Convolutional Neural Network for semantic segmentation in tumor detection on brain MRI

Massetti Matteo
m.massetti2@studenti.unipi.it

Date: March 2022

1 Introduction

This project will explore the use of *Convolutional Neural Network* for image processing, in particular of brain magnetic resonance imaging.
When we are talking about deep learning applied in the Image domain, we refer to a several tasks:

- *Classification*, classify the whole image in one (or more) of the given classes
- *Object Detection*, identify a region where a specific object is located in the image
- *Image Segmentation*, classify each pixel, assigning each one a label indicating the class to which it belongs (in this case it is called *Semantic Segmentation*), or a label distinguishing between the different entities present in the image (this is called *Instance Segmentation*).

The problem I decided to tackle is *Semantic Segmentation*, so the problem labeling each pixel of an image with a corresponding class of what is being represented, a simple example is reported in figure 1 where each pixel of the image is classified in one of three classes: Person, Bicycle and Background.

1.1 Task Formalization

The aim of this project is to build a model for the Binary Semantic Segmentation, i.e. that is capable of labelling each pixel of the MRI as *Background* or *Tumor*. The desired result is shown in the figure 2, where on the left there is the original image and on the right the segmented one.

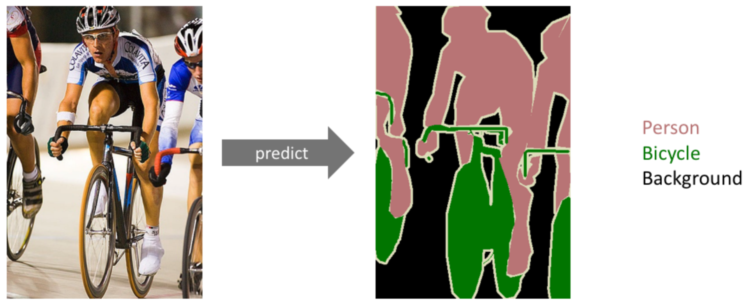


Figure 1: Semantic Segmentation Example

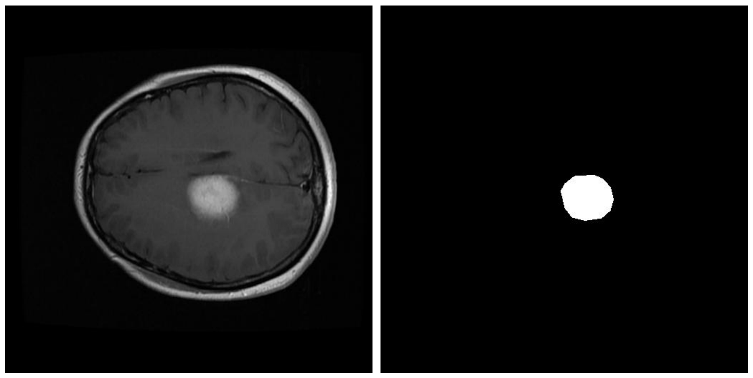


Figure 2: a) Magnetic Resonance Imaging b) Tumor Mask

2 Dataset

The Dataset [1] used for the study is formed by 3064 mat files that contains both the information about the MRI and a mask that identify the tumor.
All but a dozen images have a size of 512x512 pixels with a single colour channel, so the value of each pixel represents intensity in a grey scale.
Every mat file was saved into two distinct jpg file, one for the Brain Image and the other for the mask.

Some preprocessing was necessary, first of all the subsequent experiments were done with a scaled version of the images, i.e. with size 256x256 pixels and 128x128 pixels. As it possible to see in figure 3, even the lowest resolution image allows correct distinction between the brain and the tumour.

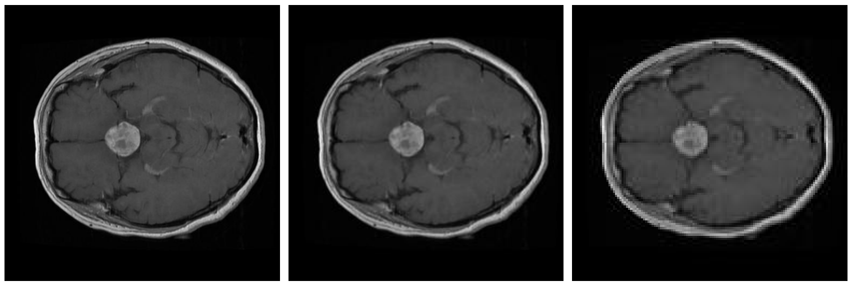


Figure 3: Brain image at different resolution

In addition to this, for the masks it was also necessary to normalise the pixels, since pixel intensity values were in the range $(0, 255)$ and few of them were neither 0 nor 255, contrary to what one would expect. To transform these values in a real binary mask, I simply used a threshold to set all values to 0 or 255 and then divided all the values by 255 so that I had the two correct labels.

