

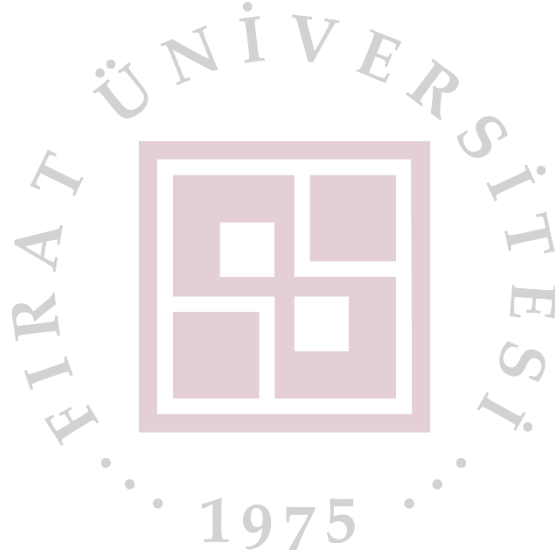
**T.C.
FIRAT ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

BMÜ-316 ALGORİTMA ANALİZİ PROJESİ

**DERİN ÖĞRENME YÖNTEMLERİ İLE ŞÜPHELİ DAVRANIŞ, NESNE
TESPİTİ VE ANALİZİ**

Proje Yazarı

DENİZ GÖL



Proje Danışmanı

Prof. Dr. MEHMET KARAKÖSE

2024

ELAZIĞ

ÖZGÜNLÜK BİLDİRİMİ

Bu çalışmada, başka kaynaklardan yapılan tüm alıntılar, ilgili kaynaklar referans gösterilerek açıkça belirtildiğini, alıntılar dışındaki bölümlerin, özellikle projenin ana konusunu oluşturan teorik çalışmaların ve yazılım/donanımın benim tarafımdan yapıldığını bildiririm.

Fırat Üniversitesi

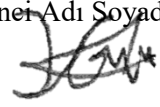
Bilgisayar Mühendisliği

23119 Elazığ

Tarih 08/06/2024

Öğrenci Adı Soyadı Deniz GÖL

İmza



İçindekiler

ÖZET	4
DERİN ÖĞRENME YÖNTEMLERİ İLE ŞÜPHELİ DAVRANIŞ, NESNE TESPİTİ VE ANALİZİ	4
ABSTRACT.....	5
1. GİRİŞ	6
1.1 Yapay zeka (AI).....	6
1.2 Temel Kavramlar ve Terimleri;	6
1.3 You Only Look Once (YOLO)	8
1.4 Şüpheli İnsan ve Nesne Analizi.....	9
1.5 Görüntü İşleme İle İlgili Çalışmalar	9
2. MATERYAL VE YÖNTEMLER.....	10
2.1 Kullanılan Veri Setleri.....	10
2.2 Kullanılan Yöntemler	12
2.3 Eğitim Sonuçları	15
2.4 Veri Setlerinin Kullanılması	22
2.5 Proje Python Kodları	23
3. BULGULAR VE SONUÇ	27
KAYNAKLAR	28

ÖZET

DERİN ÖĞRENME YÖNTEMLERİ İLE ŞÜPHELİ DAVRANIŞ, NESNE TESPİTİ VE ANALİZİ

Bu proje çalışmasında, derin öğrenme yöntemleri kullanılarak kalabalık analizinde şüpheli nesne takibi ve duran şüpheli kişi, araç veya nesne tespiti yapılmıştır. Araştırma modeli olarak YOLOv5 seçilmiş ve Google Colab ile Python kullanılmıştır. Veri toplama araçları olarak, Roboflow'dan çeşitli veri setleri alınmış ve bu veri setleri YOLOv5 için eğitilmiştir. Eğitim ve kodlama işlemleri Google Colab üzerinde gerçekleştirilmiştir.

Projenin Konusu: Bu çalışma, güvenlik ve izleme sistemlerinde kullanılmak üzere derin öğrenme yöntemleri ile kalabalıklar içinde şüpheli nesne ve kişi tespiti yapmayı amaçlamaktadır.

Problem: Kalabalık ortamlarda şüpheli nesnelerin veya kişilerin tespiti, güvenlik açısından büyük önem taşımaktadır. Geleneksel yöntemlerin yetersiz kaldığı bu alanda, derin öğrenme yöntemlerinin etkinliğini incelemek bu çalışmanın ana problemini oluşturmaktadır.

Araştırma Modeli: YOLOv5 modeli kullanılmıştır. YOLOv5, gerçek zamanlı nesne tespiti konusunda yüksek doğruluk ve hız sağlayan bir modeldir.

Bağımlı ve Bağımsız Değişkenler: Bu çalışmada bağımlı değişkenler, tespit edilen şüpheli nesne veya kişilerdir. Bağımsız değişkenler ise kullanılan veri setleri ve modelin eğitim sürecinde yapılan hiperparametre ayarlamalarıdır.

Hedef Kitle ve Örneklem: Çalışmanın hedef kitle, güvenlik sistemleri ve izleme teknolojileri üzerinde çalışan araştırmacılar ve mühendislerdir. Örneklem olarak, Roboflow'dan alınan ve şüpheli nesne, kişi ve araçları içeren veri setleri kullanılmıştır.

Verilerin Toplanması ve Analizi: Veriler, Roboflow platformundan indirilmiş ve YOLOv5 modeline uygun şekilde düzenlenmiştir. Modelin eğitimi ve değerlendirilmesi Google Colab üzerinde gerçekleştirilmiştir. Veri analizi sırasında modelin doğruluğunu artırmak için veri artırma teknikleri ve hiperparametre ayarlamaları yapılmıştır.

Tezin Sonuçları ve Önerileri: Çalışmanın sonuçları, YOLOv5 modelinin kalabalık ortamlarda şüpheli nesne ve kişi tespitinde oldukça başarılı olduğunu göstermektedir. Bu başarı, derin öğrenme yöntemlerinin güvenlik alanında etkin bir şekilde kullanılabileceğini ortaya koymaktadır. Gelecekteki çalışmalar için, daha büyük ve çeşitli veri setleri kullanılarak modelin performansının daha da artırılabilirliği öngörülmektedir.

ABSTRACT

Crowd Analysis Using Deep Learning Methods: Suspicious Object Tracking and Detection of Stationary Suspicious Persons, Vehicles, or Objects

In this thesis, crowd analysis is conducted using deep learning methods to track suspicious objects and detect stationary suspicious persons, vehicles, or objects. YOLOv5 was selected as the research model, and Google Colab and Python were utilized. Data sets were sourced from Roboflow and trained for YOLOv5. The training and coding processes were carried out on Google Colab.

Thesis Topic: This study aims to detect suspicious objects and persons in crowds using deep learning methods for use in security and surveillance systems.

Problem: Detecting suspicious objects or persons in crowded environments is crucial for security. The main problem of this study is to examine the effectiveness of deep learning methods in this area, where traditional methods fall short.

Research Model: YOLOv5 model was used. YOLOv5 provides high accuracy and speed in real-time object detection.

Dependent and Independent Variables: In this study, the dependent variables are the detected suspicious objects or persons. The independent variables are the data sets used and the hyperparameter adjustments made during the model training process.

Target Audience and Sample: The target audience of this study includes researchers and engineers working on security systems and surveillance technologies. The sample consists of data sets from Roboflow containing suspicious objects, persons, and vehicles.

Data Collection and Analysis: Data were downloaded from the Roboflow platform and formatted to suit the YOLOv5 model. The model training and evaluation were conducted on Google Colab. During data analysis, data augmentation techniques and hyperparameter adjustments were made to improve the model's accuracy.

Thesis Results and Recommendations: The results of the study indicate that the YOLOv5 model is highly successful in detecting suspicious objects and persons in crowded environments. This success demonstrates the effective application of deep learning methods in the security field. For future studies, it is recommended to use larger and more diverse data sets to further enhance the model's performance.

1. GİRİŞ

Derin öğrenme ve görüntü işleme, yapay zeka ve makine öğrenmesi alanlarında devrim niteliğinde ilerlemeler sağlayan teknolojilerdir [1]. Derin öğrenme, derin sinir ağları (deep neural networks) kullanarak, çok katmanlı yapay sinir hücreleri (neurons) aracılığıyla büyük veri kümelerinden (big data) anlamlı sonuçlar çıkarmayı hedefler [2]. Bu teknolojiye, ileri düzeyde hesaplama gücü gerektiren GPU'lar (grafik işleme birimleri) kullanılır ve modeller, geri yayılım (backpropagation) algoritması ile optimize edilir [3]. Görüntü işleme (image processing), dijital görüntülerin analiz edilmesi ve işlenmesi sürecini kapsar; bu süreçte, kenar algılama (edge detection), segmentasyon (segmentation), ve özellik çıkarımı (feature extraction) gibi teknikler kullanılır [4]. Bu teknolojiler, sağlık alanında tıbbi görüntüleme (medical imaging) teknikleriyle hastalık tespiti ve teşhisinde, otomotiv sektöründe otonom araçlar (autonomous vehicles) aracılığıyla çevre algılama (environmental perception) ve nesne tanıma (object recognition) için, güvenlikte yüz tanıma (facial recognition) sistemleriyle şüpheli davranışların tespiti için, tarımda bitki sağlığının izlenmesi ve hastalık tespiti için, eğlence sektöründe artırılmış gerçeklik (AR) ve sanal gerçeklik (VR) uygulamalarında, endüstride ise kalite kontrol (quality control) ve otomasyon (automation) süreçlerinde geniş bir yelpazede kullanılmaktadır [5]. Derin öğrenme ve görüntü işlemenin entegrasyonu, konvolüsyonel sinir ağları (CNNs) ve evrimsel katmanlar (convolutional layers) gibi ileri düzey tekniklerle gerçekleştirilir ve bu teknolojilerin gelecekte daha da yaygınlaşarak günlük yaşamın birçok alanında vazgeçilmez hale gelmesi beklenmektedir [6].

