UNIVERSITY

FACULTY

STUDY PROGRAMME:

MASTER THESIS

Catchy Title

Submitted by:
John DOE
Matriculation Number: 123456
Street 123
789654 City

Initial Examiner:
Dr. Supervisor
Secondary Examiner:
Prof. Dr. Examiner

October 9, 2023

First printed interior sheet: Recto: same text and formatting as front cover. o Verso: ISBN, theme code, NUR code(s) and legal depot number. Position: bottom of the page.

Second printed interior sheet, recto: Members of the examination board - Title: ‘Members of the Examination Board’ / ‘Leden van de examencommissie’ (same language as front cover), - Members, per category: - Chair / Voorzitter - Other members entitled to vote / Andere stemgerechtigde leden - Supervisor(s) / Promotor(en) (they are members!) - (it is no longer needed to indicate the secretary) - Every member is listed as Title Name, Affiliation (incl. country if not BE)

UNIVERSITY

Abstract

Faculty

Master Thesis

Catchy Title

by John DOE

Write your abstract here.

Keywords: LaTeX, Master Thesis, Programming

Acknowledgements

Ich bedanke mich bei Dr. Supervisor für die Betreuung dieser Arbeit.

Contents

List of Figures

List of Tables

List of Abbreviations

CNN convolutional neural network

1 Introduction

Medical image segmentation, the process of delineating one region of the image such as cancerous tissues from the rest of the image, is a crucial step in computer-assisted medical image analysis. Whether the ultimate goal is surgical planning [1], diagnosis [2], performing measurements [3], or doing population-level research [4], segmentation is often the first step in understanding 2D and 3D medical images.

Neural networks have become the standard tool to achieve biomedical image segmentation in almost all problem areas including, among others, segmenting organs or specific tissues from CT, MRI, or X-ray images [5]; cells from microscopic images [6]; and skin lesions from dermatoscopic images [7].

While achieving impressive results, these results are highly dependent on the quantity and quality of the training data [8]. However, obtaining medical imaging data is challenging due to several reasons. Firstly, capturing medical images is costly both in terms of time and finances. For instance, MRI and CT scanning can take 30 minutes or more and require equipment that is inaccessible to large parts of the world. Secondly, such data is often large in terms of file size and stored in complex systems, increasing the friction of sharing and using the data. Finally, some jurisdictions define medical images as personally identifiable data [9] and thus require explicit consent for their secondary use for the purpose of training neural networks. While valid and understandable, these patient privacy concerns can limit the use of already existing large databases in medical institutions.

After obtaining a medical image, these images need to be labeled with high-quality delineations of the target region. Such labeling is often done through a tedious and time-consuming process where multiple experts would manually draw curves or polygons on the image. While some labeling methods make this process easier [10], each image still needs to be checked by an expert in the field. For challenging problems, this often requires highly trained and experienced specialists.

These challenges make collecting large medical image segmentation datasets infeasible. Therefore, to improve performance and robustness, there is a need to develop data-efficient segmentation methods. Data efficiency, sometimes referred to as sample efficiency, measures how well a model performs with respect to its sample size. Data-efficient models achieve good results given a small amount of data.

This thesis provides an overview of medical image segmentation and efforts to increase its data efficiency. It also presents several novel methods for achieving data efficiency in various contexts. The unifying central principle of these methods is the notion that necessary network capacity (number of parameters) grows with problem complexity. However, higher-capacity networks require a larger amount of data to be trained. The thesis aims to answer the question of how can we transform the data to make the segmentation problem simpler. This would allow us to train networks of lower capacity, and thus ones that are more data efficient. This is achieved using traditional image processing techniques informed by features detected by a neural network, thus combining the two worlds of traditional and neural network-based medical image segmentation.

1.1 Contributions of this thesis

This thesis aims to develop new methods of achieving data efficiency in medical image segmentation. In particular, we propose the following original contributions to the scientific literature:

1. A new biomedical image segmentation method based on polar transform preprocessing with a learned center point.
2. An improved method of reducing input image size in neural networks using salient image crops.
3. A new neural network architecture for high-resolution image segmentation that combines object detection in low-resolution images and segmentation in high-resolution images.
4. A new method of embedding depth information in two-dimensional convolutional neural network input data.

1.2 Organization of the thesis

todo

2 Neural Network-Based Segmentation of Biomedical Images

This chapter introduces biomedical images as well as the process of segmenting them. We will start with a brief overview of the most common types of biomedical images.

Biomedical images are a broad and diverse category of images referring to imaging use for the purposes of biology or medicine. These can range from complex modalities such as CT scans all the way to photographs (such as clinical images or photographs of plants). With such a diverse set of modalities and imaging techniques, biomedical images have a low level of cross-domain consistency. Methods developed for one modality often cannot be used in a different modality without modification. This is especially true for learned models such as neural networks. Biomedical image segmentation networks are notoriously bad at out-of-sample performance, i.e. when evaluated on a different modality than they were trained on. This chapter presents a brief overview of various biomedical imaging modalities and techniques, with a focus on those relevant to the methods presented in this thesis.

2.1 Common Types of Biomedical Images

Broadly, we may classify biomedical images into 3D and 2D images. 3D images include modalities such as CT and MRI, but also some types of microscopy as well as other techniques. 2D images, among others, include X-rays, most microscopic images, and dermatoscopic images. The images could also be classified by color space. Techniques that visualize the insides of objects such as CT, MRI or X-ray are usually grayscale. On the other hand, camera-based imaging techniques such as dermoscopy or photographs use three or more color channels. As is apparent, biomedical images do not yield themselves to clean classification, so the classification here is only a general descriptive categorization. There are novel techniques that combine different modalities that escape this classification.

2.1.1 3D Modalities

In medicine, 3D imaging of the patient allows experts to look below the patient's skin. It is hard to overstate the importance of modalities such as CT and MRI on modern medical developments and patient outcomes. These modalities are often captured and stored as voxel-based 3D files, where a voxel is the smallest 3D unit equivalent to a pixel in 3D. They are usually stored alongside patient data such as age, sex, imaging parameters and other relevant information.

Computed Tomography (CT)

Computed tomography is an x-ray-based imaging technique where different planes of the subject are captured and then reconstructed into a 3D image using a process called tomography. CT, as opposed to MRI, uses ionizing radiation that can be harmful in large

doses. This makes it important to evaluate the cost and benefit of each scan since each scan presents a risk of harming the subject. In addition, image quality is correlated with radiation dose, and the dose is carefully selected based on the required level of image quality so as to not unnecessarily radiate the subject.

Low-dose and high-dose CT images differ enough to cause issues in the segmentation model's performance across these two domains . High-quality images result in cite better neural network models, but their availability is much more limited than low-dose scans.

CT can be enhanced using a contrast agent. The contrast agent is usually injected intravenously and is chosen to appear with a high intensity on the resulting image. This makes it easier to delineate blood vessels from surrounding tissue, as is the case e.g. in angiography (CTA).

In the image itself, the values of a CT scan are usually stored as $W \times H \times D$ values, where W, H, and D are the x, y, and z dimensions, respectively. Each value represents a voxel, *volumetric pixel*, in a 3D volume. The voxels are usually not cubes and can have different lengths along each dimension. In the z-axis, the voxel length is known as slice thickness, and it is chosen based on the task at hand. Thinner slices increase the spatial detail in the image, leading to more details in small tissues. However, thinner slices generally also increase the level of noise in the image, so slice thickness is selected to maintain a tradeoff between spatial resolution and level of noise. The resolution along the x- and y-axes is governed by the field of view and the scanner itself.

The values of the voxels are usually stored as 12 bit signed integer values. The value corresponds to the attenuation of the X-ray as it passes through the tissue. Denser structures such as bones attenuate the radiation more strongly than adipose tissue, and thus result in higher values. This results in higher-intensity voxels in the resulting CT image. To maintain consistency across scans and machines, the attenuation values are linearly transformed such that air has a value of -1000 while water has a value of 0 at standard pressure temperature, as:

$$\mathrm{HU}(\mu) = 1000 \cdot \frac{\mu - \mu_{\text{water}}}{\mu_{\text{water}} - \mu_{\text{air}}} \tag{2.1}$$

where μ is the attenuation value of a voxel, and μ_{water} and μ_{air} are attenuation values of water and air, respectively. This transformation is called the Hounsfield unit (HU) scale. When calibrated in this way, the attenuation of each tissue is represented as relative to the attenuation of water, and thus values are standardized across images, CT machines, imaging parameters, and centers. Various tissues broadly fall into Hounsfield unit values as is shown in Table 2.1.

Tissue	Hounsfield value
Fat	-30 to -70 HU
Muscle	20 to 40 HU
Bone	1000 HU
Blood	13 to 50 HU

Table 2.1: Approximate Hounsfield values of various tissues [11], [12].

Hounsfield values are a crucial tool in analyzing medical images. Aside from getting a better understanding of what tissues or objects are present, measured Hounsfield values can have diagnostic significance. For instance, clotted blood has a higher HU value than unclotted blood, and so the Hounsfield value is used as an indicator of intracranial

hemorrhage [12]. Another example is using the average HU value of epicardial fat as a sign of myocardial infarction [13].

Since the human eye recognizes much fewer gray levels than are available on a CT scan, the scan is typically windowed when viewed by an expert. Windowing refers to the process of shrinking the value range using two thresholds (t_1, t_2) as:

$$y(x) = \begin{cases} t_1, & \text{if } x \leq t_1 \\ x, & \text{if } t_1 < x < t_2 \\ t_2, & \text{if } x \geq t_2 \end{cases} \tag{2.2}$$

where x is a given HU value. This allows the software to visually stretch the remaining range of gray values to more easily see smaller differences in attenuation, as seen in Figure 2.1. This technique is not limited to human experts. It is common to window CT image inputs into segmentation neural networks. For instance, if segmenting fat, it is often beneficial to discard all voxels outside of the fat tissue range [14].

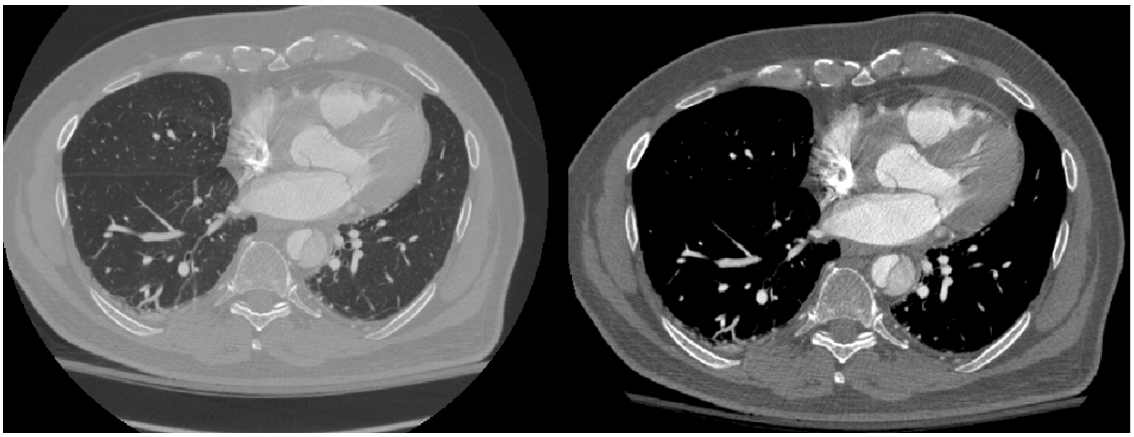

Figure 2.1: A cardiac CTA in its full range (left) and windowed (right). [15]

Magnetic Resonance Imaging (MRI)

Much like CT, MRI visualizes the insides of an object using a voxel-based image. MRI works by first applying a strong magnetic field such that protons align parallel to the z-axis. The protons are then excited using a radio frequency pulse which causes them to become misaligned. After the pulse, the protons gradually return back to alignment and induce an electric current. Unlike CT which measures the attention of X-rays, MRI measures this induced current while protons are returning to equilibrium alignment. The time to reach the equilibrium state depends on the specific tissue type. The use of magnetic fields instead of ionizing radiation means that MRI does not cause harm to the patient due to radiation. Unlike CT, MRI values are not standardized across scans and MRI machines, and the specific values can not be used diagnostically across scans, only in relation to surrounding tissues.

MRI generally provides a better contrast than CT, especially in soft tissues. This makes MRI the gold standard of imaging for a large number of tissues and diagnostic procedures. Much like CT, this contrast can be further enhanced using contrast agents.

However, MRI imaging is slower when compared to CT which causes patient discomfort, especially for claustrophobic and non-neurotypical patients. MRIs are also not possible in cases where the subject has non-removable magnetic objects such as coronary pacemakers and other implants.

The voxels in an MRI, much like a CT, are generally rectangular solids and can different in the x-, y- and z-axis lengths. The resolution is government by the field of view, the MRI scanner as well as imaging parameters. Generally, increasing resolution leads to higher levels of noise and a longer acquisition time. Thus, a compromise needs to be determined for each imaged tissue and diagnostic task.

The voxel values are weighted based on the time to reach the equilibrium state which is achieved through two independent processes called T1 and T2. T1 measures the time it takes the protons to reach equilibrium longitudinal magnetization, while T2 measures the time to regain its equilibrium transverse magnetization. Different tissues regain equilibrium states in T2 and T2 at different rates, so MRI images can be weighted according to T1 or T2 time. Water has a long T1 time while fat has a short T1 time, so in T1-weighted images fat appears with a higher intensity than water. On the other hand, water has a high T2 and appears as high-intensity on T2-weighted images. This can be seen in Figure 2.2.

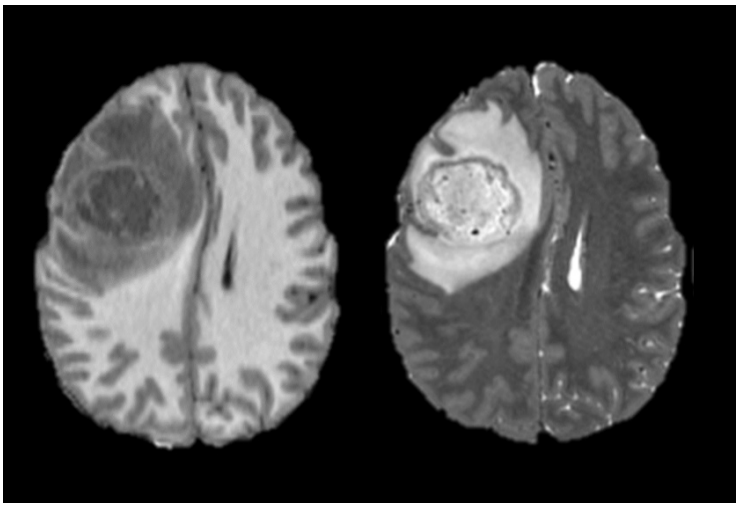

Figure 2.2: An example T1-weighted MRI (left) and a T2-weighted MRI (right) showing a diffuse glioma. [16]

The relative differences in T1 and T2 images make the two weightings more or less beneficial for imaging certain tissues. T1 weighting is useful, among others, for identifying fatty tissue or detecting liver lesions. T2 weighting is useful, among others, for identifying white matter lesions or edemas.

2.1.2 2D Modalities

While 3D modalities offer a detailed view of a subject, they are often time-consuming and invasive in the case of CT. 2D modalities such as X-ray and diagnostic ultrasound are often quicker and more available, making them ideal for screening and simpler diagnostics. The same reason also results in a much larger amount of publicly available datasets in 2D modalities compared to 3D ones. Segmentation, object detection, and classification in X-ray images, for instance, is one of the most active fields in computerized medical image analysis research [17], [18].

X-ray imaging

Similarly to CT, X-ray imaging or radiography uses ionizing radiation emitted on one side of the object and detected on the other. The intensity of the image corresponds to the attenuation of the emitted radiation. Denser materials appear with a higher intensity on the resulting image due to their high attenuation.

When capturing an X-ray image, an expert manually positions the generator. The position of the generator and the object determine the magnification and field of view in the image. If the distance between the detector and the object is larger than the distance between the generator and the object, the object will appear larger on the resulting image. This magnification means that the scale on an X-ray image is relative to the image itself and so objective measurements cannot be made on an X-ray.

The expert also determines various parameters that ultimately change the quality and quantity of the X-ray beams. The quality measures the ability of an X-ray to penetrate tissue and is proportional to the X-ray energy level. Quantity, on the other hand, measures the number of photons that constitute the beam. Increasing the beam quality allows for imaging denser tissues such as bones or through large bodies but results in lower contrast in soft tissues. This means that intensity in X-rays is not standardized across images like it is in CT, and so there is no objective unit of measuring X-ray intensity.

Dermatological images (clinical and dermatoscopic)

An increasingly active application of deep learning-based models is in dermatological applications, namely skin lesion analysis. This is driven in part by organisations such as the International Skin Imaging Collaboration (ISIC) that curates large dermatological datasets [19]. These datasets consists of clinical and dermatoscopic images. Clinical images are regular photographs of skin lesions, while dermatoscopic images are captured with a digital epiluminescence dermatoscope. This dermatoscope consists of a camera attached to a magnifying lens with a built-in light source. It allows capturing detailed and magnified images of a skin lesion while filtering out skin reflections.

The primary application of deep learning in dermatological images is for classification — predicting whether a lesion is benign or not or detecting the type of skin disease. However, skin lesion segmentation plays a role in the detection as delineating the skin lesion border can be used to provide more objective attributes of the skin lesion [19].

Microscopy

In biomedicine, one of the most used applications of computer vision is in segmenting, analyzing, and quantifying microscopic images. This encompasses various tasks in digital pathology such as detecting cancerous cells, segmenting nuclei, or counting white blood cells.

Publicly available microscopic images of cells are abundant due to how frequently they are captured and that they don't contain any personally identifiable information. However, these images sometimes present a challenge due to their size. Microscopic images can have an area of multiple megapixels, far too large for current deep-learning models. Therefore, the images are often split into patches, downscaled or analyzed in a coarse-to-fine manner [20].

With those modalities in mind, we can now cover the process of segmenting the images.

2.2 Image Segmentation: From Images to Segmentation Maps

Image segmentation is the process of categorizing each pixel (or voxel) of an image as belonging to one of several predefined classes. For instance, a 3-class problem of segmenting CT images could be liver, liver tumour, and background. Each class gets assigned an arbitrary class label, such as '0', '1' and '2' for the background, liver and tumour, respectively. The segmentation output is another image of the same size, only the value for each voxel corresponds to the class label of that voxel. Voxels belonging to the liver would each have a value of '1', etcetera. This resulting image is called a **segmentation map**, since it acts as a map for the original image. In medical image segmentation commonly only target is segmented, which is called **binary segmentation** since we are using two classes, the target, and the background.

However, multi-class segmentation problems can be reduced to a set of binary segmentation problems where a separate class-vs-background segmentation map is constructed for each class. Mathematically, given a set of K classes, and an input d-dimensional image of N channels $I(A)$, $I \in \mathbb{R}^N$, $A \in \mathbb{Z}^d_{\geq 0}$ where A is a vector representing the location of each voxel, the segmentation map can be expressed as a function $M : \mathbb{Z}^d_{\geq 0} \to \mathbb{R}^K$ mapping each pixel location to a vector of class probabilities:

$$M(A) = (\ \Pr(C_1 \,|\, I(A)),\ \Pr(C_2 \,|\, I(A)),\ \cdots,\ \Pr(C_K \,|\, I(A))\ ), \tag{2.3}$$

where $\Pr(C_i \,|\, I(A))$ denotes the probability that the voxel $I(A)$ contains an object of class C_i. Expressed this way, the segmentation map is a K-channel image of the same size as I, where each channel corresponds to a probability map of finding an object of a given class at a given voxel location.

In the case of binary segmentation where C_1 is the background and the C_2 is the target object, $\Pr(C_2 \,|\, I(A)) = 1 - \Pr(C_1 \,|\, I(A))\ \forall A$ and therefore $M(A)$ can be treated as a scalar value $M(A) = \Pr(C_2 \mid I(A))$. This representation is very common as many medical image segmentation problems are binary segmentation problems, such as segmenting organs, cell nuclei, skin lesions, etc.

Ultimately, to delineate tissues the segmentation masks $M(A)$ are commonly binarized such that voxels containing the target object become '1' and the background becomes '0'. This is why $M(A)$ is also commonly called a **segmentation mask**. In computer vision, the term mask refers to a binary image $M_{01}(A) \in \{0, 1\}$ that hides (masks) regions in another image producing a masked image $I_m(A) = I(A)M_{01}(A)$. In the context of deep learning-based image segmentation, the terms segmentation map and segmentation mask are often used interchangeably.

What follows is a broad overview of commonly used medical image segmentation methods, from traditional ones based on heuristics to complex model-based approaches broadly used today.

2.2.1 Traditional Image Processing Methods

For the purpose of this review, traditional image processing methods include techniques like thresholding, region growing, active contours, as well as morphological operations. These are often combined with heuristic strategies to refine the segmentation area, establish an initial contour, or enhance advanced methods. Heuristics are best-practice rules derived from previous samples and experiences. For instance, the longest component in the bone intensity range of an X-ray image is usually the femur.

These methods are usually developed with a very specific application in mind and are hard to translate to other tasks and domains without significant changes. They are also sensitive to parameter selection and properties of the image such as intensity level.

However, since they don't use a learning component they are easily explainable and their limitations can be known ahead of time. Their validity can also be confirmed using fewer samples than is the case for learning-based methods.

Despite the popularity of deep learning, these techniques still play a vital role in modern medical image segmentation. Traditional techniques are often used as methods for pre-processing and data augmentation, as well as refining deep learning model outputs. As we will show later in the dissertation, this can both increase the robustness and data efficiency of deep learning-based segmentation models.

Image Thresholding

As mentioned earlier in this chapter, voxel intensities on CT scans are measured using the Hounsfield scale, which represents the tissue's X-ray radiation attenuation at each pixel. Pixels indicative of adipose tissue usually fall between -250 HU and -30 HU. This range provides a means to segment adipose tissue by applying image thresholding. A large number of studies presenting CT segmentation methods use thresholding, at least partially, to segment an object. Hounsfield unit thresholding is an easy and safe way to discard irrelevant voxels and allow the model to focus on a few number of voxels.

It can be also used to greatly simplify a model's task. In the case of epicardial fat segmentation, the fatty tissue is sparsely distributed in a complex shape inside the pericardium. However, the pericardium itself has a smooth, elliptical shape and is comparatively easy to segment. By segmenting the pericardium and then thresholding the pericardium region to the fatty tissue range, we can find epicardial fat without having to segment its complex shape directly [14].

Region Growing Techniques

Region growing is a voxel-based image segmentation method [21]. Beginning from a designated seed voxel, the method assesses neighboring pixels in successive steps. If these pixels meet a specified criterion, often related to pixel values or textures, they are integrated into the region. Despite being a straightforward method, region growing's effectiveness relies heavily on the seed point selection and may struggle with nuanced transitions between regions. This process is shown in Figure 2.3.

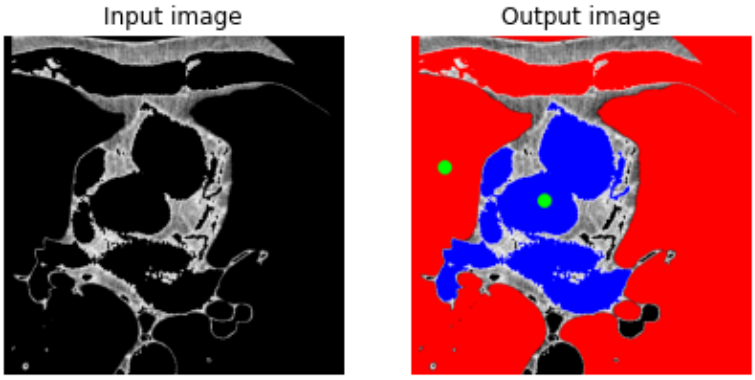

Figure 2.3: A demonstration of region growing for delineating the internal and external areas of the pericardium on a CT slice, set to the adipose tissue intensity range. The left image is the original input, and the right image depicts the segmented outcome with the heart exterior in red and the interior in blue. The green dots represent the manually chosen seed points initiating the region-growing technique. [14]

Active Contours or Snakes

Active contours, often termed "snakes", are segmentation methods that employ dynamic curves to outline image parts [22]. The process involves tightening a preliminary curve around an object iteratively until it conforms to the object's shape. The adaptation is guided by an energy function that evaluates the curve's smoothness and proximity to edges. A practical depiction of active contours is displayed in Figure 2.4.

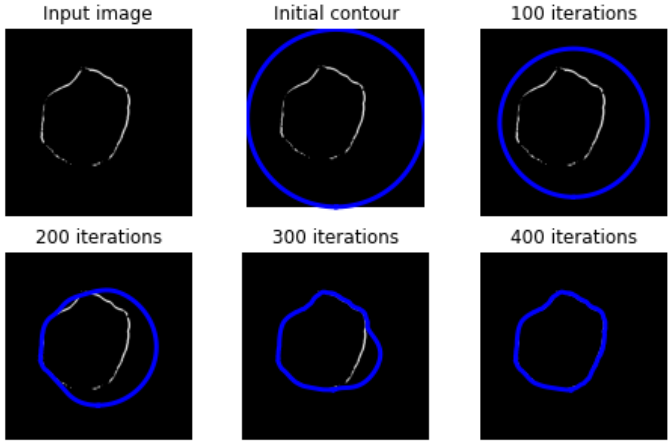

Figure 2.4: A demonstration of employing active contours to complete the absent segments of the pericardium line, displayed in white. The contour, illustrated in blue, starts as a complete circle surrounding the image. With every iteration, the contour adapts more closely to the image's shape. [14]

Atlas-Based Segmentation

Atlas-based methods, differing from contour-based ones, leverage the spatial relationships among structures [23]. An expert typically creates an atlas by manually segmenting and labeling structures in an image. Due to anatomical variations, multiple atlas images are often merged to produce a representative version. This consolidated atlas can then be employed to segment new images using a registration algorithm. Image registration is an optimization problem where one image, called the moving image, is deformed to best align with a target image according to some scoring function. In atlas-based segmentation, a new moving image is deformed to be aligned with the target image that was used to construct the atlas. The atlas can then be used as a segmentation map for the moving image, as it is now aligned with the atlas.

The scoring function usually uses the distance between heuristic-based landmarks in the target and moving images to determine how well the two images align. The atlas-based registration process is illustrated in Figure 2.5. A disadvantage of this approach is that the process can lead to a complete failure to segment the image if the target and moving images are too dissimilar.

Atlas-based segmentation has fallen out of favor due to the emergence of deep learning-based methods. However, recently there has been significant progress in image registration and atlas-based segmentation using deep learning-based techniques [24]. These approaches offer good potential for merging traditional and newer approaches will be discussed later in the chapter.

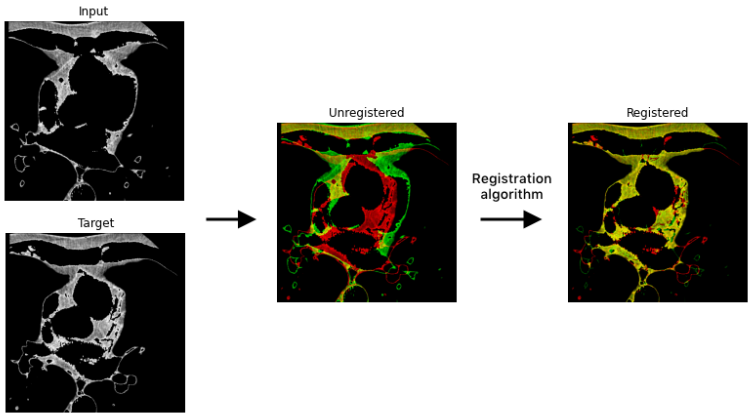

Figure 2.5: A schematic representation of the registration procedure. Initially, input and target images are chosen. Throughout the registration phase, the input image (shown in green) undergoes deformation to align with the fixed target image (shown in red). [14]

2.2.2 Machine Learning

While deep learning generally falls within the machine learning umbrella term, in this dissertation the term "machine learning" will be used to describe techniques that are not based on deep neural networks. These include methods like support vector machines, deep forests or manual feature engineering.

Within this context, it is possible to define a segmentation problem as a per-voxel classification problem. Let there be a classifier of K classes,

$$H(x;\theta) = (\ \Pr(C_0\,|\,x),\ \Pr(C_1\,|\,x),\ \cdots,\ \Pr(C_{K-1}\,|\,x)\ ), \tag{2.4}$$

parameterized by θ with input voxel-level feature vector x. The features are manually selected and constructed to represent each pixel and its surrounding region. Commonly, these include the pixel's intensity, mean intensity of the area, image moments, and other information deemed relevant for the classification. The classifier is trained to minimize a loss function by feeding each pixel's features to the classifier and comparing the output to the ground truth output. A visual demonstration of this machine-learning process for image segmentation is shown in Figure 2.6.

Another common approach is splitting the image into smaller patches, and then using a machine learning classifier to class each patch into belonging to one of a set of K classes. The classifications are then fused together to form a segmentation map.

