# Proje Başlığı

:information\_source: \*\*Dersin Kodu:\*\* [YAZ20411](https://ebp.klu.edu.tr/Ders/dersDetay/YAZ20411/716026/tr)

:information\_source: \*\*Dersin Adı:\*\* [DERİN ÖĞRENME](https://ebp.klu.edu.tr/Ders/dersDetay/YAZ20411/716026/tr)

:information\_source: \*\*Dersin Öğretim Elemanı:\*\* Öğr. Gör. Dr. Fatih BAL [Github](https://github.com/balfatih) | [Web Sayfası](https://balfatih.github.io/)

---

## Grup Bilgileri

| Öğrenci No | Adı Soyadı | Bölüm | Proje Grup No | Grup Üyelerinin Github Profilleri |

|------------|----------------------|--------------------------|---------------|---------------------------------------------------|

| 1200505046 | Muhammed Enes TURA | Yazılım Mühendisliği | PROJE\_22 | [Github](https://github.com/MuhammedEnesTURA) |

| 1200505003 | Ceren ÖZTÜRK | Yazılım Mühendisliği | PROJE\_22 | [Github](https://github.com/krutzonerec) |

| 1200505028 | Talha ÖZİNAL | Yazılım Mühendisliği | PROJE\_22 | [Github](https://github.com/TalhaOzinal) |

| 1200505031 | Sümeyye Ahsen KAYA | Yazılım Mühendisliği | PROJE\_22 | [Github](https://github.com/sumeyyeahsenkaya) |

---

## Proje Açıklaması

Projenin amacı derin öğrenme yardımıyla hem tümörlü hem de tümörsüz röntgenleri vererek, verilen bir röntgenin tümörlü veya tümörsüz olup olmadığını tespit etmek için bir sinir ağı modeli oluşturmak ve eğitmektir. Modeller eğitilerek data üzerinden test edilir. Derin öğrenme modeli oluşturmak ve eğitmek amacıyla Phyton programlama dili ve TensorFlow kütüphanesi kullanılmıştır. Google Collab ortamında çalışır. Veriler eğitim ve test olmak üzere %65-%35 ve %80-%20 olarak iki şekilde ayrılır. Data ilk olarak drivedan okunur. Tümörlü ve tümörsüz veriler ayrılıp değişkende tutulur. Daha sonra train ve test datası ayrılır. 2 seçenek için %65 train - %35 test ve %80 train - %20 test olarak ayrılır.

TensorFlow ve keras kütüphaneleri, derin öğrenme modelinin oluşturulması ve eğitilmesi için temel araçları sağlar.

ImageDataGenerator görüntü verilerini artırmak ve modele beslemek için kullanılır.

train\_test\_split veri setini eğitim, doğrulama ve test setlerine bölmek için kullanılır.

os işletim sistemi fonksiyonlarına erişim sağlar.

shutil dosya kopyalama gibi işlemleri gerçekleştirmek için kullanılır.

numpy sayısal hesaplamalar için kullanılır.

Google Drive Bağlantısı ve Veri Yolu Belirlenmesi:

drive.mount('/gdrive') Google Drive'ı Colab ortamına bağlar.

base\_path, tumor\_path, ve normal\_path gibi değişkenler, veri setinin bulunduğu dosya yollarını tanımlar.

Veri Setinin Yüklenmesi ve Etiketlenmesi:

os.listdir fonksiyonu ile tümör ve normal hücre dosyalarının yolları alınır.

Data listesi tümör ve normal hücre dosyalarını içerir, labels listesi her dosyanın sınıf etiketini içerir.

Eğitim, Doğrulama ve Test Setlerinin Oluşturulması:

train\_test\_split fonksiyonu ile veri seti eğitim, doğrulama ve test setlerine ayrılır.

Veri Artırma ve Modelin Oluşturulması:

ImageDataGenerator ile veri artırma işlemi yapılır. Bu, eğitim sırasında modele daha fazla çeşitlilik katmak için kullanılır.

Sequential modeli oluşturulur ve katmanlar eklenir: Convolutional, MaxPooling, Flatten, Dense vb.

Modelin Eğitilmesi:

model.compile ile modelin kayıp fonksiyonu, optimize edici ve metrikleri belirlenir.

model.fit ile model eğitilir. Eğitim verileri, doğrulama verileri, epoch sayısı ve callback fonksiyonları gibi parametreler belirlenir.

Transfer Learning (VGG16):

VGG16 modeli önceden eğitilmiş ağırlıklar ile yüklenir.

Transfer öğrenme için, VGG16 modelinin üzerine özel katmanlar eklenir ve model oluşturulur.

Kendi Sinir Ağı Modelinin Oluşturulması:

Giriş şekli, giriş katmanı ve çıkış katmanı belirlenir.

