metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**KÜTAHYA DUMLUPINAR ÜNİVERSİTESİ  
SİMAV M.Y.O**

**BİLGİSAYAR PROGRAMCILIĞI**

**SİSTEM ANALİZİ DERSİ**

**PROJE RAPORU**

**İnsan tespiti**

**Öğrenci**

**Anıl Akyol Muahmmed Kerem Üstün**

**202242501055 202242501040**

**Mayıs/2024**

**İçindekiler**

**Sonuçlar**

1. **Hedeflenen sonuçlar**
2. **Elde edilen sonuçlar**
3. **Elde edilemeyen sonuçlar.**
4. **Grafikler ve karşılaştırmalar**
5. **Kullanım alanları**

**Süreç**

1. **Süreç özeti**
   1. **Karşılaşılan sorunlar**
2. **Kısaca deneme çalışması**
   1. **Gerekli uygulamaların kurulumu**
   2. **Dataset hazırlığı**
   3. **Kütüphanelerin kurulumu**
   4. **Kodlar**
3. **Gerçek çalışma**
   1. **Roboflow dan dataset bulma**
   2. **Dataset’i hazırlama**
   3. **Drive’ı hazırlama**
   4. **Colab’e giriş**
   5. **Ekran kartına bağlanma**
   6. **Drive’a bağlanma**
   7. **Bulunduğumuz dizini kontrol etme**
   8. **Doğru dizine gitme**
   9. **Ultralytics kütüphanesini indirip kontrol etme**
   10. **Test için Ultralytics kütüphanesinden bir fotoğraf indirme**
   11. **Cv2 ve imutils ktütüphaneleri indirme ve Bus.jpg resimini test etme**
   12. **Kutucukların rengi ve şeklini değiştirme**
   13. **Zip dosyasını ayıklama**
   14. **config.yaml dosyasının içini ayarlama**
   15. **Eğitimi başlatma**
   16. **Pycharm’a giriş**
   17. **Webcam’e bağlanma , modeli entegre etme ve çıkış yama**

**Sonuçlar**

**Hedeflenen Sonuçlar:**

Projemizin hedefi bir güvenlik kamerası sistemi olmaktı ancak o kadar basit değil normal bir güvenlik kamerası sisteminden farklı olarak burada yaptığımız sistem ile birlikte aynı hareket sensörlü kamera sistemlerine benzer olarak bizde harekete duyarlı değil de insan duyarlı bir sistem geliştirdik kaydettiği her hareketi değil sadece içinde insan olan kareleri kaydedecek ve her tespit ettiği insanı kullanıcıya rapor bildirim atarak rapor verecek.

**Elde edilen sonuçlar:**

Projemiz bir güvenlik kamerası sistemi bağlandığı kameranın görüş alanındaki insanları canlı olarak tespit ediyor ve bu görüntüler canlı olarak monitörden izlenebiliyor. Bunun sonucunu bilgisayar kameramız ile test edip onayladık.

**Elde edilemeyen sonuçlar:**

Projemiz için bir de bildirim gönderen bir uygulama yapmak istedik fakat bir sonuç elde edemedik. Önce Python’a flask kurdum ardından app.py diye bir kütüphane açıp basit bir flask sunucusu oluşturduk ardından person\_detection.py kısmına geçip http isteği göndermek için request kütüphanesini kullandık devamında Android studio’ya Android Manifest.xml kısmına geçip internet izni ekledik ardından gradle kısmına okhttp bağımlılığı ekledik flask sunucusundan gelen bildirimleri almak için MainActivity.java kısmına gerekli kodları ekledik devamında MainActivity.xml kısmına geçtik güzel bir tasarım yaptık devamında app.py çalıştırdım gayet güzel çalıştı devamında person\_detection.py kısmını çalıştırdık oda çalıştı Android studio kısmına gelince bir hata almadık ama herhangi bir sonuçta vermedi öyle boş ekran kaldı ardından flask kullanarak basit bir hata kontrolü yaptık acaba Python da sorun olabilir mi diye sonucunda Python sorunsuz çıktı Android studio ile çok uğraştık bir sonuç alamadık mecburen yarım kaldı.

Python flask kurulumu.

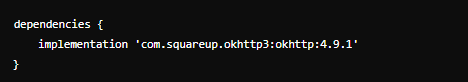
Python flask sunucusu oluşturma

from flask import Flask, request  
  
app = Flask(\_\_name\_\_)  
  
@app.route('/send\_notification', methods=['POST'])  
def send\_notification():  
 print("Notification request received!")  
 return "Notification sent!", 200  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app.run(debug=True, port=5000)

Python’da person\_detection.py kısmına geçip http isteği göndermek için request kütüphanesi oluşturma

import cv2  
from ultralytics import YOLO  
import requests  
  
  
def main():  
 cap = cv2.VideoCapture(0)  
  
 model\_path = "C:/Users/SyST/Desktop/yolov8/person\_detection/yolov8n.pt"  
 model = YOLO(model\_path)  
  
 while True:  
 ret, frame = cap.read()  
  
 if not ret:  
 break  
  
 results = model(frame)  
  
 annotated\_frame = frame.copy()  
 for result in results:  
 boxes = result.boxes  
 for box in boxes:  
 cls = int(box.cls[0])  
 if cls == 0: # '0' class is typically 'person' for YOLO  
 x1, y1, x2, y2 = map(int, box.xyxy[0])  
 cv2.rectangle(annotated\_frame, (x1, y1), (x2, y2), (0, 255, 0), 2)  
 cv2.putText(annotated\_frame, 'Person', (x1, y1 - 10), cv2.FONT\_HERSHEY\_SIMPLEX, 0.9, (0, 255, 0), 2)  
  
 # HTTP isteği gönder  
 try:  
 response = requests.post("http://127.0.0.1:5000/send\_notification")  
 print("Notification sent:", response.status\_code)  
 except Exception as e:  
 print("Failed to send notification:", e)  
  
 cv2.imshow("YOLOv8", annotated\_frame)  
  
 if cv2.waitKey(30) == 27:  
 break  
  
 cap.release()  
 cv2.destroyAllWindows()  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

Android studio’da Android Manifest.xml kısmını ekleme

Android studio’da gradle kısmına okhttp bağımlılığı ekleme

Flask sunucusundan gelen bildirimleri almak için MainActivity.java kısmına gerekli kodları ekleme

package com.calisma.person\_detection  
  
import android.app.NotificationChannel  
import android.app.NotificationManager  
import android.content.Context  
import android.os.Build  
import android.os.Bundle  
import android.os.Handler  
import android.os.Looper  
import androidx.appcompat.app.AppCompatActivity  
import androidx.core.app.NotificationCompat  
import androidx.core.app.NotificationManagerCompat  
import okhttp3.\*  
import java.io.IOException  
  
class MainActivity : AppCompatActivity() {  
  
 private val client = OkHttpClient()  
 private val handler = Handler(Looper.getMainLooper())  
  
 companion object {  
 const val CHANNEL\_ID = "person\_detection\_channel"  
 const val NOTIFICATION\_ID = 1  
 }  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.activity\_main)  
  
 createNotificationChannel()  
  
 handler.postDelayed(object : Runnable {  
 override fun run() {  
 checkForNotification()  
 handler.postDelayed(this, 5000) // 5 saniyede bir kontrol et  
 }  
 }, 5000)  
 }  
  
 private fun createNotificationChannel() {  
 if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.O) {  
 val name = "Person Detection Channel"  
 val descriptionText = "Channel for person detection notifications"  
 val importance = NotificationManager.IMPORTANCE\_DEFAULT  
 val channel = NotificationChannel(CHANNEL\_ID, name, importance).apply **{** *description* = descriptionText  
 **}** val notificationManager: NotificationManager =  
 getSystemService(Context.NOTIFICATION\_SERVICE) as NotificationManager  
 notificationManager.createNotificationChannel(channel)  
 }  
 }  
  
 private fun showNotification() {  
 val builder = NotificationCompat.Builder(this, CHANNEL\_ID)  
 .setSmallIcon(R.drawable.ic\_launcher\_foreground) // Bildirim simgesi  
 .setContentTitle("Güvenlik İhlali")  
 .setContentText("İnsan tespit edildi!")  
 .setPriority(NotificationCompat.PRIORITY\_DEFAULT)  
  
 with(NotificationManagerCompat.from(this)) **{** notify(NOTIFICATION\_ID, builder.build())  
 **}** }  
  
 private fun checkForNotification() {  
 val request = Request.Builder()  
 .url("http://10.0.2.2:5000/send\_notification")  
 .build()  
  
 client.newCall(request).enqueue(object : Callback {  
 override fun onFailure(call: Call, e: IOException) {  
 e.printStackTrace()  
 }  
  
 override fun onResponse(call: Call, response: Response) {  
 if (response.isSuccessful) {  
 runOnUiThread **{** showNotification()  
 **}** }  
 }  
 })  
 }  
}

Android studio’da MainActivity.xml kısmı

<?xml version="1.0" encoding="utf-8"?>  
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:background="@android:color/white">  
  
 <!-- Kameradan gelen görüntünün gösterileceği yer -->  
 <SurfaceView  
 android:id="@+id/surfaceView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent" />  
  
 <!-- Görüntü üzerine kişi tespiti sonucunun gösterileceği alan -->  
 <TextView  
 android:id="@+id/textViewDetection"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_centerHorizontal="true"  
 android:layout\_marginTop="16dp"  
 android:text="Kişi tespiti sonucu"  
 android:textSize="18sp" />  
  
</RelativeLayout>

Python’da bildirim gönderme işleminin başarılı olup olmadığını kontrol etmek için send\_notification endpoint’in çağırıldığından emin olmama yarıyacak bir log yazıyorum.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Python’un çalıştığından emin oluyoruz bu kod sayesinde ama Android studio’da çalıştırmayı başarmış olsakta hata almadığımız halde bildirim almakta başarısız oluyoruz farklı bir yöntemlere yönelme kararı alıyoruz.

**Grafikler ve Karşılaştırmalar**

**metin, diyagram, harita, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu**

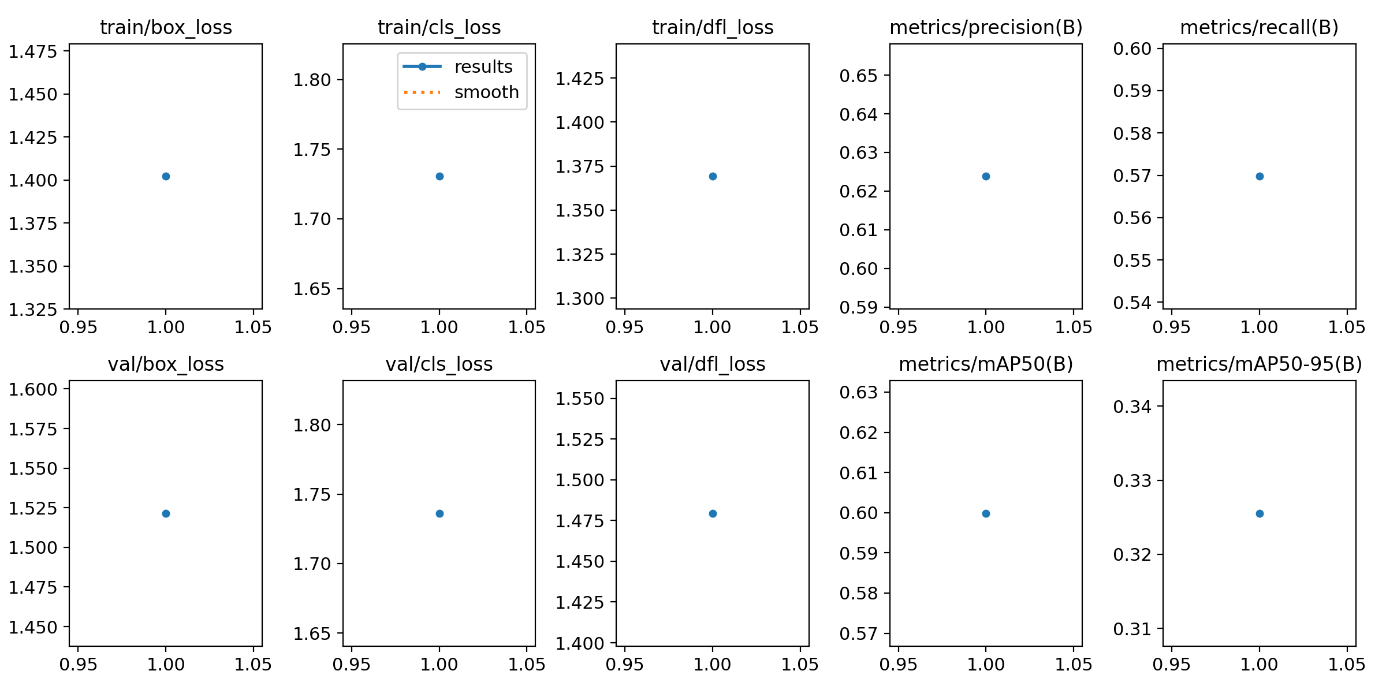
**Kayıp Grafikleri:**

* **Kutuyu Kaybetme (Box Loss):** Bu grafik, modelin nesnelerin sınır kutularını doğru tahmin etmedeki hatasını gösterir. Eğitim boyunca kayıp azalır ve 100. epoch'ta minimum seviyeye ulaşır. Bu, modelin nesneleri daha doğru bir şekilde sınır kutulayabildiğini gösterir.
* **Sınıflandırma Kaybı (Cls Loss):** Bu grafik, modelin nesnelerin sınıflarını doğru tahmin etmedeki hatasını gösterir. Eğitim boyunca kayıp azalır ve 100. epoch'ta minimum seviyeye ulaşır. Bu, modelin nesneleri daha doğru bir şekilde sınıflandırabildiğini gösterir.
* **DFL Kaybı (DFL Loss):** Bu grafik, modelin nesnelerin boyutunu doğru tahmin etmedeki hatasını gösterir. Eğitim boyunca kayıp azalır ve 100. epoch'ta minimum seviyeye ulaşır. Bu, modelin nesnelerin boyutlarını daha doğru bir şekilde tahmin edebildiğini gösterir.

**Performans Metrik Grafikleri:**

* **Hassasiyet (Precision):** Bu grafik, modelin tahmin ettiği nesnelerin gerçekte nesne olup olmadıklarının ne kadarının doğru olduğunu gösterir. Eğitim boyunca hassasiyet artar ve 100. epoch'ta maksimum seviyeye ulaşır. Bu, modelin hatalı tespitleri daha az yaptığı anlamına gelir.
* **Geri Çağırma (Recall):** Bu grafik, modelin mevcut tüm nesneleri ne kadarının doğru bir şekilde tahmin ettiğini gösterir. Eğitim boyunca geri çağırma artar ve 100. epoch'ta maksimum seviyeye ulaşır. Bu, modelin nesneleri kaçırma olasılığının daha düşük olduğunu gösterir.
* **mAP50 (B):** Bu grafik, modelin 50 IOU (Intersection Over Union) eşiğinde nesneleri doğru bir şekilde tespit etmedeki ortalama hassasiyetini gösterir. Eğitim boyunca mAP50 (B) artar ve 100. epoch'ta maksimum seviyeye ulaşır. Bu, modelin genel nesne tespit performansının arttığını gösterir.
* **mAP50-95 (B):** Bu grafik, modelin 50-95 IOU aralığındaki IOU eşiklerinde nesneleri doğru bir şekilde tespit etmedeki ortalama hassasiyetini gösterir. Eğitim boyunca mAP50-95 (B) artar ve 100. epoch'ta maksimum seviyeye ulaşır. Bu, modelin nesneleri farklı IOU eşiklerinde de doğru bir şekilde tespit edebildiğini gösterir.

Şimdi 1 epoch luk yaptığımız ilk denme modelimizle karşılaştıralım



Gördüğünüz gibi herhangibir şeye yakınsam yok sadece 1 epoch luk bir model olduğu için doğal olarak sonuç böyle

Şimdi diğer grafiklere bir bakalım

**Ana proje:**

**metin, ekran görüntüsü, diyagram, öykü gelişim çizgisi; kumpas; grafiğini çıkarma içeren bir resim

Açıklama otomatik olarak oluşturuldu**

 **Eğitim Verimliliği:** Eğitim, 100 epoch boyunca istikrarlı bir şekilde ilerliyor ve her epoch'ta artan bir doğruluk ve geri çağırma performansı gösteriyor.

 **Doğruluk:** Eğitim tamamlandığında (100. epoch), kişi için tüm sınıflar için %97'lik yüksek bir doğruluk elde edilmiş.

 **Geri Çağırma:** Geri çağırma da 100. epoch'ta tüm sınıflar için %97'ye ulaşarak modelin doğru sınıflandırılan nesnelerin çoğunu tespit ettiğini gösteriyor.

 **Genel Eğilim:** Eğitim grafiği, modelin her epoch'ta artan bir şekilde öğrendiğini ve hem doğruluk hem de geri çağırma açısından optimize olduğunu gösteriyor.

**Deneme:**

**metin, ekran görüntüsü, diyagram, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**Ana proje:**

**metin, ekran görüntüsü, diyagram, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu**

* **Eğitim Verimliliği:** Eğitim, 100 epoch boyunca istikrarlı ilerliyor ve her epoch'ta artan bir doğruluk ve geri çağırma performansı gösteriyor.
* **Doğruluk:** Eğitim tamamlandığında (100. epoch), kişi için tüm sınıflar için %97'lik yüksek bir doğruluk elde edilmiş.
* **Geri Çağırma:** Geri çağırma da 100. epoch'ta tüm sınıflar için %97'ye ulaşarak modelin doğru sınıflandırılan nesnelerin çoğunu tespit ettiğini gösteriyor.
* **Genel Eğilim:** Eğitim grafiği, modelin her epoch'ta artan bir şekilde öğrendiğini ve hem doğruluk hem de geri çağırma açısından optimize olduğunu gösteriyor.

**Denem:**

**metin, ekran görüntüsü, diyagram, öykü gelişim çizgisi; kumpas; grafiğini çıkarma içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**Ana proje:**

**metin, ekran görüntüsü, diyagram, öykü gelişim çizgisi; kumpas; grafiğini çıkarma içeren bir resim

Açıklama otomatik olarak oluşturuldu**

 **Eğitim Verimliliği:** Eğitim, 100 epoch boyunca istikrarlı ilerliyor ve her epoch'ta artan bir doğruluk ve geri çağırma performansı gösteriyor.

 **Doğruluk:** Eğitim tamamlandığında (100. epoch), tüm sınıflar için %95,2'lik yüksek bir doğruluk elde edilmiş.

 **Geri Çağırma:** Geri çağırma da 100. epoch'ta tüm sınıflar için %95,2'ye ulaşarak modelin doğru sınıflandırılan nesnelerin çoğunu tespit ettiğini gösteriyor.

 **Genel Eğilim:** Eğitim grafiği, modelin her epoch'ta artan bir şekilde öğrendiğini ve hem doğruluk hem de geri çağırma açısından optimize olduğunu gösteriyor.

**Deneme:**

**metin, diyagram, öykü gelişim çizgisi; kumpas; grafiğini çıkarma, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**Ana proje:**

**metin, ekran görüntüsü, diyagram, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu**

 **Eğitim Verimliliği:** Eğitim, 100 epoch boyunca istikrarlı ilerliyor ve her epoch'ta artan bir doğruluk ve geri çağırma performansı gösteriyor.

