



MARMARA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ



DERİN ÖĞRENME YÖNTEMLERİ İLE ŞÜPHELİ DAVRANIŞ TESPİTİ

DUYGU ÇALIŞKAN

YÜKSEK LİSANS TEZİ

Bilgisayar Mühendisliği Anabilim Dalı
Bilgisayar Mühendisliği Yüksek Lisans Programı

DANIŞMAN

Doç. Dr. Önder Demir

İSTANBUL, 2022

MARMARA ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

Marmara Üniversitesi Fen Bilimleri Enstitüsü **Yüksek Lisans Öğrencisi Duygu ÇALIŞKAN**'nın “**Derin Öğrenme Yöntemleri ile Şüpheli Davranış Tespiti**” başlıklı tez çalışması, **Şubat 2022** tarihinde savunulmuş ve jüri üyeleri tarafından başarılı bulunmuştur.

Jüri Üyeleri

Doç. Dr. Önder DEMİR (Danışman)

Marmara Üniversitesi (İMZA).....

Doç. Dr. Buket DOĞAN (Üye)

Marmara Üniversitesi (İMZA).....

Dr. Öğr. Üyesi Erbil AKBAY (Üye)

Haliç Üniversitesi (İMZA).....

ONAY

Marmara Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun.....tarih ve sayılı kararı ile **Duygu ÇALIŞKAN**'nın **Bilgisayar Mühendisliği Anabilim Dalı Bilgisayar Mühendisliği Programında Yüksek Lisans** derecesi alması onanmıştır.

Fen Bilimleri Enstitüsü Müdürü

Prof. Dr. Bülent EKİCİ

ÖNSÖZ

Tez çalışmam süresince desteklerini, rehberliklerini, zamanlarını, yardımlarını ve sabırlarını esirgemeyen değerli hocalarım Doç. Dr. Önder Demir ve Doç. Dr. Buket Doğan'a

Bana olan inançları ve koşulsuz sevgileri ile hayatımın her anında yanımda olan çok kıymetli Annem Semire Çalışkan ve Babam Bedir Çalışkan'a,

Destekleri, saygı ve sevgileri ile her an yanımda olan kardeşim Fatma Çalışkan ve Ali Çalışkan'a,

Eğitim hayatım boyunca her zaman yanımda olan kıymetli Eniştem Süleyman İşlek, Teyzem Emine İşlek ve Kuzenim Yağmur İşlek'e

Yardımlarını esirgemeyen ve veri setime destek olan değerli arkadaşlarım Rahime Özden, Merve Özbey ve Özcan Yılmaz'a,

Ve tüm katılımcılara,

Teşekkürü bir borç bilirim.

Şubat, 2022

Duygu ÇALIŞKAN

İÇİNDEKİLER

ÖNSÖZ	II
İÇİNDEKİLER	III
ÖZET	V
ABSTRACT	VI
SEMBOLLER	VII
KISALTMALAR	VIII
ŞEKİL LİSTESİ	IX
TABLO LİSTESİ	XI
1. GİRİŞ	1
1.1. Yapay Zekâ	2
1.1.1 Yapay Sinir Ağları	3
1.1.2 Derin Öğrenme	4
1.1.2.1 Derin Öğrenme Mimarileri	6
1.1.2.2 Evrişimli Sinir Ağı	6
1.1.2.3 Evrişimli Sinir Ağı Algoritmaları	9
1.2 Şüpheli Davranış	12
1.3 Amaç ve Önem	13
1.4 Literatür Araştırması	15
1.4.2 Görüntü İşleme İle İlgili Çalışmalar	15
1.4.3 Şüpheli Davranış, Poz Tanımlama ile İlgili Çalışmalar	16
2 METARYAL VE YÖNTEMLER	19
2.1 Kullanılan Veri Seti	20
2.1.2 Veri Çoğullama	21
2.1.3 Veri Temizleme	22
2.1.4 Veri Etiketleme	22
2.1.5 Veri Setinin Normalize Edilmesi	24
2.1.6 Test ve Eğitim Sınıflarının Belirlenmesi	25

2.2	Kullanılan Yöntemler	27
2.2.2	Colabratory ve Darknet	27
2.2.2.3	Darknet	27
2.2.2.4	Darknet 'in Yüklenmesi ve Konfigürasyonu	27
2.2.2.5	Colabratory	33
2.2.2.6	Colabratoy'nin Yüklenmesi ve Konfigürasyonu	34
2.2.1	YOLOv4 (You Only Look Once)	38
3	BULGULAR VE TARTIŞMA	40
3.1	Kullanılan Metrik Yöntemler	40
3.1.2	Kesinlik (Precision), Duyarlılık (Recall) , F1 skor ve mAP	40
4	SONUÇ	44
	REFERANSLAR	46

ÖZET

DERİN ÖĞRENME YÖNTEMLERİ İLE ŞÜPHELİ DAVRANIŞ TESPİTİ

Hızla gelişen teknoloji ile birlikte askeri, güvenlik ve bilişim sektöründe bilgi güvenliği sorunları meydana gelmiş bu sorunların çözümü için donanımsal ve yazılımsal yöntemler geliştirilmiştir. Görüntü işleme teknikleri yardımıyla bu güvenlik sorununa görüntüden şüpheli davranış tespiti ile gerçek zamanlı bir çözüm getirilmeye çalışılmaktadır. Görüntüden nesne ve pozisyon tanımlama, davranış belirleme ve derin öğrenme üzerine yapılmış birçok çalışma mevcuttur.

Derin öğrenme, insanın düşünce yapısını veri setlerindeki örüntüler ile deneyimleyerek bilgisayara öğretme, yapay zekâ kavramının içinde barındırdığı makine öğrenmesi konusunun özel bir halidir. Makine öğrenme algoritmaları, belirlenmiş bir denkleme dayanmaksızın doğrudan bilgi verilerinden hesaplama yöntemlerini kullanarak öğrenir ve modellenirler. Derin öğrenme, yüz tanıma, plaka tanıma, nesne algılama, insansız hava araçlarında nesne tespiti, hareket algılama, otonom sürüş teknolojileri ile şerit tespiti gibi birçok gelişmiş sürücü yardım teknolojisi gibi problemleri çözmek için kullanılır.

YOLO(You Only Look Once) algoritması, konvolüsyonel sinir ağlarını (CNN) kullanarak nesne tespiti yapan bir derin öğrenme algoritmasıdır. YOLO, konvolüsyonel sinir ağları ve benzerleri algoritmalara göre daha yüksek performansa sahip bir algoritma olduğundan; çalışma kapsamında gerçek zamanlı tespiti en yakın performans elde edilmek istendiğinden, bu çalışmada YOLO algoritması ile çalışılmıştır.

Gerçekleştirilen tez çalışması 3 adımdan oluşmaktadır. Birinci adımda 1116 etiketli görsel ile Marmara Üniversitesi Kriminal Davranış/Nesne Veri Seti(MÜKDN) oluşturulmuş, ikinci adımda derin öğrenme için kullanılan evrimsel sinir ağı YOLOv4 modeli tasarımı ve konfigürasyonu yapılmıştır. Son olarak üçüncü adımda sistem eğitilmiştir. Evrimsel sinir ağı ağırlıkları farklı sistemlerde kullanılmak üzere elde edilmiş ve proaktif model gerçekleştirilmiştir.

ABSTRACT

SUSPICIOUS BEHAVIOR DETECTION WITH DEEP LEARNING METHODS

With the rapidly developing technology, information security problem have occurred in the military, security and information sector, and hardware and software methods have been developed to solve these problems. With the help of image processing techniques, a real-time solution is tried to be brought to this security question by detecting suspicious behavior from the image. There are many studies on object and position identification, behavior determination and deep learning from the image.

Machine learning algorithms learn and model using computational methods directly from information data without relying on a defined equation. It is used to solve problems such as deep learning, face recognition, license plate recognition, object detection, object detection in unmanned aerial vehicles, motion detection, autonomous driving technologies and many advanced driver assistance technologies such as lane detection. YOLO (You Only Look Once) algorithm is a deep learning algorithm that performs detects objects using convolutional neural networks (CNN). YOLO is an algorithm that can respond faster in performance than convolutional neural networks and their derivatives. For this reason, since it is desired to exhibit the closest performance to real-time detection with in the scope of the study, the YOLO algorithm will be used in this study.

The thesis work carried out consists of 3 steps. In the first step, the Marmara University Criminal Behavior/Object Data Set (MÜKDN) was created with 1116 labeled images, in the second step, the YOLOv4 model of the convolutional neural network used for deep learning was designed and configured. Finally, in the third step, the system is trained. Convolutional neural network weights were obtained for use in different systems and a proactive model was implemented.

February, 2022

Duygu ÇALIŞKAN

SEMBOLLER

X : Etiketli verinin orta noktasının X koordinatı.

Y : Etiketli verinin orta noktasının Y koordinatı.

W : Etiketli verinin genişliği

H : Etiketli verinin yüksekliği

dW : Görüntü genişliği

dH : Görüntü yüksekliği

w_{ij} : Nöron ağırlık değeri

μ : Ortalama

λ : Sabit Katsayı

Σ : Toplam

1^{obj} : Nesne bulunma durum

1^{nonobj} :Nesne bulunmama durumu

P : Olasılık

KISALTMALAR

Cnn : Convolutional Neural Networks

Esa : Evrişimsel Sinir Ağları

Iou : Intersection Over Union

Fp : False Positive (Yanlış Pozitif)

Fn : False Negative (Yanlış Negatif)

Map : Ortalama Hassasiyet (Mean Average Precision)

Mükdn: Marmara Üniversitesi Kriminal Davranış/Nesne Veri Seti

Ssd : Single Shot Detector

Şdh :Şüpheli Davranış Hareketi

Tn : True Negative(Doğru Negatif)

Tp : True Positive (Doğru Pozitif)

ŞEKİL LİSTESİ

Şekil 1.1 Nöron Biyolojik Modeli[11].....	3
Şekil 1.2 Nöron Matematiksel Modeli[11]	4
Şekil 1.3 YSA Yapısı[11]	5
Şekil 1.4 Yapay Zekâ Tarihsel Gelişim Süreci[13]	6
Şekil 1.5 Klasik Bir CNN Mimarisi[3]	7
Şekil 1.6 Single Shot Multi Box Detector (SSD)[20]	10
Şekil 1.7 You Only Look Once(YOLO)[21]	11
Şekil 1.8 VaakEye Hırsızlık Algoritma Çıktısı[26]	13
Şekil 2.1 Sistem Mimarisi.....	20
Şekil 2.2 Office, Homeoffice’ lerden Elde Edilen Veri Setinden Örnekler	21
Şekil 2.3 Veri İsimlendirme İşlemi	22
Şekil 2.4 Veri Etiketleme İşlemi	23
Şekil 2.5 Etiketli Veri Örneği	23
Şekil 2.6 Verinin Normalize Edilmesi ve Etiketlenmesi.....	25
Şekil 2.7 Test ve Eğitim için *.txt, *.data ve names Dosyaların Hazırlanması	25
Şekil 2.8 Test ve Eğitim İçin Dosya Yollarının Sisteme Tanıtılması	26
Şekil 2.9 testing.txt ve training.txt dosyaları	26
Şekil 2.10 Git’in Yüklenmesi.....	28
Şekil 2.11 Darknetin Klonlanması	28
Şekil 2.12 Darknet Dosya Dizininin Oluşturulması.....	29
Şekil 2.13 Makefile Konfigürasyonu	30
Şekil 2.14 yolov4.conv.137’nin indirilmesi	31
Şekil 2.15 yolov4.weights Dosyasının İndirilmesi Ve İlgili Dizine Eklenmesi.....	31
Şekil 2.16 supheli_davranis_yolov4.cfg Konfigürasyonu	32
Şekil 2.17 supheli_davranis_yolov4.cfg Konfigürasyonu	33
Şekil 2.18 Google Drive Hesabı Açılması	34

Şekil 2.19 Colabratory Yüklmesi	34
Şekil 2.20 supheli _davranis _model.ipynb Dosyasının Oluşturulması	35
Şekil 2.21 Google Colab Çalıştırma Ortamı	35
Şekil 2.22 Weights Değerlerinin Hesaplanması.....	36
Şekil 2.23 Weights Değerlerinin Hesaplanması.....	37
Şekil 2.24 Şüpheli Davranış İzinsiz Görüntü Alınması Tespit Edilmiş Görüntüler.....	37
Şekil 2.25 Şüpheli Davranış İzinsiz Görüntü Alınması Tespit Edilememiş Görüntüler.....	38
Şekil 2.26 YOLOv4 Mimarisi.....	39
Şekil 3.1 Şüpheli Davranış Eğitim Grafiği	43

TABLO LİSTESİ

Tablo 3.1 Karmaşıklık Matrisi.....	41
Tablo 3.2 YOLOv4’ün Metrik Hesaplama Değerleri.....	42
Tablo 3.3 YOLO’ nun Tek Sınıf İle Öğretildiğinde Başarısı	43



1. GİRİŞ

İnsan davranışının ve çevresiyle etkileşiminin otomatik olarak anlaşılması, çeşitli alanlardaki potansiyel uygulamaları nedeniyle günümüzde ilgi çekici bir araştırma alanı olmuştur. Bu araştırma alanları insan davranışını çok yönlü (duygular, ilişkisel tutumlar, eylemler, yüz ifadeleri vb.) modellemeye çalışır.

Gerek uluslararası, gerek ulusal veya ticari özel sektör girişimlerinde bilgi insanlık tarihi kadar köklü bir geçmişe dayanan ve insanlığın var olduğu sürece de önemini korumaya; vazgeçilmez bir güç olmaya devam edecek, çok önemli bir olgu ve bir teknolojidir.

Yüz yıllardır bilgi sızıntılarının neden olduğu olgular genelde eylemler, davranışlar gerçekleştikten ve istenmeyen sonuçlarla karşılaşıldıktan sonra tespit edilebilmektedir. 21. yüzyılda teknolojinin de hayatımızın her alanına girmesi ile bu bilgi sızıntılarında önceki dönemlere göre kıyaslandığında teknoloji ile; yani akıllı cihazlar aracılığı ile gerçekleştirilmekte ve her geçen gün artış eğiliminde olduğu görünmektedir[1-2].

Yapay Zekâ, en genel ifade ile insana özgü bilişsel, öğreneme, muhakeme etme özelliklerinin yüz yıllardır makinelere aktarma çalışmaları olarak ifade edilebilecek geniş bir çalışma alanıdır.

Derin öğrenme, insanın düşünce yapısını veri setlerindeki örüntüler ile deneyimleyerek bilgisayara öğreten, yapay zekâ kavramını içinde barındırdığı makine öğrenmesi konusunun birçok multidisipliner alan ile çalışılabilir özel bir halidir[3].

Yüz ifadelerimiz, hareket pozisyonumuz, nesneler suç eğilimi hakkında beklenenden daha fazlasını ifşa edebilir. Bir kişinin davranışları hakkında bilgi sahibi olabilir ve insanların yüzlerine bakarak duygularını çıkarılabilir; ve insan davranışının spesifik hali şüpheli eğilimleri hakkında önemli çıkarımlar elde edilebilir.

Derin öğrenme modellerindeki çeşitli son gelişmeler, görüntüleri kullanarak anlamsal örüntü tanıma performansını büyük ölçüde artırdı. Bir bireyin duygusal durumu ve diğer belirli karakter özellikleri veya pozisyon özellikleri gibi çeşitli durum tahminleri, yüz görüntülerinden tahmin edilebilir. Bu motivasyonla, bu çalışmada, çeşitli derin öğrenme mimarilerinin öğrenme yeteneklerini kullanarak, yüz ve nesne görüntülerinden suç eğilimi veya (suç tahmini/tespiti) çıkarımı yapılması hedeflenmiştir. Çalışma, suçluların, tam olarak görüntü hırsızlığı yaparken yakalanırsa, bilinmeyen bir kişi tarafından

herhangi bir suç eğilimini belirlemek için kullanılabilecek bir dizi yüz/nesne özelliği olduğunu ortaya koyacaktır. Geliştirilecek bu uygulamalardan doğru ve güvenilir sonuçlar üretilmesi ile proaktif bir yöntem elde edilmesi beklenmektedir.

1.1. Yapay Zekâ

Yapay zekâ, antik roma uygarlığına kadar dayanan insan gibi düşünebilme, insan bilincini oluşturmaya hedefleyen, muhakeme edebilme, ön görüde bulunma gibi kompleks işlemlerin modellenerek makinelere aktarılması olarak karşımıza birçok farklı problemin çözümünde kullanılan önemli bir bilim alanıdır.

Yapay Zekâ, en genel ifade ile insana özgü bilişsel, öğrenme, muhakeme etme özelliklerinin yüz yıllardır makinelere aktarma çalışmaları olarak ifade edebilecek geniş bir çalışma alanıdır[4].

Ramon Lull 1302 yılında yayımlanan The Ultimate General Art adlı kitabında yapay zekâdan; kavramların kombinasyonlarından oluşan yeni bir bilgi türünden ilk kez bahsetmiştir[5].

Akabinde 2. Dünya savaşında Allen Turingin "Makineler Düşünebilir mi? " sorusu ve Turing testi ile yapay zekâ fikri ortaya atıldı[6] .

1956 yılında ise Mc Carthy ve arkadaşları tarafından gerçekleştirilen konferansta Ramon Lull' un bahsettiği yeni bilgi türünün adı yapay zekâ olarak isimlendirilmiştir. Mc Carthy, yapay zekâ isminin babası olarak tarihte yerini aldı [7] .

Yapay Zekâ gelişimi dikkate alındığında elbette ki karşımıza elektriksel ve donanımsal çalışmaların sınırlamaları çıkmaktadır. Bu donanımsal ilerlemeler ve algoritmalarındaki gelişmelerle araştırmacıların dönemsel duraksamaları tarih sürecinde gözlemlense de. 1970'ler ve 1980' lerden sonraki donanımsal gelişmeler ile birlikte farklı yeni modeller geliştirilmiştir ve makine öğrenmesi ile farklı bilgi işlem teknikleri ortaya çıkmıştır. Bu tekniklerden en önemlisi de insan sinir sistemi modelini örnek alan yapay sinir ağıdır.

Bilgisayar donanımlarındaki gelişmeler insan sinir ağını modellenenebilmesini mümkün kılacak düzeyde işlemleri tamamlayabilen yeni yapay zekâ modelleri tasarlanmıştır. Bu çalışma keşfine derin öğrenme modeli denilmektedir. Bu model görüntü işleme, ses işleme, genom yapıları, öngörü sistemleri vb. hayal gücünüze kalmış sistemlere katkıda bulunmuştur.

1.1.1 Yapay Sinir Ağları

1943 yılında ilk yapay sinir ağı modeli nörolog Warren McCulloch ve matematikçi Walter Pitts tarafından tanımlanmıştır. Biyolojik bir nöron hücresi; dentrit, hücre gövdesi, aksonlar ve sinapslardan oluşmaktadır diye tanımlanmıştır [8].

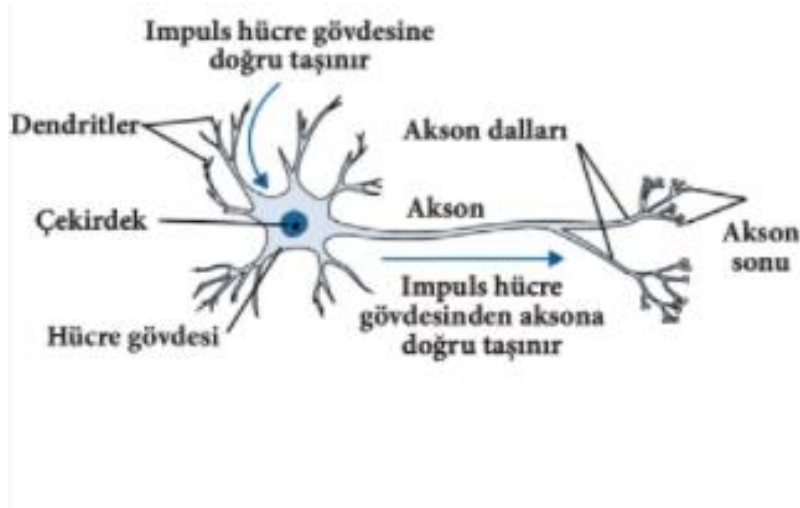
Biyoloji hücresinden esinlendiğini belirttiğimiz sinir ağıımız, 1957’de Rosenblatt yapay sinir ağlarının temel yapı taşı olan algılayıcı (perceptron) tanımını yapmıştır [9].

Şekil 1’de gösterilen McCulloch-Pitts modeline dayanan nöron m tane giriş parametresi(x_j) almaktadır ve alınan her giriş parametresi için bir yönlendirilmiş ağırlık parametresine(w_j) bağlanır. Sistemde giriş ve ağırlıklar doğrusal bir şekilde toplanır ve aktivasyon fonksiyonuna (ϕ) gönderilir. Aktivasyon fonksiyonunun çıkışı nöronun çıkışını(y_k) üretir. Bu işlem Denklem 1.1’ deki gibi formülize edilebilir [8].

$$y_k = \phi(sk) = \phi(\sum w_k j x_j m_j = 0) \quad (1.1)$$

İstenen çıktının üretilebilmesi için sistemin ağırlıkları dikkatle seçilerek nöronun eğitimi gerçekleştirilir. Birden fazla nöronun kullanılmasıyla oluşturulan sitemlere yapay sinir ağı denilmektedir[10]

Tez çalışmasında kullanılan yapay sinir sisteminde bu yapılar; işlemci elemanları, toplama fonksiyonları, transfer fonksiyonları, yapay nöron çıkışları ve ağırlıkları ifade etmektedir.

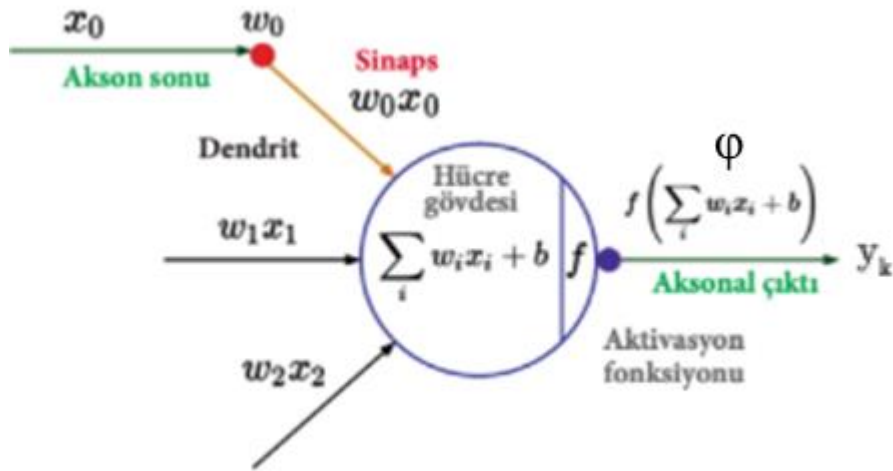


Şekil 1.1 Nöron Biyolojik Modeli[11]

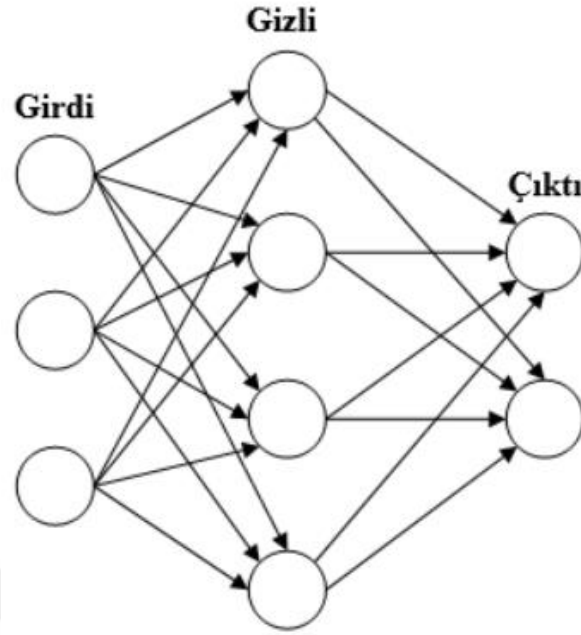
1.1.2 Derin Öğrenme

Derin öğrenme, yapay sinir ağları algoritmalarını kullanarak çok katmanlı mimarilerde çok boyutlu veriler ile çalışma imkânı sağlayan, makine öğrenmesi alanının bir alt dalıdır. Derin öğrenme; yüz tanıma, plaka tanıma, nesne algılama, hareket algılama, şerit tespiti, yaya algılama, otomatik park etme ve gelişmiş araç sürüş yardımcısı, pozisyon tanımlama, duygu algılama, doğal dil işleme, ilaç keşfi gibi birçok karmaşık problemleri çözmek için kullanılmıştır ve bu çalışmalarda ciddi bir başarı oranı elde edilmiştir. Resim tanıma ve görüntü işleme için özellikle uygun bir alandır[3].

Şekil 1.1’deki biyolojik bir sinir hücresi referans alınarak Şekil 1.3’teki basit bir yapay sinir ağı modeli gösterilmiş ve bu yapay sinir ağının matematiksel modelinde Şekil 1.2’de gösterilmiştir.



Şekil 1.2 Nöron Matematiksel Modeli[11]



Şekil 1.3 YSA Yapısı[11]

Yapay sinir ağı mimarisi bir sinir hücresi referans alınarak incelendiğinde giriş katmanı, gizli katman veya katmanlar ve çıkış katmanından oluşmaktadır.

- **Giriş Katmanı;** yapay sinir ağı sistemine dışardan verilen verileri temsil etmektedir. Sisteme giriş verisi olarak giren girdi kadar hücre bulunmaktadır. Sisteme verdiğimiz girdilere herhangi bir işlem uygulamadan doğrudan gizli katmana iletildiği katmandır.

- **Gizli Katmanlar;** giriş katmanından sisteme alınan girdinin işlendiği katmandır. Tasarlanan ysa modeline göre çeşitli fonksiyonlara (örneğin sigmoid vb.) tabi tutulduğu katmandır. Gizli katmandaki hücre sayısı ve katman sayısı girdi ve çıktı verisinden bağımsız olarak spesifik olarak tasarlanabilir.

- **Çıkış Katmanı;** giriş verisi ve gizli katmandan gelen veriye uygun olarak sistemin çıkışı olarak verinin dış dünyaya gönderildiği katmandır. Her bir çıkış hücresinin bir adet çıktısı oluşmaktadır. Her bir hücre bir önceki katmandaki bütün hücelere bağlı ve ilişkilidir.

Derin öğrenme metotları danışmanlı öğrenme ve danışmansız öğrenme olarak iki ana gruba ayrılmaktadır [12].

Danışmansız öğrenmede veri setinde veri ile alakalı herhangi bir etiket mevcut değildir. Danışmalı da ise bunun aksine veri setinde gerekli etiketlemeler yapılarak veriler sisteme etkin bir şekilde verilerek öğrenilmesi sağlanmaktadır. Yapılan çalışmamızda veri setimiz öncelikle belirlediğimiz sınıflara uygun şekilde etiketlenip danışmalı öğrenme kullanılarak eğitim ve test işlemleri gerçekleştirilmiştir.

1.1.2.1 Derin Öğrenme Mimarileri

Derin öğrenme, sisteme verilen bir veri seti ile sonuçları tahmin eden ve birden fazla katmandan oluşan makine öğrenmesi yönteminin özelleşmiş halidir. Derin öğrenme metotları danışmalı ve danışmansız öğrenme olarak iki gruba ayrılmaktadır. Derin öğrenme, makine öğrenmesinin; makine öğrenmesi ise yapay zekânın alt dalını oluşturmaktadır. Şekil 1.4 'de yapay zekâdan derin öğrenmeye geline sürecin kronolojisi görülmektedir.



Şekil 1.4 Yapay Zekâ Tarihsel Gelişim Süreci[13]

1.1.2.2 Evrişimli Sinir Ağı

CNN, görüntü analizi için tasarlanmış derin bir sinir ağıdır. Geleneksel çok katmanlı algılayıcıların aksine, bir görüntüyü temel özelliklerine indirgemek için evrişimsel(convolution) ve havuzlama(pooling) adı verilen iki işlem kullanır. Klasik bir

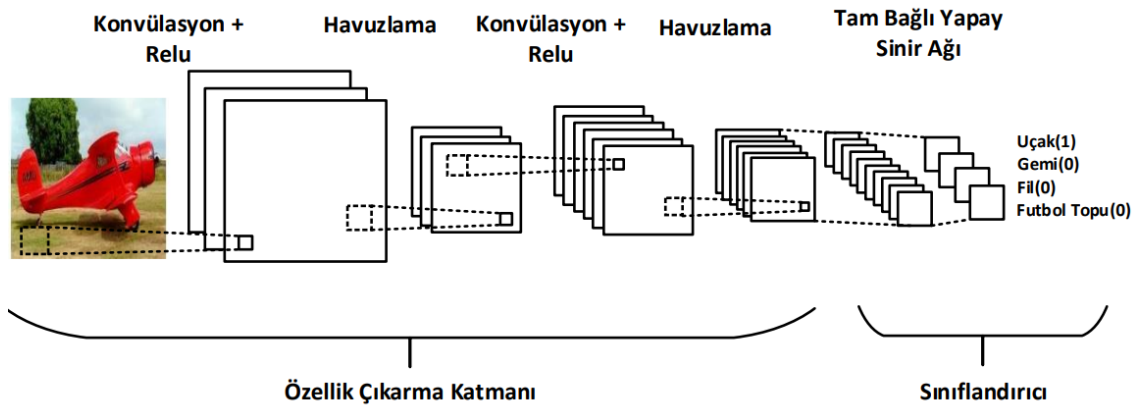
CNN yapısı Şekil 1.5' de gösterilmiştir. Evrişim katmanı konvolüsyon (convolution), aktivasyon fonksiyonu (relu) ve havuzlama (pooling) basamaklarını içermektedir.

CNN çalışma prensibi bakımından yüksek boyutlu verilerden düşük boyutlu verilerin çıkarılması prensibine dayanmaktadır. Sınıflandırma katmanında düşük boyutlu öznitelikler ile kategorilerin eşleşmesi sağlanır ve tam bağlı yapay sinir ağı mimarisine sahiptir[14].

Evrişimsel Sinir Ağları (Convolutional Neural Network) öğrenebilir ağırlık değerlerine sahip sıradan yapay sinir ağlarıdır. İnsan göreme modelini referans alarak, bilgisayar ile görü olarak özellikle nesne tespiti için kullanılmaktadır.

Çalışma yapısı incelendiğinde sinir ağında bulunan her bir nöron verileri girdi olarak alır, verileri işler ve doğrusal olmayan bir şekilde hesaplanan sonucu bir sonraki nörona aktarır. Ağda bulunan her bir nöron, ağı başında verilen ham görüntü piksellerini işleyerek ağı sonunda bir sınıfa ait skor değerini hesaplamaktadır. Hesaplanan değer en son tamamen bağlı (fully connected) katmanda işlenerek görüntünün hangi sınıfa ait olduğu bilgisini veren tahmin değeri üretilir.

1998 yılında Yann LeCun ve arkadaşları tarafından yayınlanmış ve ilk başarılı sonucu veren çok katmanlı yapay sinir ağı modelini Şekil 1.5' de görmek mümkündür[3].



Şekil 1.5 Klasik Bir CNN Mimarisi[3]

Konvolüsyonel sinir ağları biyoloji, matematik ve bilgisayar bilimlerinin ortak bir birleşimidir. Görüntü sınıflandırması, insan göreme sisteminde özelliklerin algılanması

ile merkezi sinir sistemi içerisinde gerçekleşmektedir. CNN' de görüntü sınıflandırması, bir giriş görüntüsünün alınması ve bir sınıfın çıktısı veya görüntüyü özellik çıkararak en iyi tanımlama olasılığıdır.

CNN Katmanları

Convolutional Layer (Evrişim Katmanı);

CNN algoritmalarında görüntüyü ele alan ilk katmandır. Görüntüler, belirli piksellerden oluşan matrislerdir. Görüntü özelliğine göre boyut ve derinlik hakkında bir takım bilgiler barındırır. Evrişim katmanında da orijinal görsel boyutlarından daha küçük bir filtre görselin üzerinde gezerek belirli özellikleri saptamada kullanılmaktadır. CNN algoritmalarında öğrenilen parametreler bu filtrelerdeki değerlerdir. Model sürekli olarak öğrenilen değerleri günceller ve özellikleri saptar.

Non-Linearity Katman;

Bu katman aktivasyon fonksiyonlarından birini kullandığından dolayı aktivasyon katmanı (Activation Layer) olarak da adlandırılır.

Sigmoid ve tahn gibi doğrusal olmayan fonksiyonların yanı sıra sinir ağı eğitiminin hızı konusunda en iyi sonucu veren Rectifier(ReLU) fonksiyonu bu katmanda sıklıkla kullanılır.

Pooling Layer:

Pooling "Convolutinal Neural Network yani evrişimsel sinir ağlarının en temel kavramlarından birisi olarak karşımıza çıkar. Bir görselin kendisinden daha küçük bir filtre yardımıyla boyutunun küçültülmesi işlemi olarak tanımlanabilir.

Pooling uygulamamızın nedeni bir görselin boyutunun azalması sebebiyle derin öğrenme sürecinde gerçekleştirilecek diğer tüm işlemlerin sürelerinin kısaltılmasını sağlamak ve eğitilen sinir ağının overfit (fazla öğrenme) gibi istenmeyen bir duruma düşmesinin istenmemesidir.

Flattening Layer

Bu katmanda Fully Connected Layer' a giriş verisi olarak verilecek veriler hazırlanmaktadır. Yapay sinir ağları genellikle giriş verilerini tek boyutlu bir diziden

aldığından Convolutinal ve Pooling katmanından gelen matrislerin tek boyutlu çevrilmiş hali bu katmanda oluşturulmaktadır.

Fully-Connected Layer

Bu katman evrişimsel sinir ağının son katmanıdır. Verileri Flattening işleminden alır ve yapay sinir ağı yoluyla öğrenme işlemini gerçekleştirir.

1.1.2.3 Evrişimli Sinir Ağı Algoritmaları

R-CNN, Fast R-CNN, Faster R-CNN, Single Shot Multi Box Detector (SSD) ve You Only Look Once(YOLO) algoritmaları nesne tespitinde son yıllarda en fazla tercih edilen algoritmalarlardır. Bu bölümde bu algoritmalar hakkında bilgi verilecektir.

R-CNN

R-CNN, R. Girshick ve ark tarafından 2013 yılında çoklu objeye sahip görsellerde kolay bir şekilde CNN çalıştıramadığı için R-CNN mimarisi önerilmiştir. R-CNN nesne tespitinde CNN tabanlı önerilen ilk model olduğundan daha sonra geliştirilen diğer modellere referans olmuştur. Fakat bazı eksiklik ve problemleri vardır. Bu problemlerden ilki, eğitimin ayrı aşamalardan oluşması, ikincisi ise her bölge teklifinde özellikler çıkarılarak kaydedilmekte ve bu işlem ayrıca DVM algoritması içinde uygulanmakta olduğundan çok fazla depolama alanına gereksinim duymakta bu da eğitimin maliyetli olmasına neden olmaktadır. Üçüncüsü ise grafik işlemcilerin kullanılmasına rağmen eğitimdten sonra nesne tespiti için yaklaşık olarak bir dakikaya ihtiyaç duymasıdır[15].

Fast R-CNN

Fast R-CNN , Girshick ve ark. tarafından 2015 yılında R-CNN'in katmanlarındaki hesaplamaların her bölge teklifi için ayrı ayrı değil tüm resim için tek bir defa yapılması ile geliştirilerek sunulmuştur. Bu yolla eğitim süresi azalmış ve R-CNN'den daha iyi bir şekilde eğitilmesi sağlanmıştır[16-17].

Faster R-CNN

Faster R-CNN, Ren ve ark. tarafından Fast R-CNN algoritmasının bölge teklifleri üretme adımını geliştirerek sunulmuştur. Faster R-CNN algoritmasının çalışma prensibi konvolüsyon

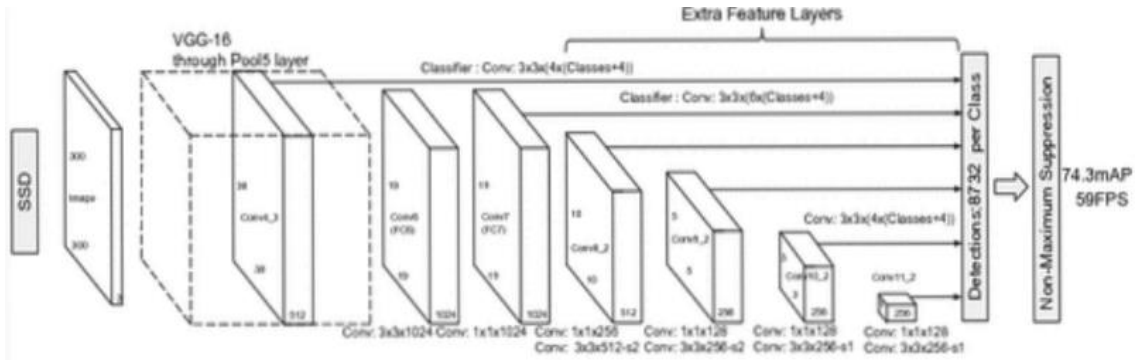
katmanlarını hem nesne hem de bölge tekliflerini üretmekte ortak olarak kullanmasına dayanmaktadır[17-18].

Faster R-CNN iki aşamada açıklayalım ilk aşama olarak Region Proposal Network, herhangi bir boyutta girdiyi alır ve obje skoruna göre dikdörtgen teklifi ortaya çıkarır bunu, evrişim katman tarafından oluşturan öznetelik haritası üzerinde küçük bir ağı kaydırarak yapar.

Fast R-CNN mimarisine sokulur ve bir sınıflandırıcı ile objenin sınıfı, regressor ile de bounding box'u tahmin edilir[18].

Single Shot Multi Box Detector (SSD)

Gerçek zamanlı sistemler için Liu ve arkadaşları tarafından 2016 yılında öne sürülen Single Shot Multi Box Detector (SSD) modeli arka plan bilgisini kullanarak nesneyi algılamaktadır. SSD gerçek zamanlı olarak nesne algılama için tasarlanmıştır. SSD nesne algılama için iki aşamadan oluşmaktadır; özellik haritalarını çıkarmak ve nesne algılamak için evrişimsel filtreleri kullanmak olarak. Doğruluk açısından teknolojinin başlangıcı olarak kabul edilen bu modelde düşük çözünürlükteki görüntülerin kullanılması ile daha hızlı R-CNN elde edilebilir. Tüm süreç saniyede 7 defa çalışmaktadır. [19-20]. Single Shot Multi Box Detector (SSD) mimarisi Şekil 1.6' da gösterilmiştir.



