In [290	<pre>labels = ['glioma_tumor', 'meningioma_tumor', 'no_tumor', 'pituitary_tumor']</pre>
	<pre>X_train = [] Y_train = []</pre>
	<pre>X_test = [] Y_test = []</pre>
	<pre>image_size=200 for label in labels: trainPath = as noth isin(!brain tumor(train! label))</pre>
	<pre>trainPath = os.path.join('brain_tumor/train',label) for file in tqdm(os.listdir(trainPath)): image = cv2.imread(os.path.join(trainPath, file),0)</pre>
	<pre>image = cv2.resize(image, (image_size, image_size)) X_train.append(image)</pre>
	Y_train.append(label)
	<pre>X_train = np.array(X_train)</pre>
	100% 826/826 [00:00<00:00, 1152.85it/s] 100% 822/822 [00:00<00:00, 1192.28it/s] 100% 395/395 [00:00<00:00, 1270.34it/s]
	100% 395/395 [00:00<00:00, 1270:341t/s] 100% 827/827 [00:00<00:00, 948.43it/s]
In [291	for label in labels:
	<pre>testPath = os.path.join('brain_tumor/test',label) for file in tqdm(os.listdir(testPath)):</pre>
	<pre>image = cv2.imread(os.path.join(testPath, file),0) image = cv2.resize(image, (image_size, image_size))</pre>
	<pre>X_test.append(image) Y_test.append(label)</pre>
	<pre>X_test = np.array(X_test)</pre>
	100% 100/100 [00:00<00:00, 1165.89it/s] 100% 115/115 [00:00<00:00, 1589.40it/s]
	100% 113/113 [00:00<00:00, 1303:4011/3] 100% 1
In [292	<pre>X_train, Y_train = shuffle(X_train, Y_train, random_state=28)</pre>
T. 5000	
In [293	<pre>y_train_ = [] for i in Y_train:</pre>
	<pre>y_trainappend(labels.index(i)) Y_train = y_train_</pre>
	Y_train = tf.keras.utils.to_categorical(Y_train)
	<pre>y_test_ = [] for i in Y_test:</pre>
	<pre>y_testappend(labels.index(i)) Y_test = y_test_</pre>
	Y_test = tf.keras.utils.to_categorical(Y_test)
In [294	<pre>model=Sequential()</pre>
	<pre>model.add(Conv2D(16, kernel_size=(3, 3),activation='relu',input_shape=(200,200,1),padding='same')) model.add(MaxPooling2D(pool_size=2))</pre>
	<pre>model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',padding='same'))</pre>
	<pre>model.add(MaxPooling2D(pool_size=2))</pre>
	<pre>model.add(Conv2D(64, kernel_size=(3, 3), activation='relu', padding='same')) model.add(MaxPooling2D(pool_size=2))</pre>
	<pre>model.add(Flatten()) model.add(Dense(32, activation='relu'))</pre>
	<pre>model.add(BatchNormalization()) model.add(Dense(16, activation='relu')) model.add(BatchNormalization())</pre>
	<pre>model.add(BatchNormalization()) model.add(Dense(4, activation='softmax')) model.summary()</pre>
	Model: "sequential_25"
	Layer (type) Output Shape Param #
	======================================
	max_pooling2d_75 (MaxPoolin (None, 100, 100, 16) 0 g2D)
	conv2d_76 (Conv2D) (None, 100, 100, 32) 4640
	max_pooling2d_76 (MaxPoolin (None, 50, 50, 32) 0 g2D)
	conv2d_77 (Conv2D) (None, 50, 50, 64) 18496
	max_pooling2d_77 (MaxPoolin (None, 25, 25, 64) 0
	g2D) flatten_25 (Flatten) (None, 40000) 0
	dense_75 (Dense) (None, 32) 1280032
	batch_normalization_50 (Bat (None, 32) 128
	chNormalization) dense_76 (Dense) (None, 16) 528
	batch_normalization_51 (Bat (None, 16) 64
	chNormalization)
	dance 77 (Dance) (None 4) 69
	dense_77 (Dense) (None, 4) 68
	======================================
	======================================
	Total params: 1,304,116 Trainable params: 96 X_train, X_val, Y_train, Y_val = train_test_split(X_train, Y_train, test_size=0.2, random_state=28)
In [295	Total params: 1,304,116 Trainable params: 96
In [295 Out[295	Total params: 1,304,116 Trainable params: 96 X_train, X_val, Y_train, Y_val = train_test_split(X_train, Y_train, test_size=0.2, random_state=28) X_train.shape, X_test.shape ((2296, 200, 200), (394, 200, 200))
In [295 Out[295	Total params: 1,304,116 Trainable params: 96 X_train, X_val, Y_train, Y_val = train_test_split(X_train, Y_train, test_size=0.2, random_state=28) X_train.shape, X_test.shape
