ZONGULDAK BÜLENT ECEVİT ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ

MASKE DEDEKTÖRÜ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

DİPLOMA ÇALIŞMASI

Osman SIRAKAYA	Batuhan Buğra AKÇA	Deniz ERTEPE
192106206005	191106206005	192106206008

Haziran 2022

ZONGULDAK BÜLENT ECEVİT ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

MASKE DEDEKTÖRÜ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

DİPLOMA ÇALIŞMASI

Osman SIRAKAYA	Batuhan Buğra AKÇA	Deniz ERTEPE
192106206005	191106206005	192106206008

DANIŞMAN: Dr. Öğr. Üyesi Erkan ÇETİNER

ZONGULDAK

Haziran 2022

KABUL:

Osman SIRAKAYA, Batuhan Buğra AKÇA, Deniz ERTEPE tarafından hazırlanan "Maske Dedektörü" başlıklı bu çalışma jürimiz tarafından değerlendirilerek Zonguldak Bülent Ecevit Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü Diploma Çalışması olarak oy birliğiyle/oy çokluğuyla kabul edilmiştir. 03/06/2022

Danışman: Dr. Öğr. Üyesi Erkan ÇETİNER

Zonguldak Bülent Ecevit Üniversitesi, Müh. Fakültesi, Bilgisayar Müh. Bölümü

Üye: Dr. Öğr. Üyesi Umut KONUR

Zonguldak Bülent Ecevit Üniversitesi, Müh. Fakültesi, Bilgisayar Müh. Bölümü

Üye: Dr. Öğr. Üyesi Abdullah Oğuz KIZILÇAY

Zonguldak Bülent Ecevit Üniversitesi, Müh. Fakültesi, Bilgisayar Müh. Bölümü

ONAY:

Yukarıdaki imzaların, adı geçen öğretim üyelerine ait olduğunu onaylarım.

Dr. Öğr. Üyesi Abdullah Oğuz KIZILÇAY
Bilgisayar Mühendisliği
Bölüm Başkanı

"Bu tezdeki tüm bilgilerin akademik kurallara ve etik ilkelere uygun olarak elde edildiğini ve sunulduğunu; ayrıca bu kuralların ve ilkelerin gerektirdiği şekilde, bu çalışmadan kaynaklanmayan bütün atıfları yaptığımı beyan ederim."

Osman SIRAKAYA	Batuhan Buğra AKÇA	Deniz ERTEPE
192106206005	191106206005	192106206002

ÖZET

Diploma Çalışması

MASKE DEDEKTÖRÜ

Osman SIRAKAYA Batuhan Buğra AKÇA Deniz ERTEPE

Zonguldak Bülent Ecevit Üniversitesi
Mühendislik Fakültesi
Bilgisayar Mühendisliği Bölümü

Tez Danışmanı: Dr. Öğr. Üyesi Erkan ÇETİNER

Haziran 2022, 35 sayfa

Tüm dünya da olduğu gibi ülkemizi de saran Covid-19 salgınından korunmanın en etkili yollarından birinin maske takmak olduğu bilinmektedir ama özellikle vakaların arttığı yerlere bakılırsa bu artışın sebebinin tedbirsizlik olduğu görülmektedir. Maske Dedektörü kamu kurumlarında, özel sektörde, kafelerde ve hatta mobese kameralarında bile kullanılabilir ve tedbirin az olduğu yerler tespit edilebilir böylelikle doğru önlemler alınabilir. Program geliştirilirken kullanılan Python v3.8.7 ile genellikle Yapay Zekâ ve Makine Öğreniminde kullanılan Tensorflow kütüphanesi kullanılmıştır. Elimizde bulunan yaklaşık 4000 tane maskeli ve maskesiz olarak sınıflandırılan insan fotoğrafları eğitilerek bir model oluşturulmuştur. Bu model sayesinde insanların maskeyi yüzde oranında takıp takmadığını göstermektedir. Program geliştirilirken kullanılan diğer kütüphaneler ise Numpy, Keras, Imutils, OpenCv, Matplotlib, Scipy'dir.

Anahtar Kelimeler: Maske dedektörü, Yapay zekâ, Makine öğrenimi, Görüntü işleme

ABSTRACT

Dissertation Study

MASK DETECTOR

Osman SIRAKAYA **Batuhan Buğra AKÇA** **Deniz ERTEPE**

Zonguldak Bülent Ecevit University
Faculty of Engineering
Department of Computer Engineering

Thesis Advisor: Assist. Prof. Dr. Erkan ÇETİNER

June 2022, 35 pages

It is known that one of the most effective ways to protect ourselves from the Covid-19 epidemic that surrounds our country as well as the whole world is to wear a mask, but especially when we look at the places where the cases have increased, it is seen that the reason for this increase is imprudence. The Mask Detector can be used in public institutions, the private sector, cafes and even security cameras, and the places where the precautions are less can be detected, so that the right precautions can be taken. While developing the program, Python v3.8.7 and Tensorflow library, which is generally used in Artificial Intelligence and Machine Learning, were used. A model was created by training about 4000 masked and unmasked human photographs that we have. Thanks to this model, it shows whether people are wearing the mask by percentage. Other libraries used while developing the program are Numpy, Keras, Imutils, OpenCv, Matplotlib, Scipy.

Keywords: Mask detector, Artificial intelligence, Machine learning, Image processing

TEŞEKKÜR

Tez çalışmamız sırasında kıymetli bilgi, birikim ve tecrübeleri ile bize yol gösterici ve destek olan değerli danışman hocamız sayın Dr. Öğr. Üyesi Erkan ÇETİNER'e, ilgisini ve önerilerini göstermekten kaçınmayan, değerli vaktini bizler için ayıran kıymetli hocamız sayın Dr. Öğr. Görevlisi Oğuz ÖZBEK'e sonsuz teşekkür ve saygılarımızı sunarız. Teşekkürlerin az kalacağı diğer üniversite hocalarımızın da bize üniversite hayatımız boyunca kazandırdıkları her şey için ve bizi gelecekte söz sahibi yapacak bilgilerle donattıkları için hepsine teker teker teşekkürlerimizi sunuyoruz ve son olarak çalışmamızda desteğini ve bize olan güvenini bizden esirgemeyen arkadaşlarımıza ve bizi bu günlere sevgi ve saygı kelimelerinin anlamlarını bilecek şekilde yetiştirerek getiren ve bizden hiçbir zaman desteğini esirgemeyen ailelerimize sonsuz teşekkürler.

İÇİNDEKİLER

İÇİNDEKİLER (devam ediyor)

ŞEKİLLER DİZİNİ

No Sayfa

ÇİZELGELER DİZİNİ

No Sayfa

SİMGELER VE KISALTMALAR DİZİNİ

SİMGELER

∑ : Toplama fonksiyonu

ϴ : Bias

KISALTMALAR

ARDS **:** Acute Respiratory Distress Syndrome

AI **:** Yapay Zekâ

TV **:** Televizyon

BEÜ **:** Bülent Ecevit Üniversitesi

BSD **:** Berkeley Software Distribution

IBM **:** International Business Machines

RGB **:** Red, Green, Blue

YSA **:** Yapay Sinir Ağları

LVQ **:** Linear Vector Quantization

SOM **:** Self Organizing Maps

OCR **:** Optical Character Recognition

CPU **:** Central Process Unit

GPU **:** Graphics Processing Unit

DS-CNN **:** Derinlemesine Ayrılabilir Evrişim Sinir Ağı

BÖLÜM 1

GİRİŞ

2019 korona virüs hastalığı (Covid-19) pandemisinden önce, solunum yolu enfeksiyonlarının yayılmasını azaltmak için topluluk maskelerinin kullanımını destekleyen somut bir kanıt yoktu. Maskeler öncelikle kullanıcının viral damlacıkları yaymasını önlemeyi amaçlar COVID-19 ve diğer solunum yolu enfeksiyonları öncelikle öksürme, hapşırma, konuşma veya nefes alma yoluyla üretilen solunum aerosollerinin solunması yoluyla yayılır. Virüs solunum yollarında yayılır ve aşağı doğru göç eder ve zatürreye, akut solunum sıkıntısı sendromuna (ARDS) ve hatta ölüme neden olabilir. Devam eden pandemi ve hızla ortaya çıkan varyantlar, bu solunum hastalığını günlük bir manşet haline getirdi. Enfeksiyonun yayılmasını önlemek için insanların kişisel güvenlik ekipmanlarının bir parçası olarak ve halk sağlığı önlemi olarak yüz maskeleri kullanmaları önerilir. Bunun ışığında, günümüz dünyasında maske takan kişileri tespit edebilecek bir sistemin geliştirilmesi kritik önem taşımaktadır. Bu çalışma sayesinde mevcut problem çözülmüştür.

Bilim adamları, ortak alanlarda yüz maskelerinin kullanılmasını sağlamak için halka açık yerlerde otomatik yüz maskesi tanıma sistemleri kurmaya çalışmaktadır. COVID-19 salgınının ardından diğer araştırmacılar ortak alanlarda yüz maskelerini izlemek için kendi tekniklerini geliştirmektedir. Kalabalık ortamlarda kimsenin yüzünün görünmemesini sağlamak için görüntü işleme algoritmaları kullanan gözetleme sistemleri, kamusal alanların izlenmesinde kullanılmaktadır. Nesne tanımlama ve görüntü analitiği için derin öğrenmeye dayalı yaklaşımlar, yıllar içinde giderek daha popüler hale gelmiştir.. Geçmişteki araştırmaların çoğu, evrişimli sinir ağı modelleri kullanılarak yapılmıştır. Mevcut yüz maskesi algılama algoritmalarının maskeleri güvenilir bir şekilde tanımlayamadığı iki durum vardır. Tek bir görüntü veya video karesinde çok sayıda insan olduğunda, tüm yüzleri "maskeli ve maskesiz" tam olarak belirlemek zordur. Yüz maskesi algılama tekniklerinin mobil ortamda uygulanması için daha verimli ve doğru bir sınıflandırma yaklaşımının nasıl oluşturulacağı kilit bir husustur. Bununla birlikte, birkaç derin öğrenme modeli, değerlendirme adımlarında maliyetli ve zaman alıcıdır, bu da onları mobil bir ortamda yüz görüntüsü paradigmasında maske tespiti için uygun hale getirmemektedir. Mevcut yaklaşımın eksikliklerinin üstesinden gelmek için önerilen yöntem, yüz görüntülerinde maske tespiti için

Mobilenet ile Derinlemesine Ayrılabilir Evrişimleri kullanır. Derinlemesine ayrılabilir evrişim (DSC) ilk olarak önerilmiştir ve şimdi sınıflandırma görevleri için görüntü işlemede yaygın olarak kullanılmaktadır. Program bu teknolojiden yararlanarak tamamlandı. Yüz maskesi algılama problemi uygulanmadan önce, görüntülerin nasıl ele alınacağının anlaşılması gerekir. Görüntüler sadece kırmızı, yeşil ve mavi formatta bir renk koleksiyonudur. Bir insan olarak, içinde bir nesne veya şekil olan bir görüntü görüyoruz, ancak bilgisayar için bu sadece 0 ila 255 arasında renk değerlerine sahip bir dizidir. Bilgisayarın herhangi bir şeyi görme biçimi, insanın bir görüntüyü görme biçiminden farklıdır. Ama bu bizim için iyi haber çünkü görüntünün bir dizisini alırsak, o diziye herhangi bir algoritma uygulamak bizim için daha kolay hale gelir.

BÖLÜM 2

ÖN BİLGİ VE TEMEL KAVRAMLAR

2.1 YAPAY ZEKA

AI kısaltmasıyla da ifade edilen Yapay Zekâ, görevleri yerine getirmek için insan zekasını taklit eden ve topladığı bilgilerle kendisini kademeli olarak geliştirebilen sistemler veya makineler anlamına gelir. Yapay zekâ pek çok biçimde kendini gösterir. Örneğin:

- Sohbet robotları, müşterilerin sorunlarını daha hızlı bir şekilde anlamak ve daha verimli cevaplar vermek için yapay zekâdan yararlanır
- Akıllı asistanlar, zamanlamayı iyileştirmek için büyük kullanıcı tanımlı veri kümelerinden kritik bilgileri çekmek için yapay zekâdan yararlanır
- Öneri motorları kullanıcıların izleme alışkanlıklarına göre TV programları için otomatik önerilerde bulunabilir

Yapay Zekâ, herhangi bir özel biçim veya işlevden ziyade süper güçlendirilmiş düşünce ve veri analizi yeteneği ve süreciyle ilgilidir. Yapay zekâ üst seviye işleve sahip insan benzeri robotların dünyayı ele geçirmesine ilişkin görüntüler sunsa da yapay zekanın amacı insanların yerini almak değildir. Amaç insan yeteneklerini belirgin şekilde geliştirmek ve bunlara katkıda bulunmaktır. Bu nedenle oldukça değerli bir ticari varlıktır.

2.1.1 Görüntü İşlemede Yapay Zekâ

İnsan beyninin algıladığı, işlediği ve yorumladığı en önemli bilgi türü görsel verilerdir. Psikolog Albert Mehrabian'a göre; iletişimin %93'ü sözel olmayan verilerden sağlanıyor ve beyin görsel bilgileri yazılı verilere kıyasla 60.000 kat daha hızlı işliyor.

Görsel verinin bu gücü, ürün ve içerik pazarlama gibi birçok sahada pazarlama stratejisi olarak kullanımını da yaygınlaştırdı. Görsel materyallerin insan faktörünün olduğu her alanda etkili bir araç olması, gelişen teknoloji ile birlikte görsellerin oluşturulmasının ve yayılmasının kolaylaşması gibi faktörler milyonlarca görseli barındıran veri yığınları oluşturuyor. Elde edilen bu veri yığınları aynı zamanda yeni nesil teknolojiler için mükemmel bir veri tabanı sağlıyor.

Yapay zekanın da sahaya çıkışıyla, görsel verilerin içerdiği bilgiler sadece insan algısına hitap eden basit bir reklam aracı olmaktan çıkıp medikal tanı uygulamalarından malzeme bilimi

teknolojilerine, savunma sanayisindeki sensör sistemlerinden film endüstrisine kadar geniş ölçekteki teknolojilerin kaynağına dönüşüyor.

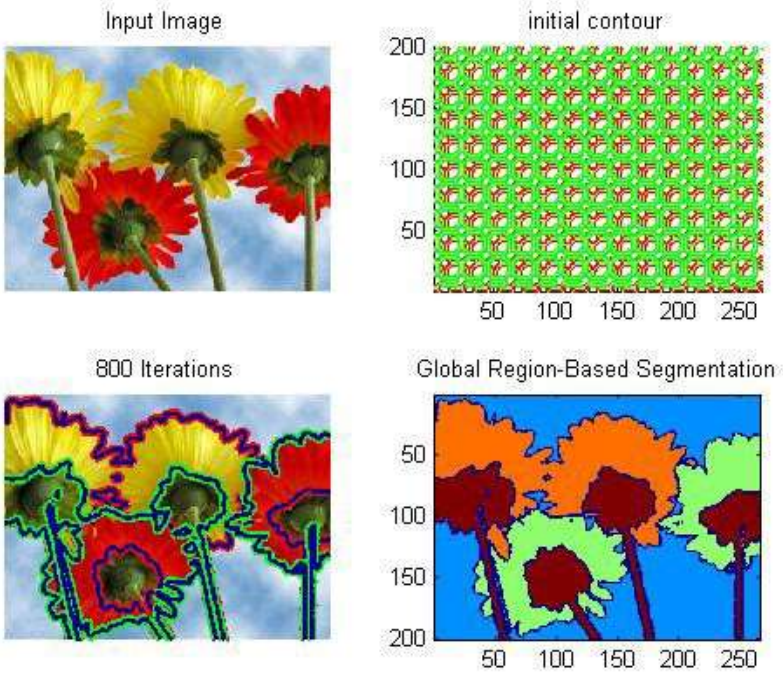
Görsel işleme ile elde edilmesi beklenen bilgiler fotoğraf üzerindeki bir kedinin ya da köpeğin etiketlenmesinden çok daha öteye gitmiştir. Beklentideki bu artışa neden olan faktörler, hem işlenebilecek veri sayısındaki artış hem de kullanılan algoritmaların gelişmesidir. Artık görsel işleme teknikleri ile kanserli hücre tanısı yapmak veya askeri sahalarda kullanılacak sensörler geliştirmek gibi hayati önem taşıyan sahalarda çalışabiliyoruz.

Görüntü işlemedeki bu ileri teknoloji, başta makine öğrenmesi olmak üzere, derin öğrenme ve yapay sinir ağları gibi yan başlıkların kullanımıyla mümkün olmaktadır. Yüz tanıma teknolojileri, sürücüsüz araçlar veya Google'ın kamera ile çeviri modu makine öğrenmesi tabanlı görüntü işleme sistemlerinin en basit örnekleri arasında. Google Translate çeviri uygulamasındaki görüntü işleme sistemini makine öğrenmesi ve yapay sinir ağları algoritmalarıyla oluşturuyor. Google Translate'in kamera ile çeviri modu, herhangi bir dilin harflerinin görüntü işleme ve makine öğrenimi ile veri tabanına kaydedilmesini sağlar. Kamera ile sözcüğü taradığımızda sistem veri tabanında kayıtlı harfleri birleştirerek kelimeleri ortaya çıkarır. Kelime ya da cümlenin ortaya çıkmasından sonra klasik Google Translate uygulamasında olduğu gibi kendi dil işleme algoritmasıyla kelimeleri çevirir. Bu modun çalışabilmesi için önemli olan nokta kamerayla okuttuğunuz kelimelerin sistem tarafından tanınmasıdır ki bu da makine öğrenmesi sayesinde olur.

Yapay zekâ, görüntü işlemede ileri teknoloji sağlamanın yanında işleme süresini de kısaltıyor. Askeri alanda kullanılan elektro-optik sensör sistemleri uzun yıllardan beri hareketli veya sabit hedeflerin belirlenmesinde kullanılıyordu. Günümüzde ise bu sistemler yapay zekâ ile güçlendirilerek hem daha hızlı hem de daha kesin hedef tespiti yapmayı sağlıyor.

2.1.2 Görüntü Bölütleme

Görüntü bölütleme görüntüyü kendini oluşturan bileşenlere ve nesnelere ayırma işlemidir. Uygulama konusuna göre görüntü bölütleme işleminin ne zaman durdurulacağı belirlenir. Aranan nesne veya bileşen görüntüden elde edildiği anda bölütleme işlemi durdurulur. Örneğin, havadan hedef tespitinde ilk olarak yol ve araçlar bütünü olan fotoğraf, yol ve araç olarak ayrılır. Her bir araç ve yol tek başına bu görüntünün bölünmüş parçalarını oluşturur. Hedef kriterlerine göre uygun olan alt resme ulaşıldığında görüntü bölütleme işlemi durdurulur.



Şekil 2.1 Görüntü Bölütleme

Görüntü bölütleme için görüntü eşiği oluşturma yöntemleri kullanılır. Bu sınırlama yöntemlerinde nesne pikselleri bir gri seviyesine arka plan ise farklı piksellere ayarlanır. Genellikle nesne pikselleri siyah ve arka plan beyazdır. Piksel farkından kaynaklanan bu ikili görüntüler gri ölçeğe göre değerlendirilerek görüntü ayrımı yapılır.

2.2 MAKİNE ÖĞRENMESİ

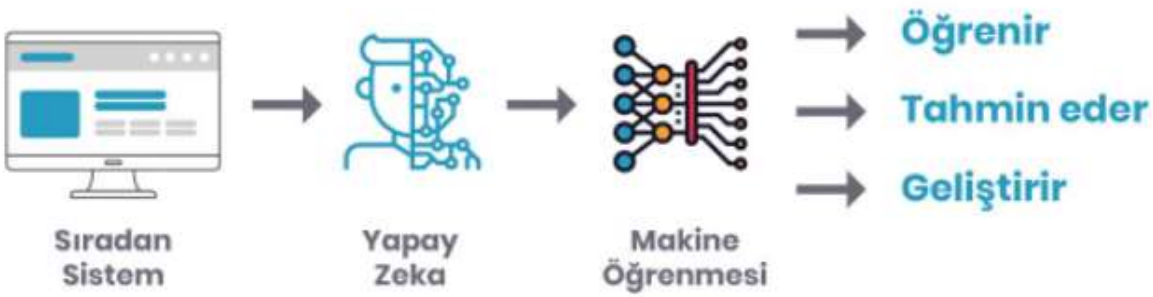
Makine öğrenmesi terimini ilk kez IBM'in içinden biri olan Arthur Samuel'in, dama oyunu ile ilgili olan araştırmasında kullandığı biliniyor. Uzman dama oyuncusu Robert Nealey, oyunu 1962 yılında bir IBM 7094 bilgisayarında oynamış ve bilgisayara karşı kaybetmiştir. Günümüzde mümkün olanlar düşünüldüğünde bu başarı neredeyse önemsiz gelebilir; ancak yapay zekâ alanında önemli bir dönüm noktası olarak görülür.

