```
import os
from PIL import Image
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from glob import glob
#---------------------------------------
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
#---------------------------------------
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Flatten
from tensorflow.keras.optimizers import Adamax
from tensorflow.keras.metrics import Precision, Recall
from tensorflow.keras.preprocessing.image import ImageDataGenerator
#---------------------------------------
import warnings
warnings.filterwarnings("ignore")
import kagglehub

# Download latest version
path = kagglehub.dataset_download("masoudnickparvar/brain-tumor-mri-
dataset")

print("Path to dataset files:", path)
```

```
Using Colab cache for faster access to the 'brain-tumor-mri-dataset'
dataset.
Path to dataset files: /kaggle/input/brain-tumor-mri-dataset
```

```
def train_df(tr_path):
    classes, class_paths = zip(*[(label, os.path.join(tr_path, label,
image))
                                 for label in os.listdir(tr_path) if
os.path.isdir(os.path.join(tr_path, label))
                                 for image in
os.listdir(os.path.join(tr_path, label))])

    tr_df = pd.DataFrame({'Class Path': class_paths, 'Class':
classes})
    return tr_df

def test_df(ts_path):
    classes, class_paths = zip(*[(label, os.path.join(ts_path, label,
image))
                                 for label in os.listdir(ts_path) if
os.path.isdir(os.path.join(ts_path, label))
                                 for image in
```

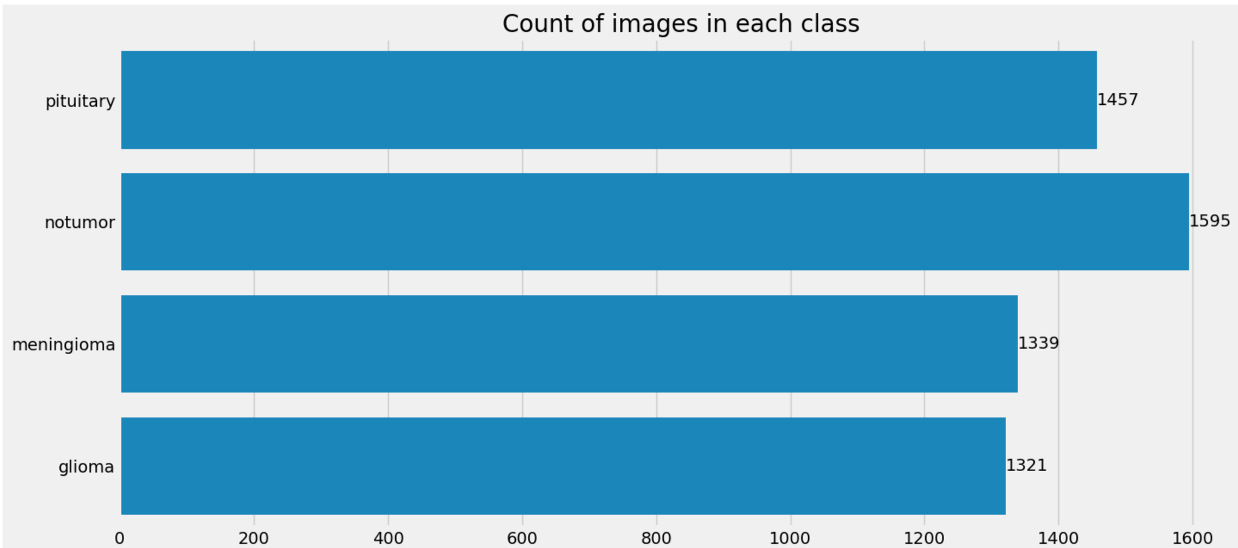
```
os.listdir(os.path.join(ts_path, label))])

    ts_df = pd.DataFrame({'Class Path': class_paths, 'Class':
classes})
    return ts_df

tr_df = train_df('/kaggle/input/brain-tumor-mri-dataset/Training')

tr_df
```

{"summary":"{\n \"name\": \"tr_df\",\n \"rows\": 5712,\n
\"fields\": [\n   {\n   \"column\": \"Class Path\",\n
\"properties\": {\n   \"dtype\": \"string\",\n
\"num_unique_values\": 5712,\n   \"samples\": [\n
\"/kaggle/input/brain-tumor-mri-dataset/Training/notumor/Tr-
no_0294.jpg\",\n
\"/kaggle/input/brain-tumor-mri-dataset/Training/meningioma/Tr-
me_0921.jpg\",\n
\"/kaggle/input/brain-tumor-mri-dataset/Training/pituitary/Tr-
pi_0594.jpg\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Class\",\n   \"properties\": {\n   \"dtype\": \"category\",\
n   \"num_unique_values\": 4,\n   \"samples\": [\n
\"notumor\",\n   \"glioma\",\n   \"pituitary\"\
n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"tr_df"}

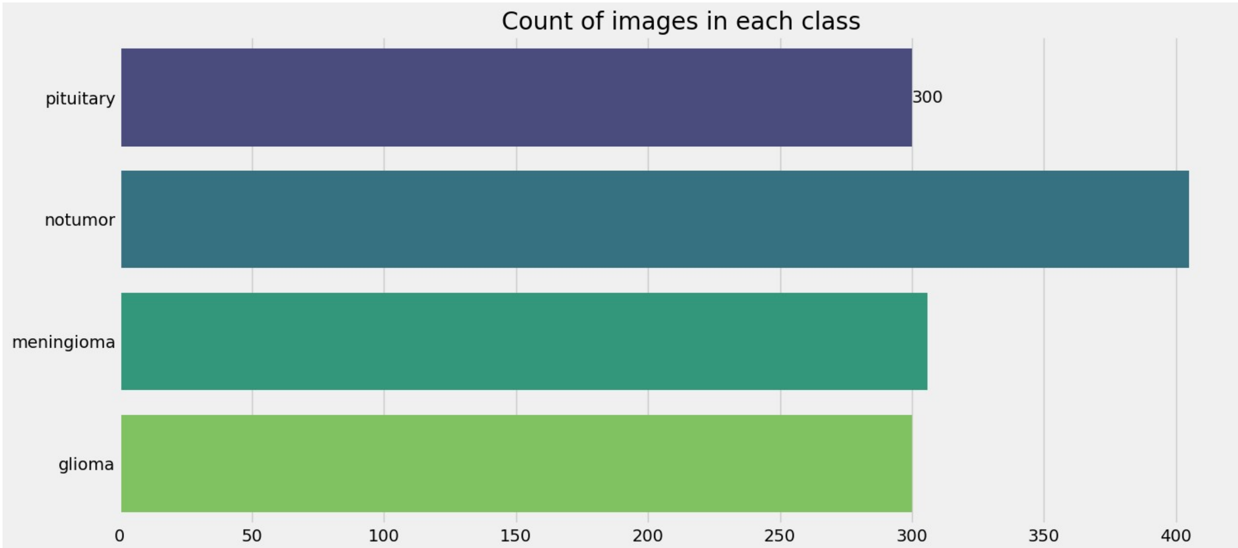
```
ts_df = test_df('/kaggle/input/brain-tumor-mri-dataset/Testing')

ts_df
```

{"summary":"{\n \"name\": \"ts_df\",\n \"rows\": 1311,\n
\"fields\": [\n   {\n   \"column\": \"Class Path\",\n
\"properties\": {\n   \"dtype\": \"string\",\n
\"num_unique_values\": 1311,\n   \"samples\": [\n
\"/kaggle/input/brain-tumor-mri-dataset/Testing/glioma/Te-
glTr_0008.jpg\",\n
\"/kaggle/input/brain-tumor-mri-dataset/Testing/glioma/Te-
gl_0033.jpg\",\n
\"/kaggle/input/brain-tumor-mri-dataset/Testing/pituitary/Te-
pi_0287.jpg\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Class\",\n   \"properties\": {\n   \"dtype\": \"category\",\
n   \"num_unique_values\": 4,\n   \"samples\": [\n
\"notumor\",\n   \"glioma\",\n   \"pituitary\"\
n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"ts_df"}

```python
# Count of images in each class in train data
plt.figure(figsize=(15,7))
ax = sns.countplot(data=tr_df , y=tr_df['Class'])

plt.xlabel('')
plt.ylabel('')
plt.title('Count of images in each class', fontsize=20)
ax.bar_label(ax.containers[0])
plt.show()
```




```python
#Count each class in test data
plt.figure(figsize=(15, 7))
ax = sns.countplot(y=ts_df['Class'], palette='viridis')

ax.set(xlabel='', ylabel='', title='Count of images in each class')
ax.bar_label(ax.containers[0])

plt.show()
```




```
valid_df, ts_df = train_test_split(ts_df, train_size=0.5,
random_state=20, stratify=ts_df['Class'])

valid_df
```

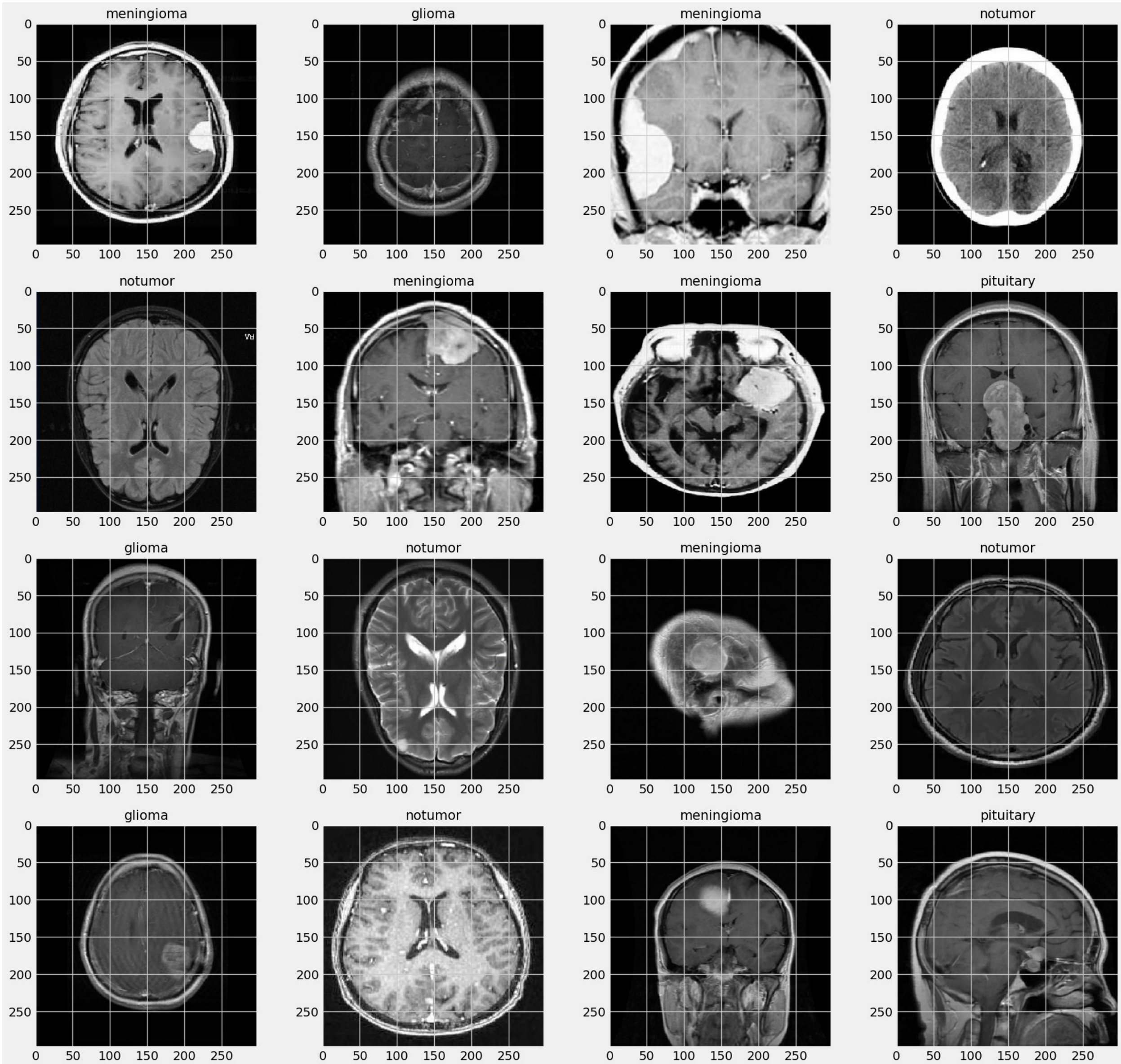
{"summary":"{\n \"name\": \"valid_df\",\n \"rows\": 655,\n
\"fields\": [\n   {\n   \"column\": \"Class Path\",\n
\"properties\": {\n   \"dtype\": \"string\",\n
\"num_unique_values\": 655,\n   \"samples\": [\n
\"/kaggle/input/brain-tumor-mri-dataset/Testing/pituitary/Te-
pi_0063.jpg\",\n
