

DÜZCE ÜNİVERSİTESİ

AKÇAKOCA MESLEK YÜKSEKOKULU

BİLGİSAYAR PROGRAMCILIĞI BÖLÜMÜ

Staj Projesi Raporu

Teslim Eden:

İsim-Soyisim : Sami DURAN

Numara : 221501013

Kuruluşun İsimi ve Yeri : Konveyör

Tuzla, İstanbul, Türkiye

Staj Tarihleri : 10.06.2024 - 25.07.2024

**İçindekiler**

[Giriş 2](#_Toc172737717)

[Proje Hakkında Kısa Bilgi 2](#_Toc172737718)

[Şirket Hakkında Bilgi 2](#_Toc172737719)

[Proje ve Yapılan İş 3](#_Toc172737720)

[Ekler 5](#_Toc172737721)

[Programın Çalışma Mantığı 5](#_Toc172737722)

[Programın Kullanımı 5](#_Toc172737723)

[Programın Önemli Kod Blokları 7](#_Toc172737724)

[Main.py’deki Önemli Kodlar 8](#_Toc172737725)

[Test.py’deki Önemli Kodlar 18](#_Toc172737726)

[Sonuç 22](#_Toc172737727)

# Giriş

Bu rapor, Konveyör Şirketi’nde gerçekleşen staj süresince yapılan projeyi ve işleri ayrıntılı bir şekilde ele almaktadır. Stajın temel amacı, Bilgisayar Teknolojileri ve Yazılım dünyasına giriş yapmak, üretim süreçlerini iyileştirmek, verimliliği artırmak ve maliyetleri düşürmek için çeşitli uygulamalar ve projeler gerçekleştirmektir. Bu süreçte hem teorik bilgilerin pratiğe dökülmesi hem de gerçek dünya iş tecrübelerinin kazanılması hedeflenmiştir.

## Proje Hakkında Kısa Bilgi

Bu proje, bir güvenlik ip kamerası izleme ve yönetim uygulamasıdır. Proje, kullanıcı arayüzü oluşturmak için PyQt5, video işleme için OpenCV, nesne tespiti için ssd\_mobilenet\_v2 modülünü ve SMTP üzerinden e-posta gönderimi için smtplib kütüphanesini kullanılmıştır. Ayrıca önemli bilgileri kaydetmek için Database oluşturmak yerine Json kullanılmıştır. İşte kütüphanelerin kullanım amaçları ve projedeki özellikleri:

* PyQt5: Projenin arayüz kısmını oluşturmak için kullanılmıştır.
* OpenCV: Görüntü işlemek için kullanılır.
* JSON: Kamera verilerini (IP ve isim) saklamak ve okumak için kullanılır.
* Smtplib: E-posta gönderimi için kullanılır. Nesne tespiti durumunda görüntüleri e-posta ile göndermek için kullanılır.
* TensorFlow: Derin öğrenme modelini çalıştırmak için kullanılır.

## Proje ve Yapılan İş

Staj süresince üzerinde çalışılan proje, Konveyör şirketinin güvenlik sistemiyle alakalıdır. Projede Python yazılım dilini kullanarak; güvenlik sistemine bağlı olan IP Kameraların görüntülerini, görüntü işleme tekniklerini kullanarak insan ve araç tespiti yapılması ve yapılan tespitin fotoğraflanıp geri bildirimi amaçlanmıştır.

Projeye ilk başladığımda, yapacağım projeye en uygun programlama dilinin Python olduğuna karar verdim. Pythonda kullanmak için görüntü işleme yapay zeka modellerini araştırdım ve en sonunda kendi veri setimi çıkartıp kendi yapay zeka modelimi yazmak istedim bunun için araştırmalara başladım. Kaggle da bulduğum “Large Scale Multi-Camera Detection Dataset” adlı veri seti ile modelimi oluşturmaya karar verdim.

Verisetindeki fotoğrafları “labelimg” yazılımıyla etiketlemeye karar verdim. Yaklaşık bir 650 fotoğraf etiketledikten sonra modelimi oluşturmak istedim. Modelimi oluştururken TensorFlow kütüphanesini kullandım. Eğitim verisini hazırlamak için fotoğraflar ve etiketlendirilmiş XML dosyalarını TensorFlow TFRecord formatına dönüştürmek için “generate\_tfrecord.py” scriptini kullandım. TensorFlow Object Detection API kullanarak model eğitimi gerçekleştirdim.

Eğittiğim modeli test etmek için gerekli hazırlıkları yaptım.

* 'Precision/mAP': 0.10, # Ortalama Doğruluk (Mean Average Precision)
* 'Precision/mAP@.50IOU': 0.20, # Ortalama Doğruluk (IoU %50)
* 'Precision/mAP@.75IOU': 0.05, # Ortalama Doğruluk (IoU %75)
* 'Recall/AR@1': 0.05, # 1 tespit başına Ortalama Geri Çağırma
* 'Recall/AR@10': 0.15, # 10 tespit başına Ortalama Geri Çağırma
* 'Recall/AR ': 0.25 # Nesneler için Ortalama Geri Çağırma

Test sonuçlarımda aldığım sonuçlar beni hüsrana uğratmasına rağmen farklı bir algoritma kullanıp modelimi tekrardan eğittim. Aldığım sonuçlar yine hüsrandı ve benim eğittiğim model sadece 1 tespit için çalışıyordu, yani fotoğrafta 2 tane araba olsa bile 1 tane araba döndürüyordu. Bunu nasıl düzeltirim diye biraz araştırma yaptım fakat bir sonuca ulaşamadım. Her çözüm kendi kullandığım algoritmanın dışına çıkıyordu ve kodlar karmaşıklaşmaya başlıyordu. Bende kendi modelimi oluşturmaktan vazgeçip hazır model kullanmaya karar verdim.

Araştırmalarımda projeme en uygun görüntü işleme yapay zeka modelinin “ssd mobilenet v2” olduğuna karar verdim. Modeli resmi github sayfasından indirdim ve proje dosyalarımın içine koydum. Hazır modeli test etmek için ilk kodlarımı yazdım. Ssd mobilenet 90 farklı nesneyi algılaya bildiği için ilk başta kameramda gördüğü çoğu şeyi tanıdı. Ama bana sadece insan ve araba lazım. Bunu düzeltmek için modelin categori\_indexinden gelen 90 farklı nesneyi insanı ve arabayı tanımlayan 1. İndex ve 2. İndex olacak şekilde kısıtladım. Bunu yapmam hem programı hafifletti hemde istediğim kısma geldi. Modeli düzenlediğim haliyle tekrardan çalıştırdım ve istediğim sonuçları aldım.

Ama şu ana kadar hep webcam kamerasıyla testlerimi gerçekleştirdim. İp kamerasına bağlanmak için araştırma yaptım. İlk denemelerimde hep hatalar aldım. Kameraların arayüz sayfalarına bağlanabilmeme rağmen kameraların görüntüleri gelmiyordu. Sonra fark ettim ki bağlı olduğum ağın ip kameralarına erişimi yokmuş. Bu sorunu düzeltmek için yetkili çalışanlara danıştım ama kameralarla bağlantı kurabilmem için güvenlik serverine bağlanmam gerekiyormuş. Güvenlik serverine girersemde ofiste olan diğer işlerim aksayacağı için bağlanmadım. Depoda boşta gördüğüm ip kamerasını temin ettim. Elime ulaştığında LAN kablosu ile bağladım ve kameradan gelen görüntüye eriştim. Modeli ip kamerada denedim ve çalıştığını gördüm.

Kurduğum görüntü işleme algoritması ile ip kamera bağlantısı başarılı bir şekilde çalıştıktan sonra GUI (Graphical User Interface) oluşturmaya karar verdim. GUI oluşturmak için PYQT5 kullandım. Programı indirdikten sonra sürükle bırak mantığıyla çalışan arayüzünde, kendi main formumu oluşturdum. PYQT5 designer de oluşturulan arayüzler ‘.ui’ şeklinde kaydedilir. Bunu Python koduna çevirmek için gerekli olan terminal kodunu yazdım ve Python dosyasına çevirdim. Python dosyasında da gerekli değişiklikleri yaptım ve kendi algoritmamı içine entegre ettim.

Arayüzde, farklı ip kameralar eklemek için bir kısım ve gönderen e-posta adresi, gönderilen e-posta adresini değiştirmek için bir kısım bulunuyor. İp kamerayı ekleyip seçtikten sonra başlat kısmına basıyoruz ve kameramız açılıyor. Tespit edilen nesneleri ekranda dikdörtgen içine alıyor ve ekran görüntüsünü alıyor. İlk nesneyi tespit ettikten sonraki 60 saniye içerisinde 5 ekran görüntüsünü alıp, gönderilecek olan e-posta adresine gönderiliyor.

Projenin önemli kod parçacıkları ve uygulama kullanımını detaylı bir şekilde açıklayan döküman **Ekler** kısmında mevcuttur.

# Ekler

Bu kısımda projenin nasıl çalıştığını ve önemli kod bloklarını anlatacağım.

## Programın Çalışma Mantığı

Program ip kameralardan çektiği görüntüleri, kare kare işleyerek nesne tespiti yapar. Eğer araba ya da insan tespit ederse, tespit edilen kareleri kaydeder. İlk tespit etme işlemi gerçekleştikten sonra 60 sn içerisinde 5 kare kaydeder ve kayıtlı olan e-posta adresine bu kareler gönderilir. 300 saniyelik bekleme süresine girdikten sonra tekrardan nesne tespit etmek için hazır hale gelir.

## Programın Kullanımı

Programın kullanımı, kullanıcı için en basit seviyeye indirilmiştir. Basit bir arayüze sahiptir her kullanıcı kullanabilir.

metin, ekran görüntüsü, bilgisayar, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

Programı ilk başlattığımızda karşımıza böyle boş bir ekran gelecektir. Ip kamera eklemek için sol altta bulunan IP ekle kısmına, IP Kameranın ipsini ve kaydedileceği adı giriyoruz.

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

Açıklama otomatik olarak oluşturuldu

Yukarıdaki fotoğraftaki gibi bilgileri girdikten sonra IP Ekle buttonuna basıyoruz ve sol üstte bulunan list box nesnesinin içine dahil oluyor.

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

Açıklama otomatik olarak oluşturuldu

Sağ tarafta list box’a eklediğimiz ip kameranın ismi görünüyor.

Üstüne tıklayıp ekle diyoruz. Seçilen item uygulamanın sağ

tarafında bulunan list box’a eklendi. En fazla 4 tane kamera

ekleyebiliyoruz. Daha sonrasında orta alt kısımda bulunan başlat

butonuna basıp, görüntü işlemeye başlıyoruz.

Sağ altta bulunan ‘Ayarlar’ butonuna bastığımızda, karşımıza e-posta ayarları çıkıyor.

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

Açıklama otomatik olarak oluşturuldu

Bu sayfada şirketimze ait bilgileri giriyoruz. SMTP(Simple Mail Transfer Protocol) Server kısmına serverimizin adını giriyoruz. (ör: Gmail için server etki alanı adı : “smtp.gmail.com”). SMTP Server Port kısmınada serverimizin portunu giriyoruz (ör: Gmail için port : 587). Kaydet dedikten sonra artık e posta gönderme işlemlerimiz de tamamlanıyor.

## Programın Önemli Kod Blokları

İlk öncelikle PYQT5 in kısaca çalışma mantığını anlatacağım. Sonrasında ise kullandığım önemli fonksiyonları anlatacağım.

PYQT5 Designer ile oluşturduğumuz arayüzü dönüştürmek için cmd den ui dosyasını oluşturduğumuz konuma geliyoruz ve “pyuic5 -x dosya\_Adi.ui -o dosaya\_Adi.py” komutunu çalıştırıyoruz. Aynı konuma bir Python dosyası oluşturur. Oluşturduğu Python dosyasında Class oluşturur ve bu Classın içine setupUİ fonksiyonu oluşturur. Bu fonksiyonda Designer’de oluşturduğumuz nesnelerin kodları bulunur.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek bir kod

## Main.py’deki Önemli Kodlar

**Load**

PYQT5in mantığını bu şekildedir. Program başladığında kayıtlı olan verileri list box nesnesine yüklemesi için bir “load” fonksiyonu oluşturdum. Bu fonksiyon program başladığında çalışması için setupUi in içinde çağırılır.

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

Açıklama otomatik olarak oluşturuldu**

**Read\_data**

Read\_data fonksiyonu, json ile kaydedilen dosyayı okumak için kullanılır. Bu kodda json kütüphanesi kullanılmıştır.

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

Açıklama otomatik olarak oluşturuldu

Data\_file dan gelen veriyi file değişkeninin için atar ve sonrasında json.load ile okur. Gelen veriyi geri döndürür. Eğer bir dosya bulamazsa boş bir değer döndürür.

**Save\_data**

Save\_data fonksiyonu, json kütüphanesini kullanarak aldığı bilgileri kaydeder. Bu fonksiyonun 2 parametre ister “ip” ve “name”. Daha önceden kaydedilmiş veriler kaybolmaması için bulduğu dosyanın içindeki verileri alır, yeni verileri ekleyerek kaydeder.

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

Açıklama otomatik olarak oluşturuldu

**Btn\_add\_click**

Bu fonksiyon btn\_add nesnesinin click olayını gerçekleştirir. List\_free\_cam de seçili olan nesneyi List\_add\_cam’e aktarır. Eğer List\_add\_cam’de 4 eleman varsa uyarı mesajı verir.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Btn\_remove\_click**

Bu fonksiyon, btn\_remove nesnesinin click olayını gerçekleştirir. List\_add\_cam’de seçili olan nesneyi silmek için kullanılır. Silinen nesne List\_free\_cam’e aktarılır.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Btn\_ipadd\_click**

Bu fonksiyon, Btn\_ipadd nesnesinin click olayını gerçekleştirir. Girilen ip ve isim bilgisini kaydeder.

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

Açıklama otomatik olarak oluşturuldu

**Btn\_start\_click**

Bu fonksiyon, Btn\_start nesnesinin click olayını gerçekleştirir. List\_add’de olan verinin ismini ve List\_add’nin item sayısını döndürerek Add\_cam fonksiyonunu çalıştırır.

**metin, ekran görüntüsü, yazı tipi, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**Find\_ip\_by\_name**

İsminden de anlaşılacağı gibi isme ait olan ipyi bulup çevirir.

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

Açıklama otomatik olarak oluşturuldu

**Add\_cam**

Programın en önemli fonksiyonlarından biridir. 2 taneparametresi vardır, url ve camIndex. metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

starttime: Fonksiyonun çalışmaya başladığı zamanı tutar.

object\_detected: Bir nesne tespit edilip edilmediğini belirlemek için kullanılan bayrak değişkeni.

ip: Ui\_Dialog.find\_ip\_by\_name fonksiyonu ile url parametresinden alınan IP adresi.

cap: OpenCV kullanarak IP adresinden video akışını başlatır.

cap.set(...): Video codec formatını H.264 olarak ayarlar. Bu formatı kullanmamızın sebebi ip kameralarımızdan aldığımız görüntü formatı H.264 olmasıdır.

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

Açıklama otomatik olarak oluşturuldu

Görüntü gelmişse ve Self.running ture ise döngümüz başlıyor. Ret değeri boşsa döngüyü kırıyoruz.

frame: Kameranın indeks numarasına bağlı olarak kareyi yeniden boyutlandırır.

aframe, class\_name: test.detector fonksiyonu ile karedeki nesneyi tespit eder ve sonuçları döner.

self.display\_Image: Fonksiyonu görüntüyü ekrana yazdırır.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Eğer sleeptime 120 saniyeden fazla ise:

self.totaltime: Başlangıçtan bu yana geçen toplam süreyi hesaplar.

class\_name 'person' veya 'car' ise:

sleeptime sıfırlanır ve object\_detected True olarak ayarlanır.

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

Açıklama otomatik olarak oluşturuldu**

Eğer bir nesne tespit edilmişse ve toplam süre 60 saniyeden az ise: Her 12 saniyede bir kareyi temp klasörüne kaydeder.

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

Açıklama otomatik olarak oluşturuldu**

Eğer toplam süre 59 saniyeden fazla ise ve bir nesne tespit edilmişse:

object\_detected False yapılır.

self.first True olarak ayarlanır.

