

# Gerçek Zamanlı Nesne Tespiti, Takibi ve Hareket Tahmini için Entegre Derin Öğrenme Yaklaşımı

## BİTİRME ÖDEVİ

AHMAD AJAJ  
Numara: 21155914

MERSİN ÜNİVERSİTESİ  
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

DANIŞMAN  
DOÇ. DR. FARUK SERİN

MERSİN  
GÜZ DÖNEMİ

## ÖZET

# Gerçek Zamanlı Nesne Tespiti, Takibi ve Hareket Tahmini için Entegre Derin Öğrenme Yaklaşımı

Bu çalışma, gerçek zamanlı nesne tespiti, takibi ve hareket tahmini için entegre bir derin öğrenme hattı sunmaktadır. Sistem, YOLO tabanlı bir tespit modeli, centroid tabanlı bir takip algoritması ve LSTM tabanlı bir zaman serisi tahmin modeli içermektedir. Video verileri öncelikle karelere dönüştürülerek, nesneler etiketlenip YOLO modeli ile eğitilmekte ve konum koordinatları elde edilmektedir. Bu koordinatlar daha sonra LSTM modeline aktarılmakta ve gelecekteki konumların tahmini gerçekleştirilmektedir.

Önerilen yapı; spor analitiği, otonom sistemler ve gözetim gibi uygulamalara yönelik esnek ve modüler bir mimari sağlamaktadır.

**Anahtar Kelimeler:** Nesne Tespiti, Takip, Hareket Tahmini, YOLO, LSTM, Derin Öğrenme

**Danışman:** DOÇ. DR. FARUK SERİN, Mersin Üniversitesi, BİLGİSAYAR MÜHENDİSLİĞİ  
BÖLÜMÜ Anabilim Dalı, Mersin.

## İÇİNDEKİLER

	Sayfa
<b>İÇ KAPAK</b>	i
<b>ÖZET</b>	ii
<b>İÇİNDEKİLER</b>	iii
<b>TABLOLAR DİZİNİ</b>	iv
<b>ŞEKİLLER DİZİNİ</b>	v
<b>SİMGELER VE KISALTMALAR</b>	vi
<b>1. GİRİŞ</b>	<b>1</b>
1.1. PROJE ANLATIMI	1
1.2. Hedefler	1
<b>2. YÖNTEM</b>	<b>2</b>
2.1. İş Paketleri	2
2.2. Pipeline	2
2.3. Kullanılan Araçlar ve Tanımları	4
2.4. Riskler ve Önlemler	4
<b>3. Veri Hazırlığı</b>	<b>5</b>
3.1. Giriş Videosunun Karelere Ayrılması	5
3.2. Roboflow ile Etiketleme ve Veri Çıkarımı	6
3.3. Veri Artırma (Augmentation) ve Veri Kümesi Dağılımı	7
3.4. YOLOv8 Formатı ve Klasör Yapısı	9
<b>4. YOLOv8 Eğitimi</b>	<b>10</b>
4.1. Kurulum ve Açıklama	11
4.2. Model Eğitimi ve Çıktılar	11
4.3. Örnek Görseller Üzerinde Test	16
4.4. Koordinatların CSV Dosyasına Aktarılması	17
<b>5. LSTM Eğitimi</b>	<b>18</b>
5.1. Veri yükleme, ön işleme ve normalizasyon.	18
5.2. LSTM girişi için ardışık veri dizilerinin hazırlanması.	19
5.3. Eğitim ve test veri setlerinin ayrılması.	19
5.4. LSTM modelinin tanımlanması ve derlenmesi.	20
5.5. Modelin early stopping ile eğitimi.	21
5.6. Test veri setinde değerlendirme.	21
5.7. Topun gerçek ve tahmin edilen yörüngelerinin karşılaştırmalı görselleştirilmesi.	22
5.8. Hata ve doğruluk metriklerinin hesaplanması ve analizi.	24
<b>6. Uçtan Uca Ürün</b>	<b>25</b>
6.1. Tüm Pipeline'in Birleştirilmesi	25
6.2. Testler ve Nihai Sonuçlar	27
6.3 Backend	30
6.4 Frontend	32
6.5 Docker	35
6.6 Dağıtım	37
<b>KAYNAKLAR</b>	<b>38</b>

## TABLOLAR DİZİNİ

	Sayfa
<b>Tablo 2.1.</b> İş Paketleri ve Çıktıları	2
<b>Tablo 2.2.</b> Proje Teknolojileri	4
<b>Tablo 2.3.</b> Risk Yönetimi	4
<b>Tablo 3.1</b> Video Ön İşleme Komutları	5
<b>Tablo 3.2</b> Veri Artırma (Augmentation) Teknikleri	7
<b>Tablo 4.1</b> YOLOv8 modelinin eğitim parametreleri, değerleri ve belirlenme gerekçeleri	13
<b>Tablo 4.2</b> YOLOv8 modelinin eğitim sürecinde mAP ve geri çağrıma (recall) metriklerinin değişimi	14
<b>Tablo 4.3</b> YOLOv8 modelinin doğrulama kayıplarının (val/box_loss ve val/cls_loss) epoch bazında Değişimi	15
<b>Tablo 6.1</b> YOLOv8 ile Veri Toplama ve Ön İşleme Adımları (PASS 1)	26
<b>Tablo 6.2</b> LSTM ile Yörünge Tahmini Adımları	26
<b>Tablo 6.3</b> Video Görüntüleme Adımları (PASS 2)	27
<b>Tablo 6.4</b> Projede Kullanılan Docker Komutlarının İşlevsel Özeti	36

## ŞEKİLLER DİZİNİ

	Sayfa
<u>Şekil 2.1.</u> Proje Çalışma Akışı Diyagramı	3
<u>Şekil 3.1</u> Roboflow ortamında kırmızı top nesnesinin etiketlenmesi (bounding box oluşturma aşaması)	6
<u>Şekil 3.2</u> Veri kümесinin eğitim, doğrulama ve test olarak bölünmesi ile son veri artırma (augmentation) adımlarının gösterimi	8
<u>Şekil 3.3.</u> Veri kümесinin dışa aktarımında YOLOv8 formatının seçilmesi aşaması	9
<u>Şekil 3.4.</u> İndirilen veri kümесinin dosya ve klasör yapısının gösterimi	10
<u>Şekil 4.1</u> YOLOv8 Eğitim Kayıp Eğrileri ve Yakınsama Analizi	14
<u>Şekil 4.2.</u> YOLOv8 Çıktılarının LSTM İçin Zaman Serisi Formatında CSV Dosyasına Kaydedilmesi	17
<u>Şekil 5.1.</u> Top Yörüngesi: Gerçek vs Tahmin Edilen (Test Seti)	22
<u>Şekil 5.2.</u> Y Koordinatı: Gerçek vs Tahmin Edilen (Test Seti)	23
<u>Şekil 5.3.</u> X Koordinatı: Gerçek vs Tahmin Edilen (Test Seti)	23
<u>Şekil 6.1</u> Top Yörüngesi: Gerçek vs Tahmin	28
<u>Şekil 6.2</u> X ve Y Koordinatları: Gerçek vs Tahmin Edilen	28
<u>Şekil 6.3</u> Top Tespiti ve Tahmini Konum Noktası	29
<u>Şekil 6.4</u> Swagger Üzerinden API Uç Noktaları	30
<u>Şekil 6.5</u> Projenin web arayüzü genel görünümü	32
<u>Şekil 6.6</u> Fotoğraf yükleme seçeneği ile top tespiti arayüzü	33
<u>Şekil 6.7</u> Video işleme sırasında ilerleme göstergesi ve çıktı önizlemesi	34
<u>Şekil 6.8</u> İşlem tamamlandıktan sonra video sonuçları, metrikler, grafikler ve CSV çıktıları ile indirme seçenekler	34
<u>Şekil 6.9</u> Render üzerinde çalışan uygulamanın web arayüzü.	37

## SİMGELER VE KISALTMALAR

Kısaltma/Simge	Tanım
YOLO	You Only Look Once (Nesne Tespit Modeli)
LSTM	Long Short-Term Memory (Uzun Kısa Süreli Bellek)
CNN	Convolutional Neural Network (Evrişimli Sinir Ağısı)
RNN	Recurrent Neural Network (Tekrarlayan Sinir Ağısı)
IP	İş Paketi
CSV	Comma-Separated Values (Virgülle Ayrılmış Değerler)
ML	Machine Learning (Makine Öğrenmesi)
GPU	Graphics Processing Unit (Grafik İşleme Birimi)
FPS	Frames Per Second(Saniyedeki Kare Sayısı)
IoU	Intersection over Union (Kesişim bölgesi / Birleşim bölgesi)
mAP	Mean Average Precision (Ortalama Hassasiyet)
CUDA	Compute Unified Device Architecture (NVIDIA GPU paralel işlem mimarisi)
T4	NVIDIA Tesla T4 (GPU modeli)
W,H	Video Width and Height (Video Genişlik ve Yükseklik)
Patience	Early Stopping Patience (Erken Durdurma Sabır Parametresi)
Augment	Augmentation(Veri Arıtırma)
MAE	Mean Absolute Error (Ortalama Mutlak Hata)
MSE	Mean Squared Error (Ortalama Kare Hata)
API	Application Programming Interface (Uygulama Programlama Arayüzü)

## 1. GİRİŞ

### 1.1 PROJE ANLATIMI

Bu proje, video tabanlı nesne tespiti, takibi ve hareket tahminini tek bir sistem içinde birleştiren entegre bir derin öğrenme çözümü geliştirmeyi amaçlamaktadır. Günümüzde video analitiği, güvenlik, spor teknolojileri ve otonom sistemlerde büyük önem taşımaktadır. Ancak bu sistemlerin büyük çoğunluğu yalnızca tespit veya takip aşamasına odaklanmakta, hareketin gelecekteki yönünü veya hızını tahmin etme yeteneğine sahip olmamaktadır. Bu çalışma, bu eksikliği gidermek üzere YOLOv8 ve LSTM modellerini bir araya getiren uça bir yapay zekâ pipeline’ı önermektedir.

Sistem üç ana aşamadan oluşmaktadır. İlk olarak, video verisi karelere ayrılmakta ve YOLOv8 modeli ile her karedeki nesneler tespit edilmektedir. İkinci aşamada, tespit edilen nesneler centroid tabanlı bir algoritma kullanılarak kareler arasında takip edilmekte ve konum koordinatları kaydedilmektedir. Üçüncü aşamada ise bu koordinatlar, LSTM tabanlı zaman serisi modeliyle işlenerek nesnenin gelecekteki konumu tahmin edilmektedir.

Bu yöntem sayesinde sistem, yalnızca mevcut konumları değil, aynı zamanda hareketin olası yönünü ve hızını da tahmin edebilmektedir. Böylece model, gerçek zamanlı uygulamalarda karar verme süreçlerini destekleyebilecek bir altyapı sunmaktadır. Spor analitiğinde top veya oyuncu hareketlerinin izlenmesi, otonom sistemlerde engellerin öngörülmesi ve güvenlik kameralarında şüpheli hareketlerin önceden tespit edilmesi gibi pek çok senaryoda kullanılabilecek bir çözüm ortaya konmuştur.

Proje, derin öğrenmenin iki güçlü bileşeni olan CNN (YOLOv8) ve RNN (LSTM) mimarilerini birleştirerek zamansal ve mekânsal bilgi çıkarımını tek bir sistem altında toplamaktadır. Bu yaklaşım, hem doğruluk hem de işlem süresi açısından verimlilik sağlamaktadır.

### 1.2 HEDEFLER

1. YOLOv8 modeli ile nesne tespiti gerçekleştirmek.
2. Tespit edilen nesnelerin zamanındaki hareketini takip etmek.
3. LSTM modeliyle nesnelerin gelecekteki konumlarını tahmin etmek.
4. Gerçek zamanlı çalışabilecek, modüler bir pipeline tasarlamak.
5. Çıktıları görselleştirerek tahmin performansını değerlendirmek.

## 2.YÖNTEM

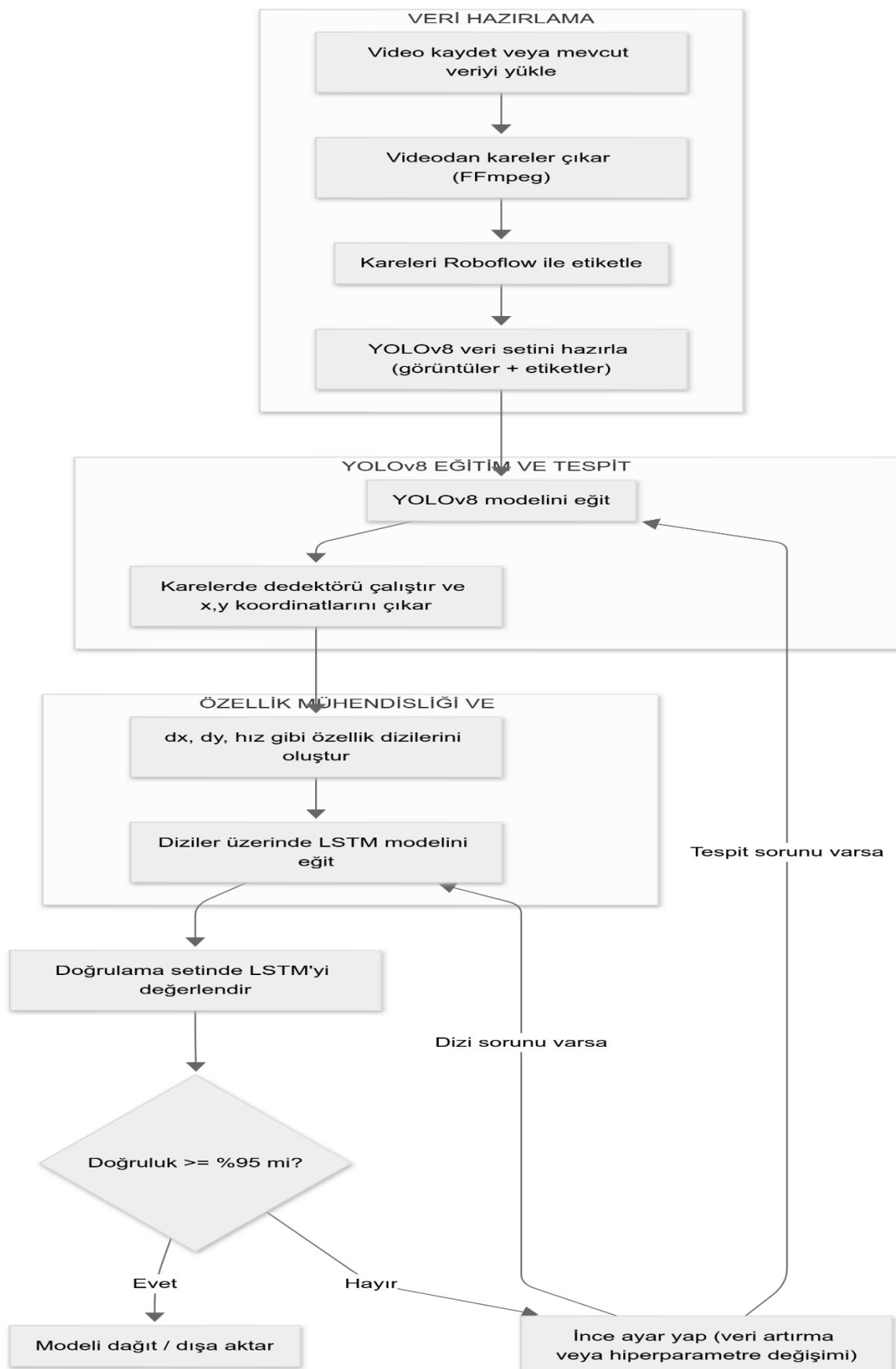
### 2.1. İş Paketleri

İş Paketi No	İş Paketi Adı	Açıklama	Çıktı
İP1	Veri Hazırlığı	Video verisinin toplanması, karelere dönüştürülmesi ve etiketlenmesi	YOLOv8 veri seti
İP2	Nesne Tespitı	YOLOv8 modelinin eğitilmesi ve test edilmesi	Eğitimli YOLOv8 modeli
İP3	Takip ve Koordinat Çıkarımı	Her karede tespit edilen nesnelerin takip edilmesi, x-y koordinatlarının çıkarılması	Koordinat verisi (CSV)
İP4	Hareket Tahmini	LSTM modeliyle zaman serisi tahmini yapılması	Tahmin edilen koordinatlar
İP5	Pipeline Entegrasyonu	Tüm aşamaların uçtan uca birleştirilmesi ve test edilmesi	Entegre sistem çıktısı
İP6	Raporlama ve Sonuçlar	Bulguların değerlendirilmesi, raporun hazırlanması	Nihai proje raporu

Tablo 2.1. İş Paketleri ve Çıktıları

### 2.2. Pipeline (Çalışma Akışı)

Pipeline, üç temel aşamadan oluşmaktadır: veri hazırlığı, nesne tespiti ve hareket tahmini. İlk aşamada video karelere bölünür ve etiketlenir. İkinci aşamada YOLOv8 modeli kullanılarak nesnelerin konumları tespit edilir. Üçüncü aşamada bu konum verileri LSTM modeline aktarılır ve gelecekteki hareket yönleri tahmin edilir. Süreç, model doğruluğu istenen seviyeye ulaşana kadar yinelemeli olarak devam eder.