\"/kaggle/input/brain-tumor-mri-dataset/Testing/notumor/Te-
no_0169.jpg\",\n
\"/kaggle/input/brain-tumor-mri-dataset/Testing/notumor/Te-
no_0226.jpg\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Class\",\n   \"properties\": {\n   \"dtype\": \"category\",\
n   \"num_unique_values\": 4,\n   \"samples\": [\n
\"meningioma\",\n   \"notumor\",\n   \"glioma\"\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   }\n]\n}","type":"dataframe","variable_name":"valid_df"}

```
batch_size = 32
img_size = (299, 299)

_gen = ImageDataGenerator(rescale=1/255,
                          brightness_range=(0.8, 1.2))

ts_gen = ImageDataGenerator(rescale=1/255)


tr_gen = _gen.flow_from_dataframe(tr_df, x_col='Class Path',
                                  y_col='Class',
```

```
batch_size=batch_size,
                                  target_size=img_size)

valid_gen = _gen.flow_from_dataframe(valid_df, x_col='Class Path',
                                     y_col='Class',
batch_size=batch_size,
                                     target_size=img_size)

ts_gen = ts_gen.flow_from_dataframe(ts_df, x_col='Class Path',
                                  y_col='Class', batch_size=16,
                                  target_size=img_size, shuffle=False)
```

```
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.
```

```
class_dict = tr_gen.class_indices
classes = list(class_dict.keys())
images, labels = next(ts_gen)

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

for i, (image, label) in enumerate(zip(images, labels)):
    plt.subplot(4,4, i + 1)
    plt.imshow(image)
    class_name = classes[np.argmax(label)]
    plt.title(class_name, color='k', fontsize=15)

plt.show()
```

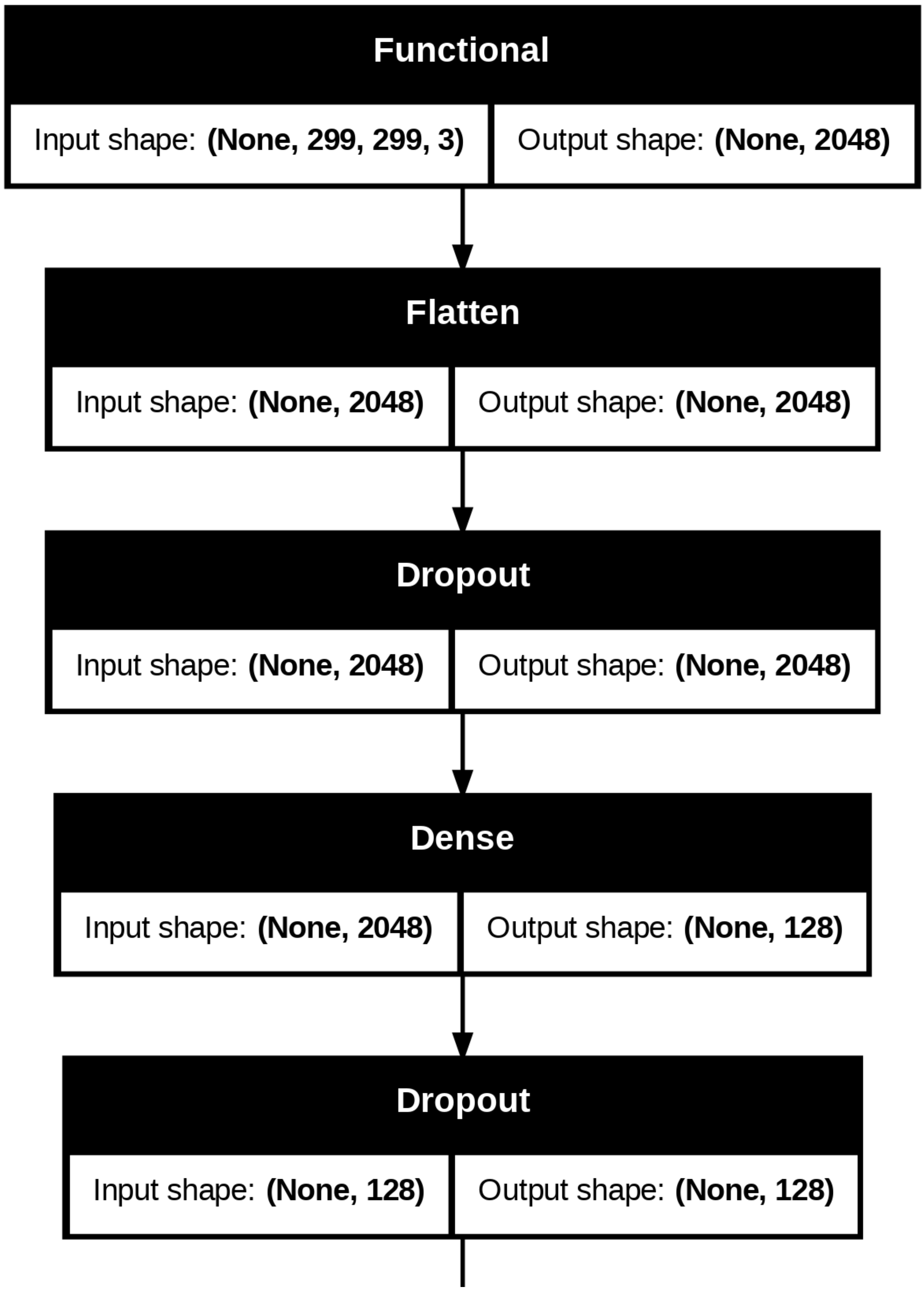



```
#Deep-Learning Model

img_shape=(299,299,3)
base_model = tf.keras.applications.Xception(include_top= False,
weights= "imagenet",
                                            input_shape= img_shape, pooling= 'max')

# for layer in base_model.layers:
#     layer.trainable = False

model = Sequential([
    base_model,
    Flatten(),
    Dropout(rate= 0.3),
```

```
    Dense(128, activation= 'relu'),
    Dropout(rate= 0.25),
