amblem, logo, ticari marka, simge, sembol içeren bir resim

Açıklama otomatik olarak oluşturuldu

**T.C**

**KOCAELİ SAĞLIK VE TEKNOLOJİ ÜNİVERSİTESİ**

**MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ**

**YAZILIM MÜHENDİSLİĞİ**

**PROJE KONUSU:**

**AKILLI ÇÖP AYRIŞTIRMA SİSTEMİ**

**FİNAL RAPORU**

**GRUP NUMARASI:6**

**220502025 EMRAH ŞAHİN**

**220502015 TALHA TUNA**

**220502021 NURDAN BULUT**

**220502041 GÜREL BİLGİN**

**DERS SORUMLUSU:**

**DR. ÖĞR. ÜYESİ ELİF PINAR HACIBEYOĞLU**

**TARİH:22.01.2025**

İçindekiler

[**1.** **GİRİŞ** 3](#_Toc188194751)

[**1.1** **Projenin Adı** 3](#_Toc188194752)

[**1.2** **Projenin Amacı ve Önemi** 3](#_Toc188194753)

[**2.** **GÖREV DAĞILIMI VE ZAMAN ÇİZELGESİ** 3](#_Toc188194754)

[**3.** **GEREKSİNİMLER** 4](#_Toc188194755)

[**4.** **UYGULAMA VE TASARIM** 5](#_Toc188194756)

[**4.1** **Kod Bileşenleri** 5](#_Toc188194757)

[**4.2** **Arayüz Bileşenleri** 7](#_Toc188194758)

[**4.3** **Karşılaşılan Zorluklar ve Çömzümleri** 11](#_Toc188194759)

[**5.** **TESTLER VE SONUÇLAR** 12](#_Toc188194760)

[**5.1** **Test Tablosu** 12](#_Toc188194761)

[**5.2** **Eğitim Sonuçları** 13](#_Toc188194762)

[**5.3** **Eğitim ve Doğrulama (Validation) Kayıpları ve Metrikleri** 23](#_Toc188194763)

[**5.4** **Nesne Tespiti Program Çıktıları** 25](#_Toc188194764)

[**6.** **PROJENİN GELECEĞİ** 28](#_Toc188194765)

## **GİRİŞ**

## **Projenin Adı**

Akıllı Çöp Ayrıştırma Sistemi

## **Projenin Amacı ve Önemi**

Günümüzde hızlı kentleşme, artan nüfus ve sanayi faaliyetleri nedeniyle atık miktarı hızla artmaktadır. Bu durum, çevre üzerinde ciddi bir baskı oluşturmakta ve etkili atık yönetimi sistemlerinin geliştirilmesini zorunlu kılmaktadır. Yanlış ayrıştırılan atıklar geri dönüşüm süreçlerini sekteye uğratarak, geri dönüştürülebilir materyallerin çöp sahalarına gitmesine ve doğal kaynakların gereksiz yere tüketilmesine yol açmaktadır. Bu nedenle, doğru atık ayrıştırma, sürdürülebilir bir çevre yaratma yolunda kritik bir adım olarak görülmektedir.

Bu projenin amacı, yapay zeka destekli bir atık ayrıştırma sistemi geliştirerek, atık yönetimi süreçlerini daha verimli ve etkili hale getirmektir. Sistem, kamera görüntüleri aracılığıyla atıkların türünü (plastik, cam, kağıt, metal) tespit ederek doğru sınıflandırmayı sağlar. Böylece, geri dönüşüm oranlarını artırarak doğal kaynak kullanımını optimize etmeyi ve çevre kirliliğini azaltmayı hedefler.

Bu tür bir sistemin kullanımı, yalnızca çevresel fayda sağlamakla kalmaz, aynı zamanda ekonomik açıdan da önemli avantajlar sunar. Geri dönüşüm süreçlerinin hızlanması ve doğru ayrıştırma sayesinde atık işleme maliyetleri düşer. akıllı çöp ayrıştırma sistemi, çevresel sorunların çözümünde teknolojiyi kullanarak sürdürülebilir bir geleceğe katkı sunmayı hedeflemektedir. Bu proje, atık yönetiminde devrim niteliğinde bir adım atarak, çevre bilincinin artırılmasını ve kaynakların daha verimli kullanılmasını sağlayacak önemli bir uygulamadır.

## **GÖREV DAĞILIMI VE ZAMAN ÇİZELGESİ**

* 1. **Görev Dağılımı**

**Tablo1: Görev Dağılımı**

|  |  |
| --- | --- |
| **Grup Üyesi** | **Görevi** |
| Emrah Şahin (Proje Yöneticisi) | Yazılım geliştirici, Model Eğitimi |
| Gürel Bilgin | Yazılım geliştirici, Arayüz |
| Nurdan Bulut | Dokümantasyon, Veri Etiketleme |
| Talha Tuna | Dokümantasyon, Test |

**1.tablo görev dağılımını göstermektedir**

Proje sürecinde, ekip üyeleri belirli görev alanlarına ayrılarak iş yükü dengeli bir şekilde paylaştırılmıştır. Proje yöneticisi olarak görev alan Emrah Şahin, yazılım geliştirme ve model eğitimi süreçlerini üstlenmiştir. Gürel Bilgin, yazılım geliştirme çalışmalarına katkıda bulunarak arayüz tasarımında görev almıştır. Nurdan Bulut, projenin dokümantasyonunu hazırlamış ve veri etiketleme işlemlerinde aktif rol almıştır. Talha Tuna ise dokümantasyon süreçlerine katkı sağlarken test aşamalarını yürütmüştür. Görev dağılımının bu şekilde yapılması, proje hedeflerine planlanan süre içinde ulaşılmasını ve her aşamanın eksiksiz tamamlanmasını sağlamıştır. Ekip içi iş birliği ve sorumluluk paylaşımı sayesinde proje etkin bir şekilde tamamlanmıştır.

* 1. **Zaman Çizelgesi**

**Tablo2: Zaman Yönetimi**

|  |  |
| --- | --- |
| **İş Paketi** | **Hafta** |
| Gereksinimlerin analizi ve planlama | 22-30 Kasım |
| Veri seti analizi | 1-14 Aralık |
| Model eğitimi | 14-18 Aralık |
| Program oluşturulması | 18-21 Aralık |
| Sistem entegrasyonu ve testler | 22-31 Aralık |

Proje boyunca belirlenen zaman çizelgesine uygun olarak ilerlenmiş ve iş paketi ilerleme raporunda belirtilen süre içinde tamamlanmıştır. İlk olarak 22-30 Kasım tarihleri arasında gereksinimlerin analizi ve planlama çalışmaları gerçekleştirilmiştir. Ardından, 1-14 Aralık tarihleri arasında veri seti analizi yapılmış, veri kaynakları belirlenerek veri temizleme ve düzenleme işlemleri tamamlanmıştır. 14-18 Aralık tarihleri arasında model eğitimi gerçekleştirilmiş ve modelin performansı değerlendirilmiştir. 18-21 Aralık tarihleri arasında programın oluşturulması ve arayüz tasarımı yapılmıştır. Son olarak, 22-31 Aralık tarihleri arasında sistem entegrasyonu ve testler tamamlanarak proje başarıyla sonuçlandırılmıştır. Bu zaman çizelgesi sayesinde proje adım adım planlı bir şekilde ilerlemiş ve süreçlerin kontrol altında tutulması sağlanmıştır.