These approaches can be more data efficient than deep learning-based approaches [14] but have a limited ability to model complex features and dependencies.

Now that we have an overview of more traditional methods, we can dive into deep learning-based ones.

2.3 Deep Learning-Based Segmentation Methods

While machine learning encompasses a broad range of techniques for building statistical models, deep learning focuses on using artificial neural networks. Artificial neural networks consist of simple nodes called neurons. Each neuron is a non-linear function of the sum of its inputs. This non-linear function is called the **activation function**. The

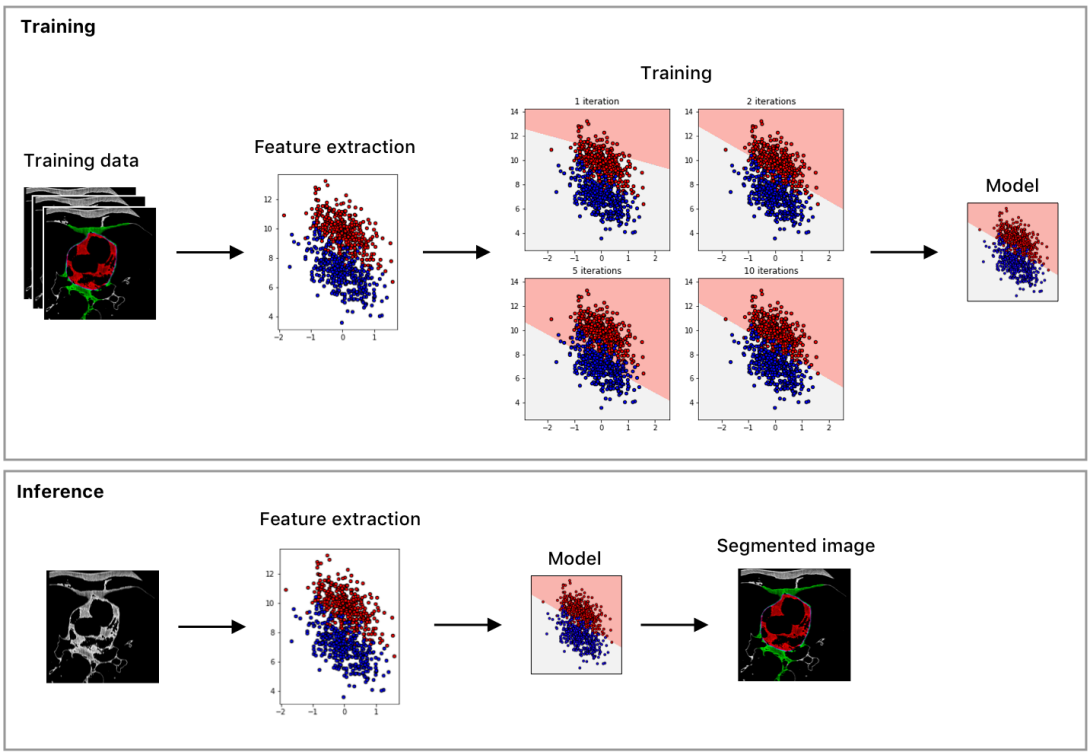

Figure 2.6: A schematic of a supervised linear classifier in a machine learning workflow. The upper section illustrates the training phase. Here, features are color-coded according to their known class from training data, depicted in red and blue. The parameters of the decision boundary, which demarcates the zones of the two classes (highlighted in light red and grey), are determined during training. The lower part of the diagram depicts the inference stage. In this phase, features are extracted from new images, and the trained model is employed to classify each pixel in the image. [14]

neurons are all arranged in a graph where the outputs of one set of neurons are connected to the input of another set of neurons. Each connection has an associated **weight** and **bias** parameter that either multiplies by the weight or adds the bias value to the output before it goes into the next neuron. The exact configuration of the graph, i.e. the number of neurons and how they are connected, is determined by hand and is referred to as the **neural network architecture**. A typical neural network architecture can be seen in Figure **??**.

Typically, neurons are arranged into **layers**, where neurons of a layer connect only to the next layer, without skipping layers in between. Current deep-learning networks usually have between ten and 100 layers.

As an example of a simple neuron, we can imagine a neuron that receives a 3-valued vector as an input and produces an output using the rectified linear unit (ReLU) function:

$$ReLU(x) = \begin{cases} x, & \text{if } x > 0, \\ x, & \text{otherwise.} \end{cases} \tag{2.5}$$

ReLU can be seen as a simple thresholding function that clamps values below zero to zero. In spite of its simplicity, ReLU is one of the most activation functions in neural networks.

First, the input vector $x = [x_0 \; x_1 \; \cdots \; x_{n-1}]$ is multiplied by the weights column vector w and added with the bias vector b:

$$z(x; w, b) = \begin{bmatrix} x_0 & x_1 & \cdots & x_{n-1} \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ \cdots \\ w_{n-1} \end{bmatrix} + \begin{bmatrix} b_0 & b_1 & \cdots & b_{n-1} \end{bmatrix} \tag{2.6}$$

Then, all of the inputs are summed and the neuron output is produced using the activation function af (in the case of this example ReLU):

$$f(x; w, b) = af\left(\sum_{i=0}^{n-1} z(x; w, b)_i\right). \tag{2.7}$$

The parameters b and w of each neuron can be stored inside a large matrix θ. This matrix is called the **parameters of the network**. Current deep learning networks usually have multiple millions of parameters.

2.3.1 Neural Network Training, Validation and Testing

Initially, the values of these parameters are usually set to random numbers. Their exact values are determined automatically through a process called **training**. During training, the values of the network's parameters are iteratively adjusted to minimize the value of a **loss function**. The loss function measures how well the network is performing its task based on known correct values. For instance, in image segmentation, the loss function would measure the similarity between a hand-labeled segmentation map and the one produced by the network. Using a process called backpropagation, each parameter is updated in the direction that will decrease the loss function's value. This is repeated multiple times for each image in a **training dataset**.

The training dataset, much like any statistical data, is a sampling of the real world. Depending on the size and quality of the dataset, it is possible that the training dataset is not representative of the true distribution of the data. Moreover, it is also possible that the network learns to produce correct solutions for each training image individually instead of learning general patterns in the data. This is called **overfitting** — the network learns to minimize the loss in the training data but does not generalize to data outside the training domain. To overcome this, two other datasets are used besides the training dataset.

The **validation dataset** is used to assess the model's performance during model development. While creating a model, one has to make decisions about the architecture, data processing, and other implementation details. To evaluate these decisions, inference is performed on the validation dataset (with known correct values) to estimate its real-world performance. However, the network is never trained on the validation dataset, it is only validated after the training process. The third **test dataset** (sometimes called a hold-out dataset) is used only once the development process is completed as a final estimate of real-world performance.

The training, validation, and testing datasets are created before the model development process. Usually, they are sampled randomly from a larger dataset with ratios such as 80%, 10% and 10% for the training, validation, and testing datasets, respectively. In the next chapter, we will discuss overfitting in more detail.

2.3.2 Encoders and Decoders

Machine and deep learning-based segmentation methods can be framed as consisting of two stages:

The first stage is an **encoder** stage $En : \mathbb{R}^{W \times H \times C} \rightarrow \mathbb{R}^{n \times m \times d}$ which maps an image of size $W \times H$ and C channels into a feature vector. This stage can be seen as encoding salient information in the image in a smaller representation, effectively compressing the image — leading to the name encoder. In deep learning parlance sometimes this is referred to as the **backbone** of the network.

The output of the encoder is given to the second stage called the **decoder** or the **segmentation head**. The decoder is a function $De : \mathbb{R}^{n \times m \times d} \rightarrow \mathbb{R}^{K}$ that maps a given feature map into a segmentation map. Given an image $I(A)$, the segmentation process can then be written as:

$$M(A) = (En \circ De)(I(A)) \tag{2.8}$$

En is a function parameterized by a vector of parameters θ_{En}, while De is parameterized by θ_{De}. In machine learning approaches, En is not a trainable function and θ_{En} consists of hand-selected parameters such that it extracts pre-selected features from the image. The value of θ_{De}, on the other hand, is determined by minimizing a loss function. The loss function measures how well the model segments an image compared to a ground truth known segmentation of that same image. However, in deep learning approaches, both θ_{En} and θ_{De} are determined by minimizing a loss function.

Thus, the difference between traditional machine learning and deep learning is in the encoder stage. In deep learning, the values of the features are determined by a neural network, while in machine learning they a produced using a hand-crafted function.

The separation of segmentation models into encoder and decoder stages is a crucial aspect of current research in deep learning. This separation allows for independent improvement of both stages and an easy combination of different encoders and decoders. For instance, a classification and segmentation network could use the same encoder architecture. The classification model would use a simpler decoder that maps the input features into a vector of length K representing the score for each of the K classes. The segmentation model would use a comparatively complex backbone that produces a segmentation map. However, both models would use the exact same encoder. In fact, there are models such as Mask R-CNN **heMaskRCNN2018** that use two parallel decoders for different tasks, both connected to the same encoder. Throughout the rest of this thesis, when describing neural network architectures, we will describe them in terms of their encoder and decoder and how the two are connected.

2.3.3 Convolutional Neural Networks

Convolutional neural networks have spurred a revolution in computer vision and made deep learning the de facto standard for solving complex computer vision tasks. The central aspect of a CNN is the convolutional layer. The convolutional layer replaces the standard neuron mentioned earlier in this chapter with a convolution-like operation.

Generally, convolution is a mathematical operation between two functions, but for the purposes of this introduction, we will only consider 2D convolutions between two square images, as that is most relevant for image segmentation. Convolution is an operation $I(A) \star k(B)$ where $I(A)$ is an image $I(A) \in \mathbb{R}^{W \times H}$ and $k(B)$ is a matrix $k(B) \in \mathbb{R}^{w \times h}$ indexed by locations $B \in \mathbb{Z}^2_{\geq 0}$ for 2-dimensional images. The second matrix is called the **convolutional kernel**. The convolution operation at $A = (a_x, a_y)$ can then be defined as:

$$(I \star k)(a_x, a_y) = \sum_{j=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} I[i,j]k[a_x - i, a_y - j]. \tag{2.9}$$

Explained differently, the resulting image is produced by sliding the kernel over the input image, one pixel at a time. At each pixel location, values where the kernel and the image overlap are multiplied, and all of the products are summed together to form the value of that pixel in the resulting image. This is shown visually in Figure **??**.

While mathematically a simple operation, convolution is exceedingly powerful and can produce almost endless transformations of an image. It's most commonly used for filtering: A convolution can elegantly find patterns in the image. One such example is the convolution with a kernel called the Prewitt operator:

$$I_y(A) = I(A) \star \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \tag{2.10}$$

When convolved with this kernel, the resulting image has high-intensity pixels in regions where vertical edges are present, and low intensity everywhere else. This can be seen in Figure **??**.

In the image, the values of vertical edges necessarily have to have a large jump in values going from left to right or right to left. Otherwise, there would be no perceptible edge. This kernel takes advantage of that fact to accentuate parts of the image where there is such a jump. It does this by replacing each pixel with the difference between the pixels on its left and its right.

Here is how this happens: for each pixel of the input image, the kernel is placed such that it is centered on that pixel. This means that the values of the pixel as well as its neighbors above and below are all multiplied by zero. The neighbors on the left are multiplied by -1, and the ones on the right are multiplied by 1. Summed together, the result represents the sum of the values on the right of the pixel, minus the sum of the values on the left.

To illustrate this, let us zoom into a region of the image where no vertical edges are present:

$$\begin{bmatrix} 128 & 130 & 136 \end{bmatrix} \star \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} = \sum_{i,j} \begin{bmatrix} -1 \times 0 + 0 \times 128 + 1 \times 130 \\ -1 \times 128 + 0 \times 130 + 1 \times 136 \\ -1 \times 130 + 0 \times 136 + 1 \times 0 \end{bmatrix} = 8$$

This section of the image does not contain a vertical edge, so the convolution result is a relatively low value. In a standard image with values in $[0, 255)$, 8 would appear almost completely black.

However, consider some section of the image where a vertical edge is indeed present:

$$\begin{bmatrix} 63 & 66 & 132 \end{bmatrix} \star \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} = \sum_{i,j} \begin{bmatrix} -1 \times 0 + 0 \times 64 + 1 \times 66 \\ -1 \times 64 + 0 \times 66 + 1 \times 132 \\ -1 \times 66 + 0 \times 132 + 1 \times 0 \end{bmatrix} = 68$$

The value is now much larger due to the difference between the left and right sides of the image.

Aside from edge detection, there are many commonly used convolution kernels to perform common task such as blurring, sharpening, or denoising images. A CNN can

leverage the power of the convolution by stringing together kernels to match complex and intricate patterns in the image.

In a CNN, convolutional layers are connected to one another in a similar fashion to neurons in a regular neural network. One convolutional layer performs a user-defined number of convolutions n, and stores the result in an n-channeled matrix called a **feature map** where each channel is a convolution of the input with a new kernel. Thus, a convolutional layer holds n convolutional kernels. The values of the kernels are the parameters of the layer — i.e. they are set during training to minimize a loss function. Finally, a convolutional layer also has an activation function and it is applied to the resulting feature map. Just like in a regular neural network, this allows non-linear operations between convolutional layers, giving them the ability to find more complex features.

The actual operation inside a convolutional neural network differs from the mathematical convolution we described earlier but achieves a similar effect. Firstly, both the kernel and the image are three-dimensional — after all, the input image usually has 3 channels, and convolutional layers themselves can have thousands of channels. The kernel is therefore a $w \times h \times c$ matrix.

In each step, the kernel volume slides across the spatial (width and height) dimensions of the input, and stretches across the input since it has the same number of channels. Just like in the 2D convolution described above, all of the values are multiplied and the products are summed to produce one scalar value. Thus, the result of each convolution is still two-dimensional, since the kernel only slides along the width and the height of the input. This is visualized in Figure 2.7.

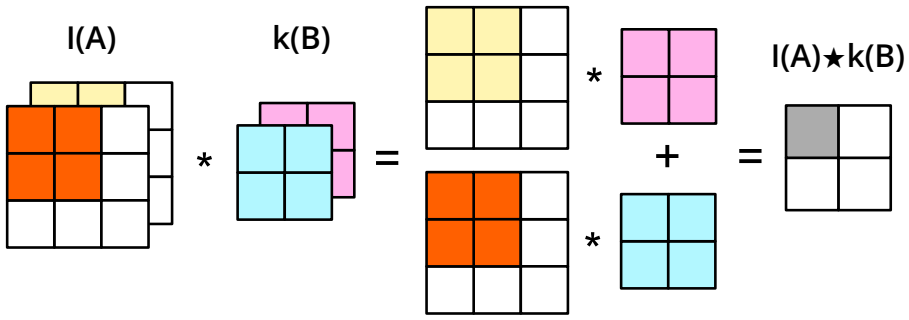

Figure 2.7: A view of one step of a single convolution operation inside a CNN layer. The layer performs multiple convolutions, each with a different kernel that has an equal number of channels as the input image. In each step, the whole kernel slides over the width and height of the image, and the overlapping channels are multiplied together and summed to produce a single output value. The output of the convolution is one channel of a n-channel image where n is the number of different kernels in the layer.

Generally, most CNN-based encoders follow a similar pattern of gradually reducing the width and height of the feature maps while increasing their depth. Depth is increased by giving each successive convolutional layer more kernels, thus increasing the output depth since each channel is the output of a convolution using one of the kernels. The spatial (width and height) dimensions are reduced using **pooling layers**. These layers downsample the image in some way, such as averaging each surrounding neighborhood of pixels. By doing so, a convolutional neural network achieves two goals. First, detailed spatial information is gradually removed from the image, compressing the relevant information into a smaller matrix. Secondly, the network is gradually able to build up large intricate features by combining small and general features. Such an architecture is shown in Figure 2.8.

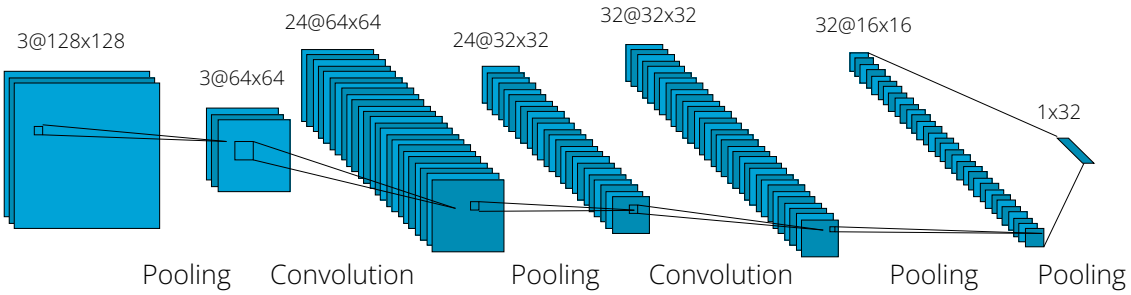

Figure 2.8: A typical architecture of a CNN encoder. The encoder consists of consecutive convolutional and pooling layers that gradually increase the feature map depth and decrease its width and height. The result is a map of features that tells the decoder what features are on the image but does not provide much spatial information about the location of those features. [25]

To illustrate this, we can think of a contrived example of a CNN feature detector to detect grapes on a vine. The first few layers could do a simple operation such as edge detection, and for that, they need a lot of spatial information. The next layer could combine the edges to detect a specific shape, such as a circle. For this, it needs less spatial information as the edges are already detected. A third layer could combine several circle locations to detect a cluster of grapes on a vine. For this, it only needs a very rough location of the circles, and thus does not need detailed width and height-wise information.

A convolutional decoder can work in much the same way, but reversed. Instead of adding channels, a convolutional decoder successively removes channels and upsamples the image until the resulting image is a segmentation map $M(A) \in C \times W \times H$.

Another way to achieve upsampling is using a **transposed convolutional layer**. These layers are very common in segmentation neural networks and allow for dynamic, learned upsampling of the image. Regular convolutions as described earlier cannot increase the image dimension, the output is always of equal size or smaller than the input image since convolution slides along each pixel of the input. To get around this, a transposed convolution performs scalar multiplication of the whole kernel and the pixel it is sliding over during each sliding window step. To illustrate this, let us look at an example. Assume one 2×2 channel i of an input feature map $m(A)$ and one 2×2 kernel $k(B)$. The transposed convolution would be calculated as follows:

$$m \star^T k = \begin{bmatrix} m_{11}k_{11} & m_{11}k_{12} & 0 \\ m_{11}k_{21} & m_{11}k_{22} & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & m_{12}k_{11} & m_{12}k_{12} \\ 0 & m_{12}k_{21} & m_{12}k_{22} \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ m_{21}k_{11} & m_{21}k_{12} & 0 \\ m_{21}k_{21} & m_{21}k_{22} & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & m_{22}k_{11} & m_{22}k_{12} \\ 0 & m_{22}k_{21} & m_{22}k_{22} \end{bmatrix}$$

The procedure can be similarly calculated for an arbitrarily sized image and kernel. Commonly, transposed convolution is referred to as *deconvolution*, however, in this thesis, we do not use the term to avoid confusion with the more standard definition of deconvolution outside of the deep learning context.

Segmentation models that use only convolutional and pooling layers are called **fully convolutional models** and are one of the most widely used types of models for medical image segmentation.