1.1 Yapay zeka (AI)

İnsan zekasını taklit etmeyi amaçlayan ve makinelerin insan benzeri bilişsel işlevleri gerçekleştirmesini sağlayan geniş bir bilgisayar bilimi alanıdır. Bu alan, makine öğrenmesi (ML), derin öğrenme (DL), doğal dil işleme (NLP), ve bilgisayarla görme (CV) gibi çeşitli alt dallardan oluşur [7].

1.2 Temel Kavramlar ve Terimleri;

Makine Öğrenmesi (ML):

Bilgisayarların belirli bir görevi, doğrudan programlanmadan, verilerden öğrenerek gerçekleştirmesini sağlayan yapay zeka alt dalıdır. Algoritmalar, veri kümesi üzerinden eğitim alarak örüntüleri tanır ve gelecekteki verilerle ilgili tahminler yapar [8].

Derin Öğrenme (DL):

Çok katmanlı yapay sinir ağları (deep neural networks) kullanarak verileri daha derin ve soyut seviyelerde işleyebilen bir tekniktir. Özellikle büyük veri setleriyle çalışmada etkilidir ve görüntü tanıma, ses tanıma gibi alanlarda başarı sağlar [9].

Doğal Dil İşleme (NLP):

İnsan dilinin bilgisayarlar tarafından anlaşılması, işlenmesi ve üretilmesi ile ilgilenen yapay zeka alanıdır. Metin ve konuşma tanıma, dil çevirisi, duygu analizi gibi uygulamaları içerir [10].

Bilgisayarla Görme (CV):

Dijital görüntülerin ve videoların analiz edilmesi ve işlenmesi ile ilgilenen yapay zeka alanıdır. Nesne tanıma, yüz tanıma, görüntü sınıflandırma gibi görevleri içerir [11].

Yapay Sinir Ağları (ANN):

İnsan beyninin sinir hücrelerinin işleyişini taklit eden algoritmalarlardır. Giriş verilerini işleyerek çıkış üretirler. Makine öğrenmesi ve derin öğrenmede temel yapı taşlarıdır [12].

Konvolüsyonel Sinir Ağları (CNNs):

Özellikle görüntü işleme ve bilgisayarla görme alanlarında kullanılan derin öğrenme modelleridir. CNN katmanları aşağıda verilmiştir. [13].

- 1. Girdi Katmanı (Input Layer):** Farklı formatlarda (resim, metin, ses vb.) veriyi kabul eder.
- 2. Evrişim Katmanı (Convolutional Layer):** Görüntüdeki kenarlar, köşeler gibi özellikleri çıkarmak için filtreler kullanır.
- 3. Aktivasyon Katmanı (Activation Layer):** Sisteme doğrusallık dışı bir işlev ekler.
- 4. Havuzlama Katmanı (Pooling Layer):** Veri boyutunu azaltır ve önemli özellikleri vurgular.
- 5. Tam Bağlantılı Katman (Fully Connected Layer):** Farklı nöronlar arasında bağlantı kurarak soyutlama yapar ve sınıflandırma gibi işlemleri gerçekleştirir.
- 6. Çıktı Katmanı (Output Layer):** Sınıflandırma sonucunu belirli bir formatta (örneğin, olasılık dağılımı) sunar.

Geri Yayılım (Backpropagation):

Yapay sinir ağlarının eğitiminde kullanılan bir algoritmadır. Ağırlıkları güncellemek için kullanılır [14].

Büyük Veri (Big Data):

Hacmi çok büyük, hızla üretilen ve çeşitli kaynaklardan gelen veri setlerini ifade eder. Anlamlı bilgi çıkarımı yapmak için kullanılır [15].

GPU (Grafik İşleme Birimi):

Derin öğrenme modellerinin eğitiminde kullanılan donanım birimleridir. Yüksek paralel işlem kapasitesi ile büyük veri setlerini hızlı bir şekilde işleyebilirler.

Bu terimler ve kavramlar, yapay zekanın temelini oluşturur ve birlikte kullanılarak çeşitli uygulamalarda etkili çözümler sağlarlar. Yapay zekanın, sağlık, finans, eğitim, ulaşım ve daha birçok sektörde devrim niteliğinde ilerlemeler kaydederek, gelecekte daha da yaygınlaşması beklenmektedir.

1.3 You Only Look Once (YOLO)

You Only Look Once (YOLO), bilgisayarla görme (computer vision) alanında nesne tespiti için kullanılan bir derin öğrenme algoritmasıdır. YOLO'nun temel amacı, bir görüntüdeki nesneleri tespit etmek ve bu nesnelerin sınırlayıcı kutularını (bounding boxes) çizmek için tek bir geçişte yüksek doğruluk sağlamaktır. Bu, diğer geleneksel nesne tespit yöntemlerine kıyasla daha hızlı ve etkili bir yaklaşımdır [16].

YOLO, birbirine bağlı bir dizi konvolüsyonel ve tam bağlantılı katmanlar içeren bir evrişimli sinir ağı (CNN) kullanır. Giriş görüntüsü, ağ boyunca ilerlerken farklı ölçeklerde öznitelik haritaları oluşturulur ve nesnelerin tahmin edilmiş sınırlayıcı kutularını ve sınıflarını belirlemek için bu öznitelik haritaları kullanılır.

YOLO'nun ana avantajlarından biri, tek bir geçişte tüm nesneleri tespit edebilmesidir, bu da hız ve verimlilik açısından önemlidir. Ayrıca, nesnelerin piksel seviyesinde kesişmeleriyle ilgili sorunları minimize eden bir non-maximum suppression (NMS) algoritması içerir.

YOLOv1

YOLO'nun orijinal sürümüdür. 2015 yılında Joseph Redmon ve diğerleri tarafından geliştirilmiştir. Nesne tespiti için tek bir YOLO katmanı kullanır.

YOLOv2 (YOLO9000)

YOLO'nun ikinci sürümüdür. 2016'da Joseph Redmon ve Ali Farhadi tarafından geliştirilmiştir. Geliştirilmiş bir CNN mimarisi kullanır ve daha iyi performans sağlar.

YOLOv3

YOLO'nun üçüncü sürümüdür. 2018'de Joseph Redmon ve Ali Farhadi tarafından geliştirilmiştir. YOLOv2'ye kıyasla daha hızlı ve daha hassas bir model sunar.

YOLOv4

YOLO'nun dördüncü sürümüdür. 2020'de Alexey Bochkovskiy ve diğerleri tarafından geliştirilmiştir. YOLOv3'ten daha hızlı ve daha hassas sonuçlar elde eder.

YOLOv5

YOLO'nun beşinci sürümüdür. 2020'de Glenn Jocher tarafından başlatılmıştır. Bu sürüm, önceki sürümlerden daha hızlı ve daha hafif olup, kullanıcı dostu bir Application Programming Interface (API) ve kolay kullanım sunar.

1.4 Şüpheli İnsan ve Nesne Analizi

Şüpheli insan davranışı ve nesnelerin tespiti, kamusal güvenliğin sağlanması ve potansiyel tehditlerin önlenmesi amacıyla kullanılan kritik bir görüntü işleme alanıdır. Bu süreç, öncelikle kameralar veya diğer görüntü sensörleriyle çevredeki görüntülerin toplanmasıyla başlar. Toplanan görüntüler daha sonra öznitelik çıkarımı adımıyla analiz edilir, bu da insanların veya nesnelerin belirli özelliklerini tanımlayan verilerin elde edilmesini sağlar. Daha sonra, derin öğrenme ve yapay zeka teknikleri kullanılarak bu özniteliklerden öğrenme modelleri oluşturulur. Oluşturulan modeller, gerçek zamanlı olarak elde edilen görüntüler üzerinde uygulanır ve şüpheli davranışlar veya nesneler tespit edildiğinde bir uyarı üretilir. Bu süreç, suç önleme, terörle mücadele ve güvenlik ihlallerinin azaltılması gibi alanlarda önemli bir rol oynar ve toplum güvenliği için kritik bir öneme sahiptir.

1.5 Görüntü İşleme İle İlgili Çalışmalar

Büyük ölçekli çalışmalar, pozitif tanımlamada önemli ilerleme kaydetmiştir. Bu çalışmalarda, genellikle halka açık alanlarda toplanan ve şüpheli ve normal davranış örneklerini içeren büyük video veri kümeleri kullanılır. Bu veriler, şüpheli davranışı otomatik olarak tanımlayabilen derin öğrenme modelleri oluşturmak için kullanılır.

UC Merced Havalimanı Veri Kümesi:

Bu veri kümesi, 200 saatten fazla video ve 100.000'den fazla insan davranışı örneği içerir. Veri kümesi, şüpheli davranışı otomatik olarak tanımlamak için derin öğrenme modelleri geliştirmek için kullanılmıştır [17].

Stanford Egocentric Perception Veri Kümesi:

Bu veri kümesi, birinci şahıs bakış açısından çekilen 300 saatten fazla video ve 50.000'den fazla insan davranışı örneği içerir. Veri kümesi, şüpheli davranışı otomatik olarak tanımlamak için birinci şahıs bakış açısından çekilen videoları kullanma potansiyelini keşfetmek için kullanılmıştır [18].

