POLITECNICO DI TORINO

Master of Science in Biomedical Engineering

Master Degree Thesis

Artificial intelligence applications in implant dentistry

Supervisor:
Prof. Massimo Salvi

Co-supervisor::
Prof. Filippo Molinari

Candidate:
Anastasio Romano

Primo Lab S.r.l.
Dr. Riccardo Mala

Academic Year 2021-2022

Abstract

In recent years, the evolution of modern implant dentistry and imaging techniques have been combined to create a new dental implantology protocol: **guided implant surgery**. With this innovative technique, a cone-beam computed tomography (CBCT) scan of the patient is taken to create 3D images of the oral structures. Afterwards, a specific software is used to design a surgical template, to establish the optimal placement (slope and depth) of dental implants. Although guided surgery ensures shorter surgical time and reduces risk of injuring important anatomical structures (i.e., nerves, bones, adjacent teeth, etc.), this technique is not commonly used as it requires longer preoperative planning time, normally performed by clinicians.

The main aims of this study conducted at the *PrimoLab S.r.l.* company are to reduce the preoperative planning time by identifying the most critical planning steps from a time-consuming point of view, and to validate an alternative protocol of guided implant surgery, in which surgical plannings are carried out by biomedical engineers and templates are 3D-printed once clinician checks the virtual designs.

Achieving these goals required operating at different levels.

First of all, a time analysis has been performed with the Guided Surgery Team of *PrimoLab*, considering 200 cases of surgery; the analysis revealed that data manipulation's step is crucial since it represents around 70% of the total preoperative planning time. Furthermore, around 85% of the total cases were categorized as belonging to mandibular planning; this can be justified considering that mandible is subjected to higher mechanical forces and consequently has a higher rate of healing than maxilla.

Second, deep learning with different 3D convolutional neural networks (CNNs) have been used for multiclass segmentation of mandible, teeth, and background in

CBCT scans. To train the models, 500 volumes were obtained from patients who had undergone orthodontic treatment in *Centri Dentistici Primo* company and segmentations were manually created with 3D Slicer. The segmentation performance of all trained CNNs was assessed by the DSC (Dice similarity coefficient) and RVD (Relative Volume Difference). Fully automated segmentation demonstrated a large overlap with manual segmentations (DSC: 0.9059 $\pm$ 0.0273 and 0.8939 $\pm$ 0.0325, respectively for mandible and teeth; RVD: -0.0074 $\pm$ 0.0514 and -0.0125$\pm$ 0.0601, respectively for mandible and teeth).

In the end, automatic segmentations of mandible and teeth have been saved in stl (Standard Triangulation Language) format and an experimental evaluation has been performed: 40 surgical templates were designed (20 patients, for each both manual and automatic segmentation). During the experiment, engineers did not know if the segmentation was done manually or automatically. Subsequently, surgical guides were compared and the deviation between the two designs (manual and automatic, for each patient) was measured using the EXOCAD software, reaching a maximum deviation of (348 $\pm$ 212.78) μm.

With the 3D CNN implemented, multiclass segmentation takes about 9 seconds for 1 CBCT scan, while manual segmentation takes about 40 minutes. This study demonstrates that multiclass mandible and teeth segmentation with deep learning is accurate and generally usable to reduce preoperative planning time in guided implant surgery.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Digital-guided implant surgery has become a widely used technique in modern dentistry, providing a more accurate and efficient way to place dental implants. One of the critical steps in this process is the segmentation of CBCT (Cone Beam computed tomography) data, which is typically done manually by expert clinicians. However, this process is time-consuming and can lead to errors, with negative consequences for patients health.

The primary focus of this thesis conducted at the *PrimoLab S.r.l.* company is to reduce the preoperative planning time by identifying the most critical planning steps from a time-consuming point of view, and to validate an alternative protocol of guided implant surgery, in which surgical plannings are carried out by biomedical engineers and templates are 3D-printed once clinician checks the virtual designs.

To achieve the above goal, a deep learning algorithm is applied. In particular, a convolutional neural network (CNN) is implemented to accurately identify mandible and teeth starting from raw CBCT volumes. To train the network a large dataset is used; it consists of 500 volumes in DICOM (Digital Imaging and Communications in Medicine) format, acquired by different dental clinics belonging to *Centri Dentistici Primo* company.
Performance of the segmentation model are evaluated by comparing it to the manual segmentation process performed by biomedical engineers of the *PrimoLab Guided*

Surgery Team.

In order to provide a comprehensive overview of the work conducted, this thesis is structured as follows.

The second chapter contains an introduction of artificial intelligence (AI), machine learning and deep learning, with a brief description of the most used basic networks.

The third chapter includes a description of the state of the art of dental technologies, specifically focusing on guided implant surgery. In detail, it presents an overview of the digital workflow for virtual surgical planning and 3D printing technologies. Moreover, it reports a real clinical case conducted by *Primo Lab* and *Centro Dentistici Primo of Carpi*, with *Dr. Cristian Gheorghiu*.

The fourth chapter illustrates the work done for segmenting the original dataset and for training the CNN network. Furthermore, metrics evaluated and hyperparameters are also presented.

The fifth chapter contains results and an experimental evaluation in which the deviation is measured between surgical guides designed starting from a segmentation done automatically and manually.

In conclusion, it is proved that multiclass mandible and teeth segmentation with network implemented and using deep learning algorithm is accurate and generally capable of automating a tedious and time-consuming activity within the company.

Chapter 2

Deep Learning

In this chapter a brief presentation of Artificial Intelligence and Deep Learning is provided.
Deep learning is a subset of machine learning composed of algorithms that permit software to train itself to perform task like speech and image recognition, by exposing multilayered neural networks to vast amounts of data.
Deep learning has dramatically improved the state-of-the-art in many different artificial intelligent tasks like object detection, speech recognition, machine translation [1].

2.1 Artificial Intelligence

The word "artificial intelligence" was born in the second half of the 20th century when Turing, in order to make a well-founded judgement about the intelligence of a machine, presented what is known today as the Turing Test (TT). The setup of the test is composed by a human interrogator in a room where she can send text messages over a digital device to two communicators, one of which is a machine. By asking questions and receiving answers, it is the interrogator's task to distinguish which of the two communicators is a human and which one is a machine. It is the objective of the machine to mislead the interrogator into making a wrong assessment by imitating human response behaviour. Since multiple interrogators in repeated trial-experiments failed to make the right assessment, the machine passed

the TT. According to Turing, for this experiment, can be said that machine's ability has intelligent behaviour equivalent to, or indistinguishable from, that of a human [2].
Consequently, in 1956, Marvin Minsky [3] coined the word artificial intelligence: he theorized that any human mental process can be described as a state machine, therefore as a calculator. In other words, he described the mental process as a succession of decisions and learning mechanisms.
Then, demonstrations such as Newell and Simon's General Problem Solver and Joseph Weizenbaum's ELIZA showed promise toward the goals of problem solving and the interpretation of spoken language respectively. These successes, as well as the advocacy of leading researchers (namely the attendees of the DSRPAI) convinced government agencies such as the Defense Advanced Research Projects Agency (DARPA) to fund AI research at several institutions. The government was particularly interested in a machine that could transcribe and translate spoken language as well as high throughput data processing. [4].
Since 1950 artificial intelligence has been increasingly used in various fields, up to the present day in which artificial intelligence accompanies us almost at every step. It is used not only for the development of autonomous vehicles and robots, but also to support medics in identifying disease entities and disturbing changes in X-rays, as well as solving many problems faced by business every day.

Nowadays, when we talk about artificial intelligence we often refer to words such as **machine learning (ML)** and **deep learning (DL)**:

- *artificial intelligence*: includes ML and DL; it is defined as any technique that allows computers to simulate human intelligence, using special logic, if-then rules, clustering, decision trees, machine Learning and Deep Learning;

- *machine learning*: a subset of AI that includes abstruse statistical techniques that enables machines to improve at tasks with experience. The category includes deep learning. ML uses statistical methods in such a way as to allow the machine to learn, i.e. to improve the resolution of certain tasks with experience (e.g. chess-playing robots or artificial neural networks that are trained and return a certain number of features from 1 to n classes);

- *deep learning*: a subset of ML which using multi-layer neural networks and a large amount of data, allows machines to learn and therefore train themselves to improve the resolution of a given task (e.g. Face ID recognition).

The main difference between machine learning and deep learning is the *feature extraction*, i.e. the process of transforming raw data into numerical features that can be processed while preserving the information in the original data set. The traditional machine learning approach, in fact, is based on handcrafted features (process called *Deep thinking - Shallow learning*); in this case, neural networks use mathematical features set by user (data mean, variance, standard deviation, concavity, convexity...): output solution complexity and performance are strongly correlated with initial choices (which features to extract from raw data). The deep learning approach is instead based on hierarchical representation learning (process called *Shallow thinking - Deep learning*), where representations from the input data are learnt by putting emphasis on buiding complicated mapping using a series of simple mappings (figure 2.1).

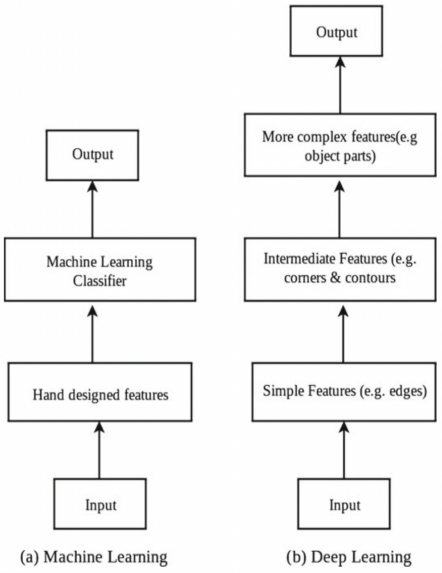


Figure 2.1: (a) Conventional machine learning using hand-designed feature extraction algorithms; (b) deep learning approach using hierarchy of representations that are learnt automatically [5].

Moreover, deep learning involves using large amounts of data to learn progressively. Its performance are higher than machine learning but if and only if a large amount of data is available 2.2. This can be a disadvantage of deep learning because, in some applications, large amounts of data are rarely available. Thus, more flexible models are required to achieve an enhanced learning ability when only a limited amount of data is available.

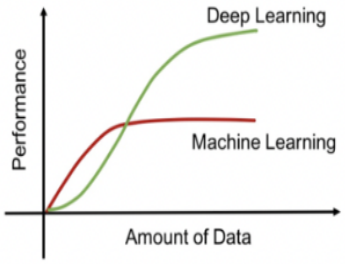


Figure 2.2: ML and DL performance in function of amount of data

2.1.1 The Artificial Neuron

Artificial neural networks try to mimic the way the human brain learns. The brain is primarily made up of a collection of neurons, all interconnected and firing electric signals back and forth to help the mind interpret things, reason and make decisions; each neuron takes information through dendrites, processes it in its cell body and returns it outside through axons. Neurons are connected to each other by synapses.

With reference to figure 2.3, the mathematical replication of the neuron in artificial neural networks is:

$$a = \sigma(z) \tag{2.1}$$

$$z = b + \sum_j w_j x_j \tag{2.2}$$

where x_j are the inputs, w_j are the weights, b is the bias term and $\sigma(\cdot)$ is the activation function.

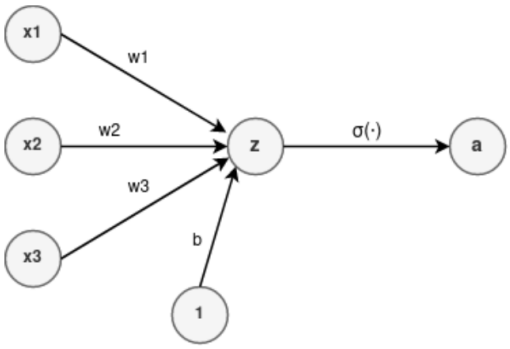


Figure 2.3: Representation of artificial neuron

The activation function is a non-linear, usually differentiable function. It's important to underline that an artificial neuron without an activation function corresponds to the prediction function of a linear regression model. The difference is that a regression model is a stochastic model, while an artificial neuron is a deterministic model. The first activation function used was the sigmoid function (formula 2.3), this because it was thought to be the most biologically plausible activation function; today the ReLU function (formula 2.4) is the default choice [6].

$$sigmoid(x) = \frac{1}{1 + \exp(-x)} \tag{2.3}$$

$$ReLU(x) = max(0, x) \tag{2.4}$$

This mathematical model is simply an approximation of human brain behavior: a neural network, once trained, will always return the same output for the same input, which does not happen for our brain (in two different moments the brain answers differently to an identical input).

2.1.2 Shallow Networks

Starting from the neuron model descripted above, the simplest neural network is built by stacking together artificial neurons in layers; It is called Shallow Net.

With reference to figure 2.4, its mathematical replication is:

$$y = b + \sum_j w_j z_j \tag{2.5}$$

$$z_j = \sigma(b_j + \sum_i w_{ij} x_i) \tag{2.6}$$

A shallow network is so composed by:

- *input layer*: it is characterized by neurons x_j. Each neuron has a single dendrite and a certain number of axons. Input layer's neurons return input adding the bias (b_j);
- *hidden layer*: it is characterized by neurons z_j. They process the information, obtaining the linear combination of the inputs
- *output layer*: it is characterized by neuron y (or more neurons). It (or they) has a certain number of dendrites and a single axon (output). Output layer's neurons return in output the weighted information;

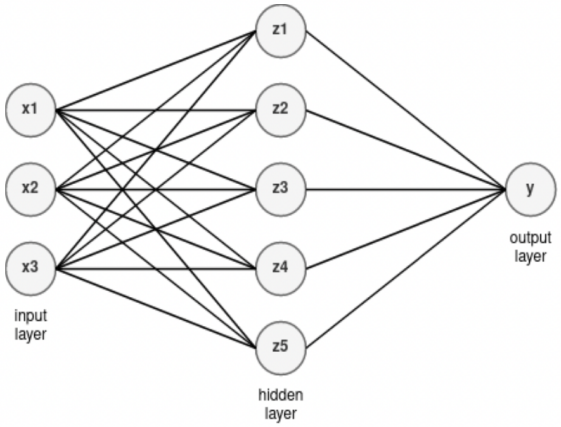


Figure 2.4: Representation of shallow network. For simplicity bias neurons, activations and edge orientation are implied

For example, in case of cytological images classification (healthy or diseased), each input can be a feature extracted from data (area, mean, standard deviation, etc...) and only one output is required (e.g. the output can be 0 for healthy cell and 1 for diseased cell).
For training, basically the net is initialized with random weights. Knowing the real class of the train set it tries, at each step (for each iteration), to minimize the error between predicted and real class by modifying the neurons weights present in the

various layers. The iterative process continues until the error stabilizes or is close to 0.

2.1.3 Deep Networks

A deep network (figure 2.5) is a network that has a certain number of input layers and hidden layers; among hidden layers each neuron is connected with all previous and next layer's neurons. It is a very dense network, full of connections. Some theoretical studies [7] find that is more computationally efficient to represent a given function with a neural network that is deep instead of shallow. This can be intuitive thinking the way humans organize knowledge, as a hierarchy with many levels.

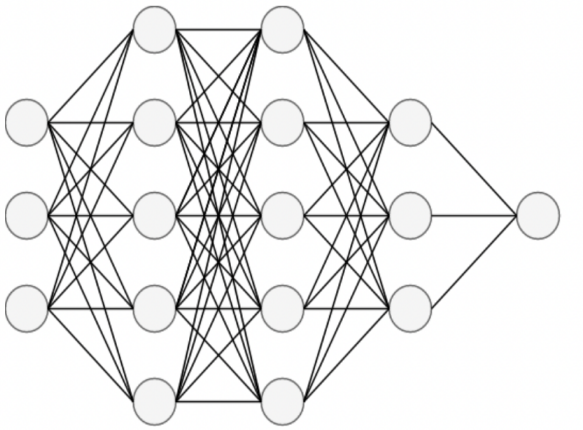


Figure 2.5: Representation of a deep network

2.1.4 Learning

The learning task is achieved through a learning algorithm that acts modifying the weights of the connections between neurons.
Two main types of learning are possible with artificial neural networks (ANNs):

- *Supervised Learning*: the ANN is provided with a training set consisting of input vectors and the corresponding targets (desired outputs). The aim of supervised learning is to adjust the connection weights such than the error between the network output and the target output is minimized;

- *Unsupervised Learning*: the ANN is provided with a training set made only of input vectors (the corresponding targets are unknown). The aim of unsupervised learning is to cluster the elements in the training set in homogeneous groups.

There are many different learning algorithms according to the learning type.

Supervised Learning process and Backpropagation

The basic approach in learning is to start with an untrained network, present a training pattern to the input layer, pass the signals through the net and determine the output at the output layer. Then, the obtained output is compared to the target value. The error is used to adjust the weights in order to reduce it. In every case the training data are presented several time to the network. A single presentation of all pattern is called *epoch*.

One of the most popular methods for training is based on gradient descent and it is called *Backpropagation* or generalized delta rule.
The goal of this algorithm is to achieve the optimal set of weights, moving back from the network's prediction to the neurons that generated that prediction.
From a simple point of view, backpropagation is a method for calculating the first derivative of the *Loss Function (LF)* with respect to each network weight to find weights that bring LF to a minimum: once the forward process is performed, i.e. once the input data is processed in the forward direction through the network, an initial prediction is generated. At this point there is an error function (E), the Loss Function, which defines how far the result is from the actual value.

More in detail, backpropagation algorithm can be described in the following steps:

- Feed-forward computation
- Backpropagation to the output layer
- Backpropagation to the hidden layers
- Parameters updates

It is an iterative algorithm and is stopped when the value of the error function has become sufficiently small. The updaiting parameters process can be explained with formulas 2.7 and 2.8, where $w_{i,j}^h$ is the weight of the connection between neuron j in layer $(h-1)$ and neuron i in layer h. b_i^h is the bias of neuron i on layer h. α is known as *learning rate* and it determines how rapidly the parameters are updated. larger values ideally require fewer iterations to reach convergence (i.e. the optimal combination of network weights to minimize the loss function). However, if the learning rate is too big the optimal values will be overshot. α is also identified as the hyperparameter of the network.

$$w_{i,j}^h = w_{i,j}^h - \alpha * \frac{\delta E}{\delta w_{i,j}^h} \tag{2.7}$$

$$b_i^h = b_i^h - \alpha * \frac{\delta E}{\delta b_i^h} \tag{2.8}$$

The chain is applied considering $\frac{\delta \sigma}{\delta z}$ the derivative of the activation function (formula 2.1):

$$\frac{\delta E}{\delta z} = \frac{\delta E}{\delta \sigma} * \frac{\delta \sigma}{\delta z} \tag{2.9}$$

$$\frac{\delta E}{\delta w} = \frac{\delta E}{\delta z} * \frac{\delta z}{\delta w} \tag{2.10}$$

Consequently, it can be write:

$$\frac{\delta E}{\delta w_{i,j}^h} = \delta \sigma_j^{(h-1)} * \frac{\delta E}{\delta z} \tag{2.11}$$

It can be done samely for b.

