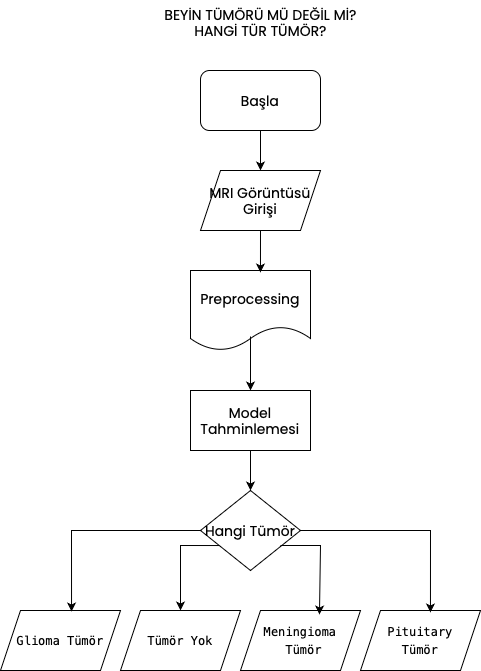
***Algoritma Aşamaları***



#Gerekli Kütüphanelerin İçe Aktarılması

import os

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

import mplcyberpunk

plt.style.use("cyberpunk")

import cv2

import tensorflow as tf

from sklearn.utils import shuffle

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

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import GlobalMaxPooling2D,GlobalAveragePooling2D, Dropout, Dense, BatchNormalization

from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau, TensorBoard, ModelCheckpoint

from sklearn.metrics import classification\_report,confusion\_matrix

from tqdm import tqdm

# Eğitim Verilerinin Yüklenmesi ve Etiketlenmesi

X\_train = []

y\_train = []

labels = ['glioma\_tumor','no\_tumor','meningioma\_tumor','pituitary\_tumor']

# Test Verilerinin Yüklenmesi ve Etiketlenmesi

image\_size = 150

for i in labels:

folderPath = os.path.join('/content/drive/MyDrive/brain/Training',i)

for j in tqdm(os.listdir(folderPath)):

img = cv2.imread(os.path.join(folderPath,j))

img = cv2.resize(img,(image\_size, image\_size))

X\_train.append(img)

y\_train.append(i)

for i in labels:

folderPath = os.path.join('/content/drive/MyDrive/brain/Testing',i)

for j in tqdm(os.listdir(folderPath)):

img = cv2.imread(os.path.join(folderPath,j))

img = cv2.resize(img,(image\_size,image\_size))

X\_train.append(img)

y\_train.append(i)

X\_train = np.array(X\_train)

y\_train = np.array(y\_train)

# Etiket Dağılımının Görselleştirilmesi

label\_counts = {label: np.sum(y\_train == label) for label in labels}

plt.figure(figsize=(8, 6))

colors = ["C0", "C1", "C2", "C3"]

plt.subplot(2, 1, 1)

bars = plt.bar(label\_counts.keys(), label\_counts.values(), color=colors)

mplcyberpunk.add\_bar\_gradient(bars=bars)

plt.ylabel('Count')

plt.title('Distribution of Labels')

k = 0

for i in labels:

j = 0

while True:

if y\_train[j] == i:

plt.subplot(2, 4, k + 5)

plt.imshow(X\_train[j])

plt.axis('off')

k += 1

break

j += 1

plt.tight\_layout()

plt.show()

# Veri Karıştırma ve Eğitim-Test Ayrımı

X\_train, y\_train = shuffle(X\_train,y\_train, random\_state=101)

X\_train,X\_test,y\_train,y\_test = train\_test\_split(X\_train,y\_train, test\_size=0.1,random\_state=101)

# Etiketlerin One-Hot Encoding ile Dönüştürülmesi

y\_train\_new = []

for i in y\_train:

y\_train\_new.append(labels.index(i))

y\_train = y\_train\_new

y\_train = tf.keras.utils.to\_categorical(y\_train)

y\_test\_new = []

for i in y\_test:

y\_test\_new.append(labels.index(i))

y\_test = y\_test\_new

y\_test = tf.keras.utils.to\_categorical(y\_test)

