

TÜRKİYE CUMHURİYETİ  
YILDIZ TEKNİK ÜNİVERSİTESİ  
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ



ARAÇLAR İÇİN GÖRÜNTÜLERDEN TRAFİK DURUMU  
TASVİRİ

20011020 – Atakan SUCU  
20011065 – Mert YÜREKLİ

BİLGİSAYAR PROJESİ

Danışman  
Prof. Dr. Mine Elif KARSLIGİL

Ocak, 2024



## **TEŞEKKÜR**

---

İstanbul'da 1911 yılında kurulan Yıldız Teknik Üniversitesi, 10 fakültesi, 2 enstitüsü ve yaklaşık 40.000 öğrencisiyle önde gelen bir devlet kurumudur. Aynı zamanda ülkenin en büyüklerinden biri olarak kabul edilmektedir.

"Araçlar İçin Görüntülerden Trafik Durumu Tasviri" konulu tez çalışmamız boyunca önemli yardımları, sürekli takibi ve yönlendirmeleri için eğitmenimiz Mine Elif Karslıgil'e teşekkür ederiz. Bilgisayar bilimleri, algoritmalar, makine öğrenmesi, yapay zekâ, görüntü işleme, örüntü tanıma, sinir ağları, teknoloji ve mühendislik alanlarındaki bilgi birikimi projenin başarıya ulaşmasında önemli rol oynadı.

Atakan SUCU  
Mert YÜREKLİ

# İÇİNDEKİLER

---

KISALTMA LİSTESİ	v
ŞEKİL LİSTESİ	vi
TABLO LİSTESİ	viii
ÖZET	ix
ABSTRACT	x
<b>1 Giriş</b>	<b>1</b>
<b>2 Ön İnceleme</b>	<b>3</b>
2.1 Literatür Taraması . . . . .	3
<b>3 Fizibilite</b>	<b>5</b>
3.1 Zaman Fizibilitesi . . . . .	5
3.2 Teknik Fizibilite . . . . .	5
3.2.1 Donanım Fizibilitesi . . . . .	5
3.2.2 Yazılım Fizibilitesi . . . . .	6
3.3 Yasal Fizibilite . . . . .	6
3.4 Ekonomik Fizibilite . . . . .	6
<b>4 Sistem Analizi</b>	<b>8</b>
4.1 YOLO . . . . .	9
4.2 Vision Transformer . . . . .	10
4.3 Veri Seti . . . . .	11
<b>5 Sistem Tasarımı</b>	<b>12</b>
5.1 Veri Seti Tasarımı . . . . .	12
5.2 Yazılım Tasarımı . . . . .	12
5.2.1 YOLOv8 . . . . .	13
5.2.2 Vision Transformer . . . . .	15
5.3 Girdi-Çıktı Tasarımı . . . . .	15

<b>6 Uygulama</b>	<b>16</b>
<b>7 Deneysel Sonuçlar</b>	<b>19</b>
7.1 YOLOv8 . . . . .	19
7.2 DETR . . . . .	25
<b>8 Performans Analizi</b>	<b>29</b>
8.1 YOLOv8 . . . . .	29
8.2 ViT - DETR . . . . .	30
<b>9 Sonuç</b>	<b>31</b>
<b>Referanslar</b>	<b>32</b>
<b>Özgeçmiş</b>	<b>33</b>

## KISALTMA LİSTESİ

---

CNN	Convolutional Neural Network
ViT	Vision Transformer
YOLO	You Only Look Once
R-CNN	Region Based Convolutional Neural Network
VGG16	Visual Geometry Group
NMS	Network Management System
mAP	Mean Average Precision
AP	Average Precision
GPU	Graphics Processing Unit
RAM	Random Access Memory
SSD	Solid State Drive
GB	Gigabyte
IDE	Integrated Development Environment
F1	F Score
IoU	Intersection Over Union
NMS	Non-Maximum Suppression
DETR	Detection Transformer
FFN	Feed Forward Network
CE	Cardinality Error
GIoU	Generalized Intersection over Union

## ŞEKİL LİSTESİ

---

Şekil 3.1 Gantt Diyagramı . . . . .	5
Şekil 4.1 YOLO Versiyonları Karşılaştırması . . . . .	9
Şekil 4.2 YOLOv8 Modellerinin Karşılaştırılması . . . . .	9
Şekil 4.3 ViT Çalışma Mekanizması . . . . .	10
Şekil 4.4 VIT-CNN Modellerinin Karşılaştırılması . . . . .	11
Şekil 5.1 YOLOv8 Model Yapısı . . . . .	14
Şekil 5.2 DETR Model Yapısı . . . . .	15
Şekil 6.1 Modelin Eğitilmesi İçin Yazılan Kod Parçası . . . . .	16
Şekil 6.2 Obje Tespiti Yapılması İçin Yazılan Kod Parçası . . . . .	16
Şekil 6.3 Yolov8 Obje Tespiti 1 . . . . .	17
Şekil 6.4 Yolov8 Obje Tespiti 2 . . . . .	17
Şekil 6.5 DETR Modelinin Eğitilmesi İçin Yazılan Kod Bloğu . . . . .	18
Şekil 6.6 DETR Obje Tespiti . . . . .	18
Şekil 7.1 Results Curve 50 Epoch . . . . .	20
Şekil 7.2 Results Curve 100 Epoch . . . . .	20
Şekil 7.3 Precision-Confidence Curve 50 Epoch . . . . .	21
Şekil 7.4 Precision-Confidence Curve 100 Epoch . . . . .	21
Şekil 7.5 Recall-Confidence Curve 50 Epoch . . . . .	22
Şekil 7.6 Recall-Confidence Curve 100 Epoch . . . . .	22
Şekil 7.7 Precision-Recall Curve 50 Epoch . . . . .	23
Şekil 7.8 Precision-Recall Curve 100 Epoch . . . . .	23
Şekil 7.9 F1 Score 50 Epoch . . . . .	24
Şekil 7.10 F1 Score 100 Epoch . . . . .	24
Şekil 7.11 Confusion Matrix 50 Epoch . . . . .	25
Şekil 7.12 Confusion Matrix 100 Epoch . . . . .	25
Şekil 7.13 Validation Loss . . . . .	27
Şekil 7.14 Validation Bounding Box Loss . . . . .	27
Şekil 7.15 Validation Loss CE . . . . .	27
Şekil 7.16 Validation Loss GIOU . . . . .	27
Şekil 7.17 Validation Cardinality Error . . . . .	27
Şekil 7.18 Training Loss . . . . .	28

Şekil 7.19 Training Bounding Box Loss . . . . .	28
Şekil 7.20 Training Loss CE . . . . .	28
Şekil 7.21 Training Loss GIOU . . . . .	28
Şekil 7.22 Training Cardinality . . . . .	28
Şekil 8.1 YOLOv8 50 Epoch mAP Değeri . . . . .	29
Şekil 8.2 YOLOv8 100 Epoch mAP Değeri . . . . .	29
Şekil 8.3 YOLOv8 50 Epoch AP Değerleri . . . . .	29
Şekil 8.4 YOLOv8 100 Epoch AP Değerleri . . . . .	30
Şekil 8.5 ViT Tabanlı DETR Modelinin AP Değerleri . . . . .	30

## **TABLO LİSTESİ**

---

Tablo 3.1 Proje İçin Gereken Minimum Donanım Elemanları . . . . . 7

## ÖZET

---

# ARAÇLAR İÇİN GÖRÜNTÜLERDEN TRAFİK DURUMU TASVİRİ

Atakan SUCU

Mert YÜREKLİ

Bilgisayar Mühendisliği Bölümü

Bilgisayar Projesi

Danışman: Prof. Dr. Mine Elif KARSLIGİL

Arabalar ortaya çıktığından beri olası kazaları en aza indirmek için çeşitli trafik kuralları belirlenmiştir. Gelişen teknoloji ile birlikte bu kurallara uymakta yardımcı olan yarı otonom araçlar ve kurallara sürücüsüz uyabilen otonom araçlar geliştirmek mümkün olmuştur. Bu durum trafik durumunu anlayabilen makinelerle sağlanmıştır.

Trafik durumu tasviri, trafikte bulunan levhaların ve trafik ışıklarının tespit edilerek ne anlama geldiğinin belirlenmesi için görüntü işleme metodlarıyla yapılır. Bu metodlar trafiğe çıkan tüm otonom araçlarda kullanılabilir.

Trafik durumu tasviri için araç içlerinden çekilen fotoğraflar ile veri setleri oluşturulmuştur. Bu veri setleri ile birçok model eğitilmiştir. Bu modeller eğitilirken CNN mimarisini yaygın olarak kullanılmıştır. Bu projede görüntü işleme alanında yeni bir yaklaşım olan ViT mimarisinin trafik durumu tasvirinde nasıl performans gösterdiği ölçülmüştür. ViT mimarisini test etmek için Meta tarafından geliştirilmiş DETR modeli seçilmiştir. Ayrıca Ocak 2023'te çıkış yapmış olan ve ViT mimarisile geliştirilen modellerden daha iyi performans gösterdiğine dair örnekleri olan YOLOv8 modelinin de performansı ölçülmüştür. Yolov8 ile DETR metodlarının performansları karşılaştırılmıştır. YOLOv8 modeli %93.6 doğruluk oranında sonuç verirken DETR modeli %89.4 doğruluk oranına ulaşmıştır.

**Anahtar Kelimeler:** Görüntü işleme, Trafik, Otonom araçlar, Trafik Tasviri, YOLOv8, DETR, Vision Transformer.

## **ABSTRACT**

---

# **TRAFFIC SITUATION DEPICTION FROM IN-CAR IMAGES**

Atakan SUCU

Mert YÜREKLİ

Department of Computer Engineering

Computer Project

Advisor: Prof. Dr. Mine Elif KARSLIGİL

Since the advent of cars, various traffic rules have been established to minimize potential accidents. With advances in technology, it has been possible to develop semi-autonomous vehicles that help to follow these rules and autonomous vehicles that can follow the rules without a driver. This has been achieved with machines that can understand the traffic situation.

Traffic situation description is done by using image processing methods to identify traffic signs and traffic lights and determine what they mean. These methods can be used in all autonomous vehicles on the road.

Data sets were created with photographs taken from inside the vehicles for traffic description. Many models were trained with these data sets. While training these models, CNN architecture was widely used. In this project, the performance of the ViT architecture, which is a new approach in the field of image processing, is measured. The DETR model developed by Meta was chosen to test the ViT architecture. We also measured the performance of the Yolov8 model, which was released in January 2023 and has examples of outperforming models developed with the ViT architecture. The performances of YOLOv8 and DETR methods were compared. The YOLOv8 model achieved an accuracy of 93.6% while the DETR model achieved an accuracy of 89.4%.

**Keywords:** Image processing, Traffic, Autonomous cars, Traffic description, YOLOv8, DETR, Vision Transformer.

# 1

## Giriş

---

Objenin tespiti günümüzde birçok alanda kullanılan, kamera aracılığı ile yakalanan görselleri tanıma ve anlamlandırmaya yarayan bir yöntemdir. Bu alanlardan en güncel ve dikkate değer konulardan biri trafikte obje tespitidir. Özellikle otonom araçların popülerleşmesi ile otomatik sürüş sistemlerinde, araçlar için görüntülerle trafik durumu analizi, görüntü işleme ve nesne tanıma yöntemleri bu konudaki çalışma alanlarındanandır. Geliştirilmiş algoritmalar ve yapay zekâ teknikleri, araç içi kameralardan elde edilen verileri kullanarak çevresel unsurları tanımlama ve anlama konusunda aktif olarak kullanılmıştır. Aynı zamanda görüntü işleme algoritmaları, trafikteki diğer araçlarla etkileşimleri anlamak ve potansiyel tehlikeleri önceden belirlemek için de kullanılır.

