PROBLEM STATEMENT

The human body is made up of many organs and brain is the most critical and vital organ of them all. One of the common reasons for dysfunction of brain is brain tumor. A tumor is nothing but excess cells growing in an uncontrolled manner. Brain tumor cells grow in a way that they eventually take up all the nutrients meant for the healthy cells and tissues, which results in brain failure. Currently, doctors locate the position and the area of brain tumor by looking at the MR Images of the brain of the patient manually. This results in inaccurate detection of the tumor and is considered very time consuming.

A Brain Cancer is very critical disease which causes deaths of many individuals. The brain tumor detection and classification system is available so that it can be diagnosed at early stages. Cancer classification is the most challenging tasks in clinical diagnosis. This project deals with such a system, which uses computer, based procedures to detect tumor blocks and classify the type of tumor using Convolution Neural Network.

Algorithm for MRI images of different patients:

Different types of image processing techniques like image segmentation, image enhancement and feature extraction are used for the brain tumor detection in the MRI images of the cancer-affected patients. Detecting Brain tumor using Image Processing techniques its involves the four stages isImage Pre-Processing, Image segmentation, Feature Extraction, and Classification. Image processing and neural network techniques are used for improve the performance of detecting and classifying brain tumor in MRI images.

The brain tumors are classified into mainly two types: Primary brain tumor (benign tumor) and secondary brain tumor (malignant tumor).The benign

tumor is one type of cell grows slowly in the brain and type of brain tumor is gliomas. It originates from non neuronal brain cells called astrocytes.

The secondary tumors are more aggressive and more quick to spread into other tissue. Secondary brain tumor originates through other part of the body. These type of tumor have a cancer cell in the body that is metastatic which spread into different areas of the body like brain, lungs etc. Secondary brain tumor is very malignant. The reason of secondary brain tumor cause is mainly due to lungs cancer, kidney cancer, bladder cancer etc.

In conclusion, with MRI we are able to visualize the details of internal structure of brain and from that we can observe the different types of tissues of human body. MRI images have a better quality as compared to other medical imaging techniques like X- ray and computer tomography. MRI is good technique for knowing the brain tumor in human body. There are different images of MRI for mapping tumor induced.

IMPLEMENTATION

1. **Import Libraries:**

    - Import the necessary libraries, including TensorFlow or Keras, for building and training the neural network, and other utility libraries for data manipulation.

2. **Load and Preprocess Data:**

    - Load the image data and corresponding labels.

    - Preprocess the data, which may include scaling pixel values, resizing images, and converting labels to a suitable format (e.g., one-hot encoding for categorical labels).

3. **Build the CNN Model:**

    - Create a sequential model using the Keras API.
    - Add an input layer with the specified input shape.
    - Add convolutional layers with appropriate parameters, such as filters, kernel size, activation function, and padding.
    - Intersperse the convolutional layers with max-pooling layers to down-sample spatial dimensions.
    - Flatten the output to a 1D vector.
    - Add dense (fully connected) layers with ReLU activation.
    - Introduce dropout layers to reduce overfitting.
    - Include the output layer with a single unit and sigmoid activation for binary
      classification.

4. **Compile the Model:**

    - Specify the optimizer (e.g., Adam), the loss function (binary crossentropy for binary classification), and metrics (accuracy).

5. **Train the Model:**

    - Use the **fit** method to train the model on the training data.
    - Specify the number of epochs and provide the training and validation data.

- Monitor training progress, and adjust hyperparameters or model architecture if needed.

6. **Evaluate the Model:**

   - Use the **evaluate** method to assess the model's performance on the test data.
   - Obtain metrics such as loss and accuracy.

7. **Make Predictions:**

   - Use the trained model to make predictions on new or unseen data (test set).
   - Optionally, apply a threshold (e.g., 0.5) to convert predicted probabilities into binary predictions.

8. **Performance Analysis:**

   - Analyze the model's performance by reviewing metrics, such as accuracy and loss, on both training and test sets.
   - Visualize training history through plots to identify trends and potential issues like overfitting or underfitting.

9. **Fine-Tuning:**

   - If necessary, fine-tune the model by adjusting hyperparameters, modifying architecture, or incorporating additional layers to enhance performance.

10. **Documentation and Reporting:**

    - Document the chosen architecture, hyperparameters, and any

modifications made during the training process.

- Provide a detailed report on the model's performance, highlighting accuracy, loss, and any insights gained from the analysis.

DATASET:

Link for dataset: https://www.kaggle.com/datasets/praneet0327/brain-tumor-dataset

The visual examination of a representative subset from the training dataset underscores the complexity and heterogeneity inherent in brain tumor images. Employing Matplotlib, a 4x4 grid of images is presented, where each image is not only a visual representation but also a data point for the neural network. The corresponding class labels, "No Tumor" or "Tumor," are dynamically assigned based on the actual ground truth labels stored in the training set (y_train). This qualitative inspection allows for an appreciation of the intricate details in brain images, emphasizing the necessity for a robust classification model capable of discerning nuanced patterns. The variations in image content, structures, and tumor characteristics pose challenges for accurate classification. The success of any subsequent machine learning model trained on this dataset will depend on its capacity to generalize across this diversity and effectively differentiate between tumor and non-tumor instances. This initial exploration serves as a crucial step in understanding the intricacies of the dataset, guiding further model development and performance evaluation.

ALGORITHM:

This project delves into the implementation of a Convolutional Neural Network (CNN) using the Keras API with a sequential model architecture, specifically designed for image classification tasks. The initial layer of the model is

configured to handle input images with dimensions of 224 pixels in height and width, with three color channels (RGB). Convolutional layers, characterized by a kernel size of (3,3) and Rectified Linear Unit (ReLU) activation, are strategically placed to enable the model to learn hierarchical spatial features from the input images. Max-pooling layers follow each convolutional block, effectively reducing spatial dimensions and promoting translation invariance. The subsequent Flatten layer transforms the 2D feature maps into a flat, one-dimensional vector, facilitating the transition to fully connected layers.

Dense layers, employing ReLU activation, are strategically inserted to capture complex, non- linear relationships within the data. Dropout layers are incorporated to introduce a form of regularization during training, randomly deactivating a specified fraction of input units to prevent overfitting. The output layer, utilizing a sigmoid activation function, renders the model well-suited for binary classification tasks. In the compilation phase, the model is configured with the Adam optimizer, known for its adaptive learning rate properties. Binary crossentropy serves as the loss function, fitting the binary nature of the classification task, and accuracy is chosen as the primary metric to monitor during training.

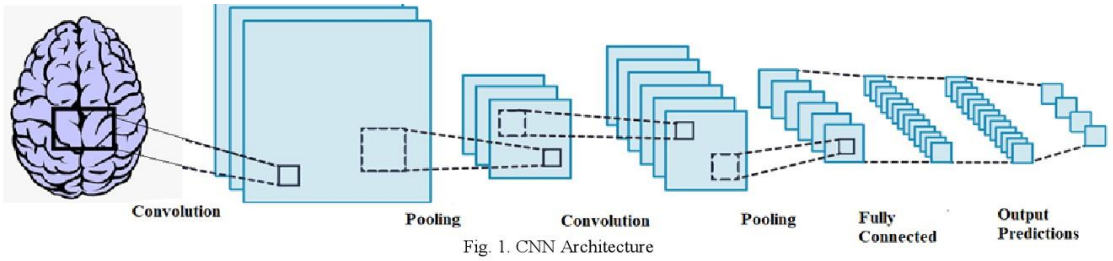
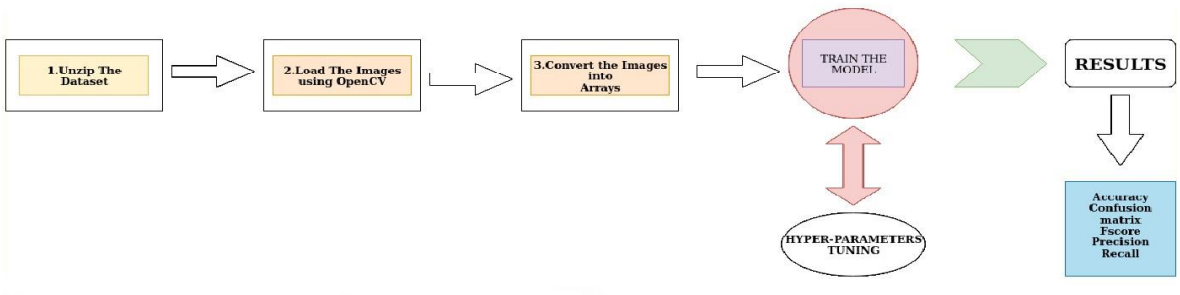


Fig. 1. CNN Architecture



Training unfolds over 10 epochs using scaled training data, while the model's performance is regularly assessed on a separate validation set, represented by the scaled testing data. After training, the model undergoes an evaluation phase on the test data to quantify its generalization capabilities. Predictions are generated on the test set, and a binary decision is made based on a threshold of 0.5 for the predicted probabilities.

This detailed algorithmic framework underscores the meticulous design choices made in constructing a deep learning model for image classification. The architecture's adaptability, demonstrated through the use of hyperparameters and the incorporation of regularization techniques, ensures a robust and flexible approach for a diverse range of classification tasks. The assessment of model performance through metrics like accuracy provides crucial insights into its efficacy in discerning patterns and features within the dataset, thereby contributing to the broader understanding of deep learning applications in image classification.

CODE

```
import warnings
warnings.filterwarnings('ign
ore')

import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.metrics import classification_report , confusion_matrix ,
accuracy_score from sklearn.model_selection import train_test_split
```

```
import cv2
#from google.colab.patches import cv2_imshow
from PIL import
Image import
tensorflow as tf
from tensorflow import
keras from keras import
Sequential
from keras.layers import Input, Dense,Conv2D , MaxPooling2D,
Flatten,BatchNormalization,Dropout
from tensorflow.keras.preprocessing import
image_dataset_from_directory import tensorflow_hub as hub

folder_path_no = "/kaggle/input/brain-mri-images-for-brain-tumor-
detection/no" folder_path_yes = "/kaggle/input/brain-mri-images-for-
brain-tumor-detection/yes" folder_no = os.listdir(folder_path_no)
folder_yes =
os.listdir(folder_path_yes)
print(len(folder_no))
print(len(folder_yes))
folder_no[:4]


no_label =
[0]*len(folder_no) yes_label
= [1]*len(folder_yes) labels
= no_label + yes_label
print(len(labels))
print(labels[:5
])
```

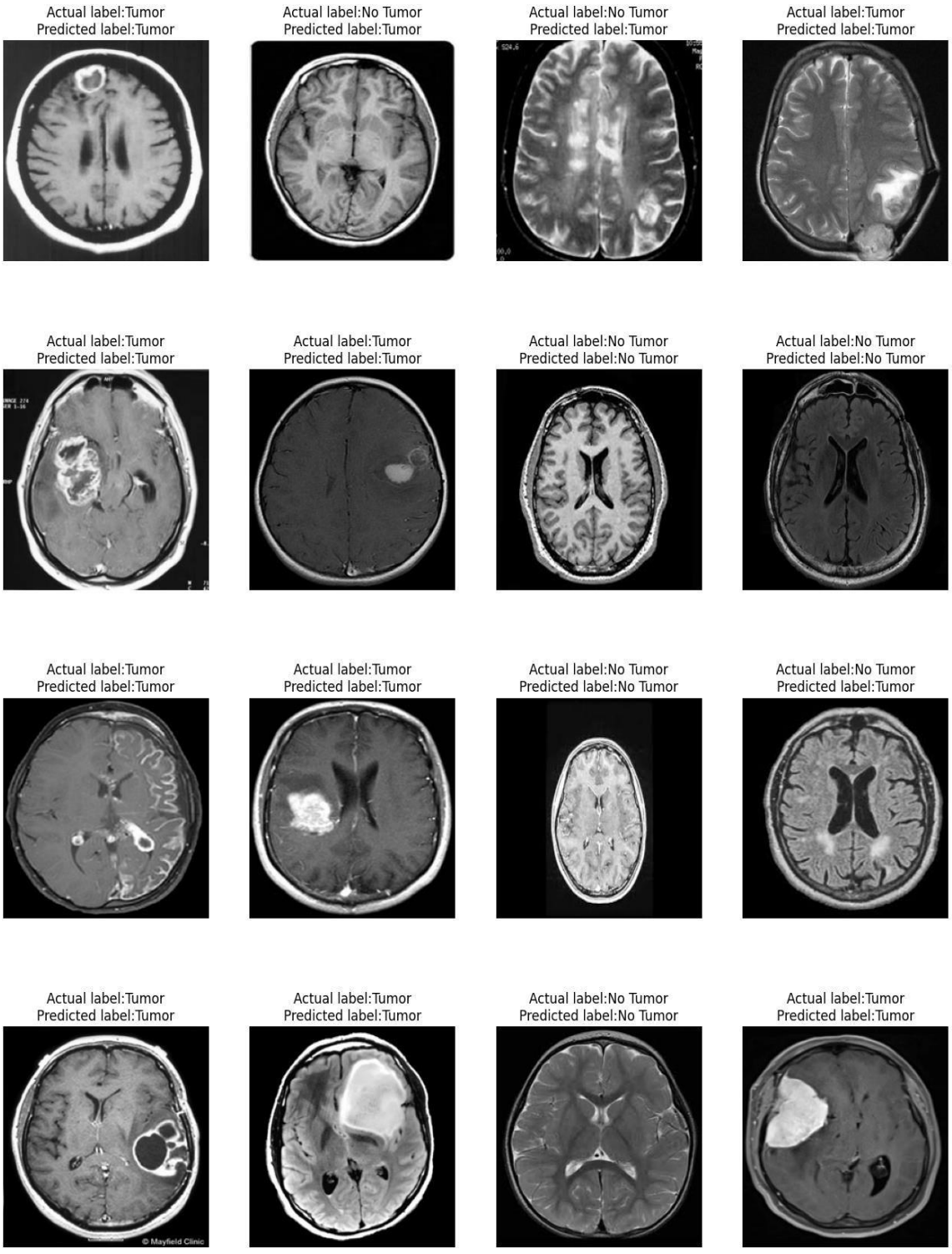
```
print(labels[-
5:])

data = []
for img in folder_no:
   image = Image.open("/kaggle/input/brain-mri-images-for-brain-tumor-
   detection/no/"+img) image = image.resize((224,224))
   image = image.convert("RGB")
   image = np.array(image)
   data.append(image)
for img in folder_yes:
   image = Image.open("/kaggle/input/brain-mri-images-for-brain-tumor-
detection/yes/"+img)
   image =
   image.resize((224,224)) image
   = image.convert("RGB") image
   = np.array(image)
   data.append(image)

len(data)

data[0].shape

x =
np.array(data)  y
=
np.array(labels)

x[0]
```

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.10,shuffle=True)
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)

x_train[0]

y_train

class_labels=["No
Tumor","Tumor"]
plt.figure(figsize=(16,20))
for i in range(16):
   plt.subplot(4,4,i+1)
   plt.imshow(x_train[i])
   plt.title(f"{class_labels[y_train[i
   ]]}") plt.axis("off")

x_train_scaled = x_train/255
x_test_scaled = x_test/255

model = Sequential()
model.add(Input(shape=(224,224,
3)))
model.add(Conv2D(filters=80,kernel_size=(3,3),padding="valid",  strides=(1,1)
,activation="relu"))
model.add(MaxPooling2D(pool_size=(2,2
)))
model.add(Conv2D(filters=64,kernel_size=(3,3),padding="valid", strides=(1,1),
```

```
activation="relu"))
model.add(MaxPooling2D(pool_size=(2,2)))
#model.add(Conv2D(filters=34,kernel_size=(3,3),padding="valid",
strides=(1,1), activation="relu"))
#model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())
model.add(Dense(units=500,
activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(units=500,
activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(units=1,
activation='sigmoid'))
model.compile(optimizer="adam",
        loss="binary_crossentropy", metrics=["accuracy"])
model.summary()

history = model.fit(x_train_scaled, y_train,

epochs=10,validation_data=(x_test_scaled,y_test) loss, acc =

model.evaluate(x_test_scaled,y_test)
print("Accuracy on Test Data:",acc)

y_pred = model.predict(x_test_scaled)
y_pred = [1 if i>=0.5 else 0 for i in
y_pred] y_pred[:6]

y_test[:10]
```

```
y_pred[:10]


print("Confusion
Matrix:\n",confusion_matrix(y_test,y_pred)) print()
print("Classification Report:\n",classification_report(y_test,y_pred))
class_labels=["No
Tumor","Tumor"]
plt.figure(figsize=(16,20))
for i in range(16):
   plt.subplot(4,4,i+1)
   plt.imshow(x_test[i])
   plt.title(f"Actual label:{class_labels[y_test[i]]}\nPredicted
   label:{class_labels[y_pred[i]]}") plt.axis("off")
```

OUTPUT SCREENSHOTS



RESULTS

Brain tumor detection is a critical challenge in medical imaging. Automated brain tumor detection systems have the potential to improve the timeliness and accuracy of diagnosis, and to reduce the workload on radiologists. More research is needed to develop systems that are robust and reliable enough to be used in clinical practice. Medical image segmentation is a challenging issue due to the complexity of the images, as well as the lack of anatomical models that fully capture the potential deformations in each structure. This proposed method works very effectively to the initial cluster size and cluster centers. The segmentation is done by using BWT techniques whose accuracy and computation speed are less. This work recommends a system that requires negligible human intrusion to partition the brain tissue. The main aim of this recommended system is to aid the human experts or neurosurgeons in identifying the patients with minimal time