Önümüzdeki birkaç on yıl içinde, depolama ve işlem gücü etrafındaki teknolojik gelişmeler, Netflix'in tavsiye motoru veya sürücüsüz araçlar gibi günümüzde tanıdığımız ve sevdiğimiz bazı yenilikçi ürünleri mümkün kılacak.

Makine öğrenmesi, büyüyen veri birimi alanının önemli bir bileşenidir. İstatistiksel yöntemler kullanılarak, algoritmalar; sınıflandırmalar veya tahminler yapmak üzere eğitilir ve veri

madenciliği projelerinde temel içgörüleri ortaya çıkarmaktadır. Bu içgörüler, sonrasında uygulamalar ve işler dahilinde karar verme sürecini teşvik ederek, ideal anlamda önemli büyüme ölçütlerini etkiler. Büyük veri genişleyip büyümeye devam ederken, veri mühendislerine yönelik piyasa talebi artacak ve bu mühendislerin en önemli iş sorularını ve nihayetinde bu sorulara yanıt vermek için kullanılacak verilerin tanımlanmasına yardımcı olmaları gerekecek.

2.2.1 Makine Öğrenmesinin Çalışma Yapısı



Şekil 2.2 Makine öğrenmesi

Makine öğrenmesi algoritmaları, makinelerin öğrenmesine izin vererek onları daha akıllı hale getiren beyinler olarak tanımlanabilir. Bu algoritmaların düzenli olarak yeni verilere ve deneyimlere maruz bırakılması; sınıflandırma, tahmine dayalı modelleme ve verilerin analiziyle ilgili çeşitli görevler konusunda büyük işler başarılmasına olanak tanır.

Öğrenme süreci, verilerdeki kalıpları aramak için örnekler, doğrudan deneyim, talimatlar, gözlemler gibi eğitim verilerinin seçilen algoritmaya girilmesiyle başlar. Bu algoritmanın doğru çalışıp çalışmadığını test etmek için, yeni girdi verileri makine öğrenimi algoritmasına eklenir. Tahmin ve sonuçlar daha sonra kontrol edilir. Makinenin daha fazla veriyle beslenmesi, "öğrenmesine" neden olan algoritmaların etkinleştirilmesi ve elde edilen sonuçların iyileştirilmesi demektir. Tahmin beklendiği gibi değilse, algoritma, istenen çıktı bulunana kadar birçok kez yeniden eğitilir. Bu, makine öğrenimi algoritmasının sürekli olarak kendi başına öğrenmesini ve zaman içinde doğruluğu kademeli olarak artacak en uygun cevabı üretmesini sağlar. Algoritma öğrenme aşamasını geçtikten sonra, edindiği bilgileri farklı veri kümelerine dayalı benzer problemleri çözmek için kullanabilir.

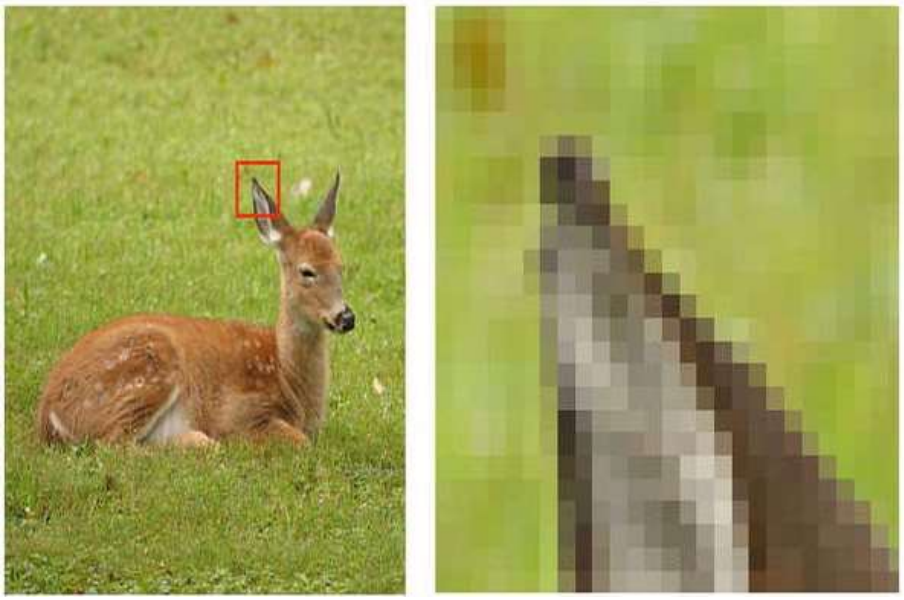
2.3 GÖRÜNTÜ İŞLEME

Görüntü işleme, elimizde bulunan görüntüden anlamlı ifadeler çıkarmamıza yarayan işlemler bütünüdür. Bu işlemler, görüntüyü oluşturan pikseller üzerinde gerçekleştirilecek matematiksel işlemler sayesinde gerçekleştirilir. Görüntü elde edildikten sonra, yapılması istenen göreve göre bir algoritma tasarlanır ve görüntü bu aşamalardan geçerek istenen görevi yerine getirir. Görüntü işleme teknikleri kullanılarak yapılabilecek işlemlere göz atacak olursak şunları görebiliriz;

- Görüntü üzerinde bulunan gürültülerin arındırılması ve temiz bir görüntü elde edilmesi
- Görüntü üzerinde bulunan ve insan algısının görmekte zorlandığı nesnelerin tespiti
- Görüntünün daha kaliteli bir hale getirilmesi
- Nesne takibi yapılması
- Görüntü üzerindeki farklı nesnelerin birbirinden ayırt edilmesi

2.3.1 Görüntü Nedir

Görüntü, sürekli gibi görünse de aslında parçalardan oluşan yapılardır. Bu parçalara ise piksel denmektedir. Her görüntü, iki boyutlu piksel dizisidir. Kameralarda görünen 1280 x 720, 1920 x 1080 gibi sayılar ise görüntünün çözünürlüğünü göstermektedir. Aslında bu sayılar, görüntü üzerindeki yatay ve dikey uzunluklarda bulunan piksel sayısını ifade etmektedir. Her bir piksel, RGB denilen ve üç ana renk olan kırmızı, mavi ve yeşil renklerinin yoğunluklarının ayarlanmasıyla elde edilen farklı renklerden oluşurlar. Bu sayede, farklı görüntüler elde edilir ve bu görüntüler bilgisayarların anlayabileceği şekilde ‘0’ ve ‘1’ ler ile ifade edilirler. Kameraların hayatımıza girmesi ile birlikte, içinde bulunduğumuz anı ölümsüzleştirmek amacıyla çok rahat bir şekilde telefonumuz ile görüntü veya video alıyor ve o dosyaları geçmişimize ayna tutması için hatıra olarak saklıyoruz. Aradan yıllar geçse bile eski resimlerimizi kurcalayıp birkaç dakikalığına da olsa geçmişe gidiyor ve eski anılarımızı hatırlıyoruz. Peki kamera vasıtasıyla aldığımız görüntüler sadece bu amaca mı hizmet ediyor? Tabii ki cevap “Hayır”. Günümüzde gelişen teknoloji ile alınan görüntüler, görüntü işleme teknikleri kullanılarak mühendislik ve bilim alanında etkin bir biçimde kullanılıyor.



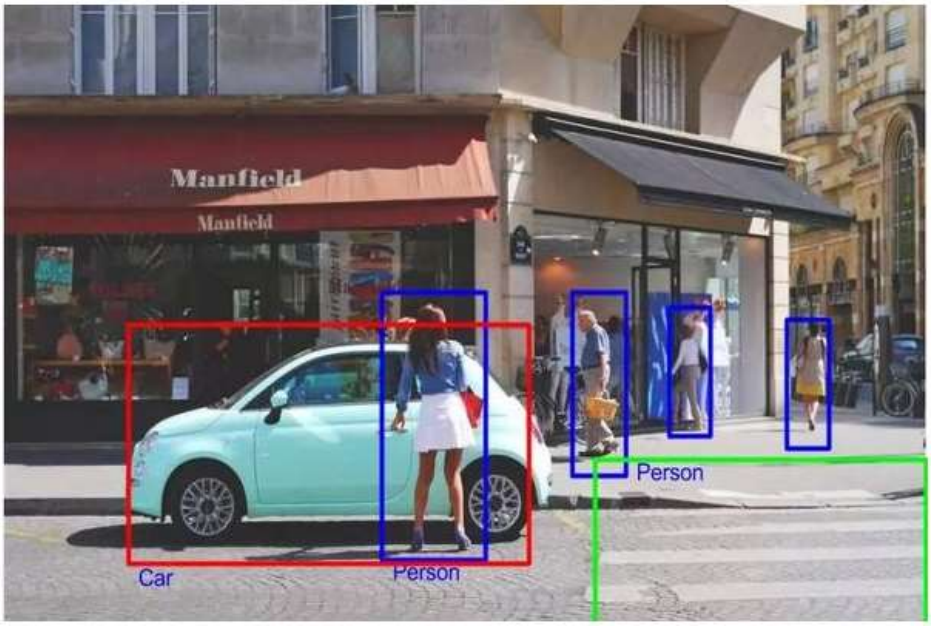
Şekil 2.3 Piksel Görünümü

2.3.2 Kullanım Alanları

Görüntü İyileştirme: Alınan görüntülerde gürültü olarak adlandırılan ve görüntü üzerinde bozulmalara neden olan bazı istenmeyen yapılar bulunabilir. Bu gürültülere örnek olarak tuz-biber gürültüsü, gauss gürültüsü, shot gürültüsü verilebilir. Görüntü işleme teknikleri içerisinde bulunan mean (ortalama) filtre, medium (ortanca) filtre gibi tekniklerle görüntü daha kaliteli ve gürültüsüz hale getirilebilir. Bu sayede görüntü üzerinde daha doğru sonuçlar elde edilecektir.

