Classification of Brain Tumors using Deep CNN

Anonymous CVPR submission

Paper ID

1. Introduction

Brain tumors are a severe medical issue that affects millions of individuals throughout the world. Under these circumstances, there is an ever-increasing need for fast and correct identification of brain tumors, which can be later used for more effective therapy. One of the most popular means for brain tumor diagnosis is medical imaging. Unfortunately, manual interpretation of medical pictures often proves to be time-consuming and error-prone, while on the other hand, the experts needed to evaluate these images are also hard to train, and the process of training such experts is time-consuming in itself. Therefore, one can arrive at the conclusion that other means are required for the processing of medical images.

1.1. Enter CAD Systems

Following the latest developments, deep learning has shown tremendous promise in a variety of medical image-processing applications, including brain tumor classification. Already Existing models of deep neural networks (DNN) and convolutional neural networks (CNN) have demonstrated cutting-edge performance in a variety of medical image categorization applications [1]. DNNs and CNNs are capable of autonomously learning high-level features from raw image data, which can increase the accuracy and efficiency of brain tumor classification, while also cutting down the overall time, and costs while also increasing the efficiency of diagnosis.

1.2. Study Goal

The goal of this research is to develop a deep convolutional neural network (CNN)-based CAD system for MRI-based brain tumor classification. The proposed method is predicted to provide high tumor classification accuracy and efficiency by utilizing CNNs' capacity to extract and learn important characteristics from input pictures. The created system will make use of a publically available dataset of brain tumor MRI images (One of the largest datasets for this task available in Kaggle) and a sequential model architecture comprised of convolutional and fully linked layers. Further on, certain experiments will be conducted, featuring partial changes in the baseline model in order to maximize both efficiency and accuracy. The suggested approach is projected to offer a viable alternative to radiologists' usual manual examination of brain tumor imaging, potentially allowing for faster and more accurate detection of brain tumors and therefore improving patient outcomes.

2. Dataset

We used a publicly available dataset of brain tumor MRI scans for training and testing our proposed model. The dataset contains images of different types of brain tumors, including glioma, meningioma, and pituitary tumors. The dataset is divided by default into a training set and a classification set. The subsets of each of these two sets were then modified to achieve the perfect balance between the three available classes for this classification (Only the balanced dataset will be provided in the attachments of this study; however a link will be provided to the original source of data). The images were preprocessed and normalized to improve the performance of our model. Toward the end of the study, specific experiments were conducted on the labeling of these images to see if the encoding (one-hot and ordinary) of the images has a noticeable level in the increase of accuracy.

2.1. Dataset Dimensions

Table 1. Dataset Dimensions

	Training set	**Validation set**
Glioma	1	2
Meningioma	3	4
Pituitary Tumors	5	6

2.2. Metrics

In this study, the featured models are trained and validated based on the following metrics:

1. Accuracy: Accuracy is a statistic that represents the proportion of true predictions made by the model out of all predictions made, independent of class. It is

computed by dividing the total number of predictions by the number of correct predictions:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

2. Precision: Precision is a statistic that represents the proportion of true positive predictions produced by the model out of all positive forecasts. It is computed by dividing the total number of positive predictions by the number of true positive predictions.

$$\text{Precision} = \frac{\text{Number of true positive predictions}}{\text{Total number of positive predictions}}$$

3. Class Accuracy: Class accuracy is a statistic that calculates the percentage of right predictions for a certain class out of all predictions made for that class. It is obtained by dividing the total number of samples in a given class by the number of valid predictions.

$$\text{Class accuracy} = \frac{\text{Number of correct predictions for a particular class}}{\text{Total number of samples in that class}}$$

3. Proposed Model

Our proposed model consists of a series of convolutional and pooling layers followed by fully connected layers. The input for the model is a grayscale image of size 128 x 128. The images are organized in batches of 200 images each for a total of 70 batches in the training set.

The output of the model is a one-hot encoding between three possible classes following the format [0,0,1].

The proposed model's design is comparable to that of several prominent CNN models for image classification, such as VGGNet and ResNet. These models generally start with multiple convolutional layers, then go on to batch normalization layers, max-pooling layers (the previous layers come in groups together for example 'Convolutional-Convolutional-Maxpool'), and lastly dense layers for classification and drop-out layers to make the number of parameters more manageable.

The model can extract features from the input photos thanks to the usage of convolutional layers. Max pooling layers are utilized to minimize the dimensionality of the feature maps, resulting in a more computationally efficient model.

Furthermore, the inclusion of dropout in the dense layers aids in the prevention of overfitting, a prevalent problem in deep neural networks. The softmax activation function was chosen for the output layer because it creates a probability distribution over the classes, which is suited for multi-class classification applications.

Overall, the architecture of this CNN model adheres to many of the best practices and design decisions that have been demonstrated in the literature to be effective for image classification tasks, making it a reasonable candidate for this purpose.

Layer (type)	Output Shape	Param #
conv2d_24 (Conv2D)	(None, 64, 64, 16)	416
conv2d_25 (Conv2D)	(None, 32, 32, 16)	2320
conv2d_26 (Conv2D)	(None, 32, 32, 32)	4640
max_pooling2d_8 (MaxPooling 2D)	(None, 16, 16, 32)	0
conv2d_27 (Conv2D)	(None, 16, 16, 32)	4128
max_pooling2d_9 (MaxPooling 2D)	(None, 8, 8, 32)	0
conv2d_28 (Conv2D)	(None, 8, 8, 64)	8256
conv2d_29 (Conv2D)	(None, 8, 8, 64)	16448
flatten_4 (Flatten)	(None, 4096)	0
dense_8 (Dense)	(None, 256)	1048832
dropout_4 (Dropout)	(None, 256)	0
dense_9 (Dense)	(None, 3)	771

Total params: 1,085,811
Trainable params: 1,085,811
Non-trainable params: 0

Figure 1. Parameters of the Proposed Model.

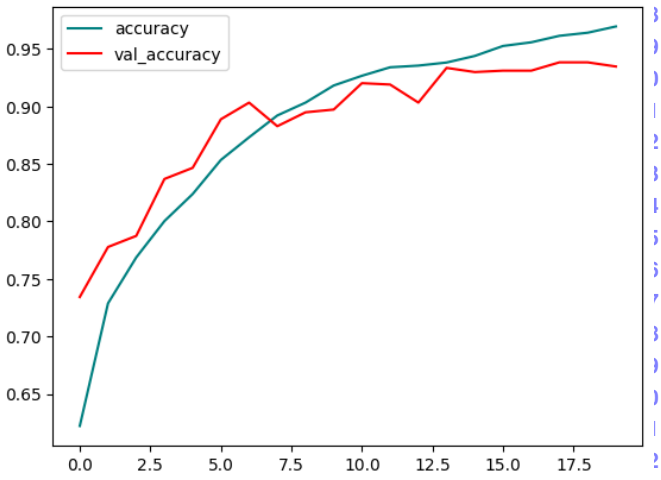

Figure 2. Accuracy of the Proposed Model after 20 epochs.

3.1. Results of the Proposed Model

The proposed model was trained for 20 epochs with a learning rate of 0.003. the results of the training are as follows:

Training Accuracy : 96.96

Validation Accuracy: 93.48

Precision: 93.47

Class Accuracy: 93.5

4. Experiments

Following the performance of the base line model many experiments were conducted; however this paper will only include the ones that reflect changes in the most important aspects of the model. The theme of the experiments will go as follows:

Addition of Batch Normalization

Switching Elu for Relu

Increasing the Training Time

Training for reduced image size

4.1. Addition of Batch Normalization

Batch normalization is important for training machine learning models because it helps to improve the stability and speed of convergence during training, as well as the generalization performance of the model.

Problem: The distribution of input to each layer of a deep neural network might change during training as the parameters of the preceding levels are changed. This might result in the so-called "internal covariate shift," which makes learning more difficult for the network and slows the convergence of the optimization method [3].

Solution: By applying a simple linear adjustment to the activations, batch normalization solves this problem by normalizing the inputs of each layer to have zero mean and unit variance. This aids in the stabilization of the input distribution, making it easier for the network to learn and lowering the chance of disappearing or bursting gradients. Furthermore, batch normalization functions as a type of regularization, lowering the model's generalization error by minimizing overfitting.

