

DEEP LEARNING MODELING TO CONTRIBUTE IN THE PROCESSING OF MEDICAL IMAGING, APPLICATION FOR BINARY, MULTI-CLASS CLASSIFICATION OF A BRAIN TUMOR

HASINAVALONA Henintsoa Seth Etienne¹, RANDRIAMITANTSOA Paul Auguste²
RAJAONARISON Tianandrasana Romeo³

¹ PhD Student, TASI, ED-STII, Antananarivo, Madagascar

² Thesis Director, TASI, ED-STII, Antananarivo, Madagascar

³TASI, ED-STII, Antananarivo, Madagascar

ABSTRACT

Computer vision is one of the fastest growing fields thanks to deep learning and the sheer numbers of data circulating today. There are many areas of application, including autonomous car driving, image classification, facial recognition, art, environmental and health domains. The computer vision algorithm uses image segmentation to know each element that makes up the image by combining the image classification algorithm with object localisation and other algorithms. Among the fields mentioned, medical imaging takes a major place in terms of computer vision research. The brain is the main organ, the center of motor activity in the human body. The diagnosis of the brain is very delicate and complex and is the subject of much research and study. Several methods such as MRI, CT scan. In the clinical diagnosis and treatment of brain tumours, the manual reading of images consumes a lot of energy and time, as the acquisition has to be repeated as many times as there are slices, which leads to patient fatigue. MRI takes 30 minutes to 1 hour, generating many unnecessary images which slows down the treatment and makes the patients tired. One solution to help with this technique is the use of a deep learning model that will train multiple medical images, fill in missing data from MRI or CT scans and test the image from an MRI or CT scan to help doctors make a diagnosis. The objective of the present work is to create a robust deep learning algorithm to assist in the binary and multi-class classification of a brain tumour. The algorithm is created in its entirety from scratch. All regularisation techniques to remove overfitting and optimisation techniques to find parameters quickly have been tested and implemented to have a robust algorithm. For the dataset, our method can achieve a maximum accuracy for validation of 96.88% for binary classification, 93.38% for multi-class classification and 98.37% for binary classification using the transfer learning technique.

Keyword: *Neural network, deep learning, medical imaging, algorithm robust, computer vision, dataset, Regularization, Batch, classification.*

1. EXISTING WORKS

There are several studies on brain tumor detection using MRI that have been done previously. Discrete Wavelet Transform (DWT), Continuous Wavelet Transform (CWT), and Support Vector Machine (SVM) methods are used to detect brain tumors [1]. This method achieves sufficiently high results to detect brain tumours; however, there are still weaknesses in the calculation. [2] Even though the results are quite high, the configuration of the created model is not included and the dataset is not explained. [2][3]

The classification of brain tumours using machine learning methods has already been studied by researchers, especially in recent years. The development of artificial intelligence and new technologies based on deep learning has had a great impact in the field of medical image analysis, especially in the field of disease diagnosis (Mehmood et al. 2020, 2021; Yaqub et al. 2020). In parallel, many studies have been conducted on the detection and multi-classification of brain tumours using CNN. [4]

The following researchers have adopted pre-trained CNN models using a transfer learning approach for brain tumour classification. For example, Çınar and Yildirim (2020) used a modified form of the pre-trained ResNet-50 CNN model by replacing its last 5 layers with 8 new layers for brain tumour detection. They achieved 97.2% accuracy using MRI images with this modified CNN model. [4]

S. Kevin Zhou, Hayit Greenspan, Dinggang Shen have written books on deep learning for medical image analysis [5]. Apart from several papers written by researchers such as Ali ARI, Davut Hanbay [6], Justin Paul [7], several competitions have been conducted in order to get maximum accuracy for a limited amount of data.

2. ALGORITHM AND IMPLEMENTATION

2.1. Deep learning technics

Definition 01:

Machine learning is a field of artificial intelligence that involves programming a machine to learn to perform tasks by studying examples of tasks. These examples are represented by data that the machine uses to develop a model. [8] A model as a function of the type

$$f(x) = wx + b \quad (01)$$

The objective is to find the parameters w and b that give the best possible model, i.e. the model that best fits our data. To do this, an optimization algorithm is programmed into the model to test different values of w and b until we obtain the combination that minimizes the distance between the model and the points (or that minimizes the errors between the model output and the expected response).

Definition 022:

Deep learning is a field of machine learning in which, instead of developing one of the machine learning models, artificial neural networks are developed instead.

The neural network receives input data parameters, processes these data with other parameters through complex calculations and returns results, sometimes a prediction. A neural network is therefore a succession of more or less complex calculations to learn a correct output.

2.1.1. Activation function

In the field of artificial neural networks, the activation function is a mathematical function applied to an output signal of an artificial neuron. The term "activation function" comes from the biological equivalent "activation potential", a stimulation threshold which, once reached, causes a response from the neuron. [9]

When building a deep learning model, one of the choices to be made is the choice of the activation function, used in the hidden layers. A popular choice in deep learning is the ReLU function

$$a(z) = \max(0, z) \quad (02)$$

By using the ReLU activation function, the neural network will often learn much faster, so it replaces all negative values received as inputs with zeros.

