In [21]:
```
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 [22]:
```
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 [00:01<00:00, 552.14it/s]
100%|██████████| 822/822 [00:01<00:00, 636.571it/s]
100%|██████████| 395/395 [00:00<00:00, 633.63it/s]
100%|██████████| 827/827 [00:01<00:00, 546.27it/s]
```

In [23]:
```
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, 557.82it/s]
100%|██████████| 115/115 [00:00<00:00, 460.54it/s]
100%|██████████| 105/105 [00:00<00:00, 836.35it/s]
100%|██████████| 74/74 [00:00<00:00, 139.26it/s]
```

In [24]:
```
X_train, Y_train = shuffle(X_train, Y_train, random_state=28)
```

In [25]:
```
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 [26]:
```
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_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 200, 200, 16)      160

max_pooling2d_3 (MaxPooling  (None, 100, 100, 16)      0
2D)

conv2d_4 (Conv2D)            (None, 100, 100, 32)      4640

max_pooling2d_4 (MaxPooling  (None, 50, 50, 32)        0
2D)

conv2d_5 (Conv2D)            (None, 50, 50, 64)        18496

max_pooling2d_5 (MaxPooling  (None, 25, 25, 64)        0
2D)

flatten_1 (Flatten)          (None, 40000)             0

dense_3 (Dense)              (None, 32)                1280032

batch_normalization_2 (Batc  (None, 32)                128
hNormalization)

dense_4 (Dense)              (None, 16)                528

batch_normalization_3 (Batc  (None, 16)                64
hNormalization)

dense_5 (Dense)              (None, 4)                 68

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

In [27]:
```
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[27]:
```
((2296, 200, 200), (394, 200, 200))
```

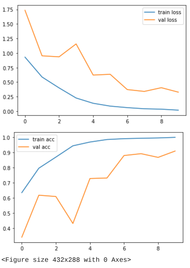
In [28]:
```
model.compile(loss='categorical_crossentropy', optimizer='Adam', metrics=['accuracy'])
```

In [29]:
```
CNN = model.fit(X_train, Y_train, batch_size=32, validation_data=(X_val, Y_val),epochs=10)
```

```
Epoch 1/10
72/72 [==============================] - 11s 123ms/step - loss: 0.9301 - accuracy: 0.6350 - val_loss: 1.7361 - val_accuracy: 0.3415
Epoch 2/10
72/72 [==============================] - 8s 116ms/step - loss: 0.5880 - accuracy: 0.7962 - val_loss: 0.9525 - val_accuracy: 0.6185
Epoch 3/10
72/72 [==============================] - 8s 113ms/step - loss: 0.4007 - accuracy: 0.8693 - val_loss: 0.9349 - val_accuracy: 0.6098
Epoch 4/10
72/72 [==============================] - 8s 113ms/step - loss: 0.2268 - accuracy: 0.9438 - val_loss: 1.1571 - val_accuracy: 0.4321
Epoch 5/10
72/72 [==============================] - 8s 113ms/step - loss: 0.1360 - accuracy: 0.9686 - val_loss: 0.6213 - val_accuracy: 0.7282
Epoch 6/10
72/72 [==============================] - 8s 113ms/step - loss: 0.0873 - accuracy: 0.9852 - val_loss: 0.6347 - val_accuracy: 0.7317
Epoch 7/10
72/72 [==============================] - 8s 112ms/step - loss: 0.0613 - accuracy: 0.9909 - val_loss: 0.3720 - val_accuracy: 0.8798
Epoch 8/10
72/72 [==============================] - 8s 113ms/step - loss: 0.0418 - accuracy: 0.9935 - val_loss: 0.3398 - val_accuracy: 0.8920
Epoch 9/10
72/72 [==============================] - 8s 113ms/step - loss: 0.0346 - accuracy: 0.9956 - val_loss: 0.4041 - val_accuracy: 0.8676
Epoch 10/10
72/72 [==============================] - 8s 113ms/step - loss: 0.0179 - accuracy: 0.9996 - val_loss: 0.3273 - val_accuracy: 0.9094
```

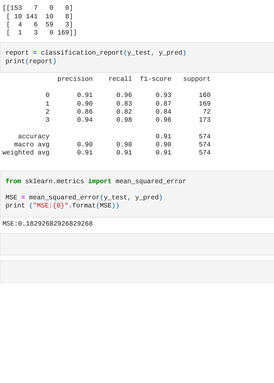
In [30]:
```
plt.plot(CNN.history['loss'], label='train loss')
plt.plot(CNN.history['val_loss'], label='val loss')
plt.legend()
plt.show()
plt.savefig('LossVal_loss')

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 [31]:
```
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   7   0   0]
 [ 10 141  10   8]
 [  4   6  59   3]
 [  1   3   0 169]]
```

In [32]:
```
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      0.83      0.87       169
           2       0.86      0.82      0.84        72
           3       0.94      0.98      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 [33]:
```
from sklearn.metrics import mean_squared_error

MSE = mean_squared_error(y_test, y_pred)
print ("MSE:{0}".format(MSE))
```

```
MSE:0.18292682926829268
```

In [ ]:

In [ ]: