

T.C.
FIRAT ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ

DERİN ÖĞRENME KULLANILARAK
GÖZ HASTALIKLARI TESPİTİ

Mehmet Tuna SELVİ
Esra GÜLMEZ

Tez Danışmanı:
Dr. Öğr.Üyesi Erdal ÖZBAY

BİTİRME TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

ELAĞİĞ
2023

T.C.
FIRAT ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ

DERİN ÖĞRENME KULLANILARAK GÖZ
HASTALIKLARI TESPİTİ

Mehmet Tuna SELVİ
Esra GÜLMEZ

BİTİRME TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

Bu bitirme tezi/....../2023 tarihinde, aşağıda belirtilen jüri tarafından oybirliği/oyçokluğu ile başarılı/başarısız olarak değerlendirilmiştir.

(İmza)

Danışman

(İmza)

Üye

(İmza)

Üye

ÖZGÜNLÜK BİLDİRİMİ

Bu çalışmada, başka kaynaklardan yapılan tüm alıntılar, ilgili kaynaklar referans gösterilerek açıkça belirtildiğini, alıntılar dışındaki bölümlerin, özellikle projenin ana konusunu oluşturan teorik çalışmaların ve yazılım/donanımın benim tarafımdan yapıldığını bildiririm.

Fırat Üniversitesi
Bilgisayar Mühendisliği
23119 Elazığ

22.06.2023
Mehmet Tuna SELVİ
Esra GÜLMEZ

BENZERLİK BİLDİRİMİ

Tez

ORJİNALLİK RAPORU

% 14	% 13	% 2	% 9
BENZERLİK ENDEKSİ	İNTERNET KAYNAKLARI	YAYINLAR	ÖĞRENCİ ÖDEVLERİ

BİRİNCİL KAYNAKLAR

1	Submitted to Fırat Üniversitesi Öğrenci Ödevi	% 3
2	Submitted to The Scientific & Technological Research Council of Turkey (TUBITAK) Öğrenci Ödevi	% 2
3	www.researchgate.net İnternet Kaynağı	% 1
4	ichi.pro İnternet Kaynağı	% 1
5	aws.amazon.com İnternet Kaynağı	% 1
6	acikerisim.sakarya.edu.tr İnternet Kaynağı	% 1
7	medium.com İnternet Kaynağı	% 1
8	www.ubcakcongress.org İnternet Kaynağı	% 1
9	Submitted to University of Monastir Öğrenci Ödevi	<% 1

TEŞEKKÜR

Bu çalışmanın gerçekleştirilmesinde değerli bilgilerini bizimle paylaşan, saygı değer danışman hocamız; Dr. Öğr. Üyesi Erdal ÖZBAY' a, çalışma süresince tüm zorlukları benimle göğüsleyen ve hayatımın her evresinde bana destek olan aileme sonsuz teşekkürlerimi sunarım.

Mehmet Tuna Selvi

Esra Gülmez

İÇİNDEKİLER

ÖZGÜNLÜK BİLDİRİMİ	I
BENZERLİK BİLDİRİMİ.....	II
TEŞEKKÜR	III
İÇİNDEKİLER	IV
ŞEKİLLER.....	VI
KISALTMALAR	VII
ÖZET	VIII
ABSTRACT.....	IX
1. GİRİŞ	1
2.DERİN ÖĞRENME İLE GÖZ HASTALIKLARI TESPİTİ	2
2.1 Projede Kullanılan Teknolojiler	2
2.1.1 Python Dili	2
2.1.2 Pandas.....	2
2.1.3 Numpy	2
2.1.4 Matplotlib	2
2.1.5 TensorFlow.....	2
2.1.6 Kaggle	3
2.2 Temel Konseptler	4
2.2.1 Makine Öğrenmesi	4
2.2.2 Yapay Sinir Ağları	4
2.2.3 Derin Öğrenme	4
2.2.4 Veri Setinin Tanıtılması	4
2.3 Enviroment Kurulumu	5
3. DOSYA İŞLEMLERİ	6
3.1 Kütüphanelerin İmport Edilmesi	6
3.2 Veri Setinin Anlaşılması.....	6
3.3 Data Frame Manipülasyonları.....	7
4. DERİN ÖĞRENME MODEL SEÇİMİ	11

4.1 VGG19 Modeli.....	11
4.2 Model Parametrelerinin Tune Edilmesi	16
4.3 Performans Metrikleri.....	18
5. SONUÇ.....	24
KAYNAKÇA.....	25
ÖZGEÇMİŞ 1	26
ÖZGEÇMİŞ 2	27

ŞEKİLLER

Şekil 2.1.5.1 TensorFlow Çalışma Mantığı	3
Şekil 2.3.1 Hastalık Etiketleri	5
Şekil 2.3.1 Enviroment Kurulumu	5
Şekil 3.1.1 Kütüphanelerin İmport Edilmesi.....	6
Şekil 3.2.1 Veri Setinin İlk 3 Satırının Gösterilmesi.....	7
Şekil 3.3.1 Katarakt'lı verilerin ayrılması.....	7
Şekil 3.3.2 Kataraktlı Göz Hücrelerinin Seçilmesi	8
Şekil 3.3.3 Normal Göz Hücrelerinin Seçilmesi	8
Şekil 3.3.4 Birleştirilmiş Dataframeler	8
Şekil 3.3.5 Veri Seti Oluşturma Metodu.....	9
Şekil 3.3.6 Katarakt ve Normal Hücrelerin Görsellerinin Ayrılması.....	9
Şekil 3.3.7 Oluşturulan Göz Hücrelerinin Gösterimi.....	10
Şekil 4.1.1 VGG19 Mimarisi[13]	12
Şekil 4.1.2 VGG19'un import edilmesi	12
Şekil 4.1.3 VGG19 Kullanarak Neural Network Yapısı Oluşturma	13
Şekil 4.1.4 Standart Lojistik Sigmoid Fonksiyonu[14].....	14
Şekil 4.1.5 Parametrelerin Gösterilmesi.....	15
Şekil 4.1.6 Compile Metrikleri	15
Şekil 4.2.1 Early Stopping ve Model Checkpoint'in Set Edilmesi	16
Şekil 4.2.2 Model Eğitimi	17
Şekil 4.2.3 Loss ve Accuracy.....	17
Şekil 4.2.4 Performans Metriklerinin İmport Edilmesi.....	18
Şekil 4.3.1 Accuracy Score	18
Şekil 4.3.2 Sınıflandırma Raporları	18
Şekil 4.3.3 Konfüzyon Matrisi Predicted-Actual Values[16]	19
Şekil 4.3.4 Konfüzyon Matrisi	20
Şekil 4.3.5 Model Accuracy ve Model Loss'un Hesaplanması	21
Şekil 4.3.6 Model Accuracy ve Model Loss'un Grafikleri	21
Şekil 4.3.7 Tahmin Edilen ve Gerçek Değerlerin Tespit Edilmesi	22
Şekil 4.3.8 Tahmin Edilen ve Gerçek Değerlerin Gösterimi	23

KISALTMALAR

VGG19: Visual Geometry Group 19

LLC: Limited Liability Company

NUMPY: Numerical Python

YSA: Yapay Sinir Ağları

WSL: Windows Subsystem for Linux

CUDA: Compute Unified Device Architecture

CNN: Artificial Neural Networks

CSV: Comma-Separated Values

RGB: Red-Green-Blue

ReLU: Rectified Linear Unit

ADAM: Adaptive Moment Estimation

ÖZET

Sağlık görüntülerinin segmente edilmesi, teşhis ve hastalık araştırmaları için her zaman çok büyük öneme sahip olmuştur. Manuel olarak yapılan teşhisler alanında uzman kişileri ve verilerin büyüklüğüne göre çokça zaman gerektirir. Günümüz teknolojilerinin vazgeçilmez bir parçası olan makine öğrenimi destekli yapay zeka uygulamaları, bizim için oldukça maliyetli olan görüntü segmentasyonunu efektif bir şekilde gerçekleştirmeyi amaçlıyor. Görüntü sınıflandırma işlemi için kullanılabilecek birçok konsept bulunmaktadır. Mevcut projemiz olarak Katarakt Teşhisi için pre-trained edilmiş eğitim ağırlıklarıyla çalışıldı. Bu ağırlıklardan popüleritesi, kompleks ve çok sayıda farklı türde görseller üzerine çalışan ImageNet kullanıldı. Eğitim için faydalandığımız bir diğer yapı ise VGG19 oldu. State-of-Art modellerinden biri olan,başarımı ve efektif çalışması çoğunluk tarafından onaylanmış VGG19 modeli, 19 katmanlı bir evrişimli sinir ağı modelidir. Görsel segmentasyon işlemlerinde yüksek başarımı ile bilinmektedir. Uygulamamızın temel amacı teşhis cümlelerinde Kataraktlı ve Normal olarak etiketlenmiş görselleri ikili şekilde sınıflandırabilecek yüksek başarılı bir eğitim modeli oluşturmaktır. Derin öğrenme modelleri görüntülerdeki karmaşık desenleri algılayabilme ve büyük veri kümelerinde bile hızlı çalışabilmesi sebebiyle hastalık teşhisleri ve tedavilerinde önemli bir araç haline gelmiştir. Tez bitişinde oluşturduğumuz modelin doğruluk, hassasiyet, öğrenme eğrisi gibi performans metriklerini analiz edilip, modelin güçlü ve zayıf yönleri incelendi. Modelin zayıf yönlerinde performans iyileştirmesi için yapılacaklar analiz edildi ve çeşitli önermeler de bulunuldu.

Anahtar Kelimeler:Derin Öğrenme, Yapay Sinir Ağları , CNN, Görsel Sınıflandırma

Haziran, 2023

Mehmet Tuna SELVİ
Esra GÜLMEZ

ABSTRACT

Segmenting health images, diagnosing diseases, and conducting disease research have always been of great importance. Manual diagnoses require expertise and can be time-consuming, depending on the size of the data. Machine learning-powered artificial intelligence applications, which have become an indispensable part of modern technologies, aim to perform image segmentation effectively, which is costly for us. There are various concepts that can be used for image classification. In our current project focusing on Cataract Diagnosis, we worked with pre-trained weights. The popular ImageNet, which works with complex and diverse types of images, was used for these weights. Another architecture we utilized for training is VGG19. VGG19 is one of the state-of-the-art models, widely recognized for its performance and effective functioning. It is a 19-layer convolutional neural network model known for its high performance in visual segmentation tasks. The main objective of our application is to create a high-performing training model that can classify images labeled as Cataract and Normal in binary form in diagnosis statements. Deep learning models have become important tools in disease diagnosis and treatments due to their ability to detect complex patterns in images and perform efficiently even with large datasets. At the end of the thesis, we analyzed the performance metrics of the model, such as accuracy, precision, and learning curves, and examined the strengths and weaknesses of the model. We also conducted an analysis of potential improvements for the model's weaknesses and provided various recommendations.