# Modelin Oluşturulması

Xception = tf.keras.applications.Xception(weights='imagenet', include\_top=False, input\_shape=(image\_size, image\_size, 3))

model = Xception.output

model = tf.keras.layers.GlobalAveragePooling2D()(model)

model = tf.keras.layers.Dense(1024,activation='relu')(model)

model = tf.keras.layers.Dropout(rate=0.4)(model)

model = tf.keras.layers.Dense(4,activation='softmax')(model)

model = tf.keras.models.Model(inputs=Xception.input, outputs = model)

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

# Modelin Özeti

model.summary()

# Callback Fonksiyonlarının Tanımlanması

tensorboard = TensorBoard(log\_dir='logs')

checkpoint = ModelCheckpoint("xceptionv2.keras", monitor="val\_accuracy", save\_best\_only=True, verbose=1)

reduce\_lr = ReduceLROnPlateau(monitor='val\_accuracy', factor=0.1, patience=2,

min\_delta=0.0001, verbose=1)

# Modelin Eğitilmesi

history = model.fit(X\_train,y\_train,validation\_split=0.1, epochs =12, verbose=1, batch\_size=16,

callbacks=[tensorboard,checkpoint,reduce\_lr])

# Eğitim ve Doğrulama Performansının Görselleştirilmesi

plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)

plt.plot(history.history['loss'], label='Train Loss')

plt.plot(history.history['val\_loss'], label='Validation Loss')

plt.title('Training and Validation Loss')

plt.xlabel('Epoch')

plt.ylabel('Loss')

plt.legend()

plt.grid(True)

mplcyberpunk.make\_lines\_glow()

plt.subplot(1, 2, 2)

plt.plot(history.history['accuracy'], label='Train Accuracy')

plt.plot(history.history['val\_accuracy'], label='Validation Accuracy')

plt.title('Training and Validation Accuracy')

plt.xlabel('Epoch')

plt.ylabel('Accuracy')

plt.legend()

plt.grid(True)

mplcyberpunk.make\_lines\_glow()

plt.tight\_layout()

plt.show()

# Confusion Matrix Görselleştirilmesi

y\_true\_test = np.argmax(y\_test, axis=1)

y\_pred\_test = np.argmax(model.predict(X\_test), axis=1)

heatmap = sns.heatmap(confusion\_matrix(y\_true\_test,y\_pred\_test), annot=True, fmt='d', cmap='Blues\_r',

xticklabels=labels, yticklabels=labels)

plt.title('Confusion Matrix')

plt.xlabel('Predicted Label')

plt.ylabel('True Label')

plt.show()

# Sınıflandırma Raporunun Görüntülenmesi

print(classification\_report(y\_true\_test,y\_pred\_test))

# Modelin Rastgele Bir Görüntü Üzerinde Tahmin Yapması

random\_index = np.random.randint(0, len(X\_test))

random\_img = X\_test[random\_index]

predictions = model.predict(random\_img.reshape(1, 150, 150, 3)) # Reshape and preprocess the image

# Modelin tahminlerini yorumlama

predicted\_class = np.argmax(predictions) # Get the index of the class with the highest probability

predicted\_label = labels[predicted\_class] # Convert class to label

confidence = predictions[0][predicted\_class]

actual\_index = y\_test[random\_index] # Get the one-hot encoded actual class

actual\_class = np.argmax(actual\_index)

actual\_label = labels[actual\_class]

# Görüntüyü ve tahmin bilgilerini gösterme

print(f"\033[94mPredicted label: {predicted\_label}\033[0m \n\033[92mActual label: {actual\_label}\033[0m \n\033[93mConfidence: {confidence\*100:.2f}%\033[0m\n")

plt.figure(figsize = (3,3))

plt.imshow(random\_img)

plt.axis('off')

plt.show()

***Kaba Kod***

***1.Gerekli Kütüphaneleri İçe Aktar***

* + *os, numpy, pandas, seaborn, matplotlib, mplcyberpunk, cv2, tensorflow, sklearn, tqdm kütüphanelerini yükle.*

***2.Veri Setini Yükle ve Hazırla***

* + *X\_train ve y\_train adında boş listeler oluştur.*
  + *Etiketleri içeren bir liste oluştur: ['glioma\_tumor', 'no\_tumor', 'meningioma\_tumor', 'pituitary\_tumor'].*
  + *Görüntü boyutunu 150x150 olarak ayarla.*

***3.Eğitim ve Test Görsellerini Yükle***

* + *Her etiket için eğitim klasöründen görselleri yükle.*
  + *Görselleri belirtilen boyuta yeniden boyutlandır ve X\_train listesine ekle.*
  + *y\_train listesine ise ilgili etiketi ekle.*
  + *Aynı işlemi test klasörü için de tekrar et.*

***4.Veriyi Numpy Dizilerine Dönüştür***

* + *X\_train ve y\_train listelerini Numpy dizilerine dönüştür.*

***5.Etiketlerin Dağılımını Görselleştir***

* + *y\_train dizisindeki her etiket için bir histogram oluştur.*
  + *Her etiketin örnek görsellerini grafik üzerinde göster.*

***6****.****Veriyi Karıştır ve Eğitim-Test Ayrımı Yap***

* + *Veriyi karıştır.*
  + *Eğitim ve test verisi olarak 0.1 oranında ayır.*

***7.Etiketleri Sayısal Hale Getir***

* *y\_train ve y\_test etiketlerini indekslerine göre sayısal değerlere dönüştür.*

***8.Önceden Eğitilmiş Xception Modelini Yükle***

* *Xception modelini ImageNet ağırlıklarıyla yükle, en üst katmanı çıkar.*
  + *Modeli özelleştir: Global avrage pooling yap, 1024 boyutunda gizli katman, 0.4 bırakma oranı (dropout), softmax katmanı ekle.*

***9.Modeli Derle***

* + *Adam optimizasyon algoritmasını kullanarak modeli derle, kayıp fonksiyonu olarak categorical\_crossentropy kullan.*

***10.Model Eğitim Süreci için Callback İşlevlerini Ayarla***

* + *TensorBoard ile eğitim metriklerini günlüğe kaydet.*
  + *En iyi modeli kaydetmek için ModelCheckpoint kullan.*
  + *Eğitim doğruluğu sabit kalırsa öğrenme oranını azaltmak için ReduceLROnPlateau ekle.*

***11.Modeli Eğit***

* + *Modeli eğit, 0.1 doğrulama oranı, 12 dönem (epoch) ve 16 batch boyutuyla.*
  + *Callback işlevlerini eğitim sürecine dahil et.*

***12.Eğitim ve Doğrulama Kayıp/Doğruluk Grafiklerini Çiz***

* + *Eğitim ve doğrulama kayıplarını grafik olarak çiz.*
  + *Eğitim ve doğrulama doğruluklarını çiz.*

***13.Test Verisinde Model Performansını Değerlendir***

* + *confusion\_matrix ile karışıklık matrisi oluştur, görselleştir.*
  + *classification\_report ile doğruluk, kesinlik, duyarlılık ve F1 skorunu yazdır.*

***14.Rastgele Bir Test Görseli Üzerinde Modelin Tahminini Görselleştir***

* + *Test verisinden rastgele bir görsel seç.*
  + *Görsel üzerinde modelin tahminini yap ve doğruluğunu göster.*
  + *Görseli, tahmini etiket ve güven yüzdesiyle beraber görüntüle.*

***15.Flask Api Yapısı Oluştur***

* + *Flask API yapısında, kullanıcıların web arayüzünden görsel yükleyerek modelin tahmin yapmasını sağlayan bir yapı oluştur.*

***16.Web Arayüzü Oluştur***

* + *Web arayüzü, kullanıcıların bir görsel yükleyebileceği, sonucu görebileceği HTML-CSS-JS temelli basit bir yapı sağla.*

***UML Diyagramı***

***metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu***