T.C.
PAMUKKALE ÜNİVERSİTESİ
İKTİSADİ VE İDARİ BİLİMLER FAKÜLTESİ
YÖNETİM BİLİŞİM SİSTEMLERİ
LİSANS BİTİRME PROJESİ

YOLO MODELLERİ KULLAN MASKE TESPİT UYGULAMASI

Hazırlayan
Hasan Hüseyin Eren

Danışman
Prof. Dr. Selçuk Burak HAŞILOĞLU

Haziran, 2021
DENİZLİ

İçindekiler

ŞEKİLLER LİSTESİ

ÖZET

Aralık 2019'da Çin'in Wuhan kentinde ortaya çıkan korona virüsü (COVID-19) ortaya çıkmış, Dünya Sağlık Örgütü (WHO) bu virüsün damlacık ve hava yoluyla bulaşabilen ölümcül bir virüs olduğunu doğrulamış akabinde dünyada COVID-19 pandemisi ilan edilmiştir. Önleme olarak insanların dışarıda insanlar arası mesafelerine dikkat etmesi önerilmiş ve maske takma zorunluluğu getirilmiştir. Bazı insanlar birtakım bahanelerle maske takmayarak insanların ve kendi hayatlarını tehlikeye atmaktadır.

Bu çalışmada, yüz maskesini tespit edebilen bir maske detektörü geliştirmeyi amaçlamaktadır. Bu tespiti gerçekleştirmek için derin öğrenme modeli olan You Only Look Once anlamına gelen YOLO kullanılmaktadır. Model maske takan ve takmayan olarak iki kategorideki insanların görüntüleri kullanılarak eğitilmiştir. Çalışma kişinin maske takıp takmamasına göre kutular çizmektedir. İnsanların yüzlerine maske takmadıysa kırmızı bir kutu, maske taktıysa yeşil bir kutu çizmekte ve maske takan, maske takmayan ve toplam insan sayısı bilgisini bize vermektedir.

1. Giriş

Korona virüs salgını sanayiden tarıma, ekonomiden ulaşıma dünyadaki çeşitli sektörleri önemli ölçüde etkiledi. Bu etki sektörlerin durmasına, işlemlerin aksamasına neden oldu. Bu virüs insandan insana damlacık ve hava yoluyla bulaşabilmektedir. DSÖ yayılmanın azaltılması için kalabalıktan kaçınılması, bağışık sisteminin güçlü tutulmasını önerdi bunun yanı sıra önlem olarak sosyal mesafeye uyulması, açık ve kapalı alanlarda zorunlu maske kullanımı uyulması zorunlu olan katı kuralları da beraberinde getirdi. İnsanlar açık ve kapalı alanlarda sosyal mesafelerine dikkat etmeli, maskelerini düzgün takmalıdır. Bu kurallarla birlikte insanların bir kısmı maskelerini takmazlar.

Bu kuralların açık alanlarda insan işçiliğiyle denetlenmesi son derece zordur. Denetleme işleminin gerçekleşmesi için nesne tespit sistemleri gereklidir. Nesne tespitini gerçekleştiren yöntemler vardır. Bunların başında gelen görüntü kaynaklarından nesne tespiti yapabilen R-CNN ve türevleridir. Bu yöntemlerin en büyük problemi ağır olmasıdır. Bu modelleri gerçek zamanlı çalıştırmak yüksek işlem gücüne ihtiyaç duyar bu da maliyetlidir. Bu nedenle bu projede aynı doğruluğu sunan ve daha hızlı çalışan YOLO modeli kullanacağız.

Makalenin geri kalanı şu şekilde düzenlenmiştir:

1. Giriş
2. Literatürde YOLO ile ilgili çalışmalar
3. Amaç ve Problemin belirlenmesi
4. Sistemin Analizi
5. Materyal ve Yöntemler
6. Yazılımın Raporu
7. Kullanım Kılavuzu
8. Sonuç

2. İlgili Çalışmalar

Bu bölümde YOLO, Derin öğrenme yöntemleri kullanılarak yapılan tespit çalışmaları bulunmaktadır.

YOLOv3 ile insan güvenliği için Covid-19 yüz maskesini sınıflandırmada derin öğrenmeye dayalı yardımcı sistem çalışmasında, YOLOv3 mimarisi kullanılarak bir kişinin maske takıp takmadığını tespit eden bir yaklaşım sunulmuştur. Yaklaşımın maskeli ve maskesiz sınıfları algılama performansını kontrol etmek için gerçek zamanlı videoya uygulanmış ve etkileyici sonuçlar alınmıştır.[1]

Başka bir çalışmada, derin öğrenme teknikleri kullanılarak oluşturulan alarm sistemiyle gerçek zamanlı maske tespiti çalışması yapılmıştır. Bu çalışmada tespit için Konvolüsyonel Sinir Ağları yoluyla derin öğrenme teknikleri kullanılmış ve kişin maske takıp takmadığını %96 doğruluk oranıyla tespit etmiştir. [2]

İnsan hareketi tanıma için yapılan öneride YOLO ile kişiler tespit edilmiş ve Konvolüsyonel Sinir Ağı(KSA) kullanılarak hareketlerin sınıflandırılması sağlanmıştır. Çalışmada insan hareketlerini otomatik olarak tanınması işlemi yapılmıştır. KSA ile popüler derin öğrenme modelleri karşılaştırılmış ve önerilen yönetim daha yüksek doğruluk oranları ile sonuçlar ürettiği görülmüştür.[3]

YOLO v3 kullanılarak yapılan çekirdek algılama çalışmasında Faster R-CNN, R-FCN ve SSD karşılaştırılmıştır. Sonuçlara baktığımızda karşılaştırma sonucunda en avantajlı yönü algılama hızı olmuş, sağladığı yüksek keskinlik ve hassasiyet ile kendini deneysel olarak kanıtlamıştır. Yöntem sonuç olarak %94.10 kesinlik, %98.98 hassasiyet ve %96.48 F-ölçüt değerleri sağlamıştır. [4]

3. Amaç ve Problem

Bu tezin amacı, YOLOv3, YOLOv3-tiny, YOLOv4, YOLOv4-tiny algoritması kullanarak ortamdaki insanların maske takıp takmadığını tespit edebilen uygulamanın geliştirilmesidir.

Çözülmesi beklenen problemler:

1. Tezde kullanılan YOLO algoritmalarından hangisinin maske tespitinde daha iyi performans gösterdiğini ölçme.
2. Gerçek zamanlı nesne tespiti çalışması yapılacaksa bunun hangi YOLO algoritmayla yapılacağının belirlenmesi.
3. Modelin ağırlık ve konfigürasyon dosyalarının verildiği için herkesin kendi çalışmalarında kullanabilmesi.

