

Bu projede, insansız hava araçlarıyla çekilen görüntüler kullanılarak araçların otomatik olarak tanınması ve sınıflandırılması amaçlanmıştır.

# Veri Madenciliği Dönem Projesi

İHA Görüntüleri Üzerinde Derin  
Öğrenmeye Dayalı Nesne Tespiti  
ve Sınıflandırma

YUSUF GÜNEY - 22360859041

## YOLOv5 Tabanlı Araç Sınıflandırma Projesi Raporu

### 1. Proje Tanımı ve Amacı

Bu projenin amacı, insansız hava araçları (İHA) ile çekilmiş görüntülerde yer alan taşıtları görsel veri üzerinden sınıflandırmak ve tespit etmektir. YOLOv5 derin öğrenme modeli kullanılarak araçların türü (Car, Bus, Truck, Taxi, Motorcycle) otomatik olarak belirlenir.

### 2. Veri Seti Tanıtımı

Proje kapsamında kullanılan veri seti, İnsansız Hava Araçları (İHA) tarafından farklı açılardan ve yüksekliklerden çekilmiş hava görüntülerinden oluşmaktadır. Görüntülerde çeşitli araçlar farklı açılardan ve çevresel koşullarda yer almakta, bu da modeli gerçek dünya verisi üzerinde test etmek için uygun bir ortam sağlamaktadır. Veri seti, 5 farklı araç türünü içerecek şekilde etiketlenmiştir.

- Car
- Motorcycle
- Bus
- Truck
- Taxi

Görseller .jpg formatında, etiket dosyaları ise Pascal VOC XML formatında sunulmuştur. Eğitimde kullanılabilmesi için etiketler YOLO formatına dönüştürülerek modele uyumlu hale getirilmiştir.

Veri seti, akademik araştırmalar için hazırlanmış ve Data in Brief adlı bilimsel platformda yayımlanmıştır. Detaylı bilgi ve veri setine erişim için bağlantı:

<https://www.sciencedirect.com/science/article/pii/S2352340924002336>

Veri setinin ham halinde fazla görsel bulunduğu için ben bir kısmını kullanmadım benim kırpıp kullandım ve drive'a kaydettim drive linki:

[https://drive.google.com/file/d/1t8VefcJBfV\\_f7g\\_11V731jzZYRFCycgz/view?usp=drive\\_link](https://drive.google.com/file/d/1t8VefcJBfV_f7g_11V731jzZYRFCycgz/view?usp=drive_link)

#### Veri Seti Yapısı:

```
data_yolo/  
├── images/  
│   ├── train/  
│   ├── val/  
│   └── test/  
├── labels/  
│   ├── train/  
│   ├── val/  
│   └── test/  
└──
```


Her bir images/ klasöründe .jpg uzantılı resimler, labels/ klasöründe ise Pascal VOC formatında .xml uzantılı etiket dosyaları bulunmaktadır.

---

### 3. Uygulanan İşlemler ve Yöntemler

#### 3.1 YOLOv5 Kurulumu


YOLOv5, Ultralytics tarafından geliştirilen bir nesne tespiti framework'üdür. Google Colab ortamında şu komutlarla kurulmuştur:



```
1 !git clone https://github.com/ultralytics/yolov5
2 %cd yolov5
3 %pip install -r requirements.txt
4
```

#### 3.2 Veri Setinin Yüklenmesi

Veri seti sıkıştırılmış .zip formatında yüklenmiş ve aşağıdaki kod ile açılmıştır:



```
1 import zipfile
2
3 with zipfile.ZipFile("/content/dataset.zip", 'r') as zip_ref:
4     zip_ref.extractall("/content/data_yolo")
5
```

#### 3.3 XML → TXT Etiket Dönüşümü

YOLOv5 modeli yalnızca .txt formatındaki etiketlerle çalıştığı için, veri setindeki .xml dosyaları.txt formatına dönüştürülmüştür.

Dönüştürme işlemi sırasında, **convert\_annotation()** adlı özel bir Python fonksiyonu kullanılmıştır. Bu fonksiyon, her bir .xml dosyasını okuyarak içindeki bounding box koordinatlarını ve sınıf bilgilerini çıkarmış; ardından bu verileri YOLOv5'in beklediği biçimde normalize ederek .txt dosyası olarak kaydetmiştir.

Koordinatlar, görüntü boyutuna göre (x\_center, y\_center, width, height) formatında normalize edilmiştir. Her satırda, sınıf indeksinin ardından bu dört değer yer almaktadır. Dönüştürme işlemi tüm train, val ve test klasörleri için otomatik olarak uygulanmıştır.

```
1 import os
2 import xml.etree.ElementTree as ET
3
4 class_map = {'Car': 0, 'Motorcycle': 1, 'Bus': 2, 'Truck': 3, 'Taxi': 4}
5
6 def convert_annotation(xml_path, txt_path):
7     tree = ET.parse(xml_path)
8     root = tree.getroot()
9     size = root.find("size")
10    img_w = int(size.find("width").text)
11    img_h = int(size.find("height").text)
12
13    with open(txt_path, "w") as out_file:
14        for obj in root.findall("object"):
15            label = obj.find("name").text
16            if label not in class_map:
17                continue
18            class_id = class_map[label]
19            bndbox = obj.find("bndbox")
20            xmin = int(bndbox.find("xmin").text)
21            xmax = int(bndbox.find("xmax").text)
22            ymin = int(bndbox.find("ymin").text)
23            ymax = int(bndbox.find("ymax").text)
24
25            x_center = ((xmin + xmax) / 2) / img_w
26            y_center = ((ymin + ymax) / 2) / img_h
27            width = (xmax - xmin) / img_w
28            height = (ymax - ymin) / img_h
29
30            out_file.write(f"{class_id} {x_center:.6f} {y_center:.6f} {width:.6f} {height:.6f}\n")
31
32
33 splits = ['train', 'val', 'test']
34
35 for split in splits:
36     label_dir = f"/content/data_yolo/labels/{split}"
37     for file in os.listdir(label_dir):
38         if file.endswith(".xml"):
39             xml_path = os.path.join(label_dir, file)
40             txt_path = os.path.join(label_dir, file.replace(".xml", ".txt"))
41             convert_annotation(xml_path, txt_path)
42
```

Örnek bir satır göstermek gerekirse

```
2 0.435 0.672 0.321 0.194
```

Bu satır, “Bus” sınıfına (class ID: 2) ait bir nesnenin görüntü üzerindeki normalize koordinatlarını temsil eder.

### 3.4 data.yaml Dosyasının Oluşturulması

YOLOv5 modeli eğitim sırasında veri yollarını ve sınıf bilgilerini data.yaml dosyasından alır:

```
1 data_yaml = """
2 train: ../data_yolo/images/train
3 val: ../data_yolo/images/val
4 test: ../data_yolo/images/test
5
6 nc: 5
7 names: ['Car', 'Motorcycle', 'Bus', 'Truck', 'Taxi']
8 """
9
10 with open("data.yaml", "w") as f:
11     f.write(data_yaml)
12
```

### 3.5 Modelin Eğitimi

YOLOv5s mimarisi (küçük ve hızlı model) kullanılarak model eğitimi aşağıdaki parametrelerle başlatılmıştır:

```
1 !python train.py --img 416 --batch 16 --epochs 20 --data data.yaml --weights yolov5s.pt --name arac_siniflandirma_gpu
```

**--img 416** → Giriş görüntüleri 416x416 boyutunda yeniden boyutlandırılmıştır.

**--batch 16** → Eğitimde her adımda 16 görüntü kullanılmıştır.

**--epochs 20** → Model toplam 20 defa tüm eğitim verisi üzerinde eğitilmiştir.

**--data data.yaml** → Eğitim, doğrulama ve test verilerinin yolu ile sınıf isimleri bu dosyadan alınmıştır.

**--weights yolov5s.pt** → Eğitim öncesi yüklenen, önceden eğitilmiş YOLOv5s ağırlıkları kullanılmıştır.

**--name arac\_siniflandirma\_gpu** → Eğitimin sonuçları bu isimle bir klasörde saklanmıştır.

Eğitim süreci boyunca kayıplar (loss), mAP (mean Average Precision), precision ve recall gibi başarı metrikleri her epoch sonunda hesaplanmış ve grafiklerle kaydedilmiştir.

Modelin eğitimi sırasında test dosyası içerisindeki kullanılan görsellerin birkaçı aşağıda verilmiştir.





## 4. Eğitim Sonrası İşlemler ve Test

### 4.1 En İyi Modelin Kullanımı (best.pt)

Eğitim tamamlandıktan sonra en iyi ağırlık dosyası şu komutla test görselleri üzerinde kullanıldı:

```
1 !python detect.py --weights runs/train/arac_siniflandirma_gpu/weights/best.pt \
2 --img 416 --conf 0.4 --source ../data_yolo/images/test --name test_sonucлари
```

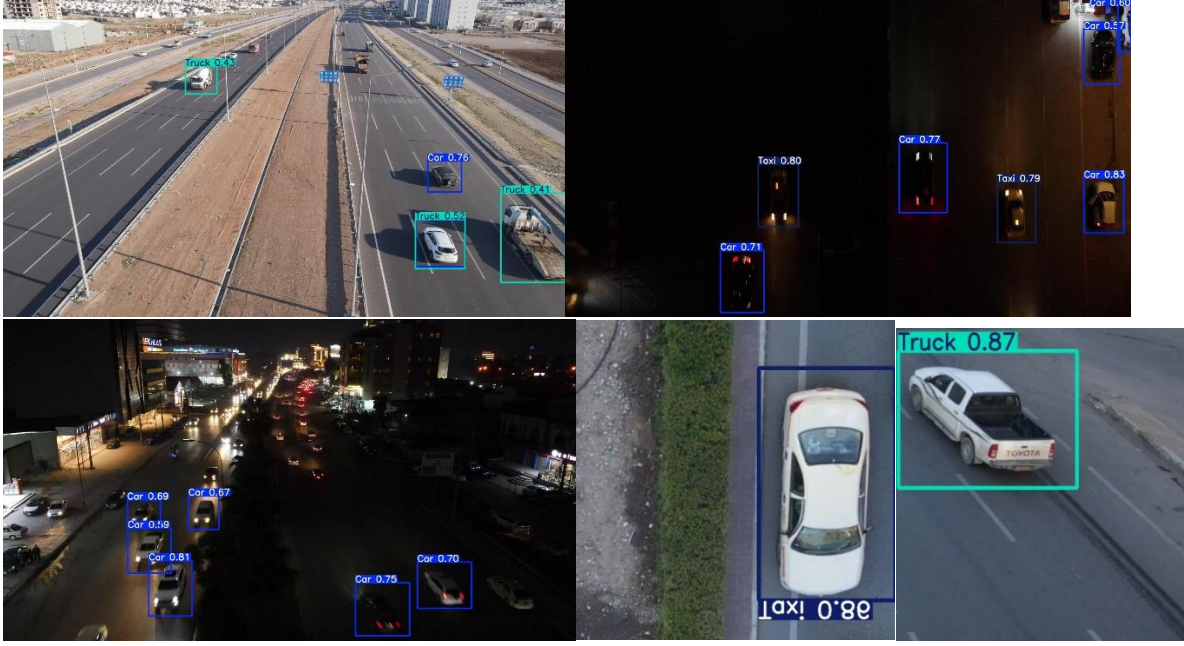
### 4.2 Test Sonuçlarının Görüntülenmesi

Tahmin edilen görsellerin gösterimi için Colab üzerinde şu kod kullanıldı:

```
1 import glob
2 from IPython.display import Image, display
3
4 image_paths = glob.glob('runs/detect/test_sonucлари/*.jpg')
5 for img_path in image_paths[:5]:
6     display(Image(filename=img_path))
```

Eğitim sonucu sınıflandırılmış araçları görüntülemek için birkaç görsel aşağıda görülmektedir.

