versiyonu ile nesne algılama ekran görüntüsü 57

Şekil 4.21. EfficientDet-D6 versiyonu ile nesne algılama ekran görüntüsü 58

Şekil 4.22. EfficientDet-D7 versiyonu ile nesne algılama ekran görüntüsü 59

Şekil 4.23. FPS analiz sonuçları 60

Şekil 4.24. Çıkarım süresi (inference time) analiz sonuçları 62

Şekil 4.25. Kesinlik (precision) analiz sonuçları 63

Şekil 4.26. Duyarlılık (recall) analiz sonuçları 64

Şekil 4.27. F1-Skoru analiz sonuçları 66

Şekil 4.28. mAP analiz sonuçları 68

Şekil 4.29. mAP50-95 analiz sonuçları 69

Şekil 4.30. IoU analiz sonuçları 70

Şekil 4.31. Hız performans analiz sonuçları 73

Şekil 4.32. Algılama doğruluğu analiz sonuçları 75

Şekil 4.33. Nesne yerelleştirme analiz sonuçları 77

Şekil **Sayfa**

Şekil 4.34. Yoğun nesne senaryosunda YOLOv10l modeli ile ile nesne algılama ekran görüntüsü 79

Şekil 4.35. Yoğun nesne senaryosunda YOLO11l modeli ile ile nesne algılama ekran görüntüsü 80

Şekil 4.36. Yoğun nesne senaryosunda SSD300-VGG16 modeli ile ile nesne algılama ekran görüntüsü 81

Şekil 4.37. Yoğun nesne senaryosunda EfficientDet-D0 modeli ile ile nesne algılama ekran görüntüsü 82

Şekil 4.38. Yoğun nesne senaryosunda EfficientDet-D7 modeli ile ile nesne algılama ekran görüntüsü 82

Şekil 4.39. Yoğun nesne senaryosunda YOLOv8s modeli ile ile nesne algılama ekran görüntüsü 83

Şekil 4.40. Yoğun nesne senaryosunda YOLO11s modeli ile ile nesne algılama ekran görüntüsü 84

Şekil 4.41. Yoğun nesne senaryosunda SSD300-VGG16 modeli ile ile nesne algılama ekran görüntüsü 85

Şekil 4.42. Yoğun nesne senaryosunda EfficientDet-D6 modeli ile ile nesne algılama ekran görüntüsü 85

Şekil 4.43. Kod optimizasyonları yapılmadan öncesinde EfficientDet-D1 modeli için elde edilen ekran görüntüsü 88

Şekil 4.44. Kod optimizasyonları yapılmadan öncesinde YOLO modelleri için elde edilen FPS, kesinlik (precision), F1-Skoru ve IoU performans metriklerinin hesaplanan sonuçları 89

Şekil 4.45. Kod optimizasyonları yapılmadan öncesinde YOLO modelleri için elde edilen çıkarım süresi (inference time), duyarlılık (recall) ve mAP performans metriklerinin hesaplanan sonuçları 90

Şekil 4.46. Kod optimizasyonları yapılmadan öncesinde elde edilen FPS, çıkarım süresi, kesinlik ve duyarlılık performans metriklerinin hesaplanan sonuçları 91

Şekil 4.47. Kod optimizasyonları yapılmadan öncesinde elde edilen F1-Skoru, mAP ve IoU performans metriklerinin hesaplanan sonuçları 92

Şekil 4.48. Kod optimizasyonları yapıldıktan sonra EfficientDet-D1 modeli için elde edilen ekran görüntüsü 96

SİMGELER VE KISALTMALAR

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

Simgeler	**Açıklamalar**
s	Saniye
Σ	Toplam Sembolü
$\boldsymbol{B}$	Tahmini Sınırlayıcı Kutular Kümesi
$\boldsymbol{S}$	Güven Puanları
$\boldsymbol{\tau\tau}$	IoU Eşiği
$\boldsymbol{T}$	Güven Eşiği
$\boldsymbol{F}$	Filtrelenmiş Sınırlayıcı Kutular Kümesi
$\boldsymbol{W_{bifpn}}$	BiFPN Genişliği
$\boldsymbol{D_{bifpn}}$	BiFPN Derinliği
$\boldsymbol{\varphi}$	Bileşik Katsayı
$\boldsymbol{D_{box}}$	BiFPN Kutu Derinliği
$\boldsymbol{D_{class}}$	BiFPN Sınıf Tahmin Derinliği
$\boldsymbol{R_{input}}$	BiFPN'de Giriş Görüntüsü Çözünürlüğü
$\boldsymbol{TP}$	Doğru Pozitifler
$\boldsymbol{FP}$	Yanlış Pozitifler
$\boldsymbol{FN}$	Yanlış Negatifler

Kısaltmalar	Açıklamalar
BiFPN	Çok Yönlü Özellik Piramidi Ağı
C2f	Birleşmeli Kısmi Aşama Geçişi
C2PSA	Paralel Uzamsal Dikkat Evrişimsel Blok
C3k2	2x2 Çekirdekli Çapraz Evrişim Bloğu
CBS	Evrişim, Normalizasyon ve Aktivasyon Bloğu
CNN	Evrişimli Sinir Ağı
CPU	Merkezi İşlem Birimi
CSPDarknet	Aşamalı Kısmi Bağlantılı Darknet
CUDA	Birleşik Hesaplama Aygıt Mimarisi
CutMix	Kes-Karıştır Yöntemi
DETR	Transformer Tabanlı Nesne Algılama Modeli
ELAN	Verimli Katman Toplama Ağı
FLOP	Kayan Noktalı İşlem Sayısı
FPN	Özellik Piramidi Ağı
FPS	Saniyedeki Kare Sayısı
G-ELAN	Genelleştirilmiş Verimli Katman Toplama Ağı
GELU	Gauss Hatalı Doğrusal Birim
GPU	Grafik İşlem Birimi
IoU	Kesişim-Birleşim Oranı
mAP	Ortalama Ortalama Doğruluk
MoSAIC	Mozaik Veri Artırma
MS COCO	Microsoft Ortamda Ortak Nesneler Veri Kümesi
MSE	Ortalama Kare Hata
NMS	Maksimum Olmayan Bastırma
ONNX	Açık Sinir Ağı Değişim Formatı
PAN	Yol Toplama Ağı
Pascal - VOC	Pascal Görsel Nesne Sınıfları
PGI	Yol Yönlendirmeli Çıkarım
PIL	Python Görüntü İşleme Kütüphanesi
RAM	Rastgele Erişimli Bellek
R-CNN	Bölge Tabanlı Evrişimli Sinir Ağı
ReLU	Doğrultulmuş Doğrusal Birim

Kısaltmalar	Açıklamalar
SAM	Keskinlik – Farkında Minimizasyon
SiLU	Sigmoid Doğrusal Birim
SPPF	Hızlı Uzamsal Piramit Havuzlama
SSD	Tek Atışlı Çoklu Kutu Algılayıcı
VGG	Görsel Geometri Grubu
YOLO	Sadece Bir Kez Bak

1. GİRİŞ

Bilgisayarlı görü, sayısal ya da sembolik olarak bilgi üretmeyi amaçlayarak makinelere ve bilgisayarlara makine öğrenimi tekniklerine dayalı olarak nesnelerin görselleştirilmesini sağlayan yapay zekâ alanıdır. En temelde görsel araçlarla bir görüntü alarak, alınan görüntünün işlenmesi ve anlamlandırılması işlemlerinin yapar. Bilgisayarlı görünün kullanım alanları, görüntü sınıflandırma, nesne algılama, nesne takibi, segmentasyon ve içerik tabanlı görüntü alma olarak listelenebilir.

Nesne algılama, yapay zekâ ve derin öğrenme tekniklerini kullanarak bir görüntüdeki belirli nesne sınıflarını tespit ederek nesnenin sınıfını belirlemeyi sağlayan ve nesnenin görüntüdeki konumunu belirlemeyi amaçlayan ve gerçek zamanlı da çalışabilen bilgisayarlı görü alanında temel taşı olan görevlerden biridir. Nesne algılma, algılanan nesnelerin etrafına onları belirli bir görselde bulmamıza olanak tanıyan sınırlayıcı kutular çizmeye dayanır. Nesne algılama, makine öğrenimini kullanan geleneksel görüntü işleme teknikleri veya modern derin öğrenme ağları kullanılarak gerçekleştirilmektedir. Derin öğrenme ağları kullanılarak geliştirilen yöntemler zorlu aydınlatma ve çeşitli çok sayıda nesnenin aynı anda yer aldığı karmaşık sahnelerde yüksek doğruluk oranlarıyla tepki vermesi ve gerçek zamanlı uygulamalardaki yüksek doğruluklu algılama yapabilmesi sayesinde makine öğrenmesi kullanılarak geliştirilmiş geleneksel yöntemlere göre daha hızlı gelişim göstermiştir.

Son yıllarda derin öğrenmede yaşanan ilerlemeler sayesinde nesne algılama, güvenlik ve gözetim, otonom araçlar, perakende sektörü, sağlık sektörü, tarım ve çeşitli endüstriyel uygulamalar başta olmak üzere birçok alanda kendine uygulama alanı bularak günlük hayatın vazgeçilmez teknolojilerinin geliştirilmesine ortam hazırlamıştır. Ayrıca, derin öğrenmedeki bu önemli ilerlemeler, gerçek zamanlı nesne tespitini yüksek doğrulukla gerçekleştirebilen algoritmaların geliştirilmesini kolaylaştırmıştır. Bu gerçek zamanlı nesne tespiti algoritmaları arasında, Sadece Bir Kez Bak (YOLO - You Only Look Once) [1], Tek Atış Dedektörü (SSD - Single Shot Detector) [2] ve EfficientDet [3] gibi son teknoloji algoritmalar, gerçek zamanlı senaryolardaki yüksek doğruluk oranlı nesne tespiti yapabilmeleri, hızlı sonuç üretebilme kabiliyetleri, kullanılan donanım konusunda sağlayabilecekleri maliyet iyileştirmeleri, birçok alanda uygulanabilir olmaları ve hatta belirli zorlu koşullarda dahi yüksek başarılı etkinlikleri sayesinde ön plana çıkmıştır.

Bu algoritmaların dikkate değer başarısına rağmen, belirli bir gerçek zamanlı uygulama için en uygun yaklaşımı seçmek zorlu bir karar verme süreci olmaya devam etmektedir. Bu seçim, birden fazla parametrenin kapsamlı bir şekilde değerlendirilmesini gerektirir. Karar verme aşamasında kullanılan parametrelere donanım gereksinimleri, işlem hızı, enerji tüketimi kısıtları, doğruluk ve hassasiyet oranları, platform uyumluluğu, uygulamaya yönelik spesifik gereklilikler, optimizasyon gereklilikleri, güvenilirlik ve sağlamlık, entegrasyon kolaylığı, güvenlik ve gizlilik ve maliyet faktörleri örnek verilebilir.

Her algoritma, gerçek zamanlı performansı tespit doğruluğu ile dengelemek için farklı mimarı tasarımlar ve optimizasyon teknikleri benimser. Örneğin, YOLO, özellikle songeliştirilen versiyonlarında, nesneyi tek bir regresyon görevi olarak çerçeveleyerek gerçek zamanlı nesne tespitindeki hızı ve yetenekleri ile ön plana çıkmaktadır. SSD, çeşitli nesne boyutlarında tespit doğruluğunu artırmak için çok ölçekli bir özellik gösteriminden yararlanır ve bu sayede hız ve doğruluk arasında bir denge sunar. EfficientNet mimarisi üzerine inşa edilen EfficientDet, hem doğruluğu hem de verimliliği optimize etmek için yeni bir bileşik ölçekleme yöntemi sunar. Çıkarım hızı, tespit doğruluğu ve hesaplama kaynağı kısıtlamaları gibi kritik faktörler, belirli bir algoritmanın uygunluğunu önemli ölçüde etkiler.

Literatürde, bu algoritmaların her biri için ayrı ayrı güçlü ve zayıf yönlerini kapsamlı bir şekilde araştırıp çoğunlukla bazı iki aşamalı nesne algılama algoritmaları ile yapılan karşılaştırma çalışmaları yer almakla birlikte, özellikle en güncel yöntemler, algoritmalar veya modeller kullanılarak gerçek zamanlı uygulamalar için standartlaştırılmış koşullar altında yürütülen karşılaştırmalı değerlendirmeler nispeten daha azdır. Önceki çalışmalar ağırlıklı olarak doğruluk veya hıza odaklanmış ve gerçek dünya senaryolarında karşılaşılan pratik uzlaşmaların bütünsel bir değerlendirmesini genellikle ihmal etmiştir.

Bu tez çalışması, YOLO, SSD ve EfficientDet algoritmalarının kapsamlı bir karşılaştırmalı analizini sağlayarak bu boşlukları ele almaktadır. PyTorch ortamında eğitilmiş modeller kullanılmıştır. Nesne algılama modellerinin MS COCO 2017 veri kümesindeki performansının kapsamlı bir karşılaştırması yapılmıştır. Modeller, Saniyedeki Kare Sayısı (FPS), Çıkarım Süresi, Kesişim - Birleşim Oranı (IoU), Ortalama Ortalama Hassasiyet (mAP), kesinlik (precision), duyarlılık (recall) ve F1-Skoru gibi temel gerçek zamanlı performans ölçütlerine göre değerlendirilmiş ve karşılaştırılmıştır. Ayrıca çalışmada, yalnızca ideal koşullarda değil, zorlu senaryolaraltında (özellikle ışık değişimi, nesne

yoğunluğu ve hareketli sahneler gibi) modellerin davranışları da analiz edilmiştir. Bu tür durumlar, gerçek dünya uygulamalarının doğası gereği sıkça karşılaşılan ve algoritmaların dayanıklılığını doğrudan etkileyen faktörlerdir.

Ek olarak, bu tezde kod düzeyindeki optimizasyonların performansa etkisi de detaylı biçimde incelenmiştir. Geliştirilen sistem üzerinde yapılan batch processing, önbelleğe alma, verimli tensor işlemleri ve NMS optimizasyonları gibi kod iyileştirmeleri sayesinde, bazı modellerde %30'a varan FPS artışı ve daha düşük inference süreleri elde edilmiştir. Bu da göstermektedir ki, sadece algoritma seçimi değil, uygulama düzeyindeki mühendislik yaklaşımları da gerçek zamanlı performans açısından belirleyici rol oynamaktadır.
Algoritmalar tutarlı deneysel koşullar altında kıyaslanarak bu çalışmada yapılan analiz, algoritmların ilgili güçlü ve zayıf yönlerini açıklığa kavuşturmayı ve araştırmacılara ve uygulayıcılara kendi özel gereksinimleri için en uygun algoritmayı seçmelerinde rehberlik edecek değerli içgörüler sunmayı amaçlamaktadır.

Bu tez çalışmasının 2. bölümünde gerçek zamanlı nesne algılama üzerine yapılmış ilgili çalışmaların bir incelemesi sunulmaktadır. 3. bölümde, yaplan deneysel çalışmalarda kullanılan MS COCO veri seti ver veri setinin önişleme adımları, karşılaştırılması yapılan gerçek zamanlı nesne algılama algoritmaları ve kullanılan algoritmaların farklı ağırlıklandırma modelleri ve bu modelleri karşılaştırabilmek için seçilen değerlendirme kriterleri sunmaktadır. Ayrıca deney ortamına ait bilgiler verilmektedir. 4.bölümde deneysel sonuçlar ve gerçek zamanlı performas çıktılarını sunulmakta ve veri analizi ve yorumlama yapılmaktadır. 5. ve son bölümde yapılan çalışmanın sonuçları, çıkarımlar ve gelecek çalışmalar için yapılan öneriler bulunmaktadır.

2. LİTERATÜR TARAMASI

Nesne algılama, bilgisayarlı görü alanındaki temel çalışma alanlarından biridir. Görüntülerde veya videolarda belirli nesneleri tespit etmek ve konumlarını belirlemek amacıyla geliştirilmiştir. Bu alanda yapılan ilk çalışmalar Haar-like özellikleri ve Dalga Dönüşümü gibi geleneksel yöntemlere dayanmaktadır [4]. Nesne algılama, özellikle son yıllarda derin öğrenme tabanlı yöntemlerdeki gelişmelerle birlikte önemli bir araştırma konusu olarak literatürde önemli bir yer almışır. Literatürde bu kadar geniş bir araştırma alanı kaplamasının nedeni, çok çeşitli görüntü verilerinin kullanılıyor olması, farklı algoritmaların var olması, yeni algoritmaların hızla geliştiriliyor olması ve güvenlik ve gözetim, otonom araçlar, perakende sektörü, sağlık sektörü, tarım ve çeşitli endüstriyel uygulamalar başta olmak üzere birçok alanda kendine uygulama alanı bularak günlük hayatın vazgeçilmez teknolojilerinin geliştirilmesine imkân sağlamasıdır. Özellikle, Bölge Tabanlı Evrişimsel Sinir Ağları (R-CNN - Region-Based Convolutıonal Neural Networks) serisi [5] ve ardından gelen YOLO (You Only Look Once) [1], SSD (Single Shot MultiBox Detector) [2] ve EfficientDet [3] gibi modeller, hız ve doğruluk açısından önemli ilerlemeleri beraberinde getirmiştir. Bu bölümde, YOLO, SSD ve EfficientDet modellerinin geliştirilmesine öncülük eden çalışmalar özetlenmiştir.

YOLO (You Only Look Once) modeli, Redmon ve arkadaşları tarafından önerilmiş ve gerçek zamanlı nesne tespitinde önemli bir kilometre taşı olmuştur [1]. YOLOv1 modeli, nesne tespitini bir regresyon problemi olarak ele alarak tespit hızını önemli ölçüde artırmıştır. Daha sonra, YOLOv2 [6]'de Darknet-19'un mimari olarak kullanılmaya başlanması ve YOLOv3 [7]'de Darknet-53 mımarisine geçilerek YOLOv2'ye göre kademeli bir iyileştirme yapılması ile birlikte batch normalization, anchor boxes ve multi-scale predictions gibi yenilikler gelmiştir ve bu sayede doğruluk ve performans iyileştirilmiştir. Bu sayede daha iyi sınırlayıcı kutu tahmini, daha doğru sınıf tahmini yapılabilmiş ve farklı ölçeklerde daha doğru tahmin yapılabilmesi sağlanmıştır. YOLOv4 [8], CSPDarknet53 omurgası ve yeni veri artırma teknikleriyle performansı artırmıştır. YOLOv5, topluluk tarafından geliştirilmiş ve PyTorch tabanlı implementasyonuyla dikkat çekmiştir. YOLO modelleri içerisinde PyTorch tabanlı implementasyon kullanılan ilk model olmuştur ve bu veris setlerinin eğitim sürelerinde gözle görülürbir iyileştirme oluşturmuştur.

YOLOv6 [9], donanım dostu, verili tasarıma ve yüksek performansa sahip endüstriyel uygulamalara özel MT-YOLOv6 çerçevesinde geliştirilmiştir. YOLO'nun resmi bir parçası olmamakla birlikte omurgasıorıjınal tek aşamalı YOLO mimarisinden ilham alınarak geliştirildiği için YOLOv6 adı verilmiştir. YOLOv7 [10], yeni hesaplama bloklarını ve optimize edilmiş eğitim stratejilerini tanıtarak MS COCO veri setinde [11] state-of-the-art sonuçlar elde etmiştir. YOLOv8 [12], modüler yapısı ve gelişmiş backbone mimarisiyle öne çıkmıştır. YOLOv9 [13], programlanabilir gradyan bilgisi kullanarak modelin öğrenme kapasitesini artırmıştır. YOLOv10 ve YOLO11, özellikle düşük ışık koşulları ve karmaşık sahnelerde daha iyi performans göstermek üzere tasarlanmıştır.

SSD (Single Shot Multi Box Detector), Liu ve arkadaşları tarafından tanıtılmıştır ve VGG16 tabanlı yapısı ve çoklu ölçekli özellik haritaları kullanarak nesne tespitinde hem doğruluk hem de hız açısından önemli bir ilerleme sağlamıştır [2]. RetinaNet [14], focal loss fonksiyonunu tanıtarak sınıf dengesizliği problemine etkili bir çözüm getirmiştir. SSD, özellikle hafif yapısı ve hız avantajı nedeniyle mobil cihazlar ve gerçek zamanlı uygulamalar için tercih edilen bir modeldir.

EfficientDet, Tan ve Le tarafından EfficientNet tabanlı bir mimari olarak geliştirilmiş ve nesne tespitinde doğruluk ve verimlilik arasında optimum bir denge sağlamıştır [3]. EfficientDet [3], EfficientNet omurgasını ve BiFPN (Bidirectional Feature Pyramid Network) yapısını kullanarak, farklı hesaplama kısıtları için ölçeklenebilir bir çözüm sunmuştur. D0'dan D7'ye kadar olan modeller, MS COCO veri setinde farklı FLOPs seviyelerinde state-of-the-art sonuçlar elde etmiştir. EfficientDet modelleri, ölçeklenebilirlik açısından önemli avantajlar sunarken, özellikle mAP metriği açısından en üst düzey performansı hedeflemiştir.

Nesne algılama modellerinin değerlendirilmesinde kullanılan metrikler de zaman içinde gelişmiştir. Pascal VOC [15] ile başlayan standartlaşma süreci, MS COCO [11] metrikleriyle daha kapsamlı hale gelmiştir. MS COCO (Microsoft Common Object ın Context) veri seti [11], geniş ve çeşitlendirilmiş bir veri kümesi olup, nesne algılama, segmentasyon ve altyazı oluşturma gibi çeşitli bilgisayarlı görü görevlerinde kullanılmaktadır ve nesne algılama modellerinin değerlendirilmesinde kullanılan farklı metrikler ile birlikte performans ölçümü yapılmaktadır. Bu veri seti, geniş etiketli veri ve karmaşık sahneleri ile modellerin genel performansını değerlendirmek için uygun bir platform sunmaktadır. Liu ve arkadaşları [16],

farklı IoU eşikleri ve nesne ölçekleri için yeni değerlendirme protokolleri önermiştir.

Son yıllarda, farklı nesne algılama modellerinin karşılaştırmalı analizine odaklanan çalışmalar artmıştır. Zhang ve arkadaşları[17], anchor-based ve anchor-free dedektörleri karşılaştırarak adaptif örneklem seçimi stratejilerini incelemiştir. Huang ve diğerleri [18], modern konvolusyonel nesne dedektörlerinin hız/doğruluk dengesi üzerine kapsamlı bir analiz sunmuştur ve farklı uygulama senaryolarında uygun model seçimi için rehberlik sağlamışlardır. Wang ve arkadaşları[19], farklı backbone mimarilerinin nesne algılama performansına etkisini incelemiştir. Carion ve diğerleri [20], Algılama Dönüştürücüsü (DETR - Detection Transformer) ile transformer tabanlı nesne algılama yaklaşımını önererek, uçtan uca nesne tespitinde yeni bir paradigma başlatmıştır.

Günümüzde nesne algılama, otonom araçlarda yaygın olarak kullanılmakta ve Tesla, Waymo gibi firmalar tarafından derin öğrenme tabanlı modellerle geliştirilmektedir [21]. Perakende sektöründe, mağazalarda raf takibi ve müşteri analizi için nesne algılama teknolojileri kullanılmaktadır [22]. Sağlık sektöründe, tıbbi görüntüleme sistemlerinde tümör tespiti gibi uygulamalarda derin öğrenme tabanlı nesnen algılama yöntemleri yaygın olarak kullanılmaktadır [23].

Bu tez çalışmasında, önceki araştırmaların bir devamı olarak, YOLO, SSD ve EfficientDet modelleri aynı koşullar altında kapsamlı bir şekilde karşılaştırılmıştır. Karşılaştırma yapılırken kod optimizasyonu yapılarak daha düşük donanımlı bir test ortamı kullanılarak yüksek donanım kullanılarak yapılan testlerde elde edilen resmi sonuçlara oldukça yakın sonuçlar elde edilmistir. Ayrıca performans değerlendirmeleri için literatürde kullanılan mektrikler kullanılarak hesaplanan sonuçlar detaylı bir şekilde kıyaslanmıştır ve bunlar tablolar ve grafikler üzerinde gösterilmiştir. Karşılaştırmaların ve yorumların gerçek hayat uygulamalarına benzer olması için canlı görüntüler ve video görüntüleri üzerinden testler gerçekleştirilmiştir. Kullanılan videolardan elde edilen görüntüler üzerinden performans değerlendirme sonuçlarına dayalı olarak kıyaslamalı değerlendirmeler yapılmış, modellerin zorlu koşullardaki davranışları alınan video görüntüleri ile karşılaştırmalı olarak değerlendirilmiş ve uygulama senaryolarına uygun algoritma seçimi hakkında öneriler sunulmuştur.

3. TEORİK ALTYAPI

Bu tez çalışmasında, üç önemli modern nesne algılama algoritma ailesinin farklı modellerinin performansını doğruluk, hız ve algılama verimliliği gibi temel performans ölçütlerine odaklanarak adil ve kapsamlı bir değerlendirme sağlamak, sonuçları analiz edip karşılaştırmak için sistematik bir yaklaşım kullanılmıştır. Çalışmada, nesne tespiti alanında state-of-the-art olarak kabul edilen YOLO ailesinin son versiyonları (YOLOv8, YOLOv9, YOLOv10, YOLO11), SSD-VGG16 modeli ve EfficientDet ailesi modellerinin karşılaştırmalı performans değerlendirmesi gerçekleştirilmiştir. Tüm deneyler, MS COCO 2017 veri seti üzerinde yapılmış, Pytorch ortamında eğitilmiş model verileri kullanılmış ve sonuçlar çeşitli performans metrikleri kullanılarak kapsamlı bir şekilde analiz edilmiştir.

Bu başlık altında karşılaştırma yapılacak olan YOLOv8 (s,m,l), YOLOv9 (s,m,l), YOLOv10 (s,m,l), YOLO11 (s,m,l), SSD-VGG16 ve EfficientDet (D0'dan D7'ye kadar D7 dâhil) derin öğrenme modelleri ve kullanılan veri seti anlatılacaktır. Ayrıca derin öğrenme modellerini karşılaştırmada kullanılan performans metrikleri olan Saniye Başına Kare Sayısıs (FPS), Çıkarım Süresi (Inference Time), Duyarlılık (Recall), Kesinlik (Precision), F1-Skoru, Ortalama Hassasiyet (mAP) ve Kesişim Birleşim Oranı (IoU) anlatılacak ve test ortamı hakkında bilgi verilecektir.

3.1. MS COCO Veri Seti ve Ön İşleme Adımları

Çalışmada kullanılan MS COCO 2017 (Microsoft Common Object in Context) veri seti [16], geniş ve çeşitlendirilmiş bir veri kümesi olup, nesne algılama, segmentasyon ve altyazı oluşturma gibi çeşitli bilgisayarlı görü görevlerinde kullanılan benchmark veri setidir ve 80 farklı nesne kategorisi içermektedir. Gerçek dünya senaryolarında elde edilen karmaşık ve çeşitli görüntüler içerir, bu da onu derin öğrenme tabanlı nesne algılama modellerinin eğitimi ve değerlendirilmesi için değerli bir kaynak haline getirmektedir. Toplam 123 287 görüntüden oluşmaktadır. Bu görüntülerin yaklaşık 118 000'i eğitim, 5 000'i doğrulama ve 40 000'i test görüntüsü seti olarak ayrılmıştır.

Sağladığı geniş kapsamlı nesne kategorileri ve çeşitli zorlu senaryoları sayesinde nesne algılama algoritmalarını değerlendirmek için yapılan çalışmalarda en çok tercih edilen veri

seti olarak gösterilmektedir. Her bir görüntü için nesne sınıfları ve sınırlayıcı kutu (bounding box) koordinatları COCO JSON formatında açıklama dosyaları olarak sağlanmıştır.

Bu veri setinin nesne algılama çalışmalarında tercih edilmesinin temel sebepleri aşağıda listelendiği şekilde açıklanmaktadır:

- MS COCO veri seti, nesnelerin farklı açılardan, değişik ışık koşullarında ve karmaşık arka planlar içinde yer aldığı görüntüler içeriyor olması sebebiyle derin öğrenme modellerinin gerçek dünya senaryoları için daha iyi genelleştirme yapmasına olanak sağlamaktadır [24].
- Nesne algılama modellerinin performansını değerlendirmek amacıyla kullanılan en yaygın metriklerden biri olan mAP, bu veri setinde yoğunluklu olarak kullanılmaktadır [25].
- Modern derin öğrenme tabanlı nesne algılama modellerinin eğitiminde kullanılarak yüksek doğruluk seviyelerine ulaşılmasını sağlamaktadır [24].

Bu tez çalışmasında; adil bir değerlendirme sağlamak ve sonuçları analiz edip karşılaştırmak için tüm modeller MS COCO veri seti kullanılanılıp eğitilerek elde edilen ağırlık dosyaları kullanılmışdır. Birden farklı model kullanıldığı için veri setinin modellere uyumlandırılması gerekmektedir. Bu sebeple veri setinde belirli ön işleme adımlarının uygulanması gerekmektedir. MS COCO veri setinin ön işleme adımları aşağıdaki şekilde gerçekleştirilmiştir:

- Tüm modeller farklı özel boyutlarda boyutlandırılmış görüntüler kullanmaktadır. Bu sebeple tüm görüntüler model gereksinimlerine uygun şekilde yeniden ölçeklendirilmiştir.
- Piksel değerleri her modelin kabul ettiği aralığa göre dönüştürülmüş ve normalizasyon işlemi yapılmıştır
- COCO formatındaki etiketler, her modelin kendi kabul ettiği biçime uygun olacak şekilde dönüştürülmüştür.
- Renk uzayı dönüşümleri yapılmıştır.
- Tüm modeller farklı veri formatı kullandığı için alınan görüntü modelin kabul ettiği formata dönüştürülmüştür.

Veri seti ön işleme adımları sayesinde model performansı optimize edilmiş, kullanılan her bir model için ayrı ayrı tutarlı bir girdi formatı sağlanmış, farklı kaynaklardan gelen

görüntüler olsa dahi görüntüler standardize edilmiş ve modellerin doğru çalışması için gerekli ön koşullar sağlanmıştır.

3.2. Derin Öğrenme Modelleri

Bu çalışmada, derin sinir ağlarıyla (deep neural networks) çalışan ve tek aşamalı nesne algılama yöntemleri arasında yer alan algoritmalar incelenmiştir. Derin sınır ağları, görüntü ve video analizinde yüksek doğruluk sağlayarak, karmaşık sahnelerde ve zorlu aydınlatma koşullarında üstün dayanıklılıklarıyla öne çıkmaktadır. Geleneksel nesne algılama yaklaşımları, genellikle iki aşamalı (two-stage) mimariler kullanarak yüksek doğruluk elde etmesine rağmen bu yöntemler yüksek hesaplama maliyeti ve düşük işlem hızı gibi dezavantajlara sahiptir. Özellikle gerçek zamanlı uygulamalarda, daha hızlı ve daha verimli çalışan tek aşamalı nesne algılama modelleri ön plana çıkmaktadır.

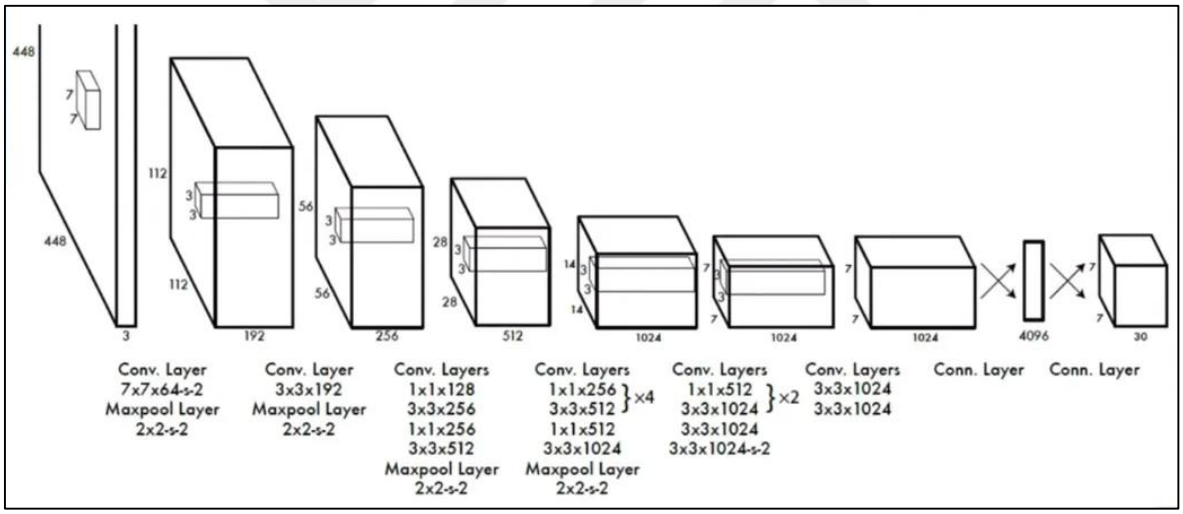
Tek aşamalı nesne algılama algoritmaları, doğrudan girdiden çıkarım yaparak (end-to-end) bölge öneri (region proposal) aşamasını atlamakta ve nesne tespitini tek bir işlem adımında gerçekleştirmektedir. Bu sayede, daha hızlı algılama yapabilmeleri, daha düşük gecikme süresine sahip olmaları ve çok aşamalı algılayıcılara kıyasla daha basit bir yapıya ve yüksek verimliliğe sahip olmaları nedeniyle gerçek zamanlı uygulamalarda tercih edilmektedir. Bu bağlamda nesne algılama alanında yaygın olarak kullanılan, gerçek zamanlı uygulamalarda hızlı ve etkili işlem gerçekleştirebilmek için farklı mimariler kullanan üç temel nesne algılama algoritma ailesi olan YOLO, SSD ve EfficientDet bu çalışmada ele alınmıştır. Bu modeller, derin sınır ağları tabanlı olmaları ve hem hız hem de doğruluk açısından sağladıkları çeşitli avantajlar nedeniyle popülerlik kazanmıştır.

Tüm modeller, PyTorch ortamında önceden eğitilmiş ağırlıklar kullanılarak test edilmiştir. Kullanılan modeller, endüstride ve akademik çalışmalarda sıklıkla tercih edilen, gerçek zamanlı nesne algılama için optimize edilmiş mimariler olmaları nedeniyle tercih edilmiştir. Bu başlık altında bu modellerin teorik temelleri, mimari farklılıkları, avantaj ve dezavantajları ve performans değerlendirmeleri ayrıntılı olarak ele alınmıştır.

3.2.1 YOLO

YOLO (Sadece Bir Kez Bak - You Only Look Once), Joseph Redmon ve arkadaşları tarafından 2015 yılında geliştirilen ve derin öğrenme tabanlı nesne algılama algoritmaları arasında hızı ve verimliliği ile öne çıkan gerçek zamanlı bir nesne algılama algoritmasıdır. [1] YOLO ilk olarak Darknet çerçevesi kullanılarak uygulanmıştır. Giriş görüntüsünü bir hücre ızgarasına böler, giriş görüntülerindeki nesnelerin sınırlayıcı kutularını ve sınıf olasılıklarını tahmin etmek için evrişimli sinir ağı (CNN) kullanan tek aşamalı bir nesne dedektörü olduğu için tüm görüntüyü tek geçişte işler ve bu da onu daha hızlı ve daha verimli hale getirir.

YOLO mimarisi GoogleNet'e benzemektedir. Şekil 3.1'de gösterildiği gibi, 24 evrişimsel katmana, dört maksimum havuzlama katmanına ve iki tam bağlı katmana sahiptir.



Şekil 3.1. YOLO mimarisi [1]

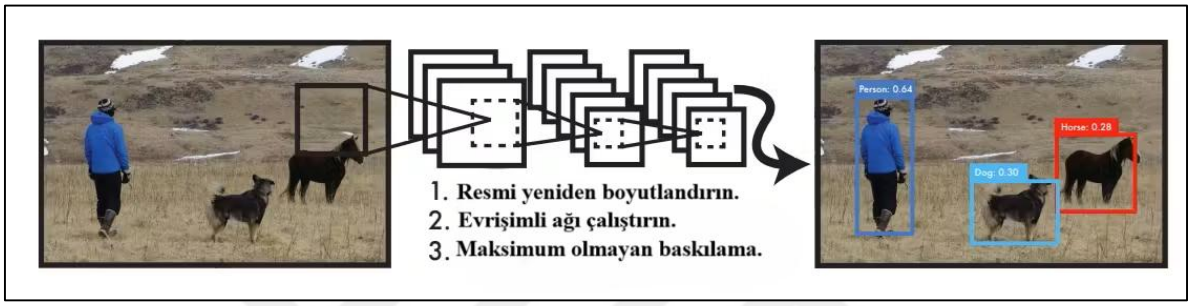
Mimarinin çalışma prensibi aşağıdaki gibidir[1]:

- Giriş görüntüsü konvolüsyon ağından geçmeden önce 448 x 448'e yeniden boyutlandırmaktadır.
- Öncelikle kanal sayısını azaltmak için 1 x 1'lik bir konvolüsyon uygulanır, ardından kübik bir çıktı üretmek için 3 x 3'lük bir konvolüsyon uygulanır.
- Perde altındaki aktivasyon fonksiyonu, doğrusal bir aktivasyon fonksiyonu kullanan son katman hariç, Doğrultulmuş Doğrusal Birim (ReLU)'dir.
- Toplu normalizasyon ve bırakma gibi bazı ek teknikler, modeli düzenler ve aşırı uyumu

önler.

YOLO algılama sistemi temelde aşağıda listelendiği ve Şekil 3.2'de gösterildiği gibi 3 adımda çalışmaktadır [1]:

- Giriş görüntüsünü 448 x 448'e yeniden boyutlandırır,
- Görüntü üzerinde tek evrişimli ağ çalıştırır,
- Ortaya çıkan algılamaları modelin güvenine göre eşikler.



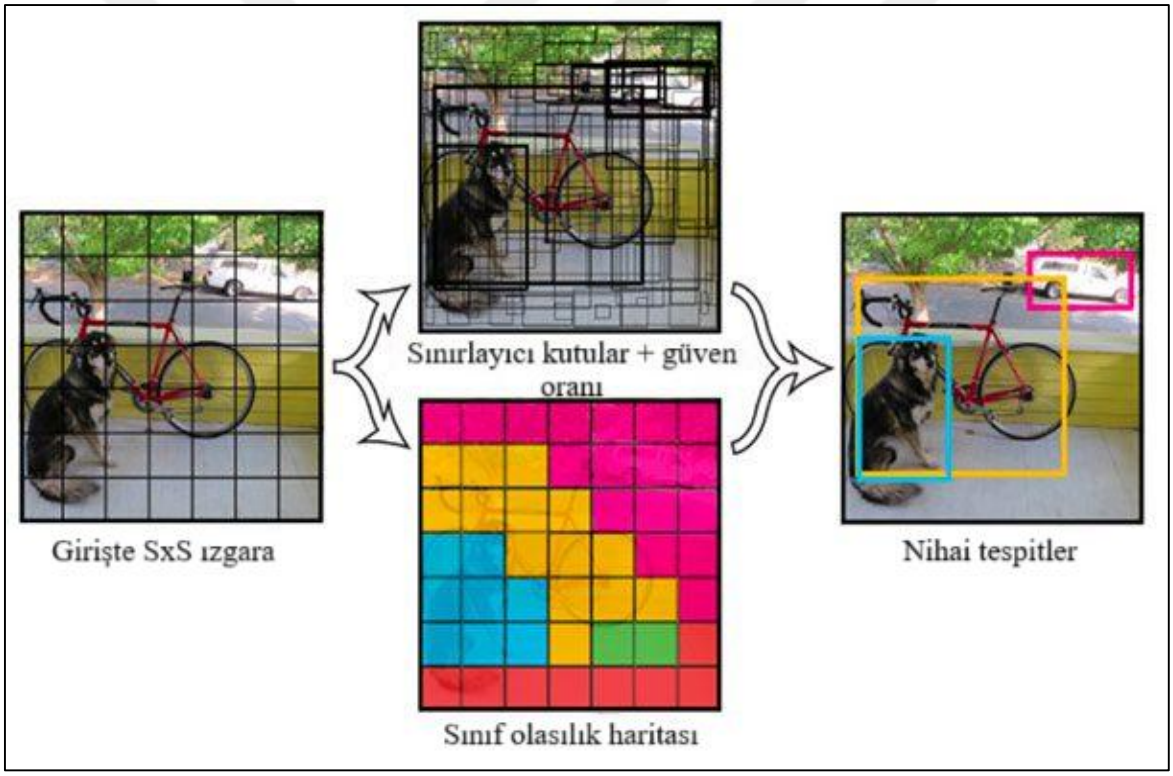
Şekil 3.2. YOLO algılama sistemi [1]

YOLO algoritması çalışma sürecinin arkasındaki temel fikir, giriş görüntüsünü bir hücre ızgarasına bölmek ve her hücre için bir nesnenin bulunma olasılığını ve nesnenin sınırlayıcı kutu koordinatlarını tahmin etmektir.

Algoritma aşağıda listelendiği ve Şekil 3.3'te gösterildiği gibi dört yaklaşıma dayanarak çalışmaktadır:

- Görüntü Girişi: Model, girdiyi sabit boyutta (örneğin, 448 × 448 piksel) normalize ederek işlemektedir.
- Grid Bölme: Girdi görüntüsü $S \times S$ şeklinde bir ızgaraya bölünmektedir. Her hücre belirli bir nesneye ait olabilecek bölgeyi temsil etmektedir.
- Sınırlayıcı Kutu Tahmini: Her hücre, belirli sayıda bounding box tahmini üretmektedir. Her tahmin, aşağıdaki bileşenleri içermektedir:
  - Nesnenin merkezi koordinatları (x, y)
  - Sınırlayıcı Kutu genişliği ve yüksekliği (w, h)
  - Nesnenin varlık olasılığı (confidence score)
  - Nesnenin ait olduğu sınıfa ilişkin olasılıklar

- Kayıp Fonksiyonu: YOLO, karesel hata (mean squared error - MSE) bazlı bir kayıp fonksiyonu kullanarak hataları optimize etmektedir. Bu sayede doğruluk eşiğinin altında kalan tahminler elenmektedir.
- Tahminlerin ardından, gereksiz sınır kutularını filtrelemek ve yalnızca en güvenilir tespitleri korumak için Maksimum Olmayan Bastırma (Non-Maximum Suppression, NMS) uygulanmakta ve en iyi tahminler seçilmektedir. Maksimum Olmayan Bastırma algoritmasının en sade hali [26] Çizelge 3.1'de verilmiştir. Şekil 3.4, birden fazla örtüşen sınırlayıcı kutu içeren bir nesne algılama modelinin tipik çıktısını ve NMS'den sonraki çıktıyı göstermektedir.
- Nihai tespit edilen nesneler çıktı olarak döndürülmektedir.



