Modular Neural Networks for detection and classification of brain cancer images

Davide Macolo[1]*, Leonardo Plini[1], Margaret Antonicelli[1]

[1] University of Rome, La Sapienza
mascolo.2001991@studenti.uniroma1.it, plini.2000543@studenti.uniroma1.it, margaret.antonicelli@uniroma1.it

Abstract: Among the various oncological pathologies, brain cancer continues to be one of the most wide-spread, as well as lethal, diseases. Within this paper we extended a previous work leveraging the early exits strategies to reduce the computational time needed to detect and classify brain tumor MRI images namely say – glioma, meningioma, and pituitary. The dataset we used consisted of 3264 2-D MRI images and 4 classes. Due to the small number of images, various data augmentation techniques were used to increase the size of the dataset. Our proposed methodology consists not only of data augmentation, but also of various techniques of image denoising, skull strip-ping, cropping and bias correction. In our working proposal, we implemented a custom CNN and we explored both a static and dynamic solution of Early Exit procedure to achieve the smallest computational effort. The purpose of this document is to distinguish between normal and abnormal pixels and classify them more rapidly but with a comparable level of accuracy.

Keywords: brain cancer, custom CNN model, detection.

1. Introduction

A brain tumor refers to an atypical cell overgrowth within brain tissue which can be categorized as either benign or malignant depending on its capacity to invade other areas of the body. The central nervous system, comprising the brain and spinal cord, plays a crucial role in coordinating various bodily functions acting on

* Corresponding Author: mascolo.2001991@studenti.uniroma1.it

both voluntary functions such as walking or talking and involuntary functions like breathing or digesting. Additionally, the central nervous system serves as the foundation for sensory functions encompassing sight, smell, touch, hearing, and taste and it plays a pivotal role in governing emotions and facilitating the so-called higher-order activities such as memory and learning.

Typical, the process of diagnosing a brain tumor starts when the patient visit the general practitioner due to the manifestation of symptoms. Subsequently, the doctor assesses whether further investigation with a neurology special ist is necessary or prescribes instrumental diagnostic tests. Nevertheless, in some cases symptoms may arise abruptly and unexpectedly, requiring a prompt evaluation by the emergency department for urgent medical attention. When there is suspicion of a brain tumor, magnetic resonance imaging (MRI) serves as the primary diagnostic test. When compared to CT scans, MRI, combined with or without the utilization of paramagnetic contrast medium (gadolin ium), offers several advantages. In fact, it facilitates the identification of lesions and nodules, it generates detailed three-dimensional (3D) images, and it provides comprehensive information about the tumor's location, dimensions, spread as well as its relationship with surrounding structures, including critical regions known as "eloquent" areas. Furthermore, functional MRI techniques like diffusion and perfusion provide additional valuable information regard ing the cellular composition and vascularization of the analyzed regions. Based on 3264 2-D MRI images, advanced analysis models will be evaluated to understand the performance and the peculiarities of Early-Exits techniques with the aim of distinguishing between normal and abnormal pixels and classifying them more rapidly with a tolerable high level of accuracy.

2. Related works

Ranit Sen, Gopinath Balaji and Harsh Kirty [Ranit Sen et al., 2022] use several pretrained networks reaching about the same performance with the Efficient Net B0 network, but they do not use any network from scratch. In paper [Abiwinanda et al., 2019], the authors use different CNN architectures, obtaining however a validation accuracy far enough from the training accuracy.

In [Chollet, 2017] the authors used a machine learning approach with a SVM classifier but the main disadvantage of this procedure is the inability to extract interesting features for classification. For what concerns the Early-Exits strategy, the work provided in [Scardapane, S., Comminiello, D., Scarpiniti, M.,Baccarelli, E.

and Uncini, A., 2020] and [Jang, E., Gu, S. and Poole, B., 2017] serve as guiding inspiration to understand the advantages of this approach. This methodology could be very useful in the healthcare sector due to the large number of images available through the entire diagnostic procedure and the time required to analyze them.

