```
# This Python 3 environment comes with many helpful analytics
libraries installed
# It is defined by the kaggle/python Docker image:
https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/"
directory
# For example, running this (by clicking run or pressing Shift+Enter)
will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        os.path.join(dirname, filename)

# You can write up to 20GB to the current directory (/kaggle/working/)
that gets preserved as output when you create a version using "Save &
Run All"
# You can also write temporary files to /kaggle/temp/, but they won't
be saved outside of the current session
```

Importing Libraries

```
import keras
from keras.models import Sequential
from keras.layers import Conv2D,Flatten,Dense,MaxPooling2D,Dropout
from sklearn.metrics import accuracy_score

import ipywidgets as widgets
import io
from PIL import Image
import tqdm
from sklearn.model_selection import train_test_split
import cv2
from sklearn.utils import shuffle
import tensorflow as tf
```

Folder Paths

```
X_train = []
Y_train = []
image_size = 150
```

```
labels =
['glioma_tumor','meningioma_tumor','no_tumor','pituitary_tumor']
for i in labels:
    folderPath = os.path.join('../input/brain-tumor-classification-
mri/Training',i)
    for j in 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('../input/brain-tumor-classification-
mri/Testing',i)
    for j in 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)

X_train,Y_train = shuffle(X_train,Y_train,random_state=101)
X_train.shape
```

```
(3264, 150, 150, 3)
```

Train - Test Split

```
X_train,X_test,y_train,y_test =
train_test_split(X_train,Y_train,test_size=0.1,random_state=101)

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)
```

Convolutional Neural Network

```
model = Sequential()
model.add(Conv2D(32,(3,3),activation =
'relu',input_shape=(150,150,3)))
model.add(Conv2D(64,(3,3),activation='relu'))
model.add(MaxPooling2D(2,2))
model.add(Dropout(0.3))
model.add(Conv2D(64,(3,3),activation='relu'))
model.add(Conv2D(64,(3,3),activation='relu'))
model.add(Dropout(0.3))
model.add(MaxPooling2D(2,2))
model.add(Dropout(0.3))
model.add(Conv2D(128,(3,3),activation='relu'))
model.add(Conv2D(128,(3,3),activation='relu'))
model.add(Conv2D(128,(3,3),activation='relu'))
model.add(MaxPooling2D(2,2))
model.add(Dropout(0.3))
model.add(Conv2D(128,(3,3),activation='relu'))
model.add(Conv2D(256,(3,3),activation='relu'))
model.add(MaxPooling2D(2,2))
model.add(Dropout(0.3))
model.add(Flatten())
model.add(Dense(512,activation = 'relu'))
model.add(Dense(512,activation = 'relu'))
model.add(Dropout(0.3))
model.add(Dense(4,activation='softmax'))

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 148, 148, 32)      896
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 146, 146, 64)      18496
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 73, 73, 64)        0
_________________________________________________________________
dropout (Dropout)            (None, 73, 73, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 71, 71, 64)        36928
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 69, 69, 64)        36928
_________________________________________________________________
dropout_1 (Dropout)          (None, 69, 69, 64)        0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 34, 34, 64)        0
_________________________________________________________________
```

```
dropout_2 (Dropout)          (None, 34, 34, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 32, 32, 128)       73856
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 30, 30, 128)       147584
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 28, 28, 128)       147584
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 14, 14, 128)       0
_________________________________________________________________
dropout_3 (Dropout)          (None, 14, 14, 128)       0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 12, 12, 128)       147584
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 10, 10, 256)       295168
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 5, 5, 256)         0
_________________________________________________________________
dropout_4 (Dropout)          (None, 5, 5, 256)         0
_________________________________________________________________
flatten (Flatten)            (None, 6400)              0
_________________________________________________________________
dense (Dense)                (None, 512)               3277312
_________________________________________________________________
dense_1 (Dense)              (None, 512)               262656
_________________________________________________________________
dropout_5 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 4)                 2052
=================================================================
Total params: 4,447,044
Trainable params: 4,447,044
Non-trainable params: 0
_________________________________________________________________
```

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

history = model.fit(X_train,y_train,epochs = 100,validation_split =
0.1)
```

```
Epoch 1/100
83/83 [==============================] - 12s 49ms/step - loss: 2.0064
- accuracy: 0.3061 - val_loss: 1.2199 - val_accuracy: 0.4048
Epoch 2/100
83/83 [==============================] - 3s 39ms/step - loss: 1.1622 -
accuracy: 0.4873 - val_loss: 1.0385 - val_accuracy: 0.6020
Epoch 3/100
83/83 [==============================] - 3s 38ms/step - loss: 0.9976 -
accuracy: 0.5687 - val_loss: 0.8276 - val_accuracy: 0.6531
```

