T.C.
SAKARYA ÜNİVERSİTESİ
BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ

BSM 401 BİLGİSAYAR MÜHENDİSLİĞİ TASARIMI

AĞIZ HASTALIKLARININ SINIFLANDIRILMASI VE MOBİL UYGULAMA

B201210008 – Beytullah YAYLA
B201210065– Mehmet Oğuz ÖZKAN

Bölüm : Bilgisayar Mühendisliği
Danışman : Dr. Öğr. Üyesi Muhammed Fatih ADAK

2023-2024 Güz Dönem

İÇİNDEKİLER

ÖNSÖZ

Günümüzde, diş sağlığının önemi giderek artmaktadır. Ancak, bu alandaki bilgilere ulaşmak genellikle zaman alıcı ve zorlayıcı olabilir. İşte tam da bu noktada, DentAI devreye giriyor. Mobil uygulama ara yüzü sayesinde kullanıcılar, dişlerinin fotoğrafları üzerinden kolayca ön tanı yapabilirler. DentAI, hem hekimlerin hem de hastaların ön tanı için kullanabileceği bir platform sunarak, ağız sağlığının daha erişilebilir hale gelmesine katkıda bulunmayı hedeflemektedir.

Bu çalışmanın hazırlanması sürecinde, bize yardımcı olan, yol gösteren, emeği geçen hocamız Dr. Öğr. Üyesi Muhammed Fatih Adak, bu zamana kadar bize maddi manevi destek olan ailemize, okul hayatımız boyunca yanımızda olan arkadaşlarımıza, çalışmalarımızda bize her türlü yardımı sağlayan bütün hocalarımıza teşekkürü borç biliriz.

KISALTMA LİSTESİ

VGG	: Visual Geometry Group
Resnet	: Residual Network
XCeption	: Extreme Inception

ŞEKİL LİSTESİ

ÖZET

Anahtar kelimeler: *Ağız Hastalığı Sınıflandırma, Sağlık Tavsiye Sistemi, Derin Öğrenme, Transfer Öğrenmesi, Obje Tespiti*

DentAI, bir mobil uygulama yardımıyla kullanıcılara ağız sağlığı ile ilgili bilgiler sunmayı amaçlar.

DentAI, ön tanı için hekimler veya hastalar tarafından kullanılabilecek bir uygulamadır. Geliştirdiğimiz mobil uygulama arayüzü sayesinde kullanıcılar kolay bir şekilde uygulamaya üye olup, dişleri gözükecek şekilde dişlerinin fotoğrafı ile ağızlarında hangi hastalıkların olabileceği, bu hastalıklara karşı alınabilecek önlemler gibi bilgileri elde edebilirler.

Diş sağlığınız hakkında bilgi almak için uzun bir süre randevu almak için beklemektense DentAI uygulamasını kullanarak dişleriniz hakkında ön bilgiye sahip olabilir, ağız sağlığınızı kişiselleştirilmiş uygulamanız ile takip edebilirsiniz.

Uygulama Python programlama dili ve bir Javascript kütüphanesi olan React Native kullanarak yazılmıştır. Veritabanı olarak MySQL kullanılmıştır. Api tünelleme aracı olarak Ngrok kullanılmıştır.

1. GİRİŞ

Günümüzde, sağlık sektörü teknolojik gelişmelerin etkisi altında büyük bir dönüşüm geçiriyor. Yapay zeka, sağlık alanında önemli bir rol oynamakta ve hastaların, sağlık profesyonellerinin ve tıp araştırmacılarının yaşamlarını kolaylaştırmak, tedaviyi iyileştirmek ve hastalıkları daha etkili bir şekilde yönetmek için kullanılmaktadır. Özellikle yapay zekadaki derin öğrenme alanının gelişmesi ile birlikte görüntü işleme, bilgisayarlı görü alanlarında devrim sayılabilecek nitelikte gelişmeler yaşanmıştır. Yapay zekanın, özellike görüntü işleme kısmının sağlık alanında çok yaygın etki bulması ile birlikte, günlük pratikte görüntülerin önemli olduğu alanlarda çok hızlı bir atılım yaşandı. Son yıllarda patoloji, radyoloji, cerrahi ve psikiyatri gibi birçok alanda birçok bilimsel yayın ve ürün ortaya çıktı. Bu uygulamaların ilk amacı, öneleme veya tedavi yöntemleriyle hasta sonuçları arasındaki ilişkiyi analiz etmektir. Özellikle tanı süreçlerinin kolaylaştırılması , kişiye özel tıp, hasta izleme, ilaç tedavisi gibi görevler için yapay zeka programları kullanılmaktadır.

Bu bilgiler doğrultusunda çeşitli araştırmalar yaptığımızda bu alanın bizim için oldukça ilgi çekici olduğunu ve geliştirilebilecek uygulama açısından zengin bir içeriğe sahip olduğunu fark etiik. Özellikle görüntü işleme ve derin öğrenme algoritmalarının gelişmesiyle bu alanda görüntü tabanlı yapılan çalışmalara baktığımızda, genellikle bu çalışmalar arasında dermatolojide cilt lekesi ve tümör tespiti, cilt hastalıklarının sınıflandırması gibi çalışmalar yapılırken; radyolojide bilgisayarlı tomografi veya manyetik rezonans(MR) gibi görüntüleme tekniklerinin analizinde derin öğrenme algoritmaları, tümör tespit, organ hasarı değerlendirmesi; nöroloji alanında ise alzheimer hastalığı gibi kompleks nörolojik durumların erken teşhisisi gibi alanlarda oldukça fazla kullanıldığını görmekteyiz.

1.1.Projenin Amacı ve Hedef

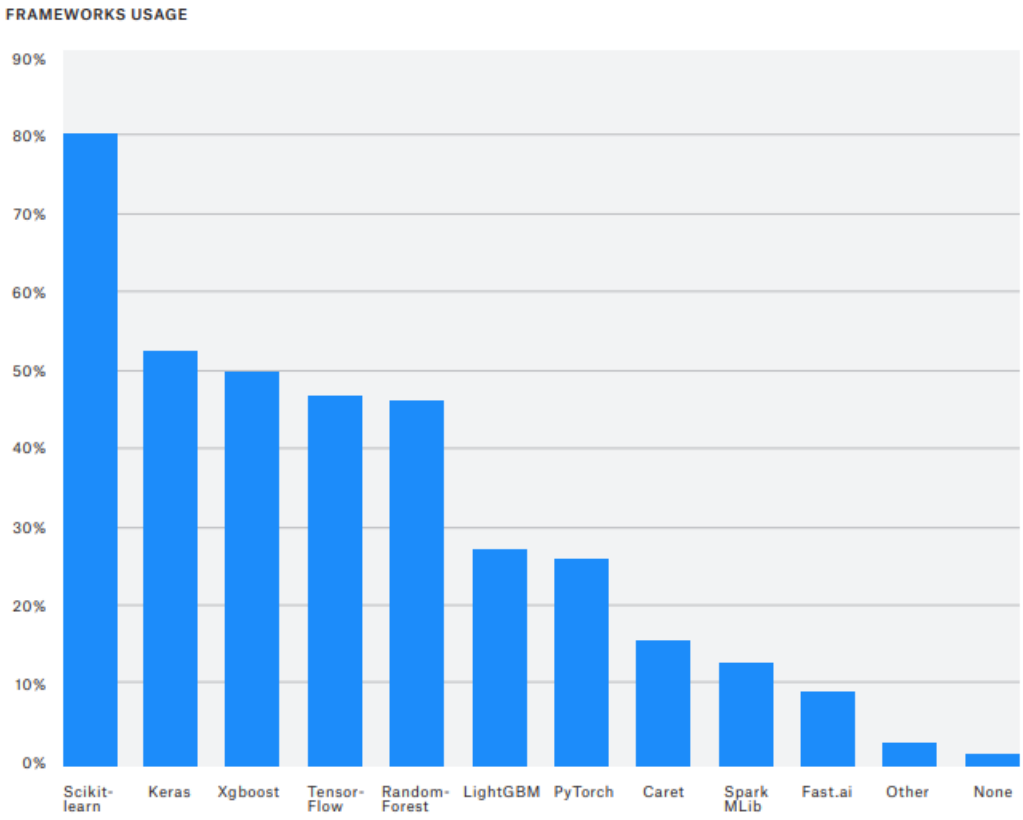
Teknolojinin ilerlemesiyle bilgisayarlı görü uygulamaların mobil uygulamalara çokça taşındığını görmekteyiz. Projenin sonunda elimizde somut bir bilgisayarlı görü uygulaması olmasını istediğimizden dolayı mobil uygulamanın kamerasını kullanabileceğimizi düşündük ve bunun için araştırmalarımız esnasında daha az rastladığımız ağız hastalıklarının telefon kamerası kullanılarak sınıflandırılması ile ilgili bir çalışma yapmayı düşündük.

Uygulamamızda yapay zeka algoritmaları, mobil uygulamamızın arka/ön kamerasından aldığı verileri görüntü işleme/derin öğrenme algoritmaları yardımıyla analiz ederek kullanıcılara bir ağız hastalığına sahip olup olmadığı, varsa hangi hastalığa sahip olduğu gibi konularda bilgi vermesini amaçlıyoruz. Bu sayede, kullanıcının kapsamlı bir muayeneden geçmeden ağız sağlığıyla alakalı önbilgi alabilmesini amaçlıyoruz.

1.2.Projede Kullanılan Teknolojiler

Bir projeyi gerçekleştirmek için kullanacağımız teknolojiler zaman yönetimi, yazım kolaylığı, bellek maliyeti, hız gibi değişkenleri direkt veya dolaylı bir şekilde etkileyebileceği için kullanacağımız teknolojileri belirlemek oldukça önemli bir bölümdür.

- **Python:** Pyhon'daki yazım kolaylığı, kütüphane desteğinin oldukça fazla olması, güçlü bir topluluğunun olması , genel bir programlama dili olduğu için yaptığımız uygulamayı dış dünyaya servis edebilme imkanı sunduğu için bizim gözümüzde diğer programlama dillerinden bir adım daha öne çıktı ve projemizde **python** dilini kullanmaya karar verdik.
- **Tensorflow:**Python'da nesne tespiti ve resim sınıflandırma görevleri için bilgisayarlı görü modellerinin oluşturulması, eğitilmesi, kaydedilmesi ve yüklenmesi gibi görevler için de içinde çeşitli konvolüsyon bloklarının, aktivasyon fonksiyonlarının, kayıp fonksiyonlarının, daha önceden eğitilmiş modellerin bulunduğu işimizi kolaylaştıran bazı kütüphaneler mevcuttur. Bunlardan en popülerleri **PyTorch**, **Keras** , **Tensorflow** ve **Caffe** kütüphaneleridir. Piyasada özellikle bilgisayarlı görü alanında daha yoğun olarak Tensorflow ,Pytorch ve Keras kullanılmaktadır. Biz projemizde ağırlıklı olarak tensorflow ve keras kütüphanelerini kullandık.



Şekil 1.1. Yapay Zeka Kütüphanelerinin Kullanım Oranları

- **FastApi:**Modelimizi bir ürün haline getirmek için ise dış dünyaya bir api aracılığıyla, tahmin sonuçlarını döndürecek şekilde açtık. Python'da backend api geliştirmek için **Django**, **Flask** ve **FastApi** gibi kütüphaneler kullanılır. Basit, performans olarak hızlı olduğu için **FastApi** kütüphanesini tercih ettik.

- **MYSQL RDMS:**Oluşturduğumuz api yardımıyla yapılan tahmin işlemlerinin sonuçlarının ve girdilerinin kayıt altına alınmasını istiyoruz. Bu veriler daha sonra hasta yönetim sistemleri gibi sistemlerde kullanılabilir. Verilerin eklenmesi, silinmesi, kaydedilmesi gibi ifadeler denilince bilgisayar bilimlerinde akla gelen veritabanı yönetim sistemi teknolojilerini kullanarak bu işlemleri gerçekleştirebiliriz. Bunun için projemizde **MySQL** ilişkisel veritabanı sistemini kullandık.

- **React Native:**Son olarak kullanıcımızın oluşturduğumuz servislerle etkileşmesi için bir arayüze ihtiyacımız var. Kamera kullanım kolaylığından dolayı bir mobil arayüz olması konusunda arkadaşımla anlaştık ve bir mobil uygulama olarak servislerimizi sunacağız. Mobil uygulama için android veya ios'ta çalışabilen **React Native** dilini tercih ettik.

2. METHOTLAR VE PROJE GELİŞTİRME SÜRECİ

2.1 Veri Toplama

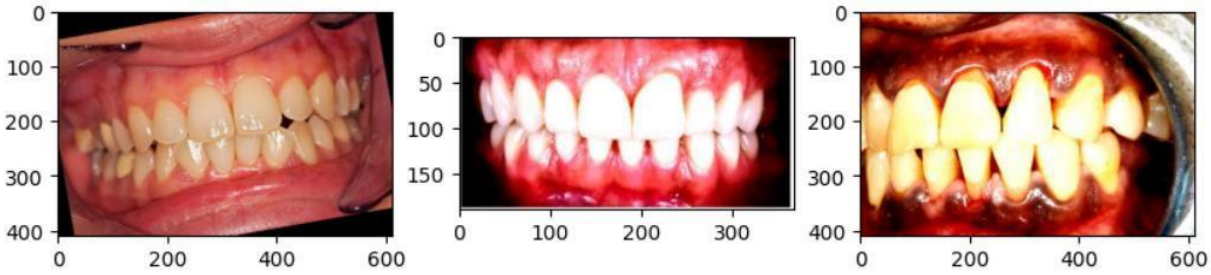
Veri toplama aşaması makine öğrenmesi tabanlı projeler için oldukça önem arz etmektedir. Çünkü işin özünde topladığımız veriler ne kadar kaliteliyse ve bu veriler ne kadar iyi etiketlenmişse(gözetimli öğrenme algoritmaları için) aslında başarılı bir proje çıktısı almamız o kadar mümkündür.

Araştırmalarımız sürecinde veri bilimciler için önemli platformlar olan **kaggle,roboflow** gibi siteler üzerinden çeşitli veri setlerini inceledik ve projemizdeki 5 farklı ağız hastalığına sahip, her klasörde o klasörün ismindeki hastalığı içeren resimlerin olduğu bir veri seti oluşturduk.

