Alma Mater Studiorum · University of Bologna

School of Science
Department of Physics and Astronomy
Master Degree in Physics

Machine Learning methods for hepatocellular malignancies segmentation and MVI prediction

Supervisor:
Prof. Gastone Castellani

Submitted by:
Laura Verzellesi

Academic Year 2020/2021

Abstract

The aim of this thesis project is to automatically localize HCC tumors in the human liver and subsequently predict if the tumor will undergo microvascular infiltration (MVI), the initial stage of metastasis development.

The input data for the work have been partially supplied by Sant'Orsola Hospital and partially downloaded from online medical databases.

Two Unet models have been implemented for the automatic segmentation of the livers and the HCC malignancies within it. The segmentation models have been evaluated with the Intersection-over-Union and the Dice Coefficient metrics. The outcomes obtained for the liver automatic segmentation are quite good (IOU = 0.82; DC = 0.35); the outcomes obtained for the tumor automatic segmentation (IOU = 0.35; DC = 0.46) are, instead, affected by some limitations: it can be state that the algorithm is almost always able to detect the location of the tumor, but it tends to underestimate its dimensions.

The purpose is to achieve the CT images of the HCC tumors, necessary for features extraction.

The 14 Haralick features calculated from the 3D-GLCM, the 120 Radiomic features and the patients' clinical information are collected to build a dataset of 153 features. Now, the goal is to build a model able to discriminate, based on the features given, the tumors that will undergo MVI and those that will not. This task can be seen as a classification problem: each tumor needs to be classified either as "MVI positive"or "MVI negative".

Techniques for features selection are implemented to identify the most descriptive features for the problem at hand and then, a set of classification models are trained and compared.

Among all, the models with the best performances (around 80-84% $\pm$ 8-15%) result to be the XGBoost Classifier, the SDG Classifier and the Logist Regression models (without penalization and with Lasso, Ridge or Elastic Net penalization).

Contents

Chapter 1

Introduction

The aim of the project is to automatically recognize HepatoCellular Carcinomas (HCC) within the CT of a human liver and predict the onset of MicroVascular Infiltrations (MVI).

Hepatocellular carcinoma is one of the most common causes for cancer death worldwide.

Microvascular Infiltration refers to the presence of tumor cells in the vascular space lined by endothelial cells, which primarily consists of portal vein branches (including intracapsular vessels) [1].

MVI is, therefore, a necessary step in the metastatic evolution of tumors and it is believed to be related to the postoperative recurrence and prognosis of cancer patients.

Thus, the prediction of the onset of the MVI in the initial stages of the tumors could be a great help in the reduction of life-threatening risks for the patients.

Sant'Orsola Hospital provided CT images of 84 patients affected by HCC tumors and a database with clinical, surgical, radiological information about each patient. The data provided by Sant'Orsola Hospital are described in Chapter 2.

Considering the goal, the project is structured in the following way: first, it is necessary to automatically segment the liver in the CT image (Chapter 3), then locate the HCC tumor within the liver (Chapter 4) and finally find features that describe the tumor and combine them with patients' clinical data to predict the onset of the MVI (Chapter 5). Since the MVI is a binary variable, the last task can be seen as a classification problem: the features extracted and collected are used to classify each tumor either as "MVI positive", *MVI = 1*, or "MVI negative", *MVI = 0*. To do so, it is necessary to determine an approach to select and keep, among the entire dataset of features assembled, only the most remarkable ones for the defined task. Finally, the subset of features thus obtained is used to train and compare a set of different machine learning models with the purpose of finding the optimal one.

Chapter 2

Sant'Orsola Hospital data

Sant'Orsola Hospital provided the CT images and the medical records of 84 patients affected by HCC tumors. Each CT slice is of dimension 512x512 and it is accompanied by the ROI coordinates (shown in red in Figure 2.1) to indicate the location of the tumor, if present.

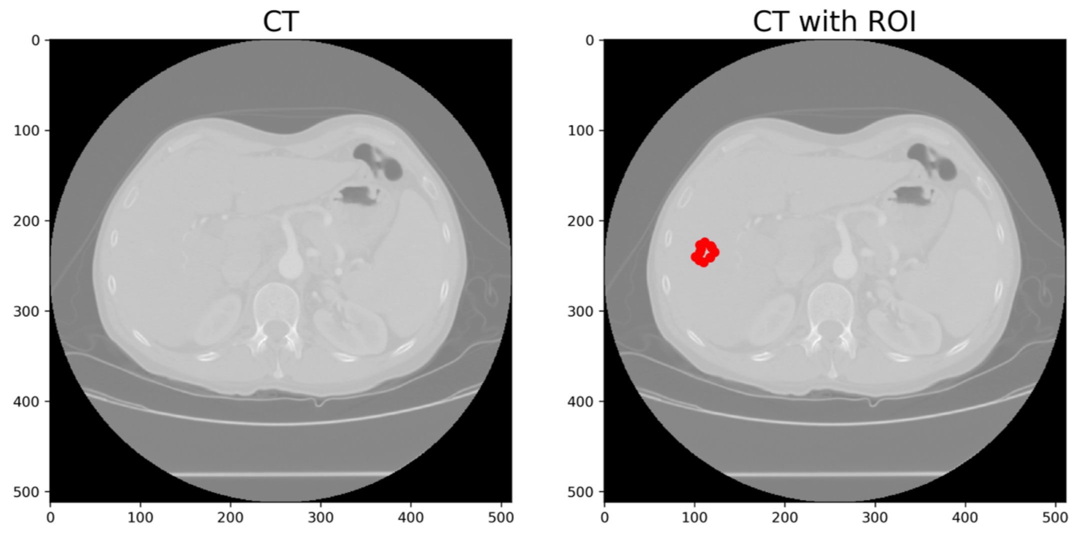
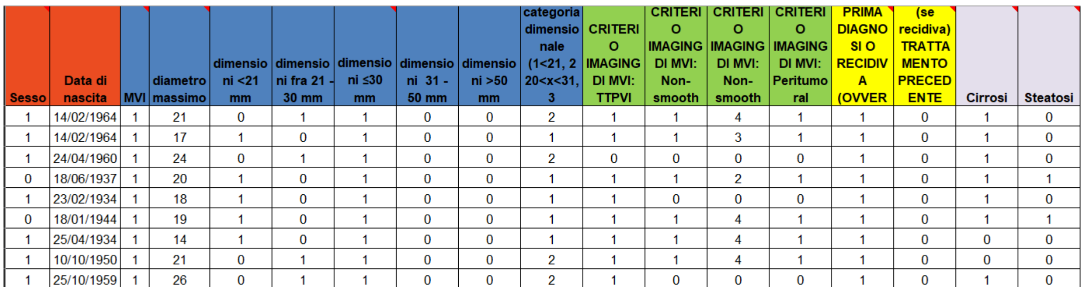


Figure 2.1: On the left: example of one CT slice of the database provided by Sant'Orsola Hospital; on the right: CT slice overlapped with the ROI coordinates (in red).

The data were furnished in two subsequent steps. At first only 57 patients were supplied and for these, both the arterial and the venous phases were disposed. Then, the arterial phases of 27 further patients were added.

Furthermore, a database containing clinical, radiological, histological, surgical and post-operative information about each patient has been supplied. An extract is shown in Figure 2.2.



Sesso	Data di nascita	MVI	diametro massimo	dimensioni <21 mm	dimensioni fra 21 - 30 mm	dimensioni ≤30 mm	dimensioni 31 - 50 mm	dimensioni >50 mm	categoria dimensionale (1<21, 2 20<x<31, 3	CRITERIO IMAGING DI MVI: TTPVI	CRITERIO IMAGING DI MVI: Non-smooth	CRITERIO IMAGING DI MVI: Non-smooth	CRITERIO IMAGING DI MVI: Peritumoral	PRIMA DIAGNOSI O RECIDIVA (OVVER	(se recidiva) TRATTAMENTO PRECEDENTE	Cirrosi	Steatosi
1	14/02/1964	1	21	0	1	1	0	0	2	1	1	4	1	1	0	1	0
1	14/02/1964	1	17	1	0	1	0	0	1	1	1	3	1	1	0	1	0
1	24/04/1960	1	24	0	1	1	0	0	2	0	0	0	0	1	0	1	0
0	18/06/1937	1	20	1	0	1	0	0	1	1	1	2	1	1	0	1	1
1	23/02/1934	1	18	1	0	1	0	0	1	1	0	0	0	1	0	1	0
0	18/01/1944	1	19	1	0	1	0	0	1	1	1	4	1	1	0	1	1
1	25/04/1934	1	14	1	0	1	0	0	1	1	1	4	1	1	0	0	0
1	10/10/1950	1	21	0	1	1	0	0	2	1	1	4	1	1	0	0	0
1	25/10/1959	1	26	0	1	1	0	0	2	1	0	0	0	1	0	1	0

Figure 2.2: Extract of the database containing medical records of each patient.

Chapter 3

Liver automatic segmentation

3.1 Input images databases

In order to develop the first step of the project it is necessary to use a different database of images. In fact, the images provided by Sant'Orsola Hospital do not have any expert manual segmentation of the liver. Manual segmentations are required for a model that has to learn how to automatically segment the liver from a medical image: they are used as "ground-truths" in the training part. Therefore, two online databases have been downloaded and used:

- **3D-IRCADb-01 database** [2], which is composed of the three-dimensional CT-scans of 10 women and 10 men with hepatic tumors in 75% of cases;
- **LiTS database** [3], which is composed of 131 3D CT-scans collected from 6 medical centres (only the portion dedicated to the training has been considered). These data were acquired with different scanners and scanning protocols and the examined patients suffered from primary to secondary liver tumor and metastases.

In both these databases, each 3D CT image comes with its corresponding liver mask. The final database is totally composed of 23122 images (accompanied by the correspondent masks) of size 512x512. Some examples are shown in Figure 3.1.

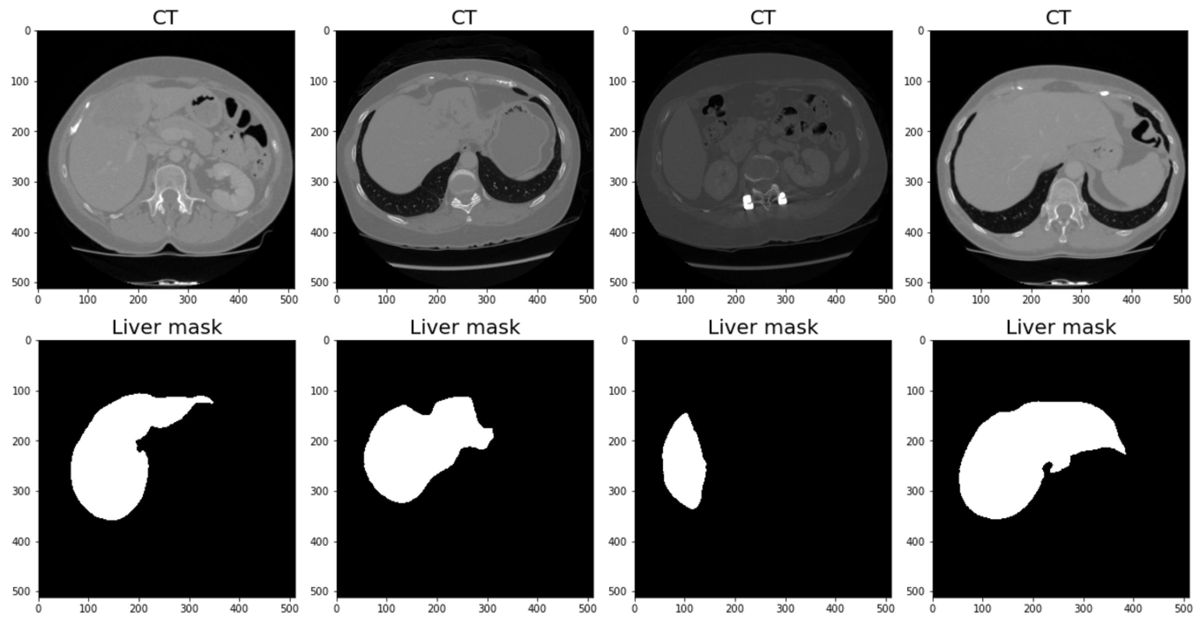


Figure 3.1: Examples of the images of the database used for the training and testing of the automatic liver segmentation model. Each CT slice is accompanied by the correspondent liver mask.

3.1.1 Images manipulation

All the CT images of the thus obtained database are rescaled in the grey-level range [0, 200]. Within this range, in fact, the liver results highlighted with respect to the surroundings. Figure 3.2 shows the differences before and after the rescale.

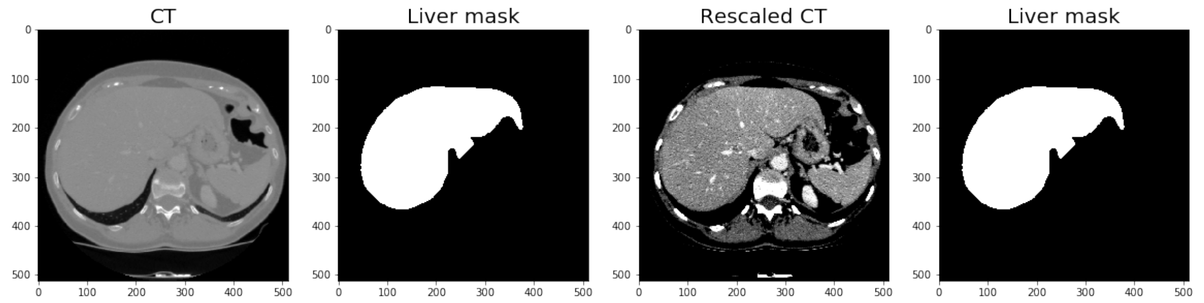


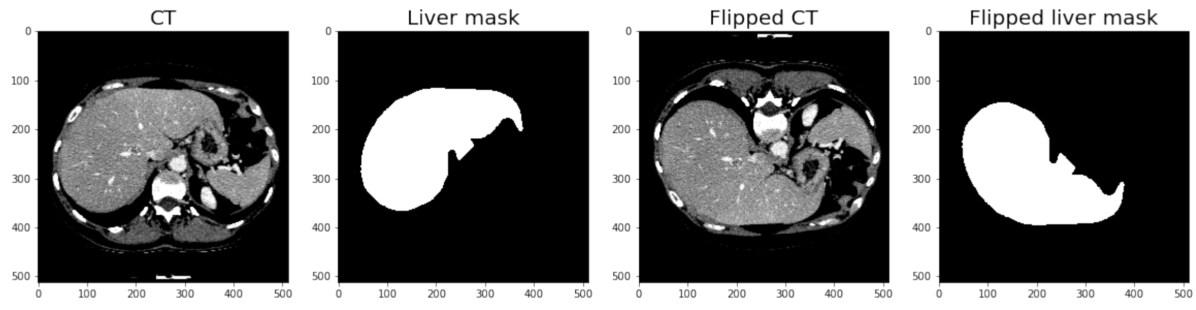
Figure 3.2: Examples of an image before and after the rescale in the grey-level range [0,200].

In machine learning models it is always advisable to validate the stability of the model as it is not taken for granted that a model trained on specific *training data* will accurately adapt to brand new data. Therefore, the database is split into two disjoint subsets: the 80% of it composes the *training set* and the remaining 20% constitutes the *testing set*. In this way it is possible to measure the performances of the model on a set of images it has not seen during its training.

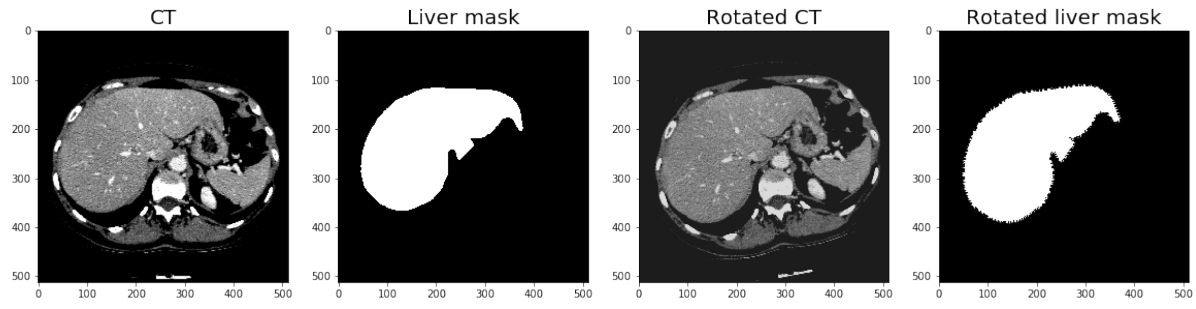
To generalize the dataset given to the model, some “manipulation”functions are applied on the *training set*. Possible transformations are:

- **Random flip**. An image flip is performed randomly. The flip can either be a horizontal flip, a vertical flip (an example is reported in Figure 3.3a) or both.
- **Random rotation**. A rotation of the image is performed by a random angle in range [-10, 10], as shown in Figure 3.3b.
- **Random distortion**. A random distortion of the image is executed according to either a sine or cosine function, as in Figure 3.3c.

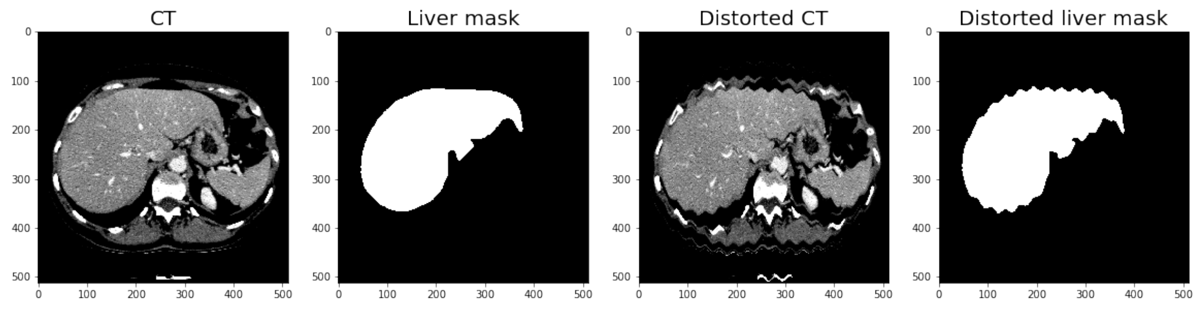
Each image can be subjected to more than one transformation.



(a) Random flip transformation example.



(b) Random rotation transformation example.



(c) Random distortion transformation example.

Figure 3.3: Image transformations to generalize the initial dataset before training.

To improve the training of the Unet model, both the images and the correspondent masks are resized into dimension 128x128 and the images are then rescaled between 0 and 1. Finally, the masks are binarized.

3.2 Unet model: train and performances

The algorithm trained to achieve the automatic segmentation of the liver from the CT images is a Unet.

The Unet belongs to the class of the Convolutional Neural Networks, which are a category of Multi Layer Perceptrons Neural Networks. The Unet architecture, shown in Figure 3.4, looks like a “U”, which justifies its name, and contains two paths that are mainly composed by a succession of convolutional (or transposed convolutional) and max pooling layers. The first path is the contraction path, which is used to capture the context in the image; the second path is the symmetric expanding path, which is used to enable precise localization.

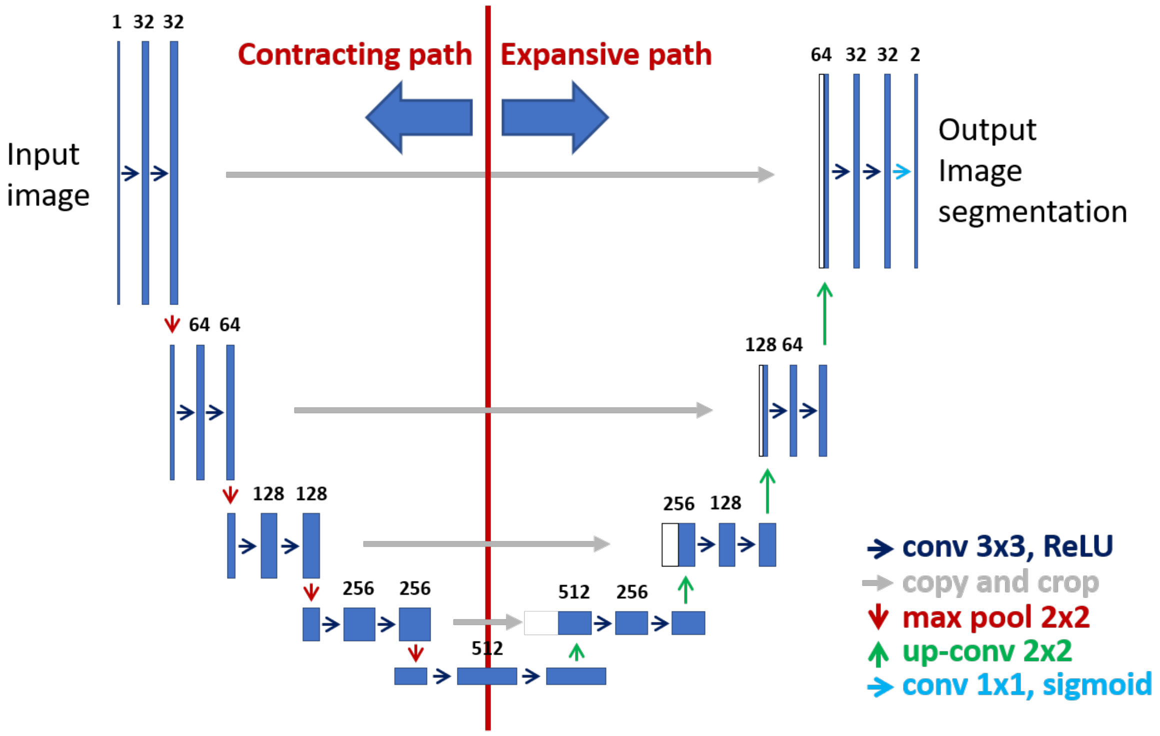


Figure 3.4: Unet architecture: the contraction path on the left is used to capture the context in the image, the expanding path on the right is used to enable precise localization.

The primary purpose of convolution in case of a Convolutional Neural Network is to extract features from the input image. A 3x3 matrix, called “filter” or “feature detector”, is slid over the original image, one by one pixel, and for every position, the multiplication between the two matrices is computed. The multiplication outputs are added together in order to get the final integer value which forms a single element of the output matrix, called “feature map”. The procedure is shown in Figure 3.5.

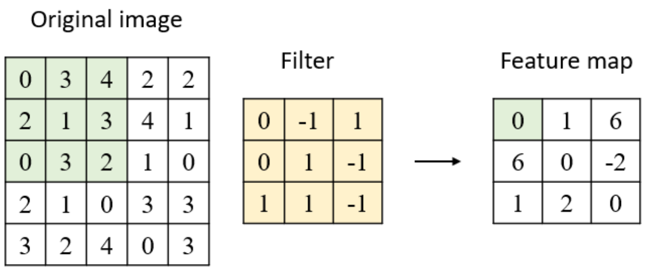


Figure 3.5: Convolution between an example of input image (on the left) and an example of a filter (in the middle). The result of the operation is a feature map (on the right).

It is evident that different values of the filter matrix will produce different feature maps for the same input image, and also, that different filters can detect different features (edges, curves, lines etc).

In practice, a Convolutional Neural Network learns the values of these filters during the training process and becomes able to apply them to a new unseen image in order to "classify" its different regions.

An additional operation called ReLU (Rectified Linear Unit) is used after every convolution operation to introduce non-linearity in the Neural Network: it replaces all negative pixel values in the feature map by zero.

The max pooling step reduces the dimensionality of each features map. A 2x2 window is defined and scrolled onto the feature map. Only the highest element from each region is taken, as shown in Figure 3.6. This operation makes the input representations smaller and more manageable.

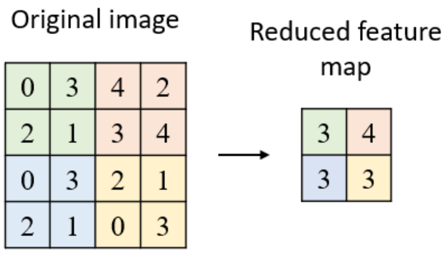
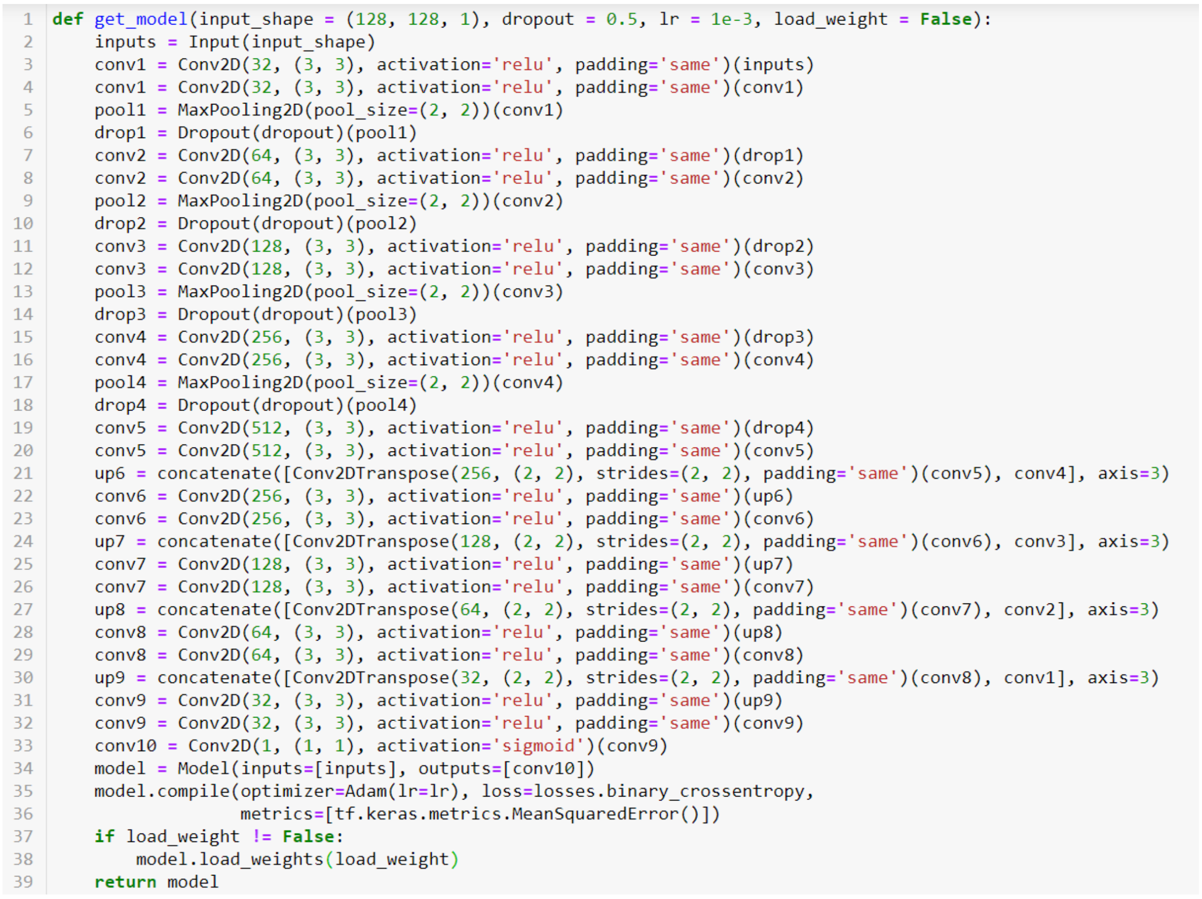


Figure 3.6: Max pooling operation between an example of input image (on the left) and a 2x2 window. The result of the operation is a reduced feature map (on the right).

In our case, the purpose is to define the best filters to detect the liver pixels in the input images in order to be able to detect them also in an unseen image.

At the beginning, all the filters are initialized with random values; the network takes a training image as input, goes through the *forward propagation* step and finds, for each pixel of the image, the output probabilities of belonging to the liver or to the background; it compares the result with the correspondent manually segmented image and it calculates the total error of the output; it than uses the *back propagation* step to update all filter values in order to minimize the output error. These steps are repeated for all the images in the *training set*. The network is trained when the filters have been optimized to correctly classify all the training examples. Now, when a new unseen image is input in the Unet, the network would go through the *forward propagation* step using those optimized filters in order to return a "probability map", where the grey-level of each pixel is related to its probability of belonging to the "liver class". Zero probability of being a "liver pixel"correspond to a black pixel, i.e. a pixel that belongs to the "background class".

The implementation of the Unet in the code is shown in Figure 3.7.



```python
def get_model(input_shape = (128, 128, 1), dropout = 0.5, lr = 1e-3, load_weight = False):
    inputs = Input(input_shape)
    conv1 = Conv2D(32, (3, 3), activation='relu', padding='same')(inputs)
    conv1 = Conv2D(32, (3, 3), activation='relu', padding='same')(conv1)
    pool1 = MaxPooling2D(pool_size=(2, 2))(conv1)
    drop1 = Dropout(dropout)(pool1)
    conv2 = Conv2D(64, (3, 3), activation='relu', padding='same')(drop1)
    conv2 = Conv2D(64, (3, 3), activation='relu', padding='same')(conv2)
    pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)
    drop2 = Dropout(dropout)(pool2)
    conv3 = Conv2D(128, (3, 3), activation='relu', padding='same')(drop2)
    conv3 = Conv2D(128, (3, 3), activation='relu', padding='same')(conv3)
    pool3 = MaxPooling2D(pool_size=(2, 2))(conv3)
    drop3 = Dropout(dropout)(pool3)
    conv4 = Conv2D(256, (3, 3), activation='relu', padding='same')(drop3)
    conv4 = Conv2D(256, (3, 3), activation='relu', padding='same')(conv4)
    pool4 = MaxPooling2D(pool_size=(2, 2))(conv4)
    drop4 = Dropout(dropout)(pool4)
    conv5 = Conv2D(512, (3, 3), activation='relu', padding='same')(drop4)
    conv5 = Conv2D(512, (3, 3), activation='relu', padding='same')(conv5)
    up6 = concatenate([Conv2DTranspose(256, (2, 2), strides=(2, 2), padding='same')(conv5), conv4], axis=3)
    conv6 = Conv2D(256, (3, 3), activation='relu', padding='same')(up6)
    conv6 = Conv2D(256, (3, 3), activation='relu', padding='same')(conv6)
    up7 = concatenate([Conv2DTranspose(128, (2, 2), strides=(2, 2), padding='same')(conv6), conv3], axis=3)
    conv7 = Conv2D(128, (3, 3), activation='relu', padding='same')(up7)
    conv7 = Conv2D(128, (3, 3), activation='relu', padding='same')(conv7)
    up8 = concatenate([Conv2DTranspose(64, (2, 2), strides=(2, 2), padding='same')(conv7), conv2], axis=3)
    conv8 = Conv2D(64, (3, 3), activation='relu', padding='same')(up8)
    conv8 = Conv2D(64, (3, 3), activation='relu', padding='same')(conv8)
    up9 = concatenate([Conv2DTranspose(32, (2, 2), strides=(2, 2), padding='same')(conv8), conv1], axis=3)
    conv9 = Conv2D(32, (3, 3), activation='relu', padding='same')(up9)
    conv9 = Conv2D(32, (3, 3), activation='relu', padding='same')(conv9)
    conv10 = Conv2D(1, (1, 1), activation='sigmoid')(conv9)
    model = Model(inputs=[inputs], outputs=[conv10])
    model.compile(optimizer=Adam(lr=lr), loss=losses.binary_crossentropy,
                  metrics=[tf.keras.metrics.MeanSquaredError()])
    if load_weight != False:
        model.load_weights(load_weight)
    return model
```

Figure 3.7: Unet architecture implemented in Python.

3.2.1 Model parameters

The choice of the model parameters is crucial for the correct realization of the purpose.

The model takes as input images of dimensions 128x128x1, the *batch size* is 64 and the *learning rate* is set at $1e-3$.

The selected *loss function* is the Binary Cross-Entropy, widely used for pixel level classification objective, exactly the segmentation purpose that is aimed in this step of the project. The Binary Cross-Entropy loss function is defined as:

$$L_{BCE}(y, \hat{y}) = -(ylog(\hat{y}) + (1-y)log(1-\hat{y})) \tag{3.1}$$

where, $\hat{y}$ is the predicted value by the prediction model. As the predicted probability diverges from the actual label, the Cross-Entropy loss increases.

The *epochs* are initialized to 100 but an *early stop* is set to monitor the value of the *val_loss* with a patience of 10. In this way, the model achieves optimal training after 21 epochs.

3.2.2 Test

At the conclusion of the training, the model is ready to be tested. In the following Figure 3.8 is shown the application of the trained model on the *testing set*'s images. The images in the second column represent the segmentations predicted by the trained model, while the images in the third column are the true liver segmentations.

In general, the results obtained appear to be good: the model is able to correctly discriminate between the pixels of the liver and the pixels of the background.

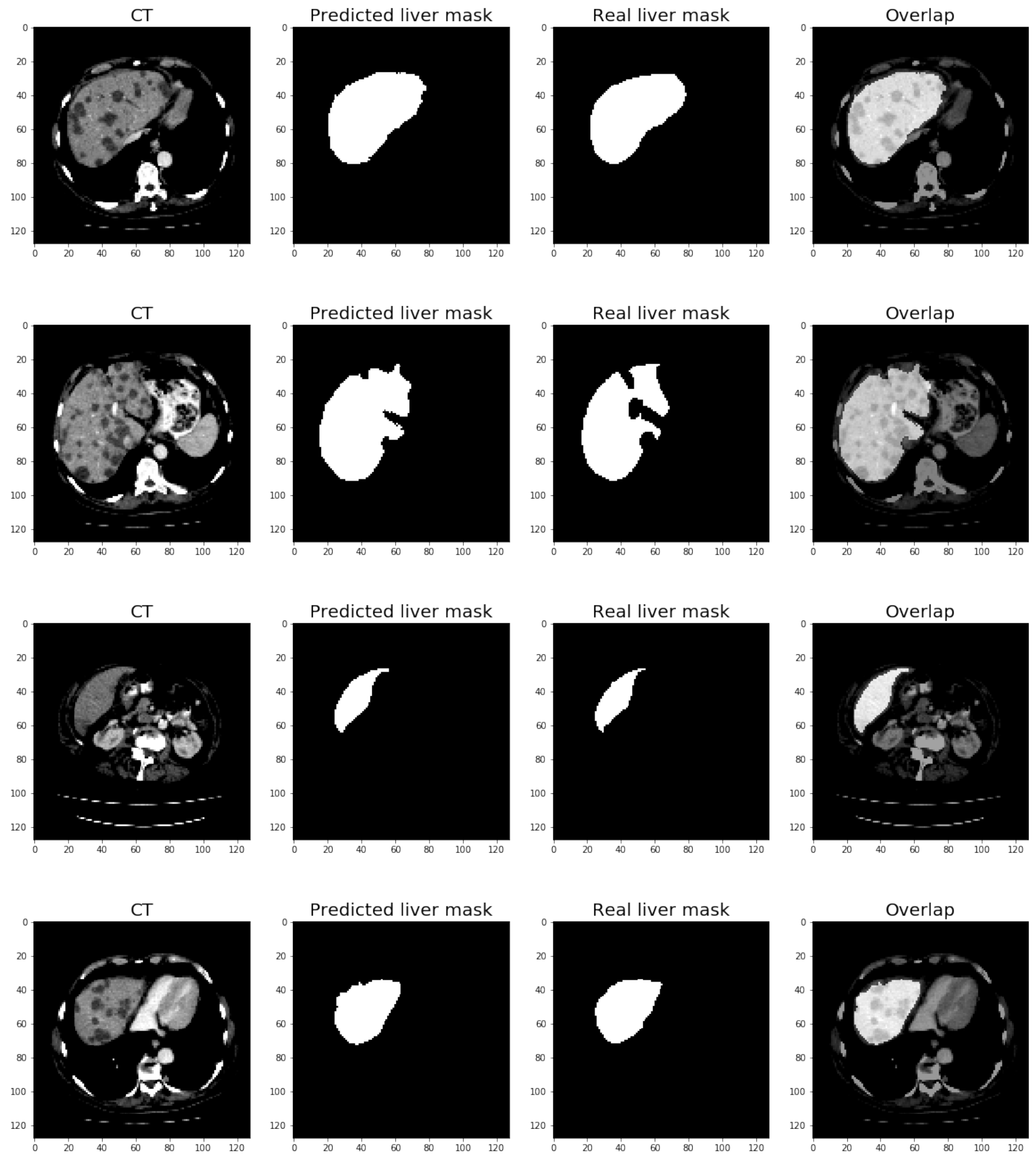


Figure 3.8: From the left: the image from the *testing set*, the liver segmentation predicted, the true segmentation, the overlap of the input image and the predicted liver mask.

