Universidade do Minho
Escola de Engenharia
Departamento de Eletrónica Industrial

Bruno André Pires da Silva

Automatic quantitative assessment of Pectus Excavatum severity using CT images and deep learning

Guimarães, Janeiro de 2021

Universidade do Minho
Escola de Engenharia
Departamento de Eletrónica Industrial

Bruno André Pires da Silva

Automatic quantitative assessment of Pectus Excavatum severity using CT images and deep learning

Dissertação de Mestrado
Mestrado Integrado em Engenharia Eletrónica Industrial
e Computadores

Trabalho efetuado sob a orientação de:
Professor Doutor Jaime Francisco Cruz Fonseca
Doutor Sandro Filipe Monteiro Queirós

Guimarães, Janeiro de 2021

AGRADECIMENTOS / ACKNOWLEDGEMENTS

Primeiramente quero agradecer ao meu orientador, professor Doutor Jaime Fonseca por me ter dado a oportunidade de fazer este trabalho e por me ter orientado, e ao meu coorientador Doutor Sandro Queirós pela orientação, por todo o apoio e pelos conhecimentos fornecidos durante todo o percurso da tese.

Um grande agradecimento ao professor Doutor Jorge Correia Pinto pelo fornecimento do *dataset* utilizado nesta tese, e pelos conhecimentos do foro clínico.

Um enorme agradecimento à Dra. Inês Pessanha pela enorme ajuda na criação do *dataset* e pelo conhecimento prestado na realização desta tese.

Um agradecimento a todos os professores do curso pelos conhecimentos e competências transmitidos durante este percurso académico, e a todos os meus amigos e colegas de curso pelo convívio e apoio dado nos últimos cinco anos.

Por fim, e não menos importante, quero agradecer à minha família e amigos por todo o apoio e carinho demonstrado ao longo destes cinco anos.

The work presented in this thesis was performed in the Life and Health Sciences Research Institute (ICVS, School of Medicine) and the Algoritmi Center (School of Engineering), University of Minho. Financial support was provided by National funds, through the Foundation for Science and Technology (FCT) - project UIDB/50026/2020 and UIDP/50026/2020.

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration. I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

ABSTRACT

Pectus excavatum (PE) is the most common congenital abnormality of the thoracic cage, characterized by the inward displacement of the sternum and the adjacent costal cartilages. To assess and quantify the patient's severity, three standard indices are extracted from computed tomography (CT) images: the Haller index; the correction index; and the asymmetry index. Currently, there is no solution to automate the extraction of these indices.

The focus of this thesis is to develop an automatic system that analyzes the CT of the patient and, as a result, provides the measurements and indices of PE with accuracy and reproducibility, saving healthcare professionals' time and increasing measurements' reproducibility, which may ultimately improve patients' diagnosis and care.

Through the use of deep learning, a framework was developed consisting of three stages. In the first two stages, two convolution neural networks (CNNs), based on the UNet++ architecture, were used to learn the CT images' features in order to select the optimal CT slice to take the measurements and identify the relevant eight keypoints. The final stage relies on geometric rules to calculate the measurements and indices of PE from the identified keypoints.

Both automatic and manual analyses were performed on 269 CTs of 92 patients. For comparative purposes, intra-observer, inter-observer, and intra-patient variability of the estimated indices were analyzed in a subset of patients. The developed system proved to be viable, showing a good agreement with the manual approach, with small bias and narrow limits of agreement (LOA), being the results comparable with the inter-observer agreement. In the intra-patient analysis, the proposed framework outperformed the expert, showing a higher reproducibility between indices extracted from distinct CTs of the same patient. The developed system achieved an average relative error against manually extracted values of 4.09%, 5.59% and 1.91% for the Haller, correction and asymmetry indices, respectively.

The results are promising, supporting the feasibility of using the developed framework for the automatic and accurate quantification of PE severity in a clinical context.

Keywords: Pectus Excavatum, Computer-aided diagnosis, Medical image analysis, Deep learning, Convolutional Neural Network.

RESUMO

O Pectus Excavatum (PE) é a anormalidade congênita mais comum da caixa torácica, sendo caracterizada pela depressão do esterno e das cartilagens costais adjacentes. Para que seja avaliada e quantificada a gravidade da patologia num dado paciente, são calculados três índices a partir de imagens de tomografia computorizada (TAC): o índice de Haller; o índice de correção; e o índice de assimetria. Atualmente não existe solução para automatizar a extração destes índices.

O foco desta tese consiste no desenvolvimento de um sistema automático que analisa a TAC do paciente e, como resultado, calcula as medidas e índices relativos ao *PE* com precisão e reprodutibilidade, minimizando desta forma o tempo despendido pelos profissionais de saúde e aumentando a reprodutibilidade das medidas, o que pode, em última análise, melhorar o diagnóstico e o cuidado dos pacientes.

Por meio de técnicas de aprendizagem profunda, foi desenvolvida uma *framework* composta por três blocos. Nos primeiros dois blocos, duas redes neuronais convolucionais, baseadas na arquitetura UNet++, foram utilizadas para aprender as características das imagens da TAC e, assim, selecionar o corte axial da TAC ideal para extrair as medidas e identificar os oito pontos-chave para o cálculo destas. Na etapa final, através de regras geométricas, são calculadas as medidas e os índices do PE a partir dos oito pontos-chave identificados.

A estratégia proposta foi validada com o recurso a 269 TACs de 92 pacientes, e comparada com a abordagem manual. Para fins comparativos, a variabilidade intra-observador, inter-observador e intra-paciente dos índices estimados foi analisada um subconjunto de pacientes. O sistema desenvolvido mostrou-se viável, apresentando uns limites de concordância estreitos com a abordagem manual, e com resultados comparáveis com a variabilidade inter-observador. Na análise intra-paciente, a *framework* proposta superou o especialista, obtendo uma maior reprodutibilidade entre índices extraídos de TACs distintas de um mesmo paciente. O sistema desenvolvido obteve, quando comparado com valores extraídos manualmente, um erro relativo médio de 4,09% para o índice de Haller, de 5,59% para o índice de correção e de 1,91% para o índice de assimetria.

Os resultados são promissores, comprovando o potencial da *framework* desenvolvida para a quantificação precisa e automática da severidade do PE no contexto clínico.

Palavras chave: Pectus Excavatum, Diagnóstico auxiliado por computador, Análise de imagens médicas, Aprendizagem profunda, Rede neural convolucional.

TABLE OF CONTENTS

LIST OF ACRONYMS

A

ANN	Artificial neural network.
APD	anteroposterior distance.

B

BN	batch normalization.

C

CNN	convolution neural network.
CT	computed tomography.

D

DL	Deep learning.
DSNT	Differential Spatial to Numerical Transform.

F

FC	fully connected.
FCN	fully convolutional network.

H

HM	heatmap matching.
HU	Hounsfield units.

K

KDE	keypoint distance error.

L

LAPD	left anteroposterior distance.
LOA	limits of agreement.

M

MIP	maximal intensity projection.
MKDE	mean keypoint distance error.
ML	Machine learning.
MLP	multi-layer perceptron.
MSE	mean squared error.

P

PAF	Part Affinity Field.
PE	Pectus excavatum.

R

RAPD	right anteroposterior distance.
ROI	region of interest.

S

SGD	Stochastic gradient descent.

T

TCD	transverse chest distance.

V

VCD	virtual correction distance.

LIST OF FIGURES

LIST OF TABLES

1

INTRODUCTION

This chapter aims to present the thematic of this thesis (what is the pectus excavatum, what are its causes and the tools used to diagnose and correct it) and afterwards presents its motivation, objectives and contributions.

1.1 Pectus excavatum

Pectus excavatum (PE) is the most common congenital abnormality of the thoracic cage, characterized by the inward displacement of the sternum and the adjacent costal cartilages. This abnormality may be present at birth or only start to develop at the beginning of puberty. The estimated occurrence of PE is 1 in 400 to 1 in 1000 live births, with males affected 3-5 times more often than females [1]. The most widely accepted theory to explain the PE formation is the overgrowth of the costal cartilages which pushes the sternum inward (illustrated with blue color on figure 1), which is frequently intensified during growth spurts. PE can be classified by the degree of depression, by the shape - either a cup-shaped concavity (deep, localized and well-defined) or a saucer-shaped concavity (wider and shallower), and by the symmetry, where can be symmetric or asymmetric with a right-side predominance (right illustration on figure 1) [1].

The main known effect caused by PE are the psychological effects, especially in teenage patients, as there is an avoidance of physical activities and patients often try to disguise the deformity with clothes or pathological posture, leading to lower than average self-esteem and quality of life [2]. This condition can also introduce cardiopulmonary complications if it is severe, once the heart can be displaced leftward with concomitant rotation [3][4]. Due to the reduced internal thoracic volume, the lung capacity can be slightly below average having no impact during daily activities, but when performing physical exercise respiratory impairment can be detected [3][5].

PE is first detected by visual examination, where it is checked if the depression is present on the anterior chest wall of the patient. However, in order to evaluate its severity and to determine whether repair surgery is needed, a computed tomography (CT) scan is performed to have a better idea of the rib cage volume. From the CT scan, it is possible to extract measurements of the patient’s ribcage and calculate various indices that allow determining the PE severity.

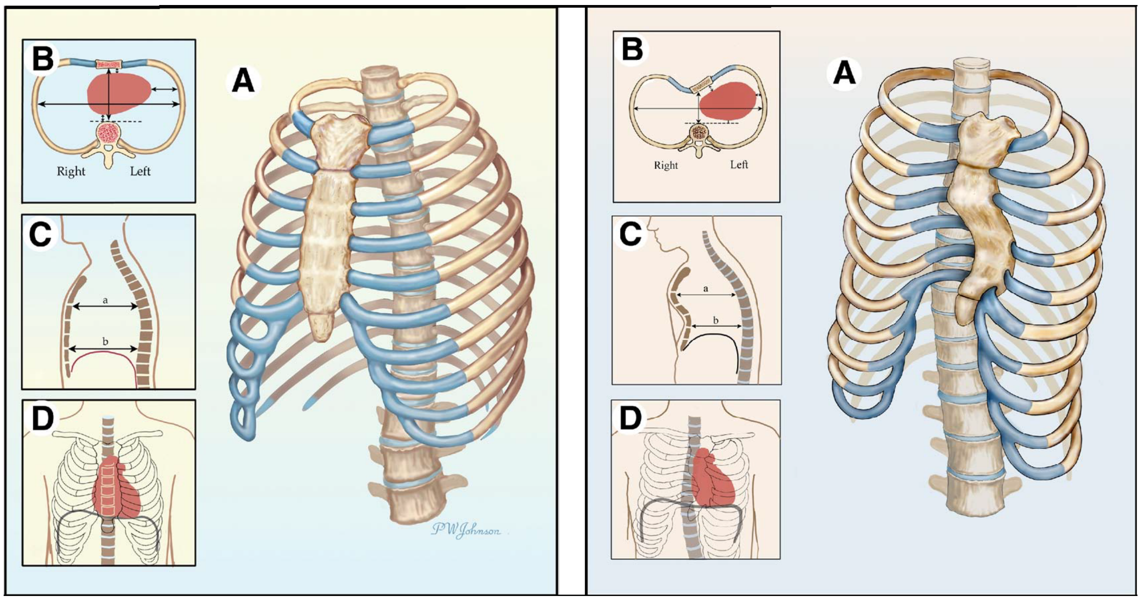

Figure 1: Anatomical representation of a normal thoracic cage (left) versus an asymmetrical PE (right) (A) Three dimensional view. (B) Axial cross-section view. (C) Sagittal cross-section view. (D) Coronal cross-section view. On the right, it is possible to observe the right-sided asymmetrical cup-shaped PE with sternal rotation (B), the deep depression of the sternum (C) and the heart shifted to the left (D). Taken from [1].

The repair of PE began in the early 20th century with techniques that resulted in unsatisfactory outcomes [6].

In 1987, Nuss developed a minimally invasive technique for the repair of PE, where no cartilage or sternal resection was required, with the procedure taking advantage of the flexibility and malleability of the chest wall. The technique consists on making two transverse incisions on each side of the lateral chest wall, a skin tunnel is raised and the intercostal space is entered with a curved Kelly clamp, which is slowly advanced until emerged on the opposite side. When the tunnel is large enough the bar is placed with the convexity facing posteriorly under the sternum with the help of umbilical tape. When the bar is on the correct position, it is turned over so that the convexity faces anteriorly and therefore the sternum and anterior wall is raised outward to the desired position. The bar is then secured with heavy sutures to the lateral chest wall muscles (figure 2) [7].

1.2 Computed tomography

CT is a non-invasive medical imaging procedure that allows detailed images of internal organs, soft tissues and bones to be created by measuring the attenuation coefficients of X-ray beams on the examined tissue. During the acquisition, a beam of X-rays is aimed at the patient while rotating around the body. On the other side of the X-ray source, there are detectors that capture and quantify the X-rays that pass through the patient's body, as illustrated in figure 3. This information is used

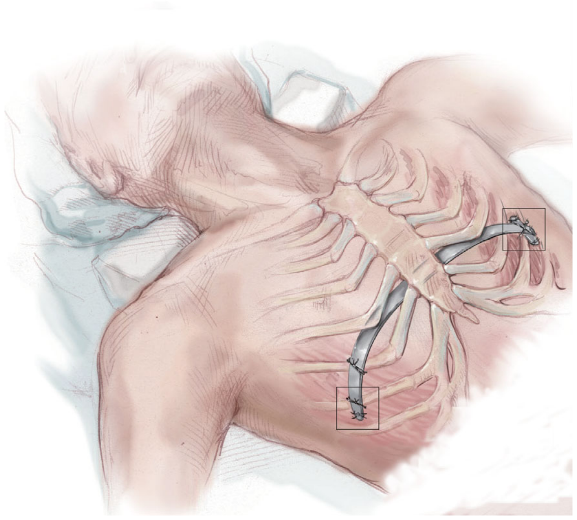

Figure 2: Patient with the bar placed under the sternum in order to raise it to the correct position. The bar is secured to the lateral chest wall with heavy sutures in order to prevent it from moving. Adapted from [8].

by reconstruction algorithms to generate cross-sectional images (slices) of the internal organs of that region. After each complete rotation around the patient's body, the bed moves forward to create another slice, generating a stack of 2D cross-sectional images with excellent spatial resolution and good soft tissue contrast.

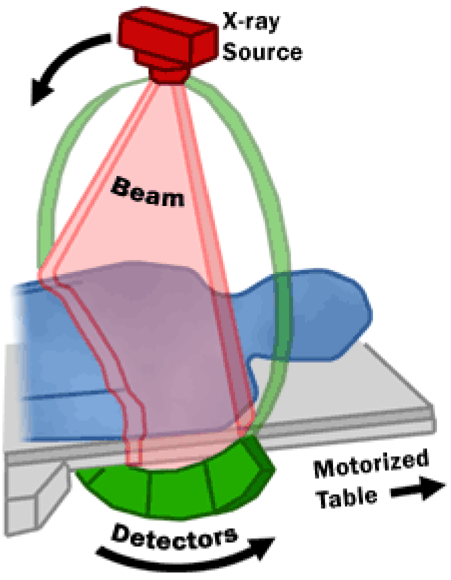

Figure 3: Computed tomography scan components.
Source: https://www.fda.gov/.

1.3 Imagiological indices of PE

In the context of PE, the imaging indices extracted from the patient's CT for the assessment of their severity are: the Haller index; the correction index; and the asymmetry index. The purpose of each one is described bellow.

The Haller index allows to describe and assess the severity of any depression that may exist in the sternum of a chest CT. It is the most common method for detecting PE, and is used during pre- and postoperative analyses of a patient. The Haller index is defined by the ratio of the transverse chest distance (TCD) (the horizontal distance of the inside of the ribcage) and the anteroposterior distance (APD) in the same axial slice, which is defined by the shortest distance from the posterior part of the sternum to the anterior face of the spine (see equation (1)) [9][10]. In healthy subjects, the Haller index has a value between 2.0 and 3.0, while patients with a significant pectus excavatum have a index greater than 3.25 [11][12].

$$Haller\,Index = TCD/APD \tag{1}$$

Although the Haller index is the most common method for determining the severity of PE, studies suggested that this index does not translate well for patients with discrepancies between the antero-posterior walls and the mediolateral walls, being the correction index more appropriate to determine the severity since it is not affected by the asymmetry in the rib cage [13][14].

Two measurements are necessary to calculate the correction index, both with respect to the tangential line that passes through the anterior spine (see red dashed line in Figure 4). The first corresponds to the distance between the spine and the position of the sternum after correction, i.e. one must perform a virtual correction of the PE (henceforth named virtual correction distance (VCD)). To this end, the maximum distance between the line placed on the anterior spine and the inner margin of the most anterior portion of the chest (i.e. a point in the inner side of the most anterior rib) is measured. The second measurement is the shortest distance between the posterior sternum and the anterior spine (as measured in the Haller index), i.e. APD. The difference between these two measurements is the amount of the deformation that exists in the chest, which is divided by the first measurement and multiplied by 100 to have the correction index, as shown in equation (2). Overall, the correction index calculates the percentage of depression in the chest [15].

$$Correction\,Index = [(VCD - APD)/VCD] \times 100 \tag{2}$$

The asymmetry index is used to characterize the degree of asymmetry present in the PE, since the Haller index and the correction index are unable to assess it correctly [16]. This index is calculated by determining the maximum anteroposterior distance from the left and right hemithorax, dividing the left anteroposterior distance (LAPD) by the right anteroposterior distance (RAPD), and multiplying by

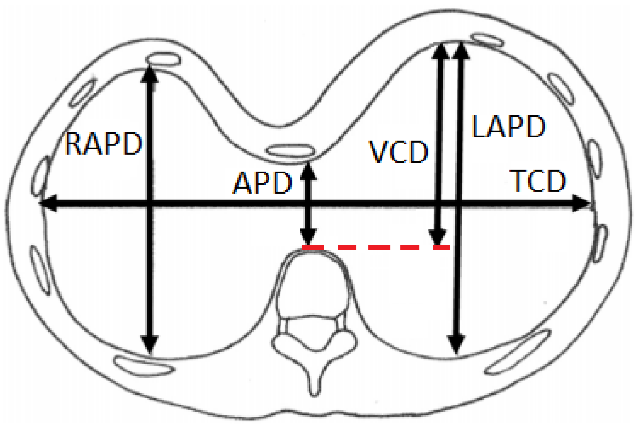

Figure 4: Illustration of the PE measurements in an axial slice of the chest.

100, resulting in the percentage of asymmetry, as shown in equation (3) [17][15]. The asymmetry index in healthy subjects fall within 95 and 105 [18].

$$Asymmetry\,Index = RAPD/LAPD \times 100 \tag{3}$$

1.4 Motivation

For a correct diagnosis of PE, besides visual assessment, the healthcare professional must analyze the patient's CT and extract the abovementioned indices by performing a series of measurements in the patients' ribcage. This analysis is tedious, time-consuming and prone to intra- and inter-observer variability, even among experts. Indeed, several sources of variability exist during this process. On the one hand, given the three-dimensional (3D) nature of the CT image, they must first identify the correct axial slice in which to perform the measurements. This slice must be located at the point of greatest depression of the chest anterior wall, so one must navigate through the CT volume and assess, using reconstructed sagittal slices, where the greatest depression of the sternum lies. On the other hand, the positioning of the points for distance measuring is intimately linked to the experience and subject to the interpretation of the healthcare professional. Despite the relevance of these indices in the context of PE, there is currently no solution to automate this process.

1.5 Aims and contributions

The focus of this thesis is to develop an automatic system that analyzes the CT of the patient and, as a result, provides the measurements and indices of PE calculated with precision and accuracy.

This system would save healthcare professionals' time and increase measurements' reproducibility, which may ultimately improve patients' diagnosis and care. The main goals/contributions of the present work are as follow:

1. Study of the state-of-the-art on keypoint/landmark detection methods in medical and non-medical images, in order to understand the challenges and difficulties present in this type of problem;

2. Develop a system that automates the measurement of parameters related to the PE on a CT, and subsequent validation of each step of the methodology, including:

   a) Development of a strategy to detect the axial slice that contains the greatest depression of the sternum, applying a keypoint/landmark detection method in a sagittal slice of the CT;

   b) Development of a strategy, using a keypoint/landmark detection method, to identify in an axial slice the 8 points (see Figure 5) necessary to calculate the measurements and indices.

   c) Development of a post-processing algorithm to correct potential errors in the detection of points in the axial slice;

   d) Development of an algorithm to calculate all measurements using the coordinates of the points in the image and thus extract the PE indices;

3. Clinical assessment of the developed framework, comparing the results against a pediatric surgeon experienced in the analysis of CTs in the context of PE.

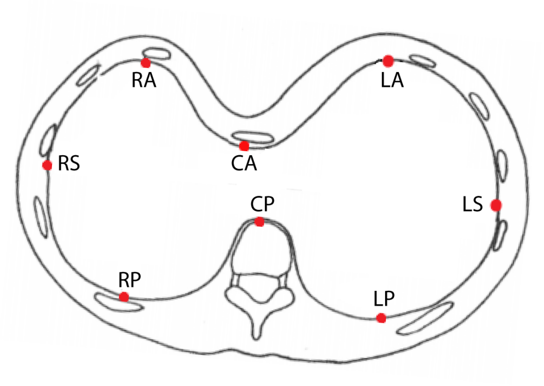

Figure 5: Keypoints to be detected on the axial slice.
CA: Central Anterior; CP: Central Posterior; RS: Right Side; RA: Right Anterior; RP: Right Posterior; LS: Left Side; LA: Left Anterior; LP: Left Posterior.

1.6 Thesis overview

The current chapter introduced the clinical context of the present thesis. First, it was given an overview of what PE is and how it is corrected, together with a description of the imaging procedure used to diagnose this problem and an explanation of the three most used indices in the clinical environment to quantify the PE severity. Finally, the motivation, the objective, and the main contributions of this work were presented. The following chapters will be dedicated to the development of the topics introduced.

The second chapter aims to present the state-of-the-art on the subject. First, it is introduced and explained the difference between machine learning and deep learning. Two deep learning techniques are then explained, describing both their basic concepts and their advantages and disadvantages, being after explained several standard architectures used for this type of problem. Finally, past works from the literature related to the thesis theme and based on deep learning techniques will be presented.

In the third chapter, the methodologies developed to achieve the objective of this work will be presented, which consists in the automatic initialization and detection of the keypoint present in a sagittal slice of a CT volume, automatic initialization and detection of the keypoints present in the previously detected axial slice and, finally, the post-processing carried out to correct those detected keypoints and extract the three relevant PE indices.

The fourth chapter will focus on the results achieved by the proposed framework and their discussion. First, it is shown an overview of the used dataset and the metrics used to assess performance. Next, the experiments carried out to assess the neural networks' accuracy and robustness will be presented. The chapter ends with the final clinical validation of the proposed framework, comparing the framework's robustness against two experts in the field.

Finally, the fifth chapter will present the main conclusions of the present thesis.

2

STATE OF THE ART

This chapter aims to introduce concepts relevant to the present thesis, such as machine learning, deep learning, supervised learning or neural networks. The basic blocks of these networks and how they are trained are described. Convolution neural networks (CNNs) are then introduced, and typical architectures relevant to this thesis are then presented. Finally, an overview on keypoint/landmark detection methods is also presented.

2.1 Machine learning and deep learning

Machine learning (ML) is a form of artificial intelligence that allows a system to learn through data instead of being previously programmed with ”intelligence”. Using a variety of algorithms, it is able to build a mathematical model from the data provided during a training phase. Using data as input, it is possible to find several relations present in the data and from these relations it is possible through algorithms to create a model that produces an accurate prediction of a determined output. This output can take a variety of forms and can be a position, a probability, a color, etc.

