



**T. C.
HALIÇ ÜNİVERSİTESİ
MESLEK YÜKSEKOKULU
BİLGİSAYAR PROGRAMCILIĞI PROGRAMI**

EVRIŞİMSEL SİNİR AĞLARI İLE MASKE TAKİP SİSTEMİ

Ön Lisans Tezi

**Hazırlayan
Muratcan LALOĞLU**

**Danışman
Öğr. Gör. Zeynep İNEL ÖZKİPER**

İstanbul, 2022

**T. C.
HALIÇ ÜNİVERSİTESİ
MESLEK YÜKSEKOKULU
BİLGİSAYAR PROGRAMCILIĞI PROGRAMI**

EVRIŞİMSEL SİNİR AĞLARI İLE MASKE TAKİP SİSTEMİ

Ön Lisans Tezi

**Hazırlayan
Muratcan LALOĞLU**

**Danışman
Öğr. Gör. Zeynep İNEL ÖZKİPER**

İstanbul, 2022

ÖN SÖZ

Bitirme projesi kapsamında fikir ve görüşleri ile yol gösteren danışman hocam Sayın Öğr. Gör. Zeynep İNEL ÖZKİPER hocama, verdikleri online eğitim ile bana büyük bilgi birikimi katan DATAI TEAM'a teşekkürü borç bilirim.

İstanbul, 2022

Muratcan LALOĞLU

İÇİNDEKİLER

KISALTMALAR	3
ŞEKİLLER LİSTESİ.....	4
ÖZET	6
SUMMARY.....	7
1. GİRİŞ	8
1.1. COVID-19.....	8
1.1.2 Bulaşma Yolları ve Alınan Önlemler.....	9
1.2 Python Programlama Dili.....	10
1.2.2 Kullanım Alanları.....	11
1.2.3 Söz Dizimi.....	11
1.2.4 Popülerlik	12
1.3 Yapay Zeka, Makine Öğrenmesi Ve Derin Öğrenme.....	13
1.3.1 Yapay Zeka	13
1.3.2 Makine Öğrenmesi	13
1.3.2.1 Denetimli, Denetimsiz ve Pekiştirmeli Öğrenme	14
1.3.3 Derin Öğrenme.....	14
1.4 Literatür Taraması.....	15
1.4.1 COVID-19 Denetiminde Yapay Zeka Çalışmaları	15
1.5 Evrişimsel Sinir Ağları.....	16
1.5.1 Evrişimsel Sinir Ağları Nasıl Çalışır?.....	17
1.5.1.1 Convolutional Layer (Evrişim Katmanı)	17
1.5.1.2 Havuzlama Katmanı (Pooling Layer)	19
1.5.1.3 Düzleştirme Katmanı (Flattening Layer)	19
1.5.1.4 Yoğunlaştırma Katmanı(Dense Layer)	21
1.5.1.5 Bırakma Katmanı(Dropout Layer).....	21
2. GEREÇLER.....	22
2.1 Veri Seti	22
2.2 Pretrained Face Detector (Önceden Eğitilmiş Yüz Dedektörü).....	22

2.3 Spyder IDE (Spyder Geliştirme Ortamı)	22
2.4 Kullanılan Kütüphaneler	23
2.4.1 Tensorflow	23
2.4.2 Keras	23
2.4.3 NumPy	23
2.4.4 Sklearn	24
2.4.5 OS.....	24
2.4.6 Matplotlib.....	24
2.4.7 Imutils	24
2.4.8 Time	24
2.4.9 OpenCv	24
3. YÖNTEMLER	26
3.1 Modelin Eğitilmesi.....	26
3.2 Modelin Görselleştirilip Kameraya Aktarılması.....	31
4. SONUÇ	36
4.1 Model Performansı.....	36
4.2 Görsel Sonuçlar	36
5. TARTIŞMA.....	37
KAYNAKLAR	38
ÖZGEÇMİŞ	42

KISALTMALAR

BGR: Blue Green Red(Mavi Yeşil Kırmızı)

BLOB: Binary Large Object(İkili Büyük Nesne)

CNN: Convolutional Neural Network (Evrişimsel Sinir Ağı)

COVID-19: Koronavirüs hastalığı

RGB: Red Green Blue(Kırmızı Yeşil Mavi)

SARS-CoV-2: Şiddetli akut solunum yolu sendromu koronavirüsü 2

ŞEKİLLER LİSTESİ

Şekil 1.1: Koronavirüsün görüntüsü (Wikipedia, 2020).	9
Şekil 1.2: Örnek CNN yapısı(researchgate.net, 2022).	16
Şekil 1.3: Örnek Evrişim Katmanı Şeması(medium.com, 2022).	17
Şekil 1.4: Somurta – Gülen Yüz (medium.com, 2022).	18
Şekil 1.5: Örnek Pooling Layer Şeması (medium.com, 2022).	19
Şekil 1.6: Örnek Flattening Şeması (medium.com, 2022).	20
Şekil 1.7: Basic Connected Layer (medium.com, 2022).	20
Şekil 1.8: Dense Layer(medium.com, 2022).	21
Şekil 1.9: Dropout Layer(medium.com, 2022).	21
Şekil 2.1: Veri setinin önizlemesi	22
Şekil 3.1: Kütüphane Kodları.	26
Şekil 3.2: Başlangıç öğrenme oranı, devir, aynı anda eğitilecek veri miktarı.	26
Şekil 3.3: Veri setindeki görüntülerin listeye alınması, kategori listesinin oluşturulması.	26
Verilerimiz işlemek için iki adet boş dizi oluşturulması(Şekil 3.4).	27
Şekil 3.4: Verilerin işlenmesi için 2 boş liste oluşturulması.	27
Şekil 3.5: Resimlerin yeniden boyutlandırıp array veri tipine dönüştüren döngü.	27
Şekil 3.6: Etiketlerin(maskeli, maskesiz) 0 ve 1'e dönüştürülmesi.	27
Oluşturduğumuz data ve labels dizilerinin NumPy dizisine dönüştürülmesi(Şekil 3.7).	27
Şekil 3.7: Resimlerin ve Etiketlerin NumPy array'a dönüştürülmesi.	27
Şekil 3.8: Verilerin %75'i eğitim için ve %25'i test için ayrılması.	28
Şekil 3.9: Veriyi çoğaltmak için generator oluşturulması.	28
Şekil 3.10: MobileNetV2 Ağının bağlanması.	28
Sinir ağı katmanlarımızın oluşturulması(Şekil 3.11).	28
Şekil 3.11: Sinir ağının oluşturulması.	28
Şekil 3.12: Oluşturduğumuz sinir ağının MobileNetV2'ye bağlanması.	29
Şekil 3.13: MobileNetV2 ağının eğitilmemesi içine yazılan döngü.	29
Şekil 3.14: Modelin Derlenmesi.	29
Şekil 3.15: Ağın başının eğitilmesi.	29
Şekil 3.16: Test setinde tahminlerin alınması.	30
Şekil 3.17: Test setindeki her bir görüntü için karşılık gelen en büyük tahmin edilen olasılığa sahip etiket dizinin bulunması.	30
Sklern kütüphanesi ile model raporunun oluşturulması(Şekil 3.18).	30
Şekil 3.18: Model raporunun oluşturulması.	30
Şekil 3.19: Model raporu sonuçları.	30
Şekil 3.20: Modelin Kaydedilmesi.	30

Şekil 3.21: Kütüphanelerin aktarılması.	31
Şekil 3.22: Çerçeve boyutlarının yakalama ve BLOB oluşturulması.	31
Şekil 3.23: Ağ üzerindeki yüz algılamalarını alan kod.	31
Şekil 3.24: Yüz listemizi, bunlara karşılık gelen konumları ve yüz maskesi ağıımızdaki tahminlerin listesinin oluşturulması.	31
Şekil 3.25: Yüz algılama kontrolü için döngü oluşturulması.	32
Şekil 3.26: Zayıf algılanmaların filtrelenmesi için yazılan koşul.	32
Tespit edilen yüzün koordinatlarının ilgili değişkenlere atanması(Şekil 3.27).	32
Şekil 3.27: Tespit edilen yüzün çerçevesinin koordinatlarının hesaplanması.	32
Şekil 3.28: Tespit edilen yüzün renk skalasının BGR'dan RGB'ye çevrilmesi, yeniden boyutlandırılması ve diziye dönüştürülmesi.	32
Şekil 3.29: Tespit edilen yüzlerin ve çerçeve koordinatlarının ilgili listeye eklenmesi.	33
Şekil 3.30: En az 1 yüz bulunduğunda tarama yapma ve konumlarını döndürme kodu.	33
Şekil 3.31: Önceden eğitilmiş yüz detektörü modelimizin yüklenmesi.	33
Değişken oluşturulup eğittiğimiz maske tespit modelinin yüklenmesi(Şekil 3.32).	33
Şekil 3.32: Eğittiğimiz maske tespit modelinin yüklenmesi.	33
Şekil 3.33: Kameranın açılması.	34
Video akışını başlatan döngünün oluşturulması, Kamera çözünürlüğünün belirlenmesi ve tespit edilen yüzlerin maske takıp takmadığının belirlenmesi(Şekil 3.34).	34
Şekil 3.34: Video akışının başlatılması, video çerçevesinin boyutunun ayarlanması, tespit edilen yüzlerin maske takıp takmadığının belirlenmesi.	34
Şekil 3.35: Algılanan yüz konumları ve bunlara karşılık gelen konumlar üzerinde döngü oluşturulması.	34
Yapılan tahminde maskeli değeri maskesiz değerinden büyükse etikete(label) "Mask" değilse "No Mask" yapan döngü, etiket değeri "Mask" ise rengi(color) yeşile değilse kırmızıya dönüştüren döngü(Şekil 3.36).	34
Şekil 3.36: Yapılan tespit Maskeliyse çerçeveyi yeşile, değilse kırmızıya boyayan kod.	34
Şekil 3.37: Çerçevenin ve maske etiketinin oluşturulması.	35
Şekil 3.38: Kameradan gelen görüntünün yansıtılması ve q tuşuna basıldığında döngünün sonra erdirilmesi.	35
Şekil 3.39: Programın sonlandırılması.	35
Şekil 3.40: Maskesiz Görünüm.	36
Şekil 3.41: Maskeli Görünüm.	36

GENEL BİLGİLER

Adı-Soyadı: Muratcan LALOĞLU

Programı: Bilgisayar Programcılığı

Tez Danışmanı: Öğr. Gör. Zeynep İNEL ÖZKİRPER

Tez Türü ve Tarihi: Ön Lisans-Haziran 2022

ÖZET

COVID-19 Karantinalarının sonra ermesiyle birlikte hükümet ve halk sağlığı kurumları koronavirüsün yayılmasını engellemek ve böylece halk sağlığına katkıda bulunmak ve bizi güvende tutmak için gerekli önlemler olarak maske kullanımını önermektedir. Tıbbi kaynaklar ve maskelerdeki farklılıklar konusundaki söylemlerden bağımsız olarak, bazı ülkeler kamuoyunda maske kullanımını zorunlu kılmaktadır.

Maskelerin kullanımını zorunlu kılmak için, bireyleri maske kullanmaya zorlayan bazı teknikler geliştirmek esastır. Bu uygulama havaalanları, toplu taşıma istasyonları, alışveriş merkezleri vb. ortak alanlarda çok yararlı olabilir. Burada kullanılan yöntem iki adımda gerçekleştirilmektedir. İlk adım, Python programlama dili kullanılarak derin öğrenme algoritmaları ile maske dedektörü modeli eğitmektir. İkinci adım, maske takıp takmadıklarını belirlemek için eğitilen modeli video, resim ve mobese kameralarında kullanmaktır.

Anahtar Kelimeler: Maske, COVID-19, Derin Öğrenme, Görüntü İşleme

GENERAL INFORMATION

Name-Surname: Muratcan Laloğlu

Program: Computer Programming

Thesis Supervisor: Öğr. Gör. Zeynep İNEL ÖZKİRPER

Thesis Type and Date: Associate Degree-June 2022

SUMMARY

With the reopening of countries from COVID-19 lockdown, Government and Public health agencies are recommending face masks as essential measures to keep us safe when venturing into public to curtail the spread of Coronavirus and thereby contributing to public healthcare. Regardless of discourse on medical resources and diversities in masks, all countries are mandating coverings over the nose and mouth in public.

To mandate the use of facemasks, it becomes essential to devise some techniques that enforce individuals to apply a mask before exposure to public places. This application can be very useful in public areas such as airports, railway stations, crowded markets, malls, etc. The proposed method used here is carried out in two steps. The first step is to train the face mask detector using deep learning with Python programming language. The second step is to use this trained face mask detector on images or videos of people to identify if they are wearing a mask.

Keywords: Mask, COVID-19, Deep Learning, Image Processing

1. GİRİŞ

1.1. COVID-19

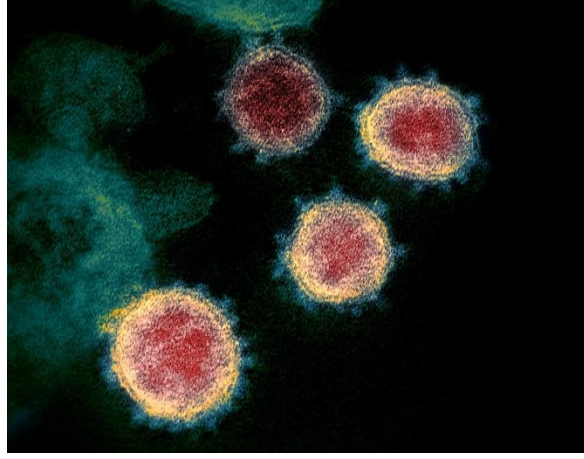
Koronavirüs hastalığı 2019 (COVID-19) şiddetli akut solunum sendromu koronavirüsü SARS-CoV-2'nin neden olduğu bulaşıcı bir hastalıktır. İlk vaka ile Çin'in Hubei eyaletinin Wuhan şehrinde Kasım 2019 tarihinde karşılaşılmıştır. O zamandan bu yana yayılmaya devam etmiş ve hala daha devam eden bir pandemiye neden olmuştur. 3 Mart 2020 itibarıyla dünya çapında ölüm oranı %3,4 olup, 20 Mayıs 2022 tarihi itibarıyla Dünya'da 525.908.358 onaylanmış vaka, 495.716.104 iyileşen varken virüs nedeniyle 6.297.130 hasta öldü.

COVID-19'un semptomları, hafif semptomlardan ağır hastalığa kadar değişkenlik göstermektedir. Yaygın semptomları arasında baş ağrısı, koku ve tat kaybı, burun tıkanıklığı ve burun akıntısı, öksürük, kas ağrısı, boğaz ağrısı, ateş ve nefes alma güçlüğü yer alır. Aynı enfeksiyona sahip kişilerde farklı semptomlar görülebilmektedir ve bu semptomlar zamanla değişiklik gösterebilir. Daha önceden kulak, burun, boğaz rahatsızlığı olmayan kişilerde koku kaybı ile tat kaybının aynı anda görülmesi % 95 özgüllük oranıyla COVID-19 ile ilişkilendirilir.

Çoğu hasta (%81) hafif ile orta şiddette semptomlar gösterirken (hafif derecede zatüreye kadar), bir kısmı (%14) ağır semptomlar göstermekte (nefes darlığı, hipoksi, %50'den fazla akciğer tutulumu gibi), az bir kısmı ise (%5) kritik semptomlar göstermektedir (solunum yetmezliği, şok, çoklu organ disfonksiyonu gibi). Enfekte kişilerden ortalama olarak beşte biri hiçbir semptom göstermemektedir. Asemptomik taşıyıcılar genellikle test olmamaktadır ve hastalığı bulaştırabilirler. Bazı enfekte kişiler daha sonradan semptom göstermeye başlayabilir veya çok hafif semptomlar gösterebilirler, bu kişilerde hastalığı bulaştırabilmektedir.

Çoğu enfeksiyonda olduğu gibi enfekte olunduktan sonra ilk semptomların ortaya çıkması için kuluçka süresi olarak bilinen belirli bir sürenin geçmesi gerekir. Covid-19'un kuluçka süresi ortalama dört ila beş gündür. Çoğu semptomik kişi temastan iki ila yedi gün arasında semptom göstermeye başlar ve neredeyse

septomik hastaların tamamı on iki gün dolmadan bir veya birden fazla semptom gösterir (Wikipedia, 2020).



Şekil 1.1: Koronavirüsün görüntüsü (Wikipedia, 2020).

1.1.2 Bulaşma Yolları ve Alınan Önlemler

COVID-19 genellikle, solunum yoluyla bulaşmaktadır; enfekte kimsenin öksürmesi, hapşırması, konuşması veya nefes alması ile bulaşabilir. Enfeksiyona virüs içeren partiküllerin solunulması, virüs içeren solunum damlacıklarının veya aerosollerinin, enfekte olan hasta ile yakın temasta bulunan kişilerin ağız, burun veya gözlerine girmesi neden olur. Bulaşıcı olan ortalama 1000 SARS-CoV-2 viryonunun insandan insana geçebildiği ve enfeksiyona neden olduğu düşünülmektedir. Temas süresi arttıkça ve bireyler arasındaki fiziksel mesafe azaldıkça COVID-19'un bulaşma riski artmaktadır. Yakın mesafeler yere düşen daha büyük damlacıkları ve aerosolleri içerebilirken, daha uzun mesafeler yalnızca aerosolleri içerir. Daha büyük damlacıklar da buharlaşarak aerosollere dönüşebilir. Kasım 2020 itibarıyla, büyük damlacıklar ile aerosoller arasındaki önem farkı net değildir, ayrıca virüsün bir odadan uzak bir mesafedeki başka bir odaya hava kanalları gibi yollarla taşınabilme ihtimalinin olup olmadığı da bilinmemektedir. Hava yoluyla bulaşma özellikle; restoranlar, korolar, spor salonları, gece kulüpleri, ofisler ve ibadethaneler gibi yüksek risk içeren kapalı alanların az havalandırılmış ve kalabalık olması durumunda gerçekleşebilmektedir.

Ayrıca, sağlık hizmeti ortamlarında da hava yoluyla bulaşma durumu COVID-19 hastalarına aerosol oluşturan tıbbi prosedürler uygulanması sonucunda yaşanabilmektedir.

Enfeksiyon olasılığını azaltmaya yönelik önleyici tedbirler arasında evde kalmak, halka açık yerlerde maske takmak, kalabalık yerlerden kaçınmak, insanlardan uzak durmak, kapalı alanları havalandırmak, elleri sık sık sabun ve suyla birlikte en az 20 saniye boyunca yıkamak, solunum sistemi hijyenine dikkat etmek; yıkanmamış ellerle yüz bölgesine dokunmaktan kaçınmak yer alır. Covid-19 tanısı konmuş kişilerin veya bu hastalığı geçirdiğinden şüphelenen kimselerin tıbbi yardım almak dışında evden ayrılmamaları, başka birisiyle aynı ortamda bulunuyorlarsa öksürme ve hapşırma durumlarında ağızlarını kapatmaları, kişisel ev eşyalarını paylaşmamaları ve bol bol ellerini yıkamaları önerilmektedir.

Koruyucu önlemler, aşı veya daha iyi tedavi yöntemleri geliştirilene kadar hastanelerdeki yükün azaltılması amacıyla uygulanmaktadır.

COVID-19 önüne geçilmesi adına hayati rol taşıyan diğer bir uygulama ise aşılama'dır. Bilindiği üzere aşılar her yıl milyonlarca hayat kurtarmaktadır. Aşılar sayesinde vücut virüslere ve bakterilere karşı doğal bağışıklık sisteminin hazırlanmasını sağlar. Sonrasında bağışıklık sistemi vücudun maruz kaldığı virüs veya bakteriyi yok etmeye hazır hale gelir(Vikipedia, 2022).

1.2 PYTHON PROGRAMLAMA DİLİ

Python, nesne yönelimli, yorumlamalı, birimsel (modüler) ve etkileşimli yüksek seviyeli bir programlama dilidir. Girintilere dayalı basit söz dizimi, dilin öğrenilmesini ve akılda kalmasını kolaylaştırır. Bu da ona söz diziminin ayrıntıları ile vakit yitirmeden programlama yapılmaya başlanabilen bir dil olma özelliği kazandırır. Modüler yapısı, sınıf dizgesini (sistem) ve her türlü veri alanı girişini destekler. Hemen hemen her türlü platformda çalışabilir (Unix, Linux, Mac, Windows, Amiga, Symbian). Python ile sistem programlama, kullanıcı arabirimi programlama, ağ programlama, web programlama, uygulama ve veri tabanı yazılımı programlama gibi birçok alanda yazılım geliştirebilirsiniz. Büyük yazılımların hızlı bir şekilde

prototiplerinin üretilmesi ve denenmesi gerektiği durumlarda da C ya da C++ gibi dillere tercih edilir.