Şekil 2.1. Proje Çalışma Akışı Diyagramı

### 2.3. Kullanılan Araçlar ve Tanımları

Araç / Kütüphane	Açıklama	Kullanım Nedeni
<b>Python</b>	Ana programlama dili	Geniş kütüphane desteği ve ML uyumluluğu
<b>YOLOv8</b>	Derin öğrenme tabanlı nesne tespit modeli	Hızlı ve doğru gerçek zamanlı tespit
<b>Roboflow</b>	Veri etiketleme ve YOLO formatına dönüştürme aracı	Etiketleme sürecini kolaylaştırmak
<b>FFmpeg</b>	Video işleme aracı	Videolardan kare çıkarmak
<b>OpenCV</b>	Görüntü işleme kütüphanesi	Takip ve koordinat çıkarımı
<b>NumPy / Pandas</b>	Sayısal veri işleme	LSTM için veri ön işleme
<b>TensorFlow / Keras</b>	Derin öğrenme kütüphanesi	LSTM modelinin eğitimi
<b>Google Colab</b>	Bulut tabanlı GPU ortamı	Hızlı model eğitimi ve kolay paylaşım

Tablo 2.2. Proje Teknolojileri

### 2.4. Riskler ve Önlemler

Risk	Açıklama	Önlem
<b>Veri Yetersizliği</b>	Etiketli veri az olduğunda model aşırı öğrenme yapabilir	Veri artırma (augmentation) teknikleri kullanılır
<b>Model Aşırı Öğrenme (Overfitting)</b>	Eğitim verisine fazla uyum, genelleme düşer	Dropout ve erken durdurma (early stopping) uygulanır
<b>Donanım Kısıtlamaları</b>	Eğitim süresi uzun olabilir	GPU ortamında (Colab) çalışma
<b>Koordinat Gürültüsü</b>	Takip hataları sonucu yanlış giriş verisi	Filtreleme (ör. Kalman veya ortalama filtre)
<b>LSTM Tahmin Hatası</b>	Zaman serisi karmaşıklığı nedeniyle düşük doğruluk	Hiperparametre ayarlaması ve model yeniden eğitimi

Tablo 2.3. Risk Yönetimi

### 3. Veri Hazırlığı

Veri hazırlığı aşaması, derin öğrenme modelinin başarısını doğrudan etkileyen en kritik adımdır. Bu projede, video verisinden yüksek kaliteli ve temizlenmiş zaman serisi verisi elde etmek amacıyla sistematik bir süreç izlenmiştir.

#### 3.1 Giriş Videosunun Karelere Ayrılması

Eğitim için toplanan sarkaç hareketlerini içeren videolar, FFmpeg komut satırı aracı kullanılarak sabit bir kare hızında (FPS) görüntülere ayrılmıştır.

VIDEO	Ffmpeg Komutu	Açıklama
red-ball.mp4	<code>ffmpeg -y -i red-ball.mp4 -vf fps=5 frames/red/frame_%04d.jpg</code>	Video, saniyede <b>5 kare (fps=5)</b> hızında jpg formatında dışa aktarılmıştır.
colored-ball.mp4	<code>ffmpeg -y -i colored-ball.mp4 -vf fps=5 frames/colored/frame_%04d.jpg</code>	Aynı işlem uygulanarak toplamda yaklaşık <b>600 adet</b> eğitim karesi elde edilmiştir.

Tablo 3.1 Video Ön İşleme Komutları

Saniyedeki kare sayısının (FPS) 5 olarak belirlenmesi, hem nesnenin hareketini yakalamak için yeterli **zamansal çözünürlük** sağlamış, hem de aşırı yüksek kare sayısı nedeniyle etiketleme ve eğitim süresinin artmasını engellemiştir. Aynı zamanda bu, LSTM modeline giren zaman serisinin birim zaman aralığını sabitlemiştir.

### 3.2 Roboflow ile Etiketleme ve Veri Çıkarımı

Karelere ayrılan ham görüntüler, etiketleme ve veri setini hazırlama sürecini hızlandırmak amacıyla **Roboflow** bulut platformuna yüklenmiştir.

#### Etiketleme Süreci:

- Sınıf Tanımlama:** Projenin tek bir nesneye (sarkaç topu) odaklanması nedeniyle **tek bir sınıf ('ball')** tanımlanmıştır.
- Otomatik Etiketleme:** Görüntülerin bir kısmı manuel olarak etiketlendikten sonra, **Roboflow'un otomatik etiketleme (auto-labeling)** özelliği kullanılarak kalan karelerin etiketlenmesi sağlanmış ve bu sayede süreçteki iş yükü ciddi ölçüde azaltılmıştır.
- Etiketleme Formatı:** Tüm nesneler, her karede kesin sınır kutusu (Bounding Box) ile etiketlenmiştir.



Şekil 3.1 Roboflow ortamında kırmızı top nesnesinin etiketlenmesi (bounding box oluşturma aşaması)

### 3.3 Veri Artırma (Augmentation) ve Veri Kümesi Dağılımı

Modelin **genelleme yeteneğini** artırmak ve veri setinin azlığından kaynaklanabilecek **aşırı öğrenme (overfitting)** riskini azaltmak amacıyla güçlü veri artırma (Augmentation) teknikleri uygulanmıştır.

#### 3.3.1 Uygulanan Artırma Teknikleri ve Etkileri:

Teknik	Ayar (Örn. %20, ±10°)	Etkisi ve Faydası
<b>Görüntü Boyutu Yeniden Boyutlandırma</b>	640x640	YOLOv8 modelinin standart girdi boyutuna sabitlenmesi. Çıkarım hızını ve model optimizasyonunu artırır
<b>Gri Tonlama (Greyscale)</b>	%20'ye kadar	Modelin nesneyi sadece renge bakarak değil, <b>geometrik şekil ve doku</b> özelliklerine göre tanımmasını sağlayarak <b>renk sapmalarına karşı dayanıklılığı</b> artırır.
<b>Doygunluk (Saturation)</b>	±30%	Aydınlatma koşullarındaki doğal değişimleri simüle eder.
<b>Sınır Kutusu Kaydırma (Bounding Box Shear)</b>	Yatay ±10°	Nesnenin kameraya göre haffi açılı görünümünü simüle ederek <b>perspektif değişimlerine karşı modelin robustlığını</b> güçlendirir.
<b>Pozlama (Exposure)</b>	±10%	Aşırı parlak veya karanlık ortamlardaki performansı iyileştirir.
<b>Gürültü (Noise)</b>	%2 (piksel)	Özellikle düşük kaliteli veya sıkıştırılmış video akışlarındaki <b>rastgele sensör gürültüsüne</b> karşı modelin direncini artırır.

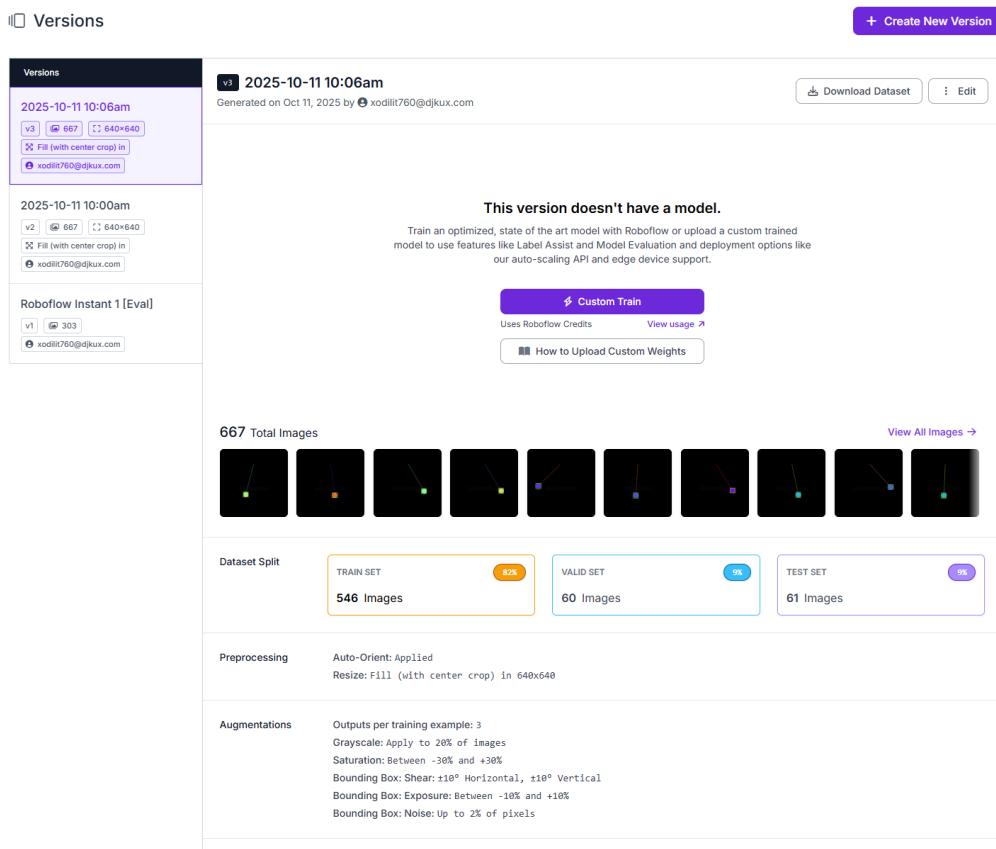
Tablo 3.2 Veri Artırma (Augmentation) Teknikleri

### 3.3.2 Veri Kümesi Dağılımı:

Roboflow'da yapılan ayrıştırma ile veri setinin dağılımı şu şekilde belirlenmiştir:

- **Eğitim (Train):** %82
- **Doğrulama (Valid):** %9
- **Test (Test):** %9

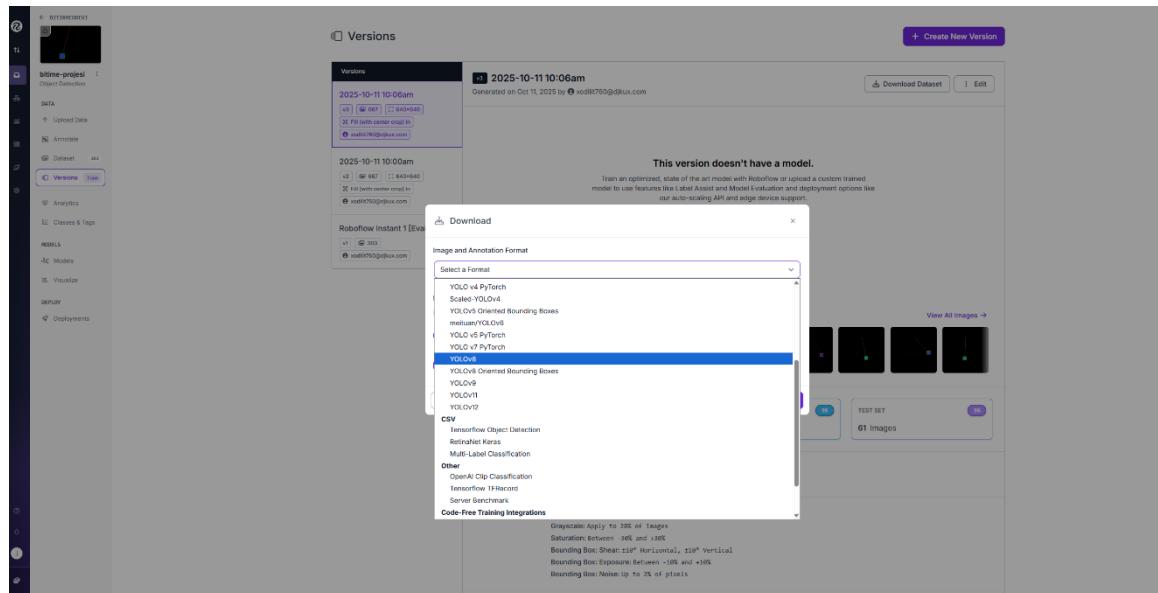
Bu dengeli dağılım, modelin bilinmeyen veriler üzerindeki performansının **objektif** bir şekilde değerlendirilmesi için standart bir mühendislik yaklaşımıdır.



Şekil 3.2 Veri kümelerinin eğitim, doğrulama ve test olarak bölünmesi ile son veri artırma (augmentation) adımlarının gösterimi

### 3.4. YOLOv8 Formatı ve Klasör Yapısı

Veri setine istenilen dönüşümler uygulandıktan sonra, Roboflow veriyi Ultralytics (YOLOv8) kütüphanesinin gerektirdiği formata otomatik olarak dönüştürülmüştür.



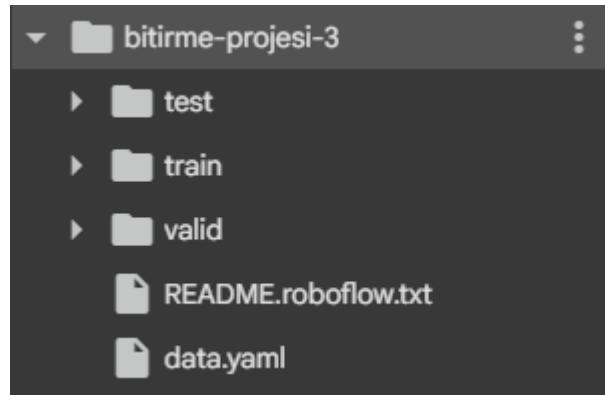
Şekil 3.3. Veri kümesinin dışa aktarımında YOLOv8 formatının seçilmesi aşaması

Kullanılan kod ile veri seti, Colab ortamina ZIP veya doğrudan bağlantı ile indirilerek aşağıdaki temel dosya ve klasör yapısını oluşturmuştur:

```
! pip install roboflow
from roboflow import Roboflow
rf = Roboflow(api_key="SUPER-SECRET-API-KEY")
project = rf.workspace("bitirmeodevi-blusn").project("bitirme-projesi-j58xp")
version = project.version(3)
dataset = version.download("yolov8")
```

OUTPUT:

```
/content/bitirme-projesi-3/
├── train/
│   └── images/ # %89 eğitim görüntüler
│       └── labels/ # %89 eğitim etiketleri (.txt dosyaları)
├── valid/
│   └── images/ # %9 doğrulama görüntüler
│       └── labels/ # %9 doğrulama etiketleri
└── test/
    └── images/ # %9 test görüntüler
        └── labels/ # %9 test etiketleri
            └── data.yaml # YOLOv8 eğitim ayarları dosyası (Kritik)
```



Şekil 3.4. İndirilen veri kümесinin dosya ve klasör yapısının gösterimi

### **data.yaml Dosyasının İçeriği:**

Bu dosya, YOLOv8 eğitimini başlatmak için gerekli olan tüm meta bilgileri içerir ve modelin hangi veriyi nerede bulacağını tanımlar.