3.2.3 Performances

Two different metrics are considered to evaluate the performances of the model [4]. They both evaluate the degree of overlap between the predicted segmentation mask and the reference segmentation mask.

Intersection-Over-Union metric. The Intersection-Over-Union metric (IOU), also known as the Jaccard Index, is given by Equation 3.2, where the numerator is the area of overlap between the predicted segmentation and the ground truth while the denominator is the area of union between the predicted segmentation and the ground truth.

$$IOU = \frac{Area\ of\ Overlap}{Area\ of\ Union} \tag{3.2}$$

This metric ranges between 0 and 1 (0%–100%) with 0 signifying no overlap and 1 signifying perfectly overlapping segmentations.

Dice Coefficient. The Dice Coefficient (DC) is defined as twice the area of overlap between the predicted segmentation and the ground truth divided by the total number of pixels in both images:

$$DC = \frac{2 \cdot Area\ of\ Overlap}{Total\ number\ of\ Pixel} \tag{3.3}$$

Like the IOU, it also ranges from 0 to 1, with 1 signifying the greatest similarity between predicted and truth.

Computing these two metric on the entire *testing set*, the Intersection over Union results to be 0.82 and the Dice Coefficient 0.88. Figure 3.9 displays the computation of the metrics on a random CT image of the *testing set.* The grey region in the last image represents the difference between the real liver mask (second image) and the prediction of the liver according to the model (third image). The values of the metrics for the image are reported in the title above the third image.

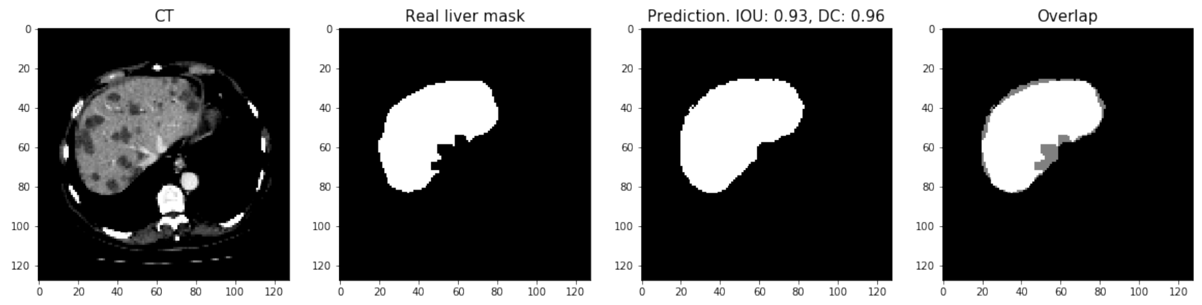


Figure 3.9: From the left: one image from the *testing set*, the true segmentation, the liver segmentation predicted by the model and the overlap between the predicted and the true liver: the white pixels are those in common between the two segmentations, while the gray pixels belong only to one of them. The title of the third image shows the values of the metrics for the selected CT.

Chapter 4

Tumor automatic segmentation

4.1 Input images database

In order to develop this step of the project, the database provided by Sant'Orsola Hospital has been used. Referring to Chapter 2, the 57 initial patients have been used for the training part and the 27 further patients for the testing part. Since for the last 27 patients the venous phase is not disposed, only the arterial phase of each patient is considered in the following discussion.

4.1.1 Manipulation on ***training set*** images

First of all, only the CT slices with tumor are selected among the entire database. Next, it is necessary to obtain a ground truth segmentation of the tumors. As already mentioned in Chapter 3.1, ground truth segmentations need to be provided to the model during the training session. As previously shown in Figure **??**, the images of Sant'Orsola's database are accompanied by the ROI (region of interest): a set of coordinates that localize the tumor in the image. The function *polygon()* from the *skimage.draw* library has therefore been used to transform the ROI coordinates into the tumor masks, as reported in Figure 4.1.

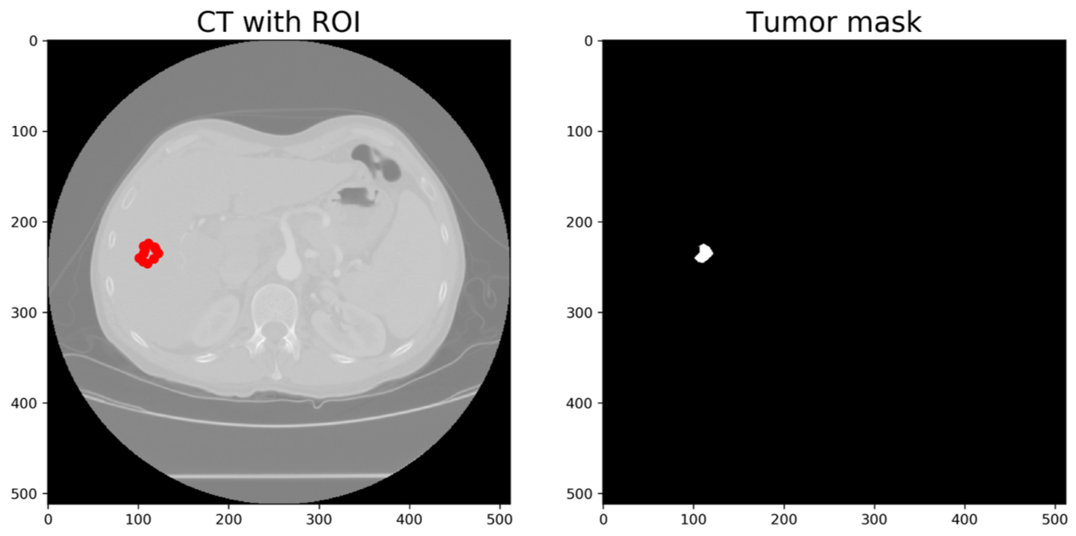


Figure 4.1: On the left: example of one CT slice with ROI coordinates (in red) overlapped; on the right: the tumor mask obtained from the ROI coordinates using *polygon()* function.

The necessary pre-processing is applied to the CT slices for liver detection: the model previously trained is used to segment the liver from the CT images.

The results of the application of the model to these images are not as precise as the ones obtained with the previous dataset. This may be due to the difference between the input images of the two databases, such as different image acquisition modalities or different initial grey-level scales. As a result, the tumor appears to be partially outside the predicted-liver's area in a significant number of slices, as can be seen in the second image of Figure 4.2.

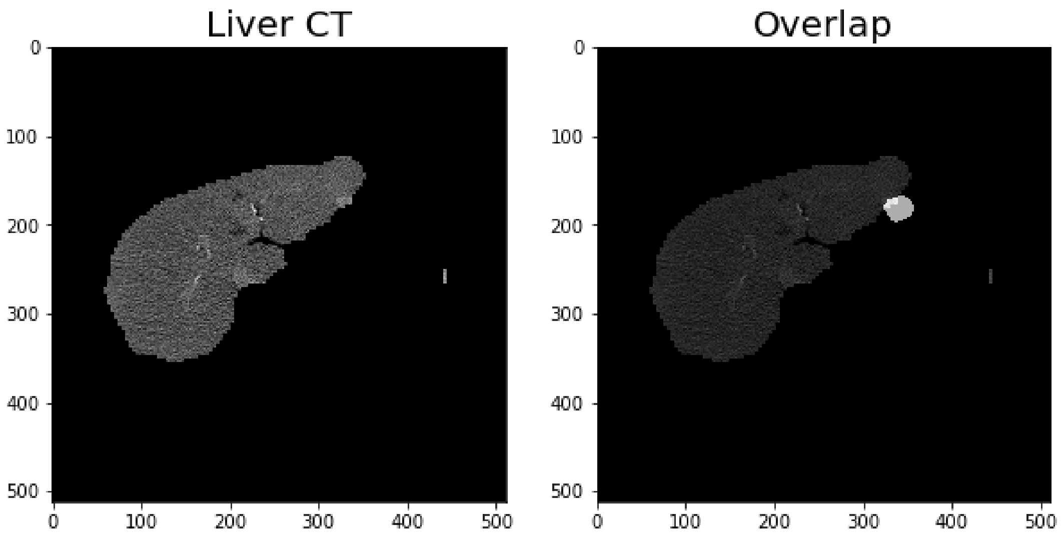


Figure 4.2: On the left: CT of the liver obtained with the previously trained model; on the right: the tumor mask overlapped to the predicted liver.

To overcome this problem, the liver mask has been dilated using the morphological function *cv2.dilate()*. Now, the predicted-dilated-liver mask is overlaid on the lesion segmentation to check if the lesions are still located outside the predicted-dilated-liver region. All the further images with tumor outside the predicted-dilated-liver must be discarded, otherwise the learning of the model would be deeply compromised. These successive steps are shown in Figures 4.3 and 4.4.

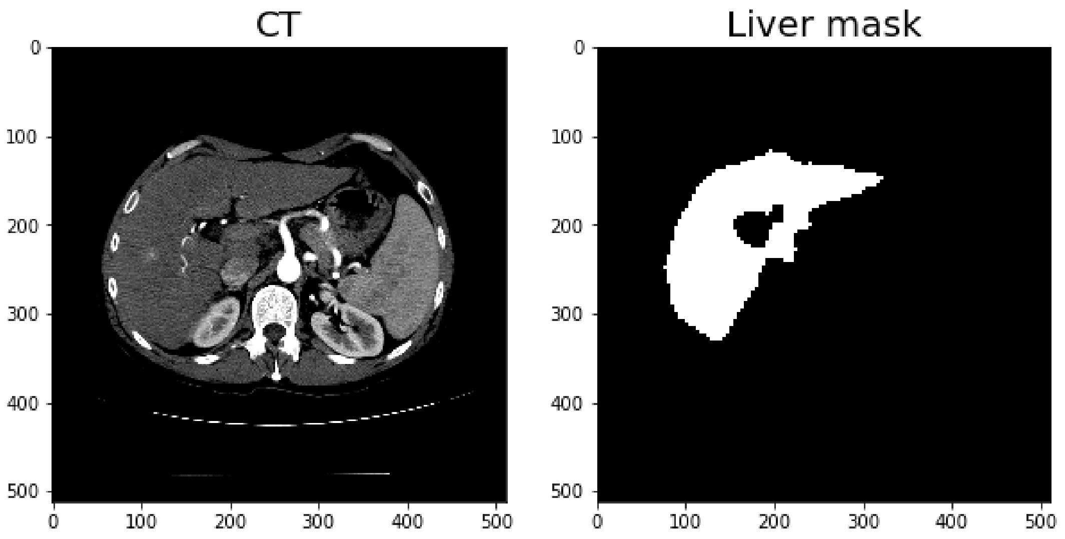


Figure 4.3: On the left: CT image pre-processed and ready to undergo the testing part of the previously trained model in order to segment the liver; on the right: the liver mask obtained.

