International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Brain Tumor Detection Using Convolutional Neural Networks

Mr. P. Srinivasa Rao[1], *Etkilallo Swarnalatha*[2], *Mamidala Adithya*[3], *Nadukuda Shiva Praneeth*[4], *K Pavan Sai*[5]

[1]Associate Professor, ACE Engineering College Hyderabad, India psvass@gmail.com
Student, CSE, ACE Engineering College Hyderabad, India
[2]swarnalatha8536@gmail.com, [3]mamidalaadithya03@gmail.com, [4]shivapraneethnadukuda0789@gmail.com, [5]kpavansai1919@gmail.com

ABSTRACT:

Brain tumour detection is the process of identifying the presence of a tumour in the brain. This can be done through a variety of methods, but the most common is medical imaging, such as Magnetic Resonance Imaging (MRI) scans. Traditionally, brain tumours are detected by radiologists who examine MRI or CT scans of the brain and neurologists, neurosurgeons. Our objective is to automate this process of manual detection and classification of brain tumours for better and faster results. Early detection can help to reduce the risk of complications from brain tumours, such as neurological damage, cognitive impairment and can also save a life in due time. The main objective of the CNN-based brain tumour detection system is to streamline the diagnosis process and improve patient outcomes. By eliminating manual interventions, enhancing accuracy, and expediting diagnosis, this innovative system has the potential to transform brain tumour detection in healthcare, offering a robust and reliable solution for improving patient outcomes.

Keywords: Brain tumour, Brain tumour detection, Brain tumour classification, MRI images, Deep learning, Convolutional Neural Network, Neural Network.

I.INTRODUCTION

A brain tumor is an abnormal growth of cells in the brain. Unlike normal cells, brain tumor cells grow and divide uncontrollably, forming a mass of tissue. Brain tumors can be benign (noncancerous) or malignant (cancerous). Brain tumors pose a significant health concern worldwide, affecting people of all ages. These abnormal growths of cells within the brain can lead to severe health complications and, if left untreated, can be life-threatening. The exact cause of brain tumors is unknown, but there are a number of risk factors, including Age, Family history, Radiation exposure, Certain genetic disorders. The aim for a Brain Tumor Detection project is to automate this process of manual detection and classification of brain tumors for better and faster results. Traditionally, brain tumors are detected using MRI scans. Variability in tumor types can result in misinterpretation, leading to false positives or negatives. Early diagnosis of brain tumors is critical for improving patient outcomes. The ability of CNNs to learn intricate patterns and features from complex data, such as brain MRI scans, has revolutionized the process of tumordetection. Brain tumors can be difficult to detect in their early stages, but CNNs have the potential to improve the accuracy and efficiency of brain tumor detection. CNN- based systems automate the detection process, reducing the dependence on 2 human interpretations.

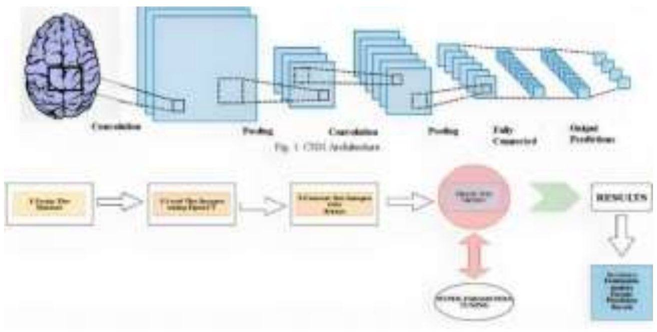

Fig: CNN Model Architecture The objectives of this project aim to achieve are:

- Early Detection: Enable early detection of brain tumors, even at their initial stages, to enhance the chances of timely medical intervention and improve patient outcomes.

- Automate Detection Process: Design an automated system that can analyses MRI scans without human intervention, reducing dependency on skilled radiologists and ensuring consistent and reliable results.
- Enhance Accuracy: Develop a CNN model capable of accurately identifying and classifying different types of brain tumors from MRI images, reducing the chances of misdiagnosis and improving overall accuracy.