- **path**: Veri setinin ana dizin yolu.
- **train**: Eğitim görüntülerinin alt klasör yolu (örneğin, train/images).
- **val**: Doğrulama görüntülerinin alt klasör yolu (örneğin, valid/images).
- **test**: Test görüntülerinin alt klasör yolu (örneğin, test/images).
- **names**: Sınıf adlarının listesi. Bu projede tek bir sınıf (örneğin, names: [ball]) içerir.

Bu yapı, YOLOv8'in **model.train(data=dataset\_path, ...)** komutu ile doğrudan eğitime başlamasına olanak tanıyan **üretim kalitesinde** bir veri seti organizasyonudur.

#### 4. YOLOv8 Eğitimi

Bu bölümde, sarkaç topu nesnesinin video karelerindeki konumunu tespit etmek için kullanılan YOLOv8 modelinin kurulumu, eğitimi ve test edilmesi süreçleri detaylandırılmıştır. YOLOv8, hız ve doğruluk arasındaki optimum dengeyi sağlama nedeniyle gerçek zamanlı uygulamalar için tercih edilmiştir.

##### 4.1 Kurulum ve Açıklama

Proje, hızlı kurulum ve GPU erişimi avantajlarından yararlanmak için **Google Colab** ortamında yürütülmüştür.

```
#YOLOv8 modelinin ana kütüphanesi yüklenmiştir.  
! pip install -- upgrade pip  
! pip install ultralytics  
  
#GPU varlığı kontrol edilmiş ve eğitim için CUDA desteği teyit  
edilmiştir.  
import torch  
print("CUDA available:", torch.cuda.is_available())  
! nvidia-smi
```

##### Model Tercihi: YOLOv8n

YOLOv8 ailesinden YOLOv8n (nano) modeli, önceden eğitilmiş ağırlıklarla (`yolov8n.pt`) seçilmiştir. Nano versiyonu, daha az parametreye sahip olup, daha yüksek çıkışım hızı (FPS) sunar. Bu, özellikle sonraki aşamada her karede koordinat tespiti yapacak olan gerçek zamanlı pipeline için kritik bir mühendislik kararıdır.

## 4.2. Model Eğitimi ve Grafik Çıktıları

Model eğitimi, hazırlanan özelleştirilmiş veri seti üzerinde transfer öğrenimi (**pretrained=True**) kullanılarak ince ayar (fine-tuning) şeklinde gerçekleştirilmiştir.

```
from ultralytics import YOLO

# Yolları Belirle
dataset_path = '/content/bitirme-projesi-3/data.yaml' # dataset.yaml dosya yol

# Önceden Eğitilmiş YOL0v8n (nano) modelini yükley
model = YOLO('yolov8n.pt')

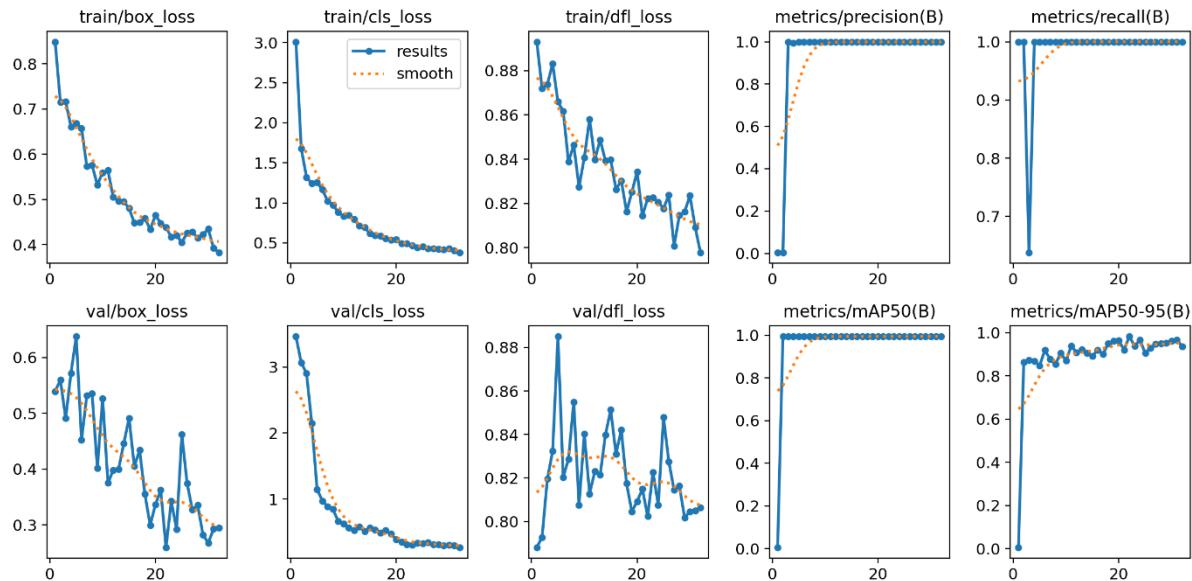
# Aşırı öğrenme önlemleri ve grafik çizimi ile eğitimi başlat
model.train(
    data=dataset_path,                      # dataset.yaml dosya yol
    epochs=50,                             # 50 epoch ile başla
    imgsz=640,                            # Görüntü boyutu
    batch=8,                               # GPU dostu toplu iş boyutu
    device=0,                             # GPU (T4) kullanımı
    name='yolov8-model',                   # Deney adı
    augment=True,                          # Veri artırımları etkinleştir
    patience=10,                           # İyileşme olmazsa erken durdurma için sabır
    (epoch sayısı)
    pretrained=True, # Önceden eğitilmiş ağırlıklarla ince ayar yap
    plots=True   # Eğitim grafiklerini otomatik olarak oluştur
)
```

Parametre	Değer	Açıklama	Gerekçesi
<code>model=YOLO('yolov8n.pt')</code>	<b>yolov8n.pt</b>	Önceden Eğitilmiş Model	Transfer Öğrenimi uygulayarak, modelin genel görüntü özelliklerini öğrenme süresini kısaltmak ve daha az veri ile daha hızlı yakınsama sağlamak.
<code>epochs</code>	<b>50</b>	Eğitim Döngüsü Sayısı	Modelin yeterince öğrenmesini sağlamak için belirlenen üst sınır.
<code>imgsz</code>	<b>640</b>	Görüntü Çözünürlüğü	Roboflow'da hazırlanan veri boyutıyla eşleştirilmiştir; YOLOv8 için standart ve optimize edilmiş çözünürlüktür.
<code>batch</code>	<b>8</b>	Toplu İş Boyutu	Google Colab ortamındaki T4 GPU'nun bellek kapasitesine uygun olarak belirlenmiş, eğitim verimliliğini artıran boyut.
<code>patience</code>	<b>10</b>	Erken Durdurma Sabrı	Aşırı Öğrenmeyi (Overfitting) Önleme Mekanizması. Doğrulama kaybı 10 ardışık epoch boyunca iyileşme göstermezse, eğitimi durdurarak en iyi ağırlıkların korunmasını sağlar.
<code>augment</code>	<b>True</b>	Veri Artırma	Bölüm 3'te tanımlanan gri tonlama, gürültü ve kaydırma gibi artırma tekniklerini eğitim sırasında uygulayarak genelleme yeteneğini artırmak.
<code>plots</code>	<b>True</b>	Grafik Çıktıları	Eğitim sırasında Kayıp (Loss) ve mAP eğrilerini otomatik olarak oluşturarak, modelin öğrenme dinamiklerini (yakınsama hızı ve aşırı öğrenme başlangıcı) görsel olarak analiz edebilmek.

Tablo 4.1 YOLOv8 modelinin eğitim parametreleri, değerleri ve belirlenme gerekçeleri

## Eğitim Performans Grafikleri (plots=True Çıktıları)

**plots=True** parametresi sayesinde, modelin öğrenme dinamğini gösteren temel metrikler otomatik olarak üretilmiştir. Bu grafikler, modelin eğitim verisini ne kadar iyi öğrendiğini ve aşırı öğrenme eğilimi gösterip göstermediğini anlamak için hayatı önehme sahiptir.



Şekil 4.1 YOLOv8 Eğitim Kayıp Eğrileri ve Yakınsama Analizi

Metrik	Epoch 1	Epoch 32	Analiz
metrics/mAP50(B)	0.00646	0.995	mAP50, sadece 32 epoch'ta maksimum değeri olan 0.995'e ulaşmıştır. Bu, modelin topu %50 IoU eşığında neredeyse kusursuz bir şekilde tespit ettiğini gösterir.
metrics/mAP50-95(B)	0.00570	0.93661	Daha zorlu ve geniş aralıktaki mAP değeri, 0.93'ün üzerinde bir değere ulaşarak, sınır kutusunun hem doğru konumda hem de yüksek hassasiyette olduğunu kanıtlar.
metrics/recall(B)	1.0	1.0	Model, 1. epoch'tan itibaren eğitim verisindeki tüm nesneleri ('ball') başarıyla geri çağrılmıştır (tam Geri Çağırma). Bu, topun tespit edilmeme (False Negative) oranının sıfır olduğunu gösterir.

Tablo 4.2 YOLOv8 modelinin eğitim sürecinde mAP ve geri çağrıma (recall) metriklerinin değişimi

Metrik	Epoch 1	En İyi Epoch (Min Val Loss)	Son Epoch (32)	Analiz
val/box_loss	0.53948	0.25995 (Epoch 22)	0.29464	Sınır kutusu doğrulama kaybı 22. epoch'ta en düşük seviyeye (0.25995) inmiştir. Bu noktadan sonra kayıp tekrar yükselmeye başlamıştır, bu da aşırı öğrenmenin başladığının sinyalidir.
val/cls_loss	3.46372	0.30208 (Epoch 29)	0.27074	Sınıflandırma kaybı da dramatik bir şekilde düşmüştür.

Tablo 4.3 YOLOv8 modelinin doğrulama kayıplarının (val/box\_loss ve val/cls\_loss) epoch bazında değişimi

Yorum (Erken Durdurma Etkisi):

Model, 22. epoch civarında val/box\_loss değerinde en iyi performansı göstermiş, ardından kayıp artma eğilimine girmiştir. Uygulanan patience=10 mekanızması, eğer eğitim 50. epoch'a ulaşsaydı modelin performansının düşmesini engellemek için kritik rol oynayacaktı. Bu loglar, modelin 20 ila 30. epoch aralığında optimal öğrenme noktasına ulaştığını kesin olarak göstermektedir.

Eğitim sonunda en iyi performans gösteren ağırlıklar, sonraki LSTM entegrasyonu için </content/runs/detect/yolov8-model/weights/best.pt> yoluna kaydedilmiştir.

## Grafik Çıktıları Analizi (plots=True Sonuçları)

Oluşturulan eğitim grafikleri (Kayıp ve mAP eğrileri), sayısal verilerle tam bir uyum sergilemektedir:

- Kayıp Eğrileri :** train/box\_loss eğrisi sürekli düşerken, val/box\_loss eğrisi bir noktadan sonra düzleşmiş ve hafifçe yükselmiştir. Bu **kesesme noktası, Erken Durdurma** mekanizmasının doğru çalıştığını ve en iyi genelleme yeteneğine sahip ağırlıkların (genellikle Epoch 22) kaydedildiğini gösterir.
- mAP Eğrileri :** Özellikle metrics/mAP50(B) eğrisi, ilk birkaç epoch içinde dik bir şekilde yükselerek hızlı öğrenmeyi göstermiş ve kısa sürede **%99.5** değerinde düz bir çizgiye ulaşarak modelin kalitesini kanıtlamıştır.

Bu bulgular, **YOLOv8n** modelinin sarkaç topu tespiti görevinde yüksek doğruluk ve güvenilirlik seviyesine ulaştığını ve bir sonraki aşama olan LSTM'e girdi sağlayacak **temiz ve güvenilir koordinat verisi** üretebileceğini doğrulamaktadır.

#### 4.3. Örnek GörSELLER Üzerinde Test

Eğitilen modelin, hiç görmediği **test veri seti** üzerindeki genellemeye yeteneğini ölçmek amacıyla doğrulama (val) işlemi gerçekleştirilmiştir.

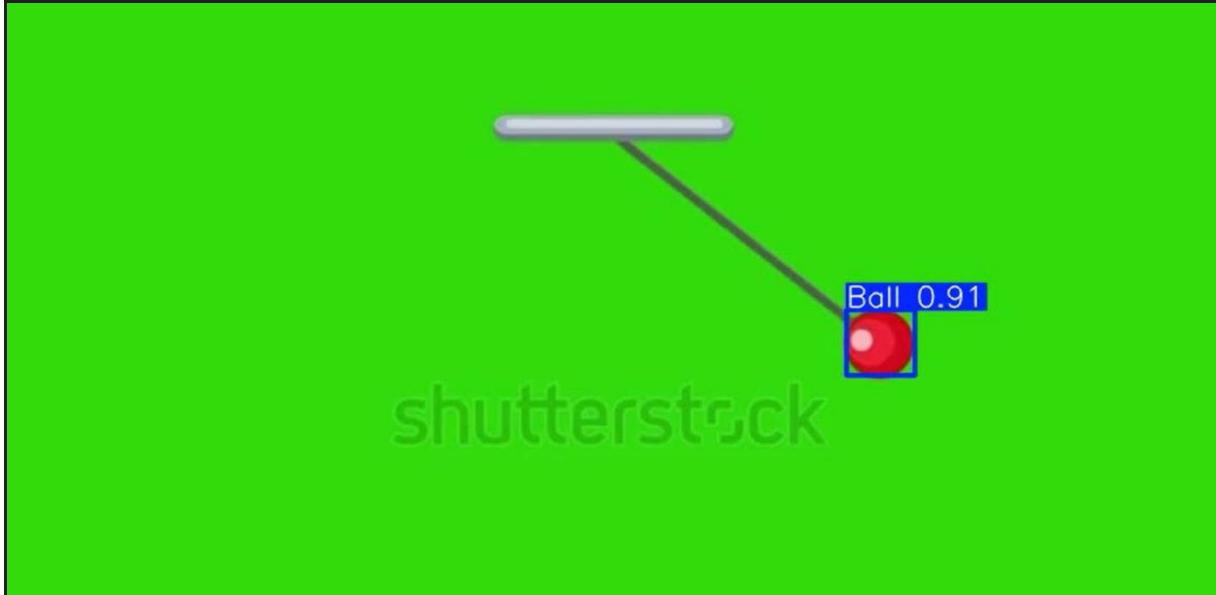
```
# Test Veri Seti Üzerinde Doğrulama (Validation)
model = YOLO('/content/runs/detect/yolov8-model2/weights/best.pt')
results = model.val(data='/content/bitime-projesi-3/data.yaml',
split='test')
```