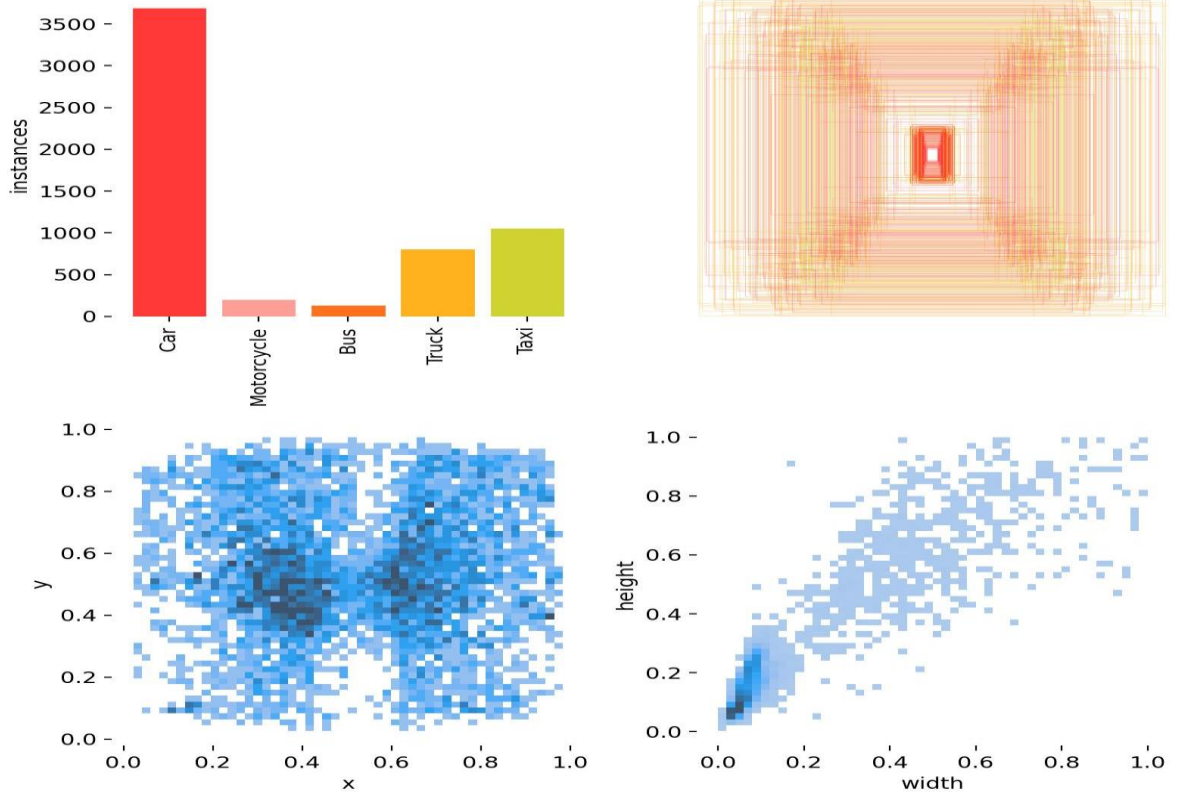




## 5. Analiz ve Değerlendirme

### 5.1 Araç Dağılım Grafiği

Etiket dağılım grafiği, veri setindeki her sınıfa ait örneklerin sayısını ve bounding box'ların konum yoğunluklarını gösterir. Bu görsel yardımıyla veri setinin sınıf dengesi analiz edilebilir. Özellikle bazı sınıfların eksikliği veya aşırı baskınlığı modelin öğrenme dengesini etkileyebilir.