In the basic version of gradient descent, parameters are updated after the whole training set has passed over the forward and backward propagation. This is usually not practical when working with big sets of data. For this reason another important hyperparameter to set before training is the *batch size*, which is the number of samples used in every epoch to train the network: a batch of samples is run in one big forward pass, and then backpropagation is performed on the aggregate result. Setting this hyperparameter too high can make the network take too long to achieve convergence (no more gain in accuracy); however, if it is too low, it will make the network bounce back and forth without achieving acceptable performance. Also, the nature of the dataset can have an impact on the batch size, especially the medical dataset because of its complexity [8].

Overfitting

An ANN is in overfitting when it is good at learning the training set but do not has the ability to generalize that gained knowledge to additional, unseen examples. Overfitting is a fundamental issue in supervised learning; presence of noise, limited size of training set, and complexity of tasks can be considered the main causes.
To reduce the effects of overfitting, Xue Ying et al. [9] reported various strategies:

- Early stopping strategy: during training, the model is evaluated on a holdout validation dataset after each epoch. If the performance of the model on the validation dataset starts to degrade (e.g. loss begins to increase or accuracy begins to decrease), then the training process is stopped;
- Network reduction strategy: it is used to exclude the noises in training set;
- Data expansion strategy: it is proposed for complicated models to fine tune the hyperparameters sets with a great amount of data
- Regularization strategy: it is proposed to guarantee models performance to a great extent while dealing with real world issues by feature selection, and by distinguishing more useful and less useful features

2.2 Convolutional Neural Networks

The most widespread algorithm among various deep learning models is convolutional neural network (CNN), a class of artificial neural networks that has been a dominant method in computer vision tasks since the astonishing results were shared on the object recognition competition known as the ImageNet Large Scale Visual Recognition Competition (ILSVRC) in 2012 [10].
CNNs represents the most established models in the image processing field; they are used for both segmentation and classification problems.
The main concept of these networks, which differentiates them from ML networks (where the features extraction is set by user) is that the features extraction is done internally to the network through *convolution operations*.
Generally, convolution is a mathematical operation on two functions $f(t)$ and $k(t)$

which can be defined as:

$$(f * k)(t) = \int_{-\infty}^{+\infty} f(\tau)k(t-\tau)d\tau \tag{2.12}$$

Since data are discrete quantities, the integral is substituted with a finite sum. Moreover, the convolution presented above is limited to one dimensional data, like temporal signals but it can be extended for 2D and 3D data, like images and CBCTs, with formulas 2.13 and 2.14.

$$(I * K)(x, y) = \sum_{\Delta x} \sum_{\Delta y} K(\Delta x, \Delta y) I(x - \Delta x, y - \Delta y) \tag{2.13}$$

$$(I * K)(x, y, z) = \sum_{\Delta x} \sum_{\Delta y} \sum_{\Delta z} K(\Delta x, \Delta y, \Delta z) I(x - \Delta x, y - \Delta y, z - \Delta z) \tag{2.14}$$

In formula 2.12 the kernel $k(\tau)$ is reflected $k(-\tau)$ and translated $k(t-\tau)$. The reflection makes convolution symmetric, but this does not happen with images (both 2D or 3D like CBCTs). In that context, in fact, the role of the input image I and kernel K is not equivalent: the kernel is smaller than images (for 2D convolution, it tipically has shape 3x3, 5x5, 7x7) and through convolution it is used to extract features from input data.

2.2.1 Overall architecture

CNNs are comprised of three types of layers. These are convolutional layers, pooling layers and fully-connected layers. When these layers are stacked, a CNN architecture has been formed.

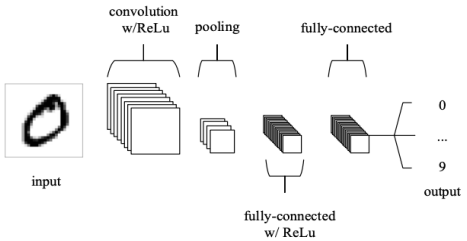


Figure 2.6: Simply CNN architecture for classification, [11]

The Convolution Layer

The goal of the convolutional layer is simply to extract features and the layers parameters focus around the use of learnable kernels.
These kernels are usually small in spatial dimensionality, but spreads along the entirety of the depth of the input. The kernel is glided all over the input image: the center element of the kernel is placed over the input vector, of which is then calculated and replaced with a weighted sum of itself and any nearby pixels (figure 2.7).

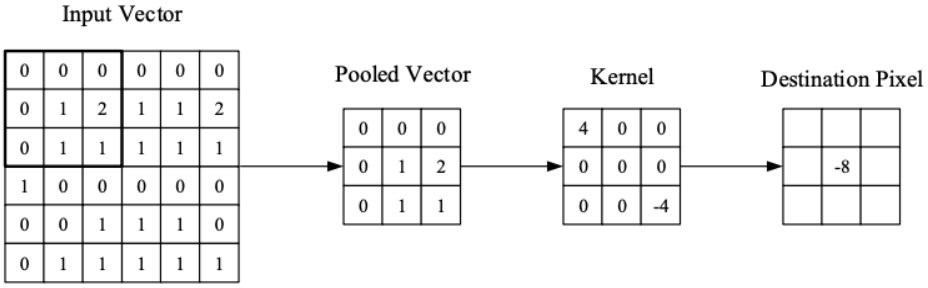


Figure 2.7: Representation of a convolution layer, [11]

This is the way how convolutional layer aggregates features maps and outputs new ones which contain higher-level features. Many convolution layers in sequence extract features at multiple levels: at lower levels they detect lines and edges, then they detect patterns and simple shapes and finally at higher levels they detect objects.
Since kernel plays a vital role in how CNNs operate, we need to define:

- **Kernel size**: as mentioned previously, it is smaller than the input and, for 2D tasks, it has tipically shape 3x3, 5x5 or 7x7;
- **Stride**: number of pixels by which the kernel is shifted when it is convolved for the image (a stride of 1 means that the kernel is shifted by one pixel between one operation and the next);
- **Zero padding**: is the simple process of padding the border of the input with zeros. It is an effective method to give further control as to the dimensionality

of the output;

- **Depth of the convolutional layer**, i.e. the number of filters (hence the number of kernels) applied. It determines the dimension along z of the output matrix; each layer of the output matrix is the result of the i-th kernel applied to input.

To introduce non-linearity to the system, the convolutional layer is characterized by an **activation function** (e.g. RELU, which keeps the positive values of the matrix and sets the negative values to zero).

The Pooling Layer

The aim of the pooling layer is to make feature selection. It reduces the size of each matrix obtained by keeping the most important information according to the type of pooling chosen (max pooling, average pooling or sum pooling). The feature selection takes place, also in this case, through a kernel. Therefore, the parameters to be defined are:

- **Kernel size**;
- **Stride (e.g. equal to 2)**;
- **Type of pooling**:
  - *Max pooling*: keeps the maximum value among the pixels selected by the kernel
  - *Average pooling*: keeps the average of the pixels selected by the kernel
  - *Sum pooling*: keeps the sum of pixels selected by the kernel
  - *L1/L2 normalization*: keeps the L1/L2 norm of pixels selected by the kernel

The features obtained from this level are high-level features, i.e. features for which it is not possible to give a mathematical definition, since they are obtained from a convolution of many kernels in both the convolutional layer and the pooling layer. In most CNNs, these come in the form of max-pooling layers with kernels of a

dimensionality of 2x2 applied with a stride of 2 along the spatial dimensions of the input [11].
The presence of these high-level features distinguishes deep learning from machine learning and this is ad advantage since for many applications it is impossible to establish best mathematical features to extract.

The Fully-connected Layer

The fully connected layer contains neurons of which are directly connected to the neurons in the two adjacent layers, without being connected to any layers within them. The term fully-connected refers to the fact that each neuron of the previous layer is connected to each neuron of the following layer, therefore the fully connected is nothing more than a classic ANN presented in the previous section.
In a network for a classification task, the fully connected layer has the purpose of vectorizing the output matrix from the pooling layer, i.e. it vectorizes the high-level features and provides, through the appropriate weights, a certain number of outputs according to the classes in which you want to classify the input.

2.3 CNN for classification

Image Classification was historically the first vision task addressed by Con- volutional Networks. Given a dataset of labeled images, the algorithm has to classify correctly the given image, i.e its predicted label has to match the true label.
The CNN architecture for classification includes convolutional layers, max-pooling layers, and fully connected layers. Convolution and max-pooling layers are used for feature extraction. While convolution layers are meant for feature detection, max-pooling layers are meant for feature selection. Max-pooling layers are employed when there are instances when the picture doesn't require all of the high-resolution details or an output with smaller regions extracted by CNN's is needed after performing downsampling operation on input data. The output from convolution and pooling layers is fed into the fully connected layers for classification. The examples of classification learning task where CNN is used are image classification, object detection, and facial recognition. The diagram in figure 2.8 represents the basic

CNN architecture for image classification.

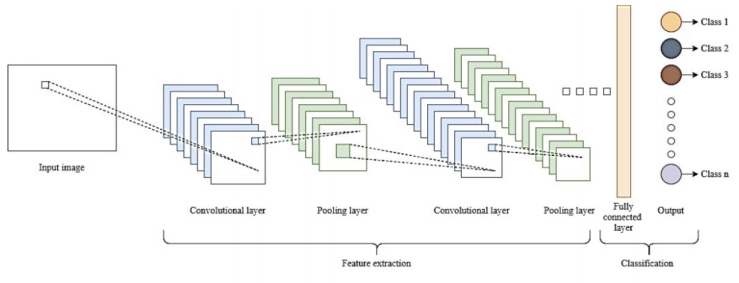


Figure 2.8: CNN for classification tasks [12]

2.3.1 Brief History

LeNet

LeNet, also known as LeNet-5, is a convolutional neural network architecture proposed by Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner in 1998 [13]. It is considered as one of the pioneering works in the field of deep learning and computer vision.

LeNet was originally designed for the recognition of handwritten digits and was later applied to other computer vision tasks, such as face recognition and traffic sign classification. The architecture of LeNet consists of seven layers, including two convolutional layers, two subsampling layers, and three fully connected layers.

The first two layers of LeNet are convolutional layers that extract features from the input image using a set of learnable filters. The output of these layers is then passed through subsampling layers that reduce the dimensionality of the feature maps by performing a downsampling operation such as max pooling.

The last three layers of LeNet are fully connected layers that classify the input image based on the features extracted by the earlier layers. The final layer produces an output vector with the same dimensionality as the number of classes in the dataset, with each element of the vector representing the probability of the input image belonging to a particular class.

LeNet was trained using the backpropagation algorithm and the stochastic gradient descent optimization method. It achieved a state-of-the-art performance on the

MNIST dataset of handwritten digits, which has since become a benchmark dataset in the field of computer vision.
LeNet's success paved the way for the development of many other convolutional neural network architectures, including AlexNet, VGGNet, and ResNet. Today, convolutional neural networks have become the standard approach for many computer vision tasks, such as image classification, object detection, and image segmentation.

AlexNet

AlexNet is a convolutional neural network architecture proposed by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton in 2012. It is considered as one of the pioneering works in the field of deep learning and computer vision.
AlexNet was designed for the ImageNet Large Scale Visual Recognition Challenge, a competition in which participants were required to classify images into 1,000 categories. The architecture of AlexNet consists of eight layers, including five convolutional layers, two fully connected layers, and a softmax layer.
The first layer of AlexNet is a convolutional layer that uses a set of 96 learnable filters to extract features from the input image. The output of this layer is then passed through a rectified linear unit (ReLU) activation function, which introduces non-linearity into the model. The subsequent layers of the network follow a similar pattern, with multiple convolutional layers followed by ReLU activation functions and max pooling layers.
The last layer of AlexNet is a softmax layer that produces an output vector with 1,000 elements, with each element representing the probability of the input image belonging to a particular class.
AlexNet was trained using a technique called dropout, which randomly drops out some of the neurons in the network during training to prevent overfitting. It also used data augmentation, which involves generating new training examples by applying transformations such as rotation, translation, and scaling to the existing training data.
AlexNet achieved a top-5 error rate of 15.3% on the ImageNet dataset, which was significantly better than the next best algorithm at the time, which had an error rate

of 26.2%. This breakthrough performance demonstrated the power of convolutional neural networks for image classification and paved the way for the development of many other deep learning architectures [14].

ResNet

ResNet, short for Residual Network, is a deep neural network architecture proposed by Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun in 2015 [15]. It is designed to address the problem of vanishing gradients that can occur in very deep neural networks.

The architecture of ResNet is based on the idea of residual learning, which involves adding **skip connections** (see figure 2.9) that allow information to bypass some of the layers in the network. Specifically, instead of trying to learn the direct mapping from input to output, ResNet learns the residual mapping by adding the input to the output of each layer.

This is achieved through the use of residual blocks, which consist of two or more convolutional layers followed by skip connections. The skip connections allow the gradient to flow directly from the output to the input, bypassing some of the layers in between.

The ResNet architecture comes in various depths, ranging from 18 layers to hundreds of layers. The deeper versions of ResNet have achieved state-of-the-art performance on many computer vision tasks, including image classification, object detection, and segmentation.

ResNet has also been adapted for other domains, such as natural language processing, with the use of residual connections between the layers of a recurrent neural network. The success of ResNet has led to the development of many other neural network architectures that incorporate skip connections, such as DenseNet and Highway Networks.

Researchers suggest that loss function landscape with skip connections is nearly convex whereas without skip connections is highly non-convex.

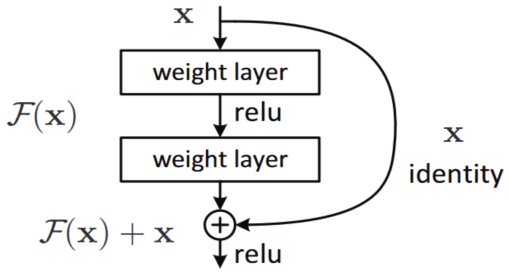


Figure 2.9: Skip connections [16]

2.4 CNN for segmentation

Computer vision deals with images, and image segmentation is one of the most important steps. It involves dividing a visual input into segments to make image analysis easier. Segments are made up of sets of one or more pixels. Image segmentation sorts pixels into larger components while also eliminating the need to consider each pixel as a unit. It is the process of dividing image into manageable sections or "tiles". The process of image segmentation starts with defining small regions on an image that should not be divided. These regions are called seeds, and the position of these seeds defines the tiles.
Image segmentation has two levels of granularity. They are as following:

- **Semantic segmentation**: Semantic segmentation classifies image pixels into one or more classes which are semantically interpretable, rather, real-world objects. Categorizing the pixel values into distinct groups using CNN is known as region proposal and annotation. Region proposals are also called candidate objects patches (COMPs), which can be thought of as small groups of pixels that are likely to belong to the same object. Semantic Segmentation is the vision task we are concerned with in this thesis.

- **Instance segmentation:** In case of instance segmentation, each instance of each object is identified. The difference between instance segmentation and semantic segmentation is that semantic segmentation doesn't categorize every pixel. If there are three objects of same type (say, bicycle) in an image, each individual bicycle is identified in instance segmentation while semantic

segmentation classifies all the bicycles as one instance.

CNN architecture for segmentation makes use of encoder and decoder models. The encoders are used to encode the input into a representation that can be sent through the network, and then decoders are used to decode the representation back. Encoders can be convolutional neural networks and the decoders can be based on the deconvolutional or transposed neural networks with the purpose of creating a segmentation map.

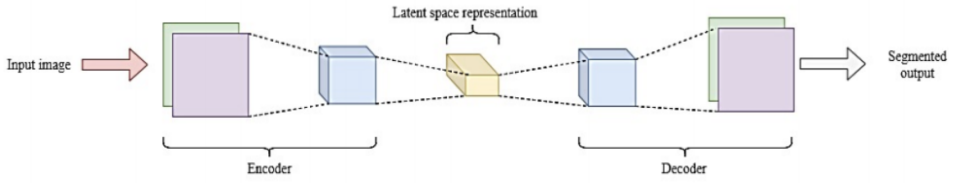


Figure 2.10: CNN for segmentation tasks [12]

2.4.1 Brief History

Fully Convolutional Networks

Fully Convolutional Networks (FCNs) are a type of deep neural network architecture that can be used for semantic segmentation. FCNs were introduced by Jonathan Long, Evan Shelhamer, and Trevor Darrell in 2015 [17] and have since become a popular approach for pixel-level prediction tasks in computer vision.
FCNs are based on the idea of replacing the fully connected layers in a traditional neural network with convolutional layers. This allows the network to take an input of any size and produce an output of the same spatial dimensionality, which is necessary for semantic segmentation tasks where the goal is to assign a label to each pixel in an image.
The architecture of an FCN typically consists of a series of convolutional and pooling layers, followed by a series of upsampling and deconvolutional layers. The

upsampling layers increase the spatial resolution of the feature maps, while the deconvolutional layers learn to reconstruct the original input from the low-resolution feature maps.
FCNs also use skip connections, which allow the network to combine high-level information from the later layers with low-level information from the earlier layers. This helps to preserve fine-grained details and improves the accuracy of the segmentation.
FCNs have been applied to a wide range of semantic segmentation tasks, including image segmentation, video segmentation, and medical image analysis. They have also been extended to handle multi-scale inputs and to incorporate contextual information from neighboring pixels.
Overall, FCNs represent a powerful and flexible approach to semantic segmentation that has led to significant improvements in many computer vision applications.

Unet

U-Net [18] is a convolutional neural network architecture designed for semantic segmentation of images. It was proposed by Olaf Ronneberger, Philipp Fischer, and Thomas Brox in 2015, and has since become a popular choice for medical image segmentation.
The U-Net architecture consists of an encoder path and a decoder path, connected by a bottleneck layer (figure 2.11). The encoder path is similar to the convolutional layers in a standard neural network, and gradually reduces the spatial dimensions of the input while increasing the number of channels. The decoder path, on the other hand, gradually upsamples the feature maps to the original size of the input.
One key feature of U-Net is the use of skip connections, which allow the network to combine high-level semantic information from the encoder path with low-level spatial information from the decoder path. This helps to preserve spatial details in the segmentation map.
U-Net also includes data augmentation techniques such as random rotations, scaling, and flipping to generate more training data and improve the robustness of the model.

