UCLouvain

epl

École polytechnique de Louvain

Federated Learning for organ segmentation

Author: **Thibaud Misonne**
Supervisor: **Sébastien Jodogne**
Readers: **Sébastien Jodogne, Christophe De Vleeschouwer, Jean Léger**
Academic year 2021–2022
Master [120] in Computer Science and Engineering

Abstract

Training deep learning models for medical imaging requires access to large volumes of sensitive patient data. To this end, the models are generally trained on centralised, deidentified databases that are hard to collect because of privacy requirements.

Federated Learning proposes an alternative approach, in which a coalition of hospitals collaboratively trains a central model without exchanging any clinical data. This thesis explores the combination of Federated Learning with U-Net models, and applies it to the task of image segmentation of the heart, the lungs and the oesophagus. The experiments are made locally, using three distinct datasets to simulate different hospitals, with the inter-variability that it involves.

A variant of Federated Learning referred to as Federated Equal-Chances that improves segmentation performance on unbalanced datasets is introduced as well. The necessary steps to implement it in a real-life scenario are detailed and the potential of Federated Learning to improve medical image segmentation is proven.

Acknowledgements

First and foremost, I would like to thank my supervisor Sébastien Jodogne for his support during the thesis. I am particularly thankful for all the time he devoted to me, for his motivation and his curiosity regarding my results. I also highly appreciated that he made researches to give me access to a cluster of GPU to process the training of my models. Finally, I am grateful for the opportunity to highlight my work in a conference, the IVMSP 2022 where my work will be presented.

Next, I would like to thank Jean Léger and Eliott Brion for their help regarding U-Net and deep learning for organ segmentation in general. Their advice was precious to give me confidence about my work and to put me in the right direction.

I would also like to thank Viviane Pierrard and Alexis Bedoret who carefully revised my final work and gave me a precious feedback about it. Their help was precious to guarantee a well-written and clear report, and to avoid English mistakes.

Finally, I would like to thank my family and friends for their constant love and support.

Acronyms

Acronym	**Extended form**
CT	Computed Tomography
DSC	Dice Similarity Coefficient
DICOM	Digital Imaging and COmmunications in Medicine
fedAvg	Federated Averaging
fedEq	Federated Equal-Chances
HU	Hounsfield Unit
iid	independent and identically distributed
LCTSC	Lung CT Segmentation Challenge
NSCLC	Non-Small Cell Lung Cancer
PACS	Picture Archiving and Communication System
PReLU	Parametric Rectified Linear Unit
ReLU	Rectified Linear Unit
ROI	Region Of Interest
SGD	Stochastic Gradient Descent
Slurm	Simple linux utility for resource management
TCIA	The Cancer Imaging Archive
UID	Unique IDentifier

Contents

1 Introduction **1**

2 Medical image segmentation **3**
2.1 Anatomy of a CT scan 3
2.2 Standardisation of the pixel array 4
2.3 Anatomy of a RT-STRUCT 6
2.4 From contours to masks 7

3 Deep learning for image segmentation **9**
3.1 Sliding-window approach 9
3.1.1 Dataset generation 9
3.1.2 Segmentation 10
3.1.3 Drawback of that approach 11
3.2 U-Net 12
3.2.1 Dataset generation 13
3.2.2 Detailed architecture 14
3.2.3 Loss functions 15
3.2.4 Preventing overfitting 18
3.2.5 Optimisation of the hyperparameters 19
3.2.6 Recent evolutions of U-Net 20
3.3 Evaluation metric 20
3.4 Implementation 21
3.4.1 Defining the model 21
3.4.2 Loss functions 23
3.4.3 Data augmentation 23
3.4.4 Main function 24
3.4.5 Evaluation function 25

4 Federated Learning **26**
4.1 State of the art 26
4.1.1 Federated Averaging 27
4.1.2 Distribution of the data between hospitals . 28
4.2 Local implementation of Federated Averaging . . . 29
4.3 Dealing with inter-variability of the data 30
4.4 Federated Equal-Chances 31
4.4.1 From the local simulation to a real framework 32
4.5 Implementation 36

5 Results **39**
5.1 Datasets 39
5.1.1 NSCLC-Radiomics 39
5.1.2 LCTSC 40
5.1.3 Pediatric CT-SEG 40
5.2 CISM and CÉCI cluster 40
5.3 Evaluation procedure 41
5.4 Two steps training 41
5.5 Results for the heart 42
5.5.1 Datasets distribution 42
5.5.2 Comparison of the strategies 43
5.5.3 Visualisation of the results 45
5.5.4 Robustness of the strategy 47
5.6 Results for the lungs 48
5.6.1 Datasets distribution 48
5.6.2 Comparison of the strategies 49
5.6.3 Visualisation of the results 50
5.7 Results for the oesophagus 51
5.7.1 Datasets distribution 51
5.7.2 Comparison of the strategies 52
5.7.3 Visualisation of the results 53
5.8 Computation times 54

6 Conclusion **55**
6.1 Main results 55
6.2 Future work 56
6.2.1 Local updates reception 57
6.2.2 Expensive communication 57
6.2.3 Security 58

A Conference paper **i**

B Recent evolutions of U-Net **vii**

B.0.1 3D U-net vii

B.0.2 V-Net viii

B.0.3 MultiResUNet x

C Submission script on the cluster **xii**

Chapter 1

Introduction

Today, cancers are often treated by radiotherapy, which makes heavy use of medical imaging and computers [1]. To do so, precise localisation of the organs is needed. Indeed, organs at risk must be preserved in order to avoid damaging side effects. Such a localisation can be manually defined by doctors, but it has two main drawbacks: These manual segmentations are costly and time consuming for the them.

An alternative solution to perform organ segmentation is to train a deep learning model to automatically recognise and localise the organs. If working efficiently, this solution can be faster than doctors and save them a consequent amount of time [2]. Radiotherapy offers a great opportunity to study such segmentation algorithms in the context of human radiology. Indeed, the oncologists working in any radiotherapy department all around the world have to delineate organs-at-risk and tumors so as to propose efficient, highly personalised treatments to their patients. This results in radiotherapy departments gathering large volumes of labeled images that could be extremely useful to train artificial intelligence algorithms.

However, such labeled medical images are not easily accessible to researchers, because they correspond to clinical data that must be carefully protected, and that should not be communicated outside of the hospitals without patient's consent. As a consequence, even though free and open-source software are widely available to extract DICOM images from the Picture Archiving and Communication System (PACS) of the hospital, to de-identify them, then to send them to external artificial intelligence experts [3], such clinical trials necessitate the setup of complex research protocols because they typically associate two distinct entities (one hospital and one uni-

versity). Some large datasets have nevertheless been released as open data, notably thanks to the efforts of The Cancer Imaging Archive project [4], but this is a small fraction of all the data that is produced by hospitals on a daily basis.

An alternative approach would be to train deep learning models directly inside the hospitals, without having to communicate patient data to the outside world. In practice however, training a complex segmentation model requires a lot of annotated data, and the amount of data available inside one single hospital may not be sufficient to effectively train the model. Furthermore, few hospitals employ data scientists who are granted to access patient data in order to do in-house artificial intelligence research, which limits the applicability of this approach to a handful of university hospitals.

Federated Learning might be a game changer to train deep learning models for healthcare [5]. In the typical Federated Learning setup, many mobile devices collaborate to train a shared model in a decentralised way, without ever communicating personal data to protect privacy [6]. Even though it was not originally designed for medical data, Federated Learning can be a very interesting solution as this would allow to respect privacy by design, avoiding the need to centralise the data to learn the model.

The goal of this thesis is to evaluate the potential of Federated Learning applied to organ segmentation by comparing the predictions done using local data, centralised data and Federated Learning. To do so, three organs are segmented using these strategies based on Computed Tomography (CT) scans: The heart, the lungs and the oesophagus. A new Federated Learning algorithm called Federated Equal-Chances is also introduced, inspired from the most common approach but with the intent of improving the shortcomings of this strategy.

All source code used for this thesis is available using the following link: `https://github.com/yoyomanhihi/Master_thesis`.

Chapter 2

Medical image segmentation

Medical segmentation is the process of contouring a region of interest on a medical image. This task comes with a number of challenges, from a file format to store the medical related information to an evaluation metric to assess the quality of a segmentation. This chapter aims to explain the important aspects of medical image segmentation and to introduce the DICOM file format and the manipulations that can be made on it.

2.1 Anatomy of a CT scan

Digital Imaging and Communications in Medicine (DICOM) is the de facto, open standard for medical imaging. It defines a file format as well as a protocol for transmission. It is used for the exchange, storage and network communication of medical images and images related information [7].

In a nutshell, a medical CT scan is stored as a series of DICOM files (typically a few hundreds), each one storing one slice of the patient's scan, which can be used to recreate the full 3D scan. In addition to the image of the scan itself, each file stores images related information, which is called the metadata. The metadata stores a lot of information, including data about the patient and about how to interpret the scans. The first important sections that are stored are unique identifiers (UID) of the file. UIDs are respectively stored for the patient, the study, the series and the instance. The series's UID allows to recognise that two files come from the same 3D scan, and the instance's UID is a unique identifier per slice.

A simplified DICOM file can be found in Table 2.1. The Tag

Tag	Name	Value
(0010, 0020)	Patient ID	LUNG1-001
(0020, 000d)	Study Instance UID	1.3.6.1.4.1.32722.99.99.239...
(0020, 000e)	Series Instance UID	1.3.6.1.4.1.32722.99.99.298...
(0020, 0013)	Instance Number	56
(7fe0, 0010)	Pixel Data	(512 x 512) array

Table 2.1: Example of a simplified DICOM file with anonymised data

field is a unique tag that is common to all DICOM files, allowing to easily retrieve the related information. For example, the tag (0010, 0020) corresponds to the Patient ID in every DICOM file, independently of the scan or the study. The most important field that is needed is the Pixel Data, as it represents the slice of the scan itself. However, this pixel array does not represent a standardised value from one study to another. Some pre-processing can however be applied thanks to the metadata to standardise their value.

2.2 Standardisation of the pixel array

To understand why this array is not standardised by default, it is first needed to understand how a CT scan works. In a nutshell, the patient rests on a table in the centre of a circular scanner which is about 2.4 meters tall. The scanner is composed of an X-ray source emitting beams and a detector facing it, measuring the density. An illustration of the scanner is shown on figure 2.1. To make the scan, the source and detector rotate around the patient. The beams from the source pass through the patient and are analysed from many angles by the detector, due to the 360° rotation. Thanks to the Radon transform [8], the measurements can be used to create a picture of the varying densities along a single slice of our object [9].

The unity of measurement of the scanner is the Hounsfield scale, measuring the radiodensity. Some important values and their respective hounsfield Unit (HU) are shown in Table 2.2.

There are however two complications with these values, making it impossible to be correctly displayed on the computer screen in this scale. A traditional grayscale image indeed consists of pixels between 0 and 255 if encoded in 8 bits or between 0 and 65565 if encoded in

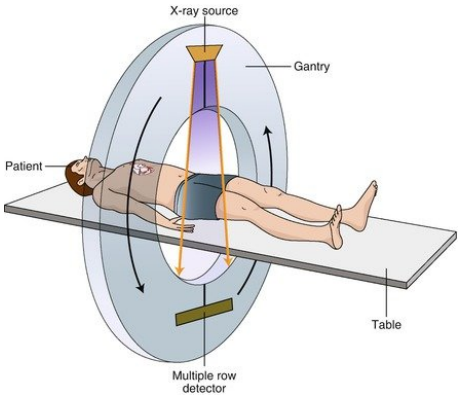


Figure 2.1: Illustration of the process of a CT scanner. From [10]

Substance	HU
Air	-1000
Fat	-120 to -90
Water	0
Soft tissue	+100 to +300
Bone	+700 to +3000

Table 2.2: Examples of Hounsfield units values

16 bits. Unfortunately, the Hounsfield scale includes negative values and has a range of approximately 4000 values. To deal with these problems, the computer transforms this unit by multiplying the values by a small number and adding a constant. As the coefficients of transformation depend on the scan, the standardisation of the data is lost. Fortunately, metadata is stored in order to recover the original HU data. Using Rescale Slope s in tag (0028, 1053) and Rescale Intercept i in tag (0028, 1054), it is possible to recover the original data from the value stored v using the following formula for each element of the array:

$$HU = s * v + i \tag{2.1}$$

The second standardisation comes from the pixels out of scanner. The rotation of the scanner indeed creates a circular scan, but a scan stored in a DICOM file contains a rectangle pixel array. In consequence, part of the pixel array is in fact out of the scanned zone. This can be seen on Figure 2.2, where the zone out of the scanner has artificially been put in evidence in order to see it. The value given to these pixels however depends on the scanner, which results in a crucial difference between different scans. It is there-

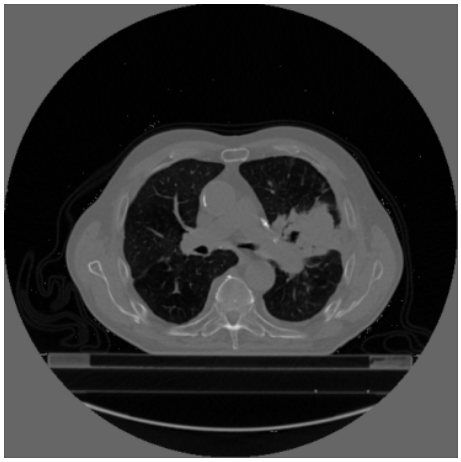


Figure 2.2: CT slice with out of scan zone artificially put in evidence. Medical image modified from [11].

fore important to find their value for each scan and to give them the constant value -1000, which correspond to the value of air in HU.

2.3 Anatomy of a RT-STRUCT

The doctor's segmentations are stored in the DICOM RT-STRUCT format. A single DICOM file is used for the full scan. The most important part of this kind of DICOM file is the tag (3006, 0039) that describes the ROI contour sequence (ROI stands for region of interest). The ROI is defined as a set of contour sequences stored in the (3006, 0040) tag. Each contour sequence is represented as an array containing a multiple of 3 elements. Each three elements can be interpreted as the (x, y, z) coordinates in mm. Together, the coordinates define the contours of the region of interest as a closed polygon.

In order to be useful, the coordinates (x, y) have to be converted from the spatial positions in mm to the current position on the scan. To do so, two other parameters are necessary. First, the tag (0020, 0032), which is called Image Position, stores the coordinates of the top left corner (t_x, t_y). Then, the Pixel Spacing (s_x, s_y) stores the distance in mm between two pixels. Using these values, the corresponding contour can be placed at the correct position (c_x, c_y) in the image using the formulas:

$$c_x = \frac{x - t_x}{s_x} \qquad\qquad c_y = \frac{y - t_y}{s_y} \tag{2.2}$$

2.4 From contours to masks

In practice, the contours of the organ are not very practical to work with. Instead, a classification of each pixel, depending of its inclusion in the organ, is necessary. To transform the coordinates of the borders into a mask, a standard scan-line polygonal filling algorithm has to be applied. In some cases, a RT-STRUCT can also be fully inside another one. When this case happens, it means that the internal one in fact delimits a region that must not be part of the mask. To deal with such cases, a XOR gate can be applied between the masks produced by the different borders. Figure 2.3 illustrates the process. The first image (top left) is a 2D slice of the scan. The second image (top right) shows the RT-STRUCT as encoded by the doctor. Finally, the last image is the mask as created by filling the polygons and applying a XOR gate.

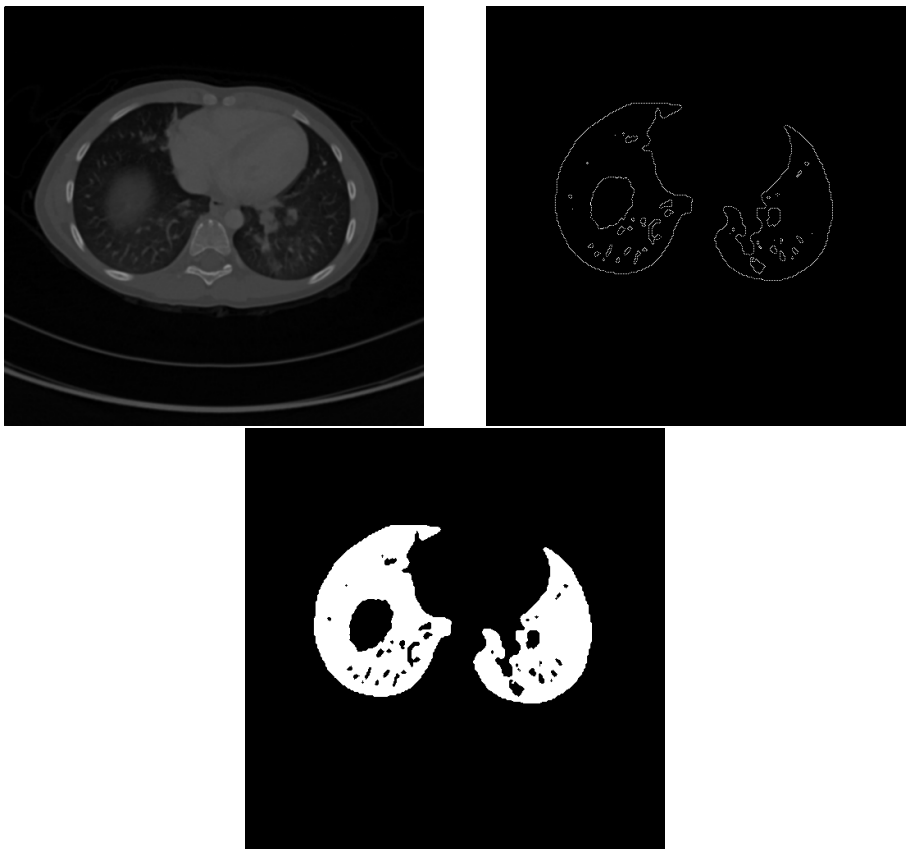


Figure 2.3: Medical image, RT-STRUCT contours and corresponding mask. Medical image from [11].

Chapter 3

Deep learning for image segmentation

Many strategies exist for image segmentation. In this section, different methods are explained and reviewed, and the ones that were implemented for this thesis are detailed. This chapter includes every useful aspect to reproduce the procedure, including the prevention of overfitting, the optimal training parameters and the loss functions. Finally, an evaluation metric is defined to give a score to a predicted segmentation, depending on its quality.

3.1 Sliding-window approach

The idea of the sliding-window strategy is to train a model smaller than the typical image size, using a pixel and some context around it (patch) [12] [13]. The model can then be applied everywhere on the full image, predicting a value of each pixel. This way of proceeding has two main advantages. First, it can be used to localise the region of interest, which is exactly the goal of image segmentation. Secondly, the number of training data in terms of patches is much larger than the number of images.

3.1.1 Dataset generation

To apply the sliding-window method, the first step is to decide the size of the patches that will be used to train the model. This is a compromise between accuracy and localisation. Small patches allow to localise with a high precision but can capture only little context,

while larger patches lose localisation accuracy due to the need of more max-pooling layers. For this thesis, patches of 31x31 were used. As a comparison, typical CT slices have a shape of 512x512.

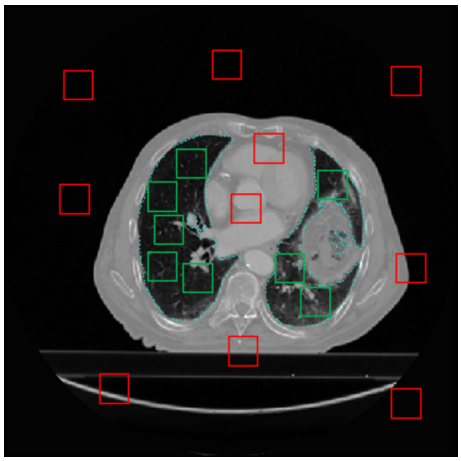


Figure 3.1: Example of patches for lung segmentation. Red patches are labeled with 0 and green patches are labeled with 1.

In order to create the patches, the pre-processing step simply takes the CT slices one after the other and extracts interesting parts of the image. For example, for lung segmentation, the algorithm checks if the slice in question contains lungs. If it does, it crops the image in the zone of the lungs in order to get 31x31 patches containing it. Then, it associates the label 1, meaning that this patch contains a lung. Of course, as seeing only lung images is not enough to learn something, the pre-processing step also crops the slice in order to have patches without any lung, and associates them with the label 0. To help the model to learn better, the new dataset is built with approximately the same number of lung and non-lung patches. Figure 3.1 illustrates a typical pre-processing of a slice for lung segmentation.

3.1.2 Segmentation

Once the model has been trained using the pre-processed data described above, it is ready to start the segmentation. In order to do it, the model starts to make a prediction of the 31x31 top left corner of the image to be predicted. The result of the prediction is then saved and corresponds to the central pixel of the patch. Then,

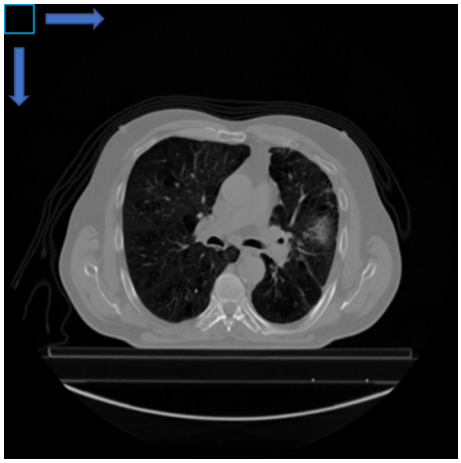


Figure 3.2: Segmentation of the image using the sliding window approach. 31x31 squares are analyzed everywhere on the image using the model trained and the central pixel is assigned the prediction.

the model is applied again one pixel to the right, in order to predict the next pixel. The model moves this way on the image until it has predicted all of them. This is illustrated in Figure 3.2. In the end, all pixels predicted to be 1 correspond to the prediction of the organ.

Notice that a 15 pixels margin is created from each border where the prediction can't be made as it misses context to apply the model. However, it doesn't matter here as this region is out of the human body in a CT scan and you can therefore assign a prediction of 0 to this margin.

3.1.3 Drawback of that approach

This approach has two important drawbacks. First of all, a patch is not sufficient to efficiently detect an organ. Indeed, it doesn't include the information of the entire image and a lot of spatial information is lost. Different organs can be very similar and spot the difference between them can't be done efficiently without using the spatial information of the full image.

Secondly, the time needed to make a segmentation is huge. To apply a model of length 31 pixel by pixel to the entire row of the image of length 512, the model has to be applied 512 - 30 = 482 times (as there is a margin of 15 at each side). Multiplying by the number of rows contained into the image of height 512, the model has to be called 482 * 482 = 232324 times in order to get the

whole segmentation done on one single slice. Using a powerful GPU (NVIDIA RTX 2080 Ti), the model takes 0.0005 seconds in order to make one prediction. The time needed to scan a whole CT slice is therefore $232324 * 0.0005 = 116,162$ seconds, which is almost two minutes. Finally, as what interests us is a full segmentation of the 3D CT scan, hundreds of slices may have to be passed through the model. Scanning all of them would take several hours, which is not acceptable in practice.

3.2 U-Net

U-Net [14] is built upon a fully convolutional network [15] in order to capture all the context of the images. The architecture can be seen in Figure 3.3. The name of this model comes from its u-shape.

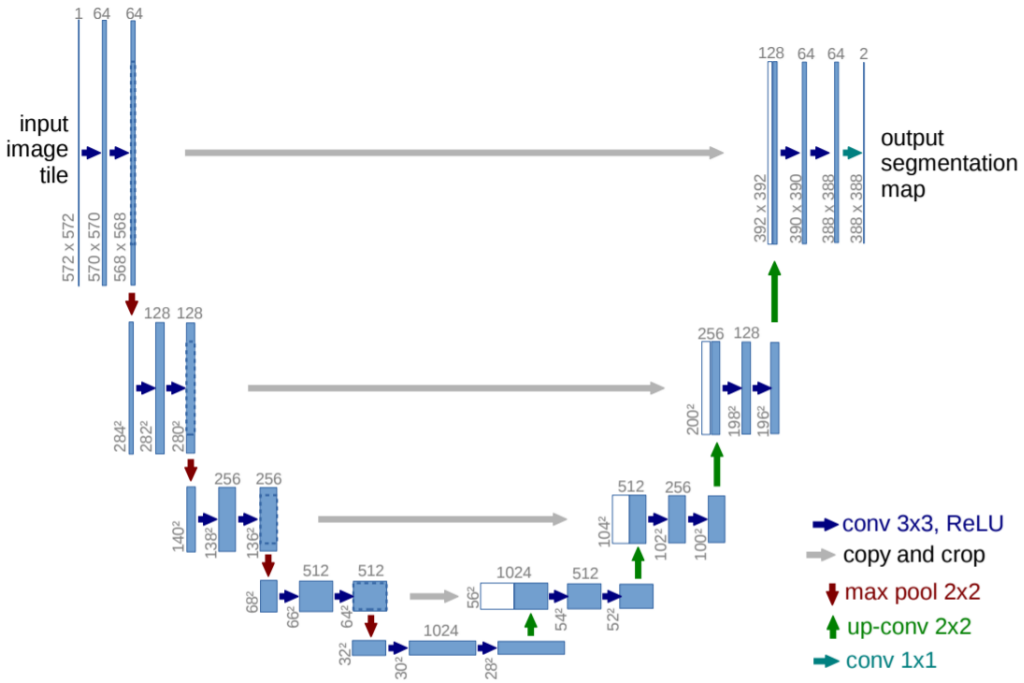


Figure 3.3: U-Net Architecture. Each arrow corresponds to operations that are described in the bottom-right corner. Blue rectangles represent multi-channel feature maps and white boxes correspond to copied feature maps. The number of channels are written on top of the boxes and the sizes are given at the lower left edge of the boxes. From [14]

The main idea is to use a contracting path (left size) in order

to capture context of the image, followed by an expanding path (right size) that enables precise localisation. The contracting path is composed of successive series of two 3x3 convolutions with a Rectified Linear Unit (ReLU) activation, followed by a 2x2 max-pooling operator. Each time this process is repeated, the number of feature maps is doubled. This allows the contracting path to learn many high-level features. Then comes the expanding path. It uses the features learned from the contracting path and applies successive series of 2x2 transposed convolutions followed by two 3x3 convolutions, again with a ReLU activation. This path allows to increase the resolution and to recover the original image size. Finally, the last layer is a 1x1 convolution that maps the feature vector to the desired number of classes and gives it as a final output.

Considering the disadvantages and the poor results of the sliding-window, this approach was replaced by the U-net model. This architecture has the advantage of taking the full image as input, giving it more context and resulting therefore in more accuracy. Finally, one segmentation takes less than one second.

3.2.1 Dataset generation

Using the manipulations made in chapter 2, the U-Net method can easily be applied to DICOM data. A valid dataset simply consists in the pixel array representing the images and their corresponding segmentation. For a better convergence, images that do not include the organ of interest must also be included to the dataset. For this thesis, an arbitrary number of 50% of slices before and after the ones containing the organ are added in the dataset for learning as well as for the evaluation of the score. The dataset therefore contains at most the same number of images that do not include the organ than images that include it. In practice, the number of images is not doubled because a scan does not always include 50% of images before or after the ones including the organ.

Each of the individual datasets are split into three independent parts: a training, a validation, and a test set, with 70%, 20% and 10% of the available patient's data respectively. The decomposition in terms of patients and not of 2D slices is very important here. Indeed, decomposing by slices would mean that the data of the same patient would be present in the three datasets that are supposed to

be independent, which would therefore bias the results.

3.2.2 Detailed architecture

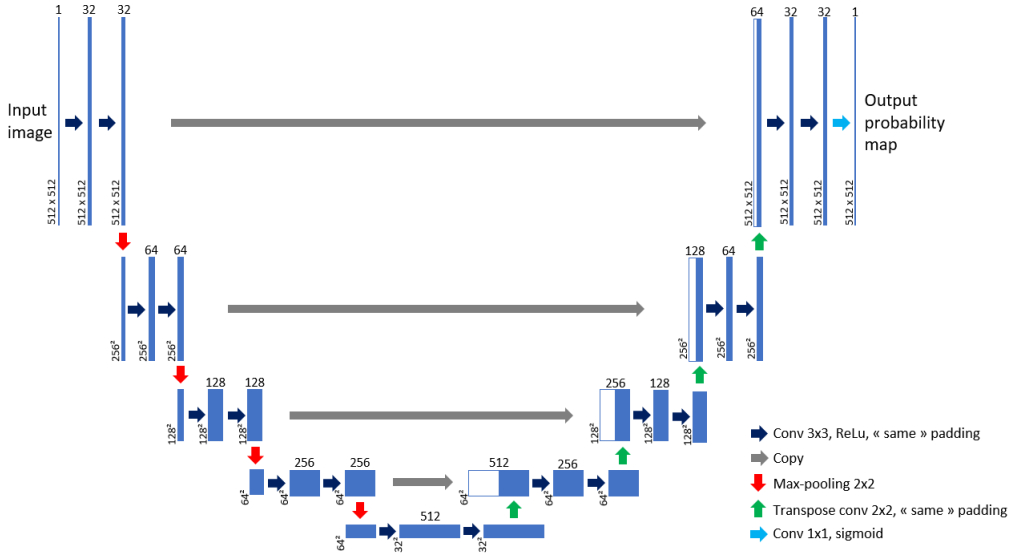


Figure 3.4: Detailed U-Net architecture used in this thesis. Blue boxes represent the features map resulting from a convolution. White boxes represent features map copied from a previous one. Figure adapted from [14]

.

My first contribution was to adapt the U-Net architecture presented in Figure 3.3 in order to be functional and efficient for this particular problem. The detailed architecture used is presented in Figure 3.4.

The main difference with Figure 3.3 comes from the input size of the image and the number of filters used for the convolutions. This version takes 512x512 images (as it is the image format of the CT scans) and starts with 32 to go up to 512 filters for the convolutions.

Another modification is the zero padding ("same" padding). Using a zero padding allows to preserve the x-y size of the image after the convolutions. As a result, the input size is equal to the output size and the full image is predicted.

After each max-pooling operator, a dropout layer is applied to the model. The dropout is an important improvement as it reduces

overfitting [16].

The very last convolution uses a sigmoid activation function for binary classification. As a result, the last 512x512 boxes represent the probability for each pixel of the initial image to be part of the organ of interest. In order to have a final prediction, the pixels with a probability higher than 0.5 are classified as being part of the organ of interest and the other ones are classified as outside of it.

Despite the capability of the U-Net method to segment different organs at the same time, binary classification was chosen in this thesis instead. The reason for that choice simply comes from the datasets. Indeed, every scan appearing in the datasets has different regions that are segmented. As an example, only 127 of the 422 patients of the first dataset, "NSCLC-Radiomics" [17], contain a segmentation of the heart. For the lungs and the oesophagus, there are 312 and 354 scans respectively that include these segmentations. To train U-Net using multi-organ segmentation, only the images containing all of these three organs could have been used, which represents a much lower amount of data, and is therefore not optimal.

Another advantage of doing three different segmentations is that it creates three different series of tests to analyse, with different distributions of the images over the datasets. This enables to have more results.

3.2.3 Loss functions

The model is initialised with random weights that are optimised during training using a loss function $\mathcal{L}(\hat{y}, y)$, where $\hat{y}$ is the class prediction for each pixel and y is the correct class.

Medical image segmentation is often highly unbalanced. Indeed, in a CT image, the organ can be a very small portion of the image. This is illustrated in Figure 3.5. The figure represents a segmentation of the oesophagus on a 2D slice of a CT scan. The mask of the oesophagus is a very small fraction of the image. Such unbalanced data is problematic when trying to maximise the classical metrics of accuracy. Indeed, a model predicting the class 0 for each pixel would result in a large score. To solve this problem, different loss functions can be used.

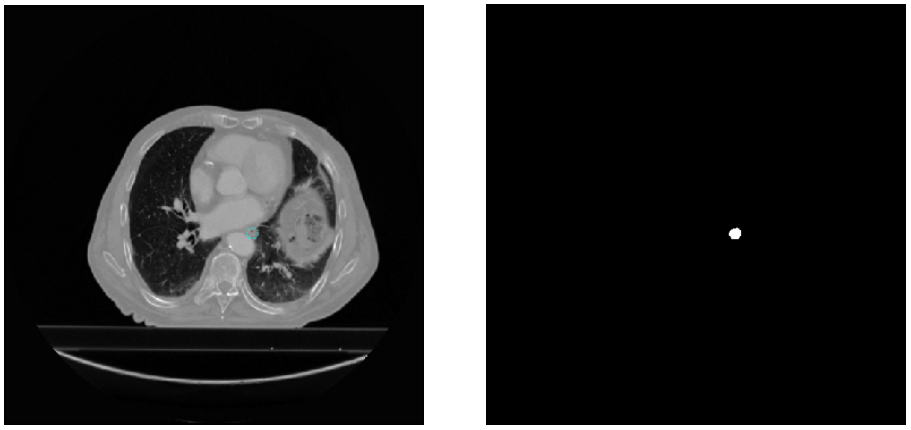


Figure 3.5: Medical image and its corresponding oesophagus segmentation. Medical image from [17]

.

Dice's loss

The Dice similarity coefficient (DSC) is a coefficient measuring the similarity between two sets of values. This value can be computed between the segmentation prediction and the mask in order to give a score to this prediction. The coefficient is defined as follows:

$$DSC = \frac{2TP}{2TP + FP + FN} \tag{3.1}$$

with:

- TP, the number of true positives
- FP, the number of false positives
- FN, the number of false negatives

This function can be rewritten as:

$$DSC = \frac{2|X \cap Y|}{|X| + |Y|} \tag{3.2}$$

with:

- X, the set of pixels predicted as true

- Y, the set of true pixels

The opposite of this coefficient can be used as a loss function and forces the model to predict positive items in order to reduce the loss [18]. Replacing X and Y by the indices and values of the pixels, the formula of the Dice's loss function used in this thesis is:

$$\mathcal{L}(\hat{y}, y) = -\frac{2\sum_i \hat{y}(i)y(i) + \epsilon}{\sum_{i'} \hat{y}(i') + \sum_{i''} y(i'') + \epsilon} \tag{3.3}$$

where i, i' and i'' are all possible indices of the pixels and ϵ, a non-zero constant.

The introduction of the ϵ parameter, which is not in the original Dice's formula, has two main reasons. First of all, it avoids the division by 0 when there are only true negatives. Secondly, this coefficient makes the derivability smoother when the organ is not present and helps the algorithm to learn more efficiently. Finding an optimal value for ϵ must therefore not be underestimated. If too small (e.g. 1), the training will not converge because of the images that do not contain the organ. Indeed, predicting only one positive pixel when there is none would make the coefficient instantly drop from 1 to 0.5. On the contrary, a too large value (e.g 100 000) will not penalise enough the predictions which includes an organ that is not present. The optimal value for ϵ is detailed for each organ in chapter 5.

The advantage of this loss is that predicting only zeros would result in a poor score.

Jaccard loss

Another interesting metric of similarity is the Jaccard index, also called "intersection over union". As its name suggests, it simply takes the ratio of the intersection over the union of two sets of values:

$$J = \frac{|X \cap Y|}{|X \cup Y|} \tag{3.4}$$

Based on this coefficient, the Jaccard loss can be defined as:

$$\mathcal{L}(\hat{y}, y) = -\frac{\sum_i \hat{y}(i)y(i) + \epsilon}{\sum_{i'} \hat{y}(i') + \sum_{i''} y(i'') - \sum_{i'''} \hat{y}(i''')y(i''') + \epsilon} \tag{3.5}$$

where i, i', i'' and i''' are all possible indices of the pixels. The ϵ parameter is present in the loss exactly for the same reason as the ϵ of the Dice's loss function

In practice, the Jaccard loss is very similar to the Dice's loss. Indeed, the relation between them is shown as below:

$$J = \frac{DSC}{2 - DSC} \qquad\qquad DSC = \frac{2J}{J + 1} \tag{3.6}$$

3.2.4 Preventing overfitting

Overfitting is the problem of corresponding too closely to the training set and unknowingly extracting features that are specific to it and do not generalise well [19]. In addition to the dropout that is included in the architecture itself, different strategies are used to avoid overfitting.

Data augmentation

Data augmentation consists in artificially increasing the size of a dataset using label-preserving transformations [20]. This is widely used to reduce overfitting in the segmentation of medical images because it gives more diversity without requiring additional annotated data.

To add more robustness to the network, data augmentation is used. During training, a generator creates slight modifications of the initial images at each epoch. This way, the images seen by the network are different at each epoch and the dataset size is artificially increased. This strategy is called online augmentation

[21]. However, as medical images have the advantage of being quite standardised (centred, same orientation), the modifications done by the generator must be restricted. Indeed, reversing or completely deforming the image might break this standardisation and make the recognition task harder for the network. Therefore, only slight layout transformations are used in practice [22].

Validation set

To get the optimal number of epochs, a checkpointer was added as an early stopping criteria. A training set (70% of the data) is used for learning and, after each epoch, a validation set (20% of the data) is used to evaluate the quality of the network. The model is only saved if the evaluation results in a better score than the previous best model. After a predefined number of epochs without improvement, the training is automatically stopped. This optimisation is a good way to prevent overfitting and to get the number of epochs resulting in the best generalisation.

3.2.5 Optimisation of the hyperparameters

Learning rate

The learning rate is a critical parameter that can make any model completely unable to perform if badly chosen for training. The importance of its optimisation must therefore not be underestimated. Many tests were executed for each organ in order to get the best possible value for this parameter.