3. Method

Deep neural networks (DNNs) have emerged as the leading techniques for various tasks, thanks to their capability to extract features during the learning process and to the possibility of optimizing the network parameters in order to minimize the error. Recently, DNNs have demonstrated excellent performance in analyzing medical images com pared to traditional machine learning methods [Bauer et a., 2013].

Nevertheless, the enhanced performance achieved by additional layers in a deep network comes at the expense of increased latency and energy consumption during training and inference phase. As networks grow deeper and larger, these costs become increasingly challenging to manage, particularly in real-time and energy-sensitive applications creating possible logistic difficulties during the distribution of the application. The standard configuration of deep neural networks involves a series of differentiable layers, yielding predictions only at the final stage. Researchers have proposed the integration of Early Exits within these networks, enabling predictions to be acquired at interme diate points. Multi-output networks offer numerous advantages, such as (i) substantial inference time reduction, (ii) decreased likelihood of over-fitting and vanishing gradients. For this reason, in this work we proposed a custom CNN using both a statistic and dynamic Early-Exits solutions.

The typical pipeline of an image-based cancer classification study using a CNN is shown in Figure 1.

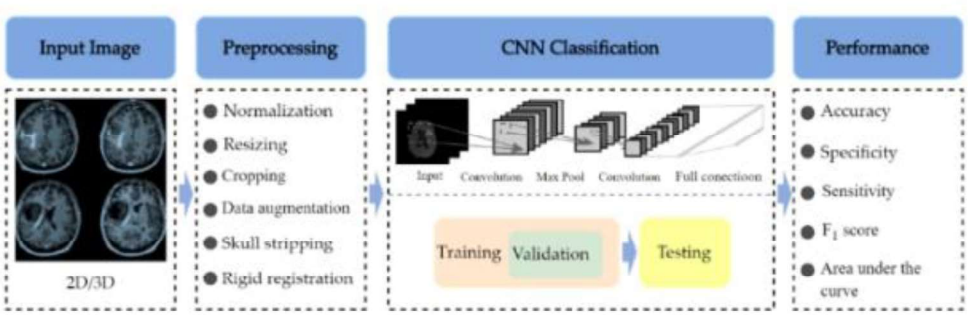

Figure 1: Pipeline of CNNs-based Brain Cancer Classification

3.1 Pre-processing

As a first operation, the dataset was split into Training Set and Test Set. Subsequently the preprocessing operations were applied only to the Training data and during the training phase of the models, 10% of the Training data was used as Validation Set for parameters tuning. Fig. 2 shows the data split.

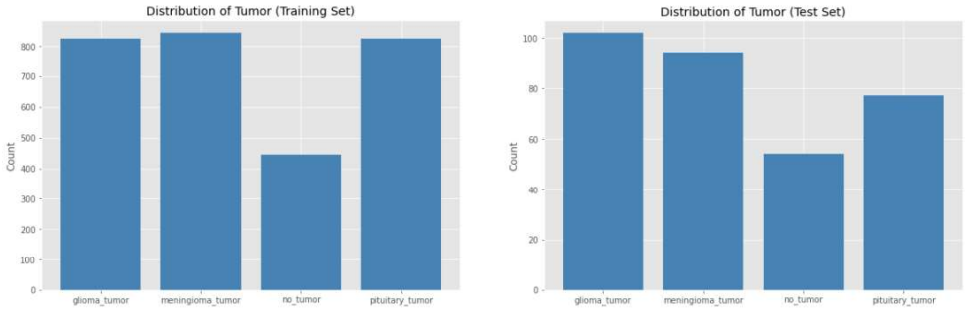

Figure 2: Distribution of classes for Training Set and Test Set

3.1.1 Pre-processing

1. Resize: The first pre-processing operation applied was to resize the image to a size of (150, 150).
2. Noise Reduction: To change the resolution of the original images, a Gaussian filter has been applied to reduce noise and blur the images. In Fig. 3 it's possible to

see an example of an image filtered.