A disadvantage of the ReLU function is that its derivative becomes zero when the input is negative which can prevent backpropagation of the gradient. We can then introduce a version called Leaky ReLU defined by:

$$a(z) = \max(\epsilon z, z) \text{ for } z \text{ real} \tag{03}$$

The parameter ϵ is a strictly positive real less than 1

$$\epsilon \in]0,1[$$

The derivative is then equal to ϵ when x is strictly negative, which allows to keep the update of the weights of a perceptron using this activation function. For a binary classification, we will use rectifier functions as activation, i.e. ReLU functions, and the sigmoid function as output.

2.1.2. Logistic regression

This is a learning algorithm that is used when the output labels y in a supervised learning problem is $\{0,1\}$ In logistic regression, we can use a sigmoid function, which solves the output between $\{0,1\}$ Because a sigmoid function is represented as follows:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{04}$$

Si $\begin{cases} z \text{ larger , } \sigma \text{ tends toward } 1 \\ z \text{ smaller , } \sigma \text{ tends toward } 0 \end{cases}$

For a simpler representation, here is a model with an input layer, a hidden layer and an output layer

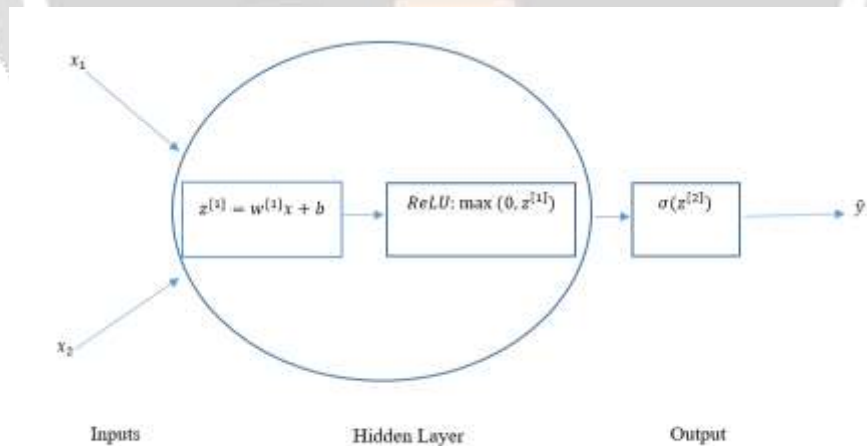


Figure 01: Overview of a network model with a connected layer and an output layer

2.1.3. The loss function

To evaluate the quality of a neural network model, i.e. to find the w and b values that fit our model minimizing the cost function. The objective is to make the model understand whether its prediction is satisfactory or not. [15] There are several cost functions like:

- The squared error

$$\mathcal{L}(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2 \tag{05}$$

It turns out that we can use this function, but in logistic regression we don't use this method because when we have to learn the parameters, we will find that the optimization problem becomes non-convex, and the gradient descent may not find the global optimum. So what we will use in linear regression is in fact the cross entropy loss function. [15]

- Cross entropy

$$\mathcal{L}(\hat{y}, y) = -(y \log \hat{y} + (1 - y) \log(1 - \hat{y})) \tag{06}$$

Note that the aim is for this loss function to be as small as possible

if $y = 1$ $\mathcal{L}(\hat{y}, y) = -y \log \hat{y}$
 if $y = 0$ $\mathcal{L}(\hat{y}, y) = -\log(1 - \hat{y})$

The loss function is defined for a single learning example (x^1, y^1) so it measures performance on a single learning example.

2.1.4. Cost function

The cost function is the sum of the loss function which measures the performance on the integer data sets $(x^1, x^1), \dots, (x^m, y^m)$ this function is defined by :

$$J(\omega, b) = \frac{1}{m} \sum_{i=1}^m \mathcal{L}(\hat{y}^{(i)}, y^{(i)}) \tag{07}$$

$$J(\omega, b) = -\frac{1}{m} \sum_{i=1}^m [y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})] \tag{08}$$

We can find the values of w and b

$$\omega = \omega - \alpha \frac{dJ(\omega, b)}{d\omega} \tag{09}$$

$$b = b - \alpha \frac{dJ(\omega, b)}{db} \tag{10}$$

α is the learning rate

2.1.5. Steps to develop and train a neural network model

So to develop and train artificial neural networks, then we repeat the following steps until we find the value of w and b that minimizes the cost function:

- Forward propagation: data is passed from the first layer to the last layer to produce an output \hat{y}
- Calculate the cost function: we calculate the error between this output \hat{y} and the reference output y that we wish to have
- Backward propagation: we measure how this cost function varies with respect to each layer in our model, using gradient descent algorithm to find parameters.

2.2. Convolutional neural networks

Definition 013 :

Convolutional neural networks are a sub-category of neural networks: they therefore have all the characteristics and different methodologies listed above. However, CNNs are specifically designed to process input images. [10]
 The architecture of a convolutional neural network is formed by a succession of processing blocks to extract the features that discriminate the class of membership of the image from others. A processing block consists of one or more:

- Convolution layers (CONV) which process data, extract features from a receiving field
- Correction layers (ReLU)
- Pooling layers (POOL), which compress the information by reducing the size of the intermediate image (often by subsampling).