For the following experiments the dataset was divided into Test set and Training set, reserving for the latter about 90% of the images in the dataset. The test set will be used only to verify the capacities of the final model, while the training set will be further divided in training set (about 90 %) and validation set (remaining 10 %), this last one will be used in the gridsearch to evaluate the different configurations of the models.

3 Model

The classical Convolutional Neural Network architecture is formed by a certain number of Convolutional and Pooling layers followed by one or more dense layers that output a vector, this particular structure does not allow pixelwise classifcation. There is a different architecture from the classic CNNs that was designed specifically for the Semantic Segmentation task called **Fully Convolutional Networks**.

FCN, first introduced by Long et al in 2015 [3], is a family of networks with a structure (figure 4) divided into two parts:

- *Contracting path*: a sequence of Convolution Layer, Pooling Layer and Activation Layer as in the classical CNN. This allows to create feature map at different granularity (using Pooling Layers).
- *Expansive path*: sequence of Deconvolution (or Upsampling) Layer, Convolution Layer with kernel size (1x1) and combination of Layers. It reconstruct the original size of the input image by Deconvolution or Upsampling. Deconvolution can be seen as the inverse of the Convolution

operator, while Upsampling allows to expand the input following a certain policy (similarly, but opposite, to the Pooling).

In addiction to this there are some *Skip Connection* that allows to combine information from previous feature maps with the final result, in order to combine finer and coarse information, to make local predictions that respect the global structure.
This particular structure allows the network to be trained and tested using images of different sizes, although this will not be the case in this project.

The structure of the network I have implemented is different from that presented in the figure 4, mainly for practical limitations. Specifically, I have used an architecture with less levels in the Contracting path, intended as sequences of convolutions and pooling, and two convolotional layers (instead of four) in every level.

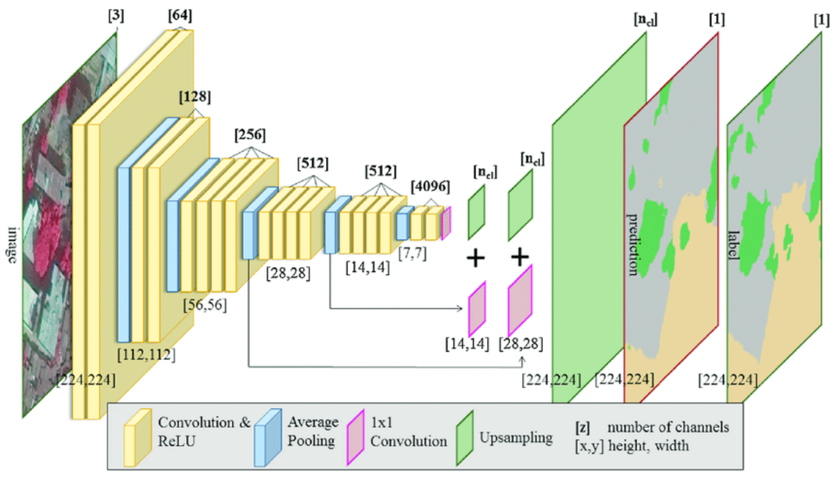


Figure 4: Fully Convolutional Network Architecture

3.1 U-Net

An evolution of FCN is the U-Net architecture [4], which is again formed by a contracting path and an expansive path.

The contracting path is the typical architecture of a convolutional network, so repeated application of two 3x3 convolutions (with ReLu), 2x2 max pooling operations for downsampling.

Every step in the expansive path consists of an upsampling of the feature map followed by a 2x2 convolution, a concatenation with the correspondingly cropped feature map from the contracting path, followed by again two 3x3 convolutions with ReLu. At the final layer a 1x1 convolution is used to map each component feature vector to the desired number of classes.

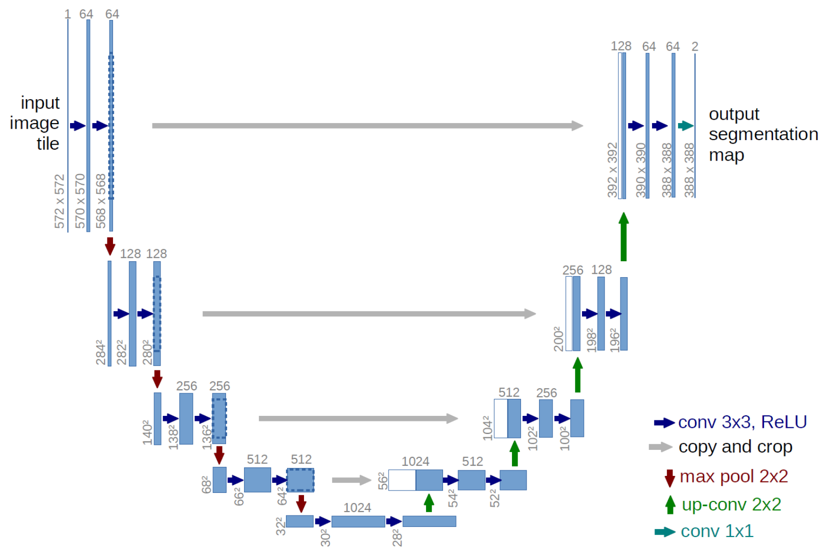


Figure 5: U-Net Architecture

Whenever the size of the feature map is halved (by pooling) the number of features is doubled, and vice versa, when the size of the feature map is doubled (upsampling or deconvolution) the number of features is halved.

The model that I have implemented is slightly different from the one presented in figure 5, in fact the convolutional layers have a padding that allows to obtain (in the same layer) feature maps of equal size. In this way, the concatenation of the layers from the contracting path to the expansive path can be done without cropping the vectors. Another difference is the way you expand the size of the feature map, instead of using an upsampling followed by a convolution I used directly a 2x2 deconvolution.

4 Experiments

All experiments were carried out using notebooks running in Google Colab. While this gave me access to a GPU to speed up training, it did not allow me to use too large a network or test too many different configurations in the two grid searches.

4.1 Evaluation Metrics

The evaluation metrics used for understanding the capabilities of the model are:

- **Accuracy**: it computes the number of correctly classified pixels compared to the total number of pixels. In the context of semantic segmentation, this is not a good representation of the model capabilities, due to the fact that the interested region is small compared to the whole image, and so if the model classified all pixels as background it would still get an Accuracy around 80-90%.

- **Dice Coefficient**: it can be defined over two sets X and Y as:

$$\frac{2 * |X \cap Y|}{|X| + |Y|}$$

  To be able to use this metric in the context of binary masks, just consider each position of the mask (hence each pixel) as an object in the set and the mask itself as a membership matrix that indicates whether an object belongs to the set.
  This metric allows to solve the problems of Accuracy, as it considers only the pixels that have label 1, then those classified as *Tumor*. Another name with which this coefficient is known is *F1* and, in this definition, it relates both the precision and the recall and therefore obtains a more truthful estimate of the performance of the model.

4.2 Loss Function

There are several loss functions that can be used in contexts such as this one [2], the ones I chose to test are:

- **Binary Crossentropy**, classical loss function for binary problems, it measures the difference between two probability distributions for a given random variable or set of events

- **Dice Loss**, As the name suggests, this loss function is closely related to the Dice Coefficient and in fact is defined as:

$$DiceLoss = 1 - DiceCoefficient$$

  The differentiability of this loss function is given by how it is implemented, in fact I used sums and products between real numbers instead of intersections between booleans.

- **Tversky Loss**, this loss function is similar to the previous one, in fact it is calculated starting from a coefficient, called Tvserky index, which can be seen as a generalization of the Dice Coefficient.
  Another interpretation of the Dice coefficient makes use of the confusion matrix:

$$DiceCoefficient = \frac{2TP}{2TP + FP + FN}$$

  Tvserky index assigns different weights to false positives and false negatives:

$$TvserkyIndex = \frac{TP}{TP + \alpha FP + \beta FN}$$

So the loss function:

$$TvserkyLoss = 1 - TvserkyIndex$$

4.3 GridSearch

For both architectures, a grid search was performed to find the combination of number of filters, dropout and loss function that gives the best performance. The specific values tested are given in Table 1 for FCN and Table 2 for UNET.

Hyperparameter	**Values**
Filters per layer	[32, 64, 128, 256, 512] [64, 128, 256, 512, 1024, 512]
Dropout	0, 0.1
Loss Function	binary crossentropy, dice loss, Tversky loss

Table 1: Hyperparameter list for FCN

Hyperparameter	**Values**
Filters per layer	[32, 64, 128, 256, 512, 256, 128, 64, 32] [64, 128, 256, 512, 1024, 512, 256, 128, 64]
Dropout	0, 0.1
Loss Function	binary crossentropy, dice loss, Tversky loss

Table 2: Hyperparameter list for UNet

The tests on these two types of networks have shown how the FCN is not able to model the problem correctly and to provide an acceptable result, while the U-Net has managed to carry out a fairly good pixel-by-pixel classification.

Another consideration can be made on the effectiveness of the two implemented Loss functions, that is the Dice Loss and the Tversky Loss. In fact, these did not bring the hoped for effects and, on the contrary, made the models converge to a state in which, independently of the starting image, a completely black image was given as a result. This may be due to the imbalance between the two classes of pixels, with the tumour area averaging 2% of the entire image.

Finally, there were no major differences in performance between cases using 256x256 pixel images and those using 128x128 pixel images, except in terms of training time. For this reason, it was decided to use the lower resolution images.

That said, the best configuration turned out to be the U-net with the configuration:

- binary crossentropy as Loss function
- Dropout layers after every convolutional layer with parameter 0.1
- 32, 64, 128, 256, 512, 256, 128, 64, 32 filters per layer

Figure 6 shows the graphs concerning the training statistics in the network, specifically the trend of the model accuracy, the dice coefficient and the loss function. What should be noted in these graphs, is the fact that the accuracy is already at 98%, while the dice coefficient starts from very close to zero and only after twenty epochs reaches around 70%, this highlights exactly what I said before about the problems related to accuracy in these contexts.

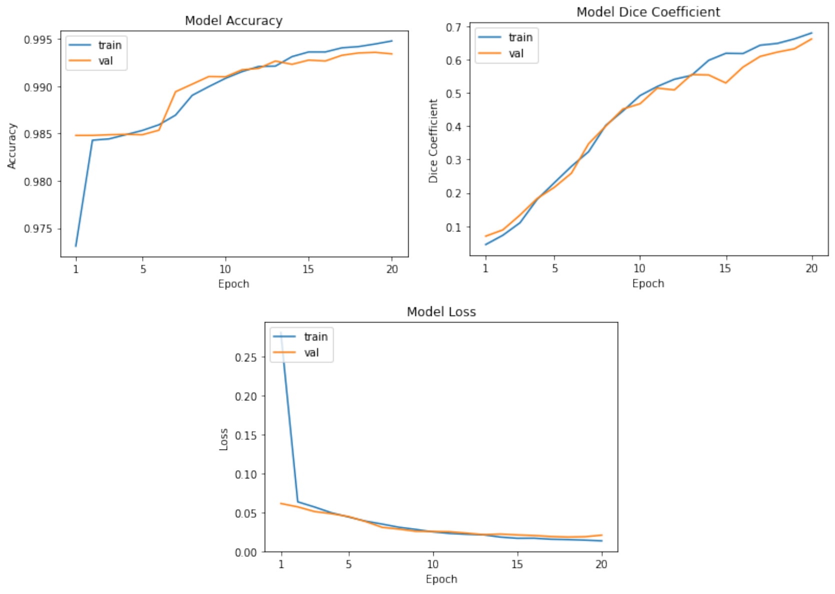


Figure 6: a) Accuracy b) Dice coefficient c) Binary Crossentropy loss

4.4 Result

The best performing configuration was used to train a model with the entire training set (thus including the validation set) and was able to obtain a dice coefficient on the test set equal to 0.72 , after transforming the output of the model into zeros and ones through the use of a threshold (0.5).

An example of model prediction is shown in figure 7, where it's possible to see the initial image, ground truth and mask predicted by the model. Although the prediction of the model is not perfect, it manages to identify the position, shape and size of the tumor quite well.
Figure 8, on the other hand, shows an example of a record in which there are some misclassified pixels, constituting cases of false positives, and this is the main reason why the dice score is only around 0.7

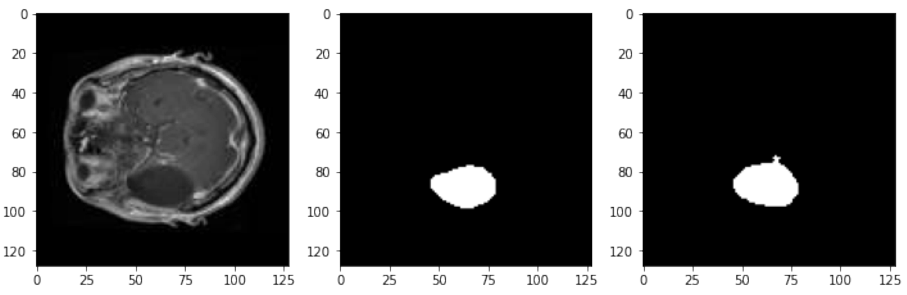


Figure 7: a) MRI b) Ground truth c) Prediction

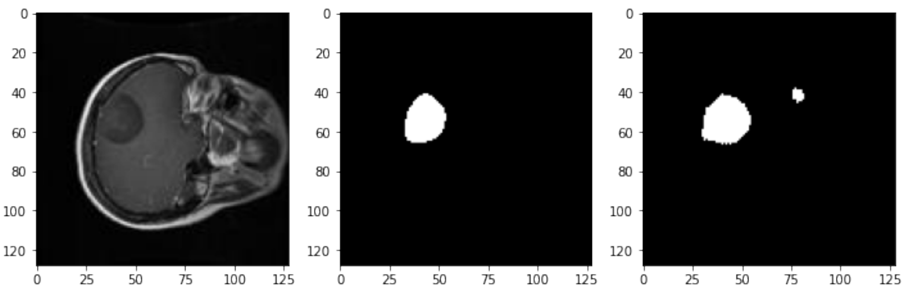
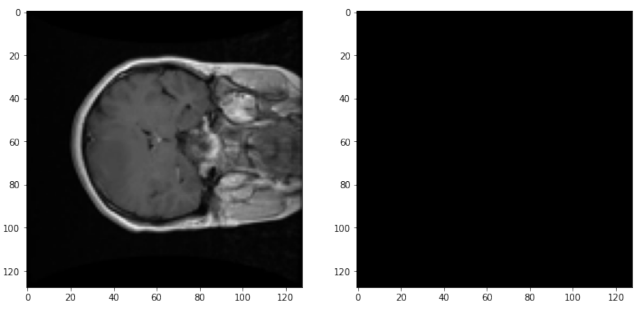


Figure 8: a) MRI b) Ground truth c) Prediction

4.5 Further Observations

Despite the fact that every image in the dataset, and therefore used in training the network, contained a tumor; the network also manages to correctly classify images of healthy brains. This can be seen from image 9, where each pixel of the two MRIs, representing brains without tumours, has been classified as background.



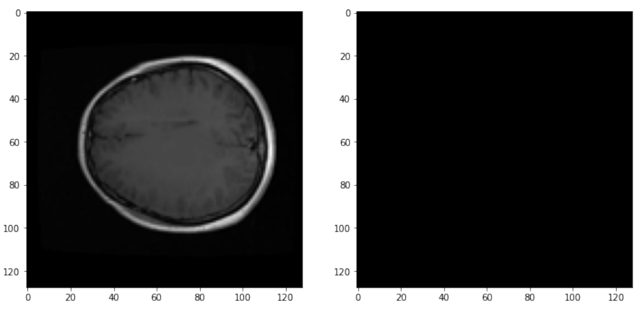


Figure 9: Original MRIs, on the left, and predicted mask, on the right

4.6 Conclusion

In this project, I analysed the use of convolutional neural networks in the context of semantic segmentation in the Magnetic Resonance Image domain. The aim was to build a model that could classify each pixel as either Background or Tumour.

The architectures I decided to test are the Fully Convolutional Network and an evolution of it, the U-net. These particular neural networks use only convolutional, deconvolutional and pooling layers, in addition to this, there are also skip connections which allow information to be combined at different granularities.

As well as testing different architectures, I also tested different Loss functions in order to find the one that gave the best results on the dataset under consideration. Experiments have shown that the U-net in combination with the binary crossentropy loss function gives very good results in identifying tumours within MRIs. The final network, trained with the entire dataset, was able to achieve very good results, given the complexity of the task.

References

[1] MRI Dataset. https://figshare.com/articles/dataset/brain_tumor_dataset/1512427/5.

[2] Shruti Jadon. A survey of loss functions for semantic segmentation. In *2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, pages 1–7. IEEE, 2020.

[3] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[4] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.