4. Sistem Analizi

Sistemin Planlanması: Konunun araştırılması, alternatif konuların ortaya konması

Sistemin Analizi: Konun belirlenmesi, konu hakkında neler yapılabilir, kullanılacak teknolojilerin artı ve eksi yönlerinin belirlenmesi, değerlendirilmesi ve kararlaştırılması.

Sistemin Tasarımı: Uygulamanın nasıl çalışacağı, tasarımın nasıl olacağının karar verilmesi. Kullanıcılar için kullanılabilecek en kolay ara yüzün tasarlanması.

Yazılımın Raporu: Planlama, analiz, tasarımda aşamalarında yapılan işlemlerin, geçen sürecin anlatılması.

Yazılım Aşaması: Maske Tespit Uygulaması, modelin eğitimi için Google'ın ücretsiz sunduğu Colab, uygulamanın yazılması için Python, arayüz için EasyGUI, görüntüler üzerinde işlemler için OpenCV kullanılmıştır.

5. Materyal ve Yöntem

Şekil 1 Problemin Çözümü İçin Takip Edilecek Aşamalar

5.1. Materyaller

5.1.1. YOLO Algoritması

YOLO son zamanlarda ortaya çıkmış, nesne tespiti için kullanılan algoritmalardan biri olup gerçek zamanlı görüntü işleme için kullanılan bir nesne algılama sistemidir.

YOLO diğer algoritmalara kıyasla daha hızlı tahmin yapabilmektedir. Nedeni ise YOLO resme tek bir CNN uygular, resmi ızgaralara böler, sınırlayıcı kutular ve her ızgara için güven puanını hesaplar, güven puanı ile sınırlayıcı kutular analiz edilir. [5]

YOLO algoritması 40-90 FPS arasında görüntüleri işleyebilir. Bu yüzden diğer algoritmalara göre oldukça hızlıdır. YOLO, R-CNN ile kıyaslandığında 1000 kat, Faster R-CNN ile kıyaslandığında iste 100 kat daha hızlı olduğu söylenmektedir. [5]

5.1.2. NumPy

NumPy, bilimsel hesaplamalarda kullanılan temel pakettir. Matris ve çok boyutlu dizi veri formatlarından oluşur. İstatistiksel, cebirsel ve trigonometrik rutinler gibi diziler üzerindeki işlemler NumPy yardımıyla gerçekleştirilebilir. Dizileri hesaplama ve denetlemek için gerekli araçları sunar. SciPy bunun üzerine geliştirilmiştir. Yüksek performans sunar ve çeşitli mühendislik ve bilimsel uygulamalar için gereklidir. Ön işleme aşamalarında bir görüntü 416x416 olarak ayarlanır. Daha sonra NumPy dizi stiline dönüştürülür. Daha sonra veri kümesi görüntülerine kesin etiketler dahil edilir.

5.1.3. OpenCV

OpenCV, açık kaynak kodlu bir görüntü işleme kütüphanesidir. Bilgisayarlı görü ve derin öğrenmede kullanılır. Bize yüz maskesi tespitinde, nesne tespitinde ve derin öğrenme algoritmalarında kullanılabilecek kütüphaneler sunar. Biz bu modelde algılama sürecinde, tespit edilen nesnenin etiketini basma, yüzdesini gösterme ve karesini çizdirmede kullanıyoruz.

5.1.4. EasyGUI

EasyGUI, basit arayüzler yapabileceğimiz bir GUI kütüphanesidir. Basit arayüz tasarımlarında sıkça kullanılır. Biz model, resim, video, kamera gibi seçimler yapmamıza yardımcı olmanın yanı sıra dosya açma, kaydetme, mesaj kutusu oluşturmada kullanıyoruz.

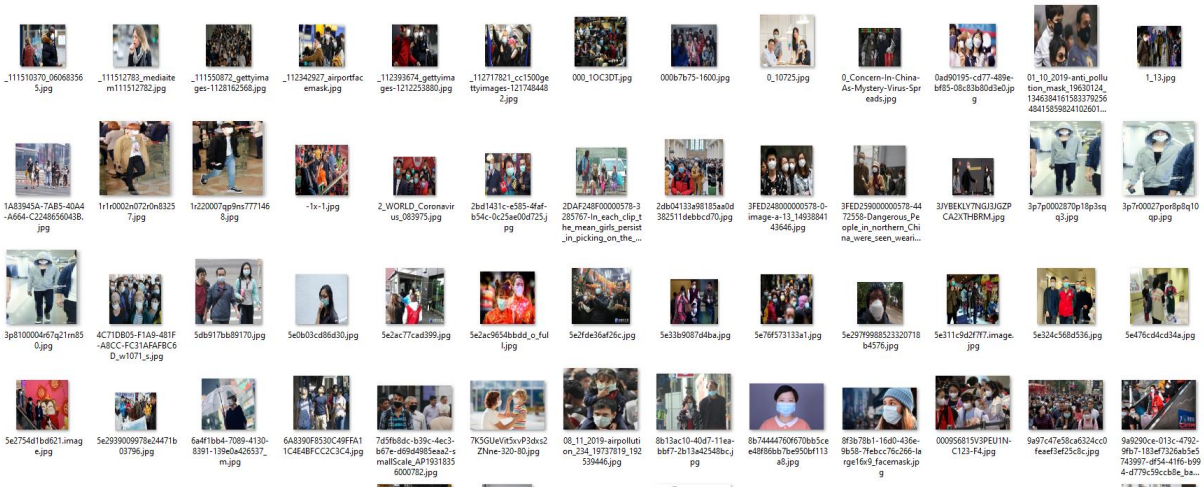
5.1.5. Google Colab

Google Colab, Jupyter Notebook benzeri makine öğrenmesi ve eğitimini yaygılaştırmak için oluşturulan bir bulut hizmetidir. Derin öğrenme için yapılandırılan bir çalışma zamanı ve güçlü bir GPU kullanımına ücretsiz erişim sağlar.

5.2. Yöntem

5.2.1. Veri Setinin Oluşturulması

İlk olarak tanınması istenen nesnenin farklı tarzdaki fotoğrafları elde edilmiştir. Farklı tarzlarda olmasının nedeni, uygulamanın nesneyi herhangi bir ortam fark etmezsiniz tanıma olasılığının yüksek olmasının istenmesidir. Fotoğrafların piksellerinde değişiklik yapılmamış olup piksel düştükçe tutarlılık düşecek eğitimi kolaylaşacak, piksel arttıkça tutarlılık artacak eğitimi zorlaşacaktır.



