brain-tumor-mri-classification

March 9, 2024

[2]:
```
import os
from PIL import Image

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

[3]:
```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Conv2D,MaxPooling2D,Flatten,Dense,Dropout
from tensorflow.keras.optimizers import Adamax
```

```
2024-03-09 17:58:33.635513: E
external/local_xla/xla/stream_executor/cuda/cuda_dnn.cc:9261] Unable to register
cuDNN factory: Attempting to register factory for plugin cuDNN when one has
already been registered
2024-03-09 17:58:33.635648: E
external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:607] Unable to register
cuFFT factory: Attempting to register factory for plugin cuFFT when one has
already been registered
2024-03-09 17:58:33.752133: E
external/local_xla/xla/stream_executor/cuda/cuda_blas.cc:1515] Unable to
register cuBLAS factory: Attempting to register factory for plugin cuBLAS when
one has already been registered
```

[4]:
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix,classification_report
```

[5]:
```
# Generate data paths with labels
train_data_dir = '/kaggle/input/brain-tumor-mri-dataset/Training'
filepaths = []
labels = []

folds = os.listdir(train_data_dir)
```

```
# print(folds)

for fold in folds:
    foldpath = os.path.join(train_data_dir, fold)
    filelist = os.listdir(foldpath)
    for file in filelist:
        fpath = os.path.join(foldpath, file)

        filepaths.append(fpath)
        labels.append(fold)

# Concatenate data paths with labels into one dataframe
Fseries = pd.Series(filepaths, name= 'filepaths')
Lseries = pd.Series(labels, name='labels')

train_df = pd.concat([Fseries, Lseries], axis= 1)
```

[6]:
```
train_df
```

[6]:

	filepaths	labels
0	/kaggle/input/brain-tumor-mri-dataset/Training…	pituitary
1	/kaggle/input/brain-tumor-mri-dataset/Training…	pituitary
2	/kaggle/input/brain-tumor-mri-dataset/Training…	pituitary
3	/kaggle/input/brain-tumor-mri-dataset/Training…	pituitary
4	/kaggle/input/brain-tumor-mri-dataset/Training…	pituitary
…	…	…
5707	/kaggle/input/brain-tumor-mri-dataset/Training…	glioma
5708	/kaggle/input/brain-tumor-mri-dataset/Training…	glioma
5709	/kaggle/input/brain-tumor-mri-dataset/Training…	glioma
5710	/kaggle/input/brain-tumor-mri-dataset/Training…	glioma
5711	/kaggle/input/brain-tumor-mri-dataset/Training…	glioma

[5712 rows x 2 columns]

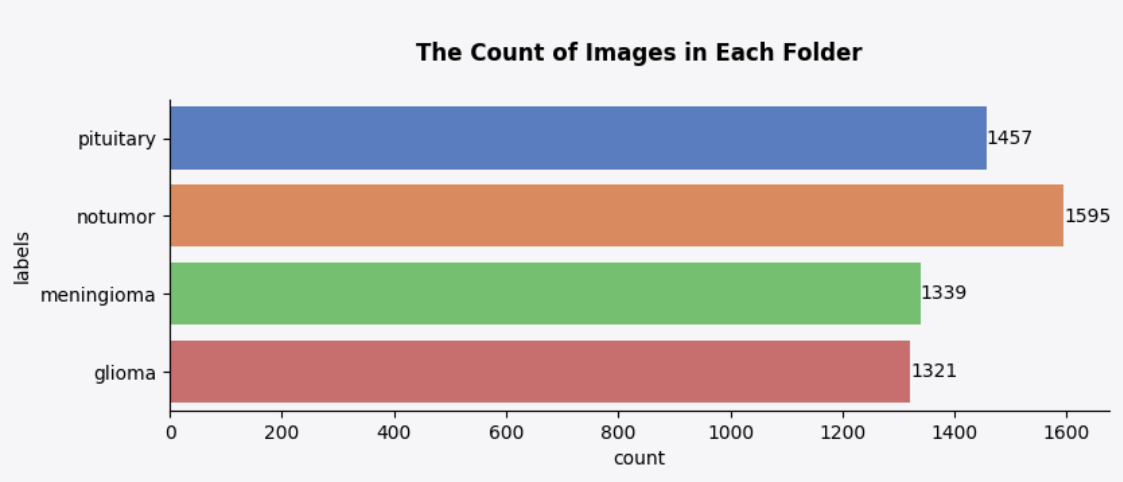
[7]:
```
test_data_dir = '/kaggle/input/brain-tumor-mri-dataset/Testing'
filepaths = []
labels = []

folds = os.listdir(test_data_dir)
for fold in folds:
    foldpath = os.path.join(test_data_dir, fold)
    filelist = os.listdir(foldpath)
    for file in filelist:
        fpath = os.path.join(foldpath, file)

        filepaths.append(fpath)
        labels.append(fold)
```

```
# Concatenate data paths with labels into one dataframe
Fseries = pd.Series(filepaths, name= 'filepaths')
Lseries = pd.Series(labels, name='labels')
ts_df = pd.concat([Fseries, Lseries], axis= 1)
```