The processing blocks follow each other to the final layers of the network, which classify the image and calculate the error between the prediction and the target value:

- Fully connected layer (FC), which is a perceptron layer;
- Loss layer (LOSS)

So for our convolution *layer* [l], the different parameters and hyper-parameters are :

- For the input, the size is :

$$n_H^{l-1} \times n_W^{l-1} \times n_c^{l-1} \tag{11}$$

- The filter size *f* is : $f^{[l]}$
- The padding is : $p^{[l]}$
- The stride is : $s^{[l]}$
- For the output, the size is :

$$n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]} \tag{12}$$

Wich

$$n_H^{[l]} = \frac{n_H^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \tag{13}$$

- Filter number: $n_c^{[l]}$
- For each filter, the size is : $f^{[l]} \times f^{[l]} \times n_c^{[l-1]}$
- Activation : $a^{[l]} : n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$
- Weights : $f^{[l]} \times f^{[l]} \times n_c^{[l-1]} \times n_c^{[l]}$
- Bias : $n_c^{[l]}$

Let us illustrate using the following figure the structure of a convolution layer

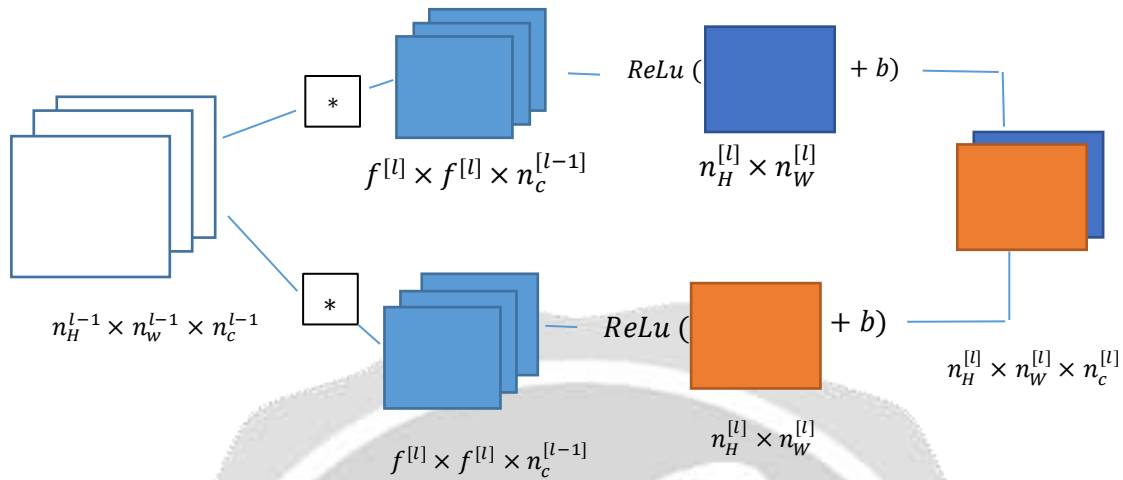


Figure 02: Computational graph for convolutional network overview

3. MODEL REALIZATION

The choice of hyper-parameters in deep learning is a very important step to optimize the algorithm and make it robust. These choices lie on:

- The number of layers
- The number of filters
- The choices on the filter sizes
- Which activation function to use for the different layers
- The numbers of layers connected
- The optimizer to use, hyper-parameters of the optimizer like β_1, β_2
- The learning rate

3.1. The datasets

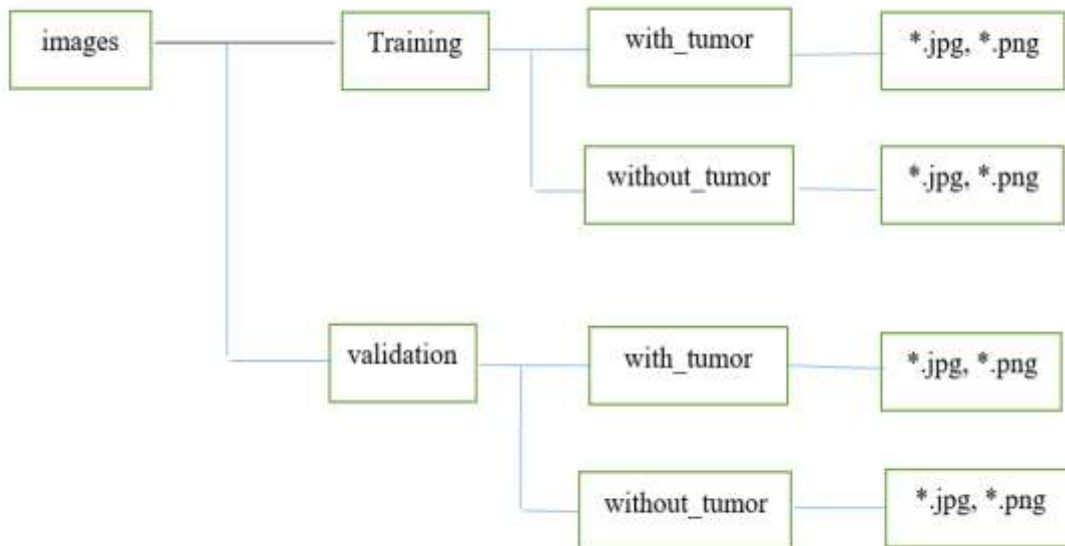
3.1.1. Data structure

The training and validation data were downloaded from the KAGGLE website [11]. The data is divided into two parts: one part containing tumors and one part without a tumor presence. The images look like



Figure 03: With tumor and without tumor images

The files and data in our machine are structured as follows



We have 3000 images belonging to both classes.

As our data is limited, we decided to divide the training data and the validation data into 80% and 20%.

Training data :80%	Validation data : 20%
--------------------	-----------------------

3.1.2. Data augmentation

Insufficient data risks unlearning of the model, as the model cannot even learn from the data. The lack of data also leads to overfitting of the network, as the model trains on the limited training data and has good training accuracy, but fails to recognize a new image that it has not studied.

One solution to this is image augmentation. The different techniques are:

Axial symmetry, Rotation, Random cropping, Color change, Noise addition, Zoom, Contrast change [12].

The following is a representation of an image after augmentation:

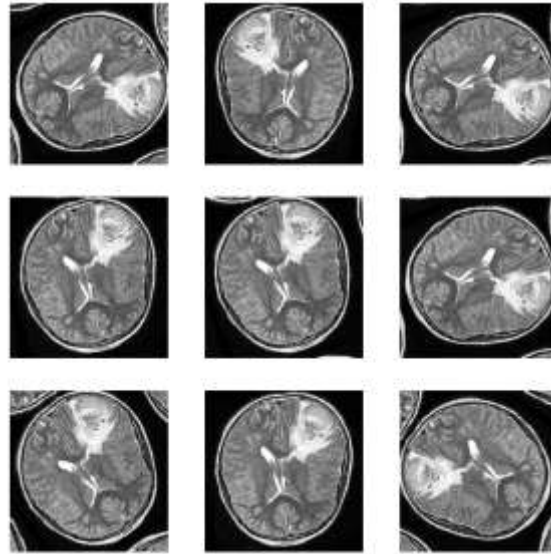


Figure 04:overview after a data augmentation

3.2. Numbers and size of filters

When designing the architecture of a neural network model, the effectiveness of a model lies in the choice of hyper-parameters. Among these hyper-parameters, there are the choices on the filters to be used like, how to choose between filters of size 1×1 or 3×3 or 5×5 and many others. So the inception network allows you to use all these choices, it makes the architecture complicated, but it works remarkably well.

Introduced by Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke and Andrew Rabinovich in their paper "Going deeper with convolutions", the basic idea is that instead of having to choose the size of the filters or the pooling layer, one can test all the options and then concatenate the outputs. Here is the computation graph for an input size, let's take 28×28 as an example

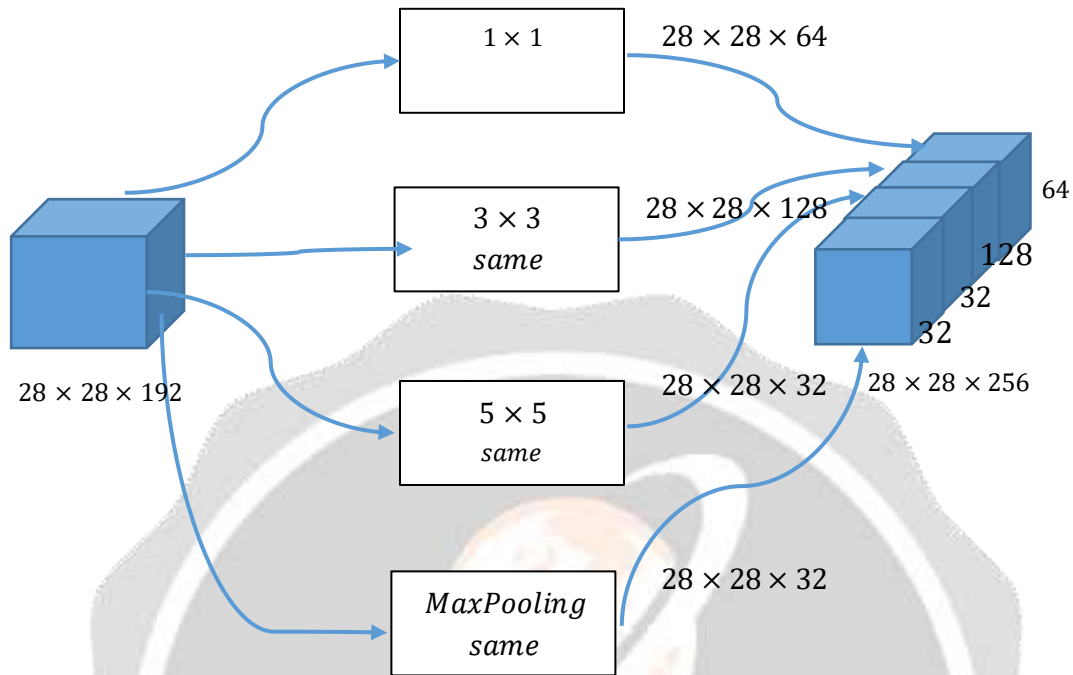


Figure 05:Inception model

Inception model makes the network architecture more complicated, makes the calculation very slow and takes up resources but it works perfectly well. The numbers and sizes of the filters we have chosen are detailed in the next section.

3.3. Improving learning through normalization and regularization techniques

We used the following different regularizations and normalizations

Batch normalization: Batch normalization is a step that normalizes the batch $\{x_i\}$, with a choice of parameters γ, β . Noting μ_B, σ_B^2 , the mean and variance of what we want to correct for the batch [12].

$$x_i \leftarrow \gamma \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} + \beta \tag{15}$$

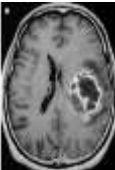
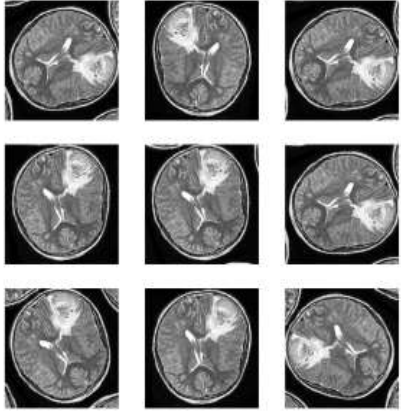
Dropout: Often in a connected layer, nodes have the same weights. Dropout is a technique that is intended to prevent overfitting on training data by dropping units in a neural network with probability $p > 0$. This forces the model to avoid relying too heavily on a particular set of features. [12]

Coefficient regularization: To ensure that the coefficients are not too large and that the model does not overfit the training set, regularization techniques are used on the model coefficients. We used an L2 regularization [12].

Early stopping: Early stopping is a regularization technique that consists of stopping the training step as soon as the validation loss reaches a plateau or starts to increase. [12]

The structure of our binary classification model is as follows

Tableau 01:Structure of our binary classification model

Input image (150,150,3)	
Image after data augmentation	
Convolutional layer N°01	Number of filter=64, size=5X5, padding = same
BatchNormalisation	BatchNormalization ()
Activation	ReLU
MaxPooling	Size : (2X2)
Convolutional layer N°02	Number of filter =64, size =5X5, padding = same
BatchNormalisation	BatchNormalization ()
Activation	ReLU
MaxPooling	Size: (2x2)
Convolutional layer N°03	Number of filter =64, size =5X5, padding = same
BatchNormalisation	BatchNormalization ()
Activation	ReLU
MaxPooling	Size: (2x2)
Convolutional layer N°04	Number of filter =128, size =5X5, padding=same
BatchNormalisation	BatchNormalization ()
Activation	ReLU
MaxPooling	Size: (2X2)
Convolutional layer N°05	Number of filter =256, size =3X3, padding=same
BatchNormalisation	BatchNormalisation ()
Activation	ReLU
MaxPooling	Size: (2X2)
Flatten	Flatten ()
Régularization	Dropout (0.35)
Dense	Number = 512
Régularization	L2 :0.001
Output Layer	Number of class = 1, activation = sigmoid

4. RESULTS AND DISCUSSION

For this simulation, we used the stochastic gradient algorithm which is therefore an optimized iterative gradient descent method. This method is the method of SGD with momentum which keeps in memory the update at each step of Δw , and calculates the next one as a convex combination of the current gradient and the previous modification. If we note w_n the coefficients obtained after n iterations, as well as Δw_n the n -th update of the parameters, then

$$\Delta w_{n+1} := \eta \nabla Q_i(w) + \alpha \Delta w_n \quad (16)$$

$$w_{n+1} := w_n - \Delta w_{n+1} \quad (17)$$

Learning rate $\alpha = 0.01$

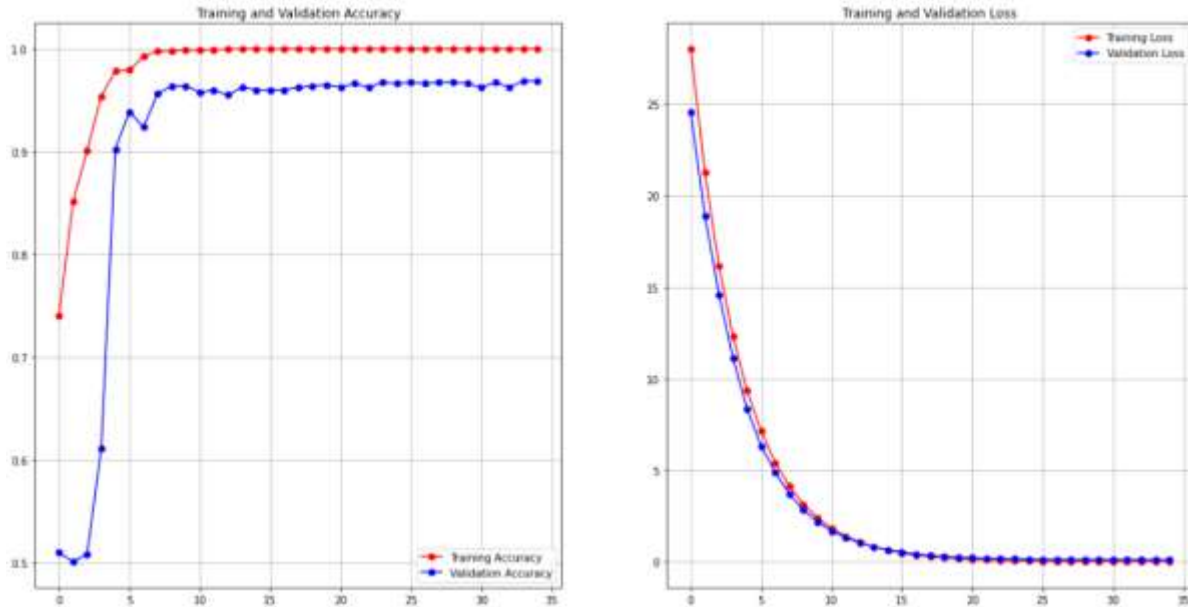


Figure 06:Result of our binary classification model, at 35 epochs and using the SGD as optimizer

The first observation is that the model manages to train correctly, there is no real inconsistency in training and validation.

The gap between validation and training between epochs 0 and 2 is normal, as the model starts to learn the parameters, and from the seventh epoch onwards the accuracy keeps increasing for training and validation.

The accuracy for training tends towards 1.0, in terms of probability, and reaches the value of 1.0 from the eighth epoch, and the maximum accuracy for validation reaches 0.9688. There is no loss or gradient explosion, and the model is not complicated, which makes learning faster.

The good configuration of the batch size suppressed the fluctuation in our result.

In summary, the model is able to identify both tumor and normal brain images and has better accuracy, no fluctuation and is fast to learn.

5. MULTI-CLASS CLASSIFICATION

Our classification model can detect whether or not there is a brain tumor in a given MRI image, and given the above result, the model is able to identify the images with better accuracy. This classification is only binary, but if we want to interpret the images as what type of disease or cancer it is, then our model is limited. The solution is to use a multi-class classification.

The data is downloaded from the KAGGLE website [8], and classified into training data and validation data, and then subdivided into four types:

Tableau 2: Number of images and data structure for multi-class classification

Type	Number of training dataset	Number of validation dataset
glioma_tumor	826	102
meningioma_tumor	822	111
pituitary_tumor	827	156
no_tumor	395	98

Below is a global structure of our model using the softmax function, with four classes, as the activation function at the output of our model:

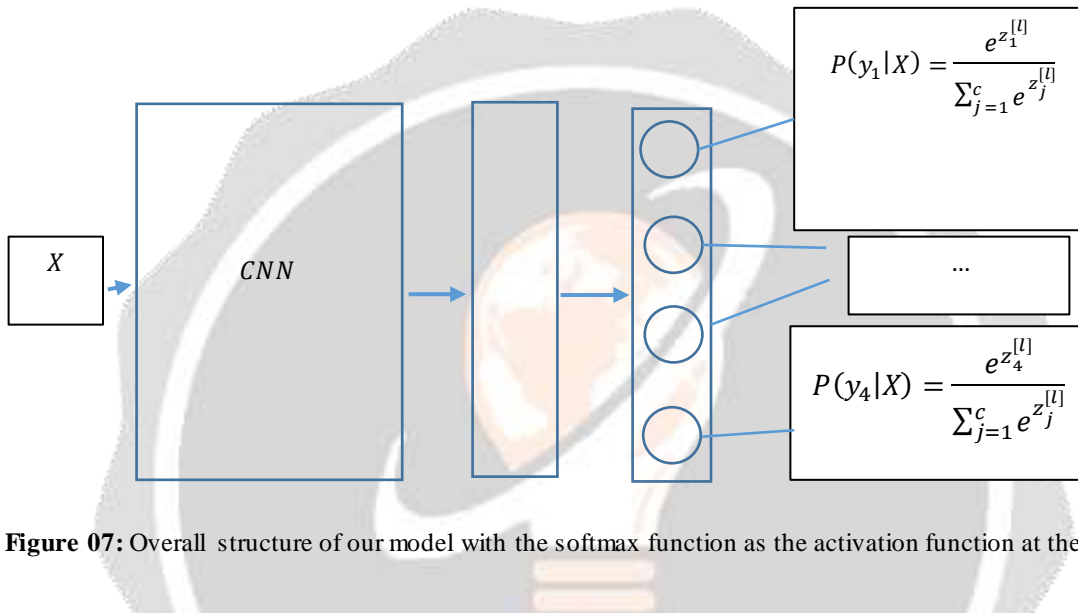


Figure 07: Overall structure of our model with the softmax function as the activation function at the output