Şekil 1.6 Single Shot Multi Box Detector (SSD)[20]

You Only Look Once(YOLO)

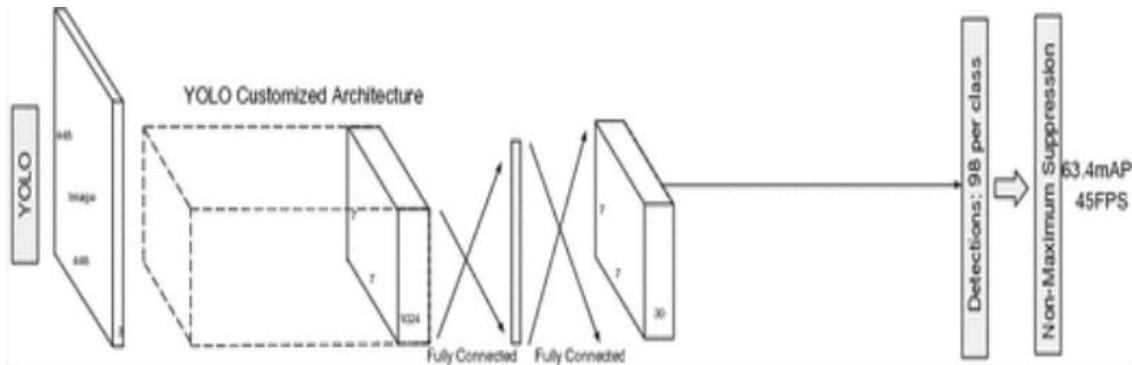
YOLO nesne algılama ve nesne sınıflandırma için tek bir CNN ağı kullanan bir derin öğrenme algoritmasıdır. “You Only Look Once”(Sadece Bir Kez Baksın) anlamına gelmektedir.

Joseph Redmon ve arkadaşları nesne algılamayı, doğrudan görüntü piksellerinden sınırlayıcı kutu koordinatlarına ve sınıf olasılıklarına kadar tek bir regresyon problemi olarak ele almışlardır. GoogLeNet modelinden esinlenmiştir. Tek bir evrişimli ağı ile aynı anda birden çok sınırlayıcı kutuyu ve bu kutular için sınıf olasılıklarını tahmin eder. Her sınıf için görüntüde tekrar tekrar işlem yapmak yerine algoritmanın adından da anlaşılacağı üzere görüntüye bir defa bakılır ve tüm sınıflar için olasılıkları ve sınırlayıcı kutu koordinatları oluşturulur[21].

Ağda 24 evrişim katmanı ve ardından 2 tam bağlantılı katman vardır. GoogLeNet tarafından kullanılan başlangıç modülleri yerine, Lin ve diğerlerine benzer şekilde 1×1 azaltma katmanlarını ve ardından 3×3 evrişim katmanlarını kullanmışlar, tam görüntü üzerinde eğitim gerçekleştirdiklerinden algılama performansını doğrudan optimize etmişlerdir. Bu sayede nesneyi ve nesnenin koordinat bilgisini öğrenmek oldukça hızlı bir şekilde gerçekleşmektedir.

YOLO birleşik mimarisi sayesinde ileri derecede hızlıdır. YOLO 'nun ilk versiyonu saniyede 45 kare (fps) işleyebilecek yetenektir. Fast YOLO olan versiyonu ise saniyede 155 kare işleyebilmektedir. Bu sayesinde gerçek zamanlı nesne tespitinde çok başarılı bir sistemdir[21].

YOLO mimarisi Şekil 1.7'de gösterilmiştir.



Şekil 1.7 You Only Look Once(YOLO)[21]

YOLO1

YOLOv1, tek aşamalı bir nesne algılama modelidir. Nesne algılama, uzamsal olarak ayrılmış sınırlayıcı kutulara ve ilişkili sınıf olasılıklarına bir regresyon problemi olarak

çerçevesi. Tek bir sinir ağı, tek bir değerlendirmede doğrudan tam görüntülerden sınırlayıcı kutuları ve sınıf olasılıklarını tahmin eder. Algılama hattının tamamı tek bir ağ olduğundan, uçtan uca doğrudan algılama performansına göre optimize edilebilir.

Ağ, her bir sınırlayıcı kutuyu tahmin etmek için görüntünün tamamındaki özellikleri kullanır. Aynı zamanda bir görüntü için tüm sınıflardaki tüm sınırlayıcı kutuları aynı anda tahmin eder. Yolo mimarisinin ilk versiyondur[22].

YOLO2

YOLOv2 veya YOLO9000, tek aşamalı gerçek zamanlı nesne algılama modelidir. Darknet-19'un omurga olarak kullanılması, toplu normalleştirme, yüksek çözünürlüklü bir sınıflandırıcı kullanımı ve sınırlayıcı kutuları tahmin etmek için bağlantı kutularının kullanımı ve daha fazlası dahil olmak üzere YOLOv1'i çeşitli şekillerde geliştirir[23].

YOLO3

YOLOv3, çeşitli iyileştirmelerle YOLOv2'yi temel alan gerçek zamanlı, tek aşamalı bir nesne algılama modelidir. İyileştirmeler arasında, artık bağlantıları kullanan yeni bir omurga ağı olan Darknet-53'ün veya yazarın sözleriyle bu yeni moda artık ağ öğelerinin yanı sıra sınırlayıcı kutu tahmin adımı bazı iyileştirmeler ve üç özelliklerin çıkarılacağı farklı ölçekler. YOLOv3 Darknet53 omurgasına sahip[24].

YOLO4

YOLOv4, Alexey Bochkovskiy tarafından 2020'de yayınlandı. YOLOv4'ün Evrimsel Sinir Ağı (CNN) doğruluğunu iyileştirdiği söylenen çok sayıda özellik var. YOLOv4'ün YOLOv3'ten farkı omurgasıdır. YOLOv4 CSPDarknet53 omurgasına sahiptir. YOLOv4 kafaları YOLOv3 ile aynıdır. Başlıklar tahmin bölümüdür ve iki türü vardır. Biri Yoğun Tahmin (Tek aşamalı dedektör) diğeri ise Seyrek Tahmin (İki aşamalı dedektör)[25].

1.2 Şüpheli Davranış

Uluslararası, ulusal ve şirketler özelinde bilgi çok büyük bir güç olup mevcut otoritelerce gizliliğine önem verilmektedir. Bilgi sızıntılarında dünyada bazen büyük bir yatırım hamlesine, bazen de ulusal veya uluslararası savaşa sürüklenme boyutuna varacak dünyada örnekleri bulunan bir olgudur. Bilgi sızıntılarındaki donanım ve yazılımsal alınan tedbirlere rağmen insan faktörü bu önlemlere rağmen sızıntıya yol açabilecek

faaliyetlerde bulunmaktadır. Derin öğrenme algoritması ile cep telefonu veya küçük kamera aracılığı ile görüntü alırken bu olayı kayıt edip sisteme şüpheli hareket diye rapor edecek bir sistem üzerinde çalışılmıştır.

Davranışları inceleyen ve bunu güvenlik sistemlerine entegre eden market hırsızlığını raporlayabilecek bir uygulama VaakEye güvenlik sistemleri tarafından çalışılmış ve hızlı hareket etme, kamerayı tarama ve nesneyi üzerindeki çanta, cep vb. yerlere sıkıştırma üzerinde derin algoritma ile çalışmışlardır. Ve bu hareketler tespit edildiğinde müşteriyi olası şüpheli olarak kayıt etmektedir. VaakEye 'in bu algoritmasının görselleri Şekil 1.8' de gösterilmiştir[26].



Şekil 1.8 VaakEye Hırsızlık Algoritma Çıktısı[26]

1.3 Amaç ve Önem

Bu çalışmada askeri, güvenlik ve bilişim sektöründeki bilgi güvenliği sorunlarının çözümündeki teknolojilerde kullanılabilecek real-time proaktif sistemlerin, videolarda ve gerçek zamanlı çalışan sistemlerde şüpheli hareket/davranış, şüpheli yüz ifadesi ve nesne yüzeylerini tespit edebilen bir model gerçekleştirilmesi ile bilgi güvenliğini tehdit edecek insan davranışlarının tespiti amaçlanmıştır. Örneğin görüntü almanın yasak olduğu bir ortamda cep telefonu ile bilgisayarın ekran görüntüsünün almaya çalışılması tespit edilebilmesi amaçlanmıştır.

Derin öğrenme yöntemlerinin görüntü işleme ile multidisipliner bir şekilde çalışmaya uygun olması göz önünde bulundurularak bu tez çalışmasında görüntüden şüpheli davranış tespiti modellemesi için görüntü işleme ile temel olarak derin öğrenme algoritmaları kullanılmıştır.

İnsan davranışının ve çevresiyle etkileşiminin otomatik olarak anlaşılması, çeşitli alanlardaki potansiyel uygulamaları nedeniyle günümüzde ilgi çekici bir araştırma alanı olmuştur. Bu araştırma alanları insan davranışını çok yönlü (duygular, ilişkisel tutumlar, eylemler vb.) altında modellemeye çalışır.

Yapılan Katkıları:

- Gerçekleştirilen çalışmada; Kriminal davranış tespiti alanında yeni ve büyük veri seti olan Marmara Üniversitesi Kriminal Davranış/Nesne (MÜKDN) oluşturulmuştur ve elde edilen veri seti üzerinde YOLO algoritması kullanılarak, kamera açısından bağımsız şüpheli bir şekilde görüntü alma eylemi tespiti gerçekleştirilmiştir. Derin öğrenme yöntemlerinin şüpheli davranış-nesne tespit etme alanında kullanılabilirliği kanıtlanmıştır. Şüpheli davranış-nesne yüzeyi tespit etme alanında elde edilen yüksek hız ve yüksek doğruluk değerleri, gerçek zamanlı çalışan güvenlik sistemlerinde kullanım imkânı sağlanması amaçlanmaktadır.
- Yüzlerimiz, hareket pozisyonumuz, nesneler bize suç eğilimi hakkında beklediğimizden daha fazlasını ifşa etmektedir. Bir kişinin davranışları hakkında bilgi sahibi olabilir ve insanların yüzlerine bakarak duyguları analiz edilebilmektedir. Derin öğrenme modellerindeki çeşitli son gelişmeler, görüntüleri kullanarak anlamsal örüntü tanıma performansını büyük ölçüde artırdı. Bir bireyin duygusal durumu ve diğer belirli karakter özellikleri veya pozisyon özellikleri gibi çeşitli durum tahminleri, yüz görüntülerinden tahmin edilebilmektedir. Bu motivasyonla, bu çalışmada, çeşitli derin öğrenme mimarilerinin öğrenme yeteneklerini kullanarak, yüz ve nesne görüntülerinden suç eğilimi veya (suç tahmini/tespiti) çıkarımı yapmaya çalıştık. Çalışmamızda, suçluların, tam olarak görüntü hırsızlığı yaparken yakalanırsa, bilinmeyen bir kişi tarafından herhangi bir suç eğilimini belirlemek için kullanılabilecek bir dizi

yüz/nesne özelliği olduğunu ortaya koymaktadır. Geliştirilen bu uygulamalarda doğru ve güvenilir sonuçlar üretilerek proaktif bir yöntem elde edilmiş ve bu sonuçların güvenlik sistemlerine doğru bir şekilde entegrasi ile suçun tezimizde savunduğumuz bilgi sızıntısının tespit edilebileceği hatta önüne geçilebileceği yaklaşımı sunulmuştur.

- Derin öğrenme yöntemlerinin askeri, bilişim ve güvenlik sektörlerinde şüpheli davranış ve hareketin tespit etme alanında kullanılabilirliği kanıtlanmıştır.

1.4 Literatür Araştırması

Derin öğrenme günümüzde oldukça popüler bir çalışma alanıdır. Görüntü işleme algoritmaları ve derin öğrenme yöntemleri birlikte kullanılarak birçok alanda uygulamalar geliştirilmiştir. Bu başlık altında, literatürde bulunan görüntü işleme algoritmaları ile derin öğrenme yöntemlerinin birlikte kullanıldığı ve şüpheli davranış, poz tanımlama ile ilgili önemli çalışmalara değinilmektedir.

1.4.2 Görüntü İşleme İle İlgili Çalışmalar

Tıbbi görüntülerden hastalık teşhisinde, ilaç üretimine, kalite kontrol uygulamalarına, insansız hava araçları kullanımından, kalabalık analizine, dil işlemeye kadar daha bir çok konuda derin öğrenme ile görüntü işleme alanında çalışmalar yapılmıştır.

Halgurg ve ark. 2021' de, BT ve X-ray gibi göğüs radyografilerini kullanarak koronavirüs vakalarını tespit etmede X-ışınları ve BT tarama görüntüleri üzerine basit bir evrişim sinir ağı (CNN) ve modifiye edilmiş önceden eğitilmiş AlexNet modeli uygulanmıştır. Deneylerin sonucu, kullanılan modellerin önceden eğitilmiş ağ üzerinden %98'e kadar doğruluk ve modifiye CNN kullanılarak %94,1 doğruluk sağlayabildiğini göstermektedir[27].

Mehdi ve ark. 2020 yılında hastalık teşhisi ve rehabilitasyon süreci için, optimize edilmiş özellik çıkarma ve ardından doğrusal diskriminant analizi sınıflandırması (TLRN-LDA) modelini önermiş ResNet50 Derin öğrenme modeli ile transfer öğrenme konsepti kullanılarak geliştirilmiştir. 31 sınıfın zorlu standart karşılaştırmalı ImageCLEF-2012 veri kümesi üzerinde kapsamlı deneyler gerçekleştirmişlerdir. Geliştirilen yaklaşım, aynı veri kümesindeki en son teknoloji yaklaşımlarla karşılaştırıldığında %10'a kadar daha yüksek olan %87,91'lik gelişmiş ortalama sınıflandırma doğruluğu sağlar[28].

1.4.3 Şüpheli Davranış, Poz Tanımlama ile İlgili Çalışmalar

Jauredi ve arkadaşları, 2020 yılında İnsan eylemi tanıma için yeni bir hibrit derin öğrenme modeli ile insan eylemi tanıma alanında yeni bir yaklaşım önermişlerdir. Bu yaklaşım, video içeriğinin analizine ve çıkarma özelliklerine dayanmaktadır. Hareket özelliklerini, GMM ve KF yöntemleri kullanılarak insan hareketi takibi ile sunmaktadır. Bir diğer özellikleri, Kapılı Tekrarlı Birimli Tekrarlayan Sinir Ağları modeli kullanılarak video dizisindeki her bir karenin tüm görsel özelliklerine dayandırmalarıdır. Bu yaklaşımın temel avantajları, videonun her karesinde ve her anındaki tüm özelliklerin analizi ve çıkarılmasıdır. Bu çalışmanın deneysel sonuçları, yüksek sınıflandırma oranı elde etmek için yeni yaklaşımın güçlü performansını göstermektedir. Gerçekten de önerilen yöntem, insan etkinliği analizi gibi çeşitli uygulamalarda ve alanlarda kullanılabilir. Vücut hareketine dayalı tıbbi uygulama, el hareketlerini kullanan akıllı ara yüz ve daha fazlası. Gelecekteki çalışmalar için amaç, video sınıflandırma süresini azaltmak ve UCF Sport, UCF101 ve daha zorlu veri kümelerinin sınıflandırma oranını iyileştirmek için yaklaşımlarının değerlendirilmesi olarak sunulmuştur[29].

Ashwin ve arkadaşlar 2020 yılında, hintli öğrencilerin yüzlerini, el hareketlerini ve vücut duruşlarını kullanan e-öğrenme ve sınıf ortamları için duygusal veri tabanı oluşturarak yaptıkları çalışmada öğrencinin yüz ifadelerini tanımayı ve bunları Ekman' ın temel duygularına göre sınıflandırmayı içermektedir. Duyuşsal durumlar, öğrencilerin yüz ifadeleri, el hareketleri ve vücut duruşları olmak üzere üç bileşenin tümü kullanılarak sınıflandırılır. İfadelerin sınıflandırılması evrimsel sinir ağları (CNN) kullanılarak gerçekleştirmişlerdir. Çalışma sonucunda öğrencilerin yüz ifadeleri, el hareketleri ve vücut duruşları kullanılarak hem e-öğrenme (tek görüntü çerçevesinde tek öğrenci) hem de sınıf ortamları (tek görüntü çerçevesinde birden çok öğrenci) için çok modlu duygusal veri tabanı başarıyla oluşturulmuştur. Nötr de dahil olmak üzere 11 farklı duygusal durum için altın standart çalışması kullanılarak açıklamalar yapıldı. Anlatıcılar, Cohen' in kabul ettiği duygulanım durumlarına karşı ayırım yaparken güvenilir bir şekilde hemfikirlerdir. $\kappa=0.48$. Her öğrencinin her modalitesinde nesne lokalizasyonu gerçekleştirilir ve sınırlayıcı kutu koordinatları duygusal durumla birlikte saklanır. Duyuşsal durum sınıflandırmasının tespiti ve sınıflandırılmasında %83 ve %76 doğruluk elde etmişlerdir.

Mevcut çalışma görüntüden duygu analizi konusunda güncel bir çalışma niteliğindedir[30].

Jammalamadaka ve arkadaşları 2017 yılında, bir sorgu modalitesi olarak poz kullanarak görüntü ve video arama için yeni bir yaklaşımı başarıyla gösterdiler. Dans ve spor video verileri üzerinde poz tanımlayıcılarını elde etmek ve poz alımını gerçekleştirmek için derin pozlet yöntemi ve derin poz gömme yöntemi olmak üzere iki yol önerdiler. İlk yöntemde, poz uzayının 'poza duyarlı' derin pozletler kullanılarak ayrıklaştırılabileceğini gösterdiler. Bu derin pozlet dedektörleri, belirli bir pozda vücut parçalarının bir alt kümesini modeller. CNN' yi kullanmışlardır. Poz için bir özellik temsili oluşturmada temel yapıtaşları olarak kullanılmıştır. İkinci yöntemde, bir görüntünün daha düşük boyutlu, poza duyarlı bir Alana doğrudan nasıl eşlenebileceğine değinmişlerdir. Daha sonra, her iki yöntemi kullanarak pozlamanın, rakip poz alma yöntemleriyle eşit olduğunu ampirik olarak göstermişlerdir[31].

Dieogo ve arkadaşları 2019 yılında durağan görüntülerden insan poz tahmini için yeni bir regresyon yöntemi sundular. Yöntem parça tabanlı algılama haritalarını dolaylı olarak öğrenmek için derin bir evrişimli ağa entegre edilebilen türevlenebilir bir işlem olan soft-argmax işlemine dayanmaktadır ve bu, regresyon yöntemlerinden elde edilen son teknoloji puanlar üzerinde önemli bir gelişme sağlamıştır. Gelecekteki bir çalışma olarak, 3B poz tahmini veya pozdan insan eylemi tanımayı tamamen ayırt edilebilir bir şekilde sağlamak için yaklaşımlarına başka yöntemler de eklenebilir olduğunu belirtmişlerdir[32].

Baccouche ve arkadaşları 2011 yılında, KTH veri setini kullanarak önsel bir modelleme olmaksızın, yalnızca eğitim örneklerinden otomatik öğrenmeye dayanarak, 3D Evrişimsel ağının genişletilmesiyle mekansal-zamansal özellikleri otomatik öğrenen model geliştirdiler. İnsan eylemlerinin dizilerini sınıflandırmak için nöral tabanlı bir derin model kullandılar. İki adımlı şema olarak sunulan modelde, spatiotemporal özellikleri otomatik olarak öğrenir ve bunları tüm dizileri sınıflandırmak için kullanmışlardır[33].

İnsan eylemlerini tanımak için en popüler modern yöntemler arasında Laptev ve ark. [34] Dollar ve ark. [35] Chen ve ark. [36] Gao ve ark. Liu ve ark[37], hepsi manuel olarak tasarlanmış mekansal-zamansal ilgi noktaları etrafında hesaplanan mühendislik hareket ve doku tanımlayıcılarını kullanır.

Daş ve ark., 2019 yılındaki derin öğrenme yöntemleri ile hareketli nesne takibi çalışmalarında Google'ın açık kaynak kodlu olarak kullanıma sunduğu Tensorflow kütüphanesini kullanarak nesne takibi için Faster R-CNN modeli ele alınmışlardır. Bu kütüphane yardımıyla fotoğraflar, video görüntüler üzerinde nesne tanıma gerçekleştirmiş; kütüphanelerin güçlü ve zayıf yönlerini değerlendirmişlerdir[38].

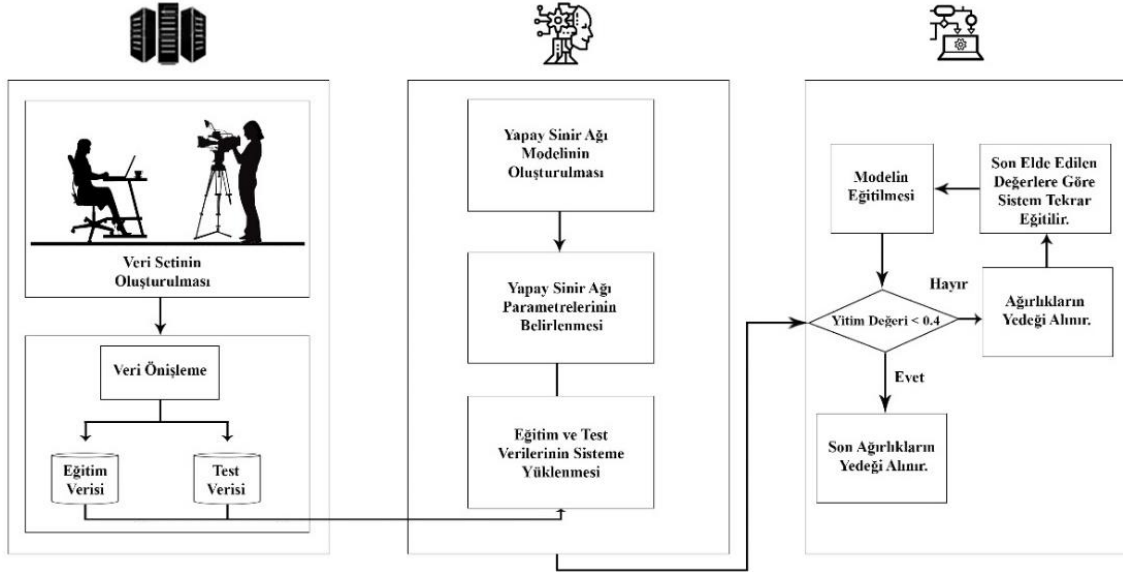
Yılmaz ve ark., 2020 yılındaki çalışmalarında, Faster CNN(Faster Region Based Convolutional Networks) ağı kullanarak geliştirdikleri çalışmada derin öğrenme teknikleriyle beraber Faster R-CNN ağı kullanılmışlardır. 502 Adet resim verisinden oluşan veri seti oluşturulmuş. Eğitilen model resim, video ve kamera üzerinden alınan anlık görüntüler ile çalıştırılmış ve yüksek başarı oranı elde edilmiştir. Modelin real-time gıda üretim tesisinde kalite kontrol sürecinde kullanılabileceği belirtilmiştir[39].

2 METARYAL VE YÖNTEMLER

Gerçekleştirilen tez çalışması Şekil 2.1'deki sistem mimarisinde görüldüğü gibi temel olarak 3 adımdan oluşmaktadır.

İlk olarak çalışmada kendi veri setimiz oluşturulmuştur. İç mekanlar da(Office, homeoffice, derslik, laboratuvar vb.) çekilen görüntü ve video kaydı gibi görsel kaynaklar belli ön işleme evresinden geçirilerek etiketli bir veri seti elde edilmiştir. Yeterli verinin elde edilebilmesi için Opencv ile veri çoğullama işlemi uygulanmıştır. Veri çoğullama işleminden sonra kendini tekrar eden veriler yapay sinir ağı modelinde overfitting 'e neden olmaması için temizlenmiştir. Ve son olarak etiketleme işlemini yapmak için python programlama (pilow kütüphanesi), makesense vb. programlar ile LabelTool program kullanılmıştır. Ve bu alanda yeni ve büyük veri seti olan Marmara Üniversitesi Kriminal Davranış/Nesne (MÜKDN) oluşturulmuştur. Veri seti üzerinde yapılan ön işleme tamamlandıktan sonra veri seti eğitim ve test için ; %70 eğitim , % 30 test olarak rasgele ikiye ayrılmıştır

İkinci adım olarak derin öğrenme için kullanılacak evrişimsel sinir ağının tasarlanması ve konfigrasyonu yapılmıştır. Veri seti üzerinde YOLO algoritması kullanılarak, kamera açısından bağımsız şüpheli görüntü alma eylemi, şüpheli davranış yüzeyleri tespiti gerçekleştirilmiştir. Son adımda ise sistemin transfer learning metodu ile eğitim işlemi gerçekleştirilmiş olup, diğer sistemlerde kullanılmak üzere evrişimsel sinir ağının ağırlıkları hesaplanmıştır.



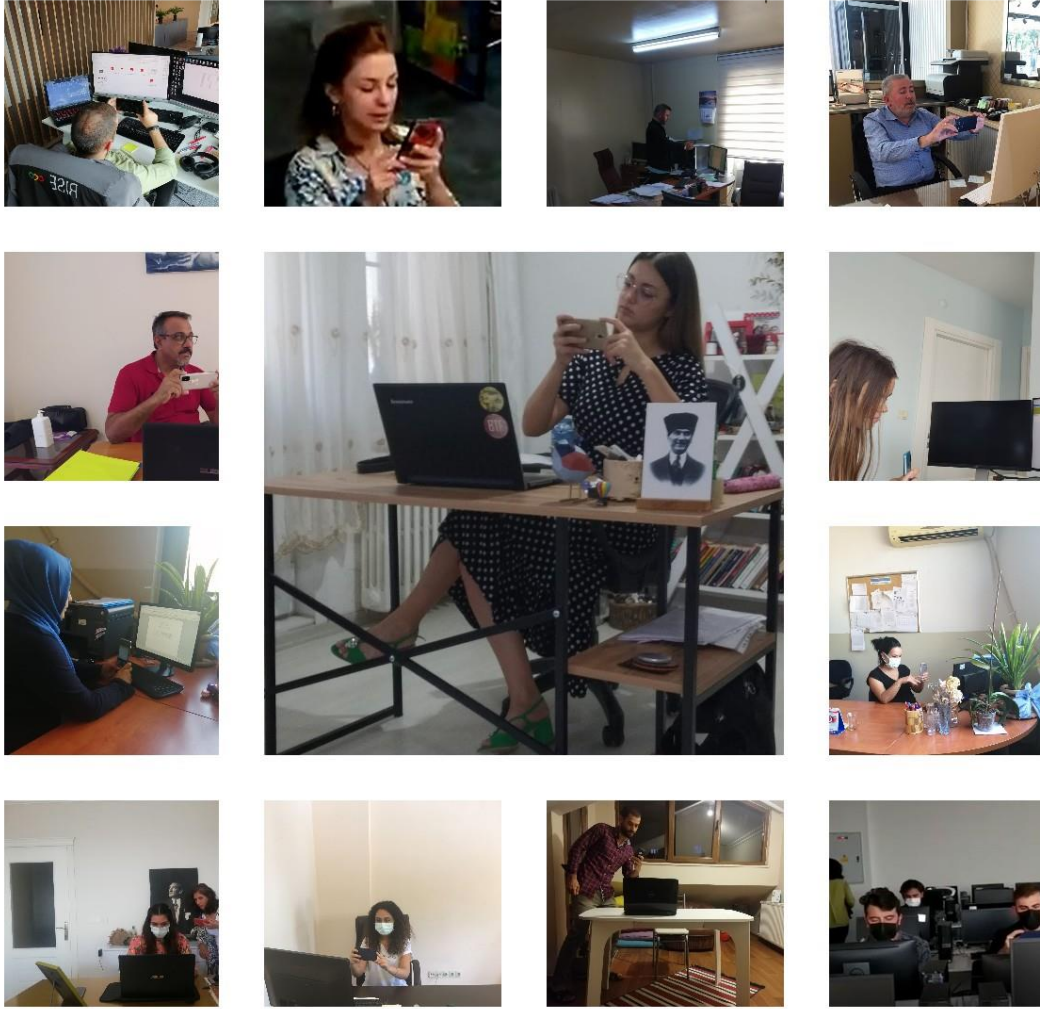
Şekil 2.1 Sistem Mimarisi

2.1 Kullanılan Veri Seti

Yapılan çalışmada araştırmacıların video çekimlerini ile elde edilmiş MÜKDN veri seti kullanılmıştır. Veri hazırlama ve önleme evresinde, iç mekanlarda çekilen görüntü ve video gibi görsel kaynaklar belirli önleme işlemlerinden geçirilerek derin öğrenmede kullanılmak üzere etiketli MÜKDN veri seti haline getirilmiştir.

Görüntüler Eskişehir, Hatay ve İstanbul'da, bulunan office, homeoffice, Marmara Üniversitesi Teknoloji Fakültesi Bilgisayar laboratuvarlarında katılımcıların gönüllü olarak verilen görüntülerinden elde edilmiştir. Şekil 2.2'de veri setindeki görüntülerden örnekler gösterilmiştir.

Görüntüler temin edilirken Samsung HMX-QF30 kamerası ve Sony HDR-PJ410 kamerası, bilimüm cep telefonu kameraları kullanılmıştır. Görüntü temini sırasında gönüllülük esasına dayalı çekimler yapıldığından herhangi bir kişinin özel hayatı ihlal edilmemiştir.



Şekil 2.2 Office, Homeoffice’ lerden Elde Edilen Veri Setinden Örnekler

2.1.2 Veri Çoğullama

Elde edilen görüntüleri sisteme uygun bir şekilde kullanmak için hem de YOLO küçük nesnelerde iyi performans göstermediğinden YOLO algoritmasına uygun formatta kullanabilmek adına görüntülerde yeniden boyutlandırma işlemi yapılmıştır. Görüntüler 1024x768 boyutuna getirilmiştir.

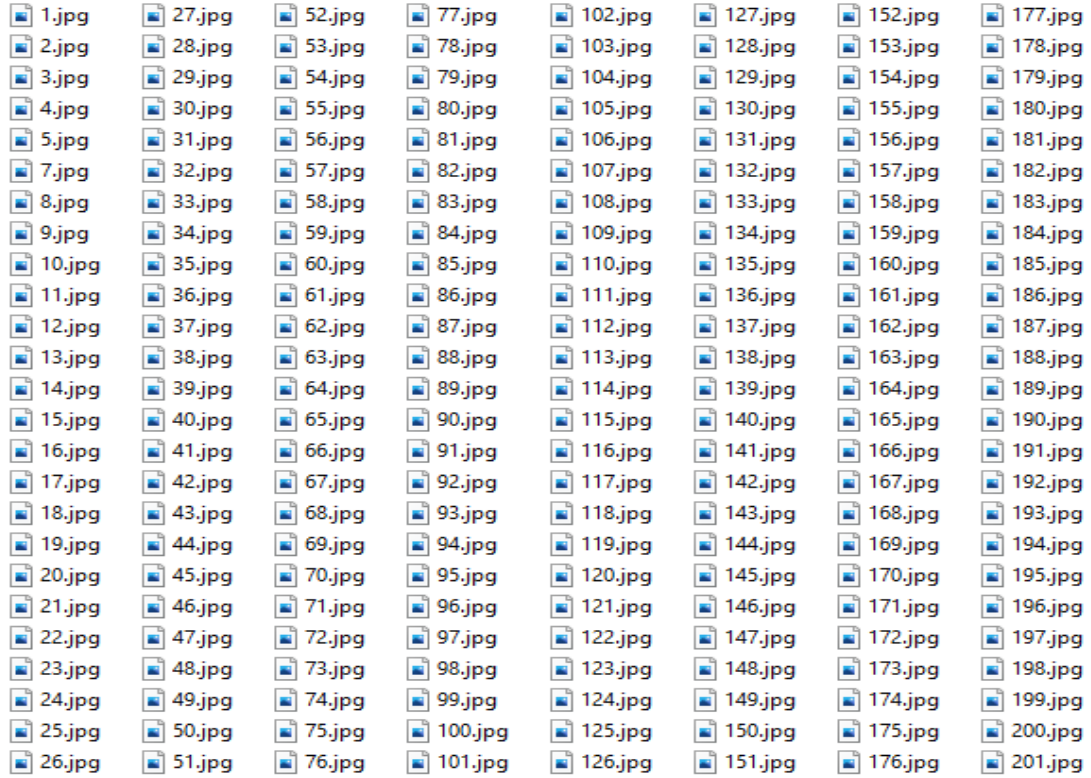
Konvolüsyonel sinir ağları ile danışmanlı öğrenme çalışmalarında veri setinde birbirini tekrar etmeyen çok fazla örneğe gereksinim duyulmaktadır. Bunun nedeni örnek sayısı ile uygulama performansının doğru orantılı olarak artmasıdır. Uygulama performansını arttırmak için ham veri opencv veri arttırma işlemlerinden çoğullama, aynalama, beyazlatma vb. yöntemlerini kullanarkattırılmıştır.

2.1.3 Veri Temizleme

Elde edilen tüm video ve fotoğraflardan belirlenen senaryoya uygun olan görüntüler seçilip temizlenmiştir. Senaryoda karşılaşmayacağımız ve veri de overfitting' e neden olacak tekrarlar temizlenmiştir.

2.1.4 Veri Etiketleme

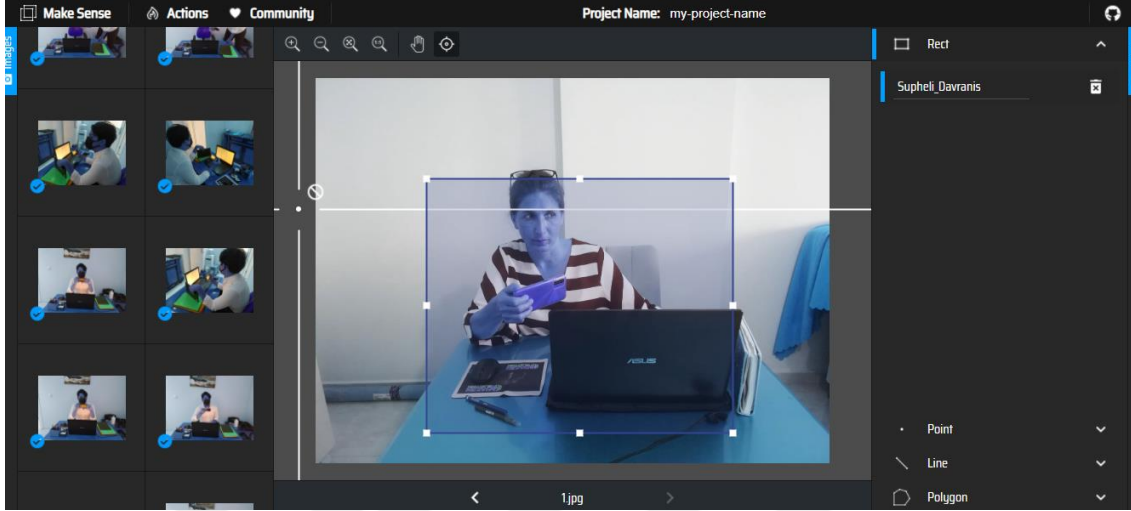
Etiketleme aşaması için veri setimize 1 den başlayarak veri setinin sonuna kadar sıralı isimlendirme işlemi uygulanmıştır. Şekil 2.3' de yapılan isimlendirmeden bir kesit sunulmuştur.



Şekil 2.3 Veri İsimlendirme İşlemi

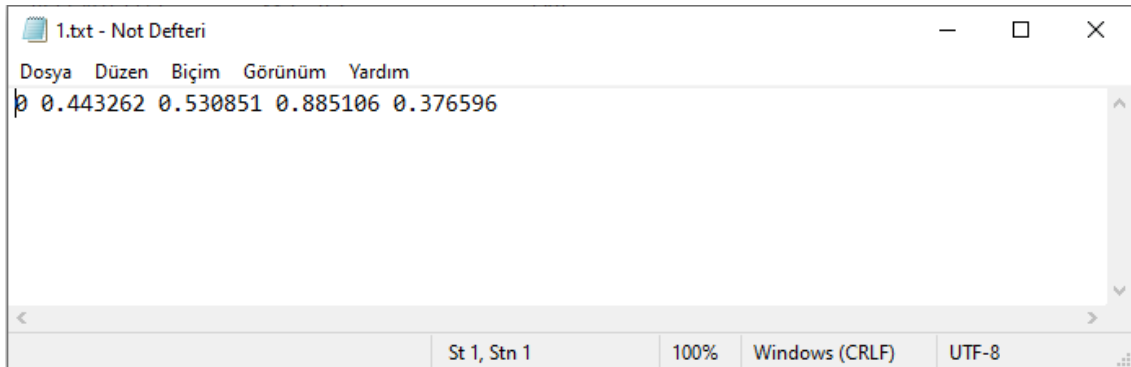
YOLO algoritmasına uygun şekilde etiketleme yapabilmek için kullanılabilecek alternatif programlar mevcuttur. Bunlara örnek olarak Supervise.ly, Hasty.ai, CVAT, Darwin, Heartex, Scalabel, Segments.ai, Make-Sense, LinkedAI, LabelImg, RectLabel, LabelBox, LabelMe ve DataTurks ve makesense.ai vb. programlar verilebilir. Bu çalışma

kapsamında makesense.ai ve labelBox ile tüm veri seti görüntüleri belirlenen senaryoya uygun şekilde Şekil 2.4’ de görüldüğü gibi etiketlenmiştir.



Şekil 2.4 Veri Etiketleme İşlemi

Her bir görüntü için tespit edilen koordinatlar ve görüntüde kaç adet nesne olduğu bilgisi görüntü adı ile aynı isimde oluşturulan bir text dosyasına yazılarak kayıt edilmektedir. Şekil 2.5’ de görüldüğü gibi her bir görüntü için etiketli .txt uzantılı dosya oluşturulmuştur.



Şekil 2.5 Etiketli Veri Örneği

2.1.5 Veri Setinin Normalize Edilmesi

Etiketlenen görüntülerin uzunluk(dH) ve genişliği(dW), şüpheli davranış hareketinin hangi koordinatta olduğu bilgisini veren (X_0, X_1, Y_0, Y_1) noktaları ve ŞDH etiket değeri görüntü verisi ile aynı isimdeki *.txt uzantılı(text) dosyası içerisine yazılmaktadır. Veri setindeki tüm örneklere normalizasyon işlemi uygulanarak etiketli MÜKDN veri seti elde edilmiştir. Bu işlem sistemde kullanılacak YOLO mimarisine uygun hale getirmek için yapılmıştır.