```
Epoch 4/100
83/83 [==============================] - 3s 38ms/step - loss: 0.8881 -
accuracy: 0.6216 - val_loss: 0.7721 - val_accuracy: 0.7347
Epoch 5/100
83/83 [==============================] - 3s 39ms/step - loss: 0.7646 -
accuracy: 0.6833 - val_loss: 0.7475 - val_accuracy: 0.6429
Epoch 6/100
83/83 [==============================] - 3s 39ms/step - loss: 0.6877 -
accuracy: 0.7102 - val_loss: 0.7223 - val_accuracy: 0.6769
Epoch 7/100
83/83 [==============================] - 3s 38ms/step - loss: 0.5785 -
accuracy: 0.7582 - val_loss: 0.6717 - val_accuracy: 0.6837
Epoch 8/100
83/83 [==============================] - 3s 39ms/step - loss: 0.5407 -
accuracy: 0.7779 - val_loss: 0.5407 - val_accuracy: 0.7619
Epoch 9/100
83/83 [==============================] - 3s 38ms/step - loss: 0.4540 -
accuracy: 0.8237 - val_loss: 0.6045 - val_accuracy: 0.7211
Epoch 10/100
83/83 [==============================] - 3s 38ms/step - loss: 0.3713 -
accuracy: 0.8577 - val_loss: 0.5622 - val_accuracy: 0.7483
Epoch 11/100
83/83 [==============================] - 3s 38ms/step - loss: 0.3655 -
accuracy: 0.8676 - val_loss: 0.4490 - val_accuracy: 0.8027
Epoch 12/100
83/83 [==============================] - 3s 39ms/step - loss: 0.2938 -
accuracy: 0.8899 - val_loss: 0.3249 - val_accuracy: 0.8980
Epoch 13/100
83/83 [==============================] - 3s 38ms/step - loss: 0.2687 -
accuracy: 0.9016 - val_loss: 0.4430 - val_accuracy: 0.8673
Epoch 14/100
83/83 [==============================] - 3s 39ms/step - loss: 0.2379 -
accuracy: 0.9134 - val_loss: 0.3591 - val_accuracy: 0.8878
Epoch 15/100
83/83 [==============================] - 3s 38ms/step - loss: 0.1894 -
accuracy: 0.9266 - val_loss: 0.3559 - val_accuracy: 0.8878
Epoch 16/100
83/83 [==============================] - 3s 38ms/step - loss: 0.1897 -
accuracy: 0.9353 - val_loss: 0.3875 - val_accuracy: 0.8878
Epoch 17/100
83/83 [==============================] - 3s 38ms/step - loss: 0.1993 -
accuracy: 0.9270 - val_loss: 0.4620 - val_accuracy: 0.8537
Epoch 18/100
83/83 [==============================] - 3s 38ms/step - loss: 0.1846 -
accuracy: 0.9319 - val_loss: 0.2990 - val_accuracy: 0.9014
Epoch 19/100
83/83 [==============================] - 3s 39ms/step - loss: 0.1471 -
accuracy: 0.9478 - val_loss: 0.2992 - val_accuracy: 0.8878
Epoch 20/100
```