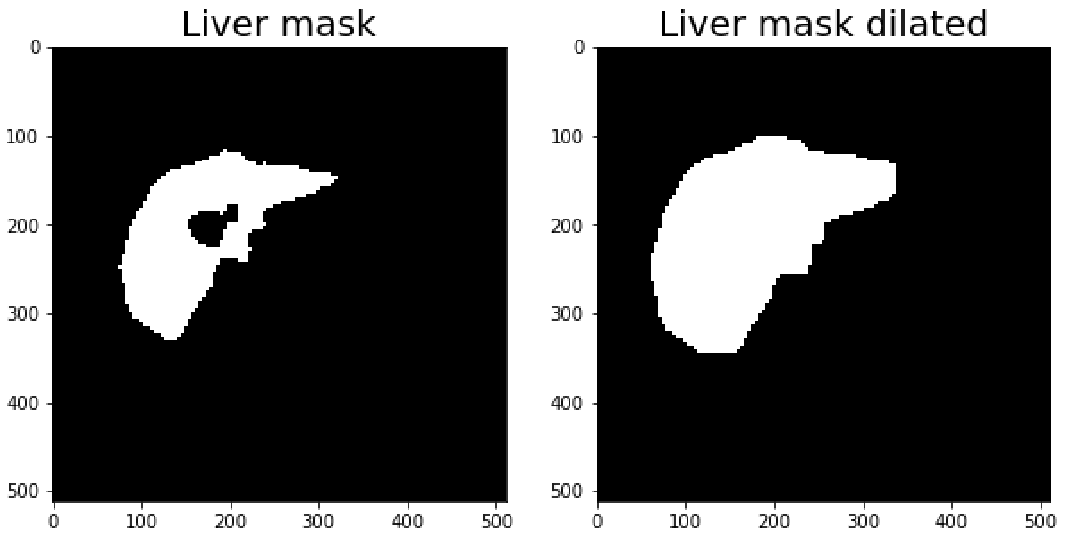


Figure 4.4: On the left: the liver mask obtained with the trained model; on the right: the liver mask after the morphological dilation.

At this point, the dataset is composed of 440 CT images and 440 correspondent tumor masks, all of dimensions 512x512.

To enhance the tumor contrast inside the CTs, a CLAHE equalization is applied on each slice. The function *cv2.createCLAHE()* computes several histograms, each corresponding to a distinct section of the image, and uses them to redistribute the lightness values of the image [5]. An example of the result is reported in Figure 4.5.

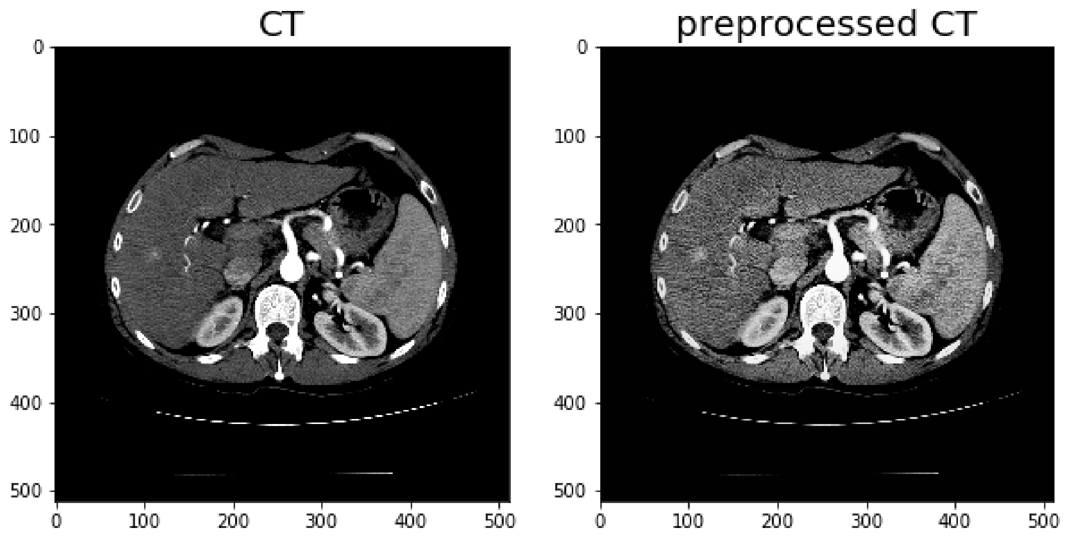


Figure 4.5: On the left: CT image before the CLAHE equalization; on the right: CT image after the CLAHE equalization, performed to increase the contrast on the tumor.

A *shuffle()* function is then applied to randomly change the order of the slices and the corresponding tumor masks.

Finally, *data augmentation* with a factor 3 is performed in order to enlarge and generalize the dataset. As it has been done in Chapter 3.1.1, possible transformations are: random flip, random rotation and random distortion.

Again, the images rescalation in range [0, 1] and the masks binarization are preparatory pre-processing to improve the performances of the model.

At this point the dilated-liver mask is overlapped to the entire image in order to extract only the CT image of the liver (first image of Figure 4.6).

The last necessary manipulation before the training process is the selection of patches inside the CT images and the correspondent masks. In fact, the proportion between the tumor pixels and all the other pixels in 512x512 images is extremely small. The model, that has to classify each pixel as either belonging to the tumor or not, receives a dataset deeply uneven: the majority of the pixels are labeled “background”and only a small percentage are labeled “tumor”. This strongly influence and compromise the training of the model. Therefore, instead of resizing the images and losing information, patches of dimension 128x128 from both the CTs and the masks are obtained. To do so, a dedicated function has been implemented. The function takes as input each image and its corresponding mask, it slides over the mask until a pixel of the tumor is found, it then moves of a random shift before and above that pixel and, finally, it considers and crops a 128x128 patch from that point on the CT image. In this way the model is fed with 128x128 crops around the tumor, as represented in Figure 4.6.

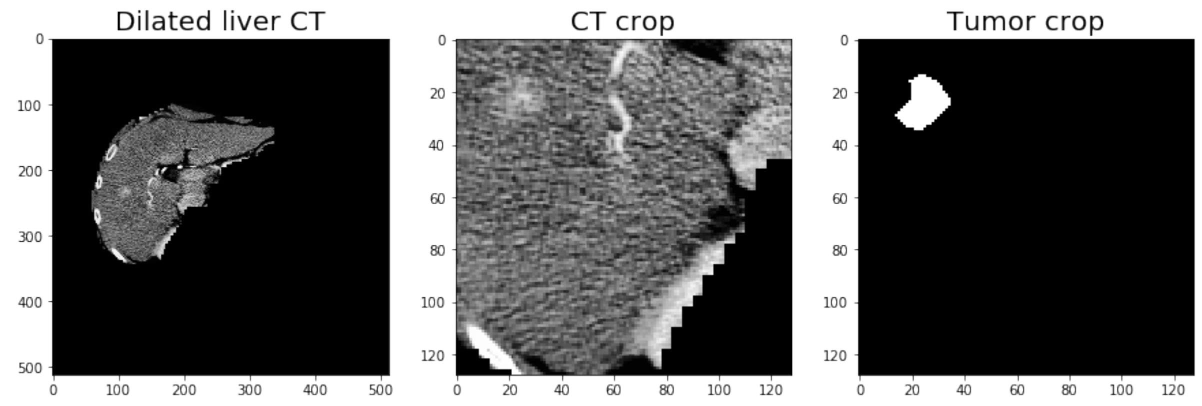


Figure 4.6: On the left: CT image of the dilated liver after CLAHE equalization; in the middle: 128x128 patch around the tumor of the CT image; on the right: correspondent 128x128 patch of the tumor mask.

Two examples of the final training dataset are illustrated in Figure 4.7.

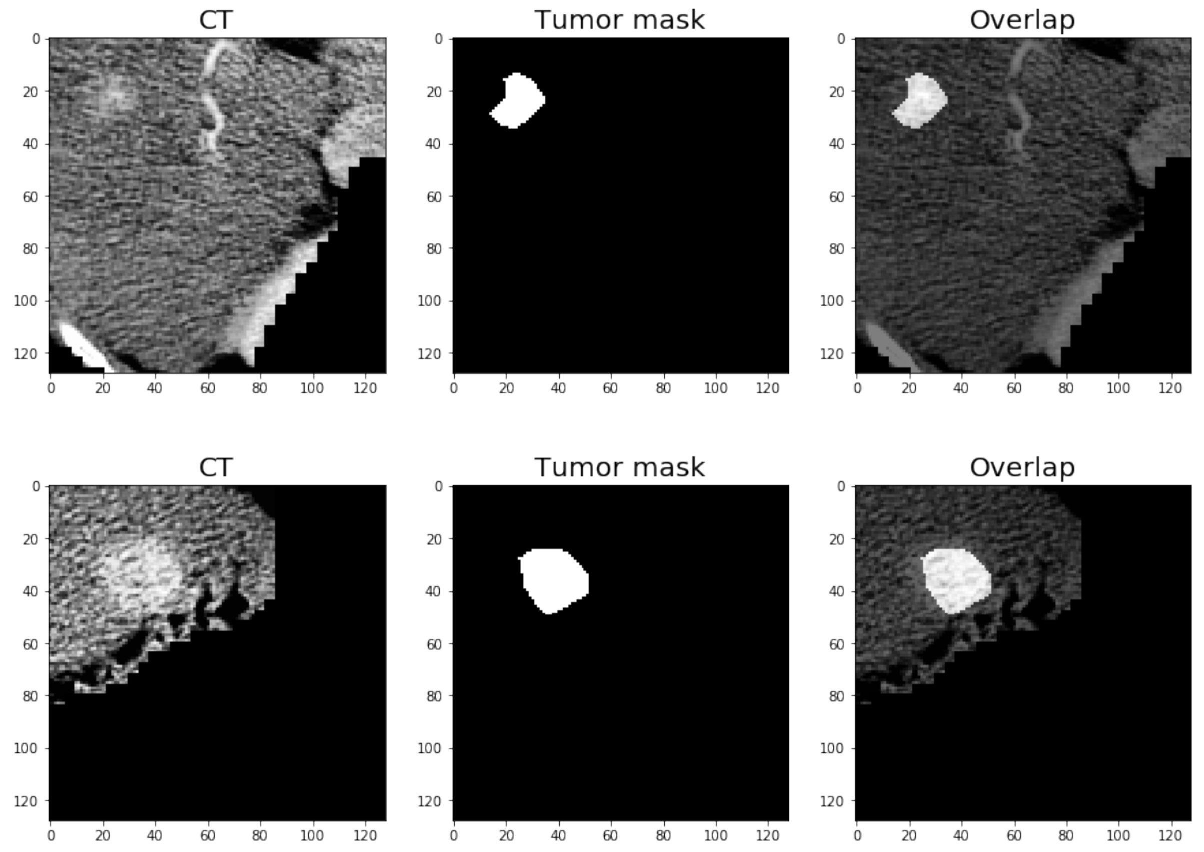


Figure 4.7: Examples of the images of the database used for the training of the automatic tumor segmentation model. From the left: the pre-processed CT image from the *training set*, the tumor segmentation obtained from the ROI, the overlap of the input image and the tumor mask.

4.1.2 Manipulation on *testing set* images

Exactly the same procedure illustrated in Chapter 4.1.1 has been applied also on the images of the *testing set*, with the only exception of the *data augmentation* function, that is unnecessary for the testing purpose.

4.2 Unet model: train and performances

The algorithm trained to achieve the automatic segmentation of the HCC malignancies from the CT images is a Unet with exactly the same architecture as the one described in Chapter 3.2. The important difference is in the choice of the model parameters. In fact, even though the goal is still to train a model able to discriminate between pixels that belongs to two different classes (either “tumor ”or “not tumor ”), in this case the task is made more difficult by the small dimensions of the tumors with respect to the whole image. The model parameters used before did not allow an appropriate learning.

4.2.1 Model parameters

As the previous model, input images have dimensions 128x128x1; differently, both the *batch size* and the *learning rate* are quite smaller: the first is 8 and the latter is set at $5e-5$.

Another difference is *loss function* used. For this task, the Focal Tversky Loss [6] has been selected.

