BRAIN TUMOR CLASSIFICATION FROM MRI

Team – 6

Alkesh Shukla – S20210020252

Gaurav Anand – S20210020273

Rohit N – S20210020297

AGENDA

Introduction

Data Preprocessing

Feature Extraction - Wavelet

Logistic Regression

Support Vector Machine

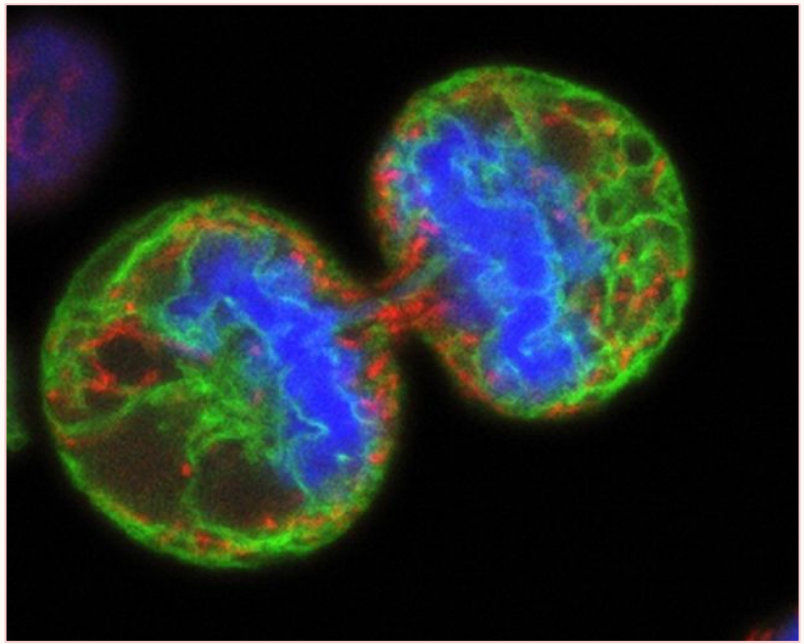
Artificial Neural Network

Convolution Neural Network

BRAIN TUMOR

- A collection of abnormal cells that grows in the brain or central spine canal.

- One abnormal cell
  becomes two,
  two becomes four,
  four becomes eight,
  until there is a
  lump of abnormal cells.



DIAGNOSING BRAIN TUMOR - MRI

Magnetic Resonance Imaging (MRI):

- Utilizes powerful magnets and radio waves.
- Generates detailed cross-sectional images of the body's internal structures.
- Non-invasive and does not involve ionizing radiation.

Key Features:

- Safe diagnostic tool.
- Provides comprehensive views of soft tissues, organs, and the brain.
- Utilizes principles of nuclear magnetic resonance.

Advantages:

- Enables accurate diagnosis.
- Facilitates precise treatment planning.
- Offers detailed insights without the use of harmful radiation.

CATEGORIES

Classes of Brain Tumors	Basic Description
Glioma	Originates in glial cells. - Diverse appearances on MRI aid precise identification.
Meningioma	Arises from the meninges. - Distinct features on scans for clear diagnosis.
No Tumor	Absence of detectable tumor. - Provides reassurance, eliminating tumor-related concerns.
Pituitary Tumor	Develops in the pituitary gland. - Recognizable patterns aiding endocrine-related evaluations.

METRICS

| Evaluation Metrics

Precision

Precision measures the ability of the model to correctly identify positive instances for each class among all instances predicted as positive.

$$\text{Precision}_c = \frac{\text{TP}_c}{\text{TP}_c + \text{FP}_c}$$

Recall (Sensitivity or True Positive Rate)

Recall calculates the ability of the model to correctly identify positive instances for each class among all actual positive instances.

$$\text{Recall}_c = \frac{\text{TP}_c}{\text{TP}_c + \text{FN}_c}$$

F1-Score

The F1-score is the harmonic mean of precision and recall. It provides a balanced measure that combines both metrics for each class.

$$\text{F1-Score}_c = 2 \times \frac{\text{Precision}_c \times \text{Recall}_c}{\text{Precision}_c + \text{Recall}_c}$$

Accuracy

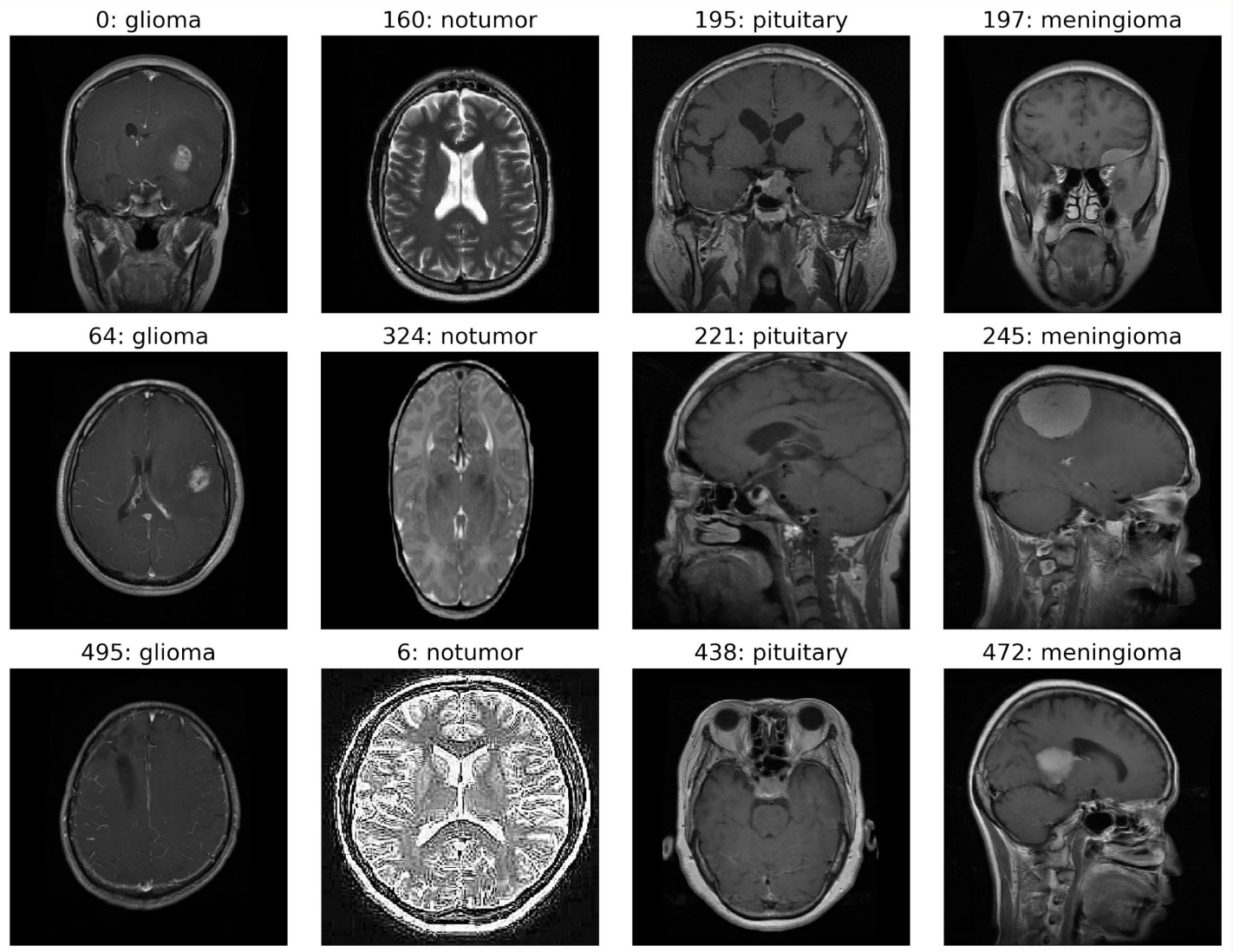
Accuracy measures the overall correctness of the model's predictions across all classes.

$$\text{Accuracy} = \frac{\text{TP}_1 + \text{TP}_2 + \ldots + \text{TP}_N}{\text{TP}_1 + \text{TP}_2 + \ldots + \text{TP}_N + \text{FP}_1 + \text{FP}_2 + \ldots + \text{FP}_N + \text{FN}_1 + \text{FN}_2 + \ldots + \text{FN}_N}$$

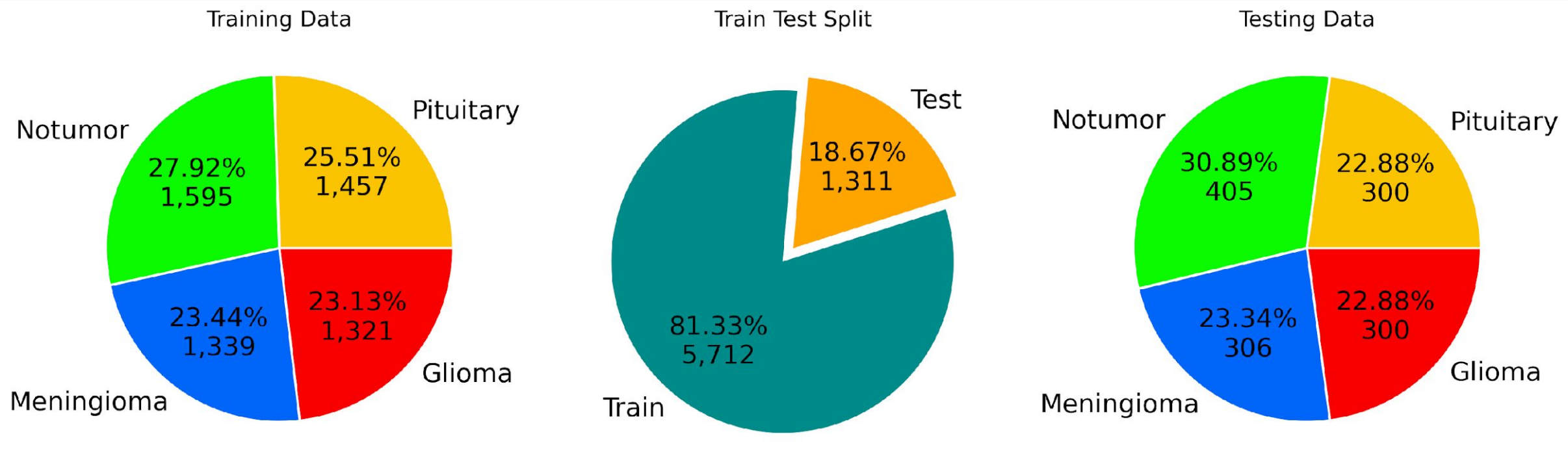
DATA PREPROCESSING

Exploring & Augmenting the dataset.

MRI IMAGE DATASET



DATA DISTRIBUTION



DATA AUGMENTATION

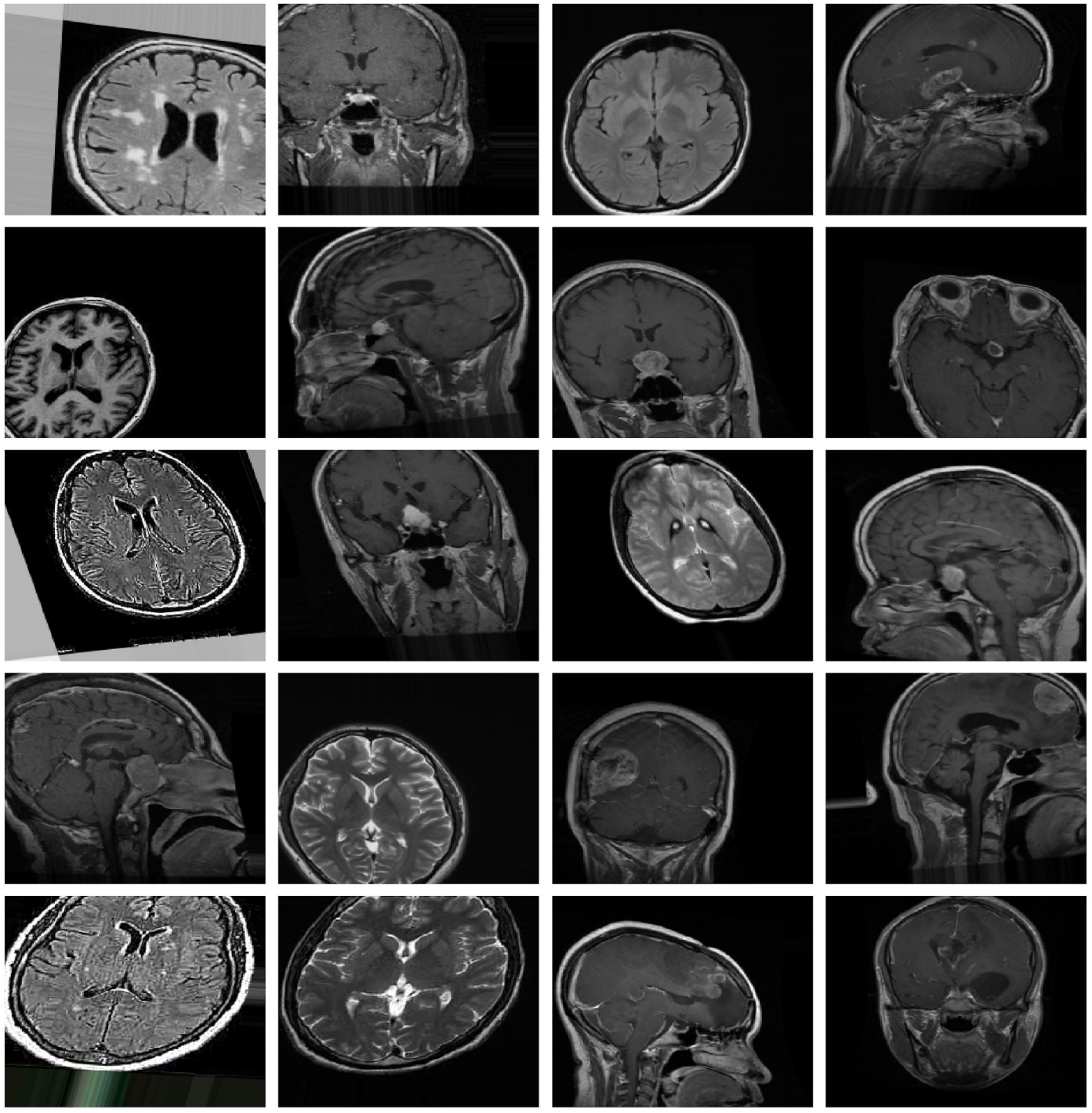
- Applying Data Augmentation techniques on the dataset, by using ImageDataGenerator.
- Applied the following techniques:
  - Rescale
  - Rotate Range : 10 degree
  - Shifting Width : 0.2 * img_width
  - Shifting Height : 0.2 * img_height
  - Sheer : 12.5 * img_height
  - Horizontal flip

- Used flow_from_directory to load data from dataset directories.

```
# Data augmentation and preprocessing
train_datagen = ImageDataGenerator(rescale=1./255,
                                   rotation_range=10,
                                   brightness_range=(0.85, 1.15),
                                   width_shift_range=0.2,
                                   height_shift_range=0.2,
                                   shear_range=12.5,
                                   zoom_range=0,
                                   horizontal_flip=True,
                                   vertical_flip=False,
                                   fill_mode="nearest")


# applying the generator to training data with constant seed
train_generator = train_datagen.flow_from_directory(train_dir,
                                                    target_size=image_size,
                                                    batch_size=batch_size,
                                                    class_mode="categorical",
                                                    seed=SEED)

# No augmentation of the test data, just rescaling
test_datagen = ImageDataGenerator(rescale=1./255)

# applying the generator to testing data with constant seed
test_generator = test_datagen.flow_from_directory(test_dir,
                                                  target_size=image_size,
                                                  batch_size=batch_size,
                                                  class_mode="categorical",
                                                  shuffle=False,
                                                  seed=SEED)
```

AUGMENTED DATASET

Augmented Dataset involves expanding data collections by adding more examples, enhancing the accuracy and adaptability of artificial intelligence systems. It's like broadening a library for better real-world performance.



FEATURE EXTRACTION

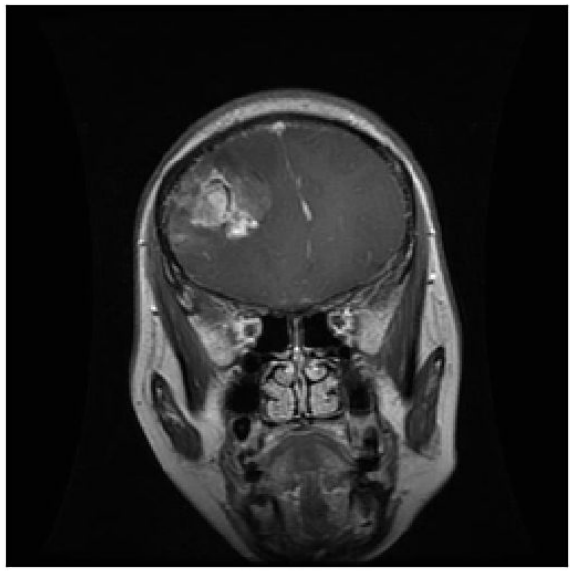
Wavelet Feature Extraction

WAVELET FEATURE EXTRACTION

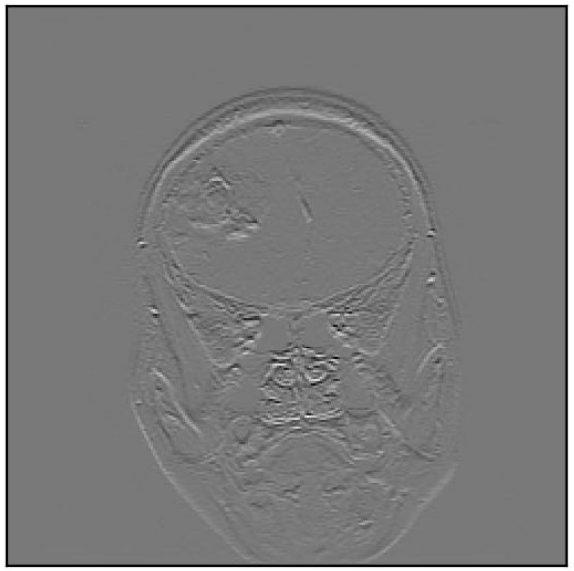
- Applied a 2D Discrete Wavelet Transform to an image.
- The LL component represents the approximation of the image at a lower resolution.
- The LH component contains the high-frequency details in the horizontal direction.
- The HL component contains the high-frequency details in the vertical direction.
- The HH component contains high-frequency details in both the horizontal and vertical directions.

```python
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
import pywt
import pywt.data


def Apply_wavelet(img_path,label):
  original = Image.open(img_path)
  titles = [f'{label} Approximation', ' Horizontal detail',
            'Vertical detail', 'Diagonal detail']
  coeffs2 = pywt.dwt2(original, 'bior1.3')
  LL, (LH, HL, HH) = coeffs2
  fig = plt.figure(figsize=(12, 3))
  for i, a in enumerate([LL, LH, HL, HH]):
              ax = fig.add_subplot(1, 4, i + 1)
              ax.imshow(a, interpolation="nearest", cmap=plt.cm.gray)
              ax.set_title(titles[i], fontsize=10)
              ax.set_xticks([])
              ax.set_yticks([])
  fig.tight_layout()
  plt.show()
```