U-Net has been applied to a variety of medical image segmentation tasks, including cell segmentation, brain tumor segmentation, and lung segmentation, and has shown state-of-the-art performance compared to other segmentation methods. Overall, U-Net is a powerful architecture for semantic segmentation that has shown promise in a variety of domains and continues to be an active area of research.

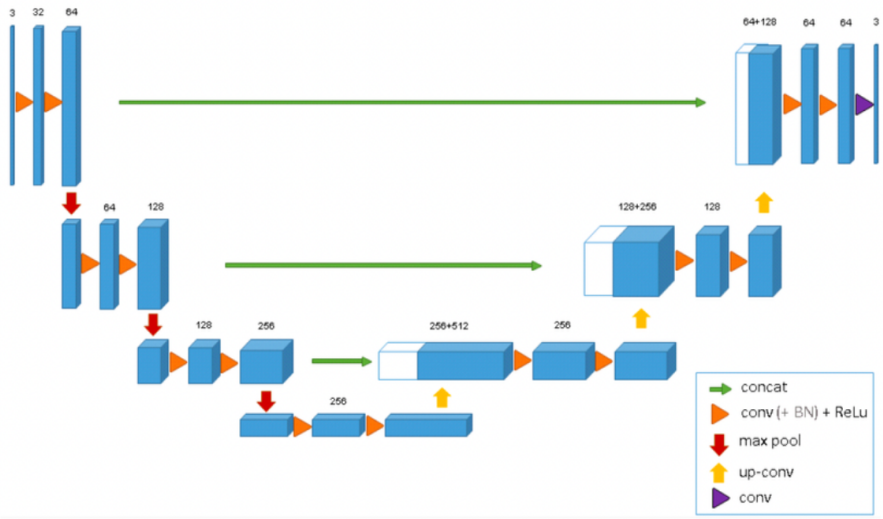


Figure 2.11: 3-D UNet architecture

Chapter 3

State of the Art of Dental Technology

There have been enormous technological advances over the last decade in dental laboratories. Digital manufacturing equipment such as 3D printers or computer-aided design/manufacturing (CAD/CAM) systems deliver a range of high-quality custom products, prosthetics, and appliances with superior fit and repeatable results, increasing clinical acceptance by dental practices and by the patient, and resulting in fewer errors and adjustments while lowering costs. Computer-aided design, computer-aided manufacturing and SLA (stereolithography) technology, in fact, have vastly spread among dental technicians worldwide and the reducing cost of processing power will ensure that this development will continue with the recent introduction of digital intra-oral scanners [19].

3.1 Digital-guided implant surgery

In recent years, the evolution of modern implant dentistry, imaging techniques and the digital revolution have combined to create a new dental implantology protocol: **computer-assisted implant dentistry (CAI)** or **guided implant dentistry**. With this innovative technique, a digitally made guide is used during the surgery as a drilling template, thanks to which the implants slope and depth established in the virtual placement (via software) is transferred to the patient.

In particular, the implant insertion can be executed without the surgical guide (simply with a pre-clinical virtual evaluation of implants position) or through a fully guided approach (using the 3D printed guide, that is the result of the virtual implants placement, during the surgery).
During the digital guide design, dental implants are placed without violating the basic rule of modern implantology, according to which the prosthetic project controls the position of the implant and therefore the surgery.
The main advantages of utilizing a surgical guide is to increase the implant insertion accuracy, and to reduce surgery time, trauma, pain, and swelling for patients. In turn, it leads to a shorter recovery time for patients [20].
Given the awkward angles and the inability to see clearly during the surgery, in fact, placing dental implants in the proper location without a guide can lead dental clinicians to excessively incise the patient's gum or to place implants improperly, resulting in oral health complications and millions of dollars spent on insurance claims [21].
For this reasons, digital-guided implant surgery is increasingly used in recent years, promoting both consistent implant placement and predictable patient care.

3.1.1 Digital workflow

Surgical guide manufacturing provides the processing of 3D data obtained from cone-beam computed tomography (CBCT) and digital intraoral impression, as well as CAD/CAM technology for virtual implant planning and guide fabrication.

The digital workflow for guided implant surgery is generally composed by 6 steps: (1) Patient assessment, (2) data collection, (3) data manipulation, (4) virtual implant planning, (5) guide and prosthesis manufacture, and (6) execution of surgery and delivery of a provisional prosthesis [22].

In this paragraph it will be explained the process followed in *Primo Lab S.r.l.* (Figure 3.1), one of the largest dental laboratories in Italy which offers support for over 160 clinics throughout Italy. Due to the large amount of work, in this company the virtual implant planning is not performed by clinicians (as it normally happens)

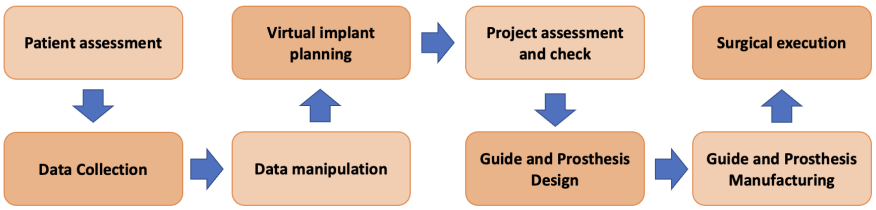


Figure 3.1: Principal steps of surgical guide manufacturing.

but is initially carried out by the *Guided Surgery Team* composed by biomedical engineers. Obviously, the surgical guide is produced once the clinician has accepted the engineer's virtual design. This allows the laboratory to ensure both production speed and quality.

1. Initial Patient Assessment

The process starts from the first appointment with the patient. At this stage the clinician carries out a comprehensive esthetic and functionale evalutation. The assessment should include:

- **Dentition status**: both the periodontal and restorative status of the remaining teeth must be assessed. In case of edentulism, an evaluation of the existing dentition must be performed.

- **Initial radiographic assessment**: the residual bone should be assessed to determine whether a graft or graftless approach is appropriate. This can be done using 2D radiographs.

- **Occlusion**: occlusal evaluation is essential for acceptable esthetics and function. Adequate mouth opening must be assessed, as guided surgery requires additional access, especially in the posterior region.

- **Aesthetic evaluation and prosthetic consideration**: the prosthetic design should provide adequate lip support and a white/pink appearance. The prosthetic plan includes bone-reducing or augmentative procedures.

2. Data Collection

Considering the second level evaluation essential for surgical guide manufacturing, data collection includes CBCT acquisition and digital or physical intraoral impression.

- **CBCT acquisition**: since scattering affects the image quality and thus influences the accuracy of the guided surgery, it is necessary to use different settings of the X-ray device in some particular cases. For example, if the patient has metal implants and/or biocompatible filling material in several areas of the oral cavity, it is necessary to set the parameters in order to have a greater power of the device. It is also important to recognise the distortion of the soft tissue, which can be caused by patient movement.

- **Surface optical scanning**: the inaccuracy of soft tissue and teeth in CBCT can be compensated by an optical surface scan, which depicts the tooth surface and soft tissue contour.
  The dental impression can be obtained with 2 methods:

  - *indirect mode*: clinicians takes a negative imprint of hard (teeth) and soft tissues and, then, it is scanned using a laboratory-scanner to obtain the positive imprint;
  - *direct mode*: an intraoral scanner is used by clinician to scan the area of interest of the patinet's dental arch.

  CBCT data is stored in Digital Imaging Communication in Medicine format (DICOM), and the optical surface scan is stored and transmitted in a Standard Triangle Language format (STL). Both files are sent to the laboratory.

3. Data Manipulation

Engineers provide to import data into a digital implant planning software, to continue with data manipulation which consists of:

Segmentation: CBCT shows both soft and hard tissue, and segmentation of the raw data allows differentiation between them, staining this two anatomical

structures in different colour. In addition, segmentation is useful to reduce image distortion caused by metal scatter and motion artifacts. About segmentation, the first step is to determine manually an appropriate threshold for density (also known as gray level threshold) to clearly visualize hard tissue (bone and teeth, Figure 3.3 and Figure 3.4). Each portion of the scan (called *voxel*) with the same or higher density (compared with the selected threshold) will be visible in the selected volume.
Furthermore, in the digital implant planning software, the gray scale is converted in Hounsfield units. In particular, low threshold indicates soft tissue, instead higher threshold value indicates significant bone.

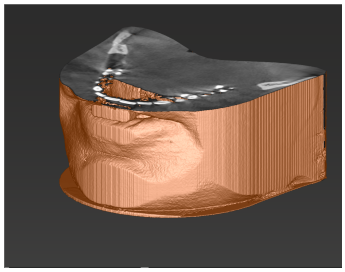


Figure 3.2: Visualization of CBCT data before the application of gray level threshold.

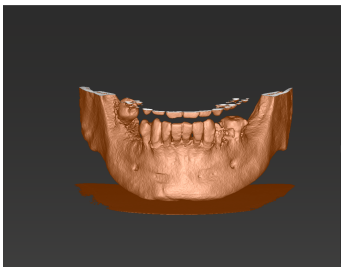


Figure 3.3: Visualization of mandibular after the application of gray level threshold.

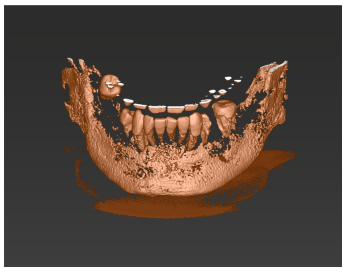


Figure 3.4: Visualization of teeth after the application of gray level threshold.

Since the difference in density between mandibular bone and teeth is not significantly different, engineers have to manually adjust the segmentation. As it is shown in figures 3.5 and 3.6, this manual operation allows to obtain the teeth and mandibular segmentation respectively. For this purpose, the virtual software provides selection and cut tools, such as brushes and scissors. This step is the most problematic in terms of time-consuming and performance.
Separation into teeth and mandibular is necessary if immediate loading is performed. In this case, in fact, the extraction of some teeth will take place directly during the surgery and virtually on the optical impression through appropriate software such as *Exocad* and *3Shape*. Therefore, engineers can use the teeth and jaw pair for alignment with the digital impression but then they will have to exclude

tooth segmentation during virtual planning.

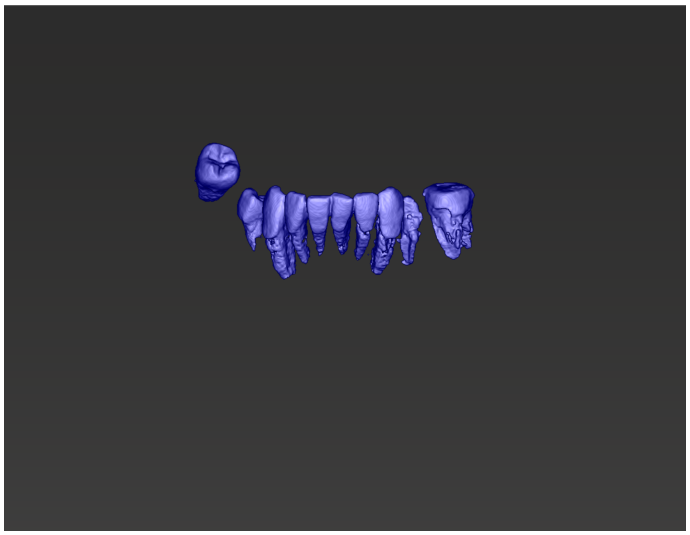


Figure 3.5: Visualization of teeth after the application of gray level threshold and manual segmentation.

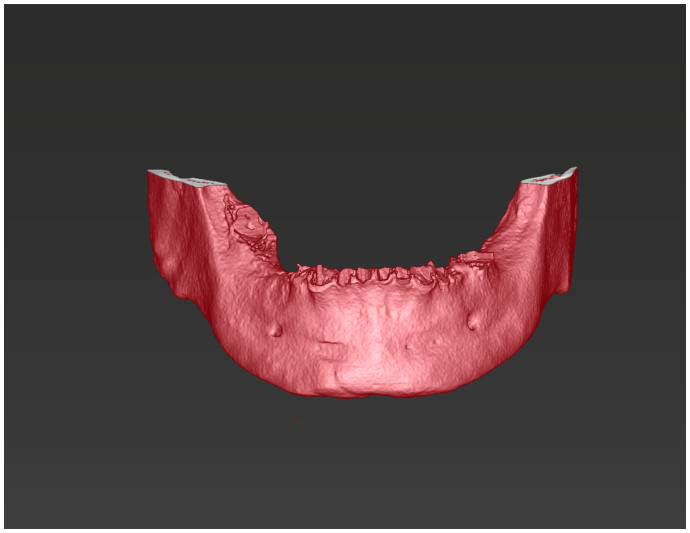


Figure 3.6: Visualization of mandibular after the application of gray level threshold and manual segmentation.

Another problem concerning the segmentation step is that different CBCT devices provide different gray-scale values. For this reason, the threshold selection is subjective because it may be influenced by bone irregularities and maturation. **This**

aspects are important because gray level thresholds may also affect 3D reconstruction and surgical guide fitting.

Identification of panoramic curve: the rendered reconstructed volume has to be correctly oriented into the 3 planes (Figure 3.7), after which the panoramic arch needs to be defined (Figure 3.8).

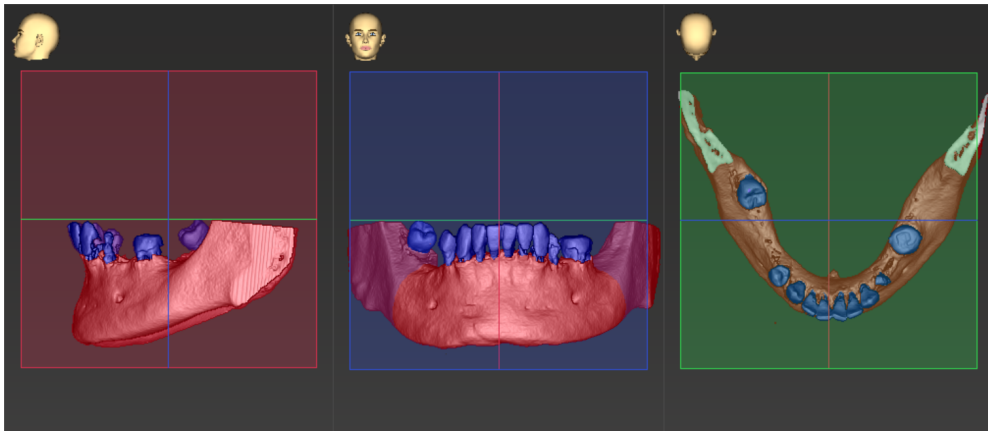


Figure 3.7: Orientation of the reconstructed volume into the 3 planes (longitudinal, transversal and parallel plane).

Inferior alveolar nerve tracking: the software provides a nerve-tracking tool to detect the inferior alveolar canal by placing dots along its path. At this point the software show the teeth, mandibular and nerve segmentation (Figure 3.9). **The nerve tracking is also problematic in terms of time-consuming and performance**.

Merging of CBCT and surface datasets: the STL files of digital impression and segmentations are merged by selecting identical anatomical landmarks on the impression and on the teeth segmentation (Figure 3.10). The resulting alignment was visually examined by a technician to check the correctness and repeat the procedure if necessary.

This step is important because a misalignment between CBCT segmented file and impression data negatively affects the surgical guide stability during the surgery.

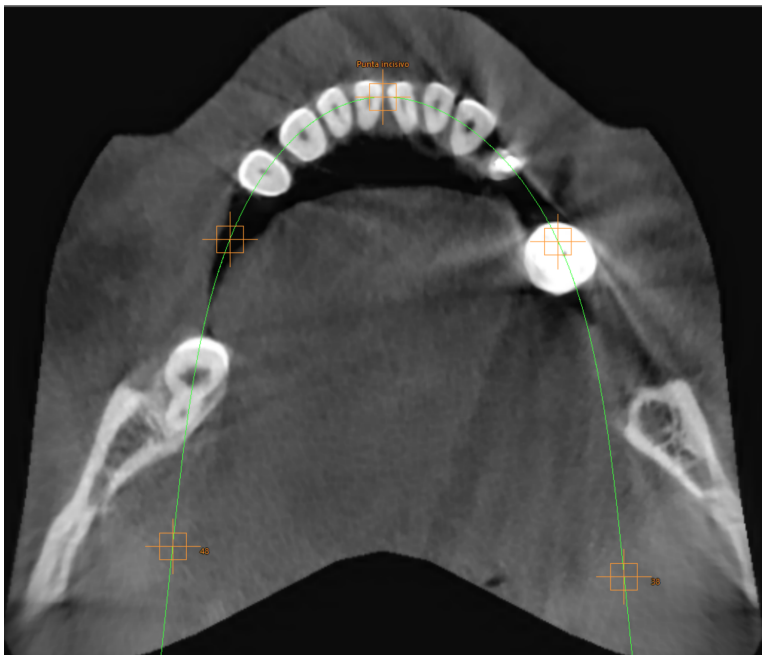


Figure 3.8: Identification of panoramic curve.

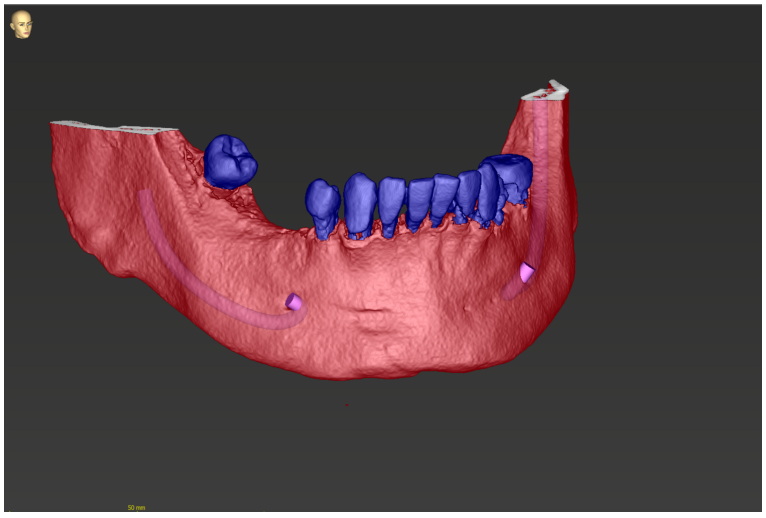


Figure 3.9: Teeth, Mandibular and nerve segmentation.

