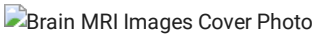
Detection and Classification of Brain Tumor through Brain MRI Images Using Deep Learning



1.Introduction

- The occurrence of brain tumor patients in India is steadily rising, more and more number of cases are reported each year in India across various age groups.
- The International Association of Cancer Registries (IARC) reported that there are over $28,000$ cases of brain tumours reported in India each year and more than $24,000$ people reportedly i.e. 85.72% of the total reported die due to brain tumours annually. Brain tumour's are a serious condition and in most cases fatal if not detected & treated in early stages.

2. Setting Up Local Storage for Dataset

2.1 Giving Access To Google Drive

```
from google.colab import drive
drive.mount('/content/gdrive/')
```

```
Mounted at /content/gdrive/
```

2.2 Checking OS Version and Details

```
print("OS Version & Details: ")
!lsb_release -a
```

```
OS Version & Details:
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 22.04.4 LTS
Release:        22.04
Codename:       jammy
```

3. Importing Required Libraries

```
import sys
import os
import math

import numpy as np
import pandas as pd

from matplotlib import pyplot as plt
from matplotlib import rcParams
rcParams['figure.dpi'] = 300
%matplotlib inline
import seaborn as sns
import missingno as msno
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import *
from tensorflow.keras.optimizers import *
from tensorflow.keras.applications import InceptionV3
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
from sklearn.model_selection import train_test_split, KFold
from sklearn.metrics import *

from PIL import Image, ImageEnhance
from tensorflow.keras.preprocessing.image import *

print(f'Tensorflow Version: {tf.__version__}.')
```

```
Tensorflow Version: 2.18.0.
```

4. Setting Up the Environment

```
gpu_device_location = tpu_device_location = cpu_device_location = None

if os.environ['COLAB_GPU'] == '1':
    print("Allocated GPU Runtime Details:")
    !nvidia-smi
    print()
    try:
        import pynvml
        pynvml.nvmlInit()
        handle = pynvml.nvmlDeviceGetHandleByIndex(0)
        gpu_device_name = pynvml.nvmlDeviceGetName(handle)

        if gpu_device_name not in {b'Tesla T4', b'Tesla P4', b'Tesla P100-PCIE-16GB'}:
            raise Exception("Unfortunately this instance does not have a T4, P4 or P100 GPU.\nSometimes Colab allocates a Tesla K80
    except Exception as hardware_exception:
        print(hardware_exception, end = '\n\n')
    gpu_device_location = tf.test.gpu_device_name()
    print(f"{gpu_device_name} is allocated sucessfully at location: {gpu_device_location}")
elif 'COLAB_TPU_ADDR' in os.environ:
    tpu_device_location = f"grpc://{os.environ['COLAB_TPU_ADDR']}"
    print(f"TPU is allocated successfully at location: {tpu_device_location}.")
    resolver = tf.distribute.cluster_resolver.TPUClusterResolver(tpu_location)
    tf.config.experimental_connect_to_cluster(resolver)
    tf.tpu.experimental.initialize_tpu_system(resolver)
    tpu_strategy = tf.distribute.TPUStrategy()
else:
    cpu_device_location = "/cpu:0"
    print("GPUs and TPUs are not allocated successfully, hence runtime fallbacked to CPU.")
```

```
Allocated GPU Runtime Details:
Tue Apr  1 04:51:38 2025
+-----------------------------------------------------------------------------------------+
| NVIDIA-SMI 550.54.15              Driver Version: 550.54.15      CUDA Version: 12.4     |
|-----------------------------------------+------------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util  Compute M. |
|                                         |                        |               MIG M. |
|=========================================+========================+======================|
|   0  Tesla T4                       Off |   00000000:00:04.0 Off |                    0 |
| N/A   43C    P0             28W /   70W |     102MiB /  15360MiB |      0%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+

+-----------------------------------------------------------------------------------------+
| Processes:                                                                              |
|  GPU   GI   CI        PID   Type   Process name                              GPU Memory |
```

```
|        ID   ID                                                                 Usage      |
|=============================================================================================|
+---------------------------------------------------------------------------------------------+

Unfortunately this instance does not have a T4, P4 or P100 GPU.
Sometimes Colab allocates a Tesla K80 instead of a T4, P4 or P100.
If you get Tesla K80 then you can factory reset your runtime to get another GPUs.

Tesla T4 is allocated sucessfully at location: /device:GPU:0
```

4.1 Installation of `tree` Utility Using `Bash`.

```
%%bash
RED_COLOR='\033[0;31m'
NO_COLOR='\033[0m'
pkg_name=tree
dpkg -s $pkg_name &> /dev/null
if [ "$?" -ne "0" ]
    then
        echo "Installing tree utility..."
        apt-get autoclean
        apt-get autoremove
        apt-get install $pkg_name
        if [ "$?" -eq "0" ]
            then
                echo -e ${RED_COLOR}"tree utility installed sucessfully.\n"${NO_COLOR}
        fi
    else
        echo "tree utility is already installed."
fi
tree --version
```

```
Installing tree utility...
Reading package lists...
Building dependency tree...
Reading state information...
Reading package lists...
Building dependency tree...
Reading state information...
0 upgraded, 0 newly installed, 0 to remove and 29 not upgraded.
Reading package lists...
Building dependency tree...
Reading state information...
The following NEW packages will be installed:
  tree
0 upgraded, 1 newly installed, 0 to remove and 29 not upgraded.
Need to get 47.9 kB of archives.
After this operation, 116 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy/universe amd64 tree amd64 2.0.2-1 [47.9 kB]
Fetched 47.9 kB in 1s (71.7 kB/s)
Selecting previously unselected package tree.
(Reading database ... 126210 files and directories currently installed.)
Preparing to unpack .../tree_2.0.2-1_amd64.deb ...
Unpacking tree (2.0.2-1) ...
Setting up tree (2.0.2-1) ...
Processing triggers for man-db (2.10.2-1) ...
tree utility installed sucessfully.

tree v2.0.2 (c) 1996 - 2022 by Steve Baker, Thomas Moore, Francesc Rocher, Florian Sesser, Kyosuke Tokoro
```

Extraction

```
import zipfile
import os

zip_path = "/content/gdrive/MyDrive/Dataset/drive-download-20250331T145914Z-001.zip"
extract_path = "/content/gdrive/MyDrive/Dataset/Extracted"

# Extract the zip file
with zipfile.ZipFile(zip_path, 'r') as zip_ref:
    zip_ref.extractall(extract_path)

print("Extraction completed.")
```

```
Extraction completed.
```

```
import os

dataset_path = "/content/gdrive/MyDrive/Dataset/Extracted"
```

```
print("Directories inside dataset path:", os.listdir(dataset_path))
```

```
Directories inside dataset path: ['README.txt', 'Testing', 'Tumor-Mask', 'Training', 'Brain-Tumor-Images-Mat-Files']
```

```
for root, dirs, files in os.walk(dataset_path):
    print(f"Inside {root}:")
    print("Directories:", dirs)
    print("Files:", files)
```

```
Inside /content/gdrive/MyDrive/Dataset/Extracted:
Directories: ['Testing', 'Tumor-Mask', 'Training', 'Brain-Tumor-Images-Mat-Files']
Files: ['README.txt']
Inside /content/gdrive/MyDrive/Dataset/Extracted/Testing:
Directories: ['Yes', 'No']
Files: []
Inside /content/gdrive/MyDrive/Dataset/Extracted/Testing/Yes:
Directories: []
Files: ['Y149.JPG', 'Y1.jpg', 'Y182.png', 'Y181.jpg', 'Y180.jpg', 'Y170.jpg', 'Y183.png', 'Y169.jpg', 'Y168.jpeg', 'Y176.jpg',
Inside /content/gdrive/MyDrive/Dataset/Extracted/Testing/No:
Directories: []
Files: ['N100.jpg', 'N7.jpg', 'N1.jpg', 'N99.jpg', 'N5.jpg', 'N90.jpg', 'N38.jpg', 'N76.jpg', 'N83.jpg', 'N23.jpg', 'N95.jpg',
Inside /content/gdrive/MyDrive/Dataset/Extracted/Tumor-Mask:
Directories: ['glioma', 'pituitary_tumor', 'meningioma']
Files: []
Inside /content/gdrive/MyDrive/Dataset/Extracted/Tumor-Mask/glioma:
Directories: []
Files: ['G1426.jpg', 'G1425.jpg', 'G1424.jpg', 'G1423.jpg', 'G1422.jpg', 'G1421.jpg', 'G1420.jpg', 'G1419.jpg', 'G1417.jpg', '(
Inside /content/gdrive/MyDrive/Dataset/Extracted/Tumor-Mask/pituitary_tumor:
Directories: []
Files: ['P930.jpg', 'P929.jpg', 'P928.jpg', 'P927.jpg', 'P926.jpg', 'P925.jpg', 'P924.jpg', 'P923.jpg', 'P922.jpg', 'P920.jpg',
Inside /content/gdrive/MyDrive/Dataset/Extracted/Tumor-Mask/meningioma:
Directories: []
Files: ['M707.jpg', 'M708.jpg', 'M703.jpg', 'M706.jpg', 'M699.jpg', 'M696.jpg', 'M702.jpg', 'M700.jpg', 'M693.jpg', 'M705.jpg',
Inside /content/gdrive/MyDrive/Dataset/Extracted/Training:
Directories: ['no_tumor', 'meningioma', 'pituitary_tumor', 'glioma']
Files: []
Inside /content/gdrive/MyDrive/Dataset/Extracted/Training/no_tumor:
Directories: []
Files: ['N391.jpg', 'N130.jpg', 'N71.jpg', 'N111.jpg', 'N304.jpg', 'N10.jpg', 'N372.jpg', 'N272.jpg', 'N245.jpg', 'N208.jpg',
Inside /content/gdrive/MyDrive/Dataset/Extracted/Training/meningioma:
Directories: []
Files: ['M706.jpg', 'M705.jpg', 'M708.jpg', 'M704.jpg', 'M702.jpg', 'M700.jpg', 'M703.jpg', 'M701.jpg', 'M694.jpg', 'M697.jpg',
Inside /content/gdrive/MyDrive/Dataset/Extracted/Training/pituitary_tumor:
Directories: []
Files: ['P930.jpg', 'P929.jpg', 'P928.jpg', 'P927.jpg', 'P926.jpg', 'P925.jpg', 'P924.jpg', 'P923.jpg', 'P922.jpg', 'P920.jpg',
Inside /content/gdrive/MyDrive/Dataset/Extracted/Training/glioma:
Directories: []
Files: ['G1425.jpg', 'G1423.jpg', 'G1424.jpg', 'G1426.jpg', 'G1420.jpg', 'G1422.jpg', 'G1418.jpg', 'G1419.jpg', 'G1409.jpg', '(
Inside /content/gdrive/MyDrive/Dataset/Extracted/Brain-Tumor-Images-Mat-Files:
Directories: []
Files: ['3060.mat', '3056.mat', '3062.mat', '3061.mat', '3057.mat', '3058.mat', '3063.mat', '3052.mat', '3059.mat', '3046.mat',
```

4.2 Display of File Structure

```
!tree -d -C "/content/gdrive/MyDrive/Dataset/Extracted"
```

```
/content/gdrive/MyDrive/Dataset/Extracted
├── Brain-Tumor-Images-Mat-Files
├── Testing
│   ├── No
│   └── Yes
├── Training
│   ├── glioma
│   ├── meningioma
│   ├── no_tumor
│   └── pituitary_tumor
└── Tumor-Mask
    ├── glioma
    ├── meningioma
    └── pituitary_tumor

13 directories
```

4.3 Setting Up Paths to Root and Data Directories

```
ROOT_DIR = r"/content/gdrive/MyDrive/Dataset/Extracted"
DATA_ROOT_DIR = os.path.join(ROOT_DIR, "/content/gdrive/MyDrive/Dataset/Extracted/Brain-Tumor-Images-Mat-Files")
TRAIN_DIR = os.path.join(DATA_ROOT_DIR, '/content/gdrive/MyDrive/Dataset/Extracted/Training')
MASK_DIR = os.path.join(DATA_ROOT_DIR, '/content/gdrive/MyDrive/Dataset/Extracted/Tumor-Mask')
TEST_DIR = os.path.join(DATA_ROOT_DIR, '/content/gdrive/MyDrive/Dataset/Extracted/Testing')
```

```
assert os.path.isdir(ROOT_DIR) and os.path.isdir(DATA_ROOT_DIR) and os.path.isdir(TRAIN_DIR) and os.path.isdir(MASK_DIR)
TUMOR_CLASS = ['meningioma', 'glioma', 'pituitary_tumor', 'no_tumor']
IMAGE_DATA_PATHS = [os.path.join(TRAIN_DIR, tumor_class) for tumor_class in TUMOR_CLASS]
MASK_DATA_PATHS = [os.path.join(MASK_DIR, tumor_name) for tumor_name in TUMOR_CLASS[:-1]]
```

5. Data Preprocessing and Exploratory Data Analysis

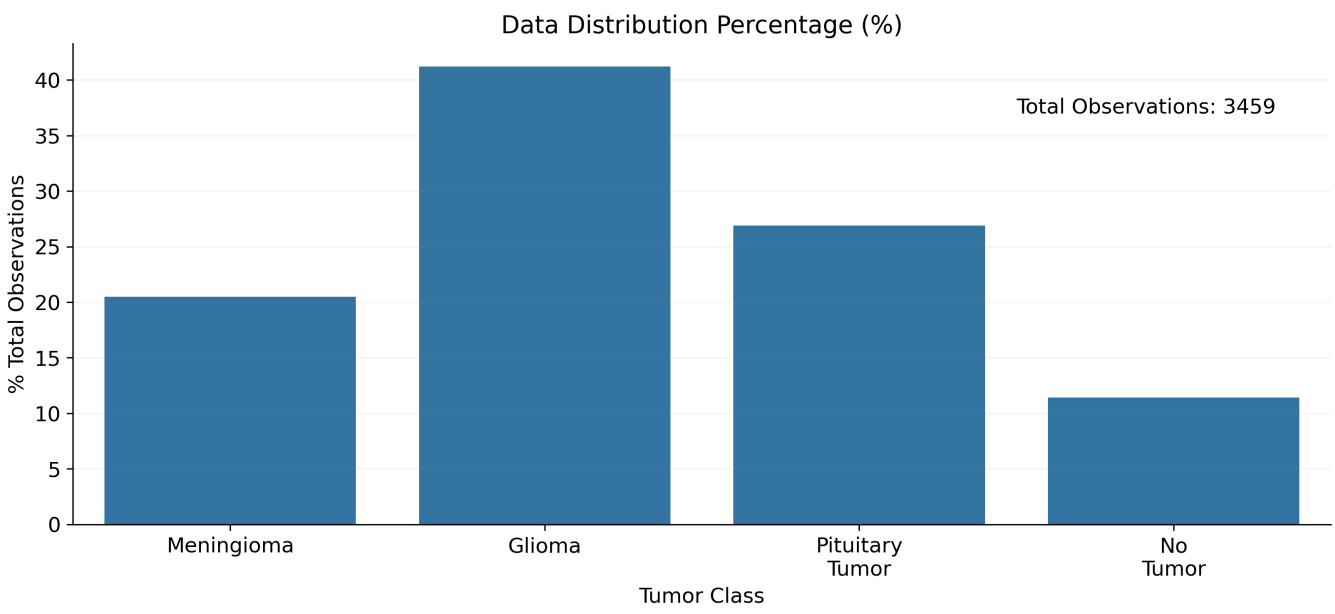
```
data_distribution_count = pd.Series([len(os.listdir(path)) for path in IMAGE_DATA_PATHS if os.path.exists(path) and os.path.isdir(
                                     index = TUMOR_CLASS)
data_distribution_count
```

	0
meningioma	708
glioma	1426
pituitary_tumor	930
no_tumor	395

dtype: int64

5.1 Data Distribution Visualization

```
fig, axis = plt.subplots(figsize = (13, 5))
axis.grid(True, alpha = 0.1)
axis.set_title("Data Distribution Percentage (%)", fontsize = 14)
sns.barplot(x = ['\n'.join(curr_index.strip().split('_')).title() for curr_index in data_distribution_count.index],
            y = 100 * (data_distribution_count / data_distribution_count.sum()), ax = axis)
axis.set_xlabel("Tumor Class", fontsize = 12)
axis.set_ylabel("% Total Observations", fontsize = 12)
axis.tick_params(which = 'major', labelsize = 12)
axis.text(2.5, 37, f'Total Observations: {data_distribution_count.sum()}', fontdict = dict(size = 12))
sns.despine()
```



5.2 Visualisation of Brain MRI Dataset

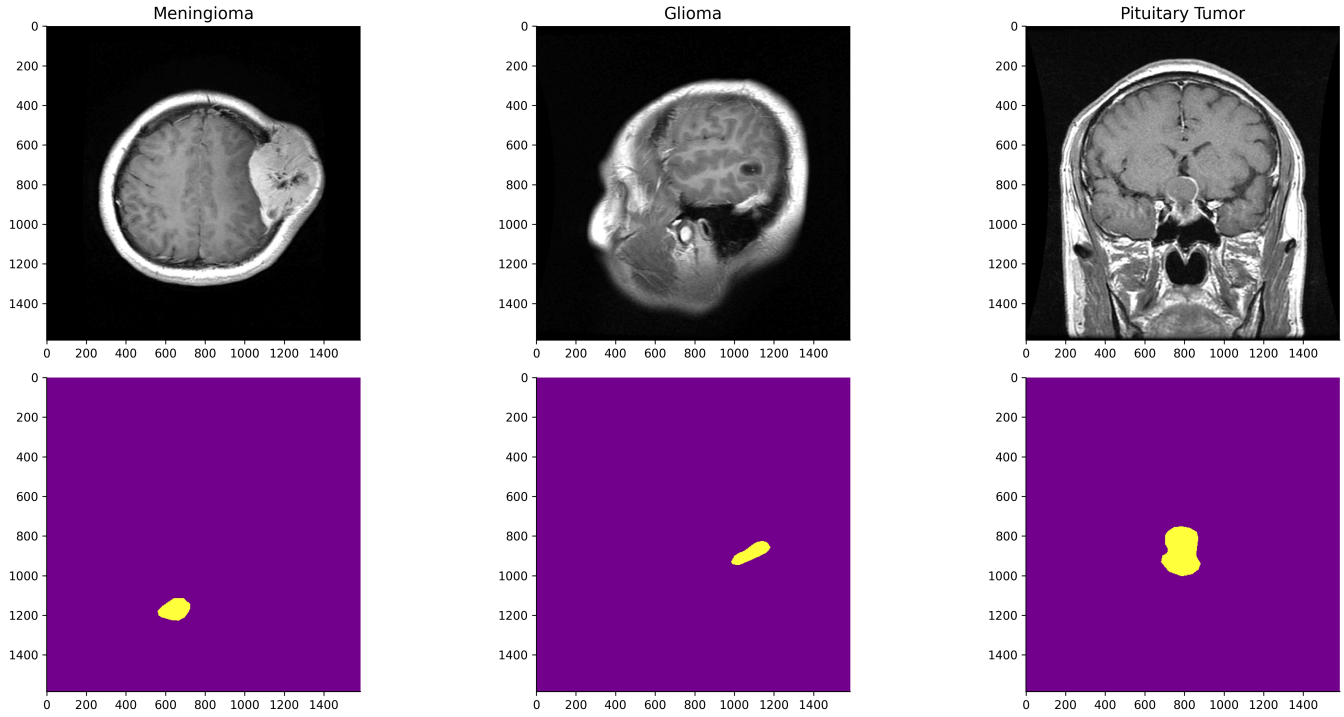
Dataset Source: https://figshare.com/articles/dataset/brain_tumor_dataset/1512427

Source Code for Conversion of `.mat` file to `.jpg`: Google Colab Notebook Link

Final Dataset Link: https://drive.google.com/drive/folders/11QIC82FBdAyq0PUwLVNd22i-oq6lcat1?usp=sharing

```
fig, axes = plt.subplots(nrows = 2, ncols = 3, figsize = (18, 9))
axes = axes.flatten()
fig.suptitle("/content/gdrive/MyDrive/Dataset/Extracted/Brain-Tumor-Images-Mat-Files",
             fontsize=16,
             fontweight='bold', # Changed 'weight' to 'fontweight'
             y=1.04)
for curr_title, filename, curr_axis in zip(TUMOR_CLASS[:-1], IMAGE_DATA_PATHS[:-1], axes[:3]):
    curr_image = Image.open(os.path.join(filename, os.listdir(filename)[2]))
    img_enhancer = ImageEnhance.Brightness(curr_image)
    curr_axis.imshow(img_enhancer.enhance(BRIGHTNESS_FACTOR))
    curr_axis.set_title(" ".join(curr_title.split('_')).title(), fontsize = 14)

for filename, curr_axis in zip(MASK_DATA_PATHS, axes[3:]):
    curr_image = Image.open(os.path.join(filename, os.listdir(filename)[2]))
    mask_enhancer = ImageEnhance.Brightness(curr_image)
    curr_axis.imshow(mask_enhancer.enhance(BRIGHTNESS_FACTOR))
fig.tight_layout()
sns.despine()
```

/content/gdrive/MyDrive/Dataset/Extracted/Brain-Tumor-Images-Mat-Files



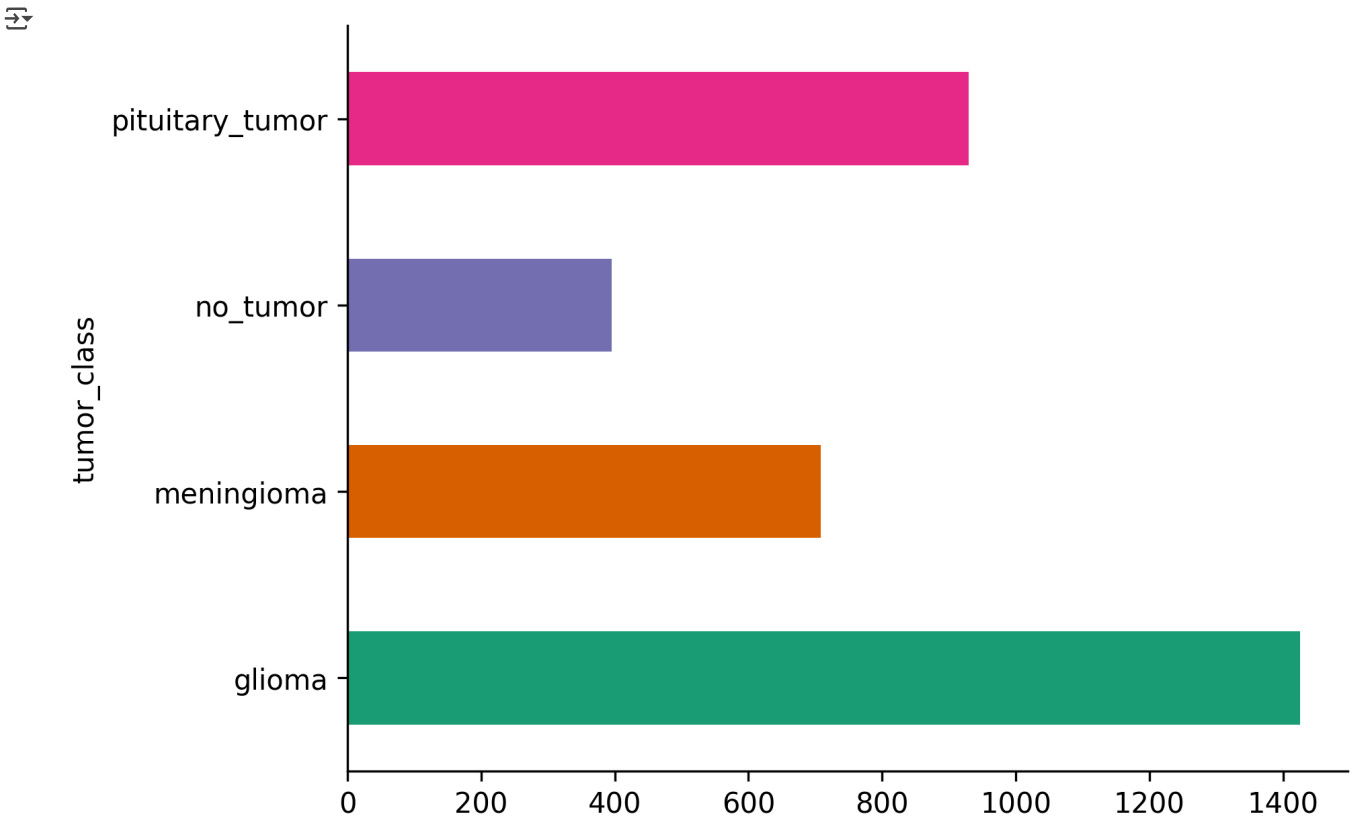
6. Development of Training, Validation & Testing Dataset

```
image_data_paths = []
for curr_path, tumor_name in zip(IMAGE_DATA_PATHS, TUMOR_CLASS):
    if os.path.exists(curr_path) and os.path.isdir(curr_path):
        image_data_paths.extend(map(lambda filename: (os.path.join(curr_path, filename), tumor_name), os.listdir(curr_path)))
```

image_data_paths_df = pd.DataFrame(image_data_paths, columns = ['image_filepaths', 'tumor_class']).sample(frac = 1, random_state = 42).reset_index(drop = True) image_data_paths_df.head()

tumor_class

Show code




```
image_data_paths_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3459 entries, 0 to 3458
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   image_filepaths  3459 non-null   object
 1   tumor_class      3459 non-null   object
dtypes: object(2)
memory usage: 54.2+ KB
```

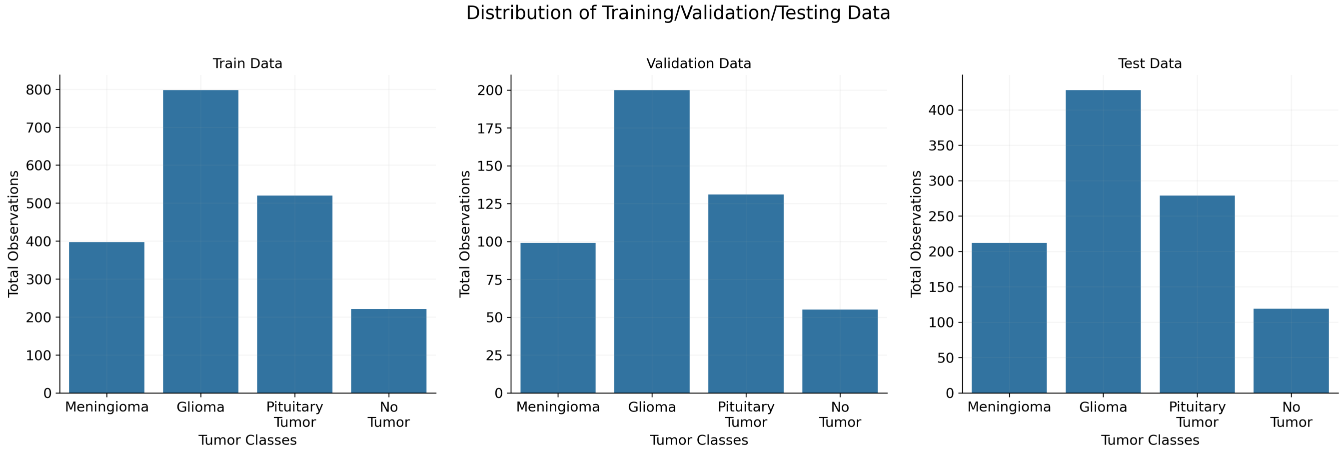
```
intermediate_train_data, test_data = train_test_split(image_data_paths_df,
                                                      train_size = 0.70,
                                                      random_state = 42,
                                                      stratify = image_data_paths_df.tumor_class)

train_data, validation_data = train_test_split(intermediate_train_data,
                                               train_size = 0.80,
                                               random_state = 42,
                                               stratify = intermediate_train_data.tumor_class)
```

6.1 Training, Validation and Testing Dataset Data Distribution Visualization

```
fig, axes = plt.subplots(ncols = 3, figsize = (20, 5))
# Removed the 'weight' key from the fontdict argument
fig.suptitle("Distribution of Training/Validation/Testing Data", fontsize = 16, fontdict = dict(), y = 1.05)
sns.countplot(x = train_data.tumor_class, order = TUMOR_CLASS, ax = axes[0])
sns.countplot(x = validation_data.tumor_class, order = TUMOR_CLASS, ax = axes[1])
sns.countplot(x = test_data.tumor_class, order = TUMOR_CLASS, ax = axes[2])
```

```
for curr_axis, curr_title in zip(axes, ['Train Data', 'Validation Data', 'Test Data']):
    curr_axis.grid(False, alpha = 0.1)
    curr_axis.set_title(curr_title, fontsize = 12)
    curr_axis.set_xlabel("Tumor Classes", fontsize = 12)
    curr_axis.set_ylabel("Total Observations", fontsize = 12)
    curr_axis.tick_params(which = 'major', labelsize = 12)
    curr_axis.set_xticklabels(["\n".join(xtick.split("_")).title() for xtick in TUMOR_CLASS])
sns.despine()
```

```
<ipython-input-37-450ec51165a4>:8: UserWarning: First parameter to grid() is false, but line properties are supplied. The grid
  curr_axis.grid(False, alpha = 0.1)
<ipython-input-37-450ec51165a4>:13: UserWarning: set_ticklabels() should only be used with a fixed number of ticks, i.e. after
  curr_axis.set_xticklabels(["\n".join(xtick.split("_")).title() for xtick in TUMOR_CLASS])
<ipython-input-37-450ec51165a4>:8: UserWarning: First parameter to grid() is false, but line properties are supplied. The grid
  curr_axis.grid(False, alpha = 0.1)
<ipython-input-37-450ec51165a4>:13: UserWarning: set_ticklabels() should only be used with a fixed number of ticks, i.e. after
  curr_axis.set_xticklabels(["\n".join(xtick.split("_")).title() for xtick in TUMOR_CLASS])
<ipython-input-37-450ec51165a4>:8: UserWarning: First parameter to grid() is false, but line properties are supplied. The grid
  curr_axis.grid(False, alpha = 0.1)
<ipython-input-37-450ec51165a4>:13: UserWarning: set_ticklabels() should only be used with a fixed number of ticks, i.e. after
  curr_axis.set_xticklabels(["\n".join(xtick.split("_")).title() for xtick in TUMOR_CLASS])
```



7. Data/Image Augmentation

- Image augmentation is usually used to increase the image dataset and also to make the network more robust against translation invariance. Image augmentation is defined as creating duplicates of the original image datasets by flipping, rotating, zooming, and adjusting brightness.
- We will use data/image augmentation using `ImageDataGenerator` class to train the model on different types of combinations formed by rotation, flipping, changing the brightness etc of an image so as to increase our model accuracy.

```
image_size = 128
batch_size = 32

image_datagen_kwargs = dict(rescale = 1 / 255,
                            rotation_range = 15,
                            width_shift_range = 0.1,
                            zoom_range = 0.01,
                            shear_range = 0.01,
                            brightness_range = [0.3, 1.5],
                            horizontal_flip = True,
                            vertical_flip = True)


train_image_datagen = ImageDataGenerator(**image_datagen_kwargs)
validation_image_datagen = ImageDataGenerator(**image_datagen_kwargs)
test_image_datagen = ImageDataGenerator(**image_datagen_kwargs)


train_dataset = train_image_datagen.flow_from_dataframe(train_data,
                                                        x_col = 'image_filepaths',
                                                        y_col = 'tumor_class',
                                                        seed = 42,
                                                        batch_size = batch_size,
                                                        target_size = (image_size, image_size),
                                                        color_mode = 'rgb')
```

```
validation_dataset = validation_image_datagen.flow_from_dataframe(validation_data,
                                                                  x_col = 'image_filepaths',
                                                                  y_col = 'tumor_class',
                                                                  seed = 42,
                                                                  batch_size = batch_size,
                                                                  target_size = (image_size, image_size),
                                                                  color_mode = 'rgb')
test_dataset = test_image_datagen.flow_from_dataframe(test_data,
                                                      x_col = 'image_filepaths',
                                                      y_col = 'tumor_class',
                                                      seed = 42,
                                                      batch_size = batch_size,
                                                      target_size = (image_size, image_size),
                                                      color_mode = 'rgb')
```