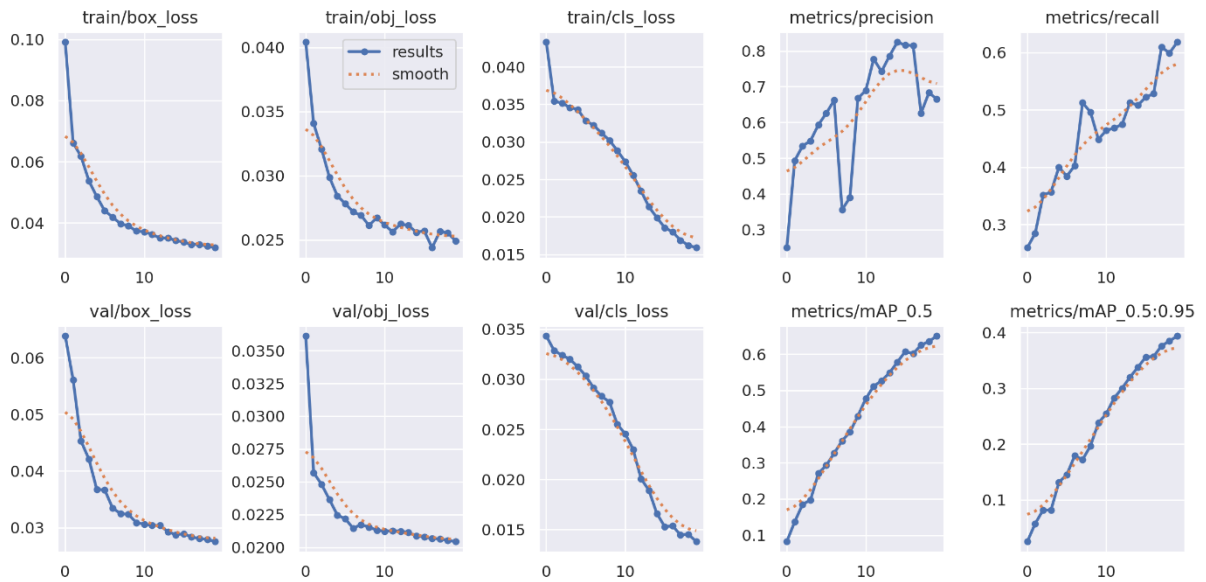


## 5.2 Eğitim Performans Grafikleri

Aşağıdaki metrikler zamana (epoch sayısına) göre ayrı ayrı çizilmiştir:

- **box\_loss**: Modelin bounding box koordinatlarını tahmin etmedeki hata değeri.
- **obj\_loss**: Modelin bir nesne var/yok tahminine ilişkin hata değeri.
- **cls\_loss**: Doğru sınıf tahmini yapamama durumundaki hata.
- **mAP@0.5 ve mAP@0.5:0.95**: Tüm sınıflar için ortalama doğruluk, Intersection over Union (IoU) oranına göre ölçülür.
- **Precision & Recall**: Modelin sınıf tahminlerindeki doğruluğu ve tüm doğru nesneleri bulma yeteneği.

Bu grafik yardımıyla modelin zaman içinde nasıl optimize olduğu, öğrenme eğrisinin durağanlaşıp durağanlaşmadığı ve aşırı öğrenme (overfitting) olup olmadığı analiz edilebilir.



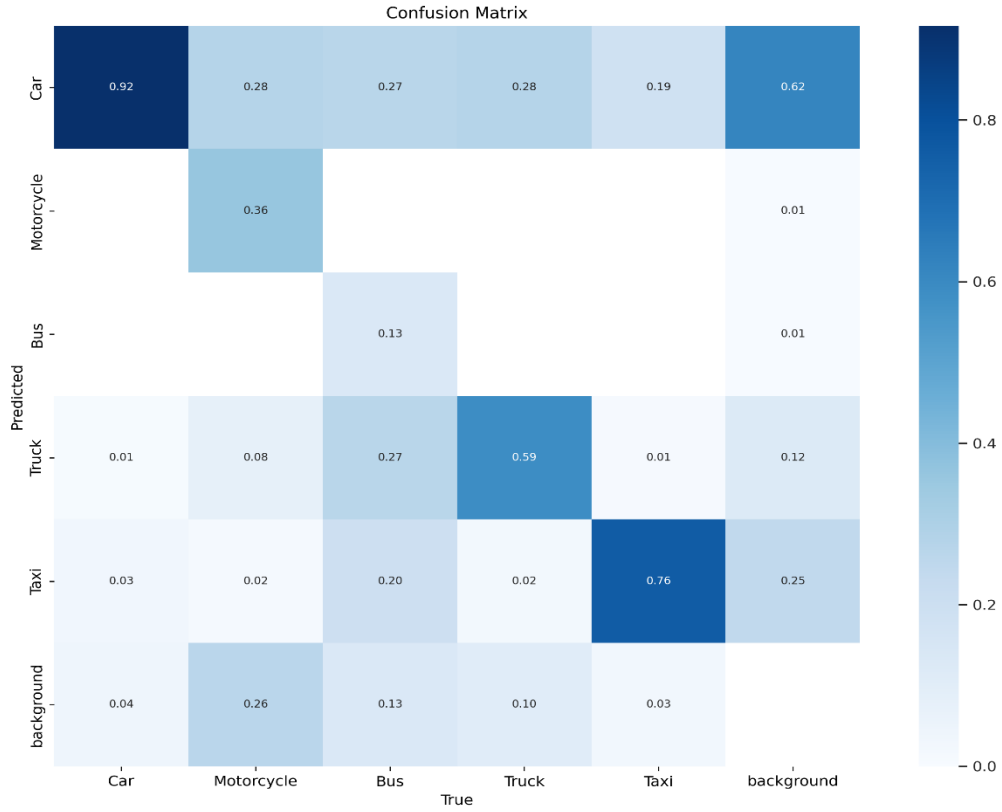
## 5.3 Karışıklık Matrisi (Confusion Matrix)

Karışıklık matrisi, modelin tahmin ettiği sınıflar ile gerçek sınıflar arasındaki örtüşmeyi gösteren bir görseldir. Her satır, **gerçek sınıfı**, her sütun ise **modelin tahmin ettiği sınıfı** temsil eder. Diagonal üzerindeki değerler doğru tahminleri, diagonal dışındakiler ise yanlış sınıf tahminlerini temsil eder.

Bu grafik, özellikle şu analizler için çok önemlidir:

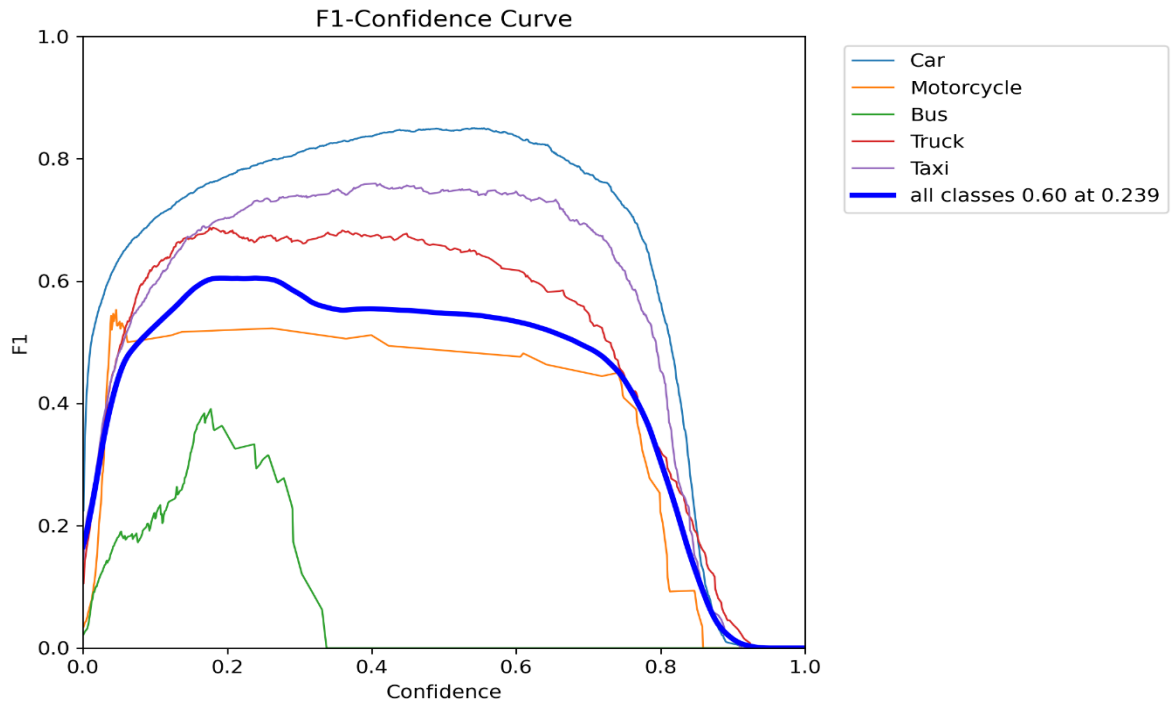
- Hangi sınıfların birbiriyle karıştırıldığı
- Belirli sınıfların tespitinde zorlanma olup olmadığı
- Veri dengesizliğinin etkisi

Örneğin “Car” sınıfının sıkça “Taxi” olarak tahmin edilmesi, bu iki sınıfın görüntüsel olarak benzer olduğunu ve modelin ayırmada zorlandığını gösterir.



