Brain Tumor Images Classification Using MRI Dataset And CNN

Riccardo Rossi

***Abstract*—Medical images classification plays an essential role in clinical treatment, but the traditional method has reached its ceiling on performance. Moreover, by using them, much time and effort need to be spent on extracting and selecting classification features. The deep neural network is a working method that in the last ten years has proven its potential for different classification tasks. Notably, the convolutional neural network dominates with the best results on varying image classification tasks. Medical images datasets are hard to collect because it needs a lot of professional expertise to label them, therefore, this paper researches how to apply the convolutional neural network (CNN) based algorithm and a VGG16 transfer learning methods on a human brain MRI images dataset to speed up this process. For this reason, these two models was developed with the aim of classifying the following three types of tumors: meningioma tumor, pituitary tumor, glioma tumor, and the absence of the tumor. The performance of the model was evaluated through graphs and metrics and the latter was optimized in the best possible way, but due to the excessive diversity of the test data from the training data, the results achieved present an overall accuracy of 41% and 40% respectively.**

I. Introduction

Brain cancer represents a leading cause of death worldwide, with a significant impact on public health. Global statistics indicate that approximately 296,851 new cases of brain tumors were diagnosed in 2020, with an estimated 241,037 deaths related to this disease. These data highlight the relevance of brain cancer as a global health problem. Distinguishing between different types of brain tumors is critical for accurate diagnosis and targeted treatment. Meningioma, which develops in the meninges, may usually be benign, but its growth can generate harmful compression on surrounding tissues. Pituitary adenoma, located in the pituitary gland, can affect hormone production and cause various disorders. Gliomas, however, are more widespread tumors and can affect different areas of the brain, posing a significant challenge for treatment. For detecting brain tumors, magnetic resonance imaging (MRI) has proven to be a highly effective method. MRI provides detailed anatomical details and allows clear visualization of brain structures, facilitating early detection of tumors. Its ability to distinguish between healthy and abnormal tissue, along with the ability to obtain images in different planes, makes MRI a crucial diagnostic tool in the fight against brain cancer.

The advent of deep learning and deep neural networks, in particular Convolutional Neural Networks (CNN), has revolutionized the field of Image Classification. CNNs were designed to automatically learn features directly from image data, eliminating the need for manual feature extraction. This allowed us to obtain exceptional results in terms of accuracy and performance. CNNs use convolutional filters to capture patterns and features from images at different scales and levels of abstraction. The use of pooling layers and fully connected layers allows you to build highly adaptable and scalable models. CNNs have been able to overcome the limitations of traditional methods, achieving notable successes in object recognition, image classification and computer vision applications in general. To understand what we are talking about, the following image shows the substantial difference between the traditional machine learning approach and that of deep learning [1].

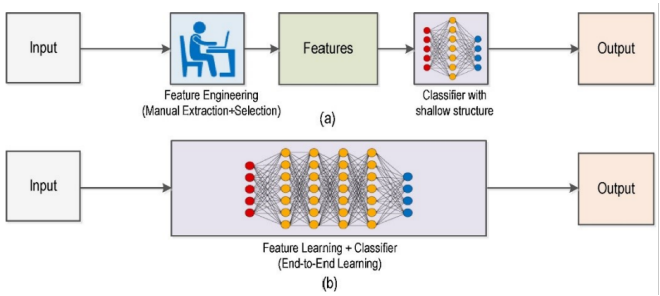

Fig. 1: Comparison between the Deep Learning approach and the traditional Machine learning approach.

Scientists began to provide good results in image classification tasks starting in 1998, where LeNet-5 [2], the first CNN architecture capable of recognizing handwritten numeric characters taken from the MNIST dataset was developed. In 2012 there was another significant breakthrough in this field thanks to AlexNet [3], this convolutional neural network (CNN) was presented in 2012 during the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) competition, where it achieved considerable success, significantly winning compared to other participants and setting new standards in the field of image recognition. We continue with GoogLeNet [4] and VGGnet [5] , ranked first and second respectively in the 2014 ILSVRC, these architectures were able to lower the loss function even further compared to previous architectures. With the latter we began to delve into the world of deep learning, as they presented 22 and 19 layers respectively, more than double the layers of the previous networks. We could go on and on with ResNet [6], Xception [7], MobileNet [8], DenseNet [9], ViT [10], SWIN Tranformer [11].

The paper is organized as follows: Section II will present some of the current works in the scientific world for the task of image classification. Section III describes the development environment, the network architecture used, and the evaluation metrics used. Section IV will present network training graphs

and results. Finally we conclude in section V.

II. Related Work

There are several methods for making classifications of MRI images nowadays, the ones that will be addressed in this paper are respectively those that include the use of custom CNN architectures and the one that makes use of architectures pre-trained through transfer learning. However, very positive results have been found through the use of supervised learning algorithms that process categorical and numerical features extracted from MRI images, as can be seen in the paper [12], where models such as DT, SVM, KNN, NN have presented respectively, after a 30-fold cross validation an overall accuracy of 96.2%, 94.9%, 94.8% and 83.5%.

A. Pre-trained Architectures

Transfer learning is a technique in machine learning in which a model pre-trained on one dataset is used as a starting point to train a new model on a similar or related dataset. The main idea is that the pre-trained model has already learned useful representations of the data and can be transferred to a new problem, significantly speeding up the training process and requiring less data than training from scratch. There are three types of transfer learning:

CNN as a fixed feature extractor: What if my task is different from the pre-trained model's task? Instead of discarding the entire model, I discard only the final part of the network, i.e. the FC layer where the classification takes place, I replace them with my layers with the right output (imagenet envisaged a classification of 1000 categories, maybe I'm interested in 10) and I will only retrain these last layers. Why does this make sense? The purpose of the first layers is to do representation learning, not to do classification. This is the case of paper [13], where the dataset used was too small to be able to fine tune and the risk of overfitting was too high. Through this methodology they managed to achieve important results such as 89% accuracy on ResNet-50, 96% on VGG16 and 75% on InceptionV3. The same approach was also adopted in [14], where an accuracy of 91.8% was achieved on VGG16, 94.83% on Alexnet and 98.88% with ResNet-50.

Fine tuning: The FC layers at the bottom of the network are always discarded, but in addition I don't even like the features on the previous layers too well, because perhaps they are too different for my task, they work quite well for me but I would like to refine them. Refinement means attacking new FC layers, but also propagating the gradient backwards (either across the entire network or only on some layers) and not freezing the gradient as in transfer learning. Doing it only on high-level layers is a better choice because the features extracted in the first layers are very generic (edge, blob, corner) and are always useful to maintain, while the deeper layers are more specific to the dataset on which it was trained the network, so it is better to replace them.

B. Custom Architectures

Preferring a custom convolutional neural network (CNN) over a pre-trained one depends on the specific needs of the problem and available resources. Training a custom CNN allows you to adapt the model to the specific data of the problem, which can lead to better performance in certain situations. Additionally, custom training allows for more flexibility in model design and control of the training process, like in [15], where through the use of a custom network composed of 15 convolutional laters, it is possible to achieve surprising results with an accuracy of 99% on all three classes: pituitary, meningioma and glioma.

III. Network architecture and development environment

The development of this experiment was done through Google Collaboratory, using the T4 GPU virtual machine. The NVIDIA Tesla T4 GPU is designed for Artificial Intelligence (AI) inference and offers several key features. Introduces revolutionary Turing Tensor Core technology with multi-precision computing to handle diverse workloads. It has 16GB of GDDR6 memory and offers performance of up to 8.1 TFLOPS in FP32 and 130 TOPS in INT8. It is suitable for a wide range of cloud workloads, including high-performance computing, deep learning training and inference, machine learning, and data analytics The experiment was also conducted locally using a configuration as follows: Intel(R) Core(TM) i5-7600 CPU @ 3.50GHz, 16.0 GB DDR3 RAM, NVIDIA GeForce GTX 1060 6GB. All the code has been optimized to work completely on GPU, achieving training speeds under five minutes. The elapsed time of only the training phase was measured and since it was almost identical in both environments, only the times for Colab were reported, as it is the environment in which the project was presented.

A. Custom CNN Network Architecture

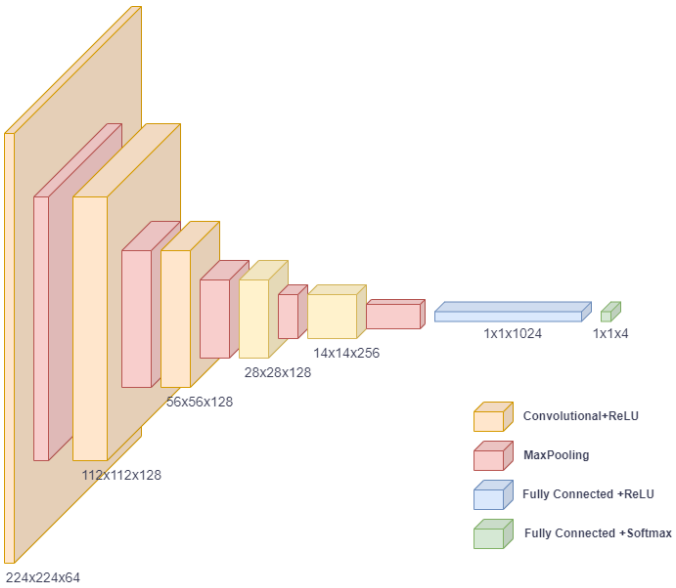

Fig. 2: CNN architecture.

The architecture that was used to carry out this classification task is composed by the following scheme: ($Conv \rightarrow$

$MaxPool \rightarrow DropOut)\times 5 \rightarrow FC \rightarrow DropOut \rightarrow FC$. This is one of the most common patterns for convolutional neural network implementations. The choice to stop at five convolutional layers was made on the basis of the fact that there are only four classification classes, therefore it made no sense to complicate the network even further and risk falling into overfitting. In CNN theory, best practice is to place a 3x3 filter in the first layer, a 5x5 in the second and then all the rest of the 3x3 filters, this is because it is demonstrable that networks with small filter stacks work better than those with large. This is because the receptive field of a network after three layers of 3x3 filters is the same as the first layer of a network with 7x7 filters, plus the network with 3x3 filters has many fewer superfluous parameters to learn and introduces much more non-linearity [16]. Nonetheless, sometimes it is useful to make a compromise so as not to immediately exhaust the GPU memory and you prefer to use a larger filter, for example 5x5 or 7x7 only in the first layer and then use smaller ones in all the others. The choice of MaxPool was made because it is essential to reduce the size of the feature maps, to maintain the most important features and finally to give invariance to the translations and therefore to give robustness. The dropout layer is put in place to reduce overfitting of the network by 'turning off' a percentage of neurons in each layer [17]. Finally the network ends with two FC layers. The first is used to combine features extracted from different parts of the image. The second is used for the actual classification, providing the percentages relating to each class as output. TensorFlow uses "Glorot" initialization (also known as "Xavier" initialization) as the default initialization for dense (fully connected) layer weights and for convolutional layers when no initialization method is specified, this is because, usually it seems work very well [18]. Finally, the Rectified linear activation function was chosen as the activation function, as the effectiveness of the ReLU is often evident in deep learning problems, where its simplicity and the overcoming of the vanishing gradient problem are considered advantages [3]:

$$ReLU(x) = max(0, x) \tag{1}$$

B. VGG16 Network Architecture

VGG16 is a convolutional neural network (CNN) that has achieved considerable success in the field of computer vision. It was introduced in 2014 by a team of researchers from the Visual Graphics Group (VGG) at the University of Oxford. The network features remarkable depth, with 13 convolution layers, 5 pooling, followed by three fully connected layers for classification. VGG16 was designed for image processing and has demonstrated excellent performance in image recognition and classification on datasets such as ImageNet. Its architecture has been fundamental in the development of ever deeper and more complex neural networks. The choice fell on this network as transfer learning was done on some architectures such as VGG19 and DenseNet but VGG16 was the best, so only this one was reported.