Keywords: Deep Learning, Artificial Neural Networks, CNN, Visual Classification

June,2023

Mehmet Tuna SELVİ

Esra Gülmez

1. GİRİŞ

Sağlık görüntülerinin segmente edilmesi ve hastalıkların doğru teşhisi, sağlık alanında her zaman büyük öneme sahip olmuştur. Ancak manuel teşhisler, uzmanlık gerektiren ve özellikle büyük veri kümeleriyle uğraşıldığında zaman alıcı bir süreç olabilmektedir. Son yıllarda, makine öğrenimi destekli yapay zeka uygulamaları, maliyetli olan görüntü segmentasyonunu etkin bir şekilde gerçekleştirmeyi amaçlayan modern teknolojilerin vazgeçilmez bir parçası haline gelmiştir. Görüntü sınıflandırması için çeşitli kavramlar ve teknikler bulunmaktadır. Mevcut projemiz olan Katarakt Teşhisi üzerinde çalışırken, eğitim sürecini hızlandırmak için önceden eğitilmiş ağırlıklardan faydalanmaktayız. Özellikle, karmaşık ve çeşitli görsel verileri içeren popüler ImageNet veri setini kullanmaktayız. Ayrıca, eğitim çerçevemizin bir parçası olarak VGG19 modelini kullanmaktayız. VGG19 modeli, etkileyici performansı ve etkin işlevselliğiyle tanınan state-of-the-art bir mimaridir. 19 katmanlı evrişimli sinir ağı yapısıyla VGG19 modeli, yüksek performanslı görsel segmentasyon görevlerinde olağanüstü yetenekler sergilemiştir. Uygulamamızın temel amacı, teşhis ifadelerine dayanarak görüntüleri Katarakt ve Normal olarak doğru bir şekilde sınıflandıran yüksek performanslı bir eğitim modeli geliştirmektir. Derin öğrenme modelleri, görüntülerdeki karmaşık desenleri ayırt edebilme ve büyük ölçekli veri kümeleriyle bile verimli performans sergileme yetenekleri nedeniyle hastalık teşhislerinde ve tedavilerinde önemli bir araç haline gelmiştir. Bu tezde, geliştirilen modelin doğruluk, hassasiyet ve öğrenme eğrileri gibi performans metriklerini ayrıntılı bir şekilde analiz ediyoruz. Ayrıca, modelin güçlü ve zayıf yönlerini kapsamlı bir şekilde araştırıyoruz. Modelin zayıf yönlerini tespit ederek, performansı artırmak için potansiyel stratejiler önermekte ve çeşitli öneriler sunmaktayız.

2.DERİN ÖĞRENME İLE GÖZ HASTALIKLARI TESPİTİ

2.1 Projede Kullanılan Teknolojiler

2.1.1 Python Dili

Web uygulamaları, yazılım geliştirme, veri bilimi ve makine öğreniminde (ML) yaygın olarak kullanılan bir programlama dilidir. Geliştiriciler, etkili ve öğrenmesi kolay olduğu ve birçok farklı platformda çalıştırılabildiği için Python'ı kullanır. Python yazılımı ücretsiz olarak indirilebilir, her türlü sistemle iyi bir entegrasyon sağlar ve geliştirme hızını artırır[1]. Bu sebeple bu proje de python dili kullanılmaktadır.

2.1.2 Pandas

Pandas, veri işlemesi ve analizi için Python programlama dilinde yazılmış olan bir yazılım kütüphanesidir [2]. Sahip olduğu birçok özellik sebebiyle veri ile uğraşan, makine öğrenmesi algoritmalarıyla çalışanlara büyük kolaylıklar sağlamaktadır [3]. Kısaca data frame'ler üzerinde çeşitli manipülasyonlar ve veri seti üzerinde atamalar yapmamızı kolaylaştıran bir python tool'udur.

2.1.3 Numpy

NumPy, çok boyutlu dizilerle ve matrislerle çalışmamızı sağlayan ve matematiksel işlemler yapabileceğimiz Python dili kütüphanelerindendir[4]. Proje de görsel verilerin matris dizilerine dönüştürülüp kullanılmasını sağlar.

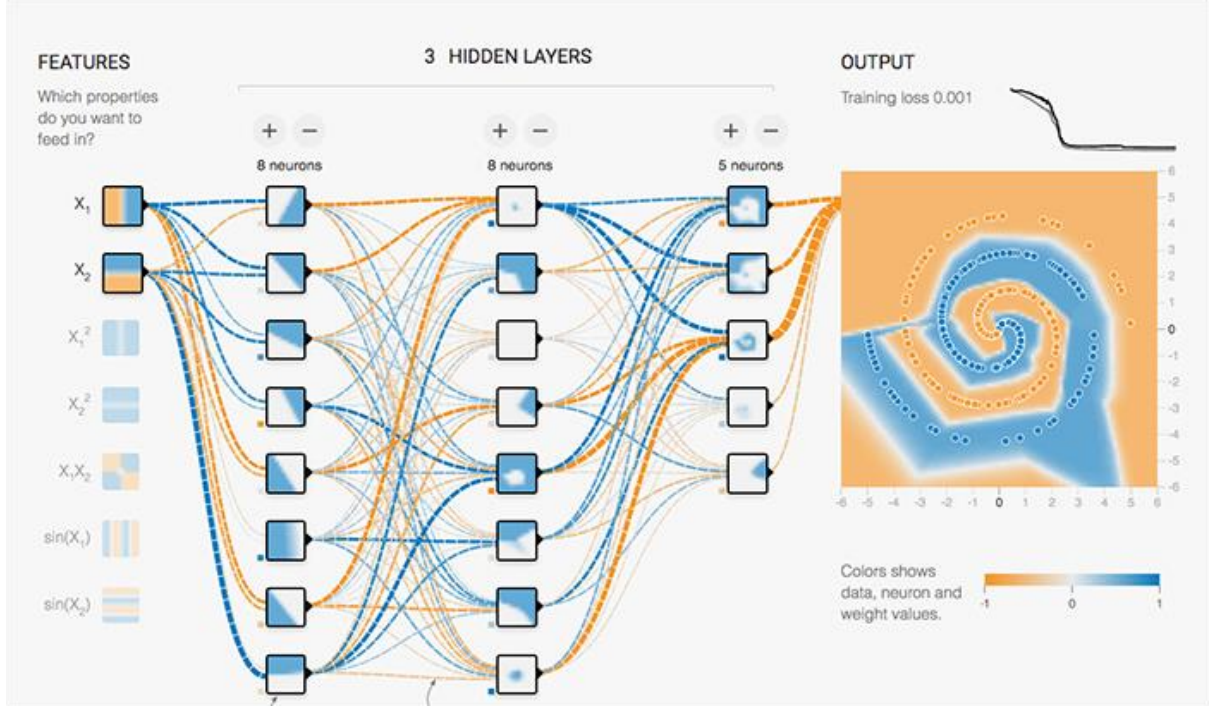
2.1.4 Matplotlib

Matplotlib; veri görselleştirmesinde kullandığımız temel python kütüphanesidir. 2 ve 3 boyutlu çizimler yapmamızı sağlar[5]. Learning curve, overfitting, underfitting gibi istatistiksel çıkarımların daha kolay anlaşılabilmesi için kullanılan görsel tool'dur.

2.1.5 TensorFlow

Tensorflow'un resmi dokümantasyonuna göre: TensorFlow, Google tarafından geliştirilen ve makine öğrenimi ve derin öğrenme modelleri oluşturmak için kullanılan açık kaynaklı bir yazılım kütüphanesidir. TensorFlow, veri akış grafikleri şeklinde çalışır ve bu grafiklerde matematiksel işlemler, veri akışı ve dönüşümler gibi hesaplamalar tanımlanır.

Bu sayede, büyük veri kümelerindeki karmaşık hesaplamaların paralel olarak gerçekleştirilmesini ve verimli bir şekilde dağıtılmasını sağlar[6].



Şekil 2.1.5.1 TensorFlow Çalışma Mantığı

2.1.6 Kaggle

Google LLC'nin bir yan kuruluşu olan Kaggle, veri bilimciler ve makine öğrenimi uygulayıcılarından oluşan çevrimiçi bir topluluktur. Kaggle, kullanıcıların veri kümelerini bulmasına ve yayınlamasına, web tabanlı bir veri bilimi ortamında modeller keşfetmesine ve oluşturmaya, diğer veri bilimcileri ve makine öğrenimi mühendisleriyle çalışmasına ve veri bilimi zorluklarını çözmek için yarışmalara katılmasına olanak tanır[7]. Bununla birlikte Kaggle, üzerinde çalışabileceğimiz birçok veri setini ve derin öğrenme için gerekli ortamı sağlar. Tarayıcı üzerinden local bilgisayarımızda ki kaynakların yanı sıra ücretsiz GPU kullanımını destekler.

2.2 Temel Konseptler

2.2.1 Makine Öğrenmesi

Makine öğrenmesi, matematik ve istatistik biliminden yararlanarak verilerin üzerinde yapılan işlemlerden çıkarımlarda bulunarak tahminler yapan sistemlerin bilgisayarlarla modellenmesidir. Model, mevcut veri seti ve kullanılan algoritmayla oluşturulur. Makine öğrenmesi, modellerden maksimum performansı elde etmek üzere kullanılır[8].

2.2.2 Yapay Sinir Ağları

Genel anlamda YSA, beynin bir işlevi yerine getirme yöntemini modellemek için tasarlanan bir sistemdir. YSA, yapay sinir hücrelerinin birbirleri ile çeşitli şekillerde bağlanmasından oluşur ve genellikle katmanlar halinde düzenlenir. Donanım olarak elektronik devrelerle veya bilgisayarlarda yazılım olarak gerçekleştirilebilir [9].