OUTPUT:

```
⚡ Ultralytics 8.3.209 🚀 Python-3.12.11 torch-2.8.0+cu126 CUDA:0 (Tesla T4, 15095MiB)
Model summary (fused): 72 layers, 3,005,843 parameters, 0 gradients, 8.1 GFLOPs
val: Fast image access ✅ (ping: 0.0±0.0 ms, read: 466.1±146.4 MB/s, size: 9.7 KB)
val: Scanning /content/bitime-projesi-3/test/labels... 61 images, 0 backgrounds, 0 corrupt: 100% ━━━━━━━━ 61/61 2.4Kit/s 0.0s
val: New cache created: /content/bitime-projesi-3/test/labels.cache
      Class   Images Instances   Box(P     R     mAP50   mAP50-95): 100% ━━━━━━━━ 4/4 2.0it/s 2.0s
          all       61        61   0.999    1   0.995   0.976
Speed: 6.3ms preprocess, 12.9ms inference, 0.0ms loss, 2.4ms postprocess per image
Results saved to /content/runs/detect/val
```

Ardından, bir video dosyası üzerinde **çıkarım (inference)** yapılarak modelin dinamik bir ortamda gerçek zamanlı performansı gözlemlenmiştir.

```
# Örnek Video Üzerinde Çıkarım (Inference)
results = model.predict(
    source='/content/bitirme-projesi-3/red-ball.mp4',
    save=True,
    conf=0.25, # Düşük güven eşiği, nesneyi kaybetme riskini azaltır
    device=0    # GPU kullanımı
)
OUTPUT:
```



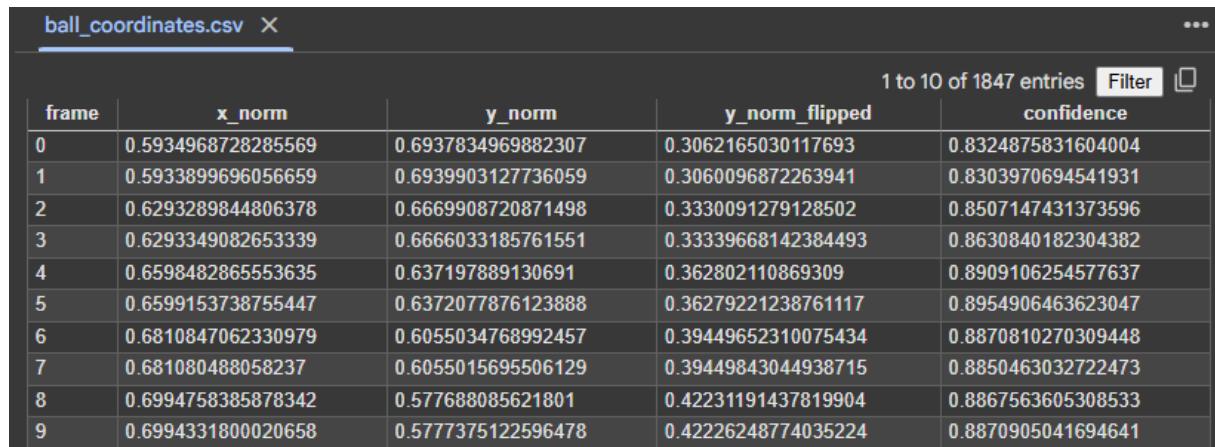
Bu adımın çıktısı, modelin tespit ettiği sınır kutularının ve güvenilirlik skorlarının görülebildiği etiketlenmiş bir video olmuştur.

#### 4.4. Koordinatların CSV Dosyasına Aktarılması (LSTM Veri Mühendisliği)

Bu adım, CNN (YOLOv8) ve RNN (LSTM) arasında köprü görevi görerek, tespit edilen sınır kutusu koordinatlarını LSTM için uygun bir **zaman serisi formatına** dönüştürür.

##### Veri Dönüşüm ve Normalizasyon:

- Merkez Koordinat Hesaplama:** Tespit edilen sınır kutusunun merkez noktası ( $x$  center,  $y$  center) piksel cinsinden hesaplanır.
- Normalize Etme:** Koordinatlar, video boyutlarına ( $W, H$ ) bölünerek  $[0,1]$  aralığına normalize edilir:  $x$  norm =  $x$  center/ $W$ ,  $y$  norm =  $y$  center/ $H$ .
- Y-Eksen Çevirme (Flipping):** Modelin kinematik hareketleri daha iyi öğrenmesi amacıyla, dikey koordinat ters çevrilir.  $y$  norm\_flipped =  $1 - y$  norm formülü,  $y$ -eksenini alttan yukarıya sayan (fiziksel) bir koordinat sistemine dönüştürür.
- CSV Kaydı:** Her kare için elde edilen verileri [ $frame\_idx, x_{norm}, y_{norm\_flipped}, confidence$ ], pandas kütüphanesi kullanılarak **zaman serisi verisi** olarak CSV dosyasına kaydedilmiştir.



frame	x_norm	y_norm	y_norm_flipped	confidence
0	0.5934968728285569	0.6937834969882307	0.3062165030117693	0.8324875831604004
1	0.5933899696056659	0.6939903127736059	0.3060096872263941	0.8303970694541931
2	0.6293289844806378	0.6669908720871498	0.3330091279128502	0.8507147431373596
3	0.6293349082653339	0.6666033185761551	0.33339668142384493	0.8630840182304382
4	0.6598482865553635	0.637197889130691	0.362802110869309	0.8909106254577637
5	0.6599153738755447	0.6372077876123888	0.36279221238761117	0.8954906463623047
6	0.6810847062330979	0.6055034768992457	0.39449652310075434	0.8870810270309448
7	0.681080488058237	0.6055015695506129	0.39449843044938715	0.8850463032722473
8	0.6994758385878342	0.577688085621801	0.42231191437819904	0.8867563605308533
9	0.6994331800020658	0.5777375122596478	0.42226248774035224	0.8870905041694641

Şekil 4.2. YOLOv8 Çıktılarının LSTM İçin Zaman Serisi Formatında CSV Dosyasına Kaydedilmesi

Bu düzenlenmiş ve temizlenmiş CSV dosyası, bir sonraki bölümde (Bölüm 5) LSTM modelinin eğitim girdisi olarak kullanılmaya准备dir.

## 5. LSTM Eğitimi

Bu bölümde, YOLOv8 tarafından tespit edilen top koordinatları kullanılarak topun yörungesini tahmin eden bir LSTM (Uzun Kısa Süreli Hafiza) modeli geliştirilmiştir ve eğitilmiştir. Model, geçmiş pozisyon verilerini kullanarak gelecekteki top konumlarını birkaç kare önceden tahmin etmeye yöneliktedir. Bu sayede topun hareket dinamikleri öğrenilerek tahmin doğruluğu artırılmıştır.

### Adımlar:

1. Veri yükleme, ön işleme ve normalizasyon.
2. LSTM girişi için ardışık veri dizilerinin hazırlanması.
3. Eğitim ve test veri setlerinin ayrılması.
4. LSTM modelinin tanımlanması ve derlenmesi.
5. Modelin early stopping ile eğitimi.
6. Test veri setinde değerlendirme.
7. Topun gerçek ve tahmin edilen yörüngelerinin karşılaştırılmalı görselleştirilmesi.
8. Hata ve doğruluk metriklerinin hesaplanması ve analizi.

### 5.1. Veri Yükleme ve Ön İşleme

CSV dosyasındaki normalize edilmiş top koordinatları (x\_norm ve y\_norm\_flipped) yüklenir:

```
csv_path = "/content/ball_coordinates.csv"
df = pd.read_csv(csv_path)
coords = df[['x_norm', 'y_norm_flipped']].values
```

y\_norm\_flipped: Fiziksel koordinat sisteminde 0 = alt, 1 = üst olacak şekilde y koordinatını ters çevirir.

Koordinatlar 0-1 aralığında normalize edilir, bu sinir ağı eğitimi için önemlidir:

```
scaler = MinMaxScaler()
coords_scaled = scaler.fit_transform(coords)
```

## 5.2. LSTM girişi için ardışık veri dizilerinin hazırlanması.

- SEQ\_LENGTH = 10: Her giriş örneği önceki 10 kareyi içerir.
- PRED\_AHEAD = 5: 5 kare ileri tahmin yapılır.

```
def create_sequences(data, seq_length=SEQ_LENGTH,
pred_ahead=PRED_AHEAD):
    X, y = [], [] # Giriş (X) ve hedef (y) dizilerini başlat
    for i in range(len(data) - seq_length - pred_ahead):
        # Tüm veri boyunca kaydılmalı pencere oluştur
        X.append(data[i:i+seq_length])
        # Giriş dizisine ardışık seq_length kareleri ekle
        y.append(data[i+seq_length+pred_ahead-1])
    # Hedef dizisine pred_ahead sonrası kareyi ekle
    return np.array(X), np.array(y)
# Numpy dizisi olarak döndür

X, y = create_sequences(coords_scaled)
# Ölçeklendirilmiş koordinatlardan ardışık diziler oluştur
```

## 5.3. Eğitim ve test veri setlerinin ayrılması.

Veri seti %80 eğitim, %20 test olarak ayrılır.

Zaman sıralaması korunur, karıştırılmaz (`shuffle=False`):

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, shuffle=False
)
```

Bu, modelin gelecekteki kareleri tahmin etme yeteneğini gerçekçi bir şekilde test etmemizi sağlar.

#### 5.4. LSTM modelinin tanımlanması ve derlenmesi.

İki katmanlı LSTM modeli tanımlanır, dropout ile aşırı öğrenme önlenir:

```
from tensorflow.keras.models import Sequential
# Sıralı model oluşturmak için gerekli sınıfı içe aktar
from tensorflow.keras.layers import LSTM, Dense, Dropout
# LSTM, yoğun katman ve dropout katmanlarını içe aktar

model = Sequential([
    LSTM(64, input_shape=(SEQ_LENGTH, 2), return_sequences=True),
    # 64 birimli LSTM, giriş şekli (SEQ_LENGTH, 2), tüm zaman adımlarını
    döndür
    Dropout(0.3), # Aşırı öğrenmeyi önlemek için %30 dropout uygula
    LSTM(64, return_sequences=False),
    # 64 birimli ikinci LSTM, sadece son zaman adımını döndür
    Dropout(0.3), # Tekrar %30 dropout uygula
    Dense(32, activation='relu'),
    # 32 birimli yoğun katman, ReLU aktivasyonu ile ara temsiller üretir
    Dense(2, activation='linear')
    # 2 birimli çıkış katmanı, [x, y] koordinatlarını tahmin eder
])

model.compile(
    optimizer='adam',
    # Adam optimizasyon algoritması ile ağırlıkları güncelle
    loss='mse',
    # Ortalama Kare Hata (MSE) kayıp fonksiyonu, büyük hataları cezalandırır
    metrics=['mae']
)
```

### 5.5. Modelin early stopping ile eğitimi

```
from tensorflow.keras.callbacks import EarlyStopping
# EarlyStopping callback sınıfını içe aktar

early_stop = EarlyStopping(
    monitor='val_loss',          # Doğrulama kaybını izler
    patience=10,                 # 10 epoch boyunca iyileşme yoksa eğitimi durdur
    restore_best_weights=True # En iyi model ağırlıklarını geri yükle
)

history = model.fit(
    X_train, y_train,           # Eğitim verisi ve hedef değerleri
    epochs=50,                  # Maksimum 50 epoch boyunca eğitim
    batch_size=16,              # Her güncellemede 16 örnek kullan
    validation_split=0.2,        # Eğitim verisinin %20'sini doğrulama için
    ayır
    callbacks=[early_stop],     # EarlyStopping callback ile aşırı
    öğrenmeyi önle
    verbose=2                  # Eğitim sürecini detaylı şekilde göster
)

model.save("trained_lstm_ball.keras") # Eğitilen modeli .keras
formatında kaydet
```

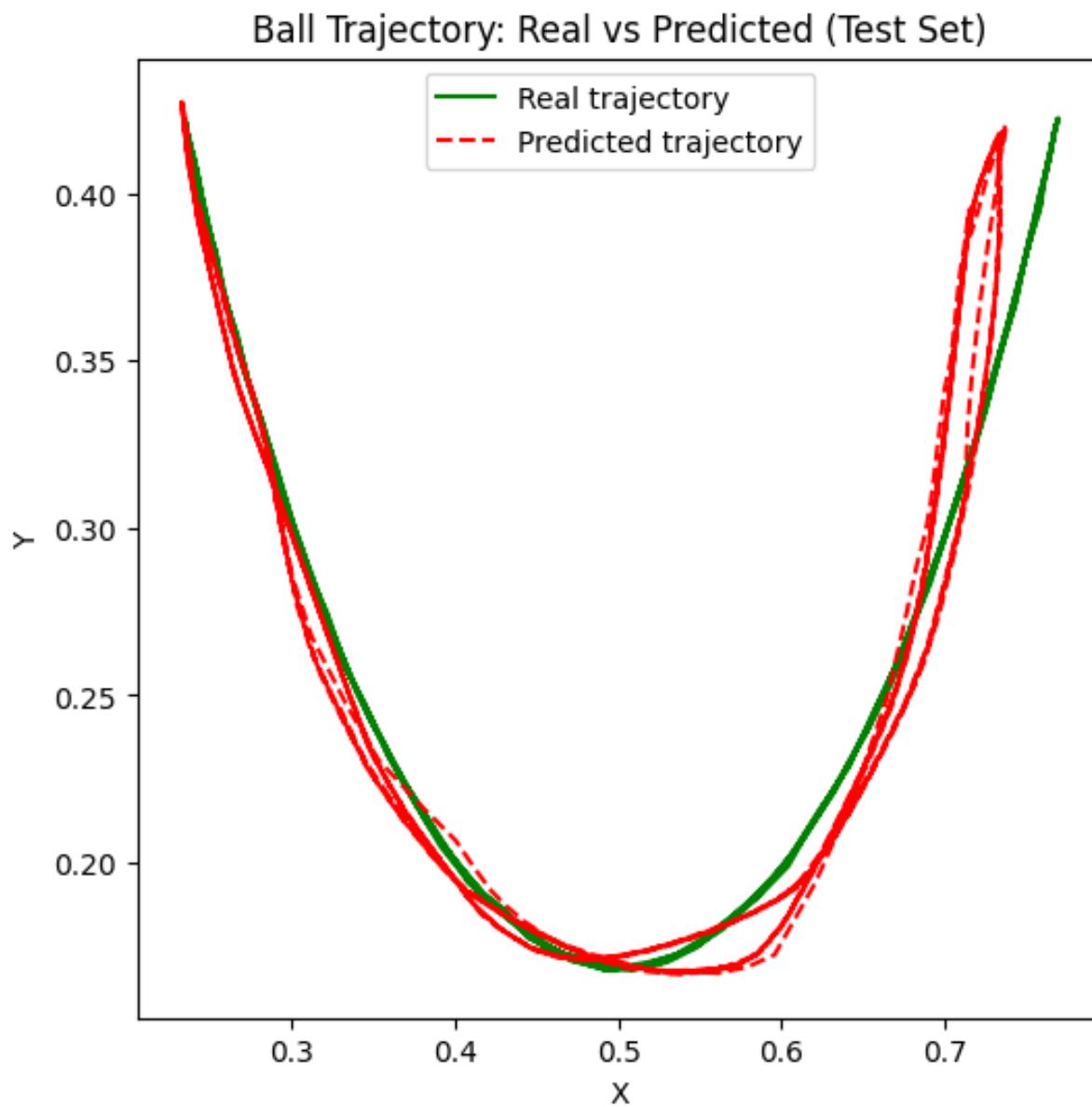
### 5.6. Test veri setinde değerlendirme.

Model test setinde tahmin yapar ve koordinatlar orijinal ölçüye geri dönüştürülür:

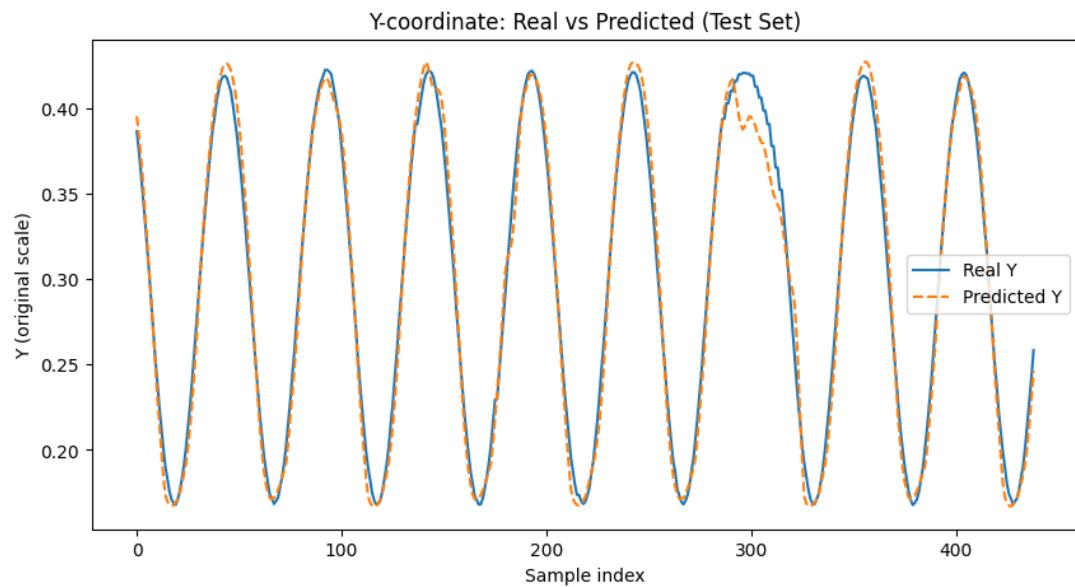
```
y_pred = model.predict(X_test) # Test verisi Üzerinde tahmin yap