(X_0, X_1, Y_0, Y_1) koordinatları normalizasyon işlemi ile 0 ve 1 aralığına indirgenmiştir. Bu işlem ile görüntü içerisindeki etiketli ŞDH'nin, merkez noktası koordinatları (X, Y) , yüksekliği(H) ve genişliği(W) bilgileri elde edilmektedir. Yapılan normalizasyon işleminin formülü Denklem 2.1 'de verilmiştir.

$$X = \frac{X_1 + X_0}{2} \times \frac{1}{dW}$$

$$Y = \frac{Y_1 + Y_0}{2} \times \frac{1}{dH}$$

$$W = (X_1 - X_0) \times \frac{1}{dW}$$

$$H = (Y_1 - Y_0) \times \frac{1}{dH}$$

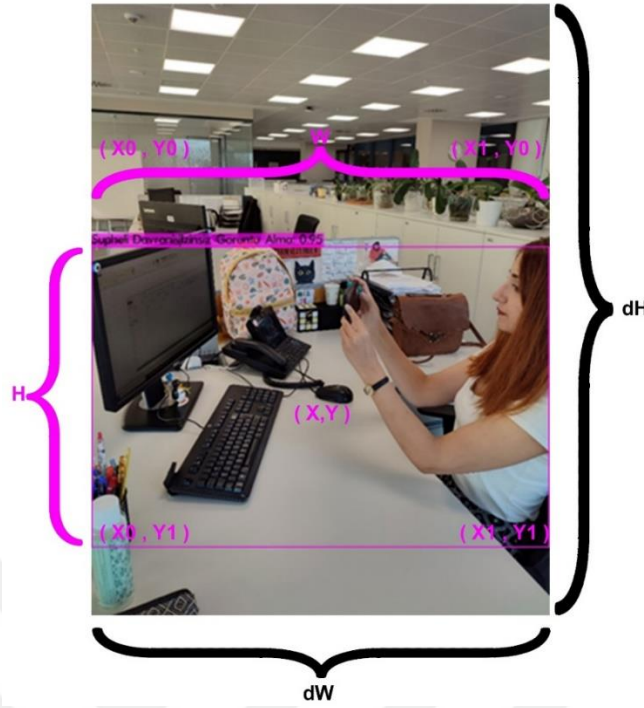
X : Etiketli verinin orta noktasının X koordinatı.

Y : Etiketli verinin orta noktasının Y koordinatı.

W : Etiketli verinin genişliği

H : Etiketli verinin yüksekliği

Normalize edilmiş ve etiketlenmiş görüntü örneği Şekil 2.6' da gösterilmiştir.



Şekil 2.6 Verinin Normalize Edilmesi ve Etiketlenmesi

2.1.6 Test ve Eğitim Sınıflarının Belirlenmesi

Ön işleme tamamlandıktan sonra etiketli MÜKDN veri setindeki örnekler, %70 eğitim ve %30 test verisi olarak ikiye ayrılmıştır.

- Veri setinin dosya dizini ve gerekli *.txt, *.data ve *.names dosyaları Şekil 2.6'daki gibi oluşturulmuştur.

Masaüstü > 28122021 > darknet > supheli_davranis_data				Ara: supheli_davranis_data	
Ad	Değiştirme tarihi	Tür	Boyut		
supheli_davranis_images	28.12.2021 19:25	Dosya klasörü			
supheli_davranis_labels	28.12.2021 19:25	Dosya klasörü			
supheli_davranis.data	20.12.2021 11:14	DATA Dosyası	1 KB		
supheli_davranis.names	19.12.2021 15:04	NAMES Dosyası	1 KB		
supheli_davranis_testing.txt	21.12.2021 20:45	Metin Belgesi	20 KB		
supheli_davranis_training.txt	21.12.2021 20:47	Metin Belgesi	41 KB		

Şekil 2.7 Test ve Eğitim için *.txt, *.data ve names Dosyaların Hazırlanması

- Hazırlanan testing.txt ve training.txt dosyalarının sistemce kullanılmasının bildirilmesi için *.data dosyasına ilgili konfigürasyon işlemleri yapılmıştır. Şekil 2.7’ de gösterilmiştir.

```

classes = 1
train = supheli_davranis_data/supheli_davranis_training.txt
valid = supheli_davranis_data/supheli_davranis_testing.txt
names = supheli_davranis_data/supheli_davranis.names
backup = backup

```

Şekil 2.8 Test ve Eğitim İçin Dosya Yollarının Sisteme Tanıtılması

- Eğitim ve testte kullanılmak üzere etiketli görüntülerin dosya adreslerini testing.txt ve training.txt dosyalarına belirlenen %70 eğitim ve %30 test oranı olmak üzere Şekil 2.8’de gösterildiği gibi kayıt edilmiştir.

```

supheli_davranis_data/supheli_davranis_images/1.jpg
supheli_davranis_data/supheli_davranis_images/2.jpg
supheli_davranis_data/supheli_davranis_images/3.jpg
supheli_davranis_data/supheli_davranis_images/4.jpg
supheli_davranis_data/supheli_davranis_images/5.jpg
supheli_davranis_data/supheli_davranis_images/6.jpg
supheli_davranis_data/supheli_davranis_images/7.jpg
supheli_davranis_data/supheli_davranis_images/8.jpg
supheli_davranis_data/supheli_davranis_images/9.jpg
supheli_davranis_data/supheli_davranis_images/10.jpg
supheli_davranis_data/supheli_davranis_images/11.jpg
supheli_davranis_data/supheli_davranis_images/12.jpg
supheli_davranis_data/supheli_davranis_images/13.jpg
supheli_davranis_data/supheli_davranis_images/14.jpg
supheli_davranis_data/supheli_davranis_images/15.jpg
supheli_davranis_data/supheli_davranis_images/16.jpg
supheli_davranis_data/supheli_davranis_images/17.jpg
supheli_davranis_data/supheli_davranis_images/18.jpg
supheli_davranis_data/supheli_davranis_images/19.jpg
supheli_davranis_data/supheli_davranis_images/20.jpg
supheli_davranis_data/supheli_davranis_images/21.jpg
supheli_davranis_data/supheli_davranis_images/22.jpg
supheli_davranis_data/supheli_davranis_images/23.jpg
supheli_davranis_data/supheli_davranis_images/24.jpg
supheli_davranis_data/supheli_davranis_images/25.jpg
supheli_davranis_data/supheli_davranis_images/26.jpg
supheli_davranis_data/supheli_davranis_images/27.jpg
supheli_davranis_data/supheli_davranis_images/28.jpg
supheli_davranis_data/supheli_davranis_images/29.jpg
supheli_davranis_data/supheli_davranis_images/30.jpg

```

```

supheli_davranis_data/supheli_davranis_images/763.jpg
supheli_davranis_data/supheli_davranis_images/764.jpg
supheli_davranis_data/supheli_davranis_images/765.jpg
supheli_davranis_data/supheli_davranis_images/766.jpg
supheli_davranis_data/supheli_davranis_images/767.jpg
supheli_davranis_data/supheli_davranis_images/768.jpg
supheli_davranis_data/supheli_davranis_images/769.jpg
supheli_davranis_data/supheli_davranis_images/770.jpg
supheli_davranis_data/supheli_davranis_images/771.jpg
supheli_davranis_data/supheli_davranis_images/772.jpg
supheli_davranis_data/supheli_davranis_images/773.jpg
supheli_davranis_data/supheli_davranis_images/774.jpg
supheli_davranis_data/supheli_davranis_images/775.jpg
supheli_davranis_data/supheli_davranis_images/776.jpg
supheli_davranis_data/supheli_davranis_images/777.jpg
supheli_davranis_data/supheli_davranis_images/778.jpg
supheli_davranis_data/supheli_davranis_images/779.jpg
supheli_davranis_data/supheli_davranis_images/780.jpg
supheli_davranis_data/supheli_davranis_images/781.jpg
supheli_davranis_data/supheli_davranis_images/782.jpg
supheli_davranis_data/supheli_davranis_images/783.jpg
supheli_davranis_data/supheli_davranis_images/784.jpg
supheli_davranis_data/supheli_davranis_images/785.jpg
supheli_davranis_data/supheli_davranis_images/786.jpg
supheli_davranis_data/supheli_davranis_images/787.jpg
supheli_davranis_data/supheli_davranis_images/788.jpg
supheli_davranis_data/supheli_davranis_images/789.jpg
supheli_davranis_data/supheli_davranis_images/790.jpg
supheli_davranis_data/supheli_davranis_images/791.jpg
supheli_davranis_data/supheli_davranis_images/792.jpg

```

Şekil 2.9 testing.txt ve training.txt dosyaları

2.2 Kullanılan Yöntemler

Gerçekleştirilen sistemde, CSPDarknet53'ün omurgasını kullanan YOLOv4 modeli kullanılmıştır.

2.2.2 Colabratory ve Darknet

Sistem mimarisini YOLOv4 modeline göre derin öğrenme yöntemleri ile eğitmek için darknet ağı ve colabratory ide 'si tercih edilmiştir. Bu bölümde darknet ve Colabratory kurulumu ve konfigürasyonları hakkında bilgi verilecektir.

2.2.2.3 Darknet

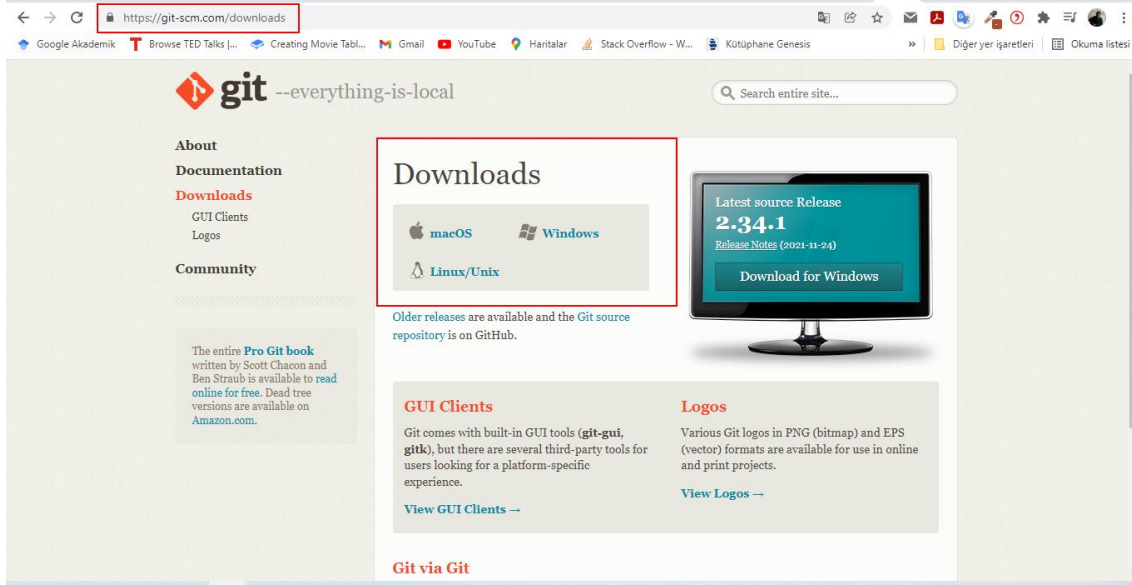
Darknet, Joseph Redmon tarafından C dili ve CUDA ile geliştirilmiş açık kaynak kodlu bir Sinir Ağı İskeletidir (Neural Network Framework)[40].

Darkneti hazır haliyle kullanabilmek için bir Linux veya Unix sisteme ihtiyaç duyulmaktadır.

Darknet'i Windows işletim sisteminde kullanabilmek için GitHub'da AlexeyAB takma adlı kullanıcının pjreddie(Joseph Redmon) 'nin kaynak kodundan geliştirdiği versiyonu kullanılabilir (AlexeyAB). Bu çalışmada AlexeyAB'nin kodu kullanılmış olup, AlexeyAB'nin paylaştığı bilgilerden de faydalanılmıştır.

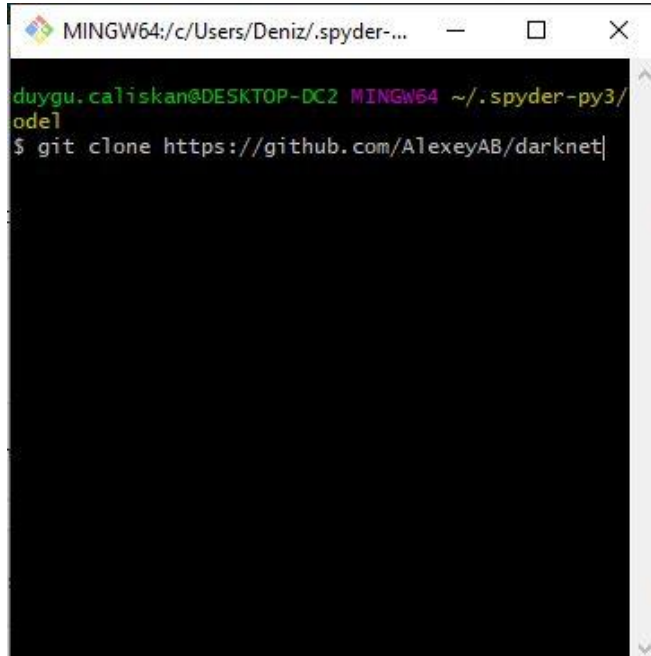
2.2.2.4 Darknet 'in Yüklenmesi ve Konfigürasyonu

- Git-scm/download adresinden sahip olunan işletim sistemine uygun git versiyonu Şekil 2.10 da görüldüğü gibi indirilmiştir.



Şekil 2.10 Git'in Yüklmesi

- Bilgisayarımızda C:\Users\Duygu\.spyder-py3\custom_yolo_model\yolov4 içerisinde Şekil 2.11'de görüldüğü gibi kurulan Git Bash'i çalıştırılarak “ \$ git clone https://github.com/AlexeyAB/Darknet ” komutunu koşturarak darknet dosyaları klonlanmıştır. Şekil 2.12'deki dosya dizini oluşturulmuştur[41].

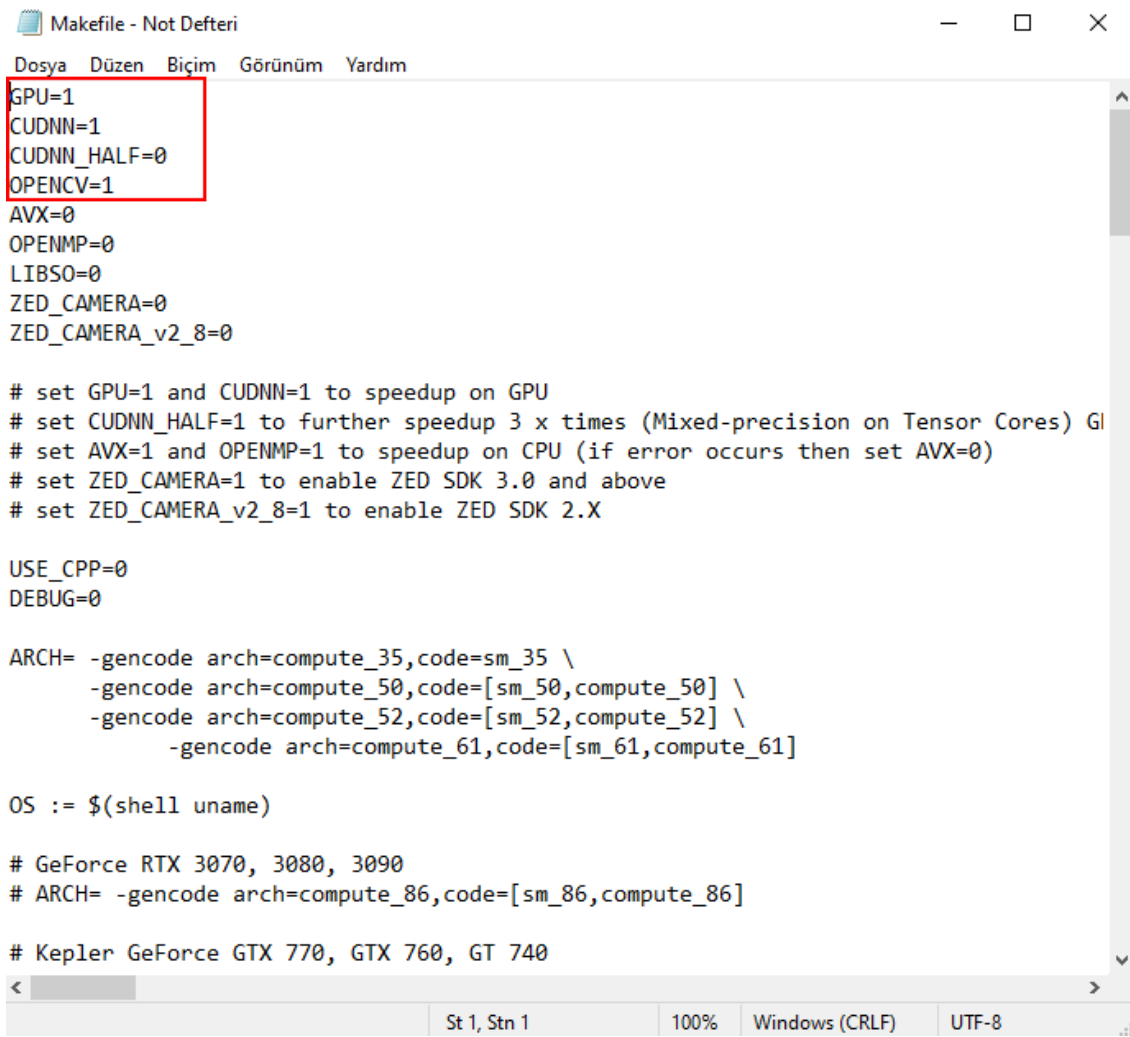


Şekil 2.11 Darknetin Klonlanması

YOLO > custom_yolo_model > yolov4 > darknet				
Ad	Değiştirme tarihi	Tür	Boyut	
.github	26.08.2021 13:44	Dosya klasörü		
3rdparty	26.08.2021 13:44	Dosya klasörü		
build	26.08.2021 13:44	Dosya klasörü		
cfg	26.08.2021 13:44	Dosya klasörü		
cmake	26.08.2021 13:44	Dosya klasörü		
data	26.08.2021 13:44	Dosya klasörü		
include	26.08.2021 13:44	Dosya klasörü		
results	26.08.2021 13:44	Dosya klasörü		
scripts	26.08.2021 13:44	Dosya klasörü		
src	26.08.2021 13:44	Dosya klasörü		
supheli_davranis_data	19.12.2021 17:21	Dosya klasörü		
.gitignore	26.08.2021 13:44	Metin Belgesi	1 KB	
.travis.yml	26.08.2021 13:44	YML Dosyası	6 KB	
build.ps1	26.08.2021 13:44	Windows PowerS...	28 KB	
CMakeLists.txt	26.08.2021 13:44	Metin Belgesi	25 KB	
darknet.py	26.08.2021 13:44	PY Dosyası	11 KB	
darknet_images.py	26.08.2021 13:44	PY Dosyası	10 KB	
darknet_video.py	26.08.2021 13:44	PY Dosyası	7 KB	
DarknetConfig.cmake.in	26.08.2021 13:44	IN Dosyası	2 KB	
image_yolov3.sh	26.08.2021 13:44	Shell Script	1 KB	
image_yolov4.sh	26.08.2021 13:44	Shell Script	1 KB	
json_mjpeg_streams.sh	26.08.2021 13:44	Shell Script	1 KB	
LICENSE	26.08.2021 13:44	Dosya	1 KB	
Makefile	26.08.2021 13:47	Dosya	7 KB	
net_cam_v3.sh	26.08.2021 13:44	Shell Script	1 KB	

Şekil 2.12 Darknet Dosya Dizinini Oluşturulması

- Dosya dizini oluşturulduktan sonra Şekil2.13’de görüldüğü üzere Makefile dosyasını notepad ile açılıp CPU, CUDNN ve OPENCV değerleri 1’e set edilmiştir.



```
GPU=1
CUDNN=1
CUDNN_HALF=0
OPENCV=1
AVX=0
OPENMP=0
LIBSO=0
ZED_CAMERA=0
ZED_CAMERA_v2_8=0

# set GPU=1 and CUDNN=1 to speedup on GPU
# set CUDNN_HALF=1 to further speedup 3 x times (Mixed-precision on Tensor Cores) G
# set AVX=1 and OPENMP=1 to speedup on CPU (if error occurs then set AVX=0)
# set ZED_CAMERA=1 to enable ZED SDK 3.0 and above
# set ZED_CAMERA_v2_8=1 to enable ZED SDK 2.X

USE_CPP=0
DEBUG=0

ARCH= -gencode arch=compute_35,code=sm_35 \
      -gencode arch=compute_50,code=[sm_50,compute_50] \
      -gencode arch=compute_52,code=[sm_52,compute_52] \
      -gencode arch=compute_61,code=[sm_61,compute_61]

OS := $(shell uname)

# GeForce RTX 3070, 3080, 3090
# ARCH= -gencode arch=compute_86,code=[sm_86,compute_86]

# Kepler GeForce GTX 770, GTX 760, GT 740
```

Şekil 2.13 Makefile Konfigürasyonu

- Modelimizi eğitirken transfer learning metodu kullanıldığından; <https://github.com/AlexeyAB/Darknet> adresinden yolov4.conv.137 adlı dosya Şekil 2.14’de gösterildiği gibi indirilmiş ve darknet klasörümüzün içerisine kayıt edilmiştir.

