Alma Mater Studiorum · Università di Bologna

Scuola di Scienze
Dipartimento di Fisica e Astronomia
Corso di Laurea Magistrale in Fisica

A deep learning approach to bone segmentation in CT scans

Relatore:
Prof. Renato Campanini

Correlatore:
Dott. Markus Wenzel

Presentata da:
Francesco La Rosa

Anno Accademico 2016/2017

Abstract

This thesis proposes a deep learning approach to bone segmentation in abdominal CT scans. Segmentation is a common initial step in medical images analysis, often fundamental for computer-aided detection and diagnosis systems. The extraction of bones in CT scans is a challenging task, which if done manually by experts requires a time consuming process and that has not today a broadly recognized automatic solution. The method presented is based on a convolutional neural network, inspired by the U-Net and trained end-to-end, that performs a semantic segmentation of the data. The training dataset is made up of 21 abdominal CT scans, each one containing between 403 and 994 2D transversal images. Those images are in full resolution, 512x512 voxels, and each voxel is classified by the network into one of the following classes: background, femoral bones, hips, sacrum, sternum, spine and ribs. The output is therefore a bone mask where the bones are recognized and divided into six different classes. In the testing dataset, labeled by experts, the best model achieves a Dice coefficient as average of all bone classes of 0.93. This work demonstrates, to the best of my knowledge for the first time, the feasibility of automatic bone segmentation and classification for CT scans using a convolutional neural network.

Sommario

Questa tesi propone un approccio di deep learning per la segmentazione di ossa in scan CT addominali. La segmentazione è un primo passo comune nell'analisi di immagini mediche, spesso fondamentale per sistemi di diagnosi assistita dal computer. L'estrazione di ossa nelle scan CT è un processo complicato che se effettuato manualmente da esperti richiede un lungo tempo e che non ha oggi una soluzione automatica ampiamente riconosciuta. Il metodo presentato è basato su una rete neurale convoluzionale, ispirata alla U-Net e allenata end-to-end, che esegue una segmentazione semantica dei dati. Il dataset di allenamento è costituito da 21 scan CT addominali, ognuna delle quali contiene tra le 403 e le 944 immagini 2D trasversali. Queste immagini sono in full-resolution, 512x512 voxels, e ogni voxel viene classificato dalla rete in una delle seguenti classi: sfondo, femore, bacino, sacro, sterno, colonna vertebrale e costole. L'output è pertanto dato da una maschera di ossa dove esse sono state riconosciute e divise in sei classi differenti. Nel dataset di test, annotato da esperti, il modello migliore raggiunge un Dice coefficient come media di tutte le classi di ossa pari a 0.93. Questo lavoro dimostra, a mia conoscenza per la prima volta, la possibilità di effettuare una segmentazione e classificazione automatica di ossa in scan CT usando una rete neurale convoluzionale.

Contents

Chapter 1

Introduction

Deep learning and artificial intelligence are terms that can be heard almost every day regarding the most various tasks. From self-driving cars to face recognition systems, financial forecasts or automatic diagnosis. It looks like through those algorithms automatic systems will be able to revolutionize the world in few years; however, until now the expansion of this discipline is mostly due to the success of convolutional neural networks. Convolutional neural networks are a type of deep neural networks and have recently achieved outstanding results in speech recognition, object recognition and natural language processing tasks [1]. Furthermore, several studies have successfully applied them for analyzing medical images and performing tasks such as image classification, object detection, segmentation and registration [2]. The next main issue is to implement those methods in a clinical environment, with a simple, easy-to-use interface for the physician. If achieved this could completely revolutionize healthcare. In this work it is explored the possibility of convolutional neural networks to perform automatic bone segmentation in CT scans. This is a challenging task that would be very helpful to support physicians in their decision-making process while analyzing those images. The second chapter of the thesis deals mainly with fundamental concepts of deep learning and convolutional neural networks, with a particular emphasis on their applications to medical images.
Image segmentation is the process of subdividing an image into its constituent regions or objects [3], obtaining a representation easier to analyze and study further. This is done assigning to every pixel of the image a label. Thus, segmentation is essentially a pixel classification task. Pixels belonging to the same object or sharing certain features have the same label and are in this way grouped together. Image segmentation has several real-world applications, for example in machine vision, object detection, medical imaging, video surveillance and recognition tasks. A considerable number of traditional computer vision and machine learning techniques have been developed and successfully applied to this task. Regarding medical images there is not a universal algorithm used for automatic segmentation. The choice of the method applied depends on the imaging modality, part of the body analyzed and goal of the study [4]. In certain cases a manual segmentation still represents the gold standard, even if this process is tedious, time-consuming and prone to errors. Different types of artifacts, noise due to the acquisition system and partial volume effect are all factors that need to be considered both in the manual and automatic segmentation methods. In the third chapter some of the most common segmentation techniques are briefly introduced before focusing on the novel deep learning approaches to this task. Here it is also discussed the U-Net [5], the fully convolutional neural network from which the network of this work takes inspiration. The U-Net was developed for biomedical image segmentation and its main innovative feature is the so-called encoder-decoder structure. This allows to obtain in output a segmentation map of the same size of the input image.

The segmentation of bones in CT scans is a crucial step for helping physicians in several medical tasks. For example, it is widely used in orthopedic surgery, in locating fractures, diagnosing bone diseases and to support planning for therapies. Although most of the bones can be visually identified in CT images without difficulties, a precise automated segmentation is still very challenging. CT scans present often a low signal-to-noise ratio, insufficient space resolution and several artifacts that make it a difficult task. The fourth chapter has a concise paragraph about computed tomography and then it deals with bone segmentation in CT scans, analyzing some relevant and recent publications on the topic.
One of the factors that has allowed such an incredible expansion of deep learning in the last years is the increased GPU computational power. Modern GPUs drastically reduced the training time of complex networks and outstanding results were achieved by deep learning models in computer vision tasks and competitions. In the fifth chapter this is briefly described, as well as the deep learning frameworks and computational tools used in the code of this work.
The method proposed in this thesis is based on a convolutional neural network and has the goal of segmenting bones and classifying them into separate classes. In particular, the bones are separated from other structures and tissues of the body and divided into six classes: femoral bones, hips, sacrum, spine, ribs and sternum. The femoral bones class includes the upper part of the femur present in the scans, the femoral necks and femoral heads. All the other classes include the bone they are named after. The sixth chapter describes the network architecture and the pre-processing and data augmentation applied to the data.
The training dataset consists of 21 abdominal CT scans, each one containing between 403 and 994 transversal slices. Every slice is in full resolution, 512x512 voxels. A single CT scan is kept as validation dataset. The labeling of bones for both the training and validation dataset ground truth is made with a semi-automatic method and then checked manually by the author of the thesis. The testing dataset is made up of 4 CT scans manually annotated by experts. In the seventh chapter the training process and the results obtained on the testing dataset are discussed. Moreover, here a further analysis about the model robustness to gaussian noise is shown.
For the work it has been decided to choose a deep learning framework that is compatible with the MEVIS infrastructure for the rapid development and deployment of clinical prototypes including deep learning. Thus, the translation of the results into clinical applicability is secured. The main purpose of the thesis is to test a certain type of convolutional neural network on CT scans and verifying its performance in an important medical task that has not been addressed with a deep learning approach yet. The eighth and last chapter presents the conclusion of this work and possible future developments.
The work of thesis was mainly developed at Fraunhofer MEVIS in Bremen, Germany, a research institute that develops real-world medical software solutions for image-supported early detection, diagnosis, and therapy. All the medical data used belong to this institute.

Chapter 2

Deep learning and convolutional neural networks

2.1 Deep learning

Artificial intelligence (AI) is today one of the most actual and discussed field, both by researchers and the general public. AI is by definition the capability of computers to perform tasks that normally require human intelligence. There is currently a huge number of active research topics regarding AI, all with the common aim of automating some type of function. Some examples are understanding and recognizing speech and images, competing in strategic game systems, self-driving cars, interpreting complex data and making diagnoses in medicine. Moreover, these techniques are constantly applied to new problems and their full potential has not been totally investigated yet. Initially, AI methods successfully dealt with tasks which were considered complex for human beings, but could be described by a list of logical and mathematical rules, like solving problems in algebra or proving logical theorems. It turned out later that is much more challenging to solve problems which people perform intuitively like recognizing a visual pattern or a certain sound.
A possible approach that is widely explored nowadays is to allow the computers to learn from experience, letting them to tackle a problem based on what they have already seen before. They acquire knowledge of the world building up a hierarchy of concepts, where each single concept is interpreted as an ensemble of simpler ones. Therefore abstract and complex representations are learned from easier factors. In this way the machines automatically build hierarchical statistical models, without anyone programming exactly what they need to learn. If we visualize this hierarchy of concepts, it would contain many layers and that is one of the reasons why this approach is often referred to as deep learning [1].
Deep learning, a branch of machine learning, regards all those methods based on learning representations of data. On one hand, common machine learning algorithms perform very differently depending on the representation of the data they are given. Any information included in those representations is called feature and those designed by human beings are referred to as hand-crafted features. Selecting the right features usually improves the machine learning algorithm performance. On the other hand, representation learning is an approach that involves not only learning the mapping from representation to output, but also the representation itself. This obviates the need of hand-crafted features and often gives better results, generalizing also to new tasks with very little human effort needed. In certain cases, obtaining a proper representation can be as difficult as solving the problem itself. For this reason deep learning is based on multiple levels of concepts or features of the data, where the higher level ones are built upon simpler ones. This gives a great power and flexibility that allow to tackle complex real-world problems. Another characteristic of

deep learning networks is that they are made of a series of successive layers. At each layer the signal is processed by a specific unit, whose parameters are learned during training, and passed to the next one. In a common network, the layers between the input and the output are referred to as hidden layers and allow to apply various linear and non-linear transformations. The networks having architectures like those are the so-called deep neural networks, because of the large number of hidden layers between the input and the output. In general, deep learning algorithms can be divided into supervised and unsupervised: the former regard learning features from labeled data, whereas the latter aim at a pattern analysis. From now on I will mainly focus on supervised learning methods.

2.2 Supervised learning

In a supervised learning task the dataset consists of a set of training samples where each one contains the input object and the desired output, which is often known as label or target. The algorithm is responsible for analyzing the data, extracting significant features from it and producing a function which can make predictions on new, previously unseen data. Those samples make up what is generally referred to as testing dataset.
A few guidelines are generally followed while solving a problem of supervised learning. First of all, the training dataset gathered should be representative of the real-world distribution of the samples. In other words, the samples belonging to the training and test dataset should all be independent and identically distributed. Secondly, the input representation of the data should be carefully chosen. It should not contain a too large amount of information, but enough for the algorithm to be able of inferring an appropriate function. Furthermore, a proper learning algorithm is determined and its parameter are adjusted. Finally, the performance should be evaluated on a new dataset which contains samples unseen by the algorithm before.
Supervised learning methods deal mainly with two categories of problems: regression and classification. Regression is the process of predicting a variable with continuous values, whereas a classification task predicts only discrete values or categories into which the data is separated. Some common algorithms employed for both purposes are support vector machines, artificial neural network and decision trees, but many more are used as well. In the next paragraph the main features of convolutional neural networks (CNNs), a type of artificial neural network, are summarized.

2.3 Convolutional neural networks

Convolutional neural networks (CNNs) are a type of feed-forward artificial neural networks successfully employed today to tackle a wide range of problems. They are inspired by the animal visual cortex, in which the neurons are arranged in such a way that are sensible only to a small sub-region of the visual field, called receptive field. The receptive fields of all neurons are then tiled in order to cover the whole visual field. Convolution is the name of the mathematical operation mainly employed by these networks.
CNNs are very similar to common neural networks, but they make the important assumption that the input data is arranged in a grid-like topology. The most straightforward example of this kind of data are images, having pixels in a 2D grid. The architecture of CNNs takes advantage of this fact in order to optimize the learning. The core building

block of CNNs is the convolutional layer, in which neurons are arranged in three dimensions and a 3D volume input is processed to output another 3D volume. In Figure 2.1 a scheme of this arrangement is shown.
A common CNN architecture is composed of several convolutional and pooling layers followed by one or more fully connected layers before the output. In a fully connected layer each neuron has connections to every neuron of the previous layer and therefore a large number of parameters is required. Those layers are used before the output to reduce the size of the activation maps to a single vector of class scores. In the next paragraphs the main features of convolutional and pooling layers are described.

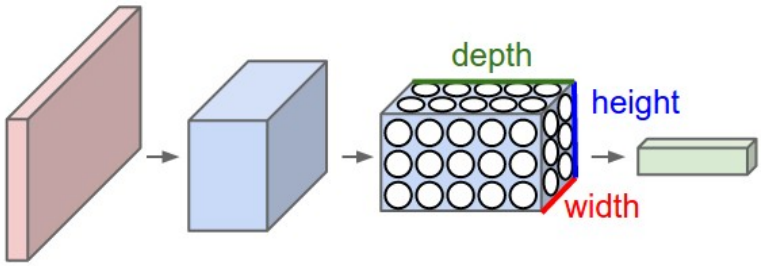


Figure 2.1: An schematic example of how neurons are arranged in three dimensions in a convolutional layer [6].

2.3.1 Convolutional layer

Convolutional layers are the core building block of CNNs. They have a set of parameters, known as filters, whose values are adjusted during training. The filters are small spatially, but each one spans over the whole depth of the 3D input volume. Those characteristics allow to store fewer parameters and compute less operations. During the training phase, each filter is convolved with every 2D input image, generating a 2D activation map. All these maps are then stacked together producing a 3D output volume. Three main features characterize a convolutional layer: local connectivity, sparse arrangement and parameter sharing.

- Local connectivity. Local connectivity refers to the fact that neurons in convolutional layers are connected only to certain neurons of the previous layer and therefore they see only a subregion of the input. The local filter size represents a hyperparameter of the model and is adjusted depending on the specific task. Larger filters are computationally more expensive, because more parameters are stored and when the filter is convolved over the input image more operations are computed. However, these sparse interaction regard only the height and width of the 3D input volume and filters cover always their full depth (see Figure 2.1). Through a series of different types of layers, the deepest parts of a network are able to see a wider context, obtained as the accumulation of the previous layers. This is referred to as receptive field and for the first layer is equal to the filter size.

- Sparse arrangement. As mentioned above, the output of a convolutional layer is a 3D volume and its sizes are determined by three hyperparameters, which are values initially set by the operator that then remain fixed during the training process. These are depth, stride and zero-padding. Depth refers to the number of filters applied and

controls the depth dimension of the output volume. Stride specifies how to shift the filter over the input image at each step. With a stride equal to 1 the filter is moved one pixel at a time, producing a large output map. Higher stride values signify that the filter slides over the image with bigger steps and therefore a smaller feature map is generated. Zero-padding indicates the number of zeros to add around the borders of the input image and it is useful in order to preserve the same sizes between input and ouput.

- Parameter sharing. Parameter sharing is a very efficient technique to reduce the total number of parameters in a layer. It is based on the assumption that a feature computed at some spatial location could be also found at a different location. Therefore this scheme constrains all neurons of a 2D slice to have the same weights and bias. This dramatically reduces the number of operations required in the forward pass. Moreover, as there is only one filter per feature map generated, this operation simply consists in the convolution of the filter over the input image.

The size of the output volume of a convolutional layer can be determined considering the parameters defined above. In particular, the depth is equal to the number of filters, whereas the width W and height H can be derived as follows:

$$W = \frac{w - F + 2P}{S} + 1 \qquad H = \frac{h - F + 2P}{S} + 1 \tag{2.1}$$

where w and h are respectively the width and height of the input image, F is the filter size, P is the amount of zero-padding and S is the stride.
The reverse of convolution, commonly referred to as up-convolution or deconvolution, is also often used in CNNs. This operation essentially performs, with learnable filters as the convolution itself, an up-sampling of the feature maps given as input. It turns out to be useful in case of semantic segmentation tasks, where after a series of convolutional and pooling layers the size of the feature maps needs to be restored equal to the input image.

2.3.2 Pooling layer

Pooling layers are another type of widely used layers in CNNs. They are often inserted between convolutional layers and have the function of downsampling the input volume spatially, allowing to extract features at different resolutions. Applied independently to each single slice of the input, pooling layers reduce its size and also the total number of parameters. The most common operation employed is max-pooling, which simply returns the highest value between the pixels analyzed. Generally, pooling is applied with filters of size 2x2 and stride 2. In this case, the output height and width of the volume are halved compared to the input, whereas the depth remains unchanged. An example is shown in Figure 2.2.

