Contents

1 Introduction

Central Nervous System (CNS) tumors are the 10th leading cause of death worldwide, among which brain tumors account for 85% to 90%. The survival rate of the population with a CNS tumor at 5 years is 36% and at 10 years is 31% [1]. Brain tumors can be classified into two macro areas, namely benign and malignant, which are then subdivided in different tumor types. A benign (non-cancerous) brain tumor is a mass of cells that grows in one place of the brain and does not spread to other tissues. Being confined to a specific area it can be removed with surgery, and once removed it usually doesn't come back [2]. The most common type of benign brain tumor are Meningiomas and Pituitary tumors. A meningioma is a tumor arising from the meninges and that may apply pressure on the adjacent brain. The main risk of meningiomas is that they may compromise some brain functionalities, but since they usually do not spread to the surrounding brain tissue, they have a high chance of being removed by surgery. For this reason, the 5-year survival rate is very high, being 97% in patients aged 15 to 39 and 87% for patients aged 40 or older [3]. Pituitary tumors are abnormal growths (adenomas) that develop and remain in the pituitary gland. Most are benign and result in an alteration in the production of hormones that regulate important functions of the body. They make up near 17% of all primary brain tumors and have an average 97% 5-year survival rate [4]. Instead, a malignant brain tumor is a mass of cancerous cells that growth in the brain tissue. It tends to grow more rapidly with respect to benign tumors, and usually spreads from other organs such as lungs, intestine, etc. The most common type of malignant tumor are Gliomas. This class of tumors usually do not spread to other organs but can extend to the surrounding tis-

sues of the brain. For this category the 5-year survival rate is 22% for patients aged between 22 and 49, lowering drastically to 6% for patients aged between 55 and 64 [5]. This statistics heavily depend on the stage at which the brain tumor is diagnosed, hence an early accurate detection and diagnosis is crucial in order to determine the appropriate treatment [6]. Currently, the most common approach to diagnose the different tumor type consists of first examining the Magnetic Resonance Imaging (MRI) of the brain, and secondly performing a biopsy to analyze a portion of the tissue identified as tumorous. This, however, is invasive, since biopsy can have potential negative effects [7], and subject to human error. Therefore, the implementation of machine learning techniques for the task of tumor segmentation and classification from the MRI images could be highly beneficial. It would allow to support doctors with more accurate predictions and without the need for surgery.

Consequently, the aim of the study is to firstly present a Convolutional Neural Network (CNN) architecture for brain tumor classification in four different categories: no tumor, glioma, meningioma, and pituitary tumor. Then introduce an implementation of the U-Net architecture for the segmentation of the brain tumor. These two should be thought as a basis from which more accurate and complex models could be built in order to reach the goal of clinical application. The work is structured as follows: Section 2 is dedicated to an overview of the literature related to brain tumor segmentation and classification. Section 3 explains the theoretical foundation of the Convolutional Neural Networks. In section 4 the proposed CNN for the classification task is presented and the results are discussed, while section 5 illustrates an example of segmentation with the U-Net architecture. Finally, section 6 concludes the study and discusses some future developments.

2 Literature Overview

The application of artificial intelligence and deep learning-based technologies to the field of medical image analysis has thoroughly impacted the work related to disease diagnosis. As a consequence, many researches have been done on the classification and segmentation of brain MRI images using CNN.

In the literature, there are two common approaches to perform brain tumor classification, some researchers designed their own CNN models in order to accomplish the task, while others decided to adopt transfer learning for the same purpose [8]. Abiwinanda et al. [9] designed a simple CNN to classify the three brain tumor classes (Meningioma, Glioma, and Pituitary) achieving a training accuracy of 98.51% and a validation accuracy of 84.19%. Das et al. [10] decided to apply different image processing techniques to the brain MRI images and then classify them using a CNN, achieving 94.39% accuracy with an average recall of 93%. Sultan et al. [11] proposed a Deep Learning model to classify the three brain tumor classes (Meningioma, Glioma and Pituitary) and to differentiate between the three glioma grades (Grade II, Grade III and Grade IV). The network architecture proposed results in an overall accuracy of 96.13% and 98.7%, respectively, for the two studies. Irmak et al. [12] proposed three different CNN models for three different classification tasks. The first CNN model performs brain tumor detection achieving an accuracy of 99.33%. The second CNN model performs classification of the brain tumors into five different classes (Normal, Glioma, Meningioma, Pituitary and Metastatic) with an accuracy of 92.66%. The third CNN model performs classification of the brain tumors into three grades (Grade II, Grade III and Grade IV) with an accuracy of 98.14%. Khan

et al. [13] designed a CNN model and compared its performance with pre-trained ResNet-50, Inception-v3, and VGG-16 models using the transfer learning approach. The researchers' model achieved an accuracy of 100%, while the pre-trained models achieved an accuracy of 89% (ResNet-50), 75% (Inception-V3), and 96% (VGG-16). However, when evaluating the performances, it is important to consider that the research has been conducted on a very small dataset containing only 253 images. This may imply that the network proposed has poor generalization capabilities. Yang et al. [14] proposed an approach for classifying Low-Grade Glioma (LGG) and High-Grade Glioma (HGG) from brain MRI images using two different CNN architectures, GoogLeNet and AlexNet. It consists of first segmenting the tumor image with a rectangular region of interest (ROI), and then evaluate the two models with five-fold cross validation. Both models were trained from scratch and fine-tuned from models pre-trained on the ImageNet dataset. In the latter case, GoogLeNet achieved an accuracy of 94.5% and AlexNet achieved an accuracy of 92.7%. Basheera et al. [15] proposed a similar method for brain tumors classification. First, the tumor is segmented from the brain MRI images and then from the segmented image deep features are extracted and classified into four different classes (Normal, Glioblastoma, Sarcoma and Bronchogenic carcinoma) using a pre-trained CNN called CNN_S. The models for transfer learning that are used in the mentioned literature are pre-trained on datasets containing an enormous number of images such as ImageNet. Such models pre-trained on natural images have shown surprisingly accurate results [8] but a drawback of this approach is that having a fixed input size, the images would need to be adjusted according to the pre-trained model's input size.

3 Convolutional Neural Networks

Suppose we need to solve a classification problem on an image dataset, Artificial Neural Networks (ANNs) could be used to accomplish the task. For instance, a practical example may consist in classifying images belonging to the common machine learning benchmarking dataset MNIST [16].
Notwithstanding, the use of a simple fully connected neural network to extract features from an image presents two significant problems. First of all, ANNs tend to struggle with the computational complexity required to compute image data. For example, to represent an image in RGB format having size 10x10 pixels, three 10x10 matrices are required i.e., a matrix for each channel of the image, which can be represented as a single vector having size 10x10x3, also known as tensor. This would imply having 300 neurons in the input layer of the ANN. The number of neurons tends to increase considerably as the size of an image increases: for example, an image in RGB format having size 128x128 pixels would require 49152 neurons in the first layer. Therefore, to connect all the neurons of the input layer to the ones of the first hidden layer would require $49152 \cdot h1$ weights, where h1 is the number of neurons in the first hidden layer. Second, and most important, is that this type of network is not invariant to translations and local distortions of the inputs [16].

A solution to the problem of image processing was proposed by Yann LeCun [17], who designed a new method for feature extraction: each image is divided into different areas and the most important features will be extracted from them through the use of filters. This led to the birth of Convolutional Neural Networks (CNNs), which are a type

of neural network used mainly for image processing. The term convolutional refers to the presence of convolutional layers within the network, which are layers having convolution operations that replace the product between matrices.

Convolution Operation

Let f ang g be two continuous functions, define the convolution of f and g, written as $f * g$, with the following integral:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau) g(t - \tau)\, d\tau \tag{1}$$

In case f and g are two discrete functions, the convolution of f and g is defined as:

$$(f * g)(t) = \sum_{m=-\infty}^{\infty} f(m) g(t - m) \tag{2}$$

The result of a convolution can be interpreted as a mixing of the two functions obtained by sliding the second function, also called filter, over the first. In a CNN, the convolution will take place between an image we would like to process and a filter. Since an image is a two-dimensional array, or a matrix, the filter associated with it will also be a matrix, also known as a convolution matrix or kernel. The convolution operator, in the case of images, will therefore be equal to:

$$(I * K)(i, j) = \sum_{m} \sum_{n} I(m, n)\, K(i - m, j - n) \tag{3}$$

where I is a two-dimensional image and K is a two-dimensional kernel [18].

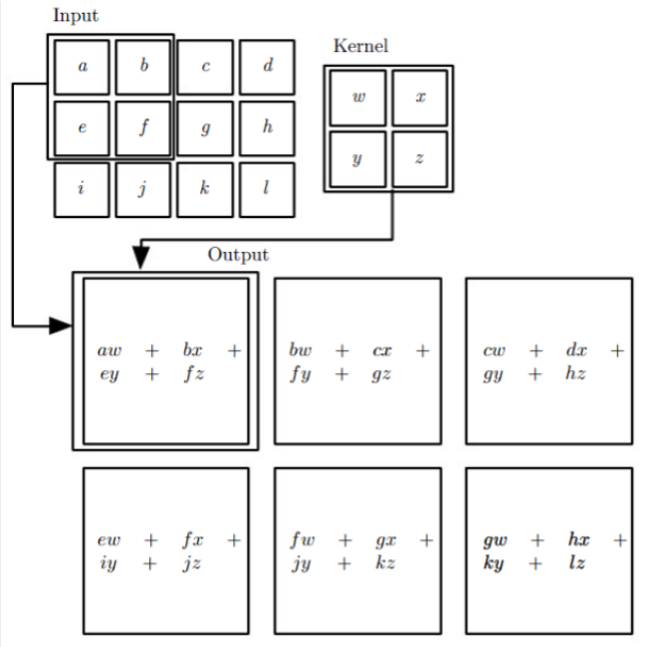

Figure 1: Example of convolution of a 3x4 matrix and a 2x2 matrix [18]

3.1 The CNN Architecture

Three types of layers are mainly used to build a CNN: the aforementioned convolutional layer, the pooling layer and the fully-connected layer.

In order to guarantee a certain degree of invariance with respect to translations and distortions, each convolutional neural network is based on three important concepts: the local receptive field, shared weights and biases, and pooling (an operation to reduce the size of the feature maps).

3.1.1 Convolutional Layer

This layer plays a vital role in the CNN architecture. First of all, it is important to introduce the concept of local receptive field. In CNNs, it is possible to represent each image of size n x m as a matrix having dimension n x m, where the neuron in position (i,j) will be associated to the intensity of the pixel in position (i,j) of the input image. Contrary to ANNs, not every neuron of the input layer will be connected to every neuron of the first hidden layer. Instead, each neuron of the first hidden layer will be connected only to a small region of the input layer, and this region denotes his local receptive field. The result of the convolution of the receptive field and the kernel, as expressed by equation 3, is the value of the neuron inside the hidden layer. As shown in Figure 1, the same filter is used to determine the value of the output neuron, and such filter will be shared to determine the value attributed to each neuron of the hidden layer. Therefore, it will be used a number of weights equal to the size of the filter (in case of Figure 2, the number of weights is 5x5=25) plus the bias, on each neuron within the convolutional layer. Always considering Figure 2, if instead it was constructed as an ANN, neurons would contain 28x28=784 weights each.

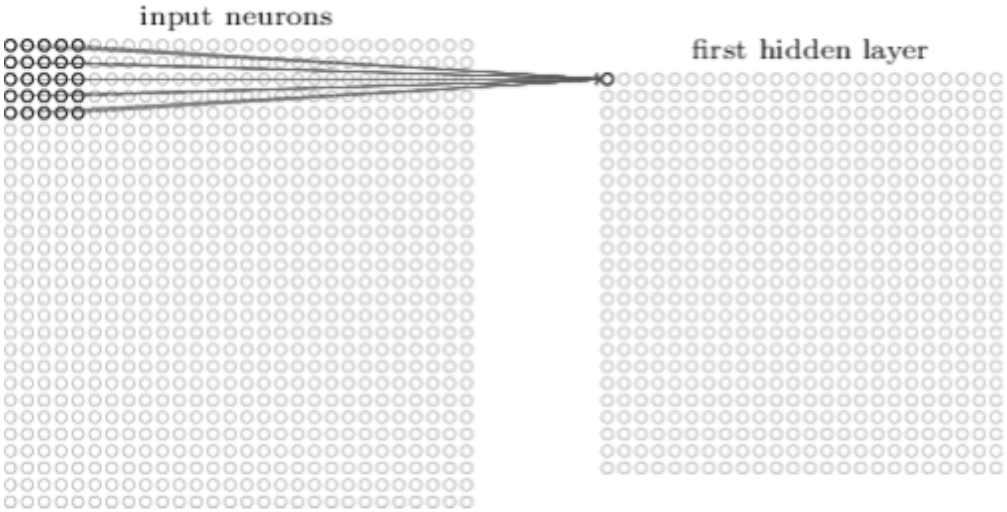

Figure 2: Example of connection between local receptive field and first neuron of the first hidden layer [19]