## **GEREKSİNİMLER**

**Tablo3: Gereksinim**

|  |  |  |
| --- | --- | --- |
| **No** | **Öncelik** | **Gereksinim** |
| G1 | 5 | Sistem, kamera aracılığıyla bir atığı algıladığında, atığın türünü (örneğin, plastik, cam, kâğıt vb.) tanımlayacaktır. |
| G2 | 4 | Sistem, tespit edilen atık türünü görsel ile kullanıcıya iletecektir |
| G3 | 5 | Sistem, gerçek zamanlı olarak atık algılama ve tanımlama işlemini maksimum 2 saniyelik bir gecikmeyle gerçekleştirecektir. |
| G4 | 3 | Sistem, her atık algılama işlemini bir veri tabanına kaydedecek ve bu kayıtlar analiz için erişilebilir olacaktır. |
| G5 | 4 | Sistem, doğru bir şekilde sınıflandırılamayan atıklar için bir "tanımlanamayan atık" bildirimi oluşturacaktır. |
| G6 | 1 | Sistem, belirli bir süre boyunca kamera verisi alamazsa bir hata bildiriminde bulunacaktır. |
| G7 | 3 | Sistem, farklı aydınlatma koşullarında da atıkları doğru şekilde algılayacak şekilde optimize edilecektir. |

Proje kapsamında belirlenen gereksinimler, sistemin işlevselliğini sağlamak ve kullanıcıların ihtiyaçlarını karşılamak amacıyla öncelik sırasına göre listelenmiştir. G1, G2 ve G3 gibi yüksek öncelikli gereksinimler, sistemin temel işlevlerini oluşturan atık algılama, tür belirleme ve gerçek zamanlı sınıflandırma özelliklerini içermektedir. G4 gereksinimi, sistemin her atık algılama işlemini veri tabanına kaydetmesini ve bu kayıtların analiz için erişilebilir olmasını öngörmekteydi. Ancak, bu gereksinim proje kapsamından çıkarılmış ve final sürümüne dahil edilmemiştir. Bunun nedeni, proje süresi ve kaynakların verimli kullanımı açısından öncelikli hedeflere odaklanılmasıdır.

## **UYGULAMA VE TASARIM**

### **Kod Bileşenleri**

1. **Kütüphaneler:**metin, yazı tipi, ekran görüntüsü, beyaz içeren bir resim

   Açıklama otomatik olarak oluşturulducv2 (OpenCV), bilgisayarla görme işlemleri için kullanılan güçlü bir kütüphanedir; kamera veya video kaynağını açmak (cv2.VideoCapture), video çözünürlüğünü ayarlamak (cv2.CAP\_PROP\_FRAME\_WIDTH, cv2.CAP\_PROP\_FRAME\_HEIGHT) ve görüntüyü renk uzayında dönüştürmek (cv2.cvtColor) gibi işlemler için kullanılmıştır. ultralytics (YOLO), gerçek zamanlı nesne algılama ve sınıflandırma algoritmasını kolayca kullanmayı sağlayan bir kütüphanedir; bu kodda YOLO modeli yüklenmiş (YOLO sınıfı) ve her video karesi üzerinde nesne algılama (self.model.predict) yapılarak algılama sonuçları görsel olarak işlenmiştir (results[0].plot). tkinter, Python'da masaüstü uygulamaları geliştirmek için kullanılan standart GUI (Graphical User Interface) kütüphanesidir; pencere (tk.Tk()), tuşlar (tk.Button) ve görüntü göstermek için kanvas (tk.Canvas) yaratılmıştır. PIL (Pillow), görüntü işleme işlemleri için kullanılan bir kütüphanedir; OpenCV tarafından alınan kareler dönüştürülmüş (Image.fromarray), yeniden boyutlandırılmış (image.resize) ve Tkinter kanvasında gösterilmiştir (ImageTk.PhotoImage). threading, çoklu iş parçacığı oluşturmak için kullanılan bir kütüphanedir; bu kodda, nesne algılama işlemini ana GUI döngüsünden bağımsız bir iş parçacığında çalıştırmak için (threading.Thread) kullanılmış ve böylece uygulamanın arayüzü donmadan algılama işlemi devam etmiştir. queue, güvenli ve iş parçacığı uyumlu bir veri yapısıdır; algılama iş parçacığı ile GUI arasındaki veri paylaşımını sağlamak için (queue.Queue) kullanılmıştır. Algılanan kareler bu sıraya eklenip (self.frame\_queue.put), GUI döngüsü tarafından alınarak (self.frame\_queue.get) kanvasa çizilmiştir. Bu yapı, gerçek zamanlı nesne algılama yapan bir masaüstü uygulaması tasarlanmasını sağlamıştır.
2. **class YOLOv11ObjectDetection:**

**def \_\_init\_\_(self, model\_path, source, canvas, frame\_queue):**

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturulduBu kodda yer alan YOLOv11ObjectDetection sınıfının \_\_init\_\_ metodu, sınıfın başlatıcısıdır ve sınıfın bir örneği oluşturulduğunda gerekli olan başlangıç ayarlarını yapar. model\_path, YOLO modelinin eğitilmiş ağırlık dosyasının yolunu temsil eder ve bu dosya YOLO(model\_path) ile yüklenir. source, görüntü kaynağını (örneğin bir kamera indeksi veya video dosyası yolu) ifade eder. canvas, görüntülerin gösterileceği Tkinter kanvas bileşenidir. frame\_queue, algılama işleminden elde edilen karelerin GUI ile paylaşılması için kullanılan bir sıradır. Ayrıca, detection\_running adlı bir durum bayrağı, nesne algılama işleminin etkin olup olmadığını izlemek için başlangıçta False olarak ayarlanır. Bu başlatıcı metod, nesne algılama işlemi için gerekli olan tüm bileşenleri başlatır ve sınıfın diğer fonksiyonlarının çalışması için temel oluşturur.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu**def detect\_objects(self):**

detect\_objects fonksiyonu, video kaynağından görüntü alarak gerçek zamanlı nesne algılama işlemini gerçekleştirir ve sonuçları Tkinter arayüzünde göstermek için işler. cv2.VideoCapture(self.source) ile belirtilen video kaynağı (örneğin, bir kamera veya video dosyası) açılır ve görüntü çözünürlüğü 1280x720 olarak ayarlanır. Eğer kaynak açılamazsa, hata mesajı yazdırılır ve işlem durdurulur.

Bir döngü içinde, video kaynağından kareler okunur (cap.read()) ve kare alınamazsa döngü sonlandırılır. self.detection\_running bayrağı kontrol edilir; eğer algılama etkinse, YOLO modeli (self.model.predict) kare üzerinde çalıştırılır, algılama sonuçları çerçeve üzerine çizilir (results[0].plot), ve çerçeve renk uzayı dönüştürülerek (cv2.cvtColor) PIL görüntüsüne çevrilir. Eğer algılama devre dışıysa, alınan çerçeve doğrudan PIL görüntüsüne çevrilir.

Görüntü, Tkinter kanvasının boyutuna göre yeniden boyutlandırılır (image.resize) ve Tkinter ile uyumlu hale getirilir (ImageTk.PhotoImage). İşlenmiş kare, arayüz tarafından gösterilmek üzere bir sıraya (self.frame\_queue.put(imgtk)) eklenir. Döngü, kaynak serbest bırakılana kadar devam eder ve işlem sonunda video kaynağı kapatılır (cap.release()). Bu fonksiyon, nesne algılama sonuçlarını gerçek zamanlı olarak işleyen ve gösteren ana işlemi yürütür.

**def start\_detection(self):**start\_detection fonksiyonu, self.detection\_running bayrağını True olarak ayarlayarak nesne algılama işlemini başlatır. Bu bayrak, detect\_objects fonksiyonu içinde algılama işleminin etkin olup olmadığını kontrol etmek için kullanılır. Fonksiyonun çağrılmasıyla birlikte nesne algılama etkin hale gelir ve yeni karelerde YOLO modeliyle algılama yapılır. Bu fonksiyon, algılama işlemini başlatmak için basit bir tetikleyici görevindedir.

**def stop\_detection(self):**

**stop\_detection** fonksiyonu, **self.detection\_running** bayrağını **False** olarak ayarlayarak nesne algılama işlemini durdurur. Bu bayrak, **detect\_objects** fonksiyonu içinde algılama işleminin durdurulması gerektiğini kontrol etmek için kullanılır. Fonksiyon çağrıldığında, algılama devre dışı bırakılır ve alınan kareler üzerinde YOLO modeliyle herhangi bir algılama yapılmaz. Bu fonksiyon, algılama işlemini sonlandırmak için basit bir tetikleyici görevindedir.

### **Arayüz Bileşenleri**

1. **class App:**

metin, ekran görüntüsü, doküman, belge, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu**def \_\_init\_\_(self, root):**

App sınıfının \_\_init\_\_ metodu, Tkinter tabanlı bir uygulamanın arayüzünü oluşturur ve temel ayarlarını yapar. self.root, Tkinter ana pencere nesnesidir ve başlık "Akıllı Çöp Ayıştırma" olarak ayarlanır, ayrıca pencere simgesi icon.ico ile değiştirilir. Pencere boyutları 1280x800 piksel olarak ayarlanır ve ekranın ortasına konumlandırılır. Arka plan rengi #121212 olarak tanımlanır.