Ui\_Dialog.send\_mail: E-posta göndermek için fonksiyon çağırıldı.

Ui\_Dialog.clean: Temp klasörünü temizler.

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

Açıklama otomatik olarak oluşturuldu**

Eğer toplam süre 59 saniyeden fazla ise ve self.first True ise:

self.first False yapılır.

sleeptime ve totaltime sıfırlanır.

self.reload\_cam(): Kamerayı yeniden yükler.

**Display\_image**

Bu fonksiyonun 2 parametresi vardır. Fonksiyon Parametreleri:

* img: Görüntü verisi (numpy array formatında).
* labelIndex: Görüntünün hangi kamera penceresine yerleştirileceğini belirten indeks (varsayılan değeri 0).

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

Açıklama otomatik olarak oluşturuldu**

qformat: Görüntü formatını belirler. Varsayılan olarak QImage.Format\_Indexed8 kullanılır.

Eğer img 4 kanal içeriyorsa (şeffaflık kanalını da içerir), qformat QImage.Format\_RGBA888 olarak ayarlanır.

Diğer durumlarda (3 kanal, yani RGB), qformat QImage.Format\_RGB888 olarak ayarlanır.



QImage(img, img.shape[1], img.shape[0], qformat): Numpy array formatındaki görüntüyü QImage formatına dönüştürür.

img.rgbSwapped(): RGB ve BGR renk düzenini değiştirir (OpenCV genellikle BGR formatını kullanır).

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

Açıklama otomatik olarak oluşturuldu

Eğer labelIndex 0 ise:

self.cam0.setGeometry(...): cam0 widget'ının geometrisini (konum ve boyut) ayarlar.

self.cam0.setPixmap(…): Görüntüyü cam0 widget'ına yerleştirir.

self.cam0.setAlignment(…): Görüntüyü yatay ve dikey olarak merkezler.

Diğer sonuçlarda da qLabelların yeri ve konumu ayarlanır.

**Clean ve Send\_mail**

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Clean** fonksiyonu, temp klasöründeki tüm dosya ve alt klasörleri silmek için kullanılır. Os ve shutil kütüphanelerini kullanır.

**Send\_mail** fonksiyonu, belirli ayarlarla bir e-posta gönderir ve temp klasöründeki dosyaları ekler.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Ayarları Okuma: Ui\_Dialog.read\_settings() fonksiyonundan e-posta gönderim ayarlarını alır (gönderen, alıcı, gönderenin şifresi, SMTP sunucu bilgileri).

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

E-posta Bilgileri: Gönderen (fromaddr) ve alıcı (toaddr) adreslerini belirler.

E-posta Mesajı Oluşturma: MIMEMultipart() kullanarak e-posta mesajını oluşturur ve başlık, içeriği ekler (MIMEText ile mesajın metnini belirtir).

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Ek Dosyaları Ekleme: temp klasöründeki tüm dosyaları ekler. Her dosyayı okur, base64 formatında kodlar ve e-posta mesajına ekler.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

E-posta Gönderimi: SMTP sunucusuna bağlanır (smtplib.SMTP), TLS bağlantısını başlatır, e-posta adresi ve şifre ile giriş yapar, e-postayı gönderir ve bağlantıyı kapatır.

## Test.py’deki Önemli Kodlar

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

Açıklama otomatik olarak oluşturuldu**

* **Numpy**: NumPy kütüphanesi, matematiksel işlemler ve veri işleme için kullanılır.
* **TensorFlow**: TensorFlow kütüphanesi, makine öğrenmesi ve derin öğrenme modelleri ile çalışmak için kullanılır.
* **cv2**: OpenCV kütüphanesi, görüntü işleme ve video analizleri için kullanılır.
* **label\_map\_util** ve **visualization\_utils**: TensorFlow Object Detection API'den, etiketler ve görselleştirme için yardımcı fonksiyonları sağlar.
* **os**: İşletim sistemi ile ilgili işlemleri yapar, dosya yollarını yönetir.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

MODEL\_NAME: Kullanılacak modelin adını belirler.

PATH\_TO\_CKPT: Modelin eğitilmiş ağırlıklarının bulunduğu dosya yolunu belirtir.

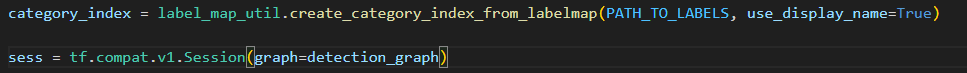
PATH\_TO\_LABELS: Etiketlerin bulunduğu dosya yolunu belirtir .

NUM\_CLASSES: Modelin tanıyabileceği toplam nesne sayısını belirtir.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

* detection\_graph: TensorFlow'da bir grafik oluşturur. Bu grafik, modelin yapısını ve hesaplama grafiğini içerir.
* with detection\_graph.as\_default(): Bu kod bloğunda detection\_graph grafik olarak varsayılan grafik olarak ayarlanır.
* od\_graph\_def: TensorFlow'un GraphDef yapısında, modelin ağırlıklarını ve yapısını tanımlayan bir nesnedir.
* tf.io.gfile.GFile(PATH\_TO\_CKPT, 'rb'): Model dosyasını ikili (binary) modda açar.
* serialized\_graph: Modelin serileştirilmiş halini okur.
* od\_graph\_def.ParseFromString(serialized\_graph): Serileştirilmiş grafiği GraphDef nesnesine dönüştürür.
* tf.import\_graph\_def(od\_graph\_def, name=''): GraphDef nesnesini mevcut grafiğe ekler.



label\_map\_util.create\_category\_index\_from\_labelmap fonksiyonu ile etiketlerin bir indeksini oluşturur. Bu, modelin tespit ettiği nesnelerin isimlerini almayı sağlar.

sess: TensorFlow oturumu oluşturur. Bu oturum, modelin çalıştırılacağı ve verilerin işlendiği yerdir. detection\_graph grafiğini kullanır.

**Detector**

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

np.expand\_dims(): Görüntüyü, TensorFlow modelinin beklediği formata uygun hale getirir. Model, bir batch (küme) ile çalışır, bu yüzden görüntüyü batch boyutunda genişletir.

get\_tensor\_by\_name: TensorFlow grafik nesnesi (detection\_graph) üzerinden modelin çıkışlarına ve girişlerine referanslar alır. Bu, modelin hangi tensor'larına erişilmesi gerektiğini belirtir:

* image\_tensor: Modelin girdi tensor'u (görüntü verisi).
* boxes: Tespit edilen nesnelerin sınır kutularının tensor'u.
* scores: Tespit edilen nesnelerin güven skorlarının tensor'u.
* classes: Tespit edilen nesnelerin sınıf ID'lerinin tensor'u.
* num\_detections: Tespit edilen nesnelerin sayısının tensor'u.

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

Açıklama otomatik olarak oluşturuldu**

sess.run(...): TensorFlow oturumunda belirtilen tensor'ları hesaplar. feed\_dict kullanarak image\_tensor'a image\_np\_expanded verisini besler. Bu işlem sonucunda:

boxes: Tespit edilen nesnelerin sınır kutuları.

scores: Tespit edilen nesnelerin güven skorları.

classes: Tespit edilen nesnelerin sınıf ID'leri.

num\_detections: Tespit edilen nesnelerin sayısı.



np.squeeze(...): Boyutları 1 olan eksenleri kaldırır, böylece daha kullanışlı bir format elde edilir.

astype(np.int32): Sınıf ID'lerini tam sayıya dönüştürür.

metin, ekran görüntüsü, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

class\_name: Tespit edilen nesnelerin isimlerini saklar.

detected\_scores[i] > 0.6: Güven skorları belirli bir eşik değerini geçiyorsa (0.6 bu durumda).

class\_id = detected\_classes[i]: Tespit edilen nesnenin sınıf ID'sini alır.

if class\_id in category\_index: Sınıf ID'si category\_index içinde tanımlıysa:

class\_name: Sınıf ID'sini isimle eşler.

if class\_id == 1 or class\_id == 2: Sadece belirli sınıflar (ID 1 veya 2) için:

ymin, xmin, ymax, xmax = boxes[0, i]: Sınır kutusunun koordinatlarını alır.

vis\_util.draw\_bounding\_box\_on\_image\_array(...): Görüntü üzerinde sınıf adını ve sınır kutusunu çizer.