Python 1980'lerin sonunda ABC programlama diline alternatif olarak tasarlanmıştır. Python 2.0, ilk kez 2000 yılında yayınlandı. 2008'de yayınlanan Python 3.0, dilin önceki versiyonuyla tam uyumlu değildir ve Python 2.x'te yazılan kodların Python 3.x'te çalışması için değiştirilmesi gerekmektedir. Python 2 versiyonun resmi geliştirilme süreci, dilin son sürümü olan Python 2.7.x serisi versiyonların ardından 1 Ocak 2020 itibarıyla resmi olarak sona erdi. Python 2.x geliştirilme desteğinin sona ermesinin ardından, Python dilinin 3.6.x ve sonraki sürümlerinin geliştirilmesi devam etmektedir(Vikipedia, 2022).

1.2.2 Kullanım Alanları

Django, Zope uygulama sunucuları, YouTube ve orijinal BitTorrent istemcisi Python kullanan önemli projelerden bazılarıdır. Ayrıca Google, NASA ve CERN gibi büyük kurumlar da Python kullanmaktadır. Pygame ile 2D oyun yapılabilir, Blockchain uygulamaları kodlanabilir, uzaktan kontrol veya görüntü işleme yapılabilir, veri analizi veya veri kontrolü yapılabilir, TensorFlow, PyTorch, Keras gibi kütüphanelerle derin makine öğrenmesi uygulamaları yapılabilir. Ayrıca OpenOffice.org, GIMP, Inkscape, Blender, Scribus ve Paint Shop Pro gibi bazı programlarda betik dili olarak kullanılır. Pek çok Linux dağıtımında ve Apple macOS işletim sisteminde Python öntanımlı bir bileşen olarak gelir(Vikipedia, 2022).

1.2.3 Söz Dizimi

Python'un son derece kolay okunabilir olması düşünülmüştür. Bu yüzden örneğin küme parantezleri yerine girintileme işlemi kullanılır. Hatta bazı durumlarda girintileme işlemine dahi gerek kalmadan kodun ilgili bölümü tek satırda yazılabilir. Böylece Python, program kodunuzu en az çaba ile ve hızlıca yazmanıza imkân tanır. Sade sözdizimi ile diğer programlama dillerinden üstündür(Vikipedia, 2022).

Diğer diller ile karşılaştırıldığında Python'da yer alan bütün modüller ve kütüphaneler birer nesne olarak görev yapar. Bunun sayesinde Python, etkili bir kod derleyici olarak ön plana çıkmaktadır. Kendi kod özellikleri ile yazılan uygulamaları manipüle ederek,

diğer dillerde yazılması çok zor olan yada neredeyse imkansız olan uygulamaları oldukça kolay yazılabilir hale getirir.

1.2.4 Popülerlik

1991'den beri Python programlama dili sadece gereksiz programlar için tamamlayıcı bir dil olarak değerlendiriliyordu. Hatta “Automate the Boring Stuff” (Türkçe'ye "Sıkıcı Şeyleri Otomatikleştiren" olarak çevirebileceğimiz popüler bir kitap) adında bir kitap dahi yayınlanmıştır.

Bununla birlikte son birkaç yılda Python modern yazılım geliştirme, altyapı yönetimi ve veri analizinde birinci sınıf bir programlama dili olarak ön plana çıkmıştır. Artık hackerlar için bir arka kapı oluşturucusu değil, web uygulaması oluşturma ve sistem yönetiminde önemli rol alma, veri analizleri ve makine öğreniminde parlayan bir dil olarak ün kazanmıştır.

2003 yılından itibaren PythonTIOBE Programlama Topluluğu Endeksi'nde en popüler 10 programlama dili arasında istikrarlı bir şekilde yer alırken, Ekim 2021 itibarıyla Java ve C programlama dillerini geçerek en popüler dil konumunda bulunmaktadır. 2007, 2010, 2018 and 2020 yıllarında ise bir yıl içerisindeki en yüksek kademe artışı çatısı altında “Yılın Programlama Dili” seçilmiştir ve bunu 4 kez yapabilen tek dildir.

Deneyisel bir akademik çalışma, Python gibi komut dosyası yazma dillerinin, dize işlemeyi ve sözlükte aramayı içeren programlama sorunları için C ve Java gibi geleneksel dillerden daha üretken olduğunu raporlamış, bellek tüketiminin genellikle "Java'dan daha verimli ve C veya C++'dan çok daha verimsiz" olmadığını saptamıştır.

Python kullanan büyük kuruluşlar arasında Wikipedia, Google, Yahoo!, CERN, NASA, Facebook, Amazon, Instagram ve Spotify gibi bazı kuruluşlar yer almaktadır. Sosyal haber ağı sitesi Reddit, çoğunlukla Python ile yazılmıştır(Vikipedia, 2022).

1.3 YAPAY ZEKA, MAKİNE ÖĞRENMESİ VE DERİN ÖĞRENME

1.3.1 Yapay Zeka

En basit ifadeyle yapay zeka, görevleri yerine getirmek için insan zekasını taklit eden ve topladıkları bilgilere göre yinelemeli olarak kendilerini iyileştirebilen sistemler veya makineler anlamına gelir. Makine öğrenimi ve yapay zeka genellikle bir arada değerlendirilir. Kimi durumlarda birbirinin yerine kullanılır ancak aynı anlama gelmezler. Tüm makine öğrenimi çözümleri yapay zeka iken tüm yapay zeka çözümlerinin makine öğrenimi olmaması önemli bir ayrımdır.

Yapay Zeka, zayıf ve güçlü olarak iki kategoriye ayrılır. Zayıf yapay zeka, genellikle medyada duyduğumuz yapay zeka gelişmelerinin tamamıdır. Sesli komut sistemleri, yüz tanıma sistemleri, otonom arabalar, insanları Go veya satranç gibi oyunlarda yenen makineler zayıf yapay zeka örnekleridir. Güçlü yapay zeka ise genel zeka noktasına ulaşmış bir programı veya makineyi tanımlamak için kullanılır. Bu, programların insan zekasıyla eşit olduğu ve ortalama bir insanın yapabileceği her şeyi yapma yeteneğine sahip olduğu zamandır. Henüz orada değiliz, ancak yakın gelecekte o noktaya ulaşacağımız düşünülmektedir(Matematiksel.org, 2022).

1.3.2 Makine Öğrenmesi

Makine Öğrenimi, makinenin açık bir şekilde programlanmadan örneklerden ve deneyimlerden öğrenmesini sağlayan bir kavramdır. İnsan müdahalesi olmadan öğrenmek için makineler algoritmalar ve istatistiksel modeller kullanır. Makine öğrenimi, bugün kullandığımız hizmetlerin çoğunu işleten süreçtir. Örneğin; Netflix, YouTube ve Spotify’ daki öneri sistemleri, Google ve Baidu gibi arama motorları, Facebook ve Twitter gibi sosyal medya mecraları, Siri ve Alexa gibi ses asistanları. Bu liste uzar da gider. Her türlü platform sizinle ilgili mümkün olduğunca çok veri toplar. Toplanan veri sayesinde de makine öğrenimi sağlanır. Bu sayede de sizin bir sonraki seferde ne talep edeceğinize dair yüksek ihtimalle doğru olan bir tahminde bulunulur. Makine öğrenimi, üç türe ayrılmıştır: denetimli (supervised), denetimsiz (unsupervised) ve pekiştirmeli (reinforcement) öğrenme(Matematiksel.org, 2022).

1.3.2.1 Denetimli, Denetimsiz ve Pekiştirmeli Öğrenme

Denetimli Öğrenim, öğrenmenin bir öğretmen tarafından yönlendirildiğini düşünebileceğiniz bir öğrenmedir. Öğretmen olarak hareket eden bir veri setimiz vardır. Bu verinin görevi ise makineyi eğitmektir. Model eğitildikten sonra, kendisine yeni veri verildiğinde bir tahmin veya karar vermeye başlar. Denetimsiz öğrenme, makineye hiçbir etiketli veri sağlanmadığında ve algoritmaların herhangi bir rehberlik olmaksızın bilgiler üzerinde hareket etmesine izin verildiğinde gerçekleşir. Makine, diğer veri kaynaklarından önceden herhangi bir eğitim almadan, sıralanmamış bilgileri benzerliklere, kalıplara ve farklılıklara göre gruplandırır. Pekiştirmeli öğrenme de ise bir algoritma, net bir hedefe ulaşmak için deneme yanılma yoluyla öğrenir(Matematiksel.org, 2022).

1.3.3 Derin Öğrenme

Algoritmaların beynin yapısından ve işlevinden ilham aldığı Makine Öğreniminin bir alt kümesidir. Tanımda kullanılan “derin” kelimesi, ağı katman sayısının çok olmasından gelir. Bu öğrenme yöntemi yapay sinir ağlarına dayalıdır.

Derin Öğrenme algoritmaları yapay sinir ağlarının daha karmaşık hali olarak düşünülebilir. Bu sinir ağları insandaki öğrenme işleyişinden hareketle geliştirilmiştir. Algoritmanın bu şekilde; öğrenme, hafızaya alma ve veriler arasındaki ilişkiyi ortaya çıkarma kapasitesine sahip olacağı düşünülmüştür. Derin Öğrenme algoritmalarının makine öğrenmesindeki var olan algoritmalarından ayrılan yönü ise çok yüksek miktarda veriye ve karmaşık yapısı ile de bu yüksek veriyi işleyebilecek çok yüksek hesaplama gücü olan donanımlara ihtiyaç duymasıdır.

Derin öğrenme, makinelerin dünyayı algılama ve anlamasına yönelik yapay zekâ geliştirmede en popüler yaklaşımdır. Pek çok önde gelen firma başta ses ve yüz tanıma alanı olmak üzere farklı alanlar için çalışmalara başlamış ve kendi derin öğrenme kütüphanelerini oluşturmaya başlamışlardır(Matematiksel.org, 2022).