```
83/83 [==============================] - 3s 40ms/step - loss: 0.1146 -
accuracy: 0.9607 - val_loss: 0.3241 - val_accuracy: 0.9150
Epoch 21/100
83/83 [==============================] - 3s 39ms/step - loss: 0.1186 -
accuracy: 0.9546 - val_loss: 0.4318 - val_accuracy: 0.8980
Epoch 22/100
83/83 [==============================] - 3s 39ms/step - loss: 0.1128 -
accuracy: 0.9610 - val_loss: 0.3043 - val_accuracy: 0.8980
Epoch 23/100
83/83 [==============================] - 3s 39ms/step - loss: 0.1702 -
accuracy: 0.9395 - val_loss: 0.2980 - val_accuracy: 0.9082
Epoch 24/100
83/83 [==============================] - 3s 39ms/step - loss: 0.0998 -
accuracy: 0.9705 - val_loss: 0.2898 - val_accuracy: 0.9184
Epoch 25/100
83/83 [==============================] - 3s 39ms/step - loss: 0.0961 -
accuracy: 0.9686 - val_loss: 0.3282 - val_accuracy: 0.8946
Epoch 26/100
83/83 [==============================] - 3s 39ms/step - loss: 0.0795 -
accuracy: 0.9720 - val_loss: 0.4459 - val_accuracy: 0.8810
Epoch 27/100
83/83 [==============================] - 3s 39ms/step - loss: 0.0892 -
accuracy: 0.9728 - val_loss: 0.3501 - val_accuracy: 0.9116
Epoch 28/100
83/83 [==============================] - 3s 39ms/step - loss: 0.0535 -
accuracy: 0.9792 - val_loss: 0.4438 - val_accuracy: 0.9014
Epoch 29/100
83/83 [==============================] - 3s 39ms/step - loss: 0.1113 -
accuracy: 0.9629 - val_loss: 0.3327 - val_accuracy: 0.9184
Epoch 30/100
83/83 [==============================] - 3s 39ms/step - loss: 0.0924 -
accuracy: 0.9728 - val_loss: 0.3489 - val_accuracy: 0.8946
Epoch 31/100
83/83 [==============================] - 3s 39ms/step - loss: 0.0848 -
accuracy: 0.9716 - val_loss: 0.3647 - val_accuracy: 0.9116
Epoch 32/100
83/83 [==============================] - 3s 39ms/step - loss: 0.0965 -
accuracy: 0.9720 - val_loss: 0.3331 - val_accuracy: 0.9218
Epoch 33/100
83/83 [==============================] - 3s 39ms/step - loss: 0.0917 -
accuracy: 0.9694 - val_loss: 0.3029 - val_accuracy: 0.8980
Epoch 34/100
83/83 [==============================] - 3s 39ms/step - loss: 0.0710 -
accuracy: 0.9788 - val_loss: 0.3684 - val_accuracy: 0.9184
Epoch 35/100
83/83 [==============================] - 3s 39ms/step - loss: 0.0469 -
accuracy: 0.9879 - val_loss: 0.2970 - val_accuracy: 0.9320
Epoch 36/100
83/83 [==============================] - 3s 39ms/step - loss: 0.0482 -
```

```
accuracy: 0.9849 - val_loss: 0.3606 - val_accuracy: 0.9218
Epoch 37/100
83/83 [==============================] - 3s 39ms/step - loss: 0.0723 -
accuracy: 0.9777 - val_loss: 0.3272 - val_accuracy: 0.9218
Epoch 38/100
83/83 [==============================] - 3s 39ms/step - loss: 0.0558 -
accuracy: 0.9837 - val_loss: 0.3616 - val_accuracy: 0.9320
Epoch 39/100
83/83 [==============================] - 3s 38ms/step - loss: 0.0693 -
accuracy: 0.9788 - val_loss: 0.2860 - val_accuracy: 0.9150
Epoch 40/100
83/83 [==============================] - 3s 38ms/step - loss: 0.0407 -
accuracy: 0.9879 - val_loss: 0.4012 - val_accuracy: 0.9218
Epoch 41/100
83/83 [==============================] - 3s 39ms/step - loss: 0.0467 -
accuracy: 0.9834 - val_loss: 0.5523 - val_accuracy: 0.9014
Epoch 42/100
83/83 [==============================] - 3s 39ms/step - loss: 0.0427 -
accuracy: 0.9875 - val_loss: 0.4047 - val_accuracy: 0.9286
Epoch 43/100
83/83 [==============================] - 3s 39ms/step - loss: 0.0491 -
accuracy: 0.9834 - val_loss: 0.4472 - val_accuracy: 0.9218
Epoch 44/100
83/83 [==============================] - 3s 39ms/step - loss: 0.0616 -
accuracy: 0.9811 - val_loss: 0.4619 - val_accuracy: 0.9286
Epoch 45/100
83/83 [==============================] - 3s 39ms/step - loss: 0.0545 -
accuracy: 0.9837 - val_loss: 0.3963 - val_accuracy: 0.9252
Epoch 46/100
83/83 [==============================] - 3s 39ms/step - loss: 0.0488 -
accuracy: 0.9845 - val_loss: 0.2931 - val_accuracy: 0.9218
Epoch 47/100
83/83 [==============================] - 3s 39ms/step - loss: 0.0462 -
accuracy: 0.9811 - val_loss: 0.2721 - val_accuracy: 0.9252
Epoch 48/100
83/83 [==============================] - 3s 39ms/step - loss: 0.0754 -
accuracy: 0.9822 - val_loss: 0.3569 - val_accuracy: 0.9286
Epoch 49/100
83/83 [==============================] - 3s 39ms/step - loss: 0.0673 -
accuracy: 0.9799 - val_loss: 0.3407 - val_accuracy: 0.9252
Epoch 50/100
83/83 [==============================] - 3s 39ms/step - loss: 0.0494 -
accuracy: 0.9818 - val_loss: 0.3883 - val_accuracy: 0.9184
Epoch 51/100
83/83 [==============================] - 3s 39ms/step - loss: 0.0618 -
accuracy: 0.9818 - val_loss: 0.4251 - val_accuracy: 0.9388
Epoch 52/100
83/83 [==============================] - 3s 39ms/step - loss: 0.0433 -
accuracy: 0.9864 - val_loss: 0.3855 - val_accuracy: 0.9286
```

```
Epoch 53/100
83/83 [==============================] - 3s 39ms/step - loss: 0.0393 -
accuracy: 0.9871 - val_loss: 0.3134 - val_accuracy: 0.9286
Epoch 54/100
83/83 [==============================] - 3s 39ms/step - loss: 0.0512 -
accuracy: 0.9841 - val_loss: 0.3037 - val_accuracy: 0.9388
Epoch 55/100
83/83 [==============================] - 3s 38ms/step - loss: 0.0693 -
accuracy: 0.9803 - val_loss: 0.3245 - val_accuracy: 0.9184
Epoch 56/100
83/83 [==============================] - 3s 38ms/step - loss: 0.0515 -
accuracy: 0.9818 - val_loss: 0.5471 - val_accuracy: 0.9456
Epoch 57/100
83/83 [==============================] - 3s 38ms/step - loss: 0.0470 -
accuracy: 0.9837 - val_loss: 0.3904 - val_accuracy: 0.9184
Epoch 58/100
83/83 [==============================] - 3s 38ms/step - loss: 0.0323 -
accuracy: 0.9886 - val_loss: 0.5163 - val_accuracy: 0.9218
Epoch 59/100
83/83 [==============================] - 3s 38ms/step - loss: 0.0317 -
accuracy: 0.9883 - val_loss: 0.5323 - val_accuracy: 0.9116
Epoch 60/100
83/83 [==============================] - 3s 38ms/step - loss: 0.0939 -
accuracy: 0.9712 - val_loss: 0.3668 - val_accuracy: 0.9218
Epoch 61/100
83/83 [==============================] - 3s 38ms/step - loss: 0.0664 -
accuracy: 0.9788 - val_loss: 0.3360 - val_accuracy: 0.9286
Epoch 62/100
83/83 [==============================] - 3s 38ms/step - loss: 0.0677 -
accuracy: 0.9754 - val_loss: 0.3811 - val_accuracy: 0.9252
Epoch 63/100
83/83 [==============================] - 3s 39ms/step - loss: 0.0206 -
accuracy: 0.9939 - val_loss: 0.4042 - val_accuracy: 0.9184
Epoch 64/100
83/83 [==============================] - 3s 39ms/step - loss: 0.0551 -
accuracy: 0.9860 - val_loss: 0.3698 - val_accuracy: 0.9422
Epoch 65/100
83/83 [==============================] - 3s 39ms/step - loss: 0.0677 -
accuracy: 0.9773 - val_loss: 0.3146 - val_accuracy: 0.9286
Epoch 66/100
83/83 [==============================] - 3s 38ms/step - loss: 0.0402 -
accuracy: 0.9886 - val_loss: 0.3878 - val_accuracy: 0.9320
Epoch 67/100
83/83 [==============================] - 3s 38ms/step - loss: 0.0768 -
accuracy: 0.9784 - val_loss: 0.2899 - val_accuracy: 0.9320
Epoch 68/100
83/83 [==============================] - 3s 38ms/step - loss: 0.0381 -
accuracy: 0.9871 - val_loss: 0.3958 - val_accuracy: 0.9422
Epoch 69/100
```

```
83/83 [==============================] - 3s 38ms/step - loss: 0.0384 -
accuracy: 0.9917 - val_loss: 0.2992 - val_accuracy: 0.9388
Epoch 70/100
83/83 [==============================] - 3s 39ms/step - loss: 0.0224 -
accuracy: 0.9936 - val_loss: 0.4049 - val_accuracy: 0.9184
Epoch 71/100
83/83 [==============================] - 3s 39ms/step - loss: 0.0117 -
accuracy: 0.9966 - val_loss: 0.3356 - val_accuracy: 0.9456
Epoch 72/100
83/83 [==============================] - 3s 38ms/step - loss: 0.0277 -
accuracy: 0.9928 - val_loss: 0.3606 - val_accuracy: 0.9286
Epoch 73/100
83/83 [==============================] - 3s 39ms/step - loss: 0.0199 -
accuracy: 0.9928 - val_loss: 0.4271 - val_accuracy: 0.9218
Epoch 74/100
83/83 [==============================] - 3s 39ms/step - loss: 0.0388 -
accuracy: 0.9909 - val_loss: 0.3344 - val_accuracy: 0.9354
Epoch 75/100
83/83 [==============================] - 3s 39ms/step - loss: 0.0160 -
accuracy: 0.9932 - val_loss: 0.3860 - val_accuracy: 0.9388
Epoch 76/100
83/83 [==============================] - 3s 38ms/step - loss: 0.0298 -
accuracy: 0.9913 - val_loss: 0.3756 - val_accuracy: 0.9286
Epoch 77/100
83/83 [==============================] - 3s 39ms/step - loss: 0.0639 -
accuracy: 0.9837 - val_loss: 0.3115 - val_accuracy: 0.9252
Epoch 78/100
83/83 [==============================] - 3s 38ms/step - loss: 0.0424 -
accuracy: 0.9871 - val_loss: 0.2974 - val_accuracy: 0.9252
Epoch 79/100
83/83 [==============================] - 3s 39ms/step - loss: 0.0429 -
accuracy: 0.9894 - val_loss: 0.3952 - val_accuracy: 0.9252
Epoch 80/100
83/83 [==============================] - 3s 38ms/step - loss: 0.0291 -
accuracy: 0.9902 - val_loss: 0.5261 - val_accuracy: 0.9116
Epoch 81/100
83/83 [==============================] - 3s 39ms/step - loss: 0.0297 -
accuracy: 0.9886 - val_loss: 0.4262 - val_accuracy: 0.9252
Epoch 82/100
83/83 [==============================] - 3s 38ms/step - loss: 0.0346 -
accuracy: 0.9894 - val_loss: 0.5354 - val_accuracy: 0.9286
Epoch 83/100
83/83 [==============================] - 3s 39ms/step - loss: 0.0530 -
accuracy: 0.9864 - val_loss: 0.3691 - val_accuracy: 0.9184
Epoch 84/100
83/83 [==============================] - 3s 38ms/step - loss: 0.0209 -
accuracy: 0.9924 - val_loss: 0.4060 - val_accuracy: 0.9150
Epoch 85/100
83/83 [==============================] - 3s 38ms/step - loss: 0.0323 -
```