Geçtiğimiz birkaç yılda, Mercedes, Tesla, Audi gibi dünyaca ünlü birçok araba markası otonom sürüş üzerine odaklanmıştır. Otonom sürüş için ise en önemli unsur trafikteki objeleri tanımaktır. Objenin tespiti birçok farklı sensörün yardımıyla yapılabilse de bu alanda en önemli yöntemlerden birisi kuşkusuz kameralar ile görüntü tespitidir. Kameralar ile yapılan obje tespitinde kamera açısından olan araçlar, yayalar, trafik levhaları, trafik işaretleri gibi birçok unsur tespit edilebilir ve sınıflandırılabilir. Bu tespit ve sınıflandırma işlemlerinin yanında aynı zamanda şerit ve yol takibi de yapılabilir. Objenin tespitinin yapılmasında yapay zekâ, makine öğrenmesi ve derin öğrenme aktif rol oynamaktadır. Bu alandaki en yaygın yöntemlerden birisi CNN'dir. CNN konvolüsyonel katmanları kullanarak modelin öğrenmesini optimize eden bir derin öğrenme modelidir. Son yıllarda gündeme gelen bir başka yöntem ise ViT'dir. Geçmiş yıllarda yapılan görüntü işleme modellerinden farklı olarak, her bir görseli parçalara bölgerek ve bu parçaları transformer aracılığıyla işleyerek sınıflandırma için gerekli bilgiyi oluşturur. Bu yaklaşım, görüntü verilerini transformer mimarisile işleyerek geniş kapsamlı ve etkili bir performans sağlar.

Araçlar için görüntülerden trafik durumu tasviri bazı durumlarda zorlayıcı olabilir. Karmaşık trafik yapıları, birçok farklı boyutta, şekilde olan cisimler başlıca zorluklardandır. Karanlık, yağmurlu, sisli havalar gibi olumsuz hava koşulları veya

yola bağlı olan toz, duman gibi durumlarda hayatı öneme sahip olan tespitler yapılamayabilir. Ayrıca ışığın yola ve levhalara yansması ya da güneşli havalarda kamerada oluşabilecek yansımalar sebebiyle görüntü işleme daha da zor hâle gelebilir.

Sonuç olarak, trafikte obje tespiti çoğu zorluğa rağmen hayatı bir öneme sahiptir. Projenin motivasyonu, günümüzde yaygınlaşan otonom araçların trafikte güvenli bir şekilde gidebilmesi için trafikteki objeleri yüksek bir doğruluk orANIyla tanıyalıbilmesidir. Bu projenin amacı ise, farklı obje tespiti metodlarının trafik durumu tasviri üzerindeki performansını ölçmek ve kıyaslamaktır. Proje kapsamında CNN tabanlı olarak geliştirilmiş EfficientNet mimarisini kullanan YOLOv8 modeli ve son yıllarda gelişim gösteren vision transformer mimarisini kullanan DETR yöntemi kullanılarak iki farklı model geliştirilecektir. Ayrıca bu iki modelin kıyaslaması yapılacaktır.

## 2 Ön İnceleme

---

Trafikte obje tespiti, özellikle otonom araçların güvenliğinde ve sürüş deneyiminde kullanılmak üzere, 2000'li yılların başlarında önem kazanmaya başlamıştır. Bu dönemde, bilgisayarlı görüş, derin öğrenme ve nesne tespiti teknolojilerindeki gelişmeler, trafikteki nesnelerin otomatik olarak algılanması ve takip edilmesi konusunda yeni olanaklar sağlamıştır. Otonom araç teknolojilerinin yükselişiyle birlikte bu alandaki çalışmalar daha da hız kazanmıştır. Bu alanda yapılan bazı çalışmalar aşağıda verilmiştir.

### 2.1 Literatür Taraması

- [1] Bu araştırma 12 Şubat 2021'de yayınlanmıştır. Çalışmada karmaşık trafik koşullarında, küçük ve örtülü objelerin daha doğru ve hızlı tespiti amaçlanmıştır. Bu yaklaşımada R-CNN tabanlı çapraz katman birleşimli çoklu obje tanıma yöntemi uygulanmıştır. Beş katmanlı VGG16 derin öğrenme modeli daha karakteristik bilgi edinmek için kullanılmıştır. Bu fikri entegre etmek için  $1 \times 1$  konvolüsyonal kernel, maksimum pooling ve dekonvolüsyon yöntemleri uygulanmıştır. Ayrıca tespit edilmesi zor ve kolay örnekler arasındaki kontrolü sağlamak için Soft-NMS kullanılmıştır. Veri seti olarak KITTI ve Cityspaces datasetleri kullanılmıştır. Araştıraya göre detection mAP %86,2, proposal recall ise %98.6 olarak hesaplanmıştır
- [2] Bu araştırma 13 Eylül 2023 yılında yayınlanmıştır. Bu çalışmada 15 farklı sınıf içeren toplam 4650 görselden oluşan özgün bir veri seti oluşturulmuştur. Veri setleri aynı hiperparametre değerleri ile YOLOv5, YOLOv7, YOLOv8 modelleri kullanılarak eğitilmiştir. YOLOv8 mimarisinin YOLOv5 ve YOLOv7 mimarilerine göre sırasıyla %14, %13'lük mAP50-95 iyileşmesi sağlanmıştır.
- [3] Bu araştırma 1 Kasım 2023 yılında yayınlanmıştır. Çalışmada transformer modelinin etkinliğini artırmak için araştırma, bir locality inductive bias ve bir transformer modülünü entegre eden yeni bir strateji önerilmektedir. Bu, efektif

konvolüsyon bloğu ve lokal transformer bloğunun tanıtılmasını içerir. Bu da kısa vadeli ve uzun vadeli bağımlılık bilgilerini etkili bir şekilde yakalar ve hem tespit hızını hem de doğruluğunu artırır. Deneysel değerlendirmeler, özellikle GTSDB veri kümese uygulandığında bu yaklaşımın elde ettiği önemli ilerlemeleri göstermektedir. Bu yöntem sonucunda AP Cascade CNN'e göre %2.1, Faster R-CNN'e göre %2.4 artış göstermiştir.

- [4] Bu araştırma 19 Eylül 2023 yılında yayınlanmıştır. Araştırmanın amacı trafikte gündüz ve gece koşullarındaki YOLOv8 versiyonlarının (YOLOv8n, YOLOv8s, YOLOv8m, YOLOv8l, YOLOv8x) doğruluk oranlarını kıyaslamaktır. Araştırma sonuçlarına göre yolov8x modelinin diğer modellere göre precision, recall, f-measure ve accuracy değerleri daha yüksektir. Doğruluk oranları sırasıyla yolov8n 0.76, YOLOv8s 0.78, YOLOv8m 0.82, YOLOv8l 0.85, YOLOv8x 0.89 olarak hesaplanmıştır.
- [5] Bu araştırma 19 Aralık 2021'de yayınlanmıştır. Bu çalışmada CNN modeli ile ViT modelleri arasında bir karşılaştırma yapılmıştır. CNN modelinin veri setinin büyüklüğünden etkilenmeden kısa sürede öğrenme gerçekleştirmesine karşın ViT modelinin öğrenme süresi büyük veri setlerinde oldukça yavaştır. 20000 görsel içeren bir veri setinde CNN modelinin eğitilmesi 6 dakika sürerken ViT için 170 dakika sürmüştür. CNN modelinin doğruluğu %75.71 iken ViT için %75.46'dır. Bu çalışmada ViT'nin doğruluk oranının veri setinin büyüklüğüne yüksek oranda bağlı olduğu gözlemlenmiştir.
- [6] Bu araştırma 22 Temmuz 2022'de yayınlanmıştır. Dataset olarak GTSDB kullanılmıştır. Piramit transformer modeli, atrous konvolüsyon yöntemlerini kullanarak piramit küçültme katmanları ile veriyi zengin tokenlar içerisinde küçülterek gömer. Bu yöntem, %77.8 mAP değerini elde etmiştir. Bu model trafik veri setlerinin küçük olması ve sınıflarının dengesiz dağılmasından dolayı önerilmiştir.
- [7] Bu araştırma 28 Mayıs 2020'de yayınlanmıştır. Facebook tarafından yeni tanıtılan DETR modeli anlatılmaktadır. Veri seti olarak COCO object detection dataset kullanılmıştır. CNN ile görselin özellikleri çıkarıldıktan sonra vision transformer mimarisi ile obje tespiti yapılabileceğini gösteren ilk çalışmalarlardandır. Bu çalışmada doğruluk oranı ile ilgili bir bilgi verilmemiştir.

# 3 Fizibilite

Bu bölümde projenin kapsamını belirlemek için yapılan fizibilite çalışmaları açıklanmıştır.

## 3.1 Zaman Fizibilitesi

Projenin tamamlanması için 3 aylık bir sürenin uygun olduğu belirlendi. Zaman fizibilite yönünü oluşturmak için waterfall metodu 3.1 kullanıldı.



Şekil 3.1 Gantt Diyagramı

## 3.2 Teknik Fizibilite

Sistemin teknik yapısı ve olanakları yazılım ve donanım fizibilitesi alt başlıklarında incelenmiştir.

### 3.2.1 Donanım Fizibilitesi

Modelin eğitimi ilk aşamada Google Colab üzerinde yapıldı ve bir taslak oluşturuldu. Ancak Colab'ın ücretsiz deneme sürümündeki GPU ve zaman sınırlamaları nedeniyle

Colab Pro satın alındı. Ayrıca model lokal bilgisayarda da eğitildi. Modeli oluşturmak için ihtiyaç duyulan bilgisayarın özellikleri şu şekildedir:

- Hard Disk: Samsung 512GB SSD
- İşlemci: Intel i7-1065G7 4-Core
- Grafik Kartı: NVIDIA MX250
- RAM: 8GB

Model eğitiminin elimizdeki donanımlar ile çok uzun sürmesi sebebiyle geliştirme ve eğitim süreci Google Colaboratory üzerinden V100 ve A100 GPU modelleri kullanılarak gerçekleştirilmiştir. Yukarıda belirtilen spesifikasyonlar, modeli eğitmek ve çalıştırılmak için kullanılan yeterli donanımlardır.

### 3.2.2 Yazılım Fizibilitesi

Modelin geliştirilmesi için python programlama dili kullanılmıştır. Geliştirilirken TensorFlow, Matplotlib, scikit-image, Numpy, OpenCV, Torch, supervision, transformers, pytorchlightning ve ultralytics gibi kütüphaneler kullanılmıştır. Geliştirme ortamı olarak model eğitiminde Google Colab ve IDE olarak PyCharm tercih edilmiştir. Python 3.9 versiyonu ve uyumlu kütüphane versiyonları seçilmiştir.

## 3.3 Yasal Fizibilite

Bu projede faydalanan veri setleri açık kaynaklı olduğu için etik kurulun onayına ihtiyaç duyulmamaktadır. Yazılım lisansı bakımından da açık kaynaklı yazılımlar kullanılmış ve geliştirilen yazılım şahsimiza aittir. Bu sebeple herhangi bir lisans hakkı ihlal edilmemiştir.

## 3.4 Ekonomik Fizibilite

Projede iki farklı bilgisayar kullanılmıştır, bilgisayarların toplam değeri 80.000 TL'dir. Bu bilgisayarlardan Apple Macbook Pro 2019 60.000 TL, HP Pavilion 20.000 TL'dir. Bir çalışanın aylık 40.000 TL aldığı varsayılarak iki çalışan haftanın dört günü iş saatinin çeyreği kadar çalışırsa alacakları maaş  $2 \times 40.000 \times 0.25 \times 0.8 = 16.000$  TL'dir. Proje 3 ay devam edeceği için toplam maaş ödemesi  $3 \times 16.000 = 48.000$  TL olacaktır. Kullanılacak donanımla beraber projenin maliyeti  $48.000 + 80.000 = 128.000$  TL'dir.