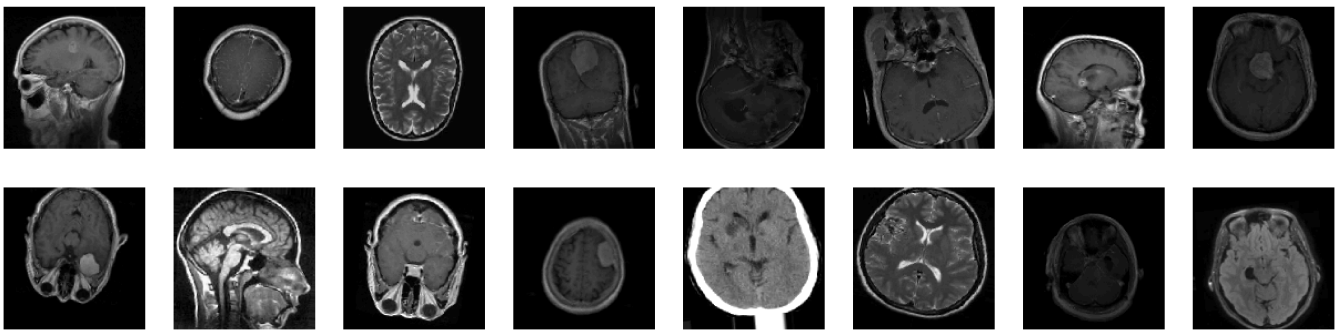
```
Found 1936 validated image filenames belonging to 4 classes.
Found 485 validated image filenames belonging to 4 classes.
Found 1038 validated image filenames belonging to 4 classes.
```

```
print("Information about Training Dataset:")
print(train_dataset.class_indices)
print(train_dataset.image_shape, end = '\n\n')

print("Information about Validation Dataset:")
print(validation_dataset.class_indices)
print(validation_dataset.image_shape, end = '\n\n')

print("Information about Testing Dataset:")
print(test_dataset.class_indices)
print(test_dataset.image_shape)
```

```
Information about Training Dataset:
{'glioma': 0, 'meningioma': 1, 'no_tumor': 2, 'pituitary_tumor': 3}
(128, 128, 3)

Information about Validation Dataset:
{'glioma': 0, 'meningioma': 1, 'no_tumor': 2, 'pituitary_tumor': 3}
(128, 128, 3)

Information about Testing Dataset:
{'glioma': 0, 'meningioma': 1, 'no_tumor': 2, 'pituitary_tumor': 3}
(128, 128, 3)
```

7.1 Training Data Images Glimpse

```
fig, axes = plt.subplots(nrows = 2, ncols = 8, figsize = (20, 5))
fig.suptitle("Distribution of Training/Validation/Testing Data", fontsize = 16, fontdict = dict(), y = 1.05)
for curr_axis, curr_image in zip(axes.flatten(), train_dataset[0][0][:16]):
    curr_axis.imshow(tf.squeeze(curr_image), cmap = 'gray')
    curr_axis.axis(False)
```



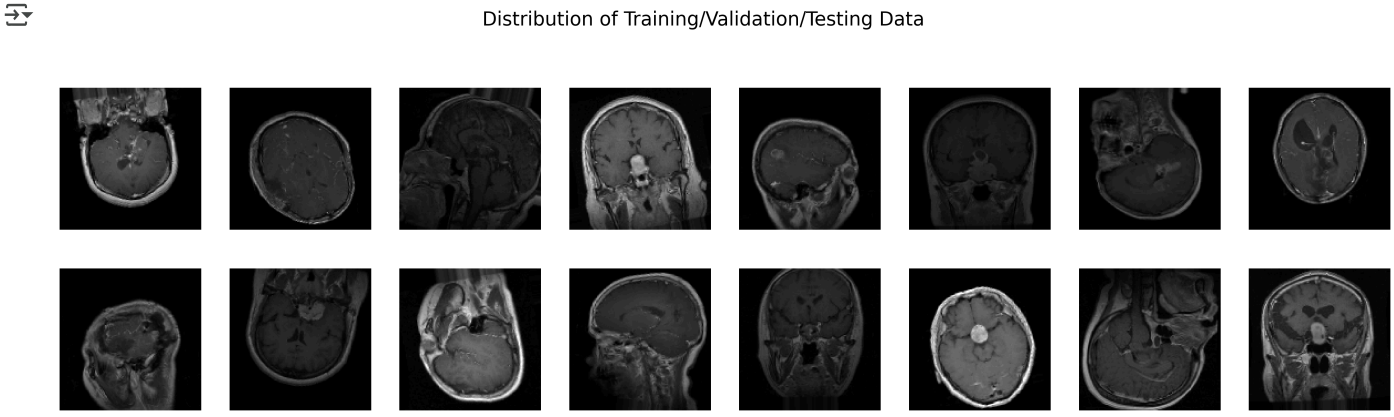
7.2 Validation Data Images Glimpse

```
fig, axes = plt.subplots(nrows = 2, ncols = 8, figsize = (20, 5))
fig.suptitle("Distribution of Training/Validation/Testing Data", fontsize = 16, fontdict = dict(), y = 1.05)
for curr_axis, curr_image in zip(axes.flatten(), validation_dataset[0][0][:16]):
```

```
    curr_axis.imshow(tf.squeeze(curr_image), cmap = 'gray')
    curr_axis.axis(False)
```

7.3 Testing Data Images Glimpse

```
fig, axes = plt.subplots(nrows = 2, ncols = 8, figsize = (20, 5))
fig.suptitle("Distribution of Training/Validation/Testing Data", fontsize = 16, fontdict = dict(), y = 1.05)
for curr_axis, curr_image in zip(axes.flatten(), test_dataset[0][0][:16]):
    curr_axis.imshow(tf.squeeze(curr_image), cmap = 'gray')
    curr_axis.axis(False)
```



8. Model Development

```
early_stopping = EarlyStopping(monitor = 'val_accuracy', patience = 10)
```

```
MLP_CHECKPOINT_DIR_PATH = "/content/gdrive/MyDrive/Dataset/Extracted/Model-Checkpoints/Multi-Layer-Perceptron"
ALEXNET_CHECKPOINT_DIR_PATH = "/content/gdrive/MyDrive/Dataset/Extracted/Model-Checkpoints/AlexNet-CNN"
INCEPTIONV3_CHECKPOINT_DIR_PATH = "/content/gdrive/MyDrive/Dataset/Extracted/Model-Checkpoints/InceptionV3"
UNET_CHECKPOINT_DIR_PATH = "/content/gdrive/MyDrive/Dataset/Extracted/Model-Checkpoints/Unet"
```

```
import os

ROOT_CHECKPOINT_DIR_PATH = os.path.join(ROOT_DIR, "Model-Checkpoints")
MLP_CHECKPOINT_DIR_PATH = os.path.join(ROOT_CHECKPOINT_DIR_PATH, "Multi-Layer-Perceptron")
ALEXNET_CHECKPOINT_DIR_PATH = os.path.join(ROOT_CHECKPOINT_DIR_PATH, "AlexNet-CNN")
INCEPTIONV3_CHECKPOINT_DIR_PATH = os.path.join(ROOT_CHECKPOINT_DIR_PATH, "InceptionV3")
UNET_CHECKPOINT_DIR_PATH = os.path.join(ROOT_CHECKPOINT_DIR_PATH, "Unet")

# Create the directories if they don't exist
os.makedirs(ROOT_CHECKPOINT_DIR_PATH, exist_ok=True)
os.makedirs(MLP_CHECKPOINT_DIR_PATH, exist_ok=True)
os.makedirs(ALEXNET_CHECKPOINT_DIR_PATH, exist_ok=True)
```

```
os.makedirs(INCEPTIONV3_CHECKPOINT_DIR_PATH, exist_ok=True)

# The assertion should now pass
assert os.path.isdir(ROOT_CHECKPOINT_DIR_PATH) and os.path.isdir(MLP_CHECKPOINT_DIR_PATH) and os.path.isdir(ALEXNET_CHECKPOINT_DIR


mlp_cp_callback = ModelCheckpoint(os.path.join(MLP_CHECKPOINT_DIR_PATH, 'mlp_weights.keras'),
                                  monitor='val_accuracy',
                                  verbose=1,
                                  save_weights_only=False,  # Change to False to save the entire model
                                  save_freq='epoch')

alexnet_cp_callback = ModelCheckpoint(os.path.join(ALEXNET_CHECKPOINT_DIR_PATH, 'alexnet_weights.keras'),
                                      monitor='val_accuracy',
                                      verbose=1,
                                      save_weights_only=False,  # Change to False to save the entire model
                                      save_freq='epoch')

inceptionv3_cp_callback = ModelCheckpoint(os.path.join(INCEPTIONV3_CHECKPOINT_DIR_PATH, 'inceptionv3_weights.keras'),
                                          monitor='val_accuracy',
                                          verbose=1,
                                          save_weights_only=False,  # Change to False to save the entire model
                                          save_freq='epoch')

Unet_cp_callback = ModelCheckpoint(os.path.join(UNET_CHECKPOINT_DIR_PATH, 'Unet.keras'),
                                          monitor='val_accuracy',
                                          verbose=1,
                                          save_weights_only=False,  # Change to False to save the entire model
                                          save_freq='epoch')



def training_process_viz(training_stats: pd.DataFrame, **plot_kwargs) -> None:
    fig, axes = plt.subplots(ncols = 2, figsize = (15, 5))
    fig.suptitle(plot_kwargs['plot_title'], fontsize = 16, fontdict = dict(weight = 'bold'), y = 1.08)
    for curr_axis, col_name in zip(axes, ['accuracy', 'loss']):
        curr_axis.grid(True, alpha = 0.3)
        curr_axis.set_title(f"Model {col_name}".title(), fontsize = 14)
        sns.lineplot(x = range(1, 1 + training_stats.shape[0]), y = training_stats[col_name], color = 'blue', ax = curr_axis)
        sns.lineplot(x = range(1, 1 + training_stats.shape[0]), y = training_stats[f"val_{col_name}"], color = 'red', ax = curr_ax
        curr_axis.set_xlabel("Epochs", fontsize = 12)
        curr_axis.set_ylabel(col_name.title(), fontsize = 12)
        curr_axis.tick_params(which = 'major', labelsize = 12)
        curr_axis.legend([col_name.title(), f"validation {col_name}".title()], title = col_name.title())
    fig.tight_layout()
    sns.despine()


def confusion_matrix_viz(model, test_dataset, **plot_kwargs) -> None:
    assert isinstance(model, Sequential)
    model_preds = [np.argmax(curr_row) for curr_row in model.predict(test_dataset)]
    fig, axis = plt.subplots(figsize = (8, 6))
    class_names = ['Glioma', 'Meningioma', 'No-Tumor', 'Pituitary\nTumor']
    sns.heatmap(confusion_matrix(test_dataset.classes, model_preds), annot = True, cmap = plt.cm.Reds, ax = axis)
    axis.set_title(plot_kwargs['plot_title'], fontsize = 14)
    axis.tick_params(which = 'major', labelsize = 12)
    axis.set_xlabel("Pedicted Class", fontsize = 12)
    axis.set_ylabel("Actual Class", fontsize = 12)
    axis.set_xticklabels(class_names, fontdict = dict(fontsize = 12))
    axis.set_yticklabels(class_names, fontdict = dict(fontsize = 12))
    fig.tight_layout()
    sns.despine()


def generate_report(*models, test_dataset, row_indexes) -> pd.DataFrame:
    assert len(models)
    report_df = pd.DataFrame(columns = ['MAE', 'MSE', 'RMSE', 'Loss', 'Accuracy', 'F1-Score'])
    y_hat = test_dataset.classes # y_hat = ground_truth
    for curr_index, curr_model in enumerate(models):
        assert isinstance(curr_model, Sequential)
        curr_model_loss, curr_model_accuracy = curr_model.evaluate(test_dataset)
        y_preds = [np.argmax(curr_preds) for curr_preds in curr_model.predict(test_dataset)]
        report_df.loc[curr_index] = [mean_absolute_error(y_hat, y_preds), mean_squared_error(y_hat, y_preds), mean_squared_error(y
    report_df.index = row_indexes
    return report_df
```

8.1 Multi-Layer Perceptron

8.1.1 Development of Multi-Layer Perceptron Model

```
mlp_model = Sequential()
mlp_model.add(Flatten(input_shape = (image_size, image_size, 3), name = 'Flatten-Layer'))
mlp_model.add(Dense(2048, activation = 'relu', name = 'Hidden-Layer-1'))
mlp_model.add(Dropout(rate = 0.2, name = 'Dropout-Layer-1'))
mlp_model.add(Dense(1024, activation = 'relu', name = 'Hidden-Layer-2'))
mlp_model.add(Dropout(rate = 0.2, name = 'Dropout-Layer-2'))
mlp_model.add(Dense(512, activation = 'relu', name = 'Hidden-Layer-3'))
mlp_model.add(Dropout(rate = 0.2, name = 'Dropout-Layer-3'))
mlp_model.add(Dense(4, activation = 'softmax', name = 'Output-Layer-1'))
mlp_model.compile(optimizer = 'Adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
mlp_model.summary()
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
Flatten-Layer (Flatten)	(None, 49152)	0
Hidden-Layer-1 (Dense)	(None, 2048)	100,665,344
Dropout-Layer-1 (Dropout)	(None, 2048)	0
Hidden-Layer-2 (Dense)	(None, 1024)	2,098,176
Dropout-Layer-2 (Dropout)	(None, 1024)	0
Hidden-Layer-3 (Dense)	(None, 512)	524,800
Dropout-Layer-3 (Dropout)	(None, 512)	0
Output-Layer-1 (Dense)	(None, 4)	2,052

Total params: 103,290,372 (394.02 MB)
Trainable params: 103,290,372 (394.02 MB)

8.1.2 Training and Validation of Multi-Layer Perceptron Based Model

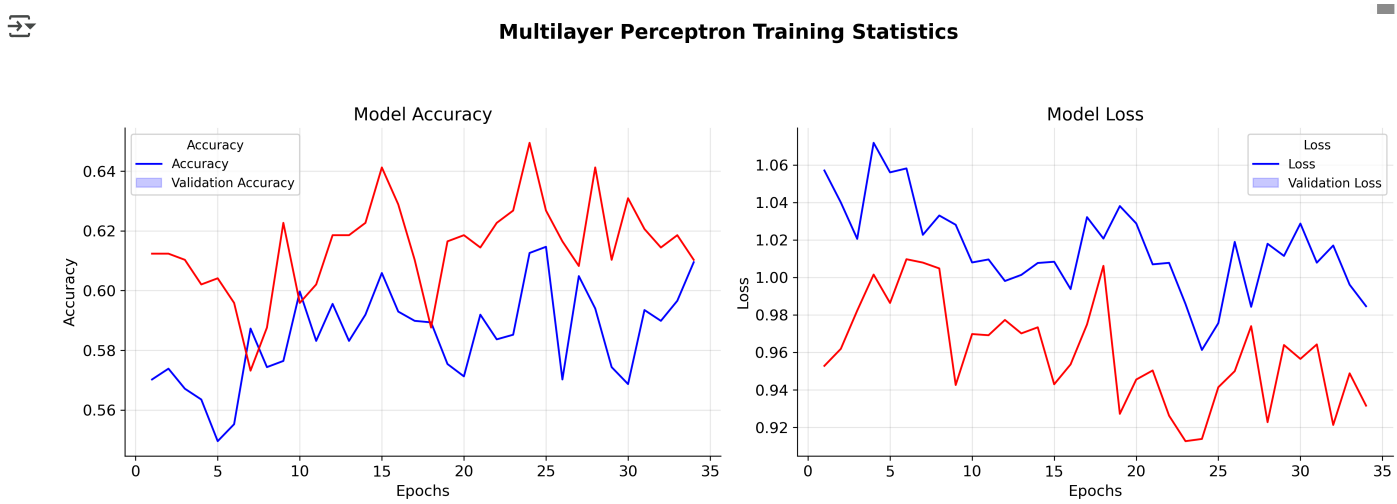
```
with tf.device(gpu_device_location) if gpu_device_location else tpu_strategy.scope() if tpu_device_location else tf.device(cpu_dev:
    mlp_train_history = mlp_model.fit(train_dataset,
                                      batch_size = batch_size,
                                      validation_data = validation_dataset,
                                      epochs = 100,
                                      callbacks = [early_stopping])
```

```
Epoch 1/100
61/61 ━━━━━━━━━━━━━━━━━━━━ 50s 823ms/step - accuracy: 0.5554 - loss: 1.1079 - val_accuracy: 0.6124 - val_loss: 0.9528
Epoch 2/100
61/61 ━━━━━━━━━━━━━━━━━━━━ 45s 748ms/step - accuracy: 0.5662 - loss: 1.0509 - val_accuracy: 0.6124 - val_loss: 0.9619
Epoch 3/100
61/61 ━━━━━━━━━━━━━━━━━━━━ 59s 981ms/step - accuracy: 0.5798 - loss: 1.0143 - val_accuracy: 0.6103 - val_loss: 0.9823
Epoch 4/100
61/61 ━━━━━━━━━━━━━━━━━━━━ 81s 950ms/step - accuracy: 0.5714 - loss: 1.0580 - val_accuracy: 0.6021 - val_loss: 1.0015
Epoch 5/100
61/61 ━━━━━━━━━━━━━━━━━━━━ 49s 810ms/step - accuracy: 0.5286 - loss: 1.0766 - val_accuracy: 0.6041 - val_loss: 0.9864
Epoch 6/100
61/61 ━━━━━━━━━━━━━━━━━━━━ 66s 1s/step - accuracy: 0.5289 - loss: 1.0873 - val_accuracy: 0.5959 - val_loss: 1.0097
Epoch 7/100
61/61 ━━━━━━━━━━━━━━━━━━━━ 49s 807ms/step - accuracy: 0.5693 - loss: 1.0528 - val_accuracy: 0.5732 - val_loss: 1.0079
Epoch 8/100
61/61 ━━━━━━━━━━━━━━━━━━━━ 52s 849ms/step - accuracy: 0.5642 - loss: 1.0495 - val_accuracy: 0.5876 - val_loss: 1.0048
Epoch 9/100
61/61 ━━━━━━━━━━━━━━━━━━━━ 47s 781ms/step - accuracy: 0.5758 - loss: 1.0504 - val_accuracy: 0.6227 - val_loss: 0.9426
Epoch 10/100
61/61 ━━━━━━━━━━━━━━━━━━━━ 48s 792ms/step - accuracy: 0.6101 - loss: 1.0070 - val_accuracy: 0.5959 - val_loss: 0.9698
Epoch 11/100
61/61 ━━━━━━━━━━━━━━━━━━━━ 59s 976ms/step - accuracy: 0.5848 - loss: 1.0072 - val_accuracy: 0.6021 - val_loss: 0.9692
Epoch 12/100
61/61 ━━━━━━━━━━━━━━━━━━━━ 51s 837ms/step - accuracy: 0.5890 - loss: 1.0281 - val_accuracy: 0.6186 - val_loss: 0.9773
Epoch 13/100
61/61 ━━━━━━━━━━━━━━━━━━━━ 62s 1s/step - accuracy: 0.5831 - loss: 1.0106 - val_accuracy: 0.6186 - val_loss: 0.9701
Epoch 14/100
61/61 ━━━━━━━━━━━━━━━━━━━━ 45s 735ms/step - accuracy: 0.5893 - loss: 1.0170 - val_accuracy: 0.6227 - val_loss: 0.9734
Epoch 15/100
61/61 ━━━━━━━━━━━━━━━━━━━━ 46s 762ms/step - accuracy: 0.5686 - loss: 1.0660 - val_accuracy: 0.6412 - val_loss: 0.9430
Epoch 16/100
61/61 ━━━━━━━━━━━━━━━━━━━━ 82s 763ms/step - accuracy: 0.6175 - loss: 0.9668 - val_accuracy: 0.6289 - val_loss: 0.9536
Epoch 17/100
61/61 ━━━━━━━━━━━━━━━━━━━━ 81s 747ms/step - accuracy: 0.5985 - loss: 1.0291 - val_accuracy: 0.6103 - val_loss: 0.9749
Epoch 18/100
61/61 ━━━━━━━━━━━━━━━━━━━━ 47s 774ms/step - accuracy: 0.5963 - loss: 1.0199 - val_accuracy: 0.5876 - val_loss: 1.0062
Epoch 19/100
61/61 ━━━━━━━━━━━━━━━━━━━━ 49s 807ms/step - accuracy: 0.5695 - loss: 1.0560 - val_accuracy: 0.6165 - val_loss: 0.9272
Epoch 20/100
61/61 ━━━━━━━━━━━━━━━━━━━━ 80s 782ms/step - accuracy: 0.5818 - loss: 1.0193 - val_accuracy: 0.6186 - val_loss: 0.9455
Epoch 21/100
61/61 ━━━━━━━━━━━━━━━━━━━━ 46s 749ms/step - accuracy: 0.5774 - loss: 1.0237 - val_accuracy: 0.6144 - val_loss: 0.9504
Epoch 22/100
```