y_test_orig = scaler.inverse_transform(y_test)    # Test verisini
orijinal ölçüye geri dönüştür
y_pred_orig = scaler.inverse_transform(y_pred)    # Tahmin edilen veriyi
orijinal ölçüye geri dönüştür
```

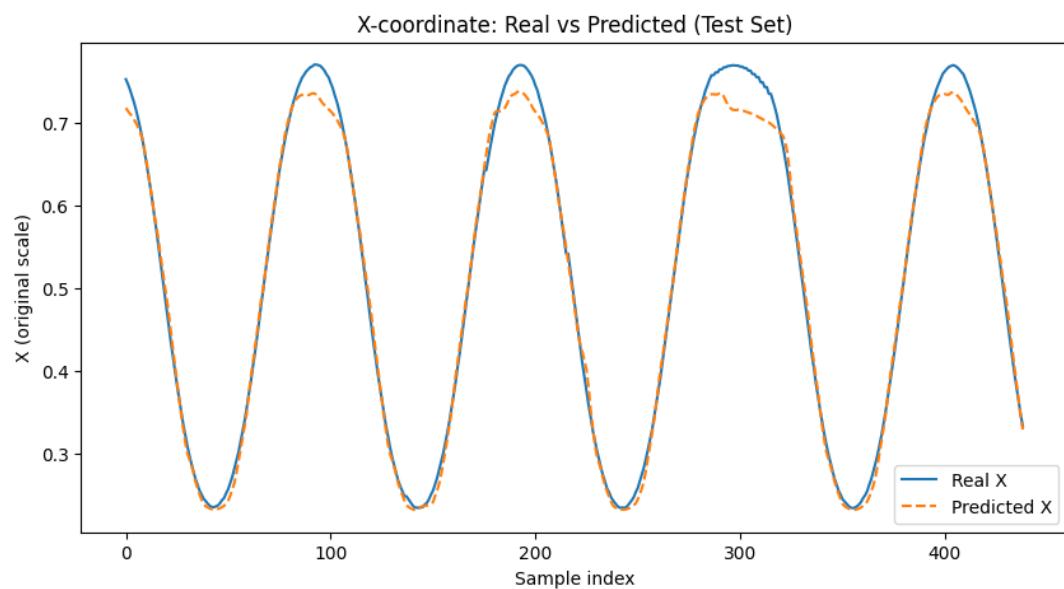
### 5.7. Topun Gerçek ve Tahmin Edilen Yörüngelerinin Karşılaştırmalı Görselleştirilmesi



Şekil 5.1 Top Yörüngesi: Gerçek vs Tahmin Edilen (Test Seti)



Şekil 5.2 Y Koordinatı: Gerçek vs Tahmin Edilen (Test Seti)



Şekil 5.3 X Koordinatı: Gerçek vs Tahmin Edilen (Test Seti)

### 5.8. Hata ve doğruluk metriklerinin hesaplanması.

```
from sklearn.metrics import mean_absolute_error, mean_squared_error
# MAE ve MSE hesaplamak için gerekli fonksiyonları içe aktar

mae_total = mean_absolute_error(y_test_orig, y_pred_orig)
# Test ve tahmin verisi arasındaki toplam Ortalama Mutlak Hata (MAE)
hesapla
mse_total = mean_squared_error(y_test_orig, y_pred_orig)
# Test ve tahmin verisi arasındaki toplam Ortalama Kare Hata (MSE)
hesapla
accuracy_pct_total = (1 - mae_total) * 100
# Yaklaşık doğruluk yüzdesini MAE üzerinden hesapla

mae_x = mean_absolute_error(y_test_orig[:,0], y_pred_orig[:,0])
# X koordinatı için MAE hesapla
mae_y = mean_absolute_error(y_test_orig[:,1], y_pred_orig[:,1])
# Y koordinatı için MAE hesapla
accuracy_x = (1 - mae_x) * 100
# X koordinatı doğruluk yüzdesi
accuracy_y = (1 - mae_y) * 100
# Y koordinatı doğruluk yüzdesi
OUTPUT:
Total MAE: 0.0105, Total MSE: 0.0002, Overall Accuracy: 98.95%
X-axis MAE: 0.0126, Accuracy: 98.74%
Y-axis MAE: 0.0083, Accuracy: 99.17%
```

#### Yorum:

- Total MAE (0.0105) ve Total MSE (0.0002) değerleri, modelin genel olarak tahminlerde yüksek doğruluk sağladığını gösterir.
- MSE çok düşük olduğu için büyük hatalar nadirdir; MAE de düşük olduğu için ortalama tahmin sapması küçüktür.
- Overall Accuracy %98.95, modelin çoğu tahmininin gerçek değerler ile oldukça uyumlu olduğunu ifade eder.
- X-axis MAE (0.0126) ve Accuracy %98.74, X koordinatında tahminlerin biraz daha sapma gösterdiğini belirtir.
- Y-axis MAE (0.0083) ve Accuracy %99.17, Y koordinatında tahminlerin X eksenine göre daha hassas olduğunu gösterir.
- Genel olarak, model hem X hem Y koordinatlarında yüksek doğruluk sağlamaktadır.
- Y koordinatındaki daha yüksek doğruluk, topun dikey hareketinin yatay harekete göre daha öngörülebilir olduğunu gösterebilir.

Bu sonuçlar, LSTM modelinin sarkaç topunun hareketini doğru şekilde öğrenip tahmin ettiğini doğrular.

## 6. Uçtan Uca Ürün

Bu bölümde, projenin temel bileşenleri olan YOLOv8 tabanlı top tespiti ve eğitilmiş LSTM tabanlı yörunge tahmini, uçtan uca bir sistemde birleştirilmiştir. Geliştirilen bu entegre çözüm, bir video akışı üzerinden topun mevcut konumunu gerçek zamanlıya yakın bir hassasiyetle tespit etme ve gelecekteki konumunu tahmin etme yeteneğine sahiptir.

### 6.1. Tüm Pipeline'ın Birleştirilmesi

Pipeline Akışı:

#### 6.1.1. Gerekli Bileşenlerin Yüklenmesi:

Sistem, önceden eğitilmiş YOLOv8 ve LSTM modellerini belleğe yükleyerek işe başlar.

```
import cv2 # Video okuma, işleme ve sonuçları görselleştirme (OpenCV) için.
import pandas as pd # Veri çerçeveleri (CSV) ile çalışmak ve tespit edilen koordinatları kaydetmek için.
import numpy as np # Vektör ve matris işlemleri (özellikle LSTM giriş verileri) için.
from ultralytics import YOLO # Nesne tespiti (top) için eğitilmiş YOLOv8 modelini kullanmak amacıyla.
from tensorflow.keras.models import load_model # Eğitilmiş LSTM yörunge tahmin modelini diskten yüklemek için.
from sklearn.preprocessing import MinMaxScaler # Koordinatları normalize etmek ve tahmin sonuçlarını geri dönüştürmek için.
import os # Dosya yolu işlemleri ve sistem etkileşimleri için.

# Yükleme yolları (Bu değişkenler kodun üst kısımlarında tanımlanmıştır.)
# yolo_weights = "/content/best.pt"
# lstm_model_path = "/content/trained_lstm_ball.keras"

try:
    # YOLOv8 ağırlıkları kullanılarak tespit modelini yükle
    yolo_model = YOLO(yolo_weights)