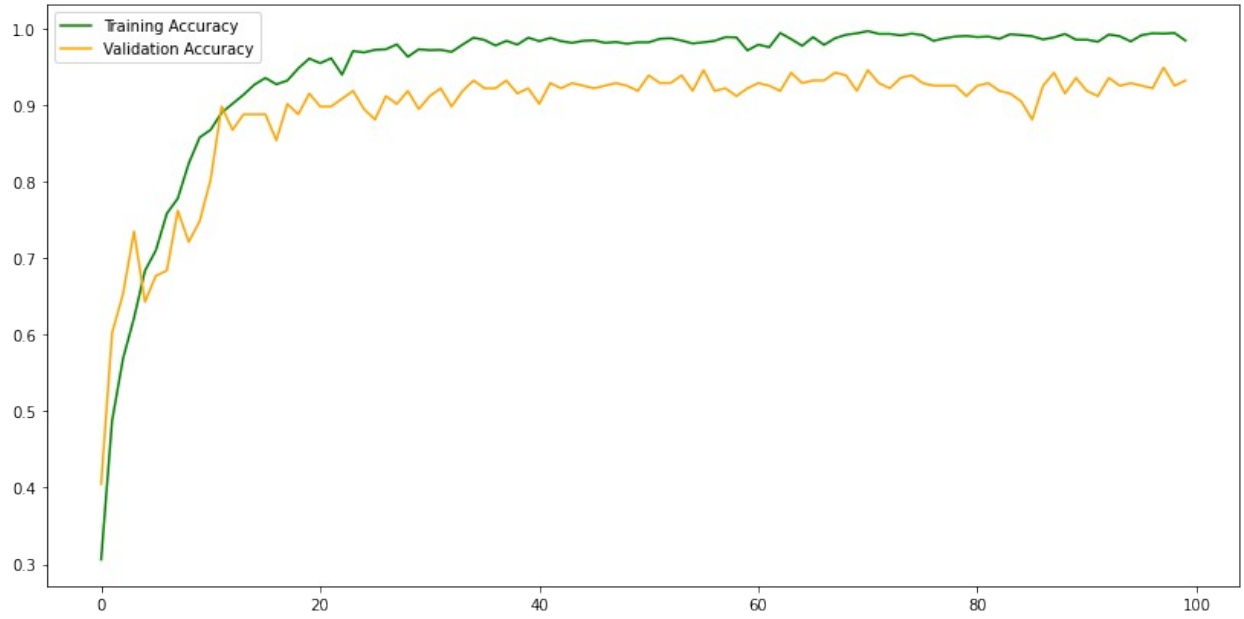
```
accuracy: 0.9913 - val_loss: 0.5798 - val_accuracy: 0.9048
Epoch 86/100
83/83 [==============================] - 3s 38ms/step - loss: 0.0448 -
accuracy: 0.9898 - val_loss: 0.5999 - val_accuracy: 0.8810
Epoch 87/100
83/83 [==============================] - 3s 38ms/step - loss: 0.0589 -
accuracy: 0.9856 - val_loss: 0.3932 - val_accuracy: 0.9252
Epoch 88/100
83/83 [==============================] - 3s 38ms/step - loss: 0.0375 -
accuracy: 0.9883 - val_loss: 0.3186 - val_accuracy: 0.9422
Epoch 89/100
83/83 [==============================] - 3s 38ms/step - loss: 0.0203 -
accuracy: 0.9928 - val_loss: 0.4508 - val_accuracy: 0.9150
Epoch 90/100
83/83 [==============================] - 3s 38ms/step - loss: 0.0486 -
accuracy: 0.9852 - val_loss: 0.3471 - val_accuracy: 0.9354
Epoch 91/100
83/83 [==============================] - 3s 39ms/step - loss: 0.0507 -
accuracy: 0.9852 - val_loss: 0.4348 - val_accuracy: 0.9184
Epoch 92/100
83/83 [==============================] - 3s 38ms/step - loss: 0.0592 -
accuracy: 0.9826 - val_loss: 0.3795 - val_accuracy: 0.9116
Epoch 93/100
83/83 [==============================] - 3s 38ms/step - loss: 0.0313 -
accuracy: 0.9921 - val_loss: 0.4590 - val_accuracy: 0.9354
Epoch 94/100
83/83 [==============================] - 3s 39ms/step - loss: 0.0267 -
accuracy: 0.9898 - val_loss: 0.5112 - val_accuracy: 0.9252
Epoch 95/100
83/83 [==============================] - 3s 39ms/step - loss: 0.0520 -
accuracy: 0.9830 - val_loss: 0.3644 - val_accuracy: 0.9286
Epoch 96/100
83/83 [==============================] - 3s 38ms/step - loss: 0.0368 -
accuracy: 0.9913 - val_loss: 0.4162 - val_accuracy: 0.9252
Epoch 97/100
83/83 [==============================] - 3s 38ms/step - loss: 0.0270 -
accuracy: 0.9936 - val_loss: 0.3861 - val_accuracy: 0.9218
Epoch 98/100
83/83 [==============================] - 3s 39ms/step - loss: 0.0198 -
accuracy: 0.9932 - val_loss: 0.4225 - val_accuracy: 0.9490
Epoch 99/100
83/83 [==============================] - 3s 38ms/step - loss: 0.0238 -
accuracy: 0.9939 - val_loss: 0.4300 - val_accuracy: 0.9252
Epoch 100/100
83/83 [==============================] - 3s 39ms/step - loss: 0.0534 -
accuracy: 0.9841 - val_loss: 0.4285 - val_accuracy: 0.9320
```

Visualisations

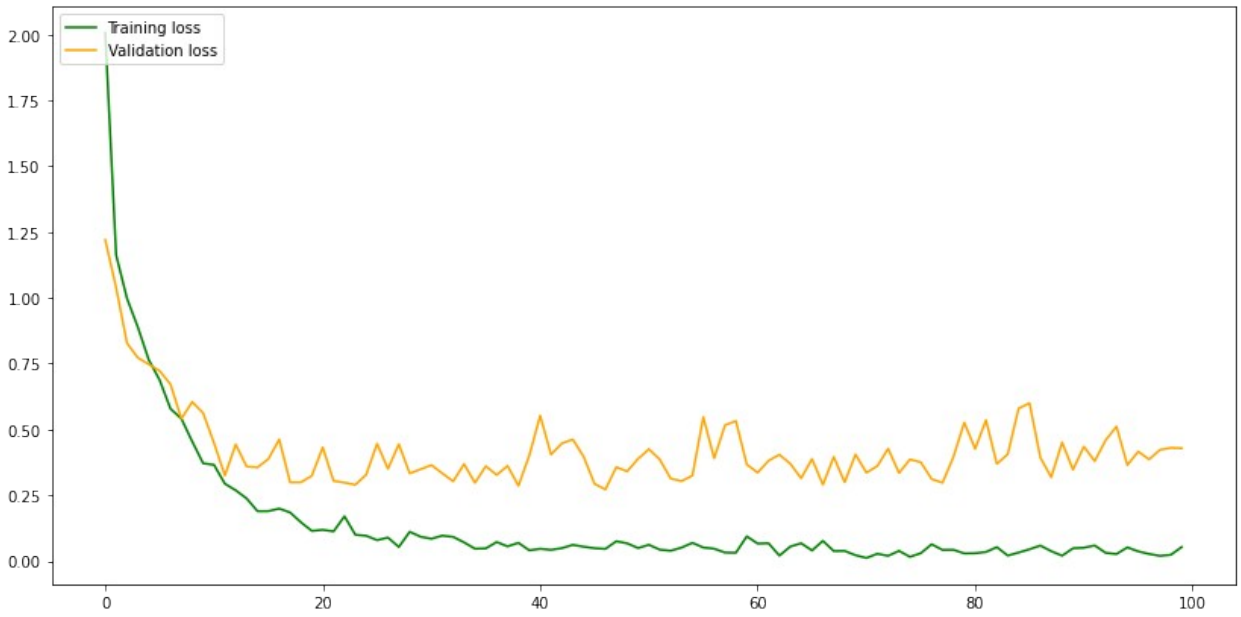
```
import matplotlib.pyplot as plt
import seaborn as sns

# model.save('braintumor.h5')

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
epochs = range(len(acc))
fig = plt.figure(figsize=(14,7))
plt.plot(epochs, acc, 'g', label="Training Accuracy")  # Green color
for training accuracy
plt.plot(epochs, val_acc, 'orange', label="Validation Accuracy")  #
Orange color for validation accuracy
plt.legend(loc='upper left')
plt.show()
```




```
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(len(loss))
fig = plt.figure(figsize=(14,7))
plt.plot(epochs,loss,'g',label="Training loss")
plt.plot(epochs,val_loss,'orange',label="Validation loss")
plt.legend(loc='upper left')
plt.show()
```



Prediction

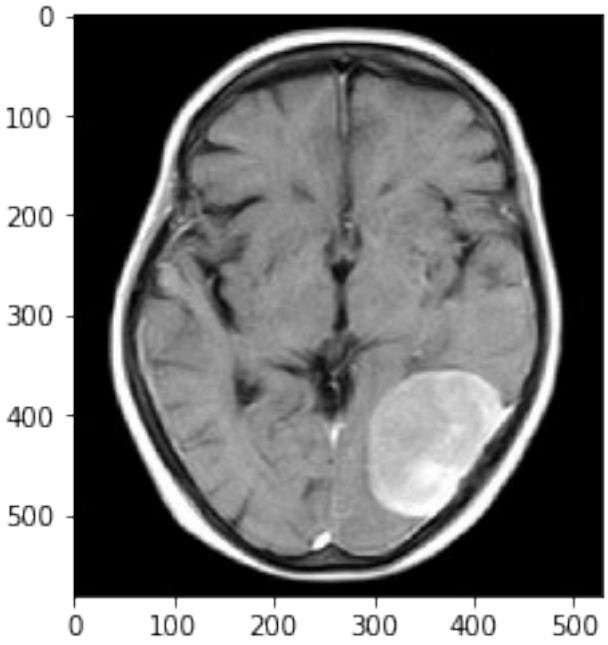
```
img =
cv2.imread('/kaggle/input/brain-tumor-classification-mri/Testing/menin
gioma_tumor/image(106).jpg')
img = cv2.resize(img,(150,150))
img_array = np.array(img)
img_array.shape
```

```
(150, 150, 3)
```

```
img_array = img_array.reshape(1,150,150,3)
img_array.shape
```

```
(1, 150, 150, 3)
```

```
from tensorflow.keras.preprocessing import image
img =
image.load_img('/kaggle/input/brain-tumor-classification-mri/Testing/
meningioma_tumor/image(106).jpg')
plt.imshow(img,interpolation='nearest')
plt.show()
```




```
a=model.predict(img_array)
indices = a.argmax()
indices
```

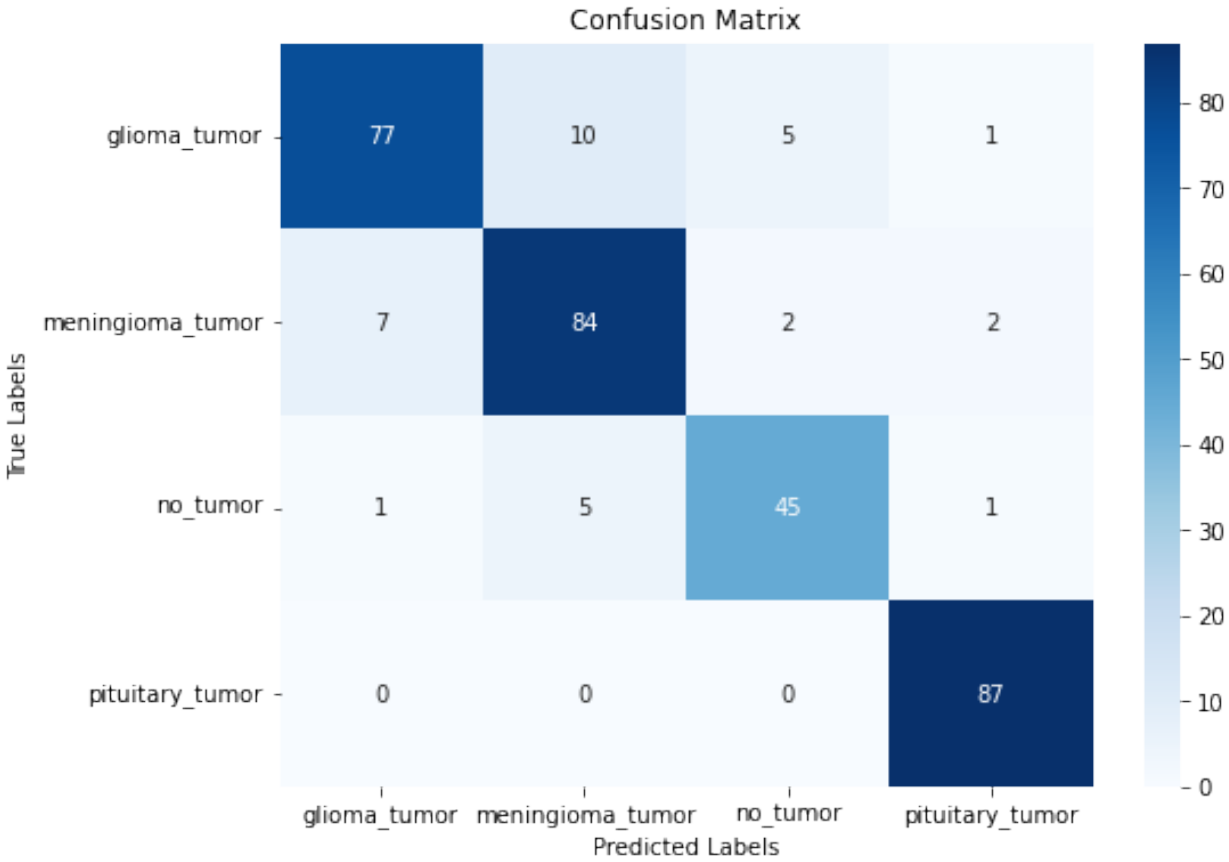
```
1
```

Confusion Matrix

```
from sklearn.metrics import confusion_matrix
conf_matrix = confusion_matrix(true_labels, pred_labels)

labels = ['glioma_tumor', 'meningioma_tumor', 'no_tumor',
'pituitary_tumor']


plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, cmap='Blues', fmt='d',
xticklabels=labels, yticklabels=labels)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()
```


Confusion Matrix
glioma_tumor
meningioma_tumor
no_tumor
pituitary_tumor
True Labels
77
10
5
1
7
84
2
2
1
5
45
1
0
0
0
87
glioma_tumor
meningioma_tumor
no_tumor
pituitary_tumor
Predicted Labels
80
70
60
50
40
30
20
10
0