```
61/61 ━━━━━━━━━━━━━━━━━━━━ 56s 931ms/step - accuracy: 0.5844 - loss: 1.0047 - val_accuracy: 0.6227 - val_loss: 0.9262
Epoch 23/100
61/61 ━━━━━━━━━━━━━━━━━━━━ 45s 747ms/step - accuracy: 0.5713 - loss: 1.0120 - val_accuracy: 0.6268 - val_loss: 0.9127
Epoch 24/100
61/61 ━━━━━━━━━━━━━━━━━━━━ 47s 770ms/step - accuracy: 0.6184 - loss: 0.9566 - val_accuracy: 0.6495 - val_loss: 0.9138
Epoch 25/100
61/61 ━━━━━━━━━━━━━━━━━━━━ 54s 893ms/step - accuracy: 0.6118 - loss: 0.9694 - val_accuracy: 0.6268 - val_loss: 0.9414
Epoch 26/100
61/61 ━━━━━━━━━━━━━━━━━━━━ 46s 763ms/step - accuracy: 0.5639 - loss: 1.0177 - val_accuracy: 0.6165 - val_loss: 0.9500
Epoch 27/100
61/61 ━━━━━━━━━━━━━━━━━━━━ 59s 973ms/step - accuracy: 0.5923 - loss: 1.0126 - val_accuracy: 0.6082 - val_loss: 0.9740
Epoch 28/100
61/61 ━━━━━━━━━━━━━━━━━━━━ 45s 745ms/step - accuracy: 0.5951 - loss: 1.0106 - val_accuracy: 0.6412 - val_loss: 0.9228
Epoch 29/100
61/61 ━━━━━━━━━━━━━━━━━━━━ 49s 795ms/step - accuracy: 0.5799 - loss: 1.0112 - val_accuracy: 0.6103 - val_loss: 0.9639
```

```
def training_process_viz(training_stats: pd.DataFrame, **plot_kwargs) -> None:
    fig, axes = plt.subplots(ncols = 2, figsize = (15, 5))
    # Use 'fontweight' instead of 'weight' in the fontdict argument
    fig.suptitle(plot_kwargs['plot_title'], fontsize = 16, fontweight='bold', y = 1.08)
    for curr_axis, col_name in zip(axes, ['accuracy', 'loss']):
        curr_axis.grid(True, alpha = 0.3)
        curr_axis.set_title(f"Model {col_name}".title(), fontsize = 14)
        sns.lineplot(x = range(1, 1 + training_stats.shape[0]), y = training_stats[col_name], color = 'blue', ax = curr_axis)
        sns.lineplot(x = range(1, 1 + training_stats.shape[0]), y = training_stats[f"val_{col_name}"], color = 'red', ax = curr_ax
        curr_axis.set_xlabel("Epochs", fontsize = 12)
        curr_axis.set_ylabel(col_name.title(), fontsize = 12)
        curr_axis.tick_params(which = 'major', labelsize = 12)
        curr_axis.legend([col_name.title(), f"validation {col_name}".title()], title = col_name.title())
    fig.tight_layout()
    sns.despine()
```

```
training_process_viz(pd.DataFrame(mlp_train_history.history),
                     plot_title = 'Multilayer Perceptron Training Statistics')
```

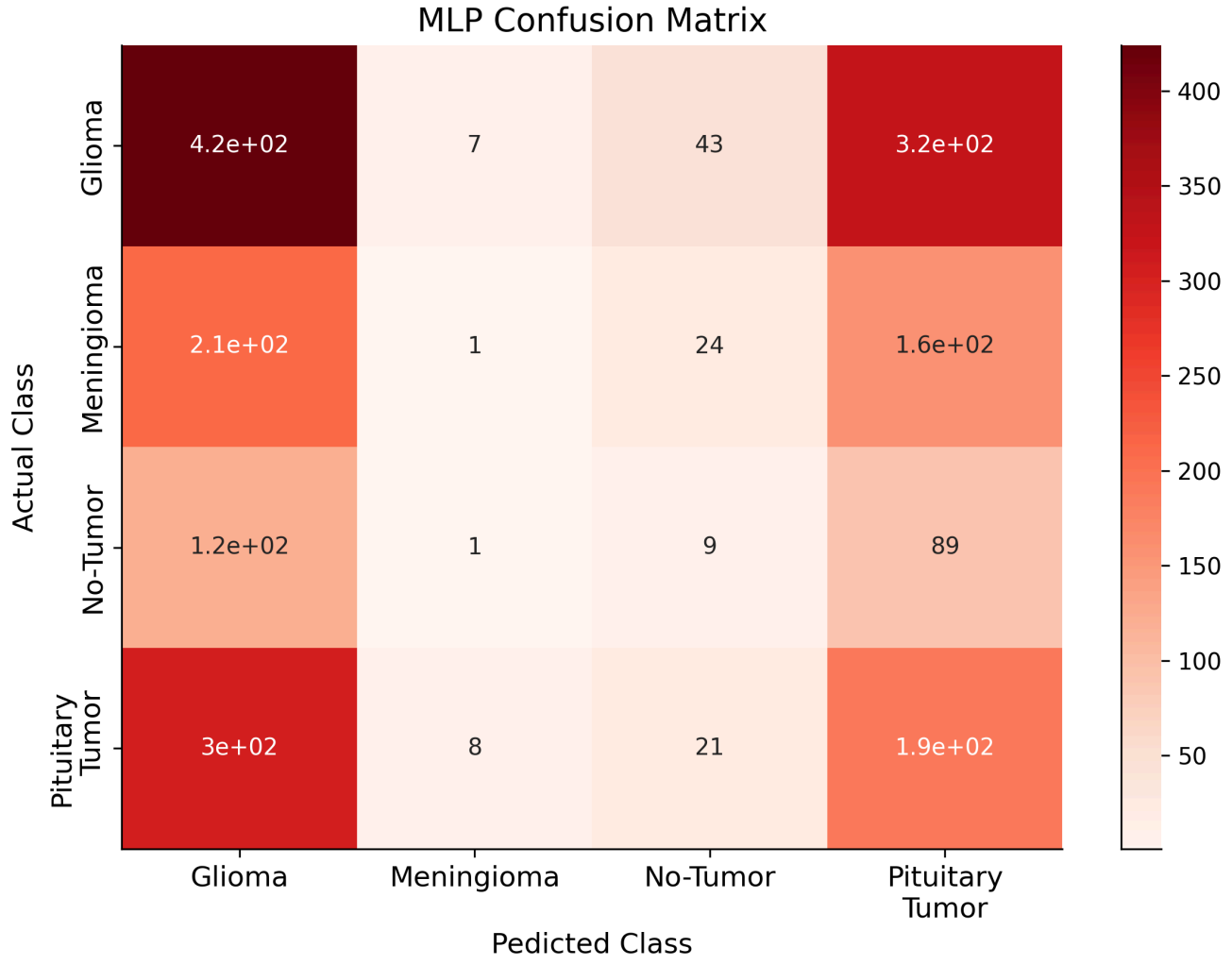


8.1.3 Multi-Layer Perceptron Based Model Training Process Statistics

8.1.4 Confusion Matrix for Multi-Layer Perceptron Based Model

```
confusion_matrix_viz(mlp_model, train_dataset, plot_title = "MLP Confusion Matrix")
```

```
61/61 ━━━━━━━━━━━━━━━━━━━━ 36s 584ms/step
```




```
def generate_report(*models, test_dataset, row_indexes) -> pd.DataFrame:
    assert len(models)
    report_df = pd.DataFrame(columns = ['MAE', 'MSE', 'RMSE', 'Loss', 'Accuracy', 'F1-Score'])
    y_hat = test_dataset.classes # y_hat = ground_truth
    for curr_index, curr_model in enumerate(models):
        assert isinstance(curr_model, Sequential)
        curr_model_loss, curr_model_accuracy = curr_model.evaluate(test_dataset)
        y_preds = [np.argmax(curr_preds) for curr_preds in curr_model.predict(test_dataset)]
        # Calculate RMSE separately, as mean_squared_error with squared=False might not be supported
        rmse = np.sqrt(mean_squared_error(y_hat, y_preds))
        report_df.loc[curr_index] = [mean_absolute_error(y_hat, y_preds), mean_squared_error(y_hat, y_preds), rmse, curr_model_los
    report_df.index = row_indexes
    return report_df

mlp_report_df = generate_report(mlp_model,
                                test_dataset = test_dataset,
                                row_indexes = ("Multi-Layer-Perceptron Model",))
mlp_report_df
```

Show hidden output

8.2 AlexNet CNN

8.2.1 Develoment of AlexNet CNN Model

```
alexnet_cnn = Sequential()
alexnet_cnn.add(Conv2D(96, kernel_size = 11, strides = 4, activation = 'relu', input_shape = (image_size, image_size, 3), name = '(
alexnet_cnn.add(BatchNormalization(name = 'Batch-Normalization-1'))
alexnet_cnn.add(MaxPool2D(pool_size = 3, strides = 2, name = 'Max-Pooling-1'))
alexnet_cnn.add(Conv2D(256, kernel_size = 5, padding = 'same', activation = 'relu', name = 'Conv2D-2'))
alexnet_cnn.add(BatchNormalization(name = 'Batch-Normalization-2'))
alexnet_cnn.add(MaxPool2D(pool_size = 3, strides = 2, name = 'Max-Pooling-2'))
alexnet_cnn.add(Conv2D(384, kernel_size = 3, padding = 'same', activation = 'relu', name = 'Conv2D-3'))
alexnet_cnn.add(BatchNormalization(name = 'Batch-Normalization-3'))
alexnet_cnn.add(Conv2D(384, kernel_size = 3, padding = 'same', activation = 'relu', name = 'Conv2D-4'))
alexnet_cnn.add(BatchNormalization(name = 'Batch-Normalization-4'))
alexnet_cnn.add(Conv2D(256, kernel_size = 3, padding = 'same', activation = 'relu', name = 'Conv2D-5'))
alexnet_cnn.add(BatchNormalization(name = 'Batch-Normalization-5'))
alexnet_cnn.add(MaxPool2D(pool_size = 3, strides = 2, name = 'Max-Pooling-3'))
alexnet_cnn.add(Flatten(name = 'Flatten-Layer-1'))
alexnet_cnn.add(Dense(128, activation = 'relu', name = 'Hidden-Layer-1'))
alexnet_cnn.add(Dropout(rate = 0.5, name = 'Dropout-Layer-1'))
alexnet_cnn.add(Dense(64, activation = 'relu', name = 'Hidden-Layer-2'))
alexnet_cnn.add(Dropout(rate = 0.5, name = 'Dropout-Layer-2'))
alexnet_cnn.add(Dense(4, activation = 'softmax', name = 'Output-Layer'))
alexnet_cnn.compile(optimizer = 'Adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
alexnet_cnn.summary()
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `input_sha
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