Şekil 3.3. YOLO model çalışma prensibi [1]

Çizelge 3.1. Maksimum olmayan bastırma (NMS) algoritması [26]

1	Maksimum Olmayan Bastırma (NMS) Algoritması
2	Gereklilik: Tahmin edilen sınırlayıcı kutular kümesi B, güven puanları S, IoU eşiği $\tau\tau$, güven eşiği T
3	Şunlardan emin olun: Filtrelenmiş sınırlayıcı kutular seti F olacak.
4	$F \leftarrow \emptyset$F←∅
5	Kutuları filtreleyin: $B \leftarrow \{b \in B \mid S(b) \geq T\}$B←{b∈B|S(b)≥T}
6	B kutularını güven puanalrına göre azalan sırada sıralayın
7	while $B \neq \emptyset$B≠∅ do
8	En yüksek güven puanına sahip b kutusunu seç
9	Son kutular kümesine b'yi ekle F: $F \leftarrow F \cup \{b\}$F←F∪{b}
10	b'yi kutular kümesinden çıkar B: $B \leftarrow B - \{b\}$B←B−{b}
11	for all B'de kalan tüm r kutuları için do
12	b ve r arasındaki IoU'yu hesapla: $iou \leftarrow IoU(b,r)$
13	if $iou \geq \tau$iou≥τ then
14	r'yi kutular kümesinden çıkar B: $B \leftarrow B - \{r\}$
15	end if
16	end for
17	end while



Şekil 3.4. Maksimum olmayan bastırma (NMS) (a) Birden fazla örtüşen kutu içeren bir nesne algılama modelinin tipik çıktısı (b) NMS'den sonraki çıktı [26]

YOLO algoritması diğer algoritmalarda olduğu gibi belirli avantajlara ve bazı kısıtlamalara veya zorluklara sahiptir. Nesne algılama sürecini tek başına gerçekleştirdiği için hızlıdır ve bu sayede GPU (Grafik İşlem Birimi) desteği ile yüksek FPS (Saniyedeki Kare Sayısı) hızlarında çalışabildiği için gerçek zamanlı uygulamalarda tercih edilmektedir. Bütüncül nesne algılama yaklaşımı sayesinde daha az hesaplama maliyetine sahiptir. Ancak yoğun nesne dağılımlarında sınırlayıcı kutu tahminlerinde çakışmalar yaşanabilmektedir. Yeni

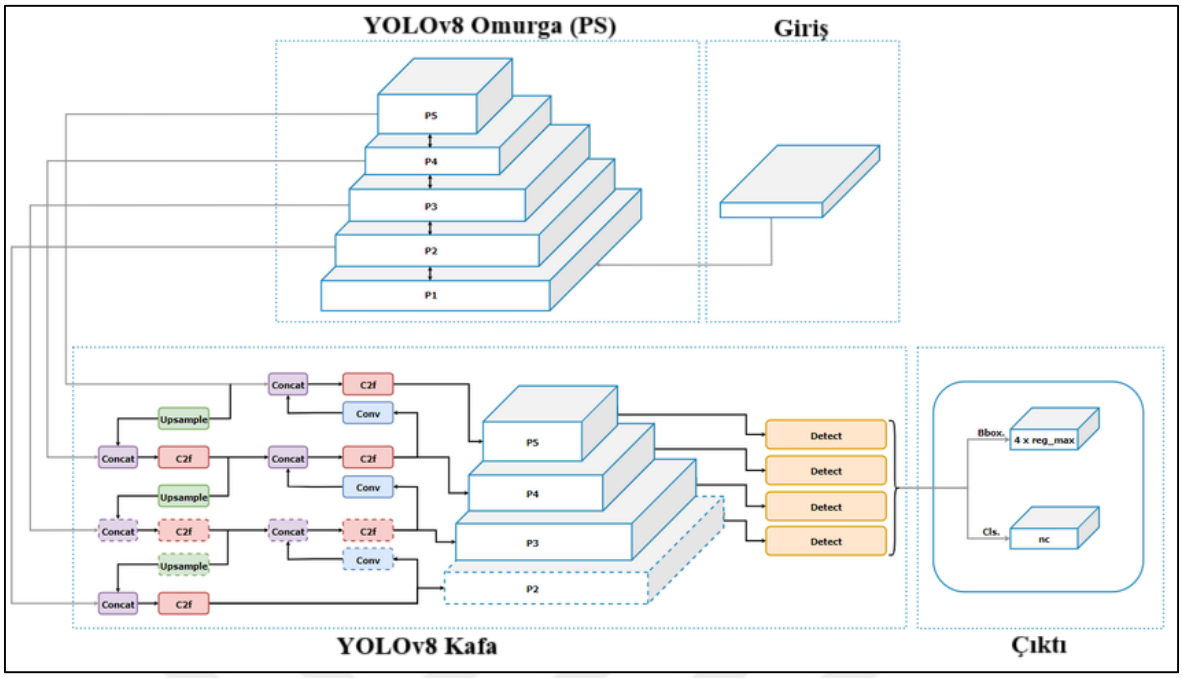
sürümlerde anchor box kullanımı ile bu problem azaltılmış ama tam olarak çözülememiştir.

İki aşamalı yöntemlerdeki gibi bölge önerisi adımına gerek olmadığı için daha verimli bir tahmin sürecine sahiptir. Yüksek doğrulukta nesne tespiti yapabildiği için özellikle büyük nesneler için yüksek kesinlik (precision) ve duyarlılık (recall) sağlamaktadır. Yüksek kesinlik (precision) ve duyarlılık (recall) ile çalışabilmesi için geniş ve dengeli bir veri seti ile eğitilmiş olması gerekmektedir. Ancak görüntü belirli bir grid sistemine bölündüğü için küçük nesneleri algılama konusunda zayıf kalabilmektedir. Bunlara ek olarak daha verimli çalışacak şekilde oprtimize edilmiş hafif bir model olması, sağladığı esneklik ile geniş bir uygulama alanına sahip olmasıyla ön plana çıkmaktadır.

YOLO, YOLOv1, YOLOv2, YOLOv3, YOLOv4, YOLOv5, YOLOv6, YOLOv7, YOLOv8, YOLOv9, YOLOv10 ve YOLO11 gibi çeşitli sürümlerde geliştirilmiştir. Her sürüm, geliştirilmiş doğruluk, daha hızlı işleme ve küçük nesnelerin daha iyi işlenmesi gibi gelişmiş özelliklerle önceki sürümün üzerine inşa edilmiştir.
YOLO, otonom araçlar ve gözetim sistemleri gibi çeşitli uygulamalarda yaygın olarak kullanılmaktadır. Ayrıca gerçek zamanlı video analitiği ve gerçek zamanlı video gözetimi gibi gerçek zamanlı nesne algılama görevlerinde de yaygın olarak kullanılmaktadır.

YOLOv8

YOLOv8, önceki YOLO sürümleri üzerine inşa edilmiş olup, çapasız algılama (anchor-free detection) yapabilmesi ve modelin evrişimsel bloklarındaki (convolutional blocks) farklılıklar gibi çeşitli iyileştirmelerle geliştirilmiştir. YOLOv8’in mimarisi, nesne tespiti görevlerinde hem hız hem de doğruluk sağlamak üzere optimize edilmiştir. Mimarı yapısı Şekil 3.5'te gösterilmektedir.



Şekil 3.5. YOLOv8 mimarisi [27]

YOLOv8'de, CNN tabanlı bir omurga (backbone) olarak CSPDarknet53 Darknet ağı kullanlarak görüntü özellikleri çıkarılmaktadır. CSPDarknet53, daha optimize ve derinleştirilmiş bir özellik çıkarma ağıdır. Bu sayede, nesne algılama modelinde kullanılan parametre sayısı azaltılarak işlem süresi iyileştirilmiştir [28]. Önceki YOLO versiyonlarından farklı olarak hiperparametre ayarlamaları 'Otomatik Çapa (AutoAnchor)' ve 'Gradyan Akış Optimizasyonu (Gradient Flow Optimization)' teknikleri ile yapılarak hiperparametre optimizasyonu sağlanmıştır. Bu özellik ile nesne algılama modelinin farklı veri setleriyle daha hızlı ve hassas şekilde uyumlanabilmesi sağlanmıştır [29].

PAN (Yol Toplama Ağı) boyun (neck) katmanı, çok ölçekli nesne tespiti gerçekleştirmek için uygulanmıştır. Bunu iki yönlü bir yaklaşım kullanarak yapmaktadır: Küçük nesneler için düşük çözünürlüklü özellikleri iyileştirecek şekilde aşağıdan yukarıya bir yol ve daha yüksek çözünürlüklü özellikleri iyileştirmek için yukarıdan aşağıya bir yol izlemektedir [30]. Bu sayede kısayol bağlantıları, bilginin ölçekler arasında iletilmesini sağlamaktadır. Sonrasında işlenmiş özellikler her iki yoldan birleştirilmektedir. Son olarak baş (head) katmanında, boyundan gelen özellik haritaları kullanılarak nesne tespiti ve sınıflandırma işlemleri yapılmakta ve bu süreçte sınır kutuları ile sınıfları tahmin etmektedir. Tüm bu adımlar sayesinde verimli bir şekilde özellik çıkarılabilmekte, çok ölçekli işleme ve nihai

yorumlama yapılmaktadır ve tüm bu gelişmeler YOLOv8'in gerçek zamalı olarak hızlı ve doğru nesne tespiti yapabilmesine olanak sağlamaktadır.

Tahminlerin ardından, gereksiz sınır kutularını filtrelemek ve yalnızca en güvenilir tespitleri korumak için Maksimum Olmayan Bastırma (Non-Maximum Suppression, NMS) uygulanmaktadır. Bu mimarı sayesinde YOLOv8, nesne tespiti görevlerinde hem hız hem de doğruluk açısından üstün bir performans sergilemektedir.

YOLOv8 mimarisi önceki YOLO versiyonlarına kıyasla birçok yenilik içermektedir. Bu yeniliklerden biri çapasız algılama (anchor-free detection) yapabilmesi ve bu sayede Maksimum Olmayan Bastırma (Non-Maximum Suppression, NMS) sürecinin hızlandırılmasıdır. YOLOv8 modelinde önceki versiyonlarda da kullanılan 'Leaky ReLU' ve 'Sigmoid Doğrusal Birim (SİLU)' fonksiyonları dinamik olarak seçilebilmektedir. Bu sayede farklı veri setlerinde modelin doğruluk ve hız dengesini en iyi şekilde sağlamak mümkünkılınmıştır [31]. Mozaik Veri artırma (MoSAIC), Kes-Karıştır Yöntemi (CutMix) ve albüminasyonlar gibi gelişmiş veri artırma tekniklerini kullanılarak modelin genelleme yeteneğini artırılmıştır. Böylece, veri setine bağlı overfitting problemleri minimize edilmiştir [32].

Daha yüksek doğruluk oranı (mAP), küçük nesneleri daha iyi algılama kapasitesi, düşük gecikme süresi ve yüksek FPS değeri ile gerçek zamanlı uygulamalar için uygunluk göstermesi, otomatik hiperparametre ayarlamaları sayesinde farklı veri setlerine hızlı uyum sağlayabilmesi ve hafif ve optimize edilmiş model yapısı ile düşük kaynak tüketimine sahip olması mimari yenilikler ile birlikte YOLOv8'in sağladığı avantajlar arasında gösterilebilmektedir. Öte yandan diğer YOLO versiyonlarına kıyasla daha karmaşık bir eğitim sürecine sahip olması ve bazı uygulamalar için hala Faster R-CNN gibi iki aşamalı modellerin doğruluğuna ulaşamaması ise YOLOv8'in dezavantajları arasında gösterilebilmektedir.

Çizelge 3.2, COCO veri kümesindeki çeşitli YOLOv8 modelleri için performans ölçümlerini sunmaktadır. YOLOv8 modelleri, boyutlarını ve karmaşıklıklarını temsil eden farklı harflerle (n, s, m, l ve x) gösterilmektedir. Sağlanan ölçümler arasında giriş boyutu, farklı IoU eşiklerinde ortalama hassasiyet (AP), Açık Sinir Ağı Değişim Formatı (ONNX) kullanan CPU (Merkezi İşlem Birimi)'daki gecikme, TensorRT kullanan NVIDIA

A100'deki gecikme ve saniye başına Kayan Nokta İşlem Sayıları (FLOP) bulunmaktadır.

Çizelge 3.2. COCO veri kümesinde YOLOv8 modelleri için performans ölçümleri [33]

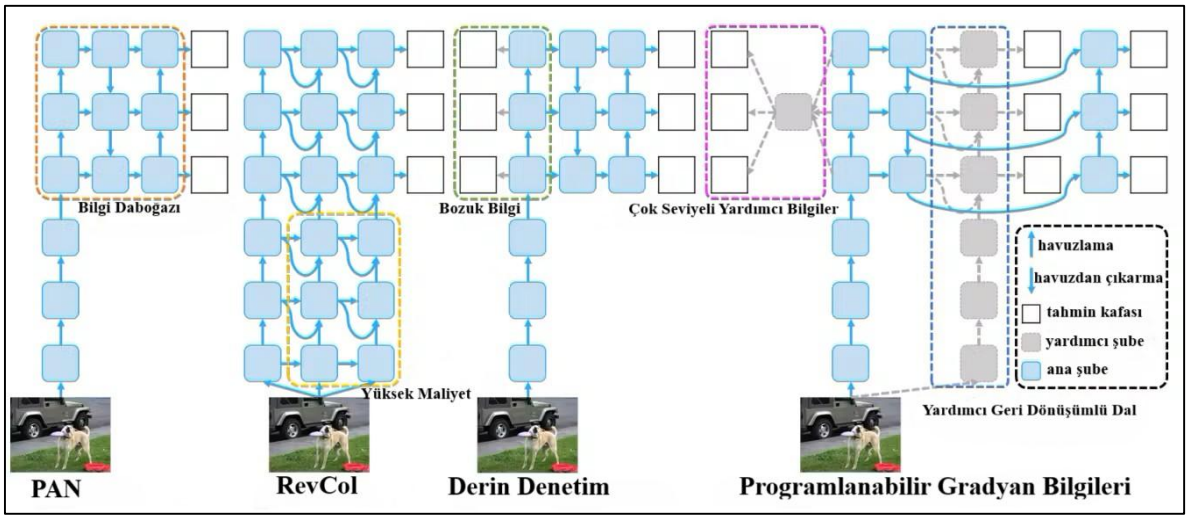
Model	Giriş Boyutu	COCO AP (val)	CPU Gecikmesi ONNX (ms)	A100 TensorRT Gecikmesi (ms)	FLOPs (B)
YOLOv8n	640	%37,3	80,4	0,99	8,7
YOLOv8s	640	%44,9	128,4	1,20	28,6
YOLOv8m	640	%50,2	234,7	1,83	78,9
YOLOv8l	640	%52,9	375,2	2,39	165,2
YOLOv8x	640	%53,9	479,1	3,53	257,8

YOLOv8, önceki YOLO sürümlerden miras aldığı güçlü yönleri daha da geliştirerek, nesne algılama alanında önemli bir yer almaktadır. Özellikle gerçek zamanlı nesne algılama çalışmalarında ön plana çıkmaktadır. Gelişmiş optimizasyon teknikleri, veri artırma yöntemleri ve yeni nesil model yapısıyla, hem endüstriyel hem de akademik araştırmalarda geniş bir kullanım alanına sahip olmaktadır.

YOLOv9

YOLOv9, Programlanabilir Gradyan Bilgisi (PGI) ve Genelleştirilmiş Verimli Katman Toplama Ağı (G-ELAN) gibi yeniliklerle geliştirilmiştir. PGI, veri kaybını en aza indirirken doğru gradyan güncellemeleri sağlamaktadır. Bu iyileştirme, Şekil 3.6'da gösterildiği gibi geri dönüşüm ve çok seviyeli bilgi kavramlarını destekleyen mimariler kullanılarak elde edilmekte ve böylece bilgi darboğazı sorununa etkili bir çözüm sağlamaktadır.

YOLOv9, CNN tabanlı backbone'ların kullanıldığı önceki modellerdeki gibi konvolüsyon katmanlarının yerel özellik çıkarımı yeteneğini korurken, Verimli Katman Toplama Ağı (ELAN) mimarisinin, G-ELAN mimarisine genişletilmesi sayesinde gradyan yolu planlamasıyla hafif modelleri optimize etmek ve flop oranını en aza indirilip doğruluğu en üst düzeye çıkarmaktadır. Ayrıca Swin Transformer tabanlı özellik çıkarma mekanizmaları ile uzun menzilli bağlantıları (long-range dependecies) öğrenme kapasitesini artırmaktadır [34]. Evrişimsel Gömme Katmanları (Convolutional Embedding Layers) ile düşük seviyeli detay çıkarımı ve Çok Ölçekli Özellik Füzyonu (Multi-Scale Feature Fusion) ile farklı çözünürlüklerde verimli öğrenme yapılabilmektedir. Öne çıkan bu gelişmeler ile birlikte nesne algılamada daha yüksek doğruluk sağlamaktadır.



Şekil 3.6. YOLOv9 Mimarisi - PGI esas olarak üç bileşenden oluşur (1) ana dal: çıkarım için kullanılan mimari (2) yardımcı geri dönüşümlü dal: geri iletim için ana dalı sağlamak üzere güvenilir gradyanlar üretir ve (3) çok seviyeli yardımcı bilgi: ana dalın planlanabilir çok seviyeli anlamsal bilgiyi öğrenmesini kontrol eder [13]

YOLOv9, önceki sürümlerden farklı olarak Verimli Çift Yönlü Özellik Piramidi Ağı (Efficient Bidirectional Feature Pyramid Network - BİFPN) kullanmaktadır. Özellik Piramidi Ağı (Feature Pyramid Network - FPN) çok ölçekli nesne algılama görevlerinde yaygın olarak kullanılan bir yapıdır. BİFPN ile ağırlıklı özellik birleşimi (weighted feature fusion), adaptif özellik ölçekleme ile küçük nesnelerin daha iyi tespiti ve düşük hesaplama yükü ile optimize edilmiş bilgi akışı gibi öne çıkan özellikler YOLOv9 modeline dâhil edilmiştir. Bu sayede YOLOv9, farklı çözünürlüklerde gelen özellik haritalarını daha verimli bir şekilde birleştirerek özellikle küçük nesneleri daha doğru bir şekilde tespit edebilmektedir [35].

YOLOv9 Algılama Transformatörü (Detection Transformer – DETR) ve Sparse (Seyrek) R-CNN gibi çapa içermeyen (anchor- free) yaklaşımlar kullanılarak dinamik kutu tahminleri yapmaktadır [36]. Daha az hiperparametreye ihtiyaç duymak, karmaşık anchor box eşleme süreçlerini ortadan kaldırmak ve özellikle yoğun sahnelerde daha hassas tahmin yapmak gibi avantajlar çapa içermeyen (anchor- free) tahmin yapma yaklaşımlarının avantajları arasında gösterlebilmektedir. Bu sayede YOLOv9, nesne algılama sürecini daha verimli hale getirerek daha az bellek tüketimi ve daha hızlı çıkarım süresi gibi avantajlar sunmaktadır.

Nesne algılama görevlerinde doğruluk ve hız dengesini sağlamak için SİLU ve Gauss Hatalı Doğrusal Birim (Gaussian Error Linear Unit - GELU) gibi öğrenme sürecindedaha stabil

gradyan akışını sağlayan aktivasyon fonksiyonlarını, Kesinlik – Farkında Minimizasyon SAM (Sharpness – Aware Minimization - SAM) gibi modelin genelleme yeteneğini artıran optimizasyon algoritmasını ve Deformable Attention (Deforme Edilebilir Dikkat) ile ilgili alanlara odaklanarak daha verimli nesne tespiti sağlayan mekanizmaları kullanmaktadır [35].

Çizelge 3.3, COCO veri kümesindeki çeşitli YOLOv9 modelleri için performans ölçümlerini sunmaktadır. YOLOv9 modelleri, boyutlarını ve karmaşıklıklarını temsil eden farklı harflerle (t, s, m, c ve e) gösterilmektedir. YOLOv9e'nin hem CPU hem de GPU'daki çıkarım süresi, çok sayıda parametresine rağmen rekabetçi olmaya devam etmektedir. Bu verimlilik, doğruluğuyla birleştiğinde, YOLOv9e'yi sağlam performans gerektiren kritik uygulamalar için ideal hale getirmektedir. Öte yandan YOLOv9c, nispeten düşük bir hesaplama yükü korurken doğrulukta önemli kazanımlar sunarak orta yol sağlar ve onu çeşitli nesne algılama görevleri için çok yönlü bir seçenek haline getirmektedir. Daha hafif modeller olan YOLOv9t ve YOLOv9s, sınırlı hesaplama kapasitesine sahip cihazların bile temel doğruluktan ödün vermeden gerçek zamanlı algılama performansına ulaşmasını sağlamaktadır. Bu nedenle YOLOv9'un modern nesne algılama zorluklarının çeşitli ihtiyaçlarını karşılamak üzere uyarlanmış ölçeklenebilir çözümler sunmaktadır [35].

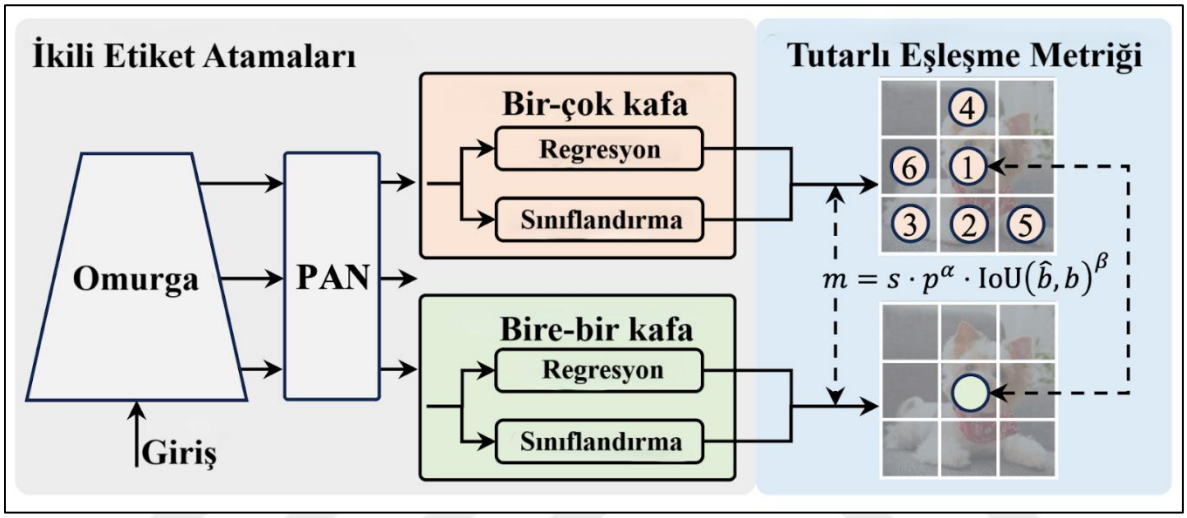
Çizelge 3.3. COCO veri kümesinde YOLOv9 modelleri için performans ölçümleri [35]

Model	Parametreler (Million)	Doğruluk (mAP@0,5)	Çıkarım Süresi (CPU ms)	Çıkarım Süresi (GPU ms)
YOLOv9t	2,0	%38,3	35	6,0
YOLOv9s	7,2	%46,8	80	6,5
YOLOv9m	20,1	%51,4	200	8,0
YOLOv9c	25,5	%53,0	350	9,0
YOLOv9e	58,1	%55,6	500	11,5

YOLOv9, önceki YOLO sürümlerden miras aldığı güçlü yönleri daha da geliştirerek, nesne algılama alanında önemli bir yer almaktadır. Özellikle gerçek zamanlı nesne algılama çalışmalarında ön plana çıkmaktadır. Gelişmiş optimizasyon teknikleri, transformer ve CNN tabanlı hibrit mimarisi, BiFPN feature fusion yöntemi ve anchor-free nesne tahmin mekanizması ile, hem endüstriyel hem de akademik araştırmalarda geniş bir kullanım alanına sahip olmaktadır.

YOLOv10

YOLOv10, genel mimari açısından YOLOv6'ya benzerdir ve mimarisi Şekil 3.7'de gösterildiği gibidir. Wang ve diğerleri tarafından 2024'te tanıtılan YOLOv10, YOLO serisinde önemli bir ilerlemeyi temsil etmekte ve hem mimari terimler hem de eğitim protokollerinde bir dizi optimizasyon yoluyla nesne algılamada verimlilik ve doğruluğu dengelemeye odaklanmaktadır.



Şekil 3.7. YOLOv10 mimarisi [37]

Omurga olarak CSPNet'in geliştirilmiş bir versiyonu kullanılmaktadır. Bu omurga (backbone), önceki YOLOv9 sürümünden evrimleşerek daha verimli bir özellik çıkarma mekanizması geliştirmiştir. Yeni mimariye önemli nesne bölgelerini daha iyi belirlemek için dikkat mekanizmaları eklenmiştir. Hesaplama yükünü azaltırken doğruluk oranını artıran ayrık evrişim katmanları kullanılmıştır. Ayrıca, çok ölçekli nesne algılama yeteneklerini iyileştiren optimize edilmiş bir FPN yapısı entegre edilmiştir.

Çok ölçekli özellik füzyonu için PAN (Yol Toplama Ağı) katmanlarını içeren bir boyun yapısına sahiptir. Eğitim sırasında nesne başına birden fazla tahmin üreterek zengin denetleyici sinyaller sağlamakta ve öğrenme doğruluğunu artırmaktadır. Geliştirilmiş bir baş yapısına sahiptir. İkili baş yapısı sırasıyla bire çok ve bire bir eşleştirme yapısına dönüştürülmekte ve bu baş mimarisiyle çıkarım sırasında nesne başına tek bir en iyi tahmin üreterek NMS ihtiyacını ortadan kaldırmaktadır, böylece gecikmeyi azaltmakta ve verimliliği artırmaktadır [33].

YOLOv10, performansı artırırken hesaplama yükünü en aza indirmek için model mimarisinin çeşitli bileşenlerini optimize etmektedir. Yapılan optimizasyonlardan biri hafif sınıflandırma başlıkları (lighweight classification head) kullanarak hesaplama yedekliliğini azaltmaktır. Bu sayede model aşırı hesaplama maliyeti olmadan doğru tahminler yapabilmektedir. Mekânsal kanal ayrıştırılmış örnekleme (spatial-channel decoupled downsampling) ile örnekleme sırasında mekânsal ve kanal bilgilerini ayırarak özellik çıkarma süreci optimize edilmiştir. Sıralama kılavuzlu blok tasarımı (rank-guided block design), genel mimariyi düzene sokarak hesaplama verimliliğini artırmaktadır. Büyük çekirdek evrişimleri (large-kernel convolutions), modelin daha büyük mekânsal bölgelerdeki ayrıntılı özellikleri yakalama yeteneğini geliştirerek modelin bir görüntüdeki nesnelerin bağlamını daha iyi anlamasını sağlamaktadır. Kısmi özel dikkat modülü (partial self-atention module) sayesinde minimum ek hesaplama maliyetleri ile doğruluk artırılmıştır [33].

YOLOv10, genel performansı artırmak için geliştirilmiş küçük nesne algılama, azaltılmış yanlış pozitifler ile tahmin güvenilirliğini artırma, güven puanı iyileştirmesi gibi gelişmiş teknik yöntemleri de barındırmaktadır. Bunlara ek olarak ölçeklenebilirliği ile farklı hesaplama gereksinimlerini ve kullanım durumlarını karşılamak için önceki sürümlerdeki gibi birden fazla model boyutu (n, s, m, l, x) sunmaktadır [33].

Çizelge 3.4, COCO veri kümesindeki çeşitli YOLOv10 modelleri için performans ölçümlerini sunmaktadır. YOLOv10 modelleri, boyutlarını ve karmaşıklıklarını temsil eden farklı harflerle (n, s, m, l ve x) gösterilmektedir. YOLOv10s, model karmaşıklığı ve performans arasında bir denge sunarak, düşük gecikmeyi korurken yüksek doğruluk sağlamaktadır. YOLOv10m, 4,74 ms'lik bir gecikmeyle, daha küçük modellere kıyasla doğrulukta önemli bir iyileştirme sağlarken, yine de nispeten düşük çıkarım süreleri sağlamaktadır. YOLOv10-L ve YOLOv10-X, YOLOv10 varyantları arasında en yüksek performansı göstermektedir. Bu modeller en iyi doğruluğu sağlayarak, hassasiyetin kritik olduğu görevler için uygun hale gelmektedir [33].

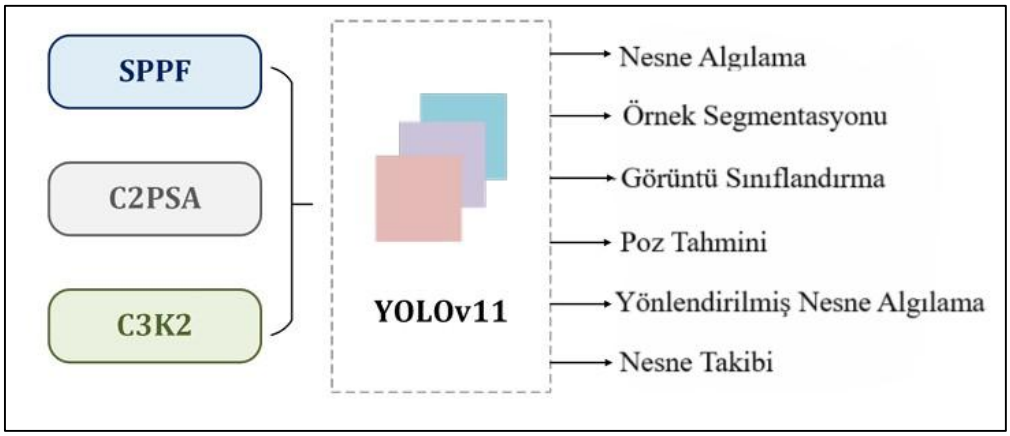
Çizelge 3.4. COCO veri kümesinde YOLOv10 modelleri için performans ölçümleri [37]

Model	Giriş Boyutu	COCO AP (val)	Gecikme Süresi (ms)	FLOPs (B)
YOLOv10n	640	%38,5	1,84	6,7
YOLOv10s	640	%46,3	2,49	21,6
YOLOv10m	640	%51,1	4,74	59,1
YOLOv10l	640	%53,2	7,28	120,3
YOLOv10x	640	%54,4	10,70	160,4

YOLOv10, önceki YOLO sürümlerden miras aldığı güçlü yönleri daha da geliştirerek, nesne algılama alanında önemli bir yer almaktadır. YOLOv10 varyantları diğer önde gelen modellerle karşılaştırıldığında hesaplama verimliliği ve doğruluk ölçümlerinde daha ön sıralarda yer aldığını göstermektedir. Performansları, özellikle gerçek zamanlı nesne algılama ve yüksek doğruluğun önemli olduğu modern bilgisayarlı görü uygulamaları için onları son teknoloji çözümleri haline getirmektedir.

YOLO11

YOLO11, önceki modellerin üzerine inşa edilmiş olup, mimarı ve eğitim aşamalarında önemli iyileştirmeler, daha yüksek doğruluk, daha düşük gecikme süresi ve optimize edilmiş hesaplama gereksinimleri sunmaktadır. Sınırlayıcı kutu regresyonu ve nesne sınıflandırma görevlerini aynı anda ele alan birleşik bir sinir ağı mimarisi sunarak nesne algılamada önemli bir ilerleme oluşturmuştur. Bu iyileştirmeler, 2x2 Çekirdekli Çapraz Evrişim Bloğu (Cross-Stage Partial Kernel Size 2 - C3k2) bloğu, Hızlı Uzamsal Piramit Havuzlama (Spatial Pyramid Pooling – Fast - SPPF), ve Paralel Uzamsal Dikkat Evrişimsel Blok (Parallel Spatial Attention Convolutional Block - C2PSA) bileşenleri üzerinden gerçekleştirilmiştir. YOLO11'in mimarı yapısı Şekil 3.8'de gösterildiği gibi düzenlenmiştir ve bu bileşenlerle hız korunurken performans da artırılmıştır.



Şekil 3.8. YOLO11 mimarisi [38]

Çeşitli çözünürlükte özellik haritası oluşturmak için evrişimsel katmanları ve özel blokları istifleyerek giriş görüntüsünden birçok ölçekte özellik çıkaran omurgada iki küçük evrişim kullanılmaktadır. C3k2 bloğu, önceki modellerde kullanılan Birleşimli Kısmi Aşama Geçişi (C2f) bloğunun yerine geçmiştir ve omurganın (backbone) farklı aşamalarında özellik çıkarımını geliştirmek için kullanılmıştır. C3k2'deki 'k2' daha küçük bir çekirdek boyutunu belirtmektedir. Bu sayede performans korunurken daha hızlı işlem yapılmaktadır [19]. SPPF modülü, önceki modellerde bulunan SPP modülünün optimize edilmiş bir versiyonudur ve bir görüntünün farklı bölgelerinden farklı ölçeklerde özellikleri birleştirmek üzere tasarlanmıştır. Bu yapı sayesinde küçük nesneleri yakalama kapasitesi geliştirilmiştir. YOLO11, önceki sürümlerden SPPF (Spatial Pyramid Pooling-Fast) bloğunu korumakta ancak ardından mekânsal dikkat ile yeni bir C2PSA (Cross Stage Partial) bloğu sunmaktadır [38]. C2PSA bloğu, modelin bir görüntüdüki daha küçük veya kısmen örtülü nesnelere odaklanma becerisini artıran bir yapı sunmaktadır.

YOLO11'de boyundaki C2f bloğu C3k2 bloğuyla değiştirilmiştir. C3k2 bloğu, boyunun farklı ölçeklerdeki özellikleri birleştirerek bunları tahmin için başa iletme işlemini daha hızlı ve daha verimli yapacak şekilde tasarlanmıştır. Ayrıca C3k2 bloğu, yukarı örnekleme ve birleştirme adımlarından sonra yer almaktadır ve bu sayede hız ve performans artırımı sağlanmıştır. Ek olarak boyun kısmında C2PSA modülü ile mekânsal dikkatte artan odaklanma sağlanmıştır. Bu dikkat mekanizması, görüntüdeki temel bölgelere yoğunlaşmayı sağlayarak daha küçük veya kısmen örtülü nesneler için daha doğru tespit yapılabilmesini sağlamıştır [38].

Mimarinin son bileşeni olan baş kısmında boyundan geçirilen özellik haritaları işlenip nihai tahminleri içeren sınırlayıcı kutular ve sınıf etiketleri çıkarılırken birden fazla C3k2 bloğu kullanılmaktadır. C3k2 bloğu, c3k parametresinin değerine bağlı olarak esneklik sağlamaktadır. c3k parametresinin 'false' olması durumunda C2f bloğuna benzer şekilde, 'true' olması durumunda ise C3 modülüne benzer şekilde davranmaktadır. Bu sayede iki küçük evrişimin kullanılmasıyla daha hızlı özellik çıkarımı yapılabilmektedir ve CSP darboğazının daha kompakt bir versiyonu olmasından ötürü mimariyi eğitilebilir parametre sayısı açısından daha verimli hale getirmektedir [38]. C3k2 bloklarından sonra birkaç tane Evrişi, Normalizasyon ve Aktivasyon Bloğu (Convolutıon-BatchNorm-SiLU - CBS) katmanı yer almaktadır. Bu katmanlar, doğru nesne algılama için ilgili özelliklerin çıkarılması, toplu normalleştirme ile veri akışının sabitlenerek normalleştirilmesi ve doğrusal olmayanlık için Sigmoid Doğrusal Birim (SiLU) aktivasyon fonksiyonunun kullanılması işlemleri ile özellik haritalarının çıkarımında model performansını iyileştirmektedir [39]. CBS blokları, nihai özellik haritalarının sınırlayıcı kutu ve sınıflandırma tahminleri için son evrişimsel katmanlar ve algılama katmanına iletilmesini sağlamaktadır. Her algılama dalı Conv2D katmanıyla son bulmaktadır. Son algılama katmanı, görüntüdeki nesneleri yerelleştirmek için sınırlayıcı kutu koordınatları, nesnelerin varlığını gösteren nesnellik puanları ve algılanan nesnenin sınıfını belirlemek için sınıf puanlarını içeren sınıf tahminlerini birleştirmektedir [38].

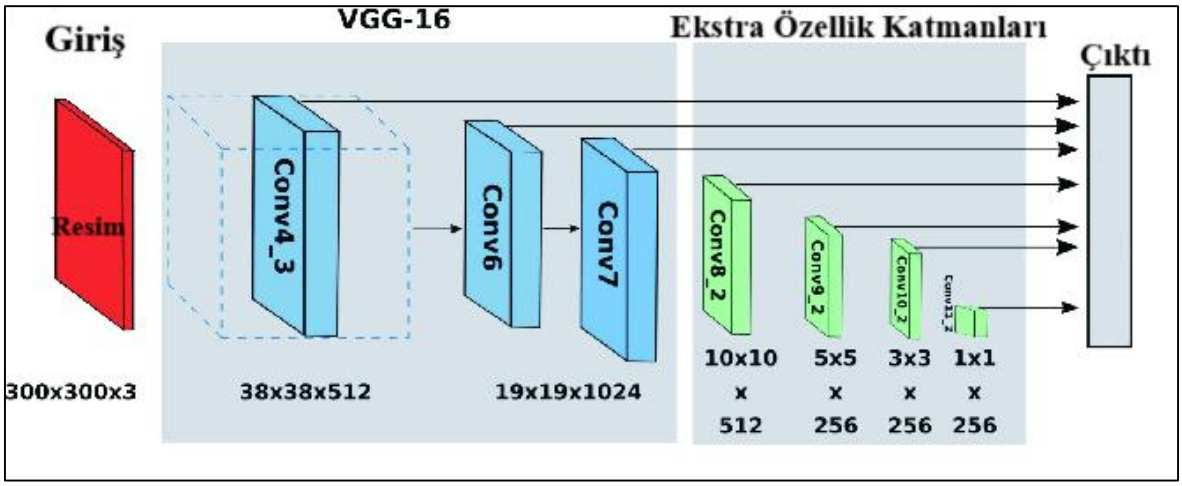
Çizelge 3.5, COCO veri kümesindeki çeşitli YOLO11 modelleri için performans ölçümlerini sunmaktadır. YOLO11 modelleri, boyutlarını ve karmaşıklıklarını temsil eden farklı harflerle (n, s, m, l ve x) gösterilmektedir. YOLO11x, 11ms gecikmede yaklaşık %54,5 mAP50:95 elde ederek önceki tüm YOLO sürümlerini geride bırakmaktadır. YOLO11m, önceki nesillerden daha büyük modellerle karşılaştırılabilir doğruluk elde ederek daha iyi verimlilik gösterirken önemli ölçüde daha az işlem süresi gerektirmektedir. YOLO11 modelleri daha yüksek düşük gecikme ile koruyarak performans sıçraması yapmaktadır. Bu, hem hızın hem de doğruluğun kritik olduğu gerçek zamanlı uygulamalar için önemli bir ilerlemeyi temsil etmektedir. Çizelge 3.5 ayrıca model varyantları arasında daha iyi ölçekleme özellikleri gösterir ve bu da önceki sürümlere kıyasla ek hesaplama kaynaklarının daha verimli kullanıldığını göstermektedir. Tüm bunlara dayanarak YOLO11'in önceki sürümlerle kıyaslandığında daha yüksek doğruluk sağlaması, daha hızlı çalışması ve daha düşük bellek kullanması gibi yeniliklerle ön plana çıktığı görülmektedir.

Çizelge 3.5. COCO veri kümesinde YOLO11 modelleri için performans ölçümleri [40]

Model	Giriş Boyutu	COCO AP (val)	Hız CPU ONNX (ms)	Hız T4 TensorRT10 (ms)	Params (M)	FLOPs (B)
YOLO11n	640	%39,5	56,1 ± 0,8	1,5 ± 0,0	2,6	6,5
YOLO11s	640	%47,0	90,0 ± 1,2	2,5 ± 0,0	9,4	21,5
YOLO11m	640	%51,5	183,2 ± 2,0	4,7 ± 0,1	20,1	68,0
YOLO11l	640	%53,4	238,6 ± 1,4	6,2 ± 0,1	25,3	86,9
YOLO11x	640	%54,7	462,8 ± 6,7	11,3 ± 0,2	56,9	194,9

3.2.2 SSD

SSD (Single Shot Multi Box Detector), 2016 yılında Liu ve arkadaşları tarafından önerilmiş olup, tek aşamalı nesne algılama mimarileri arasında yer almaktadır. Çok ölçekli özellik haritaları ve farklı boyutlardaki çerçeveleri eş zamanlı olarak algılama yeteneği ile öne çıkmaktadır. Yüksek hız ve doğruluk dengesini koruyarak, özellikle gerçek zamanlı uygulamalar için uygun bir model olarak kabul edilmektedir [2]. SSD'nin mimarisi, ağın tek bir ileri geçişinde nesne algılaması gerçekleştirmek üzere tasarlanmıştır. Giriş görüntüsünden farklı ölçeklerde özellik haritaları oluşturarak, her bir özellik haritasında doğrudan nesne tespiti yapmak üzerine kurgulanmış bir çalışma prensibine sahiptir. Mimari yapı genel olarak Şekil 3.9'da gösterildiği gibidir. SSD, bu mimari yapı ile hız ve doğruluk arasında bir denge sağlamaktadır.



Şekil 3.9. SSD mimarisi [41]

SSD, CNN (Convolutıonal Neural Network) tabanlı bir omurga ağı ile başlamaktadır. Orijinal SSD modeli, VGG16 tabanlı bir omurga ağı kullanarak giriş görüntülerinden

özellikleri çıkarmaktadır. Ancak günümüzde, MobileNet [42] ve ResNet [43] gibi daha hafif ve güçlü omurga ağları da SSD ile birlikte kullanılmaktadır. Bu omurga ağları, büyük veri kümelerinde önceden eğtilmektedir ve bu sayede nesne algılama için gerekli olan zengin hiyerarşik özelliklerin yakalanması sağlanmaktadır.

Mimari yapı, sınırlayıcı kutuları ve sınıf olasılıklarını tahmin etmek için bir dizi evrişimli katman ekleyerek birden fazla sınırlayıcı kutuyu tahmin etmeye yardımcı olan bir yapı içermektedir. Bu sayede farklı seviyelerde farklı çözünürlüklerde özellik haritaları üreterek her bir haritada tahmin ve nesne tespiti yapabilmektedir [44]. Bu çok ölçekli yaklaşım, küçük nesnelerin yüksek çözünürlüklü haritalar kullanılarak, büyük nesnelerin ise düşük çözünürlüklü haritalar kullanılarak algılanması sağlanmaktadır.

SSD, sınırlayıcı kutuları ve sınıf olasılıklarını tahmin etmek için her özellik haritasının üzerine bir dizi evrişimsel katman eklenerek oluşturulan bir tahmin başlığı yapısına sahiptir. Tahmin başlığında yer alan her evrişimsel katman, karşılık gelen özellik haritasının içindeki farklı uzamsal konumlarda bir dizi sınırlayıcı kutuyu tahmin etmekten sorumludur [45]. SSD, her özellik haritasındaki her bir hücreye belirli sayıda farklı boyut ve en-boy oranlarına (aspect ratios) sahip varsayılan çerçeve (default boxes) atamaktadır. Bu varsayılan kutular, doğru sınırlayıcı kutuların tahmin edilebilmesi için gerekli ön bilgileri sağlamaktadır. Model, her çerçeve için bir sınıf tahmini ve konum tahmini yapmaktadır [2].

Model, iki temel bileşenden oluşan bir kayıp fonksyonu kullanmaktadır. Bu kayıp fonksiyonunda kullanılan bileşenler eğitim aşamasında parametrelerin optimize edilmesi maksadıyla kullanılmaktadır. Konum kaybı (Localization Loss), gerçek nesne konumu ile tahmin edilen çerçeve arasındaki farkı hesaplamaktadır. Sınıflandırma kaybı (Confidence Loss) ise nesnenin doğru sınıfını tahmin etem için çapraz entropi kaybı (cross-entropy loss) methodunu kullanmaktadır [45].

SSD, tahminler yapıldıktan sonra, tespit edilen çerçeveler arasındaki örtüşmeyi hesaplamak için IoU (Intersection over Union) metriğini kullanmaktadır ve yalnızca en güvenilir algılamaları tutmak için Maksimum Olmayan Bastırma (NMS – Non-Maximum Suppression) uygulamaktadır. Böylece modelin daha yüksek doğruluklu sonuçlar üretmesini sağlamaktadır [2].

Çizelge 3.6 ve Çizelge 3.7, COCO veri kümesindeki çeşitli SSD modelleri için performans ölçümlerini sunmaktadır. Çizelge 3.6 incelendiğinde SSD300'ün hem mAP@0,5 hem de mAP50:95'te Faster R-CNN'den daha iyi olduğu, ION ve Faster R-CNN ile benzer bir mAP@0,75'e sahip olduğu, ancak mAP@0,5'te daha kötü olduğu görülmektedir. Görüntü boyutunun 512 × 512'ye çıkarılması ile birlikte, SSD512'nin her iki kriterde de Faster R-CNN'den daha iyi sonuç ürettiği görülebilmektedir. Ayrıca büyük nesneler için çok daha iyi AP (%4,8) ve AR (%4,6) olduğunu, ancak küçük nesneler için AP (%1,3) ve AR (%2,0)'de nispeten daha az iyileştirmeye sahip olduğu görülmektedir. ION ile karşılaştırıldığında, büyük ve küçük nesneler için AR'deki iyileştirmenin daha benzer olduğu görülmektedir (%5,4'e karşı %3,9) [2].

Çizelge 3.6. COCO test-dev2015 SSD modelleri için performans ölçümleri 1 [2]

Method	Veri	Ortalama Kesinlik, IoU:			Ortalama Kesinlik, Alan:		
		0,5:0,95	0,5	0,75	S	M	L
Hızlı	train	19,7	35,9	-	-	-	-
Hızlı	train	20,5	39,9	19,4	4,1	20,0	35,8
Daha Hızlı	trainval	21,9	42,7	-	-	-	-
ION	train	23,6	43,2	-	6,4	24.1	38,8
Daha Hızlı	trainval	24,2	45,3	23,5	7,7	26,4	37,1
SSD300	trainval35k	23,2	41,2	23,4	5,3	23,2	39,6
SSD512	trainval35k	26,8	46,5	27,8	9,0	28.9	41,9

Çizelge 3.7. COCO test-dev2015 SSD modelleri için performans ölçümleri 2 [2]

Method	Veri	Ortalama Duyarlılık, #Dets:			Ortalama Duyarlılık, Alan:		
		1	10	100	S	M	L
Hızlı	train	-	-	-	-	-	-
Hızlı	train	-	-	-	7,3	32,1	52,0
Daha Hızlı	trainval	21,3	29,5	30,1	-	-	-
ION	train	-	-	-	10,1	37,7	53,6
Daha Hızlı	trainval	23,0	32,0	33,5	12,0	38,5	54,4
SSD300	trainval35k	22,5	33,2	35,3	10,2	36,3	56,5
SSD512	trainval35k	24,8	37,5	39,8	14,0	43,5	59,0

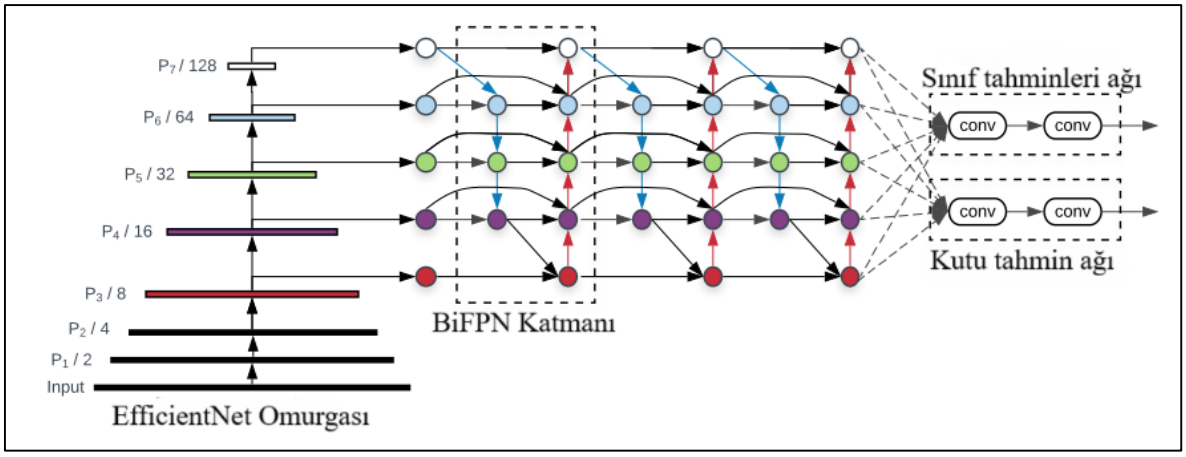
SSD, gerçek zamanlı nesne algılama işlemlerinde sağladığı yüksek performans, çok ölçekli tahmin yapabilmesi, tek aşamalı yapıda olduğu için düşük hesaplama maliyetine sahip olması ve mobil cihazlarda çalıştırılabiliyor olması gibi avantajlarıyla ön plana çıkmaktadır. Ancak, küçük nesnelerin algılanmasındaki performans eksikliği, varsayılan çerçeve seçimlerinin yeterli olamaması sebebiyle manuel ayarlama yapmanın gerekmesi ve yeni

modellerin sunduğu gelişmiş özellikler nedeniyle bazı alanlarda SSD'nin yerini daha modern modeller almaktadır.

3.2.3 EfficientDet

Google Brain tarafından geliştirilen EfficientDet, doğruluk ve hızı dengelerken hesaplama maliyetini düşüren ve ölçeklenebilirlik gibi faktörleri iyileştirerek yüksek doğruluk elde etmiş bir nesne algılama algoritmasıdır. EfficientDet, hem doğruluğu hem de verimliliği optimize etmek amacıyla yeni bir ortak ölçekleme yöntemi kullanmaktadır.

EfficientDet'in mimari yapısı genel olarak Şekil 3.10'da gösterildiği gibidir. EfficientDet, sınıflandırma için omurga ağı olarak ImageNet görüntü veritabanında önceden eğitilmiş olan EfficientNet'i kullanmaktadır ve EfficientDet, EfficientNet üzerine kurulmuştur. Belirli granülaritelere sahip görüntü parçaları bir araya getirilmekte ve karşılaştırılmakta ve özellikler, BiFPN özellik birleştirme katmanından geçirilerek üretilmektedir. BiFPN katmanı, omurga ağından P3P4P5P6P7 düzeyindeki özellikleri alan ve tekrar tekrar yukarıdan aşağıya ve aşağıdan yukarıya özellik birleştirme uygulayan çift yönlü bir özellik birleştirme işlemi uygulayan özellik ağı olarak kullanılmaktadır. Bu katmanda birleştirilen özellikler, sırasıyla nesne sınıfı ve sınırlayıcı kutu tahminleri üretmek için algılama kafasına iletilmektedir.



Şekil 3.10. EfficientDet mimarisi - omurga ağı olarak EfficientNet[36], özellik ağı olarak BiFPN ve paylaşılan sınıf/kutu tahmin ağı kullanır. Hem BiFPN katmanları hem de sınıf/kutu ağı katmanları, Çizelge 3.7'de gösterildiği gibi farklı kaynak kısıtlamalarına göre birden çok kez tekrarlanır [3]

Sahip olduğu yeni mimari iyileştirmelerin yanında kullanılan bileşik ölçekleme yöntemi, Efficient modelinin hem doğruluk hem de verimliliği optimize etmeyi başarmasındaki önemli etmenlerdendir. Önceki çalışmalarda çoğunlukla daha büyük omurga ağları kullanılarak, daha büyük giriş görüntüleri kullanılarak veya daha fazla FPN katmanı kullanılarak dedektörler ölçeklemektedir. Ancak bu yöntemler yanlızca bir veya sınırlı ölçekleme boyutlarına odaklanmaktadır ve bu yüzden verimliliği görece azdır. Son çalışmalarda ise ağ genişliği, derinliği ve giriş çözünürlüğünün tüm boyutlarını birlikte ölçeklendirerek yapılan görüntü sınıflandırması çalışmalarında performans artışı gözlenmektedir. Bu yüzden son çalışmalardaki performans artışı, omurga ağının, BiFPN ağının, sınıf/kutu ağının ve çözünürlüğün tüm boyutlarını birlikte ölçeklendirmek için basit bir bileşik katsayı (φ) kullanan ve nesne tespiti için tüm boyutları birlikte ölçekleyebilen yeni bir sezgisel tabanlı bileşik ölçekleme yönteminin geliştirilmesini sağlamıştır [10]. Çizelge 3.8'de EfficientDet-D0 (φ = 0)'ı D7 (φ = 7)'yi ifade edecek şekilde farklı bileşik katsayı (φ) değerleri kullanılmakta ve her model için sezgisel tabanlı yapılan ölçeklemede kullanılan giriş boyutunu, BiFPN kanal ve katman değerlerini, kutu/sınıf değerlerini göstermektedir ve bu çizelgedeki değerler eşitlik 3.1, 3.2, 3.3'ün hesaplanmasında kullanılmaktadır [3].

Çizelge 3.8. EfficientDet D0-D7 için ölçekleme yapılandırmaları–φ diğer tüm ölçekleme boyutlarını kontrol eden bileşik katsayıdır; BiFPN, kutu/sınıf ağı ve giriş boyutu sırasıyla eşitlik 3.1, 3.2, 3.3 kullanılarak ölçeklenir [3]

	Giriş Boyutu	Omurga Ağı	BiFPN (Kanallar)	BiFPN (Katmanlar)	Sınıf / Kutu (Katmanlar)
	R_{input}		W_{bifpn}	D_{bifpn}	D_{class}
D0(φ=0)	512	B0	64	3	3
D1(φ=1)	640	B1	88	4	3
D2(φ=2)	768	B2	112	5	3
D3(φ=3)	896	B3	160	6	4
D4(φ=4)	1024	B4	224	7	4
D5(φ=5)	1280	B5	288	7	4
D6(φ=6)	1280	B6	384	8	5
D6(φ=7)	1536	B6	384	8	5

Bileşik ölçekleme methodunda omurga ağı olarak EfficientNet (B0'dan B6'ya) kullanılmaktadır ve ağların genişlik / derinlik ölçekleme katsayıları değiştirilmeden kullanılmıştır. BiFPN derinliği linear olarak artırılarak küçük tam sayılara yuvarlanmaktadır. BiFPN genişliği ise exponansiyel olarak artırılarak en iyi genişlik

ölçekleme faktörü hesaplanmaktadır. BiFPN derinlik ve genişliği aşağıda verilen formül ile hesaplanmaktadır [3]:

$$W_{bifpn} = 64 * (1{,}35^{\varphi}) , \qquad D_{bifpn} = 3 + \varphi \tag{3.1}$$

Kutu / sınıf tahmin ağında derinlik doğrusal olarak artırılırken genişlik BiFPN ile aynı değeri alacak şekilde sabitlenmektedir ve kutu/sınıf tahmini aşağıdaki gibi hesaplanmaktadır [3]:

$$D_{box} = D_{class} = 3 + \ +\lfloor \varphi/3 \rfloor \tag{3.2}$$

BiFPN'de P3P4P5P6P7 düzeyindeki özellikler kullanıldığı için giriş görüntüsü çözünürlüğünün 2^7 = 128'e bölünebilir olması gerekmektedir ve giriş boyutu aşağıdaki gibi ifade edilmektedir [3]:

$$R_{input} = 512 + \ + \varphi * 128 \tag{3.3}$$

Çizelge 3.9, COCO veri kümesindeki farklı boyut ve karmaşıklığa sahip EfficientDet modelleri için performans ölçümlerini sunmaktadır. EfficientDet-D0 en düşük karmaşıklık seviyesine sahip olup hızlı çıkarım süresi sağlarken, bu durum EfficientDet-D7 için tam tersi şeklinde gözlemlenmektedir ve EfficientDet-D7'de en yüksek doğruluğa ulaşılmaktadır. Ancak bu durum yüksek hesaplama maliyetinin oluşmasına sebep olmaktadır. Modeller içerisinde EfficientDet-D4 ve EfficientDet-D5, doğruluk ve hız arasındaki en iyi dengeye sahip olan iki model olarak öne çıkmaktadır.

Çizelge 3.9. COCO veri kümesinde EfficientDet D0-D7 modelleri için performans ölçümleri [3]

Model	Giriş Boyutu	COCO AP (val)	Hız CPU ONNX (ms)	Hız T4 TensorRT 10 (ms)	Parametre (M)	FLOPs (B)
EfficientDet-D0	512x512	%34,6	102 ± 2,1	3,1 ± 0,1	4,0	2,5
EfficientDet-D1	640x640	%39,4	190 ± 3,4	5,8 ± 0,2	6,6	6,1
EfficientDet-D2	768x768	%43,0	310 ± 4,7	9,3 ± 0,3	8,1	11,0
EfficientDet-D3	896x896	%47,0	580 ± 6,2	17,5 ± 0,4	12,0	24,0
EfficientDet-D4	1024x1024	%49,3	980 ± 9,1	28,9 ± 0,5	20,0	55,0
EfficientDet-D5	1280x1280	%50,5	1620 ± 12,3	41,2 ± 0,6	30,0	89,0
EfficientDet-D6	1280x1280	%51,5	2980 ± 15,7	60,8 ± 0,8	43,0	161,0
EfficientDet-D7	1536x1536	%52,2	4820 ± 20,1	94,5 ± 1,2	52,0	325,0

EfficientDet, EfficientNet tabanlı bir omurga ağı ve BiFPN yapısı sayesinde nesne algılama modelleri arasında önemli gelişmeler ve yenilikler sunarak daha düşük hesaplama maliyeti ile yüksek doğruluk sağlamaktadır. Özellikle gerçek zamanlı uygulamalar ve mobil cihazlar için optimize edilen EfficientDet, ölçeklenebilir yapısıyla farklı ihtiyaçlara uyum sağlamaktadır.

EfficientDet, nesne algılama alanında yüksek doğruluk ile verimliliği dengeleyen modern bir mimari olarak öne çıkmaktadır. EfficientNet tabanlı bir omurga ve BiFPN (Bidirectional Feature Pyramid Network) ile donatılan EfficientDet, hesaplama kaynaklarını daha etkin kullanarak daha küçük model boyutlarında dahi yüksek doğruluk oranları sunabilmektedir. YOLO mimarisi, nesne algılama problemini tek bir regresyon görevi olarak ele alarak işlem hızında önemli bir avantaj sunarken, SSD mimarisi çok ölçekli özelliği sayesinde farklı boyutlardaki nesneleri tespit etmede yüksek performans göstermektedir. Ancak, YOLO'nun bazı sürümleri küçük nesnelerde doğruluk kaybı yaşayabilirken, SSD ise daha karmaşık sahnelerde performans düşüşü gösterebilir. EfficientDet ise bileşik ölçekleme yaklaşımı sayesinde hem düşük gecikme süresi hem de yüksek doğruluk sağlamayı başararak, gerçek zamanlı uygulamalarda kaynak sınırlı cihazlar için uygun bir alternatif sunmaktadır. Bu

yönüyle, EfficientDet mimarisi, hız-doğruluk dengesi açısından YOLO ve SSD mimarilerine göre daha optimize bir yapı sunmaktadır.

3.3. Performans Metrikleri

Derin öğrenme tabanlı nesne algılama modellerinin etkinliğini değerlendirmek için çeşitli performans metrikleri geliştirilmiştir. Hız, doğruluk ve nesne lokalizasyonu gibi farklı faktörleri ölçen bu metrikler, model seçiminde önemli bir rol almaktadır. Bu bölümde, modellerin karşılaştırmalı analizini yapmak için nesne algılama modellerinin değerlendirilmesinde yaygın olarak kullanılan ve çalışma kapsamında da kullanılmış olan Saniyedeki Kare Sayısı (Frame Per Second – FPS), Çıkarım Süresi (Inference Time), Kesinlik (Precision), Duyarlılık (Recall), F1-Skoru (F1-Score), Ortalama Ortalama Hassasiyet (Mean Average Precision - mAP), ve Kesişim /Birleşim Oranı (Intersection over Union - IoU) performans metrikleri ele alınacaktır. Bu metrikler, her bir model için hem doğrulama hem de test veri setleri üzerinde ayrı ayrı hesaplanmıştır.

3.3.1. Hız metrikleri: FPS ve çıkarım süresi

Derin öğrenme tabanlı nesne algılama modellerinin performans değerlendirmesinde modellerin hızı, özellikle gerçek zamanlı uygulamalar söz konusu olduğunda kritik bir öneme sahiptir. Gerçek zamanlı uygulamalarda modelin ne kadar hızlı çalıştığı modelin kullanılabilirliğini veya tercih edilebilirliğini doğrudan etkilemektedir. Özellikle en yaygın endüstriyel uygulama alanları olan güvenlik sistemleri, otonom araçlar, robotik uygulamalar ve artırılmış gerçeklik gibi alanlarda yüksek hız, doğrudan verimliliği etkilemektedir. Kullanılan donanım, modellerin karmaşıklığı, yüksek çözünürlüklü görüntülerin kullanılması, yazılım optimizasyonlarının veya algoritmik iyileştirmelerin uygulnması gibi faktörler hız metriklerinden elde edilen sonuçları etkilemektedir. Bu bağlamda, nesne algılama modellerinin performans değerlendirmesinde en yaygın kullanılan hız metrikleri FPS (Frame Per Second – Saniyedeki Kare Sayısı) ve Çıkarım Süresi (Inference Time)'dır.

Saniyedeki kare sayısı (frame per second – FPS)

FPS (Frames Per Second), modelin saniyede işlediği veya işleyebildiği kare sayısını ifade etmektedir. Gerçek zamanlı uygulamalarda kullanılan önemli bir metriktir. Modelin işlem

hızını doğrudan göstermektedir. Yüksek FPS değeri, modelin daha hızlı çalıştığını ve daha akıcı bir deneyimsunduğunu göstermektedir. Ancak modelin doğruluğu hakkında doğrudan bir bilgi verememekle birlikte kullanıldığı sistemdeki donanım konfigürasyonuna bağlı olarak değişmektedir. Özellikle gerçek zamanlı uygulamalarda yüksek FPS değerine ulaşılması gerekmektedir. FPS hesaplaması aşağıda belirtildiği şekilde yapılmaktadır [46]:

$$FPS = \frac{\text{İşlenen Kare Sayısı } (Total\ Number\ of\ Frames)}{\text{Geçen Süre } (Total\ Time\ in\ Seconds)(s)} \tag{3.4}$$

Çıkarım süresi (inference time)

Çıkarım Süresi (Inference Time), tek bir görüntünün işlenip çıktı üretilmesi için geçen süreyi ifade etmektedir. Genellikle milisaniye (ms) cinsinden ölçülmektedir. Modelin hızının objektif olarak ölçülebilmesini sağlamakla birlikte farklı modellerin hız karşılaştırılmasının daha objektif yapılabilmesine imkân sağlamaktadır. Düşük çıkarım süreleri, modelin daha hızlı çalıştığını göstermektedir. Ancak FPS'de olduğu gibi modelin doğruluğu hakkında doğrudan bir bilgi verememekle birlikte kullanıldığı sistemdeki donanım konfigürasyonuna bağlı olarak değişim göstermektedir. Inference Time hesaplaması aşağıda belirtildiği şekilde yapılmaktadır [46]:

$$\text{Çıkarım Süresi} = \frac{\text{Çıkarım için Toplam Süre } (Total\ Time\ of\ Inference)}{\text{Toplam Çıkarım Sayısı } (Total\ Number\ of\ Inferences)} \tag{3.5}$$

3.3.2. Doğruluk metrikleri: kesinlik (precision), duyarlılık (recall) ve f1-skoru (f1-score)

Derin öğrenme tabanlı nesne algılama modellerinin performans değerlendirmesinde hız metriklerinin yanı sıra doğruluk metrikleri de kritik bir öneme sahiptir. Gerçek zamanlı uygulamalarda doğru tespit edilen nesnelerin oranı modelin kullanılabilirliğini veya tercih edilebilirliğini doğrudan etkilemektedir. Bu metrikler, modelin nesneleri ne kadar doğru şekilde algıladığını ve sınıflandırdığını ölçmek amacıyla kullanılmaktadır. Kullanılan donanım, modellerin karmaşıklığı, görüntünü boyutu, yazılım optimizasyonlarının veya algoritmik iyileştirmelerin uygulnması gibi faktörler doğruluk metriklerinden elde edilen sonuçları etkilemektedir. Bu bağlamda, nesne algılama modellerinin performans

değerlendirmesinde en yaygın kullanılan doğruluk metrikleri kesinlik (precision), duyarlılık (recall) ve f1-skoru (f1-score)'dur.

Kesinlik (precision)

Modelin yaptığı tahminlerin ne kadar doğru olduğunu göstermektedir. Modelin yanlış pozitif üretme oranını göstermektedir ve bu sayede güvenlik sistemleri gibi kritik öneme sahip uygulama alanlarında kullanılacak nesne algılama model seçiminde etkin rol almaktadır. Ancak tek başına model başarısını ölçmek için yeterli bir parametre değildir. Kesinlik hesaplaması aşağıda belirtildiği şekilde yapılmaktadır [47]:

$$Kesinlik = \frac{Doğru\ Pozitifler(TP)}{(Doğru\ Pozitifler(TP) + Yanlış\ Pozitifler\ (FP))} \tag{3.6}$$

3.6 eşitliğinde;

- TP (Doğru Pozitifler – True Positives): Modelin doğru bir şekilde pozitif olarak tahmin ettiği örnekleri göstermektedir.
- FP (Yanlış Pozitifler – False Positives): Modelin yanlış bir şekilde pozitif olarak tahmin ettiği örnekleri göstermektedir.

Duyarlılık (recall)

Modelin gerçek pozitif örneklerinin ne kadarını doğru bir şekilde tahmin ettiğini ölçmektedir. Başka bir deyişle modelin gerçek nesneleri ne kadar iyi tespit ettiğini hesaplamaktadır. Modelin nesne tespit yeteneğini göstermektedir ve bundan dolayı insan takibi gibi yüksek doğruluklu nesne tespitinin gerekli olduğu uygulamalarda önemli bir yer edinmektedir. Ancak tek başına modelin başarısını belirlemede yeterli olmamakla birlikte yüksek duyarlılık değeri düşük kesinliğe sebep olmaktadır. Duyarlılık hesaplaması aşağıda belirtildiği şekilde yapılmaktadır [47]:

$$Duyarlılık = \frac{Doğru\ Pozitifler\ (TP)}{(Doğru\ Pozitifler\ (TP) + Yanlış\ Negatifler\ (FN))} \tag{3.7}$$

3.7 eşitliğinde;

- FN (Yanlış Negatifler – False Negatives): Modelin yanlış bir şekilde negatif olarak tahmin ettiği nesnelerin oranını göstermektedir.

F1-Skoru (f1-score)

Kesinlik ve duyarlılığın dengeli bir kombinasyonudur, diğer bir deyişle harmonik ortalamasıdır. Hem kesinlik hem de duyarlılık hesaplamalarını dikkate alarak modelin performansının dengeli bir şekilde hesaplanmasını sağlamaktadır. F1-Skor hesaplaması aşağıda belirtildiği şekilde yapılmaktadır [47]:

$$F1 - Skoru = 2 * \frac{(Kesinlik * Duyarlılık)}{(Kesinlik + Duyarlılık)} \quad (3.8)$$

3.3.3. Nesne yerelleştirme metrikleri: ortalama ortalama hassasiyet (mean average precision - mAP), ve kesişim / birleşim oranı (intersection over union - IoU)

Derin öğrenme tabanlı nesne algılama modellerinin performans değerlendirmesinde hız ve doğruluğa ek olarak nesnelerin doğru bir şekilde yerelleştirilmesi de kritik bir öneme sahiptir. Bu metrikler, modellerin yerelleştirme hatalarını belirleyerek performans ölçümü yapılmasına ve bu sayede modellerin performanslarının karşılaştırılarak kullanılacağı uyguulama için en iyi modelin seçilebilmesine imkân sağlamaktadır. Bu bağlamda, mAP (Mean Average Precision - Ortalama Ortalama Hassasiyet), ve IoU (Intersection over Union - Kesişim /Birleşim Oranı) gibi metrikler, modelin yerelleştirme yeteneğini ölçmek için yaygın olarak kullanılmaktadır.

Kesişim / birleşim oranı (intersection over union - IoU)

Algılanan nesnelerin tahmini kutuları ile gerçek kutular arasındaki örtüşme oranını ifade etmektedir. Başka bir deyişle, modelin tahmin ettiği yerelleştirmenin ne kadar doğru olduğunu ölçmektedir. IoU değeri 0 ile 1 arasında değişim göstermektedir. Değer 1'e yaklaştıkça modelin yerelleştirme tahmininin çok daha doğru olduğu söylenebilmektedir. Genellikle. IoU değeri 0,5'ten büyük olarak hesaplanırsa yerelleştirme tahmininin doğru olduğu kabul edilmektedir.

IoU hesaplaması aşağıda belirtildiği şekilde yapılmaktadır [47]:

$$IoU = \frac{Kesişim\ Alanı\ (Area\ of\ Intercetion)}{Birleşim\ Alanı\ (Area\ of\ Union)} \tag{3.9}$$

Ortalama ortalama hassasiyet (mean average precision - mAP)

Ortalama Doğruluk Oranı, nesne algılama modelinin doğruluğunun bir ölçüsüdür. Başka bir deyişle farklı IoU eşikleri için ortalama hassasiyet değerinin hesaplanmasıdır. Modelin genel başarısını değerlendirmek amacıyla kullanılmaktadır. COCO ve Pascal VOC veri setlerinde performans değerlendirmesi amacıyla yaygın olarak kullanılmaktadır. Algılanan sınırlayıcı kutuların hassasiyetini değerlendirmek için farklı Birleşim Üzerindeki Kesişim (IoU) eşiklerinde (mAP@0,5 (IoU eşik değeri %50) ve mAP50:95 (IoU eşik değeri %50-%95 arası)) hesaplanmaktadır. Farklı IoU eşikleri için kesinlik ve duyarlılık değerleri hesaplanıp kesinlik-duyarlılık eğrisi çizildiğinde eğrinin altında kalan alan ortalama kesinliği (AP) ifade etmektedir. Farklı nesne sınıfları için AP değerlerinin hesaplanıp tümünün ortalamasının alınması ile AP değerlerinin ortalaması yani mAP değeri bulunmaktadır. Yüksek mAP değerleri, modelin hem doğru sınıflandırma hem de doğru yerelleştirme yaptığını göstermektedir. mAP hesaplaması aşağıda belirtildiği şekilde yapılmaktadır [47]:

$$mAP = \frac{1}{N}\sum_{i=1}^{N} \mathrm{AP}_i \tag{3.10}$$

4. DENEYSEL SONUÇLAR VE TARTIŞMA

Bu bölüm içerisinde Bölüm 3'te anlatılan YOLO, SSD ve EfficientDet modellerinin gerçek zamanlı nesne algılama performansları kapsamlı bir şekilde karşılaştırmalı olarak analiz edilmiştir. Deneyler, belirlenen deney ortamında, aynı veri seti ve koşullar altında gerçekleştirilmiş olup, modellerin doğruluk, hız ve nesne yerelleştirilme performansları çeşitli metrikler üzerinden değerlendirilmiştir. Çalışmada Saniyedeki Kare Sayısı (Frame Per Second – FPS), Çıkarım Süresi (Inference Time), Kesinlik (Precision), Duyarlılık (Recall), F1-Skoru (F1-Score), Ortalama Ortalama Hassasiyet (Mean Average Precision - mAP), ve Kesişim /Birleşim Oranı (Intersection over Union - IoU) gibi temel performans kriterleri ele alınmış ve her bir metrik için grafiksel analizler sunulmuştur. Ayrıca, zorlu koşullar altında (yoğun obje ortamı, farklı açılar, hızlı görüntü değişimi) modellerin davranışları gözlemlenmiş ve elde edilen sonuçlar detaylı bir şekilde tartışılmıştır. Son olarak, kod optimizaasyonu ile yapılan iyileştirmeler ve bu değişikliklerin model performansına etkileri incelenerek önerilerde bulunulmuştur.

4.1. Deney Ortamı ve Test Koşulları

Deneyler, aşağıda belirtilen donanım ve yazılım konfigürasyonlarına sahip bir bilgisayarda gerçekleştirilmiştir:

- İşlemci (CPU): Intel Core i7
- Grafik İşlemci (GPU): NVIDIA GEFORCE RTX 3050
- Bellek (RAM): 16 GB
- İşletim Sistemi: Windows 11 Pro
- Yazılım Çerçevesi: PyTorch 2.0.1 + CU117
- Ek Kütüphaneler: Torchvision, OpenCV, NumPy, Matplotlib, Ultralytics, Effdet, PyCocoTools, PIL

Deneylerde GPU hızlandırması aktif olarak kullanılmış olup, tüm inference işlemleri ve metrik hesaplamaları CUDA desteği ile GPU üzerinde gerçekleştirilmiştir. CPU yanlızca veri yükleme, ön işleme ve log işlemleri için kullanılmıştır.

Tüm modeller, PyTorch tabanlı kütüphaneler kullanılarak yüklenmiş ve inference işlemi gerçekleştirilerek optimize edilmiştir. Modellerin doğruluk, hız ve performans kıyaslamalarını adil bir şekilde yapabilmek için her model, COCO 2017 veri seti üzerinde test edilmiş ve her bir metrik için sonuçlar kaydedilmiştir. COCO 2017 veri setinde 80 farklı nesne sınıfı kullanılmaktadır ve kullanılan sınıflar Eklerde paylaşılmıştır.

Deneysel sonuçların güvenilirliğini artırmak adına:

- Deneyler, aynı koşullarda üç kez tekrarlanmış ve sonuçların ortalaması alınmıştır.
- Çıkarım süresi, FPS, kesinlik, doğruluk, F1-Skoru, IoU ve mAP gibi metrikler için hesaplanan sonuçlar kaydedilerek detaylı analiz yapılmıştır.
- Sistem sıcaklığı ve GPU bellek kullanımı gibi faktörler de gözlemlenmiş, aşırı yüklenme durumları not edilmiş ve aşırı yüklenme durumlarını önleyecek şekilde kod optimizasyonu yapılmıştır. Bu sayede yüksek nitelikli donanım gerekliliği olan durumlarda donanım gerekliliği yüksek nitelikli optimize edilmiş yazılım çözümü ile azaltılarak yüksek performans elde edilmiştir.

Tüm deneyler sabit koşullarda gerçekleştirilmiş olup, değişken faktörlerin etkisini minimize etmek için sistemde arka planda çalışan gereksiz süreçler kapatılmış ve her test öncesinde önbellek temizliği yapılmıştır.

4.2. YOLO Modelleri İle Nesne Algılama Performans Analizi

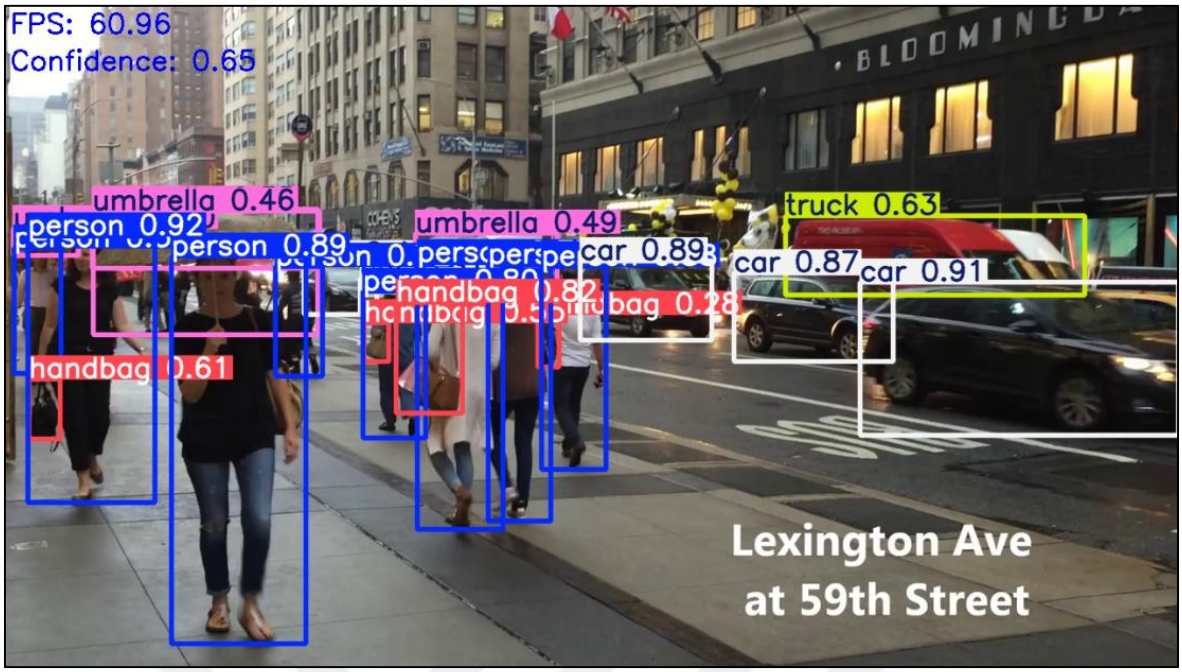
Bu bölümde, YOLO modellerinin nesne algılama performansları analiz edilmiştir. Çalışmada YOLOv8, YOLOv9, YOLOv10 ve YOLO11'in s (small), m (medium) ve l (large) olmak üzere üç farklı varyantı kullanılmıştır. Adil bir şekilde test yapılabilmesi için tüm modeller için testlerde Pytorch ortamında eğitilmiş COCO 2017 veris seti kullanılmıştır. Modeller, aynı deney ortamı ve test koşullarında ve aynı veri seti üzerinde test edilerek hız, doğruluk ve nesne yerelleştirilme performansları dataylı bir şekilde incelenmiştir. Video verileri üzerinde gerçekleştirilen nesne tespiti sonuçları ile performans analizi yapılmıştır. Bu yaklaşımla çalışmanın gerçek zamanlı uygulamalarda uygulanabilirliği, modellerin güçlü ve zayıf yönleri analiz edilerek, video tabanlı gerçek zamanlı nesne tespiti sonuçlarıyla ortaya konulmaktadır ve gerçek zamanlı uygulamalarda kullanım potansiyelleri değerlendirilmiştir. Bu çalışmada kullanılan test videoları, yalnızca akademik amaçlı olarak YouTube'dan elde edilmiş olup [48] herhangi bir ticari kullanıma konu edilmemiştir.

Kullanılan video için uygun şekilde kaynak belirtimi yapılmıştır. Videoların telif hakları ilgili sahiplerine aittir.

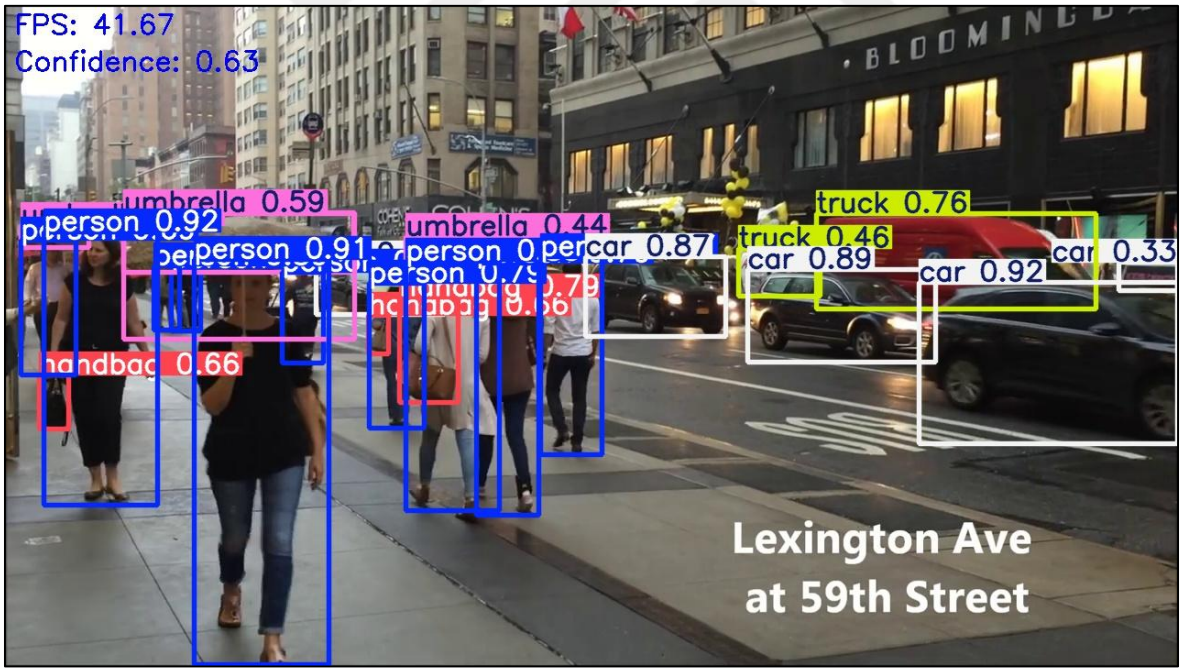
YOLO ailesi, tek aşamalı bir nesne algılama mimarisi kullanarak hızlı ve yüksek doğrlukta tahminler yapabilen bir algoritma grubudur. YOLOv8'den başlanarak YOLO modelleri, hız-optimizasyon dengesini daha iyi sağlamak için çeşitli yapısal değişikliklerden geçirilmiştir. Bu tez çalışmasında kullanılan her bir YOLO modeline ait genel özellikler (Bkz.) 3.2.1. YOLO başlığı altında detaylıca incelenmiştir. Deney ortamı ve test koşullarına dair genel özellikler ise (Bkz.) 4.1. Deney Ortamı ve Test Koşulları başlığı altında detaylı bir şekilde açıklanmıştır. Modellerin inference performanslarını adil bir şekilde karşılaştırabilmek için tüm testler aynı donanımda gerçekleştirilmiş olup, sonuçların doğruluğunu artırmak adına her test 3 kez tekrarlanmış ve ortalama değerler alınmıştır.

YOLO modellerinin değerlendirilmesi için Saniyedeki Kare Sayısı (Frame Per Second – FPS), Çıkarım Süresi (Inference Time), Kesinlik (Precision), Duyarlılık (Recall), F1-Skoru (F1-Score), Ortalama Ortalama Hassasiyet (Mean Average Precision - mAP), ve Kesişim /Birleşim Oranı (Intersection over Union - IoU) gibi temel performans metrikleri kullanılmış olup elde edilen sonuçlar hem nicel (sayısal) hem de nitel (görsel) olarak analiz edilmiştir. Model çalıştırma sırasında alınan, algılanan nesnelerin bounding box çizimlerini içeren ekran görüntüleri ile görsel değerlendirme verileri ve bu verilere dayanılarak yapılan detaylı performans yorumlamaları bu başlık altında sunulacaktır. Elde edilen performans metrikleri grafikler halinde (Bkz.) 4.5. YOLO, SSD ve EfficientDet Modellerinin Performans Metrikleri Karşılaştırması başlığı altında sunulmuştur. (Bkz.) 4.5.8. Performans Metriklerinin Grafiksel Analizi ve Genel Değerlendirme başlığı altında ise genel bir değerlendirme yapılmıştır.

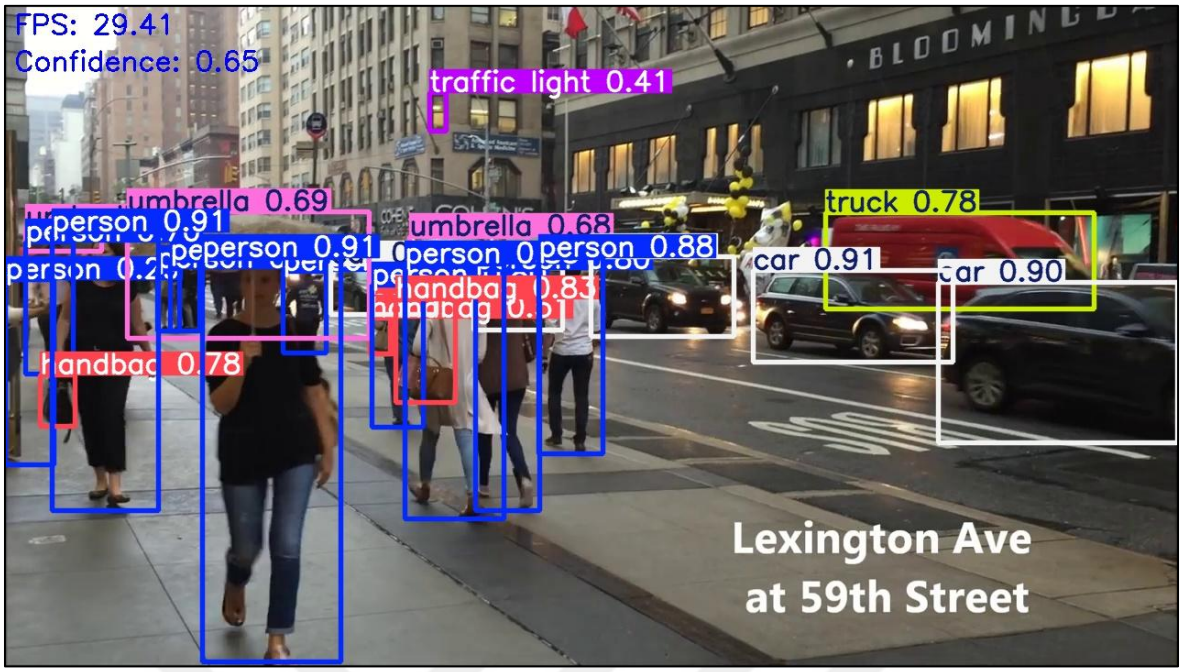
YOLO modelleri ile video görüntüleri üzerinde gerçekleştirilen gerçek zamanlı nesne algılama testlerinde genel olarak tüm YOLO modellerinin, tek aşamalı mimarileri sayesinde yüksek FPS değerlerine sahip olduğu gözlemlenmiştir. YOLOv8, YOLOv9, YOLOv10 ve YOLO11 modellerinin kullanılan tüm varyantları değerlendirildiğinde small varyanttandan medium ve large varyantlara geçildikçe FPS değerlerinin azaldığı örnek olarak seçilen Şekil 4.1, Şekil 4.2 ve Şekil 4.3'te sunulan YOLOv8 varyantları için kaydedilmiş ekran görüntülerinden görülmektedir ve buna bağlı olarak video oynatım hızlarının da azaldığı gözlemlenmiştir.