Because the training time of the network is quite long (often more than 10 hours using all the data), a learning rate scheduler was implemented to try to speed up the process. The idea is to start the training procedure with a higher learning rate that would slowly decrease after each minibatch to tend to the optimal learning rate found with the previous tests. Despite looking great, this idea was given up due to disappointing results in practice. Indeed, starting with a higher learning rate has the consequence of making the training procedure more unstable which resulted in a Dice score sometimes dropping to 0. Moreover, even when working fine, the Dice coefficients of the final model were slightly inferior in practice to the ones that were observed without this strategy.

Batch size

Due to the large size of the images (512 x 512, 16 bits per pixel), the batch size is limited by the memory limits. Counter-intuitively, after different tries with various batch sizes, the optimal one was simply found to be 1. This value corresponds to the batch size used in the original U-Net paper [14].

3.2.6 Recent evolutions of U-Net

U-Net is a well-studied topic which is used as a basis of many U-shaped networks, with the goal of outperforming the original version. Despite the fact that the original one remains the state of the art nowadays, some of its evolutions give very promising results and might replace it in the future. This section briefly introduces these approaches as well as the intuition behind them. All of them are detailed in Appendix B.

- **3D U-Net:** 3D U-Net is similar to the original U-Net, with the difference that the input is extended to three dimensions instead of two.
- **V-Net:** This approach, also in three dimensions, introduces variations in the layers used by the network. The most notable one is the replacement of the original convolution filters by volumetric kernels of size 5x5x5 voxels.
- **MultiResUNet:** MultiResUNet tries to correct the shortcomings of the original U-Net to deal with objects of different sizes. To do so, the series of two successive convolutions of U-Net are replaced by newly introduced "MultiRes blocks" which have their own structures. Furthermore, the skip connections are replaced by "Res paths", resulting in a much more complex architecture.

3.3 Evaluation metric

For the reasons explained in subsection 3.2.3, accuracy cannot be used as an efficient evaluation metric for semantic segmentation. The Dice's coefficient is a better alternative. However, to have a relevant measurement of performance for a model, evaluating the score for a set of slices and taking the average of the scores is not meaningful. Indeed, in three dimensions, a slice containing many

pixels of the organ of interest is more important to be accurate than a slice with only a few of them. To have a meaningful evaluation metric, the 3D version of the DSC is used instead.

$$DSC(3D) = \frac{2\sum_i \hat{y}(i)y(i)}{\sum_{i'} \hat{y}(i') + \sum_{i''} y(i'')} \tag{3.7}$$

where i, i' and i'' are all possible indices of the pixels. The coefficient is exactly the same as the classical DSC, except that the evaluation is made volume by volume instead of slice by slice. Of course, the ϵ parameter of the loss function is not needed for the evaluation metric as the coefficient must not be derivable, and the tests are only made on 3D scans where the organ is present, ensuring that there will not be any division by 0.

3.4 Implementation

This section explains the most important functions that were implemented in order to use the U-Net method in practice. The code was written in Python using the Keras API from the Tensorflow library for deep learning.

3.4.1 Defining the model

The creation of the model is encapsulated in the function *get_model* which creates and returns the architecture presented in subsection 3.2.2. Figure 3.6 illustrates the beginning of the function creating the model. The first line defines the input of the model. Then the two first convolutions, the max-pooling layer and the droupout layer are defined. This corresponds to the top left part of the U-Net architecture.

These kinds of layers are repeated four times, until reaching the maximum number of features map. The corresponding code is shown in Figure 3.7 which corresponds to the bottom of the U-Net model.

```
inputs = Input((512, 512, 1)) # size of the images

conv1 = Conv2D(32, (3, 3), activation='relu', padding='same')(inputs) # 3x3 convolution
conv1 = Conv2D(32, (3, 3), activation='relu', padding='same')(conv1)  # 3x3 convolution
pool1 = MaxPooling2D(pool_size=(2, 2))(conv1) # max pooling layer
drop1 = Dropout(0.5)(pool1) # dropout layer

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(0.5)(pool2)
```

Figure 3.6: Beginning of the function get_model.

```
conv5 = Conv2D(512, (3, 3), activation='relu', padding='same')(drop4)
conv5 = Conv2D(512, (3, 3), activation='relu', padding='same')(conv5)
```

Figure 3.7: The middle of the function *get_model*, between the contracting and the expanding path, at the point where the number of features map is maximum.

Finally, Figure 3.8 illustrates the code at the end of the function. It creates the skip connections, which references the results of the first convolutions. Then, two 3x3 convolutions are defined, followed by the 1x1 sigmoid activation that defines the output. In the last lines, the model is created using the input and output layers, and the function returns it.

```
up9 = concatenate([Conv2DTranspose(32, (2, 2),
                                   strides=(2, 2),
                                   padding='same')
                   (conv8), conv1], axis=3)         # skip connections

conv9 = Conv2D(32, (3, 3), activation='relu', padding='same')(up9)   # 3x3 convolution
conv9 = Conv2D(32, (3, 3), activation='relu', padding='same')(conv9) # 3x3 convolution

conv10 = Conv2D(1, (1, 1), activation='sigmoid')(conv9) # 1x1 sigmoid activation

model = Model(inputs=[inputs], outputs=[conv10]) # define the model

return model
```

Figure 3.8: End of the function *get_model*.

3.4.2 Loss functions

Other important functions of this part of the code are the loss functions. The Dice's loss function and Jaccard's loss were implemented using their standard formula with the backend from Keras, as illustrated in Figure 3.9.

```
y_true_f = K.flatten(y_true)
y_pred_f = K.flatten(y_pred)
intersection = K.sum(y_true_f * y_pred_f)
return -(2. * intersection + smooth) / (K.sum(y_true_f) + K.sum(y_pred_f) + smooth)
```

Figure 3.9: The body of the Dice's loss function.

3.4.3 Data augmentation

The function *dataAugmentation* was introduced in order to make the data augmentation as discussed in subsection 3.2.4. To start this function, the dataset must be organised as follows:

```
images
├── 0.0.png
├── 0.1.png
├── ...
└── m.n.png
masks
├── 0.0.png
├── 0.1.png
├── ...
└── m.n.png
```

The images and the masks are stored as "x_y.png", where x is the patient's number and y is the number of the slice. In the above directory tree, the training set therefore contains m patients, and the last patient has slices up to index n.

This function takes as arguments the path to the directory of the data and a parameter telling the function if the data augmentation has to be made for training or not. If the data is for training, the function creates two generators with active modifications (zoom and rotation). The two generators are declared using the same arguments, including a common seed. They respectively produce

images and masks from the original images included in the dataset. The common parameters allow to force the generators to perform the exact same modifications on the images and the masks, such that the masks remain accurate. If the data is not for training, the zoom and rotation are deactivated, as the goal of a validation and a test set is to try the model with the original images, and not the modified ones. The function ends by adjusting the images and masks generated. A small random brightness change is applied at this step only if the data is defined to be for training. The masks are also ensured to have only pixels with a value of 0 or 1 in any case.

3.4.4 Main function

The main function that implements the centralised model is called *SimpleSGD*.

The first argument of the main function is the path to the dataset. To work, the dataset must include both a training and a validation set, each organised as for the *dataAugmentation* function. The dataset is as follows:

```
dataset
├── training
└── validation
```

The second argument is a string giving the name of the model to load in order to use transfer learning, or *None* if the weights must be initialised from scratch. The function starts by creating the model using *get_model*, and then compiles it using the Adam optimiser and one of the loss function defined above.

The function then creates the training and the validation generators, using the function *dataAugmentation*. The model can then be fitted using the generators. The callbacks are used in order to keep the best models and to interrupt the training when no improvement is being made on the model for a certain amount of time. When the model has been trained, the function returns it.

3.4.5 Evaluation function

Finally, the function *test_model_3d* is used in order to evaluate the 3D Dice's coefficient of the model. Its first argument is the path to the test set which must be organised as the training and the validation set. The second argument is the name of the trained model.

The function starts by ordering the slices. Then, for each slice, the model is used to make a prediction based on the image. The number of intersection points and the number of positive pixels of the true mask and the predicted masks are then collected until the end of a patient's scan has been completely ran. When a slice is the last one of a patient's scan, the values collected make it possible to compute the 3D Dice's coefficient of this patient and the process is repeated for the next one until each patient's score has been predicted. Once done, the function returns the mean of the patients' score, which is the final 3D Dice's coefficient.

Chapter 4

Federated Learning

In order to efficiently train machine learning algorithms, abundant annotated data is a key part. However, in the medical field, producing manual segmentation is expensive and time consuming for the doctors. For this reason, it is very complex for a single hospital to reach a large enough sample to build an efficient personal model for automatic images segmentation [23] [24]. A naive solution would be to centralise the data from many different hospitals in order to create a large annotated dataset usable to train the network. Unfortunately, this is not feasible in practice due to the privacy of the medical data.

This chapter presents an alternative solution called Federated Learning [25] which raises a lot of interest in the medical field nowadays [26]. The specificity of Federated Learning is to train the network using a decentralised approach [6]. The strategy and its implementation are detailed, as well as the related challenges involved. Finally, a new algorithm called Federated Equal-Chances from my contribution is introduced, as well as a detailed protocol necessary for its implementation in practice.

4.1 State of the art

In Federated Learning, the hospitals only need to communicate updates of the model, and the need of extracting the medical data is lost. As a consequence, the protection of clinical data is guaranteed by design [5].

4.1.1 Federated Averaging

Algorithm 1 Original Federated Averaging. Rewritten from [6]

function SERVER
  initialize w_0 ▷ initial global model
  for each round t=1,2,... **do**
    $m \leftarrow max(C * K, 1)$
    $S_t \leftarrow$ (random set of m clients) ▷ round's clients
    for each client $k \in S_t$ **in parallel do**
      $w_{t+1}^k \leftarrow ClientUpdate(k, w_t)$
    end for
    $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$ ▷ central model update
  end for
end function
function CLIENTUPDATE(k, w)
  $\beta \leftarrow$ (Split P_k into batches of size B)
  for each local epoch i from 1 to E **do**
    for batch $b \in \beta$ **do**
      $w \leftarrow w - \eta \nabla \ell(w; b)$
    end for
  end for
  return w to server
end function

The Federated Averaging algorithm (*fedAvg*) is a simple yet very efficient algorithm from decentralised data. The pseudocode of the algorithm can be seen in Algorithm 1. As seen in Figure 4.1, a central model communicates with the hospitals in order to update its weights. The number K of hospitals is fixed. Federated Averaging starts by selecting a fraction C of the hospitals and sends them the current state of the model. Each selected hospital then uses its local dataset to perform a computation, using E epochs and a batch size B. Once a client finished its computation, it only sends its local updates to the central server that changes its weights in consequence. The step is repeated a number N of rounds. The values of these parameters are detailed in section 4.2.

To perform its update, the central model computes the weighted average of the new weights of the clients that made the update. In mathematical terms, if a client k trains the model at time t, the local update for $t + 1$ is w_{t+1}^k and the global update is given by:

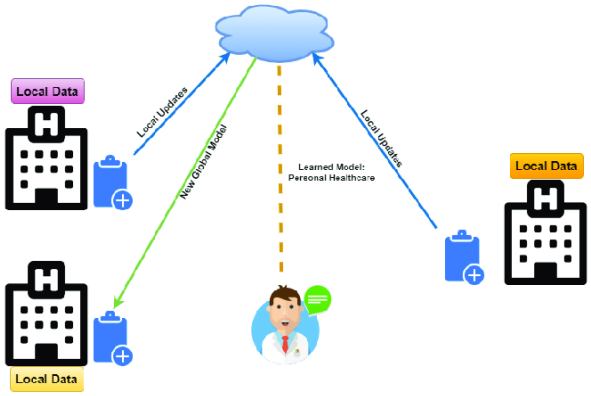


Figure 4.1: Federated Learning architecture applied in a hospital setting. Illustration from [27].

$$w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k \tag{4.1}$$

with:

- n_k, the size of the local dataset
- $n = \sum_{k=1}^{K} n_k$, the total size of the different datasets

4.1.2 Distribution of the data between hospitals

Despite its ability to apply machine learning while preserving the privacy of the data, Federated Learning has the drawback of being dependent of the distribution of the data over the decentralised datasets [28]. Indeed, as the global model is trained locally by the different clients, a client that does not well represent the overall distribution of the data cannot efficiently train the model. Furthermore, the performance of the global network is also affected.

More formally, a neural network can be trained efficiently if the data between the clients is independent and identically distributed (iid). Unfortunately, in the case of hospitals, the overall data is non-iid. Figure 4.2 illustrates the distribution of the data in this case. As you can see, the data inside a local hospital is globally iid (despite some intra-variability like the motivation of a same physician for a segmentation). However, the data between the different hospitals is non-iid. The differences are explained by the inter-variability including the scanners, the settings, the doctors,

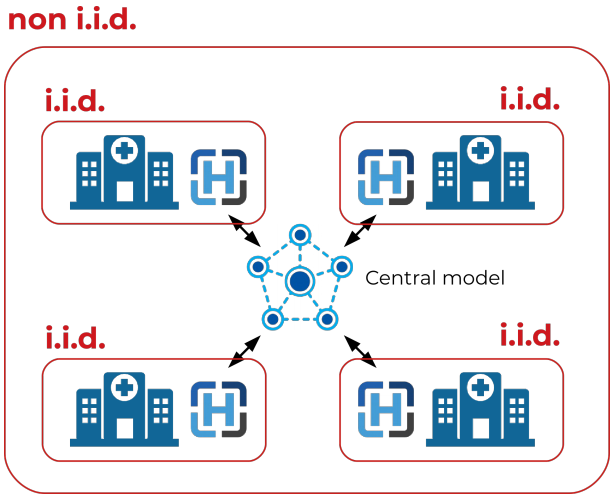


Figure 4.2: Illustration of Federated Learning by Sébastien Jodogne

the amount of data, etc.

4.2 Local implementation of Federated Averaging

To evaluate the performances of Federated Learning without the need of a real communication with hospitals, Federated Averaging can be implemented locally on one computer. The only difference comes from the communication between the datasets. As illustrated on Figure 4.3, in a local computer, three different datasets (described in section 5.1) are used in three separated folders. Federated Averaging is then applied on the distinct datasets and only the weights are used to update the central model.

In regards to the similarity with the real case scenario, the non-iid distribution of the data is respected. Indeed, three distinct datasets are used to simulate the data of a hospital. As the datasets are unrelated to each other, the global data is non-iid. Finally, the sizes of the datasets are very different (60 patients for the smallest one and 422 for the biggest one), which simulates well the behaviour of the hospitals, where small hospitals can be used for training as well as bigger ones.

In this implementation, as the number of datasets is small, all of them are selected at each round. In order to update the central

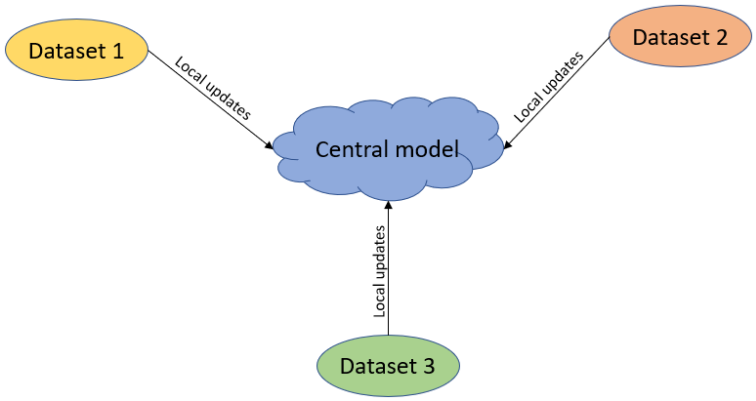


Figure 4.3: Federated Learning applied in a local computer with three different datasets.

model as often as possible, only one epoch is used for each dataset in a round. As explained in subsection 3.2.4 the batch size is equal to 1. Finally, in order to decide on the optimal number of rounds, we use the same strategy as in subsection 3.2.5 (a checkpointer with a validation set). The procedure to apply this strategy in a Federated Learning framework is detailed in subsection 4.4.1. The values are therefore $K = 3$, $C = 1$, $E = 1$, $B = 1$ and N without a fixed value.

4.3 Dealing with inter-variability of the data

The main difficulty with the distribution of the datasets is that there are multiple factors of inter-variability between them. On the other hand, intra-variability is more limited. Consequently, the local training of the models might only represent an efficient update regarding the local data but their average is not guaranteed as such.

To deal with this problem, one might suggest to use data augmentation on the local datasets. This will lead to increase the general updates to the local trainings and to make the global model more efficient. As for the data augmentation used in a centralised model, the variations on the images might remain small and represent real possible differences between one CT scan and another. The idea is to transform a dataset with very similar scans into an artificially more general dataset but to keep realistic scan images. Full rotation of the images has for example no real interest in this case.

Three different data augmentation techniques are used in this purpose. First of all, a slight zoom is applied. Indeed, depending on the doctor and the scanner, the zoom can naturally be different between a scan and another. Secondly, a slight rotation is applied on the image, simulating possible irregularities between the position of the patients. Finally, the intensity of the pixels is modified to add noise in the brightness of the images. This last change might sound surprising as the pixels intensity is supposed to be standardised, as explained in section 2.2. However, in practice, the calibration of the scanner can result in slightly incorrect results (e.g a scanner that would give HU values slightly under the reality). Therefore, small brightness changes can have their importance in this case to avoid a misunderstanding between local models. The optimal parameters for these three techniques is detailed in chapter 5.

4.4 Federated Equal-Chances

When different hospitals present large differences in sizes, the classical Federated Averaging approach might tend to discriminate the smallest ones. Indeed, the algorithm updates the central model proportionally to the amount of data made available in the hospitals. Despite being relevant for most of the data, this might affect the quality of the final model for the smallest hospitals.

To deal with this problem, my contribution was to modify the original *fedAvg* in order to give more credit to the smaller datasets. The idea is to use data augmentation to generate more images on the small datasets, as opposed to the big ones. Doing so, every client can have the same training size than the biggest hospital at each round. More formally, if the size of the biggest dataset used in a round is M, every datasets contributing will train a number M of images. For a dataset of size x, the number of augmented images generated for each original image is therefore $\frac{M}{x}$. With this new strategy, the local updates of all datasets have exactly the same weight and the simple average can be used for the global update of each round. This new version of the algorithm will be called Federated Equal-Chances, as its goal is to give the same credit to small and big datasets. The scores of these two algorithms are detailed in chapter 5.

4.4.1 From the local simulation to a real framework

To be able to expand the local simulation of Federated Equal-Chances in a real case scenario, a full protocol is needed as some steps need values that would not be accessible by default. This section describes the whole process to implement it in real life.

The main difference between the Federated Equal-Chances (Algorithm 2) and the original Federated Averaging (Algorithm 1) comes from the bigger amount of generated images on the smaller datasets in order to have the same number of images at each epoch for all of them. In practice, knowing the size of the biggest dataset of the round is mandatory to do so. To deal with this problem, the GetClientSize(k) method was added to the algorithm. This function simply takes a client and asks it the size of its training set. The server starts the round by contacting the m clients and waits for their response. The server can then simply take the maximum s_{max} of the values and get the necessary information.

This step is illustrated in Figure 4.4. In this example, there are a total of 5 hospitals and 3 of them communicate during the round ($K = 5$, $m = 3$).

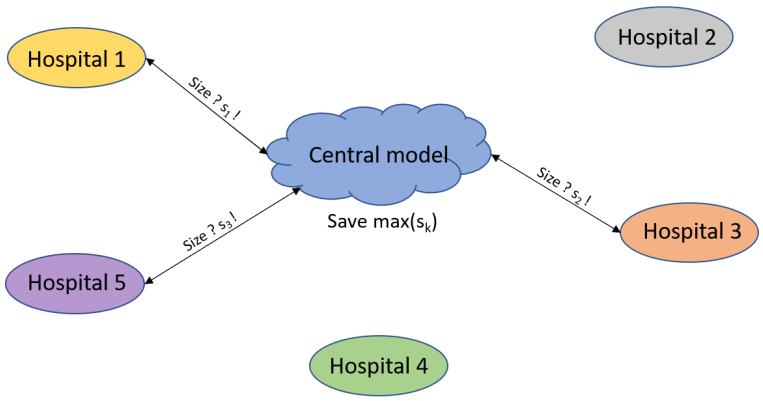


Figure 4.4: The first step of the round is represented here. The server asks the sizes of the clients and saves the maximum.

This solution generates more client-server communication as the server has to ask the clients their maximum size. Fortunately, the communication added is limited compared to the original one coming from *ClientUpdate*. Indeed, in the latter, the clients have

Algorithm 2 Federated Equal-Chances. Modified from [6]

function SERVER
  initialize w_0 ▷ initial global model
  $t \leftarrow 1$ ▷ round number
  $hs \leftarrow 0$ ▷ highest score so far
  while not overfitting **do**
    $m \leftarrow max(C * K, 1)$ ▷ number of clients selected
    $S_t \leftarrow$ (random set of m clients) ▷ round's clients
    for each client $k \in S_t$ **in parallel do**
      $s_k \leftarrow GetClientSize(k)$
    end for
    $s_{max} \leftarrow \max(s_k)$ ▷ biggest client's length
    for each client $k \in S_t$ **in parallel do**
      $w_{t+1}^k \leftarrow ClientUpdate(k, w_t, s_{max})$
    end for
    $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{1}{m} w_{t+1}^k$ ▷ central model update
    for each client $k \in S_t$ **in parallel do**
      $sc_k \leftarrow EvaluateModel(k, w_{t+1})$ ▷ test score of k
    end for
    $sc_{max} \leftarrow \frac{1}{m} \sum_{1=k}^{m} sc_k$ ▷ compute the average of the scores
    if $sc_{max} > hs$ **then** ▷ if score better than current best
      $hs \leftarrow sc_{max}$ ▷ update best score
    end if
  end while
end function
function EVALUATEMODEL(k, w)
  return (test score of model w on client k) to server
end function
function GETCLIENTSIZE(k)
  return (size of dataset k) to server
end function
function CLIENTUPDATE(k, w, s)
  for each local epoch i from 1 to E **do**
    $b \leftarrow 0$ ▷ batch counter
    while $b < s$ **do**
      (Generate new augmented images)
      $w \leftarrow w - \eta \nabla \ell(w; b)$ ▷ minibatch's update
      $b \leftarrow b + bs$ ▷ b incremented with batch size
    end while
  end for
  return w to server
end function

to communicate the update of the weights which can be expensive due to the large number of weights of the network (here, U-Net has over 7 million trainable parameters). The added communication of the client's size, which is just an integer, is therefore negligible and might not lead to serious complications. Another possible strategy would also be to start the algorithm by asking all the clients' dataset size. Both methods work and give the same results.

When the first step is over, the algorithm can communicate the weights of its central model as well as the s_{max} that was just computed, as seen in Figure 4.5. The hospitals can then locally train this model with their local data, using data augmentation in order to train the model on s_{max} images.

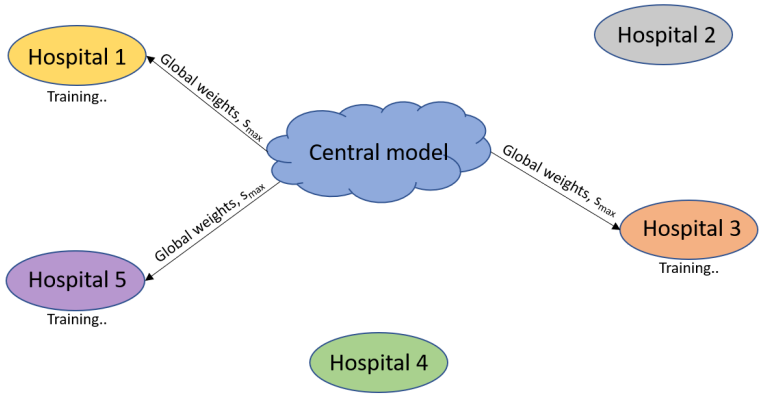


Figure 4.5: The second step of the round is represented here. The server sends the weights of the central model as well as the size of the biggest hospital's dataset of the round. The hospitals train this model with their local data using s_{max} images in total.

When the hospitals finish their training, they send the updates to the central server. As the small hospitals used more data augmentation, all of them can be treated equally and the server simply updates its central model with the average of the updates. This step is illustrated in Figure 4.6.

The second difference comes from the early stopping criteria. To evaluate the model and know when to stop, the central model has to refer to the test evaluation made by the clients (as a reminder, only the clients hold the data). Again, this requires a new communication with the server. The server must receive the updates, change its global model accordingly and send the new weights again

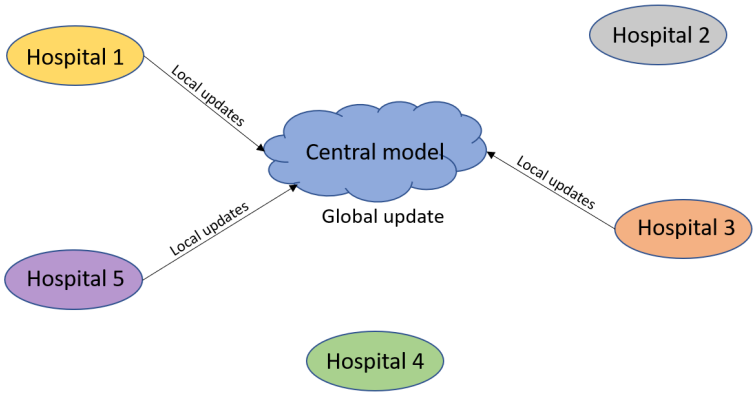


Figure 4.6: The third step of the round is represented here. The hospitals send the updates to the server that updates its central model.

to the clients, as showed in Figure 4.7. The clients can then test the new model with their validation set and transmit their score to the server, as in Figure 4.8. The server can use them to save the best version only and stop training when the average validation score stops improving.

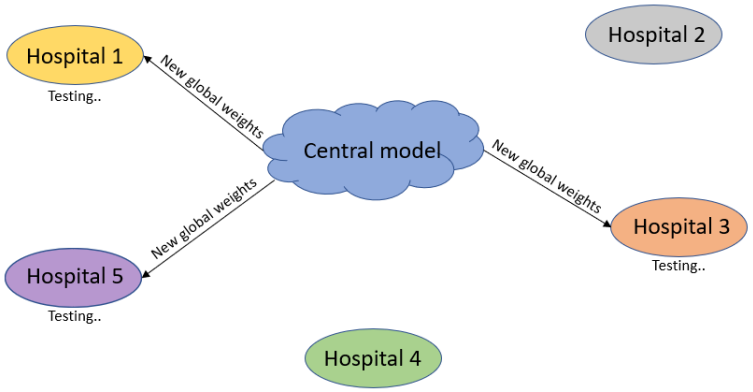


Figure 4.7: The fourth step of the round is represented here. The server transmits the new weights to the hospitals that can then test the model locally.

Doing so is quite expensive because the heavy weights have to be transmitted again. To avoid such double communication, it would be possible to ask the clients of the next round to start by evaluating the last model before training it for updates. This would avoid the double weights communication.

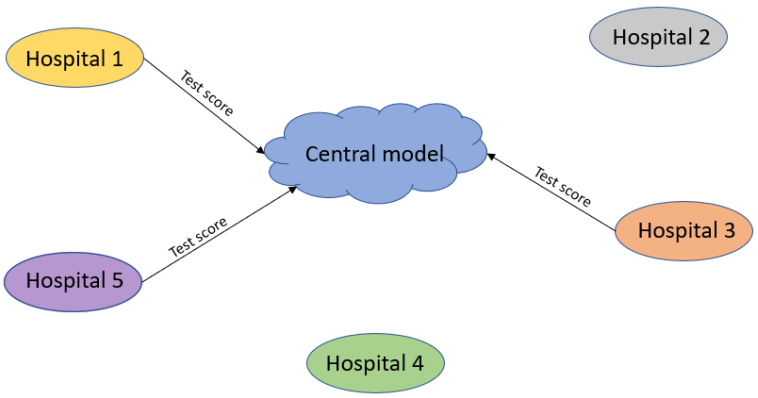


Figure 4.8: The last step of the round is represented here. The hospitals send their test score to the server that uses this information to know when to stop or save the model.

4.5 Implementation

To process Federated Learning, the dataset has to be organised as follows:

```
dataset
├── client0
│   ├── training
│   └── validation
├── client1
│   ├── training
│   └── validation
└── client2
    ├── training
    └── validation
```

where each client has its own training and validation set. Some intermediate functions are also necessary. First, *get_ratio_of_clients* takes the path to the dataset and the number of clients to compute the relative size of each client's training set. Two other functions, *scale_model_weights* and *sum_scaled_weights*, respectively scale the weights of a model relatively to its size and sum the scaled weights of multiple clients.

The main function starts by computing the relative size of all clients using *get_ratio_of_clients* defined earlier. Then, like the

standard SGD algorithm of section 3.4, the function defines the central model and compiles it. However, instead of simply training the model, the function starts a loop that is repeated N times, where N is the number of communication rounds. The loop starts by storing the weights of the central model and by defining the set of clients of size $C * K$ that will participate to the round. As a reminder, $C = 1$ and $K = 3$ in this thesis so all the clients participate at each round.

The function continues with a second loop inside the first one, representing the operations done by each client in the round. The loop starts by creating a model and setting the weights of that model to the ones of the central model that were stored before. The client then creates a training generator with its own data and fits the model with it. In this case, a callback is used to store the best model only, but no *patience* is used from Keras directly. The client ends by getting its new weights, and scales it with its relative size compared to the other clients.

Once all the clients have trained the model based on their weights, the function calculates the average of the scaled weights computed by each client using the functions defined earlier and sets the new weights to the central model to these new weights. Another loop then starts for each client, which creates a validation generator to evaluate and store the score of the new global model on their test set. Once the evaluation was made for all the clients, the function takes the average and compares it to the previous best score of an earlier round (the *best_score* is equal to 0 initially). If the score is the best one so far, the *wait* parameter is reset to 0 and the model is saved. If the score is not the best, the *wait* parameter is incremented by 1. If $wait > patience$, the main loop ends and the model is returned.

Federated Equal-Chances is implemented in another function called *fedEq*. In practice, the implementation of these two functions are very similar. The difference lies in the training of the model. Figure 4.9 shows the line of code defined to fit the model in Federated Averaging while Figure 4.10 shows this line in Federated Equal-Chances. The difference between the two lies in the *steps_per_epoch*. The first one simply takes the size of the client while the second one uses the relative weights of the client to train on the same number of images as the biggest model.

After the training, the new weights are not scaled as the central

```
local_model.fit(training_generator,
                steps_per_epoch=len_training, epochs=epo, shuffle=True,
                callbacks=callbacks, verbose=1)
```

Figure 4.9: Training the local model of a client in Federated Averaging using Keras.

```
local_model.fit(training_generator,
                steps_per_epoch=len_training*(clients_weight[biggest_client]/clients_weight[client]),
                epochs=epo, shuffle=True,
                callbacks=callbacks, verbose=1)
```

Figure 4.10: Training the local model of a client in Federated Equal-Chances using Keras.

model simply takes the average of the new weights in this version of the algorithm.

Chapter 5

Results

5.1 Datasets

All the datasets used in this thesis come from The Cancer Imaging Archive (TCIA) [4], a service which hosts anonymised medical images of cancers accessible for public domain. The datasets use the DICOM format. The input images used are CT scans and the human-made segmentations are encoded in the RT-STRUCT format. In total, three unrelated datasets are used in this thesis. This is an important aspect as it makes the datasets non-iid, which permits a more realistic evaluation of Federated Learning applied to hospitals. The three datasets are detailed below.

5.1.1 NSCLC-Radiomics

NSCLC-Radiomics [17] is a dataset containing 422 non-small cell lung cancer (NSCLC) patients. For all these patients, a radiation oncologist has made a manual delineation of the gross tumour volume. Most of the patients also contain a manual segmentation of the organs of interest, but not all of them.

This dataset was created for a publication which goal is to decode tumour phenotype by noninvasive imaging, applying a radiomic approach to CT data of patients with lung or head-neck cancer [29]. In total, 312, 127 and 354 patients have an available segmentation of the lungs, the heart and the oesophagus respectively.

5.1.2 LCTSC

Lung CT Segmentation Challenge (LCTSC) [30] is a dataset which was provided for a challenge competition which was conducted at the AAPM 2017 Annual Meeting. The objective of the competition was to provide a platform for comparison of various auto-segmentation algorithms to delineate organs at risk from CT images for thoracic patients in radiation treatment planning [31].

This collection consists in CT images of 60 patients. Out of these 60 patients, 59 of them have an available heart, lung and oesophagus segmentation.

5.1.3 Pediatric CT-SEG

This dataset [11] was collected by a collaboration of researchers from Children's Wisconsin, Marquette University, Varian Medical Systems, Medical College of Wisconsin, and Stanford University. This was part of a project funded by the National Institute of Biomedical imaging and Bioengineering. The goal was to develop tools for fast patient-specific CT organ dose estimation [32].

This dataset contains CT scans of 359 pediatric chest-abdomen-petris or abdomen-pelvis exams acquired from three scanners. The patients were selected from random pediatric cases from routine clinical indication. In other words, these are normal children without a particular disease such as cancer. The dataset totals 349, 345 and 347 patients with a valid segmentation of the heart, lungs and oesophagus.

5.2 CISM and CÉCI cluster

Computational resources have been provided by the supercomputing facilities of the Université Catholique de Louvain (CISM/UCL) and the Consortium des Équipements de Calcul Intensif en Fédération Wallonie Bruxelles (CÉCI) funded by the Fond de la Recherche Scientifique de Belgique (F.R.S.-FNRS) under convention 2.5020.11 and by the Walloon Region.

Different steps are necessary to make the training on the cluster possible. First, a *ssh* connection has to be made with the cluster

and a profile session has to be created. Then, all the files with the project have to be put on the cluster so that it has access to it and can be run. The most standard way is simply to clone the project using github. The next step is to transfer the dataset. Once done, the cluster has all it needs to make the code run and only needs to know what must be done.

The cluster works using *Slurm* (Simple Linux Utility for Resource Management). To make it work, a bash script has to be defined and run, indicating useful information such as the time and resources needed, where to store the output and what has to be run. *Slurm* uses this system to share the resources of the cluster fairly between all users, using criteria such as the time and resources requested. A submission script is available in Appendix C.

5.3 Evaluation procedure

To have a consistent measure, 50% of the slices before and after the organ to be predicted are included in the test set. The different datasets indeed have a very different number of slices due to some pre-processing that were made to them before submission. It is therefore useful to make the evaluation on a similar basis to have a consistent comparison between the datasets. Moreover, trying the algorithm far away from the organ does not really have sense. As an example, using a model to detect the heart on a scan of the leg doesn't have any interest.

5.4 Two steps training

As explained in subsection 3.2.1, the training set includes images with the organ but also images that do not contain it. The problem of this method is that any batch size, learning rate, loss function or ϵ (the smooth of the loss) was able to make it converge from an empty model. The large number of images without the organ makes the algorithm drops to a prediction containing only zeros and the organ is never predicted. To deal with this problem, the training is made in two separate steps.

In the first step, the dataset only contains the images including the heart. At this step, $\epsilon = 1$ since every slice already contains true pixels. At this point, a first model is then created, recognising the

organ well when it is present. The model however gives strange predictions on slices beyond or after the organ as it was not trained for it.

In a second time, a new model is created that loads the weights of the previous model. This is called transfer learning. 50% of the slices before and after the organ are added to the dataset for each patient. Every parameter remains the same, except for ϵ that becomes greater in order to make the derivative of the loss function smoother on the new slices. The new value for ϵ was optimised independently for each organ. After this second step, the prediction preforms well in both the images including and not including the organ and can therefore be used in a real life scenario.

5.5 Results for the heart

From the results of the heart, we made a conference paper that was accepted for publication and will be presented in the IVMSP 2022 (26 June 2022 - 29 June 2022) in Nafplio, Greece. The paper can be found in Appendix A. The results were obtained using hyperparameters that were optimised for this organ. The best results were found using a learning rate of $5e-5$ and a batch size of 1. ϵ, the smooth term, was found to be optimal with a value of 5000 in the second step of the training. This allows to be more derivable on the images without the heart. Finally, the optimal data augmentation is a rotation between $-25°$ and $25°$, a maximum zoom of 8% and a brightness change between -1.5% and 1.5%. The loss function used is the Dice's loss.

5.5.1 Datasets distribution

Table 5.1 summarises the distribution of the data available for the heart over the three datasets. All the patients for which the segmentation of the heart was available are used. It's important to mention that all images used during the second phase of the training are counted in this table. As 50% of the images before and after the organ is used, the number of slices containing the heart is approximately half the number of images written in the table.

The main particularity for the heart is that most of the patients of the first dataset (NSCLC Radiomics) do not include a segmentation

	NSCLC Radiomics	Pediatric CT-SEG	LCTSC
Total patients	127	349	59
Training patients	89	244	41
Training images	5162	22006	2936
Validation patients	25	70	12
Validation images	1489	5789	767
Test patients	13	35	6
Test images	815	3791	399
Percentage of training images	17,147%	73,100%	9,753%

Table 5.1: Distribution of the data containing the heart over the three datasets

of this organ. In this scenario, the second dataset (Pediatric CT-SEG) almost includes 75% of the images and is therefore much bigger than the two others. This scenario is interesting as it represents well a situation where a big hospital would communicate with smaller hospitals. Is that interesting for the small hospitals? And for the biggest one? The results for the heart will help answering these questions.

5.5.2 Comparison of the strategies

Dice 3D	local models	centralised model	fedAvg	equal-chances
NSCLC Radiomics	0.85830	**0.88445**	0.86664	0.87940
Pediatric CT-SEG	0.93879	**0.94487**	0.94348	0.94035
LCTSC	0.90089	0.90032	0.89054	**0.92227**

Table 5.2: 3D Dice's coefficient of the four strategies on the three test sets.

Equal-chances and local model

Table 5.2 summarises the results of all strategies on the three datasets. The most important conclusion is the following: Federated Equal-Chances leads to better results than the local for all of the three datasets. The smaller datasets have a major difference of over 2% of Dice's coefficient which is not negligible. Small hospitals therefore have a real interest to do Federated Learning with bigger hospitals. This improvement is not so surprising as the amount of data of these small datasets is significantly increased by doing so.

A most surprising result is the comparison of the score between these strategies on the biggest dataset, Pediatric CT-SEG. Indeed, Federated Equal-Chances improves the score in this dataset compared to the local model, despite the fact that only a small amount of data is added from this communication. Moreover, giving more credit to datasets 1 and 3 could have affected the results on the big dataset and lead to a decreasing quality over the local model, but it is not the case.

To the main question of the thesis: "Is it worth doing Federated Learning instead of keeping the local data for training?", the positive results for the heart answers with a yes.

Comparison of the federated strategies

In this thesis, *fedAvg* was modified in order to give more credit to the smaller datasets using data augmentation, as explained in subsection 4.4.1. From 5.2, we can compare how the two algorithms perform in practice to see if the idea is interesting for the future. As expected, datasets 1 and 3 (the smaller ones) perform better using this new approach. The difference is quite big as it is more than 1% for the first one and more than 3% for the third.

Interestingly, the smallest dataset, LCTSC, gives a lower score using the original fedAvg compared to the local model. This comes from the low capacity of a small dataset to update the weights of the model, as the updates are weighted with the amount of data. The internal iid behaviour of the dataset is then lost and leads to decreasing performances with this strategy. Unlike Federated Equal-Chances, *fedAvg* might therefore not be interesting in some cases for hospitals with a small amount of data.

The biggest dataset however gives better results using the original Federated Averaging algorithm than using the new version, which is quite intuitive regarding the fact that Federated Equal-Chances decreases the modification factor of this dataset to $\frac{1}{3}$. This difference is however much smaller (around 0.3%) than the improvements on the small datasets. Considering the fact that small hospitals might lead to decreasing performances using the original *fedAvg*, Federated Equal-Chances gives more interesting results on the heart.

Comparison with the centralised model

As expected, the classical model using centralised data gets the best performances. For the biggest dataset, the score obtained is the best among the four strategies, but the difference is very small with the original Federated Averaging, around 0.15%. On the first dataset, this algorithm also gives the best performances among them all.

Despite that, we cannot conclude that the centralised algorithm is strictly the best. Indeed, the score is more than 2% lower than Federated Equal-Chances on the smallest dataset. Moreover, the results of the centralised algorithm on this dataset are even slightly inferior to the ones of the local model, which means that the score can drop using data from other hospitals. In the end, only Federated Equal-Chances gives better results than the local models of every datasets. This is a very important conclusion as this might give interest to this strategy even when there are no privacy reasons to use it.

5.5.3 Visualisation of the results

Despite the efficiency of the Dice's coefficient for describing the quality of a prediction, what really matters are the predictions themselves. Visualising a prediction is important to understand what these numbers really mean and how good a model is. Figure 5.1 presents the different segmentations of a patient from the third dataset. The model used to make these predictions is the one that was trained using Federated Equal-Chances. Obviously, this patient comes from the test set and has therefore not been used during training.

In the first image (top left), the mask of the heart is shown as segmented by the doctor. This is therefore the reference to compute the Dice's coefficient. The slices that are provided in the scan are horizontal in this plot. The slices are therefore the 65 planes where z is fixed in this graph.

The second image (top right) shows the prediction of the first model, the one that was made using only images including the heart, as explained in section 5.4. At this point, the heart is very well localised and predicted when it is present on the slices where the prediction is made. However, on the slices before and after the heart, the model continues to predict something at the place where

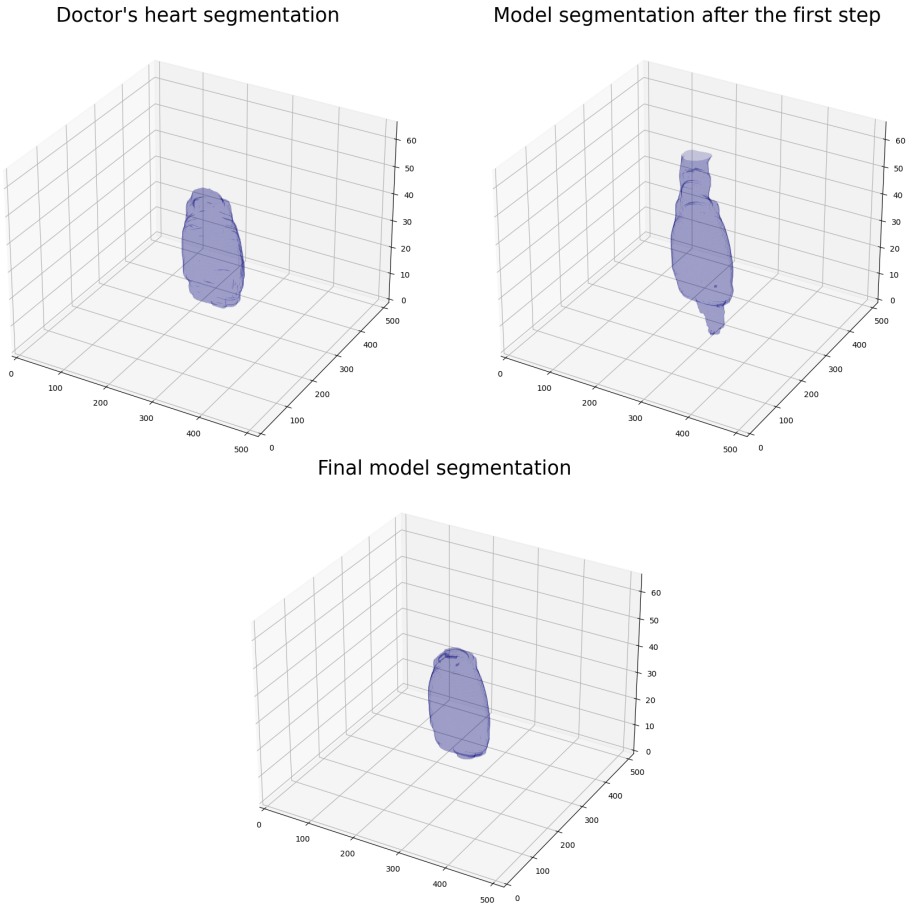


Figure 5.1

the heart was. This is quite logical as this first model was never trained on these kinds of images. The 3D Dice's coefficient for this prediction is 0.86600 at this step. Notice that false predictions above and below the heart are hollows, which is why the score is not so bad. If it was filled, the score would have been lower.

The last image (in the bottom) is the segmentation of the final model: the one where the other images were added. At this point, the false predictions disappeared and the prediction is very similar to the doctor's segmentation of the heart. The resulting Dice's coefficient is 0.93189, which is a standard score with this model on this dataset.

5.5.4 Robustness of the strategy

In the results, the first dataset systematically has a lower score than the other ones. This observation might seem surprising as it is not the smallest one and it has more than double of patients than the third one. The reason for this difference comes from the datasets directly, and more precisely from the doctor's segmentations.

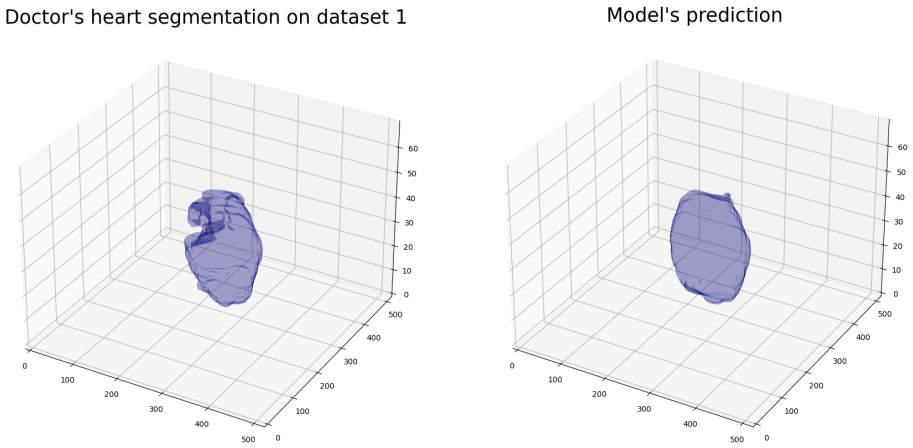


Figure 5.2: Comparison of the doctor's segmentation of the heart and the prediction of the local model for a patient of the first dataset.

Figure 5.2 shows the problem with this dataset. This segmentation results in a 3D Dice's coefficient of 0.85959, which is a quite neutral score for this dataset. In fact, the doctor's segmentations are made totally differently than the ones from datasets 2 and 3. As you can see on the left image, the segmentation includes the precise structure of the heart, leading to a more complex shape. The other datasets however, segment the heart more like a sphere and didn't waste time on the details. As datasets 2 and 3 include most of the data, the predictions respect these conventions, which results in a prediction that does not fit parts of the doctor's segmentation.

This difference is a good example of the inter-variability between the datasets. Despite leading to lower scores on this dataset, this difference is very interesting for the analysis. Indeed, this is a good way to evaluate the robustness of the strategies. As the federated strategies learn locally from the different datasets, this difference could have affected the model, resulting in lower scores using Federated Learning than using simply the local data. Dataset 2, for example, could have decreasing performances due to it, as it already

owned a good part of the data. However, in practice, the modified version of Federated Averaging led to better results for all three individual datasets. These results tend to prove that the strategy is robust and that even big differences between the data from one dataset to another do not affect the score too much, and that this algorithm remains interesting even in that case.

5.6 Results for the lungs

The best results for the lungs were found with a learning rate of $2e-5$, a batch size of 1. ϵ was found to be optimal with a value of 2000 in the second step of the training. The optimal data augmentation is a rotation between $-25°$ and $25°$, a maximum zoom of 8% and a brightness change between -1.5% and 1.5%. The loss function used is the Dice's loss.

5.6.1 Datasets distribution

As for the heart, Table 5.3 summarises the distribution of the data available for the lungs over the three datasets.

	NSCLC Radiomics	Pediatric CT-SEG	LCTSC
Total patients	312	345	59
Training patients	218	242	41
Training images	25328	36578	6481
Validation patients	62	69	12
Validation images	7456	9896	1798
Test patients	32	34	6
Test images	3571	6513	900
Percentage of training images	37,036%	53,487%	9,477 %

Table 5.3: Distribution of the data containing the lungs over the three datasets.

The configuration for the lungs is very different than for the heart. NSCLC Radiomics indeed has much more data for the lungs. Therefore, the scenario is not one big hospital with two small ones anymore but two big (Pediatric remaining the biggest) and one small.

5.6.2 Comparison of the strategies

Dice 3D	local models	centralised model	fedAvg	equal-chances
NSCLC Radiomics	**0.98590**	0.98174	0.97110	0.97614
Pediatric CT-SEG	**0.95879**	0.94843	0.95645	0.95205
LCTSC	**0.97323**	0.97057	0.97241	0.97410

Table 5.4: 3D Dice's coefficient of the four strategies on the three test sets.

Performances of the local models

In this configuration, the local models perform better than the other ones. A model built using local data works therefore better without external sources. Two main reasons can be used to explain these surprising results.

Figure 5.3 illustrates a crucial difference between the conventions used by the doctors for the segmentation. Indeed, the second dataset treats the lungs as a more complex shape than the others, resulting in a major inter-variability. This can also be observed on the results, where the second dataset systematically reaches lower results, because the segmentation is more complex and harder to be accurate.

The second main reason is the high quality of the local models due to the ease of the task compared to the segmentation of the heart. The amount of data inside a local dataset is indeed sufficient to have a very high score (97% Dice's score on the smallest dataset) and the additional images from the other datasets have therefore less interest. These two reasons combined lead to conclude that the extra images add more inter-variability in this case than the benefit of the number of images.

This illustrates an important aspect of the usability of Federated Learning. Federated Learning can be useful to improve the results locally in a hospital, but it can also lead to decreasing performances in such cases. To ensure an improvement in quality, a consensus in the procedure of the annotation of the images is a major aspect. However, as the centralised model using the traditional approach also has lower performances than the local models, this problem is not a flaw of Federated Learning, but simply a consequence of using data from sources with different conventions.

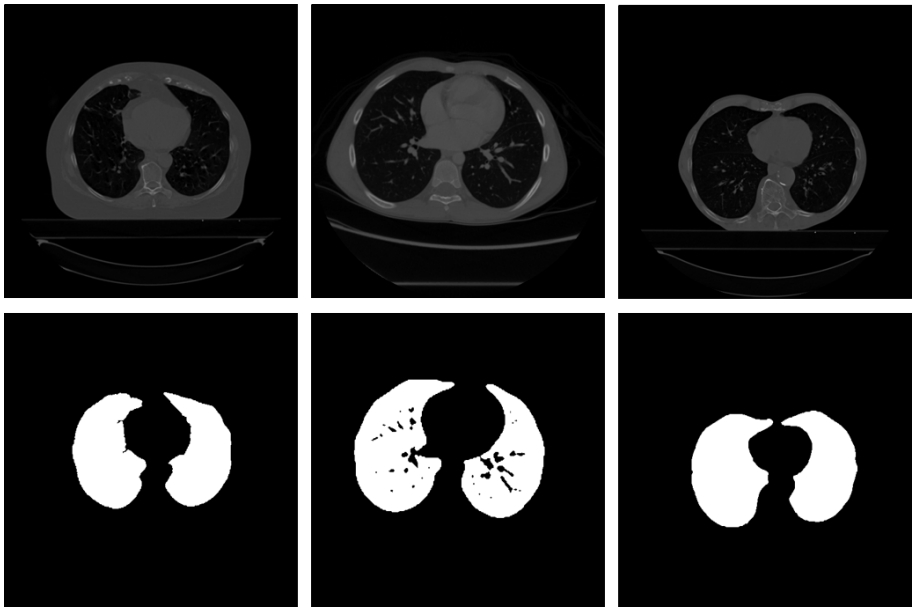


Figure 5.3: Comparison for each dataset of one slice and its corresponding mask, as segmented by the doctors. The first column is NSCLC Radiomics, the second is Pediatric CT SEG and the last one is LCTSC.

Comparison of the federated strategies

As for the heart, Federated Equal-Chances outperforms Federated Averaging on two of the three datasets. As expected, the results on the biggest dataset are slightly less efficient than using *fedAvg* because the weight of the updates made by this one is smaller. However, the two other datasets lead to better results, even for the first dataset which owns more than a third of the data.

5.6.3 Visualisation of the results

In order to visualise the quality of the segmentations produced for the lungs, Figure 5.4 shows the 3D segmentation of the lungs as made by doctors on the left, and the prediction found by the Federated Equal-Chances on the right. This prediction results in a score of 0.98185.

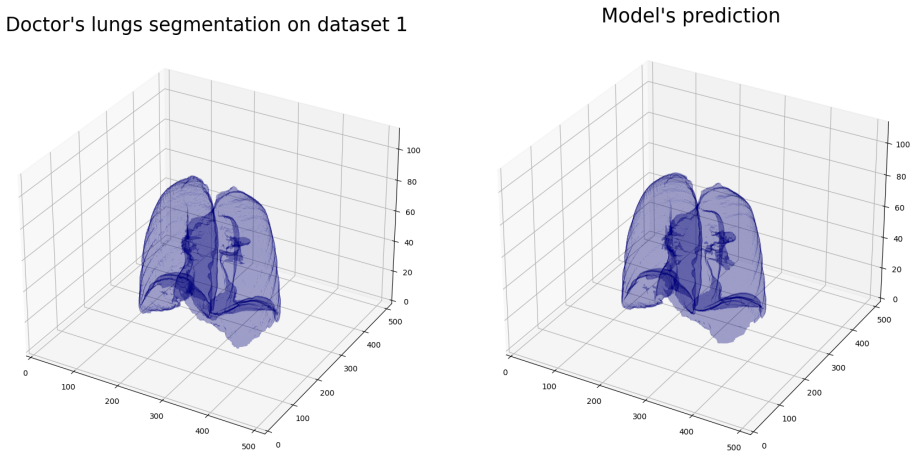


Figure 5.4: Comparison of the doctor's segmentation of the lungs and the prediction of the Federated Equal-Chances model for a patient of the first dataset.

5.7 Results for the oesophagus

5.7.1 Datasets distribution

Table 5.5 summarises the distribution of the data available for the oesophagus over the three datasets.

	NSCLC Radiomics	Pediatric CT-SEG	LCTSC
Total patients	354	347	59
Training patients	248	243	41
Training images	28559	35178	6427
Validation patients	71	69	12
Validation images	8687	9625	1754
Test patients	35	35	6
Test images	4263	6415	892
Percentage of training images	40,7%	50,1%	9,2 %

Table 5.5: Distribution of the data containing the oesophagus over the three datasets.

In this configuration, NSCLC Radiomics has more patients available than for the other organs and even more than Pediatric CT-SEG, which was the biggest in the two other cases. However, in terms of images, the second dataset remains the biggest with slightly more than half of the total training images.

5.7.2 Comparison of the strategies

Dice 3D	local models	centralised model	fedAvg	equal-chances
NSCLC Radiomics	0.72230	**0.73421**	0.71814	0.72953
Pediatric CT-SEG	0.73113	**0.75445**	0.72353	0.71668
LCTSC	0.68542	**0.77241**	0.70075	0.76369

Table 5.6: 3D Dice's coefficient of the four strategies on the three test sets.

Table 5.6 compares the scores of the four strategies on the oesophagus over the three test sets. As this table illustrates, the scores obtained for the oesophagus are inferior than the ones observed for the heart and the lungs. This is due to the fact that the organ is much smaller, which results in a harder segmentation. Moreover, the Dice formula gives a bigger penalty for incorrect pixels when the mask to predict is small.

Performances of the centralised model

For the oesophagus, the scores obtained using the centralised model are clearly higher than the ones for the local models. This is particularly notable for the smallest dataset where the score increases by almost 9%. This major improvement is due to the fact there is much less inter-variability possible for the oesophagus. Indeed, unlike the lungs and the heart which have complex shapes with nested structures and a lot of possible inter-variability between the segmentations, the oesophagus is more constant. Therefore, the inter-variability here is minor compared to the benefits of adding more images to the training set.

Federated Learning comparison

On the first and the third dataset, Federated Equal-Chances outperforms Federated Averaging. This is particularly visible on the smallest dataset, where the difference is higher than 6%. However, on Pediatric CT-SEG which is the dataset with the biggest number of images, the score of Federated Averaging is better. This observation was the same for all three organs. Federated Equal-Chances has a slightly inferior score on the biggest dataset but leads to better

results on the two other ones.

Compared to the local models, Federated Equal-Chances led to a major improvement of almost 8% on the smallest dataset, and also improved the results on the first one. Despite the lower score on the second dataset, Federated Equal-Chances is globally much better than the local models and is therefore a promising alternative to respect privacy. Moreover, the second dataset holds more than 50% of the training images. In practice, it is therefore not surprising that doing less than double the number of training images is not enough to improve performances when adding images from external sources. However, in a real-life implementation, every client should see its amount of data significantly increase and might probably behave more like the smallest dataset.

5.7.3 Visualisation of the results

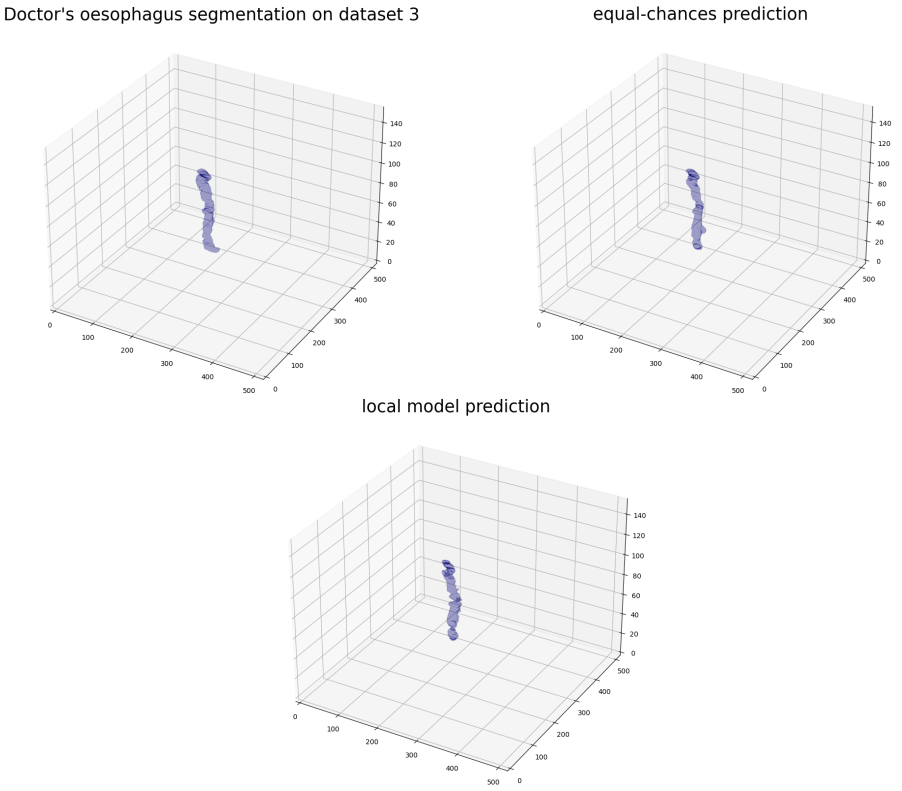


Figure 5.5: Comparison of the doctor's segmentation of the oesophagus, the prediction of the Federated Equal-Chances model and the prediction of the local model for a patient of the third dataset.

Figure 5.5 illustrates the difference in quality of a prediction of the Federated Equal-Chances model and the local model on a patient of the smallest dataset. The Dice scores of the segmentations are respectively 0.747 and 0.652. These segmentations are interesting as they both give a score close to the mean score of the model on the dataset. This is therefore representative of the improvement that Federated Equal-Chances provides compared to the local model on this dataset.

5.8 Computation times

Table 5.7 reports the time that is necessary to learn these different models on an Intel Xeon 6244 server equipped with a NVIDIA RTX 2080 Ti GPU.

	heart	lungs	oesophagus
Local NSCLS-Radiomics	0.8	12.6	6.25
Local Pediatric CT-SEG	5.3	23.7	11.4
Local LCTSC	0.5	11.2	4.5
Central database	9	39	17
Federated Averaging	18.3	66	28
Federated Equal-Chances	30.5	70	35.5

Table 5.7: Total computation times in hours of the different strategies used on the three organs.

It is however important to mention that the times needed using Federated Learning strategies are not representative of the computation times that would be necessary in a real framework. Indeed, in this local simulation, the clients train the central model one after the other. In practice, the computation made by the different hospitals are made in parallel, which should result in significantly faster training.

Chapter 6

Conclusion

The results obtained in the present work have crucial societal impacts because they might significantly improve automated medical images segmentation. This could lead to a major gain of time for doctors to do more important tasks and could result in an improvement of the treatment of important diseases such as cancer. Federated Learning also guarantees the security of protected medical data as they are kept local by design.

The goal of this thesis was to evaluate the potential of Federated Learning for organ segmentation. To this end, three datasets from different sources were used in order to simulate hospitals in a real-life scenario. Thanks to these datasets, U-Net models were trained using different strategies: local models using each only one dataset, a global model using the data centralised from these sources and models using Federated Learning strategies. These models were built for the heart, the lungs and the oesophagus. For all of them, the hyperparameters were optimised in order to get the highest possible Dice coefficient, which is a common metric used to evaluate the quality of a segmentation.

6.1 Main results

As experimental results showed that the smallest datasets might give low results using the classical Federated Averaging approach, a new variant of this algorithm referred to as Federated Equal-Chances was introduced to improve the segmentation performance by balancing the size of the datasets through data augmentation. In practice, this new approach outperforms Federated Averaging in most of the

cases, mostly for the small datasets.

From the promising results observed, we can conclude that Federated Learning might improve a lot the quality of images segmentations in the medical field. Indeed, it was proven that Federated Learning, and mostly the newly introduced Federated Equal-Chances, outperforms the local models made by one dataset in the vast majority of cases. Moreover, in a real-life scenario, the number of hospitals that might contribute to the process is much higher than the number of datasets that were used in this thesis. Therefore, a hospital might see its amount of data increase significantly more in real-life than in this simulation, resulting in potentially much better improvements than the ones observed here.

Despite indicating that Federated Learning by a coalition of hospitals is a viable alternative to the traditional centralised learning, the experimental results also showed that using different sources might decrease the performances in some cases, which is particularly notable for the lungs. On this organ, the results were better using the local models than using any other strategy, which proves that in this case, the problem does not come from Federated Learning. The reason comes in fact from the inconsistencies of the segmentation strategies of the different datasets, which lead to the conclusion that consistent segmentation is a primordial factor to make this strategy relevant in practice. Therefore, hospitals have to make sure to be using common segmentation conventions in their local data in order to make Federated Learning efficient in practice, and to avoid the risk of building a model with lower performances than using only their local data.

The results for the heart were used to make a conference paper that was accepted for publication and will be presented in the IVMSP 2022. This paper can be found in Appendix A.

6.2 Future work

From a software engineering point of view, it is planned to develop a framework to actually deploy Federated Learning inside clinical environments as a plugin to the Orthanc server [3]. This development will require to setup a network infrastructure in which the hospitals connect as clients to a central server that manages the global model.

In this thesis, a local simulation of Federated Learning has proven that this strategy is very promising and might take medical images segmentation to another level. However, implementing it in real life is not easy and many challenges have to be dealt carefully [33]. This section introduces some of the most important ones. All these perspectives are open thanks to the results presented in the present work.

6.2.1 Local updates reception

In the full local implementation, the clients that have to do local updates will always do their training successfully and their resulting weights will be used for the global update. In practice, this is not so simple. Due to various reasons including the network connection and the battery or the available data, some updates may be lost. Depending on the number of losses, the quality of the global model might be significantly affected leading to poor results.

Another challenge of network losses comes from the time needed to train the global model. Indeed, what is the optimal time for a global step? When a hospital does not give its local updates, how much time does the process has to wait before giving up and starting the next step? These questions have to be answered carefully and the algorithm has to deal with such situations in a way that it can keep an optimal global model in every circumstances.

6.2.2 Expensive communication

In this implementation of Federated Averaging, three different datasets were used and each of them were considered as one client. Considering the rarity of anonymised annotated medical datasets, three was already an interesting number to perform a small simulation on. However, as the goal of Federated Learning is to respect the privacy by design, the amount of possible data can be much higher. More importantly, the amount of possible clients can be huge.

The risk of having an important number of hospitals is that the communication network can become slower than the local computations. Intelligent communication protocols as the one described in

[34] might be important in the future in order to reduce the costs of communications of the updates through the server. However, as the number of hospitals around the world remains relatively small compared to other application cases of Federated Learning (i.e the number of smartphones around the world), this potential problem might not be the major concern of Federated Learning in the medical field. Moreover, unlike smartphones, hospitals largely feature good network connectivity.

6.2.3 Security

As the main reason for needing Federated Learning in the medical field is to respect the privacy of the patients, security has to be the major aspect of the protocol. Trojan horse, viruses, malware, spyware, phishing and worms are critical attacks that have to be dealt efficiently in the communication protocol.

Appendix A

Conference paper

Federated learning for heart segmentation

Thibaud Misonne
Louvain School of Engineering,
Louvain-la-Neuve, Belgium
Email: thibaud.misonne@student.uclouvain.be

Sébastien Jodogne
Institute of Information and Communication Technologies, Electronics and Applied Mathematics,
Louvain-la-Neuve, Belgium
Email: sebastien.jodogne@uclouvain.be

***Abstract*—Training deep learning models for medical imaging requires access to large volumes of sensitive patient data. To this end, the models are generally trained on centralized, de-identified databases that are hard to collect because of privacy requirements. Federated learning proposes an alternative approach, in which a coalition of hospitals collaboratively trains a central model without exchanging any clinical data. This paper explores the combination of federated learning with U-Net models, and applies it to the task of image segmentation of the heart. A variant of federated learning referred to as *federated equal-chances* that improves segmentation performance on unbalanced datasets is introduced as well.**

I. Introduction

Artificial intelligence is being increasingly applied to medical imaging. In particular, the task of the automated segmentation of organs or biological structures has attracted a lot of interest. The family of the U-Net deep learning models has proved to be highly successful on bioimaging segmentation tasks [1]. Radiotherapy offers a great opportunity to study such segmentation algorithms in the context of human radiology. Indeed, the oncologists working in any radiotherapy department all around the world have to delineate organs-at-risk and tumors so as to propose efficient, highly personalized treatments to their patients [2]. This results in radiotherapy departments gathering large volumes of labeled images that could be extremely useful to train artificial intelligence algorithms.

However, such labeled medical images are not easily accessible to researchers, because they correspond to clinical data that must be carefully protected, and that should not be communicated outside of the hospitals without patient's consent. As a consequence, even though free and open-source software are widely available to extract DICOM images from the Picture Archiving and Communication System (PACS) of the hospital, to de-identify them, then to send them to external artificial intelligence experts [3, 4], such clinical trials necessitate the setup of complex research protocols because they typically associate two distinct entities (one hospital and one university). Some large datasets have nevertheless been released as open data, notably thanks to the efforts of *The Cancer Imaging Archive* project [5], but this is a small fraction of all the data that is produced by hospitals on a daily basis.

An alternative approach would be to train deep learning models directly inside the hospitals, without having to communicate patient data to the outside world. In practice however, training a complex segmentation model requires a lot of annotated data, and the amount of data available inside one single hospital may not be sufficient to effectively train the model. Furthermore, few hospitals employ data scientists who are granted to access patient data in order to do in-house artificial intelligence research, which limits the applicability of this approach to a handful of university hospitals.

Federated learning might be a game changer to train deep learning models for healthcare [6]. In the typical federated learning setup, many mobile devices collaborate to train a shared model in a decentralized way, without ever communicating personal data to protect privacy [7]. Federated learning has originally been motivated by supervised learning applications characterized by four key properties: non-i.i.d. (non independent and identically distributed) data [8], unbalanced data, massively distributed data, and limited communications.

Federated learning allows to envision the training of a central deep learning model for medical applications by a coalition of several hospitals and research teams, while avoiding any communication of patient data between these entities. Figure 1 transposes the typical setup of federated learning to the context of medical imaging [9–11]. The imaging data tends to be non-i.i.d. between hospitals because of different clinical teams and possibly different medical guidelines. Note that even within one single hospital, data related to image segmentation tasks might possibly be non-i.i.d. because of intra-observer and inter-observer variability [12]. Furthermore, datasets can be strongly unbalanced between the hospitals of the coalition, typically if smaller hospitals collaborate with larger hospitals. Summarizing, similarly to the original federated learning setup, the non-i.i.d. and unbalanced key properties of the data hold. However, the size of a healthcare coalition is much smaller than a coalition involving mobile devices. In addition, most hospitals benefit from good network connectivity. This implies that the two last key properties of original federated learning (i.e. massively distributed data and limited communications) are not necessarily fulfilled, which hopefully implies that learning is facilitated.

According to this discussion, this paper studies the combination of U-Net architectures with federated learning, and applies it to the segmentation of the heart on CT-scan images. Experimental results compare the predictions of the classical, centralized learning strategy with a federated learning strategy.

II. Federated learning for U-Nets

In this section, it is explained how U-Nets can be used in conjunction with federated learning.

A. U-Net architecture

U-Net is a highly successful deep learning architecture for bioimaging segmentation [1]. Contrarily to the sliding-window approach, in which all the possible patches of a

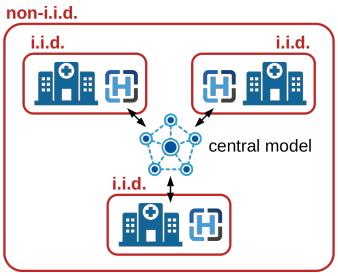


Fig. 1: Federated learning in the context of medical imaging. Several hospitals contribute to the training of one single deep learning model. No patient data leaves any hospital at any time: Only the parameters of the central model are exchanged.

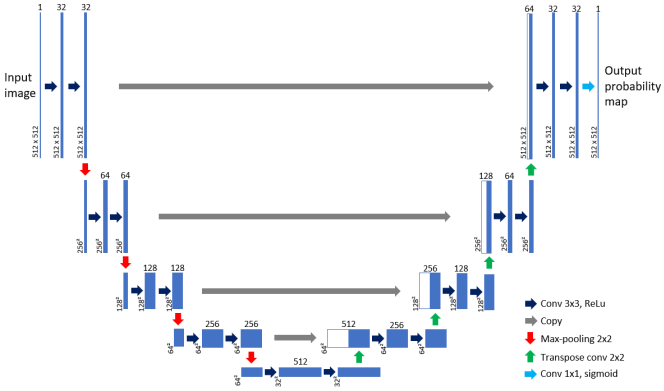


Fig. 2: The U-Net architecture that is used throughout this work. The main differences with respect to the original U-Net architecture [1] are the size of the input and output images (both 512×512 pixels), the number of filters used for the convolutions (it ranges from 32 to 512 filters), and the zero-padding that is used in convolutions (which preserves the image sizes throughout the network). The resulting U-Net models have exactly 7,759,521 floating-point parameters.

fixed size in an image are scanned by a convolutional neural network producing one binary output for each patch [13], U-Nets consider the input image as a whole. U-Nets first apply a contraction path to generate high-level features, then apply an expansion path to output the segmentation mask, resulting in the typical "U" shape that is depicted in Figure 2. This architecture allows the global spatial structure of the medical images to be taken into account. Generating a segmentation mask given an input image is also much faster than using the sliding-window approach.

In this work, the U-Net models are trained on the individual axial slices of segmented CT-scan images so as to minimize a smoothed version of the 2D Dice loss function [14], that is defined as:

$$\mathcal{L}(\hat{Y}, Y) = -\frac{2\left(\sum_{(x,y)} \hat{Y}(x,y) Y(x,y)\right) + \epsilon}{\left(\sum_{(x,y)} \hat{Y}(x,y)\right) + \left(\sum_{(x,y)} Y(x,y)\right) + \epsilon},$$

where $Y(x,y) \in \{0,1\}$ (resp. $\hat{Y}(x,y)$) denotes the expected binary value (resp. the actual prediction of the U-Net) of the pixel at coordinates (x,y), and where ϵ is a coefficient that prevents divisions by zeros as well as vanishing gradients induced by the slices that do not contain any pixel belonging to the segmentation mask.

B. Federated averaging

Federated averaging is a well-known, effective algorithm for federated learning [7]. In this algorithm, the weights of the central deep learning model are collaboratively optimized by a swarm of clients, each of them using their own private dataset.

Federated averaging is divided in a number of rounds. At each round t, federated averaging randomly selects a subset of K clients, sends them the current parameters w_t of the central model, asks the selected K clients to optimize the parameters on their own local dataset during a fixed number of epochs, then collects back the parameters $w_{t+1}^{(k)}$ as independently optimized by each client k. The parameters of the central model are then updated by taking the weighted average of the locally optimized parameters as follows:

$$w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^{(k)}, \tag{1}$$

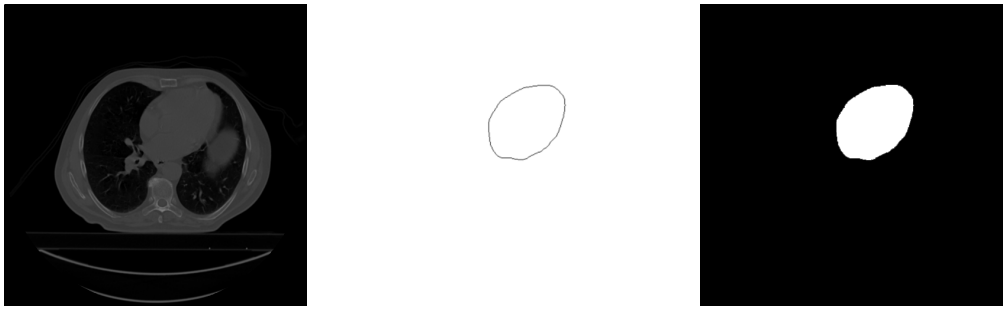
where n_k is the size of the local dataset of client k, and where $n = \sum_{k=1}^{K} n_k$ is the sum of the sizes of all the local datasets of the clients selected at round t.

In this work, the clients correspond to the hospitals. Because a typical learning coalition for healthcare will only contain a handful of hospitals, there is no random selection of the clients: All the hospitals optimize the model at each round t. Furthermore, this paper applies federated averaging to U-Nets: The weights w_t thus correspond to the 7 millions of floating-point parameters of our U-Nets. Because each parameter is encoded as a 4-bytes floating-point number in our setup, exchanging one full U-Net model through a computer network requires around 30MB of data. This is not an issue in practice because hospitals largely feature good network connectivity, and because the time that is needed for one round of U-Net optimization is typically much higher than for the transmission of data.

C. Equal chances through data augmentation

The classical federated averaging algorithm described in Section II-B can be problematic if the local datasets of the different hospitals are significantly unbalanced. Indeed, Equation 1 gives a much stronger importance to the model updates that stem from the larger datasets, hereby affecting the quality of the final model when tested on the smaller datasets.

As a consequence, we propose to leverage *data augmentation* so that every client in the coalition uses a training set of equal size. Data augmentation is commonly used to prevent overfitting of deep learning models in computer vision [15]: It consists in artificially increasing the size of a dataset using label-preserving transformations, such as rotations, flips or zooms. Let m_k denote the size of the original local dataset of the client k, and let M be the size of the largest dataset. The *federated equal-chances* algorithm consists, for each client k at each round t, in replacing the m_k original items in the



(a) (b) (c)

Fig. 3: Example of one typical entry in the datasets: (a) the 2D slice from the CT-scan image, (b) the contour of the heart in this slice, as delineated by the physician and encoded as a sequence of points defining a closed polygon, and (c) the binary segmentation mask that is obtained by filling up the polygon using a standard scan-line algorithm.

dataset, by $n_k = M$ new items obtained through online data augmentation. In this flavor of federated averaging, $n = KM$ and Equation 1 becomes an unweighted average of the updates.

III. Experimental results

The federated averaging and federated equal-chances algorithms are now applied to the task of heart segmentation on CT-scan images using U-Nets.

A. Datasets

Our experiments emulate an environment similar to the one depicted in Figure 1. The three hospitals in the figure are replaced by three unrelated datasets provided by *The Cancer Imaging Archive* project [5]. The three considered datasets are NSCLC-Radiomics [16, 17], Pediatric-CT-SEG [18, 19] and Lung CT Segmentation Challenge 2017 (LCTSC) [20, 21]. They provide DICOM CT-scan images together with the delineation of the heart encoded as DICOM RT-STRUCT instances, as routinely used in radiotherapy and nuclear medicine departments. Figure 3 illustrates a typical entry in these datasets.

Because these three datasets have been gathered and delineated separately by different research teams for different purposes, they provide non-i.i.d. distributions by design. Furthermore, they have varying sizes: NSCLC-Radiomics, Pediatric-CT-SEG and LCTSC respectively contain 127, 349 and 59 patients with a valid segmentation of the heart. These datasets are thus representative of a federated learning setup with non-i.i.d. and unbalanced data. Also, the three considered datasets contain images of whole chests, many axial slices of which do not include the heart. The datasets have thus been pre-processed in order to remove the axial slices that are farther than 50% the height of the heart, with respect to the top and bottom slices of the heart. The pre-processed datasets therefore contain the same number of slices including the heart than slices not including it.

Each dataset is randomly split into three independent parts: a train, a validation and a test set, containing respectively 70%, 20% and 10% of the patients. The validation set is used for the early stopping of the training to avoid overfitting. Importantly, the train/validation/test sets are defined at the patient level, not at the slice level. Working at the slice level would totally bias the results, as this would induce a strong correlation between the train/validation/test sets.

B. Learning parameters and evaluation metrics

One single 3D segmentation of the heart defines multiple binary segmentation masks, one for each 2D axial slice of the CT-scan image: The U-Net is trained to map those individual 2D axial slices to their associated binary segmentation masks. However, the performance metrics related to organ segmentation should consider the heart as a whole in the 3D space. This motivates the introduction of a 3D version of Dice similarity coefficient defined as follows [22]:

$$\mathrm{DSC}_{3\mathrm{D}}(\hat{Y}, Y) = \frac{2\left(\sum\limits_{(x,y,z)} \hat{Y}(x,y,z) Y(x,y,z)\right)}{\left(\sum\limits_{(x,y,z)} \hat{Y}(x,y,z)\right) + \left(\sum\limits_{(x,y,z)} Y(x,y,z)\right)},$$

where $Y(x,y,z) \in \{0,1\}$ (resp. $\hat{Y}(x,y,z)$) denotes the expected binary value (resp. the prediction of the U-Net) of the voxel (x,y) in the axial slice whose coordinate in the 3D stack is z. This 3D performance metrics was used to assess the final quality of the models.

The source code was written in Python using Keras with the TensorFlow back-end. The Adam optimizer was used, with random initial weights, a learning rate of $5 \cdot 10^{-5}$ and a batch size of 1. Note that this batch size of 1 corresponds to the value that was used in the original U-Net paper [1]. Dropout layers were introduced to prevent overfitting [23]. Online data augmentation was applied through Keras data generators, which means that the augmented images were different at each epoch. The following transformations were experimentally found to provide best results for data augmentation: rotations between $-25°$ and $25°$, zoom-in and zoom-out up to 8%, and slight modifications of the Hounsfield values of the CT-scan between -1.5% and 1.5%. It was found that using these parameters led to an improvement of about 1% in the Dice loss function if training the central model.

The learning was divided in two phases: Firstly, the U-Nets were trained only on the slices that contain a part of the heart up to convergence, with $\epsilon = 1$ in the loss function.

TABLE I: Value of DSC_{3D} for the different strategies

	NSCLC *127 patients*	Pediatric-CT-SEG *349 patients*	LCTSC *59 patients*
Centralized learning	*0.884*	*0.945*	*0.900*
Local dataset only	0.858	0.939	0.901
Federated averaging	0.867	**0.943**	0.891
Federated equal-chances	**0.879**	0.940	**0.922**

TABLE II: Approximate learning time

Centralized learning (535 patients)	9 *hours*
Local NSCLC-Radiomics *(127 patients)*	0.8 hour
Local Pediatric-CT-SEG *(349 patients)*	5.3 hours
Local LCTSC *(59 patients)*	0.5 hour
Federated averaging (accumulated time)	18.3 hours
Federated equal-chances (accumulated time)	30.5 hours

Secondly, the U-Nets were refined by adding the slices that do not contain the heart, with $\epsilon = 5000$. Without this two-phases approach, the learning process tends to diverge.

C. Results and discussion

U-Net models were trained using the four strategies presented in this paper. The obtained values for the 3D Dice similarity coefficient are reported in Table I. In this table, the centralized learning is the traditional, reference scenario in which the training is done on one central database that gathers the three datasets. Table II reports the time that is necessary to learn these different models on an Intel Xeon 6244 server equipped with a NVIDIA RTX 2080 Ti GPU.

The first interesting result is that the federated equal-chances algorithm systematically provides better scores than if the model was trained only on one of the three considered datasets. In the real world, this confirms the intuition that a hospital with a limited dataset should consider joining a coalition of hospitals. Table I also indicates that the federated equal-chances algorithm obtains better results than the standard federated averaging algorithm on the smaller datasets, while not being significantly detrimental to the larger dataset.

As expected, centralized learning provides slightly better scores in general, except on the LCTSC dataset, while having an accumulated runtime that is lower than the federated learning algorithms. However, as explained in the Introduction, centralized learning comes with the difficulty of gathering de-identified data outside of hospitals, which may or may not be a viable option depending on the real-world scenario. Furthermore, the reported runtimes for federated learning are accumulated, as though the trainings were done sequentially by the three hospitals. In practice, these trainings would be done concurrently, which would divide the total runtime.

Finally, it is interesting to note that NSCLC-Radiomics has a lower score than the two other datasets for all the strategies. The explanation is provided in Figure 4: It appears that there is an inter-observer variability were the contours of the heart have not be delineated using the same conventions throughout the dataset, which results in lower score on the LCTSC dataset in Table I. This however does not prevent the U-Nets from converging and providing valuable segmentations.

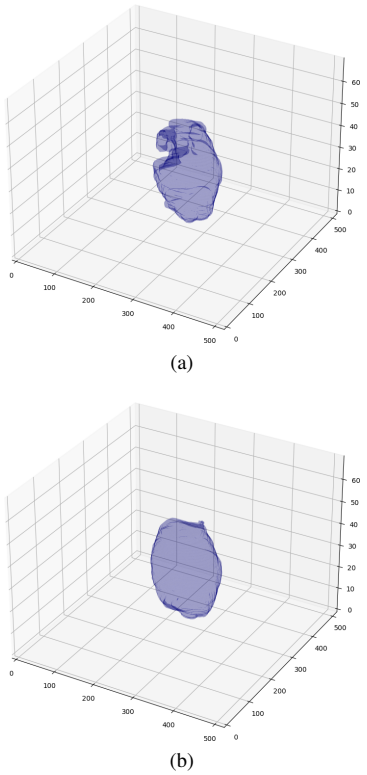


Fig. 4: 3D rendering of the segmentation of the heart in one DICOM study taken from the NSCLC-Radiomics dataset that illustrates the non-i.i.d. problem: (a) contour delineated by the physician, (b) contour segmented by the U-Net model. The difference in the results is explained by the fact that this particular physician has not followed the same conventions for delineating the heart than other physicians, including many more details in the contours. This is an example of inter-observer variability [12].

IV. Conclusion

In this paper, the combination of federated learning with U-Net models was applied to the task of heart segmentation in CT-scan images. A new variant of the federated averaging algorithm referred to as *federated equal-chances* was introduced to improve the segmentation performance by balancing the size of the datasets through data augmentation. Experimental results indicate that federated learning by a coalition of hospitals is a viable alternative to the traditional centralized learning on a global database gathering patient data from different hospitals.

Future work will apply federated learning to other bioimaging tasks, notably the segmentation of the lungs and of the esophagus, and to other deep learning models than U-Nets [24]. From a software engineering point of view, it is planned to develop a framework to actually deploy federated learning inside clinical environments as a plugin to the Orthanc server [4]. This development will require to setup a network infrastructure in which the hospitals connect as clients to a central server that manages the global model, which necessitates a focus on the security of the communications, on data management, and on the orchestration of data augmentation [25].

ACKNOWLEDGMENTS

Computational resources have been provided by the supercomputing facilities of the *Université catholique de Louvain* (CISM/UCL) and the *Consortium des Équipements de Calcul Intensif en Fédération Wallonie Bruxelles* (CÉCI) funded by the *Fonds de la Recherche Scientifique de Belgique* (F.R.S.-FNRS) under convention 2.5020.11 and by the Walloon Region. The authors also wish to thank Eliott Brion and Jean Léger for their helpful insights regarding U-Net architectures.

REFERENCES

[1] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Springer International Publishing, 2015, pp. 234–241.

[2] C. Sawyers, "Targeted cancer therapy," *Nature*, vol. 432, no. 7015, pp. 294–297, Nov. 2004. [Online]. Available: https://doi.org/10.1038/nature03095

[3] National Electrical Manufacturers Association, "ISO 12052 / NEMA PS3, Digital Imaging and Communications in Medicine (DICOM) standard," Rosslyn, VA, USA, 2022. [Online]. Available: http://www.dicomstandard.org/

[4] S. Jodogne, "The Orthanc ecosystem for medical imaging," *Journal of Digital Imaging*, vol. 31, no. 3, pp. 341–352, 2018. [Online]. Available: http://hdl.handle.net/2078.1/257255

[5] K. Clark, B. Vendt, K. Smith, J. Freymann, J. Kirby, P. Koppel, S. Moore, S. Phillips, D. Maffitt, M. Pringle, L. Tarbox, and F. Prior, "The Cancer Imaging Archive (TCIA): Maintaining and operating a public information repository," *Journal of Digital Imaging*, vol. 26, no. 6, pp. 1045–1057, Jul. 2013. [Online]. Available: https://doi.org/10.1007/s10278-013-9622-7

[6] N. Rieke, J. Hancox, W. Li, F. Milletarì, H. R. Roth, S. Albarqouni, S. Bakas, M. N. Galtier, B. A. Landman, K. Maier-Hein, S. Ourselin, M. Sheller, R. M. Summers, A. Trask, D. Xu, M. Baust, and M. J. Cardoso, "The future of digital health with federated learning," *npj Digital Medicine*, vol. 3, no. 1, Sep. 2020. [Online]. Available: https://doi.org/10.1038/s41746-020-00323-1

[7] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, A. Singh and J. Zhu, Eds., vol. 54. PMLR, 20–22 Apr 2017, pp. 1273–1282. [Online]. Available: https://proceedings.mlr.press/v54/mcmahan17a.html

[8] H. Zhu, J. Xu, S. Liu, and Y. Jin, "Federated learning on non-IID data: A survey," *Neurocomputing*, vol. 465, pp. 371–390, Nov. 2021. [Online]. Available: https://doi.org/10.1016/j.neucom.2021.07.098

[9] M. J. Sheller, G. A. Reina, B. Edwards, J. Martin, and S. Bakas, "Multi-institutional deep learning modeling without sharing patient data: A feasibility study on brain tumor segmentation," in *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*. Springer International Publishing, 2019, pp. 92–104. [Online]. Available: https://doi.org/10.1007/978-3-030-11723-8_9

[10] A. G. Roy, S. Siddiqui, S. Pölsterl, N. Navab, and C. Wachinger, "BrainTorrent: A peer-to-peer environment for decentralized federated learning," *arXiv preprint arXiv:1905.06731*, 2019.

[11] X. Li, Y. Gu, N. Dvornek, L. H. Staib, P. Ventola, and J. S. Duncan, "Multi-site fMRI analysis using privacy-preserving federated learning and domain adaptation: ABIDE results," *Medical Image Analysis*, vol. 65, p. 101765, Oct. 2020. [Online]. Available: https://doi.org/10.1016/j.media.2020.101765

[12] C. Fiorino, M. Reni, A. Bolognesi, G. M. Cattaneo, and R. Calandrino, "Intra- and inter-observer variability in contouring prostate and seminal vesicles: Implications for conformal treatment planning," *Radiotherapy and Oncology*, vol. 47, no. 3, pp. 285–292, Jun. 1998. [Online]. Available: https://doi.org/10.1016/s0167-8140(98)00021-8

[13] D. Ciresan, A. Giusti, L. Gambardella, and J. Schmidhuber, "Deep neural networks segment neuronal membranes in electron microscopy images," in *Advances in Neural Information Processing Systems*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012. [Online]. Available: http://papers.nips.cc/paper/by-source-2012-1292

[14] C. H. Sudre, W. Li, T. Vercauteren, S. Ourselin, and M. J. Cardoso, "Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations," in *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*. Springer International Publishing, 2017, pp. 240–248. [Online]. Available: https://doi.org/10.1007/978-3-319-67558-9_28

[15] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, Jul. 2019. [Online]. Available: https://doi.org/10.1186/s40537-019-0197-0

[16] H. J. W. L. Aerts, E. R. Velazquez, R. T. H. Leijenaar, C. Parmar, P. Grossmann, S. Carvalho, J. Bussink, R. Monshouwer, B. Haibe-Kains, D. Rietveld, F. Hoebers, M. M. Rietbergen, C. R. Leemans, A. Dekker, J. Quackenbush, R. J. Gillies, and P. Lambin, "Decoding tumour phenotype by noninvasive imaging using a quantitative radiomics approach," *Nature Communications*, vol. 5, no. 1, Jun. 2014. [Online]. Available: https://doi.org/10.1038/ncomms5006

[17] H. J. W. L. Aerts, L. Wee, E. Rios Velazquez, R. T. H. Leijenaar, C. Parmar, P. Grossmann, S. Carvalho, J. Bussink, R. Monshouwer, B. Haibe-Kains, D. Rietveld, F. Hoebers, M. M. Rietbergen, C. R. Leemans, A. Dekker, J. Quackenbush, R. J. Gillies, and P. Lambin, "Data from NSCLC-Radiomics," 2019. [Online]. Available: https://wiki.cancerimagingarchive.net/x/FgL1

[18] P. Jordan, P. M. Adamson, V. Bhattbhatt, S. Beriwal, S. Shen, O. Radermecker, S. Bose, L. S. Strain, M. Offe, D. Fraley, S. Principi, D. H. Ye, A. S. Wang, J. Heteren, N.-J. Vo, and T. G. Schmidt, "Pediatric chest-abdomen-pelvis and abdomen-pelvis CT images with expert organ contours," *Medical Physics*, Feb. 2022. [Online]. Available: https://doi.org/10.1002/mp.15485

[19] P. Jordan, P. M. Adamson, V. Bhattbhatt, S. Beriwal, S. Shen, O. Radermecker, S. Bose, L. S. Strain, M. Offe, D. Fraley, S. Principi, D. H. Ye, A. S. Wang, J. Van Heteren, N.-J. Vo, and T. G. Schmidt, "Pediatric chest/abdomen/pelvic CT exams with expert organ contours (Pediatric-CT-SEG)," 2021. [Online]. Available: https://wiki.cancerimagingarchive.net/x/jIFPBQ

[20] J. Yang, H. Veeraraghavan, S. G. Armato, K. Farahani, J. S. Kirby, J. Kalpathy-Kramer, W. van Elmpt, A. Dekker, X. Han, X. Feng, P. Aljabar, B. Oliveira, B. van der Heyden, L. Zamdborg, D. Lam, M. Gooding, and G. C. Sharp, "Autosegmentation for thoracic radiation treatment planning: A grand challenge at AAPM 2017," *Medical Physics*, vol. 45, no. 10, pp. 4568–4581, Sep. 2018. [Online]. Available: https://doi.org/10.1002/mp.13141

[21] J. Yang, G. Sharp, H. Veeraraghavan, W. Van Elmpt, A. Dekker, T. Lustberg, and M. Gooding, "Data from lung CT segmentation challenge," 2017. [Online]. Available: https://wiki.cancerimagingarchive.net/x/e41yAQ

[22] E. Brion, J. Léger, A. Barragán-Montero, N. Meert, J. A. Lee, and B. Macq, "Domain adversarial networks and intensity-based data augmentation for male pelvic organ segmentation in cone beam CT," *Computers in Biology and Medicine*, vol. 131, p. 104269, Apr. 2021. [Online]. Available: https://doi.org/10.1016/j.compbiomed.2021.104269

[23] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[24] L. Liu, J. Cheng, Q. Quan, F.-X. Wu, Y.-P. Wang, and J. Wang, "A survey on U-shaped networks in medical image segmentations," *Neurocomputing*, vol. 409, pp. 244–258, Oct. 2020. [Online]. Available: https://doi.org/10.1016/j.neucom.2020.05.070

[25] P. Singh, M. K. Singh, R. Singh, and N. Singh, "Federated learning: Challenges, methods, and future directions," in *Federated Learning for IoT Applications*. Springer International Publishing, 2022, pp. 199–214. [Online]. Available: https://doi.org/10.1007/978-3-030-85559-8_13

Appendix B

Recent evolutions of U-Net

For this thesis, sliding-window and U-net were implemented to perform the segmentation. However, other techniques exist and could be used in a future work as a possible improvement. Some of the most important are detailed in this section [35].

B.0.1 3D U-net

3D U-net [36] is an extension of U-Net to three dimensions. The architecture can be seen on Figure B.1. The principle and the architecture are identical, except that it uses an additional dimension. Theoretically, it would be possible to use all slices of the scans to reconstruct the full 3D images and use them for training and evaluation. Doing so might increase a lot the performances as the z coordinate would be taken into account. However, doing so is not so simple in practice.

The first problem comes from the available memory of a GPU. It is indeed not possible to load all of the slices on most GPU since loading all of them would require too much memory. This comes from the big size of the images (512x512) and the obligation of using 16 bits to store their value as the range needs more than 256 intensities. An alternative would be to use patches instead of the full image, but this method would lose some of its interest as it would lose the knowledge of the z coordinate.

Another drawback of loading all of the slices to perform a 3D U-Net on the full scan comes from the size of the datasets. Indeed, in term of slices, the three datasets combine more than 20000 slices available containing the heart. However, when it comes to the full scan, this number drops to 535 as many slices come from the same

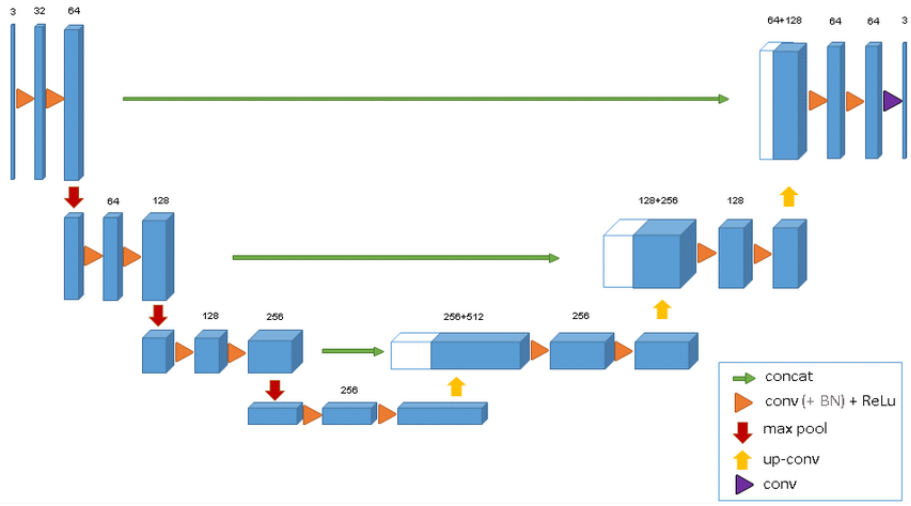


Figure B.1: Architecture of 3D Unet. Illustration from [36].

patients. As some of them have to be kept for the evaluation, only about 400 of them would be used for training which is much smaller and would not make sure to have a better performance than the classical U-Net. Moreover, considering that the smaller dataset only has 59 valid scans, the local updates of this dataset would have a very small number of data, which could bias it.

In practice, both strategies lead to comparable results on the bladder, the rectum and the prostate [37]. 3D U-Net is therefore not perfect but could however have advantages compared to the simple version. This alternative is interesting and might be interesting for future work.

B.0.2 V-Net

The architecture of V-Net [38], which is shown in Figure B.2, is inspired from the 3D version of U-Net. As for U-Net, the left part of the network is a compression path and the right side decompresses the signal to recover the original size. There are however differences compared to U-Net, both in the left and the right side.

First, in the left part, the number of convolutions is not fixed to two at each step but vary from one to three. The convolutions use volumetric kernels of size 5x5x5 voxels. The resolution is reduced in the contracting path using 2x2x2 voxels wide kernels with stride 2.

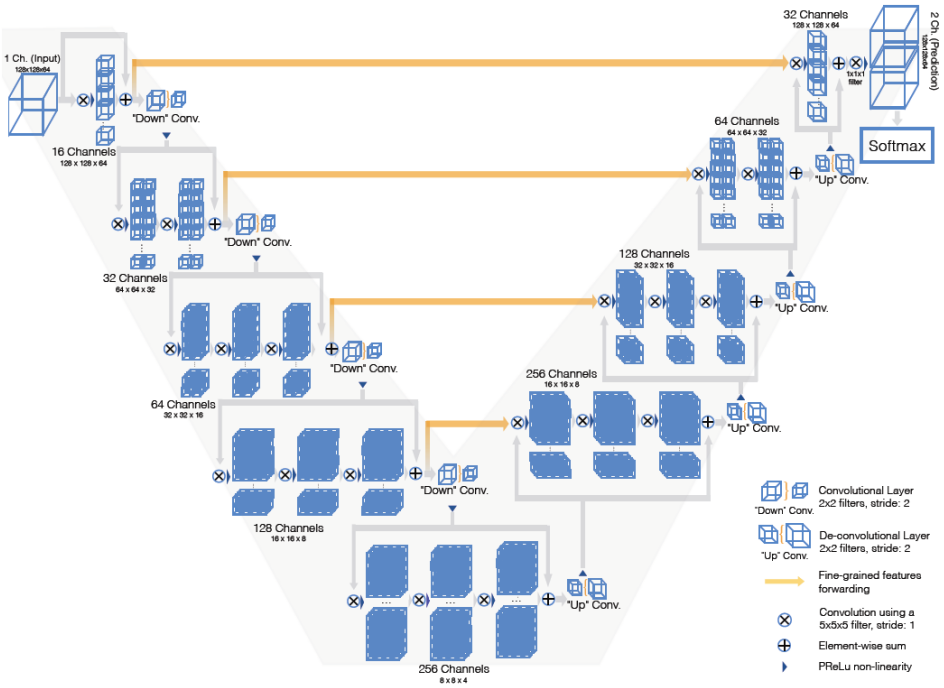


Figure B.2: V-Net architecture. Illustration from [38].

This replaces the max-pooling operators of the original U-Net that had the same utility of dividing the size of the image by a factor 2. This new strategy reduces the memory-footprint when training the model. Also, the input of the stages are used in the convolutional layers. They are then processed through the non-linearities and added to the output of the last convolutional layer of that stage. This way, the architecture allows to learn a residual function at each stage and the convergence is ensured much faster. The non-linearity activation function used is PReLU (Parametric Rectified Linear Unit).

The differences of the expanding path are similar to the ones of the compression path. After the deconvolution steps, one to three 5x5x5 kernels are applied and residual function are learnt. In the last layer, two features maps are computed using a 1x1x1 kernel. By applying soft-max voxelwise, these two outputs are the probabilistic segmentations of the foreground and the background regions.

Despite the potential of V-Net, U-Net remains the state of the art today and there is not enough hindsight on V-Net yet. This might however be an interesting path to explore for a future work.

B.0.3 MultiResUNet

MultiResUNet [39] is another attempt of improvement of the original U-Net architecture. The main idea of the architecture comes from the possible variations of scale of the objects of interest. To deal with such a problem, the idea is to make a multi-resolutional analysis by using special MultiRes blocks instead of the classical series of two 3x3 convolutions of U-Net. After developing an efficient block and trying to reduce the memory consumption as much as possible, the final MultiRes block is illustrated on Figure B.3.

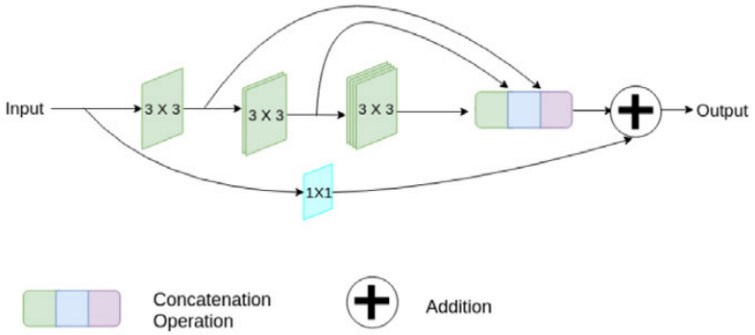


Figure B.3: MultiRes block replacing the two 3x3 convolutions of the original U-Net. Illustration from [39].

Another improvable component of U-Net are the skip connections between the contracting and the expanding side. The speculation is that a semantic gap might appear between the encoder and the decoder. As an example, think about the very first skip connection of U-Net (the one just after the two first convolutions). At this point, the features from the encoder are of low level as they are computed in the earliest layers of the network. On the opposite, the corresponding layers of the decoder are high level features. They conjecture that this semantic gap could cause some discrepancy that might affect the learning procedure.

To deal with this potential problem, they introduced a residual path 'Res path', to replace the original shortcut connections of U-Net. The idea is to add convolutions into the path itself in order to make the encoder's features map of higher levels. As the difference is bigger for the first connections, each Res path is different. Respectively, they use 4, 3, 2 and 1 convolution blocks along the Res paths. The Res paths also use different number of filters in their blocks: 32, 64, 128 and 256 respectively. The first Res path is

illustrated on Figure B.4.

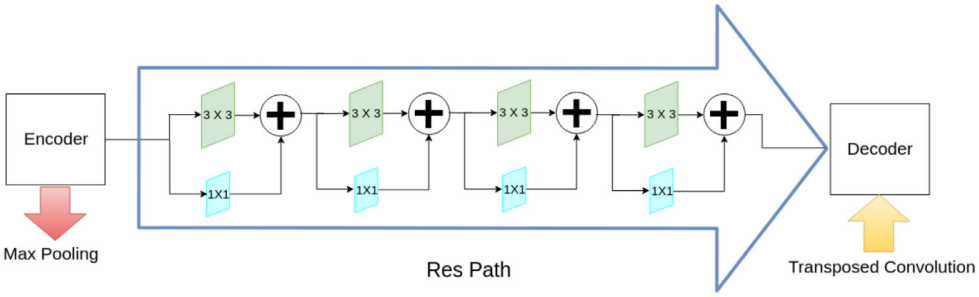


Figure B.4: Res path replacing skip connections of the original U-Net. Illustration from [39].

Putting everything together, the MutiResUNet architecture is shown in Figure B.5.

MultiResUnet introduces interesting ideas that might bring the U-Net architecture to a new level. However, the approach is quite recent and part of it rely on intuition and improvisation to reduce the high memory consumption. In the future, this type of architecture might replace the current U-Net but it is too early to know because of the lack of hindsight, which is why the classical U-Net was more interesting for this thesis.

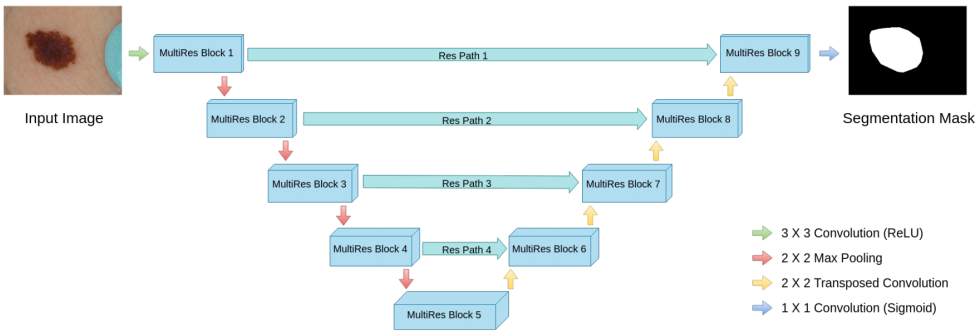


Figure B.5: MultiResUNet architecture. Illustration from [39].

Appendix C

Submission script on the cluster

```
#!/bin/sh
#
#SBATCH --job-name=lr1e6
#SBATCH --output="lr1e6.txt"
#
#SBATCH -w, --nodelist=mb-cas001
#SBATCH --ntasks=1
#SBATCH --time=36:00:00
#SBATCH --mem-per-cpu=8192
#SBATCH --partition=gpu
#SBATCH --gres=gpu:1

module load releases/2019b
module load TensorFlow/2.3.1-fosscuda-2019b-Python-3.7.4

srun pip3 install --user --upgrade pip
srun pip3 install --user scikit-build
srun pip3 install --user cmake
srun pip3 install --user -r requirements.txt
srun pip3 install --user tensorflow
srun nvidia-smi
srun python3 -c 'from tensorflow.python.client import device_lib'
srun python3 main.py lr1e6
```

Bibliography

[1] C. Sawyers, "Targeted cancer therapy," *Nature*, vol. 432, no. 7015, pp. 294–297, 2004.

[2] B. M. Dawant and A. P. Zijdenbos, "Image segmentation," *Handbook of medical imaging*, vol. 2, pp. 71–127, 2000.

[3] S. Jodogne, "The orthanc ecosystem for medical imaging," *Journal of digital imaging*, vol. 31, no. 3, pp. 341–352, 2018.

[4] K. Clark, B. Vendt, K. Smith, J. Freymann, J. Kirby, P. Koppel, S. Moore, S. Phillips, D. Maffitt, M. Pringle, *et al.*, "The cancer imaging archive (tcia): maintaining and operating a public information repository," *Journal of digital imaging*, vol. 26, no. 6, pp. 1045–1057, 2013.

[5] N. Rieke, J. Hancox, W. Li, F. Milletari, H. R. Roth, S. Albarqouni, S. Bakas, M. N. Galtier, B. A. Landman, K. Maier-Hein, *et al.*, "The future of digital health with federated learning," *NPJ digital medicine*, vol. 3, no. 1, pp. 1–7, 2020.

[6] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*, pp. 1273–1282, PMLR, 2017.

[7] D. A. Clunie, *DICOM structured reporting*. PixelMed publishing, 2000.

[8] E. T. Quinto, "An introduction to x-ray tomography and radon transforms," in *Proceedings of symposia in Applied Mathematics*, vol. 63, p. 1, 2006.

[9] J. Beatty, "The radon transform and the mathematics of medical imaging," 2012.

[10] R. Garnett, "A comprehensive review of dual-energy and multi-spectral computed tomography," *Clinical Imaging*, vol. 67, pp. 160–169, 2020.

[11] P. Jordan, P. M. Adamson, V. Bhattbhatt, S. Beriwal, S. Shen, O. Radermecker, S. Bose, L. S. Strain, M. Offe, D. Fraley, S. Principi, D. H. Ye, A. S. Wang, J. Van Heteren, N.-J. Vo, and T. G. Schmidt, "Pediatric chest/abdomen/pelvic ct exams with expert organ contours (pediatric-ct-seg)," 2021.

[12] J. Dong, Q. Chen, S. Yan, and A. Yuille, "Towards unified object detection and semantic segmentation," in *European Conference on Computer Vision*, pp. 299–314, Springer, 2014.

[13] M. Thoma, "A survey of semantic segmentation," *arXiv preprint arXiv:1602.06541*, 2016.

[14] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241, Springer, 2015.

[15] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

[16] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[17] H. J. W. L. Aerts, L. Wee, E. Rios Velazquez, R. T. H. Leijenaar, C. Parmar, P. Grossmann, S. Carvalho, J. Bussink, R. Monshouwer, B. Haibe-Kains, D. Rietveld, F. Hoebers, M. M. Rietbergen, C. R. Leemans, A. Dekker, J. Quackenbush, R. J. Gillies, and P. Lambin, "Data from nsclc-radiomics," 2019.

[18] C. H. Sudre, W. Li, T. Vercauteren, S. Ourselin, and M. J. Cardoso, "Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations," in *Deep learning in medical image analysis and multimodal learning for clinical decision support*, pp. 240–248, Springer, 2017.

[19] D. M. Hawkins, "The problem of overfitting," *Journal of chemical information and computer sciences*, vol. 44, no. 1, pp. 1–12, 2004.

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[21] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of big data*, vol. 6, no. 1, pp. 1–48, 2019.

[22] J. Léger, *Training biomedical image segmentation CNNs with scarce, cross-domain, prior, or noisy supervision*. PhD thesis, UCL-Université Catholique de Louvain, 2021.

[23] A. G. Roy, S. Siddiqui, S. Pölsterl, N. Navab, and C. Wachinger, "Braintorrent: A peer-to-peer environment for decentralized federated learning," *arXiv preprint arXiv:1905.06731*, 2019.

[24] G. Szegedi, P. Kiss, and T. Horváth, "Evolutionary federated learning on eeg-data.," in *ITAT*, pp. 71–78, 2019.

[25] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[26] L. Li, Y. Fan, M. Tse, and K.-Y. Lin, "A review of applications in federated learning," *Computers & Industrial Engineering*, vol. 149, p. 106854, 2020.

[27] M. Aledhari, R. Razzak, R. M. Parizi, and F. Saeed, "Federated learning: A survey on enabling technologies, protocols, and applications," *IEEE Access*, vol. 8, pp. 140699–140725, 2020.

[28] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[29] H. J. Aerts, E. R. Velazquez, R. T. Leijenaar, C. Parmar, P. Grossmann, S. Carvalho, J. Bussink, R. Monshouwer, B. Haibe-Kains, D. Rietveld, *et al.*, "Decoding tumour phenotype by noninvasive imaging using a quantitative radiomics approach," *Nature communications*, vol. 5, no. 1, pp. 1–9, 2014.

[30] J. Yang, G. Sharp, H. Veeraraghavan, W. Van Elmpt, A. Dekker, T. Lustberg, and M. Gooding, "Data from lung ct segmentation challenge," 2017.

[31] J. Yang, H. Veeraraghavan, S. G. Armato III, K. Farahani, J. S. Kirby, J. Kalpathy-Kramer, W. van Elmpt, A. Dekker, X. Han, X. Feng, *et al.*, "Autosegmentation for thoracic radiation treatment planning: a grand challenge at aapm 2017," *Medical physics*, vol. 45, no. 10, pp. 4568–4581, 2018.

[32] P. Jordan, P. M. Adamson, V. Bhattbhatt, S. Beriwal, S. Shen, O. Radermecker, S. Bose, L. S. Strain, M. Offe, D. Fraley, *et al.*, "Pediatric chest-abdomen-pelvis and abdomen-pelvis ct images with expert organ contours," *Medical Physics*.

[33] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečnỳ, S. Mazzocchi, H. B. McMahan, *et al.*, "Towards federated learning at scale: System design," *arXiv preprint arXiv:1902.01046*, 2019.

[34] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-iid data," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 9, pp. 3400–3413, 2019.

[35] L. Liu, J. Cheng, Q. Quan, F.-X. Wu, Y.-P. Wang, and J. Wang, "A survey on u-shaped networks in medical image segmentations," *Neurocomputing*, vol. 409, pp. 244–258, 2020.

[36] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, "3d u-net: learning dense volumetric segmentation from sparse annotation," in *International conference on medical image computing and computer-assisted intervention*, pp. 424–432, Springer, 2016.

[37] E. Brion, *Deep learning for organ segmentation in radiotherapy: federated learning, contour propagation, and domain adaptation*. PhD thesis, Université catholique de Louvain, 2020.

[38] F. Milletari, N. Navab, and S.-A. Ahmadi, "V-net: Fully convolutional neural networks for volumetric medical image segmentation," in *2016 fourth international conference on 3D vision (3DV)*, pp. 565–571, IEEE, 2016.

[39] N. Ibtehaz and M. S. Rahman, "Multiresunet: Rethinking the u-net architecture for multimodal biomedical image segmentation," *Neural Networks*, vol. 121, pp. 74–87, 2020.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**