Resim 1 Veri Setinin Oluşturulması

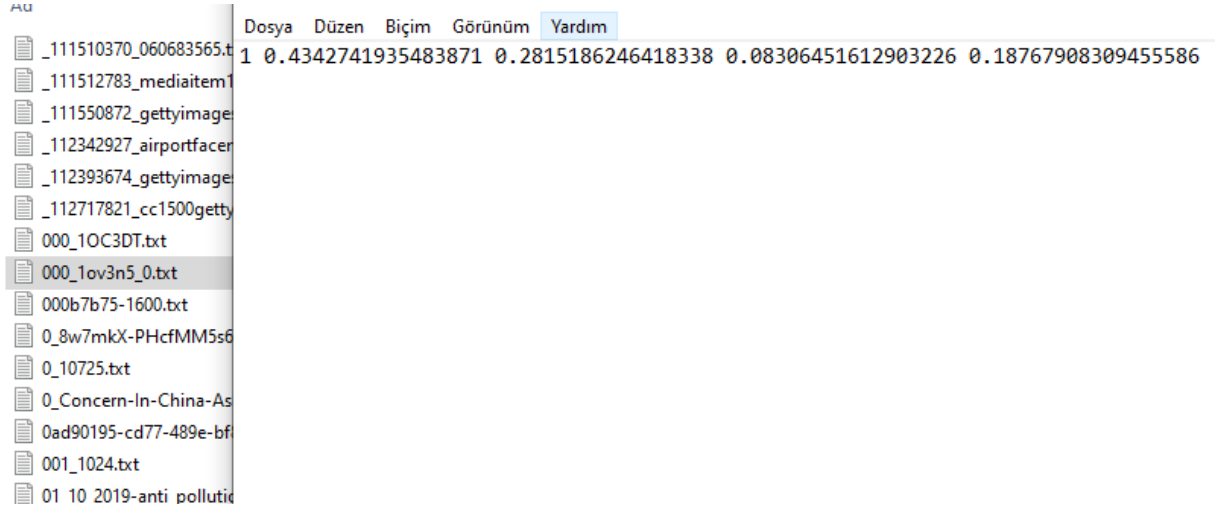
5.2.2. Veri Setinin Etiketlenmesi

Oluşturduğumuz veri setindeki fotoğraflar etiketlenerek koordinatlarını belirlemek amacıyla makesense.ai sitesi kullanılmıştır.

Veri setindeki tüm görüntüler siteye eklenmiş maskeli ve maskesiz olarak tek tek etiketlenmiştir. Etiketleme yaparken çizilen sınırların aşılmamasına dikkat edilmiştir. Sınırlar aşıldığında nesne tanıma olasılığı düşecektir.

Resim 2 Görüntüleri Etiketleme İşlemi

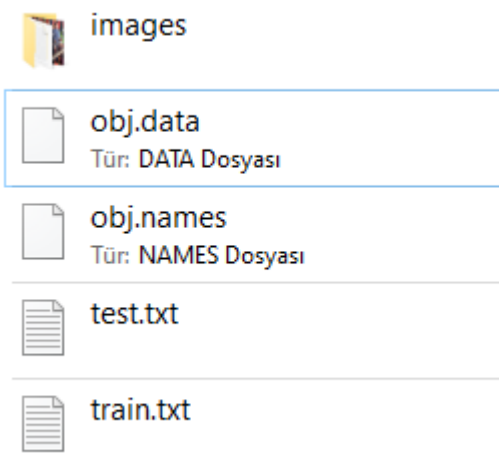
Etiketleme işlemi bittikten sonra her görüntüye özel oluşturulmuş koordinatların bulunduğu etiketli metin dosyası oluşturulmuş bir klasöre kaydedilmiştir.



Resim 3 Etiketlenen Görüntülerin Koordinatları

5.2.3. Eğitim ve Test verilerinin oluşturulması

Algoritmayı kendi verilerimiz ile eğitebilmemiz için verileri test ve eğitim olmak üzere ikiye ayırmamız gerekmekte. Bu yüzden görüntülerin yüzde 10'u test, yüzde 90'ı eğitim için kullanılmıştır. Görüntüler rastgele seçilmiştir. Test ve eğitim için belirlenen görüntüler "test.txt" ve "train.txt" dosyasında belirlenmiştir.



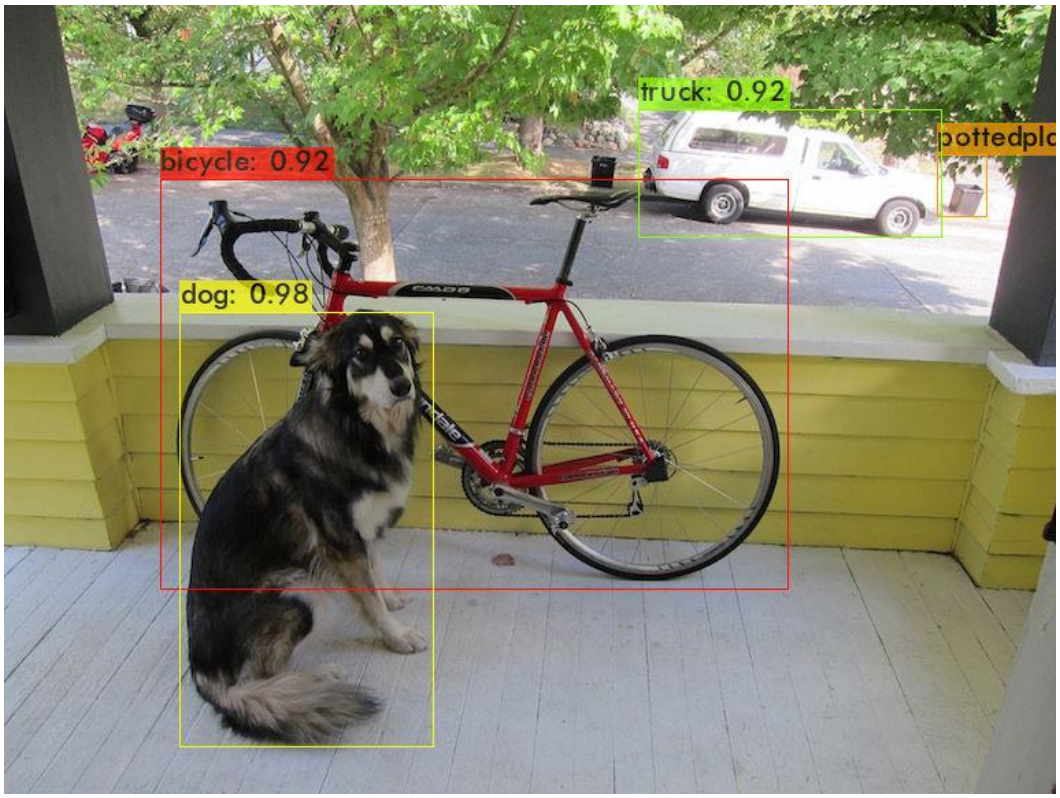
Resim 4 Test ve Eğitim Verilerinin Oluşturulması