How to train with multi-GPU

1. Train it first on 1 GPU for like 1000 iterations: `darknet.exe detector train cfg/coco.data cfg/yolov4.cfg yolov4.conv.137`
2. Then stop and by using partially-trained model `/backup/yolov4_1000.weights` run training with multigpu (up to 4 GPUs): `darknet.exe detector train cfg/coco.data cfg/yolov4.cfg /backup/yolov4_1000.weights -gpus 0,1,2,3`

If you get a Nan, then for some datasets better to decrease learning rate, for 4 GPUs set `learning_rate = 0,00065` (i.e. `learning_rate = 0.00261 / GPUs`). In this case also increase 4x times `burn_in =` in your cfg-file. I.e. use `burn_in = 4000` instead of `1000`.

<https://groups.google.com/d/msg/darknet/NbJqonJBTSY/Te5PflpuCAAJ>

How to train (to detect your custom objects)

(to train old Yolo v2 `yolo-v2-voc.cfg`, `yolo-v2-tiny-voc.cfg`, `yolo-voc.cfg`, `yolo-voc.2.0.cfg`, ... [click by the link](#))

Training Yolo v4 (and v3):

0. For training `cfg/yolov4-custom.cfg` download the pre-trained weights-file (162 MB): `yolov4.conv.137` (Google drive mirror `yolov4.conv.137`)
1. Create file `yolo-obj.cfg` with the same content as in `yolov4-custom.cfg` (or copy `yolov4-custom.cfg` to `yolo-obj.cfg`) and:

Şekil 2.14 yolov4.conv.137'nin indirilmesi

- Şekil 2.15'de gösterildiği üzere <https://github.com/AlexeyAB/Darknet> adresinden “yolov4.weights” adlı ağırlıkların bulunduğu dosya indirilmiş ve transfer learning’ te kullanmak üzere darknet dizininin içerisine kayıt edilmiştir[41].

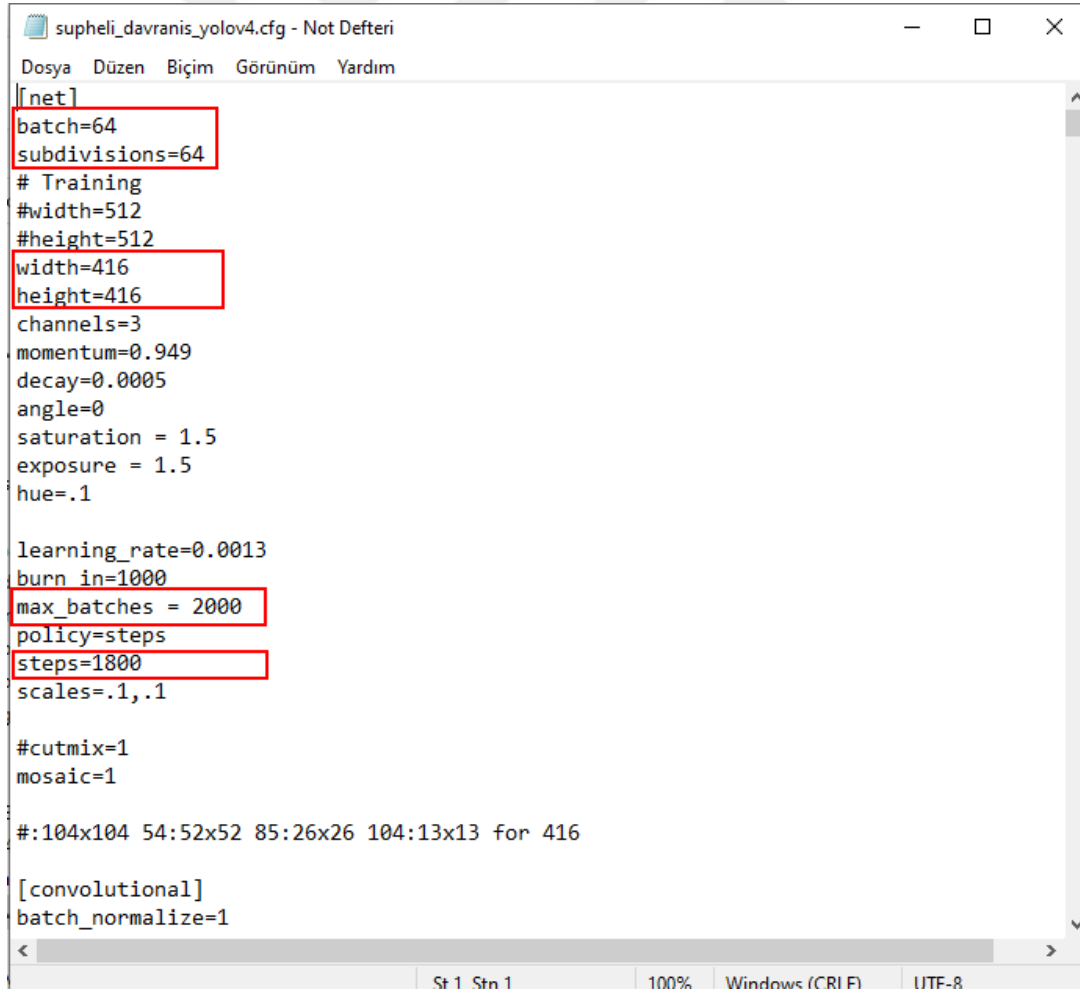
How to evaluate AP of YOLOv4 on the MS COCO evaluation server

1. Download and unzip test-dev2017 dataset from MS COCO server:
<http://images.cocodataset.org/zips/test2017.zip>
2. Download list of images for Detection tasks and replace the paths with yours:
<https://raw.githubusercontent.com/AlexeyAB/darknet/master/scripts/testdev2017.txt>
3. Download `yolov4.weights` file 245 MB: `yolov4.weights` (Google-drive mirror `yolov4.weights`)
4. Content of the file `cfg/coco.data` should be

```
classes= 80
train = <replace with your path>/trainvalno5k.txt
valid = <replace with your path>/testdev2017.txt
names = data/coco.names
backup = backup
eval=coco
```

Şekil 2.15 yolov4.weights Dosyasının İndirilmesi Ve İlgili Dizine Eklenmesi

- Şekil 2.16’da görüldüğü gibi *.cfg uzantılı dosya kopyalanıp darknet dizinine yapılandırılmıştır. Modelde kullanılan supheli_davranis_yolov4.cfg dosyası YOLOv4 modeline göre yapılandırılmıştır. Yapılandırma işleminde ilk olarak batch değeri ayarlanmıştır. Batch; her bir iterasyonda konvolüsyonel sinir ağına alınacak resim sayısını ifade etmektedir. Batch değeri varsayılan 64 olarak ayarlanmıştır. İkinci olarak subdivisions değeri 64 olarak ayarlanmıştır. Subdivisions değeri çok yüksek seçilirse eğitim süresi uzar, düşük seçilirse de kalitesiz bir eğitim işlemi yapılmış olur; bu nedenle en uygun değer olarak 64 belirlenip, her bir batchin 64 adıma bölünmesi sağlanmıştır. Üçüncü olarak max batch değeri de 2000’e ayarlanıp, step değerini de batch değerinin %80-90’ı arasında olması gerektiğinden; step olarak 1800 olarak tercih edilmiştir.



```
[net]
batch=64
subdivisions=64
# Training
#width=512
#height=512
width=416
height=416
channels=3
momentum=0.949
decay=0.0005
angle=0
saturation = 1.5
exposure = 1.5
hue=.1

learning_rate=0.0013
burn_in=1000
max_batches = 2000
policy=steps
steps=1800
scales=.1,.1

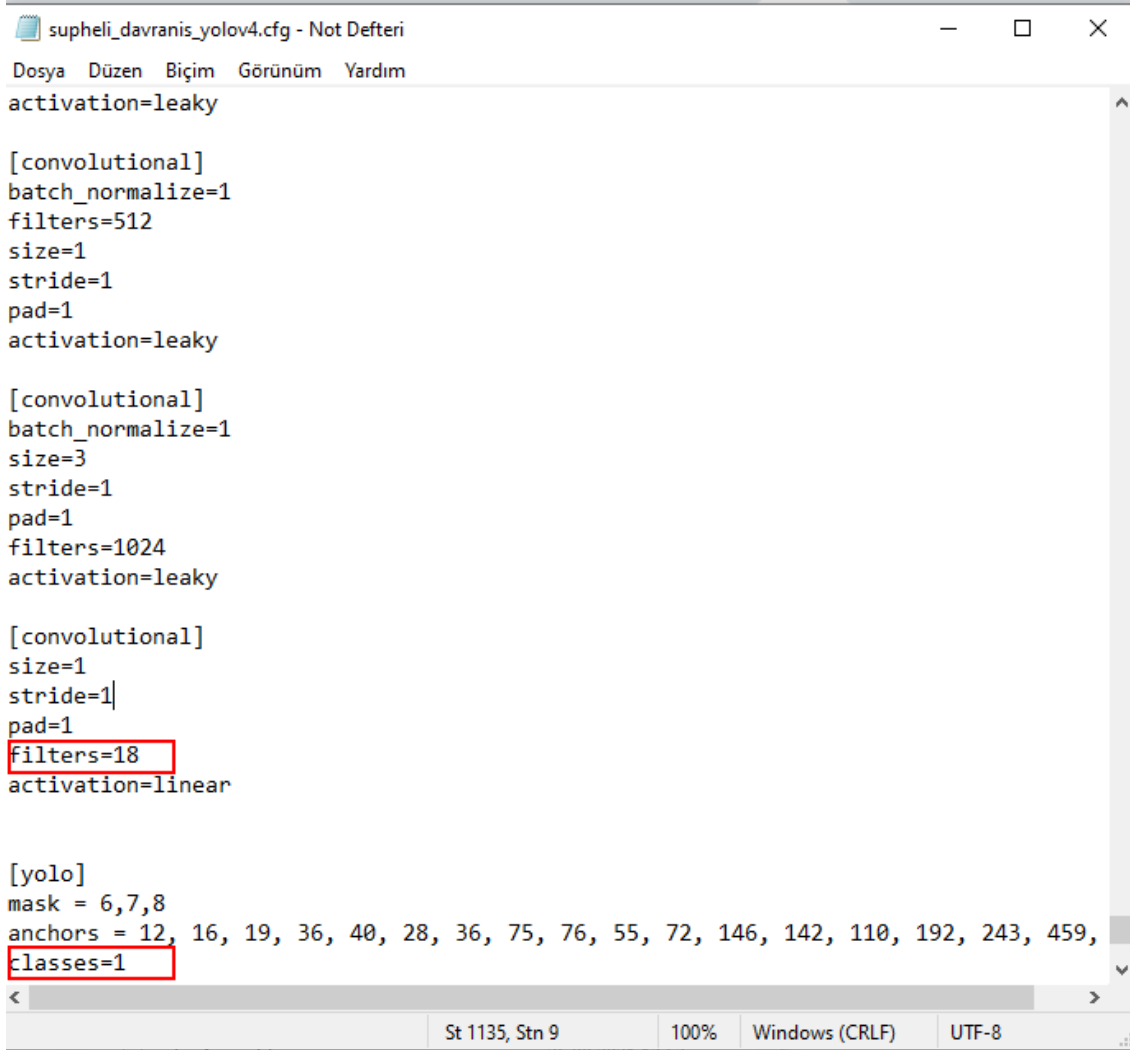
#cutmix=1
mosaic=1

#:104x104 54:52x52 85:26x26 104:13x13 for 416

[convolutional]
batch_normalize=1
```

Şekil 2.16 supheli_davranis_yolov4.cfg Konfigürasyonu

- Son olarak Şekil 2.17’de görüldüğü gibi sınıf ve filtre değerleri ayarlanmıştır. Kullanılan modelde bir adet sınıf bulunduğu için sınıfı 1 olarak ayarlayıp, filtre değerini de sınıf sayısının beş fazlasının üç katı olarak tercih edilmesi gerektiği için; filtre değeri 18 olarak ayarlanmıştır.



```
supheli_davranis_yolov4.cfg - Not Defteri
Dosya Düzen Biçim Görünüm Yardım
activation=leaky

[convolutional]
batch_normalize=1
filters=512
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=1024
activation=leaky

[convolutional]
size=1
stride=1
pad=1
filters=18
activation=linear

[yolo]
mask = 6,7,8
anchors = 12, 16, 19, 36, 40, 28, 36, 75, 76, 55, 72, 146, 142, 110, 192, 243, 459,
classes=1

St 1135, Stn 9 100% Windows (CRLF) UTF-8
```

Şekil 2.17 supheli_davranis_yolov4.cfg Konfigürasyonu

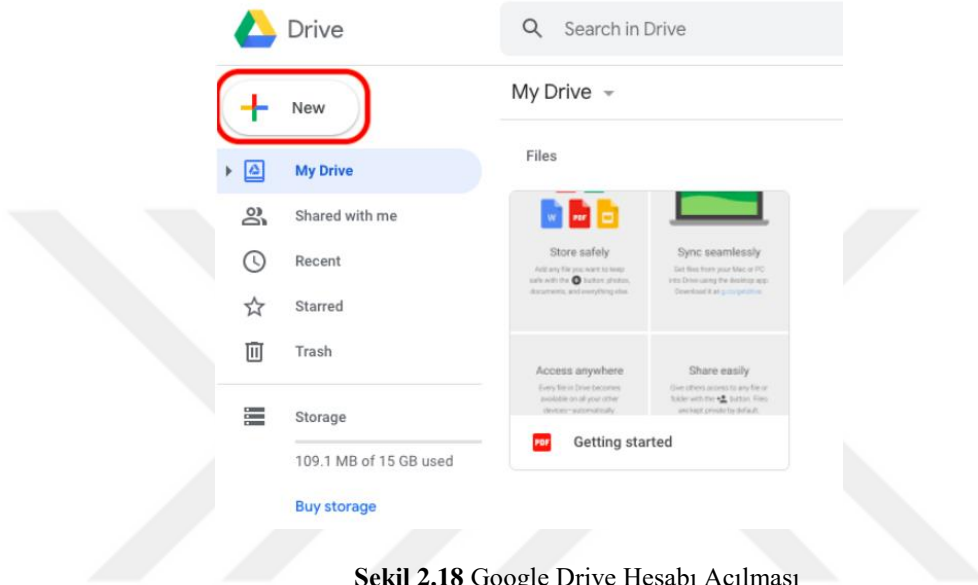
2.2.2.5 Colabratory

Google resarch tarafından sunulan; python programlama dilini destekleyen, makine öğrenimi ve veri analizi yapmaya imkan sağlayan, jupyter notebook barındıran ücretsiz

bir idedir. Sınırlı erişime sahip olsa da bize günümüzdeki en gelişmiş grafik işlemcisi Tesla K80'i kullanım imkânı vermektedir.

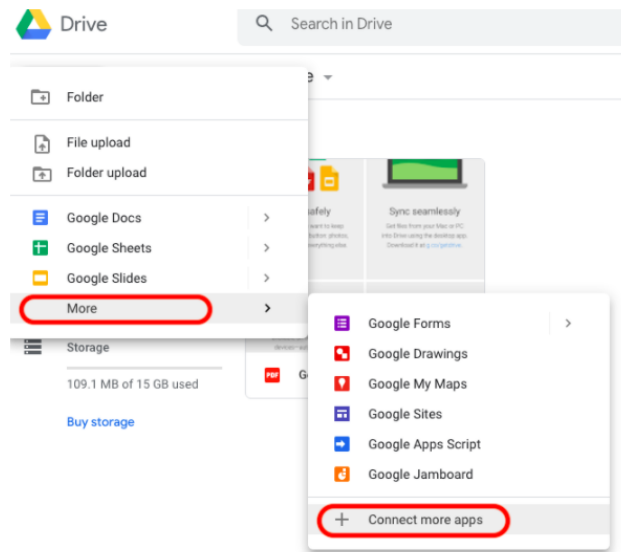
2.2.2.6 Colabratoy'nin Yüklenmesi ve Konfigürasyonu

- Google Colabı kurmak için öncelikle Şekil 2.18'de gösterildiği gibi Google Drive'da hesap oluşturulmuş ve Google Drive hesabınızda oturum açılmıştır.



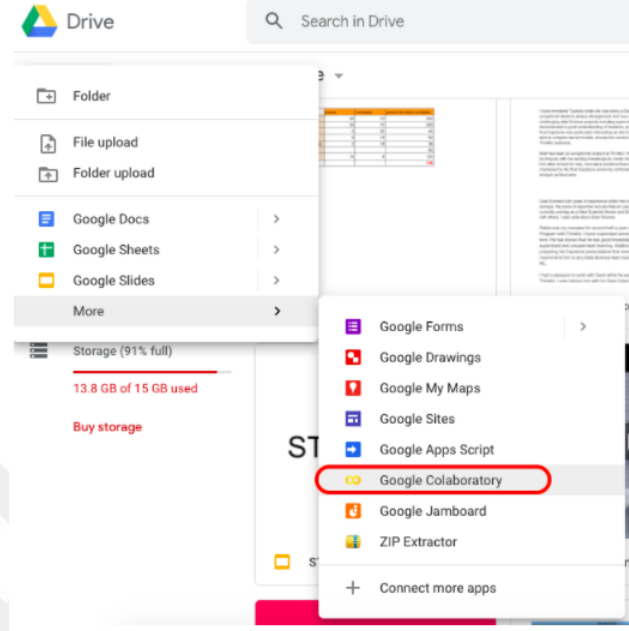
Şekil 2.18 Google Drive Hesabı Açılması

- Şekil 2.19'daki gösterildiği gibi "+ Yeni " açılır menüsünden "Daha Fazla " seçilip ve ardından ek açılır menüden " + Daha fazla uygulama bağla " yı seçerek colabratoy kurulumu sağlanmıştır.