The ML model that is included in applications to make inferences of future predictions is generated by an algorithm upon training with a large amount of data. This model is a kind of function, in which several variables that were blank or random before training were filled in during the training phase with the correct values, so that given an input to the function its output has a prediction with the smallest possible error.

In recent years, ML has received a lot of attention, and there has been a huge growth in the integration of these algorithms in different products, enabling consumers and companies to have experiences that where not possible before: target marketing based on the browsing history, fraud transactions detection in real-time, semi-autonomous cars that can detect lines and cars on the road, detecting faces in photos on social networks, etc. This was all driven by the fact that nowadays there is a production of huge amounts of data, through the internet and various equipments that connect to it, as well as the reduced cost of computing power and storage.

Although ML has had all this attention recently, it is not something new, with this area in artificial intelligence existing since the 1950s. The first neural network, called SNARC, was created in 1951 by Minsky et al. [19][20], and the perceptron that was one of the bases for the creation of the current Artificial Neural networks was created in 1957 by Rosenblatt et al. [21].

Deep learning (DL) is a type of machine learning that allows us to solve more complex problems, thanks to artificial neural networks. Unlike machine learning in which human intervention is needed to identify which features are most useful for the model to perform well, a deep learning system tries to learn high-level features from the data without human intervention. This feature is advantageous when solving a problem since it is possible to use pre-trained deep learning models from other problems to detect specific features in a different problem, saving the necessary training. Unlike machine learning algorithms, deep learning algorithms can solve complex problems end-to-end without having to break the problem into different pieces. One of the disadvantages of deep learning algorithms is the training time since there are too many parameters within a network to adjust. Another disadvantage is the model interpretation, since there are thousands of parameters in a network, being many times difficult to understand what the network is detecting to give a specific result. Another major disadvantage is the amount of data needed for training, as the data required is usually proportional to the complexity of the problem, which are very complex in DL.

2.2 Supervised learning vs unsupervised learning

Machine learning algorithms are generally divided into two categories: supervised and unsupervised.

Supervised learning algorithms are able to generalize the information that was learned during training and apply it to new data never seen before, using examples that have already been correctly labeled to predict future events. Starting with the analysis of a labeled training dataset, which consists of samples that contain the input and the respective desired output, the algorithm infers a function to make predictions of these input values. Overall, a learning algorithm compares the output result of the inference function with the labeled output value that is present in the training dataset, and according to the resulting error from this comparison, the function is then adjusted to try to correct the error of the inference.

Unlike supervised learning, unsupervised learning algorithms are trained only with input data, with no notes on the corresponding sample output. Having no information on the output of the training samples, these algorithms do not try to find out what is the right output, but analyze the data in order to model the fundamental structure or distribution of the dataset, as to better understand the data. There are three sub-branches of unsupervised learning: clustering analysis, where the dataset is segmented by discovering groups of similar examples; density estimation, where distribution of input data is determined; and, finally, dimensionality reduction, where data from a high dimensional space is projected to a lower dimensional space, easing its visualization and interpretation, or allowing further analysis in other contexts [22][23].

2.3 Artificial Neural Networks

Artificial neural networks (ANNs) started with the single-layer perceptron, an algorithm developed in 1957 by Rosenblatt et al.[24] that tried to replicate the neurons present in the brain. Although this algorithm was promising at the beginning, it quickly proved that perceptrons could not be trained to recognize many classes of patterns, since it is a linear classifier and could only learn linearly separable patterns. Although in 1961 Rosenblatt[25] introduced the multi-layer perceptrons (MLPs) which made it possible to solve many of the classifications that were not separable by the single-layer perceptrons, the MLPs were not yet recognized as non-linear classifiers. These initial difficulties made the area of neural networks to stagnate for many years. It was only around the 1980s that this area was reborn. In the context of ANNs, a single-layer perceptron is the most basic type of neural feedforward, which uses the Heaviside step function as the activation function.

ANNs or MLPs or Feed-Forward Neural networks are one of the most fundamental types of neural networks that exist. ANNs try to model the architecture of the human brain in a simple way so that it is possible to perform difficult computational tasks. The key components of ANNs are the artificial neurons that are loosely based on neurons in the brain, and the interconnections between neurons, which resembles the synapses since it is from these connections that information, in this case, real numbers are passed from one neuron to the next.

Each neuron receives as its input the outputs of many other neurons, multiplies the input by weights that are adjusted during training, adds them up, and passes the sum to one or more neurons. Some artificial neurons can pass this sum by an activation function before passing it on to the following neurons, as represented in figure 6.

The idea behind the activation function is to introduce nonlinearities in the network so it can learn complex data, as the output of the neuron would otherwise be always linear with respect to its inputs. There are many types of activation functions. Figure 7 illustrates some of the most used ones.

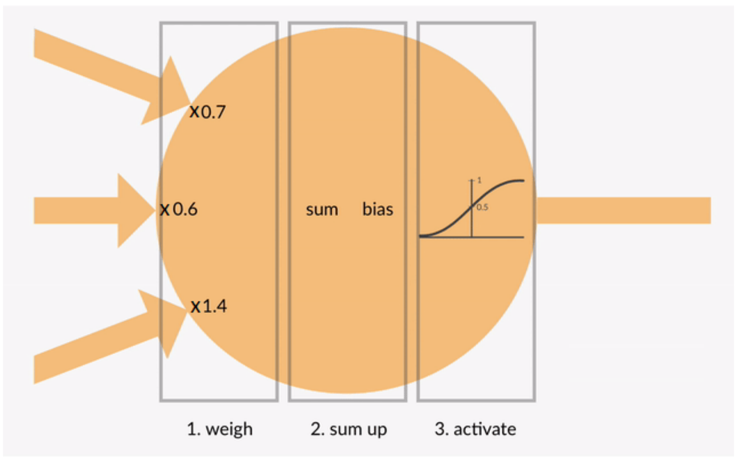

Figure 6: Illustration of a single neuron used in ANNs.
Source: https://becominghuman.ai/.

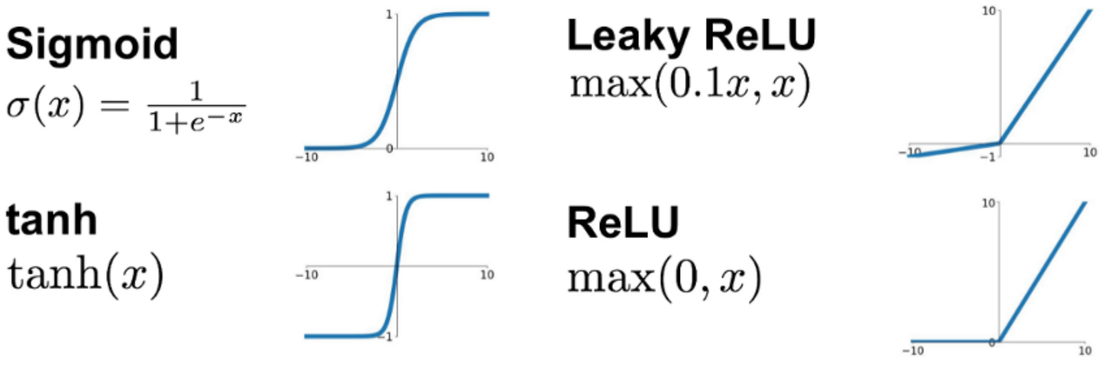

Figure 7: ReLU, Leaky ReLU, tanh and Sigmoid activation functions.
Source: https://medium.com/.

ANNs differ from single-layer perceptron in that they have nonlinear activation functions and contain multiple layers of multiple neurons. They contain at least 3 layers: the first layer is the input layer, where the access to the input data from external sources is; the second and subsequent hidden layers is where the data is processed and where the different features present in the data are detected; and finally, the last layer corresponds to the output layer where the results from the network function are provided, as seen in figure 8.

To adjust the network parameters, it is necessary to first quantify the error present in the model's predictions. In supervised models, this is done during training with a function called loss, in which the network's output is compared to the correct output given by the labeler. This allows us to see through a scalar value how good the model is at making predictions with the current parameters.

The loss function allows to have a metric to be minimized during training to improve the network performance, which is called loss optimization. To minimize the loss, it is necessary to find the correct value of weights that decrease the loss value. A way to see this more intuitively is to imagine a network that contains only two weights, this enables the representation of the loss function and the respective weights through a three-dimensional graph (figure 9), with the x and y-axis representing the possible values of the weights 0 and 1, and on the z-axis the loss resulting from these weights.

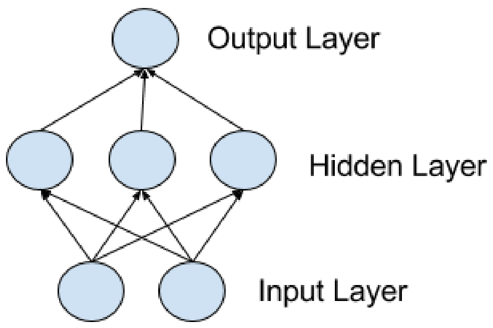

Figure 8: Basic layers of an artificial neural network.
Source: https://machinelearningmastery.com/.

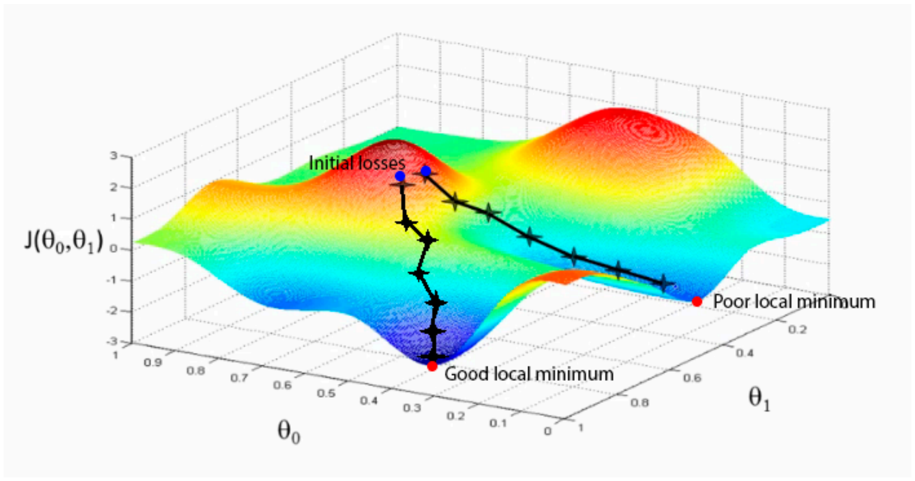

Figure 9: Example of gradient descent for the loss function of an ANN, with two weights θ_0 and θ_1. We can see that two different weights initializations can lead to different local minima. Adapted from: http://www.holehouse.org/mlclass/, by Andrew Ng.

The goal of loss optimization is to find the local minimum in this function and update the weights of the network with the respective x and y values.

Currently, the most common way to optimize the neural network loss function is through the gradient descent optimization algorithm. Gradient descent consists of calculating the gradient of the network's loss iteratively and updating the weights in the opposite direction of the gradient, thus descending in several steps in the steepest direction of the loss function until it converges to a local minimum (figure 9). The weights are refined along the way allowing the network to have a smaller error. The gradient calculation is performed by measuring the slope (derivative) in all dimensions of the input.

One method used to calculate the gradient efficiently is backpropagation. Backpropagation is a method that calculates the gradients of each weight by applying a chain rule, thus calculating the gradient of each layer at a time and passing the values to the previous layers, repeatedly, so as to know how the loss is affected by each weight (see figure 10) [26].

The gradient descent algorithm has three variants, each differing in terms of the data size used to calculate the gradient. Thus, there is a trade-off between the parameter update's precision and the time it takes to do the parameter update.

Batch gradient descent - Uses all the samples in a dataset to calculate the gradient, making only one update per epoch (an epoch is when the entire dataset is passed through the machine learning algorithm). This makes each calculated gradient have a very precise direction. However, it is very slow, and according to the dataset's size, it can take up a lot of memory, since the gradient is calculated for all weights of all samples in the dataset [27].

Stochastic gradient descent (SGD) - Contrary to the previous one, SGD calculates the gradient in each example present in the dataset, thus making a parameters update after each sample. The calculation of the gradient is more efficient and, therefore, faster. On the other hand, the variance of

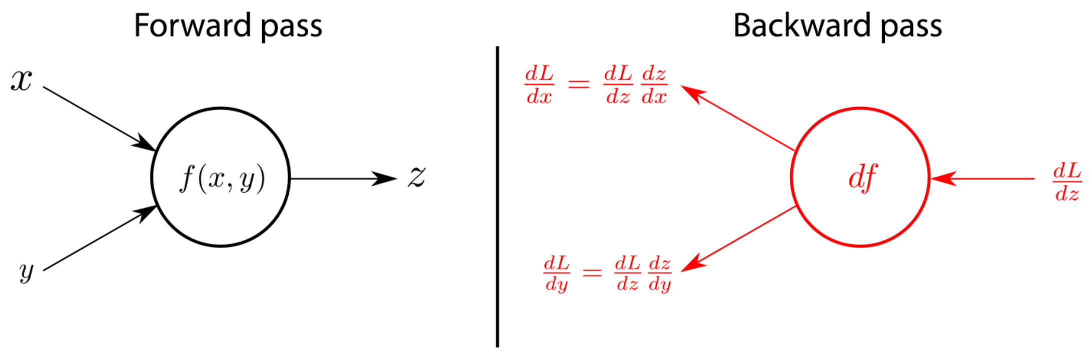
Forward pass
x
y
$f(x,y)$
z
Backward pass
$\frac{dL}{dx} = \frac{dL}{dz}\frac{dz}{dx}$
df
$\frac{dL}{dz}$
$\frac{dL}{dy} = \frac{dL}{dz}\frac{dz}{dy}$

Figure 10: Example of a forward and backward pass.
The forward pass on the left represents the operation of a neuron during training or inference. The right side of the figure shows the backward pass executed during backpropagation, receiving the gradient of the loss function with respect to z from above, and applying the chain rule to calculate the gradients of the loss function with respect to x and y. Adapted from: https://kratzert.github.io/.

the chosen direction for updating the parameters increases since the gradient calculated takes into account only one sample and not a set of samples. Thus, it is difficult for the network to converge in most cases [27].

Mini-batch gradient descent - It is a combination of the two previous methods, where the calculated gradient uses a batch of samples from the dataset, making a parameter update every *n* samples (being *n* the number of samples contained in each batch). This type of gradient descent reduces the variance in the parameter update present in the traditional SGD, since the gradient is calculated with information from several samples, thus stabilizing the model's convergence. This method also allows the use of matrix optimizations that make the calculation of the gradient more efficient [27].

The step size taken in the opposite direction of the gradient is a hyperparameter called learning rate. The learning rate value influences how the network converges to the local minimum. If it has a very low value, the network's convergence will be very slow. On the other hand, when the value is very high, the convergence will be faster but riskier, since there is the possibility of overshot and not converge to the desired local minimum.

Optimizers are algorithms that help the descent gradient to have a good convergence in adverse situations.

Momentum is a method used to help accelerate the SGD in the relevant direction, helping to dampen oscillations resulting from SGD navigation in areas with steeper surfaces in one dimension than another, something common in places close to the local minimum [27].

Adagrad [28] is an optimization algorithm that allows to adapt the learning rate for the gradient descent parameters, allowing to make larger updates in less frequent parameters and smaller updates for more frequent parameters. It is advised to use when dealing with sparse data.

Adadelta [29] is an extension of Adagrad that aims to decrease its aggressiveness, reducing the window of accumulated past gradients to a fixed value, thus counteracting one of Adagrad's weaknesses: the shrinkage of the learning rate.

RMSProp has the same purpose as Adadelta in that it tries to correct Adagrad's radically diminishing learning rates, thus allowing better performance in noisy problems [27].

Adam [30] is an optimizer that combines the best properties of AdaGrad and RMSProp, allowing to solve problems with noisy or sparse gradients.

Through these concepts, it is possible to fine-tune the parameters present in each neuron in the network, allowing through mathematical operations performed within thousands or millions of neurons in multiple layers, to obtain an artificial network capable of performing tasks in which traditional algorithms have little success. ANNs, mainly thanks to the nonlinear activation functions that allow the introduction of nonlinear properties in the network, are able to learn any function, including nonlinear ones, having the ability to learn and adjust the weights so that they can map any complex relationship that may exist between input and output. This type of networks are popularly known as Universal Function Approximators.

2.4 Convolution Neural Networks

CNNs are a subtype of neural networks that are designed in a way to make it possible to map data that contains some form of spatial structure to an output variable (Figure 11). These data can be images that can be viewed as a 2D grid of pixels, time-series data that can be viewed as 1D, or video being 3D. CNNs started with the Neocognitron introduced in 1980 by Fukushima [31], which first described the two basic layers of CNNs: the convolutional layer and the downsampling layer. In 1989, LeCun [32] started to use back-propagation to learn the convolution kernel coefficients from the images to recognize handwritten zip codes, thus automating the learning of the parameters present on the CNN and, as a result, the algorithm had a better performance.

CNNs are made up of convolution, activation function, pooling, and fully connected (FC) layers. The convolution layers are where the trainable parameters are located and where the parameters will be convoluted by the input volume. The activation function layer is where the linear feature maps resulting from the convolution layers will pass through a non-linear activation function resulting in non-linear feature maps. The pooling layers transform the size of feature maps using a fixed function, and the fully connected layers are the final layers that exist in several CNNs in order to aggregate the features collected in the network and get a final result from it. A more thorough description follows.

The convolution layers are the main building blocks of convolutional networks, and it is where most of the computational power is spent, being composed of filters and feature maps.

Convolution is a linear operation that involves multiplying the matrix of a group of parameters with the input, as in traditional neural networks. Since the structure of input data differs from ANNs,

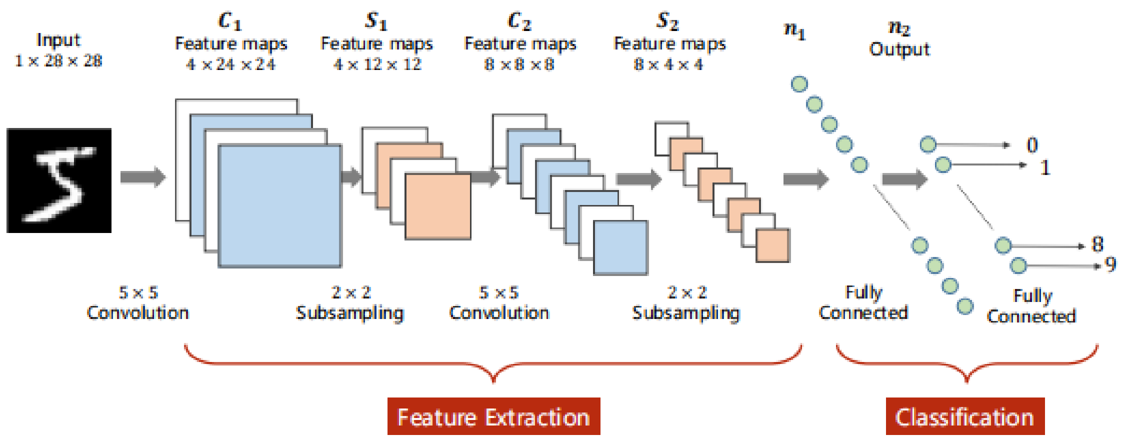

Figure 11: Example of a convolutional neural network used to recognize handwritten numbers. Source: https://www.ccyh.xyz/.

multiplication performed on CNNs is done through the use of a filter or kernel. The filters or kernels are the neurons of the convolution layer, containing the parameters that will be adjusted during training. As a rule, the size of the filters is smaller than the size of the input (usually the size of these filters is 3×3 or 5×5). The projection of each filter (which is called a receptive field) is convoluted across the width and height of the input volume originating a feature map, as illustrated in figure 12. The dot product is calculated between the input and the value of the filter at each spatial position. As a result, the network can learn filters that can detect specific features in any spatial position of the image.

In case the input has multiple channels, the filter will have the same depth as the input, being calculated a feature map for each channel with the corresponding filter. Then the feature maps for

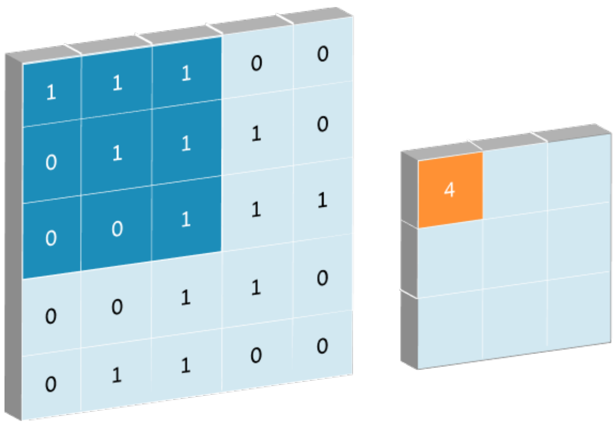

Figure 12: Example of a convolution with a 3×3 kernel
The dark blue area represents the receptive field of the kernel and the orange area is the convoluted feature, resulting from convolving the input with the kernel within the receptive field. Source: https://missinglink.ai/.

each channel are added together plus a bias, resulting in a feature map flattened across the depth dimension.

When the kernel is slid across the input, if the stride is bigger than one, the kernel may not be able to fit on the input. In these cases padding can be used to pad the input with zeros (zero-padding) so that the kernel can fit, or drop the input that the kernel is not able to fit (valid padding). Figure 13 illustrates a zero-padding example, where the input was padded with zeros so that the kernel could be slid across the input with a stride of 2.

In the activation function layer, the linear feature map resulting from the convolution is fed to a non-linear activation function to introduce nonlinearity. As mentioned previously, there are many types of activation functions, but currently the most commonly used on CNNs is the ReLU.

A pooling layer is a non-linear form of down-sampling. It is used to modify the output of the previous layer, reducing it in size so that the number of parameters, memory footprint, and the amount of computation needed are reduced. The pooling function replaces the output value of a certain location on the feature map with a summary statistic from neighboring outputs. This causes a translational invariance since when a given input is changed by a small amount, the pooled outputs do not change [22].

The max pooling [34] operation reports the maximum output within a rectangular neighborhood of a certain size, called pool size. The average pooling layer operation reports the average output within a rectangular neighborhood. Recently, average pooling layers have fallen out of use, with max pooling being used most of the times. The two operations are shown in figure 14, assuming a pool size 2×2 and a stride of 2.

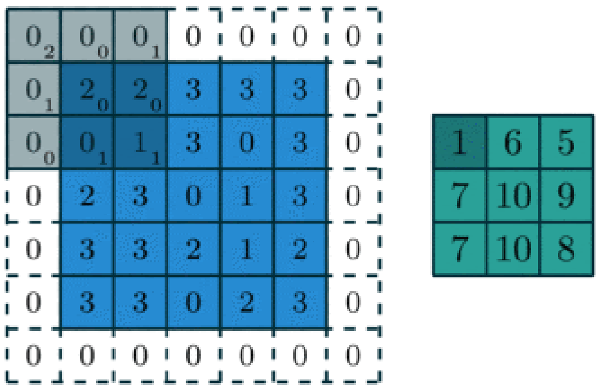

Figure 13: Example of zero-padding operation.
In white, it is represented the added zero padding, in grey the kernel, in blue the input data, and in cyan the output from the convolution with zero-padding. Taken from [33].

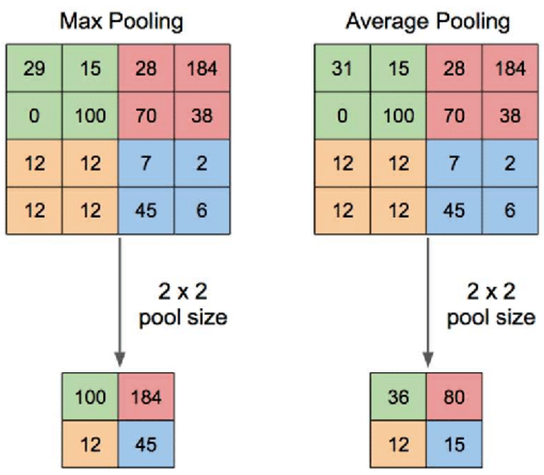

Figure 14: Max polling and average polling operations.
Taken from [35].