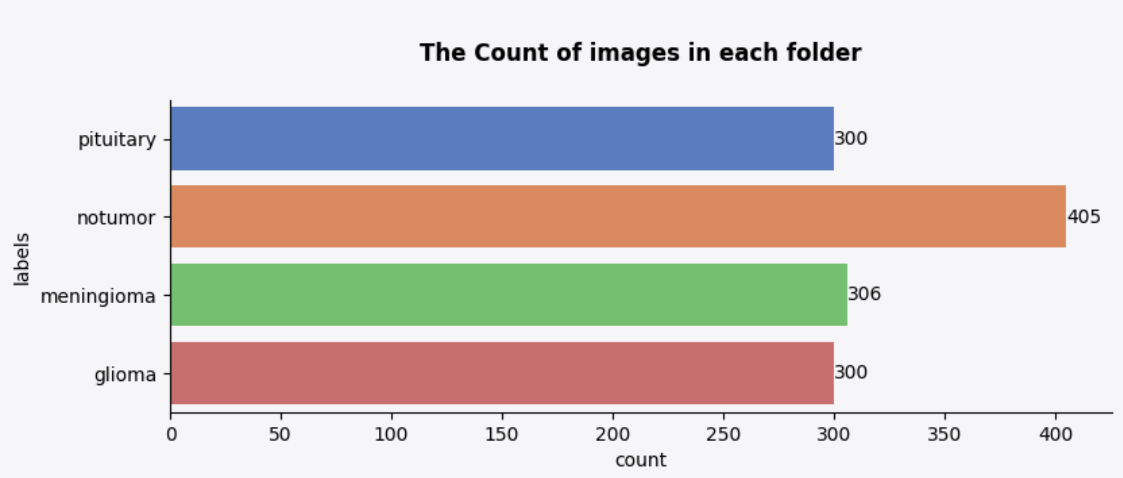
[8]:
```
ts_df
```

[8]:
```
                                              filepaths     labels
0     /kaggle/input/brain-tumor-mri-dataset/Testing/…  pituitary
1     /kaggle/input/brain-tumor-mri-dataset/Testing/…  pituitary
2     /kaggle/input/brain-tumor-mri-dataset/Testing/…  pituitary
3     /kaggle/input/brain-tumor-mri-dataset/Testing/…  pituitary
4     /kaggle/input/brain-tumor-mri-dataset/Testing/…  pituitary
…                                                   …         …
1306  /kaggle/input/brain-tumor-mri-dataset/Testing/…     glioma
1307  /kaggle/input/brain-tumor-mri-dataset/Testing/…     glioma
1308  /kaggle/input/brain-tumor-mri-dataset/Testing/…     glioma
1309  /kaggle/input/brain-tumor-mri-dataset/Testing/…     glioma
1310  /kaggle/input/brain-tumor-mri-dataset/Testing/…     glioma

[1311 rows x 2 columns]
```

[9]:
```
fig, ax = plt.subplots(figsize=(9, 3))
fig.patch.set_facecolor("#f6f5f7")
ax.set_facecolor("#f6f5f7")
for spine in ["top", "right"]:
    ax.spines[spine].set_visible(False)

x = sns.countplot(data=train_df, y=train_df["labels"], palette='muted')
for container in x.containers:
    x.bar_label(container)
plt.title("\nThe Count of Images in Each Folder\n", weight="bold")
plt.show()
```



[10]:
```
fig,ax=plt.subplots(figsize=(9,3))
fig.patch.set_facecolor("#f6f5f7")
ax.set_facecolor("#f6f5f7")
for i in ["right","top"]:
    ax.spines[i].set_visible(False)

i=sns.countplot(data=ts_df,y=ts_df["labels"],palette='muted')
for container in i.containers:
    i.bar_label(container)

plt.title("\nThe Count of images in each folder\n",weight="bold");
```



[11]:
```
ts_df.shape
```

[11]: (1311, 2)

0.1 Split dataframe into train, valid, and test

[12]:
```
# valid and test dataframe
valid_df, test_df = train_test_split(ts_df,  train_size= 0.5, shuffle= True,␣
 ↪random_state= 123)
```

0.2 Create image data generator

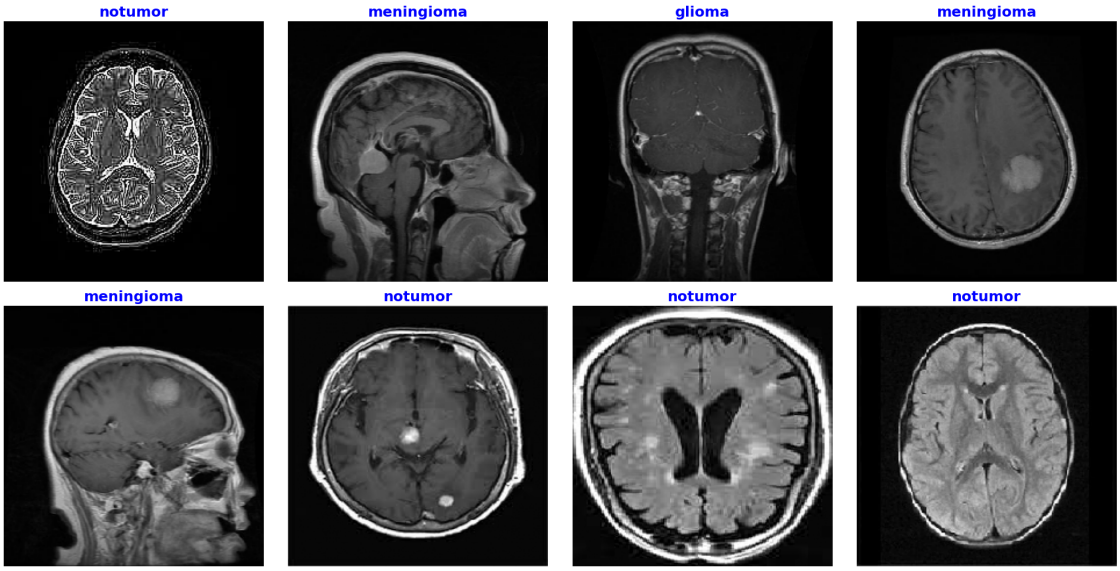
[13]:
```
batch_size = 16
img_size = (224, 224)

tr_gen = ImageDataGenerator()
ts_gen = ImageDataGenerator()

train_gen = tr_gen.flow_from_dataframe( train_df, x_col= 'filepaths', y_col=␣
 ↪'labels', target_size= img_size, class_mode= 'categorical',
                                    color_mode= 'rgb', shuffle= True,␣
 ↪batch_size= batch_size)

valid_gen = ts_gen.flow_from_dataframe( valid_df, x_col= 'filepaths', y_col=␣
 ↪'labels', target_size= img_size, class_mode= 'categorical',
                                    color_mode= 'rgb', shuffle= True,␣
 ↪batch_size= batch_size)

test_gen = ts_gen.flow_from_dataframe( test_df, x_col= 'filepaths', y_col=␣
 ↪'labels', target_size= img_size, class_mode= 'categorical',
                                    color_mode= 'rgb', shuffle= False,␣
 ↪batch_size= batch_size)
```

```
Found 5712 validated image filenames belonging to 4 classes.
Found 655 validated image filenames belonging to 4 classes.
Found 656 validated image filenames belonging to 4 classes.
```

0.3 Show sample from train data

[14]:
```
g_dict = train_gen.class_indices
classes = list(g_dict.keys())
images, labels = next(train_gen)


plt.figure(figsize= (20, 20))

for i in range(8):
    plt.subplot(4, 4, i + 1)
```

```
    image = images[i] / 255
    plt.imshow(image)
    index = np.argmax(labels[i])
    class_name = classes[index]
    plt.title(class_name, color= 'blue', fontsize= 18, weight="bold")
    plt.axis('off')
plt.tight_layout()
plt.show()
```



0.4 Building Deep Learning Model

[15]:
```
img_shape = (224, 224, 3)
class_count = len(list(train_gen.class_indices.keys()))
model2 = Sequential([
    Conv2D(filters=64, kernel_size=(3,3), padding="same", activation="relu",␣
 ↪input_shape= img_shape),
    Conv2D(filters=64, kernel_size=(3,3), padding="same", activation="relu"),
    MaxPooling2D((2, 2)),

    Conv2D(filters=128, kernel_size=(3,3), padding="same", activation="relu"),
    Conv2D(filters=128, kernel_size=(3,3), padding="same", activation="relu"),
    MaxPooling2D((2, 2)),

    Conv2D(filters=256, kernel_size=(3,3), padding="same", activation="relu"),
    Conv2D(filters=256, kernel_size=(3,3), padding="same", activation="relu"),
    Conv2D(filters=256, kernel_size=(3,3), padding="same", activation="relu"),
    MaxPooling2D((2, 2)),
```

```
    Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"),
    Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"),
    Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"),
    MaxPooling2D((2, 2)),

    Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"),
    Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"),
    Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"),
    MaxPooling2D((2, 2)),

    Flatten(),

    Dense(256,activation = "relu"),
    Dense(64,activation = "relu"),
    Dense(class_count, activation = "softmax")
])

model2.compile(Adamax(learning_rate= 0.001), loss= 'categorical_crossentropy',␣
 ↪metrics= ['accuracy'])

model2.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 224, 224, 64)	1,792
conv2d_1 (Conv2D)	(None, 224, 224, 64)	36,928
max_pooling2d (MaxPooling2D)	(None, 112, 112, 64)	0
conv2d_2 (Conv2D)	(None, 112, 112, 128)	73,856
conv2d_3 (Conv2D)	(None, 112, 112, 128)	147,584
max_pooling2d_1 (MaxPooling2D)	(None, 56, 56, 128)	0
conv2d_4 (Conv2D)	(None, 56, 56, 256)	295,168
conv2d_5 (Conv2D)	(None, 56, 56, 256)	590,080
conv2d_6 (Conv2D)	(None, 56, 56, 256)	590,080
max_pooling2d_2 (MaxPooling2D)	(None, 28, 28, 256)	0