 **Doğruluk:** Eğitim tamamlandığında (100. epoch), kişi için tüm sınıflar için %97'lik yüksek bir doğruluk elde edilmiş.

 **Geri Çağırma:** Geri çağırma da 100. epoch'ta tüm sınıflar için %97'ye ulaşarak modelin doğru sınıflandırılan nesnelerin çoğunu tespit ettiğini gösteriyor.

 **Genel Eğilim:** Eğitim grafiği, modelin her epoch'ta artan bir şekilde öğrendiğini ve hem doğruluk hem de geri çağırma açısından optimize olduğunu gösteriyor.

**Deneme:**

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

Açıklama otomatik olarak oluşturuldu**

**Ana proje:**

**metin, ekran görüntüsü, dikdörtgen, diyagram içeren bir resim

Açıklama otomatik olarak oluşturuldu**

 **Eğitim Verimliliği:** Eğitim, 100 epoch boyunca istikrarlı ilerliyor ve her epoch'ta artan bir doğruluk ve geri çağırma performansı gösteriyor.

 **Doğruluk:** Eğitim tamamlandığında (100. epoch), tüm sınıflar için %94,5'lik yüksek bir doğruluk elde edilmiş.

 **Geri Çağırma:** Geri çağırma da 100. epoch'ta tüm sınıflar için %94,5'e ulaşarak modelin doğru sınıflandırılan nesnelerin çoğunu tespit ettiğini gösteriyor.

 **Genel Eğilim:** Eğitim grafiği, modelin her epoch'ta artan bir şekilde öğrendiğini ve hem doğruluk hem de geri çağırma açısından optimize olduğunu gösteriyor.

**Deneme:**

**metin, ekran görüntüsü, diyagram, dikdörtgen içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**Ana poje:**

**metin, taslak içeren bir resim

Açıklama otomatik olarak oluşturuldu**

 **Eğitim Verimliliği:** Eğitim, 100 epoch boyunca istikrarlı ilerliyor ve her epoch'ta artan bir doğruluk ve geri çağırma performansı gösteriyor.

 **Doğruluk:** Eğitim tamamlandığında (100. epoch), tüm sınıflar için %94,5'lik yüksek bir doğruluk elde edilmiş.

 **Geri Çağırma:** Geri çağırma da 100. epoch'ta tüm sınıflar için %94,5'e ulaşarak modelin doğru sınıflandırılan nesnelerin çoğunu tespit ettiğini gösteriyor.

 **Genel Eğilim:** Eğitim grafiği, modelin her epoch'ta artan bir şekilde öğrendiğini ve hem doğruluk hem de geri çağırma açısından optimize olduğunu gösteriyor.

**Deneme:**

**metin, taslak içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**Train bach:**

****

**Kullanım alanları:**

Bu proje Güvenlik ihtiyacı olan her yerde kullanılabilir.

Büyük şirketler, küçük esnaflar, marketler, depolar ve hatta evlerde bile kullanıma uygundur.

**Süreç:**

**Süreç özeti:**

Projemiz aşamalardan oluşmakta: ilk olarak fikir bulma, ne kullanarak yapılabileceğini araştırmak; projeye gerekli yazılım ve donanımları öğrenip uygulamaya geçmek. Fikir bulmak için öncelikle etrafta olan sorunları analiz ettik ve önce bir sorunumuz olursa buna çözüm sunacak bir şey geliştirirsek gerçekten de insanların alabileceği ve ihtiyaç duyacağı bir proje tasarlamış olacaktık. Başlarda sorun bulmakta zorluk çekiyorduk, dışarıdan bir gözle bakmak insanlara sorun yaratan şeyleri anlamamızı zorlaştırıyordu. Ancak dışarıdan insanların yaşadığı sorunlara bakmaktansa kendi yaşadığımız sorunlara bakmanın çok daha kolay olduğuna karar verdik.

Birçok fikir ortaya çıkmaya başlamıştı ama bulduğumuz sorunlardan çözülmesi o kadar da zor olmayan bir şey bulmalıydık; hatta zor olmayan bir şey bile olsa, kendi ekonomik imkanlarımızın yettiği ve birkaç ay içerisinde çözebileceğimiz bir sorun olmalıydı. Buna göre sorunları elemeye başladık ve elimizde kalan birkaç sorunu değerlendirdikten sonra ev ve dükkan güvenliği hakkında olan sorun en mantıklı geldi. Bu sorun için ne tarz bir çözüm bulabiliriz diye aramaya başladık. Bu sorun için başka yapılmış çözümleri araştırmaya koyulduk. Bu çözümlerden kendi yapabileceklerimizi belirledik. En mantıklısının bir kamera sistemi olduğuna karar verdik. Ancak normal bir kamera sistemi yapmak istemedik; bu işe kendi yorumumuzu katmalıydık. Bu sırada kamera sistemlerinde nasıl farklı yaklaşımlar var diye bir göz attık ve orada hareket sensörlü kameralar gözümüze çarptı. Bunu görünce aklıma bu sistemdeki bir sorun geldi: insan dışında herhangi bir hareket olsa bile kullanıcıya haber verilmesi. Bu sorunu çözebileceğimi düşündüm ve araştırmaya başladım. O sırada hareket algılamaktansa direkt insan olduğunu tespit edebilecek bir şey bulursak bu sorunu çözebileceğimi fark ettim. Nasıl insan tespiti yapabileceğimizi anlamak için bir araştırma yaptık ve üniversitedeki bir öğretmenimizin tavsiyesiyle YOLOv8 yapay zeka modelini kullanarak bunu yapabileceğimizi öğrendik.

YOLOv8: YOLOv8, derin öğrenmeye, özellikle Konvolüsyonel Sinir Ağlarına (CNN'ler) dayalı açık kaynaklı bir nesne algılama algoritmasıdır. Nesneleri gerçek zamanlı olarak algılamadaki hızları ve doğrulukları ile bilinen YOLO (You Only Look Once) nesne algılama algoritmaları ailesinin bir parçasıdır.

