applied sciences

MDPI

Article

Mixed-Sized Biomedical Image Segmentation Based on U-Net Architectures

Priscilla Benedetti [1,2], Mauro Femminella [1,3,*] and Gianluca Reali [1,3]

1 Department of Engineering, University of Perugia, Via G. Duranti 93, 06125 Perugia, Italy
2 Department of Electronics and Informatics (ETRO), Vrije Universiteit Brussel, Pleinlaan 2, 1050 Brussels, Belgium
3 Consorzio Nazionale Interuniversitario per le Telecomunicazioni (CNIT), 43124 Parma, Italy
* Correspondence: mauro.femminella@unipg.it

Abstract: Convolutional neural networks (CNNs) are becoming increasingly popular in medical Image Segmentation. Among them, U-Net is a widely used model that can lead to cutting-edge results for 2D biomedical Image Segmentation. However, U-Net performance can be influenced by many factors, such as the size of the training dataset, the performance metrics used, the quality of the images and, in particular, the shape and size of the organ to be segmented. This could entail a loss of robustness of the U-Net-based models. In this paper, the performance of the considered networks is determined by using the publicly available images from the 3D-IRCADb-01 dataset. Different organs with different features are considered. Experimental results show that the U-Net-based segmentation performance decreases when organs with sparse binary masks are considered. The solution proposed in this paper, based on automated zooming of the parts of interest, allows improving the performance of the segmentation model by up to 20% in terms of Dice coefficient metric, when very sparse segmentation images are used, without affecting the cost of the learning process.

Keywords: Image Segmentation; convolutional neural networks; biomedical image analysis

check for updates

Citation: Benedetti, P.; Femminella, M.; Reali, G. Mixed-Sized Biomedical Image Segmentation Based on U-Net Architectures. *Appl. Sci.* **2023**, *13*, 329. https://doi.org/10.3390/app13010329

Academic Editor: Cosimo Nardi

Received: 21 November 2022
Revised: 14 December 2022
Accepted: 22 December 2022
Published: 27 December 2022

1. Introduction

Technological progress and advanced tools for medical analysis have significantly contributed to reducing waiting times for the diagnosis of various diseases. In particular, in oncology, the increase in the number of screening tests associated with the drastic decrease in diagnosis times has contributed to significantly reducing the mortality rate of diseases. Since the late 1980s, diagnostic imaging has been essential to visualizing organs and tissues in detail in order to detect tumors even in their early stages [1,2]. However, the tomographic images of the human body, obtained by CT-Scan (Computed Tomography), require a specialist to manually identify and segment the area of interest.

In the field of image processing, segmentation is defined as the process of decomposing an image into its constituent regions or into the objects that compose it [3]. Since the manual approach to segmentation, still used by a large part of the medical staff, is time-consuming and tedious, some techniques have been proposed to make it automatic. This can be done on the basis of certain criteria concerning the pixels belonging to a region. This is a complex objective, also because the accuracy of the result depends on the type of information to be extracted from the image.

This paper shows a solution, based on Artificial Intelligence (AI) technologies, to automate, speed up and possibly improve the analysis of the images compared to what a human operator could do. This tool incorporates and enhances some known results based on deep learning. In particular, it is based on the use of a deep convolutional neural network (CNN) that allows one to automatically process and analyze multi-scale digital images, known in the literature as U-Net [4]. Our solution includes image processing techniques that improve visualization in terms of quality, such as increasing contrast and provide a valuable tool for the visual analysis of specific morphological characteristics of objects present within the image, such as their perimeter and area. In more detail, the main focus of our research is the definition of an appropriate automatic segmentation method based on the current state-of-the-art and the comparison of its performance across multiple types of images at different scales. The developed neural network allows processing and identifying a wide typology of anatomical organs, following an adequate training of the model using the CT-Scans of the patients. This feature determines a very versatile algorithm behavior, extensible to any organ. However, if the loss function used to drive the training process cannot suitably capture the information present in the mask of the organs at different scales, the resulting prediction could be unsatisfactory in operation.

The main contribution of this paper is to show how to improve performance in the presence of target images characterized by very sparse signals without significantly improving the cost of the learning process and without introducing different learning algorithms. Furthermore, by making slight changes to the model parameters, the algorithm could be used not only in the medical field, but also in other fields that require semantic segmentation of images. In fact, the applied techniques have universal value, although in this paper they have been treated in relation to the prefixed purpose. The contribution of this paper is twofold:

- *First*, a performance analysis of the baseline segmentation model on different types of organs is shown. In some case studies, the model provides good results. However, the performance is often unsatisfactory when small organs or restricted regions of interest, which are important in the diagnosis of serious pathologies, are considered [5].
- *Then*, a proposal for improving multi-scale segmentation, based on lightweight image preprocessing is shown. This proposal leads to 20% improvement in the score evaluated by using a metric based on the Dice coefficient [6], also known as F1-score [7]. This method can be generally applied to any target image characterized by a sparse binary mask.

The remainder of this paper is organized as follows: Section 2 presents some insights on the background and challenges motivating this work, along with the related works. Then, Section 3 presents the implemented model and the relevant parameters. Section 4 includes the results obtained by using the proposed methodology. Some final remarks are reported in Section 5.

2. Background and Related Works

2.1. Segmentation of Medical Images

In radiology, a CT-Scan is a diagnostic technique used for reproducing sectional and three-dimensional images of the human body. Images are obtained from the computerized analysis of the information present in the X-ray scans. Since each image is the projection of the object itself from one of multiple angles, it is possible to reconstruct three-dimensional objects.

In general, medical Image Segmentation can be useful for multiple purposes [8], such as:

- To diagnose conditions, including damage and injury to internal organs and bones, stroke, cancer and problems with blood flow;
- To guide tests and treatments. For example, before radiotherapy treatment, a CT scan is performed to determine the location, size and shape of the tumor to be treated;
- To monitor the evolution of patients' conditions, such as the presence and size of the tumor during and after specific treatments.

In order to quantitatively describe radiodensity, the Hounsfield scale, or Hounsfield Unit (HU), is typically regarded as the reference unit of measurement [9]. The HU values give preliminary information on the nature of the observed tissues.

Image Segmentation is a technique of partitioning an image into distinct and meaningful parts, called segments. The purpose of this process is to simplify and change the representation of images, for identifying and extracting some objects of particular interest and making it easier to analyze individually. In fact, it is particularly useful in applications of computer vision, such as object recognition, image compression and analysis of digital image content. In the medical area, segmentation is useful for many purposes, such as to locate and identify tumor cells, measure tissue volumes, perform virtual surgery simulations and intra-surgical navigation and integrate slower and more subjective manual human labor.

It is possible to categorize segmentation techniques into three classes: clustering, edge detection and region extraction [3]. The two macro problems to avoid are sub-segmentation, which means merging semantically different objects in the same area and over-segmentation, or the subdivision of the same object into multiple areas. In order to make algorithms autonomous, they should not assume any prior knowledge of the image to be available; otherwise, it would be difficult to ensure that results are satisfactory for any type of analysis. It is also possible to obtain the segmentation of the image in a number of regions, such that each of them is homogeneous and coherent with respect to a certain criterion and at the same time their union returns the original image. To this end, the following formal definition of the segmentation problem is given [10]: let X be the image domain and let P be a homogeneity predicate, that is a feature extracted from the image and associated with each pixel, defined on a set S of connected pixels of X. The segmentation of X consists of the partition of X into a number of n sub-images or regions S_i, with $i = 1, \cdots, n$, such that:

$$\bigcup_{i=1}^{n} S_i = X \tag{1}$$

$$\begin{aligned} S_i \cap S_j = \emptyset \qquad & \forall (i,j)(i \neq j) \\ P(S_i) = true \qquad & \forall i = 1, \cdots, n \\ P(S_i \cup S_j) = false \qquad & \forall (i,j)(i \neq j) \end{aligned}$$

However, the partition of an image into homogeneous regions with respect to certain characteristics does not guarantee the correct subdivision into semantic objects, especially in particularly complex images. In this regard, it is not certain that the automatic segmentation of images admits a single solution that is also robust. Nevertheless, it is possible to reach the expected result with a good approximation by implementing a multi-level convolutional model [11].

2.2. Convolutional Neural Networks

CNN is an artificial neural network architecture widely used in deep learning for image processing. It analyzes images through artificial neurons placed in three dimensions, called channels: height, width and depth [12]. It is specialized to detect and classify images and extract their features, such as corners and edges. For example, its ability to recognize objects allows the detection of tumors, in the medical field, and of obstacles, in the field of autonomous car driving. A convolutional neural network has a structure consisting of multiple levels of feature detectors:

- Convolutional layer;
- Non-linear activation functions;
- Pooling;
- Fully connected network (optional);
- Dropout layer (optional).

Our proposal makes use of the results presented in the pioneering paper [4], in which the author proposes a particular CNN, called U-Net, to solve the problem of automatic semantic segmentation of biomedical images. While a CNN is implemented in order to learn the feature map of an image and exploit it to create a more nuanced mapping by converting the image itself into a vector, when segmenting an image it is also necessary to reconstruct the image from this vector obtained. This task is particularly difficult as it is more complex to convert a vector to an image rather than the other way around. The basic idea of a U-Net is to take advantage of the functionality mapping, learned during the conversion of an image into a vector, and use it to reconstruct the output image. In this way, the integrity of the image structure is preserved and distortion is greatly reduced.

The resulting structure includes elements typical of deep learning tools, such as convolutional layers and max pooling. Figure 1 shows the structure of the U-Net, with the parameters used in this work. Details about the configuration that has been actually used in this work are given in Section 3.2.