2.1.1. Veri Setindeki Sınıflar

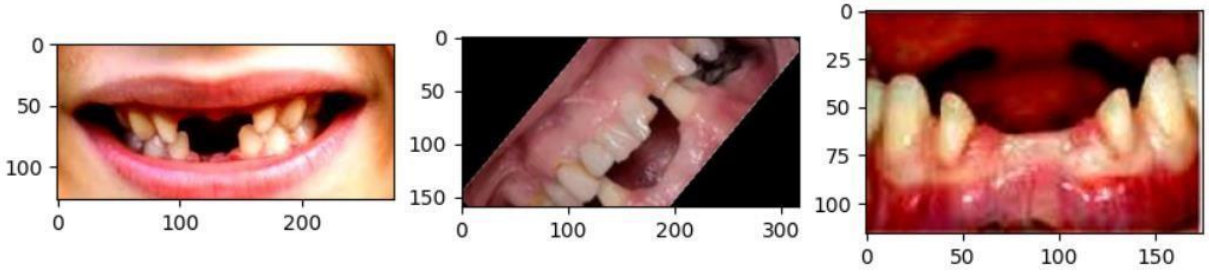
Topladığımız resimlerde 5 farklı sınıf vardır. Bu 5 farklı sınıf aslında birbirlerinden farklı hastalıkları temsil ediyor. Aşağıda bu hastalıklar detaylı bir şekilde açıklanmıştır.

- **Diş Eti Şişmesi(Gingivitis):** Diş eti dokusunda kızarıklık, şişme veya özellikle dişlerinizi fırçaladığınızda veya diş ipi kullandığınızda kanamaya yol açma eğilimi gösterir.



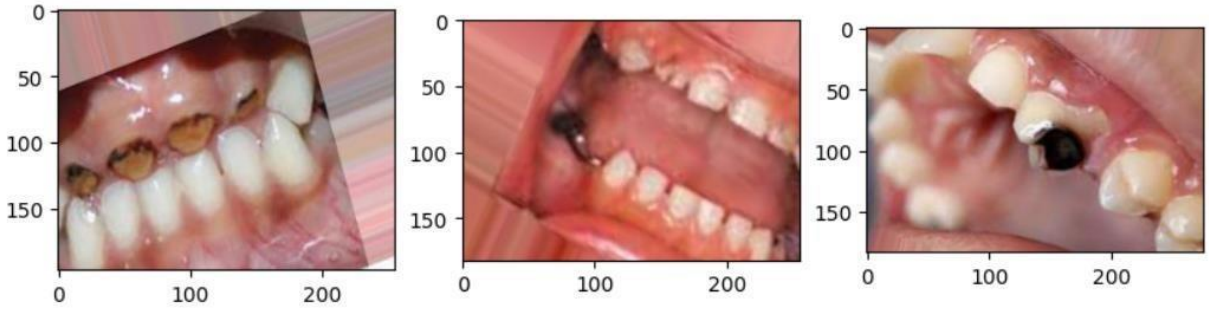
Şekil 2.1 Gingivitis

- **Diş Eksikliği Sendromu(Hypodontia)**Hipodonti diş hekimlerinin, doğuştan eksik dişleri tanımlamak için kullandıkları bir terimdir. Ağızın herhangi bir bölgesinde meydana gelebilir.



Şekil 2.2 Hipodonti

- **Diş Çürükleri(Tooth Caries):**Diş çürüğü diş minesinin ve dişin iç kısmındaki dentin tabakasının bakteriler tarafından neden olan bir enfeksiyon sonucu oluşan bir durumdur. Bu enfeksiyon, ağızda bulunan bakterilerin şeker ve karbonhidratlı yiyeceklerle etkileşime girmesiyle başlar.

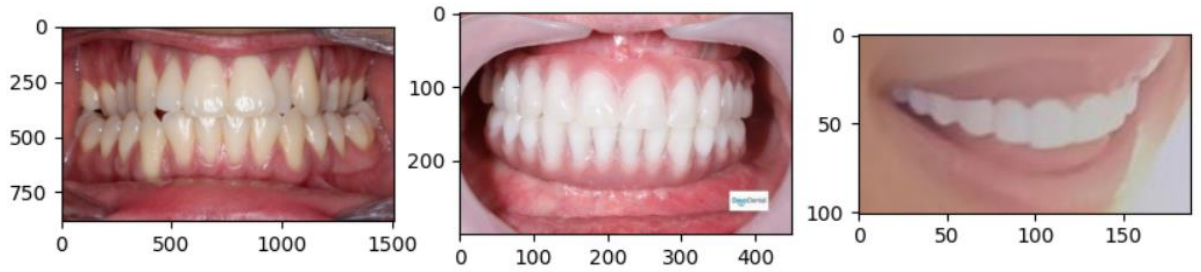


Şekil 2.3. Diş Çürüğü

- **Tartar(Calculus):** Tartar olarak da bilinen diş taşı, sertleşen plak ve mineral birikimi olarak tanımlanır. Plağın uzun süre temizlenmemesiyle ortaya çıkan diş taşı, dişlerin büyük bir kısmını kaplayabilir ve diş eti çizgisinin altına kadar yayılabilir. Genellikle dişlerin arkasına ve arasına yerleşen diş taşı birikintileri kahverengi veya sarı renktedir.

Şekil 2.4 Tartar

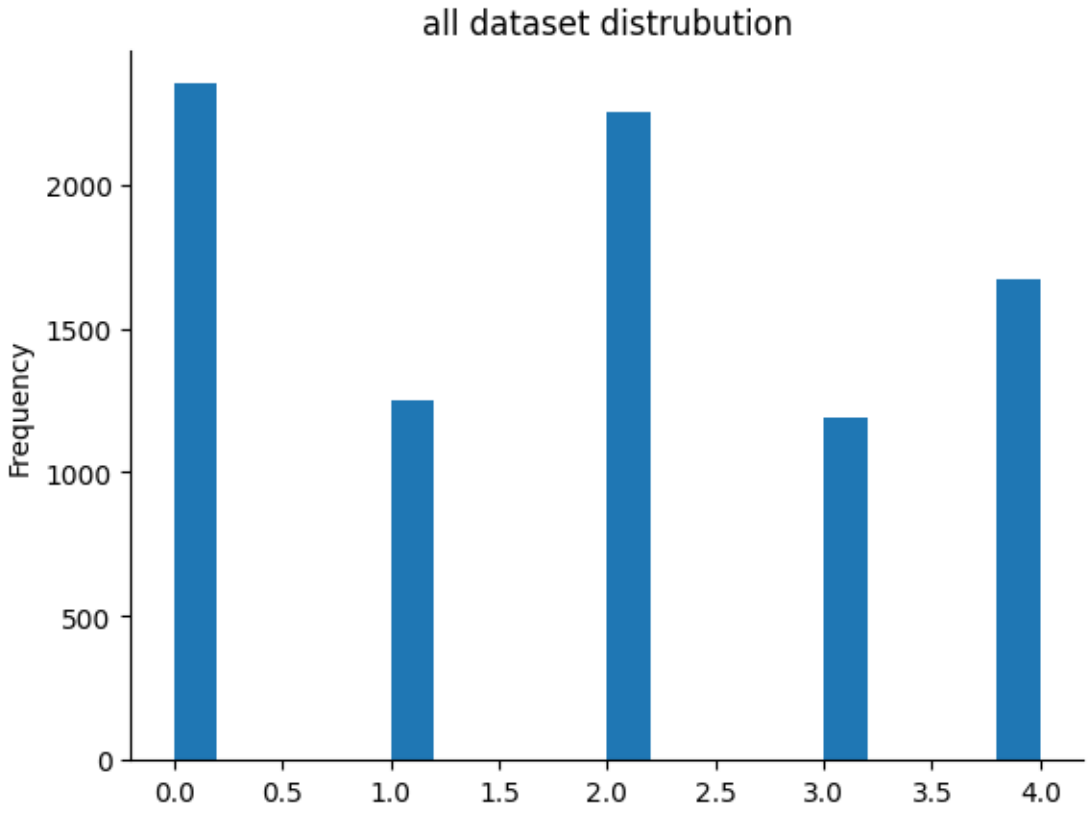
- **Sağlıklı(Healthy):** Sağlıklı dişler çürüksüz,kırıksız ve diş minesini kaybetmemiş dişlerdir.



Şekil 2.5 Sağlıklı Dişler

2.1.2. Veri Setindeki Sınıfların Dağılımı

Veri setimizde toplamda 1191 tane tartar verisi, 2254 tane diş çürüğü verisi, 2349 tane gingivitis verisi, 1251 tane hipodonti verisi ve son olarak 1670 tane sağlıklı veri kullanılmıştır. Toplam veri sayısı 8715'tir. Tartar verisi ve diş çürüğü verisi sayı olarak diğer sınıflardan biraz daha fazla örnek içerse de yine de sayılar birbirine yakın olduğundan, topladığımız tüm verileri kullandık. Aşağıdaki grafikte **0:Gingivitis**, **1:hypodontia, 2:Tooth Caries**, **3:Calculus**, **4:Healthy** sınıflarını temsil etmektedir.



Şekil 2.6 Veri Setinin Dağılımı

2.2 Veri Ön İşleme

Veri ön işleme adımı yapay zeka modelini efektif bir şekilde eğitebilmemiz için veri üzerinde yapılan değişiklikler olarak tanımlanabilir. Bu kısımda birbirinden farklı boyutlara sahip olan verilerin boyutlarının ayarlanması, verinin normalize edilmesi gibi işlemler yapılmıştır.

2.2.1.Veri Seti İçin Dataframe Yapısının Oluşturulması

Verilerimiz klasör yapısı şeklinde tutulmaktadır. Bu verileri pythonda kullanmak için pythonda karşılığı olan bir formata dönüştürmemiz gerekiyor. Her sınıfın bilgisayarımızda bulunan tam yollarını tanımladıktan sonra bir dataframe yapısı oluşturup, resmin tam yolunun ve o resme karşılık gelen sınıfın etiket değerinin olduğu bir pandas dataframe yapısı oluşturduk. Dataframe'in içeriği aşağıdaki figürdeki gibidir:

	filename	class
0	/content/drive/MyDrive/Tasarım Çalışması Dat...	2
1	/content/drive/MyDrive/Tasarım Çalışması Dat...	1
2	/content/drive/MyDrive/Tasarım Çalışması Dat...	2
3	/content/drive/MyDrive/Tasarım Çalışması Dat...	4
4	/content/drive/MyDrive/Tasarım Çalışması Dat...	1

Şekil 2.7 Dataframe Yapısı

Daha sonra pythondaki sınıflardan biri olan ImageDataGenerator sınıfını kullanarak bir batch generator oluşturacağız.

2.2.2.Eğitim İçin Generator Yapısının Oluşturulması

Bu adımda kerastaki ImageDataGenerator sınıfını kullanarak bir data generator nesnesi oluşturuyoruz. Bu nesneyle resimler üzerinde ilk olarak normalizasyon işlemi sağlamış oluyoruz.

```
from keras.preprocessing.image import ImageDataGenerator
datagen = ImageDataGenerator(
    rescale = 1/255.0
)
```

Şekil 2.8 Image Data Generator

Daha sonrasında eğitim sırasında kullanacağımız verileri bu nesnedeki bir method olan flow_from_dataframe() adlı methodu kullanarak gerçekleştiriyoruz.

```
train_generator = datagen.flow_from_dataframe(
    dataframe=train_data,
    xcol='filename',
    ycol='class',
    batch_size=32,
    target_size=(150, 150),
    class_mode='categorical',
    color_mode='rgb'
)
```

Şekil 2.9 Image Data Generator

Bu sayede daha önceden oluşturduğumuz dataframe yapısını kullanarak generator'lar oluşturabiliriz. Bu veri setinin daha düzenli ve kolay yönetilebilir olmasını sağlar. Dataframe, veri seti üzerinde hızlı ve etkili işlemler yapma olasılığı sunduğu için eğitim hızını bir miktar arttırabilir.

- 'class_mode=categorical' parametresi, çoklu sınıflandırma problemleri için olan bir çıktı formatını belirtir. Bu etiketlerin 'one-hot-encoding' şeklinde oluşturulmasını sağlar.
- 'color_mode=rgb' parametresi, RGB renk modunu belirtir.Bu görüntülerin renkli olduğunu ve 3 kanaldan oluştuğunu belirtir.
- 'target_size=(150,150)' parametresi, görüntülerin yeniden boyutlandırılacağı hedef boyutu belirtir. Bu, görüntülerin model tarafından işlenirken aynı boyutta olmalarını sağlar.

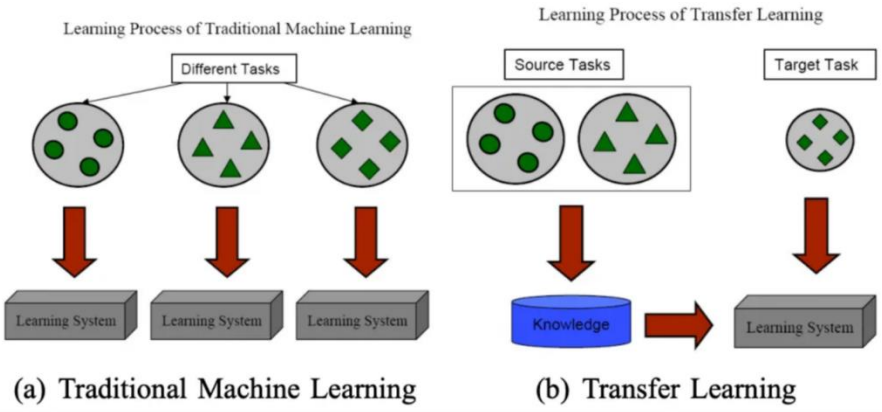
Bu avantajlar, özellikle büyük verisetlerini etkili bir şekilde yönetmek, eğitim veri setini çeşitlendirmek, ve sınıflandırma problemleri için uygun formatları belirlemek açısından değerlidir.

2.3 Sınıflandırma Modeli Geliştirme

Bu bölümde hazırladığımız verisetini kullanarak verilen resmin hangi hastalığa karşılık geldiğini tahmin etmemizi sağlayacak bir çoklu sınıflandırma modeli eğiteceğiz. Bu model ile birlikte, modele verdiğimiz resmin sahip olduğumuz 5 sınıftan hangisine sahip olduğunu öğrenmek istiyoruz.