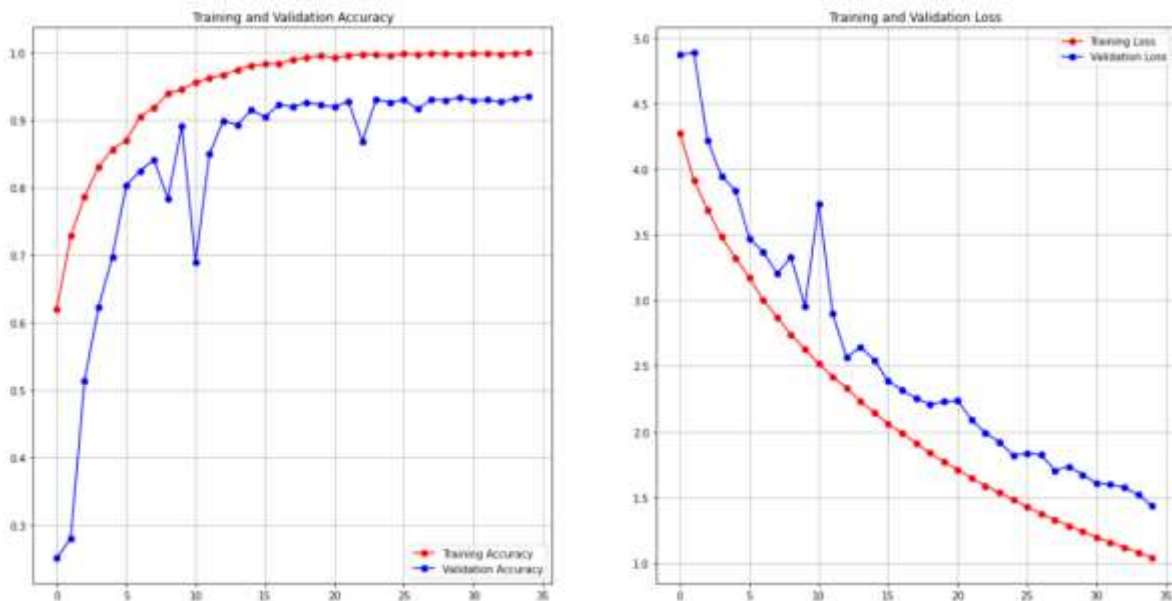


Figure 08: Result of our multi-class classification model

The first observation is that the model manages to train correctly, there is no real inconsistency in training and validation. The accuracy for training tends towards 1.0 and reaches the value of 1.0 from the eighteenth epoch onwards, and the accuracy for validation reaches 0.9338 at the 35th epoch.

The maximum accuracy for validation is 0.9338. Based on these results, the model has a better accuracy.

There is no loss or gradient explosion, and the model is not complicated, which makes learning faster.

The good configuration of the batch size removed the fluctuation in our result.

In summary, the model is able to identify the images well and is able to classify each image into its corresponding type. An increase in the data sets can further improve this model to have more accurate prediction of the outputs.

6. TRANSFER LEARNING

We created our model from scratch, choosing the different parameters and hyper-parameters, improved our model with different techniques to make our algorithm robust. We have 3000 images for the two classes, and our model is able to train correctly, has a better accuracy. However, insufficient data can cause the model to overfit. The reason is that the training data is very small and there are so many common features that can be extracted even if we have made a data augmentation.

The solution is to take an existing model, already pre-trained and trained on much more data, this is the concept of transfer learning.

We took the Inception V3 model [13] to test with our data, this was pre-trained on an ImageNet dataset which contains 1.4 million images in 1000 different classes [14].

Here is the result testing on our data over 35 epochs

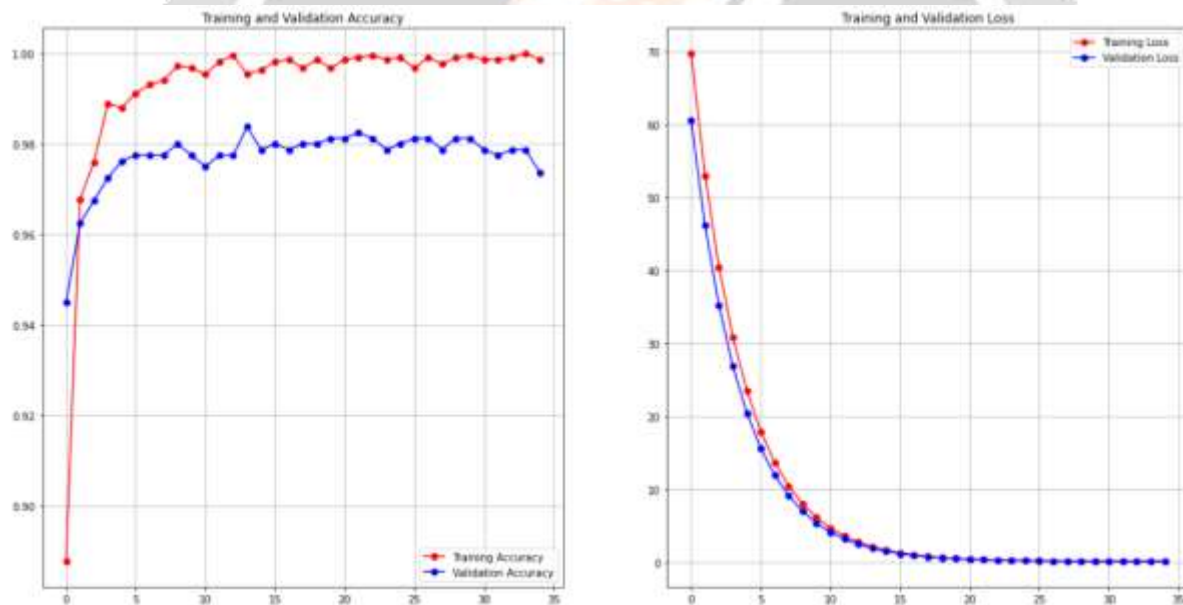


Figure 09: Transfer learning result of the Inception model using our datasets.