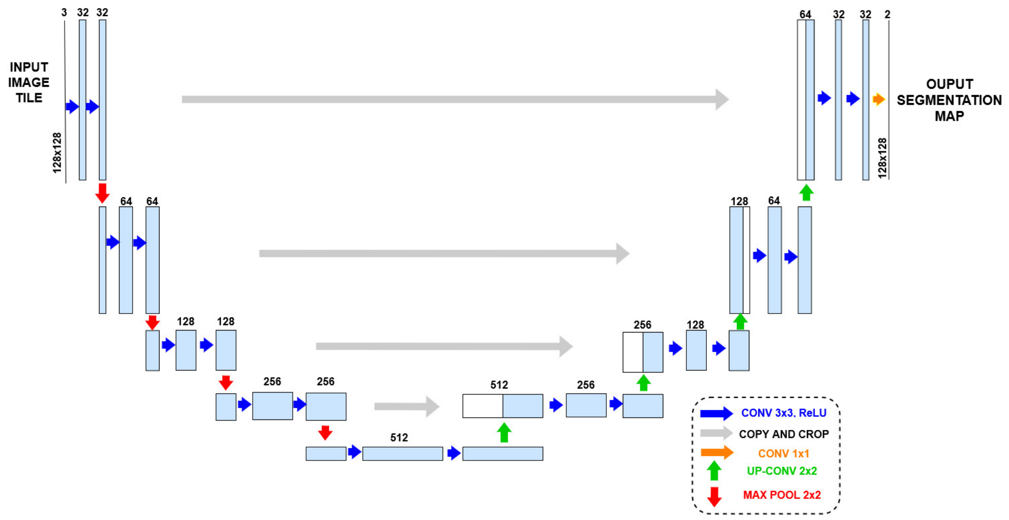

Figure 1. Model of the U-Net architecture used in this work. Each blue rectangle represents a multi-channel feature map. The number of channels is written above each box and represents the value used in the experiments. The white boxes represent the copied feature maps. The arrows denote the type of operation performed.

2.3. Related Works

In [13], the authors present a deep learning based segmentation model that relies on image-specific fine tuning. The presented model performs bounding based binary segmentation with a P-Net [14], a structure adapted from VGG-16 [15], a well known CNN model. The proposed model shows good performance, but it requires an extended number of training iterations and samples to reach good performance. Ref. [16] makes use of a modified U-Net network [4] for Image Segmentation. After an initial phase of image augmentation, the proposed network shows a good performance, but it is focused only on liver images. Other medical images with more sparse binary masks are not considered. An interesting analysis of the impact of the different parts of the U-Net architecture on segmentation accuracy is presented in [17]. In such a paper, the authors propose a reduced version of the U-Net network that sensibly reduces the number of operations required for segmentation. Nevertheless, this model does not always reach the same performance results of the standard U-Net network. Ref. [18] presents an exhaustive survey of the state-of-the-art U-Net-based Image Segmentation, with its numerous application fields. In [18], the need for improvement techniques without relying on extended datasets is mentioned as a research challenge.

For what concerns the segmentation of images with a sparse target signal, ref. [19] adds the attention mechanism ECA-Net (Efficient Channel Attention Neural Networks) into the standard U-Net architecture in order to improve the ability of the model to segment small items in the target images. This approach was applied on insulator string images and was not tested for biomedical Image Segmentation. In [20], the performance of U-Net for COVID-19 lesion segmentation on lung CT-Scans is compared with the achievable performance of another deep learning model, SegNet. Although SegNet produces better lesion detection accuracy, U-Net turns out to be better for multi-class segmentation. A solution for segmentation tasks dealing with organs of highly varying dimensions is proposed in [21]. The authors consider segmentation in head and neck (HaN) CT images, which is characterized by the presence of big and small organs. The proposed solution consists of combining a standard deep learning network for 3D images, the S-net, with a smaller one specialized for smaller segments. The main network provides the secondary one with the probabilistic location of the small organs in the HaN samples. This ensemble, called FocusNet, produced good performance using the publicly available MICCAI 2015 Head and Neck dataset. Nevertheless, it requires the introduction of a new module, which increases the model complexity and the training effort. In this regard, a lightweight solution based on data pre-processing is proposed. Similarly, ref. [22] shows a solution for small organ segmentation, considering whole-body Magnetic Resonance Imaging (MRI) scans. With the implementation of a two-staged fully CNN, a coarse-scale segmentation is first executed and then refined in order to refine the segmentation of the considered organs. Although the proposed method has a 50% gain with respect to the the state-of-the-art for small organ MRI segmentation, the overall performance is still unsatisfactory, with a 0.56 Dice similarity coefficient. Finally, blood vessel segmentation is investigated in [23,24]. In [23], a U-Net is used to perform coronary artery stenosis detection on X-ray coronary angiograms, including a module that leverages the temporal consistency of consecutive frames to limit the number of false positives. Ref. [24] is focused on the segmentation of coronary angiograms. The proposal relies on a nested encoder–decoder architecture named T-Net, which produced an accuracy of 83%. None of these mentioned proposals were used for CT-Scan blood vessel segmentation.

Finally, it is worth mentioning that a significant effort has gone into using annotated 2D organ sections to provide a three-dimensional representation of organs. In this regard, the 3D U-Net [25] represents a pioneering paper, extending the previous achievements of Olaf Ronneberger et al. obtained with the initial U-Net [4], achieving good results. However, this goal is different from that of our research, which investigates automatic segmentation of 2D CT-Scan slices that, in any case, are fundamental for also building the volumetric segmentation, if needed.

3. Dataset, Model and Parameters Configuration

3.1. Dataset

This paper shows the performance achievable by using the U-Net architecture for medical image segmentation during both the training and test phases. For this purpose, a publicly available dataset, the *3Dircadb1* dataset (https://www.ircad.fr/research/data-sets/liver-segmentation-3d-ircadb-01/, accessed on 1 November 2022), was used. It includes 3D CT-Scan images of 10 women and 10 men with liver tumors in 75% of cases. The dataset is anonymized. For each patient, it includes masks of different organs. For our analysis the images of three different organs available in the dataset have been selected. They are characterized by very different shape, size, compactness and mask features: liver, bone and portal vein. The difference between these organs and relevant masks allows performing a multi-scale and multi-shape analysis of the achievable U-Net performance. In fact, while liver binary masks show a considerable amount of information (for instance, with reference to the slice in Figure 2, see the white portion in Figure 3a), in case of bone and portal vein, a gradual and significant decrease of information appear in their masks. In such situations, the shortage of the available white pixels, representing the information needed for training the network, limits the effectiveness of the segmentation process. Thus, processing images of these different organs allowed us not only to test the effectiveness of the U-Net, but also to introduce simple yet effective strategies to overcome such limitations.

The selected three organ images are provided for all the patients of the dataset. Each patient's folder includes four subfolders, in which the DICOM-formatted images, the labelled images, the mask images and the surface meshes can be found.

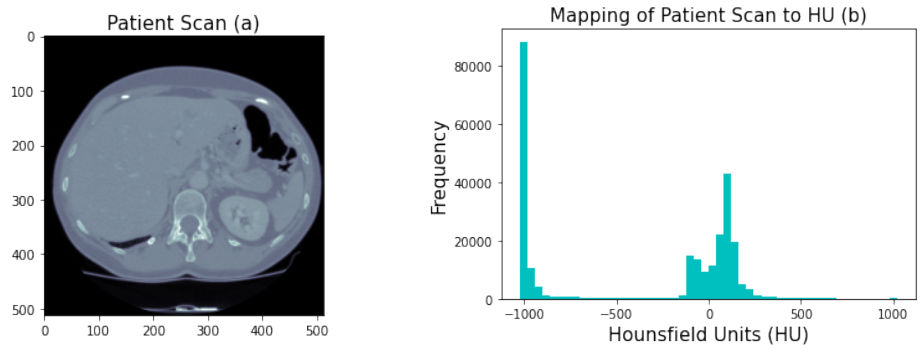

Figure 2. CT-Scan plot from '*PATIENT_DICOM*' (**a**) and corresponding histogram of HU values (**b**).

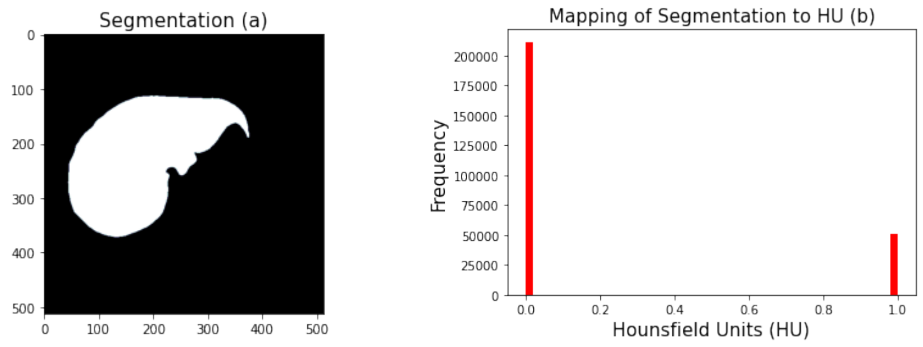

Figure 3. Binary mask plot from '*MASKS_DICOM*' (**a**) and corresponding histogram of HU values (**b**).

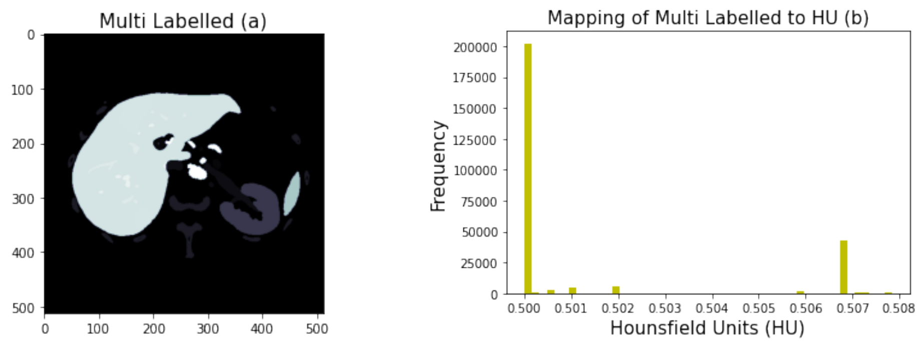

Figure 4. Multi segmented mask plot from '*LABELLED_DICOM*' (**a**) and corresponding histogram of HU values (**b**).

All images are of the same size, equal to 512 $\times$ 512 pixels. To process the dataset, 20 folders have been used, each one corresponding to a different patient. These folders are called '*3Dircadb1.number*', with *number* varying between 01 and 20 and each of them, containing the following four subfolders:

- '*PATIENT_DICOM*': The patient's images in DICOM format. Figure 2a shows an example of a complete CT-Scan slice, with relevant frequency of HU values reported in the companion histogram of Figure 2b;
- '*MASKS_DICOM*': A set of subfolders corresponding to the names of the various segmented organs, containing the binary DICOM image of each original mask. Figure 3a shows an example of a liver mask, with the relevant frequency of HU values (corresponding to black and white in this case) in the histogram of Figure 3b. In this subfolder, the segmentation masks of liver, bone and portal vein used in this work can be found for each patient;
- '*LABELLED_DICOM*': The ensemble of segmented images corresponding to all the patient's analyzed organs, including the ones considered in this work (liver, bone and portal vein), in DICOM format. Figure 4a shows an example of a multi-segmented

mask, with relevant frequency of HU values (one for each mask) shown in a histogram in Figure 4b;

- '*MESHES_VTK*': All the files corresponding to the surface meshes of the various areas of interest in Visualization Toolkit (VTK) format.