2.3.1.Öğrenme Aktarımı

Öğrenme aktarımı(Transfer Learning) makine öğrenmesi yöntemlerinin aynı bizim gibi bir problemi çözerken elde ettiği bilgileri saklayıp, başka bir problem ile karşılaştığında o bilgiyi kullanmasıdır. Bu şekilde daha az veri ile daha yüksek başarı elde edebiliriz.[1]



Şekil 2.10 Transfer Learning[1]

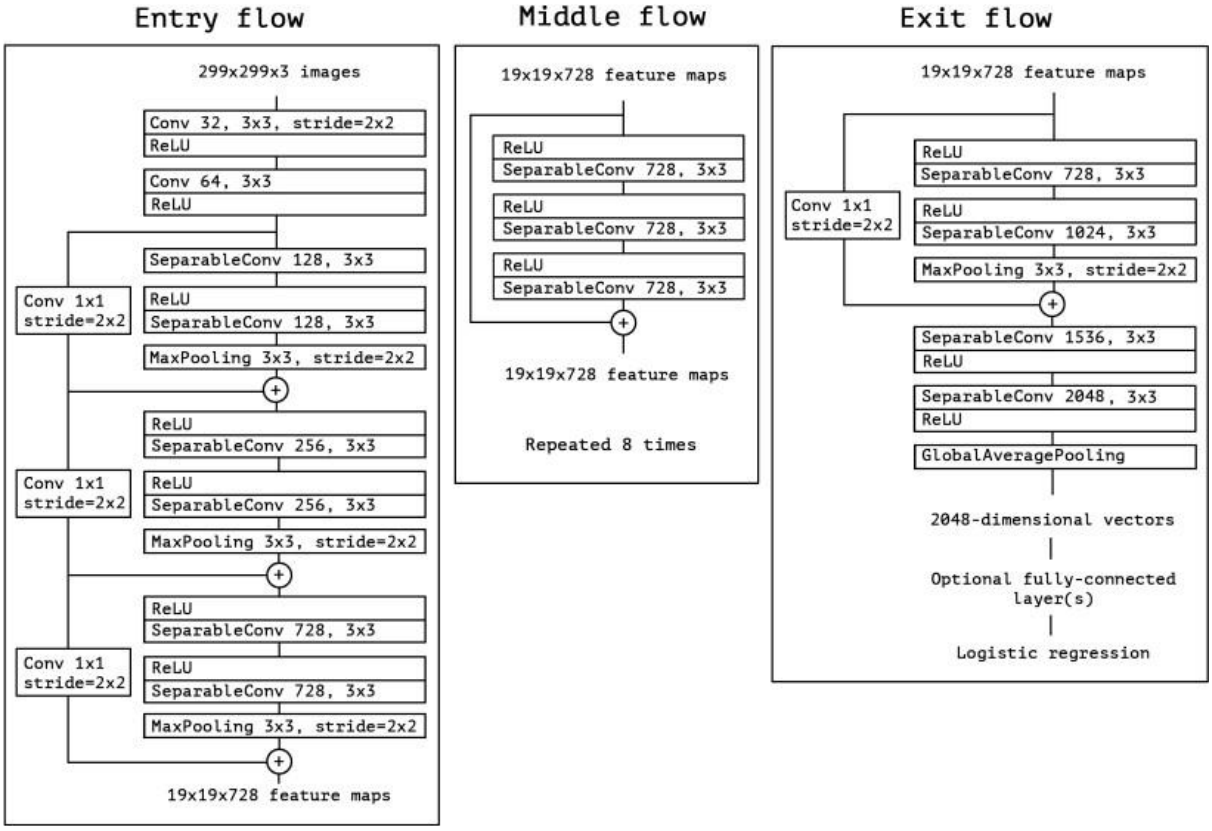
Öğrenme aktarımı için aslında birçok önceden eğitilmiş model kullanılmaktadır. Bunlardan bazıları:

- **VGG(Visual Geometry Group):**Bilgisayarlı görü üzerine odaklanan bir derin öğrenme modelidir. VGG-16 ve VGG-19 versiyonlarına sahiptir.
- **Resnet(Resiudal Geometry Group):**Resnet mimarisi öğrenme aktarımı modelleri için popüler hale gelmiştir çünkü resiudal blok yapısı daha derin ağların eğitilmesini mümkün kılar.
- **Mobilenet:**Mobilenet, özellikle hafif ve taşınabilir cihazlarda kullanılmak üzere tasarlanmış bir görüntü sınıflandırma modelidir. Transfer öğrenme uygulamalarında, özellikle düşük kaynaklı cihazlarda etkili olabilmektedir.
- **Xception(Xtreme Inception):**Bu mimari Inception model yapısına dayanır ve birçok alanda başarılı sonuçlar elde etmiştir. Inception modelinin temel fikirlerini temel alırken, evrişim katmanlarını da geliştirmek amacıyla derinlemesine ağaç yapısını kullanır. Her bir özellik haritası çıkarımı için Depthwise Seperable Convolution yapısını benimser.

2.3.2.Model Mimarisi

Model eğitimi için daha önce imagenet veri setiyle olan Xception modelini tercih ettik. Xception modeli(Extended Perception) derin öğrenme alanında bir sinir ağı modelidir.

- Derin bir ağ yapısına sahiptir.
- Depthwise Seperable Convolutions yapısına sahiptir. Yani geleneksel evrişim işlemlerinin yerine daha hafif bir yaklaşım sunar ve ağı daha ölçeklenebilir hale getirir.
- Transfer learning için uygun bir networktür. Bu sayede çok uzun eğitim süreleri yerine daha kısa sürede eğtiim yapıp daha iyi sonuçlar alabiliriz.



Şekil 2.11 Xception Ağ Mimarisi[2]

Depthwise Seperable Convolutions genelde başka bir adım olan derinlemesine ayrılabilir evrişim ile birlikte uygulanır. Bu iki bölümden oluşur:

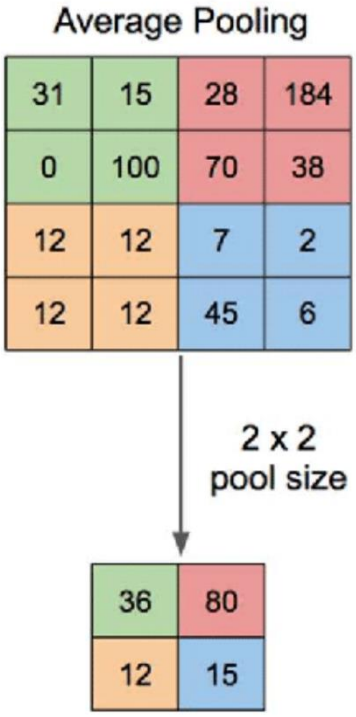
- **Filtreleme:** Bu adımda, her giriş kanalı kendi filtre seti ile ayrı ayrı birleştirilir. Bu, her kanalın bağımsız olarak işlendiği ve her giriş kanalı için bir dizi filtrelenmiş özellik haritasıyla sonuçlandığı anlamına gelir. Bu adımın amacı, her giriş kanalındaki ayırt edici desenleri yakalamak ve vurgulamaktır. Kanallar arasında bilgi karıştırmadan modelin her kanal için belirli özellikleri öğrenmesini sağlar.
- **Birleştirme:** Derinliksel evrişimden sonra, her bir giriş kanalından elde edilen özellik haritaları, 1x1 evrişimler kullanılarak doğrusal olarak birleştirilir. Bu 1x1 evrişimler, 1x1 filtrelere sahiptir.

Depthwise Seperable Convolutions'lar genellikle daha az parametreye sahiptir. Daha az parametre de genellikle daha az hesaplama anlamına gelir. Eğitim ve çalışma süresini kısaltır. Bu özellikle sınırlı hesaplama kaynaklarına sahip cihazlar veya uygulamalar için elverişlidir.

Öğrenme aktarımının temelinde daha önce öğrenilen bilginin, networkün son katmanlarını kendi görevimize göre düzenleyip modelimizin daha az veriyle daha hızlı bir şekilde öğrenmesi yatar. Biz de bu özelliği kullanarak temel ağ(base network) olarak keras kütüphanesinin sağladığı Xception ağını kullandık. Bu önceden eğitilmiş ağ imagenet datasetinin ağırlıklarını kullanır. Ağımız giriş değeri olarak 150x150x3'lük resimler alır ve bu boyut üzerinden tahminlerini yapar.

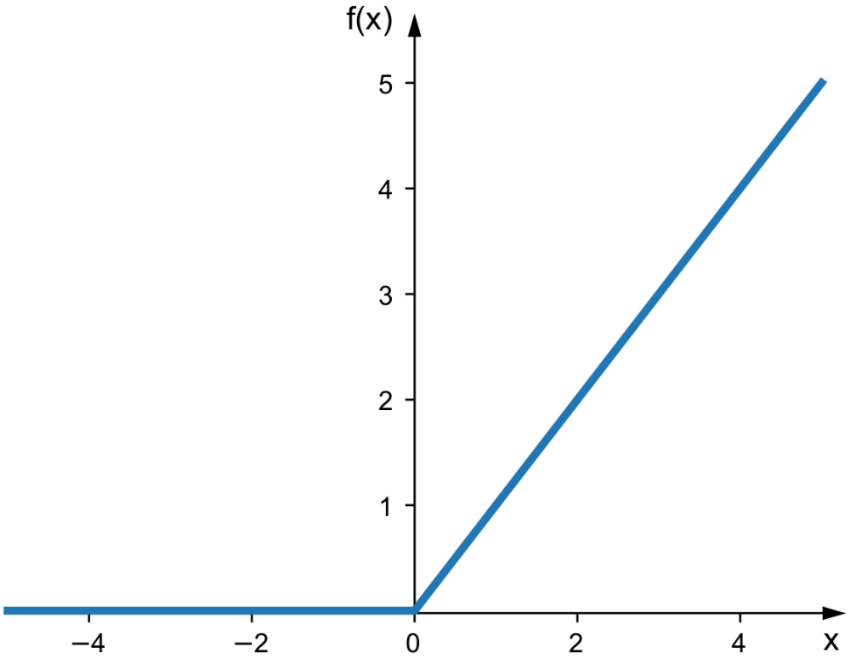
Modeli kendi görevimize uyarlamak için bazı saklı katmanlar(hidden layers),havuzlama katmanları(Pooling Layers) ve batch normalization katmanları ekledik.

Havuzlama katmanı olarak AveragePooling2D katmanları kullanıldı. Havuzlama, evrişimli sinir ağlarında genellikle hesaplama maliyetini azaltmak için kullanılır. Average pooling yöntemiyle, özellik haritasındaki her grup için o grubu oluşturan değerlerin ortalaması alınır ve yeni haritada ilgili indise ortalaması yazılır.



Şekil 2.12 Average Pooling[3]

Eklediğimiz ara katmanlarda aktivasyon fonksiyonu olarak ReLU(Rectified Linear Unit) fonksiyonunu kullandık. Aktivasyon fonksiyonlarının temel görevi, modele lineer olmayan özelliklerin tanıtılmasıdır. ReLU aktivasyon fonksiyonu genellikle vanishing gradient descent problemini çözer. ReLU aktivasyon fonksiyonu genel olarak eğer aldığı girdi değeri 0'dan küçük ve 0'a eşit bir değerse 0, 0'dan büyük değerler için girdinin değerini döndürür. ReLU derin öğrenme çalışmalarında en çok kullanılan ve en başarılı sonuçların alındığı aktivasyon fonksiyonlarının başında gelir. Aşağıda ReLU aktivasyon fonksiyonunun grafiği verilmiştir.

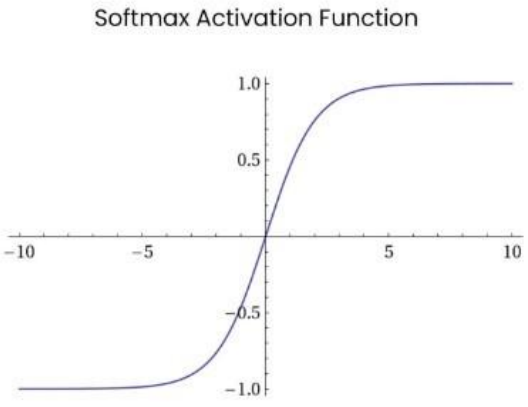


Şekil 2.13 Relu Aktivasyon Fonksiyonu[4]

Sonuç katmanı olarak, sınıflandırma yaptığımız için toplam 5 nörondan oluşan bir katman ekliyoruz. Bu katman aslında bizim sınıflandırma katmanızdır. 5 sınıfımız olduğundan dolayı 5 nöron kullandık.

Aktivasyon fonskiyonu olarak çoklu sınıflandırma problemleri için kullanılan **softmax** aktivasyon fonksiyonunu kullandık. Softmax girdi olarak aldığı değerleri kullanarak aslında her bir girdinin bizim gerçek değerimiz olma olasılığını hesaplar ve bize döndürür. Biz de uygulamamızda bu uygulamaları görmek istediğimizden dolayı buradan sonra herhangi bir encoding tekniği kullanmadık.

$$softmax(z)i = e^{zi} / \sum_{j=1}^{N} e^{zj}$$



Şekil 2.14. Softmax Aktivasyon Fonksiyonu[5]

Modelimizin katmanlarını hazırladıktan sonra modelimizi keras'ta derlemeliyiz.

Bunun için öğrenme katsayımızı 0,001 olarak kullandık. Birçok derin öğrenme probleminde bu değere yakın değerler kullanıldığını görebiliriz. Biz de öğrenme hızımızın çok hızlı ve tutarsız olmasını istemediğimizden dolayı öğrenme katsayımızı 0,001 olarak seçtik.

