In [289…

```
import numpy as np
import pandas as pd
import os
import cv2
from tqdm import tqdm
import tensorflow as tf
from sklearn.utils import shuffle
from tensorflow.keras.layers import Input, Lambda, Dense, Flatten,Conv2D, BatchNormalization
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator,load_img
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
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
```

In [290…

```
labels = ['glioma_tumor', 'meningioma_tumor', 'no_tumor', 'pituitary_tumor']
X_train = []
Y_train  = []
X_test = []
Y_test = []
image_size=200
for label in labels:
    trainPath = os.path.join('brain_tumor/train',label)
    for file in tqdm(os.listdir(trainPath)):
        image = cv2.imread(os.path.join(trainPath, file),0)
        image = cv2.resize(image, (image_size, image_size))
        X_train.append(image)
        Y_train.append(label)

X_train = np.array(X_train)
```

```
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, 1279.34it/s]
100%|██████████| 827/827 [00:00<00:00, 948.43it/s]
```

In [291…

```
for label in labels:

    testPath = os.path.join('brain_tumor/test',label)
    for file in tqdm(os.listdir(testPath)):
        image = cv2.imread(os.path.join(testPath, file),0)
        image = cv2.resize(image, (image_size, image_size))
        X_test.append(image)
        Y_test.append(label)

X_test = np.array(X_test)
```

```
100%|██████████| 100/100 [00:00<00:00, 1165.89it/s]
100%|██████████| 115/115 [00:00<00:00, 1589.40it/s]
100%|██████████| 105/105 [00:00<00:00, 2367.28it/s]
100%|██████████| 74/74 [00:00<00:00, 495.65it/s]
```

In [292…

```
X_train, Y_train = shuffle(X_train, Y_train, random_state=28)
```

In [293…

```
y_train_ = []
for i in Y_train:
    y_train_.append(labels.index(i))
Y_train = y_train_

Y_train = tf.keras.utils.to_categorical(Y_train)

y_test_ = []
for i in Y_test:
    y_test_.append(labels.index(i))
Y_test = y_test_

Y_test = tf.keras.utils.to_categorical(Y_test)
```

In [294…

```
model=Sequential()
model.add(Conv2D(16, kernel_size=(3, 3),activation='relu',input_shape=(200,200,1),padding='same'))
model.add(MaxPooling2D(pool_size=2))


model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',padding='same'))
model.add(MaxPooling2D(pool_size=2))

model.add(Conv2D(64, kernel_size=(3, 3), activation='relu',padding='same'))
model.add(MaxPooling2D(pool_size=2))

model.add(Flatten())
model.add(Dense(32, activation='relu'))
model.add(BatchNormalization())
model.add(Dense(16, activation='relu'))
model.add(BatchNormalization())
model.add(Dense(4, activation='softmax'))
model.summary()
```

```
Model: "sequential_25"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_75 (Conv2D)          (None, 200, 200, 16)      160

 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)

 dense_77 (Dense)            (None, 4)                 68

=================================================================
Total params: 1,304,116
Trainable params: 1,304,020
Non-trainable params: 96
_________________________________________________________________
```

In [295…

```
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
```

Out[295…

```
((2296, 200, 200), (394, 200, 200))
```

In [296…

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

In [297…

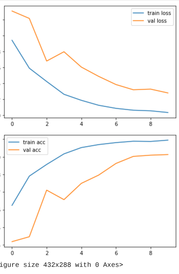
```
CNN = model.fit(X_train, Y_train, batch_size=32, validation_data=(X_val, Y_val),epochs=10)
```

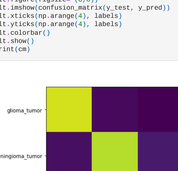
```
Epoch 1/10
72/72 [==============================] - 3s 33ms/step - loss: 0.9418 - accuracy: 0.6241 - val_loss: 1.3095 - val_accuracy: 0.4181
Epoch 2/10
72/72 [==============================] - 2s 30ms/step - loss: 0.5900 - accuracy: 0.7905 - val_loss: 1.2163 - val_accuracy: 0.4460
Epoch 3/10
72/72 [==============================] - 2s 30ms/step - loss: 0.4227 - accuracy: 0.8567 - val_loss: 0.6798 - val_accuracy: 0.7108
Epoch 4/10
72/72 [==============================] - 2s 30ms/step - loss: 0.2616 - accuracy: 0.9168 - val_loss: 0.7970 - val_accuracy: 0.6568
Epoch 5/10
72/72 [==============================] - 2s 30ms/step - loss: 0.1859 - accuracy: 0.9525 - val_loss: 0.6044 - val_accuracy: 0.7491
Epoch 6/10
72/72 [==============================] - 2s 30ms/step - loss: 0.1235 - accuracy: 0.9695 - val_loss: 0.4872 - val_accuracy: 0.7962
Epoch 7/10
72/72 [==============================] - 2s 30ms/step - loss: 0.0847 - accuracy: 0.9808 - val_loss: 0.3842 - val_accuracy: 0.8624
Epoch 8/10
72/72 [==============================] - 2s 30ms/step - loss: 0.0612 - accuracy: 0.9887 - val_loss: 0.3186 - val_accuracy: 0.9024
Epoch 9/10
72/72 [==============================] - 2s 30ms/step - loss: 0.0537 - accuracy: 0.9874 - val_loss: 0.3266 - val_accuracy: 0.9094
Epoch 10/10
72/72 [==============================] - 2s 30ms/step - loss: 0.0323 - accuracy: 0.9948 - val_loss: 0.2784 - val_accuracy: 0.9129
```

In [298…

```
plt.plot(CNN.history['loss'], label='train loss')
plt.plot(CNN.history['val_loss'], label='val loss')
plt.legend()
plt.show()
plt.savefig('LossVal_loss')

plt.plot(CNN.history['accuracy'], label='train acc')
plt.plot(CNN.history['val_accuracy'], label='val acc')
plt.legend()
plt.show()
plt.savefig('AccVal_acc')
```

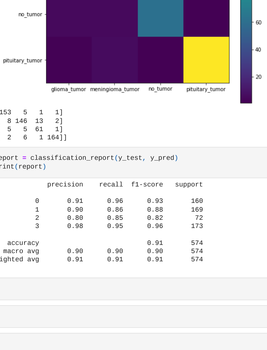





```
<Figure size 432x288 with 0 Axes>
```

In [299…

```
y_test=np.argmax(Y_val,axis=1)
y_pred = np.argmax(model.predict(X_val), axis=-1)
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize= (8,8))
plt.imshow(confusion_matrix(y_test, y_pred))
plt.xticks(np.arange(4), labels)
plt.yticks(np.arange(4), labels)
plt.colorbar()
plt.show()
print(cm)
```




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

In [300…

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

```
              precision    recall  f1-score   support

           0       0.91      0.96      0.93       160
           1       0.90      0.86      0.88       169
           2       0.88      0.85      0.82        72
           3       0.98      0.95      0.96       173

    accuracy                           0.91       574
   macro avg       0.90      0.90      0.90       574
weighted avg       0.91      0.91      0.91       574
```

In [ ]:

In [ ]:

In [ ]: