brain-tumor-mobilenet

November 11, 2023

[1]:
```
from tensorflow.compat.v1 import ConfigProto
from tensorflow.compat.v1 import InteractiveSession
config = ConfigProto()
config.gpu_options.per_process_gpu_memory_fraction = 0.5
config.gpu_options.allow_growth = True
session = InteractiveSession(config=config)
```

```
/opt/conda/lib/python3.10/site-packages/scipy/__init__.py:146: UserWarning: A
NumPy version >=1.16.5 and <1.23.0 is required for this version of SciPy
(detected version 1.24.3
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

[2]:
```
from tensorflow.keras.layers import Input, Lambda, Dense, Flatten
from tensorflow.keras.models import Model
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator,load_img
from tensorflow.keras.models import Sequential
import pandas as pd
import seaborn as sn
import tensorflow as tf
import numpy as np
from glob import glob
import matplotlib.pyplot as plt
import os

from tensorflow.keras.optimizers import Adam
from tensorflow.keras.applications import MobileNet
from tensorflow.keras.applications.mobilenet import preprocess_input,␣
 ↪decode_predictions
```

[3]:
```
img_SIZE = [224, 224]

train_path = '/kaggle/input/brain-tumor/Training'
test_path = '/kaggle/input/brain-tumor/Testing'
```

[4]:
```
mobileNet = MobileNet(input_shape=img_SIZE + [3], weights='imagenet',␣
 ↪include_top=False)
```

[5]:
```
for layer in mobileNet.layers:
    layer.trainable = False
```

[6]:
```
folders = glob('/kaggle/input/brain-tumor/Training/*')
```

[7]:
```
x = Flatten()(mobileNet.output)
x = Dense(1024,activation='relu')(x)
```

[8]:
```
prediction = Dense(len(folders), activation='softmax')(x)
model = Model(inputs=mobileNet.input, outputs=prediction)
```

[9]:
```
model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 conv1 (Conv2D)              (None, 112, 112, 32)      864

 conv1_bn (BatchNormalizati  (None, 112, 112, 32)      128
 on)

 conv1_relu (ReLU)           (None, 112, 112, 32)      0

 conv_dw_1 (DepthwiseConv2D  (None, 112, 112, 32)      288
 )

 conv_dw_1_bn (BatchNormali  (None, 112, 112, 32)      128
 zation)

 conv_dw_1_relu (ReLU)       (None, 112, 112, 32)      0

 conv_pw_1 (Conv2D)          (None, 112, 112, 64)      2048

 conv_pw_1_bn (BatchNormali  (None, 112, 112, 64)      256
 zation)

 conv_pw_1_relu (ReLU)       (None, 112, 112, 64)      0

 conv_pad_2 (ZeroPadding2D)  (None, 113, 113, 64)      0

 conv_dw_2 (DepthwiseConv2D  (None, 56, 56, 64)        576
 )

 conv_dw_2_bn (BatchNormali  (None, 56, 56, 64)        256
```

```
zation)

conv_dw_2_relu (ReLU)       (None, 56, 56, 64)        0

conv_pw_2 (Conv2D)          (None, 56, 56, 128)       8192

conv_pw_2_bn (BatchNormali  (None, 56, 56, 128)       512
zation)

conv_pw_2_relu (ReLU)       (None, 56, 56, 128)       0

conv_dw_3 (DepthwiseConv2D  (None, 56, 56, 128)       1152
)

conv_dw_3_bn (BatchNormali  (None, 56, 56, 128)       512
zation)

conv_dw_3_relu (ReLU)       (None, 56, 56, 128)       0

conv_pw_3 (Conv2D)          (None, 56, 56, 128)       16384

conv_pw_3_bn (BatchNormali  (None, 56, 56, 128)       512
zation)

conv_pw_3_relu (ReLU)       (None, 56, 56, 128)       0

conv_pad_4 (ZeroPadding2D)  (None, 57, 57, 128)       0

conv_dw_4 (DepthwiseConv2D  (None, 28, 28, 128)       1152
)

conv_dw_4_bn (BatchNormali  (None, 28, 28, 128)       512
zation)

conv_dw_4_relu (ReLU)       (None, 28, 28, 128)       0

conv_pw_4 (Conv2D)          (None, 28, 28, 256)       32768

conv_pw_4_bn (BatchNormali  (None, 28, 28, 256)       1024
zation)

conv_pw_4_relu (ReLU)       (None, 28, 28, 256)       0

conv_dw_5 (DepthwiseConv2D  (None, 28, 28, 256)       2304
)

conv_dw_5_bn (BatchNormali  (None, 28, 28, 256)       1024
zation)
```

```
conv_dw_5_relu (ReLU)       (None, 28, 28, 256)       0

conv_pw_5 (Conv2D)          (None, 28, 28, 256)       65536

conv_pw_5_bn (BatchNormali  (None, 28, 28, 256)       1024
zation)

conv_pw_5_relu (ReLU)       (None, 28, 28, 256)       0

conv_pad_6 (ZeroPadding2D)  (None, 29, 29, 256)       0

conv_dw_6 (DepthwiseConv2D  (None, 14, 14, 256)       2304
)

conv_dw_6_bn (BatchNormali  (None, 14, 14, 256)       1024
zation)

conv_dw_6_relu (ReLU)       (None, 14, 14, 256)       0

conv_pw_6 (Conv2D)          (None, 14, 14, 512)       131072

conv_pw_6_bn (BatchNormali  (None, 14, 14, 512)       2048
zation)

conv_pw_6_relu (ReLU)       (None, 14, 14, 512)       0

conv_dw_7 (DepthwiseConv2D  (None, 14, 14, 512)       4608
)

conv_dw_7_bn (BatchNormali  (None, 14, 14, 512)       2048
zation)

conv_dw_7_relu (ReLU)       (None, 14, 14, 512)       0

conv_pw_7 (Conv2D)          (None, 14, 14, 512)       262144

conv_pw_7_bn (BatchNormali  (None, 14, 14, 512)       2048
zation)

conv_pw_7_relu (ReLU)       (None, 14, 14, 512)       0

conv_dw_8 (DepthwiseConv2D  (None, 14, 14, 512)       4608
)

conv_dw_8_bn (BatchNormali  (None, 14, 14, 512)       2048
zation)
```

```
conv_dw_8_relu (ReLU)       (None, 14, 14, 512)       0

conv_pw_8 (Conv2D)          (None, 14, 14, 512)       262144

conv_pw_8_bn (BatchNormali  (None, 14, 14, 512)       2048
zation)

conv_pw_8_relu (ReLU)       (None, 14, 14, 512)       0

conv_dw_9 (DepthwiseConv2D  (None, 14, 14, 512)       4608
)

conv_dw_9_bn (BatchNormali  (None, 14, 14, 512)       2048
zation)

conv_dw_9_relu (ReLU)       (None, 14, 14, 512)       0

conv_pw_9 (Conv2D)          (None, 14, 14, 512)       262144

conv_pw_9_bn (BatchNormali  (None, 14, 14, 512)       2048
zation)

conv_pw_9_relu (ReLU)       (None, 14, 14, 512)       0

conv_dw_10 (DepthwiseConv2  (None, 14, 14, 512)       4608
D)

conv_dw_10_bn (BatchNormal  (None, 14, 14, 512)       2048
ization)

conv_dw_10_relu (ReLU)      (None, 14, 14, 512)       0

conv_pw_10 (Conv2D)         (None, 14, 14, 512)       262144

conv_pw_10_bn (BatchNormal  (None, 14, 14, 512)       2048
ization)

conv_pw_10_relu (ReLU)      (None, 14, 14, 512)       0

conv_dw_11 (DepthwiseConv2  (None, 14, 14, 512)       4608
D)

conv_dw_11_bn (BatchNormal  (None, 14, 14, 512)       2048
ization)

conv_dw_11_relu (ReLU)      (None, 14, 14, 512)       0

conv_pw_11 (Conv2D)         (None, 14, 14, 512)       262144
```

```
conv_pw_11_bn (BatchNormal  (None, 14, 14, 512)       2048
ization)

conv_pw_11_relu (ReLU)      (None, 14, 14, 512)       0

conv_pad_12 (ZeroPadding2D  (None, 15, 15, 512)       0
)

conv_dw_12 (DepthwiseConv2  (None, 7, 7, 512)         4608
D)

conv_dw_12_bn (BatchNormal  (None, 7, 7, 512)         2048
ization)

conv_dw_12_relu (ReLU)      (None, 7, 7, 512)         0

conv_pw_12 (Conv2D)         (None, 7, 7, 1024)        524288

conv_pw_12_bn (BatchNormal  (None, 7, 7, 1024)        4096
ization)

conv_pw_12_relu (ReLU)      (None, 7, 7, 1024)        0

conv_dw_13 (DepthwiseConv2  (None, 7, 7, 1024)        9216
D)

conv_dw_13_bn (BatchNormal  (None, 7, 7, 1024)        4096
ization)

conv_dw_13_relu (ReLU)      (None, 7, 7, 1024)        0

conv_pw_13 (Conv2D)         (None, 7, 7, 1024)        1048576

conv_pw_13_bn (BatchNormal  (None, 7, 7, 1024)        4096
ization)

conv_pw_13_relu (ReLU)      (None, 7, 7, 1024)        0

flatten (Flatten)           (None, 50176)             0

dense (Dense)               (None, 1024)              51381248

dense_1 (Dense)             (None, 4)                 4100

=================================================================
Total params: 54614212 (208.34 MB)
Trainable params: 51385348 (196.02 MB)
```

```
Non-trainable params: 3228864 (12.32 MB)
_________________________________________________________________
```

[10]:
```
adam = Adam(lr=0.0001)
```

[11]:
```
model.compile(
  loss='categorical_crossentropy',
  optimizer= adam,
  metrics=['accuracy']
)
```

[12]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

train_dataset = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)

test_dataset = ImageDataGenerator(rescale = 1./255)
```

[13]:
```
training_set = train_dataset.flow_from_directory('/kaggle/input/brain-tumor/
 ↪Training',
                                                 target_size = (224, 224),
                                                 batch_size = 64,
                                                 class_mode = 'categorical')

test_set = test_dataset.flow_from_directory('/kaggle/input/brain-tumor/Testing',
                                            target_size = (224, 224),
                                            batch_size = 1,
                                            class_mode = 'categorical')
```

```
Found 2870 images belonging to 4 classes.
Found 394 images belonging to 4 classes.
```

[14]:
```
r = model.fit(
  training_set,
  validation_data=test_set,
  epochs=50,
  steps_per_epoch=len(training_set),
  validation_steps=len(test_set)
)
```

```
Epoch 1/50
45/45 [==============================] - 40s 790ms/step - loss: 13.0573 -
accuracy: 0.7108 - val_loss: 5.3915 - val_accuracy: 0.6396
Epoch 2/50
45/45 [==============================] - 37s 815ms/step - loss: 0.6759 -
accuracy: 0.8983 - val_loss: 3.1965 - val_accuracy: 0.7614
```

```
Epoch 3/50
45/45 [==============================] - 36s 799ms/step - loss: 0.3746 -
accuracy: 0.9338 - val_loss: 2.9937 - val_accuracy: 0.7640
Epoch 4/50
45/45 [==============================] - 37s 826ms/step - loss: 0.3499 -
accuracy: 0.9251 - val_loss: 2.7141 - val_accuracy: 0.8020
Epoch 5/50
45/45 [==============================] - 37s 830ms/step - loss: 0.2294 -
accuracy: 0.9488 - val_loss: 4.0656 - val_accuracy: 0.7513
Epoch 6/50
45/45 [==============================] - 37s 828ms/step - loss: 0.1189 -
accuracy: 0.9707 - val_loss: 2.7265 - val_accuracy: 0.8020
Epoch 7/50
45/45 [==============================] - 36s 805ms/step - loss: 0.0832 -
accuracy: 0.9770 - val_loss: 3.5139 - val_accuracy: 0.7614
Epoch 8/50
45/45 [==============================] - 36s 807ms/step - loss: 0.1135 -
accuracy: 0.9707 - val_loss: 3.1002 - val_accuracy: 0.7944
Epoch 9/50
45/45 [==============================] - 36s 797ms/step - loss: 0.0782 -
accuracy: 0.9770 - val_loss: 3.3375 - val_accuracy: 0.8020
Epoch 10/50
45/45 [==============================] - 36s 794ms/step - loss: 0.0685 -
accuracy: 0.9808 - val_loss: 3.2081 - val_accuracy: 0.8020
Epoch 11/50
45/45 [==============================] - 36s 809ms/step - loss: 0.0663 -
accuracy: 0.9819 - val_loss: 2.7338 - val_accuracy: 0.8020
Epoch 12/50
45/45 [==============================] - 36s 811ms/step - loss: 0.0450 -
accuracy: 0.9861 - val_loss: 2.7988 - val_accuracy: 0.8071
Epoch 13/50
45/45 [==============================] - 36s 804ms/step - loss: 0.0236 -
accuracy: 0.9930 - val_loss: 3.0318 - val_accuracy: 0.8020
Epoch 14/50
45/45 [==============================] - 36s 802ms/step - loss: 0.0339 -
accuracy: 0.9882 - val_loss: 3.5835 - val_accuracy: 0.7817
Epoch 15/50
45/45 [==============================] - 36s 804ms/step - loss: 0.0494 -
accuracy: 0.9868 - val_loss: 2.4135 - val_accuracy: 0.8020
Epoch 16/50
45/45 [==============================] - 36s 808ms/step - loss: 0.0527 -
accuracy: 0.9843 - val_loss: 3.9843 - val_accuracy: 0.7513
Epoch 17/50
45/45 [==============================] - 36s 807ms/step - loss: 0.0849 -
accuracy: 0.9756 - val_loss: 3.1656 - val_accuracy: 0.7944
Epoch 18/50
45/45 [==============================] - 36s 801ms/step - loss: 0.0849 -
accuracy: 0.9780 - val_loss: 3.5992 - val_accuracy: 0.7970
```

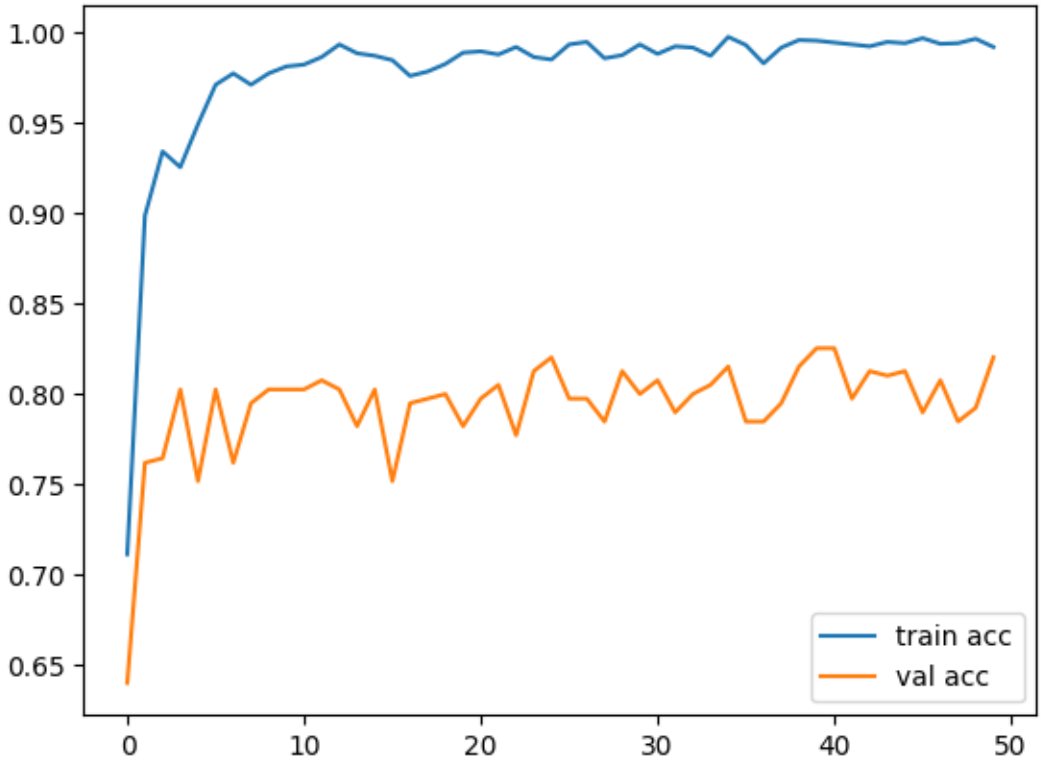
```
Epoch 19/50
45/45 [==============================] - 36s 802ms/step - loss: 0.0732 -
accuracy: 0.9822 - val_loss: 3.1542 - val_accuracy: 0.7995
Epoch 20/50
45/45 [==============================] - 36s 795ms/step - loss: 0.0417 -
accuracy: 0.9885 - val_loss: 4.2527 - val_accuracy: 0.7817
Epoch 21/50
45/45 [==============================] - 37s 812ms/step - loss: 0.0377 -
accuracy: 0.9892 - val_loss: 4.1330 - val_accuracy: 0.7970
Epoch 22/50
45/45 [==============================] - 36s 804ms/step - loss: 0.0513 -
accuracy: 0.9875 - val_loss: 3.5606 - val_accuracy: 0.8046
Epoch 23/50
45/45 [==============================] - 36s 803ms/step - loss: 0.0335 -
accuracy: 0.9916 - val_loss: 4.0703 - val_accuracy: 0.7766
Epoch 24/50
45/45 [==============================] - 36s 799ms/step - loss: 0.0476 -
accuracy: 0.9861 - val_loss: 2.7960 - val_accuracy: 0.8122
Epoch 25/50
45/45 [==============================] - 36s 808ms/step - loss: 0.0600 -
accuracy: 0.9847 - val_loss: 2.9812 - val_accuracy: 0.8198
Epoch 26/50
45/45 [==============================] - 36s 800ms/step - loss: 0.0161 -
accuracy: 0.9930 - val_loss: 3.6994 - val_accuracy: 0.7970
Epoch 27/50
45/45 [==============================] - 36s 797ms/step - loss: 0.0211 -
accuracy: 0.9944 - val_loss: 3.5189 - val_accuracy: 0.7970
Epoch 28/50
45/45 [==============================] - 36s 793ms/step - loss: 0.0462 -
accuracy: 0.9854 - val_loss: 3.7417 - val_accuracy: 0.7843
Epoch 29/50
45/45 [==============================] - 36s 800ms/step - loss: 0.0402 -
accuracy: 0.9871 - val_loss: 3.1270 - val_accuracy: 0.8122
Epoch 30/50
45/45 [==============================] - 36s 796ms/step - loss: 0.0270 -
accuracy: 0.9930 - val_loss: 2.8882 - val_accuracy: 0.7995
Epoch 31/50
45/45 [==============================] - 37s 823ms/step - loss: 0.0406 -
accuracy: 0.9878 - val_loss: 3.1953 - val_accuracy: 0.8071
Epoch 32/50
45/45 [==============================] - 36s 802ms/step - loss: 0.0251 -
accuracy: 0.9920 - val_loss: 3.4539 - val_accuracy: 0.7893
Epoch 33/50
45/45 [==============================] - 36s 801ms/step - loss: 0.0284 -
accuracy: 0.9913 - val_loss: 3.1459 - val_accuracy: 0.7995
Epoch 34/50
45/45 [==============================] - 36s 800ms/step - loss: 0.0493 -
accuracy: 0.9868 - val_loss: 3.5981 - val_accuracy: 0.8046
```

```
Epoch 35/50
45/45 [==============================] - 36s 810ms/step - loss: 0.0154 -
accuracy: 0.9972 - val_loss: 3.3497 - val_accuracy: 0.8147
Epoch 36/50
45/45 [==============================] - 36s 803ms/step - loss: 0.0311 -
accuracy: 0.9927 - val_loss: 4.0916 - val_accuracy: 0.7843
Epoch 37/50
45/45 [==============================] - 36s 800ms/step - loss: 0.0686 -
accuracy: 0.9826 - val_loss: 4.4307 - val_accuracy: 0.7843
Epoch 38/50
45/45 [==============================] - 36s 800ms/step - loss: 0.0294 -
accuracy: 0.9913 - val_loss: 3.8135 - val_accuracy: 0.7944
Epoch 39/50
45/45 [==============================] - 36s 800ms/step - loss: 0.0128 -
accuracy: 0.9955 - val_loss: 3.2756 - val_accuracy: 0.8147
Epoch 40/50
45/45 [==============================] - 36s 799ms/step - loss: 0.0187 -
accuracy: 0.9951 - val_loss: 2.9785 - val_accuracy: 0.8249
Epoch 41/50
45/45 [==============================] - 36s 792ms/step - loss: 0.0175 -
accuracy: 0.9941 - val_loss: 2.9954 - val_accuracy: 0.8249
Epoch 42/50
45/45 [==============================] - 36s 797ms/step - loss: 0.0250 -
accuracy: 0.9930 - val_loss: 3.8773 - val_accuracy: 0.7970
Epoch 43/50
45/45 [==============================] - 36s 792ms/step - loss: 0.0364 -
accuracy: 0.9920 - val_loss: 3.5567 - val_accuracy: 0.8122
Epoch 44/50
45/45 [==============================] - 36s 801ms/step - loss: 0.0172 -
accuracy: 0.9944 - val_loss: 3.3339 - val_accuracy: 0.8096
Epoch 45/50
45/45 [==============================] - 36s 800ms/step - loss: 0.0155 -
accuracy: 0.9937 - val_loss: 3.2495 - val_accuracy: 0.8122
Epoch 46/50
45/45 [==============================] - 37s 815ms/step - loss: 0.0130 -
accuracy: 0.9965 - val_loss: 3.8859 - val_accuracy: 0.7893
Epoch 47/50
45/45 [==============================] - 36s 801ms/step - loss: 0.0216 -
accuracy: 0.9934 - val_loss: 3.3370 - val_accuracy: 0.8071
Epoch 48/50
45/45 [==============================] - 36s 802ms/step - loss: 0.0191 -
accuracy: 0.9937 - val_loss: 3.9226 - val_accuracy: 0.7843
Epoch 49/50
45/45 [==============================] - 36s 800ms/step - loss: 0.0171 -
accuracy: 0.9962 - val_loss: 3.7045 - val_accuracy: 0.7919
Epoch 50/50
45/45 [==============================] - 36s 798ms/step - loss: 0.0283 -
accuracy: 0.9916 - val_loss: 2.8616 - val_accuracy: 0.8198
```

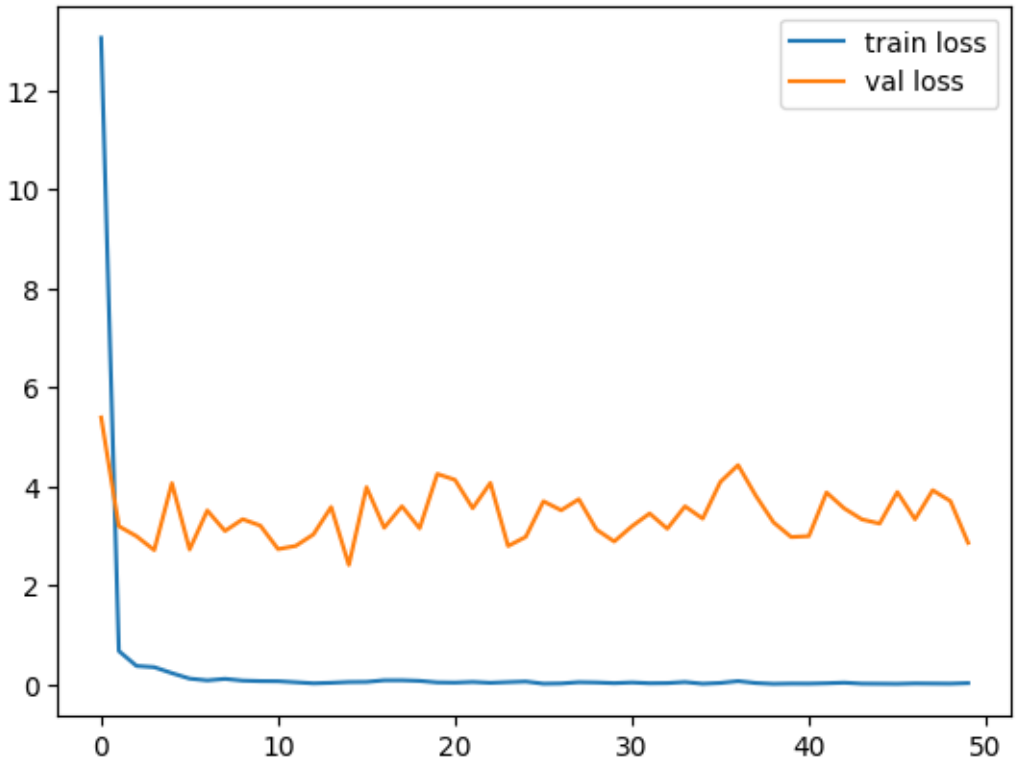
[15]:
```
test_loss,test_acc = model.evaluate(test_set, verbose=2)
print('\nTest accuracy:', test_acc)
```

```
394/394 - 2s - loss: 2.8616 - accuracy: 0.8198 - 2s/epoch - 5ms/step

Test accuracy: 0.8197969794273376
```

[16]:
```
plt.plot(r.history['accuracy'], label='train acc')
plt.plot(r.history['val_accuracy'], label='val acc')
plt.legend()
plt.show()
```



[17]:
```
plt.plot(r.history['loss'], label='train loss')
plt.plot(r.history['val_loss'], label='val loss')
plt.legend()
plt.show()
```



[18]:
```
from tensorflow.keras.models import load_model

model.save('mobileNet.keras')
```

1 Manually Check

[55]:
```
from tensorflow.keras.preprocessing import image
import numpy as np
import cv2
```

[56]:
```
from tensorflow.keras.models import load_model
model = load_model('/kaggle/working/mobileNet.keras')
```

[57]:
```
dataset_path = '/kaggle/input/brain-tumor/Training'
class_folders = os.listdir(dataset_path)
for class_folder in class_folders:
    print(f'Class: {class_folder}')
```

```
Class: no_tumor
Class: pituitary_tumor
Class: meningioma_tumor
```

```
Class: glioma_tumor
```

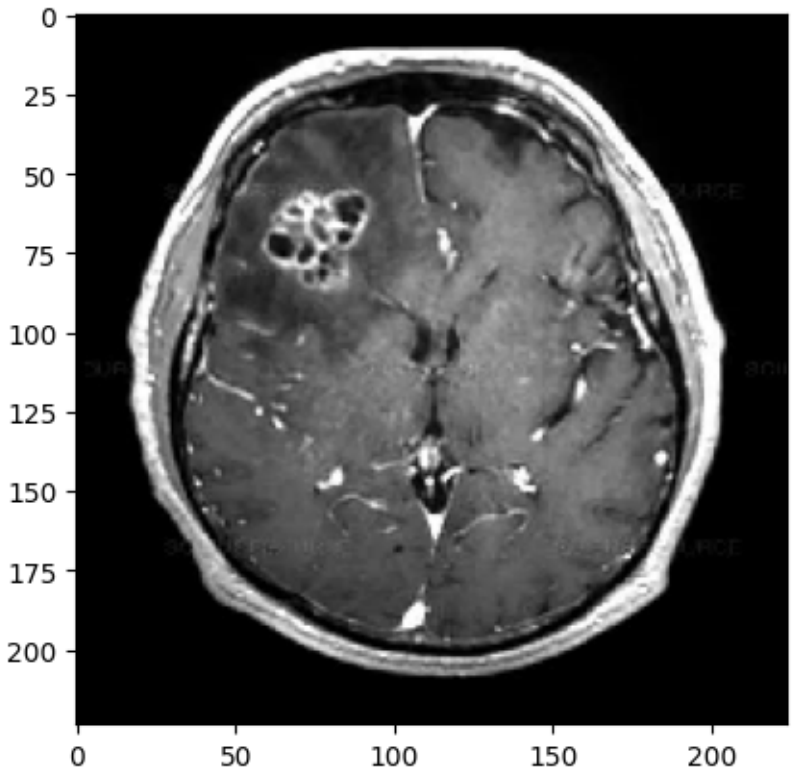
[58]:
```
from IPython.display import display, Markdown
```

[59]:
```
imagePath = "/kaggle/input/brain-tumor/Testing/glioma_tumor/image(21).jpg"

img = image.load_img(imagePath, target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

predictions = model.predict(x)

class_labels = ['no_tumor', 'pituitary_tumor', 'meningioma_tumor',␣
 ↪'glioma_tumor']
predicted_class_index = np.argmax(predictions)
predicted_class_label = class_labels[predicted_class_index]
confidence = predictions[0][predicted_class_index]

markdown_text = f"<h3 style='color:green;'>Predicted class:␣
 ↪{predicted_class_label}</h3>"
display(Markdown(markdown_text))

markdown_text2 = f"<h3 style='color:blue;'>Confidence: {confidence*100:.2f}%</
 ↪h3>"
display(Markdown(markdown_text2))

plt.imshow(img)
plt.show()
```

```
1/1 [==============================] - 0s 431ms/step
```

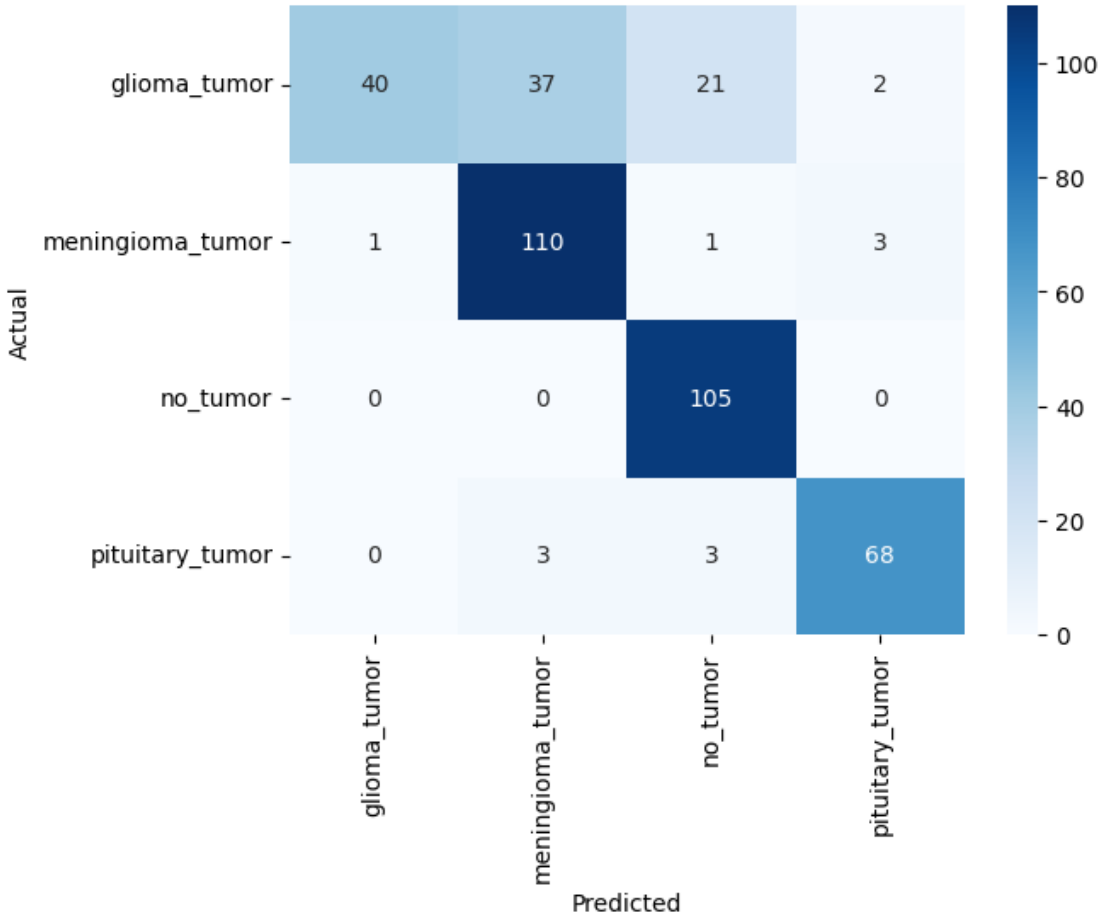
Predicted class: glioma_tumor

Confidence: 99.07%



[60]:
```
model = tf.keras.models.load_model("/kaggle/working/mobileNet.keras")
filenames = test_set.filenames
y_prob = []
y_act = []
test_set.reset()
nb_samples = len(test_set)

for _ in range(nb_samples):
    X_test, Y_test = test_set.next()
    y_prob.append(model.predict(X_test, verbose=0))  # Set verbose to 0 to␣
 ↪suppress progress bars
    y_act.append(Y_test)

predicted_class = [list(training_set.class_indices.keys())[i.argmax()] for i in␣
 ↪y_prob]
actual_class = [list(training_set.class_indices.keys())[i.argmax()] for i in␣
 ↪y_act]

out_df = pd.DataFrame(np.vstack([predicted_class, actual_class]).T,␣
 ↪columns=['predicted_class', 'actual_class'])
```

```
confusion_matrix = pd.crosstab(out_df['actual_class'],␣
 ↪out_df['predicted_class'], rownames=['Actual'], colnames=['Predicted'])

sn.heatmap(confusion_matrix, cmap='Blues', annot=True, fmt='d')
plt.show()

test_accuracy = (np.diagonal(confusion_matrix).sum() / confusion_matrix.sum().
 ↪sum() * 100)

markdown_text3 = f"<h3 style='color:green;'>Test Accuracy: {test_accuracy:.
 ↪2f}%</h3>"

display(Markdown(markdown_text3))
```



Test Accuracy: 81.98%