Öğrenme katsayımızdan sonra modelimizi keras üzerinde derleyebilmemiz için bir diğer parametre olan optimizer; modelin parametrelerini hata veya kayıp fonksiyonu dediğimiz fonksiyonu minimum olacak şekilde eğitim sırasında güncellemekten sorumludur. Stochastic Gradient Descent(SGD), Adam(Adaptive Moment Estimation),RMSProb(Root Mean Square Propagation) gibi birçok optimizer vardır. Bu optimizer'lar sağladıkları performans başarısına göre karşılaştırıldıklarında hangisinin daha iyi sonuçlar sağladığı senaryoya göre değişse de, yapılan çalışmalar incelendiğinde özellikle Adam Optimizer oldukça tatmin edici sonuçlar vermiştir. Bu yüzden biz de projemizde Adam Optimizer kullanmaya karar verdik. Araştırma yaparken rastladığımız optimizasyon algoritmalarının konvolüsyonel sinir ağlarındaki performanslarının karşılaştırılması ile ilgili görseli aşağıdaki figürden inceleyebilirsiniz:

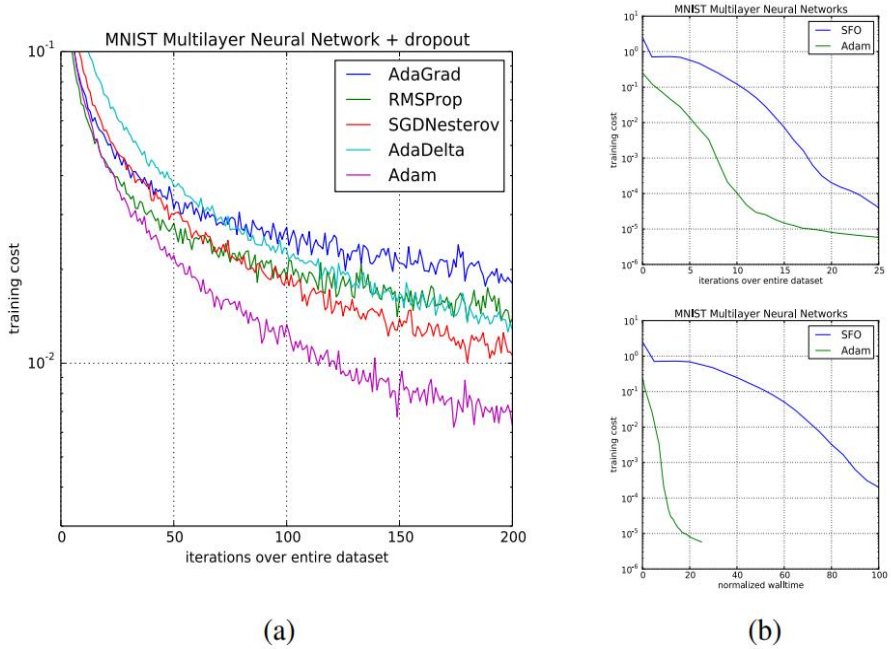


Figure 2: Training of multilayer neural networks on MNIST images. (a) Neural networks using dropout stochastic regularization. (b) Neural networks with deterministic cost function. We compare with the sum-of-functions (SFO) optimizer (Sohl-Dickstein et al., 2014)

Şekil 2.15 Optimizasyon Algoritmalarının MNIST veriseti üzerinde Karşılaştırılması[6]

Son olarak öğrenme aşamasında modelimizin tahminlerinin gerçek değerlerle ne kadar eşleştiğini ölçmemizi sağlayacak bir kayıp fonksiyonuna ihtiyacımız var. Bunun için çoklu sınıflandırma görevlerinde sıkça kullanılan **categorical crossentropy** fonksiyonunu kullandık. Belirli bir girdi için model, softmax aktivasyon fonksiyonunu kullanarak sınıflar üzerinde bir olasılık dağılımı üretir. Softmax işlevi, çıktı logitlerini (ham puanlar) olasılıklara göre normalleştirir ve toplamlarının 1 olmasını sağlar. Kategorik çapraz entropi kaybı daha sonra tahmin edilen olasılık dağılımı ile gerçek dağılım arasındaki farklılığı ölçer.

$$L(y_{label}, y_{prediction}) = -\sum_{i} y_{label_i} * \log\left(y_{prediction_i}\right)$$

Şekil 2.16 Categorical Cross Entropy Loss Function Matematiksel Formülü

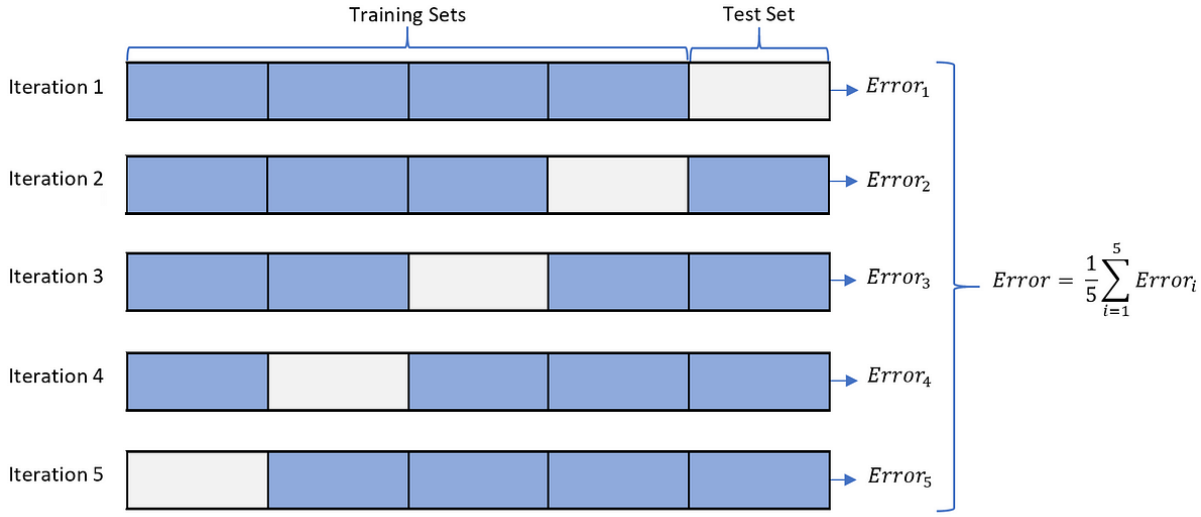
Modelimizi derlemek için gerekli parametreleri belirlediğimize göre gerçekleme aşamasına geçebiliriz ve modelimizi eğitime hazır hale getirebiliriz. Bunun için yazdığımıız **build_model_xception()** adlı fonksiyon aşağıdaki figürde gösterilmiştir.

```
def build_model_xception():
  model=Sequential()
  base_model = Xception(weights="imagenet", include_top=False,input_shape=(150,150,3))

  x = base_model.output
  x = GlobalAveragePooling2D()(x)
  x = BatchNormalization()(x)

  x = Dense(512, activation ='relu')(x)
  x = BatchNormalization()(x)


  x = Dense(256, activation ='relu')(x)
  x = BatchNormalization()(x)


  predictions = Dense(5, activation ='softmax')(x)
  model = Model(inputs=base_model.input, outputs=predictions)

  model.compile(optimizer=Adam(lr=0.001), loss='categorical_crossentropy', metrics=['accuracy'])
  return model
```

Şekil 2.17 Model Gerçeklemesi

2.3.3.K-Fold Çarpraz Doğrulama(Cross Validation) Yöntemi

Modelimizi eğitirken tek bir validasyon seti kullandığımızda skor yüksek olsa bile gerçek hayatta daha düşük başarılar elde edebiliriz. İşte çarpraz doğrulama burada devreye giriyor. Datasetin dağılımına göre tüm veri setini kullanarak daha genel bir skor elde ediyoruz.
K-Fold cross validation aslında temel olarak elimizdeki verinin belirli bir kısmını test için ayırdıktan sonra belirli fold yapılarına bölünüp daha sonra fold sayısı kadar döngü sayısıyla veri setinin o foldunun validation seti olarak kullanılarak aslında daha doğru bir doğrulama skoru elde etmemizi sağlar.
Projemizde sınıflandırma görevlerinde kullanılan stratified k-fold cross validation yöntemini kullanarak fold sayısı olarak 5'i kullandık.
Kütüphane ve method olarak **sklearn** kütüphanesinin sağladığı **StratifiedKFold** methodunu kullandık ve aslında farklı validation verileriyle modelimizi 5 kez eğitmiş olduk. Her döngüde validation ve training accuracy skorlarını bir listede daha sonra görselleştirme adımında kullanılmak üzere tuttuk. Bu hatalardan ortalama hatayı bularak K-Fold Cross Validation hatasını elde etmiş oluyoruz.



Şekil 2.18 K-fold Cross Validation[7]

2.3.4.Eğitim Döngüsü

Eğitim döngüsünde, ilk olarak her fold için, bir önceki adımda anlattığımız gibi eğitim ve test generator nesneleri oluşturuyoruz ve bu nesneleri kullanarak aslında epoch sayısı kadar verisetimizi baştan sona kadar ağımıza veriyoruz. Her epoch sonunda training hatası ve validation hatasını elde ediyoruz. Modelimizi her fold için 15 epoch boyunca eğittik. Batch size olarak farklı denemeler sonucunda 200 değerinin daha iyi sonuçlar verdiğini gördük. Bu yüzden Batch Size olarak 200 değerini kullandık. Batch size aslında modelimizde 1 adımda verdiğimiz veri sayısı olarak tanımlanabilir. Biz bir adımda modelimize 200 tane resim veriyoruz.

Her farklı foldun eğitimi sonucunda modelin doğruluk(accuracy),kesinlik(precision) ve hatırlama(recall) değerlerini boş bir listeye ekledik ve bu sayede model başarısını değerlendirirken bu değerlerden faydalandık.

```
Fold 1: Precision = 0.7959903693294752
Fold 2: Precision = 0.8047829846757959
Fold 3: Precision = 0.752427701944948
Fold 4: Precision = 0.720613354554844
Fold 5: Precision = 0.7813146453460742
```

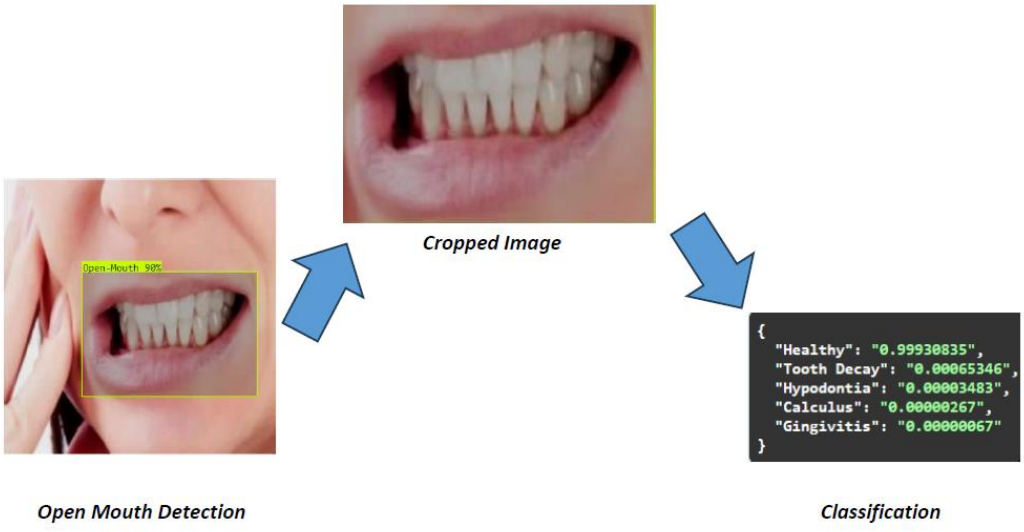
Şekil 2.19 Precision Değerleri

```
Fold 1: Recall   = 0.7928858290304074
Fold 2: Recall   = 0.7016637980493402
Fold 3: Recall   = 0.5920826161790017
Fold 4: Recall   = 0.6184738955823293
Fold 5: Recall   = 0.5220883534136547
```

Şekil 2.20 Recall Değerleri

2.3 Açık Ağız Tespit Modeli Geliştirme

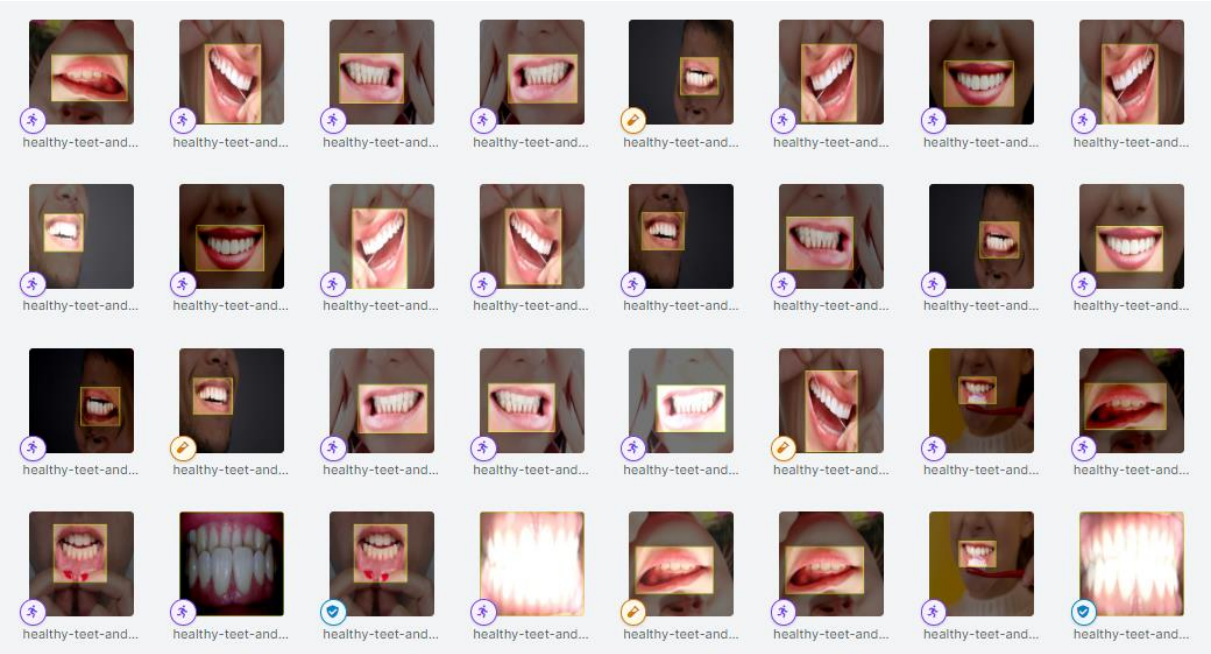
Ağız hastalıklarının sınıflandırılmasını sağlayan modeli geliştirdikten sonra, aklımıza şu sorular geldi: “Kullanıcı eğer yanlışlıkla dişlerini değil de farklı bir yeri çekerse ne olacak?” ve “Eğer kullanıcı dişlerini uzaktan çekerse bu durum sınıflandırma işlemini olumsuz etkiler mi?”. İlk soruyu şu şekilde cevaplandırdık:”Diş olmayan bir yerin fotoğraf üzerinden tahmin yapılması işlemi hem sunucu tarafında bir yoğunluk yaratacaktır hem de kullanıcı deneyimini olumsuz etkileyecektir.”. Diğer soru için ise:”Kullanıcının dişlerini uzaktan çekmesi modelin doğruluk oranını düşürecektir. Dişler uzakta olduğundan dişlerin özellik haritalarının çıkartılması daha zor olacaktır.”. Bu cevaplardan yola çıkarak bir dişlerin gözüktüğü bir ağız tespit modeli geliştirmeye karar verdik. Bu şekilde tespit edilen kutuyu resimden kırparak daha doğru bir sınıflandırma yapabiliriz.