**Tablo 3.1** Proje İçin Gereken Minimum Donanım Elemanları

Eleman	Tutar(TL)
Samsung 512GB SSD	2.000
Intel i7-1065G7 4-Core	8.000
NVIDIA MX250	5.000
8 GB RAM	1.500

## 4 Sistem Analizi

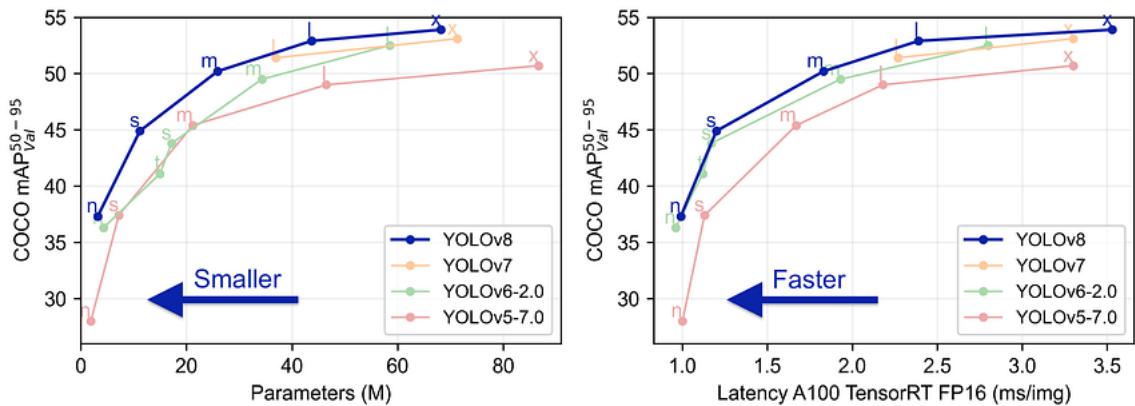
---

Bu projenin amacı trafik içerisinde bir sürücünün gördüğü levhaları ve trafik ışıklarını tespit edebilmek ve anlamlandırmaktır. Bu sayede otonom veya yarı otonom araçlar sürücü ve yolcuların trafikteki güvenliğini sağlayabilecektir. Bunun yanında güvenlik için birincil öncelik olan trafik kurallarına uymada sürücülere destek olacaktır. Bu sistemin gerçek zamanlı olarak güvenilir şekilde çalışması için aşağıda verilen metrikler üzerinden değerlendirme yapılabilir:

- **Intersection over Union (IoU):** IoU, tahmin edilen objenin bounding box'ı ile dataset içerisinde verilen bounding box arasındaki örtüşmeyi temsil eder. Obje lokalizasyonunda önemli bir ölçütür.
- **Average Precision (AP):** AP, precision-recall grafiğinde sınırın altında kalan alanı temsil eder. Tek bir sınıfın hangi oranda doğru tespit edildiğini gösterir.
- **Mean Average Precision (mAP):** mAP değeri, hesaplanan AP değerlerinin ortalaması alınarak bulunur. Modelin performansını belirlemek için birden fazla sınıf içeren veri setlerinde obje tespiti ile ilgili kapsayıcı bilgi sunar.
- **Precision and Recall:** Precision, tüm pozitif tahminler arasında doğru pozitiflerin oranını ölçerek modelin yanlış pozitiflerden kaçınma kabiliyetini değerlendirir. Öte yandan, recall tüm gerçek pozitifler arasındaki gerçek pozitiflerin oranını hesaplayarak modelin bir sınıfın tüm örneklerini tespit etme yeteneğini ölçer.
- **F1 Score:** F1 skoru, precision ve recall değerlerinin harmonik ortalamasıdır ve hem yanlış pozitifleri hem de yanlış negatifleri dikkate alırken bir modelin performansının dengeli bir değerlendirmesini sağlar. 0 ile 1 arasında olan bu değer 1'e yaklaşıkça modelin doğruluğu artar.

## 4.1 YOLO

Yolo Joseph Redman tarafından Washington Üniversitesi'nde 2015 yılında yayınlanmıştır. İlk versiyon olan YOLOv1, COCO veri seti ile eğitilerek %63.4 doğruluk oranı elde etmiştir. Konvolüsyonel sinir ağları (CNN) temelini kullanan model, nesne tespiti için gerekli olan özelliklerin çıkarımını, sınıflandırmasını ve lokalizasyonunu tek bir geçişte gerçekleştirebilir. Diğer modellerden farklı olarak, bir görseli  $M \times M$ 'lik bloklara bölerek her bir blokta aynı anda sınıflandırma yapar. Bu sebepten diğer görüntü işleme yöntemlerine göre çok daha hızlı tespit yapabilen bir görüntü işleme algoritmasıdır. Trafikte gerçek zamanlı olarak yüksek FPS'te obje tespiti yapılması gereği için bu projede tercih edilmiştir. Projede, yapılan araştırmalar sonucunda 4.1 Yolo'nun en güncel versiyonu olan Yolov8 kullanılması uygun görülmüştür. Yolov8 versiyonu önceki versiyonları da kapsarken, bunun üzerine esneklik ve verimlilik için yeni özellikler sunmaktadır.



Şekil 4.1 YOLO Versiyonları Karşılaştırması

Aşağıdaki tabloda ise 4.2 aynı veri seti üzerinde Yolov8 modellerinin karşılaştırılması yapılmıştır.

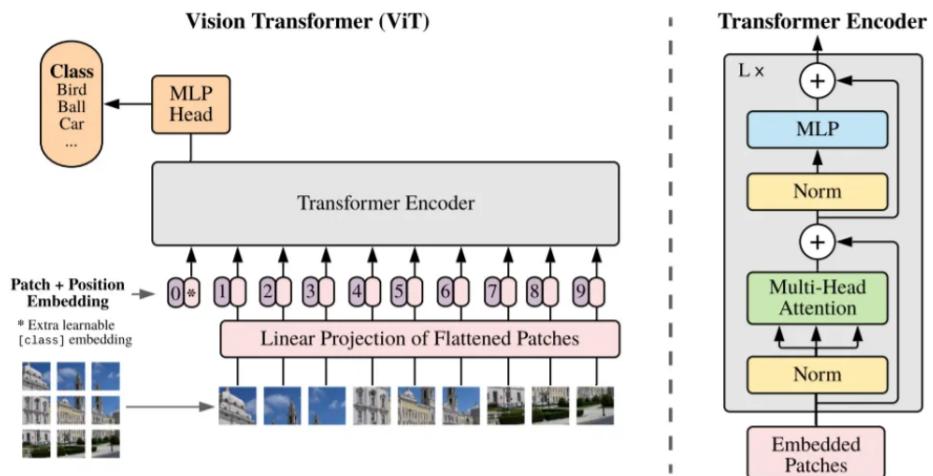
Model	size (pixels)	mAP <sub>val</sub> 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

Şekil 4.2 YOLOv8 Modellerinin Karşılaştırılması

## 4.2 Vision Transformer

Son yıllarda CNN tabanlı yöntemlere alternatif olarak ortaya transformerlar çıkmıştır. Başlangıçta doğal dil işleme modelleri için oluşturulmuş olan transformerlar bir modelin anlama, yorumlama ve üretme özelliklerini dikkat mekanizması kullanarak geliştirmede kullanılır. Dikkat mekanizması modelin girdideki spesifik yerlere odaklanması sağlayıcı bir yöntemdir. Daha sonrasında görüntü işleme alanında uygulamaları başlayan transformerlar vision transformer olarak adlandırılmıştır. Yapılan çalışmalarda ViT'in CNN modellerine göre daha iyi sonuçlar verebildiği gözlemlenmiştir.

ViT mimarisi birçok aşamadan oluşmaktadır. Alınan görsel parçalara bölünür ve her parçanın işlem parçasını belirlemek için bir numara atanır. İki boyutlu dizi halinde saklanan görsel parçaları düzleştirilerek tek boyutlu dizi haline getirilir. Daha sonra linear projeksiyon yöntemi ile her bir dizi küçültülmektedir. Linear projeksiyon yöntemi iki adımdan oluşur. İlk adım weight matrix multiplication, ikinci adım ise bias addition'dır. Bu yöntem sayesinde daha az hafıza gereklidir ve çalışma hızı artar. Daha sonrasında position embedding yöntemi ile her bir görsel parçasına ana görseldeki konumu eklenir. Transformer encoder içerisinde gömülü dikkat mekanizması görsel parçaları arasındaki ilişki ve bağımlılıkları inceleyerek modeli eğitir.



**Şekil 4.3** ViT Çalışma Mekanizması

Transformerların popülerleşmesiyle ViT mimarisine ek olarak CNN ve transformerların birleştirilmesiyle birçok farklı metod oluşturulmuştur. Sadece transformer ile oluşturulan mimarilere kıyasla CNN ile kullanılan mimarilerin daha iyi performans verdiği tablo 4.4'de gözlemlenebilir.

Method	Epochs	AP	AP <sub>S0</sub>	AP <sub>75</sub>	AP <sub>S</sub>	AP <sub>M</sub>	AP <sub>L</sub>	#Params (M)	GFLOPs	FPS
<i>CNN based</i>										
FCOS [127]	36	41.0	59.8	44.1	26.2	44.6	52.2	-	177	23 <sup>†</sup>
Faster R-CNN + FPN [13]	109	42.0	62.1	45.5	26.6	45.4	53.4	42	180	26
<i>CNN Backbone + Transformer Head</i>										
DETR [16]	500	42.0	62.4	44.2	20.5	45.8	61.1	41	86	28
DETR-DC5 [16]	500	43.3	63.1	45.9	22.5	47.3	61.1	41	187	12
Deformable DETR [17]	50	46.2	65.2	50.0	28.8	49.2	61.7	40	173	19
TSP-FCOS [122]	36	43.1	62.3	47.0	26.6	46.8	55.9	-	189	20 <sup>†</sup>
TSP-RCNN [122]	96	45.0	64.5	49.6	29.7	47.7	58.0	-	188	15 <sup>†</sup>
ACT+MKKD (L=32) [123]	-	43.1	-	-	61.4	47.1	22.2	-	169	14 <sup>†</sup>
SMCA [125]	108	45.6	65.5	49.1	25.9	49.3	62.6	-	-	-
Efficient DETR [126]	36	45.1	63.1	49.1	28.3	48.4	59.0	35	210	-
UP-DETR [33]	150	40.5	60.8	42.6	19.0	44.4	60.0	41	-	-
UP-DETR [33]	300	42.8	63.0	45.3	20.8	47.1	61.7	41	-	-
<i>Transformer Backbone + CNN Head</i>										
ViT-B/16-FRCNN <sup>‡</sup> [115]	21	36.6	56.3	39.3	17.4	40.0	55.5	-	-	-
ViT-B/16-FRCNN* [115]	21	37.8	57.4	40.1	17.8	41.4	57.3	-	-	-
PVT-Small+RetinaNet [73]	12	40.4	61.3	43.0	25.0	42.9	55.7	34.2	118	-
Twins-SVT-S+RetinaNet [63]	12	43.0	64.2	46.3	28.0	46.4	57.5	34.3	104	-
Swin-T+RetinaNet [61]	12	41.5	62.1	44.2	25.1	44.9	55.5	38.5	118	-
Swin-T+ATSS [61]	36	47.2	66.5	51.3	-	-	-	36	215	-
<i>Pure Transformer based</i>										
PVT-Small+DETR [73]	50	34.7	55.7	35.4	12.0	36.4	56.7	40	-	-
TNT-S+DETR [29]	50	38.2	58.9	39.4	15.5	41.1	58.8	39	-	-
YOLOS-Ti [128]	300	30.0	-	-	-	-	-	6.5	21	-
YOLOS-S [128]	150	37.6	57.6	39.2	15.9	40.2	57.3	28	179	-
YOLOS-B [128]	150	42.0	62.2	44.5	19.5	45.3	62.1	127	537	-

**Şekil 4.4** VIT-CNN Modellerinin Karşılaştırılması

### 4.3 Veri Seti

Modelin eğitilmesinde kullanılacak bir diğer unsur ise veri setidir. Modelin doğruluk, hız, hassasiyet gibi parametreleri sağlayabilmesi için veri setinin seçilmesinde dikkat edilmesi gerken adımlar vardır. Bunlardan biri trafikte olan sınıf sayısı ihtiyacı sebebiyle sınıf sayısı fazla olan bir veri seti seçilmesidir. İkinci adım, modelin tekrarlı eğitim sonrası ezbere düşerek yanlış tespit etmesini engellemek için büyük bir veri seti seçilmesidir. Bu sayede model çeşitli koşullarda tespit yapabilir hâle getirilir. Üçüncü adım seçilen veri setinde, sınıflardaki görsel sayısının dengeli olmasıdır. Bir sınıftaki görsel sayısının diğerine göre çok olması obje tespiti sırasında çok görülen görselin az görülen ile karıştırılmasına yol açabilir.

# 5

## Sistem Tasarımı

---

### 5.1 Veri Seti Tasarımı

Veri seti seçilirken birçok kaynaka tarama yapılmıştır. Sistem analizinde verilen işlem adımları dikkate alınarak "Road Signs" veri seti seçilmiştir. Bu veri seti içerisinde 2093 görsel bulunmaktadır. Train seti içerisinde 1376, valid seti içerisinde 488, test setinde ise 229 görsel içermektedir. Veri setindeki sınıflarda bulunan görseller dengeli bir şekilde dağıtılmış ve her sınıfta ortalama 100 adet görsel bulunmaktadır. Görseller 640x640 çözünürlükte verilmiştir. Veri seti çeşitli koşullardaki trafik levhaları, trafik ışıkları gibi örnekler içermektedir.

### 5.2 Yazılım Tasarımı

Derin öğrenme, yapay sinir ağlarını eğitmeye odaklanan bir makine öğrenimi konusudur. Bu ağlar, bilgiyi tanımak ve tahminlerde bulunmak için eğitilmiştir. Algoritmalar, büyük veri hacimlerinden desenleri tanımlamak ve ilgili bilgiyi çıkarmak amacıyla geliştirilir. Bunlar, giriş verileri için katmanlı yapıları öğrenmek üzere tasarlanmıştır, bu süreçte nesnel çıktıya yönelik olarak yapay sinir ağı iç parametreleri ayarlanır.

Yapay sinir ağı, birbirine bağlı birçok katmandan oluşan en önemli bileşenidir. Ağdaki her düğüm temel bir hesaplama yapar ve sonuçları ağındaki diğer düğümlere gönderir. Ağı oluşturan düğümler arasındaki bağlantılar ağırlıklar denir ve bu ağırlıklar, eğitim aşamasında giriş verileri ve istenen çıktıya dayalı olarak ayarlanır, böylece ağın performansı optimize edilir.

Proje iki farklı algoritma kullanılarak gerçekleştirilecektir. İlk yöntem, görüntü tespiti için EfficientDet tabanlı YOLOv8 yazılımının uygulaması ile oluşturulmuştur. İkinci yöntem DETR kullanılarak gerçekleştirilecektir.

### 5.2.1 YOLOv8

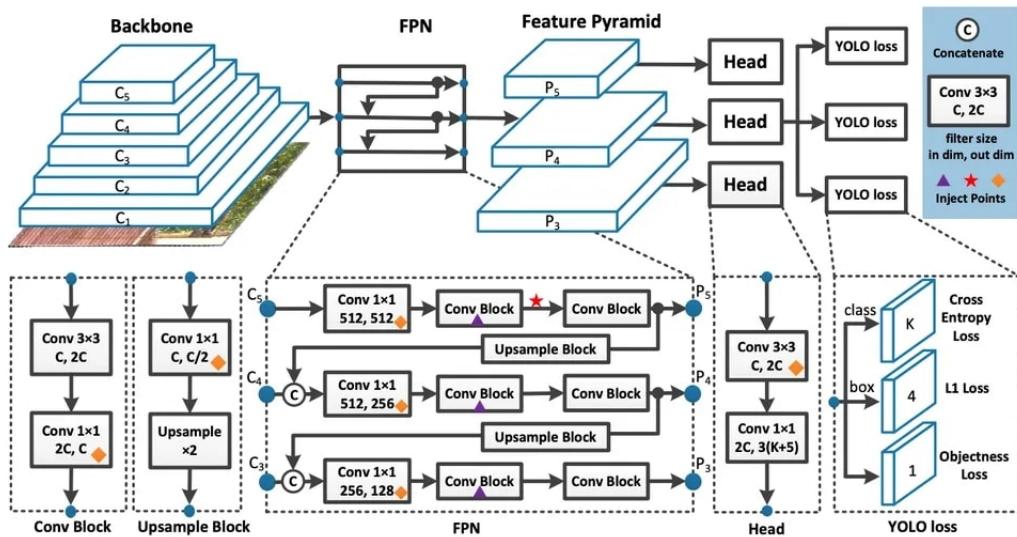
YOLOv8'in temel amacı diğer nesne algılama algoritmalarından farklı olarak görüntünün farklı bölgelerinde birden çok kez nesne algılamayı gerçekleştirmek yerine tek bir görüntüden algılamayı gerçekleştirmektir. Bu, YOLOv8'in nesne algılamada hızlı ve verimli olmasını sağlar.

YOLOv8 hızlı ve yüksek doğruluk sağlayan EfficientDet adlı derin bir sinir ağının mimarisini kullanır. Bu ağ, gerçek dünyadaki nesnelerin modellerini ve özelliklerini öğrenmek için çok sayıda görüntü verisinden yararlanarak eğitilmiştir. Eğitim tamamlandıktan sonra ağ, yeni görüntüler üzerinde nesne algılama yapmak için kullanılabilir. Bu, görüntüdeki nesnelerin varlığını ve konumlarını hızlı ve doğru bir şekilde belirlemek için kullanılabilir.

YOLOv8 mimarisinin üç ana kısımdan oluşmaktadır.

- **Backbone Network:** Bu bileşen YOLOv8'in temelini oluşturur. Giriş görüntüsünden özellikler çıkarmak için ResNet-50 derin konvolüsyonel sinir ağının (CNN) kullanılır. CNN, oluşturduğumuz veri seti gibi büyük veri kümelerinde eğitilir ve görüntülerdeki yüksek düzeyde özelliklerin tanımlanması için kullanılır.
- **Neck Network:** Bu bileşen backbone network head network'üne bağlar. Görevi ise özellik haritasının boyutunu azaltmak ve özelliklerin çözünürlüğünü artırmaktır. Neck network'ü doğruluk ve hız arasında bir denge bulma amacıyla güder, bu da YOLOv8'i hem hızlı hem de verimli kılar.
- **Head Network:** Bu, nesnelerin sınıfını ve konumunu tahmin eden bileşendir. Anchor kutularını kullanarak nesnenin konumunu ve sınıf puanlarını tahmin etmek için kullanılır. Head network, tahmin edilen ve gerçek değer kutuları arasındaki IoU'yu optimize etmek için eğitilir. Ayrıca NMS algoritmasını içerir, bu da örtüsen sınırlama kutularını filtreler ve yalnızca en güvenilir tahminlerin çıktısını sağlar.

Sistem analizi kısmında yapılan karşılaştırmalarda modelin eğitim süresi ve doğruluk oranı baz alındığında en uygun seçenekin YOLOv8s modeli olduğu kanaatine varılmıştır. YOLOv8s modeli, doğruluk oranı, diğer YOLOv8 modellerinden çok farklı olmamakla beraber, eğitim süresi oldukça kısadır. Bu sonucu doğrulamak için YOLOv8'in model eğitilirken ürettiği grafikler göz önüne alınmıştır.

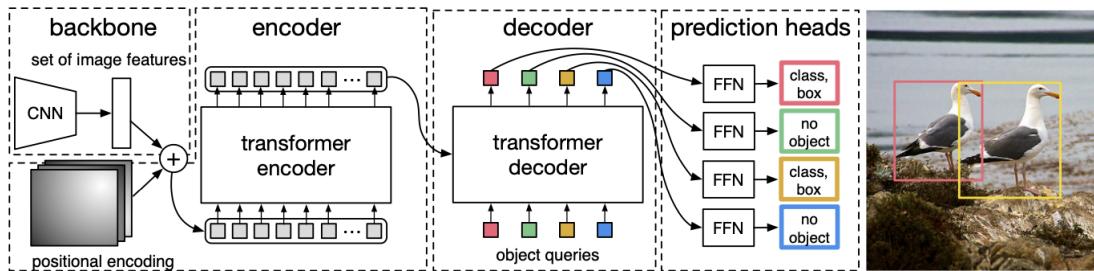


**Şekil 5.1** YOLOv8 Model Yapısı

- **F1 Score Curve:** Bu eğri, çeşitli eşiklerdeki F1 puanını temsil eder. Bu eğrinin yorumlanması, modelin farklı eşiklerdeki yanlış pozitifler ve yanlış negatifler arasındaki dengesini hakkında fikir verebilir.
- **Precision-Recall Curve:** Herhangi bir sınıflandırma problemi için olmazsa olmaz bir görselleştirme olan bu eğri, çeşitli eşiklerde hassasiyet ve geri çağrı arasındaki değişimleri gösterir. Özellikle dengesiz sınıflarla çalışırken önemli hâle gelir.
- **Precision Curve:** Bir sınıflandırma modelinin farklı kesim noktalarında hassasiyet değerlerini gösteren bu grafik hassasiyet eğrisi olarak bilinir.
- **Recall Curve:** Geri çağrı (recall) değerleri, sınıflandırma modelinin farklı kesim noktalarında gösterilen bir grafiktir. Hassasiyet ve geri çağrı arasındaki dengeyi değerlendirme ve modelin performansını ayrıntılı bir şekilde inceleme olanağı sağlar. Bir sınıflandırma modelinin doğruluk ve sınıf dengesizliği üzerindeki performansını bütünsel bir şekilde değerlendiren bir ölçütür.
- **F1 Score:** Bir sınıflandırma modelinin doğruluk ve sınıf dengesizliği üzerindeki performansını bütünsel bir şekilde değerlendiren bir ölçütür. Hassasiyet (doğru pozitiflerin oranı) ve geri çağrı (gerçek pozitiflerin oranı) ölçütlerinin harmonik ortalamasını temsil eder.
- **Confusion Matrix:** Bir sınıflandırma modelinin ne kadar iyi çalıştığını değerlendirmek için kullanılan bir matristir. Bu matris, modelin tahminlerinin doğruluğunu ölçer ve özellikle sınıflandırma problemlerindeki performansı daha iyi anlamak için kullanılır.

### 5.2.2 Vision Transformer

Model olarak Facebook tarafından geliştirilmiş olan DETR kullanılmaya karar verilmiştir. DETR transformer ve CNN metodlarının birleştirilmesi ile oluşturulmuş bir yöntemdir. DETR, CNN kullanarak bir görseldeki özellikleri ayıklar. Daha sonrasında oluşan çıktıya positional encoding ekleyerek bunu transformer encoder mimarisine girdi olarak ileter. Pozisyon bilgisi eklenmiş olan bu girdi parçalarına object queries denir. Transformer encoder ilk olarak gelen girdiyi küçültür. Daha sonrasında her bir encoder içerisinde bulunan self-attention mekanizmasına iletir. Burada işlenen görüntüler decoder ile çözümlenir ve FFN'e ilettilir. FFN 3 katmanlı bir yapıdır. Bu katmanlar ReLU aktivasyon fonksiyonu, gömülü bir dizi ve lineer projeksiyondur. FFN bounding box'ın merkez noktasını, genişliğin ve boyunu tahmin eder. Böylece obje tespit edilmiş olur.



Şekil 5.2 DETR Model Yapısı

### 5.3 Girdi-Çıktı Tasarımı

Araç içerisinde elde edilen gerçek zamanlı görüntülerde veya fotoğraflarda istenen objelerin tespit edilip bounding-box içerisinde alınarak sınıf isminin ve tespitin doğruluk oranının belirtilmesi sistemin çıktıları olarak belirlenmiştir.