2.2.3 Derin Öğrenme

Derin öğrenme, eldeki verinin birden çok soyutlama seviyesinde temsil edilebilmesi için birden çok işlem katmanı bulunan hesaplama modellerini kapsamaktadır. Derin ağlar olarak da bilinen derin öğrenme yöntemlerinde, verinin temsili için üst üste olacak şekilde farklı katmanlar mevcuttur. Derin öğrenme yöntemleri ham verilerden etkin bir üst seviye soyutlaması yapmakla, otomatik öznitelik kümeleri oluşturabilmekte, böylelikle normalde çoğunlukla insanlar tarafından belirlenen özniteliklerin otomatik olarak çıkarılıp kullanılması sağlanmaktadır[10]. State-of-art modelleri olarak bilinen derin öğrenme modelleri başarıyı herkes tarafından onaylanmış modellerdir. Ağırlıkları spesifik veri setlerinin eğitiminde kullanılmış ve projemizde bu modellerden VGG19 modeli efektif bir şekilde kullanılmaya çalışılmıştır.

2.2.4 Veri Setinin Tanıtılması

Kaggle üzerinden aldığımız; yaş, sol ve sağ gözlerden renkli fundus fotoğrafları ve doktorların doktorlardan teşhis anahtar kelimeleri ile 5.000 hastanın yapılandırılmış bir oftalmik veri tabanıdır.

Bu veri seti, Shangong Medical Technology Co., Ltd. tarafından Çin'deki farklı hastanelerden/tıbbi merkezlerden toplanan "gerçek hayattaki" hasta bilgileri kümesini temsil eder. Bu kurumlarda, Canon, Zeiss ve Kowa gibi piyasadaki çeşitli kameralar tarafından fundus görüntüleri çekilerek çeşitli görüntü çözünürlükleri elde edilmektedir.

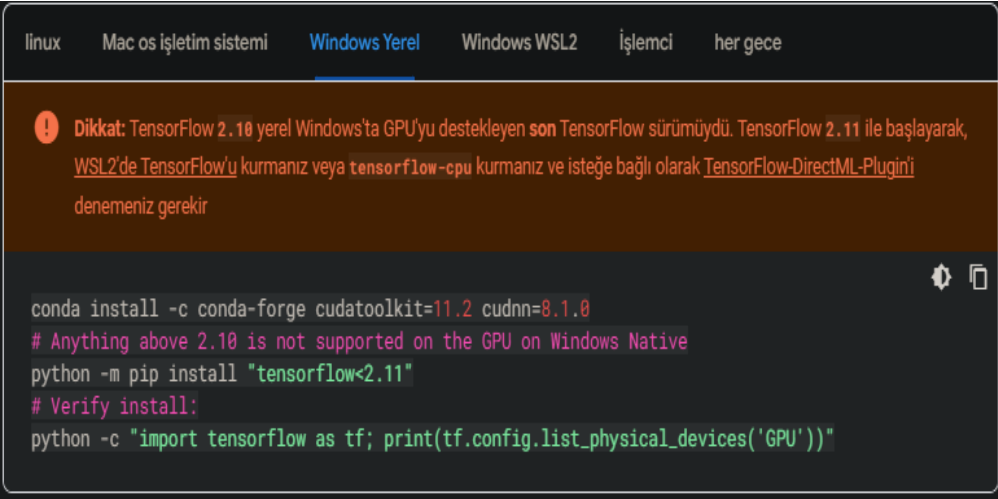
Ek açıklamalar, kalite kontrol yönetimi ile eğitilmiş insan okuyucular tarafından etiketlendi. Hastayı aşağıdakileri içeren sekiz etikete ayırırlar[11]:

- Normal (N),
- Diabetes (D),
- Glaucoma (G),
- Cataract (C),
- Age related Macular Degeneration (A),
- Hypertension (H),
- Pathological Myopia (M),
- Other diseases/abnormalities (O)

Şekil 2.3.1 Hastalık Etiketleri

2.3 Enviroment Kurulumu

Makine öğrenmesi modellerinin çalışabilmesi için gerekli alt yapıları oluşturduk. WSL üzerinden model eğitmek için Tensorflow ve keras, frame manipülasyonları için pandas ve numpy, görselleştirme için matplotlib, seaborn gibi frameworkleri kernel içerisine entegre edip testlerini gerçekleştirdik. Enviroment'ı kullandığım işletim sisteminden ziyade WSL'e kurmamdaki temel sebep tensorflow'un web sitesindeki aşağıdaki uyarı oldu:



```
linux  Mac os işletim sistemi  Windows Yerel  Windows WSL2  İşlemci  her gece

! Dikkat: TensorFlow 2.10 yerel Windows'ta GPU'yu destekleyen son TensorFlow sürümüydü. TensorFlow 2.11 ile başlayarak,
WSL2'de TensorFlow'u kurmanız veya tensorflow-cpu kurmanız ve isteğe bağlı olarak TensorFlow-DirectML-Plugin'i
denemeniz gerekir

conda install -c conda-forge cudatoolkit=11.2 cudnn=8.1.0
# Anything above 2.10 is not supported on the GPU on Windows Native
python -m pip install "tensorflow<2.11"
# Verify install:
python -c "import tensorflow as tf; print(tf.config.list_physical_devices('GPU'))"
```

Şekil 2.3.1 Enviroment Kurulumu

WSL'i paralel hesaplamalı veri işlemlerinde Gpu kullanımını aktif hale getirebilmek için CUDA kurulumlarını ve testlerini gerçekleştirdik.

3. DOSYA İŞLEMLERİ

3.1 Kütüphanelerin İmport Edilmesi

Projede gerekli olan lineer cebiri, dataframe manipölasyonları, görsel okuma, grafik oluşturma gibi gerekli olan kütüphanelerin import edilmesi işlemi okla gösterilen 1 numaralı cell de tanımlanmıştır. Çalışma ortamı olarak Kaggle kullandığımız için 2 numaralı cell de üzerinde çalışacağımız dosyanın directory'sinde gezinme işlemi verilmiştir.

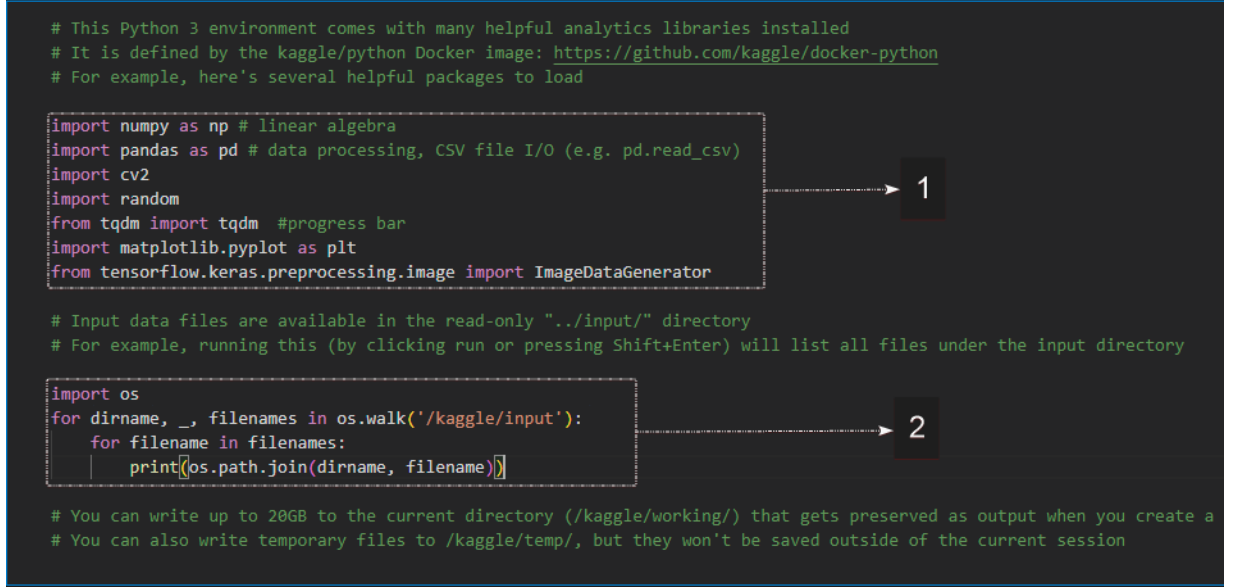
```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import cv2
import random
from tqdm import tqdm #progress bar
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```



Şekil 3.1.1 Kütüphanelerin İmport Edilmesi

3.2 Veri Setinin Anlaşılması

Çalıştığımız veri setini pandas kullanarak csv dosyasından belirtilen dosya yolu parametre olarak verilerek okundu ve ilk 3 satırı ekrana şekildeki gibi yazdırıldı.

```
df = pd.read_csv("../kaggle/input/ocular-disease-recognition-odir5k/full_df.csv")
df.head(3)
```

✓
0.0s
Python

ID	Patient Age	Patient Sex	Left-Fundus	Right-Fundus	Left-Diagnostic Keywords	Right-Diagnostic Keywords	N	D	G	C	A	H	M	O	filepath	labels	target	filename
0	0	69	Female	0_left.jpg	0_right.jpg	cataract	normal fundus	0	0	0	1	0	0	0	../input/ocular-disease-recognition-odir5k/ODI...	['N']	[1, 0, 0, 0, 0, 0]	0_right.jpg
1	1	57	Male	1_left.jpg	1_right.jpg	normal fundus	normal fundus	1	0	0	0	0	0	0	../input/ocular-disease-recognition-odir5k/ODI...	['N']	[1, 0, 0, 0, 0, 0]	1_right.jpg
2	2	42	Male	2_left.jpg	2_right.jpg	laser spot, moderate non proliferative retinopathy	moderate non proliferative retinopathy	0	1	0	0	0	0	1	../input/ocular-disease-recognition-odir5k/ODI...	['D']	[0, 1, 0, 0, 0, 0]	2_right.jpg

Şekil 3.2.1 Veri Setinin İlk 3 Satırının Gösterilmesi

3.3 Data Frame Manipülasyonları

full_df.csv içerisindeki doktor teşhislerine göre "cataract" etiketi ile etiketlenmiş verileri ayırmak için metodumuzu oluşturduk.