Charades Veri Kümesi:

Bu veri kümesi, 10.000'den fazla insan tarafından canlandırılan 300'den fazla kelimeyi içeren 300.000 video klipten oluşur. Veri kümesi, şüpheli davranışı otomatik olarak tanımlamak için insan davranış modellerini öğrenmek için kullanılmıştır [19]

2. MATERYAL VE YÖNTEMLER

Projem üç aşamadan oluşmaktadır. İlk aşamada, gerekli veri setlerini hazırlamak için çeşitli kaynaklardan görüntüler toplandı ve bu görüntüler insan davranışı ve şüpheli nesnelerin çeşitli örneklerini içerecek şekilde etiketlendi [20] [21]. İkinci aşamada, hazırlanan veri setleri kullanılarak derin öğrenme algoritmalarıyla eğitim gerçekleştirildi. Bu aşamada, özellikle nesne tespiti ve insan davranışı tanıma konularında uzmanlaşmış derin öğrenme modelleri tercih edildi [22] [23]. Son olarak, eğitilen modeller YOLOv5 gibi modern nesne tespiti algoritmalarıyla kullanılarak gerçek zamanlı görüntü analizi süreci gerçekleştirildi. [24] [25].

2.1 Kullanılan Veri Setleri

Aggressive Behavior Dataset Veri Seti

529 resimden oluşan ve alanında saldırgan davranışları tespit etmek için kullanılan bir veri setidir [26].

Veri Setinin Özellikleri:

Format: TFRecord, CreateML JSON

Average Image Size: 2.07 mp (Megapiksel)

Median Image Ratio: 1920x1080

1234 Computer Vision Project

425 resimden oluşan ve alanında çeşitli sırt çantalarını tespit etmek için kullanılan veri setidir [27].

Veri Setinin Özellikleri:

Average Image Size: 0.41 mp

Median Image Ratio: 640x640

Bump cap

665 resimden oluşan ve alanında çeşitli şapkaları tespit etmek için kullanılan veri setidir [28].

Average Image Size: 0.05 mp

Median Image Ratio: 275x192

Glasses Detector Computer Vision Project

363 resimden oluşan ve alanında çeşitli gözlükleri tespit etmek için kullanılan veri setidir [29].

Average Image Size: 0.05 mp

Median Image Ratio: 224x224

Anamoly detection Computer Vision Project

838 resimden oluşan ve alanında kavga eden insan olup olmadığını tespit etmek için kullanılan veri setidir [30].

Average Image Size: 0.23 mp

Median Image Ratio: 640x360

Packages on Conveyor Computer Vision Project

2320 resimden oluşan ve alanında çeşitli kutları tespit etmek için kullanılan veri setidir [31].

Average Image Size: 0.96 mp

Median Image Ratio: 1036x930

Guns-Knives Computer Vision Project

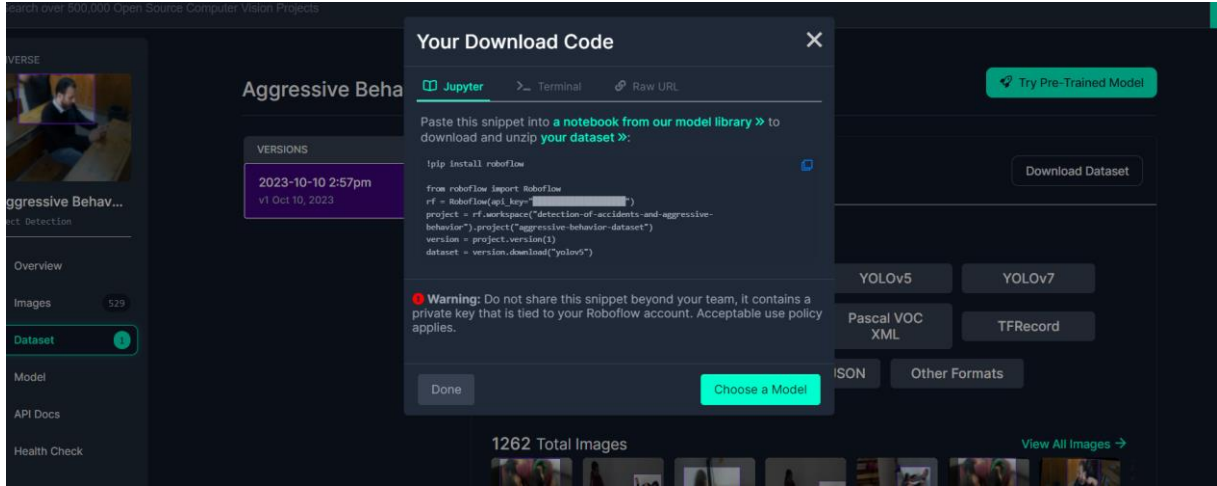
6918 resimden oluşan ve çeşitli silah ve bıçak türlerini tespit etmek için kullanılan veri setidir [32].

Average Image Size: 0.41 mp

Median Image Ratio: 640x640

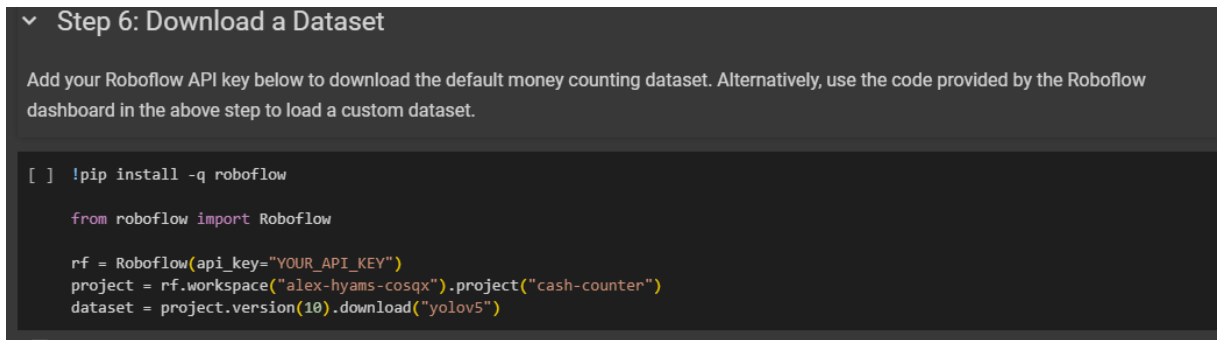
2.2 Kullanılan Yöntemler

- Bu bölümde kullanacağım veri setlerinin “YOLOv5-Custom-Training.ipynb [33]” not defterini (notebook) kullanarak eğitimlerini gerçekleştireceğim. İlk aşamada Şekil 1.1 de görüldüğü gibi Roboflow sitesinden kullanacağımız veri setinin YOLOv5 formatında indirme kodunu alıyoruz.



Şekil 1.1 YOLOv5 Formatında İndirme Kodunun Alınması

- Şekil 1.2'de görüldüğü üzere, Notebook' ta bulunan "Step 6: 'Download a Dataset'" hücresindeki kodu Roboflow' dan alınan ilgili kodu kullanarak değiştiriyoruz.



Şekil 1.2 Kodun Değiştirilmesi

- İlk hücreden başlayarak tüm hücreleri çalıştır seçeneğine basarak not defteri dosyasını Colab üzerinden çalıştırıyoruz. Bu aşama uygun bir makineye bağlanmanız halinde, veri setimizin ve epochs parametreniz büyüklüğü ile doğru orantılı olarak uzun sürecektir. Epoch sayısı, modelin eğitim veri seti üzerinden kaç kez tam geçiş yapacağını belirten bir parametredir.

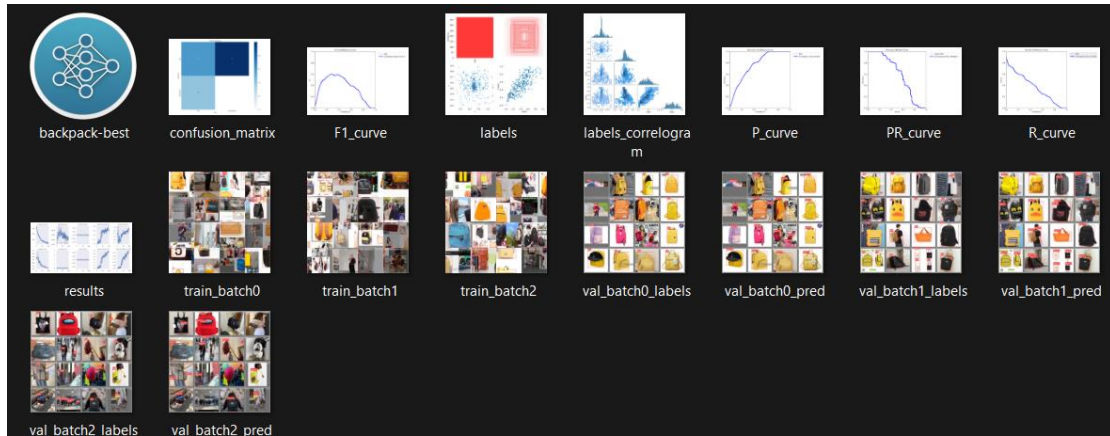
- Şekil 1.3'te görünen hücredeki kod tamamlandıktan sonra “yolov5s_results” adlı bir deney olarak kaydedilir. Bu deneyin içinde ağırlık sonuçlarından en iyi (best) ağırlık dosyadan seçilerek ilgili makineye indirilir.

```
# train yolov5s on custom data for 100 epochs
# time its performance
%%time
%cd /content/yolov5/
!python train.py --img 416 --batch 16 --epochs 100 --data {dataset.location}/data.yaml --cfg ./models/custom_yolov5s.yaml --weights '' --name yolov5s_results --cache
```

	77/99	1.84G	0.0254	0.02536	0.021	38	416: 100%	167/167	[00:28:00:00, 5.79it/s]
Class	all	249	1379	0.487	0.78	0.602	0.409		

Şekil 1.3 En İyi Ağırlığın Elde Edilmesi

- Sonuçlar klasörünün içinde Şekil 1.4 te gözüktüğü gibi confusion matrix¹, F1 curve², labels³, labels_correlogram⁴, P_curve⁵, PR_curve⁶, R_curve⁷, results⁸, train_batch⁹, val_batch0_labels⁹ gibi sonuçlar oluşacaktır. Bu değerler kontrol edilerek modelimiz hakkındaki detaylı bilgileri öğrenebiliriz.