The first finding is that using our datasets, the transfer learning technique shows a better result.

The accuracy for training tends towards 1 and reaches the value of 1.0 at the 34th epoch, and the values of the accuracy for validation range from 0.96 to 0.9999.

The accuracy for validation shows a better result, with values around 0.98.

The maximum accuracy for validation is 0.9825. Given these results, the model has a better accuracy.

In summary, the model is able to identify both tumor and normal brain images well and has better accuracy, without fluctuation.

7. CONCLUSION

This work allowed us to create our binary and multi-class classification model from scratch using neural networks with a deep learning technique. Our binary classification model has a better accuracy in terms of training and

validation, these values tend to 1 at a certain time. There is no gradient loss or explosion, and the model is not complicated. Our multi-class classification model has better accuracy at the training and validation level. But the accuracy for validation can be further improved with more data. In deep learning, it is not always possible to generalize certain optimization rules or choices on parameters and hyper-parameters. In this work, we used and elaborated different optimization and regularization techniques such as batch normalization, input normalization, dropout, coefficient regularization, early stopping and data augmentation technique to make our model optimal and robust. We used several combinations to choose the numbers of layers, numbers and sizes of filters. We also used transfer learning with our data, and tested its accuracy. To conclude, our method can achieve a maximum accuracy of 96.88% for binary classification, 93.38% for multi-class classification and 98.37% for binary classification using the transfer learning technique.

8. REFERENCES

- [1]. M. Gurbină, M. Lascu, and D. Lascu, « *Tumor Detection and Classification of MRI Brain Image Using Different Wavelet Transforms and Support Vector Machines,* » 42nd International Conference on Telecommunications and Signal Processing (TSP), 2019, pp. 505–508, Jul 2019.
- [2]. Krisna Nuresa Qodri, Indah Soesanti, Hanung Adi Nugroho, « *Image Analysis for MRI-Based Brain Tumor Classification Using Deep Learning* », Mars 2021.
- [3]. S.K. Chandra, « *Effective Algorithm For Benign Brain Tumor Detection Using Fractional Calculus,* » TENCON 2018 - 2018 IEEE Region 10 Conference, 2018, pp. 2408–2413, Feb 2019.
- [4]. Emrah Irmak, « *Multi-Classification of Brain Tumor MRI Images Using Deep Convolutional Neural Network with Fully Optimized Framework* », Avril 2021.
- [5]. Kevin Zhou, Hayit Greenspan, Dinggang Shen, « *Deep learning for Medical Image Analysis* », Janv 2017.
- [6]. Ali ARI, Davut HANBAY, « *Deep learning based brain tumor classification and detection system* », Sept 2018.
- [7]. Justin Paul, thesis, « *Deep Learning for Brain Tumor Classification* », Mai 2016.
- [8]. Guillaume Saint-Cirgue, « *Formation deep learning complète* », <https://www.youtube.com/watch?v=XUFLq6dKQok>, Machine learnia, 2021.
- [9]. « *Fonction d'activation* », https://fr.wikipedia.org/wiki/Fonction_d%27activation, 2021.
- [10]. Pascal Monasse, Kimia Nadjahi « *Classez et segmentez des données visuelles* », <https://openclassrooms.com/fr/courses/4470531-classez-et-segmentez-des-donnees-visuelles/5082166-quest-ce-quun-reseau-de-neurones-convolutif-ou-cnn>, 2021.
- [11]. Ahmed Hamada, « *Brain Tumor Detection* », <https://www.kaggle.com/ahmedhamada0/brain-tumor-detection>, 2021.
- [12]. Shervine Amidi, Afshine Amidi, « *Réseaux convolutionnels* », <https://stanford.edu/~shervine/l/fr/teaching/cs-230/pense-bete-reseaux-neurones-convolutionnels>, 2021.
- [13]. Christian Szegedy, Vincent Vanhoucke, Serge Ioffe, Jonah Shlens, Zhigang Wojna, « *Rethinking the inception Architecture for computer vision* », Dec 2015.
- [14]. Laurence Moroney, « *Convolutional Neural Networks in TensorFlow* », <https://www.coursera.org/learn/convolutional-neural-networks-tensorflow/home/welcome>, DeepLearning.AI, 2021.

[15]. Andrew Ng, « *Convolutional Neural Networks* », <https://www.coursera.org/learn/convolutional-neural-networks/home/welcome>, DeepLearning.AI, 2021.