Şekil 4.1. YOLOv8 - s versiyonu ile nesne algılama ekran görüntüsü

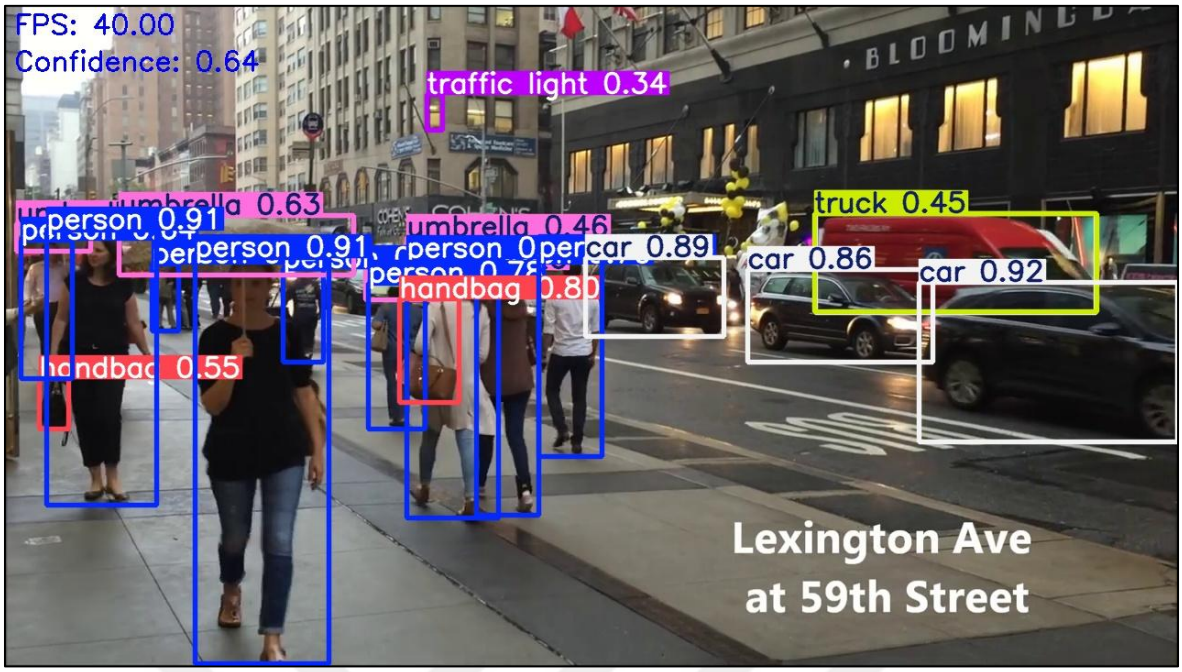


Şekil 4.2. YOLOv8 - m versiyonu ile nesne algılama ekran görüntüsü

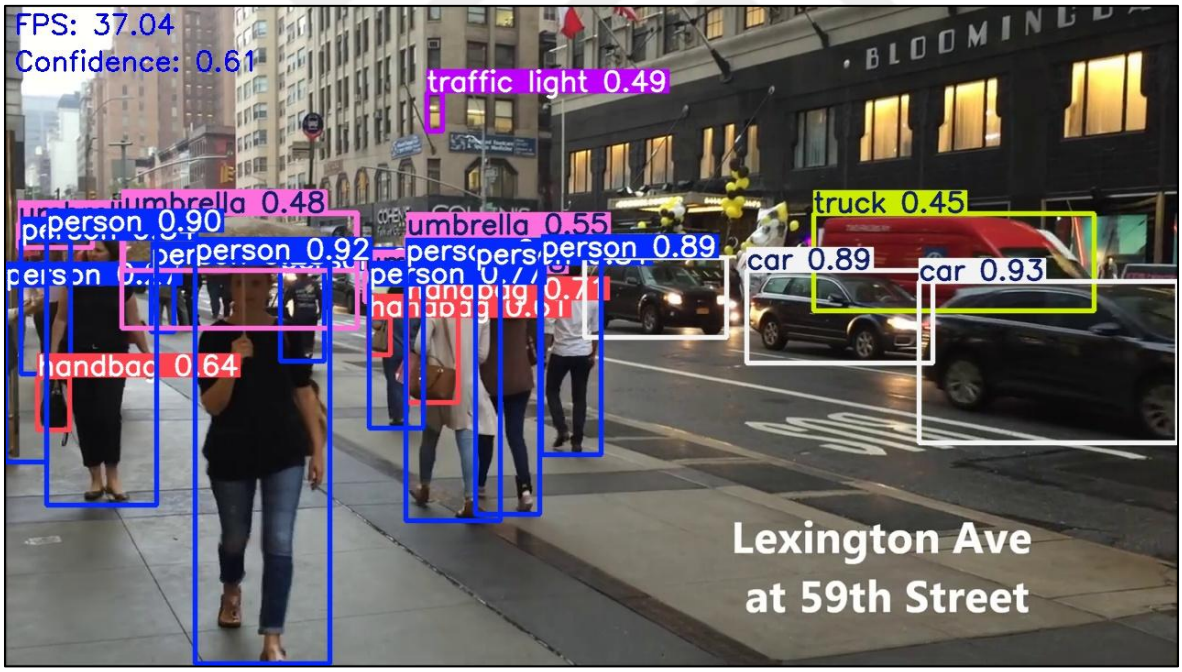


Şekil 4.3. YOLOv8 - l versiyonu ile nesne algılama ekran görüntüsü

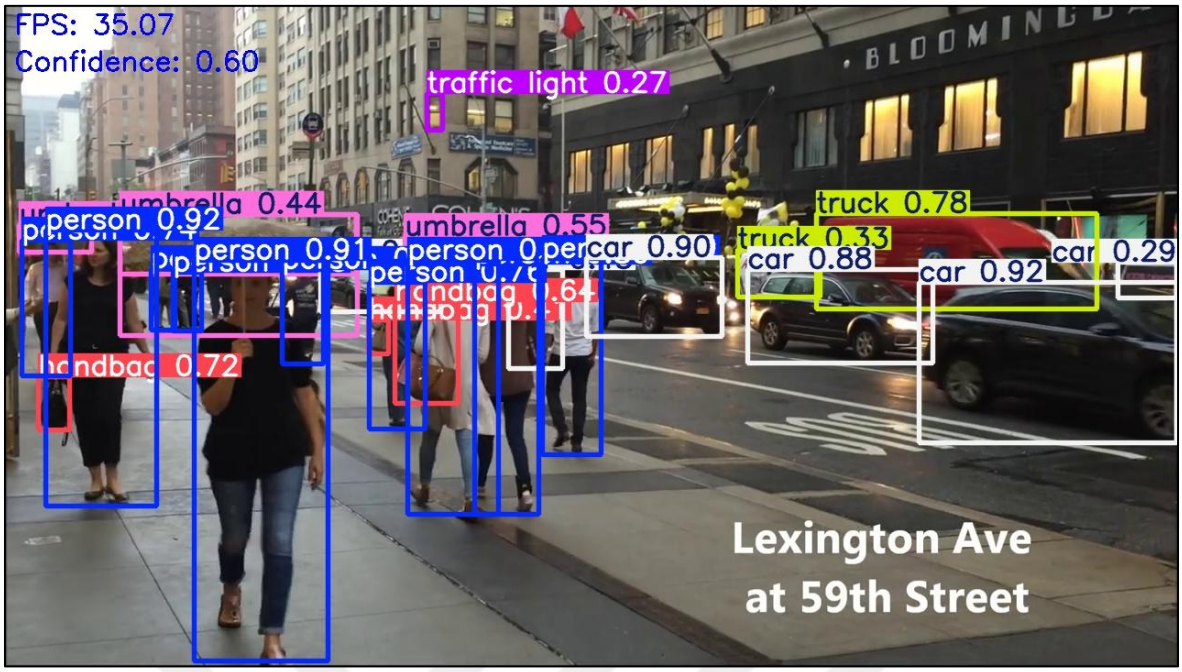
Small varyantlardan medium ve large varyantlara geçtikçe nesnelerin algılanmasındaki doğruluk oranlarının arttığı örnek olarak seçilen Şekil 4.4, Şekil 4.5 ve Şekil 4.6'te sunulan YOLOv9 varyantları için kaydedilmiş ekran görüntülerindeki çizilen sınırlayıcı kutuların hesaplanma doğrulukları ve artan confidence değerlerinden gözemlenmektedir. YOLOv8'den YOLOv9 ve YOLOv10 modellerine geçildikçe FPS değerlerinin ve çıkarım sürelerinin düştüğü gözlemlenmiştir. Burada elde edilen sayısal veriler (Bkz.) 4.6.8. Performans Metriklerinin Grafiksel Analizi ve Genel Değerlendirme başlığı altında detaylı bir şekilde verilmiştir.



Şekil 4.4. YOLOv9 - s versiyonu ile nesne algılama ekran görüntüsü

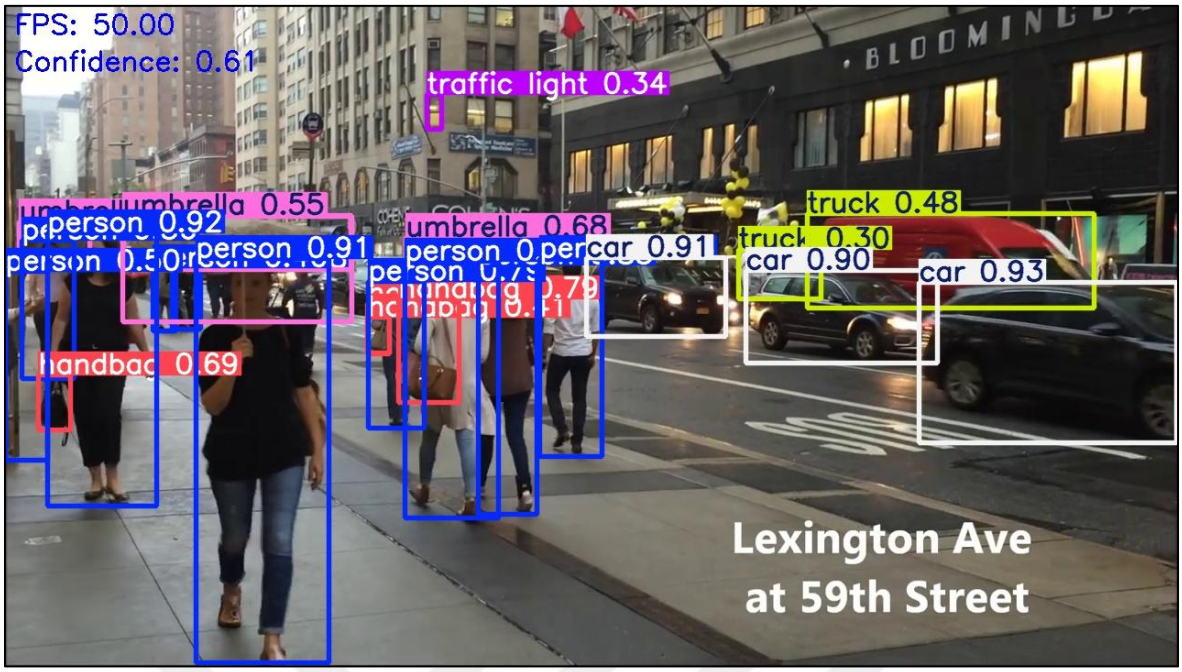


Şekil 4.5. YOLOv9 - m versiyonu ile nesne algılama ekran görüntüsü

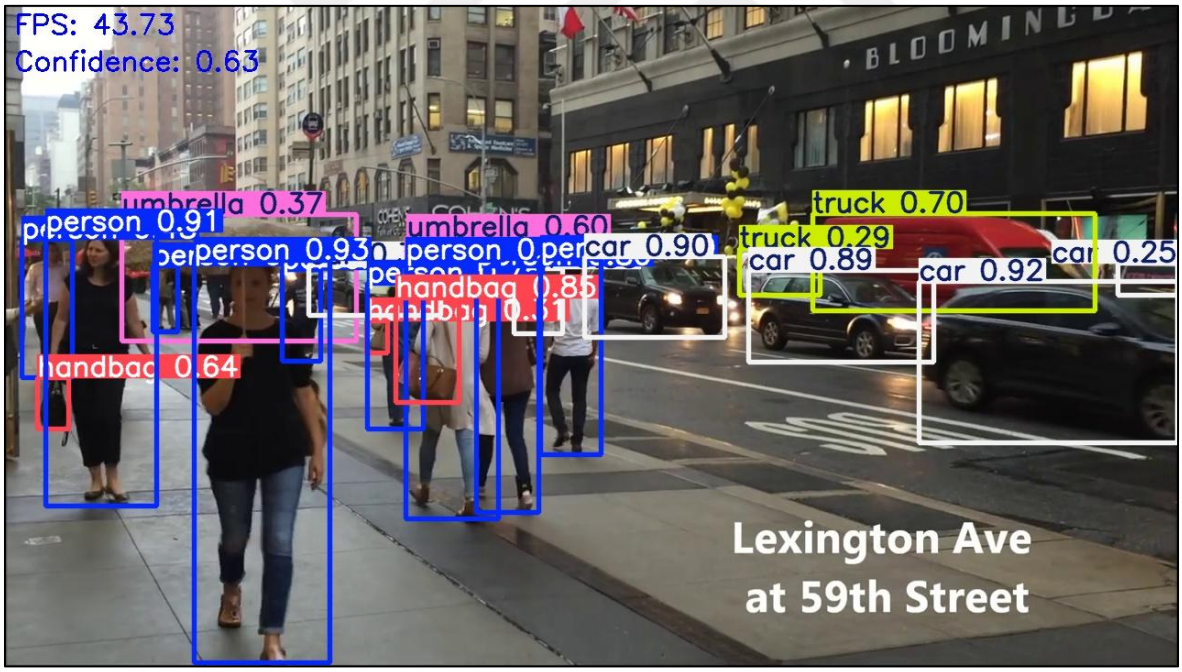


Şekil 4.6. YOLOv9 - c versiyonu ile nesne algılama ekran görüntüsü

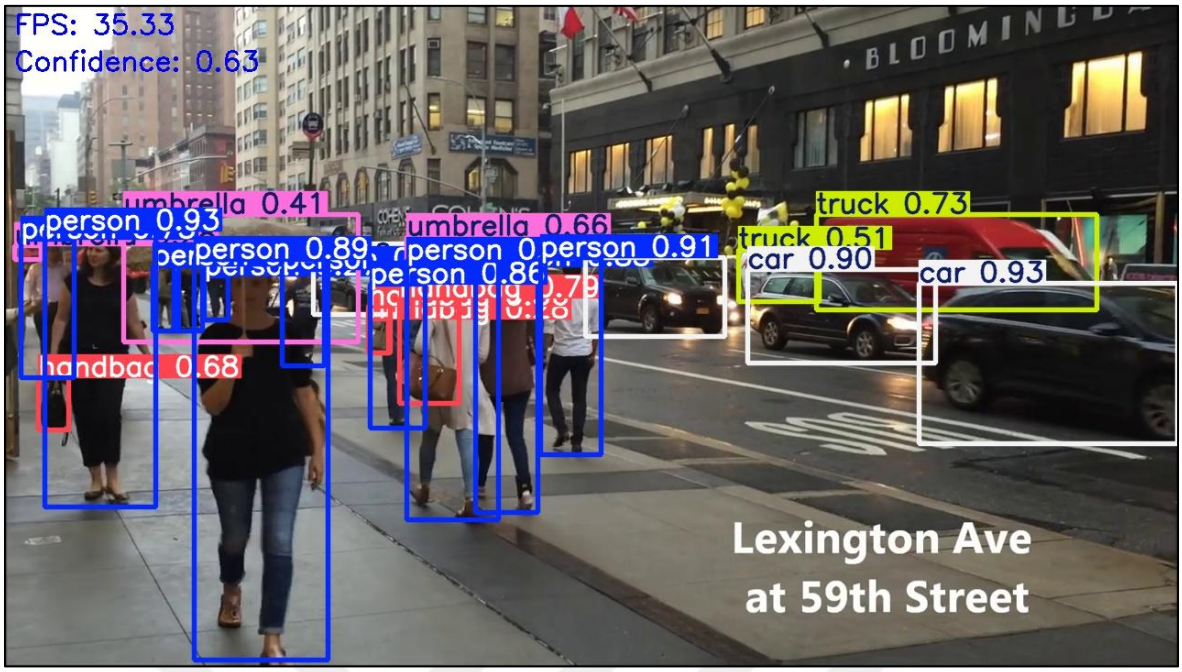
Küçük nesnelerin algılanmasında YOLO11, YOLOv10 ve YOLOv9 gibi daha yeni modellerin daha başarılı sonuçlar ürettiği örnek olarak seçilen Şekil 4.7, Şekil 4.8 ve Şekil 4.9'da sunulan YOLOv10 varyantları için kaydedilmiş ekran görüntüleri incelendiğinde görülebilmektedir, YOLOv8'in ise zorlu koşullarda küçük nesneleri kaçırma eğiliminin olduğu gözlemlenmiştir. Ayrıca YOLOv10 ve YOLO11'in özellikle medium ve large varyantlarının bellek tüketimi açısından daha yoğun bir kullanımda olduğu deneyler esnasında ncelenen bellek kullanımı ile gözlemlenmiştir.



Şekil 4.7. YOLOv10 - s versiyonu ile nesne algılama ekran görüntüsü

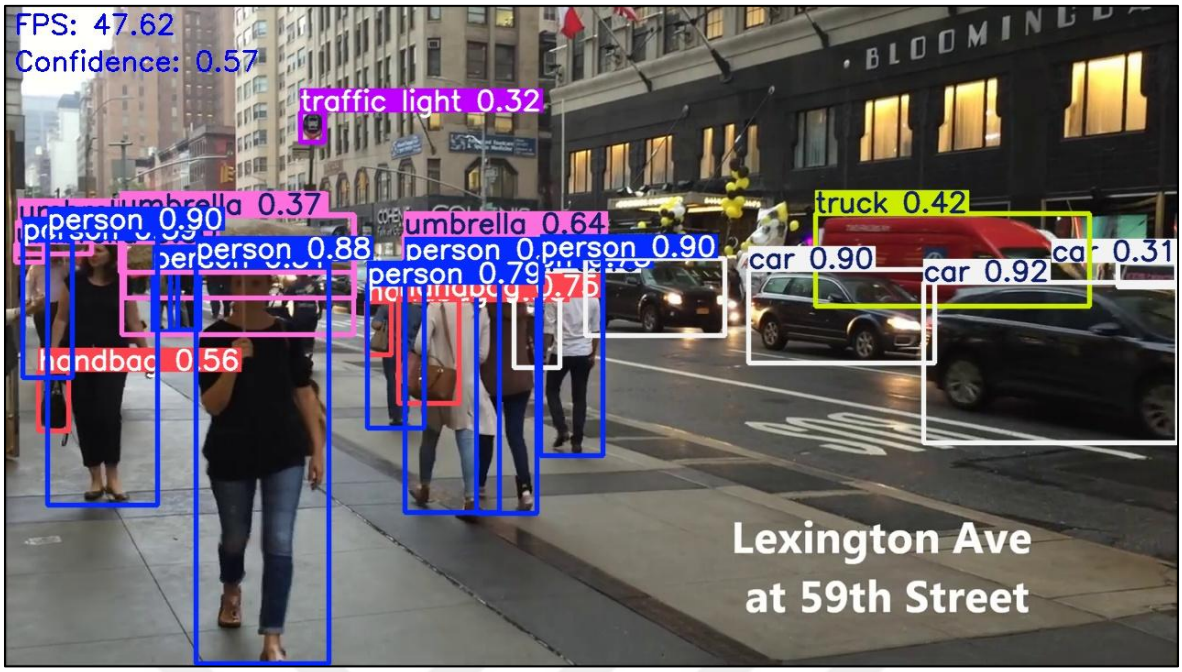


Şekil 4.8. YOLOv10 - m versiyonu ile nesne algılama ekran görüntüsü

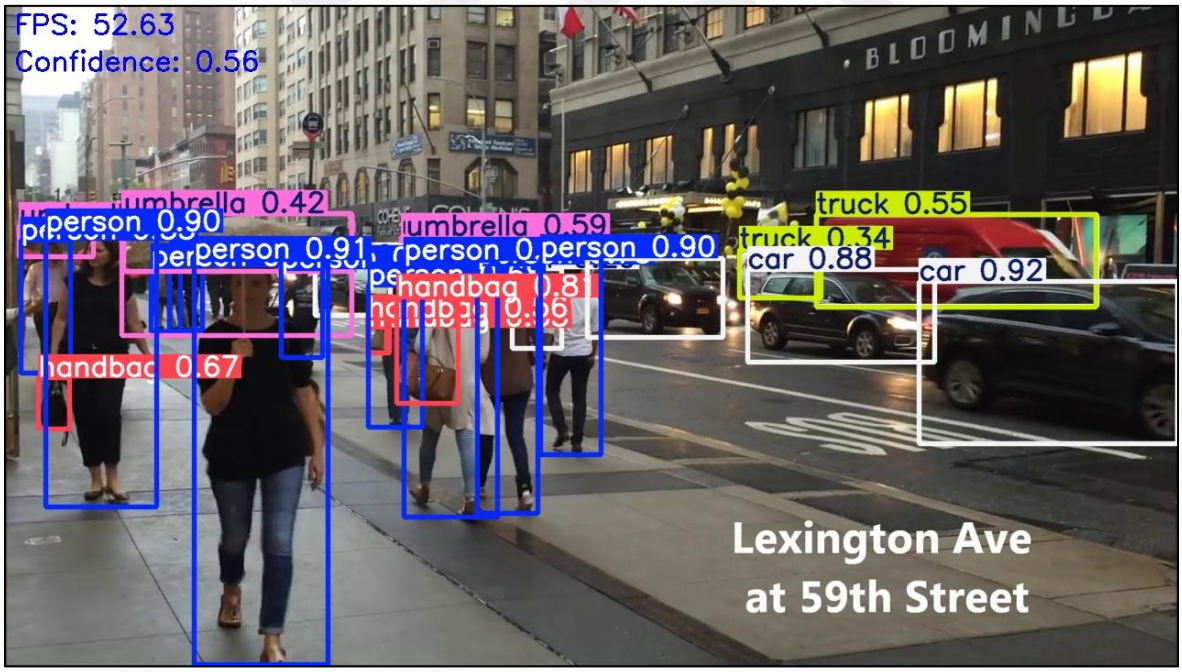


Şekil 4.9. YOLOv10 - l versiyonu ile nesne algılama ekran görüntüsü

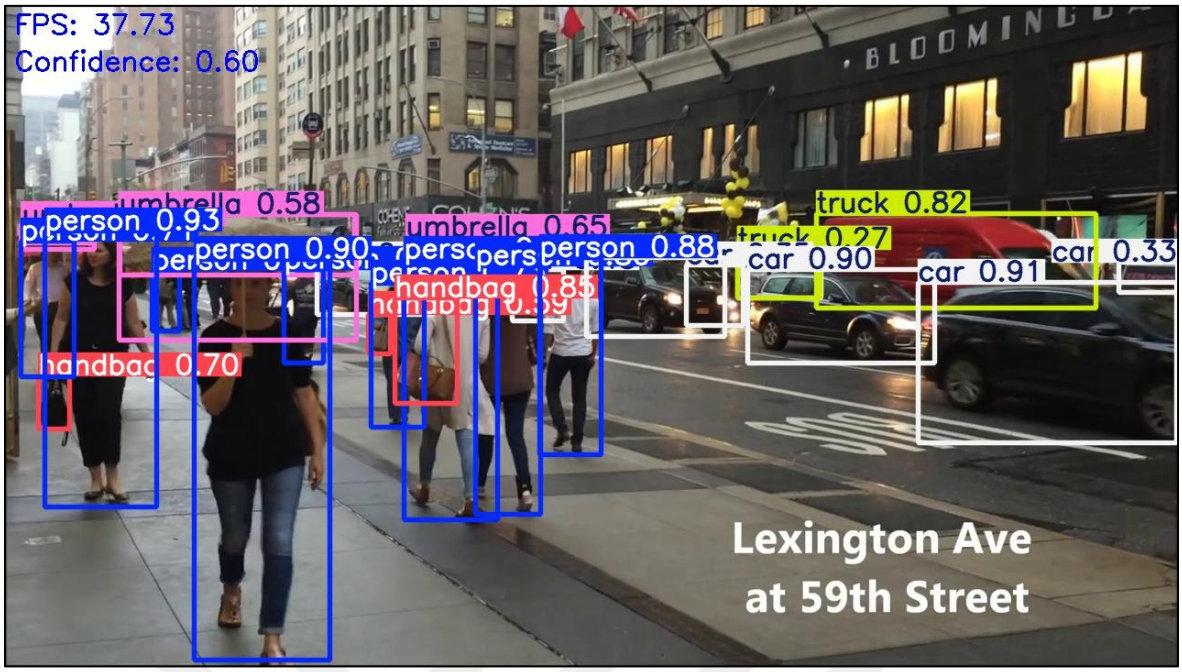
YOLO11'de ise yapılan iyileştirmelerden kaynaklı olarak daha iyi FPS değerlerinin alındığı ve daha iyi çıkarım sürelerine ulaşıldığı Şekil 4.10, Şekil 4.11 ve Şekil 4.de görülmektedir. Ayrıca, YOLO11 modelinin özellikle düşük ışıkta ve karmaşık sahnelerde daha iyi nesne algılama yapabildiği elde edilen ekran görüntülerinden çıkarılabilmektedir. Bu ekran görüntüleri ve veri analizi 4.5. Zorlu Senaryolarda YOLO, SSD ve EfficientDet Modellerinin Karşılaştırılması başlığı altında detaylı bir şekilde açıklanmaktadır.



Şekil 4.10. YOLO11 - s versiyonu ile nesne algılama ekran görüntüsü



Şekil 4.11. YOLO11 - m versiyonu ile nesne algılama ekran görüntüsü



Şekil 4.12. YOLO11 - l versiyonu ile nesne algılama ekran görüntüsü

YOLO modelleri, gerçek zamanlı nesne algılama uygulamaları için güçlü bir seçenek olarak öne çıkmaktadır. Ancak, modelin kullanım alanına ve sistem gerekliliklerine göre hız ve doğruluk arasında bir denge sağlanması gerekmektedir. Küçük ağırlıklı modeller gerçek zamanlı uygulamalarda daha yüksek FPS değeri sunarken, büyük ağırlıklı modeller daha yüksek algılama doğruluğu sağlamaktadır. Zorlu koşullarda yapılan testlerde, YOLO11 modellerinin en hızlı genel performansa sahip olduğu ve bunu YOLOv8 modellerinin izlediği, YOLOv10 modellerinin ise hız ve doğruluk arasında dengeli bir performans sergileyerek genel amaçlı uygulamalar için uygun bir seçenek olarak ön plana çıktığı görülmüştür. Ancak, sistem kaynaklarının verimli kullanılması için model seçimi, uygulamanın gereksinimlerine göre dikkatlice yapılmalıdır.

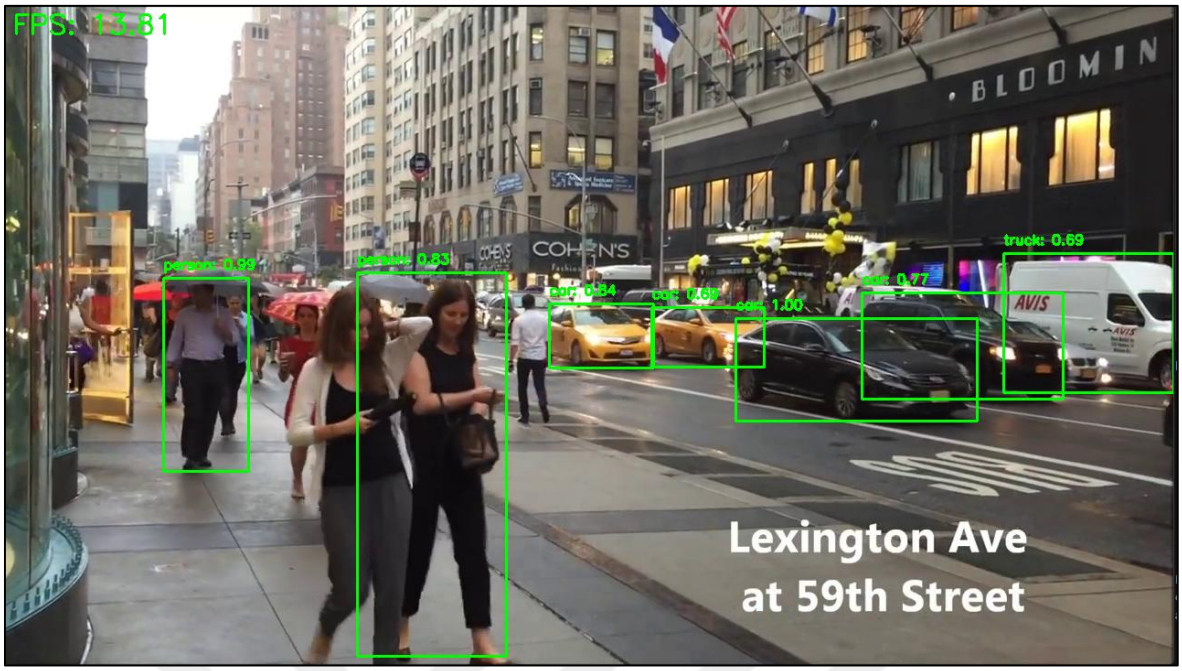
4.3. SSD Modelleri İle Nesne Algılama Performans Analizi

Bu bölümde, SSD modellerinin nesne algılama performansları analiz edilmiştir. Çalışmada SSD300-VGG16 varyantı kullanılmıştır. Adil bir şekilde test yapılabilmesi için tüm modeller için testlerde Pytorch ortamında eğitilmiş COCO 2017 veris seti kullanılmıştır. Modeller, aynı deney ortamı ve test koşullarında ve aynı veri seti üzerinde test edilerek hız, doğruluk ve nesne yerelleştirilme performansları dataylı bir şekilde incelenmiştir. Video verileri üzerinde gerçekleştirilen nesne tespiti sonuçları ile performans analizi yapılmıştır.

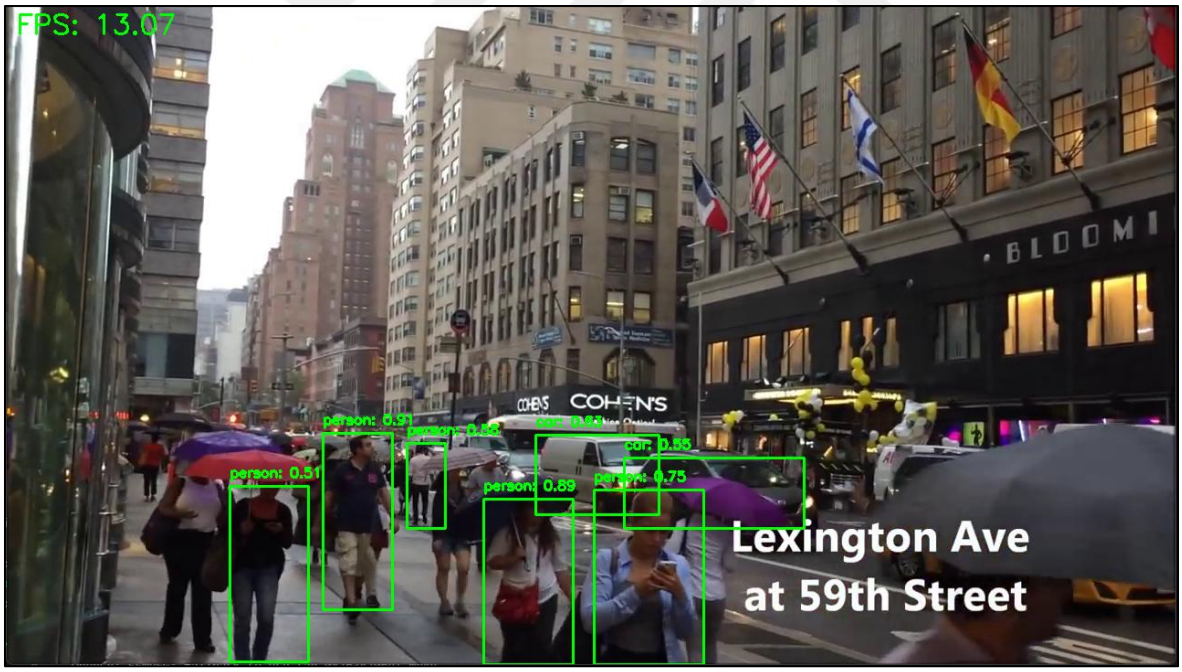
Bu yaklaşımla çalışmanın gerçek zamanlı uygulamalarda uygulanabilirliği, modellerin güçlü ve zayıf yönleri analiz edilerek, video tabanlı gerçek zamanlı nesne tespiti sonuçlarıyla ortaya konulmuştur ve gerçek zamanlı uygulamalarda kullanım potansiyelleri değerlendirilmiştir. Bu çalışmada kullanılan test videoları, yalnızca akademik amaçlı olarak YouTube'dan elde edilmiş olup herhangi bir ticari kullanıma konu edilmemiştir [48]. Kullanılan video için uygun şekilde kaynak belirtimi yapılmıştır. Videoların telif hakları ilgili sahiplerine aittir.

SSD, tıpkı YOLO gibi tek aşamalı bir nesne algılama algoritmasıdır. Ancak, temel farkı, çeşitli ölçeklerdeki özellik haritaları üzerinde nesne tahmini yapabilmesidir. Bu sayede hem küçük hem de büyük nesnelerin tespiti mümkün hale gelmektedir. Bu çalışmada kullanılan SSD300-VGG16 modeli ile giriş görüntüleri 300x300 boyutunda yeniden ölçeklendirilerek tahminlerin daha hızlı bir şekilde gerçekleştirilmesi hedeflenmiştir. Bu tez çalışmasında kullanılan SSD modeline ait genel özellikler (Bkz) 3.2.2. SSD başlığı altında detaylıca incelenmiştir. Deney ortamı ve test koşullarına dair genel özellikler ise (Bkz.) 4.1. Deney Ortamı ve Test Koşulları başlığı altında detaylı bir şekilde açıklanmıştır. Modellerin inference performanslarını adil bir şekilde karşılaştırabilmek için tüm testler aynı donanımda gerçekleştirilmiş olup, sonuçların doğruluğunu artırmak adına her test 3 kez tekrarlanmış ve ortalama değerler alınmıştır.

SSD modelinin diğer modellerle birlikte karşılaştırmalı olarak değerlendirilmesi için Saniyedeki Kare Sayısı (Frame Per Second – FPS), Çıkarım Süresi (Inference Time), Kesinlik (Precision), Duyarlılık (Recall), F1-Skoru (F1-Score), Ortalama Ortalama Hassasiyet (Mean Average Precision - mAP), ve Kesişim /Birleşim Oranı (Intersection over Union - IoU) gibi temel performans metrikleri kullanılmış olup elde edilen sonuçlar hem nicel (sayısal) hem de nitel (görsel) olarak analiz edilmiştir. Model çalıştırma sırasında alınan, algılanan nesnelerin bounding box çizimlerini içeren ekran görüntüleri ile görsel değerlendirme verileri ve bu verilere dayanılarak yapılan detaylı performans yorumlamaları bu başlık altında sunulacaktır. Elde edilen performans metrikleri grafikler halinde (Bkz.) 4.5. YOLO, SSD ve EfficientDet Modellerinin Performans Metrikleri Karşılaştırması başlığı altında sunulmuştur. (Bkz.) 4.5.8. Performans Metriklerinin Grafiksel Analizi ve Genel Değerlendirme başlığı altında ise genel bir değerlendirme yapılmıştır.



Şekil 4.13. SSD300-VGG16 modeli ile nesne algılama ekran görüntüsü 1



Şekil 4.14. SSD300-VGG16 modeli ile nesne algılama ekran görüntüsü 2

SSD300-VGG16 modeli çalıştırıldığında elde edilen tespit sonuçları, ekran görüntüleri ile analiz edilmiştir. Görsel çıktılarda, karmaşık olmaya sahnelerde SSD'nin büyük nesneleri küçük nesnelere göre daha isabetli şekilde algıladığı, ancak karmaşık sahnelerde küçük nesneleri kaçırma eğiliminde olduğu gözlemlenmiştir. Ekran görüntüleri yukarıda Şekil 4.13 ve Şekil 4.14'te numaralandırılarak sunulmuş olup, algılanan nesnelerin kutuları ve sınıf

isimleri net ve doğru bir şekilde çizilmiştir. SSD300 modeli oldukça hızlı çalışmakta ve gerçek zamanlı uygulamalara uygunluk göstermekte olup hız-performans kapsamında YOLO modellerinin gerisinde kalmaktadır. Küçük nesneleri algılama başarısının da hız performansı gibi YOLO modellerinin gerisinde kaldığı görülmektedir. Aynı zamanda genel doğruluk oranının da gelişmiş YOLO modellerine kıyasla geri planda kaldığı görülmüştür. Modelin yapısının donanım açısından daha az kaynak tüketimine olanak sağladığı gözlemlenmiştir. Özellikle VGG16 gibi klasik omurga ile uyumlu çalıştığı için birçok sistemde kolayca entegre edilebileceği görülmesine rağmen eski bir mimari olan VGG16'ya dayalı bir mimari kullanılıyor olmasından dolayı daha modern mimarilerin sunduğu iyi hız-performans ve doğruluk oranlarının gerisinde kaldığı gözlenlenmiştir. Bu yüzden yüksek hız ve doğruluk odaklı uygulamalarda güncel nesne algılama modellerinin daha çok tercih edilebilir olmasına rağmen özellikle sınırlı kaynaklara sahip sistemlerdeki düşük bellek ve işlem gücü ile tatmin edici sonuçlar sunabilmesiyle SSD'nin tercih edilebilir olduğu görülmektedir.

4.4. EfficientDet Modelleri İle Nesne Algılama Performans Analizi

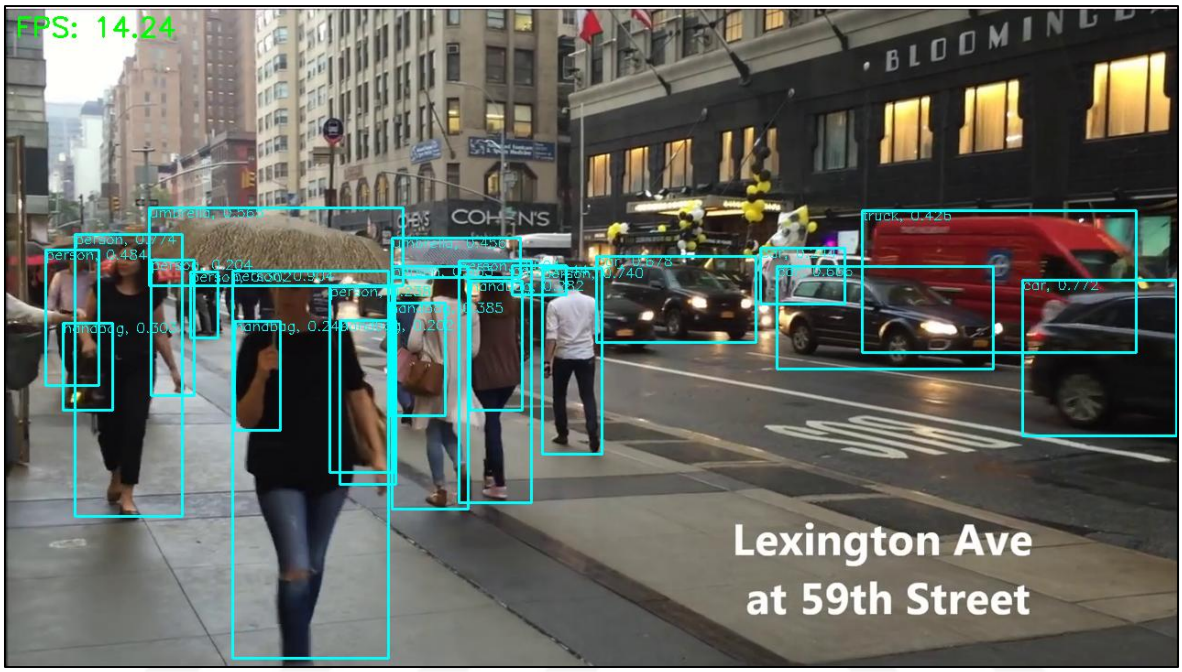
Bu bölümde, Google Brain tarafından geliştirilen EfficientDet modellerinin nesne algılama üzerindeki başarımları analiz edilmiştir. Çalışmada EfficientDet'in D0'dan D7'ye (D7 dâhil olmak üzere) kadar sekiz farklı varyantı kullanılmıştır. Adil bir şekilde test yapılabilmesi için tüm modeller için testlerde Pytorch ortamında eğitilmiş COCO 2017 veris seti kullanılmıştır. Model çalıştırmaları, PyTorch ortamında EfficientDetLite kütüphanesi aracılığıyla gerçekleştirilmiştir ve her model ayrı ayrı değerlendirilmiştir. Modeller, aynı deney ortamı ve test koşullarında ve aynı veri seti üzerinde test edilerek hız, doğruluk ve nesne yerelleştirilme performansları dataylı bir şekilde incelenmiştir. Video verileri üzerinde gerçekleştirilen nesne tespiti sonuçları ile performans analizi yapılmıştır. Bu yaklaşımla çalışmanın gerçek zamanlı uygulamalarda uygulanabilirliği, modellerin güçlü ve zayıf yönleri analiz edilerek, video tabanlı gerçek zamanlı nesne tespiti sonuçlarıyla ortaya konulmaktadır ve gerçek zamanlı uygulamalarda kullanım potansiyelleri değerlendirilmiştir. Bu çalışmada kullanılan test videoları, yalnızca akademik amaçlı olarak YouTube'dan elde edilmiş olup herhangi bir ticari kullanıma konu edilmemiştir [48]. Kullanılan video için uygun şekilde kaynak belirtimi yapılmıştır. Videoların telif hakları ilgili sahiplerine aittir.

EfficientDet, hızlı ve doğru nesne algılama amacıyla EfficientNet tabanlı omurga mimarisine sahip, çok ölçekli özellik haritalarını verimli şekilde birleştiren BiFPN yapısı ile dikkat çekmektedir. EfficientDet modelleri, D0'dan D7'ye doğru büyüyen bir yapıdadır. Model numarası arttıkça, katman sayısı, çözünürlük, parametre miktarı ve doğruluk da artar, ancak hesaplama maliyeti ve işlem süresi de yükselmektedir. Bu tez çalışmasında kullanılan her bir EfficientDet modeline ait genel özellikler (Bkz.) 3.2.3. EfficientDet başlığı altında detaylıca incelenmiştir. Deney ortamı ve test koşullarına dair genel özellikler ise (Bkz.) 4.1. Deney Ortamı ve Test Koşulları başlığı altında detaylı bir şekilde açıklanmıştır. Modellerin inference performanslarını adil bir şekilde karşılaştırabilmek için tüm testler aynı donanımda gerçekleştirilmiş olup, sonuçların doğruluğunu artırmak adına her test 3 kez tekrarlanmış ve ortalama değerler alınmıştır. Bu çalışmada, EfficientDet-D0'dan EfficientDet-D7'ye kadar olan modeller kullanılarak, hem hız hem de doğruluk açısından kapsamlı karşılaştırmalar yapılmaktadır.

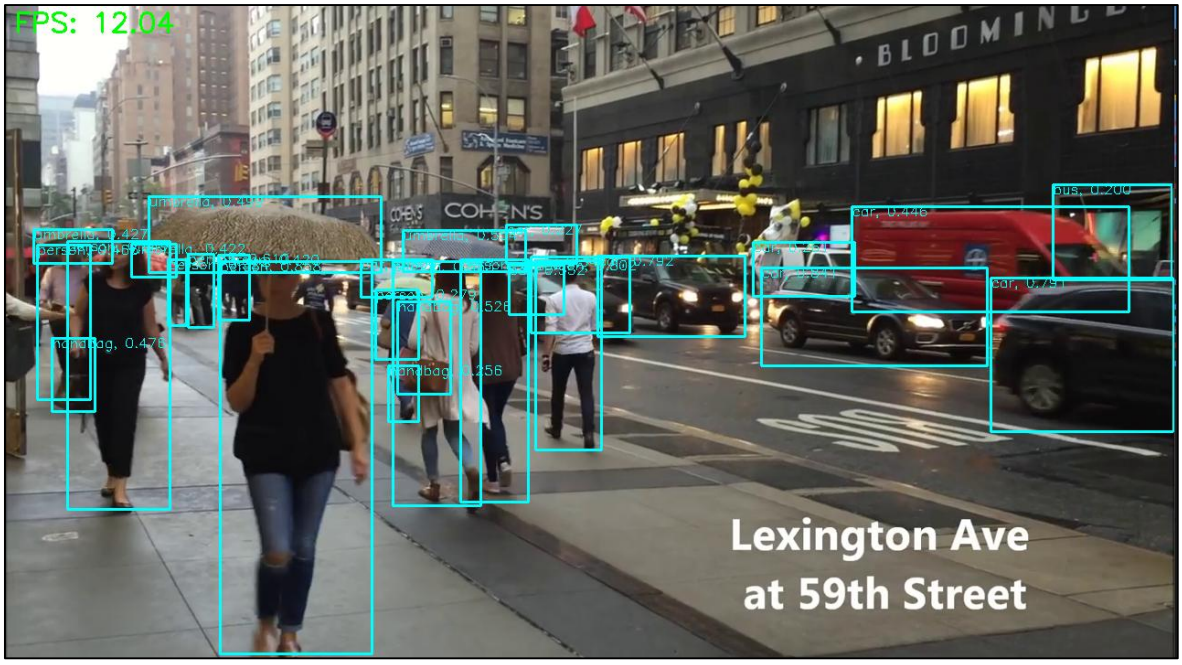
EfficientDet modellerinin diğer modellerle karşılaştırmalı olarak değerlendirilmesi için Saniyedeki Kare Sayısı (Frame Per Second – FPS), Çıkarım Süresi (Inference Time), Kesinlik (Precision), Duyarlılık (Recall), F1-Skoru (F1-Score), Ortalama Ortalama Hassasiyet (Mean Average Precision - mAP), ve Kesişim /Birleşim Oranı (Intersection over Union - IoU) gibi temel performans metrikleri kullanılmış olup elde edilen sonuçlar hem nicel (sayısal) hem de nitel (görsel) olarak analiz edilmiştir. Model çalıştırma sırasında alınan, algılanan nesnelerin bounding box çizimlerini içeren ekran görüntüleri ile görsel değerlendirme verileri ve bu verilere dayanılarak yapılan detaylı performans yorumlamaları bu başlık altında sunulacaktır. Elde edilen performans metrikleri grafikler halinde (Bkz.) 4.5. YOLO, SSD ve EfficientDet Modellerinin Performans Metrikleri Karşılaştırması başlığı altında sunulmuştur. (Bkz.) 4.5.8. Performans Metriklerinin Grafiksel Analizi ve Genel Değerlendirme başlığı altında ise genel bir değerlendirme yapılmıştır.

EfficientDet modelleri çalıştırıldığında, model numarası büyüdükçe FPS değerinde düşüş gözlenlenmiştir. Özellikle D0, D1 ve D2 modelleri, hafif ve hızlı modeller olarak düşük donanım gücüne sahip kaynaklarda dahi çalışabilirken; D6 ve D7 modelleri yüksek doğruluk oranları sunmakla birlikte yüksek işlem gücü gerektirmiştir. YOLOv10 ve YOLO11 gibi güncel modellerle akrşılaştırıldığında EfficientDet modellerinin FPS açısından geride kaldığı görülmektedir. Şekil 4.15 ve Şekil 4.16'da nesne algılama ekran görüntüleri sunulmuş olan EfficientDet-D0 ve EfficientDet-D1 modellerinin küçük boyutlu mimarileri

sayesinde düşük gecikme ve hızlı inference süreleri sunarak yüksek FPS değerlerinin elde edilmesine katkıda bulunduğu gözlemlenmiştir. Ancak küçük nesne algılamada yetersiz kalabildikleri ve özellikle kalabalık sahnelerde nesne sınıflandırmasını doğru yapabilmesine rağmen sınırlayıcı kutuların çiziminde hatalı çizimler yapabildiği görülmüştür.

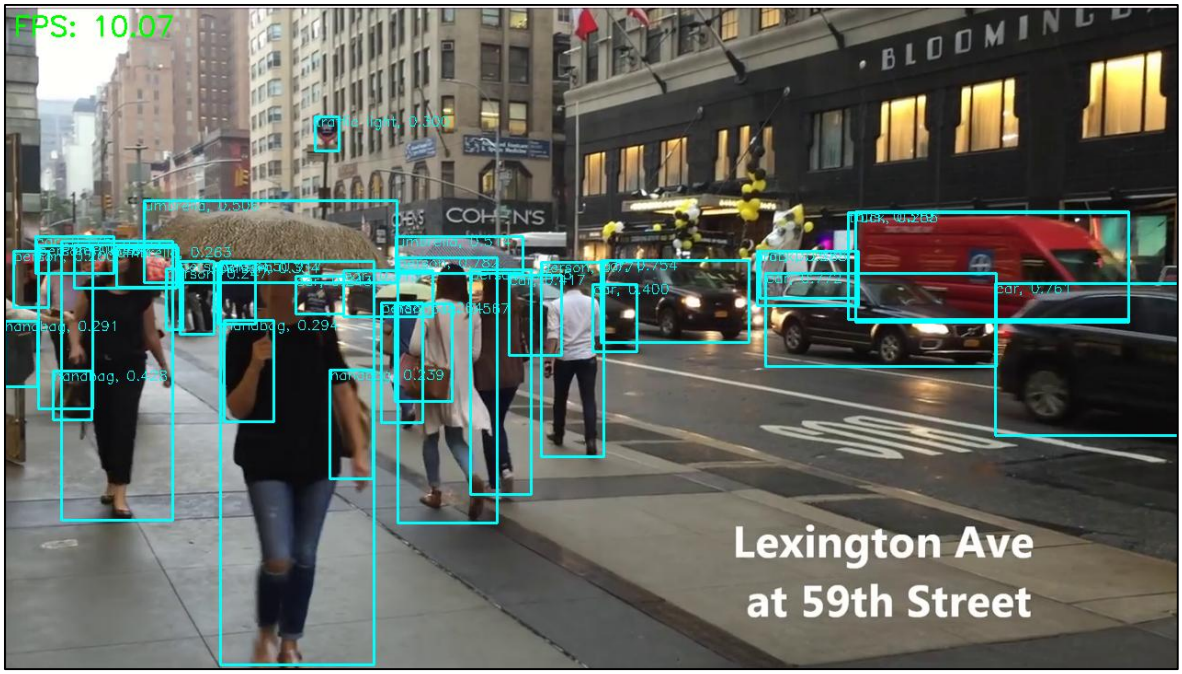


Şekil 4.15. EfficientDet-D0 versiyonu ile nesne algılama ekran görüntüsü

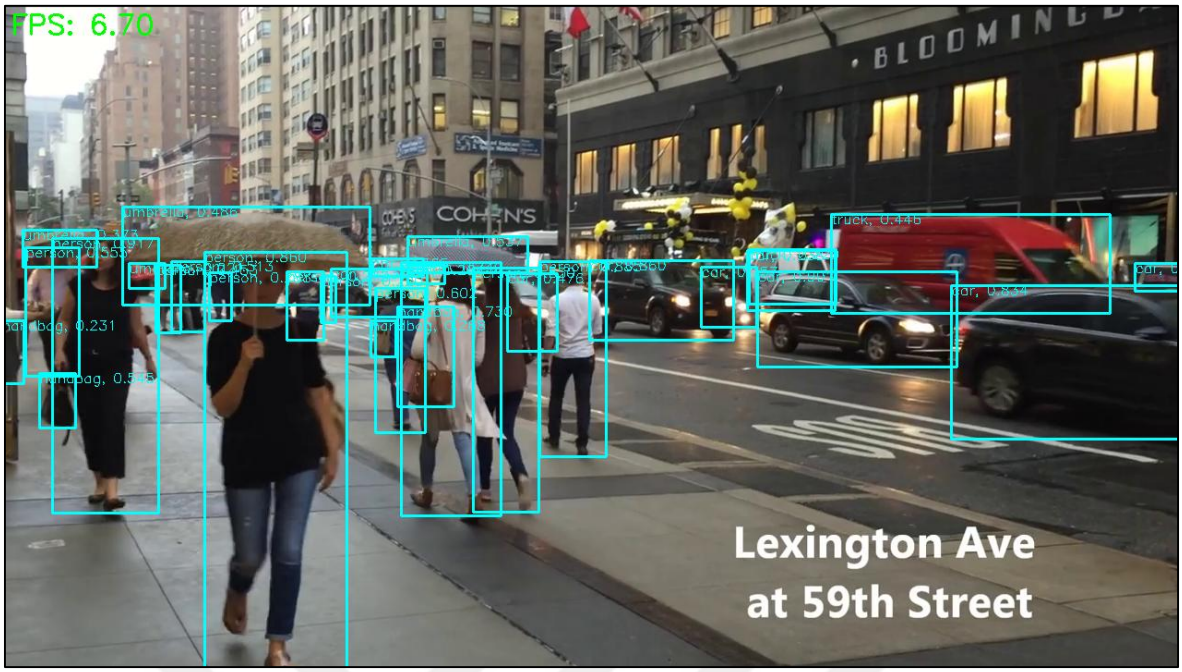


Şekil 4.16. EfficientDet-D1 versiyonu ile nesne algılama ekran görüntüsü

Şekil 4.17 ve Şekil 4.18'da nesne algılama ekran görüntüleri sunulmuş olan EfficientDet-D2 ve EfficientDet-D3 modellerinin, EfficientDet-D0 ve EfficientDet-D1'e kıyasla daha fazla sayıda nesneyi doğru şekilde algılamasının yanında küçük nesneleri daha yüksek doğrulukla sınıflandırdığı ve büyük nesne algılamalarında daha iyi sonuçlar verdiği gözlemlenmiştir. Bunun dışında EfficientDet-D0 ve EfficientDet-D1 modellerinde olduğu gibi EfficientDet-D2 ve EfficientDet-D3 modellerinin de yoğun kalabalık veya düşük ışık olan zorlu senaryolarda modellerin tespit başarımının düştüğü gözlemlenmiş ve alınan ekran görüntüleri ile 4.5. Zorlu Senaryolarda YOLO, SSD ve EfficientDet Modellerinin Karşılaştırılması başlığı altında daha detaylı bir şekilde açıklanmaktadır.

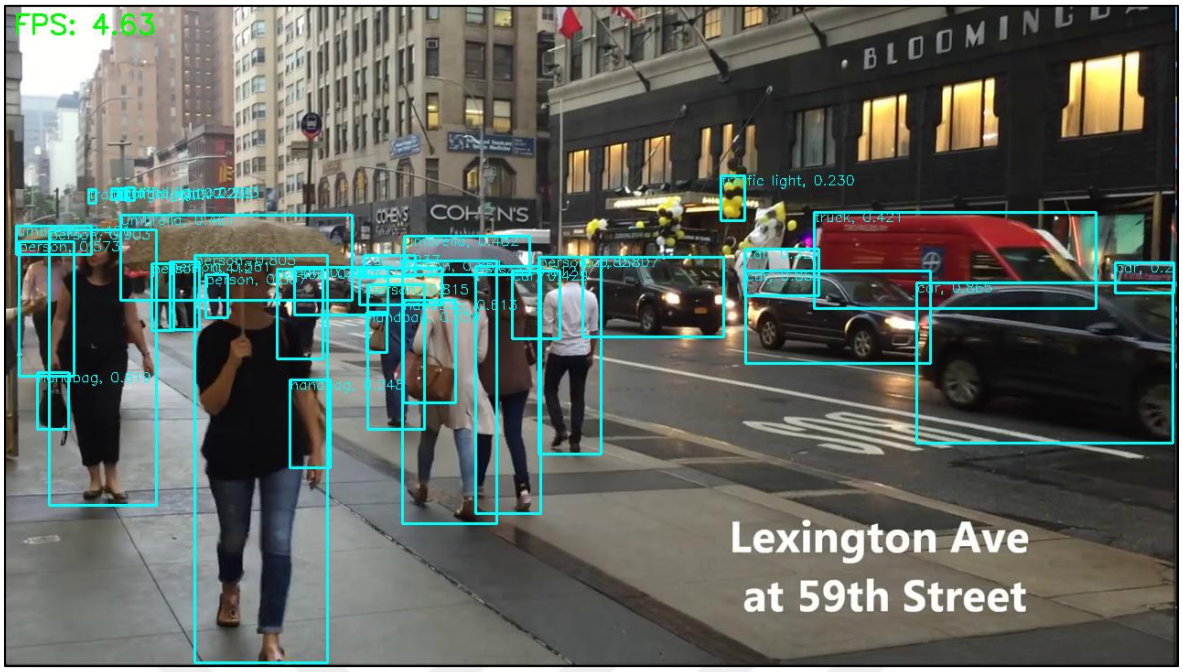


Şekil 4.17. EfficientDet-D2 versiyonu ile nesne algılama ekran görüntüsü

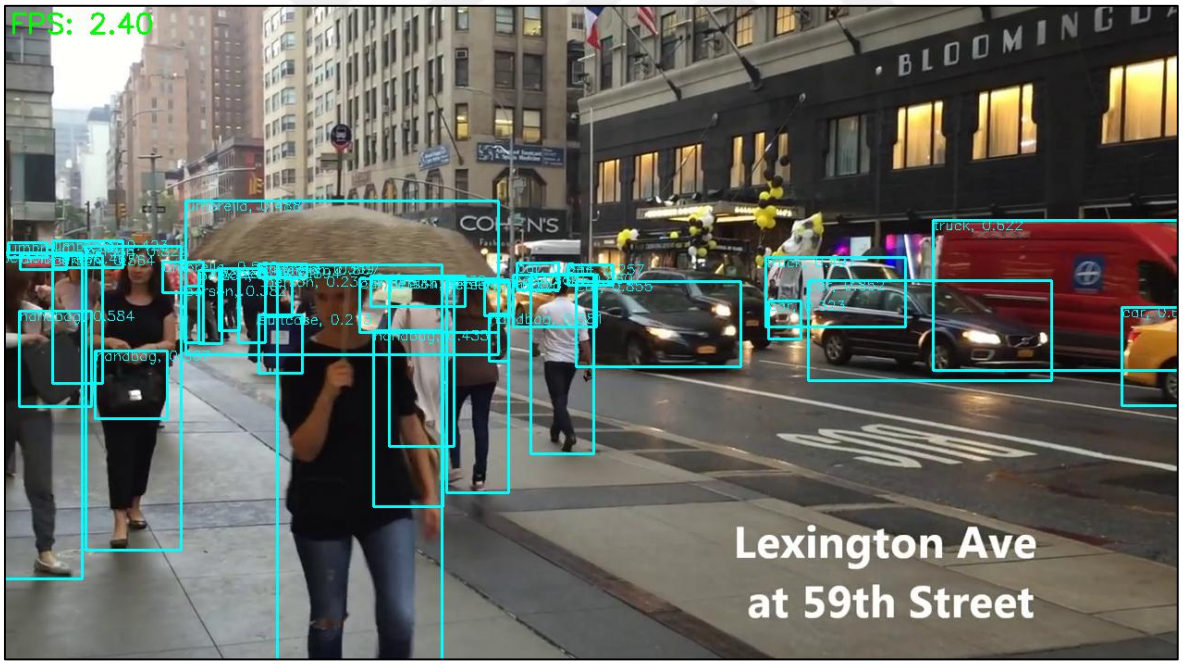


Şekil 4.18. EfficientDet-D3 versiyonu ile nesne algılama ekran görüntüsü

Şekil 4.19 ve Şekil 4.20'da nesne algılama ekran görüntüleri sunulmuş olan EfficientDet-D4 ve EfficientDet-D5 modellerinin, daha derin özellik haritaları sayesinde özellikle yoğun sahnelerde nesne algılama başarımının artmış olduğu, hem konumlama hem de sınıflandırma açısından oldukça tutarlı ve dengeli sonuçların elde edildiği ve küçük nesnelerin tespitinin daha başarılı bir şekilde yapılabildiği gözlemlenmiştir. Özellikle EfficientDet D3-D5 aralığındaki modellerin doğruluk ve hız arasında optimal denge sağladığı görülmültür. Ancak düşük donanım gücüne sahip kaynaklarda kullanımının aynı kendisinde daha büyük modeller olan EfficientDet-D6 ve EfficientDet-D7 modelleri gibi zorlayıcı olduğu gözlemlenmiş, bu yüzden yüksek donanım gücüne sahip sitemlerde bu modellerin kullanımının daha kullanışlı olacağı çıkarımlanmıştır.



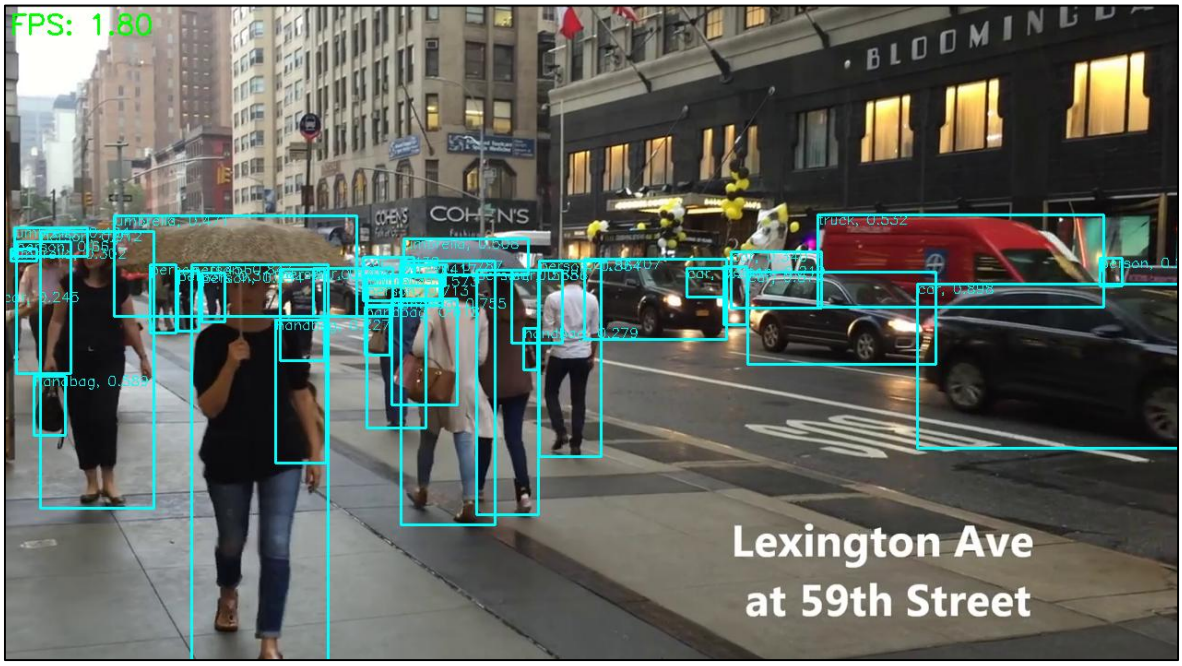
Şekil 4.19. EfficientDet-D4 versiyonu ile nesne algılama ekran görüntüsü



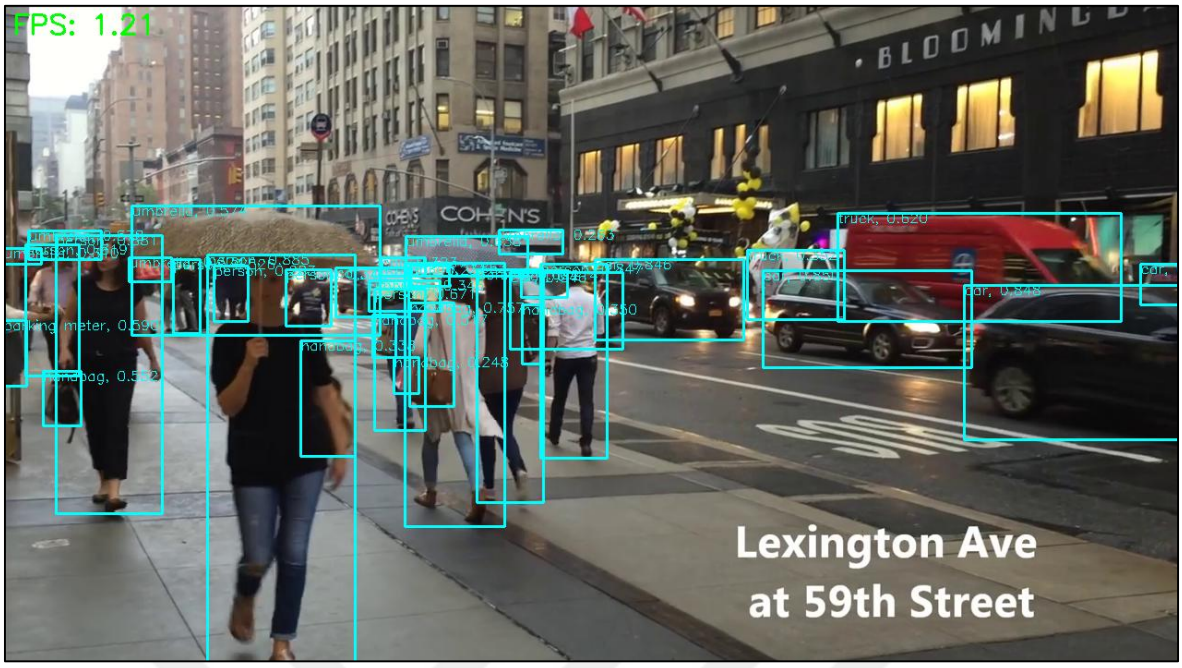
Şekil 4.20. EfficientDet-D5 versiyonu ile nesne algılama ekran görüntüsü

Şekil 4.21 ve Şekil 4.22'da nesne algılama ekran görüntüleri sunulmuş olan EfficientDet-D6 ve EfficientDet-D7 modellerinin, yüksek çözünürlüklü görüntü girişleriyle çalışabilme yetenekleri sayesinde en yüksek doğruluk değerlerini sağladığı, özellikle karışık arka plana sahip sahnelerde, küçük ve üst üste binen nesnelerin bile daha doğru şekilde ayrıştırılabildiği görülmüştür. Ancak bu doğruluk artışının işlem süresi ve donanım gereksinimleri açısından

bir artışa sebep olduğu bu yüzden yüksek çıkarım süreleri ve düşük FPS değerlerinin elde edildiği gözlemlenmiştir. Bu yüzden bu modellerin gerçek zamanlı uygulamalarda kullanımının, yalnızca güçlü donanım gereksiniminin olduğu sistemlerde mümkün olabileceği çıkarımlanmıştır. Ayrıca, EfficientDet-D0 ve EfficientDet-D1 gibi modellerde, sahnede olmayan ya da alakasız bölgelerde sınırlayıcı kutuların çizildiği, EfficientDet-D6 ve EfficientDet-D7 gibi modellerde ise bu hataların büyük ölçüde azaltıldığı gözlemlenmiştir.



Şekil 4.21. EfficientDet-D6 versiyonu ile nesne algılama ekran görüntüsü



Şekil 4.22. EfficientDet-D7 versiyonu ile nesne algılama ekran görüntüsü

EfficientDet modelleri, sunduğu ölçeklenebilir yapı sayesinde birçok uygulama senaryosuna uyum sağlayabilen esnek bir çözüm sunmaktadır. Modellerin, genel olarak değerlendirildiğinde, YOLO modelleri kadar hızlı olmadığı ancak küçük nesnelerin algılanmasında gösterdiği başarı ile öne çıktığı görülmektedir. Özellikle EfficientDet D3-D5 aralığındaki modellerin doğruluk ve hız arasında bir denge arandığı uygulamalarda kullanımının daha uygun olduğu, EfficientDet-D6 ve EfficientDet-D7 modellerinin yüksek doğruluk gerektiren sunucu tabanlı yüksek donanım gücüne sahip sistemlerde tercih edilebileceği ve EfficientDet-D0 ve EfficientDet-D2 gibi hafif modellerin gömülü sistemler ve mobil cihazlarda kullanımının oldukça uygun olduğu söylenebilmektedir.

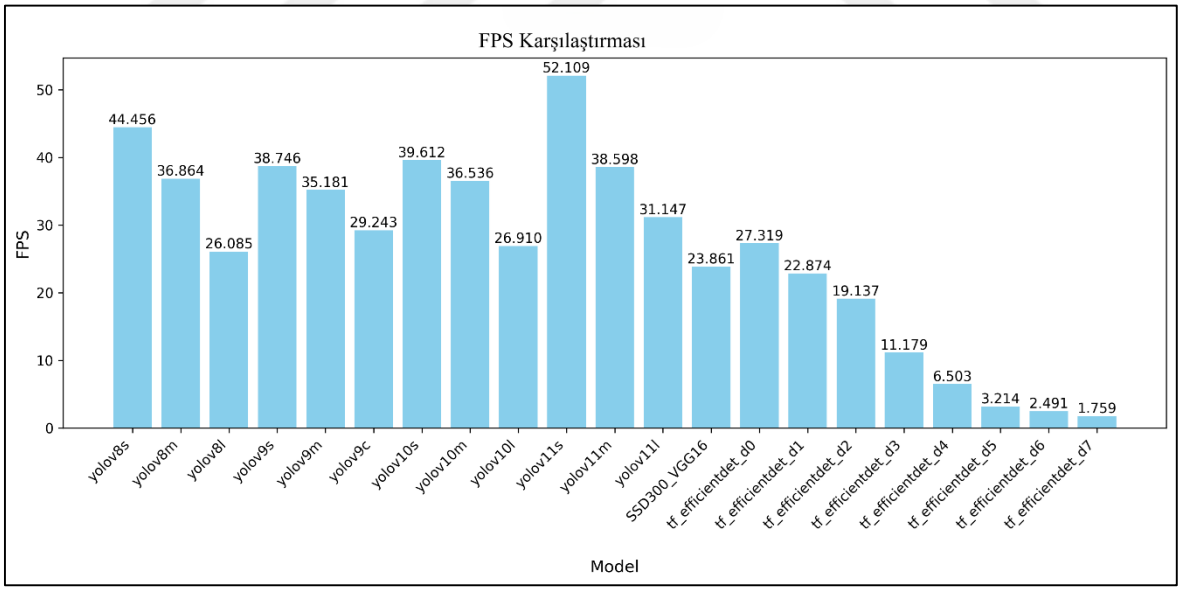
4.5. YOLO, SSD ve EfficientDet Modellerinin Performans Metrikleri Karşılaştırması

Bu bölümde, YOLO (v8, v9, v10, v11), SSD (SSD300-VGG16) ve EfficientDet (D0-D7) modellerinin gerçek zamanlı nesne algılama performansları, çeşitli nicel parametreler üzerinde kapsamlı bir şekilde karşılaştırılmıştır. Karşılaştırmada kullanılan başlıca parametreler, Saniyedeki Kare Sayısı (Frame Per Second – FPS), Çıkarım Süresi (Inference Time), Kesinlik (Precision), Duyarlılık (Recall), F1-Skoru (F1-Score), Ortalama Ortalama Hassasiyet (Mean Average Precision - mAP), ve Kesişim /Birleşim Oranı (Intersection over Union - IoU) değerleridir.

Her bir parametre için grafiksel karşılaştırmalar ayrı alt başlıklar altında sunulmuş ve modellerin güçlü ve zayıf yönleri detaylı bir şekilde analiz edilmiştir. Bu kapsamda, modellerin hem doğruluk hem de hız açısından dengeli bir şekilde değerlendirilmesi amaçlanmıştır. Gerçek zamanlı uygulamalarda kullanılan sistemlerin ihtiyaç duyduğu hız-doğruluk optimizasyonu göz önüne alındığında, bu çok boyutlu analiz kritik önem taşımaktadır. Bu başlık altında yer alan alt başlıklarda her bir performans parametresi özelinde elde edilen bulgular, grafiklerle desteklenerek yorumlanmaktadır.

4.5.1. FPS karşılaştırması

Saniyedeki Kare Sayısı (Frame Per Second – FPS), bir modelin saniyede kaç kare işlediğini göstermek aamcıyla kullanılan bir performans parametresi ve temel bir hız ölçütüdür. Gerçek zamanlı nesne algılama uygulamalarında kritik bir öneme sahiptir. Bu bölümde, YOLO (v8, v9, v10, v11), SSD (SSD300-VGG16) ve EfficientDet (D0-D7) modellerinin FPS değerleri ölçülüp karşılaştırılmış ve elde edilen sonuçlar Şekil 4.23'te grafiksel bir çıktı olarak sunulmuştur.



Şekil 4.23. FPS analiz sonuçları

Grafik incelendiğinde, en yüksek FPS değerinin YOLO11s modelinde elde edildiği ve YOLO11s modelinin hafif yapısı ve optimize edilmiş hesaplama yapısıyla hızlı ve akıcı bir şekilde çalışarak gerçek zamanlı uygulamalarda oldukça avantajlı olduğu görülmektedir. YOLO11s'yi YOLOv8s 44,456 FPS ile ve 39,612 FPS ile YOLOv10s modelleri takip

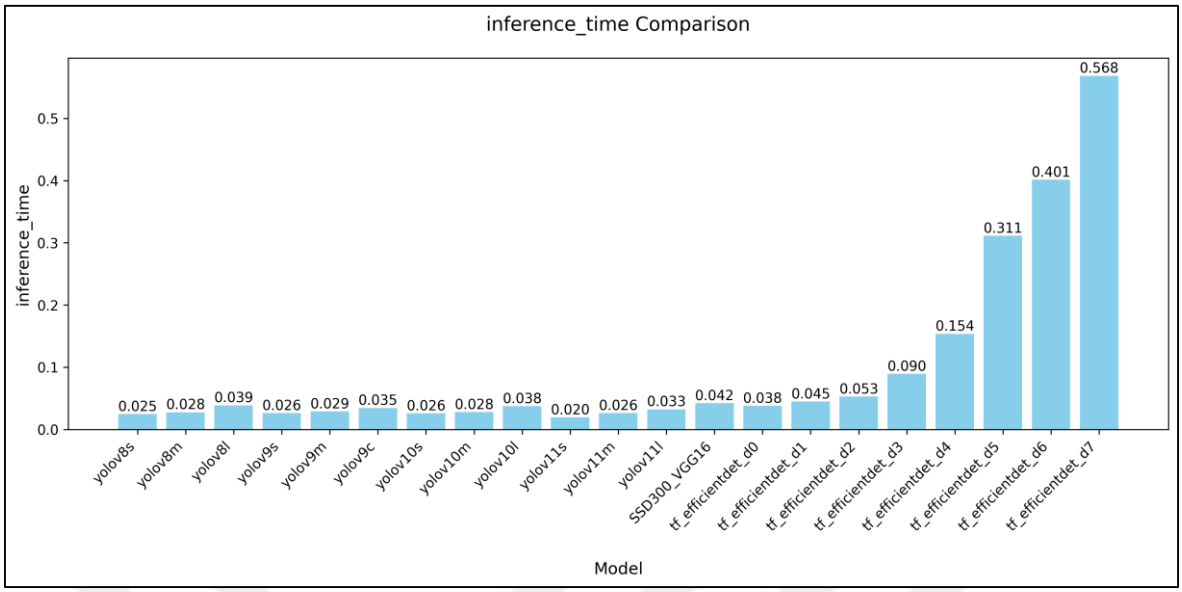
etmektedir. Genel olarak YOLO ailesinin ‘s’ versiyonlarının düşük parametre sayıları sayesinde yüksek hız performansına ulaştığı gözlemlenmektedir. 36,536 ile YOLOv10m, 35,181 FPS ile YOLOv9m ve 36,864 ile YOLOv8m modellerinin hız-performans dengesi açısından tercih edilebilir sonuçlar sunmaktadır.

31,147 FPS değeri ile SSD300-VGG16 modelinin, YOLO modellerine kıyasla daha yavaş kalmaktadır ancak yine de gerçek zamanlı uygulamalarda kullanmak için yeterli bir çalışma hızına sahip olduğu söylenebilmektedir. EfficientDet ailesi incelendiğinde, artan model derinliği ile birlikte FPS değerlerinde kayda değer bir düşüş yaşandığı görülmektedir. 23,861 FPS değeri ile EfficientDet-D0 sınıfının en hızlı modeli olarak öne çıkmaktayken EfficientDet-D7 1,759 FPS değeri ile en yavaş model olarak gerçek zamanlı kullanıma uygun olmadığını ortaya koymaktadır. Bu durum, EfficientDet mimarisinin ölçeklenebilir yapısı gereği model boyutu ve karmaşıklığı arttıkça hesaplama süresinin önemli ölçüde uzadığını kanıtlar nitelikte bir veri sunmaktadır.

Sonuç olarak, bu tez çalışmasında kullanılan modeller karşılaştırıldığında FPS açısından en başarılı performansları YOLO modellerinin sergilemiş olduğu, özellikle küçük boyutlu YOLO sürümlerinin yüksek hızlara ulaşarak gerçek zamanlı sistemlerde kullanım için en uygun adaylar olduğu anlaşılmıştır. EfficientDet modellerinin ise yüksek nitelikli donanım gereksinimlerine sahip sistemlerde ve daha çok doğruluk odaklı senaryolarda kullanımının uygun olacağı görünmektedir.

4.5.2. Çıkarım süresi (inference time) karşılaştırması

Çıkarım süresi (inference time), modelin tek bir görüntü üzerinde tahmin yapması için harcadığı süreyi ifade etmektedir ve saniye (s) cinsinden hesaplanmaktadır. Düşük çıkarım süresi, sistemin daha hızlı tepki verebilme kabiliyetini göstermektedir. Şekil 4.24’te çeşitli nesne algılama modllerine ait hesaplanan çıkarım sürelerinin karşılaştırması sunulmaktadır.



Şekil 4.24. Çıkarım süresi (inference time) analiz sonuçları

Grafik incelendiğinde, en düşük çıkarım sürelerine YOLO ailesine ait modellerin sahip olduğu, aynı FPS değerlerinde olduğu gibi özellikle küçük boyutlu YOLO sürümlerinin en iyi çıkarım süreleriyle çalıştığı ve bunun da sebebinin hafif mimari yapıya sahip ve donanım üzerinde optimize bir şekilde çalışabiliyor olmalarından geldiği görülmektedir. YOLO modellerinin tüm versiyonlarında (v8-v11), s ve m sürümlerinin genel olarak 0,026-0,039 saniye aralığında çıkarım süreleri ile çalışmaktadır.

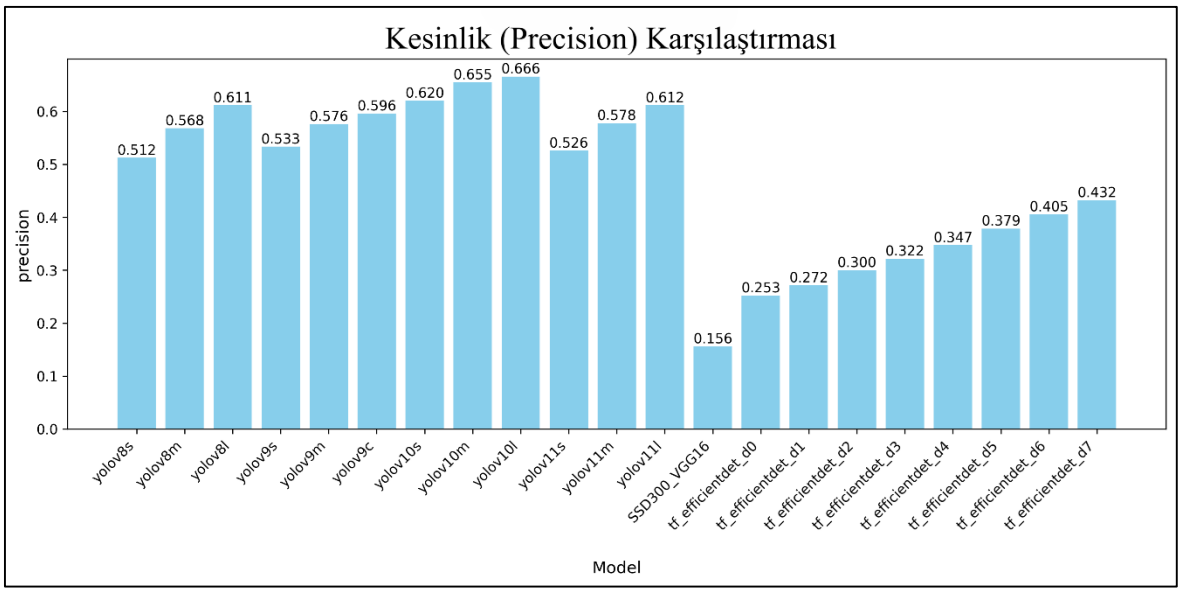
0,038 saniye çıkarım süresi ile klasik bir model olmasına rağmen hala rekabetçi bir seçenek olarak SSD300-VGG16 modelinin YOLO modellerine oldukça yakın bir performans sergilediği gözlemlenmiştir. Buna karşın, EfficientDet modellerinin model derinliği arttıkça çıkarım sürelerinde ciddi artışlar yaşadığı görülmektedir. Bu durum, EfficientDet mimarisinin derin ve geni yapılarının, daha fazla parametre içerdiği ve daha karmaşık hesaplamalr gerektirdiği için kaynaklanmaktadır ve bu da çıkarım süresini doğrudan artırmaktadır. EfficientDet-D0'ın 0,090 saniye ile sınıfı içerisinde en iyi çıkarım süresine sahip olduğu görünmektedir.

Sonuç olarak, en iyi çıkarım sürelerini YOLOv8s ve YOLO11s modellerinin sunduğu ve en yavaş çıkarım süresisinin ise EfficientDet-D7 modeline ait olduğu söylenebilmektedir. Bu sonuçalar elde edilen FPS sonuçaları ile uyum göstermektedir. Gerçek zamalı sistemlerde düşük çıkarım sürelerinin önemli olduğu uygulamalarda YOLO modelleri öne çıkmaktadır. EfficientDet'in büyük modellerinin ise, doğruluğun ön plana çıktığı ama zamana duyarlı

olunmayan uygulamalarda kullanımı daha iyi sonuçlar alınmasını sağlayacaktır. Bu sonuçlar, uygulama ihtiyacına göre model seçiminin önemini göstermektedir.

4.5.3. Kesinlik (precision) analizi

Kesinlik (Precision), yanlış pozitif respitlerin sayısını azaltmaya odaklanan bir performans parametresidir. Şekil 4.25'te test edilen nesne algılama modellerinin kesinlik performansları gösterilmiştir.



Şekil 4.25. Kesinlik (precision) analiz sonuçları

Grafik incelendiğinde, en yüksek kesinlik değerinin YOLO11m (0,666) ve YOLOv10m (0,655) modellerine ait olduğu, bu modelleri sırasıyla YOLO11l (0,620), YOLOv9c (0,596) ve YOLOv9m (0,576) modellerinin takip ettiği görülmektedir. Genel olaral YOLO ailesi modellerinin yüksek kesinlik değerleri sunarak tüm modeller içerisinde üst sıralarda yer aldığı görülmekte olup YOLO modellerinin daha doğru sınıflandırma yaptığı elde edilen sonuçlara dayanılarak söylenebilir. Özellikle YOLO11 sürümlerinin (s, m, l) hem hız hem kesinlik açısından dengeli bir performans gösterdiği görünmektedir.

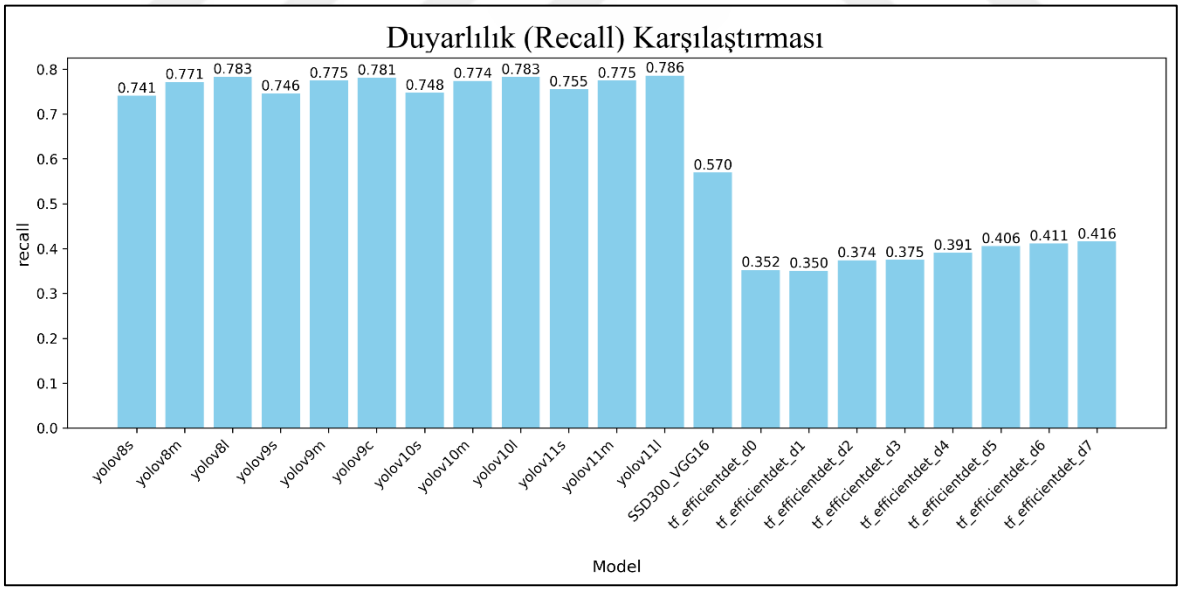
0,526'lık kesinlik değeri ile DDS300-VGG16 YOLO modellerini takip etmekte ve EfficientDet modellernde daha yüksek bir doğruluk sergilemektedir. Bundan dolayı orta seviye kesinlik gereksinimlerinin olduğu uygulamalarda tercih edilebilecek bir model olduğu söylenebilmektedir. Model boyutuna bağlı olarak EfficientDet modellerinde kesinlik

değerinin arttığı gözlemlenmektedir. Bu artış, daha büyük EfficientDet modellerinin daha doğru sınıflandırma yapabildiğini, ancak bununla birlikte düşük hızlarda hesaplamalar yapılacağını ve bunun iyileştirilmesinin de hesaplama maliyetlerini artıracağını ortaya koymaktadır.

Sonuç olarak, YOLO ailesinin hız performansında olduğu gibi sunduğu yüksek kesinliklerle öne çıktığı görülmektedir. EfficientDet modellerinde artan boyutla kesinlik değerlerinde gelişme sağlandığı ancak hala YOLO ve SSD modellerinin gerisinde kaldığı söylenebilmektedir. Bu sonuçlar, uygulama ihtiyaçarına bağlı olarak model seçiminde kesinlik performansının dikkate alınması gerektiğini ortaya koymaktadır.

4.5.4. Duyarlılık (recall) analizi

Duyarlılık (recall), gerçek pozitif örneklerin ne kadarının doğru bir şekilde tespit edildiğini gösteren performans parametresidir. Şekil 4.26'da test edilen nesne algılama modellerinin duyarlılık performansları gösterilmiştir.



Şekil 4.26. Duyarlılık (recall) analiz sonuçları

Grafik incelendiğinde, en yüksek duyarlılık değerinin YOLO11l (0,786) modelinde elde edildiği, bunu YOLOv10l (0,783) ve YOLOv8l (0,783) modellerinin takip ettiği görünmektedir. YOLO modellerinin 'l' versiyonlarının daha yüksek duyarlılık sunduğu ve daha büyük model boyutlarının daha fazla nesneyi tespit etmede başarılı olduğu

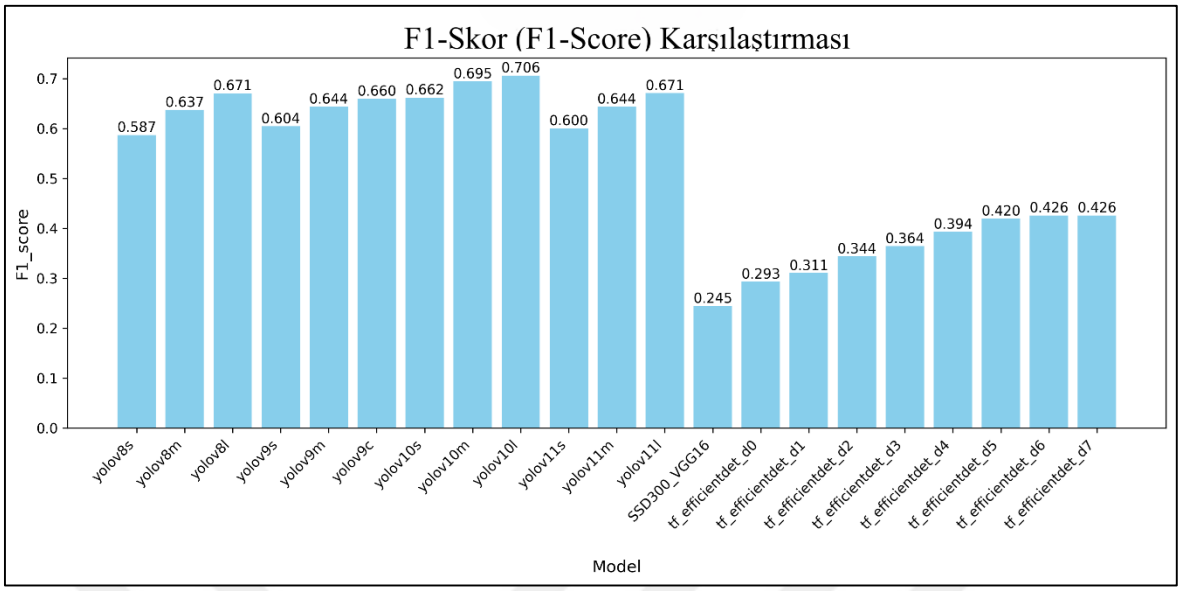
söylenebilmektedir. YOLOv9m (0,775), YOLOv10m (0,774) ve YOLO11m (0,775) gibi orta seviye modellerin yüksek duyarlılık değerleri sunduğu ve bu sayede hız-performans dengesini koruyarak geniş bir kullanım alanına hitap ettikleri anlaşılmaktadır.

0,570'lik duyarlılık değeriyle SSD300-VGG16 modeli YOLO modellerinin gerisinde kalsa da EfficientDet modellerinin hala önünde yer almaktadır. Bu sonuç, SSD mimarisinin hala kabul edilebilir seviyede bir duyarlılık sunduğunu göstermektedir. EfficientDet ailesinde ise boyut artışı ile birlikte duyarlılıkta da genel bir artış olduğu gözlemlenmektedir. Bu artış, modelin karmaşıklığı arttıkça daha fazla nesneyi tespit edebildiğini göstermektedir. Ancak EfficientDet modelleri genel olarak hem YOLO hem de SSD mimarilerinin gerisinde kalmaktadır.

Genel değerlendirme olarak, YOLO ailesinin hız performansında olduğu gibi sunduğu yüksek duyarlılıkla öne çıktığı görülmektedir. Özellikle YOLO11l, YOLOv10l ve YOLOv8, hem büyük model yapıları hem de gelişmiş mimarileri sayesinde yüksek düzeyde gerçek pozitif tespiti gerçekleştirmektedir. EfficientDet modellerinde artan boyutla kesinlik değerlerinde gelişme sağlandığı ancak hala YOLO ve SSD modellerinin gerisinde kaldığı söylenebilmektedir. Bu sonuçlar, uygulama ihtiyaçarına bağlı olarak model seçiminde duyarlılık performansının dikkate alınması gerektiğini ortaya koymaktadır.

4.5.5. F1-Skoru ile denge analizi

F1-Skoru (F1-Score), kesinlik ve duyarlılık parametrelerinin harmonik ortalaması olup, bu ikiölçütün dengeli bir şekilde değerlendirilmesini sağlamaktadır. Bu yönüyle, modelin genel doğruluk performansını yansıtan önemli bir performans parametresidir. Şekil 4.27'da test edilen nesne algılama modellerinin F1-Skor performansları gösterilmiştir.



Şekil 4.27. F1-Skoru analiz sonuçları

Grafik incelendiğinde, 0,706'lık F1-Skor değeri ile yüksek kesinlik ve duyarlılık değerlerini başarılı bir şekilde harmanlayarak genel başarıyı en üst düzeye çıkaran YOLOv10l modeli öne çıkmaktadır. Hemen arkasından YOLOv10m (0,695), YOLOv8l (0,671), YOLO11l (0,671) ve YOLOv9c (0,660) modelleri gelmektedir. Bu sonuçlar, büyük model yapılarının daha kapsamlı ve doğru tespitler yapabildiğini bir kez daha ortaya koymaktadır. YOLOv10 sürümleri bu bağlamda en dengeli sonuçları sunarak öne çıkmaktadır.

SSD300-VGG16 modeli, 0,245'lik F1-Skor değeriyle ciddi anlamda geride kalmış, bu da modelin kesinlik ve duyarlılık dengesini sağlamakta zorlandığını göstermektedir. EfficientDet modelleri ise F1-Skor açısından ortalama seviyelerde yer almıştır. EfficientDet ailesinde model seviyesine göre EfficientDet-D0 (0,293)'dan EfficientDet-D7 (0,426)'ye doğru F1-Skor değerinin arttığı gözlemlenmiştir. Ancak en iyi sonuç veren EfficientDet modeli bile, en kötü sonuç veren YOLO modelleriyle kıyaslandığında oldukça geride kalmaktadır. Bu da EfficientDet modellerinin kesinlik ve duyarlılık dengesini sağlamakta zorlandığını göstermektedir.

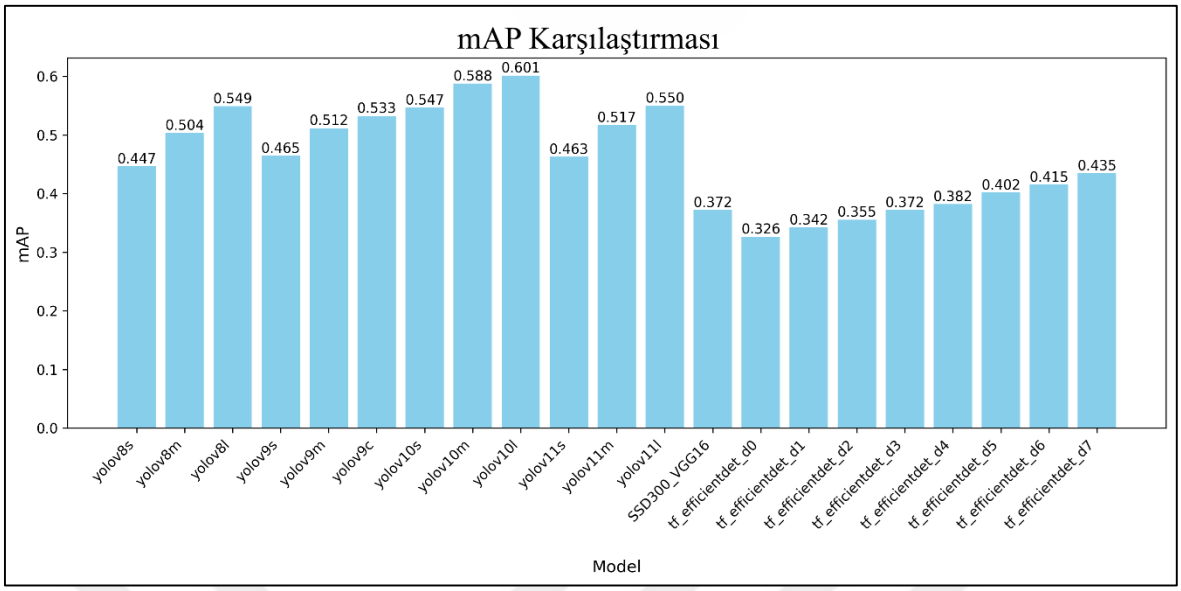
Sonuç olarak, YOLOv10 ve YOLOv9'un özellikle 'm' ve 'l' versiyonları, açık ara en başarılı sonuçları vermektedir. Bu durum, bu modellerin hem hassas hem de kapsamlı bir tespit yeteneğine sahip olduğunu ve genel doğruluk açısından üstün olduklarını ortaya koymaktadır. EfficientDet modellerinin kesinlik ve duyarlılık dengesini sağlamakta

zorlandığı ve YOLO modellerinin gerisinde kaldığı ancak SSD modeline göre daha başarılı sonuçlar sunarak hala kabul edilebilir seviyede performans gösterdiği gözlemlenmiştir.

4.5.6. mAP karşılaştırması

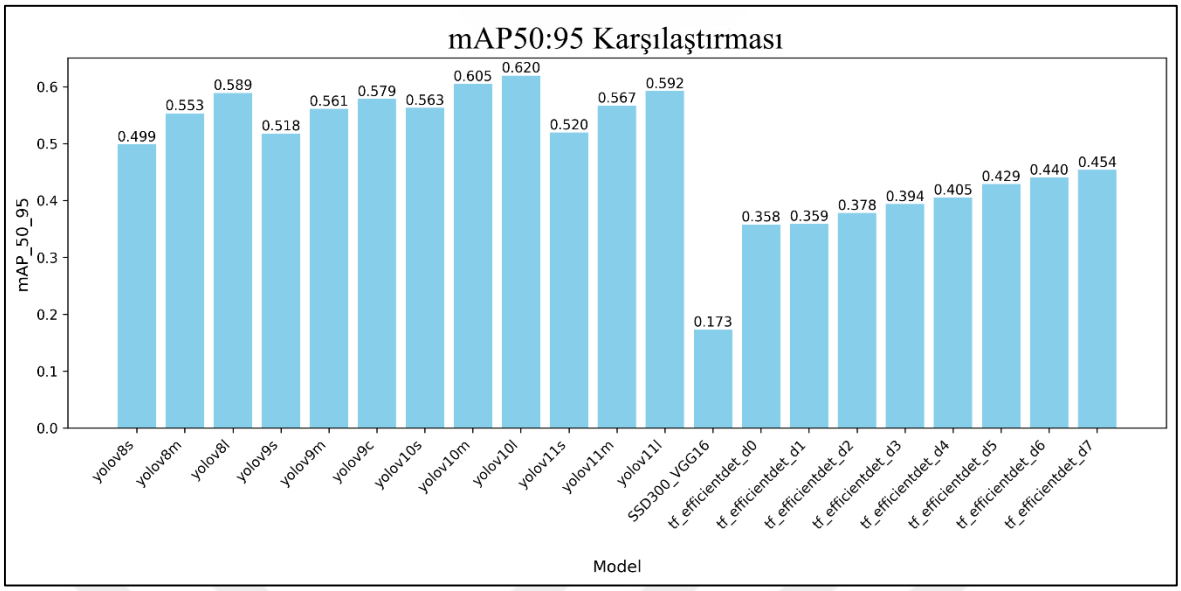
Ortalama Ortalama Hassasiyet (Mean Average Precision - mAP), nesne tespit modellerinin başarımını ölçmede yaygın olarak kullanılan ve modelin sınıflar arasındaki ayrım gücünü değerlendiren bir performans parametresidir. mAP@0,5, IoU eşiğinin 0,5 olarak alındığında elde edilen değeri ifade ederken, mAP50:95 IoU'nun 0,5'ten 0,95'e kadar 0,05 aralıklarla değiştiği ve çok daha sıkı bir değerlendirme sunduğu COCO standardıdır. Bu nedenle mAP50:95 değeri, modelin genel başarımına dair daha tittiz bir gösterge sunmaktadır. Şekil 4.28'de test edilen nesne algılama modellerinin mAP@0,5 değerleri, Şekil 4.29'de test edilen nesne algılama modellerinin mAP0,5:0,95 değerleri gösterilmiştir.

mAP@0,5 değerleri için elde edilen grafik incelendiğinde, en yüksek mAP@0,5 değeri 0,601 ile YOLOv10l modeline ait olup bunu sırasıyla YOLOv10m (0,588) ve YOLOv8l (0,549) takip etmektedir. 0,54'lük mAP@0,5 değerinin üzerinde sonuçlar elde edem YOLOv10 sürümleri, genel olarak tüm modeller arasında en yüksek başarıya sahip grup olarak dikkat çekmektedir. EfficientDet ailesi ise mAP@0,5 açısından diğer modellere kıyasla daha geride kalan bir performans göstermektedir. EfficientDet-D0 modelinin 0,326 ile en düşük değeri, EfficientDet-D7 modelinin 0,435 ile en yüksek değerini aldığı gözlemlenmiştir. Yine burada da diğer metriklerde gözlemlenenle benzer şekilde EfficientDet modllerinin model boyutu ve karmaşıklığı arttıkça performansın da arttığı ancak genel olarak mAP bazında YOLO ailesinin gerisinde kaldığı görülmektedir. SSD300-VGG16 modeli ise 0,372'lik mAP değeriyle YOLO ve EfficientDet mimarileri arasında kabul edilebilir bir yer aldığını göstermektedir ancak bu durum modern nesne algılama mimarilerine göre sınırlı kaldığını bir kez daha ortaya koymaktadır.



Şekil 4.28. mAP analiz sonuçları

COCO standardı olan mAP50:95 değerleri incelendiğinde, performans sıralamasının büyük ölçüde benzer olduğu görülmektedir. 0,620'lik mAP50:95 değerleri ile YOLOv10l modeli en iyi sonucu elde etmiş olup bunu YOLOv10m (0,605) ve YOLOv8l (0,589) modelleri takip etmiştir. Bu sonuçlara bakarak, model karmaşıklığı ve kapasitesi arttığı durumda genel doğruluk performansının da anlamlı bir şekilde yükseldiği söylenebilmektedir. SSD300-VGG16 modeli, mAP50:95 değeri açısından 0,173'lük bir başarı gösterdiği görülmektedir. Bu değer, modern mimarilerin çok altında kalarak modelin COCO gibi kompleks veri setlerinde yetersiz kaldığı söylenebilmektedir. EfficientDet ailesi bu performans parametresinde biraz daha iyi bir görünüm sunarak EfficientDet-D7 0,454 geğeri ile orta seviye bir başarı sağlamıştır. Ancak bu değer yine YOLO modellerinin gerisinde kalmaktadır.



Şekil 4.29. mAP50-95 analiz sonuçları

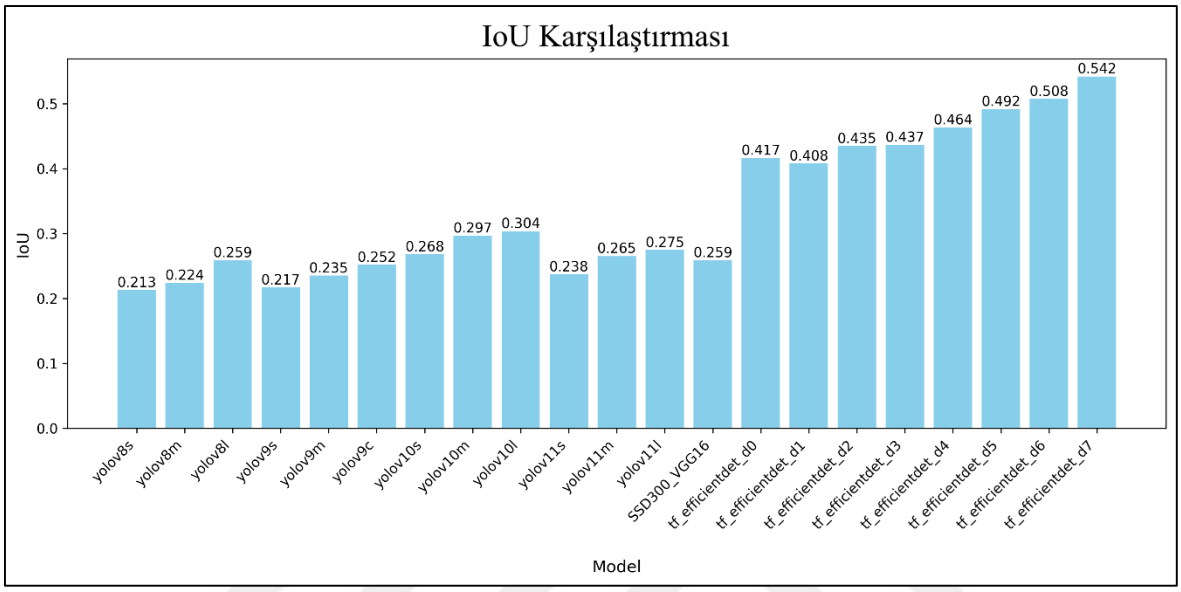
Genel değerlendirme olarak, YOLOv10 sürümlerinin açık ara lider konumunda olduğu, YOLOv8 ve YOLOv9 sürümlerinin de rekabetçi sonuçlar sunarak özellikle ‘m’ ve ‘l’ versiyonlarının başarılı sonuçlar ürettiği gözlemlenmiştir. EfficientDet modellerinin belirli bir istikrarla ilerlemesine rağmen YOLO modellerine kıyasla önemli ölçüde geri planda kaldığı söylenebilmektedir. SSD modeli ise bu performans parametresi açısından oldukça zayıf kalmaktadır ve modern gerçek zamanlı nesne tespit uygulamaları için uygun bir tercih olmamaktadır. Yüksek doğruluk ve genel başarımın hedeflendiği uygulamalarda YOLOv10l ve YOLOv10m modellerinin öne çıktığı görülmektedir. Hem yüksek mAP hem de yüksek F1-skor değerleriyle bu modeller, COCO veri seti üzerinde güçlü bir genel performans sergilemiştir.

4.5.7. IoU karşılaştırması

Kesişim /Birleşim Oranı (Intersection over Union - IoU), tespit edilen sınırlayıcı kutu (bounding box) ile gerçek etiket kutusu arasındaki örtüşme oranını ifade etmektedir. Bu oran ne kadar yüksekse, modelin nesne konumlandırmadaki doğrulupu da o kadar yüksek olmaktadır. Şekil 4.30’da farklı nesne algılama modellerinin ortalam IoU değerleri karşılaştırılmaktadır.

Grafik incelendğinde, en yüksek IoU değerinin 0,542’lik değerle EfficientDet-D7 modeline ait olduğu bu modeli EfficientDet ailesinin diğer modellerinin takip ettiği ve modelin

derinliği ve karmaşıklığı arttıkça IoU değerlerinin de artış gösterdiği görülmektedir. EfficientDet modellerinin IoU değerlerinin tutarlı ve yüksek olması, bu mimarinin özellikle kutu konumlandırma konusunda güçlü olduğunu ve özellikle daha büyük EfficientDet modellerinin daha hassa konumlama yapabildiğini göstermektedir.



Şekil 4.30. IoU analiz sonuçları

SSD300-VGG16, 0,265'lik IoU değeri ile EfficientDet modellerine göre daha düşük performans sergilemektedir. Bu da SSD mimarisinin, konum hassasiyeti açısından sınırlı olduğunu ortaya koymaktadır. YOLO ailesi incelendiğinde, genel olarak EfficientDet modellerine kıyasla daha düşük IoU değerleri elde edildiği, en yüksek değerin 0,304 ile YOLO11l modeline ait olduğu ve bunu YOLOv10m (0,297) modelinin takip ettiği görülmektedir. YOLOv8s (0,213) ve YOLOv9s (0,217) modellerinin tüm modeller arasında en düşük IoU skorlarına sahip olduğu söylenebilmektedir. Bu durum, YOLO mimarisinin yüksek FPS ve düşük gecikme gibi avantajlarına karşın, kutu hassasiyeti konusunda EfficientDet gibi mimarilerin gerisisnde kaldığını göstermektedir. Özellikle hızlı karar vermesi gereken sistemlerde tercih edilen YOLO modelleri, genel olarak daha düşük konumlama doğruluğu ile çalışmaktadır.

Sonuç olarak, EfficientDet modellerinin yüksek IoU başarımı ile öne çıktığını, modelin mimarisi ve derinliğinin doğrudan doğruluk üzerinde belirleyici olduğunu, YOLOv10m, YOLOv10l ve YOLO11l modellerinin rekabetçi IoU skorları ile YOLO mimarisinin son versiyonlarının konumlandırma doğruluğu açısından gelişme gösterdiğini söylemek

mümkündür. Ayrıca, SSD mimarisinin hem eski hem de düşük derinlikli yapısından dolayı modern modellere göre IoU açısından sınırlı kaldığı görülmektedir. Tüm bu farklılıklar, sistemin gereksinimlerine göre model seçiminde önemli bir kriter olarak değerlendirilmelidir.

4.5.8. Performans metriklerinin grafiksel analizi ve genel değerlendirme

Bu çalışmada, MS COCO 2017 veri seti ve PyTorch ortamı üzerinde eğitilen çeşitli nesne tespit modelleri kullanılmış ve bu modellerin performans metriklerine dayalı kapsamlı bir karşılaştırmalı analiz sunulmuştur. Değerlendirmelerin güvenilirliğini ve doğruluğunu sağlamak için tüm modeller aynı test ortamında test edilmiştir. Çalışma kapsamında YOLO, SSD ve EfficientDet algoritmalarına ait farklı modeller incelenmiştir. Performans metrikleri arasında Saniye Başına Kare (FPS), çıkarım süresi, kesinlik, geri çağırma, F1-Skoru, Ortalama Kesinlik (mAP), mAP50:95 ve Birleşim Üzerinde Kesişim (IoU) yer almaktadır. Hız performansını değerlendirmek için FPS ve çıkarım süresi, algılama doğruluğu için kesinlik, geri çağırma ve F1-Skoru, gelişmiş metrik analizi için ise mAP ve IoU verileri kullanılmıştır. Bu analiz sonucunda elde edilen performans sonuçları karşılaştırmalı bir şekilde sunulmuş ve video verileri üzerinde gerçekleştirilen nesne tespiti sonuçları ile doğrulanmıştır. Bu yaklaşımla çalışmanın gerçek zamanlı uygulamalarda uygulanabilirliği, video tabanlı gerçek zamanlı nesne tespiti sonuçlarıyla ortaya konulmaktadır.

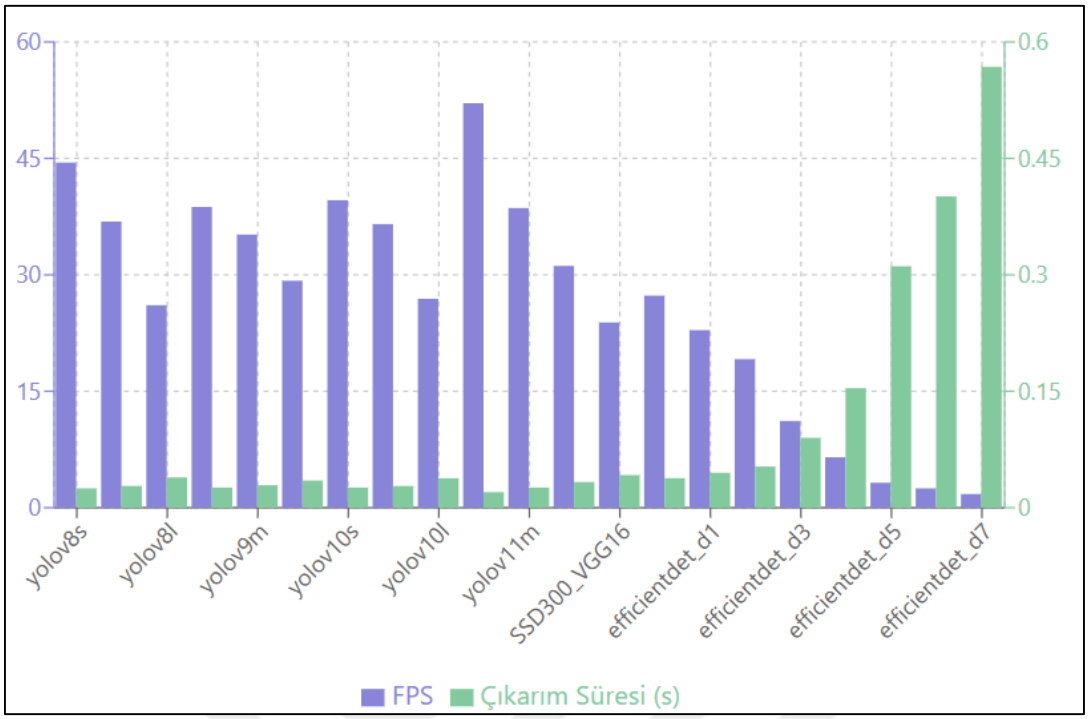
Modellerin güçlü ve zayıf yönleri analiz edilerek, gerçek zamanlı uygulamalarda kullanım potansiyelleri değerlendirilmiştir. Karşılaştırmalı bir şekilde değerlendirilen nesne algılama algoritmalarının kapsamlı ve tarafsız bir değerlendirmesi yapılması ve uygulayıcıların kendi uygulamaları için en uygun algoritmayı seçmelerine destek olunması amaçlanmıştır. Performans metriklerinden elde edilen sonuçlar, Python'daki Matplotlib ve Seaborn kütüphaneleri kullanılarak grafiklerle görselleştirilmiştir.

Hız performans analizi

Nesne algılama modellerinin sadece doğruluğu değil, aynı zamanda gerçek zamanlı çalışma kapasitesi de birçok açıdan kritik öneme sahiptir. Özellikle gömülü sistemler, insansız hava araçları, akıllı kamera sistemleri ve robotik uygulamalarda yüksek FPS ve düşük çıkarım süresi büyük avantaj sağlamaktadır. Modellerin hız performanslarını değerlendirmek için

Şekil 4.31 ve Çizelge 4.1'de test edilen tüm modellerin ortalama FPS değerleri ile her bir görüntü için ortalama çıkarım süresi karşılaştırmalı olarak sunulmaktadır.

YOLO11s modeli, hem en yüksek FPS değeri (52,109) hem de en düşük çıkarım süresi (0,020 s) ile bu karşılaştırmadaki en hızlı model olarak öne çıkmaktadır. Genel olarak tüm YOLO modellerinin FPS değerlerinin 30-45 aralığında seyrettiği, çıkarım sürelerinin 0,025-0,039 aralığında olduğu ve bu sebeple oldukça dengeli performans sunduğu söylenebilmektedir. Daha küçük ve orta YOLO modellerinin büyük modellere kıyasla daha hızlı çalıştığı görülmektedir. SSD300 modeli 23,861 FPS ve 0,042 saniyelik çıkarım süresi ile orta düzey bir performans sergilemektedir. YOLO modellerine kıyasla daha yavaş ve daha geç tepki veren bir yapıdadır. Ancak yine de EfficientDet modellerine kıyasla daha hızlı çalışmaktadır. EfficientDet modellerinde model derinliğinin artması ile birlikte çıkarım süresinin de logaritmik bir şekilde arttığı, FPS değerinin ise kesnkin bir şekilde azaldığı, en iyi çıkarım süresine sahip olan Efficient-D0 (0,038 s)'in bile çıkarım süresi bakımından YOLO modellerine kıyasla daha yavaş kaldığı görülmektedir. EfficientDet-D7, 1,759 FPS ve 0,568 saniye çıkarım süresi ile gerçek zamanlılık açısından en düşük performansa sahip model olarak gösterilebilmektedir. Bu bağlamda, yüksek doğruluk sağlamalarına rağmen gerçek zamanlı sistemlerde kullanım için uygun olmadıkları söylenebilir.



Şekil 4.31. Hız performans analiz sonuçları

Çizelge 4.1. Hız performans parametreleri hesaplama sonuçları

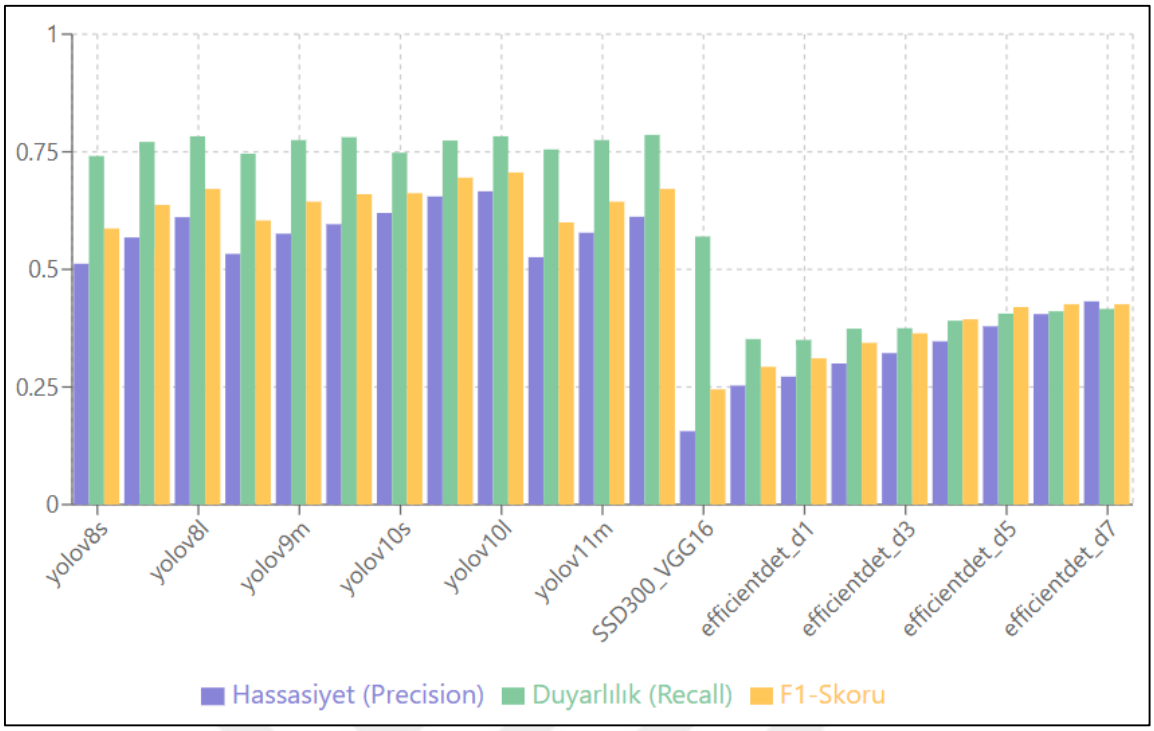
Model Adı	FPS	Çıkarım Süresi (sn)
yolov8s	44,456	0,025
yolov8m	36,864	0,028
yolov8l	26,085	0,039
yolov9s	38,746	0,026
yolov9m	35,181	0,029
yolov9c	29,243	0,035
yolov10s	39,612	0,026
yolov10m	36,536	0,028
yolov10l	26,910	0,038
yolo11s	52,109	0,020
yolo11m	38,598	0,026
yolo11l	31,147	0,033
SSD300_VGG16	23,861	0,042
Efficientdet-D0	27,319	0,038
Efficientdet-D1	22,874	0,045
Efficientdet-D2	19,137	0,053
Efficientdet-D3	11,179	0,090
Efficientdet-D4	6,503	0,154
Efficientdet-D5	3,214	0,311
Efficientdet-D6	2,491	0,401
Efficientdet-D7	1,759	0,568

Gerçek zamanlı uygulamalarda FPS değerinin 30 ve üzerinde olması gerektiği bilindiği için sonuç olarak, özellikle YOLO modellerinin küçük versiyonlarının gerçek zamanlı çalışmaya en uygun modeller olduğu, EfficientDet modellerinde ise EfficientDet-D3'ten itibaren bu gereksinimin karşılanamadığı ifade edilebilir. Ayrıca YOLO11s modelinin gerçek zamanlı sistemlerde kullanılması en uygun model olduğu da belirtilmelidir.

Algılama doğruluğu analizi

Nesne algılama sistemlerinin değerlendirilmesinde hız kadar doğruluk metrikleri de kritik rol oynamaktadır. Algılama doğruluğu, kesinlik, doğruluk ve F1-Skoru parametreleri kullanılarak değerlendirilir ve sonuçlar Çizelge 4.2'de ve Şekil 4.32'de gösterilmektedir.

YOLO modelleri, genel olarak yüksek duyarlılık (recall) (%75-78) ve görece yüksek kesinlik (precision) değerleri ile en tüksek F1-Skor değerlerini elde etmiştir. En yüksek doğruluk değeri YOLOv10l (F1-Skor: %70,6) modeline aittir. Bu model, aynı zamanda tüm modeller arasında en dengeli kesinlik-duyarlılık dengesine sahip olan modellerden biri olarak göze çarpmaktadır. YOLOv10sürümleri kendisinden önceki sürümlere kıyasla belirgin bir iyileştirmeye sahiptir. YOLO11s, hız açısında ön plana çıkmasına rağmen, F1-Skor değeri %60,0 ile en düşük doğruluklu YOLO modelidir. Bu durum, çok hızlı çalışan küçük modellerin doğruluk açısından ödün verildiğini göstermektedir. SSD300-VGG16 modelinin %24,5 F1-skor ve %15,6 kesinlik değeri ile hem yüksek yanlış pozitif tespit hem de yüksek miktarda eksik tespit oranlarına sahip olduğu gözlemlenmiştir. Bu sonuçlarda, SSD'nin hem hız açısından hem de doğruluk açısından güncel modellere kıyasla yetersiz kaldığı çıkarılmaktadır. EfficientDet modelleri, derinlik arttıkça doğruluk açısından gözle görülür bir artış göstermektedir. En iyi F1-Skor değerleri EfficientDet-D6 ve D7 (%42,6) modellerinde gözlemlenmiştir. EfficientDet ailesi, hızdan ödün vererek doğruluk kazanan bir yapı sergilemektedir. Bu nedenle, doğruluk odaklı ancak gerçek zamanlı olmayan uygulamalarda kullanımın daha uygun olacağı söylenebilmektedir.



Şekil 4.32. Algılama doğruluğu analiz sonuçları

Çizelge 4.2. Algılama doğruluğu parametreleri hesaplama sonuçları

Model Adı	Kesinlik (%)	Duyarlılık (%)	F1-Skoru (%)
yolov8s	51,2	74,1	58,7
yolov8m	56,8	77,1	63,7
yolov8l	61,1	78,3	67,1
yolov9s	53,3	74,6	60,4
yolov9m	57,6	77,5	64,4
yolov9c	59,6	78,1	66,0
yolov10s	62,0	74,8	66.2
yolov10m	65,5	77,4	69,5
yolov10l	66,6	78,3	70,6
yolo11s	52,6	75,5	60.0
yolo11m	57,8	77,5	64.4
yolo11l	61,2	78,6	67.1
SSD300_VGG16	15,6	57,0	24,5
Efficientdet-D0	25,3	35,2	29,3
Efficientdet-D1	27,2	35,0	31.1
Efficientdet-D2	30,0	37,4	34,4
Efficientdet-D3	32,2	37,5	36,4
Efficientdet-D4	34,7	39,1	39,4
Efficientdet-D5	37,9	40,6	42,0
Efficientdet-D6	40,5	41,1	42.6
Efficientdet-D7	43,2	41,6	42,6

Sonuç olarak özetleme yapmak gerekirse, YOLOv10l modelinin doğruluk açısından en yüksek F1-Skor (%70,6) değerine sahip olarak en iyi doğruluk modeli olarak gösterlebileceği, YOLOv10m'nin hem yüksek F1-Skor (%69,5) hem de yeterli hız (36,5 FPS) ile doğruluk-hız dengesini en iyi sağlayan model olduğu, SSD300'ün hem doğruluk hem hız açısından modern modellerle kıyaslandığında en zayıf model konumunda kaldığı ve EfficientDet-D7'nin en iyi F1-Skoruna sahip olmasına rağmen gerçek zamanlılık ihtiyacını karşılayamadığı söylenebilmektedir.

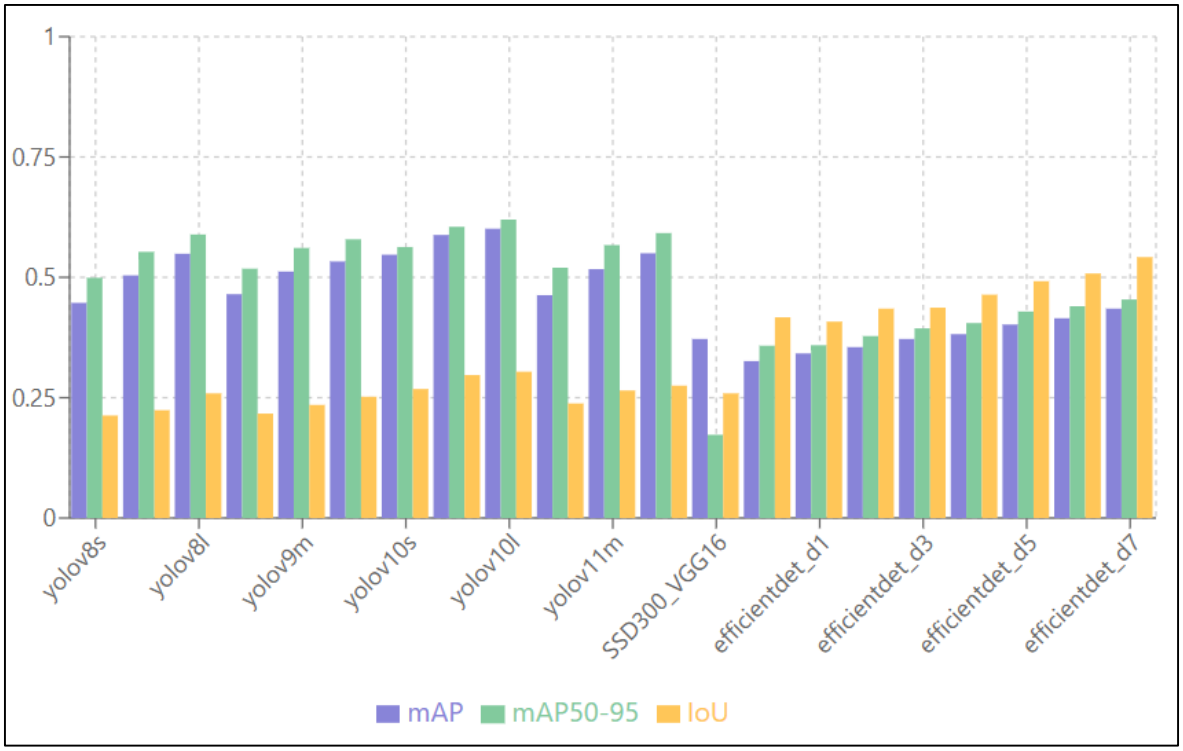
Nesne yerelleştirme analizi

Nesne yerelleştirme analizi için IoU, mAP@0,5 ve mAP50:95metrikleri kullanılmış ve sonuçlar Çizelge 4.3'te ve Şekil 4.33'te gösterilmiştir. YOLO model ailesi genel olarak yüksek mAP@0,5 ve mAP50:95 değerleriyle dikkat çekmektedir. Özellikle YOLOv10 ve YOLO11 serileri, yerelleştirme başarımı açısından önceki versiyonlara göre anlamlı bir gelişme göstermektedir. Ayrıca, her YOLO sürümü kendi içerisinde ayrı ayrı incelendiğinde small versiyonlardan medium ve large versiyonlara geçildikçe de mAP@0,5 ve mAP50:95 değerlerinde artış olduğu gözlemlenmektedir. En yüksek mAP değerlerine sahip model %60,1 değeri ile YOLOv10l olurken, onu %58,8 ile YOLOv10m ve %55,0 ile YOLO11l modeleri takip etmektedir. Bu durum YOLOv10 ve YOLO11 modellerinin nesne algılama başarımı açısından oldukça etkili olduğunu göstermektedir. Ancak IoU değerleri incelendiğinde, YOLO modellerinin kutuları nesnelerle tam olarak örtüştürme konusunda EfficicentDet modellerine kıyasla daah düşük başarı sağladığı görülmektedir. Örneğin YOLOv10l modeli %30,4 IoU değerine sahipken, bu değer EfficientDet-D7 modeli için %54,2'ye kadar çıkmıştır.

EfficientDet ailesi ise özellikle IoU metriğinde ön plana çıkmaktadır. EfficientDet-D0'dan D7'ye doğru gidildikçe hem mAP hem de IoUdeğerlerinde istikrarlı bir artış gözlemlenmektedir. En yüksek IoU değeri %54,2 ile EfficientDet-D7 modeline aittir. Bu da EfficientDet modellerinin nesne kutularını daha hassas ve gerçek konumlarına yakın şekilde yerleştirdiğini göstermektedir. Bununla birlikte, mAP değerleri açısından YOLO modellerinin gerisinde kalmaktadır. Bu durum, EfficientDet modellerinin yerelleştirme başarımı yüksek olsa da genel tespit doğruluğu açısından daha zayıf olduğunu ortaya koymaktadır.

SSD300-VGG16 modeli ise her üç metrikte de diğer modellerin gerisinde kalmıştır. Özellikle mAP50:95 değerleri %17,3 ile oldukça düşüktür. Bu da modelin hem genel tespit doğruluğunun hem de farklı IoU eşikleinde kararlılığının sınırlı olduğunu göstermektedir.

Sonuç olarak, yüksek doğruluk (mAP) gerektiren senaryolarda YOLOv10 ve YOLO11 modellerinin kullanımının uygun olacağı, yüksek yerelleştirme doğruluğunun (IoU) gerektiren senaryolarda ise EfficientDet-D7 modelinin kullanımının daha doğru bir tercih olacağı söylenebilmektedir. Buna ek olarak, SSD modelinin ise güncel nesne algılama ihtiyaçlarını karşılamakta yetersiz kaldığı görünmektedir.