1.4 LİTERATÜR TARAMASI

1.4.1 COVID-19 Denetiminde Yapay Zeka Çalışmaları

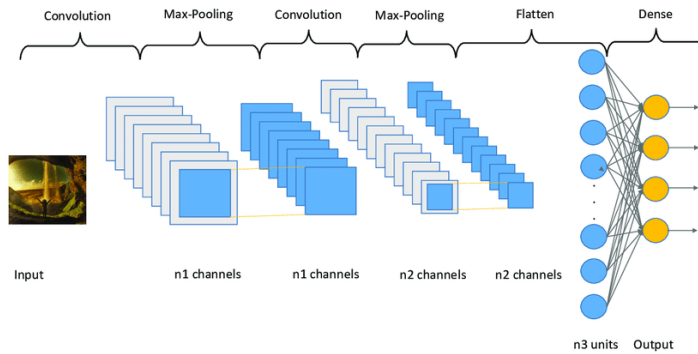
COVID-19 sürecinde maske kullanımı ve sosyal mesafe kurallarına uyulması ve bunun denetiminin yapılması büyük önem taşımaktadır. Bu sayede hastalığın bulaşma ve yayılma hızında yavaşlama, ölümlerde azalma sağlanacak ve hastalığın zirve yapmasının önüne geçilecektir (Worby ve Chang, 2020). Maske kullanımı ve sosyal mesafe kurallarına uyulması önem taşımaya devam ederken bunun denetiminin yapılması da bir o kadar önemlidir. En ufak bir ihmal bile hastalığın birçok kişiye yayılmasına ve katlanarak devam etmesine neden olabilecektir. Bu yüzden otonom denetim yapmak ve buna karşı önlem almak zararı en aza indirmek açısından büyük önem taşımaktadır. Bu bağlamda görüntü işleme ve yapay zeka teknolojilerinden yararlanmak son derece mantıklı olacaktır. COVID-19 için yüz maskesi denetimi ve sosyal mesafe tespiti yapan birçok uygulama ve çalışma bulunmaktadır. Görüntüler üzerinde sınıflandırma, bölütleme, anlamlandırma gibi problemlere çözüm üretmede CNN tabanlı ağlar ve geliştirilen modeller başarısını kanıtlamış bulunmaktadır. Ayrıca yüz maskelerinin tespiti ve sosyal mesafe tespitinde sıklıkla kullanılan konulardan birisi aktarmalı öğrenmedir. Bunun nedeni daha önceden eğitilmiş ve yüksek performans göstermiş ağların başka bir özel problem için kullanılmasındaki kolaylık, performans başarısı ve eğitim hızıdır. Bu bağlamda derin öğrenme tabanlı uygulamalar ile başarılı tespit işlemleri gerçekleştirilmektedir. Loey, 10 Manogaran, Taha ve Khalifa (2021a), öznitelik çıkarımı için ResNet50 tabanlı derin aktarımlı öğrenme ve tıbbi yüz maskelerinin tespiti için YOLOv2 mimarisini kullanan bir model önerdiler. Sonucunda ise %81 ortalama hassasiyet yüzdesi elde ettiler. Yaptıkları bir diğer çalışmalarında (2021b) ise derin öğrenme ve klasik makine öğrenmesini kullanan hibrit bir model tasarladılar. Öznitelik çıkarma ve karar ağaçları oluşturmak amaçlı ResNet-50, sınıflandırma işlemi için Destek Vektör Makineleri (SVM) ve topluluk algoritmasını kullandılar. Üç farklı veri seti için sırasıyla %99.64, %99.49 ve %100 test doğruluğu elde ettiler. Chowdary ve diğerleri (2020), Simulated Masked Face Dataset (SMFD) üzerinde InceptionV3 modeli ile aktarmalı öğrenmeyi kullanarak eğitim sırasında %99.9 ve test sırasında ise %100 doğruluk yüzdesi elde ettiler.

Sanjaya ve Rakhmawan (2020) birlikte yaptıkları çalışmada görüntü sınıflandırma ile yüz maskesi tanıma amacıyla MobileNetV2 modelini kullandılar. Kullandıkları bu model ile %96.85 doğruluk yüzdesi elde ettiler ve maske takmanın ve bunun COVID-19 için önemi arasında güçlü bir korelasyon elde ettiler. Fan ve Jiang(2021), yüksek seviyeli anlamsal bilgileri çoklu özellik haritalarıyla birleştirmek için ve yüzdeki maskeleri algılama amaçlı tek aşamalı bir detektör olan RetinaFaceMask'ı önerdiler. Ayrıca, düşük güvenilirlik ve yüksek birleşim kesişimi ile tahminleri reddetmek amaçlı nesne kaldırma algoritmasını sundular. Halka açık olarak erişim sağlanabilen veri seti ile temel sonuçlara oranla daha başarılı sonuçlar elde ettiler. Kullanılan Alexnet, Inception, ResNet, MobileNet, VGG ve YOLO gibi birçok CNN tabanlı mimariler ve türevleri ile yüksek başarı sonuçları elde edilmiştir (Hariri, 2021; Yu ve Zhang, 2021; Mahurkar ve Gadge, 2021).

1.5 EVRİŞİMSEL SİNİR AĞLARI

Bir evrişimsel sinir ağı (Convolutional neural networks -CNN), bir girdi görüntüsünü alıp, görüntüdeki çeşitli görünüşleri/nesneleri birbirinden ayırabilen derin öğrenme algoritmasıdır.

Evrişimli sinir ağları(Şekil 1.2), temel olarak görüntüleri sınıflandırmak (örneğin gördüklerini isimlendirmek), benzerlikle kümelemek (fotoğraf arama) ve sahnelerde nesne tanıma yapmak için kullanılan derin yapay sinir ağlarıdır(medium.com, 2022).



Şekil 1.2: Örnek CNN yapısı(researchgate.net, 2022).

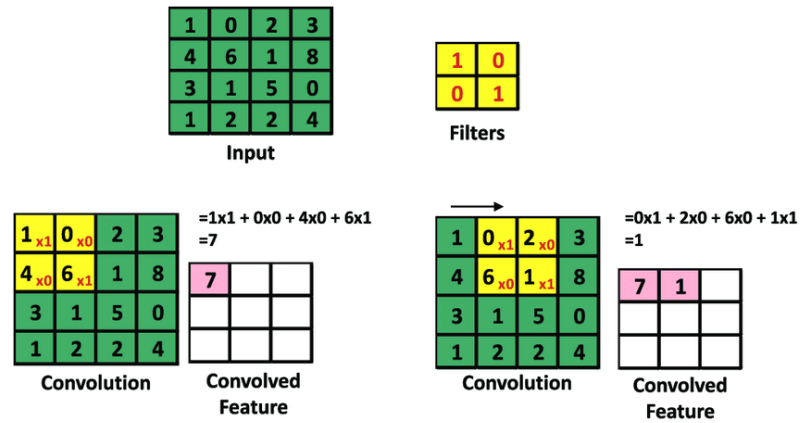
1.5.1 Evrişimsel Sinir Ağları Nasıl Çalışır?

Evrişimsel sinir ağları yapısı gereği input olarak resim ya da videolar alır. Tabi ki resimleri alırken ilgili formata çevrilmiş olması gerekir. Örneğin bir konvolüsyonel sinir ağına bir resim veriyorsak bunu matris formatında vermemiz gerekiyor.

Convolutional Neural Network işlemleri etkili bir şekilde gerçekleştirebilmek için bazı katmanlara sahiptir. Aşağıda bu katmanlar sırayla incelenmiştir.

1.5.1.1 Convolutional Layer (Evrişim Katmanı)

Bu katmanda girdi olarak gelen resim bir filtreden geçirilir. Filtreden geçirilme sonucu oluşan değerler öznitelik haritasını oluşturur.

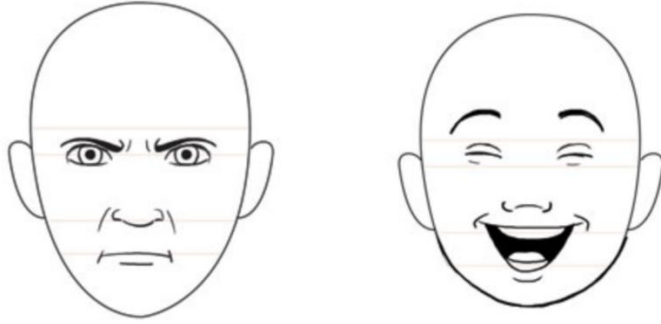


Şekil 1.3: Örnek Evrişim Katmanı Şeması(medium.com, 2022).

Bu aşama Şekil 1.3'de aşama aşama görülebilir. Input olarak elimizde 4x4 lük matris vardır. Belirlenen filtre ise 2x2 boyutundadır. (Tabi gerçek resimlerde kullanılan filtreler genellikle 3x3, 5x5 boyutlarında olabilir. AlexNet gibi ünlü modellerden bazıları ise 7x7lik filtreler kullanmaktadır.) Burada input olarak gelen matrisin tamamında bu filtre gezdirilerek, input üzerinde oluşan her 2x2 matris ile filtre matrisi çarpılarak öznitelik matrisi oluşturulur. Birden fazla filtre olması durumunda bu işlemler her bir filtre için uygulanır. Şekil 2.3'deki durum için 2x2 lik toplamda 4 adet farklı filtre kullanılmış olsaydı sonuç olarak ortaya 4x3x3 lük öznitelik matrisi oluşurdu. Görüntü işleme problemlerinde genel olarak resimler siyah-beyaz yapılarak modele input olarak verilir. Ancak bazı durumlarda renklerin etiket

üzerindeki etkiler çok büyüktür (Bir meyve veya sebzenin kalitesinin rengiyle doğrudan alakalı olduğu durumlar gibi). Bu gibi durumlarda resimleri renkli şekilde işlemek zorunludur. Eğer modele renkli bir input verilirse yukarıdaki işlemler her bir renk katmanı için ayrı ayrı yapılır (RGB).

Convolutional layer kullanılmadan resimlerdeki her bir piksel için modele bir input katmanı eklenebilir. Her bir input için gereken bağlantılar sağlanabilir. Ancak çok fazla bağlantı ve işlem gücü gerektirdiği için bu işlemi doğrudan yapmak yerine öznitelik haritası çıkartarak daha az sayıda bağlantı ile bu işlem gerçekleştirilebilir. Örnek olarak elimizde insan yüzlerinden oluşan bir veri tabanı olduğunu düşünelim. Bu yüzleri mutlu ve mutsuz insanlar olarak da sınıflandırmak istiyoruz.

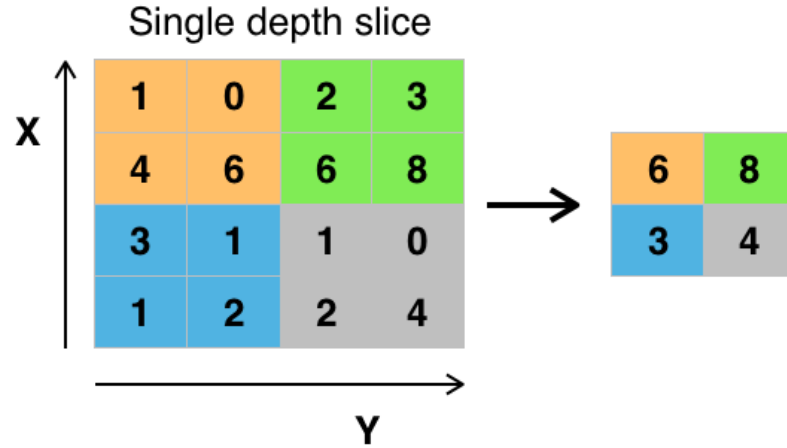


Şekil 1.4: Somurta – Gülen Yüz (medium.com, 2022).