    # Keras formatında kaydedilmiş LSTM tahmin modelini yükle
    lstm_model = load_model(lstm_model_path)

except Exception as e:
    # Model yüklenemezse hatayı göster ve programı sonlandır
    print(f"Hata: Modeller yüklenemedi: {e}")
    exit()
```

### 6.1.2. Veri Toplama ve Ön İşleme (PASS 1)

Adım	Açıklama	Kod Parçası
<b>Tespit</b>	YOLOv8 ile mevcut karede top tespiti yapılır.	<pre>results = yolo_model.predict(frame, verbose=False, conf=CONF_THRESHOLD)</pre>
<b>Koordinat Hesaplama</b>	Tespit edilen sınırlayıcı kutudan ( $x_1$ , $y_1$ , $x_2$ , $y_2$ ), topun merkez koordinatları hesaplanır.	<pre>x_center = (x1 + x2) / 2</pre>
<b>Normalizasyon</b>	Merkez koordinatları video boyutlarına göre 0-1 aralığında normalize edilir.	<pre>x_norm = x_center / frame_width</pre>
<b>Eksen Ters Çevirme</b>	Fiziksel koordinat sistemi için y ekseni ters çevrilir (0=alt, 1=üst).	<pre>y_norm_flipped = 1 - y_norm</pre>
<b>Veri Kaydı</b>	Normalize edilmiş, ters çevrilmiş koordinat verisi LSTM dizisi için toplanır.	<pre>coords_list.append([x_norm, y_norm_flipped])</pre>

Tablo 6.1 YOLOv8 ile Veri Toplama ve Ön İşleme Adımları (PASS 1)

### 6.1.3. LSTM ile Yörünge Tahmini

Adım	Açıklama	Kod Parçası
<b>Ölçekleyiciyi Uydurma</b>	Toplanan normalize edilmiş verilere ölçekleyici uydurulur (Eğitimdeki MinMaxScaler ile tutarlılık).	<pre>scaler = MinMaxScaler() coords_scaled= scaler.fit_transform(coords_scaled)</pre>
<b>Dizi Oluşturma &amp; Tahmin</b>	\$SEQ\_LENGTH\$ (10 kare) uzunlığundaki kayar penceler oluşturularak LSTM modeline verilir ve tahmin alınır.	<pre>seq = coords_scaled[i:i+SEQ_LENGTH].reshape(1, SEQ_LENGTH, 2) pred = lstm_model.predict(seq, verbose=0)</pre>
<b>Ters Dönüşümre</b>	Tahmin edilen ölçekli çıktı, orijinal 0-1 aralığındaki normalize koordinatlara geri dönürülür.	<pre>predicted_points = scaler.inverse_transform(predicted_points_scaled)</pre>

Tablo 6.2 LSTM ile Yörünge Tahmini Adımları

#### 6.1.4. Video İle Görselleştirme (PASS 2)

Adım	Açıklama	Kod Parçası
<b>Tespit Çizimi</b>	Mevcut kare için kaydedilen sınırlayıcı kutu çizilir (Yeşil Kutu).	<code>cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)</code>
<b>Tahmin Konumunu Hesaplama</b>	LSTM'den gelen normalize edilmiş tahmin Open CV'ye uygun piksel koordinatına çevrilir.	<code>x_pred = int(predicted_x_norm * frame_width)</code>  <code>y_pred = int((1 - predicted_y_flipped) * frame_height)</code>
<b>Tahmin Çizimi</b>	Hesaplanan piksel koordinatına gelecek tahminini belirten nokta çizilir (Turuncu Daire).	<code>cv2.circle(frame, (x_pred, y_pred), 8, (0, 165, 255), -1)</code>
<b>Kare Kaydı</b>	Üzerine çizim yapılmış kare video dosyasına yazılır.	<code>out.write(frame)</code>

Tablo 6.3 Video Görselleştirme Adımları (PASS 2)

#### 6.2. Testler ve Nihai Sonuçlar

##### Tahmin Metrikleri:

Toplam MAE: 0.0434, Toplam MSE: 0.0033, Genel Doğruluk: 95.66%

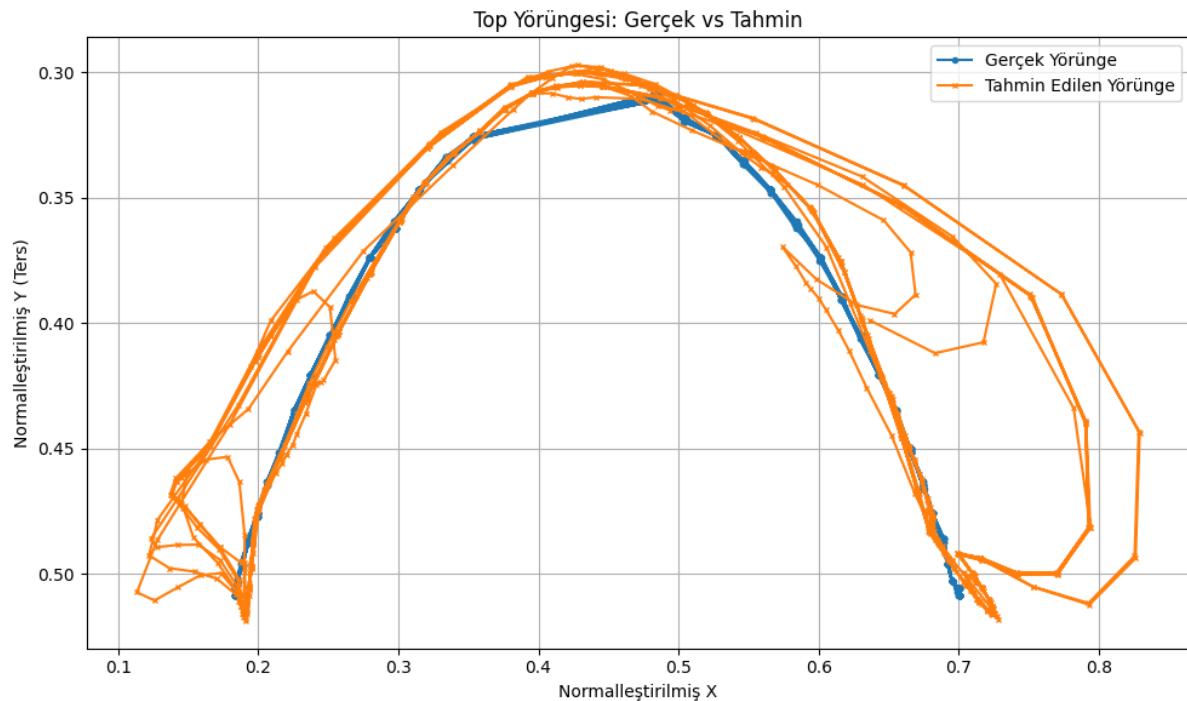
X ekseni MAE: 0.0514, Doğruluk: 94.86%

Y ekseni MAE: 0.0354, Doğruluk: 96.46%

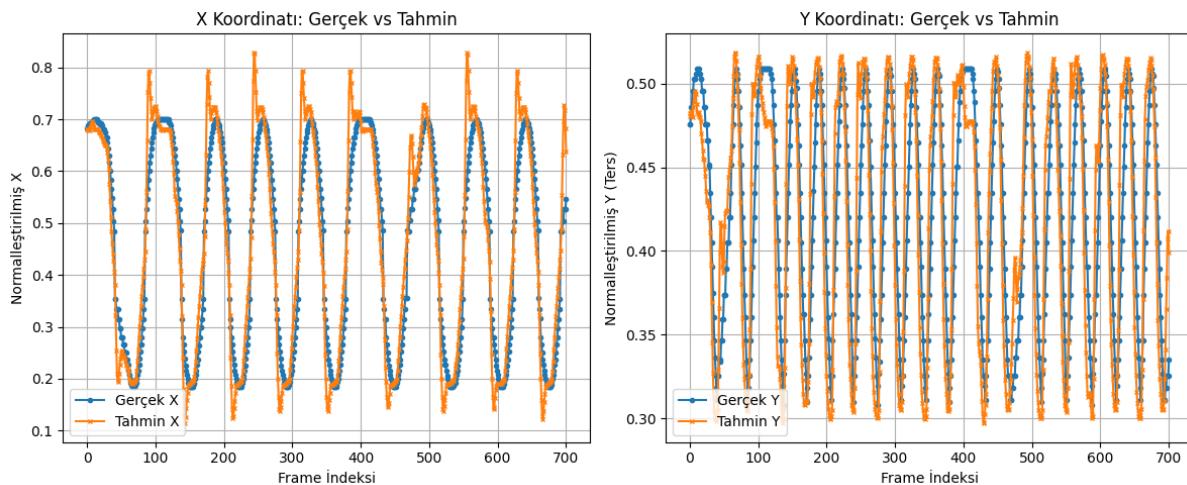
##### Yorum:

- Total MAE (0.0434) ve Total MSE (0.0033), modelin yüksek doğruluk sağladığını gösterir.
- Overall Accuracy %95.66, tahminlerin gerçek değerlerle uyumlu olduğunu gösterir.
- X-axis MAE (0.0514) ve Accuracy %94.86, X koordinatında biraz sapma var.
- Y-axis MAE (0.0354) ve Accuracy %96.46, Y koordinatında tahminler daha hassas.
- Model hem X hem Y'de yüksek doğruluk sağlıyor; dikey hareket yataydan daha öngörülebilir.

Bu sonuçlar, LSTM modelinin sarkaç topunun hareketini doğru şekilde öğrenip tahmin ettiğini doğrular.



Şekil 6.1 Top Yörüngesi: Gerçek vs Tahmin



Şekil 6.2 X ve Y Koordinatları: Gerçek vs Tahmin Edilen



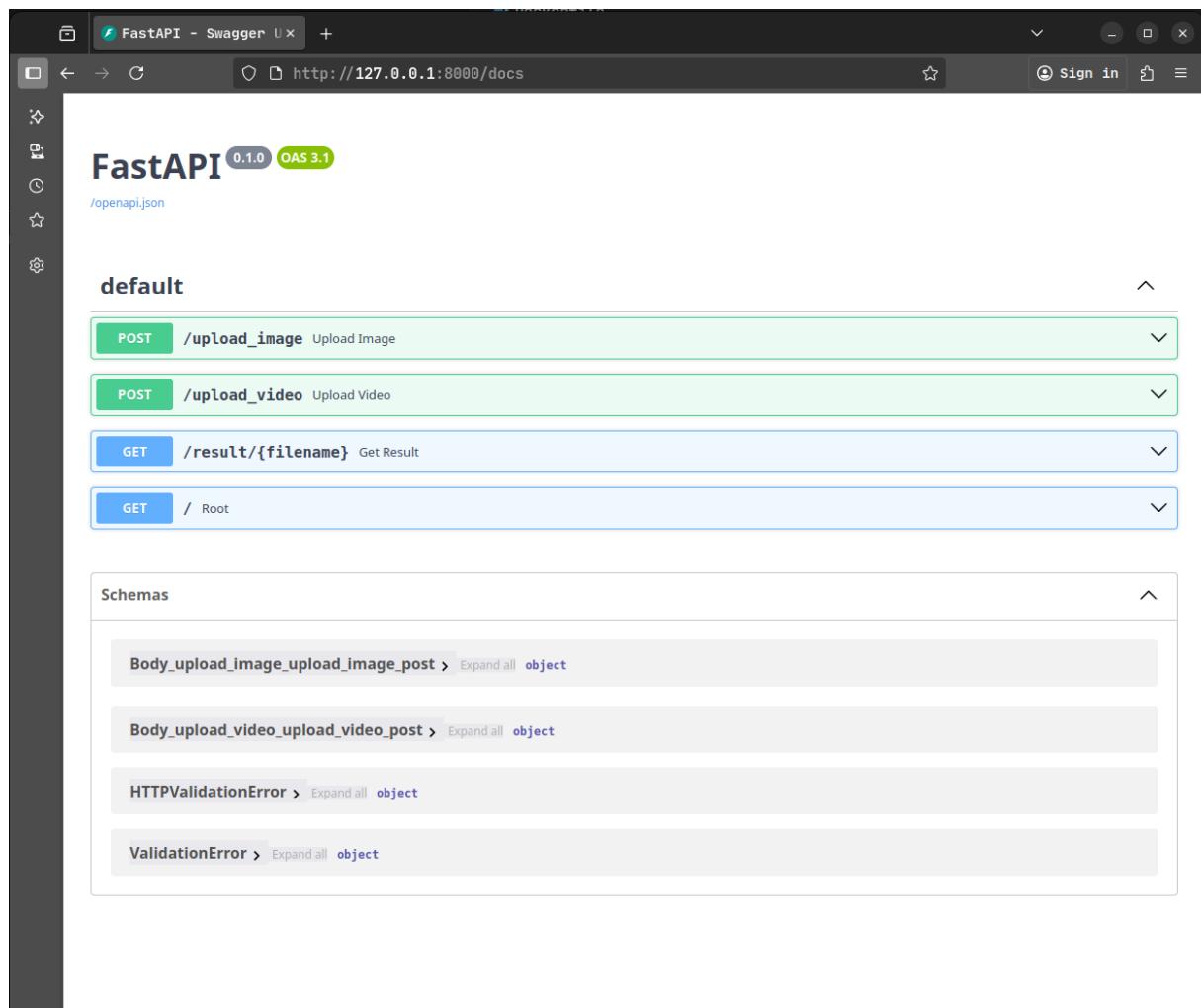
Şekil 6.3 Top Tespiti ve Tahmini Konum Noktası

### 6.3. Backend

Bu projenin backend kısmı FastAPI kullanılarak geliştirilmiştir. Sistem, kullanıcının gönderdiği görsel veya video verisini alır, ilgili modellerle işleme tabi tutar ve çıktıları doğrudan dosya olarak kullanıcıya sunar. API dört temel uç noktadan oluşmaktadır:

- **POST/upload\_image**: JPG, PNG veya JPEG formatındaki tek bir resmi alır ve top tespitini gerçekleştirerek anotasyonlu görseli döndürür.
- **POST/upload\_video**: Sadece MP4 formatındaki videoyu kabul eder ve önceki bölümlerde anlatılan pipeline'ı çalıştırır; top tespiti, LSTM tahmini ve anotasyonlu video üretir.
- **GET/result**: İşlem tamamlandıktan sonra üretilen çıktı dosyalarını getirir.
- **GET/**: Frontend arayüzüne barındırır; detaylar bir sonraki bölümde açıklanacaktır.

Backend kodları modüler ve sürdürülebilir olacak şekilde yapılandırılmıştır. Her Python dosyası (module) belirli bir amaca hizmet eder ve anlaşılabilirliği artırmak için yorum satırlarıyla desteklenmiştir.



Şekil 6.4 Swagger Üzerinden API Uç Noktaları

```
backend
  └── core
      ├── config.py
      └── exceptions.py
  # Konfigürasyon sabitleri ve klasör yolları
  # Video işleme ve dosya hataları için özel istisnalar
  # └── progress.py
  # Uzun süren görevler için ilerleme takibi sınıfı
  └── models
      └── best.pt
  # Top tespiti için YOLO model ağırlıkları
  # └── trained_Istm_ball.keras
  # Top yörüngesi tahmini için LSTM modeli
  └── pipeline
      └── annotate.py
  # Tespit edilen veya tahmin edilen koordinatlarla video karelerini
  işaretler
  |   └── csv_export.py
  # Koordinatları CSV olarak dışa aktarır
  |   └── detect.py
  # YOLO tabanlı top tespit pipeline'sı
  |   └── graphs.py
  # Koordinat verilerinden yörünge grafikleri oluşturur
  |   └── init.py
  |   └── predict.py
  # Top yörüngesi için LSTM tahmini
  └── routes
      ├── image_routes.py
      # Resim ile ilgili API uç noktaları
      └── video_routes.py
      # Video yükleme ve işleme API uç noktaları (isimde yazım hatası:
  video_routes.py olmalı)
      └── websocket_routes.py
  # Gerçek zamanlı güncellemler/ilerleme için WebSocket uç noktaları
  └── api.py
  # FastAPI uygulama giriş noktası, rotaların kaydı
  └── Istm.py
  # LSTM modelini yükleme ve tahmin sarmalayıcısı
  └── yolo.py
  # YOLO tespiti, yükleme ve çıkışım sarmalayıcısı