    Dense(4, activation= 'softmax')
])

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

model.summary()
```

```
Model: "sequential_1"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ xception (Functional)           │ (None, 2048)           │    20,861,480 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ flatten_1 (Flatten)             │ (None, 2048)           │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dropout_2 (Dropout)             │ (None, 2048)           │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_2 (Dense)                 │ (None, 128)            │       262,272 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dropout_3 (Dropout)             │ (None, 128)            │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_3 (Dense)                 │ (None, 4)              │           516 │
└─────────────────────────────────┴────────────────────────┴───────────────┘

 Total params: 21,124,268 (80.58 MB)

 Trainable params: 21,069,740 (80.37 MB)

 Non-trainable params: 54,528 (213.00 KB)
```

```
tf.keras.utils.plot_model(model, show_shapes=True)
```


Functional
Input shape: (None, 299, 299, 3)
Output shape: (None, 2048)
Flatten
Input shape: (None, 2048)
Output shape: (None, 2048)
Dropout
Input shape: (None, 2048)
Output shape: (None, 2048)
Dense
Input shape: (None, 2048)
Output shape: (None, 128)
Dropout
Input shape: (None, 128)
Output shape: (None, 128)

```
#Testing
hist = model.fit(tr_gen,
                 epochs=10,
                 validation_data=valid_gen,
                 shuffle= False)

Epoch 1/10
179/179 ━━━━━━━━━━━━━━━━━━━━ 150s 838ms/step - accuracy: 0.9966 -
loss: 0.0082 - precision: 0.9966 - recall: 0.9964 - val_accuracy:
0.9863 - val_loss: 0.0334 - val_precision: 0.9863 - val_recall: 0.9863
Epoch 2/10
179/179 ━━━━━━━━━━━━━━━━━━━━ 154s 860ms/step - accuracy: 0.9971 -
loss: 0.0099 - precision: 0.9971 - recall: 0.9971 - val_accuracy:
0.9969 - val_loss: 0.0183 - val_precision: 0.9969 - val_recall: 0.9969
Epoch 3/10
179/179 ━━━━━━━━━━━━━━━━━━━━ 151s 844ms/step - accuracy: 0.9977 -
loss: 0.0092 - precision: 0.9979 - recall: 0.9977 - val_accuracy:
0.9924 - val_loss: 0.0346 - val_precision: 0.9924 - val_recall: 0.9924
Epoch 4/10
179/179 ━━━━━━━━━━━━━━━━━━━━ 152s 848ms/step - accuracy: 0.9989 -
loss: 0.0035 - precision: 0.9989 - recall: 0.9989 - val_accuracy:
0.9969 - val_loss: 0.0051 - val_precision: 0.9969 - val_recall: 0.9969
Epoch 5/10
179/179 ━━━━━━━━━━━━━━━━━━━━ 152s 848ms/step - accuracy: 0.9998 -
loss: 0.0010 - precision: 0.9998 - recall: 0.9998 - val_accuracy:
0.9908 - val_loss: 0.0371 - val_precision: 0.9908 - val_recall: 0.9908
Epoch 6/10
179/179 ━━━━━━━━━━━━━━━━━━━━ 152s 847ms/step - accuracy: 0.9988 -
loss: 0.0075 - precision: 0.9988 - recall: 0.9988 - val_accuracy:
0.9939 - val_loss: 0.0148 - val_precision: 0.9939 - val_recall: 0.9939
Epoch 7/10
179/179 ━━━━━━━━━━━━━━━━━━━━ 152s 850ms/step - accuracy: 0.9994 -
loss: 0.0025 - precision: 0.9994 - recall: 0.9994 - val_accuracy:
0.9924 - val_loss: 0.0126 - val_precision: 0.9924 - val_recall: 0.9924
Epoch 8/10
179/179 ━━━━━━━━━━━━━━━━━━━━ 152s 848ms/step - accuracy: 0.9992 -
loss: 0.0028 - precision: 0.9992 - recall: 0.9992 - val_accuracy:
0.9985 - val_loss: 0.0107 - val_precision: 0.9985 - val_recall: 0.9985
Epoch 9/10
179/179 ━━━━━━━━━━━━━━━━━━━━ 152s 850ms/step - accuracy: 0.9988 -
loss: 0.0039 - precision: 0.9992 - recall: 0.9988 - val_accuracy:
0.9924 - val_loss: 0.0292 - val_precision: 0.9924 - val_recall: 0.9924
Epoch 10/10
179/179 ━━━━━━━━━━━━━━━━━━━━ 152s 846ms/step - accuracy: 1.0000 -
loss: 5.7295e-04 - precision: 1.0000 - recall: 1.0000 - val_accuracy:
0.9908 - val_loss: 0.0335 - val_precision: 0.9908 - val_recall: 0.9908

#Training
hist = model.fit(tr_gen,
                 epochs=10,
```

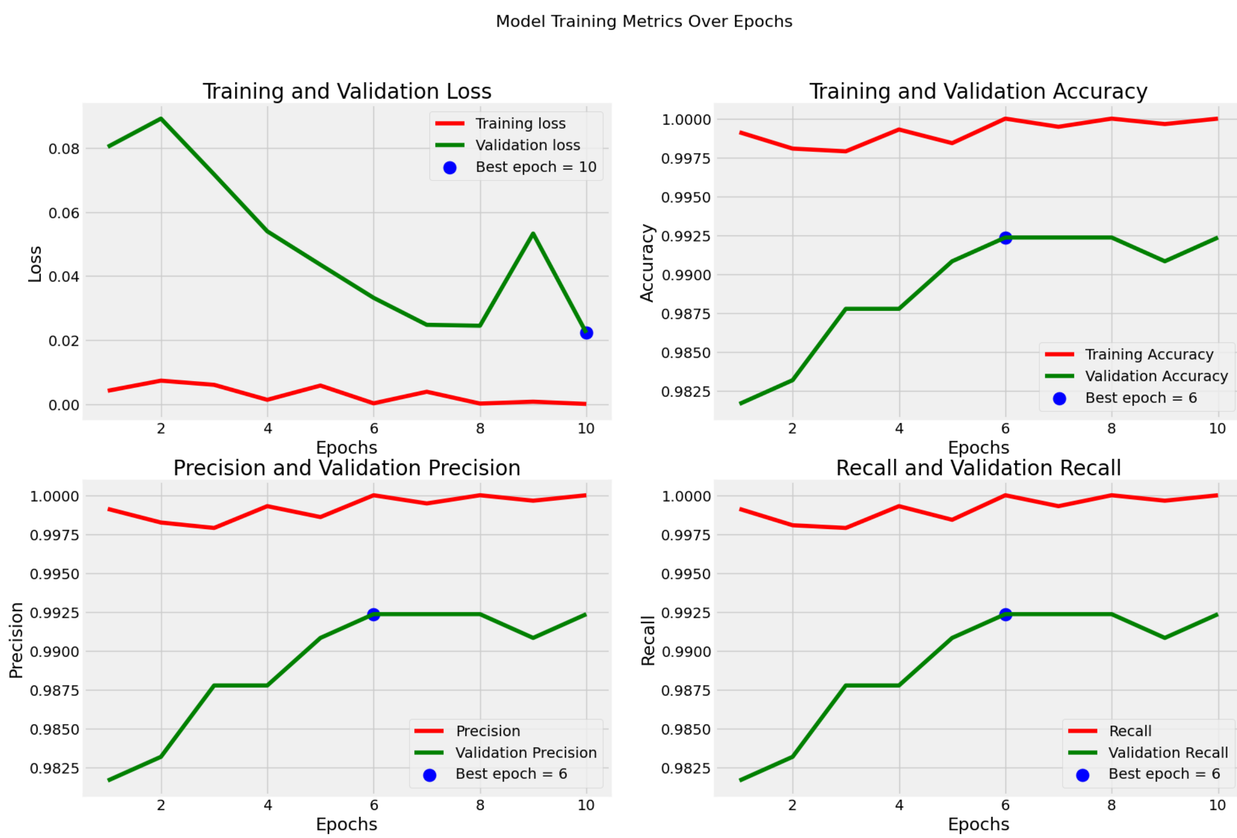
```
                 validation_data=valid_gen,
                 shuffle= False)
```

```
Epoch 1/10
179/179 ━━━━━━━━━━━━━━━━━━━━ 153s 852ms/step - accuracy: 0.9997 -
loss: 0.0017 - precision: 0.9997 - recall: 0.9997 - val_accuracy:
0.9817 - val_loss: 0.0804 - val_precision: 0.9817 - val_recall: 0.9817
Epoch 2/10
179/179 ━━━━━━━━━━━━━━━━━━━━ 152s 848ms/step - accuracy: 0.9968 -
loss: 0.0115 - precision: 0.9971 - recall: 0.9968 - val_accuracy:
0.9832 - val_loss: 0.0892 - val_precision: 0.9832 - val_recall: 0.9832
Epoch 3/10