# 6

## Uygulama

Uygulamayı geliştirmek için genellikle yapay zeka, görüntü işleme ve derin öğrenme modellerinin geliştirildiği, Python kodlarının yazıldığı Google Colab'ten faydalanilmıştır. Bu platformun kullanılmasının sebebi sunmuş olduğu GPU(V100)'nun bilgisayarlarındaki CPU'dan çok daha hızlı olmasıdır. İlk olarak YOLO modelini eğitmek için gerekli olan ultralytics, roboflow gibi kütüphaneler eklenmiştir, sonrasında veri seti Google Drive üzerinden çekileceğinden dolayı mount işlemi yapılmıştır. Daha sonra modelin eğitilmesinde kullanılan fonksiyonlar ve parametreler belirlendikten sonra aşağıdaki 6.1 kod parçası yazılarak YOLO modeli 50 epoch ile 640x640 pikselden oluşan görseller eğitilmeye başlanmıştır.

```
[ ] !yolo task=detect mode=train model=yolov8s.pt data="/content/drive/MyDrive/TrafficSignDetection50/project/datasets/road_signs/data.yaml" epochs=50 imgsz=640
```

Şekil 6.1 Modelin Eğitilmesi İçin Yazılan Kod Parçası

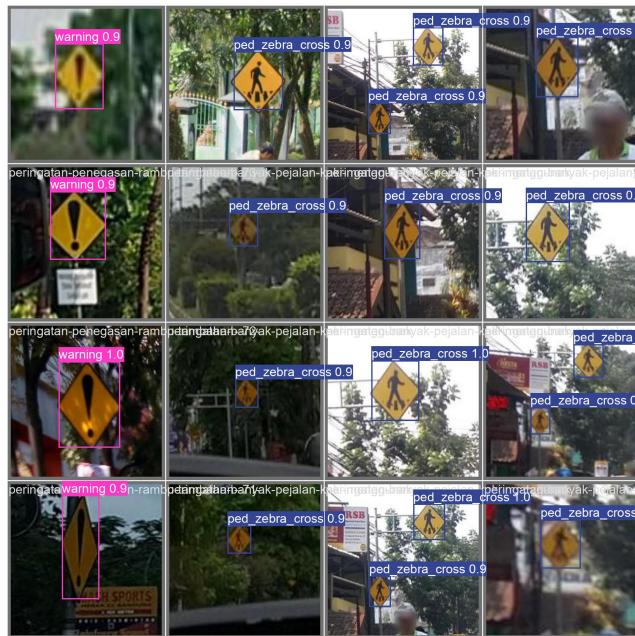
Model eğitildikten sonra weights klasörünün altında "last.pt" ve "best.pt" dosyaları oluşmuştur. "last.pt" dosyası, model eğitimi tamamlandıktan sonraki durumu temsil eder. "best.pt" dosyası ise modelin en iyi doğruluk veya en düşük kayıp değeri elde edildiği noktada kaydedilen ağırlıkları içerir. Bu sebepten dolayı objeleri tespit etmek üzere "best.pt" modeli kullanılır. Aşağıda trafik esnasında araç içinden çekilen bir görsel 6.3 ve objenin yüzde kaç oranda tespit ettiğini belirten görsel eklenmiştir.

```
[ ] !yolo task=detect mode=predict model={HOME}/runs/detect/train/weights/best.pt conf=0.25 source='/content/drive/MyDrive/TrafficSignDetection50/Images/stopsign.jpg'
```

Şekil 6.2 Obje Tespiti Yapılması İçin Yazılan Kod Parçası



Şekil 6.3 Yolov8 Obje Tespiti 1



Şekil 6.4 Yolov8 Obje Tespiti 2

DETR modelini eğitirken ise aynı şekilde Google Colab kullanılmış, GPU olarak Colab'in sağlamış olduğu yüksek performans gösteren V100 seçilmiştir. RoadSigns dataseti COCO formatında google drive üzerinden bağlanmıştır. Vision Transformer için kullanılan transformers, supervision, torch, cv2 gibi kütüphaneler import edilmiştir. Sonrasında gerekli fonksiyonlar yazılmış modelin 100 epoch değerinde eğitilmesi için aşağıdaki kod 6.5 yazılmıştır. Her bir epoch değerinden sonra oluşan loss değerlerini gözlelemek için TensorBoard kullanılmıştır.

```
▶ from pytorch_lightning import Trainer  
  %cd {HOME}  
MAX_EPOCHS = 100  
trainer = Trainer(devices=1, accelerator="gpu", max_epochs=MAX_EPOCHS, gradient_clip_val=0.1, accumulate_grad_batches=8, log_every_n_steps=5)  
trainer.fit(model)
```

Şekil 6.5 DETR Modelinin Eğitilmesi İçin Yazılan Kod Bloğu

Model eğitimi tamamlandıktan sonra modelin doğru çalışıp çalışmadığını gözlemllemek için veri seti içerisindeki test klasöründen rastgele olarak görsel seçip bunların tespiti yapılmıştır. Aşağıda verilen görsel 6.6'da DETR obje tespiti örneği görülmektedir.



Şekil 6.6 DETR Objə Tespiti

# 7

## Deneysel Sonuçlar

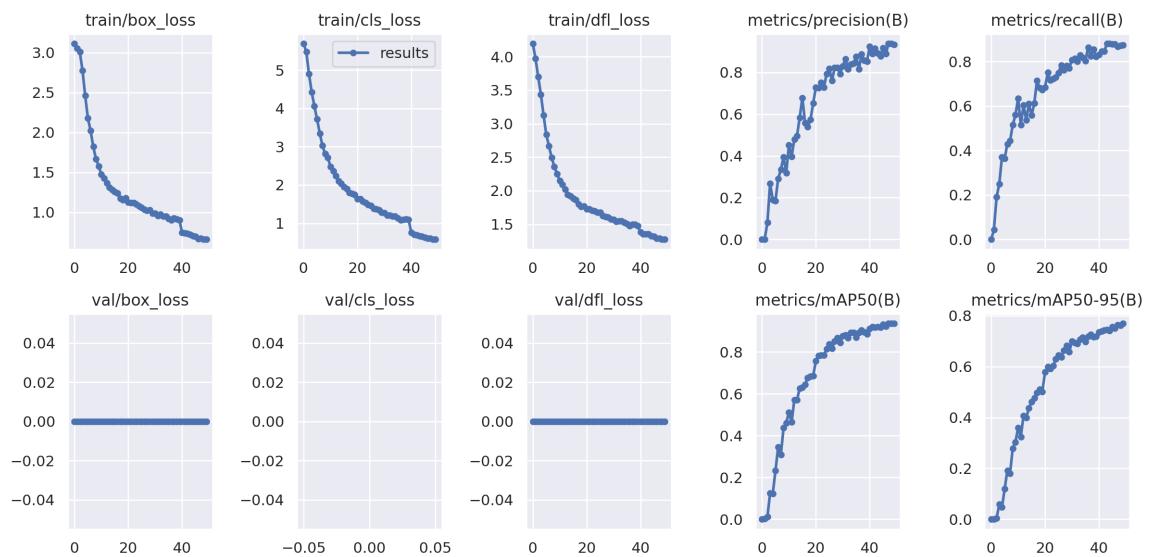
Modellerin performansını ölçmek için kullanılan kriterler "IoU", "AP", "mAP", "Precision", "Recall", "F1 Score" şeklidir. Model eğitildikten sonra 229 farklı görsel ile test edilmiştir. Testlerin açıklamaları ve sonuçları aşağıda anlatılmaktadır.

### 7.1 YOLOv8

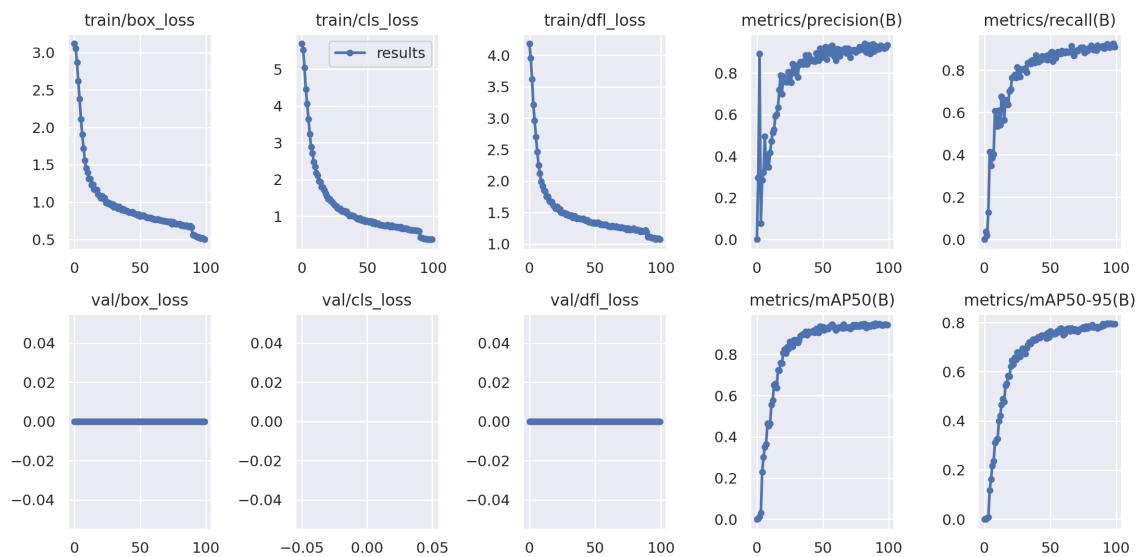
YOLOv8 mimarisinde modelin eğitimi ekleyen bir çok hiperparametre bulunmaktadır. Bu parametrelerden birisi "Epoch"tur. Epoch bir modelin eğitimindeki iterasyon sayısını ifade eden parametredir. Bir iterasyona karşılık gelen bir epoch, veri setinde verilen tüm görsellerin modele girdi olarak verilmesidir. Epoch'u belirlerken her bir iterasyondaki precision ve loss fonksiyonlarının çıktısı dikkate alınmaktadır. Precision değeri bire yaklaştıkça modelin doğruluk oranının arttığı anlamına gelir. Kullanılan loss fonksiyonları ve çıktılarının anlamları aşağıda verilmiştir.

- **Bounding Box Loss:** Modelin tahmin ettiği bounding box ile veri setindeki bounding box arasındaki farkı gösterir. Nesnenin konumunu daha iyi bir şekilde öğrenmesini sağlar.
- **Classification Loss:** Modelin tahmin ettiği sınıf ile veri setindeki sınıf arasındaki uyuşmayı gösterir. Nesne sınıflarının doğru belirlenmesinde kullanılır.
- **Distance IoU Loss:** Bounding boxların içe içe geçme durumlarında daha hassas bir obje tespti elde etmek için kullanılır. Bu değer ne kadar düşükse hassasiyet o kadar fazla demektir.

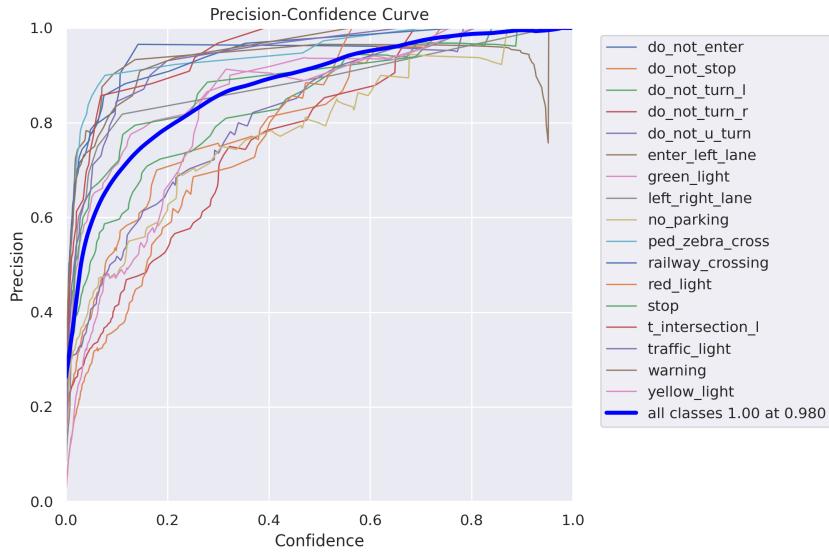
Model, birden çok epoch değerleri ile denemeler yapılarak eğitilmiş ve grafikler ile sonuçlar değerlendirilmiştir. 50 ve 100 epoch değerlerinde eğitilmiş modeller ile üretilen sonuçlar grafiklere dönüştürülmüştür. Aşağıdaki grafiklerde bu sonuçlar gözlemlenmektedir.