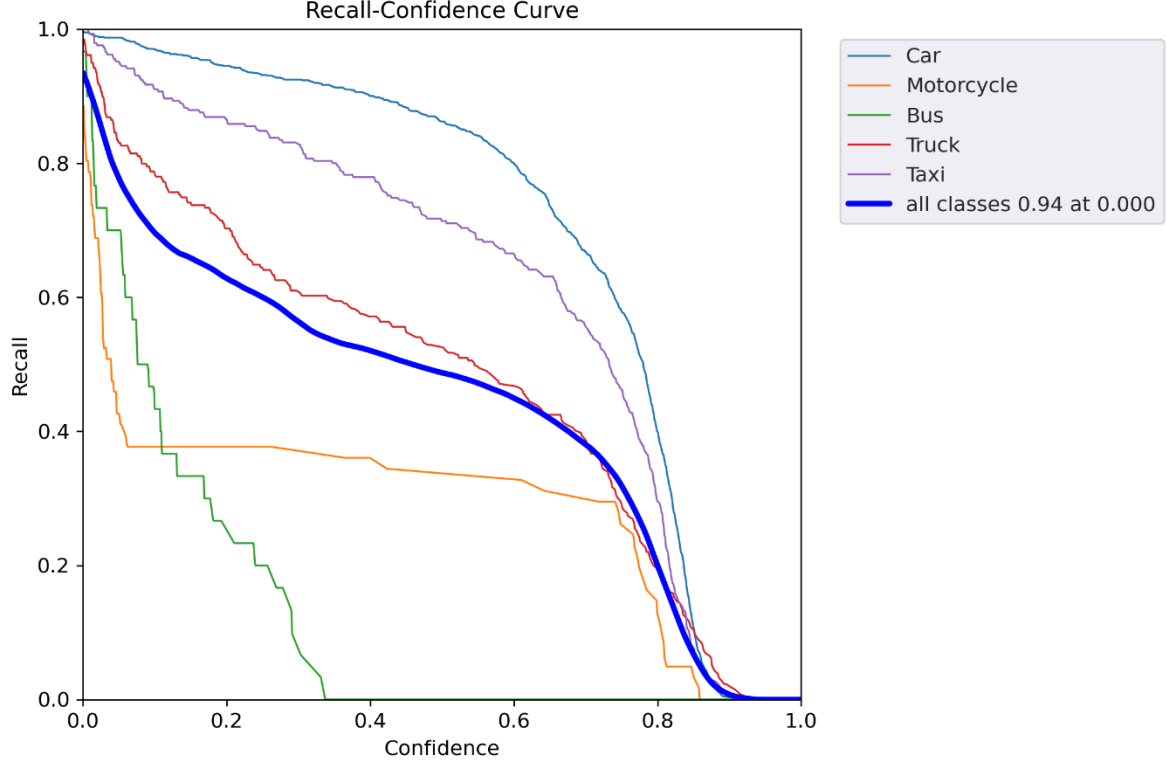
#### 5.4 F1 Score Eğrisi

F1 eğrisi, modelin doğruluğu (precision) ile geri çağırma (recall) arasındaki dengeyi yansıtır. F1 skoru, her iki metriği de göz önünde bulunduran bir başarı ölçüsüdür ve bu grafik, modelin hangi güven aralıklarında daha dengeli sonuçlar verdiğini gösterir. Eğrinin yüksek ve dengeli olması, modelin tüm sınıflarda güvenilir performans sergilediğini gösterir.



## 5.5 Recall Eğrisi

Recall eğrisi, confidence skoru arttıkça her bir sınıf için geri çağırma oranının nasıl değiştiğini görselleştirir. Bu grafik, modelin nesneleri kaçırma eğilimini ortaya koyar. Eğrinin yukarıya yakın olması, modelin daha az hata ile nesneleri tespit edebildiğini gösterir.



## 6. Sonuç ve Değerlendirme

- YOLOv5 modeli, İHA ile çekilmiş görüntülerdeki araçları yüksek doğrulukla tespit edebilmiştir.
- Eğitim boyunca kayıpların azalması ve mAP değerlerinin yükselmesi gözlemlenmiştir.
- Tahmin sonuçları gerçek görüntüler üzerinde başarıyla gözlemlenmiştir.
- Karışıklık matrisinde bazı sınıflar arasında karışma (Car vs Taxi) olmuştur.