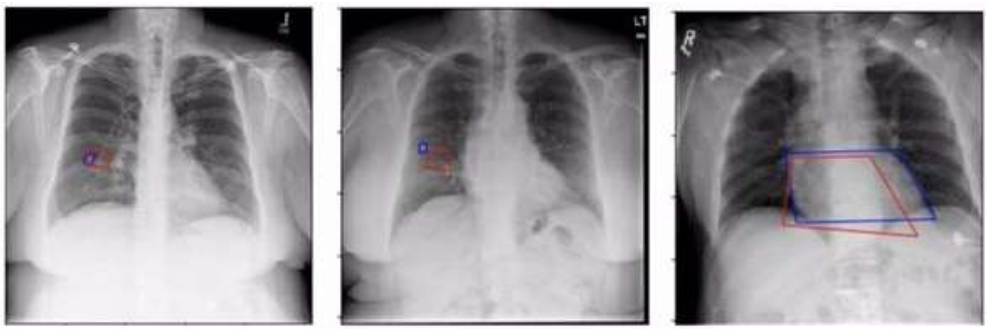
Şekil 2.4 Görüntü İyileştirme

Cisim Tanıma: Tespit edilecek cisme göre gerekli yöntemler ve algoritmalar kullanılarak görüntü üzerinden herhangi bir cismin tespiti ve takibi gerçekleştirilebilir. Örneğin yurt dışında birçok ülkede suçluların tespiti bu yöntem ile gerçekleştirilmektedir. Mevcut olan kamera düzeneklerinden alınan görüntüler üzerinden herhangi bir insanın tespiti sağlanabilir. Bunun dışında trafik alanında da kullanımı mevcuttur. Trafik içerisinde bulunan araçları sayabilir ve araçların hızı ölçülebilir. Bu sayede trafik yoğunluğu olma veya aşırı hız yapma gibi durumların tespiti gerçekleştirilip merkeze gerekli bildirimler yapılabilir.



Şekil 2.5 Cisim Tanıma

Sağlık Sektörü: Görüntü işleme teknikleri sayesinde birçok hastalığın teşhisi gerçekleştirilebilmektedir. Doğum öncesi fetüsün oluşumu ve takibi, tıbbi görüntülerin incelenmesi, şüpheli dokuların belirgin hale getirilip uzmanlara doğru tanı koyabilme olanağı tanıması, meme kanserinin erken teşhisi gibi alanlarda görüntü işleme teknikleri kullanılmaktadır. Bunların yanı sıra beyin görüntüleme, kemik şeklinin ve yapısının analizi, kanser tanısı koyma ve tümörü fark etme gibi işlemlerde tıp biliminde kullanılabilmektedir.



Şekil 2.6 Sağlık Sektörü

2.3.3 Görüntü İşleme Aşamaları

Görüntü işleme temel olarak beş aşamada incelenir.

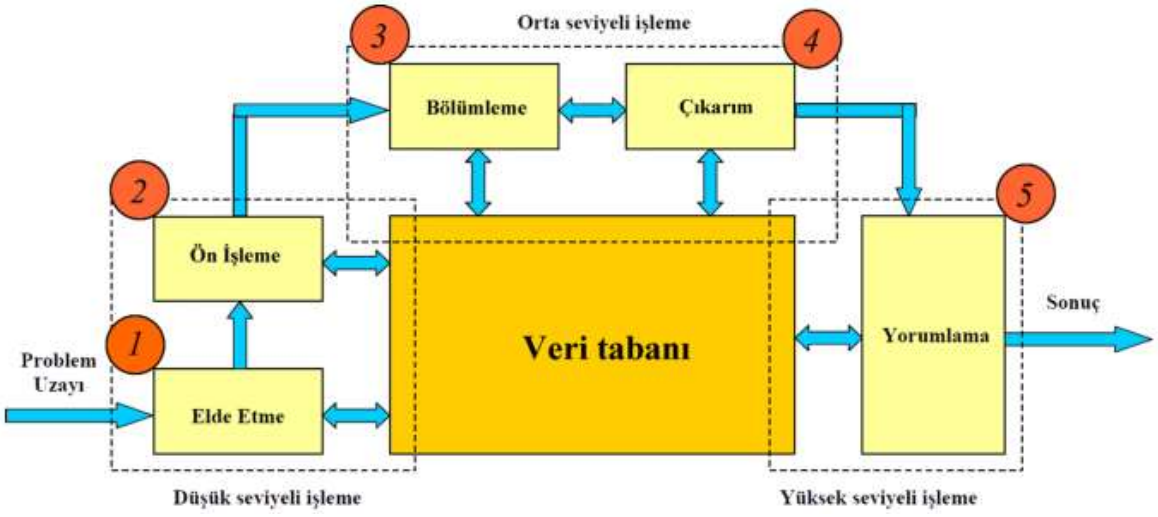
Elde Etme: Sayısal görüntü sayısal kamera ile elde edilir.

Ön İşleme: Sayısal görüntüyü kullanmadan önce bazı ön işlemlerden geçirmek gerekir. Bunlar; görüntü iyileştirme, görüntü onarma, görüntü sıkıştırma işlemleridir. Görüntünün elde edilmesi ve ön işleme tabi tutulması düşük seviyeli görüntü işleme olarak adlandırılır.

Bölümleme: Bu işlemin alt başlıkları; bölütleme, parçalara ayırma, segmentasyon olarak adlandırılır. Bu kısım görüntü işlemenin daha zor kısmıdır. Bu kısımda amaç görüntü içerisindeki nesne ve alanların değişik özelliklerinin tespit edilip birbirinden ayrılmasıdır. Görüntü içindeki nesne ve alan sınırlarının tespiti ile şekil üzerinde ham -işlenmemiş- bilgiler üretir. Eğer amaç görüntü içerisinde nesnelerin sınırlarını, kenarlarını, köşelerini bulmaksa bölümleme ile tespit edilir. Aynı zamanda görüntü içerisinde nesnenin kapladığı alan, renk gibi özellikler bölgesel bölümleme yöntemi ile elde edilir. Karakter veya örüntü tanıma gibi problemlerin çözümü için her iki yöntem de kullanılır.

Çıkarım: Elde edilen ham bilgilerin, istenilen farklı özelliklerin, ayrıntıların ön plana çıkarılmasıdır.

Yorumlama: Bu son kısımda artık bazı yapay zekâ algoritmaları ile işlenen görüntü içerisindeki farklı nesnelerin ve alanların sınıflara ayrılması ve etiketlendirilmesi yapılır.



Şekil 2.7 Görüntü İşleme Aşamaları

2.3.4 Piksel Üzerinde Yapılan İşlemler

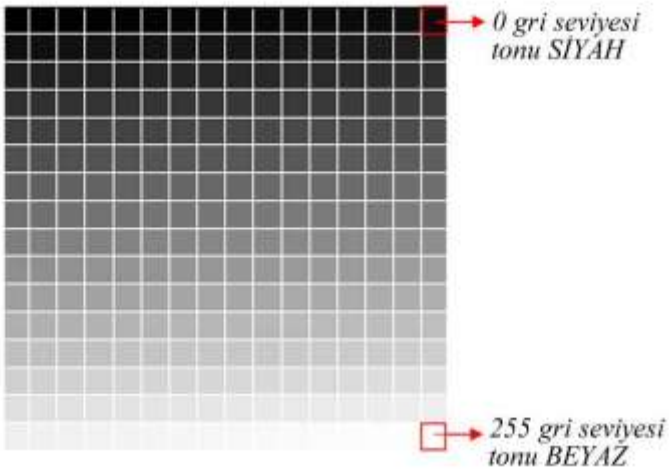
Görüntü işleme aşamasında yapılacak işlemlerin tümü görüntüyü oluşturan pikseller üzerinde yani bu piksellerin sahip olduğu renk bilgisi üzerinde gerçekleştirilir. Bu işlemler; nokta işlemleri, yerel (bölgesel) işlemler ve bütünsel işlemler olmak üzere üç grupta toplanabilir.

Nokta İşlemleri: Çıkış resminin bir pikselini oluşturmak için, giriş resminin bir pikseli üzerinde yapılan işlemlerdir. Kendisine komşu olan piksellerden bağımsız olarak pikselin kendi renk bilgisinin değiştirilmesiyle yapılır. Yani girdi görüntüsündeki her piksel üzerinde yapılan işlem çıktı görüntüsündeki karşılığı olan pikseli oluşturur.

Yerel (Local) İşlemler: Burada çıkış resmini oluşturacak bir noktanın rengi, giriş resminde bu noktanın komşularının renk özelliklerine de bağlıdır. Hangi komşuların rengine bağlı olacağı belirlenen maskenin büyüklüğüne bağlıdır. Bu maskeler, görüntüdeki tüm pikseller üzerinde kaydırılarak görüntünün filtrelenmesi sağlanır. Bu anlamda; görüntüdeki bulanıklaşmanın yok edilmesi, gürültünün temizlenmesi, kenar ve bölge özelliklerinin saptanması yerel işlemlere birer örnek olarak verilebilir. Özetle bu işlemde çıktı görüntüsündeki bir pikselin değeri girdi görüntüsündeki birçok pikselin değerine bağlıdır.

Bütünsel (Global) İşlemler: Bütünsel (global) işlemlerde resmin tüm piksellerinin renk özellikleri çıkış resminin oluşturulacak pikselini etkiler. Yani bu işlemde çıktı görüntüsündeki bir pikselin değeri girdi görüntüsündeki tüm piksellerin değerlerine bağlıdır.

Gri Görüntü: Sayısallaştırma işleminde, görüntü boyutlarının ve her bir pikselin sahip olabileceği parlaklık değerinin belirlenmesi gerekir. Sayısal görüntünün her bir pikselinin sahip olduğu parlaklık değeri gri seviyeler olarak adlandırılır. Her bir pikseldeki parlaklık değerinin kodlandığı bit sayısına göre gri seviye aralığı belirlenir. Gri seviye sınırlarında iki renk vardır, siyah ve beyaz. Bu ikisi arasında kodlanan görüntülere ise gri-ton (gray scale, monochromatic) görüntüler adı verilir. Uygulamada yaygın olarak kullanılan her bir piksel 8 bit ile kodlanmıştır. Bu tip görüntülerde her bir piksel 28 = 256 farklı gri ton karşılığı (parlaklık seviyesi) değerlerinden oluşur ve gri değer aralığı G = {0, 1, 2,…, 255}biçiminde ifade edilir. Kural olarak; 0 gri seviyesi siyah renge, 255 gri seviyesi ise beyaz renge ve bu değerler arasındaki gri seviyeler ise gri tonlara karşılık gelir. Şekil de N × M = 16 × 16 ‘lık bir ızgara üzerinde 256 farklı gri seviyenin gösterimi verilmiştir.



Şekil 2.8 Izgara Üzerinde 256 farklı gri seviyenin gösterimi

Renkli görüntülerin her bir pikseli bilgisayar ekranlarında 24 bit'lik veri olarak görüntülenir. Şöyle ki, her bir renk 8 bit (2.2.2.2.2.2.2.2=28=256) ile kodlanacağına göre üç renk (RGB) 3x8 = 24 bit ile kodlanacaktır. Bu durumda, RGB görüntülerin her bir pikseli 2^8. 2^8. 2^8= 2^{24} = 16.777.216 (yaklaşık 17 milyon) farklı renk değerine sahip olabilir ve bu üç rengin birleşiminin değer aralığı RGB = (0, 0, 0),…, (255, 255, 255) biçiminde gösterilir. Aşağıdaki tabloda bazı örnek renkler ve değerleri verilmiştir.