The Tversky similarity Index allows for flexibility in balancing False Positive and False Negative detections. This feature is extremely important with highly imbalanced data such as in case of small ROIs (small HCC lesions): False Negative detections need, in these cases, to be weighted higher than False Positives to improve the recall rate. The Tversky Index is, in fact, defined as:

$$TI_c = \frac{\sum_{i=1}^{N} p_{ic} g_{ic} + \epsilon}{\sum_{i=1}^{N} p_{ic} g_{ic} + \alpha \sum_{i=1}^{N} p_{i\bar{c}} g_{ic} + \beta \sum_{i=1}^{N} p_{ic} g_{i\bar{c}} + \epsilon} \tag{4.1}$$

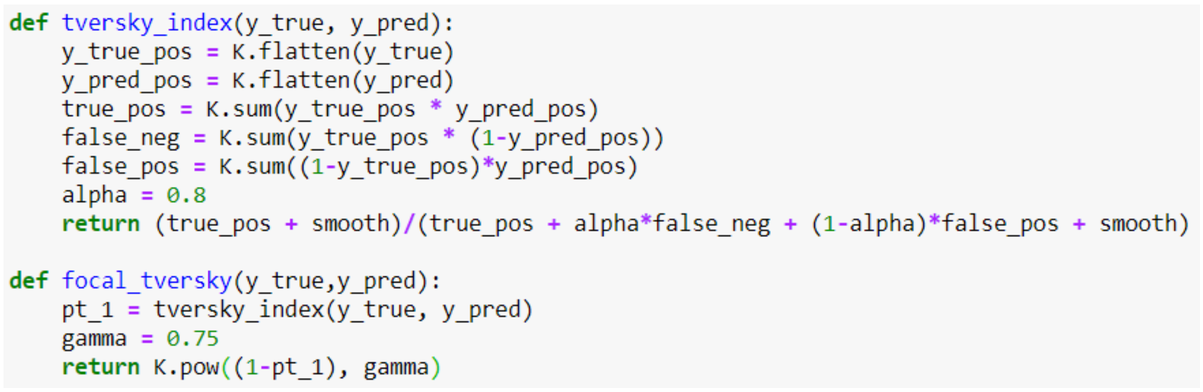
where, $g_{ic} \in \{0, 1\}$ and $p_{ic} \in [0, 1]$ represent the ground truth label and the predicted label, respectively; N is the total number of pixels. In particular, p_{ic} is the probability that pixel i is of the lesion class c and $p_{i\bar{c}}$ is the probability pixel i is of the non-lesion class, $\bar{c}$. The same is true for g_{ic} and $g_{i\bar{c}}$, respectively. Hyperparameters α and β can be tuned to shift the emphasis to improve recall in the case of large class imbalance. Therefore, the value $\sum_{i=1}^{N} p_{ic} g_{ic}$ represents the True Positives; $\sum_{i=1}^{N} p_{i\bar{c}} g_{ic}$ represents the False Negatives and $\sum_{i=1}^{N} p_{ic} g_{i\bar{c}}$ the False Positives.

The Tversky index is adapted to a loss function (L_{TL}) by minimizing $\sum_c (1 - TI_c)$. In addition, to improve the segmentation of small ROIs, the Tversky Loss function can be parametrized by γ for focus on hard classes detected with lower probability. The Focal Tversky Loss (L_{FTL}) function is therefore defined as [7]:

$$L_{FTL} = \sum_c (1 - TI_c)^{\gamma} \tag{4.2}$$

where TI indicates Tversky Index of Equation 4.1 and γ can range from [1,3]. Clearly if $\gamma = 1$, the L_{FTL} simplifies to the L_{TL} and for this purpose it has been set to 0.75. The parameter α is set to 0.8, β is set to 0.2 and the smooth parameter ϵ to 1.

The implementation of the Focal Tversky Loss Function in the code is shown in Figure 4.8.



```python
def tversky_index(y_true, y_pred):
    y_true_pos = K.flatten(y_true)
    y_pred_pos = K.flatten(y_pred)
    true_pos = K.sum(y_true_pos * y_pred_pos)
    false_neg = K.sum(y_true_pos * (1-y_pred_pos))
    false_pos = K.sum((1-y_true_pos)*y_pred_pos)
    alpha = 0.8
    return (true_pos + smooth)/(true_pos + alpha*false_neg + (1-alpha)*false_pos + smooth)

def focal_tversky(y_true,y_pred):
    pt_1 = tversky_index(y_true, y_pred)
    gamma = 0.75
    return K.pow((1-pt_1), gamma)
```

Figure 4.8: Focal Tversky Loss Function's implementation in Python.

The *epochs* are set to 300 but an *early stop* is introduced on the value of the *val_loss* with a patience of 20. In this way, the model achieves optimal training after 39 epochs.

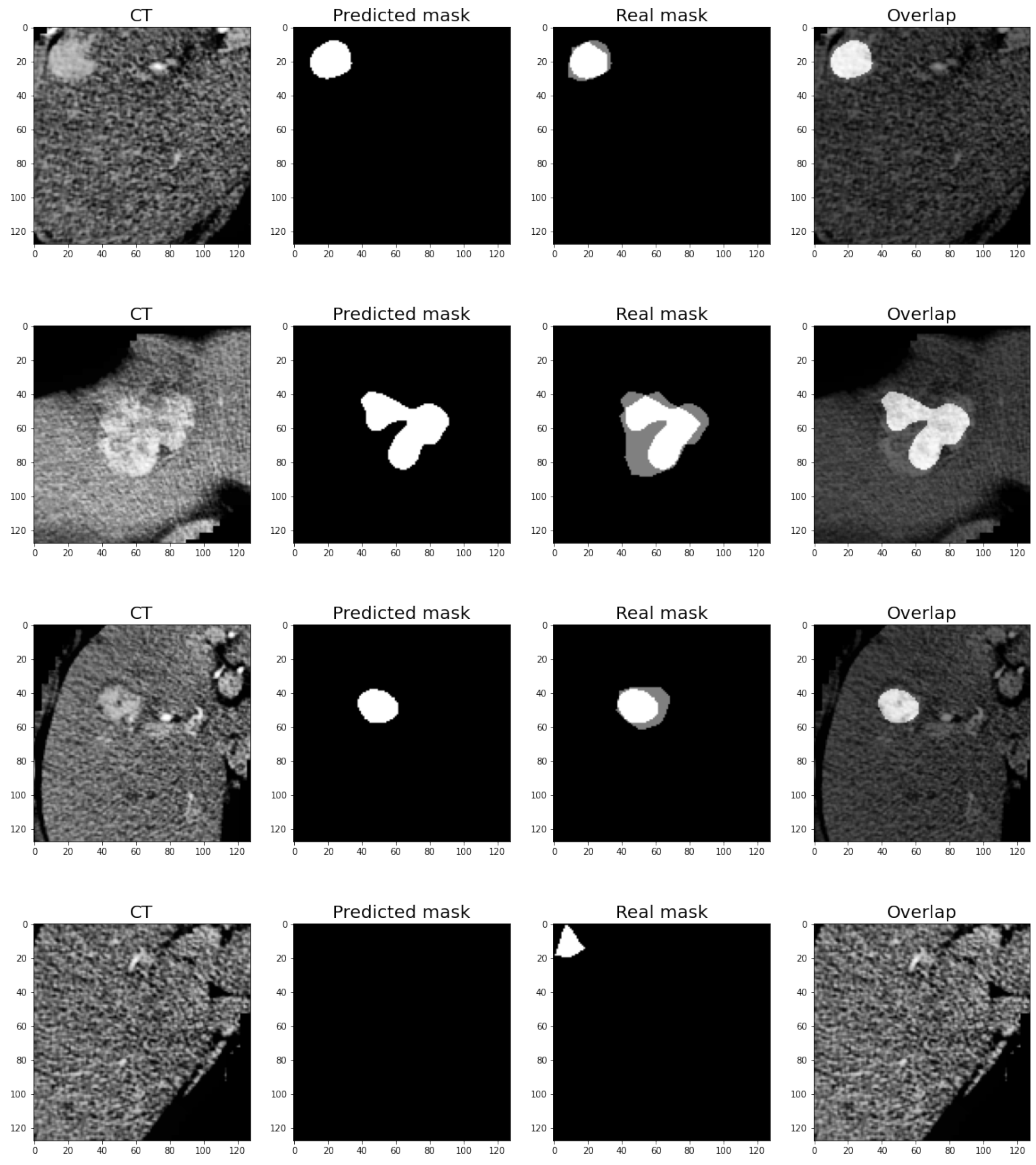
4.2.2 Test

In Figure 4.9 are displayed some results on the *testing set*'s images. In this case, not only the algorithm has never seen these images, but the *testing set* is in this case composed of completely different patients.

The images in the second column represent the segmentations predicted by the model, while the images in the third column are the overlap between the true tumor lesions obtained from the ROIs and their predicted segmentations.

In general, it can be state that the model is almost always able to detect the location of the tumor, but it tends to underestimate its dimensions.

One possible reason to explain this quite low accuracy in the detection can be related to the imprecise transformation of ROIs into masks and to the process followed to obtain the ROIs coordinates in the beginning.


CT
Predicted mask
Real mask
Overlap
CT
Predicted mask
Real mask
Overlap
CT
Predicted mask
Real mask
Overlap
CT
Predicted mask
Real mask
Overlap

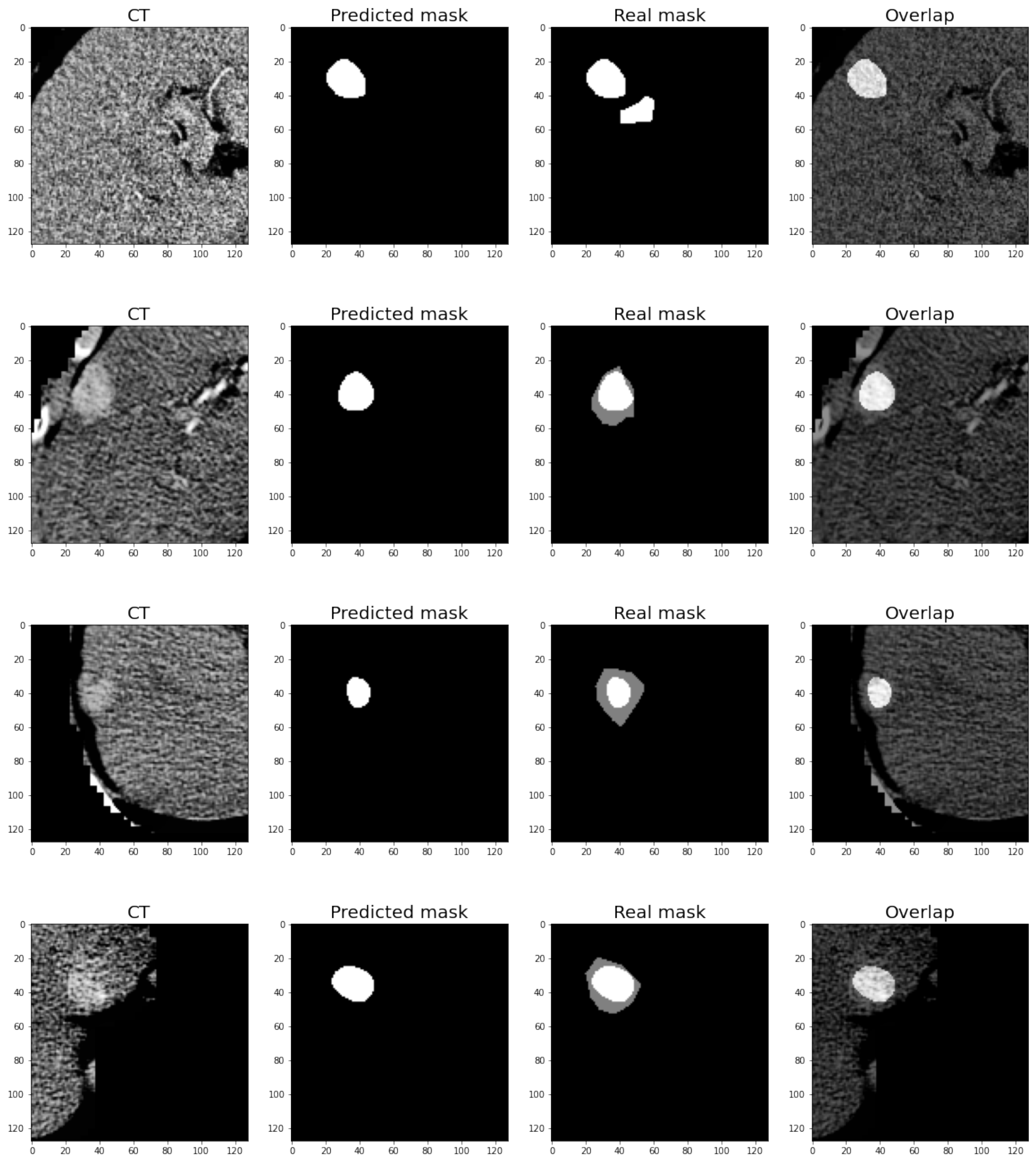


Figure 4.9: From the left: the image from the *testing set*, the tumor segmentation predicted, the overlap between the predicted and the true segmentation (white pixels results to be in common between the two segmentations, while gray pixels belong only to one of them), the overlap of the input image and the segmented tumor.

4.2.3 Performances

The same metrics described in Chapter 3.2.3 have been used to quantify the performances of the tumor model.

Computing these two metrics on the entire *testing set*, the Intersection over Union results to be 0.35 and the Dice Coefficient 0.46. For sure, the values of these metrics are affected by the problems mentioned in Chapter 4.2.2.

Figure 4.10 displays the computation of the metrics on a CT image of the *testing set*. The grey region in the last image represents the difference between the real tumor mask (second image) and the prediction of the tumor according to the model (third image). The values of the metrics for the selected CT slice are reported in the title above the third image.

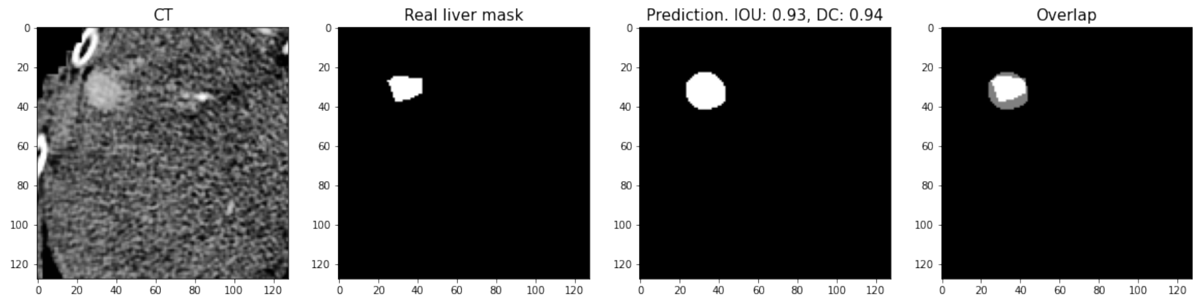


Figure 4.10: From the left: one image from the *testing set*, the true tumor segmentation, the tumor segmentation predicted by the model and the overlap between the predicted and the true tumor. The title of the third image shows the values of the metrics for the image.

Chapter 5

Features extraction and MVI prediction

5.1 Input data

In this section all the patients provided by Sant'Orsola are considered.

The results previously obtained should be used to extract the CT image of the tumor but given the not optimal results achieved for the tumor segmentation, the tumor masks got from the ROI coordinates are used instead. Further studies will be developed to improve the tumor segmentation and complete the passages of Figure 5.1.

The tumor CT obtained in the end can be used for feature extraction.

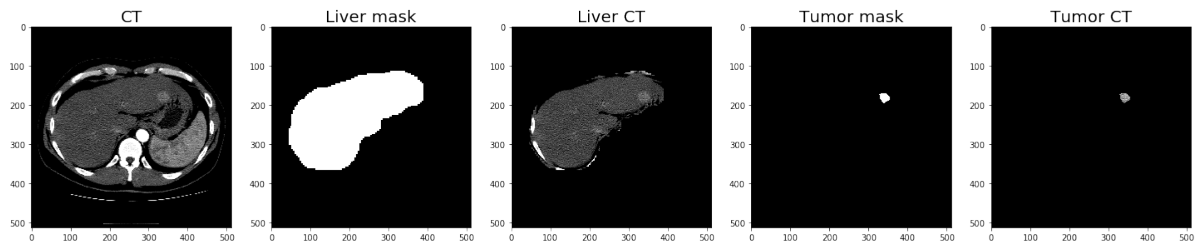


Figure 5.1: Steps followed to obtain the CT image of the tumor, used for features extraction. The initial CT is provided by Sant'Orsola Hospital; the liver mask is achieved as described in Chapters 3; the tumor mask is obtained from the ROI coordinates, as described in Chapters 4; the liver ad tumor CTs are the multiplication between the initial CTs and the respective masks.

At each tumor is then assigned the MVI label (either 0 or 1). This information is contained in the medical record's database supplied by Sant'Orsola Hospital (Figure 2.2) and it consists in the variable that needs to be predicted.

5.2 Features extraction

In order to predict the onset of MicroVascular Infiltration, a set of different features have been considered. The idea is to collect a group of explanatory variables able to describe and identify the tumor and the patient, combine them together and look for a connection between them and the onset of the MVI in those patients affected by those peculiar tumors.

Two groups of features are extracted from the CTs (Haralick features obtained considering the 3D Co-occurrence Matrix and Radiomic features) and a subset of the clinical information provided by Sant'Orsola Hospital is considered.

The goal is to build a model that, given as input information about the HCC tumor and the patient, is able to predict whether the tumor will undergo MicroVascular Infiltration or not. Since the MVI is a binary variable, the task can be seen as a classification problem: the features extracted and collected are used to classify each tumor as "MVI positive", *MVI = 1*, or "MVI negative", *MVI = 0*.

Haralick features

In 1973, Haralick introduced 14 common statistical functions used as texture descriptors for an image. In his article [8] Haralick states that textures are an intrinsic property of any surface, and, as such, they contain important information regarding the structural arrangement of surfaces and their relationships with the surrounding environment. The procedure for extracting texture from an image is based on the assumption that the texture information of the image is contained in the overall or average relationship between the gray tones in the image. This spatial relationship is described by the Gray-Level Co-occurrence Matrix (GLCM). From the computation of the GLCM it is possible to calculate the 14 textural Haralick features that provide information such as homogeneity, contrast, entropy, variance, and so on.

For the purpose of this work, the occurrences of gray level values between each pair of pixels are considered in the three-dimensional space, obtaining therefore a three-dimensional GLCM. In this way, not only the texture of each single bi-dimensional slice is considered, but also the relationship between the successive slices that make up the image.

Radiomic features

An incredibly high number of Radiomic features can be extracted from medical images using the *featureextractor.RadiomicsFeatureExtractor()* function from the *pyradiomics* package [9].

The Radiomic features can be subdivided into the following classes:

- First Order Statistics (19 features)

- Shape-based (3D) (16 features)
- Shape-based (2D) (10 features)
- Gray Level Cooccurence Matrix (24 features)
- Gray Level Run Length Matrix (16 features)
- Gray Level Size Zone Matrix (16 features)
- Neighbouring Gray Tone Difference Matrix (5 features)
- Gray Level Dependence Matrix (14 features)

.

Clinical information

Sant'Orsola Hospital provided a database containing clinical data and information about each patient. In particular, it contains:

- some basic information about the patients (such as age, sex, tumor dimensions);
- radiological criteria (such as TTPVI, tumor margin, ...);
- liver anamnesis information (cirrhosis, steatosis, metabolic syndrome, NAFLD, Child-Pugh Score, MELD Score, HBV, HCV, HIV, Eradicated with DAA, Potus);
- results of histological examinations;
- surgical and post-operative exams;
- follow-up information.

Among all the supplied features, only those known before the onset of the MVI are considered.

5.3 Features selection

Combining all these information together and removing the non-informative ones, it is obtained a database composed of 153 different features, a number of features that largely exceed the number of available samples.
It is important to define a features selection strategy to identify and keep only the most significant features for the problem at hand.

First of all, all the features are standardized by removing the mean and scaling to unit variance. This step can be helpful or necessary, depending on the model chosen for the classification.

Then, a set of models have been used as base estimator for the *SelectFromModel()*'s function, from the library *sklearn.feature_selection*, to select features based on importance weights. Each one of the models tried gave as output a different subset of features.

It has been demonstrated that the subset of features selected by a Logistic Regression model with Lasso regularization and inverse of regularization strength equal to 2.0 guaranteed the best classification performances.

In this way, 23 features (reported in Table 5.1) are selected as the most important ones: 10 of them are features extracted from the CTs (either Haralick or Radiomic features) and the further 13 are clinical information about the patient.

Haralick 3D			Radiomics						
Sum Variance	Informational Measure of Correlation	Maximal correlation coefficient	Minimum	Correlation	SRLGLE	GLNN	LAHGLE	SZNUN	SALGLE
0.191	0.509	0.669	0.029	-0.002	0.269	0.605	0.145	0.307	0.045

Clinic												
TTPVI	Steatosis	Metabolic Syndrome	MELD Score	HBV	Eradicated with DAA	Potus	BCLC stage	Edmondson Grading	Presence of tumor capsule	Oncological radicality	METAVIR	Transplant
1.355	0.658	-0.416	-0.953	0.239	-0.587	-0.953	-0.036	1.618	-0.122	0.194	-0.025	-0.033

Table 5.1: Features selected with the function *SelectFromModel()*, using a Logistic Regression model with Lasso regularization as base estimator.

The first three are obtained by the computation of the 3D-Gray-Level Co-occurrence Matrix; the informational measure of correlation and the maximal correlation coefficient are both measures of the complexity of the texture.

The minimum is referred to the minimum gray-level value within the image region defined by the tumor mask [9]. The correlation expresses the linear dependence of the gray levels between neighboring pixels [9]. The Short Run Low Gray Level Emphasis (SRLGLE) measures the joint distribution of shorter run lengths with lower gray-level values [9]. The Gray Level Non-Uniformity Normalized (GLNN) is a measure of the variability of gray-level intensity values in the image [9]. The Large Area High Gray Level Emphasis (LAHGLE) and the Small Area Low Gray Level Emphasis (SALGLE) measure, respectively, the proportion in the image of the joint distribution of larger, and smaller, size zones with higher, and lower, gray-level values [9]. The Size-Zone Non-Uniformity Normalized (SZNUN) measures the variability of size zone volumes throughout the image, with a lower value indicating more homogeneity among zone size volumes in the image [9].

The BCLC stage and the TTPVI are radiological features assessed by a radiologist; in particular the TTPVI is identified considering the presence of internal arteries and of hypoattenuating halos. Steatosis, Metabolic Syndrome, MELD Score, HBV, Eradicated with DAA and Potus are features that derive from the liver anamnesis. Finally, Edmondson Grading, Presence of tumor capsule, Oncological radicality and METAVIR are the results of histological examinations.

5.4 Pipeline for optimal models train

After identifying the feature subset described in Table 5.1, a set of different machine learning models has been considered with the aim of finding the one with the best performances for the given task: classify each tumor either as "MVI positive", *MVI = 1*, or "MVI negative", *MVI = 0*. In order to compare the performances of these different models and choose the optimal candidate, each model has been validated using an adequate score function or metric.

The best results have been obtained following this pipeline:

- First of all, the dataset composed of the selected features of Table 5.1 has been divided into two disjoint subsets, one for the training and one for the testing part; in this way it is possible to validate the model on a test dataset containing unseen data. The *training set* results to be composed of 67 patients and the *testing set* of 27 patients. While performing the split it is important to check if the MVI labels are equally distributed between the *training* and *testing* subsets, to avoid the situation with test data only containing *MVI = 1* or the opposite. The *training set* contains 23 patients with *MVI = 1* and 44 patients with *MVI = 0*; the *testing set*, 6 patients with *MVI = 1* and 11 with *MVI = 0*.