5.2.4. Veri Setinin Google Colaba yüklenmesi

Bütün bu işlemlerin ardından Google Colab için açılan hesapta bilgisayarda oluşturulan görüntülerin, etiket, test, eğitim gibi dosyaların bulunduğu klasör Google Drive'a yüklenmiştir. Bu dosyalar Colab üzerinde yapılan işlemler ile Colab üzerinde entegre hale getirilmiştir.

5.2.5. Veri Setinin Eğitime hazırlanması

YOLO'nun kendi sitesinde bulunan katsayıları Colab üzerine indirilmiştir. Oluşturduğumuz veri setimizi eğitmeden önce internetten alınan örnek uygulamayla YOLO'nun çalışabilirliği test edilmiştir.



Resim 5 Örnek Uygulama Çıktısı

Resimde göründüğü gibi test başarı ile sonuçlanmıştır. Algoritma %98 doğru bir şekilde tanımıştır.

```
data/dog.jpg: Predicted in 78.254000 milli-seconds.
bicycle: 92%
dog: 98%
truck: 92%
pottedplant: 33%
```

Resim 6 Örnek Uygulamanın Güven Skoru ve Tespit Süresi

5.2.6. Veri Setinin Eğitilmesi

Bir önceki adımda örnek uygulama ile eğitilmişti. Eğitilen katsayılar 1000.weights dosyasına kaydedilmiştir. Eğitimler YOLOv3 için 11, YOLOv4 için 9, YOLOv3-tiny için 5, YOLOv4-tiny için 3 saat sürmüş ve 6000 iterasyonda sonlanmıştır. Algoritma her bin iterasyonda bin.weights dosyasına eğitim katsayılarını kaydetmiş, en iyi mAp değeri için best.weights son eğitim için last.weights dosyaların eğitim katsayılarını kaydetmiştir.

Eğitim sonunda elde edilen eğitim katsayılarıyla algoritmanın çalışıp çalışmadığı Spyder üzerinde resimlerden bir tanesi kullanılarak test edilmiş ve YOLO maske tespitini doğru bir şekilde yapmıştır.

YOLOnun tespit edilen maskeyi nasıl etiketlediği aşağıdaki görüntüde görünmektedir.

Resim 7 Tespit Çıktısı

6. **Yazılımın Raporu**

```
import easygui
import numpy as np
from time import time
import cv2
from datetime import datetime
from easygui import *
from MailGonder import MailGonder
```

Resim 8 Kütüphanelerin Yüklenmesi

Uygulamada kullanılacak kütüphanelerin yüklenmesi.

```
labels = ["Maskesiz", "Maskeli"]
renkler = ["0, 0, 255", "0, 255, 0"]
renkler = [np.array(renk.split(",")).astype("int") for renk in renkler]
```

Resim 9 Etiket ve Renklerin Belirlenmesi

Tespit edilen yüzün durumuna göre Bounding Boxa verilecek etiketin ismi ve etikete verilecek rengin belirlenmesi. Maskesiz için kırmızı, maskeli iyin yeşil.

```
text = "Model seçiniz.."
title = "Model Seç"
choices = ["YoloV3 mAp = %90", "YoloV3-Tiny mAp = %79", "YoloV4 mAp = %93", "YoloV4-Tiny mAp = %87", "Kapat"]
modelchoice = easygui.choicebox(text, title, choices) #Model seçim ekranının yapılması
```

Resim 10 Kullanıcının Model Tercih Arayüzü

Uygulamanın açılış arayüzünde yazacakların belirlenmesi. Burada kullanıcı tespit için kullanacağı modeli seçer. YoloV3, V3-Tiny, YoloV4, V4-Tiny den hangisini seçeceği bir choicebox çıkar.

```
if modelchoice == choices[0]: #Modelchoice ve choice değerlerine göre modelin seçilmesi
    print(modelchoice + " seçildi.")
    model = cv2.dnn.readNetFromDarknet("Model/yolov3.cfg", "Model/yolov3_last.weights")
elif modelchoice == choices[1]:
    print(modelchoice + " seçildi.")
    model = cv2.dnn.readNetFromDarknet("Model/yolov3-tiny.cfg", "Model/yolov3-tiny_last.weights")
elif modelchoice == choices[2]:
    print(modelchoice + " seçildi.")
    model = cv2.dnn.readNetFromDarknet("Model/yolov4.cfg", "Model/yolov4_last.weights")
elif modelchoice == choices[3]:
    print(modelchoice + " seçildi.")
    model = cv2.dnn.readNetFromDarknet("Model/yolov4-tiny-custom.cfg", "Model/yolov4-tiny-custom_last.weights")
else:
    print("Kapatılıyor...")
    break
```

Resim 11 Kullanıcının Seçtiği Modelin Sisteme Yüklenmesi

Seçtiği modele göre modelin cfg ve weights dosyalarının uygulamaya verilmesi.

```
model.setPreferableBackend(cv2.dnn.DNN_BACKEND_CUDA)
model.setPreferableTarget(cv2.dnn.DNN_TARGET_CUDA)
```

Resim 12 GPU Ayarı

Tespit için GPU kullanılacaksa CUDA çekirdeklerinin aktif edilmesi.

```
layers = model.getLayerNames()
layers = [layers[i[0] - 1] for i in model.getUnconnectedOutLayers()]
```

Resim 13 Model İçinden Katmanların Seçilmesi

```
text = "Hangisini kullanmak istersiniz?"
title = "Kullanım tercihi"
choices = ["Resim", "Video", "Kamera", "Model seçimine geri dön"]
rvkchoice = easygui.choicebox(text, title, choices)
```