Model sınıfı kullanılarak kendi sinir ağı modeli oluşturulur.

Modelin Eğitilmesi (Kendi Sinir Ağı):

model.compile ile modelin kayıp fonksiyonu, optimize edici ve metrikleri belirlenir.

model.fit ile kendi sinir ağı modeli eğitilir. Eğitim verileri, doğrulama verileri, epoch sayısı ve callback fonksiyonları gibi parametreler belirlenir.

Bu adımlar, bir derin öğrenme modeli oluşturmak için tipik olarak izlenen temel adımları içerir. Ancak, kodun çalışabilmesi için bazı eksik veya hatalı kısımları olabilir. Bu kodun doğru bir şekilde çalışması için giriş verilerini kontrol etmek, modelin tanımlanmasını ve eğitilmesini takip etmek önemlidir.

---

## Kod Açıklamaları

import tensorflow as tf

from tensorflow import keras

from keras.layers import Dense

from keras.models import Sequential, load\_model

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from sklearn.model\_selection import train\_test\_split

import os

from google.colab import drive

import shutil

import numpy as np

Kullanılacak kütüphanelerin import edilmesini sağlar.

drive.mount('/gdrive')

base\_path = '/gdrive/My Drive/Kidney\_Cancer/Kidney Cancer'

tumor\_path = os.path.join(base\_path, 'Tumor')

normal\_path = os.path.join(base\_path, 'Normal')

Google Drive bağlantısı kurulur ve veri setinin bulunduğu yol belirlenir.

tumor\_files = [os.path.join(tumor\_path, f) for f in os.listdir(tumor\_path) if os.path.isfile(os.path.join(tumor\_path, f))]

normal\_files = [os.path.join(normal\_path, f) for f in os.listdir(normal\_path) if os.path.isfile(os.path.join(normal\_path, f))]

data = tumor\_files + normal\_files

labels = ['tumor'] \* len(tumor\_files) + ['normal'] \* len(normal\_files)

Tümör ve normal hücrelerin dosya yolları alınır, veri seti oluşturulur ve etiketlenir.

X\_train, X\_temp, y\_train, y\_temp = train\_test\_split(data, labels, test\_size=0.35, random\_state=42)

X\_val, X\_test, y\_val, y\_test = train\_test\_split(X\_temp, y\_temp, test\_size=(10/35), random\_state=42)

Veri seti eğitim, doğrulama ve test setlerine ayrılır.

train\_datagen = ImageDataGenerator(rescale=1./255)

val\_test\_datagen = ImageDataGenerator(rescale=1./255)

model = tf.keras.models.Sequential([

])

Modelin katmanları burada tanımlanır.

Veri arttırma için ImageDataGenerator kullanılır ve bir sinir ağı modeli oluşturulur.

model.compile(optimizer=optimizer, loss='binary\_crossentropy', metrics=['accuracy'])

history = model.fit(

)

Model eğitimi için gerekli olan parametreler burada belirtilir.

Model, eğitim verileri kullanılarak belirli sayıda epoch (iterasyon) boyunca eğitilir.

base\_model = VGG16(weights='imagenet', include\_top=False, input\_shape=(32, 32, 3))

VGG16 modeli üzerine özel katmanlar eklenerek yeni bir model oluşturulur ve eğitilir.

Bu kısımda VGG16 transfer öğrenme algoritması kullanılarak yeni bir model oluşturulur ve eğitilir.

input\_shape = (32, 32, 1)

x\_input = Input(shape=input\_shape)

x = Dense(128, activation='relu')(x\_input)

x = Dense(64, activation='relu')(x)

output = Dense(num\_classes, activation='softmax')(x)

model = Model(inputs=x\_input, outputs=output)

Bu kısımda kendi özel sinir ağı modeli oluşturulur.

model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])

history = model.fit(

)

Model eğitimi için gerekli olan parametreler burada belirtilir.

Oluşturulan kendi sinir ağı modeli, belirli sayıda epoch boyunca eğitilir.

Bu adımlar, kodun bir sinir ağı modeli oluşturmak ve eğitmek için nasıl kullanıldığını göstermektedir.

train\_dataset = tf.data.Dataset.from\_tensor\_slices((X\_train, y\_train))

val\_dataset = tf.data.Dataset.from\_tensor\_slices((X\_val, y\_val))

test\_dataset = tf.data.Dataset.from\_tensor\_slices((X\_test, y\_test))

Elimizdeki datayı test, train ve validation olarak üçe böler.