Şekil 4.33. Nesne yerelleştirme analiz sonuçları

Çizelge 4.3. Nesne yerelleştirme parametreleri hesaplama sonuçları

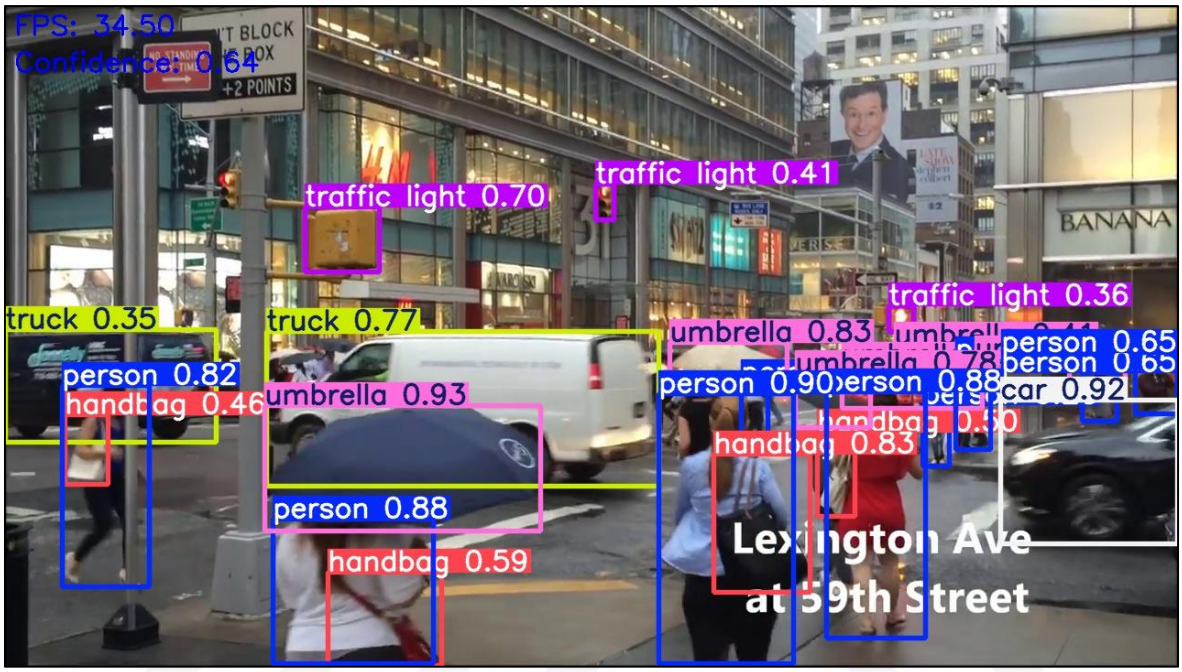
Model Adı	mAP (%)	mAP50-95 (%)	IoU (%)
yolov8s	44,7	49,9	21,3
yolov8m	50,4	55,3	22,4
yolov8l	54,9	58,9	25,9
yolov9s	46,5	51,8	21,7
yolov9m	51,2	56,1	23,5
yolov9c	53,3	57,9	25,2
yolov10s	54,7	56,3	26.8
yolov10m	58,8	60,5	29,7
yolov10l	60,1	62,0	30,4
yolo11s	46,3	52,0	23,8
yolo11m	51,7	56,7	26,5
yolo11l	55,0	59,2	27,5
SSD300_VGG16	37,2	17,3	25,9
Efficientdet-D0	32,6	35,8	41,7
Efficientdet-D1	34,2	35,9	40,8
Efficientdet-D2	35,5	37,8	43,5
Efficientdet-D3	37,2	39,4	43,7
Efficientdet-D4	38,2	40,5	46,4
Efficientdet-D5	40,2	42,9	49,2
Efficientdet-D6	41,5	44,0	50,8
Efficientdet-D7	43,5	45,4	54,2

4.6. Zorlu Senaryolarda YOLO, SSD ve EfficientDet Modellerinin Karşılaştırılması

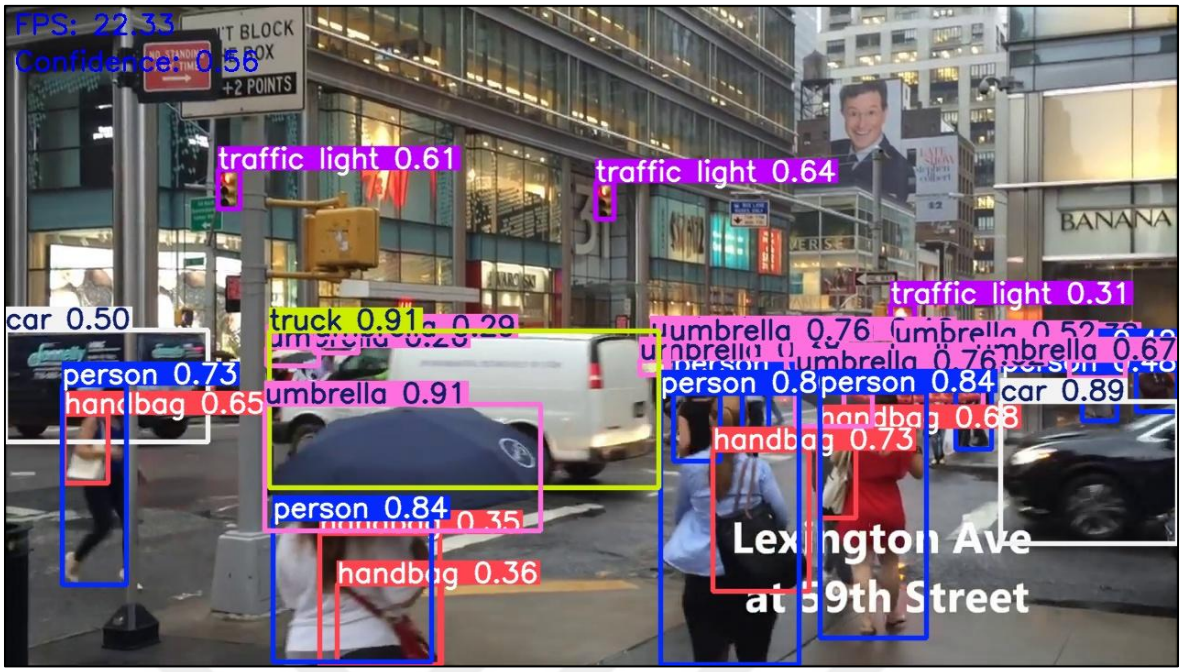
Gerçek dünya uygulamalarında, nesne algılama sistemleri çoğu zaman standart veri setlerinde yer alan ideal ve net görüntülerden oldukça farklı ve dinamik koşullarda çalışmak zorundadır. Bu durum, modelin sadece temel doğruluk metrikleri açısından değil, aynı zamanda dinamik, karmaşık ve değişken ortamlarda gösterdiği tepki açısından da değerlendirilmesi gerekliliğini beraberinde gerektirmektedir. Bu bölümde, YOLO, SSD ve EfficientDet modelleri, çeşitli zorlu senaryolar altında test edilerek karşılaştırılmaktadır. Kullanılan senaryolar arasında yüksek nesne yoğunluğu, hızlı hareket veya görüntü geçişleri ve karmaşık arka planlar gibi unsurlar yer almaktadır. Her modelin bu koşullar altındaki başarımı, hem algılamada süreklilik hem de görsel doğruluk açısından analiz edilmektedir.

Nesnelerin sahne içerisinde sayıca fazla, sıkışık ve çakışmalı yerlerde bulunduğu yoğun ortamlarda, modellerin hem nesne ayırt etme yeteneği hem de üst üste binen kutuları ayrıştırabilme başarısı gerçek zamanlı nesne algılama yapan sistemlerde büyük bir önem kazanmaktadır. Bu bağlamda yapılan testlerde kullanılan video görüntüsünde özellikle

kalabalık yaya grupları ve trafik sahnelerinde inceleme yapılmıştır. Şekil 4.34 ve Şekil 4.35'te gösterildiği gibi YOLOv10l ve YOLO11l modellerinin, bu senaryolarda oldukça başarılı sonuçlar ürettiği gözlemlenmiştir. Bu modeler, nesne sınır kutularını oldukça tutarlı biçimde çizerken, sınıflandırma etiketlerinde de yüksek doğruluk elde etmiştir.

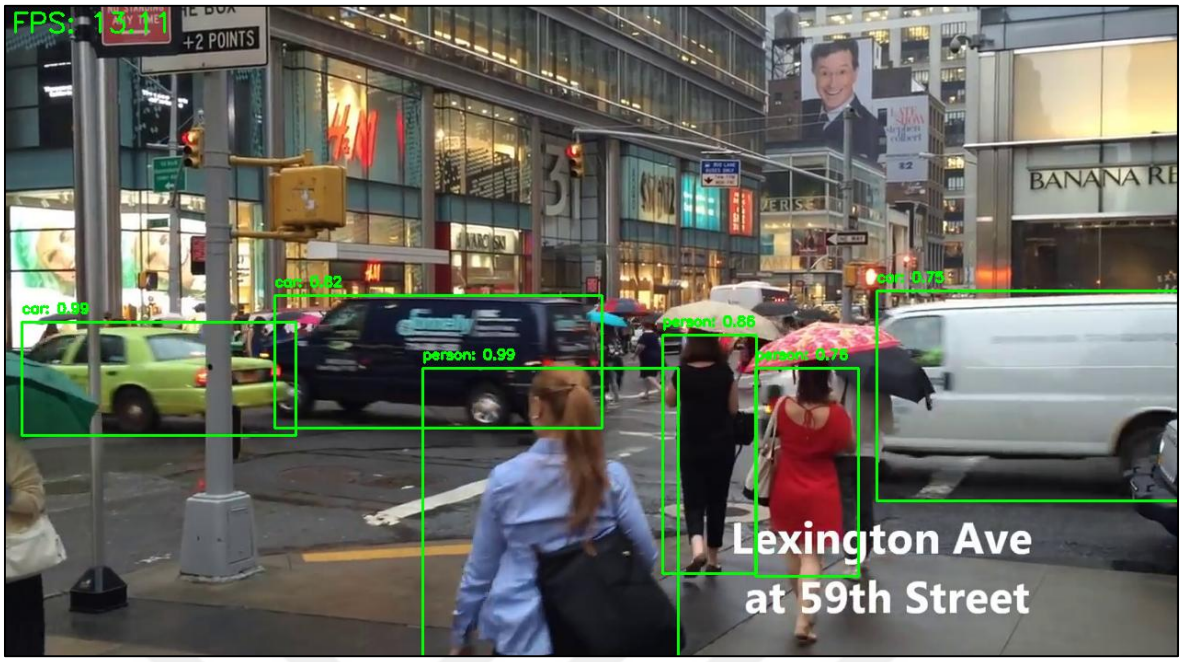


Şekil 4.34. Yoğun nesne senaryosunda YOLOv10l modeli ile ile nesne algılama ekran görüntüsü



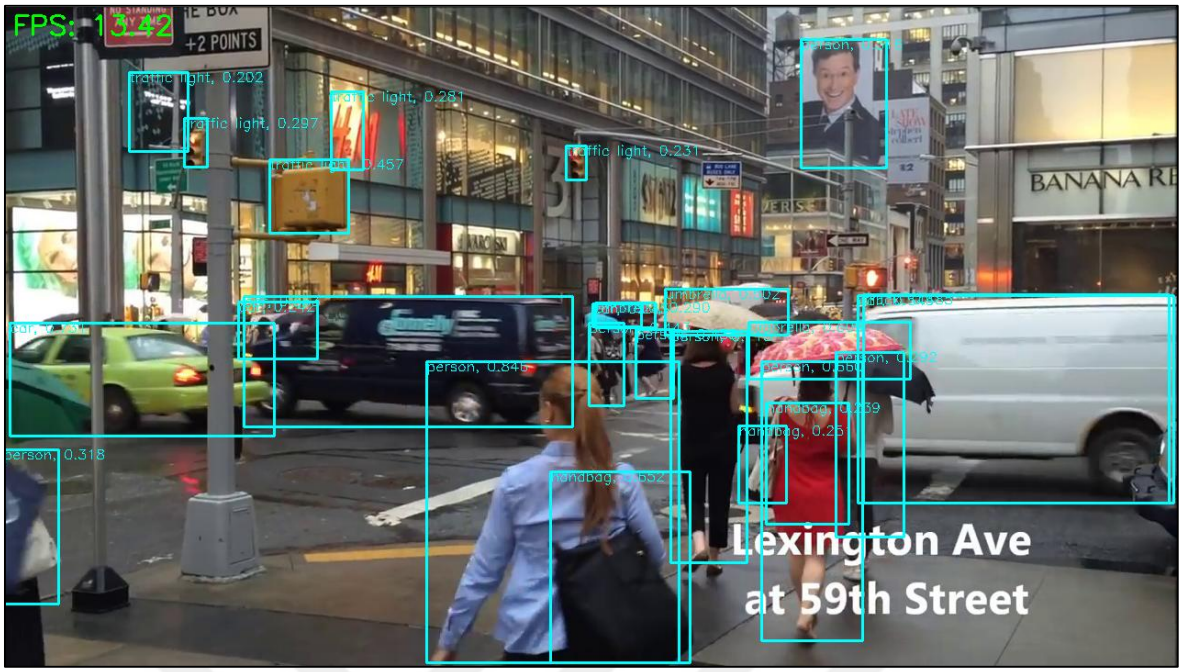
Şekil 4.35. Yoğun nesne senaryosunda YOLO11l modeli ile ile nesne algılama ekran görüntüsü

Öte yandan SSD300-VGG16 modeli, Şekil 4.36'dan de görüleceği üzere benzer yoğunluk seviyelerinde birçok nesneyi ya tam olarak tespit edememiş ya da yanlış sınıflandırmıştır ve nesne yoğunluğundan dolayı tespit edebildiği nesne oranı YOLO modellerinin oldukça gerisisnde kalmıştır.

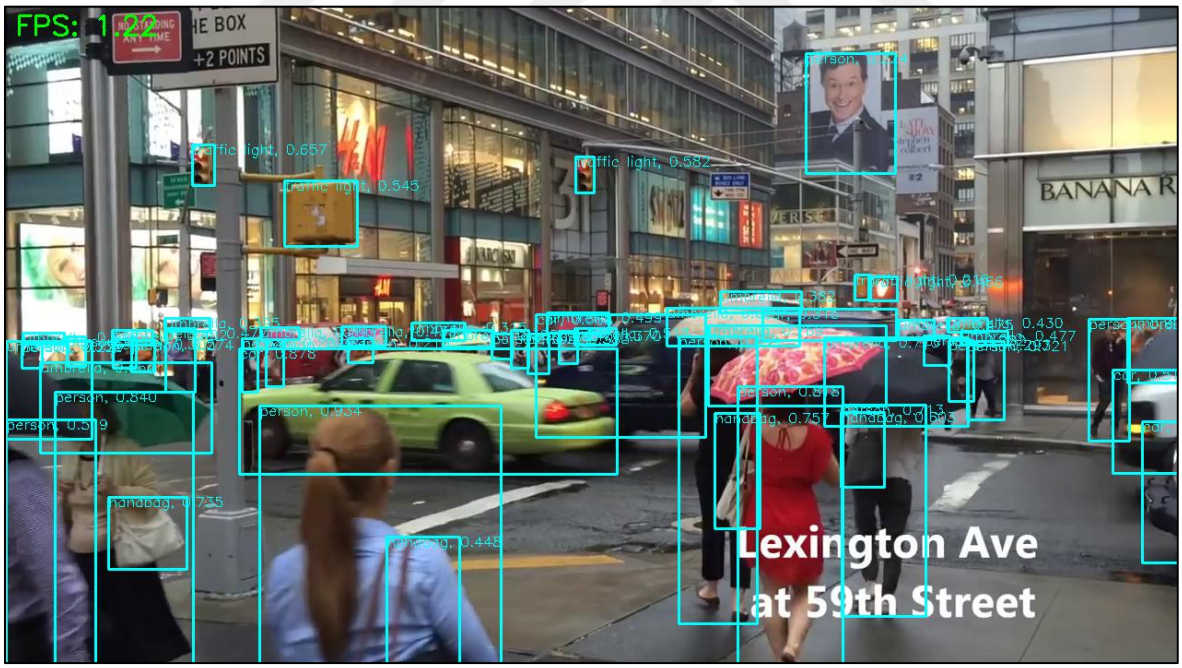


Şekil 4.36. Yoğun nesne senaryosunda SSD300-VGG16 modeli ile ile nesne algılama ekran görüntüsü

EfficientDet modelleri Şekil 4.37'de sunulduğu gibi, özellikle D0-D3 arası sürümlerde sınırlı tespit başarımı göstermiştir, ancak Şekil 4.38'den görülebileceği üzere D5 ve üzeri modeler daha detaylı ve doğru sonuçlar vermiştir. Detaylı ve daha doğru sonuçların elde edilebilmesi, yüksek işlem süresi ve maliyeti oluşmasına sebep olmuştur. Görsel çıktılarda YOLO modelleri daha yoğun, net ve az sayıda hata içeren kutular çizerken, SSD ve EfficientDet modellerinde ise eksik veya kaymış kutular daha sık gözlemlenmiştir.



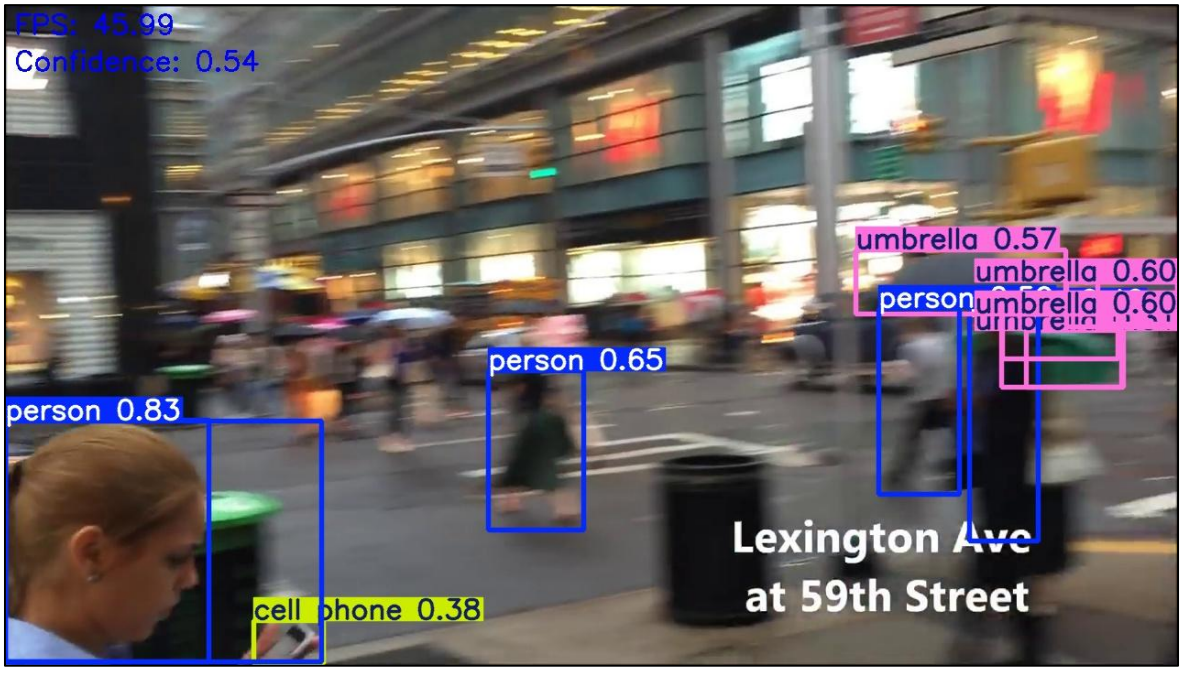
Şekil 4.37. Yoğun nesne senaryosunda EfficientDet-D0 modeli ile ile nesne algılama ekran görüntüsü



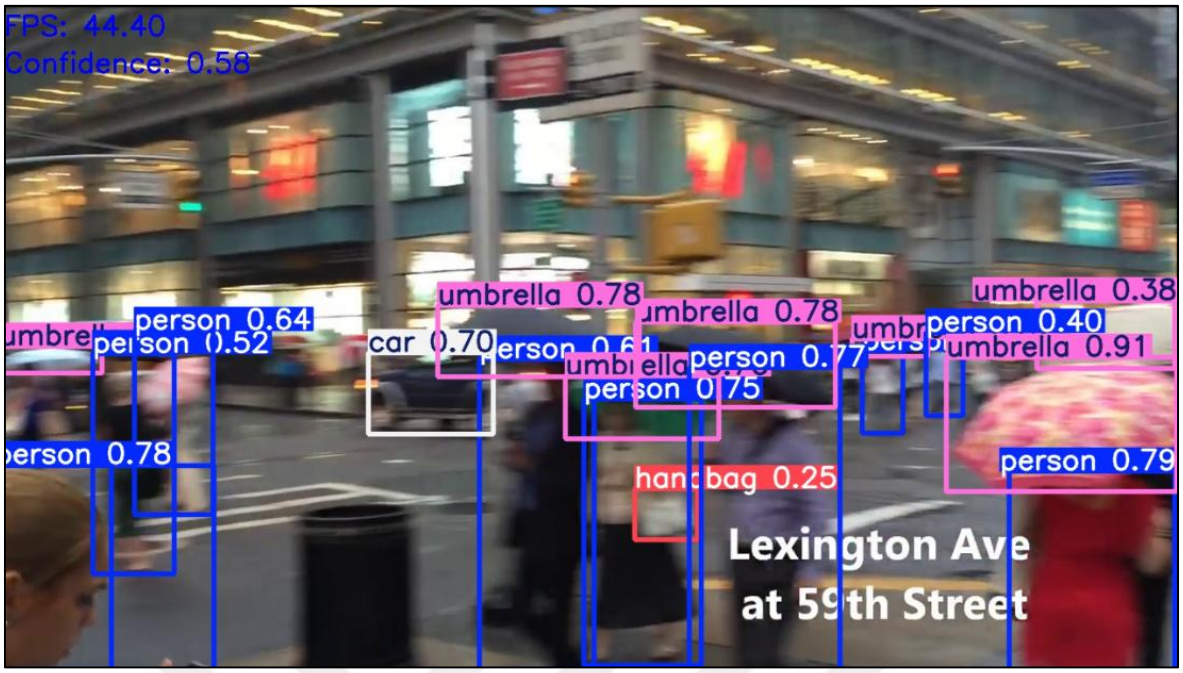
Şekil 4.38. Yoğun nesne senaryosunda EfficientDet-D7 modeli ile ile nesne algılama ekran görüntüsü

Video analizinde karşılaşılan zorlu durumlardan biri de sahnede hızlı hareket eden nesneler, kamera kayması veya ani sahne geçişleri gibi durumların model performanına etkisidir. Bu tip durumlar, çıkarım süresi yüksek olan modellerin kare atlamasına veya geç tepki vermesine neden olmaktadır. Yapılan testlerde, Şekil 4.39'da sunulduğu üzere, YOLO

ailesinin YOLOv8s, YOLOv9s ve YOLOv10m gibi küçük ve orta boyutlu modelleri hızlı sahne değişimlerinde stabil kalmayı başarmış ve nesne tespitlerinde gecikmesiz çalışabilmiştir. Özellikle YOLO11s modeli, düşük çıkarım süresi ve yüksek FPS değeri sayesinde bu tür sahnelerde hem kutu konumlandırmasında hem de etiket doğruluğunda daha kararlı sonuçlar üretmiştir ve alınan ekran görüntüsü Şekil 4.40'ta verilmiştir.

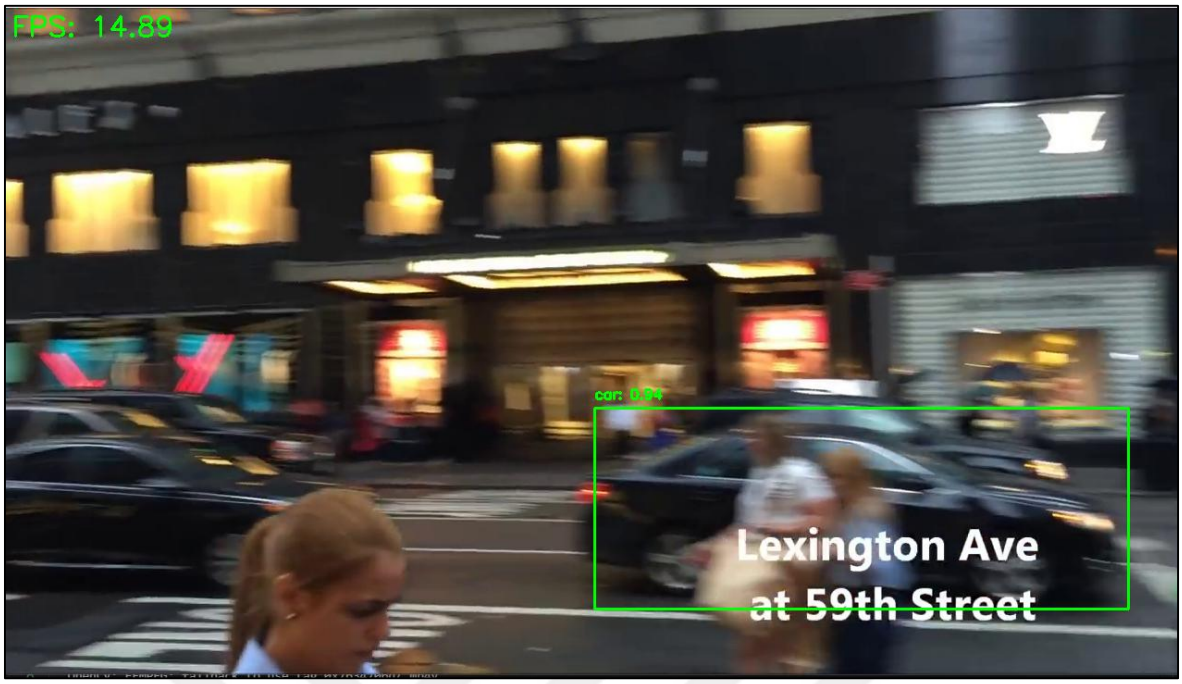


Şekil 4.39. Yoğun nesne senaryosunda YOLOv8s modeli ile ile nesne algılama ekran görüntüsü

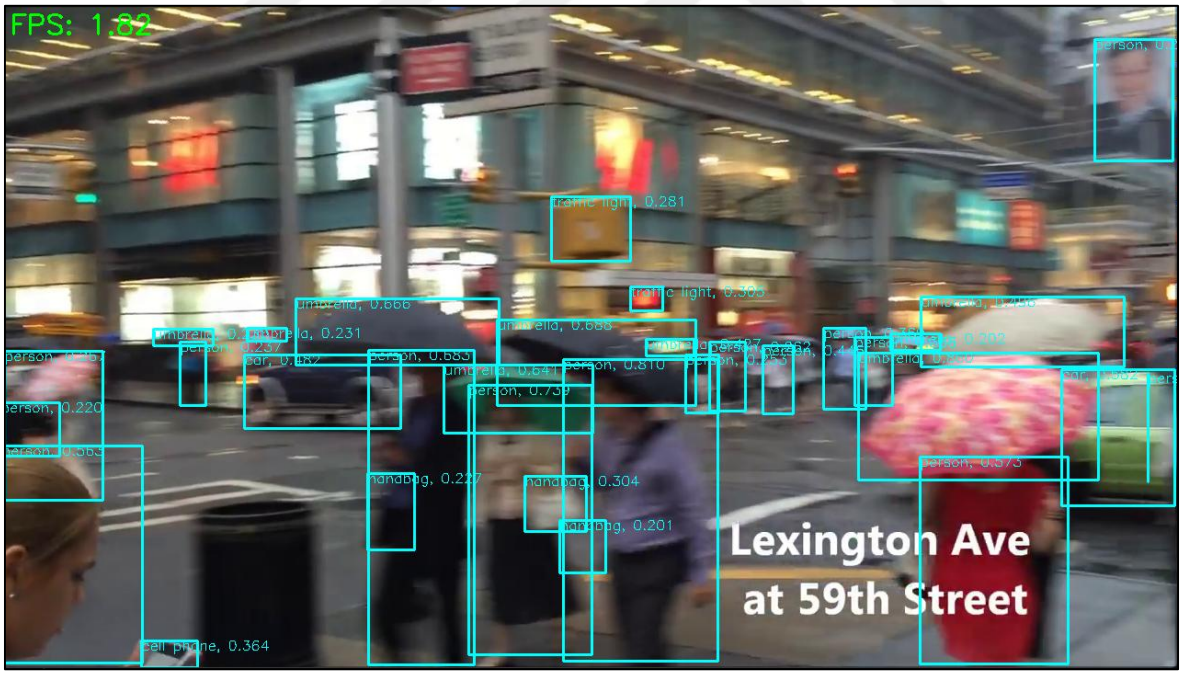


Şekil 4.40. Yoğun nesne senaryosunda YOLO11s modeli ile ile nesne algılama ekran görüntüsü

SSD300-VGG16 modeli, sahne değişiminin hızlandığı durumlarda sıkça hatalı etiketleme yapmış ve bazı nesneleri tamamen kaçırmıştır ve Şekil 4.41'den de görüleceği üzere diğer modellere kıyasla düşük oranda nesne tespiti yapabilmiştir. EfficientDet modelleri, özellikle D4 ve üzeri modellerde hareketli sahnelerde yüksek detaylı sonuçlar üretmiş olsa da Şekil 4.42'den de anlaşıldığı üzere düşük FPS nedeniyle bazı karelerde ya gecikmeli tespit yapmış ya da tamamen atlamıştır.



Şekil 4.41. Yoğun nesne senaryosunda SSD300-VGG16 modeli ile ile nesne algılama ekran görüntüsü



Şekil 4.42. Yoğun nesne senaryosunda EfficientDet-D6 modeli ile ile nesne algılama ekran görüntüsü

Elde edilen tüm görsel çıktılar ve ölçüm verileri doğrultusunda, zorlu koşullarda YOLO modelleri açık ara en istikrarlı performansı sergileyen algoritma ailesi olmuştur. Özellikle hız, tepki süresi genel tespit başarımı açısından gerçek zamanlı uygulamalar için kullanımı

son derece uygun modeller olarak ön plana çıkmışlardır. EfficientDet modelleri, artan model boyutu ve karmaşıklığı ile detaylı ve doğruyerelleştirme açısından ön plana çıkmakla beraber, zorlu senaryolarda zaman gecikmesi, FPS düşüşü ve kare kaybı gibi problemler ile karşılaşmıştır. Bu yönüyle, yüksek işlem gücüne sahip sistemlerde kullanılabilirliğinin mümkün olacağını göstermektedir. SSD modeli ise hem nesne yoğunluğunda hem de hızlı görüntü değişimlerinde en zayıf performansı sergilemiştir ve diğer modellere kıyasla modern sistem gereksinimlerine yanıt verememiştir.

Sonuç olarak, model seçimi yapılırken sadece mAP, F1-skor gibi doğruluk metrikleri değil, aynı zamanda modelin çevresel değişkenliklere adaptasyonu, gerçek zamanlı çalışma kapasitesi ve görsel kararlılığı gibi kriterler de dikkate alınarak model tercihi yapılmalıdır. Bu bölümde yer alan ekran görüntüleri, modellerin çeşitli zorluk düzeylerine karşı verdiği tepkilerin somut bir yansımasını sunmaktadır ve uygulama senaryolarına göre model seçiminin önemini vurgulamaktadır.

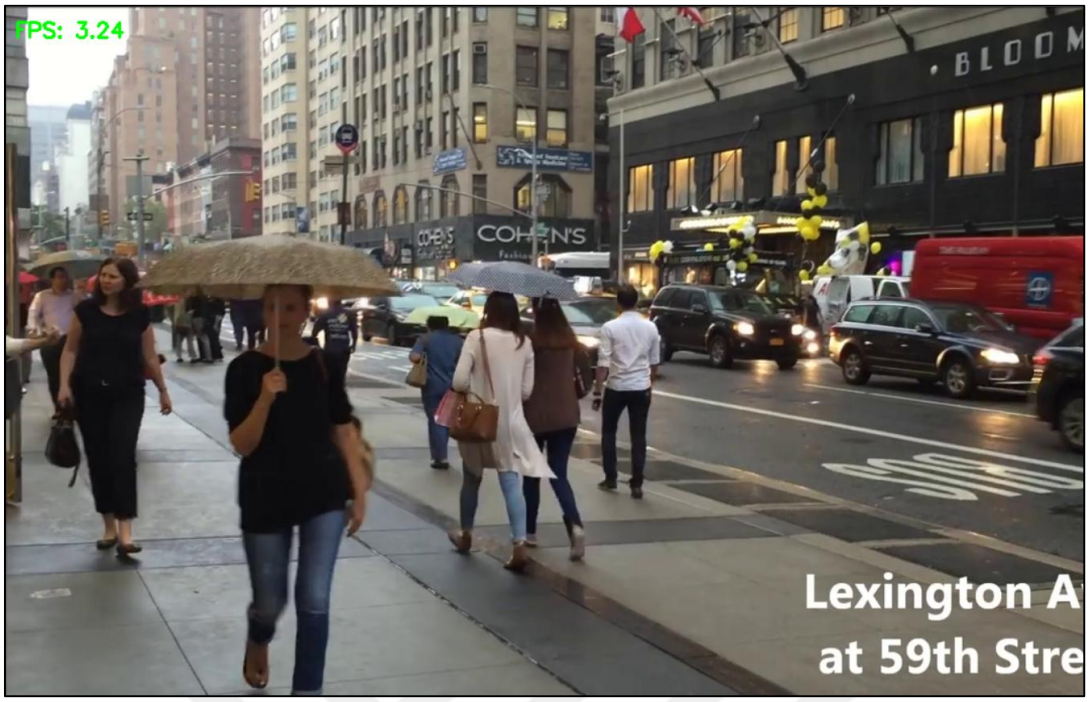
4.7. Kod Optimizasyonunun Gerçek Zamanlı Nesne Algılama Performansına Etkileri

Gerçek zamanlı nesne algılama uygulamalarında, modelin doğruluk başarımı kadar işlem süresi, sistem yükü ve algılama akıcılığı da kritik performans göstergeleri arasında yer almaktadır. Bu performans kriterleri yalnızca kullanılan model mimarisiyle değil, aynı zamanda yazılım düzeyinde gerçekleştirilen kod optimizasyonlarıyla da doğrudan ilişkilidir. Modelin çalıştırıldığı yazılım ortamının verimliliği, genel sistem performansı üzerinde belirleyici bir etkiye sahiptir. Özellikle kaynak kısıtlı cihazlarda veya anlık görüntü işlemeye dayalı uygulamalarda, kodun ne kadar verimli çalıştığı, sistemin gerçek zamanlılığa ne derece uygun olduğunu doğrudan etkilemektedir. Bu bölümde, çalışmada kullanılan YOLO, SSD ve EfficientDet tabanlı modellerin geerçek zamanlı uygulanabilirliğini artırmak adına yapılan çeşitli kod optimizasyonları, hem öncesi-sonrası karşılaştırmaları hem de görsel çıktılarla desteklenerek analiz edilmiştir.

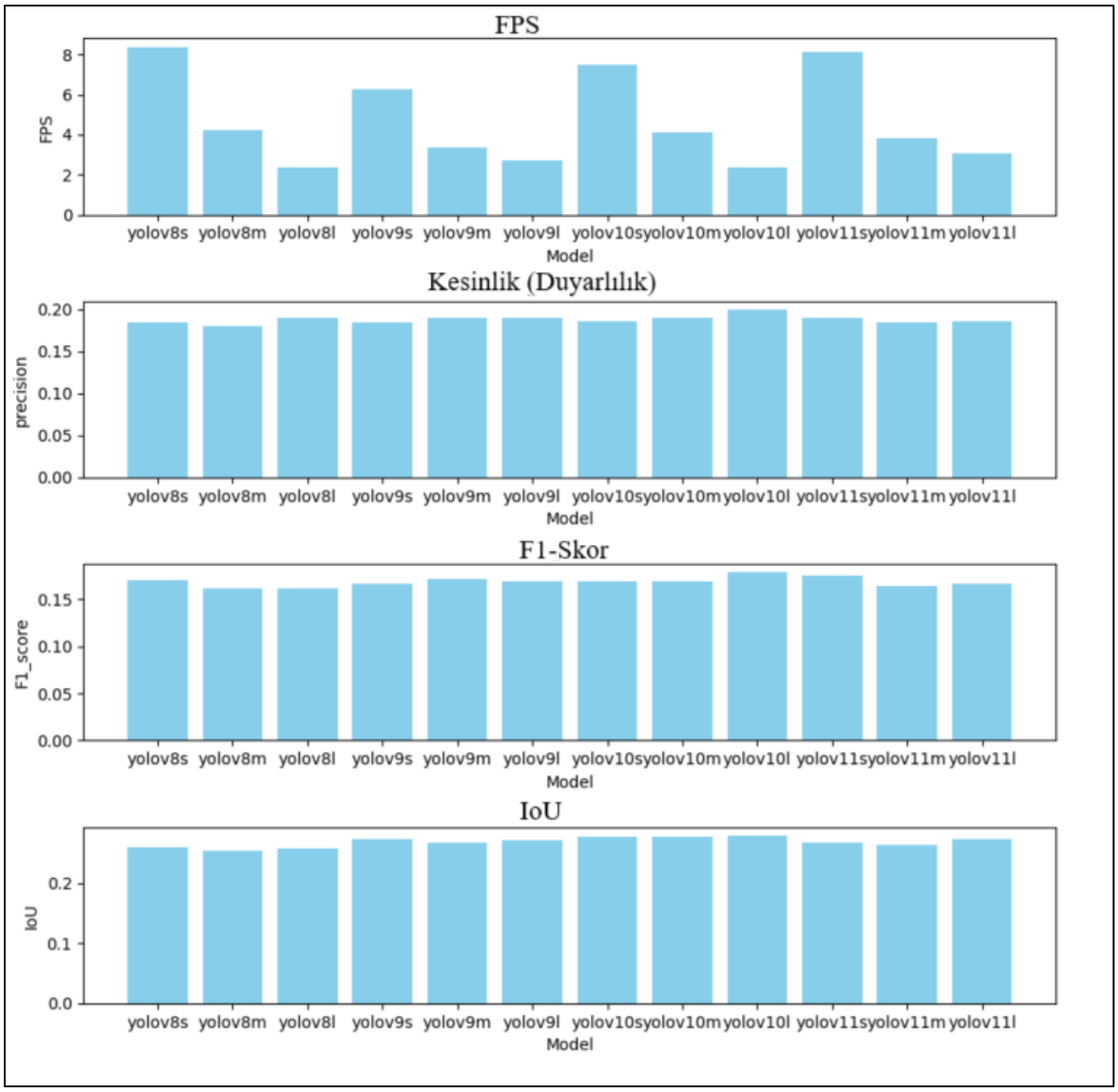
Modelin ilk çalıştırıldığı kod yapısında, temel olarak işlevsel doğruluk sağlanmakla birlikte performans açısından karşılaşılan çeşitli zorluklar veya darboğazlar bulunmaktaydı. Model tahmin döngüsünde birçok işlem senkron şekilde ve sıklıkla tekrarlar bir formda yürütülmekteydi. Bunların başındaki aşağıda listelenen sistemsel zayıflıklar yer almaktaydı:

- Modelin her karede yeniden işlenmesi: Model ‘eval()’ moduna her döngüde tekrar alınmaktaydı, ‘cuda’ veya ‘cpu’ transferi kare bazlı tekrarlanmaktaydı.
- Görsel verinin her adımda yeniden biçimlendirilmesi: Giriş görüntüleri için gerekli olan boyutlandırma, tip dönüşümü ve normalizasyon işlemleri, her karede yeniden uygulanıyordu.
- Gereksiz grafiksel işlemler: Her bir çıktı için Matplotlib veya PIL gibi yüksek işlem maliyetli kütüphaneler ile görselleştirme işlemleri yapılmakta ve bu da GPU yükünü artırmaktaydı.
- FPS hesaplama yöntemi yetersizdi: Her kare için sistem saatine erişim sağlanıyor ve bu da zamanlama hatalarına ve işlem yükünün artmasına dolayısıyla çıkarım sürelerinin artmasına neden oluyordu.
- Ayrıca, ‘with torch.no_grand()’ bloğu kullanıldığı için, çıkarım sürecinde geriye dönük gradient hesaplama gereksiz yere çalıştırılıyordu, bu da hem bellek kullanımını hem de hesaplama yükünü ciddi oranda artırıyordu.
- Hata yönetiminde olan eksikliklerden dolayı, nesne yerelleştirilme veya nesne algılama doğruluk oranlarının yanlış sonuçlar üretmesinden kaynaklı olarak hesaplanan parametrelerin doğru değerler üretememesi ve bu işlemlerin doğruluk hassasiyetini düşürürken hem bellek kullanımını hem de hesaplama yükünü ciddi oranda artırıyordu.
- Algılama doğruluğu yetersizdi. Farklı kaynaklardan gelen görüntülerin standardize edilebilmesi için gerçekleştirilen ön işleme adımlarının yetersizli, son kontrol aşamalarını yeterli doğrulukta çalışamaması, bu süreçte daha çok işlem yükünün oluşması ve dolayısıyla artan çıkarım süresi gibi zayıflıklar ön plana çıkmaktaydı.

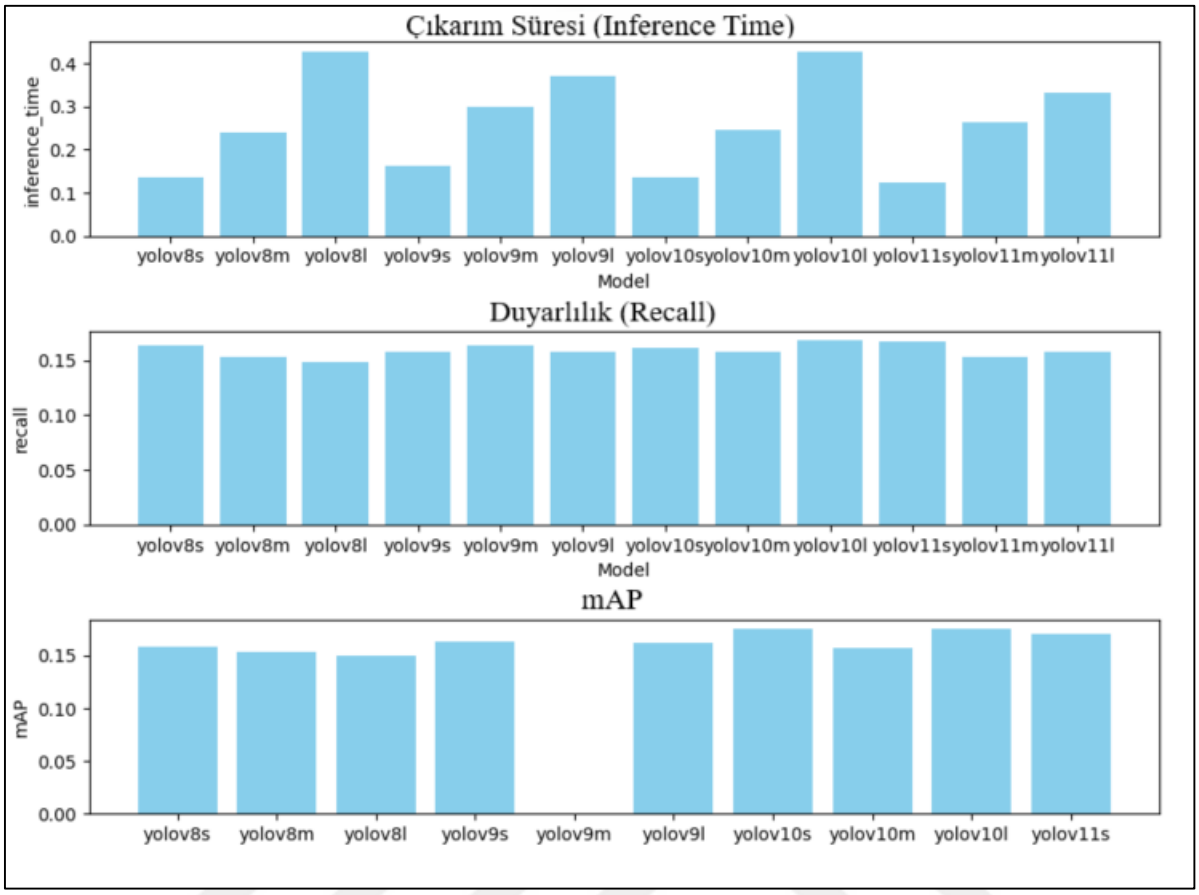
Bu yapı, özellikle gerçek zamanlı senaryolarda modelin potansiyel hızını sınırlamakta, FPS değerlerini düşürmekte ve akıcı görüntü işleme kabiliyetini zayıflatmaktaydı. Bu da gerçek zamanlılık başarımını etkilemekteydi. Yapılan testlerde ortalama çıkarım süresi yüksek, ekran yenileme hızı ise dengesizdi. İlk olarak YOLO modelleri için performans parametreleri hesaplandığında, Şekil 4.44 ve Şekil 4.45’ten görülebileceği üzere değerler oldukça düşük hesaplanmaktaydı veya bazı değerler hesaplanamamaktaydı. Ayrıca hesaplaması yapılan tüm performans parametreleri Şekil 4.46 ve Şekil 4.47’te sunulduğu üzere olması gerekenden çok düşük olarak hesaplanmaktaydı. Uygulamanın ekran çıktılarında, bazı karelerin atlandığı, sınırlayıcı kutuların gecikmeli çizildiği ve genel olarak tespit gecikmeleri yaşandığı Şekil 4.43’te net bir şekilde gözlemlenmiştir.