Şekil 2.21 Açık Ağız Tespiti

2.3.1 Ağız Tespiti İçin Veri Toplama

Makine öğrenmesi ve derin öğrenme için kullanılan açık kaynak verisetlerini incelediğimizde direkt olarak bizim yapmak istediğimiz görev için etiketli veri seti bulamadık. Biz de farklı görevler için kullanılan verisetlerini kendi projemize uygun olacak şekilde ayıkladık ve bunları manuel olarak bounding box formatında etiketledik. Etiketleme yaparken **roboflow** sitesini kullandık. Roboflow, görüntü verilerini işlemek ve öğrenmek için çeşitli araçlar sunar. Amacımıza uygun diğer datasetlerden toplam 1380 tane veri topladıktan sonra bu verileri etiketlemeye başladık. Yaklaşık 500 tanesini manuel olarak etiketleyip, bu veriler ile bir tespit modeli geliştirdik. Kalan kısmı ise bu modeli kullanarak, roboflowun sağladığı labelassist özelliğiyle çok daha hızlı bir şekilde etiketledik.



Şekil 2.22 Verisetinden Örnekler

2.3.2 Ağız Tespit Modeli Geliştirme

Ağızın tespiti için nesne tespiti algoritmalarının en popülerlerinden oln YOLO(You Only Look Once) algoritmasını kullandık. YOLO sürekli güncelleme alan popüler bir algoritma olduğundan birçok versiyonu mevcuttur. Biz projemizde diğer modellere göre genelde daha yüksek performans sağlaması, kolay implemente edilebilir olması gibi nedenlerden dolayı en son versiyonlarından biri olan YOLOV8 algoritmasını tercih ettik.

Eğitimimizi COLAB platformumunun sağladığı ücretsiz T4 GPU donanımını kullanarak gerçekleştirdik. Eğitimi 25 epoch boyunca gerçekleştirdik. Eğitim ile ilgili grafiklerin bulunduğu grafiklerin bulunduğu resime göre ve testlerimize göre modelin oldukça iyi sonuçlar verdiğini söyleyebiliriz.

2.4 Backend Sunucusu Geliştirme

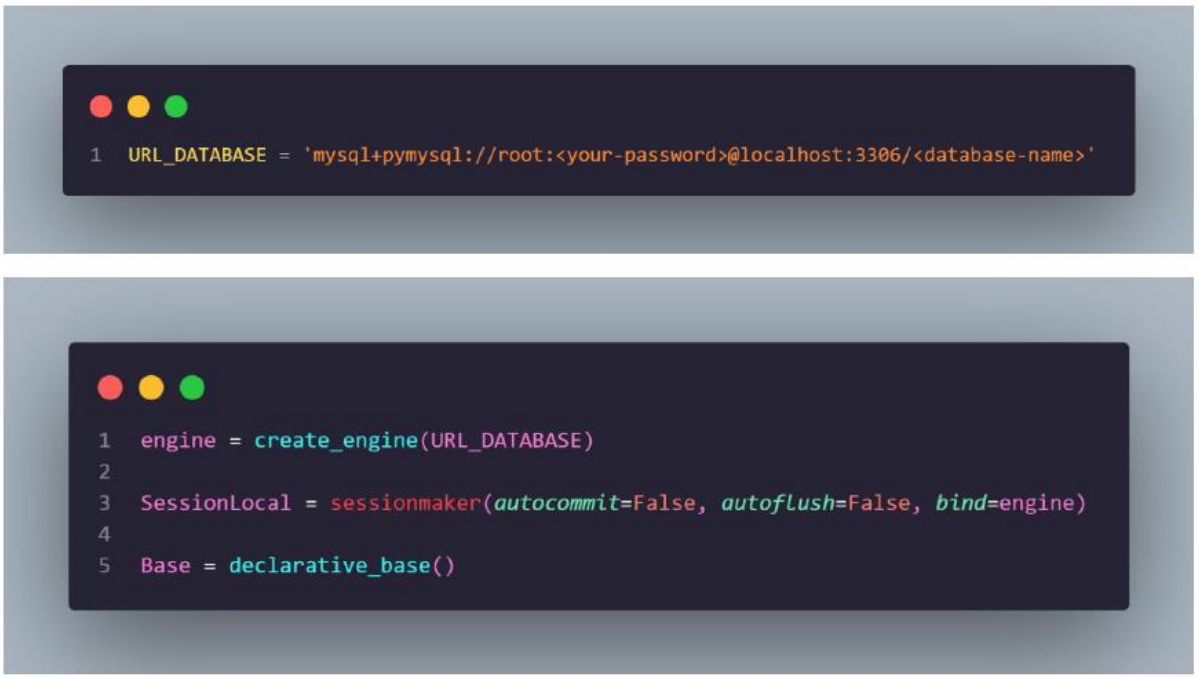
Projemizin tasarlandığı mimaride, backend sunucunun temel görevi, kullanıcıların mobil arayüz aracılığıyla sisteme gönderdikleri veri ve istekleri almak ve bunları eğitilmiş modele ileterek elde edilen sonuçları tekrar kullanıcı arayüzüne iletmektir.

FastAPI kütüphanesini kullanarak dinamik ve hızlı bir Restful API yapısı oluşturduk. Bu API, kullanıcı isteklerini alıp eğitilmiş modele ileterek elde edilen sonuçları etkili bir şekilde kullanıcı arayüzüne iletmekte başarılı oldu. Asenkron çalışma yeteneği sayesinde, yüksek performans elde ettik ve sistemimizin hızını artırdık.

Python diline tam entegrasyon sağlayan FastAPI'nin sunduğu esneklik ve kolaylık, geliştirme sürecini optimize etmemize ve projemizi daha verimli hale getirmemize yardımcı oldu.

2.4.1 Veritabanı Bağlantısı

Diş analizlerini uygulamada kişiye özel göstermemiz gerektiği için bir User nesnesi, yapılacak analizler için Analyze nesnesi tasarlamayı planladık. Modelden analiz sonuçları her hastalık için yüzdelik geldiği için hastalıkları tutmak için nesne tasarlamanın anlamsız olduğuna karar verdik. Onun yerine her bir hastalık için Analyze nesnesinde bir sütun ayırdık.
Veritabanı yönetim sistemi olarak MYSQL kullandık. Veri tabanımızı kurduğumuz API yapısına bağlamak için SQLAlchemy ve PyMySQL kütüphanesini kullandık.

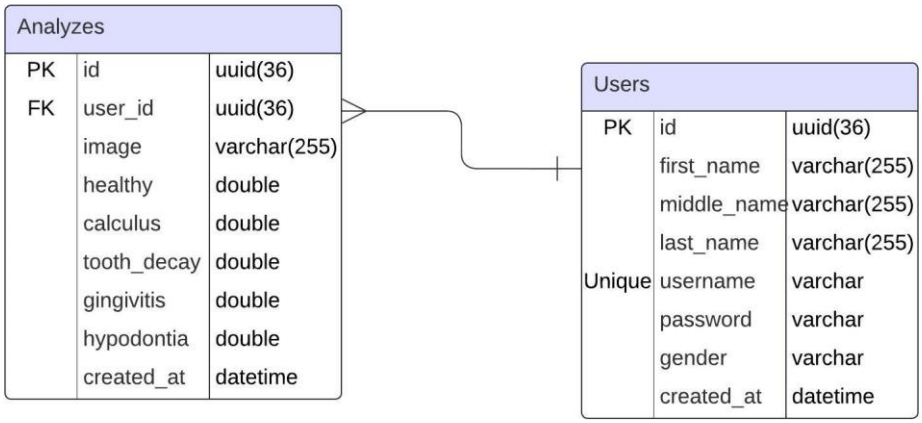


Şekil 2.23. Fast Api ve Veritabanı Bağlantısı

Yukarıdaki veri tabanı bağlama sorgusu ve kod yapısı ile veri tabanımız ile API'yı bağladık. API'de yapılan tablo değişiklikleri veri tabanında anlık olarak karşılık bulabilecek ve API-veri tabani bağlantısı dinamik olarak oluşturulacaktır.

2.4.2 Tablo Tasarımları

Tablo tasarımımız “users” ve “analyzes” tablolar olmak üzere toplam 2 tablodan oluşmaktadır.



Şekil 2.24 Tablo Tasarımları

User nesnemizde primary key olan id’yi **UUID** olarak tanımladık. Bu sayede kullanıcı id’leri eşsiz ve benzersiz oldu. Ayrıca şifreyi hashleyerek veri tabanında tuttuğum için veri tabanı güvenliği oldukça yükseldi.

Analyze nesnesinin primary key’ini **UUID** yapısında olması analizlerin eşsiz olması konusunda yardımcı oldu. Ayrıca her analiz sadece bir kişiye ait olabileceği için user_id’yi foreign key olarak tanımladım. Analiz için gerekli resmi ve modelden gelecek hastalık yüzdelerini ayrı ayrı sütunlarda tuttuk.

2.4.3 Yapay Sinir Ağı ve Server Bağlantısı

Eğitilmiş yapay sinir ağının modellenmiş hali olan .h5 uzantılı dosyayı dosya sistemimize ekledik. Daha sonra Tensorflow.keras kütüphanesini içinde bulunan load_model() metodu yardımıyla eğitilmiş modelimizi sistemimize tanıttık.

Şekil 2.25 Yapay Sinir Ağı ve Server Bağlantısı

2.4.4 Backend Uç Noktalarının(Endpoint) Tasarımı

2.4.4.1 User Methodları ve Uç Noktaları

User nesnelerinin şifrelerini veri tabanımızda olduğu gibi tutmak riskli olabileceğinden şifreleri hashleyerek tutuyoruz. Şifreleri hashlemek için ise PassLib kütüphanesini kullandık. Şifre girildikten sonra kullanıcıya girdiğine dair bir token vermek için ise PyJWT kütüphanesini kullandık.

Password Hashing: PassLib kütüphanesinin bir ürünü olan CryptContext'i kullanarak şifreleri hashledik. Şifreleri hashlemek için yukarıda da görüldüğü gibi bcrypt şemasını kullandık. Daha sonra yine aynı nesnenin verify() metodu ile şifreyi tekrardan kontrol edebildik.

Gerçek Şifre	Hashlenmiş Şifre
Beytullah.123	12eSEUs/.1c8SWhODLDiurB.japyEBUBpx.a0rM.bJEI6hnpBriRFrC

Şekil 2.26 Password Hashleme

Jwt Token Oluşturma: PyJWT kütüphanesi kullanarak kullanıcı için token ürettik. Bu tokenları üretirken HS256 algoritmasını kullandık. HS256, Jwt'lerin imzalanmasında ve doğrulanmasında kullanılan bir şifreleme algoritmasıdır. Aşağıdaki şekilde örnek bir erişim tokeni verilmiştir.


```
{
  "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJCZX10dWxsYWgxIn0.pWKA1tX1Q4BmKF9HHKrhez_LbdFq_kPWq3Ji0EFvGpU"
}
```


Şekil 2.27 Jwt Access Token

/register uç noktası: POST metodunu kullanarak tasarladığımız bu endpoint dışarıdan kullanıcı hakkında gerekli bilgileri alacak. Kullanıcının veri tabanında hali hazırda var olup olmadığını kontrol edip yoksa şifresini hashleyerek veri tabanına ekliyor.


```
{
  "id": "dcbdc88f-5e0a-4613-b79d-c6051e1f8c6c",
  "middle_name": "Hüseyin",
  "username": "hasan2",
  "gender": "male",
  "last_name": "Yayla",
  "first_name": "Hasan",
  "password": "$2b$12$4LR46gmEFpfOVTX66eJMWOxZuzn/TjEPCUuNpoEQjjVX..eKHKcHS",
  "created_at": "2024-01-08T20:46:07"
}
```


Şekil 2.28 Başarılı Register Uç Noktası Sonucu

/login uç noktası: POST metodunu kullanarak tasarladığım bu endpoint dışarıdan username ve password alıyor. Aldığı bilgilerle username ve sifrenin doğruluğunu kontrol ediyor. Doğru olduğu takdirde yeni bir JWT Token oluşturup kullanıcıya dönüyor.


```
{
  "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJoYXNhbjIifQ.tayYi6btB35_T1YnxzxyXXTJY4wtS8SHJCx_pN9L4FE"
}
```


Şekil 2.29 Başarılı Login Uç Noktası Sonucu

2.4.4.2 Analyze Uç Noktaları

/analyzes uç noktası: Bu fonksiyonda POST metodunu kullandım. Bu endpointi çağırmak için userId ve resim dosyasına ihtiyacımız var. Kullanıcıdan aldığı resmi ilk önce daha önceki bölümlerde bahsettiğimiz ağız tespit modeline veriyor ve dişlerin gözüktüğü ağız kırpılarak daha sonra bu resim üzerinden sınıflandırma yapılıyor. Gelen cevapları, kullanıcıdan alınan userId ve image'i veri tabanına kaydediyor.


```
{
  "healthy": 0.42,
  "calculus": 0.03,
  "tooth_decay": 98.48,
  "gingivitis": 0.06,
  "hypodontia": 1.01,
  "description": "Ne yazık ki dişlerinizde çürük var gibi görünüyor. Çürük, diş minenizde asit salgıları ve bakteri etkisiyle oluşan hasar ve çökmeye verilen isimdir. Çürüğün
olarak size uygulanacak tedavi de değişiklik göstermektedir. Bunun için lütfen daha fazla zaman geçirmeden bir profesyonele başvurunuz."
}
```


Şekil 2.30 Başarılı Analyzes Uç Noktası Sonucu

/analyzes/last uç noktası: Bu fonksiyonda GET metodunu kullandım. userId'ye çoklu analiz listesi veya analyzeId'ye göre tekli analiz döndürebilecek bir GET metodu tasarladım. Bu metod sayesinde kişilerin analiz geçmişlerini gösterebilecek ve analizler hakkında detaylı bilgi almak istendiğinde kullanıcılara detaylı bilgi verebileceğiz.

