NITTE EDUCATION TRUST | N.M.A.M. INSTITUTE OF TECHNOLOGY
(An Autonomous Institution affiliated to Visvesvaraya Technological University, Belagavi)
Nitte – 574 110, Karnataka, India

(ISO 9001:2015 Certified), Accredited with 'A' Grade by
NAAC 08258 - 281039 – 281263, Fax: 08258 – 281265

Department of Computer Science and Engineering

Machine Learning

MINI PROJECT

On

INTRACRANIAL TUMOR DETECTION USING MACHINE LEARNING

Course Code: 20CSE71

Academic Year – 2022-2023

Semester: 6 Section: C

Submitted To,

Course Instructor:

Ms. Soumya Ashwanth
Assistant professor GD-1
Department of CSE,
NMAMIT, Nitte.

Submitted By:

Name: Sharanya Rao USN: 4NM20CS163

Name: Raksha Kamath USN: 4NM20CS143

Date of submission: 08-05-2023
Signature of Faculty

NITTE EDUCATION TRUST | N.M.A.M. INSTITUTE OF TECHNOLOGY
(An Autonomous Institution affiliated to Visvesvaraya Technological University, Belagavi)
Nitte – 574 110, Karnataka, India

(ISO 9001:2015 Certified), Accredited
with 'A' Grade by NAAC 08258 - 281039 -
281263, Fax: 08258 - 281265

Department of Computer Science and Engineering

Certificate

"Intracranial Tumor detection using Machine Learning" is a bonafide work carried out by Sharanya Rao (4NM20CS163) and Raksha Kamath(4NM20CS143) in partial fulfillment of the requirements for the award of Bachelor of Engineering Degree in Computer Science and Engineering prescribed by Visvesvaraya Technological University, Belagavi during the year 2022-2023.

It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report. The Mini project report has been approved as it satisfies the academic requirements in respect of the project work prescribed for the Bachelor of Engineering Degree.

Signature of Guide Signature of HOD

Abstract

Tumors are the second leading cause of cancer. Due to cancer, a large number of patients are in danger. The medical field needs fast, automated, efficient and reliable techniques to detect tumors like brain tumor. Detection plays a very important role in treatment. If proper detection of the tumor is possible then doctors keep a patient out of danger. Various image processing techniques are used in this application. Using this application doctors provide proper treatment and save a number of tumor patients. 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 tumor is a mass of tissue it grows out of control. We can use a Deep Learning architectures CNN (Convolution Neural Network) generally known as NN (Neural Network) to detect the brain tumor. The performance of model is predict image tumor is present or not in image. If the tumor is present it return tumour found otherwise return no tumour found.

Table of Contents

Introduction

All the other organs that make up the human body are secondary to the brain in importance and necessity. One of the common reasons for neurological impairment is a brain tumor. A tumor is made up of cells that are overabundant and continue to multiply unchecked. Brain failure happens when the nutrients meant for healthy cells and tissues are eventually fully absorbed by brain cancer cells as a result of their expansion. Currently, clinicians manually determine the location and size of the brain tumor using the MRI scans of the patient's brain. This is considered to be incredibly time-consuming and results in an unreliable cancer detection. Many people die each year from this awful disease known as brain cancer.

A technique for the detection and classification of brain tumors has been created to aid in the early diagnosis of these diseases. The categorization of cancer is the clinical diagnosis's most challenging problem. The objective of this research is to create a system that, using convolution neural network methods, can recognise tumor blocks and classify the type of cancer from MRI scans of distinct patients. Using a range of image processing techniques, such as picture segmentation, image enhancement, and feature extraction, brain tumor identification in MRI scans of cancer patients is accomplished. The four steps involved in identifying a brain tumor using image processing methods are picture pre-processing, image segmentation, feature extraction, and classification. Image processing and neural network techniques are used to improve object detection performance.

The performance of detecting and categorizing brain tumors in MRI images is improved by using image processing and neural network approaches.

Literature Review

Sl.no	Topic	Journal Publication Year	Objectives	Methods	Scope of Improvement/ Challenges	Conclusion
1	Machine learning imaging applications in the differentiation of true tumour progression from treatment-related effects in brain tumours: A systematic review and meta-analysis	22 May 2022	Chemotherapy and radiotherapy can produce treatment-related effects, which may mimic tumour progression. Advances in Artificial Intelligence (AI) offer the potential to provide a more consistent approach of diagnosis with improved accuracy. The aim of this study was to determine the efficacy of machine learning models to differentiate treatment-related effects (TRE), consisting of pseudoprogression (PsP) and radiation necrosis (RN), and true tumour progression (TTP)..	The systematic review was conducted in accordance with PRISMA-DTA guidelines. Searches were performed on PubMed, Scopus, Embase, Medline (Ovid) and ProQuest databases. Quality was assessed according to the PROBAST and CLAIM criteria. There were 25 original full-text journal articles eligible for inclusion.	Our review has a number of limitations that affect the strength of the analysis. There was a small number of studies in three of the four categories investigated. Furthermore, there was substantial heterogeneity between studies, with studies varying in the imaging modalities used, ML methods and tumour grades. There was also a lack of consistency in the definitions of PsP and RN. This highlights the lack of consensus surrounding the reporting of the methodology and results in AI studies in medical imaging. The application of CLAIM for further studies will prove valuable for the consistency in reporting required for the comparative analysis of ML papers. Additionally, articles that combined metastasis and gliomas were excluded, which	TRE can be distinguished from TTP with good performance using machine learning-based imaging models. There remain issues with the quality of articles and the integration of models into clinical practice. Future studies should focus on the external validation of models and utilize standardized criteria such as CLAIM to allow for consistency in reporting.

					limited the number of studies available for meta-analysis; however, this was necessary to reflect clinical practice more accurately. Future research looks promising for the integration of AI methodologies into clinical practice. Efforts are already being made by the development of guidelines such as the CONSORT-AI extension which aims to streamline the reporting of clinical trials involving AI.	
2	Tumor Analysis Using Deep Learning and VGG-16 Ensembling Learning Approaches	23 November 2021	This study aimed to critically analyze the proposed literature solutions, use the Visual Geometry Group (VGG 16) for discovering brain tumors, implement a convolutional neural network (CNN) model framework, and set parameters to train the model for this challenge. VGG is used as one of the highest-performing CNN models because of its simplicity. Furthermore, the study developed an effective	Our suggested strategy uses CNN and VGG 16 to detect brain tumors employing brain MRI data. The brain MRI image was used as the primary input image. Data operations, including thresholding and refractive error, were carried out to reduce noise. The database of brain MRI images was analyzed and improved. The images were then resized for use as input to the model. A VGG 16 pre-trained convolution layer was used to improve and increase classification accuracy into two classes: yes and	The limitation of the research lies in its exclusive focus on brain tumors that can be extended to account for different types of cancer attacks in MRI images. Future research can make use of a variety of image modalities and diverse segmentation techniques to acquire the best approximation of affected regions in the brain to isolate these regions from unaffected parts of the brain. Different modalities having image registration distinctions from each other could be utilized for presenting the missing image	The proposed approach for brain tumor diagnosis was based on deep learning, and brain magnetic resonance was used for tumor evaluation using automated classification methods. For image improvement and better classification, a CNN model was used in this paper. The study successfully yielded better outcomes while requiring less computing time. Our method was used to identify brain tumors in MR images. The

			approach to detect brain tumors using MRI to aid in making quick, efficient, and precise decisions. Faster CNN used the VGG 16 architecture as a primary network to generate convolutional feature maps, then classified these to yield tumor region suggestions. The prediction accuracy was used to assess performance.	no. VGG 16 is a significantly improved performance VGGNet variant among the most advanced classification networks.depicts the structure of the proposed approach.	features in the fixed image and conducting the best classification. To achieve higher precision and accuracy, ensembles could be further used. .	algorithm significantly outperformed previously studies. The methodologies used for detecting brain tumors in the testing data (Precision = 96%, 98.15%, 98.41%, and F1-score = 91.78%, 92.6%, and 91.29%) achieved high accuracy of CNN 96%, VGG 16 98.5%, and Ensemble Model is 98.14%. The reliability of the validation and learning was discovered to be increasing, and the findings higher. They could be utilized to diagnose the existence of a tumor in the brain..

Design

The planned project design will involve the below listed functionalities:

1. Data collection and Preprocessing: The first step in project design to develop Intracranial tumor detection system is to collect and preprocess the dataset. The dataset was obtained from Kaggle, and has 2 folders yes and no. The folder yes contains 1500 Brain MRI Images that are tumorous and The folder no contains 1500 Brain MRI Images that are non-tumorous. Most of the images in the dataset are of jpg format. Before using the data for training and testing the models, we will preprocess it by cleaning and normalizing the data. The cleaning includes resizing of the scan images in the dataset and then normalizing it.
2. Feature extraction: Using numpy array we are extracting the grid values in the form of a matrix and labeling them as 1 and 0 corresponding to yes and no.
3. Model Selection and training: In this project we've used CNN algorithm to predict if the images have tumor or not with Keras library. Next we we split the dataset into 80 to 20, and kept 80% for training and 20% for testing, x_train and x_test for the dataset numpy array and y_train and y_test for the label numpy array followed by training the model in using fit() method.
4. Model Evaluation: Based on the accuracy we say if the model is accurate or not, if it's feasible for the dataset.
5. Prediction: Predicting whether the input images are with or without tumor based on our model.

Implementation

- Modules and packages:
    - cv2
    - tensorflow
    - keras
    - PIL
    - numpy
    - sklearn
    - matplotlib

- Programming Language used: Python

- Editor used: Jupyter Notebook

Model creation and training the dataset according to CNN algorithm:

We have used CNN algorithm to create our model using Keras; these are the following steps:

Convolutional neural networks (CNNs) are a type of deep learning model that are widely used for image classification tasks. They typically consist of multiple layers, including convolutional layers, pooling layers, and fully connected layers.

In order to build a CNN using Keras, we need to import several classes and functions from the Keras library:

- Sequential: This is a Keras class that allows us to create a neural network model by simply adding layers to it in sequence. We can think of the model as a "pipeline" that processes the input data by passing it through each layer in turn.
- Conv2D: This is a Keras layer class that performs a 2D convolution operation on the input data. Convolutional layers are the key building blocks of CNNs, and they are used to learn spatial features from the input images.
- MaxPooling2D: This is a Keras layer class that performs max pooling on the output of a convolutional layer. Max pooling is used to reduce the spatial dimensions of the feature maps, which helps to reduce the number of parameters
  in the model and prevent overfitting.
- Activation: This is a Keras layer class that applies an activation function to the output of a layer. Activation functions introduce non-linearity into the model and help it to learn more complex features.
- Dropout: This is a Keras layer class that randomly drops out some of the neurons in a layer during training. Dropout is a regularization technique that helps to prevent overfitting.
- Flatten: This is a Keras layer class that flattens the output of a convolutional layer into a 1D vector. This is necessary in order to pass the output to a fully connected layer.
- Dense: This is a Keras layer class that represents a fully connected neural network layer. Fully connected layers are typically used at the end of a CNN to perform the final classification.

- to_categorical: This is a utility function that converts integer class labels into one-hot encoded vectors. One-hot encoding is a common way to represent categorical data in deep learning models.

By importing these classes and functions, we can use Keras to easily build and train a CNN for image classification tasks.

As seen in the snippets above; we start with model=Sequential(). This initializes an empty neural network model that we can then add layers to in order to build a deep learning architecture.

In [15]:
```
model.add(Conv2D(32, (3,3), input_shape=(INPUT_SIZE, INPUT_SIZE, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
```

In [16]:
```
model.add(Conv2D(32, (3,3), kernel_initializer='he_uniform'))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
```

In [17]:
```
model.add(Conv2D(64, (3,3), kernel_initializer='he_uniform'))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
```

The above code snippet builds a convolutional neural network (CNN) with three convolutional layers followed by max pooling layers.

- Conv2D(32, (3,3), input_shape=(INPUT_SIZE, INPUT_SIZE, 3)): This adds a 2D convolutional layer with 32 filters of size 3x3 to the model, with the input shape specified as (INPUT_SIZE, INPUT_SIZE, 3), which corresponds to the shape of the input images.
- Activation('relu'): This adds a ReLU activation function to the output of the previous layer.

- MaxPooling2D(pool_size=(2,2)): This adds a max pooling layer with a pool size of 2x2, which reduces the spatial dimensions of the output of the previous layer by a factor of 2.

These three layers are then repeated twice more (forming 9 layers), with the number of filters in the convolutional layers increased to 32 and 64, respectively, and the kernel initializer or the weight initializer set to 'he_uniform'. The 'he_uniform' initializer is a type of weight initialization method that helps to prevent vanishing gradients during training by initializing the weights in a way that keeps the variance of the activations roughly constant across layers. This can help to improve the performance of deep neural networks.

model.add(Flatten()) : flattens the output of the last convolutional layer into a 1D vector, which can then be fed into a fully connected layer.

model.add(Dense(64)) : adds a fully connected layer with 64 neurons.

model.add(Activation('relu')) applies the ReLU activation function to the output of the fully connected layer.

model.add(Dropout(0.5)) : applies dropout regularization with a probability of 0.5. Dropout is a regularization technique that randomly drops out (i.e., sets to zero) some of the output values from the previous layer during training, which can help prevent overfitting.

model.add(Dense(2)) : adds a final fully connected layer with 2 neurons, which corresponds to the two classes (tumor and no tumor) in the binary classification problem.

model.add(Activation('softmax')) : applies the softmax activation function to the output of the final layer, which converts the outputs into probabilities that sum up to 1. This allows the model to output the probability of the input image belonging to each of the two classes.

From the code snippet;

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

After defining the architecture of a neural network using the Sequential() or functional API with Keras we compiled the model using the compile() method before training it.

The arguments are:

- loss: This argument specifies the loss function that the model will use to evaluate how well it is performing during training. In this case, categorical_crossentropy is used as the loss function for multi-class classification problems with one-hot encoded labels.
- optimizer: This argument specifies the optimization algorithm that the model will use to update the weights of the network during training. In this case, the adam optimizer is used, which is a popular optimization algorithm that adapts the learning rate dynamically during training.

- metrics: This argument is used to specify the evaluation metric(s) that the model will use to measure its performance during training and testing. In this case, the model will evaluate its accuracy, which is the fraction of correctly classified examples out of all the examples in the dataset.

```
model.fit(x_train, y_train,
batch_size=16,
verbose=1, epochs=10,
validation_data=(x_test, y_test),
shuffle=False)
```

From the above snippet; after compiling we train the model on a dataset using the fit() method.

The arguments used for this method are:

- x_train: This argument specifies the input data for training the model. It should be a Numpy array or a tuple of Numpy arrays if the model has multiple inputs.
- y_train: This argument specifies the target labels for the training data. It should be a Numpy array or a tuple of Numpy arrays if the model has multiple outputs.
- batch_size: This argument specifies the number of samples per gradient update. It controls how often the model's weights are updated during training.
- verbose: This argument specifies how much output the method will produce during training. A value of 0 means silent, 1 means progress bar, and 2 means one line per epoch.
- epochs: This argument specifies the number of times the entire training dataset will be used to train the model.

- validation_data: This argument specifies the data on which to evaluate the loss and any model metrics at the end of each epoch. It should be a tuple of the form (x_test, y_test).
- shuffle: This argument specifies whether to shuffle the training data at the beginning of each epoch.

```
In [25]: loss, accuracy = model.evaluate(x_test, y_test, verbose=0)
         print('Test accuracy:', accuracy)

         Test accuracy: 0.9783333539962769
```

The created model gives an accuracy of 97.83%. Since this a good accuracy rate, the model we created using CNN algorithm is a good model.

Testing the images with the trained dataset to predict of there's tumor or not:

Libraries used here:

- cv2: This is the OpenCV library for computer vision and image processing.
- load_model from keras.models: This is a function from the Keras API that loads a pre-trained neural network model.
- Image from PIL: This is the Python Imaging Library for opening and manipulating image files.
- numpy as np: This is the NumPy library for numerical computing in Python.
- pyplot from matplotlib: This is a module from the Matplotlib library for data visualization.
- Next we load the created model into a variable called model in a new jupyter notebook, test.py.

- Next we put the path of a random image from the dataset into a variable called image using openCV library cv2; this is our input image to test if the brain scan image has tumor or not.
- Next we create a PIL.Image object from a NumPy array of image data ie; image is assumed to be a NumPy array representing an image in memory, and Image.fromarray() is used to create a PIL.Image object from this array.
- Resize this image and put the np.array of the image into a variable called img.

```
In [29]: result = model.predict(input_img)
         predicted_class = np.argmax(result, axis=-1)
         #print(predicted_class)

1/1 [==============================] - 0s 179ms/step
[0]

In [30]: plt.imshow(image)

         if predicted_class == 0:
             print("No tumor found")
         else:
             print("Tumor found")

No tumor found
```

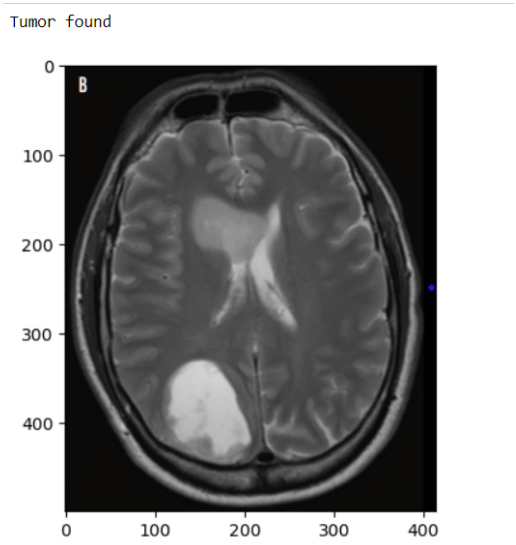
- result = model.predict(input_img):
  This line of code uses a trained machine learning model (specifically, a keras model) to make a prediction on an input image. The input image (input_img) is a NumPy array of pixel values representing an image. The output of this prediction is a NumPy array (result) that contains the predicted probabilities of each class.
- predicted_class = np.argmax(result, axis=-1):
  This line of code takes the output of the prediction (result) and finds the class with the highest predicted probability. This is done using the np.argmax() function, which returns the index of the maximum value in a NumPy array along a specified axis. In this case, we are using axis=-1 to find the maximum value in the last dimension of the result, which corresponds to the predicted probability of each class.

- The resulting index (predicted_class) is the predicted class label for the input image.
- If the image has a tumor, then it'll return [1] otherwise it'll return [0].

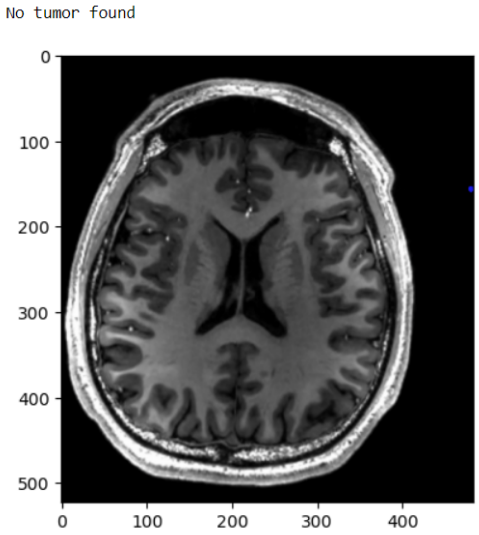
Results

An MRI image from dataset used predict if the image has a brain scan with tumor or without tumor:

Test Image with tumor predicted as:



Test image without tumor predicted as:



Conclusion

In brain tumor classification using machine learning, we built a model to detect brain tumors from MRI scan images and used Categorical cross-entropy.
A major challenge for brain tumor detection arises from the variations in tumor location, shape, and size also for absolute safety that cannot be directly applied and cannot escape human expertise verification.

One of the key difficulties in using the machine learning-based automated detection of brain tumor is the requirement for a substantial amount of annotated images collected by a qualified physician or radiologist. In order to make a robust deep learning model, we would require a large dataset. To the best of our knowledge, the majority of contemporary machine learning tools for medical imaging have this constraint. Although the majority of earlier studies are currently making their datasets available to the public in an effort to address this problem. Still, the amount of properly and accurately annotated data is still very limited.

Reference

- https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9545346/
- https://www.mdpi.com/2076-3417/12/14/7282
- https://data-flair.training/blogs/brain-tumor-classification-machine-learning/
- https://www.mayoclinic.org/diseases-conditions/brain-tumor/symptoms-causes/syc-20350084
- https://www.kaggle.com/datasets/ahmedhamada0/brain-tumor-detection