```
def kataraktli(text):
    if "cataract" in text:
        return 1
    else:
        return 0
```

Şekil 3.3.1 Katarakt'lı verilerin ayrılması

CSV dosyasımızda sağ ve sol veri grupları için ayrı ayrı teşhisler verildiği için aynı metodu her iki grup içinde çağırıyoruz. Çağırdığımız veri setlerini sağ ve sol katarakt şeklinde ayrı bir dataframe de tuttuk. Katarakt verileri veri setinde 'C' olarak ve kataraktlı hastaları 1, katarakt hastası olmayanlar 0 şeklinde tutulmuş. full_df.csv içerisindeki doktor teşhislerine göre "cataract" etiketi ile etiketlenmiş verileri hem sağ hemde sol göz verileri için ayırıp çalıştırıyoruz.

```

df["sol_katarakt"] = df["Left-Diagnostic Keywords"].apply(lambda x: kataraktli(x))
df["sag_katarakt"] = df["Right-Diagnostic Keywords"].apply(lambda x: kataraktli(x))

sol_katarakt = df.loc[(df.C == 1) & (df.sol_katarakt == 1)][["Left-Fundus"].values
sol_katarakt[:10]

array(['0_left.jpg', '81_left.jpg', '103_left.jpg', '119_left.jpg',
      '254_left.jpg', '294_left.jpg', '330_left.jpg', '448_left.jpg',
      '465_left.jpg', '477_left.jpg'], dtype=object)

sag_katarakt = df.loc[(df.C == 1) & (df.sag_katarakt == 1)][["Right-Fundus"].values
sag_katarakt[:10]

array(['24_right.jpg', '81_right.jpg', '112_right.jpg', '188_right.jpg',
      '218_right.jpg', '345_right.jpg', '354_right.jpg', '477_right.jpg',
      '553_right.jpg', '560_right.jpg'], dtype=object)

```

Şekil 3.3.2 Kataraktlı Göz Hücrelerinin Seçilmesi

Multiclass regresyon yapmak yerine sadece normal ve cataraktlı göz hücreleri alınacağı için normal göz hücrelerinin de veri setinden çekilmesi. Veri setinde normal göz hücrelerinin verisi kataraktlı verilerden çok daha fazla olduğu için verinin dengesiz olmaması için normal göz hücresi için sağ ve sol göz için 250 sample alıyoruz.

```

sol_normal = df.loc[(df.C == 0) & (df["Left-Diagnostic Keywords"] == "normal fundus")][["Left-Fundus"].sample(250, random_state = 42).values
sag_normal = df.loc[(df.C == 0) & (df["Right-Diagnostic Keywords"] == "normal fundus")][["Right-Fundus"].sample(250, random_state = 42).values
sag_normal[:15]

array(['2964_right.jpg', '680_right.jpg', '500_right.jpg',
      '2368_right.jpg', '2820_right.jpg', '2769_right.jpg',
      '2696_right.jpg', '2890_right.jpg', '940_right.jpg',
      '2553_right.jpg', '3371_right.jpg', '3042_right.jpg',
      '919_right.jpg', '3427_right.jpg', '379_right.jpg'], dtype=object)

```

Şekil 3.3.3 Normal Göz Hücrelerinin Seçilmesi

Sağ ve sol göz verisi olarak ayrılmış verileri sadece normal ve kataraktlı gözler için dataframe içerisinde birleştiriyoruz.

```

cataract = np.concatenate((sol_katarakt, sag_katarakt), axis = 0)
normal = np.concatenate((sol_normal, sag_normal), axis = 0)
print(len(cataract), len(normal))

594 500

```

Şekil 3.3.4 Birleştirilmiş Dataframeler

Görsellerin directory'si üzerinden görsele erişip, kerasın preprocessing kütüphanesinden faydalanarak görsellerin numpy dizisi şeklinde formatlandırdık. Dataset değişkeninde istediğimiz formatta verilerimiz ve o veriye ait etiketler bulunmakta. Etiketler ile path üzerindeki kataraktlı verileri joinledik.

```
from tensorflow.keras.preprocessing.image import load_img, img_to_array
dataset_dir = "/kaggle/input/ocular-disease-recognition-odir5k/preprocessed_images/"
gorsel_boyut = 224
labels = []
dataset = []
def veriseti_olustur(image_category, label):
    for img in tqdm(image_category):
        #gorsel ve veriyolunu joinleme islemi
        image_path = os.path.join(dataset_dir, img)
        try:
            image = cv2.imread(image_path, cv2.IMREAD_COLOR)
            image = cv2.resize(image, (gorsel_boyut, gorsel_boyut))
        except:
            continue

        dataset.append([np.array(image), np.array(label)])

    random.shuffle(dataset)
    return dataset
```

Şekil 3.3.5 Veri Seti Oluşturma Metodu

Aynı işlemleri hem kataraktlı göz hücresi görselleri ve normal göz hücresi görselleri üzerinde **veriseti_olustur** metodunun çağırarak uyguladık.

```
dataset = veriseti_olustur(cataract, 1)

100%|██████████| 594/594 [00:06<00:00, 94.31it/s]

len(dataset)

588

dataset = veriseti_olustur(normal, 0)

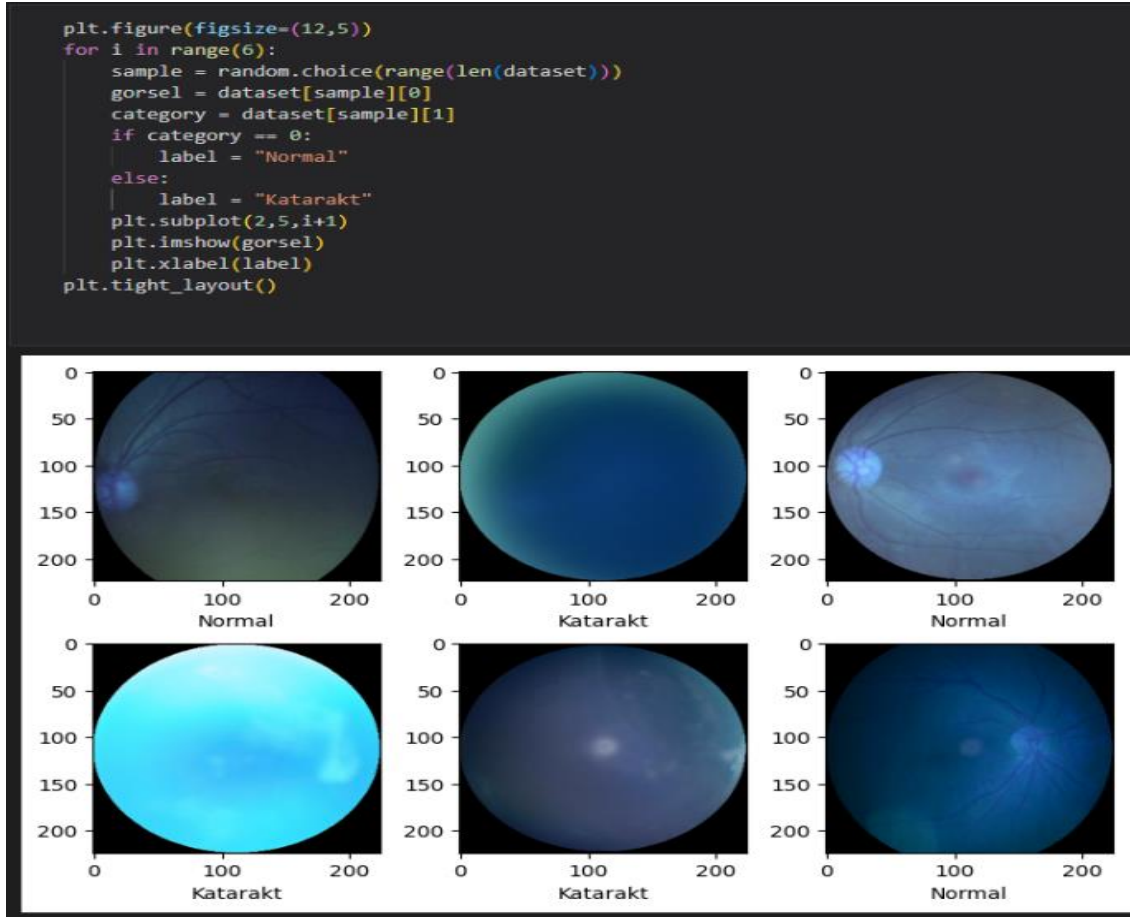
100%|██████████| 500/500 [00:07<00:00, 69.13it/s]

len(dataset)