```
conv2d_7 (Conv2D)               (None, 28, 28, 512)         1,180,160
conv2d_8 (Conv2D)               (None, 28, 28, 512)         2,359,808
conv2d_9 (Conv2D)               (None, 28, 28, 512)         2,359,808
max_pooling2d_3 (MaxPooling2D)  (None, 14, 14, 512)                 0
conv2d_10 (Conv2D)              (None, 14, 14, 512)         2,359,808
conv2d_11 (Conv2D)              (None, 14, 14, 512)         2,359,808
conv2d_12 (Conv2D)              (None, 14, 14, 512)         2,359,808
max_pooling2d_4 (MaxPooling2D)  (None, 7, 7, 512)                   0
flatten (Flatten)               (None, 25088)                       0
dense (Dense)                   (None, 256)                 6,422,784
dense_1 (Dense)                 (None, 64)                     16,448
dense_2 (Dense)                 (None, 4)                         260
```

Total params: 21,154,180 (80.70 MB)

Trainable params: 21,154,180 (80.70 MB)

Non-trainable params: 0 (0.00 B)

[16]:
```
img_shape = (224,224, 3)

model = Sequential([
    Conv2D(filters=64, kernel_size=(3,3), padding="same", activation="relu",␣
 ↪input_shape= img_shape),
    Conv2D(filters=64, kernel_size=(3,3), padding="same", activation="relu"),
    MaxPooling2D((2, 2)),

    Conv2D(filters=128, kernel_size=(3,3), padding="same", activation="relu"),
    Conv2D(filters=128, kernel_size=(3,3), padding="same", activation="relu"),
    MaxPooling2D((2, 2)),

    Conv2D(filters=256, kernel_size=(3,3), padding="same", activation="relu"),
    Conv2D(filters=256, kernel_size=(3,3), padding="same", activation="relu"),
```

```
    Conv2D(filters=256, kernel_size=(3,3), padding="same", activation="relu"),
    MaxPooling2D((2, 2)),

    Conv2D(filters=256, kernel_size=(3,3), padding="same", activation="relu"),
    Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"),
    Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"),
    Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"),
    MaxPooling2D((2, 2)),

    Flatten(),

    Dense(128,activation = "relu"),
    Dense(64,activation = "relu"),
    Dense(4, activation = "softmax")
])

model.compile(Adamax(learning_rate= 0.001), loss= 'categorical_crossentropy',␣
 ↪metrics= ['accuracy'])

model.summary()
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d_13 (Conv2D)	(None, 224, 224, 64)	1,792
conv2d_14 (Conv2D)	(None, 224, 224, 64)	36,928
max_pooling2d_5 (MaxPooling2D)	(None, 112, 112, 64)	0
conv2d_15 (Conv2D)	(None, 112, 112, 128)	73,856
conv2d_16 (Conv2D)	(None, 112, 112, 128)	147,584
max_pooling2d_6 (MaxPooling2D)	(None, 56, 56, 128)	0
conv2d_17 (Conv2D)	(None, 56, 56, 256)	295,168
conv2d_18 (Conv2D)	(None, 56, 56, 256)	590,080
conv2d_19 (Conv2D)	(None, 56, 56, 256)	590,080
max_pooling2d_7 (MaxPooling2D)	(None, 28, 28, 256)	0
conv2d_20 (Conv2D)	(None, 28, 28, 256)	590,080

```
conv2d_21 (Conv2D)              (None, 28, 28, 512)         1,180,160
conv2d_22 (Conv2D)              (None, 28, 28, 512)         2,359,808
conv2d_23 (Conv2D)              (None, 28, 28, 512)         2,359,808
max_pooling2d_8 (MaxPooling2D)  (None, 14, 14, 512)                 0
flatten_1 (Flatten)             (None, 100352)                      0
dense_3 (Dense)                 (None, 128)                12,845,184
dense_4 (Dense)                 (None, 64)                      8,256
dense_5 (Dense)                 (None, 4)                         260
```

Total params: 21,079,044 (80.41 MB)

Trainable params: 21,079,044 (80.41 MB)

Non-trainable params: 0 (0.00 B)

[17]:
```
history = model.fit(train_gen, epochs= 10, verbose= 1, validation_data=␣
↪valid_gen, shuffle= False)
```

```
Epoch 1/10
2024-03-09 17:59:03.378212: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137] Difference at 0:
3.89842, expected 3.37692
2024-03-09 17:59:03.378271: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137] Difference at 12:
3.13412, expected 2.61262
2024-03-09 17:59:03.378281: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137] Difference at 13:
3.08377, expected 2.56227
2024-03-09 17:59:03.378289: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137] Difference at 14:
4.1021, expected 3.5806
2024-03-09 17:59:03.378297: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137] Difference at 29:
3.76539, expected 3.24388
2024-03-09 17:59:03.378305: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137] Difference at 30:
```

```
4.19841, expected 3.6769
2024-03-09 17:59:03.378313: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137] Difference at 63:
3.88366, expected 3.36216
2024-03-09 17:59:03.378321: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137] Difference at 64:
3.84985, expected 3.32834
2024-03-09 17:59:03.378329: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137] Difference at 67:
3.61908, expected 3.09757
2024-03-09 17:59:03.378337: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137] Difference at 84:
3.73098, expected 3.20948
2024-03-09 17:59:03.400623: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:705] Results
mismatch between different convolution algorithms. This is likely a
bug/unexpected loss of precision in cudnn.
(f32[16,64,224,224]{3,2,1,0}, u8[0]{0}) custom-call(f32[16,3,224,224]{3,2,1,0},
f32[64,3,3,3]{3,2,1,0}, f32[64]{0}), window={size=3x3 pad=1_1x1_1},
dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBiasActivationForward", backend_config={"conv_re
sult_scale":1,"activation_mode":"kRelu","side_input_scale":0,"leakyrelu_alpha":0
} for eng20{k2=1,k4=1,k5=1,k6=0,k7=0} vs eng15{k5=1,k6=0,k7=1,k10=1}
2024-03-09 17:59:03.400658: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:270] Device: Tesla
P100-PCIE-16GB
2024-03-09 17:59:03.400670: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:271] Platform:
Compute Capability 6.0
2024-03-09 17:59:03.400678: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:272] Driver: 12020
(535.129.3)
2024-03-09 17:59:03.400686: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:273] Runtime:
<undefined>
2024-03-09 17:59:03.400702: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:280] cudnn version:
8.9.0
2024-03-09 17:59:04.069936: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137] Difference at 0:
3.89842, expected 3.37692
2024-03-09 17:59:04.069990: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137] Difference at 12:
3.13412, expected 2.61262
2024-03-09 17:59:04.069999: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137] Difference at 13:
3.08377, expected 2.56227
2024-03-09 17:59:04.070007: E
```

```
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137] Difference at 14:
4.1021, expected 3.5806
2024-03-09 17:59:04.070016: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137] Difference at 29:
3.76539, expected 3.24388
2024-03-09 17:59:04.070024: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137] Difference at 30:
4.19841, expected 3.6769
2024-03-09 17:59:04.070032: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137] Difference at 63:
3.88366, expected 3.36216
2024-03-09 17:59:04.070040: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137] Difference at 64:
3.84985, expected 3.32834
2024-03-09 17:59:04.070048: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137] Difference at 67:
3.61908, expected 3.09757
2024-03-09 17:59:04.070056: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137] Difference at 84:
3.73098, expected 3.20948
2024-03-09 17:59:04.093354: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:705] Results
mismatch between different convolution algorithms. This is likely a
bug/unexpected loss of precision in cudnn.
(f32[16,64,224,224]{3,2,1,0}, u8[0]{0}) custom-call(f32[16,3,224,224]{3,2,1,0},
f32[64,3,3,3]{3,2,1,0}, f32[64]{0}), window={size=3x3 pad=1_1x1_1},
dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBiasActivationForward", backend_config={"conv_re
sult_scale":1,"activation_mode":"kRelu","side_input_scale":0,"leakyrelu_alpha":0
} for eng20{k2=1,k4=1,k5=1,k6=0,k7=0} vs eng15{k5=1,k6=0,k7=1,k10=1}
2024-03-09 17:59:04.093405: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:270] Device: Tesla
P100-PCIE-16GB
2024-03-09 17:59:04.093415: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:271] Platform:
Compute Capability 6.0
2024-03-09 17:59:04.093422: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:272] Driver: 12020
(535.129.3)
2024-03-09 17:59:04.093429: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:273] Runtime:
<undefined>
2024-03-09 17:59:04.093444: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:280] cudnn version:
8.9.0

  1/357             2:26:18 25s/step -
accuracy: 0.3125 - loss: 3.4722
```

```
WARNING: All log messages before absl::InitializeLog() is called are written to
STDERR
I0000 00:00:1710007163.247731     106 device_compiler.h:186] Compiled cluster
using XLA!  This line is logged at most once for the lifetime of the process.
W0000 00:00:1710007163.270617     106 graph_launch.cc:671] Fallback to op-by-op
mode because memset node breaks graph update

357/357            0s 159ms/step -
accuracy: 0.5298 - loss: 8.0966

W0000 00:00:1710007221.280390     106 graph_launch.cc:671] Fallback to op-by-op
mode because memset node breaks graph update
2024-03-09 18:00:27.766265: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137] Difference at 0:
4.00564, expected 3.36194
2024-03-09 18:00:27.766323: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137] Difference at 8:
5.21235, expected 4.56865
2024-03-09 18:00:27.766332: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137] Difference at 10:
5.39965, expected 4.75595
2024-03-09 18:00:27.766341: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137] Difference at 11:
5.25668, expected 4.61298
2024-03-09 18:00:27.766364: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137] Difference at 12:
3.55266, expected 2.90896
2024-03-09 18:00:27.766372: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137] Difference at 13:
4.84224, expected 4.19854
2024-03-09 18:00:27.766387: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137] Difference at 29:
4.23174, expected 3.58804
2024-03-09 18:00:27.766395: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137] Difference at 30:
4.7388, expected 4.0951
2024-03-09 18:00:27.766402: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137] Difference at 31:
4.74565, expected 4.10195
2024-03-09 18:00:27.766410: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137] Difference at 42:
5.39528, expected 4.75158
2024-03-09 18:00:27.786938: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:705] Results
mismatch between different convolution algorithms. This is likely a
bug/unexpected loss of precision in cudnn.
(f32[15,64,224,224]{3,2,1,0}, u8[0]{0}) custom-call(f32[15,3,224,224]{3,2,1,0},
f32[64,3,3,3]{3,2,1,0}, f32[64]{0}), window={size=3x3 pad=1_1x1_1},
dim_labels=bf01_oi01->bf01,
```

```
custom_call_target="__cudnn$convBiasActivationForward", backend_config={"conv_re
sult_scale":1,"activation_mode":"kRelu","side_input_scale":0,"leakyrelu_alpha":0
} for eng20{k2=1,k4=1,k5=1,k6=0,k7=0} vs eng15{k5=1,k6=0,k7=1,k10=1}
2024-03-09 18:00:27.786967: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:270] Device: Tesla
P100-PCIE-16GB
2024-03-09 18:00:27.786976: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:271] Platform:
Compute Capability 6.0
2024-03-09 18:00:27.786983: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:272] Driver: 12020
(535.129.3)
2024-03-09 18:00:27.786990: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:273] Runtime:
<undefined>
2024-03-09 18:00:27.787004: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:280] cudnn version:
8.9.0
2024-03-09 18:00:28.280564: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137] Difference at 0:
4.00564, expected 3.36194
2024-03-09 18:00:28.280628: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137] Difference at 8:
5.21235, expected 4.56865
2024-03-09 18:00:28.280641: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137] Difference at 10:
5.39965, expected 4.75595
2024-03-09 18:00:28.280652: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137] Difference at 11:
5.25668, expected 4.61298
2024-03-09 18:00:28.280662: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137] Difference at 12:
3.55266, expected 2.90896
2024-03-09 18:00:28.280672: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137] Difference at 13:
4.84224, expected 4.19854
2024-03-09 18:00:28.280683: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137] Difference at 29:
4.23174, expected 3.58804
2024-03-09 18:00:28.280694: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137] Difference at 30:
4.7388, expected 4.0951
2024-03-09 18:00:28.280704: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137] Difference at 31:
4.74565, expected 4.10195
2024-03-09 18:00:28.280715: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137] Difference at 42:
5.39528, expected 4.75158
```

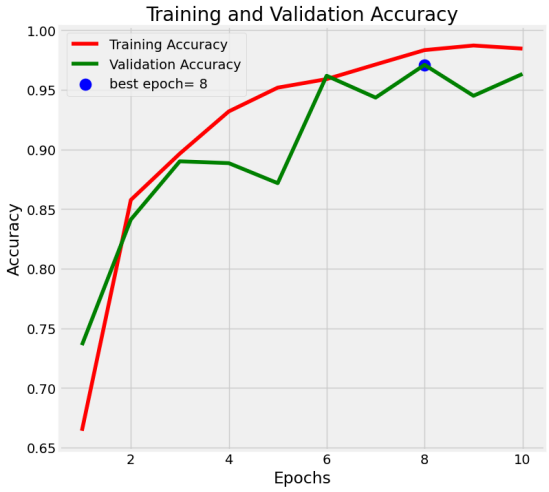
```
2024-03-09 18:00:28.303853: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:705] Results
mismatch between different convolution algorithms. This is likely a
bug/unexpected loss of precision in cudnn.
(f32[15,64,224,224]{3,2,1,0}, u8[0]{0}) custom-call(f32[15,3,224,224]{3,2,1,0},
f32[64,3,3,3]{3,2,1,0}, f32[64]{0}), window={size=3x3 pad=1_1x1_1},
dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBiasActivationForward", backend_config={"conv_re
sult_scale":1,"activation_mode":"kRelu","side_input_scale":0,"leakyrelu_alpha":0
} for eng20{k2=1,k4=1,k5=1,k6=0,k7=0} vs eng15{k5=1,k6=0,k7=1,k10=1}
2024-03-09 18:00:28.303895: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:270] Device: Tesla
P100-PCIE-16GB
2024-03-09 18:00:28.303904: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:271] Platform:
Compute Capability 6.0
2024-03-09 18:00:28.303911: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:272] Driver: 12020
(535.129.3)
2024-03-09 18:00:28.303918: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:273] Runtime:
<undefined>
2024-03-09 18:00:28.303950: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:280] cudnn version:
8.9.0

357/357            96s 201ms/step -
accuracy: 0.5302 - loss: 8.0799 - val_accuracy: 0.7359 - val_loss: 0.7348
Epoch 2/10
357/357            37s 105ms/step -
accuracy: 0.8386 - loss: 0.4525 - val_accuracy: 0.8412 - val_loss: 0.4360
Epoch 3/10
357/357            37s 104ms/step -
accuracy: 0.9014 - loss: 0.2868 - val_accuracy: 0.8901 - val_loss: 0.3047
Epoch 4/10
357/357            37s 104ms/step -
accuracy: 0.9374 - loss: 0.1771 - val_accuracy: 0.8885 - val_loss: 0.3183
Epoch 5/10
357/357            37s 104ms/step -
accuracy: 0.9506 - loss: 0.1518 - val_accuracy: 0.8718 - val_loss: 0.4080
Epoch 6/10
357/357            37s 104ms/step -
accuracy: 0.9474 - loss: 0.1391 - val_accuracy: 0.9618 - val_loss: 0.1387
Epoch 7/10
357/357            37s 104ms/step -
accuracy: 0.9751 - loss: 0.0673 - val_accuracy: 0.9435 - val_loss: 0.2181
Epoch 8/10
357/357            37s 104ms/step -
```

```
accuracy: 0.9840 - loss: 0.0566 - val_accuracy: 0.9710 - val_loss: 0.1389
Epoch 9/10
357/357             37s 104ms/step -
accuracy: 0.9897 - loss: 0.0322 - val_accuracy: 0.9450 - val_loss: 0.3019
Epoch 10/10
357/357             37s 104ms/step -
accuracy: 0.9860 - loss: 0.0441 - val_accuracy: 0.9634 - val_loss: 0.1342
```