Model: "sequential_2"

Layer (type)	Output Shape	Param #
Conv2D-1 (Conv2D)	(None, 30, 30, 96)	34,944
Batch-Normalization-1 (BatchNormalization)	(None, 30, 30, 96)	384
Max-Pooling-1 (MaxPooling2D)	(None, 14, 14, 96)	0
Conv2D-2 (Conv2D)	(None, 14, 14, 256)	614,656
Batch-Normalization-2 (BatchNormalization)	(None, 14, 14, 256)	1,024
Max-Pooling-2 (MaxPooling2D)	(None, 6, 6, 256)	0
Conv2D-3 (Conv2D)	(None, 6, 6, 384)	885,120
Batch-Normalization-3 (BatchNormalization)	(None, 6, 6, 384)	1,536
Conv2D-4 (Conv2D)	(None, 6, 6, 384)	1,327,488
Batch-Normalization-4 (BatchNormalization)	(None, 6, 6, 384)	1,536
Conv2D-5 (Conv2D)	(None, 6, 6, 256)	884,992
Batch-Normalization-5 (BatchNormalization)	(None, 6, 6, 256)	1,024
Max-Pooling-3 (MaxPooling2D)	(None, 2, 2, 256)	0
Flatten-Layer-1 (Flatten)	(None, 1024)	0
Hidden-Layer-1 (Dense)	(None, 128)	131,200
Dropout-Layer-1 (Dropout)	(None, 128)	0
Hidden-Layer-2 (Dense)	(None, 64)	8,256
Dropout-Layer-2 (Dropout)	(None, 64)	0
Output-Layer (Dense)	(None, 4)	260

Total params: 3,892,420 (14.85 MB)

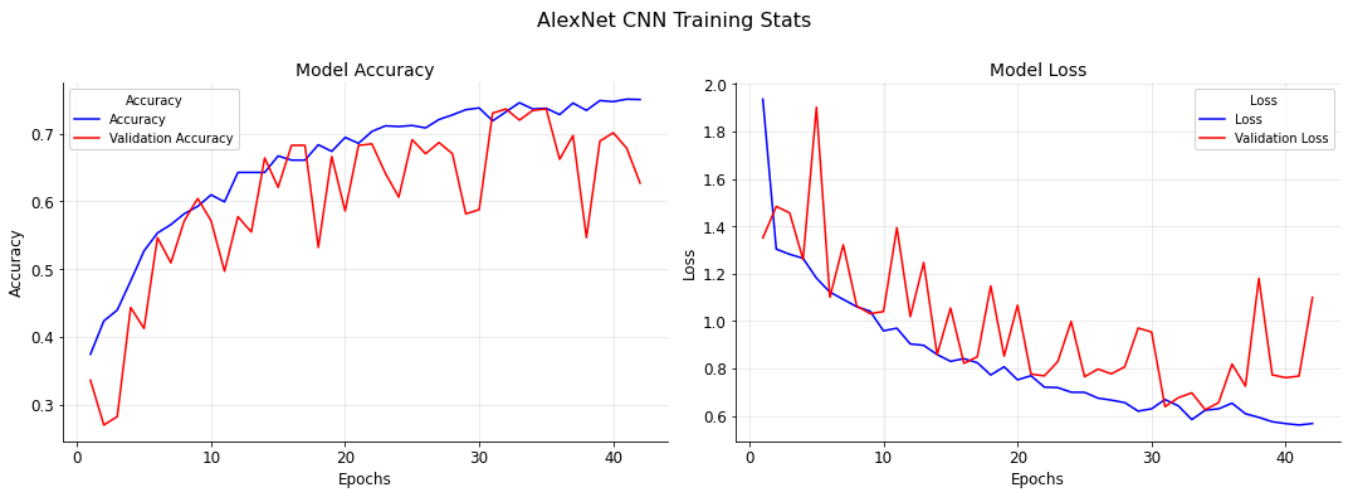
8.2.2 Training and Validation of AlexNet CNN Model

```
with tf.device(gpu_device_location) if gpu_device_location else tpu_strategy.scope() if tpu_device_location else tf.device(cpu_dev
    alexnet_train_history = alexnet_cnn.fit(train_dataset,
                                            batch_size = batch_size,
                                            validation_data = validation_dataset,
                                            epochs = 100,
                                            callbacks = [early_stopping, alexnet_cp_callback])
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-3-55f35f090c88> in <cell line: 0>()
----> 1 with tf.device(gpu_device_location) if gpu_device_location else tpu_strategy.scope() if tpu_device_location else
tf.device(cpu_device_location):
      2     alexnet_train_history = alexnet_cnn.fit(train_dataset,
      3                                             batch_size = batch_size,
      4                                             validation_data = validation_dataset,
      5                                             epochs = 100,

NameError: name 'gpu_device_location' is not defined
```

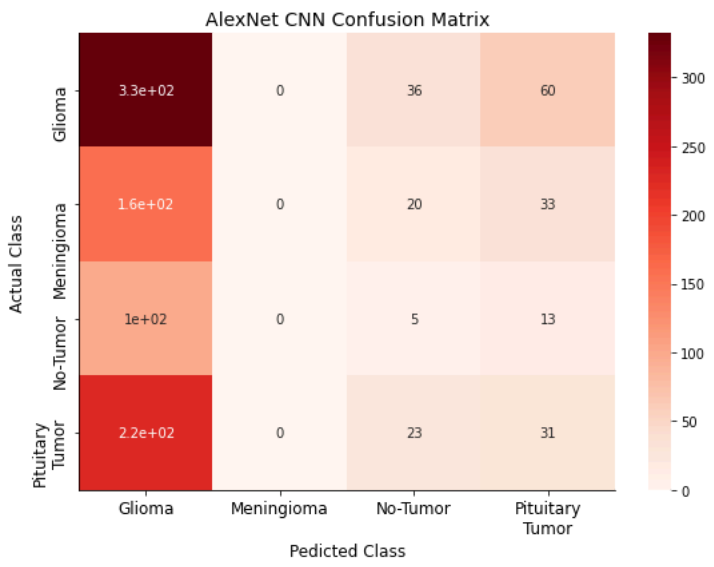
8.2.3 AlexNet CNN Model Training Process Statistics

```
training_process_viz(pd.DataFrame(alexnet_train_history.history), plot_title = 'AlexNet CNN Training Stats')
```



8.2.4 Confusion Matrix for AlexNet CNN Model

```
with tf.device(gpu_device_location) if gpu_device_location else tpu_strategy.scope() if tpu_device_name else tf.device(cpu_device_
    confusion_matrix_viz(alexnet_cnn,
                         test_dataset,
                         plot_title = "AlexNet CNN Confusion Matrix")
```




```
alexnet_report_df = generate_report(alexnet_cnn, test_dataset = test_dataset, row_indexes = ['AlexNet CNN'])
alexnet_report_df
```

```
33/33 [==============================] - 33s 988ms/step - loss: 1.1357 - accuracy: 0.6079
```

	MAE	MSE	RMSE	Loss	Accuracy	F1-Score
AlexNet CNN	1.301541	3.205202	1.790308	1.135692	0.6079	0.382466

8.3 Inception V3

8.3.1 Developement of InceptionV3

```
inception_v3_model = InceptionV3(include_top = False,
                                 input_shape = (image_size, image_size, 3),
                                 pooling = 'avg')
inception_v3_model.trainable = False
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/inception_v3/inception_v3_weights_tf_dim_or
87916544/87910968 [==============================] - 1s 0us/step
```

```
inception_cnn_model = Sequential()
inception_cnn_model.add(inception_v3_model)
inception_cnn_model.add(Flatten())
inception_cnn_model.add(Dense(1024, activation = 'relu', name = 'Hidden-Layer-1'))
inception_cnn_model.add(Dense(4, activation = 'softmax', name = 'Output-Layer'))
inception_cnn_model.compile(optimizer = 'Adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
inception_cnn_model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
inception_v3 (Functional)    (None, 2048)              21802784
_________________________________________________________________
flatten (Flatten)            (None, 2048)              0
_________________________________________________________________
Hidden-Layer-1 (Dense)       (None, 1024)              2098176
_________________________________________________________________
Output-Layer (Dense)         (None, 4)                 4100
=================================================================
Total params: 23,905,060
Trainable params: 2,102,276
Non-trainable params: 21,802,784
_________________________________________________________________
```