II.OBJECTIVES

In our project there are several important objectives. They can be listed as:

- Deep Learning (DL) approaches have recently been popular in developing automated systems capable of accurately diagnosing or segmenting brain tumors in less time.
- DL enables a pre-trained Convolutional Neural Network (CNN) model for medical images, specifically for classifying brain cancers.

III. METHODOLOGY

In our project, firstly we collected raw data and then collected data should be preprocessed. Now, split the data into Trained data and Testing data. Fit the model to continue the process. Next is Scoring over Training model. Then Prediction is done. Confusion Matrix and Classification Report can help to get predicted output. Finally, Accuracy of the model present the predicted output.

IV. LITERATURE SURVEY

The significance The project "Brain Tumor Detection Using Active Contour Model" by Sagar Shankarrao Dake; Minh Nguyen; Wei Qi Yan; Shazeba Kazi proposes a method for automatically segmenting brain tumors from magnetic resonance imaging (MRI) images. The method is based on an active contour model (ACM), which is a type of snake model that can be used to segment objects in images. The ACM is first initialized with an approximate contour of the tumor. The ACM then iteratively updates its contour to minimize an energy function that includes terms for both image information and contour smoothness.

Brain Tumor Detection Using Fuzzy C-Means Clustering and Artificial Neural Networks (SR-FCM- CNN) by M. Sivakumar, S. Anand, and S. Ramakrishnan is a method for detecting brain tumors uses super-resolution, fuzzy C-means clustering, feature extraction and classification. The segmented tumor is then classified using an ANN. This method is sensitive to the selection of the number of clusters and the ANN architecture.

In a study conducted by H. Liu, S. Jiang, H. Zhang, S. Li, and H. Tian, the researchers developed a brain tumor detection method utilizing support vector machine (SVM) and texture features. Texture features were extracted from images using the gray-level co-occurrence matrix (GLCM). The SVM model was trained to categorize images as either tumorous or non- tumorous.

Project	**Author(s)**	**Method**	**Advantages**	**Disadvantages**
Brain Tumor Detection Using Active Contour Model	Ahmed G. El-Baz, Mohamed S. Abdel-Moniem, and Mohamed K. Aboul-Seoud	Active contour model (ACM)	Simple to implement, can be used to segment tumors of different shapes and sizes	Semi-automatic method that requires manual initialization of the tumor contour, sensitive to noise and artifacts in the image
Brain Tumor Detection Using Gabor Wavelets and Support Vector Machines	Amit Kumar Jain, Sanjay Kumar Singh, and Dheeraj Kumar Jain	Gabor wavelets and support vector machines (SVMs)	Can extract robust features from images, SVMs are good at classifying high-dimensional data	Computationally expensive, sensitive to the selection of hyperparameters
Brain Tumor	M. Sivakumar,	Fuzzy c-means	Can segment	Sensitive to the selection

Detection Using Fuzzy C-Means Clustering and Artificial Neural Networks	S. Anand, and S. Ramakrishnan	clustering and artificial neural networks (ANNs)	tumors of different shapes and sizes, ANNs can learn complex relationships between features	of the number of clusters, ANNs can be prone to overfitting
Brain Tumor Detection Using Deep Belief Networks	Geoffrey E. Hinton, Honglak Lee, Yoshua Bengio, and Aaron Courville	Deep belief networks (DBNs)	Can learn complex relationships between features, DBNs are relatively robust to noise	Computationally expensive to train, DBNs can be prone to overfitting

V. PROPOSED SYSYTEM

The brain tumor detection method using Convolutional Neural Networks (CNN) involves two main phases: training and testing. During training, the computer learns from a labeled dataset of brain MRI scans images, including both normal and tumor-affected brain tissues. The CNN then extracts essential features from these images, identifying unique patterns associated with tumors. The model predicts whether an image contains a tumor or not, continuously refining itself by comparing its predictions to the actual labels, minimizing errors. The CNN analyzes these images, detects tumor-related patterns, and produces an output indicating the presence or absence of a tumor.