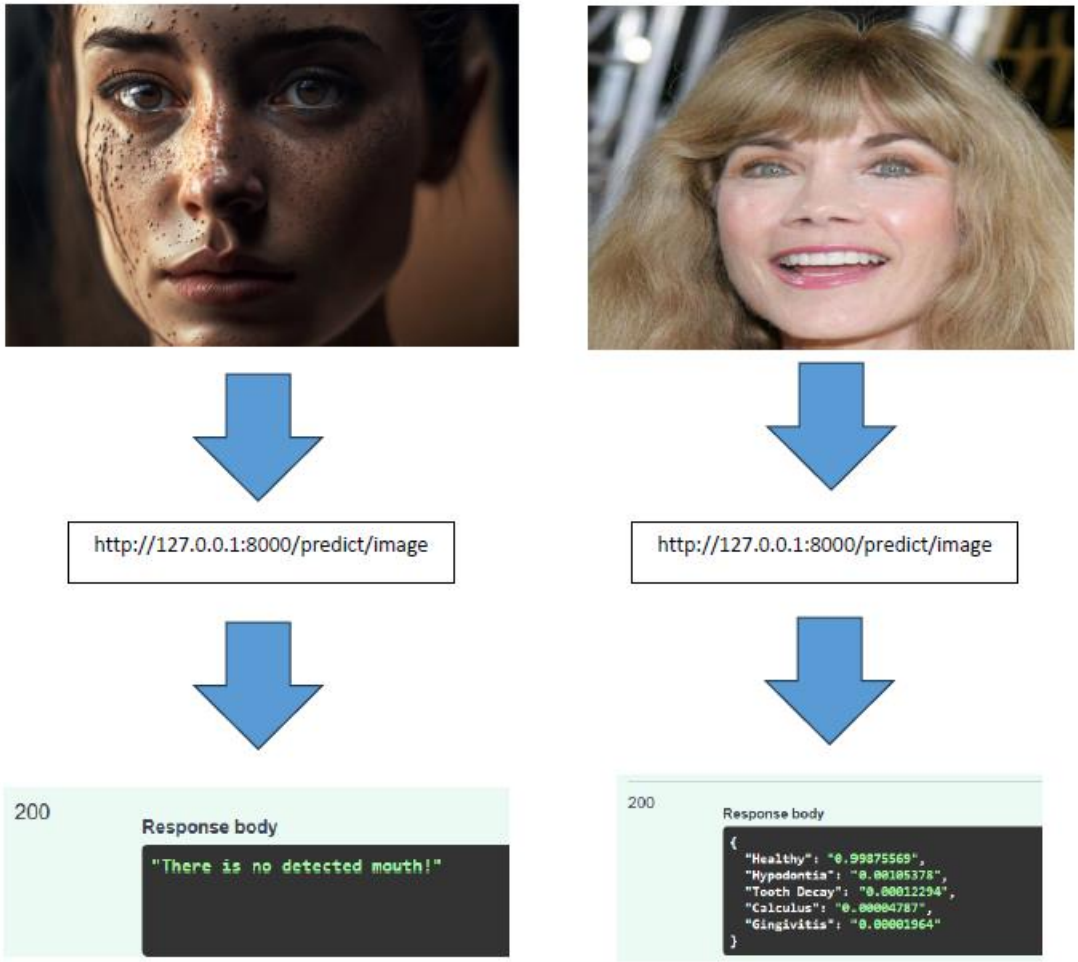

```
[
  {
    "id": "d6a2d0b8-2ab7-4a99-b274-31fe7c481a04",
    "image": "<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=612x403 at 0x1FD878F61C0>",
    "healthy": 7.97,
    "calculus": 11.01,
    "tooth_decay": 1.27,
    "gingivitis": 78.64,
    "hypodontia": 1.11,
    "created_at": "2023-12-28T11:19:41",
    "description": "Ne yazık ki diş etlerinizde iltihap var gibi görünüyor. Diş eti iltihabi, bakteri etkisiyle diş etlerinizin iltihaplanması sonucu ortaya çıkar; kızarık
gibi belirtiler gösterebilir. Diş eti iltihabini düzenli diş bakımı (fırçalama, ip kullanma, antiseptik gargara…) ile çözebilirsiniz. Eğer iltihap ilerlediyse diş hekimini
ğru olacaktır."
  }
]
```


Şekil 2.31 Başarılı /analyzes/last Uç Noktası Sonucu

2.4.5 Ağız Tespit Modelinin Backend Sunucusuna Uygulanması

Ağız tespiti için oluşturduğumuz modeli kaydedip backend serverda **serve_model.py** dosyasında ilk olarak Yolo modelinin yüklendiği **load_yolo_model** adında bir method yazdık. Bu methotta temel olarak kaydettiğimiz modelin dosya yolunu kullanarak bir model oluşturuyoruz. Daha sonrasında parametre olarak resim alan bir **mouth_detection_yolo** adında bir method yazdık. Bu fonksiyon aslında gelen load_yolo_model methodunu kullanarak modeli yükledikten sonra resim üzerinden **confidence_threshold** dediğimiz skoru **0.7**'den büyük veya eşit olan kutuların koordiatlarını bize döndürür.

Api kısmında ise yaptığımız şey temel olarak method_detection_yolo methodunu kullanarka frontendden gelen resimden ağız koordinatlarını bulduktan sonra resimden bu koordinatların olduğu kısmı kırpar ve bu kırpılmış resim üzerinden sınıflandırma yapmamıza olanak verir. Ayrıca resimde ağız tespit edilememişse hiç sınıflandırma yapmaz ve **"There is no detected mouth!!"** cevabını string
olarak döndürür. Örneğin aşağıdaki resimde herhangi bir diş görünen ağız tespit edemeyeceği için sınıflandırma işlemi yapmaz ve bir mesaj döner.



Şekil 2.32 Açık Ağız Tespiti API

2.4.6 Backend Server İçin Docstring'lerin Yazılması

Hepimizin bildiği gibi bir projenin en önemli aşamalarından biri projenin iyi bir şekilde dökümante edilmesidir. Biz de özellikle projemizde methodların açıklamalarının olmasına ve dökümantasyona dikkat etmeye çalıştık.

Docstring yazmanın bizim için avantajları olacağını düşünüyoruz. Bunlardan kısaca bahsetmek gerekirse:

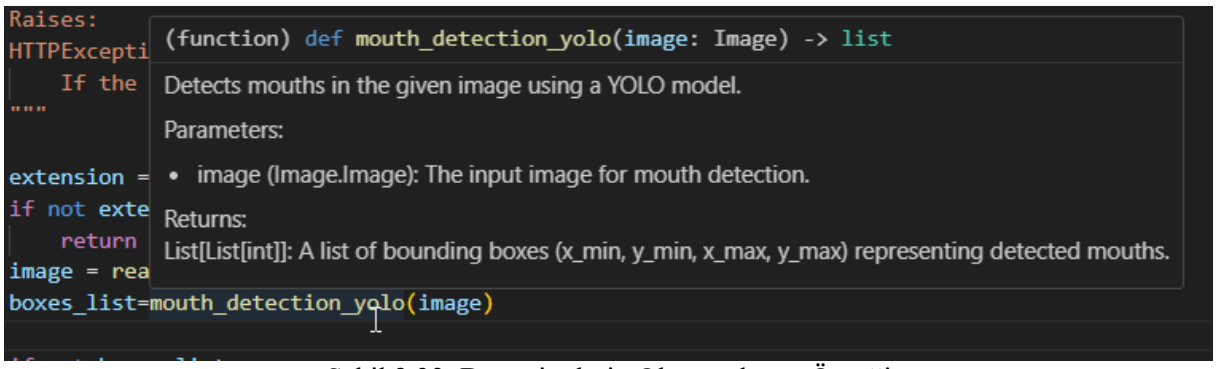
- **Kod Belgelendirmesi ve Anlaşılabilirlik:** Docstring'ler, kodumuzu okuyan diğer geliştiricilere, kodunuzun amacını, kullanımını ve iç işleyişini anlatma imkanı sağlar. Kodunuzun üzerinde çalışanlar veya projenizi devralan geliştiriciler, docstring'leri okuyarak işlevselliği daha hızlı anlayabilir ve kullanabilir.

- **Otomatik Belgelendirme Olanakları:** Docstring'ler aynı zamanda otomatik belgelendirme araçlarının kullanılmasını sağlar. Örneğin, Sphinx gibi araçlar, docstring'lerden belgeler üretebilir. Bu belgeler, projenizin API'si veya modüllerinin resmi belgeleri olabilir ve dış dünyaya projenizi tanıtmak için kullanılabilir.

- **Test ve Debug Desteği:** Docstring'ler, bir fonksiyonun beklediği parametreleri, döndüğü değerleri ve genel çalışma mantığını açıklar. Bu bilgiler, fonksiyonları test etmek ve hataları ayıklamak için kullanıldığında oldukça faydalıdır.

- **Geliştiriciler Arası Etkileşim:** Docstring'ler, sadece bilgisayarlarla değil, aynı zamanda diğer insanlarla iletişim kurmanın bir yolu olarak da düşünülmelidir. İyi yazılmış docstring'ler, takım içinde daha etkili bir iletişim sağlar ve kodun anlaşılabilirliğini artırır.

Bir fonksiyonun döndürdüğü değerleri, ne işe yaradığını öğrenmek isteyen biri aşağıdaki örnekteki gibi fonskiyonun üzerine gelerek, fonksiyonun hakkında bilgi sahibi olabilir.



Şekil 2.33. Docstringlerin Oluşturulması Örneği

2.5 Ngrok İle Api Tünelleme

Backend sunucumuzu dış dünyaya açmak için Ngrok aracını kullandık. Ngrok gibi araçlar genel olarak aslında halka açık bir url oluşturarak yerel bir sunucuya gelen talepleri yönlendirir ve bu sayede dış dünya ile güvenli bir bağlantı sağlar.Önceki raporda bahsettiğimiz backend ve frontend uygulamalarının haberleşememesi hatasının buradan kaynaklandığını farkettik ve tunelling işleminden sonra frontend kısmında tünellenmiş adrese istek attığımızda sorunun çözüldüğünü gördük. Aşağıdaki figürde biraz daha somut bir şeyler ortaya koymak adına hem tünelleme yapısını hem de sistemin genel yapısını anlatan bir diyagram oluşturduk.

Şekil 2.34 Ngrok İle Api Tünelleme

2.6 Frontend ve Api Haberleşmsi

Ngrok yardımı ile tünellediğimiz backend adresimizi axios'a vererek istek yollamak istediğimiz adreslere istek yollamaya başladık. Login işlemi için API'ye username ve password içeren bir bir post isteği yolladık karşılığında ise bir token nesnesi aldık. Kullanıcı burada doğru username ve password girerse Home sayfasına ulaşabilmektedir.

```
axios
  .post(`https://db33-176-55-94-231.ngrok-free.app/login`, {
    username: username,
    password: password,
  })
  .then((res) => {
    if (res.status === 200) {
      navigation.navigate("Home");
    } else {
      Alert.alert("Hata", "Kullanıcı adı veya şifre yanlış.");
    }
  })
  .catch((err) => {
    console.log(err);
  });
```

Şekil 2.35 Login işlemi sırasında kullanılan axios post isteği

Register işlemi için API'ye firstname, middlename, lastname, username, password ve gender içeren bir bir post isteği yolladık. Kullanıcı burada gerekli bilgilerin hepsini girerse Login sayfasına tekrar yönlendirilir.

```
axios
  .post(`https://db33-176-55-94-231.ngrok-free.app/register`, {
    first_name: firstname,
    middle_name: middlename,
    last_name: lastname,
    username: username,
    password: password,
    gender: gender,
  })
  .then((res) => {
    console.log(res);
    console.log(res.status);
    if (res.status === 201) {
      navigation.goBack();
    } else {
      Alert.alert("Hata", "Lütfen gerekli tüm alanları doldurunuz.");
    }
  })
  .catch((err) => {
    console.log(err);
  });
```

Şekil 2.35 Register işlemi sırasında kullanılan axios post isteği

Predict işlemi için API'ye resmin uri, tip ve isminden oluşan FormData içeren bir bir post isteği yolladık. Username ve FormData ile API'ye giden istek geriye modelin verdiği predict sonucunu döndürür.

```
const postAnalyze = () => {
  const formData = ImageToFormData(imageTeeth);
  axios.post(`https://db33-176-55-94-231.ngrok-free.app/analyzes?username=${username}`,
    formData,
    {
      headers: {
        'Content-Type': 'multipart/form-data', // Set the content type for FormData
      },
    }
  ).then((res) => {
    if (res.status === 200) {
      setNames(Object.keys(res.data))
      setValues(Object.values(res.data))
      reverseAnalyzed()
    } else {
      Alert.alert("Hata", "Beklenmeyen bir hata oluştu.");
    }
  }).catch((err) => {
    console.log(err + " " + err.response);
  });
```

Şekil 2.37 Predict işlemi sırasında kullanılan axios post isteği

2.7 Mobil Uygulama Tasarımı

Bu bölümde login,register, tahmin ve daha önceki analizleri gösteren sayfaların tasarımını yaptık. Kullanıcı deneyimini arttırmak adına minimal ve şık bir tasarım anlayışıyla hareket ettik. Tasarımın her aşamasında kullanıcı odaklı düşündük ve geliştirdik. Kullanıcılarımızın uygulamamızı daha rahat, etkileşimli ve kullanışlı bulmalarını sağlamak için geribildirimlerini alıp uygulama tasarımımızı optimize etmeye çalıştık.

