International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF (2022): 7.942

Convolutional Neural Networks - Based Intelligent Model for Brain Tumor Detection on MRI Images

James Oladimeji[1], Olushola Ogunniyi[2]

[1]University of East London
Email: *ojames314[at]gmail.com*

[2]Edinburg Napier University
Email: *olushola.ogunniyi[at]napier.ac.uk*

Abstract: ***The processing of medical images is critical in assisting people in determining whether or not an MRI image contains a tumor. To assist clinicians, an intelligent system for detecting brain tumors is required. The study's goal is to use deep learning to interpret MRI images and determine whether or not they contain tumors. The early detection of tumors is critical for a speedy and effective cure. Medical image processing utilizing a convolutional neural network (CNN) is providing outstanding results in this regard. The proposed model has a precision of 93.7 percent and a loss of 6 percent, making it in line with previous methods for detecting brain cancers. In the field of medicine, the proposed system will give clinical help.***

Keywords: medical image processing, brain tumor detection, MRI interpretation, deep learning, convolutional neural network CNN

1. Introduction

The brain is an important organ that controls the central nervous system. By linking the bone marrow, the human brain completes the central nervous system. Controlling the actions of the human body is the job of the brain. It takes data from a variety of senses and, after making decisions, transmits commands to the body (Bozdağ-Pehlivan, 2017). The brain is the major portion of the human body's administration division, and itis responsible for all of the body's actions through the use of neurons. The two most common types of major brain tumors are malignant and benign (Thomas, 1984). Adults and children alike are at risk of developing a brain tumor. A brain tumor develops when brain tissues develop abnormally in comparison to healthy cells, resulting in a mass of cells that eventually transforms into tumors (Thomas, 1984).

From 2016 to 2018, there were approximately 12000 cases of brain tumors, according to Cancer Research UK (*Brain, Other CNS and Intracranial Tumours Statistics*, 2015). The brain tumor is still one of the most serious cancers. The early and accurate diagnosis of a brain tumor is critical in this form of malignancy (Komisarow & Sampson, 2017). Deep learning approaches have been found to be more effective in detecting brain tumors (Khan et al., 2022).

According to Balaji & Lavanya, (2019), Deep Learning is an area of machine learning dealing with artificial neural networks, which are algorithms inspired by the structure and function of the brain. When there is a dearth of domain awareness for feature introspection, deep learning techniques outperform others since feature engineering is less of a concern. When it comes to complicated issues like image classification, natural language processing, and speech recognition, deep learning really shines. In contrast to machine learning, which produces the same results regardless of the data presented, the more data provided to the deep learning model, the better the outcomes. Deep learning's use cases have grown in the areas of customer service, health care, self- driving cars, Google translation, and other areas over time. There are several different forms of deep neural networks, including CNN. CNNs are popular because they can be used to recognize images in a variety of ways. A convolutional Neural Network (ConvNet/CNN) is a deep learning system that can take an input image, give importance (learnable weights and biases) to distinct characteristics or objects in the image, and distinguish one from the other (Teuwen & Moriakov, 2020). Despite the fact that convolutional neural networks (CNN) have the benefit of learning representative complex features for both healthy brain tissues and malignant tissues directly from multi-modal MRI images, Chattopadhyay & Maitra (2022) opted to improve its accuracy by incorporating SVM into CNN. Template matching, convolutional filters, pooling layers, padding layers, and flatten operations are the components of CNN. Template matching is a digital image processing approach for locating small portions of an image that match the template image (Teuwen & Moriakov, 2020). CNN employs convolution filters, which are regarded as the most important feature of the network because they alter an image by changing the shades and colors of the pixels, as well as increasing the brightness and contrast of the image and filtering out unnecessary pixels, noise, and particles, allowing the image to focus on the most important parts of the image. The pooling layers are used to minimize the feature maps' dimension. As a result, the number of parameters to learn and the quantity of computation done in the network are reduced. The quantity of pixels added to an image when it is processed by the kernel or filter is referred to as the padding layers aspect of CNN. The flatten process converts the data into a one-dimensional array, which is then passed on to the next layer.

Brain tumors are difficult to understand. There are numerous variations in the sizes and locations of brain tumors (s). This makes a thorough knowledge of the tumor's nature extremely challenging. For MRI analysis, a skilled Neurosurgeon is also required. The dearth of skilled doctors and information about malignancies makes generating results from MRI's extremely difficult and time-consuming in developing countries. As a result, a cloud-based

automated solution can address this issue. In this study, we will use a convolutional neural network to determine whether or not a person has a tumor. A CNN deep learning model was created to detect brain malignancies on the MRI images, and the classification procedure was carried out utilizing MRI images.

2. Methodology

The processes for developing this model are the same as for any other machine learning project: Data is collected, images are pre- processed, the model is built, and the result is validated.

2.1 Architecture

Memory use, parameter counts, and computational complexity are all common metrics used to evaluate CNN designs. AlexNet, VGG, GoogLeNet, Residual Networks, DenseNets, MobileNets and LeNet model are examples of typical CNN architectures. LeNet model architecture is employed in this research. LeNet is a CNN model that was created with the goal of providing the highest level of accuracy. The first LeNet was created in the 1980s. In the 1998 research publication Gradient-Based Learning Applied to Document Recognition, Yann LeCun and others proposed Lenet-5 architecture as one of the first pre-trained models. This architecture was utilized to distinguish between characters that were handwritten and those that were machine- printed (Zhu et al., 2021). This type's popularity stemmed mostly from its simple and straightforward architecture. A multi- layer convolutional neural network is used to categorize image. It operates by passing 32by32 input through a single channel. The image is black and white due to the single channel. It was initially created using a 5by5 matrix, but as time went on, it was improved to a 3 by 3 matrix to build a feature map. As seen in fig 1, the operation proceeds from resampling to max-pooling over the convolutional layers, flattening images, and finally producing 10 classes of output

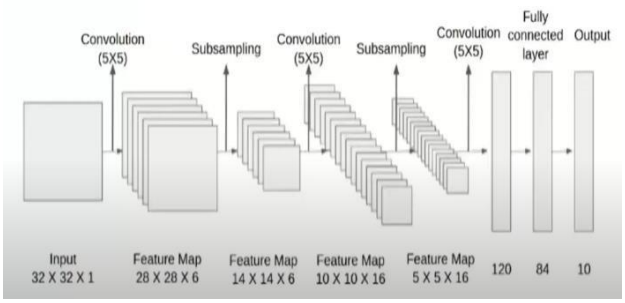

Figure 1: LeNet Model Architecture
Source: *https://www.edureka.co/*

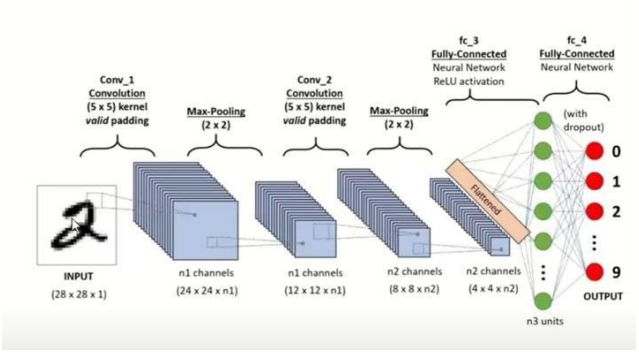

Figure 2: Connected architecture of CNN in the study.
Source: *https://www.edureka.co/*

The end-to-end implementation of CNN in this study is depicted in Fig 2. The convolutional layers transmit the MRI images forward to the mass pooling layers. After that, the images are removed and flattened into a 1- dimensional array. The flattened characteristics will then flow through the dense layers, resulting in equal input neuron output. The cross-entropy loss of classification and weighted classification asks with mutually exclusive classes are computed by a classification layer.

2.2 Dataset

The dataset employ in this research is MRI images. The MRI scans that were used in this study were provided by https://www.kaggle.com/.Thedatasetis divided into two folders: yes and no. Yes, there are 1500 tumorous brain MRI images in the yes folder. There are 1500 non-tumorous brain MRI images in the no folder. Data collection takes up around 40% of every project because once the necessary data is gathered, the rest of the work looks to fall into place. To implement the model, this project will employ karas, tensor flow, jupyter notebook, matplotlib, pandas, numpy, and other tools. To code the implementation of CNN in this study, the Colab code editor by https://colab.research.google.com/ was utilized as an Integration Development Environment (IDE). This was taken into account because training a model on a large dataset takes longer, and the Google Colab editor contains both GPU and CPU accelerator technology. GPUs are useful when a task takes a long time to finish.

3. Implementation

The implementation is simple and will take place in four steps. The data must first be uploaded. 2. Pre-process the data before using it in the model. 3. Train the model, and then put it to the test.

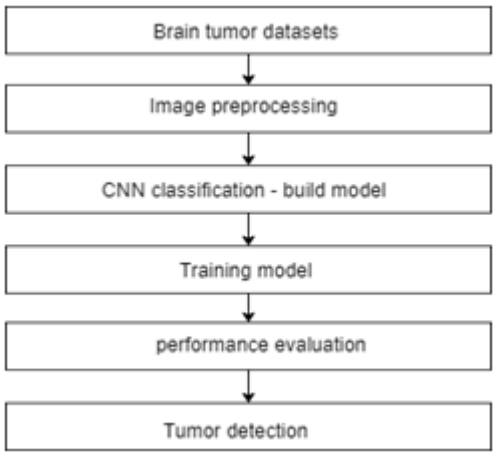

Figure 3: Flowchart of the study

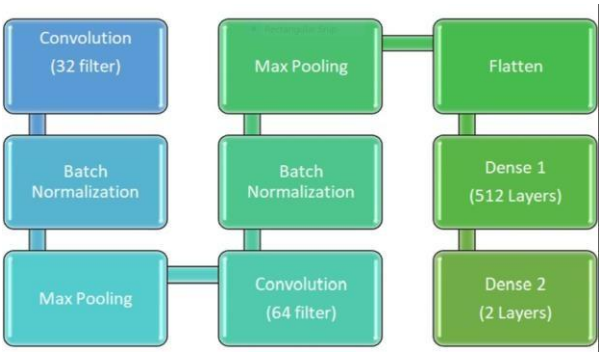

Figure 4: CNN flow for brain tumor detection

2.3 Algorithm

The modern conception of a thinking machine, whether it is the human mind or a modern computer, revolves around the concept of algorithm. A well-defined mathematical formula for solving a well- defined problem is known as an algorithm (Martignon, 2001). It is provided as a finite collection of instructions or processes that can be applied to an infinite number of scenarios. The set of instruction for accomplishing the implementation of this model is as follows:

1) Upload the MRI images data from https://www.kaggle.com/repository into the code editor using wget and unzip commands. Images with tumors and images without tumors are included in the data. The tumor images are filed in a separate folder called brain tumor, while the healthy images are filed in a separate folder called healthy.
2) We'll import the common libraries we'll need for our project, such as numpy for calculations, matplotlib for graph plotting, os for any operating- system-related activities, math library, shutil, and globe libraries.
3) Count how many images there are in each class. That is, images with a brain tumor will be represented by 0 and healthy images by 1.
4) The data will then be divided into three parts: 70 percent for training, 15 percent for validation, and the remaining 15 percent for testing.
5) It's time to build our CNN model now that we've successfully segregated our data into train, test, and validation folders. We'll start by importing the keras framework's essential libraries, such as keras.layers, keras.model, keras.preprocessing, and Keras itself.
6) Now that our model has been developed, we'll perform early stopping (ES) and model checking point (MC) on it before training to improve its validation accuracy.
7) It's time to train the model. The phase in which we strive to match the best combination of weights and bias to the algorithm in order to minimize a loss function over the prediction range.
8) Test the model accuracy. The keras library's load model API can be used to test the model's accuracy. One method of evaluating a model's performance is to divide the number of categories a model successfully predicts by the total number of predictions made.
9) Lastly, because we're doing study on people, we're dealing with their lives. Because our model is predicting whether or not a person has cancer, it is necessary to improve model accuracy through transfer learning or hyperparameter tuning (by using Keras tuner).

2.4 Codebase

A codebase is a collection of all the source code for a particular software program or application. C++, Java, Python, C#, Rust, and other programming languages can be used to create source code. A programming language is a set of grammatical rules and a vocabulary for instructing a computer or computing device to carryout a certain task. The code for the model is written in the Python programming language in this study. Guido van Rossum created the Python programming language, which is platform-independent (i.eit can run in any operating system). Python is an object-oriented, high-level programming language that prioritizes code readability by the use of substantial identification. Python is a programming language that was created in 1994 and has grown in prominence in thefield of data science (Sturm et al., 2017). As a result, when it comes to machine learning projects, is always the best option. Below is the study's python source code.

```
#ignorethewarnings import warnings
warnings.filterwarnings('ignore') # get the data from kaggle.com
!wgethttps://www.kaggle.com/datasets/navoneel/brain-mri-images-for-brain-tumor-detection/download# unzip the data
!unzip/content/download
#let'simportthecommonlibraries import numpyas np
importmatplotlib.pyplotaspltimport os
import math importshutilimport glob
#countthenumberofimagesintherespectiveclass es.0 - Brain tumorand 1 - healthy
#createdirectory
ROOT_DIR="/content/sample_data/BrainTumor Dataset"
number_of_images = {}#iteratetherootdirectory
for dirin os.listdir(ROOT_DIR): number_of_images[dir] = os.listdir(os.path.join(RO OT_DIR, dir))
#countnumberofimagesineachdirectoryandputi t in a dictionary
number_of_images_count = {}
number_of_images_count['BrainTumor']=len(numb er_of_images.get('BrainTumor'))
number_of_images_count['Healthy']=len(number_of_images .get('Healthy'))
#splitdatasuchthat70%fortraining, 15%forvalid ation and the rest 15% for testing.
#let'sstartbycreatingafolderfortraining, validatio n and
```

```
testing
defdataFolder(path, split):
ifnotos.path.exists("./"+path): os.mkdir("./"+path)
fordirinos.listdir(ROOT_DIR):
if dir != '.ipynb_checkpoints': os.makedirs("./"+path+"/"+dir)
forimginnp.random.choice(a=os.listdir(os.pa
th.join(ROOT_DIR, dir)),
size=(math.floor(split*number_of_images_count[dir])-5),
replace=False):
source=os.path.join(ROOT_DIR, dir, img) #original
destination=os.path.join("./"+path, dir) shutil.copy(source,
destination) os.remove(source)
else:
print(f"{path}folderalreadyexist") #create train folder
dataFolder('train', 0.7) #create test folder dataFolder('test',
0.15) #createvalidationfolder dataFolder('val', 0.15) #build
model
fromkeras.layersimport Conv2D, MaxPool2D, Dro pout,
Flatten, Dense, BatchNormalization, GlobalAvgPool2D
fromkeras.modelsimportSequential
fromkeras.preprocessing.imageimportload_img,
img_to_array, ImageDataGenerator
import keras#CNNModel
model= Sequential() #firstconvolutionallayer
model.add(Conv2D(filters=16, kernel_size=(3, 3), ac
tivation='relu', input_shape=(224, 224, 3)))
#secondconvolutionallayer
model.add(Conv2D(filters=36, kernel_size=(3, 3), ac
tivation='relu', ))
#add pooling layer model.add(MaxPool2D(pool_size=(2,
2))) #third convolutional layer
model.add(Conv2D(filters=64, kernel_size=(3, 3), ac
tivation='relu', ))
#add pooling layer model.add(MaxPool2D(pool_size=(2,
2))) #fourth convolutional layer
model.add(Conv2D(filters=128, kernel_size=(3, 3), a
ctivation='relu', ))
#add pooling layer model.add(MaxPool2D(pool_size=(2,
2))) #add dropout layer model.add(Dropout(rate=0.25))
#flatten the image model.add(Flatten()) #add dense layer
model.add(Dense(units=64, activation='relu'))
#addanotherdropoutlayertopreventmodelfromoverfitting
model.add(Dropout(rate=0.25)) model.add(Dense(units=1,
activation='sigmoid')) #model.summary()
#compilemodel
model.compile(optimizer='adam', loss=keras.losses.
binary_crossentropy, metrics='accuracy')
#preparingourdatausingimagedatageneratorand data
augumentation
defpreprocessingImage(path):
"""
Input:path
Output:Preprocessedimages """
image_data=ImageDataGenerator(zoom_range=0. 2,
shear_range=0.2, rescale=1/255, horizontal_flip= True)
image=image_data.flow_from_directory(directory
=path, target_size=(224, 224), batch_size=32, class_
mode='binary')
returnimage
#preparingourdatausingimagedatageneratorand data
augumentation
defpreprocessingImageForValidagtionAndTes(path)
:
"""
Input:path
Output:Preprocessedimages """
#fortestandvalidationdata, iwantthedatatobea dded naturally
as possible
# in other to have more generalized model image_data =
ImageDataGenerator(rescale=1/255)
image=image_data.flow_from_directory(directory
=path, target_size=(224, 224), batch_size=32, class_
mode='binary')
returnimage
#processtrain, testandvalidationimages #train
path="/content/train"
train_data=preprocessingImage(path) #test
test_path="/content/test"
test_data=preprocessingImageForValidagtionAndTes(test_p
ath)
#validation data val_path="/content/val"
val_data=preprocessingImageForValidagtionAndTes(val_pa
th)
#Earlystoppingandmodelcheckpoint
fromkeras.callbacksimportEarlyStopping, ModelCheckpoint
#earlystopping
es=EarlyStopping(monitor="val_accurracy",
min_delta=0.01, patience=3, verbose=1, mode='auto')
#model checkpoint
mc=ModelCheckpoint(monitor="val_accuracy", fil
epath="./bestmodel.h5", verbose=1, save_best_only
=True, mode="auto") cd = [es, mc]
#Modeltraining
h5=model.fit_generator(generator=train_data, steps
_per_epoch=8, epochs=10, verbose=1, validation_da
ta=val_data, validation_steps=6, callbacks=cd)
#Modelgraphicalinterpretation N = epochs
plt.style.use("ggplot")plt.figure()
plt.plot(np.arrange(0, N), history.history["loss"], label=
"train_loss") plt.plot(np.arrange(0, N),
history.history["val_loss"], label= "val_loss")
plt.plot(np.arrange(0, N), history.history["accuracy"],
label="train_acc") plt.plot(np.arrange(0, N),
history.history["val_accuracy"], label= "val_acc")
plt.title("TrainingLossandAccuracyonBrainDataset")
plt.xlabel("Epoch") plt.ylabel("Loss/Accuracy")
plt.legend(loc= "lower left") plt.savefig("plot.jpg")
```

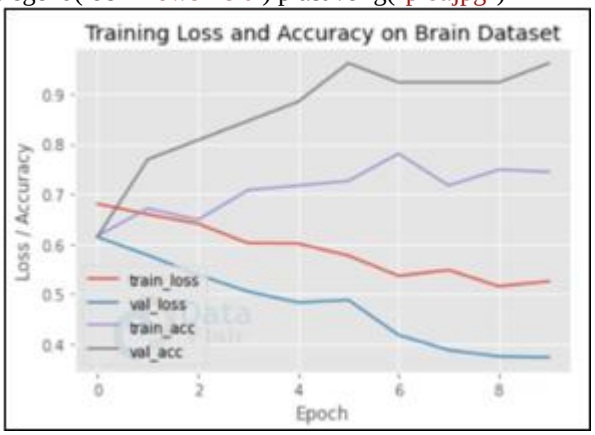

Figure 5: Train loss vs val_loss vs train_acc vs val_acc

```
#Modelaccuracy
fromkeras.modelsimportload_model
model=load_model("/content/bestmodel.h5") acc =
model.evaluate(test_data)[1]
```

```
print(f"Theaccuracyofourmodelis{acc *100}
%")
Theaccuracyofourmodelis85.18518805503845% # Let's how
well our model can predict
#0-braintumor, 1-healthy
fromkeras_preprocessing.image.utilsimportload_i mg,
img_to_array
path="/content/sample_data/BrainTumorDataset/
BrainTumor/Y165.JPG"
img=load_img(path, target_size=(224, 224)) input_arr =
img_to_array(img)/255 plt.imshow(input_arr)
plt.show()
input_arr=np.expand_dims(input_arr, axis=0)
predict_x=model.predict(input_arr)
predict=np.argmax(predict_x, axis=-1)[0]
ifpredict==0:
print("TheMRIishavingtumor") else:
print("TheMRIisNOThavingtumor")
```

4. Results

In this study, we used CNN deep learning to do Image Classification on MRI images, which yielded an accuracy of roughly 94%. The image path is passed to the prediction function. The function that determines whether or not an MRI image contains a tumor. Because each row of the prediction array has two values for the respective labels, the numpy library's argmax function was employed. Images with tumor are given a score of 0 while images with no tumors are given a score of 1.

4.1 Training

A dataset used to train a machine learning algorithm is referred to as a training model. It consists of sample output data as well as the corresponding sets of input data that influence the outcome. In order to compare the processed output to the sample output, the training model is utilized to process the input data via the algorithm. The study data isn't large, and the training takes only5 minutes on the CPU and a few seconds on the GPU. 70% of the data provided was used for training purposes. To forecast with 99 percent accuracy, a model requires a large amount of data for training. Figure 4 depicts the training graphical representation. Red and blue colors represent training loss and training accuracy, respectively. The training loss is approximately 3 percent, and the accuracy is approximately 94 percent. It shows that training accuracy perform way better.

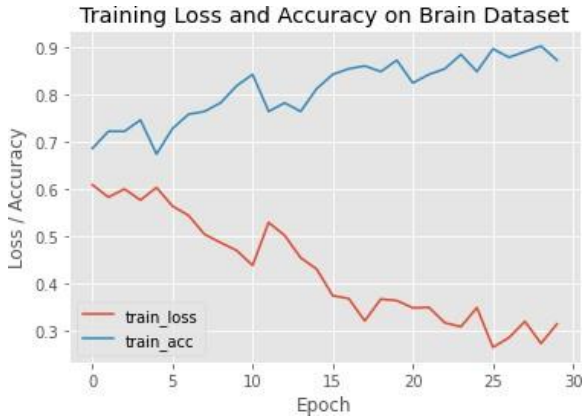

Figure 6: Training loss vs training accuracy

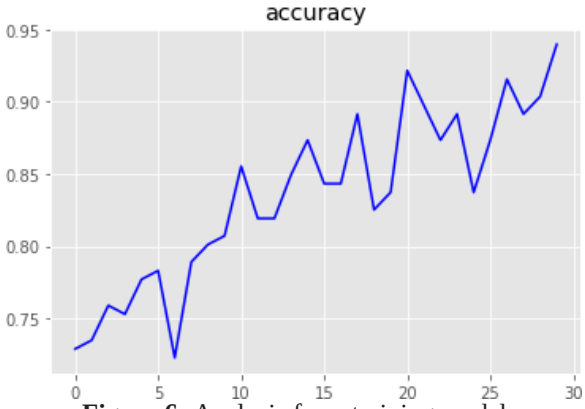

Figure 6: Analysis from training model

4.2 Testing.

Model testing is the process of evaluating the performance of a fully trained model on a testing set. The testing set, which is independent from the training and validation sets but follows the same probability distribution as the training set, consists of15% of the testing samples. Testing model estimates how well a model is trained. The model has a 94 percent accuracy rate, which indicates that it was well-trained.

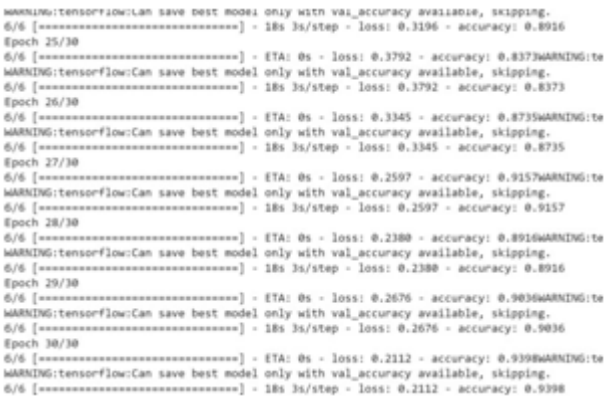

Figure 7: Model accuracy

The testing dataset determines the model accuracy. Humans write the logic in traditional software systems, which interact with data to produce the intended behavior. The software tests guarantee that the written logic corresponds to the expected behavior. In machine learning systems, on the other hand, humans contribute desired behavior as examples during training, and the model optimization process generates the system's logic. Model testing is doing explicit checks on the expected behaviour of our model.

4.3 Performance

According to Breiman (2001), a model's performance might be based on the evaluation of goodness-of-fit (GoF) or predictive accuracy (which we will term goodness-of-prediction, GoP). In general, GoF is used to create explanatory models, while GoP is utilized to create predictive models. Typically, GoF is concerned with the question of how well the model's predictions explain (fit) dependent-variable values of the data used in the model's development. GoP, on the other hand, is concerned with the question of how well the model predicts the dependent

variable's value for a new observation. In closing, model performance is how accurate our model can predict. From the code snippets below, it shows that our model performs accordingly.

```
#Let'sseehowwellourmodelcan predict
fromkeras_preprocessing.image.utilsimportload_i mg, img_to_array
path="/content/test/Healthy/27no.jpg"
img=load_img(path, target_size=(224, 224)) input_arr = img_to_array(img)/255 plt.imshow(input_arr)
plt.show()
input_arr=np.expand_dims(input_arr, axis=0)
predict_x=model.predict(input_arr)
predict=np.argmax(predict_x, axis=-1)[0]
ifpredict==0:
print("TheMRIishavingtumor") else:
print("TheMRIisNOThavingtumor")
```

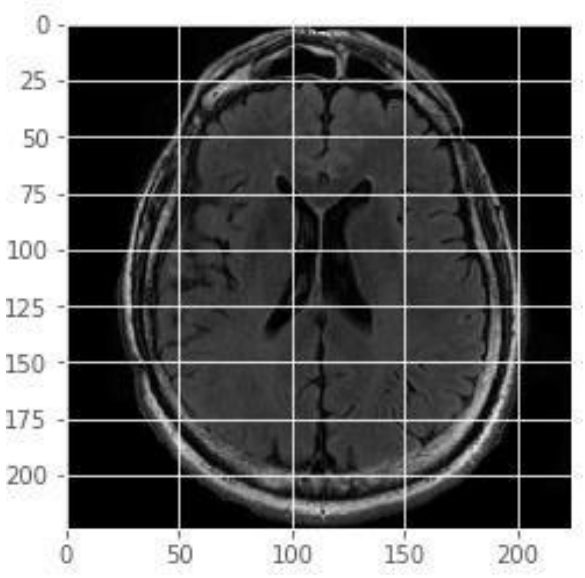

Figure 8: The MRI is NOT having tumor

5. Critical Analysis of this study

The author appears to have a good understanding of image classification and deep learning in general. Convolutional Neural Networks are part of the Deep Learning branch of Machine Learning (CNN). The human brain analyzes data in the same way that Deep Learning systems do. Around 86 billion neurons make up the human brain. Extracting characteristics from images in order to find patterns in a dataset is known as image classification. To detect brain tumors on MRI scans, the author creates an intelligent model that leverages CNN for image classification. The capacity of the CNN model to determine whether or not a person has brain cancer is the key message from this study. In this type of cancer, an early and precise detection of a brain tumor is crucial.

The author creates a CNN model with a 95% accuracy rate that will aid clinicians in detecting brain tumors on MRI images. My overall assessment of this work is that the author builds this model using a step-by-step deep learning technique, although the model may not be 100 percent accurate in production. Because we're dealing with human lives, a model that provides 101 percent validation accuracy will suffice in production, while the model can be enhanced further with a transfer learning method. Transfer learning is a machine learning technique in which we reuse a previously trained model as the basis for a new model on a different task. In a nutshell, transfer learning is a process in which a model is trained on one job and then repurposed on a second, similar work as an optimization that allows for faster modeling progress on the second task.

5.1 Decision

The overall goal of this study is to demonstrate the usefulness of deep learning in health fields by developing an intelligent model that uses CNN to determine whether or not a patient has a brain tumor. The research demonstrates that, in addition to standard methods of diagnosing brain tumors, there is another option that appears to be less expensive and more reproducible. Finally, the study steps through the brain identification model, which has a 95 percent accuracy rate and is based on deep learning, from data collecting to image preprocessing, image classification, and even displaying a comprehensible code to follow for reproductive purposes.

5.2 Justification

The followings are my justification for this study:

5.2.1 Information accuracy. The author's material appears to be fairly accurate, as it is backed up by current citations. Furthermore, the author went out of his way to diverge into health-related subjects, which is commendable, and offered a decent background on brain tumors. The information provided about image classification and deep learning provide is well-founded. The author appears to be thorough in his research; however the information lacks practical backing because the author did not provide any related practical work to back up his statements.

5.2.2 Definition of key terms (or lack of): The author use well-chosen phrases from the study's field. Especially in the field of medicine, the terminology are thoroughly defined and discussed. By choosing the appropriate terminology, the author accomplished a good job.

5.2.3 Hidden assumption: This is an implicit assumption that supports a claim orhypothesis without being stated directly. The author made a few hidden assumptions, such as that Python is the greatest programming language for doing machine learning, that deep learning is the cheapest approach to detect brain tumors, and that CNN is the best way to detect whether or not a person has a tumor. There are many different types of brain tumors, and the human brain includes approximately 86 billion neurons.

5.2.4 Clarity of language: This is the case of clearly defined purpose; logical arrangement, well-constructed sentences, and exact word choice are all characteristics of properly written prose. The sentences linked to the topic of study are carefully constructed by the author. In addition, the authors plan the research session in a standardized manner and adhere to the general guidelines for writing a research report.

5.2.5 Logical and organization: The basic goal of writing is to convey information and ideas. In essence, the author arranges the written elements in such a way that they make sense. The functional order of sentences in paragraphs aids readers in following the progression of the thoughts in this example. Furthermore, the authors produce written works that are up to academic norms. The organizing of ideas from general to specific, which improves the overall quality of papers, is referred to as logical order, and this work fulfills that logical criterion to some level.

5.3 Observation

While the author did an excellent job with this research, there are portions of it that I agree with and others that I disagree with. I agree that deep learning is more powerful than previous methods for detecting brain cancers, but you can't discount the intensive work that goes into traditional methods for detecting brain tumors. I disagree that putting this model into production is a certain means of detecting brain cancer; I feel it should be tested under rigorous conditions and with a huge dataset to be certain. The author, in my opinion, got the progressive development of the model correct. The author got the portion about brain cancer killing people every day right, as well as the background study of the brain tumor, but I believe there is still a more sophisticated technique to achieve the same or better in terms of training and testing the model. Finally, I will endorse the research as a reliable source for deep learning-based brain tumor identification.

5.4 Explanations

Finally, the goal of this research is to use CNN to preprocess MRI images for brain tumor detection. To achieve this goal, the author devised a model. Although the model building was extensive and instructive, the quality of the model would have been considerably greater if the author had used transfer learning instead of starting from scratch. The ultimate result of this work is demonstrating the power of deep learning in the health industry, particularly the multi-class image classification part utilizing CNN.

6. Conclusions

In both adults and children, a brain tumor is considered a fatal cancer. Glioma, meningioma, and pituitary tumors are the most frequent primary tumors in adults (Khan et al., 2022). Several strategies for detecting a brain tumor have been proposed in the literature in order to increase the treatment options and patient endurance. In this paper, CNN is used to propose a Deep Learning- based Brain Tumor Classifier. The model divided the data into two categories: timorous (has tumor) and non-tumorigenic (no tumor). The suggested model outperformed existing brain tumor identification and segmentation approaches with a 94 percent accuracy and a 6 percent loss. In the medical field, the suggested system will give clinical support.

6.1 Summary

This study employs a deep learning model for intelligent identification of brain cancers using a Convolutional Neural Network. The data was obtained from Kaggle and is divided into two categories: no tumor and tumor types (*Brain MRI Images for Brain Tumor Detection*, n.d.). The model employed a total of 3000 images, with 1500 images for no tumor and 1500 images for the has-tumor type. Training, validation, and testing are the three steps of the model. The training phase takes up 70% of the images, while the validation and testing phases take up 15% each.

Each epoch includes training steps for the training set and validation steps for the validation set, for a total of 30 epochs (complete iterations). For each epoch, the batch size is set to 6. A few improvements were made, such as early stopping and model checkpoints, so that the model would cease training when it reached the highest level of accuracy. The model achieved 93.7 percent accuracy on the test set after training. The model will then be evaluated using the predict () method. The model's predictions will be an array, with each value representing the chance that the image belongs to that category. As a result, we choose the highest of all such probability and assign that picture input the projected label.

Finally, we look at how machine learning maybe used to classify brain tumors. A binary classifier was created to detect brain tumors using MRI scan images. The classifier was created using deep learning and had a 93.7 percent accuracy rate, as well as a visual representation of the model's overall performance (see fig 4).

7. Future work

Despite the model's high accuracy, we are still concerned about deploying it into production (in a real-time environment), for example, into a hospital application. The model's accuracy is 93.7 percent, which means there's a 7% chance it will forecast something incorrectly. In order to improve accuracy, we suggest that future research focus on transfer learning or hyper parameter modeling utilizing a keras tuner. Also, In the future, we will not only use the model to determine whether a person has a tumor or not, but we will also extends the model algorithm to determine the type of tumor. It could be a meningioma or a schwannoma, for example.

References

[1] Balaji, K., & Lavanya, K. (2019). Chapter 5 - Medical Image Analysis with Deep Neural Networks. In A. K. Sangaiah (Ed.), *Deep Learning and Parallel Computing Environment for Bioengineering Systems* (pp. 75-97). Academic Press. https://doi. org/10.1016/B978-0-12-816718-2.00012-9

[2] Bozdağ-Pehlivan, S. (2017). Chapter 17- Brain Tumors. InY. Gürsoy-Özdemir, S. Bozdağ-Pehlivan, & E. Sekerdag (Eds.), *Nanotechnology Methods for Neurological Diseases and Brain Tumors* (pp. 319-344). Academic Press. https://doi. org/10.1016/B978-0-12-803796-6.00017-4

[3] *Brain MRI Images for Brain Tumor Detection*. (n. d.). Retrieved 25 April 2022, from https://www. kaggle.com/navoneel/brain-mri-images-for-brain-tumor-

detection

[4] *Brain, other CNS and intracranial tumours statistics*. (2015, May 14). Cancer Research UK. https://www. cancerresearchuk.org/health-professional/cancer-statistics/statistics-by-cancer-type/brain-other-cns-and-intracranial-tumours

[5] Breiman, L. (2001). Random Forests. *Machine Learning*, *45* (1), 5-32. https://doi. org/10.1023/A: 1010933404 324

[6] Chattopadhyay, A., & Maitra, M. (2022). MRI-based brain tumour image detection using CNN based deep learning method. *Neuroscience Informatics*, *2* (4), 100060. https://doi. org/10.1016/j. neuri.2022.10 0060

[7] Khan, A. H., Abbas, S., Khan, M. A., Farooq, U., Khan, W. A., Siddiqui, S. Y., & Ahmad, A. (2022). Intelligent Model for Brain Tumor Identification Using Deep Learning. *Applied Computational Intelligence and Soft Computing*, *2022*, 1-10. https://doi. org/10.1155/2022/8104054

[8] Komisarow, J. M., & Sampson, J. H. (2017). Chapter 1-An Introduction to Immunotherapy in the Treatment of Brain Tumors. InJ. H. Sampson (Ed.), *Translational Immunotherapy of Brain Tumors (*pp. 1-10). Academic Press. https://doi. org/10.1016/B978-0-12-802420-1.00001-6

[9] Martignon, L. (2001). Algorithms. In N. J. Smelser & P. B. Baltes (Eds.), *International Encyclopedia of the Social & Behavioral Sciences (*pp. 382-385). Pergamon. https://doi. org/10.1016/B0-08-043076-7/00549-0

[10] Sturm, R., Pollard, C., & Craig, J. (2017). Chapter 11-Application Programming Interfaces and Connected Systems. In R. Sturm, C. Pollard, & J. Craig (Eds.), *Application Performance Management (APM) in the Digital Enterprise (*pp. 137-150). Morgan Kaufmann. https://doi. org/10.1016/B978-0-12-804018-8.00011-5

[11] Teuwen, J., & Moriakov, N. (2020). Chapter 20-Convolutional neural networks. In S. K. Zhou, D. Rueckert, & G. Fichtinger (Eds.), *Handbook of Medical Image Computing and Computer Assisted Intervention (*pp. 481-501). Academic Press. https://doi. org/10.1016/B978-0-12-816176-0.00025-9

[12] Thomas, D. G. T. (1984). Chapter56-Brain tumours. In M. J. G. Harrison (Ed.), *Contemporary Neurology (*pp. 511-530). Butterworth-Heinemann. https://doi. org/10.1016/B978-0-407-00308-8.50060-4

[13] Zhu, Y., Li, G., Wang, R., Tang, S., Su, H., & Cao, K. (2021). Intelligent fault diagnosis of hydraulic piston pump combining improved LeNet-5 and PSO hyperparameter optimization. *Applied Acoustics*, *183*, 108336. https://doi. org/10.1016/j. apacoust.202 1.108336