### ◆ GitHub Paylaşımı ve Proje Dosya Yapısı

Proje kapsamında oluşturulan tüm veri, model, analiz ve dökümantasyon dosyaları, açık kaynaklı erişim amacıyla **GitHub** üzerinden paylaşılmıştır. Bu paylaşım sayesinde hem proje süreci hem de sonuçları şeffaf biçimde takip edilebilir.

GitHub deposu şu dosya yapısını içermektedir:

```
YOLOv5_Arac_Siniflandirma/  
├─ analysis/  
├─ eğitim_görselleri/  
├─ google_colab_defteri/  
├─ model/  
├─ test_sonuçlari/  
├─ yolov5_arac_siniflandirma_raporu.pdf  
├─ data.yaml  
└─ README.md
```

- analysis/: Eğitim süreci boyunca oluşturulan grafik ve performans görselleri.
- eğitim\_görselleri/: Eğitim sırasında kullanılan örnek görüntüler.
- google\_colab\_defteri/: Google Colab ortamında kullanılan .ipynb dosyası.
- model/: YOLOv5s mimarisiyle eğitilmiş model dosyası (best.pt).
- test\_sonuçlari/: Test görüntüleri üzerinde yapılan tahminlerin işaretlenmiş sonuçları.
- yolov5\_arac\_siniflandirma\_raporu.pdf: Bu belgenin PDF halidir.
- data.yaml: Eğitim, doğrulama ve test veri yolları ile sınıf etiketlerini içeren yapılandırma dosyası.
- README.md: Proje hakkında genel bilgiler, bağlantılar ve dizin açıklamalarını içeren tanıtım dosyası.

Tüm bu dosyalar ve klasörler GitHub üzerinde açık erişime sunulmuştur. Bağlantı ve erişim bilgileri için README dosyasına başvurulabilir.

---

## 8. Yaşanılan Zorluklar

Proje süresince karşılaşılan bazı teknik zorluklar ve bunlara bulunan çözümler aşağıda özetlenmiştir:

### 1. Model Eğitimi Süresinin Aşırı Uzun Olması:

- İlk aşamada Colab üzerinde GPU seçeneği aktif edilmeden eğitim başlatıldığı için modelin eğitim süresi yaklaşık 18-20 saat olarak tahmin edilmiştir.
- Daha sonra yapılan araştırmalar sonucunda "Runtime > Change runtime type" ayarlarından donanım hızlandırıcı olarak **GPU (örneğin T4)** seçilerek model eğitimi yaklaşık 1 saat içinde tamamlanmıştır.

### 2. Veri Setinin Formatının Uygun Olmaması:

- Başlangıçta etiketler .xml formatında olduğu için YOLOv5 ile uyumlu değildi.
- XML to TXT dönüşüm fonksiyonu (convert\_annotation()) yazılarak bu sorun başarıyla giderildi.

### 3. Klasör Yapısında Hatalar ve Erişim Sorunları:

- .zip dosyasının içeriği yanlış konumda açıldığı veya bazı klasörlerin eksik olması nedeniyle model eğitimi sırasında FileNotFoundError hataları alındı.
- Klasör yapısı dikkatle kontrol edilerek images/ ve labels/ dizinleri yeniden yapılandırıldı.

---

## 9. Kaynakça

1. Mustafa, N. E., & Alizadeh, F. (2024). Unmanned aerial vehicle (UAV) images of road vehicles dataset. *ScienceDirect*, 2352340924002336.  
<https://www.sciencedirect.com/science/article/pii/S2352340924002336>
2. Ultralytics YOLOv5 GitHub Reposu: <https://github.com/ultralytics/yolov5>
3. Google Colab: <https://colab.research.google.com/>
4. Pascal VOC Formatı Açıklamaları: <http://host.robots.ox.ac.uk/pascal/VOC/>
5. PyTorch Resmi Sitesi: <https://pytorch.org/>

Github Linki: <https://github.com/Yusuf-Guney/Derin-Ogrenme-ile-Arac-Siniflandirma-Projesi>

Video Linki: <https://www.youtube.com/watch?v=wUR8a0XJOE>