Çizelge 1.1 Renkler ve değerleri

Renk	R	G	B	Görünüm
Kırmızı	255	0	0	
Yeşil	0	255	0	
Mavi	0	0	255	
Beyaz	255	255	255	
Siyah	0	0	0	
Açık Gri	200	200	200	
Koyu Gri	100	100	100	
Sarı	255	255	0	
Turkuaz	0	255	255	
Eflatun	255	0	255	

2.4 YAPAY SİNİR AĞLARI

Yapay sinir ağları (YSA), insan beyninin özelliklerinden olan öğrenme yolu ile yeni bilgiler türetebilme, yeni bilgiler oluşturabilme ve keşfedebilme gibi yetenekleri, herhangi bir yardım almadan otomatik olarak gerçekleştirebilmek amacı ile geliştirilen bilgisayar sistemleridir

Yapay sinir ağları insan beyni örnek alınarak, öğrenme sürecinin matematiksel olarak modellenmesi sonucu ortaya çıkmıştır. Beyindeki biyolojik sinir ağlarının yapısını, öğrenme, hatırlama ve genelleme kabiliyetlerini taklit eder. Yapay sinir ağlarında öğrenme işlemi örnekler kullanılarak gerçekleştirilir. Öğrenme esnasında giriş çıkış bilgileri verilerek, kurallar koyulur.

Çizelge 1.2 Biyolojik ve yapay sinir ağlarının karşılaştırılması

Biyolojik Sinir Ağı	**Yapay Sinir Ağı**
Sinir Sistemi	Sinirsel Hesaplama Sistemi
Sinir	Düğüm (yapay sinir)
Sinaps	Sinirler arası bağlantı ağırlıkları
Dendrit	Toplama İşlevi
Hücre Gövdesi	Etkinlik (Aktivasyon) İşlevi
Akson	Sinir Çıkışı

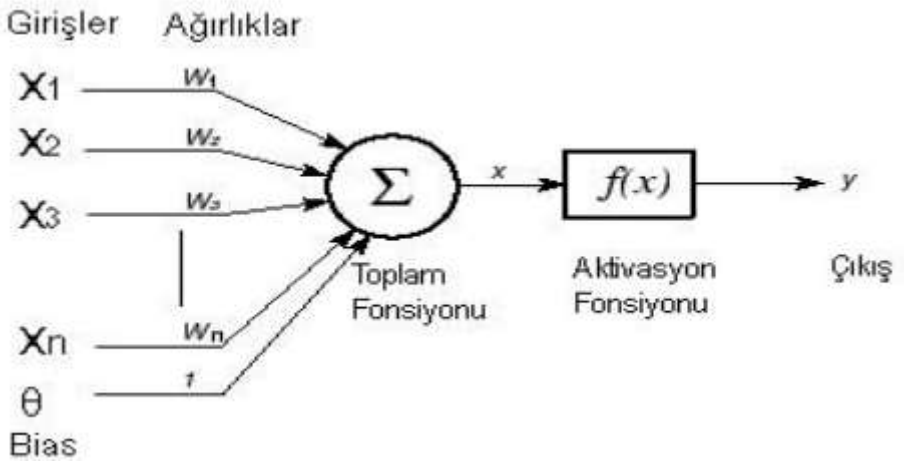
Yapay sinir ağlarının genel özelliklerini şu şekilde sıralamak mümkündür;

- Öğrenebilen yapılardır.
- Genelleme yapabilirler.
- Bilgiyi bir veri tabanında saklamak yerine ağın bütününe dağıtılmış şekilde saklarlar.
- Hata tolere etme yönleri gelişmiştir.
- Eksik bilgiyle çalışabilirler.

- Yapay Sinir ağında bir bilginin kaybolması birdenbire değil, kademeli bozulma gösterir.
- Görmedikleri örnekler hakkında çözüm önerisinde bulunabilirler.
- Örüntü tanıma ve sınıflandırma yapabilirler.

2.4.1 Yapay Sinir Hücresinin Temel Elemanları

Bir yapay sinir ağı, paralel işlem yapan, birbirine bağlı çok sayıda düğüm ya da yapay sinir de denebilen işlem elemanlarından oluşmaktadır.



Şekil 2.9 Yapay sinir hücresi

Girişler (X1, X2, …,Xn): Yapay sinir hücresine (düğüm de denmektedir) dışarıdan ya da kendisinden önceki düğümden gelen sayısal bilgilerdir.

Bias (θ): Bir düğümün yerel optimum noktalarına takılıp yanlış sonuç üretmemesi için girişlerle birlikte toplam fonksiyonuna girilen sayısal bir değerdir.

Ağırlıklar (W1, W2,…, Wn): Bir düğüme giren değerlerin ağdaki etkinliğini sağlayan değerdir. Başlangıçta rastgele değerler atanan bu değişkenler, ağın öğrenmesi sırasında negatif ya da pozitif değer alabilirler.

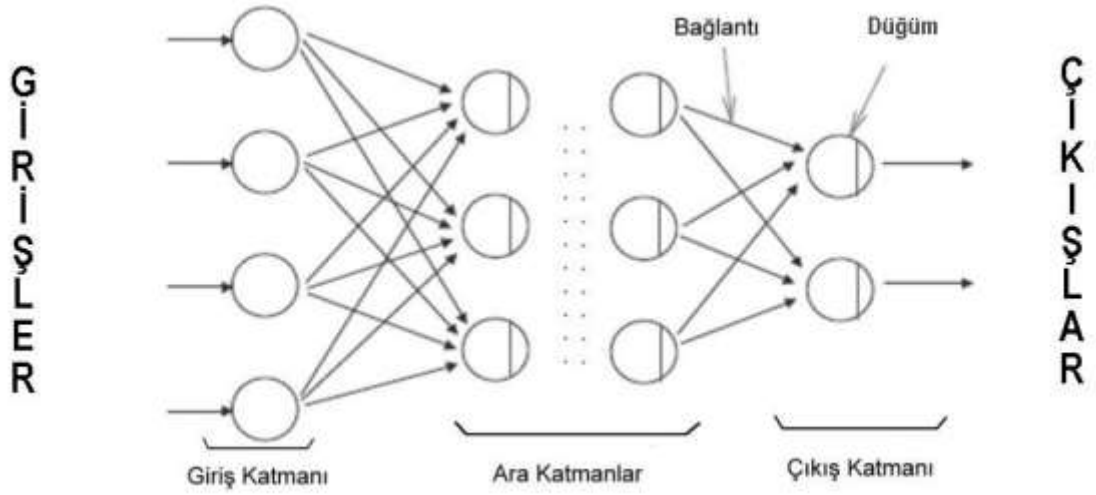
Toplam Fonksiyonu (∑): Bu fonksiyon, bir düğümdeki net girdiyi hesaplar. Bunu hesaplarken birçok farklı fonksiyon kullanılmakla birlikte yaygın olan girdiler ile ilgili ağırlık değerlerini çarparak, çıkan sonuçların eşik değeriyle toplanmasıyla bulunur.

Çizelge 1.3 Toplam fonksiyonları

Toplam $\text{Net} = \sum_{i=1}^{N} X_i * W_i$	Ağırlık değerleri girdiler ile çarpılır ve bulunan değerler birbirleriyle toplanarak Net girdi hesaplanır.
Çarpım $\text{Net} = \prod_{i=1}^{N} X_i * W_i$	Ağırlık değerleri girdiler ile çarpılır ve daha sonra bulunan değerler birbirleriyle çarpılarak Net Girdi Hesaplanır.
Maksimum $\text{Net} = \text{Max}(X_i * W_i)$	n adet girdi içinden ağırlıklar girdilerle çarpıldıktan sonra içlerinden en büyüğü Net girdi olarak kabul edilir.
Minimum $\text{Net} = \text{Min}(X_i * W_i)$	n adet girdi içinden ağırlıklar girdilerle çarpıldıktan sonra içlerinden en küçüğü Net girdi olarak kabul edilir.
Çoğunluk $\text{Net} = \sum_{i=1}^{N} Sgn(X_i * W_i)$	n adet girdi içinden girdilerle ağırlıklar çarpıldıktan sonra pozitif ile negatif olanların sayısı bulunur. Büyük olan sayı hücrenin net girdisi olarak kabul edilir.
Kumilatif Toplam $\text{Net} = \text{Net(eski)} + \sum_{i=1}^{N} X_i * W_i$	Hücreye gelen bilgiler ağırlıklı olarak toplanır. Daha önce hücreye gelen bilgilere yeni hesaplanan girdi değerleri eklenerek hücrenin net girdisi hesaplanır.

2.4.2 Yapay Sinir Ağının Yapısı

Yapay sinir ağları, çok sayıdaki düğüm ve bu düğümleri birbirine bağlayan ağırlıklardan meydana gelir. Yapay sinir ağına girilen sayısal bir bilgi, düğümler arası bağlantılar vasıtasıyla çıkışa kadar iletilir.



Şekil 2.10 Yapay sinir ağının yapısı

İlk yapay sinir ağının bulunmasından günümüze dek birçok yapay sinir ağı modeli ve bu modellerde kullanılacak birçok öğrenme stratejisi geliştirilmiştir. Düğümler arasındaki bilgi iletim yönüne göre; İleri Beslemeli (Feed Forward) ve Geri Beslemeli (Feed Backward) ya da Özyinelemeli (Feedback ya da Recurrent) iki temel ağ yapısından söz etmek mümkündür. Girişler ağa giriş katmanından girer, ağın çıktısının verildiği katman olan çıkış katmanına

doğru ileri yönlü yayılır. Geri yayılımlı ağlarda en az bir düğüm kendisinden önceki katmana bilgi gönderir. Yapay sinir ağlarının bir diğer özelliği öğrenebilmeleridir. Bunu da belirlenen öğrenme stratejisine göre düğümler arasındaki bağlantı değerlerini değiştirerek yaparlar. Temelde, Öğretmenli Öğrenme (Supervised Learning), Öğretmensiz Öğrenme (Unsupervised Learning) ve Takviyeli Öğrenme (Reinforcement Learning) olmak üzere üç çeşit öğrenme stratejisi mevcuttur. Öğretmenli öğrenmede, giriş setiyle birlikte çıkış seti ağa birlikte verilir ve ağın istenen çıkışı üretmesi beklenir. Bu tip ağlara örnek olarak Çok Katmanlı Algılayıcı veya ADALİN/MADALİN ağlarını vermek mümkündür. Öğretmensiz öğrenmede, girişler ağa verilir, ağdan herhangi bir çıkış istenmez, ağ bu girilen nesneleri ortak özelliklerine göre gruplara ayırır. Çıkışta bu grupların etiketlenmesi gerekmektedir. Bu tip ağlara Hebbian Öğrenme, Yarışmacı Öğrenme (Competitive Learning) ya da Özörgütlemeli Harita Ağları (Self-Organizing Map -SOM) örnek verilebilir. Takviyeli öğrenmede ise, giriş seti ağa verilir, ağın verdiği çıkış her seferinde öğretmene sorulur, öğretmen ağın doğru ya da yanlış yaptığını ağa bildirir. Burada ağ, bir nevi ödül-ceza sistemine göre çalışır. Bu tip ağlara LVQ (Linear Vector Quantization) ağları örnek verilebilir.