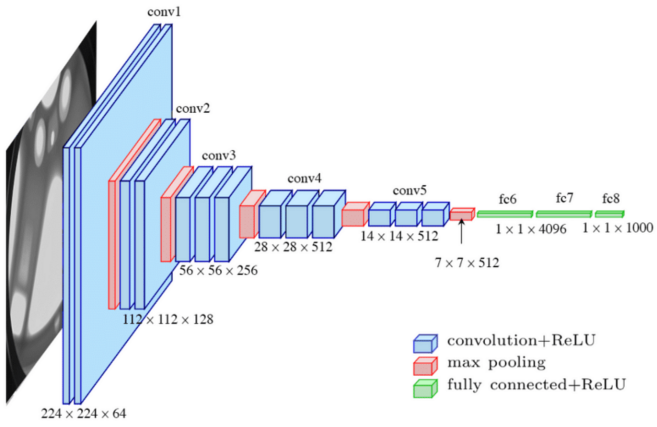

Fig. 3: VGG16 architecture.

C. Training the Networks

Parameters and images were included completely in the GPU memory. As regards model selection, Adam was chosen as the optimizer with *learning_rate=0.001*, as it was the best optimizer among all those tested. The learning rate value was chosen by testing from a set of values ranging from [0.0001, 0.001, 0.01, 0.1]. An Early stopping mechanism was implemented to monitor the accuracy value in the validation set with a patience value of 5, usually the loss is monitored so that as soon as it stops growing or even begins to rise, the training is interrupted to avoid overfitting, the choice to monitor validation accuracy was made because we wanted to achieve the best possible result for the latter. Every time the validation accuracy value improves, the weights are saved to a file so they can be reused without the need to retrain everything. The categorical cross entropy loss was used as the loss as it appears to be the best for classification tasks. The same configuration was implemented for both the custom CNN network and VGG16. Furthermore, since the dataset is balanced, accuracy was chosen as the evaluation metric, defined as:

$$Accuracy = \frac{\#\ Correct\ Prediction}{\#\ Total\ Cases} \tag{2}$$

IV. Experiments

In the following section, the dataset used for the experiment and the results obtained both through the use of the custom CNN network and the pre-trained VGG16 network are presented respectively. In the first case it was decided to present two subcases in turn, one in which the training is done on the training folder and the prediction on the test folder, the second instead combines training and testing in a single folder and then divides everything because we want to highlight the substantial difference between the images of the training folder and those of the test folder.

A. Dataset

To evaluate the proposed approach we perform the experiment on a public dataset taken from Kaggle and entitled "Brain Tumor Classification (MRI)". This dataset is quite popular among the scientific community. The dataset consists of a total

of 3264 MRI images, already divided between Training and Test folders, mostly 512 x 512 in size, but they were resized to 224x224 during training for memory reasons.

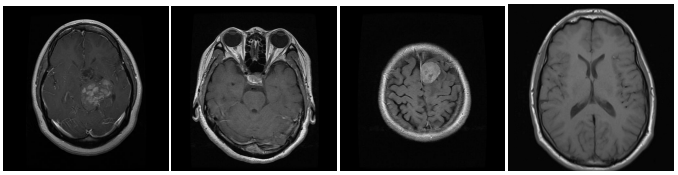

Fig. 4: Respectively Glioma tumor, Pituitary tumor, Meningioma tumor, No tumor

At first glance it is not noticed, but when you analyze the training and prediction results, it is clear that the images in the Test folder are completely different from those in the Training folder, leading to results that will be analyzed in section IV-B. To try to succumb to this problem, a first attempt at data augmentation was implemented, creating a final amount of data equal to three times the amount of images in the starting dataset. Transformations were applied that added Gaussian noise, salt and pepper noise, and brightness and contrast alterations to the images, respectively.

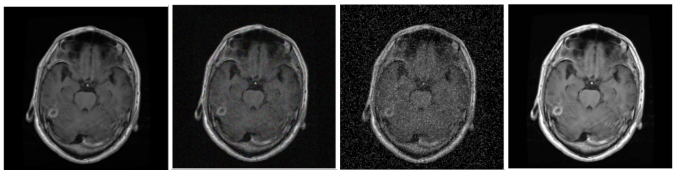

Fig. 5: Respectively original image, Gaussian noise, Salt and Pepper noise, increase brightness and contrast.

These transformations were chosen as they were considered the most suitable for the purposes of a context such as MRI images, unfortunately, however, this modification of the dataset was not sufficient to modify the performance of the model which, despite this, was unable to improve its accuracy. Analyzing the dataset even more in depth, it was noticed that within the Training and Test set there were duplicates, but the even more disconcerting thing, which made all the changes unravel as we will see in the next section, is that some Training images are also found in the Test folder. All these duplicates have been identified using a function that attributes a unique hash to each image, created on the basis of the difference between adjacent pixels. After eliminating these samples the size of the dataset increased from 3264 samples to 2867 samples.

B. Custom CNN

Let's first analyze the case in which the dataset without duplicates is used but without the union of the two training and test folders. You can immediately realize that something is wrong by comparing the training/validation results and the test results. As can be seen in Fig. 6, the model apparently seems to work reasonably well, reporting the results reported in Tab. I for the training and validation phases respectively.

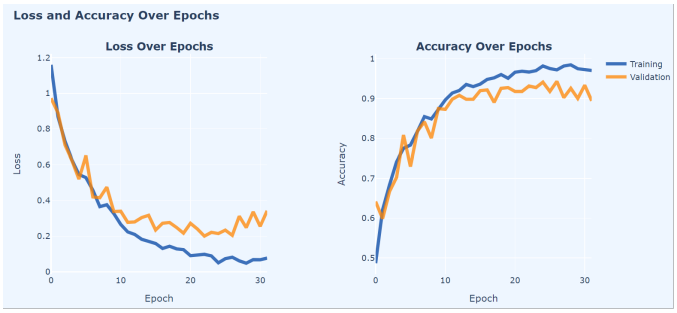

Fig. 6: Training, validation loss and training, validation accuracy respectively for the unmerged dataset.

TABLE I: Performance of the unmerged dataset

	Accuracy
Training	97%
Validation	94%
Test	41%

The only plausible explanation is that the test images are completely different from those for which training and validation was done. Initially it is not easy to discover because when the dataset was analyzed using EDA, a human does not notice this substantial difference with the naked eye. Now let's move on to the case where the training and test folder were merged, mixed, and then split into two new training and test folders. By doing so we ensured that the network also trained on those very different images that characterized the test dataset. Obviously this is not a completely correct approach, in a competition or in a real context this method could never have been applied, it is reported here to make it clear the reason for the poor performances. The training is therefore repeated producing the results reported in Tab. II and Fig. 7.

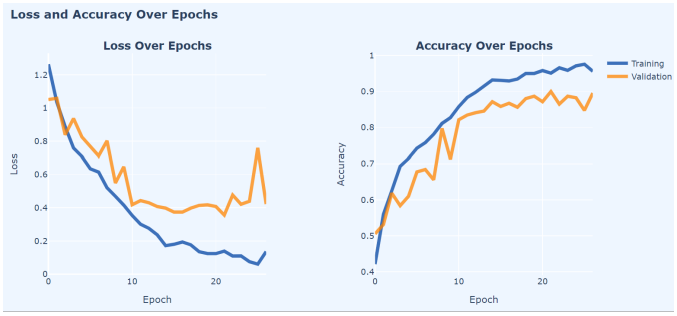

Fig. 7: Training, validation loss and training, validation accuracy respectively for the merged dataset.

TABLE II: Performance of the merged dataset

	Accuracy
Training	95%
Validation	89%
Test	87%

It is easy to see from the images and table above that the model learns well from the training data and is able to predict well even in the testing phase, albeit with a slight overfitting.

This confirms that it was not the architecture of the model or some parameter set in a non-optimal manner that caused the bad performance, but precisely the diversity of the test images.

C. VGG16

Finally, we wanted to test a solution using the famous Oxford network, VGG16 pre-trained on ImageNet. Transfer learning was done, in particular since the weights we used on VGG16 came from the ImageNet dataset, which is a very different dataset from the one examined in this experiment, and since our dataset in question is very small , it was not possible to do fine tuning (risk of overfitting), nor to replace only the Fully Connected layer (learned features too different from my task). We could detach the network from a little before the FC layers, that is, we remove some convolutional layers, leaving only the first ones unchanged. The training and validation graph are represented in Fig. 8 while the test performances can be viewed in Tab. III. It can be seen that not even a powerful network like VGG16 was able to learn the features necessary for correct classification of the test images.

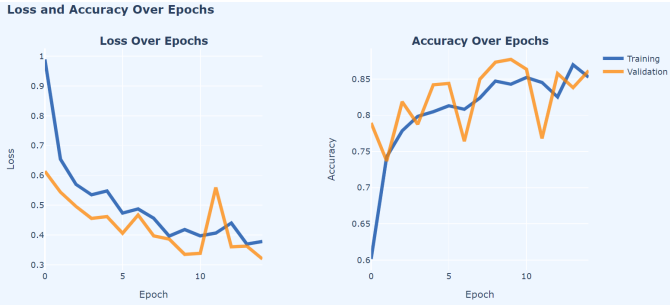

Fig. 8: Training, validation loss and training, validation accuracy respectively for the VGG16 model.

TABLE III: Performance of VGG16 on unmerged dataset

	Accuracy
Training	85%
Validation	86%
Test	40%

V. Conclusion

In this paper we wanted to address the problem of classifying MRI images of brain tumors through two types of architectures, one custom and one pre-trained by transfer learning. It has been shown that the network learns very well from the training data and is also able to make a good prediction in the validation data, unfortunately, however, when the prediction is made on the test data, the performance drops dramatically, this is because the images of tests present a completely different structure from those for which the network was trained, even if it would not be obvious to the naked eye. When the two training and test folders were mixed and then split again, the performance increased significantly, which implies that the tested models learn well from the data they see. Tasks similar to the one in this paper present good results both for the pre-trained [13] architectures, with 96% accuracy on VGG16, or [14] which presents 91.38% accuracy again on VGG16, and for the custom CNN architectures [19], which is a much more simple architecture than the one presented in this article and presents an accuracy of 84% in the validation set. The problem with this experiment was the dataset, although it was taken from kaggle and was a highly rated dataset, it presents a very difficult structure for the classification task. Possible future work could be to make a more relevant data augmentation, presenting different combinations of transformations, perhaps the resized crop, rotation or translation, or to generate new data through the use of Generative Adversarial Network.

References

[1] N. O'Mahony, S. Campbell, A. Carvalho, S. Harapanahalli, G. V. Hernandez, L. Krpalkova, D. Riordan, and J. Walsh, "Deep learning vs. traditional computer vision," in *Advances in Computer Vision*, K. Arai and S. Kapoor, Eds. Cham: Springer International Publishing, 2020, pp. 128–144.

[2] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[4] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *CoRR*, vol. abs/1409.4842, 2014. [Online]. Available: http://arxiv.org/abs/1409.4842

[5] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition." Computational and Biological Learning Society, 2015, pp. 1–14.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[7] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," *CoRR*, vol. abs/1610.02357, 2016. [Online]. Available: http://arxiv.org/abs/1610.02357

[8] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017.

[9] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," 2018.

[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023.

[11] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," 2021.

[12] Z. Yu, Q. He, J. Yang, and M. Luo, "A supervised ml applied classification model for brain tumors mri," *Frontiers in Pharmacology*, vol. 13, 2022. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fphar.2022.884495

[13] H. A. Khan, W. Jue, M. Mushtaq, and M. U. Mushtaq, "Brain tumor classification in mri image using convolutional neural network," *Mathematical Biosciences and Engineering*, vol. 17, no. 5, pp. 6203–6216, 2020. [Online]. Available: https://www.aimspress.com/article/doi/10.3934/mbe.2020328

[14] S. Kuraparthi, M. K. Reddy, C. N. Sujatha, H. B. Valiveti, L. C. Duggineni, M. Kollati, P. Kora, and V. Sravan, "Brain tumor classification of mri images using deep convolutional neural network," *Traitement du Signal*, vol. 38, pp. 1171–1179, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:243831043

[15] M. M. Badza and M. C. Barjaktarović, "Classification of brain tumors from mri images using a convolutional neural network," *Applied Sciences*, vol. 10, no. 6, 2020. [Online]. Available: https://www.mdpi.com/2076-3417/10/6/1999

[16] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 818–833.

[17] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," 2012.

[18] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, Y. W. Teh and M. Titterington, Eds., vol. 9. Chia Laguna Resort, Sardinia, Italy: PMLR, 13–15 May 2010, pp. 249–256. [Online]. Available: https://proceedings.mlr.press/v9/glorot10a.html

[19] N. Abiwinanda, M. Hanif, S. Hesaputra, A. Handayani, and T. Mengko, *Brain Tumor Classification Using Convolutional Neural Network*, 05 2019, pp. 183–189.