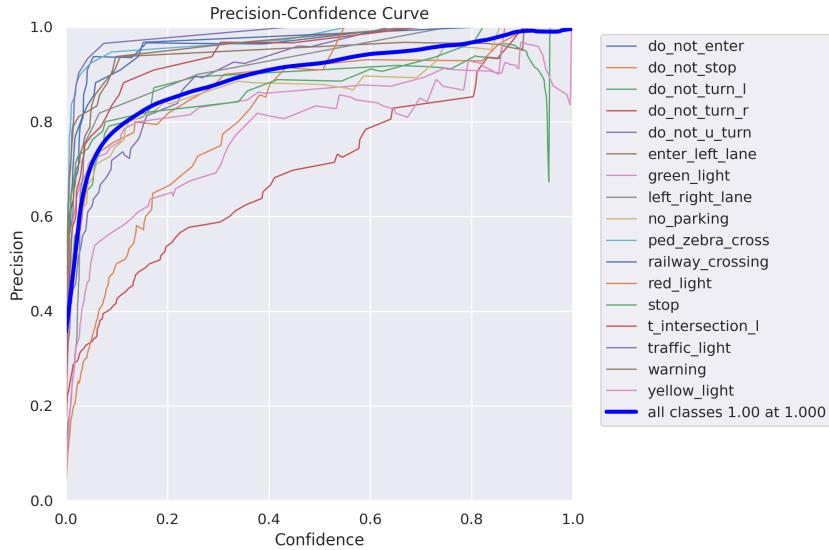
**Şekil 7.1 Results Curve 50 Epoch**



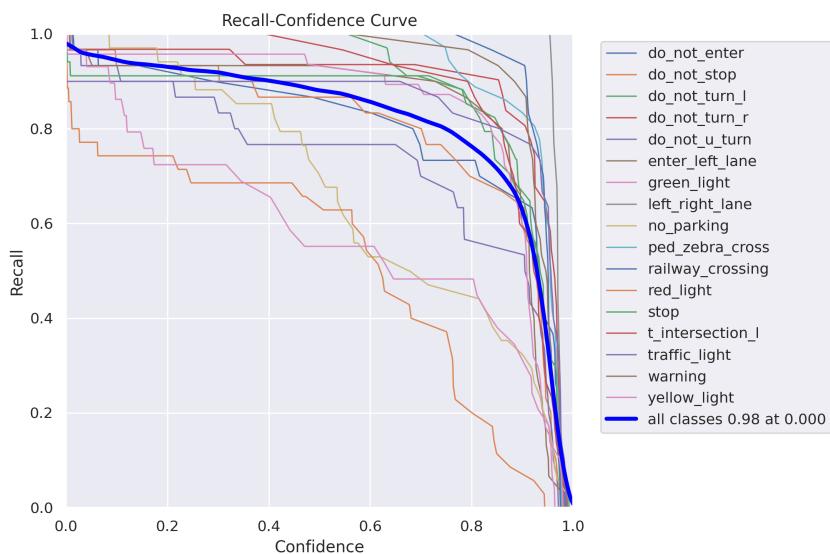
**Şekil 7.2 Results Curve 100 Epoch**



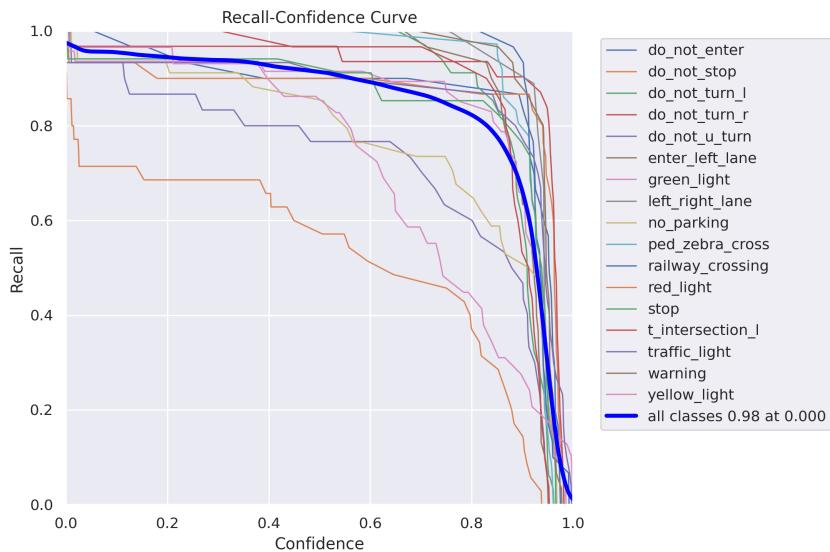
**Şekil 7.3** Precision-Confidence Curve 50 Epoch



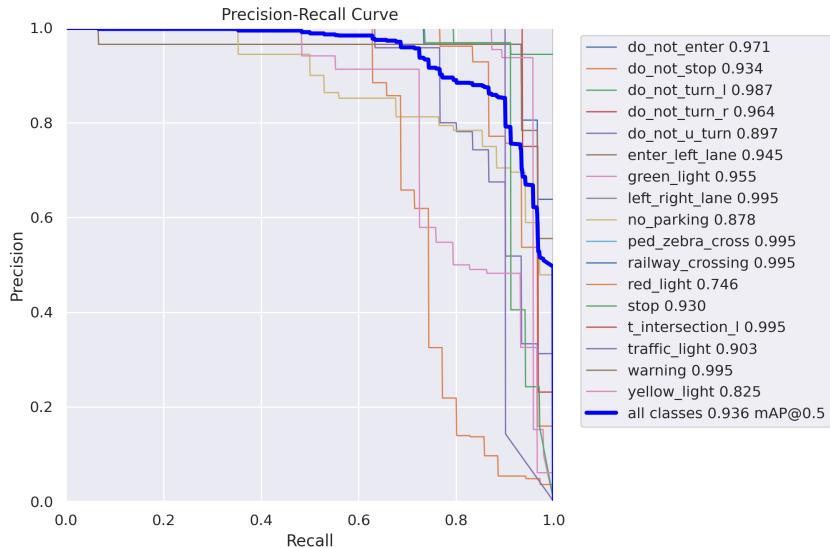
**Şekil 7.4** Precision-Confidence Curve 100 Epoch



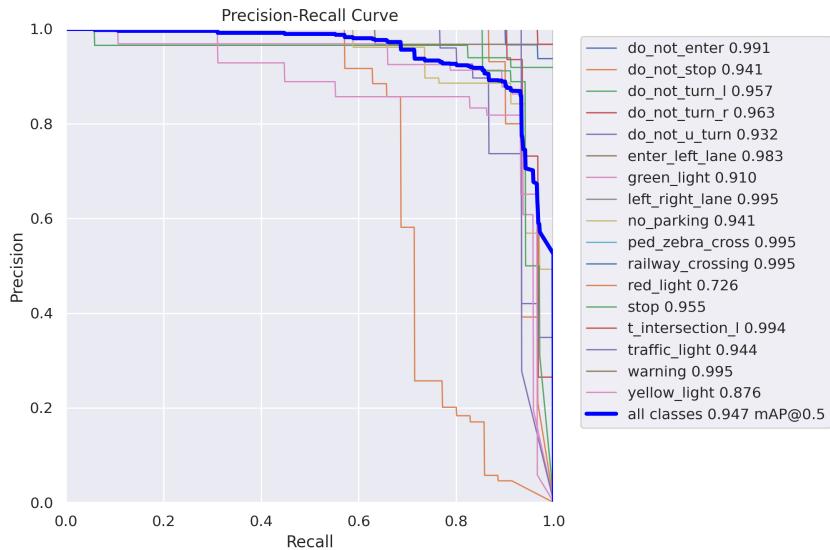
**Şekil 7.5 Recall-Confidence Curve 50 Epoch**



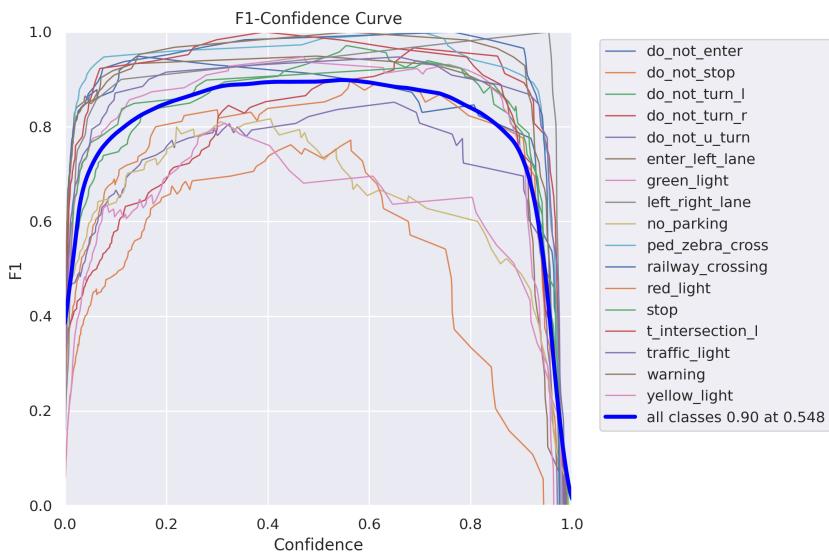
**Şekil 7.6 Recall-Confidence Curve 100 Epoch**



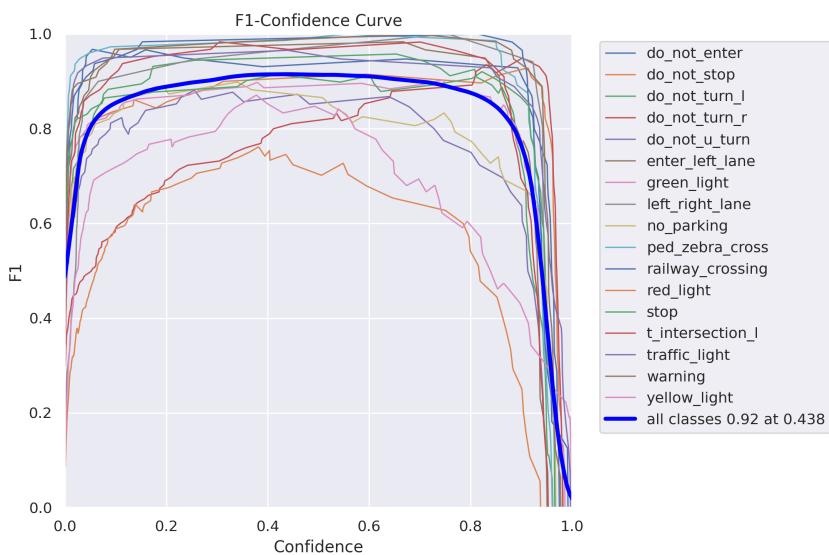
Şekil 7.7 Precision-Recall Curve 50 Epoch



