

1 ALMA MATER STUDIORUM · UNIVERSITY OF BOLOGNA
2

3

4 School of Science
5 Department of Physics and Astronomy
6 Master Degree in Physics

7

Dataset Generation for the Training of 8 Neural Networks Oriented toward 9 Histological Image Segmentation

10

Supervisor:
Dr. Enrico Giampieri

Co-supervisor:
Dr. Nico Curti

11

Submitted by:
Alessandro d'Agostino

12

Academic Year 2019/2020

¹⁴ **Abstract**

¹⁵ The project is inspired by an actual problem of timing and accessibility in the analysis
¹⁶ of histological samples in the health-care system. In this project, I face the problem
¹⁷ of synthetic histological image generation for the purpose of training Neural Networks
¹⁸ for the segmentation of real histological images. The method I propose is based on the
¹⁹ replication of the traditional specimen preparation technique in a virtual environment.
²⁰ The first step is the creation of a 3D virtual model of a region of the target human
²¹ tissue. The model should encapture all the key features of the tissue, and the richer it
²² is the better will be the yielded result. The second step is to perform a sampling of the
²³ model through a virtual sectioning process, which produces a first image of the section,
²⁴ which will act as a segmentation mask. This image is then processed with different tools
²⁵ to achieve a histological-like aspect. The most significant contribution is given by the
²⁶ action of a style transfer neural network that implants the typical visual texture of a
²⁷ histological sample onto the synthetic image. This procedure is presented in detail for
²⁸ two specific models of human tissue: one of pancreatic tissue and one of dermal tissue.
²⁹ The two resulting images compose a pair of images and corresponding ground-truth
³⁰ image, which is perfectly suitable for a supervised learning technique. The generation
³¹ process is completely automatized and does not require the intervention of any human
³² operator, hence it can be used to produce arbitrary large datasets. The synthetic images
³³ are inevitably less complex than the real samples and they offer an easier segmentation
³⁴ task to solve for the NN. However, the synthetic images are very abundant, and the
³⁵ training of a NN can take advantage of this feature, following the so-called curriculum
³⁶ learning strategy.

³⁷ Table of Contents

³⁸	Introduction	6
³⁹	1 Theoretical Background	12
⁴⁰	1.1 Histological Images	12
⁴¹	1.1.1 Slides Preparation for Optic Microscopic Observation	14
⁴²	1.2 Introduction to Deep Learning	17
⁴³	1.2.1 Perceptrons and Multilayer Feedforward Architecture	17
⁴⁴	1.2.2 Training of a NN - Error Back-Propagation	19
⁴⁵	1.3 Deep Learning-Based Segmentation Algorithms	25
⁴⁶	1.3.1 State of the Art on Deep Learning Segmentation	26
⁴⁷	1.3.2 Image Segmentation Datasets	31
⁴⁸	2 Technical Tools for Model Development	34
⁴⁹	2.1 Quaternions	34
⁵⁰	2.2 Parametric L-Systems	36
⁵¹	2.3 Voronoi Tassellation	38
⁵²	2.4 Saltelli Algorithm - Randon Number Generation	40
⁵³	2.5 Planar Section of a Polyhedron	42
⁵⁴	2.6 Perlin Noise	44
⁵⁵	2.7 Style-Transfer Neural Network	45
⁵⁶	2.8 Working Environment	48
⁵⁷	3 Tissue Models Development	50
⁵⁸	3.1 Pancreatic Tissue Model	50
⁵⁹	3.1.1 2D Ramification	51
⁶⁰	3.1.2 Expansion to 3D	52
⁶¹	3.1.3 Subdivision in Cells	54

62	3.1.4 Cells Identity Assignment	56
63	3.2 Dermal Tissue Model	57
64	3.3 Synthetic Images Production	61
65	3.3.1 Sectioning Process	61
66	3.3.2 Appearance Makeover	63
67	Conclusions	70
68	Bibliography	73

⁷⁰ Introduction

⁷¹ In the last decades, the development of Machine Learning (ML) and Deep Learning (DL)
⁷² techniques has contaminated every aspect of the scientific world, with interesting results
⁷³ in many different research fields. The biomedical field is no exception to this and a
⁷⁴ lot of promising applications are taking form, especially as Computer-Aided Detection
⁷⁵ (CAD) systems which are tools for the support for physicians during the diagnostic
⁷⁶ process. Medical doctors and the healthcare system in general collect a huge amount of
⁷⁷ data from patients during all the treatment, screening, and analysis activities in many
⁷⁸ different shapes, from anographical data to blood analysis to clinical images.

⁷⁹ In fact in medicine, the study of images is ubiquitous and countless diagnostic proce-
⁸⁰ dures rely on it, such as X-ray imaging (CAT), nuclear imaging (SPECT, PET), Magnetic
⁸¹ resonance, and visual inspection of histological specimens after biopsies. The branch of
⁸² artificial intelligence in the biomedical field that handles image analysis to assist physi-
⁸³ cians in their clinical decisions goes under the name of Digital Pathology Image Analysis
⁸⁴ (DPIA). In this thesis work, I want to focus on some of the beneficial aspects intro-
⁸⁵ duced by DPIA in the histological images analysis and some particular issues in the
⁸⁶ development of DL models able to handle this kind of procedure.

⁸⁷ Nowadays the great majority of analysis of histological specimens occurs through
⁸⁸ visual inspection, carried out by highly qualified experts. Some analysis, as cancer
⁸⁹ detection, requires the ability to distinguish if a region of tissue is healthy or not with high
⁹⁰ precision in very wide specimens. This kind of procedure is typically very complex and
⁹¹ requires prolonged times of analysis besides substantial economic efforts. Furthermore,
⁹² the designated personnel for this type of analysis is often limited, leading to delicate issues
⁹³ of priority assignment while scheduling analysis, based on the estimated patient's clinical
⁹⁴ development. Some sort of support to this analysis procedure is therefore necessary.

⁹⁵ The problem of recognizing regions with different features within an image and de-
⁹⁶ tect their borders is known in computer vision as the segmentation task, and it's quite
⁹⁷ widespread with countless different applications, allowing a sort of automatic image in-
⁹⁸ terpretation. In ML the segmentation problem is usually faced as a supervised task,
⁹⁹ hence the algorithm in order to be trained properly requires an appropriate quantity
¹⁰⁰ of pre-labeled images, from which learn the rules through which distinguish different
regions. This means that the development of segmentation algorithms for a specific ap-

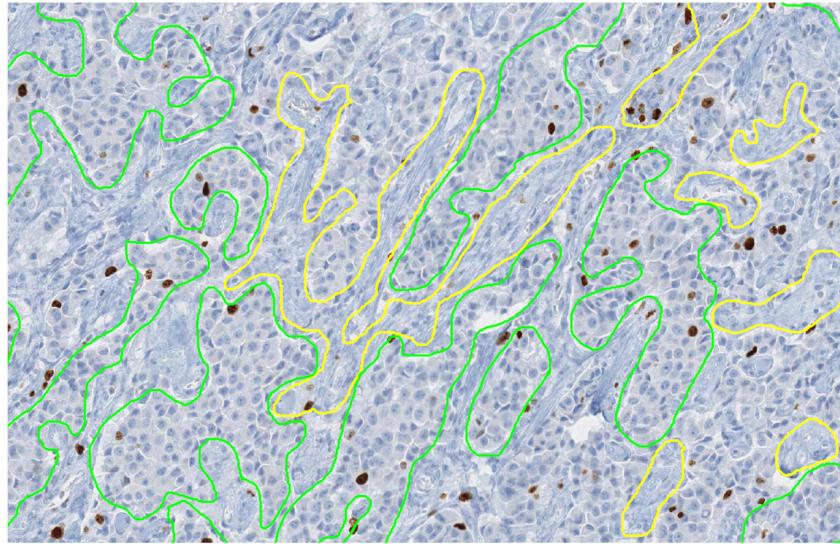


Figure 1: Interleaving of tumor (green annotation) and non-tumor (yellow annotation) regions [28].

102 plication, as would be the one on histological images, would require a lot of starting
 103 material, previously analyzed from the same qualified expert encharged of the visual
 104 inspection mentioned before. A human operator thus is required to manually track the
 105 boundaries, for example, between healthy and tumoral regions within a sample of tissue
 106 and to label them with their identity, as in Figure 1. The more the algorithm to train
 107 is complex the more starting material is required to adjust the model’s parameters and
 108 reach the desired efficacy.

109 The latest developed segmentation algorithms are based on DL techniques, hence
 110 based on the implementation of intricated Neural Networks (NN) which process the
 111 input images and produce the corresponding segmentation. Those models are typically
 112 very complex, with millions of parameters to adjust and tune, therefore they need a
 113 huge amount of pre-labeled images to learn their segmentation rules. This need for data
 114 is exactly the main focus of my thesis work. The shortage of ground truth images is
 115 indeed one of the toughest hurdles to overcome during the development of DL-based
 116 algorithms. Another important aspect to bear in mind is the quality of the ground truth
 117 material. It’s impossible for humans to label boundaries of different regions with pixel-
 118 perfect precision, while for machines the more precise is the input the more effective is
 119 the resulting algorithm.

120 Different approaches have already been explored to overcome this problem, and they
 121 are mainly based on the generation of synthetic labeled data to use during the training
 122 phase. Some techniques achieve data augmentation manipulating already available
 123 images and then generating *new* images, but as we will see this approach suffers from

124 different issues. Here, I want to make an overview of some other interesting works on
125 the generation of synthetic histological images, which have followed completely different
126 paths and strategies from mine.

127 The first work I want to cite is a work from Ben Cheikh *et al.* from 2017 [9]. In
128 this work, they present a methodology for the generation of synthetic images of different
129 types of breast carcinomas. They propose a method completely based on two-dimensional
130 morphology operation, as successive image dilations and erosions. With the modulation
131 of a very restricted number of parameters, regulating the abundance of the objects, their
132 distribution in the image, and their shapes they are able to reconstruct realistic images
133 reflecting different histological situations. In Figure 2 is shown an example of generated
134 material besides a real histological H&E stained sample. Starting from a generated
135 segmentation mask which defines the *tumoral pattern* the production of the synthetic
136 image passes through successive steps, as the generation of characteristic collagen fibers
137 around the structure, the injection of all the immune system cells, and some general final
138 refinements.

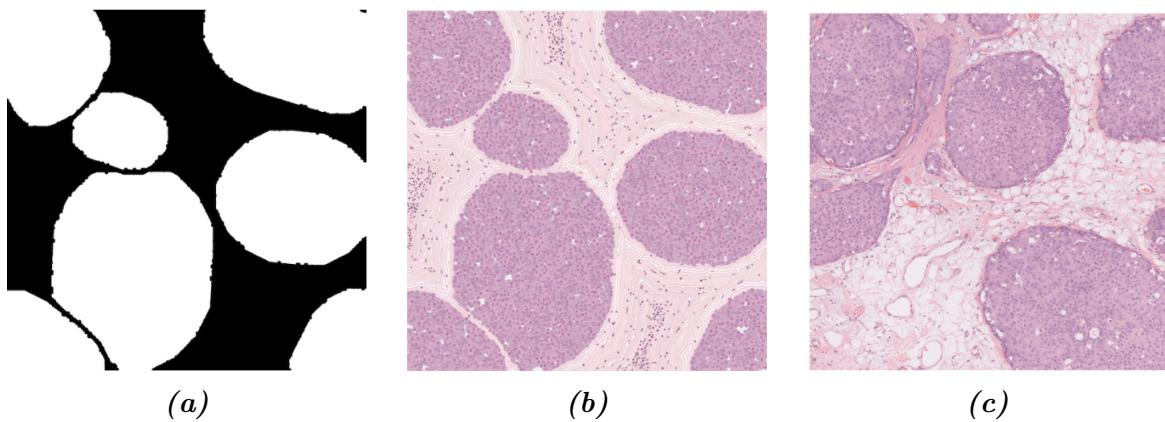


Figure 2: Example of generated tumoral pattern (left), which acts as segmentation mask, of generated image (center) and a real example of the tissue to recreate, from [9].

139 The second work I want to mention is based on a DL-base technique, which ap-
140 proaches synthetic image generation using a specific cGAN architecture inspired to the
141 “U-net” [38] model, as will be described in Figure 1.13 in section 1.3.1. This model
142 works with Ki67 stained samples of breast cancer tissue, and it is able to generate high-
143 fidelity images starting from a given segmentation mask. Those starting segmentation
144 masks tough are obtained through the processing of other real histological samples, via
145 a nuclei-detection algorithm. The differences between real and synthetic samples are
146 imperceptible, and the material generated in this work has effectively fooled experts,
147 who qualified it has indistinguishable from the real one. In Figure 3 an example of a real
148 image, a generated one, and their corresponding segmentation mask.

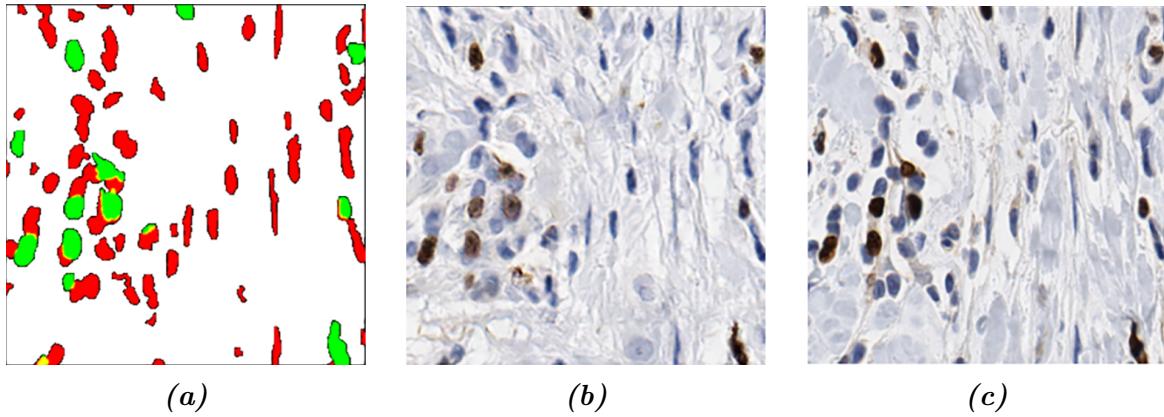


Figure 3: Example of generated tumoral pattern (left), which acts as segmentation mask, of generated image (center) and a real example of the tissue to recreate, from [38].

Both of the two before-mentioned strategies produces realistic (or even perfect) results, but they are based onto considerations and analysis limited only to the aspect of the images. In the first work, the segmentation mask is produced in an almost full-random way, while in the second the segmentation mask is extracted starting from an actual real histological samples. The target of the present work instead lies in between those two approaches, and wants to produce randomized new images following a plausible modelization based on physical and histological considerations.

The technique I propose in this work follows a generation from scratch of entire datasets suitable for the training of new algorithms, based on the 3D modelization of a region of human tissue at the cellular level. The entire traditional sectioning process, which is made on real histological samples, is recreated virtually on this virtual model. This yields pairs of synthetic images with their corresponding ground-truth. Using this technique one would be able to collect sufficient material for the training (the entire phase or the preliminary part) of a model, overcoming the shortage of hand-labeled data. The 3D modeling of a region of particular human tissue is a very complex task, and it is almost impossible to capture all the physiological richness of a histological system. The models I implemented thus are inevitably less sophisticated respect the target biological structures. I'll show two models: one of pancreatic tissue and another of dermal tissue, besides all the tools I used and the choices I made during the designing phase.

In order to present organically all the steps of my work the thesis is organized in chapters as follows:

1. Theoretical Background .

In this chapter, I will describe how real histological images are obtained and their digitalization process works. Afterward, I will introduce the reader to the Deep

173 Learning framework, explaining the key elements of this discipline and how they
174 work. Finally, I will dedicate a section to the image segmentation problem, and the
175 state of the art of segmentation DL-based algorithms, with particular attention to
176 the applications in the bio-medical field.

177 **2. Technical Tools for Model Development .**

178 I will dedicate this chapter to the thorough description of every technical tool I
179 needed during the designing phase of this project. The development has required
180 the harmonization of many different technologies and mathematical tools, some of
181 which not so popular. In this chapter, I will also describe my working environment.

182 **3. Tissue Models Development .**

183 This third chapter is the heart of the project. I will describe in detail all the
184 steps necessary to create the two models, one of pancreatic tissue and the other
185 of dermal tissue, and how I am able to produce the synthetic images. The first
186 section is devoted to the modeling of the histological structures, while the second
187 in entirely dedicated to the sectioning process and the subsequent refinements to
188 the images.

¹⁹⁰ **Chapter 1**

¹⁹¹ **Theoretical Background**

¹⁹² In this first chapter, I will depict the theoretical context of the work. Section 1.1 will
¹⁹³ be dedicated to histological images, and the different techniques used to prepare the
¹⁹⁴ samples to analyze. Histological images recover a fundamental role in medicine and are
¹⁹⁵ the pillar of many diagnosis techniques. This discipline borns traditionally from the
¹⁹⁶ optical inspection of the tissue slides using a microscope, and it is gradually developing
¹⁹⁷ and improving with the advent of computers and digital image processing. It is important
¹⁹⁸ tough to understand how the samples are physically prepared, the final target of this
¹⁹⁹ work is in fact the virtual reconstruction of this process. In section 1.2 I will introduce
²⁰⁰ the Deep Learning framework and describe how a Neural Network works and actually
²⁰¹ learns. The most advanced techniques for the automatic image processing implement
²⁰² Deep Learning algorithm, and understanding the general rules behind this discipline is
²⁰³ crucial for a good comprehension of this work. In section 1.3 I will discuss in particular
²⁰⁴ the problem of image segmentation and how it is tackled with different Neural Network
²⁰⁵ architectures, showing what it is the state of the art of this research field.

²⁰⁶ **1.1 Histological Images**

²⁰⁷ Modern histopathology is essentially based on the careful interpretation of microscopic
²⁰⁸ images, with the intention of correctly diagnose patients and to guide therapeutic
²⁰⁹ decisions. In the last years, thanks to the quick development of scanning techniques and
²¹⁰ image processing, the discipline of histology have seen radical improvements: the main
²¹¹ of which undoubtedly is the passage from the microscope's oculars to the computer's
²¹² screen. This digitalization process has brought several advantages, that were previously
²¹³ impossible in classical histology, like telepathology and remote assistance in diagnosis
²¹⁴ processes, the integration with other digitalized clinical workflows, and patients' history,
²¹⁵ and most importantly the opening to applications of artificial intelligence. The name
²¹⁶ Whole Slide Imaging (WSI) refers to the modern virtual microscopy discipline, which

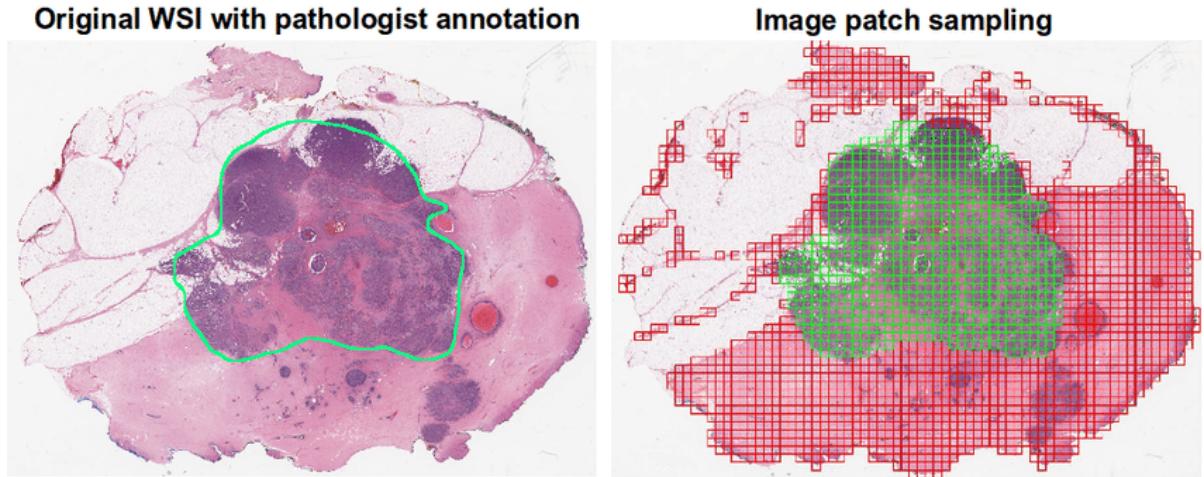


Figure 1.1: An example of whole slide image, with its grid decomposition in patches. It is visible the correspondence between a region of interest manually annotated and the patches that matches that region. From [13]

217 consists of scanning a complete microscope slide and creating a single high-resolution
 218 digital file. This is commonly achieved by capturing many small high-resolution image
 219 tiles or strips and then montaging them to create a full image of a histological section.
 220 The four key steps of this process are image acquisition (scansion), editing, and on-screen
 221 image visualization.

222 In the field of Digital Pathology (DP) an essential concept in image understanding
 223 is the magnification factor, which indicates the scale of representation of the image and
 224 allows dimension referencing. This factor is usually indicated as the magnification power
 225 of the microscope's lenses used during the analysis. After the digitalization process,
 226 this original magnification factor is prone to change, depending on the resolution of the
 227 visualization screen. Therefore, image resolution is measured in μm per pixel, and it is set
 228 by the different composition of the acquisition chain, as the optical sensor and the lenses.
 229 Histological scanner are usually equipped with $20\times$ or $40\times$ objectives, which correspond
 230 to 0.5 and 0.25 mm/pixels resolution values. Lenses with $20\times$ magnification factor are
 231 the most suitable for the great majority of histopathological evaluations, and it is the
 232 golden standard for scansions, for its good trade-off between image quality and time of
 233 acquisition. Scansions with $40\times$ magnification could increase four-fold acquisition and
 234 processing time, final file's dimension, and storage cost. A single WSI image, acquired
 235 with $20\times$ will occupy more than 600 MB alone.

236 Despite the WSI is a relatively mature discipline, it still struggles to integrate itself in
 237 the standard primitive diagnosis phase in histopathological laboratories. This is primar-
 238 ily due to some disadvantages, like images' resolution, image compression's artifacts, and
 239 auto-focusing algorithms, which plays a key role in the specimen interpretation. Fur-

thermore, the scansion of histological samples is an additional step in the analysis which takes time. Despite the technological improvements the average time for the acquisition of a sample is around 5/10 minutes, depending on the number of slices in the slide, for just a single level of magnification. While in traditional histology, the pathologist has access to all the magnification levels at the same time. The real advantage, in fact, is in the long term. Once the images have been acquired they can be archived and consulted remotely almost instantaneously, helping clinical analysis and allowing remote assistance (telemedicine). Furthermore, the images now can be processed by artificial intelligence algorithms, allowing the application of technologies like Deep Learning which could revolutionize the research field, as already has been on many different disciplines in the scientific world.

In order to allow to automatically process, such big images as the ones obtained through WSI, it is necessary to subdivide them in smaller patches. The dimension of which should be big enough to allow interpretation and to preserve a certain degree of representability of the original image. In Figure 1.1 is shown an example of whole slide image, with its grid decomposition in patches. If the patches are too small, it should be over-specified for a particular region of tissue, loosing its general features. This could lead the learning algorithm to misinterpretation. However, this is not an exclusive limit of digital pathology, for a human pathologist would be impossible too to make solid decisions on a too limited sample of tissue. After the subdivision in patches, a typical process for biomedical images is the so-called *data augmentation* of images, that is the process of creating re-newed images from the starting material through simple geometrical transformations, like translation, rotation, reflection, zoom in/out.

The analysis of histological images usually consists in detecting the different components in the samples and to recognise their arrangement as an healthy or pathological pattern. It is necessary to recognize every sign of vitality of the cells, evaluating the state of the nucleus. There are many additional indicators to consider like the presence of inflammatory cells or tumoral cells. Furthermore, samples taken from different parts of the human body present completely different characteristics, and this increase greatly the complexity of the analysis. A reliable examination of a sample thus require a careful inspection made by an high qualified expert. The automatization of this procedure would be extremely helpful, giving an increadible boost both in timing and in accessibility. However, this is not a simple task and in section 1.3.1 I will show some actual model for biomedical image processing in detail.

1.1.1 Slides Preparation for Optic Microscopic Observation

In modern, as in traditional, histology regardless on the final support of the image the slide has to be physically prepared, starting from the sample of tissue. The sample and slide preparation is a crucial step for histological or cytological observation. It is essential to highlight what needs to be observed and to *immobilize* the sample at a particular point

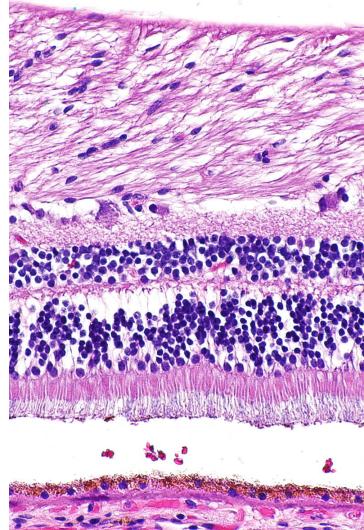


Figure 1.2: A sample of tissue from a retina (a part of the eye) stained with hematoxylin and eosin, cell nuclei stained blue-purple and extracellular material stained pink.

279 in time and with characteristics close to those of its living state. There are five key steps
280 for the preparation of samples [4]:

- 281 1) **Fixation** is carried out immediately after the removal of the sample to be observed.
282 It is used to immobilize and preserve the sample permanently in as life-like state
283 as possible. It can be performed immersing the biological material in a formalin
284 solution or by freezing, so immersing the sample in a tissue freezing medium which
285 is then cooled in liquid nitrogen.
- 286 2) **Embedding** if the sample has been stabilized in a fixative solution, this is the sub-
287 sequent step. It consists in hardening the sample in a paraffin embedding medium,
288 in order to be able to carry out the sectioning. It is necessary to dehydrate the
289 sample beforehand, by replacing the water molecules in the sample with ethanol.
- 290 3) **Sectioning** Sectioning is performed using microtomy or cryotomy. Sectioning is an
291 important step for the preparation of slides as it ensures a proper observation of the
292 sample by microscopy. Paraffin-embedded samples are cut by cross section, using a
293 microtome, into thin slices of $5\ \mu m$. Frozen samples are cut using a cryostat. The
294 frozen sections are then placed on a glass slide for storage at $-80^{\circ}C$. The choice of
295 these preparation conditions is crucial in order to minimize the artifacts. Paraffin
296 embedding is favored for preserving tissues; freezing is more suitable for preserving
297 DNA and RNA and for the labeling of water-soluble elements or of those sensitive
298 to the fixation medium.

299 **4) Staining** Staining increases contrasts in order to recognize and differentiate the dif-
300 ferent components of the biological material. The sample is first deparaffinized and
301 rehydrated so that polar dyes can impregnate the tissues. The different dyes can
302 thus interact with the components to be stained according to their affinities. Once
303 staining is completed, the slide is rinsed and dehydrated for the mounting step.

304 Hematoxylin and eosin stain (H&E) is one of the principal tissue stains used in
305 histology [44], and it is the most widely used stain in medical diagnosis and is often
306 the gold standard [35]. H&E is the combination of two histological stains: hematoxylin
307 and eosin. The hematoxylin stains cell nuclei blue, and eosin stains the extracellular
308 matrix and cytoplasm pink, with other structures taking on different shades, hues, and
309 combinations of these colors. An example of H&E stained is shown in Figure 1.2, in
310 which we can see the typical colour palette of an histological specimen.

311 1.2 Introduction to Deep Learning

312 Deep Learning is part of the broader framework of Machine Learning and Artificial
313 Intelligence. Indeed all the problems typically faced using ML can also be addressed
314 with DL techniques, for instance, regression, classification, clustering, and segmentation
315 problems. We can think of DL as a universal methodology for iterative function ap-
316 proximation with a great level of complexity. In the last decades, this technology has
317 seen a frenetic diffusion and an incredible development, thanks to the always increasing
318 available computational power, and it has become a staple tool in all sorts of scientific
319 applications.

320 1.2.1 Perceptrons and Multilayer Feedforward Architecture

321 Like other artificial learning techniques, DL models aim to *learn* a relationship between
322 some sort of input and a specific kind of output. In other words, approximating nu-
323 merically the function that processes the input data and produces the desired response.
324 For example, one could be interested in clustering data in a multidimensional features
325 space, or the detection of objects in a picture, or text manipulation/generation. The
326 function is approximated employing a greatly complex network of simple linear and non-
327 linear mathematical operations arranged in a so-called Neural Network (typically with
328 millions of parameters). The seed idea behind this discipline is to recreate the function-
329 ing of actual neurons in the human brain: their entangled connection system and their
330 “ON/OFF” behavior [41].

331 The fundamental unit of a neural network is called perceptron, and it acts as a digital
332 counterpart of a human neuron. As shown in Figure 1.3 a perceptron collects in input a
333 series on n numerical signals $\vec{x} = 1, x_1, \dots, x_n$ and computes a linear weighted combination
334 with the weights vectors $\vec{w} = w_0, w_1, \dots, w_n$, where w_0 is a bias factor:

$$f(\vec{x}, \vec{w}) = \chi(\vec{x} \cdot \vec{w}). \quad (1.1)$$

335 The results of this linear combination are given as input to a non-linear function $\chi(x)$
336 called the activation function. Typical choices as activation function are any sigmoidal
337 function like $\text{sign}(x)$ and $\tanh(x)$, but in more advanced applications other functions
338 like ReLU [1] are used. The resulting function $f(\vec{x}, \vec{w})$ has then a simple non linear
339 behaviour. It produces a binary output: 1 if the weighted combination is high enough
340 and 0 if it is low enough, with a smooth modulation in-between the two values.

341 The most common architecture for a NN is the so-called *feed-forward* architecture,
342 where many individual perceptrons are arranged in chained layers, which take as input
343 the output of previous layers along with a straight information flux. More complex ar-
344 chitectures could implement also recursive connection, linking a layer to itself, but it
345 should be regarded as sophistication to the standard case. There are endless possibilities

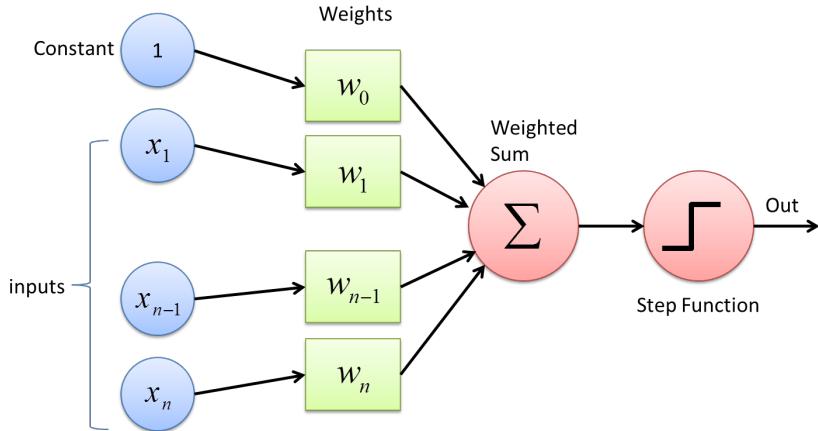


Figure 1.3: Schematic picture of a single layer perceptron. The input vector is linearly combined with the bias factor and sent to an activation function to produce the numerical "binary" output.

of combination and arrangement of neurons inside a NN's layer, but the most simple ones are known as *fully-connected* layers, where every neuron is linked with each other neuron of the following layer, as shown in Figure 1.4. Each connection has its weight, which contributes to modulate the overall combination of signals. The training of a NN consists then in the adjustment and fine-tuning of all the network's weights and parameters through iterative techniques until the desired precision in the output generation is reached.

Although a fully connected network represents the simplest linking choice, the insertion of each weight increases the number of overall parameters, and so the complexity of the model. Thus we want to create links between neurons smartly, rejecting the less useful ones. Depending on the type of data under analysis there are many different established typologies of layers. For example, in the image processing field, the most common choice is the convolutional layer, which implements a sort of discrete convolution on the input data, as shown in Figure 1.5. While processing images, the convolution operation confers to the perception of correlation between adjacent pixels of an image and their color channels, allowing a sort of spatial awareness. Furthermore, the majority of traditional computer vision techniques are based on the discrete convolution of images, and on the features extracted from them.

As a matter of principle a NN with just two successive layers, which is called a *shallow* network, and with an arbitrary number of neurons per layer, can approximate arbitrary well any kind of smooth enough function [32]. However, direct experience suggests that networks with multiple layers, called *deep* networks, can reach equivalent results exploiting a lower number of parameters overall. This is the reason why this discipline goes under the name of *deep* learning: it focuses on deep networks with up

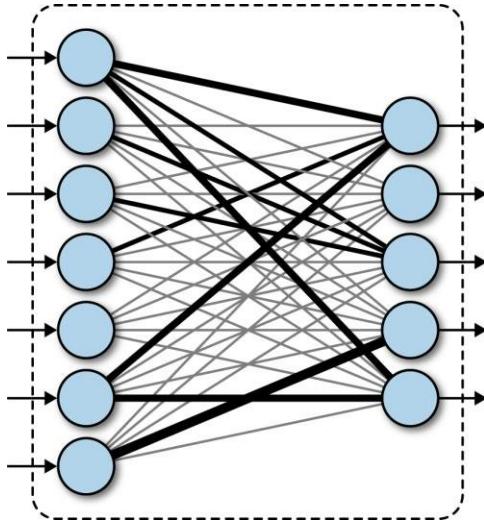


Figure 1.4: Schematic representation of a fully connected (or dense) layer. Every neuron from the first layer is connected with every output neuron. The link thickness represent the absolute value of the combination weight for that particular value.

370 to tens of hidden layers. Such deep structures allow the computation of what is called
 371 deep features, so features of the features of the input data, that allows the network to
 372 easily manage concepts that would be barely understandable for humans.

373 1.2.2 Training of a NN - Error Back-Propagation

374 Depending on the task the NN is designed for, it will have a different architecture and
 375 number of parameters. Those parameters are initialized to completely random values,
 376 tough. The training process is exactly the process of seeking iteratively the right values
 377 to assign to each parameter in the network in order to accomplish the task. The best
 378 start to understanding the training procedure is to look at how a supervised problem is
 379 solved. In supervised problems, we start with a series of examples of true connections
 380 between inputs and correspondent outputs and we try to generalize the rule behind those
 381 examples. After the rule has been picked up the final aim is to exploit it and to apply
 382 it to unknown data, so the new problem could be solved. In opposition to the concept
 383 of supervised problems, there are the *unsupervised* problems, where the algorithm does
 384 not try to learn a rule from a practical example but try to devise it from scratch. A task
 385 typically posed as unsupervised is clustering, when different data are separated in groups
 386 based on the values of their features in the feature space. Usually, only the number of
 387 groups is taken in input from the algorithm, and the subdivision is completely performed
 388 by the machine. In the real world, by the way, there are many different and creative

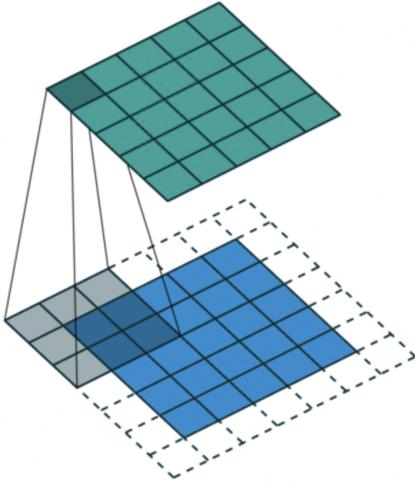


Figure 1.5: Schematic representation of a convolutional layer. The input data are processed by a window kernel that slides all over the image. This operation can recreate almost all the traditional computer vision techniques, and can overcome them, creating new operations, which would be unthinkable to hand-engineered.

389 shapes between pure supervised and pure unsupervised learning, based on the actual
 390 availability of data and specific limitations to the individual task.

391 An interesting mention in this regards should be made about *semi-supervised* learn-
 392 ing, which is typically used in bio-medical applications. This learning technique combines
 393 a large quantity of unlabeled data during training with a limited number of pre-labeled
 394 example. The blend between data can produce considerable improvement in learning
 395 accuracy, and leading to better results respect to pure unsupervised techniques. The
 396 typical situation of usage of this technique is when the acquisition of labeled data re-
 397 quires a highly trained human agent (as an anatomo-pathologist) or a complex physical
 398 experiment. The actual cost of building entire and suitable fully labeled training sets
 399 in these situations would be unbearable, and semi-supervised learnign comes in great
 400 practical help.

401 Another important training technique which worth mentioning is the so-called *cur-
 402 riculum learning* [5]. This is a learning technique inspired by the typical learning curve
 403 human being and animal, that are used to face problems of always increasing difficulty
 404 while learning something new or a new skill. The example presented to the Neural Net-
 405 work are not randomly presented but organized in a meaningful order which illustrates
 406 gradually more concepts, and gradually more complex ones. The experiments show that
 407 through curriculum learning significant improvements in generalization can be achieved.
 408 This approach has both an effect on the speed of convergence of the training process to a

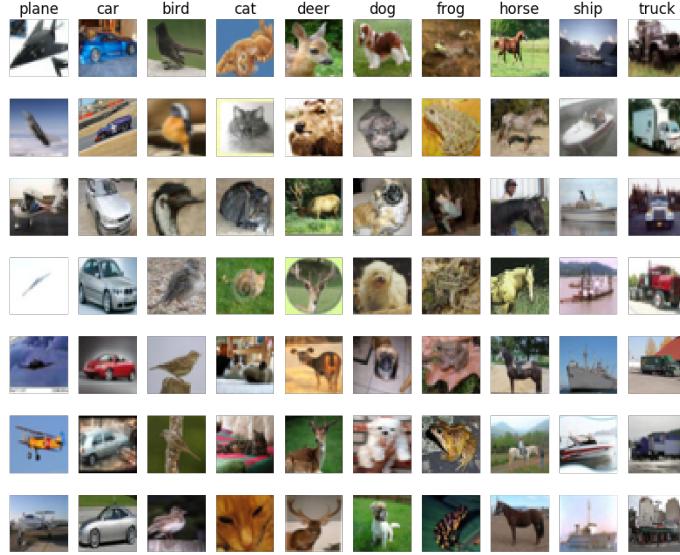


Figure 1.6: Sample grid of images from the CIFAR10 dataset. Each one of the 32×32 image is labeled with one of the ten classes of objects: plane, car, bird, cat, deer, dog, frog, horse, ship, truck.

409 minimum and, in the case of non-convex criteria, on the quality of the local minima ob-
 410 tained. This technique is of particular interest for this work: the generated images, which
 411 will be shown in section 3.3 , will offer a segmentation task much less complex respect to
 412 the analysis of real histological tissue. In the optical of training a DL-based model the
 413 abundantly produced images can be used for the preliminary phase of training, setting
 414 aside the more complex and more valuable hand-labeled images for the finalization of
 415 the training process.

416 A good example of supervised problems tough is the classification of images. Let's
 417 assume we have a whole dataset of pictures of different objects (as cats, dogs, cars, etc.)
 418 like the CIFAR10 [22] dataset. This famous dataset is made of over $60K$ labeled colored
 419 images 32×32 divided into 10 categories of objects as shown in Figure 1.6. We could be
 420 interested in the creation of a NN able to assign at every image its belonging class. This
 421 NN could be arbitrarily complex but it certainly will take as input a $32 \times 32 \times 3$ RGB
 422 image and the output will be the predicted class. A typical output for this problem
 423 would be a probability distribution over all the 10 classes like:

$$\vec{p} = (p_1, p_2, \dots, p_{10}), \quad (1.2)$$

$$\sum_{i=1}^{10} p_i = 1, \quad (1.3)$$

424 and it should be compared with the true label, that is represented just as a binary
 425 sequence \vec{t} with the bit correspondent to the belonging class set as 1, and all the others
 426 value set to 0:

$$\vec{t} = (0, 0, \dots, 1, \dots, 0, 0). \quad (1.4)$$

427 Every time an image is given to the model an estimate of the output is produced.
 428 Thus, we need to measure the *distance* between that prediction and the true value, to
 429 quantify the error made by the algorithm and try to improve the model's predictive
 430 power. The functions used for this purpose are called loss functions. The most common
 431 choice is the Mean Squared Error (MSE) function that is simply the averaged L^2 norm
 432 of the difference vector between \vec{p} and \vec{t} :

$$MSE = \frac{1}{n} \sum_{i=0}^n (t_i - p_i)^2. \quad (1.5)$$

433 Let's say the NN under training has L consecutive layers, each one with its activation
 434 function f^k and its weights vector \vec{w}^k , hence the prediction vector \vec{p} could be seen as the
 435 result of the consecutive, nested, application through all the layers:

$$\vec{p} = f^L(\vec{w}^L \cdot (f^{L-1}(\vec{w}^{L-1} \cdot \dots \cdot f^1(\vec{w}^1 \cdot \vec{x}))). \quad (1.6)$$

436 From both equations 1.5 and 1.6 it is clear that the loss function could be seen as
 437 a function of all the weights vectors of every layer of the network. So if we want to
 438 reduce the distance between the NN prediction and the true value we need to modify
 439 those weights to minimize the loss function. The most established algorithm to do so for
 440 a supervised task in a feed-forward network is the so-called *error back-propagation*.

441 The back-propagation method is an iterative technique that works essentially com-
 442 puting the gradient of the loss function with respect to the weights using the derivative
 443 chain rule and updating by a small amount the value of each parameter to lower the
 444 overall loss function at each step. Each weight is *moved* counter-gradient, and summing
 445 all the contribution to every parameter the loss function approaches its minimum. In
 446 equation 1.7 is represented the variation applied to the j^{th} weight in the i^{th} layer in a
 447 single step of the method:

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}}, \quad (1.7)$$

448 where E is the error function, and η is the *learning coefficient*, that modulate the effect of
 449 learning through all the training process. This iterative procedure is applied completely
 450 to each image in the training set several times, each time the whole dataset is reprocessed
 451 is called an *epoch*. The great majority of the dataset is exploited in the training phase

452 to keep running this trial and error process and just a small portion is left out (typically
453 10% of the data) for a final performance test.

454 The loss function shall inevitably be differentiable, and its behavior heavily influences
455 the success of the training. If the loss function presents a gradient landscape rich of
456 local minima the gradient descent process would probably get stuck in one of them.
457 More sophisticated algorithms capable of avoiding this issue have been devised, with
458 the insertion of some degree of randomness in them, as the Stochastic Gradient Descent
459 algorithm, or the wide used *Adam* optimizer [21].

460 While Error-Back Propagation is the most established standard in DL applications, it
461 suffers from some problems. The most common one is the so-called vanishing or explod-
462 ing gradient issue, which is due to the iterative chain derivation through all the nested
463 level of composition of the function. Without a careful choice of the right activation
464 function and the tuning of the learning hyper-parameters it is very easy to bump into
465 this pitfall. Furthermore the heavy use of derivation rises the inability to handle non-
466 differentiable components and hinders the possibility of parallel computation. However,
467 there are many alternative approaches to network learning beside EBP. The Minimiza-
468 tion with Auxiliary Variables (MAV) method builds upon previously proposed methods
469 that break the nested objective into easier-to-solve local subproblems via inserting aux-
470 illiary variables corresponding to activations in each layer. Such method avoids gradient
471 chain computation and the potential issues associated with it [10]. A further alterna-
472 tive approach to train the network is the Local Error Signals (LES), which is based on
473 layer-wise loss functions. In [29], is shown that layer-wise training can approach the
474 state-of-the-art on a variety of image datasets. It is used a single-layer sub-networks
475 and two different supervised loss functions to generate local error signals for the hidden
476 layers, and it is shown that the combination of these losses helps with optimization in
477 the context of local learning.

478 The training phase is the pulsing heart of a DL model development and it could
479 take even weeks on top-level computers for the most complicated networks. In fact,
480 one of the great limits to the complexity of a network during the designing phase is
481 exactly the available computational power. There are many more further technical details
482 necessary for proper training, the adjustment of which can heavily impact the quality of
483 the algorithm. However, after the training phase, we need to test the performance of the
484 NN. This is usually done running the trained algorithm on never seen before inputs (the
485 test dataset) and comparing the prediction with the ground-truth value. A good way to
486 evaluate the quality of the results is to use the same function used as the loss function
487 during the training, but there is no technical restriction to the choice of this quality
488 metric. The average score on the whole test set is then used as a numerical score for
489 the network, and it allows straightforward comparison with other models' performances,
490 trained for the same task. All this training procedure is coherently customized to every
491 different application, depending on which the problem is posed as supervised or not and
492 depending on the more or less complex network's architecture. The leitmotif is always

⁴⁹³ finding a suitable loss function that quantifies how well the network does what it has
⁴⁹⁴ been designed to do and trying to minimize it, operating on the parameters that define
⁴⁹⁵ the network structure.



Figure 1.7: Example of the resulting segmentation mask of an image of an urban landscape. Every interesting object of the image is detected and a solid color region replaces it in the segmentation mask. Every color corresponds to a different class of objects, for example, persons are highlighted in magenta and scooters in blue. The shape and the boundaries of every region should match as precisely as possible the edges of the objects.

496 1.3 Deep Learning-Based Segmentation Algorithms

497 In digital image processing, image segmentation is the process of recognizing and sub-
 498 dividing an image into different regions of pixels that show similar features, like color,
 499 texture, or intensity. Typically, the task of segmentation is to recognize the edges and
 500 boundaries of the different objects in the image and assigning a different label to every
 501 detected region. The result of the segmentation process is an image with the same dimen-
 502 sions of the starting one made of solid color regions, representing the detected objects.
 503 This image is called *segmentation mask*. In Figure 1.7 is shown an example of segmenta-
 504 tion of a picture of an urban landscape: different colors are linked to different classes of
 505 objects like persons in magenta and scooters in blue. This technology has a significant
 506 role in a wide variety of application fields such as scene understanding, medical image
 507 analysis, augmented reality, etc.

508 A relatively easy segmentation problem, and one of the first to be tackled, could
 509 be distinguishing an object from the background in a grey-scale image, like in Figure
 510 1.8. The easiest technique to perform segmentation in this kind of problem is based on
 511 thresholding. Thresholding is a binarization technique based on the image's grey-level
 512 histogram: to every pixel with luminosity above that threshold is assigned the color
 513 *white*, and vice versa the color *black*. However, this is a very primitive and fallacious, yet
 514 very fast method, and it manages poorly complex images or images with un-uniformity
 515 in the background.

516 A lot of other traditional techniques improve this first segmentation method [11].
 517 Some are based on the object's edges recognition, exploiting the sharp change in lumi-
 518 nosity typically in correspondence of the boundary of a shape. Other techniques exploit
 519 instead a region-growing technology, according to which some *seed* region markers are



Figure 1.8: Example of the resulting segmentation mask of an image of a fingerprint obtained through a thresholding algorithm. The result is not extremely good, but this technique is very easy to implement and runs very quickly.

520 scattered on the image, and the regions corresponding to the objects in the image are
521 grown to incorporate adjacent pixels with similar properties.

522 Every development of traditional computer vision or of Machine Learning-based seg-
523 mentation algorithm suffers from the same, inevitable limitation. For every one of those
524 techniques is the designing phase in which the operator should decide precisely which fea-
525 tures to extract from the image, like different directional derivatives in the image plane
526 or image entropy, and how to process them for the rest of the analysis. There is thus an
527 intrinsic limitation in the human comprehension of those quantities and in the possible
528 way to combine them. The choice is made on the previous experimental results in other
529 image processing works, and on their theoretical interpretation. Neural Networks instead
530 are relieved from this limitation, allowing themselves to learn which features are best
531 suited for the task and how they should be processed during the training phase. The
532 complexity is then moved on to the design of the DL model and on its learning phase
533 rather than on the hand-engineering design of the features to extract. In Figure 1.9 an
534 example high-level feature extracted from a DL model trained for the segmentation of
535 nuclei in a histological sample. The model learns the typical pattern of arrangement of
536 nuclei, which would have been impossible to describe equally in advance.

537 1.3.1 State of the Art on Deep Learning Segmentation

538 Similarly to many other traditional tasks, also for segmentation, there has been a thriv-
539 ing development lead by the diffusion of deep learning, that boosted the performances
540 resulting in what many regards as a paradigm shift in the field [27].

541 In further detail, image segmentation can be formulated as a classification problem of
542 pixels with semantic labels (semantic segmentation) or partitioning of individual objects
543 (instance segmentation). Semantic segmentation performs pixel-level labeling with a set
544 of object categories (e.g. boat, car, person, tree) for all the pixels in the image, hence it
545 is typically a harder task than image classification, which requires just a single label for

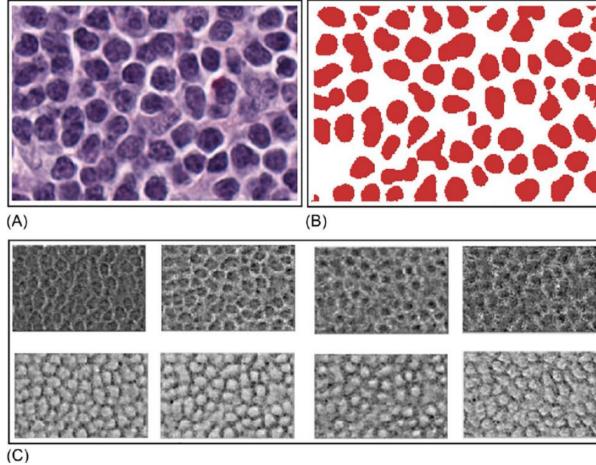


Figure 1.9: (A) An extract from an histological samples, used as input image for the model. (B) The exact segmentation mask. (C) Example of accentuated features during the training: (1-4) for back-ground recognition, (5-8) for nuclei detection. From [2].

546 the whole image. Instance segmentation extends semantic segmentation scope further
 547 by detecting and delineating each object of interest in the image (e.g. partitioning of
 548 individual nuclei in a histological image).

549 There are many prominent Neural Network architectures used in the computer vision
 550 community nowadays, based on very different ideas such as convolution, recursion, di-
 551 mensionality reduction, and image generation. This section will provide an overview of
 552 the state of the art of this technology and will dwell briefly on the details behind some
 553 of those innovative architectures.

554 Recurrent Neural Networks (RNNs) and the LSTM

555 The typical application for RNN is processing sequential data, as written text,
 556 speech or video clips, or any other kind of time-series signal. In this kind of
 557 data, there is a strong dependency between values at a given time/position and
 558 values previously processed. Those models try to implement the concept of *memory*
 559 weaving connections, outside the main information flow of the network, with the
 560 previous NN's input. At each time-stamp, the model collects the input from the
 561 current time X_i and the hidden state from the previous step h_{i-1} and outputs a
 562 target value and a new hidden state (Figure 1.10). Typically RNN cannot manage
 563 easily long-term dependencies in long sequences of signals. There is no theoretical
 564 limitation in this direction, but often it arises vanishing (or exploding) gradient
 565 problematics during the training phase. A specific type of RNN has been designed
 566 to avoid this situation, the so-called Long Short Term Memory (LSTM) [19]. The
 567 LSTM architecture includes three gates (input gate, output gate, forget gate),

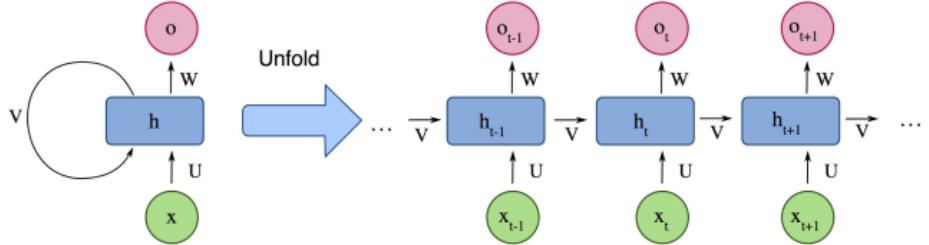


Figure 1.10: Example of the structure of a simple Recurrent Neural Network from [27].

568 which regulate the flow of information into and out from a memory cell, which
569 stores values over arbitrary time intervals.

570 Encoder-Decoder and Auto-Encoder Models

571 Encoder-Decoder models try to learn the relation between an input and the corre-
572 sponding output with a two steps process. The first step is the so-called *encoding*
573 process, in which the input x is compressed in what is called the *latent-space* rep-
574 resentation $z = f(x)$. The second step is the *decoding* process, where the NN
575 predicts the output starting from the latent-space representation $y = g(z)$. The
576 idea underneath this approach is to capture in the latent-space representation the
577 underlying semantic information of the input that is useful for predicting the out-
578 put. ED models are widely used in image-to-image problems (where both input
579 and output are images) and for sequential-data processing (like Natural Language
580 Processing, NLP). In Figure 1.11 is shown a schematic representation of this ar-
581 chitecture. Usually, these model follow a supervised training, trying to reduce the
582 reconstruction loss between the predicted output and the ground-truth output pro-
583 vided while training. Typical applications for this technology are image-enhancing
584 techniques like de-noising or super-resolution, where the output image is an im-
585 proved version of the input image. Or image generation problems (e.g. plausible
586 new human faces generation) in which all the properties which define the type of
587 image under analysis should be learned in the representation latent space.

Generative Adversarial Networks (GANs)

The peculiarity of Generative Adversarial Network (GAN) lies in its structure. It is actually made of two distinct and independent modules: a generator and a discriminator, as shown in Figure 1.12. The first module G , responsible for the generation, typically learns to map a prior random distribution of input z to a target distribution y , as similar as possible to the target $G = z \rightarrow y$ (i.e. almost any kind of image-to-image problem could be addressed with GANs, as in [20]). The

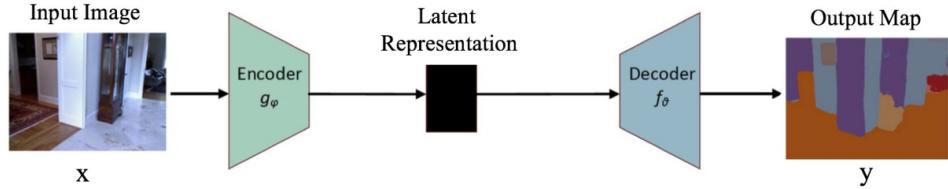


Figure 1.11: Example of the structure of a simple Encoder-Decoder Neural Network from [27].

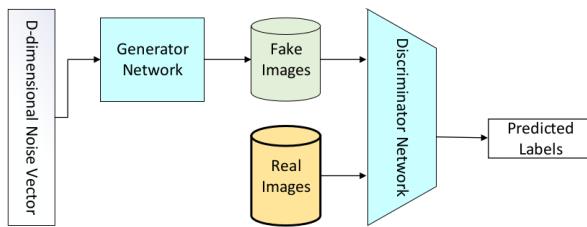


Figure 1.12: Schematic representation of a Generative Adversarial Networks, form [27].

second module, the discriminator D , instead is trained to distinguish between *real* and *fake* images of the target category. These two networks are trained alternately in the same training process. The generator tries to fool the discriminator and vice versa. The name adversarial is actually due to this *competition* within different parts of the network. The formal manner to set up this adversarial training lies in the accurate choice of a suitable loss function, that will look like:

$$L_{GAN} = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

. The GAN is thus based on a min-max game between G and D . D aims to reduce the classification error in distinguishing fake samples from real ones, and as a consequence maximizing the L_{GAN} . On the other hand, G wants to maximize the D 's error, hence minimizing L_{GAN} . The result of the training process is the trained generator G^* , capable of produce an arbitrary number of new data (images, text, or whatever else):

$$G^* = \arg \min_G \max_D L_{GAN}$$

. This peculiar architecture has yielded several interesting results and it has been developed in many different directions, with influences and contaminations with other architectures [20].

588 . Convolutional Neural Networks (CNNs)

589

590

592 As stated before CNNs are a staple choice in image processing DL applications.
593 They mainly consist of three types of layers:

- 594 i convolutional layers, where a kernel window of parameters is convolved with
595 the image pixels and produce numerical features maps.
- 596 ii nonlinear layers, which apply an activation function on feature maps (usually
597 element-wise). This step allows the network to introduce non-linear behavior
598 and then increasing its modeling capabilities.
- 599 iii pooling layers, which replace a small neighborhood of a feature map with some
600 statistical information (mean, max, etc.) about the neighborhood and reduce
601 the spatial resolution.

602 Given the arrangement of successive layers, each unit receives weighted inputs from
603 a small neighborhood, known as the receptive field, of units in the previous layer.
604 The stack of layers allows the NN to perceive different resolutions: the higher-
605 level layers learn features from increasingly wider receptive fields. The leading
606 computational advantage given by CNN architecture lies in the sharing of kernels'
607 weights within a convolutional layer. The result is a significantly smaller number
608 of parameters than fully-connected neural networks. In section 2.7 will be shown a
609 particular application of this architecture, known as *style-transfer* network, which
610 is a particular algorithm capable of implanting the visual texture of a *style* image
611 onto the content of a different image, producing interesting hybrid images. Some
612 of the most notorious CNN architectures include: AlexNet [23], VGGNet [40], and
613 U-Net [34].

614 For this work, U-net architecture is particularly interesting. The U-net model was
615 initially developed for biomedical image segmentation, and in its structure reflects char-
616 acteristics of both CNN and Encoder-Decoder models. Ronneberger et al.[34] proposed
617 this model for segmenting biological microscopy images in 2015. The U-Net architecture
618 is made of two branches, a contracting path to capture context, and a symmetric expand-
619 ing path (see Figure 1.13). The down-sampling flow is made of a Fully Convolutional
620 Network (FCN)-like architecture that computes features with 3×3 kernel convolutions.
621 On the other hand, the up-sampling branch exploits up-convolution operations (or de-
622 convolution), reducing the number of feature maps while increasing their dimensions.
623 Another characteristic of this architecture is the presence of direct connections between
624 layers of a similar level of compression in compressing and decompressing branches.
625 Those links allow the NN to preserve spatial and pattern information. The Network
626 flow eventually ends with a 1×1 convolution layer responsible for the generation of the
627 segmentation mask of the input image.

628 A recent example of a practical application of a CNN to histological images could
629 be found in [28]. In this work the Inception v3 is trained on hand-labeled samples of

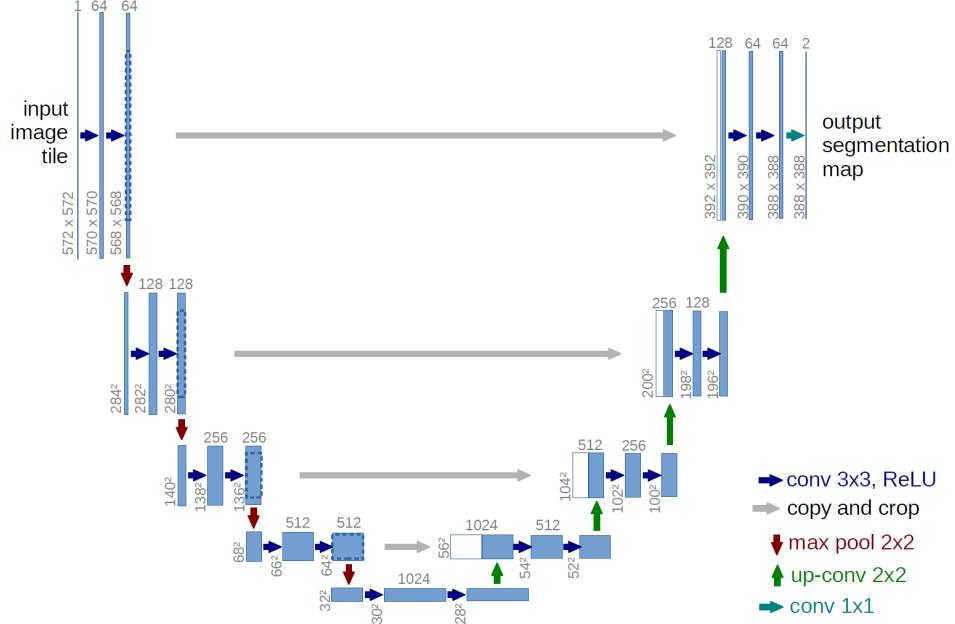


Figure 1.13: Scheme of the typical architecture of a U-net NN. This particular model was firstly proposed by Ronneberger et al. in [34].

pancreatic tissue (like in Figure 1) to recognise tumoral regions from healthy ones in a pancreatic tissue specimen treated with Ki67 staining. The Inception v3 [33] network is a deep convolutional network developed by Google, trained for object detection and image classification on the ImageNet dataset [15]. Recognition of tumoral region of Ki67 stained pancreatic tissue samples is based on the detection and counting of some specific marker cells. In Figure 1.14 is shown a pair of original image and the computed segmentation mask, which label in red tumoral regions and in green the healthy ones. This work is based on a technique called **transfer learning**, which consists in the customization and specialization of a pre-trained NN previously trained for similar, but essentially different, tasks. The final part of the training of this version of Inception v3 has been performed of a dataset of 33 whole slide images of Ki67 stained neuroendocrine tumor biopsies acquired from 33 different patients, digitized with a $20\times$ magnification factor and successively divided in 64×64 patches.

1.3.2 Image Segmentation Datasets

Besides the choice of suitable architecture the most important aspect while developing a NN is the dataset on which perform the training process. Let's confine the discussion only to image-to-image problems, like segmentation problems. There are a lot of widely used datasets, but I want to mention just a few of them to give the idea of their typical

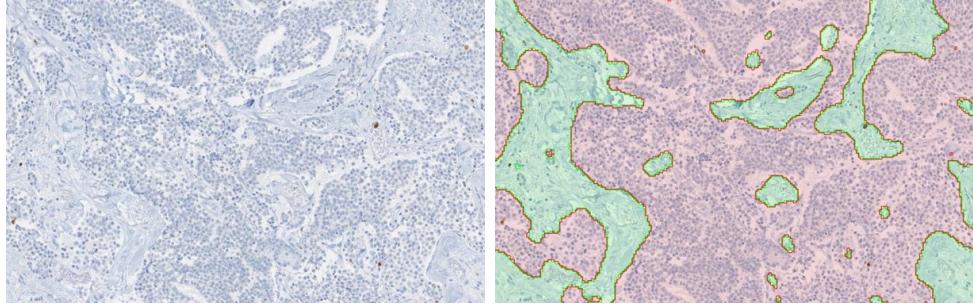


Figure 1.14: (left) Original image of a Ki67 stained pancreatic tissue sample, (right) the corresponding segmentation mask, which label in red tumoral regions and in green the healthy ones. From [28].

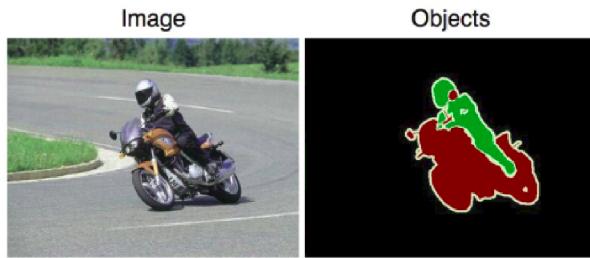


Figure 1.15: An example image from the PASCAL dataset and its corresponding segmentation mask [17].

648 characteristics.

649 A good example of segmentation is the Cityscapes dataset [12], which is a large-scale
 650 database with a focus on semantic understanding of urban street scenes. The dataset
 651 is made of video sequences from the point of view of a car in the road traffic, from 50
 652 different cities in the world. The clips are made of 5K frames, labeled with extremely high
 653 quality at pixel-level and an additional set of 20K weakly-annotated frames. Each pixel
 654 in the segmentation mask contains the semantic classification, among over 30 classes of
 655 objects. An example of an image from this dataset is shown in Figure 1.7.

656 The PASCAL Visual Object Classes (VOC) [17] is another of the most popular
 657 datasets in computer vision. This dataset is designed to support the training of algo-
 658 rithms for 5 different tasks: segmentation, classification, detection, person layout, and
 659 action recognition. In particular, for segmentation, there are over 20 classes of labeled
 660 objects (e.g. planes, bus, car, sofa, TV, dogs, person, etc.). The dataset comes divided
 661 into two portions: training and validation, with 1,464 and 1,449 images, respectively. In
 662 Figure 1.15 is shown an example of an image and its corresponding segmentation mask.

663 As last mention I would report t he ImageNet project [15], which is a large visual
 664 database designed for use in visual object recognition software research. It consists

665 of more than 14 million images have been hand-annotated by the project to indicate
666 what objects are pictured and in at least one million of the images, bounding boxes
667 are also provided. ImageNet contains more than 20,000 categories of objects. Since
668 2010, the ImageNet project runs an annual software contest, the ImageNet Large Scale
669 Visual Recognition Challenge (ILSVRC), where software programs compete to correctly
670 classify and detect objects and scenes. This kind of competitions is very important
671 for the research field, as it inspire and encourage the development of new models and
672 architectures.

673 It is worth mentioning that in the medical image processing domain typically the
674 available dataset is definitely not that rich and vast (that is actually the seed of this
675 work) and thus many techniques of data augmentation have been devised, to get the
676 best out of the restricted amount of material. Generally, data augmentation manipu-
677 lates the starting material applying a set of transformation to create new material, like
678 rotation, reflection, scaling, cropping and shifting, etc. Data augmentation has been
679 proven to improve the efficacy of the training, making the model less prone to over-
680 fitting, increasing the generalization power of the model, and helping the convergence to
681 a stable solution during the training process.

⁶⁸² Chapter 2

⁶⁸³ Technical Tools for Model ⁶⁸⁴ Development

⁶⁸⁵ As mentioned in the introduction, this project wants to produce synthetic histological
⁶⁸⁶ images paired with their corresponding segmentation mask, to train Neural Networks
⁶⁸⁷ for the automatization of real histological images analysis. The production of artificial
⁶⁸⁸ images passes through the processing of a three dimensional, virtual model of a his-
⁶⁸⁹ tological structure, which is the heart of this thesis work. The detailed description of
⁶⁹⁰ the development of the two proposed histological models will follow the present chapter
⁶⁹¹ and will occupy all the chapter 3. Here I will dwell, instead, on every less common
⁶⁹² tool employed during the models' designing phase. From the practical point of view, this
⁶⁹³ project is quite articulated and the development has required the harmonization of many
⁶⁹⁴ different technologies, tools, and code libraries. The current chapter should be seen as a
⁶⁹⁵ theoretical complement for chapter 3, and its reading is suggested to the reader for any
⁶⁹⁶ theoretical gap or for any further technical deepening. The reader already familiar with
⁶⁹⁷ those technical tools should freely jump to the models' description.

⁶⁹⁸ All the code necessary for the work has been written in a pure Python environment,
⁶⁹⁹ using several already established libraries and writing by my self the missing code for
⁷⁰⁰ some specific applications. I decided to code in Python given the thriving variety of
⁷⁰¹ available libraries geared toward scientific computation, image processing, data analysis,
⁷⁰² and last but not least for its ease of use (compared to other programming languages).
⁷⁰³ In each one of the following subsections I will mention the specific code libraries which
⁷⁰⁴ have been employed in this project for every technical necessity.

⁷⁰⁵ 2.1 Quaternions

⁷⁰⁶ Quaternions are, in mathematics, a number system that expands to four dimensions the
⁷⁰⁷ complex numbers. They have been described for the first time by the famous mathemati-

708 cian William Rowan Hamilton in 1843. This number system define three independent
 709 *imaginary* units $\mathbf{i}, \mathbf{j}, \mathbf{k}$ as in (2.1), which allows the general representation of a quaternion
 710 \mathbf{q} is (2.2) and its inverse \mathbf{q}^{-1} (2.3) where a, b, c, d are real numbers:

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1, \quad (2.1)$$

$$\mathbf{q} = a + bi + cj + dk, \quad (2.2)$$

$$\mathbf{q}^{-1} = (a + bi + cj + dk)^{-1} = \frac{1}{a^2 + b^2 + c^2 + d^2} (a - bi - cj - dk). \quad (2.3)$$

711 Furthermore, the multiplication operation between quaternionn does not benefit from
 712 commutativity, hence the product between basis elements will behave as follows:

$$\begin{aligned} \mathbf{i} \cdot 1 &= 1 \cdot \mathbf{i} = \mathbf{i}, & \mathbf{j} \cdot 1 &= 1 \cdot \mathbf{j} = \mathbf{j}, & \mathbf{k} \cdot 1 &= 1 \cdot \mathbf{k} = \mathbf{k} \\ \mathbf{i} \cdot \mathbf{j} &= \mathbf{k}, & \mathbf{j} \cdot \mathbf{i} &= -\mathbf{k} \\ \mathbf{k} \cdot \mathbf{i} &= \mathbf{j}, & \mathbf{i} \cdot \mathbf{k} &= -\mathbf{j} \\ \mathbf{j} \cdot \mathbf{k} &= \mathbf{i}, & \mathbf{k} \cdot \mathbf{j} &= -\mathbf{i}. \end{aligned} \quad (2.4)$$

713 This number system has plenty of peculiar properties and applications, but for this
 714 project, quaternions are important for their ability to represent, in a very convenient way,
 715 rotations in three dimensions. The particular subset of quaternions with vanishing real
 716 part ($a = 0$) has a useful, yet redundant, correspondence with the group of rotations in
 717 tridimensional space $\text{SO}(3)$. Every 3D rotation of an object can be represented by a 3D
 718 vector \vec{u} : the vector's direction indicates the axis of rotation and the vector magnitude $|\vec{u}|$
 719 express the angular extent of rotation. However, the matrix operation which expresses
 720 the rotation around an arbitrary vector \vec{u} it is quite complex and does not scale easily
 721 for multiple rotations [7], which brings to very heavy and entangled computations.

722 Using quaternions for expressing rotations in space, instead, it is very convinient.
 723 Given the unit rotation vector \vec{u} and the rotation angle θ , the corresponding rotation
 724 quaternion \mathbf{q} becomes (2.6):

$$\vec{u} = (u_x, u_y, u_z) = u_x \mathbf{i} + u_y \mathbf{j} + u_z \mathbf{k}, \quad (2.5)$$

$$\mathbf{q} = e^{\frac{\theta}{2}(u_x \mathbf{i} + u_y \mathbf{j} + u_z \mathbf{k})} = \cos \frac{\theta}{2} + (u_x \mathbf{i} + u_y \mathbf{j} + u_z \mathbf{k}) \sin \frac{\theta}{2}, \quad (2.6)$$

$$\mathbf{q}^{-1} = \cos \frac{\theta}{2} - (u_x \mathbf{i} + u_y \mathbf{j} + u_z \mathbf{k}) \sin \frac{\theta}{2}, \quad (2.7)$$

725 where in (2.6) we can clearly see a generalization of the Euler's formula for the
 726 exponential notation of complex numbers, which hold for quaternions. It can be shown

727 that the application of the rotation represented by \mathbf{q} on an arbitrary 3D vector \vec{v} should
728 be easily expressed as:

$$\vec{v}' = \mathbf{q}\vec{v}\mathbf{q}^{-1}, \quad (2.8)$$

729 using the Hamilton product defined on quaternions (2.4). This rule raises a very con-
730 vinient and an extremely scalable way to compute consecutive rotations in space. Given
731 two independent and consecutive rotations represented by the two quaternions \mathbf{q} and \mathbf{p}
732 applied on the vector \vec{v} the resulting rotated vector \vec{v}' is simply yielded as:

$$\vec{v}' = \mathbf{p}(\mathbf{q}\vec{v}\mathbf{q}^{-1})\mathbf{p}^{-1} = (\mathbf{pq})\vec{v}(\mathbf{qp})^{-1}, \quad (2.9)$$

733 which essentially is the application of the rotation $\mathbf{r} = \mathbf{qp}$ on the vector \vec{v} . This repre-
734 sentation is completely coherent with the algebra of 3D rotations, which does not benefit
735 from commutativity in turn.

736 Given this property, quaternions are indeed widely used in all sorts of applications
737 of digital 3D space design, as for simulations and videogame. The position of an object
738 in the space in simulations is generally given by the application of several independent
739 rotations, typically in the order of a tenth of rotations, which with quaternions is given
740 easily by the product of simple objects. Every other alternative method would imply the
741 use of matrix representation of rotations or other rotation systems as Euler's angles and
742 would eventually make the computation prohibitive.

743 The use of quaternions in this work will be justified in section 3.1, while speaking of
744 parametric L-systems in 3D space, used to build the backbone of the ramified structure
745 of blood vessels in the reconstruction of a sample of pancreatic tissue.

746 I was able to find many Python libraries for computation with quaternions, but the
747 one I appreciated the most for its interface and ease of use was the `pyquaternion`. With
748 this library, it's immediate the definition of a quaternion by its correspondent rotation
749 vector, and the multiplication between quaternions is straightforward.

750 2.2 Parametric L-Systems

751 Lindenmayer systems, or simply L-systems, were conceived as a mathematical theory of
752 plant development [25] in 1968 by Aristid Lindenmayer. Successively, a lot of geometri-
753 cal interpretations of L-systems were proposed to make them a versatile instrument for
754 modeling the morphology typical of plants and other organic structures. As a biologist,
755 Lindenmayer studied different species of yeast and fungi and worked the growth patterns
756 of various types of bacteria (e.g. as the *cyanobacteria Anabaena catenula*). The main
757 purpose for which L-systems were devised was to allow a formal description of the devel-
758 opment of simple multicellular living organisms. Subsequently, the potentiality of these
759 systems was expanded to describe higher-order plants and complex branching structures.

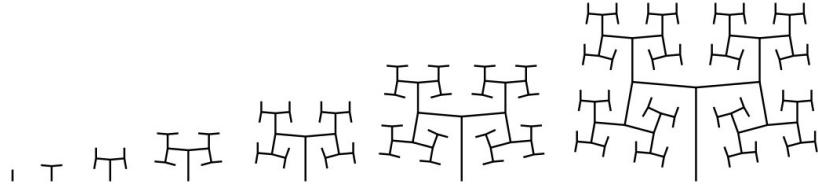


Figure 2.1: Growth pattern for the space-filling fractal-like system, used to mimic the blood vessel bifurcations in sec 3.1.

760 An L-system is in general defined by an *axiom* sequence and some development *rules*,
 761 which are recursively applied to the sequence and lead its development. The original
 762 proposed L-system was fairly simple and shows really well the idea underneath:

$$\begin{aligned} \textit{axiom} &: A \\ \textit{rules} &: (A \rightarrow AB), \quad (B \rightarrow A) \end{aligned}$$

763 where A and B could be any two different patterns in the morphology of an algae, or
 764 could be different bifurcations in a ramified structure. The iterative application of the
 765 rules to the axiom sequence, let's say for 7 times, will produce the following sequence:

$$\begin{aligned} n = 0 & : A \\ n = 1 & : AB \\ n = 2 & : ABA \\ n = 3 & : ABAAB \\ n = 4 & : ABAABABA \\ n = 5 & : ABAABABAABAAB \\ n = 6 & : ABAABABAABAABABAABABA \\ n = 7 & : ABAABABAABAABABAABAABABAABAAB . \end{aligned}$$

766 This kind of tool, as will be shown also in 3.1, is particularly suited for the creation of
 767 structures with fractal behavior, and it has been used in this work to create the backbone
 768 of the entangled bifurcation in blood vessels in the modelization of pancreatic tissue. In
 769 particular, there was the need for a fractal-like space-filling ramification as the one shown
 770 in Figures 2.1.

771 The system in Figure 2.1 represent the successive ramification of a structure which
 772 grows adding segments gradually shorter, by a lenght ratio parameter R and inclined
 773 of $\delta = \pm 85^\circ$ respect the previous branch. The axiom and the rules that produce this
 774 structure are the following:

$$\begin{aligned}
& axiom : A \\
& rule_1 : A \rightarrow F(1)[+A][-A] \\
& rule_2 : F(s) \rightarrow F(s \cdot R)
\end{aligned} \tag{2.10}$$

775 where A represent the start of a new branch and $F(s)$ represent a branch of lenght s .
776 The presence of a rule which acts differently depending on the target object, is an further
777 sophistication respect to the standard L-system. For this reason these systems are called
778 *parametric* L-systems.

779 The use of standard L-systems turned out to be widespread, and there were a lot of
780 different Python libraries at my disposal for coding. By the way, parametric L-systems
781 were not just as popular, and I was not able to find a reliable library on which to build
782 my work. I decided then to code a parametric branching system able to recreate the
783 structure with rules (2.10) at any desired level of iteration. Having created the tool I
784 needed on my own I was able to add all the optional features I would have needed during
785 the development, like an adjustable degree of angular noise in the branch generation.

786 2.3 Voronoi Tassellation

787 Voronoi diagrams, or Voronoi decompositions, are space-partitioning systems, which
788 divides an n -dimensional Euclidian space into sub-regions depending on the proximity
789 to a given set of objects. More precisely, given an n -dimensional space and m starting
790 point p_1, \dots, p_m inside it, the whole space will be subdivided in m adjacent regions. Every
791 point of the space is assigned to the region correspondent to the nearest starting point.
792 In Figure 2.2 is shown a practical example of a Voronoi decomposition of a plane into
793 20 regions corresponding to the 20 starting points. Informal use of Voronoi diagrams
794 can be traced back to Descartes in 1644, and many other mathematicians after him.
795 But, Voronoi diagrams are named after Georgy Feodosievych Voronoy who defined and
796 studied the general n-dimensional case in 1908 [46].

797 More precisely, let X be a metric space and d the distance defined on it. Let K be
798 the set of indices and let $(P_k)_{k \in K}$ be the tuple of sites in the space X . The k^{th} Voronoi
799 cell R_k , associated with the site P_k is the set of all the points in X whose distance to P_k
800 is smaller than the distance to any other site P_j , with $j \neq k$, or in other words:

$$R_k = \{x \in X \mid d(x, P_k) \leq d(x, P_j) \forall j \in K, j \neq k\}, \tag{2.11}$$

801 depending on the notion of distance defined on the space X the final redistribution in
802 subregions will look very differently.

803 In addition to the choice of the distance function, another fundamental factor is the
804 distribution of sites in the space to be divided. If the points are chosen equally and ho-
805 mogenously distributed the final distribution will appear as a simple regular lattice, while

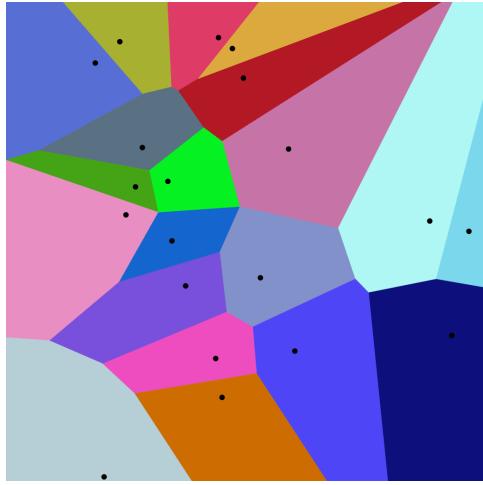


Figure 2.2: Example of a Voronoi decomposition of a plane into 20 regions corresponding to 20 starting points.

806 a completely random distribution of points in the space will provide a decomposition in
 807 cells with very different shapes and volumes, as shown in Figure 2.3. Interesting results
 808 concerning points from a semi-random distribution will be shown in section 3.1, which
 809 leads to decomposition with a good richness in shapes but with the desired homogeneity
 810 in volumes.

811 The Voronoi decomposition has been of great interest in this project for the division
 812 of a 3D space in subregions, to recreate the spatial distribution of cells in a sample of
 813 human tissue, as will be shown in section 3.1. The formal definition of Voronoi regions
 814 (2.11) ensures the convexity of each decomposition's tassel, which in three-dimensional
 815 space would be adjacent convex polyhedrons. Every tassel of the decomposition will
 816 be represented by a bounded 3-dimensional convex hull¹, with except for those most
 817 external cells which are unbounded and requires special attention while using.

818 The most widespread tool for the computation of Voronoi decompositions in Python
 819 is contained in the `spatial` submodule of the famous library `SciPy` [45], which is a staple
 820 tool for an incredible variety of scientific algorithms. The `Voronoi` object from `Scipy`
 821 library offers a very efficient algorithm for space-partitioning, and it has been one of the
 822 pillars for the modelization of tissues. Unluckily this module does not easily allow to
 823 perform Voronoi decomposition with different definitions of distance functions d other
 824 than the Euclidian distance, which would have allowed interesting studies.

¹See section 2.5 for further details.

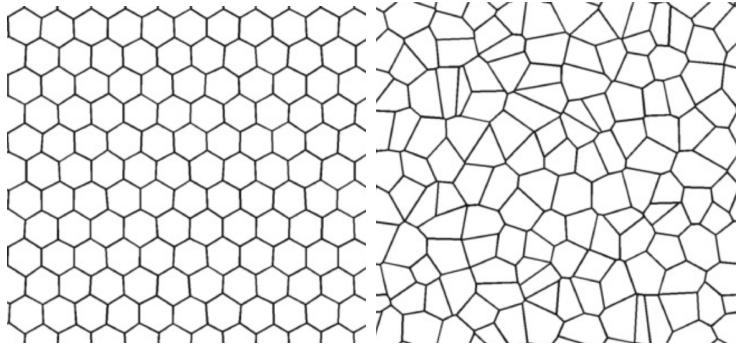


Figure 2.3: On the right an example of 2D Voronoi decomposition resulting from homogeneously distributed points in the plane. On the left the resulting decomposition obtained from randomly distributed points in the plane, from [3].

825 2.4 Saltelli Algorithm - Random Number Generation

826 As mentioned in section 2.3, in this project there was the need for quasi-random number
 827 generation for the production of Voronoi tessellations. Quasi-random sequences (or low-
 828 discrepancy sequences) are patterns of numbers that emulate the behavior of uniform
 829 random distributions but have a more homogeneous and quick coverage of the sampling
 830 domain, which provides an important advantage in applications as in quasi-Monte Carlo
 831 integration techniques, as shown in Figure 2.4. In computer science there is not any
 832 possibility of recreating *true* random sequences, hence any stochasticity is completely
 833 deterministic in its essence even if produced by very chaotic processes². Indeed, every
 834 algorithm for random number generation is completely repeatable given its starting sta-
 835 tus. Quasi-random sequences are completely deterministic too, but implements more
 836 *regular*, well-behaved algorithms.

837 A first good example to understand the concept of quasi-random generation could be
 838 an additive recurrence, as the following:

$$s_{n+1} = (s_n + \alpha) \bmod 1, \quad (2.12)$$

839 which for every seed element s_0 and real parameter α produced completely different
 840 sequences.

841 In the bottom line of Figure 2.4 is clearly visible the good and homogeneous coverage
 842 of the sampling domain, although it is strongly visible a regular pattern between points,
 843 which does not convey an *organic* sensation at all. However, increasing the complexity of

²A chaotic process is a deterministic process which has an extremely sensible dependence on its starting conditions. This property mimics very effectively the behavior of true random processes, which are intrinsically forbidden in computer science.

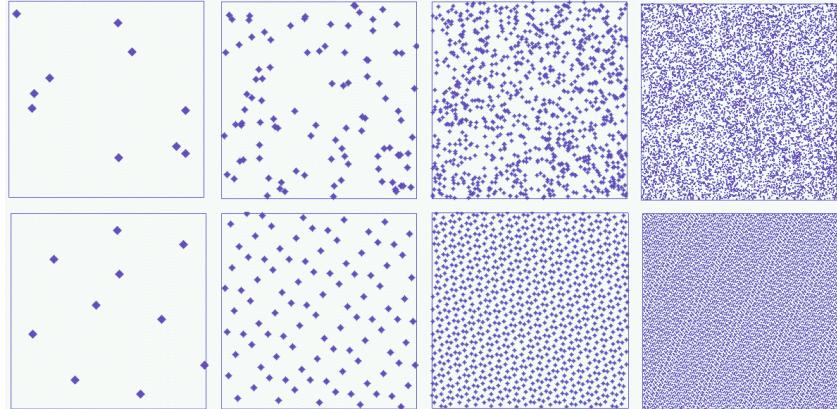


Figure 2.4: Coverage of the unit square with an additive quasirandom numbers sequence as in 2.12 (up) and for uniformly sampled random numbers (bottom). From left to right: 10, 100, 1000, 10000 points.

844 our very simple starting model 2.12 it is possible to overcome this *artificial* appearance
 845 of sampled points and to produce very good samples.

846 A notorious algorithm for quasi-random number generation is the Sobol sequence,
 847 introduced by the russian mathematician Ilya M. Sobol in 1967 [43]. In its work, Sobol
 848 wanted to construct a sequence x_n of points in the s -dimensional unitary hypercube
 849 $I^s = [0, 1]^s$ such as for any integrable function f :

$$\lim_{n \rightarrow \infty} \frac{1}{n} \sum_{i=1}^n f(x_i) = \int_{I_s} f. \quad (2.13)$$

850 Sobol wanted to minimize the *holes* in the sampled domain (which it could be shown
 851 to be a property that helps the convergence of the sequence) and minimize as well
 852 the *holes* in every lower-dimension projection of the sampled points. The particularly
 853 good distributions that fulfill those requirements are known as (t, m, s) -nets and (t, s) -
 854 sequences in base b .

855 To better understand them we need first to define the concept of s -interval in base b ,
 856 which is a subset of I_s such as:

$$E_s^b = \prod_{j=1}^s \left[\frac{a_j}{b^{d_j}}, \frac{a_j + 1}{b^{d_j}} \right), \quad (2.14)$$

857 where a_j and d_j are non-negative integers, and $a_j < b^{d_j}$ for all j in $\{1, \dots, s\}$.

858 Let be t and m two integers such as $0 \leq t \leq m$. A (t, m, s) -net in base b is defined
 859 as a sequence x_n of b^m points of I_s such that:

$$\text{Card } \mathbf{P} \cap \{x_1, \dots, x_n\} = b^t \quad (2.15)$$

860 for all the elementary interval \mathbf{P} in base b of hypervolume $\lambda(\mathbf{P}) = b^{t-m}$.

861 Given a non-negative integer t , a (t, s) -sequence in base b is an infinite sequence of
862 points x_n such that for all integers $k \geq 0$, $m \geq t$ the sequence $\{x_{kb^m}, \dots, x_{(k+1)b^m-1}\}$ is
863 a (t, m, s) -net in base b .

864 Sobol in his article described in particular (t, m, s) -net and (t, s) -sequence in base 2.
865 A more thorough description of all the formal properties of those particular sequences
866 could be found in [42].

867 In order to perform the actual sampling during the modelization, it has been used the
868 `saltelli` module from the `SALib` library, which performs sampling in an s -dimensional
869 space following the Saltelli algorithm, which is a specific improved version of the Sobol
870 algorithms oriented toward the parameter sensitivity analysis [36], [37].

871 2.5 Planar Section of a Polyhedron

872 As will be shown in section 3.1 a fundamental step for the functioning of the modelization
873 is the planar section of a three-dimensional polyhedron. It turned out that there is
874 no general rule to perform a planar section of a convex polyhedron with an arbitrary
875 number of faces, respect to an arbitrary sectioning plane. Hence, I devised an algorithm
876 to handle this task. In the case of a full intersection, the result of the sectioning process
877 of a polyhedron is a polygonal surface, otherwise, it could be an empty set of points or
878 a segment in case of particular tangency, but those two cases are not of interest to the
879 model. This tool has been created from scratch by me in the preliminar design phase and
880 also the implemented algorithm has been devised by me and chosen as the best option
881 among other possibilities. It is not fruit of an extended and thorough formal analysis,
882 hence it has not the prententious to be an extremely optimized algortihm. Nevertheless,
883 this tool passed all the test I required and yielded the expected results.

884 Given a convex polyhedron with n vertices and a sectioning plane p , let V be the set
885 of all the vertices and $f_p(\vec{x})$ the equation defining the plane. The algorithm is defined
886 by the following steps:

- 887 1. Divide V in two subsets: A made of those vertices which lie above and B , made of
those which lie below the sectioning plane. Like in 2.16:

$$A = \{\vec{v} \in V \mid f_p(\vec{v}) \geq 0\} \quad (2.16)$$
$$B = \{\vec{v} \in V \mid f_p(\vec{v}) \leq 0\}$$

887 If any of the two subsets tourns out to be empty the plane p does not intersect the
888 polyhedron, and the section is empty. A and B are represented in different colors
889 in Figure 2.5.

- 890 2. Detect, and *draw*, any possible line that crosses two points respectively from A and
 B . If n_A and n_B are the numbers of points above and below the plane then there

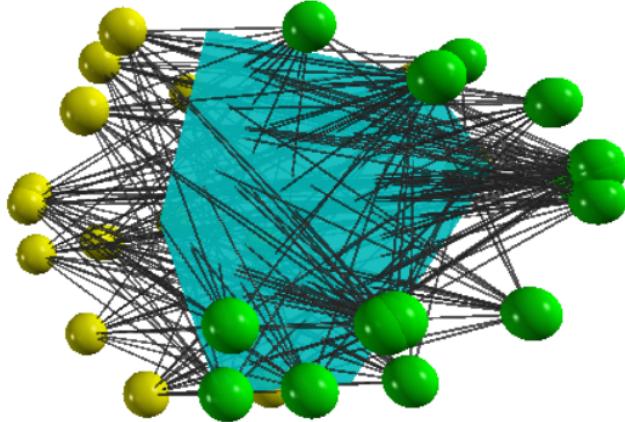


Figure 2.5: In this picture is shown an example of the application of the algorithm for the planar section on a polyhedron. The vertices are divided into two groups, with different colors yellow and green. All the possible lines between any couple of vertices picked from the two classes are drawn in dark gray. In Turquoise the resulting planar section, obtained as the convex hull containing all the intersections between the lines and the plane.

will be $n_A \times n_B$ possible lines. In Figure 2.5 all the lines between the two classes of points are drawn in dark gray.

- 892 3. Detect P , the set of all the points from the intersection between the $n_A \times n_B$ lines
893 from the previous step and the sectioning plane p . All these points will lie on the
894 same plane, within the boundaries of the polygonal section.
- 895 4. The final polygon is then yielded by computing the convex hull of the points in
896 P . The convexity of the starting polyhedron in fact ensures the convexity of any
897 section of the solid.

900 The result of the algorithm is, as just stated, a convex hull, which in geometry
901 is defined as the smallest convex envelope or convex closure of a set of points. In 2
902 dimensions is the smallest convex polygon containing a certain set of points in a plane
903 (In Figure 2.6 is shown the so-called “rubber band effect”), and in 3 dimensions it is the
904 smallest convex polyhedron containing a set of points in the space.

905 In Python, the most convenient way to work with convex hulls was to use the sub-
906 module `spatial.ConvexHull` from the `SciPy` library [45]. This module allows also a
907 convenient way for plotting images with `MatPlotLib`, which is the point of reference for
908 plotting and image formation in Python.

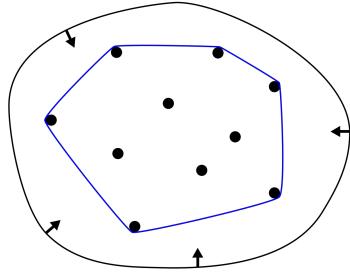


Figure 2.6: Representation of the convex hull of a bounded planar set of points. This particular enclosure goes under the name of "rubber band effect".

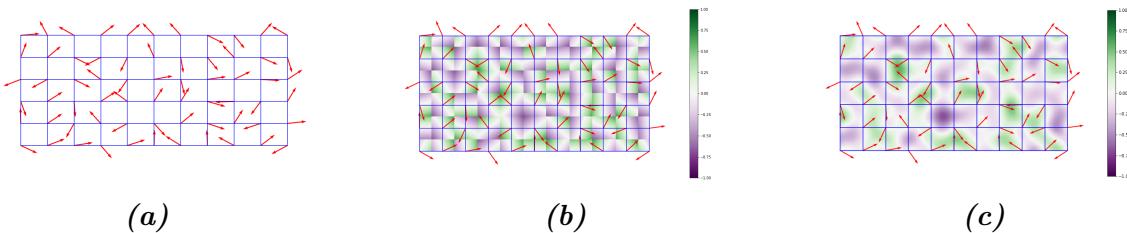


Figure 2.7: The three main steps of the algorithm to produce Perlin noise. In 2.7a the plane discretization and the assignment of a gradient vector to every node of the grid. In 2.7b the computation of the dot product with all the points inside the discretization and in 2.7c the interpolation of the values to create the final function.

909 2.6 Perlin Noise

910 Perlin noise is a widely used form of noise in computer graphics, which mimics very
 911 well natural and smooth fluctuations around a constant value. It has been developed by
 912 Ken Perlin in 1983, and it is now the staple tool for giving texture to object in virtual
 913 modelization, often considered the *salt* of computer graphics texturization. The Perlin
 914 noise is a gradient-based algorithm defined on grid discretization of a n -dimensional
 915 space. The algorithm involves three subsequent steps:

- 916 1. The first step is to discretize the n -dimensional space in a regular lattice: the
 917 dimension of the grid will impact heavily on the scale of the noise. As in Figure
 918 2.7a at every node of the grid is assigned a randomly oriented n -dimensional unitary
 919 gradient vector. This is the preliminary setup which will allow the computation of
 920 the actual noise function in every point of the space.
- 921 2. Given the candidate point \vec{x} between the grid nodes onto which evaluate the noise
 922 there are 2^n nearest grid nodes. For each one of these 2^n nodes, it is evaluated the

923 distance vector from \vec{x} as the offset between the two points. Then it is computed
924 the dot product between every pair made of nearby gradient vectors and the offset
925 vector. This operation should be thought of as made on every point in the lattice,
926 as in Figure 2.7b, where at every point of the grid is represented just one of the
927 $2^2 = 4$ series of dot products.

- 928 3. The final step is the interpolation between the 2^n series of dot products. To per-
929 form the interpolation usually is used a function with vanishing first degree (and
930 preferably also second degree) derivative in correspondence of the 2^n grid nodes ³.
931 This means that the noise function will pass through zero at every node and have
932 a gradient equal to the pre-computed grid node gradient. These properties give
933 Perlin noise its characteristic spatial scale and smoothness.

934 In general, the final result of the algorithm is a smooth function with a random-
935 like behavior that mimics really well an organic appearance, like in Figure 2.8, with
936 fluctuation around the value 0, with amplitude $\in [0; 1]$. The surface in Figure 2.8 has
937 been produced plotting in 3D the results of the function `pnoise2` from the library `noise`,
938 which offers a tool for the production of different type of noise. In order to put in practical
939 use this module some adjustment were required. The particular function `pnoise2` simply
940 yields the value of the Perlin noise surface in correspondence to a single (x, y) point in the
941 plane in a deterministic way. There was not the possibility to generate different whole
942 Perlin surfaces every run. To overcome this limitation I made a vectorized⁴ version of
943 `pnoise2` able to evaluate the function over an arbitrary wide grid of points expressed as
944 set of all the pairs of coordinates (x, y) of the grid's nodes. In this way a single call to
945 the function is able to produce the entire surface covering the grid, in the form of a single
946 NumPy 3D-array. Furthermore, to recover the possibility of generating always different
947 surfaces as in a random generation, I inserted an offset coordinate (x_O, y_O) , which moves
948 in the plane the origin of the surface generation. This pair of offset coordinates then acts
949 also as a *seed* in the generation, allowing to completely recreate previously generated
950 material in a controlled way.

951 2.7 Style-Transfer Neural Network

952 Style-Transfer Neural Networks are common models, able of creating new hybrid images
953 implanting the visual style from an image preserving the visual content of another image.

³Usually are used functions with a sigmoidal behavior, like any smoothstep function, which is a family of very common items in computer graphics.

⁴In Python the staple tool for scientific, number-cruncher computation is the NumPy library, which allows a fast, complete and efficient way to perform computation between number structure. The operation of trasnforming a function which acts on a single value (or pair of values) to a function able to perform on a suitable datastructure is called *vectorization*, and its the recomended way to proceed when handling numerical functions in Python.

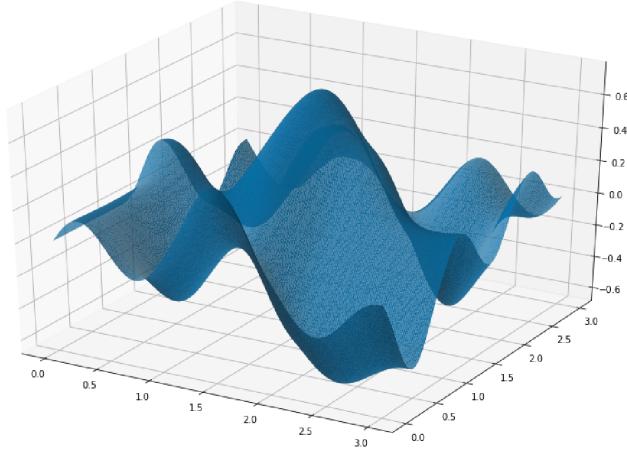


Figure 2.8: Example of Perlin noise 2D function produced while working on this project. This surface offers a smooth variation around the value 0 with amplitude $\in [0; 1]$.

954 The two images necessary for the algorithm are called *style* image S and *content* image
 955 C , and the resulting *styled* picture X , as in Figure 2.9.

956 There are many different tested and comparable architectures to compute this kind
 957 of algorithm. In my work I decided to use in particular the procedure described in [18],
 958 using the PyTorch ecosystem to implement the necessary code.

959 The backbone of the architecture is the VGG-19 network, which is a convolutional
 960 neural network 19 layers deep, as in Figure 2.10. This huge model has been pre-trained
 961 on over a million images from the ImageNet database [16], for the classification into
 962 over than 1000 classes of objects. As a result, the network has learned rich feature
 963 representations for a wide range of images. The best (and conceptually the only) way to
 964 load a pre-trained model is to load the ordered set of weights that define the network and
 965 to initialize an empty module with those values. This is the perfect start for creating a
 966 style transfer network, which requires a further and briefer training phase, to completely
 967 customize the network.

968 The key ingredient for finalizing the model is to insert some little but fundamental
 969 modifications and to extend the training on the pair of input images. This final training
 970 should be aware of the *concepts* of the visual style and visual content of the image, and
 971 the operation should try to preserve them both. This is usually done minimizing two
 972 new loss function, computed between the staring image and the produced image:

973 Content Loss

974 The content loss is a function that represents a weighted version of the content
 975 distance for an individual layer. The most commonly used function to evaluate the
 976 preservation of content between two images is the simple Mean Squared Error as



Figure 2.9: Different examples of application of a style-transfer NN on the same content image, with different style images, from [18]. The original picture depicts the Neckarfront in Tbingen, Germany (TOP-LEFT). The painting used as style image shown in the bottom left corner of each panel are in clockwise order: • The Shipwreck of the Minotaur by J.M.W. Turner, 1805 • Femme nue assise by Pablo Picasso, 1910 • Composition VII by Wassily Kandinsky, 1913 • Der Schrei by Edvard Munch, 1893 • The Starry Night by Vincent van Gogh, 1889.

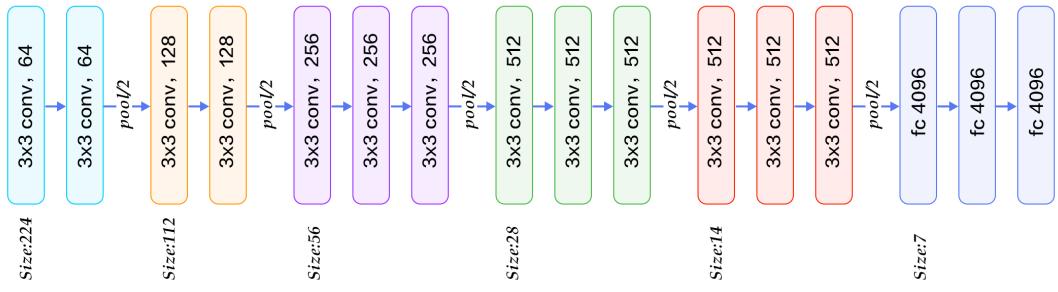


Figure 2.10: The structure of the VGG 19 network. It is for its most convolutional NN with 224×224 input size and with some downsampling layers which reduce the first two dimensions of the tensors along with the information flux of the network. At the very end of the architecture, there are three subsequent fully connected layers, responsible for the actual classification based on the features extracted from the previous layers.

977 in equation (1.5). It can be computed between any couple of same-sized object,
978 hence also between the results of the same feature maps on the images X and C
979 at the same layer L :

$$L_{Cont} = \|F_{XL}F_{CL}\|^2. \quad (2.17)$$

980 In order to evaluate this content loss, it is necessary to insert a custom transparent⁵
981 layer directly after the convolution layer(s) that are being used to compute the
982 content distance.

983 Style Loss

984 The concept of *style* loss function is the true novelty introduced by [18]. This loss
985 function is implemented similarly to the content loss module, as it will act as a
986 transparent layer in the network. The computation of the style loss requires in
987 advanced the evaluation of the Gram matrix G_{XL} at a certain layer L . A Gram
988 matrix is a result of multiplying a given matrix by its transposed matrix. In this
989 case, the matrix to multiplicate is a reshaped version of the feature maps F_{XL} :
990 \hat{F}_{XL} , a $K \times N$ matrix, where K is the number of feature maps at layer L and N
991 is the length of any vectorized feature map F_{XL}^k . Furthermore, the Gram matrix
992 must be normalized by dividing each element by the total number of elements in the
993 matrix. The style distance is now computed using the mean square error between
994 G_{XL} and G_{SL} :

$$L_{Style} = \|G_{XL}G_{SL}\|^2. \quad (2.18)$$

995 After the appropriate insertion of the loss-function evaluator layers, one last piece for
996 finalizing the model is the right choice of the gradient descent optimizer. As in [18] and
997 according to the Deep Learning community the optimizer which suite best this role is the
998 Limited Memory-BFGS [8],[39]. L-BFGS is an iterative algorithm in the family of quasi-
999 Newton methods that approximates the Broyden-Fletcher-Goldfarb-Shanno algorithm
1000 (BFGS) using a limited amount of computer memory, and it is a popular choice when
1001 estimating parameters of a non-linear differentiable scalar function.

1002 The final phase of the training process thus makes run the optimizer for hundreds of
1003 epochs on X , C , and S , and reduce the two loss functions values acting on the network's
1004 parameters. After the fine-tuning of the weights, the hybrid image is produced, as in
1005 Figure 2.9.

1006 2.8 Working Environment

1007 In this section I will briefly describe the machine I used to develop my project and the
1008 working environment I built. All the work has been done on my personal computer,

⁵A transparent layer is a layer that performs some operations, like evaluating a function on its input, but returns as output an unchanged copy of its input.

1009 mounting a **GNU/linux** operating system, in particular 18.04 LTS **Ubuntu** version. The
1010 computer mounts an Intel i7 core, 8 Gb of RAM beside an 2Gb NVidia 940MX GPU.
1011 All the Python libraries have been installed and harmonized in a virtual environment
1012 mounting **Python 3.7.6**. All the code produced during the development, the images,
1013 and the data produced have been collected in a devoted repository on GitHub [14], which
1014 is freely available.

1015 As a conclusion for this chapter I will re-collect all the references to the different
1016 Python libraries I used during the development of this work:

1017 **NumPy:** NumPy is the pillar of every scientific computation-oriented library. Is the most
1018 spread library for heavy multidimensional numerical computation, and it offers a
1019 broad variety of tools like random number generators, and pre-implemented linear
1020 algebra utilities [30].

1021 **SciPy:** The SciPy library is one of the core packages that make up the SciPy stack.
1022 It provides many user-friendly and efficient numerical routines, such as routines
1023 for numerical integration, interpolation, optimization, linear algebra, and statistics
1024 [45]. Two modules in particular from this library have covered an essential
1025 role in this project: the `SciPy.spatial.Voronoi` module for the computation
1026 of the 3D Voronoi decomposition, as mentioned in section 2.3, and the
1027 `SciPy.spatial.ConvexHull` module for the computation of 3D and 2D convex
1028 hulls (section 2.5).

1029 **PyTorch:** PyTorch is a rich ecosystem of tools and libraries geared toward Machine
1030 Learning and Deep Learning. The application of the style-transfer NN described
1031 in section 2.7 has required the use of this framework [31].

1032 **SALib:** The SALib is a library which collects many tools for the Sensitivity Analysis of
1033 parameters. In particular the `SALib.saltelli` submodel was used for the quasi-
1034 random numerical sampling in a three-dimensional box, and it has been described
1035 in section 2.4.

1036 **pnoise2:** pnoise2 contains many tools for the production of specific types of noise.
1037 The module `noise` was tweaked for the production of two-dimensional Perlin noise
1038 surfaces, and it has been introduced in section 2.6.

1039 **pyquaternion:** The `pyquaternion` library provides a framework for handling quaternions.
1040 It has been widely used in section 2.1 for the design of three-dimensional
1041 ramifications for handling multiple spatial rotations.

1042 Chapter 3

1043 Tissue Models Development

1044 The main goal of the present work, as stated before, is to recreate a three-dimensional
1045 virtual model of histological tissue as faithfully as possible and then, to perform planar
1046 sectioning on it to emulate virtually the traditional histological specimen preparation
1047 procedure. The creation of a model of such complex structures is definitely a high-
1048 level problem, and it has required a careful designing, made of subsequent stages of
1049 improvements. In section 3.1, I will describe all the necessary steps to create the model
1050 of a small region of pancreatic tissue, while in section 3.2 I will expose the steps I
1051 followed to build a model of dermal tissue. In section 3.3, instead, I will show the
1052 resulting synthetic images from the sectioning process performed on both the models
1053 and all the enrichments and processing necessary to give them the most realistic look I
1054 was able to recreate.

1055 3.1 Pancreatic Tissue Model

1056 The Pancreas is an internal organ of the human body, part of both the digestive system
1057 and the endocrine system. It acts as a gland with both endocrine and exocrine functions,
1058 and it is located in the abdomen behind the stomach. Its main endocrine duty is the
1059 secretion of hormones like insulin and glucagon which are responsible for the regulation of
1060 sugar levels in the blood. As a part of the digestive system instead it acts as an exocrine
1061 gland secreting pancreatic juice. The majority of pancreatic tissue has a digestive role,
1062 and the cells with this role form clusters (*acini*) around the small pancreatic ducts. The
1063 acinus secrete inactive digestive enzymes called zymogens into the small intercalated
1064 ducts which they surround, and then in the pancreatic blood vessels system [26]. In
1065 Figure 3.1 is shown a picture of the pancreas, with its structure and its placement in the
1066 human body.

1067 All the tissue is actually rich in other important elements as the islets of Langerhans,
1068 and sporadic connective tissue all over the structure, which are clearly visible in the

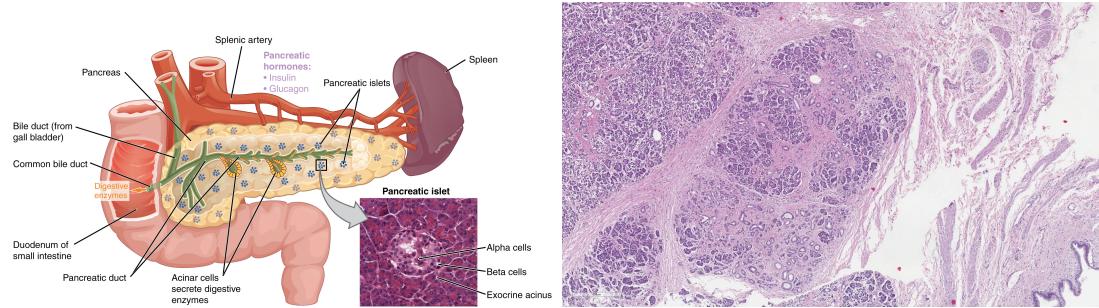


Figure 3.1: (left) A picture of pancreas' structure in its phisiologiacl context. In this picture is clearly visible the macroscopic structure and the galndular organization at microscopic level, and how it reflects in the histological sample. (right) A real pancreatic tissue sample with H&E staining.

1069 traditional histological specimens. In this first attempt of modelization from scratch
 1070 this second layer of complexity has not been already considered, and the main focus
 1071 has been to reflect only the main structural features on the virtual specimens. Given
 1072 pancreatic tissue's organization the first features I decided to put emphasis on were: 1)
 1073 The iterative (with a fractal-like behavior) ramification of blood vessels for the irrigation
 1074 of glandular acinus, 2) The space-filling distribution of acinus in the tissue, in fact, we
 1075 expect a homogeneous density in the organ and to not see *holes* at all inside it. In this
 1076 section I will describe step by step all the process I followed to create the model of a
 1077 portion of pancreatic tissue, and all the interesting pitfalls I overcame.

1078 3.1.1 2D Ramification

1079 The first step was took in two dimensions, and it was the choice of the right *structure*
 1080 to emulate the ramification of blood vessels in pancreatic tissue. The choice fell on a
 1081 particular parametric L-system, as the one shown in Figure 2.1, in section 2.2. This
 1082 structure is made of an iterative bifurcation of gradually shorter segments, with an angle
 1083 of $\pm 85^\circ$ respect the main direction. For a start I added some features to give a more
 1084 realistic look to the structure, which are all well represented in Figure 3.2:

- 1085 • A progressive thickness of the bifurcation's segments, starting from a thick main
 1086 branch that dwindle every junction. The idea is that the main blood vessel be-
 1087 comes gradually smaller becoming capillaries for single-cell irrigation.
- 1088 • A progressive randomness in the angular deflection at every fork. Perfectly repeated
 1089 angles are almost nonexistent in nature, so I decided to introduce an increasing
 1090 indetermination in the angle of bifurcation from the main branch to the free ends
 1091 of the structures' branches.

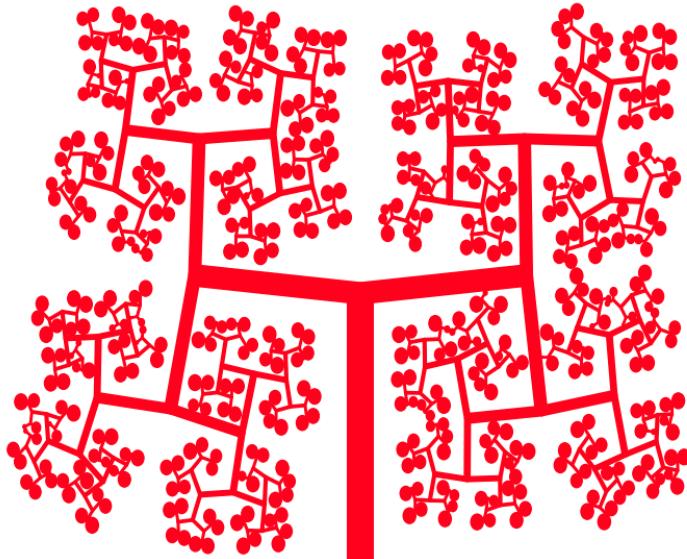


Figure 3.2: The development of the simple ramification in Figure 2.1, with some features to give it a more realistic look, like progressive thickness, angular noise in bifurcation and spheres at free ends of the ramification. The image is made using the tools exposed in section 2.2.

- Spheres at the ends of each branch, which acts as glandular acini. The maximum radius is comparable to the length of the final segments.
- A mechanism to avoid self-superimposition between branches and spheres. After the insertion of noise, the cumulative effect on the final segments might lead different branches to intersect. This is clearly a paradoxical situation, as real tissues while growing naturally occupy the space in a gradual way.

To produce the specific image in Figure 3.2 I used a particular setting of the tool described in section 2.2, which have a greatly wider range of customization and could be used to create many other different structures to the need.

3.1.2 Expansion to 3D

The successive step I followed was to expand this structure in three dimensions and fill the space in each of the three directions. The idea to evolve the structure in Figure 3.2 is simply to twist of 90° the ramification at every junction point, in such a way to exit the previous belonging plane. However, putting into practice this development has not been easy. The organization of the structure in a 3D space requires an appropriate system of reference for handling subsequent rotations in three dimensions. The best option for

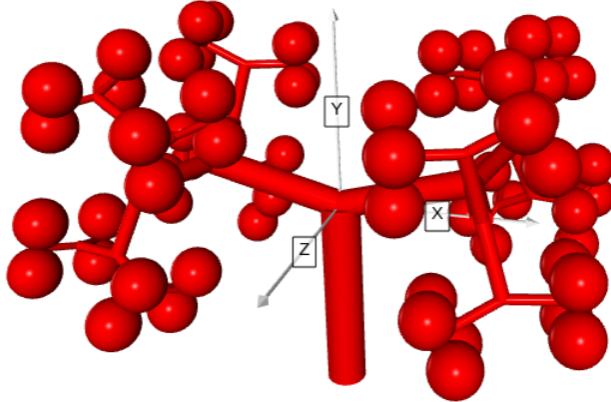


Figure 3.3: The three-dimensional expansion of the 2D ramification in Figure 3.2.

1108 handling relative 3D rotations, often used in computer graphics and every kind of 3D
 1109 modelization, are quaternions, as shown in section 2.1.

1110 In this new structure, segments are replaced with cylinders, and circles are replaced
 1111 with spheres. At every bifurcation to every cylinder are applied the following transfor-
 1112 mations:

- 1113 • a contraction in its extensions, regulated by an adjustable parameter R .
- 1114 • the usual deviation of $\pm 85^\circ$ respect to the direction of the parent branch.
- 1115 • a 90° specific rotation along the axis of its parent branch.

1116 The result of this procedure is a 3D ramification like the one in Figure 3.3, in which
 1117 we can recognize a good coverage of the space defined by the structure's boundaries and
 1118 immediate relation with the 2D structure in Figure 3.2. It should be noted that, in the
 1119 further refinements of the model from now on, there won't be present the progressive
 1120 angular indetermination on the direction of branches. Although it is a feature already
 1121 implemented and working, it requires efficient control to avoid reciprocal overlapping
 1122 between elements to produce a realistic structure. This second element has not been
 1123 already developed and it would certainly enrich the representative power of the model.

1124 As for the 2D ramification the production of this structure has required the imple-
 1125 mentation of a tool for the 3D generation with a greatly wider power, able to produce
 1126 almost any type of three-dimensional iterative structure after the right adjustment, and
 1127 with a high degree of customization. It is necessary to mention the fundamental tool
 1128 which allowed me to accomplish this step of the development, which is the Python library
 1129 VPython: a library for 3D graphics visualization. This library allows a convenient and

1130 powerful interface to draw many types of objects and to move them around in space,
1131 which has been priceless to orient my self in three dimensions while developing the model
1132 and to produce all the 3D images visible in this work.

1133 3.1.3 Subdivision in Cells

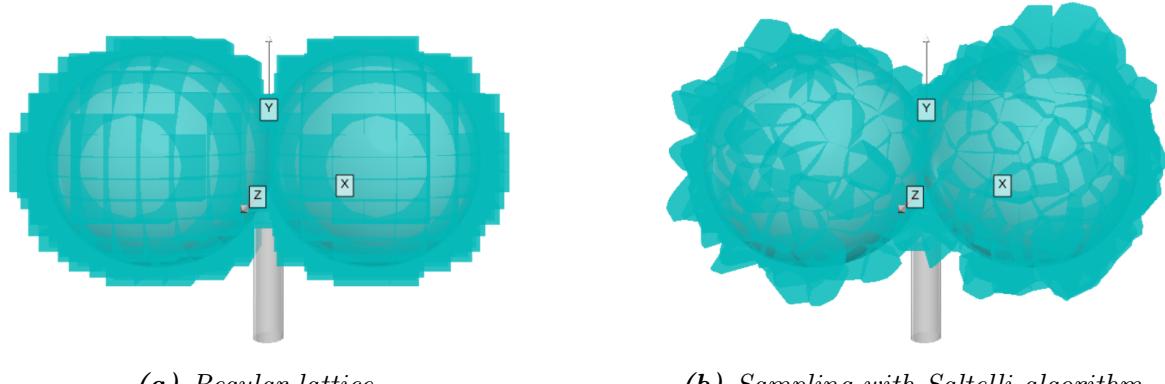
1134 Once the 3D backbone of the pancreatic tissue blood vessels ramification system has
1135 taken shape, the next step was to embed all this structure in a spatial partitioning
1136 process, to create the subdivision into single cells. To perform this important task I
1137 used a 3D Voronoi decomposition, as shown in section 2.3. Depending on the choice
1138 of the starting points, the Voronoi tessellation could be an excellent item to recreate
1139 individual cells because it could guarantee some important properties: all the regions
1140 are convex, adjacent, with similar size and volume, with different shapes, and without
1141 holes. These have been chosen as the most significant properties to be reflected in the
1142 first modelization of cells.

1143 As shown in section 2.3, the Voronoi decomposition strongly depends on the choice of
1144 the starting point. Points spread uniformly on a 3D regular lattice will produce a series
1145 of parallelepipeds repeated in the space. An example of uniform tessellation is shown in
1146 Figure 3.4a. On the other hand, a decomposition based on a quasi-random generated
1147 point can present all the good properties we mentioned before, including the diversity in
1148 shapes. In Figure 3.4b is shown an example of a Voronoi decomposition based on points
1149 sampled in a 3D with the Saltelli algorithm, in reference to section 2.4. Regardless of
1150 the points sampling technique, the boundaries of the sampling 3D box have been chosen
1151 to loosely contain the ramification.

1152 There are tough some delicate considerations to be highlighted about the decomposi-
1153 tion procedure. The first regards the most external pieces of the decomposition. Whilst
1154 the internal pieces are neatly bounded and defined, the most external layer instead is
1155 made on unbounded regions, which extend themselves to infinity. Those regions have
1156 clearly to be rejected, as it would be absurd for a cell to have an infinity volume. Typi-
1157 cally those unbounded regions are resized in order to adhere to some limiting boundaries,
1158 with an operation known as *cropping*. In Figure 3.5 is shown an example of circular crop-
1159 ping in a 2D Voronoi decomposition: all the regions which intersect the circumference
1160 have to be resized.

1161 The cropping operation in 3D is extremely complex, tough. Thus, a more simple and
1162 efficient, yet less elegant, technique has been used. Instead of resizing the regions which
1163 lie on the boundaries of the sampling region, those regions have directly been rejected.
1164 This process is really fast and it does not lead to any danger of representativity loss if
1165 the boundaries are loose enough and if the density of sampling is not too low.

1166 The other important consideration regards the density of sampling points. Increasing
1167 the number of points to be extracted from the same volume automatically the number
1168 of cells in the box will rise, and in contrast, their relative dimension will decrease. This



(a) Regular lattice.

(b) Sampling with Saltelli algorithm.

Figure 3.4: Comparison between two Voronoi decompositions. The first (left) is created from a regular lattice of starting points, and every piece is exactly equal to all the others, creating a regular subdivision of the space. The second (right) is created instead from a sampling made following the Saltelli quasi-random algorithm. The pieces are all different in shape, but they all have similar sizes and volumes. In this pictures in particular have been shown only the pieces of the tessellation which lie in correspondence to the boundaries of the spheres underneath. While watching this picture one should immagine the decomposition extended similarly in all the space around the ramification, within certain boundaries, which loosely contains the structure. This limitation was necessary to enhance the interpretability of the image.

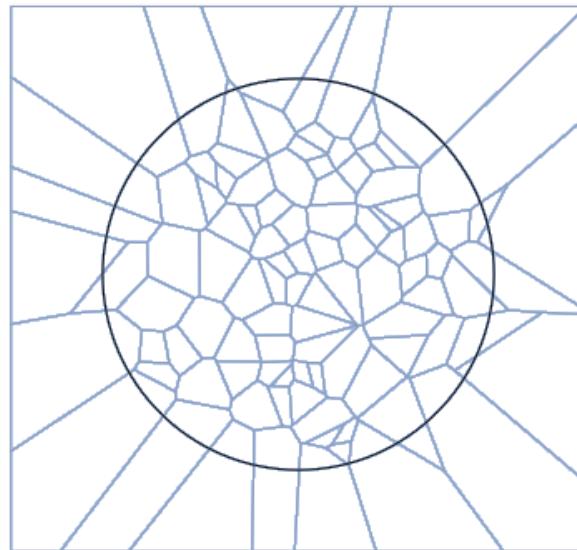


Figure 3.5: Example of circular cropping in a 2D Voronoi decomposition: all the regions which intersect the circumference have to be resized.

is a key element of the model: a too rarified decomposition would not be able to reflect the complexity of the structure underneath, but a too crowded decomposition on the other side would lead to an unrealistic dimension of the cells in the tissue. Furthermore, this parameter has a huge influence on the computing time necessary to generate the model and to process it for the sectioning as will be shown in section 3.3. In almost all the applications so far, the density parameter has been tuned by eye, with a trial and error procedure. Although, a more rigorous way to adjust this parameter would be to consider the average dimension of the cells and make some microanatomical considerations to define the correct relative dimensions. The measure of the volume¹ of the decomposition's regions is an accessible parameter, thus an easy way to estimate the average linear dimension of the cells can be to approximate all the cells to cuboid seeing their volumes as $V \approx L^3$. Averaging all the L measures an estimate \hat{L} can be done. This average length may be compared to the length of the blood vessel ramification, allowing a good reference tool.

3.1.4 Cells Identity Assignment

The great power of creating all the models virtually is to know exactly the identity of every point in the structure. Although, this identity has to be reflected at the cellular level, assigning to every region a label. Imagining the Voronoi decomposition represented in Figure 3.4b extended to the entire box containing the ramification, good discrimination would distinguish three classes of cells: those which lie completely inside a sphere, those which lie completely outside a sphere, and those which lie on the boundaries of a sphere. In Figure 3.6 is shown a portion of the complete decomposition where the three classes of cells are reported with different colors: the internal cells in red, those on the boundaries in turquoise, and the external cells in gray.

In this particular case to find the relative position between every sphere in the structure and each cell it has been used a test on the proximity between the spheres' centers and the vertices of every polyhedral cell. If all the vertices of a region lie within a distance lower than the radius from the center of the same sphere then that region can be said to be an internal one. If none of the vertices lie within the radius distance from any center then that region is said to be external. In any other case, the region is said to be on the boundaries of some sphere, and this third label is assigned to it. As could be imagined the number of cells inside the volume can grow very quickly, and in the more rich ramifications also the number of spheres could be high. If we think that any polyhedron has a number of vertices of the order of 20/30 then it is clear that the number of distance evaluations could grow very quickly, requiring some relevant computational power in the more extended simulations. In order to optimize this computation, I decided

¹The volume is expressed in the same arbitrary length unit of measures used during the ramification structure. This allows a coherent reference tool.

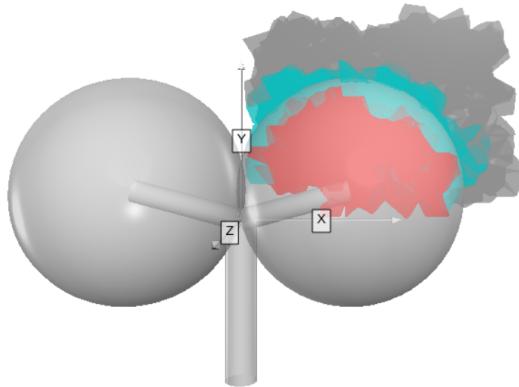


Figure 3.6: Portion of the complete Voronoi decomposition, showing the three different classes of cell in three different colors: the internal cells in red, those on the boundaries in turquoise and the external cells in gray.

1205 to use a python implementation of a K-dimensional Tree, which is a space-partitioning
 1206 data structure especially suited for fast and optimized computation of distances [6]. A
 1207 K-d Tree is an algorithm that iteratively binary splits the space: every node of the three
 1208 could be thought as a splitting $(k - 1)$ -hyperplane dividing the space into two semi-
 1209 hyperspace. The result is an optimized algorithm for repeated distance evaluations. As
 1210 for many other tools, in my code I used a pre-implemented module `KDTree` from the
 1211 `Scipy` library.

1212 This procedure of labeling the regions is completely customizable, and it should be
 1213 adapted to the specific application. By the way, the principle will always be to perform
 1214 some sort of spatial consideration respect to the primary structure and assign all the
 1215 interesting labels accordingly to the cells in the volume.

1216 After labeling the cells in the decomposition the model is considered complete. Every
 1217 enrichment to the structure should be reflected in some type of label for the cells, which
 1218 are chosen as the fundamental unit in the model. As we will see in section 3.3 during
 1219 the sectioning process in the produced image will be printed mainly the identity of the
 1220 cells, hence any detail on a finer scale in the model would not be conveyed properly on
 1221 the final image.

1222 3.2 Dermal Tissue Model

1223 Skin is the layer of soft, flexible outer tissue covering the body of a vertebrate animal,
 1224 with the three main functions of protection, regulation, and sensation. Mammalian
 1225 skin is composed of two primary layers: the epidermis, which provides waterproofing
 1226 and serves as a barrier to infection, and the dermis, which serves as a location for the

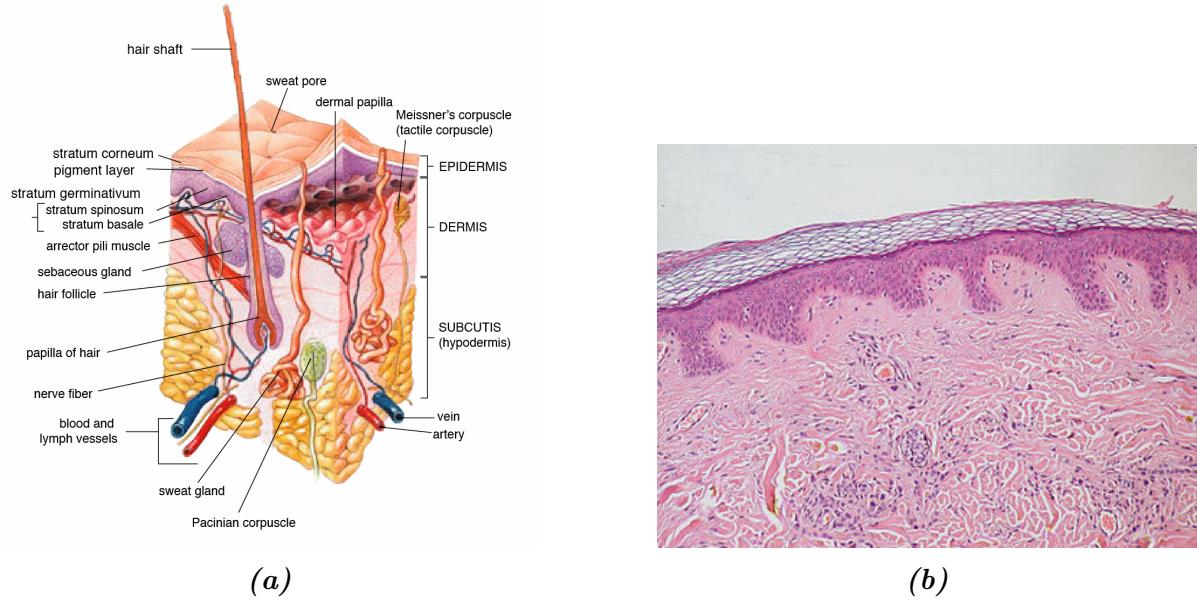


Figure 3.7: (left) Microanatomical description of a region of dermal tissue and all the interesting elements present in cutis, and subcutaneous layer. (right) An actual histological specimen from a sample of dermal tissue.

appendages of skin. The epidermis is composed of the outermost layers of the skin. It forms a protective barrier over the body's surface, responsible for keeping water in the body and preventing pathogens from entering, and is a stratified squamous epithelium, composed of proliferating basal and differentiated suprabasal keratinocytes. The dermis is the layer of skin beneath the epidermis that consists of connective tissue and cushions the body from stress and strain. The dermis provides tensile strength and elasticity to the skin through an extracellular matrix composed of collagen fibrils, microfibrils, and elastic fibers, embedded in hyaluronan and proteoglycans.

The modeling of the dermal tissue has followed analogous schedule respect to the previous model, hence many procedures and considerations have been repeated. The main target for this model was to recreate the stratification of different specific tissues in the section of a dermal sample. As clearly visible in Figure 3.7b in a dermal histological specimen one can distinguish the lighter and wider region of proper *dermal* tissue, underneath a more shallow and darker region of *epidermal* tissue. It is very interesting the boundary between those two regions, which can be seen as an irregular and smooth surface, populated of dermal lobes. On the other side of the epidermal layer lies a layer of *keratin*, with a smooth and regular boundary surface. Keratin is a family of fibrous structural proteins known as scleroproteins, a key structural material for hair, nails, and the outer layer of skin. The upper white region in Figure 3.7b instead is the support for the samples to perform optical analysis, and has no histological meaning.

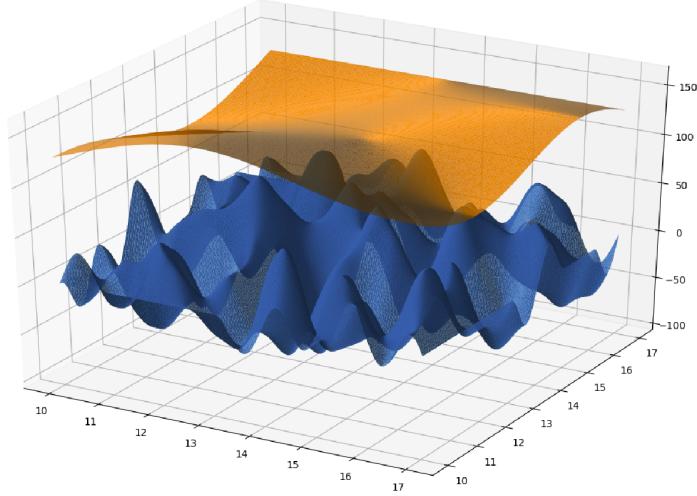


Figure 3.8: Pictures of two different Perlin noise surfaces used to separate dermal from epidermal tissue (blue) and epidermal from keratine layer (orange). The two surface are made by the same Perlin noise function, but the latter is stretched and compressed in order to have a more regular behavior.