2.4 CNN Architectures for Medical Image Segmentation

A majority of deep learning-based medical image segmentation models to date use convolutional layers to achieve their task. Over the years, a large number of architectures have been proposed and standardized. Some of the them have been created for the task of natural image classification or segmentation, while others are made for more domain-specific domains in medical imaging. In this chapter, we will go over some of the most seminal architectures for medical image segmentation in chronological order.

2.4.1 Fully Convolutional Network (FCN)

One of the simplest contemporary architectures for image segmentation is the fully convolutional network (FCN) [26]. FCN aims to solve the problem of how do we produce an output value for each pixel of the input while leveraging convolutional encoders. As mentioned earlier, convolutional encoders gradually downsample the image while increasing the number of channels in the feature maps. There is then a need to reverse the process and decode the features into an image of equal width and height of the input image.

FCN solves this by using convolutional layers to reverse the encoding process. In a sense, they interpret the last layer of the encoder as a very downsampled heatmap of features in the original image. They then connected that heatmap to a convolutional layer with a number of channels equal to the number of channels in the segmentation map. This produces a very coarse segmentation map for each class that needs to be segmented.

To be useful, this segmentation map needs to be upsampled to the original image resolution. They achieve this using deconvolution as explained earlier in this chapter. However, as noted earlier, the high-level feature maps have very coarse spatial information. To inject additional spatial information into the final segmentation, FCN combines outputs of multiple encoder layers to form the final prediction. They do this by upsampling outputs of different layers to the same size and summing the feature maps together. The summed feature map is then upsampled to the final image resolution. This process is shown in Figure 2.9. A slightly simpler variant of this architecture is U-Net, which we will look at next.

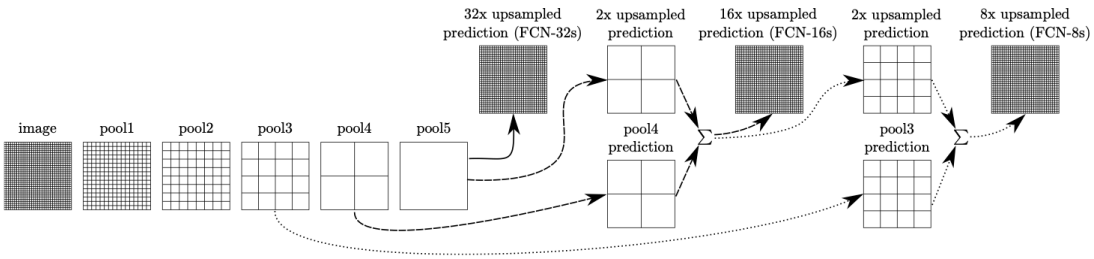

Figure 2.9: A diagram of how FCN forms predictions based on the output of different encoder layers. Encoder layers are shown on the left and the grid represents the coarseness of the feature map. The maps are combined at three different levels to produce three predictions. Each prediction is compared with the ground truth during training, but for inference only the 8x upsampled prediction is used, as it retains the most spatial information. [26]

2.4.2 U-Net and Its Variants

U-Net [27] is one of the most influential and commonly used architectures in medical image segmentation. Aside from being a standard baseline model, a well-tuned U-Net network with data preprocessing is the state of the art for various medical image segmentation tasks including segmenting various organs, cerebral aneurysms, infarctions, edemas, tumors and other tasks on CT and MRI images [28].

U-Net follows a relatively simple architectural pattern. Like other networks, it consists of an encoder and decoder branch. Both the encoder and decoder are fully convolutional, meaning that they consist only of convolutions, pooling layers and transposed convolutions. They have a symmetrical structure where the encoder gradually decreases the width and height of the feature maps while increasing the number of channels, and the decoder does the opposite using upsampling or transposed convolutions.

Earlier we mentioned that encoders gradually reduce spatial information while detecting features in the image. For segmentation, however, the spatial information is important for the precise localization of the features. To overcome this issue, FCN used the feature maps of multiple layers in the encoder. U-Net also takes advantage of feature maps at various levels but does so in a different way.

In U-Net, the output of every encoder layer is added to the input of its corresponding decoder layer on the other side of the network. This gives each of the decoder's layer access to spatial information in the form of feature maps that are the same size as the layer's input. The architecture can be seen in Figure 2.10.

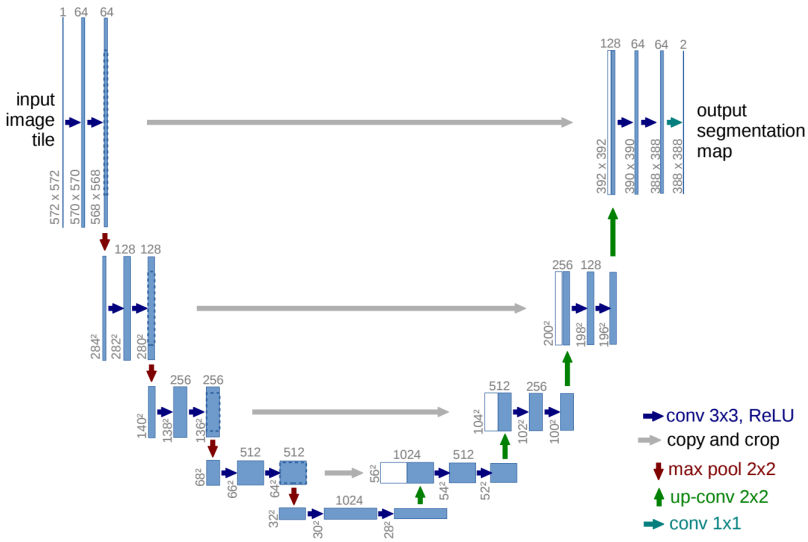

Figure 2.10: A diagram of the U-Net model. The output of each layer of the encoder is concatenated to the input of its corresponding layer in the decoder. [27]

Because of its symmetrical structure and skip connections, U-Net is often visualized in the shape of the letter U, giving it its name. U-Net is only one of a class of networks that are sometimes called U-shaped networks. Aside from being a state-of-the-art architecture for medical image segmentation, U-Net is also widely used as a component of networks in other domains such as image generation [29] and registration [24].

nnU-Net

U-Net is a powerful baseline model that can be carefully tuned using various heretics and combined with beneficial data preprocessing to achieve state-of-the-art results, even when compared to much more complex models. This process has been automated in nnU-Net [28]. nnU-Net is a general method that automatically produces a data pre- and postprocessing pipeline and a U-Net-based architecture to achieve optimal segmentation. This is achieved through a set of heuristics, rules, and optimization techniques.

nnU-Net first calculates various parameters about the data distribution such as the intensity distribution and median image size. These parameters are used as inputs into an interdependent set of rules that produces parameters of the network such as the number of layers, training parameters, and preprocessing strategy, among others. Using these parameters, nnU-Net trains three different networks, one using only 2D slices of the input data, one using the whole 3D volume, and finally, one using only the region of interest in the 3D volume. During inference, the result of each model is preprocessed and combined to form an ensemble prediction. Some of the process can be seen in Figure 2.11.

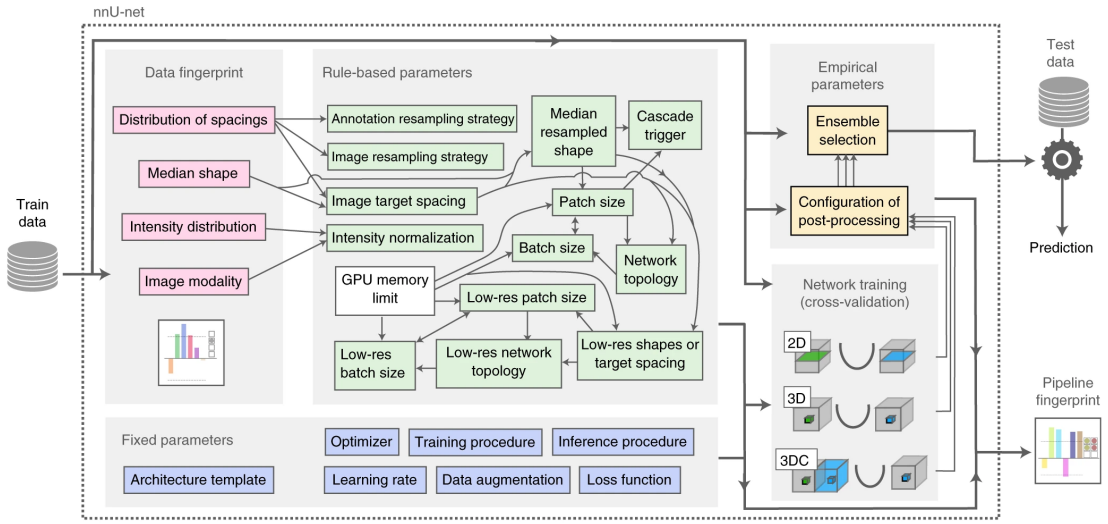

Figure 2.11: A diagram of the nnU-Net procedure of creating a training configuration. [28]

nnU-Net achieved state-of-the-art results in a wide variety of tasks in CT and MRI segmentation and still remains state-of-the-art in many of those tasks. Aside from that, nnU-Net is a hugely important tool for both academia and industry, as it allows for creating well-tuned baseline models with very little manual intervention or technical know-how.

nnU-Net also makes it plainly obvious how much of an impact preprocessing and parameter tuning can have on model results when training on small datasets. This is one of the central premises of this thesis and is what allows us to achieve data efficiency with small neural networks.

U-Net++

U-Net has motivated many variants over the years that are state-of-the-art models for a variety of tasks in medical image segmentation. In this section, we will cover the most seminal of these U-Net modifications, however, there are many more presented in the literature.

In the standard U-Net, the skip connections, i.e. the connections between the outputs of the encoder layers and inputs to the decoder layers, are simple concatenation operations. The outputs are placed to the beginning of the input matrix. However, there is an opportunity here to further combine the two in more complex ways. Furthermore, U-Net only combines features at the same level of encoder and decoder. There is, then, an opportunity to enhance U-Net by giving each decoder layer access to features of multiple encoder layers. U-Net++ [30] takes advantage of both of these opportunities.

In U-Net++, the skip connections are designed as a set of convolutional layers. Each encoder layer is connected to a decoder layer via several convolutional layers. In addition, each encoder layer's output (except for the first one) is upsampled and given to the skip connection of the earlier layer as an additional input to the skip convolutional layers. The outputs of the skip connections themselves are also upscaled and given to the next corresponding decoder layer. What results is a much more complex network where each decoder layer is receives features not just from its corresponding encoder layer but for every encoder layer deeper than the corresponding one. Visually, this is presented in Figure 2.12.

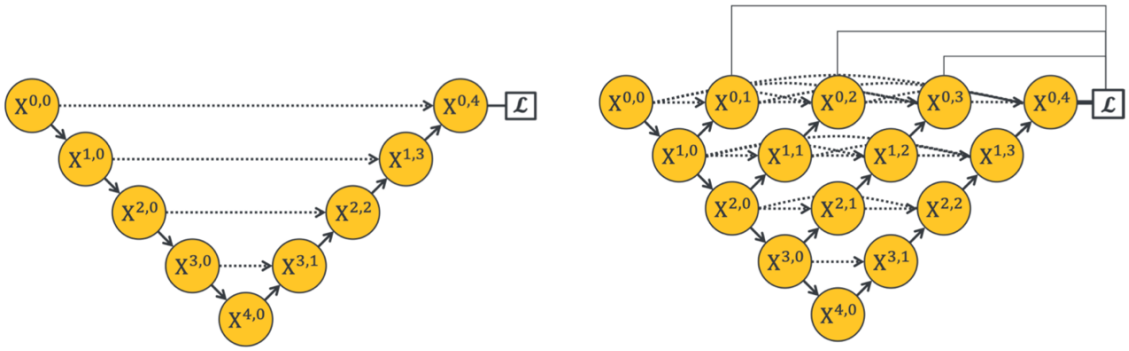

Figure 2.12: A comparison between U-Net (left) and U-Net++ (right). Each node in the graph represents convolutional layers. The dashed arrows represent skip connections, while full arrows are downsampling or upsampling operations. [30]

There are several advantages to U-Net++ over U-Net. Each decoder layer has access to multi-scale features of the input image and thus can produce more accurate outputs. In addition, U-Net++ can be seen as multiple U-Nets of varying depths. This also improves the time it takes the network to converge when compared to a U-Net of similar depth. Further, instead of finding the optimal U-Net depth, due to the flexibility of neural networks, in U-Net++ the optimal network depth is learned via the parameters of the network. U-Net++ results in a more flexible architecture as less design decisions need to be made by hand. However, U-Net++ still has a large parameter count and does not necessarily improve data efficiency. In some cases, the performance advantages achieved by using U-Net++ can be surpassed using a U-net with better data preprocessing and a well-tuned network [28].

2.4.3 Mask R-CNN

Earlier in the chapter we mentioned that CNN-based segmentation networks generally consist of encoder and decoder stages. In segmentation networks, as seen in U-Net and other architectures, the decoder branch progressively upsamples the features map and

builds out a segmentation map using convolutional layers. This differs in object detection networks.

Object detection aims to find the location of an object in an image. Instead of classifying each pixel as in segmentation, object detection usually aims to find a bounding box of the object. Therefore, instead of producing segmentation maps, object detection networks generally produce the coordinates of the object's bounding box. In technical terms, the output is not an image but rather a vector of four numbers for each object. As such, the decoders in object detection networks are usually much shallower and employ fully connected layers instead of convolutional layers.

However, CNN-based object detection and segmentation networks still use the same encoders. Therefore it is possible to combine object detection and segmentation into a single network by using two decoder branches — this is done by Mask R-CNN [31].

In Mask R-CNN, a standard CNN encoder is used to encode a given image into a feature map. This feature map is then given to two separate decoder branches. One branch consists of convolutional layers and upsampling to produce a segmentation map, just like in other segmentation CNNs. The other branch consists of downsampling and fully connected layers in order to produce a bounding box and class label for each object on the image. This means that, for each object, the network outputs a bounding box, a class label as well as a segmentation mask for that object. The network is trained using a combination of segmentation, detection, and classification loss functions such that all three tasks are optimized jointly.

The three tasks have synergy between them, as the information learned for e.g. classifying an object is relevant to its detection (for instance, parrots are more likely to be purple than dogs). Similarly, by using a bounding box the segmentation decoder can only focus on the relevant portion of the image and be invariant to the scale of the object on the image. Jointly learning the three tasks results in better performance on each than training them using separate networks. Mask R-CNN, however, due to its relative complexity, presents challenges when trained on very small datasets compared to simpler segmentation-focused architectures like U-Net.

2.4.4 Other Notable Segmentation CNNs

One of the key challenges of CNN-based segmentation is the tradeoff between spatial information and feature complexity. As the network increases the feature map depth and decreases width and height, spatial information is erased but more complex features can be detection on the image. For segmentation, however, both sources of information are necessary. U-Net solves this problem using skip connections, but there are many other ways to combine features at various spatial scales.

One approach is by using pyramid pooling, as is suggested in PSPNet [32]. PSPNet first employs a standard decoder to derive a feature map. This feature map is then upscaled to various scales, and each scale go through convolutional layers to further decode each scale separately. The outputs of these layers are all upsampled to the same spatial size, concatenated together, and given to a final set of convolutional layers that produces a segmentation map.

Pyramid pooling is also employed by DeepLabV3 [33] albeit in a different way. Like in PSPNet, the encoder first encodes the image into a feature map. This feature map then goes through a series of parallel convolutional layers that each use an atrous convolution. Atrous convolution, also known as dilated convolution, works by dilating the kernel such that there are gaps between the kernel values. During each sliding window step, the gaps are ignored and otherwise a regular convolution procedure is done by multiplying and summing the overlapping values. This allows the kernel to access a larger region

of the image without having to increase the number of parameters in the network. In DeepLabV3, each of the parallel layers has a different number of gap pixels between each kernel element. This produces features at different scales that are all upsampled and concatenated together to be further decoded into a segmentation map.

MA-Net [34] attempts to combine multi-scale features using an attention mechanism. Instead of concatenating the skip connections as U-Net does, MANet introduces a multi-scale attention block. This block aims to capture the interdependencies of different features to focus on more salient ones before they are concatenated to the decoder layer input. In addition, the last decoder layer goes through a position-wise attention block that aims to emphasize features in important locations on the image. MA-Net was applied to the task of liver tumor segmentation in CT images reaching performance improvements over both U-Net and U-Net++.

2.5 Fully Connected Transformers for Medical Image Segmentation

Bibliography

[1] D. Selle, B. Preim, A. Schenk, and H.-O. Peitgen, "Analysis of vasculature for liver surgical planning", *IEEE Transactions on Medical Imaging*, vol. 21, no. 11, pp. 1344–1357, Nov. 2002, ISSN: 0278-0062. DOI: 10.1109/TMI.2002.801166. (visited on 09/27/2023).

[2] S. Devunooru, A. Alsadoon, P. W. C. Chandana, and A. Beg, "Deep learning neural networks for medical image segmentation of brain tumours for diagnosis: A recent review and taxonomy", *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 1, pp. 455–483, Jan. 2021, ISSN: 1868-5137, 1868-5145. DOI: 10.1007/s12652-020-01998-w. (visited on 09/27/2023).

[3] Z. Sobhaninia, S. Rafiei, A. Emami, N. Karimi, K. Najarian, S. Samavi, and S. M. Reza Soroushmehr, "Fetal Ultrasound Image Segmentation for Measuring Biometric Parameters Using Multi-Task Deep Learning", in *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Berlin, Germany: IEEE, Jul. 2019, pp. 6545–6548, ISBN: 978-1-5386-1311-5. DOI: 10.1109/EMBC.2019.8856981. (visited on 09/27/2023).

[4] G. Bastarrika, J. Broncano, U. J. Schoepf, F. Schwarz, Y. S. Lee, J. A. Abro, P. Costello, and P. L. Zwerner, "Relationship Between Coronary Artery Disease and Epicardial Adipose Tissue Quantification at Cardiac CT", *Academic Radiology*, vol. 17, no. 6, pp. 727–734, Jun. 2010, ISSN: 10766332. DOI: 10.1016/j.acra.2010.01.015. (visited on 09/27/2023).

[5] M. Antonelli, A. Reinke, S. Bakas, K. Farahani, A. Kopp-Schneider, B. A. Landman, G. Litjens, B. Menze, O. Ronneberger, R. M. Summers, B. Van Ginneken, M. Bilello, P. Bilic, P. F. Christ, R. K. G. Do, M. J. Gollub, S. H. Heckers, H. Huisman, W. R. Jarnagin, M. K. McHugo, S. Napel, J. S. G. Pernicka, K. Rhode, C. Tobon-Gomez, E. Vorontsov, J. A. Meakin, S. Ourselin, M. Wiesenfarth, P. Arbeláez, B. Bae, S. Chen, L. Daza, J. Feng, B. He, F. Isensee, Y. Ji, F. Jia, I. Kim, K. Maier-Hein, D. Merhof, A. Pai, B. Park, M. Perslev, R. Rezaiifar, O. Rippel, I. Sarasua, W. Shen, J. Son, C. Wachinger, L. Wang, Y. Wang, Y. Xia, D. Xu, Z. Xu, Y. Zheng, A. L. Simpson, L. Maier-Hein, and M. J. Cardoso, "The Medical Segmentation Decathlon", *Nature Communications*, vol. 13, no. 1, p. 4128, Jul. 2022, ISSN: 2041-1723. DOI: 10.1038/s41467-022-30695-9. (visited on 09/28/2023).