Şekil 1.4 Sonuç Klasörü

Makine Öğrenimi Modeli Değerlendirme Metrikleri ve Görselleştirmeleri

1. Karmaşıklık Matrisi (Confusion Matrix): Bu tablo, modelin tahmin ettiği sınıfları gerçek sınıflarla karşılaştırır. Her satır, modelin belirli bir sınıfı nasıl tahmin ettiğini gösterirken, her sütun gerçek sınıfı temsil eder. Köşegen değerleri, modelin doğru tahmin ettiği örneklerin sayısını gösterirken, diğer hücreler hatalı tahminleri gösterir. Karmaşıklık matrisi, modelin hangi sınıflarda daha iyi veya daha kötü performans gösterdiğini görselleştirmek için kullanılır.

2. F1 Eğrisi (F1 Curve): Bu grafik, modelin F1 skorunu farklı eşik değerlerine göre gösterir. F1 skoru, bir modelin sınıflandırma performansını ölçen bir metriktir ve hassasiyet (precision) ve hatırlama (recall) değerlerinin harmonik ortalamasıdır. F1 eğrisi, modelin en yüksek F1 skorunu hangi

eşik değeriinde elde ettiğini gösterir ve farklı eşik değeriinde modelin performansının nasıl değıştiğini görselleştirmek için kullanılır.

3. Etiketler (Labels): Bu tablo, eğitim veri setindeki etiketlerin dağılımını gösterir. Her etiket, veri setinde kaç örneğe karşılık geldiğini gösterir. Etiket dağılımı, modelin hangi sınıflarda daha fazla veya daha az veriye sahip olduğunu görselleştirmek için kullanılır.

4. Etiket Korelogramı (Labels Correlogram): Bu grafik, etiketler arasındaki ilişkileri ve ortaklıkları gösterir. Her hücre, iki etiketin birlikte ne sıklıkla görüldüğünü gösterir. Korelogram, veri setindeki etiketler arasındaki bağımlılıkları ve ilişkileri görselleştirmek için kullanılır.

5. Hassasiyet Eğrisi (P Curve): Bu grafik, modelin hassasiyetini (precision) farklı eşik değeriine göre gösterir. Hassasiyet, modelin doğru olarak tahmin ettiğı örneklerin oranını temsil eder. P eğrisi, modelin en yüksek hassasiyet seviyesini hangi eşik değeriinde elde ettiğini gösterir ve farklı eşik değeriinde modelin hassasiyetinin nasıl değıştiğini görselleştirmek için kullanılır.

6. Hassasiyet-Hatırlama Eğrisi (PR Curve): Bu grafik, modelin hassasiyet (precision) ve hatırlama (recall) oranlarını birlikte gösterir. Hatırlama, modelin tüm doğru örnekleri ne kadar iyi tahmin ettiğini temsil eder. PR eğrisi, modelin hassasiyet ve hatırlama arasında nasıl bir denge kurduğunu gösterir ve modelin hangi eşik değeriinde en iyi dengeyi sağladığını görselleştirmek için kullanılır.

7. Hatırlama Eğrisi (R Curve): Bu grafik, modelin hatırlamasını (recall) farklı eşik değeriine göre gösterir. R eğrisi, modelin en yüksek hatırlama seviyesini hangi eşik değeriinde elde ettiğini gösterir ve farklı eşik değeriinde modelin hatırlamasının nasıl değıştiğini görselleştirmek için kullanılır.

8. Sonuçlar (Results): Bu dosya, eğitim sürecinin genel performans metriklerini ve sonuçlarını içerir. Bu dosyada, eğitim ve doğrulama veri setleri için hassasiyet, hatırlama, F1 skoru, kayıp fonksiyonu değeri gibi birçok metrik bulunur. Sonuçlar dosyası, modelin genel performansını değerlendirmek ve farklı modeller arasında karşılaştırma yapmak için kullanılır.

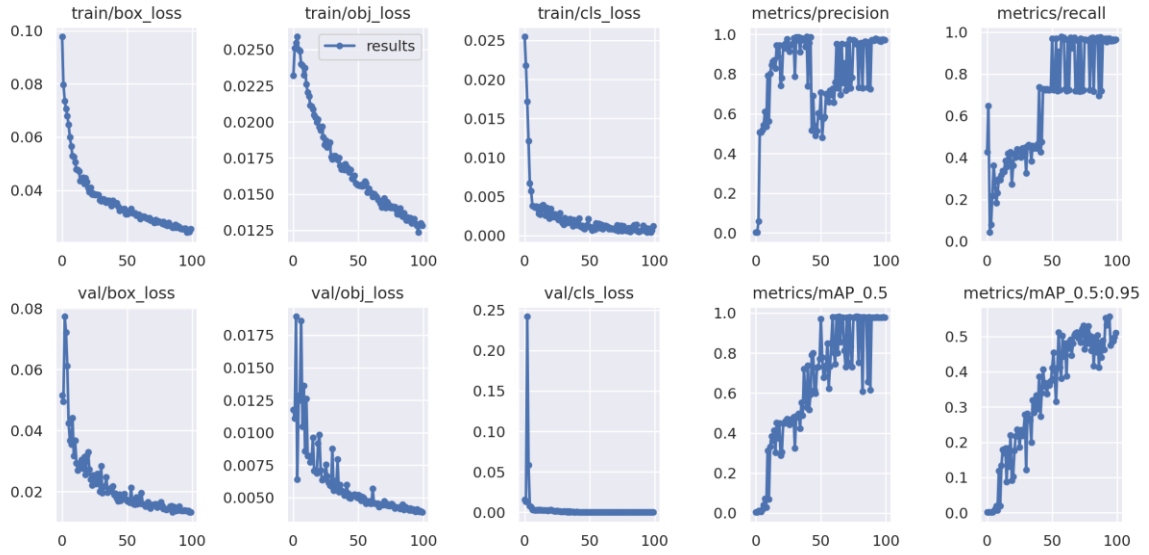
9. Eğitim Batch'i (Train Batch): Bu terim, eğitim veri setindeki örneklerin bir batch içerisindeki görünümünü temsil eder. Eğitim sırasında, model her seferinde bir batch'teki örnekleri görerek parametrelerini günceller. Batch boyutu, modelin hafıza kullanımı ve eğitim süresi üzerinde önemli bir etkiye sahip olabilir.

10. Doğrulama Batch'i 0 Etiketleri (Val Batch0 Labels): Bu tablo, doğrulama veri setindeki ilk batch'teki örneklerin etiketlerini gösterir. Doğrulama veri seti, modelin eğitim verisi üzerinde aşırı uyum (overfitting) yapıp yapmadığını kontrol etmek için kullanılır. Doğrulama veri setindeki etiketler, modelin performansını değerlendirmek için kullanılır.

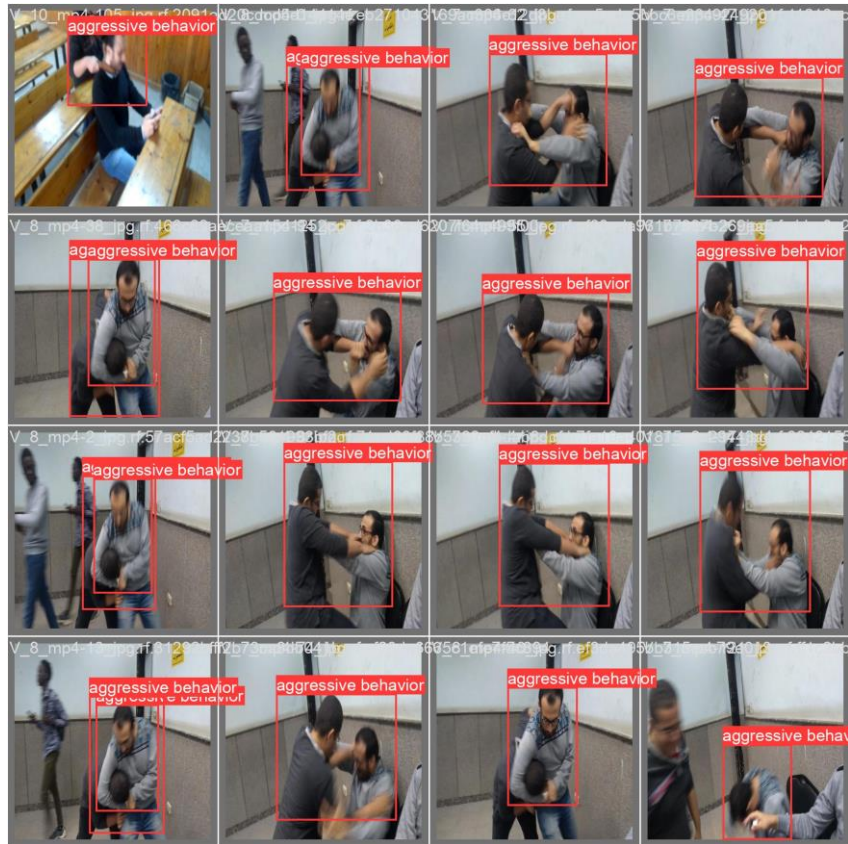
2.3 Eğitim Sonuçları

Bu proje için eğittiğim veri setlerinin results ve val_batch0_labels sonuçları aşağıda verilmiştir.