1) Stratified Structure

In order to represent faithfully the stratification of different tissue layers, I decided to use one flat plane to separate epidermal tissue and the keratine layer and two boundary surfaces modulated by a Perlin noise function on different scales for the other two separations, as shown in Figure 3.8. As stated in section 2.6, after some easy customization the generation of different Perlin noise surfaces is easy and straightforward. By the way, to achieve the regularity and the smoothness of the orange surface in Figure 3.8 it was needed a horizontal stretching.

Following the scale of the image, the standard blue surface is created in a 7×7 square, while the orange surface has primarily been generated in a 1×1 square, and then has been stretched to cover the same 7×7 square, multiplying the values in its grid points respectively for the stretching factors $R_x = R_y = \frac{1}{7}$. In this primary structure, there are many important parameters defining the surfaces, like the distance between the two surfaes' average values and the amplitudes of the peaks and the valleys of the surfaces. In its standard version the Perlin noise covers the $[-1; 1]$ range, but with a simple multiplication for an amplitude factor A_S the values can be adjusted. Those particular values have been adjusted after some tries to recreate the proportions typical of a real specimen. To sum up, each one of the two surfaces is stored as a discretized three-dimensional array, or better as an array of 3-tuples in the form $(x, y, f_{(x,y)})$, one tuple for every (x, y) node of the grid, while the discretization grid was the same for both the surfaces.

1268 **2) Subdivision in Cells**

1269 The subdivision in cells of the volume containing the structure has followed the
1270 exact same steps described in section 3.1, hence in this paragraph I will shortly
1271 resume the process. The first step is the definition of a suitable volume containing
1272 the structure. Then is the time for the generation of the decomposition's starting
1273 points according to a quasi-random number generation technique, as described in
1274 section 2.4. Afterward, the points are used as a base for the decomposition, and
1275 all the cells with undefined boundaries are rejected

1276 **3) Cells Identity Assignment**

1277 The identity assignment procedure instead has been customized for this particular
1278 application. In this model there are no *boundary* cells as the one lying on the
1279 spherical surfaces in the pancreatic model, thus there is no need for a test on the
1280 position of each vertex of every cell. In this model, the starting point for every
1281 Voronoi region has been used as a reference, and its relative position respect to the
1282 boundary surfaces was the discriminating factor for assessing the identity. In order
1283 to perform a coherent test on the relative position between the regions' center and
1284 the boundaries surfaces the positions of all the centers had to be discretized on
1285 the same grid onto which were defined the surfaces. The comparison with the flat
1286 horizontal plane defining the boundary between epidermal tissue and the keratine
1287 layer instead was simply a test on the z coordinate of the point: $z = f_{(x,y)} \leq \hat{z}_{plane}$.
1288 The result of this procedure is that every region is assigned a label corresponding
1289 to the belonging tissue layer: *D* for dermal, *E* for epidermal, *K* for keratine, and
1290 *V*, which stands for *void* for the white empty space above the sample.

1291 In this model, as in section 3.1, after the assignment of the identity to all the cells in
1292 the volume the modelization process is considered complete. Any sort of improvement
1293 and enrichment should be inserted during the structure designing phase and should be
1294 linked to an identity label.

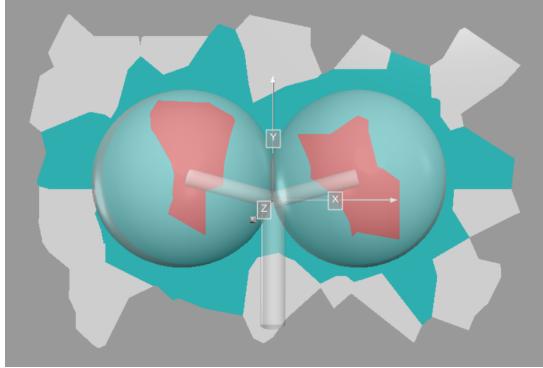
1295 3.3 Synthetic Images Production

1296 After the model is complete we have a three-dimensional representation of the tissue
1297 under study. The aim of the work is though to produce synthetic images from that
1298 structure, more precisely we want pairs of images composed of the synthetic-histological
1299 images and its related segmentation mask. The transition from 3D structure to a 2D
1300 image is the last step in the process, and it is inspired by the actual traditional technique
1301 for the preparation of the histological specimen, as described back in section 1.1.1. As
1302 the biopsy sample is treated and then sectioned with the microtome, the virtual model
1303 is sectioned in a random direction, producing an image representing the slice. This
1304 first image contains all the information of the section but its appearance is completely
1305 arbitrary and its look has nothing to share with a realistic sample. The original slice
1306 then acts perfectly as a segmentation mask, but some careful and dramatic makeover is
1307 needed to produce the final realistic looking image. In this section I will describe the
1308 general procedure to produce virtual slices from the two 3D virtual models described
1309 before in section 3 and the technique used to edit the images and give them the desired
1310 appearance.

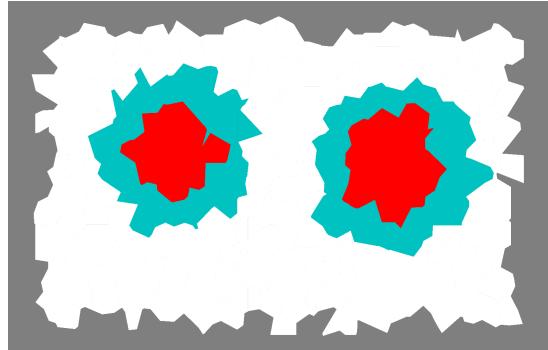
1311 3.3.1 Sectioning Process

1312 For any model created following the general procedure described in 3, even more so for
1313 the two particular models of pancreatic and dermal tissue, the sectioning process will
1314 be almost the same, and it will rely mainly on the algorithm for the general section of
1315 polyhedron described back in section 2.5. As stated before the models are essentially
1316 composed by labeled polyhedron spatially organized in a 3D volume. The ordered section
1317 of each polyhedron will yield all the polygons that shall be assembled in the final section.
1318 In Figure 3.9a is shown the three-dimensional representation of the section of a simple
1319 ramification, as the one in Figure 3.6. All the polygons that compose the section are
1320 drawn with the color correspondent to their label, following the same idea of Figure 3.6.
1321 In Figure 3.9b is printed the final result of the sectioning algorithm applied to the model,
1322 which will be the segmentation mask in the single pair of synthetic images. The colors
1323 in the produced slice match the colors used for the different identities in the 3D model.

1324 The simplest, yet over-abundant, way to proceed is to create the model in its entirety
1325 and subsequently choosing the sectioning plane. Afterward, it is necessary to select
1326 only the regions that intersect the plane and section them all. Actually, the test on the
1327 intersection passes through the check on the relative position of the polyhedron's vertices
1328 respect to the sectioning plane: if all the vertices lie on the same semi-space then the
1329 intersection would be null and the polyhedron is not of interest for that particular section.
1330 This procedure is exactly the first step of the algorithm in 2.5, thus the filtering on the
1331 regions is actually made during construction for optimization. The alternative method
1332 could be to choose in the first place the direction of the sectioning plane, and in second



(a) 2D polygonal sections represented in a 3D environment.



(b) The final section image, produced directly on a planar picture, skipping completely the 3D representation.

Figure 3.9: In this Figure is shown the idea of the correspondence between the section of a simple structure in the space and the correspondent section. The correspondence is not perfect for representation requirements, in fact the two images even if very similar are produced with two completely different methods. At the left an image of 2D polygonal section embeded in a 3D space, made using 3D visualization tools. At the right a simple image produced printing the polygons in a planar picture. Printing 2D polygons in a 3D space is much more complicated than one would think using the same tool used to produce the other images like Figure 3.6 and 3.4. This choice has been done for the sake of the overall homogeneity in pictures style.

1333 place to generate the model's decomposition only in the volume adjacent to that plane.
 1334 This method enables the sparing of a good amount of computation, without any negative
 1335 impact on the final result. The only delicate step is the choice of a wide enough region
 1336 of space around the plane, which doesn't compromise the representation.

1337 As a guarantee for richness and diversification among the images, there is the need
 1338 for some degree of controlled randomness in the sectioning process, for example in the
 1339 determination of the sectioning plane direction. All the sectioning process is then based
 1340 on a single starting *seed*, which determines the direction of the sectioning plane in a
 1341 deterministic way, and all the rest of the model is generated as a consequence. In this
 1342 way, all the possible angulations are equally probable and will be sampled in view of
 1343 multiple applications of this process. In Figure 3.10 are shown two different sections,
 1344 along two random planes on two simple ramified structures with four spheres.

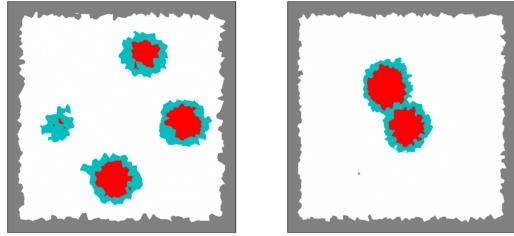


Figure 3.10: Two different section, along two random planes on two simple ramified structure with four spheres.

1345 3.3.2 Appearance Makeover

1346 After the application of the sectioning algorithm of the previous section, the image
 1347 which will act as a segmentation mask is ready. The last and most complex task that
 1348 remains is to transform the image and to give it a realistic look. I tried many different
 1349 transformations, more or less complicated, and there was not a final decision on which is
 1350 the best blend of them. In this section, I will describe them as an arsenal of possibilities
 1351 and show their impact on the images.

1352 Color Palette

1353 The first correction to do to the images will inevitably be a change in the colors
 1354 of the image. Gray, Turquoise, and Red are the perfect choice for label-colors but
 1355 act poorly as physiological colors. In Figure 3.11 is shown an example of an image
 1356 produced re-mapping the colors to a new palette, inspired to the coloring of the
 1357 real specimen in Figure 3.1 and 1.2, given by the traditional hematoxylin and eosin
 1358 staining process.

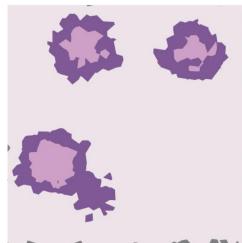


Figure 3.11: Example of images produced re-mapping the colors with a color palette inspired to a real H&E stained histological sample.

1359 Nuclei Projection

1360 Another fundamental processing needed was the projection of cells' nuclei on the
 1361 image. Usually, nuclei are clearly visible in histological samples and guide the

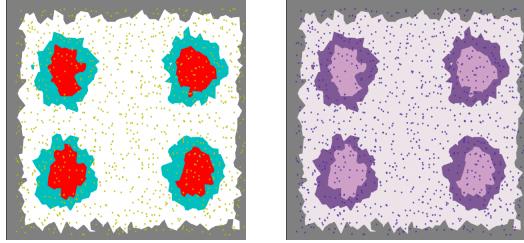


Figure 3.12: Nuclei projection on the image: (left) in yellow in the segmentation mask and (right) in purple in the image under makeover.

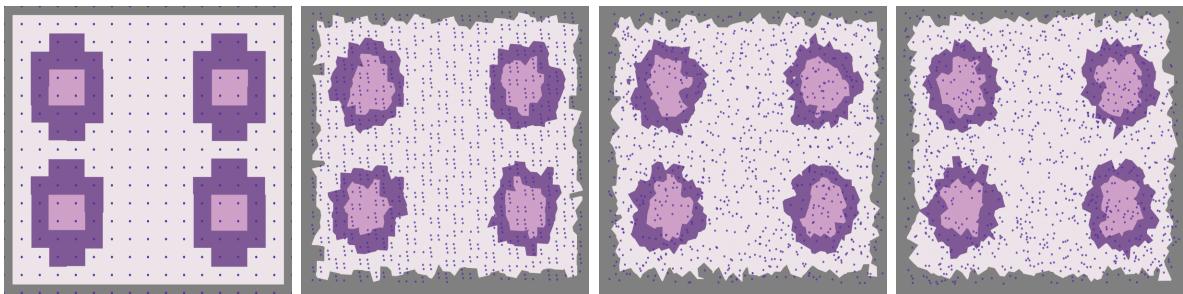


Figure 3.13: Four different sections produced with the same density on cells but with four different method for the sampling of starting decomposition's ponits (from left to right): • points sampled on a regular lattice, • sampling following a simple recursion sequence as the one in equation (2.12), • following the saltelli algorithm, • following a fully-random distribution.

analysis allowing to detect individual cells in the specimen. As a reference for the nucleus position the starting point of every polyhedral region has been used and projected on the sectioning plane as a little dark circle. The diameter of those circles as been chosen to be a submultiple (10%) of the linear estimated dimension \hat{L} of the cells in the decomposition².

Nuclei projection, among the other things, is an excellent tool to perceive the different effects obtained with different choices of quasi-random distributions or fully-random distributions (with reference to section 2.4). The different impact on the overall image is huge, and it really changes the overall sense of the image. In Figure 3.13 are reported four different sections, produced with the same density on cells but with four different methods for the sampling of starting decomposition's points.

Boundaries Projection

²Following the same logic of step 3 of section 3.1.

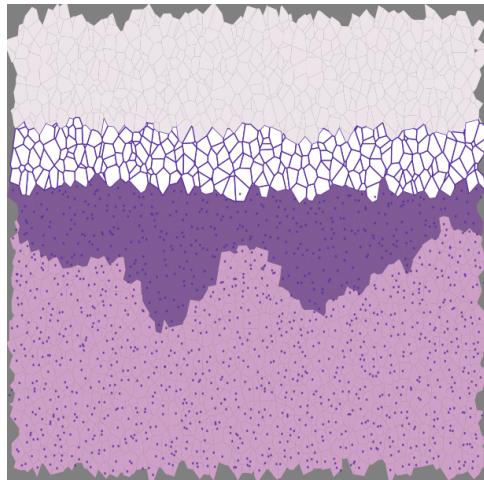


Figure 3.14: Example of images produced re-mapping the colors with a color palette inspired to a real H&E stained histological sample.

On the same wave of the previous tool, another operation that can help the appearance of an image is the projection of the boundaries of each (or just a part) of the polygonal sections. The drawing can be clearly tuned and customized depending on the specific necessities. In Figure 3.14 is shown an example of a section on the dermal tissue model in 3.2, in which the boundaries of all the cells have been lightly marked, and the boundaries of the cells in the keratine layer have been heavily marked instead.

Blurring Effects

In all the images produced so far the boundaries between polygonal sections are perfectly sharp and without any smudge. To give a more realistic feeling to those pictures I tried different forms of blurring. As a first try, I applied a Gaussian blurring filter, which is an extremely common blurring operation in computer vision, which consists of a simple discrete convolution with a 2D Gaussian kernel. The effect is a regular and diffuse blur all over the image, as in Figure 3.15a. The second blurring effect I implemented was based instead on the averaging of parallel and adjacent slices on the same model. This method is inspired by the real sectioning technique (section 1.1.1), in which every slice is not an infinitesimal layer of matter, but a finite sample, which suffers from mechanical dragging during the process. The idea is that the average between three or more slices equally spaced above and below the *main* slice should recreate a realistic blurring effect. An example of an image produced with this process is shown in Figure 3.15b.

Perlin RGB Noise

A further attempt to give visual texture to the image was done using again the

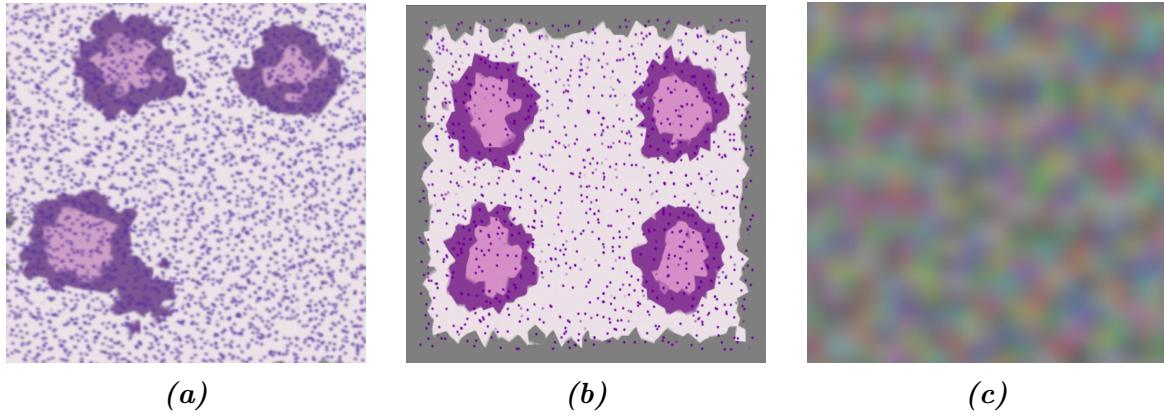


Figure 3.15: The two blurring effects used in this work: (left) A standard Gaussian-filter blur and (center) a specific blur introduced averaging adjacent parallel slices on the same image, (right) an example of RGB color noise built joining three different Perlin noise surfaces, one for each color channel.

Perlin noise, described in section 2.6. The idea is to create some fluctuation among the color channels of the image around the sharp values of the image produced by the sectioning algorithm. From a practical point of view, I created three different and independent Perlin noise surfaces, one for every color channel (Red, Green, and Blue), and added them to create an RGB noise on the image. An example of the resulting image is shown in Figure 3.15c.

Style Transfer

This last tool I will describe is the most sophisticated so far. It consists of the application of a style-transfer neural network (STNN) on the image obtained through the sectioning process, for the implantation of the visual texture from a real sample of the corresponding tissue. Style-transfer NNs, and their functioning, have been described in detail in section 2.7, and here I will cover just the particular applications on the two type of section produced.

The first manipulation I report is the one on a section from the pancreatic tissue model. The image of which to conserve the visual content is a section with some simple pre-processing picked from the ones described before (Figure 3.16a): a more accurate color palette, the projection of nuclei, and the average on five adjacent slices. The image from which to pick the style, thus the visual texture, is a portion of an actual histological sample of the pancreas, and it is shown in Figure 3.16b. The application of the STNN yields a hybrid image, shown in Figure 3.16c. The second application was made on a section on dermal tissue. The three content, style and styled images are shown respectively in Figure 3.17a, 3.17b, and 3.17c.

In Figures 3.16, and 3.17 are reported the best results among all the different tests

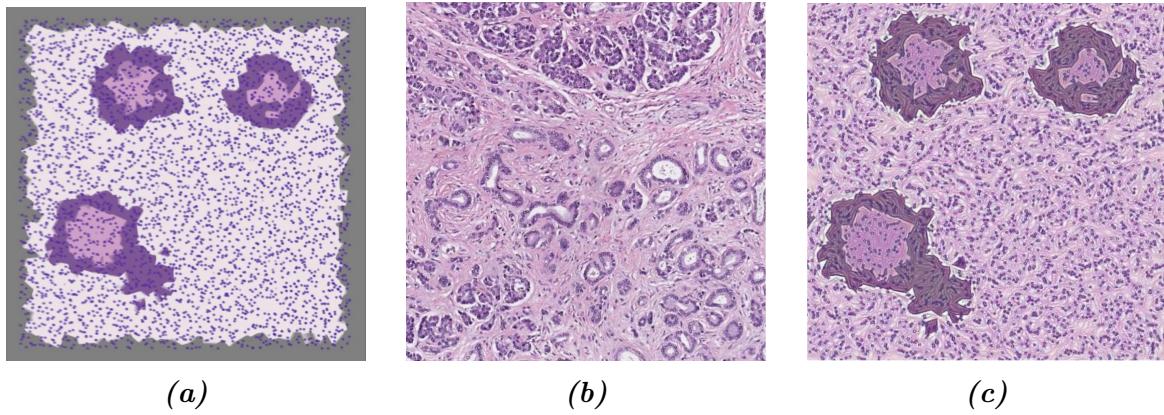


Figure 3.16: Application of the style-transfer NN on a section of the pancreatic tissue model: 3.16a the content image, 3.16b the style image, 3.16c the hybrid resulting image.

made on the sections. I made different tries on the same image with different processing before the manipulation with the STNN, to see the impact of the different adjustment on the resulting styled image. It turned out that the presence of nuclei is essential to give a homogeneous texture to the image and avoid unrealistic artifacts. On the other hand, the choice of the color palette has a way lighter effect than what one would think: the model yields almost the same result with a grey-levels image or with any other palette.

It is interesting to notice the timing cost of this style transfer operation. While all the other manipulation described in this chapter requires a very short time (seconds) to be applied, and are in practice *instantaneous*, the transfer of style is a way more robust operation, which implies the finalization of the training of a pre-trained neural network. On a computer with the technical specification described in section 2.8 this operation instead took minutes, which is a time two full orders of magnitude greater.

It should be noted that the presented results are obtained from the application of a pre-trained STNN model. The development of a specialized model for histological texture transfer could improve extremely the ability to produce realistic images, way further the present results.