BÖLÜM 3

KULLANILAN TEKNOLOJİLER

3.1 PYTHON

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.

3.1.1 Tensorflow

Tensorflow 1.15.2 Yazılım sektöründe son yılların en popüler ve önemli konusu yapay zekâ diyebiliriz sanırım. Bu alanda makine öğrenmesi, derin öğrenme gibi konularda birçok gelişmeyi duyuyor ve okuyoruz. Derin öğrenme alanında en çok kullanılan kütüphanelerden birisi Tensorflow. Google tarafından geliştirilen açık kaynaklı Tensorflow ile derin öğrenme destekli yapay zekâ uygulamaları geliştirebiliyoruz. Açık kaynak kodlu bir deep learning (derin öğrenme) kütüphanesidir. Esnek yapısı sayesinde, tek bir API ile platform fark etmeksizin hesaplamaları, bir veya birden fazla CPU, GPU kullanarak deploy etmenize olanak sağlar. Temelinde Python kullanılarak geliştirilen bu framework, günümüzde Python ‘ın yanı sıra C++, Java, C#, Javascript ve R gibi birçok dili desteklemektedir.

3.1.2 Keras

Keras 2.3.1, neredeyse her tür derin öğrenme modelini tanımlamak ve eğitmek için uygun bir yol sağlayan Python için bir derin öğrenme kütüphanesidir. Keras, Tensorflow, Theano ve CNTK üzerinde çalışabilen Python ile yazılmış bir üst düzey sinir ağları API’sıdır**.** İçerdiği çok fazla işlevsel fonksiyon sayesinde Keras kolayca bir derin öğrenme modeli oluşturmamızı ve onu eğitmemizi sağlıyor. Bu nedenle derin öğrenmeye yeni başlayanlara önerilen kütüphanelerin başında Keras geliyor.

3.1.3 Numpy

NumPy 1.18.2, dizilerle çalışmak için kullanılan bir Python kütüphanesidir. Ayrıca doğrusal cebir, fourier dönüşümü ve matrisler alanında çalışmak için de gerekli işlevlere sahiptir. NumPy, 2005 yılında Travis Oliphant tarafından oluşturulmuştur. Açık kaynak kodlu bir projedir ve özgürce kullanabilirsiniz. NumPy, Numeriacal Python (Sayısal Python)'un kısaltmasıdır.

3.1.4 Opencv

Opencv 4.2.0 (Open Source Computer Vision) açık kaynak kodlu görüntü işleme kütüphanesidir. 1999 yılında İntel tarafından geliştirilmeye başlanmış daha sonra Itseez, Willow, Nvidia, AMD, Google gibi şirket ve toplulukların desteği ile gelişim süreci devam etmektedir. İlk sürüm olan Opencv alfa 2000 yılında piyasaya çıkmıştır. İlk etapta C programlama dili ile geliştirilmeye başlanmış ve daha sonra birçok algoritması C++ dili ile geliştirilmiştir. Open source yani açık kaynak kodlu bir kütüphanedir ve BSD lisansı ile altında geliştirilmektedir. BSD lisansına sahip olması bu kütüphaneyi istediğiniz projede ücretsiz olarak kullanabileceğiniz anlamına gelmektedir. Opencv platform bağımsız bir kütüphanedir, bu sayede Windows, Linux, FreeBSD, Android, Mac OS ve iOS platformlarında çalışabilmektedir. C++, C, Python, Java, Matlab, EmguCV kütüphanesi aracılığıyla da Visual Basic.Net, C# ve Visual C++ dilleri ile topluluklar tarafından geliştirilen farklı wrapperlar aracılığıyla Perl ve Ruby programlama dilleri ile kolaylıkla OpenCV uygulamaları geliştirilebilir. OpenCV kütüphanesi içerisinde görüntü işlemeye (image processing) ve makine öğrenmesine (machine learning) yönelik 2500'den fazla algoritma bulunmaktadır. Bu algoritmalar ile yüz tanıma, nesneleri ayırt etme, insan hareketlerini tespit edebilme, nesne sınıflandırma, plaka tanıma, üç boyutlu görüntü üzerinde işlem yapabilme, görüntü karşılaştırma, optik karakter tanımlama OCR gibi işlemler rahatlıkla yapılabilmektedir.

3.1.5 Matplotlib

Matplotlib 3.2.1; veri görselleştirmesinde kullandığımız temel Python kütüphanesidir. 2 ve 3 boyutlu çizimler yapmamızı sağlar. Matplotlib genelde 2 boyutlu çizimlerde kullanılırken, 3 boyutlu çizimlerde başka kütüphanelerden yararlanılır.

3.1.6 Scipy

Scipy 1.4.1; Scipy, mühendislerin Python ‘da algoritma geliştirme sürecine girmelerine izin veren genel amaçlı algoritma kaynaklarını da içeren bir dizi araçta bulunan ücretsiz bir açık kaynaklı Python kütüphanesidir. Scipy, Berkeley Software Distribution açık kaynak lisansı altında geliştiricilerin ve diğerlerinin makine öğrenimi hedeflerine ulaşmalarına yardımcı olan kendi kaynak paketine sahiptir. Makine öğrenimi mühendisleri, makine öğrenimi gelişimi için algoritmaların oluşturulmasına ve geliştirilmesine yardımcı olmak için SciPy'yi birçok farklı şekilde kullanır. SciPy'nin ana kullanımlarından biri modüler yaklaşımı ile ilgilidir. Algoritma optimizasyonu, entegrasyon, lineer cebir ve sinyal işleme ile ilgili modüller ile SciPy, makine öğrenimi projeleri için birçok faydaya sahiptir. Görselleştirme hedeflerine ulaşmak için matplotlib gibi diğer araçlarla da çalışır. Genel olarak, bu araçların tümü, insan karar vericilerinin anlayış geliştirmelerine ve rafine bir algoritma yaklaşımı gösteren makine öğrenimi çıktıları üzerinde çalışmasına yardımcı olmak için birlikte çalışır.

BÖLÜM 4

YÖNTEM

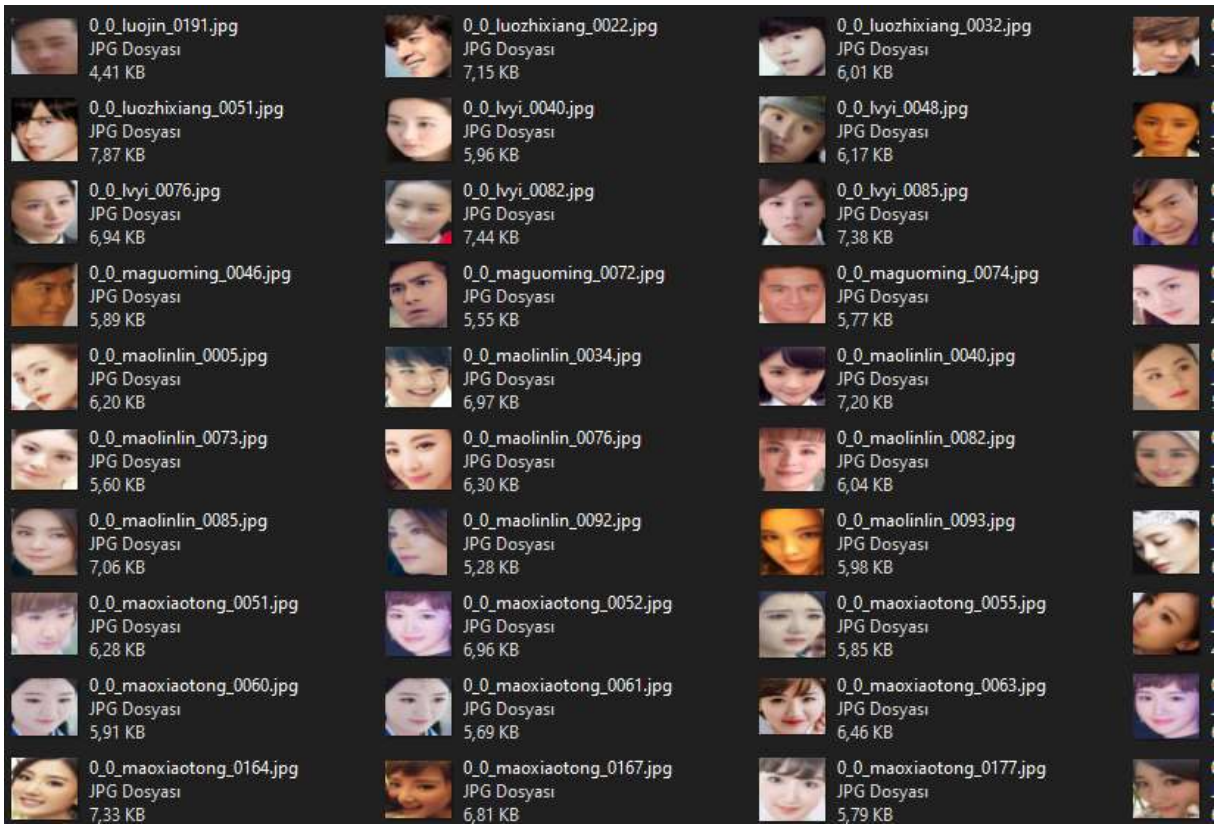
4.1 VERİ SETİ

Uygulamada kullanılacak veri setinde bulunan yaklaşık 4000 adet, 2000 tanesi maskeli, 2000 tanesi maskesiz olarak sınıflandırdığımız verimizi eğiterek bir model oluşturulmuştur. Bu modele göre kullanıcının maske takıp takmadığını tespit etmekledir.

4.1.1 Veri toplama ve sınıflandırma

Elimizde ne kadar çok veri (fotoğraf) olursa modelimiz o kadar doğru sonuç vereceğinden ötürü hazır veri seti kullandık çalışmamızda. İnternetten çektiğimiz veri setimizden bazı görüntüler **Şekil 4.1** ve **Şekil 4.2'**de verilmiştir.