Bir Canvas bileşeni oluşturulur ve pencereyi dolduracak şekilde yerleştirilir; bu kanvas, kamera karelerinin ve algılama sonuçlarının görüntülenmesi için kullanılır. Pencerenin alt kısmına bir Frame eklenir ve bu çerçevede dört düğme (Open Camera, Start Detection, Stop Detection, Close) yerleştirilir. Bu düğmeler, create\_button fonksiyonu kullanılarak oluşturulur ve her bir düğmeye ilgili işlevler atanır.

Nesne algılama işlemi için gerekli olan self.model\_path (YOLO model dosyasının yolu), self.source (video kaynağı) ve self.frame\_queue (kare paylaşımı için iş parçacığı uyumlu bir sıra) değişkenleri tanımlanır. self.detector, algılama işlemlerini gerçekleştirecek olan YOLOv11ObjectDetection sınıfının bir örneği olarak kullanılmak üzere başlangıçta None olarak ayarlanır. Ayrıca, sürekli olarak kanvasın güncellenmesi için self.update\_canvas() fonksiyonu çağrılır. Bu yapı, uygulamanın arayüzünü başlatmak ve algılama işlemleri için gerekli bileşenleri hazırlamak için kullanılır.

metin, ekran, görüntüleme, ekran görüntüsü, bilgisayar içeren bir resim

Açıklama otomatik olarak oluşturuldu

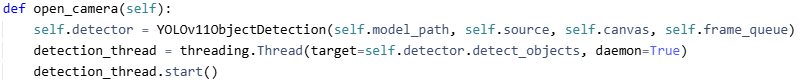
Görsel Arayüz görüntüsü

**def create\_button(self, parent, text, command, bg\_color, active\_bg):**

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturulducreate\_button fonksiyonu, Tkinter uygulaması için özel tasarımlı düğmeler oluşturur ve her düğme için hover (fareyle üzerine gelme) ve tıklama efektleri ekler. parent, düğmenin ait olacağı üst bileşeni ifade eder. text, düğme üzerinde görüntülenecek metindir. command, düğmeye tıklandığında çalıştırılacak işlevdir. bg\_color, düğmenin varsayılan arka plan rengini belirlerken, active\_bg, fare düğmenin üzerine geldiğinde değişecek olan arka plan rengini belirtir.

Düğme, Tkinter Button sınıfı kullanılarak oluşturulur ve özellikleri ayarlanır: metin yazı tipi Helvetica, 12, bold, düz bir görünüm için relief="flat", genişlik ve yükseklik ayarları ile beyaz yazı rengi belirlenir. Fareyle düğmenin üzerine gelindiğinde arka plan rengini değiştirmek için <Enter> ve düğmeden ayrıldığında rengi geri döndürmek için <Leave> olayları atanır. Son olarak, düğme, üst bileşen içine side=tk.LEFT yönlendirmesiyle yerleştirilir ve iki düğme arasına boşluk bırakmak için yatay dolgu (padx=10) eklenir. Bu fonksiyon, kullanıcı arayüzünde stilize edilmiş ve işlevselliği kolayca atanmış düğmeler oluşturmak için bir yardımcı görevindedir.

**def open\_camera(self):**

open\_camera fonksiyonu, kamera veya belirtilen video kaynağını açar ve gerçek zamanlı nesne algılama işlemini başlatır. Öncelikle, YOLOv11ObjectDetection sınıfının bir örneği (self.detector) oluşturulur; bu örnek, model dosyasının yolu (self.model\_path), görüntü kaynağı (self.source), sonuçların gösterileceği kanvas (self.canvas) ve karelerin sıraya ekleneceği self.frame\_queue gibi parametrelerle başlatılır.

Algılama işleminin ana döngüsü, ana uygulama arayüzünden bağımsız olarak çalışması için bir iş parçacığında başlatılır. threading.Thread, algılama işlemini yürüten self.detector.detect\_objects fonksiyonunu bir arka plan iş parçacığında çalıştırır. daemon=True, bu iş parçacığının ana program sonlandığında otomatik olarak kapanmasını sağlar.

Bu fonksiyon, kamera veya video kaynağını açarak algılama işlemine hazırlık yapar ve kullanıcı arayüzünün donmasını engellemek için işlemi ayrı bir iş parçacığında yürütür.

**def start\_detection(self):**

metin, yazı tipi, ekran görüntüsü, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldustart\_detection fonksiyonu, daha önce başlatılmış bir nesne algılama işleminde algılamayı aktif hale getirir. Öncelikle, self.detector nesnesinin var olup olmadığını kontrol eder; bu nesne, YOLOv11ObjectDetection sınıfının bir örneğidir ve kamera açıldığında oluşturulmuştur. Eğer self.detector mevcutsa, algılama işlemini başlatan start\_detection fonksiyonu çağrılır. Bu fonksiyon, algılama işleminin etkinliğini kontrol eden self.detection\_running bayrağını True olarak ayarlayarak, detect\_objects döngüsü içinde algılama işlemini aktif hale getirir. Bu şekilde, model her kare üzerinde nesne algılama yapmaya başlar. Fonksiyon, algılama işlemini başlatmak için bir tetikleyici görevindedir.

metin, yazı tipi, ekran görüntüsü, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu**def stop\_detection(self):**start\_detection fonksiyonu, daha önce başlatılmış bir nesne algılama işleminde algılamayı aktif hale getirir. Öncelikle, self.detector nesnesinin var olup olmadığını kontrol eder; bu nesne, YOLOv11ObjectDetection sınıfının bir örneğidir ve kamera açıldığında oluşturulmuştur. Eğer self.detector mevcutsa, algılama işlemini başlatan start\_detection fonksiyonu çağrılır. Bu fonksiyon, algılama işleminin etkinliğini kontrol eden self.detection\_running bayrağını True olarak ayarlayarak, detect\_objects döngüsü içinde algılama işlemini aktif hale getirir. Bu şekilde, model her kare üzerinde nesne algılama yapmaya başlar. Fonksiyon, algılama işlemini başlatmak için bir tetikleyici görevindedir.

**def close\_application(self):**

metin, yazı tipi, ekran görüntüsü, beyaz içeren bir resim

Açıklama otomatik olarak oluşturulduclose\_application fonksiyonu, uygulamayı düzgün bir şekilde kapatmak için tasarlanmıştır. Öncelikle, self.detector nesnesinin var olup olmadığını kontrol eder; bu nesne, nesne algılama işlemini yöneten YOLOv11ObjectDetection sınıfının bir örneğidir. Eğer bu nesne mevcutsa, algılama işlemini durduran stop\_detection fonksiyonu çağrılarak algılama işlemi sonlandırılır. Daha sonra, self.root.quit() çağrısıyla Tkinter ana döngüsü sonlandırılır ve uygulama kapatılır. Bu fonksiyon, algılama işlemini durdurup kaynakları serbest bırakarak uygulamayı düzgün bir şekilde kapatmak için bir çıkış mekanizması sağlar.

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu**def update\_canvas(self):**update\_canvas fonksiyonu, Tkinter arayüzünde kameradan alınan veya işlenmiş kareleri sürekli olarak günceller ve görüntüler. Öncelikle, self.frame\_queue.empty() kontrol edilerek sırada görüntü olup olmadığına bakılır; eğer kare varsa, self.frame\_queue.get() ile sıradan alınır. Alınan kare, self.canvas.create\_image ile kanvasa yerleştirilir ve görüntünün kaybolmasını önlemek için self.canvas.imgtk üzerinden referans atanır. Eğer bir hata oluşursa, hata mesajı konsola yazdırılır. Daha sonra, self.root.after(20, self.update\_canvas) ile fonksiyon her 20 milisaniyede bir yeniden çağrılarak düzenli bir güncelleme sağlanır. Bu yapı, kameradan veya nesne algılama işlemlerinden gelen kareleri gerçek zamanlı olarak Tkinter arayüzünde göstermeyi sağlar.

metin, yazı tipi, çizgi, beyaz içeren bir resim

Açıklama otomatik olarak oluşturuldu

Eğer bu dosya doğrudan çalıştırılırsa (if \_\_name\_\_ == "\_\_main\_\_" kontrolü), önce tk.Tk() ile bir Tkinter ana pencere nesnesi oluşturulur ve root değişkenine atanır. Daha sonra, bu root nesnesi, uygulama sınıfının bir örneği olan App sınıfına geçirilir ve app adında bir nesne oluşturulur. Bu, uygulamanın arayüzünün ve işlevlerinin oluşturulmasını sağlar. Son olarak, root.mainloop() çağrılarak Tkinter’in olay döngüsü başlatılır; bu döngü, kullanıcı etkileşimlerini algılar ve uygulamanın sürekli çalışmasını sağlar. Bu yapı, Tkinter tabanlı bir masaüstü uygulamasını başlatmak için temel bir kod yapısıdır.

### **Karşılaşılan Zorluklar ve Çözümleri**

Proje geliştirme sürecinde hem teknik hem de donanımsal çeşitli zorluklarla karşılaşıldı. Bu zorluklar ve bunlara yönelik geliştirilen çözümler şu şekilde özetlenebilir:

**1. Yetersiz Veri ve Veri Çeşitliliği**

Zorluk: Projenin başlangıcında kullanılan veri seti, atık türlerini tanımlamak için yeterince çeşitli ve geniş değildi. Veri setindeki yetersizlik, modelin genelleme kabiliyetini sınırlandırdı ve doğruluk oranını düşürdü.

Çözüm: Farklı ışık, açı ve materyal türlerini içeren yeni görüntüler toplandı ve mevcut veri artırma teknikleri (döndürme, parlaklık değişikliği, ölçeklendirme) kullanılarak veri seti genişletildi. Bu sayede modelin çeşitli koşullarda daha iyi performans göstermesi sağlandı.

**2. Işık, Açı ve Renk Farklılıklarının Etkisi**

Zorluk: Düşük ışık koşulları, açılar veya benzer renk tonlarına sahip atık türleri, modelin doğru sınıflandırma yapmasını zorlaştırdı.

Çözüm: Görüntüler üzerinde histogram eşitleme ve gama düzeltmesi gibi ön işleme teknikleri uygulandı. Ayrıca, veri setine farklı ışık ve açılarda çekilmiş görüntüler eklenerek modelin genelleme kapasitesi artırıldı.

**3. Donanım Kısıtlamaları**

Zorluk: Eğitim süreci okul bilgisayarlarının sınırlı işlem gücü ve bellek kapasitesi nedeniyle uzun sürdü. Düşük donanım kapasitesi, eğitim sürelerini uzatarak çalışmayı sınırladı.

Çözüm: Model boyutunu küçültmek için daha verimli algoritmalar kullanıldı ve paralel işleme teknikleri ile süreç hızlandırıldı. Ayrıca, hafif model mimarileri üzerinde çalışılarak donanım sınırlamaları minimize edildi.

**4. Eğitim Sürecinde Tekrarlar ve Optimizasyon**

Zorluk: Model, ilk eğitim denemelerinde beklenen doğruluk oranına ulaşamadı. Bu, sürecin birkaç kez tekrarlanmasını gerektirdi.

Çözüm: Modelin hiperparametreleri (örneğin, öğrenme hızı, optimizasyon algoritmaları) optimize edildi. Eksik veri türleri belirlenerek bu türlere yönelik yeni veriler toplandı ve model yeniden eğitildi. Sonuçta, daha yüksek doğruluk oranlarına ulaşıldı.

**5. Webcam ve Harici Kamera Kullanımı**

Zorluk: Projenin geliştirilme aşamasında okul bilgisayarının yerleşik web kamerası kullanıldı. Ancak, bu kameranın düşük çözünürlüğü ve ışık duyarlılığı, görüntü kalitesini ve modelin performansını etkiledi.

Çözüm: Webcam kullanılarak temel sistem işlevselliği başarıyla test edildi. Proje teslim videosunda ise daha yüksek çözünürlüklü bir harici kamera kullanılarak sistemin performansı daha iyi bir şekilde sergilendi. Harici kamera, netlik ve çözünürlük açısından büyük avantaj sağladı.

**6. Karmaşık ve Tanımlanamayan Atıklar**

Zorluk: Bazı atıklar karmaşık yapıları nedeniyle doğru sınıflandırılamadı ve “tanımlanamayan” olarak işaretlendi.

Çözüm: Tanımlanamayan atıkların kaydedilmesi ve geri bildirim mekanizmalarıyla modelin iyileştirilmesi sağlandı. Ayrıca, bu tür atıklar için yeni alt modeller geliştirilmesi planlandı.

**Sonuç**

Proje sürecinde karşılaşılan zorluklar, çözüm odaklı yaklaşımlar ile ele alındı. Veri çeşitliliği artırıldı, donanımsal sınırlamalara yönelik optimizasyonlar yapıldı ve model eğitim süreci tekrarlanarak daha yüksek doğruluk oranlarına ulaşıldı. Webcam ve harici kamera kullanımı ile farklı donanım koşullarında sistemin performansı test edildi ve sistemin potansiyeli daha iyi ortaya kondu. Bu süreç hem teknik hem de ekip çalışması açısından önemli kazanımlar sağladı.

## **TESTLER VE SONUÇLAR**

### **Test Tablosu**

**Tablo4: Test**

|  |  |
| --- | --- |
| **Test Senaryosu** | **Sistem Cevabı** |
| Sisteme plastik atıkların gösterilmesi | Sistem kameraya tutulan nesneyi analiz ederek "Plastik" sınıfını belirler. |
| Sisteme metal atıkların gösterilmesi | Sistem kameraya tutulan nesneyi analiz ederek "Metal" sınıfını belirtir. |
| Sisteme kâğıt atıkların gösterilmesi | Sistem kameraya tutulan nesneyi analiz ederek "Kağıt" sınıfını belirtir. |
| Sisteme cam atıkların gösterilmesi | Sistem kameraya tutulan nesneyi analiz ederek "Cam" sınıfını belirtir. |
| Sisteme bilinmeyen türde atık gösterilmesi | Sistem kameraya tutulan nesneyi analiz ederek "Undefined" olduğunu belirtir. |
| Sistemin kamerayı algılayamaması | Sistem kameranın algılanamadığı uyarısını verir. |

Proje kapsamında geliştirilen sistemin performansı, belirli test senaryoları ile değerlendirilmiştir. Yapılan testlerde, kameraya tutulan nesnelerin türüne göre sistemin doğru sınıflandırma yapıp yapmadığı kontrol edilmiştir. Plastik, metal, kâğıt ve cam gibi yaygın atık türleri sistem tarafından başarıyla tanınmış ve her bir atık türü için doğru sınıf belirlenmiştir. Sistem, nesnenin türünü belirledikten sonra kullanıcıya ilgili atık türünü ekranda göstermiştir.

Bilinmeyen veya veri setinde yer almayan türde bir atığın gösterilmesi durumunda, sistem nesneyi "Undefined" olarak işaretlemiş ve sınıflandıramamıştır. Bu senaryoda, sistemin kullanıcıya bir uyarı vermesi sağlanarak doğru kullanımın teşvik edilmesi hedeflenmiştir. Ayrıca, sistemin kamera bağlantısını kontrol etme işlevi de test edilmiş ve kameranı algılanmadığı durumlarda uygun bir hata mesajı verilmiştir.

### **Eğitim Sonuçları**

1. **Confusion Matrix (Ham Sayılarla)**

ekran görüntüsü, dikdörtgen, kare, diyagram içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil Confusion Matrix (Ham Sayılarla)

**Ne gösterir?**

* Bu matris, modelinizin **altı sınıf** (glass, metal, paper, plastic, undefined, background) için kaç örneği doğru ya da yanlış etiketlediğini gösterir.
* Satırlar (y ekseni) genelde **modelin verdiği tahmini** (Predicted), sütunlar (x ekseni) ise **gerçek sınıfı** (True) temsil eder.
* Diyagonal (kare biçimindeki ana köşegen) üzerindeki sayılar, doğru tahmin edilen örneklerin sayısını gösterir. Diyagonelin dışındaki her hücre, o iki sınıf arasında yaşanan karışıklığın (confusion) sayısını temsil eder.

**Nasıl yorumlanmalı?**

* Örneğin, glass (1. satır) ve glass (1. sütun) kesişiminde gördüğümüz **413**, gerçek sınıfı glass olan nesnelerin 413 tanesini doğru tahmin ettiğinizi gösteriyor.
* Bazı durumlarda model, gerçekte metal veya paper olan nesneleri glass gibi tahmin etmiş olabilir (matrisin aynı satırındaki farklı sütunlara bakarak görebilirsiniz).
* Satır içindeki en büyük sayının köşegende olması (örneğin metal satırındaki 194, paper satırındaki 403, plastic satırındaki 286 gibi) sınıfın genel olarak doğru şekilde tahmin edildiğini gösterir. Ancak sağ üst kısımda undefined ve background sınıflarında da göreceğiniz üzere karışıklıklar (yüksek sayılar) varsa, modelin bu sınıflar arasında çokça hata yapabildiğini ifade eder.

Kısaca, Confusion Matrix ham sayıları bize hangi sınıfın nerede iyi ya da kötü tahmin edildiğini ve hangi sınıflar arasında en çok karışıklık olduğunu açıkça gösterir.

1. **Confusion Matrix (Normalize Edilmiş)**

ekran görüntüsü, dikdörtgen, kare, diyagram içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil Confusion Matrix (Normalize Edilmiş)

**Ne gösterir?**

* Bu matris, yukarıdaki Confusion Matrix’in **oransal** (normalize) halidir.
* Her bir hücredeki değer, o hücrenin satır toplamına bölünerek “% olarak” ifade edilir (veya bazı durumlarda sütun toplamına bölünür; burada görsele bakarsak satır bazında normalizasyon yapılmış gibi görünüyor).
* 0.92, 0.75, 0.89 gibi oranlar, tahmin edilen sınıf ile gerçek sınıfın ne oranda eşleştiğini gösterir.

**Nasıl yorumlanmalı?**

* Örneğin, glass satırında 0.92 değeri, “modelin glass olarak tahmin ettiği nesnelerin %92’sinin gerçekten de glass olduğu” ya da “gerçekten glass olan nesnelerin %92’sini doğru yakaladığı” şeklinde (matrisi nasıl normalize ettiğinize bağlı) yorumlanabilir.
* Modelin hangi sınıfı hangi oranda karıştırdığını yüzdesel olarak görmek, hangi sınıfların daha çok eğitime veya veri artırmaya ihtiyaç duyduğunu daha net gösterir.
* Örneğin undefined satırı ve background sütunundaki 0.56 değeri (koyu maviye yakınsa), “gerçekte background olan verilerde model undefined demeye eğilimli” veya tersi bir karışıklık olduğu anlamına gelir.

Normalize Confusion Matrix, genel performansı **yüzdelik** olarak görmeyi sağladığı için ham matristen daha hızlı yorumlanır.

1. **F1-Confidence Curve**

metin, diyagram, çizgi, öykü gelişim çizgisi; kumpas; grafiğini çıkarma içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil F1-Confidence Curve

**Ne gösterir?**

* Şekil 3’teki grafikte x ekseninde **modelin tahmin için kullandığı güven eşiği (confidence threshold)**; y ekseninde ise **F1 skoru** var.
* F1 skoru, modelin **hassasiyet (precision)** ve **duyarlılık (recall)** değerleri arasındaki dengeyi ölçen bir metrik.
* Farklı renkli eğriler, her bir sınıf (glass, metal, paper, plastic, undefined) için F1 değerlerinin, ilgili confidence eşiğine göre nasıl değiştiğini gösteriyor.
* Kalın mavi çizgi, tüm sınıfların **ortalama F1** değeri. Grafikte “all classes 0.81 at 0.332” notu, modelin **0.332** confidence eşiğinde, ortalamada **0.81** F1 skoruna ulaştığını anlatıyor.

**Nasıl yorumlanmalı?**

* Confidence eşiğini yükselttikçe model sadece daha emin olduğu nesneleri tahmin olarak verir (bu genelde precision’ı arttırıp recall’ı düşürebilir).
* Grafik, hangi eşiğin genelde en yüksek F1’e karşılık geldiğini gösterir.
* Her sınıfın eğrisindeki değişimleri gözlemleyerek, modelin hangi eşiğe kadar istikrarlı olduğu, hangi eşiğin üzerinde hızla F1 değerlerinin düştüğü anlaşılabilir.
* Buradaki amaç, projenize en uygun dengeyi yakalayacak **ideal eşiği (threshold)** seçmektir.

1. **Veri Dağılımı ve Bounding Box Dağılımları**

metin, ekran görüntüsü, renklilik, diyagram içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil Grafik Kolaj

Şelik 4 görsel dört farklı alt kısımdan oluşuyor:

1. **Sol üstteki çubuk grafiği**:
   1. Her bir sınıfa ait veri adedini (instance sayısını) gösterir.
   2. Örneğin glass verisi yaklaşık 4000, metal verisi yine 4000 civarı, paper biraz daha az (~3500), plastic ve undefined ise ona yakın miktarda şeklinde bir dağılım gözüküyor.
   3. Bu tablo, veri setindeki dengesizlik veya dengesizlik risklerini görmenizi sağlar.
2. **Sağ üstteki turkuaz çizgilerle dolu kare**:
   1. Bu kare, **Bounding Box** çizimlerinin bir “ısıl haritası” veya “topluca gösterimi” olabilir.
   2. Görselde her bir box’ın görüntü içindeki konumu (x,y) ve boyutu (width,height) üst üste bindirilmiş gibi görünüyor.
   3. Merkeze doğru çizgilerin yoğunlaştığı alanlar, nesnelerin (özellikle metal, glass vb.) daha sık ortaya çıktığı, yani bounding box’ların merkezlendiği bölgeleri işaret edebilir.
3. **Alt soldaki x-y grafiği**:
   1. Burada x ve y eksenleri 0 ile 1 arasında ölçeklenmiş ve bounding box’ların **merkezi** veya sol-üst köşesi (hangi koordinat kullanıldıysa) dağılımı gösteriliyor.
   2. Ortada yoğunlaşma görüyorsanız, bu demek oluyor ki nesnelerin çoğu bu koordinat bölgesinde bulunuyor.
   3. Bazı kenarlarda az sayıda verinin olması, o bölgelerde nesne az çıktığını anlatabilir.
4. **Alt sağdaki width-height grafiği**:
   1. Bu da her bir bounding box’ın **genişlik (width)** ve **yükseklik (height)** oranlarını 0–1 aralığında gösteriyor.
   2. 1.0’a yaklaşan bir width veya height, neredeyse tam ekran boyutunda bir bounding box olduğunu ifade eder.
   3. Grafikte genişlik ve yükseklik dağılımına bakarak, ortalamada nesnelerin ne kadar büyük/küçük etiketlendiğini anlayabilirsiniz.

Kısaca bu son görsel, veri setinizin farklı sınıflardaki **miktar dağılımını** ve **etiketlenmiş bounding box’ların konum ve boyut dağılımını** bir arada sunuyor. Modelin öğrenme süreci ve hatalarının kaynakları hakkında fikir sahibi olmak için bu tür görseller oldukça faydalıdır.

Bu görseller birlikte incelendiğinde:

* **Confusion Matrix’ler** yardımıyla hangi sınıflarda modelin daha başarılı/başarısız olduğunu görebilirsiniz.
* **Normalize Confusion Matrix**, ham matrisin yüzdesel görünümünü sunarak hızlı kıyaslama yapmayı kolaylaştırır.
* **F1-Confidence Curve**, belli bir confidence eşiğinde nasıl bir F1 performansı aldığınızı ve hangi eşiğin en verimli olduğunu gösterir.
* **Veri dağılımı grafikleri**, veri setinizdeki dengesizlikleri ve bounding box’ların konum/boyut dağılımını anlamanıza imkân tanır.

Raporunuza bu açıklamaları ekleyerek, okuyucuların hem model performansını hem de veri setinin yapısını net bir şekilde anlamasını sağlayabilirsiniz.

1. **Bounding Box Özellikleri Dağılımı (Pairplot)**

diyagram, metin, tasarım içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil Bounding Box

Şekil 5’teki görsel, veri setinizdeki bounding box’lara ait dört özelliği birlikte göstermeyi amaçlayan bir **scatter matrix** (ya da “pairplot”) şeklinde görünüyor:

* **x**: Box’ın genellikle merkez veya sol-üst köşesinin x koordinatı (0 ile 1 arasında normalize).
* **y**: Box’ın y koordinatı (0 ile 1 arasında normalize).
* **width**: Box’ın genişliği (0 ile 1 arasında normalize).
* **height**: Box’ın yüksekliği (0 ile 1 arasında normalize).

**Nasıl okunur?**

* Diyagonaldeki histogramlar, ilgili özelliğin (örneğin x, y, width, height) tek boyutlu dağılımını gösteriyor.
* Diğer hücrelerde ise iki değişken arasındaki dağılımın saçılım (scatter) grafiği bulunuyor. Örneğin, x ile y’nin ilişkisi, width ile height’in ilişkisi vb.
* Her bir grafik, veri setinizde bounding box’ların nerede yoğunlaştığını, konum ve boyut olarak hangi değerlerin daha sık olduğunu gösterir.

**Nasıl yorumlanmalı?**

* X ve Y histogramlarında ortada belirgin bir pik (tepe) varsa, bounding box’ların merkezlerinin çoğunlukla görüntünün ortalarına yakın olduğunu veya annotation yaparken genelde resmin orta kısımlarında nesnelerin etiketlendiğini düşünebilirsiniz.
* Width ve height dağılımlarında 0’a yakın ve 1’e yakın değerlerin çokluğu, bazı nesnelerin çok küçük (ör. width ~ 0.05 gibi) veya neredeyse tüm resmi kaplayacak kadar büyük (ör. width ~ 0.9) olabildiğini gösterir.
* İki değişkenin birlikte incelendiği alt grafiklerde, örneğin width–height üçgeni, veri setinde sık görülen boyut oranlarını işaret eder. Eğer net bir üçgen formu görüyorsanız, bazı en-boy oranlarının daha yoğun olduğunu veya annotation aşamasında belli bir kalıp olduğunu anlayabilirsiniz.

Bu tür bir dağılım analizi, modelin hangi tür box’larda zorlanabileceğini veya veri setindeki çeşitliliğin (nerede çok, nerede az) nasıl olduğunu anlamanız açısından çok değerli.

1. Precision–Confidence Curve

metin, diyagram, ekran görüntüsü, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil Precision–Confidence Curve

Şekil 6’ daki grafikte:

* **X ekseni**: Modelin tahmin için kullandığı **güven eşiği (confidence threshold)**.
* **Y ekseni**: İlgili confidence değerinde elde edilen **precision** (kesinlik).
* **Renkli eğriler**: Her bir sınıf için precision değerinin, confidence threshold’a göre değişimi. (glass, metal, paper, plastic, undefined)
* **Kalın mavi çizgi**: Tüm sınıfların ortalama precision değeri. Grafikte “all classes 1.00 at 0.999” şeklinde bir not var. Bu da, **0.999** confidence eşiği kullanılırsa, ortalama (ya da toplam) precision değerinin **1.00** olduğunu gösteriyor (yani bu kadar yüksek eşikte model hata yapmamış ya da hata çok düşük).

**Nasıl yorumlanmalı?**

* Confidence threshold arttıkça, model “sadece en emin olduğu tahminleri” output olarak vermeye başlar. Bu sayede **precision** (doğru tahmin yüzdesi) yükselir, ancak model tahmin yapmaktan çekindiği için **recall** genelde düşmeye başlar (grafikte recall’ı görmüyoruz ama bu tipik davranıştır).
* Eğrinin sol tarafında, threshold çok düşükken model birçok tahmin çıkarır; bu sırada yanlış tahminler de artabileceğinden precision düşüktür.
* Sağ tarafa, eşik değeri ~0.9’a yaklaştıkça, precision değerlerinin çoğunun 0.9–1.0 civarına çıktığını görürsünüz.
* Belirli bir uygulamada hangi eşiğin size uygun olduğunu karar verirken, precision mi yoksa recall mu daha kritik, bu soruya cevap vererek hareket edebilirsiniz.

1. **Precision–Recall Curve**

metin, diyagram, çizgi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil Precision–Recall Curve

Şekil 7’ deki grafikte:

* **X ekseni**: **Recall** (duyarlılık) — Gerçek pozitiflerin ne kadarını yakalayabildiğiniz.
* **Y ekseni**: **Precision** (kesinlik) — Tahmin ettiğiniz pozitiflerin ne kadarının gerçekte doğru olduğu.
* **Renkli eğriler**: Her bir sınıf (glass, metal, paper, plastic, undefined) için precision ve recall’un değişim eğrisi.
* **Kalın mavi çizgi**: Tüm sınıfların ortalaması. Üzerinde “all classes 0.851 [mAP@0.5](mailto:mAP@0.5)” ifadesi var. Yani IoU eşiğinin 0.5 alındığı durumda, tüm sınıflar için **ortalama ortalama duyarlılık (mAP)** 0.851 değerine ulaşmış.

**Nasıl yorumlanmalı?**

* Eğrinin sol üst köşesi, genelde precision’ın oldukça yüksek, recall’ın da orta-yüksek olduğu bölgeler.
* Eğrinin sağ tarafında recall arttıkça, genelde precision düşer; çünkü modele “daha fazla nesne algıla” demek, yanlış pozitifleri de artırabilir.
* Her bir sınıf için görmüş olduğunuz değerler (ör. glass 0.948, metal 0.809 vb.), o sınıfın **ortalama hassasiyet (AP)** değerleridir. Yani tüm recall seviyeleri boyunca ortalama precision puanı.
* glass ve paper gibi sınıflar neredeyse 0.93–0.95 civarında AP değerine ulaşırken, metal (0.809) ve özellikle undefined (0.642) biraz daha düşük kalmıştır. Bu da bu sınıflarda modelin daha çok hata yaptığını (daha düşük precision-recall eğrisi) gösterir.

Bu tip **precision-recall** eğrileri, özellikle **dengesiz veri setlerinde** (bazı sınıflar az, bazıları çok) model performansını değerlendirirken **ROC** eğrisine göre daha anlamlı olabilmektedir.

1. **Recall–Confidence Curve**

metin, diyagram, çizgi, öykü gelişim çizgisi; kumpas; grafiğini çıkarma içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil Recall–Confidence Curve

Şekil 8’deki grafikte:

* **X ekseni**: **Confidence threshold** (tahmin eşiği).
* **Y ekseni**: **Recall** (duyarlılık).
* **Renkli eğriler**: Yine her bir sınıf (glass, metal, paper, plastic, undefined) için recall’un, confidence eşiğine göre nasıl değiştiğini gösteriyor.
* **Kalın mavi çizgi**: Tüm sınıfların ortalama recall değeri. Grafikte “all classes 0.95 at 0.000” şeklinde bir not var. Yani eşiği 0’a (modelin en ufak bir güveni bile “pozitif” kabul ettiği durum) çektiğinizde, ortalama recall 0.95’e ulaşmış demek. Ancak bu noktada precision’ın muhtemelen çok düşük olacağını da unutmamak gerek.

**Nasıl yorumlanmalı?**

* Threshold düşükken (0.0–0.1 aralığında), model ne kadar emin olursa olsun nesneleri “pozitif” diye etiketler, dolayısıyla **recall** yüksek olur. Ama bu aşamada precision düşer (çünkü yanlış pozitifler de artar).
* Eşik değeri arttıkça, recall genelde düşmeye başlar, çünkü artık model yalnızca çok emin olduğu örnekleri pozitif olarak etiketler. Bazı gerçek pozitifleri ise düşük confidence yüzünden kaçırabilir (false negatif sayısı artar).
* Grafikte undefined sınıfının (mor eğri) recall değerinin threshold arttıkça sert düşmesi, bu sınıfı doğru tespit edebilmek için modelin çok da emin olamadığını (ya da verisetindeki karmaşıklık nedeniyle bu sınıfta yanlış negatiflerin arttığını) gösterir.
* Geriye dönük olarak, undefined sınıfında modelin düşük performansı, önceki Precision–Recall eğrisinde ve Confusion Matrix’lerde de dikkat çekmişti. Bu tip sınıfları düzeltmek için veri miktarını artırma, etiketlemeyi iyileştirme gibi yöntemler düşünülebilir.

**Genel Özet ve Birleştirilmiş Yorum**

* **Bounding Box Özellikleri Dağılımı (Pairplot):** Veri setindeki etiketlenmiş nesnelerin resim içi konumları ve boyutlarının nasıl dağıldığını gösterir. Bu, annotation kalitesi ve veri çeşitliliği hakkında fikir verir.
* **Precision–Confidence Curve:** Eşik değeri arttıkça precision’ın nasıl değiştiğini gösterir. Uygulamanızda hataya ne kadar tolerans göstereceğinize bağlı olarak optimum eşiği seçmenizde yardımcı olur.
* **Precision–Recall Curve:** Modelin her bir sınıfta (ve ortalama olarak) “pozitif tespit başarısı”nı (recall) artırdıkça, “yanlış tespit oranı”nın (precision) nasıl değiştiğini gösterir. mAP (mean Average Precision) değeri, farklı eşiğe sahip senaryolardaki ortalama başarıyı özetler.
* **Recall–Confidence Curve:** Düşük threshold’larda modelin çok fazla nesne tespit ederek recall’ı yüksek tuttuğunu, ancak threshold yükseldikçe recall’ın düştüğünü görürüz. Bu da yine optimum eşiği belirlemede diğer grafiklerle birlikte değerlendirilir.

Bu dört görselin tamamı, hem veri setindeki etiket dağılımını hem de modelinizin performans metriklerini (precision, recall, F1, mAP) farklı perspektiflerden ele alır. Raporunuzda bu grafikleri bir arada sunarak okuyuculara, modelin hangi ayarlarla (özellikle confidence threshold) nasıl bir performans sergilediğini gösterebilir, ayrıca veri seti içeriğini ve olası iyileştirme alanlarını daha net ifade edebilirsiniz.

### **Eğitim ve Doğrulama (Validation) Kayıpları ve Metrikleri**

metin, diyagram, plan, harita içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil Eğitim Metrikleri

Şekil 9 kolaj hâlinde sunulmuş sekiz adet grafik ve iki metrik grafiğinden (mAP50, mAP50-95) oluşuyor. Genellikle **YOLO veya benzeri bir obje tespiti (object detection) modelinin** eğitim sürecinde elde edilen **kayıp (loss) ve metrik** değerlerini zaman (epoch) bazında takip etmek için oluşturulan bir görsel düzen.

1. **train/box\_loss**
   1. Modelin **bounding box (konum, boyut) tahmini** sırasında oluşan hata miktarını (loss) gösteriyor.
   2. Eğitimin başında (0.9 civarı) yüksek olan kayıp değerinin, epoch’lar ilerledikçe 0.3’e kadar düştüğünü görüyoruz. Bu, modelin nesne konumlamada giderek daha iyi hale geldiğini ifade eder.
2. **train/cls\_loss**
   1. Modelin **sınıflandırma (class) tahmininde** (hangi etiket olduğuna dair) yaptığı hatayı ölçüyor.
   2. Başlangıçta 1.75 civarlarında olan loss, 0.5–0.6 seviyelerine kadar geriliyor. Yani model, eğitim verisinde hangi sınıfın hangi obje olduğunu giderek daha iyi öğrenmiş.
3. **train/dfl\_loss** (Distribution Focal Loss)
   1. YOLOv5/YOLOv8 gibi modellerde kullanılan, konum bilgisini iyileştirmeyi hedefleyen bir ek kayıptır.
   2. 1.4 seviyelerinden 1.0’a kadar düşerek modelin kutu (box) lokalizasyonunda ve skor tahmininde daha da iyileştiğini gösteriyor.
4. **metrics/precision(B)**
   1. Eğitimin her epoch’unda **ortalama precision** değeriniz nasıl değişiyor? (Burada (B) muhtemelen “box” ya da belirli bir model ismini ifade ediyor.)
   2. 0.6–0.8 arasında seyredip en sonunda 0.8 üstüne oturuyor. Bu, tahmin edilen nesnelerin çoğunlukla doğru sınıfa ait olduğunu, yanlış pozitiflerin sınırlı kaldığını gösterir.
5. **metrics/recall(B)**
   1. Modelin **recall** değerini (gerçekte var olan nesnelerin ne kadarını tespit ettiğini) epoch bazında takip ediyorsunuz.
   2. İlk epoch’ta 0.5 civarında başlarken hızla 0.8–0.9 seviyelerine yükselmiş ve stabil hâle gelmiş. Yani model çoğu nesneyi yakalamayı öğrenmiş.
6. **val/box\_loss, val/cls\_loss, val/dfl\_loss**
   1. Yukarıdaki **train** kayıplarıyla aynı metriklerin **doğrulama (validation) veri seti** üzerindeki karşılıklarını gösterir.
   2. Eğitimin başında daha yüksek olan kayıplar, epoch’lar ilerledikçe düşüyor ve **train** ile **val** eğrileri birbirine yakın seyrediyorsa, **aşırı öğrenme (overfitting)** riskinin düşük veya iyi yönetilmiş olduğunu gösterir.
7. **metrics/mAP50(B)**
   1. Nesne tespiti modelinde en sık kullanılan metriklerden biridir. **mAP (mean Average Precision)**, farklı sınıflar için hesaplanan **AP** değerlerinin ortalamasıdır.
   2. Burada “0.50” ifadesi, **IoU eşiğinin %50** olarak alındığı anlamına gelir. Yani gerçek bounding box ile tahmin kutusu arasındaki kesişim/birleşim oranı (IoU) %50 veya üstü ise “doğru tahmin” sayılır.
   3. Grafiğin 0.6 seviyelerinden başlayıp 0.85–0.9 civarında doruğa ulaştığını görüyorsunuz, bu da modelinizin performansının eğitimle birlikte önemli ölçüde arttığını gösteriyor.
8. **metrics/mAP50-95(B)**
   1. Burada mAP, **IoU eşiklerini 0.5’ten 0.95’e kadar (genellikle 0.05’lik artışlarla)** ortalayarak hesaplanır.
   2. Bu metrik, modelin sadece %50 IoU değil, daha sıkı eşiklerle de (örneğin %75, %90) ne kadar başarılı tespit yaptığını gösterir.
   3. Grafikte 0.3 civarından 0.7’nin üzerine çıkıyor olması, modelin oldukça iyi konum tahminine ulaştığının (dar IoU aralıklarında bile tutarlı tahmin yaptığının) bir işaretidir.

Özetle, bu ilk görsel topluluğu, **eğitim süreci** boyunca model kayıplarının istikrarlı bir şekilde azaldığını ve doğrulama (val) sonuçlarının da buna paralel ilerlediğini; ayrıca **precision, recall, mAP50 ve mAP50-95** değerlerinin zamanla yükselip belirli bir platoda (kararlı seviyede) seyrettiğini göstermektedir. Bu, iyi bir eğitim performansı ve modelin doğru yönde öğrendiğine dair güçlü bir işarettir.

### **Nesne Tespiti Program Çıktıları**

1. **Nesne Tespiti Sonuç Görselleri (Kolajlar)**

kolaj, ekran görüntüsü, fotomontaj, çizgi film içeren bir resim

Açıklama otomatik olarak oluşturuldu

devre, iç mekan içeren bir resim

Açıklama otomatik olarak orta güvenilirlik düzeyiyle oluşturuldukolaj, ekran görüntüsü, mozaik, moda içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil Girdi resim tahminleri

Şekil 10’daki görsel kolaj, eğitilmiş modelin **girdi resimler üzerindeki tahminlerinin** nasıl göründüğünü gösteriyor. Her bir küçük resim parçasında:

* Mavi veya turkuaz renkli **bounding box’lar** (kutu çerçeveler) var.
* Kutuların üzerinde **sınıf etiketleri** ve genellikle tahmin skorları (confidence) bulunuyor. Örneğin:
  + 0 (glass)
  + 1 (metal)
  + 2 (paper)
  + 3 (plastic)
  + 4 (undefined)
  + Veya kullandığınız veri setine göre farklı sınıf numaraları
* Her bir kare, tek bir görüntüden ya da verisetinin içinden alınmış bir kesitten modelin tespit ettiği nesneleri ve bu nesnelere ait tahmin etiketlerini gösterir.

**Nasıl yorumlanmalı?**

* Eğer model, görselde bir **kutu** çizmiş ve üzerinde doğru sınıf etiketini (örneğin metal) göstermişse, bu tahmin büyük olasılıkla doğrudur.
* Bir karede birden fazla kutu olabilir; model, **birbirine yakın ya da benzer** nesneleri de ayırt edebildiyse başarılı bir genel performanstan söz edilebilir.
* Yanlış etiketlemeler veya hiç tespit edilememiş nesneler varsa, bunlar modelin zayıf noktalarına işaret edebilir. Bu tür kolaj görsellerde, nerede hata yapıldığını görsel olarak tespit etmek oldukça kolaydır.

**Dikkat edilecek noktalar:**

* Bazı nesneler tam olarak çerçevelenmemiş veya sınıfı yanlış etiketlenmiş olabilir. Bunları not alarak, modelin hangi tip durumlarda yanıldığını anlamak mümkündür (örneğin parlak ışık, gölgeler, nesnelerin yıpranmışlığı vb.).
* Çerçevelerin çok fazla üst üste binmesi (çoklu tespitler) modelin aşırı tespit (over-detection) yaptığını gösterebilir.
* Skor değerleri (confidence) oldukça yüksekse (örneğin 0.8–0.9 üstü), modelin tahmini için güven oranı da yüksektir.

Bu görseller, projenizdeki **gerçek çıktı örneklerini** göstererek, okuyucuların “Model gerçekten çalışıyor mu?”, “Görüntüler üzerindeki tahminler nasıl görünüyor?” gibi sorularına cevap vermek adına oldukça faydalı olacaktır.

ekran görüntüsü, kolaj, kompakt disk içeren bir resim

Açıklama otomatik olarak oluşturulduekran görüntüsü, çizgi film, kolaj, kompakt disk içeren bir resim

Açıklama otomatik olarak oluşturuldu

ekran görüntüsü, kolaj, metin içeren bir resim

Açıklama otomatik olarak oluşturulduekran görüntüsü, kolaj, mozaik içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil Çıktı örnekleri

Şekil 11’ deki görseller, yine modelin çöp kutusu içerisindeki farklı atıkları (metal, plastik, cam vb.) nasıl tespit ettiği ve bazı nesneleri “undefined” olarak etiketlediğini gösteriyor. Bu kolajlar, **aynı veya benzer sahnenin** (örneğin içinde atık bulunan bir çöp kutusu) farklı **confidence eşiği**, **tahmin ayarı** veya **tahmin aşaması**yla işlenmiş çıktıları olabilir. Her bir karede modelin:

* **Bounding box** çizdiği nesne veya nesneleri,
* **Sınıf etiketini** (metal, plastic, glass, undefined),
* **Tahmin güven skorunu** (0.3, 0.6, 0.8 vb.),

yansıttığını görüyorsunuz.

1. “undefined” Etiketleri

* Görsellerin çoğunda “undefined” etiketli kutular var. Bu, modelin veritabanındaki sınıflardan herhangi birine (metal, plastic, paper, glass vb.) tam olarak uymayan veya net tanımlayamadığı nesneleri “undefined” olarak işaretlediği anlamına geliyor.
* Ayrıca, bazı karelerde “undefined” etiketine verilen confidence değerlerinin (örn. 0.3, 0.4, 0.5, 0.6…) farklı olduğunu görüyorsunuz. Düşük değerler modelin emin olmadığını, yüksek değerler ise modelin “Bu nesne ‘undefined’ sınıfına daha çok benziyor” şeklinde düşündüğünü gösteriyor.

1. Metal, Plastik ve Glass Tahminleri

* Aynı sahnede, metal kutuların (örneğin içecek kutusu) turkuaz çerçeveyle **“metal 0.8”** gibi yüksek bir confidence değeriyle işaretlendiği kareler göze çarpıyor.
* Bazı karelerde beyaz veya metalik görünümlü nesneler yine “metal” olarak algılanıyor.
* Yeşil veya şeffaf şişeler ise “glass 0.9”, “glass 1.0” gibi etiketlenmiş. Şişelerin boyutu, açısı veya ışık yansıması biraz değişse de model çoğunlukla doğru şekilde cam (glass) olarak sınıflandırıyor.
* Yine gri veya diğer renkteki nesnelerin bazıları da “plastic” olarak tespit edilmiş (örneğin 2. kolajın üst satırında “plastic” yazan kutular).

1. Farklı Confidence Değerleri

* Bu kolajlarda karelerin tekrar etmesi, aynı çöp sahnesinin **farklı confidence eşiğinde** ya da **farklı tahmin adımlarında** nasıl etiketlendiğini gösteriyor.
* Örneğin aynı kırmızı nesne (muhtemelen bir plastik veya metal kutu) için bazen “metal 0.4” ya da “metal 0.6” ya da “metal 0.9” gibi farklı skorlar çıkabiliyor. Bunlar modelin tahmin sırasında kullandığı parametrelere bağlı olarak değişiyor.

Genel Değerlendirme

* Tekrarlanan veya çok benzer karelerdeki etiket farklılıkları, modelin **eşik değeri** ve **dersini ne kadar pekiştirdiği** hakkında fikir veriyor.
* “Undefined” tahminlerin yüksek olması, o sahnede modelin net şekilde sınıflandırma yapamadığı bazı karmaşık, bulanık veya eğitilmemiş atık tiplerinin bulunduğuna işaret eder. Örneğin kırmızı renkli bir atık, eğer eğitimde “metal” ya da “plastic” olarak çok açık örnekle görülmediyse, model onu tam ayırt edemeyip “undefined” da diyebilir ya da zaman zaman “metal” olarak etiketleyebilir.
* **Cam şişelerin** (yeşil veya şeffaf) çoğunlukla doğru etiketlendiği ve yüksek confidence değerleri (0.9 veya 1.0) aldığı gözlemleniyor. Bu, veri setinde cam örneklerinin bol ve tutarlı şekilde etiketlenmiş olabileceğini gösterir.
* “plastic” etiketinin görece az sayıda ve “undefined” ile karışma ihtimalinin yüksek olması, modelin plastik nesneleri bazen tanımakta zorlanabildiğine veya veri setinde yeterince çeşitli plastik örneği olmayabileceğine işaret edebilir.

Sonuç olarak, bu görseller, modelin **karmaşık sahnelerde** (örneğin çöp kutusu içinde farklı atık türleri) verdiği tahminlerin görsel bir özetini sunuyor. Her karede etiketlerin ve güven skorlarının nasıl değiştiğini inceleyerek, modelin güçlü ve zayıf yanlarını, hangi sınıfları ayırt etmede başarılı olduğunu ve hangi tip nesnelerde (çoğunlukla “undefined”) kararsız kaldığını net bir şekilde görebilirsiniz.

## **PROJENİN GELECEĞİ**

1. **Teknolojik Gelişim ve İyileştirmeler**

* **Model Performansını Artırma**: Atık türlerini tanımlamada daha yüksek doğruluk oranlarına ulaşmak için derin öğrenme modellerinin geliştirilmesi.
* **Yeni Materyal Türleri**: Sistem, daha fazla materyal türünü tanıyabilir hale getirilebilir, örneğin organik atıklar veya karmaşık kompozit malzemeler.
* **IoT Entegrasyonu**: Çöp kutularının birbirleriyle ve merkezi bir yönetim sistemiyle iletişim kurabileceği, akıllı şehir altyapısına entegre edilebilecek bir sistem.

1. **Kapsam Genişletme**

* **Farklı Ortamlarda Kullanım**: Ev, ofis, sanayi ve kamu alanları gibi farklı mekanlarda kullanım için optimize edilmiş sürümler geliştirmek.
* **Uluslararası Uygulama**: Sistem, farklı ülkelerin geri dönüşüm politikalarına göre adapte edilebilir.

1. **Veri Yönetimi ve Analiz**

* **Büyük Veri Analitiği**: Toplanan atık verilerinin analiz edilmesiyle atık üretim alışkanlıklarının tespiti ve çevresel stratejiler geliştirilmesi.
* **Gerçek Zamanlı İzleme**: Geri dönüşüm merkezleri için gerçek zamanlı atık akışını izleyerek daha etkili lojistik planlama yapılması.

1. **Sürdürülebilirlik ve Çevreye Katkı**

* **Karbon Ayak İzi Azaltımı**: Daha etkin bir geri dönüşüm süreci ile karbon emisyonlarının azaltılmasına katkı sağlanması.
* **Eğitim ve Farkındalık**: Toplumun çevre bilincini artırmak için sistemle entegre eğitim materyalleri ve uygulamalar geliştirmek.

1. **Ticari Potansiyel**

* **Pazarlama ve İş Modelleri**: Akıllı çöp ayrıştırma sisteminin özel şirketlere ve belediyelere yönelik bir hizmet modeli olarak sunulması.
* **Patent ve Lisanslama**: Sistemin teknolojik yeniliklerinin patentlenmesi ve lisanslanarak gelir elde edilmesi.

1. **Yapay Zeka ve Otomasyon**

* **Otonom Çöp Yönetimi**: Gelişmiş robotik çözümler ile tam otomatik çöp toplama ve ayrıştırma sistemlerinin geliştirilmesi.
* **Hata Düzeltme**: Sistemin kendi kendine öğrenerek tanımlanamayan atık türlerini zamanla doğru sınıflandırabilir hale gelmesi.