The quality of the alignment obviously depends from the segmentation quality and from digital impression accuracy.

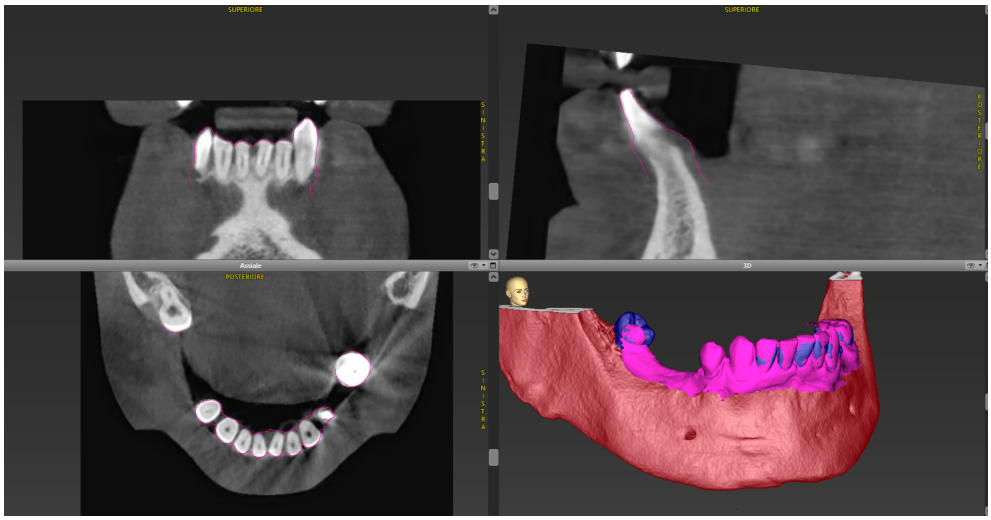


Figure 3.10: Alignment between intraoral scan (pink) and segmented data.

4. Virtual Implant Planning

Once the alignment was verified, the engineer could then perform a virtual implant treatment plan with the software (*coDiagnostiX* is the software used in *PrimoLab*). Considering the ideal prosthetic and anatomic conditions, an implant with proper diameter and length was virtually planned with the implant planning software. Implant position and axis are adjusted according to the existing bone. In case of multiple implants, a parallelization tool can be used. Moreover, during the virtual placement the system offers the possibility to define a safety boundary around and between the implants (Figure 3.11); accordingly, the software warns the user if these boundaries are violated. At this time, the possibility of a flapless approach or the need for bone augmentation is determined.

5. Project assessment and check

Engineers can generate a detailed report from the software, which includes the drilling protocol with corresponding implants. The report is sent to the reference clinic and clinician provide to check the virtual plan.

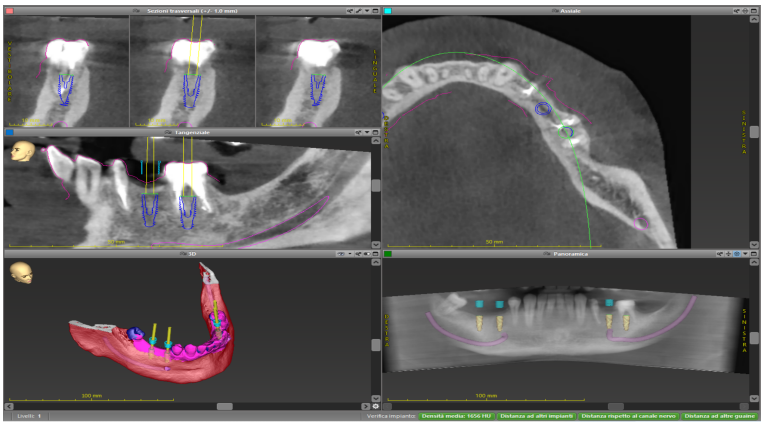


Figure 3.11: Virtual Planning performed by engineers.

6. Guide and Prosthesis Design

Once the virtual plan is settled, engineers can design the surgical guide; it can be supported by teeth, mucosa or bone and it must be designed considering a minimum of 2 teeth for support (Figure 3.12). Moreover, mini-implants, screws, or pins housing can be added to improve the stability of the guide during surgery.

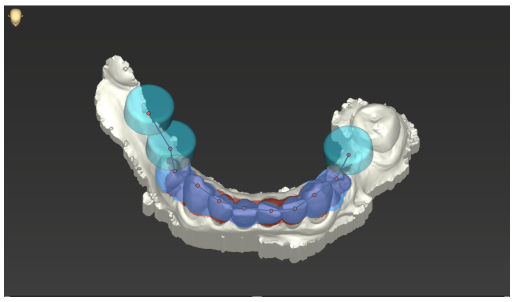


Figure 3.12: Surgical guide project.

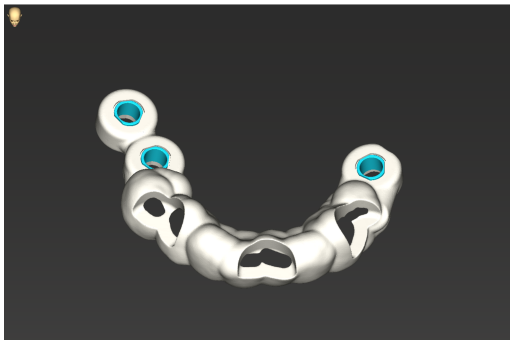


Figure 3.13: Designed Surgical Guide.

7. Guide and Prosthesis Manufacturing

Using a computer-aided design (CAD) software, technicians provide to design the provisional prosthesis which can be full or partial-arch prosthesis.
Subsequently, both surgical guide and prosthesis are exported in STL format and then developed through 3D printing technology (section 3.2).

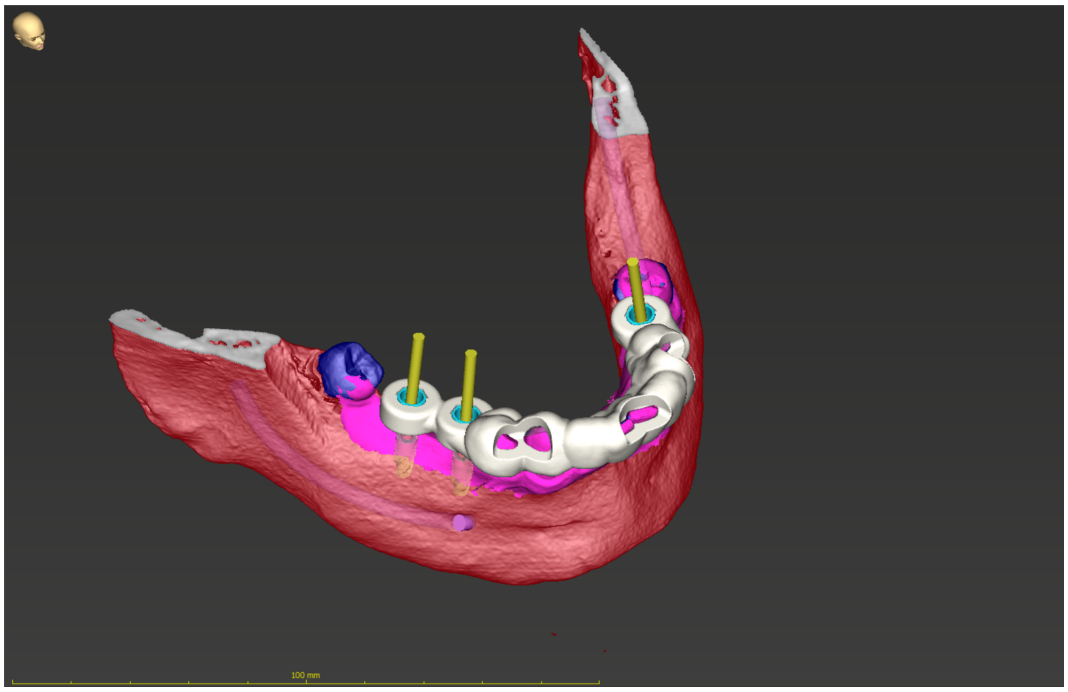


Figure 3.14: Visualization of the designed guide coupled with the segmented data and the virtual planning carried out.

Once the printing is finalized, metal sleeves specific to the surgical guide system are inserted into the surgical guide.

8. Surgical Execution

Once the guide is well positioned in the patient's mouth, the dental clinician follows the planned drilling protocol, which includes implant size and drilling sequence. During surgical treatment, the clinician uses a guided implant kit which is specific to the implant system.

3.1.2 Real case of study

Digital-guided implant surgery, as previously mentioned, has several advantages, such as precise planning and simplified surgical approach, which result in a faster and more predictable surgery. Nevertheless, previous studies have emphasized some disantdavtages, such as deviations between the planned and placed implant position and inaccuracies of the intraoral scanning technique [23].

This paragraph reports a clinical case in implantology in which digital workflow was used throughout the process, pointing out angular and linear deviations between virtual and real treatment. The case is conducted by *Primo Lab* and *Centro Dentistici Primo of Carpi*, with *Dr. Cristian Gheorghiu*.

A 60-year-old female patient, who reported not consuming alcoholic beverages, nonsmoker, not taking any longterm medication, and having no clinically relevant family history information, complained of great difficulty in chewing, evident aesthetic problems and repeated pain due to periodontal abscesses.
Clinical examination showed an end stage periodontitis, with grade II-III mobility of all dental elements (figure 4.8)

Figure 3.15: Initial clinical image.

After the first visit, including an in-depth anamnesis and OPT, there were 2 solutions: mobile total prostheses or total prostheses fixed on implants. The patient

chose the second option so the management option was a fully digital treatment, from surgical planning to final crowns fabrication. For the rehabilitation, CBCT scans of the complete jaw and maxilla were acquired through the *PaX-i3D Smart-PHT-30LFO (Vatech)* device in DICOM format; intraoral scanning was instead performed using the *3Shape TRIOS 4* scanner.
Once these 2 files had been obtained, the *coDiagnostiX (Dental Wings)* software was used for virtual planning: *PrimoLab*'s biomedical engineers provided to segment the DICOM data and to superimpose it to the intraoral scans. Then, the *Exocad* software was used to remove teeth from scans of the upper and lower archs (figure 3.16b); this STL files was merged later with the original intraoral scans (figure 3.16a), in order to subsequently fabricate a mucosa-supported surgical guide.

(a) (b)

Figure 3.16: (a) original upper intraoral scan; (b) upper intraoral scan without teeth.

In the mean time, PrimoLab's dental technicians have developed the double immediate load temporary prostheses: the two projects (upper and lower) were uploaded on the surgical-guided software to begin the virtual plan.
It was conducted following the *All On 4* technique, a dental implant technique where the upper and/or lower set of teeth are replaced with just four implants.
The final virtually designed guide and prostheses were then printed (figure 3.17) respectively with the *FormLabs Form 3+* and *Kulzer Cura 4.0* printers (section 3.2).
During the surgery, *Dr. Christian Gheorghiu* and the staff of the *Centro Dentistico Primo of Carpi*, under local anesthesia, antibiotic prophylaxis, pain-relieving and anti-inflammatory therapy, carried out the remediation, the insertion of the

Figure 3.17: Prosthesis, models of intraoral scans and surgical guide printed after the virtual plan.

implants and the delivery of the 2 temporary screw-retained prostheses in 4 hours. The surgery was conducted using the *Straumann Guided Surgery System Instruments* (figure 3.18).

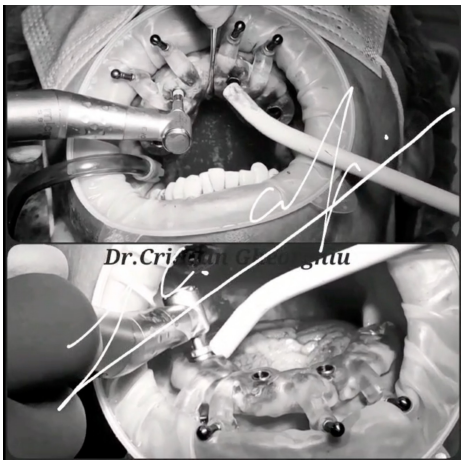


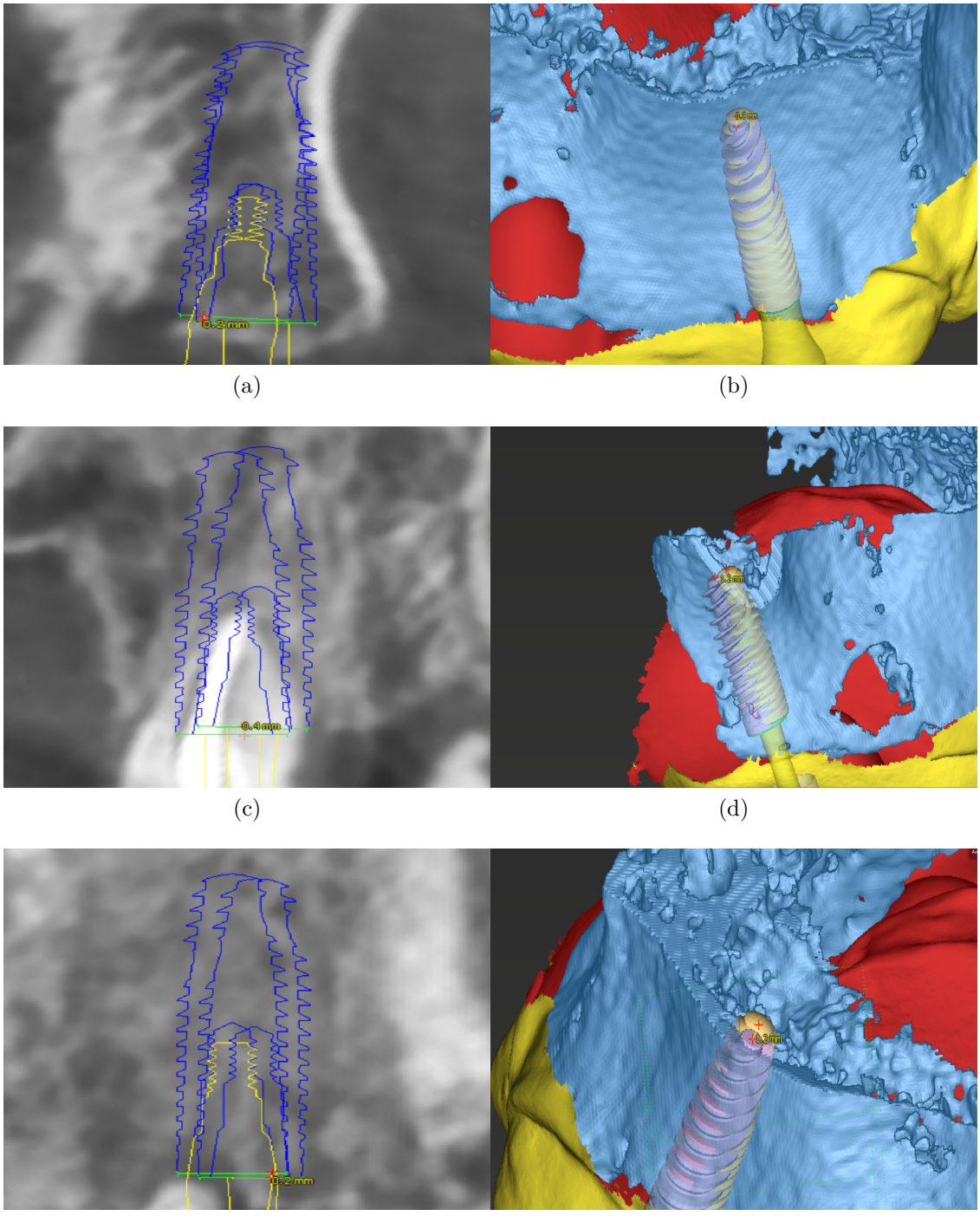
Figure 3.18: Fully guided implant placement, by *Dr. Christian Gheorghiu.*

Afterwards, a Cone Beam CT was acquired to check the implants' position and instructions to follow at home were given to the patient. After a week the suture was removed and the final result is shown in the figure 3.19.
The patient reported that the procedures were not painful, especially the surgical

Figure 3.19: Aesthetic result (with temporary prostheses) one week after the surgery. After 4 months the work was finalized with definitive prostheses.

procedure, for which she had great expectation of success. Also, she was comfortable in the postoperative period, and the prosthetic appointments were rapid and objective, meeting the patient's aesthetic and functional expectations.

The CBCT acquired after the surgery and the virtual plan were compared (Table 3.1, figures 3.20, 3.21, 3.22), finding average deviation of 0.34 mm at the entry point, 0.99 mm at the apex of the implant, and an average angular deviation of 3.9°.
These results can be considered better than those reported in a systematic review by Gustavo Vargas da Silva Salomão et al. [20], who observed a deviation of 0.93 mm at the entry point, 2.2 mm at the apex of the implant, and 5.5° of angular deviation.
This suggests that the case reported in this paragraph is more accurate and precise, also taking into account the lower theoretical stability of the mucosa-supported guide used, compared to the tooth-supported guide of the report described by Gustavo Vargas da Silva Salomão et al. [20].

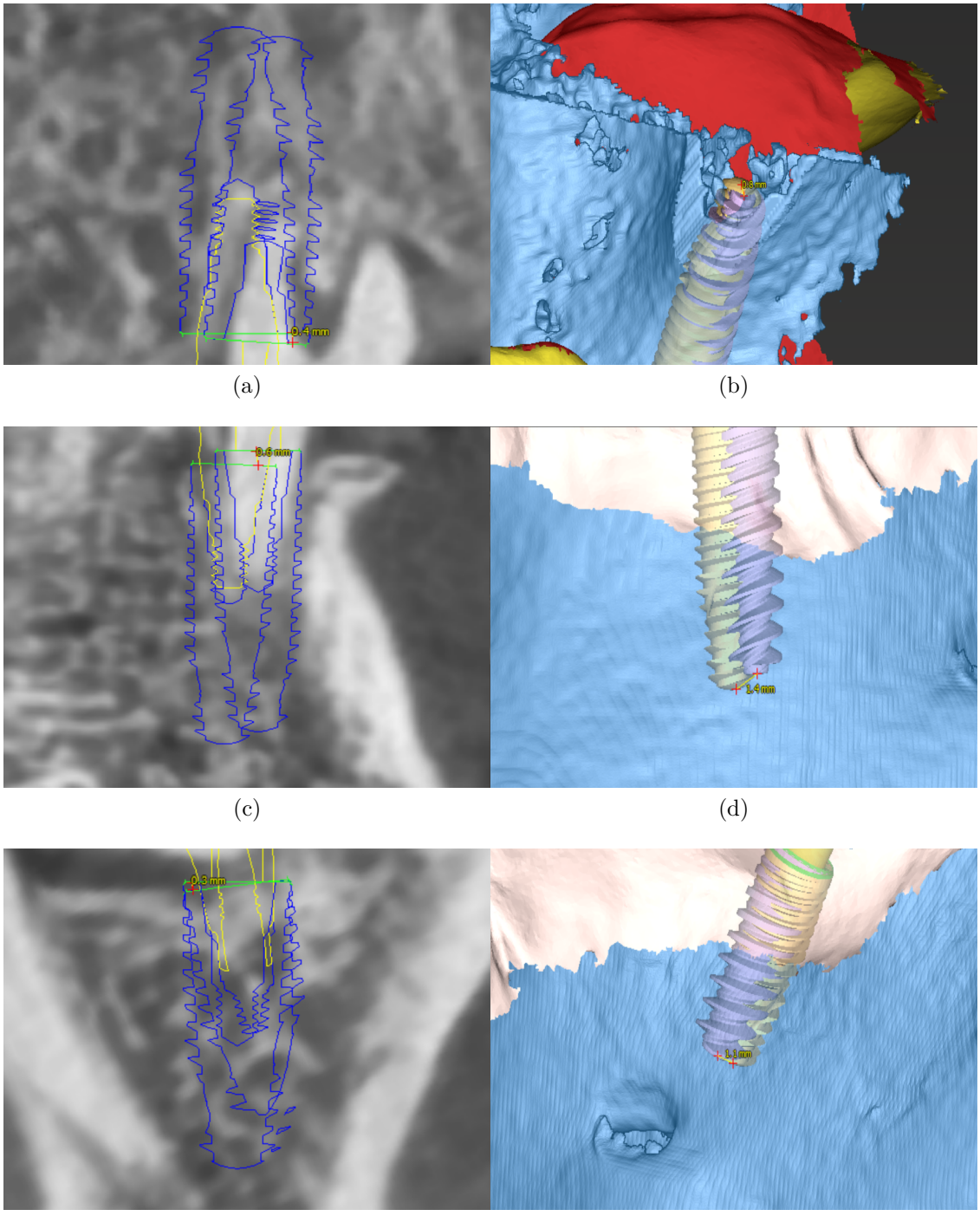


(a) (b)

(c) (d)

(e) (f)

Figure 3.20: Deviations observed in superimposed images of the placed and planned treatment on position: (a, b) 13, (c, d) 16, (e, f) 23.

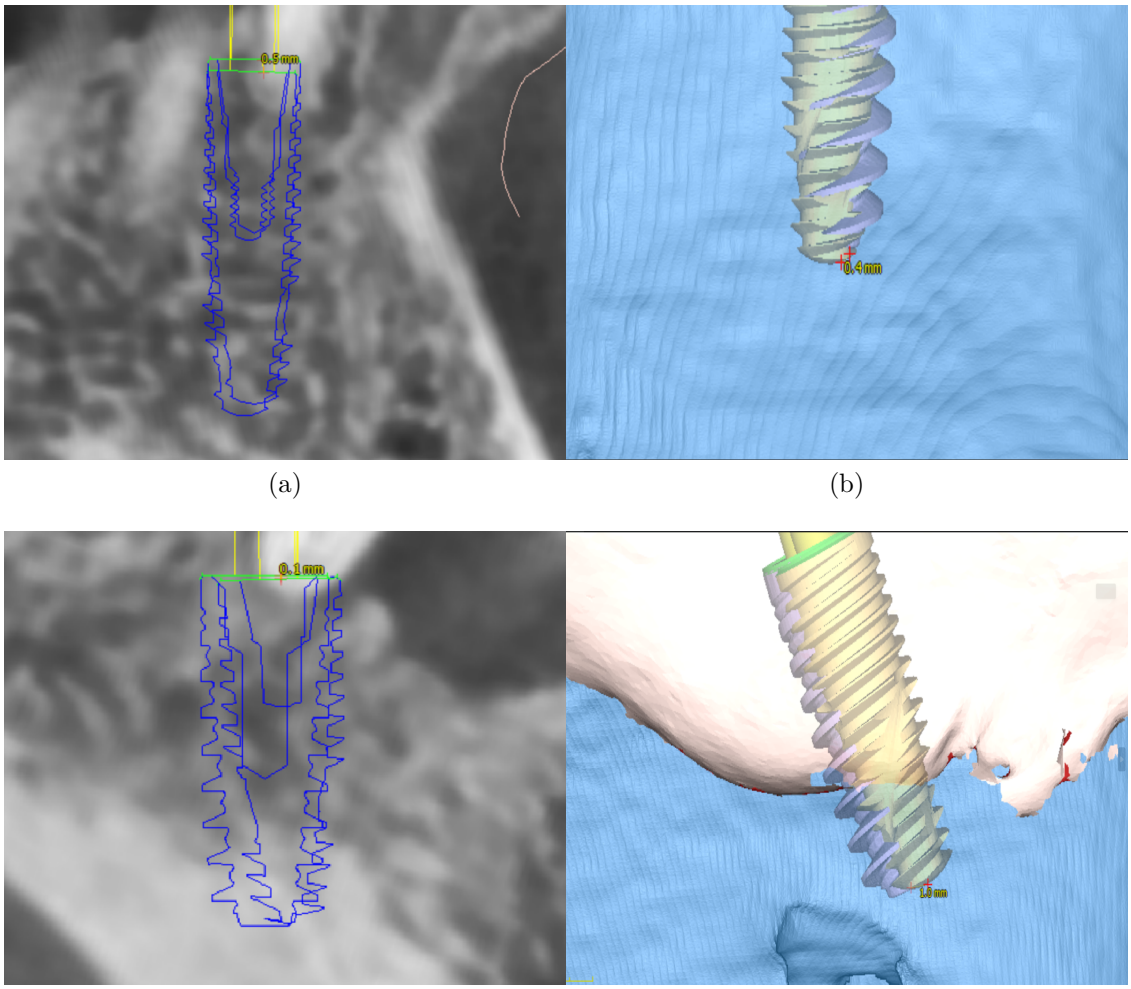


(a) (b)

(c) (d)

(e) (f)

Figure 3.21: Deviations observed in superimposed images of the placed and planned treatment on position: (a, b) 26, (c, d) 33, (e, f) 36.



(a) (b)

(c) (d)

Figure 3.22: Deviations observed in superimposed images of the placed and planned treatment on position: (a, b) 43, (c, d) 46.

Current case report			
Implant position	**Entry point deviation (mm)**	**Apex deviation (mm)**	**Angular deviation (degrees)**
tooth 13	0.2	0.8	5.1
tooth 16	0.4	1.2	4.3
tooth 23	0.2	1.2	5.3
tooth 26	0.4	0.8	4.2
tooth 33	0.6	1.4	2.9
tooth 36	0.3	1.1	6
tooth 43	0.5	0.4	1.9
tooth 46	0.1	1	1.7
Gustavo Vargas da Silva Salomão et al. [20]			
Implant position	**Entry point deviation (mm)**	**Apex deviation (mm)**	**Angular deviation (degrees)**
tooth 11	0.93	2.2	5.5

Table 3.1: Evaluation of deviations between the planned and placed implants position, demonstrating the accuracy of the technique.

3.2 3D printing

Three-dimensional printing can be described as the key technology in the field of dentistry. It is an **additive manufacturing process** based on computer-aided design (CAD) digital models, which uses standardized materials to create 3D objects **layer by layer**.
In contrast to subtractive methods, one of the most important advantages of this technology is the ability to manufacture complex geometries with **low costs** and **time-saving procedures**. For this reasons, it is widely used in dentistry and its applications range from the use of printed drilling templates to transfer virtual implant planning into maxillofacial surgery, to patient-specific prosthesis teeth, occlusal splints and orthodontic applicances.

Generally, there are different printing techniques such as **stereolitography**, **digital light processing**, binder jetting, photopolymer jetting, material jetting, focused filament frabrication, selective laser melting and selective laser sintering [24]. In principle, the printing process followed in all techniques is to stored STL (Standard Transformation Language) file of the object into the CAM software (which generates the machine command (*G-code*) for the printer). The STL format contains the surface triangulation of 3D bodies and each triangular facet is characterized by three corner points and the surface normal vector. Once the file has been uploaded to the software, before printing, the CAM software cuts the STL file into multiple horizontal layers; this operation is called *slicing*. The slice thickness (layer thickness) influences the vertical accuracy of the printer [25].

Stereolithography (SLA) and digital light processing (DLP) 3D printing are the two most common processes for resin 3D printing used in the dental lab's workflow.

SLA is one of the earliest technologies used in 3D printing; its device consists of a **vat** into which is poured the photosensitive liquid resin, a **model build platform**, and an **ultraviolet (UV) laser** to polymerized the resin. Generally, the build platform is submerged in a liquid resin, and the resin is polymerized using the UV laser.

Depending on the type of the platform movement, two styles of 3D printing can be distinguished (Figure 3.23):

- **Top-down**: the build platform is soaked in the resin vat and, after lighting the first layer, the platform moves down and a new layer of resin is added. This process is repeated until the object is created and the light exposure is continuous.

- **Bottom-up**: the platform is submerged at the bottom of the resin vat and the gap between the platform and bottom reached only the distance of one layer. This process is repeated until the object is created and the light exposure is sequential.

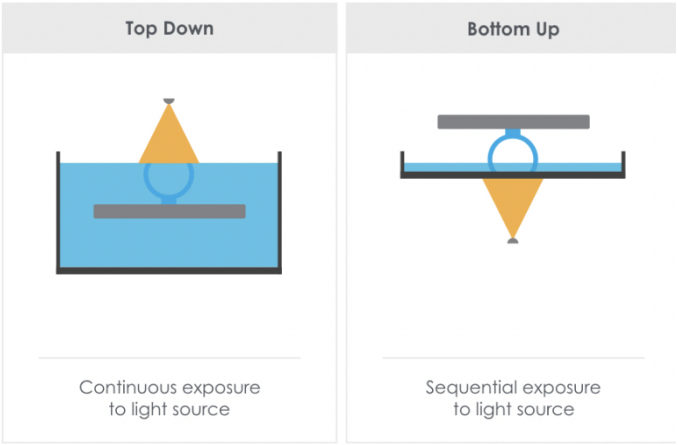


Figure 3.23: SLA technologies

The platform-bottom-up approach has several advantages over the platform top down approach. First, in the second approach, the resin is in direct contact with oxygen as it undergoes polymerization, whereas light-curing occurs at the bottom to avoid oxygen interference in the platform-bottom-up approach. Second, the laser is located at the bottom, which reduces the potential for injury to the operators. Third, the resin can be refilled automatically owing to gravity. For this reason, most SLA printers (as the *FormLabs Form 3+* used in *Primo Lab S.r.l.*, Figure 3.24) currently use this technology.
The second printer technique that is commonly used is the digital light processing

Figure 3.24: FormLabs Form 3+, SLA Printer

(DLP); it is a projection-based SLA techinique and its device is similar to the SLA printer but it is characterized by a microsystem with a rectangular mirror arrangement called a *digital micromirror device* (Figure 3.25).

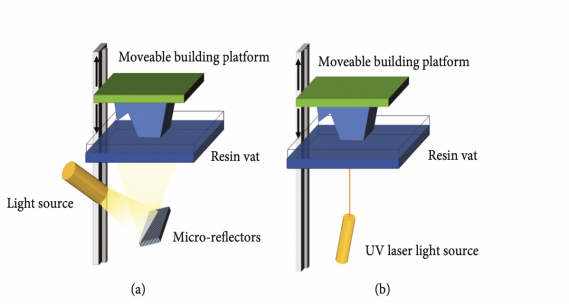


Figure 3.25: (a) Digital light processing (DLP). (b) SLA processing

This techonology is usually applied in the bottom-up approach. Adjusting the angle of the micromirros, the light can be refracted from the source onto the bottom of the vat (which is the projection surface), therefore micromirrors act as light switches and the resolution of the projected image corresponds to the number of mirrors. The advantage of DLP technology in comparison to the SLA is that the entire layer can be built by a single shot of laser exposure rather than scanning each pixel one after the other as it is in SLA technology. For this reason, the construction time is indipendent of the layer geometry and the number of objets. Furthermore, the resolution can be higher, depending on the integrated DLP micro-reflector system, but the cost of the printer (Figure 3.26) is undoubtedly higher.

Figure 3.26: Kulzer Cura 4.0, DLP Printer

Both regarding SLA and DLP technology, the maximum resolution achievable with printers of this type ranges from 25 μm to 100 μm. A lower layer thickness leads to high resolution object surfaces but is not conductive to a fast produce time. In addition, the layer thickness is influenced by the amount of photoinitiators (3-5 wt%) contained in the resin, the irradiation conditions (wavelength of UV laser, power, and exposure time), and the temperature of the monomer and absorbent ingredients. Depending on the printer, the wavelength of the laser is set at 385 nm or 405 nm. The most commonly used monomers are methacrylate, epoxy, and functionalized vinyl ether resin, instead photoinitiators used frequently in dentistry include hydroxyacetophenone, benzoin, benzoin ethers, and posphine oxides [26].

Chapter 4

Materials and Methods

4.1 Segmentation time analysis

In order to identify the most critical planning steps and consequently to reduce the digital treatment planning time, last 200 cases of surgery performed in *PrimoLab* between December 2022 and January 2023 are compared.
Planning analyzed were performed according to the guided surgery protocol detailed in Chapter 3.

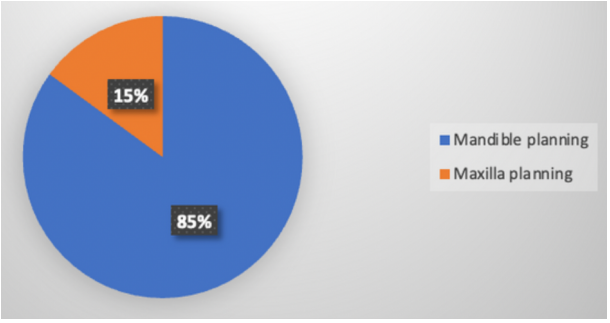


Figure 4.1: Amount of digital-guided surgery for mandible and maxilla

The analysis shows that 85% of cases belong to mandibular planning, and only 15% to maxillary planning.
This result can be justified considering that the mandible is more subjected to the resorption process and, consequently, more patients are interested by loss of lower

teeth.
Resorption, or the process of breaking down and reabsorbing bone tissue, occurs naturally in the human body as part of bone remodeling. This process helps maintain bone strength and shape by removing old or damaged bone tissue and replacing it with new bone tissue. On the other hand, excessive bone resorption can lead to osteoporosis and other bone-related disorders, which can have negative consequences for overall health and quality of life.
However, resorption is generally more pronounced in the mandible than in the maxilla for several reasons:

- Mechanical stress: The mandible is subjected to greater mechanical stress than the maxilla due to its role in mastication (chewing), speech, and other jaw movements. This increased stress causes more frequent and intense bone remodeling, including resorption [27].

- Hormonal factors: The mandible has a higher level of sensitivity to certain hormones, such as parathyroid hormone (PTH) and calcitonin, which play a key role in regulating bone metabolism. PTH, for example, stimulates bone resorption, and studies have shown that the mandible is more responsive to this hormone than the maxilla [28].

- Anatomic factors: The shape and structure of the mandible also contribute to its greater degree of resorption. The mandible has a thinner cortex (outer layer of bone) and a higher ratio of trabecular (spongy) bone to cortical bone than the maxilla. This makes it more susceptible to resorption, as trabecular bone is more metabolically active and responsive to hormonal signals than cortical bone [29].

Furthermore, lower teeth are more commonly lost than upper teeth because they are more susceptible to dental caries due to their location in the mouth, where they are more exposed to food and drink particles and harder to clean than the upper teeth.
In addition, when placing dental implants in the posterior mandible, there is a risk of damaging the inferior alveolar nerve (a sensory nerve that provides sensation to the lower lip, chin, and teeth), which can cause temporary or permanent numbness,

pain, or other sensory disturbances; this risk can be minimized by using guided implant surgery techniques.

Once it has been established that mandibular planning is the most frequent one, a temporal analysis is conducted for each mandible planning step (*data manipulation*, *virtual implant planning* and *guide and prosthesis design*).

For this purpose, time required for entire virtual planning of 40 patients is measured, as shown in the table below.

Task	**Time (Mean ± std)**
Teeth segmentation	25.07 ± 4.10 min
Mandible segmentation	12.33 ± 5.21 min
Inferior alveolar nerve tracking	4.13 ± 1.04 min
Merging of segmentation and dental impression	2.08 ± 1.53 min
Virtual Implant Planning	8.25 ± 3.11 min
Guide and Prosthesis Design	5.27 ± 1.45 min

Table 4.1: Time analysis results for mandibular planning

The results have shown that the segmentation of teeth and mandible from the initial CBCT volumes accounts for approximately 60% of the total planning time. As shown in figure 4.2, moreover, teeth segmentation generally requires more time (25.07 ± 4.10 min for teeth, 12.33 ± 5.21 min for mandible).

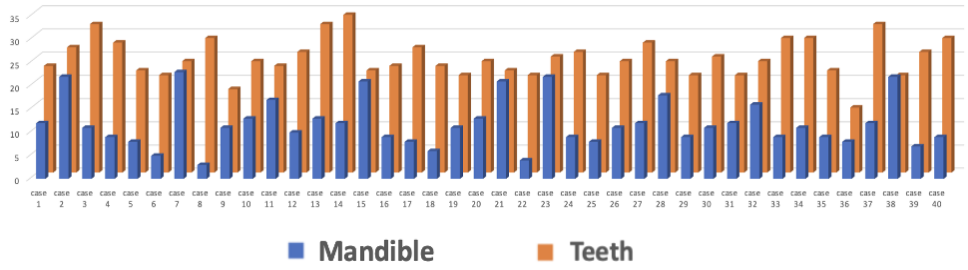


Figure 4.2: Segmentation time divided into teeth and mandible for 40 cases.