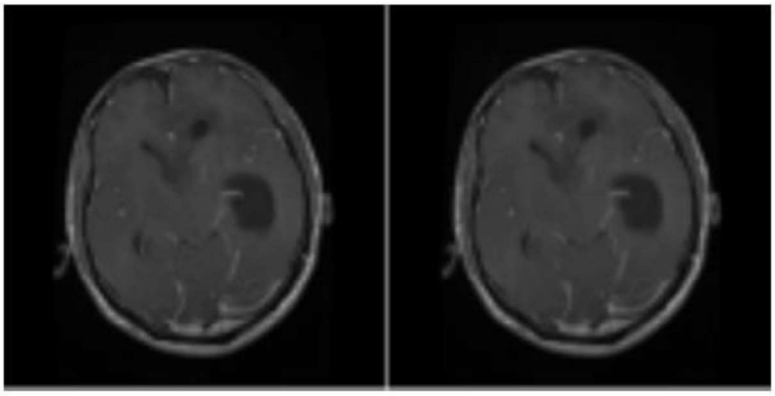

Figure 3: Original Image and Filtered Image with Gaussian Method

3. Data Augmentation: To make the proposed methods more robust and reduce the over-fitting effect, Data Aug mentation was applied with various transformations, in particular:

- rescaling
- rotation with an angle of 30 degrees
- vertical and horizontal shift
- image zoom
- horizontal flipping

In Fig. 4 it is possible to observe the example of an original image and the respective transformation.

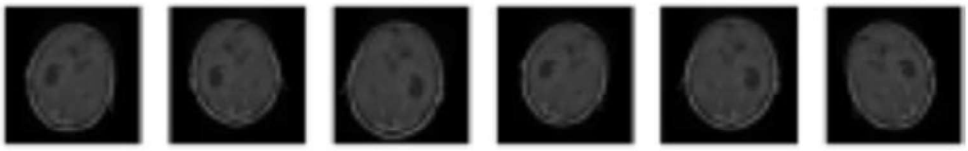

Figure 4: Example of Original Image and Respective Transformations

4. One-Hot Encode: The last operation carried out was the transformation of the reference variable which was originally coded as numeric into a categorical variable.

3.2 Models

We chose to use a CNN architecture over any other model because CNN employs several convolutional filters to scan the whole feature matrix and perform dimensionality reduction, hence making them well suited for image processing and classification.

3.3 Baseline CNN Model

We used different parameters to train the model in the best possible way. To speed up the training a batch size of 64 was used and the models were trained for 40 epochs. The optimization algorithm chosen to train the models is Adam because it is computationally efficient, easy to implement and works well for problems that involve large amounts of data or parameters. The learning rate was chosen in a dynamic way, starting from a value of 1e-3 and monitoring the validation loss: if the loss does not decrease by at least 0.2 for two consecutive epochs, the learning rate is reduced of a factor of 0.3. The loss function chosen is the Categorical Cross-Entropy which is used to calculate and minimize the error of the model during the optimization process. This is a typical choice for a multi-class classification model. The loss is defined by:

$$L = \sum_{i=1}^{\text{output size}} y_i \cdot log(\hat{y}_i)$$

To make a fair comparison between the models, we fixed the number of epochs to 40 and we evaluated both the performances and the dynamics of the models at the end of the training. For the CNN implemented from scratch, in addition to the above configuration, several techniques have been used to improve their performance. The model contains 6 layers including input and output layers and 2 Early Exits. Each layer consists of 2D convolution, Max Pooling 2D, Batch Normalization and Dropout operation. The final layer (Fully Connected Layer) is composed of 1024 neurons connected to the 4 output neurons (one for each class to be provided) and managed by the softmax function. The activation function chosen is the ReLu function. For the dropout operation, the value of p is 0.25 and 0.4 for the output layer. For performance evaluation, the following metrics were used: Accuracy, Precision, Recall and F1-Score.

3.4. The dimensions for a Frugal Society and Sustainable Value

Consider the output of a CNN model defined as fe(x) for some layer e. An Early Exit can be considered as a small classifier added on top of it:

$$\hat{y}_e = c_e(f_e(x))$$

The advantage of this approach is that the neural network provides a sequence of predictions $\hat{y}_1$, $\hat{y}_2$,..., one for each Early Exit, that can achieve different values of accuracy, in addition to the final classification $\hat{y}$. This approach presents three major problems:

1. Determine the optimal placement of an Early Exit. A solution could be to calculate the relative floating-point operations per second (FLOPS) after each computational block and strategically incorporate Early Exits at specific percentages of the total compute cost. For instance, we could add exits after 25%,50%, and 75% of the total compute cost of the model. Another alternative is to consider the amount of time needed during the inference phase.
2. How can we train a model that incorporates multiple exits for each input?
3. During the inference phase, how do we determine whether to exit the model at a specific point or proceed to the next available exit?

In order to make a joint training of the exits, consider a pair (x, y), there are E early predictions $\hat{y}_1$,..., $\hat{y}_E$ and a final prediction $\hat{y}$. It's useful to define a joint loss by adding all the losses:

$$Total\ loss = CE(\hat{y}, y) + \sum_e \lambda_e \cdot CE(\hat{y_e}, y)$$

where CE($\hat{y}$, y) is the loss of Early Exit e and the weights λe can be set to 1 or become hyper-parameters. During the inference time, considering pe as the confidence associated to Early Exit e. We can decide to stop the procedure if pe>γ, where γ is a threshold decided by the user, assuming the exits are well calibrated. In a different way, it is possible to consider the entropy H[ye] of the predictions, measuring a broader level of uncertainty on the prediction:

$$H[y] = -\sum_c y_c \cdot \log(y_c) < \gamma$$

3.5 Dynamic Early-Exit Model

The previous schema provide a general strategy to build an early exit. However, it does not allow the model learn where to exit defining the exit strategy in a static way.

To overcome this drawback, we can insert a separate block, called ”Exit Selector”, that chooses the best exit for each input considering its confidence level. Basically, the Exit Selector is a classifier that outputs the confidence related to each exit for a given input. If we refer to the confidence at exit e with c_e, we might train the network considering an aggregate prediction weighted with the relative confidence:

$$\tilde{y} = \sum_e softmax(c_e)\hat{y}_e, \quad \hat{y}_e = \text{prediction at exit e}$$

Our intention is to force the exit probabilities to be similar to some pre-defined probabilities by leveraging the KL-divergence and the exit selector is responsible for deciding the exit at inference time. The difficulty is to define a compatible approach both during training and testing since it is not possible to consider an argmax operation during training without occuring into gradient problems.

The solution to this issue is provided by the Straigh-Through Estimator that allows to consider the argmax during the forward pass and the original probabilities during the backward pass, letting the gradient flow only through the components that gave a contribution to the predicted exit. Nevertheless, this strategy can lead the network to never explore different exits, trapped into the initial prediction. The last element we need to introduce to face this problem is a differentiable sampling mechanism that provides probabilities p_e to sample as exit at training time. In practice, given the probabilities p_e, we sample from the categorical distribution $z \sim Cat(p_e)$. By using the soft selection strategy over the Gumbel softmax trick, we obtain the following formulation:

$$z = softmax\left(\frac{log(p_e) + g}{\tau}\right)$$

where τ is a temperature scale parameter that regulates how the softmax should approximate the argmax.

4. Result and discussion

We used different curves (Fig. 5) in order to evaluate the metrics of interest. It is important to remember that, in the specific case of MRI cancer images, we want to maximize the number of True Positive which means correctly classifying patients with the disease and minimizing False Negative, i.e. reducing the risk of not treating patients who are actually ill. Furthermore, by applying the Early-Exit solution, we want also to check the training time and achieve the best trade-off between training time and performance. Let's start evaluating the benchmark CNN model implemented without any Early Exit approach.

We can see in the Figure 5 the behaviour of the model with respect to the training and validation set. As expected, the training loss decreases very quickly while the validation loss has a certain amount of variability but after 15 epochs it decreases. Looking the other plots, we can see a slight overfitting just after 30 epochs and in general the performances are very good. In order to understand if the model is able to generalize on unseen data, we evaluate it using test set. In Table 1 is possible to see the performances of the model using the test set.

The NN has good performances overall considering the metrics that are in line with the results found using the validation set. In Table 2, the training time and the inference time are reported.

Figure 5: Performances Training and Validation data

Model	Loss	Accuracy	Precision	Recall	F1-Score
CNN (Benchmark)	0.534	0.773	0.831	0.762	0.795

Table 1: Performance Test data

Starting from this point, we discuss the results of the Static Early Exit Model which has a structure similar to the previous model. However, two early exits were includes after the second and the fourth layer as reported in Figure 6.

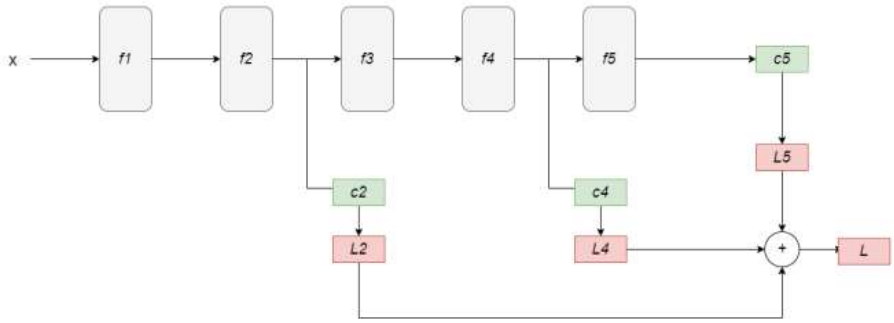

Figure 6: Early Exit Model Architecture, image inspired by Simone Scardapane's lectures

In this case, a threshold procedure is used to decide the exit in a static way and this formulation does not create any problem in the gradient flow. In Figure 7, we reported the performances obtained on test set using the model with Static Early Exit.

For this model, we keep trace also of the early exits results and the second early exit provides very good results.

However, this cannot be consider a general conclusion since we decided the threshold in a static way.

In Table 3 is possible to see the performances of the model on the test set that are in line with the previous model.

The time required for the test set is negligible, but we should consider that on large datasets this procedure could save a lot of time. In Table 4, the training time and the time during the inference phase are reported. It is worth noticing that in this case, the inference time can be bigger than the one seen for the baseline model. Even if this results might appear inconsistent, in really it makes sense. In fact, we are performing a bigger number of computations with respect to the baseline model given that it is not possible to distinguish a priori the exits for each input which makes the pipeline more sophisticated.

Model	**Training Time (sec.)**	**Inference Time (sec.)**
CNN (Benchmark)	801.275	0.267

Table 2: Training and Inference Time

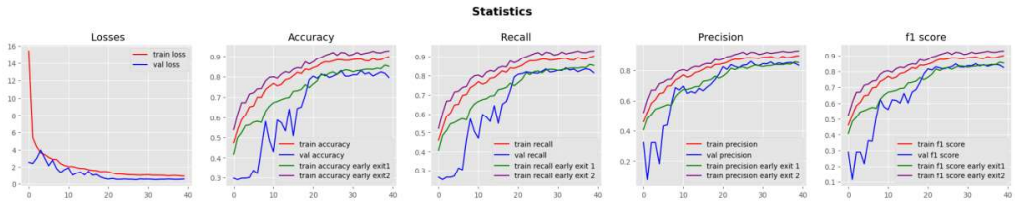

Figure 7: Performance Training and Validation data Early Exit CNN

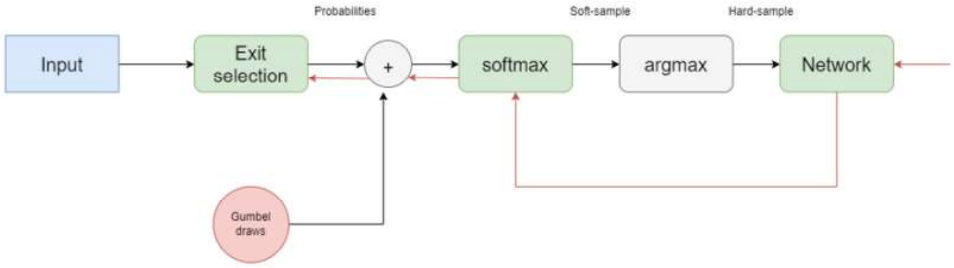

Figure 8: Gumbel Softmax Architecture, image inspired by Simone Scardapane's lectures

The second block is composed by the network itself (represented as the green network in the Figure 8) which takes the output of the early after the Gumbel Softmax Trick and the images. The model was implemented using masks and a matrix M that is initialized with zeros to store all the output in the correct position. This seems the most appropriate approach after having experimented with einsum and other methods that do not exploit the early exits intuition to its maximum. In Figure 9, we reported the performances obtained on test set using the model with Gumbel Softmax approach.

It is possible to see that the model did not reach an overfitting stage and all the metrics seem to have an increasing behavior. However, to obtain consistent results with the previous models, we decided to reach 40 epochs and then stop the training. In Table 5 are reported the results on test data and the percentages of samples exited for each Early

Exit. In Table 6, the training and inference time of the model are reported. It is important to note that the training time is considerably smaller than the one seen in the Static Early Exit. In fact, in this case, it is possible to extract the exits a priori due to the presence of the Exit Selector and this both simplifies the computations and reduces the time needed.

Model	Loss	Accuracy	Precision	Recall	F1-Score
CNN (Benchmark)	0.534	0.773	0.831	0.762	0.795

Table 3: Performance Test data

Model	Training Time (sec.)	Inference Time (sec.)
CNN (Benchmark)	801.275	0.267
CNN - Static Early Exit	2570.185	36.336

Table 4: Training and Inference Time

Figure 9: Performance Gumbel Softmax Model

Model	Loss	Accuracy	% samples in Exit 1	% samples in Exit 2	% samples in Exit 3
CNN - Gumbell Softmax	0.612	0.761	0.218	0.437	0.345

Table 5: Performance on Validation data

Model	Training Time (sec.)	Inference Time (sec.)
CNN - Gumbell Softmax	1147.030	1.079

Table 6: Training and Inference Time - Gumbell Softmax

5. Conclusion and future work

For both the Static and Dynamic Early Exit the computational time is longer. However, the benefits should be considerable in case of bigger datasets and models composed of several layers where deciding if one should exit early can save a good amount of time. In particular, the Static formulation forces the model to perform a large number of computation that can eventually lead to bigger computational time

in case of small model, like the one proposed in this paper, reducing the effectiveness of intuition behind the Early Exit. On the other hand, the dynamic formulation allows for a flexible decision process that effectively reduces the test time and, for this reason, it should be the preferred option given the results.

The purpose of this paper is to provide a comprehensive application to the world of neural networks featuring multiple early exits. We emphasize the advantages of these models, such as their heightened resilience to gradient issues and adaptability to advanced training and inference techniques. Furthermore, they offer a viable pathway to exploring alternatives to conventional back-propagation algorithms. Of course, there are different open questions. While these techniques have been applied to some extent in CNNs and RNNs, their effectiveness in modern neural layers like graph convolutional networks and transformers requires exploration. Moreover, implementing these networks in a distributed environment presents open questions regarding architectural constraints, accuracy and communication costs.

References

Nyoman Abiwinanda et al. ” Brain tumor classification using convolutional neural network”. World congress on medical physics and biomedi- cal engineering 2018, pages 183–189, 2019.

Bakhtiarnia, A., Zhang, Q. and Iosifidis, A., 2021. Multi-exit vision transformer for dynamic inference. arXiv.

Stefan Bauer et a. ”A survey of mri-based medical image analysis for brain tumor studies”. Physics in Medicine Biology, page 58.13, 2013.

Francois Chollet. ”Xception: Deep learning with depthwise separable convolutions”. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1251–1258, 2017.

Jang, E., Gu, S. and Poole, B., 2017. Categorical Reparametrization with Gumbel-Softmax. In ICLR 2017

Ranit Sen et al.” Detection and classification of brain tumors using deep convolutional neural networks”. 2022.

Scardapane, S., Scarpiniti, M., Baccarelli, E. and Uncini, A., 2020. Why should we add early exits to neural networks?. Cognitive Computation.

Scardapane, S., Comminiello, D., Scarpiniti, M.,Baccarelli, E. and Uncini, A., 2020. Differentiable branching in deep networks for fast inference. In IEEE ICASSP 2020.