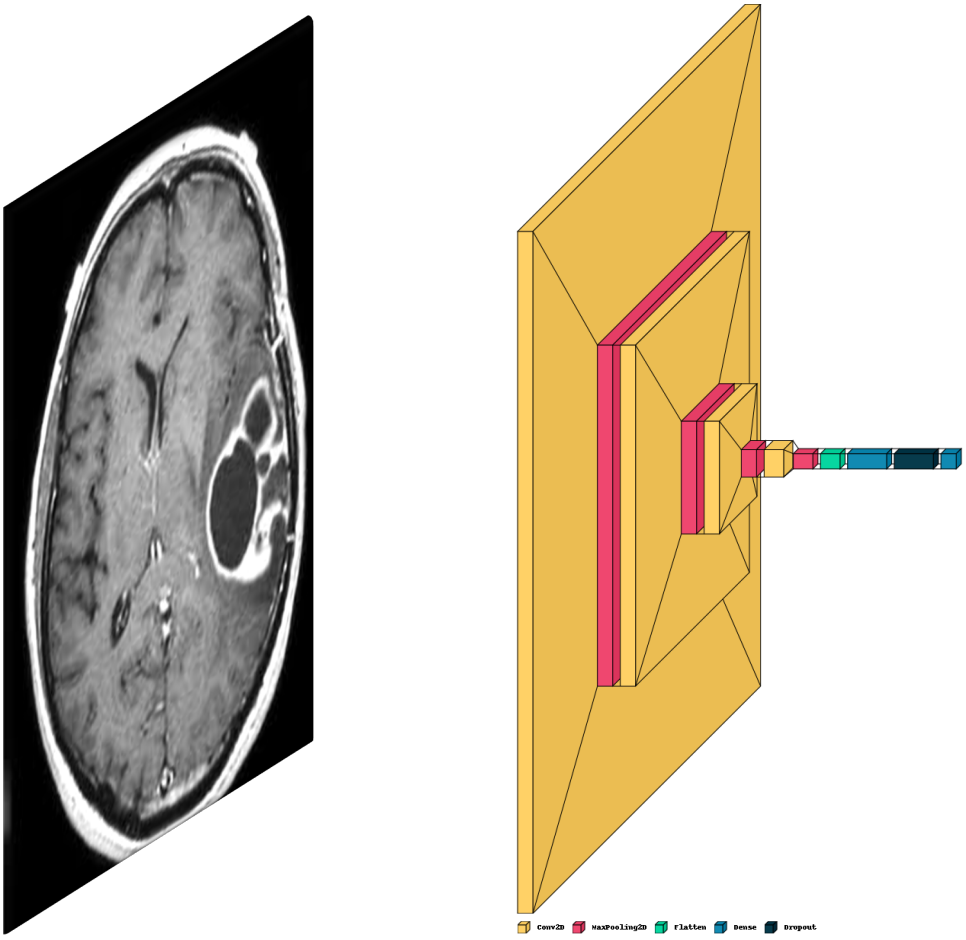
University of Pisa

Brain Tumor Classifier

An application of Deep Learning techniques for the classification of brain tumors from MRI



Di Donato Mattia, Giorgi Matteo

Summary

Abstract

Brain tumors are recognized as highly aggressive diseases, affecting both children and adults. They account for a significant majority, ranging from 85% to 90% of all primary Central Nervous System (CNS) tumors. Annually, approximately 11,700 individuals receive a diagnosis of a brain tumor. The 5-year survival rates for cancerous brain or CNS tumors are estimated at around 34% for men and 36% for women. Given the gravity of this condition, it is imperative to employ proper treatment strategies, meticulous planning, and precise diagnostic methods to enhance the life expectancy of affected patients.

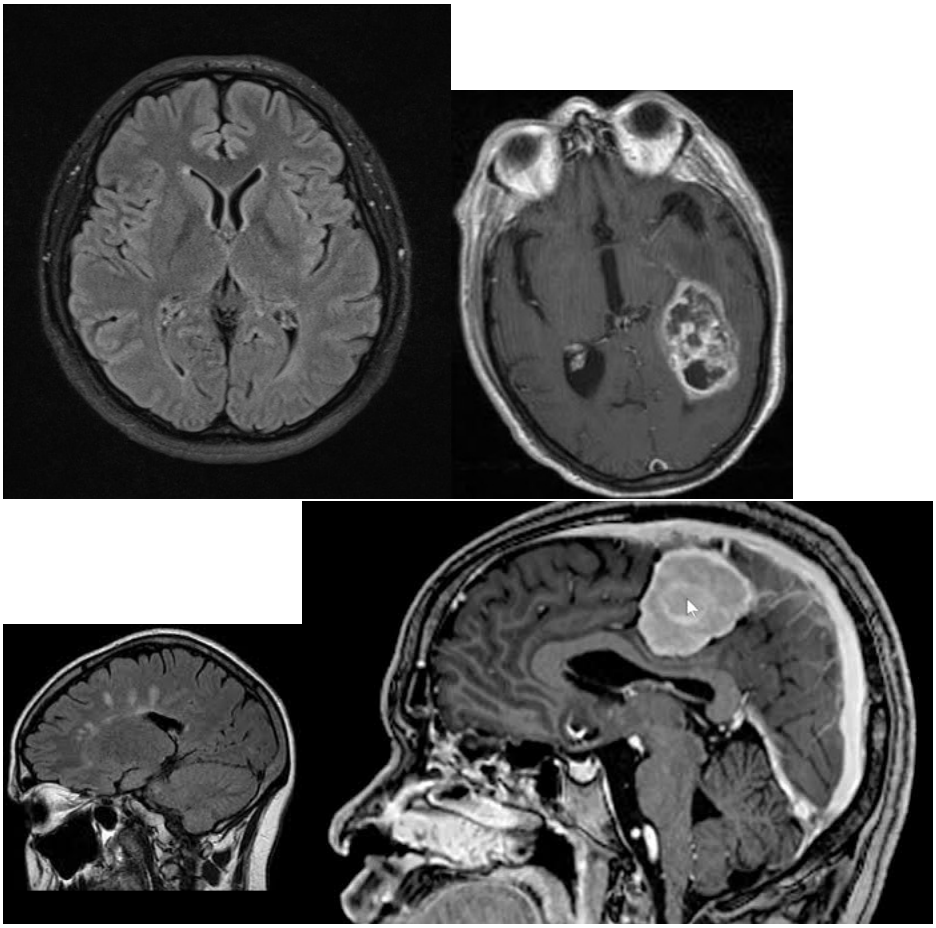
Among the various techniques available for brain tumor detection, Magnetic Resonance Imaging (MRI) stands out as the most effective modality. A huge number of images are generated through scans, and they are examined by radiologists. A manual examination can be error-prone due to the level of complexities involved in brain tumors and their properties. Application of automated classification techniques using Deep Learning Algorithms for instance Convolution Neural Network (CNN) and Transfer Learning (TL) have gained considerable attention as they would help doctors in the detection of this disease improving accuracy and efficiency.

Dataset

The dataset used for the realization of this application is taken from Kaggle (https://www.kaggle.com/datasets/sartajbhuvaji/brain-tumor-classification-mri).
It has a dimension of 90MB and is a collection of 3264 MRI (gray scale) of different resolution. These images have been classified by experts and have been divided into 4 categories with the related support.

Tumor typology	Instances
Glioma tumor	926
Meningioma tumor	937
No tumor	500
Pituitary Tumor	901

These are some sample images from it.



Methods and Experiments

Data Preprocessing

The data preprocessing phase involved several key steps to prepare the dataset for training our model.

First, we conducted an analysis of the dataset to understand the class distribution and ensure class balance. This analysis allowed us to identify a class imbalance, found in the part of the dataset dedicated to negative MRIs.

Next, we performed the division of the data set into training, validation, and test sets. The training set was used to train the model, the validation set to tune the hyperparameters and monitor performance, and the test set to evaluate the generalization ability of the final model.

To address class imbalances, we applied data augmentation techniques. Initially, we focused on augmenting the minority class to increase its representation in the data set. This involved generating additional synthetic images using transformations such as rotate, scale, and flip.

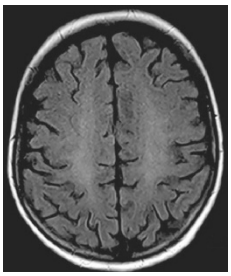


Fig 1 - Original MRI

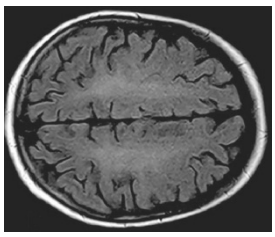


Fig 2 - 90° rotation

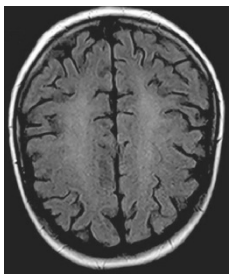
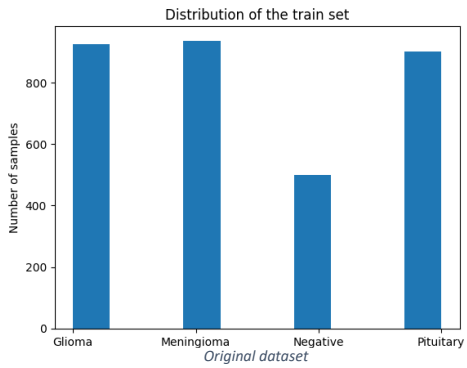


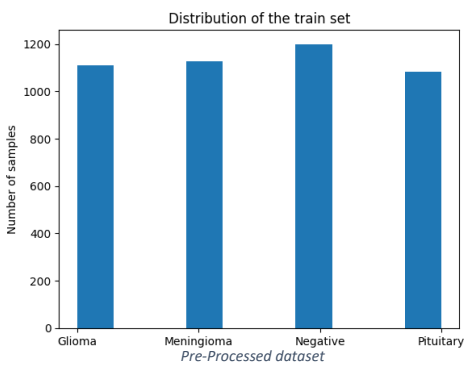
Fig 3 - 180° rotation

Additionally, we applied data augmentation to the entire training dataset, including all classes. This larger data increments further enriched the training data, introducing variations that could improve the model's ability to generalize and improve its robustness.

After the data augmentation process, we performed a final analysis of the dataset to ensure that class equilibrium was indeed achieved. This analysis allowed us to verify that the augmentation techniques successfully balanced the classes, ensuring that each class had a comparable number of samples to train.



Original dataset

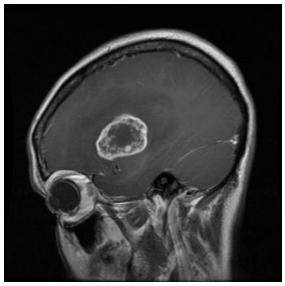


Pre-Processed dataset

Following this comprehensive data preprocessing pipeline, including class analysis, dataset splitting, targeted and general data augmentation, and a final class balance analysis, we aimed to create a set of data, as robust and balanced as possible that would effectively contribute to the training and evaluation of our model despite the size of the initial dataset being quite small.

CLAHE

To improve the quality and feature representation of images in our dataset, we incorporated the Contrast Limited Adaptive Histogram Equalization (CLAHE) method as a preprocessing technique by creating a parallel dataset for use with the various models tested.



Original image

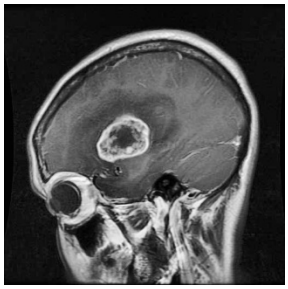


Image with CLAHE

CLAHE (Contrast Limited Adaptive Histogram Equalization) is an image enhancement technique that aims to improve contrast by redistributing the luminance values of the image. The CLAHE algorithm consists of three main parts: tile generation, histogram equalization, and bilinear interpolation.

In the CLAHE algorithm, the input image is divided into tiles, where each tile represents a distinct section of the image. Histogram equalization is then applied to each tile independently, using a predefined clip limit. The histogram equalization process involves calculating the histogram for each tile, computing excess values, redistributing the excess values, and finally scaling and mapping the values using a cumulative distribution function (CDF).

During histogram equalization, if the histogram bin values exceed the clip limit, they are accumulated and redistributed into other bins to prevent over-amplification of noise. The CDF values of each tile's histogram are then scaled and mapped using the pixel values of the input image.

After histogram equalization is applied to all the tiles, the resulting tiles are stitched together using bilinear interpolation. This process creates an output image with improved contrast and enhanced edge definitions.

CNN from scratch

In our project, we conducted a series of experiments to develop a CNN model for classifying brain tumor images into four different classes.

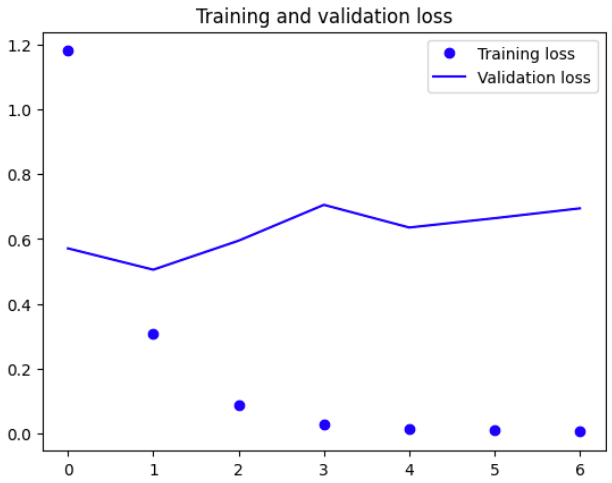
We started with a small model then gradually increase its size with the idea of starting with a less performant model going towards a more accurate one. The images given in input were resized to 250x250 pixel and normalized with Rescaling, so that each pixel value became in the range [0,1].
We chose to use 32 as batch size, which is the number of samples considered in each iteration. Then we decided to use as padding the "same" technique, to obtain the same dimension of the output image, as activation functions we used the ReLU.

We adopted an incremental architecture increasing size of the max pooling layers as we move towards the last layers, this approach allows capturing features at different sizes as you progress through the network. It can be particularly useful to emphasize wider features at higher levels of the network. As last layer we use a dense one with 4 neurons and Softmax activation function that is used to obtain a probability distribution overall output classes. Each neuron in this layer represents the probability of belonging to a specific class.

CNN Experiments

First Experiment

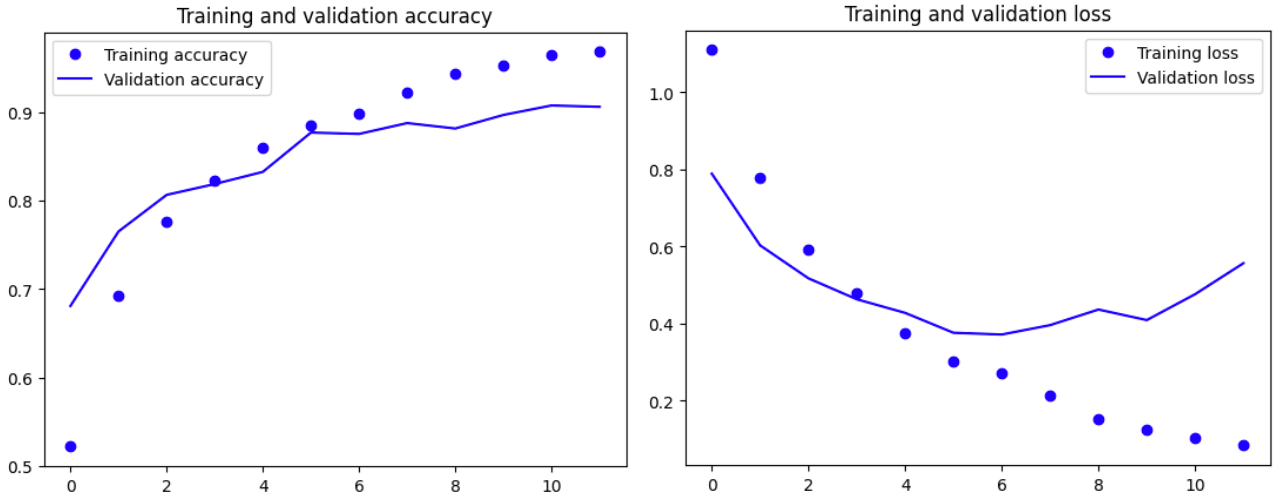
We started with a simple CNN model consisting of one convolutional layer with 32 filters with 3x3 kernel size and one max pooling layer with 2x2 window size. This was followed by a Flatten layer and the Dense layer with four neurons.
Due to the simplicity of this model, the results were poor and strongly conditioned by overfitting. We obtain a validation loss of 0.6947 and as can be seen from the graph, the difference between training and validation is quite high. Given the simplicity of the network, it has certainly managed to learn a mapping of the input images but fails to generalize well. Therefore, we have decided to make the model more complex to extract better information.



Second Experiment

We addressed the issue of overfitting by introducing a dropout layer with a probability of 0.5. Additionally, we incorporated three convolutional layers with a kernel size of 3x3 and an incremental number of filters (32-64-128) and three max pooling layers with progressively

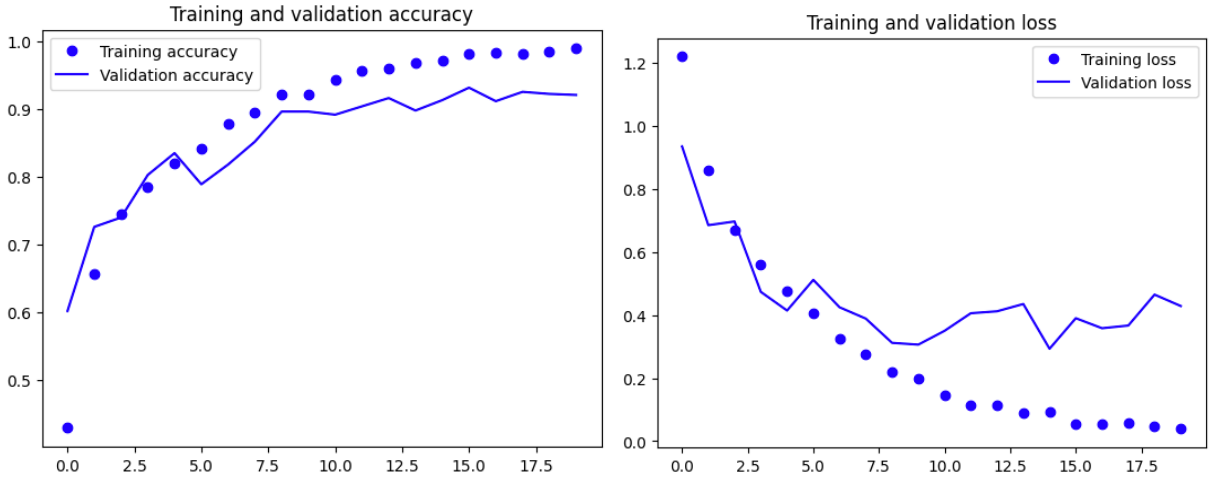
increasing window size from 2 to 4. To enhance the complexity, we added a dense layer with 256 units. We notice improvements but the validation loss, that was at 0.3719, was still higher than desired.



Looking at the graph, we can see that the more complex structure of the network has allowed better extraction of features from images, resulting in a decrease of approximately 0.3 in validation loss from the first experiment.
Despite the improvements, we still observe a slight overfitting of the model. In particular, from the loss graph, we can see that after the seventh epoch, the validation loss no longer improves and even tends to increase, while the training loss continues to improve.

Third Experiment

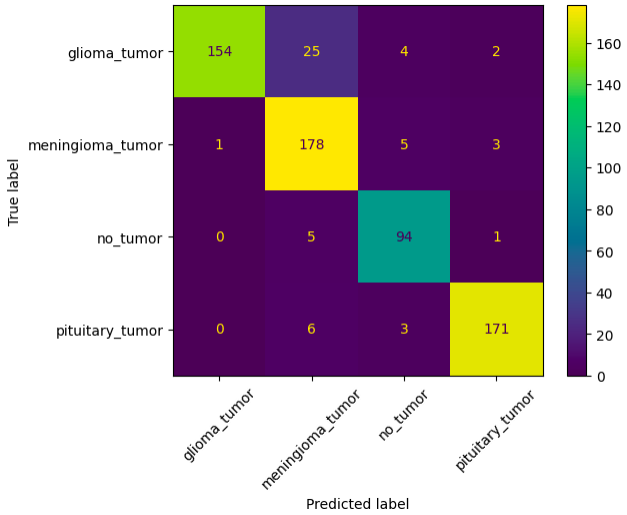
We attempted to overcome this challenge by introducing an additional convolutional layer with 256 feature maps and an extra max pooling layer of 5x5 window size. This modification aimed to extract more meaningful features from image.

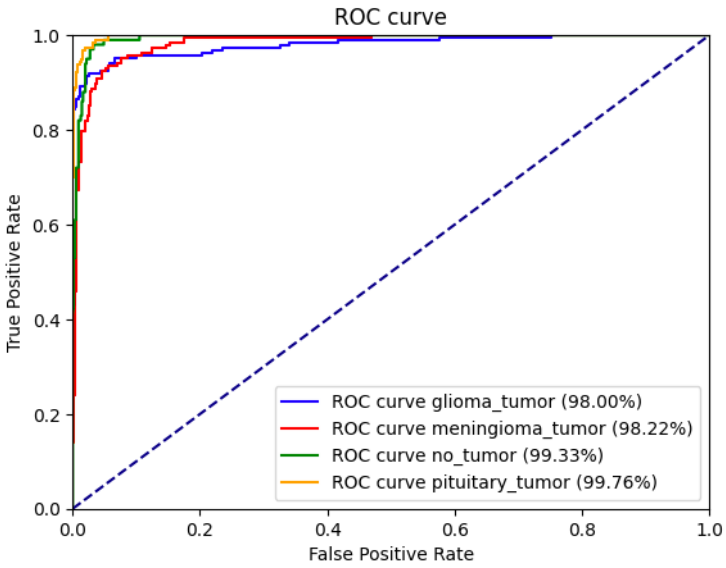


From the graphs, we can see that the validation accuracy closely follows the training accuracy, reaching a value of 0.9126. In this experiment, we also manage to achieve the best validation loss, of 0.2930. However, the oscillation of this value in the graph could be an indication of model overfitting or simply reaching a local optimum of the model weights.
We report below the confusion matrix and the roc curves of each class.

Classification report:

	precision	recall	f1-score	support
glioma_tumor	0.9935	0.8324	0.9059	185
meningioma_tumor	0.8318	0.9519	0.8878	187
no_tumor	0.8868	0.9400	0.9126	100
pituitary_tumor	0.9661	0.9500	0.9580	180
accuracy			0.9156	652
macro avg	0.9196	0.9186	0.9161	652
weighted avg	0.9232	0.9156	0.9161	652





We obtain a value of 0.92 for the weighted F1 on the test set.

Fourth Experiment

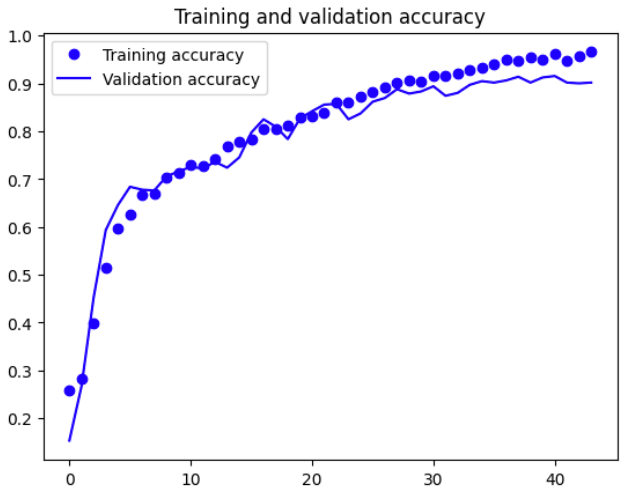
Continuing our efforts to improve performance and decrease the validation loss value, in the fourth experiment we decide to increase the number of units in the dense layer to 512. This adjustment aimed to further enhance the model's capacity to learn more pattern however the obtained graphs are like those of the previous experiment, with a slightly lower performance about validation loss and accuracy and a 0.90 for the weighted F1.

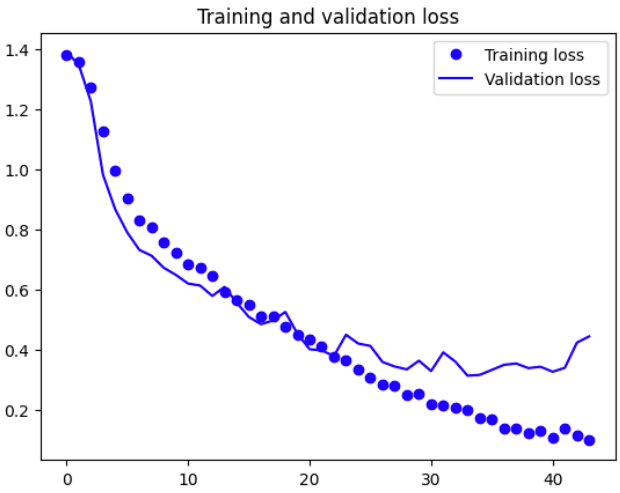
Fifth Experiment

Our last experiment introduced three additional dense layers with 128, 256, and 512 units, respectively, following the initial dense layer in the fifth experiment.

Each dense layer was accompanied by a dropout layer with a probability of 0.5. We have also decreased the learning rate to 0.0001 and increased the patience to 10, allowing us to wait for more epochs in case the validation loss does not improve.

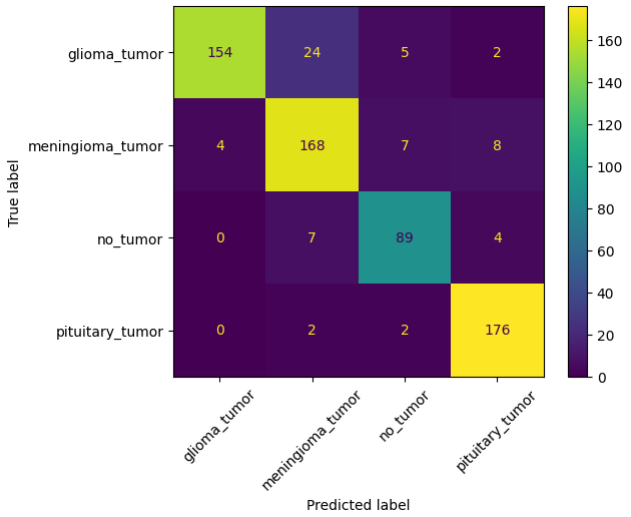
These modifications aimed to further mitigate overfitting and improve the model's generalization capabilities. With this last experiment we obtain a training loss of 0.2015 and validation loss of 0.3143.

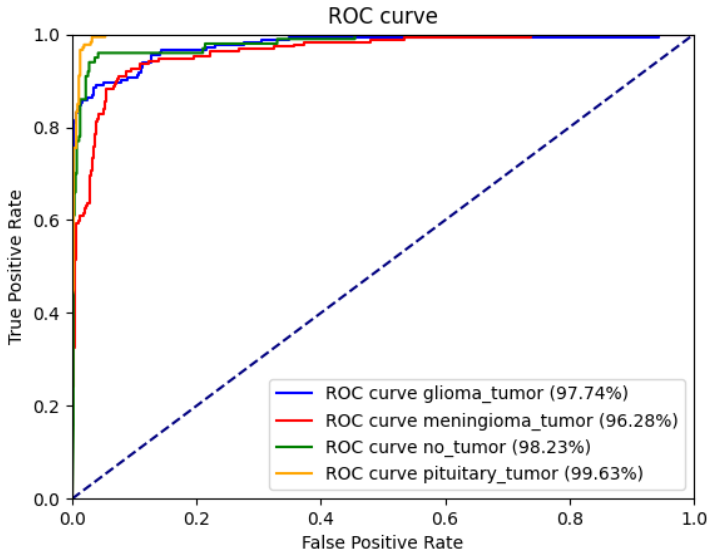




Classification report:

	precision	recall	f1-score	support
glioma_tumor	0.9747	0.8324	0.8980	185
meningioma_tumor	0.8358	0.8984	0.8660	187
no_tumor	0.8641	0.8900	0.8768	100
pituitary_tumor	0.9263	0.9778	0.9514	180
accuracy			0.9003	652
macro avg	0.9002	0.8997	0.8980	652
weighted avg	0.9045	0.9003	0.9003	652





Conclusion

Based on the above graphs and the obtained results, it can be deduced that this latest model has a good validation loss while it is also capable of generalizing well, with a 0.90 of weighted F1 on the test set.

According to the information provided, the best model in terms of F1 score on the test set is the one from experiment 3, achieving an F1 score of 0.92.

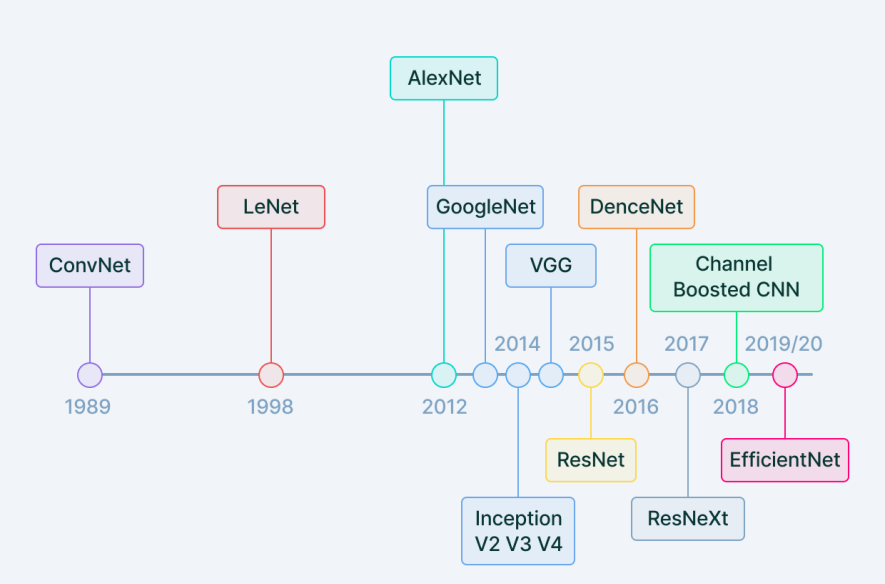
Additionally, the last experiment resulted in a model that does not suffer from overfitting or underfitting and achieves also a good F1 score.

CLAHE experiments

We conducted further experiments with CNN models using a preprocessed dataset subjected to the Contrast Limited Adaptive Histogram Equalization (CLAHE) technique.

CLAHE is a method that enhances the contrast and detail of an image as explained above. The intention was to improve the visibility of features and improve the performance of our models in image classification of brain tumors with CNN. Unfortunately, however, we have not obtained good results compared to the previous ones.

Pretrained models



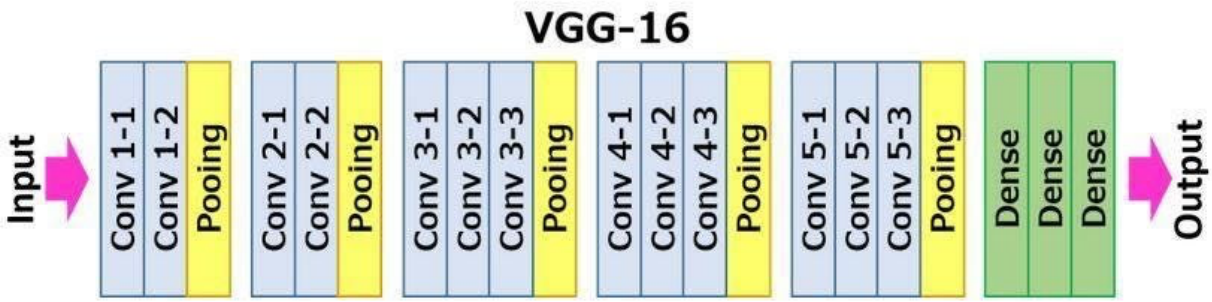
In recent years, pretrained models have revolutionized the field of computer vision by leveraging the power of deep learning. These models, trained on large-scale datasets, capture rich visual representations and can be readily applied to various image-related tasks. In this report, we explore the effectiveness of several popular pretrained models, including VGG16, ResNet50, and InceptionV3, and present our findings in the subsequent sections.

It is important to note that while conducting our research, we came across the EfficientNetB0 architecture, which has shown remarkable performance, including a Weighted F1 score of 0.98 reported on Kaggle. However, it is worth mentioning that the training and test sets used on Kaggle have different distributions compared to our dataset. To ensure a fair and meaningful comparison of EfficientNetB0's performance, it would be necessary to train and evaluate the model on our specific dataset.
https://www.kaggle.com/code/jaykumar1607/brain-tumor-mri-classification-tensorflow-cnn#Evaluation

VGG16

VGG16 is a highly popular CNN architecture used for image classification. The acronym stands for Visual Geometry Group and 16 derives from the fact that the architecture consists of 16 layers, where 13 layers are convolutional layers followed by 3 fully connected layers.
The input image fed into the neural network has a resolution of 224x224x3 pixels (three channels because it is trained on colored images)
The convolutional base consists of five blocks with 2 or 3 convolutional layers whose filters have a size of 3×3, a peculiarity of this architecture that uses very small convolutional filters (receptive fields). Padding and stride are adjusted such that the spatial resolution remains constant after the filter is applied.
The sequence of convolutional blocks is succeeded by a densely connected classifier comprising of three layers. The last layer is a Softmax layer with 1000 nodes representing the probability of the 1000 classes of the ImageNet dataset, on which is trained. All the hidden layers utilize the

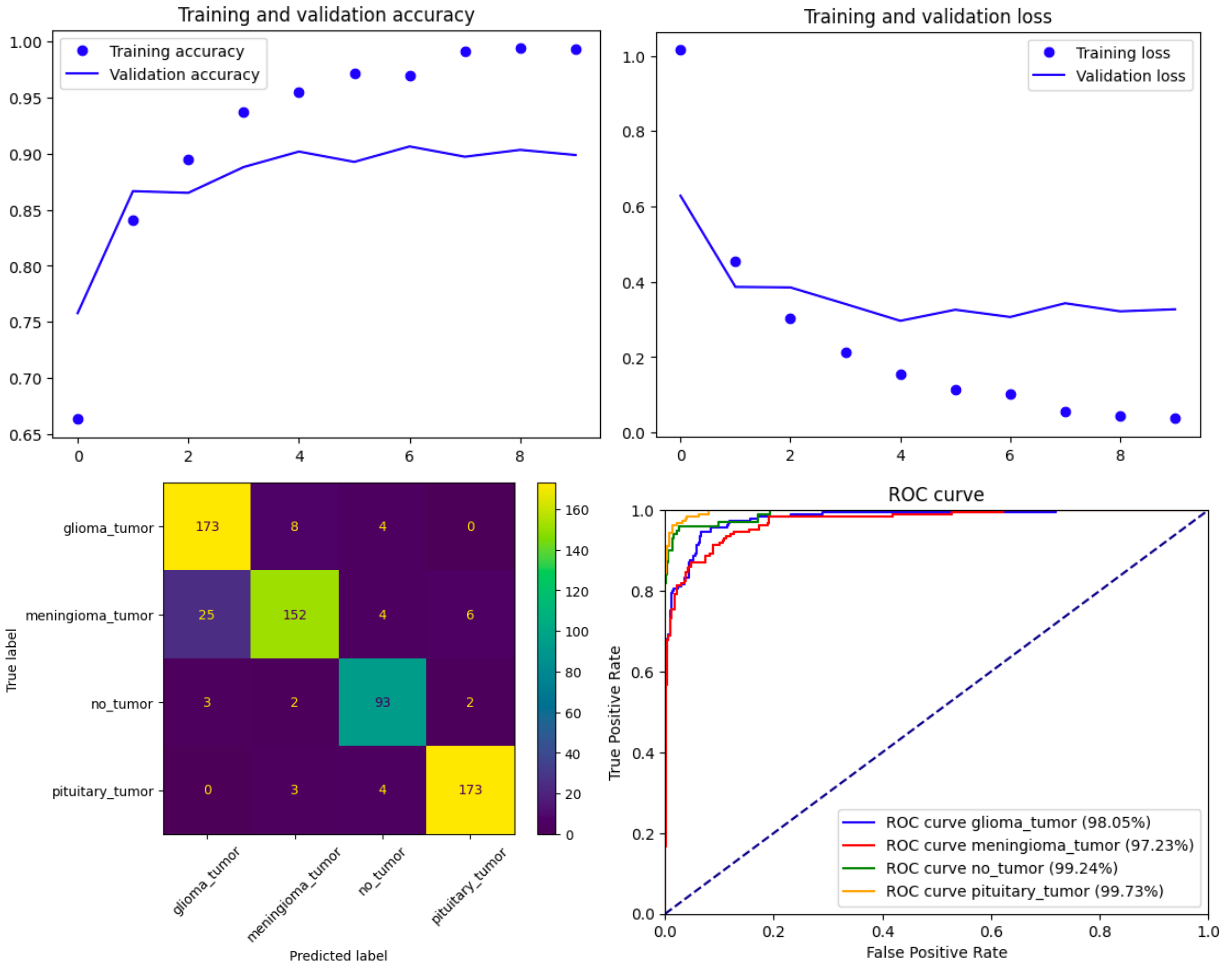
rectification function (ReLU). Additionally, max-pooling is applied between the blocks, using a window size of 2x2 and a stride of 2.



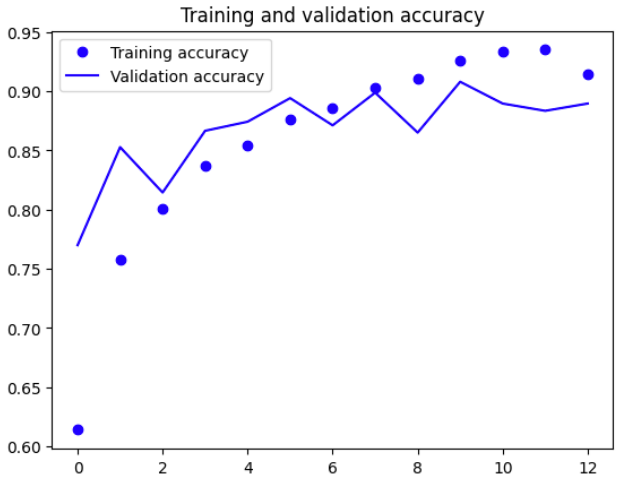
Feature Extraction

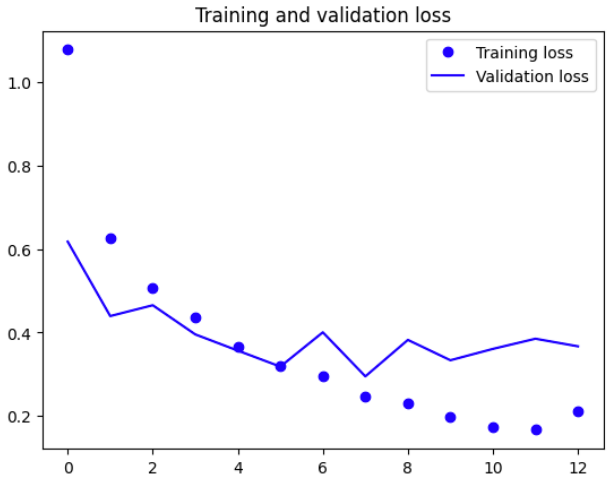
In this part of experiments, we use the vgg16 network with frozen weights adding other layers at the end of it with trainable parameters. Each experiment has a final dense layer with 4 neurons and as activation the Softmax function as these 4 neurons representing the probability of each class.