Biomedical engineers from the *PrimoLab Guided Surgery Team* have identified the **segmentation step** as the most tedious and time-consuming activity also from a macroscopic perspective. This is due to the insignificant difference in density between mandibular bone and teeth, which requires engineers to manually adjust the segmentation resulting from threshold selection using virtual planning software, as explained in Chapter 3: separating teeth and mandibular is necessary for immediate loading, as some teeth will be extracted directly during surgery and virtually on the optical impression through appropriate software. This allows engineers to use teeth and jaw pairs for alignment with the digital impression, but they must exclude tooth segmentation during virtual planning. Another challenge with the segmentation step is that various CBCT devices provide different gray-scale values. This leads to subjective threshold selection, which may be influenced by bone irregularities and maturation. This is significant because gray level thresholds can also affect 3D reconstruction and surgical guide fitting.

In summary, automating data segmentation in implant guided surgery can be considered a smart decision for several reasons:

- Efficiency: Automating data manipulation can significantly reduce the time required to plan and perform implant guided surgery. This not only increases the efficiency of the surgical process, but also allows for faster treatment times and improved patient outcomes.

- Accuracy: Automated data manipulation eliminates the possibility of human error, ensuring that the data used in implant guided surgery is accurate and

reliable. This is especially important in complex cases where even a small error in data can have significant consequences for the patient.

- Consistency: Automated data manipulation ensures that the same procedures and protocols are followed consistently across all cases. This helps to reduce the risk of errors or variations in treatment outcomes, and ensures that patients receive the same high-quality care regardless of the surgeon performing the procedure.

- Integration: Automated data manipulation can be integrated with other technologies, such as computer-aided design and manufacturing (CAD/CAM) systems and robotic surgery platforms. This allows for even greater precision and accuracy in implant guided surgery, and can further improve patient outcomes.

Given the possibility of automating the segmentation of initial CBCT volumes, a deep learning approach is proposed: different 3D convolutional neural networks (CNNs) have been used for multiclass segmentation of mandible, teeth, and background in CBCT scans.

4.2 Data preparation

The available original dataset consists of 500 CBCT (Cone Beam Computed Tomography) volumes in DICOM (Digital Imaging and Communications in Medicine) format, acquired by different dental clinics belonging to *Centri Dentistici Primo* company. Each volume includes the entire patient's jaw and maxilla centered in the imaging volume. Every patient was anonymized, hence there is the possibility to access few personal details - namely gender, age and year of the scan. Specifically, 62% of the patients are female, all the scans were performed between 2018 and 2022, and volumes belong to patients with ages in range [21-82] with the highest frequency in range [60-70]
Considering the original dataset, the 500 volumes obtained by the *PaX-i3D Smart-PHT-30LFO (Vatech)* device have different shapes, (450,550,550) or (625,625,625) for the Z, Y and X axes respectively, where the Z axis represents the number of slices which composed each subject. This difference in shape depends from the

device settings used during acquisition, in which the user can choose a voxel size of 0.2 mm (to obtain a volume with shape (450,550,550)) or 0.08 mm (to obtain a volume with shape (625,625,625)). Consequently, pixel spacing and slice thickness can be 0.2 mm or 0.08 mm. In addition, the FOV size (field of view) is 100X80 mm and the tube voltage/current ranges from 50 kVp/4mA to 99 kVp/16 mA; tipically higher X-ray intensity is set for larger circumferences of the patients' head. For all volumes the distance source to patient and the distance source to detector are respectively 428 mm and 600 mm. The images are in gray scale, encoded on 14 bits.

4.2.1 Data segmentation

In order to train nets with the dataset decripted above, it was necessary to build the manual segmentation masks in which each voxel is identified as teeth, mandible or background.

Nowadays, several software tools for processing DICOM files are available freely online and on the market. Among these tools the most used and best known segmentation software are *3D Slicer*, *Mimics*, *Medviso*, *ITK-SNAP, MeVisLab*, and *ImageJ*. They present different segmentation algorithms, reconstruction accuracy, levels of usability, and commercial costs and can be grouped into three different categories: CT embedded, high-end licensed, and open source software [30].

Many studies in the scientific literature have tried to compare segmentation software, focusing on the reconstruction of specific anatomical districts or related to particular kinds of diseases. In particular, considering maxillofacial surgery, Johari Yap Abdullah et al. [31] compared commercial software *Mimics* (used as the gold standard) with open source *3D Slicer*, *ITK-Snap*, *Dolphin 3D*, *Medical Imaging Interaction Toolkit (MITK)* and *InVesalius*. The parameters used for the comparisons include the Hausdorff distance, Wilcoxon signed rank test, Dice score, the segmentation volume, time, and the number of voxels. The authors found out that the 3D models produced using Mimics and the other tools were comparable and they suggested using open source software to minimise the operational cost.

For this reason, due to the simplicity of usage and free availability, the chosen software tool to segment data is *3D Slicer*, uploaded on a laptop with a 16GB NVIDIA

GeForce RTX 3080 Ti, Intel i9-12900HK and 32 GB of RAM.

Thresholding Segmentation

Each DICOM volume had been loaded on *3D Slicer*, with the possibility to scroll through slices in the axial, coronal and sagittal view (figure 4.3).

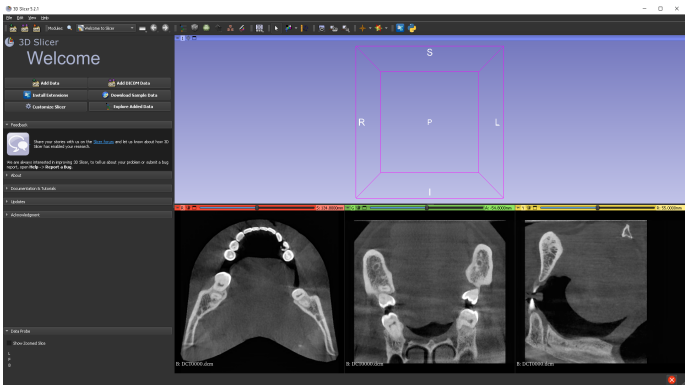


Figure 4.3: From left to right the axial, coronal and sagittal view of a 2D slice belonging to the first volume ('P1' subject).

Wishing to use a single network for the segmentation of the volume into teeth, mandible and background (as explained in the following chapter), each volume had been segmented using the 3D slicer *Segment editor* module, applying a threshold range (from 1000 to 4000) on the entire volume. This module is for specifying segments (structures of interest) in 2D/3D/4D images and mimics a painting interface like photoshop or gimp, but work on 3D arrays of voxels rather than on 2D pixels. Moreover, the module offers editing of overlapping segments, display in both 2D and 3D views, fine-grained visualization options, editing in 3D views, create segmentation by interpolating or extrapolating segmentation on a few slices, editing on slices in any orientation.
The segmentation method used is called *Global threshold segmentation method*. It is the basis for a number of segmentation methods, mainly for its simple implementation and computational simplicity. The basic step in thresholding is to choose the right threshold. Later, the image is divided into two groups. The first group contains pixels whose degree of brightness is greater than the threshold value and

the second one whose degree is less than or equal to the threshold value. Subsequently, the image is converted into binary form, i.e. black and white image.
In this specific case, segmentation thresholding refers to setting the pixels whose gray value is greater than the lower threshold value ($t_1 = 1000$) and less than the greater threshold value ($t_2 = 4000$) as white and the other pixels as black. In particular, assuming that the input volume is V, the height is H, the width is W and the depth is D, and that $V(r, c, d)$ represents the gray value of column c and row r of $d-th$ slice of the V volume, $0 \leq r < H$, $0 \leq c < W$, $0 \leq d < D$, the output volume after global threshold processing is O, $O(r, c, d)$ represents the gray value of column c and row r of $d-th$ slice of the O volume, then:

$$O(r,c,d) = \begin{cases} 1, & \text{if } t_1 \leq V(r,c,d) \leq t_2 \\ 0, & \text{otherwise} \end{cases}$$

Once this global threshold method is applied, *3D Slicer* shows the entire volume selected through the application of the threshold range (figure 4.4). The segmen-

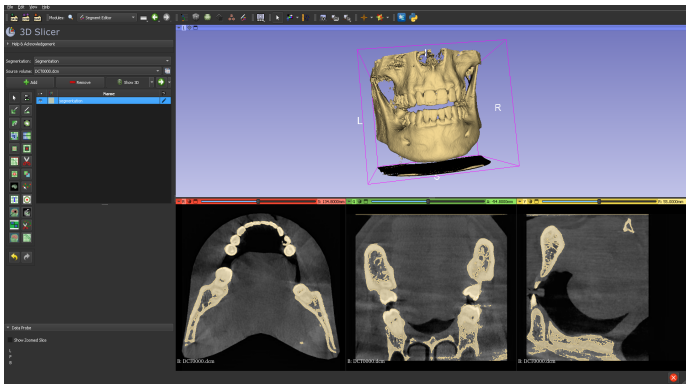
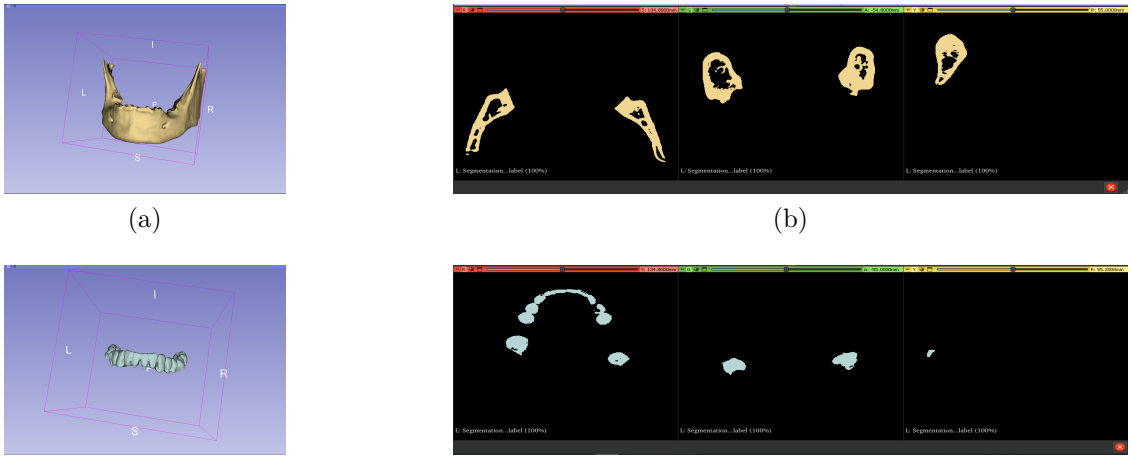


Figure 4.4: 3D view of the entire volume selected through the application of the threshold range ('P1' subject).

tation of teeth and jaw had been conducted separately; the chosen threshold range is the same ($t_1 = 1000$ and $t_2 = 4000$).
Thanks to the *Scissors* tool available with the *Segment Editor* module, the mandible region was clipped; this tool works on both slice view and 3D views. Then the mandible segment was exported to binary labelmap, in DICOM format. In this way each masked volume has the original shape ((450,550,550) or (625,625,625)), with voxels of value 1 representing the jaw and the others equal to 0. The same

process was followed for teeth segmentation, with voxels of value 1 representing the teeth and the others equal to 0 (figure 4.5).



(a) (b)

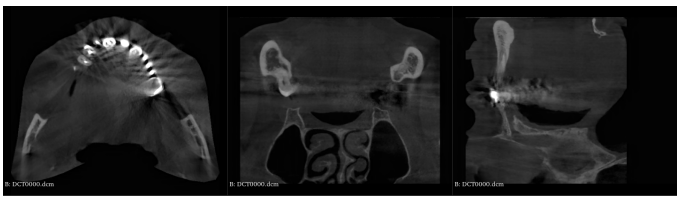
(c) (d)

Figure 4.5: Segmentation view for 'P1' subject: (a) 3D view of mandible segmentation; (b) 2D view of mandible segmentation; (c) 3D view of teeth segmentation; (d) 2D view of teeth segmentation.

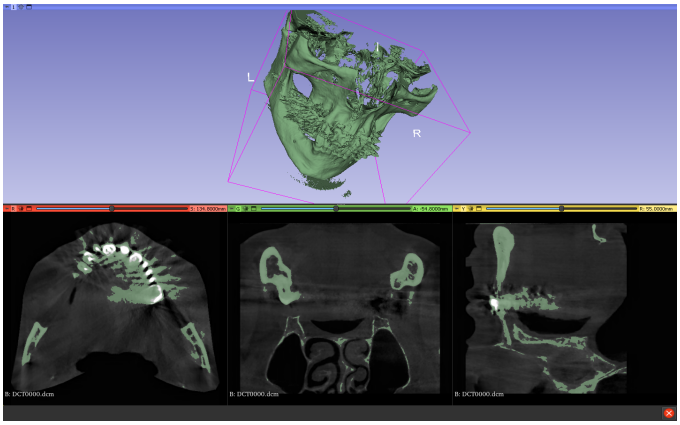
4.2.2 Data correction and masking

Metal artifacts

During the process of segmenting the mandible and teeth, it was necessary to eliminate some volumes from the original dataset. This was particularly challenging due to the prevalence of metal artifacts in most subjects, which made it impossible to achieve accurate manual segmentation (as illustrated in Figure 4.6).



(a)



(b)

Figure 4.6: (a) axial, coronal and sagittal view of a 2D slice belonging to 'P212' subject; (b) Segmentation view for 'P212' subject after the application of the threshold range. As can be seen, metal artifacts make manual segmentation impossible.

It is widely acknowledged that the presence of high-density and high-atomic number metallic structures can lead to the generation of artifacts on CBCT images, resulting in a significant reduction of image quality and rendering the image diagnostically unreliable. The artifacts caused by metal implants or crowns manifest as black and white stripes on the reconstructed CBCT image, which can overlap and obscure the anatomical structures present. Specifically, in the context of radiotherapy treatments, these artifacts not only pose challenges in accurately

delineating tumor volumes and critical organs near them, but also affect the precision of dose calculations made by treatment planning systems that rely on CT images as a foundation for their algorithms.[32].
Passing through metal objects causes extreme hardening of the bundle giving rise to:

- *streak artifacts* after passing through a single metal object;
- *beam cancellation artifacts (beam "starvation")* between two metal objects at close range, as is often the case in dental imaging due to amalgam fillings.

This may cause more time to evaluate the images and even compromise the diagnosis in areas where artifacts are observed [33].

Therefore, following the elimination of certain subjects, the dataset now comprises 148 volumes, each with dimensions of (450, 550, 550). The uniformity in shape across all volumes can be attributed to the presence of a relatively small number of volumes with dimensions of (650, 650, 650) in the original dataset, which accounts for approximately 2% of the total dataset.

Masking

The dataset obtained as explained in the previous paragraph was divided into a (train, validation, test) set with proportions (70%, 10%, 20%). So, there were 104 subjects for the train set, 16 subjects for the validation set and 28 subjects for the test set; all characterized by shape (450, 550, 550).
Since the goal is to use a single network for the volume segmentation into three classes (background, mandible and teeth), a 3D mask in DICOM format was obtained for each subject of the train set and validation set; it has shape (450, 550, 550) and contains zeros for the background, ones for the mandible and two for teeth.
For each volume the new 3D mask is the result of difference between the mandible mask and teeth mask, forcing to two the negative values (figure 4.7).

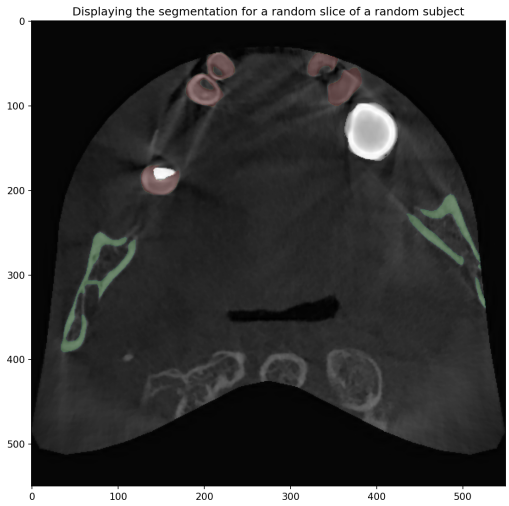


Figure 4.7: Overlapping of a random slice belonging to a random subject of the train set and a slice of the new 3D mask. Green pixels are equal to one, red pixels are equal to two; others are equal to zero.

4.3 Preprocessing

The preprocessing pipeline (figure 4.8) for input volumes can be described as follow.

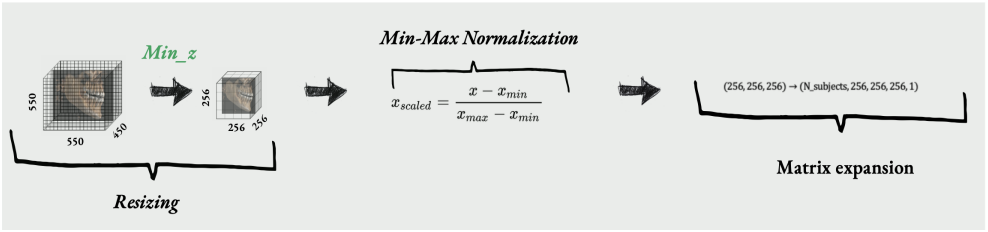


Figure 4.8: Preprocessing pipeline for input volumes

1. ***Region of interest (ROI)* definition:** starting from the volumes and masks processed as previously described, a Region of Interest (ROI) is defined. The sought-after ROI can be identified as the minimum volume containing all manual segmentations of mandible and teeth belonging to the Training Set, increased by 20% for each dimension. The increase in size is done to ensure that all cases in the other Sets are also included. Of course, if, as a result of the

increase in size, the ROI exceeds the maximum borders, the limit is set to the edge of the volume itself (450, 550, 550). The result of this algorithm is a ROI with shape (281, 550, 550).

2. **Resizing**: in order to use a network model provided by *Keras* [34], it is necessary to have input volumes whose dimensions are multiples of 32, preferably cubic. Volumes are so further resized to a resolution of 256x256x256 or 128x128x128 (more details later). The choice is made in order to reduce GPU memory usage.

3. **Min-Max Normalization:** input data are min-max normalized to the range [0, 1] according to the formula 4.1.

$$x_{norm} = \frac{x - min(x)}{max(x) - min(x)} \tag{4.1}$$

   Min-max normalization is important in deep learning because it helps to rescale the input features to a common range, typically between 0 and 1. This is essential for many deep learning algorithms because it ensures that the input features have similar magnitudes, which can improve the performance of the optimization process during training.
   Without normalization, features with large values may dominate the optimization process, leading to slow convergence and poor performance. Additionally, deep learning models can be sensitive to the scale of the input features, and normalization can help to reduce this sensitivity and make the model more robust.
   Furthermore, normalization can help to improve the generalization performance of the model by preventing overfitting. Without normalization, the model may memorize specific patterns in the training data that are not relevant for the test data, leading to poor performance on new, unseen examples. By rescaling the input features, normalization can help to prevent overfitting and improve the generalization performance of the model [35].

4. **Matrix expansion:** in order to use a set of Keras models for segmentation of 3D volumes [36], one dimension is added.

The preprocessing pipeline (figure 4.8) for masked volumes is similar and can be described as follow.

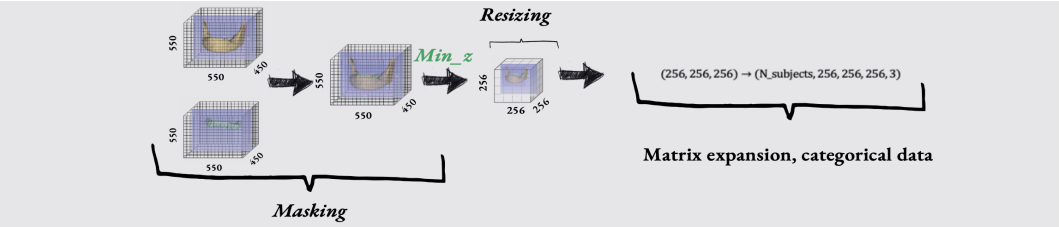


Figure 4.9: Preprocessing pipeline for masked volumes

1. ***Region of interest (ROI)* definition;**
2. **Resizing;**
3. **Maximization of the distance between voxels:** after the resize operation, the voxels of the masked volumes are no longer populated by integers (0 for background, 1 for mandible, 2 for teeth). To avoid penalizing any class, two thresholds were selected ($th_1 = 2/3$ and $th_2 = 4/3$). Specifically, voxels with values less than th_1 are set to 0, voxels with values between th_1 and th_2 are set to 1, and voxels with values greater than th_2 are set to 2.
4. **Matrix expansion and conversion to categorical data:** volumes are converted to categorical data; the result is a matrix with shape 128x128x128x3 (or 256x256x256x3) and each layer (represented by the fourth dimension) corresponds to each classes (background, mandible and teeth).

4.4 Data augmentation

Data augmentation is a powerful technique used in deep learning to artificially increase the size of a training dataset by generating new examples through various transformations. This can help to improve the generalization performance of the model by exposing it to more diverse and realistic examples.
There are many types of data augmentation techniques, including geometric transformations (e.g., rotation, scaling, and translation), color and contrast adjustments, noise injection, and others. These techniques can be applied individually or in combination, and their effectiveness may depend on the specific application and dataset.
As described by Phillip Chlap et al. [37], data augmentation can lead to significant improvements in accuracy, particularly when the size of the training dataset is limited. The authors discuss some challenges associated with data augmentation, such as increased computational cost and the risk of overfitting to augmented examples. They suggest that careful selection of augmentation techniques and hyperparameters can mitigate these issues and improve the robustness of the model.
Overall, data augmentation is a valuable tool for improving the performance and generalization ability of deep learning models. By generating new examples that capture the variability and complexity of real-world data, data augmentation can help to make deep learning models more effective and applicable to a wide range of problems.

The only operation carried out in this regard was a rotation of volumes along the vertical axis, simulating the different positioning of the human skull during the CBCT examination. The chosen range for random rotations is $\pm$ 20°, which allows for adding variability and avoiding overfitting.

4.5 Postprocessing

About postprocessing, the preprocessing pipeline was retraced in the opposite direction (figure 4.10), in order to restore volumes in their original shape and size. In addition, a cleaning of the automatic masks obtained in output from the networks has been carried out. All the post-processing steps performed are described below.

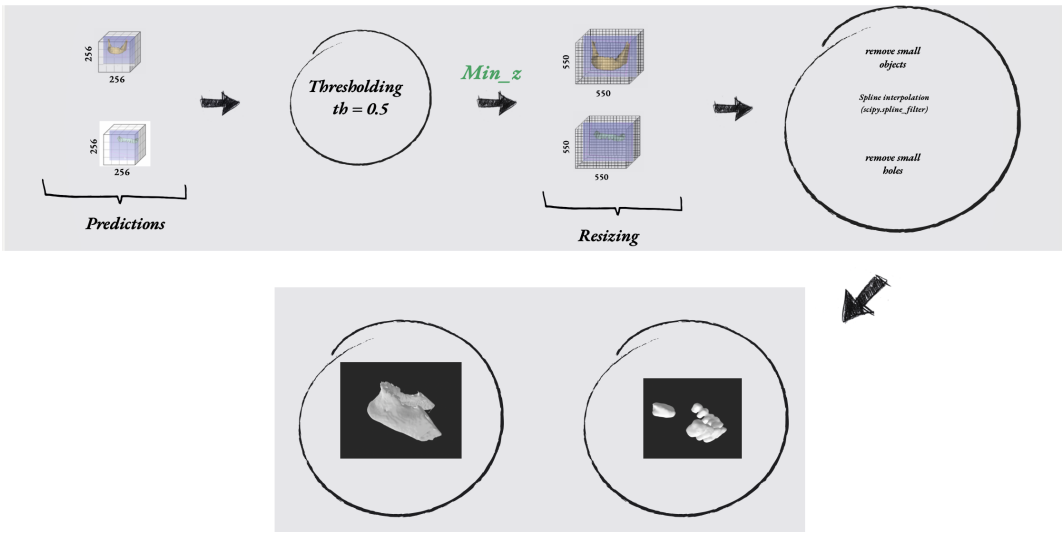


Figure 4.10: Postprocessing pipeline

1. **Extraction of predicted masks from the network**: after the prediction, a thresholding (threshold equal to 0.5) is performed on background, mandible and teeth masks, in order to maximize the distance between the expected values (0 and 1).

2. **Resizing and *Zero-Padding***: volumes are resized to the original ROI shape (281, 550, 550). Then a *zero-padding* operation is performed to have the original shape for each volume (450, 550, 550).

3. **Spline filter**: as shown in figure 4.11, output volume appears to be non-continuous; this happens mainly due to the small shapes of the input data with which the network works. For this reason, a *spline filter* is applied in order to smooth the output masks.

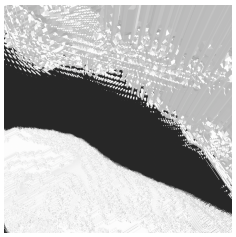


Figure 4.11: Output volume before spline filtering

In imaging, a spline filter is a type of digital filter used to smooth or interpolate images. It works by fitting a spline function to the intensity values of the image pixels, which allows for flexible and continuous modeling of the underlying image features.
Spline filters are often used in image processing to remove noise or artifacts from images, or to interpolate missing data. They can also be used for edge detection, image registration, and other applications where smoothing or interpolation of images is needed.
There are different types of spline filters for imaging, such as cubic or higher-order splines. The choice of filter type and other parameters (such as the degree of smoothing or the number of knots) can affect the performance and accuracy of the filter [38].
In this postprocessing pipeline is used a *quintic-order spline filter* supplied by *scipy.ndimage* package [39].

4. **Removing small objects, small holes and incomplete segmentations**: an improvement of the automatic masks is implemented by removing small incorrectly segmented areas from the network, and filling them in case small holes are present. The operation was performed through the appropriate *remove small objects* and *remove small holes functions* taken from the *skimage.morphology* library [40].

5. **Saving in STL format:** output volumes are finally saved in STL format (with the *meshlib* package [41]). This is necessary so that the automatic teeth and mandible segmentations can be used for virtual planning.

4.6 Network Architecture and Training configuration

We define a model for image segmentation using the **Unet architecture** from the *segmentation models 3D* library [42]. The model architecture is based on the **ResNet50 backbone**, which is a deep convolutional neural network that consists of 50 layers. We set the *encoder weights* variable to *"None"*, so the backbone network is initialized with random weights and not using the pre-trained weights that were learned on the ImageNet dataset (so *transfer learning* is not used). Usually, transfer learning is typically used in scenarios where you have limited training data or computational resources, and want to leverage the learned feature representations from a pre-trained network to improve the performance of your new model on a specific task. Since we have a large amount of data, we are able to train a good model from scratch without using transfer learning.

The decoder block used is the **transpose** type; it refers to the use of transposed convolutional layers (also known as deconvolutional layers or fractionally-strided convolutions) in the decoder part of a neural network. The decoder block is responsible for upsampling the feature maps that were downsampled by the encoder block, in order to obtain a dense pixel-wise prediction for the image. The transpose convolutional layers perform this upsampling by reversing the operation of the convolutional layers in the encoder block, and inserting zeros between the pixels to increase the spatial resolution of the feature maps.
The "transpose" decoder block type is one of several options that can be used to perform this upsampling, and it has become a popular choice due to its simplicity and effectiveness. Other decoder block types include "upsampling" (using simple upsampling and convolutional layers), "depthwise" (using depthwise separable convolutions), and "atrous" (using dilated convolutions).

The model takes grayscale images as input with dimensions $(img_w, img_h, img_d, 1)$, where img_w, img_h, and img_d are the width, height, and depth of the input image, respectively.

The model has **three output classes for multiclass segmentation**, which are specified by the *nclasses* variable. The activation function used for the output layer is **softmax**, which is appropriate for multiclass classification. The loss function used is **categorical cross-entropy**, which measures the dissimilarity between the predicted probability distribution and the true distribution of classes.

The model is trained using the **Adam optimizer** with a **polynomial learning rate** schedule, which decreases the learning rate over time. The **batch size** is set to 4, which means that four images are processed in each training iteration. The training process is monitored using the **categorical accuracy metric**, which measures the proportion of correctly classified pixels.

Two callbacks are used during training: a CSV logger that records training metrics to a file, and **early stopping** that stops training if the validation categorical accuracy metric does not improve for 15 epochs.

With the network model described above, several trainings were carried out in order to identify the best configuration. In the various experimental tests, the greatest differences were found using data augmentation and with the highest possible resolution of the volumes (256x256x256).

4.6.1 Loss Function

In deep learning, the loss function is a mathematical function that measures the difference between the predicted output and the actual output. The goal of training a deep learning model is to minimize the value of the loss function, which indicates how well the model is performing on the given task.
There are many different types of loss functions used in deep learning, depending on the task at hand. In this study the loss function used is the **Categorical Cross-Entropy**.

Categorical crossentropy (CCE) is a popular loss function used in deep learning for multiclass segmentation tasks, where it is necessary to classify each voxel in

an image into one of multiple classes. In particular, the categorical crossentropy measures the dissimilarity between the predicted probability distribution and the actual probability distribution of the classes. The crossentropy function is also called **logarithmic loss, log loss** or **logistic loss**. Each predicted class probability is compared to the actual class desired output 0 or 1 and a score/loss is calculated that penalizes the probability based on how far it is from the actual expected value. The penalty is logarithmic in nature yielding a large score for large differences close to 1 and small score for small differences tending to 0. Ideally, a perfect model has a cross-entropy loss of 0 [43].

Cross-entropy is defined as:

$$L_{CE} = -\sum_{i=1}^{n} t_i \log(p_i) \tag{4.2}$$

for n classes, where t_i is the truth label and p_i is the Softmax probability for the i^{th} class. The log function is calculated to base 2, that is the same as $\ln$.
CCE is used when true labels are one-hot encoded, for example when there are the following true label for 3-class classification problem [1, 0, 0], [0, 1, 0] and [0, 0, 1] as in the cases of this study.
Finally, CEE encourages the model to assign high probabilities to the correct class and low probabilities to the incorrect ones, which is crucial for accurate segmentation [44].

4.6.2 Adam optimization algorithm

Adam optimization algorithm is a popular optimization algorithm used in deep learning for training neural networks. It stands for Adaptive Moment Estimation, and it combines the benefits of two other optimization algorithms - Stochastic Gradient Descent (SGD) and Root Mean Square Propagation (RMSprop).
This algorithm is designed to compute adaptive learning rates for each parameter based on the estimates of the first and second moments of the gradients. In simple terms, it adjusts the learning rate for each parameter based on the magnitude of the gradient, and it also takes into account the momentum of the gradients.
To use the Adam optimizer for training, we first initialize the neural network weights

and biases randomly. Then, we compute the gradients of the loss function with respect to each parameter using backpropagation. Finally, we update the parameters using the Adam update rule, which adjusts the learning rate based on the moving average of the gradient and its squared gradient.

4.6.3 Learning Rate

Using a polynomial learning rate schedule with the Adam optimizer can be a good choice for semantic multiclass segmentation in deep learning, as it can help to improve convergence, reduce overfitting, and improve the overall efficiency of the training process [45].
So we use a Polynomial Decay as learning rate schedule which is defined as:

$$lr = (start_{lr} - end_{lr})(1 - \frac{epoch - start_{epoch}}{decay_{epochs}})^{power} + end_{lr} \tag{4.3}$$

where lr stands for learning rate, $start_{lr} = 10^{-2}$, $end_{lr} = 10^{-4}$, $decay_{epochs} = training_{epochs}$ and $power = 1.5$. We train the model for 150 epochs; as we will explain later, the last epoch trained was always lower than the maximum number of preset epochs. This appends for **early stopping** conditions imposed.

4.6.4 Batch Size

Batch size refers to the number of examples or samples from a dataset that are processed together in one iteration of a deep learning algorithm. In other words, the batch size determines how many samples are fed to the model at once to update its parameters, rather than updating the model after processing each individual sample.
Larger batch sizes can speed up training by reducing the number of iterations needed to process the entire dataset, but they require more memory to store the intermediate computations. Smaller batch sizes can make training slower, but they can help the model generalize better by introducing more variability in the data seen during each iteration. The appropriate batch size depends on factors such as the size of the dataset, the complexity of the model, the available computational

resources, and the desired level of accuracy.
In our case we choose a batch size equal to 4; larger and smaller quantities have been tried but without any performance improvement.

4.6.5 Early stopping

Early stopping is a technique used in deep learning to prevent overfitting of the model to the training data. Overfitting occurs when the model learns the noise in the training data, resulting in a model that performs well on the training data but poorly on new, unseen data.
Early stopping involves monitoring the model's performance on a separate validation dataset during training. The training is stopped when the model's performance on the validation dataset stops improving or starts to worsen. This is typically done by monitoring a metric such as the validation loss or validation accuracy. By stopping the training early, the model is prevented from overfitting to the training data and generalizes better to new data. Early stopping can also save time and computational resources, as training is stopped once the model achieves good generalization performance.

In our case we use the early stopping class [46] provided by *Keras*, that is useful to stop training when a monitored metric has stopped improving. The parameters set are the following:

- **monitor** = *categorical accuracy on ValidationSet*. The "monitor" argument identified the quantity to be monitored.
- **min delta** = *0.0005*. The "min delta" argument identified the minimum change in the monitored quantity to qualify as an improvement, i.e. an absolute change of less than min delta, will count as no improvement.
- **patience** = *15*. The "patience" quantity identified the number of epochs with no improvement after which training will be stopped.
- **mode** = *max*. It can be one of "auto", "min", "max". In min mode, training will stop when the quantity monitored has stopped decreasing; in "max" mode

it will stop when the quantity monitored has stopped increasing; in "auto" mode, the direction is automatically inferred from the name of the monitored quantity.

4.6.6 CNN Performance Evaluation

The segmentation performance of the network was evaluated with the **Dice similarity coefficient (DSC)** and **Relative Volume Difference (RVD)** coefficient. They are two commonly used evaluation metrics. In our case, DSCs and RVDs are calculated on the patient level, which means that a single DSC was calculated for each segmented CBCT volume.

DSC is a statistical measure of the similarity between two sets of data, and it is widely used in the evaluation of image segmentation models. It measures the overlap between the predicted segmentation and the ground truth segmentation, and is defined as the ratio of the intersection of the two sets to their average size. A DSC score of 1 indicates perfect segmentation, while a score of 0 indicates no overlap between the two sets. DSC is a good parameter to check semantic segmentation because it considers both the false positives and false negatives in the segmentation, and is sensitive to the size of the segmented regions.
DSC coefficient is calculated as follow:

$$DSC(X,Y) = \frac{2|X \cap Y|}{|X| + |Y|} \tag{4.4}$$

where X and Y represent the manual and predicted mask respectively.

RVD, on the other hand, measures the volume difference between the predicted segmentation and the ground truth segmentation, and is defined as the ratio of the difference in volume between the two sets to the volume of the ground truth segmentation. RVD is a useful metric for semantic segmentation evaluation because it is sensitive to the differences in the size of the predicted and ground truth segmentation regions, and provides an estimate of the degree of under-segmentation or over-segmentation in the predicted segmentation.

RVD coefficient is calculated as follow:

$$RVD(X, Y) = \frac{|Y| - |X|}{|X|} \tag{4.5}$$

where X and Y represent the manual and predicted mask respectively.

In conclusion, DSC and RVD can be considered best parameters to check the quality of the segmentation because they consider both the size of the segmented regions and the degree of under-segmentation or over-segmentation in the predicted segmentation.

4.6.7 Experimental Equipment

We develop and train all our models using *Tensorflow-2.4*. The experiments have been carried out on *Google Colab Pro +*.

Chapter 5

Results

In this chapter we provide the results of the various experiments we've run, together with some comments and insights on training 3D CNNs and a few visualizations.

U-net with *resnet50* as encoder network							
volume size	Batch	training epochs	Data augmentation (on TrainingSet)	**DSC mandible**	**RVD mandible**	**DSC teeth**	**RVD teeth**
128x128x128	4	150 (83 with early stopping)	No	0.7663 ± 0.1934	-0.0574 ± 0.1286	0.6529± 0.1368	-0.1143 ± 0.2419
256x256x256	4	150 (75 with early stopping)	No	0.8513 ± 0.1199	-0.0134 ± 0.1634	0.8123± 0.1861	-0.0753 ± 0.1582
256x256x256	4	150 (111 with early stopping)	Yes **(Volumentations-3D with tf.data, +/- 15°)**	0.9059 ± 0.0273	-0.0074 ± 0.0514	0.8939 ± 0.0325	-0.0125 ± 0.0601

Figure 5.1: Metrics evaluation on TestSet

Each row of table in figure 5.1 represents a different experiment which consists of the same CNN architecture (Unet with ResNet50 as backbone and Transpose as decoder block), the batch size used, the training epochs and a flag to indicate whether we perform data augmentation or not. For each evaluation metric we compute its

average on the TestSet volumes. Because of the randomness in the training process, we repeat each experiment three times and report the average across the different runs of the same experiment together with its standard deviation.