This results in the concept of shared weights and biases. Each neuron in the hidden layer uses the same weights and biases, which allows the neurons to learn the same feature in different parts of the image. This highlights an important property of convolutional networks: the invariance with respect to translations. For these reasons, the map generated starting from the input layer by means of a filter is also called a feature map. To reduce the complexity of the model, convolutional layers can be optimized through the choice of three hyperparameters, namely the depth, the stride and setting zero-padding [20]:

1. Depth: this is the number of filters used. Each filter will be used to learn a specific feature of the input image;

2. Stride: this is the step with which we move the filter around the spatial dimensionality of the input. Setting the stride to 1 will result in heavily overlapped receptive fields, while higher values

will result in less overlapping and output of lower spatial dimension;

3. Zero-padding: this is a technique consisting of padding the border of the input with zeros to have control over the dimensionality of the output volumes.

These three hyperparameters alter the spatial dimensionality of the convolutional layers output. Then, the dimension of the output volumes is given by the following formula:

$$\frac{(I - R) + 2Z}{S + 1} \tag{4}$$

Where I is the input volume size, R is the receptive field size, Z is the amount of zero-padding set, and S is the stride [20]. The result from previous equation 4 should be equal to a round integer. If this is not the case, then the stride has been wrongly decided and the neurons will be unable to fit precisely across the given input.
Once the feature map is obtained, the ReLU activation function is applied. Its functionality will be to set all nodes with negative values equal to 0 and speed up the training process. Note that usually a CNN has multiple convolutional layers, where the first convolutional layers extract low-level characteristics of the image (such as angles, and lines) and the latest convolutional layers extract complex features of the image, such as objects or irregular shapes.

3.1.2 Pooling Layer

Pooling layers aim to gradually reduce the size of the representation. The basic concept of pooling is the following: each pixel matrix will be divided into different subgroups and, for each of them, they will be replaced by their summary statistics to obtain a matrix with reduced

dimensions. The pooling operation is thus used to scale the dimensionality of each activation map, leading to a reduction in the length and width of the matrix, but unchanged depth. Generally speaking, two different pooling techniques are used: max pooling and average pooling.

Max Pooling

This is the most used pooling technique. It consists of sliding a kernel (usually of size 2x2) along the spatial dimensions of the input with a stride of 2, and for each subset of pixels, it selects only the pixel with the highest value. This technique is illustrated in Figure 3.

Average Pooling

Unlike max pooling, dimensionality reduction will not occur by selecting the maximum value present in each subset, but by averaging all the values present within the block. This technique is illustrated in Figure 3.

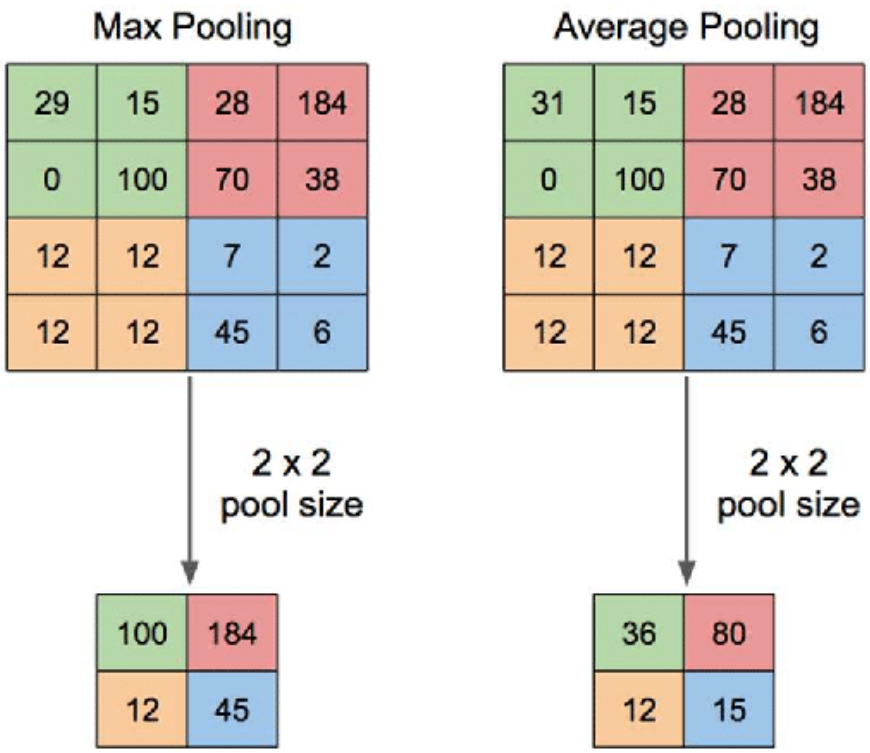

Figure 3: Example of max and average pooling with kernel size 2x2 and stride 2 on a 4x4 matrix [21]

3.1.3 Fully Connected Layer

Once all the operations involving the convolutional and pooling layers have been terminated, an output of dimension WxHxD will be obtained. Through an operation called flattening, it is possible to represent the output having dimensions WxHxD as a vector of dimension W·H·D. At this point it is possible to apply fully-connected layers.
The structure of all fully-connected layers will be the same as in traditional forms of ANNs. The last fully-connected layer will have a number of nodes equal to the total number of classes defined for the classification problem.

3.2 Data Augmentation

The best way to increase the model's ability to adapt to new data is to increase the size of the dataset. Unfortunately, this is not always possible. A solution to this problem is performing data augmentation, which consists in manipulating the data already present by adding particular effects in order to provide different contexts to the network.
The most common data augmentation practices consist in increasing data by modifying the geometry of the images, such as rotation, translation, reflection, or the alteration of contrast and brightness. Since the addition of new significant details to the images can lead the model to better learn certain features, it is very important to identify a data augmentation pipeline that is able to adapt to the training data and maintain excellent generalization capabilities.

4 Brain Tumor Classification

4.1 Data Collection and Exploratory Analysis

The image dataset used in this research contains 3264 contrast-enhanced brain MRI images [22]. There are four types of images: no tumor (500 images), glioma (926 images), meningioma (937 images), and pituitary tumor (901 images). All images have been acquired in three different planes: axial, coronal, and sagittal planes. Figure 4 provides some examples of the different types of tumors for each plane.

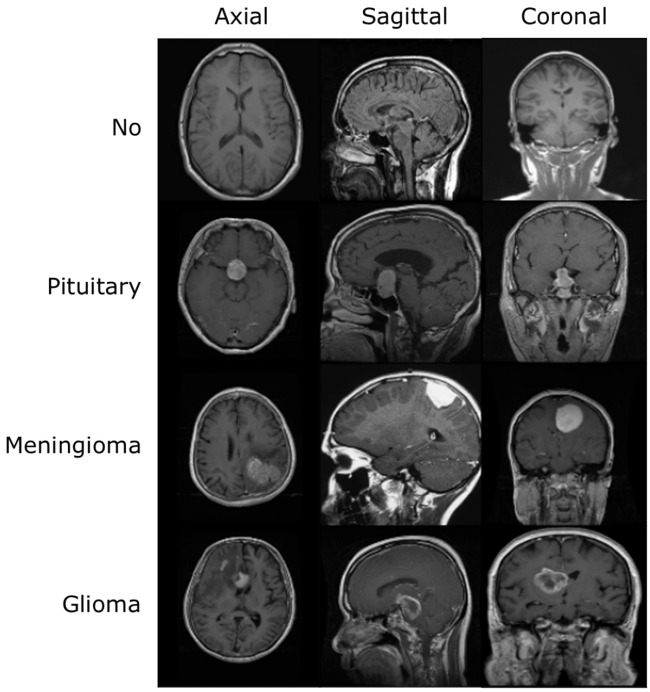

Figure 4: examples of the different types of tumors for each plane.

The MRI images from the database have been normalized, resized from 256x256 to 128×128 pixels, and converted to grayscale color channel in order to improve time and computational efficiency. To

augment the dataset and guarantee invariance to translations and distortions, each image has been transformed in two ways. The first transformation consisted of flipping the image horizontally, while the second transformation consisted in rotating the image by 30 degrees. Given the number of images, the dataset has been split into 3 separate segments for training, validation and testing having the ratio of 80:10:10 as shown in Table 1. In order to keep the portion of classes constant across the three segments, stratified sampling has been applied.

Classes	Each Group	Total	Training set	Validation Set	Test set
No Tumor	500	3264	2611	326	327
Pituitary	901				
Meningioma	937				
Glioma	926				

Table 1: Dataset split

4.2 Model Training and Evaluation

Model Training

Tumor classification has been performed using a simple CNN model which received as input the augmented MRI image data of size 128 × 128, having grayscale color channel. The model consists of five convolutional layers, each followed by the rectified linear unit (ReLU) activation layer, the max-pooling layer with a 2x2 filter, and the dropout layer. The first convolutional layer consists of 64 filters with size 5x5, the following ones consist of 128 filters with size 3x3, and the last one consists of 256 filters with size 2x2. This allows to combine the small patterns detected in the first convolutional layer as the num-

ber of filters increases and finds bigger patterns. Ultimately, a fully connected dense layer of 1024 neurons has been applied, along with the softmax output layer, to calculate the probability score for each class and classify the final decision labels as detecting the presence of a glioma, pituitary, meningioma, or no tumor. The proposed CNN architecture is shown in Table 2.

The network was trained using the Cross Entropy loss function, the adaptive moment estimation (Adam) optimizer, and a batch size equal to 32. The reasons for this configuration are that the Cross Entropy loss function allows for better generalization and faster training [23], and Adam is the best performing in practice with respect to other stochastic optimization methods [24]. Finally, the early-stop condition, which determines when the process of network training will stop, corresponds to three epochs.

Model Evaluation

A good way to see the features learned and understand if the model is not overly complex is to display the activations of the convolutional layers of the CNN. The activations of the first and the last convolutional layers are shown in Figure 5 a and b, respectively. The first convolutional layer is used to learn low level features like color, edges or angles. At this step the information present in the initial image is almost fully retained, although some filters are not activated and remained blank. The following convolutional layers develop their features by combining those learned by the previous layers, thus the activations become increasingly abstract and less visually interpretable. These higher presentations contain more information related to the class of the image and are able to learn significant features like objects.

Layer No.	Layer Name	Layer Properties
1	Image Input	128x128x1 images
2	Convolutional	64 5×5×1 convolutions with padding ‘same’
3	Rectified Linear Unit	Rectified Linear Unit
4	Max Pooling	2×2 max pooling with stride [2 2]
5	Dropout	20% dropout
6	Convolutional	128 3×3×1 convolutions with padding ‘same’
7	Rectified Linear Unit	Rectified Linear Unit
8	Max Pooling	2×2 max pooling with stride [2 2]
9	Dropout	20% dropout
10	Convolutional	128 3×3×1 convolutions with padding ‘same’
11	Rectified Linear Unit	Rectified Linear Unit
12	Max Pooling	2×2 max pooling with stride [2 2]
13	Dropout	20% dropout
14	Convolutional	128 3×3×1 convolutions with padding ‘same’
15	Rectified Linear Unit	Rectified Linear Unit
16	Max Pooling	2×2 max pooling with stride [2 2]
17	Dropout	20% dropout
18	Convolutional	256 2×2×1 convolutions with padding ‘same’
19	Rectified Linear Unit	Rectified Linear Unit
20	Max Pooling	2×2 max pooling with stride [2 2]
21	Dropout	20% dropout
22	Fully Connected	1024 hidden neurons in fully connected layer
23	Rectified Linear Unit	Rectified Linear Unit
24	Fully Connected	4 hidden neurons in fully connected layer
25	Softmax	Softmax
26	Classification Output	4 output classes

Table 2: Proposed CNN Architecture

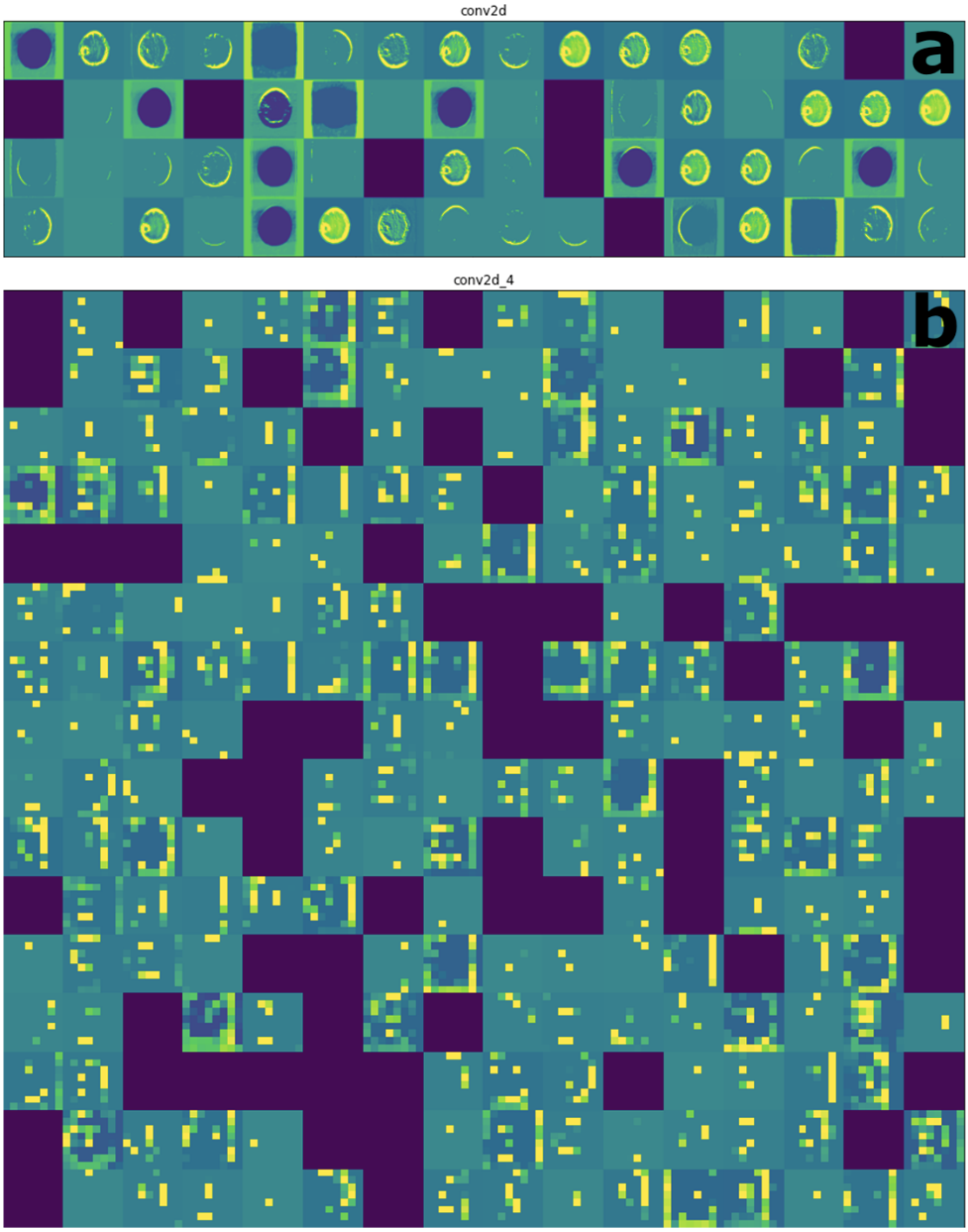

Figure 5: (**a**) The first and (**b**) the last (fifth) convolutional layer activations. The images are in grayscale.

The proposed model showed 94.51% accuracy on training data, 92.94% accuracy on validation data, and 94.50% accuracy on the test data. Figure 6 displays the accuracy and loss plot of the training and validation phase.

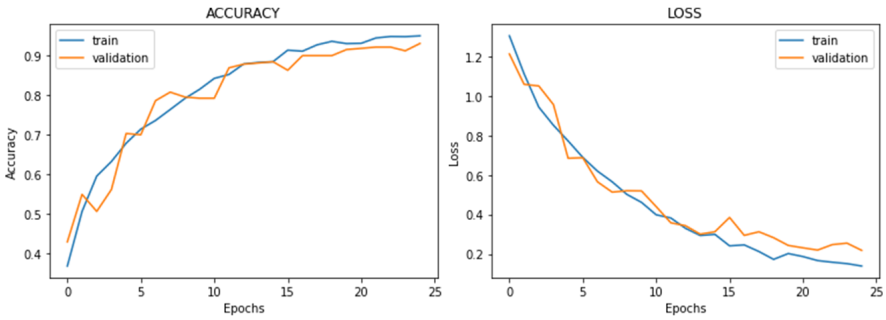

Figure 6: On the left, the plot of accuracy against the number of epochs. On the right, the plot of loss against the number of epochs.

Table 3 displays the model accuracy, precision, recall, and F-1 score. Note that in this context, the interest should be in minimizing the number of false negatives. This is equivalent to maximize the model recall, given that recall is calculated as,

$$Recall = \frac{TruePositive(TP)}{TruePositive(TP) + FalseNegative(FN)}$$

Classes	Accuracy	Precision	Recall	F-1 Score	Total
No Tumor	0.99	0.98	0.96	0.97	52
Pituitary	0.99	0.99	0.98	0.98	96
Meningioma	0.95	0.89	0.96	0.92	90
Glioma	0.95	0.94	0.89	0.91	89

Table 3: Accuracy metrics

Figure 7 shows the confusion matrix for testing data and Figure 8 shows examples of classified images from testing data.

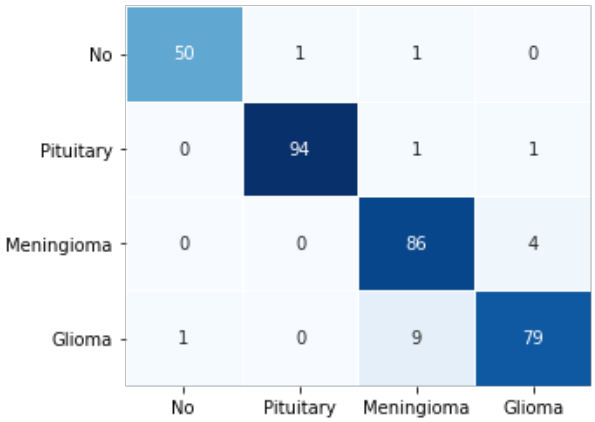

Figure 7: Confusion matrix.

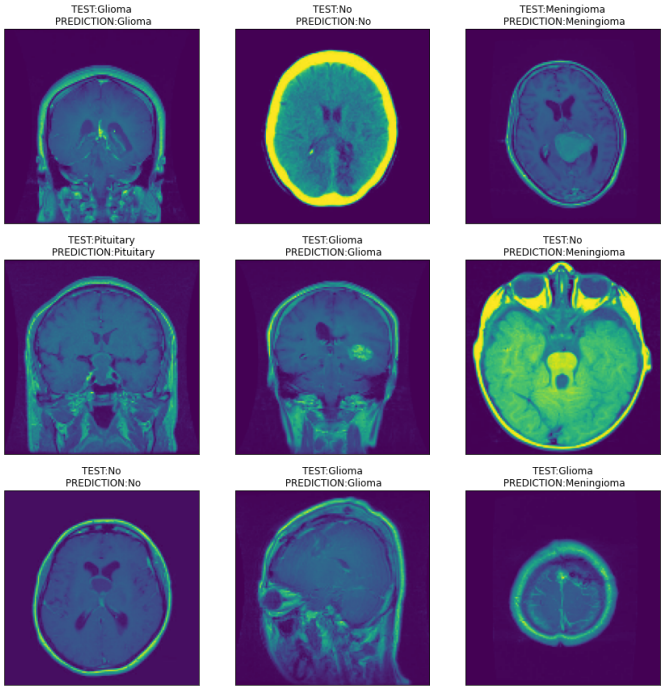

Figure 8: Some examples of the classification result. The images are in grayscale.

From these results we can see that the model has very good generalization capabilities, but it is more prone to confuse meningiomas with

gliomas, and vice versa. A way to avoid this could be to increase the number of images of meningioma and glioma tumors in training set so that the model could extract more features useful to classify them correctly. On the whole, these results proof the ability of the proposed CNN model to correctly classify different brain tumor types.

5 Brain Tumor Segmentation

5.1 U-Net Architecture

CNNs are usually implemented for classification tasks, where for each image received as input the network should output a single class label. However, many visual tasks in biomedical image processing require the desired output to include localization, meaning that a class label is supposed to be assigned to each pixel of the input image. Furthermore, the CNNs used for image classification require a reasonably high number of training images to achieve a good level of accuracy, but this is rather difficult [25]. For these reasons, Ronneberger et al. [25] developed the U-Net for biomedical image segmentation. U-Net has been built upon the architecture of a Fully Convolutional Network, which can accept any image size since it only contains convolutional layers and not dense layers. This architecture has been modified in order to work with few training images and yield more precise segmentations with respect to the previous state-of-the-art models [25]. As can be seen from Figure 9, the model architecture consists of two paths: first a contracting path (left side), also referred to as encoder, and then the symmetric expansive path (right side), also referred to as decoder. The contracting path is used to capture the image context and follows the architecture of a regular convolutional network where convolutional (with ReLU activation function) and max pooling layers are stacked together. Since at each downsampling step the number of feature channels is doubled, the size of the image is shrinked while the depth increased. For instance, if the encoder receives a 128x128x1 image as input, it will output an 8x8x256 image. Then the expansive path is used to enable precise localization by gradually upsampling the feature map. The decoder consists of transposed

convolutions and regular convolutions, where each step involves a 2x2 transposed convolution (or "up-convolution") to halve the number of feature channels, skip connections to concatenate the output of the transposed convolution layers with the feature maps from the encoder at the same level, and two consecutive 3x3 convolutions (with ReLU activation function) so that the model can construct a more precise output. Since at each upsampling step the number of feature channels is halved, the size of the image is increased while the depth reduced. For instance, if the decoder receives an 8x8x256 image as input, it will output a 128x128x1 image.

The contracting path followed by the expansive path gives the architecture a symmetric U-shape, from which derives the name U-Net.

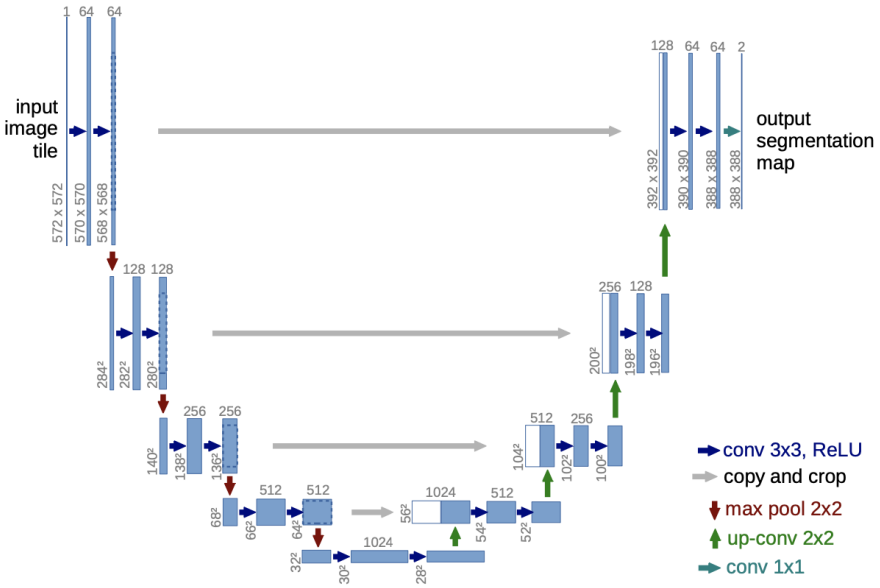

Figure 9: U-net architecture. Each blue box corresponds to a multi-channel feature map and white boxes represent copied feature maps [25].

5.2 Model Training and Evaluation

Model Training

Tumor segmentation has been performed using the original U-Net architecture with a different input size of 256x256 but same number of convolutional layers, transposed layers, and filters (64, 128, 256, 512, 1024). Then, for the output layer a convolutional layer of size 1x1 with sigmoid activation function is used for binary segmentation. This results in a total of 31,043,521 parameters (out of which 5,888 are non-trainable). The model is compiled using the adaptive moment estimation (Adam) optimizer, the binary Cross Entropy loss function (since there are only two classes of pixels: tumor and no tumor), and a batch size equal to 16. Finally, it has been trained for 150 epochs with a standard learning rate of 1e-4 and early stopping to 10 epochs. The metrics used for the evaluation are accuracy, Dice similarity score and IoU (Intersection over Union) score. Dice and IoU scores are computed as follows:

$$Dice = \frac{2 \cdot TruePositive(TP)}{2 \cdot TruePositive(TP) + FalsePositive(FP) + FalseNegative(FN)}$$

$$IoU = \frac{TruePositive(TP)}{TruePositive(TP) + FalsePositive(FP) + FalseNegative(FN)}$$

Model Evaluation

The image database used in this section contains only 109 contrast-enhanced brain MRI images, which I manually annotated using the tool at `https://www.labelbox.com/`. Given the size of the dataset, it has been split into 3 separate segments for training (80 images), validation (20 images) and testing (9 images).

The proposed model showed 99.50% and 97.50% (binary) accuracy on training and validation data respectively. The training IoU and

Dice scores are 77.36% and 87.18%, while the validation IoU and Dice scores are 47.68% and 64.29%, respectively. On test data the model achieved 97.66% (binary) accuracy, 45.29% IoU and 62.16% Dice score. These results are presented in Table 4, while Figure 10 displays the accuracy and loss plot of the training and validation phase. Obviously, accuracy is very high due to the fact that the masked area is small, hence a great number of pixels are of the same class (no tumor). This implies that accuracy would be high even if the network learned to classify each pixel as no tumor, so it is not a very insightful metric.

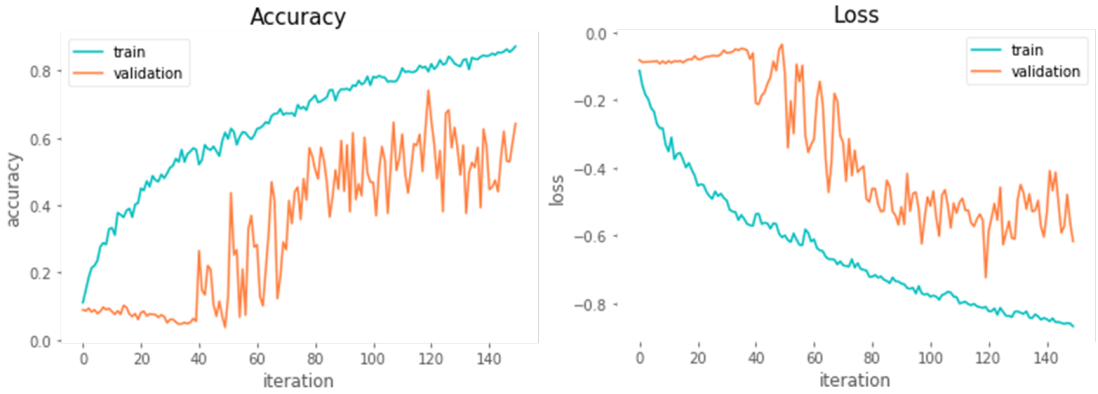

Figure 10: On the left the plot of accuracy against the number of epochs, while on the right the plot of loss against the number of epochs.

	Accuracy	Loss	IoU	Dice
Train	99.50%	-0.87	77.36%	87.18%
Validation	97.50%	-0.61	47.68%	64.29%
Test	97.66%	-0.62	45.29%	62.16%

Table 4: Accuracy metrics

Figure 11 displays some examples of the predicted mask (green) versus the ground truth (red) over the test data.

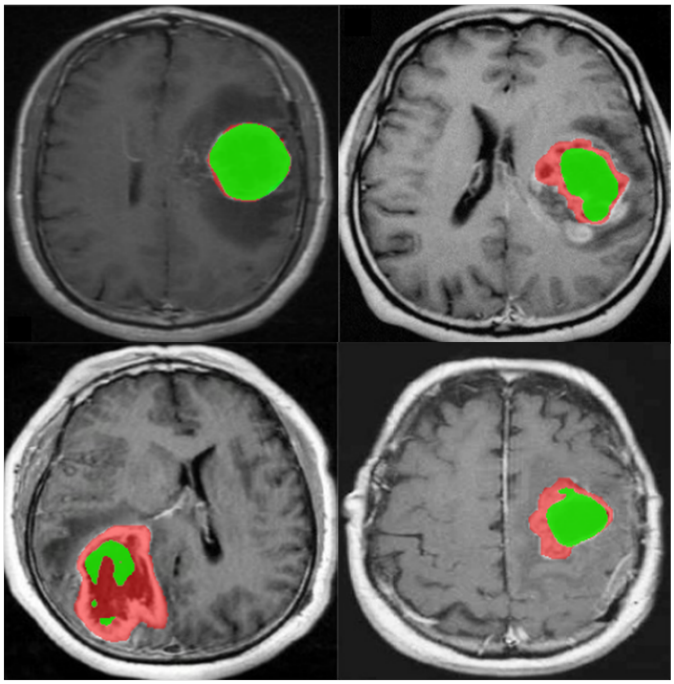

Figure 11: Examples of predicted mask. The green are the masks predicted while the red are the ground truth.

From these results it is possible to conclude that the model has discrete generalization capabilities, but there could also be a lot of scope to tune the hyperparameters (like number of epochs, model depth, image input size and color scale, etc.) and further improve the model performance. Due to lack of availability of annotated data, the dataset used contained very few images, hence the model prediction precision could be greatly improved by adding new annotated images. Moreover, there could also be the possibility to apply transfer learning using a backbone (e.g ResNeXt50) trained on image datasets like ImageNet to reduce the computational power required.