Programın önemli kod satırları ve çalışma mantığı bu şekildedir.

# Projenin Örnek Fotoğrafları



Örnek bir e-posta görüntüsü

metin, iç mekan, duvar, bilgisayar içeren bir resim

Açıklama otomatik olarak oluşturuldu metin, iç mekan, duvar, bilgisayar içeren bir resim

Açıklama otomatik olarak oluşturuldu

İp kamera ile algılanan örnek nesneler.

# Sonuç

Bu proje, Konveyör Şirketi’nin güvenlik sistemine entegre edilebilecek bir güvenlik kamerası izleme ve yönetim uygulamasının geliştirilmesini hedeflemiştir. Proje kapsamında kullanılan teknolojiler ve yöntemler sayesinde güvenlik kamerası görüntüleri üzerinde etkin bir şekilde nesne tespiti gerçekleştirilmiş ve bu tespitler kullanıcılara e-posta yoluyla bildirilmiştir. Aşağıda proje sürecinde elde edilen ana sonuçlar ve karşılaşılan zorluklar özetlenmiştir:

1. Başarılar
   1. Nesne Tespiti: Projeye dahil edilen "ssd\_mobilenet\_v2" modeli, kullanıcıların belirlediği nesneleri (insan ve araç) etkili bir şekilde tespit edebilmiştir. Modelin belirlenen nesneler üzerinde yüksek doğruluk oranıyla çalışması sağlanmıştır.
   2. GUI Oluşumu: PyQt5 kullanılarak geliştirilmiş kullanıcı dostu arayüz, IP kameraların eklenmesi ve yönetilmesi işlemlerini basit ve etkili bir şekilde yapabilen bir platform sunmuştur. Kullanıcılar, kameraları ekleyip yönetme işlemlerini kolayca gerçekleştirebilirler.
   3. E-posta Entegrasyonu: E-posta gönderim özelliği, tespit edilen nesnelerle ilgili bilgilerin hızlı bir şekilde kullanıcıya iletilmesini sağlamıştır. Bu özellik, güvenlik bildirimlerinin anlık olarak yapılabilmesine olanak tanımıştır.
2. Karşılaşılan sorunlar ve çözümler
   1. Model Eğitimi: İlk başta kendi yapay zeka modelini eğitme sürecinde düşük doğruluk oranları ve sınırlı tespit başarısı gibi problemler yaşanmıştır. Bu sorunlar, farklı algoritmalar denemekle çözülmeye çalışılmıştır. Ancak en uygun çözüm olarak, önceden eğitilmiş "ssd\_mobilenet\_v2" modeline geçilmiştir.
   2. IP Kamera Bağlantısı: IP kameraların ağ üzerinden erişimi konusunda yaşanan teknik zorluklar, başlangıçta kameraların görüntülerine erişimi engellemiştir. Bu sorunu aşmak için ofis içindeki ağ yapılandırmaları ve yetkililerle koordinasyon sağlanmış, alternatif olarak bağımsız bir IP kamera kullanılmıştır.
   3. Performans Sorunları: Uygulamanın belirli koşullarda (örneğin, çok sayıda kamera bağlantısı) performans düşüşleri yaşadığı gözlemlenmiştir. Bu sorun, yazılımın optimize edilmesi ve kaynak yönetiminin iyileştirilmesi ile ele alınmıştır.
3. Gelişmeler ve gelecek adımları
   1. Proje, temel işlevsellik açısından başarılı bir şekilde tamamlanmıştır. Ancak, ilerleyen dönemlerde model doğruluğunun artırılması ve sistemin daha fazla kamerayı desteklemesi hedeflenebilir.
   2. Ek olarak, daha geniş bir test süreci ve farklı senaryolar altında performans değerlendirmeleri yapılması önerilmektedir.

Sonuç olarak, proje başarılı bir şekilde tamamlanmış ve istenen hedeflere ulaşılmıştır. Geliştirilen sistem, bazı güvenlik ihtiyaçlarını karşılamak için kullanılabilir bir çözüm sunmaktadır. Gelecek çalışmalarda, sistemin performansını ve doğruluğunu artırmak için ek iyileştirmeler yapılması faydalı olacaktır.