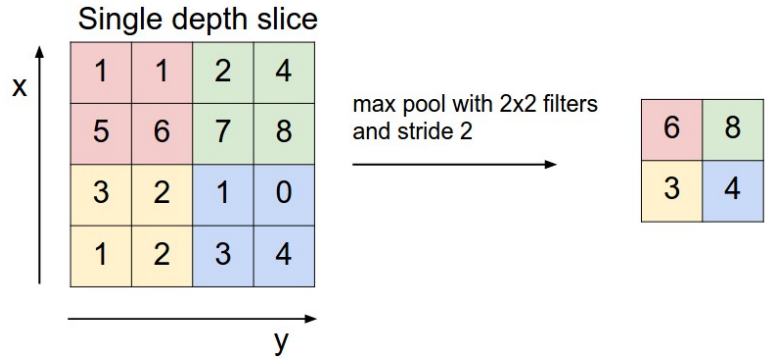


Figure 2.2: Example of a 2x2 max pooling operation[6].

2.3.3 Activation function

As in common neural networks, activation functions are usually applied to the output of convolutional layers. In this case, they have to be non-linear in order to bring the needed non-linearity properties to the network. An activation function performs a pixel-by-pixel mathematical operation which is fixed and does not require any parameters. Some examples of activations functions used in deep neural networks are the sigmoid, the tanh and the Rectified Linear Unit (ReLU). This last one is the most popular in CNNs and implements the following function:

$$f(x) = \max(0, x) \tag{2.2}$$

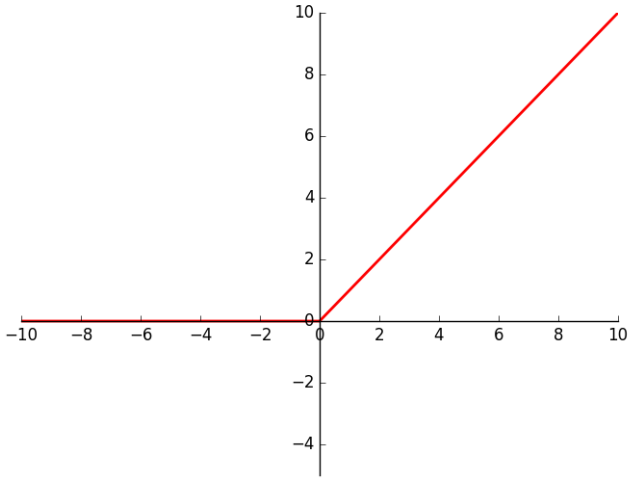


Figure 2.3: Graph of the ReLu activation function.

which is shown in Figure 2.3. It is a very straightforward thresholding at zero, which however has proved to be very efficient for the convergence of stochastic gradient descent compared to other functions. Another advantage is that the operation involved is very simple and not computationally expensive, making it quick to calculate. The main drawback on the other side is encountered with very large flows of gradient descent, which cause

the ReLU units to "die" and not reactivate any further. One reason of this issue can be a too high learning rate. A possible remedy is the use of leaky ReLU, a function exactly as the ReLU, but with a small negative slope when x<0. Results on the effectiveness of this activation function are, however, inconsistent [6].

2.3.4 CNNs architecture

The architecture of a CNN is in the simplest case made of an input layer, a series of convolutional and pooling layers, one fully connected layer that computes the class scores and finally the output. A different kernel size, number of activation maps or convolutional layers are then adjusted depending on the specific task, as well as the value of the hyperparameters. An interesting development of CNNs is represented by fully-convolutional neural networks. Those are networks without fully-connected layers, which generally produce an output of the same size of the input. This type of networks is successfully used for semantic segmentation of images [7].
It goes beyond the scope of this work to discuss complex deep neural networks in details, a recommended reference is [1].

2.3.5 Backpropagation algorithm

During training, the process of producing an output $\hat{y}$ from an input x is known as forward propagation or forward pass. A cost function $J(\theta)$ is then computed comparing $\hat{y}$ with the desired output. Next this value is propagated backward until each unit of the model has an associated error. The back-propagation algorithm computes then its gradient with respect to the weights of the units. Finally, another algorithm, as for example the stochastic gradient descent (SGD), is used for updating the weights starting from the gradients. The whole learning process therefore requires both algorithms and is repeated for each batch of the training data.
Back-propagation is the most common method of computing gradients in artificial neural networks and CNNs. It just requires that the activation function of the network units is differentiable and is able to calculate the gradient in an efficient and inexpensive way. This method is based on the chain rule for differentiating compositions of functions, which can be expressed as follows:

$$\frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx} \tag{2.3}$$

Back-propagation applies the chain rule iteratively for the neurons of each layer, starting from the output and moving backward, and in this way computes the gradient of the loss function. This gradient is then used by an optimization algorithm to perform learning and update the weights of the neurons in the direction of the global minimum of the loss function. In Subsection 6.3.4 it is briefly described Adam, the optimization algorithm chosen for the network of this thesis.

2.4 Deep learning applied to medical images

In the past two years there has been an incredible expansion in the usage of deep learning algorithms for medical images analysis. An increasing number of papers is being published on the topic and various of them have reached human expert-level performance [2]. The most explored tasks so far are image classification, object detection, segmentation and registration, but many more are being investigated. Compared to other computer algorithms, deep learning has the crucial advantage of finding the informative representations of the data by itself. Therefore the complex and time-consuming step of manual features engineering can be avoided.
Nowadays a major challenge in applying deep learning to medical images analysis is the limited amount of data available to researchers. This can lead to an overfitting of the training data with a final low performance in the test dataset. In order to cope with this problem, several strategies are being investigated. Some of them artificially generate more data applying affine transformations to the initial dataset (data-augmentation), others attempt to reduce to total number of parameters of the models or initialize those with pre-trained models from non-medical images and then fine-tune them on the specific task. However, the data itself exists, as millions of medical images are stored in the hospital archives. Gaining access to those archives is the main problem nowadays because of the various regulations present. Each image is also generally stored with patient information, so a process of data anonymization is required as well before a study can be undertaken. In the last years several dataset have been made publicly available and this trend is expected to accelerate in the future.
CNNs have drawn a great interest on the topic because of their intrinsic capability of accepting images as input. They can perform a classification or segmentation task and have proved to be the most successful type of artificial neural network for image analysis problems.
Deep learning offers exciting solutions and prospectives for medical image analysis. There is room for improvements regarding both the algorithms and the way to acquire large training datasets. As this last challenge will be overcome, in the next years deep learning will really play a key role also in medical imaging.

Chapter 3

Segmentation of medical images

Image segmentation is the process of automatically or semi-automatically subdividing an image into significant regions. Its aim is to locate the voxels that make up either the borders or the interior of the objects analyzed. A label is then assigned to every single voxel in what is commonly referred to as semantic segmentation and the result is an image where voxels with the same label share certain characteristics. Therefore image segmentation provides a more meaningful representation of the data and it is a crucial step for fully understanding the content of medical images and doing diagnosis.
In medical images processing, this task generally extracts essential information for a quantitative analysis of clinical parameters. The segmentation of bones, organs or other substructure is indeed a fundamental step for the vast majority of computer-aided detection (CAD) systems. Different approaches have been proposed in literature for this scope and in the last years this has been the most common subject of papers applying deep learning to medical images [2]. However it still remains a challenging problem due to various factors as image noise, artifacts, low contrast, inhomogeneities and many more. An ideal automatic segmentation method should have all the following features: accuracy, reliability, repeatability, robustness and least dependency on the operator [4].
In the next paragraphs the most successful medical images segmentation methods, conveniently divided into techniques based on gray level features, on texture features and supervised and unsupervised methods are briefly described. Then it is analyzed how this task is being tackled by novel deep learning algorithms, which in the last years have achieved surprising results. Toward the end of the chapter some recent and significant publications on the topic are mentioned and discussed. Those include the U-Net [5], a convolutional network for biomedical images segmentation on which is based the architecture of the network employed in this thesis work.

3.1 Gray level features methods

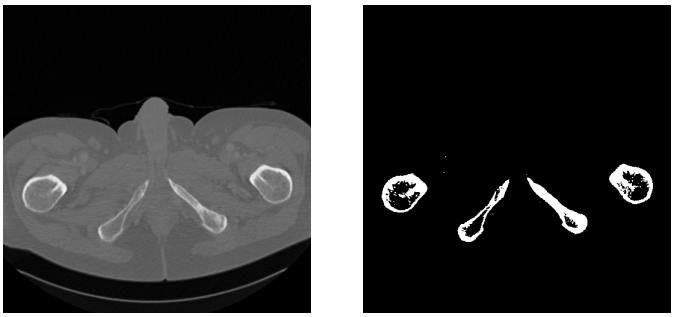
By gray level features methods it is generally referred to the techniques that perform segmentation based on the intensity values of the pixels. Those include thresholding operations, region growing, region split and merge, edge based segmentation and a few more. In the next paragraphs the most used methods dealing with medical images are described.

3.1.1 Thresholding

Thresholding is probably the simplest segmentation method and it is based on the assumption that different regions of the image have different intensity values. It tries to find a suitable gray value that divides the pixels in foreground, with an equal or higher intensity, and background, with a lower intensity. Therefore if can be defined mathematically as follows:

$$f_{k,j} = \begin{cases} 1 & i_{k,j} \geq t \\ 0 & i_{k,j} < t \end{cases}$$

where t is the threshold value, $i_{k,j}$ is the initial value of the pixel at coordinate (k, j) and $f_{k,j}$ is the resulting value after the threshold is applied. In Figure 3.1 an example of thresholding applied to a training image of this work is shown. This method works well only when two classes are present in the image and all the pixels of each class have similar intensities. In order to segment more than one object with different gray values, another similar method called multithresholding or band thresholding is applied. In this case multiple threshold values are chosen and different classes are identified.



(a) (b)

Figure 3.1: Thresholding applied on a CT image for bone segmentation. (a) Original image. (b) Resulting image after thresholding. The image belongs to the training dataset of this work.

3.1.2 Otsu's thresholding

Generally, analyzing the histogram of an image helps to find a suitable thresholding value. However, doing it visually can lead to problems; as a consequence various automated methods have been proposed. Otsu's thresholding is an algorithm that automatically finds an optimal thresholding value, assuming that the image contains only two classes of pixels. This value is computed as the one that minimizes the intra-class variance or similarly maximizes the inter-class variance. Otsu's method is based on the assumption that well-thresholded classes should be distinct with respect to the intensity values of their pixels [3]. As this is a totally automated method, which requires only the histogram of the image, it is generally straightforward to implement, but shows some limitations when the object and background have means close to each other and large variances.

3.1.3 Region growing

Region growing is a segmentation method based on some predefined similarity criteria between the pixels. It requires the initial selection of representative pixels, known as seed points, which are chosen according to this homogeneity criterion (most of the times their intensity value). Neighboring pixels with similar properties are then clustered together to form bigger and bigger regions. The process terminates when all pixels of the image have been assigned to a region. The main drawback of this method is that the final result significantly depends on the initial choice of the seed points.
Region growing has been extensively applied for the segmentation of medical images, for example in mammographic images [8]. Based on this method, further algorithms able to learn the homogeneity criterion automatically from the characteristics of the regions have been developed [9].

3.1.4 Region split and merge

Region split and merge is a segmentation approach similar to the one described in the last paragraph, as it is based on a homogeneity criterion as well. Starting from the entire image, the algorithm checks if it is uniform in properties. If not, the image is split into four equal-size subsections. Then the process continues iteratively for each subsection until only homogeneous subregions are left. However, between one step and the following one, adjacent subregions satisfying the condition of homogeneity are merged together. When no further split and merge of regions is possible the algorithm stops.

3.1.5 Edge based segmentation

These methods perform segmentation starting from the detection of boundaries which divide different regions. Their aim is to find all the factors of discontinuity in gray level, color, etc. and depending on them subdividing the initial image into different subregions. Various edge detection operators based on gradient are generally applied, as Prewitt, Canny, Sobel, Robert and Laplacian. Then the detected borders are combined and different regions are identified.
It goes beyond the scope of this work to discuss these filters in details. Due to the presence of noise, weak edges and other factors these methods have many limitations and are often used in conjunction with other more advanced techniques.

3.2 Texture features methods

Texture is defined as something consisting of mutually related elements. A texture may be fine, coarse, smooth, or grained depending upon its tone and structure. While tone is based on pixel intensity properties, structure is the spatial relationship between pixels [4]. Textures features can be interpreted as defining characteristics of certain regions of an image and have been widely used for both segmentation and classification. Compared to gray level based methods, the techniques based on these features have proved to perform better in certain segmentation tasks of medical images. See this reference [4] for more details.

3.3 Supervised and unsupervised methods

Another approach for medical images segmentation includes several artificial intelligence algorithms, which can be classified as supervised and unsupervised. In the following paragraphs some examples of both categories are described.

3.3.1 Clustering methods

The unsupervised or clustering methods aim at dividing the initial data in a way that similar objects end up in the same group. They do not need training data and the operator involvement is minimal. These algorithms have the advantages of generalizing well on different set of data and generally require a small amount of time. However, being very sensible to noise they do not always show optimal results.
K-means is a broadly used method for cluster analysis. It consists in a quite straightforward iterative algorithm. Initially k centroids are chosen at random locations in the input space and subsequently this procedure is followed:

- for all objects the Euclidean distance from the centroids is computed
- every object is assigned to the nearest centroid
- the centroids are updated as mean value of each cluster

and then all the previous steps are repeated until the components of each group do not change anymore. On one side this algorithm is pretty straightforward to apply, but on the other side it has a few disadvantages as the number of clusters, that have to be selected by the operator, and its sensibility to noise, outliers and initial values. Algorithms based on K-means clustering has been widely applied in the medical images analysis field, for example for brain and brain tumor segmentation in MRI scan [10, 11].
Another unsupervised algorithm, which has performed well on medical image, is Fuzzy C-means. This approach is very similar to k-means, with the main difference that each point has a weight associated to every cluster. As a consequence a so called soft segmentation is produced, which means that a point does not belong only to a single cluster and overlapping regions are possible. This last aspect is significant for medical images where tissues clearly overlap. Indeed, Fuzzy C-means has been successfully applied in medical images, especially for segmentation on MRI scans [12]. A study has also compared k-means and Fuzzy C-means on MR brain images [13], showing that one algorithm is not always better than the other, but their relative performance depend on the specific task.

3.3.2 Supervised methods

Several supervised methods, also known as classifiers, have been proposed for image segmentation. They are based on supervised learning (see Section 2.2) and therefore need a training dataset, made up of objects and their target labels. Manually segmenting the training dataset often requires some effort and it is a process that depends on human ability. However, these methods are able to adapt very well to different tasks and solve complex problems.
K-nearest-neighbour (k-nn) is a widely used supervised method. This approach needs a large amount of labeled samples and it is considered a nonparametric classifier, as it does not consider the statistical structure of the data. In the k-nn algorithm each pixel is classified in the class to which belong the majority of his k closest neighbors, where k is a

parameter selected by the operator. The performance of this method are affected by the choice of k, by the measurement for distance and by the way of counting votes.
Another commonly used method is the maximum likelihood, which assumes that the data is made up of independent and identically distributed samples, belonging to an unknown, usually Gaussian, distribution. The parameters of the distribution are then estimated maximizing the likelihood principle.
To the supervised methods category belong also the supervised artificial neural network based algorithms. Feed-forward and feedback networks have been widely used for medical images segmentation [2]. In the last years several studies using deep learning for medical image segmentation have been made and in the following paragraphs the most relevant ones are analyzed.

3.4 Deep learning approaches

In the last years deep learning based segmentation methods have been successfully applied to many medical images segmentation tasks. For example of brain regions, coronary artery calcifications, organs and various substructures [2]. All these studies are based on CNNs, but each one applies a different type of architecture. In the next paragraph it is briefly described how common classification CNNs have been transformed into segmentation networks. In the last two paragraphs of the chapter some specific studies about this topic are analyzed.

3.4.1 CNNs for medical image segmentation

Common CNNs are in their intrinsic form classification networks [1]. They take as input an image and give as output a vector containing the probabilities of the image to belong to each possible class. CNNs usually deal with the full pipeline for solving the task, from feature extraction to learning the desired results. Those methods, which omit any hand-crafted intermediary algorithms, are called end-to-end training. Thus, end-to-end approaches reduce the human effort and they have achieved incredible results in different applications, from self-driving cars to medical imaging segmentation tasks [14, 15].
The CNN architectures can indeed be easily adapted for a segmentation task, where each single pixel or voxel is assigned to a class. This was initially achieved with a patch-wise classification based approach. In this approach a single pixel is classified considering a patch around it and then the patch is moved in a sliding window way until every pixel of the image is analyzed. However, two relevant drawbacks were observed. Firstly, as the patch generally includes only a small area around the pixel, the network learns mostly local features, ignoring global patterns. Secondly, due to multiple overlapping patches, the convolutions are computed redundantly.
A different approach, proposed to overcome these limitations, employs the common CNN architecture replacing the fully connected layers with convolutions. In this way a multi-class prediction for multiple pixels is achieved with less computations. The output of the network, however, ends up smaller than the input, due to the convolutions and pooling layers (see Section 2.3). In order to generate a segmentation map with the same size of the original image several strategies have been investigated. A relevant one proposed by Long et al.[7] introduces deconvolution operations to upsample the reduced size feature maps. Deconvolution is described by the authors of [7] as backwards-strided convolution: apply-

ing it as many times as the pooling layer restores the initial size of the image. This type of network, without fully-connected layers, is commonly referred to as "fully-convolutional network".
Ronneberger et al. [5] started from this last study and developed another architecture (called U-Net) for biomedical image segmentation. It consists of a contracting path, made of a common CNN, followed by an expanding part where deconvolution is used to restore the initial size of the image. Due to its structure, this type of network is also known as encoder-decoder. In the next paragraph this study is described in details.

3.4.2 U-Net

The U-Net is definitely the most well-know CNN architecture for medical images segmentation and it is the one my personal work is based on. In Figure 3.2 is shown the fully-convolutional network architecture, which presents a design similar to a convolutional autoencoder. The study took the main idea of the deconvolutional layers from [7], but it has a relevant difference: so-called skip connections between feature maps in the same depth level of the contracting and expanding path are present. The feature maps are then concatenated and therefore the U-Net has a significant advantage compared to the patch-wise approaches: the global features are allowed to propagate to higher resolution layers. For this reason the skip connections characteristic is found in many other segmentation works, with the features maps combined in different ways.

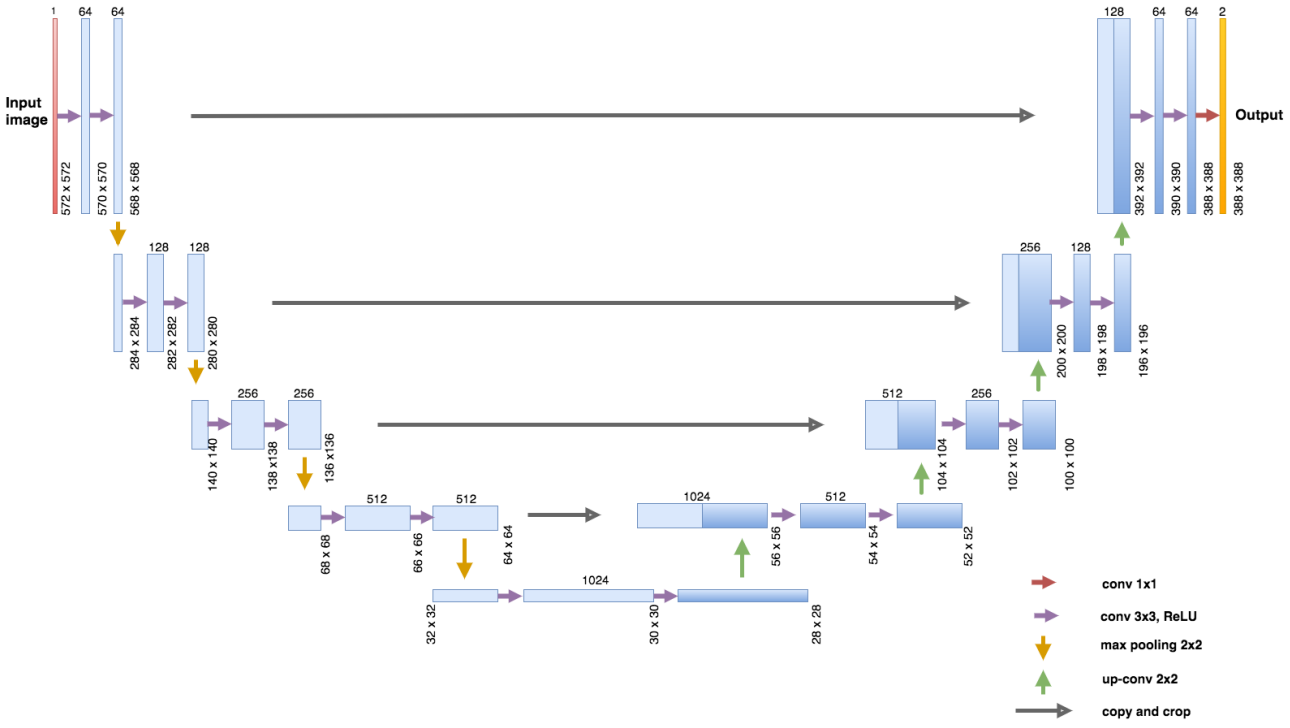


Figure 3.2: U-Net architecture [5]

The network architecture is made up of five consecutive depth levels. In the contracting path each level applies two 3x3 convolutions, each followed by a rectifier linear unit (ReLU)

activation function, and a 2x2 max pooling operation which reduces by half size the feature map. The feature channels are initially 64 and double at each following level. The expanding path is similar with the exceptions that the feature maps from the contracting part are concatenated and instead of max pooling a 2x2 up-convolution is applied. This last operation doubles the size of the image at each layer. Following the last expanding layer a convolution with filters of size 1x1 is used to output the final segmentation map.
In the original paper the U-Net was developed in Caffe [16], a widely used deep learning framework, and trained with stochastic gradient descent as optimization algorithm. A pixel-wise softmax is applied to the last feature map and then the cross entropy is used as loss function. A relevant detail is that a weight map is introduced in the cross entropy expression in order to compensate classes frequency imbalance. In medical images segmentation works the classes imbalance is an issue encountered often and the loss function weighting represents a possible solution. Another method consists in using the Dice coefficient as metric [14], which considers the overlapping regions between the original image and the segmentation mask.
Data augmentation is an essential technique applied in this work. It is always complicated to acquire very large dataset for medical imaging analysis and this is a straightforward and effective way to deal with this problem. In this work shift, rotations and random elastic deformations of the original images were applied, allowing to achieve high performance with only few training samples available.
The U-Net was trained end-to-end with only a few training images available and achieved outstanding results in different biomedical images segmentation tasks. It outperformed the previous best method on the ISBI challenge for the segmentation of neuronal structures in electron microscopic stacks and won the ISBI cell tracking challenge of 2015 regarding transmitted light microscopy images.

3.4.3 Following studies

The U-Net architecture was extended to 3D implementations by Cicek et al. [17]. Their network is trained with 2D annotated slices and it generates dense volumetric segmentations. This is an interesting development of the original U-Net as a high percentage of medical data is in 3D, with similar size in each dimension, and a slide-by-slides application of 2D convolutional operations can be inefficient. Inspired by the U-Net, another volumetric, fully-convolutional neural network for 3D image segmentation was proposed by Milletari et al. [14] with the name of V-Net. The V-Net performs 3D convolutions with volumetric kernels and employs the so-called residual blocks to tackle the vanishing gradient problem. The residual block is a learning framework recently introduced in order to make the optimization process easier in image recognition tasks [18]. At the end of these blocks the signal, processed through convolutions and non-linearities, is summed with the initial input of the stage. In this way less time is required to reach convergence. In Figure 3.3 a building block of residual learning is shown. Another relevant difference of the V-Net compared to the U-Net is the objective function. The cross-entropy is indeed not used anymore and to cope with the unbalanced classes problem an objective function based on the Dice coefficient is proposed. Maximizing the dice similarity coefficient does not require any hyperparameters (as the weight map, for example) and that is therefore an advantage of this approach. In the original study the V-Net was trained end-to-end on a dataset of prostate scans in MRI, given the relative manual ground truth annotations. It achieved a fast and accurate segmentation based on the Dice coefficient.
A recent study has also investigated the possibility of using the same CNN for different segmentation tasks in images acquired with diverse modalities [19]. In this study it was

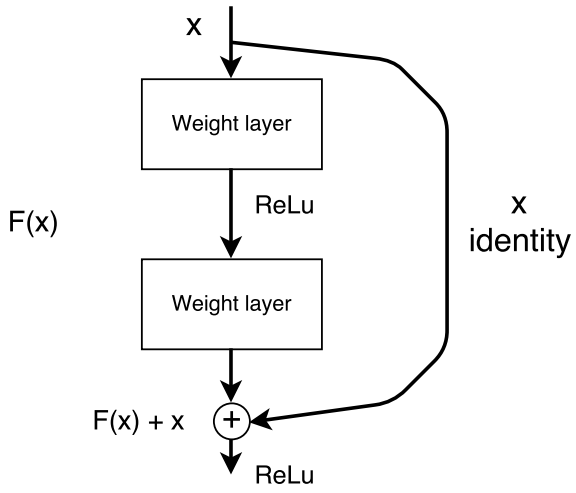


Figure 3.3: Scheme of a residual block, as proposed in [18].

proposed a single convolutional network which performed well in the segmentation of tissues in MR brain images, of the pectoral muscle in MR breast images and of the coronary arteries in cardiac CTA. Due to their ability to adapt and capability to learn, the CNNs could therefore tackle in the future different segmentation problems without a previous task-specific training.
In addition, it is worth mentioning a brand new segmentation approach based on adversarial training. Goodfellow et al. [20] introduced in 2014 generative adversarial networks, a new framework for estimating generative models via an adversarial process. Here two networks are trained simultaneously, one that generates data and one that learns to determine whether a sample comes from the training data or the generative model. The competition between the two models drives the generative network to improve its results, producing data that the discriminative model cannot distinguish from the original one. Recently, adversarial networks have been introduced also for semantic segmentation [21]. This framework has proved to be effective to correct inconsistencies between the ground truth and the segmentation map produced by a segmentation CNN. This approach has achieved interesting results in the medical imaging field regarding the segmentation of prostate cancer in MRI [22] , organs in chest X-Rays [23] and brain MRI [24]. All those studies show segmentation performance improvements compared to previous methods based on a single segmentation net. A further investigation of this framework, also regarding different tasks, represents an interesting deep learning topic of research of the future.
To conclude, in the last years various medical image segmentation tasks have been tackled with different deep learning approaches. The ones receiving as input the whole image or a large subregion have been prevailing over the "sliding window" methods, but both can be improved and not always designing a custom architecture for the specific task has turned to be the only optimal solution.

Chapter 4

Bone structure labeling in CT scans

4.1 Computed Tomography scans

Computed tomography (CT) is a medical imaging technique that uses a combination of X-rays to produce cross-sectional images of the body of the patient. Each image represents a slice of the patient and is obtained with a reconstruction algorithm. This technique is a valuable tool for the physician with multiple uses: diagnosing and analyzing diseases or abnormalities as tumours or abscesses, planning and guiding interventional or therapeutic procedures, monitoring the effectiveness of a therapy and many more.
Current CT scanners use an X-Ray source that rotates continuously in a helical path around the patient placed in the gantry [25]. The patient is then smoothly moved forward to analyze different part of the body. Data acquisition continues in this way and a whole volume is analyzed. This allows an easier reconstruction and lower dose compared to previous scanners which acquired only transversal images. The output of this process is a 3D map where each voxel value indicates the mean attenuation coefficient of the tissue at that location. Those values are generally converted in the Hounsfield unit (HU), a quantitative and more descriptive scale. The following linear transformation is applied [25]:

$$HU = K * \frac{\mu - \mu_{water}}{\mu_{water}} \tag{4.1}$$

where K is an integer constant, μ is the attenuation coefficient measured and μ_{water} is the coefficient of the water, taken as reference. K is generally considered as 1000. Therefore the CT number of a voxel containing water is 0 and the one of air (which has a $\mu = 0$) is -1000. Bones, which have a density approximately two times greater then the one of water, show CT numbers around 1000.
A common CT scan resolution is 512x512 pixels with a depth of some mm, depending on the scanner and on the size of the volume analyzed.

4.2 Bone segmentation

The segmentation of bones from CT images is a process of absolute importance for many medical tasks. It has several applications related to image-based computer assisted orthopedic surgery, it is fundamental for locating fractures and diagnosing bone diseases and in general it helps the radiologist in the medical-decision process. They also offer some stable references for analyzing and segmenting other parts of the body as the organs. For

these reasons an algorithm which performs an automated segmentation has been searched for a long time. Despite a considerable research, however, current solutions do not offer a robust segmentation to image inhomogeneities or noise and often require an algorithm initialization or parameters setting. This can be a relevant drawback in a clinical environment where a method to simplify the radiologist work is required. Furthermore, most of these approaches are not able to distinguish different bone structures and divide them into classes. They perform segmentation only of a limited number of bones structures and their lack of generalization is also a problem.
The bones are challenging to segment because of various reasons. First of all, the osseous tissue does not always appear easily distinguishable from soft tissue regions in CT images. There is a significant overlap between the HU values of the bones and several surrounding tissues, so a segmentation based only on the intensity value would not be accurate. Secondly, the bones structures themselves do not have uniform density properties and therefore the HU inside a single bone can be quite different. Complicating matters further, some diseases (such as Osteoarthritis, Rheumatoid, Arthritis, Osteoporosis) may alter the bone density in different areas. Finally, the limited resolution and low quality of CT scans can represent a problem. Adjacent bones often appear as being in direct contact and the presence of noise or artifacts affects the automatic segmentation performance.

4.3 Related work

The main approaches of bone segmentation in CT images have been classified in four categories [26]: intensity-based, edge-based, region-based and deformable. An additional technique that has recently achieved surprising results is the atlas based segmentation.
Intensity-based techniques are the most straightforward segmentation methods, based on local or global thresholding as explained in the previous chapter. They are usually simple to implement, but their main disadvantage is the lack of adaptability, as they require the regions to be segmented to have similar intensity values. This drawback can be partially overcome applying an adaptive threshold [27]. In this study published in 2010, it is proposed an automatic 3D adaptive thresholding method for bone segmentation in CT scans. It achieves good results and requires on average less than 10s per 2D slice. In more recent studies thresholding is often an important step of the pre-processing pipeline of the segmentation algorithm.
Edge-based methods take advantage of various edge detection operators to find the boundaries of the bones and segment them. This study [28], for example, achieved a good segmentation of the proximal femur, the knee and the skull in CT scans with different edge detection techniques. Furthermore, it only needs a very simple initialization. In order to be applied to other anatomic location, however, it requires a site specific separation of individual bones. In general those edge-based techniques are very sensible to noise and not always achieve optimal results.
Region-based approaches subdivide an image following a selected homogeneity criterion. In the previous chapter a few more details are explained.
Finally, by deformable techniques generally researchers refer to models that have active contours or surfaces. During the segmentation process those curves change under the influence of different forces and the model remains smooth. Those methods are the most investigated in the last years for bone segmentation in CT scans and several studies have achieved satisfying results [29]. However they present several drawbacks as well, as an excessive sensibility to the initial conditions.
Hybrid methods that use multiple techniques mentioned above are also been investigated

with mixed results. In [30] for example, the authors have proposed an algorithm which consists in a pre-processing followed by an image contouring and label filtering.
Atlas based methods represent a quite novel approach to image segmentation. They rely on a dataset of expert labeled images and aim at applying this knowledge to new, unseen data. In order to segment a new image, those methods compute a transformation that registers the atlas to the image analyzed (i.e establishes a point-to-point correspondence). The mask is then deformed from the atlas onto the patient image to segment it [31]. Among the previously described methods, atlas based techniques are the ones that have recently achieved greater accuracy and robustness in different segmentation tasks. It is worth mentioning this study [32] published in 2017 about human skeleton segmentation in CT images. It proposes a novel atlas and articulated registration approach for the segmentation of individual bones. In particular, it is able to segment 62 major bones, including 24 vertebrae and 24 ribs with high accuracy. An overall average Dice coefficient of 0.90 is reached, improving the results obtained by several previous studies.
In addition, deep learning based methods will surely play a major role in bone segmentation in the next years. Segmentation is already the most common subject of papers applying deep learning to medical imaging [2]. Recent studies have described successful methods of CNN-based automatic segmentation of knee cartilage and proximal femur [33, 34] in MRI scans. In this last study cited a CNN with a U-Net inspired architecture performs an automatic segmentation of the proximal femur from MR images. The network implementation is similar to the one proposed in this thesis, but its output is a two channels feature map for a binary segmentation of proximal femur and background. The approach described is interesting as it requires only a limited post-processing and achieves with the best model a Dice coefficient of 0.95. Even if it does not regard CT images this is an example of how bone segmentation with CNNs is feasible. As it does not require any hand-crafted features and learns the informative representations of the data automatically, it is a quite different method from previous approaches. In the next years there will be surely several other studies on the topic.
However, to the best of my knowledge, this thesis proposes the first approach of a CNN applied to the segmentation of bones structures in CT scans.

Chapter 5

Deep learning framework and computational tools

The whole code of this thesis is implemented in Python 2.7.12 and it makes use of a few libraries on top of it. The deep learning framework chosen to work with is Keras [35], which runs on TensorFlow [36] with a GPU implementation. This chapter begins with an overview of those libraries, it follows a brief description of the GPU implementation and finally some performance parameters are analyzed.

5.1 Keras

Keras [35] is a minimalistic deep learning framework written in Python. It runs either on top of TensorFlow or Theano and is considered to be of a higher level of abstraction compared to those. However, it is so without sacrificing the benefits of full control through the underlying framework. Its Application Programming Interface (API) is very intuitive for a Python user and allows to implement convolutional or recurrent neural networks in a limited number of lines. The main characteristics of Keras are its user friendliness, modularity, easy extensibility and the fact that works with Python.

- User friendliness. As mentioned above, a great advantage of Keras is its simplicity which allows to implement complex neural network in a relative small amount of time. In addition to that, it is also well documented and has a very active and helpful community.
- Modularity. In Keras pretty much everything is represented by modules and the user is allowed to combine them in any way he likes. A model is then made up of a sequence or a graph of these modules.
- Easy extensibility. By that it is meant that the modules can be easily modified or extended and new ones can be created. The full code of Keras is available on its GitHub repository.
- Work with Python. The modules are all implemented in Python, which makes it easier to understand, install, debug or extend the code. Keras is compatible with Python 2.7-3.5.

The two main drawbacks are the fact that for complicated tasks Keras does not allow the same functionalities as lower level languages as Theano or TensorFlow and it does not officially provide yet support for multi-GPUs training.
In this work Keras 2.0.2 was used with GPU support and TensorFlow as backend.

5.2 TensorFlow

TensorFlow [36] is an open-source machine learning library developed by the Google Brain Team and suitable for a large range of tasks. It is widely employed for implementing deep neural network models and conducting research in various fields, including medical imaging, speech recognition, natural language processing, computer vision, robotics, game playing, business analysis. State-of-art results have been achieved with this framework in several studies. Moreover, TensorFlow is currently used by some of the most well-known companies in the world such as Google, Dropbox, Ibm, Ebay, Airbus, Uber, Twitter and many more. The core of TensorFlow is mainly written in C++ and CUDA, but Python is the first supported language for creating and training models. As the first version of the software was released on November 9, 2015, it is a relatively new project, but has already a very active community and it is arguably considered the most popular deep learning framework.
In TensorFlow the operations of a model are described by a computation graph, through which the data flows in multidimensional arrays, referred to as tensors. The graph is made of a series of nodes, where each one represents a single operation and has zero or more input and output. The nodes are linked together following the sequential order of the operations to be computed. A simple example of a computational graph is shown in Figure 5.1. Here, given an input x, a convolutional layer with weights W and bias b is implemented, followed by a ReLU activation function, some other possible operations and producing an output C.

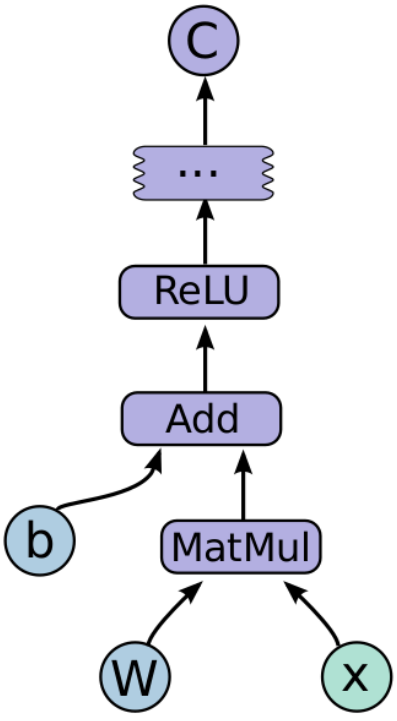


Figure 5.1: Example of a simple TensorFlow computation graph [36].

In order to execute graphs a session must be created. A session allocates resources on the selected devices and provides methods to run the computations. TensorFlow can indeed run on multiple CPUs or GPUs and is available on various platforms as Linux, macOS, Windows and Android. By default TensorFlow allocates roughly all the GPU memory

available for the running process. However, with a few lines of code it is possible to select the exact fraction of overall amount of memory that TensorFlow should map.
Together with TensorFlow it is automatically installed a very interesting neural networks visualization toll called TensorBoard. TensorBoard reads summary files from TensorFlow and helps the user visualizing, understanding and debugging its graphs. Moreover, it shows the plots of quantitative metrics of the models and additional data that gives an insight into the training process of the model. In Figure 5.2 it is reported a screenshot of a TensorBoard window during the training of the network proposed in this thesis. The tab "SCALARS" shows plots of the loss, metrics or other selected variables versus the number of epochs or training time; in this case the dice coefficient and the loss are chosen. Under the tabs "IMAGES" and "AUDIO" the input images or audio files can be analyzed. "GRAPHS" shows the flow graph of the model with the possibility to zoom in and have a closer look at each single node and its connections. "DISTRIBUTIONS" and "HISTOGRAM" can be used to have a better understanding of how some parameters, as for example the weights, change over the training time. Finally "EMBEDDINGS" presents an interactive visualization of high-dimensional data like embeddings. Several of these features have been used during this thesis to have a better understanding of the TensorFlow graph generated and of the training process.

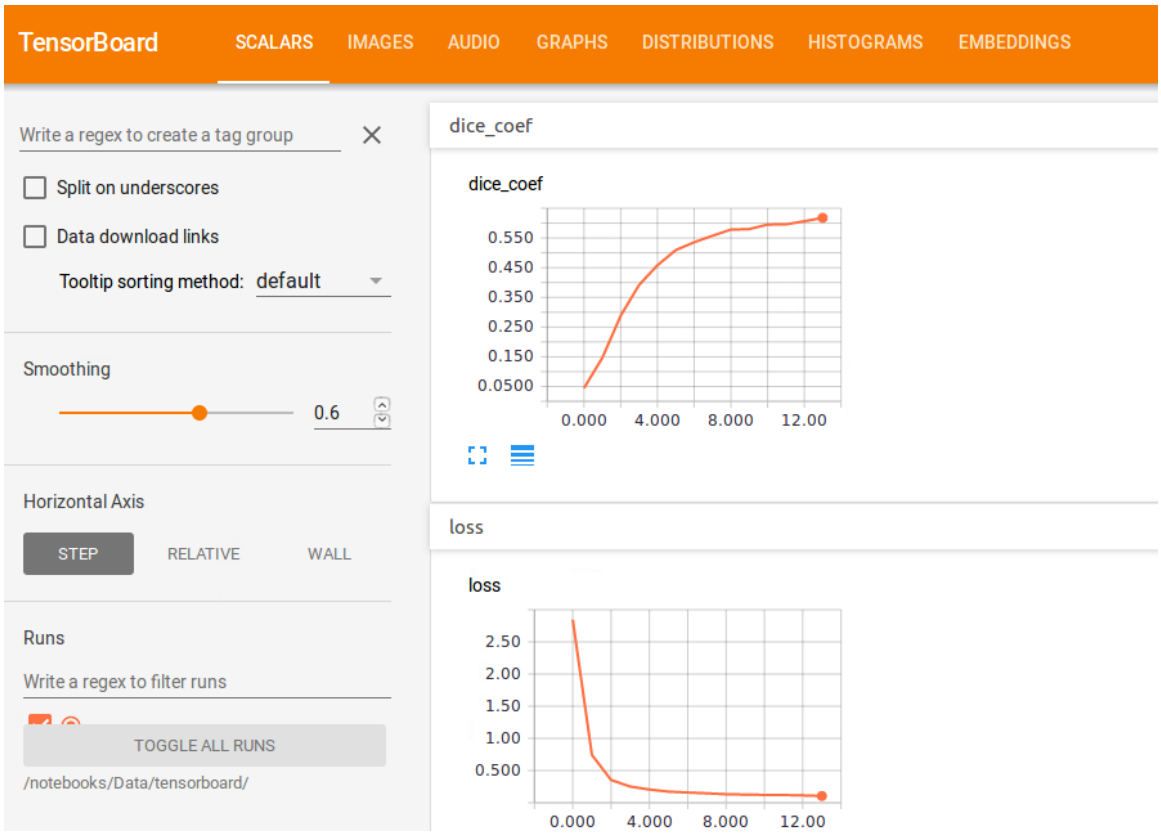


Figure 5.2: Example of a TensorBoard window taken from this study.

5.3 GPU computing

A fundamental factor that has allowed the expansion of the deep learning field in the last years is the improvement of the GPUs performances. In Figure 5.3 is illustrated the tremendous increase in GPU computational power and its comparison with standard Intel CPU. GPUs stands for Graphics Processing Units and they were initially designed to accelerate graphics computations. Nowadays they are used as well to parallelize and therefore speed up computations of algorithms as they are much more efficient than general-purpose CPUs. The main reason behind this fact is that GPUs have a larger number of transistors dedicated to data processing, instead that data caching and flow control. This makes them much faster than CPUs for tasks where computations can be executed in parallel and a relative small amount of memory operations is required. Modern GPUs are widely applied for deep learning computations and have allowed in the last years to dramatically reduce the training time of complex models.

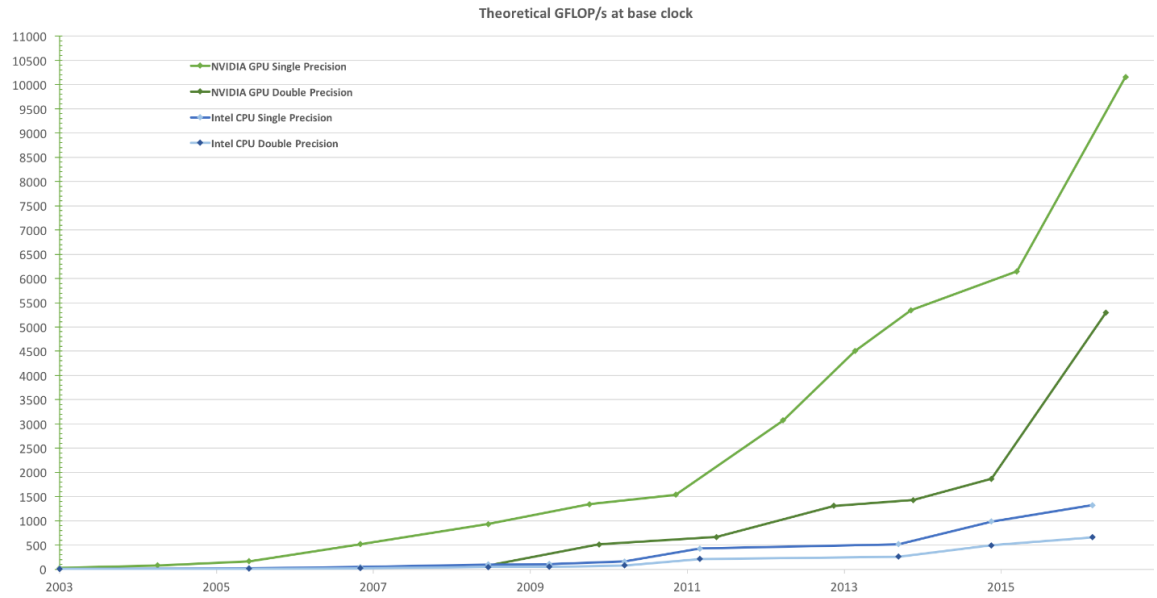


Figure 5.3: Floating-Point Operations per second for the CPU and GPU [37].

NVIDIA is a GPUs manufacturing company that in the last two decades has hugely contributed to their increase in computational power. In November 2006 it firstly introduced CUDA, a parallel computing platform that makes general purpose processing possible in CUDA-enable GPUs. CUDA and in particular CUDA Deep Neural Network library (cuDNN) accelerates TensorFlow and Keras, the deep learning frameworks used in this thesis.

5.4 Packaging and distribution

The code of the thesis is written in Python 2.7.12 and based on Keras 2.0.2 and TensorFlow 1.0. In addition to that, the following Python libraries are used:

- numpy>=1.12.1 Package for numerical computations and analysis.
- matplotlib>=1.5.3 Package for plotting and visualization.
- SimpleITK>= 1.0.0 Package for image analysis.

5.5 GPU

The training and test code of the network is run on the following GPU:

- NVIDIA GeForce GTX 1080:
  - CUDA Cores: 2560
  - Total Memory: 8192 MB
  - Architecture: Pascal

It takes about 23.5 minutes to complete a training epoch with the whole training dataset made of 15653 image samples, divided in batches of size 8. The testing phase, however, is much quicker. To make predictions of a testing dataset of 811 slices, for example, it requires approximately 24 seconds.

Chapter 6

Proposed method

This approach proposes a CNN, inspired by the U-Net [5], for automatic segmentation of bones in abdominal CT scans. The network is mainly based on 2D convolutional layers with a 3x3 kernel size and accepts as input full-resolution 512x512 voxels images. It is made of a compression path, where the image size is progressively reduced, followed by an expanding one, which allows to have a segmentation mask in output with the same sizes of the input image. Its aim is the segmentation of bones and also their classification in six different classes: femoral bones, hips, sacrum, spine, hips and ribs. For this purpose, the last convolutional layer of the network is modified, compared to the original U-Net architecture, in order to have seven features maps: one for each bone class and one for the background. The bone mask is then obtained applying a threshold to those probabilities maps. A major challenge of this work is the fact that the bones analyzed are rare compared to all the other structures of the body and especially to the background present in a scan. To overcome this problem each output map is multiplied by an appropriate weight map in the computation of the cross-entropy loss. In this way the network is forced to learn the most and least represented classes in the same way. The data used are full-resolution 512x512 voxels transversal slices of abdominal CT scans. An example of these scans displayed in 3D is shown in Figure 6.2 and in Figure 6.3 its corresponding bone mask can be observed. The network, however, is trained only on 2D transversal slices, as shown in Figure 6.1, and the 3D visualization has the only purpose of a visual inspection of the whole bone structures. This turned out to be very useful to spot major mistakes that could not be noticed observing just the single 2D sections.
In this chapter it is firstly described the data preprocessing, secondly the data augmentation applied to cope with the limited number of annotated data and finally the network architecture is analyzed in details.

6.1 Data preprocessing

In a segmentation task tackled with deep learning techniques it is generally challenging to obtain a sufficient training set of annotated data. The process of labeling, even when it is done by experts, is time-consuming and its results depend on the operator ability. In this work, in order to obtain the ground true images a series of operations were applied to the original scans. For this purpose, a semi-automatic preprocessing network was elaborated in MeVisLab [38], a framework for image processing research developed by MeVis Medical Solutions Ag in close cooperation with the research institute Fraunhofer MEVIS. The original scans were preprocessed with this network and then manually checked slice by slice to correct the frequent mistakes.

6.1.1 MeVisLab network

The network developed in MeVisLab includes various thresholding and image processing operations in order to extract only the bones from an abdominal CT scan. In Figure 6.1 on the left side three axial sections of abdominal CT scans are shown and on the right side the final ground truth images. The bones, especially in their boundaries, have generally a lighter color than other structures and therefore a basic discrimination method can be based on the gray value.
The MeVisLab network used consists of a series of successive steps which are now briefly explained. Firstly, a few global and Otsu thresholding are applied, followed by a convolution filter that smooths the image and partially removes the noise previously generated. Secondly, a connected components analysis is applied. Since the bones are mainly connected structures, this allows to discard some small regions which were erroneously found in the previous step. Thirdly, a convolution filter is applied again to smooth the image. Finally, through a ROI selection tool the operator assigns a different label to each one of the six classes. The classes chosen are the following: femoral bones (which refer to the femoral head, femoral neck and upper part of the femur), hips, sacrum, spine, ribs and sternum. The clavicles and the scapulae, which were present only in certain scans, are not considered. Between all the classes, the ribs and the spine turn out to be the most challenging ones to be identified correctly. The former due to their small and repeated shape when seen in the transversal view and the latter mainly because of the spinal canal, often identified erroneously as bone structure. As several other mistakes are found after the semi-automatic segmentation through the MeVisLab network, a manual labeling is then required.

6.1.2 Manual data labeling

After the semi-automatic preprocessing the scans are checked slice by slices manually. Looking at a medical atlas the mistakes are corrected. This process is time-consuming, but at the same time essential for the rest of the work. The manual labeling is done through MeVisLab mainly with the CSI tool, that allows to select or deselect a region in a 2D transversal view of an image. 3D views are also possible and have been of great help, especially to identify noisy voxels.
Here is a brief overview of the mistakes encountered more often for each bone class during this process:

- **Spine**. The spine is probably the most challenging bone class to label due to its complex shape. The semi-automatic segmentation often fills the spinal canal, recognizing it wrongly as bone, or fails at labeling correctly the borders with the ribs. Moreover, the slices of overlapping with the sacrum represent another critical region that has to be carefully checked.

- **Hips**. The hips are initially segmented quite well. The borders with the femoral heads and, due to their thin structure, some inner regions present the most frequent mistakes.

- **Sternum**. This long and flat bone has only very small regions in a transversal view image. The semi-automatic segmentation often fails at recognizing correctly its extremities and the borders with the cartilage that connects it to the ribs.

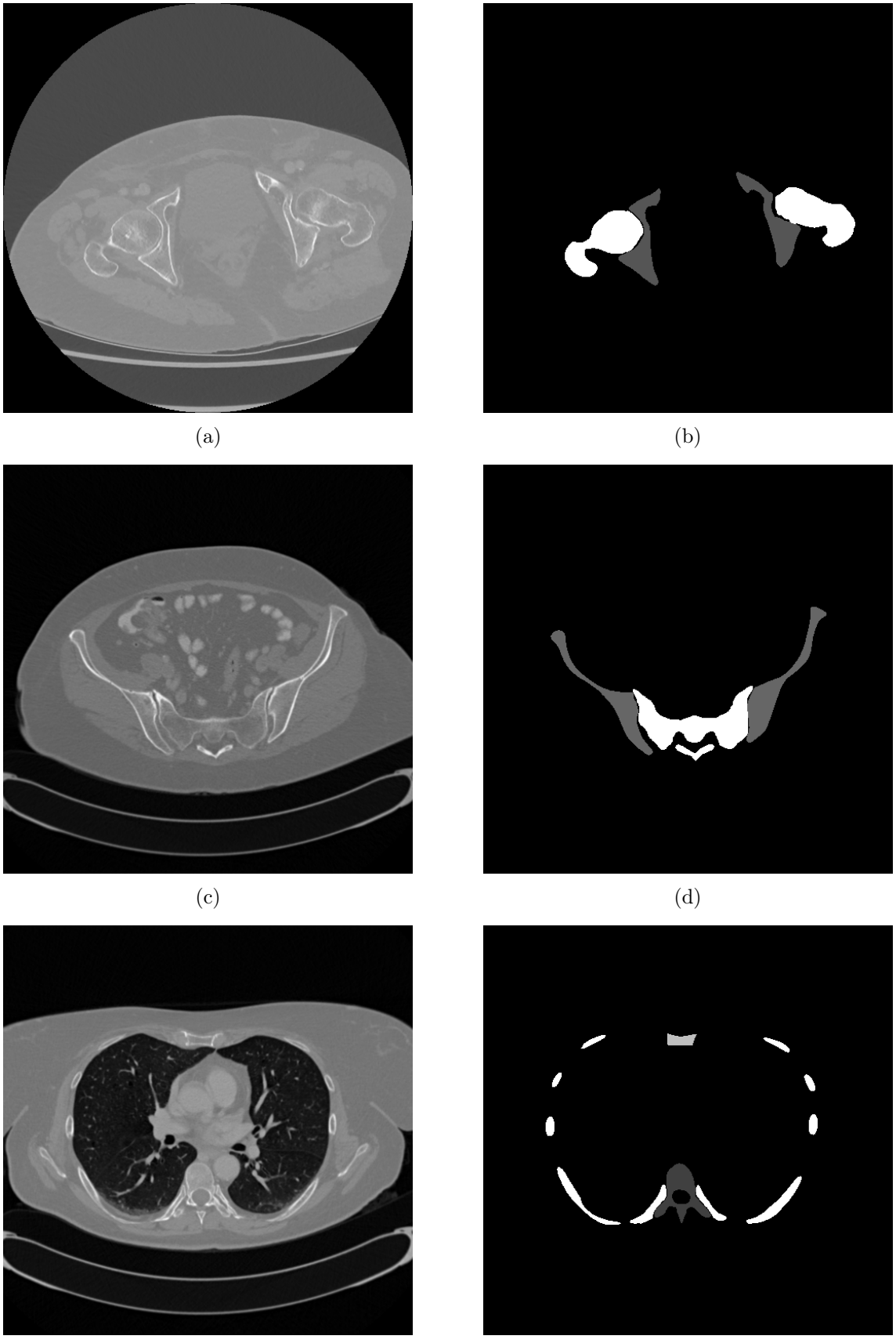


Figure 6.1: Three 2D images used for training. On the left side ((a),(c),(e)) the original images and on the right side ((b),(d),(f)) the respective bone masks.

- **Ribs** This class is also not straightforward to segment due to its shape in a transversal view. It presents several small structures all separated from one another, as it can be seen in Figure 6.1 (e),(f). In this case a 3D view is very useful to check if the regions segmented are then all connected in the three-dimensional space. Avoiding the adjacent cartilage represents another challenge.

- **Sacrum**. The sacrum is a challenging class to segment due to its complex structure. Starting from the top, the first transversal slices present an overlapping with the spine. Furthermore, in the inner structure the four pairs of holes, known are sacral foramica, are often identified wrongly as bone.

- **Femoral bones**. Regarding the femoral bones the main issues are due to the inner parts of those bones, which result to be a bit darker than the borders. The intersection with the hips needs also to be checked carefully.

Furthermore, some initial scans included also the clavicles and the scapulae. As those bones are not considered in this segmentation task, they had to be removed in the manual post-processing. In Figure 6.1 three 2D original images are reported with their respective masks. Those masks were obtained after the manual data labeling process. In Figure 6.3 a 3D view of a labeled scan used for training is shown. Each bone class is presented with a different color for a better visualization, whereas in the labeling process they are assigned a gray value from 1 until 6, leaving the value 0 for the background.

6.2 Data augmentation

Data Augmentation is the process of generating additional viable training data applying some transformations to the initial training dataset. It is considered a regularization approach and it has been successfully applied in various of the most influential publications in the deep learning field over the last years [39],[40]. The main transformations used are generally horizontal and vertical translations, rotations, reflections, patch extraction, gray or color value variations and elastic deformations. Data augmentation has been used extensively in previous segmentation works of medical images [5],[17],[14] in order to achieve a better generalization starting with only few annotated samples.
In this work data augmentation was required because of the limited number of labeled scans. Indeed, using only the original training dataset, the problem of overfitting arises and a lower accuracy in the validation dataset is observed. The random transformations applied are vertical and horizontal shift, rotation, zoom and vertical reflection. In particular each image is shifted vertically and horizontally by a random number from 0 to the 5% of the image sizes. Then it is rotated by a random degree in the range between 0 and 5 and it is randomly zoomed in a range which goes up to the 95% and 90% of its horizontal and vertical size respectively. Finally a random number between 0 and 1 is generated and if this number is greater than 0.5 the image is reflected along its vertical axis. For this purpose the ImageDataGenerator function of Keras [35] was modified in order to apply the same transformation to the sample and the mask simultaneously. The Keras function generates batches of tensor image data with real-time data augmentation. In the network implementation the data is augmented on the CPU, while the GPU is training on the previously generated batches. There are always not more than ten batches of transformed images saved in the disk. Therefore the data augmentation process has a very small computational cost and does not increase the training time considerably. Furthermore, as

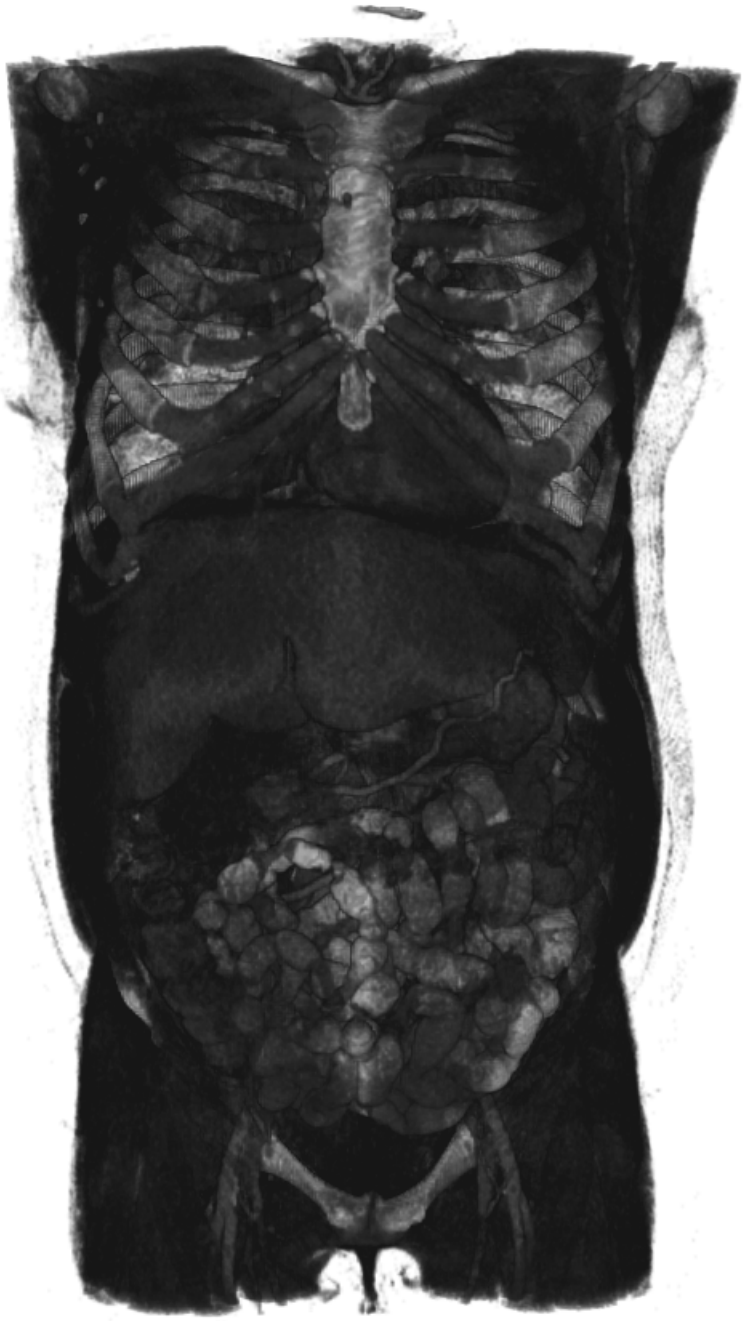


Figure 6.2: 3D view of an abdominal CT scan. The LUT parameters of the MeVisLab viewer are adjusted to show some of the internal structures of the body and not only the skin. The corresponding ground truth is shown in Figure 6.3

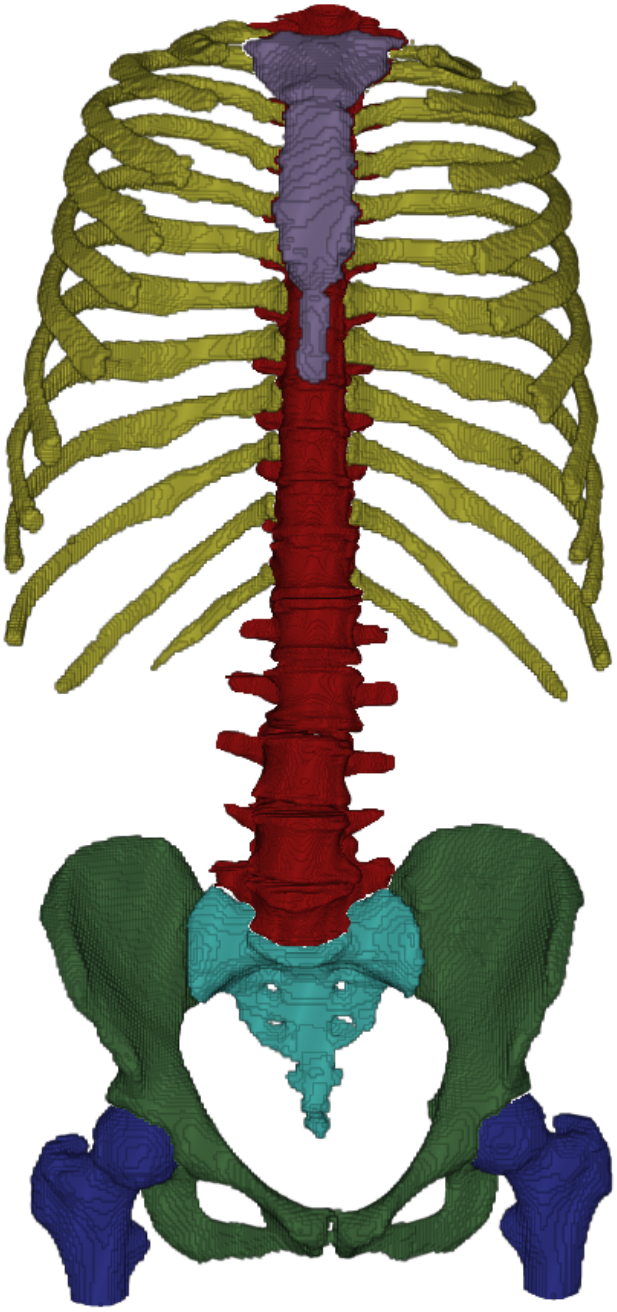


Figure 6.3: 3D view of the ground truth of the scan shown in Figure 6.2. The 6 bone classes segmented are shown with different colors: blue for the femoral bones, green for the hips, turquoise for the sacrum, red for the spine, yellow for the ribs and violet for the sternum.

the transformations are applied randomly, the network sees different data each epoch and this helps reducing the difference in performance between the training and validation or testing datasets.

6.3 Network implementation

The network architecture is strongly based on the U-Net [5]. It presents a contracting path, where the resolution of the image decreases and the number of feature channels is increasing, and an expanding path that works the other way around. The output of the network is a 7 channels map which represents for each pixel of the image a class probability vector. The CNN can therefore be trained end-to-end and only a probability thresholding is required to obtain the final segmentation.
In Figure 6.4 the architecture of the network is shown. The blue boxes represent the feature maps, whereas the red and yellow one indicate respectively the input image and the 7 channels output map. There are five depth levels and skipped connections between the ones on the same level, as in the original U-Net. However, the first level has only 16 feature channels and the following ones double the channels until the fifth which presents 256 maps. This significant reduction of parameters was necessary due to the large initial sizes of the images (512x512 pixels) which limits the GPU memory available to store the model. In the contracting path each level has two 3x3 convolutional layers, each one followed by a batch normalization layer before its ReLu activation function. A 2x2 max pooling operation is then applied before the next level. The expanding path has a similar structure, but an up-sampling ("deconvolution") that doubles the features maps sizes is applied instead of max pooling and the concatenation with the features maps of the corresponding contracting level takes place. In both paths, between the two convolutional layers of each level, dropout is applied. The last layer is a 1x1 convolution with 7 feature channels, one for each class plus one for the background, and a linear activation function. Finally, the features maps are reshaped in a 2D vector and softmax is applied. This is necessary as at the time of the network implementation Keras did not support softmax on 3D data. Finally, the network is trained with the back-propagation algorithm and Adam as optimizer.

6.3.1 Loss function

The softmax function is a generalized logistic function widely applied to the output of a classifier in CNNs. It converts a vector of activations values to a vector of real values in the range $[0, 1]$ which all sum up to 1. Therefore they represent a valid probability distribution [1]. The softmax is defined as follows:

$$p_k(x) = \frac{e^{a_k(x)}}{\sum_{n=1}^{N} e^{a_n(x)}} \tag{6.1}$$

where x refers to the pixel considered, $a_k(x)$ and $a_n(x)$ are the activation values of channel k and n for that pixel, N is the number of classes and p_k is the softmax of class k. As in the original U-Net [5], after applying a pixel-wise softmax, a weighted cross entropy is computed as loss function. Cross entropy is a common choice for the cost function in

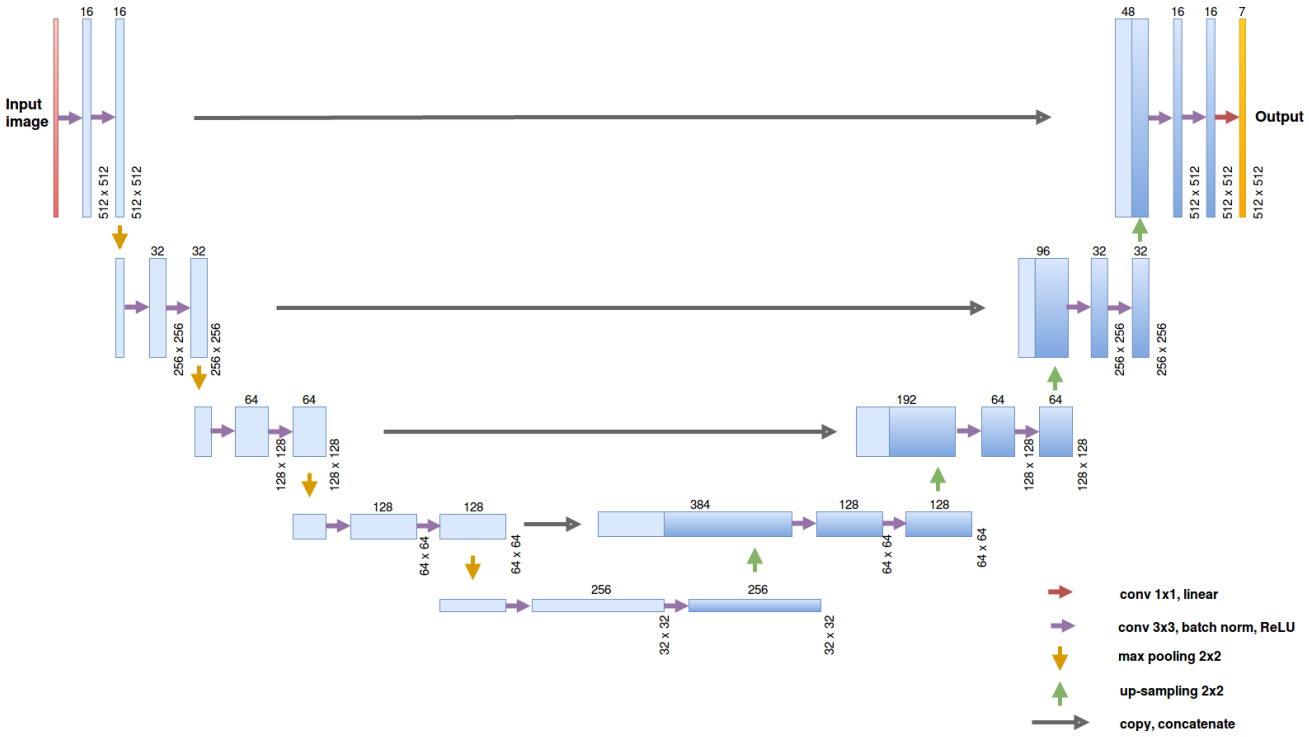


Figure 6.4: Network architecture. The feature maps of the contracting path are represented in light blue, whereas the ones of the expanding path in darker blue. On top of each box the number of channels is denoted and the size is shown on the bottom right corner.

CNNs; for a multi-class task it is known as categorical cross entropy. Cross entropy gives a measure of the error of the predicted probabilities compared to the true labels and it is generally preferred to mean square error for classification tasks. It is defined for discrete probability distributions as follows:

$$H(p,q) = -\sum_{x} q(y)\log(p(x)) \tag{6.2}$$

where q(y) is the true probability distribution and p(y) is the predicted one. In this work the cross entropy is used with the so-called "hard" labels (they are either 1 or 0) and it is multiplied by a weight map with the same sizes of the image. Considering M pixels belonging to N different classes, the total loss based on a weighted cross-entropy is computed as follows:

$$L(l,q) = -\frac{1}{M}\sum_{x=1}^{M} w_n(x)\left[\sum_{n=1}^{N} l_n(x)\log(p_n(x))\right] \tag{6.3}$$

where $w(x)$ refers to the weight map and for each pixel x l_n is the true label of class n and p_n is the predicted probability for that class. In other words, the loss is defined as the mean over the total number of pixels of the weighted cross entropy.

6.3.2 Weight map

In this work the term weight map refers to a matrix, with the same sizes of the input image, where each element represents a specific weight. The weight mask is introduced in the computation of the loss function in order to cope with the problem of unbalanced classes. Several papers tackling this problem have shown that the weight map is an appropriate solution [5, 41]. In this task the background is approximately two orders of magnitude more frequent than the spine, which is the most frequent bone class. In Table 6.1 are shown the number of labeled pixels of the training dataset for each class and the relative percentage over the total number. There is a relevant difference between the background and all bone classes and between those the sternum is by far the least represented.

Class	Number of pixel	Percentage
Background	4012178597	97.7783%
Spine	27914929	0.6803%
Hips	25842329	0.6298%
Sternum	2449034	0.0597%
Ribs	14657864	0.3571%
Sacrum	8403887	0.2048%
Femoral bones	11893392	0.2898%

Table 6.1: Number of labeled pixel for each class in the training dataset and relative percentage over the total number of pixels. The training dataset includes 15653 scans of 512x512 pixels each.

In order to deal with with those significant differences, the median frequency balance [42] is applied. This means that for each class a weight α_c is computed as follows:

$$\alpha_c = \frac{freq_m}{freq(c)} \tag{6.4}$$

where $freq(c)$ is the frequency of the class, equal to the number of pixels of that class divided by the total number of pixels in images where c is present, and $freq_m$ is the median of the frequencies of all classes. In this way the least represented classes are assigned bigger weights (greater than 1) whereas the most present ones have weights smaller than 1 (see Table 6.2).
During training the weight map is automatically computed for each batch of images, assigning to each pixel the weight of its true labeled class. Finally, as it can be seen in equation (6.3), an element-wise multiplication is performed between the weight map and the value of the cross entropy function of each pixel. Erroneously predicting a pixel label of a less frequent class adds therefore a bigger term in the loss function compared to one belonging to a more frequent class.

Class	α_c
Background	0.0098
Spine	1.0161
Hips	0.4651
Sternum	4.0314
Ribs	1.3401
Sacrum	1.0000
Femoral bones	0.5364

Table 6.2: Weights relative to each class used during training.

6.3.3 Regularization techniques

Regularization techniques are represented by all the strategies that aim at avoiding overfitting and reducing the error on the test dataset. Regarding deep learning, most of them are based on regularizing estimators [1], but others are being investigated as well. An example is data augmentation, described in Paragraph 6.2 and extensively used in this thesis. Two other regularization techniques are applied to this network: dropout and max norm constrain.
The main idea of dropout is to randomly remove some hidden units and train only on the ones left; an example is shown in Figure 6.5. During testing all neurons are active again. As a consequence different networks are created for training and this can be seen as a form of ensemble learning. Dropout has proved to be very effective against overfitting [43] and is today one of the most used regularization techniques. In the network proposed, dropout is applied at each depth level, between the two convolutional layers with a drop probability of 0.4.
Max norm constrain is a different technique that selects an upper bound for the magnitude of the weight vector of each neuron. Forcing the weights not to exceed a certain value allows also bigger learning rates to be used. It has been shown that combining dropout with max norm constrain can cause improvements in the network performance [43]. In this work max norm constrain is applied to each convolutional layer with a magnitude upper limit of 4.

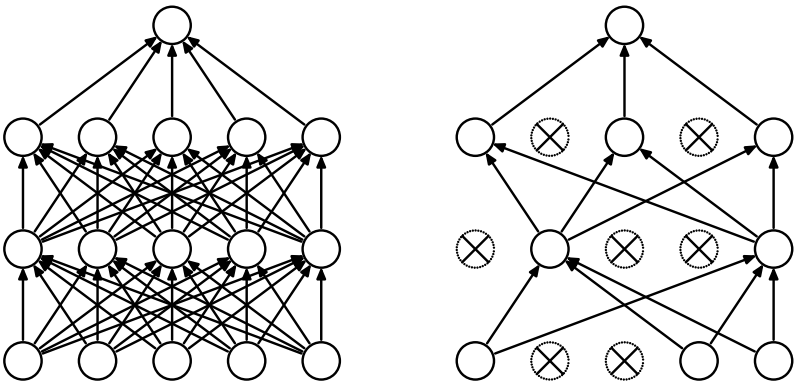


Figure 6.5: On the left a simple neural network with two fully connected layers and on the right the same network after dropout is applied. Figure from the dropout paper [43].

6.3.4 Adam optimizer

A significant difference of this work compared to the original U-Net paper is the use of Adaptive Moment Estimation (Adam) [44] instead of stochastic gradient descent (SGD) as optimization algorithm. Adam turns out to work better for this type of problem and does not require the operator to tune the learning rate decay hyperparameter.
Adam is a gradient-based optimization method that automatically computes adaptive learning rates for individual parameters. As its name suggests it relies on the first and second moment estimations of the gradient of the objective function. Denoting with $g_t = \nabla_\theta f_t(\theta)$ the gradient of a differentiable function $f_t(\theta)$ with respect to its parameters θ, Adam computes at each time step t the exponential moving average of the first (m) and second moment (v) as follows:

$$m_t = \beta_1 * m_{t-1} + (1 - \beta_1) * g_t \tag{6.5}$$

$$v_t = \beta_2 * m_{t-1} + (1 - \beta_2) * g_t^2 \tag{6.6}$$

where β_1 and β_2 are the hyperparameters that control the exponential decay. They generally have values close to 1 and in the original paper [44] they are set to 0.9 and 0.999 respectively. As the first and second moment are initialized as zeros, a bias correction turns out to be necessary. Therefore bias-corrected moments $\hat{m}_t$ and $\hat{v}_t$ are computed as follows:

$$\hat{m}_t = \frac{m_t}{(1 - \beta_1^t)} \tag{6.7}$$

$$\hat{v}_t = \frac{v_t}{(1 - \beta_2^t)} \tag{6.8}$$

Finally at the end of each time step t the parameters of the stochastic objective function are updated:

$$\theta_t = \theta_{t-1} - \frac{\alpha * \hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{6.9}$$

where α is the learning rate and ϵ a very small number to avoid division by 0.
The authors of the original paper demonstrated that Adam is more efficient in practice than other adaptive rate methods as Adadelta and RMSprop and therefore it was chosen as optimization algorithm for this task.

Chapter 7

Training and results

In a machine learning problem the tuning of the hyperparameters is a fundamental step preceding the learning process. In this work the learning rate, batch size, dropout rate and other parameters are accurately adjusted in order to allow the network learning efficiently. Another important point of this work is the choice of the threshold values, applied for each bone class to the output masks generated by the network. Both those issues have been addressed and their solutions are explained in the following paragraphs.
This chapter deals mainly with the training of the network previously described and with its evaluation. Firstly, the Dice coefficient, precision and recall, the metrics chosen to evaluate the network performance, are explained. Secondly, the whole training process is illustrated, with a particular emphasis on the training and validation datasets and their evaluation. Thirdly, the results obtained testing the model on a previously unseen dataset, labeled by experts, are shown. Finally, gaussian noise is added to the images and the model robustness to it is analyzed.

7.1 Metrics

The metrics considered to monitor the training and evaluate the test dataset are the Dice coefficient, precision and recall.
Precision and recall (also known as sensitivity) are defined as follows [45]:

$$precision = \frac{TP}{TP + FP} \qquad recall = \frac{TP}{TP + FN} \tag{7.1}$$

where TP, FP and FN are respectively the number of true positives, false positives and false negative. Therefore, on one hand, precision is the fraction of correctly segmented pixels over the total number of pixels identified as positive by the model. On the other hand, recall indicates the portion of ground truth pixels found by the method being evaluated.
The Dice coefficient, often referred to as overlap index, is a statistical parameter that shows the similarity between two samples. Its range of values is between 0 and 1, where 0 indicates that the samples do not overlap at all and 1 that they are completely similar. It is widely used in image segmentation tasks to compare the output of an algorithm and its relative ground truth image, known also as mask. In particular, regarding medical imaging, it is considered the most used metric to validate medical volume segmentations

[45]. Another common index in those tasks is the Jaccard, which is related to the Dice coefficient by the following relation:

$$JAC = \frac{DICE}{2 - DICE} \tag{7.2}$$

Therefore as those two metrics measure the same overlapping aspects, monitoring both of them does not provide any further information [45] and in this work only the Dice coefficient is considered. In a binary segmentation problem the Dice coefficient D is defined as follows:

$$D = \frac{2TP}{2TP + FP + FN} \tag{7.3}$$

where TP stands for true positive, FP false positive and FN false negative.
In this work it was monitored the Dice coefficient for each class, computed pixel by pixel, and also the mean of the Dice coefficients of all bone classes. The background is not considered in this last computation as due to the large classes unbalance its Dice coefficient easily reaches values close to 1. In the metric computation the images of the dataset considered are extracted randomly and then evaluated one by one; hence the batch size is equal to 1. This requires more time, but it is important in order to obtain consistent results that do not depend on how the batches are extracted.

7.2 Training and validation datasets

The training dataset is made up of 21 full-abdominal 3D scans, for a total of 15653 2D axial images with an in-plane resolution of 512x512 voxels. Every scan covers from the femoral necks until slightly above the first rib. For each sample there is the respective ground truth image, with the same size and labeled in six different bone classes plus the background, as explained in the previous chapter. Refer to Figure 6.1 to see three samples used for training. The volumetric scans have between 403 and 994 axial sections. Each section thickness is between 0.7 and 1.5 mm.
A single scan made of 426 2D axial images is used as validation dataset. Those sections are 1.0 mm thick. The validation scan is labeled in the same way of the training dataset.

7.3 Training

The CNN used is shown in figure 6.4 and its weights are initialized with the so-called Xavier method, which automatically scales them into an appropriate range to obtain an effective learning. The bias are initialized to 0, as there is not yet a complete agreement in literature on the effectiveness of initializing them with a small constant to avoid dead neurons [6]. Adam is the optimizing algorithm chosen, with an initial learning rate of 1e-4. Different learning rates have been tested during this study, with this one ending up to be the optimal choice. On one hand a larger learning rate allows the network to learn quicker, but reaching a lower Dice coefficient after the same number of epochs. On the

other hand a lower learning rate increases the training time without producing appreciable improvements. Due to the large memory size of the network, the batch size is limited to 8 samples. Smaller batch sizes increase the training time as well, without any benefit. The training lasts 150 epochs, with each epoch taking approximately 23.5 minutes. All those hyperparameter values used for the network initialization and during training are reported in Table 7.1.

Phase	Parameter	Value
Initialization	Weight	Xavier
	Bias	0
Training	Learning rate	1e-4
	Dropout rate	0.4
	Max norm	4
	Batch size	8
	Epochs	150

Table 7.1: Hyperparameters of the CNN.

During training data augmentation is performed and each image is also pre-processed in real-time, subtracting to each pixel value the image mean and then dividing the result for the standard deviation on the same image. In this way the data results to be zero-centered and normalized.
The softmax layer of the CNN produces an output which is a probability map, where the probabilities of all classes sum up to 1. In order to obtain the best segmentation an optimal threshold value for each bone class has to be selected. To do so, the precision-recall curves of the training samples for each bone class are analyzed. In a binary segmentation or classification task precision and recall are typically inversely related. Therefore analyzing the precision-recall curve helps to find a balance between the two that optimizes the results obtained. In Figure 7.1 is shown the precision-recall curve for each bone class, computed at threshold intervals of 0.005. There are evident differences between the curves. The femoral bones, for example, look like the easiest class to segment, whereas the ribs represent the one where more mistakes are expected.
The optimal threshold values were chosen as the ones that minimize the Euclidean distance to the point of maximum precision and recall, as done in [34]. In Table 7.2 the threshold values selected for each class are reported. It is worth pointing out that a lower threshold produces a more noisy mask, whereas a higher one causes the bones to be slightly under-segmented. Therefore the threshold values that maximize the Dice coefficient might not be always the best choice. In certain circumstances, especially if a post-processing algorithm to remove noisy pixels from the masks is applied, a slightly lower threshold is advisable.
The loss and metrics obtained on the validation dataset are monitored during training at the end of each epoch. A model in Keras has two different modes, one for training and one for testing. At testing time dropout and regularization techniques are turned off. Thus, as the validation dataset is evaluated in a testing regime, its performance can be slightly different. However for the whole training over-fitting has not been noticed, meaning that the data augmentation and the regularization methods applied were effective.
In Table 7.3 are shown the Dice coefficient values obtained on the training and validation datasets. Here and for the testing the best model based on validation after the 150 training epochs is used. As mentioned above, there are not large differences between the Dice coefficient of the training and validation dataset and therefore over-fitting is not taking

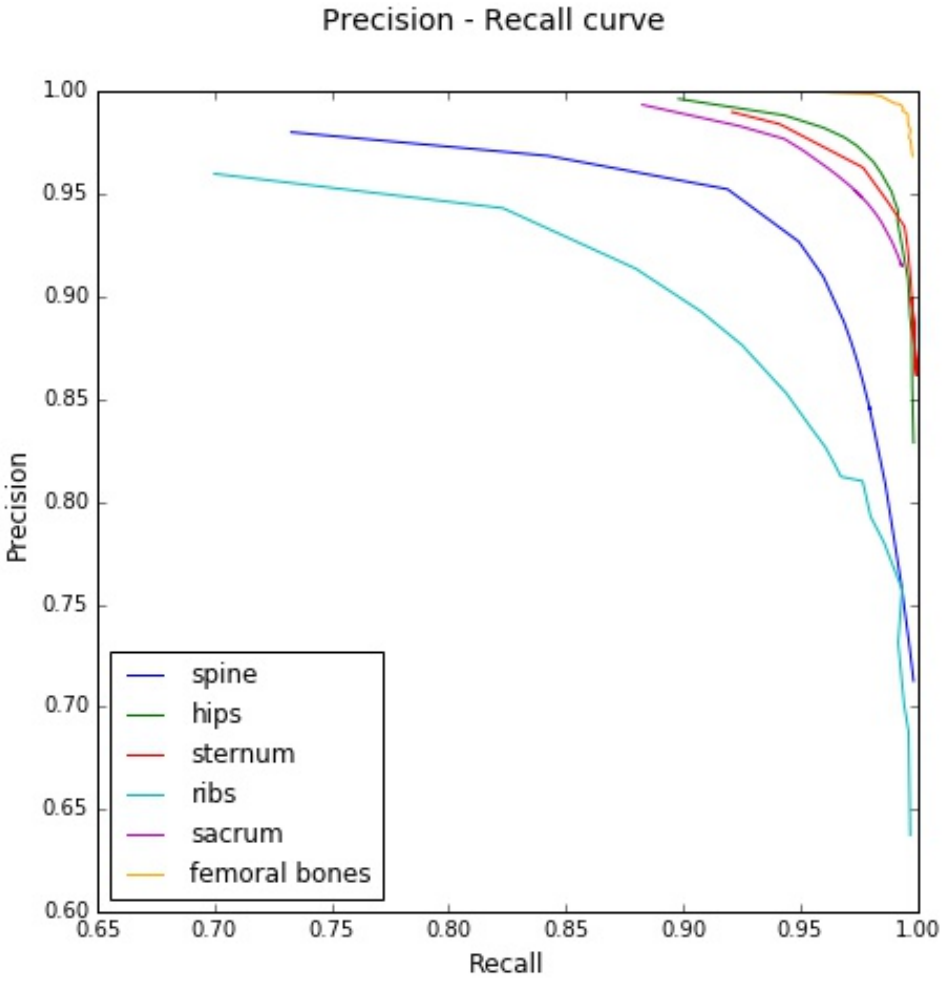


Figure 7.1: Precision-recall curve for each bone class.

Class	Threshold
Spine	0.980
Hips	0.980
Sternum	0.995
Ribs	0.990
Sacrum	0.945
Femoral bones	0.950

Table 7.2: Optimal threshold values selected for each class, analyzing the precision-recall curve of the training dataset.

place. Except for the ribs, on all other classes the model achieves a Dice coefficient of 0.90 or higher in both the training and validation dataset. The ribs represent definitely the most challenging class to segment, mainly due to their small and separated structure in the transversal slices. It is also worth noticing that there is quite a difference between their Dice coefficient in the training and validation dataset, 0.88025 and 0.84949 respectively. However, because of the limited number of slices of the validation dataset and since the other classes do not show the same behavior, once again it can be said that the network is not over-fitting the training data. As it can be inferred from Figure 7.1 as well, the femoral bones are the class better segmented, with a very high metric in both datasets.
In Table 7.4 and 7.5 are reported the precision and recall respectively for the training and validation dataset. There are not significant differences apart from the precision of the ribs, which shows a low value of about 0.81 in the validation dataset. This means that a high number of false positives are generated by the automated algorithm. Perhaps some cartilage is erroneously recognizing as bone or the ribs themselves are over-segmented. A slightly higher thresholding value could in this case improve the results. A further discussion on this topic is proposed in the next paragraph, where the metrics evaluated in the testing dataset are analyzed.

Dice coefficient

Class	Training dataset	Validation dataset
Spine	0.93353	0.95395
Hips	0.96749	0.97537
Sternum	0.95455	0.95054
Ribs	0.88025	0.84949
Sacrum	0.94663	0.90885
Femoral bones	0.97623	0.98020
All bones	0.94307	0.93653

Table 7.3: The first six rows show the dice coefficient computed on the training and validation datasets for each bone class. The last row refers to the average dice coefficient of bone classes on the same datasets. This is computed image by image and therefore it is not the mean of the values above.

Precision

Class	Training dataset	Validation dataset
Spine	0.92577	0.97065
Hips	0.97349	0.97782
Sternum	0.94901	0.95225
Ribs	0.91386	0.81249
Sacrum	0.94729	0.94625
Femoral bones	0.97827	0.98707
All bones	0.94787	0.94167

Table 7.4: The first six rows show the precision computed on the training and validation datasets for each bone class. The last row refers to the average precision of bone classes on the same datasets. This is computed image by image and therefore it is not the mean of the values above.

Recall		
Class	Training dataset	Validation dataset
Spine	0.95238	0.94219
Hips	0.97438	0.98070
Sternum	0.98220	0.97206
Ribs	0.89981	0.93557
Sacrum	0.97256	0.92212
Femoral bones	0.99141	0.98636
All bones	0.96677	0.9494

Table 7.5: The first six rows show the recall computed on the training and validation datasets for each bone class. The last row refers to the average dice coefficient of bone classes on the same datasets. This is computed image by image and therefore it is not the mean of the values above.

7.4 Evaluation and results

Considering the validation results, the best model obtained is saved and then evaluated with a testing dataset previously unseen by the network. This dataset is made up of 4 full-abdominal scans, each one including between 811 and 881 axial sections with thickness between 0.7 and 0.8 mm, for a total of 3370 transversal slices. Contrary to what explained for the previous datasets, this one was labeled by experts. It is worth pointing out that those scans were carefully checked one by one in a very time-consuming process and their annotations are surely more precise than the ones belonging to the training and validation datasets.
In order to obtain the segmentation mask from the output of the CNN, the optimal thresholding values previously found are selected. In Table 7.6 the values obtained for the Dice coefficient, precision and recall on the test dataset are reported. Furthermore, in Figure 7.2 the dice coefficient values obtained for each bone class in the training, validation and testing dataset are compared. The graph shows that there are not significant differences between the datasets and therefore the network is not over-fitting the training data. Once again it is worth pointing out that ribs represent the most difficult bone class to be segmented. Overall the results are satisfying as every bone class presents a dice coefficient of about 0.88 or higher, meaning that the network performs a pretty accurate segmentation and classification. Regarding precision and recall, the only uncertainty observed is the low value of recall for the ribs. Contrary to the validation dataset, where the precision was low, here a recall of about 0.82 suggests that the ribs are slightly under-segmented. In this case a higher threshold value for the class would probably help improving the results.
For just a visualization purpose in Figure 7.3 a prediction of a test scan is shown. The prediction is displayed in 3D and to each bone class is assigned a different color. It's worth noticing a few easily recognizable mistakes made by the model:

- The extremities of the ribs are slightly fragmented, with cartilage possibly identified as bone and some parts of the ribs not recognized. This is in agreement with the low recall value founded in the evaluation of the testing dataset and previously discussed.
- The slices including the intersection between the spine and the sacrum show some mistakes, with a few sacrum voxels erroneously identified as spine.
- Some parts of the clavicles are identified as hips, probably because of the similar shape in an axial view.

Class	Dice coefficient	Precision	Recall
Spine	0.91262	0.91467	0.93623
Hips	0.95356	0.96160	0.97982
Sternum	0.92502	0.95506	0.95079
Ribs	0.87780	0.98573	0.82411
Sacrum	0.94597	0.96617	0.96443
Femoral bones	0.98247	0.98404	0.99340
All bones	0.93290	0.96153	0.94851

Table 7.6: The first six rows show the dice coefficient, precision and recall computed on the testing dataset for each bone class. The last row refers to the average dice coefficient, precision and recall of bone classes on the same dataset. This is computed image by image and therefore it is not the mean of the values above.

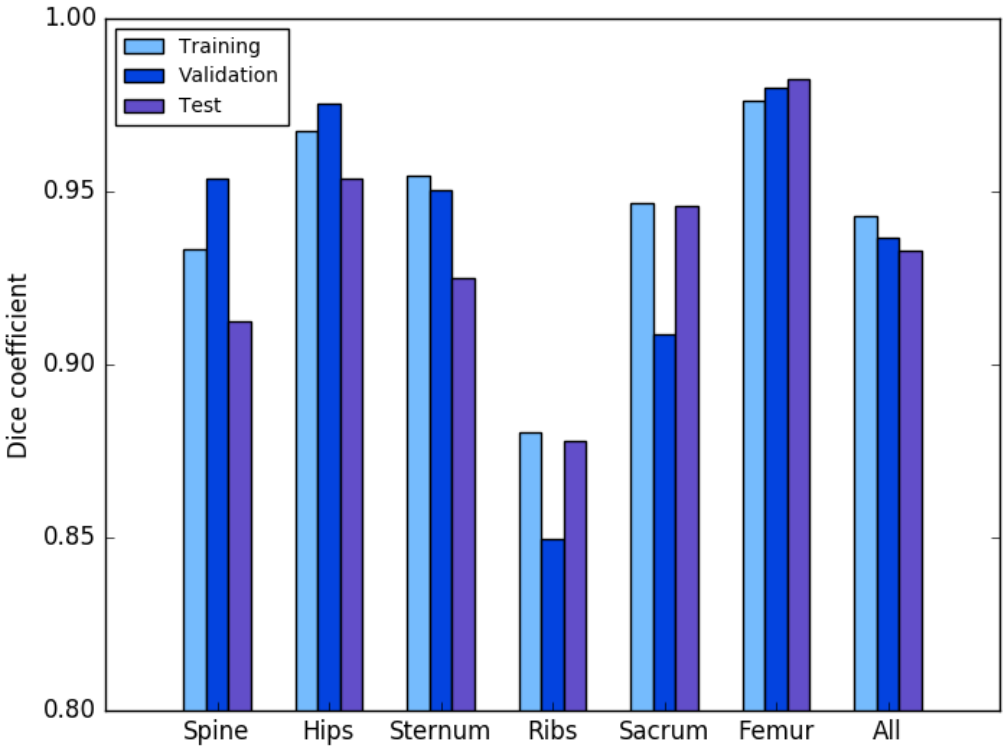


Figure 7.2: Comparison of the dice coefficient of all bone classes obtained testing the network in the training, validation and testing datasets.

- A few noisy voxels sparse in the whole scan can be observed.

Besides the noisy voxels, that could be easily removed with a quick post-processing, the others are intrinsic mistakes of the segmentation method. Possible methods to improve those results are discussed in the next chapter.

7.5 Noise testing

Noise affects CT scans degrading their quality and resulting in a loss of image details. Regarding a new generation CT scanner, noise is associated with the number of x-rays contributing to each detector measurement, as the electronic and other noise sources should be minimal in comparison [25]. A limited number of x-rays causes indeed fluctuations in the CT number. There are several techniques applied to medical images to reduce noise. However, there is not a clear agreement in literature regarding how noise affects the performance of CNNs. Therefore it is interesting to see how the model developed in this work behaves with noisy images. For this purpose, gaussian noise was added to the testing dataset with mean equal to 500 and its standard deviation which assumes 6 different values: 0, 20, 40, 60, 80 and 100. Figure 7.5 shows the visual degradation of a zoomed-in slice with those standard deviation values of gaussian noise. In Table 7.7 the values of the Dice coefficient of all bone classes obtained are reported. As expected, increasing the standard deviation of the gaussian noise the Dice coefficient decreases. The model, however, turns out to be quite robust to gaussian noise, as its metric does not drop to a low value. With a standard deviation of 100 the network still achieved a Dice coefficient of 0.86.

Standard deviation	Dice coefficient
0	0.93085
20	0.92938
40	0.92786
60	0.92285
80	0.90154
100	0.86478

Table 7.7: Dice coefficient of all bone classes obtained with a noisy testing dataset. The left column shows the standard deviation of the gaussian noise, with mean 500, added to the testing dataset. The right column the relative Dice coefficient.

In order to do a further test another model is trained, with the same network architecture and hyperparameters. The only difference consists in adding a gaussian noise function in the data augmentation code. This function adds to each training sample gaussian noise with mean 500 and standard deviation randomly extracted between 0 and 100. In this way the model is trained with noisy images and has the chance to increase its robustness during training. After the usual 150 epochs the model is saved and tested with both the original and noisy testing dataset. The results are reported in Table 7.8. In Figure 7.4 the Dice coefficient vales of all bone classes are shown and compared to the ones previously obtained. It can be seen how on one hand the last trained model keeps a Dice coefficient more or less constant around 0.90. On the other hand the original model has a higher initial Dice

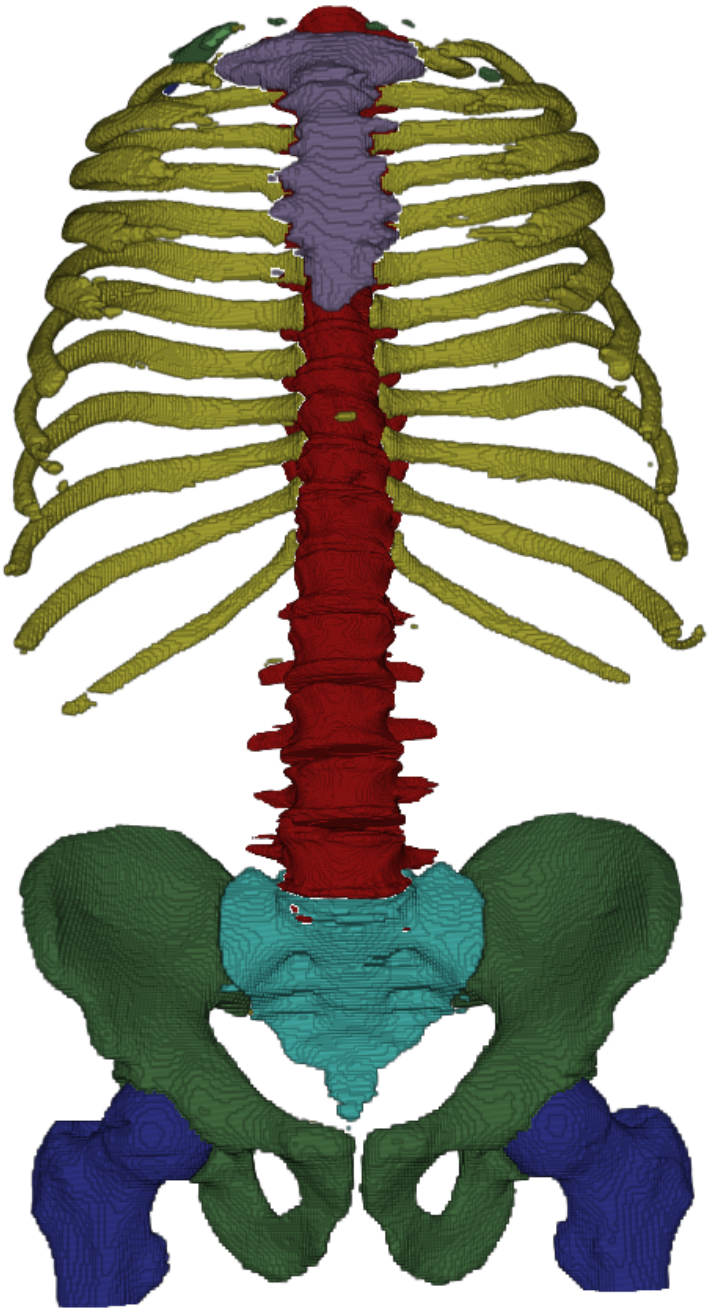


Figure 7.3: 3D view of a prediction of a test image. A few mistakes and noisy voxels can be noted.

coefficient, which then decreases as the standard deviation of the noise increases. Until a standard deviation of 60, however, the model is almost insensitive to noise.
From those results it can be deduced that the network is not capable of learning more than a certain amount of information. Providing noisy training samples increases the performance on a noisy testing dataset, but degrades it on clean images. Thus, it is advisable to train the model on samples which contains exactly the level of noise expected in the testing dataset.

