Journal of Open Aviation Science JOAS

ARTICLE (*Pre-print*)

Refined Brain Tumor Detection with Adapted Deep CNN Architecture

Damilola Eniola Olowu Msc.Engineering Data Science *

University of Houston, Houston, United States

*Corresponding author: Olowueniola@ymail.com

Abstract

With the advancement of artificial intelligence and machine learning in contemporary society, it is no surprise that this field has extended to the healthcare industry, with numerous healthcare professionals taking advantage of its possibilities. In the case of brain tumor detection and classification, Magnetic Resonance Imaging (MRI) has traditionally served as a non-invasive imaging technique that utilizes magnetic fields and radio waves to generate detailed images inside the body for brain tumor classification among other purposes. The varied types of brain tumors, such as gliomas, meningiomas, and metastatic tumors, have distinct appearances on MRI, aiding in their classification. Unfortunately, MRI scans typically take some time to complete, so naturally, an alternative method is being researched as a potential solution. With the help of deep learning methods, highly accurate classification results have been explored. This paper proposes a deep learning model to identify both the presence and absence of brain tumors using a publicly available dataset, Br35H, which includes 3450 MRI images. To build the model, a 19-layer Convolutional Neural Network (CNN) is applied. The experimental results indicate that the model achieves over 95 percent classification accuracy for the employed dataset. Additionally, the methodology draws inspiration from previous work, particularly the "Accurate Brain Tumor Detection Using Deep Convolutional Neural Network" paper, which employed a combination of a 23-layer CNN and Visual Geometry Group (VGG) 16 architecture on datasets containing 3064 and 884 MRI images, respectively. The proposed model, employed dataset, and all the source codes are publicly available at

https://github.com/DamilolaEniola/bio-6306-brain-tumor-detection.git

Keywords: Brain Tumor; Convolutional Neural Networks; Magnetic Reasoning Imaging

1. Introduction

A brain tumor can be described as a growth of cells directly in the brain or near it [1]. Brain tumors can affect different lobes of the brain, and a headache typically serves as the most common type of symptom. Brain cancer serves as the 10th leading cause of death for men and women with an estimated predicted death of 18,990 in 2023. Over 15 million people are diagnosed with cancer across the world yearly, and more than 80 percent will have to undergo surgery for treatment [2]. Gliomas and primary Central Nervous System (CNS) lymphomas serve as the most frequent brain tumors in adults. Meningioma, glioma, and pituitary are the most popular ones that have been detected as well as identified [3]. Brain tumors can either be malignant or benign. Malignant tumors tend to grow quickly across the surrounding brain tissues, whereas benign tumors tend to expand relatively slower. Percentage-wise, most tumors are benign with about 70 percent being identified as benign and 30 percent being identified as malignant.

When a brain tumor is clinically suspected, an MRI scan is the best method to define the characteristics of the mass, giving specific details such as location, size, degree of edema as well as contrast enhancement. An MRI scanner can be used to take snapshots of any part of the body in any imaging direction [4]. Additionally, it has advantages over Computed Tomography (CT) scanners because it provides a more enhanced soft tissue contrast and can differentiate better between soft tissue. In terms of its disadvantages, the time needed for an MRI is longer than needed for a CT. Other disadvantages also include reactions to the contrast agent, claustrophobia, as well as the effects of the magnetic field on metal devices implanted in the body. Consequently, over the last years, an interest in deep learning in the field of brain tumor detection has surged. It has been tested and extensively used in the application of brain tumor classification. This research suggests a novel MRI brain tumor detection technique based on artificial intelligence.

Many of the suggested machine learning models are focused on the binary identification of brain tumors. State-of-the-art methods being used today for such problems include Convolutional Neural Networks (CNN) [5]. These deep neural networks are well-suited for image analysis tasks. In the case of brain tumor classification detection, CNNs are typically trained on large datasets of MRI or CT scans to learn the features of the images, extract relevant features, and classify them into different categories of tumors as well as differentiate between tumors and non-tumors. Additionally, transfer learning, which occurs when a pre-trained neural network and its weights are fine-tuned for a specific task, like ResNet or VGG16, has been used in brain tumor detection to improve performance. Lastly, attention mechanisms allow deep learning models to focus on relevant segments of the image while being undistracted by irrelevant or noisy information. This results in the prioritization of regions of interest within the MRI scans in the context of brain tumor detection to improve accuracy and efficiency.

There are also machine learning models that are focused on multi-class brain tumor detection. For example, Sultan et al. suggested a CNN model with 16 layers [6]. The CNN model identified types of tumors and achieved over 95 percent accuracy. Hossain et al. used a type of clustering technique, referred to as the Fuzzy C-Means clustering technique, to extract the tumor area from the MRI images [7]. They proposed a new CNN-based model and reported 97.9 percent prediction accuracy, outperforming all other compared prior models. Lastly, the inspiration for this paper focused on applying a 23-layer CNN to the first dataset [8]. To address the second dataset, a combination of the VGG16 architecture along with the reflection of the proposed 23 layers CNN architecture was used. The experimental results achieved up to 97.8 percent and 100 percent classification accuracy for the employed datasets, respectively, exceeding all other state-of-the-art models.

There is no doubt that previous studies achieved significant improvement in brain tumor diagnosis. The focus of this research paper is mainly concentrated on improving the deep learning models and forecast accuracy.

2. Method

The proposed methodology for binary class brain tumor detection is shown in Fig 1. The architecture begins with extracting images and loading labels from the dataset. The extracted images are then split into training, validation, and test sets. Lastly, my proposed 19-layer CNN architecture is applied to the Br35H Mask dataset, and performance metrics are recorded. In the following sections, detailed descriptions of the blocks in my proposed method are discussed.