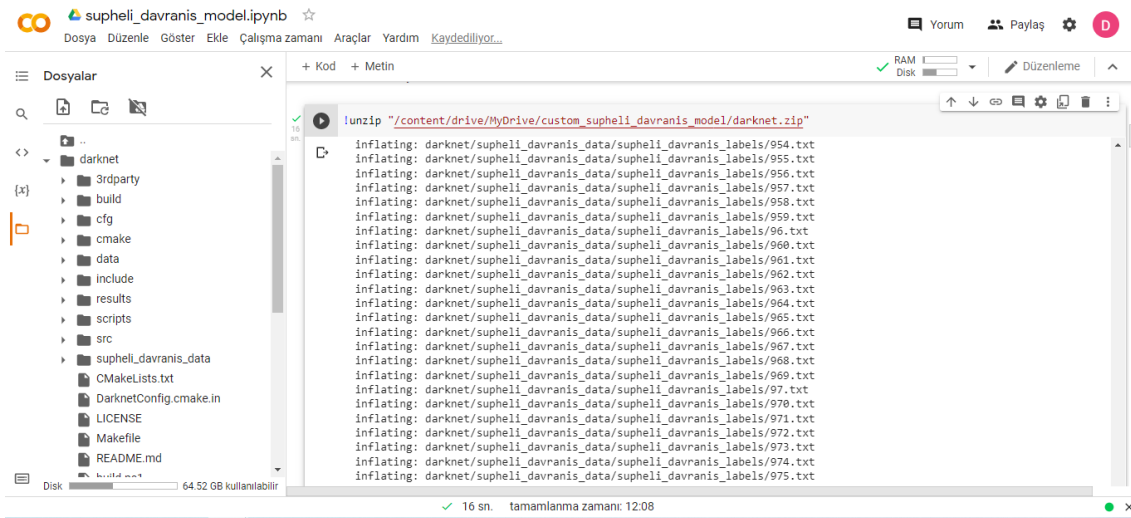
Şekil 2.19 Colabratoy Yüklenmesi

- Google Colab 'i install ettikten sonra, Google Drive'dan '+ Yeni' düğmesini seçtiğinizde açılır listenizde colab Şekil 2.20’deki gibi görüntülenecektir.



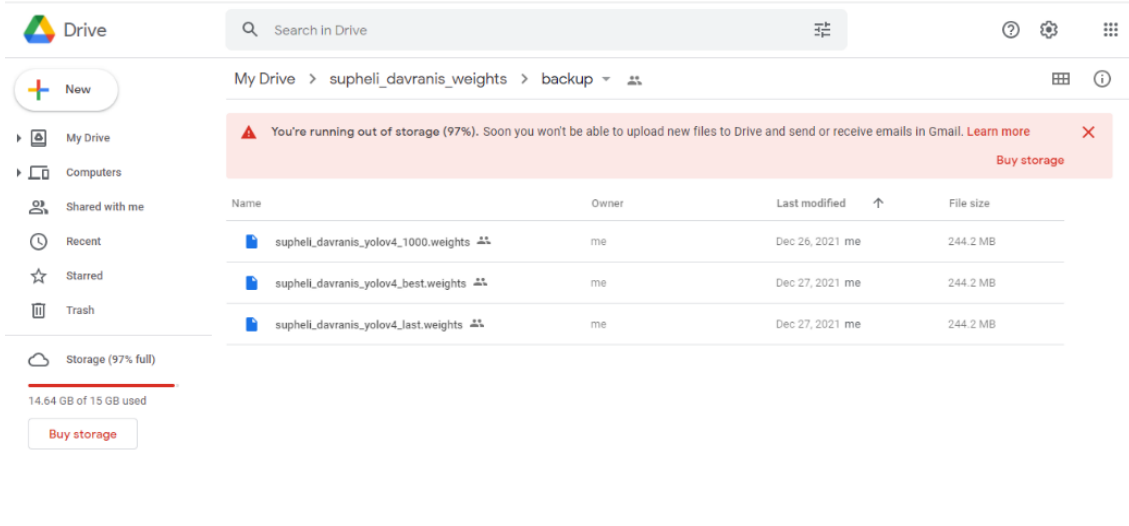
Şekil 2.20 supheli_davranis_model.ipynb Dosyasının Oluşturulması

- Colabratory’ye tıklandığında *.ipynb uzantılı çalışma ortamı açılmaktadır. Modele göre dosya adı supheli_davranis_model.ipynb olarak isimlendirip drive entegrasyonunu sağlanmıştır. *.ipynb uzantılı dosya normal bir Jupiter not defteri olarak kullanılmış ve makine öğrenmesi kodlarının çalıştırılması sağlanmıştır.



Şekil 2.21 Google Colab Çalıştırma Ortamı

- Şekil 2.21’de gösterildiği üzere konfigürasyonları yapılan darknet klasörü içerisine ön işlemesi tamamlanan etiketli veri seti sıkıştırılarak drive hesabına yüklenmiştir, `supheli_davranis_model.ipynb` dosyasında `unzip` komutu çalıştırılarak darknete colabratory üzerinden erişim sağlanmıştır. Eğitim işlemi için *.ipynb uzantılı dosyada ilgi makin öğrenmesi eğitim kodları çalıştırılmış olup; çalışmanın modeline özgü ağırlıklar hesaplanıp Şekil 2.22’de görüldüğü gibi daha sonra kullanmak üzere kayıt edilmiştir.



Şekil 2.22 Weights Değerlerinin Hesaplanması

- Şekil 2.23’de görüldüğü gibi elde edilen ağırlık (weights) değerlerine göre colabratoryde test işlemi yapılmış ve Şekil 2.24’de görüldüğü gibi örnek veriler tahmin edilmiştir.


```

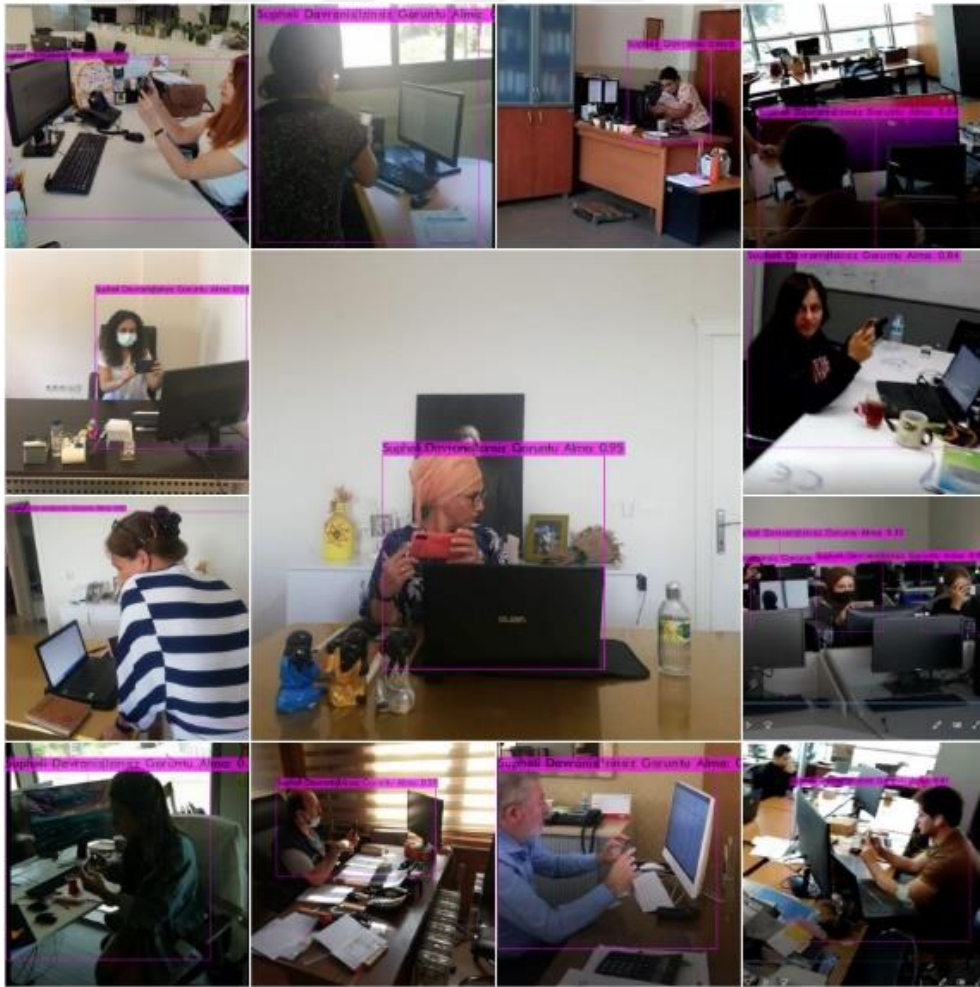
truth.x = 0.288281, truth.y = 0.589724, truth.w = 0.256491, truth.h = 0.828551, class_id = -1
Warning: in txt-labels class_id=-1 >= classes=1 in cfg-file. In txt-labels class_id should be [from 0 to 0]

TEST
./darknet_detector test suphelli_davranis_data/suphelli_davranis.data suphelli_yolov4.cfg /content/drive/MyDrive/suphelli_weights/backup/suphelli_davranis_yolov4_best

CUDA-version: 11010 (11020), cuDNN: 7.6.5, GPU count: 1
opencv version: 3.2.0
0 : compute_capability = 370, cudnn_half = 0, GPU: Tesla K80
net_optimized_memory = 0
mini_batch = 1, batch = 64, time_steps = 1, train = 0
layer filters size/stride/11 input output
0 Create CUDA-stream - 0
0 Create cudnn-handle 0
conv 32 3 x 3/ 1 416 x 416 x 3 -> 416 x 416 x 32 0.299 BF
1 conv 64 3 x 3/ 2 416 x 416 x 32 -> 208 x 208 x 64 1.595 BF
2 conv 64 1 x 1/ 1 208 x 208 x 64 -> 208 x 208 x 64 0.354 BF
3 route 1 208 x 208 x 64 -> 208 x 208 x 64
4 conv 64 1 x 1/ 1 208 x 208 x 64 -> 208 x 208 x 64 0.354 BF
5 conv 32 1 x 1/ 1 208 x 208 x 64 -> 208 x 208 x 32 0.177 BF
6 conv 64 3 x 3/ 1 208 x 208 x 32 -> 208 x 208 x 64 1.595 BF
7 Shortcut Layer: 4, wt = 0, wn = 0, outputs: 208 x 208 x 64 0.003 BF
8 conv 64 1 x 1/ 1 208 x 208 x 64 -> 208 x 208 x 64 0.354 BF
9 route 8 2 208 x 208 x 128 -> 208 x 208 x 128
10 conv 64 1 x 1/ 1 208 x 208 x 128 -> 208 x 208 x 64 0.709 BF
11 conv 128 3 x 3/ 2 208 x 208 x 64 -> 104 x 104 x 128 1.595 BF
12 conv 64 1 x 1/ 1 104 x 104 x 128 -> 104 x 104 x 64 0.177 BF
13 route 11 104 x 104 x 128 -> 104 x 104 x 128
14 conv 64 1 x 1/ 1 104 x 104 x 128 -> 104 x 104 x 64 0.177 BF
15 conv 64 1 x 1/ 1 104 x 104 x 64 -> 104 x 104 x 64 0.003 BF
16 conv 64 3 x 3/ 1 104 x 104 x 64 -> 104 x 104 x 64 0.797 BF
17 Shortcut Layer: 14, wt = 0, wn = 0, outputs: 104 x 104 x 64 0.001 BF
18 conv 64 1 x 1/ 1 104 x 104 x 64 -> 104 x 104 x 64 0.003 BF
19 conv 64 3 x 3/ 1 104 x 104 x 64 -> 104 x 104 x 64 0.797 BF
20 conv 64 3 x 3/ 1 104 x 104 x 64 -> 104 x 104 x 64 0.797 BF

```

Şekil 2.23 Weights Değerlerinin Hesaplanması



Şekil 2.24 Şüpheli Davranış | İzinsiz Görüntü Alınması Tespit Edilmiş Görüntüler

- Şekil 2.25’te modelin tahmin edemediği örnekler gösterilmiştir. Görseller incelendiğinde şüpheliler cep telefonlarını iki avuçlarının içine sakladıklarında veya iki ellerini havaya kaldırırken başka bir cisim tuttuklarında model izinsiz görüntü alınmasını tespit edememiştir.



Şekil 2.25 Şüpheli Davranış | İzinsiz Görüntü Alınması Tespit Edilememiş Görüntüler

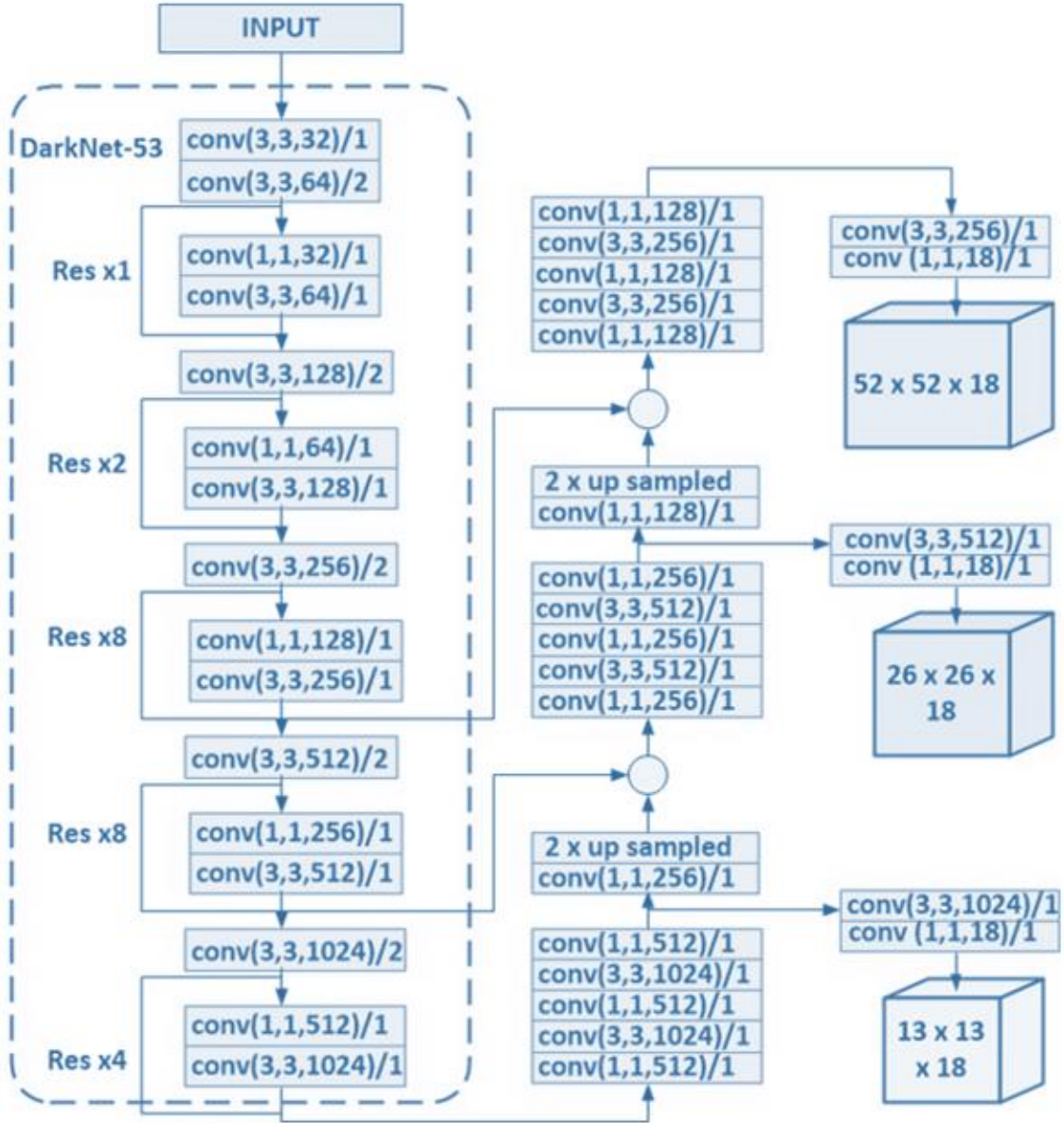
2.2.1 YOLOv4 (You Only Look Once)

Joseph Redmon ve arkadaşlarının geliştirdikleri YOLO, darknetin omurgası ile çalışabilen bir sinir ağıdır ve nesne tespitinde yüksek performans değerlerine sahip gerçek zamanlı sistem mimarilerine uygun yeni bir yaklaşımdır[21].

Mevcut nesne tespit algoritmaları bir görüntü üzerinde çeşitli işlemler gerçekleştirerek, sınıflandırıcı fonksiyonlarını tekrar tekrar kullanarak nesne tespiti gerçekleştirir. YOLO ise tek bir sinir ağı ve bileşik mimarisi ile direkt tüm görüntü üzerinde hem sınırlama

kutularını(bounding boxes) hem de sınıf olasılıklarını tahmin eder. Bu nedenle de oldukça hızlıdır. Şekil 2.26’da YOLOv4’un mimarisi gösterilmiştir.

YOLO’ nun güncel versiyonu olan YOLOv4 ise YOLOv3’e göre bazı değişiklikler yapılmış halidir. Bu çalışmada YOLOv4 kullanılmıştır.



Şekil 2.26 YOLOv4 Mimarisi

3 BULGULAR VE TARTIŞMA

Bu tez çalışmasında bir yapay sinir ağı çatısı(framework) olan darknet ve gerçek zamanlı nesne tespit aracı olan YOLO, kriminal/şüpheli görüntüler üzerinde şüpheli görüntü tespiti yapmak(izinsiz görüntü alma)amacıyla kullanılmıştır. Veri seti olarak Marmara Üniversitesi Kriminal Davranış/Nesne (MÜKDN) veri seti oluşturulmuş ve kullanılmıştır. Veri seti YOLO formatına uygun bir şekilde makesense.ai'de etiketlenmiştir.

YOLO ile derin öğrenme yapmadan önce darknet 'in okuyacağı sinir ağı konfigürasyon dosyasında bazı değişiklikler yapılarak, sinir ağının küçük nesneleri de tespit edebilir hale gelmesi sağlanmış ve eğitim işleminin süresi için optimizasyonlar yapılmıştır.

YOLO 'nun eğitim işlemi colabratory frameworkünde GPU üzerinde gerçekleştirilmiştir. Eğitim yaklaşık olarak 12 saat sürmüştür. supheli_davranis_yolov4_best.weights , supheli_davranis_yolov4_last.weights , supheli_davranis_yolov4_1000.weights ve supheli_davranis_yolov4_2000.weights değerleri elde edilmiştir.

3.1 Kullanılan Metrik Yöntemler

Derin öğrenme sistemlerinde modelin eğitim süreci sonunda elde ettiği başarı ver performansı belli ölçütlere göre değerlendirilmektedir. Bu bölümde çalışmalarda kullanılan modelin değerlendirilmesi için kesinlik (precision), duyarlılık (recall) , F1 skor ve mAP sonuçlarına değinilmiştir.

3.1.2 Kesinlik (Precision), Duyarlılık (Recall) , F1 skor ve mAP

Derin öğrenmede öğrenme başarısı olasılık modeli olan kesinlik (precision) değeri üzerinden hesaplanmaktadır. Bu skala için bazı temel kavramlar vardır. Bu kavramlardan dördü gerçekleşen öğrenmeyi test verisi üzerinde uygulandığında elde edilen tahmini sınıflandırmak için kullanılmaktadır. Bu kavramlar doğru pozitif(TP), yanlış pozitif(FP), doğru negatif(TN) ve yanlış negatif(FN) değerleridir.

Bu değerler, kullanıcı tarafından şüpheli davranış olarak etiketlenmiş verinin etiket değeri ve yapıla eğitim sonucundaki ağırlık değerlerine göre sistem tarafından yapılan tahmin sonucunun karşılaştırması ile elde edilmektedir. Şüpheli davranış olarak etiketlenmiş verinin pozitif olduğu varsayılırsa, tahmin sonucu eğer pozitif ise TP, negatif ise FN değeri oluşmaktadır. Yine aynı şekilde şüpheli davranış olarak etiketlenmiş verinin

negatif olduğu varsayılırsa, tahmin sonucu eğer pozitif ise FP, negatif ise TN değeri oluşmaktadır. Karmaşıklık matrisini Tablo 3.1’de görmek mümkündür.