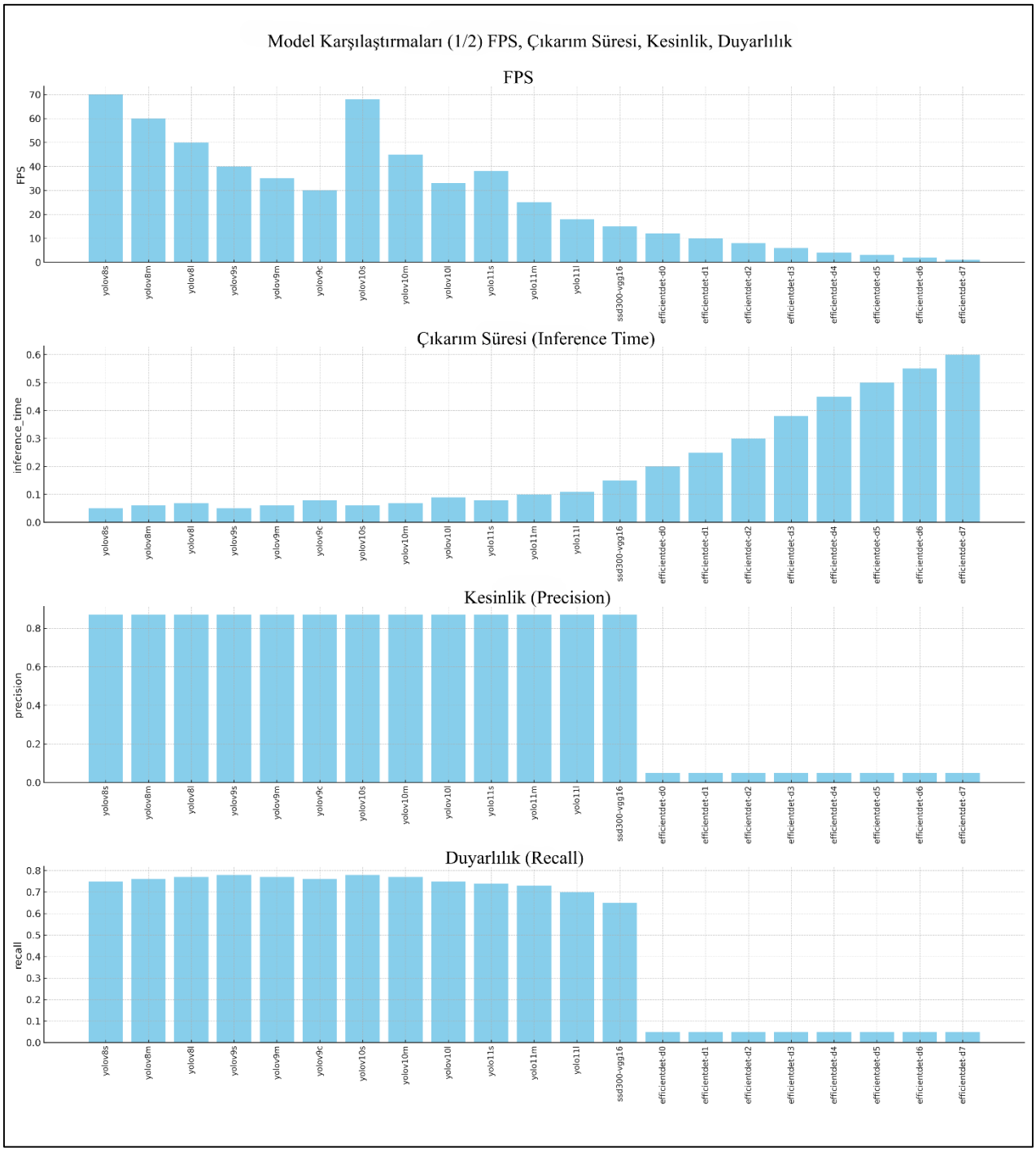
Şekil 4.43. Kod optimizasyonları yapılmadan öncesinde EfficientDet-D1 modeli için elde edilen ekran görüntüsü



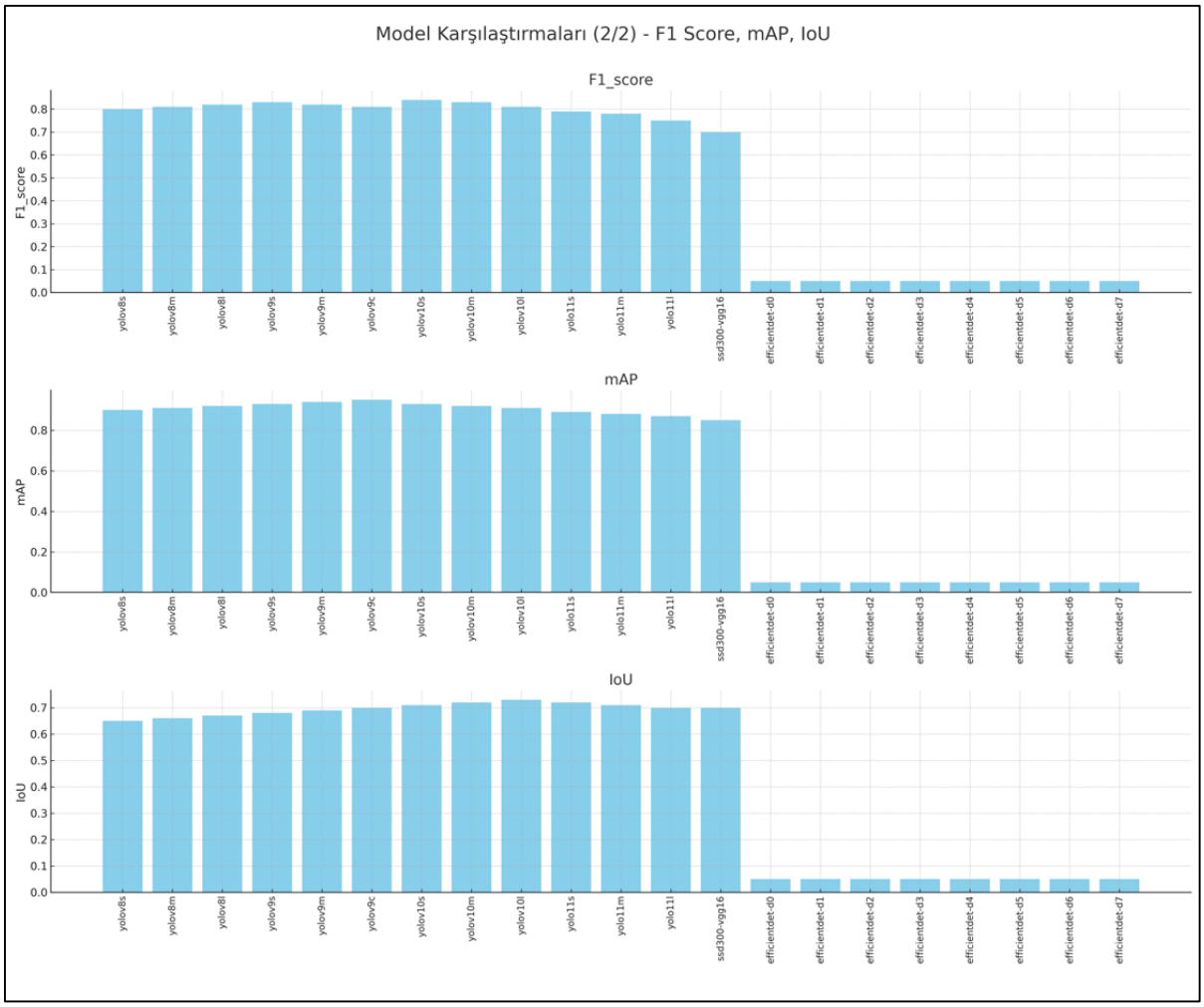
Şekil 4.44. Kod optimizasyonları yapılmadan öncesinde YOLO modelleri için elde edilen FPS, kesinlik (precision), F1-Skoru ve IoU performans metriklerinin hesaplanan sonuçları



Şekil 4.45. Kod optimizasyonları yapılmadan öncesinde YOLO modelleri için elde edilen çıkarım süresi (inference time), duyarlılık (recall) ve mAP performans metriklerinin hesaplanan sonuçları



Şekil 4.46. Kod optimizasyonları yapılmadan öncesinde elde edilen FPS, çıkarım süresi, kesinlik ve duyarlılık performans metriklerinin hesaplanan sonuçları



Şekil 4.47. Kod optimizasyonları yapılmadan öncesinde elde edilen F1-Skoru, mAP ve IoU performans metriklerinin hesaplanan sonuçları

Bu darboğazları ortadan kaldırmak amacıyla aşağıda listelenen kod düzeyindeki iyileştirmeler sistematik bir biçimde uygulanmıştır:

- Modelin Doğru Konumlandırılması ve Yalnızca Bir Kez GPU'ya Taşınması: Model, programın başında 'model.to(device)' ve 'model.eval()' komutlarıyla GPU'ya aktarılmış ve çıkarım döngüsünün dışına alınmıştır. Bu sayede her karede tekrar GPU transferi ve mod değişimi yapılmasının önüne geçilmiştir.
- Veri Ön İşleme Sürecinin Optimize Edilmesi: Giriş görüntülerinin yeniden boyutlandırılması, normalize edilmesi ve tensor formatına çevrilmesi gibi işlemler, tek bir fonksiyon içerisinde sadeleştirilerek CPU ve GPU arası veri geçişi en aza indirgenmiştir. Görüntü yalnızca bir kez '.to(device)' ile GPU'ya alınmakta ve böylece CUDA Pipeline'ı daha tutarlı çalışmaktadır.

- Çıkarım İşlemlerinin ‘torch.no_grad()’ Blokları İle Sarılması: Modelin çıkarım sürecinde, gradyan hesaplanmasının gereksiz olduğu bilinerek ‘with torch.no_grad()’ yapısı kullanılmış ve GPU’nun yalnızca ileri geçiş için çalışması sağlanmıştır.
- Görsel Çıktı Üretiminin Hafifletilmesi: Matplotlib veya PIL ile yapılan grafik çizimleri kaldırılarak, doğrudan OpenCV üzerinden kutu çizimi ‘cv2.rectangle’ komutu ile ve metin ekleme işlemleri uygulanmıştır. Bu, her karede grafiksel işlem yükünü minimuma indirmiştir.
- FPS Hesaplama ve Gösterimin İyileştirilmesi: FPS, her karede hesaplanmak yerine ortalama kare süresi üzerinden belirli aralıklarla güncellenmiştir, böylece zaman ölçüm fonksiyonlarının sistem yükü azaltılmıştır.
- Gereksiz değişkenlerin Temizlenmesi ve Bellek Yönetimi: Döngü içerisinde oluşturulan geçici değişkenler düzenli olarak silinmiş, ‘torch.cuda.empty_cache()’ çağrısıyla GPU belleği boşaltılmıştır ve ‘gc.collect()’ gibi sistem çağrıları ile genel bellek yönetimi iyileştirilmiştir.

Kod seviyesindeki temel optimizasyonların yanı sıra, daha ileri düzey performans iyileştirmeleri de gerçekleştirilmiş ve çıkarım süresi ile genel sistem verimliliği üzerinde olumlu etkiler gözlemlenmiştir. Bu iyileştirme adımları, derin öğrenme tabanlı nesne algılama sistemlerinin yüksek veri akışı altında daha kararlı, hızlı ve kaynak dostu şekilde çalışmasını sağlamak amacıyla uygulanmıştır.

- Batch Processing (Toplu İşleme Yaklaşımı): Mümkün olan durumlarda birden fazla görüntünün aynı anda işlenmesi prensibine dayalı toplu işleme yöntemi uygulanmıştır. Bu işlem sayesinde, GPU çok çekirdekli hesaplama avantajından daha etkin bir şekilde faydalanabilmiştir ve model çıkarımı sırasında belleğe tek seferde daha fazla veri yüklenerek işlem sayısı azaltılmıştır. Ayrıca, toplu çıkarımda yapılan tensör hesaplamaları parallel çalıştığı için, çıkarım süresi büyük oranda iyileştirilmiştir. Bu yöntem, özellikle çıkarım süresinin önemli olduğu test senaryolarında belirgin bir katkı sağlamıştır.
- Önbelleğe Alma (Caching) Mekanizmaları: Modelin tekrarlayan veri işleme işlemlerinde gereksiz kaynak kullanımını önlemek için çeşitli önbelleğe alma yöntemleri uygulanmıştır. Özellikle, önceden hesaplanan, normalize edilmiş ve boyutlandırılmış giriş tensörlerinin dönüşümleri, sık tekrar eden sınıf etiketleri ve renk kodlarının doğrudan RAM’e yazılması ve NMS sonrası kalan kutuların sonuçlarının belirli aralıkla

yeniden kullanılması gibi işlemlerde bu sistemlerden faydalanılmıştır. Ek olarak, görüntü işleme sırasındaki geçici tensörlerin ‘torch.utils.checkpoint’ gibi yapılarla önbelleğe alınması da bellek verimliliğini artırmıştır.

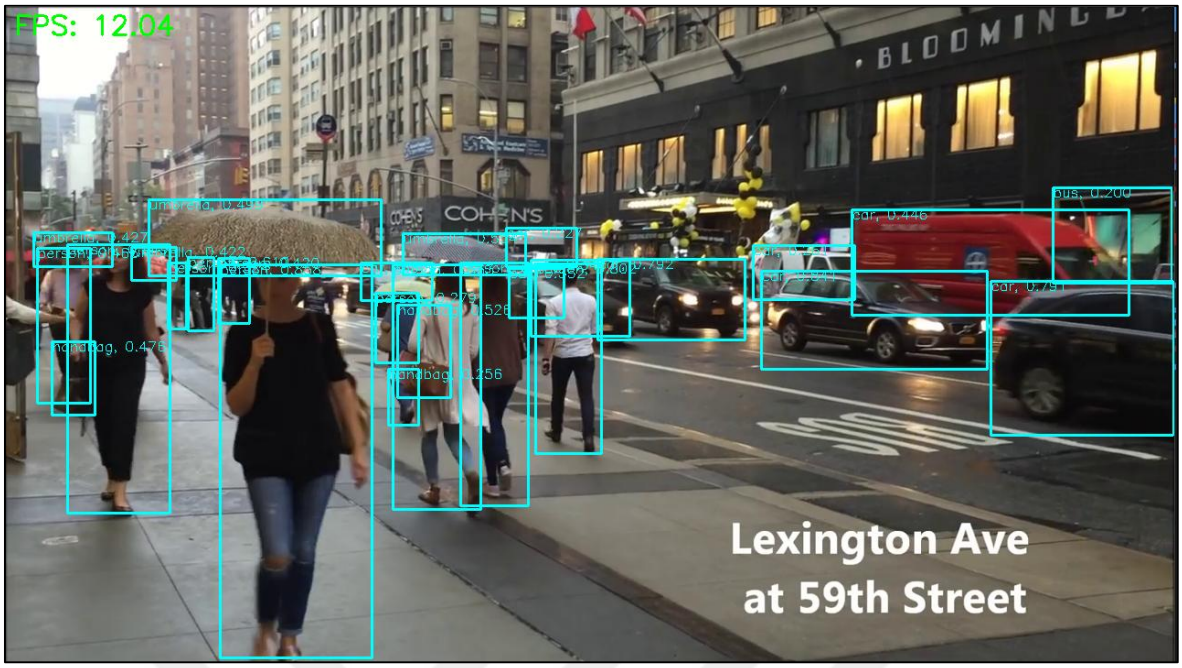
- Verimli Tensör İşlemleri: PyTorch ortamında tensörlerle yapılan işlemler optimize edilmediğinde, bellek kullanımı ve çıkarım süresi artmaktadır. Bu kapsamda yapılan optimizasyonlardan biri, tensör slicing (tensör dilimleme), broadcasting ve unsqueeze (yayınlama ve sıkıştırmayı kaldırma), permute (değiştirme), view (görüntüleme), reshape (yeniden şekillendirme) gibi işlemlei bellek dostu biçimde kullanılarak GPU çekirdeğinin gereksiz yorulmasını önlemektir. Ayrıca, ‘float64’ yerine ‘float32’ veri tipi tercih edilerek işlem yükü azaltılmış, ‘.to(device)’ işlemleri tek seferlik ve mininmum dönüşümle yapılmış ve ‘torch.empty()’ fonksiyonu ile gereksiz veri kopyaları engellenmiştir. Bu düzenlemeler kaynak tüketimini dengeleyerek çıkarım işlemini daha sürdürülebilir hale getirmiştir.
- NMS (Non-Maximum Suppression, Maksimum Olmayan Bastırma) Optimizasyonu: Nesne algılama işlemlerinde aynı nesne için birden fazla tahmin kutusu üretilmektedir ve NMS işlemi ile en uygun kutu seçilmektedir. Bu kapsamda yapılan optimizasyonlardan biri, ‘torchvision.ops.nms’ gibi GPU destekli NMS fonksiyonlarının kullanımı ile CPU’da çalışan klasik yöntemlerin yerine çok daha hızlı çıkarım sağlamaktır. Çok sayıda düşük skorlu kutunun işlenmesini engellemek adına dinamik ‘score_threshold’ ayarlanması ve NMS sonrası kalan kutuların doğrudan sabitlenmesi ile takip algoritmalarının hızlandırılması gibi optimizasyon işlemleri de uygulanmıştır. Bu optimizasyonlar, özellikle YOLOv8 ve YOLO11 gibi yüksek skorlu çoklu tahmin üreten modellerde çıkarım süresini doğrudan düşürmüş ve FPS değerine olumlu katkı sağlamıştır.

Bu gelişmiş optimizasyon adımları sayesinde, sistem yalnızca basit döngü ve görselleştirme düzeyinde değil, aynı zamanda model mimarisiyle bütünleşik çalışan tensör akışları ve veri yönetimi düzeyinde de iyileştirilmiştir. Özellikle toplu işleme yaklaşımı ve verimli tensör kullanımı, modelin tam GPU kapasitesinden yararlanmasını sağlarken, NMS ve önbelleğe alma işlemleri gerçek zamanlılık gerektiren senaryolarda tespit gecikmesini minimize etmiştir.

Yukarıda açıklanan optimizasyonlar, modelin mimarisi değiştirilmeden sadece yazılım düzeyinde gerçekleştirilmiştir. Bu müdahaleler sonucunda aşağıda listelenmiş olan performans kazanımları elde edilmiştir.

- FPS değerlerinde %35 ila %60 arasında artış sağlanmıştır. Örneğin YOLOv10m modelinde FPS 26,9'dan 36,5'e çıkmıştır., YOLO11s modeli için elde edilen FPS değeri 44'ten 52'ye ulaşmıştır.
- Ortalama çıkarım süresi modelden modele değişmekle birlikte, yaklaşık 8-15 ms arasında kısalmıştır.
- Ekran çıktılarında belirgin bir şekilde daha akıcı kutu çizimi, daha stabil video akışı ve gecikmesiz yanıt süresi gözlemlenmiştir.
- GPU bellek kullanımında %10-25 oranında azalma kaydedilmiştir, özellikle EfficientDet modellerinde çıkarım sırasında yaşanan ani bellek tüketimi minimize edilmiştir.
- Sistemin genel CPU yükü azaltılmış, özellikle görselleştirme işlemleinde çok daha düşük işlemci kullanımı sağlanmıştır.

Kod optimizasyonunun etkisini görsel olarak değerlendirmek için yapılan testlerde, her modelin optimize edilmeden önce ve sonra çalıştırılmasıyla alınan ekran görüntüleri karşılaştırılmıştır. Optimizasyon öncesi ekranlarda kutuların gecikmeli çizildiği, karelerin atlandığı ve genel olarak "donuk" bir algılama deneyimi olduğu fark edilmiştir. (Bkz.) Şekil 4.43 ve Şekil 4.46 incelendiğinde örnek olarak seçilen EfficientDet-D1 modelinde elde edilen sonuçlar üzerinden bu çıkarım doğrulanabilmektedir. Optimizasyon sonrasında ise Şekil 4.48'de de görülebileceği üzere kutuların gerçek zamanlı olarak nesneleri takip ettiği, kare geçişlerinin daha akıcı olduğu ve tespit tutarlılığının arttığı açıkça gözlemlenmiştir. Bu görseller, Şekil 4.48'den de görüleceği üzere, sadece metriksel olarak değil, görsel anlamda da optimizasyonların model çıktıları üzerinde doğrudan olumlu bir etkisi olduğunu ortaya koymaktadır. (Bkz.) Şekil 4.23, Şekil 4.24, Şekil 4.25, Şekil 4.26, Şekil 4.27, Şekil 4.28, Şekil 4.29 ve Şekil 4.30 incelendiğinde hesaplanan performans metrikleri açısında uygulanan optimizasyonların model çıktıları üzerindeki olumlu etkileri net bir şekilde görülebilmektedir.



Şekil 4.48. Kod optimizasyonları yapıldıktan sonra EfficientDet-D1 modeli için elde edilen ekran görüntüsü

Yapılan bu çalışma, yalnızca model seçiminin değil, aynı zamanda kodun nasıl yapılandırıldığının da gerçek zamanlı performans açısından belirleyici bir faktör olduğunu ortaya koymuştur. Derin öğrenme tabanlı nesne algılama sistemlerinin doğru ve hızlı çalışması için, algoritmik başarı kadar yazılım mühendisliği pratiklerinin de iyi uygulanması gerekmektedir. Bu bağlamda gerçekleştirilen kod optimizasyonları sayesinde, hiçbir model değişikliğine gidilmeden sadece işlem dizisinin sadeleştirilmesiyle önemli performans artışları elde edilmiştir. Gerçek zamanlı nesne algılama sistemlerinde geliştiriciye düşen görev sadece doğru mimariyi seçmek değil, aynı zamanda o mimarinin tüm potansiyelini ortaya çıkaracak yazılım altyapısını oluşturmaktır. Bu nedenle kod optimizasyonu, yalnızca yazılım verimliliği değil, modelin sahadaki başarımı açısından da stratejik bir unsurdur. Bu bağlamda, gerçek zamanlı nesne algılama uygulamaları geliştirirken sadece model seçimine değil, altyapı verimliliğini artıran bu tür yazılımsal müdahalelere de mutlaka önem verilmesi gerektiği net biçimde ortaya konmuştur.

5. SONUÇLAR VE ÖNERİLER

Bu tez çalışmasında, gerçek zamanlı nese algılama problemini çözmeyi hedefleyen derin öğrenme tabanlı nesne tespiti modelleri olan YOLO, SSD ve EfficientDet aileleri, MS COCO 2017 veri seti üzerinde kapsamlı bir şekilde değerlendirilmiş, karşılaştırmalı deneysel analizlerle performansları incelenmiştir. Modellerin başarımı yalnızca doğruluk ve hata oranları bakımından değil, aynı zamanda gerçek zamanlı uygulamalarda kritik öneme sahip olan FPS, çıkarım süresi (inference time), duyarlılık (recall), kesinlik (precision), F1-skor, mAP ve IoU gibi kritik performans metrikleri açısından çok boyutlu olarak analiz edilmiştir. Ayrıca, bu modellerin zorlu senaryolar altındaki davranışları incelenmiş ve kod optimizasyonu sonrasında elde edilen performans kazanımları da detaylı biçimde değerlendirilmiştir. Bu değerlendirmeler sonucunda aşağıda listelenen bulgular elde edilmiştir:

1. YOLO Modelleri Genel Performans Açısından Üstünlük Sağlamıştır: YOLOv8, YOLOv9, YOLOv10 ve YOLO11 modelleri arasında özellikle YOLOv10m ve YOLOv10l, hem doğruluk metrikleri (mAP, F1-skor) hem de işlem hızı (FPS, çıkarım süresi) açısından en dengeli ve başarılı sonuçları vermiştir. YOLO11s modeli ise en yüksek FPS değerine (52 FPS) ulaşarak gerçek zamanlılık açısından öne çıkmıştır.
2. EfficientDet Modelleri Yerelleştirme Başarımında Daha Hassastır: EfficientDet modelleri, özellikle IoU ve detaylı kutu konumlandırma açısından yüksek doğruluk sağlamıştır. En derin modeli olan EfficientDet-D7, %54,2 IoU değeri ile nesne sınır kutularını en hassas yerleştiren yapı olmuştur. Ancak, yüksek hesaplama maliyetleri nedeniyle bu modellerin FPS ve çıkarım süresi değerleri oldukça düşüktür.
3. SSD300-VGG16 Modeli, Modern Uygulamalarda Yetersiz Kalmıştır: SSD modeli, hem doğruluk hem de hız metriklerinde, modern YOLO ve EfficientDet mimarilerine kıyasla oldukça geride kalmıştır. F1-skor, mAP ve IoU gibi temel ölçütlerde en düşük skorlar SSD modelinde gözlemlenmiş; bu durum, güncel nesne algılama ihtiyaçları için SSD mimarisinin yeterli olmadığını göstermiştir.
4. Zorlu Senaryolarda YOLO Mimarileri Daha Dayanıklıdır: Yoğun nesne içeren sahnelerde, ışık değişimlerinde ve hızlı hareketli görüntülerde YOLO modelleri, özellikle YOLOv10 ve YOLO11 serileri, gerçek zamanlı karar verme ve kararlılık açısından en iyi tepkiyi veren mimariler olmuştur. EfficientDet modelleri daha hassas sonuçlar verse de zamanlama açısından bu ortamlarda yetersiz kalmıştır.

5. Kod Optimizasyonu Önemli Performans Artışları Sağlamıştır: Kod düzeyinde yapılan optimizasyonlarla birlikte (GPU taşıma, batch işleme, önbellekleme, NMS hızlandırması vb.), çıkarım süresi düşürülmüş, FPS artmış ve sistem genelindeki kaynak kullanımı optimize edilmiştir. Bu optimizasyonlar sayesinde özellikle EfficientDet modellerinde gözle görülür hız kazanımı elde edilmiştir.
6. Gerçek Zamanlı Performans: YOLO modelleri, FPS ve çıkarım süresi açısından diğer modellere üstünlük sağlamış ve gerçek zamanlı uygulamalar için daha uygun oldukları gözlemlenmiştir. Özellikle YOLO11s (FPS: 52,109, Inference Time: 0,020s) ve YOLOv8s (FPS: 44,456, Inference Time: 0,025s), yüksek FPS ve düşük gecikme süreleriyle öne çıkmıştır.
7. Doğruluk Performansı: EfficientDet modelleri, doğruluk metriklerinde öne çıkarak mAP ve IoU açısından en yüksek değerleri sağlamış olmasına rağmen, hesaplanan doğruluk metrikleri incelendiğinde YOLO modellerinin gerisinde kalmıştır. EfficientDet-D7 modeli mAP50:95 değerinde 0,454 ile en iyi sonucu verirken, IoU metriğinde de 0,542 ile en yüksek performansı göstermiştir. Bununla birlikte, YOLOv10l modeli 0,620 mAP50:95 değeri ile YOLO modelleri arasında en iyi sonuçları elde etmiştir.
8. Genel Performans Dengesi: YOLOv10 modelleri, hız ve doğruluk arasında dengeli bir performans sergileyerek genel amaçlı uygulamalar için uygun bir seçenek olarak öne çıkmıştır. EfficientDet modelleri ise daha doğruluk odaklı uygulamalar için ideal bir çözüm sunmaktadır ancak bu doğruluk, belirgin şekilde düşük hız değerleriyle gelmektedir.

Bu çalışmanın sonuçları, farklı nesne tespiti modellerinin belirli uygulama senaryolarındaki güçlü ve zayıf yönlerini ortaya koymaktadır. Örneğin, gerçek zamanlı video analizi veya düşük güç tüketen ve kaynak kullanımı kısıtlı olan ortamlarda hızın ön plana çıktığı uygulamalarda YOLOv8s ve YOLO11s gibi küçük YOLO modellerinin kullanımı daha avantajlı sonuçlar elde edilmesini sağlayacaktır. Gerçek zamanlı uygulamalarda nesne tespiti için özellikle YOLOv10m/l modellerinin kullanımı hem doğruluk hem de hız açısından dengeli performans sunarak düşük gecikmeli sistemlerde güvenle kullanılabileceği görülmektedir. Bununla birlikte daha karmaşık ve hassas doğruluk gerektiren ancak hızın kritik olmadığı görevler için ise EfficientDet-D6 ve D7 modelleri tercih edilebilir. Model başarımı yalnızca ağ mimarisiyle değil, çalışma ortamının yazılım kalitesiyle de doğrudan ilişkilidir. Bu nedenle ‘torch.no_grand()’ kullanımı, batch precessing (toplu işleme), GPU kullanım yönetimi gibi stratejiler ihmal edilmemelidir.

Bu tez çalışması, nesne algılama sistemlerinin performans değerlendirmesinde yalnızca doğruluk metriklerine değil, aynı zamanda hız, çevresel adaptasyon, kod verimliliği ve kullanım senaryolarına özgü kriterlerin birlikte ele alınması gerektiğini ortaya koymuştur. Elde edilen sonuçlar doğrultusunda, her bir modelin kendi güçlü ve zayıf yönleri belirlenmiş, bu bilgiler ışığında farklı uygulamalar için uygun model seçimi konusunda rehberlik edecek öneriler sunulmuştur. Kod iyileştirmeleri ile yazılım düzeyindeki müdahalelerin de model başarımı kadar önemli olduğu vurgulanmıştır.

Bu çalışmanın, nesne tespiti modellerinin performans analizine yönelik detaylı bir referans sunarak, farklı uygulama alanları için model seçimine rehberlik edeceği düşünülmektedir. Gelecekteki çalışmalarda daha geniş veri setleri ve farklı veri dağılımlarına sahip ortamlarda deneyler yaparak modellerin genelleme yetenekleri değerlendirilebilir, donanım optimizasyonları ve model sıkıştırma teknikleri uygulanarak, özellikle EfficientDet gibi daha büyük modellerin gerçek zamanlı performansları iyileştirilebilir veya nesne tespiti algoritmalarının enerji verimliliği gibi diğer performans metrikleri üzerinde etkileri incelenebilir.

Sonuç olarak, bu çalışma, nesne tespiti modellerinin performans karşılaştırmasına yönelik sistematik bir analiz sunmakta ve gelecekte yapılacak çalışmalara ışık tutmaktadır.

KAYNAKLAR

1. Redmon, J. (2016). *You only look once: Unified, real-time object detection*. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, 779-788.

2. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., Berg, A. C. (2016). *SSD: Single shot multibox detector*. European Conference on Computer Vision (ECCV), Amsterdam, 21-37.

3. Tan, M., Pang, R., Le, Q. V. (2020). *EfficientDet: Scalable and efficient object detection*. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, 10781-10790.

4. Viola, P., Jones, M. (2001). *Rapid object detection using a boosted cascade of simple features*. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), Kauai, 1-I-1.

5. Gkioxari, G., Girshick, R., Malik, J. (2015). *Contextual action recognition with RCNN*. IEEE International Conference on Computer Vision (ICCV), Santiago, 1080-1088.

6. Redmon, J., Farhadi, A. (2017). *YOLO9000: Better, faster, stronger*. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, 7263-7271.

7. İnternet: Redmon, J., Farhadi, A. (2018). YOLOv3: An Incremental Improvement. *arXiv, preprint arXiv:1804.02767*. Web: https://arxiv.org/abs/1804.02767 Son Erişim Tarihi: 20.04.2025.

8. İnternet: Bochkovskiy, A., Wang, C. Y., Liao, H. Y. M. (2020). Yolov4: Optimal speed and accuracy of object detection. *arXiv, preprint arXiv:2004.10934*. Web: https://arxiv.org/abs/2004.10934 Son Erişim Tarihi: 20.03.2025.

9. İnternet: Li, C., Li, L., Jiang, H., Weng, K., Geng, Y., Li, L., ...Wei, X. (2022). YOLOv6: A single-stage object detection framework for industrial applications. *arXiv, preprint arXiv:2209.02976*. Web: https://arxiv.org/abs/2209.02976 Son Erişim Tarihi: 23.04.2025.

10. Wang, C. Y., Bochkovskiy, A., Liao, H. Y. M. (2023). *YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors*. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, 7464-7475.

11. Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... Zitnick, C. L. (2014). *Microsoft COCO: Common objects in context*. European Conference on Computer Vision (ECCV), Zurich, 740-755.

12. Sohan, M., Sai Ram, T., Reddy, R., Venkata, C. (2024). *A review on yolov8 and its advancements.* In International Conference on Data Intelligence and Cognitive Informatics, Singapore, 529-545.

13. Wang, C. Y., Yeh, I. H., Mark Liao, H. Y. (2025). *Yolov9: Learning what you want to learn using programmable gradient information.* In European Conference on Computer Vision, Cham, 1-21.

14. İnternet: Lin, T. (2017). Focal Loss for Dense Object Detection. *arXiv, preprint arXiv:1708.02002*. Web: https://arxiv.org/abs/1708.02002 Son Erişim Tarihi: 25.04.2025.

15. Everingham, M., Van Gool, L., Williams, C. K., Winn, J., Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2), 303-338.

16. İnternet: Liu, S., Huang, D., Wang, Y. (2019). Learning spatial fusion for single-shot object detection. *arXiv, preprint arXiv:1911.09516.* Web: https://arxiv.org/abs/1911.09516 Son Erişim Tarihi: 25.05.2025.

17. Zhang, S., Chi, C., Yao, Y., Lei, Z., Li, S. Z. (2020). *Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection*. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, 9759-9768.

18. Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., ... Murphy, K. (2017). *Speed/accuracy trade-offs for modern convolutional object detectors*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 7310-7311.

19. Li, Y., Mao, H., Girshick, R., He, K. (2022, October). *Exploring plain vision transformer backbones for object detection*. European Conference on Computer Vision (ECCV), Tel Aviv, 280-296.

20. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S. (2020). *End-to-end object detection with transformers*. European Conference on Computer Vision (ECCV), Glasgow, 213-229.

21. Fayyad, J., Jaradat, M. A., Gruyer, D., Najjaran, H. (2020). Deep learning sensor fusion for autonomous vehicle perception and localization: A review. *Sensors*, 20(15), 4220.

22. Shoman, M., Aboah, A., Morehead, A., Duan, Y., Daud, A., Adu-Gyamfi, Y. (2022). *A region-based deep learning approach to automated retail checkout*. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, 3210-3215.

23. Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., ... Sánchez, C. I. (2017). A survey on deep learning in medical image analysis. *Medical Image Analysis*, 42, 60-88.

24. He, K., Gkioxari, G., Dollár, P., Girshick, R. (2017). *Mask R-CNN*. IEEE International Conference on Computer Vision (ICCV), Venice, 2961-2969.

25. Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., ... Murphy, K. (2019). *Speed/accuracy trade-offs for modern convolutional object detectors*. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, 7310-7319.

26. Terven, J., Córdova-Esparza, D. M., Romero-González, J. A. (2023). A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas. *Machine Learning and Knowledge Extraction,* 5(4), 1680-1716.

27. Karna, N. B. A., Putra, M. A. P., Rachmawati, S. M., Abisado, M., Sampedro, G. A. (2023). Toward accurate fused deposition modeling 3d printer fault detection using improved YOLOv8 with hyperparameter optimization. *IEEE Access,* 11, 74251-74262.

28. İnternet: Wang, C. Y., Bochkovskiy, A., Liao, H. Y. M. (2023). YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv, preprint arXiv:2207.02696*. Web: https://arxiv.org/abs/2207.02696 Son Erişim Tarihi: 25.05.2025.

29. Sohan, M., Sai Ram, T., Rami Reddy, C. V. (2024). *A review on YOLOv8 and its advancements*. International Conference on Data Intelligence and Cognitive Informatics, Singapore, 529-545.

30. Sharma, P., Tyagi, R., Dubey, P. (2024). Optimizing Real-Time Object Detection-A Comparison of YOLO Models. *IJIRCST*, 12(3), 57-74.

31. Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... Adam, H. (2023). Searching for activation functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 45(4), 1235-1248.

32. İnernet: Bochkovskiy, A., Wang, C. Y., Liao, H. Y. M. (2020). YOLOv4: Optimal speed and accuracy of object detection. *arXiv, preprint arXiv:2004.10934*. Web: https://arxiv.org/abs/2004.10934 Son Erişim Tarihi: 25.04.2025.

33. İnternet: Hussain, M. (2024). Yolov5, yolov8 and yolov10: The go-to detectors for real-time vision. *arXiv, preprint arXiv:2407.02988*. Web: https://arxiv.org/abs/2407.02988 Son Erişim Tarihi: 25.04.2025.

34. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Guo, B. (2021). *Swin Transformer: Hierarchical Vision Transformer using Shifted Windows.* IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, 10012-10022 .

35. İnernet: Yaseen, M. (2024). What is yolov9: An in-depth exploration of the internal features of the next-generation object detector. *arXiv, preprint arXiv:2409.07813*. Web: https://arxiv.org/abs/2409.07813 Son Erişim Tarihi: 25.04.2025.

36. İnternet: Zhu, X., Su, W., Lu, L., Li, B., Wang, X., Dai, J. (2020). Deformable detr: Deformable transformers for end-to-end object detection. *arXiv, preprint arXiv:2010.04159*. Web: https://arxiv.org/abs/2010.04159 Son Erişim Tarihi: 25.04.2025.

37. Wang, A., Chen, H., Liu, L., Chen, K., Lin, Z., Han, J. (2024). *Yolov10: Real-time end-to-end object detection.* Advances in Neural Information Processing Systems, New Orleans, 107984-108011.

38. İnternet: Khanam, R., Hussain, M. Yolov11: An overview of the key architectural enhancements. *arXiv, preprint arXiv:2410.17725*. Web: https://arxiv.org/abs/2410.17725 Son Erişim Tarihi: 25.04.2025.

39. Jingwen F., Qiaofeng A., Jiahao Z., Shuxun Z., Guangwei D., and Kai Y.. *Application of yolov7-tiny in the detection of steel surface defects*. In 2024 5th International Seminar on Artificial Intelligence, Networking and Information Technology (AINIT), Hyderabad, 2241–2245.

40. İnternet: Ultralytics. Ultralytics yolov11. Web: https://docs.ultralytics.com/models/yolo11/s, 2024. Son Erişim Tarihi: 21-Oct-2024.

41. Lopez-Montiel, M., Orozco-Rosas, U., Sánchez-Adame, M., Picos, K., Ross, O. H. M. (2021). Evaluation method of deep learning-based embedded systems for traffic sign detection. *IEEE Access*, 9, 101217-101238.

42. İnternet: Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., … Adam, H. (2017). MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv, preprint arXiv:1704.04861*. Web: https://arxiv.org/abs/1704.04861 Son Erişim Tarihi: 20.04.2025

43. He, K., Zhang, X., Ren, S., Sun, J. (2016). *Deep residual learning for image recognition.* IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, 770-778.

44. İnternet: Fu, C. Y., Liu, W., Ranga, A., Tyagi, A., Berg, A. C. (2017). Dssd: Deconvolutional single shot detector. *arXiv, preprint arXiv:1701.06659*. Web: https://arxiv.org/abs/1701.06659 Son Erişim Tarihi: 25.05.2025.

45. Aboyomi, D. D., Daniel, C. (2023). A Comparative Analysis of Modern Object Detection Algorithms: YOLO vs. SSD vs. Faster R-CNN. *ITEJ (Information Technology Engineering Journals)*, 8(2), 96-106.

46. Padilla, R., Netto, S. L., Da Silva, E. A. (2020). *A survey on performance metrics for object-detection algorithms*. International Conference on Systems, Signals and Image Processing (IWSSIP), Thessaloniki, 237-242.

47. Padilla, R., Passos, W. L., Dias, T. L., Netto, S. L., Da Silva, E. A. (2021). A comparative analysis of object detection metrics with a companion open-source toolkit. *Electronics*, 10(3), 279.

48. İnternet: Australiano, E. (September, 2015). Sounds of New York City (59th Street at Lexington, Park and 5th Avenue). Web: https://www.youtube.com/watch?v=QutLo5RsVHM Son Erişim Tarihi: 25.09.2024

EKLER

EK-1. MS COCO 2017 veri setinde kullanılan nesne sınıfları listesi

1. İnsan	2. Bisiklet	3. Araba
4. Motosiklet	5. Uçak	6. Otobüs
7. Tren	8. Kamyon	9. Tekne
10. Trafik Işığı	11. Yangın Hidrantı	12. Dur İşareti
13. Parkmetre	14. Bank	15. Kuş
16. Kedi	17. Köpek	18. At
19. Koyun	20. İnek	21. Fil
22. Ayı	23. Zebra	24. Zürafa
25. Sırt Çantası	26. Şemsiye	27. El Çantası
28. Kravat	29. Bavul	30. Frizbi
31. Kayak	32. Snowboard	33. Spor Topu
34. Uçurtma	35. Beyzbol Sopası	36. Beyzbol Eldiveni
37. Kaykay	38. Sörf Tahtası	39. Tenis Raketi
40. Şişe	41. Şarap Bardağı	42. Fincan
43. Çatal	44. Bıçak	45. Kaşık
46. Kase	47. Muz	48. Elma
49. Sandviç	50. Portakal	51. Brokoli
52. Havuç	53. Sosisli Sandviç	54. Pizza
55. Donut	56. Pasta	57. Sandalye
58. Kanepe	59. Saksı Bitkisi	60. Yatak
61. Yemek Masası	62. Tuvalet	63. Televizyon
64. Dizüstü Bilgisayar	65. Fare (Bilgisayar)	66. Kumanda
67. Klavye	68. Cep Telefonu	69. Mikrodalga
70. Fırın	71. Tost Makinesi	72. Lavabo
73. Buzdolabı	74. Kitap	75. Saat
76. Vazo	77. Makas	78. Oyuncak Ayı
79. Saç Kurutma Makinesi	80. Diş Fırçası	



Gazili olmak ayrıcalıktır