Uygulamamızın API ile bağlantı kurması için en çok tercih edilen Javascript kütüphanelerinden olan Axios'u tercih ettik. ***Axios,*** modern web tarayıcılarında ve Node.js'de kullanılan bir JavaScript kütüphanesidir. Axios, XMLHttpRequest (XHR) ve Fetch API ile aynı işlevi görür, ancak daha kolay bir kullanıma sahiptir ve promise tabanlıdır.

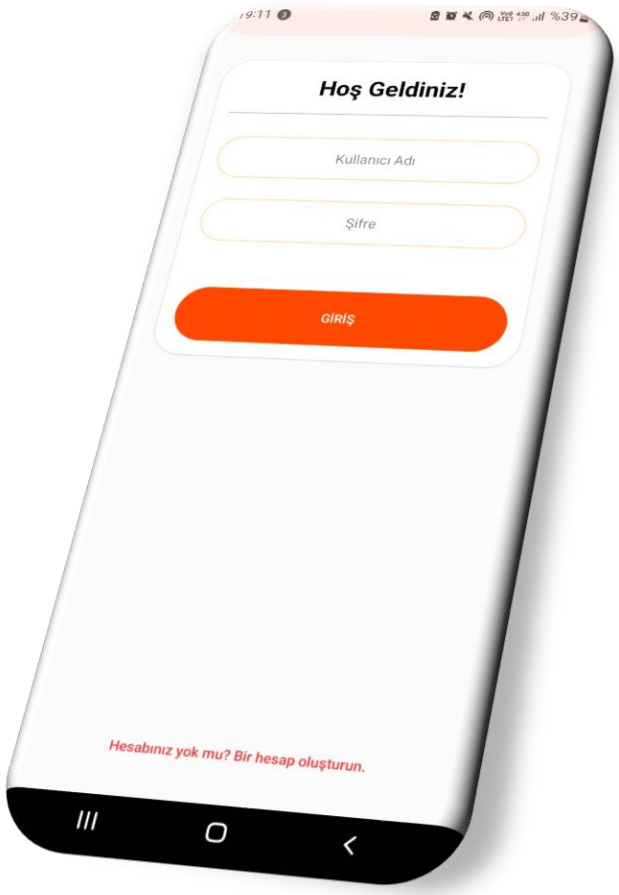
Uygulamamızda yapılacak işlemler kullanıcı bazlı olduğu için kullanıcı uygulamaya giriş yaptığında kullanıcın unique username'ini saklayabilmek ve diğer componentlerda ihtiyaç olduğunda kullanabilmek için react kütüphanesi tarafından sunulan ***useContext*** hookunu kullanmayı planladık. useContext, React'te yer alan bir hook'tur ve uygulamanın farklı bileşenleri arasında veri iletmeyi kolaylaştırır. Özellikle global state'e erişim sağlamak için kullanılır. Bu, veri iletimini bileşenler arasında manuel olarak iletmek zorunda kalmadan yapmamıza olanak tanır.

Login, Register ve Entry sayfalarında kullanacak görselleştirmeler için ***react-native-elements*** ve ***react-native-svg*** kütüphanelerinden yardım almayı planladık. react-native-elements Button, Image, Card gibi uygulamada kullanılacak parçalarını daha kullanışlı ve özellik havuzunu

genişleterek sunan bir UI kütüphanesidir. react-native-svg'yi projede kullanma amacımız ise kütüphanenin bize sunduklarını kullanarak analiz sonuçlarını anlaşılabilir grafiklerle kullanıcıya sunma isteğimizdir.

2.7.1 Login Sayfası Tasarımı

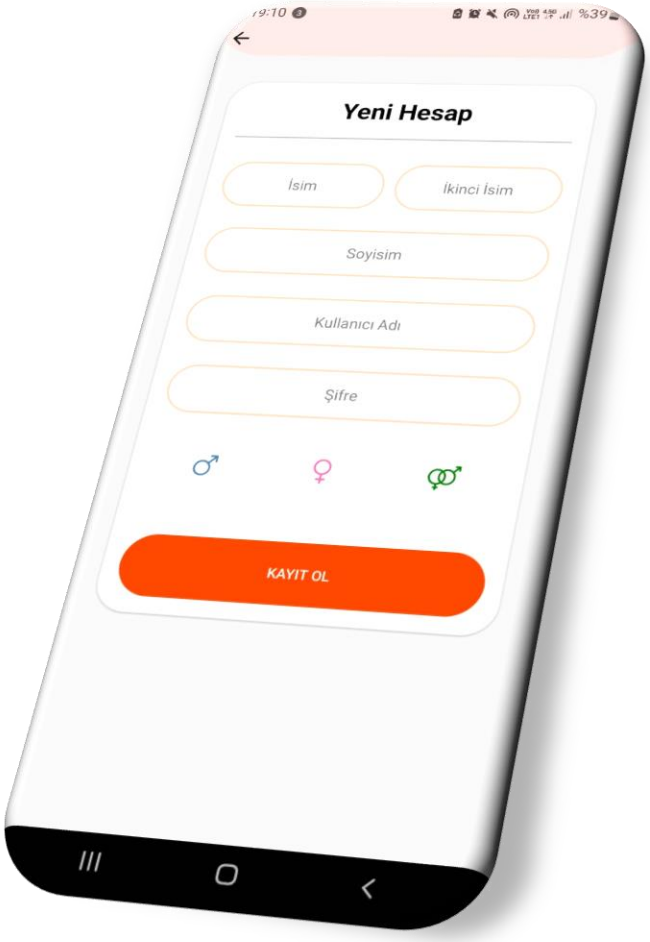
Login sayfamızda Username ve Password için 2 input girişi ve Login işlemi yapılması için bir Login butonu bulunmaktadır. Aşağıdaki yazıya tıklayarak yeni hesap açma sayfasına giriş yapılabilmektedir.



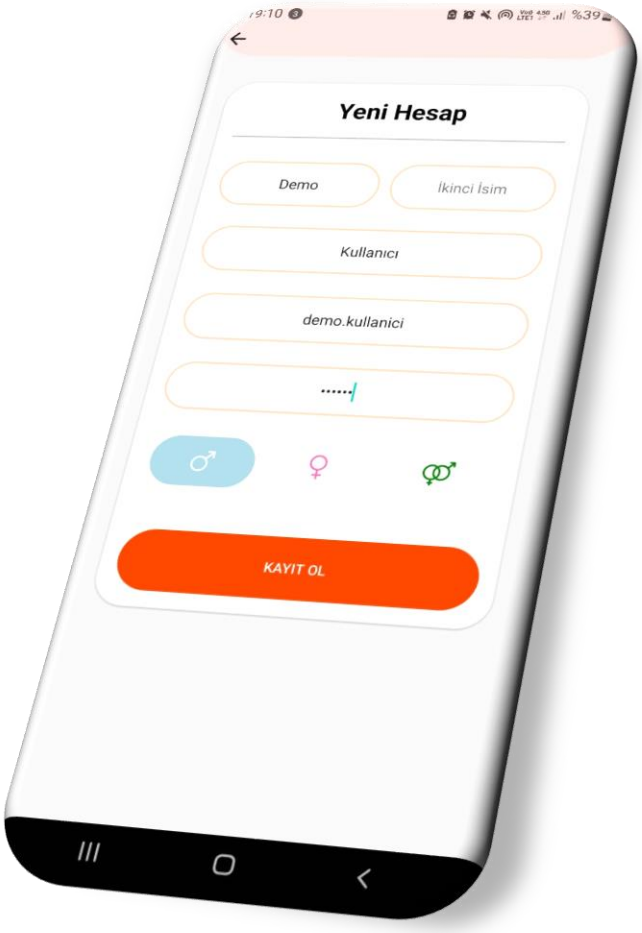
Şekil 2.38. Login Sayfası Tasarımı

2.7.2 Register Sayfası Tasarımı

Register sayfamızda First Name, Middle Name, Last Name, Username, Password ve Gender için input girişleri ve Signin işlemi yapılması için bir Signin butonu bulunmaktadır. Başarılı giriş yapıldıktan sonra veriler veri tabanına kaydedilir ve tekrar Login sayfasına dönülür.



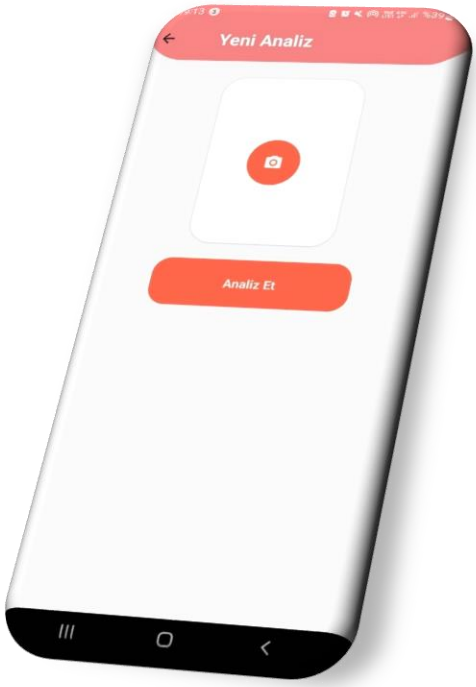
Şekil 2.39 Register Sayfası Tasarımı



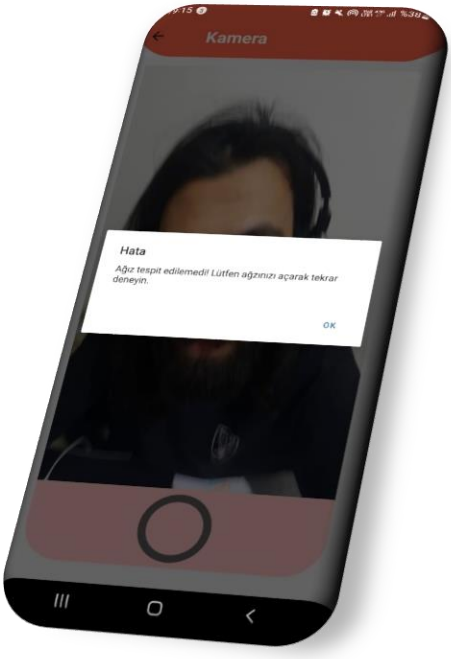
Şekil 2.40 Register Sayfası Örnek Kullanıcı Kaydı

2.7.3 Predict Sayfası Tasarımı

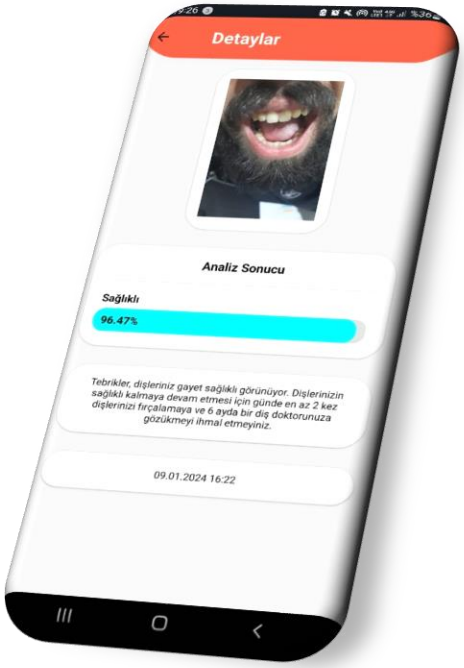
Entry sayfamızda kullanıcıdan aldığımız sonuçlar API'ye post edilip dönen predict sonucu alınır. Alınan predict sonucu aşağıdaki görünen yatay barlara dağıtılır. Predict sonucu belirlediğimiz eşik değerinden az gelen sonuçlar barlara konulmayacak şekilde ayarlanmıştır.



Şekil 2.41 Örnek Analiz Sayfası



Şekil 2.42 Kamera Ağız Tespit Edemedi



Şekil 2.43 Örnek Detay Sayfası

2.7.4 Geçmiş Sayfası Tasarımı

Ana sayfadaki buton yardımıyla ulaşılabilen history sayfasında bir ScrollView içerisinde geçmiş testleri ve sonuçları görebiliriz. Kullanici bu sayfada geçmiş analizlerini kısa özeti ile görebileceği gibi içlerine girerek detaylarını da görebilecektir.

Şekil 2.44.Geçmiş Sayfası

3. BULGULAR

3.1 Sınıflandırma Modelinin Değerlendirilmesi

Ağız hastalıklarının sınıflandırılması için geliştirdiğimiz modeli değerlendirirken, sınıflandırma görevi değerlendirme metriklerinden doğruluk(accuracy), precision(hassasiyet), recall ve f1 skoru metriklerini kullandık. Bunun için önceki adımlarda da anlattığımız K-fold cross validation ile eğitim aşamasında her fold sonunda o fold için test verisi olarak kullandığımız veriler üzerinden tahminler yaptık ve yukarıda söylediğimiz metrikleri hesaplayıp o metrikle alakalı listeye ekledik. Daha sonrasında da ortalama bir değer bulmak için listedeki değerlerin ortalamasını aldık.

Model Doğruluk Değeri(Accuracy): Bu metrik makine öğrenimi modelinin verilen girdilere ne kadar doğru bir şekilde tahmin yapabildiğini ölçen bir metriktir. Genellikle doğru tahmin edilen örneklerin toplamının, toplam örnek sayısına oranı olarak ifade edilir. Bu oran genellikle yüzde olarak ifade edilir.

$$\boldsymbol{Accuracy} = \frac{\boldsymbol{Doğru\ Tahminler}}{\boldsymbol{Toplam\ Örnek\ Sayısı}}$$

Modelimizin ortalama validasyon doğruluğunu değerlendirdiğimizde bu değerin **0.85** civarında olduğunu gördük. Yani ortalama 100 tane resimden 85 tanesini doğru olarak sınıflandırdığını söyleyebiliriz.

Hassasiyet(Precision): Model hassasiyeti, pozitif olarak tahmin edilen örneklerin gerçekte ne kadarını tahmin ettiğini gösteren bir metriktir.

$$\boldsymbol{Precision} = \frac{\boldsymbol{True\ Positives}}{\boldsymbol{True\ Positives + False\ Positives}}$$

True positives, gerçekte pozitif olan örneklerin doğru şekilde pozitif olarak tahmin edilen sayı.

False positives, gerçekte negatif olan örneklerin yanlış bir şekilde pozitif olarak tahmin edilen sayı.

Genellikle bir sınıf için yapılan tahminlerin ne kadar güvenilir olduğunu değerlendirmeye yarayan bir metriktir. Bizim sınıflandırma modelimizin ortalama **precision değeri 76.8** olarak hesaplanmıştır.

Duyarlılık(Recall): Bir sınıfın gerçek pozitif örneklerinin ne kadarını doğru bir şekilde tahmin ettiğini ölçen bir performans metriğidir.

$$\boldsymbol{Recall} = \frac{\boldsymbol{True\ Positives}}{\boldsymbol{True\ Positives + False\ Negatives}}$$

True positives, gerçekte pozitif olan örneklerin doğru şekilde pozitif olarak tahmin edilen sayı.

False negatives, gerçekte negatif olan örneklerin yanlış bir şekilde negatif olarak tahmin edilen sayı.

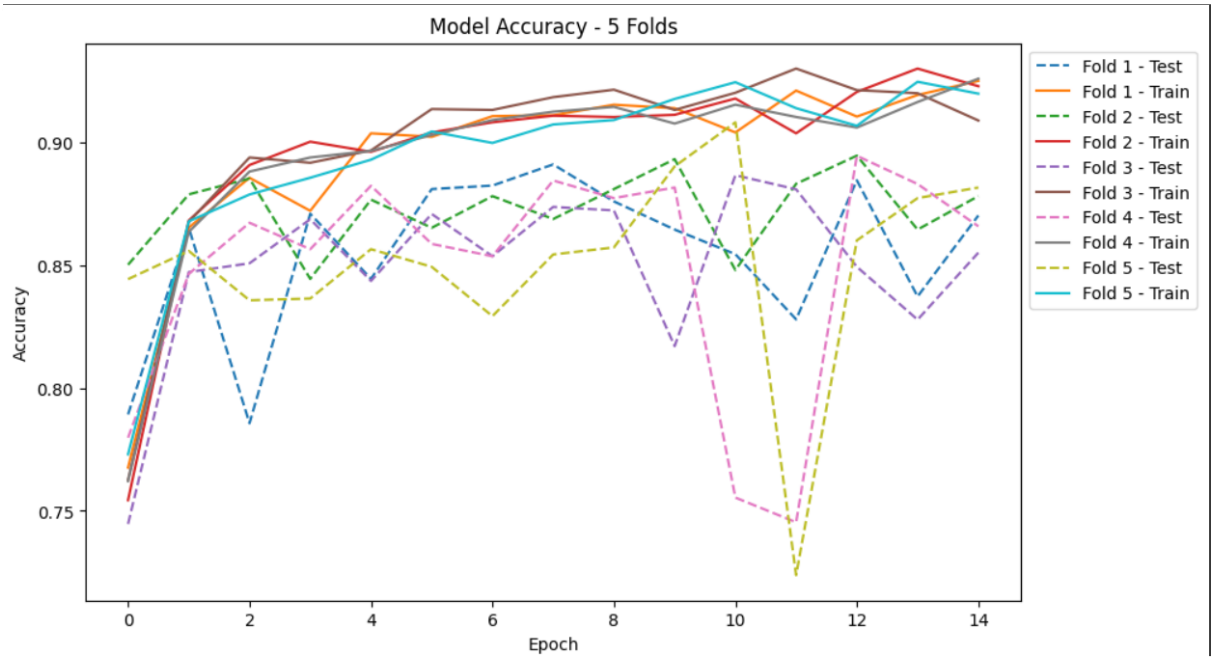
Recall, bir sınıfın ne kadarını kaçırma eğiliminde olduğunu gösterir. Yüksek recall değerleri, gerçek pozitif örneklerin büyük bir kısmının doğru bir şekilde tahmin edildiği anlamına gelse de modelin false positive sayısını arttırma eğiliminde olduğunu gösterebilir. Bizim sınıflandırma modelimizin ortalama **recall değerini 64.2** olarak elde ettik.

F1-Score: Bu metrik sınıflandırma modelinin precision ve recall değerlerini birleşitiren bir metriktir. Genellikle precision ve recall arasındaki dengeyi değerlendirmek için kullanılır.

$$\boldsymbol{F1 = 2 * \frac{Precision * Recall}{Precision + Recall}}$$

Dengesiz sınıflara sahip veri setleri olduğu durumlarda bu metrik kullanışlıdır. Yüksek f1 skoru, hem false positive hem de false negative azaltan bir modeli ifade eder. Bizim projemizde elde ettiğimiz ortalama **f1 skorumuz 62.6'dır.** Bu skora göre modelimizin iyi olmayan ortalama bir preformans gösterdiği sonucunu çıkarabiliriz.

Modelimizi değerlendirirken her epochta validation ve train accuracy değerlerini hesaplayıp bir listede daha sonra görselleştirmek üzere tuttuk. Bu şekilde modelimizin performansını görselleştirebiliriz.



Şekil 3.1 Her fold için validasyon doğruluklarının görselleştirilmesi

Bu şekilde model doğruluğunu her epoch için çizgi grafiği olarak görselleştirdiğimizde eğitim başarısının daha tutarlı bir şekilde ilerlediği, doğrulama skorunun ise çok fazla dalgalandığını görüyoruz.

3.1 Açık Ağız Modelinin Değerlendirilmesi

Bu modeli değerlendirirken mAP(mean average precision), precision ve recall metriklerini temel aldık.

mAP: İlk olarak, her sınıf için ayrı hassasiyet ve duyarlılık değerleri hesaplanır. Ardından bu değerler kullanılarak ortalama hassasiyet hesaplanır. Bir modelin birden çok sınıftaki performansını tek bir sayıda özetler.Eğitim sonunda %99.1'luk bir mAP değeri elde ettik.

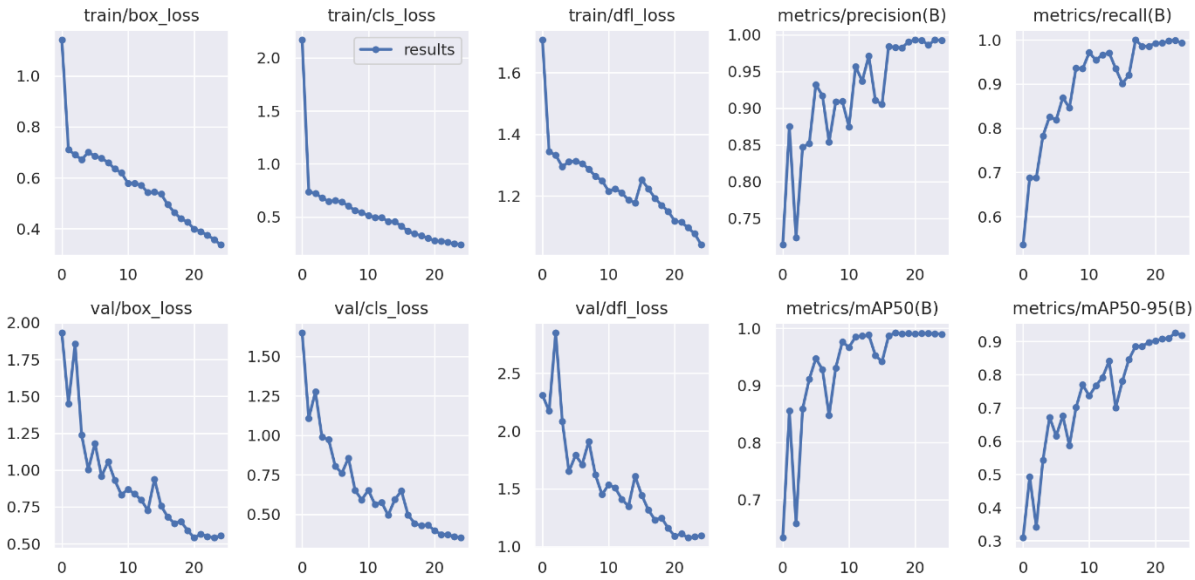
Precision(Hassasiyet): Bir modelin pozitif olarak tahmin ettiği örneklerin gerçekte pozitif olma olasılığını gösterir. Yüksek hassasiyet, modelin yanlış pozitifleri sınırlı bir şekilde tahmin ettiği anlamına gelir. Eğitim sonunda %99.3'lük bir precision değeri elde ettik.

Recall(Duyarlılık): Duyarlılık, gerçekten pozitif olan örneklerin ne kadarının doğru bir şekilde tahmin edildiğini gösterir. Yüksek duyarlılık, modelin pozitif örnekleri daha iyi kapsadığı anlamına gelir. Eğitim sonunda %99.9'luk bir recall değeri elde ettik.

Bounding Box Loss(bbox_loss): Nesnelerin sınırlayıcı kutularının (bounding box) koordinatlarını belirleyen kayıptır. Bu, ağın nesneleri doğru bir şekilde yerelleştirip yerelleştirmediğini değerlendirmek için kullanılır.

Classification Loss(cls_loss): Nesnelerin sınıflarını belirleyen kayıptır. YOLO, her hücredeki nesnelerin sınıfını ve bir hücrenin birden fazla sınıf içermesi durumunda en olası sınıfı tahmin eder. cls_loss, bu sınıflandırma tahminlerinin gerçek etiketlerle karşılaştırılmasıyla hesaplanır.

Bu değerleri incelediğimizde modelimizin oldukça başarılı bir şekilde açık ağızları tespit edebildiğini söyleyebiliriz.



Şekil 3.2 Ağız tespiti metriklerinin görselleştirilmesi

4. SONUÇLAR VE ÖNERİLER

Bu çalışmada sağlık alanında baştan sona bir bilgisayarlı görü uygulaması geliştirmeyi amaçladık.

Geliştirilen bu projede, telefon kamerasıyla alınan bir ağız görüntüsünden ağız hastalığı sınıflandırması başarılı bir şekilde yapılmıştır. Bu, teşhis sürecini biraz olsun hızlandırabilir ve tedaviye daha erken başlanmasına yardımcı olabilir.

Telefon kamerasıyla yapılan bu tür teşhisler, hastaların kendi evlerinde ağız sağlıklarını izlemelerine ve erken uyarılar alabilmelerine olanak sağlayabilir. Bu sağlık hizmetlerine daha kolay bir erişim sağlayabilir ve maliyeti düşürebilir.

Bu çalışma bilgisayarlı görü teknolojilerinin ağız sağlığı sektöründe nasıl etkin bir rol oynayayabileceğini göstermektedir. Benzer uygulamalar farklı sağlık sektörleri için de geliştirilebilir.

Öneriler:

- **Genişletilmiş Veri Seti:**Özellikle veri setindeki bazı görsellerde birden fazla hastalığın aynı anda bulunması, gerçeğe uygun olmayan görsellerin verisetinde bulunması, bazı sınıflar için dengesizliklerin bulunması sınıflandırma modelimizin performansını olumsuz etkilediğini düşünüyoruz. Özellikle bu alanda uzman bir hekimden yeni bir verisetinin oluşturulması veya mevcut verisetinin iyileştirilmesi için yardım alınabilir. Veri kümesi özellikle daha az örneğe sahip sınıflar için arttırılabilir. Modelin daha geniş bir veri kümesi üzerinde eğitilmesi, daha çeşitli ağız hastalıklarını tanıma yeteneğini artırabilir.
- **Tensorflow Lite Kullanımı:**Tensorflow modelimizi tensorflow lite modeline çevirerek modelin tahmin yapma süresi azaltılabilir. Fakat bu doğruluk değerinde bir miktar azalmaya sebep olacaktır.
- **Ağız Tespit Sonuçlarının Mobil Uygulamada Gerçek Zamanlı Gösterilmesi:**Kullanıcının kamera ile dişlerini çekerken gerçek zamanlı bir şekilde modelimizin ağzı tespit edip dikdörtgen içine alması sağlanılabilir. Bu şekilde kullanıcı da hangi resim üzerinden sınıflandırma yapıldığını gözlemleyebilir.
- **Mobil Uygulama Fonksiyonlarının Genişletilmesi:**Mobil uygulama kısmında sadece ön dişlerden değil de arka dişlerden de tahmin veya çürük diş tespiti gibi işlevlere sahip olan fonksiyonlar geliştirilebilir. Buna ek olarak röntgen resimlerinden çürük diş tespiti yapılan bir model geliştirilip uygulamamıza entegre edilebilir.
- **Klinik Doğrulama:**Geliştirdiğimiz uygulamamızı gerçek senaryolarda test etmek oldukça önemli bir adımdır. Geliştirilen uygulamanın klinik ortamlarda gerçek hastalar üzerinde test edilmesi, güvenilirlik ve doğruluk açısından önemlidir.
- **Farklı Kullanıcı Arayüzlerinin Geliştirilmesi:**Mobil uygulamanın yanında masaüstü ve web uygulamaları geliştirerek sadece mobil uygulamaya bağımlı kalmanın önüne geçilebilir.

ÖZGEÇMİŞ

Beytullah Yayla, 2002 Sakarya doğumludur. İlk ve orta öğrenimini Sakarya'da tamamlamış olup şuan Sakarya Üniversitesi Bilgisayar Mühendisliği 4. Sınıf öğrencisidir. Daha önce Ford Otosan'da derin öğrenme stajı yapmıştır. Şu an Koçtaş'ta proje stajyeri olarak çalışmaktadır.

Mehmet Oğuz Özkan, 2001 Adana doğumludur. İlk ve orta öğrenimini Osmaniye'de tamamlamış olup şuan Sakarya Üniversitesi Bilgisayar Mühendisliği 4. Sınıf öğrencisidir. Şu anda Eczacıbaşı Bilişim şirketinde staj yapmaktadır.

REFERANSLAR

[1] https://medium.com/novaresearchlab/öğrenme-aktarımı-transfer-learning-c0b8126965c4

[2] François Chollet “Xception: Deep Learning with Depthwise Separable Convolutions”

[3] Muhamad Yani , Budhi Irawan, S, Si., M.T. and Casi Setiningsih, S.T., M.T. Setiningsih “Aplication of Transfer Learning Using Convolutional Neural Network Method for Early Detection of Terry’s Nail”

[4] https://www.researchgate.net/figure/ReLU-function-graph_fig2_346250677

[5] https://insideaiml.com/blog/SoftMaxActivation-Function-1034

[6] Diederik P. Kingma, Jimmy Ba “Adam: A Method For Stochastic Optimization”

[7] https://www.v7labs.com/blog/overfitting-vs-underfitting