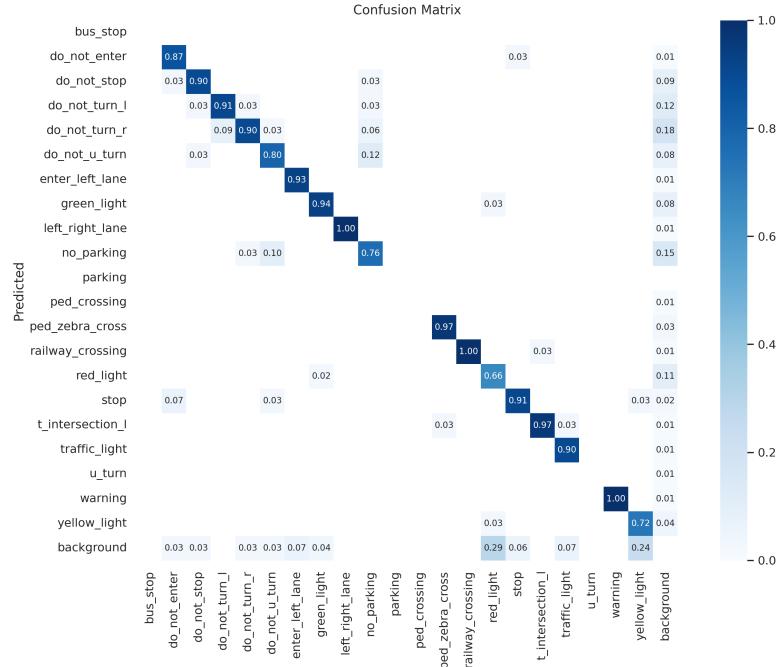
Şekil 7.8 Precision-Recall Curve 100 Epoch



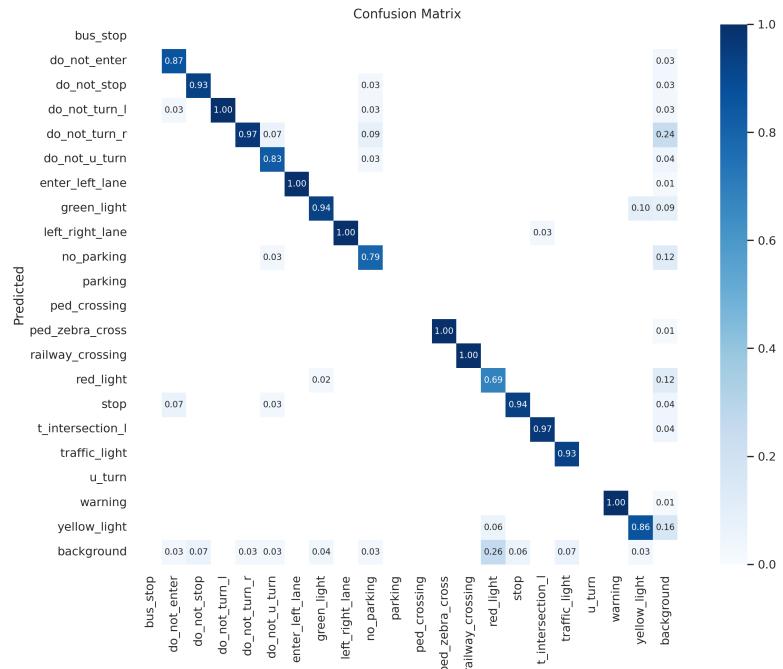
**Şekil 7.9 F1 Score 50 Epoch**



**Şekil 7.10 F1 Score 100 Epoch**



**Şekil 7.11** Confusion Matrix 50 Epoch

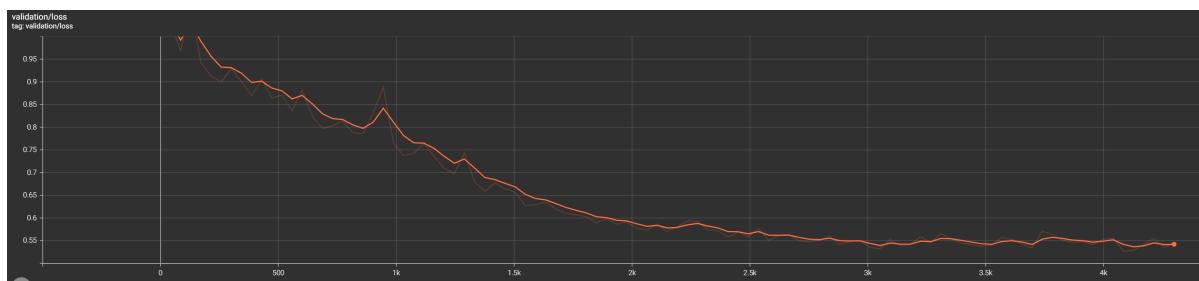


**Şekil 7.12** Confusion Matrix 100 Epoch

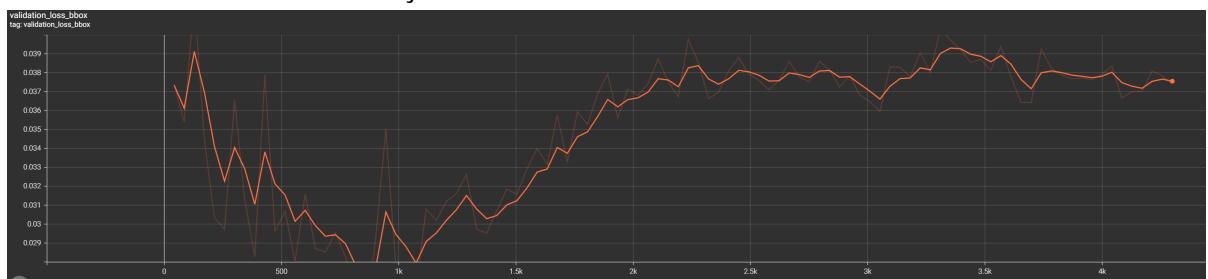
## 7.2 DETR

Transformer ve CNN mimarilerini kullanan DETR'de de YOLOv8 modelinde kullanılan test fonksiyonları kullanılmıştır. Bu iki farklı modelin karşılaştırılmasında objektifliği sağlamak için önem arz etmektedir. Aşağıda bu fonksiyonlara ait çıktılar bulunmaktadır.

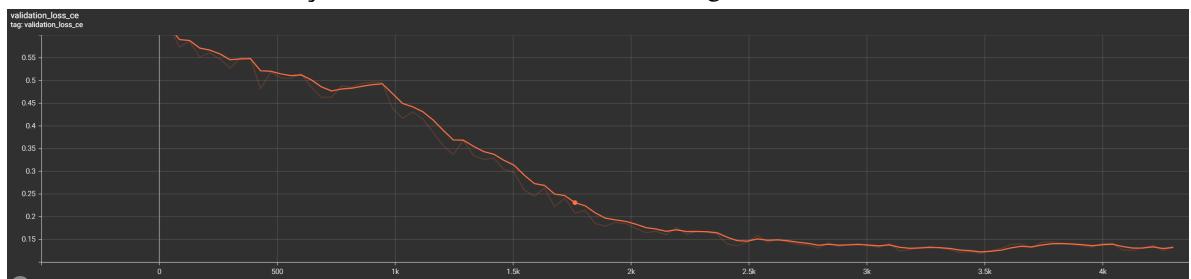
- **Training Loss:** Modelin eğitim verilerindeki genel performansını gösterir. Bu değer, modelin öğrenme sürecindeki başarısını ifade eder.
- **Training Bounding Box Loss:** Nesnelerin konumlarını tahmin etmedeki hata miktarını gösterir. Yani, modelin nesnelerin yerleşimini ne kadar doğru öğrendiğini belirtir.
- **Training Loss CE:** Modelin nesneleri doğru bir şekilde sınıflandırma becerisini gösterir. Sınıflandırma ile ilgili hataları ifade eder.
- **Training Loss GIOU:** Modelin nesnelerin konumları ve segmentasyonları üzerindeki doğruluğunu değerlendirir.
- **Training Cardinality:** Modelin eğitim verilerinde nesne sayısını doğru tahmin edip edemediğini belirtir.
- **Validation Loss:** Modelin doğrulama verilerindeki genel başarısını gösterir. Bu, modelin eğitim verilerinden bağımsız bir şekilde ne kadar iyi performans gösterdiğini ifade eder.
- **Validation Bounding Box Loss:** Doğrulama verilerindeki nesnelerin konumlarını tahmin etmedeki hata miktarını gösterir.
- **Validation Loss CE:** Doğrulama verilerindeki nesneleri doğru bir şekilde sınıflandırma becerisini gösterir.
- **Validation Loss GIOU:** Doğrulama verilerindeki nesnelerin konumları ve segmentasyonları üzerindeki doğruluğunu değerlendirir.
- **Validation Cardinality Error:** Modelin doğrulama verilerindeki nesne sayısını doğru tahmin edip edemediğini belirtir.



Şekil 7.13 Validation Loss



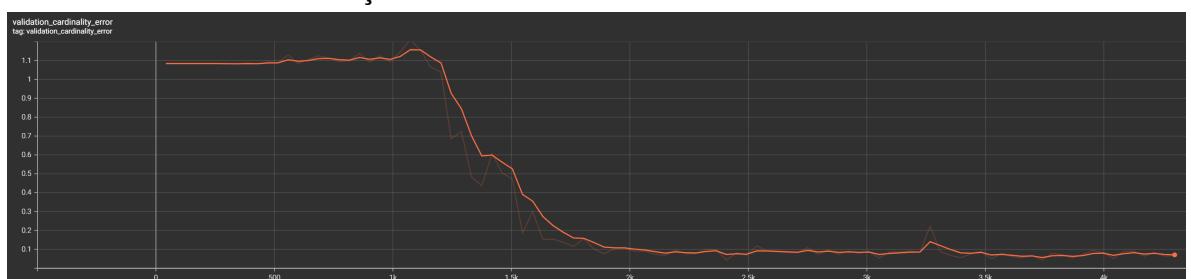
Şekil 7.14 Validation Bounding Box Loss



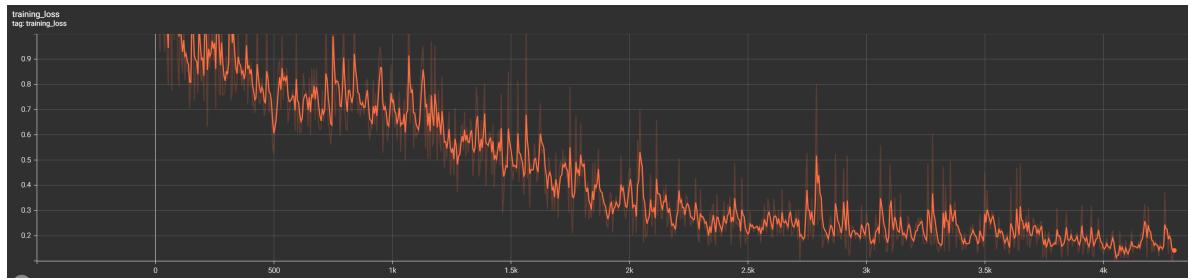
Şekil 7.15 Validation Loss CE



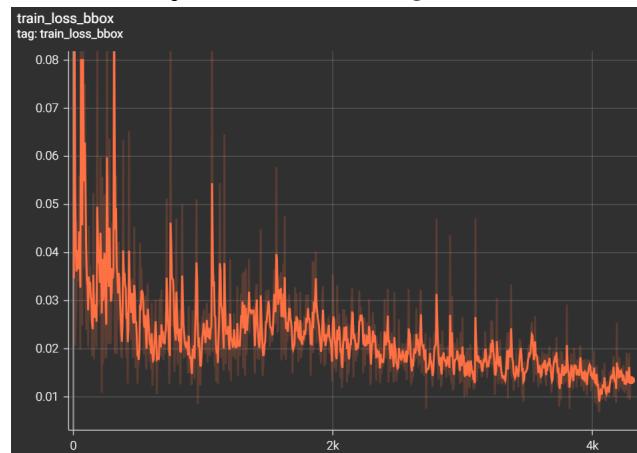
Şekil 7.16 Validation Loss GIOU



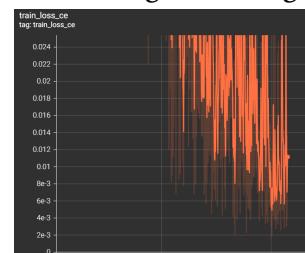
Şekil 7.17 Validation Cardinality Error



Şekil 7.18 Training Loss



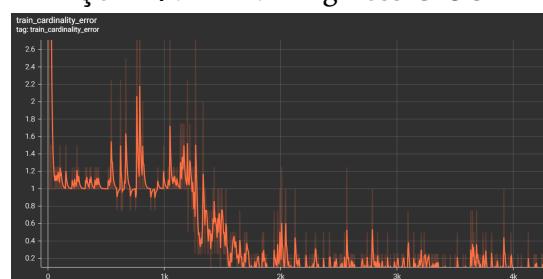
Şekil 7.19 Training Bounding Box Loss



Şekil 7.20 Training Loss CE



Şekil 7.21 Training Loss GIOU



Şekil 7.22 Training Cardinality

# 8

## Performans Analizi

1376 farklı görsel ile eğitilen model, 50 ve 100 epoch değerleri ile eğitilmiştir. 50 epoch ile eğitilen modelin eğitimi 4 saat sürerken 100 epoch ile eğitilen model 8 saatte eğitilmiştir. DETR modeli ise 100 epochla 10 saatte eğitilmiştir.