Bu noktada elimizde Şekil 1.4'de olduğu gibi farklı yüz ifadelerine sahip insan resimleri olacaktır. Burada model ağız, göz ve kaş çevrelerindeki matris farklılıklarının, etiket üzerinde büyük etkisinin olduğunu öğrenerek eğitimini buna göre tamamlayacaktır. Ancak yine Şekil 1.4'de görüldüğü üzere alın ve kulak bölgelerinde her iki resimde aynı gözükmektedir. Haliyle bu değerlerin eğitim ve tahmin üzerindeki etkileri çok az olacak veya tamamen etkisiz olacaklardır. CNN'in gücü burada ortaya çıkarak Convolutional katmanı ile resimlerdeki sadece gerekli ve önemli olan öznitelikler belirlenir. Farklılık oluşturmeyen öznitelikler ise hesaba dahil edilmez.

1.5.1.2 Havuzlama Katmanı (Pooling Layer)

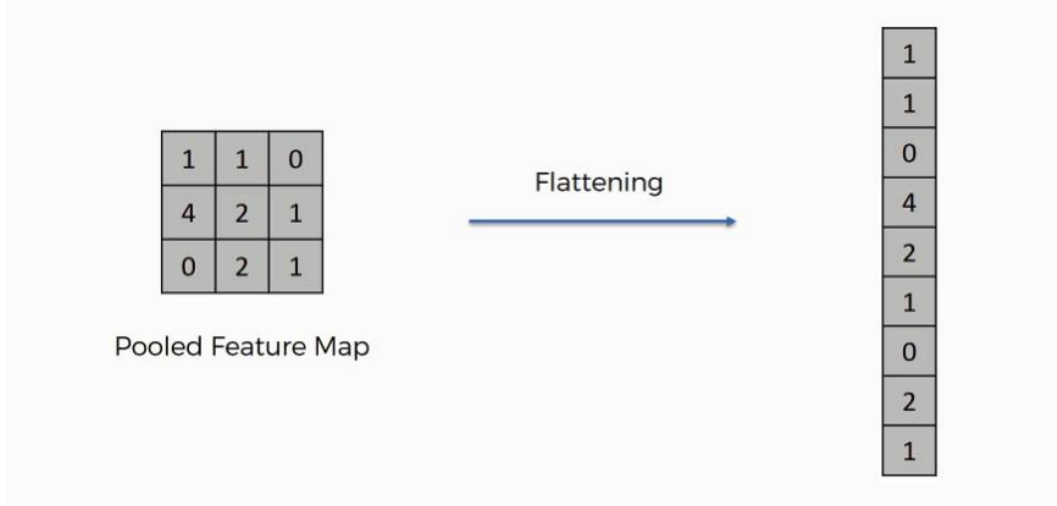
Pooling layer genellikle oluşturulan öznitelik matrislerine uygulanır. Şekil 1.5'de 4x4 lük bir Öznitelik matrisi görülmektedir. Burada 2x2 lik bir maximum pooling uygulanırsa sağdaki 2x2 lik matris oluşur. Öznitelik matrisinde pooling için belirlenen boyutlarda (Şekil 1.5 için 2x2) matrisler oluşturularak, o matris içerisindeki en büyük değer alınır. Turuncu olarak gösterilen ilk matriste 6, yeşil olarak belirlenen matriste ise bu değer 8'dir. Genellikle kullanılan yöntem maximum pooling olsa da mean pooling ve minimum pooling gibi yöntemler de mevcuttur. Mean pooling yönteminde ortalama alınırken, minimum pooling yönteminde ise en düşük değer alınarak işlem gerçekleştirilir.



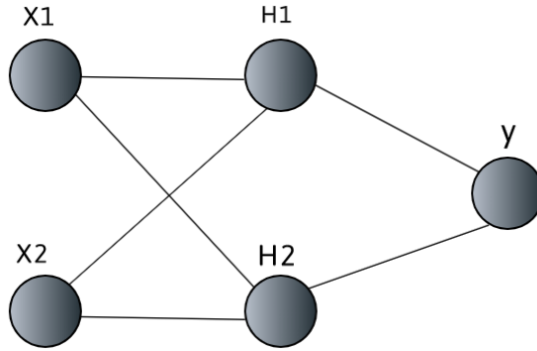
Şekil 1.5: Örnek Pooling Layer Şeması (medium.com, 2022).

1.5.1.3 Düzleştirme Katmanı (Flattening Layer)

Şekil 1.6'da katmanlar modelde birden fazla kez kullanılabilir. Bu işlemler tamamlandıktan sonra elde edilen matris Tam Bağlı(Fully Connected) katmanında kullanılabilmesi için düzleştirilmesi gerekir. Bu sebepten ötürü Şekil 1.6'da örnek olarak görülebilecek flattening işlemi gerçekleştirilir. Bu aşamadan sonra Fully Connected Layer'ın(Şekil 1.7) inputları hazırlanmış olur.



Şekil 1.6: Örnek Flattening Şeması (medium.com, 2022).

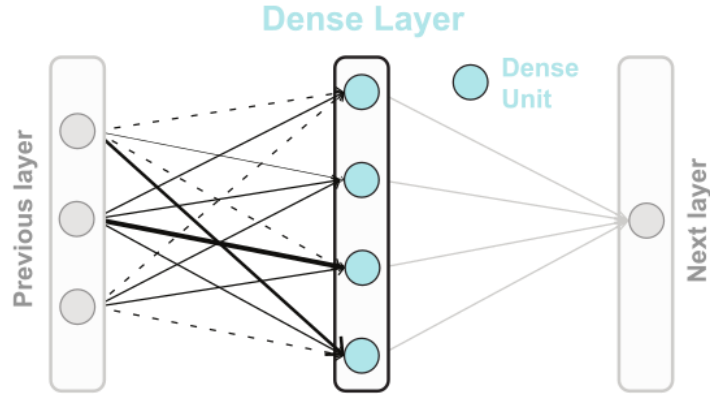


Şekil 1.7: Basic Connected Layer (medium.com, 2022).

Şekil 1.7'de basit bir sinir ağı(neural networks) yapısı görülmektedir. Bu yapı fully connected layer yapısının daha basitleşmiş hali olarak da görülebilir. Haliyle bu yapıyı anlamak fully connected layer'ın çalışma prensibini anlamamıza yardımcı olacaktır. Öncelikle x1 ve x2 ile flattening yaparak elde ettiğimiz input layer kısmı oluşturulur. Sonrasında input katmanından gelen değerler h1 ve h2 ile hidden layer dediğimiz gizli katmanlarda, model tarafından belirlenen katsayılarla (çoğu zaman fonksiyon) işleme girer. Bu işlem sonucunda belirlenen aktivasyon fonksiyonuna göre y, yani output değeri üretilir.

1.5.1.4 Yoğunlaştırma Katmanı(Dense Layer)

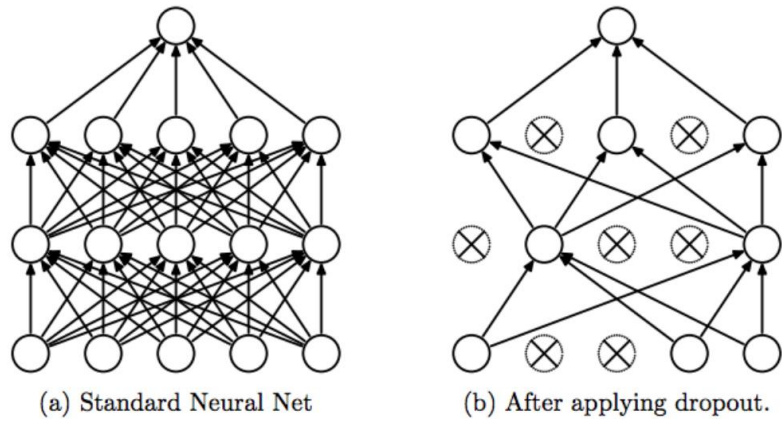
Dense, çoğu durumda çalışan standart bir katman türüdür. Yoğun bir katmanda, önceki katmandaki tüm düğümler mevcut katmandaki düğümlere bağlanır(Şekil 1.8).



Şekil 1.8: Dense Layer(medium.com, 2022).

1.5.1.5 Bırakma Katmanı(Dropout Layer)

Çok katlı yapay sinir ağları aşırı öğrenme adı verilen bir durumla karşılaşmaktadır. Bu istenmediği için dropout katmanında ağda ezber yapan özelliklerin kaldırılması gerçekleşir(Şekil 1.9).

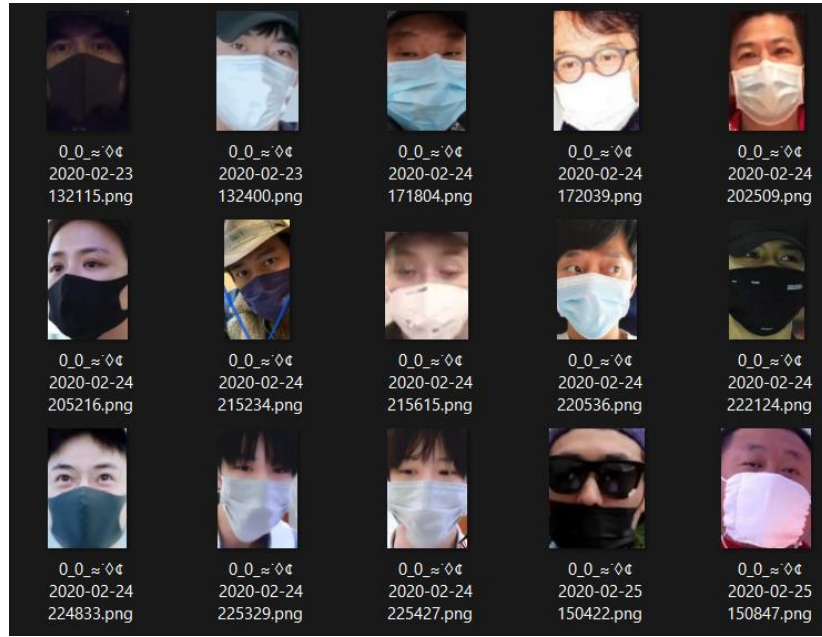


Şekil 1.9: Dropout Layer(medium.com, 2022).