Fully connected layers are the normal flat feed-forward neural network layer. They are used at the end of the network after extracting and consolidating the features to make the final prediction of the network (network classification section on figure 11).

Convolutions, in addition to making it possible to work with images of varying sizes while maintaining the same number of parameters, also employ three important ideas that help to differentiate themselves from other neural networks:

Sparse interactions: while in traditional neural networks the matrix multiplication performed in the output units require an interaction with all previous input units, in typical CNNs (where the kernel is smaller than the input image) the output unit only interacts with the input units around which the kernel is located. This makes it possible to detect important features that only occupy a subset of pixels of the total image; for example, in the detection of edges in an image, the kernel will only focus on detecting the edges based on the pixels located where the kernel is and not in the entire image. Another advantage is efficiency, since the kernel is smaller than the image, it is necessary to save fewer parameters, which reduces the memory needed to run the model and reduces the number of operations needed to calculate a given output [22].

Parameter Sharing refers to the fact that the same parameter present in the kernel is used in various locations of the input, as the detection of a particular feature can be useful in different spatial positions of an image. This is something that does not happen in ANNs since each element in the weight matrix is used only once to calculate a given output in relation to a given input. This sharing of parameters in CNNs results in a lower number of parameters and reduces the computational power required [22].

The **Equivariant representation** property is achieved thanks to the polling layers, which prevent small changes in the input from altering the output, and the Parameter Sharing that is used in convolutions, since translation changes in the input do not affect the output [22].

2.5 Common CNN architectures

In this section, given their relevance to the present thesis, three common CNN architectures, and some of their variants, will be described.

2.5.1 *VGG*

The VGG [36] is a CNN that consists of 16 or 19 convolution layers interleaved with pooling layers and finishing with a FC layer, giving it an uniform architecture (figure 15).

Contrarily to previous networks, VGG takes advantage of an important aspect of CNNs, depth. Indeed, instead of relatively large windows and strides, VGG opts for smaller kernel sizes (3×3) and a stride of 1, taking advantage of the network's larger depth to extract more complex features.

After each convolution layer, a ReLU is used as an activation function to introduce non-linearity. In-between the convolution+ReLU layers, there are periodically max-pooling layers using a 2×2 pixel window with stride of 2, which allows the image size to be halved.

Since the network was designed to classify 1000 labels, at the end of the VGG, there are three FCs, the first two having 4096 channels each and the third having 1000 channels, one channel for each output class.

The advantages present in this architecture in comparison to the other competing models were: the use of several convolution layers with very small receptive fields (3×3), which allows to have a receptive field equivalent to only one convolution with a larger filter size (for example 7×7), with the advantage of being able to incorporate more ReLUs; and the use of small size convolution filters (3×3) that allows VGG to have a greater number of weight layers, improving the network's capabilities, since it can extract more complex features, but maintaining a smaller number of parameters in comparison with a convolution with a larger filter size.

2.5.2 *MobileNet*

MobileNet networks were designed to make it possible to run deep networks on mobile devices. So far there are three versions of the MobileNet, but in this section, we will talk about only two: MobileNetV1 [37] and MobileNetV2 [38].

The general idea behind MobileNetV1 is the use of depthwise separable convolutions that allow to reduce the model size and complexity. They were introduced by [39] and subsequently used in the first layers of the Inception models [40] to reduce the computation needed in the first layers of the network. In a normal convolution layer, a convolution filter is applied to all input channels, that filter is slid across the input, combining through the operation of the convolution all channels into one, as shown in figure 16.

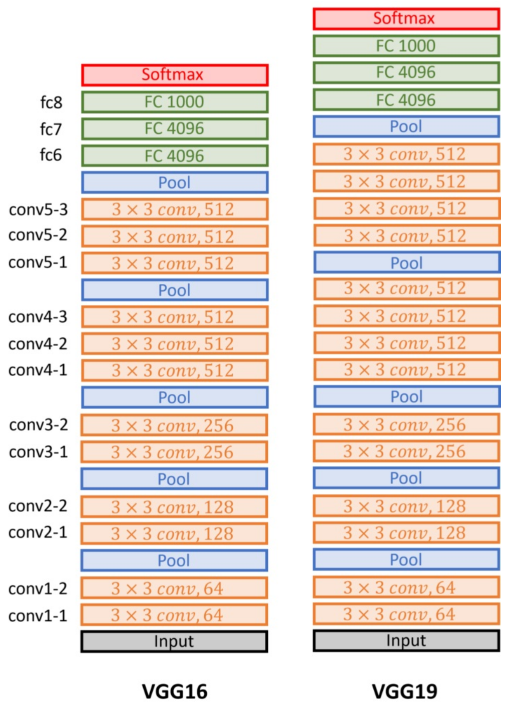

Figure 15: Layers of VGG16 and VGG19.
Source: http://datahacker.rs/.

MobileNetV1 uses this standard convolution only once, in the first layer of the network, using in the following ones a depthwise separable convolution. A depthwise separable convolution is a combination of two different types of convolution: a depthwise convolution and a pointwise convolution.

Unlike normal convolutions, a depthwise convolution does not combine the input channels, it only performs the convolution operations on each separate channel, as shown in figure 17, with each channel having its own set of weights. The purpose of this convolution is to filter each input channel, to detect some type of feature.

The depthwise convolution is followed by the pointwise convolution. The purpose of the pointwise convolution is to combine all of the output channels of the depthwise convolution, as depicted in

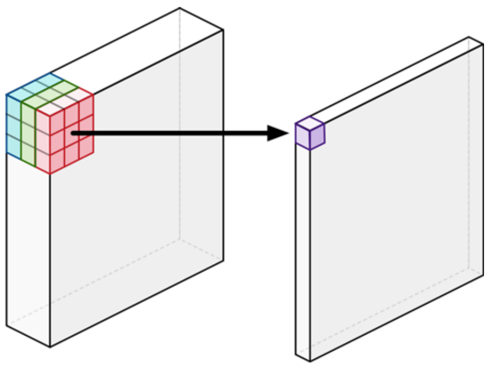

Figure 16: Example of a regular convolution.
Source: https://machinethink.net/.

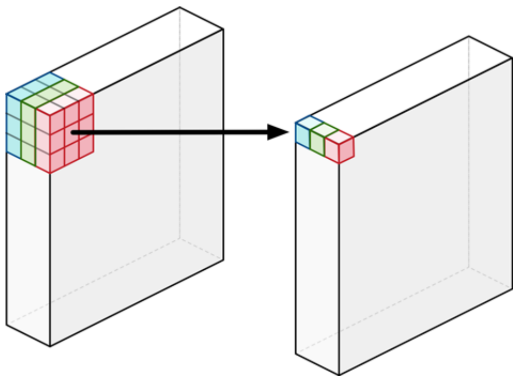
Figure 17: Example of a depthwise convolution.
Source: https://machinethink.net/.

figure 18, so that new features can be created. Pointwise convolution consists of applying a normal convolution, but with a 1×1 filter.

Together, the depthwise convolution followed by the pointwise one allows filtering and combining features in two separate steps, unlike normal convolutions that do everything at once. The advantage of using a depthwise separable convolution over a normal convolution is that although both present a similar result, depthwise separable convolutions contains less complexity, since they have fewer multiplications and additions, and have fewer parameters thus requiring less computational work. Another advantage present in the depthwise separable convolution is the possibility to apply batch normalization (BN) (a normalization of the input feature maps by re-centering and re-scaling them [40]) and ReLU twice instead of once.

MobileNetV1 has 28 layers (counting depthwise and pointwise convolutions as separate layers). The first layer is a standard convolution with a stride of 2 to reduce the spatial resolution of the image. The following layers are composed of repeated depthwise separable convolutions. Downsampling is handled with a strided convolution in the depthwise convolution and in the first convolution of the first layer. After each convolution, a BN layer is applied followed by a ReLU. Then, an average pooling layer is applied to reduce the spatial resolution to 1 and allow the use of a fully connected layer and a softmax to predict the output, as described in table 1.

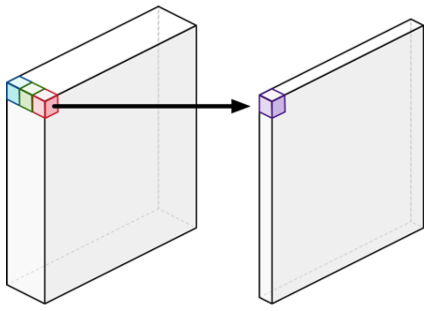
Figure 18: Example of a pointwise convolution.
Source: https://machinethink.net/.

MobileNetV2 expands the MobileNetV1 idea by introducing inverted residuals connections and linear bottlenecks through the bottleneck residual blocks, as shown in figure 19, further reducing the parameters and mathematical operations required.

Bottleneck residual blocks are made up of three convolution blocks: an expansion layer, a depthwise layer, and a projection layer, and a residual connection that connects the expansion layer input to the output of the projection layer.

An expansion layer is the first layer of the block, consisting of a 1×1 convolution in which the purpose is to expand the dimensionality of the channels so that the depthwise layer has more information to filter. The expansion factor is a hyperparameter that determines how many times the data is expanded, and has a default value of 6.

The depthwise layer has the same purpose as in the MobileNetV1 version, in which it filters the data received through a 3×3 depthwise convolution, resulting in an output with the same number of channels as the input.

The depthwise layer is followed by the projection layer, whose purpose is to compress the number of channels that come from the depthwise layer. It is called the projection layer since it projects data

Table 1: MobileNetV1 architecture

Type / Stride	**Filter Shape**	**Input Size**
Conv / s2	3×3×3×32	224×224×3
Conv dw / s1	3×3×32 dw	112×112×32
Conv / s1	1×1×32×64	112×112×32
Conv dw / s2	3×3×64 dw	112×112×64
Conv / s1	1×1×64×128	56×56×64
Conv dw / s1	3×3×128 dw	56×56×128
Conv / s1	1×1×128×128	56×56×128
Conv dw / s2	3×3×128 dw	56×56×128
Conv / s1	1×1×128×256	28×28×128
Conv dw / s1	3×3×256 dw	28×28×256
Conv / s1	1×1×256×256	28×28×256
Conv dw / s2	3×3×256 dw	28×28×256
Conv / s1	1×1×256×512	14×14×256
5× Conv dw / s1	3×3×512 dw	14×14×512
5× Conv / s1	1×1×512×512	14×14×512
Conv dw / s2	3×3×512 dw	14×14×512
Conv / s1	1×1×512×1024	7×7×512
Conv dw / s2	3×3×1024 dw	7×7×1024
Conv / s1	1×1×1024×1024	7×7×1024
Avg Pool / s1	Pool 7×7	7×7×1024
FC / s1	1024×1000	1×1×1024
Softmax / s1	Classifier	1×1×1000

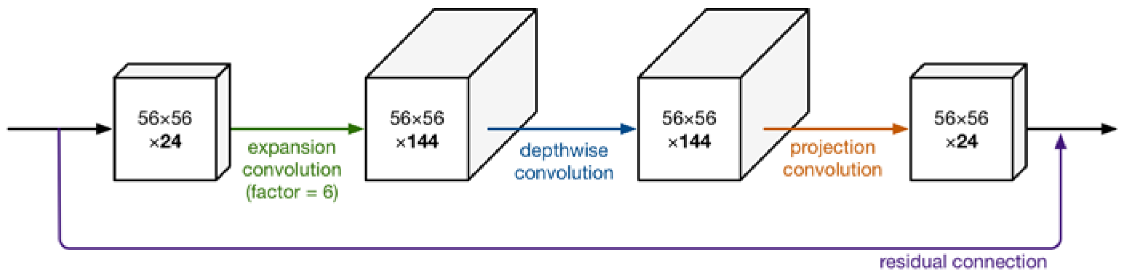

Figure 19: Example of a bottleneck residual block.
Source: https://machinethink.net/.

from a high dimension to a lower dimension. This type of layer is called the bottleneck layer, because it reduces the amount of data that flows through the network, saving computational resources. This is accomplished through the convolution with a 1×1 filter in which the number of output channels is less than the number of input channels.

Each layer in the bottleneck block is followed by BN and the ReLU activation function, except the projection layer, in which the activation function is not applied, making this bottleneck a linear bottleneck once the output of the last convolution is linear. An activation function is not used in this last convolution as the tests performed in the original paper have shown that applying a non-linear function to the low-dimensional data destroys useful information.

One of the novelties introduced in MobilenetV2 was the inverted residual connections in the bottleneck residual blocks. Inverted residual connections are residual connections also called inverted skip connections that, instead of following a wide→narrow→wide approach concerning the number of channels, it follows narrow→wide→narrow approach, as depicted in figure 19, being more efficient and obtaining better results. As in ResNet [41], residual connections connect the beginning and the end of a convolution block through a skip connection. This allows the gradients to flow through the network directly, without having to go through nonlinear activation functions that by nature tend to cause the gradients to explode or vanish.

The advantage of MobilenetV2 over MobilenetV1 is the replacement of separable convolutions by the residual bottleneck, which allows the number of channels to remain relatively small throughout the network, thus reducing the calculations made by convolutions. To combat the fact that it is not possible to extract a lot of information from low-dimensional data, the expansion layers decompress the data to expand the available information (by increasing the number of channels) for the filtration performed on the depthwise layer, and then filtered data will be compressed again in the projection layer to reduce the number of channels. The network will be trained to compress and decompress the data in the best possible way through the parameters present in the filters of these layers.

The MobilenetV2 network consists of 20 layers, the first being a convolution with 32 filters, followed by 17 bottleneck residuals in series that connect to a 1×1 convolution, as shown in table 2.

The first bottleneck residuals block is slightly different, as it contains an expansion rate of 1 instead of 6. There are two types of stride applied to the bottleneck residuals, stride = 1 to maintain the

Table 2: MobileNetV2 architecture

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1×1	-	1280	1	1
$7^2 \times 1280$	avgpool 7×7	-	-	1	-
1×1×1280	conv2d 1×1	-	k	-	

t: expansion factor; c: number of channels; n: number of repetitions; s: stride.

dimensionality of the data and stride = 2 to do the downsampling. Residual connections are applied only in blocks where the number of input channels is equal to the number of output channels.

2.5.3 *UNet*

UNet [42] is a fully convolutional network (FCN) designed with the task of image segmentation problems in mind, especially segmentation of biomedical images.

Image segmentation is a practice used in the computer vision field in which the objective is to assign a label to every pixel present in an image with the class that corresponds to what the pixel represents. The output of such classification is a high resolution image (usually the size of the input) in which each pixel, that represents some part of an object of a class, is classified as being that class. Unlike other object detection models, image segmentation allows visualization of the contours of classified objects.

UNet presents an encoder-decoder type architecture, in which the encoder consists of a contracting path that can capture the context of the images, and the decoder consists of an expansion path symmetric to the contracting path that allows adding spatial information to the features. The name of this network results from the similarity of the architecture to the letter U, as shown in figure 20.

The contraction path consists of multiple levels, each with two consecutive convolutions with a 3×3 filter each followed by a ReLU, followed by a 2×2 max-pooling with stride 2 to halve the data. After each downsampling step, the number of channels grows to double. The purpose of the contraction path is to extract local features, without retaining much spatial information.

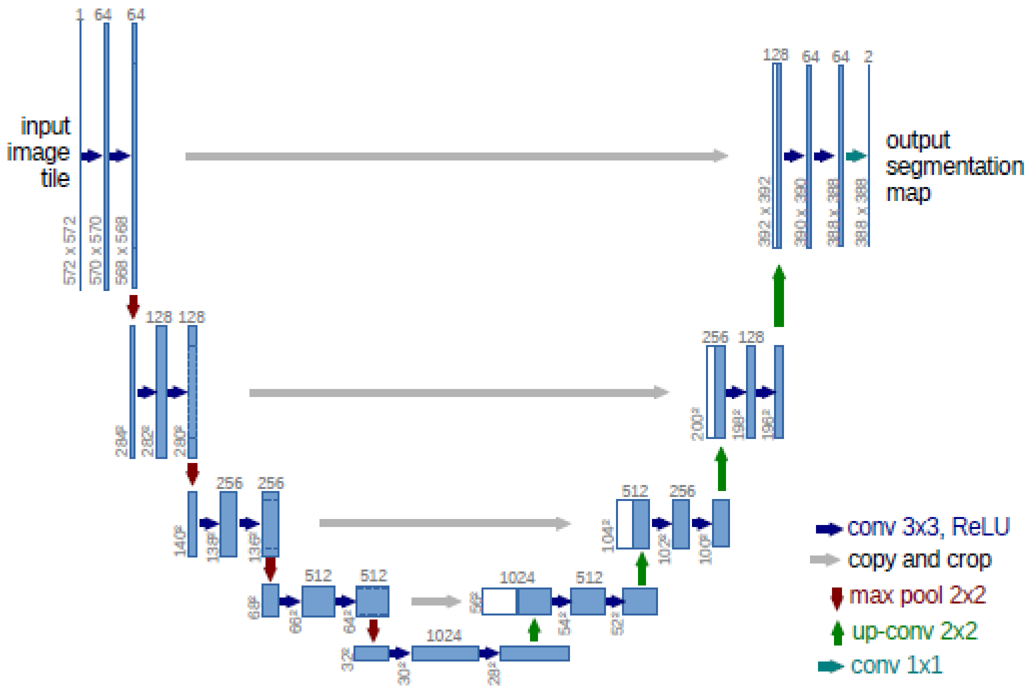

Figure 20: UNet architecture.
Taken from [42].

The expansion path consists of layers that contain a 2×2 up-convolution also called deconvolution to do upsampling to double the resolution of the data, and two consecutive 3×3 convolutions each one followed by a ReLU. After each 2×2 deconvolution, the number of channels is reduced by half and the concatenation of the feature maps of the respective level in the contraction path is done through a skip connection, thus adding spatial information that had been lost during contraction. A 1×1 convolution is applied to the last layer. In total, UNet includes 23 convolution layers.

The transposed convolution, also called deconvolution, is an operation performed on the data that is equivalent to the inverse operation performed by convolutions. Although this layer can be replaced by a regular upsampling operation, since it has parameters that can be learned, deconvolution allows the network to learn how to upsample the data in a more optimized way. Figure 21 shows an example of a decovolution with padding and stride 2, and a kernel of 3×3, where the blue squares are the input, and in cyan is the output.

UNet++ [43] expands UNet's idea by introducing deep supervision, redesigned skip pathways, and adding dense skip connections, with the ultimate goal of improving segmentation accuracy.

The redesigned skip pathways (shown in green in figure 22) consists in the introduction of convolutions in the skip connections so that the semantic gap between the feature maps on the contracting path and expanding path are reduced, decreasing the complexity of the optimization problem at hand. All convolutions used on skip pathways use a 3×3 kernel size.

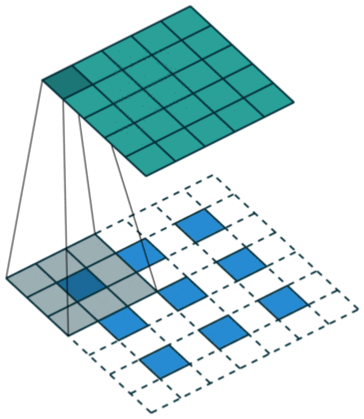

Figure 21: Example of a deconvolution operation.
Convolving a 3×3 kernel over a 5×5 input padded, with a 1×1 border of zeros and using stride 2. Taken from [33].

The use of dense skip connections (shown in blue in figure 22), which is inspired by DenseNet [44], has the purpose of gradually propagating the high resolution feature maps coming from the encoder to the convolution blocks present on the skip pathways, improving accuracy and gradient flow.

Finally, deep supervision (highlighted in red in figure 22) allows the model to be pruned to adjust its complexity, balancing the inference time (by selecting one of the output segmentation branches) and its performance (either using one of the branches or averaging the output of all).

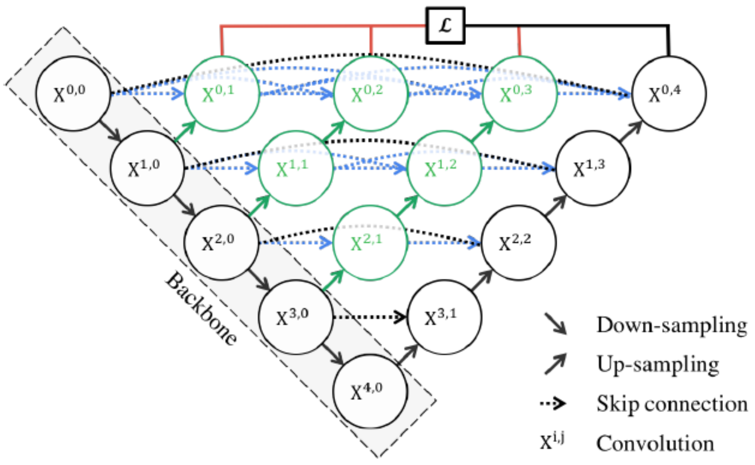

Figure 22: UNet++ architecture.
The new skip connections are shown in green, the dense skip connections are shown in blue and the deep supervision is highlighted in red. Taken from [43].

2.6 Keypoint/Landmark detection

In order to be able to calculate the indices related to the diagnosis of the Pectus Excavatum, we raised the hypothesis of using keypoint/landmark detection methods to automatically determine the correct CT axial slice and the positions of the keypoints in it. Although no similar studies exist in the context of PE (or even using other strategies towards the same goal), keypoint/landmark detection methods are widely used in several areas of computer vision, including the medical field. Most of these methods depend on CNNs to return confidence maps (heatmap regression-based methods, in which the maximum of the heatmap corresponds to the keypoint location) or the coordinates of the keypoints' positions themselves (numerical regression-based methods, based on FC layers).

To fill the gap between numerical regression-based methods and heatmap regression-based methods, Nibali et al. [45] proposed in 2018 a new approach to infer numerical coordinates from confidence maps called Differential Spatial to Numerical Transform (DSNT). This approach calculates the center of mass of the network's standardized heatmap, resulting in a numerical coordinate (figure 23). The DSNT layer offers, without adding trainable parameters, a good spatial generalization, something that is not present in the numerical methods due to the FC layer. It also allows the network to be fully differentiable since the output is the keypoints' locations. The possibility to calculate the loss directly from the predicted coordinates, thus propagating the network's gradient from the output to the input image, is something that is not possible with heatmap-based approaches (in which the final position is obtained through the use of the argmax function on the heatmap, an operation not differentiable) (figure 23-A). The properties of each regression method can be seen in table 3.

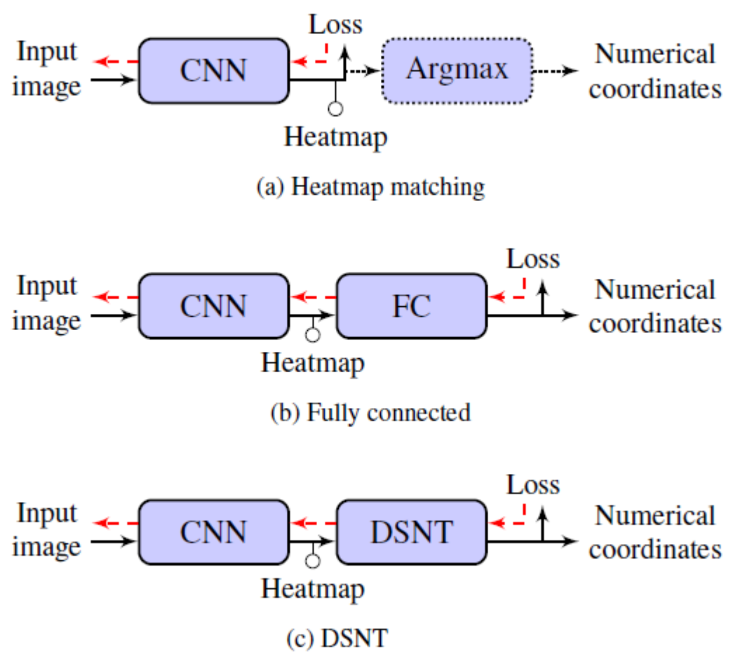

Figure 23: Comparison of coordinate regression model architectures.
The arrows indicate inference (black) and gradient flow (dashed red). Adapted from [45].

Table 3: Comparison between different keypoints' regression-based methods

	HM	FC	DSNT
Fully differentiable	✗	✓	✓
Spatially generalizable	✓	✗	✓
No parameters	✓	✗	✓
Good for high-res output	✓	✗	✓
Good for low-res output	✗	✓	✓
Direct coordinate loss	✗	✓	✓

Desirable properties of heatmap matching (HM), FC output , and DSNT. Taken from [45].

This section will describe some studies that have been carried out with one of these types of regression methods.

In 2017, Cao et al. [46] presented a method that provides real-time multi-person 2D pose estimation, where a bottom-up approach is used. A convolution network analyzes the input image (in this case, the first 10 layers of VGG19) and different image features are extracted. Then, a two-brach multi-stage CNN processes the resulting feature maps. The first branch provides 2D confidence maps of each joint present in the image. The second branch provides Part Affinity Fields (PAFs), which correspond to 2D vectors that allow associating different joints with each other. A greedy algorithm then processes confidence maps and PAFs to obtain the position of the multiple people present in the images without penalizing code execution time.

Raaj et al., in 2019 [47], expanded the idea of the previous study by proposing an architecture that encodes and predicts the spatio-temporal field over a sequence of images. The VGG is used again as a backbone to extract the features present in each frame. Then through a recurrent convolutional neural network, the joints 2D heatmaps are generated and are established the connections between joints (PAFs) on each frame and the connections across frames as temporal affinity fields. These modules are called Recurrent Spatio-Temporal Affinity Fields. Each module ingests the outputs of the other modules of the current frame and from the previous frames, thus allowing them to detect and follow the joints' positions in a video in real-time.

In the medical field, the use of keypoint/landmark detection algorithms is advantageous since there is the possibility to automate the analysis of medical images (as CT, ultrasound or magnetic resonance images), benefiting health professionals when making diagnoses.

Wang et al. [48] purposed a framework which, through two UNet-based cross-connected branches, could simultaneously segment and locate fetal femurs in prenatal ultrasound. In order to reduce the processed volume, these two branches are fed by the features extracted by a previous UNet that locates the fetal femur region of interest (ROI) on the ultrasound. Since landmark localization tends to suffer from false positives, a loss based on the distance of the output heatmap with the label's Gaussian map is used to improve the location map. To improve the correspondence between segmentation and landmark localization, an adversarial module is also used.

The topic of slice detection using keypoint/landmark detection methods has also been addressed in the literature. In 2017, Belharbi et al. [49] proposed a system to automate the search for the slice that contains the third lumbar vertebra (L3) in a CT volume. The system was formulated as a numerical regression-based method, in which the CT is converted into a 2D image via maximal intensity projection (MIP), and it is processed by a CNN pre-trained in the ImageNet dataset.

Kanavati et al. [50] proposed in 2018 a system that tackles the same problem, but using a heatmap regression-based method with a 2D or 1D confidence map as output. The 2D MIP image generated from the CT is used to feed the network based on the UNet architecture in which the resulting output is a 2D confidence map. In this paper, it is proposed a variation of the same network in which the generated output is a 1D heatmap, thanks to the introduction of a global horizontal max-pooling along the UNet upsampling path. The network's 1D variation managed to obtain results slightly higher than the network that returned 2D confidence maps. Using the heatmap regression-based method, Kanavati et al. managed to have errors comparable to the two annotators of the study, having obtained state-of-art results for the task at hand.

In 2019, Galbusera et al. [51] proposed an automatic method to extract anatomical parameters from biplane radiographs of the spine by locating 78 anatomical landmarks. Two FCNs were used to locate the points' 3D position, namely a network to analyze the sagittal plane, and another to analyze the coronal plane. The networks presented a DSNT layer at the end in order to allow the network to be fully differentiable and obtain spatial generalization.

Another article using a DSNT layer was that of Andreassen et al. [52] with the purpose of automatically segmenting the mitral annulus from a 3D transesophageal echocardiography. Each 3D volume was decomposed into a set of 2D planes, taking advantage of the symmetry of the problem. An UNet network was trained to predict the mitral annulus coordinates from the input 2D echocardiographic image (for each 2D plane). By spinning the 2D heatmaps, it is possible to iteratively aggregate the coordinates of the points resulting from these heatmaps in a 3D space, being possible to construct a 3D curve segmentation of the mitral annulus from these points and allowing for a quick and automatic way to detect the mitral valve's position and orientation in 3D images.

3

METHODOLOGY

3.1 General Overview

The main goal of this thesis is to implement a framework that allows calculating with precision and accuracy the measures and indices that are regularly used during the diagnosis of patients with PE in order to determine their severity. To accomplish this objective, and based on the methods studied in the state-of-the-art, the work of this thesis will be divided into three fundamental blocks, which together form the pipeline of this framework, as seen in figure 24.

The first block of this pipeline aims to locate, from a single sagittal slice extracted from the CT, the axial slice that contains the greatest depression of the sternum. This block will be divided into two separate modules. The first module focuses on extracting and preparing the sagittal slice, by applying a MIP to the CT (figure 24-A). The second module contains a CNN that processes the extracted sagittal slice in order to return the slice axial location (figure 24-B).

The second block of this framework aims to identify the eight keypoints necessary to calculate the PE measurements and indices from each axial slice within a ROI centered on the axial slice returned by the previous block. This block is also divided into two modules: the first one is responsible for the extraction and preparation of the axial slices (figure 24-C), while the second one must identify, through a CNN, the keypoints present in each extracted axial slice (figure 24-D).

Finally, the third block of the pipeline is a post-processing block. In a first stage, an algorithm processes the keypoints in order to validate them and, if needed, corrects any errors present in the network's prediction (Figure 24-E). In a second stage, a final module corrects the keypoints' positions to guarantee the orthogonality of the measured distances, and selects the axial slice from the ROI with the lowest APD distance (figure 24-F), from which the PE measurements and indices are finally extracted.

The two networks used in the first two blocks of the pipeline are both based on UNet++. The advantage of using UNet++ in this work, in addition to the advantages already discussed in section 2.5.3, is the fact that the output has the same resolution as the input image, which in itself reduces the artefacts introduced in the final result when extracting the maximum value from the heatmap.

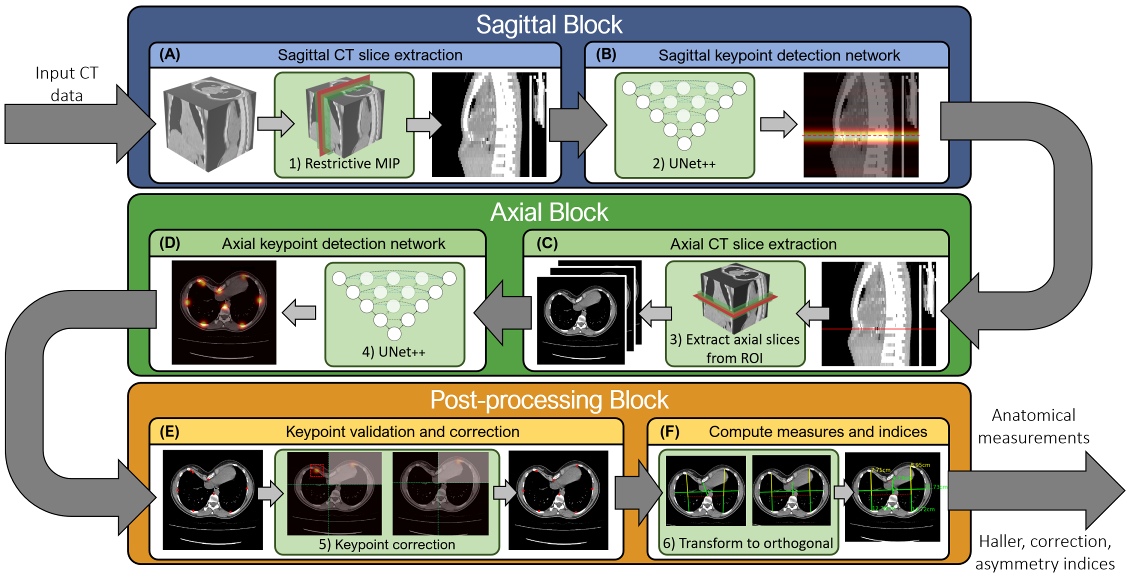
Sagittal Block
(A) Sagittal CT slice extraction
(B) Sagittal keypoint detection network
Input CT data
1) Restrictive MIP
2) UNet++
Axial Block
(D) Axial keypoint detection network
(C) Axial CT slice extraction
4) UNet++
3) Extract axial slices from ROI
Post-processing Block
(E) Keypoint validation and correction
(F) Compute measures and indices
5) Keypoint correction
6) Transform to orthogonal
Anatomical measurements
Haller, correction, asymmetry indices

Figure 24: Framework's pipeline overview.
This framework is divided in 3 blocks: (1) the sagittal block, consisting in a module (A) that extracts and prepares the sagittal slice to be used in module (B) for the detection of the axial slice with the greatest sternum depression; (2) the axial block, containing a module (C) that extracts and prepares the slices inside the ROI centered on the detected axial slice, which are used as input in module (D) to detect the heatmaps of the 8 relevant keypoints; and (3) the post-processing block, whose first module (E) computes the keypoints' positions from the corresponding heatmaps, validates and corrects them if needed, while the second module (F) extracts the PE measurements and indices from the axial slice with the lowest APD distance, while also ensuring the orthogonality of the measured distances.

The training and testing of this framework was carried out on a workstation with a Nvidia RTX 2080 Ti GPU, with 11GB of VRAM.

3.2 Sagittal Block

This section is dedicated to the presentation and explanation of the algorithms present in the sagittal block. The objective of this block is to detect, in a CT volume, the axial slice where the greatest depression of the sternum is located. This block is divided into two modules: sagittal CT slice preparation, and axial slice detection network (figure 24-A and B, respectively). This section will be divided into three parts, the first two referring to the modules constituting this block and a third with their implementation details.

3.2.1 *Sagittal CT slice preparation*

This module aims to prepare the data that will be fed to the network detecting the axial slice with the sternum's greatest depression. The proposed approach is to use a sagittal 2D image as the network's input, instead of feeding it with the full 3D CT volume. The latter approach would have substantial disadvantages, such as: the increase of memory and computational power required; a significant increase in the number of parameters on the CNN, which results in the need for a more extensive training dataset (which is a limiting factor in this thesis); and the effective reduction of the dataset size, as the use of the full 3D volume as input precludes one from generating multiple training images with distinct appearances from a single CT (an option available for a 2D input as several sagittal slices can be extracted from the same volume).

A problem resulting from using as input a 2D sagittal image is the fact that the depression of the sternum may not be centered on the CT volume. This factor makes the decision of which slice to extract challenging since the chosen slice may completely miss the depression of the sternum and, as a result, give to the network a non-representative image of the patient's chest. To mitigate this issue, it is proposed to convert the 3D CT volume into a 2D image through the use of a restrictive MIP. The MIP creates the 2D image by projecting the voxels that contain the greatest intensity in the line of the chosen projection plane, in this case, the sagittal plane, thus flattening the 3D volume into a 2D image but retaining relevant high-intensity information (such as the bones). Instead of projecting the entire CT in the sagittal plane, a restrictive MIP is here proposed to obtain an image with a clearer view of the sternum. In other words, the MIP is only applied to a ROI in the middle of the CT, avoiding the appearance of the lateral ribs in the final MIP image. Since the depression is often relatively centered in the CT, the ROI chosen for the MIP is 10 cm wide, ranging from -5 to 5 cm from the center of the CT (figure 24-A-1, the green highlight represents the ROI, and the red plane the middle of the CT).

Since the CT volumes are acquired with significant variability in slice thickness, the pixels of the 2D MIP image are normalized so that each pixel corresponds to an area of 0.5×0.5 mm^2, thus guaranteeing the consistency of the network's input.

Finally, in order to normalize and highlight the useful information present in the slice, a windowing function between 50 Hounsfield units (HU) and 400 HU (a typical soft tissue window) was applied (normalizing the intensities to the 0-255 range), eliminating the details within the lungs but allowing visualization of both bones and cartilages present at the end of the sternum, as well as the surrounding tissues and organs.

3.2.2 *Sagittal keypoint detection network*

This module aims to solve the problem proposed in this block, that is, from a given sagittal image of a CT to detect the axial slice where the sternum presents its greater depression.

To solve this problem, it is proposed to implement a CNN based on the UNet++ architecture [43] (figure 24-B-2), where the input is the image returned by the previous module (a 2D restricted MIP from the sagittal view) scaled to the size of 256×256 pixels. Of note, this size was chosen so that the memory needed to process the input is reduced without losing a lot of information. The pixel values are also centered and standardized with an average value of 0 and with a unit standard deviation to improve the network's convergence stability (by reducing the occurrence of exploding gradients). Following the approaches in [46][50], the network's output corresponds to two vectors with size 256×1, one being a 1D Gaussian map that represents the network's confidence in the slice location (figure 25), and the other representing a 1D background map, which corresponds to the inverse of the first output vector.

As aforementioned, the implemented network is based on the UNet++, which presents an encoder-decoder type of architecture, consisting of a contraction path and a symmetrical expansion path, with skip pathways with convolutions to reduce the semantic gap between the two paths, and dense skip connections for propagating the high resolution feature maps from the contraction path to the convolution blocks present in the skip pathways. Deep supervision is not implemented in the present network (figure 26).

The network's contraction and expansion paths are each composed of 5 levels, each containing two sequences of convolution (with kernel 3×3), BN (momentum = 0.99 and epsilon = 1×10^{-3}) and ReLU (see blue arrows in figure 26). Each block in the skip pathways is also composed by two sequences of convolution (kernel 3×3), BN and ReLU layers. The input of each one of these blocks results from the concatenation of feature maps from the skip connections (black arrows in figure 26), dense skip connections (brown arrows in figure 26) and the upsampled feature maps from the lower level (signaled with green arrows in figure 26). The number of kernels applied in each convolution is doubled at each level of the network, being the initial value of 32 kernels.

At the end of each contraction path level, a 2×2 max-pooling layer (red arrows in figure 26) is applied to halve the feature maps' resolution.

At the end of each expansion path level, and on the skip pathways, 2×2 upsampling with nearest-neighbor interpolation is applied to double the resolution of the feature map (marked in green figure 26).

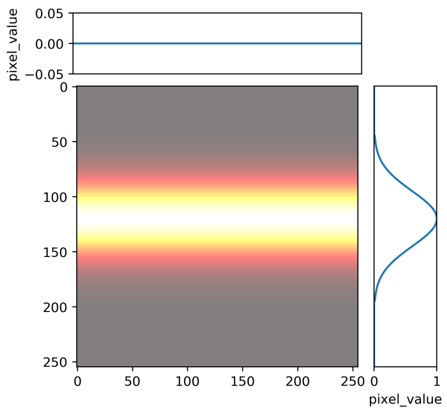

Figure 25: 1D heatmap consisting of a 1D Gaussian distribution on the y-axis.

An average pooling layer with a pool size of 1×256 is attached to the last level of the expansion path to horizontally flatten its output to 256×1 feature maps. This 2D/1D conversion block (dashed in red in figure 26) allows the network to focus on predicting the positioning of the keypoint along the y-axis only, as its position along the x-axis does not affect the selected axial slice. Lastly, one 1×1 convolution layer is used to aggregate the feature maps, resulting in an output with two classes.

3.2.3 *Implementation details*

The network was implemented in the Keras framework, and mini-batch gradient descent was used (batch size of 8) together with mean squared error (MSE) as loss and Adam as optimizer. The weights were initialized with a normal distribution as proposed by He et al. [53], and a l2-regularization (with weight 1×10^{-5}) was added to the loss. The initial learning rate used during training was 1×10^{-3}. This value is reduced to one third when there is no new loss minimum for 5 consecutive epochs. The training limit was set to 300 epochs, with an early stopping set for 10 consecutive epochs without loss minimization, thus allowing to reduce overfitting to the training data and reducing the training time. This model has 9166432 parameters.

The 1D heatmaps used during the training of the network were generated from the labelers' keypoints by applying equation (4), where y is equal to the pixel's y-axis position, $center_y$ is equal to the y-axis position of the keypoint given by the labeler, and σ is the sigma of the Gaussian heatmap, being in this case equal to 12.5 mm (converted to pixels following input resizing).

$$g_\sigma(y) = exp\left(-\frac{(y - center_y)^2}{2\sigma^2}\right) \tag{4}$$

During training, data augmentation was used to artificially increase the size of the training dataset. The ImageDataGenerator class present in Keras was used, so that it was possible to generate batches with data augmentation on-the-fly, while the model was training, thus saving disk space.

Besides the usual image transformations, including translation (between -15% and 15% of the image size), rotation (ranging from -5° to 5°) and scaling (with a factor between 0.95 and 1.05), the Keras class was modified to also augment the images by simulating various slice thicknesses (from 0.5 to 10 mm). The idea is to mimic the possible variability in CT acquisition parameters and, therefore, allow the network to generalize better, especially in cases with less common slice thickness values. This is accomplished by downsampling the training image along the vertical axis using bicubic interpolation, and then upsample it back to the original size but using nearest neighbors interpolation. All transformations have a 50% probability of being applied, and the generated sample is verified to ensure that the keypoint is still present within the image after augmenting it.

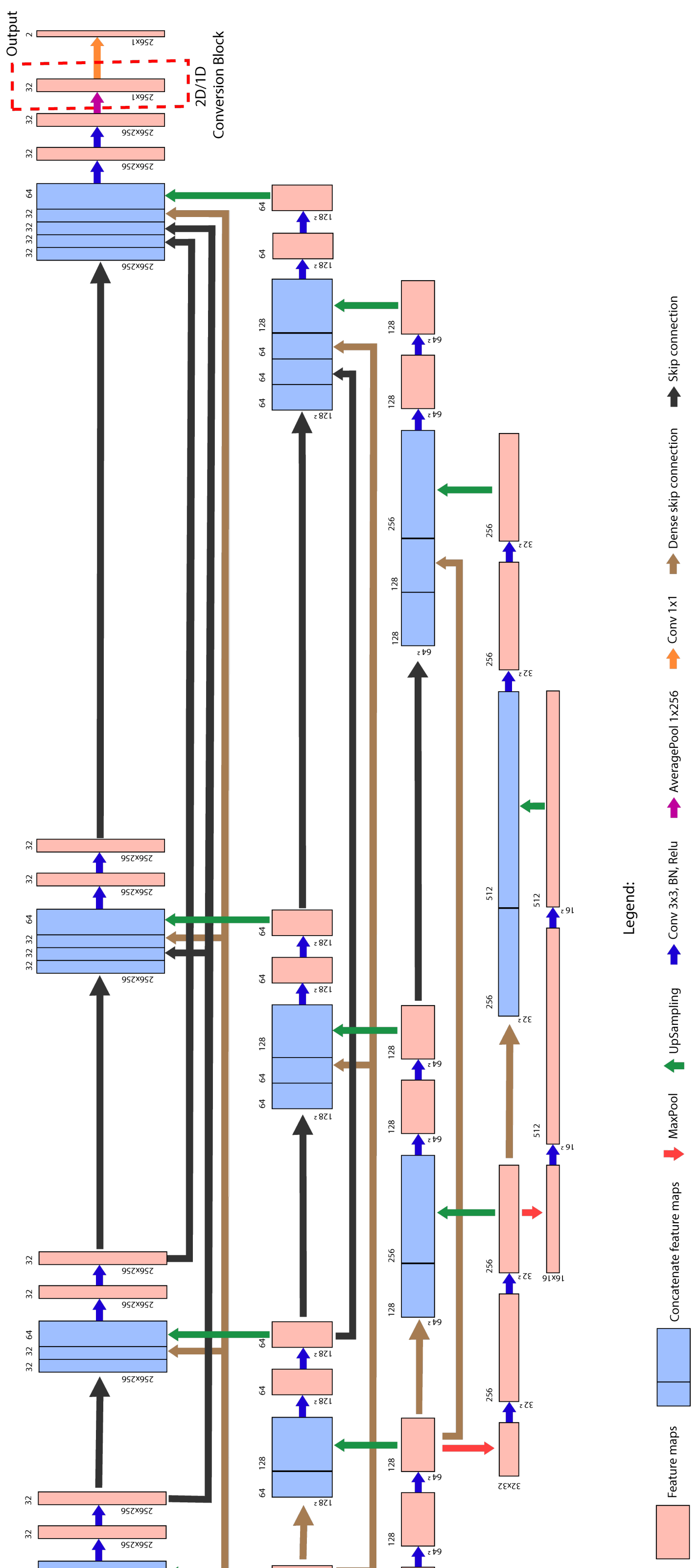

Figure 26: Axial slice detection network.
The network is based on the UNet++ architecture, with a 2D/1D conversion block attached at the output (dashed in red).

3.3 Axial Block

This section is dedicated to the presentation and explanation of the algorithms present in the axial block. The aim of this block is to generate a ROI centered on the axial slice given by the previous block, and then detect the eight keypoints required to measure and calculate the PE indices of each axial slice within the ROI. This block is divided into two modules: axial CT slice preparation, and the keypoint detection network (figure 24-C and D, respectively). This section will be divided into three parts, the first two referring to the two modules constituting this block and a third with their implementation details.

3.3.1 *Axial CT slice preparation*

This is the module responsible for the extraction and preparation of the axial slices contained in a ROI centered on the axial slice detected by the previous block in the pipeline. These axial slices will be fed to the network in the next module.

The first step is to extract the exact axial slice number from the 1D confidence map provided by the previous block. To extract the slice number, the 1D heatmap is first up-scaled from 256×1 pixels to the original image height using bicubic interpolation. With the up-scaled vector, the argmax function is applied to identify the pixel position with the highest value in the heatmap. Since we know the pixel size of the images (0.5 mm) and we know the slice thickness of the original CT, it is possible to calculate which slice has the network predicted.

After obtaining the number of the axial slice with the greatest depression of the sternum, a ROI with 2 cm is created centered on this slice (figure 24-C-3, the green highlight represents the ROI, and the red plane the axial slice returned by the previous block). Then, all axial slices inside this ROI are extracted from the CT to be analyzed by the keypoint detection network found in the next module (figure 24-C-3). As in the previous block of the pipeline, to make the input consistent across samples, the 2D images corresponding to the extracted slices are normalized so that each pixel corresponds to an area of 0.5×0.5 mm^2. Similarly, the same windowing function was also used.

3.3.2 *Axial keypoint detection network*

This module aims to detect, from an image corresponding to an axial slice of a CT, the eight keypoints that allow to extract the necessary measurements for the indices referred in section 1.3 (figure 5).

To address this problem, a CNN based on the UNet++ architecture is once again used (figure 24-D-4). The input of the network is each one of the axial slice images returned by the previous module, resized to 256×256. The values of the input pixels are normalized to achieve a zero average pixel value and a standard deviation of one. The network output has nine classes, each one with a size of 256×256 pixels. Eight of these nine classes are 2D heatmaps of each keypoint, each representing, through a 2D Gaussian, the network's confidence on the location of the corresponding keypoint (figure 27). Similar to the approach in [46], the ninth class is a 2D heatmap of the background of the eight classes (the inverse of the sum of the eight previous heatmaps).

The implemented network is identical to the network implemented in the sagittal block, with the difference of not having a 2D/1D conversion block at the end, as the output is 2D (figure 28).

3.3.3 *Implementation Details*

The weight initialization and regularization, optimizer, loss and training scheme are the same as the previous block (see section 3.2.3), and the network has a total of 9166656 parameters. Similarly, during training, 2D heatmaps were generated from the keypoints given by the labeler, with the difference that equation (5) was used to account for the 2D nature of the heatmap.

$$g_\sigma(x,y) = exp\left(-\frac{(x - center_x)^2 + (y - center_y)^2}{2\sigma^2}\right) \tag{5}$$

where x and y represent the pixel position, $center_x$ and $center_y$ the coordinates of the keypoint given by the annotator, and σ the sigma of the Gaussian distribution, being in this case equal to 7.5 mm.

Following the approach of the sagittal network, data augmentation was used on-the-fly, accounting for translation (between -5 and 5% of the image size), rotation (ranging from -5° to 5°) and scaling (with a factor of 0.95 to 1.05) transformations. As in the previous block, the probability of each transformation was set to 50%, and the generated sample is verified to guarantee that all keypoints are still present in the image after augmenting it.

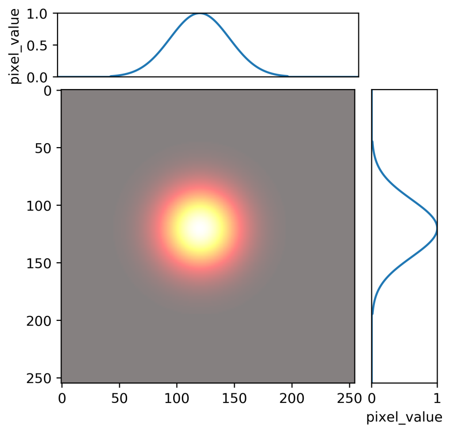

Figure 27: 2D heatmap consisting of a 2D Gaussian distribution over the image.

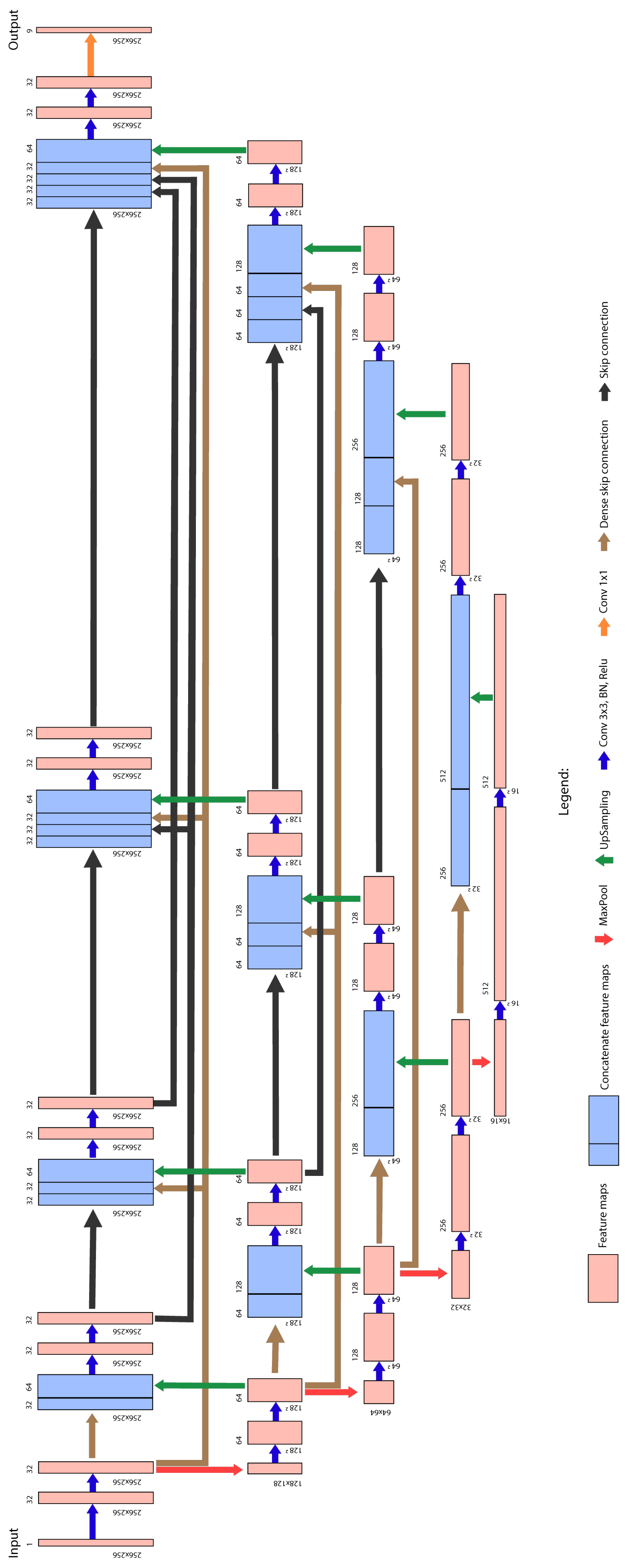

Figure 28: Axial keypoint detection network.
The network is based on the UNet++ architecture.

3.4 Post-Processing Block

Post-processing is the last block in the pipeline (figure 24). It is in this block that the keypoints detected by the previous block are validated and corrected (figure 24-E), and the distances between them measured and the PE imagiological indices calculated (figure 24-F).

This section is divided into two parts, the first describing and explaining the algorithm used to validate and correct the keypoints, and the second describing the algorithm that selects, among the axial slices in the region of interest, the one from which to extract the measurements and calculate the PE indices.

3.4.1 *Keypoint validation and correction*

In the previous block, the proposed network predicts eight heatmaps from each axial slice within a region of interest. These heatmaps usually have one single peak, with high confindence. Nonetheless, there is always the possibility that the network fails to correctly predict one of them, in which case the heatmap may have either a scattered Gaussian distribution with small confidence values or two separate high-confidence peaks. This may occur when the input image differs significantly from those images seen during training (a simulated example is shown in the output of module D in figure 24, where an additional heatmap peak is present on the left anterior side).

To solve this issue, it is proposed an algorithm that corrects these errors by dividing the image into four distinct regions and automatically detecting the misplacement of the detected keypoints (figure 29-B).

These four zones are delineated based on the center of mass of all keypoints (eight in total), whose coordinates are calculated using equation (6), with m_i being the keypoint's confidence value (the highest pixel value in the heatmap), and x_i and y_i the keypoint's coordinates on the x- and y-axes.

$$Kpt\_x_{cm} = \frac{\sum_{i=0}^{8} m_i x_i}{\sum_{i=0}^{8} m_i} \qquad Kpt\_y_{cm} = \frac{\sum_{i=0}^{8} m_i y_i}{\sum_{i=0}^{8} m_i} \tag{6}$$

After establishing the center of mass of all keypoints, and upon defining each keypoint's valid region with respect to the center of mass (figure 29-B), it is straightforward to validate whether the keypoints are within their specific region. If any keypoint is not within their region, the corresponding heatmap is zero-masked with a squared window centered on the detected peak, whose size is equal to 2 times the sigma of the Gaussian used during training (i.e., 15 mm), therefore deleting the wrongly predicted peak (Figure 24-E-4, the applied mask is dashed in red). After this, the center of mass is re-calculated and the keypoints validated again, being this process repeated until the argmax of all heatmaps are in their correct region. Note that this validation and correction algorithm is applied after

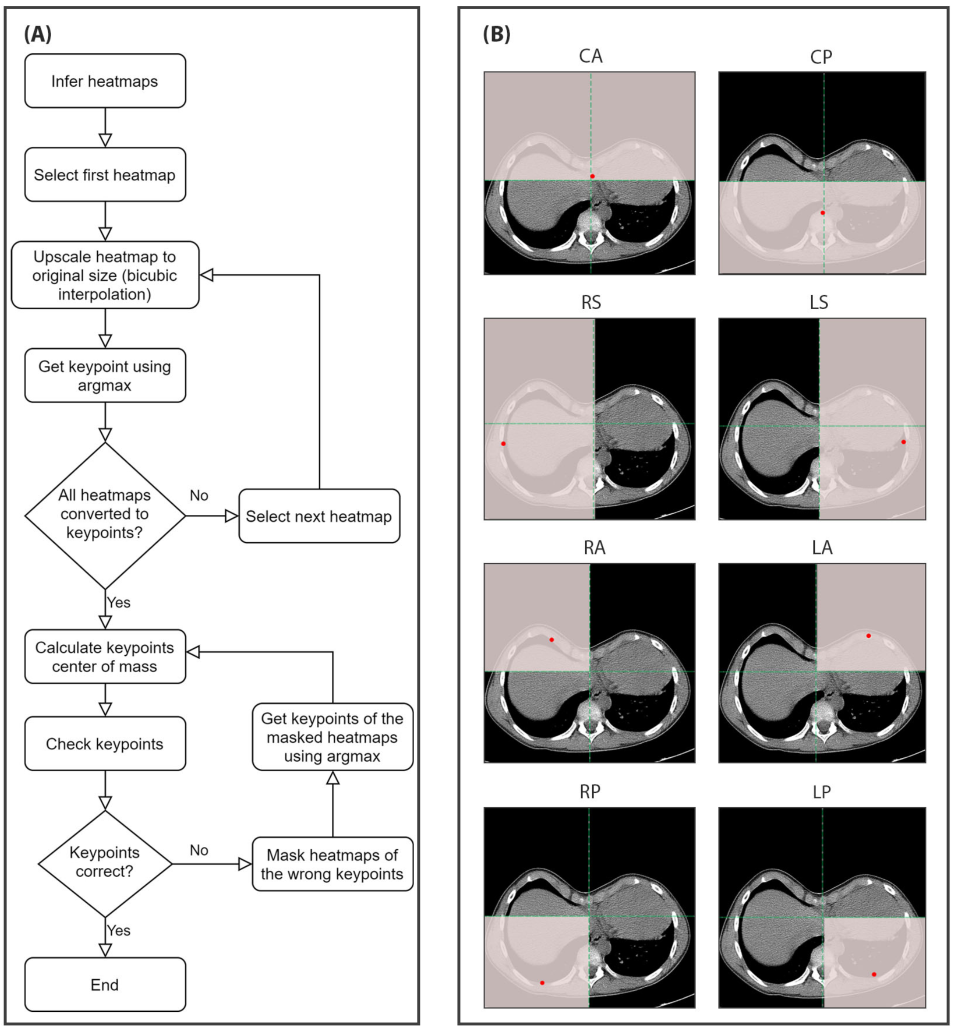
(A)
Infer heatmaps
Select first heatmap
Upscale heatmap to original size (bicubic interpolation)
Get keypoint using argmax
All heatmaps converted to keypoints?
No
Select next heatmap
Yes
Calculate keypoints center of mass
Check keypoints
Get keypoints of the masked heatmaps using argmax
Keypoints correct?
No
Mask heatmaps of the wrong keypoints
Yes
End
(B)
CA
CP
RS
LS
RA
LA
RP
LP

Figure 29: Flowchart of the correction algorithm (A), and valid regions for each keypoint (B).

scaling the heatmaps back to the original image size through bicubic interpolation. The proposed algorithm is summarized in the flowchart present in figure 29-A.

3.4.2 *Measurements and indices extraction*

This section describes and explains the algorithm used to select the correct axial slice from the ROI to measure the distances between the various keypoints validated by the previous module (figure 30-A), and calculates the imagiological indices of PE, namely the Haller, correction and asymmetry indices.

To calculate the measurements in figure 4, and following the experts' practice, it is first necessary to guarantee that the measured distances are orthogonal to each other. This is usually not the case, both due to the patients' ribcage geometrical variation and due to the network's own errors. To solve this issue, a geometrical analysis will be used to calculate five new points on every axial slice to guarantee orthogonality (figure 24-F-5).

Since all measurements should be orthogonal to the TCD measure (line between the RS and LS keypoints), one must calculate the vector that describes this link, named vector RL (figure 30-B). This vector also describes the patient's orientation in the CT bed.

After having established the vector RL, the first point to be defined is the new CA (blue point in figure 30-C). The distance between this point and the CP point will form the APD distance. The new CA is the result of the interception of the vector parallel to the vector RL that passes through point CA (dashed in red in figure 30-C), and the vector perpendicular to vector RL that passes through point CP (dashed in blue in figure 30-C).

After having calculated the APD distance on all axial slices from the ROI, only the axial slice with the lowest APD (i.e. the one with the deepest depression of the sternum) is kept and used to extract all measurements. In this sense, upon selecting the correct slice, and having both APD and TCD already computed, the Haller index can be immediately determined using equation (1) from section 1.3.

As described in section 1.3, to calculate the correction index one must compute the distance between the "corrected" sternum and the spine, called VCD (figure 4). To this end, one must compute two new keypoints (blue points in figure 30-D), termed mid-RP and mid-LP. These points are the result of the interception between the vector parallel to vector RL that passes through point CP (dashed in red in figure 30-D) and the vectors perpendicular to vector RL that pass through points RA and LA, respectively (dashed in blue in figure 30-D). Upon determining the two new keypoints, one computes the distance towards the respective anterior keypoints (RA and LA, respectively) and the maximum value corresponds to the VCD. Having both APD and VCD, the correction index is then calculated using equation (2) from section 1.3.

To calculate the asymmetry index (see equation (3) in section 1.3), one must compute both LAPD and RAPD, making sure they are both taken perpendicular to the TCD. In this sense, a new LP point is calculated as the result of the interception of the vector parallel to vector RL that passes through point LP (dashed in red in figure 30-E) and the vector perpendicular to vector RL that passes in the point LA (dashed in blue in figure 30-E). A similar analysis is performed to extract the new RP point (figure 30-F).

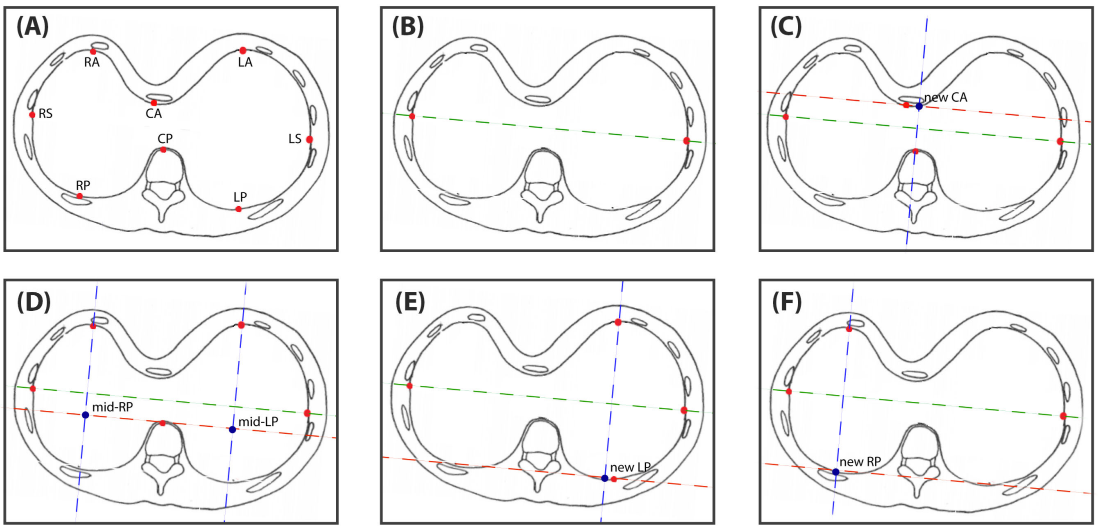

Figure 30: Keypoints given by the network (A), and geometrical analysis performed to identify 5 corrected points (B-F).

(B) The vector RL has the orientation of the patient's ribcage in the CT bed. (C) Calculation of the new CA keypoint. (D) Calculation of mid-RP and mid-LP keypoints. (E) Calculation of the new LP keypoint. (F) Calculation of the new RP keypoint.

4 RESULTS AND DISCUSSION

4.1 Dataset

4.1.1 *General description*

A total of 269 thoracic CTs were used to implement and evaluate the proposed framework, originating from 92 patients who underwent PE correction surgery.

All CTs were acquired as part of the preoperative planning performed prior to the correction surgery, and were retrospectively provided by Dr. Jorge Correia Pinto. The CTs came from 3 different hospitals in Portugal: Grupo Trofa Saúde (Braga), CUF Porto Hospital, and Braga Hospital, having been acquired with 12 different scanners (with various acquisition parameters and reconstruction kernels).

Each CT was captured with an image resolution and size ranging from 0.3346 to 0.8223 mm and 512×512 to 768×768 pixels, respectively. The slice thickness varied between 0.5 mm and 10 mm, with a total number of slices between 24 and 824.

Of the 92 patients, 77 were male (mean age, 15.66 ± 3.88 years; range, 10 - 36 years) and 15 female (mean age, 14.73 ± 3.04 years; range, 12 - 25 years). The dataset includes 17, 23 and 40 patients with a clinical assessment of mild, moderate and severe/extreme PE severity, respectively (no information was available for the other 12 patients).

Figure 31 exemplifies the dataset's variability in terms of subjects' positioning, PE severity, and image acquisition (slice thickness, reconstruction kernels, etc.).

4.1.2 *Dataset annotation*

All CTs were processed and analyzed using custom MATLAB (MathWorks Inc., USA) scripts, and were annotated by a pediatric surgeon using the Labelbox platform (Labelbox, CA, USA). In short, each CT was first analyzed to identify the sagittal slice that roughly corresponded to the center of the depression. In this slice, the expert identified the point with the greatest depression, corresponding to the axial slice in which the measurements should be made. Then, the region delimited by the most prominent area of the ribs' cage was identified, and multiple sagittal slices within this region (spaced 1 cm apart) were extracted. The keypoint provided by the expert was considered the ground truth for all sagittal cuts. By extracting multiple sagittal slices from one single CT, one artificially enlarges the dataset for training the corresponding network. In addition, given that this network receives as input the sagittal slice from the middle of the CT volume (which may not be the center of the PE deformity, as can be seen in figure 31-A), this approach increases the ability of the network to deal with images acquired from patients with asymmetric PE or those not centrally positioned in the CT bed.

A similar strategy was followed for the axial slices. Specifically, the valid region for PE bar placement (from the base of the heart to the base of the manubrium) was identified, and axial slices spaced 1 cm apart were extracted. For each slice, the same expert identified the eight relevant keypoints (figure 5).

4.2 Evaluation Metrics

The average keypoint distance error (KDE) was used to assess the performance of the sagittal keypoint detection network. In short, for each predicted slice, one measures the distance between the predicted slice and the ground truth slice. This distance was measured in mm to objectively account for the slice thickness of each CT.

Similarly, to assess the performance of the axial keypoint detection network, one averages the KDE for all keypoints in a given slice (measured in mm, as a 2D Euclidean distance). This metric was

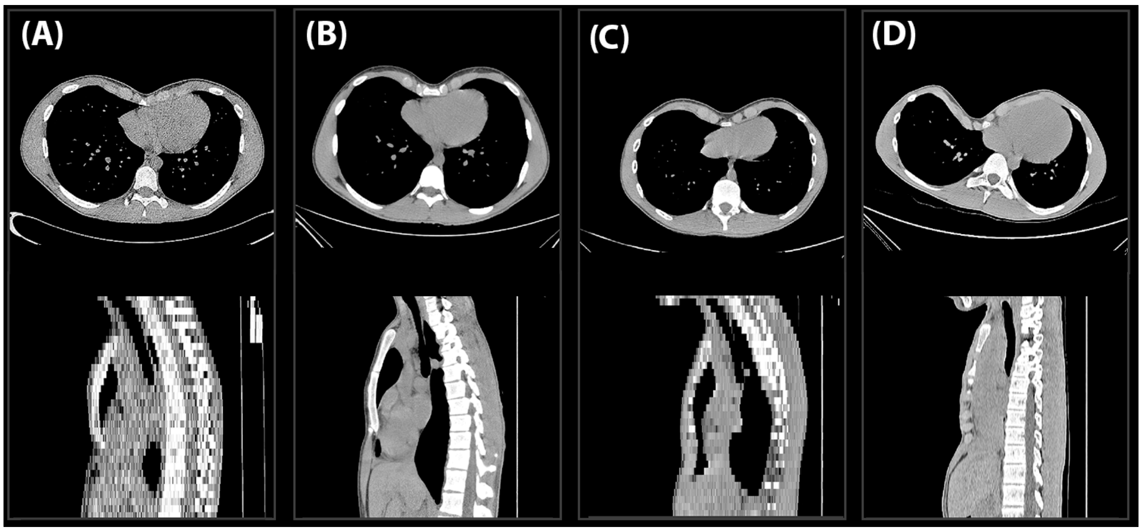

Figure 31: Example axial and sagittal CT slices from four patients.
The axial slices correspond to the slice with greater depression of the sternum, and the sagittal slice corresponds to the middle slice of the same CT. A clear variability in slice thickness (sagittal), PE severity (axial) and overall image quality (both) can be observed. For patient (C), note that the sagittal slice does not capture the region of greatest depression, given that the subject was not centered on the CT bed.

named as the mean keypoint distance error (MKDE) and is formally expressed by equation (7). The network's performance was reported as the average MKDE over the entire set of predicted images.

$$MKDE = \frac{\sum_{i}^{N} \|K_i - GT_i\|}{N} \tag{7}$$

where $K_i = \{x_i, y_i\}$ represents the coordinates of keypoint *i*, $GT_i = \{x_{\{i,gt\}}), y_{\{i,gt\}}\}$ represents the ground truth coordinates of keypoint *i*, N is the number of keypoints (8), and $\|\cdot\|$ is the Euclidean distance.

4.3 Network Performance

The dataset was divided into two groups: (1) train/validation set, with 74 patients (∼80%), used to train the networks and tune the models' hyperparameters; and (2) testing set, with 18 patients (∼20%), used for the final unbiased evaluation of the models. During training, a 5-fold cross validation was used, with the result representing the average of the evaluation metrics for the 5 validation folds. All splits were performed in a patient-disjoint manner, thus avoiding the presence of samples from the same patient (even if different CTs) in more than one group (i.e. each patient is either in the training, validation or test set only).

Since the axial block is the most complex one (larger variability in anatomic appearance and larger number of keypoints to be predicted) and the one with larger influence on the pipeline's performance, it will be presented first and will comprehend the evaluation of most hyperparameters/algorithmic choices. The analysis of the sagittal block will take into account the conclusions taken in the axial block, and will include the analyses done on its specific (or contrasting) hyperparameters/algorithmic choices.

4.3.1 *Axial Block*

Table 4 summarizes the average performance of the axial keypoint detection network on the 5 validation sets. The results obtained when replacing the proposed architecture (UNet++) by five other common CNN architectures are also presented. The details of each architecture are presented in Appendix A. All networks were trained under the same conditions (input size, heatmaps' sigma size, weight initialization, data augmentation, optimizer, loss and training scheme).

Through the analysis of Table 4, it is possible to conclude that the proposed model presented the best performance (i.e. lower MKDE), being the result consistent across all keypoints. Its superior performance may be linked to the fact that the network outputs a heatmap of the same size as the input, which mitigates the errors linked to the upscaling/interpolation operations. Among the other tested architectures, only UNet shares the same advantage (the output is 4 times smaller for

Table 4: Performance of the proposed axial keypoint detection network in a 5-fold cross validation, and comparison against five other CNN architectures

Model	MKDE	KDE$_{CP}$	KDE$_{CA}$	KDE$_{RS}$	KDE$_{LS}$	KDE$_{RA}$	KDE$_{RP}$	KDE$_{LA}$	KDE$_{LP}$
UNet++ (proposed)	2.99 ±1.18	1.29 ±0.82	2.90 ±2.56	3.45 ±5.58	4.15 ±3.30	3.09 ±2.46	2.81 ±2.19	3.15 ±3.66	3.09 ±2.46
UNet	3.18 ±1.64	1.32 ±1.42	3.31 ±2.69	3.90 ±7.07	4.28 ±3.26	3.25 ±2.45	2.85 ±2.15	3.44 ±5.11	3.10 ±2.45
MobileNetV2	3.17 ±1.21	1.42 ±0.94	3.25 ±2.82	3.67 ±2.80	4.35 ±3.37	3.32 ±4.35	2.88 ±2.10	3.23 ±3.67	3.22 ±2.48
MobileNetV1	3.40 ±1.47	1.34 ±0.89	3.80 ±4.23	4.45 ±3.32	4.55 ±3.59	3.31 ±2.58	2.98 ±3.75	3.43 ±5.42	3.30 ±2.68
VGG16	3.64 ±1.56	2.18 ±1.15	3.89 ±2.83	4.23 ±4.84	4.68 ±3.40	3.77 ±5.52	3.20 ±2.12	3.71 ±5.02	3.44 ±2.34
VGG19	3.62 ±1.51	2.22 ±1.16	3.68 ±2.71	4.30 ±6.47	4.70 ±3.33	3.56 ±2.45	3.42 ±5.16	3.61 ±2.65	3.49 ±2.36

Values are mean ± standard deviation, presented in mm.

MobileNets, and 8 times smaller for the VGGs). Another factor to be considered is the greater number of trainable weights, resulting from the kernel size (multiples of 32) and the additional convolutions on the skip pathways, that may increase the network's representational capability. As stated by the original authors [43], the modified skip pathways may also allow to reduce the semantic gap between contracting and expanding paths, decreasing the complexity of the optimization problem and ultimately leading to a better convergence and thus performance.

Figure 32 illustrates some example results obtained by the proposed model on samples from the validation set (in red), together with the respective ground truth (in green).

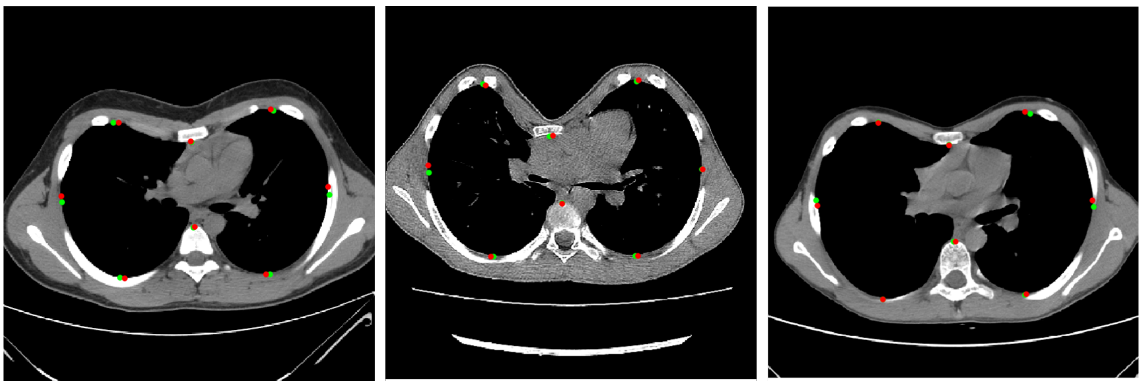

Figure 32: Axial keypoints' detection examples for three representative samples from the validation set. The red keypoints represent the proposed model's output, and the green keypoints the ground truth.

Figure 33 presents the influence of key algorithmic choices and training parameters (input size, type of windowing function used for image normalization, heatmaps' sigma size, and type of output) on the proposed model's performance. Having verified the non-normality in the distribution of the results, the non-parametric Wilcoxon matched-pairs signed rank test was applied with a significance level of 0.05 to verify if there is a statistical difference between the proposed model's parameters and their variations, being marked with $*$ the results with statistical significance.

Starting by analyzing the influence of the input image's size, it was possible to observe that there is a statistical significant difference between the tested input sizes (192×192 and 320×320, figure 33-A) and the proposed one. On the one hand, a smaller input image's size (192×192) represents an image with less spatial resolution that ultimately leads to worst localization capabilities. On the other hand, contrary to what could be expected, there is also a slightly higher average error when the size of the input image is larger (320×320). This may indicate that the kernel size (3×3) may be too small to capture meaningful features in this image size. Another hypothesis is the possibility that the number of levels present in UNet++ (5) is insufficient to extract sufficiently small feature maps on the last level to present accurate localization capabilities.

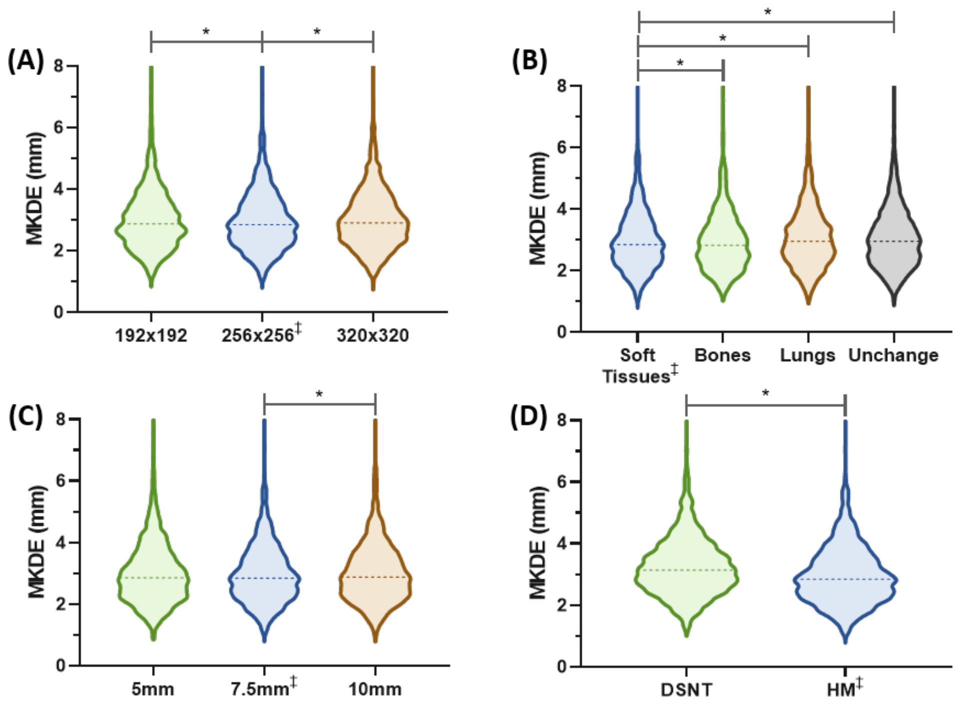

Figure 33: Influence of (A) input size, (B) type of windowing function for image normalization, (C) heatmaps' sigma size, and (D) type of output, on the performance of the axial block (assessed in a 5-fold cross validation).

* p<0.05, in a Wilcoxon matched-pairs signed rank test against the proposed parameter's value (marked with ‡).

In the tests performed to assess the influence of the input normalization (figure 33-B), three different windowing functions (soft tissues, bones, and lungs) or no windowing normalization (unchange) were considered. A statistically significant difference was observed between the proposed normalization approach (soft tissue) and the bone window, with the former presenting a superior performance. Although not initially expected, since a good level of detail of the bones is present in both window functions (soft tissues and bone), this result may indicate that the network relies not only on the bone structures but also on the cartilage and soft tissues within the chest. The lung window also showed a significantly worse result with a wide error variation, possibly due to the loss of details of the bone structures as a result of the saturation of their intensities when employing such window targeting lower HU values. When no normalization is employed, a significantly higher error was also observed, most probably due to the heterogeneous spectrum of tissue intensities across samples when normalizing the input to a 0-255 range (i.e. the same structure/organ has a distinct intensity value in different training images upon normalizing the intensity range with the default acquisition windowing function).

With respect to the heatmaps' sigma size, a significant difference was observed only between the proposed sigma size (7.5 mm) and the larger sigma (10 mm), with the smaller sigma (5 mm) presenting slightly higher average errors but without reaching the level of statistical significancy (figure 33-C). It is important to notice that this result cannot be extrapolated to other CNN architectures (in Appendix B), or even if a different input image size is used (data not shown). This result proves that the network needs a sigma with a size between 5 and 7.5 mm to converge well. Below this range, the network presents convergence difficulties due to limited gradient information provided by the ground truth heatmaps (data not shown). When the sigma is larger than 7.5 mm, the error increases because the heatmap starts to occupy a larger region, becoming more diffuse and not showing a defined peak, affecting the network's ability to accurately locate the keypoint. This phenomenon is even more significant with larger sigmas (data not shown). The same phenomenon was observed on other CNN architectures (see results in Appendix B).

When comparing the type of regression method (Figure 33-D), one observes that the heatmap matching regression method (proposed) obtained a significantly lower error when compared to the DSNT approach. Similar results were observed for other CNN architectures (Appendix B). In what concerns the proposed architecture, this result may be explained by the different approximation errors induced when interpolating the network's prediction to account for the mismatch between output's size (256×256, as the input) and the original CT image's size (between 512×512 and 768×768). Indeed, this operation confers a disadvantage to the DSNT method, since the interpolation operates on the keypoints' coordinates directly predicted by the network. On the contrary, the heatmap regression approach (proposed) predicts a 2D heatmap, that is first interpolated to the correct size and only after are the keypoints extracted by the argmax function.

Appendix C includes other experiments performed on the proposed model's parameters (augmentation, kernel size, output prediction without background class).

After evaluating and tuning the proposed model with the cross-validation results, an analysis was made with the test dataset (figure 34-A). Since several models were trained during cross-validation, all models were used to predict the test results and different approaches were employed to obtain the final prediction. In short, one compared the results obtained by all folds (seen as independent estimates), by the best fold only (upon computing the average error for each model), and 3 distinct strategies of combining the 5 folds' outputs (termed as ensembles), namely: (1) average of the keypoints' locations (Keypoint ensemble); (2) weighted average of the keypoints' locations using the network's score value (Weighted keypoint ensemble); and (3) averaging the heatmaps before applying the argmax function to extract the keypoints' locations (Heatmap ensemble). With a significance level of 0.05, the Friedman's test was performed to determine if there is statistically significant differences between the different strategies.

Through the tests carried out, it is possible to observe a statistically significant improvement for all ensemble strategies when compared to the best fold. This result proves that there are benefits in aggregating all models' outputs, which was expected since each model has been trained with a distinct set of training data and may have learnt different features to identify the keypoints. Between ensemble strategies, no significant difference was observed. Indeed, ensemble strategies that take into account the network's score (weighted keypoint ensemble and heatmap ensemble) did not show an

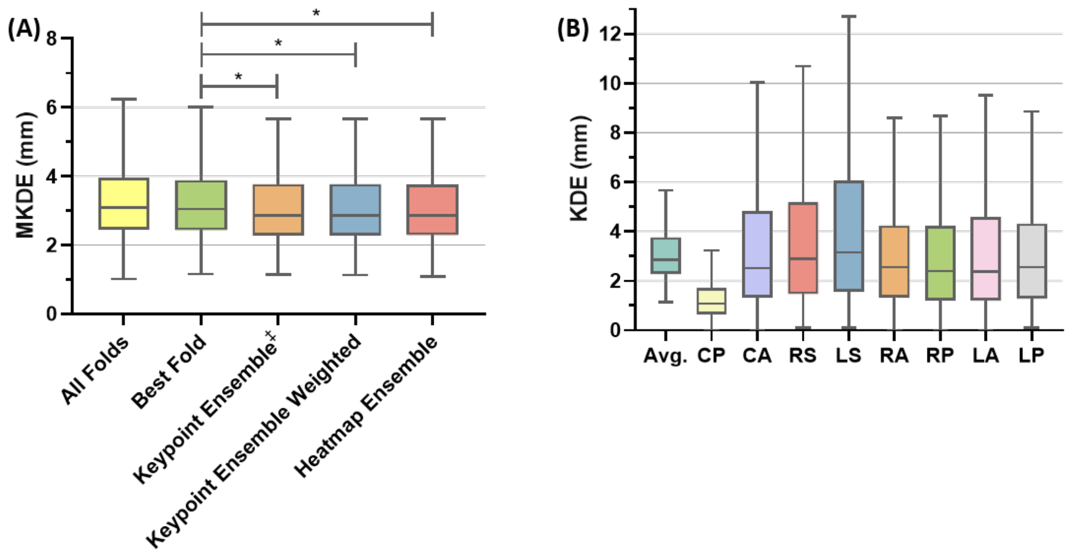

Figure 34: Axial block results in the test set.
(A) Results obtained for the best fold, and three distinct ensemble strategies. (B) Performance for each keypoint individually, and averaged, for the ensemble strategy with the best result on the test set (marked with ‡). * p<0.05 in a paired *t*-test between strategies.

improved performance, suggesting that there is no perfect correlation between the network's scores and the task error. Ultimately, the keypoint ensemble strategy was used for all other experiments.

One of the disadvantages of using ensembles is their computational load since it is necessary to run the 5 folds to analyze a single axial image. One possible solution to alleviate the computational load is to replace the traditional convolution blocks in the Unet++ architecture by separable convolution blocks (as proposed in the MobileNet architecture). Such hypothesis was tested on an additional experiment, and the model with separable convolutions obtained a MKDE of 3.37 ± 1.99 mm, which compares favorably to the MKDE of 3.31 ± 1.82 mm obtained when using the traditional convolutions. Another solution would be to use a fewer number of models to create the ensemble. When using only the 2 or 3 best folds, the model obtained a MKDE of 3.36 ± 1.81 mm and 3.33 ± 1.81 mm, respectively, which compares favorably to the proposed strategy (using the 5 folds). Although both hypotheses could be interesting when devising a software for clinical use, none of them were employed during the remaining experiments (including the clinical validation performed in section 4.4).

Figure 34-B presents the errors obtained by the proposed keypoint ensemble strategy for each keypoint independently. It is possible to observe that the keypoints CA, RS, and LS present a larger error, which may be explained by the also larger variability in positioning them (given the lack of exact anatomical landmarks). Indeed, in the case of CA, the keypoint must be positioned on the sternum but no consensus rule exists on where to position it (may be on the center or on the most posterior part). In the case of the RS and LS keypoints, the increased errors are also explained by the fact that the annotation of these keypoints on the sides of the rib cage varies greatly - since the rib cage is relatively elongated and flattened on the lateral walls, there is no exact place to position these points. This affects the LS point in particular since the PE is predominant on the right side, with the left side of the ribcage being frequently more elongated, thus giving more room for error.

Figure 35 illustrates some example results, on the test set, of the proposed network (red dots) compared to the expert's annotations (green dots).

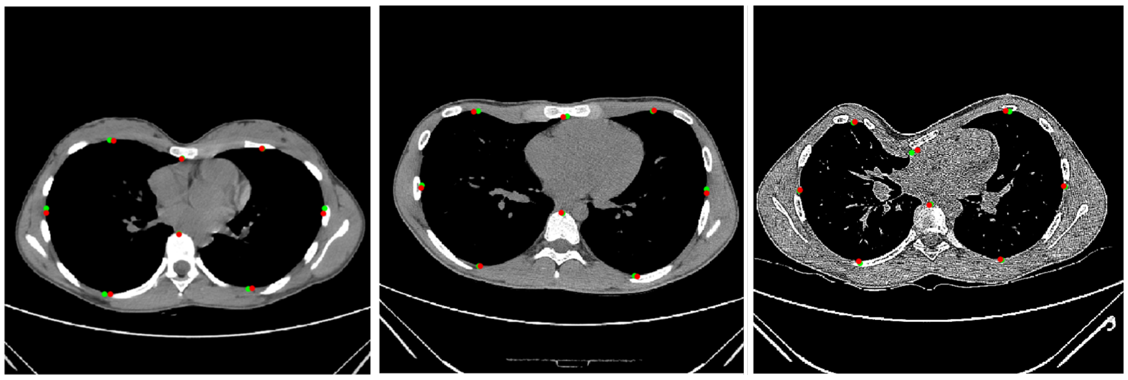

Figure 35: Axial keypoints' detection examples for three representative samples from the test set. The red keypoints represent the proposed model's output, and the green keypoints the ground truth.

4.3.2 *Sagittal Block*

The experiments performed on the sagittal block were based on the analyses' results obtained for the axial block since their architecture is very similar. The proposed method for detecting the keypoint corresponding to the axial slice with the greatest sternum depression had a KDE = 5.08 ± 4.64 mm over the 5-fold cross validation.

Figure 36 illustrates some example results obtained by the proposed model on samples from the validation set (red line) compared to the ground truth (green line).

As in the previous block, an analysis of some proposed method's key algorithmic choices and parameters (ROI width for MIP, type of windowing function used for image normalization, amount of data augmentation, and type of dimensionality reduction block) was carried out to evaluate the model's sensitivity to the defined values/choices (figure 37). In order to check if there are statistically significant differences ($p < 0.05$) between the proposed value/choice and its variations, the Wilcoxon matched-pairs signed rank test was performed in each experiment.

From such analysis, it was possible to observe that the ROI size used for MIP calculation during sagittal slice preparation is a critical parameter for the method's performance. Indeed, all variations presented a significant difference with respect to the proposed value (10 cm, figure 37-A). This result is expected since varying the MIP's ROI size directly influences the information available on the input sagittal slice, which can ultimately be useful or counterproductive to the detection. When looking to the influence that the ROI size has on the generated sagittal image (figure 38), one observes that neither very narrow or wide ROIs are helpful. The former is sensitive to the body positioning on the CT (with off-center slices potentially having insufficient information about the sternum and the point of its greatest depression; see first row) and the latter leading to reconstructed images with overlapping details (particularly the lateral ribs obstructing the visibility of the sternum; see last row). Hence, a median value is preferred to guarantee enough information to increase robustness to body positioning without compromising the visibility of the sternum.

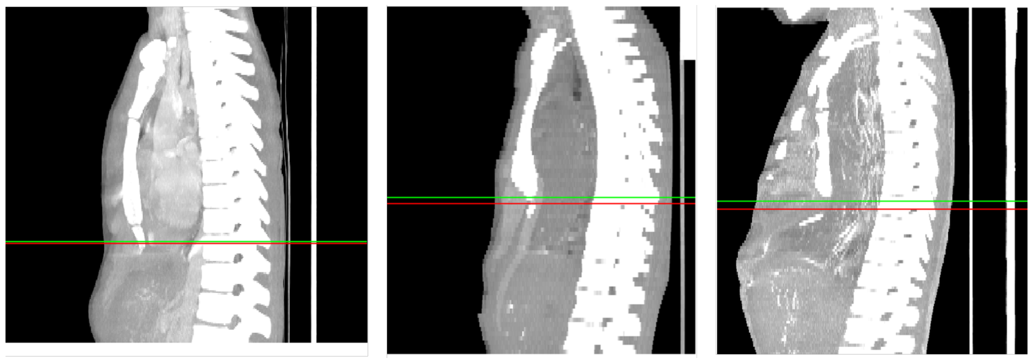

Figure 36: Sagittal keypoint's detection examples for three representative samples from the validation set. The red line represents the proposed model's output, and the green line the ground truth.

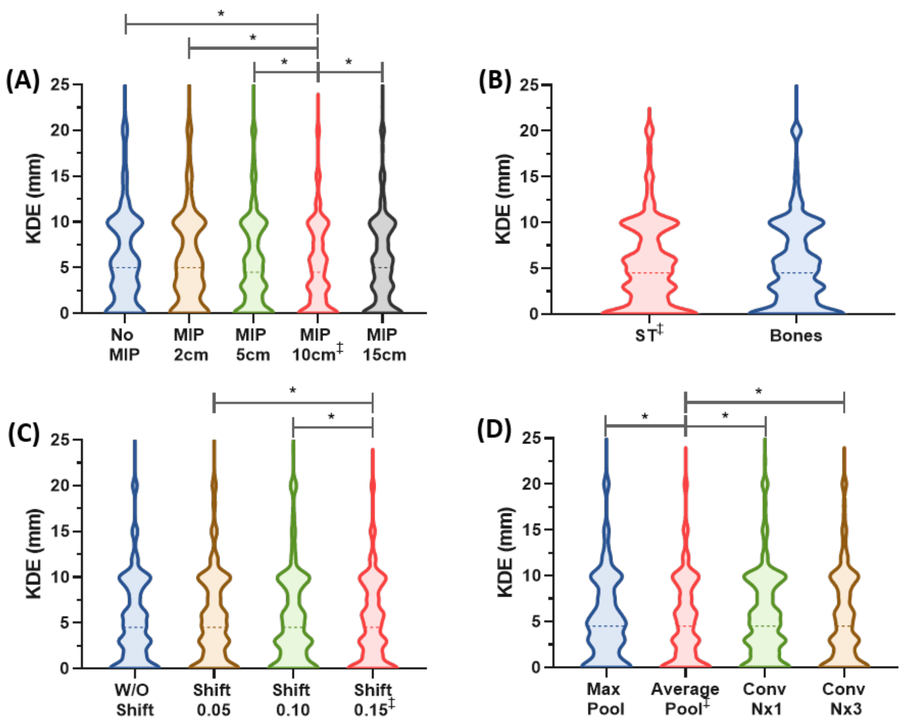

Figure 37: Influence of (A) MIP's ROI size, (B) type of windowing function for image normalization, (C) amount of translation augmentation, and (D) type of dimensionality reduction block, on the performance of the sagital block (assessed in a 5-fold cross validation).

* p<0.05, in a Wilcoxon matched-pairs signed rank test against the proposed parameter's value (marked with ‡).

In the experiments carried out to analyze the sensitivity to different types of CT normalization, it is possible to observe that the soft tissue window has a slightly lower average error when compared to the bone window, although not statistically significant (figure 37-B). This experiment differs from what was observed in the axial block analysis (figure 33-B), which suggests that the soft tissues have a lower influence on the model's ability to locate the sternum's greatest depression point.

Since one may expect a large variability in patient's relative positioning in the sagittal image (as a result of the patient's positioning in the CT bed, but also dependent on the acquired region including the thorax only or part of the abdomen also), an experiment was performed to analyze the error when considering different amounts of translation augmentation (figure 37-C). As can be seen from figure 37-C, there is only a statistically significant difference between the different types of augmentation, with no statistically differences when no augmentation is applied. However, it is possible to see that the variability decreases when a translation of 0.15 is applied to the image. Overall, no consistent result was observed, with significant differences being found for different amounts of augmentation, but no statistical significance when stopping to augment the images with random translations. This result contradicts the expected influence of data augmentation on the network's performance. An explanation for this result is that the network is solving a relatively easy problem, managing to reach

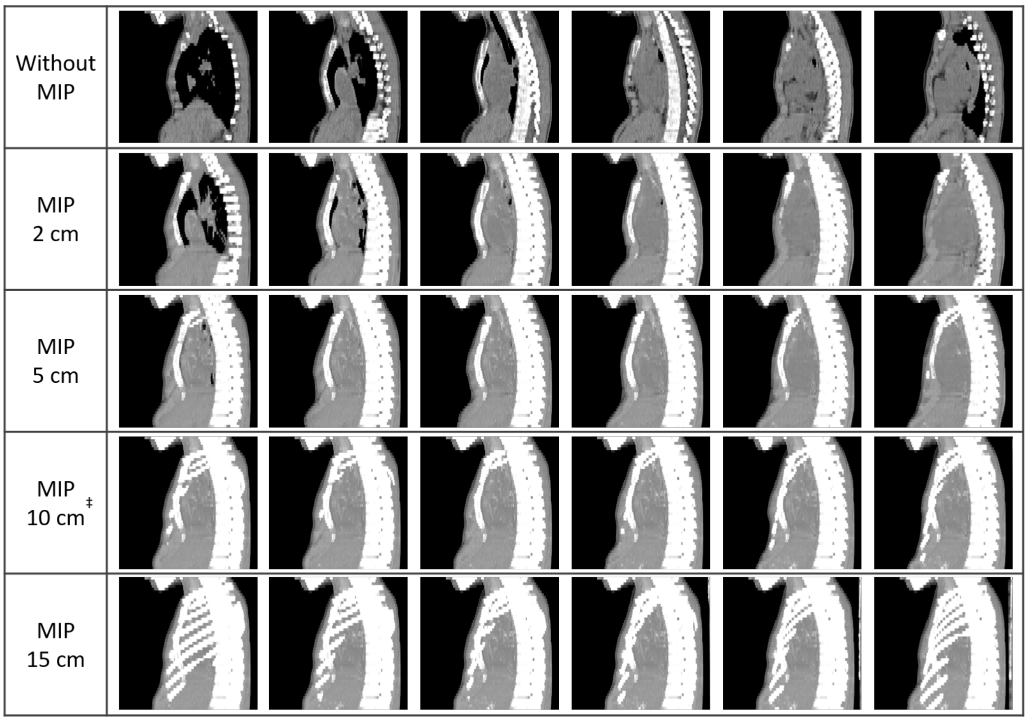

Figure 38: Influence of the MIP's ROI size on the sagittal slice extracted for keypoint detection. Taking a single CT as example, each row represents a different ROI size, and each column assumes a different center for the ROI in the CT volume (to simulate off-centered acquisitions). ‡ = proposed ROI size.

an almost optimal result even with data that has a small variation between them. Another explanation may be the fact that the variability seen in terms of patient's positioning in the image is similar between the training and validation sets (irrespectively of being either small or large across the used CTs).

With respect to the method used to reduce the dimensionality of the output from 2D to 1D in the 2D-1D conversion block, there is a significant statistical difference between the proposed approach and all tested variants (figure 37-D), with the former showing lower average errors.

As in the axial block, an analysis was carried out with the test set to unbiasedly evaluate the proposed method's performance (figure 39). Once again, the plot includes the results for all trained models (from the cross-validation), the one with the best average error on the test set, and the ensemble approaches presented previously. Contrarily to what was observed in the axial block (figure 34-A), there was no significant difference between approaches. The best fold showed a median KDE slightly lower than the ensembles but with much larger error dispersion (more outliers), especially when compared to the keypoint-based ensembles. Similarly to the axial block, no superiority was observed for ensemble strategies that take into account the network's score.

Figure 40 presents some detection results for samples included in the test set, when compared to the expert annotations.

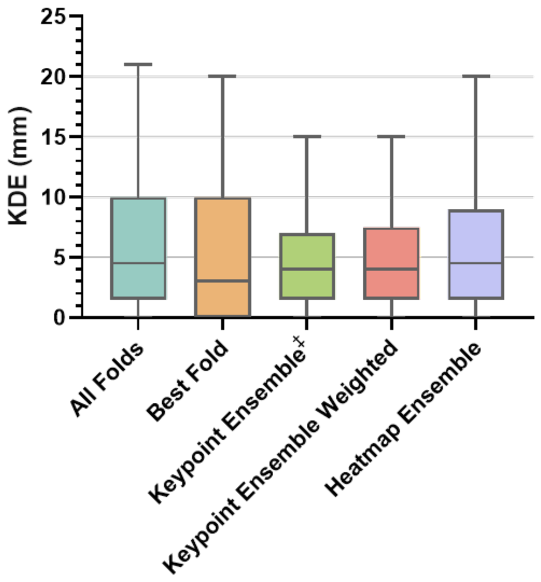

Figure 39: Sagittal block results obtained with the test set, for all folds, for the best fold, and for various ensemble approaches.

No statistical difference between approaches was observed in a Wilcoxon matched-pairs signed rank test ($p<0.05$).

4.4 Clinical Validation

This section intends to validate the pipeline of the proposed framework (named PAMnet) as a whole, including the post-processing block and the selection of the correct axial slice for measurements' extraction.

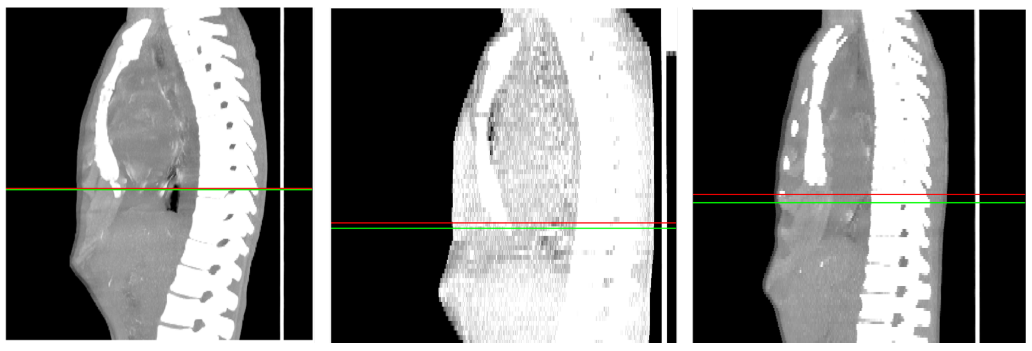

Figure 40: Sagittal keypoints' detection examples for three representative samples from the test set. The red line represents the proposed model's output, and the green line the ground truth.

4.4.1 *Experiment Description*

All CTs from the test set's patients (52 volumes from 18 patients) were used to carry out this experiment, which corresponds to ≃20% of all available patients.

The expert annotated all CTs to generate the ground truth to analyze the framework's performance. Upon analyzing the sagittal images and identifying the keypoint corresponding to the sternum's greatest depression, the corresponding axial slice was generated. In this axial slice, the expert annotated the 8 relevant keypoints. The identified keypoints were then regularized to guarantee the orthogonality of the extracted measurements (see example in figure 41) and, finally, the measurements were extracted (similarly to the framework's pipeline process). This procedure was repeated two times by the same expert in order to obtain data for an intra-observer analysis. A second observer repeated the annotation process, thus providing data for an inter-observer analysis.

The pipeline as a whole was executed to obtain the framework's results (figure 24). In the first block of the pipeline (sagittal block), the sagittal slice was extracted and analyzed using the keypoint ensemble method. In the second block (axial block), it was created a 2 cm ROI centered on the keypoint located in the previous block. The axial slices contained in that ROI were extracted, and the 8 keypoints present in each axial slice were located using the keypoint ensemble method. The last block (post-processing block) corrected possible errors present in the detection of the 8 keypoints, and searched in the ROI for the correct axial slice based on the APD (the shortest one), guaranteeing the orthogonality of the extracted measurements and calculating the PE indices.

Importantly, the axial keypoint detection network has presented no case of keypoint inversion (swap of a left/right or anterior/posterior keypoint pair) or other substantial positioning error, and thus the keypoint correction algorithm was not triggered during the post-processing stage.

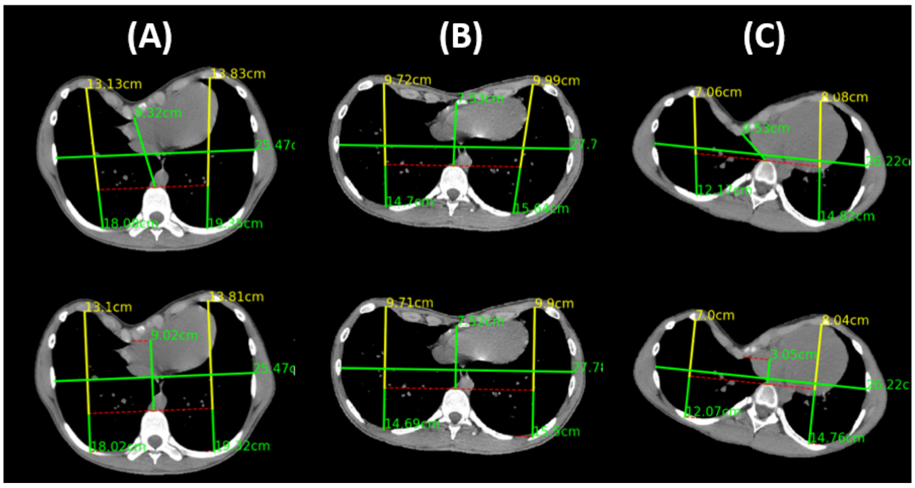

Figure 41: Examples of the transformation applied to the keypoints to guarantee the orthogonality of the extracted measurements.
Top row: unconstrained measurements; bottom row: orthogonal measurements.

4.4.2 *Results and Discussion*

Table 5 summarizes the distances (in cm) estimated by both the proposed framework and the expert for the necessary measures to calculate the PE indices. The table also shows the linear regression (correlation coefficient, slope and two-tailed paired *t*-test between measurements) and Bland-Altman (bias and limits of agreement (LOA)) analyses between the PAMnet's estimated values and the expert ones. Table 6 replicates the same analysis for the PE indices.

According to tables 5 and 6, it is possible to verify that all automatically extracted measurements and indices have a remarkable correlation against the expert's values, with only statistically significant differences on the biases for the VCD, RAPD, and LAPD measures. Notwithstanding, PAMnet consistently presented a slightly narrower range of extracted PE indices when compared to the expert.

To further understand the performance of the proposed framework, the agreement between the automatically extracted indices and the expert ones was compared against the inter- and intra-observer variabilities (figure 42 and tables 6 and 7). With respect to the Haller index, a small bias was found (figure 42-A), with a LOA slightly wider than the intra-observer agreement (figure 42-D and table 7) but not significantly different in a two-tailed F-test to the inter-observer one (42-G and table 7). The wider LOA compared to the intra-observer analysis seem to be linked to a few outlier cases with a remarkably high Haller index. Indeed, as previously mentioned and as observed in table 5, PAMnet seems to present a slightly narrower range of computed values, with a tendency to underestimate in cases with a severe deformity (see outliers in figure 42-A). A close inspection of these sub-optimal

Table 5: Linear regression (correlation coefficient, slope and two-tailed paired *t*-test between measures) and Bland-Altman (bias and limits of agreement) analyses between PAMnet and expert's extracted measurements

	TCD	APD	VCD	RAPD	LAPD
PAMnet[1] (proposed)	25.5 ± 2.09 [20.9; 29.9]	6.53 ± 2.06 [2.96; 10.77]	10.5 ± 1.93 [7.8; 13.9]	15.2 ± 1.80 [12.0; 18.2]	15.8 ± 1.99 [12.6; 19.3]
Expert [1]	25.5 ± 2.08 [20.7; 29.7]	6.61 ± 2.23 [2.60; 11.05]	10.6 ± 1.96 [7.8; 14.2]	15.1 ± 1.93 [10.8; 18.3]	15.7 ± 2.02 [12.3; 19.5]
Correlation coefficient	0.996	0.992	0.927	0.993	0.996
Slope	1.001	0.916	0.905	0.927	0.980
t-test	0.237	0.086	0.000*	0.011*	0.046*
Bias [2]	0.032	0.077	-0.075	0.092	0.051
LOA [2]	[-0.35; 0.42]	[-0.70; 0.54]	[-0.43; 0.28]	[-0.40; 0.58]	[-0.30; 0.40]

[1] Values are mean ± standard deviation [range], presented in cm.
[2] Values are presented in cm.
* $p < 0.05$, two-tailed paired *t*-test against PAMnet/Expert.

Table 6: Linear regression (correlation coefficient, slope and two-tailed paired *t*-test between measurements) and Bland-Altman (bias and limits of agreement) analyses between PAMnet and expert's extracted indices

	Haller index	Correction index	Asymmetry index
PAMnet (proposed) [1]	4.32 ± 1.49 [2.47; 8.83]	38.7 ± 11.3 [21.4; 63.2]	96.8 ± 6.14 [81.5; 107.2]
Expert [1]	4.35 ± 1.73 [2.29; 10.15]	36.2 ±- 12.7 [16.7; 62.3]	96.4 ±- 6.6 [76.0; 106.5]
Correlation coefficient	0.988	0.970	0.963
Slope	0.849	0.863	0.896
t-test	0.431	0.883	0.196
Bias [2]	-0.039	-0.066	0.324
LOA [2]	[-0.73; 0.65]	[-6.26; 6.40]	[-3.17; 3.82]

[1] Values are mean ± standard deviation [range], presented in cm.
[2] Values are presented in cm.

cases allowed to identify that this error is caused by the non-optimal prediction of the patient's rotation on the CT bed (the example in figure 43-A allows to observe that the TCD vector should be slightly more oblique with respect to the image axes, to accurately capture the patient's orientation) and explained by the Haller index equation (1) itself. Since the APD distance is in the denominator and the TCD distance is in the numerator, and being the latter much higher than the former (average TCD = 25.5 cm vs. APD = 6.61 cm), this means that in extreme cases where the APD distance is minimal, a few mm in error results in a large difference in the calculated Haller index.

Regarding the correction index, the proposed framework presents a LOA slightly narrower than the inter-observer variability (figure 42-B and 42-E, although not significant in a two-tailed *F*-test; table 7), and slightly wider than the intra-observer one (figure 42-H, although not significant also; table 7). These results suggest that the framework does not overfit to the expert that provided the annotations used for training, while showing a LOA similar to those obtained by a second observer (blinded to the 1st expert's results).

The agreement with the measurements of the experts' asymmetry index revealed a statistically similar performance of the proposed framework when compared to the intra- and inter-observer variability (table 7). The framework's LOA present a value slightly lower than the inter-observer and a value slightly higher than the intra-observer, presenting a similar number of outliers (see figures 42-C, 42-F, and 42-I). In terms of bias, the framework showed a higher bias with respect to the expert than the ones reported in the intra- and inter-observer analyses. Nonetheless, the reported value (+0.32, on average) is irrelevant given the asymmetry index's average value (96.76 in the present dataset).

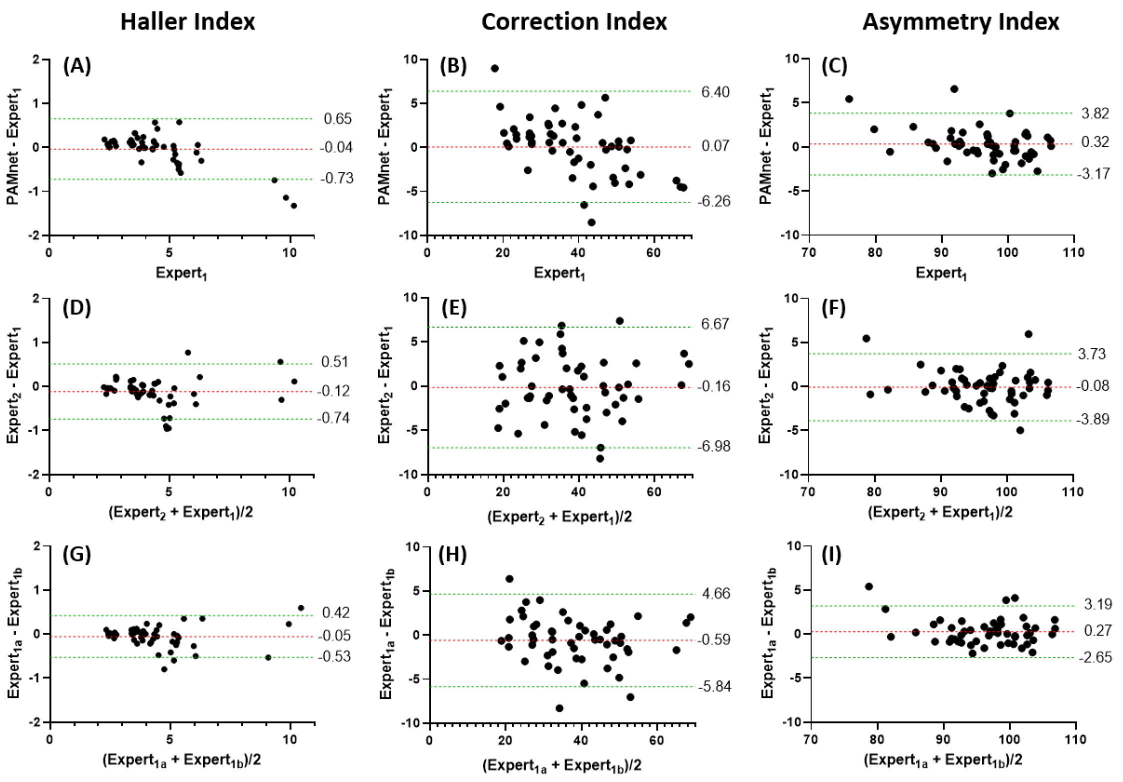

Figure 42: Bland–Altman analysis of PAMnet vs. expert (A–C), inter- (A–C) and intra-observer variability (E–H), for the Haller (A, D and G), correction (B, E and H) and asymmetry (C, F, and I) indices. Both bias (mean difference) and 95% LOAs (+1.96 SD around the mean difference) are depicted.

Additionally, an intra-patient analysis was performed to assess the robustness (or sensitivity) of the framework with respect to the CT used as input, and compare it to the variability obtained by the expert. Since multiple CTs are usually acquired for each patient during diagnosis and preoperative planning, one can compute the PE indices from each CT and verify the variability across acquisitions. As a goal, these variations are expected to be very small since the information of the patient's ribcage is the same across all CTs. The intra-patient analysis consisted in calculating, for

Table 7: Two-tailed *F*-test results for the multiple analyses performed

	Haller index	Correction index	Asymmetry index
Inter-observer vs PAMnet-Expert	0.514	0.594	0.546
Intra-observer vs PAMnet-Expert	0.010	0.185	0.206
Intra-patient expert vs Intra-patient PAMnet	<0.0001	<0.0001	<0.0001

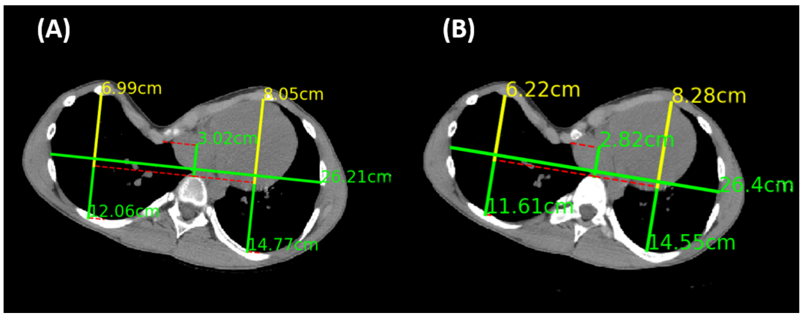

Figure 43: Comparison between (A) PAMnet and (B) expert measurements for a patient with an extremely high Haller index.

each patient, the error between the indices computed for each CT and the average value across that patient's CTs. Importantly, the developed framework demonstrated a significantly lower variability in all indices compared to the one obtained in the expert's intra-patient analysis (figure 44 and table 7). This experiment demonstrates that the proposed framework is more reproducible than the expert, extracting similar values irrespective of the CT available to be analyzed.

Finally, the automatically extracted values (PAMnet) were compared against those extracted by the expert in the clinics. The approach followed in the latter consisted on using a DICOM viewer to freely navigate the CT volume and, after selecting the axial slice with the sternum's greatest depression point, draw several lines to extract the necessary distances to calculate the indices (without guarantees of the orthogonality of the measures made). The analysis results are shown in figure 45. Overall, all indices presented a great correlation with the manually-extracted values (between 0.94 and 0.96), with a low bias and a close to unitary slope (only slightly lower for the asymmetry index, figure 45-C).

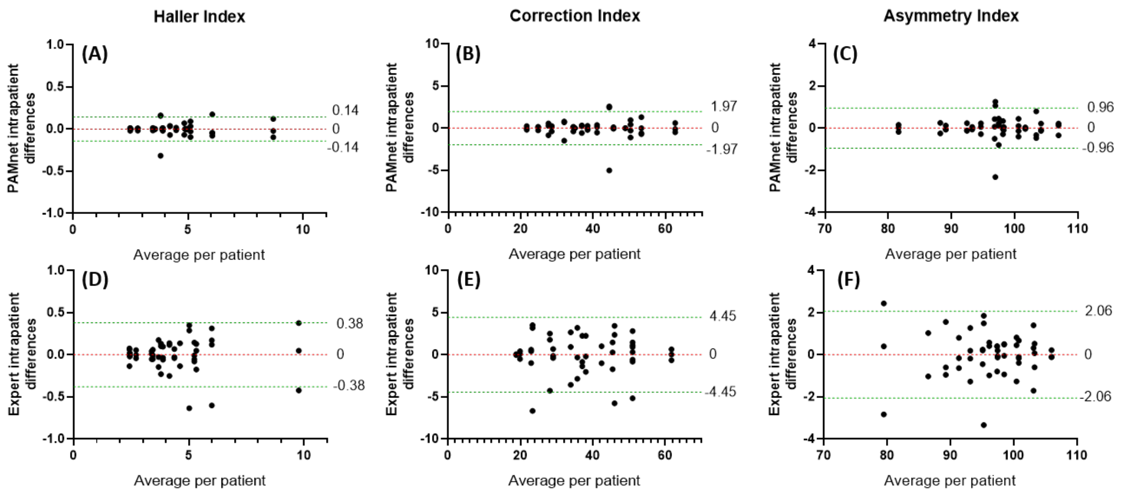

Figure 44: Bland-Altman analysis for PAMnet's (A-C) and expert's (D-F) intra-patient variability for the Haller (A and D), correction (B and E) and asymmetry (C and F) indices.
Both bias (mean difference) and 95% LOAs (+1.96 SD around the mean difference) are depicted.

The average relative error was 4.09%, 5.59% and 1.91% for the Haller, correction and asymmetry indices, respectively.

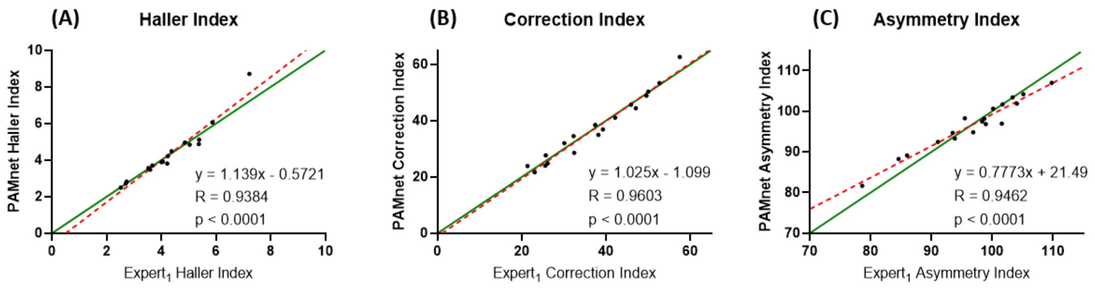

Figure 45: Linear regression analysis between PAMnet's automatically extracted PE indices and those manually extracted by the expert in a clinical environment.
Analysis for the Haller index (A); correction index (B); and asymmetry index (C). The green line represents the identity line, and the red dashed line represents PAMnet's linear regression line.

5

CONCLUSION

In this thesis, a framework was proposed to calculate, through deep learning techniques, three standard indices used by experts to quantify the severity of a patient's PE: Haller index, correction index and asymmetry index. Using 2 CNNs and some geometric rules, a framework was developed to detect keypoints in the CT volume and calculate the PE indices, automating the current tedious and variability-prone manual procedure.

The framework's performance was evaluated by comparing the automatically calculated indices against the experts' indices. The experimental results revealed greater accuracy for networks based on the UNet++ architecture, showing a good agreement with the manually calculated indices, presenting low biases and narrow LOAs and being comparable to the inter-observer variability. The intra-patient analysis revealed a wider variability between indices manually extracted from distinct CTs of the same patient, a characteristic that the developed framework does not present, offering very high reproducibility. In conclusion, the developed framework presents itself as a viable solution for automatic quantification of the PE severity in a clinical setting, achieving an average relative error of 4.09%, 5.59% and 1.91% for the Haller, correction and asymmetry indices, respectively.

Despite the contributions presented in this work, future developments are still envisioned. On the one hand, one intends to integrate the developed framework into a software package for easy use in clinical practice. On the other hand, one intends to perform a more extensive clinical validation with data from other hospitals.

BIBLIOGRAPHY

[1] A. A. Fokin, N. M. Steuerwald, W. A. Ahrens, and K. E. Allen, "Anatomical, histologic, and genetic characteristics of congenital chest wall deformities," in *Seminars in thoracic and cardiovascular surgery*, vol. 21, pp. 44–57, Elsevier, 2009.

[2] M. W. Lam, A. F. Klassen, C. J. Montgomery, J. G. LeBlanc, and E. D. Skarsgard, "Quality-of-life outcomes after surgical correction of pectus excavatum: a comparison of the ravitch and nuss procedures," *Journal of pediatric surgery*, vol. 43, no. 5, pp. 819–825, 2008.

[3] E. Coln, J. Carrasco, and D. Coln, "Demonstrating relief of cardiac compression with the nuss minimally invasive repair for pectus excavatum," *Journal of pediatric surgery*, vol. 41, no. 4, pp. 683–686, 2006.

[4] D. Crossland, A. Auldist, and A. Davis, "Malignant pectus excavatum," *Heart*, vol. 92, no. 10, pp. 1511–1511, 2006.

[5] A. C. Koumbourlis and C. J. Stolar, "Lung growth and function in children and adolescents with idiopathic pectus excavatum," *Pediatric pulmonology*, vol. 38, no. 4, pp. 339–343, 2004.

[6] H. J. Bigelow, "Insensibility during surgical operations produced by inhalation," *The Boston medical and surgical journal*, vol. 35, no. 16, pp. 309–317, 1846.

[7] D. Nuss, R. E. Kelly Jr, D. P. Croitoru, and M. E. Katz, "A 10-year review of a minimally invasive technique for the correction of pectus excavatum," *Journal of pediatric surgery*, vol. 33, no. 4, pp. 545–552, 1998.

[8] D. Nuss, R. J. Obermeyer, and R. E. Kelly, "Nuss bar procedure: past, present and future," *Annals of cardiothoracic surgery*, vol. 5, no. 5, p. 422, 2016.

[9] S. W. Daunt, J. H. Cohen, and S. F. Miller, "Age-related normal ranges for the haller index in children," *Pediatric radiology*, vol. 34, no. 4, pp. 326–330, 2004.

[10] J. A. Haller, S. S. Kramer, and S. A. Lietman, "Use of ct scans in selection of patients for pectusexcavatum surgery: a preliminary report," *Journal of pediatric surgery*, vol. 22, no. 10, pp. 904–906, 1987.

[11] D. Nuss, "Minimally invasive surgical repair of pectus excavatum," in *Seminars in pediatric surgery*, vol. 17, pp. 209–217, Elsevier, 2008.

[12] R. E. Kelly, M. J. Goretsky, R. Obermeyer, M. A. Kuhn, R. Redlinger, T. S. Haney, A. Moskowitz, and D. Nuss, "Twenty-one years of experience with minimally invasive repair of pectus excavatum by the nuss procedure in 1215 patients," *Annals of surgery*, vol. 252, no. 6, pp. 1072–1081, 2010.

[13] P. M. Poston, S. S. Patel, M. Rajput, N. O. Rossi, M. S. Ghanamah, J. E. Davis, and J. W. Turek, "The correction index: setting the standard for recommending operative repair of pectus excavatum," *The Annals of thoracic surgery*, vol. 97, no. 4, pp. 1176–1180, 2014.

[14] S. D. S. Peter, D. Juang, C. L. Garey, C. A. Laituri, D. J. Ostlie, R. J. Sharp, and C. L. Snyder, "A novel measure for pectus excavatum: the correction index," *Journal of pediatric surgery*, vol. 46, no. 12, pp. 2270–2273, 2011.

[15] J. A. Sujka and S. D. S. Peter, "Quantification of pectus excavatum: anatomic indices," in *Seminars in pediatric surgery*, vol. 27, pp. 122–126, Elsevier, 2018.

[16] R. E. Kelly, T. F. Cash, R. C. Shamberger, K. K. Mitchell, R. B. Mellins, M. L. Lawson, K. Oldham, R. G. Azizkhan, A. V. Hebra, D. Nuss, *et al.*, "Surgical repair of pectus excavatum markedly improves body image and perceived ability for physical activity: multicenter study," *Pediatrics*, vol. 122, no. 6, pp. 1218–1222, 2008.

[17] M. L. Lawson, M. Barnes-Eley, B. L. Burke, K. Mitchell, M. E. Katz, C. L. Dory, S. F. Miller, D. Nuss, D. P. Croitoru, M. J. Goretsky, *et al.*, "Reliability of a standardized protocol to calculate cross-sectional chest area and severity indices to evaluate pectus excavatum," *Journal of pediatric surgery*, vol. 41, no. 7, pp. 1219–1225, 2006.

[18] S. B. Sesia, M. Heitzelmann, S. Schaedelin, O. Magerkurth, G. J. Kocher, R. A. Schmid, and F.-M. Haecker, "Standardized haller and asymmetry index combined for a more accurate assessment of pectus excavatum," *The Annals of thoracic surgery*, vol. 107, no. 1, pp. 271–276, 2019.

[19] D. Crevier, *AI: the tumultuous history of the search for artificial intelligence*. Basic Books, Inc., 1993.

[20] S. Russell and P. Norvig, "Artificial intelligence: a modern approach," 2002.

[21] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain.," *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[22] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[23] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[24] F. Rosenblatt, *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

[25] F. Rosenblatt, "Principles of neurodynamics. perceptrons and the theory of brain mechanisms," tech. rep., Cornell Aeronautical Lab Inc Buffalo NY, 1961.

[26] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.

[27] S. Ruder, "An overview of gradient descent optimization algorithms," pp. 1–14, 2016.

[28] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization.," *Journal of machine learning research*, vol. 12, no. 7, 2011.

[29] M. D. Zeiler, "Adadelta: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[31] K. Fukushima and S. Miyake, "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition," in *Competition and cooperation in neural nets*, pp. 267–285, Springer, 1982.

[32] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[33] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," *arXiv preprint arXiv:1603.07285*, 2016.

[34] Y.-T. Zhou and R. Chellappa, "Computation of optical flow using a neural network.," in *ICNN*, pp. 71–78, 1988.

[35] M. Yani *et al.*, "Application of transfer learning using convolutional neural network method for early detection of terry's nail," in *Journal of Physics: Conference Series*, vol. 1201, p. 012052, IOP Publishing, 2019.

[36] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[37] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[38] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.

[39] L. SIfre and S. Mallat, "Rigid-motion scattering for texture classification," *arXiv preprint arXiv:1403.1687*, 2014.

[40] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[41] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

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

[43] Z. Zhou, M. M. R. Siddiquee, N. Tajbakhsh, and J. Liang, "Unet++: A nested u-net architecture for medical image segmentation," in *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, pp. 3–11, Springer, 2018.

[44] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

[45] A. Nibali, Z. He, S. Morgan, and L. Prendergast, "Numerical coordinate regression with convolutional neural networks," *arXiv preprint arXiv:1801.07372*, 2018.

[46] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, "Openpose: realtime multi-person 2d pose estimation using part affinity fields," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 1, pp. 172–186, 2019.

[47] Y. Raaj, H. Idrees, G. Hidalgo, and Y. Sheikh, "Efficient online multi-person 2d pose tracking with recurrent spatio-temporal affinity fields," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4620–4628, 2019.

[48] X. Wang, X. Yang, H. Dou, S. Li, P.-A. Heng, and D. Ni, "Joint segmentation and landmark localization of fetal femur in ultrasound volumes," in *2019 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI)*, pp. 1–5, IEEE, 2019.

[49] S. Belharbi, C. Chatelain, R. Hérault, S. Adam, S. Thureau, M. Chastan, and R. Modzelewski, "Spotting l3 slice in ct scans using deep convolutional network and transfer learning," *Computers in biology and medicine*, vol. 87, pp. 95–103, 2017.

[50] F. Kanavati, S. Islam, E. O. Aboagye, and A. Rockall, "Automatic l3 slice detection in 3d ct images using fully-convolutional networks," *arXiv preprint arXiv:1811.09244*, 2018.

[51] F. Galbusera, F. Niemeyer, H.-J. Wilke, T. Bassani, G. Casaroli, C. Anania, F. Costa, M. Brayda-Bruno, and L. M. Sconfienza, "Fully automated radiological analysis of spinal disorders and deformities: a deep learning approach," *European Spine Journal*, vol. 28, no. 5, pp. 951–960, 2019.

[52] B. S. Andreassen, F. Veronesi, O. Gerard, A. H. S. Solberg, and E. Samset, "Mitral annulus segmentation using deep learning in 3-d transesophageal echocardiography," *IEEE Journal of Biomedical and Health Informatics*, vol. 24, no. 4, pp. 994–1003, 2019.

[53] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

[54] I. Kim, "tf-pose-estimation." https://github.com/ildoonet/tf-pose-estimation/, 2019.

APPENDIXES

Appendix A

Figures 46 and 47 present the alternative architectures tested for the axial keypoint detection network. For each network, the amount of l2-regularization was tuned to reduce the overfit to the training set. Thus, a weight of 0, 0, $5{\times}10^{-5}$, $5{\times}10^{-5}$ and $1{\times}10^{-4}$ was used for VGG19, VGG16, MobileNetV1, MobileNetV2 and UNet, respectively.

The architectures in figure 46 were based on the architectures used by the *OpenPose* framework (and *tf-pose* implementation [54], specifically) [46]. Four different backbones were used: VGG19, VGG16, MobileNetV1, and MobileNetV2. Each architecture has a head to extract features coming from the backbones (dashed in red). The heads were based on the CMU heads used in openpose [46], which are composed of 4 convolution blocks (in the case of MobileNets, the depthwise separable convolution or residual bottleneck blocks are used), in which the first 3 blocks contain a total of 128 filters and the last 512 filters. In the architecture with the MobileNetV1 backbone, features from 3 different levels are concatenated (with the respective up or downsample to match the second level's size, i.e. 4 times lower than the input image) and are used as feature maps for the network's head. Contrarily, in the MobileNetV2, the feature maps used in the head come from 4 distinct levels of the backbone, all resampled to match the second level's size (also 4 times lower than the input image size).

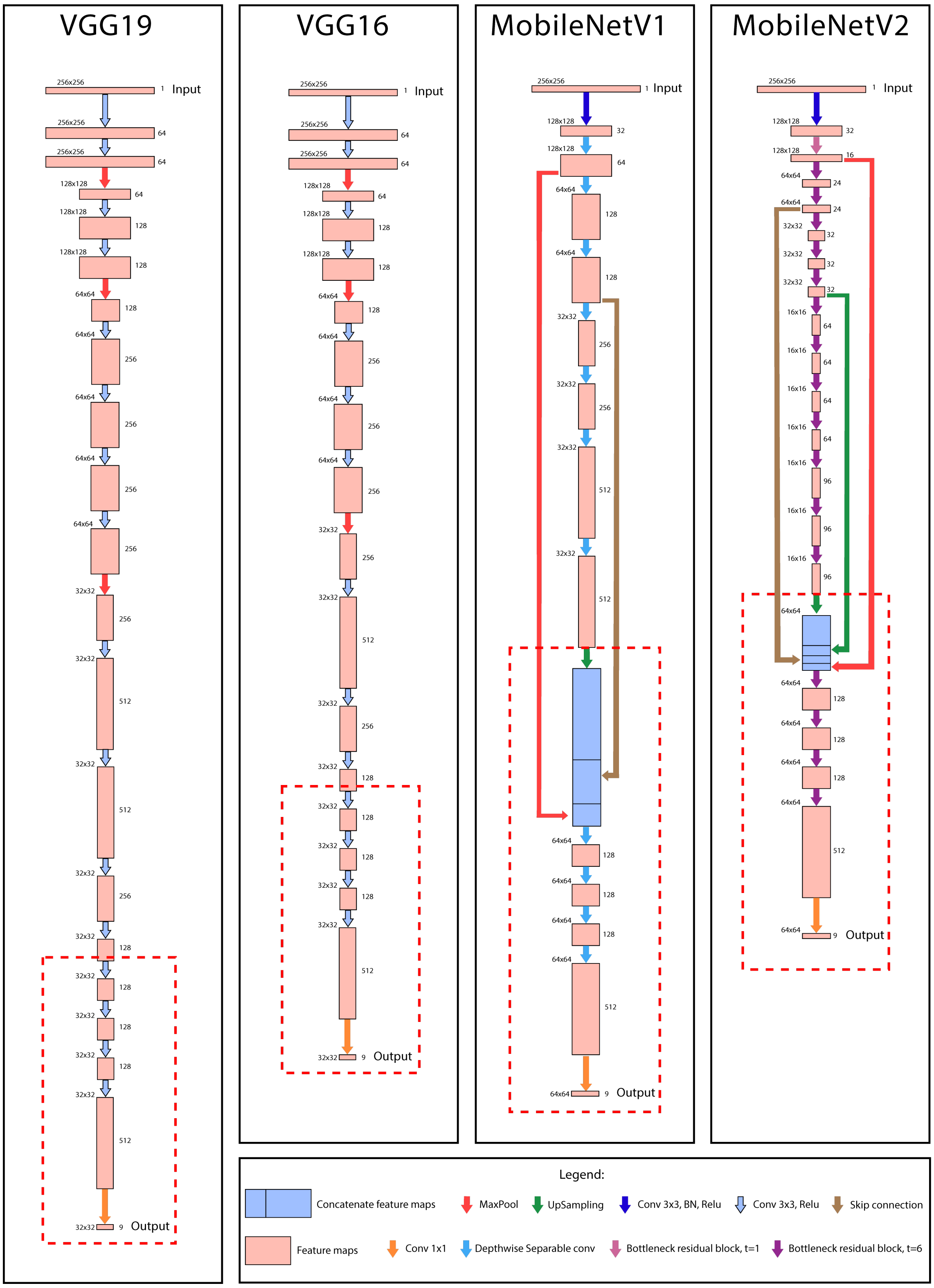

Figure 46: Illustrations of the alternative architectures tested for the axial keypoint detection network. Four different architectures were used (based on [46][54]), contemplating a backbone (VGG19, VGG16, MobileNetV1 and MobileNetV2) to extract a set of feature maps and a head (dashed red rectangle) to regress the keypoints' heatmaps.

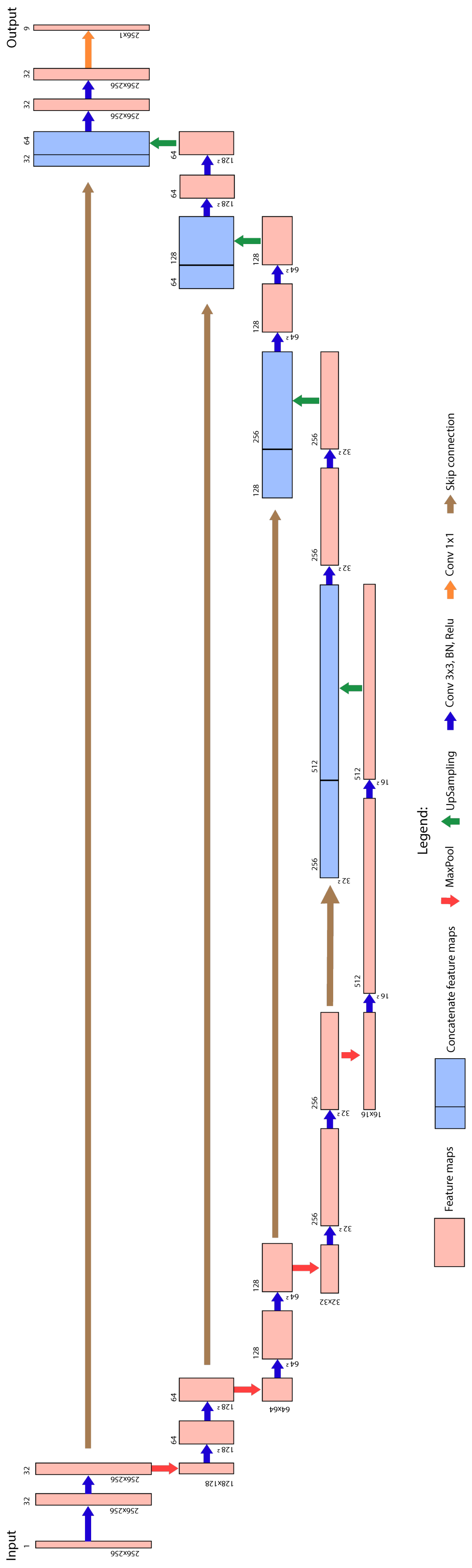

Figure 47: Axial keypoint detection network based on the UNet architecture.

Appendix B

In order to verify if the trends observed in the parameter study of the proposed axial block are maintained (figure 33), the same analysis was repeated for the two alternative architectures with the best result during the 5-fold cross-validation (table 4), i.e. MobileNetV2 and UNet (see illustrations in Appendix A). The results are summarized in figures 48 and 49.

Since the distributions resulting from the results are non-normal, the non-parametric Wilcoxon matched-pairs signed rank test was performed between the proposed parameters' values and their variations, with a significance level of 0.05.

Analyzing figure 48-A, the MobileNetV2's architecture presents a statistically significant difference for all input sizes, having the size 256×256 the smallest error. As in the proposed method's axial block the increased error in the larger image size may indicate that the kernel size is too small, or the number of scales is too small. The upscale operation probably causes the larger error seen with the smallest image size (192×192), since the output image size is only 48×48 pixels. In the UNet architecture, the same was not verified, with the larger image size showing the lowest error (figure 49-A).

Contrary to what was observed in the analysis of the proposed architecture (figure 33-B), the two additionally tested architectures shown a lower error when the CT is normalized using the bone window (figures 48-B and 49-B).

In terms of sigma size, the 7.5 mm size continued to be the one with the lower average error in both architectures (figures 48-C and 49-C). Contrarily to the proposed Unet++ architecture (figure 33-C, that only shown significant differences to the larger sigma value), both architectures had a statistically significant lower accuracy when trained with smaller or larger sigma sizes.

When analyzing the type of regression output, it is possible to observe that the DSNT method presents a significantly larger average error. This error is more pronounced in the MobileNetV2's architecture (48-D), showing the DSNT method's weakness. Since the output size is small (64×64 pixels), the network's resolution is lower and the residual errors during the interpolation/upscaling operation will be larger. The same does not happen with a regression based on heatmap matching, since the predicted heatmap is interpolated first (with a cubic method) prior to computing the keypoints' coordinates (with an overall lower residual interpolation error).

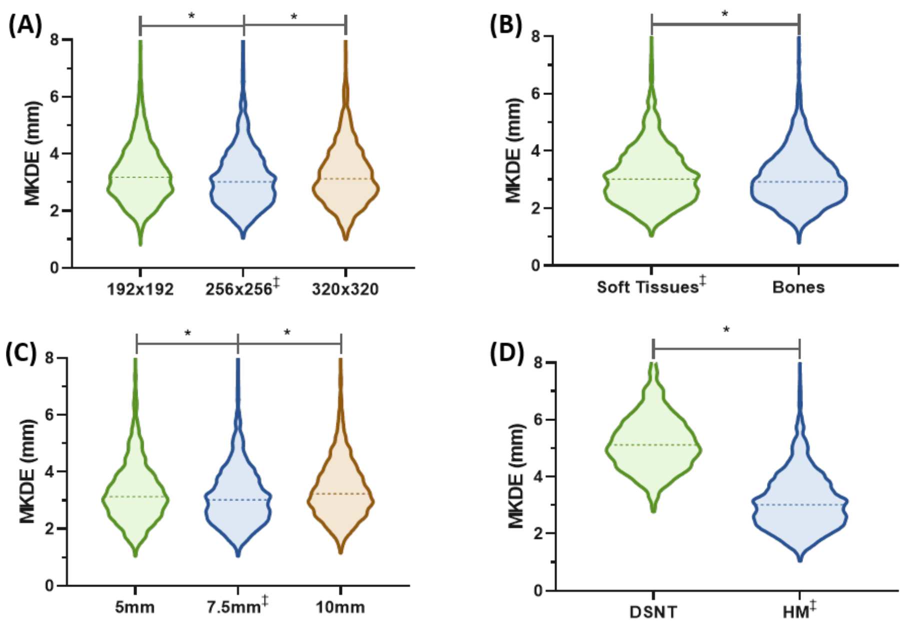

Figure 48: Influence of (A) input size, (B) type of windowing function for image normalization, (C) heatmaps' sigma size, and (D) type of output, on the performance of the axial block for the MobileNetV2 architecture (assessed in a 5-fold cross validation).

* p<0.05, in a Wilcoxon matched-pairs signed rank test against the proposed parameter's value (marked with ‡)

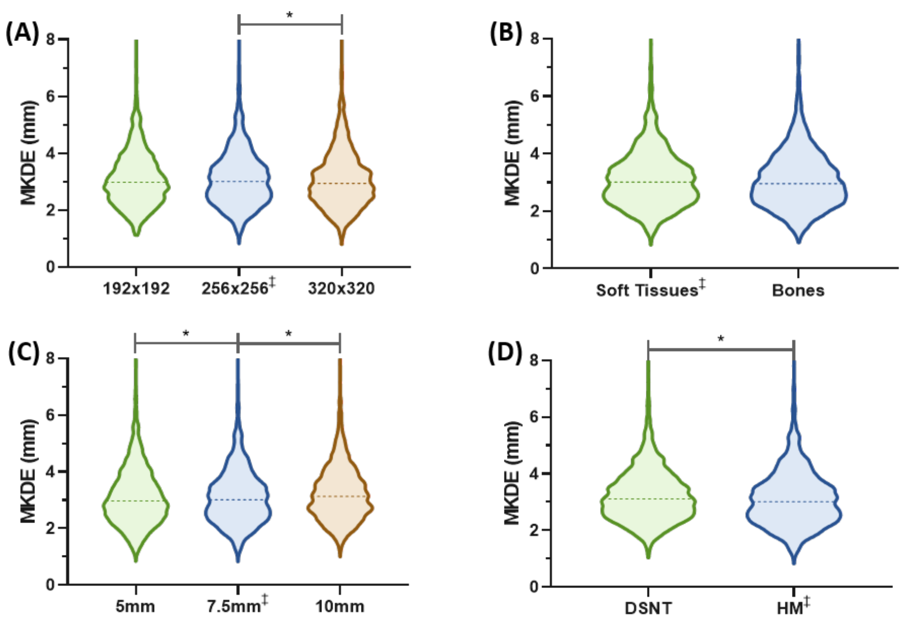

Figure 49: Influence of (A) input size, (B) type of windowing function for image normalization, (C) heatmaps' sigma size, and (D) type of output, on the performance of the axial block for the UNet architecture (assessed in a 5-fold cross validation).

* p<0.05, in a Wilcoxon matched-pairs signed rank test against the proposed parameter's value (marked with ‡)

Appendix C

Table 8 presents some additional experiments performed while tuning the proposed network for the proposed method's axial block (based on the Unet++ architecture).

Table 8: Additional parameter study for the axial block

Experiment settings	MKDE
Proposed method [1]	2.99 ±1.18
Output without background heatmap [1]	3.12 ±1.84
Kernels multiple of 24 [1]	3.19 ±2.01
Separable convolution block with kernels multiple of 24 [1]	3.04 ±1.21
Proposed method [2]	3.01 ±0.97
Without augmentation [2]	3.19 ±1.06
Small augmentation [2] [3]	2.99 ±0.98
Large augmentation [2] [4]	3.09 ±1.18

[1] Results of the 5 (all) folds.

[2] Results of only 2 folds.

[3] Data augmentation consisting of rotational transformations in the range of [-2.5°; +2.5°], translations between -2.5 and +2.5% of the image size, and a scaling factor between 0.975 and 1.025.

[4] Data augmentation consisting of rotational transformations in the range of [-10°; +10°], translations between -10 and +10% of the image size, and a scaling factor between 0.9 and 1.1.