Standard deviation	Dice coefficient
0	0.90510
20	0.90507
40	0.90597
60	0.90476
80	0.90388
100	0.90215

Table 7.8: Dice coefficient of all bone classes obtained with a noisy testing dataset, with a model trained with noisy images. The left column shows the standard deviation of the gaussian noise, with mean 500, added to the testing dataset. The right column the relative Dice coefficient.

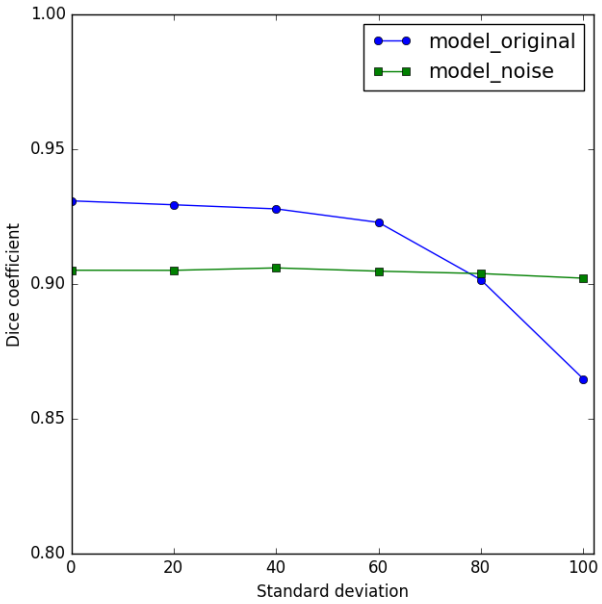
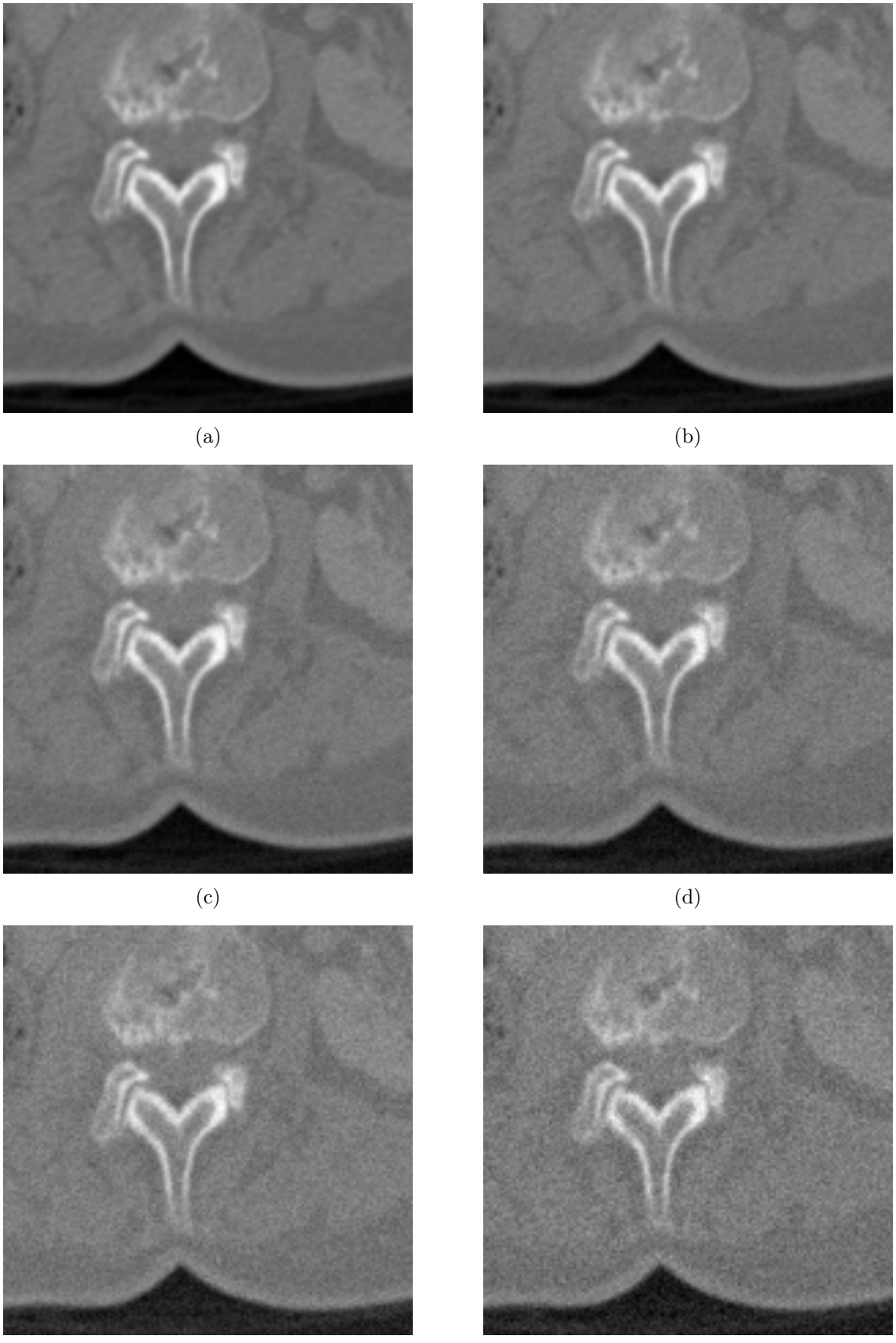


Figure 7.4: Comparison of the dice coefficient of all bone classes obtained with the two models. model_original refers to the original one, whereas model_noise is the one trained with noisy images.



(a) (b) (c) (d) (e) (f)

Figure 7.5: Zoomed-in images of a single slice that show the visual degradation due to gaussian noise. From left to right and from top to bottom, images with gaussian noise of mean 500 and increasing standard deviation, respectively of 0 (a), 20 (b), 40 (c), 60 (d), 80 (e) and 100 (f).

Chapter 8

Conclusions and future developments

The purpose of this work, mainly done at Fraunhofer MEVIS in Bremen, Germany, is to test the feasibility of automatic bone segmentation and classification of abdominal CT images using a CNN. In the first chapters the basic principles of CNNs and the most used segmentation techniques are summarized. Then the deep learning approach proposed is described, including the network implementation and the pre-processing and data augmentation applied. Finally, in the last chapter the results obtained on an expert labeled, testing dataset, are discussed.
The best model achieves a Dice coefficient of 0.91 or higher for all bone classes apart from the ribs, for which it is equal to 0.87. They turn out to be the most challenging class to be segmented. The femoral bones is the class better found, with a Dice coefficient of 0.98. Overall, considering the average of all bone classes on each image, the model reaches a Dice coefficient of 0.93. Introducing some gaussian noise to the testing dataset the performance of the model remain stable until a certain intensity and then start to slightly degrade. Another model, with the same architecture and hyperparameters, is also trained adding some random gaussian noise to the training dataset. Its results show a stable Dice coefficient: it has improved on the very noisy images, but has decreased on the testing dataset with a low noise level. Therefore the CNN is able to learn from noisy image, but unable to reach a very high segmentation accuracy on both clean and noisy images. Those results are satisfying and show the capability of CNNs to tackle this challenging segmentation task. A single CNN has proved to be able not only of segmenting bones, but also of classifying them in six classes with a good accuracy.
It is not viable to compare directly the results obtained with the ones of previous bone segmentation studies. This work presents a novel approach which performs an end-to-end segmentation without any hand-crafted features extraction or parameters tuning after its training. Compared to conventional segmentation methods, previously proposed in literature, it is totally automatic and has a least dependency on the operator. Furthermore, those relevant studies, analyzed in Chapter 4, do not tackle the exact same problem, focusing on the segmentation of a different number of bones. With an overall average Dice coefficient of 0.93 this approach achieves performance comparable to all the most recent publications regarding bone segmentation in CT scans.
However, this study has limitations and with a few adjustments its results could improve. First, the CNN developed in this study deals only with transversal 2D images. In order to have a more accurate 3D bone labeling a possible solution could be to train three models on the three orthogonal projections of the images. Then the output masks of each model would be merged based on a majority voting rule that determines to which class assigning every voxel. Second, the data used for training is limited to only 21 scans. Moreover, the training dataset was labeled in a different way from the testing one, manually checked by

experts. Therefore with a larger training dataset, annotated by experts, the performance of the model are expected to improve. Third, the number of layers and of feature maps of the network were limited by the GPU memory. This does not allow to test deeper models which would likely be able to learn more features from the data. The batch size is also limited for the same reason and it slows down the training, requiring to keep a low learning rate. Finally, it is worth pointing out that this model is trained end-to-end and does not apply any post-processing to the output masks. An algorithm of this kind could easily remove isolated voxels assigned to a bone class and reduce the noise.
In the future, a very interesting development would consist in implementing the network with 3D convolutional layers. As the bones are 3D structures the network would be able to extract more features and likely achieve a higher segmentation accuracy. Especially the ribs, which present a very fragmented shape in transversal slices, are expected to be better segmented. However, once more, this would require a larger GPU memory and probably a longer training time. In addition, further progress could be made using an adversarial segmentation framework. As described in Chapter 3, recent publications have shown that adversarial training is able to improve the output segmentation masks given by a single CNN. Thus, training an adversarial network along with the CNN proposed in this work could improve the segmentation accuracy.
In conclusion, this work demonstrates the feasibility of automatic bone segmentation and classification of CT scans using a CNN. The model achieves a high Dice coefficient and turns out to be quite robust to gaussian noise. However, several limitations and improvements that would surely increase the performance are identified. An improved, but similar model could be useful in several clinical and research applications in multiple medical tasks.

Bibliography

[1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[2] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen AWM van der Laak, Bram van Ginneken, and Clara I Sánchez. A survey on deep learning in medical image analysis. *arXiv preprint arXiv:1702.05747*, 2017.

[3] Rafael C Gonzalez. Digital image processing. Technical report, 1977.

[4] Neeraj Sharma, Lalit M Aggarwal, et al. Automated medical image segmentation techniques. *Journal of medical physics*, 35(1):3, 2010.

[5] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015.

[6] Stanford University. Cs231n: Convolutional neural networks for visual recognition. `http://cs231n.stanford.edu/`. (Accessed on 25/5/2017).

[7] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[8] Arnau Oliver, Jordi Freixenet, Joan Marti, Elsa Pérez, Josep Pont, Erika RE Denton, and Reyer Zwiggelaar. A review of automatic mass detection and segmentation in mammographic images. *Medical image analysis*, 14(2):87–110, 2010.

[9] Regina Pohle and Klaus D Toennies. Segmentation of medical images using adaptive region growing. In *Medical Imaging 2001*, pages 1337–1346. International Society for Optics and Photonics, 2001.

[10] MK Date and Mr SP Akarte. Brain image segmentation algorithm using k-means clustering. *Int. J. Comput. Sci. Appl*, 6(2), 2013.

[11] Alan Jose, S Ravi, and M Sambath. Brain tumor segmentation using k-means clustering and fuzzy c-means algorithms and its area calculation. *International Journal of Innovative Research in Computer and Communication Engineering*, 2(3):32–57, 2014.

[12] Miin-Shen Yang, Yu-Jen Hu, Karen Chia-Ren Lin, and Charles Chia-Lee Lin. Segmentation techniques for tissue differentiation in mri of ophthalmology using fuzzy clustering algorithms. *Magnetic Resonance Imaging*, 20(2):173–179, 2002.

[13] S Madhukumar and N Santhiyakumari. Evaluation of k-means and fuzzy c-means segmentation on mr images of brain. *The Egyptian Journal of Radiology and Nuclear Medicine*, 46(2):475–479, 2015.

[14] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. *arXiv preprint arXiv:1606.04797*, 2016.

[15] Alex Fedorov, Jeremy Johnson, Eswar Damaraju, Alexei Ozerin, Vince Calhoun, and Sergey Plis. End-to-end learning of brain tissue segmentation from imperfect labeling. In *Neural Networks (IJCNN), 2017 International Joint Conference on*, pages 3785–3792. IEEE, 2017.

[16] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[17] Özgün Çiçek, Ahmed Abdulkadir, Soeren S Lienkamp, Thomas Brox, and Olaf Ronneberger. 3d u-net: learning dense volumetric segmentation from sparse annotation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 424–432. Springer, 2016.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[19] Pim Moeskops, Jelmer M Wolterink, Bas HM van der Velden, Kenneth GA Gilhuijs, Tim Leiner, Max A Viergever, and Ivana Išgum. Deep learning for multi-task medical image segmentation in multiple modalities. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 478–486. Springer, 2016.

[20] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[21] Pauline Luc, Camille Couprie, Soumith Chintala, and Jakob Verbeek. Semantic segmentation using adversarial networks. *arXiv preprint arXiv:1611.08408*, 2016.

[22] Simon Kohl, David Bonekamp, Heinz-Peter Schlemmer, Kaneschka Yaqubi, Markus Hohenfellner, Boris Hadaschik, Jan-Philipp Radtke, and Klaus Maier-Hein. Adversarial networks for the detection of aggressive prostate cancer. *arXiv preprint arXiv:1702.08014*, 2017.

[23] Wei Dai, Joseph Doyle, Xiaodan Liang, Hao Zhang, Nanqing Dong, Yuan Li, and Eric P Xing. Scan: Structure correcting adversarial network for chest x-rays organ segmentation. *arXiv preprint arXiv:1703.08770*, 2017.

[24] Pim Moeskops, Mitko Veta, Maxime W Lafarge, Koen AJ Eppenhof, and Josien PW Pluim. Adversarial training and dilated convolutions for brain mri segmentation. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, pages 56–64. Springer, 2017.

[25] Lee W Goldman. Principles of ct and ct technology. *Journal of nuclear medicine technology*, 35(3):115–128, 2007.

[26] Liping Ingrid Wang, Michael Greenspan, and Randy Ellis. Validation of bone segmentation and improved 3-d registration using contour coherency in ct data. *IEEE transactions on medical imaging*, 25(3):324–334, 2006.

[27] Jing Zhang, C-H Yan, C-K Chui, and S-H Ong. Fast segmentation of bone in ct images using 3d adaptive thresholding. *Computers in biology and medicine*, 40(2):231–236, 2010.

[28] Yan Kang, Klaus Engelke, and Willi A Kalender. A new accurate and precise 3-d segmentation method for skeletal structures in volumetric ct data. *IEEE transactions on medical imaging*, 22(5):586–598, 2003.

[29] Jeff Calder, Amir M Tahmasebi, and Abdol-Reza Mansouri. A variational approach to bone segmentation in ct images. In *SPIE Medical Imaging*, pages 79620B–79620B. International Society for Optics and Photonics, 2011.

[30] K Punnam Chandar and T Satyasavithri. Segmentation and 3d visualization of pelvic bone from ct scan images. In *Advanced Computing (IACC), 2016 IEEE 6th International Conference on*, pages 430–433. IEEE, 2016.

[31] Valérie Duay, Nawal Houhou, and J-P Thiran. Atlas-based segmentation of medical images locally constrained by level sets. In *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, volume 2, pages II–1286. IEEE, 2005.

[32] Yabo Fu, Shi Liu, H Harold Li, and Deshan Yang. Automatic and hierarchical segmentation of the human skeleton in ct images. *Physics in Medicine and Biology*, 62(7):2812, 2017.

[33] Adhish Prasoon, Kersten Petersen, Christian Igel, François Lauze, Erik Dam, and Mads Nielsen. Deep feature learning for knee cartilage segmentation using a triplanar convolutional neural network. In *International conference on medical image computing and computer-assisted intervention*, pages 246–253. Springer, 2013.

[34] Cem M Deniz, Spencer Hallyburton, Arakua Welbeck, Stephen Honig, Kyunghyun Cho, and Gregory Chang. Segmentation of the proximal femur from mr images using deep convolutional neural networks. *arXiv preprint arXiv:1704.06176*, 2017.

[35] François Chollet et al. Keras. `https://github.com/fchollet/keras`, 2015.

[36] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[37] NVIDIA Corporation. *CUDA C Programming Guide*. NVIDIA Corporation, Version 8.0. 2017.

[38] Felix Ritter, Tobias Boskamp, André Homeyer, Hendrik Laue, Michael Schwier, Florian Link, and H-O Peitgen. Medical image analysis. *IEEE pulse*, 2(6):60–70, 2011.

[39] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[40] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[41] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint arXiv:1511.00561*, 2015.

[42] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2650–2658, 2015.

[43] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[44] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[45] Abdel Aziz Taha and Allan Hanbury. Metrics for evaluating 3d medical image segmentation: analysis, selection, and tool. *BMC medical imaging*, 15(1):29, 2015.