Şüpheli davranış pozitif, normal davranış negatif kabul edilirse;

- ***True Positive(Doğru Pozitif - TP)*** : Etiketlenmiş verinin pozitif, sistem pozitif sınıfta tespit etmiş. Yani Şüpheli davranışı şüpheli davranış olarak tespit etmiş.

- ***False Positive(Yanlış Pozitif - FP)*** : Etiketlenmiş verinin pozitif, sistem negatif sınıfta tespit etmiş. Yani Şüpheli davranışı şüpheli davranış değil olarak tespit etmiş.

- ***True Negative(Doğru Negatif - TN)*** : Etiketlenmeyen veri(Normal Davranış) negatif, sistem negatif sınıfta tespit etmiş. Yani şüpheli olmayan davranışı şüpheli davranış değil olarak tespit etmiş.

- ***False Negative(Yanlış Negatif – FN)*** : Etiketlenmeyen veri(Normal Davranış) negatif, sistem şüpheli davranış olarak tespit etmiş

Yani normal davranışı şüpheli olarak tespit etmiş.

Tablo 3.1 Karmaşıklık Matrisi

Tahmin Edilen Durum	Etiketlenmiş Durum		
	Toplam Durum	Etiket Pozitif	Etiket Negatif
	Tahmin Pozitif	Doğru Pozitif (TP)	Yanlış Pozitif (FP)
	Tahmin Negatif	Yanlış Negatif (FN)	Doğru Negatif (TN)

Karmaşıklık matrisinin terimleri kullanılarak her bir örnek için Precision(hassasiyet) ve Recall(geri çağırma) değerleri hesaplanır.

Kesinlik(Precision) değerinin formülünü Denklem 3.1’de görmek mümkündür.

$$\text{Precision} = p = \frac{\text{Doğru Pozitifler}}{\text{Toplam Pozitif Sonuçlar}} = \frac{TP}{TP+FP} \quad (3.1)$$

Duyarlılık(Recall) değerinin formülünü Denklem 3.2’de görmek mümkündür.

$$\text{Recall} = r = \frac{\text{Doğru Pozitifler}}{\text{Tespit edilen şüpheli davranış sayısı}} = \frac{TP}{TP+FN} \quad (3.2)$$

Test verisindeki her bir örnek için bir test çalıştırılır ve elde edilen sonuca göre recall(r) ve precision(p) değerleri yüzdelik olarak hesaplanır. Bu değerlere göre bir r-p grafiği oluşur. r-p grafiği altında kalan alan Average Precision(Ortalama hassasiyet - AP) olarak hesaplanmaktadır. Average Precision formülünü Denklem 3.3’ de görmek mümkündür.

$$AP = \int_0^1 p(r)dr \quad (3.3)$$

Bu formül pratikte sonlu bir toplam olarak Denklem 3.4’ deki gibi kullanılır.

$$AP = \sum_{k=1}^n P(k) \cdot \Delta r(k) \quad (3.4)$$

n: nesne sayısı

P(k) : Her bir k değeri için precision

$\Delta r(k)$: Her bir k değeri için recall değerindeki değişim.

Her bir sınıf için AP hesabı yapıldıktan sonra tüm sınıflar için Mean Average Precision(mAP) değeri Denklem 3.5’ deki gibi hesaplanır.

$$mAP = \frac{\sum_{i \in \text{sınıflar}} AP_i}{\text{Toplam Sınıf Sayısı}} \quad (3.5)$$

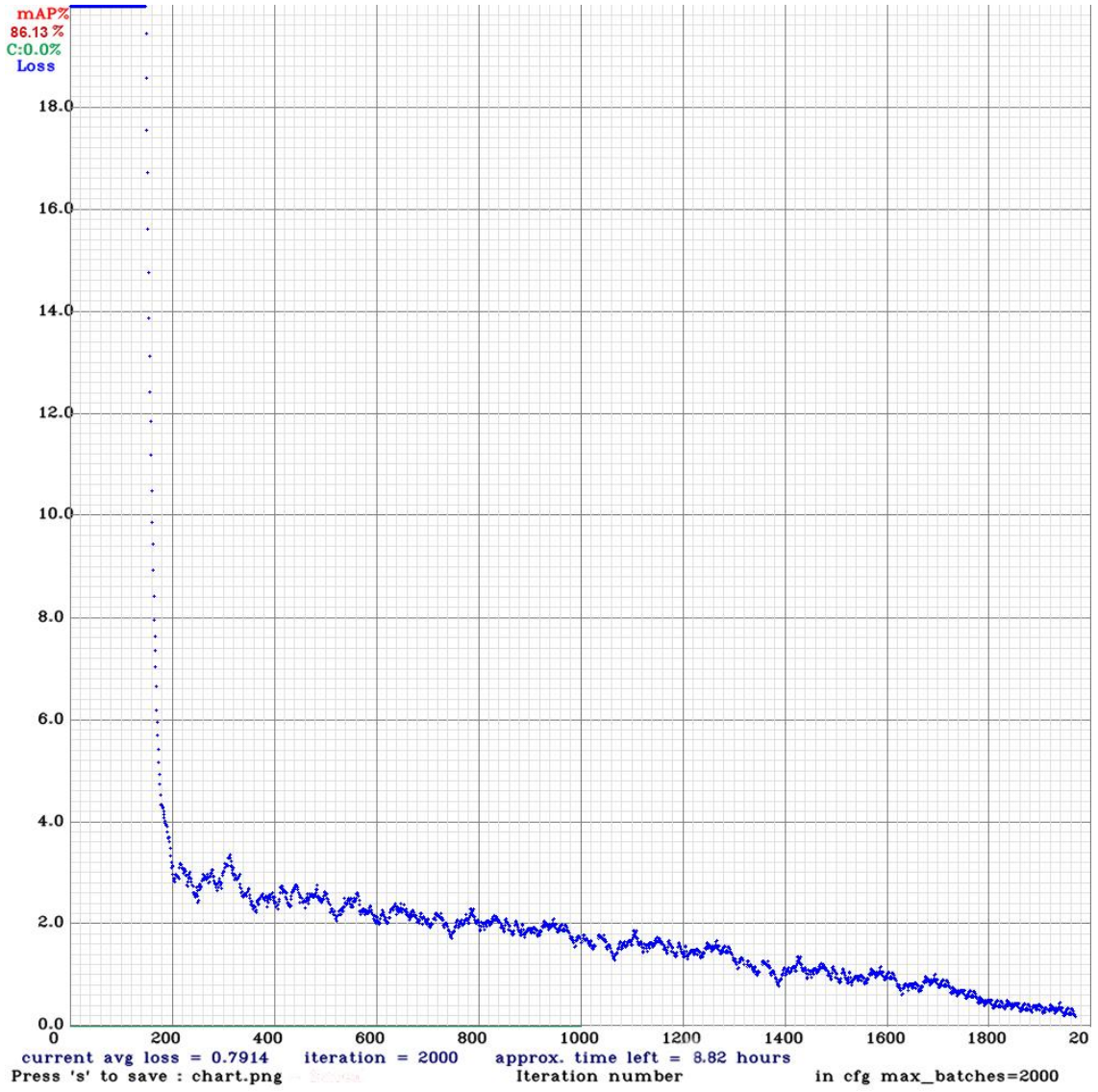
Tablo 3.2 YOLOv4’ün Metrik Hesaplama Değerleri

Model Adı	F1 Skor	mAP	Iou	Tespit Zamanı	Güven Skoru Eşik Değeri
YOLOv4	0,721	0,50	0,79	0,232	0,25

Tablo 3.2’ de YOLOv4’ e ait F1-Skor, mAP, IoU, tespit zamanı ve güven skoru eşik değerleri gösterilmiştir. Tablo 3.3te YOLOv4’ün tek sınıf ile eğitim başarısı mAP değerleri iterasyon sayısına göre gösterilmiştir. Şekil 3.1’ de şüpheli davranış tespit modelimizin YOLOv4 algoritması kullanılarak 2000 iterasyonda tamamlanan eğitiminin çizdirilen mAP değerleri grafiği gösterilmiştir.

Tablo 3.3 YOLO' nun Tek Sınıf İle Öğretildiğinde Başarısı

Tekrar(Iteration)	mAP
500	29,68
1000	78,31
1500	82,86
2000	86,13



Şekil 3.1 Şüpheli Davranış Eğitim Grafiği

4 SONUÇ

Bu çalışmada gerçek zamanlı bir nesne tespit aracı olan YOLO Kriminal/şüpheli davranış görüntülerinde şüpheli davranış tespiti amacıyla kullanılmıştır ve başarılı sonuçlar elde edilmiştir. YOLO ile yapılan çalışmalar çoğunlukla günlük hayatta gözle görülebilen nesnelerin makineye öğretilip tespit etmesi amacıyla kullanılmaktadır. Bu alandaki yeni veri seti olan Marmara Üniversitesi Kriminal Davranış/Nesne (MÜKDN) veriseti üzerinde yapılan bu çalışma sayesinde YOLO' nun davranış tespitinde kullanılabilirliği desteklenmiştir. Bir nesne tespit aracı olan YOLO ile şüpheli davranış tespitinde %70'in üzerinde başarı elde edilmiştir.

Derin öğrenmede başarıyı etkileyen en önemli faktörlerden biri elde edilmesi oldukça maliyetli olan eğitim verisinin çok olmasıdır. Bu çalışmada veri seti %70 eğitim - %30 test şeklinde ayarlanarak eğitim gerçekleştirildiğinde, mAP değeri 0.5' in altında çıkmıştır. %86,66'in üzerindeki başarıya ulaşılmıştır. Daha büyük bir veri seti ile başarının daha da yüksek çıkması beklenmektedir.

GPU gücünün makine öğrenmesi için fazlasıyla yeterli olduğu günümüzde en önemli ihtiyaç veridir. Makine ne kadar çok veri ile eğitilirse, insan davranışının makinelerce taklit edilebilmesi o kadar iyi sağlanabilir. Günümüzde teknoloji kullanımının (sosyal medya vb.) artması sayesinde çok fazla veri birikmektedir. Etik kurallarına da uyararak, kişisel veri olmayacak verilerden veya kişisel verileri barındırmayacak şekilde toplanacak verilerle veri setleri oluşturulması ve bu verilerin gelecekte yapılacak bilimsel çalışmalara da fayda sağlaması için paylaşılması önemlidir.

Bu çalışmada kullanılan Marmara Üniversitesi Kriminal Davranış/Nesne (MÜKDN) veri seti oluşturulmuş olup, YOLO formatında etiket yapısı kazandırılmıştır, bu veriler paylaşılacaktır.

Bu çalışmada kullanılan *.cfg dosyasındaki değerler optimum değerler değildir. Ancak mevcut hali ile şüpheli davranış tespiti yapılabilmektedir. *.cfg dosyasında yapılabilecek başka ayarlar ve YOLO katman sayısının artırılması gibi değişiklikler ile YOLO'nun başarısını artırabilir.

YOLO tek sınıf ile eğitilmiş olup, sınıf sayısının eğitimdeki başarıya etki ettiği ve tek sınıflı eğitimde yüksek başarıya daha hızlı ulaşıldığı görülmüştür.

Bu alıřmada sadece izinsiz grnt alma řpheli davranıřı tespiti yapılmıřtır. YOLO ile bařka davranıř trleri (hırsızlık, řiddet vb.) zerinde de alıřılabilir ve farklı bařarı oranlarına ulařılabilir.



REFERANSLAR

- [1] Radu, R. G. (2012). The monopoly of violence in the cyber space: Challenges of cyber security. In *Power in the 21st Century* (pp. 137-150). Springer, Berlin, Heidelberg.
- [2] Bıçakcı, S. (2014). NATO'nun gelişen tehdit algısı: 21. yüzyılda siber güvenlik. *Uluslararası İlişkiler Dergisi*, 10(40), 100-130.
- [3] Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278–2324. <https://doi.org/10.1109/5.726791>
- [4] Wikipedia (Ed.). (2021). Yapay zekâ. (2021, September 13). https://tr.wikipedia.org/w/index.php?title=Yapay_zekâ&oldid=25937799
- [5] Bonner, A. (2007). The art and logic of Ramon Lull: A user's guide. *Studien und Texte zur Geistesgeschichte des Mittelalters: Bd. 95*. Brill.
- [6] Turing, A. M. (2009). Computing machinery and intelligence. In *Parsing the turing test* (pp. 23-65). Springer, Dordrecht.
- [7] John McCarthy, Marvin L. Minsky, Nathaniel Rochester, & Claude E. Shannon. (1955). A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence (27 Number 4(2006)). *AI Magazine Volume 27 Number 4* (2006) (© AAAI).
- [8] McCulloch, W. (1943). A logical calculus of the ideas immanent in nervous activity, 5.
- [9] Rosenblatt, F. (1957). *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory..
- [10] Christopher, M. B. (2016). *Pattern Recognition And Machine Learning*: Springer-Verlag New York.
- [11] Karpathy, A. (2018). Stanford university cs231n: Convolutional neural networks for visual recognition. URL: <http://cs231n.stanford.edu/syllabus.html>.
- [12] Yapay Sinir Ağları ~ Papatya Bilim Yayınevi. (2020, October 24). <http://www.papatyabilim.com.tr/yapaySinirAglari.htm>

- [13] nvidia.com ,” Deep Learning”,(2022, January 4).
<https://developer.nvidia.com/deep-learning>
- [14] Fırıldak, K., & Talu, M. F. (2019). Evrişimsel Sinir Ağlarında Kullanılan Transfer Öğrenme Yaklaşımlarının İncelenmesi. *Computer Science*, 4(2), 88-95.
- [15] Girshick, R., Donahue, J., Darrell, T., Malik, J., & Mercan, E. (2014). R-CNN for Object Detection. In *IEEE Conference*.
- [16] Girshick, R., Donahue, J., Darrell, T., & Malik, J. (Eds.). Rich feature hierarchies for accurate object detection and semantic segmentation.
- [17] Gkioxari, G., Girshick, R., & Malik, J. (Eds.). Contextual Action Recognition with R*CNN.
- [18] Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6), 1137–1149.
<https://doi.org/10.1109/TPAMI.2016.2577031>
- [19] Leibe, B., Matas, J., Sebe, N., & Welling, M. (Eds.) (2016a). *Computer Vision – ECCV 2016. : Vol. 9905*. Springer International Publishing.
- [20] Leibe, B., Matas, J., Sebe, N., & Welling, M. (2016b). *Computer Vision – ECCV 2016, 9905*. <https://doi.org/10.1007/978-3-319-46448-0>
- [21] Joseph Redmon, Santosh Divvala, Ross Girshick, & Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection.
- [22] Papers with Code - YOLOv1 Explained. (2021, September 13).
<https://paperswithcode.com/method/yolov1>
- [23] Papers with Code - YOLOv2 Explained. (2021, September 13).
<https://paperswithcode.com/method/yolov2>
- [24] Papers with Code - YOLOv3 Explained. (2021, September 13).
<https://paperswithcode.com/method/yolov3>
- [25] Papers with Code - YOLOv4 Explained. (2021, September 13).
<https://paperswithcode.com/method/yolov4>

- [26] bigumigu.com -Hırsızlığı % 81 Doğrulukla Önceden Tespit Edebilen Algoritma.(2021, Sebtember 17). <https://bigumigu.com/haber/vaakeye-hirsizligi-81-dogrulukla-onceden-tespit-edebilen-algoritma/>
- [27] Maghdid, H., Asaad, A. T., Ghafoor, K. Z. G., Sadiq, A. S., Mirjalili, S., & Khan, M. K. K. Diagnosing COVID-19 pneumonia from x-ray and CT images using deep learning and transfer learning algorithms, 26. <https://doi.org/10.1117/12.2588672>
- [28] Hassan, M., Ali, S., Alquhayz, H., & Safdar, K. (2020). Developing intelligent medical image modality classification system using deep transfer learning and LDA. *Scientific Reports*, 10(1), 12868. <https://doi.org/10.1038/s41598-020-69813-2>
- [29] Jaouedi, N., Boujnah, N., & Bouhlel, M. S. (2020). A new hybrid deep learning model for human action recognition. *Journal of King Saud University - Computer and Information Sciences*, 32(4), 447–453. <https://doi.org/10.1016/j.jksuci.2019.09.004>
- [30] T.S., A., & Guddeti, R. M. R. (2020). Affective database for e-learning and classroom environments using Indian students' faces, hand gestures and body postures. *Future Generation Computer Systems*, 108, 334–348. <https://doi.org/10.1016/j.future.2020.02.075>
- [31] Jammalamadaka, N., Zisserman, A., & C.V., J. (2017). Human pose search using deep networks. *Image and Vision Computing*, 59, 31–43. <https://doi.org/10.1016/j.imavis.2016.12.002>
- [32] Luvizon, D. C., Tabia, H., & Picard, D. (2019). Human pose regression by combining indirect part detection and contextual information. *Computers & Graphics*, 85, 15–22. <https://doi.org/10.1016/j.cag.2019.09.002>
- [33] Moez Baccouche, Franck Mamalet, Christian Wolf, Christophe Garcia, and Atilla Baskurt. LNCS 7065 - Sequential Deep Learning for Human Action Recognition.
- [34] Laptev, I., Marszalek, M., Schmid, C., & Rozenfeld, B. Learning realistic human actions from movies, 1–8. <https://doi.org/10.1109/CVPR.2008.4587756>

- [35] Dollar, P., Rabaud, V., Cottrell, G., & Belongie, S. Behavior Recognition via Sparse Spatio-Temporal Features, 65–72. <https://doi.org/10.1109/VSPETS.2005.1570899>
- [36] Ming-Yu Chen and Alexander Hauptmann. MoSIFT: Recognizing Human Actions in Surveillance Videos.
- [37] Liu, J., & Shah, M. Learning human actions via information maximization, 1–8. <https://doi.org/10.1109/CVPR.2008.4587723>
- [38] Daş, R., Polat, B., & Tuna, G. (2019). Derin Öğrenme ile Resim ve Videolarda Nesnelerin Tanınması ve Takibi. Fırat Üniversitesi Mühendislik Bilimleri Dergisi, 571–581. <https://doi.org/10.35234/fumbd.608778>
- [39] Yılmaz, O., Aydın, H., & Çetinkaya, A. (2020). Faster R-CNN Evrimsel Sinir Ağı Üzerinde Geliştirilen Modelin Derin Öğrenme Yöntemleri ile Doğruluk Tahmini ve Analizi: Nesne Tespiti Uygulaması. *Avrupa Bilim ve Teknoloji Dergisi*, (20), 783-795.
- [40] pjreddie.com, "Darknet: Open Source Neural Networks in C", (2021, September13). <https://pjreddie.com/darknet/>
- [41] github.com," YOLOv4 / Scaled-YOLOv4 / YOLO - Neural Networks for Object Detection (Windows and Linux version of Darknet) ",(2021, September 13). <https://github.com/AlexeyAB/Darknet>

ÖZGEÇMİŞ

Adı Soyadı : Duygu ÇALIŞKAN

Yabancı Dili : İngilizce, Arapça

Öğrenim Durumu

Derece	Bölüm/Program	Üniversite/Lise	Mezuniyet Yılı
Lise	Fen Bilimleri	Fatih Anadolu Lisesi	2009
Lisans	Elektronik ve Haberleşme Mühendisliği	Kocaeli Üniversitesi	2016

İş Deneyimi

Yıl	Firma/Kurum	Görevi
2016-2020	Marmara Üniversitesi Yazılım Birimi	Software Developer
2020-Halen	İçişleri Bakanlığı EGM Bilgi Teknolojileri ve Haberleşme Şube Müdürlüğü	Elektronik ve Haberleşme Mühendisi