2.1 Dataset

In this study, the dataset used is the publicly available Br35H Dataset [8]. The dataset comprises a total of 3450 contrast MRI images, which can be categorized as either normal (no tumor) or abnormal (tumor present). It is accessible at

https://github.com/DamilolaEniola/bio-6306-brain-tumor-detection.git

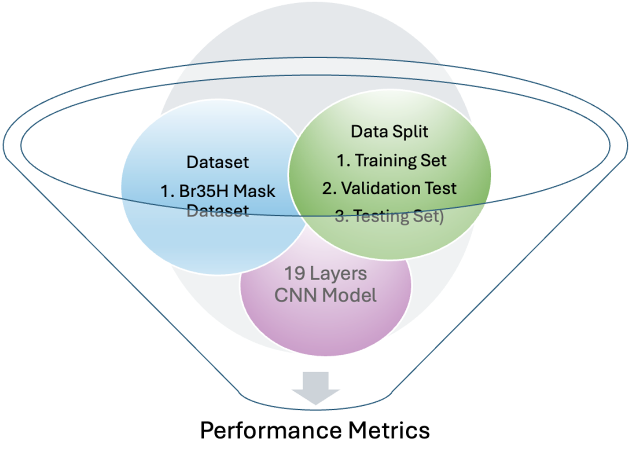


Figure 1. Proposed Architecture for Brain Tumor Detection

2.2 Proposed 19-layer CNN Architecture

Figure 2 demonstrates the proposed 19-layer CNN architecture used to classify whether a tumor exists or not. These layers are employed to process the images. The layers can be broken down as follows:

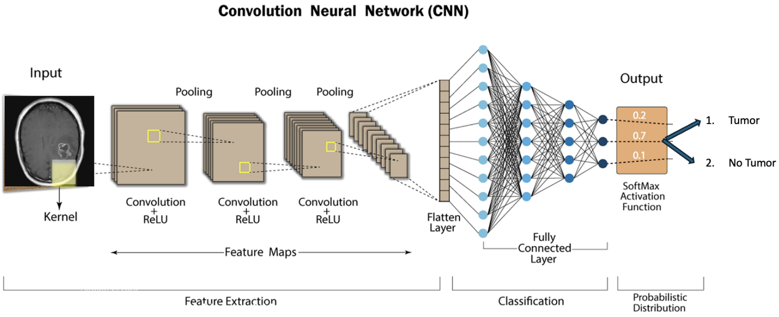
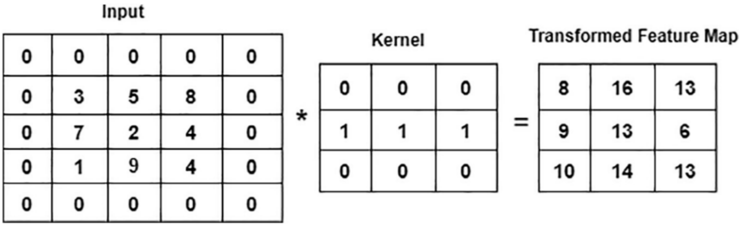


Figure 2. Proposed 19-layers CNN Architecture.

1. **Input Layer**
   (a) The model begins with an input layer that is designed to accept images with a height and width of 512x512 pixels and the appropriate channel.
2. **First Convolutional Layer**
   (a) This layer applies 64 kernels of size 22x22 to the input image with a stride of 2, thereby reducing the spatial dimensions of the feature maps (output that results from a kernel applied to the input image). An example of this is illustrated in Figure 3.
3. **Max Pooling Layer**
   (a) Thereafter, a max-pooling layer with a pool size of 4x4 is employed to further reduce the spatial dimensions while retaining the most relevant features.
4. **Batch Normalization Layer**
   (a) This is followed by a batch normalization layer that is applied to normalize the activations.



$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 5 & 8 & 0 \\ 0 & 7 & 2 & 4 & 0 \\ 0 & 1 & 9 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 8 & 16 & 13 \\ 9 & 13 & 6 \\ 10 & 14 & 13 \end{bmatrix}$$

Figure 3. Convolution Operation on a 5x5 Image using a 3x3 Kernel.

This helps stabilize and accelerate the training process by reducing internal covariate shift and allowing the network to learn more quickly and efficiently.

5. **Second Convolutional Layer**
   (a) This layer applies 128 kernels of size 11x11 with a stride of 2, maintaining spacial dimensions with the help of padding.
6. **Max Pooling Layer**
   (a) Thereafter, a max-pooling layer with a pool size of 2x2 is employed.
7. **Batch Normalization Layer**
   (a) This is followed by a batch normalization layer that is applied.
8. **Third Convolutional Layer**
   (a) This layer applies 256 kernels of size 7x7 with a stride of 2, thereby maintaining the spatial dimensions with padding.
9. **Max Pooling Layer**
   (a) Thereafter, a max-pooling layer with a pool size of 2x2 is employed.
10. **Batch Normalization Layer**
    (a) This is followed by a batch normalization layer that is applied.
11. **Fourth Convolutional Layer**
    (a) This layer applies 512 kernels of size 3x3 with a stride of 2 and padding.
12. **Max Pooling Layer**
    (a) Thereafter, a max-pooling layer with a pool size of 2x2 is employed.
13. **Batch Normalization Layer**
    (a) This is followed by a batch normalization layer that is applied.
14. **Global Average Pooling Layer**
    (a) After the convolutional layers, a global average pooling layer reduces the spatial dimensions of the feature maps to 1x1, essentially summarizing the information across spatial locations.
15. **First Fully Connected Layer**
    (a) This layer consists of 1024 units with ReLU activation. With the help of the ReLU activation function and fully connected layers, neural networks can learn low-dimensional representations of the input data by capturing and transforming relevant features into a more compact and informative space. This process enables the network to effectively model complex relationships in the data and make accurate predictions.
16. **Batch Normalization Layer**
    (a) This is followed by a batch normalization layer that is applied.
17. **Second Fully Connected Layer**
    (a) This layer consists of 512 units with ReLU activation.
18. **Batch Normalization Layer**
    (a) This is followed by a batch normalization layer.
19. **Third Fully Connected Layer**

(a) This layer consists of 256 units with Relu activation and batch normalization. Also, a dropout layer with a dropout rate of 20 percent is employed to prevent combat.
The output layer consists of a single unit with the sigmoid activation function, suitable for binary classification tasks. The sigmoid activation squashes the network's output to a value between 0 and 1, representing the probability of the input belonging to the positive class. A threshold of 0.5 is used further categorize the outputs.

Lastly the model is compiled, using the desired input and output layers ready for training and evaluation.

2.3 Training Validation Testing

The proposed models are implemented in TensorFlow using Keras in Python. The implementation was conducted on Google Colab. For the loss function, binary cross entropy was chosen, as it is commonly employed in binary classification tasks. This function computes the cross-entropy loss between the true labels and the predicted probabilities for each class independently. The Adam optimizer is utilized, and accuracy is employed as a common metric for evaluating performance in this problem domain. Once the data is divided into training, validation, and testing sets, the basic CNN performance is evaluated, while accuracy and loss values are plotted. Finally, the model is assessed on the test data.

2.4 Performance Metrics

To evaluate the performance of the 19-layer CNN, various evaluation metrics such as accuracy, precision, recall, false-positive rate (FPR), true negative rate (TNR), and F1 score are utilized. It is important to note that TP stands for True Positive, FP stands for False Positive, TN stands for True Negative, and FN stands for False Negative. These metrics are incorporated into the confusion matrix and the Receiver Operating Characteristic (ROC) curve. article amsmath

In the context of binary classification, various performance metrics are commonly used:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{5}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{6}$$

$$\text{FPR} = \frac{FP}{TN + FP} \tag{7}$$

$$\text{TNR} = \frac{TN}{TN + FP} \tag{8}$$

$$\text{F1 score} = 2 \cdot \frac{\text{recall} \times \text{precision}}{\text{recall} + \text{precision}} \tag{1}$$

These metrics provide insights into different aspects of the classifier's performance.

Please be aware that the depicted figures were either generated entirely by myself, taken from the inspired research paper, or sourced from the internet and subsequently modified to suit the specific requirements of my paper.

3. Results

The confusion matrix and the ROC curve for the Br35H-Mask-RCNN dataset are depicted in Fig. 4. It can be observed that a total of 220 MRI images are correctly classified for no tumor, and 230 MRI images are correctly classified for tumor. Only 7 images are misclassified by the proposed architecture. Other performance metrics, including accuracy, precision, recall, FPR, TNR, and F1-score, are presented in Table 1. As illustrated in Table 1, the overall prediction accuracy achieved on the dataset is 0.98. For the other performance metrics, an average precision of 0.985, an average recall of 0.985, and an average F1-score of 0.98 are achieved. The false-positive rate is approximately 0, and the true negative rate appears to be close to 1, demonstrating that the 19-layer CNN architecture can achieve excellent efficiency on the dataset. From the ROC curve, the area value is 0.985, indicating the consistency and generality of our model.

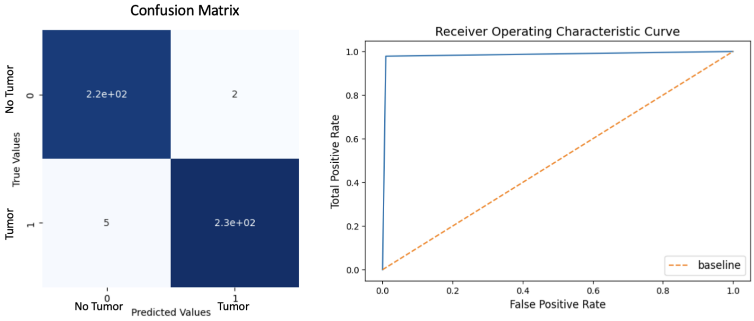


Figure 4. CNN Model's Performance a) Confusion Matrix and b)ROC Curve

Table 1. The Results Obtained using the CNN Model on the Br35H-Mask-RCNN Dataset

Metrics	Tumor Class	TP	TN	FP	FN	Accuracy	Precision	Recall	FPR	TNR	F1-score
Br35H-Mask-RCNN	Tumor	230	220	2	5		0.99	0.98	0.02	0.98	0.98
	No Tumor	220	230	5	2		0.98	0.99	0.02	0.98	0.98
	Average					0.98					

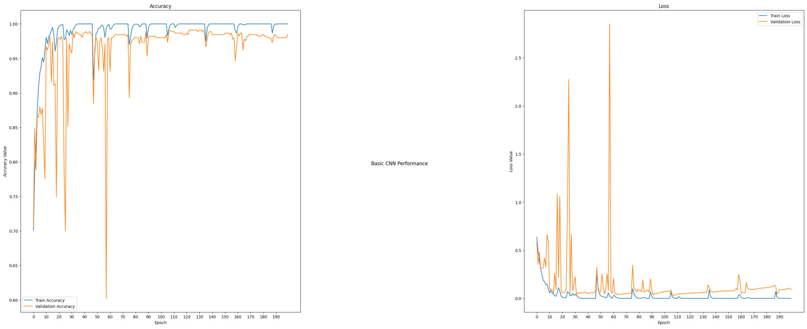


Figure 5. Progress for (a) Accuracy Value During Training and Validation Process and (b) Loss Value During Training and Validation Process

To train the model, a batch size of 10 was utilized with 200 epochs. Additionally, a binary cross-entropy loss function was employed for the binary classification problem. It can be observed from

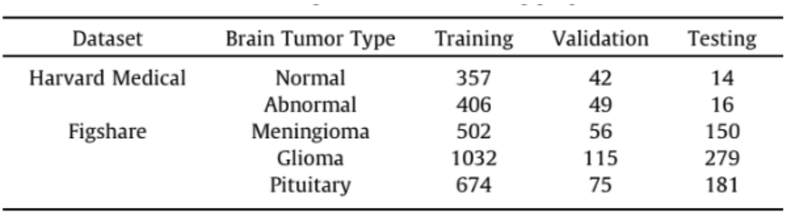
Fig. 5 that, right after the 36th epoch, 100 percent training accuracy is achieved. As depicted in Fig. 5(b), the loss value starts decreasing and approaches zero for both the training and validation sets after the 36th epoch. Overall, the training dataset demonstrates a more stable convergence compared to the validation dataset. Eventually, the validation dataset exhibits reduced instability after several epochs. Please utilize the zoom feature to see figures clearly.

3.1 Working Code Explanation vs Past Provided Code

This paper draws inspiration from the original work titled "Accurate Brain Tumour Detection Using Deep Convolutional Neural Network" by Md. Saikat Islam Khan et al. The subsequent section will present comparisons between the methodologies employed in the original article and the modifications implemented herein. Please utilize the zoom feature to see figures clearly.

Example 1

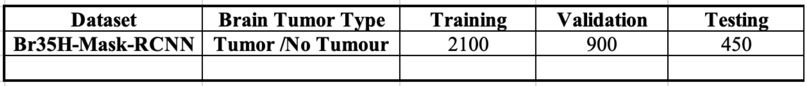
1. **Original Paper**: Utilized two sets of datasets. The Harvard Medical dataset focused on binary classification, while the Figshare dataset addressed multi-classification. See Figure 6.



Dataset	Brain Tumor Type	Training	Validation	Testing
Harvard Medical	Normal	357	42	14
	Abnormal	406	49	16
Figshare	Meningioma	502	56	150
	Glioma	1032	115	279
	Pituitary	674	75	181

Figure 6. Original Paper Dataset

2. **Modification**: Utilized the Br35H-Mask-RCNN dataset and addressed a binary classification problem. See Figure 7.



Dataset	Brain Tumor Type	Training	Validation	Testing
Br35H-Mask-RCNN	**Tumor /No Tumour**	2100	900	450

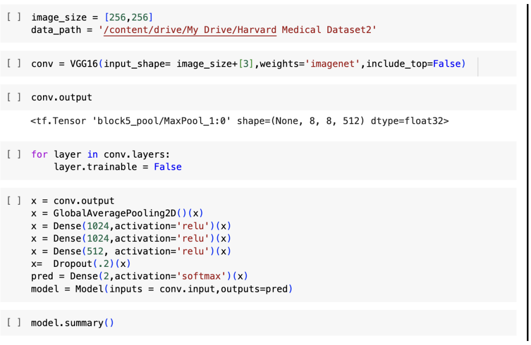
Figure 7. Modified Paper Dataset

Example 2

1. **Original Paper:** Incorporated the VGG-16 model along with a 23-layer CNN architecture. When dealing with limited volumes of data, which is the case in the second dataset, the proposed 23-layers CNN architecture faces an over fitting problem. To address this issue, transfer learning combined with VGG16 architecture and 23 layers CNN was used. See Figure 8.
2. **Modification:** Utilized a 19-layer CNN architecture, as an abundance of data was available. See Figure 9.

Example 3

1. **Original Paper:** Utilized the softmax function, which is an activation function commonly used for multiclass classification tasks, for the Figshare Dataset responsible for the classification of Meningioma, Glioma, and Pituitary. See Figure 10.



```
[ ]  image_size = [256,256]
     data_path = '/content/drive/My Drive/Harvard Medical Dataset2'

[ ]  conv = VGG16(input_shape= image_size+[3],weights='imagenet',include_top=False)

[ ]  conv.output

     <tf.Tensor 'block5_pool/MaxPool_1:0' shape=(None, 8, 8, 512) dtype=float32>

[ ]  for layer in conv.layers:
         layer.trainable = False

[ ]  x = conv.output
     x = GlobalAveragePooling2D()(x)
     x = Dense(1024,activation='relu')(x)
     x = Dense(1024,activation='relu')(x)
     x = Dense(512, activation='relu')(x)
     x=  Dropout(.2)(x)
     pred = Dense(2,activation='softmax')(x)
     model = Model(inputs = conv.input,outputs=pred)

[ ]  model.summary()
```

Figure 8. Original Paper VGG-16 Snippet

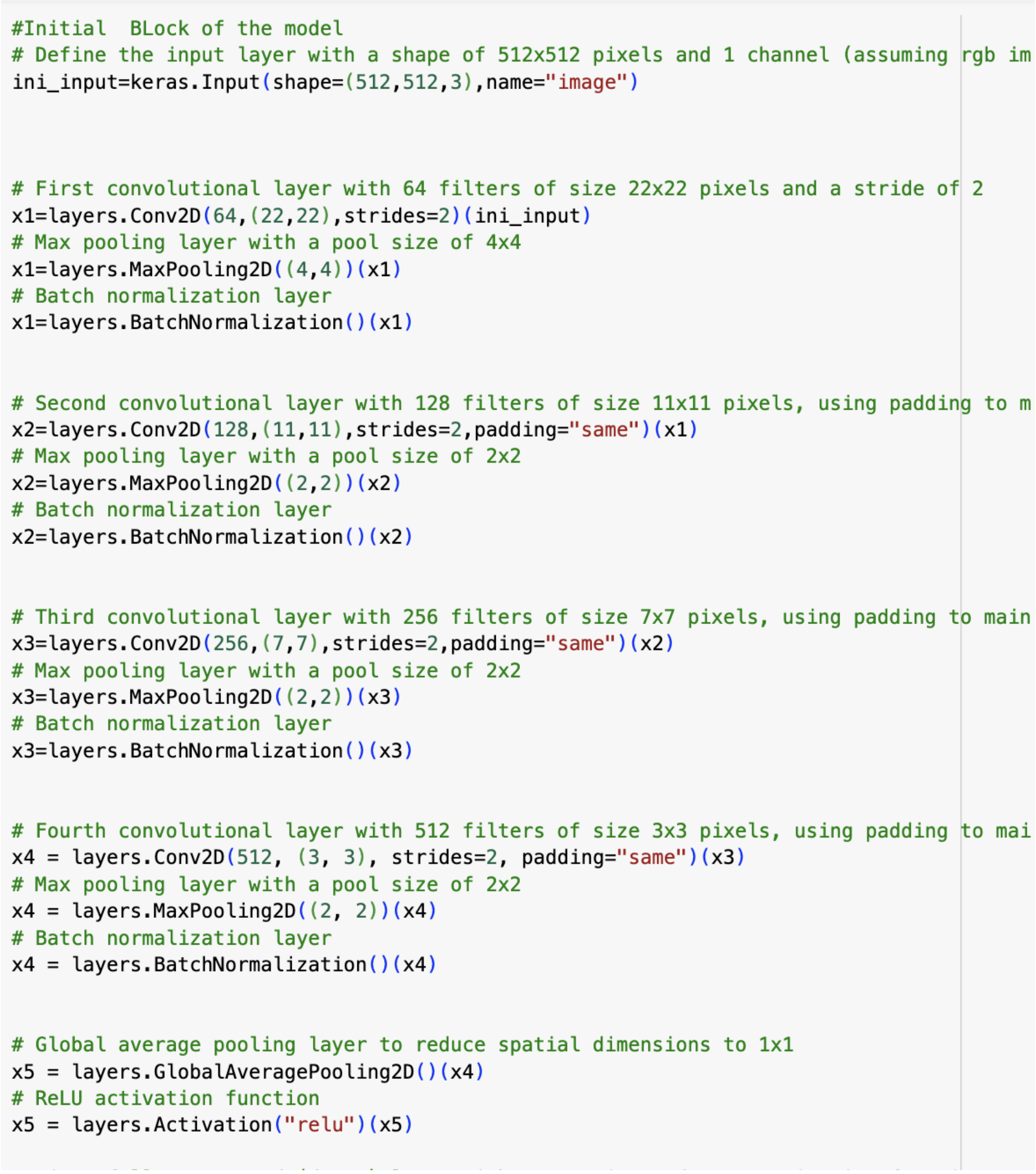
2. **Modification**: Utilized the sigmoid function, which is an activation function commonly used for binary classification tasks, for the Br35H-Mask-RCNN Dataset responsible for the classification of tumors or non-tumors. See Figure 11.

Example 4

1. **Original Paper**: In the CNN that was used for the binary classification task, and implemented for the Harvard Medical Dataset, there was a total parameter value of 5,063,362, and there was a trainable parameter value of 5,057,858. See Figure 12.
2. **Modification**: In the CNN that was used for the binary classification task, and implemented for the Br35H-Mask-RCNN Dataset, there was a total parameter value of 5,063,105, and there was a trainable parameter value of 5,057,601. See Figure 13.

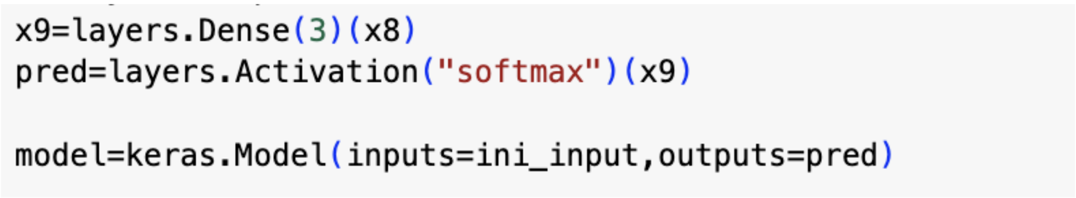
Example 5

1. **Original Paper**: In the CNN that was used for the binary classification task, and implemented for the Harvard Medical Dataset, a batch size of 10 and epoch of 80 was used. See Figure 14.
2. **Modification**: In the CNN that was used for the binary classification task, and implemented for the Br35H-Mask-RCNN Dataset, a batch size of 10 and epoch of 200 was used to combat overfitting. See Figure 15.