The proper way to make comparisons between experiments would be through statistical hypothesis testing, which requires a sample large enough to make inference about its distribution; Unfortunately training Deep Neural Networks on large datasets takes a lot of time and prohibits this approach, therefore we only compare the intervals centered at the mean and with margin equal to the standard deviation.

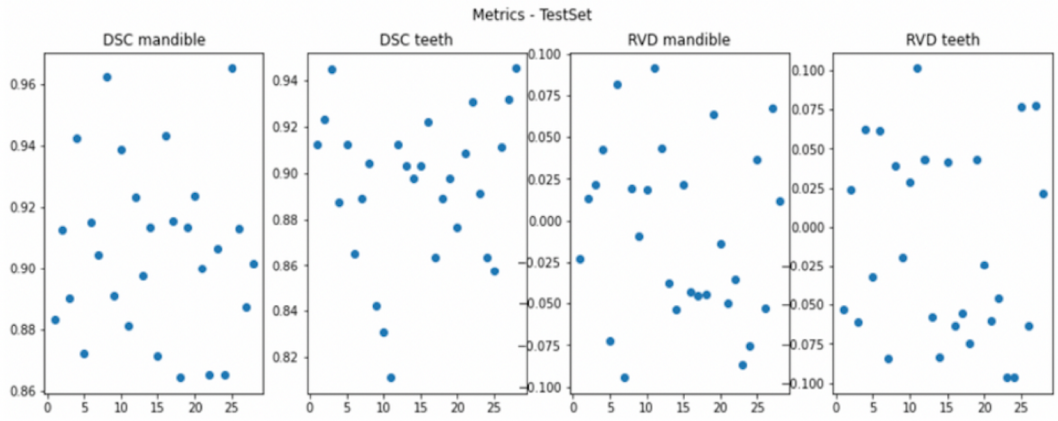


Figure 5.2: DSC and RVD for mandible and teeth

The third configuration in table 5.1 shows a large overlap between automatic and manual segmentations (DSC: 0.9059 ± 0.0273 and 0.8939 ± 0.0325, respectively for mandible and teeth; RVD: -0.0074 ± 0.0514 and -0.0125± 0.0601, respectively for mandible and teeth). For this configuration, results for each patient of TestSet are shown in figure 5.2.

In figure 5.3 manual and automatic mask obtained with input images of size 256x256x256, batch size equal to 4 and data augmentation (third configuration in table 5.1) are plotted.

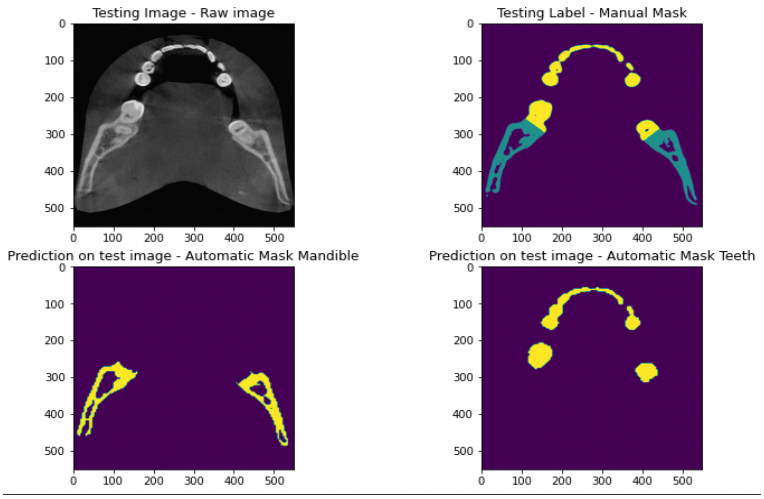


Figure 5.3: Manual and Automatic segmentation

Here we show the plots of the loss (categorical cross-entropy) and metric (categorical accuracy on ValSet) as function of the training epoch (figure 5.4) and make some comments to anticipate some questions the reader might have.

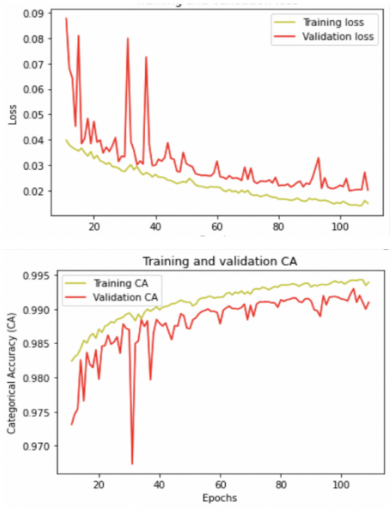
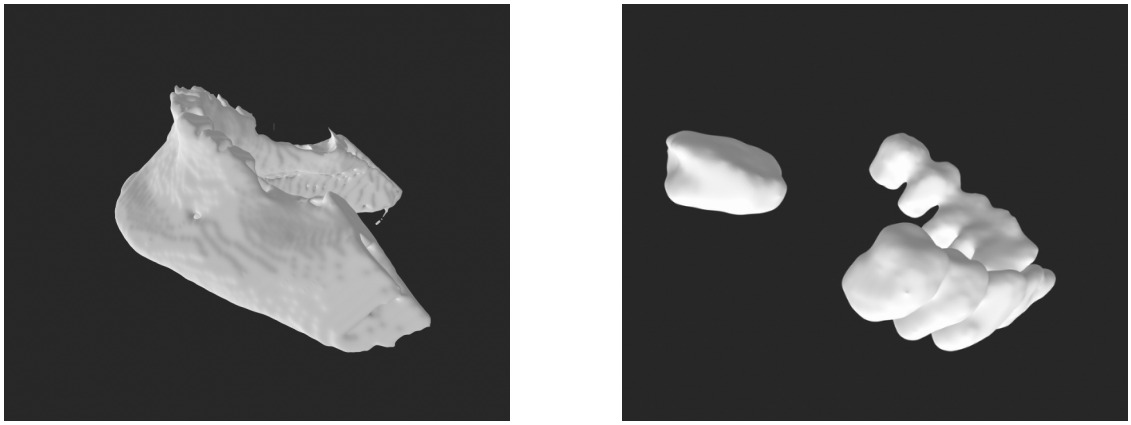


Figure 5.4: Loss and Metric plot

The loss plot shows a decreasing trend as the training progresses. This is because the goal of training is to minimize the loss function, and as the model learns to

make better predictions, the loss decreases. However, the plot shows some fluctuations due to the stochastic nature of the optimization process.
The metric plot shows an increasing trend as the training progresses. This is because the goal of training is to improve the performance of the model on the validation set, and the metric measures how well the model is doing on this set. In other words, as the model learns to make better predictions, the metric increases, indicating improved performance.

Automatic segmentations of mandible and teeth have been saved in stl (Standard Triangulation Language) format (figure 5.5) in order to use them in virtual surgery planning.



(a) (b)

Figure 5.5: 3D prediction in stl format of (a) mandible and (b) teeth

5.1 Experimental evaluation

After saving automatic mandible and teeth segmentations in the stl format, an experimental assessment was conducted. Engineers designed 40 surgical templates, (20 patients), using both manual and automatic segmentations. During the experiment, engineers were not aware of whether the segmentation was done manually or automatically.

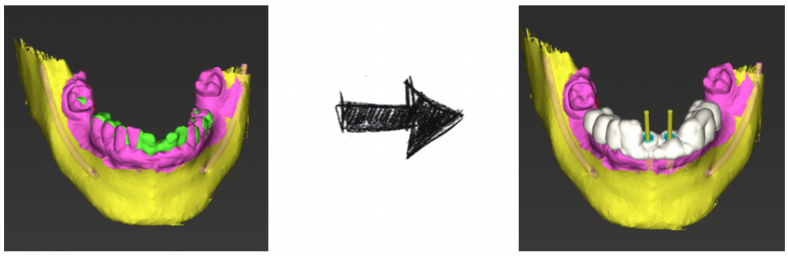


Figure 5.6: Surgical guide design with automatic segmentation

Following this, surgical guides were compared, and the deviation between the two designs (manual and automatic, for each patient) was measured using the EXOCAD software, with a maximum deviation of (348 $\pm$ 212.78) μm.

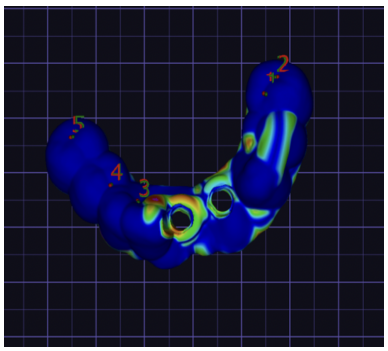


Figure 5.7: Experimental evaluation with EXOCAD

Chapter 6

Conclusion

Segmentation of mandible and teeth from CBCT requires analyzing a large amount of three-dimensional information, which cannot be fully captured by two-dimensional networks. As a result, working with 3D networks is generally considered better for this task as they can capture **spatial information** and relationships between structures, resulting in more accurate and precise segmentation. Additionally, 3D networks are better equipped to handle the varying shapes and sizes of mandibles and teeth in different patients, which can be challenging for 2D networks. Overall, using 3D networks for mandible and teeth segmentation from CBCT data is a more effective approach due to their ability to capture complex spatial relationships and accurately segment complex structures. On the other hand, there are certain drawbacks associated with working with 3D data. Firstly, Deep Learning frameworks are typically optimized for 2D images, and input pipelines that handle volumes tend to be less efficient. Secondly, image augmentation libraries are typically designed for 2D images, which means working with 3D images often requires more coding effort. Finally, working with volumes requires more computing resources, particularly in terms of **RAM** and **VRAM**.

In this thesis, 3D CNN implementation for multiclass segmentation has proven to be a valuable tool in reducing preoperative planning time for guided implant surgery. This study's findings demonstrate that the use of deep learning for mandible and

teeth segmentation is both accurate and generally usable. With an average segmentation time of only **9 seconds** per CBCT scan, this method significantly outperforms manual segmentation, which can take up to **40 minutes**. These results highlight the potential for deep learning to revolutionize preoperative planning in dentistry, improving the accuracy and efficiency of surgical procedures while also saving valuable time and resources.

Furthermore, the real case of guided implant surgery reported serves as evidence that the **alternative protocol**, in which biomedical engineers perform the surgical planning and templates are 3D-printed after being verified by clinicians, can achieve optimal outcomes that may even surpass those achieved when clinicians carry out the planning themselves, both in terms of quality and efficiency.

Bibliography

[1] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *nature* 521.7553 (2015), pp. 436–444.

[2] Arthur C Schwaninger. "The Philosophising Machine–a Specification of the Turing Test". In: *Philosophia* (2022), pp. 1–17.

[3] Marvin Minsky and Seymour A Papert. "Artificial intelligence progress report". In: (1972).

[4] Allen Newell, John C Shaw, and Herbert A Simon. "Report on a general problem solving program". In: *IFIP congress*. Vol. 256. Pittsburgh, PA. 1959, p. 64.

[5] M Arif Wani et al. "Introduction to deep learning". In: *Advances in Deep Learning*. Springer, 2020, pp. 1–11.

[6] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. "Deep sparse rectifier neural networks". In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2011, pp. 315–323.

[7] Jimmy Ba and Rich Caruana. "Do deep nets really need to be deep?" In: *Advances in neural information processing systems* 27 (2014).

[8] Ibrahem Kandel and Mauro Castelli. "The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset". In: *ICT express* 6.4 (2020), pp. 312–315.

[9] Xue Ying. "An overview of overfitting and its solutions". In: *Journal of physics: Conference series*. Vol. 1168. 2. IOP Publishing. 2019, p. 022022.

[10] Olga Russakovsky et al. “Imagenet large scale visual recognition challenge”. In: *International journal of computer vision* 115.3 (2015), pp. 211–252.

[11] Keiron O’Shea and Ryan Nash. “An introduction to convolutional neural networks”. In: *arXiv preprint arXiv:1511.08458* (2015).

[12] Ajitesh Kumar. *"CNN Basic Architecture for Classification and Segmentation"*. 2015. URL: `https://vitalflux.com/cnn-basic-architecture-for-classification-segmentation/` (visited on 2023).

[13] Yann LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Communications of the ACM* 60.6 (2017), pp. 84–90.

[15] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[16] Nikolas Adaloglouon. *"Intuitive Explanation of Skip Connections in Deep Learning"*. 2020. URL: `https://theaisummer.com/skip-connections/` (visited on 2023).

[17] Jonathan Long, Evan Shelhamer, and Trevor Darrell. “Fully convolutional networks for semantic segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.

[18] Andriy Myronenko. “3D MRI brain tumor segmentation using autoencoder regularization”. In: *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries: 4th International Workshop, BrainLes 2018, Held in Conjunction with MICCAI 2018, Granada, Spain, September 16, 2018, Revised Selected Papers, Part II 4*. Springer. 2019, pp. 311–320.

[19] Richard Van Noort. “The future of dental devices is digital”. In: *Dental materials* 28.1 (2012), pp. 3–12.

[20] Gustavo Vargas da Silva Salomão et al. “Analysis of digital workflow in Implantology”. In: *Case Reports in Dentistry* 2021 (2021).

[21] Marco Tallarico et al. “Errors in implant positioning due to lack of planning: A clinical case report of new prosthetic materials and solutions”. In: *Materials* 13.8 (2020), p. 1883.

[22] Firas Al Yafi, Brittany Camenisch, and Mohanad Al-Sabbagh. “Is digital guided implant surgery accurate and reliable?” In: *Dental Clinics* 63.3 (2019), pp. 381–397.

[23] FC Yogui et al. “Comparison between computer-guided and freehand dental implant placement surgery: A systematic review and meta-analysis”. In: *International Journal of Oral and Maxillofacial Surgery* 50.2 (2021), pp. 242–250.

[24] Yueyi Tian et al. “A review of 3D printing in dentistry: Technologies, affecting factors, and applications”. In: *Scanning* 2021 (2021).

[25] A Kessler, R Hickel, and M Reymus. “3D printing in dentistry—State of the art”. In: *Operative dentistry* 45.1 (2020), pp. 30–40.

[26] Alexey Unkovskiy et al. “Objects build orientation, positioning, and curing influence dimensional accuracy and flexural properties of stereolithographically printed resin”. In: *Dental Materials* 34.12 (2018), e324–e333.

[27] K Yamamoto et al. “Effects of mechanical stress on cytokine production in mandible-derived osteoblasts”. In: *Oral Diseases* 17.7 (2011), pp. 712–719.

[28] Percy H Carter and Ernestina Schipani. “The roles of parathyroid hormone and calcitonin in bone remodeling: prospects for novel therapeutics”. In: *Endocrine, Metabolic & Immune Disorders-Drug Targets (Formerly Current Drug Targets-Immune, Endocrine & Metabolic Disorders)* 6.1 (2006), pp. 59–76.

[29] Marthinus J Kotze et al. “A comparison of mandibular and maxillary alveolar osteogenesis over six weeks: a radiological examination”. In: *Head & Face Medicine* 10.1 (2014), pp. 1–7.

[30] Marco Mandolini et al. “Comparison of Three 3D Segmentation Software Tools for Hip Surgical Planning”. In: *Sensors* 22.14 (2022), p. 5242.

[31] Johari Yap Abdullah et al. “Comparison of 3D reconstruction of mandible for pre-operative planning using commercial and open-source software”. In: *AIP Conference Proceedings*. Vol. 1791. 1. AIP Publishing LLC. 2016, p. 020001.

[32] Ralf Schulze et al. “Artefacts in CBCT: a review”. In: *Dentomaxillofacial Radiology* 40.5 (2011), pp. 265–273.

[33] Polyane Mazucatto Queiroz et al. “Metal artifact production and reduction in CBCT with different numbers of basis images”. In: *Imaging science in dentistry* 48.1 (2018), pp. 41–44.

[34] *Keras: the Python deep learning API*. `https://keras.io/`. (visited on 2023).

[35] AKSU Gökhan, Cem Oktay Güzeller, and Mehmet Taha Eser. “The effect of the normalization method used in different sample sizes on the success of artificial neural network model”. In: *International Journal of Assessment Tools in Education* 6.2 (2019), pp. 170–192.

[36] *segmentation-models-3D · PyPI*. `https://pypi.org/project/segmentation-models-3D/`. (visited on 2023).

[37] Phillip Chlap et al. “A review of medical image data augmentation techniques for deep learning applications”. In: *Journal of Medical Imaging and Radiation Oncology* 65.5 (2021), pp. 545–563.

[38] Xinyi Li et al. “Spline smoothing of 3d geometric data”. In: *arXiv e-prints* (2021), arXiv–2106.

[39] *Multidimensional image processing (scipy.ndimage) — SciPy v1.10.1 Manual*. `https://docs.scipy.org/doc/scipy/reference/ndimage.html`. (visited on 2023).

[40] *scikit-image 0.19.2 docs — skimage v0.19.2 docs*. `https://scikit-image.org/docs/stable/index.html`. (visited on 2023).

[41] *meshlib · PyPI*. `https://pypi.org/project/meshlib/`. (visited on 2023).

[42] *segmentation-models-3D · PyPI*. `https://pypi.org/project/segmentation-models-3D/`. (Accessed on 2023).

[43] Zhilu Zhang and Mert Sabuncu. "Generalized cross entropy loss for training deep neural networks with noisy labels". In: *Advances in neural information processing systems* 31 (2018).

[44] *Cross-Entropy Loss Function. A loss function used in most... | by Kiprono Elijah Koech | Towards Data Science*. https://towardsdatascience.com/cross-entropy-loss-function-f38c4ec8643e. (Accessed on 2023).

[45] Purnendu Mishra and Kishor Sarawadekar. "Polynomial learning rate policy with warm restart for deep neural network". In: *TENCON 2019-2019 IEEE Region 10 Conference (TENCON)*. IEEE. 2019, pp. 2087–2092.

[46] *EarlyStopping*. https://keras.io/api/callbacks/early_stopping/. (Accessed on 2023).

[47] Sercan Sabancı et al. "Is Manual Segmentation the Real Gold Standard for Tooth Segmentation? A Preliminary in vivo Study Using Cone-beam Computed Tomography Images". In: *Meandros Medical and Dental Journal* 22.3 (2021), pp. 263–273.

[48] Chen Sun et al. "Revisiting unreasonable effectiveness of data in deep learning era". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 843–852.