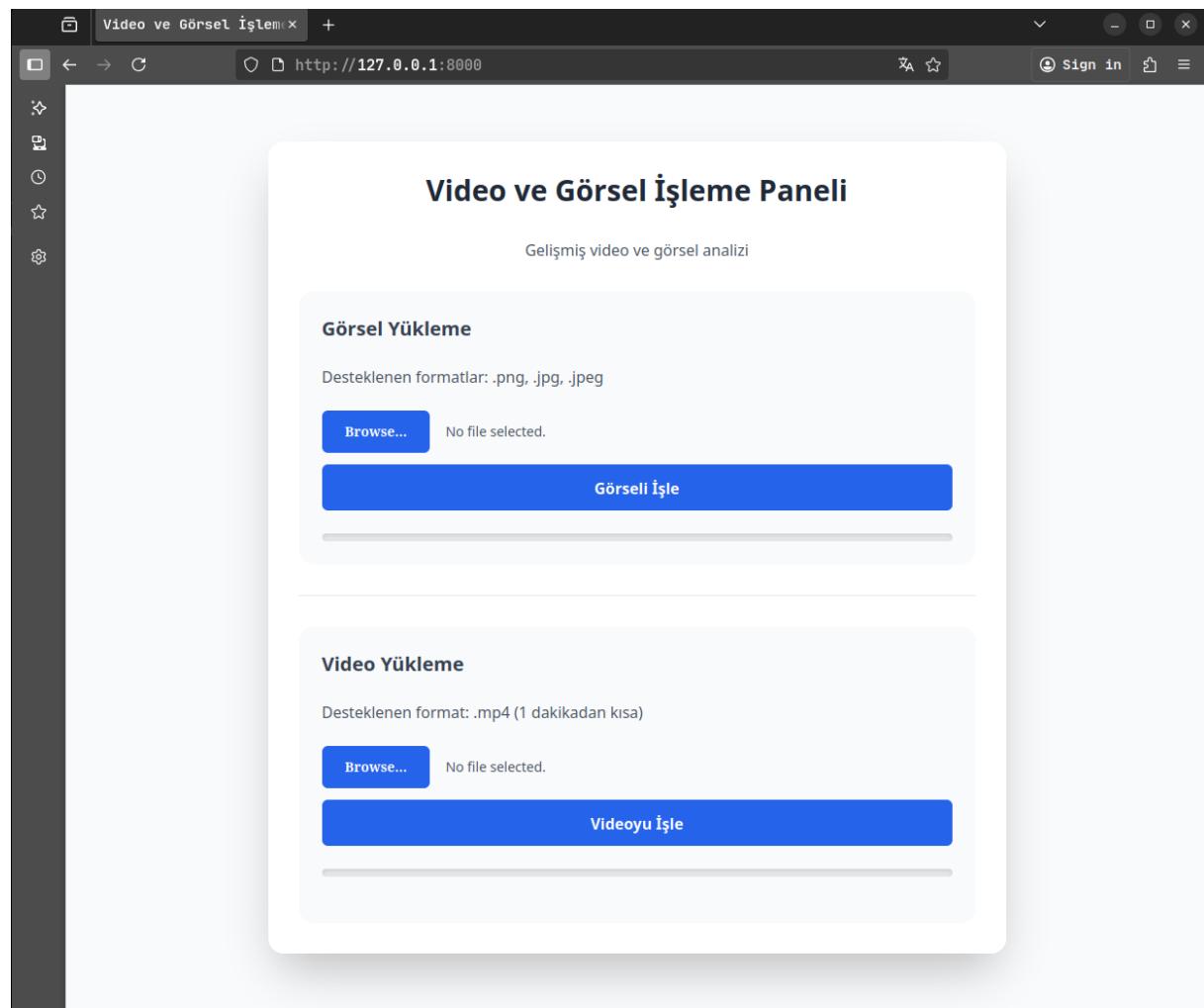
frontend                                # Ön yüz HTML/CSS/JS dosyaları
.gitignore                               # Git ignore yapılandırması
Dockerfile                                # Docker konteyner kurulumu
README.md                                 # Proje README dosyası
requirements.txt                          # Python bağımlılıkları
```

## 6.4 Frontend

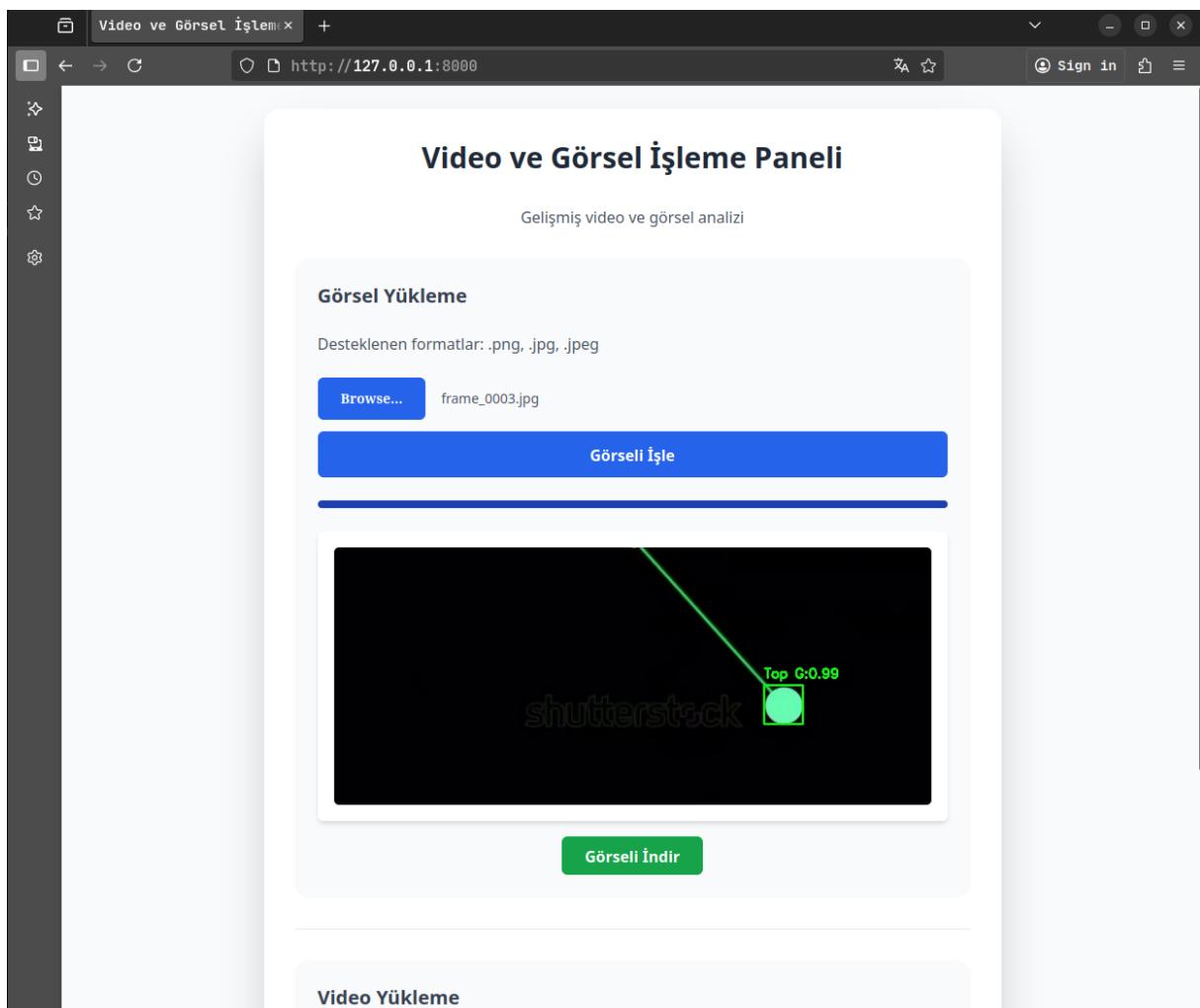
Projede ön yüz tasarımlı için HTML ve JavaScript kullanılmış, Tailwind CSS ile modern ve kullanıcı dostu bir arayüz oluşturulmuştur. Ön yüz, kullanıcıya adım adım rehberlik eder, işlemlerin ilerleyişini açıkça gösterir ve sonuçları anında sunar. Ayrıca, hatalar veya eksik bilgiler durumunda kullanıcıya dostane uyarılar sağlar.

Arayüzün temel özellikleri:

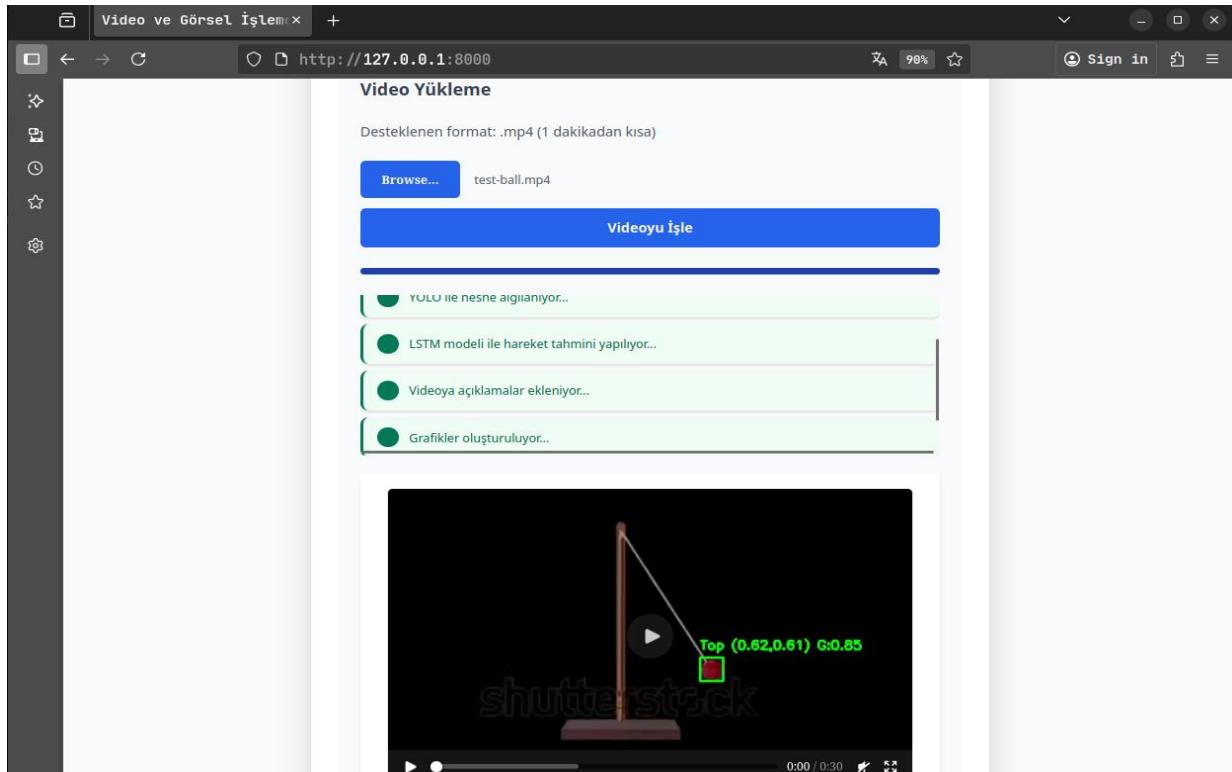
- **Basit ve modern tasarım:** Karmaşıklıktan uzak, anlaşılır bir yapı.
- **İşlem ilerleme göstergesi:** Video yükleme ve işleme sırasında süreci görselleştirme.
- **Sonuçların sunumu:** Tahmin edilen koordinatlar, yörunge grafikleri ve CSV dosyalarını görüntüleme ve indirme imkanı.
- **Kullanıcı dostu uyarılar:** Dosya formatı hataları veya eksik girişlerde bilgilendirme.



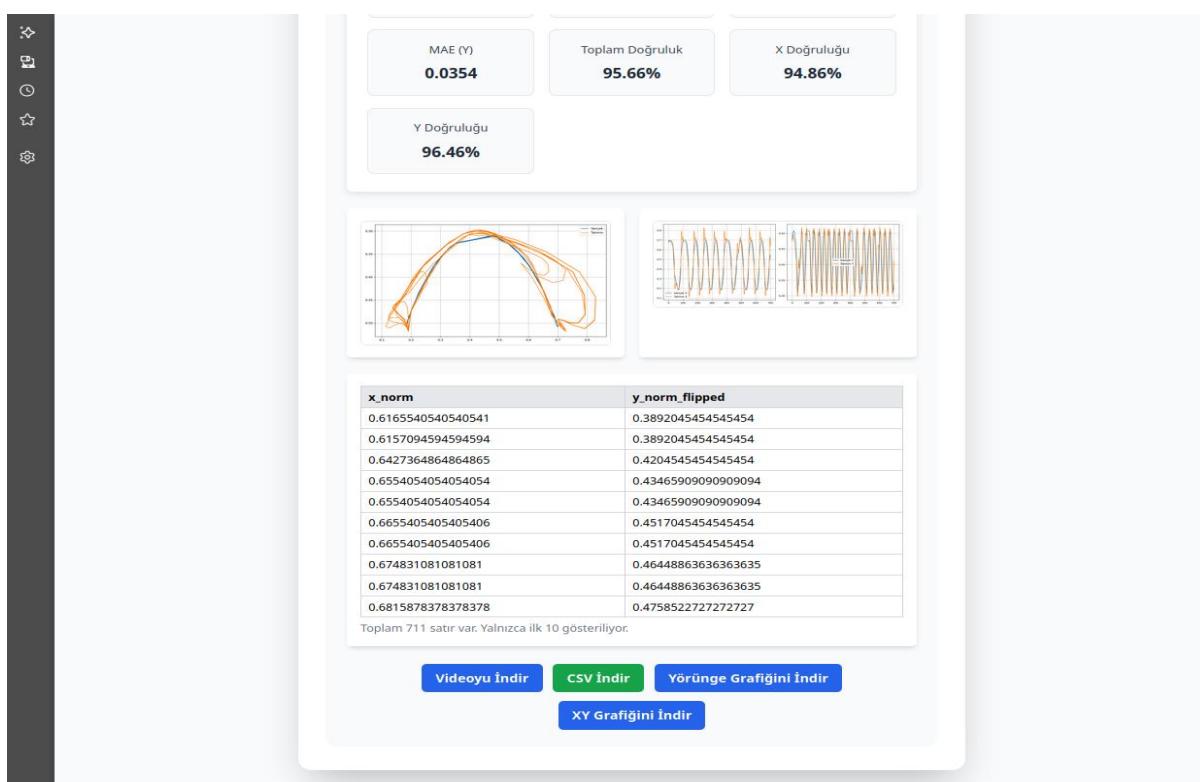
Şekil 6.5 Projenin web arayüzü genel görünümü



Şekil 6.6 Fotoğraf yükleme seçeneği ile top tespiti arayüzü



Şekil 6.7 Video işleme sırasında ilerleme göstergesi ve çıktı önizlemesi



Şekil 6.8 İşlem tamamlandıktan sonra video sonuçları, metrikler, grafikler ve CSV çıktıları ile indirme seçenekleri

## 6.5 Docker

Projeyi taşınabilir ve kolay dağıtılabılır hale getirmek amacıyla Dockerfile hazırlanmıştır. Dockerfile, gerekli tüm bağımlılıkları içerir ve adım adım yorumlarla açıklanmıştır, bu sayede yapı ve kurulum süreci anlaşılır ve tekrarlanabilir hale getirilmiştir.

Docker ile proje, herhangi bir ek kurulum gerektirmeden çalıştırılabilir; bu sayede farklı platformlarda hızlıca test ve dağıtım yapılabilir. Proje Docker Hub üzerinde de yayınlanmış olup, ilgili bağlantı raporun sonunda verilmiştir.

### Başlıca avantajları:

- Taşınabilirlik:** Proje bağımlılıkları Docker imajı içinde paketlendiği için her ortamda çalıştırılabilir.
- Kolay dağıtım:** Tek bir komut ile konteyner çalıştırılabilir.
- Tekrarlanabilir yapı:** Dockerfile yorumları ve adımları, kullanıcıların projeyi kendi ortamlarında sorunsuz kurmasını sağlar.

```
# Python 3.13 taban imajı
FROM python:3.13-slim

# Çalışma dizinini ayarla
WORKDIR /app

# Minimal bağımlılıkları ve OpenCV kütüphanelerini yükle
RUN apt-get update && apt-get install -y --no-install-recommends \
    build-essential curl git libgl1 libglib2.0-0 && \
    rm -rf /var/lib/apt/lists/*

# Önce requirements.txt'yi kopyala, önbellek için
COPY requirements.txt .

# PyTorch CPU ve diğer Python bağımlılıklarını yükle
RUN pip install --no-cache-dir torch torchvision torchaudio --index-url \
    https://download.pytorch.org/whl/cpu \
    && pip install --no-cache-dir -r requirements.txt