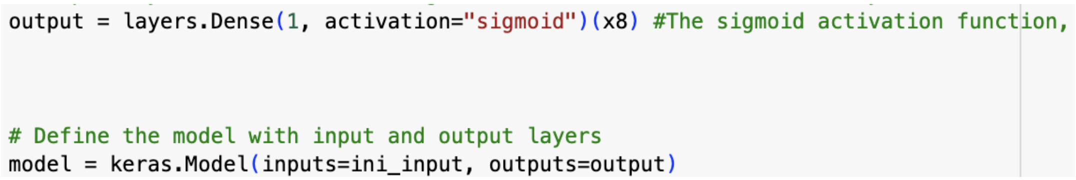
```
#Initial  BLock of the model
# Define the input layer with a shape of 512x512 pixels and 1 channel (assuming rgb im
ini_input=keras.Input(shape=(512,512,3),name="image")



# First convolutional layer with 64 filters of size 22x22 pixels and a stride of 2
x1=layers.Conv2D(64,(22,22),strides=2)(ini_input)
# Max pooling layer with a pool size of 4x4
x1=layers.MaxPooling2D((4,4))(x1)
# Batch normalization layer
x1=layers.BatchNormalization()(x1)


# Second convolutional layer with 128 filters of size 11x11 pixels, using padding to m
x2=layers.Conv2D(128,(11,11),strides=2,padding="same")(x1)
# Max pooling layer with a pool size of 2x2
x2=layers.MaxPooling2D((2,2))(x2)
# Batch normalization layer
x2=layers.BatchNormalization()(x2)


# Third convolutional layer with 256 filters of size 7x7 pixels, using padding to main
x3=layers.Conv2D(256,(7,7),strides=2,padding="same")(x2)
# Max pooling layer with a pool size of 2x2
x3=layers.MaxPooling2D((2,2))(x3)
# Batch normalization layer
x3=layers.BatchNormalization()(x3)


# Fourth convolutional layer with 512 filters of size 3x3 pixels, using padding to mai
x4 = layers.Conv2D(512, (3, 3), strides=2, padding="same")(x3)
# Max pooling layer with a pool size of 2x2
x4 = layers.MaxPooling2D((2, 2))(x4)
# Batch normalization layer
x4 = layers.BatchNormalization()(x4)


# Global average pooling layer to reduce spatial dimensions to 1x1
x5 = layers.GlobalAveragePooling2D()(x4)
# ReLU activation function
x5 = layers.Activation("relu")(x5)
```

Figure 9. Modified Paper CNN Snippet



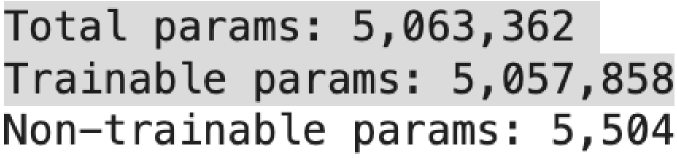
```
x9=layers.Dense(3)(x8)
pred=layers.Activation("softmax")(x9)

model=keras.Model(inputs=ini_input,outputs=pred)
```

Figure 10. Original Paper Softmax Function



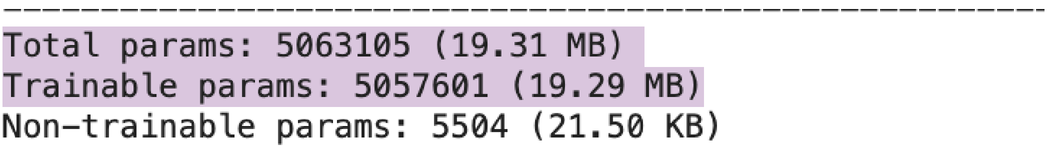
```
output = layers.Dense(1, activation="sigmoid")(x8) #The sigmoid activation function,


# Define the model with input and output layers
model = keras.Model(inputs=ini_input, outputs=output)
```

Figure 11. Modified Paper Sigmoid Function



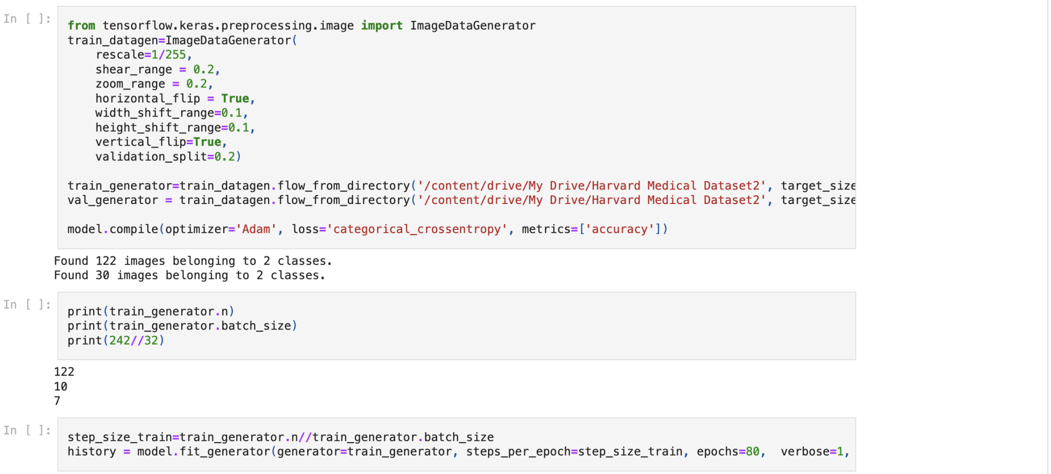
```
Total params: 5,063,362
Trainable params: 5,057,858
Non-trainable params: 5,504
```

Figure 12. Original Paper Dataset: CNN on Harvard Medical Dataset



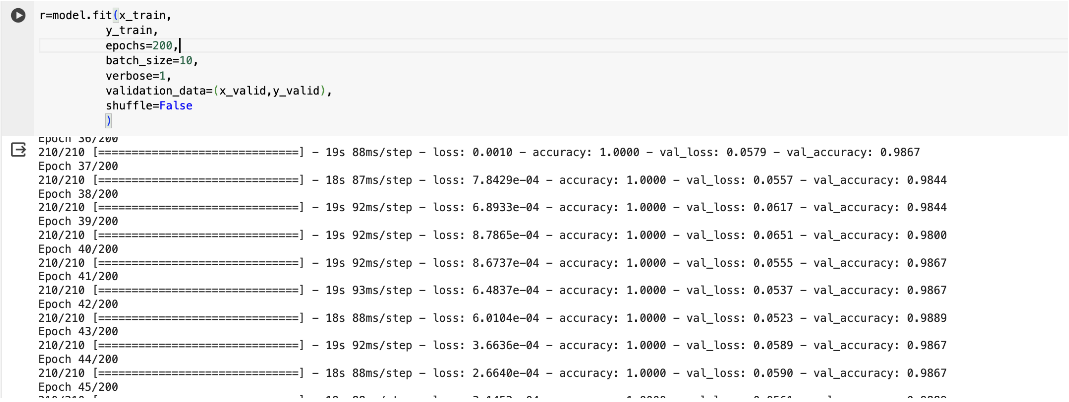
```
Total params: 5063105 (19.31 MB)
Trainable params: 5057601 (19.29 MB)
Non-trainable params: 5504 (21.50 KB)
```

Figure 13. Modified Paper Dataset



```
In [ ]: from tensorflow.keras.preprocessing.image import ImageDataGenerator
        train_datagen=ImageDataGenerator(
            rescale=1/255,
            shear_range = 0.2,
            zoom_range = 0.2,
            horizontal_flip = True,
            width_shift_range=0.1,
            height_shift_range=0.1,
            vertical_flip=True,
            validation_split=0.2)

        train_generator=train_datagen.flow_from_directory('/content/drive/My Drive/Harvard Medical Dataset2', target_size
        val_generator = train_datagen.flow_from_directory('/content/drive/My Drive/Harvard Medical Dataset2', target_size

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

       Found 122 images belonging to 2 classes.
       Found 30 images belonging to 2 classes.

In [ ]: print(train_generator.n)
        print(train_generator.batch_size)
        print(242//32)

       122
       10
       7

In [ ]: step_size_train=train_generator.n//train_generator.batch_size
        history = model.fit_generator(generator=train_generator, steps_per_epoch=step_size_train, epochs=80,  verbose=1,
```

Figure 14. Original Paper Dataset: CNN on Harvard Medical Dataset



```
r=model.fit(x_train,
            y_train,
            epochs=200,
            batch_size=10,
            verbose=1,
            validation_data=(x_valid,y_valid),
            shuffle=False
            )
```

```
Epoch 36/200
210/210 [==============================] - 19s 88ms/step - loss: 0.0010 - accuracy: 1.0000 - val_loss: 0.0579 - val_accuracy: 0.9867
Epoch 37/200
210/210 [==============================] - 18s 87ms/step - loss: 7.8429e-04 - accuracy: 1.0000 - val_loss: 0.0557 - val_accuracy: 0.9844
Epoch 38/200
210/210 [==============================] - 19s 92ms/step - loss: 6.8933e-04 - accuracy: 1.0000 - val_loss: 0.0617 - val_accuracy: 0.9844
Epoch 39/200
210/210 [==============================] - 19s 92ms/step - loss: 8.7865e-04 - accuracy: 1.0000 - val_loss: 0.0651 - val_accuracy: 0.9800
Epoch 40/200
210/210 [==============================] - 19s 92ms/step - loss: 8.6737e-04 - accuracy: 1.0000 - val_loss: 0.0555 - val_accuracy: 0.9867
Epoch 41/200
210/210 [==============================] - 19s 93ms/step - loss: 6.4837e-04 - accuracy: 1.0000 - val_loss: 0.0537 - val_accuracy: 0.9867
Epoch 42/200
210/210 [==============================] - 18s 88ms/step - loss: 6.0104e-04 - accuracy: 1.0000 - val_loss: 0.0523 - val_accuracy: 0.9889
Epoch 43/200
210/210 [==============================] - 19s 92ms/step - loss: 3.6636e-04 - accuracy: 1.0000 - val_loss: 0.0589 - val_accuracy: 0.9867
Epoch 44/200
210/210 [==============================] - 18s 88ms/step - loss: 2.6640e-04 - accuracy: 1.0000 - val_loss: 0.0590 - val_accuracy: 0.9867
Epoch 45/200
```

Figure 15. Modified Paper Dataset

4. Discussions

In this study, a model is used to diagnose binary (normal and abnormal) brain tumors. The created model is compared to the existing state-of-the-art models found in the literature, as shown in Table 3 below. It is evident from Table 3 that my proposed 19-layer CNN and the Support Vector Machine (SVM) algorithm demonstrate the same prediction performance, whereas the K-Nearest Neighbour (KNN) algorithm demonstrates the best prediction performance for the identification of binary tumors compared to other methods found in the literature. This is interesting and unexpected because despite the simplicity and reliance on local similarity measures, the KNN seems to be well-suited for the characteristics of binary tumor identification tasks. It appears that its ability to discern patterns within the data points might be advantageous in this context, outperforming more complex and non-subtle models like the 19-layer CNN. This surprising finding highlights the importance of exploring a diverse set of algorithms when exploring predictive models.

It is interesting to note that SVM with the combination of Particle Swarm Optimization (PSO) led to a slightly lower prediction accuracy than SVM alone. This may be due to the fact that while PSO is effective in fine-tuning SVM parameters, the addition of an optimization layer could potentially disrupt the inherent simplicity and robustness of the SVM algorithm. Alternatively, there is a computational overhead that PSO introduces that may negatively affect the efficiency of the overall modeling process, resulting in a slight loss of prediction accuracy.

4.1 Implications

The findings of this study have implications for both research and practical applications in the field of brain tumor diagnosis. For instance, the valuable insights extracted from this research regarding algorithm performance techniques contribute to the overall understanding of predictive modeling in medical diagnosis. Similar approaches can be applied to other imaging tasks, leading to the development of more robust tools across various healthcare domains, providing generalizability and, ultimately, better patient diagnostic outcomes.

4.2 Limitations and Future Work

Although my proposed model achieved promising classification outcomes, there are still a number of issues that can be resolved in future work. For example, while the model produced high prediction accuracy, it has not been validated in an actual clinical study. This serves as a limitation, as the validity of our model cannot be endorsed in clinical or industrial practices. Additionally, although

Table 3. Comparison of the proposed framework with the other state of art models

Method	Number of Images	Classifier	Classification Type	Accuracy
Gudigar et al. [9]	612	PSO + SVM	Binary Class	0.974
El-Dahshan et al. [10]	70	KNN	Binary Class	0.986
Chaplot et al [11]	52	SVM	Binary Class	0.98
Proposed Method	450	CNN	Binary Class	0.98

the model was trained on over 2000 datasets, more MRI images are still necessary to further test the model and improve its prediction tendencies. Thus, one future direction would be to test the model on actual clinical data when available. This would allow for a direct comparison of the performance of the proposed model with experimental approaches. Another future direction would be to employ more regularization techniques and adding more layers to the CNN model.

5. Conclusion

This research introduces adapted deep learning models for identifying brain abnormalities. The proposed 19-layer CNN architecture is designed to handle a relatively large volume of image data. Our experimental results demonstrate that the model enhances the prediction performance of diagnosing brain tumors. A prediction accuracy of 98 percent was achieved. Therefore, I believe that my proposed algorithm makes an excellent candidate for brain tumor detection. My proposed model, employed dataset, and all the source codes are publicly available at

https://github.com/DamilolaEniola/bio-6306-brain-tumor-detection.git

Acknowledgment

I would like to express my sincere gratitude to the University of Houston, Engineering Data Science department for their invaluable contributions to my research. Their assistance in offering pertinent courses such as Machine Learning, Data Science, Digital Image Processing, and Probability and Statistics has been instrumental to the successful completion of this study.

Author contributions

DEO reproduced the original study. For the original study, SIK, AR, and MKN conceived and initiated the study. SIK, AR, RK, and TD performed the experiments. SIK, AR, TD, SSB, AM, and ID wrote the manuscript. SIK, AR, MKN, SSB, MS, and ID helped with the literature review. AR, SSB, AM, ID, and TD mentored and analytically reviewed the paper. All the authors reviewed the article.

Funding statement

Not applicable.

Open data statement

I am committed to transparency and open science practices. All data supporting our findings in this paper are openly available in the bio-6306-brain-tumor-detection repository. The data and relevant files can be accessed at https://github.com/DamilolaEniola/bio-6306-brain-tumor-detection.git .This open access to data ensures the reproducibility of my results and enables further investigation and validation by the scientific community.

Reproducibility statement

To ensure the reproducibility of my research, I have made my code used in my analysis freely available in the bio-6306-brain-tumor-detection repository. This code encompasses all the computational methods employed in my study, from data preprocessing to statistical analysis. I encourage researchers to utilize this code to validate my findings and build upon my work.

References

[1] Mayo Foundation for Medical Education and Research. (2023, April 21). *Brain Tumor.* Mayo Clinic. https://www.mayoclinic.org/diseases-conditions/brain-tumor/symptoms-causes /syc-20350084#:~:text=A%

[2] *Brain Tumor - statistics.* Cancer.Net. (2023, May 31). https://www.cancer.net/cancer-types/brain-tumor/statistics

[3] Behin A, Hoang-Xuan K, Carpentier AF, Delattre J-Y. Primary brain tumours in adults. Lancet 2003;361(9354):323–31.

[4] Center for Devices and Radiological Health. (2017). *Benefits and risks.* U.S. Food and Drug Administration. https://www.fda.gov/radiation-emitting-products/mri-magnetic-resonance-imaging/benefits-and-risks

[5] Pareek, K., Tiwari, P. K., & Bhatnagar, V. (1970, January 1). *State of the art and prediction model for Brain tumor detection.* SpringerLink. https://link.springer.com/chapter/10.1007/978-981-16-2877-1_51

[6] Sultan HH, Salem NM, Al-Atabany W. Multi-classification of brain tumor images using deep neural network. IEEE Access 2019;7:69215–25.

[7] T. Hossain, F. Shishir, M. Ashraf, M. Al Nasim, F. Shah, Brain tumor detection using convolutional neural network, in: (pp. 1–6). IEEE., 2019 May 3.

[8] K. Md., R. Anichur.,. Debnath T., Karim Md., Nasir M., Band S., Mosavi A., Dehzangi I., Accurate Brain Tumor Detection Using Deep Convolutional Neural Network, in: (pp. 1–6). Elsevier., 2022 April 17.

[9] Gudigar A, Raghavendra U, San T, Ciaccio E, Acharya U. Application of multiresolution analysis for automated detection of brain abnormality using mr images: A comparative study. Future Gener Comput Syst 2019;90:359–67.

[10] El-Dahshan E-SA, Hosny T, Salem A-BM. Hybrid intelligent techniques for mri brain images classification. Digital Signal Process 2010;20(2):433–41.

[11] Chaplot S, Patnaik LM, Jagannathan N. Classification of magnetic resonance brain images using wavelets as input to support vector machine and neural network. Biomed Signal Process Control 2006;1(1):86–92.