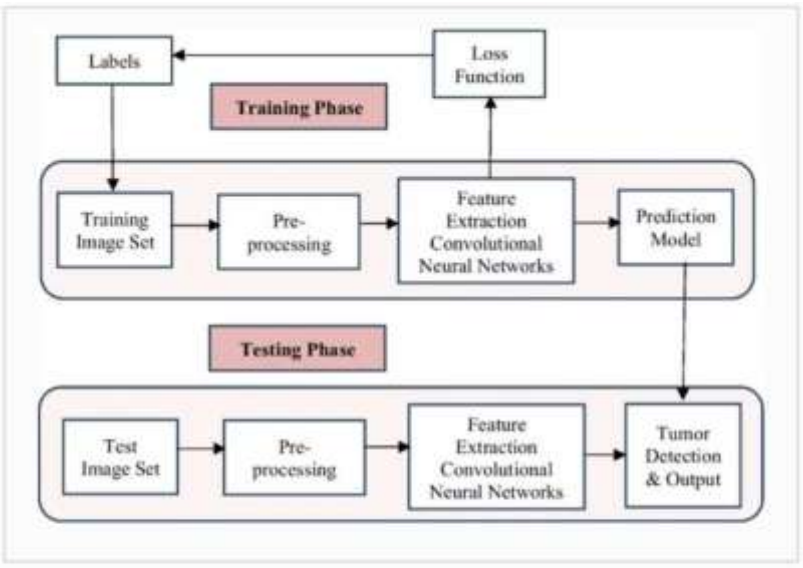

Fig : Proposed Workflow

DATASET COLLECTION

The dataset contains MRI images of the brain which is been taken from the Kaggle website. This dataset contains 3000 images of human brain MRI images which are classified into 4 classes:

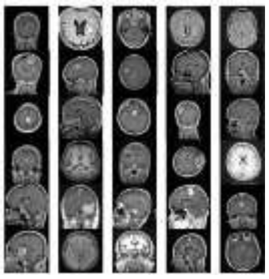
Fig: MRI Scan Dataset

- Glioma
- Meningioma
- Pituitary
- No Tumor Dataset:

https://www.kaggle.com/datasets/sartajbhuvaji/brain-tumor- classification-mri

DATA PREPROCESSING

Preprocessing is a crucial step in preparing image data for

NumPy

VIII. PACKAGES USED

deep learning models. Preprocessing involves refining raw data into a format suitable for training predictive models. It involves a series of operations aimed at enhancing the quality, consistency, and suitability of the data for training.

The preprocessing steps involved in this project begins with resizing, ensuring all images are of a uniform size (150x150 pixels), preventing discrepancies in input dimensions. Brightness adjustment is applied, optimizing image contrast and improving visibility. Data augmentation techniques are then employed, generating additional training samples by applying random transformations such as rotation, shifting, shearing, zooming, and flipping.

VI.HARDWARE AND SOFTWARE REQUIREMENTS HARDWARE REQUIREMENTS:

- Processor: Min. Core i5 processor
- RAM: 4GB (Min.) or 8GB (Recommended)
- Hard Disk Space: 50GB+ SOFTWARE REQUIREMENTS:
- Programming Language: Python
- Operating System: Windows 8 or later versions of windows.

VII.TECHNOLOGY DESCRIPTION

Python

Python is a versatile and easy-to-learn programming language widely used in data science and machine learning projects.

NumPy:

NumPy is a fundamental package for scientific computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of high-level mathematical functions.

TensorFlow:

TensorFlow is an open-source machine learning framework developed by Google. It allows developers to build and train machine learning models, especially deep learning models, efficiently.

Keras:

Keras is an intuitive and user-friendly neural networks API written in Python. It acts as an interface for TensorFlow and other deep learning libraries, simplifying the process of building and training neural networks.

PIL (Python Imaging Library) / Pillow:

PIL, now maintained as Pillow, is a Python library for opening, manipulating, and saving many different image file formats. It is widely used for image processing tasks, including resizing, cropping, and enhancing images.

Matplotlib:

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension, NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits.

NumPy, a powerful Python library, facilitates efficient numerical operations and array manipulations. Widely used in scientific computing, it provides essential tools for tasks like linear algebra, statistical analysis, and mathematical operations.

Matplotlib

Matplotlib, a prominent Python library, empowers data visualization with diverse plotting capabilities. Offering a user-friendly interface, it enables the creation of insightful charts, graphs, and plots for effective data representation.

Optim

Optim, a Python library, plays a crucial role in optimization tasks, offering a rich set of algorithms for mathematical optimization.