0.5 Display model performance

[18]:
```
# Define needed variables
tr_acc = history.history['accuracy']
tr_loss = history.history['loss']
val_acc = history.history['val_accuracy']
val_loss = history.history['val_loss']
index_loss = np.argmin(val_loss)
val_lowest = val_loss[index_loss]
index_acc = np.argmax(val_acc)
acc_highest = val_acc[index_acc]

Epochs = [i+1 for i in range(len(tr_acc))]
loss_label = f'best epoch= {str(index_loss + 1)}'
acc_label = f'best epoch= {str(index_acc + 1)}'

# Plot training history
plt.figure(figsize= (20, 8))
plt.style.use('fivethirtyeight')

plt.subplot(1, 2, 1)
plt.plot(Epochs, tr_loss, 'r', label= 'Training loss')
plt.plot(Epochs, val_loss, 'g', label= 'Validation loss')
plt.scatter(index_loss + 1, val_lowest, s= 150, c= 'blue', label= loss_label)
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(Epochs, tr_acc, 'r', label= 'Training Accuracy')
plt.plot(Epochs, val_acc, 'g', label= 'Validation Accuracy')
plt.scatter(index_acc + 1 , acc_highest, s= 150, c= 'blue', label= acc_label)
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.tight_layout
```

```
plt.show()
```



0.6 Evaluate model

[19]:
```
train_score = model.evaluate(train_gen, verbose= 1)
valid_score = model.evaluate(valid_gen, verbose= 1)
test_score = model.evaluate(test_gen, verbose= 1)

print("Train Loss: ", train_score[0])
print("Train Accuracy: ", train_score[1])
print('-' * 20)
print("Validation Loss: ", valid_score[0])
print("Validation Accuracy: ", valid_score[1])
print('-' * 20)
print("Test Loss: ", test_score[0])
print("Test Accuracy: ", test_score[1])
```

```
357/357             16s 46ms/step -
accuracy: 0.9955 - loss: 0.0169
41/41               2s 44ms/step -
accuracy: 0.9609 - loss: 0.1533
41/41               6s 157ms/step -
accuracy: 0.9566 - loss: 0.1207
Train Loss:  0.016636308282613754
Train Accuracy:  0.9964985847473145
--------------------
Validation Loss:  0.1341988444328308
Validation Accuracy:  0.9633587598800659
--------------------
Test Loss:  0.13041119277477264
```

```
Test Accuracy:  0.957317054271698
```