YOLOv8'i kullanacağımıza karar verdikten sonra yeniden araştırmaya başladık. Ne kadar ne kullanarak yapabileceğimizi anlamış da olsak onu nasıl kullanacağımızı bilmiyorduk, hatta nereden öğrenebileceğimizi bile bilmiyorduk. Bunu nereden öğrenebileceğimizi araştırırken Udemy.com'da karşımıza YOLOv8 kursu veren biri çıktı. Kursu satın aldık ve izlemeye başladık. Kursta belirtilen gerekli driverları kurup aynen izleyerek eğitmenin yaptıklarını, kendi yapacağımız insan tespiti projesine uygun bir şekilde düzenleyerek yapmaya başladık. Tabi sadece yapmayı bilmek yetmiyordu; bu işi yapmamız için etiketlemesi yapılmış insan resimlerinden oluşan büyük bir dataset’e ihtiyacımız vardı.

Etiketleme: YOLOv8'de etiketleme (labeling), modelin nesne tespiti veya segmentasyon gibi görevlerde kullanılmak üzere eğitim verilerini hazırlamak için görsellerdeki nesnelerin sınıflarını ve konumlarını belirleme sürecidir. Etiketleme, her nesne için doğru sınıf etiketleri ve konum bilgilerinin belirlenmesini içerir. Bu süreç, modelin doğru ve verimli bir şekilde öğrenmesi ve tahminler yapması için kritiktir.

Dataset: Bir veri seti (dataset), belirli bir analiz, eğitim veya test amacıyla toplanmış ve düzenlenmiş veri kümesidir. Veri setleri, makine öğrenimi ve yapay zeka modellerinin eğitimi, değerlendirilmesi ve test edilmesi için kritik öneme sahiptir. YOLOv8 gibi nesne tespiti modelleri için veri setleri, genellikle etiketlenmiş görüntülerden oluşur ve bu görüntülerdeki nesnelerin konumları ve sınıfları belirlenmiştir.

Roboflow.com sitesinden bize gereken datasetleri bulduktan sonra onları kendimize göre düzenledik. Artık geriye sadece eğitimi izleyip modeli tasarlamak kalmıştı. İnsan tespiti kısmını yaptıktan sonra, insanları tespit ettikten sonra bunu kullanıcıya belirten bir şey gerekiyordu. Bunu bir site veya mobil uygulama olarak yapmak arasında kaldık ama mobil uygulamayı seçtik. Android Studio kullanarak yapacaktık. Bu sistemin çalışabilmesi için de Raspberry Pi kartını kullanmaya karar verdik.

Android Studio: Android Studio, Google tarafından geliştirilen ve Android uygulamaları oluşturmak için kullanılan resmi entegre geliştirme ortamıdır (IDE). Android Studio, geliştiricilere Android platformunda uygulama geliştirme süreçlerini kolaylaştıran bir dizi araç ve özellik sunar.

Raspberry Pi: Raspberry Pi, İngiltere merkezli Raspberry Pi Vakfı tarafından geliştirilen ve düşük maliyetli, kredi kartı boyutunda bir tek kartlı bilgisayardır. Raspberry Pi, eğitimde bilgisayar bilimi öğretimini desteklemek ve her yaş grubundan insanın programlama ve elektronikle uğraşmasını teşvik etmek amacıyla geliştirilmiştir. İlk modeli 2012 yılında piyasaya sürülmüştür ve o zamandan bu yana çeşitli modelleri piyasaya çıkmıştır.

Udemy kursunu izlemeye başladık ancak başlarda hızlıca ilerliyor olsakta bir süre sonra motivasyonumuz düştü hem okuldan kaynaklı sınavlar benzeri şeylerin araya girmesi hem de kişisel zamanımızın oldukça azalması yüzünden kendimizi projeye verememeye başladık. Tam olarak öğrenmeden işe başlamak istemememizden kaynaklanan bir problem vardı etrafımızdaki insanlardan etkileniyorduk diğer öğrenciler projelerinde hızlan ilerler iken biz daha başlayamamıştık. Bunun gibi sıkıntıları geride bırakarak işin sonunda kursu bitirdiğimizde yapmamız gereken şeyler konusunda baya bir fikrimiz olmuş ve hızlıca bitirebilecektik datasetlerimizi uygun bir şekilde düzenledikten sonra Colab ortamını ve drive ortamını hazırladık datasetimizi drive a zip şeklinde aktardıktan ve config.yaml dosyamızın yolunda doğru girince colab da gerekli kodları yazdık ve eğitmeye başladık ancak ilk seferimizde birkaç hatayla karşılaştık config.yaml nin yolu hakkında bir sorun vardı ama biz sorunu çözmek için ne kadar çaba gösterdiysek te olmadı. O yüzden bizde ana projeden önce colab kullanmadan hataları daha rahat belirleyebileceğimiz anaconda ortamında bir deneme projesi geliştirmeye karar kıldık. Biraz uğraştıktan sonra datasetimiz’i sıfırdan udemy den de yardım alarak bir tane kurdum ve bununla birlikte çalıştırmayı başardık. Bu sefer yeniden Colab ortamında yeni hazırladığımız dataseti kullanarak eğitimi başlattığımızda bu sefer olmuştu yaklaşık bir saat te yüz epoche luk eğitim bitti. Eğitim bitmişti ama bunu kamerada çalıştırmamız gerekiyordu Python charm da biraz chatgpt ve youtube den de yardım alarak bir kod yazdık ve evet artık kamerada da çalışıyordu ama ne yazık ki çok uğraşmamıza rağmen insan tespit ettiğinde bildirim yollama kısmını tamamlayamadık.

**Karşılaşılan sorunlar:**

1. **config.yaml path(yol) hatası:** ismi fark etmeksizin yolov8 uygulamalarımızda dataset’imizin dosya yollarını belirttiğimiz .yaml uzantılı belgenin içeriğini kontrol etmeniz gerek bazen yanlış yol verme verdiğiniz yönde “ / ” işaretinin yönü hakkında da olabilir ters ise düz düz ise ters deneyin ve verdiğiniz yolların tam yol olduğundan emin olun
2. **Gerekli kütüphanelerin aslında kurulamamış olma ihtimali:** tekrar kontrol edin
3. **Dataset’inizdeki resimlerinin isimlerinin 1 2 şeklinde ilerlemiyor olması:** Bunu **Dataseti Hazırlama** kısmında verdiğimiz Python kodu ile çözebilirsiniz,
4. **Colab de ekran kartına bağlanamama:** bu sorunu ücretsiz bir şekilde yeni bir hesap açarak yada aylık colab premium alarak daha da hızlı bir ekran kartına bağlanarak da çözebilirsiniz
5. **Colab de drivedeki dataset’imizin olduğu .zip dosyasının yanlış çıkması:** Bu sorunu 3.maddedeki işlemi yapıp ardından dosyalarınızın düzgün olduğuna emin olduktan sonra tekrar drive a atarak çözebilirsiniz.

**Kısaca Denme Çalışması:**

**Gerekli uygulamaların kurulumu:**

1. Anaconda
2. Python

**Dataset hazırlığı:**

Datasetimiz için olan bütün imageleri ve labelleri dataset adında bir dosyada topluyoruz ve dataset dosyasının bulunduğu dosyaya split.py ve rename.py isimli iki Python belgesi oluşturuyoruz ve bir tane de **config.yaml** adında belge oluşturuyoruz **rename** resim ve labelları sıralı sayılar şeklinde isimlendirmek için **split** ise dosyalarımızı kullanılabilir bir dosya düzenine getirmek için.

**rename.py(resimler için) :**

import os

import glob

# Resimlerin bulunduğu klasörün yolu

klasor\_yolu = "dataset/"

# Klasördeki tüm resim/txt dosyalarını bul

resimler\_txt = glob.glob(os.path.join(klasor\_yolu, "\*.jpg"))

# print(resimler\_txt)

# Yeniden adlandırma işlemi

for i, eski\_ad in enumerate(resimler\_txt, start=1):

    yeni\_ad = os.path.join(klasor\_yolu, f"{i}.jpg")

    os.rename(eski\_ad, yeni\_ad)

    print(f"{eski\_ad} dosyası {yeni\_ad} olarak yeniden adlandırıldı.")

**rename.py(labellar için) :**

import os

import glob

# Resimlerin bulunduğu klasörün yolu

klasor\_yolu = "dataset/"

# Klasördeki tüm resim/txt dosyalarını bul

resimler\_txt = glob.glob(os.path.join(klasor\_yolu, "\*.txt"))

# print(resimler\_txt)

# Yeniden adlandırma işlemi

for i, eski\_ad in enumerate(resimler\_txt, start=1):

    yeni\_ad = os.path.join(klasor\_yolu, f"{i}.txt")

    os.rename(eski\_ad, yeni\_ad)

    print(f"{eski\_ad} dosyası {yeni\_ad} olarak yeniden adlandırıldı.")

bu iki versiyon dada kodu çalıştırıyoruz

**split.py :**

import argparse

import os

import shutil

from random import seed

from random import sample

# Add Parser

parser = argparse.ArgumentParser()

parser.add\_argument("--train", type=int, default=80, help="Percentage of train set")

parser.add\_argument("--validation", type=int, default=10, help="Percentage of validation set")

parser.add\_argument("--test", type=int, default=10, help="Percentage of test set")

parser.add\_argument("--folder", type=str, default="img", help="Folder that contain image")

parser.add\_argument("--dest", type=str, default="img-dest", help="Destination")

args = parser.parse\_args()

def get\_difference\_from\_2\_list(list1, list2):

    set\_list1 = set(list1)

    set\_list2 = set(list2)

    diff = list(set\_list1.difference(set\_list2))

    return diff

def get\_split\_data(list\_id):

    # Train Set

    # Count number of training data

    n\_train = (count \* args.train) / 100

    # Random

    train = sample(list\_id, int(n\_train))

    list\_id = get\_difference\_from\_2\_list(list\_id, train)

    # Validation Set

    # Count number of validation data

    n\_valid = (count \* args.validation) / 100

    # Random

    valid = sample(list\_id, int(n\_valid))

    # Test Set

    # Count number of testing data

    test = get\_difference\_from\_2\_list(list\_id, valid)

    return train, valid, test

def make\_folder():

    folders = ["images", "labels"]

    inner\_folders = ["train", "val", "test"]

    if(not os.path.isdir(args.dest)):

        os.mkdir(args.dest)

    for folder in folders:

        path = os.path.join(args.dest, folder)

        # Check existing folder

        if(not os.path.isdir(path)):

            os.mkdir(path)

        for in\_folder in inner\_folders:

            inner\_path = os.path.join(path, in\_folder)

            # Check existing inner folder

            if(not os.path.isdir(inner\_path)):

                os.mkdir(inner\_path)

def copy\_image(file, id\_folder):

    inner\_folders = ["train", "val", "test"]

    # Image

    source = os.path.join(args.folder, file)

    out\_dest = os.path.join(args.dest, 'images')

    destination = os.path.join(out\_dest, inner\_folders[id\_folder])

    try:

        shutil.copy(source, destination)

        # print("File copied successfully.")

    # If source and destination are same

    except shutil.SameFileError:

        print("Source and destination represents the same file.")

    # labels

    separator = file.find(".")

    filename = file[0:separator]+".txt"

    source = os.path.join(args.folder, filename)

    out\_dest = os.path.join(args.dest, 'labels')

    destination = os.path.join(out\_dest, inner\_folders[id\_folder])

    try:

        shutil.copy(source, destination)

        # print("File copied successfully.")

    # If source and destination are same

    except shutil.SameFileError:

        print("Source and destination represents the same file.")

# Check train set

if((args.train < args.validation) or (args.train < args.test) ):

    print("Train set must has a biggest Percentage")

    exit()

# Check total percentage

total = args.train + args.validation + args.test

if(total > 100):

    print("Total Percentage must 100%")

    exit()

# Count number of data

count = 0

list\_id = []

for file in os.listdir(args.folder):

    if ((file.endswith(".jpg")) or (file.endswith(".png"))):

        list\_id.append(count)

        count+=1

train, valid, test = get\_split\_data(list\_id)

make\_folder()

count = 0

for file in os.listdir(args.folder):

    if ((file.endswith(".jpg")) or (file.endswith(".png"))):

        if(count in train):

            copy\_image(file, 0)

        elif(count in valid):

            copy\_image(file, 1)

        else:

            copy\_image(file, 2)

        count+=1

bu kodu çalıştırdığımızda dosyalarımız düzenlenmiş oluyor.

**Config.yaml :**

path: sizin datasetinizin yolu/dataset

train: images/train # relative to path

val: images/val # relative to path

test: images/test # relative to path

# Class Names

names: ["tespitettğiniz nesnenin ismi"]

**Gerekli kütüphanelerin kurulumu:**

%cd sizin yolunuz/yolov8

%pip install ultralytics

**Kodlar:**

**Anakonda ortam hazırlığı:**

datasetimizin bulunduğu dosyaya cd komudu ile giriyoruz

**kod:**

cd /datasetimizin bulunduğu dosyaya/

conda create -n isim python=3.10 #python kurulumu

conda info –envs #envs lerimizi konrolediyoruz

conda activate isim #envs yi aktif ettik

cd /eğitimi yapıcağımız dosyanın yolu/

yolo detect train model=yolov8n.pt data=data/config.yaml epochs=1 imgsz=640 workers=8 batch=8 name=oluşucak dosyanın ismi #eğitimi başlatıyoruz

**Gerçek Çalışma:**

**Robofow’dan dataset bulma:**

Roboflow.com adresinden universe kısmına giriyoruz ve kendi çalışmamıza uygun dataset buluyoruz veya yapmak isteyen kendi dataset’ini oluşturabilir

**Dataseti hazırlama:**

Datasetimiz için olan bütün imageleri ve labelleri dataset adında bir dosyada topluyoruz ve dataset dosyasının bulunduğu dosyaya split.py ve rename.py isimli iki Python belgesi oluşturuyoruz ve bir tane de **config.yaml** adında belge oluşturuyoruz **rename** resim ve labelları sıralı sayılar şeklinde isimlendirmek için **split** ise dosyalarımızı kullanılabilir bir dosya düzenine getirmek için.

**rename.py(resimler için) :**

import os

import glob

# Resimlerin bulunduğu klasörün yolu

klasor\_yolu = "dataset/"

# Klasördeki tüm resim/txt dosyalarını bul

resimler\_txt = glob.glob(os.path.join(klasor\_yolu, "\*.jpg"))

# print(resimler\_txt)

# Yeniden adlandırma işlemi

for i, eski\_ad in enumerate(resimler\_txt, start=1):

    yeni\_ad = os.path.join(klasor\_yolu, f"{i}.jpg")

    os.rename(eski\_ad, yeni\_ad)

    print(f"{eski\_ad} dosyası {yeni\_ad} olarak yeniden adlandırıldı.")

**rename.py(labellar için) :**

import os

import glob

# Resimlerin bulunduğu klasörün yolu

klasor\_yolu = "dataset/"

# Klasördeki tüm resim/txt dosyalarını bul

resimler\_txt = glob.glob(os.path.join(klasor\_yolu, "\*.txt"))

# print(resimler\_txt)

# Yeniden adlandırma işlemi

for i, eski\_ad in enumerate(resimler\_txt, start=1):

    yeni\_ad = os.path.join(klasor\_yolu, f"{i}.txt")

    os.rename(eski\_ad, yeni\_ad)

    print(f"{eski\_ad} dosyası {yeni\_ad} olarak yeniden adlandırıldı.")

bu iki versiyon dada kodu çalıştırıyoruz

**split.py :**

import argparse

import os

import shutil

from random import seed

from random import sample

# Add Parser

parser = argparse.ArgumentParser()

parser.add\_argument("--train", type=int, default=80, help="Percentage of train set")

parser.add\_argument("--validation", type=int, default=10, help="Percentage of validation set")

parser.add\_argument("--test", type=int, default=10, help="Percentage of test set")

parser.add\_argument("--folder", type=str, default="img", help="Folder that contain image")

parser.add\_argument("--dest", type=str, default="img-dest", help="Destination")

args = parser.parse\_args()

def get\_difference\_from\_2\_list(list1, list2):

    set\_list1 = set(list1)

    set\_list2 = set(list2)

    diff = list(set\_list1.difference(set\_list2))

    return diff

def get\_split\_data(list\_id):

    # Train Set

    # Count number of training data

    n\_train = (count \* args.train) / 100

    # Random

    train = sample(list\_id, int(n\_train))

    list\_id = get\_difference\_from\_2\_list(list\_id, train)

    # Validation Set

    # Count number of validation data

    n\_valid = (count \* args.validation) / 100

    # Random

    valid = sample(list\_id, int(n\_valid))

    # Test Set

    # Count number of testing data

    test = get\_difference\_from\_2\_list(list\_id, valid)

    return train, valid, test

def make\_folder():

    folders = ["images", "labels"]

    inner\_folders = ["train", "val", "test"]

    if(not os.path.isdir(args.dest)):

        os.mkdir(args.dest)

    for folder in folders:

        path = os.path.join(args.dest, folder)

        # Check existing folder

        if(not os.path.isdir(path)):

            os.mkdir(path)

        for in\_folder in inner\_folders:

            inner\_path = os.path.join(path, in\_folder)

            # Check existing inner folder

            if(not os.path.isdir(inner\_path)):

                os.mkdir(inner\_path)

def copy\_image(file, id\_folder):

    inner\_folders = ["train", "val", "test"]

    # Image

    source = os.path.join(args.folder, file)

    out\_dest = os.path.join(args.dest, 'images')

    destination = os.path.join(out\_dest, inner\_folders[id\_folder])

    try:

        shutil.copy(source, destination)

        # print("File copied successfully.")

    # If source and destination are same

    except shutil.SameFileError:

        print("Source and destination represents the same file.")

    # labels

    separator = file.find(".")

    filename = file[0:separator]+".txt"

    source = os.path.join(args.folder, filename)

    out\_dest = os.path.join(args.dest, 'labels')

    destination = os.path.join(out\_dest, inner\_folders[id\_folder])

    try:

        shutil.copy(source, destination)

        # print("File copied successfully.")

    # If source and destination are same

    except shutil.SameFileError:

        print("Source and destination represents the same file.")

# Check train set

if((args.train < args.validation) or (args.train < args.test) ):

    print("Train set must has a biggest Percentage")

    exit()

# Check total percentage

total = args.train + args.validation + args.test

if(total > 100):

    print("Total Percentage must 100%")

    exit()

# Count number of data

count = 0

list\_id = []

for file in os.listdir(args.folder):

    if ((file.endswith(".jpg")) or (file.endswith(".png"))):

        list\_id.append(count)

        count+=1

train, valid, test = get\_split\_data(list\_id)

make\_folder()

count = 0

for file in os.listdir(args.folder):

    if ((file.endswith(".jpg")) or (file.endswith(".png"))):

        if(count in train):

            copy\_image(file, 0)

        elif(count in valid):

            copy\_image(file, 1)

        else:

            copy\_image(file, 2)

        count+=1

bu kodu çalıştırdığımızda dosyalarımız düzenlenmiş oluyor.

**Config.yaml :**

path: sizin datasetinizin yolu/dataset

train: images/train # relative to path

val: images/val # relative to path

test: images/test # relative to path

# Class Names

names: ["tespitettğiniz nesnenin ismi"]

**Drive’ı hazırlama:**

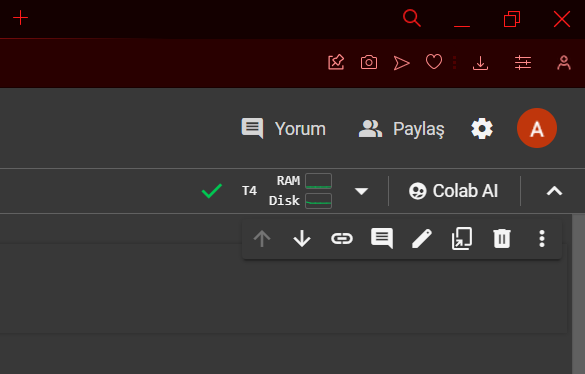
Google drive’ı açıyoruz içine istediğimiz bir isimde bir dosya oluşturup onun içine de data isimli bir dosya açıyoruz. Oluşan dosyamızın içine hazırladığımız dataseti zip e dönüştürüp datanın içine atıyoruz ve yanına da hazırlamış olduğumuz config.yaml belgesini atıyoruz.

**Colab’e giriş:**

Data dosyasının yanına sağ tık ile diğer uygulamalardan bulduğumuz Google colab belgesi oluşturuyoruz ve içine giriyoruz.

**Ekran kartına bağlanma:**

Sağ üst kısımda tam resimdeki t4 yazan yerde bağlan kısmı olacak ona tıkayın

****

**Drive’a bağlanma:**

from google.colab import drive

drive.mount("/content/drive")

**Bulunduğumuz dizini kontrol etme:**

%pwd

**Doğru dizine gitme:**

%cd /content/drive/MyDrive/yolov8/person\_detection

**Ultralytics kütüphanesini indirip kontrol etme:**

# ultralytics'i indir

%pip install ultralytics

import ultralytics

ultralytics.checks()

**Test için Ultralytics kütüphanesinden: bir fotoğraf indirme:**

!yolo detect predict model=yolov8n.pt source='https://ultralytics.com/images/bus.jpg'

**Cv2 ve imutils kütüphaneleri indirme ve Bus.jpg resimini test etme:**

import cv2

import imutils

from google.colab.patches import cv2\_imshow

img=cv2.imread("runs/detect/predict/bus.jpg")

img=imutils.resize(img, width=360)

cv2\_imshow(img)

**Kutucukların rengi ve şeklini değiştirme:**

import cv2

from ultralytics import YOLO

from google.colab.patches import cv2\_imshow

font = cv2.FONT\_HERSHEY\_SIMPLEX

img\_path="bus.jpg"

model\_path="yolov8n.pt"

model=YOLO(model\_path)

img=cv2.imread(img\_path)

img = imutils.resize(img, width=360)

results=model(img)[0]

# print(results)

threshold=0.5

for result in results.boxes.data.tolist():

 x1,y1,x2,y2,score,class\_id=result

 x1 = int(x1)

 y1 = int(y1)

 x2 = int(x2)

 y2 = int(y2)

 class\_id = int(class\_id)

 #print(x1,y1,x2,y2,score,class\_id)

 #print(result)

 if score > threshold:

  cv2.rectangle(img,(x1,y1),(x2,y2),(0,255,0),2)

  class\_name=results.names[class\_id]

  score=score\*100

  text = f"{class\_name}: %{score:.2f}"

  cv2.putText(img, text, (x1,y1-10),font,0.5,(0,255,0),1,cv2.LINE\_AA)

cv2\_imshow(img)

**Zip dosyasını ayıklama:**

!unzip data/person\_dataset.zip -d ./data

**config.yaml dosyasının içini ayarlama:**

path: /content/drive/MyDrive/yolov8/person\_detection/data/person\_dataset

train: images/train # relative to path

val: images/val # relative to path

test: images/test # relative to path

# Class Names

names: ["Person"]

**Eğitimi başlatma :**

!yolo detect train data=data/config.yaml model=yolov8n.pt epochs=100 imgsz=640 workers=8 batch=8 device=0 name=yolov8\_person\_detection

**Pycharm’a giriş:**

Pycharm veya muadili bir uygulamayı açıyoruz ve gerekli kütüphaneleri kuruyoruz Alt+f12 ile terminali açıp alttaki kodu yazıyoruz.

Pip install ultralytics opencv - python – headless flask

**Webcam’e bağlanma ,modeli entegre etme ve çıkış yapma:**

import cv2  
import smtplib  
import ssl  
from email.mime.text import MIMEText  
from email.mime.multipart import MIMEMultipart  
from ultralytics import YOLO  
  
def main():  
 # Webcam'den video yakalama  
 cap = cv2.VideoCapture(0)  
  
 # Modelin yolunu belirtin  
 model\_path = "C:/Users/mamito/Desktop/yolov8/person\_detection/yolov8n.pt"  
 model = YOLO(model\_path)  
  
 # İnsan tespiti bayrağı  
 person\_detected = False  
  
 while True:  
 # Video karelerini oku  
 ret, frame = cap.read()  
  
 if not ret:  
 break  
  
 # Modeli kullanarak tahmin yap  
 results = model(frame)  
 # print(results)  
 #print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  
 #print(type(results))  
 #print("00",results[0][2])  
 #print("1",results[0][3])  
 #print("--------------------------------")  
  
 # İnsan tespiti kontrolü  
 person\_in\_frame = False  
 for result in results:  
 boxes = result.boxes # tespit edilen kutular  
 for box in boxes:  
 cls = int(box.cls[0]) # sınıf indeksini al  
 if cls == 0: # '0' sınıfı insan olarak eğitilmişse  
 person\_in\_frame = True  
 x1, y1, x2, y2 = map(int, box.xyxy[0]) # kutunun koordinatlarını al  
 cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)  
 cv2.putText(frame, 'Person', (x1, y1 - 10), cv2.FONT\_HERSHEY\_SIMPLEX, 0.9, (0, 255, 0), 2)  
  
  
 # İşlenmiş görüntüyü göster  
 cv2.imshow("YOLOv8", frame)  
  
 # 'ESC' tuşuna basılarak çıkış yap  
 if cv2.waitKey(30) == 27:  
 break  
  
 # Kaynakları serbest bırak ve pencereleri kapat  
 cap.release()  
 cv2.destroyAllWindows()  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

eğer import kısmında hata veriyorsa verdiği kısma tıklatıp Alt+enter yapıp install kısmına enterlerseniz sorun çözülecektir

**Projemiz burada sona ermiştir kodu çalıştırdığınızda**

**Bilgisayarınızın kendi webcam’ine bağlanıp orada gözüken insanları kutucuk içine alıp üstünde Person yaması gerekiyor sonuç fotoğrafı aşağıda**

metin, insan yüzü, yazılım, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu

**REFERANSLAR & KAYNAKÇA:**

Udemy: YOLOv8 Sıfırdan İleri Seviye Projeler YOLOv8 Kursu

ChatGPT

Youtube: Mr çilek