Şekil 4.1 Maskeli veri seti



Şekil 4.2 Maskesiz veri seti

BÖLÜM 5

KAYNAK KODLAR

5.1 UYGULAMA

Derin öğrenme modellerinin hesaplama karmaşıklığını azaltmak için etkili bir yöntem olan MobileNet mimarisine dayalı Derinlemesine Ayrılabilir Evrişimli Sinir Ağı kullandık. Her darbede (24) bağımsız olarak gerçekleştirilen uzaysal evrişim ile 1 × 1 evrişim çıkış düğümünden oluşur Evrişim katmanının çıktısını, sonuçta 1 boyutlu bir maksimum havuzla Rektifiye Edilmiş Doğrusal Birim etkinleştirme işlevine besleme olarak kullandık. Birinci evrişim katmanının filtreleme boyutu ve derinliği 60'a ayarlanırken havuz katmanının filtre boyutu 2 adım sayısıyla 20'ye sabitlenir.

Tam bağlantılı katman girişi için, sonraki evrişim işleminin sonucu altı adıma düzleştirilir. Maksimum havuzlama katmanından bir giriş elde ettikten sonra, evrişim katmanı, maksimum havuzlama katmanının karmaşıklığının %10'una sahip olan çeşitli boyutlarda bir filtre kullanır. Tamamen bağlı katman girişi için sonuç yumuşatılır. Yukarıda bahsedilen tasarıma göre, tamamen bağlantılı katman, 1.001 nöron içerir. Hiperbolik tanjant işlevi, geçerli katmandaki doğrusal olmayanlığı temsil eder. Karşılık gelen hedef etiketlerin olasılığını hesaplamak için Softmax işlevi kullanılır. Potansiyel log-olabilirlik amaç fonksiyonlarını azaltmak için stokastik gradyan iniş optimizasyon yaklaşımı kullanıldı. Her işlevsel haritanın matris açıklaması, düzleştirme katmanı aracılığıyla bir vektöre dönüştürülür. Aşırı takma olasılığını ortadan kaldırmak için havuz katmanının üstüne birkaç bırakma monte edilir. Önerilen sistem, evrişim katmanları tarafından oluşturulan özellik haritalarını toplayan ve bilgi işlem maliyetlerini azaltan bir maksimum havuzlama katmanı içerir. Derinlemesine Ayrılabilir Evrişim Sinir Ağı'nın (DS-CNN) çalışması için, işlev eşlemelerinin hacmi, boyutlarıyla birlikte her zaman küçültülmelidir. Önerilen model konfigürasyonunda, son katman, yüz görüntülerini verimli bir şekilde sınıflandırmak için bir Softmax sınıflandırıcının eşlik ettiği tamamen bağlantılı bir katmandır. Önerilen MobileNet yönteminde derinlikle ayrılabilir evrişimler kullanılır. Geleneksel evrişimleri kullanan aynı sistem için 14.362'nin aksine, sistemimizdeki toplam öğrenilebilir parametre sayısı 6.844'tür.

5.1.1 Ana Fonksiyon

```
# gerekli paketler
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
```

```
def detect_and_predict_mask(frame, faceNet, maskNet):
# çerçevenin boyutlarını yakalama ve ondan bir frame oluşturma kısmı
    (h, w) = frame.shape[:2]
    blob = cv2.dnn.blobFromImage(frame, 1.0, (224, 224),
    (104.0, 177.0, 123.0))
    # ağ üzerinden yüz algılamalarını alıyoruz
    faceNet.setInput(blob)
    detections = faceNet.forward()
#   print(detections.shape) Çerçeve koordinatları gösteriyor
```

```
# yüz listemizi, bunlara karşılık gelen konumları ve yüz maskesi
ağımızdaki tahminlerin listesini başlatma kodu
faces = []
locs = []
preds = []
```

```
# Yüz algılama kontrol kısmı
    for i in range(0, detections.shape[2]):
        confidence = detections[0, 0, i, 2]
```

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

```
 # sınırlayıcı kutuların çerçevenin boyutları dahilinde olmasını
sağlayan kod
            (startX, startY) = (max(0, startX), max(0, startY))
            (endX, endY) = (min(w - 1, endX), min(h - 1, endY))
```

```
# opencv BGR görüntü formatını kullanır
# BGR görüntüsünü RGB'ye dönüştürmek için cvtColor() metodunu
kullanıyoruz
            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)

            # yüzü ve sınırlayıcı kutuları ilgili listelerine ekleme
            faces.append(face)
            locs.append((startX, startY, endX, endY))
```

```
# En az bir yüz algıladığında tarama yapmasını istiyoruz
    if len(faces) > 0:
        faces = np.array(faces, dtype="float32")
        preds = maskNet.predict(faces, batch_size=32)
    return (locs, preds)
```

```
# serileştirilmiş yüz dedektörü modelimizi diskten yükle
prototxtPath = r"face_detector\deploy.prototxt"
weightsPath =
r"face_detector\res10_300x300_ssd_iter_140000.caffemodel"
faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)

# yüz maskesi dedektör modelini diskten yükle
maskNet = load_model("mask_detector.model")

# kamera açma
print("[Bilgi] Sistem Başlatılıyor...")
vs = VideoStream(src=0).start()
```

```
while True:
    # video akışındaki çerçeveyi 800 piksel genişliğinde
boyutlandırır
    frame = vs.read()
    frame = imutils.resize(frame, width=800)

    # çerçevedeki yüzleri algılar ve maske takıp takmadığını
belirler
    (locs, preds) = detect_and_predict_mask(frame, faceNet, maskNet)
```

```
# tespit edilen yüz konumları ve bunlara karşılık gelen konumlar
üzerinde döngü kodu
    for (box, pred) in zip(locs, preds):
        # sınırlayıcı kutuyu ve tahminleri paketinden çıkarma kodu
        (startX, startY, endX, endY) = box
        (mask, withoutMask) = pred
```

```
# Sınırlayıcı kutuyu ve metni çizmek için kullanacağımız sınıf
etiketini ve rengini belirleme kodu
     label = "Maskeli" if mask > withoutMask else "Maskesiz"
     color = (0, 255, 0) if label == "Maskeli" else (0, 0, 255)
     label = "{}: {:.2f}%".format(label, max(mask, withoutMask)*100)

#etiket ve sınırlayıcı kutu dikdörtgenini çıktı karesinde
görüntüleme kodu
       cv2.putText(frame, label, (startX, startY - 10),
       cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
       cv2.rectangle(frame, (startX, startY), (endX, endY), color,2)
```

```
# çıktı çerçevesini göster
    cv2.imshow("Dedektor", frame)
    key = cv2.waitKey(1) & 0xFF
# "q" tuşuna basıldıysa döngüden çıkın
    if key == ord("q"):
        break
# program sonu
cv2.destroyAllWindows()
vs.stop()
```

5.1.2 Model Eğitme

```
# gerekli paketler
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import MobileNetV2
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
```

```
# başlangıç öğrenme oranını, eğitilecek dönem sayısını belirleyen
parametreler.
INIT_LR = 1e-4
EPOCHS = 20 # yapay sinir ağı eğitimimizin süresini etkiler
BS = 32 # ancak kesinliğinide arttırır
```

```
DIRECTORY = r"C:\Users\osman\OneDrive\Masaüstü\Maske
Dedektörü\dataset"
CATEGORIES = ["with_mask", "without_mask"]

# veri kümesi dizinimizdeki görüntülerin listesini alın, ardından
veri listesini (yani görüntüler) ve sınıf görüntülerini başlatın
print("[Bilgi]veriler yükleniyor...")
data = []
labels = []
```

```
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)
```

```
# etiketlerde tek sıcak kodlama gerçekleştirin
lb = LabelBinarizer()
labels = lb.fit_transform(labels)
labels = to_categorical(labels)

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

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

```
# veri büyütme için eğitim görüntü oluşturucusunu oluşturun
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")
```

```
# MobileNetV2 ağını yükleyin, baş FC katman setlerinin kapalı
kalmasını sağlayın
baseModel = MobileNetV2(weights="imagenet", include_top=False,
    input_tensor=Input(shape=(224, 224, 3)))
```

```
# temel modelin üstüne yerleştirilecek modelin başını oluşturun
headModel = baseModel.output
headModel = AveragePooling2D(pool_size=(7, 7))(headModel)
headModel = Flatten(name="flatten")(headModel)
headModel = Dense(128, activation="relu")(headModel)
headModel = Dropout(0.5)(headModel)
headModel = Dense(2, activation="softmax")(headModel)
```

```
#kafa FC modelini temel modelin üstüne yerleştirin (bu, eğiteceğimiz
gerçek model olacaktır)
model = Model(inputs=baseModel.input, outputs=headModel)
```

```
# temel modeldeki tüm katmanlar üzerinde döngü yapın ve ilk eğitim
sürecinde güncellenmemeleri için onları dondurun
for layer in baseModel.layers:
    layer.trainable = False
```

```
# modelimizi derleyin
print("[Bilgi] Model Derleniyor...")
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
model.compile(loss="binary_crossentropy", optimizer=opt,
    metrics=["accuracy"])
```

```
# ağın başını eğitmek
print("[Bilgi] kafa eğitimi...")
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)
```

```
# test setiyle ilgili tahminlerde bulunun
print("[Bilgi] Tahmin oluşturuluyor ...")
predIdxs = model.predict(testX, batch_size=BS)
```

```
# test setindeki her bir görüntü için, karşılık gelen en büyük
tahmin edilen olasılığa sahip etiketin dizinini bulmamız gerekir
predIdxs = np.argmax(predIdxs, axis=1)
```

```
# güzel biçimlendirilmiş bir sınıflandırma raporu gösterin
print(classification_report(testY.argmax(axis=1), predIdxs,
    target_names=lb.classes_))
```

```
print("[Bilgi] Maske Dedektörü kayıt ediliyor...")
model.save("mask_detector.model", save_format="h5")
```