In [295 Out[295 In [296	Total params: 1,304,116 Trainable params: 96 X_train, X_val, Y_train, Y_val = train_test_split(X_train, Y_train, test_size=0.2, random_state=28) X_train.shape, X_test.shape ((2296, 200, 200), (394, 200, 200))
In [295 Out[295 In [296 In [297	Total params: 1,304,116 Trainable params: 1,304,020 Non-trainable params: 96 X_train, X_val, Y_train, Y_val = train_test_split(X_train, Y_train, test_size=0.2, random_state=28) X_train.shape, X_test.shape ((2296, 200, 200), (394, 200, 200)) model.compile(loss='categorical_crossentropy', optimizer='Adam', metrics=['accuracy'])
In [295 Out[295 In [296 In [297	Total params: 1,304,116 Trainable params: 96 X_train, X_val, Y_train, Y_val = train_test_split(X_train, Y_train, test_size=0.2, random_state=28) X_train.shape, X_test.shape ((2296, 200, 200), (394, 200, 200)) model.compile(loss='categorical_crossentropy', optimizer='Adam', metrics=['accuracy']) CNN = model.fit(X_train, Y_train, batch_size=32, validation_data=(X_val, Y_val),epochs=10) Epoch 1/10 72/72 [==========] - 3s 33ms/step - loss: 0.9418 - accuracy: 0.6241 - val_loss: 1.3095 - val_accuracy: 0.4181 Epoch 2/10 72/72 [=========] - 2s 30ms/step - loss: 0.5900 - accuracy: 0.7905 - val_loss: 1.2163 - val_accuracy: 0.4460
In [295 Out[295 In [296 In [297	Total params: 1,304,116 Trainable params: 1,304,120 Non-trainable params: 96 X_train, X_val, Y_train, Y_val = train_test_split(X_train, Y_train, test_size=0.2, random_state=28) X_train.shape, X_test.shape ((2296, 200, 200), (394, 200, 200)) model.compile(loss='categorical_crossentropy', optimizer='Adam', metrics=['accuracy']) CNN = model.fit(X_train, Y_train, batch_size=32, validation_data=(X_val, Y_val),epochs=10) Epoch 1/10 72/72 [====================================
In [295 Out[295 In [296 In [297	Total params: 1,384,116 Trainable params: 96 X_train, X_val, Y_train, Y_val = train_test_split(X_train, Y_train, test_size=0.2, random_state=28) X_train, shape, X_test_shape ((2296, 280, 280), (394, 280, 280)) model.compile(loss='categorical_crossentropy', optimizer='Adam', metrics=['accuracy']) CNN = model.fit(X_train, Y_train, batch_size=32, validation_data=(X_val, Y_val),epochs=10) Epoch 1/10 72/72 [====================================
In [295 Out[295 In [296 In [297	Total params: 1,304,016 Trainable params: 96 X.train, X val, Y.train, Y.val = train test.split(X.train, Y.train, test.size=0.2, random.state=28) X.train, shape, X.test.shape ((2296, 200, 200), (394, 200, 200)) model.compile(loss='categorical_crossentropy', optimizer='Adam', metrics=['accuracy']) CNN = model.fit(X.train, Y.train, batch_size=32, validation_data=(X.val, Y.val),epochs=10) Epoch 1/10 72/72 [====================================
In [295 Out[295 In [296 In [297	Total params: 1,304,116 Trainable params: 1,304,120 Non-trainable params: 96 X_train, X_val, Y_train, Y_val = train_test_split(X_train, Y_train, test_size=0.2, random_state=28) X_train.shape, X_test.shape ((2296, 200, 200), (394, 200, 200)) model.compile(loss='categorical_crossentropy', optimizer='Adam', metrics=['accuracy']) CNN = model.fit(X_train, Y_train, batch_size=32, validation_data=(X_val, Y_val), epochs=10) Epoch 1/10 72/72 [====================================
In [295 Out[295 In [296 In [297	Total params: 1,304,116 Trainable params: 96 Non-trainable params: 96 X_train, X_val, Y_train, Y_val = train_test_split(X_train, Y_train, test_size=0.2, random_state=28) X_train.shape, X_Test.shape ((2296, 200, 200), (394, 200, 200)) model.compile(loss='categorical_crossentropy', optimizer='Adam', metrics=['accuracy']) CMN = model.fit(X_train, Y_train, batch size=32, validation data=(X_val, Y_val),epochs=10) Epoch 1/10 72/72 [==========] - 35 33ms/step - loss: 0.9418 - accuracy: 0.6241 - val_loss: 1.3905 - val_accuracy: 0.4181 [Epoch 2/10] 72/72 [=========] - 25 30ms/step - loss: 0.9420 - accuracy: 0.7905 - val_accuracy: 0.4460 72/72 [===========] - 25 30ms/step - loss: 0.4227 - accuracy: 0.7905 - val_accuracy: 0.7108 Epoch 4/10 72/72 [===========] - 25 30ms/step - loss: 0.4227 - accuracy: 0.9168 - val_accuracy: 0.7108 Epoch 5/10 72/72 [============] - 25 30ms/step - loss: 0.8590 - accuracy: 0.9168 - val_loss: 0.6798 - val_accuracy: 0.7108 Epoch 6/10 72/72 [============] - 25 30ms/step - loss: 0.8255 - accuracy: 0.9168 - val_loss: 0.8944 - val_accuracy: 0.7491 Epoch 6/10 72/72 [=============] - 25 30ms/step - loss: 0.8255 - accuracy: 0.9855 - val_loss: 0.8042 - val_accuracy: 0.7962 Epoch 8/10 72/72 [=============] - 25 30ms/step - loss: 0.8047 - accuracy: 0.9888 - val_loss: 0.3842 - val_accuracy: 0.8924 Epoch 8/10 72/72 [=============] - 25 30ms/step - loss: 0.8047 - accuracy: 0.9888 - val_loss: 0.3864 - val_accuracy: 0.9924
In [295 Out[295 In [296 In [297	Total params: 1,304,116 Trainable params: 96 X_train, X_val, Y_train, Y_val = train_test_split(X_train, Y_train, test_size=0.2, random_state=28) X_train.shape, X_test.shape ((2296, 200, 200), (394, 200, 200)) model.compile(loss='categorical_crossentropy', optimizer='Adam', metrics=['accuracy']) CNN = model.fit(X_train, Y_train, batch_size=32, validation_data=(X_val, Y_val), epochs=18) ### Poor 1/18 ##
In [295 Out[295 In [296 In [297	Total params: 1,304,116 Trainable params: 96 X.train, X.val, Y.train, Y.val = train_test split(X.train, Y.train, test_size=0.2, random_state=28) X.train.shape, X.test.shape ((2296, 209, 200), (394, 209, 200)) model.compile(loss='categorical_crossentropy', optimizer='Adam', metrics=['accuracy']) CNN = model.fit(X.train, Y.train, batch size=32, validation_data=(X.val, Y.val),epochs=10) Epoch 1/10 T.77/2 [====================================
In [295 Out[295 In [296 In [297	Total params: 1, 304,116 Trainable params: 3, 304,126 Non-trainable params: 3, 304,126 Non-trainable params: 3, 304,220 Non-trainable params: 4, 304,220 Non-trainabl
In [295 Out[295 In [296 In [297	Total parames: 1, 304,116 Total parames: 1, 304,208 Word-Trainable parames: 90 X_train, X_val, Y_train, Y_val = train_test_split(X_train, Y_train, test_size=0.2, random_state=28) X_train, shape, X_test.shape ((2206, 200, 200), (304, 206, 200)) model_compile(loss='categorical_crossentropy', oprimizer='Adam', metrics=['accuracy']) CDW = model_fit(X_train, Y_train, batch_size=32, validation_date=(X_val, Y_val), epoche=10) Egoch_1/18 Tyr777 =================================
In [295 Out[295 In [296 In [297	Train, x val, Y train, Y val = train test split(X train, Y train, test sizes0.2, random states28)
In [295 Out[295 In [296 In [297	Total parass: 1,384,138 (1908
In [295 Out[295 In [296 In [297	Total paras: 4,304,115 Non-trainable paras: 4,304,115 Non-trainable paras: 4,004,115 Non-trainable paras: 90,000 Non-trainable paras: 90,000 Non-trainable paras: 90,000 X train, X vai, Y train, Y vai = Irain test split(X train, Y train, test sizes0.2, random states20) X train, Suppe, X test. shape ((2206, 200, 200), (394, 200, 200)) model. compile(loss='categorical_crossentropy', optimizers'Adam', metrics=['accuracy']) CON = model.tit(X.train, Y train, batch sizes2, validation.datas(X vai, Y.vai),epochs=10) Epoch 3/10 T2/72 [
In [295 Out[295 In [296 In [297	Train per men: 1,304,210 Train Per men: 1,304,070 Train Per me
In [295 Out[295 In [296 In [297	Total parents: 1,304,020 Total parents: 1,3
In [295 Out[295 In [296 In [297	Train per men: 1,304,210 Train Per men: 1,304,070 Train Per me
In [295 Out [295 In [296 In [297	Train prome: 1, 204, 208 Train prome: 1, 204, 208, 208) Train prome: 1, 204, 208, 208) Train prome: 1, 204, 208, 208, 208) Train prome: 1, 204, 208, 208, 208) Train prome: 1, 204, 208, 208, 208) Train prome: 1, 204, 208, 208, 208, 208, 208) Train prome: 1, 204, 208, 208, 208, 208, 208, 208) Train prome: 1, 204, 208, 208, 208, 208, 208, 208, 208, 208
In [295 Out [295 In [296 In [297 In [298	Description 1944 10 10 10 10 10 10 10
In [295 Out [295 In [296 In [297	Total paramet; 1,904; 196 Hairaidle paramet; 1,904; 196 Hairaidle paramet; 1,904; 196 Hairaidle paramet; 1,904; 196 We retrained paramet; 1,904; 196 X trans, X val, Y train, V val = train fact split(% train, Y train, isst strain)
In [295 Out [295 In [297 In [298	Triangle parame: 1,394,125 Triangle parame: 2,394,225 X. Irani, X. val. Y. Lovin, Y. val. = Ursin_test_split(X_train, Y_train, test_slace=0.2, nambor_slate=20) X. Irani, X. val. Y. Lovin, Y. val. = Ursin_test_split(X_train, Y_train, test_slace=0.2, nambor_slate=20) X. Irani, X. val. Y. Lovin, Y. val. = Ursin_test_split(X_train, Y_train, test_slace=0.2, nambor_slate=20) X. Irani, X. val. Y. Lovin, Y. val. = Ursin_test_split(X_train, Y_train, test_slace=0.2, nambor_slate=20) X. Irani, X. val. Y. Lovin, Y. val. = Ursin_test_split(X_train, Y_train, test_slace=0.2, nambor_slate=20) X. Irani, X. val. Y. Lovin, Y. val. = Ursin_test_split(X_train, Y_train, test_slace=0.2, nambor_slate=20) X. Irani, X. val. Y. Val. = Ursin_test_split(X_train, Y_train, test_slace=0.2, nambor_slate=20) X. Irani, X. val. Y. Val. = Ursin_test_split(X_train, Y_train, test_slace=0.2, nambor_slate=20) X. Irani, X. val. Y. Val. = Ursin_test_split(X_train, Y_train, test_slace=0.2, nambor_slate=20) X. Irani, X. val. Y. Val. = Ursin_test_split(X_train, Y_train, test_slace=0.2, nambor_slate=20) X. Irani, X. val. Y. Val. = Ursin_test_split(X_train, Y_train, test_slace=0.2, nambor_slate=20) X. Irani, X. val. Y. Val. = Ursin_test_split(X_train, Y_train, test_slace=0.2, nambor_slate=20) X. Irani, X. val. Y. Val. = Ursin_test_split(X_train, Y_train, test_slace=0.2, nambor_slate=20) X. Irani, X. val. Y. Val. = Ursin_test_split(X_train, Y_train, test_slate=0.2, nambor_slate=20) X. Irani, X. val. Y. Val. = Ursin_test_split(X_train, Y_train, test_slate=20) X. Irani, X. val. Y. Val. = Ursin_test_split(X_train, Val. = Val. = Val. = Val. = Ursin_test_split(X_train, Val. =
In [295 Out [295 In [297 In [298	Note parase 1,200.116
In [295 Out [295 In [297 In [298	Total paramet 3.304,316 Total paramet 3.00,316 x Coal, x end, y (rain), y val = (rain) (set splic(x (rain, y (rain), test size=5.0, random size(end)) x Coal, x end, y (rain), y val = (rain) (set splic(x (rain, y (rain), test size=5.0, random size(end)) x Coal, x end, y (rain), y val = (rain) (set splic(x (rain, y (rain), test size=5.0, random size(end)) x Coal, x end, y (rain), y
In [295 Out [295 In [297 In [298	Note parase 1,200.116
In [295 Out [295 In [297 In [298	Trainable parmer 1, 200, 400 X (rai), x val), y (rain, x val) = train test split(x train, x train, jest size=2, notice state=3) X (rain, rain), x (rain, x val) = train test split(x train, x train, jest size=2, notice state=3) X (rain, rain), x (rain, rain) = train test split(x train, x train, jest size=2, notice state=3) X (rain, rain), x (rain, rain, rain, latel size=2, split(x train, x train, x train, latel size=2, split(x train, x tr
In [295 Out [295 In [297 In [298	Training parces 1.364,306
In [295 Out [295 In [297 In [298	Train Strain 1.184,118 Train College (1997) 1.194 (1997) Train (1998) 1.194 (1997) 1.194 (1997) Train (1998) 1.194 (1997) 1.194 (199
In [295 Out [295 In [297 In [298	Trainable parame: 1,784,125 Tr
In [295 Out [295 In [297 In [298	Miles 2000 1.18
In [295 Out [295 In [297 In [298	Miles Section 1, 164, 126
In [295 Out [295 In [297 In [298	Table Tabl
In [295 Out [295 In [297 In [298	Compared 1,204.15 20 20 20 20 20 20 20 2
In [295 Out [295 In [297 In [298	Control Cont
In [295 Out [295 In [297 In [298	Cold Descript 1, 1985 188 Cold
In [295 Out [295 In [297 In [298	Part (1998) 1, 16 m. 1 m