# İmaj boyutunu azaltmak için build araçlarını kaldır
RUN apt-get purge -y build-essential git && apt-get autoremove -y

# Proje dosyalarını kopyala
COPY . .

# FastAPI varsayılan portunu aç
EXPOSE 8000

# FastAPI uygulamasını uvicorn ile çalıştır
CMD ["python3", "-m", "uvicorn", "backend.api:app", "--host", "0.0.0.0",
    "--port", "8000"]
```

Komut	Açıklama	Amacı
<code>sudo docker build -t theanywayguy/bitirme_odevi_1:latest .</code>	Mevcut dizinde bulunan Dockerfile dosyasını kullanarak bir Docker İmajı oluşturur. -t (tag) bayrağı, oluşturulan imaja ball-tracking-app adını ve latest etiketini verir. Sondaki nokta (.) ise Dockerfile'in bulunduğu bağlam dizinini belirtir.	Projenin tüm bağımlılıklarını, kodunu ve çalışma ortamını içeren, platformdan bağımsız, taşınabilir bir kalıp (imaj) oluşturmak.
<code>sudo docker pull theanywayguy/bitirme_odevi_1:latest</code>	Belirtilen kullanıcı adına ait Docker İmajını uzaktan (örneğin Docker Hub'dan) yerel makineye indirir.	İmajı yerel olarak derlemeye gerek kalmadan, doğrudan kullanımına hazır önceden oluşturulmuş imajı indirmek ve dağıtım sürecini hızlandırmak.
<code>sudo docker run -d -p 8000:8000 --name bitirme_odevi_container theanywayguy/bitirme_odevi_1:latest</code>	İndirilen veya yerelde oluşturulan ball-tracking-app:latest imajından yeni bir konteyner başlatır.	Uygulamayı çalışırmak ve erişilebilir hale getirmek. Konteyneri arka planda (-d) çalışırmak ve port yönlendirmesi yapmak (-p 8000:8000).

Tablo 6.4 Projede Kullanılan Docker Komutlarının İşlevsel Özeti

## 6.6 Dağıtım (Deployment)

Docker imajı Docker Hub'a gönderildikten sonra, proje Render platformu kullanılarak dağıtılmıştır. Ücretsiz platform olması ve GPU desteğinin bulunmaması nedeniyle bazı işlemler yavaş çalışsa da, sistemin tamamı işlevsel bir şekilde çalışmaktadır ve kullanıcılar tüm fonksiyonları test edebilir.

Proje, aşağıdaki link üzerinden erişilebilir:

<https://bitirme-odevi-1.onrender.com/>

**Avantajlar:**

- Docker ile paketlenmiş proje, Render üzerinde kolayca çalıştırılabilir.
- Tüm backend ve frontend fonksiyonları tek bir platformda erişilebilir hale gelmiştir.
- Kullanıcılar, video yükleme, tahmin, sonuç görselleştirme ve indirme işlemlerini sorunsuz gerçekleştirebilir.

## Video ve Görsel İşleme Paneli

Gelişmiş video ve görsel analizi

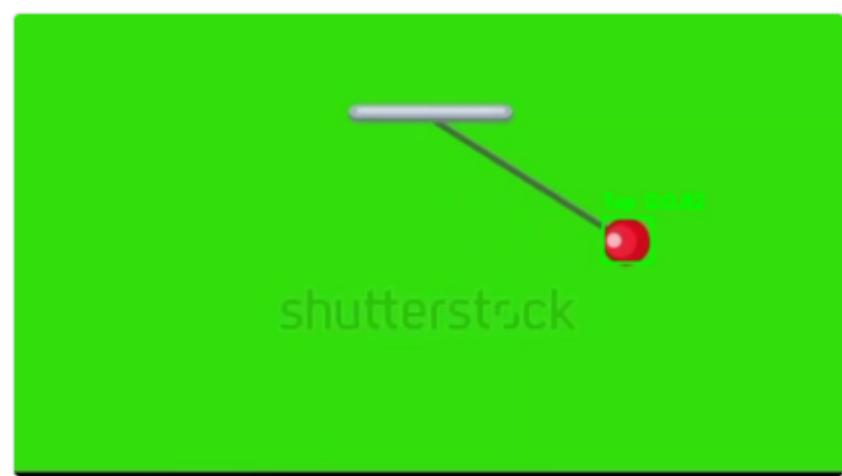
### Görsel Yükleme

Desteklenen formatlar: .png, jpg, jpeg

Choose File

frame\_0002.jpg

Görseli İşle



Şekil 6.9: Render üzerinde çalışan uygulamanın web arayüzü.

**EKLER:**

**Dockerhub Linki:**

[https://hub.docker.com/r/theanywayguy/bitirme\\_odevi\\_1](https://hub.docker.com/r/theanywayguy/bitirme_odevi_1)

**Github Linki:**

[https://github.com/theanywayguy/Bitirme\\_Odevi1-Final-Project1](https://github.com/theanywayguy/Bitirme_Odevi1-Final-Project1)

**Website linki:**

<https://bitirme-odevi-1.onrender.com/>



T.C.  
MERSİN ÜNİVERSİTESİ  
MÜHENDİSLİK FAKÜLTESİ  
BİLGİSAYAR MÜHENDİSLİĞİ  
HAFTALIK RAPOR

[06/10/2025]

Öğrencinin Adı ve Soyadı	AHMAD AJAJ
Danışmanın Unvanı, Adı ve Soyadı	Doç. Dr. FARUK SERİN
Bitirme Projesi Başlığı	Gerçek Zamanlı Nesne Tespiti, Takibi ve Hareket Tahmini için Entegre Derin Öğrenme Yaklaşımı

#### 1. Bu Hafta Yapılan Çalışmalar:

- Sarkacın videoları toplandı.
- Videolar **FFmpeg** kullanılarak karelere dönüştürüldü.
- Sarkacın konumları kareler üzerinde **Roboflow** ile etiketlendi.
- Etiketlenmiş kareler, eğitim, doğrulama ve test setlerini içeren **YOLOv8 uyumlu veri setleri** olarak çıkarıldı.

#### 2. Gelecek Hafta Planlanan Çalışmalar:

- Hazırlanan veri seti kullanılarak **YOLOv8 modeli** Google Colab GPU ortamında eğitilecek.
- Tespit doğruluğu test edilecek ve video kareleri üzerinde ön testler yapılacak.

#### 3. Karşılaşılan Sorunlar / Zorluklar:

- Video koşullarının sınırlı çeşitliliği, modelin genelleme kabiliyetini azaltabilir.
- Küçük veri seti, YOLOv8 eğitimi sırasında aşırı öğrenmeye(overfitting) neden olabilir.

#### 4. Alınan Önlemler / Çözümler

- Veri setinin çeşitliliğini artırmak için veri artırma (rotation, scaling, parlaklık ayarları vb.) uygulanacak.
- Eğitim metrikleri yakından izlenecek ve aşırı öğrenme gözlemlenirse model veya veri seti üzerinde gerekli düzenlemeler yapılacak.



**T.C.  
MERSİN ÜNİVERSİTESİ  
MÜHENDİSLİK FAKÜLTESİ  
BİLGİSAYAR MÜHENDİSLİĞİ  
HAFTALIK RAPOR-2**

[13/10/2025]

Öğrencinin Adı ve Soyadı	AHMAD AJAJ
Danışmanın Unvanı, Adı ve Soyadı	Doç. Dr. FARUK SERİN
Bitirme Projesi Başlığı	Gerçek Zamanlı Nesne Tespiti, Takibi ve Hareket Tahmini için Entegre Derin Öğrenme Yaklaşımı

**1. Bu Hafta Yapılan Çalışmalar:**

- Roboflow veri seti Colab ortamında YOLOv8 ile eğitildi.
- Eğitim sürecinde aşırı öğrenme (overfitting) gözlemlendi
- Bunu azaltmak için veri artırma, erken durdurma ve sınırlı epoch kullanıldı.
- Model test edildi ve topun konumunu başarıyla tespit ettiği doğrulandı.

**2. Gelecek Hafta Planlanan Çalışmalar:**

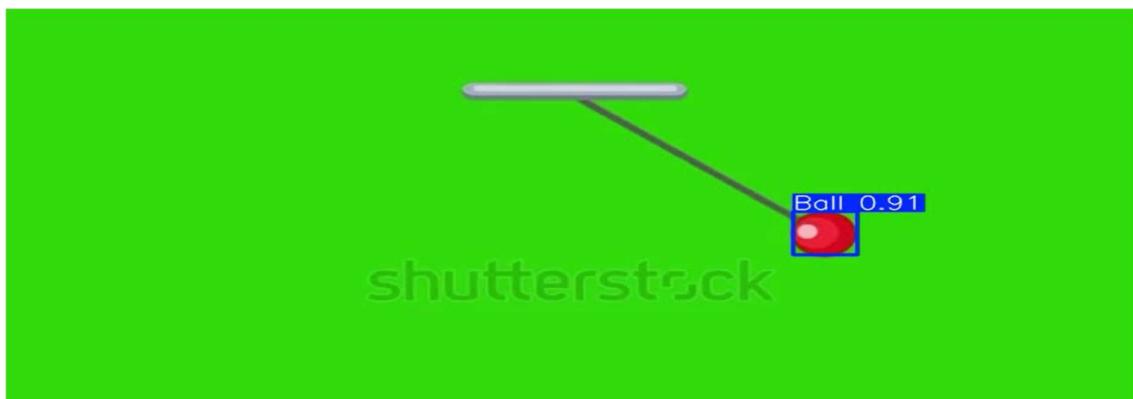
- YOLO modelinden çıkarılan (x, y) koordinatları CSV dosyalarına aktarılacak.
- Bu veriler, geçmiş haftalardaki bilgilerle birleştirilerek LSTM modeli eğitilecek.
- LSTM'nin tahmin performansı değerlendirilecek.

**3. Karşılaşılan Sorunlar / Zorluklar:**

- Küçük veri seti nedeniyle modelde aşırı öğrenme eğilimi görüldü.
- Arka plan çeşitliliği düşük olduğu için genelleme sınırlı kaldı.

**4. Alınan Önlemler / Çözümler**

- Veri artırma (rotation, parlaklık, gürültü vb.) teknikleri uygulandı.
- Early stopping kullanılarak gereksiz uzun eğitim önlendi.
- Eğitim metrikleri izlenerek optimum model kaydedildi.



YOLOv8 ile Gerçekleştirilen Top Tespiti (Proof of Concept)



**T.C.  
MERSİN ÜNİVERSİTESİ  
MÜHENDİSLİK FAKÜLTESİ  
BİLGİSAYAR MÜHENDİSLİĞİ  
HAFTALIK RAPOR-3**

[20/10/2025]

Öğrencinin Adı ve Soyadı	AHMAD AJAJ
Danışmanın Unvanı, Adı ve Soyadı	Doç. Dr. FARUK SERİN
Bitirme Projesi Başlığı	Gerçek Zamanlı Nesne Tespiti, Takibi ve Hareket Tahmini için Entegre Derin Öğrenme Yaklaşımı

**1. Bu Hafta Yapılan Çalışmalar:**

- YOLOv8'den çıkarılan merkez koordinatları ([x\_norm, y\_norm\_flipped]) CSV dosyasına aktarıldı.
- Oluşturulan CSV verisi kullanılarak LSTM modeli eğitildi ve model .keras formatında kaydedildi.
- LSTM modelinin tahmin performansı değerlendirildi ve tatmin edici bulundu.

**2. Gelecek Hafta Planlanan Çalışmalar:**

- YOLOv8 ve LSTM modelleri birleştirilerek tek bir pipeline oluşturulacak.
- Bu pipeline üzerinde nesne tespiti, takip ve hareket tahmini test edilecek.

**3. Karşılaşılan Sorunlar / Zorluklar:**

- LSTM eğitimi sırasında zaman serisi uzunluğuna bağlı öğrenme zorlukları gözlandı.
- Koordinat normalizasyonunda bazı karelere sapmalar görülebilir.

**4. Alınan Önlemler / Çözümler**

- LSTM eğitimi sonrası model performansı izlenerek, en iyi ağırlıklar kaydedildi.
- LSTM eğitim parametreleri (epoch, batch, learning rate) optimize edildi.
- Koordinat normalizasyon ve y-eksen çevirme işlemleri doğrulandı ve tutarlı hale getirildi.



T.C.  
MERSİN ÜNİVERSİTESİ  
MÜHENDİSLİK FAKÜLTESİ  
BİLGİSAYAR MÜHENDİSLİĞİ  
HAFTALIK RAPOR-4

[27/10/2025]

Öğrencinin Adı ve Soyadı	AHMAD AJAJ
Danışmanın Unvanı, Adı ve Soyadı	Doç. Dr. FARUK SERİN
Bitirme Projesi Başlığı	Gerçek Zamanlı Nesne Tespiti, Takibi ve Hareket Tahmini için Entegre Derin Öğrenme Yaklaşımı

**1. Bu Hafta Yapılan Çalışmalar:**

- YOLOv8 ve LSTM modelleri birleştirilerek tek bir pipeline oluşturuldu.
- Pipeline, video veya görsel girişi alarak tespit edilen nesnelerin koordinatlarını CSV formatında kaydetmektedir.
- LSTM modeli bu koordinatları kullanarak topun gelecekteki konumlarını tahmin etmektedir.
- Çıktı olarak hem tahmin edilen koordinatlar (CSV) hem de tahminle görselleştirilmiş video dosyası üretilmiştir.

**2. Gelecek Hafta Planlanan Çalışmalar:**

- Pipeline'ı arka planda (backend) çalıştıracak bir FastAPI tabanlı servis geliştirilecektir.
- Backend, video/görsel girdisini alıp CSV ve video çıktısı döndürecek şekilde yapılandırılacaktır.

**3. Karşılaşılan Sorunlar / Zorluklar:**

- YOLOv8 çıktı formatının LSTM girişiyile uyumlantırılması sırasında veri boyutları üzerinde ayarlamalar gerekmıştır.
- Büyük boyutlu videolarda işlem süresi uzamıştır.

**4. Alınan Önlemler / Çözümler**

- LSTM girişi için normalize edilmiş ve sıralı veri oluşturma fonksiyonu geliştirildi.
- Pipeline performansı artırmak için gereksiz video kareleri filtreldi.



**T.C.  
MERSİN ÜNİVERSİTESİ  
MÜHENDİSLİK FAKÜLTESİ  
BİLGİSAYAR MÜHENDİSLİĞİ  
HAFTALIK RAPOR-5**

[3/11/2025]

Öğrencinin Adı ve Soyadı	AHMAD AJAJ
Danışmanın Unvanı, Adı ve Soyadı	Doç. Dr. FARUK SERİN
Bitirme Projesi Başlığı	Gerçek Zamanlı Nesne Tespiti, Takibi ve Hareket Tahmini için Entegre Derin Öğrenme Yaklaşımı

**1. Bu Hafta Yapılan Çalışmalar:**

- Pipeline'ı çalıştırınan **FastAPI tabanlı backend** geliştirildi.
- Backend, video veya görsel dosyasını alıp YOLOv8 + LSTM tahminini çalıştmakta ve sonuçları:
- `predicted_trajectory.csv` (koordinatlar), `output_with_predictions.mp4` veya `.png` (tahminli görsel) olarak üretmektedir.
- Backend testleri yerel ortamda başarıyla tamamlandı.

**2. Gelecek Hafta Planlanan Çalışmalar:**

- Backend için kullanıcı arayüzü (frontend) geliştirilecek.
- Frontend üzerinden dosya yükleme ve tahmin sonuçlarını görüntüleme işlemleri sağlanacaktır.

**3. Karşılaşılan Sorunlar / Zorluklar:**

- Büyük video dosyalarının API'ye yüklenmesi zaman almıştır.
- Video işleme sırasında bellek tüketimi artmıştır.

**4. Alınan Önlemler / Çözümler**

- Maksimum video uzunluğu ve çözünürlüğü için kısıtlamalar eklendi.
- Dosya yükleme boyutu sınırlaması FastAPI tarafından ayarlandı.



T.C.  
MERSİN ÜNİVERSİTESİ  
MÜHENDİSLİK FAKÜLTESİ  
BİLGİSAYAR MÜHENDİSLİĞİ  
HAFTALIK RAPOR-6

[17/11/2025]

Öğrencinin Adı ve Soyadı	AHMAD AJAJ
Danışmanın Unvanı, Adı ve Soyadı	Doç. Dr. FARUK SERİN
Bitirme Projesi Başlığı	Gerçek Zamanlı Nesne Tespiti, Takibi ve Hareket Tahmini için Entegre Derin Öğrenme Yaklaşımı

#### **1. Bu Hafta Yapılan Çalışmalar:**

- Backend'e bağlanan basit bir frontend arayüzü geliştirildi.
- Kullanıcılar video veya görsel yükleyerek tahmin işlemini başlatabilmekte.
- Tahmin sonrası sistem, yalnızca tespit yapıldığında fotoğraf; video ve hareket tahmini içeren durumlarda ise video ve CSV çıktısını doğrudan görüntülemekte ve kullanıcıya sunmaktadır.
- Arayüzde işlem ilerleme durumu ve tahmin doğruluk yüzdesi (%95-98 civarı) gösterilmektedir.

#### **2. Gelecek Hafta Planlanan Çalışmalar:**

- Tüm proje bileşenleri Docker konteyneri içerisine alınarak tek paket haline getirilecektir.
- Docker üzerinden hem frontend hem backend birlikte çalıştırılacaktır.

#### **3. Karşılaşılan Sorunlar / Zorluklar:**

- Frontend ile backend arasında dosya transferinde gecikmeler yaşandı.
- Büyük dosyalarda tarayıcı tarafında geçici donmalar görüldü.

#### **4. Alınan Önlemler / Çözümler**

- Video yükleme sırasında asenkron işlem desteği eklendi.
- Frontend tarafında dosya boyut kontrolü ve kullanıcıya uyarı mekanizması eklendi.



**T.C.  
MERSİN ÜNİVERSİTESİ  
MÜHENDİSLİK FAKÜLTESİ  
BİLGİSAYAR MÜHENDİSLİĞİ  
HAFTALIK RAPOR-7**

[24/11/2025]

Öğrencinin Adı ve Soyadı	AHMAD AJAJ
Danışmanın Unvanı, Adı ve Soyadı	Doç. Dr. FARUK SERİN
Bitirme Projesi Başlığı	Gerçek Zamanlı Nesne Tespiti, Takibi ve Hareket Tahmini için Entegre Derin Öğrenme Yaklaşımı

**1. Bu Hafta Yapılan Çalışmalar:**

- Proje tamamen Docker ortamına taşındı.
- Tek Docker imajı içerisinde backend ,frontend, ve model dosyaları (.pt, .keras) birleştirildi.
- Yerel testlerde pipeline başarıyla çalıştı, tüm çıktı dosyaları üretildi.

**2. Gelecek Hafta Planlanan Çalışmalar:**

- Uygulamanın bulut ortamına (ör. Render, Railway, Hugging Face Spaces veya DockerHub tabanlı servis) dağıtımları planlanmaktadır.
- Uzaktan erişim testleri ve kullanıcı arayüzü optimizasyonu yapılacaktır.



**T.C.  
MERSİN ÜNİVERSİTESİ  
MÜHENDİSLİK FAKÜLTESİ  
BİLGİSAYAR MÜHENDİSLİĞİ  
HAFTALIK RAPOR-8**

[1/12/2025]

Öğrencinin Adı ve Soyadı	AHMAD AJAJ
Danışmanın Unvanı, Adı ve Soyadı	Doç. Dr. FARUK SERİN
Bitirme Projesi Başlığı	Gerçek Zamanlı Nesne Tespiti, Takibi ve Hareket Tahmini için Entegre Derin Öğrenme Yaklaşımı

**1. Bu Hafta Yapılan Çalışmalar:**

- Docker konteyneri Render üzerinden başarıyla dağıtıldı.
- Tüm API uç noktaları uzaktan test edilerek doğrulandı.
- Sistem bütününen çalışma durumu doğrulandı ve tüm fonksiyonların sorunsuz çalıştığı teyit edildi.

**2. Gelecek Hafta Planlanan Çalışmalar:**

- Final raporunun hazırlanması