1088
```

Şekil 3.3.6 Katarakt ve Normal Hücrelerin Görsellerinin Ayrılması

Rastgele olarak formatlanmış görsellerin gösterimi:



Şekil 3.3.7 Oluşturulan Göz Hücrelerinin Gösterimi

4. DERİN ÖĞRENME MODEL SEÇİMİ

Problemimiz görsel tanıma ve sınıflandırma problemi olduğu için ya kendimiz bir neural network oluşturmaliyiz ya da "state-of-the-art modelleri olarak bilinen derin öğrenme modellerini tercih etmeliyiz. Görsel tanıma ve sınıflandırma için bir çok model bulunmakta. Öncelikli amaç herhangi bir modelin veri setinde uygulanıp, ezberlemenin önüne geçilmesi ve seçilen model üzerindeki ince ayar(fine-tune) işlemlerinin gerçekleştirilmesidir. Veri setimizi eğittikten sonra performans ölçütleri(hata oranı, overfeeding, underfitting vb) gibi niceliklerin tez sonunda sunulması planlanmaktadır.

4.1 VGG19 Modeli

VGG19, Oxford Üniversitesi'nde geliştirilen derin evrişimli sinir ağlarının (Convolutional Neural Networks - CNN) bir modelidir. VGG ailesinin bir üyesi olarak, 2014 yılında VGG16 modeline birkaç ek katman eklenerek oluşturulmuştur. VGG19, görüntü sınıflandırma görevlerinde büyük başarı elde etmiş ve derin öğrenme alanında önemli bir kilometre taşı olmuştur.

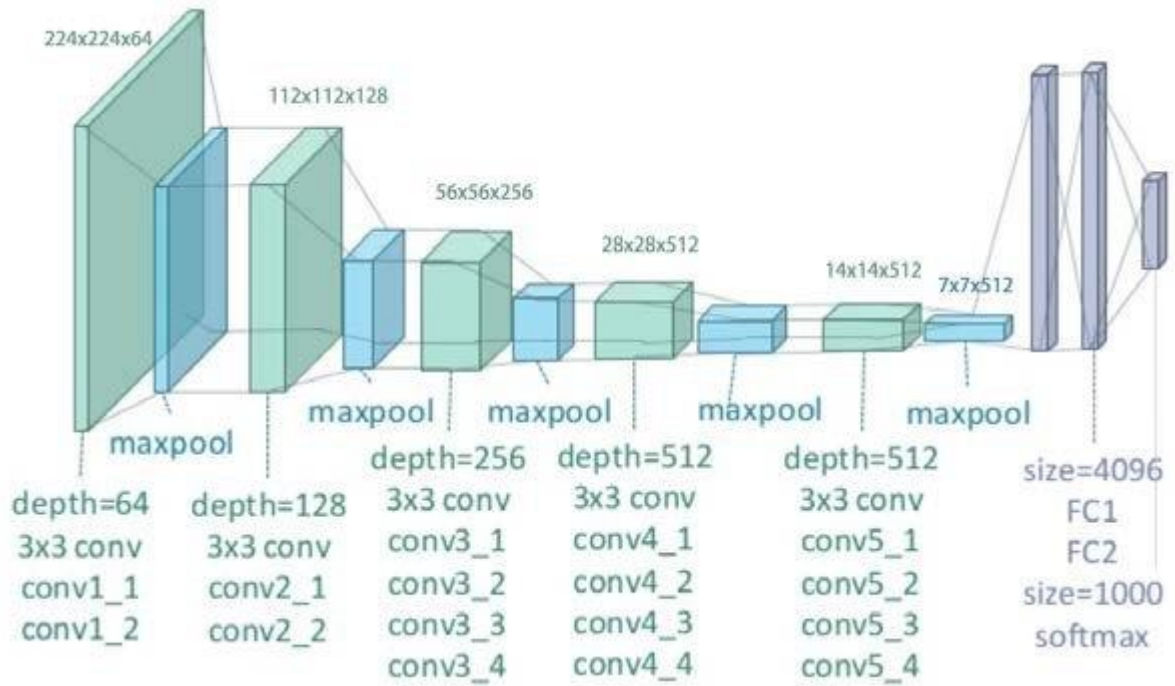
VGG19 mimarisi, tamamen evrişimli katmanlardan oluşur. Toplamda 19 katmana sahip olan bu model, çeşitli evrişim ve havuzlama katmanlarından oluşan bloklar halinde düzenlenmiştir. Her bir blokta 3x3 boyutunda evrişim filtreleri kullanılmıştır.

Aşağıda, VGG19 mimarisinin ayrıntıları verilmiştir:

- Giriş Katmanı: 224x224x3 boyutunda bir giriş görüntüsünü alır.
- Evrişim Blokları: 2 adet 2D evrişim katmanı, ardışık olarak 64 adet filtre ile uygulanır. Ardından her iki evrişim katmanının ardından 2x2 boyutunda max pooling uygulanır.
- Evrişim Blok: 2 adet 2D evrişim katmanı, 128 adet filtre ile uygulanır. Ardından max pooling işlemi gerçekleştirilir.
- Evrişim Blok: 4 adet 2D evrişim katmanı, 256 adet filtre ile uygulanır. Ardından max pooling işlemi gerçekleştirilir.
- Evrişim Blok: 4 adet 2D evrişim katmanı, 512 adet filtre ile uygulanır. Ardından max pooling işlemi gerçekleştirilir.
- Evrişim Blok: 4 adet 2D evrişim katmanı, 512 adet filtre ile uygulanır. Ardından max pooling işlemi gerçekleştirilir.
- Tam Bağlantılı Katmanlar: 3 adet tam bağlantılı (fully connected) katman bulunur. Her biri 4096 nöron içerir. Aktivasyon fonksiyonu olarak ReLU kullanılır. İlk iki tam bağlantılı katmanın ardından dropout işlemi uygulanır.

- Çıkış Katmanı: 1000 sınıf için softmax aktivasyon fonksiyonu ile oluşturulmuş bir çıkış katmanı bulunur[12].

Sonuç: VGG19, 19 katmanlı derin evrişimli sinir ağı yapısıyla görüntü sınıflandırma görevlerinde yüksek performans gösteren bir modeldir. Evrişim ve pooling katmanlarının tekrar eden yapısı, daha karmaşık özelliklerin çıkarılmasına olanak sağlar. Bu özellikleri sayesinde VGG19, mevcut projemiz için kullanılabilir. Bu modelin mimarisi **Şekil 4.1.1** 'de verilmiştir.



Şekil 4.1.1 VGG19 Mimarisi[13]

Önceden VGG19 ile eğitilmiş veri ağırlıkları imagenet olan veri setinin bizim gorsellerimizin boyutlarıyla aynı olacak şekilde import edilmesi.

```
from tensorflow.keras.applications.vgg19 import VGG19
vgg = VGG19(weights="imagenet", include_top = False, input_shape=(gorsel_boyut, gorsel_boyut, 3))

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg19/vgg19_weights_tf_dim_ordering_tf_kernels_notop.h5
80134624/80134624 [=====] - 1s 0us/step
```

Şekil 4.1.2 VGG19'un import edilmesi

VGG'nin derin öğrenme modelindeki katmanları başlangıçta eğilemez olarak set ediyoruz. Bunun temel sebebi VGG19 modelinin önceden eğitilmiş ağırlıkları ImageNet gibi

büyük bir veri seti üzerinden eğitildiği için (pre-trained data) önceden öğrenilmiş özelliklerin korunması ve üzerine eklenen yeni sınıflandırma katmanlarının daha spesifik veri setine uygun hale getirilmesi amaçlanır. Yani özetle amacımız; modelin önceden eğitilmiş ağırlıkları korunması ve yalnızca eklenen yeni sınıflandırma katmanlarının genel özellikleri kullanarak eğitilmesidir. Bu, genellikle fine-tuning işleminin ilk adımıdır ve daha sonra eklenen katmanlarla birlikte model eğitilir.

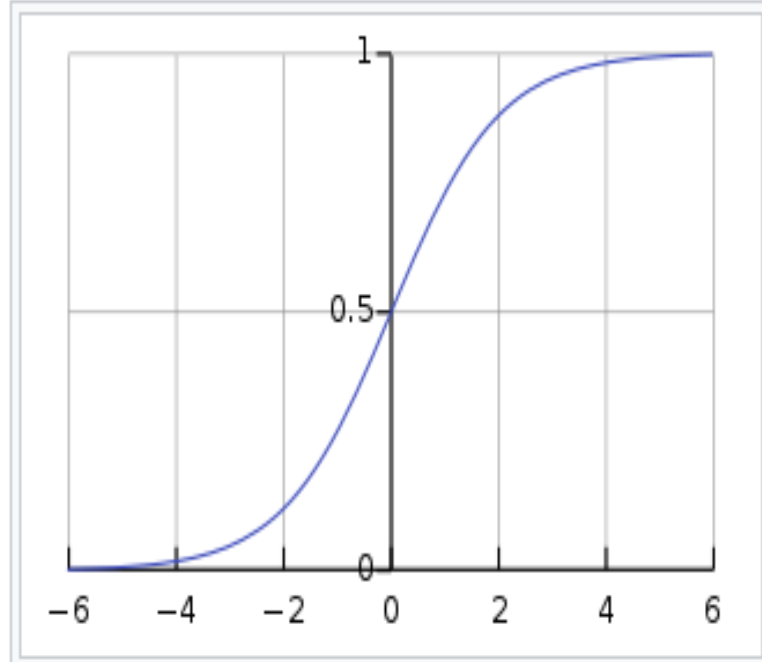
VGG modelinin katmanları, flatten katmanı,ve dense katmanlarını import ettik ve bir neural network yapısı oluşturduk.Flatten katmanı çok boyutlu giriş verilerini tek boyutlu vektörlere dönüştürerek sinir ağının ardışık katmanlarında işlenebilmesini sağlayan bir yapıdır. Bizim çalışmamızda 3 boyutlu(yükseklik,genişlik,RGB) tensor olarak algılanan renkli görsellerin işlenmesi gerekmektedir. Dense(yoğunluk) katmanı: katmanlar arasında tam bağlantılı bir yapı oluşturup sinir ağında girdi verilerini çıktı verilerine dönüştürür.

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

from tensorflow.keras import Sequential
from tensorflow.keras.layers import Flatten,Dense
model = Sequential()
model.add(vgg)
model.add(Flatten())
model.add(Dense(1,activation="sigmoid")) #çikti birim,aktivasyon(0-1)secim ikili
```

Şekil 4.1.3 VGG19 Kullanarak Neural Network Yapısı Oluşturma

Aktivasyon fonksiyonu olarak sigmoid seçmemizdeki sebep sınıflandırmamızın ikili sınıflandırma olmasıdır. Sigmoid fonksiyonu çıktıyı 0-1 aralında bir olasılık değerine dönüştürür, 0.5'i eşik değeri olarak seçer ve karşılaştırma yapar. Özetle ikili sınıflandırma için ideal bir aktivasyon fonksiyonudur. Çalışmanın devamında hiper parametrelerin en iyilenmesi kısmında değiştirilebilir.



Şekil 4.1.4 Standart Lojistik Sigmoid Fonksiyonu[14]

Şekil 4.1.5’de `summary()` metoduyla katmanlardaki parametreleri gösterdik.

```
model.summary()
```

Model: "sequential"		
Layer (type)	Output Shape	Param #
vgg19 (Functional)	(None, 7, 7, 512)	20024384
flatten (Flatten)	(None, 25088)	0
dense (Dense)	(None, 1)	25089
Total params: 20,049,473		
Trainable params: 25,089		
Non-trainable params: 20,024,384		

Şekil 4.1.5 Parametrelerin Gösterilmesi

Oluşturduğumuz sinir ağı yapısını compile ettik. İlk parametre olan **ADAM** optimizezeri loss değerimizi minimize etmeyi sağlar. Çalışma prensibi olarak adaptif öğrenme hızı, momentum (dalgalanma) düzenleme ve hafıza işlemlerini efektif olarak yapar. Çalıştığımız veri seti gibi verilerin çok boyutlu ve çok sayıda veri barındıran veri setlerinde ideal çalışma imkanı sunar.

```
model.compile(optimizer="adam",loss="binary_crossentropy",metrics=["accuracy"])
```

Şekil 4.1.6 Compile Metrikleri

Loss parametresi metodun gerçek değeri ile tahmin edilen değer arasındaki farkın hangi metotla yapılacağını belirler. Bizim örneğimizde **binary_crossentropy** kullanıldı. Bunun en temel sebebi multiclass regresyon/multiclass sınıflandırma yerine Katarakt ya da Normal sınıflandırma yapmamızdır. Kayıp fonksiyonun metriği ise **accuracy** olarak verdik. Gerçek tahmin ile tahmin edilen arasındaki farklı bulmak istiyoruz. Alternatif olarak f1-score, precision ve recall gibi parametrelerde verilebilirdi.

4.2 Model Parametrelerinin Tune Edilmesi

Elimizdeki derin öğrenme modelinin parametrelerinin en iyilenmesi için öncelikle olarak **Model Checkpoint** ve **Early Stopping** metotlarının açıklanmasıyla ve modele uygun set edilmesiyle başlayacağız.

Model Checkpoint: Tensorflow'un resmi dökümantasyonuna göre modelcheckpoint metodolojisi, eğitim süreci sırasında ve sonrasında modelin ilerlemesini ve son durumu kaydetmek için kullanılan bir mekanizmadır, temelde 3 konsepti vardır. Bunlar eğitim sürecinin devam ettirilmesi, model performansının farklı eğitim süreçlerinde değerlendirilmesi ve model paylaşımı. Kendi örneğimizde model performansını değerlendirilirken en iyi sonucu veren kontrol noktasını; ağırlıkları VGG19 ağırlıklarına sahip ve daha isabetli sonuçların öncelikli olacağı şekilde set edeceğiz.

Prechelt, L. 'in kitabındaki tanımında ise Early stopping, aşırı öğrenme (overfitting) problemini çözmek için kullanılan bir tekniktir. Eğitim süreci sırasında modelin performansı sürekli olarak izlenir ve belirli bir sayıda ardışık epoch boyunca doğrulama veri kümesinde performansta bir iyileşme olmadığı durumlarda eğitim durdurulur. Bu, modelin genelleştirme yeteneğini artırır ve gereksiz hesaplama kaynaklarını kullanmamızı engeller[15].

Bizim modelimizde modelin isabeti üzerinden bir Early Stopping işlemi gerçekleştireceğiz.

```
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping
checkpoint = ModelCheckpoint("vgg19.h5", monitor="val_acc", verbose=1, save_best_only=True,
                             save_weights_only=False, period=1)
earlystop = EarlyStopping(monitor="val_acc", patience=5, verbose=1)
```

Şekil 4.2.1 Early Stopping ve Model Checkpoint'in Set Edilmesi

Verilerimizi monitör ederken accuracy değeri yerine validation accuracy (çoğu kaynakta overall accuracy olarak geçer) değerine göre yapmamızın sebebi modelin train veri setinde ezberleme yapıp yapmadığını kontrol etmektir. Yüksek accuracy değerine sahip olsak bile modelimiz veriyi ezberliyorsa (test de efektif çalışmıyorsa) model yüksek başarıma sahip olduğu söylenemez. Set edilen ölçeklerde model eğitimi yapılır. Her bir epochun isabet oranı ve kayıplarını ekrana yazdırırsak:

```
history = model.fit(x_train,y_train,batch_size=32,epochs=25,validation_data=(x_test,y_test),
                    verbose=1,callbacks=[checkpoint,earlystop])
```

Output exceeds the [size limit](#). Open the full output data [in a text editor](#)

Epoch 1/25
28/28 [=====] - 29s 423ms/step - loss: 1.5185 - accuracy: 0.8908 - val_loss: 0.9576 - val_accuracy: 0.9312
Epoch 2/25
28/28 [=====] - 5s 185ms/step - loss: 0.2971 - accuracy: 0.9678 - val_loss: 0.4157 - val_accuracy: 0.9450
Epoch 3/25
28/28 [=====] - 5s 186ms/step - loss: 0.0676 - accuracy: 0.9897 - val_loss: 0.4594 - val_accuracy: 0.9633
Epoch 4/25
28/28 [=====] - 5s 175ms/step - loss: 0.0116 - accuracy: 0.9943 - val_loss: 0.4429 - val_accuracy: 0.9495
Epoch 5/25
28/28 [=====] - 5s 187ms/step - loss: 0.0069 - accuracy: 0.9966 - val_loss: 0.3738 - val_accuracy: 0.9495
Epoch 6/25
28/28 [=====] - 5s 187ms/step - loss: 0.0238 - accuracy: 0.9908 - val_loss: 0.5775 - val_accuracy: 0.9587
Epoch 7/25
28/28 [=====] - 5s 187ms/step - loss: 0.1308 - accuracy: 0.9736 - val_loss: 0.5521 - val_accuracy: 0.9679
Epoch 8/25
28/28 [=====] - 5s 189ms/step - loss: 0.0864 - accuracy: 0.9897 - val_loss: 0.4423 - val_accuracy: 0.9495
Epoch 9/25
28/28 [=====] - 5s 182ms/step - loss: 0.0090 - accuracy: 0.9966 - val_loss: 0.4228 - val_accuracy: 0.9633
Epoch 10/25
28/28 [=====] - 5s 194ms/step - loss: 0.0083 - accuracy: 0.9966 - val_loss: 0.5563 - val_accuracy: 0.9587
Epoch 11/25
28/28 [=====] - 5s 196ms/step - loss: 0.0066 - accuracy: 0.9977 - val_loss: 0.6954 - val_accuracy: 0.9587
Epoch 12/25
28/28 [=====] - 5s 188ms/step - loss: 3.6448e-04 - accuracy: 1.0000 - val_loss: 0.6500 - val_accuracy: 0.9633
Epoch 13/25
...
Epoch 24/25
28/28 [=====] - 6s 201ms/step - loss: 5.0590e-06 - accuracy: 1.0000 - val_loss: 0.6010 - val_accuracy: 0.9633
Epoch 25/25
28/28 [=====] - 6s 206ms/step - loss: 4.8876e-06 - accuracy: 1.0000 - val_loss: 0.6002 - val_accuracy: 0.9633

Şekil 4.2.2 Model Eğitimi

Şekil 4.2.2’deki çıktıya göre başlangıçta **1.5185** olan hata oranı **4.8876e-06** 'ya kadar düştü. patience=5 ifadesiyle 5 epoch boyunca accuracy değerinde ilerleme kaydedilmezse eğitim durduruldu.

```
loss,accuracy = model.evaluate(x_test,y_test)
print("loss:",loss)
print("Accuracy:",accuracy)
```

7/7 [=====] - 1s 151ms/step - loss: 0.6002 - accuracy: 0.9633
loss: 0.6001932621002197
Accuracy: 0.963302731513977

Şekil 4.2.3 Loss ve Accuracy

Loss ve accuracy değerlerini evaluate metodunu kullanarak tahmin işlemi yaptık. Evaluate metodu performans değerlendirme işleminde loss ve accuracy değişkenlerini hesaplamak için kullanılır. Predict ise performans ölçmek yerine modelin son durumdaki girişlerine karşılık gelen çıktıyı verir. Şu an modelimiz yüksek 4.8876e-06 ve düşük loss değerine sahip ve bu bizim için ideal fakat oluşturduğumuz modelin performansını ölçmek için konfüzyon matrisi ve accuracy score gibi metriklerle de kontrol etmemiz gerekir.

```
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
# y_pred = model.predict_classes(x_test)
y_pred = (model.predict(x_test) > 0.5).astype("int32")

7/7 [=====] - 1s 149ms/step
```

Şekil 4.2.4 Performans Metriklerinin İmport Edilmesi

Şekil 4.2.4’de belirtildiği üzere ikili sınıflandırma yapılacağı için normal ve kataraktlı göz hücreleri 0.5 eşik değeri ile predict edilir.

4.3 Performans Metrikleri

```
accuracy_score(y_test,y_pred)

0.963302752293578
```

Şekil 4.3.1 Accuracy Score

Modelimiz 0.96 isabet oranıyla verileri doğru tahmin ediyor. Test veri setinde tahmin edilen değerlerin sınıflandırma raporları aşağıdaki gibidir.

```
print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
0	0.98	0.94	0.96	103
1	0.95	0.98	0.97	115
accuracy			0.96	218
macro avg	0.96	0.96	0.96	218
weighted avg	0.96	0.96	0.96	218


Şekil 4.3.2 Sınıflandırma Raporları

Modelimizdeki performans metriklerini incelediğimizde ortalama 0.97 precision (kesinlik) tahminleri yapıyor. Yani pozitif olarak tahmin edilen örneklerin toplam örneklere oranını veriyor. **Precision = True Positive / (True Positive + False Positive)**. Recall(duyarlılık) değeri gerçek pozitiflerin ne kadarının doğru tahmin edildiğini veriyor. F1- score precision ve recall değerlerinin doğruluğunu, support sınıfı ise belirli sınıfa ait gerçek etiketlere sahip örneklerin sayısını gösteriyor. Bizim raporumuzda bu oranların hepsinin yüksek olduğunu görebiliyoruz

.İstatistiksel sınıflandırmada, verileri belirli bir sınıf kümesine tahmin etmek veya sınıflandırmak için algoritmalar veya modeller oluştururuz. Modeller mükemmel olmadığından dolayı, bazı veri noktaları yanlış şekilde sınıflandırılacaktır. Konfüzyon matrisi, modelin ne kadar iyi performans gösterdiğini gösteren bir tablo özeti olarak kullanılır.

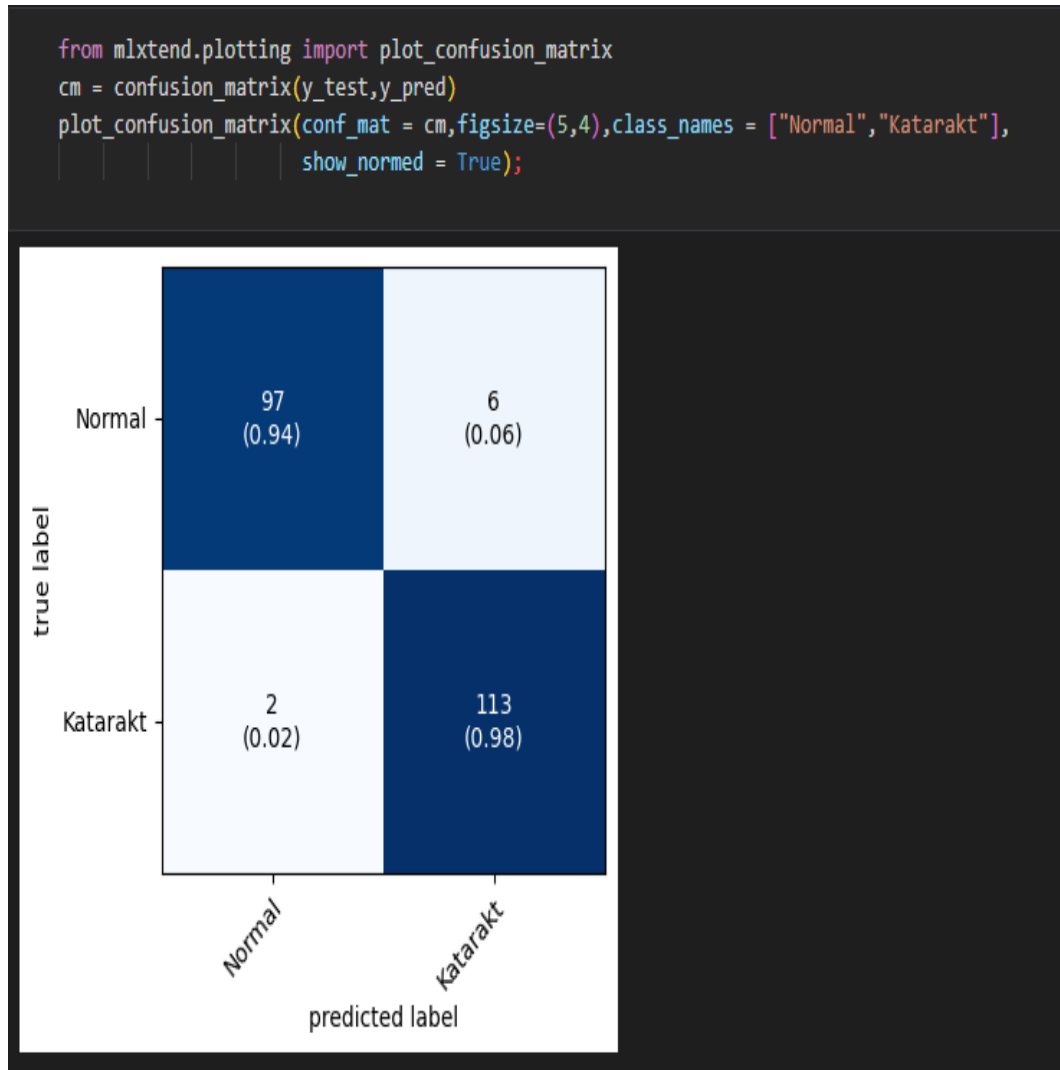
Test veri setinde konfüzyon matrisimizi çizdirerek hangi örneklerde hatalı hangi örneklerde başarılı tahminler yaptığımızı görebiliriz.

		Actual Value (as confirmed by experiment)	
		positives	negatives
Predicted Value (predicted by the test)	positives	TP True Positive	FP False Positive
	negatives	FN False Negative	TN True Negative

Confusion Matrix. Source: Idris 2018. 

Şekil 4.3.3 Konfüzyon Matrisi Predicted-Actual Values[16]

Şekil 4.3.4'delki Matrisi yorumlayacak olursak Normal olarak sınıflandırılan 105 örneğin gerçek değerinin de normal olduğunu(True Positive) 6 tane örneğin ise gerçek değerinin kataraktlı olmasına rağmen normal sınıfına (False Negatif) tahmin edilmiş. 115 Kataraktlı örneğin 113 tanesi kataraktlı (True Negatif), 2 tanesi de normal olarak tahmin edilmiş (False Negatif). İleride model üzerinde optimazsyon yapmak istersek normal sınıfı üzerinden yanlış tahmin edilen örnekleri gözlemleyip veriler üzerinde manipölasyon gerçekleştirerek isabet oranımızı daha da arttırabiliriz.



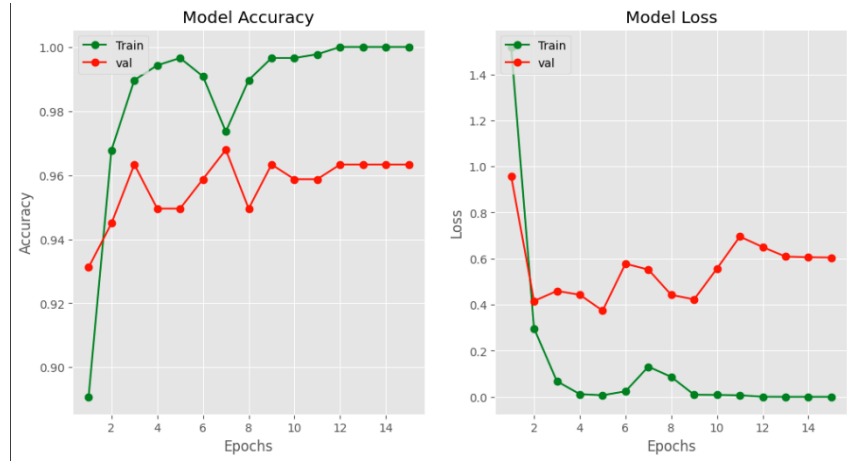
Şekil 4.3.4 Konfüzyon Matrisi

Tahmin sonuçlarımızın doğruluğu yüksek olmasına rağmen modelin Loss ve Accuracy değerlerinin hangi epochlarda azılıp arttığını gözlemlemek istersek:

```
plt.style.use("ggplot")
fig = plt.figure(figsize=(12,6))
epochs = range(1,16)
#Model Accuracy
plt.subplot(1,2,1)
plt.plot(epochs[:15], history.history["accuracy"][:15], "go-")
plt.plot(epochs[:15], history.history["val_accuracy"][:15], "ro-")
plt.title("Model Accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend(["Train","val"],loc = "upper left")
#Model Loss
plt.subplot(1,2,2)
plt.plot(epochs[:15], history.history["loss"][:15], "go-")
plt.plot(epochs[:15], history.history["val_loss"][:15], "ro-")
plt.title("Model Loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend(["Train","val"],loc = "upper left")
plt.show()
```

Şekil 4.3.5 Model Accuracy ve Model Loss'un Hesaplanması

Şekil 4.3.5'teki kodun çıktısı Şekil 4.3.6'daki gibidir. Çıktıdaki grafiğin yorumu sonuç kısmında yapılacaktır.



Şekil 4.3.6 Model Accuracy ve Model Loss'un Grafikleri

Son olarak **Şekil 4.3.4**'teki değindiğimiz konfüzyon matrisindeki True Pozitif, True Negatif, False Pozitif ve False Negatif olarak labellanan görselleri rastgele olacak şekilde göstermek istersek:

```
plt.figure(figsize=(15,8))
for i in range(6):
    sample = random.choice(range(len(x_test)))
    ornek = x_test[sample]
    kategori = y_test[sample]
    tahmin_kategori = y_pred[sample]

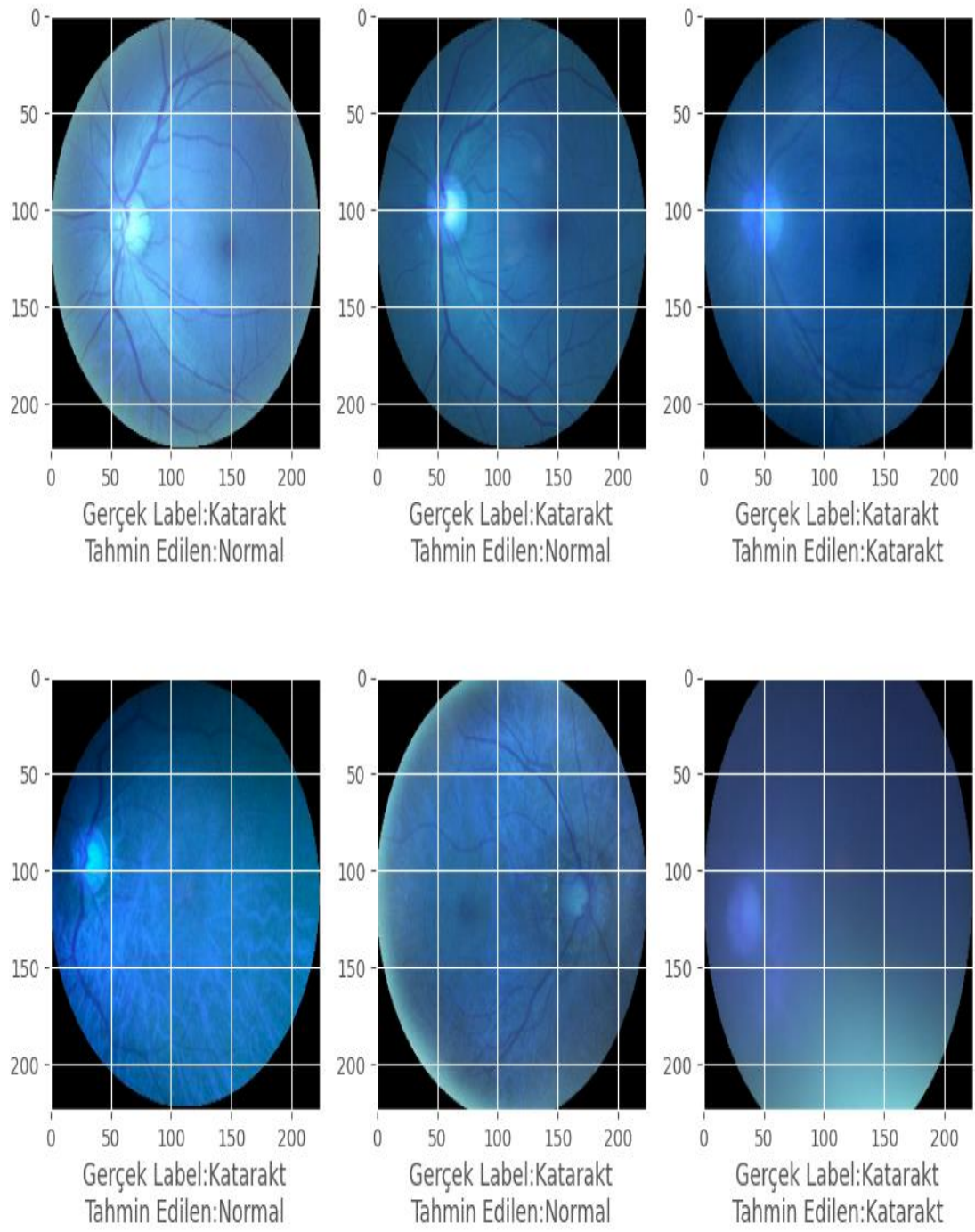
    if kategori== 0:
        label = "Normal"
    else:
        label = "Katarakt"

    if tahmin_kategori== 0:
        pred_label = "Normal"
    else:
        pred_label = "Katarakt"

    plt.subplot(2,5,i+1)
    plt.imshow(ornek)
    plt.xlabel("Gerçek Label:{}\nTahmin Edilen:{}".format(label,pred_label))
plt.tight_layout()
```

Şekil 4.3.7 Tahmin Edilen ve Gerçek Değerlerin Tespit Edilmesi

Şekil 4.3.7'deki kodun çıktısı **Şekil 4.3.8**'de verilmiştir. Örneğimizde rastgele olacak şekilde seçilen görsellerin, eğitim sonundaki tahmin edilen değerler ile gerçek değerlerin etiketli olduğu görseller birlikte gösterilmiştir.



Şekil 4.3.8 Tahmin Edilen ve Gerçek Değerlerin Gösterimi

5. SONUÇ

Şekil 4.3.6'daki grafiği yorumlayacak olursak eğitim ve doğrulama verilerinin hangi epochlarda düşüp yükseldiği gösterilmektedir. Val validasyon değeridir. Train accuracy değeri validasyon accuracy değerinden daha yüksek olduğu için yaklaşık %3'lük bir overfitting problemi olduğu söylenebilir. Aynı şekilde train loss değeri validasyon loss değerinden daha yüksek olduğu için az da olsa overfitting olduğuna varılabilir. Earlystopping, modelcheckpoint gibi overfitting önleyici metotlar dışında düzenleme tekniklerinin sayısını arttırmak(Dropout,L1/L2), katmanlardaki hiperparametreleri değiştirmek(Tune işlemlerinin gözden geçirilmesi),giriş verilerinin standartize edilmesi(fazla manipülasyonda ezberleme artadabilir olabildiğince raw data üzerinden çalışılmalı) gibi yöntemler takip edilebilir. Bunun yanı sıra outlier değerler drop edilebilir. Fakat ezberleme oranı çok düşük olduğu için modelimizi balance durumda sayabiliriz.

KAYNAKÇA

- [1] URL:<https://aws.amazon.com/tr/whatis/python/#:~:text=Python%3B%20web%20uygulamalar%C4%B1%2C%20yaz%C4%B1%C4%B1m%20geli%C5%9Firme,%C3%A7al%C4%B1%C5%9Ft%C4%B1r%C4%B1labildi%C4%9Fi%20i%C3%A7in%20Python'%C4%B1%20kullan%C4%B1r.>
- [2] URL:<https://tr.wikipedia.org/wiki/Pandas>.
- [3] URL:<https://medium.com/datarunner/python-k%C3%BCt%C3%BCphaneleri-5cbf95d5a347>
- [4] URL:<https://medium.com/datarunner/numpy-k%C3%BCt%C3%BCphanesi-f78d6cc098fa>
- [5] URL:<https://medium.com/datarunner/matplotlibkutuphanesi-1-99087692102b>
- [6] URL:<https://www.tensorflow.org/?hl=tr>
- [7] URL:<https://en.wikipedia.org/wiki/Kaggle>
- [8] Arslankaya, Seher, and Şevval Toprak. "Makine öğrenmesi ve derin öğrenme algoritmalarını kullanarak hisse senedi fiyat tahmini." *International Journal of Engineering Research and Development* 13.1 (2021): 178-192.
- [9] Ataseven, Burçin. "Yapay sinir ağları ile öngörü modellemesi." *Öneri Dergisi* 10.39 (2013): 101-115.
- [10] KÜÇÜK, Doğan, and Nursal ARICI. "Doğal Dil İşlemede Derin Öğrenme Uygulamaları Üzerine Bir Literatür Çalışması." *Uluslararası Yönetim Bilişim Sistemleri ve Bilgisayar Bilimleri Dergisi* 2.2 (2018): 76-86.
- [11] URL:<https://www.kaggle.com/datasets/andrewmvd/ocular-disease-recognition-odir5k>
- [12] Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv preprint arXiv:1409.1556.
- [13] URL:https://www.researchgate.net/figure/Illustration-of-the-network-architecture-of-VGG-19-model-conv-means-convolution-FC-means_fig2_325137356
- [14] URL:https://tr.wikipedia.org/wiki/Lojistik_fonksiyon
- [15] Prechelt, L. (1998). Early stopping - but when? In *Neural Networks: Tricks of the Trade* (pp. 55-69). Springer.
- [16] URL:<https://devopedia.org/confusion-matrix>

ÖZGEÇMİŞ 1

MEHMET TUNA SELVİ

YAZILIM GELİŞTİRİCİ

+90 0534 770 5374

mehmettunaselvi@gmail.com

Merkez / KARAMAN

<https://github.com/mehmettunaselvi>

Kariyer Hedefi

Eğitim hayatım boyunca öğrendiğim teorik bilgileri pratiğe dönüştürebileceğim ve iş ortamını gözlemleyerek kendimi hayatımın her segmentinde inove edebileceğim bir ortamda kendimi geliştirmek en büyük hedefimdir.

İş/Staj Deneyimi

Yaz Stajı-Fırat Üniversitesi, Bilgisayar Mühendisliği

08/2020-09/2020

Eğitim aldığım bölümün Donanım A.B.D. Prof. Dr. Yetkin TATAR'ın koordinatörlüğünde Siber Güvenlik ve Etik Hacker'lık kapsamında Pentesting, Sistem Güvenlik Taraması, Sistem Tasarımı ve Kurulumu alanlarında bir ay süren stajımı tamamladım.

Kullanılan Teknolojiler

GNS3, VMware, Kali Linux, Bash Script, Nmap, Wireshark ve Nessus

Yaz Stajı-İMPARK Bilişim Eğitim LTD. ŞTİ.

06/2022-07/2022

18 Temmuz - 12 Ağustos arasında 1 aylık stajımı OSTİM ODÜ teknokent de İM Park Bilişimde tamamladım. ASP.MVC mimarisinde geliştirilen dinamik yapı bir web sitesinin abstractının oluşturulması.

Kullanılan Teknolojiler

C#, ASP.NET MVC, Razor Page, ADO.NET, jQuery, Bootstrap, HTML, CSS, Entity Framework

Eğitim Hayatı

Lise Abdullah Tayyar Anadolu Lisesi

26/09/2012 - /17/06/2016

Üniversite Fırat Üniversitesi, Bilgisayar Mühendisliği

09/2017 - Halen devam etmekte

Kurslar ve Sertifikalar

Educalton and Culture Lifelong Learning Program COMENIUS Philosophfriends

10/03/2014-20/03/2015

In my high school, I and other participants are worked about Philosophers and cultural values of Norway, Belgium and Turkey. Every participant member in these student exchange program introduced their own country's Philosophers.

WT 1976 Web Geliştirici Kursu

24/04/2021-23/05/2021

DSC Gazi ve Bilge Adam ortaklığıyla Youtube üzerinden online gerçekleştirilen, toplamda 32 saat süren, başlangıç seviyesinde JQuery, Bootstrap ve C# öğretilen Web geliştirici kursunda katılımcı.

TechCareer Web Devlopment BootCamp

23/01/2023-16/02/2023

Kariyer.net'in alt oluşumu olan Tech Career bünyesinde Kayseri Teknopark'ta düzenlenen, Akın Karabulut eğitmenliğinde, .NET MVC, Entity Framework Core teknolojilerinin anlatıldığı ve uygulamalar geliştirdiğimiz, toplamda 48 saatlik eğitimde katılımcı.

Dil Bilgisi

İngilizce 

Almanca 

Referanslar

Kemal KOLCUOĞLU
NETAŞ/Devops Mühendisi
İletişim Bilgileri
Tel No: 05316128687
E-Mail: kemalko@netrd.com.tr

Projeler

ASP.MVC mimarisinde geliştirilen dinamik yapı bir içerik yönetimin sisteminin abstractının oluşturulması.

Kullanılan Teknolojiler

C#, ASP.NET MVC, Razor Page, ADO.NET, jQuery, Bootstrap, HTML, CSS, Entity Framework

Derin Öğrenme modelleri kullanılarak; ensembling ve fine tuning metodolojileriyle göğüs kanseri teşhisinin veri analizi ve sınıflandırılması.

Kullanılan Teknolojiler

Python, numpy, pandas, matplotlib, Sci-kit learn

ÖZGEÇMİŞ 2



Eğitimler

Yüreğir Halıcılar Anadolu Lisesi Lise, Adana	Eyl 2013 - Tem 2017
Fırat Üniversitesi Üniversite, Elazığ	Eyl 2017 - devam ediyor

Stajlar

Fırat Teknokent İf Yazılım
Fırat Üniversitesi Bilgi İşlem Dairesi

Özel bölüm

Java	Html, Css
Python	Javascript

Kişisel bilgiler

Doğum tarihi	25 Şubat 1999
Doğum yeri	Adana/Yüreğir
Ehliyet	B
Cinsiyet	Kadın
Uyruk	Türkiye Cumhuriyeti

Diller

İngilizce
Almanca