3.2. Model Architecture

Our proposal makes use of the U-Net, a new CNN presented in the pioneering paper [4] to solve the problem of automatic semantic segmentation of biomedical images. U-Net is a CNN whose architecture was adapted in order to make use of a reduced image dataset while continuing to produce fairly precise segmentation. Figure 1 shows the structure of the U-Net used in this work, together with the relevant parameters. The architecture is divided into two *paths*. The path on the left is called "*Contraction Path*" or "*Encoder Path*", while the one on the right is referred to as "*Expansion Path*", or "*Decoder Path*". In the middle data, concatenations are performed, indicated by the grey arrows in Figure 1. They implement acquisition of localized information from the feature maps.

The *Contraction Path* consists of a certain number of contraction blocks followed by a max 2 × 2 pooling. Each contraction block downsamples the input image, received from the previous level, in a feature map, applying two levels of 3 × 3 convolution. The number of kernels, or feature maps, after each block doubles, so that the architecture can effectively learn the complex structure of the input image. In this case, the input image is 128 × 128 pixels and 32 features are used in the first step. Parameters, such as the size of the input image and the number of features, can be changed according to the architecture to implement, as some criteria may not make the network work properly. Moving further and further down, the bottom level averages between the two paths and uses two CNN 3 × 3 levels followed by a level called "*up convolution*" 2 × 2.

Subsequently, the *Expansion Path* section, which also consists of a series of expansion blocks, allows the network to propagate information from the lowest resolution level to higher ones. This way, it amplifies (upsamples) the final image feature map. Each block passes the received input to two 3 × 3 convolutional levels and a 2 × 2 upsampling level. Symmetrically to the left branch, after each block the number of feature maps used by the convolutional layer is halved. However, as shown by the gray arrows, each time the copied feature map of the corresponding contraction level is added to the input. This ensures that the features and information learned while shrinking the image are subsequently used to reconstruct it correctly. Clearly, the initial levels of the encoder contain more information, so they guarantee a significant boost in the up-sampling process, allowing the recovery of details and significantly improving the result. The architecture, being symmetrical, is such that the number of expansion blocks is equal to the number of contraction blocks. Going up to the final level of the expansion path, the resulting map passes through the final level of 3 × 3 convolution, with the number of feature maps equal to the number of desired segments. Hence, the same feature map used for the contraction is then used to expand the vector and obtain the output image, which represents the segmentation of the input image.

3.3. Model Implementation

To train the classification algorithm, implemented with Tensorflow (https://www.tensorflow.org/, accessed on 1 November 2022), the CT-Scan records of 17 patients were used, consisting of 2445 images (and related masks), encapsulated in DICOM files (Digital Imaging and Communications in Medicine, https://www.dicomstandard.org/, accessed on 1 November 2022). The image of a single slice of a CT scan is referred to as a *sample* throughout the paper.

After importing the appropriate packages, the functions for loading and importing DICOM images with related tags are implemented. The function code is shown in Appendix A. By using the `process_path(filename)` function, it is possible to load the DICOM image and the relevant mask for each file path. In the specific example, the one necessary for the *liver* segmentation is shown: It is necessary to convert the sample values to the unit of measurement for CT-Scans, that is the HU, since, by default, the values returned from the first upload (resulting from `decode_dicom_image()`) are not expressed in the HU units. This transformation is linear and, if stored in the DICOM header at the time of image acquisition, the relevant slope and intercept can be recovered by using the tag codes (00281052, 00281053) in a completely automated way. This function is subsequently applied to each TensorFlow dataset containing the list of paths related to images, for each folder, using the `map()` method. The use of tensors is preferred in order to model and process data. In fact, through the representation in computational graphs, they facilitate parallel computing, making full use of the computing resources used (Graphic Processing Units, GPUs). Furthermore, with the use of tensors, the computation of derivatives, fundamental in the learning process of a neural network, is accelerated.

Once all the functions for image processing are defined, these methods are invoked in order to view the data collected so far. For example, Figure 2 shows an example of the images present in the 3Dircadb1.1 [26] folder of patient n.1 (a) and the associated histogram (b). Through the histogram, it is possible to see how many pixels correspond to air and how many of them to tissues. Looking at the histogram, it emerges that a lot of air is present, there is an abundance of soft tissue, mainly muscle, liver and part of the lung, but also fat. Only a small piece of bone is present in the CT scan, which appears as a tiny sliver, difficult to appreciate in the HU histogram (expected values around 400 HU) due to the small number of relevant samples.

Once data are processed, a further operation is required on the arrays of the training and test sets, in order to scale the size of the images from 512 $\times$ 512 pixels to 128 $\times$ 128 pixels and add a dimension to the channel (the gray scale, which is 1), since the built model accepts images with this resolution as input. For what concerns the construction of the actual model, it is made up of 10 convolutional levels that outline the architecture of the U-Net, described in the previous section. In addition, dropout levels are inserted after each pooling level, as the amount of information in the considered dataset is quite limited. In fact, in the absence of dropout, the training of the algorithm would require more effort to converge due to overfitting (overfitting is a phenomenon that happens when a statistical model excessively adapts to its data during the training process, thus losing generality [27]). To limit the risk of overfitting, possible solutions consist of increasing the volume of data, reducing the complexity of the architecture or adding regularization. The latter solution is implemented by adding the dropout after each level of convolution, keeping in mind the loss of information as a consequence: If part of the information is lost in the first level, it is lost for the entire network. Therefore, the final scheme of the neural network that we present starts with a low dropout rate in the first few levels to limit the loss of information, which gradually increases to limit overfitting.

The model structure is implemented by using the Keras library (https://keras.io/, accessed on 1 November 2022). In particular, the `get_model()` function is created, which receives as arguments the optimizer, the loss function, the metric and the learning rate selected. The code (Listing 1) below illustrates the first level of convolution of the Contraction Path:

Listing 1. Keras model creation: Encoder Path's first level.

```
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(0.5)(pool1)
```

The first two lines of code create a convolution kernel, which processes the received input to produce an output tensor. First, the set of images coming from the outside is used as input, then the output of the first convolution becomes the input of the second. The constructor arguments used are as follows: Number of filters (integer power of 2), from which they learn the convolutional levels, kernel size (an integer or a tuple of two odd integers), which specifies the height and the width of the 2D convolution window, `activation`, which specifies the activation function to apply after performing the convolution and padding value (`padding`), which specifies, based on the set value, whether the size of the input volume was changed. In this work, the ReLU (Rectified Linear Activation Unit) activation function is used, while the padding value is the 'same', which means that the spatial dimensions of the input are maintained, so that the volume of the output has the same size. Therefore, reliance on padding, image shrinkage or information loss is avoided.

ReLU is a simple function with better performance than other activation functions, such as Sigmoid and Tanh. Its equation is as follows [28]:

$$R(x) = max(0, x). \tag{2}$$

It returns 0 if it receives a negative input, while for any other positive input value x, it returns the same value. In fact, the function does not perform any complex calculations and therefore the model takes little time to train and converges very quickly. Another advantage is sparsity, which implies better predictive power and less model overfitting, as it increases the likelihood that neurons are actually processing meaningful data. This occurs since in neural networks, such as the one under consideration, matrices have many zero cells and for this reason they are called 'sparce matrices'. Therefore ReLU, by providing zero output for negative inputs, makes sure that the network is sparse and that neurons are not turned on to process unmeaningful data. However, the phenomenon of 'dying ReLU' could occur. A ReLU neuron is dead if it is stuck on the negative side and always returns 0, i.e., once it goes negative, it is unlikely to recover. This problem can occur when the learning rate is too high and a lower rate could solve this issue. The learning rate used in our experiments is equal to 10^{-3}.

In the first level, the aim is to reduce the spatial dimensions of the output volume through MaxPooling, applying it to the output of convolution. MaxPooling is a pooling operation that selects the maximum element from the area of the feature map covered by the filter, sub-sampling the input along its spatial dimensions. The window size is set with the `pool_size` argument, which is an integer or a tuple of two integers (if only one integer is specified, it is used for both the height and width of the window). Therefore, the output after this level is a feature map containing the most important characteristics of the previous input feature map.

Finally, as already mentioned, a dropout level is added, which receives the result of the MaxPooling as input. Inputs to dropout not set to 0 are scaled by 1 / (1 − `rate`), so that the sum across all inputs is unchanged [28]. The `rate` value is the fraction of the input units to be released. It is in the range between 0 and 1; in particular, it is set to `rate = 0.5`, since, as mentioned above, it is good to lose only a small amount of information in the first level.

To complete the Contraction Path, the other four levels of convolution are implemented in the same way as the first one, except for the input of the first convolution of each level, with the input, in this case, being the output of the previous level dropout. In level 5 of the Contraction Path, both pooling and dropout are not necessary as they are the final level of the structure (see Figure 1). The Decoder Path is built symmetrically, with the same number of levels as the Encoder Path. The syntax of the code (Listing 2) is as follows:

Listing 2. Keras model creation: Decoder Path's sixth level.

```
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)
```

In level 6, upsampling takes place first. It consists of a concatenation which takes a list of tensors as input, all of the same shape except for the concatenation axis (`axis = 3`). It returns a single tensor, which is the concatenation of all inputs [28]. The first item in the list is the result of the transposed convolution made on the output of the level 5 convolution, while the second is the output of the level 4 convolution. The transposed convolution performs an inverse transformation to the normal convolution made in the descent path, since in this way the output begins to be generated which has the same shape as the original input, while maintaining some connectivity with the shape of the convolution output. The parameters of this operation are the number of filters, the kernel size, padding, just like a normal Conv2D, and also the number of steps (`strides`) of the convolution along the height and width (an integer or a tuple of two integers). In the code `strides = (2, 2)` is set; that, is the filter is moved 2 pixels horizontally for each reading from left to right, then down 2 pixels for the next row, establishing the outputs in the feature map. During this operation, Conv2DTranspose learns during training and attempts to fill in the details as part of the upsampling process to resample the original input. Then two normal Conv2D convolutions are executed: the first takes the result of the upsamplig concatenation as input. Its output becomes the input of the second convolution. At the end of the first level of the Decoder Path, the U-Net structure continues with three other levels that have the same shape as the one just described. Finally, the tenth and final level of the network consists of a single convolution, in which the activation function (Listing 3) is the 'Sigmoid' [29] and no padding is done:

Listing 3. Keras Model creation: level 10.

```
conv10 = Conv2D(1, (1, 1), activation='sigmoid')(conv9)
```

The final step to complete the definition of the `get_model ()` function is the configuration of the model for training (Listing 4), which is also used for its validation:

Listing 4. Model configuration.

```
model.compile(optimizer=optimizer(learning_rate=learning_rate),
                        loss=loss_metric, metrics=metrics)
```

3.4. Parameters Configuration

This section analyzes the specific arguments that are passed to the `get_model ()` function and the results obtained.

As mentioned above, the hyperparameters that allow controlling the model training process are the optimizer, the loss function and the metric. Following an experimental optimization process to identify an optimal configuration from a performance point of view, which minimizes a predefined loss function on test data, the following parameters were selected:

- Optimizer = '`Adam`';
- Learning Rate = '`1e-3`';
- Loss Metric = '`dice_coef_loss`';
- Metric = '`dice_coef`'.

3.4.1. Adam Optimizer and Learning Rate

The optimizer selected implements the Adam algorithm [30], acronym for Adaptive Moment Estimation, for the optimization of the descent of the stochastic gradient, applying the principles of the RMSProp and AdaGrad optimizers [31].

Gradient descent is a technique used for determining the global maximum and minimum points of a function of several variables. The stochastic approximation is applied when the cost function is too expensive at each iteration and breaks down the addends at each iteration into a sum. In the context of artificial neural networks, the descent of the gradient evaluates the model by using an input corresponding to a known output and corrects each parameter of the model in a proportional quantity (but of opposite sign) with respect to its contribution to the result error.

The Adam algorithm is computationally efficient, requires little memory and is suitable for handling large data volumes or numbers of parameters. The fundamental parameter that Adam receives is the learning rate, that is, a floating point value that indicates the size of the passage at each iteration and that influences the criterion for evaluating whether the newly acquired information is more important than the past information item. Therefore, the learning rate must be neither too high—otherwise, learning will jump above the minima of the loss function; nor too low—otherwise the convergence will happen too slowly, with the possibility of getting stuck in an undesired local minimum. During the training phase, it is advisable to adjust and adapt the learning rate in the right way as it does not change, but remains unchanged throughout the execution of the model. For the proposed model, the value of the learning rate was set at the default value recommended for the Adam algorithm, equal to 10^{-3}. Then, the optimal value was determined experimentally, by varying it in the range $[10^{-4} \div 10^{-2}]$, using a logarithmic spacing for the search process.

From the relevant literature, the good robustness and speed of Adam, compared to other optimizers, emerges [32]. However, performance comparisons of various optimizers are strongly dependent on the specific workload and hyperparameter tuning. Hence, Adam has been selected after a preliminary experimental comparison with alternative optimizers such as SGD and Adagrad. In our analysis, Adam provided the best performance in a limited training time. Moreover, coherently with our results, the Adam optimizer is widely used in neural networks for Image Segmentation [17,33–35], since it provides better results compared with some alternatives even with varying network architectures [33].

3.4.2. Metric and Loss

A metric function based on the Sørensen–Dice coefficient [6] is used in the model. It is often referred to as the Dice coefficient and is equivalent to the F1-score [7]. It is a statistical index that measures the similarity between two sets of data. In particular, in the context of Image Segmentation, it compares the estimated output of the algorithm with the known reference masks, measuring the affinity between two binary images. The `dice_coef()`, used as accuracy metric in List 4 (model configuration), is based on the following equation:

$$DSC = \frac{2 \mid X \cap Y \mid + smooth}{\mid X \mid + \mid Y \mid + smooth}. \tag{3}$$

where $|X|$ and $|Y|$ are the sizes (expressed as number of elements) of the two sets X and Y. In this case, X and Y represent the sets of white pixels in the masks generated by the U-Net and in the reference one, respectively. The DSC is the ratio of the double of the number of elements common to both sets (size of intersection set) and the sum of the size of the two sets. The expression returns a value between 0 and 1. A Dice coefficient equal to 1 denotes a perfect and complete overlap. In our code, these sets are obtained by flattening each image by using the `flatten()` method, which returns a one-dimensional array. At this point, Equation (3) is used to implement the Dice metric. For the evaluation of the coefficient on the expected masks, the numerator is approximated by the sum of the elements of the matrix obtained by the element-wise product between the elements of the forecast and those of the input mask.The advantage of using the Dice coefficient is that it maintains sensitivity in more heterogeneous data sets and is less sensitive to outliers. In the DSC Formula (3), the parameter *smooth* indicates the smoothing coefficient, which is a value between 0 and 1 that prevents the occurrence of a zero denominator [36,37]. In our model, the value *smooth=1* is set.

Finally, the `dice_coef_loss()` loss function is defined as the opposite of the Dice coefficient.

4. Performance Evaluation

The used processing environment is the well known Google Colab (https://colab.research.google.com/, accessed on 1 November 2022). This tool allows easy repeatability of experiments and code sharing. The free version of Google Colab was used. It offers remote execution on virtual machines (VMs) with limited lifetime (12 h) [38]. The available computing resources for each VM are 12 GB of RAM, 78 GB of disk space and GPU-accelerated computing on Nvidia Tesla K80 [39]. The Tesla K80 GPU combines two graphics processors to increase performance. It is characterized by 2496 shading units, 208 texture mapping units and 48 render output units per GPU.

The comparison done in this work focused on three different organs, with very different features: liver, bone and portal vein. Each patient of the considered dataset contains segmentation images for each of these three organs.

For each organ, a target U-Net was trained and tested. From the original set of 20 patients, 3 patients were discarded due to anomalies in the images that could negatively influence the training process. Figure 5 shows two examples of altered CT-Scans from *3Dircadb1*. In the figure, it is possible to see the presence of an additional circular area that encloses the body section. This area alters the sample's HU values, blurring the difference among the body section and the external area. Hence, these images can increase the probability of unreliable segmentation in the model.

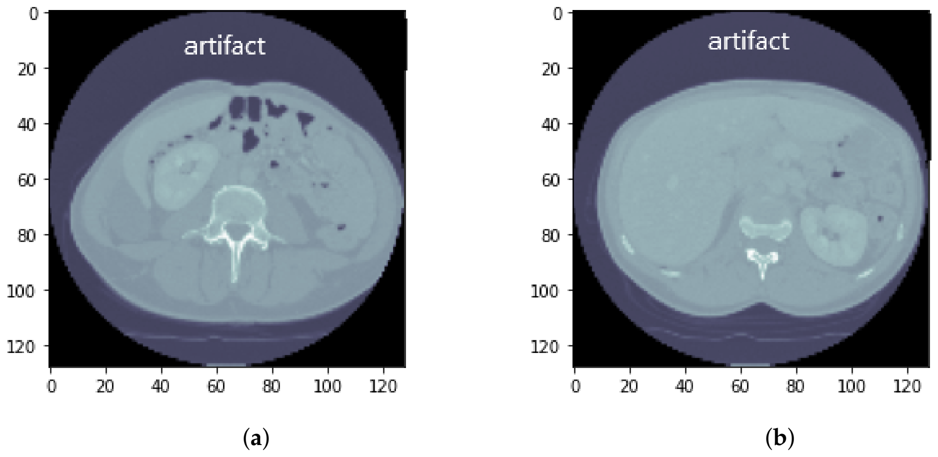

Figure 5. Two different samples of altered CT-Scans from the *3Dircadb1* dataset: presence of an additional grey circular area that encloses the body section, marked by the word 'artifact' to better highlight it.

During the data processing phase presented in Section 3, the CT-Scan images were resized to 128 × 128 pixels. This choice was motivated by the availability of limited computing resources in the Colab VM. In fact, the training of the considered U-Net with images of the original sizes (512 × 512 pixels) or on a 256 × 256 resized version is not feasible in Google Colab, leading to memory error even when considering strongly limited batches (5 or 10 images for a batch). Hence, our training phase consisted of 90 iterations for each organ on 80% of the dataset, processing 128 × 128 CT-Scan images in batches of 32 images. The CT-Scan images are sent as input to the network after a shuffling step. This procedure increases the robustness of the network to both image variability and overfitting. Processing of randomly sorted images is common in Image Segmentation tasks [20,24,40].

As mentioned above, the U-Net was implemented by Keras and Tensorflow, with the following configuration, which is referred to as 'base configuration' in this paper:

- Optimizer: Adam [30], learning rate 1×10^{-3}.
- Performance evaluation metric: Dice coefficient [6].

The first comparison was carried out with the same configuration for the three organs. The U-Net with the configuration described in the previous section provides the results reported in Table 1, tested on the remaining 20% of the samples, consisting of 489 images. The results are expressed in terms of DSC and Accuracy [7], defined as in (4). Nevertheless, due to the nature of the problem, the simple Accuracy metric provides unreliable results.

$$Accuracy = \frac{|X \cap Y| + |\overline{X} \cap \overline{Y}|}{|X| + |\overline{X}|}. \tag{4}$$

In fact, while the DSC decreases with the increasing sparsity of the binary masks, as expected, the corresponding increasing imbalance between 0 s and 1 s in the masks leads to a very high Accuracy. Hence, this high Accuracy does not reflect the actual quality of the segmentation. In fact this quality significantly decreases for small image segments, such as the bone and, in particular, the portal vein, as shown in the following analysis. For this reason, DSC will hereafter be considered as the evaluation metric of the segmentation. Samples for organ images, original masks and masks predicted by the U-Net are reported in Figures 6–9, for liver, bones and two vein samples, respectively.

Table 1. Performance comparison results on the test set.

Organ	Samples	DSC	Accuracy
Liver	489	97.85%	88.65%
Bone	489	81.35%	95.93%
Portal Vein	489	58.53%	98.15%

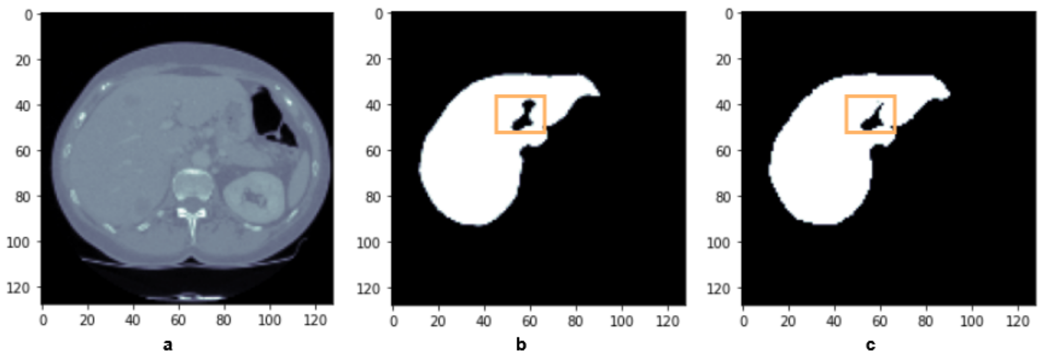

Figure 6. Sample image (**a**), original mask (**b**) and predicted mask (**c**) for the liver.

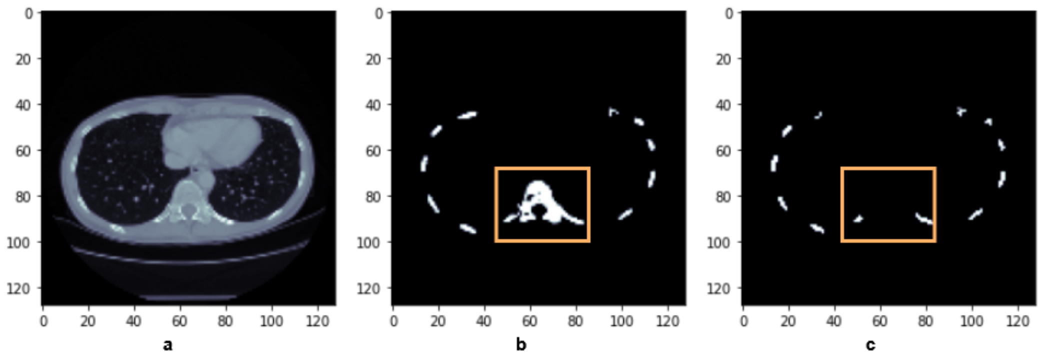

Figure 7. Sample image (**a**), original mask (**b**) and predicted mask (**c**) for the bone.

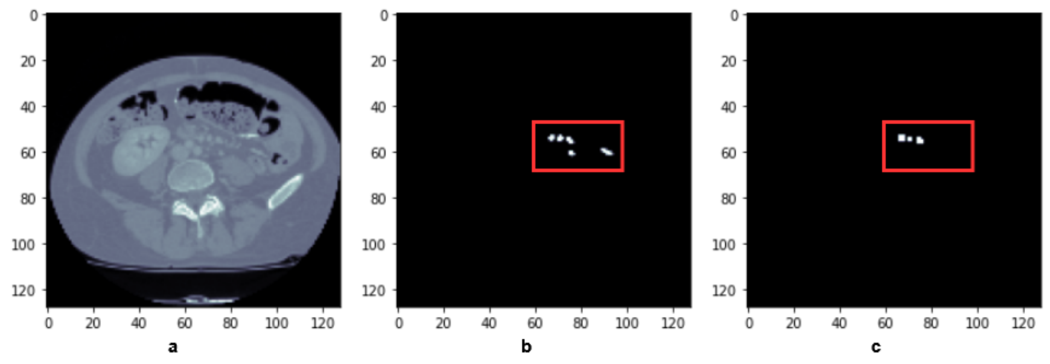

Figure 8. Portal vein sample 1: (**a**) image, (**b**) original mask, and (**c**) predicted mask without zooming in the pre-processing.

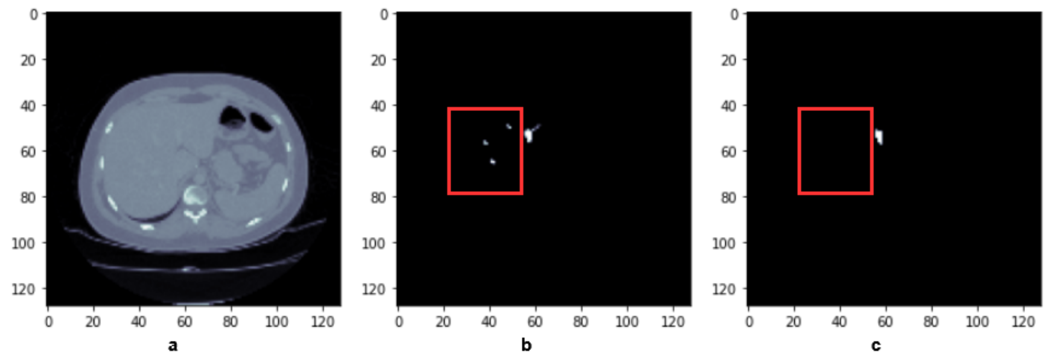

Figure 9. Portal vein sample 2: (**a**) image, (**b**) original mask and (**c**) predicted mask without zooming in the pre-processing.

As shown in Figure 6, the network produces good results (97.85%) for the liver on the test set. The overall parenchyma tissue is well defined in the predicted mask (Figure 6c), with a good precision in the segment border. The internal sections still suffer from some inaccuracy, as shown by the orange boxes in the figure. However, these results are fairly good, considering the size of the training dataset available.

The results for bone segmentation indicate a significant decrease in the DSC value, which is equal to 81.35%. This is still borderline acceptable for not very small organs [13,21,22]. This is confirmed by the analysis of the sample images shown in Figure 7. The predicted mask (Figure 7c) shows how the trained model correctly intercepted the location and perimeter of the bone structure. Nevertheless, it can be seen that in cases of increased sparsity of the target signal, the percentage of error for the same model is increasing. This causes a reduction in the segmentation accuracy of more than the 15% in terms of the DSC metric. In particular, some segments in the predicted mask are predicted with a significantly reduced area with respect to the original mask (Figure 7b), while the spinal section is missing. Even if the signal of the spinal section is not so sparse compared to the other bone sections in the original mask, its rather small dimensions lead to an unreliable segmentation.

This phenomenon becomes more evident when the model is tested by using the portal vein samples. In this case, the prediction performance drops significantly (58.53%) and becomes not acceptable at all. This can be attributed to the increased and significant sparsity of the signal in the portal vein binary masks, due to the specific features of the vasculature of the torso. Both Figures 8 and 9 depict examples of prediction outputs. As can be observed in Figure 8, the trained model cannot detect the vascularization sparse spots in the segmentation mask (Figure 8b) correctly, as it can only reproduce in the output mask the biggest segment detected from the original sample (Figure 8c). This behaviour is further confirmed in Figure 9. In the presence of two different segment regions, one with a bigger area, the other with sparse vascularization spots, the model correctly detects the first one, while it ignores most of the segments in the latter, as shown in the red box (Figure 9c).

A known approach to solve class imbalance in machine learning is the usage of a weighted cross entropy loss. The introduction of weights to penalize the misclassification of minority classes is present in medical image analysis and segmentation as well, in particular for medical Image Segmentation [41]. Hence, the application of a weighted cross entropy loss in our model was evaluated to improve the performance of portal vein images segmentation. The class weights have been configured with the *balanced* option of the Python module `compute_class_weight` (https://scikit-learn.org/stable/modules/generated/sklearn.utils.class_weight.compute_class_weight.html, accessed on 1 November 2022). Nevertheless, considering the same number of training epochs, the introduction of the weighted cross entropy as loss metric significantly degrades the segmentation performance on portal vein images. The evaluation on the test set provides a DSC value equal to 27.8%, as reported in Table 2.

Table 2. DSC metric for different percentages of cropping of the original images (portal vein).

Approach	DSC
Base configuration (no zoom)	58.53%
40% image cropping	62.04%
50% image cropping	65.23%
60% image cropping	76.39%
70% image cropping	81.45%
Weighted cross entropy loss	27.8%

Hence, in order to improve the DSC metric for the portal vein segmentation and, in general, for vascularization samples, the following solution is proposed. A preprocessing step, consisting of magnifying the portion of interest of the binary masks and of the corresponding areas in slice images, is introduced as follows:

- Zooming the image and the corresponding mask in the target area of vascularization, in order to increase the size of the vascularization segments. The zoom was configured to enhance the body section of interest, by cropping with the Tensorflow function `tf.image.central_crop()` up to 70% of the original image. Hence, in this case, the resulting image is reduced to a size of 154 $\times$ 154 pixels from the original size of 512 $\times$ 512 pixels. This resulted in the optimal percentage for the best DSC metric, found experimentally, as shown in Table 2.
- Resizing the samples to the 128 $\times$ 128 format, as in the previous experiments.

The rationale of this strategy is that the higher resolution of original images is leveraged to provide the U-Net with additional information, in order to enlarge the target area. In fact, with the original setting, the amount of the target area in the mask is too small to drive the Dice metric towards a correct recognition of it. This is due to the metric definition itself, which is based on the percentage amount of pixels of the target image (see (3)). This causes, as mentioned in the comments to Figures 8 and 9, the missing recognition of small, sparse spots of the vascularization. Although the resolution in the original setting cannot be handled by the available computing resources, it offers the opportunity to gain additional information via zooming the interested portion of the image before resolution rescaling. The advantages are multiple: not only is the information overlooked in base configuration (i.e., the one without any zoom) leveraged, but the training time and the amount of necessary computing resources are kept unmodified.

After pre-processing, a similar U-Net was trained by using the new images for 90 iterations. The new model leads to an increase in the DSC metric up to 22% for portal veins, corresponding to 81.45% on the test set with the largest and optimal zoom level. Figures 10 and 11 depict two sample outputs obtained by using the new model, specially tuned to segment vascularization in medical CT scans. The positive impact of preprocessing can also be observed on the predicted mask quality. Even in the presence of sparse signals, the model is good at detecting all the present segments, although some minor discrepancies in the vascularization segment shapes (orange box) are present (Figures 10c and 11c). Nevertheless, through the proposed approach, the model is able to fetch significantly small segments that were previously ignored, as shown in another sample in Figure 12 on the green box on the right side of the binary mask (Figure 12c). This is further confirmed by the sample in Figure 13, replicating the same two distinct segments highlighted in the green box (Figure 13c).

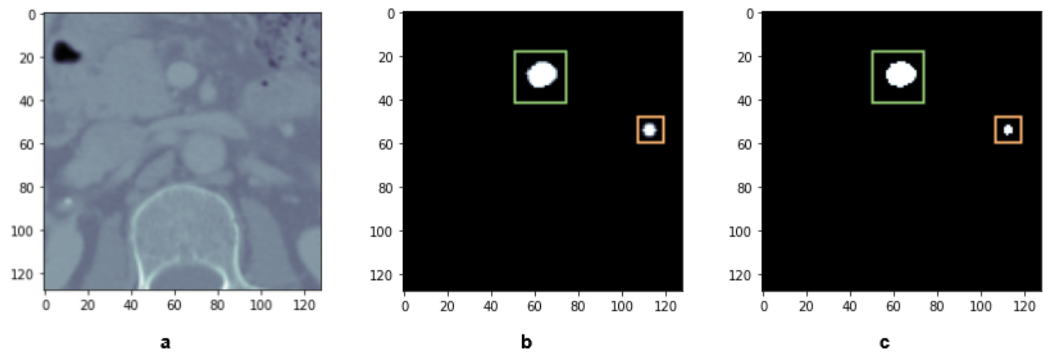

Figure 10. Portal vein sample 3: image (**a**), original mask (**b**) and predicted mask (**c**) after pre-processing with zoom.

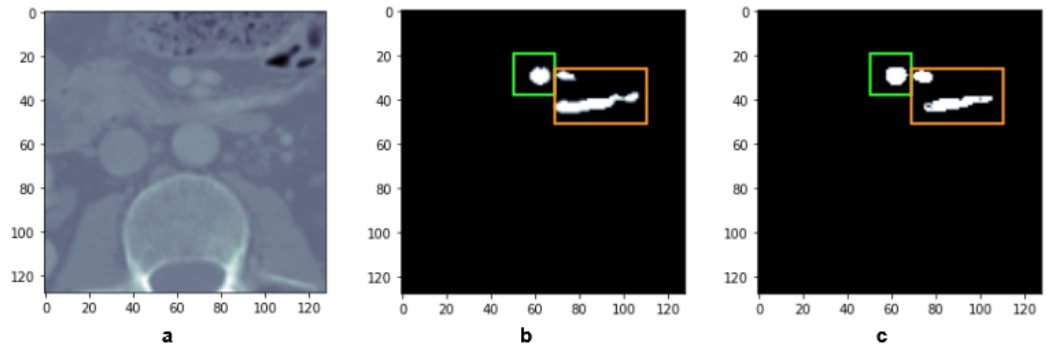

Figure 11. Portal vein sample 4: image (**a**), original mask (**b**) and predicted mask (**c**) after pre-processing with zoom.

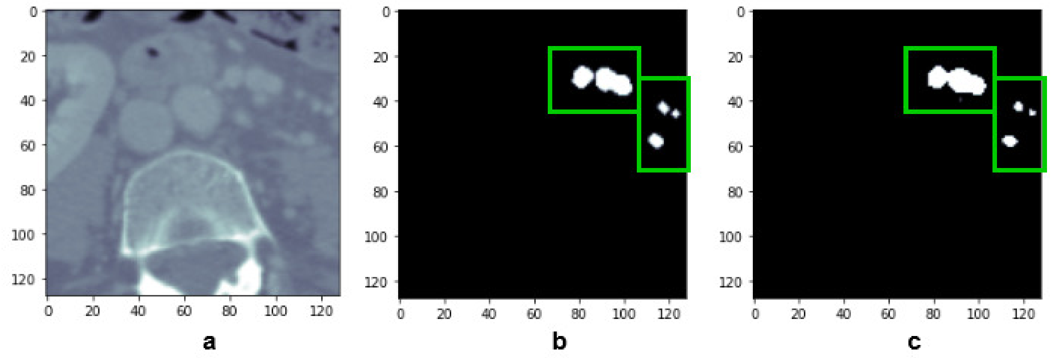

Figure 12. Portal vein sample 5: image (**a**), original mask (**b**) and predicted mask (**c**) after pre-processing with zoom.

Although results are appreciable, some margin for further improvement exists. In particular, some research is still necessary to refine the model in order to identify even the smallest segments, such as the ones missing in the prediction output of the sample shown in Figure 13c. This cannot be done by further zooming the available images, thus different techniques have to be designed in the future, or images with higher resolution are necessary.

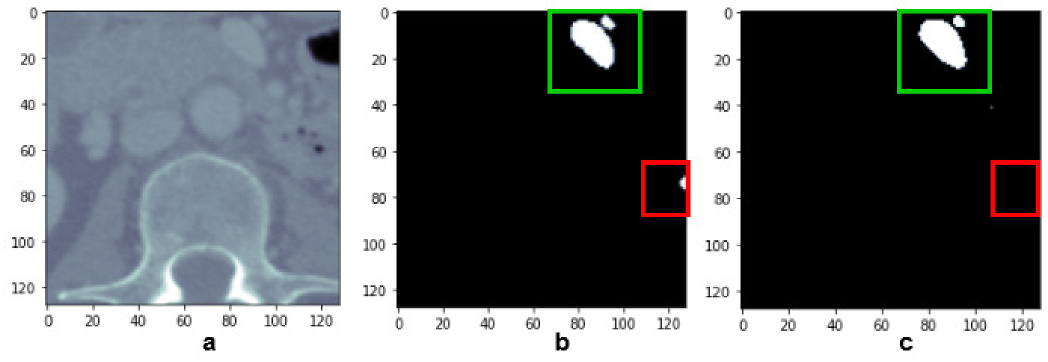

Figure 13. Portal vein sample 6: image (**a**), original mask (**b**) and predicted mask (**c**) after pre-processing with zoom.

The optimal zoom level was found by testing different configurations, as shown in Table 2. Figure 14 shows the trend of the DSC metric as a function of the zoom level. The saturation effect is evident for values higher than a zoom level of 3×. This effect can be explained by the fact that the maximum lossless zoom that can be performed on these images is 4×, since the original resolution of the images is 512 × 512 and the target one used to feed the U-Net is 128 × 128. Beyond this zoom value, the cropped image will have a size inferior to 128 × 128; thus, further improvements for larger magnification levels are not expected. In this case, it is not possible to use a zoom value of 4×, since this would also crop part of the white portion of the mask. In addition, the figure shows that the proposed zoom procedure does not impact the Accuracy metric, which remains stable in the range 98–99%, as expected. In fact, our procedure does not involve any loss of useful data for the mask of the considered organ; thus, the Accuracy is not affected by it.

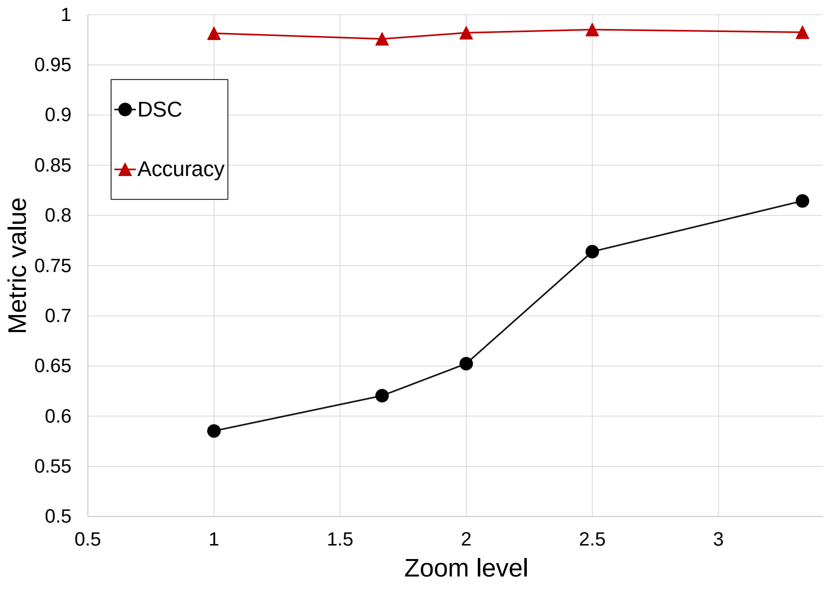

Figure 14. DSC and Accuracy metrics as a function of the zoom level.

Another evolutionary step is in the automatization of the whole process. In the current approach, the area of interest must be first identified by the medical staff. Then, the automatic zoom on the selected area is performed on the CT Scans and the corresponding masks, obtaining images in a 128 × 128 format. Although most of the operations are already automatic, it would be possible to leverage the first phase in order to identify the area deserving more attention and using the first output to automatically focus the zoom operation on the right portion of the image. This will be an objective of future investigations. In this regard, Figure 15 presents a general overview of the segmentation procedure detailed in this section. After the preliminary image preprocessing, consisting in the rescaling to a resolution of 128 × 128, the U-Net is trained on a portion of the dataset. The performance is then evaluated on the remainder of the samples. In the presence of small organs, as shown in what follows, a significant DSC decrease can be observed. Hence, if this decrease produces DSC values lower than a predefined threshold, corresponding to an 80% decrease in this paper, samples are further processed with an incremental zoom of images. This procedure stops in cases of data loss, i.e., if part of the organ signal is cut away by the zoom, or when the U-Net is able to achieve the desired performance score in terms of DSC.

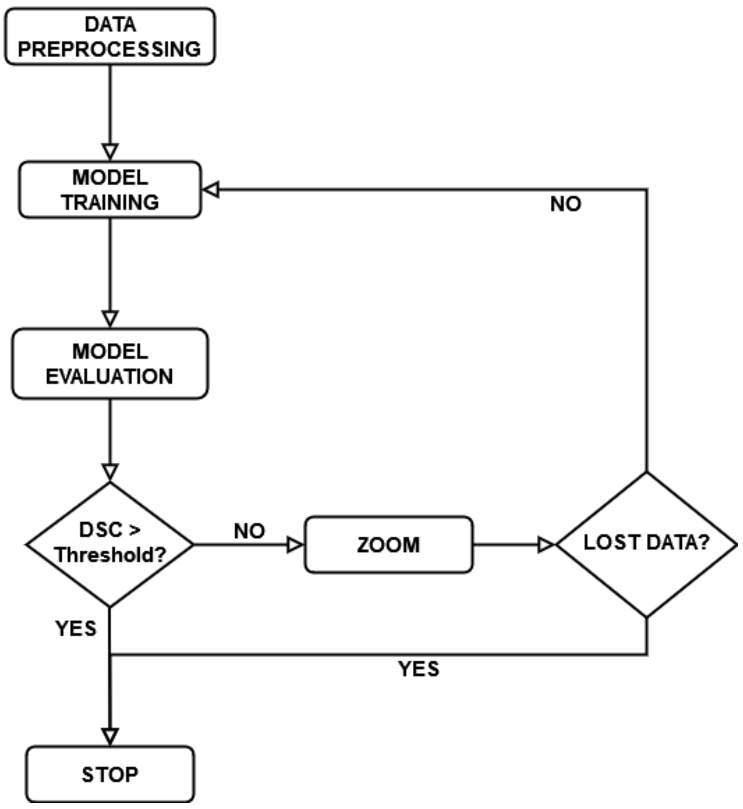

Figure 15. Flow diagram of the training of the mixed size medical segmentation procedure via U-Net.

Small organ segmentation has been previously investigated in other papers, such as [34,35]. FocusNetv2, presented in [34], was developed to segment small organs in the head and neck (HaN) area. This network is composed of various parts. They include a main segmentation network based on Snet, a U-Net variant, a small-organ localization network, to localize the center locations of small organs and a small-organ segmentation network, which refines the Snet's segmentation with the information provided by the small-organ localization network. The proposed ensemble is tested on various organs of different size, providing a maximum DSC of 82.45% on small organs. Hence, the performance obtained with our approach is comparable with the DSC obtained with more complex and computationally expensive models, at a reduced cost. A modular network is used in [35] as well, to segment pancreas images, a small organ, from the NIH dataset. The proposed network operates with two levels, a coarse-segmentation stage and a fine-segmentation stage, with the introduction of a saliency transformation module. This module converts the previous iteration's segmentation probability map into spatial weights for the current iteration. The proposed model produces a DSC of 84.5% on the pancreas NIH dataset. Hence, the performance obtained with the approach presented in this paper is comparable with the state-of-the-art, without requiring additional complexity in the network, increasing neither the computational cost nor the needed volume of training data.

In summary, with the proposed approach, the segmentation of binary masks with sparse signals, such as in the case of vascularization or small organ scans, is improved. In particular, an increased accuracy evaluated through the DSC metric is of great importance in the medical field, especially for vascularization images. In fact, the analysis of these images is fundamental to accelerating the diagnosis of various cancer types, due to the tumor angiogenesis process [5]. It is worth stressing that the achieved performance is reached with the base version of Google Colab, without the need for additional, paid resources. This is an important result, since it demonstrates that testing with U-Net can be carried out easily in a standard developing environment available to everyone, without requiring one to set up a dedicated cluster with multi-core CPUs or GPUs. Thus, the entry level of this approach is represented only by the skill of the programmers, without infrastructure costs. The price to pay is a small variability in execution times, due to the fact that resources in Google Colab are not reserved exclusively for free users and the fact that activity cannot last more than 12 consecutive hours. However, this is definitely acceptable for a developing work. Obviously, it is possible to set up a local cluster offering the Jupyter Notebook service or pay for dedicated cloud resources in order to not experience the limitations of the free account, but this is more suitable in operation, when service discontinuity is not acceptable. In any case, the software to run is exactly the same developed on the free version of Google Colab.

Finally, it is interesting to evaluate the time taken by the overall procedure to complete the test phase, i.e., in operation. The results of the evaluation campaign are reported in Table 3. The evaluation included four cases, each one repeated several times to evaluate both mean time and standard deviation, as follows: (i) single image; (ii) set of images of a single patient, with 129 slices; (iii) set of images of a single patient, with 172 slices; (iv) set of images of a single patient, with 200 slices. The usage of the dataset of different patients, each with a different number of slices, allows evaluating the scalability of the procedure. The used images are already loaded in Google Colab, to avoid including the effects of the upload time in the evaluation, since it depends on external factors, such as speed and reliability of network connection. It results that the segmentation of a single image taken from a batch of the test set by Tensorflow requires 3.286 s on average. In addition, in cases in which the zoom of images is not necessary, the initial pre-processing (rescaling operation) requires a mean time of 21 ms. In cases in which the zooming procedure is completed, it takes a slightly higher mean time, equal to 23 ms. Both of them are negligible with respect to the processing time. When the processing is executed on a larger set of images, processed in batches of 10 images, both processing and pre-processing times scale very well. In fact, they exhibit a sub-linear behaviour, with a processing time for 200 images well below 20 s and pre-processing always below 100 ms, with an impact of zooming operation that is essentially negligible. This phenomenon can be explained by the high parallelism that can be achieved with Tensorflow when batches of data are processed, that allows to fully exploit the capabilities of multi-core CPUs and GPUs. A second comment is that, in general, the variability of the processing time, evaluated by means of the standard deviation, is always quite limited, which is highly desirable. When execution times are very small, such as in the case of pre-processing, results could be slightly different from expected, due to free resource allocation in Google Colab. In short, the time taken to execute the overall procedure is negligible with respect to the time taken by a human operator to complete the same task, which makes the proposed approach definitely affordable.

Table 3. Processing and pre-processing times for different sets of images.

Number of Images	Processing Time (Avg ± Std)	Pre-Processing Time (Avg ± Std)	
		without Zoom	with Zoom
1	3.286 s ± 0.046 s	0.021 s ± 0.001 s	0.023 s ± 0.003 s
129	11.669 s ± 0.854 s	0.028 s ± 0.004 s	0.051 s ± 0.011 s
172	14.918 s ± 1.045 s	0.027 s ± 0.005 s	0.081 s ± 0.073 s
200	17.189 s ± 0.132 s	0.055 s ± 0.041 s	0.095 s ± 0.017 s

5. Conclusions

In this study, the performance of a state-of-the-art segmentation model, based on U-Net [4] is first analyzed by using CT-Scans of different organs, namely liver, bones and portal vein. Our results show that the standard U-Net network can provide very good results for segmenting large organs, approaching 98% Accuracy values in terms of DSC metric, but it exhibits poor performance when organ images characterized by small and sparse segmentation training masks are used. In particular, this happened in the case of vascularization images. In fact, segmentation of small organs is a recurring and challenging issue for automated medical Image Segmentation [22,34,35]. In order to overcome this problem, a novel approach producing a significant improvement of the DSC metric in the most critical cases is proposed. This approach does not require the use of additional data samples, nor a significant additional computational burden. In fact, for all our analyses, the free computing infrastructure made available by Google Colab was enough.

In particular, the suitable working conditions of the baseline U-Net are determined for segmenting sparse and/or small sections through image pre-processing. Since a quite general approach and metric (the DSC) to drive the segmentation is used, the solution can be used in other situations based on similar images and metrics. In the most critical case, relevant to portal vein images, the DSC improvement obtained is 20%. This comes basically at a very small cost, leaving untouched the training time and computing requirements in comparison with the baseline U-Net processing.

Our approach is based on training the U-Net by using differentiated zoom levels in different areas of test images. Thus, it follows that resorting to higher resolution images can bring further significant benefits in terms of segmentation accuracy. Clearly, an increase in resolution would require additional storage space and computing power. However, determination of the achievable performance improvement, if anything, needs further research. In fact, the observed improvement is due to the usage of the Dice metric and the impact of additional resolution, and a higher scaling factor on images cannot be easily determined; a deep investigation is necessary.

Due to its flexibility and small computational cost, the application of differentiated zoom levels, associated with the Dice metric, used to enhance the segmentation quality of small entities, can potentially be applied to other use-cases characterized by sparse segmentation signals. Nevertheless, the application of the proposed approach to other use-cases is beyond the scope of this paper. Finally, the proposed approach can have a significant impact in operation, since a correct segmentation of a vasculature or any other small organs and disease areas is essential for helping medical diagnosis.

Author Contributions: Conceptualization, methodology and validation, G.R.; software, G.R. and P.B.; writing—original draft preparation, P.B. and M.F.; writing—review and editing, M.F. and G.R.; supervision, G.R.; funding acquisition, G.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the company Molecular Horizon srl to the Department of Engineering of the University of Perugia.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: This reasearch has been conducted on the publicy availabe Liver segmentation 3D-IRCADb-01 from IRCAD institute, available at https://www.ircad.fr/research/data-sets/liver-segmentation-3d-ircadb-01/ (accessed on 1 November 2022).

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

CNN	Convolutional Neural Network
CT	Computed Tomography
HU	Hounsfield Unit
MRI	Magnetic Resonance Imaging
DICOM	Digital Imaging and Communications in Medicine
VTK	Visualization Toolkit
GPU	Graphics Processing Unit
VM	Virtual Machine

Appendix A

Listing A1. Loading CT-Scan into the related tensors starting from the file paths.

```
def process_path(filename):
    patient_bytes = tf.io.read_file(filename)
    patient_image = tfio.
                    image.
                    decode_dicom_image(patient_bytes,
                                       color_dim=False,
                                       on_error='skip',
                                       scale='preserve',
                                       dtype=tf.uint16,
                                       name=None)
    tf.cast(patient_image, tf.int32, name=None)
    patient_image = tf.image.resize(patient_image, (D,D))
    patient_image = tf.squeeze(patient_image, axis=0)
    mask_path = tf.strings.regex_replace(filename,
                                         'PATIENT_DICOM',
                                         'MASKS_DICOM/liver')
    mask_bytes = tf.io.read_file(mask_path)
    mask_image = tfio.image.decode_dicom_image(mask_bytes,
                                               scale='auto',
                                               on_error='lossy',
                                               dtype=tf.uint8)
    mask_image = tf.squeeze(mask_image, axis=0)
    intercept_tag = tfio.
                    image.
                    decode_dicom_data(patient_bytes,
                                      tags=np.uint32(int("00281052",
                                                     16)))
    intercept = tf.strings.to_number(intercept_tag, tf.float32)
    slope_tag = tfio.
                image.
                decode_dicom_data(patient_bytes,
                                  tags=np.uint32(int("00281053",
                                                 16)))
    slope = tf.strings.to_number(slope_tag, tf.float32)
    patient_image = tf.
                    math.
                    add(tf.math.multiply(patient_image, slope),
                        intercept) /100.-10.
    return patient_image, mask_image
```

References

1. Kapoor, L.; Thakur, S. A survey on brain tumor detection using image processing techniques. In Proceedings of the 2017 7th International Conference on Cloud Computing, Data Science & Engineering—Confluence, Noida, India, 12–13 January 2017; pp. 582–585. [CrossRef]
2. Jaume, S.; Ferrant, M.; Macq, B.; Hoyte, L.; Fielding, J.; Schreyer, A.; Kikinis, R.; Warfield, S. Tumor detection in the bladder wall with a measurement of abnormal thickness in CT scans. *IEEE Trans. Biomed. Eng.* **2003**, *50*, 383–390. [CrossRef] [PubMed]
3. Fu, K.; Mui, J. A survey on Image Segmentation. *Pattern Recognit.* **1981**, *13*, 3–16. [CrossRef]
4. Ronneberger, O.; Fischer, P.; Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. *arXiv* **2015**, arXiv:1505.04597[cs].
5. Ehling, J.; Theek, B.; Gremse, F.; Baetke, S.; Möckel, D.; Maynard, J.; Ricketts, S.A.; Grüll, H.; Neeman, M.; Knuechel, R.; et al. Micro-CT imaging of tumor angiogenesis: Quantitative measures describing micromorphology and vascularization. *Am. J. Pathol.* **2014**, *184*, 431–441. [CrossRef]
6. Carass, A.; Roy, S.; Gherman, A.; Reinhold, J.; Jesson, A.; Arbel, T.; Maier, O.; Handels, H.; Ghafoorian, M.; Platel, B.; et al. Evaluating White Matter Lesion Segmentations with Refined Sørensen-Dice Analysis. *Sci. Rep.* **2020**, *10*, 8242. [CrossRef]
7. Powers, D.M.W. Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation. *arXiv* **2020**, arXiv:2010.16061.
8. Wang, R.; Lei, T.; Cui, R.; Zhang, B.; Meng, H.; Nandi, A.K. Medical Image Segmentation using deep learning: A survey. *IET Image Process.* **2022**, *16*, 1243–1267. [CrossRef]
9. International Atomic Energy Agency. *Diagnostic Radiology Physics—A Handbook for Teachers and Students*; Non-Serial Publications, IAEA: Wien, Austria, 2014 .
10. Horowitz, S.L.; Pavlidis, T. Picture Segmentation by a Tree Traversal Algorithm. *J. ACM* **1976**, *23*, 368–388. [CrossRef]
11. Zaitoun, N.M.; Aqel, M.J. Survey on Image Segmentation Techniques. *Procedia Comput. Sci.* **2015**, *65*, 797–806.
12. Albawi, S.; Mohammed, T.A.; Al-Zawi, S. Understanding of a convolutional neural network. In Proceedings of the 2017 International Conference on Engineering and Technology (ICET), Antalya, Turkey, 21–23 August 2017; pp. 1–6. [CrossRef]
13. Wang, G.; Li, W.; Zuluaga, M.; Aughwane, R.; Patel, P.; Aertsen, M.; Doel, T.; David, A.; Deprest, J.; Ourselin, S.; et al. Interactive Medical Image Segmentation Using Deep Learning With Image-Specific Fine Tuning. *IEEE Trans. Med Imaging* **2018**, *37*, 1562–1573. [CrossRef]
14. Wang, G.; Zuluaga, M.A.; Li, W.; Pratt, R.; Patel, P.A.; Aertsen, M.; Doel, T.; David, A.L.; Deprest, J.A.; Ourselin, S.; et al. DeepIGeoS: A Deep Interactive Geodesic Framework for Medical Image Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2019**, *41*, 1559–1572. [CrossRef] [PubMed]
15. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. In Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, 7–9 May 2015.
16. Li, X.; Qian, W.; Xu, D.; Liu, C. Image Segmentation Based on Improved Unet. *J. Phys. Conf. Ser.* **2021**, *1815*, 012018. [CrossRef]
17. Lu, H.; She, Y.; Tie, J.; Xu, S. Half-UNet: A Simplified U-Net Architecture for Medical Image Segmentation. *Front. Neuroinform.* **2022**, *16*. [CrossRef] [PubMed]
18. Siddique, N.; Paheding, S.; Elkin, C.P.; Devabhaktuni, V. U-Net and Its Variants for Medical Image Segmentation: A Review of Theory and Applications. *IEEE Access* **2021**, *9*, 82031–82057. [CrossRef]
19. Han, G.; Zhang, M.; Wu, W.; He, M.; Liu, K.; Qin, L.; Liu, X. Improved U-Net based insulator Image Segmentation method based on attention mechanism. *Energy Rep.* **2021**, *7*, 210–217.
20. Saood, A.; Hatem, I. COVID-19 lung CT Image Segmentation using deep learning methods: U-Net versus SegNet. *BMC Med. Imaging* **2021**, *21*, 19. [CrossRef]
21. Gao, Y.; Huang, R.; Chen, M.; Wang, Z.; Deng, J.; Chen, Y.; Yang, Y.; Zhang, J.; Tao, C.; Li, H. FocusNet: Imbalanced Large and Small Organ Segmentation with an End-to-End Deep Neural Network for Head and Neck CT Images. In *Proceedings of the Medical Image Computing and Computer Assisted Intervention—MICCAI 2019: 22nd International Conference, Shenzhen, China, 13–17 October 2019, Proceedings, Part III*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 829–838. [CrossRef]
22. Valindria, V.V.; Lavdas, I.; Cerrolaza, J.J.; Aboagye, E.O.; Rockall, A.G.; Rueckert, D.; Glocker, B. Small Organ Segmentation in Whole-body MRI using a Two-stage FCN and Weighting Schemes. In *Proceedings of the Machine Learning in Medical Imaging: 9th International Workshop, MLMI 2018, Held in Conjunction with MICCAI 2018, Granada, Spain, 16 September 2018*; Springer: Berlin/Heidelberg, Germany, 2018.
23. Wu, W.; Zhang, J.; Xie, H.; Zhao, Y.; Zhang, S.; Gu, L. Automatic detection of coronary artery stenosis by convolutional neural network with temporal constraint. *Comput. Biol. Med.* **2020**, *118*, 103657. [CrossRef]
24. Jun, T.J.; Kweon, J.; Kim, Y.H.; Kim, D. T-Net: Nested encoder–decoder architecture for the main vessel segmentation in coronary angiography. *Neural Netw.* **2020**, *128*, 216–233. [CrossRef]
25. Çiçek, O.; Abdulkadir, A.; Lienkamp, S.S.; Brox, T.; Ronneberger, O. 3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*; Springer: Cham, Switzerland, 2016. [CrossRef]
26. Soler, L.; Hostettler, A.; Agnus, V.; Charnoz, A.; Fasquel, J.; Moreau, J.; Osswald, A.; Bouhadjar, M.; Marescaux, J. 3D image reconstruction for comparison of algorithm database: A patient specific anatomical and medical image database. *Tech. Rep. IRCAD* **2010**. Available online: http://www-sop.inria.fr/geometrica/events/wam/abstract-ircad.pdf (accessed on 28 September 2022).

27. Bishop, C.M. Pattern Recognition and Machine Learning. In *Information Science and Statistics*; Springer: New York, NY, USA, 2006; Volume 4, p. 738. [CrossRef]
28. Keras API. Available online: https://keras.io/api/ (accessed on 28 September 2022).
29. Han, J.; Moraga, C. The influence of the sigmoid function parameters on the speed of backpropagation learning. In *Proceedings of the From Natural to Artificial Neural Computation*; Mira, J., Sandoval, F., Eds.; Springer: Berlin/Heidelberg, Germany, 1995; pp. 195–201.
30. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. In Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, 7–9 May 2015.
31. Ruder, S. An overview of gradient descent optimization algorithms. *arXiv* **2016**, arXiv:1609.04747.
32. Mustapha, A.; Lachgar, M.; Ali, K. Comparative study of optimization techniques in deep learning: Application in the ophthalmology field Comparative study of optimization techniques in deep learning: Application in the ophthalmology field. *J. Phys. Conf. Ser.* **2021**, *1743*, 012002. [CrossRef]
33. Manugunta, R.K.; Maskeliūnas, R.; Damaševičius, R. Deep Learning Based Semantic Image Segmentation Methods for Classification of Web Page Imagery. *Future Int.* **2022**, *14*, 277. [CrossRef]
34. Gao, Y.; Huang, R.; Yang, Y.; Zhang, J.; Shao, K.; Tao, C.; Chen, Y.; Metaxas, D.N.; Li, H.; Chen, M. FocusNetv2: Imbalanced large and small organ segmentation with adversarial shape constraint for head and neck CT images. *Med. Image Anal.* **2021**, *67*, 101831. [CrossRef]
35. Yu, Q.; Xie, L.; Wang, Y.; Zhou, Y.; Fishman, E.K.; Yuille, A.L. Recurrent Saliency Transformation Network: Incorporating Multi-stage Visual Cues for Small Organ Segmentation. In *Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*; IEEE Computer Society: Los Alamitos, CA, USA, 2018; pp. 8280–8289. [CrossRef]
36. Wang, L.; Wang, C.; Sun, Z.; Chen, S. An Improved Dice Loss for Pneumothorax Segmentation by Mining the Information of Negative Areas. *IEEE Access* **2020**, *8*, 167939–167949. [CrossRef]
37. Jadon, S. A survey of loss functions for semantic segmentation. In Proceedings of the 2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB), Via del Mar, Chile, 27–29 October 2020; pp. 1–7. [CrossRef]
38. Google Colab FAQs. Available online: https://research.google.com/colaboratory/faq.html (accessed on 28 September 2022).
39. Tesla K80 | NVIDIA. Available online: https://www.nvidia.com/en-gb/data-center/tesla-k80/ (accessed on 28 September 2022).
40. Weng, Y.; Zhou, T.; Li, Y.; Qiu, X. NAS-Unet: Neural Architecture Search for Medical Image Segmentation. *IEEE Access* **2019**, *7*, 44247–44257. [CrossRef]
41. Ben naceur, M.; Akil, M.; Saouli, R.; Kachouri, R. Fully automatic brain tumor segmentation with deep learning-based selective attention using overlapping patches and multi-class weighted cross-entropy. *Med. Image Anal.* **2020**, *63*, 101692. [CrossRef]