The first experiment consists of adding a flatten and dense layer of 128 neurons. We obtained good results, for the validation loss 0.2965 and for the validation accuracy 0.9018, anyway the graphs show a bit of overfitting. The weighted f1 obtained is 0.9058 over the test set.



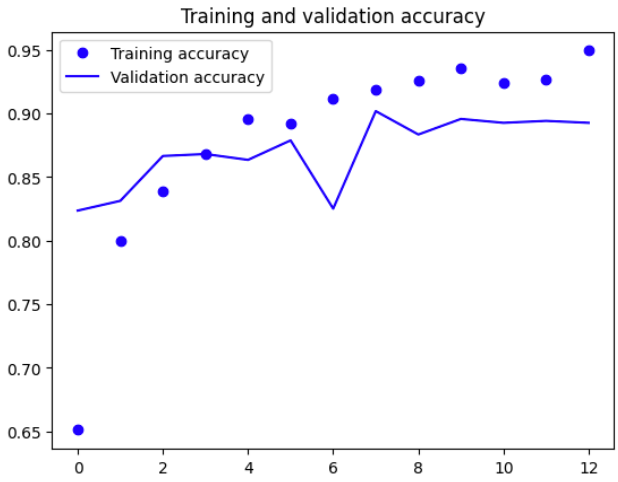
To reduce overfitting, we try to add a dropout layer, with 30% of probability, after the dense one. With this modification we obtain a validation loss of 0.2948 and only a difference of 0.05 from the training loss as also show in the graphs below. We obtained a weighted f1 of 0.8796 over the test set.

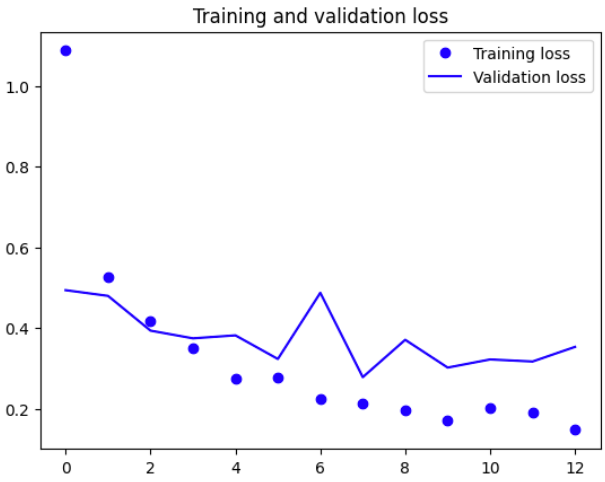




We also considered using the globalAvgPooling layer instead of the flatten. In this way we should be also able to mitigate the overfit because it reduces the number of trainable parameters. However, we are unable to obtain better results than the previous ones, in fact the validation loss is 0.4 and differs by 0.1 from the training one, furthermore the model takes more periods before completing the training.

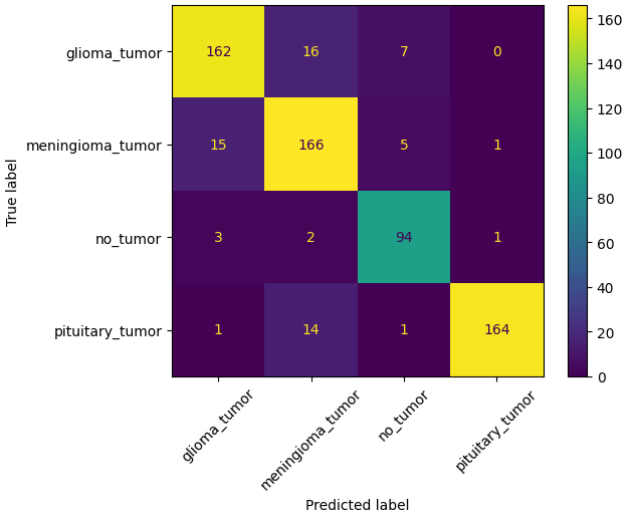
The same experiments were conducted using a dense layer with 256 neurons. The only case in which we observed improvements was when a dropout layer was also present, which allowed us to achieve a validation loss of 0.2778 and a training loss of 0.2134 as we can see from the graph below.

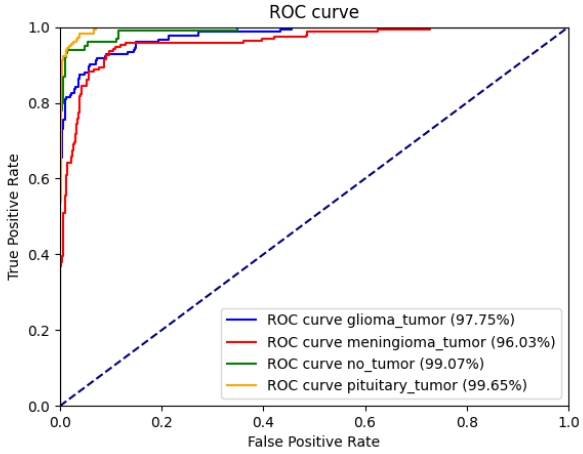




Classification report:

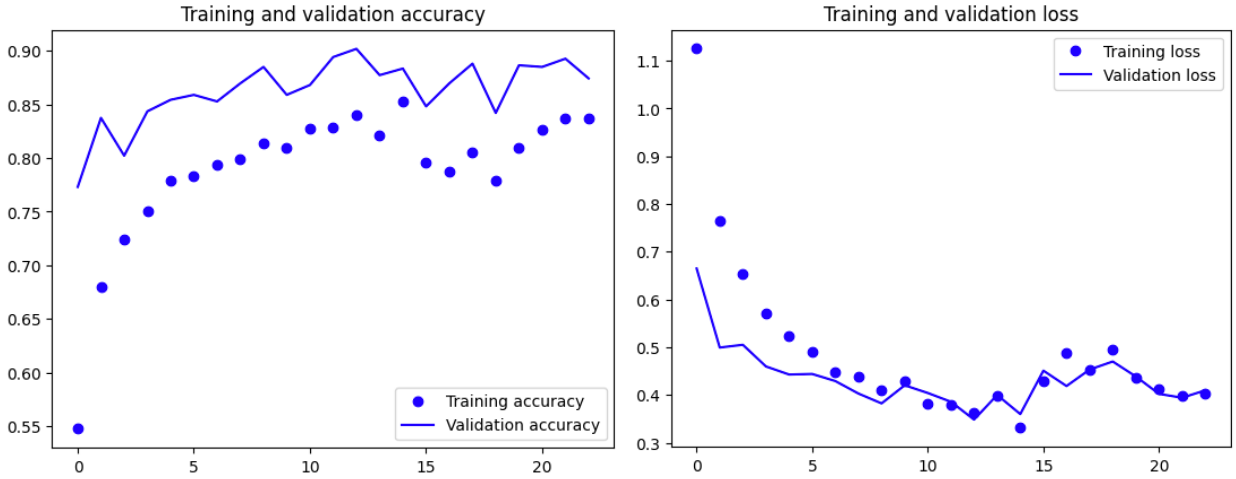
	precision	recall	f1-score	support
glioma_tumor	0.8950	0.8757	0.8852	185
meningioma_tumor	0.8384	0.8877	0.8623	187
no_tumor	0.8785	0.9400	0.9082	100
pituitary_tumor	0.9880	0.9111	0.9480	180
accuracy			0.8988	652
macro avg	0.9000	0.9036	0.9009	652
weighted avg	0.9019	0.8988	0.8995	652



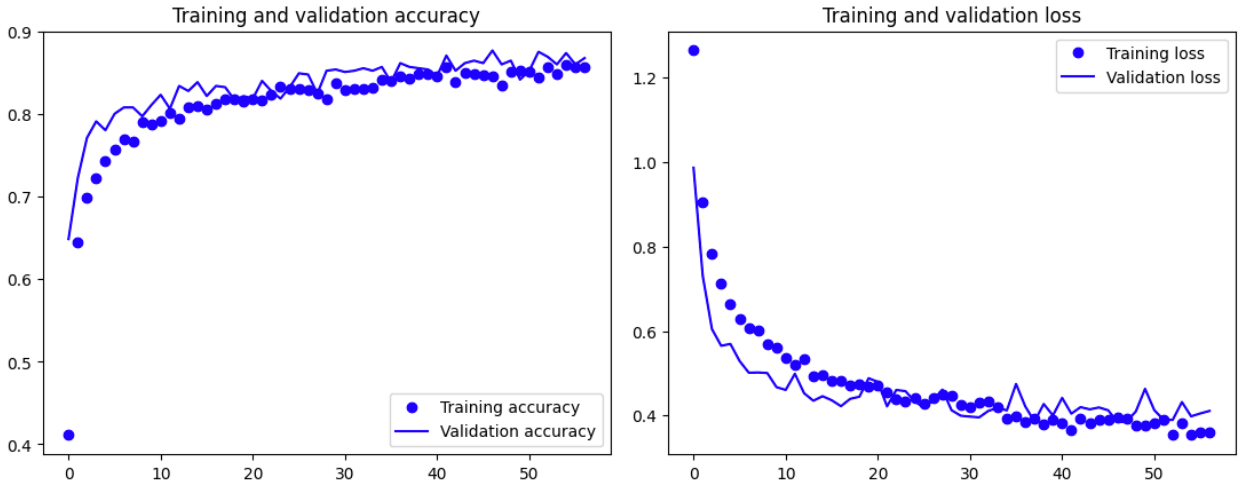


We also obtain 0.8995 for the weighted f1 on the test set.

Additional experiments were conducted, we attempted to further complicate the network to achieve even lower loss by adding two dense layers with 128 neurons each, along with a 30% dropout to prevent overfitting. However, the outcome was a model that was overly simplistic. Both the training loss and validation loss were high, and notably, the validation curve consistently remained above the training curve, indicating that the model failed to learn sufficiently from the data, resulting in underfitting.



By replacing the dense layers with 256 neurons and setting the dropout to 50%, we obtained a model with very similar training and validation curves, indicating a model that neither suffered from overfitting nor underfitting.



Nevertheless, we were unable to achieve better performance compared to the model with only the dense layer of 256 neurons and the dropout at 30%.

Fine Tuning

We experimented with fine-tuning both the model with 128 neurons and the model with 256 neurons and dropout. The model with 128 neurons showed a slight improvement, while the model with 256 neurons, upon unlocking the parameters of VGG16, started to exhibit worsening performance.

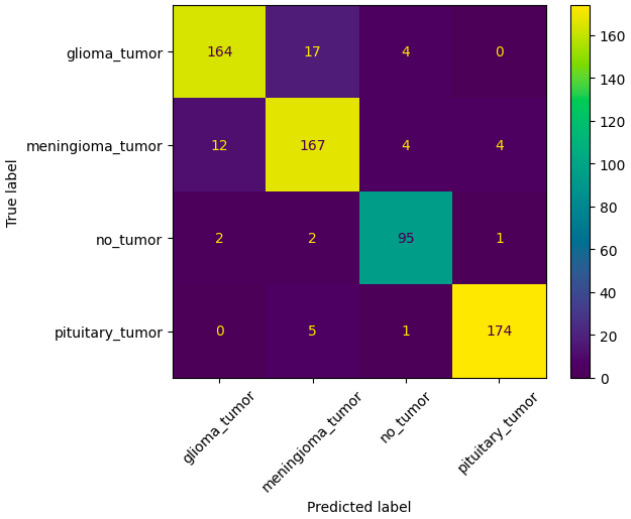
Here are the results obtained with the model containing 128 neurons during the fine-tuning experiments:

Experiment	Training Loss	Validation Loss	Weighted F1
Block5Conv3	0.0470	0.2800	0.9021
Block5Conv2-3	0.0236	0.2849	0.9141
Block5Conv1-2-3	0.0048	0.3361	0.9203
Block4-5	0.0557	0.3337	0.9078
Block3-4-5	0.0182	0.3900	0.9076

Although only the first two experiments demonstrated an improvement in validation loss, particularly when only conv3 was unlocked (reducing the validation loss from 0.2965 to 0.2800), unlocking all three convolutional layer in block 5 led to an enhancement in weighted F1 score on the test set, reaching 0.9203 for the fine-tuned model compared to the initial model's score of 0.9058.

Classification report:

	precision	recall	f1-score	support
glioma_tumor	0.9213	0.8865	0.9036	185
meningioma_tumor	0.8743	0.8930	0.8836	187
no_tumor	0.9135	0.9500	0.9314	100
pituitary_tumor	0.9721	0.9667	0.9694	180
accuracy			0.9202	652
macro avg	0.9203	0.9241	0.9220	652
weighted avg	0.9207	0.9202	0.9203	652

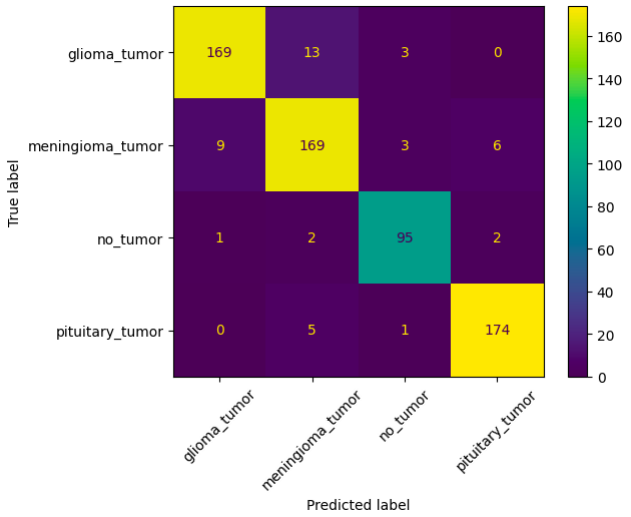


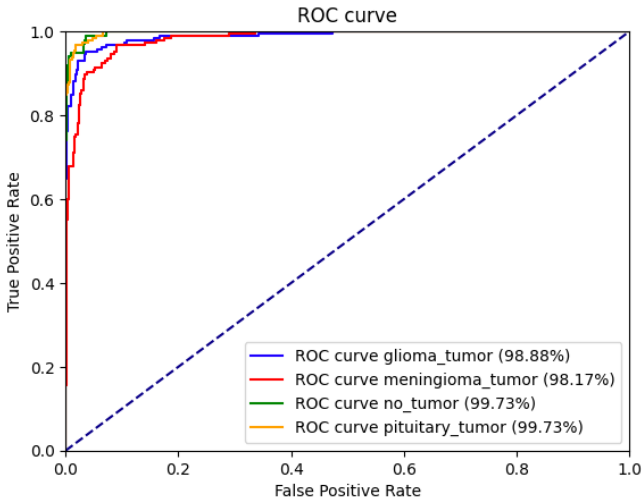
CLAHE Experiments

The same experiments were conducted with the dataset preprocessed using Clahe, but no significant improvements in terms of validation loss were obtained during training. However, we observed slight improvements in terms of the F1 score on the test set. Therefore, we have decided to present the results obtained from fine-tuning the model with 128 neurons and dropout by unlocking all the convolutional layers in Block 5.

Classification report:

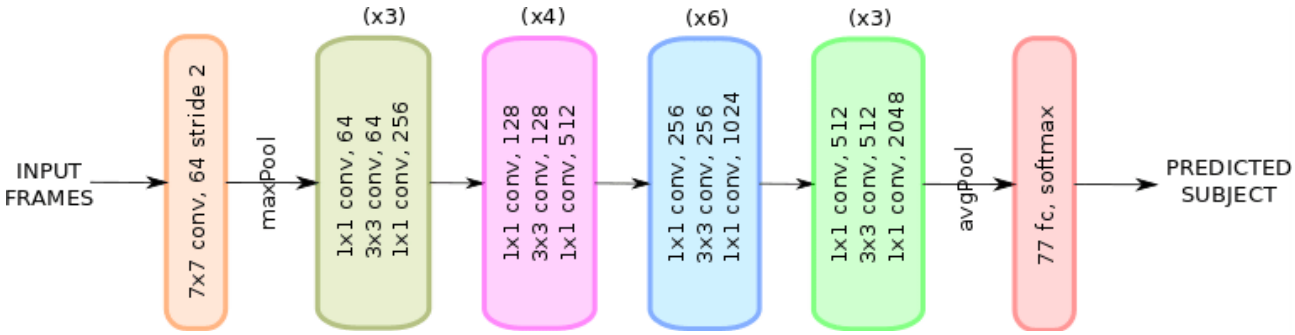
	precision	recall	f1-score	support
glioma_tumor	0.9441	0.9135	0.9286	185
meningioma_tumor	0.8942	0.9037	0.8989	187
no_tumor	0.9314	0.9500	0.9406	100
pituitary_tumor	0.9560	0.9667	0.9613	180
accuracy			0.9310	652
macro avg	0.9314	0.9335	0.9324	652
weighted avg	0.9311	0.9310	0.9310	652





ResNet50

In this part we present an overview of the experiments conducted using a pretrained ResNet50 neural network for feature extraction and fine-tuning.



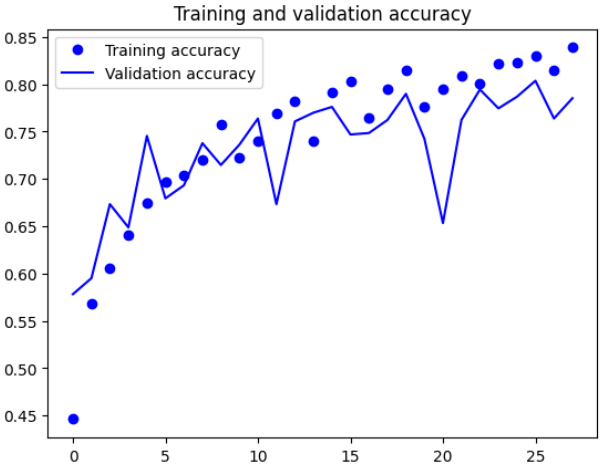
ResNet-50 is a deep convolutional neural network architecture renowned for its exceptional performance in image classification tasks. It addresses the challenges of training very deep networks by introducing residual blocks with skip connections. These connections enable the network to learn residual mappings, facilitating the flow of information and alleviating the vanishing gradient problem.
With a total of 50 layers, ResNet-50 comprises convolutional layers, pooling layers, and fully connected layers. The architecture begins with feature extraction using convolutional and pooling layers, followed by four stages of residual blocks. These blocks progressively capture increasingly complex features, enabling the network to learn intricate representations from the input data.
To condense spatial information, ResNet-50 incorporates a global average pooling layer, which summarizes the extracted features across spatial dimensions. Finally, a fully connected layer with a SoftMax activation function performs the classification task.
The depth of ResNet-50, combined with its skip connections and residual learning, allows it to effectively learn complex feature representations, leading to outstanding performance in various computer vision tasks. Its versatility and accuracy have made it a popular choice in the field of deep learning for image analysis.

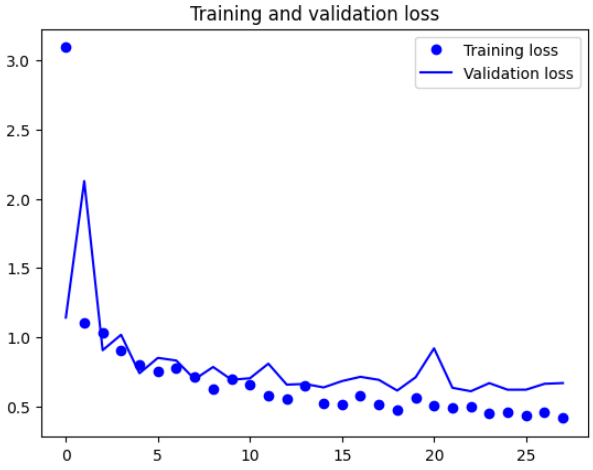
The workflow involved various steps, including flattening the network, conducting overfitting experiments, introducing dropout to address underfitting, exploring the use of Global Average Pooling, and finally applying fine-tuning techniques.

Feature Extraction

In the initial stages of our experiments, we embarked on exploring the capabilities of the pretrained ResNet50 model.
We started with a relatively simple network architecture, consisting of a Flatten layer followed by a single Dense layer with 256 neurons.

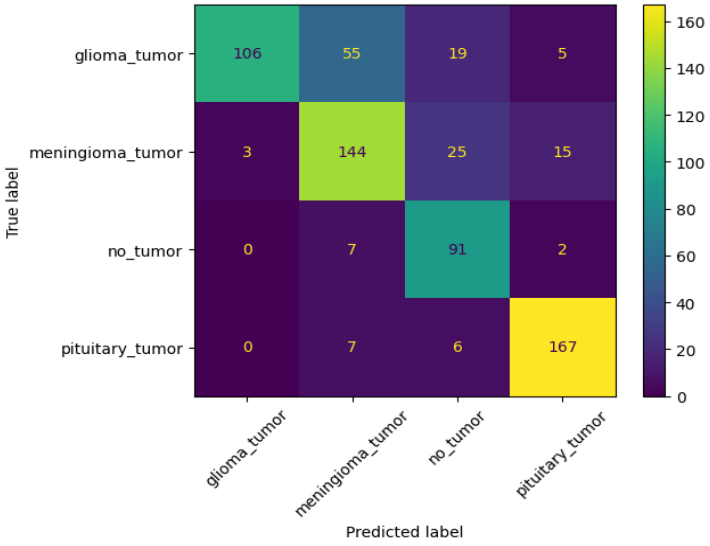
In our first experiment, we observed promising results, but they were not quite satisfactory. Upon analysing the outcomes, we identified that the model was experiencing a high loss value around 0.61, indicating a potential overfitting scenario.

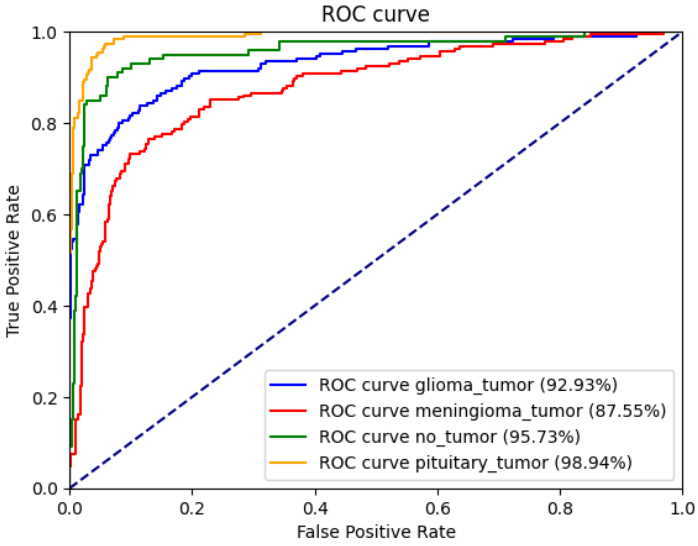




Classification report:

	precision	recall	f1-score	support
glioma_tumor	0.9725	0.5730	0.7211	185
meningioma_tumor	0.6761	0.7701	0.7200	187
no_tumor	0.6454	0.9100	0.7552	100
pituitary_tumor	0.8836	0.9278	0.9051	180
accuracy			0.7791	652
macro avg	0.7944	0.7952	0.7754	652
weighted avg	0.8128	0.7791	0.7768	652





This finding prompted us to investigate methods to address overfitting and enhance the model's generalization capabilities.

To mitigate overfitting, we introduced a Dropout layer to the network architecture. However, this modification led to an underfitting scenario, where the model struggled to capture the complexity

of the data effectively. The results obtained were loss around 1.3 and accuracy around 0.35 about training phase, loss around 1.2 and accuracy around 0.50 about validation phase.
We continued our exploration by incorporating more complex architectures.

In subsequent experiments, we experimented with various modifications to the architecture, such as adding additional layers and increasing the number of neurons. Despite these efforts, we did not achieve the desired results, and the model's performance remained suboptimal.

These initial experiments highlighted the importance of striking a balance between model complexity and generalization capabilities. It became clear that simply increasing the complexity of the architecture did not guarantee improved performance. We realized the need to explore alternative strategies and techniques to overcome the challenges of underfitting and optimize the model's performance.

We continued our experimentation by substituting the Flatten layer with the GlobalAveragePooling2D layer. This is a new technique that it condenses the spatial information from the previous convolutional layers by computing the average value of each feature map. This process reduces the dimensionality of the feature maps while preserving the essential information. By substituting the Flatten layer with the Global Average Pooling layer in our experiments, we aimed to condense the spatial information and potentially enhance the performance of the model by capturing the most important features while reducing the risk of overfitting.

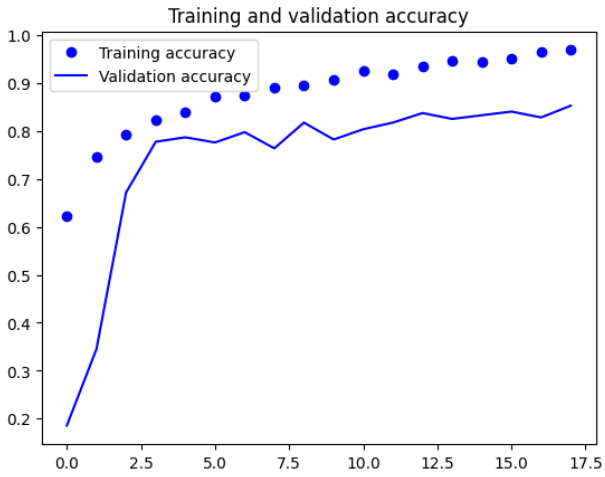
Unfortunately, we fall into a underfitting scenario and to address the underfitting problem, we made efforts to increase the complexity of our model. We experimented with various configurations, including adding additional layers and increasing the number of neurons. Despite these attempts, we were unable to completely resolve the underfitting problem and achieve the desired level of performance.

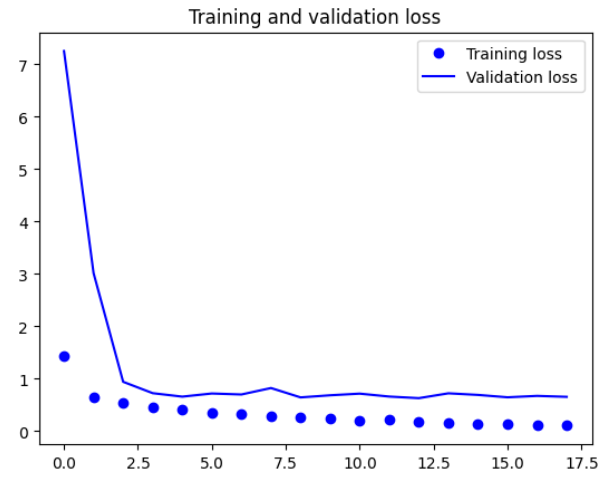
Fine-Tuning

Recognizing the limitations of our previous approaches, we decided to shift our strategy and explore fine-tuning techniques.
Fine tuning involves unlocking specific layers of the pretrained model to adapt it to our specific task.

After assessing the performance of our experiments, we decided to focus on the model that yielded the best results thus far, the one obtained with the first experiment, hoping to further enhance its performance. To achieve this, we began by unfreezing the last convolutional layer of the network. However, contrary to our expectations, the results were disappointing, indicating that this approach did not lead to the desired improvements.

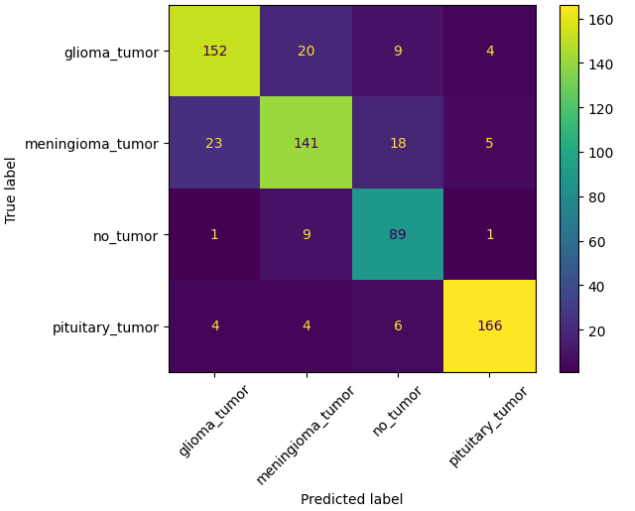
To boost performance, we turned our attention to the learning rate. By modifying its value, we observed promising results on the training set. Nevertheless, we remained unsatisfied with the validation loss, around 0.63, which continued to pose a challenge.

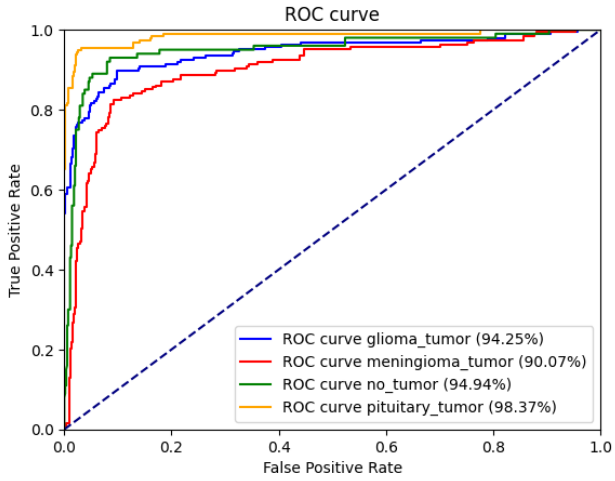




Classification report:

	precision	recall	f1-score	support
glioma_tumor	0.8444	0.8216	0.8329	185
meningioma_tumor	0.8103	0.7540	0.7812	187
no_tumor	0.7295	0.8900	0.8018	100
pituitary_tumor	0.9432	0.9222	0.9326	180
accuracy			0.8405	652
macro avg	0.8319	0.8470	0.8371	652
weighted avg	0.8443	0.8405	0.8408	652





To address this issue, we decided to explore alternative optimizers. By changing the optimizer, we aimed to optimize the model's performance and minimize the validation loss.
Additionally, we experimented with unfreezing and training additional layers, hoping to improve the model's ability to capture complex features. Despite these efforts, we were unable to achieve the desired level of performance.

As a final attempt to increase performance, we decided to remove the last block (block five) from the ResNet50 architecture. However, even with this modification, the results fell short of our expectations. We tried adding dense layers and unlocking the basic network at various levels, but despite this, the results did not improve.

Experiment	Training Loss	Validation Loss	Weighted F1
conv5_block3_1_conv	0.7365	0.8261	0.3748
conv5_block3_1_conv_lr	0.1767	0.6309	0.8408
conv5_block3_1_conv_rmsprop	0.3047	0.6391	0.8070
conv5_block1_1_conv	0.3274	0.6868	0.8386
Remove5block_exp6	1.1757	1.1503	0.5457
Remove5block_exp7	1.0438	0.9885	0.5808

Conclusion

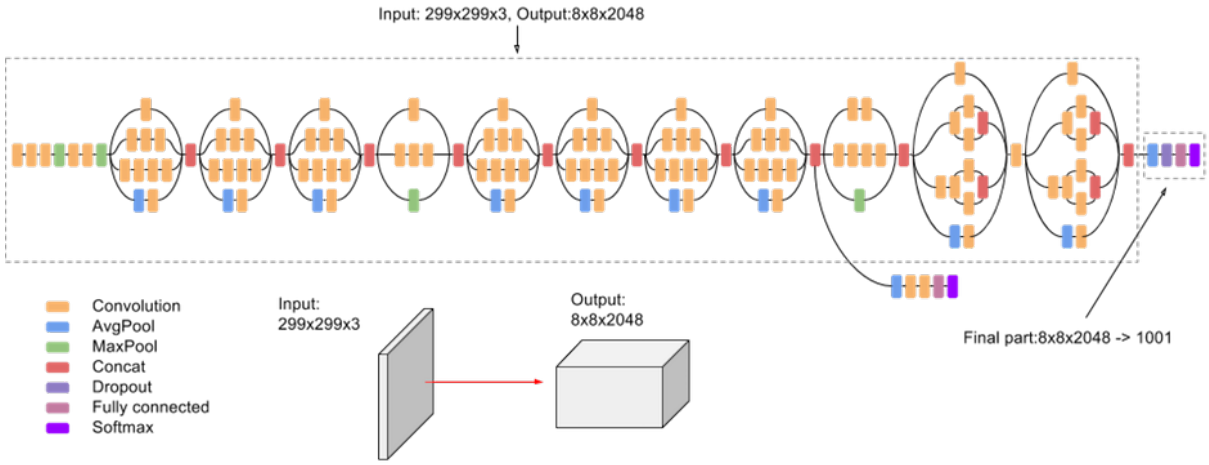
Despite our best efforts and various approaches, we were unable to fully resolve the performance limitations and achieve the desired level of accuracy with the pre-trained CNN ResNet-50 as we can see in the summary above.
Therefore, the best results obtained were those obtained thanks to the first experiment where we obtained a validation accuracy around 0.79, a validation loss around 0.61 and a F1-weighted around 0.77.

CLAHE Experiments

To further enhance the model, we experimented with pre-processing the dataset using the Contrast Limited Adaptive Histogram Equalization (CLAHE) method. This technique aims to enhance image features by increasing the contrast and highlighting important details.
Unfortunately, these attempts have produced improvements on the various experiments but without achieving remarkable results.
From the results obtained we have confirmed that the model with the best performance is created with the first experiment during the feature extraction phase.

InceptionV3

We decide to use the Inception V3 model as the basis for our feature extraction and fine-tuning experiments.



The Inception V3 model follows a deep convolutional neural network architecture. It consists of multiple layers, including convolutional layers, pooling layers, and fully connected layers.

At its core, the Inception V3 model utilizes the concept of inception modules. These modules are designed to capture features at different spatial scales by incorporating convolutional filters of varying sizes (1x1, 3x3, and 5x5) within the same layer. This allows the model to efficiently capture both local and global image features, enabling it to learn intricate patterns and structures present in the data.

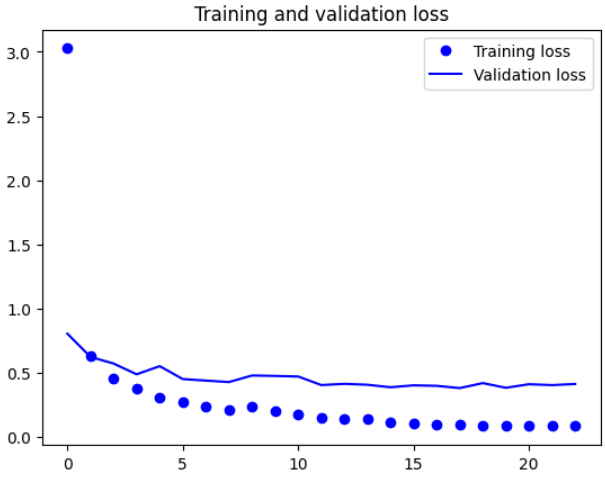
In addition to the inception modules, Inception V3 also employs auxiliary classifiers. These classifiers are inserted at intermediate layers of the network and are used during training to provide additional regularization. They help combat the vanishing gradient problem and encourage the model to learn more robust and discriminative features.

The Inception V3 model is pretrained on large-scale image classification datasets such as ImageNet, which enables it to leverage the knowledge gained from millions of images. This pretrained model can then be fine-tuned on specific tasks or domains, making it adaptable to a wide range of image classification problems.

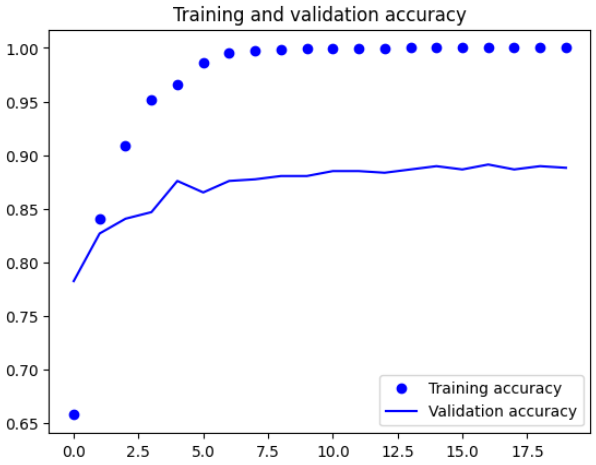
Feature Extraction

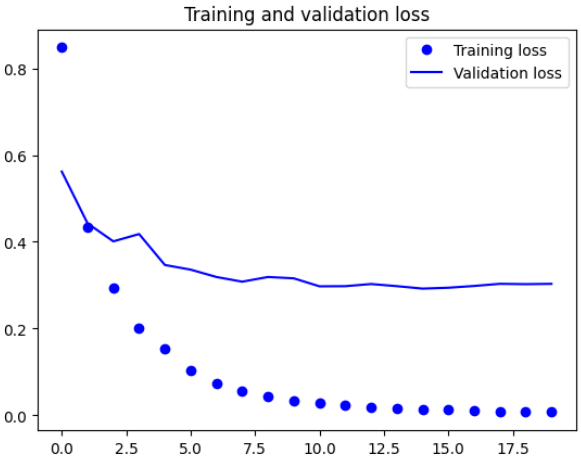
We started the feature extraction using the Inception V3 model as frozen base.

First, we added a dense layer with 128 neurons on top of the base model to further refine the extracted features. We observed that even with this first experiment the level of results was good except for the non-optimal validation loss around 0.38.