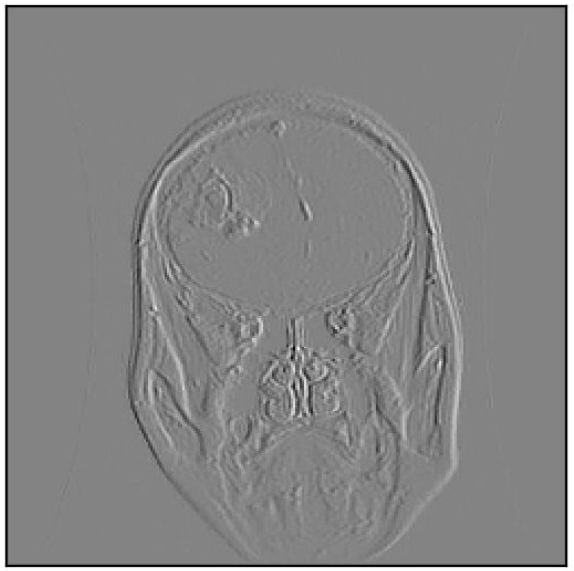
Glioma Approximation



Horizontal detail

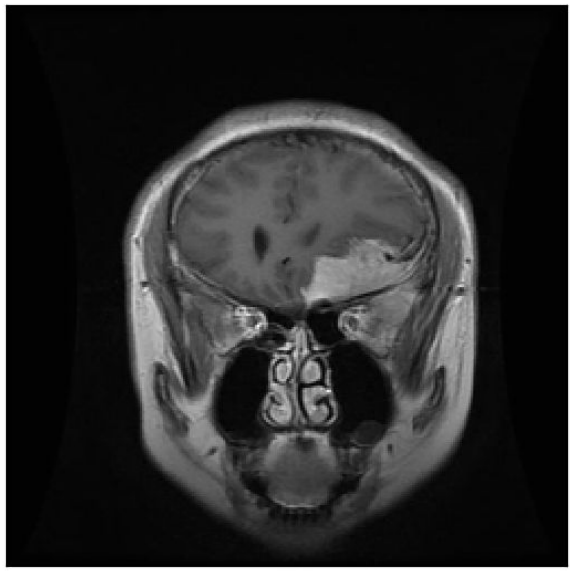


Vertical detail

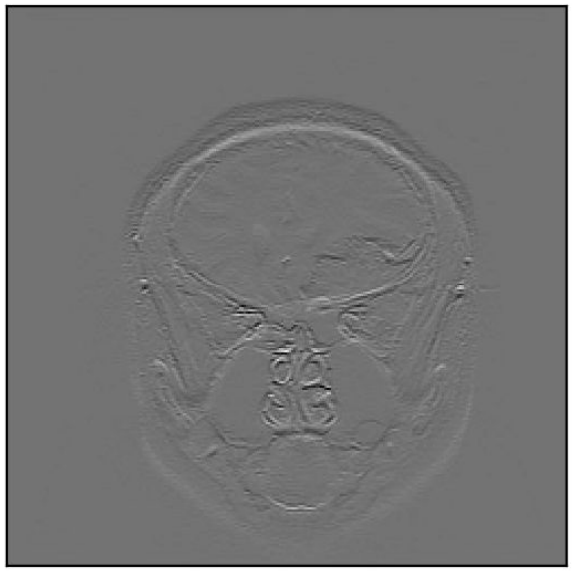


Diagonal detail

Meningioma Approximation



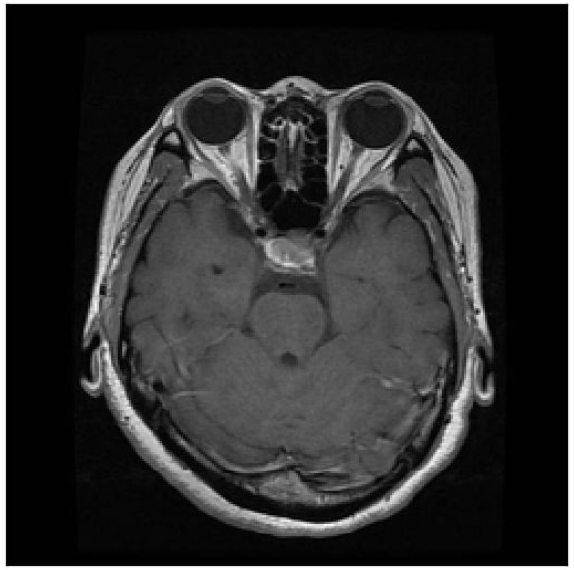
Horizontal detail



Vertical detail

Diagonal detail

Pitutary Approximation

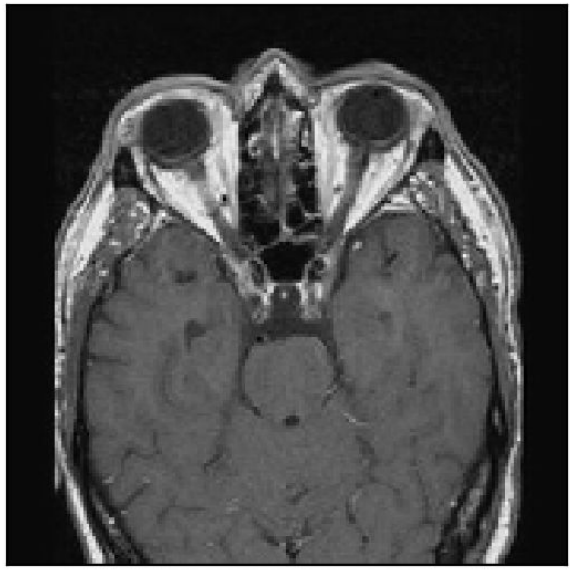


Horizontal detail

Vertical detail

Diagonal detail

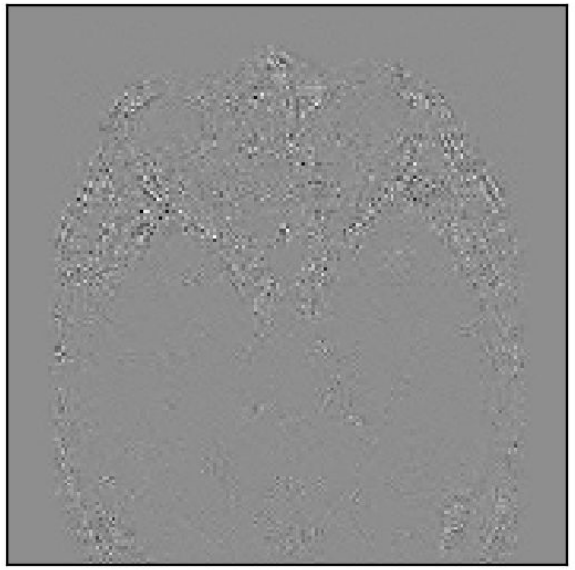
No Tumor Approximation



Horizontal detail

Vertical detail

Diagonal detail



LOGISTIC REGRESSION

Using Logistic Regression from sklearn for multiclass image classification.

SKLEARN LOGISTIC REGRESSION


```
from sklearn.linear_model import LogisticRegression
lg = LogisticRegression(C=0.1)
lg.fit(xtrain, ytrain)
```

```
▾ LogisticRegression
LogisticRegression(C=0.1)
```

```
print("Training Score:", lg.score(xtrain, ytrain))
print("Testing Score:", lg.score(xtest, ytest))
```

```
Training Score: 0.9943094769096082
Testing Score: 0.8285214348206474
```

SVM

Using Support Vector Machines from sklearn for multiclass image classification.

SKLEARN SVM


```
from sklearn.svm import SVC
sv = SVC()
sv.fit(xtrain, ytrain)
```

SVC

SVC()

```
print("Training Score:", sv.score(xtrain, ytrain))
print("Testing Score:", sv.score(xtest, ytest))
```

```
Training Score: 0.940030641278179
Testing Score: 0.8740157480314961
```

ARTIFICIAL NEURAL NETWORK

Build & trained ANN for multiclass image classification.

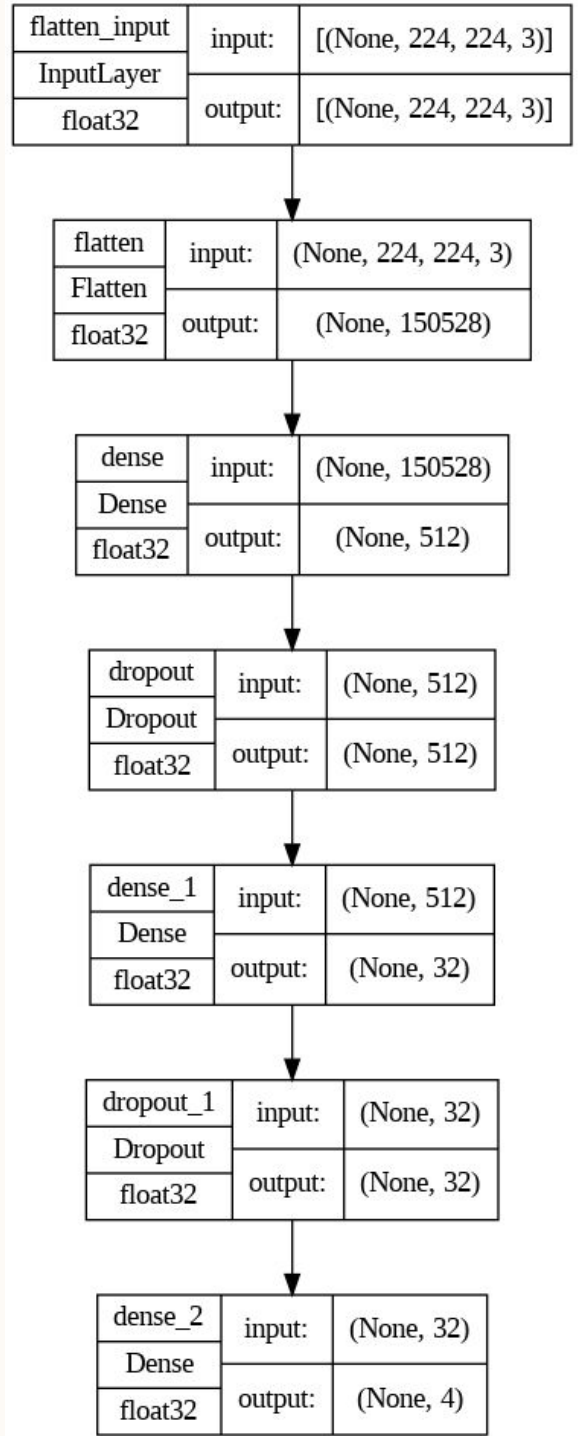
ANN MODEL USING TENSORFLOW


```python
from tensorflow.keras import models, layers


model_ANN = models.Sequential()

# Add Flatten layer to convert 2D input to 1D
model_ANN.add(layers.Flatten(input_shape = image_shape))

model_ANN.add(layers.Dense(512, activation='relu'))
model_ANN.add(layers.Dropout(0.4))
model_ANN.add(layers.Dense(32, activation='relu'))
model_ANN.add(layers.Dropout(0.4))
model_ANN.add(layers.Dense(N_TYPES, activation='softmax'))

model_ANN.summary()

optimizer = tf.keras.optimizers.Adam(learning_rate=0.001)
model_ANN.compile(optimizer=optimizer, loss='categorical_crossentropy', metrics=['accuracy'])
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, 150528)            0

 dense (Dense)               (None, 512)               77070848

 dropout (Dropout)           (None, 512)               0

 dense_1 (Dense)             (None, 32)                16416

 dropout_1 (Dropout)         (None, 32)                0

 dense_2 (Dense)             (None, 4)                 132

=================================================================
Total params: 77087396 (294.07 MB)
Trainable params: 77087396 (294.07 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

VISUALIZING ANN


```
model_ANN_visual = models.Model(inputs=model_ANN.input,
                                outputs=model_ANN.output)

# Save model architecture to a file
plot_model(model_ANN_visual , show_dtype=True,
           to_file='model_ANN_architecture.png',
           show_shapes=True)

# Display model architecture in the notebook

from IPython.display import Image
Image(retina=True,
      filename='model_ANN_architecture.png')
```



TRAINING & TESTING ANN


```
# Stop training if loss doesn't keep decreasing.
model_es = EarlyStopping(monitor='loss', patience = 5, verbose=True)

# Training the model
history_ANN = model_ANN.fit(train_generator,
                            steps_per_epoch=steps_per_epoch,
                            epochs=epochs,
                            validation_data=test_generator,
                            validation_steps=validation_steps,
                            callbacks = [model_es])
```

```
# Evaluating the model
loss_ANN, accuracy_ANN = model_ANN.evaluate(test_generator,
                                            steps=test_generator.samples//batch_size)
print(f"Test Loss for ANN model  : {loss_ANN:0.5f}")
print(f"Test Accuracy for ANN model  : {accuracy_ANN:0.5f}")
```

```
40/40 [==============================] - 4s 90ms/step - loss: 1.3806 - accuracy: 0.3164
Test Loss for ANN model  : 1.38057
Test Accuracy for ANN model  : 0.31641
```

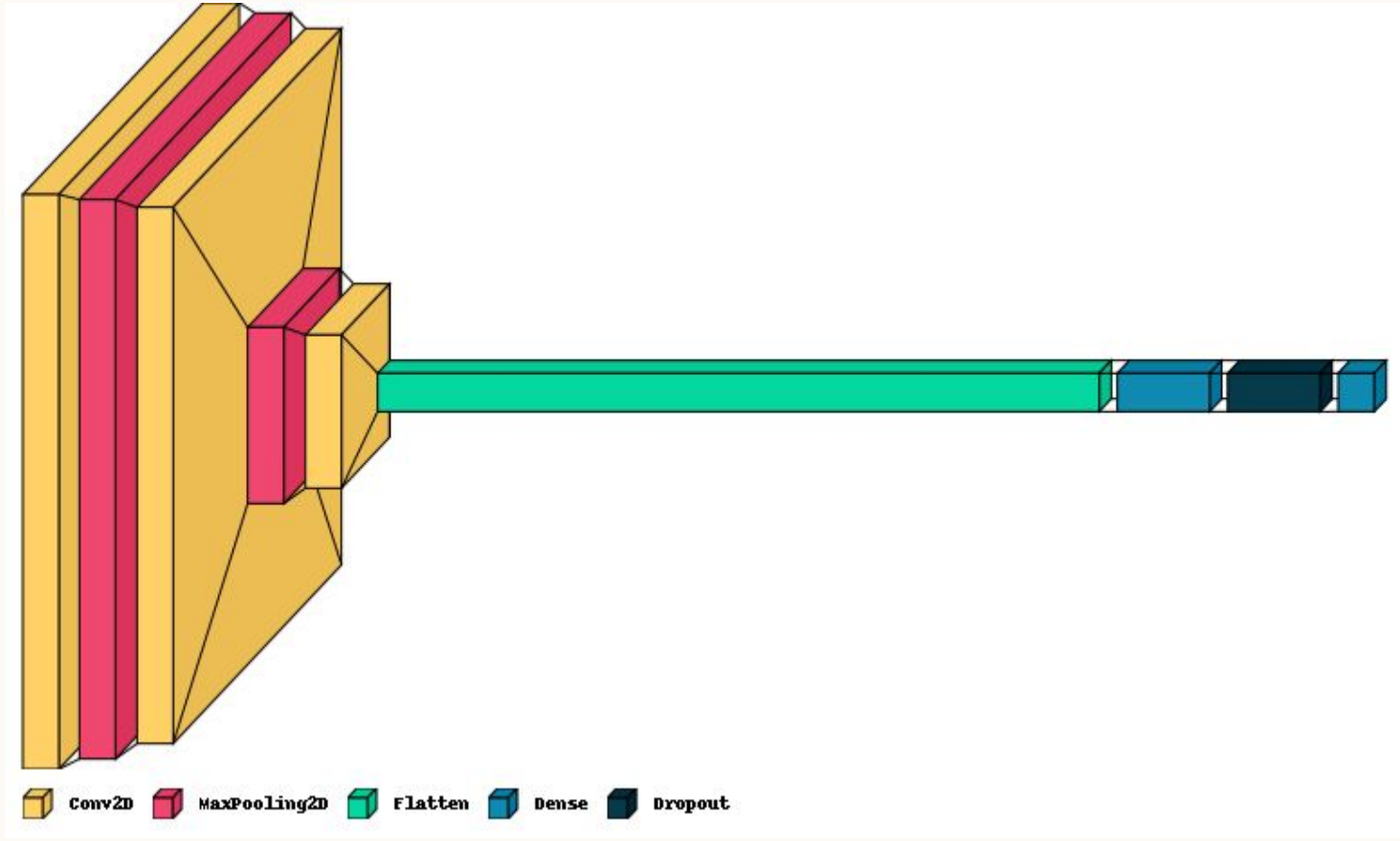
CONVOLUTIONAL NEURAL NETWORK

Build & trained CNN for multiclass image classification.

CNN USING TENSORFLOW


```python
# Define the model architecture
model_CNN = models.Sequential()

# Convolutional layer 1
model_CNN .add(Conv2D(32, (4, 4), activation="relu",
                      input_shape=image_shape))
model_CNN .add(MaxPooling2D(pool_size=(3, 3)))

# Convolutional layer 2
model_CNN .add(Conv2D(64, (4, 4), activation="relu"))
model_CNN .add(MaxPooling2D(pool_size=(3, 3)))


# Convolutional layer 3
model_CNN .add(Conv2D(128, (4, 4), activation="relu"))
model_CNN .add(Flatten())

# Full connect layers
model_CNN .add(Dense(512, activation="relu"))
model_CNN .add(Dropout(0.5, seed=SEED))
model_CNN .add(Dense(N_TYPES, activation="softmax"))

model_CNN .summary()


optimizer = legacy.Adam(learning_rate=0.001)

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

```python
from visualkeras import layered_view

# Visualize the model
layered_view(model_CNN, legend=True, max_xy=300)
```



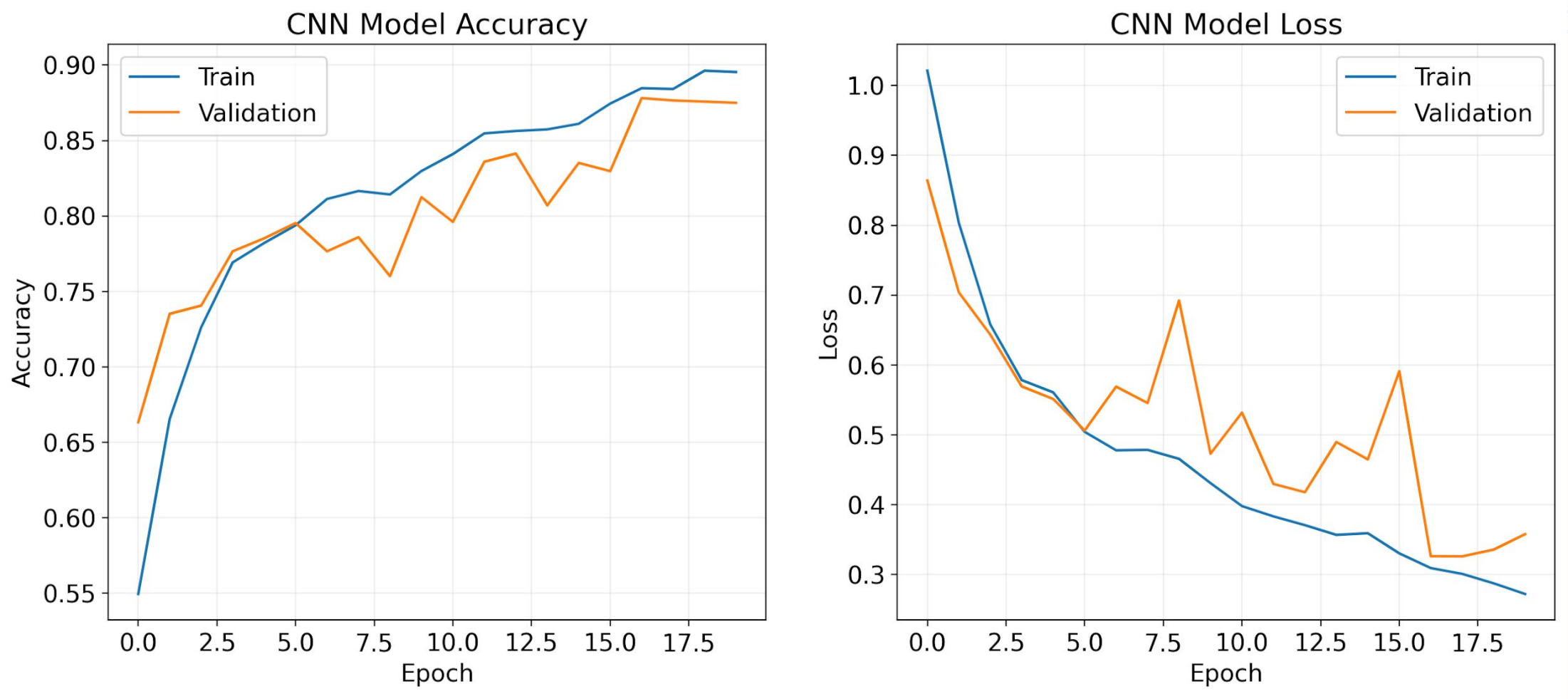
TRAINING & TESTING CNN


```
# Training the model
history_CNN = model_CNN.fit(train_generator,
                            steps_per_epoch=steps_per_epoch,
                            epochs=epochs,
                            validation_data=test_generator,
                            validation_steps=validation_steps,
                            callbacks = [model_es])
```

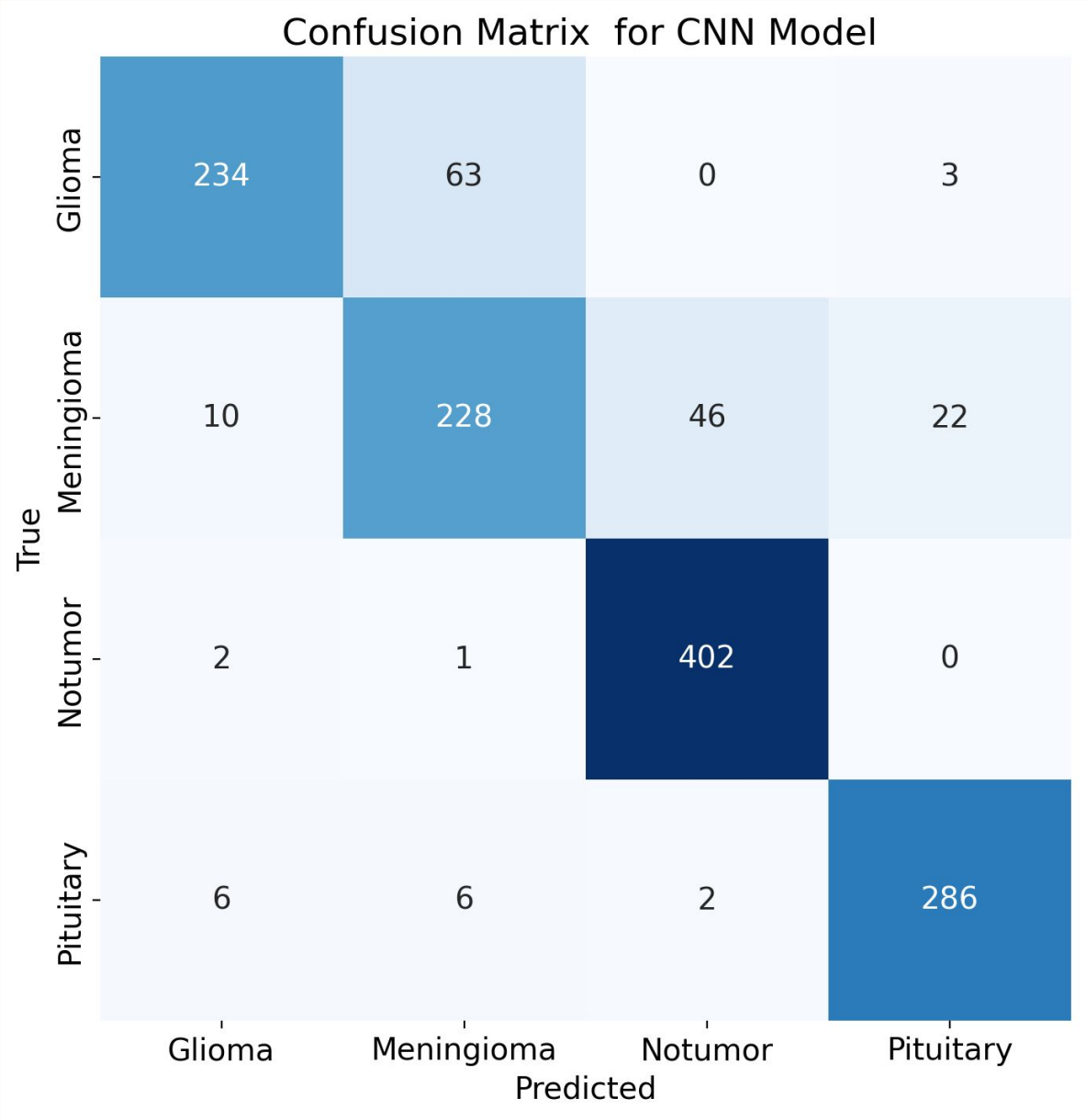
```
# Evaluating the model
loss_CNN, accuracy_CNN = model_CNN.evaluate(test_generator, steps=test_generator.samples//batch_size)
print(f"Test Loss for CNN model  : {loss_CNN:0.5f}")
print(f"Test Accuracy for CNN model  : {accuracy_CNN:0.5f}")
```

```
40/40 [==============================] - 5s 107ms/step - loss: 0.3579 - accuracy: 0.8750
Test Loss for CNN model  : 0.35788
Test Accuracy for CNN model  : 0.87500
```

LOSS & ACCURACY PLOTS



CONFUSION MATRIX & METRICS




```
# Showing metrics
calculate_metrics(confusion_matrix_CNN,
                  categories=CLASS_TYPES)
```

```
Class: Pituitary
Precision: 0.929
Recall: 0.780
F1-Score: 0.848

Class: Notumor
Precision: 0.765
Recall: 0.745
F1-Score: 0.755

Class: Meningioma
Precision: 0.893
Recall: 0.993
F1-Score: 0.940

Class: Glioma
Precision: 0.920
Recall: 0.953
F1-Score: 0.936


Accuracy: 0.877
```

Model Evaluation

Model used to classify Brain Tumors	Acuracy
Logistic Regression	**Training Score:** 0.9943094769096082 **Testing Score:** 0.8285214348206474
Support Vector Machine	**Training Score:** 0.9400306412781 79 **Testing Score:** 0.8740157480314961
Artificial Neural Network	**Test Loss for ANN model :** 1.38057 **Test Accuracy for ANN model :** 0.31641
Convolution Neural Network	**Test Loss for CNN model :** 0.35788 **Test Accuracy for CNN model :** 0.87500
MobileNet + ANN	**Test Loss :** 0.1694 **Test Accuracy :** 0.9023

THANK YOU

Team – 6

Alkesh Shukla – S20210020252

Gaurav Anand – S20210020273

Rohit N – S20210020297