One single complete application of the process consists then in the generation of a tissue model, in the sectioning along a random section plane, and in the processing of the image, in order to produce the pair of ground-truth image and the synthetic histological image. The target is to apply over and over this process to collect the necessary amount of images and constitute an entire dataset. An important feature to have for the process is thus a complete automatization, in order to scale up the

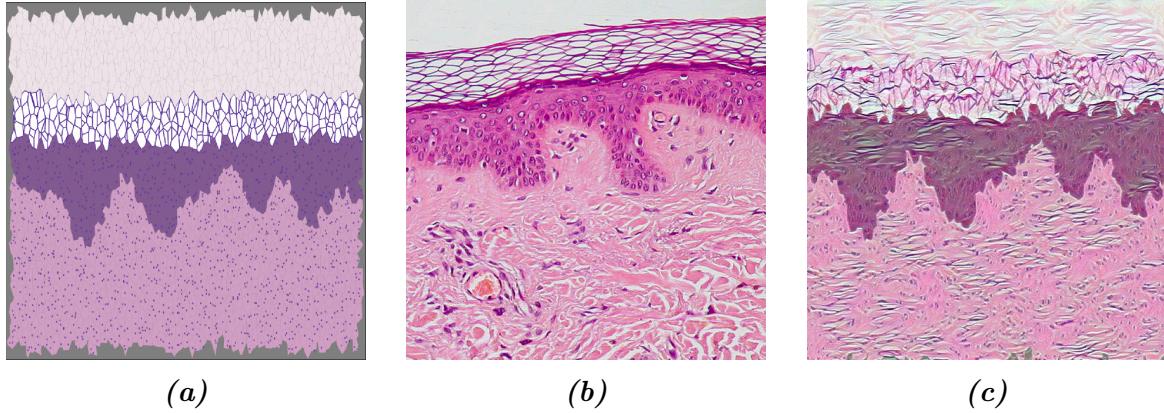


Figure 3.17: Application of the style-transfer NN on a section of the dermal tissue model: 3.17a the content image, 3.17b the style image, 3.17c the hybrid resulting image.

generation of images, possibly even in parallel computation. For this purpose, I created a pipeline workflow interface for the image generation, with an automatized harmonization of every piece of the process. The generation now requires just to fill a configuration file in which writes all the specific characteristics of the images: the type of structure, its features, the desired processing on the images, and eventually the random seeds for a supervised generation. In Figure 3.18 is reported a small scale example of a dataset produced with multiple automatized applications of the generation tool on a ramified structure inspired to a pancreatic tissue model. It is clear the correspondence between segmentation mask and synthetic histological images, and the diversification given by the supervised randomness on the generation.

The actual tool used for the set up of a working pipeline was the **Snakemake** [24] workflow management system, which is a python-based tool to create reproducible and scalable data analyses.

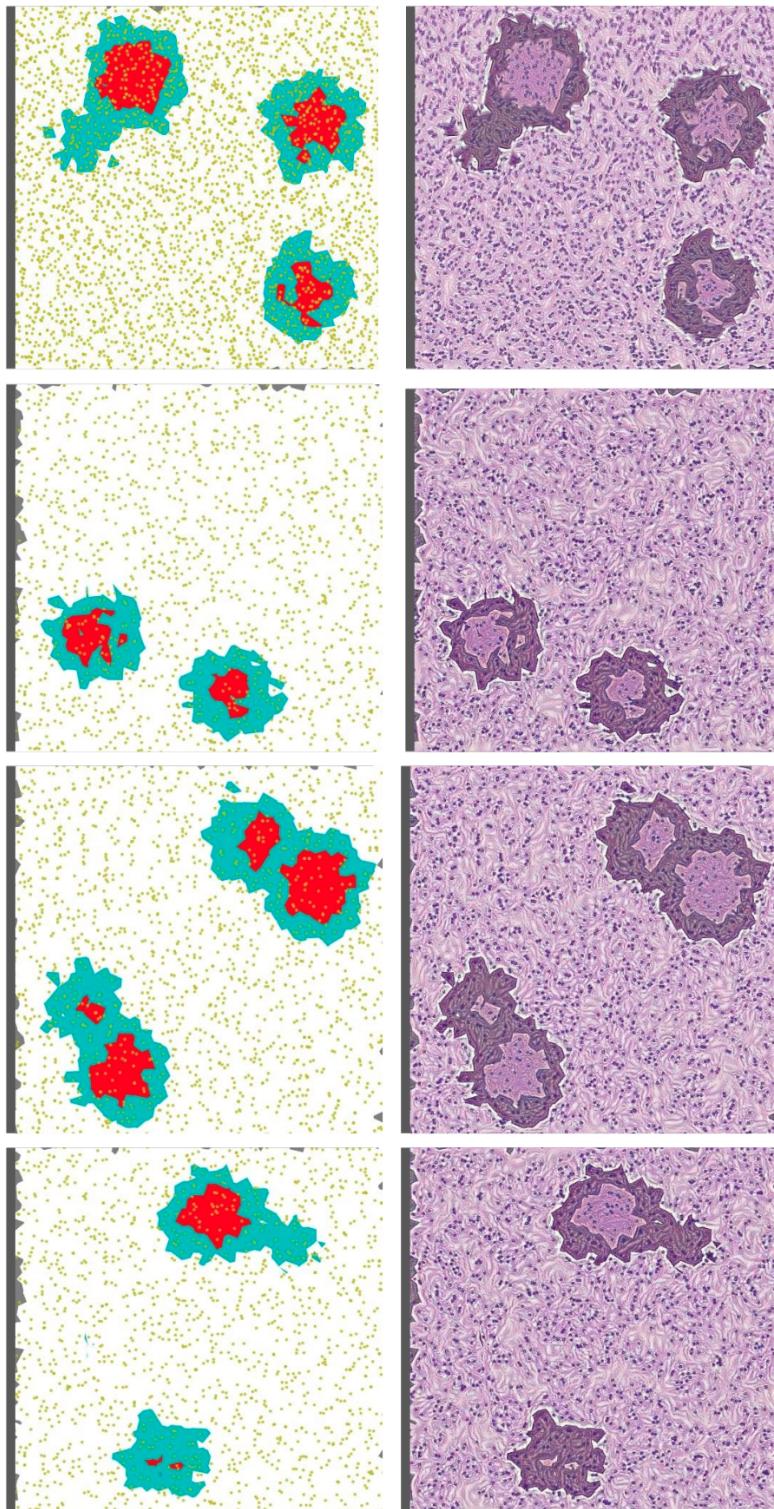


Figure 3.18: Small scale example of dataset produced with multiple automated applications of the generation tool on a ramified structure inspired to a pancreatic tissue model.

¹⁴⁵⁸ Conclusions

¹⁴⁵⁹ In this project, I face the problem of synthetic histological image generation for the
¹⁴⁶⁰ purpose of training Neural Networks for the segmentation of real histological images. The
¹⁴⁶¹ manual analysis of histological specimen is a complex, time consuming, and expensive
¹⁴⁶² task and nevertheless it is a pillar of countless diagnostic techniques. Any form of
¹⁴⁶³ support for this procedure hence is welcome and endorsed by the health-care system. In
¹⁴⁶⁴ particular, in this work, I focus on the problem of histological specimens segmentation.
¹⁴⁶⁵ The most advanced algorithms for image segmentation are based on Deep Learning
¹⁴⁶⁶ and requires the training of extensive and complex neural networks. One of the toughest
¹⁴⁶⁷ hurdles to overcome for the training of those NN is the abundance and the quantity of pre-
¹⁴⁶⁸ labeled examples of segmentation on real histological samples. The collection of hundreds
¹⁴⁶⁹ of hand-labeled histological samples, with pixel-level precision, is virtually impossible.
¹⁴⁷⁰ This work thus proposes a methodology to generate, in a completely automatic way,
¹⁴⁷¹ synthetic pre-labeled histological-like images, that can be used as training material for
¹⁴⁷² a NN.

¹⁴⁷³ The method I propose consists of the recreation of the traditional histological speci-
¹⁴⁷⁴ mens' preparation, and it is based on the sectioning of a 3D virtual model of a region of
¹⁴⁷⁵ histological tissue. The virtual 3D model of a region of a particular type of human tissue
¹⁴⁷⁶ is built after physical and physiological considerations. The model is then subject to a
¹⁴⁷⁷ virtual sectioning operation, which yields the synthetic sampling of the virtual tissue in
¹⁴⁷⁸ which the histological identity of every pixel is perfectly known. This first image will act
¹⁴⁷⁹ as a segmentation mask for a second, realistic image. In fact, on top of this first image
¹⁴⁸⁰ are applied several aesthetical processing and refinements and the final product is the
¹⁴⁸¹ synthetic histological-like image. The pair made of the two images is then perfect for
¹⁴⁸² the supervised learning of a NN oriented toward the segmentation of histological images.
¹⁴⁸³ The production of each pair of images is completely automatic and it does not require the
¹⁴⁸⁴ intervention of any human operator, it is thus a scalable process that can produce a great
¹⁴⁸⁵ abundance of images. The quality of the images is directly connected to the richness and
¹⁴⁸⁶ the quality of the model. The perfect modelization of a region of tissue, let's say human
¹⁴⁸⁷ pancreatic tissue, is by far out of reach for this, hence the richness and the fidelity of the
¹⁴⁸⁸ produced images are inevitably lower than the real sample. Nevertheless, the quality of
¹⁴⁸⁹ the produced images is sufficient to perform the preliminary phase of the training of a

1490 NN following a training strategy known as curriculum learning. This learning process
1491 consists of giving the NN a copious quantity of lower complexity level example in the
1492 first instance, reserving the few and sophisticated real hand-labeled histological samples
1493 for the finalization of the training.

1494 The first chapter of this thesis is devoted to the contextualization of the present
1495 work. It is offered a description of how the histological samples are obtained after a
1496 tissue biopsy and how the digitalization process of the images works. The reader is then
1497 introduced to the framework of Deep Learning, its fundamental aspect, and components.
1498 The concept of neural network is exposed and it is given the general idea underneath the
1499 training of a DL-based model. The chapter comes to an end with a general introduction
1500 to the segmentation task in computer vision, and how it is currently tackled with state-
1501 of-the-art algorithms.

1502 The second chapter collects all the details of every less common technical tool I used
1503 during the design of this project. A brief theoretical introduction is proposed for every
1504 item besides the thorough description of its practical use. Some of the arguments touched
1505 by this chapter are quaternions, quasi-random number generation, Voronoi decomposi-
1506 tion, style transfer neural networks. In this chapter, a section is devoted to the description
1507 of a general methodology for computing the 2D section of an arbitrary three-dimensional
1508 polyhedron. The algorithm here described has been devised and implemented all by my
1509 self, and then inserted in the workflow of the project. As a conclusion for this chapter,
1510 I describe the working environment I built for developing this project and I mention all
1511 the code libraries I employed in my work.

1512 The third chapter is the center of this work, and it contains the description of all
1513 the design choices, and the steps I followed for the development of the two human
1514 tissue models I propose: the first of pancreatic tissue and the second of dermal tissue.
1515 The development has required the harmonization of many different technical aspects
1516 and mathematical tools. The first section is then dedicated to the description of the
1517 pancreatic tissue model, which passes through different steps: from a two-dimensional
1518 ramification taken as inspiration for the behavior of blood vessels to the complete three-
1519 dimensional model, with its subdivision in labeled cells. The second section is occupied
1520 by the description of the dermal tissue model, following the same spirit. The third
1521 section instead contains a thorough description of the method to perform the sectioning
1522 onto a virtual model and how to process the resulting images. It is necessary to perform
1523 several alternative processing and adjustment to achieve the desired aspect both for the
1524 segmentation mask image and for the histological-like image after the sectioning process.

1525 The method for the generation of datasets of synthetic images I propose in my thesis
1526 work is a self-supporting project and it is formally consistent. By the way, there are
1527 many possibilities for improvement and enrichment for the project. One first aspect to
1528 strengthen could be the richness of the models: adding more elements in the structure,
1529 and refining their representation of the tissue at the cellular level. This would lead
1530 to a better quality of the synthetic images, that would assist the training of NN in

more and different applications. Another aspect that lends itself to improvements in the development of a dedicated style transfer NN targeting the histological texture transfer, which could lead to interesting signs of progress in the quality of image generation. There is also the intention to perform an actual attempt of NN training on the images produced with this process. This would complete conceptually the idea underneath the project and would be an excellent opportunity to detect weaknesses and to draw up possible lines of development. The repeated application of the generation method would allow the building of entire datasets suitable for the training of DL-based models.

₁₅₃₉

Bibliography

- ₁₅₄₀ [1] Abien Fred Agarap. Deep learning using rectified linear units (relu), 2018.
- ₁₅₄₁ [2] Salah Alheejawi, Mrinal Mandal, Hongming Xu, Cheng Lu, Richard Berendt, and
₁₅₄₂ Naresh Jha. Deep learning-based histopathological image analysis for automated
₁₅₄₃ detection and staging of melanoma. In *Deep Learning Techniques for Biomedical*
₁₅₄₄ and *Health Informatics*, pages 237–265. Elsevier, 2020.
- ₁₅₄₅ [3] J. Alsayednoor and P. Harrison. Evaluating the performance of microstructure
₁₅₄₆ generation algorithms for 2-d foam-like representative volume elements. *Mechanics*
₁₅₄₇ of *Materials*, 98:44 – 58, 2016.
- ₁₅₄₈ [4] Hani A Alturkistani, Faris M Tashkandi, and Zuhair M Mohammedsaleh. Histolog-
₁₅₄₉ ical stains: A literature review and case study. *Global Journal of Health Science*,
₁₅₅₀ 8(3):72, June 2015.
- ₁₅₅₁ [5] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curricu-
₁₅₅₂ lum learning. In *Proceedings of the 26th Annual International Conference on Ma-*
₁₅₅₃ *chine Learning*, ICML '09, page 4148, New York, NY, USA, 2009. Association for
₁₅₅₄ Computing Machinery.
- ₁₅₅₅ [6] Jon Louis Bentley. Multidimensional binary search trees used for associative search-
₁₅₅₆ ing. *Commun. ACM*, 18(9):509517, September 1975.
- ₁₅₅₇ [7] R. W. Brockett. Robotic manipulators and the product of exponentials formula.
₁₅₅₈ In P. A. Fuhrmann, editor, *Mathematical Theory of Networks and Systems*, pages
₁₅₅₉ 120–129, Berlin, Heidelberg, 1984. Springer Berlin Heidelberg.
- ₁₅₆₀ [8] C. G. BROYDEN. The Convergence of a Class of Double-rank Minimization Algo-
₁₅₆₁ rithms 1. General Considerations. *IMA Journal of Applied Mathematics*, 6(1):76–90,
₁₅₆₂ 03 1970.
- ₁₅₆₃ [9] Bassem Ben Cheikh, Catherine Bor-Angelier, and Daniel Racoceanu. A model of
₁₅₆₄ tumor architecture and spatial interactions with tumor microenvironment in breast
₁₅₆₅ carcinoma. In Metin N. Gurcan and John E. Tomaszewski, editors, *Medical Imaging*

- 1566 2017: *Digital Pathology*, volume 10140, pages 73 – 80. International Society for
 1567 Optics and Photonics, SPIE, 2017.
- 1568 [10] Anna Choromanska, Benjamin Cowen, Sadhana Kumaravel, Ronny Luss, Mattia
 1569 Rigotti, Irina Rish, Brian Kingsbury, Paolo DiAchille, Viatcheslav Gurev, Ravi
 1570 Tejwani, and Djallel Bouneffouf. Beyond backprop: Online alternating minimization
 1571 with auxiliary variables, 2018.
- 1572 [11] Siddharth Singh Chouhan, Ajay Kaul, and Uday Pratap Singh. Image segmentation
 1573 using computational intelligence techniques: Review. *Archives of Computational
 1574 Methods in Engineering*, 26(3):533–596, February 2018.
- 1575 [12] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus En-
 1576 zweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The
 1577 cityscapes dataset for semantic urban scene understanding. *CoRR*, abs/1604.01685,
 1578 2016.
- 1579 [13] Angel Cruz-Roa, Ajay Basavanhally, Fabio Gonzlez, Hannah Gilmore, Michael Feld-
 1580 man, Shridar Ganesan, Natalie Shih, John Tomaszewski, and Anant Madabhushi.
 1581 Automatic detection of invasive ductal carcinoma in whole slide images with convo-
 1582 lutional neural networks. *Progress in Biomedical Optics and Imaging - Proceedings
 1583 of SPIE*, 9041, 02 2014.
- 1584 [14] Alessandro d’Agostino. Dataset generation for the training of neural networks ori-
 1585 ented toward histological image segmentation, april 2020.
- 1586 [15] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale
 1587 hierarchical image database. In *2009 IEEE Conference on Computer Vision and
 1588 Pattern Recognition*, pages 248–255, 2009.
- 1589 [16] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-
 1590 Scale Hierarchical Image Database. In *CVPR09*, 2009.
- 1591 [17] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and An-
 1592 drew Zisserman. The pascal visual object classes (voc) challenge. *International
 1593 journal of computer vision*, 88(2):303–338, 2010.
- 1594 [18] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of
 1595 artistic style, 2015.
- 1596 [19] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural com-
 1597 putation*, 9(8):1735–1780, 1997.
- 1598 [20] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image
 1599 translation with conditional adversarial networks, 2016.

- 1600 [21] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization,
1601 2014.
- 1602 [22] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical
1603 report, 2009.
- 1604 [23] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with
1605 deep convolutional neural networks. In *Advances in neural information processing*
1606 *systems*, pages 1097–1105, 2012.
- 1607 [24] Johannes Kster and Sven Rahmann. Snakemakea scalable bioinformatics workflow
1608 engine. *Bioinformatics*, 28(19):2520–2522, 08 2012.
- 1609 [25] Aristid Lindenmayer. Mathematical models for cellular interactions in development
1610 ii. simple and branching filaments with two-sided inputs. *Journal of theoretical*
1611 *biology*, 18(3):300–315, 1968.
- 1612 [26] Daniel. Longnecker. Anatomy and histology of the pancreas, 2014.
- 1613 [27] Shervin Minaee, Yuri Boykov, Fatih Porikli, Antonio Plaza, Nasser Kehtarnavaz,
1614 and Demetri Terzopoulos. Image segmentation using deep learning: A survey, 2020.
- 1615 [28] Muhammad Niazi, Thomas Tavolara, Vidya Arole, Douglas Hartman, Liron Pan-
1616 tanowitz, and Metin Gurcan. Identifying tumor in pancreatic neuroendocrine neo-
1617 plasms from ki67 images using transfer learning. *PLOS ONE*, 13:e0195621, 04 2018.
- 1618 [29] Arild Nkland and Lars Hiller Eidnes. Training neural networks with local error
1619 signals, 2019.
- 1620 [30] Travis E Oliphant. *A guide to NumPy*, volume 1. Trelgol Publishing USA, 2006.
- 1621 [31] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gre-
1622 gory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban
1623 Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan
1624 Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith
1625 Chintala. Pytorch: An imperative style, high-performance deep learning library,
1626 2019.
- 1627 [32] Allan Pinkus. Approximation theory of the mlp model in neural networks. *Acta*
1628 *Numerica*, 8:143195, 1999.
- 1629 [33] Prabu Ravindran, Adriana Costa, Richard Soares, and Alex C Wiedenhoeft. Clas-
1630 sification of cites-listed and other neotropical meliaceae wood images using convo-
1631 lutional neural networks. *Plant methods*, 14(1):1–10, 2018.

- 1632 [34] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional net-
 1633 works for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.
- 1634 [35] Juan Rosai. Why microscopy will remain a cornerstone of surgical pathology. *Lab-*
 1635 *oratory Investigation*, 87(5):403–408, April 2007.
- 1636 [36] Andrea Saltelli. Making best use of model evaluations to compute sensitivity indices.
 1637 *Computer Physics Communications*, 145(2):280 – 297, 2002.
- 1638 [37] Andrea Saltelli, Paola Annoni, Ivano Azzini, Francesca Campolongo, Marco Ratto,
 1639 and Stefano Tarantola. Variance based sensitivity analysis of model output. design
 1640 and estimator for the total sensitivity index. *Computer Physics Communications*,
 1641 181(2):259 – 270, 2010.
- 1642 [38] Caglar Senaras, Muhammad Khalid Khan Niazi, Berkman Sahiner, Michael P. Pen-
 1643 nell, Gary Tozbikian, Gerard Lozanski, and Metin N. Gurcan. Optimized generation
 1644 of high-resolution phantom images using cGAN: Application to quantification of ki67
 1645 breast cancer images. *PLOS ONE*, 13(5):e0196846, May 2018.
- 1646 [39] David F Shanno. Conditioning of quasi-newton methods for function minimization.
 1647 *Mathematics of computation*, 24(111):647–656, 1970.
- 1648 [40] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for
 1649 large-scale image recognition, 2014.
- 1650 [41] Sandro Skansi. *Introduction to Deep Learning: From Logical Calculus to Artificial*
 1651 *Intelligence*. Springer Publishing Company, Incorporated, 1st edition, 2018.
- 1652 [42] I.M. Sobol. Uniformly distributed sequences with an additional uniform prop-
 1653 erty. *USSR Computational Mathematics and Mathematical Physics*, 16(5):236 –
 1654 242, 1976.
- 1655 [43] I.M Sobol. Global sensitivity indices for nonlinear mathematical models and their
 1656 monte carlo estimates. *Mathematics and Computers in Simulation*, 55(1):271 – 280,
 1657 2001. The Second IMACS Seminar on Monte Carlo Methods.
- 1658 [44] M Titford. The long history of hematoxylin. *Biotechnic & Histochemistry*, 80(2):73–
 1659 78, 2005.
- 1660 [45] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy,
 1661 David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan
 1662 Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman,
 1663 Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson,
 1664 CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde,

1665 Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R Harris,
1666 Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt,
1667 and SciPy 1. 0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific
1668 Computing in Python. *Nature Methods*, 17:261–272, 2020.

1669 [46] Georges Voronoi. Nouvelles applications des paramètres continus à la théorie des
1670 formes quadratiques. premier mémoire. sur quelques propriétés des formes quadra-
1671 tiques positives parfaites. *Journal für die reine und angewandte Mathematik (Crelles
1672 Journal)*, 1908:102 – 97.