Resim 14 Kullanıcının Görüntü Kaynağı Seçimi

Kullanıcının hangi görüntü kaynağını kullanacağını seçmesi. Resim, video ve kamera için bir choicebox çıkması.

```
if rvkchoice == choices[0]:
    easygui.msgbox(msg="Lütfen resim seçiniz..", title="Bilgi", ok_button="Tamam")
    path = easygui.fileopenbox(title="Resim seçiniz.", filetypes=["*.jpeg", "*.jpg", "*.png"])
    if not path:
        ynbox = easygui.ynbox("Devam etmek istiyor musunuz?", "", ("Evet","Hayır"))
        if ynbox == True:
            print("Devam ediliyor...")
            continue
        else:
            print("Kapatılıyor...")
            break
```

Resim 15 Resim Kaynağının Belirlenmesi

Görüntü kaynağı olarak resim belirlendiğinde resmin dosyası seçilmesi. Seçilmediği durumda yes-nobox ile devam durunun sorulması.

```
resim = cv2.imread(path)
width, height = 1080, 720
resize = (width, height)
resim = cv2.resize(resim, resize, interpolation=cv2.INTER_AREA)
resim_boy, resim_en = resim.shape[0], resim.shape[1]
```

Resim 16 Resmin Yeniden Boyutlandırılması

Resim görüntü kaynağının 1080x720 boyutlarına getirilerek yeniden boyutlandırılması.

```
elif rvkchoice == choices[1] or rvkchoice == choices[2]:
    if rvkchoice == choices[1]:
        easygui.msgbox(msg="Lütfen video seçiniz...", title="Bilgi", ok_button="Tamam")
        path = easygui.fileopenbox(title="Video seçiniz.", filetypes=["*.mp4", "*.avi"])
        if not path:
            ynbox = easygui.ynbox("Devam etmek istiyor musunuz?", "", ("Evet", "Hayır"))
            if ynbox == True:
                print("Devam ediliyor...")
                continue
            else:
                print("Kapatılıyor")
                break
        cap = cv2.VideoCapture(path)
    elif rvkchoice == choices[2]:
        cap = cv2.VideoCapture(0, cv2.CAP_DSHOW)
```

Resim 17 Video ya da Kameranın Kanyağının Belirlenmesi

Görüntü kaynağı olarak video ya da kamera seçildiğinde duruma göre kaynak seçimi. Video seçildiyse video dosyasının seçilmesi. Seçilmediği durumda yes-nobox ile devam durunun sorulması. Kamera seçildiyse otomatik olarak kameraya bağlanması.

```
width, height = 1080, 720
resize = (width, height)
frame = cv2.resize(frame, resize, interpolation=cv2.INTER_AREA)
frame_boy, frame_en = frame.shape[0], frame.shape[1]
```

Resim 18 Video ya da Kameranın Yeniden Boyutlandırılması

```
an = datetime.now()
kayitadi = "Output/Video/" + an.strftime('%Y-%m-%d-%H.%M.%S') + str("-result.avi")
out = cv2.VideoWriter(kayitadi, cv2.VideoWriter_fourcc('M', 'J', 'P', 'G'), 10, (width, height))
```

Resim 19 Video ya da Kamera Tespitinin Dosyaya Kaydedilmesi

Video ve kamera için tespitin video olarak kaydedilmesi amacıyla bir writer oluşturulması.

```
blob = cv2.dnn.blobFromImage(resim, 1 / 255.0, (832, 832), swapRB=True, crop=False)
model.setInput(blob)
baslangic = time()
layerOutputs = model.forward(layers) #
bitis = time()
```

Resim 20 Resmin Blob Formata Dönüştürülmesi ve Tespit Süresinin Öğrenilmesi

Görüntünün modele verilmeden önce blob formata dönüştürülmesi, modelin tespit etmesi, modelin tespit süresinin hesaplanması.

```
boxes = []
confidences = []
classIDs = []

for output in layerOutputs:
    for detection in output:
        scores = detection[5:]
        classID = np.argmax(scores)
        confidence = scores[classID]

        if confidence > 0.20:
            box = detection[0:4] * np.array([frame_en, frame_boy, frame_en, frame_boy])
            (box_mx, box_my, box_en, box_boy) = box.astype("int")

            bas_x = int(box_mx - (box_en / 2))
            bas_y = int(box_my - (box_boy / 2))

            boxes.append([bas_x, bas_y, int(box_en), int(box_boy)])
            confidences.append(float(confidence))
            classIDs.append(classID)
idxs = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)
```

Resim 21 Tespit İşlemlerinin Yapılması-1

```
if len(idxs) > 0:
    for i in idxs.flatten():
        (x, y) = (boxes[i][0], boxes[i][1])
        (w, h) = (boxes[i][2], boxes[i][3])

        renk = [int(c) for c in renkler[classIDs[i]]]
        cv2.rectangle(frame, (x, y), (x + w, y + h), renk, 1)
        text = "{}: {:.4f}".format(labels[classIDs[i]], confidences[i])
        cv2.putText(frame, text, (x, y - 5), cv2.FONT_HERSHEY_SIMPLEX, 0.5, renk, 1)
```

Resim 22 Tespit İşlemlerinin Yapılması-2

```
out.write(frame)
# && # Alta gelecek sayaçlar için 100px boşluk olutşurulması
frame = cv2.copyMakeBorder(frame, 0, 100, 0, 0, cv2.BORDER_CONSTANT)
# Ekrana basılacak maskeli, maskesiz, toplam sayılarının bulunması
filtered_classids = np.take(classIDs, idxs)
maskesiz = (filtered_classids == 0).sum()
maskeli = (filtered_classids == 1).sum()
toplam = maskeli + maskesiz
```