8.3.2 Training and Validation of InceptionV3 Model

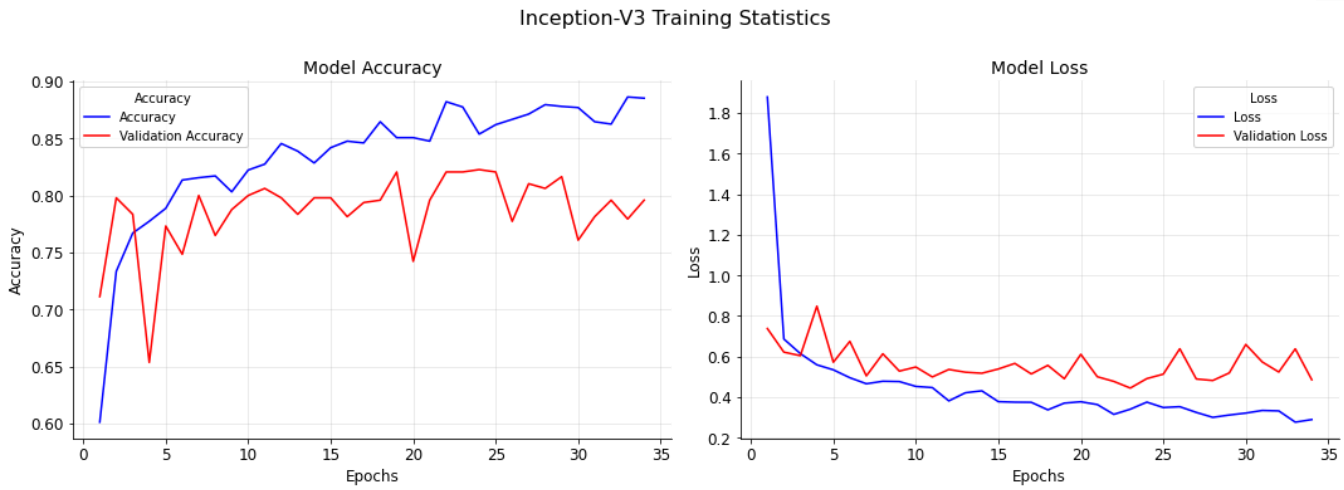
```
with tf.device(gpu_device_location) if gpu_device_location else tpu_strategy.scope() if tpu_device_location else tf.device(cpu_dev
    inception_model_train_history = inception_cnn_model.fit(train_dataset,
                                                            batch_size = batch_size,
                                                            validation_data = validation_dataset,
                                                            epochs = 100,
                                                            callbacks = [early_stopping, inceptionv3_cp_callback])
```

```
Epoch 1/100
61/61 [==============================] - 83s 1s/step - loss: 1.8779 - accuracy: 0.6012 - val_loss: 0.7379 - val_accuracy: 0.

Epoch 00001: saving model to gdrive/MyDrive/Deep_Learning_Course_Project/Model-Checkpoints/InceptionV3
Epoch 2/100
61/61 [==============================] - 76s 1s/step - loss: 0.6863 - accuracy: 0.7335 - val_loss: 0.6218 - val_accuracy: 0.

Epoch 00002: saving model to gdrive/MyDrive/Deep_Learning_Course_Project/Model-Checkpoints/InceptionV3
Epoch 3/100
61/61 [==============================] - 76s 1s/step - loss: 0.6144 - accuracy: 0.7670 - val_loss: 0.6042 - val_accuracy: 0.

Epoch 00003: saving model to gdrive/MyDrive/Deep_Learning_Course_Project/Model-Checkpoints/InceptionV3
Epoch 4/100
61/61 [==============================] - 76s 1s/step - loss: 0.5593 - accuracy: 0.7774 - val_loss: 0.8480 - val_accuracy: 0.

Epoch 00004: saving model to gdrive/MyDrive/Deep_Learning_Course_Project/Model-Checkpoints/InceptionV3
Epoch 5/100
61/61 [==============================] - 76s 1s/step - loss: 0.5345 - accuracy: 0.7887 - val_loss: 0.5721 - val_accuracy: 0.

Epoch 00005: saving model to gdrive/MyDrive/Deep_Learning_Course_Project/Model-Checkpoints/InceptionV3
Epoch 6/100
61/61 [==============================] - 75s 1s/step - loss: 0.4957 - accuracy: 0.8135 - val_loss: 0.6748 - val_accuracy: 0.

Epoch 00006: saving model to gdrive/MyDrive/Deep_Learning_Course_Project/Model-Checkpoints/InceptionV3
Epoch 7/100
61/61 [==============================] - 75s 1s/step - loss: 0.4660 - accuracy: 0.8156 - val_loss: 0.5046 - val_accuracy: 0.

Epoch 00007: saving model to gdrive/MyDrive/Deep_Learning_Course_Project/Model-Checkpoints/InceptionV3
Epoch 8/100
61/61 [==============================] - 75s 1s/step - loss: 0.4784 - accuracy: 0.8171 - val_loss: 0.6136 - val_accuracy: 0.

Epoch 00008: saving model to gdrive/MyDrive/Deep_Learning_Course_Project/Model-Checkpoints/InceptionV3
Epoch 9/100
61/61 [==============================] - 75s 1s/step - loss: 0.4769 - accuracy: 0.8032 - val_loss: 0.5283 - val_accuracy: 0.
```

```
Epoch 00009: saving model to gdrive/MyDrive/Deep_Learning_Course_Project/Model-Checkpoints/InceptionV3
Epoch 10/100
61/61 [==============================] - 75s 1s/step - loss: 0.4529 - accuracy: 0.8223 - val_loss: 0.5484 - val_accuracy: 0.

Epoch 00010: saving model to gdrive/MyDrive/Deep_Learning_Course_Project/Model-Checkpoints/InceptionV3
Epoch 11/100
61/61 [==============================] - 75s 1s/step - loss: 0.4472 - accuracy: 0.8275 - val_loss: 0.4995 - val_accuracy: 0.

Epoch 00011: saving model to gdrive/MyDrive/Deep_Learning_Course_Project/Model-Checkpoints/InceptionV3
Epoch 12/100
61/61 [==============================] - 75s 1s/step - loss: 0.3815 - accuracy: 0.8456 - val_loss: 0.5364 - val_accuracy: 0.

Epoch 00012: saving model to gdrive/MyDrive/Deep_Learning_Course_Project/Model-Checkpoints/InceptionV3
Epoch 13/100
61/61 [==============================] - 75s 1s/step - loss: 0.4216 - accuracy: 0.8388 - val_loss: 0.5230 - val_accuracy: 0.

Epoch 00013: saving model to gdrive/MyDrive/Deep_Learning_Course_Project/Model-Checkpoints/InceptionV3
Epoch 14/100
61/61 [==============================] - 75s 1s/step - loss: 0.4312 - accuracy: 0.8285 - val_loss: 0.5177 - val_accuracy: 0.

Epoch 00014: saving model to gdrive/MyDrive/Deep_Learning_Course_Project/Model-Checkpoints/InceptionV3
```

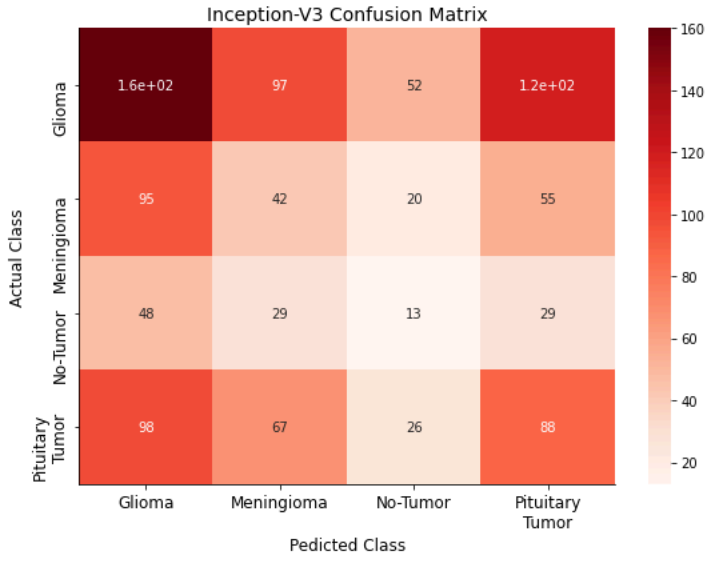
8.3.3 InceptionV3 Model Training Process Statistics

```
training_process_viz(pd.DataFrame(inception_model_train_history.history),
                     plot_title = 'Inception-V3 Training Statistics')
```



8.3.4 Confusion Matrix for InceptionV3 Model

```
with tf.device(gpu_device_location) if gpu_device_location else tpu_strategy.scope() if tpu_device_location else tf.device(cpu_dev
    confusion_matrix_viz(inception_cnn_model,
                         test_dataset,
                         plot_title = "Inception-V3 Confusion Matrix")
```



```
inceptionv3_report_df = generate_report(inception_cnn_model, test_dataset = test_dataset, row_indexes = ['InceptionV3'])
inceptionv3_report_df
```

```
33/33 [==============================] - 32s 962ms/step - loss: 0.4522 - accuracy: 0.8256
```

	MAE	MSE	RMSE	Loss	Accuracy	F1-Score
InceptionV3	1.365125	3.101156	1.76101	0.452249	0.825626	0.2842

** 8.4 Unet model code **

```
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.optimizers import Adam

# U-Net Model Definition
def unet_model(input_size=(224, 224, 3)):
    inputs = layers.Input(input_size)
```

** 8.41 **

```
    # Encoder (Contracting Path)
    c1 = layers.Conv2D(64, (3, 3), activation='relu', padding='same')(inputs)
    c1 = layers.Conv2D(64, (3, 3), activation='relu', padding='same')(c1)
    p1 = layers.MaxPooling2D((2, 2))(c1)

    c2 = layers.Conv2D(128, (3, 3), activation='relu', padding='same')(p1)
    c2 = layers.Conv2D(128, (3, 3), activation='relu', padding='same')(c2)
    p2 = layers.MaxPooling2D((2, 2))(c2)

    c3 = layers.Conv2D(256, (3, 3), activation='relu', padding='same')(p2)
    c3 = layers.Conv2D(256, (3, 3), activation='relu', padding='same')(c3)
    p3 = layers.MaxPooling2D((2, 2))(c3)

    c4 = layers.Conv2D(512, (3, 3), activation='relu', padding='same')(p3)
    c4 = layers.Conv2D(512, (3, 3), activation='relu', padding='same')(c4)
    p4 = layers.MaxPooling2D((2, 2))(c4)

    c5 = layers.Conv2D(1024, (3, 3), activation='relu', padding='same')(p4)
    c5 = layers.Conv2D(1024, (3, 3), activation='relu', padding='same')(c5)

    # Decoder (Expansive Path)
    u6 = layers.Conv2DTranspose(512, (3, 3), strides=(2, 2), padding='same')(c5)
    u6 = layers.concatenate([u6, c4])
    c6 = layers.Conv2D(512, (3, 3), activation='relu', padding='same')(u6)
    c6 = layers.Conv2D(512, (3, 3), activation='relu', padding='same')(c6)

    u7 = layers.Conv2DTranspose(256, (3, 3), strides=(2, 2), padding='same')(c6)
    u7 = layers.concatenate([u7, c3])
    c7 = layers.Conv2D(256, (3, 3), activation='relu', padding='same')(u7)
    c7 = layers.Conv2D(256, (3, 3), activation='relu', padding='same')(c7)

    u8 = layers.Conv2DTranspose(128, (3, 3), strides=(2, 2), padding='same')(c7)
    u8 = layers.concatenate([u8, c2])
    c8 = layers.Conv2D(128, (3, 3), activation='relu', padding='same')(u8)
    c8 = layers.Conv2D(128, (3, 3), activation='relu', padding='same')(c8)

    u9 = layers.Conv2DTranspose(64, (3, 3), strides=(2, 2), padding='same')(c8)
    u9 = layers.concatenate([u9, c1])
    c9 = layers.Conv2D(64, (3, 3), activation='relu', padding='same')(u9)
    c9 = layers.Conv2D(64, (3, 3), activation='relu', padding='same')(c9)

    # Output Layer
    outputs = layers.Conv2D(1, (1, 1), activation='sigmoid')(c9)  # Binary segmentation (1 for tumor, 0 for no tumor)
        # Create the model
    model = models.Model(inputs, outputs)

    # Compile the model
    model.compile(optimizer=Adam(lr=1e-4), loss='binary_crossentropy', metrics=['accuracy'])
    return model

# Model Summary
model = unet_model()
model.summary()
```

** 8.4.2 Training and Validation of UNET **

```
with tf.device(gpu_device_location) if gpu_device_location else tpu_strategy.scope() if tpu_device_location else tf.device(cpu_dev
  Unet_train_history = unet_model.fit(train_dataset,
                                      batch_size = batch_size,
                                      validation_data = validation_dataset,
                                      epochs = 100,
```