EAI Endorsed Transactions
on Pervasive Health and Technology
Research Article EAI.EU

A Novel Approach to Identify the Brain Tumour Using Convolutional Neural Network

Suraj Khari[1, *], Deepa Gupta[2], Alka Chaudhary[3], Ruchika Bhatla[4]

[1, 2, 3, 4] Amity University, Noida, Uttar Pradesh, India

Abstract

INTRODUCTION: Determining the possibility that an individual is affected by a tumour is an intricate process in today's modern technological and biological age, when feats are reaching unprecedented levels with every passing second. Machine learning modalities could dramatically enhance the accuracy of diagnosis.
OBJECTIVES: Our research makes it feasible to detect tumours early, aiding in early diagnosis, and is a necessity for the curative efforts of cancer patients.
METHODS: In our research model Convolutional Neural Network (CNN) was implemented using Jupiter to give an accurate result.
RESULTS: In our proposed model we got 99% accuracy that is higher than the other results.
CONCLUSION: Our research demonstrates the potential of using machine learning techniques to improve the accuracy and efficiency of medical diagnosis.

Keywords – CNN, brain tumour, MRI images

Received on 10 September 2023, accepted on 04 November 2023, published on 08 November 2023

doi: 10.4108/eetpht.9.4337

[1]Corresponding author. Email: surajkhari79@gmail.com

1. Introduction

An uncontrolled and abnormal proliferation of brain cells is referred to as a brain tumour. The human skull is a hard, density-limited body; depending on the section of the brain involved, any unanticipated development may prove harmful to human functions. Additionally, it will be allowed to spread to other bodily organs, further compromising human activities. One of the main causes of mortality around the globe is cancer. Only 1.4% of all new cancer cases are brain tumours, making them an uncommon kind of cancer. However, they can be very fatal, with only around 36% of survivorship after five years. Headaches, seizures, changes in vision or hearing, and trouble balancing or controlling oneself are just a few of the symptoms of brain tumours that can vary greatly depending on where they are located and how big they are. The Central Brain Tumour Registry of the United States (CBTRUS) predicts that among adults and children in the U.S. in 2021, 84,170 new cases of primary malignant (cancerous) and non-malignant (non-cancerous) brain and other CNS tumours will be discovered. This comprises an estimated 59,040 benign and 25,130 primary tumours of the brain and other CNS.

MRI (Magnetic Resonance Imaging) is an imaging technique extensively used to diagnose patients and treat tumours in clinical practise. Due to the large quantity of data involved, manually segmenting tumours consisting of MRI images or lesions is a time-consuming, difficult, and stressful process.

One of the oldest and most effective models to capture uncertainty in data is artificial neural networks (ANN). After the deep learning idea was introduced to ANN, deep neural network (DNN) models were suggested [1, 2]. DL's dense network of layers processes raw inputs both linearly and nonlinearly, layer by layer, in an ordered fashion. The broad category that DL falls under is representational learning. There are now a number of

DNN variations available. Convolutional neural networks (CNN) are one of them and have been used as a very effective model for image categorization [3].

CNNs are part of deep learning (DL) algorithms that are capable of automatically learning features from input images. They are accurate in classifying complicated patterns and characteristics found in medical images. CNNs have been demonstrated to be highly accurate in identifying brain tumours in MRI images. CNN uses data that is organised in a grid-like fashion. A two-dimensional grid that represents a multi-dimensional matrix called an image can be used to explain it. As a result, a picture is thought of as consisting of all the raw pixels that make up its characteristics. At its most basic level, a CNN may be thought of as an ANN that computes the total input in at least one of the layers using convolution rather than the usual ANN method of matrix multiplication. It has been successfully used to solve a number of real-world issues, such as text-based picture retrieval [4], audio signal classification, investigation of gene traits [5], computational biology [6], segmenting MRI of the brain [7], diabetic retinopathy [8], COVID-19 detection [9], and many others.

By switching the last layer parameter to SoftMax and the optimizer to RMSprop, Chattopadhyay and Maitra's method [10] for splitting brain tumours from MRI data attained an accuracy of 99.74%. By building a deep CNN with weights that had previously been trained on ImageNet using a weighted loss function, Tazin et. al. [11] employed transfer learning. As a result, they were able to achieve an F1-score of 92% and a classification accuracy of 92% on the test set. Sajid et. al. [10] suggested a CNN design that considers local and contextual data. In their method, they normalised the images through pre-processing and abolished false positives by means of post-processing. The obtained values for glioma detection accuracy and specificity are 0.86 and 0.91, respectively. Özyurt et. al. [11] suggest a hybrid approach for the identification of brain tumours that combines CNN and Neutrosophy. The accuracy of the approach, which is stated to be 95.62%, is reported to be superior to that of conventional CNN, SVM, and K-nearest neighbours (KNN). Mohsen et. al. [12] used the fuzzy segmentation technique (FCM) to distinguish between brain areas with and without tumours. A discrete wavelet transform (DWT) on many levels was also used to extract wavelet characteristics. Finally, deep neural networks (DNNs) were used to accurately classify brain cancers. The effectiveness of this method was evaluated against that of the KNN classifier, Linear Discriminant Analysis (LDA), and Sequential Minimum Optimisation (SMO). The DNN-based brain tumour classification analysis has a 96.97 percent accuracy rate. However, there was a huge disparity in performance and complexity.

A novel biomechanical model of cancer growth was introduced in [13] for a detailed examination of patient tumour progression. It was used to capture a major tumour impact in gliomas and distinct fringed solid tumours. To simulate cancer development, discrete and continuous techniques were merged. The suggested technique offers the potential for implicit segmentation of tumour-bearing brain pictures from atlas-based registries. Brain tissue was mostly segmented using this method. But the computation took a long time. In [14], a novel multi-feature feature (Multi FD) was utilised, and the AdaBoost classification system for the identification and segmentation of brain tumours was enhanced. Using the multi-FD feature extraction approach, the structures of brain tumour tissue were extracted. The categorization of the contributed brain tissue as cancer or non-tumour was done using advanced AdaBoost. The data set utilised in [15], which employed BRATS 2015, was a convolutional neural network as the method. In [16], ANN and CNN were employed as two distinct methods. CNN is more accurate than ANN, according to the final finding.

2. Materials and Methods

2.1. Dataset

The open-source database Kaggle was used to get the data. Both brain tumour patients and healthy people's X-ray scans were included in the collection. This collection includes photos of brain X-rays taken of patients with brain tumours. In this collection, there are 1200 healthy photographs and 1000 images of brain tumours.

2.2. Tools

Jupyter Notebook and Google Colab were the technologies selected for the experiment, and both were utilised alongside Python and its libraries. The interactive and user-friendly interface of Jupyter Notebook made it simple to create and test our code. However, Google Colab gave access to a cloud-based platform with strong computing capabilities that were particularly helpful for managing huge datasets. The diverse library ecosystem of Python also made it possible to use tools like NumPy, Pandas, Matplotlib, and Seaborn for data analysis and visualisation.

2.3. Pre-processing

Obtaining, classifying, and preparing data is a crucial phase in machine learning projects, especially in applications involving medical imaging. In this instance, a platform for sharing and analysing machine learning datasets, Kaggle, provided the dataset of 2000 MRI images.

The dataset was split into two categories—images with tumours and images without tumours—to make it easier to classify MRI images. Because it aids in the supervised learning of machine learning models, this categorisation was essential. The next step was to use suitable image processing software to scale each image to a standard size of (128, 128). It is crucial for images to be a consistent size so that the machine learning algorithm can quickly analyse them and extract useful information.

The resized photographs were then saved in arrays for each category separately to aid in the training of a machine learning model. Using a suitable ratio, in this case 80% of the images for training and 20% for testing, the data was then split into training and test sets.

The data needed to be pre-processed for it to be suitable for additional analysis, such as training machine learning algorithms to categorise MRI pictures as either having tumours or not. This method of pre-processing the data makes it possible for the machine learning model to learn from the dataset and produce precise predictions for brand-new, untested data.

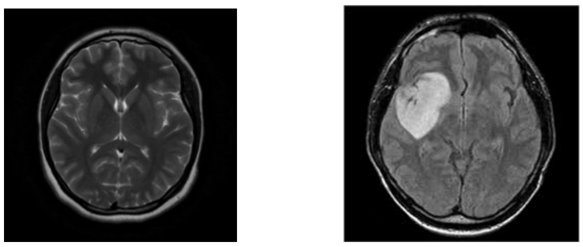


Figure 1. MRI of healthy brain and brain tumour

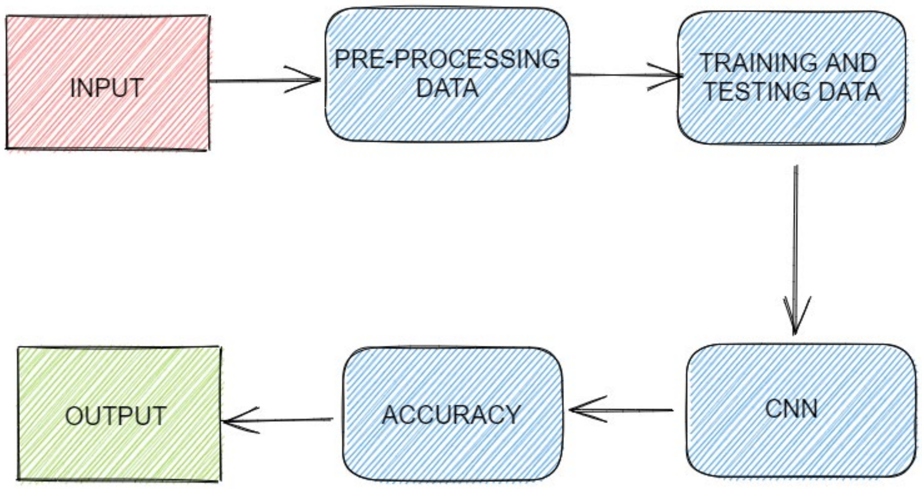


Figure 2. Workflow

2.4. Convolutional Layer

The convolutional layer, which is a CNN's initial layer, applies alterations to the input picture to extract pertinent information. The dot product between each filter and each input pixel is computed by moving the filter's minuscule matrix across the input picture. A feature map—the output of the convolutional layer—is the culmination of the process. The convolutional layer can capture spatial information in the input image, making it an effective tool for detecting patterns and edges. By using multiple filters with different parameters, the convolutional layer can detect a wide range of features, such as lines, curves, and textures.

2.5. Pooling Layer

The convolutional layer's feature maps are downcycled using the pooling layer. It reduces the number of parameters in the model, making it more efficient to compute and reducing the risk of overfitting. The most prevalent type of pooling layer we used is max pooling, which emits the maximal value found in a restricted spectrum of the feature map. The primary goal of this layer so that it can preserve the most prominent features in the image while reducing the dimensionality of the data.

2.6. Batch Normalization

We employed batch normalisation to enhance the effectiveness and stability of the training processes in the convolutional neural network. By taking the batch mean away and dividing by the batch standard deviation, it normalises the inputs of a layer. This aids in minimising the internal covariate shift that happens during training and might delay the convergence of the network.

2.7. Dropout Layer

This layer is used as a regularisation method that guards against model overfitting. During every trial iteration, a predetermined percentage of the network's neurons are randomly removed, driving the network to gradually acquire reliable and universal assets. By dropping out neurons, the dropout layer encourages the network to rely on multiple pathways to extract features from the input, reducing the risk of overfitting to specific features in the training data.

2.8. Dense Layer

We implemented a dense layer that performed the final classification of features by the preceding convolutional and pooling layers. A vector of scores or probabilities representing the anticipated class labels or regression results for the input picture is produced by the dense layer using this tensor as its input. It performs high-level abstraction and combines the learned features into a final output that is appropriate for the specific experiment, such as object identification, picture segmentation, or regression, by adding a dense layer at the end of the CNN.

2.9. Flatten Layer

Another crucial layer in a CNN is the flatten layer, which allows the multi-dimensional tensor output of the convolutional or pooling layer to be converted into a 1D feature vector that is used by a fully connected layer for additional processing. The main purpose of using a flatten layer is to reshape the multi-dimensional image into a single dimension. The input picture is subjected to a succession of convolutional and pooling processes before being passed on to one or more fully connected layers, which conduct the ultimate classification or regression. The fundamental formula for a CNN is as follows:

$$\mathbf{Y = FC\ (W2 * maxpool\ (W1 * X + b1) + b2)}$$

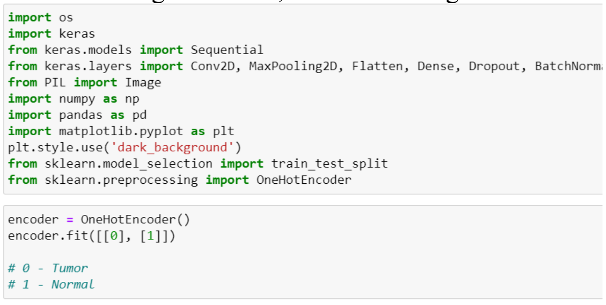
- X is the input image, represented as a matrix of pixel values.
- The weight matrix for the top convolutional layer is designated as W1.
- b1 is the bias vector for the first convolutional layer.
- Maxpool () is the max pooling operation, which reduces the spatial dimensions of the feature maps.
- W2 is the weight matrix for the second fully connected layer.
- b2 is represents bias vector for the second fully connected layer.
- FC () is the fully connected operation, which performs the final classification or regression.
- Y is the output of the network, which represents the predicted class label or regression value.

Table 1. CNN architecture

Layer(type)	Output Shape	Param
Conv2d	(None,128,128,32)	416
Conv2d	(None,128,128,32)	4128
Batch Normalization	(None,128,128,32)	128
Max Pooling	(None,64,64,32)	0
Dropout	(None,64,64,64)	0
Con2d	(None,64,64,64)	8256
Con2d	(None,64,64,64)	16448
Batch Normalization	(None,64,64,64)	256
Max Pooling	(None,32,32,64)	0
Dropout	(None,32,32,64)	0
Flatten	(None,65536)	0
Dense	(None,512)	33554944
Dropout	(None,512)	0
Dense	(None,2)	1026

3. Experiment

A. First, we imported the different libraries required for building a model and setting the encoder. Each library was imported to perform a specific function in building the model, as shown in Figure 3.

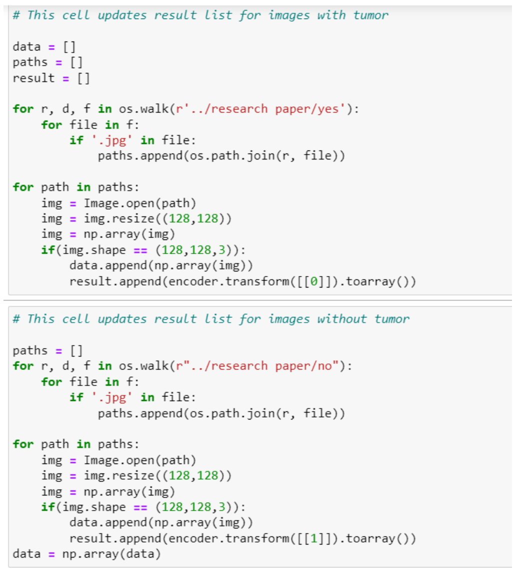


```
import os
import keras
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout, BatchNorm
from PIL import Image
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.style.use('dark_background')
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
```

```
encoder = OneHotEncoder()
encoder.fit([[0], [1]])

# 0 - Tumor
# 1 - Normal
```

Figure 3. Importing different libraries

B. Then, we created three important lists: a data list for storing image data in NumPy array form; a paths list for storing paths of all images; and a result list for storing one hot encoded form of the target class, whether normal or tumour, shown in Figure 4.

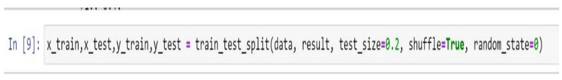


```
# This cell updates result list for images with tumor

data = []
paths = []
result = []

for r, d, f in os.walk(r'../research paper/yes'):
    for file in f:
        if '.jpg' in file:
            paths.append(os.path.join(r, file))

for path in paths:
    img = Image.open(path)
    img = img.resize((128,128))
    img = np.array(img)
    if(img.shape == (128,128,3)):
        data.append(np.array(img))
        result.append(encoder.transform([[0]]).toarray())
```

```
# This cell updates result list for images without tumor

paths = []
for r, d, f in os.walk(r"../research paper/no"):
    for file in f:
        if '.jpg' in file:
            paths.append(os.path.join(r, file))

for path in paths:
    img = Image.open(path)
    img = img.resize((128,128))
    img = np.array(img)
    if(img.shape == (128,128,3)):
        data.append(np.array(img))
        result.append(encoder.transform([[1]]).toarray())
data = np.array(data)
```

Figure 4. Creating data list, paths list and result list

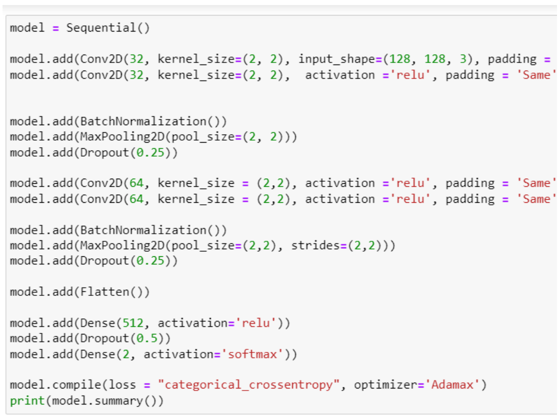
C. Next, we split the data set into training and test sets. The data was divided into two parts: one was used to train the model, and the other was used to evaluate the accuracy of the model shown in Figure 5.



```
In [9]: x_train,x_test,y_train,y_test = train_test_split(data, result, test_size=0.2, shuffle=True, random_state=0)
```

Figure 5. Splitting the data into training and test set

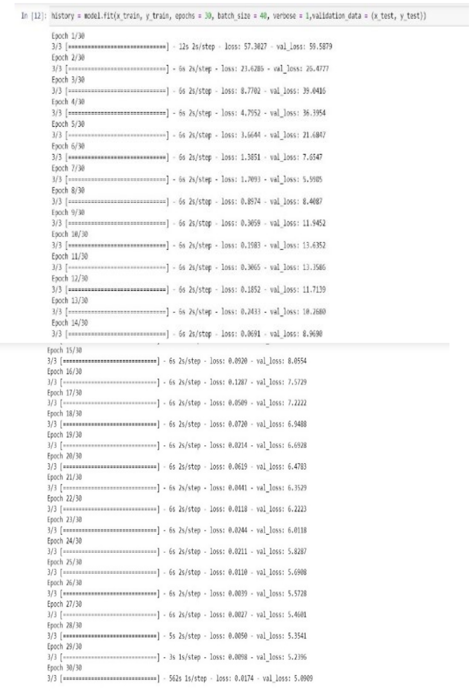
D. Then, we built a model for predicting the tumour. The model was built using the Keras library. Different layers were imported and used inside the model shown in Figure 6.



```
model = Sequential()

model.add(Conv2D(32, kernel_size=(2, 2), input_shape=(128, 128, 3), padding =
model.add(Conv2D(32, kernel_size=(2, 2),  activation ='relu', padding = 'Same'


model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(64, kernel_size = (2,2), activation ='relu', padding = 'Same'
model.add(Conv2D(64, kernel_size = (2,2), activation ='relu', padding = 'Same'

model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))
model.add(Dropout(0.25))

model.add(Flatten())

model.add(Dense(512, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(2, activation='softmax'))

model.compile(loss = "categorical_crossentropy", optimizer='Adamax')
print(model.summary())
```

Figure 6. Building model for predicting tumour

E. Then we trained the model using a training set. The model was trained using a specific data set known as the training dataset, which was part of the original data set as shown in Figure 7.



```
In [12]: history = model.fit(x_train, y_train, epochs = 30, batch_size = 40, verbose = 1,validation_data = (x_test, y_test))

Epoch 1/30
3/3 [==============================] - 12s 2s/step - loss: 57.3027 - val_loss: 59.5879
Epoch 2/30
3/3 [==============================] - 6s 2s/step - loss: 23.6285 - val_loss: 25.4777
Epoch 3/30
3/3 [==============================] - 6s 2s/step - loss: 8.7702 - val_loss: 39.0416
Epoch 4/30
3/3 [==============================] - 6s 2s/step - loss: 4.7952 - val_loss: 36.3954
Epoch 5/30
3/3 [==============================] - 6s 2s/step - loss: 3.6644 - val_loss: 21.6847
Epoch 6/30
3/3 [==============================] - 6s 2s/step - loss: 1.3851 - val_loss: 7.6547
Epoch 7/30
3/3 [==============================] - 6s 2s/step - loss: 1.7093 - val_loss: 5.5985
Epoch 8/30
3/3 [==============================] - 6s 2s/step - loss: 0.8974 - val_loss: 8.4087
Epoch 9/30
3/3 [==============================] - 6s 2s/step - loss: 0.3059 - val_loss: 11.9452
Epoch 10/30
3/3 [==============================] - 6s 2s/step - loss: 0.1983 - val_loss: 13.6352
Epoch 11/30
3/3 [==============================] - 6s 2s/step - loss: 0.3065 - val_loss: 13.3586
Epoch 12/30
3/3 [==============================] - 6s 2s/step - loss: 0.1852 - val_loss: 11.7139
Epoch 13/30
3/3 [==============================] - 6s 2s/step - loss: 0.2433 - val_loss: 10.2680
Epoch 14/30
3/3 [==============================] - 6s 2s/step - loss: 0.0691 - val_loss: 8.9690
Epoch 15/30
3/3 [==============================] - 6s 2s/step - loss: 0.0920 - val_loss: 8.0554
Epoch 16/30
3/3 [==============================] - 6s 2s/step - loss: 0.1287 - val_loss: 7.5729
Epoch 17/30
3/3 [==============================] - 6s 2s/step - loss: 0.0509 - val_loss: 7.2222
Epoch 18/30
3/3 [==============================] - 6s 2s/step - loss: 0.0720 - val_loss: 6.9488
Epoch 19/30
3/3 [==============================] - 6s 2s/step - loss: 0.0214 - val_loss: 6.6928
Epoch 20/30
3/3 [==============================] - 6s 2s/step - loss: 0.0619 - val_loss: 6.4783
Epoch 21/30
3/3 [==============================] - 6s 2s/step - loss: 0.0441 - val_loss: 6.3529
Epoch 22/30
3/3 [==============================] - 6s 2s/step - loss: 0.0118 - val_loss: 6.2223
Epoch 23/30
3/3 [==============================] - 6s 2s/step - loss: 0.0244 - val_loss: 6.0118
Epoch 24/30
3/3 [==============================] - 6s 2s/step - loss: 0.0211 - val_loss: 5.8287
Epoch 25/30
3/3 [==============================] - 6s 2s/step - loss: 0.0110 - val_loss: 5.6908
Epoch 26/30
3/3 [==============================] - 6s 2s/step - loss: 0.0039 - val_loss: 5.5728
Epoch 27/30
3/3 [==============================] - 6s 2s/step - loss: 0.0027 - val_loss: 5.4601
Epoch 28/30
3/3 [==============================] - 5s 2s/step - loss: 0.0050 - val_loss: 5.3541
Epoch 29/30
3/3 [==============================] - 3s 1s/step - loss: 0.0098 - val_loss: 5.2396
Epoch 30/30
3/3 [==============================] - 562s 1s/step - loss: 0.0174 - val_loss: 5.0909
```

Figure 7. Training the model using training set

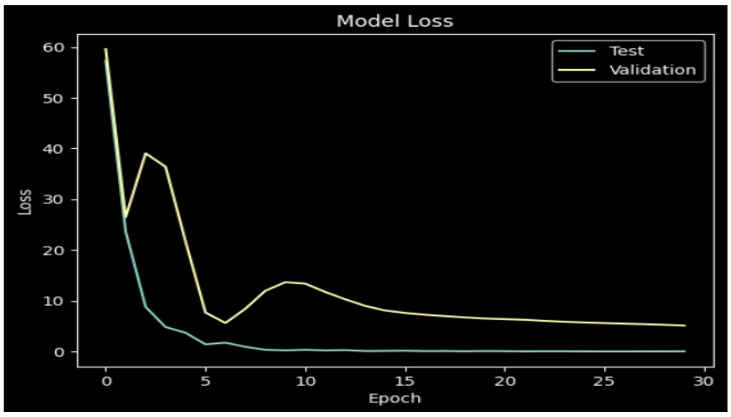
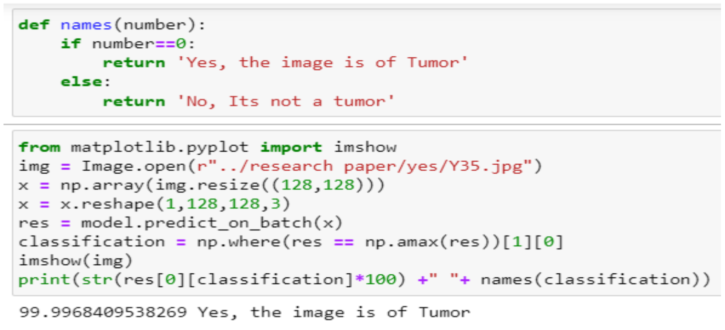


Figure 8. Graph showing training and validation loss

Figure 8. shows that the losses are decreasing as the epoch progresses, which means that the model is improving and becoming more accurate in its prediction.

F. Predicted using the model: Figure 9 shows the accuracy to which the model predicts if the image is of a tumour or not.



```
def names(number):
    if number==0:
        return 'Yes, the image is of Tumor'
    else:
        return 'No, Its not a tumor'
```

```
from matplotlib.pyplot import imshow
img = Image.open(r"../research paper/yes/Y35.jpg")
x = np.array(img.resize((128,128)))
x = x.reshape(1,128,128,3)
res = model.predict_on_batch(x)
classification = np.where(res == np.amax(res))[1][0]
imshow(img)
print(str(res[0][classification]*100) +" "+ names(classification))

 99.9968409538269 Yes, the image is of Tumor
```

Figure 9. Predicting the output using the model

Table 2. Performance comparison

Methodology	Accuracy
Proposed CNN model	99%
Sajid et. al. [12]	91%
Ozyurt et. al. [13]	95.62%

4. Result and Analysis

The model takes an input image, which is resized to a fixed size of (128, 128) pixels. This is a common pre-processing step in computer vision tasks to ensure all input images have the same dimensions and thus can be fed into the model consistently. After the input image is pre-processed, the model analyses it and generates a prediction. The output is the accuracy percentage, which indicates the confidence level of the model in classifying the image as either a tumour or a normal human brain. The accuracy percentage is a metric that reflects the proportion of correctly classified images out of all the images in the test set. The goal of this machine learning model is achieved as it accurately classifies new images as either tumour or normal human brain, based on the patterns and features it has learned from the training data. In this proposed model we got 99% accuracy that is higher than the results obtained by Sajid et. al. [12] and Ozyurt et. al. [13] as mentioned in Table 2.

5. Conclusion

In conclusion, this project aimed to develop a machine-learning model that could accurately detect brain tumours using MRI data. The project utilized a dataset of 2000 brain MRI images, which was sufficient to evaluate the model's effectiveness. A convolutional neural network (CNN) was chosen as the basis for the model, which allowed for the reduction and resizing of the images without losing any crucial information that could be used for prediction. As the number of epochs increased, the loss decreased progressively. The model achieved an accuracy of 99% when tested on the test set, indicating its high effectiveness in predicting the correct result. Overall, this project demonstrates the potential of using machine learning techniques to improve the accuracy and efficiency of medical diagnosis.

6. Future Scope

Currently, the input images are resized to a fixed size of (128, 128) pixels. Further research could investigate the impact of using higher-resolution images on the model's performance. This could involve experimenting with larger image sizes, such as (256, 256) pixels or even higher, to determine if more detailed information contributes to improved accuracy. Augmentation techniques can be applied to the training data to increase its diversity and variability. By introducing transformations like rotation, scaling, and flipping, the model can be exposed to a wider range of image variations. This augmentation process may enhance the model's ability to generalize and improve its accuracy on previously unseen data. The future scope could explore the use of ensemble models, such as combining the outputs of multiple classifiers, to enhance the overall accuracy and robustness of the system. Different models or variations of the current model architecture can be trained on the same dataset, and their predictions can be aggregated to achieve a more reliable and accurate classification.

7. References

1. Adate A, Arya D, Shaha A, Tripathy BK (2020) Impact of Deep Neural Learning on Artificial Intelligence Research. pp 69–84

2. Tripathy B, Mohanty R, Parida S (2022) Brain Tumour Detection Using Convolutional Neural Network-XGBoost

3. Maheshwari K, Shaha A, Arya D, Rajasekaran R, Tripathy BK (2020) 2 Convolutional Neural Networks: A Bottom-Up Approach. In: 2 Convolutional Neural Networks: A Bottom-Up Approach. De Gruyter, pp 21–50

4. Singhania U, Tripathy BK (2021) Text-Based Image Retrieval Using Deep Learning. In: Encyclopedia of Information Science and Technology, Fifth Edition. IGI Global, pp 87–97

5. Gupta P, Bhachawat S, Dhyani K, Tripathy BK (2022) A Study of Gene Characteristics and Their Applications Using Deep Learning. In: Roy SS, Taguchi Y-H (eds) Handbook of Machine Learning Applications for Genomics. Springer Nature, Singapore, pp 43–64

6. Bhardwaj P, Guhan T, Tripathy BK (2022) Computational Biology in the Lens of CNN. In: Roy SS, Taguchi Y-H (eds) Handbook of Machine Learning Applications for Genomics. Springer Nature, Singapore, pp 65–85

7. Tripathy BK, Parikh S, Ajay P, Magapu C (2022) 10 - Brain MRI segmentation techniques based on CNN and its variants. In: Chaki J (ed) Brain Tumor MRI Image Segmentation Using Deep Learning Techniques. Academic Press, pp 161–183

8. Prabhavathy P, Tripathy BK, Venkatesan M (2022) Analysis of Diabetic Retinopathy Detection Techniques Using CNN Models. In: Mishra S, Tripathy HK, Mallick P, Shaalan K (eds) Augmented Intelligence in Healthcare: A Pragmatic and Integrated Analysis. Springer Nature, Singapore, pp 87–102

9. Sihare P, Ullah Khan A, Bardhan P, Tripathy BK (2022) COVID-19 Detection Using Deep Learning: A Comparative Study of

Segmentation Algorithms. In: Das AK, Nayak J, Naik B, Vimal S, Pelusi D (eds) Computational Intelligence in Pattern Recognition. Springer Nature, Singapore, pp 1–10

10. Sajid S, Hussain S, Sarwar A (2019) Brain Tumor Detection and Segmentation in MR Images Using Deep Learning. Arab J Sci Eng 44:9249–9261. https://doi.org/10.1007/s13369-019-03967-8

11. Özyurt F, Sert E, Avci E, Dogantekin E (2019) Brain tumor detection based on Convolutional Neural Network with neutrosophic expert maximum fuzzy sure entropy. Measurement 147:106830. https://doi.org/10.1016/j.measurement.2019.07.058

12. Mohsen H, El-Dahshan E-SA, El-Horbaty E-SM, Salem A-BM (2018) Classification using deep learning neural networks for brain tumors. Future Comput Inform J 3:68–71. https://doi.org/10.1016/j.fcij.2017.12.001

13. Bauer S, May C, Dionysiou D, Stamatakos G, Büchler P, Reyes M (2012) Multiscale modeling for image analysis of brain tumor studies. IEEE Trans Biomed Eng 59:25–29. https://doi.org/10.1109/TBME.2011.2163406

14. Islam A, Reza SMS, Iftekharuddin KM (2013) Multifractal Texture Estimation for Detection and Segmentation of Brain Tumors. IEEE Trans Biomed Eng 60:3204–3215. https://doi.org/10.1109/TBME.2013.2271383

15. Seetha J, Raja SS (2018) Brain Tumor Classification Using Convolutional Neural Networks. Biomed Pharmacol J 11:1457–1461

16. Brindha PG, Kavinraj M, Manivasakam P, Prasanth P (2021) Brain tumor detection from MRI images using deep learning techniques. IOP Conf Ser Mater Sci Eng 1055:012115. https://doi.org/10.1088/1757-899X/1055/1/012115