179/179 ━━━━━━━━━━━━━━━━━━━━ 152s 846ms/step - accuracy: 0.9978 -
loss: 0.0065 - precision: 0.9978 - recall: 0.9978 - val_accuracy:
0.9878 - val_loss: 0.0718 - val_precision: 0.9878 - val_recall: 0.9878
Epoch 4/10
179/179 ━━━━━━━━━━━━━━━━━━━━ 152s 849ms/step - accuracy: 0.9998 -
loss: 6.8921e-04 - precision: 0.9998 - recall: 0.9998 - val_accuracy:
0.9878 - val_loss: 0.0540 - val_precision: 0.9878 - val_recall: 0.9878
Epoch 5/10
179/179 ━━━━━━━━━━━━━━━━━━━━ 152s 848ms/step - accuracy: 0.9986 -
loss: 0.0045 - precision: 0.9989 - recall: 0.9986 - val_accuracy:
0.9908 - val_loss: 0.0435 - val_precision: 0.9908 - val_recall: 0.9908
Epoch 6/10
179/179 ━━━━━━━━━━━━━━━━━━━━ 152s 848ms/step - accuracy: 1.0000 -
loss: 2.7510e-04 - precision: 1.0000 - recall: 1.0000 - val_accuracy:
0.9924 - val_loss: 0.0332 - val_precision: 0.9924 - val_recall: 0.9924
Epoch 7/10
179/179 ━━━━━━━━━━━━━━━━━━━━ 152s 849ms/step - accuracy: 0.9993 -
loss: 0.0049 - precision: 0.9993 - recall: 0.9991 - val_accuracy:
0.9924 - val_loss: 0.0247 - val_precision: 0.9924 - val_recall: 0.9924
Epoch 8/10
179/179 ━━━━━━━━━━━━━━━━━━━━ 152s 848ms/step - accuracy: 1.0000 -
loss: 1.3885e-04 - precision: 1.0000 - recall: 1.0000 - val_accuracy:
0.9924 - val_loss: 0.0245 - val_precision: 0.9924 - val_recall: 0.9924
Epoch 9/10
179/179 ━━━━━━━━━━━━━━━━━━━━ 152s 849ms/step - accuracy: 0.9995 -
loss: 0.0010 - precision: 0.9995 - recall: 0.9995 - val_accuracy:
0.9908 - val_loss: 0.0533 - val_precision: 0.9908 - val_recall: 0.9908
Epoch 10/10
179/179 ━━━━━━━━━━━━━━━━━━━━ 152s 846ms/step - accuracy: 1.0000 -
loss: 5.7918e-05 - precision: 1.0000 - recall: 1.0000 - val_accuracy:
0.9924 - val_loss: 0.0223 - val_precision: 0.9924 - val_recall: 0.9924
```

```
#Visualize model performance
tr_acc = hist.history['accuracy']
tr_loss = hist.history['loss']
tr_per = hist.history['precision']
tr_recall = hist.history['recall']
val_acc = hist.history['val_accuracy']
```

```
val_loss = hist.history['val_loss']
val_per = hist.history['val_precision']
val_recall = hist.history['val_recall']

index_loss = np.argmin(val_loss)
val_lowest = val_loss[index_loss]
index_acc = np.argmax(val_acc)