IX. ALGORITHM

Step-1: Input:

- Doodle image (content)
- Painting image (style)

Step-2: Preprocess:

- Normalize and resize doodle and painting images

Step-3: Feature Extraction:

- Utilize a pre-trained encoder (e.g., VGG16) for content image features.
- Design a separate encoder for style image features.

Step-4: Model Initialization:

- Initialize a random image or use the content image as the starting point.

Step-5: Loss Function Definition:

- Define content loss to measure the disparity between generated and content features.
- Specify style loss to capture the artistic style difference between generated and style features.
- Combine content and style losses with appropriate weights.

Step-6: Optimization:

- Apply an optimization algorithm (e.g., Adam) to minimize the combined loss.
- Iteratively update the generated image to approach the target style.

Step-7: Training:

- Train the model on a dataset of doodle and painting pairs.
- Update the model parameters through back propagation.

Step-8: Transfer Process:

- Input a new doodle image into the trained model.

Step-9: Generate Painting:

- Optimize the input doodle to minimize content and style losses.

Step-10:Post-processing:

- Adjust brightness, contrast, or apply enhancements if necessary.

Step-11: Output:

- Obtain the generated painting as the final result.

Step-12: Evaluation:

Assess the quality of the generated paintings through visual inspection and user feedback.

Code:

```
import tensorflow import keras import sklearn
from keras.models import Sequential
from keras.layers import Conv2D, Flatten, Dense,
MaxPooling2D, Dropout
from sklearn.metrics import confusion_matrix, classification_report
import io import os
from PIL import Image
from keras.preprocessing.image import ImageDataGenerator
from sklearn.model_selection import train_test_split import cv2
from sklearn.utils import shuffle import numpy as np
import matplotlib.pyplot as plt import seaborn as sns
# Data collection and preprocessing X = []
y = []
image_size = (150, 150)
brightness_factor = 1.5
labels = ['glioma_tumor', 'meningioma_tumor', 'no_tumor', 'pituitary_tumor']
for i in labels:
folderpath =
os.path.join(r'C:\Users\ADITHYA\Downloads\archive\Trai ning', i)
for j in os.listdir(folderpath):
img = cv2.imread(os.path.join(folderpath, j)) # Data Resizing
img = cv2.resize(img, image_size)
img = np.clip(img * brightness_factor, 0, 255).astype(np.uint8)
img = Image.fromarray(img, 'RGB') X.append(np.array(img)) y.append(i)
X = np.array(X) y = np.array(y)

X_train_augmented = [] y_train_augmented = [] data_augmentation_params = {
'rotation_range': 20,
'width_shift_range': 0.1,
'height_shift_range': 0.1,
'shear_range': 0.2,
'zoom_range': 0.2,
'horizontal_flip': True, 'fill_mode': 'nearest'
}
for img, label in zip(X, y): X_train_augmented.append(img) y_train_augmented.append(label) # Data Augmentation image_augmented = img.copy()
augmenter =
ImageDataGenerator(**data_augmentation_params) image_augmented =
augmenter.random_transform(image_augmented) X_train_augmented.append(image_augmented) y_train_augmented.append(label)
X_train_augmented = np.array(X_train_augmented) y_train_augmented = np.array(y_train_augmented)
X = X_train_augmented y = y_train_augmented
# Data Shuffling
X, y = shuffle(X, y, random_state=101)
# Data splitting
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=101)
y_train_new = [] for i in y_train:
y_train_new.append(labels.index(i)) y_train = y_train_new
y_train = tensorflow.keras.utils.to_categorical(y_train, num_classes=4)
y_test_new = [] for i in y_test:
y_test_new.append(labels.index(i)) y_test = y_test_new
y_test = tensorflow.keras.utils.to_categorical(y_test, num_classes=4)
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu',
input_shape=(150, 150, 3)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(2, 2)) model.add(Dropout(0.3)) model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(Conv2D(64, (3, 3), activation='relu')) model.add(Dropout(0.3)) model.add(MaxPooling2D(2, 2))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D(2, 2)) model.add(Dropout(0.3)) model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(Conv2D(256, (3, 3), activation='relu'))
model.add(MaxPooling2D(2, 2)) model.add(Dropout(0.3)) model.add(Flatten()) model.add(Dense(512, activation='relu')) model.add(Dense(512,
activation='relu')) model.add(Dropout(0.3)) model.add(Dense(4, activation='softmax')) model.summary()
model.compile(loss='categorical_crossentropy', optimizer='Adam', metrics=['accuracy'])
history = model.fit(X_train, y_train, epochs=100, validation_split=0.1, verbose=1)
model.save('C:\\Users\\ADITHYA\\Downloads\\archive\\br aintumor1.h5')
# Learning Curves
acc = history.history['accuracy']
val_acc = history.history['val_accuracy'] epochs = range(len(acc))
fig = plt.figure(figsize=(14, 7))
```

```
plt.plot(epochs, acc, 'r', label="Training Accuracy") plt.plot(epochs, val_acc, 'b', label="Validation Accuracy") plt.legend(loc='upper left')

plt.show()

loss = history.history['loss'] val_loss = history.history['val_loss'] epochs = range(len(loss))

fig = plt.figure(figsize=(14, 7)) plt.plot(epochs, loss, 'r', label="Training loss")

plt.plot(epochs, val_loss, 'b', label="Validation loss") plt.legend(loc='upper left')

plt.show()

# Confusion Matrix

y_pred = model.predict(X_test) y_pred_classes = np.argmax(y_pred, axis=1) y_true_classes = np.argmax(y_test, axis=1)

confusion_mtx      =        confusion_matrix(y_true_classes, y_pred_classes)

plt.figure(figsize=(8, 6))

sns.heatmap(confusion_mtx,     annot=True,        fmt='d', cmap='Blues', xticklabels=labels, yticklabels=labels) plt.xlabel('Predicted')

plt.ylabel('True') plt.title('Confusion Matrix') plt.show()

# Classification Report

report    =       classification_report(y_true_classes, y_pred_classes, target_names=labels)

print(report)
```