- Then, the *training set* is divided into *k* folds. *k - 1* folds are used for the actual training of the model and the one left is used for a preliminary validation of the model itself. The process of training and validating is repeated k times. The final performance score is the average of the values obtained at each iteration. For this work, *k* has been set to 5. The workflow of the 5-k-fold cross-validation can be visualised in Figure 5.2.

- Among the different machine learning models trained, the ones with the best performances are those reported in Table 5.2. The cross-validation accuracy is computed with the *sklearn.model_selection*'s function *cross_val_score()* setting '*accuracy*'for the parameter *scoring*. It is the average of the values obtained at each cross-validation iteration. Instead, the accuracy on the testing set is the fraction of correctly classified samples in the *testing set*: it is therefore the accuracy on data the model has never seen before. The regularization strengths for the Penalized Logistic Regression models are set to the values that guarantee the best performances of the models. In Figure 5.3 is shown the implementation in Python.

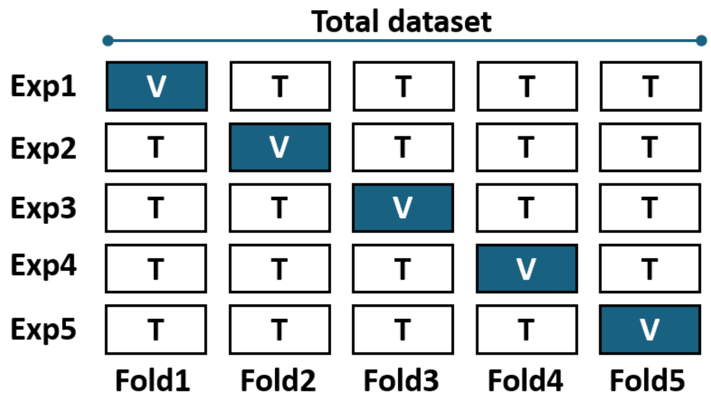
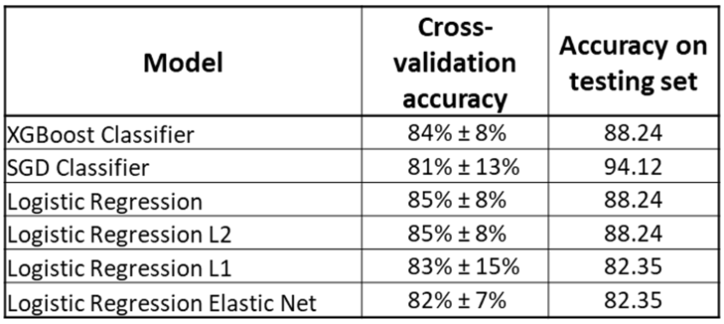
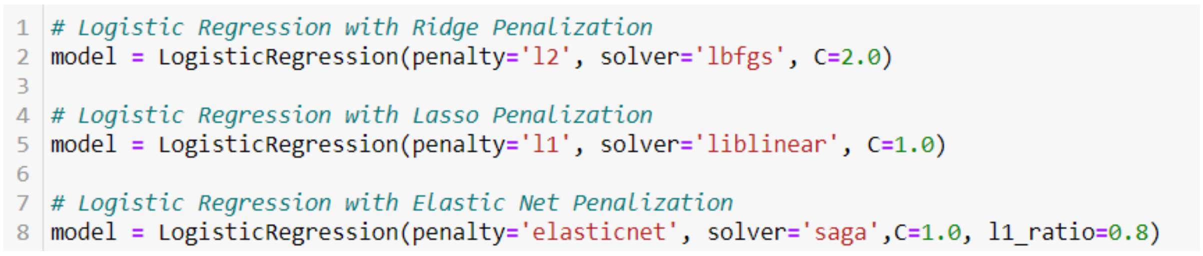


Figure 5.2: Workflow for the 5-k-fold cross-validation: the dataset is divided into 5 folds, then 4 folds are used for training and the one left is used for validation. The process of training and validation is repeated 5 times.



Model	Cross-validation accuracy	Accuracy on testing set
XGBoost Classifier	84% ± 8%	88.24
SGD Classifier	81% ± 13%	94.12
Logistic Regression	85% ± 8%	88.24
Logistic Regression L2	85% ± 8%	88.24
Logistic Regression L1	83% ± 15%	82.35
Logistic Regression Elastic Net	82% ± 7%	82.35

Table 5.2: Machine learning models trained for *MVI* classification with best performances. The cross-validation accuracy is the average of the values obtained at each cross-validation iteration. Instead, the accuracy on the testing set is the fraction of correctly classified samples in the *testing set*.



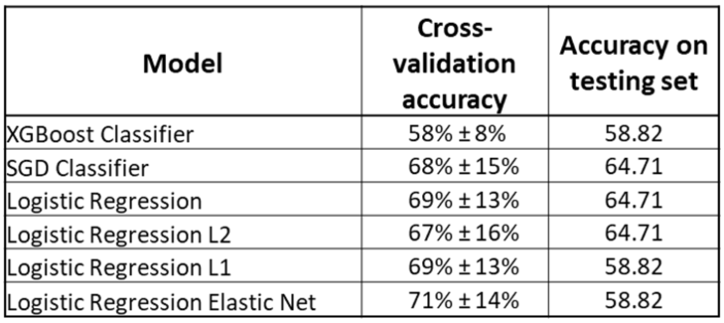
```python
# Logistic Regression with Ridge Penalization
model = LogisticRegression(penalty='l2', solver='lbfgs', C=2.0)

# Logistic Regression with Lasso Penalization
model = LogisticRegression(penalty='l1', solver='liblinear', C=1.0)

# Logistic Regression with Elastic Net Penalization
model = LogisticRegression(penalty='elasticnet', solver='saga',C=1.0, l1_ratio=0.8)
```

Figure 5.3: Choice of the regularization strengths for the Penalized Logistic Regression in Python. The parameter C is the inverse of regularization strength and the parameter *l1_ratio* is the Elastic-Net mixing parameter.

In order to demonstrate the importance of using both clinical and radiomic features, the models of Table 5.2 have also been trained using first only the clinical features and then only the radiomic features. The performances results are reported in Tables 5.3 and 5.4. As it can be seen, the accuracy are much lower with respect to the ones obtained using the mix of features. This means that the combinantion of clinical and radiomic features improves the outcome.

Model	**Cross-validation accuracy**	**Accuracy on testing set**
XGBoost Classifier	58% ± 3%	58.82
SGD Classifier	52% ± 7%	58.82
Logistic Regression	56% ± 14%	70.59
Logistic Regression L2	50% ± 15%	70.59
Logistic Regression L1	58% ± 12%	70.59
Logistic Regression Elastic Net	59% ± 15%	70.59

Table 5.3: Machine learning models trained for *MVI* classification using only clinical features. The cross-validation accuracy is the average of the values obtained at each cross-validation iteration. Instead, the accuracy on the testing set is the fraction of correctly classified samples in the *testing set*.



Model	Cross-validation accuracy	Accuracy on testing set
XGBoost Classifier	58% ± 8%	58.82
SGD Classifier	68% ± 15%	64.71
Logistic Regression	69% ± 13%	64.71
Logistic Regression L2	67% ± 16%	64.71
Logistic Regression L1	69% ± 13%	58.82
Logistic Regression Elastic Net	71% ± 14%	58.82

Table 5.4: Machine learning models trained for *MVI* classification using only radiomic features. The cross-validation accuracy is the average of the values obtained at each cross-validation iteration. Instead, the accuracy on the testing set is the fraction of correctly classified samples in the *testing set*.

Chapter 6

Conclusion

MicroVascular Infiltration (MVI) is an obliged passage in the metastatic evolution of HepatoCellular Carcinomas, and it is supposed to be related to survival of patients affected by these malignancies. Therefore, the prediction of its onset could improve the patients' prognosis.

The purpose of this work was to automatically recognize HCC tumors within the CT of a human liver and predict whether the tumor would undergo MicroVascular Infiltration.

The CT images of 84 patients affected by HCC tumors, a database with medical records about each patient, and two online databases of CT images with segmented liver have been used to build a workflow to gain the aim the project.

This has been done by detecting the liver in the CT image, locating the tumor, extracting the feature of the tumor and combing them with clinical information of the patient.

In particular, two Unet models have been implemented for the automatic segmentation of the liver and the tumor within it. Two metrics have been considered to evaluate the performances of the models: the Intersection-over-Union and the Dice Coefficient. Computing these metrics on the *testing set* portion of the databases used for the liver and the tumor segmentation, the IOU metric results to be 0.82 and 0.35, while the Dice Coefficient 0.88 and 0.46, respectively for the liver and the tumor. The outcomes obtained for the liver automatic segmentation are quite good; the outcomes obtained for the tumor automatic segmentation are, instead, affected by the limitations described in Chapter 4.2.2

The CT images of the HCC tumors are obtained to undergo features extraction.

Combining the 14 Haralick features calculated from the 3D-GLCM, the 120 Radiomic features and the patients' clinical information, a dataset of 153 features have been assembled. Among all the tries, the optimal subset of significant features has been obtained implementing the *SelectFromModel()*'s function for a Logistic Regression model with Lasso regularization.

The dataset of features has been divided into two subsets (for training and testing) to evaluate the classification models also on unseen data. Moreover, the *training set* was used to perform a 5-k-fold cross-validation of the models.

A set of classification models have been implemented, trained and compared, and the performances of the optimal ones are reported in Table 5.2. As it can be seen, the models with the best performances (around 80-84% $\pm$ 8-15%) result to be the XGBoost Classifier, the SDG Classifier and the Logist Regression models (without penalization and with Lasso, Ridge or Elastic Net penalization).

Future steps include the study of the survival probability of the patients affected by the HCC tumors, considering the features extracted from the CTs as well as the patients' clinical records (all described in Chapter 5.2). With this elaboration it will also be possible to investigate and establish the actual correlation between the onset of the MVI and the survival rate of the patients.

Bibliography

[1] Xialing Huang, Jieqin Wei, Xinping Ye, Zili Lv, Ling Zhang, Muliang Jiang, Yidi Chen, Fang Wang, Yuwei Xia, and Liling Long. Imaging and radiomics study of microvascular infiltration of primary liver cancer using a seven-point pathological sampling method. *Research Square*, 2020. doi: 10.21203/rs.3.rs-110032/v1.

[2] IRCAD Hôpitaux Universitaires. Liver segmentation – 3d-ircadb-01. URL `https://www.ircad.fr/research/3d-ircadb-01/`.

[3] CodaLab. Lits - liver tumor segmentation challenge, 2017. URL `https://competitions.codalab.org/competitions/17094`.

[4] Ekin Tiu. Metrics to evaluate your semantic segmentation model, 2019. URL `https://towardsdatascience.com/metrics-to-evaluate-your-semantic-segmentation-model-6bcb99639aa2`.

[5] OpenCV Open Source Computer Vision. Histograms-2: Histogram equalization. URL `https://docs.opencv.org/master/d5/daf/tutorial_py_histogram_equalization.html`.

[6] Shruti Jadon. A survey of loss functions for semantic segmentation. *2020 IEEE International Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, pages pp. 1–7, 2020. doi: https://doi.org/10.1109/CIBCB48159.2020.9277638.

[7] Nabila Abraham and Naimul Mefraz Khan. A novel focal tversky loss function with improved attention u-net for lesion segmentation. *2019 IEEE International Symposium on Biomedical Imaging (ISBI)*, 2019. doi: https://arxiv.org/abs/1810.07842v1.

[8] R. M. Haralick, K. Shanmugam, and I. Dinstein. Textural features for image classification. *1973 IEEE Transactions on systems, man, and cybernetics*, 1973. doi: 10.1109/TSMC.1973.4309314.

[9] van Griethuysen, Fedorov, Parmar, Hosny, Aucoin, Narayan, Beets-Tan, Fillon-Robin, Pieper, and Aerts. (2017). computational radiomics system to decode the ra-

diographic phenotype. cancer research. URL `https://pyradiomics.readthedocs.io/en/latest/`.