Resim 23 Sayaç için Maskeli Maskesiz Sayısının Tespiti

Pencerenin altında sayaçlar için 100px bir boşluk oluşturulması. Boşlukta kullanılacak veriler için maskeli ve maskesiz sayılarının alınması.

```
# Sadece maskesiz olanların sayısının ekrana basılması
text = "Maskesiz: {}".format(maskesiz)
cv2.putText(frame, text, (0, 750), cv2.FONT_HERSHEY_SIMPLEX, 0.8, [0, 0, 255], 2)

# Sadece maskeli olanların sayısının ekrana basılması
text = "Maskeli: {}".format(maskeli)
cv2.putText(frame, text, (175, 750), cv2.FONT_HERSHEY_SIMPLEX, 0.8, [0, 255, 0], 2)

# Tespit edilen tüm objelerin ekrana basılması
text = "Toplam: {}".format(toplam)
cv2.putText(frame, text, (325, 750), cv2.FONT_HERSHEY_SIMPLEX, 0.8, [255, 255, 255], 2)

# YOLO nun tespit etme süresinin ekrana basılması
text = "Tespit Suresi {:.3f} MS".format(bitis - baslangic)
cv2.putText(frame, text, (0, 800), cv2.FONT_HERSHEY_SIMPLEX, 0.8, [255, 255, 255], 2)

# FPS
text = "FPS: {:.1f}".format(fps)
cv2.putText(frame, text, (400, 800), cv2.FONT_HERSHEY_SIMPLEX, 0.8, [255, 255, 255], 2)

# Maskeli-Maskesiz oranı
if maskeli == 0 and toplam == 0:
    oran = 0
else:
    oran = (maskeli / toplam) * 100
text = "Oran: "
cv2.putText(frame, text, (800, 750), cv2.FONT_HERSHEY_SIMPLEX, 0.8, [255, 255, 255], 2)
text = "{:.0f}%".format(oran)
cv2.putText(frame, text, (900, 750), cv2.FONT_HERSHEY_SIMPLEX, 0.8, [255, 255, 255], 2)

# Maskeli-Maskesiz sayısına göre durumun belirlenmesi
text = "Durum: "
cv2.putText(frame, text, (800, 800), cv2.FONT_HERSHEY_SIMPLEX, 0.8, [255, 255, 255], 2)
```

Resim 24 Yapılan Tespitle İlgili Sayaçların Bilgileri

Sayaçta yazacak maskeli, maskesiz, toplam, tespit süresi, FPS, durum, oran gibi verilerin ekrana yazdırılması.

```
# Maskeli-Maskesiz sayısına göre durumun belirlenmesi
text = "Durum: "
cv2.putText(frame, text, (800, 800), cv2.FONT_HERSHEY_SIMPLEX, 0.8, [255, 255, 255], 2)

if oran >= 0.1 and maskesiz >= 3:
    text = "Tehlikeli"
    cv2.putText(frame, text, (900, 800), cv2.FONT_HERSHEY_SIMPLEX, 0.8, [0, 0, 255], 2)

elif oran != 0 and oran >= 50:
    text = "Riskli"
    cv2.putText(frame, text, (900, 800), cv2.FONT_HERSHEY_SIMPLEX, 0.8, [0, 255, 255], 2)

else:
    text = "Guvenli"
    cv2.putText(frame, text, (900, 800), cv2.FONT_HERSHEY_SIMPLEX, 0.8, [0, 255, 0], 2)
```

Resim 25 Sayaçtaki Verilere Göre Ortamın Durumunun Belirlenmesi

Sayaçtan alınan verilere göre ortamdaki durumun tehlikeli, riskli ve güvenli olarak belirlenmesi.

```
if oran >= 0.1 and maskesiz >= 5:
    text = "Tehlikeli"
    cv2.putText(frame, text, (900, 800), cv2.FONT_HERSHEY_SIMPLEX, 0.8, [0, 0, 255], 2)
    if rvkchoice == choices[2]:
        mesaj = "Bilgilendirme \n"
        mesaj+= "Durum: {} \n".format(text)
        mesaj+= "Maskeli: {} \n".format(maskeli)
        mesaj+= "Maskesiz: {} \n".format(maskesiz)
        mesaj+="Bilgilendirmenin zamani: "+an.strftime('%Y-%m-%d %H:%M:%S %Z')
        MailGonder(mesaj)
```

Resim 26 E-posta Bildirimi

Koşul durumu sağlandığında yollanacak mailin içeriğindeki bilgilerin alınması. Ve E-posta fonksiyonuna yollanıp mailin gönderilmesi.

```
cv2.imshow("frame", frame)
key = cv2.waitKey(1)
if key == ord("s"):
    an = datetime.now()
    filename = "Output/Image/" + an.strftime('%Y-%m-%d-%H.%M.%S') + str("-result.jpg")
    path = easygui.filesavebox(title="Kaydet", default=filename)
    cv2.imwrite(path, frame)
    print("Kaydedildi..")
    cap.release()
    out.release()
    cv2.destroyAllWindows()
    break
elif key == 27:
    print("Kapatılıyor..")
    cap.release()
    out.release()
    cv2.destroyAllWindows()
    break
```

Resim 27 Anlık Karenin Kaydı ve Uygulamanın Kapatılması

S tuşuna basıldığında o anki karenin kaydedilmesi ve ESC tuşuyla uygulamanın kapanması.

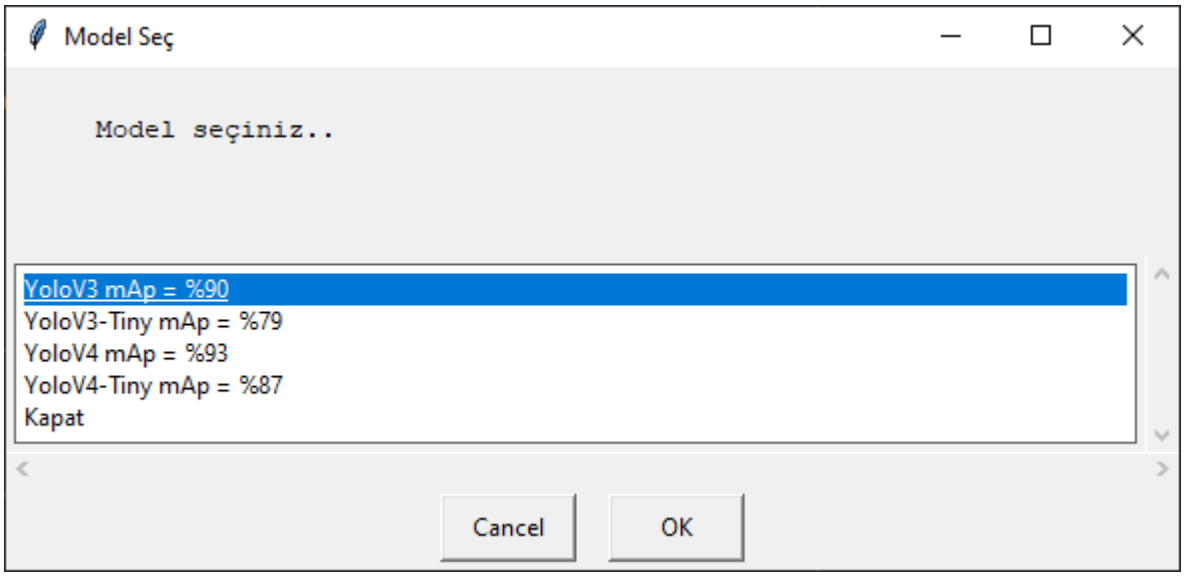
```
import smtplib

def MailGonder(mesaj):
    mailicerigi = mesaj
    mail = smtplib.SMTP("smtp.gmail.com",587)
    mail.ehlo()
    mail.starttls()
    mail.login("gönderenmail","gönderenmailşifre")
    mail.sendmail("gönderenmail","alıcımail",mailicerigi)
```

Resim 28 E-Posta Dosyasının Ayarlanması

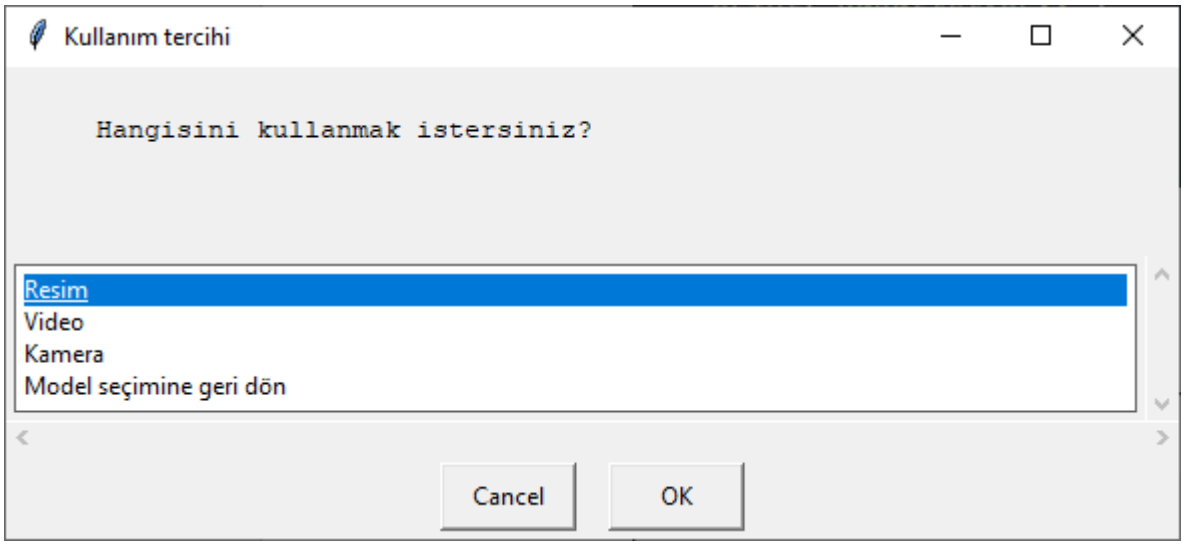
E-posta gönderme işlemi için gerekli işlemlerin yapılması. Kütüphanenin yüklenmesi, ayarların girilmesi, hesap bilgilerinin girilmesi.

7. Kullanım Kılavuzu

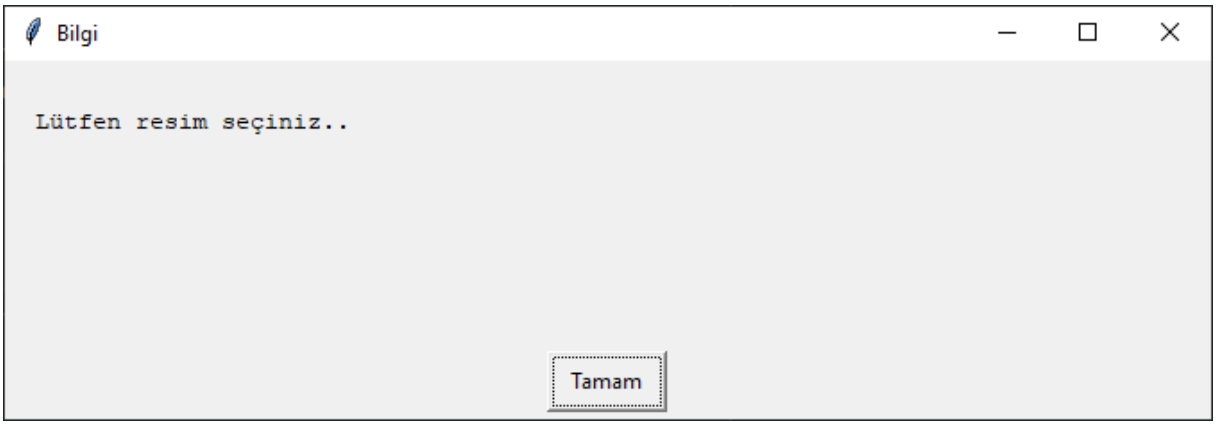


Resim 29 Uygulamanın Model Seçim Ekranı

Bu ekranda uygulamada kullanılacak modeli seçer. Modelin mAp değeri ne kadar yüksekse o kadar tutarlı tespit yapar.

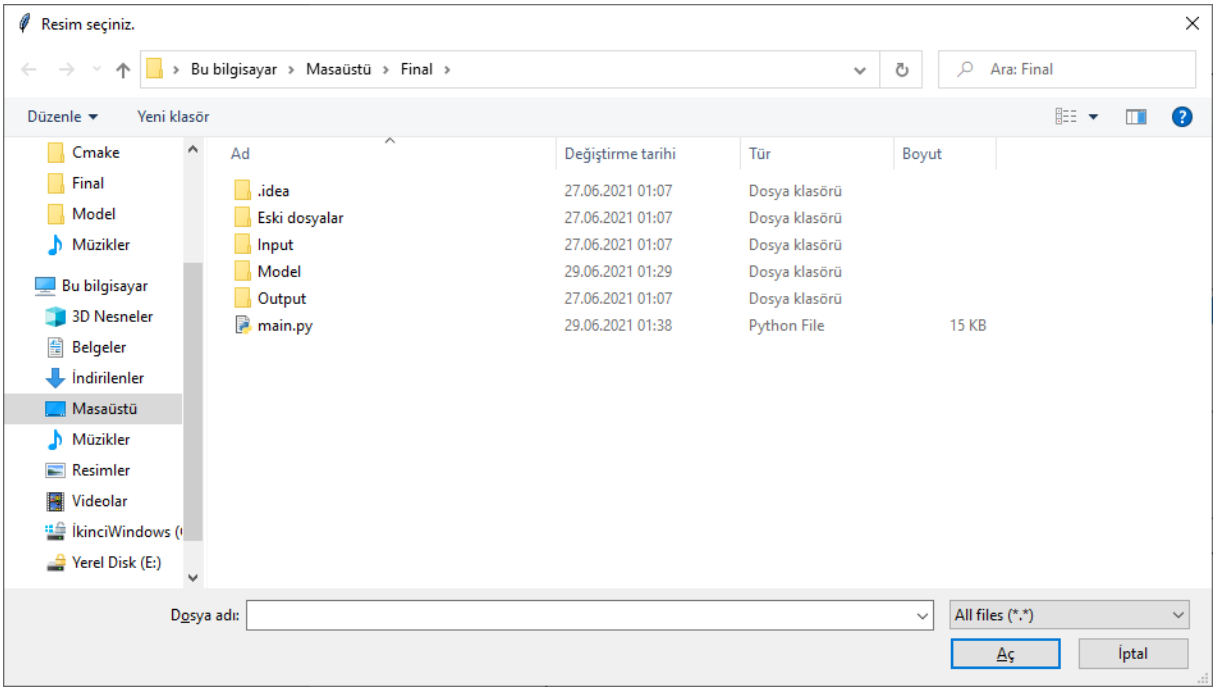


Resim 30 Hangi Görüntü Kaynağını Kullanacağının Seçilmesi



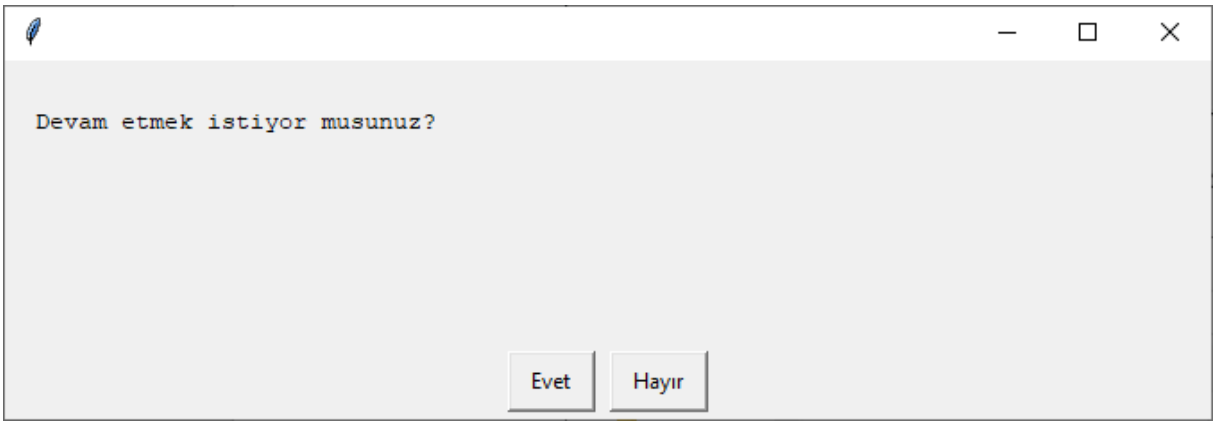
Resim 31 Resim ve Video İçin Uyarı Ekranı

Kullanıcı Resim ve Video için görüntü kaynağı seçmeden önce ufak bir uyarı.



Resim 32 Kaynak Dosyasının Seçilmesi

Resim ve video için kullanılacak görüntü dosyasının seçilmesi.



Resim 33 Kaynak Seçilmediği Durum

Kaynak seçilmediği durumda kullanıcıya devam edip etmeyeceğinin sorulması.

Resim 34 Sonuç

Tespit işleminin yapılıp sonucunda ekrana verilmesi.

8. Sonuç

Çalışmada insanların yüz görüntüsünden maske takılıp takılmadığını tespit etmek amacıyla derin öğrenme yöntemlerinden Konvolüsyonel Sinir Ağı kullanılmıştır. Model eğitim ve test veri seti ile eğitildikten sonra doğrulama veri kümesine göre doğruluk elde edilmiştir. Elde edilen doğruluk oranı YOLOv3 için %90, YOLOv3-tiny için %79, YOLOv4 için %93, YOLOv4-tiny için %87 olarak tespit edilmiştir.

YOLO v3 ve v4 modelleri tiny modellere göre daha iyi sistem gereksinimleri istemekte ve daha yavaş çalışmaktadır. V3 ve V4 modeller tiny modellere göre daha doğru sonuçlar vermekte ama bu sonuçlar görüntüden görüntüye, ortamdaki ışık miktarına, gece ya da gündüz olmasına göre farklılıklar göstermektedir. Tespit etmek için geçen tarama süresi arttırıldığında modelin algılama olasılığı düşmektedir.

Elde edilen sonuçlarda görünmektedir ki YOLO yeterli ortam koşulları sağlandığında nesne tespit performansında yüksek başarı elde etmektedir. Bu sonuçlara göre modelin maskeli ve maskesiz sınıflandırma için başarılı olduğu görülmektedir.

KAYNAKÇA VE ERİŞİM

[1] M. R. Bhuiyan, S. A. Khushbu, and M. S. Islam, “A Deep Learning Based Assistive System to Classify COVID-19 Face Mask for Human Safety with YOLOv3,” pp. 1–5, 2020, doi: 10.1109/icccnt49239.2020.9225384

[2] S. V. Militante and N. V. Dionisio, “Real-Time Facemask Recognition with Alarm System using Deep Learning,” no. August, pp. 106–110, 2020, doi: 10.1109/icsgrc49013.2020.9232610.

[3] Ö. Algur , V. Tümen and Ö. Yıldırım , "Dış Ortam Görüntülerindeki İnsan Hareketlerinin Hibrit Derin Öğrenme Yöntemleri Kullanarak Sınıflandırılması", Fırat Üniversitesi Mühendislik Bilimleri Dergisi, vol. 30, no. 3, pp. 121-129, Sep. 2018

[4] B. Kılıç , E. Baykal Kablan , H. Doğan , M. Ekinci , M. Ercin ve Ş. Ersöz , "Derin Konvolüsyonel Nesne Algılayıcı ile Plevral Efüzyon Sitopatolojisinde Otomatik Çekirdek Algılama", Türkiye Bilişim Vakfı Bilgisayar Bilimleri ve Mühendisliği Dergisi, c. 13, sayı. 1, ss. 33-42, Nis. 2020

[5] Shinde, S., Kothari, A., Gupta, V., (2018), “YOLO based Human Action Recognition and Localization”, Makale, Procedia Computer Science, Cilt 133, Sayfa 831-838, Hindistan

Erişim için:

https://github.com/pauybs/2021-maske-algilama

https://github.com/hheren/2021-maske-algilama