X. EXPERIMENTAL WORK

Convert doodles to paintings using a CNN with perceptual loss, trained on diverse data. Evaluate with SSI, MSE, and user feedback. Refine iteratively, uphold ethics, and openly share results.

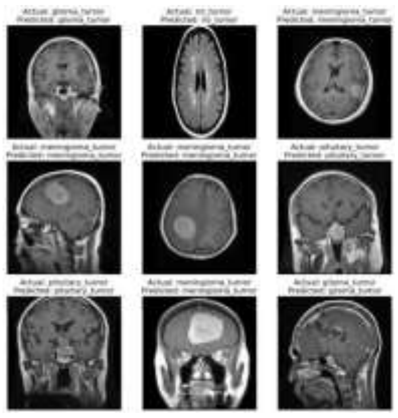

Fig: EXPERIMENTAL RESULTS

XI. DISCUSISON OF RESULTS LEARNING CURVES:

A learning curve is a graph that shows how the performance of a deep learning model changes as it is trained on more data.

There are two main types of learning curves:

- Training curve: The training curve shows how the model's performance on the training data improves as it is trained.
- Validation curve: The validation curve shows how the model's performance on the validation data improves as it is trained.

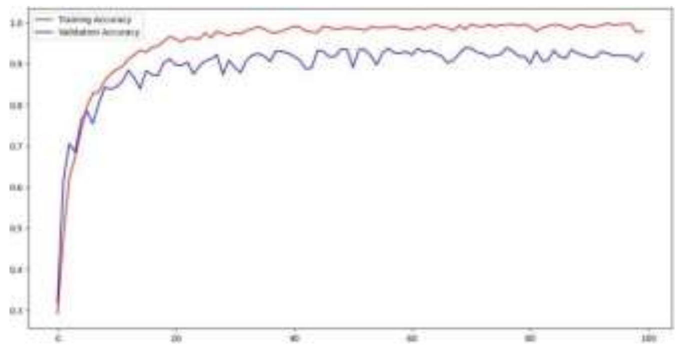

Fig: Training vs Validation accuracy graph

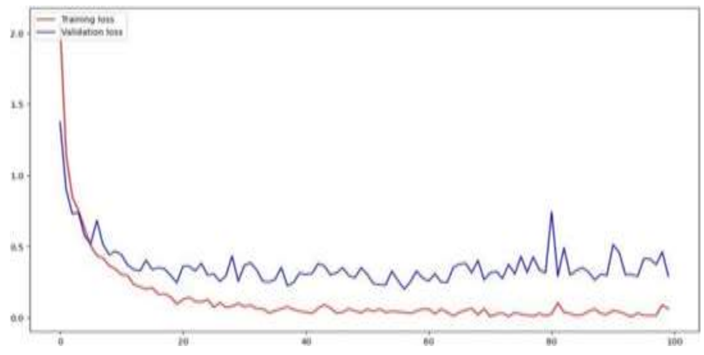

Fig: Training vs Validation Loss graph

CONFUSSION MATRIX

A confusion matrix is a specific table layout that allows visualization of the performance of a classification algorithm. Each row of the matrix represents the instances in a predicted class, while each column represents the instances in an actual class. It's a useful tool for evaluating the performance of a deep learning model, particularly in classification tasks.

XII. TESTING

1. Testing

Testing involves the preparation of test data for individual module assessment and subsequent field validation. System testing ensures the collective functionality of all system components, validating their unified operation.

1.1 SYSTEM TESTING

Testing stands as a fundamental aspect in information technology, inseparable from system and project endeavors.

1.2 MODULE TESTING

To identify errors systematically, each module undergoes individual testing, allowing for error detection and correction without impacting other modules.

1.3 INTEGRATION TESTING

Following module testing, integration testing is implemented to address potential errors that may arise during module linkage.

1.4 ACCEPTANCE TESTING

Upon user confirmation of the system's accuracy without major issues, a final acceptance test is conducted.

XIII. CONCLUSION