acc_highest = val_acc[index_acc]
index_precision = np.argmax(val_per)
per_highest = val_per[index_precision]
index_recall = np.argmax(val_recall)
recall_highest = val_recall[index_recall]

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)}'
per_label = f'Best epoch = {str(index_precision + 1)}'
recall_label = f'Best epoch = {str(index_recall + 1)}'


plt.figure(figsize=(20, 12))
plt.style.use('fivethirtyeight')


plt.subplot(2, 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.grid(True)

plt.subplot(2, 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.grid(True)

plt.subplot(2, 2, 3)
plt.plot(Epochs, tr_per, 'r', label='Precision')
plt.plot(Epochs, val_per, 'g', label='Validation Precision')
plt.scatter(index_precision + 1, per_highest, s=150, c='blue',
```

```
label=per_label)
plt.title('Precision and Validation Precision')
plt.xlabel('Epochs')
plt.ylabel('Precision')
plt.legend()
plt.grid(True)

plt.subplot(2, 2, 4)
plt.plot(Epochs, tr_recall, 'r', label='Recall')
plt.plot(Epochs, val_recall, 'g', label='Validation Recall')
plt.scatter(index_recall + 1, recall_highest, s=150, c='blue',
label=recall_label)
plt.title('Recall and Validation Recall')
plt.xlabel('Epochs')
plt.ylabel('Recall')
plt.legend()
plt.grid(True)

plt.suptitle('Model Training Metrics Over Epochs', fontsize=16)
plt.show()
```




```
#Testing and Evaluation
train_score = model.evaluate(tr_gen, verbose=1)
valid_score = model.evaluate(valid_gen, verbose=1)
```

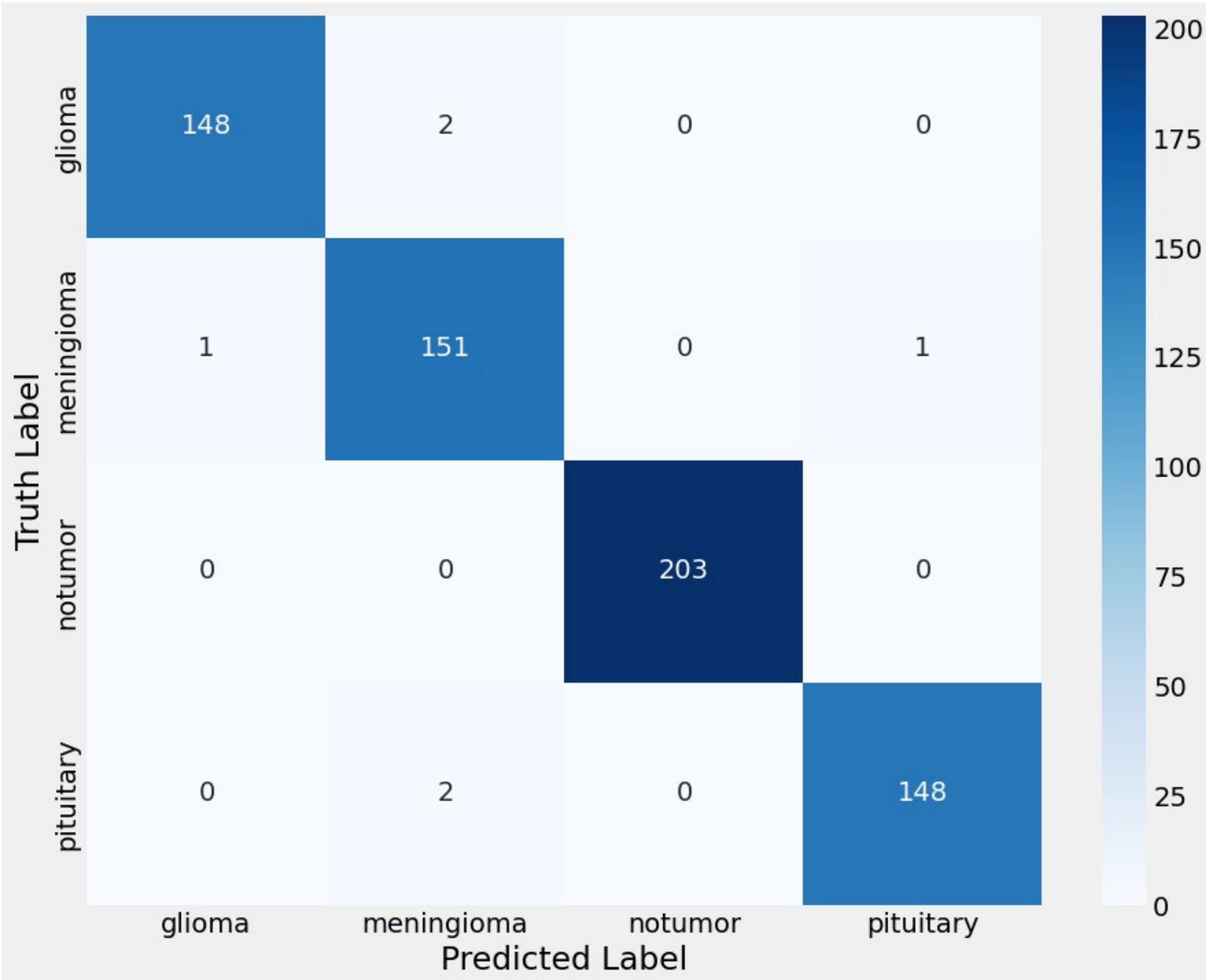
```
test_score = model.evaluate(ts_gen, verbose=1)

print(f"Train Loss: {train_score[0]:.4f}")
print(f"Train Accuracy: {train_score[1]*100:.2f}%")
print('-' * 20)
print(f"Validation Loss: {valid_score[0]:.4f}")
print(f"Validation Accuracy: {valid_score[1]*100:.2f}%")
print('-' * 20)
print(f"Test Loss: {test_score[0]:.4f}")
print(f"Test Accuracy: {test_score[1]*100:.2f}%")
```

```
179/179 ━━━━━━━━━━━━━━━━━━━━ 55s 306ms/step - accuracy: 1.0000 - loss:
1.9686e-06 - precision: 1.0000 - recall: 1.0000
21/21 ━━━━━━━━━━━━━━━━━━━━ 5s 228ms/step - accuracy: 0.9904 - loss:
0.0390 - precision: 0.9904 - recall: 0.9904
41/41 ━━━━━━━━━━━━━━━━━━━━ 6s 147ms/step - accuracy: 0.9941 - loss:
0.0746 - precision: 0.9941 - recall: 0.9941
Train Loss: 0.0000
Train Accuracy: 100.00%
--------------------
Validation Loss: 0.0397
Validation Accuracy: 99.08%
--------------------
Test Loss: 0.0944
Test Accuracy: 99.09%
```

```
preds = model.predict(ts_gen)
y_pred = np.argmax(preds, axis=1)
```

```
41/41 ━━━━━━━━━━━━━━━━━━━━ 16s 104ms/step
```

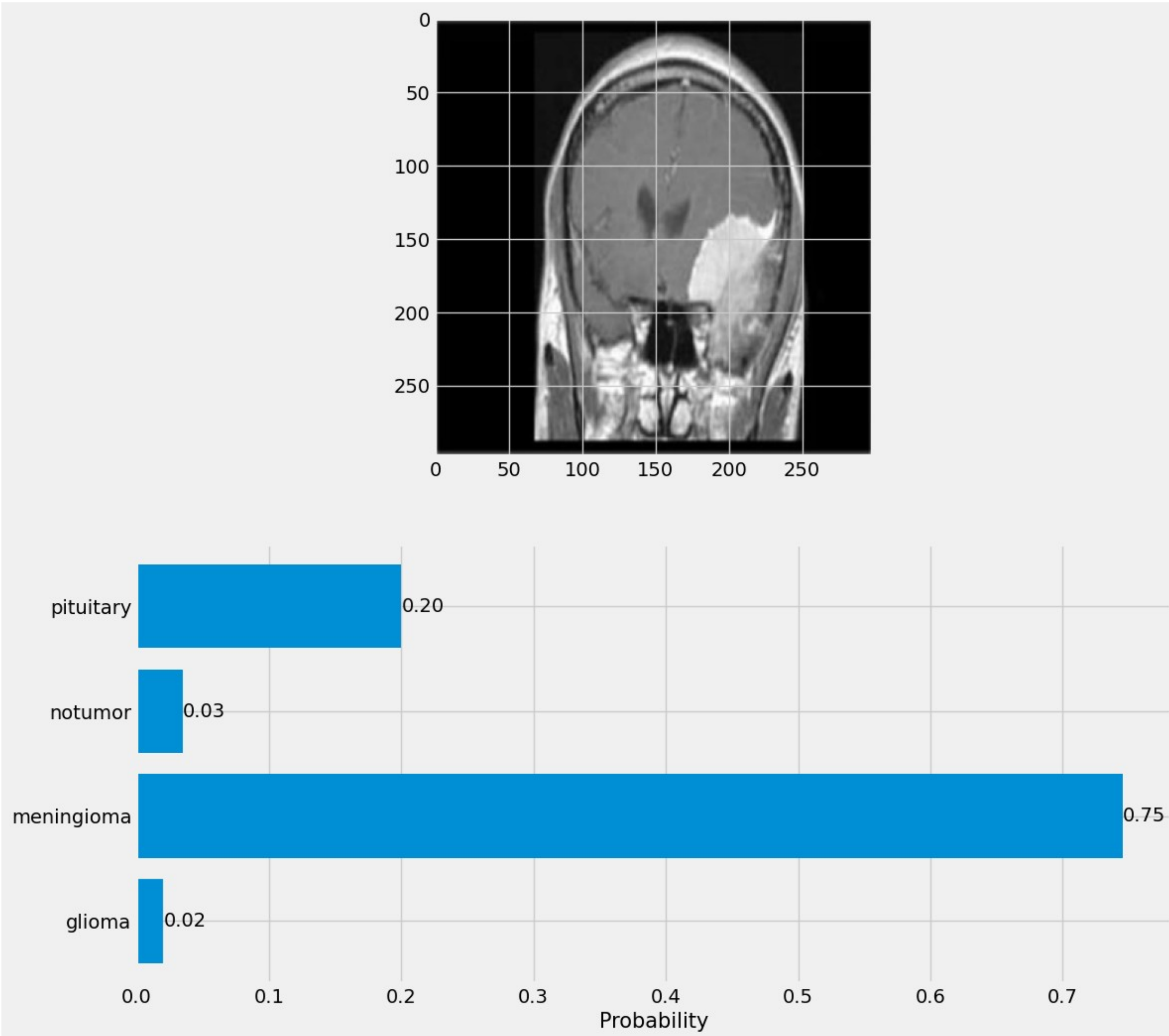
```
cm = confusion_matrix(ts_gen.classes, y_pred)
labels = list(class_dict.keys())
plt.figure(figsize=(10,8))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=labels,
yticklabels=labels)
plt.xlabel('Predicted Label')
plt.ylabel('Truth Label')
plt.show()
```




```
clr = classification_report(ts_gen.classes, y_pred)
print(clr)

              precision    recall  f1-score   support

           0       0.99      0.99      0.99       150
           1       0.97      0.99      0.98       153
           2       1.00      1.00      1.00       203
           3       0.99      0.99      0.99       150

    accuracy                           0.99       656
   macro avg       0.99      0.99      0.99       656
weighted avg       0.99      0.99      0.99       656


#Testing
def predict(img_path):
    import numpy as np
    import matplotlib.pyplot as plt
    from PIL import Image
```

```
    label = list(class_dict.keys())
    plt.figure(figsize=(12, 12))
    img = Image.open(img_path)
    resized_img = img.resize((299, 299))
    img = np.asarray(resized_img)
    img = np.expand_dims(img, axis=0)
    img = img / 255
    predictions = model.predict(img)
    probs = list(predictions[0])
    labels = label
    plt.subplot(2, 1, 1)
    plt.imshow(resized_img)
    plt.subplot(2, 1, 2)
    bars = plt.barh(labels, probs)
    plt.xlabel('Probability', fontsize=15)
    ax = plt.gca()
    ax.bar_label(bars, fmt = '%.2f')
    plt.show()

predict('/kaggle/input/brain-tumor-mri-dataset/Testing/meningioma/Te-
meTr_0000.jpg')
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 18s 18s/step
```



0
50
100
150
200
250
0
50
100
150
200
250
pituitary
notumor
meningioma
glioma
0.20
0.03
0.75
0.02
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
Probability