```
# eğitim grafiği çıkarır
N = EPOCHS
plt.style.use("ggplot")
plt.figure()
plt.plot(np.arange(0, N), H.history["loss"], label="train_loss")
plt.plot(np.arange(0, N), H.history["val_loss"], label="val_loss")
plt.plot(np.arange(0, N), H.history["accuracy"], label="train_acc")
plt.plot(np.arange(0, N), H.history["val_accuracy"],
label="val_acc")
plt.title("Training Loss and Accuracy")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend(loc="lower left")
plt.savefig("plot.png")
```

BÖLÜM 6

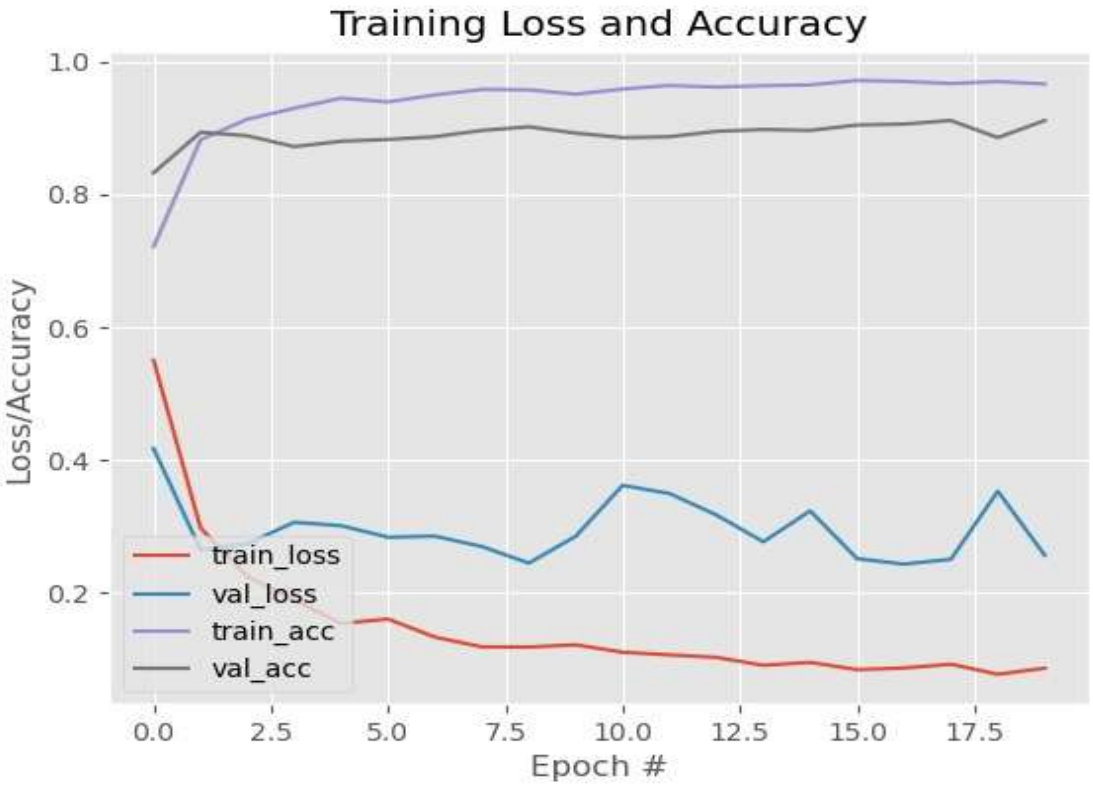
SONUÇ VE TARTIŞMA

Covid-19 günümüzü hızla etkiledi. Dünya ticaretini bozan ve günlük yaşamımızı etkileyen bir duruma geldi. Günlük yaşamımız da hastalıktan korunmak için maske takmak zorunlu hale geldi. Covid-19, klinik maske takmak çok gerekli, bu halk maskeyi takıp takmayacağını bilmeli kaynak kontrolü veya CO'dan kaçınma için VID-19 potansiyel maske kullanımını ilgili çekici noktaları azaltmada yatmaktadır. Sırasında zararlı bir kişiden gelebilecek riske karşı savunmasızlık "ön-semptomatik" dönem ve ayrıklığın damgalanması virüsün yayılmasını engellemek için maske takan kişiler, DSÖ tıbbi maskelere ve solunum cihazlarına öncelik verilmesini gerekli görmüştür. Bu nedenle yüz maskesi algılama küresel toplamda çok önemli bir görev haline gelmiştir.

Yüz maskesi algılama dahil yerini tespit etmede yüz ve ardından üzerinde bir maske olup olmadığının belirlenmesi. Sorun, genel nesne ile yakın akrabadır nesnelerin sınıflarını algılamak için algılama, yüz tanımlama olarak belirli bir grubu ayırt etmekle ilgilenir. Varlıklar yani yüz gibi çok sayıda uygulamaya sahiptir otonom sürüş, eğitim, gözetim vb. Bildirinin geri kalanı şu şekilde düzenlenmiştir yüz maskesi ile ilgili çalışmaları araştırır, kullanılan veri setinin yapısını tartışmaktadır. Dahil edilen paketlerin ayrıntılarını sunar. Yüz algılama yönteminde, bir yerden bir yüz algılanır içinde birçok özelliği olan resim, yüz tanıma araştırması, ifade tanıma gerektirir. Yüz izleme ve poz tahmini yalnız bir görüntü verildiğinde kişiler resimlerden yüzü tanımlamaktadır. Yüz algılama zor bir iştir çünkü yüzler değişir boyut, şekil, renk vb. değişmez. Sentezlenmiş sıradan yüzler birkaç yanlış ifade geri kazanılabilir ve yüz ipuçlarının üstünlüğü büyük ölçüde hafifletilir. Bildirilen çalışmaya göre bilgisayardaki evrişimli sinir ağı vizyonun boyutuyla ilgili katı bir kısıtlamayla birlikte gelir. Burada görevin ana zorluğu yüzü tespit etmektir. Görüntüden doğru bir şekilde ve ardından üzerinde bir maske olup olmadığını belirler. Gözetim görevini yerine getirmek için önerilen yöntem ayrıca hareket halindeki bir maske ile birlikte bir yüzü de algılamalıdır.

Şekil 6.1'deki görselde Machine Learning'i kullanırken, modelimizin ne kadar iyi performans gösterdiğini bilmemizi sağlayan farklı ölçümler var. Ancak bu önlemler ne anlama geldiklerini, nasıl yorumlanabileceklerini veya tam olarak ne olduklarını karıştırabilir. Bunu bilerek, modeller hakkında daha fazla bilgi çıkarırabilir.

Bu eğitimde, kayıp ve doğruluk üzerine odaklanılmıştır. Her ikisi de modelleri eğitirken dikkate alınması gereken temel değerlerdir.



Şekil 6.1 Kayıp ve Doğruluk

KAYNAKLAR

Udemy Derin Öğrenme ile Görüntü İşleme: Python, OpenCV ve Keras (08.01.2022)

URL-1 <https://www.udemy.com/course/derin-ogrenme-ile-goruntu-isleme-python-opencv-ve-keras/ >, Ziyaret tarihi 11.02.2022

Tensorflow ile görüntü işleme stackoverflow. (01.01.2020)

URL-2 < https://www.veribilimiokulu.com/tensorflow-lite-derin-ogrenme/ >, Ziyaret tarihi 11.02.2022

Python opencv nesne tanıma. (18.03.2018)

URL-3 < https://github.com/mesutpiskin/computer-vision-guide/blob/master/docs/14-nesne-tespiti.md >, Ziyaret tarihi 14.03.2022

Face Mask Detector with OpenCV, Keras/TensorFlow, and Deep Learning. (04.05.2020)

URL-4 < https://pyimagesearch.com/2020/05/04/covid-19-face-mask-detector-with-opencv-keras-tensorflow-and-deep-learning/ >, Ziyaret tarihi 16.03.2022

Pyplot tutorial Matplotlib 3.5.0 documentation. (t.y)

URL-5 < https://matplotlib.org/3.5.0/tutorials/introductory/pyplot.html>, Ziyaret tarihi 17.03.2022

Matplotlib PyPI. (t.y)

URL-6 < https://pypi.org/project/matplotlib/>, Ziyaret tarihi 17.03.2022

Yapay Sinir Ağları Derin Öğrenme Deep Learning. (04.03.2017)

URL-7 < https://www.derinogrenme.com/2017/03/04/yapay-sinir-aglari/>, Ziyaret tarihi 02.05.2022

OpenCV Face Recognition PyImageSearch. (24.09.2018)

URL-8 <https://pyimagesearch.com/2018/09/24/opencv-face-recognition/>, Ziyaret tarihi 05.04.2022

Machine learning - Wikipedia. (26.04.2019)

URL-9 < https://en.wikipedia.org/wiki/Machine_learning/>, Ziyaret tarihi 05.04.2022

OpenCV Python Course - Learn Computer Vision and AI. (07.06.2017)

URL-10 < https://www.youtube.com/watch?v=P4Z8_qe2Cu0/>, Ziyaret tarihi 05.04.2022

ÖZGEÇMİŞ

1996 yılı Ankara doğumluyum. Açık lise eğitimimden sonra Sinop Meslek Yüksek Okulu Bilgisayar Programcılığı eğitimimi tamamladım. Akabinde şu anda eğitimimi tamamlamak üzere olduğum Zonguldak Bülent Ecevit Üniversitesi Bilgisayar Mühendisliği bölümüne DGS sınavı ile geçiş yaptım. Stajımı T.C. Millî Eğitim Bakanlığı, Yenilik ve Eğitim Teknolojileri Genel Müdürlüğü'nde gerçekleştirdim.

ADRES BİLGİLERİ:

Adres: Misket Mah. 628. Cad. No: 8/6 Mamak / Ankara

Tel: 5547230404
E-posta: osmansirakaya96@gmail.com
GitHub: github.com/osmansirakaya
Linkedin: linkedin.com/in/osmansırakaya

Osman SIRAKAYA

ÖZGEÇMİŞ

Merhaba, ben Batuhan. 22 yaşındayım ve Türkiye merkezli serbest yazılım geliştiricisiyim. Zonguldak Bülent Ecevit Üniversitesi'nde okuyorum ve bu yıl Bilgisayar Mühendisliği bölümünde üçüncü yılım.

ADRES BİLGİLERİ:

Adres: Keçiören / Ankara

Tel: 5074172985
E-posta: bba.softt@gmail.com
GitHub: github.com/BatuhanBugraAkca

Batuhan Buğra AKÇA

ÖZGEÇMİŞ

1996 yılı Ankara doğumluyum. Çubuk Endüstri Meslek Lisesinden sonra Ankara Üniversitesi Elmadağ Meslek Yüksek Okulu Bilgisayar Programcılığı eğitimimi tamamladım. Şu anda eğitimimi tamamlayacağım Zonguldak Bülent Ecevit Üniversitesi Bilgisayar Mühendisliği bölümüne DGS sınavı ile geçiş yaptım. Stajlarımın birisini Personel Takip otomasyonu yaparak proje olarak teslim ettim diğer stajımı ise Çubuk Belediyesinde yaptım.

ADRES BİLGİLERİ:

Adres: Yıldırım Beyazıt Mah. Meteoroloji Sok. No: 10/9 Çubuk / Ankara

Tel: 5468747026
E-posta: denizertepe@gmail.com
Linkedin: linkedin.com/in/denizertepe

Deniz ERTEPE