6 Conclusions

This study illustrates an approach for brain tumor classification and segmentation. First, an accurate and efficient methodology for brain tumor classification is provided. The classification was performed with a structurally simple CNN taking as input the whole MRI image, allowing to perform little preprocessing. Second, an example of brain tumor segmentation is provided. This task has been achieved through the implementation of the classical U-Net architecture on a relatively small set of images. The CNN used in the classification problem is simpler than most of the pre-trained networks, and it is possible to run it on devices with modest computational power (6s per epoch). Moreover, the results obtained with the proposed CNN model and a comparison with state-of-the-art methods show that the model is effective in accurately classifying brain tumor types. As regards image segmentation, for more sophisticated and exact results, this neural network would require a larger amount of training data, and to improve further the model performance, could be considered more comprehensive hyperparameter tuning and preprocessing techniques. Therefore, these two models could be perfected in order to be integrated into existing image analysis pipelines, or even more, used to support surgeons in classifying and accurately locating brain tumors. The main limitation in this sense, is the fact that deep learning methods are often treated as "black boxes". Given the difficult predictability of the results, and the importance of liability in medicine, it may not be enough to have accurate prediction systems. Some methods developed to face this issue could be guided back-propagation [26] or deconvolution networks [27]. Accordingly, this would allow for an extensive use of deep learning methods for early disease diagnosis.

References

[1] Cancer.Net. *Brain Tumor: Statistics*. <https://www.cancer.net/cancer-types/brain-tumor/statistics>[last accesed:10/06/2021].

[2] NHS.uk. *Benign Brain Tumor*. <https://www.nhs.uk/conditions/benign-brain-tumour/>[last accesed:10/06/2021].

[3] Cancer.Net. *Meningioma: Statistics*. <https://www.cancer.net/cancer-types/meningioma/statistics>[last accesed:10/06/2021].

[4] Cancer.Net. *Pituitary Gland Tumor: Statistics*. <https://www.cancer.net/cancer-types/pituitary-gland-tumor/statistics>[last accesed:10/06/2021].

[5] Cancer.org. *Brain Tumor Survival Rate*. <https://www.cancer.org/cancer/brain-spinal-cord-tumors-adults/detection-diagnosis-staging/survival-rates.html>[last accesed:10/06/2021].

[6] WHO. *Cancer Early Diagnosis*. <https://www.who.int/activities/promoting-cancer-early-diagnosis>[last accesed:10/06/2021].

[7] Maria Alieva et al. "Preventing inflammation inhibits biopsy-mediated changes in tumor cell behavior". In: *Scientific Reports* 7 (Dec. 2017). DOI: 10.1038/s41598-017-07660-4.

[8] Geert Litjens et al. "A survey on deep learning in medical image analysis". In: *Medical Image Analysis* 42 (2017), pp. 60–88. DOI: 10.1016/j.media.2017.07.005.

[9] Nyoman Abiwinanda et al. "Brain Tumor Classification Using Convolutional Neural Network". In: *IFMBE Proceedings* 68/1 (May 2018), pp. 183–189. DOI: 10.1007/978-981-10-9035-6_33.

[10] Sunanda Das, O. F. M. Riaz Rahman Aranya, and Nishat Nayla Labiba. "Brain Tumor Classification Using Convolutional Neural Network". In: *2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)* (2019), pp. 1–5. DOI: 10.1109/ICASERT.2019.8934603.

[11] Hossam H. Sultan, Nancy M. Salem, and Walid Al-Atabany. "Multi-Classification of Brain Tumor Images Using Deep Neural Network". In: *IEEE Access* 7 (2019), pp. 69215–69225. DOI: 10.1109/ACCESS.2019.2919122.

[12] Emrah Irmak. "Multi-Classification of Brain Tumor MRI Images Using Deep Convolutional Neural Network with Fully Optimized Framework". In: *Iranian Journal of Science and Technology, Transactions of Electrical Engineering* (Apr. 2021), pp. 1–22. DOI: 10.1007/s40998-021-00426-9.

[13] Hassan Ali Khan et al. "Brain Tumor Classification in MRI Image Using Convolutional Neural Network". In: *Mathematical Biosciences and Engineering* 17(5) (2020), pp. 6203–6216. DOI: 10.3934/mbe.2020328.

[14] Yang Yang et al. "Glioma Grading on Conventional MR Images: A Deep Learning Study With Transfer Learning". In: *Frontiers in Neuroscience* 12 (Nov. 2018), p. 804. DOI: 10.3389/fnins.2018.00804.

[15] Shaik Basheera and M. Ram. "Classification of Brain Tumors Using Deep Features Extracted Using CNN". In: *Journal of Physics: Conference Series* 1172 (Mar. 2019), pp. 12–16. DOI: 10.1088/1742-6596/1172/1/012016.

[16] Yann Lecun et al. "Gradient-Based Learning Applied to Document Recognition". In: *Proceedings of the IEEE* 86 (Dec. 1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[17] Yann LeCun et al. "Backpropagation Applied to Handwritten Zip Code Recognition". In: *Neural Computation* 1.4 (1989), pp. 541–551. DOI: 10.1162/neco.1989.1.4.541.

[18] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. <http://www.deeplearningbook.org>. MIT Press, 2016.

[19] Michael A. Nielsen. *Neural Networks and Deep Learning*. <http://neuralnetworksanddeeplearning.com>. 2018.

[20] Keiron O'Shea and Ryan Nash. "An Introduction to Convolutional Neural Networks". In: *CoRR* abs/1511.08458 (Nov. 2015). URL: http://arxiv.org/abs/1511.08458.

[21] Terry's Nail. *Application of Transfer Learning Using Convolutional Neural Network Method for Early Detection - Scientific Figure on ResearchGate. 2021*. <https://www.researchgate.net/figure/Illustration-of-Max-Pooling-and-Average-Pooling-Figure-2-above-shows-an-example-of-max_fig2_333593451>[last accesed:10/06/2021].

[22] Bhuvaji Sartaj et al. *"Brain Tumor Classification (MRI)." Kaggle. 2020*. <https://www.kaggle.com/sartajbhuvaji/brain-tumor-classification-mri>[last accesed:31/05/2021].

[23] Shie Mannor, Dori Peleg, and Reuven Rubinstein. "The cross entropy method for classification". In: (Jan. 2005), pp. 561–568. DOI: 10.1145/1102351.1102422.

[24] Diederik Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *International Conference on Learning Representations* (Dec. 2014).

[25] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *LNCS* 9351 (Oct. 2015), pp. 234–241. DOI: 10.1007/978-3-319-24574-4_28.

[26] Jost Springenberg et al. "Striving for Simplicity: The All Convolutional Net". In: *CoRR* (Dec. 2014).

[27] Matthew Zeiler and Rob Fergus. "Visualizing and Understanding Convolutional Neural Networks". In: *ECCV 2014, Part I, LNCS 8689* 8689 (Nov. 2013).