2. GEREÇLER

2.1 VERİ SETİ

Veri seti masked(Maskeli) ve unmasked(Maskesiz) şeklinde 2 dosyadan oluşmaktadır(kaggle.com, 2022).



Şekil 2.1: Veri setinin önizlemesi

2.2 PRETRAINED FACE DETECTOR (ÖNCEDEN EĞİTİLMİŞ YÜZ DEDEKTÖRÜ)

Yüz maskesi detektörümüz herhangi bir biçimlendirilmiş maskeli görüntü veri seti kullanmamaktadır. MobileNetV2 mimarisinin kullanımı sayesinde, hesaplama açısından verimlidir(github.com, 2022).

2.3 SPYDER IDE (SPYDER GELİŞTİRME ORTAMI)

Spyder, Python'da bilimsel programlama için açık kaynaklı, platformlar arası entegre bir geliştirme ortamıdır (IDE). Spyder, NumPy, SciPy, Matplotlib, pandas, IPython, SymPy ve Cython'un beraberinde diğer açık kaynaklı yazılımlar da olmak

üzere bilimsel Python yığınındaki bir dizi önde gelen paketle entegre olur. MIT lisansı altında yayınlanmaktadır(Vikipedia, 2022).

2.4 KULLANILAN KÜTÜPHANELER

2.4.1 Tensorflow

TensorFlow, deep learning için geliştirilmiş açık kaynaklı bir matematik kütüphanesidir. Bahsettiğim gibi makine öğrenme kümesi altında geliştiği için TensorFlow geleneksel makine öğrenme algoritmalarının da desteklemektedir ve öncelikle Google Brain ekibi tarafından dahili Google kullanımı için geliştirilmiştir.

Temelinde Python kullanılarak geliştirilen bu framework, günümüzde Python'ın yanısıra C++, Java, C#, Javascript ve R gibi birçok dili desteklemektedir(medium.com, 2022).

2.4.2 Keras

Keras bir deep learning kütüphanesidir ve az önce bahsettiğimiz TensorFlow üzerinde çalışır. Python aracılığı ile yazılmış ve neredeyse her tür deep learning modelini tanımlayan bir kütüphanedir. Yalnızca TensorFlow değil diğer Theano ve CNTK üzerinde çalışabilen bir üst düzey sinir ağları API'sıdır.

Keras öğrenme kolaylığının ve model oluşturma kolaylığının ötesinde, geniş çapta benimsenme, çok çeşitli export seçenekleri destekler. Bunun yanında Keras Google, Microsoft, Amazon, Apple, Nvidia, Uber tarafından desteklenmektedir(medium.com, 2022).

2.4.3 NumPy

Python programlama dili için büyük, çok boyutlu dizileri ve matrisleri destekleyen, bu diziler üzerinde çalışacak üst düzey matematiksel işlevler ekleyen bir kütüphanedir. NumPy'nin atası Numeric, ilk olarak Jim Hugunin tarafından diğer birkaç geliştiricinin katkılarıyla oluşturuldu. 2005 yılında Travis Oliphant, Numarray'in özelliklerini kapsamlı değişikliklerle Numeric'e dahil ederek NumPy'yi yarattı. NumPy açık kaynaklı bir yazılımdır ve birçok katkıda bulunanlara sahiptir(Vikipedia, 2022).

2.4.4 Sklearn

Scikit-learn, veri bilimi ve makine öğrenmesi için en yaygın kullanılan Python paketlerinden biridir. Birçok işlemi gerçekleştirmenizi sağlar ve çeşitli algoritmalar sağlar. Scikit-learn ayrıca sınıfları, yöntemleri ve işlevleri ile kullanılan algoritmaların arka planıyla ilgili belgeler sunar(F4alt.com, 2022).

2.4.5 OS

Os modülü Python'da hazır olarak gelen , dosya ve dizinlerde kolaylıkla işlemler yapmamızı sağlayan bir modüldür(medium.com, 2022).

2.4.6 Matplotlib

Matplotlib, python programlama dilinde 2D grafikler için kullanılan bir çizim kitaplığıdır. Python komut dosyalarında, kabukta, web uygulama sunucularında ve diğer grafik kullanıcı arayüzü araç setlerinde kullanılabilir.

Matplotlib, Python programlama dili ve onun sayısal matematik uzantısı NumPy için bir çizim kitaplığıdır. Tkinter, wxPython, Qt veya GTK + gibi genel amaçlı GUI araç takımlarını kullanarak çizimleri uygulamalara yerleştirmek için nesne yönelimli bir API sağlamak için kullanılır. Ayrıca, kullanımı tavsiye edilmese de, MATLAB'a çok benzemek üzere tasarlanmış, durum makinesine (OpenGL gibi) dayalı yordamsal bir "pilab" arayüzü de vardır(medium.com, 2022).

2.4.7 Imutils

OpenCV ve Python ile çeviri, döndürme, yeniden boyutlandırma, iskeletleme ve Matplotlib görüntülerini görüntüleme gibi temel görüntü işleme işlemlerini kolaylaştırmak için kullanılan modüldür(GitHub.com, 2022).

2.4.8 Time

Zaman, saat ve tarihlerle ilgili işlemler yapmamızı sağlam standart python kütüphane modülüdür.

2.4.9 OpenCv

Gerçek zamanlı bilgisayar görüşü uygulamalarında kullanılan açık kaynaklı kütüphane. İlk olarak Intel tarafından geliştirilmiş, daha sonra Willow Garage ve sonra

Itseez (Intel tarafından satın alındı) tarafından sürdürüldü. Bu kütüphane çoklu platform ve BSD lisansı altında açık kaynaklı bir yazılımdır.

Aslen C++ diliyle yazıldığı için birincil arayüzü C++ içindir. Yeni geliştirilen özellikler ve algoritmalar ilk olarak C++ arayüzüne eklenir. Ancak, eski dili C için de daha kısıtlı bir arayüzü vardır. Python, Java ve MATLAB için de bağları vardır. Daha geniş kitleye ulaşmak için C#, Perl, Ch, Haskell ve Ruby dilleri için de wrapperlar geliştirilmiştir. 3.4 sürümünden beri, web platformu için OpenCV.js adıyla JavaScript bağı mevcuttur(Vikipedia.com, 2022).

3. YÖNTEMLER

3.1 MODELİN EĞİTİLMESİ

Kütüphane kodlarının yazılması(Şekil 3.1).

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.layers import AveragePooling2D
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Input
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from imutils import paths
import matplotlib.pyplot as plt
import numpy as np
import os
```

Şekil 3.1: Kütüphane Kodları.

Başlangıç öğrenme oranı(INIT_LR), devir(EPOCHS) ve aynı anda eğitilecek veri miktarı(BS) değerlerini giriyoruz(Şekil 3.2).

```
INIT_LR = 1e-4 #başlangıç öğrenme oranı
EPOCHS = 10 #veri setini eğitirken yapılan her bir tekrar için kullanılan terim
BS = 32 # aynı anda eğitilecek veri miktarı
```

Şekil 3.2: Başlangıç öğrenme oranı, devir, aynı anda eğitilecek veri miktarı.

Veri setindeki görüntüye alınması ve maskeli, maskesiz şeklinde ikiye ayırmak için dizi(array) oluşturulması(Şekil 3.3).

```
DIRECTORY = r"dataset" #veri seti dizinimizdeki görüntülerin listesini alınması
CATEGORIES = ["with_mask", "without_mask"] #maskeli ve maskesiz şeklinde 2 elemanlı kategori listesi oluşturulması
```

Şekil 3.3: Veri setindeki görüntülerin listeye alınması, kategori listesinin oluşturulması.

Verilerimiz işlemek için iki adet boş dizi oluşturulması(Şekil 3.4).

```
data = []  
labels = []
```

Şekil 3.4: Verilerin işlenmesi için 2 boş liste oluşturulması.

Verilerimizin 224x224 boyutuna dönüştürülmesi ve eğitmek için dizi(array) veri tipine dönüştürülmesi için yazılan iç içe döngü. Verilerin bir önceki adımdaki data dizisine, kategorilerin(maskeli, maskesiz) labels dizisine eklenmesi(Şekil 3.5).

```
for category in CATEGORIES:  
    path = os.path.join(DIRECTORY, category)  
    for img in os.listdir(path):  
        img_path = os.path.join(path, img)  
        image = load_img(img_path, target_size=(224, 224))  
        image = img_to_array(image)  
        image = preprocess_input(image)  
  
        data.append(image)  
        labels.append(category)
```

Şekil 3.5: Resimlerin yeniden boyutlandırıp array veri tipine dönüştüren döngü.

Label dizisindeki kategorilerimizin 0 ve 1'e dönüştürülmesi(Şekil 3.6).

```
lb = LabelBinarizer()  
labels = lb.fit_transform(labels)  
labels = to_categorical(labels)
```

Şekil 3.6: Etiketlerin(maskeli, maskesiz) 0 ve 1'e dönüştürülmesi.

Oluşturduğumuz data ve labels dizilerinin NumPy dizisine dönüştürülmesi(Şekil 3.7).

```
data = np.array(data, dtype="float32")  
labels = np.array(labels)
```

Şekil 3.7: Resimlerin ve Etiketlerin NumPy array'a dönüştürülmesi.

Oluşturduğumuz verileri(data, labels) train_test_split methodu ile %75'i eğitim %25'i test için ayrılması(Şekil 3.8).

```
(trainX, testX, trainY, testY) = train_test_split(data, labels,
test_size=0.20, stratify=labels, random_state=42)
```

Şekil 3.8: Verilerin %75'i eğitim için ve %25'i test için ayrılması.

Verimizin ImageDataGenerator methodu ile büyütülmesi(Şekil 3.9).

```
aug = ImageDataGenerator(
    rotation_range=20,
    zoom_range=0.15,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.15,
    horizontal_flip=True,
    fill_mode="nearest")
```

Şekil 3.9: Veriyi çoğaltmak için generator oluşturulması.

MobileNetV2 ağıının bağlanması(3.10).

```
baseModel = MobileNetV2(weights="imagenet", include_top=False,
input_tensor=Input(shape=(224, 224, 3)))
```

Şekil 3.10: MobileNetV2 Ağıının bağlanması.

Sinir ağı katmanlarımızın oluşturulması(Şekil 3.11).

```
headModel = baseModel.output
headModel = Conv2D(32, kernel_size=(3,3), activation="relu")(headModel) #Evrişim Katmanı
headModel = AveragePooling2D(pool_size=(7, 7))(headModel) #Havuzlama Katmanı
headModel = Flatten(name="flatten")(headModel) #Düzleştirme Katmanı
headModel = Dense(128, activation="relu")(headModel) #Yoğunlaştırma Katmanı
headModel = Dropout(0.5)(headModel) #Aşırı öğrenmeyi Engelleyen Katman
headModel = Dense(2, activation="softmax")(headModel) #2.Yoğunlaştırma Katmanı
```

Şekil 3.11: Sinir ağıının oluşturulması.

Bir değişken oluşturulup MobileNetV2 ve sinir ağımızın değişkene atanması(Şekil 3.12).

```
model = Model(inputs=baseModel.input, outputs=headModel)
```

Şekil 3.12: Oluşturduğumuz sinir ağının MobileNetV2'ye bağlanması.

MobileNetV2 ağımızı ilk bağlandığımız içinde veri olmadığını için eğitim işleminin başında eğitilmesini engellememiz için oluşturulan döngü(Şekil 3.13).

```
for layer in baseModel.layers:  
    layer.trainable = False
```

Şekil 3.13: MobileNetV2 ağının eğitilmemesi içine yazılan döngü.

Modelimizin derlenmesi(Şekil 3.14).

```
print("[INFO] compiling model...")  
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)  
model.compile(loss="binary_crossentropy", optimizer=opt,  
              metrics=["accuracy"])
```

Şekil 3.14: Modelin Derlenmesi.

Modelimizin başının eğitilmesi(Şekil 3.15).

```
print("[INFO] training head...")  
H = model.fit(  
    aug.flow(trainX, trainY, batch_size=BS),  
    steps_per_epoch=len(trainX) // BS,  
    validation_data=(testX, testY),  
    validation_steps=len(testX) // BS,  
    epochs=EPOCHS)
```

Şekil 3.15: Ağın başının eğitilmesi.

Model raporunun verilerinin alınması için oluşturulan değişken(Şekil 3.16).

```
print("[INFO] evaluating network...")
predIdxs = model.predict(testX, batch_size=BS)
```

Şekil 3.16: Test setinde tahminlerin alınması.

Test setindeki her bir görüntü için karşılık gelen en büyük tahmin edilen olasılığa sahip etiket dizinini bulunması(Şekil 3.17).

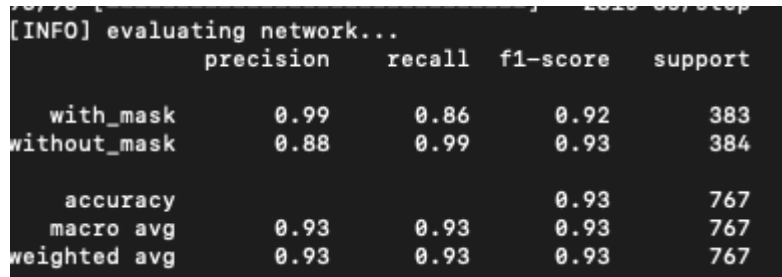
```
predIdxs = np.argmax(predIdxs, axis=1)
```

Şekil 3.17: Test setindeki her bir görüntü için karşılık gelen en büyük tahmin edilen olasılığa sahip etiketin dizinin bulunması.

Sklearn kütüphanesi ile model raporunun oluşturulması(Şekil 3.18).

```
print(classification_report(testY.argmax(axis=1), predIdxs,
    target_names=lb.classes_))
```

Şekil 3.18: Model raporunun oluşturulması.



	precision	recall	f1-score	support
with_mask	0.99	0.86	0.92	383
without_mask	0.88	0.99	0.93	384
accuracy			0.93	767
macro avg	0.93	0.93	0.93	767
weighted avg	0.93	0.93	0.93	767

Şekil 3.19: Model raporu sonuçları.

Modelin kaydedilmesi(Şekil 3.21).

```
print("[INFO] saving mask detector model...")
model.save("mask_detector.model")
```

Şekil 3.20: Modelin Kaydedilmesi.

3.2 MODELİN GÖRSELLEŞTİRİLİP KAMERAYA AKTARILMASI

Kütüphane kodlarının yazılması(Şekil 3.21).

```
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.models import load_model
from imutils.video import VideoStream
import numpy as np
import imutils
import time
import cv2
import os
```

Şekil 3.21: Kütüphanelerin aktarılması.

Ana programı barındıran def fonksiyonunun oluşturulması(Şekil 3.22).

```
#anaprogramı barındıran def fonksiyonu
def detect_and_predict_mask(frame, faceNet, maskNet):
    (h, w) = frame.shape[:2]
    blob = cv2.dnn.blobFromImage(frame, 1.0, (224, 224),
                                  (104.0, 177.0, 123.0))
```

Şekil 3.22: Çerçeve boyutlarının yakalama ve BLOB oluşturulması.

Yüz algılamalarını tespit edecek kod(Şekil 3.23).

```
faceNet.setInput(blob)
detections = faceNet.forward()
print(detections.shape)
```

Şekil 3.23: Ağ üzerindeki yüz algılamalarını alan kod.

Yüz, bunlara karşılık gelen konumları ve yüz ağındaki tahminleri içinde tutacağımız boş dizilerin oluşturulması(Şekil 3.24).

```
faces = []
locs = []
preds = []
```

Şekil 3.24: Yüz listemizi, bunlara karşılık gelen konumları ve yüz maskesi ağındaki tahminlerin listesinin oluşturulması.

Yüz tespiti için olasılığını bir değişkene atanması(Şekil 3.25).

```
for i in range(0, detections.shape[2]):  
    confidence = detections[0, 0, i, 2]
```

Şekil 3.25: Yüz algılama kontrolü için döngü oluşturulması.

Zayıf yüz algılanmalarını filtrelenmesi için yazılan koşul ve tespit edilen yüzleri çerçeve içine almamız için yazılan kod(Şekil 3.26).

```
if confidence > 0.5:  
    #yüzün x y kordinatları kodu  
    box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])  
    (startX, startY, endX, endY) = box.astype("int")
```

Şekil 3.26: Zayıf algılanmaların filtrelenmesi için yazılan koşul.

Tespit edilen yüzün koordinatlarının ilgili değişkenlere atanması(Şekil 3.27).

```
(startX, startY) = (max(0, startX), max(0, startY))  
(endX, endY) = (min(w - 1, endX), min(h - 1, endY))
```

Şekil 3.27: Tespit edilen yüzün çerçevesinin koordinatlarının hesaplanması.

Tespit edilen yüzlerin koordinatlarının face değişkenine atanması, renk skalasının BGR'dan RGB'ye dönüştürülmesi, 224x224 boyutunda tekrar boyutlandırılması, dizi veri tipine dönüştürülmesi ve modelin gerektirdiği formata uygun hale getirilmesi(Şekil 3.28).

```
face = frame[startY:endY, startX:endX]  
face = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)  
face = cv2.resize(face, (224, 224))  
face = img_to_array(face)  
face = preprocess_input(face)
```

Şekil 3.28: Tespit edilen yüzün renk skalasının BGR'dan RGB'ye çevrilmesi, yeniden boyutlandırılması ve diziye dönüştürülmesi.

Elde ettiğimiz yüz ve koordinat verilerinin oluşturduğumuz boş dizilere eklenmesi(Şekil 3.29).

```
faces.append(face)
locs.append((startX, startY, endX, endY))
```

Şekil 3.29: Tespit edilen yüzlerin ve çerçeve koordinatlarının ilgili listeye eklenmesi.

En az 1 yüz tespit edildiğinde koordinatlarının bulunup return edilmesi(Şekil 3.30).

```
if len(faces) > 0:
    faces = np.array(faces, dtype="float32")
    preds = maskNet.predict(faces, batch_size=32)
    return (locs, preds)
```

Şekil 3.30: En az 1 yüz bulunduğunda tarama yapma ve konumlarını döndürme kodu.

Önceden eğitilmiş yüz dedektörümüzün ilgili değişkenlere atanılıp, yüz ağımıza bağlanması(Şekil 3.31).

```
prototxtPath = r"face_detector\deploy.prototxt"
weightsPath = r"face_detector\res10_300x300_ssd_iter_140000.caffemodel"
faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)
```

Şekil 3.31: Önceden eğitilmiş yüz detektörü modelimizin yüklenmesi.

Değişken oluşturulup eğittiğimiz maske tespit modelinin yüklenmesi(Şekil 3.32).

```
maskNet = load_model("mask_detector.model")
```

Şekil 3.32: Eğittiğimiz maske tespit modelinin yüklenmesi.

Kameranın açılması(Şekil 3.33).

```
print("[INFO] starting video stream...")
vs = VideoStream(src=0).start()
```

Şekil 3.33: Kameranın açılması.

Video akışını başlatan döngünün oluşturulması, Kamera çözünürlüğünün belirlenmesi ve tespit edilen yüzlerin maske takıp takmadığının belirlenmesi(Şekil 3.34).

```
while True:
    frame = vs.read()
    frame = imutils.resize(frame, width=400)
    (locs, preds) = detect_and_predict_mask(frame, faceNet, maskNet)
```

Şekil 3.34: Video akışının başlatılması, video çerçevesinin boyutunun ayarlanması, tespit edilen yüzlerin maske takıp takmadığının belirlenmesi.

Algılanan yüz konumları ve tahminlerin indisinin alınması ve ilgili değişkenlere atanması(Şekil 3.35).

```
for (box, pred) in zip(locs, preds):
    (startX, startY, endX, endY) = box
    (mask, withoutMask) = pred
```

Şekil 3.35: Algılanan yüz konumları ve bunlara karşılık gelen konumlar üzerinde döngü oluşturulması.

Yapılan tahminde maskeli değeri maskesiz değerinden büyükse etikete(label) “Mask” değilse “No Mask” yapan döngü, etiket değeri “Mask” ise rengi(color) yeşile değilse kırmızıya dönüştüren döngü(Şekil 3.36).

```
label = "Mask" if mask > withoutMask else "No Mask"
color = (0, 255, 0) if label == "Mask" else (0, 0, 255)
label = "{}: {:.2f}%".format(label, max(mask, withoutMask) * 100)
```

Şekil 3.36: Yapılan tespit Maskeliyse çerçeveyi yeşile, değilse kırmızıya boyayan kod.

Çerçeve ve etiket oluşturulması(Şekil 3.37).

```
cv2.putText(frame, label, (startX, startY - 10),  
            cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)  
cv2.rectangle(frame, (startX, startY), (endX, endY), color, 2)
```

Şekil 3.37: Çerçevenin ve maske etiketinin oluşturulması.

Kameradan gelen görüntünün yansıtılması ve q tuşuna basıldığında programın kapatılması(Şekil 3.38).

```
cv2.imshow("Frame", frame)  
key = cv2.waitKey(1) & 0xFF  
if key == ord("q"):  
    break
```

Şekil 3.38: Kameradan gelen görüntünün yansıtılması ve q tuşuna basıldığında döngünün sonlandırılması.

Programı sonlandıran kod(Şekil 3.39).

```
cv2.destroyAllWindows()  
vs.stop()
```

Şekil 3.39: Programın sonlandırılması.

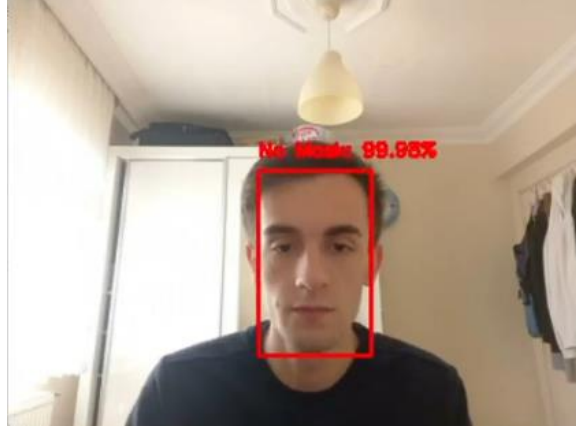
4. SONUÇ

4.1 Model Performansı

Modelin doğruluk(accuracy) oranı %93'tür(Şekil 3.19).

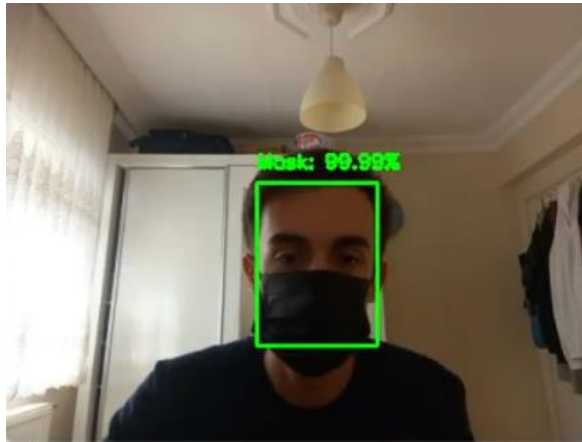
4.2 Görsel Sonuçlar

Programın maske takılı değilken yaptığı tespit(%99.98 tespit oranı, Şekil 3.40).



Şekil 3.40: Maskesiz Görünüm.

Programın maske varken ki tespit yaptığı tespit(%99.99 tespit oranı, Şekil 3.41).



Şekil 3.41: Maskeli Görünüm.

5. TARTIŞMA

Bu çalışmada 2019 yılının Aralık ayında ortaya çıkan ve tüm dünyayı etkisi altına alan COVID-19 virüsüne karşı evrimsel sinir ağları ile çözüm önerisi sunulmuştur. Kamera yardımıyla alınan video görüntüleri ile insanların maske takıp takmadığı denetlenmiştir. Maske takmayan bireylerin derin öğrenme tabanlı sistemlerle tespiti sağlanmıştır. Böylece otonom sistemler ile maske takmayan bireylere karşı hızlı önlem alınabilmektedir. Kural ihlallerinin minimuma indirilmesi ile virüsün yayılması azaltılacak ve karşılaşılan riskler en aza indirilmeye çalışılacaktır. Derin öğrenme tabanlı olarak geliştirilen bu sistemde daha önceden eğitilmiş yüz dedektörü modeli kullanılmış ve probleme özel veri seti kullanılarak maske tespit modeli eğitimi yapılmıştır.

Uygulamanın geliştirilmesi için farklı model mimarileri kullanılarak performans metriklerinde artış sağlanabilir. Böylece daha az hata oranları yakalanarak tespit işlemi daha iyi gerçekleştirilebilir. Ayrıca model için seçilen parametrelerde farklı yöntemler kullanılarak model performansı artırılabilir. Model kadar önemli olan bir diğer mesele de kullanılan veri sayısı ve veri setinin kendi içerisindeki varyasyonudur. Veri sayısındaki artış eğitim, doğrulama ve test performanslarını artıracaktır.

KAYNAKLAR

Bilginc.com(2022). Python Nedir? Python Hakkında Her şey, Erişim Tarihi: 10 Mayıs 2022,
<https://bilginc.com/tr/blog/158/python-nedir-python-hakkinda-hersey>

Chowdary, G. J., Pun, N. S., Sonbhadra, S. K. ve Agarwal, S. (2020). Face Mask Detection Using Transfer Learning of InceptionV3. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 12581 LNCS, 81–90.

F4alt.com(2022). Scikit-learn, Erişim Tarihi: 20 Mayıs 2022,
<https://www.f4alt.com/post/pai1926-python-ve-yapay-zeka-ya-giri%C5%9F-kamp%C4%B1-sorular-ve-cevaplari>

Fan, X. ve Jiang, M. (2020). RetinaMask: A Face Mask detector, Erişim Tarihi: 15 Mayıs 2022,
<http://arxiv.org/abs/2005.03950>

GitHub.com(2022). Pretrain Face Detector, Erişim Tarihi: 17 Mayıs 2022,
https://github.com/opencv/opencv/tree/3.4.0/samples/dnn/face_detector

GitHub.com(2022). Imutils, Erişim Tarihi: 20 Mayıs 2022,
<https://github.com/PyImageSearch/imutils>

Hariri, W. (2021). Efficient Masked Face Recognition Method during the COVID-19 Pandemic.

Kaggle.com(2022). Face Mask Dataset, Erişim Tarihi: 17 Mayıs 2022,
<https://www.kaggle.com/datasets/altaga/facemaskdataset>

Loey, M., Manogaran, G., Taha, M. H. N. ve Khalifa, N. E. M. (2021a). COVID-19 ile mücadele: A novel deep learning model based on YOLO-v2 with ResNet-50 for medical face mask detection. Sustainable Cities and Society, 65(October 2020), 102600.

Loey, M., Manogaran, G., Taha, M. H. N. ve Khalifa, N. E. M. (2021b). A hybrid deep transfer learning model with machine learning methods for face mask detection in the era of the COVID-19 pandemic. Measurement: Journal of the International Measurement Confederation, 167(May 2020), 108288.

Mahurkar, R. R. ve Gadge, N. G. (2021). Real-time Covid-19 Face Mask Detection with YOLOv4. Proceedings of the 2nd International Conference on Electronics and Sustainable Communication Systems, ICESC 2021, 1250–1255.

Matematiksel.org(2022). Yapay Zeka Makine Öğrenmesi ve Derin Öğrenme Fark Nedir?, Erişim Tarihi: 13 Mayıs 2022,

<https://www.matematiksel.org/yapay-zeka-makine-ogrenmesi-ve-derin-ogrenme-fark-nedir/>

Medium.com(2022). Derin Öğrenme Nedir? Nasıl Çalışır ?, Erişim Tarihi: 16 Mayıs 2022,

<https://medium.com/@rabiaokumus96/convolutional-neural-networks-evri%C5%9Fimsel-sinir-a%C4%9Flar%C4%B1-cceb887a2979>

Medium.com(2022). Keras ile Derin Öğrenme, Erişim Tarihi 24 Mayıs 2022,

<https://medium.com/@tuncerergin/keras-ile-derin-ogrenme-modeli-olusturma-4b4ffdc35323>

Medium.com(2022). Keras ve Tensorflow Nedir?, Erişim Tarihi 20 Mayıs 2022,

<https://mfatihto.medium.com/keras-ve-tensorflow-nedir-8a5b9e8d1452>

Medium.com(2022). Uygulamalı Evrişimsel Sinir Ağları (Convolutional Neural Network), Erişim Tarihi: 16 Mayıs 2022,

<https://medium.com/bili%C5%9Fim-hareketi/uygulamal%C4%B1-evri%C5%9Fimsel-sinir-a%C4%9Flar%C4%B1-convolutional-neural-network-7d643eae6a7>

Medium.com(2022). Python İle Veri Görselleştirme: Matplotlib Kütüphanesi-1, Erişim Tarihi: 20 Mayıs 2022, <https://medium.com/datarunner/matplotlibkutuphanesi-1-99087692102b>

Medium.com(2022). Python OS Modülü, Erişim Tarihi: 20 Mayıs 2022, <https://medium.com/@aykutuludag/python-os-mod%C3%BCI%C3%BC-1e6eced93069>

Researchgate.net(2022). A vanilla Convolutional Neural Network (CNN) representation, Erişim Tarihi: 16 Mayıs 2022, https://www.researchgate.net/figure/A-vanilla-Convolutional-Neural-Network-CNN-representation_fig2_339447623

Sanjaya, S. A. ve Rakhmawan, S. A. (2020). Face Mask Detection Using MobileNetV2 in the Era of COVID-19 Pandemic. 2020 International Conference on Data Analytics for Business and Industry: Way Towards a Sustainable Economy, ICDABI 2020.

Wikipedia (2022). COVID-19, Erişim Tarihi: 13 Mayıs 2022, <https://tr.wikipedia.org/wiki/COVID-19>

Wikipedia(2022). Numpy, Erişim Tarihi: 20 Mayıs 2022, <https://tr.wikipedia.org/wiki/NumPy>

Wikipedia.com(2022). OpenCV, Erişim Tarihi: 20 Mayıs 2022, <https://tr.wikipedia.org/wiki/OpenCV>

Wikipedia(2022). Python Programlama Dili, Erişim Tarihi: 13 Mayıs 2022, <https://tr.wikipedia.org/wiki/Python>

Wikipedia.com(2022). Spyder IDE, Erişim Tarihi: 20 Mayıs 2022, [https://en.wikipedia.org/wiki/Spyder_\(software\)](https://en.wikipedia.org/wiki/Spyder_(software))

Yu, J. ve Zhang, W. (2021). Face Mask Wearing Detection Algorithm Based on.

Worby, C. J. ve Chang, H. H. (2020). Face mask use in the general population and optimal resource allocation during the COVID-19 pandemic. *Nature Communications*, 11(1), 1–9.

ÖZGEÇMİŞ

Muratcan Laloğlu 18 Haziran 2002 tarihinde İstanbul'da doğdu. Ortaöğretimini Uğur Okullarında tamamlamış, yükseköğretimini ise Haliç Üniversitesi Bilgisayar Programcılığı bölümünde devam ettirmektedir.