[6] C. Edlund, T. R. Jackson, N. Khalid, N. Bevan, T. Dale, A. Dengel, S. Ahmed, J. Trygg, and R. Sjögren, "LIVECell—A large-scale dataset for label-free live cell segmentation", *Nature Methods*, vol. 18, no. 9, pp. 1038–1045, Sep. 2021, ISSN: 1548-7091, 1548-7105. DOI: 10.1038/s41592-021-01249-6. (visited on 09/28/2023).

[7] N. Codella, V. Rotemberg, P. Tschandl, M. E. Celebi, S. Dusza, D. Gutman, B. Helba, A. Kalloo, K. Liopyris, M. Marchetti, H. Kittler, and A. Halpern, *Skin Lesion Analysis Toward Melanoma Detection 2018: A Challenge Hosted by the International Skin Imaging Collaboration (ISIC)*, Mar. 2019. arXiv: 1902.03368 [cs]. (visited on 09/28/2023).

[8] J. Cho, K. Lee, E. Shin, G. Choy, and S. Do, *How much data is needed to train a medical image deep learning system to achieve necessary high accuracy?*, Jan. 2016. arXiv: 1511.06348 [cs]. (visited on 09/28/2023).

[9] E. Lotan, C. Tschider, D. K. Sodickson, A. L. Caplan, M. Bruno, B. Zhang, and Y. W. Lui, "Medical Imaging and Privacy in the Era of Artificial Intelligence: Myth, Fallacy, and the Future", *Journal of the American College of Radiology*, vol. 17, no. 9, pp. 1159–1162, Sep. 2020, ISSN: 15461440. DOI: 10.1016/j.jacr.2020.04.007. (visited on 09/28/2023).

[10] B. Lutnick, B. Ginley, D. Govind, S. D. McGarry, P. S. LaViolette, R. Yacoub, S. Jain, J. E. Tomaszewski, K.-Y. Jen, and P. Sarder, "An integrated iterative annotation technique for easing neural network training in medical image analysis", *Nature Machine Intelligence*, vol. 1, no. 2, pp. 112–119, Feb. 2019, ISSN: 2522-5839. DOI: 10.1038/s42256-019-0018-3. (visited on 09/28/2023).

[11] R. Fosbinder and D. Orth, *Essentials of Radiologic Science*. Wolters Kluwer Health/Lippincott Williams & Wilkins, 2011, ISBN: 978-0-7817-7554-0.

[12] S. Kamalian, M. H. Lev, and R. Gupta, "Computed tomography imaging and angiography – principles", in *Handbook of Clinical Neurology*, vol. 135, Elsevier, 2016, pp. 3–20, ISBN: 978-0-444-53485-9. DOI: 10.1016/B978-0-444-53485-9.00001-5. (visited on 09/29/2023).

[13] A. A. Mahabadi, B. Balcer, I. Dykun, M. Forsting, T. Schlosser, G. Heusch, and T. Rassaf, "Cardiac computed tomography-derived epicardial fat volume and attenuation independently distinguish patients with and without myocardial infarction", *PLOS ONE*, vol. 12, no. 8, M. W. Merx, Ed., e0183514, Aug. 2017, ISSN: 1932-6203. DOI: 10.1371/journal.pone.0183514. (visited on 09/29/2023).

[14] M. Benčević, I. Galić, M. Habijan, and A. Pižurica, "Recent Progress in Epicardial and Pericardial Adipose Tissue Segmentation and Quantification Based on Deep Learning: A Systematic Review", *Applied Sciences*, vol. 12, no. 10, p. 5217, May 2022, ISSN: 2076-3417. DOI: 10.3390/app12105217. (visited on 09/29/2023).

[15] L. Radl, Y. Jin, A. Pepe, J. Li, C. Gsaxner, F.-h. Zhao, and J. Egger, "AVT: Multicenter aortic vessel tree CTA dataset collection with ground truth segmentation masks", *Data in Brief*, vol. 40, p. 107 801, Feb. 2022, ISSN: 23523409. DOI: 10.1016/j.dib.2022.107801. (visited on 10/05/2023).

[16] E. Calabrese, J. E. Villanueva-Meyer, J. D. Rudie, A. M. Rauschecker, U. Baid, S. Bakas, S. Cha, J. T. Mongan, and C. P. Hess, "The University of California San Francisco Preoperative Diffuse Glioma MRI (UCSF-PDGM) Dataset", *Radiology: Artificial Intelligence*, vol. 4, no. 6, e220058, Nov. 2022, ISSN: 2638-6100. DOI: 10.1148/ryai.220058. arXiv: 2109.00356 [cs, eess]. (visited on 10/05/2023).

[17] H. Q. Nguyen, K. Lam, L. T. Le, H. H. Pham, D. Q. Tran, D. B. Nguyen, D. D. Le, C. M. Pham, H. T. T. Tong, D. H. Dinh, C. D. Do, L. T. Doan, C. N. Nguyen, B. T. Nguyen, Q. V. Nguyen, A. D. Hoang, H. N. Phan, A. T. Nguyen, P. H. Ho, D. T. Ngo, N. T. Nguyen, N. T. Nguyen, M. Dao, and V. Vu, *VinDr-CXR: An open dataset of chest X-rays with radiologist's annotations*, 2020. arXiv: 2012.15029 [eess.IV].

[18] J. Irvin, P. Rajpurkar, M. Ko, Y. Yu, S. Ciurea-Ilcus, C. Chute, H. Marklund, B. Haghgoo, R. Ball, K. Shpanskaya, J. Seekins, D. A. Mong, S. S. Halabi, J. K. Sandberg, R. Jones, D. B. Larson, C. P. Langlotz, B. N. Patel, M. P. Lungren, and A. Y. Ng, *CheXpert: A Large Chest Radiograph Dataset with Uncertainty Labels and Expert Comparison*, Jan. 2019. arXiv: 1901.07031 [cs, eess]. (visited on 10/02/2023).

[19] V. Rotemberg, N. Kurtansky, B. Betz-Stablein, L. Caffery, E. Chousakos, N. Codella, M. Combalia, S. Dusza, P. Guitera, D. Gutman, A. Halpern, B. Helba, H. Kittler, K. Kose, S. Langer, K. Lioprys, J. Malvehy, S. Musthaq, J. Nanda, O. Reiter, G. Shih, A. Stratigos, P. Tschandl, J. Weber, and H. P. Soyer, "A patient-centric dataset of images and metadata for identifying melanomas using clinical context", *Scientific Data*, vol. 8, no. 1, p. 34, Jan. 2021, ISSN: 2052-4463. DOI: 10.1038/s41597-021-00815-z. (visited on 10/02/2023).

[20] A. Jha, H. Yang, R. Deng, M. E. Kapp, A. B. Fogo, and Y. Huo, "Instance segmentation for whole slide imaging: End-to-end or detect-then-segment", *Journal of Medical Imaging*, vol. 8, no. 01, Jan. 2021, ISSN: 2329-4302. DOI: 10.1117/1.JMI.8.1.014001. (visited on 10/02/2023).

[21] R. Adams and L. Bischof, "Seeded region growing", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 6, pp. 641–647, 1994. DOI: 10.1109/34.295913.

[22] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models", *International Journal of Computer Vision*, vol. 1, no. 4, pp. 321–331, Jan. 1988. DOI: 10.1007/bf00133570.

[23] T. Rohlfing, R. Brandt, R. Menzel, D. B. Russakoff, and C. R. Maurer, "Quo vadis, atlas-based segmentation?", in *Handbook of Biomedical Image Analysis*, Springer US, 2005, pp. 435–486. DOI: 10.1007/0-306-48608-3_11.

[24] M. Sinclair, A. Schuh, K. Hahn, K. Petersen, Y. Bai, J. Batten, M. Schaap, and B. Glocker, "Atlas-ISTN: Joint segmentation, registration and atlas construction with image-and-spatial transformer networks", *Medical Image Analysis*, vol. 78, p. 102 383, May 2022, ISSN: 13618415. DOI: 10.1016/j.media.2022.102383. (visited on 10/03/2023).

[25] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition", *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov./1998, ISSN: 00189219. DOI: 10.1109/5.726791. (visited on 10/05/2023).

[26] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.

[27] O. Ronneberger, P. Fischer, and T. Brox, *U-Net: Convolutional Networks for Biomedical Image Segmentation*, May 2015. arXiv: 1505.04597 [cs]. (visited on 10/06/2023).

[28] F. Isensee, P. F. Jaeger, S. A. A. Kohl, J. Petersen, and K. H. Maier-Hein, "nnU-Net: A self-configuring method for deep learning-based biomedical image segmentation", *Nature Methods*, vol. 18, no. 2, pp. 203–211, Feb. 2021, ISSN: 1548-7091, 1548-7105. DOI: 10.1038/s41592-020-01008-z. (visited on 10/06/2023).

[29] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, *High-resolution image synthesis with latent diffusion models*, 2021. arXiv: 2112.10752 [cs.CV].

[30] Z. Zhou, M. M. R. Siddiquee, N. Tajbakhsh, and J. Liang, "UNet++: Redesigning skip connections to exploit multiscale features in image segmentation", *IEEE Transactions on Medical Imaging*, 2019.

[31] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask R-CNN", in *2017 IEEE International Conference on Computer Vision (ICCV)*, Venice: IEEE, Oct. 2017, pp. 2980–2988, ISBN: 978-1-5386-1032-9. DOI: 10.1109/ICCV.2017.322. (visited on 10/09/2023).

[32] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network", in *CVPR*, 2017.

[33] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, *Rethinking atrous convolution for semantic image segmentation*, 2017. arXiv: 1706.05587 [cs.CV].

[34] T. Fan, G. Wang, Y. Li, and H. Wang, "MA-Net: A multi-scale attention network for liver and tumor segmentation", *IEEE access : practical innovations, open solutions*, vol. 8, pp. 179 656–179 665, 2020. DOI: 10.1109/ACCESS.2020.3025372.