Aggressive Behavior Dataset Veri Seti

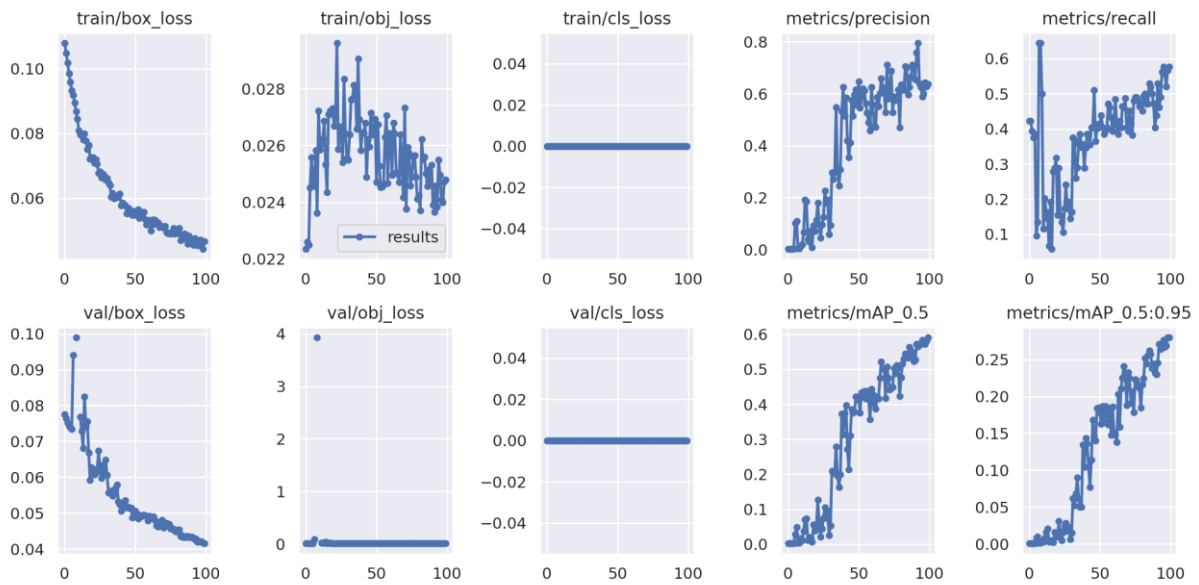


Şekil 1.5 Results



Şekil 1.6 val_batch0_labels

1234 Computer Vision Project

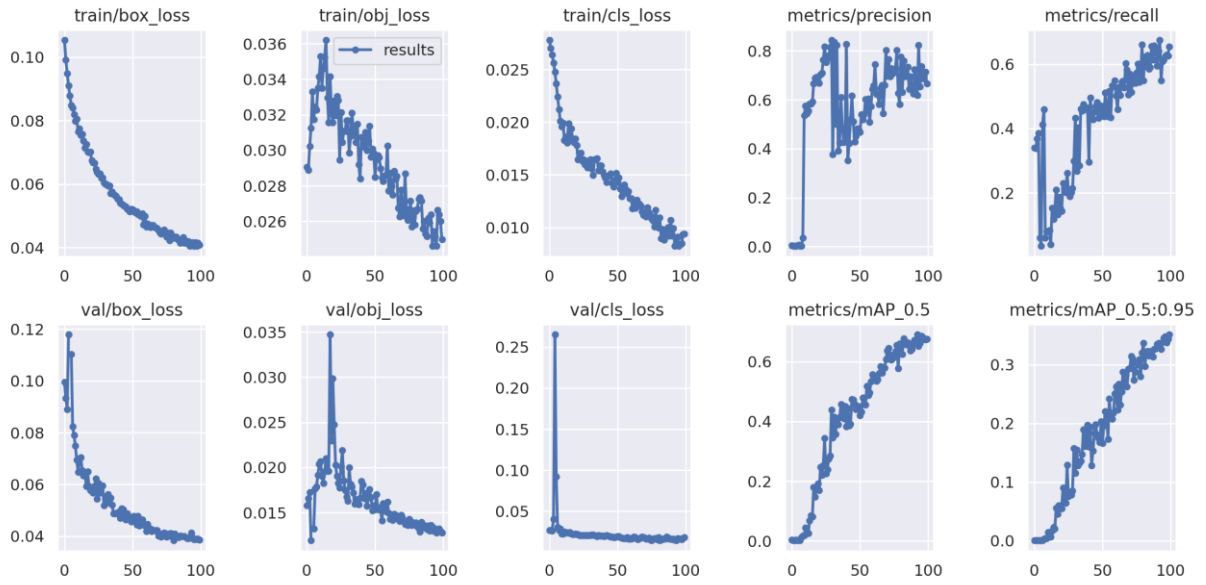


Şekil 1.7 Results

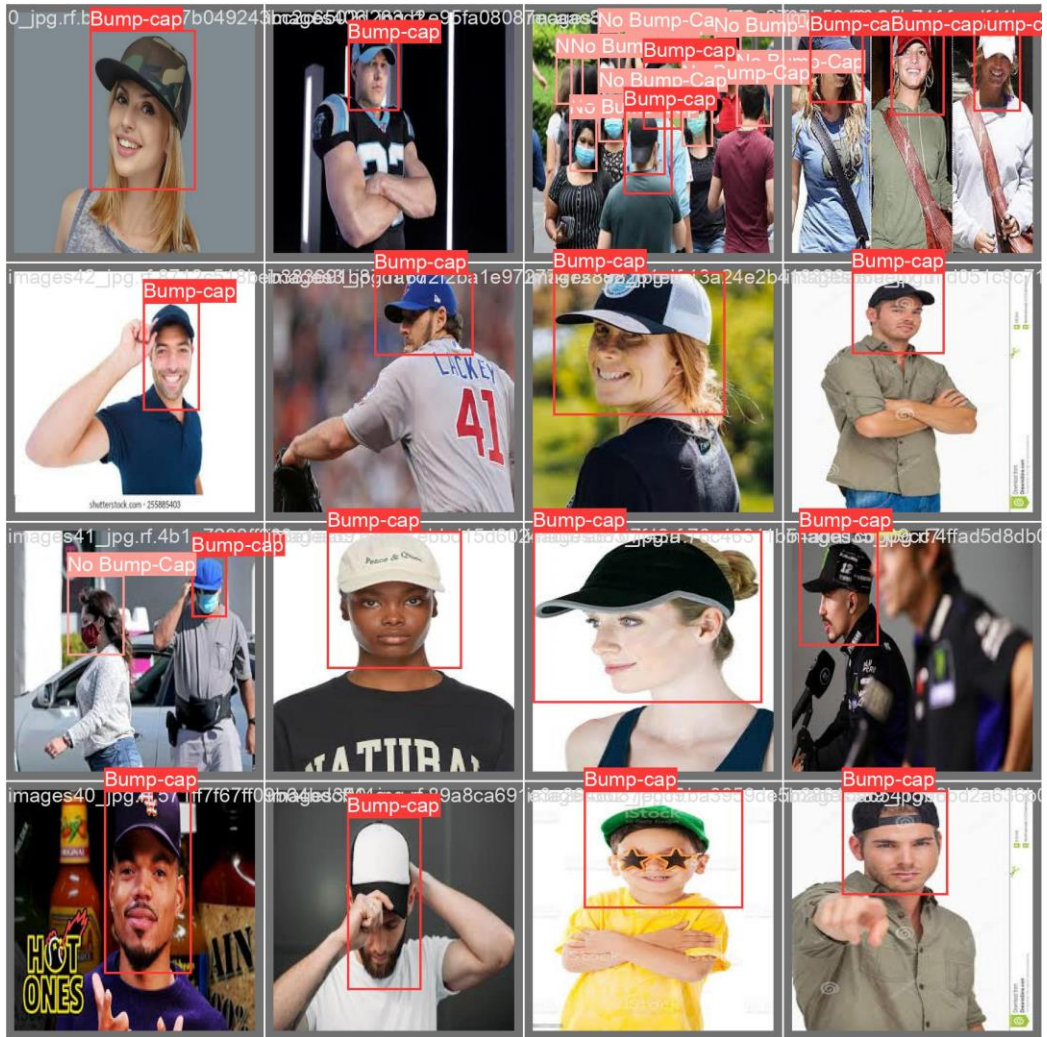


Şekil 1.8 val_batch0_labels

Bump cap

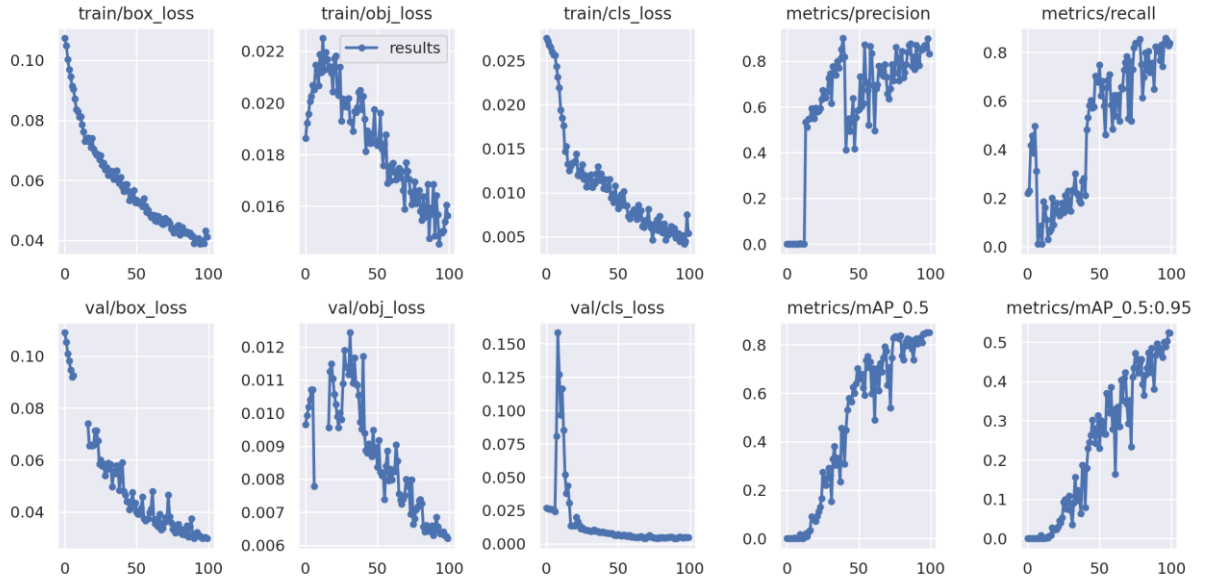


Şekil 1.9 Results



Şekil 2 val_batch0_labels

Glasses Detector Computer Vision Project

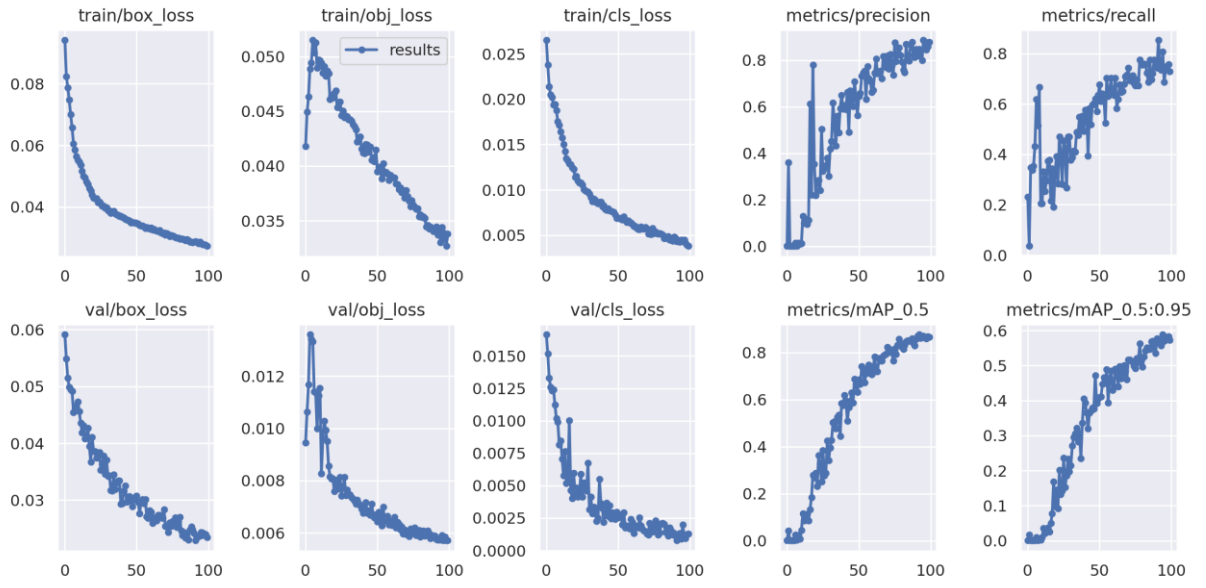


Şekil 2.1 Results



Şekil 2.2 val_batch0_labels

Anamoly detection Computer Vision Project

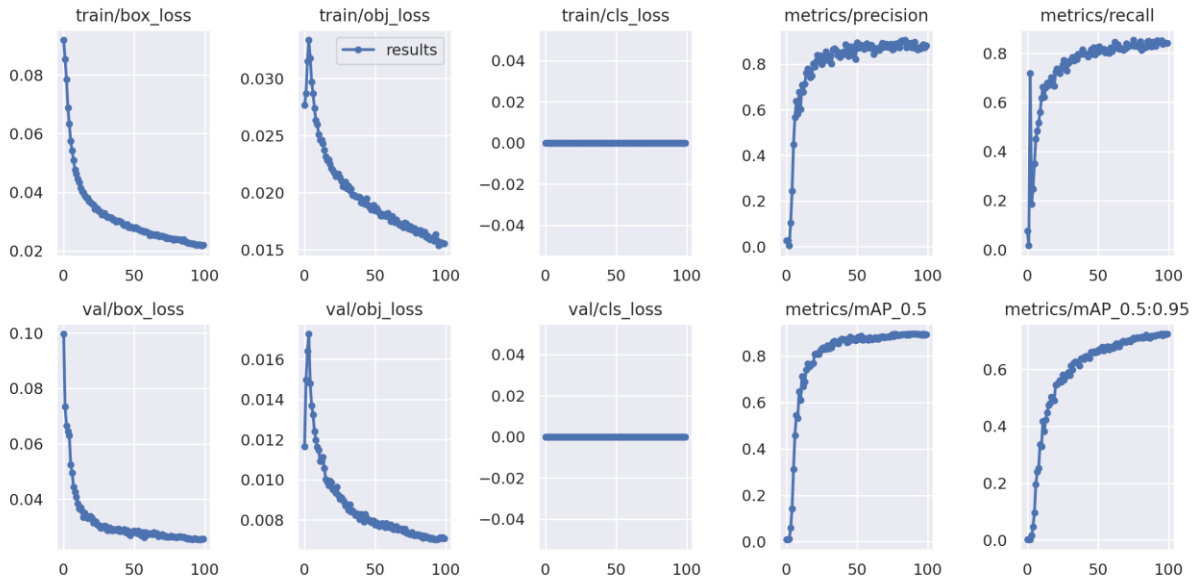


Şekil 2.3 Results

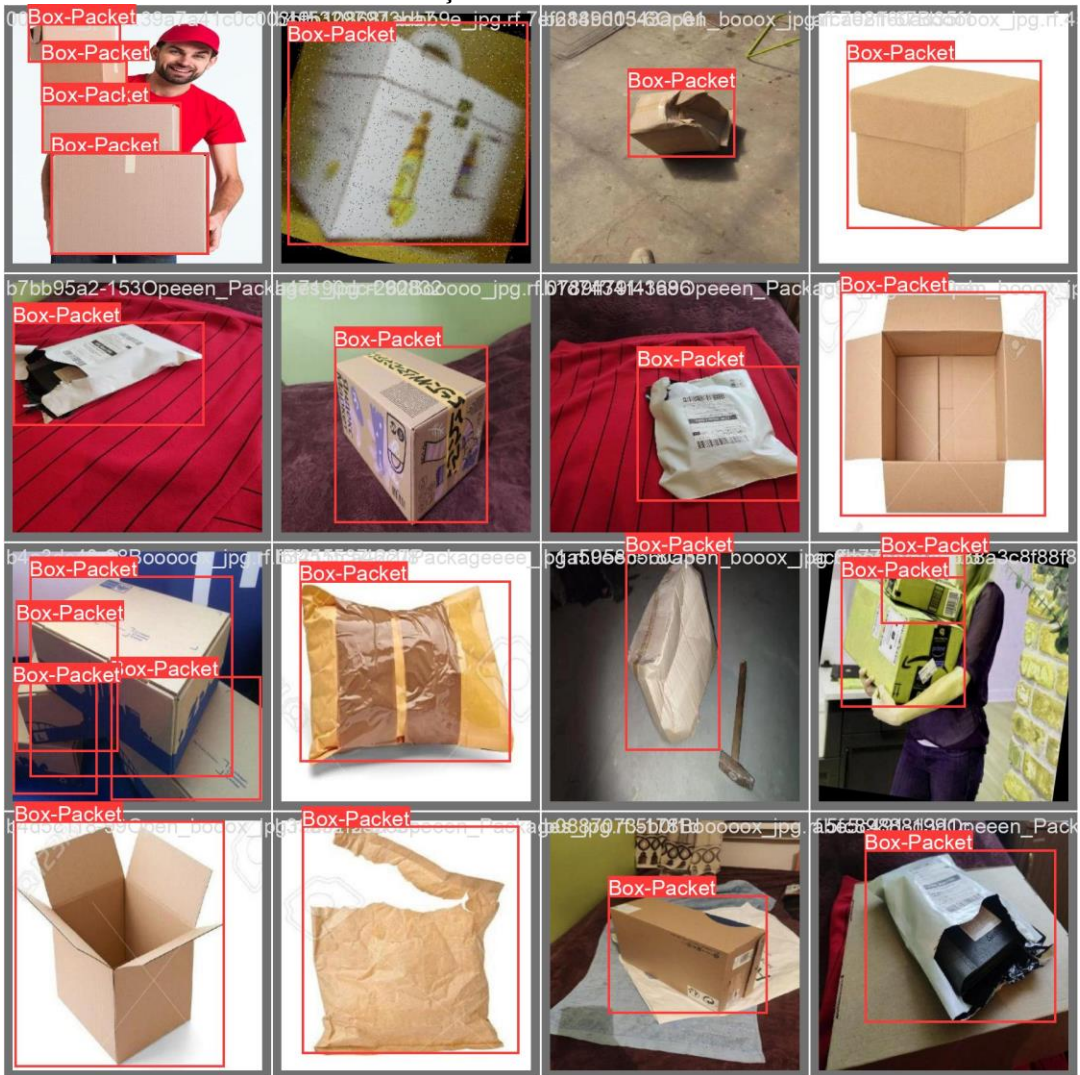


Şekil 2.4 val_batch0_labels

Packages on Conveyor Computer Vision Project

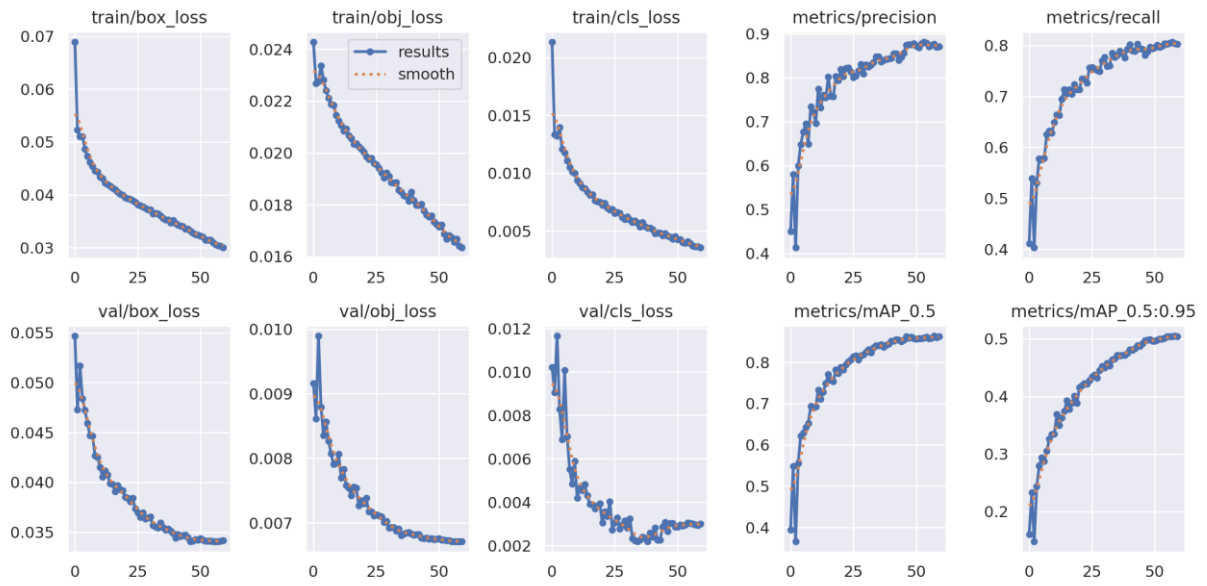


Şekil 2.5 Results

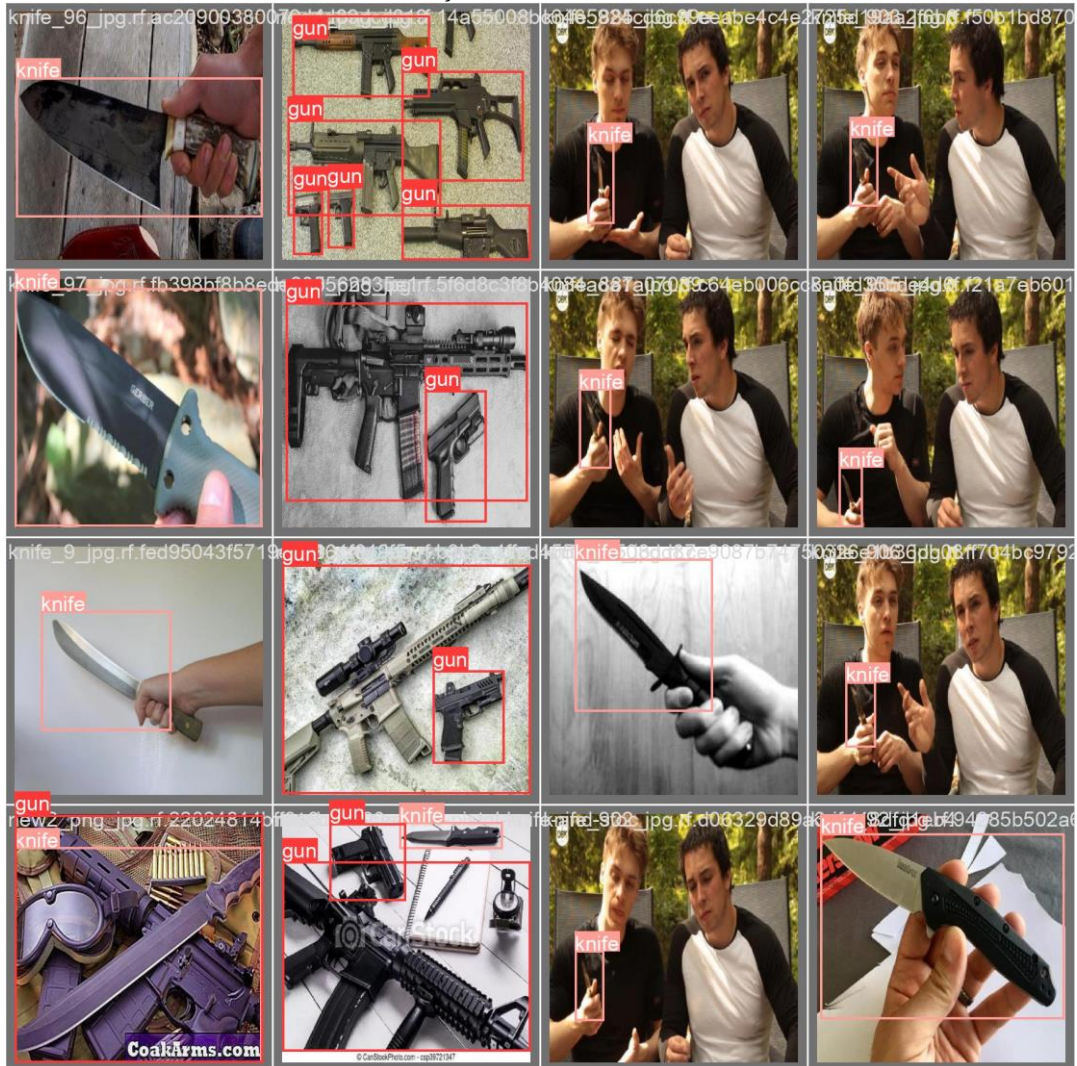


Şekil 2.6 val_batch0_labels

Guns-Knives Computer Vision Project



Şekil 2.7 Results



Şekil 2.8 val_batch0_labels

2.4 Veri Setlerinin Kullanılması

- Veri setlerinin kullanılabilmesi için gerekli YOLOv5 modüllerini ve Python kütüphaneleri yüklüyorum.
- Eğitimini yaptığım veri setlerinin en iyi ağırlıklarını ve işlenmesini istediğim görüntüleri makinede oluşturduğum bir klasöre ekleyerek not defteri (notebook) içinde çağırıyorum.
- Hücreyi bir kez çalıştırıp gerekli yüklemeler hazırlandıktan sonra sanal makineyi tekrar başlatıp modüllerin makinede çalışması için gerekli koşulları tamamliyorum.

- Ağırlık kodları çalıştıktan sonra oluşan Labels dosyalarından gelen sonuçların dizinlerini labels_dir adı altında topladım. Labels, bir görüntüde bulunan nesnelerin konumlarını ve sınıflarını içerir. Her bir nesne için, etiket dosyası genellikle bir satırda bir giriş içerir.

Şekil 2.9 da örnek bir labels dosyasının içi verilmiştir. Bu örnekte, ilk satırda bir nesne var ve sınıfı "0" (örneğin, bir araba) ve sınırlayıcı kutusu (x, y, genişlik, yükseklik) 0.12, 0.34, 0.56 ve 0.78 koordinatlarında tanımlanmıştır. İkinci satırda ise sınıfı "1" (örneğin, bir insan) ve sınırlayıcı kutusu 0.45, 0.67, 0.89 ve 0.12 koordinatlarında tanımlanmıştır.