model = tf.keras.models.Sequential([

tf.keras.layers.Conv2D(16, (3, 3), activation='relu', input\_shape=(32, 32, 3)),

tf.keras.layers.MaxPooling2D(2, 2),

tf.keras.layers.Conv2D(32, (3, 3), activation='relu'),

tf.keras.layers.MaxPooling2D(2, 2),

tf.keras.layers.Flatten(),

tf.keras.layers.Dense(64, activation='relu'),

tf.keras.layers.Dense(1, activation='sigmoid')

])

Bu TensorFlow modeli, evrişimli sinir ağı (CNN) kullanarak bir görüntü sınıflandırma modelini tanımlar. Bu modelin katmanlarının ve işlevlerinin açıklamaları:

tf.keras.layers.Conv2D(16, (3, 3), activation='relu', input\_shape=(32, 32, 3)):

Bu katman, 16 adet 3x3 boyutunda evrişim filtresini kullanarak görüntü verilerinden özellikler çıkaran bir evrişim katmanıdır.

activation='relu' ile, ReLU (Rectified Linear Activation) aktivasyon işlevi kullanılır, bu da negatif değerleri sıfır yapar ve pozitif değerleri aynı şekilde bırakır.

input\_shape=(32, 32, 3) ile, giriş verilerinin boyutu belirtilir. Bu model, 32x32 piksel boyutunda ve 3 kanallı (RGB) görüntülerle çalışır.

tf.keras.layers.MaxPooling2D(2, 2):

Bu katman, evrişim sonuçlarını küçültmek ve hesaplama karmaşıklığını azaltmak için 2x2 boyutunda bir maksimum havuzlama işlemi uygular. Özellik haritalarını daha küçük boyutlara indirir.

tf.keras.layers.Conv2D(32, (3, 3), activation='relu'):

İkinci bir evrişim katmanıdır ve ilk evrişim katmanına benzer şekilde çalışır, ancak bu sefer 32 adet farklı özellik haritası üretir.

tf.keras.layers.MaxPooling2D(2, 2):

İkinci bir maksimum havuzlama katmanıdır ve özellik haritalarını daha da küçültür.

tf.keras.layers.Flatten():

Bu katman, özellik haritalarını düz bir vektöre dönüştürür. Bu, evrişim katmanlarından gelen 2D verileri düz bir liste haline getirir.

tf.keras.layers.Dense(64, activation='relu'):

Bu tam bağlantılı (fully connected) katman, 64 nöron içerir ve önceki katmandan gelen özellikleri işleyerek daha karmaşık özellikler üretir.

Yine ReLU aktivasyon işlevi kullanılır.

tf.keras.layers.Dense(1, activation='sigmoid'):

Bu son katman, modelin sınıflandırma sonucunu verir.

Sadece 1 nöron içerir çünkü bu örnekte iki sınıfın sınıflandırılması yapılıyor ve bu nöron bu iki sınıfı temsil eder.

Sigmoid aktivasyon işlevi kullanılır çünkü bu, binary sınıflandırma problemleri için kullanılan yaygın bir aktivasyon işlevidir ve çıkışı [0, 1] aralığına sıkıştırır, olasılık değeri olarak yorumlanabilir.

Bu model, özellik convolutional katmanları ile başlar, ardından özellik haritalarını düzleştirir, ardından tam bağlantılı katmanlar ile daha yüksek seviyeli özellikler üretir ve sonunda binary sınıflandırma sonucunu verir.

---

## Proje Dosya Yapısı

- \*\*/veriseti\*\*

- `Kidney\_Cancer.zip`

- `Kidney Cancer`

- `Kidney Cancer`

- `Tumor`

- `Normal`

- \*\*/proje\*\*

- `README.md`

- `derin\_ogrenme\_65\_train\_35\_test (5).py`

---

## Kurulum

Veri Seti

- `https://drive.google.com/file/d/1-ukgRGfCDrCYXVU4HiU9gQdXdBVko9S4/view`

- `https://colab.research.google.com/drive/12017WlbxTcyW42NsIvMzO3tEBf4T8rFS?usp=sharing`

- `https://colab.research.google.com/drive/1UzDGmYbC4ssYS5Wi5BpDBXzm7Ah-wE\_R?usp=sharing`

---

## Kullanım

1 - Google Collab'a bağlanılır.

2 - Drive bağlantısı ve yolu belirlenir.

3 - Sırasıyla kodlar çalıştırılır.

---

## Katkılar

TensorFlow kütüphanesinin proje için gerekli özellikleri araştırılmıştır.

---

## İletişim

| Sümeyye Ahsen KAYA | [Mail](turaenes19@gmail.com) |

| Ceren ÖZTÜRK | [Mail](cerenozturk.ceren@gmail.com) |

| Talha ÖZİNAL | [Mail](mor\_mackle\_44@outlook.com) |

| Sümeyye Ahsen KAYA | [Mail](ahsenkaya61@hotmail.com) |asını ve eğitilmesini takip etmek önemlidir.