### 8.1 YOLOv8

Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
50/50	4.68G	0.66	0.579	1.276	20	640: 100% 86/86 [00:36<00:00, 2.36it/s]
	Class	Images	Instances	Box(P	R	mAP50 mAP50-95): 100% 16/16 [00:11<00:00, 1.43it/s]
	all	488	529	0.934	0.875	0.936 0.77

Şekil 8.1 YOLOv8 50 Epoch mAP Değeri

Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
100/100	4.92G	0.5007	0.3704	1.072	16	640: 100% 86/86 [00:10<00:00, 8.41it/s]
	Class	Images	Instances	Box(P	R	mAP50 mAP50-95): 100% 16/16 [00:06<00:00, 2.45it/s]
	all	488	529	0.936	0.91	0.942 0.795

Şekil 8.2 YOLOv8 100 Epoch mAP Değeri

Yukarıdaki görsellerde 8.1 8.2 sırasıyla 50 ve 100 epoch ile eğitilmiş YOLOv8s modellerinin mean Average Precision değerleri görülmektedir. 50 epoch ile eğitilmiş modelin mAP50 değeri 0.936 iken 100 epoch ile eğitilen modelin mAP50 değeri 0.942 olarak hesaplanmıştır.

Model summary: 168 layers, 11133711 parameters, 0 gradients, 28.5 GFLOPs							
Class	Images	Instances	Box(P	R	mAP50	mAP50-95)	: 100% 16/16 [00:13<00:00, 1.16it/s]
all	488	529	0.934	0.875	0.936	0.77	
do_not_enter	488	30	0.962	0.852	0.971	0.86	
do_not_stop	488	30	0.86	0.867	0.934	0.865	
do_not_turn_l	488	34	0.916	1	0.987	0.889	
do_not_turn_r	488	31	0.86	0.935	0.964	0.887	
do_not_u_turn	488	30	0.92	0.767	0.897	0.753	
enter_left_lane	488	30	0.965	0.928	0.945	0.816	
green_light	488	47	0.936	0.928	0.955	0.708	
left_right_lane	488	9	0.903	1	0.995	0.928	
no_parking	488	34	0.845	0.644	0.878	0.763	
ped_zebra_cross	488	36	0.977	1	0.995	0.818	
railway_crossing	488	30	0.982	1	0.995	0.734	
red_light	488	35	0.94	0.629	0.746	0.487	
stop	488	34	0.919	0.912	0.93	0.731	
t_intersection_l	488	30	1	0.968	0.995	0.78	
traffic_light	488	30	0.984	0.9	0.903	0.761	
warning	488	30	0.996	1	0.995	0.855	
yellow_light	488	29	0.914	0.552	0.825	0.452	

Şekil 8.3 YOLOv8 50 Epoch AP Değerleri

Model summary: 168 layers, 11133711 parameters, 0 gradients, 28.5 GFLOPs						
Class	Images	Instances	Box(P)	R	mAP50	mAP50-95): 100% 16/16 [00:06<00:00, 2.40it/s]
all	488	529	0.914	0.923	0.947	0.796
do_not_enter	488	30	0.968	0.9	0.991	0.895
do_not_stop	488	30	0.912	0.9	0.941	0.863
do_not_turn_l	488	34	0.905	1	0.957	0.858
do_not_turn_r	488	31	0.687	0.968	0.963	0.896
do_not_u_turn	488	30	0.947	0.8	0.932	0.799
enter_left_lane	488	30	0.955	1	0.983	0.882
green_light	488	47	0.863	0.915	0.91	0.695
left_right_lane	488	9	0.933	1	0.995	0.936
no_parking	488	34	0.881	0.871	0.941	0.821
ped_zebra_cross	488	36	0.972	1	0.995	0.836
railway_crossing	488	30	0.981	1	0.995	0.769
red_light	488	35	0.901	0.629	0.726	0.486
stop	488	34	0.889	0.94	0.955	0.774
t_intersection_l	488	30	0.967	0.973	0.994	0.784
traffic_light	488	30	0.998	0.933	0.944	0.782
warning	488	30	0.975	1	0.995	0.891
yellow_light	488	29	0.808	0.871	0.876	0.568

Şekil 8.4 YOLOv8 100 Epoch AP Değerleri

Yukarıdaki görsellerde 8.3 8.4 sırasıyla 50 ve 100 epoch ile eğitilmiş YOLOv8s modellerinin her bir sınıfında hesaplanan average precision değeri verilmiştir. Görüldüğü üzere 100 epoch ile eğitilmiş modelin doğruluk oranı diğer modele kıyasla daha iyi sonuçlar vermektedir.

## 8.2 ViT - DETR

Running evaluation...
100% [██████████] 58/58 [00:39<00:00, 2.81it/s]
Accumulating evaluation results...
DONE (t=0.05s).
IoU metric: bbox
Average Precision (AP) @[ IoU=0..50:0.95   area= all   maxDets=100 ] = 0.734
Average Precision (AP) @[ IoU=0..50   area= all   maxDets=100 ] = 0.894
Average Precision (AP) @[ IoU=0..75   area= all   maxDets=100 ] = 0.878
Average Precision (AP) @[ IoU=0..50:0.95   area= small   maxDets=100 ] = -1.000
Average Precision (AP) @[ IoU=0..50:0.95   area=medium   maxDets=100 ] = -1.000
Average Precision (AP) @[ IoU=0..50:0.95   area= large   maxDets=100 ] = 0.734
Average Recall (AR) @[ IoU=0..50:0.95   area= all   maxDets= 1 ] = 0.773
Average Recall (AR) @[ IoU=0..50:0.95   area= all   maxDets= 10 ] = 0.773
Average Recall (AR) @[ IoU=0..50:0.95   area= all   maxDets=100 ] = 0.773
Average Recall (AR) @[ IoU=0..50:0.95   area= small   maxDets=100 ] = -1.000
Average Recall (AR) @[ IoU=0..50:0.95   area=medium   maxDets=100 ] = -1.000
Average Recall (AR) @[ IoU=0..50:0.95   area= large   maxDets=100 ] = 0.773

Şekil 8.5 ViT Tabanlı DETR Modelinin AP Değerleri

100 epoch ile eğitilen ViT tabanlı DETR modelinde ulaşılan maksimum average precision değerinin 0.894 olduğu gözlemlenmiştir. Bu verilere dayanarak DETR modelinin YOLOv8 modellerine göre daha düşük doğruluk oranı yakaladığı saptanmıştır.

# 9

## Sonuç

---

Proje araçlar için görüntülerden trafik durumunu tasvir edebilecek bir model geliştirmeye odaklanmaktadır. Proje kapsamında iki farklı metod ile birden fazla model üretilmiştir. Bu modeller, 2023 yılında çıkış做的 olan CNN tabanlı YOLOv8 metodu ve doğal dil işlemede kullanılan, yakın zamanda görüntü işleme alanında kullanılmaya başlanmış olan transformerları CNN ile birlikte kullanan DETR metodu ile geliştirilmiştir. Modeller farklı epoch değerleri ile eğitilmiş 50 ve 100 epoch değeri arasında süre farkı iki kat olmasına rağmen doğruluk oranı performansı etkileyeceğin oranda değişmemiştir. YOLOv8 modeli ortalama doğruluk oranı %93.6 ile oldukça iyi bir tespit oranına sahiptir. DETR modeli ise %89.4 ortalama doğruluk oranına ulaşmıştır. YOLOv8'in DETR modeline göre daha iyi bir tespit oranına eriştiği gözlemlenmiştir. Videolar üzerinden yapılan denemelerde DETR'in yetersiz kaldığı YOLOv8'in gerçek zamanlı tespitte DETR'den daha iyi olduğu kanısına varılmıştır. Ayrıca DETR modelinin eğitim süresinin aynı epoch sayısında YOLOv8 modeline göre oldukça uzun olduğu görülmüştür. Bu sebeple trafik durumu tasvirinde YOLOv8 modeli hem hız hem de doğruluk oranı bakımından transformer tabanlı DETR modelinden daha efektif bir çözüm olarak kabul edilebilir.

## Referanslar

---

- [1] C.-j. Li, Z. Qu, S.-y. Wang, and L. Liu, “A method of cross-layer fusion multi-object detection and recognition based on improved faster r-cnn model in complex traffic environment,” *Pattern Recognition Letters*, vol. 145, pp. 127–134, 2021.
- [2] F. N. Ortataş and M. Kaya, “Performance evaluation of yolov5, yolov7, and yolov8 models in traffic sign detection,” in *2023 8th International Conference on Computer Science and Engineering (UBMK)*, IEEE, 2023, pp. 151–156.
- [3] J. M. Kaleybar, H. Khaloo, and A. Naghipour, “Efficient vision transformer for accurate traffic sign detection,” *arXiv preprint arXiv:2311.01429*, 2023.
- [4] A. Afdhal, K. Saddam, S. Sugiarto, Z. Fuadi, and N. Nasaruddin, “Real-time object detection performance of yolov8 models for self-driving cars in a mixed traffic environment,” in *2023 2nd International Conference on Computer System, Information Technology, and Electrical Engineering (COSITE)*, IEEE, 2023, pp. 260–265.
- [5] K. Lu, Y. Xu, and Y. Yang, “Comparison of the potential between transformer and cnn in image classification,” in *ICMLCA 2021; 2nd International Conference on Machine Learning and Computer Application*, VDE, 2021, pp. 1–6.
- [6] O. N. Manzari, A. Boudesh, and S. B. Shokouhi, “Pyramid transformer for traffic sign detection,” in *2022 12th International Conference on Computer and Knowledge Engineering (ICCKE)*, IEEE, 2022, pp. 112–116.
- [7] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, “End-to-end object detection with transformers,” in *European conference on computer vision*, Springer, 2020, pp. 213–229.

# Özgeçmiş

---

## BİRİNCİ ÜYE

**İsim-Soyisim:** Atakan SUCU

**Doğum Tarihi ve Yeri:** 07.02.2002, Mersin

**E-mail:** atakan.sucu@std.yildiz.edu.tr

**Telefon:** 0544 274 9488

**Staj Tecrübeleri:** Arge 24 Teknoloji A.Ş. Yazılım Departmanı

## İKİNCİ ÜYE

**İsim-Soyisim:** Mert YÜREKLİ

**Doğum Tarihi ve Yeri:** 28.06.2002, Antalya

**E-mail:** mert.yurekli@std.yildiz.edu.tr

**Telefon:** 0505 497 3662

**Staj Tecrübeleri:** TÜBİTAK BİLGE Yazılım Departmanı

## Proje Sistem Bilgileri

**Sistem ve Yazılım:** Python3, Google Colab, Tensorflow, Pytorch,

**Gerekli RAM:** 32GB

**Gerekli Disk:** 32GB