```
0 0.12 0.34 0.56 0.78
1 0.45 0.67 0.89 0.12
```

Şekil 2.9 Örnek Labels Dosyası

- İşlenecek görüntünün yol bilgisini ve boyut gibi diğer bilgilerini tanımladım.
- YOLOv5 içinde bulunan hazır bulunan parametrelerden yararlanarak insanları ve arabaları sayan (araba çeşitleri ek olarak gösteriliyor) bir kod yazdım. Koda insanların ve araçların görüntü üzerinde anlık gösterilmesi için sınırlayıcı kutu çizme parametreleri ekledim. Güvenilirliği yükseltmek adına eşik değeri sınırlaması ekledim.
- Labellardan gelen .txt uzantılı dosyaları okumak için gerekli bağlantıları ekledim. Bu dosyaları ekrana yazdırmak ve anlık olarak göstermek için farklı renkler ve koordinatlar girdim.
- Labellardan gelen içerikleri, obje sayısı, bıçak veya silah gibi çeşitli sınırlandırmalar uygulayarak işlenmiş kareleri anlık olarak gösteren kodları ekledim.

2.5 Proje Python Kodları

```
import os

# Gerekli modülleri yükleme
!pip install gitpython

# Git deposunun URL'si
repo_url = 'https://github.com/ultralytics/yolov5'

# Depoyu klonlama
!git clone $repo_url

# Dizin değiştirme yolov5
os.chdir('yolov5')

# Gereksinimleri yükleme
!pip install -r requirements.txt

# Gerekli ağırlıklar, videolar ve sonuçların ayarlanması
!python detect.py --weights /content/sample_data/fight-best.pt --source
/content/sample_data/suspicious_image_live.mp4 --project
/content/sample_data --name results --save-txt --save-conf
!python detect.py --weights /content/sample_data/box-best.pt --source
/content/sample_data/suspicious_image_live.mp4 --project
/content/sample_data --name results1 --save-txt --save-conf
!python detect.py --weights /content/sample_data/agresif-best.pt --
source /content/sample_data/suspicious_image_live.mp4 --project
/content/sample_data --name results2 --save-txt --save-conf
!python detect.py --weights /content/sample_data/cap-best.pt --source