Convolutional neural networks (CNNs) have shown promising results in brain tumor detection, achieving high accuracy and sensitivity, even in detecting early-stage tumors. CNNs are able to learn complex patterns in medical images, such as MRIs and CT scans that are difficult for human radiologists to identify. However, training CNN models for brain tumor detection requires large and diverse datasets of labeled medical images. Large-scale, annotated datasets are essential for the optimal training of deep learning models and training the CNN models to reduce the rates of errors and increase accuracy of diagnosis. Despite these challenges, CNNs have the potential to revolutionize brain tumor detection. CNN-based brain tumor detection systems may be able to detect brain tumors without the need for invasive biopsies. This could be beneficial for patients who are at high risk for complications from biopsies. This can lead to better patient outcomes and improved quality of

life. The proposed deep CNN model could classify whether a brain has a tumor or not and if yes it can classify it into one of the 4 types that it was trained on. In this work, a less complicated model is used and the accuracy obtained was around 73%. The future extension to this work includes developing CNN models that can detect a wider range of brain tumors, including rare and aggressive tumors, models that can be used to predict the prognosis of patients with brain tumors and models that can be integrated with other medical imaging modalities, such as PET scans and diffusion tensor imaging, to improve the accuracy and sensitivity of brain tumor detection.

XIV. REFERENCES

[1] Seetha, J & Selvakumar Raja, S. (2018). “Brain Tumor Classification Using Convolutional Neural Networks. Biomedical and Pharmacology Journal”

[2] Parveen and A. Singh, “Detection of brain tumor in MRI images, using combination of fuzzy cmeans and SVM”

[3] G. Raut, A. Raut, J. Bhagade, J. Bhagade and S. Gavhane, "Deep learning approach for brain tumor detection and segmentation", Proc. Int. Conf. Converg. Digit. World Quo Vadis (ICCDW), pp. 1- 5,Feb. 2020.

[4] Amin, J., Sharif, M., Yasmin, M., Fernandes, S. L. (2017). A distinctive approach in brain tumor detection and classification using MRI. Pattern Recognition Letters

[5] Zhang, S., Xu, G. (2016). A novel approach for brain tumor detection using MRI Images. Journal of Biomedical Science and Engineering, 9(10), 44-52

[6] Alfonse, M., Salem, A. B. M. (2016). An automatic classification of brain tumors through MRI using support vector machine. Egy. Comp. Sci. J, 40(3)

[7] Deepak, S., Ameer, P. M. (2019). Brain tumor classification using deep CNN features via transfer learning. Computers in biology and medicine, 111, 10334