- 120

- 100

- 80

- 60

- 40

- 20

pituitary_tumor

160 169

72

173

574

574

574

meningioma_tumor

no_tumor ·

pituitary_tumor

[[153 5 1 1] [8 146 13 2] [5 5 61 1] [2 6 1 164]]

print(report)

accuracy macro avg

weighted avg

0

1

2

In [300...

In []:

In []:

In []:

glioma_tumor meningioma_tumor no_tumor

0.96

0.86

0.85

0.95

0.90

0.91

recall f1-score support

0.93

0.88

0.82

0.96

0.91

0.90

0.91

report = classification_report(y_test, y_pred)

precision

0.91

0.90

0.80

0.98

0.90

0.91

In [289...

import numpy as np
import pandas as pd

from tqdm import tqdm
import tensorflow as tf

from sklearn.utils import shuffle

from tensorflow.keras.preprocessing import image

import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Input, Lambda, Dense, Flatten, Conv2D, BatchNormalization

from tensorflow.keras.preprocessing.image import ImageDataGenerator,load_img
from sklearn.model_selection import train_test_split

from tensorflow.keras.models import Sequential
from keras.layers.advanced_activations import LeakyReLU
from sklearn.metrics import classification_report,confusion_matrix
from keras.layers import Dense, Dropout, Flatten
import numpy as np
from glob import glob
from tensorflow.keras.layers import MaxPooling2D
from keras.models import Model

import os
import cv2