/content/sample_data/suspicious_image_live.mp4 --project
/content/sample_data --name results3 --save-txt --save-conf
!python detect.py --weights /content/sample_data/glasses-best.pt --
source /content/sample_data/suspicious_image_live.mp4 --project
/content/sample_data --name results4 --save-txt --save-conf
!python detect.py --weights /content/sample_data/backpack-best.pt --
source /content/sample_data/suspicious_image_live.mp4 --project
/content/sample_data --name results5 --save-txt --save-conf
!python detect.py --weights /content/sample_data/weapon-knife.pt --
source /content/sample_data/suspicious_image_live.mp4 --project
/content/sample_data --name results6 --save-txt --save-conf
```

```

import cv2
import numpy as np
import torch
from google.colab.patches import cv2_imshow
import os
import re

# YOLOv5 modelini yükle
model = torch.hub.load('ultralytics/yolov5', 'yolov5s')

# Sınıf isimlerini yükle
classes = model.names

# Etiket dosyalarının bulunduğu dizinler
label_dirs = [
    '/content/sample_data/results/labels', # fight detected
    '/content/sample_data/results1/labels', # package detected
    '/content/sample_data/results2/labels', # aggressive detected
    '/content/sample_data/results3/labels', # cap detected
    '/content/sample_data/results5/labels', # glasses detected
    '/content/sample_data/results6/labels', # backpack detected
    '/content/sample_data/results4/labels' # weapon detected
]

# Video dosyasının yolu
video_file = '/content/sample_data/suspicious_image_live.mp4'

# Videoyu aç
cap = cv2.VideoCapture(video_file)

# Video genişliği ve yüksekliği
width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))

# Her kareyi işle
while cap.isOpened():
    # Bir frame al
    ret, frame = cap.read()

    # Eğer frame alınamazsa ya da video biterse döngüyü kır
    if not ret:
        break

    # Kareyi YOLOv5'e besle
    results = model(frame)

    # Tahminleri al
    detections = results.pred[0]

```



```

# Kişi ve araç sayılarını tutmak için sayaçlar
person_count = 0
vehicle_count = 0

# Her tahmini işle
for detection in detections:
    # Sınıf etiketini ve güven puanını al
    class_id = int(detection[5])
    confidence = detection[4]

    # Sadece belirli bir güven eşiği üzerindeki tahminleri işle
    if confidence > 0.3: # Güven eşiği burada 0.3 olarak alındı
        # Kişi ve araçları say
        if classes[class_id] == 'person':
            person_count += 1
        elif classes[class_id] == 'car' or classes[class_id] ==
'truck':
            vehicle_count += 1

        # Nesnenin sınırlayıcı kutusunu çiz
        x1, y1, x2, y2 = int(detection[0]), int(detection[1]),
int(detection[2]), int(detection[3])
        cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)

        # Kişi ve araç sayısını ekrana yazdır (sol alt köşe)
        cv2.putText(frame, f"Person : {person_count}", (10, height - 40),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)
        cv2.putText(frame, f"Vehicle : {vehicle_count}", (10, height - 20),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)

    # Etiket dosyalarını bul
    label_files = [None] * len(label_dirs)
    frame_count = int(cap.get(cv2.CAP_PROP_POS_FRAMES))

    for i, label_dir in enumerate(label_dirs):
        for file in os.listdir(label_dir):
            if
re.match(rf'suspicious_image_live_{frame_count}\d*\\.txt', file):
                label_files[i] = os.path.join(label_dir, file)
                break

y_offset = 30
colors = [
    (0, 0, 255), # Fight Detected
    (0, 255, 0), # Object Count
    (0, 255, 255), # Suspicious Activity Detected
    (255, 0, 255), # Cap Detected
    (0, 255, 0), # Bag Detected

```

```

        (255, 255, 0), # Weapon Detected
        (255, 0, 0) # Glasses Detected
    ]
    labels = [
        "Fight Detected",
        "Package Detected: {}",
        "Suspicious Activity Detected",
        "Cap Detected: {}",
        "Bag Detected: {}",
        "Weapon Detected: {}",
        "Glasses Detected: {}"
    ]

    # Eğer etiket dosyası bulunursa, etiketleri işle
    for i, label_file in enumerate(label_files):
        if label_file:
            with open(label_file, 'r') as f:
                lines = f.readlines()

                if i == 5 or i == 6:
                    num_guns = sum(1 for line in lines if
float(line.strip().split()[1]) > 0.7 and line.strip().split()[0] ==
'0')

                    num_knives = sum(1 for line in lines if
float(line.strip().split()[1]) > 0.7 and line.strip().split()[0] ==
'1')

                    total_weapons = num_guns + num_knives

                    if total_weapons > 0:
                        label_text = labels[i].format(total_weapons)
                        cv2.putText(frame, label_text, (10, y_offset),
cv2.FONT_HERSHEY_SIMPLEX, 1, colors[i], 2)
                        y_offset += 40

                    else:
                        num_objects = sum(1 for line in lines if
float(line.strip().split()[1]) > 0.7)
                        if num_objects > 0:
                            label_text = labels[i].format(num_objects)
                            cv2.putText(frame, label_text, (10, y_offset),
cv2.FONT_HERSHEY_SIMPLEX, 1, colors[i], 2)
                            y_offset += 40

    # İşlenmiş kareyi göster
    cv2.imshow(frame)

    # Her şey bittiğinde, videoyu ve OpenCV penceresini serbest bırak
    cap.release()
    cv2.destroyAllWindows()

```

3. BULGULAR VE SONUÇ

Bu proje çalışması kapsamında, gerçek zamanlı nesne tespiti için YOLOv5 kullanılarak şüpheli nesne, insan tespiti ve kalabalık analizi başarıyla gerçekleştirilmiştir. Çalışmada, 20'den fazla veri seti kullanılarak eğitim yapılmıştır. Elde edilen en iyi ağırlıklar, projeye uygun 7 veri setinden seçilmiş ve Google Colab ortamında Tesla T4 GPU üzerinde kodlanarak işlenmiştir. Her bir veri setinin eğitim süreci yaklaşık 5 ila 8 saat arasında değişmiştir. Şüpheli nesne tespiti için silah-bıçak, çanta, paket ve sırt çantası veri setleri eğitilmiş ve kodlanmıştır. Şüpheli insan ve kalabalık analizi için ise kavga durumları, insan davranışları (agresif-panik) ve giyim tarzları (gözlük ve şapka) ile ilgili veri setleri kullanılmıştır. Ek olarak, görüntüdeki araç ve insan sayısı gibi bilgiler veri setlerinden elde edilerek kalabalık ortamlarda şüpheli durumların tespitine katkıda bulunulmuştur. Veri setlerinin genişletilmesi veya YOLO sürümünün güncellenmesi ile başarı oranlarında daha da artış sağlanması mümkündür. Bu çalışma, güvenlik ve gözetim alanlarında önemli bir potansiyele sahip olup, şüpheli durumların önceden tespit edilmesine ve önleyici tedbirlerin alınmasına yardımcı olabilir.

KAYNAKLAR

- [1] https://tr.wikipedia.org/wiki/Derin_%C3%B6%C4%9Frenme
- [2] <https://www.oracle.com/tr/artificial-intelligence/machine-learning/what-is-deep-learning/>
- [3] <https://kadirguzel.medium.com/geri-yay%C4%B1%C4%B1ml%C4%B1-%C3%A7ok-katmanl%C4%B1-yapay-sinir-a%C4%9Flar%C4%B1-2-6a47b4f3a6c>
- [4] https://tr.wikipedia.org/wiki/G%C3%B6r%C3%BCnt%C3%BC_i%C5%9Fleme
- [5] <https://medium.com/software-development-turkey/derin-%C3%B6%C4%9Frenme-hakk%C4%B1nda-neredeyse-her-%C5%9Fey-1-91bb8ddfd0>
- [6] <https://www.socar.com.tr/blog/3/goruntu-isleme-nedir>
- [7] https://en.wikipedia.org/wiki/Artificial_intelligence
- [8] https://en.wikipedia.org/wiki/Machine_learning
- [9] https://en.wikipedia.org/wiki/Deep_learning
- [10] https://en.wikipedia.org/wiki/Natural_language_processing
- [11] https://en.wikipedia.org/wiki/Computer_vision
- [12] https://en.wikipedia.org/wiki/Artificial_neural_network
- [13] https://en.wikipedia.org/wiki/Convolutional_neural_network
- [14] <https://en.wikipedia.org/wiki/Backpropagation>
- [15] https://en.wikipedia.org/wiki/Big_data
- [16] <https://docs.ultralytics.com/>
- [17] <http://weegee.vision.ucmerced.edu/datasets/landuse.html>
- [18] https://cepa.stanford.edu/sites/default/files/SEDA_documentation_v20b.pdf
- [19] https://cepa.stanford.edu/sites/default/files/SEDA_documentation_v20b.pdf
- [20] Lin, T. Y. et al. (2014). Microsoft COCO: Common Objects in Context.
- [21] Everingham, M. et al. (2010). The PASCAL Visual Object Classes (VOC) Challenge.
- [22] Redmon, J. & Farhadi, A. (2018). YOLOv3: An Incremental Improvement.
- [23] Simonyan, K. & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition.
- [24] Wang, C. et al. (2020). YOLO-LITE: A Real-Time Object Detection Algorithm Optimized for Non-GPU Computers.
- [25] Bochkovskiy, A. et al. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection.
- [26] <https://universe.roboflow.com/detection-of-accidents-and-aggressive-behavior/aggressive-behavior-dataset>
- [27] <https://universe.roboflow.com/detect-dzid1/1234-ubx7r>
- [28] <https://universe.roboflow.com/n-wto9b/bump-cap>
- [29] <https://universe.roboflow.com/glasses-8c2pr/glasses-detector-tnd9k>
- [30] <https://universe.roboflow.com/dl-project-vrsbb/anomaly-detection>

- [31] <https://universe.roboflow.com/spiderweb-labs/packages-on-conveyor>
- [32] <https://universe.roboflow.com/violence-detection-rvh0k/knife-detection-obh5j>
- [33] <https://colab.research.google.com/github/roboflow-ai/yolov5-custom-training-tutorial/blob/main/yolov5-custom-training.ipynb#scrollTo=eaFNxLJbq4J>