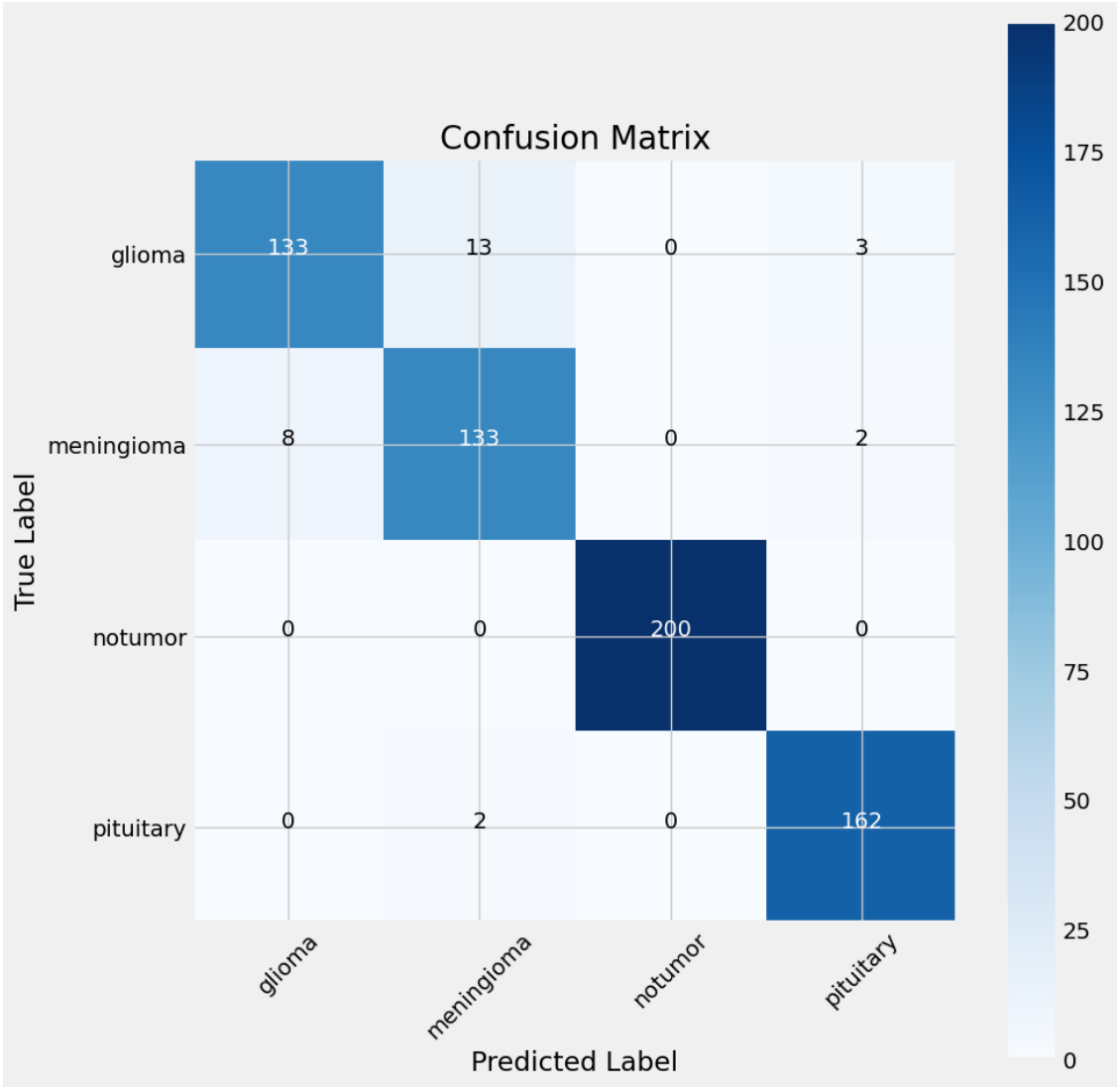
[21]:
```
preds = model.predict(test_gen)  # Assuming test_gen is your test data generator
y_pred = np.argmax(preds, axis=1)
```

```
 6/41              1s 39ms/step
```

```
W0000 00:00:1710007703.550745     106 graph_launch.cc:671] Fallback to op-by-op
mode because memset node breaks graph update
```

```
41/41              2s 43ms/step
```

[29]:
```
import itertools  # Add this line to import itertools module

g_dict = test_gen.class_indices
classes = list(g_dict.keys())
# Confusion matrix
cm = confusion_matrix(test_gen.classes, y_pred)
cm

plt.figure(figsize= (10, 10))
plt.imshow(cm, interpolation= 'nearest', cmap= plt.cm.Blues)
plt.title('Confusion Matrix')
plt.colorbar()

tick_marks = np.arange(len(classes))
plt.xticks(tick_marks, classes, rotation= 45)
plt.yticks(tick_marks, classes)

thresh = cm.max() / 2.
for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
    plt.text(j, i, cm[i, j], horizontalalignment= 'center', color= 'white' if␣
 ↪cm[i, j] > thresh else 'black')

plt.tight_layout()
plt.ylabel('True Label')
plt.xlabel('Predicted Label')

plt.show()
```



[30]:
```
# Classification report
print(classification_report(test_gen.classes, y_pred, target_names= classes))
```

```
              precision    recall  f1-score   support

      glioma       0.94      0.89      0.92       149
  meningioma       0.90      0.93      0.91       143
     notumor       1.00      1.00      1.00       200
   pituitary       0.97      0.99      0.98       164

    accuracy                           0.96       656
   macro avg       0.95      0.95      0.95       656
weighted avg       0.96      0.96      0.96       656
```