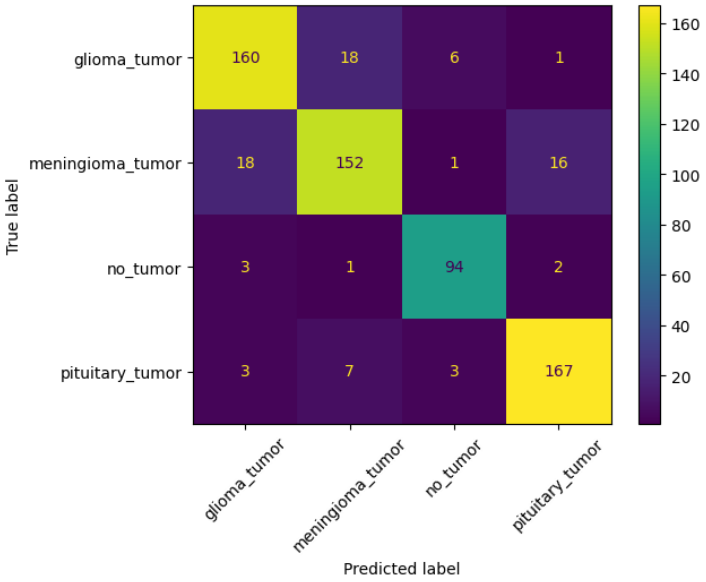
To improve the model training process, we decided to adjust the learning rate. By reducing the learning rate, we aimed to make minor updates to the model weights during each iteration, which could lead to better convergence and potentially improve validation loss.
This change led the second experiment to improve performance by substantially decreasing the validation loss around 0.29. Despite this, it can be seen from the graph that there is some overfitting due to the substantial difference in performance between the two loss curves.

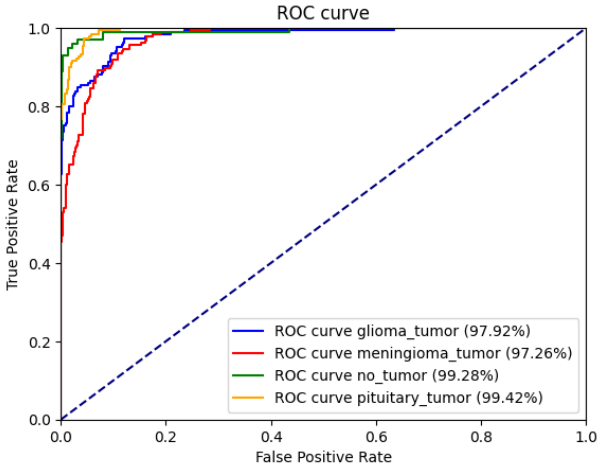




Classification report:

	precision	recall	f1-score	support
glioma_tumor	0.8696	0.8649	0.8672	185
meningioma_tumor	0.8539	0.8128	0.8329	187
no_tumor	0.9038	0.9400	0.9216	100
pituitary_tumor	0.8978	0.9278	0.9126	180
accuracy			0.8788	652
macro avg	0.8813	0.8864	0.8836	652
weighted avg	0.8781	0.8788	0.8782	652





In addition to the changes mentioned above, we've experimented with using the global average grouping instead of the flattened layer. Global average pooling reduces the spatial size of feature maps to a single value per channel, which can help capture more meaningful information and reduce overfitting. Unfortunately, it hasn't led to any noticeable improvements.

We next explored the impact of increasing the number of neurons or adding denser layers on top of the baseline model. The idea behind this change was to allow the model to learn more complex and expressive representations from the extracted features. However, validation loss remained high.

Fine-Tuning

Considering the limitations of the feature extraction approach, we proceeded to fine-tune the model obtained during the second experiment which was the best performer.
We started by unlocking mixed level number 9 without getting any improvements. Below we thawed the mixed level number 8 failing to improve the validation loss too much even though we had improvements in F1-weighted as we can see below.

Experiment	Training Loss	Validation Loss	Weighted F1
Mixed9	0.1917	0.3636	0.9122
Mixed8	0.1728	0.3033	0.9205

Conclusion

Despite the efforts made to tune the model, the validation loss has not decreased significantly from the value of 0.29 obtained with the second experiment during the Feature Extraction phase.
The best results obtained a validation accuracy around 0.88, a validation loss around 0.29 and a F1-weighted around 0.87.
This result suggests that due to dataset limitations additional data would be needed for further improvement.

CLAHE Experiments

To further improve the model results, we experimented with the CLAHE method, explained above. Unfortunately, the results obtained were not better, the improvement of the image characteristics was not enough.

Explainability

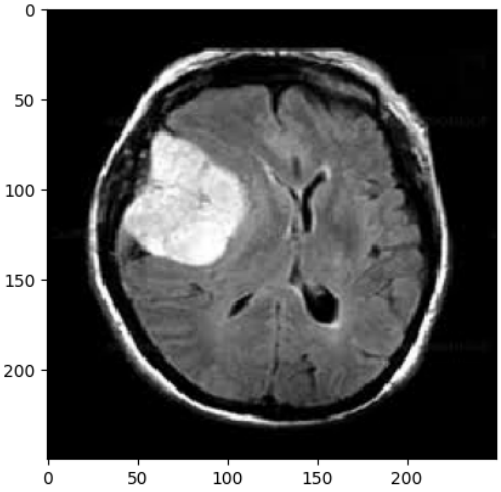
Explainability in CNNs refers to the ability to understand and interpret the inner workings of these complex models. While CNNs have demonstrated remarkable performance in various tasks, their decision-making processes often remain black-boxed and difficult to comprehend. Explainability techniques aim to shed light on how CNNs arrive at their predictions by visualizing and analyzing for instance intermediate activations and heatmaps.

In the context of medical imaging, explainability becomes particularly crucial for ensuring trust and understanding in the predictions made by CNNs. By examining the intermediate activations of CNNs, we can explore how the network progressively extracts and transforms features from the input image. This allows us to identify the layers where the network starts encoding higher-level concepts relevant to the task at hand, such as tumor detection.

In addition to intermediate activations, heatmaps provide a powerful tool for explainability in CNNs. By generating heatmaps, we can highlight the regions of an image that contribute the most to the final prediction. For instance, in the case of glioma tumor detection, the heatmap can highlight the areas that exhibit strong indications of a tumor, aiding in the diagnosis and treatment planning process.

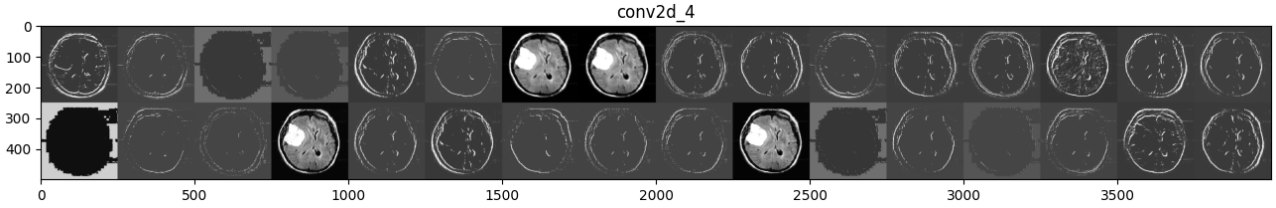
Intermediate Activations

In this section, we examine the activation of certain layers in CNNs, both from scratch and pretrained models, in response to the provided image depicting a meningioma tumor.

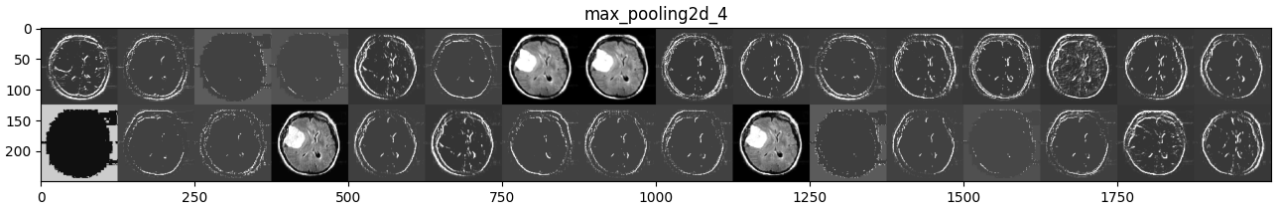


CNN from scratch

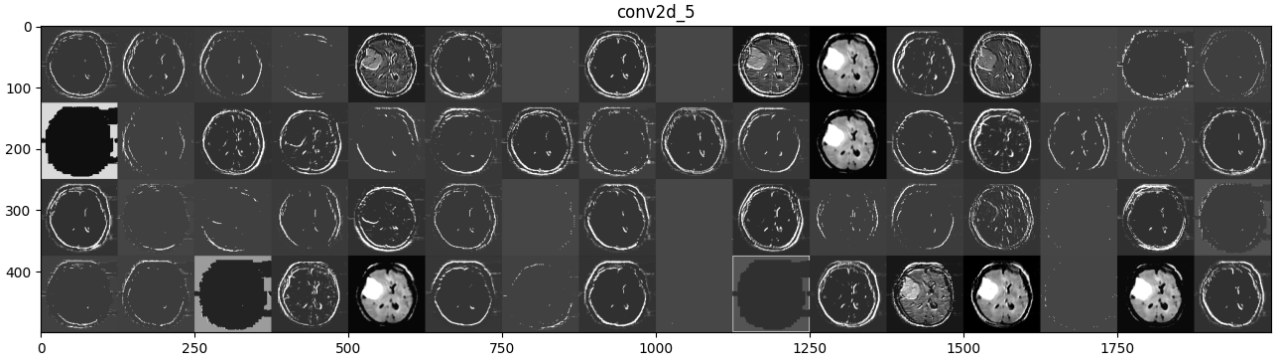
The CNN taken in consideration is the one of the third experiment with 4 conv/maxpooling layers, the one that archive the best results on the test set.



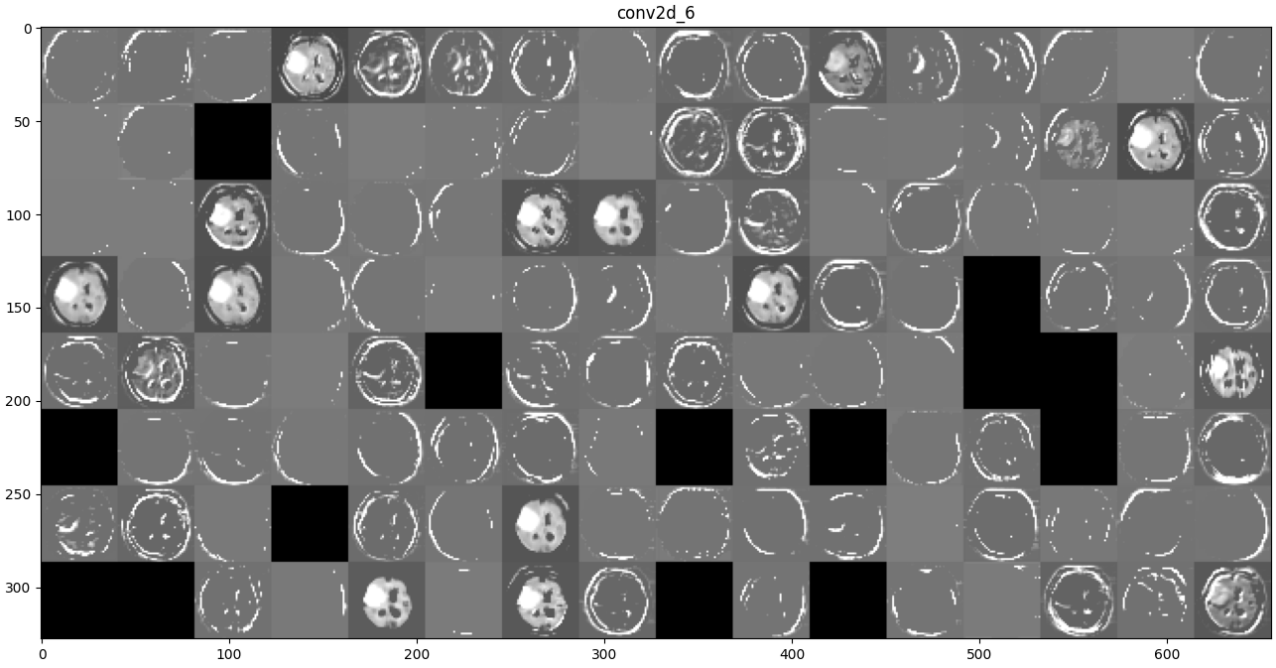
In the initial layer of the neural network, there is a set of different filters designed to detect edges in the input image. During this stage, the activations of these filters preserve nearly all the information that is contained within the original picture.



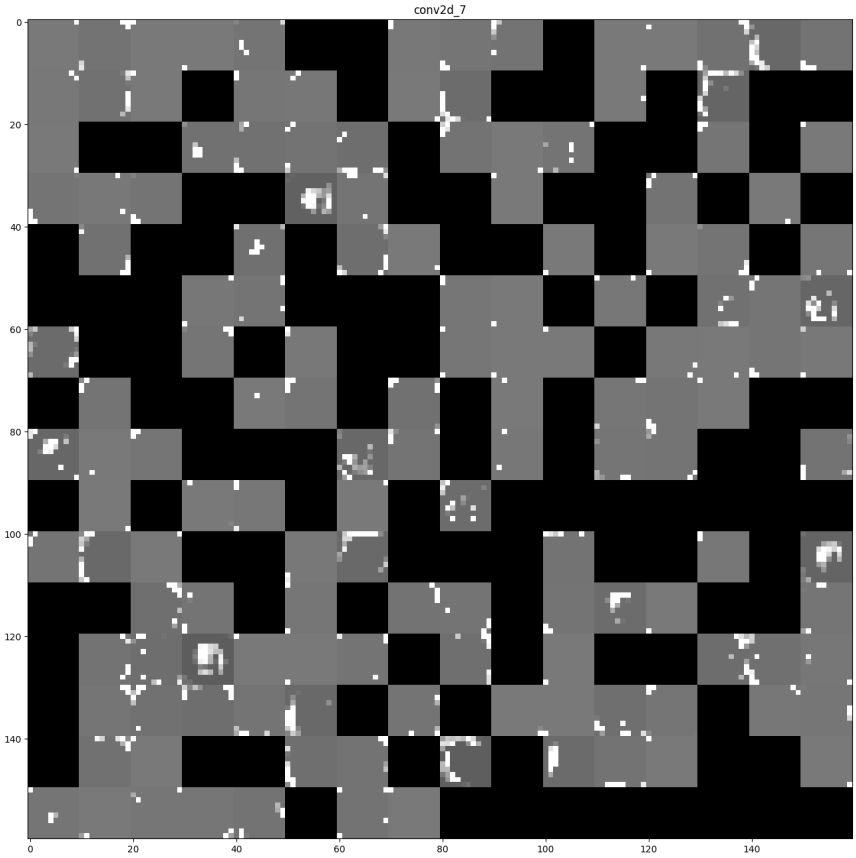
The maxpooling layer at this level tends to enhance the clarity of the image and consequently make the edges more defined.



Certain filters are designed to identify specific patterns and orientations within the input image. For example, some of them may highlight horizontal lines or edges, while others focus on vertical or diagonal lines.

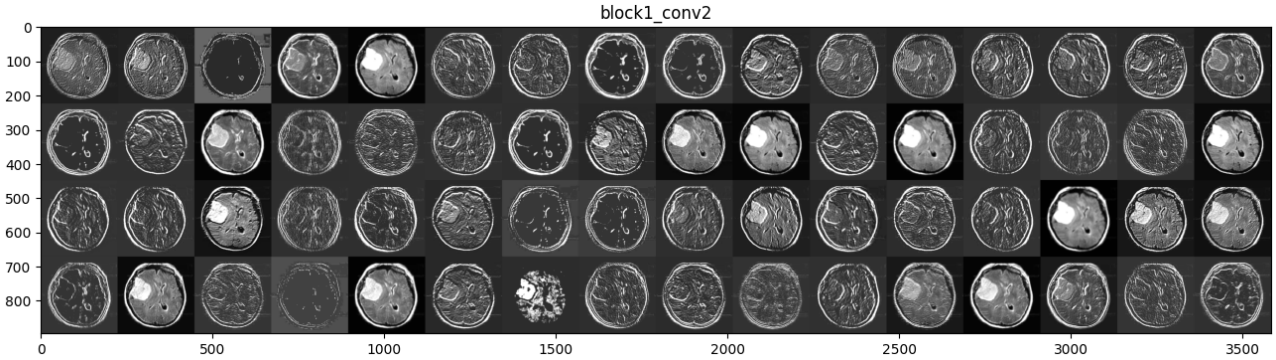


Other filters highlight features related to the brain, such as tumor regions or specific anatomical structures, meanwhile, other filters emphasize the background of the image.



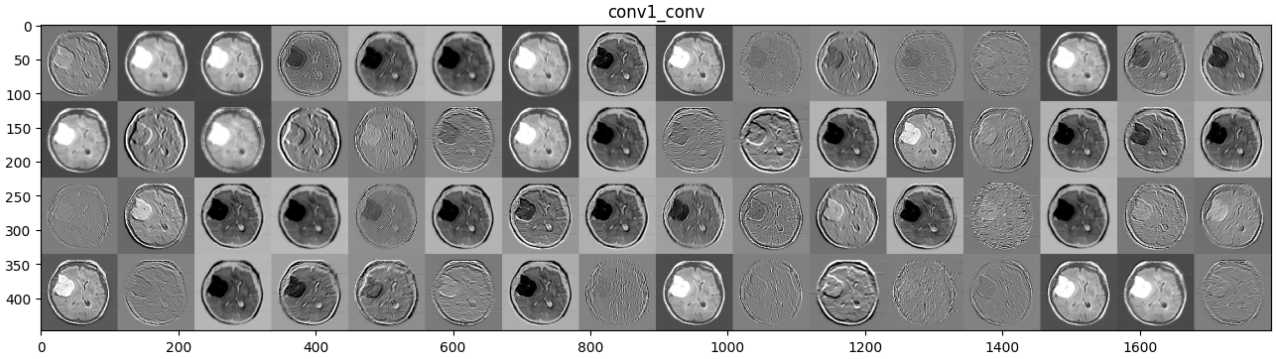
As we progress to higher layers of the neural network, the activations become more abstract and less visually comprehensible. At these stages, they begin to encode higher-level concepts such as brain symmetry, the presence of an abnormal mass, or the roundness of the skull.

VGG16

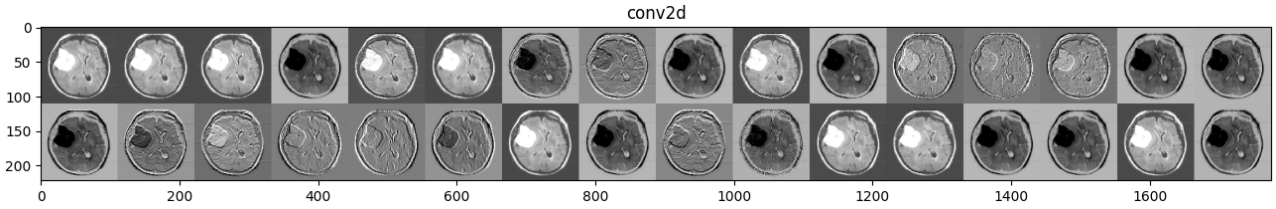


Regarding the VGG16 model, we observe that in the early layers, it immediately focuses on internal cranial features, extracting various pieces of information from them.

ResNet50

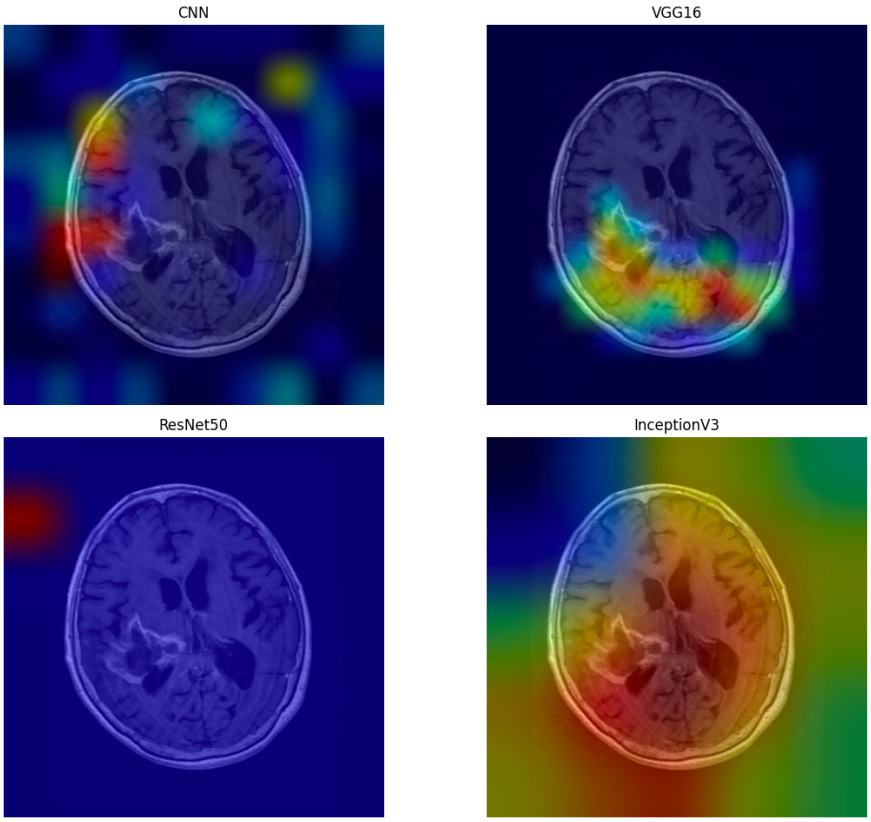


InceptionV3



For both the ResNet50 and InceptionV3 models we observe that the filters used in the early layers rapidly identify the presence of the tumor by highlighting it in either white or black, creating contrast with the rest of the MRI image.

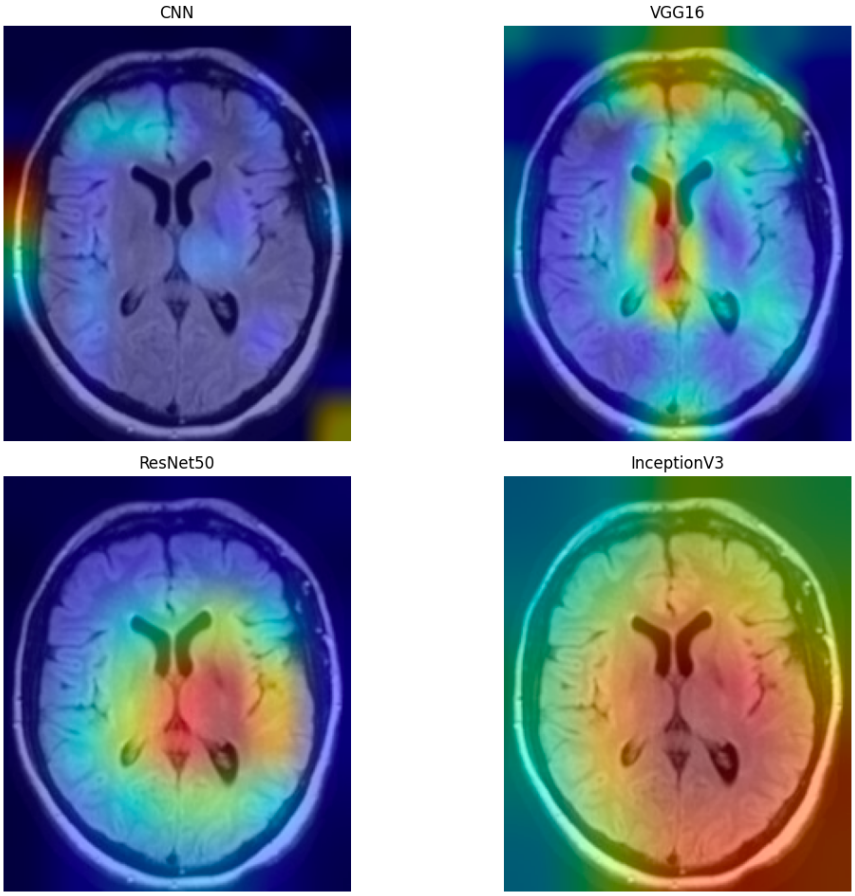
Heatmap for Glioma Tumor



From the heatmap analysis of the four models, specifically regarding the Glioma, we observe that the CNN from scratch is able to reasonably detect the tumor area. Similarly, the VGG16 model

demonstrates excellent results by activating the region affected by the tumor, although not at maximum intensity.
However, the ResNet model struggles to identify the region of interest, which may be attributed to lower performance of the model. On the other hand, the InceptionV3 model, which performs well overall, tends to highlight the entire image, with the hottest spot being the area where the tumor is present.

Heatmap for No Tumor



Regarding the image without a tumor, we observe that the CNN from scratch focuses on the edge of the skull, without highlighting any particular features. On the other hand, the pretrained networks are able to emphasize the central part of the brain, which typically appears asymmetric in the presence of a tumor.
This suggests that the CNN from scratch lacks the ability to discern subtle differences in brain structure, while the pretrained networks have learned to recognize patterns associated with tumor presence or absence.

Conclusions

In this scientific report, we presented and compared the results obtained from experiments conducted using a CNN from scratch and pre-trained models for classifying MRI into four categories. The best performance on the test set was achieved by VGG16 on the Clahe preprocessed dataset, obtaining a weighted F1 score of 0.93. However, both CNN from scratch and Inception also achieved an F1 score of 0.92 on the original dataset. Possible improvements could be obtained with a larger and more balanced dataset, particularly regarding the minority class, which is the "no tumor" class. We are still far from the optimal solution, and potential enhancements can be achieved by utilizing hyperparameter optimization, ensemble methods, image denoising techniques like Gaussian filtering or Laplacian filtering, that can reduce noise and improve the visibility of tumor structures, and more recent pre-trained neural networks such as EfficientNet.