Proposed Model Poss batch normalization:

Training Accuracy: 98.32

Validation Accuracy: 92.27

Precision: 92.38

Class Accuracy: 92.27

4.2. Switching for ELU

The ELU (Exponential Linear Unit) activation function is a popular alternative to the ReLU (Rectified Linear Unit) activation function. Some of the reasoning behind such popularity are as follows:

Smoothness: The ELU activation function is smooth throughout, including at zero, implying that it has a continuous derivative throughout. This can assist to enhance the training optimization process, especially when utilizing gradient-based optimization techniques.

ELU also helps to mitigate the vanishing gradient problem, which can arise when utilizing the ReLU activation function, particularly in deep neural networks. Because the ELU activation function contains negative values, the output for certain inputs might be negative, whereas the ReLU function is zero for all negative inputs. This can aid in the reduction of the vanishing gradient problem and the training of deeper neural networks [2].

Results for training with ELU:

Training Accuracy: 87

Validation Accuracy: 85.5

Precision: 85.8

Class Accuracy: 85.5

4.3. Extended Training

Something interesting to note in the above experiment is that there is almost no overfitting, although the accuracy after 20 epochs of training is not as high as for the relu model. Nevertheless, because of the former observation, it was decided to go for an additional 40 epochs of training to see the limits of this model.

After training for an additional 40 epochs the end results are as follows;

Training Accuracy: 97.93

Validation Accuracy: 93.24

Precision: 93.35

Class Accuracy: 93.236

4.4. Training for Smaller Images

While all the previous experiments were done using images of size 128x128x1, although they might have displayed slightly different behaviors and results, something that can not be denied is the amount of time it took to train each model. Although it is commonly known that a reduction in image size will result in a reduction of accuracy for the model; it was deemed as important, for the scope of this study to see just how big the difference is:

After reducing the image size to 64x64x1 for the Relu model after batch normalization was included we ended up with the following results:

Accuracy (20 peochs): 93Validation Accuracy (20 epochs): 71Accuracy (60 peochs): 96Validation Accuracy (60 epochs): 81

5. Conclusions

There are several conclusions that were drown through out these experiments

1: The change in the activation function, although minor can lead to noteworthy differences in training (In the second experiment, a change in the activation function and a

slightly lower learning rate brought the over fitting down; however the catch was that the model needed more training to reach comparatively the same level of accuracy as the model in the experiment one)

2: The main problem in training a machine learning model is often related not to the complexity of the task (Comprehension wise), but the training time required for any model to learn (the limitations of the CPU units). Therefore sometimes it might be worth considering following certain practices and slightly changing certain hyperparameters to achieve a desirable level of training within a reasonable time

3: Although it is possible to achieve fairly high levels of accuracy in both the training set and the validation set, in most cases there will be slight, marginal, or considerable overfitting. These might come from a plethora of reasons, and though not impossible to fix, it just goes to prove that simply achieving high levels of accuracy in a training environment does not make a model ready to for real-life usability, especially in matters of clinical importance where a single slight mistake can cost lives

6. References

Reference list [1] Arif, M., Ajesh, F., Shamsudheen, S., Geman, O., Izdrui, D. and Vicoveanu, D. (2022). Brain Tumor Detection and Classification by MRI Using Biologically Inspired Orthogonal Wavelet Transform and Deep Learning Techniques. Journal of Healthcare Engineering, [online] 2022, p.e2693621.

[2] doi:https://doi.org/10.1155/2022/2693621.KILIÇARSLAN, S., ADEM, K. and ÇELİK, M. (2021). An overview of the activation functions used in deep learning algorithms. Journal of New Results in Science, 10(3), pp.75–88.

[3] doi:https://doi.org/10.54187/jnrs.1011739.Schilling, F. (2016). The Effect of Batch Normalization on Deep Convolutional Neural Networks. [online] DEGREE PROJECT COMPUTER SCIENCE AND ENGINEERING. Available at: https://www.diva-portal.org/smash/get/diva2:955562/FULLTEXT01.pdf.

7. Other important links

Link to the moodle page containing the class code, which served as inspiration for this study: https://activity.act.edu/lms/course/view.php?id=210
Link to the google drive file containing the Google colab file and the dataset: https://drive.google.com/drive/folders/1CYbcJ$_X2hCIg2xHiutF_MvqaxXMZKw5I?usp = sharing$